BAYESIAN BLOCKS: APPLICATIONS IN BIG DATA AND SEGMENTATION

A Research Project
Presented to
Dr. Jeff Scargle
Dr. Brad Jackson
Dr. Slobodan Simić
San Jóse State University
by
CAMCOS Spring 2017 Students:
Tianyang Chen, Ying Gong, Zachary Duffy, Qiong Teng, Alan Ghazarians, Scott Lamkin, Yu Jung Yeh, Jun Wang, Ryan Shiroma, Stefanie Deo, Kristel Lenk, Mai Uyen Le, Ya-Ling Yao, Hongzhe Liu, Sucharu Gupta, Lu Liu, Fangzhou Shen, Lingjun Wang, Jiaxing Ren
May 2017

Abstract

Motivated by previous CAMCOS work and Dr. Scargle's research goals, we investigate new applications and theoretical extensions to Bayesian Blocks. We first compare different algorithms for finding change-points. Next, we explore applications for Bayesian Blocks, such as coverage and densities of Asda stores in UK, analysis of three-dimensional astronomical data and analysis of GC-rich regions of a DNA sequence. Then, we introduce new block shapes in addition to the existing constant blocks: linear, exponential and power function intensities. We also discuss the theory and applications of real-time change-point detection.

Acknowledgments

We would like to thank Dr. Slobodan Simić, the CAMCOS director for making this project possible. Our special thanks goes to Dr. Jeffrey Scargle, a research astrophysicist at NASA Ames Research Center, and to Dr. Bradley Jackson, our project advisor and a math professor at SJSU. We thank you for your guidance throughout this semester and making this project a memorable experience.

Contents

Chapter 1

Introduction

With a record number of students in this CAMCOS project, we were able to explore the applications and capabilities of Bayesian Blocks from many different angles. New applications developed in this report include finding the GC-rich regions of the DNA sequence and the analysis of two and three-dimensional astronomical data. Both of these applications fall under segmentation in Big Data, which is becoming increasingly relevant with ever-increasing computing power. This also led us to focus on optimizing the speed of calculations, such as implementing the PELT algorithm. Another area of research presented here is extending the Bayesian Blocks algorithm by allowing not only constant blocks, but also linear, exponential and power functions. This allows the algorithm to capture patterns in data, such as FREDs, more accurately. We also developed a method for finding change-points in real time. This can be applied to gamma ray burst detection, and other kinds of data, such as identifying changes in stock prices. Even though we made progress in our research on Bayesian Blocks, we have only scraped the surface of what is possible. At the end of each topic, we highlight some of the next steps that stem from our current findings. It is our hope that the next CAMCOS team will further advance the research that is presented in this report.

Chapter 2

Efficient Algorithms for Finding Optimal Partitions of Astronomical Data

Tianyang Chen, Ying Gong and Zachary Duffy

2.1 Introduction

We are trying to focus on improving the efficiency of the search algorithms for finding density of stars in the universe for this project. This paper will include descriptions of the three method being used in improving the efficiency of the search algorithms. The three methods are binary segmentation, dynamic programming and PELT(Pruned Exact Linear Time) algorithm. Comparisons of the three methods along with the data being used will also be introduced in this paper.

There are two goals we are trying to achieve in this project. The first one is finding optimal partitions of a data set on an interval of connected blocks, the second one is maximizing the likelihood function of each block.

Before getting into the three methods for change-points detection methods, we should

first understand what Change-points are and how to detect them. A change-point is an item in time where the statistical properties before and after this time point differ. Figure 1 and Figure 2 are two examples of change-point detections.

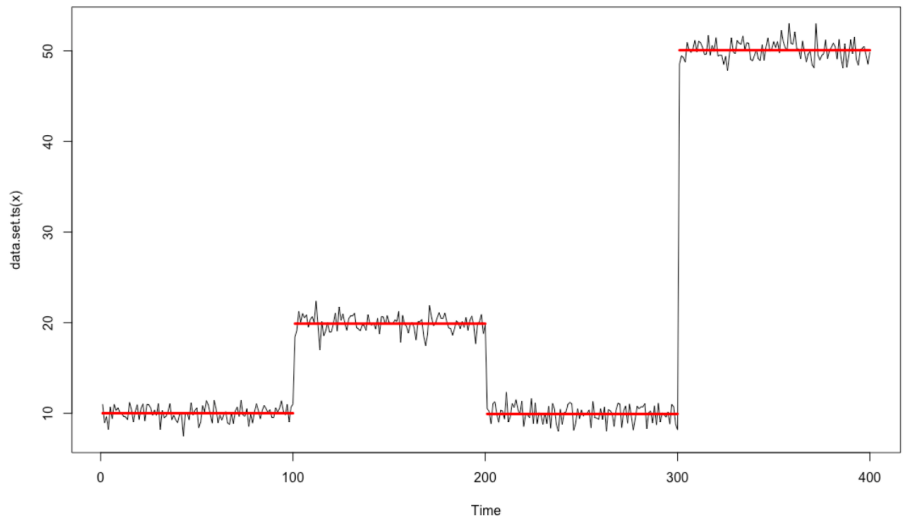


Figure 2.1: Change-point detection in the change in mean

As we can see there are four different means which are 10, 20, 10 and 50 that are shown in red lines. There are three change-points which occur at 100, 200 and 300.

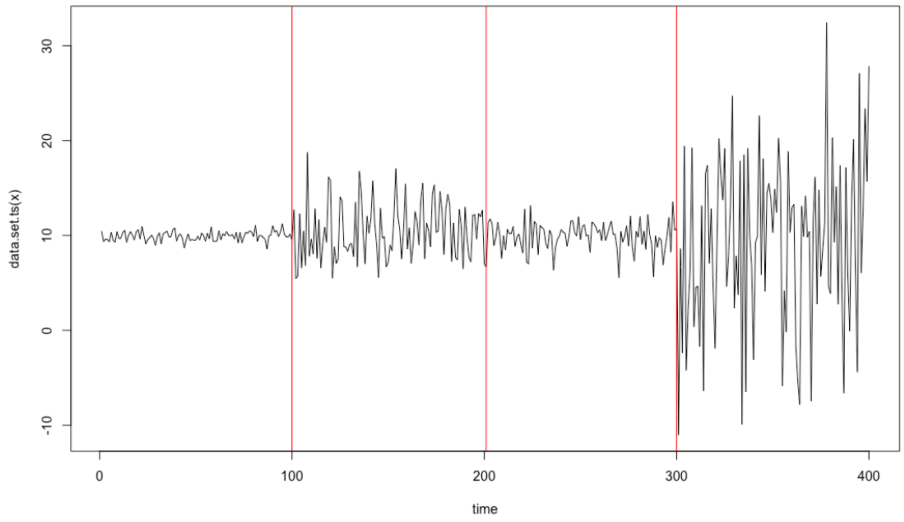


Figure 2.2: Change-point detection in the change in variance

As we can see there are four different variances which separated into four different panels by red lines. There are three change-points which occur at 100, 200 and 300.

2.2 Change-points Detection Method: Binary Segmentation

Binary Segmentation is fast but it does not guarantee to find an optimal solution. This method was developed by Edwards and Cavalli-Sforza [6], Scott and Knott[15] and Sen and Srivastava[**?**]. The idea behind Binary Segmentation is that: we start with a single change-point and then the data is split into two parts if the algorithm detects the change-point. In order to find more change-points, the process is repeated and keep dividing into two parts if change-points are detected in either of the sub-data sets. If no change-points were found in either of the sub-data sets, the process stops.

$$\sum_{i=1}^{m+1} \left[\mathcal{C}(y_{(\tau_{i-1}+1):\tau_i})\right] + \beta f(m) \tag{2.1}$$

The goal of binary segmentation is to minimized Formula (2.1) with f(m) = m. Binary segmentation is an O(nlog(n)) on average because only 2n-1 instances are being considered[11].

2.3 Change-points Detection Method: Dynamic Programming

Dynamic Programming is also called Optimal Partitioning. It is guaranteed to find an optimal partition out of all the partitions in a set with any number of blocks.

Dynamic Programming is an $O(n^2)$ algorithm since it requires $1+2+\ldots+n$ calculations overall. It "automatically determines the model order (the number of segments), has a convenient realtime mode, can be extended to higher dimensional data spaces, and solves a surprising variety of problems in signal detection and characterization, density estimation, cluster analysis and classification"[7].

For future studies, last change theorem can improve the efficiency of the Optimal Partitioning (Dynamic programming) search since it decreases time required to find the new optimal partition. It can find the optimal partition into arbitrary blocks by reducing calculations required. How last change theorem works is that: the last change vector entry is always nondecreasing when we are trying to search for optimal partition of sorted data cells. The last change-point of the optimal for $i+1$ cells are at least as large as the last changepoint of the optimal partition for i cells.

2.4 Change-points Detection Method: PELT

PELT (Pruned Exact Linear Time) algorithm is guaranteed to find an optimal partition. It was proposed by KR. Fearnhead in 2012. PELT is roughly an $O(n)$ algorithm for certain data sets with $O(n)$ scattered change-points. PELT is based on the dynamic programming with best case $O(n)$ and worst case $O(n^2)$ time complexity and it uses

a pruned dynamic programming algorithm for finding optimal partitions of data on an interval in $O(n)$ time for data sets with $O(n)$ changepoints. The efficiency of the PELT algorithm is inversely proportional to the number of changepoints. The point of pruning is to decrease unnecessary computation in the algorithm. The pruning process removes τ value which cannot be done in dynamic programming as seen in Formula (1) .

2.4.1 Methodology

By applying the above theorems to practice, to be clearer about the difference among PELT, Dynamic Programming(DP) method and Binary Segmentation (BS), a range of simulated and real examples are presented with detailed analyzing.

These simulation and experiments answer questions about "how the computational cost of PELT is affected by the amount of data, how to evaluate the computational savings of PELT over DP, and how to evaluate the increased accuracy of exact methods over BS"[11].

Before applying those methods, we first introduce datasets we used. We have two types of datasets. One type is real and complex datasets. This type of datasets came from a real world.

Thus, the number of change-points are not controllable. Other types of datasets are simulated datasets. The benefits of using this type of datasets are easy to control the number of changepoints.

From the above descriptions, we know that the PELT algorithm made by Dynamic Programming(DP) method with an additional pruning step. Therefore, in our experiments of testing the computational cost of the different algorithm, we are not going to including Dynamic Programming(DP) method because we know the worst computational cost of PELT is equal to the computational cost of Dynamic Programming(DP) method.

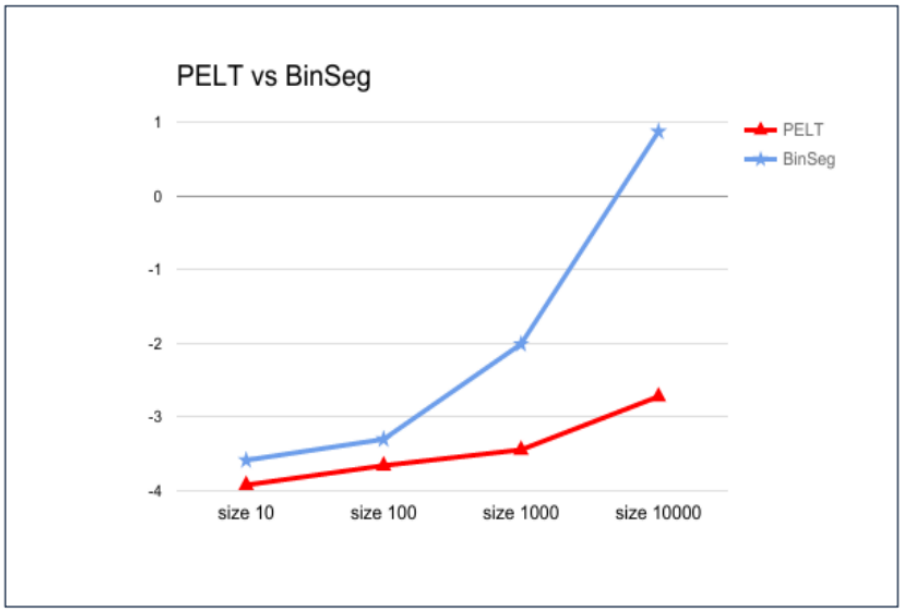


Figure 2.3

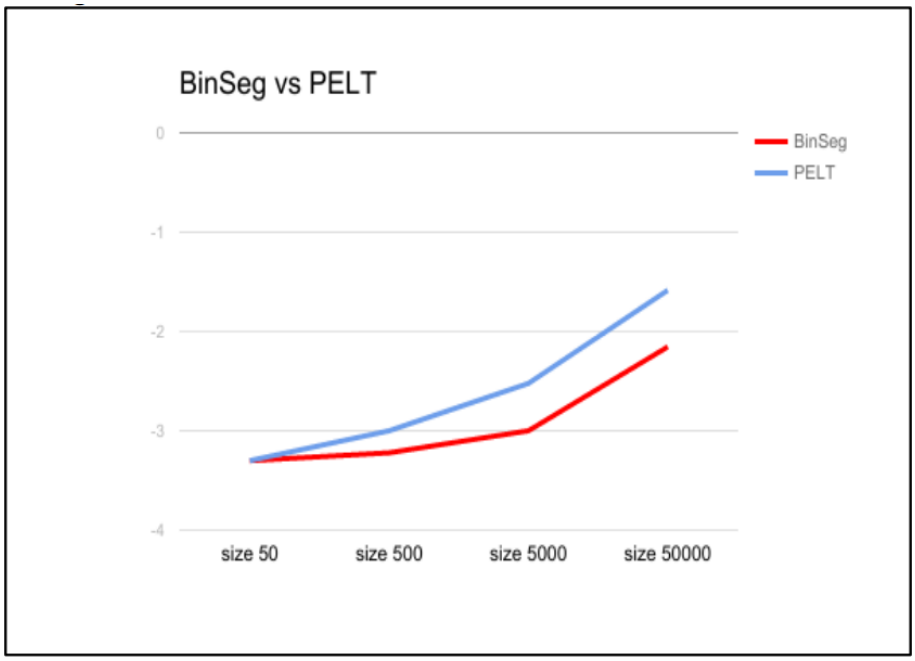


Figure 2.4

From Figure 2.3, as the datasets increasing and high numbers of change-points are observed in datasets, we see PELT run faster than Binary Segmentation. However, as the result shows in Figure 2.4 when datasets increasing and low numbers of change-points are observed in datasets, we see PELT run slower than Binary Segmentation.

Moreover, while we test the run time of these two algorithms, we also find that by increasing the size of the dataset by 10, we expect the number of change-points to increase by 10. This means if we originally have a dataset with a size 150 and 16 change-points, if we increase the size of the dataset to 1500, we should expect to see 160 change-points. As no surprised, When we run this experiment we did observed 159 change-points which is pretty closed to 160 changepoint.

The penalty function also called cost function, is another important factor we need to consider. A penalty function is a function that calculates the cost associated with the true changepoint positions. Our purpose is to minimize the cost.

To this example, a dataset of length 400 (4 segments of normally distributed data sets containing 100 data points each, the means for each segment are 0, 1, 0, 0.2, and constant variance) with multiple change-points at 100, 200, 300 is created.

Before we start to perform an algorithm to find changeponits for any dataset, the first question needs to be answered is "Would there be any change within the dataset?". Furthermore, whether we assume to find out a single change or whether multiple changes are executable. A simple method is processing visual test for the dataset, and if you can find any change in the dataset, the mean of the dataset should have changepoint too.

The complexity of multiple changepoint detections comes from ensuring the optimal number and locating the change-points along the number of solutions increasing rapidly with the size of the dataset. For example, when n = 500, there is 499 possible solutions for a single changepoint detection, for two changes there are 249001 possible solutions! Even worse we do not know how many change-points there, hence we must traverse all the possibility. Searching the large solution space obviously, needs efficient algorithm to

speed up this process.

Any of the three search methods (PELT, DP, BS) could be used to detect these changes. In this example, the PELT and binary segmentation search methods are presented to provide a comparison between exact and alternative algorithms. The dataset is assumed to be independent and normally distributed to set up the baseline for this comparison.

In this case, under the default penalty for this dataset, as Figure 2.3 shown, two change-points for PELT are achieved. By the construction of this dataset, we know that there are three changepoints within the dataset. That means a method is not too sensitive. Therefore, we manually set up a penalty value to increase the number of change-points. Thus, we find the proper number of change-points which is three change-points. The result shows on Figure 2.4. "The optimization of proper penalty remains an open question and typically depends on factors including the size of the dataset and the length of segments, both are unknown before processing analysis. In current practice, the penalty is chosen by plotting the data and change-points to be tested if they look like reasonable"[10]. Since there is not a way to find the optimized proper penalty, we will leave it for future research. As the conclusion for this practice, we know that changing the penalty does affect the sensitivity of changepoint detection.

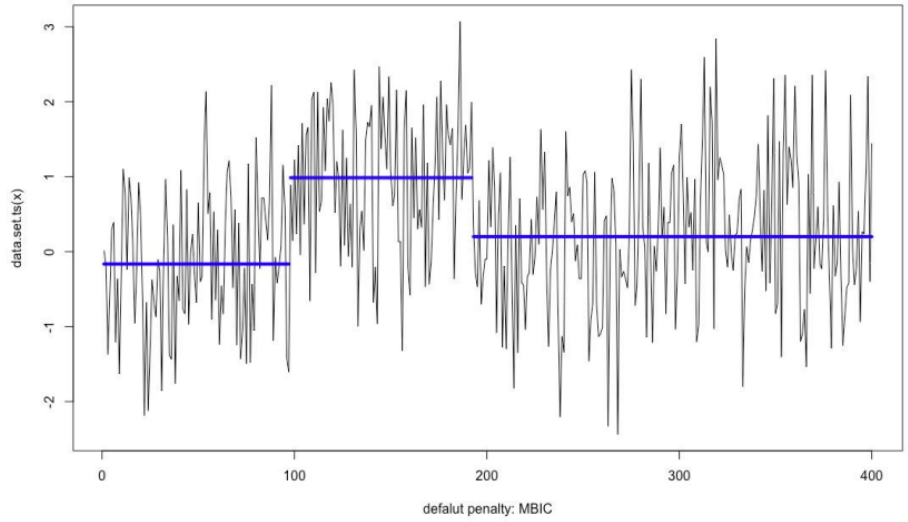


Figure 2.5

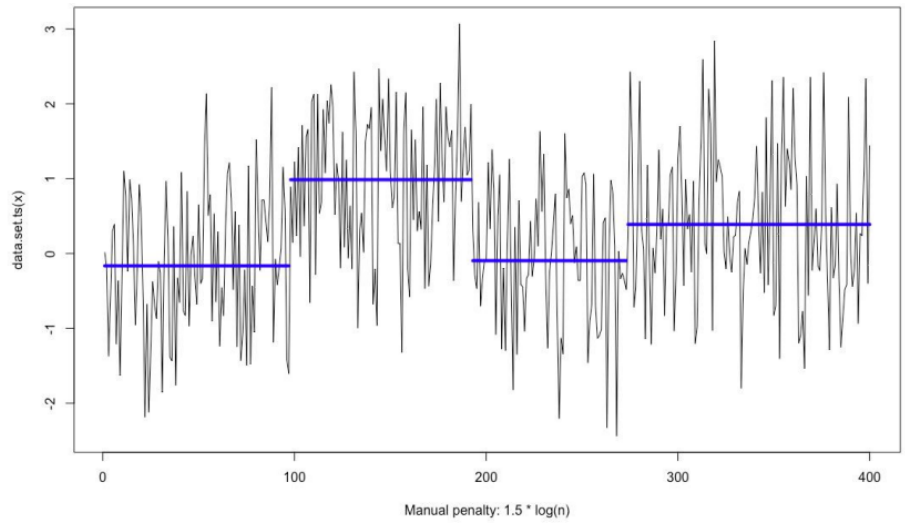


Figure 2.6

All three change-points detection methods have their own applications. Binary segmentation searching method can provide an approximate solution with constant computational time complexity O(log(n)). The dynamic programming method can be used as an initial method for small size dataset with the exact solution and can be extended to

PELT for optimization. PELT with good parameters selection such as the penalty, length of the section, and dataset construction, can be an efficient algorithm for change-points detection. Although the optimization of PELT is difficult, the saving of computational cost is so attractive with an order. Furthermore, some existed package along with R or MatLab can provide a basic PELT algorithm, which makes the PELT can be used more widely and uniformly.

2.4.2 Effects of Changepoints on Run-time

The PELT algorithm is written to remove data positions from consideration as change-points when the difference in cost of different combinations of partitions exceed the penalty value. Altering the penalty constant in any way will affect the number and position of changepoints. In order to understand how changing the number of change-points effects the time to process data through PELT, we have created simple sequences of varying sizes and partitioned them with the PELT algorithm. By comparing the number of detected changepoints to the processing time, we can identify any patterns and draw conclusions in order to establish broad assumptions about the relationship between the number of changepoints and the processing time of the PELT algorithm.

2.4.3 Methods

By creating distinct continuous sequences based on formulas of the form $y = f(x)$, we were able to examine the relationship between the number of changepoints and the processing time of the PELT algorithm. The three sequences generated for the purpose of the timed study were $y = x^2$, $y = 1/x$, and $y = x$ as seen in figure 2.7. PELT was ran with each sequence treated as an individual category, ranging in the number of data points used, and time to process was measured as the independent variable. Aside from the function used to generate each sequence and the size of the data set partitioned in PELT, all other variables in the program were made constant in order to isolate data size as the only independent variable.

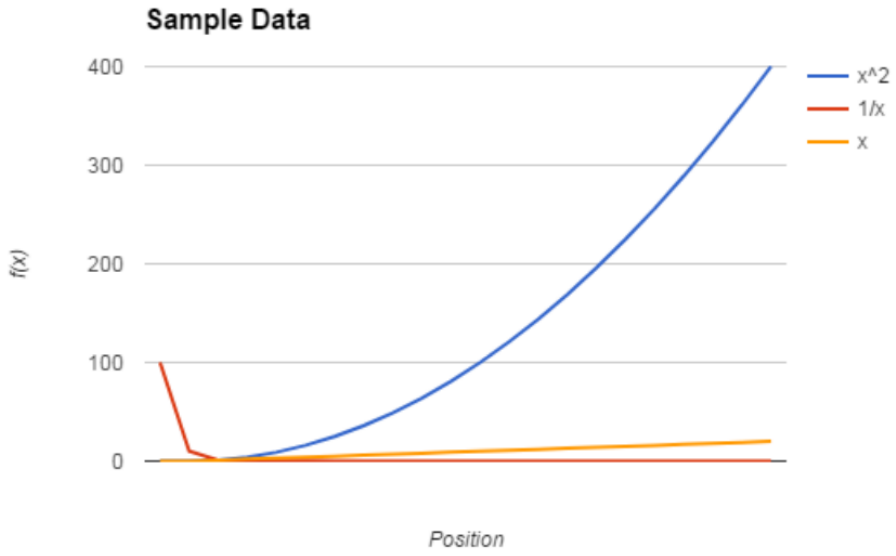


Figure 2.7

2.4.4 Observations

Once the data sets had been established, a 2-dimensional graph was developed in order to visualize the effect of data size on the number of changepoints. The two functions written to increase towards infinity as the number of data points was increased produced the largest number of changepoints. $y = x^2$ with its exponential curve generated the most changepoints at every size we tested, while $y = 1/x$ approached the x-axis asymptotically and produced the least number of changepoints. The only exception to this statement is that $y = 1/x$ produced the most changepoints in the smallest size test, which may be attributed to its asymptotic curve approaching the y-axis from the positive side as seen in figure 2.8.

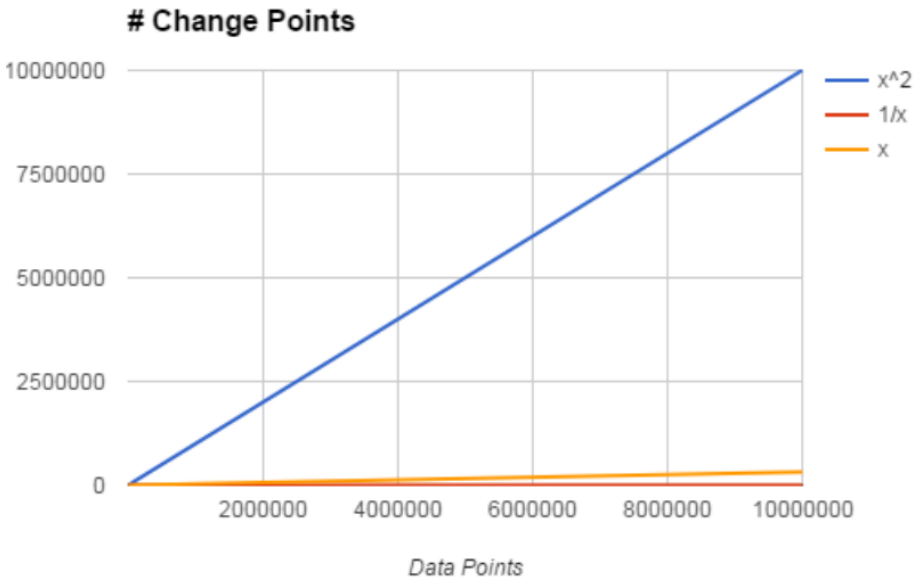


Figure 2.8

When each sequence's processing time is graphed against its respective number of changepoints, we see that an increase in the number of changepoints implies a decrease in the processing time. Figure 2.9 shows that $y = x^2$, with the largest number of changepoints, consistently has the shortest overall run time.

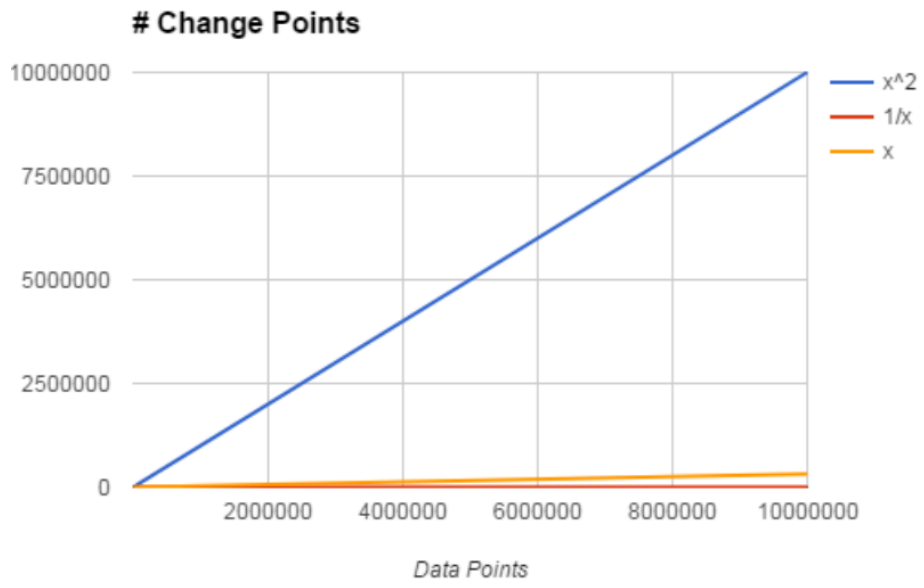


Figure 2.9

2.5 Conclusions

Based on figures 2.8 and 2.9, we conclude that as the number of changepoints in a given data set increase, the processing time of PELT will decrease. With that in mind, the method used in determining the penalty constant will have a severe impact on the number of changepoints and thus the processing time of PELT. Future research employing PELT must be mindful of the fact that several methods to determine a penalty value exist and appropriate research must be conducted in order to preserve the validity of any decisions regarding the penalty value.

Chapter 3

The Coverage and Densities of Asda Stores in U.K.: 2-Dimensional Data Analysis

Qiong Teng

3.1 Introduction

The Voronoi diagram is a fundamental structure in computational geometry and arises naturally in many different fields. In my project, I surveys properties of 2- dimensional Voronoi diagram and its geometric dual, the Delaunay triangulation. The emphasis is on calculating the areas of 2-dimensional Voronoi cells and triangles, and calculation the densities of triangles with Dynamic Programming Algorithm.

In Section 2 and Section 3, definitions of 2-dimensional Voronoi diagram and 2- dimensional Delaunay triangulation and their properties are introduced.

In Section 3, I applied 2-dimensional Voronoi Diagram, 2-dimensional Delaunay Triangulation and Dynamic Programming Algorithm on ASDA store dataset to get the optimal partition, coverage and densities. Besides, some usages of the results and future

research are explored.

In Section 4 - Section 6, I list some applications of 2-dimensional Voronoi diagram and Delaunay triangulation used in different areas, the references, and the MATLAB programs I used in my research.

3.2 2-Dimensional Voronoi diagram and Delaunay Triangulation

3.2.1 2-D Voronoi diagram

A 2 dimensional Voronoi diagram partitions a R^2 space into regions called Voronoi cells, each of which contains one data point and all of the other points in the space are closest to this data point. Voronoi diagrams under two different metrices are shown below (Figure 3.1). Each color region is a Voronoi cell containing exactly one point.

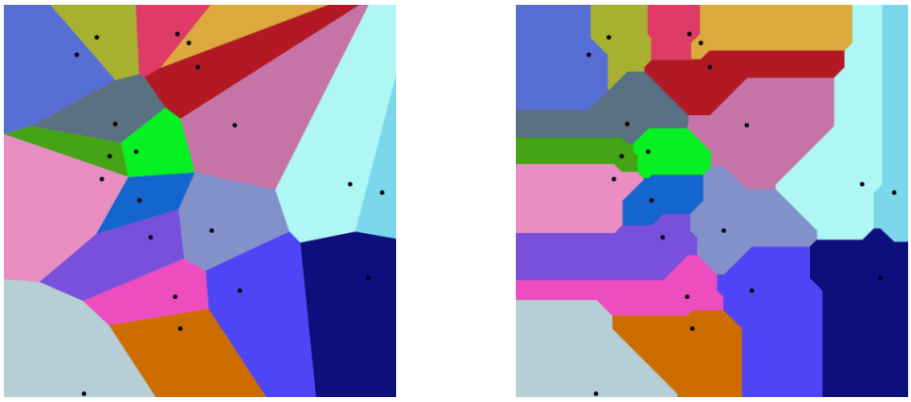


Figure 3.1: Voronoi diagrams with Euclidean distance and Manhattan distance

3.2.2 Definitions

- Let R be a set of distinct points $p_1, p_2, p_3, \ldots, p_n$ in the plane, then the Voronoi diagram of R is a subdivision of the plane into n Voronoi cells such that each cell contains exactly one of these points.

- Every point in a given cell is closer to its generating point than to any other. If a point q lies in the same region as point p_i then the Euclidean distance from p_i to q will be shorter than the Euclidean distance from p_j to q, where p_j is any other point in R which is not p_i.

3.2.3 Glossary

- **Site**: A defining object for a Voronoi diagram or Denaulay triangulation. Also generator, source, Voronoi point.
- **Voronoi cell:** The set of points for which a single site is closest (or more genereally a set of sites is closest). Also Voronoi region, Voronoi face.
- **Voronoi edge:** Every pair of adjacent Voronoi cells are separated by a Voronoi edge which is a subset of points equidistant from the generating points of the two cells.
- **Voronoi vertex:** The points on three (or more) Voronoi edges and is equidistant from three (or more) points in R.
- **Voronoi diagram:**: The set of all Voronoi cells. Also Thiessen diagram, Wigner-Seitz diagram, Blum transform, Dirichlet tessellation

Figure 3.2 is a Voronoi diagram of 11 sites , p_i is one of site, e stands for one of Voronoi edge, and v stands for one of Voronoi vertex. Each Voronoi cell contains exactly one site. q is a free point, which lies in the same region as point p_i . The Euclidean distance from p_i to q will be shorter than the Euclidean distance from q *to other sites*.

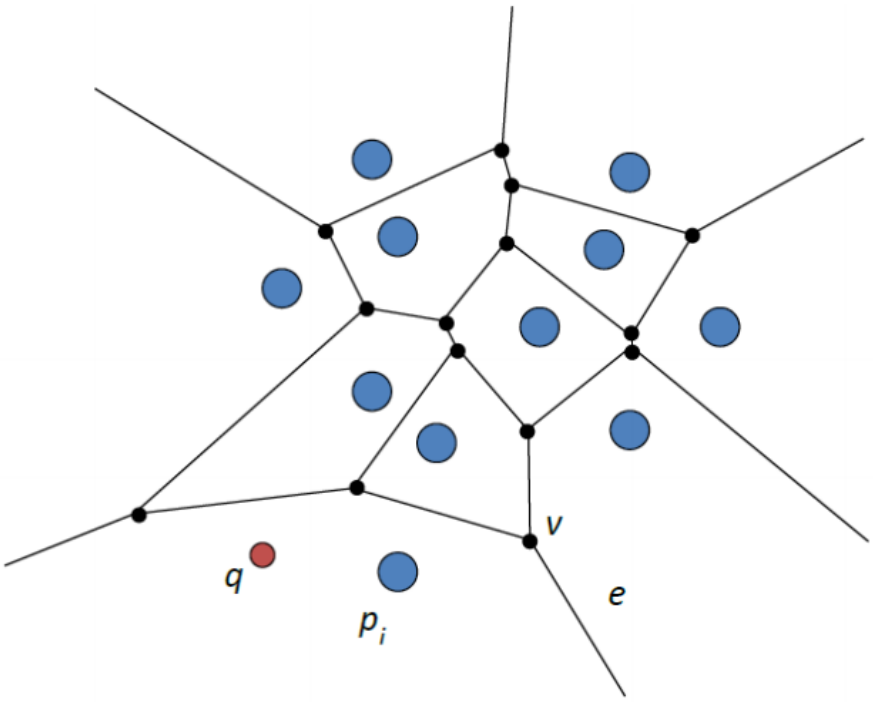


Figure 3.2: Voronoi diagram of 11 sites

3.2.4 Properties

- The dual graph for a Voronoi corresponds to the Delaunay triangulation for the same set of points.
- The closest pair of points corresponds to two adjacent cells in the Voronoi diagram.
- Assume the setting is the Euclidean plane and a group of different points are given. Then two points are adjacent on the convex hull if and only if their Voronoi cells share an infinitely long side.

3.3 2-Dimensional Delaunay triangulation

3.3.1 Definition

A Delaunay triangulation (Figure 3.4) by definition is the unique triangulation of the convex hull of the points in a Voronoi diagram wherein every circumcircle of a triangle is an empty circle (contains no other points). It is used as a method for binning data. It can also be defined as the dual of a Voronoi diagram. Therefore a Delaunay triangulation can be formed by taking the dual of a Voronoi diagram (Figure 3.3). Nodes are sites, and Edges of Delaunay Triangulation are "neighboring" sites in Voronoi diagram

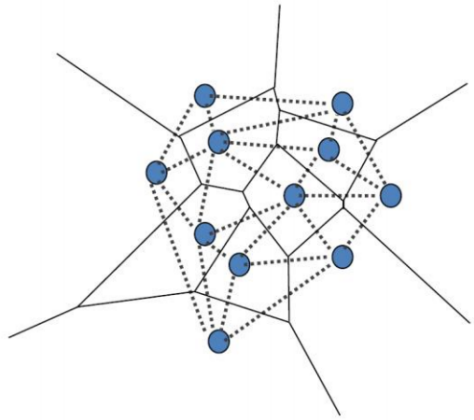


Figure 3.3: Dual graph

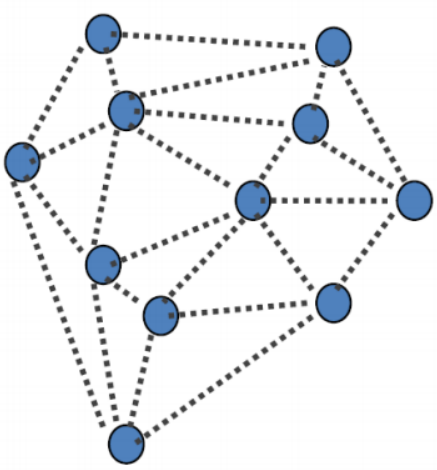


Figure 3.4: Delaunay Triangulation

A Delaunay triangle consists of three data points, each point is assumed to have a 360 degree angle, and a triangle in general has a total of 180 degrees; therefore, a Delaunay triangle contains half a data point. If a Delaunay triangle is considered as a cell, then its nucleus would be a Voronoi vertex. Each Voronoi vertex is adjacent to three Voronoi cells. Since a Voronoi vertex is adjacent to three cells, then it would have three edges connected to it which forms the triangular shape when its dual is taken.

3.3.2 Properties

Let n be the number of points.

- The union of all simplices in the triangulation is the convex hull of the points.
- The Delaunay triangulation contains $O(n)$ simplices.
- If there are b vertices on the convex hull, then any triangulation of the points has at most 2n − 2 − b triangles, plus one exterior face.
- In the plane, each vertex has on average six surrounding triangles.
- In the plane, the Delaunay triangulation maximizes the minimum angle. Compared to any other triangulation of the points, the smallest angle in the Delaunay triangulation is at least as large as the smallest angle in any other. However, the Delaunay triangulation does not necessarily minimize the maximum angle. The Delaunay triangulation also does not necessarily minimize the length of the edges.
- A circle circumscribing any Delaunay triangle does not contain any other input points in its interior.
- If a circle passing through two of the input points doesn't contain any other of them in its interior, then the segment connecting the two points is an edge of a Delaunay triangulation of the given points.

3.4 Application of Voronoi diagram and Delaunay Triangulation

3.4.1 ASDA store data description

Asda Stores Ltd. is a British supermarket retailer, headquartered in Leeds, West Yorkshire. The company was founded in 1965 when the supermarket owning Asquith family merged with the Associated Dairies company of Yorkshire. It expanded in to the south

of England during the 1970s and 1980s. There are in total 631 Asda stores across British. In my project, the dataset contains 545 stores of them. The variables of the dataset are: ID, Latitude, Longitude.

3.4.2 Calculating coverage of Asda store by using Voronoi diagram

One of the purposes is calculating the coverage of Asda store. Voronoi diagram is a good way for partitioning the locations of all stores. There are 3 steps for calculating the ares of Voronoi cells and total areas.

- **Step 1:** Drawing clipped Voronoi diagram The Clipped Voronoi diagram of Asta store (Figure 3.5) shows that each Voronoi cell contains exactly one site, which is perfect for we know the partition of Asda stores. Red dots stand for Asda stores.

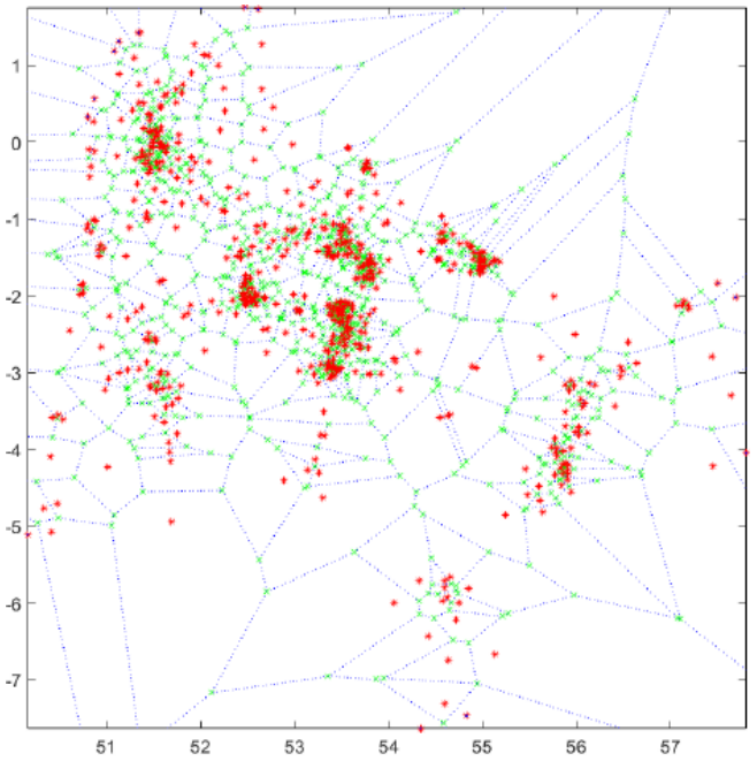


Figure 3.5: Clipped Voronoi diagram of Asda stores

- **Step 2:** Finding vertices of Clipped Voronoi diagram.
  - A Voronoi vertex is an intersection of 3 (or more) segments, each equidistant from a pair of sites
  - A point q is a Voronoi vertex iff its largest empty circle centered at q touches at least 3 (or more) sites.

  In Figure 3.6, q is the Voronoi vertex which is the center of the largest empty circle that touches three sites p_i, p_j and p_k. The green dots in figure 3.5 are Voronoi Vertices of Asda store dataset.

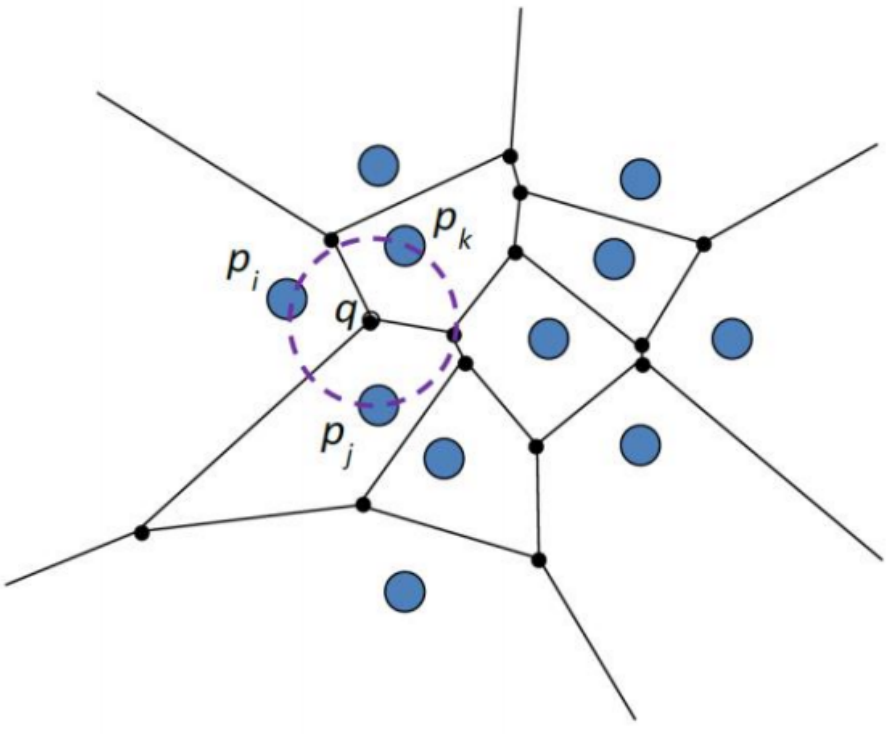


Figure 3.6: q is an example of Voronoi vertex

- **Step 3:** Calculating total coverage of Asda store Since we have already found all the Voronoi Vertices, we could calculate the areas of each cell, which is the coverage

of each Asda store, then calculate the total areas, which is the total coverage of Asda sotes.

Total area of Clipped Voronoi diagram	71.72 units
Accuracy	90%
Mean Absolute Error	5.684e-14

Table 1. Result by using Voronoi Diagram

From Table 1, we can see the accuracy is 90% which is pretty high, and the mean absolute error is5.684e-14, which is pretty low. So the result is precise.

3.5 Calculating the coverage of Asda store by using Delaunay Triangulation

Using Delaunay triangulation to calculate the total coverage of Asda store is another popular way. There are also 3 steps:

- **Step 1:** Draw a Delaunay triangulation(Figure 3.7).

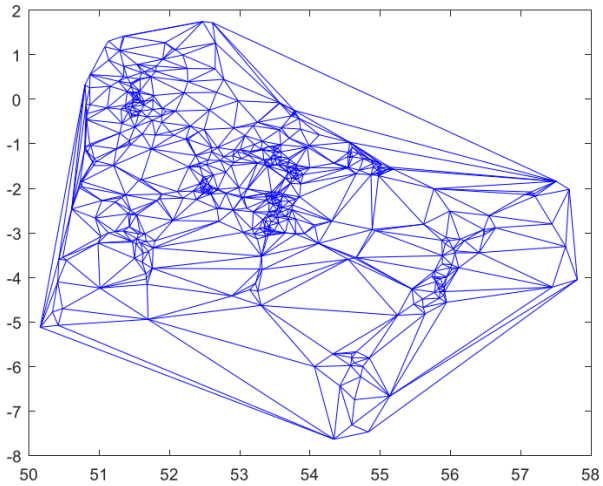


Figure 3.7: Delaunay triangulation using Dynamic Programming Algorithm

- **Step 2:** Finding vertices of triangles. In Delaunay triangulation, the vertices of triangles are the sites.

- **Step 3:** Calculating areas of each triangles (in Figure 7) which is the coverages of each store, and calculating total areas which is the total coverage of Asda store. The total area is 47.57 units.

3.6 Calculating the densities of each Asda store

Dynamic Programming Algorithm is the practical algorithm for finding the optimal partition of the data cells into arbitrary blocks (not necessarily connected). The inputs of Dynamic Programming Algorithm are N, A and c. Table 2 is the detailed description of these inputs.

Input	Description of Input
N	A vector that containing the number of data points in each cell. (.5 for a Delaunay triangulation, and 1 for a Voronoi diagram)
A	A vector that stands for areas of each data cell.
c	Penalty constant which changes the number of blocks in the optimal partition.

Table 2. Inputs of Dynamic Programming

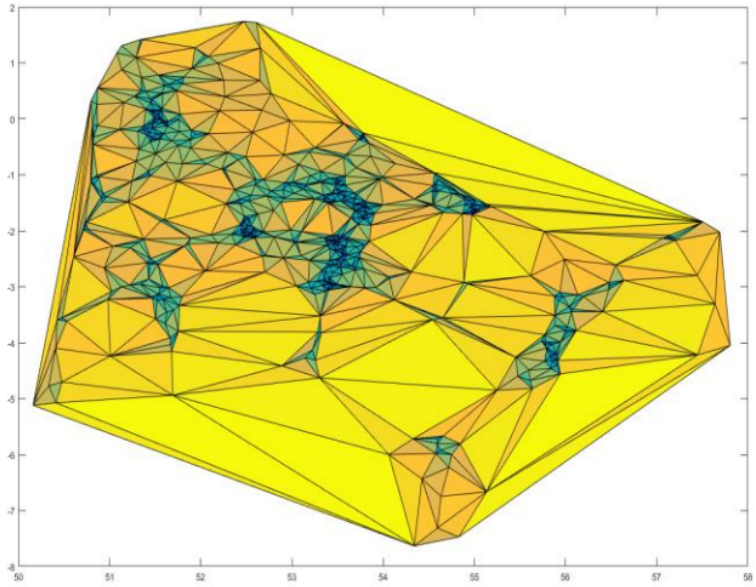


Figure 3.8: Delaunay triangulation using Dynamic Programming Algorithm

In Figure 3.8, different colors for the various blocks represent areas with different levels of densities of Asda stores. Since each triangle contains 0.5 site, the density = 0.5/areas. So the highest density region are the smallest data cells, which are dark blue triangles in Figure 3.8, and the lowest density region are the largest data cells, which are yellow triangles in Figure 3.8.

3.7 Usage of results

In business, the results coverage and densities of Asda store are helpful for calculating the market shares of the Asda Stores with store coverage and density. Besides, they are very useful for location model, because decision makers always consider the store coverage and density seriously with other key parameters, such as transportation distances and costs. And because the results provides the decision maker further insight in the structure of business and lets them better visualize some of the business aspects, such as concerning warehouses and branch offices... Here is an example of other usage of the result. In Figure 3.9, yellow dots represent Asda stores, and pink dot represent post office. If you

would like to go both post office and Asda store, which store is the most convenient to you? GPS will pick the nearest store for you, because the nearest store and the post office are in the same Voronoi cells. So the results can be used to find the nearest neighbor of points in the plane. In Statistics, Voronoi Cells could be used in K-Nearest Neighbour to set the boundary.

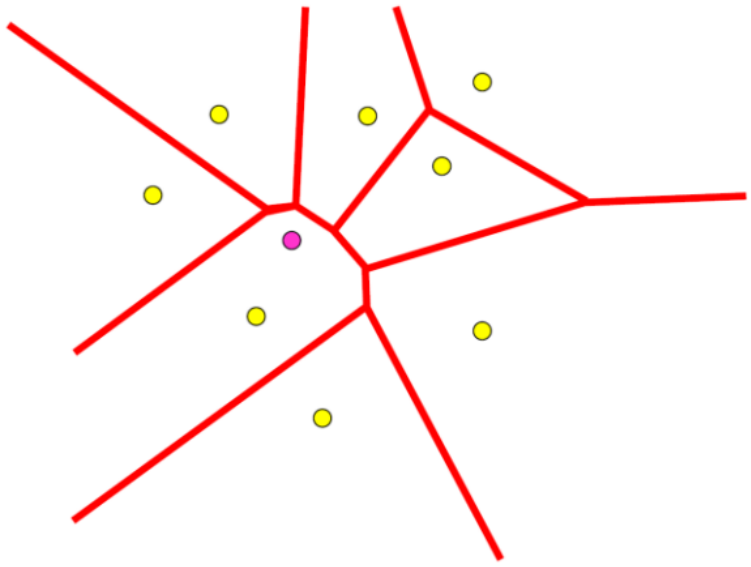


Figure 3.9: An example of using the result for finding the nearest Asda store

3.8 Future direction of research

As we know PELT Algorithm and Lastchange Algorithm are other practical algorithm for finding optimal partition. In the future research, I will

- Apply PELT Algorithm to ASDA data set.
- Apply Lastchange Algorithm to ASDA data.
- Compare the running times of three Algorithm to see which one is the most more quickly and efficient algorithm and how much faster to get the optimal partition.

3.9 Other application of Voronoi Diagram and Delaunay Triangulation

- In biology, they are used to model a number of different biological structures, including cells and bone microarchitecture.
- In ecology, they are used to study the growth patterns of forests and forest canopies, and may also be helpful in developing predictive models for forest fires.
- In astrophysics, Voronoi diagrams are used to generate adaptative smoothing zones on images, adding signal fluxes on each one.
- In medical diagnosis, Voronoi diagrams-based models for muscle tissue can be used to detect neuromuscular diseases.

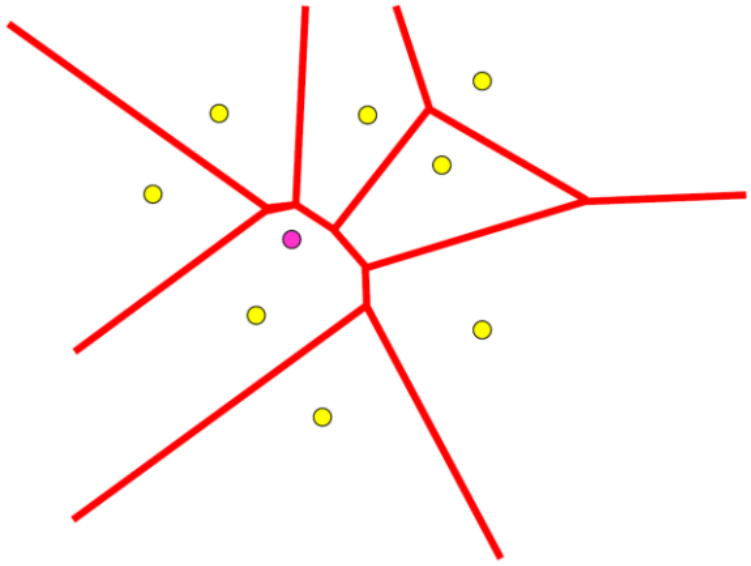


Figure 3.10: Voronoi Map of the USA

- Figure 3.10 is Voronoi Map of the USA. State borders are redrawn based on the locations of the state capitals; the land at any given point is owned by the state with the geographically closest capital city

Chapter 4

Analysis of Three-Dimensional Astronomical Data

Alan Ghazarians and Scott Lamkin

In most of the other sections the algorithms run and data analyzed are one dimensional using a Delaunay triangulation or Voronoi diagram of one dimension. Our motivation is to gauge the viability of implementing these objects in higher dimensions.

4.1 Astronomical Data

In this portion of the project we use the Sloan Digital Sky Survery (SDSS) database, which contains information of over 100,000 galaxies in order to partition these galaxies based on their density. In our project we use a three dimensional extension of the Delaunay triangulation, an object used commonly in mathematics and computational geometry, in order to construct our partition.

4.2 Voronoi Diagram

Before introducing our main geometric object, the Delaunay triangulation, we introduce its dual object, the Voronoi diagram, and its properties. First, we will denote the Euclidian distance between two points x and y in $\mathbb{R}^2$ as

$$d(x, y) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

To construct the Voronoi diagram (in two dimensions) we will consider a set of n distinct data points $V = \{v_1, v_2, \ldots v_n\}$. The Voronoi diagram will be defined as the collection of n cells, which surround a data point v_i such that a point q lies inside the cell of v_i if and only if $d(v_i, q) < d(v_j, q)$ for all other data points $v_j \in V$ such that $i \neq j$. In other words, every point inside a Voronoi cell is cloest to the data point corresponding to that Voronoi cell.

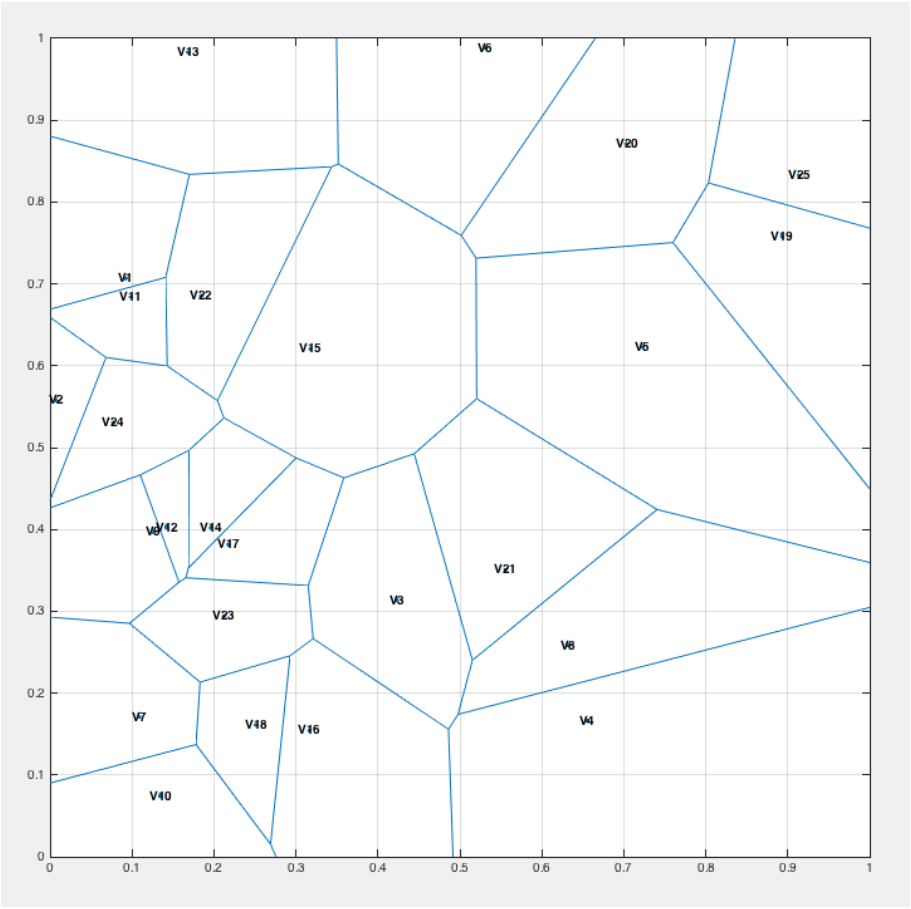


Figure 4.1: An example of a Voronoi diagram with 25 random data points generated by MATLAB.

4.3 Delaunay Triangulation

We now can discuss the mathematical object that our project focuses on: the Delaunay triangulation. As stated before, the Delaunay triangulation is the dual graph to the Voronoi diagram. The data points of the set V are now the vertices of triangles in a Delaunay triangulation if and only if no point in V is in the circumcircle of any triangle in the Delaunay triangulation.

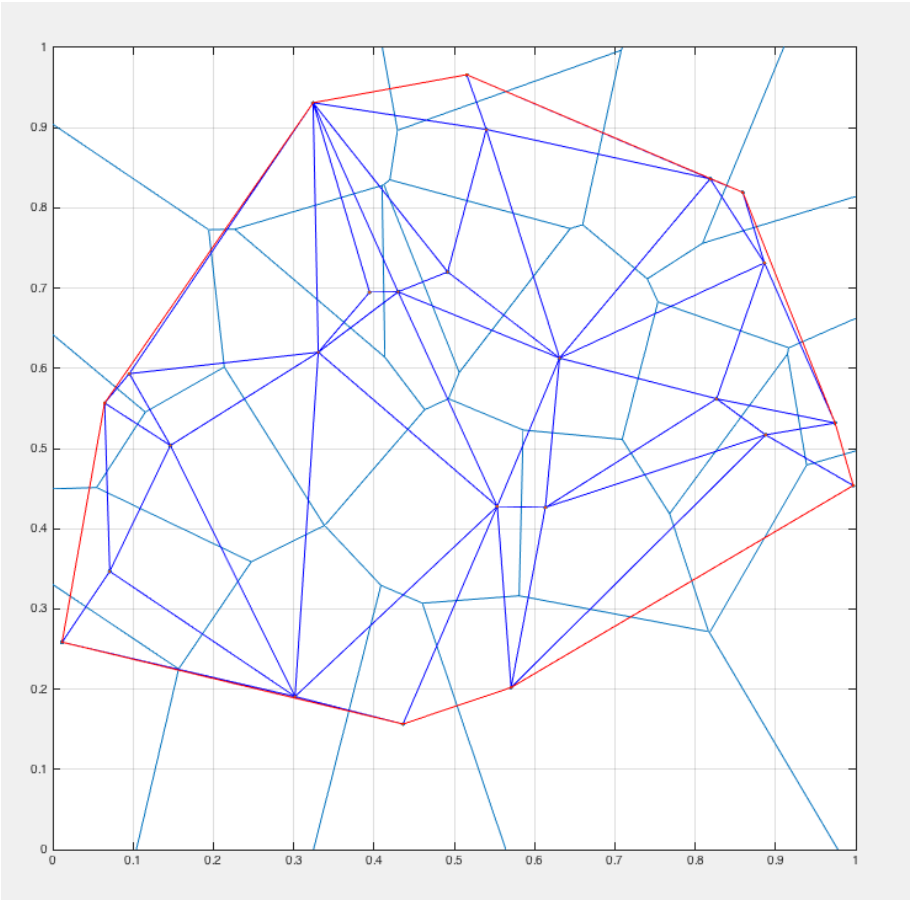


Figure 4.2: An example of a Voronoi diagram and its dual Delaunay triangulation with 25 random data points generated by MATLAB. The convex hull boundary of the Delaunay triangulation is shown in red

When restricting our region to a unit square, there are clear advantages to using a Delaunay triangulation over a Voronoi diagram albeit the fact that they are dual graphs. Comparing these two, we can see that Voronoi diagrams have edges intersecting the boundary and extending infinitely outward while Delaunay triangulations are always within our bounding region (as along as our points are within the region as well) (Figure 4.3).

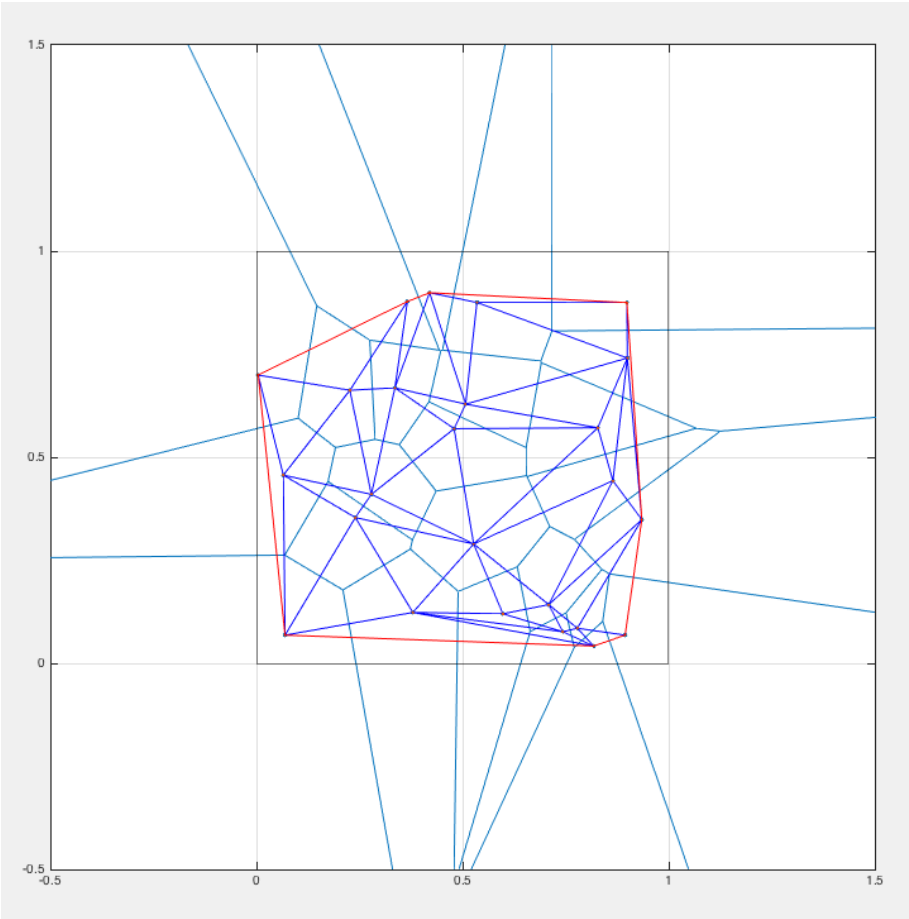


Figure 4.3: An example of a Voronoi diagram and its dual Delaunay triangulation bounded by a unit square generated by MATLAB.

These infinite edges cause "incomplete" Voronoi cells whose areas are considered to be infinite. In other words, when using a Delaunay triangulation we are working with a finite object while a Voronoi diagram is an infinite object. That being said, there are techniques to dealing with the infinite nature of a Voronoi diagram by "clipping" the infinite edges by a boundary. This new object is called a "clipped Voronoi diagram," but we will not be using this object. One disadvantage to using a Delaunay triangulation is associating a point count to the triangle. When using a Voronoi diagram, each Voronoi cell is given a point count of 1 because there is one data point inside each cell. Comparing

that to a Delaunay triangulation where the data points are the vertices of the triangles. This is easily taken care of in two dimensions, but becomes more complicated in three dimensions as we will discuss later.

4.4 Delaunay Tetrahedralization

In this section, we will introduce a three-dimensional extension of the Delaunay triangulation. A Delaunay tetrahedralization will use a set of data points $W = \{w_1, w_2, \ldots w_n\}$ which are the vertices of tetrahedrons in $\mathbb{R}^3$ instead of triangles in $\mathbb{R}^2$. More formally, the data points of the set W are vertices of tetrahedrons in a Delaunay tetrahedralization if and only if no point in W is in the circumsphere of any tetrahedron in the Delaunay tetrahedralization. The Delaunay tetrahedralization shares the same properties and features as the Delaunay triangulation such as the convex hull is contained inside the bounding region.

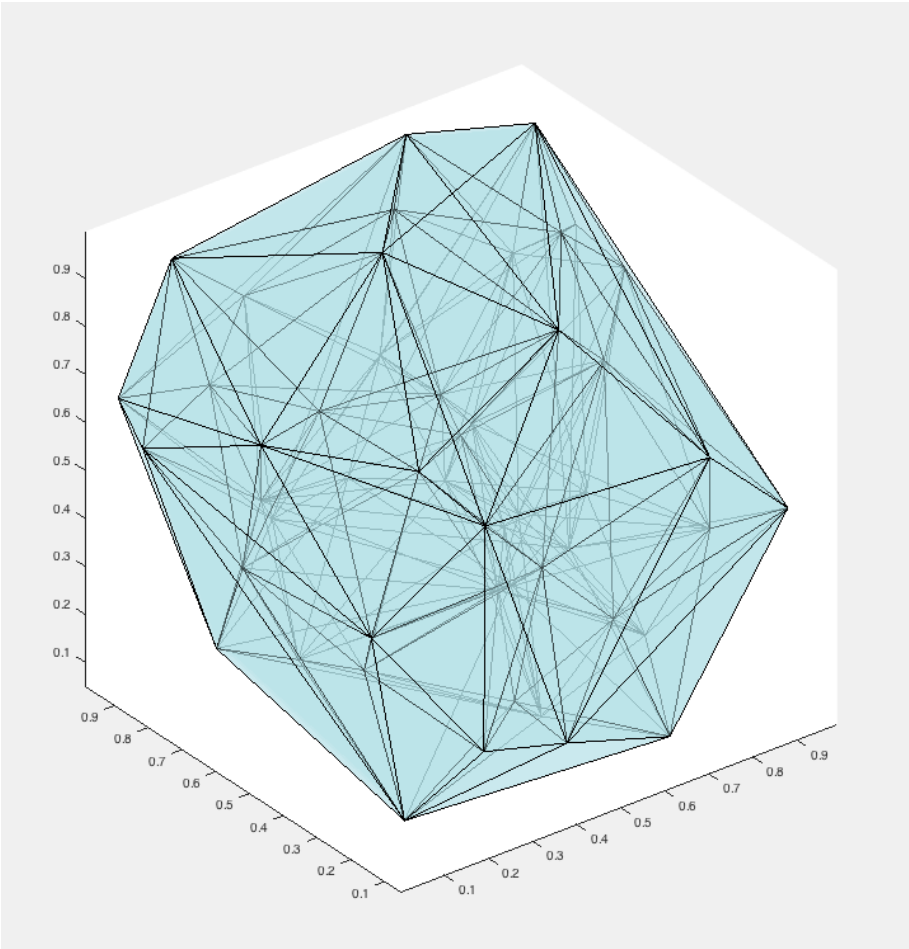


Figure 4.4: An example of a Delaunay tetrahedralization with 50 random points generated by MATLAB.

4.4.1 Volumes, Solid Angles, and Density

In order to go forward with our project, we used MATLAB in order to calculate volumes, solid angles, and densities of our tetrahedrons. With our goal of calculating the density for every tetrahedra in our set, we needed to calculate the volume of every tetrahedra. This lends us to the natural connection of giving each tetrahedra a volume and a mass. The former was simply computed using the convex hull function in MATLAB. However, attributing a certain mass to each galaxy was not feasible for us. Therefore, for simplicity,

in this project we treated every galaxy as having the same mass. Furthermore, since every tetrahedra intersects four of our data points at three dimensional angles, we used solid angles to split up our point count among the relevant tetrahedra.

In the two dimensional case, the Delaunay Triangulation intersects the points as well, but the fix is much simpler. Since every circle around a point has 2π radians and every triangle has π radians, every triangle is attributed half of a point. The three dimensional case becomes more complicated since the intersection at the vertex of any tetrahedra are in three dimensions. This is the motivation for using solid angles to appropriately divide the points among the many tetrahedra that may be intersecting any given point.

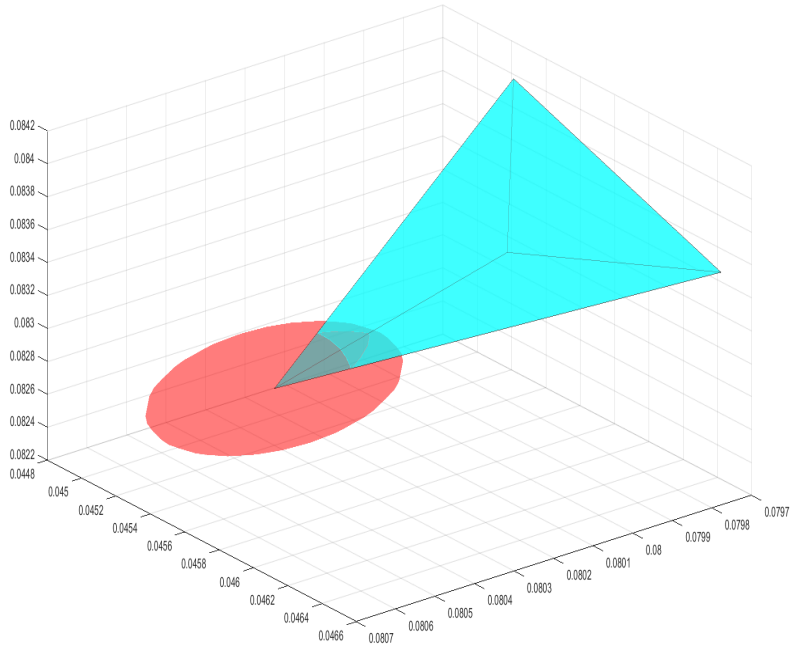


Figure 4.5: An example of a single solid angle being measured.

The solid angle is measured out of 4π and is the proportion of surface area lost on the sphere, due to the intersecting tetrahedron. Note that any single solid angle cannot exceed 2π. The solid angle is measured in steradians, which is a unitless measurement

since it is a ratio. Less formally, for our example, the red sphere in Figure 4.5 can be thought of as an extension of the point that is being intersected by the multiple tetrahedra. Therefore, we can measure the sum of all four solid angles in each tetrahedra and divide by 4π to allocate the correct number of points to each tetrahedra. This is the method we used to assign points to the tetrahedra in our tetrahedralization.

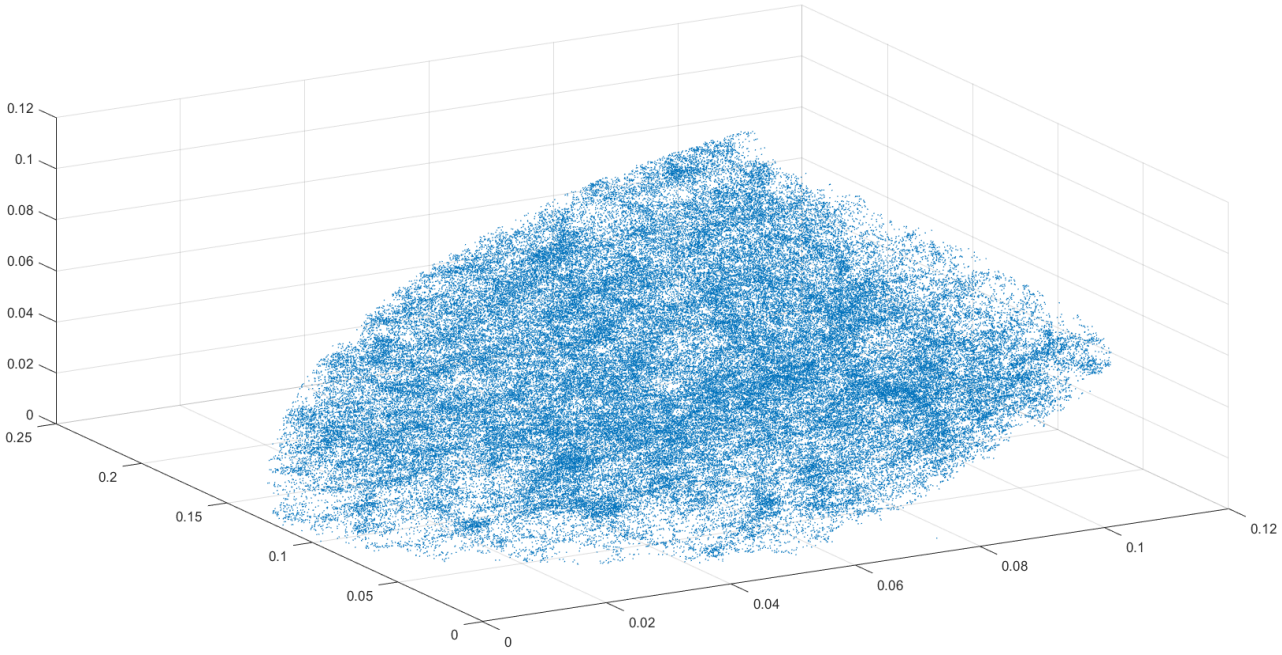


Figure 4.6: Sloan Digital Sky Survey Database

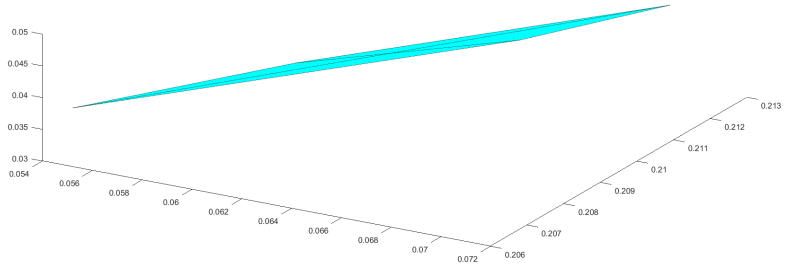
4.5 Tetrahedralization of Sloan Digital Sky Survey Database

The Sloan Digital Sky Survey Database provided by Dr. Scargle contained 139,798 unique galaxies. The program we wrote in MATLAB takes in unique three dimensional cartesian coordinates of data points and creates a Delaunay Tetrahedralization from

those points using the built in MATLAB function. Our tetrahedralization created a total of 915,131 tetrahedra splitting our view of this portion of the universe into tetrahedra. From there our goal was to create a density mapping using the tetrahedra as constant blocks containing a density value.

4.5.1 Problems in MATLAB and a Pathological Example

The tetrahedralization of the SDSS database was created using the inbuilt MATLAB function. As such we would need to spend more time stress testing the function and ensuring that the function is creating an ideal tetrahedralization. The motivation for stress testing our function is best exemplified in the poor behavior of one specific tetrahedron, number 527,105 in our unsorted set. This pathological tetrahedron has a point count of 8.22624781699110e-06 and a volume of 1.24567476978586e-11.



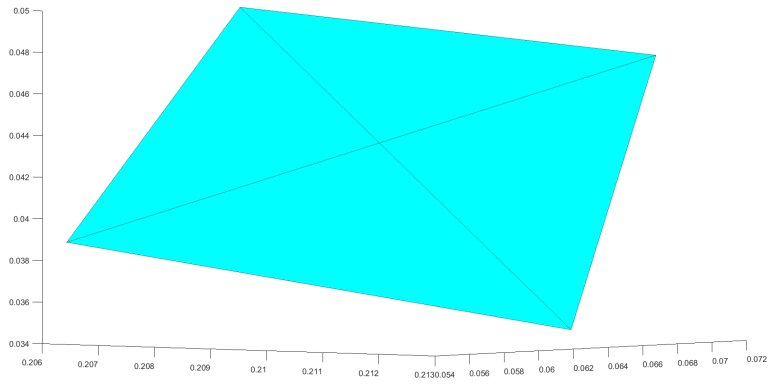
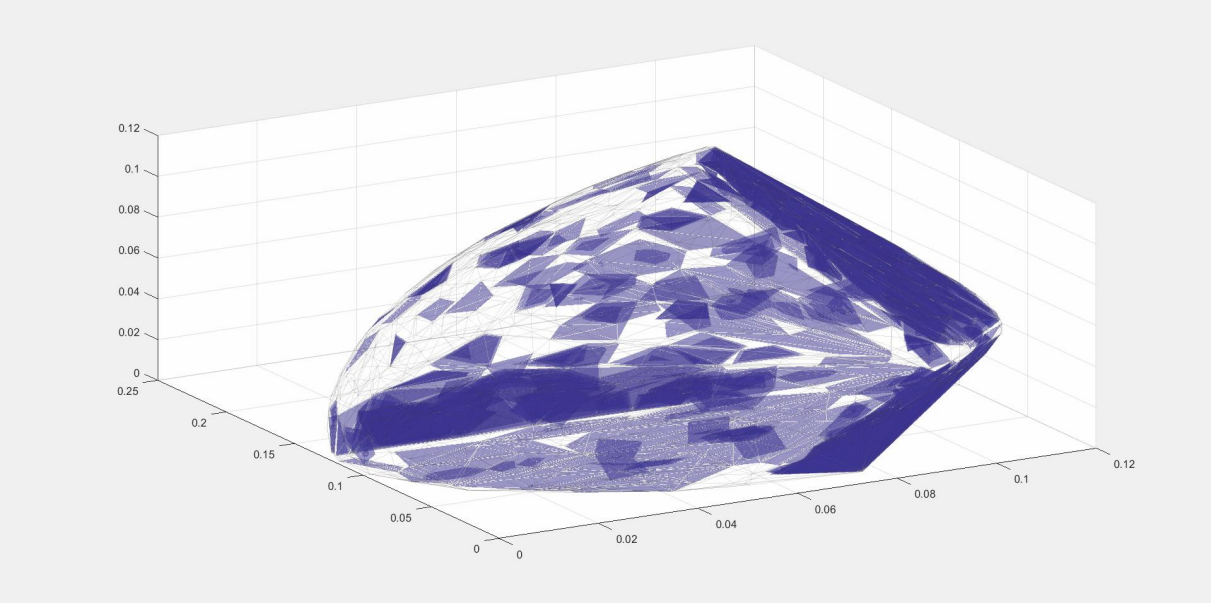


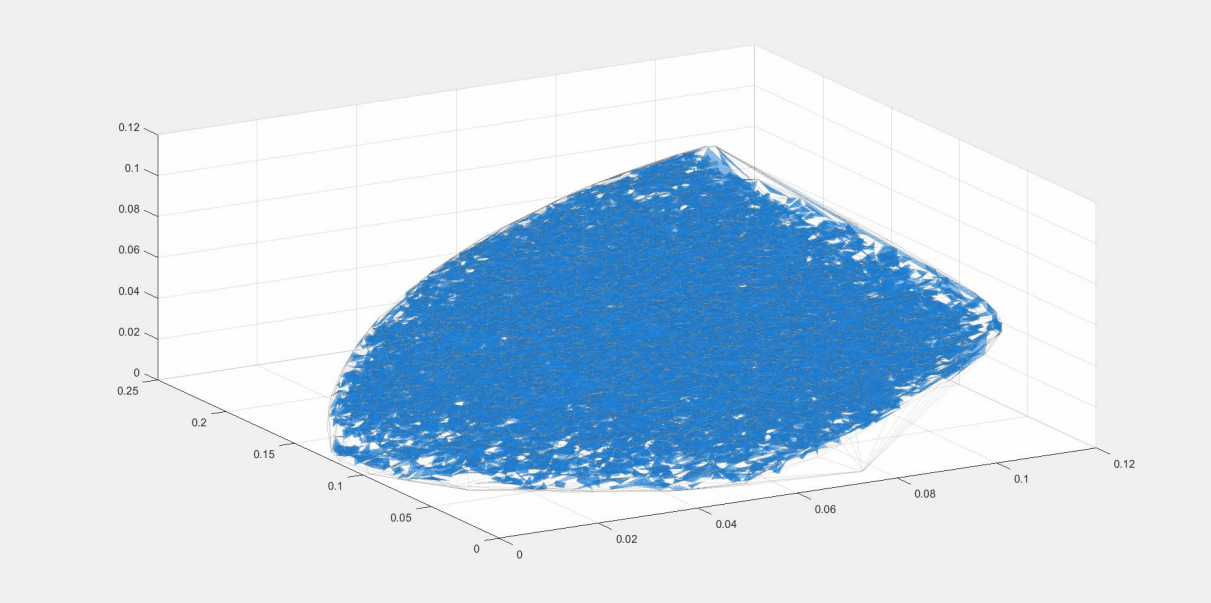
Figure 4.7: Pathological example, same tetrahedron

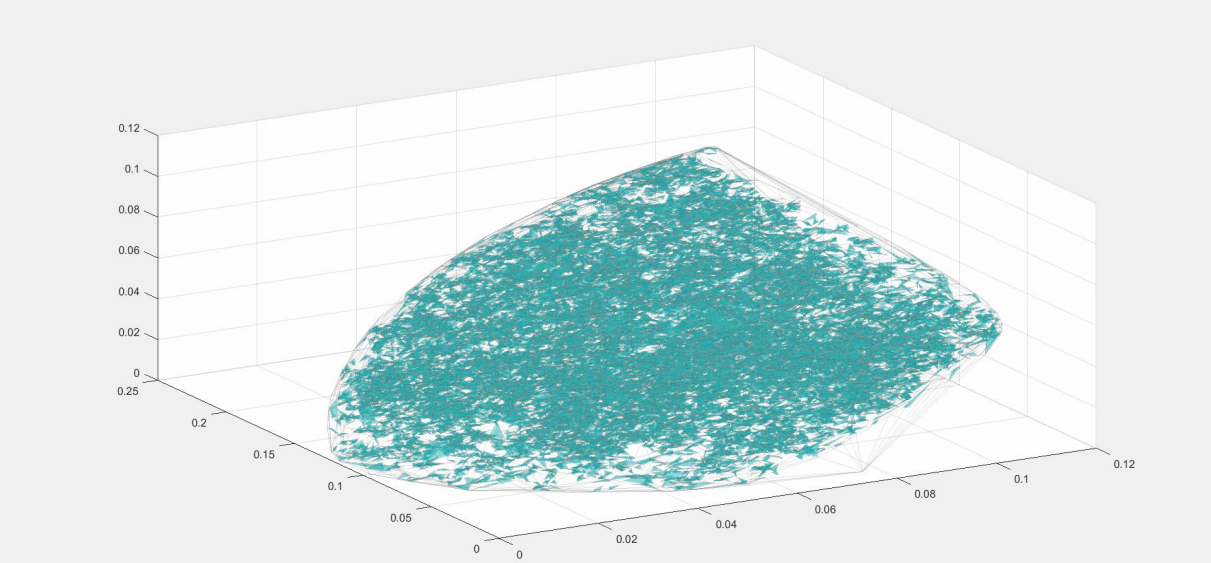
The reason this example is so important is the motivation for using Delaunay Triangulations in the first place. The idea is to avoid as many sliver triangles as possible and maximize the minimum angle. The natural extension of this idea in three dimensions would be to hope to maximize the minimum solid angle of all tetrahedrons in the tetrahedralization. In this respect, our tetrahedralization does not accomplish this. In our tetrahedralization there are 184 tetrahedra with less than a thousandth of a point attributed to them with our pathological having the least amount of points attributed. Since the point attribution is directly calculated from the sum of the solid angles of any given tetrahedron, we can say these are relatively poorly behaved tetrahedra in our tetrahedralization.

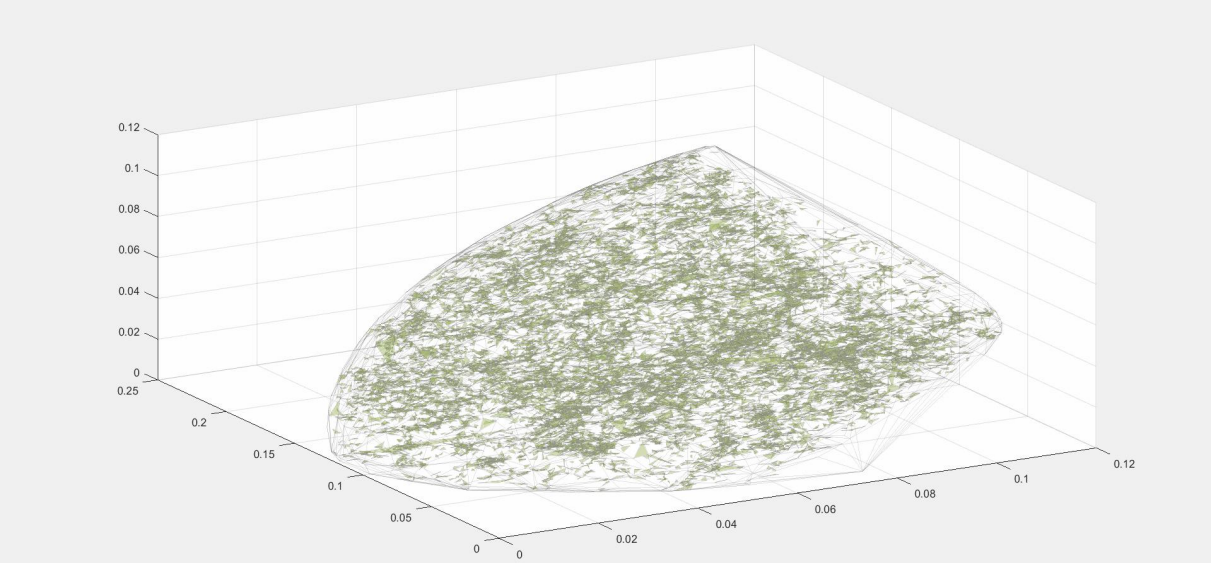
4.5.2 Results

For sorting our tetrahedra by density and allowing us to compare the densities of tetrahedra against each other we used an implementation of PELT written by Dr. Jackson. In our set we treated the tetrahedra as unconnected blocks and used the PELT algorithm to group similar densities. We created a .gif file showing the ten sets of tetrahedra at a time of increasing densities.


0.12
0.1
0.08
0.06
0.04
0.02
0
0.25
0.2
0.15
0.1
0.05
0
0
0.02
0.04
0.06
0.08
0.1
0.12


0.12
0.1
0.08
0.06
0.04
0.02
0
0.25
0.2
0.15
0.1
0.05
0
0
0.02
0.04
0.06
0.08
0.1
0.12


0.12
0.1
0.08
0.06
0.04
0.02
0
0.25
0.2
0.15
0.1
0.05
0
0
0.02
0.04
0.06
0.08
0.1
0.12


0.12
0.1
0.08
0.06
0.04
0.02
0
0.25
0.2
0.15
0.1
0.05
0
0
0.02
0.04
0.06
0.08
0.1
0.12

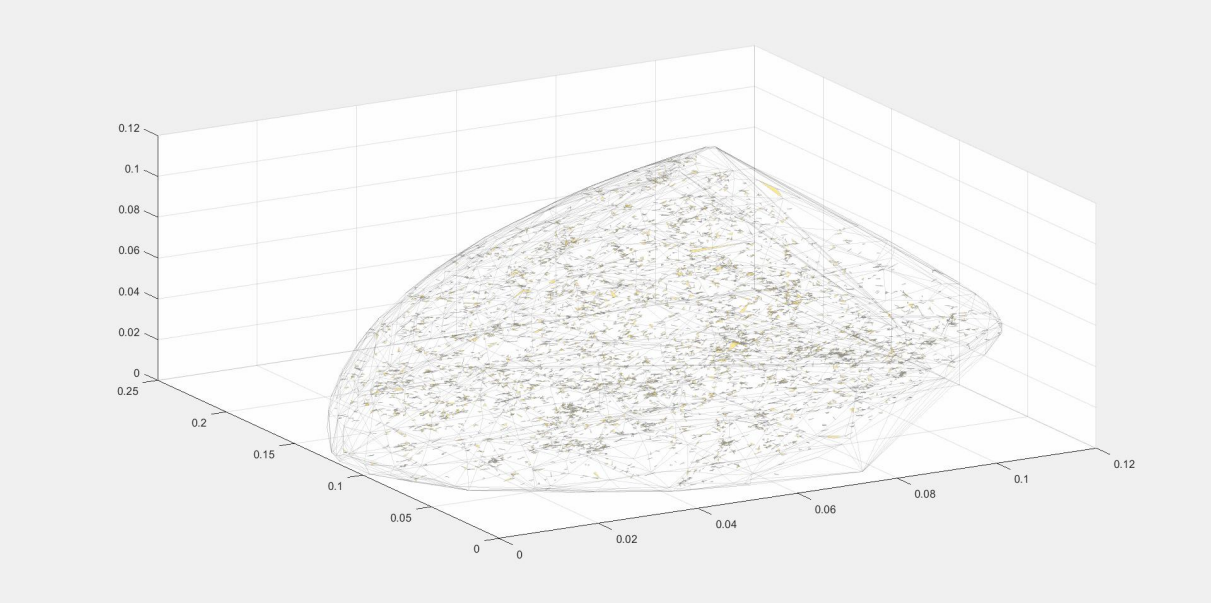


Figure 4.8: Select Snapshots of Density Sets

In the first snapshot we can see the misbehaved tetrahedra and see that they do, for the most part, make up the border regions of our space. Whether this is a natural aspect of the tetrahedralization, the data, or some error on our part is yet to be determined. The following snapshots are representative of how all other steps appear. Larger tetrahedra related to the smaller densities near the beginning and smaller tetrahedra with higher densities at the end.

4.6 Future Work

We have alluded to possible future work throughout this chapter, but have not included everything. Possible areas of future work include implementing different databases, creating the three dimensional Voronoi diagram, incorporating connectivity into PELT partitioning, using different blocks and models for three dimensions, and comprehensive error analysis.

For this project, due to time constraints, we restricted ourselves to the SDSS, the first comprehensive database we had access. Early on we used an incomplete database HYG and found that it was glaringly incomplete and so not conducive to being tetrahedralized. However, we should look into other databases and see if they present similar problems or not.

Creating the Voronoi dual of our tetrahedralization should be possible now that we have the Delaunay tetrahedralization. For anyone interested in pursuing this should work at the work of Yan, Wang, Lévy, and Liu[18]. Incorporating connectivity is easier in the Delaunay tetrahedralization as any given tetrahedron has exactly four neighbors. Of course, only if you consider the neighbors that any tetrahedron shares a face with. This connectivity can be much more complicated if every tetrahedron that shares a vertex is considered connected or in the Voronoi case, where there can be an irregular number of neighbors.

In our project, for simplicity, we used a constant model for the density of all the tetrahedra, which corresponds to saying the space between galaxies has as much density as the galaxies themselves. This is obviously an absurd claim, so attempting to find better three dimensional models to fit any type of data is a challenge. In addition, there would be error analysis associated with each type of fit. We also need to complete error analysis for our particular methods, as well as reduction in computation time.

Chapter 5

Analysis of the GC-Rich Regions of a DNA Sequence

Yu Jung Yeh and Jun Wang

5.1 Introduction

DNA, or deoxyribonucleic acid, is the hereditary material in humans and almost all other organisms. The information in DNA is stored as a code made up of four chemical bases: Adenine (A), Thymine (T), Cytosine (C), and Guanine(G).[1] This project is interested about the GC rich region in DNA sequence, which is defined as a certain DNA sequence carries long stretches of repeated G and C bases.[5] From pervious publications, this has been study that GC rich region play an important role in regulation of functional sequence which is certain piece of DNA sequence can be transcribed into mRNA then translated into protein. [5] The purpose to explore GC rich region is for understanding the disease mechanism, such as cancer, and developing gene therapy.[8]

Apply mathematical model to conduct GC rich region analysis has been study for many years.[2, 3] In this project, we will demonstrate a novel method named Bayesian Block Model to GC rich region analysis. Bayesian Blocks Model was first developed by Dr. Scargle in 1998 and combined with optimal partition model by Dr. Jackson in 2013.

[14] Bayesian Blocks model is a piecewise-constant model and originally applies on time series data which is Gamma-ray burst data. The purpose is find the optimum partition of the data in the observation interval then maximizing the fitness step function and segment the intervals into blocks. The Fitness Step Function been used in this project is Maximum Likelihood Function with two parameters, one is N which represented the number of data points in a block, the other one is M which represented the size of a block.

The goal of this project is detect the GC rich region by Bayesian Blocks Model and we will compare the results with Hidden Markov Model, which has been apply in GC rich region for many years. Besides, study about the interval structure between GC rich region also one of the target in this project. According to Dr. Wheelan's assumption, the intervals between GC rich region have a specific pattern. Assuming the interval length between first two GC rich region is x then the interval between second and third region would be x times a constant r, which r is a constant slightly over 1. The intervals will increase from one end to the middle of sequence by x times r with increasing power. So the intervals between GC rich region are increasing from one end to the middle and decreasing from middle to the another end. (Figure 5.1)

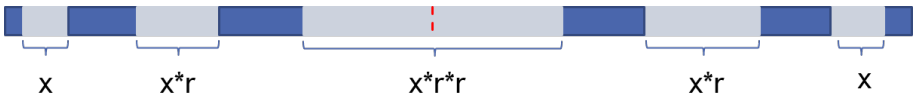


Figure 5.1: GC rich region

x = the interval length between first two GC rich region

r=1

5.2 Material

The DNA sequence is 58 million base-pair long which contains around 52% of AT and 48% of CG. Because this project only interesting about the G and C base, so whole sequence was converted into 0 and 1 sequence, which A or T convert into 0 and C or G convert into 1. Besides, for research convenience all "N" bases were removed.

5.3 Algorithm

5.3.1 Likelihood Function

Approach I

In the first approach, we treat each DNA base as a voronoi interval with length equals one. In this case, we can use the basic log-likelihood function

$$log(L) = N * log(\frac{N}{M}) - c$$

If the DNA base is A or T, we record the signal in that interval as 0; if the DNA base is C or G, the signal is 1, so the N is a vector of 0s and 1s and the length of N is 58 million. M is a vector of 1s with same length.

Issue N=0

When it is A or T in the interval, the log-likelihood function is 0*log(0), which is undefined and cannot be calculated in R. The mathematical definition of 0*log(0) is 0, so we manually assign it to 0 in the R code and the algorithm works well.

Approach II

In approach 2, we put a separator on the midway between two consecutive C or G base, and then we get the M vector: the number of DNA bases in each voronoi interval. Because each voronoi interval contains 1 C base or G base, the N vector in approach 2 is a

sequence of 1s whose length equals number of voronoi intervals, around 28 million. The DNA bases in a voronoi interval follows a binomial distribution with p = N/M. As a result, the log-likelihood function of approach 2 is

$$log(L) = N * log(\frac{N}{M}) + (M - N) * log(\frac{M - N}{M}) - c$$

Issue M=N

Approach 2 also have the issue of undefined 0*log(0) when M = N = 1. It happens when C or G base occurs more than 3 times consecutively. To solve this problem, we tried three solutions. The first one is assigning 0*log(0) to 0. The second one is assigning 0*log(0) to negative infinity because from biometric aspect, there is no changing point within a consecutive GC sequence. The third solution is adding a very small number e to all M to avoid the situation M=N. The number e is so small that it will not change the result.

$$log(L) = N * log(\frac{N}{(M + e)}) + (M + e - N) * log(\frac{M + e - N}{(M + e)}) - c$$

5.3.2 Hidden Markov Model

Viterbi Algorithm is dynamic programming algorithm which is one of Hidden Markov Model. We use Viterbi Algorithm for finding the most likely path to previous state, which are "Non-GC rich region" or "GC-rich region", then find the most likely transition from previous state to this state, eventually, fine the most likely path to the end state. For build up a Viterbi Hidden Markov Model, the space of states is defined as "Non-GC rich region" and "GC rich region" and the space of observation is "A", "C", "G", "T". The whole DNA sequence was combined with "A", "C", "G", "T" which can be

observed but each observation emits from" Non-GC rich" or "GC-rich" was an unknown information. There are three prior matrixes need to be set, first one is initial matrix, which are the probabilities of beginning from each state. The second one is transition matrix, which are the probabilities of transit from one state to another, for example, 0.85 means the probability of transit from Non-GC rich to Non-GC rich and 0.25 means probability of transit from GC rich to Non-GC rich. The third one is emission matrix, which are the probabilities of each observation emit from each state, for example, 0.25 means the probability of observation base "A" emit from "Non-GC rich region" and 0.15 means the probability observation base "A" emit from GC-rich region. All the numbers are references from previous publication.

5.4 Results

5.4.1 Bayesian Block Algorithm

Both Approach I (Figure 5.2) and II Bayesian Block algorithm are demonstrated to the unbinned whole DNA sequence. In addition, for reference information, the whole DNA sequence was binned with 100 bp first and computed the G and C intensity within each bin then show the result as histogram plot.

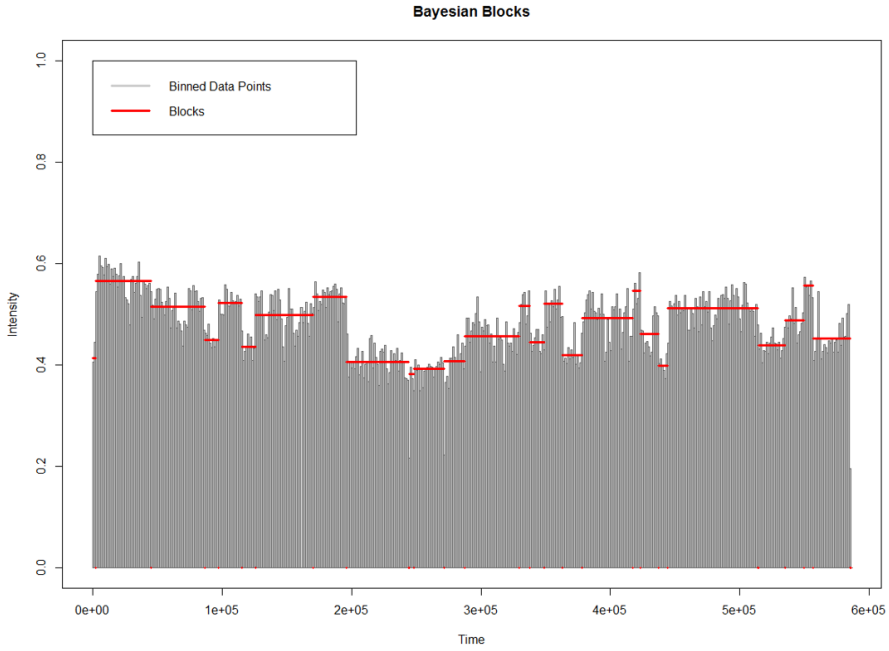


Figure 5.2: Approach I

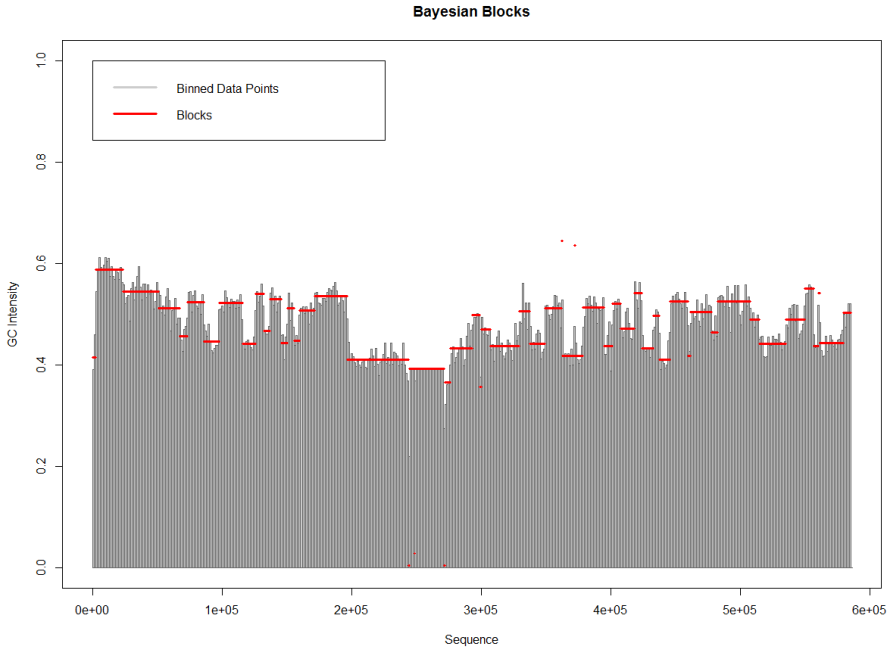


Figure 5.3: Approach II Solution 1

The three solutions of Approach II have identical results with appropriate constant value. (Figure 5.3, 5.4, 5.5) Solution 1 takes longest processing time and solution 3 takes shortest processing time. Also, we find that on our data, when e is less than or equal 0.01, solution 3 generate same results with the other two.

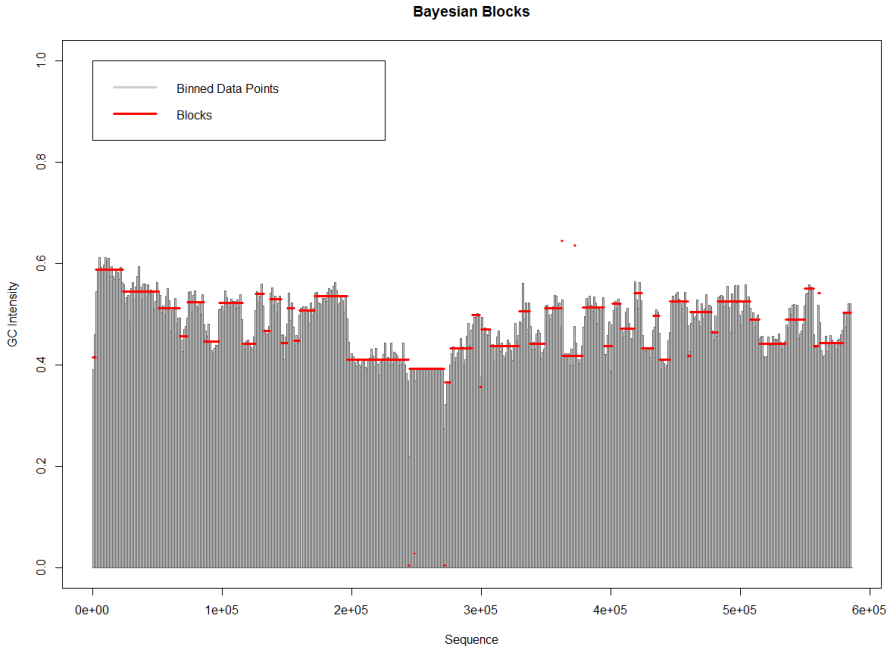


Figure 5.4: Approach II Solution 2

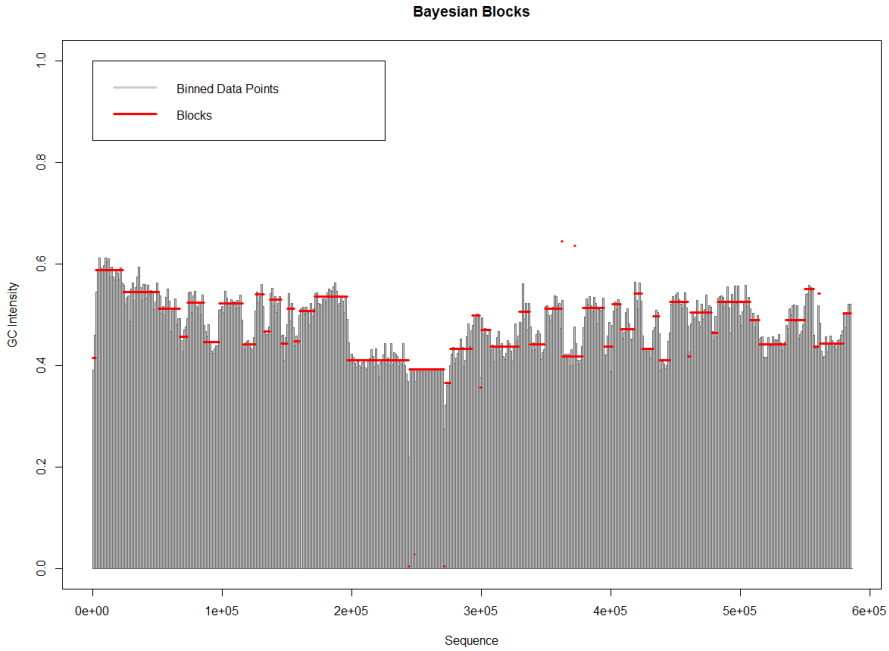


Figure 5.5: Approach II Solution 3

We compare the two solutions and both have similar segmentation. Approach I has simpler log-likelihood function but calculates 58 million intervals. Approach II calculates only 28 million intervals, however its log-likelihood function is complicated. We tried both approaches and find approach II is faster with less calculation, so we adopt approach II with log-likelihood function

$$log(L) = N * log(\frac{N}{M + 0.01}) + ((M + 0.01) - N) * log(\frac{(M + 0.01) - N}{M + 0.01}) - c$$

5.4.2 Constant Issue

The penalty constant is related to the prior distribution of number of blocks. Larger c generates fewer blocks but requires longer processing time. In this project, c=200 is acceptable but still has too many short intervals at some locations. (Figure 5.6) However, increasing the penalty constant and the process becomes significantly inefficient because the data size is too large. The PELT algorithm has been introduced, which is prunes large amount of potential changing points and reduce the processing time by around 50%, but it still takes around 24 hr to generate the plot with c = 200. Consequently, the separation of whole DNA sequence for further study was been considered.

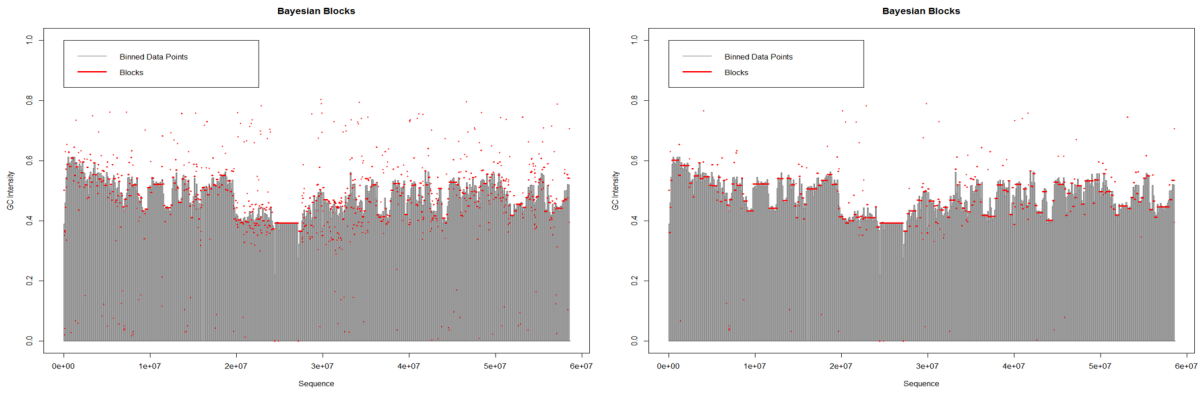


Figure 5.6: Left, Constant C=50; Right,Constant C=200

5.4.3 Separating DNA Sequence

At first we think about separate the data at the middle point since according to Dr. Wheelan's assumption, the DNA sequence has symmetric GC rich regions and the GC ratio is low at the middle. However, only the low GC ratio in the middle part of the DNA sequence has been observed. The length and position of GC rich region is not symmetric. (Figure 5.7)

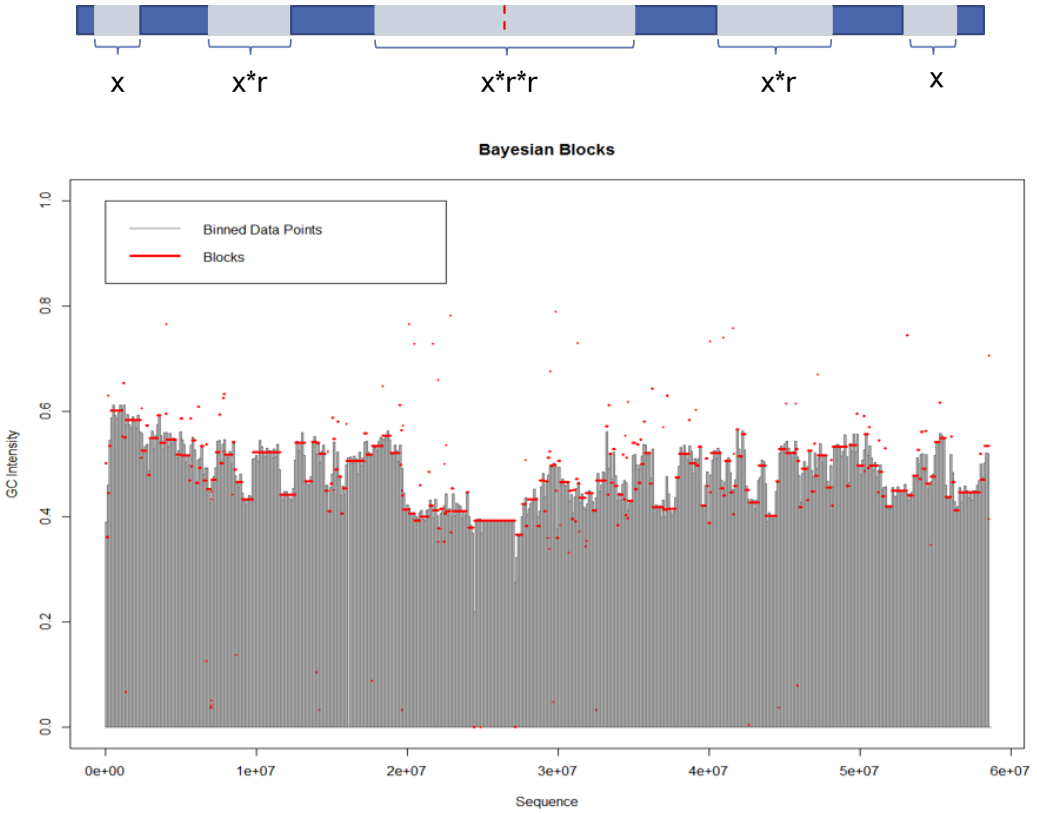


Figure 5.7: The sequence is not symmetric as previous assumption.

Then we consider to adopt k% from the data evenly. One method is choosing one data point every 1/k% data points from the original data. In this way, the number of Voronoi cell interval decrease to k% of original number, and the length of the interval is almost the same. The other method is choosing k% from the vector of G or C bases' positions. In this method, the number of Voronoi cell interval also decreases to k%, but

the length of each Voronoi interval increase to 1/k% of the original length, so we need to multiply the M vector by k% to neutralize the change. The first method is faster but the second one can keep more information from the whole data, so we choose the second method. We tried 1% of our data, from the plot, we can see that 1% can keep most information of the original data and c = 10 in partial data analysis can return an appropriate result. (Figure 5.8)

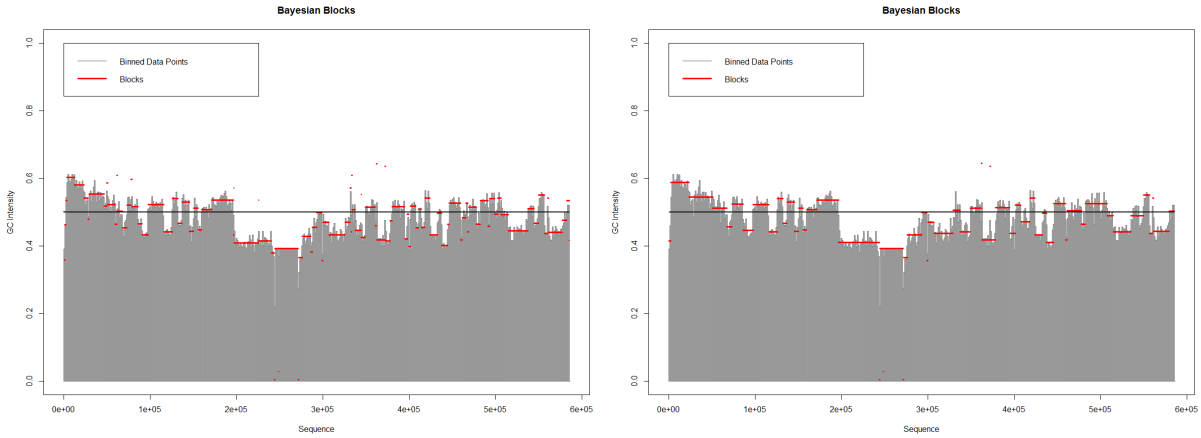


Figure 5.8: 1% of data with constant c = 5 (left) and c = 10 (right)

We find that when we take 1% of the data and use c = 2, the analysis generates similar plot as we use whole data and set c = 200. (Figure 5.9) The partial data analysis only takes few minutes while the whole data analysis takes days. We conclude it is an efficient procedure to estimate penalty constant in large data application: try different c in k% data and get the appropriate value, then divide the value by k% and use that in the whole data analysis. In our project, we estimate the appropriate c is 1000.

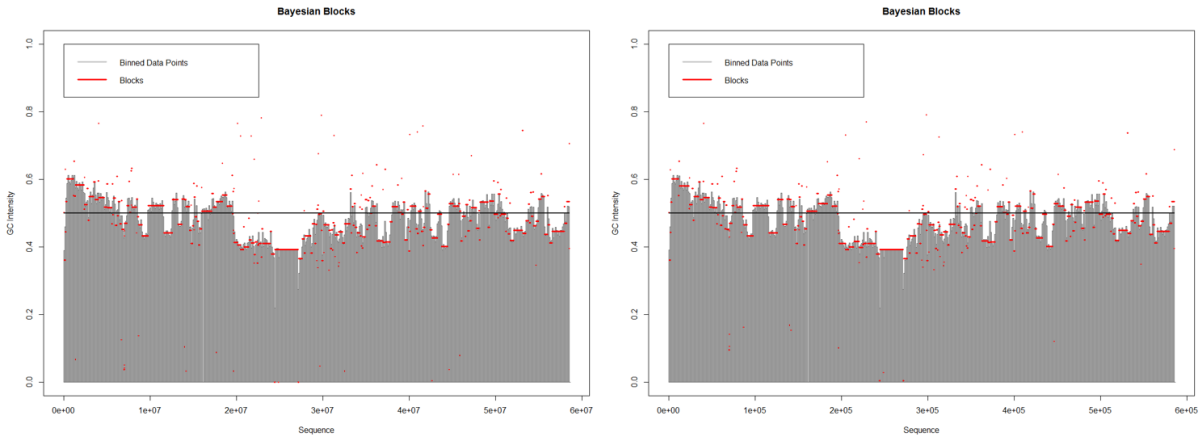


Figure 5.9: 100 % data with constant c = 200 vs 1% data with constant c = 2

5.4.4 Comparison with Hidden Markov Model

Using the set of parameters to generate a 5000 bp mimic DNA sequence as the validated dataset, and apply Bayesian Blocks and Viterbi Model to the mimic sequence. The result shows, within the high GC intensity region (histogram plot) Bayesian Blocks Model can define more clear blocks or "regions" than Viterbi Model which implies that Bayesian Blocks can detect a high GC intensity squence as one blocks and Viterbi Model detect as many small segments. In other word, Bayesian Blocks Model defines each high GC intensity sequence as one region(block) but Viterbi Model defines as many different regions(states). (Figure 5.10) Then apply Bayesian Blocks Model and Viterbi Algorithm in 1% of real DNA sequence data and we get the similar results. However, hidden Markov Model gives shorter programming processing time. (Figure 5.11)

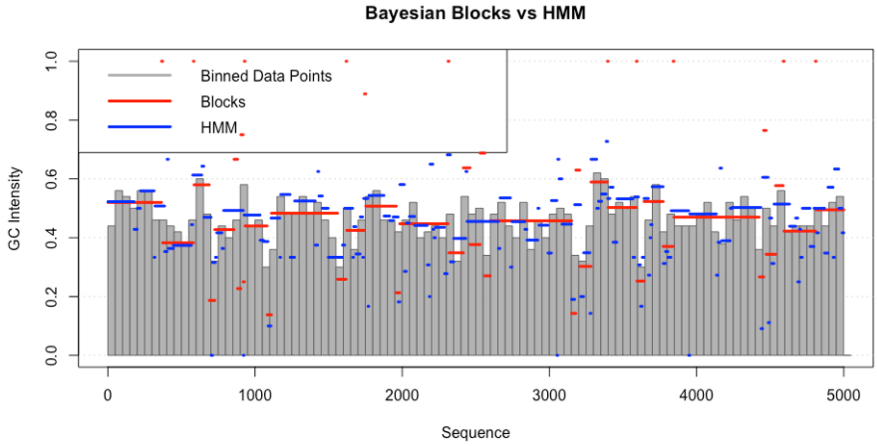


Figure 5.10: Bayesian Block vs HMM. Apply Bayesian Block and HMM to the mimic DNA sequence

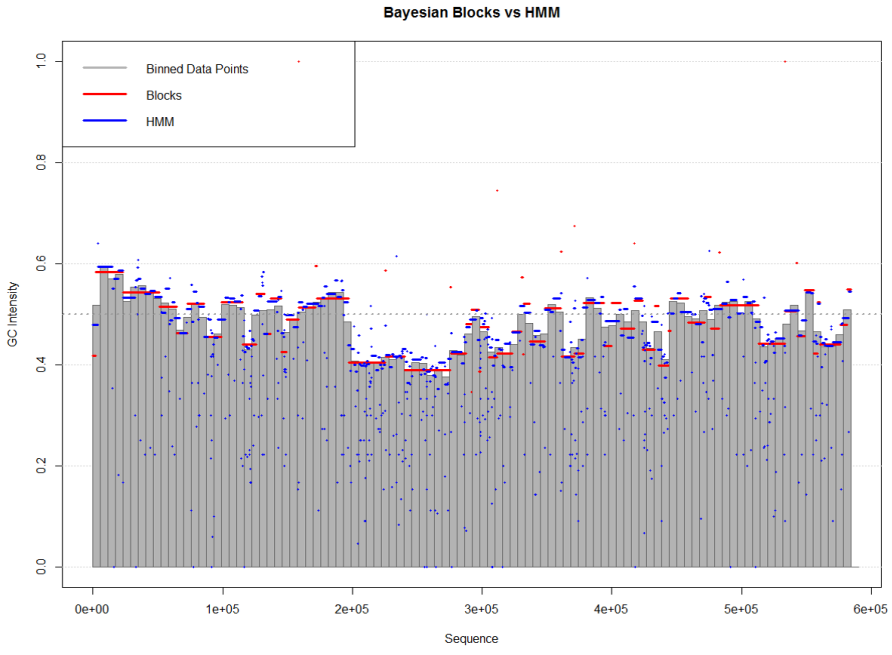


Figure 5.11: Bayesian Block vs HMH. Apply Bayesian Block and HMM to 1% of real data.

5.5 Discussion and Conclusion

Bayesian Blocks Model is a promising algorithm in DNA sequence analysis. However, the programming efficiency is a critical point and the solution been used to overcome the efficiency is carefully pick up Likelihood function and Penalty Constant. In addition, the PELT Algorithm can save around 50% of the processing time. Both Bayesian Blocks and Hidden Markov Model are unsupervised method, and the selection of prior is important. In Bayesian Blocks Model, only needs focus on constant selection. However, In HMM, the consideration of a set of parameters is require. Therefore, we believe, Bayesian Blocks Model could be a good choice in DNA sequence analysis besides the traditional Hidden Markov Model. In this project, we know the penalty constant selection is matter for Bayesian Blocks Model so we need further study about the selection of penalty constant. We fit each blocks with piecewise constant model in this project, in further study we could consider linear wise or exponential wise model. We did not found the evidence to prove Dr. Wheelan's assumption so understand the interval structure between GC-rich region would be the next step of our study. We know the GC-rich regions play roles in transcription regulation so we also want to see the relationship between transcription start site and GC rich region.

Chapter 6

Generalized Block Shapes

Ryan Shiroma, Stefanie Deo, Kristel Lenk, Mai Uyen Le, Ya-Ling Yao, Hongzhe Liu, Sucharu Gupta, and Lingjun Wang

6.1 Introduction to Piecewise Likelihood Modeling

6.1.1 Introduction

The idea of this project is to model arrival-time data with the ultimate goal being to model the shape of gamma ray burst radiation intensities. We assume that gamma ray bursts follow some underlying radiation intensity that varies over time. We might expect to see intensity to start off constant for some time(background radiation), then a spike of radiation intensity at the start of the burst, then a gradual reduction in intensity thereafter. This intensity comes in the form of individual photons hitting a sensor in a gamma ray telescope. The photon data can be collected and analyzed in a variety of ways, mostly involving some form of binning photons. We would therefore like to capture the shape of this intensity over time over multiple piecewise intensity functions. The current method, Bayesian blocks developed by Jeff Scargle[14], captures this gamma ray burst shape by representing it as a step function over time(piecewise constant blocks). Our method takes it one step further and generalizes Bayesian blocks to allow for non-constant blocks. If we can model differently shaped blocks rather than

patterns of piecewise constant blocks, we can hopefully better identify gamma ray bursts in the data.

6.1.2 Data Modes

There are multiple types of time series data relevant for our analysis for example, Point data, binned data, and unbinned data. We will focus our research on the second and third types,since the point measurement modeling are well discussed in current literature.

Point Data

The simplest data mode is **point data**. Point data are measurements from a sensor at specified time intervals. These point measurements generally come with some amount of normally distributed error. The variance on this error might be fixed or vary with time. It does not, however, vary with the magnitude of the measurement itself. Therefore, fitting a piecewise model on this type of data can be done simply with standard linear and non-linear regression techniques where each piecewise model will be chosen where the sum of squared errors is minimized in each block. In the case of variance changing over time(heteroskedasticity), a weighted least squares regression approach can be used.

Binned Data

The second data mode we use are **binned data**. These data points are recorded as the counts of photons that arrive within a time interval. Since we are dealing with counts here, each data point count follows a Poisson distribution with the λ parameter being the true intensity during that interval. And since the variance of a Poisson distribution is equal to λ, we can use linear or non-linear weighted least squares

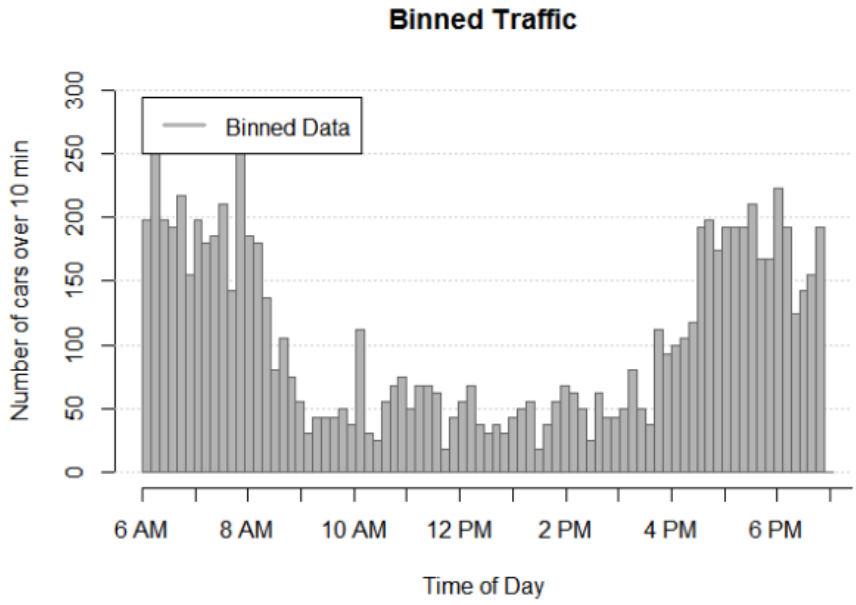


Figure 6.1: An example of binned data

Another type of binned data is **time-to-spill data** in which a bin is created for every k photons. This will mean that each bin will have variable width depending on the intensity of the radiation.

Unbinned Data

Unbinned data are the time arrivals of every individual photon. This type of data can be assumed to follow some form of piecewise non-homogeneous Poisson processes. We will discuss in more detail how we model this in 6.2. We can think of this as time-to-spill data where $k = 1$, or binned data where each bin has at most one photon.

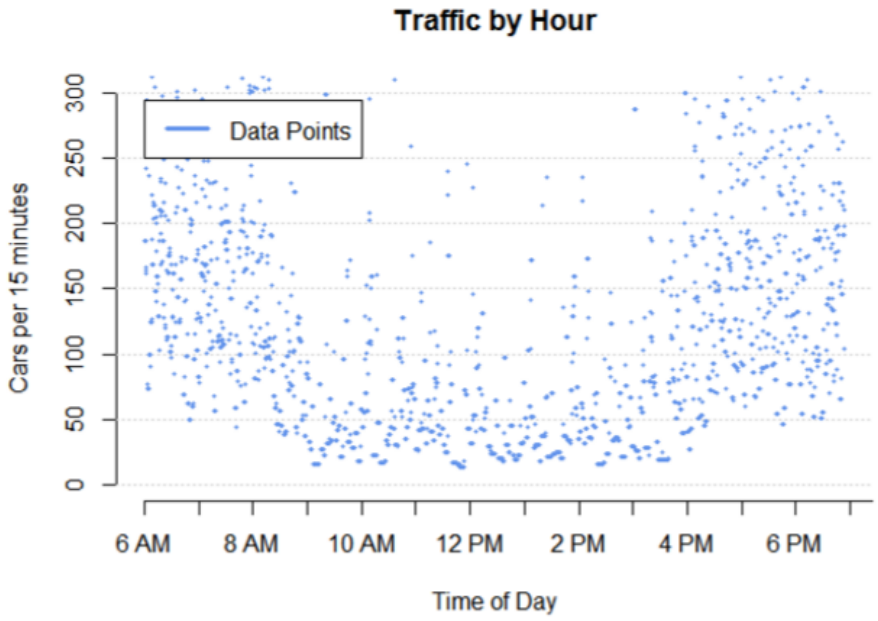


Figure 6.2: An example of unbinned data

6.1.3 Determining Model Cost

In order to find best fitting shapes to the data, we'll need a way to measure the fit of a shape to the data. Let's begin by looking at binned data. A bin count follows a Poisson distribution where the parameter λ is the intensity for that bin interval.

In other words, if we look at a bin with time length of T where the intensity per time unit is λ, the distribution of the number of photons that arrived in that bin is:

$$X \sim \text{Poisson}(\lambda T) \tag{6.1}$$

This brings us to the derivation of our cost function.

Let	$a = (a_1, a_2, ..., a_n)$	be a vector of n individual photon arrival times.
	$x = (x_1, x_2, ..., x_B)$	be a vector of the counts of photons in B sequential bins.
	T	be the total observation period in a block.
	dt	be the length of each bin.
	$\lambda(t)$	be the estimated intensity at time $t \in (0, T)$.

To get the probability that we received x given intensity $\lambda(t)$ we can simply get the product of the probabilities of each bin count. We can do this because independence between each photon arrival, and thus each bin, is assumed in a Poisson process.

$$\mathrm{P}(x|\lambda(t)) = \prod_{i=1}^{B} \mathrm{P}(x_i|\lambda(t_i)) \tag{6.2}$$

$$= \prod_{i=1}^{n} \frac{\big(\lambda(t_i)dt\big)^{x_i} e^{-\lambda(t_i)dt}}{x_i!} \tag{6.3}$$

This calculation can prove to be computationally expensive given that B factorials need to be calculated where some values of x_i are possibly very large. One method formulated by Thompkins [16] is to reduce the bin size to arbitrarily small widths where each bin has only either 0 or 1 photons. Using this approach we can model binned, unbinned, and time-to-spill data. Or in general, all Poisson based data.

Let	U	be the set of all arbitrarily small bins over T
	S	be the set of bins where 1 photon is captured. ($\lvert S\rvert = n$)

$$\mathrm{P}(a|\lambda(t)) = \prod_{b\in U} \mathrm{P}(b|\lambda(t)) \tag{6.4}$$

$$= \prod_{b\in S} \mathrm{P}(b|\lambda(t)) \prod_{b\in U\setminus S} \mathrm{P}(b|\lambda(t)) \tag{6.5}$$

$$= \prod_{b\in S} \big(\lambda(t)dt\big) e^{-\lambda(t)dt} \prod_{b\in U\setminus S} e^{-\lambda(t)dt} \tag{6.6}$$

$$\log(\mathrm{P}(a|\lambda(t)) = \sum_{b\in S} \log\Big[\big(\lambda(t)dt\big) e^{-\lambda(t)dt}\Big] - \sum_{b\in U\setminus S} \lambda(t)dt \tag{6.7}$$

$$= \sum_{b\in S} \Big[\log\big(\lambda(t)\big) + \log(dt) - (\lambda(t)dt)\Big] - \sum_{b\in U\setminus S} \lambda(t)dt \tag{6.8}$$

$$= \sum_{b\in S} \log\big(\lambda(t)\big) + n\log(dt) - \sum_{b\in S} \lambda(t)dt \tag{6.9}$$

Let $dt \to 0,\ \tau = n\log(dt)$

$$= \sum_{b\in S} \log\big(\lambda(t)\big) + \tau - \int_0^T \lambda(t)\mathrm{d}t \tag{6.10}$$

Therefore for k blocks in the entire dataset, the total cost is represented as,

$$L = \sum_{i=1}^{k} \Big[\sum_{b\in S_i} \log\big(\lambda_i(t)\big) + \tau_i\Big] - \sum_{i=1}^{k} \int_0^{T_i} \lambda_i(t)\mathrm{d}t \tag{6.11}$$

$$L = \sum_{i=1}^{k} \Big(\sum_{b\in S_i} \log\big(\lambda_i(t)\big)\Big) + \sum_{i=1}^{k} \tau_i - \sum_{i=1}^{k} \int_0^{T_i} \lambda_i(t)\mathrm{d}t \tag{6.12}$$

where $\sum_{i=1}^{k} \tau_i = \sum_{i=1}^{k} n_i \log(dt) = N\log(dt)$ which is a constant with respect to the data and can therefore be dropped for likelihood ratio tests and model comparison.

$$L^* = \sum_{i=1}^{k} \Big(\sum_{b\in S_i} \log\big(\lambda_i(t)\big)\Big) - \sum_{i=1}^{k} \int_0^{T_i} \lambda_i(t)\mathrm{d}t \tag{6.13}$$

$$= \sum_{i=1}^{k} L(\theta_k) \qquad \text{where } \theta_k \text{ are the intensity function parameters of the } k\text{th block} \tag{6.14}$$

Maximizing this quantity with respect to all θ_i intensity function parameters and their respective change-points over the dataset will result in the optimal block segmentation.

$$\underset{\theta_k}{\operatorname{argmax}} \sum_{i=1}^{k} L(\theta_k) \tag{6.15}$$

However, without any regularization, maximizing this quantity will result in $k = N$, ie. one block per data point. To penalize small block sizes, we incorporate a penalization parameter, ncp_{prior} for each additional block. This is equivalent to adding a prior on the number of points per block,

$$\underset{\theta_k}{\operatorname{argmax}} \sum_{i=1}^{k} \Big(L(\theta_k) - ncp_{prior} \Big) \tag{6.16}$$

6.2 Relevant block shape MLE derivations

Although the cost function can handle any intensity function with any number of parameters, we have identified four shapes that are relevant to gamma ray burst modeling.

6.2.1 Constant Function Intensity: $\lambda(t) = a$

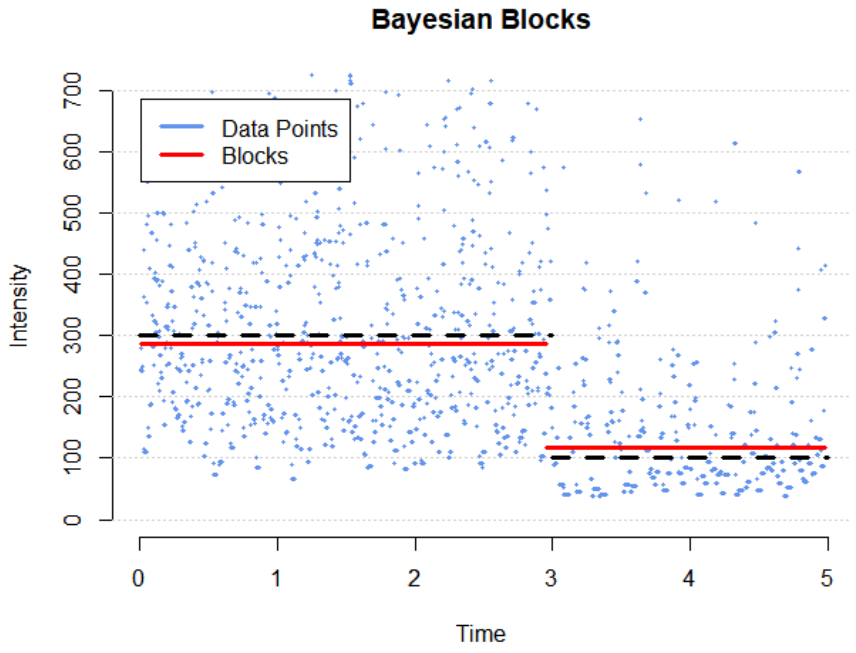


Figure 6.3: An example of a piecewise constant model

In this section, the event rate for a block is assumed to be **constant**. Using the

$$L(a) = \sum_{i=1}^{n} \log(a) - \int_0^T (a)dt \tag{6.17}$$

$$= n\log(a) - aT \tag{6.18}$$

To derive the maximum likelihood estimator of a, we take the derivative with respect to a.

$$\frac{\partial L(a)}{\partial a} = \frac{n}{a} - T \stackrel{\text{set}}{=} 0 \tag{6.19}$$

$$\rightarrow \hat{a}_{\text{MLE}} = \frac{n}{T} \tag{6.20}$$

Substituting this estimate back in to the likelihood function we get,

$$L(a) = n\log(\frac{n}{T}) - \frac{n}{T}T \tag{6.21}$$

$$L(a) = n\log(\frac{n}{T}) - n \tag{6.22}$$

In the special case that only constant blocks are being used over the entire data set, the $-n$ can be dropped.

6.2.2 Linear Function Intensity: $\lambda(t) = at + b$

In this section, the event rate for a block is assumed to be **linear**. Let N be number of events in the block, $M = t_n - t_1$ is the length of the block, a is the rate of signal variation over the block, and b is the signal at the beginning of the block. Then, the block likelihood for the case of event data t_i is

$$L(a,b) = \sum_{i=1}^{n} log(at_i + b) - \int_{t_1}^{t_n} (at + b)dt \tag{6.23}$$

$$= \sum_{i=1}^{n} log(at_i + b) - \frac{a}{2}(t_n^2 - t_1^2) - b(t_n - t_1) \tag{6.24}$$

$$= \sum_{i=1}^{n} log(at_i + b) - aS - bM \qquad \text{where } S = \frac{1}{2}(t_n^2 - t_1^2) \tag{6.25}$$

To derive the maximum likelihood estimators of a and b, we take the derivatives with respect to a and b, respectively.

$$\frac{\partial L(a,b)}{\partial a} = \sum_{i=1}^{n} \frac{t_i}{at_i + b} - S \stackrel{\text{set}}{=} 0 \tag{6.26}$$

$$\frac{\partial L(a,b)}{\partial b} = \sum_{i=1}^{n} \frac{1}{at_i + b} - M \stackrel{\text{set}}{=} 0 \tag{6.27}$$

Newton's method is used to solve for the converged values of the estimates of a and b. Figure 6.4 illustrates an example of linear block model, where the dashed lines are the true intensities of the simulated data and the solid lines are the fitted blocks.

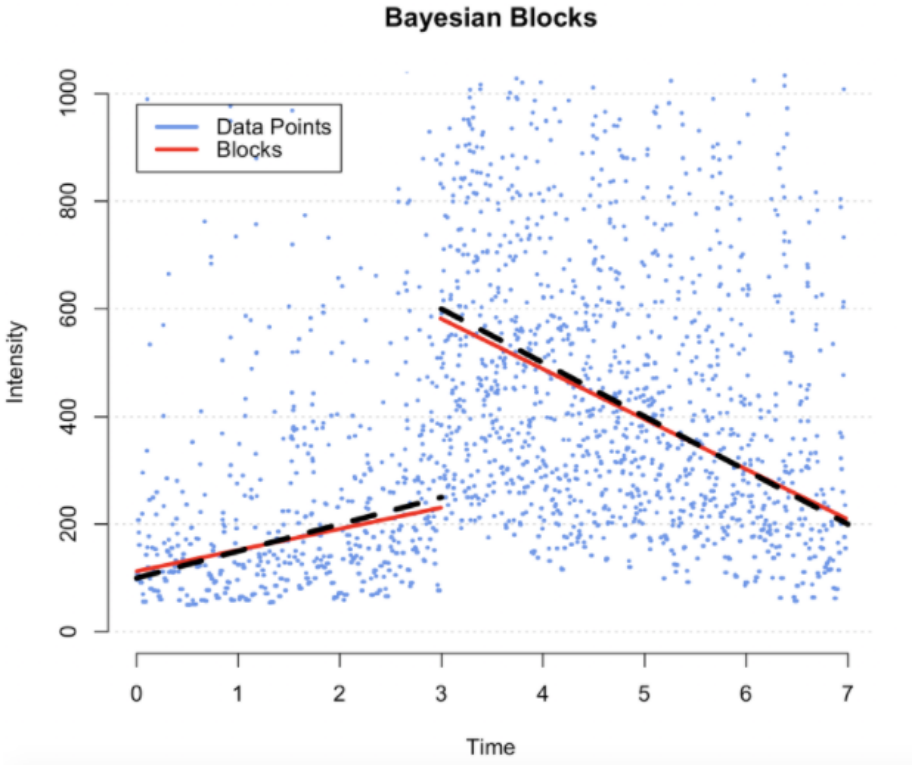


Figure 6.4: An example of a linear model.

6.2.3 Exponential Function Intensity: $\lambda(t) = ae^{bt} + c$

In this section, the event rate for a block is assumed to be **exponential**. Let N be number of events in the block, $M = t_n - t_1$ is the length of the block, and c is the

background noise constant. Then, the block likelihood for the case of event data t_i is

$$L(a,b,c) = \sum_{i=1}^{n} log(ae^{bt_i} + c) - \int_{t_1}^{t_n} (ae^{bt_i} + c)dt \tag{6.28}$$

$$= \sum_{i=1}^{n} log(ae^{bt_i} + c) - \frac{a}{b}\left[e^{b(t_n - t_1)} + c(t_n - t_1)\right] \tag{6.29}$$

$$= \sum_{i=1}^{n} log(ae^{bt_i} + c) - \frac{a}{b}\left(e^{bM} + cM\right) \tag{6.30}$$

To derive the maximum likelihood estimators of a and b, we take the derivatives with respect to a and b, respectively.

$$\frac{\partial L(a,b,c)}{\partial a} = \sum_{i=1}^{n} \frac{e^{t_i b}}{ae^{t_i b} + c} - \frac{1}{b}\left[e^{bM} + cM\right] \stackrel{\text{set}}{=} 0 \tag{6.31}$$

$$\frac{\partial L(a,b,c)}{\partial b} = \sum_{i=1}^{n} \frac{t_i e^{t_i b}}{ae^{t_i b} + c} + \frac{a}{b^2}\left[e^{bM} + cM\right] - \frac{a}{b}\left[Me^{bM}\right] \stackrel{\text{set}}{=} 0 \tag{6.32}$$

$$\frac{\partial L(a,b,c)}{\partial c} = \sum_{i=1}^{n} \frac{1}{ae^{t_i b} + c} - \frac{a}{b}M \stackrel{\text{set}}{=} 0 \tag{6.33}$$

Newton's method is used to solve for the converged values of the estimates of a, b, and c. Figure 6.5 illustrates an example of exponential block model, where the dashed line is the true intensity of the simulated data and the solid line is the fitted block.

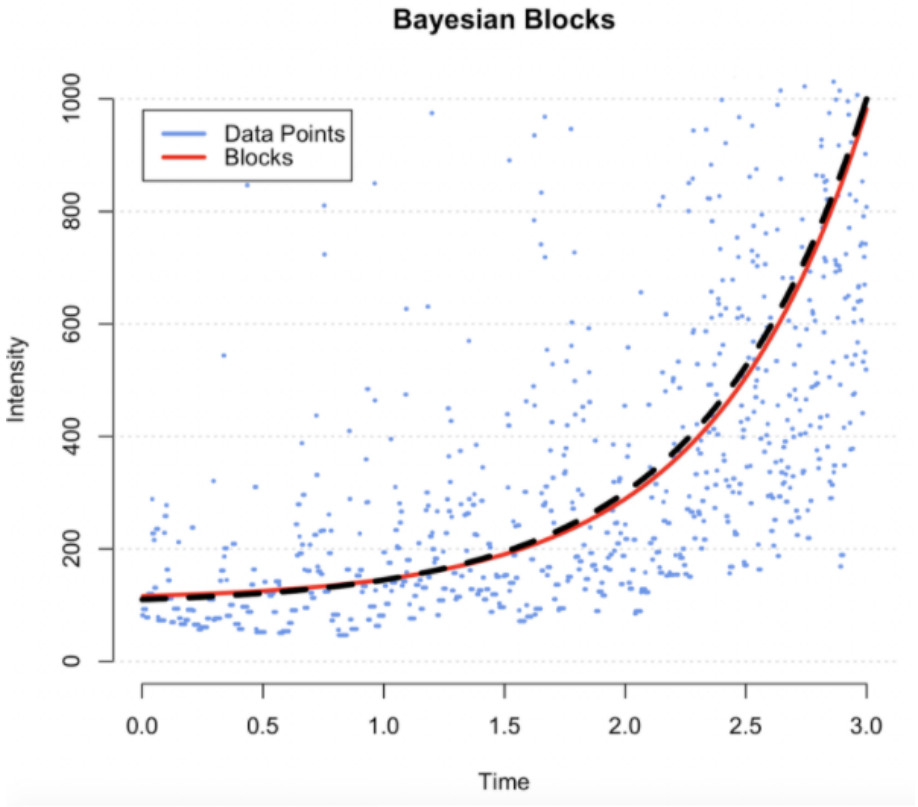


Figure 6.5: An example of an exponential model.

6.2.4 Power Function Intensity: $\lambda(t) = at^b + c$

In this section, the event rate for a block is assumed to come from a **power** function. Let N be number of events in the block, $M = t_n - t_1$ is the length of the block, and c is the background noise constant. Then, the block likelihood for the case of event data

t_i is

$$L(a,b,c) = \sum_{i=1}^{n} log(at_i^b + c) - \int_{t_1}^{t_n} (at_i^b + c)dt \tag{6.34}$$

$$= \sum_{i=1}^{n} log(at_i^b + c) + \left[\frac{a}{b+1}(t_n - t_1)^{b+1} + c(t_n - t_1)\right] \tag{6.35}$$

$$= \sum_{i=1}^{n} log(at_i^b + c) + \left[\frac{a}{b+1}M^{b+1} + cM\right] \tag{6.36}$$

To derive the maximum likelihood estimators of a,b, and c, we take the derivatives with respect to a, b, and c, respectively.

$$\frac{\partial L(a,b,c)}{\partial a} = \sum_{i=1}^{n} \frac{t_i^b}{at_i^b + c} - \frac{M^{b+1}}{b+1} \stackrel{\text{set}}{=} 0 \tag{6.37}$$

$$\frac{\partial L(a,b,c)}{\partial b} = \sum_{i=1}^{n} \frac{at_i^b ln(t_i)}{at_i^b + c} + \frac{aM^{b+1}}{(b+1)^2} - \frac{aM^{b+1}lnM}{b+1} \stackrel{\text{set}}{=} 0 \tag{6.38}$$

$$\frac{\partial L(a,b,c)}{\partial c} = \sum_{i=1}^{n} \frac{1}{at_i^b + c} - M \stackrel{\text{set}}{=} 0 \tag{6.39}$$

Newton's method is used to solve for the converged values of the estimates of a, b, and c. Figure 6.6 illustrates an example of power block model, where the dashed line is the true intensity of the simulated data and the solid line is the fitted block.

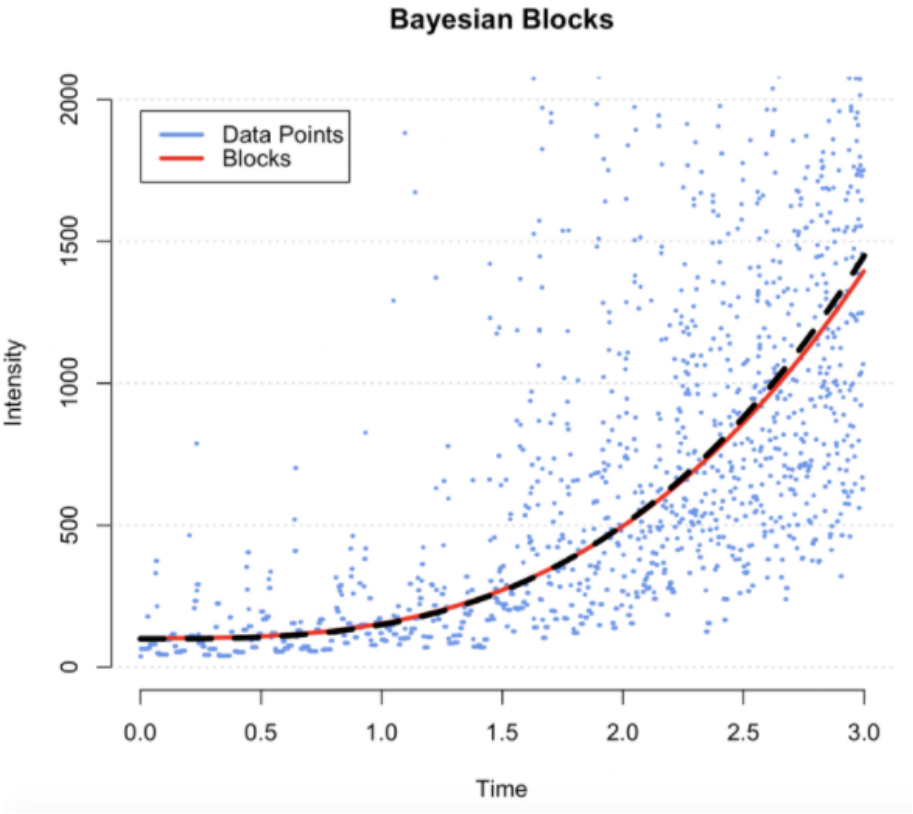


Figure 6.6: An example of a power model.

6.3 Comparing and Combining Different Block Shapes

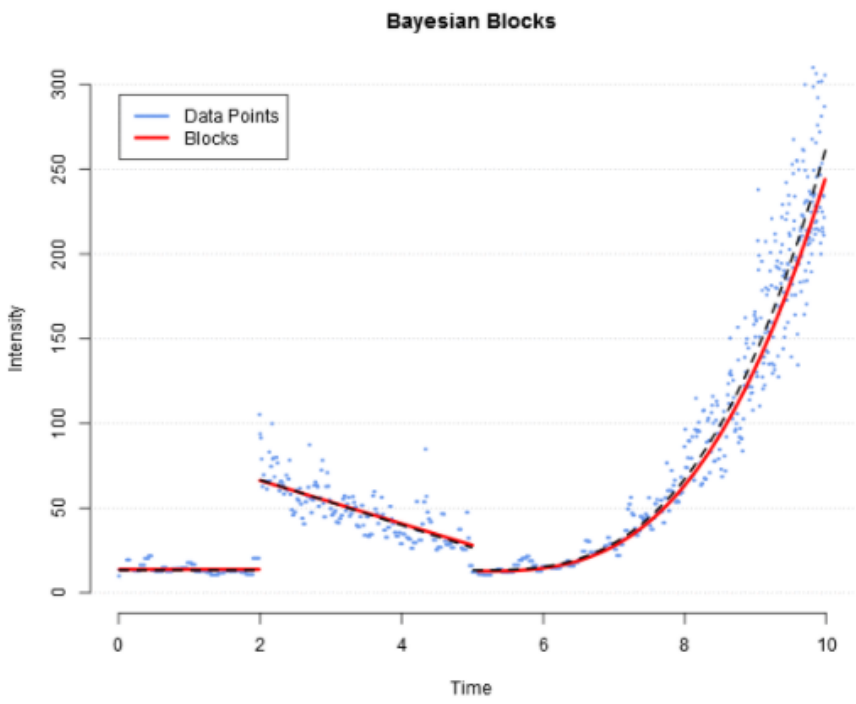


Figure 6.7: A combination of constant, linear and exponential models

The main goal of this project is to generalize the work of Jackson [7] and Killick [11] from a piecewise constant model to a piecewise model of any combination of function shapes. Above we described how we can find a best fit of any intensity function shape to any general Poisson process data. Obviously, the more complicated the intensity function shape is allowed to be, the better the fit to the data will be, thus best log-likelihood cost. We *could* fit the above data in the plot with three separate exponential functions and get the same three red block lines. However in the case of the first two blocks, an exponential function is overly complex. Simple constant and linear blocks are sufficient. This overfitting problem necessitates a way to penalize overly complex block shapes.

Due to their simplicities, we tried three methods of comparing different block shapes by using the number of estimated parameters to represent each block shape's complexity

and imposing additional penalties.

6.3.1 Likelihood Ratio Test

One way to compare two nested models with log likelihoods is to perform a likelihood ratio test. In our work we used Wilks'[17] χ^2 distribution approximation to the test statistic to calculate an appropriate p-value for the test. In this test we have,

H_0 : A simple model is sufficient(ex. constant block)

H_a : A more complicated model should be considered (ex. linear block)

According to Wilks' theorem, twice the difference between the log likelihoods will be distributed under a χ^2 distribution with the degrees of freedom equaling the difference in the number of parameters between the two models. So to compare a linear fit to a constant fit we can test the hypothesis with the following test statistic.

$$2(L(\theta_{\text{linear}}) - L(\theta_{\text{constant}})) \sim \chi^2_1 \tag{6.40}$$

Wilks' theorem requires that observations be independent, which they are in a Poisson process. It also requires that models are nested, which is true for constant-linear, constant-power, and linear-power comparisons. Finally the number of observations must be large. We performed 10,000 simulations for different values of n as low as 20 and the distribution approximation is still adequate. For shape comparisons with less than 20 observations we just go with the simpler model regardless of the test statistic value.

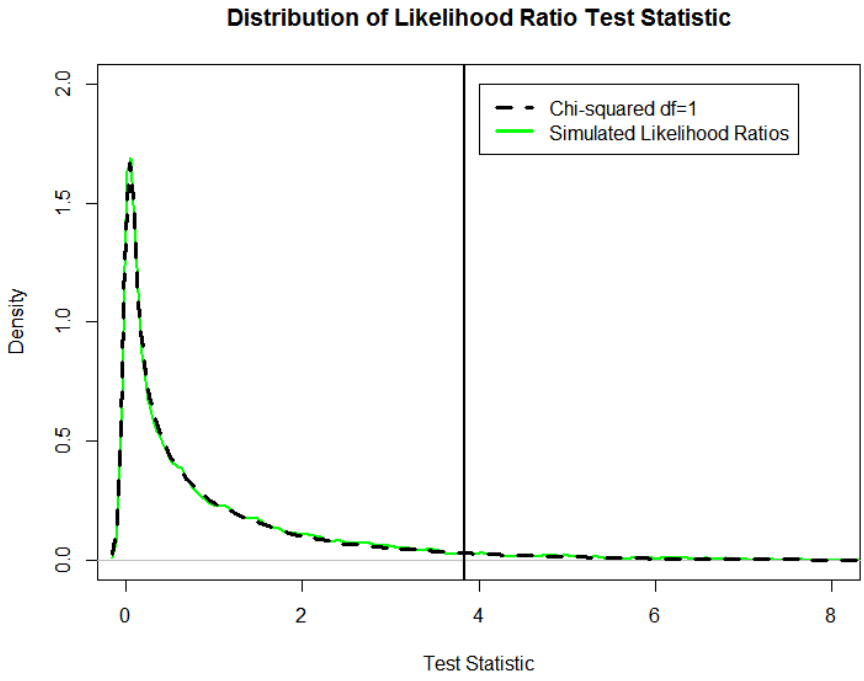


Figure 6.8: 10,000 simulations of test-statistics with n=20

We then reject the simpler model in favor of the more complex model when the p-value of the test statistic is below some threshold (arbitrarily set here at 0.05). An alpha(type I error) of 0.05 corresponds to 3.84 from a $\chi^2_{df=1}$. This critical value can be thought of as the additional penalty for a shape with one additional parameter. In other words, an example of the costs of two competing shapes can be as follows:

$$\text{Constant Block Cost} = L(\theta_{\text{constant}}) - ncp_{prior} \tag{6.41}$$

$$\text{Linear Block Cost} = L(\theta_{\text{linear}}) - ncp_{prior} - \chi^2_{df=1,1-\alpha} \tag{6.42}$$

$$\text{Power Block Cost} = L(\theta_{\text{power}}) - ncp_{prior} - 2\chi^2_{df=1,1-\alpha} \tag{6.43}$$

Which ever cost is higher is chosen as the appropriate block shape. A major benefit to likelihood ratio tests over AIC/BIC is that type I and type II error rates are adjustable with the alpha parameter.

6.3.2 AIC and BIC

Another method of model comparison is to use AIC or BIC. These are another two special cases of penalizated likelihoods shown in the papers by Killick[11] and Jong[9]. In both cases our goal is to maximize the following quantities:

$$\text{AIC} = 2\left(\sum_{i=1}^{k} L(\theta_k)\right) - 2\sum_{i=1}^{k} p_i \tag{6.44}$$

$$\text{BIC} = 2\left(\sum_{i=1}^{k} L(\theta_k)\right) - \log(N)\sum_{i=1}^{k} p_i \tag{6.45}$$

where p_i represents the number of parameters needed for block i's intensity function. Note that this formulation is equivalent to using the likelihood ratio test with alpha=0.15.

This corresponds to the following costs under each block shape for AIC:

$$\text{Constant Block Cost} = L(\theta_{\text{constant}}) - ncp_{prior} - 1 \tag{6.46}$$

$$\text{Linear Block Cost} = L(\theta_{\text{linear}}) - ncp_{prior} - 2 \tag{6.47}$$

$$\text{Exponential Block Cost} = L(\theta_{\text{exp}}) - ncp_{prior} - 3 \tag{6.48}$$

$$\text{Power Block Cost} = L(\theta_{\text{power}}) - ncp_{prior} - 3 \tag{6.49}$$

And similarly for BIC:

$$\text{Constant Block Cost} = L(\theta_{\text{constant}}) - ncp_{prior} - \frac{1}{2}\log(N) \tag{6.50}$$

$$\text{Linear Block Cost} = L(\theta_{\text{linear}}) - ncp_{prior} - \log(N) \tag{6.51}$$

$$\text{Exponential Block Cost} = L(\theta_{\text{exp}}) - ncp_{prior} - \frac{3}{2}\log(N) \tag{6.52}$$

$$\text{Power Block Cost} = L(\theta_{\text{power}}) - ncp_{prior} - \frac{3}{2}\log(N) \tag{6.53}$$

A few benefits to AIC and BIC over the likelihood ratio test is that it does not require the models to be nested and is parameterless.

6.4 Computational Considerations

We have found that the fitting complicated intensity functions is computationally expensive and thus makes this block shapes generalization impractical. Two possible solutions to reduce this runtime are to instead refit on a subsample of previous points or to bin previous points and fit to the bin counts. Of course both of these options should sacrifice shape parameter estimate accuracy for computational speed. We explored the binned approach below.

6.4.1 Iterated Least squares method

The iterated least squares method discussed below was developed by Massey, Parker and Whitt [12].
We start by only considering linear intensity. Over the interval $[0, T]$, the intensity function is

$$\lambda(t) = at + b, \qquad 0 \leq t \leq T \tag{6.54}$$

We then assume that the time interval $(0, T]$ is divided into N subintervals

$$\left(\frac{(k-1)T}{N}, \frac{kT}{N}\right), \qquad 1 \leq k \leq N \tag{6.55}$$

and count the number of data points in each bin. This results in N mutually independent Poisson random variables Y_k with means

$$\lambda_k = \frac{T}{N}(ax_k + b), \tag{6.56}$$

where

$$x_k = \left(k - \frac{1}{2}\right)\frac{T}{N}, \qquad 1 \leq k \leq N. \tag{6.57}$$

We then construct the linear model $Y = \alpha + \beta x + \epsilon$ and estimate α and β by minimizing the weighted sum of squared errors

$$\min_{\alpha,\beta} \sum_{k=1}^{N} w_k (Y_k - [\alpha + \beta x_k])^2 \tag{6.58}$$

where

$$w_k = \frac{\frac{N}{\lambda_k}}{\sum_{k=1}^{N} \left(\frac{1}{\lambda_k}\right)}, \qquad 1 \leq k \leq N. \tag{6.59}$$

The iterative approach is as follows: we first estimate α and β the usual way using OLS. Then, we calculate the values for λ_k (it is both the mean and variance of Y_k) and use it to calculate the weights w_k. For the second iteration, we again estimate α and β but this time using the calculated weights, w_k, to account for heteroskedasticity. Convergence in estimates is quick, usually in less than five iterations.

Next, we will compare the binned and unbinned methods in speed and accuracy of estimates. Figure 6.9 shows the speed it takes for each method to estimate the model parameters. First, we generated samples with various sizes. Then, at each sample size, model parameters were estimated 50 times. The solid lines represent mean speed, the bars 25^{th} and 75^{th} quantiles. The number of bins in the binned method is set to increase linearly with sample size, hence the linear upward trend in mean speed. We see that the unbinned method should be preferred for large samples, if computational speed is an important factor.

Figures 6.10 and 6.11 result from generating 50 different data sets for each sample size and estimating model parameters. The solid lines represent the mean error, while the dotted lines show the 95^{th} and 5^{th} quantiles. Estimate error refers to the percent difference between the estimation error and the known true parameter value. Accuracy of estimation is comparable in both methods.

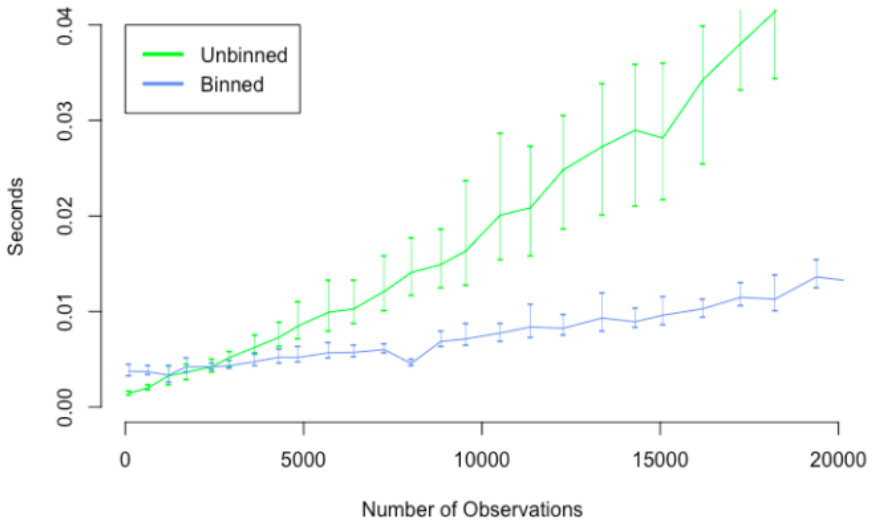


Figure 6.9: Comparing the Speed of Parameter Estimation

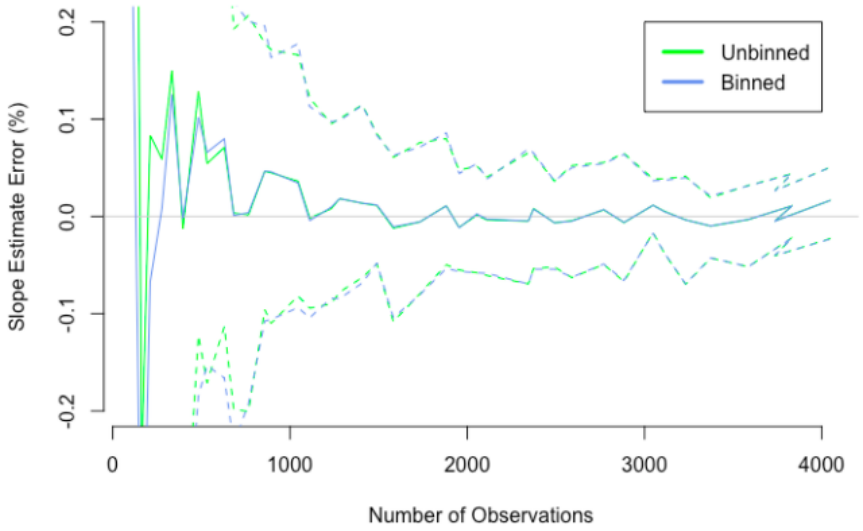


Figure 6.10: Comparing the Accuracy of Slope Estimates

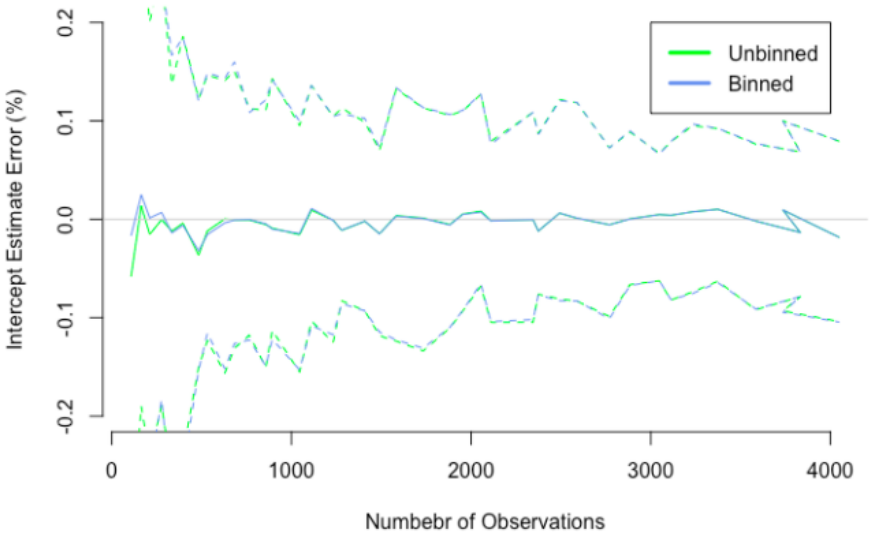


Figure 6.11: Comparing the Accuracy of Intercept Estimates

It is worth noting that this method can also be applied to exponential models using log-transformation. Starting with exponential intensity,

$$\lambda(t) = ae^{bt} \tag{6.60}$$

we apply log transformation

$$\log \lambda(t) = \log a + bt \tag{6.61}$$

$$\Rightarrow Y'_k = \log Y_k \tag{6.62}$$

$$\Rightarrow \hat{Y_k} = \log \alpha + \beta x_k \tag{6.63}$$

which results in an updated objective function (equation 6.58)

$$\min \sum_{k=1}^{N} w_k (Y'_k - \hat{Y_k})^2 \tag{6.64}$$

6.5 Future Work

Due to the time constraints we have only grazed the surface of what is possible with Bayesian blocks. These are a few ideas we would like to do for future work.

- We would like to try adding the gamma ray burst shape function formulation to our model $\lambda(t) = \frac{a}{e^{b/t}e^{t/c}} + d$ This formulation comes from Norris et al.[13]
- A more thorough analysis of the sensitivities of ncp_{prior} and block complexity penalties on false positive rates.
- Research more intensity function fitting methods to improve algorithm speed.
- A comparison between our method and existing methods for identifying GRB's.
- Incorporate Bayes factor to allow for prior information on model parameters.

Chapter 7

Real-time Analysis

7.1 Applications to Real-time analysis

In many cases, it is fine to analyze existing data, or data that has already been collected in the past. However, in other cases, researchers are interested in detecting activity in real time. For example, in astronomy a scientist may often observe a part of the sky which is a "blank field" where there is no object of obvious interest. The scientist may be interested in detecting a "transient", which is an object that was previously invisible and suddenly brightens, such as a supernova. Figure1.1 is a picture of such a transient, a cosmic flash which was first detected by NASA's Hubble telescope on February 26, 2006. It steadily rose in brightness for 100 days, and then dimmed back to oblivion after another 100 days. The left part of Figure1.1 is of a blank field, and for this we would expect only noise in our intensity data as shown in the left area of the plot below(Figure1.2). When the transient appears, the signal might look something like the burst of intensity as shown in the middle of the plot.

7.2 Bayesian Block Triggers vs Thresholding Triggers

7.2.1 Thresholding

The current method of detecting transients is by setting a threshold value. There is usually a low, but steady, incoming signal stream which is just noise. The idea is that when the incoming signal rises above the threshold, a transient detection is declared. The problem is that the threshold needs to be pre-set, and typically its value is determined based on some time-averaging of the data. One also needs to specify a window of time over which an average is drawn. The idea here is that we have a left window (the green one), and we have another window (the red one) in Figure1.3 that "slides" to the right whenever new data comes in. The average of the current sliding window is compared to the green window. So, since there are 2 parameters that need to be chosen, the threshold method can be quite sensitive. Some aspects of this problem can be avoided by adopting several different thresholds, but this can be problematic.

7.2.2 Bayesian Block algorithm

By contrast, the Bayesian Block algorithm does not rely on any pre-set threshold or pre-set window. This makes it ideal for transient detection. When every new data comes in, the algorithm would recompute it. If the algorithm determines that the intensity does not change a lot and best fit is still one constant block, no change point is detected. But if the algorithm determines the intensity change a lot and the best fit is a new block, then a change point is declared. So here on Figure1.4, the green line: the first block, is the best fit block for the background noise. At around the 5th second, BB algorithm finds the intensity had shifted, and the red line is the best guess for the second block, which is a linear block.

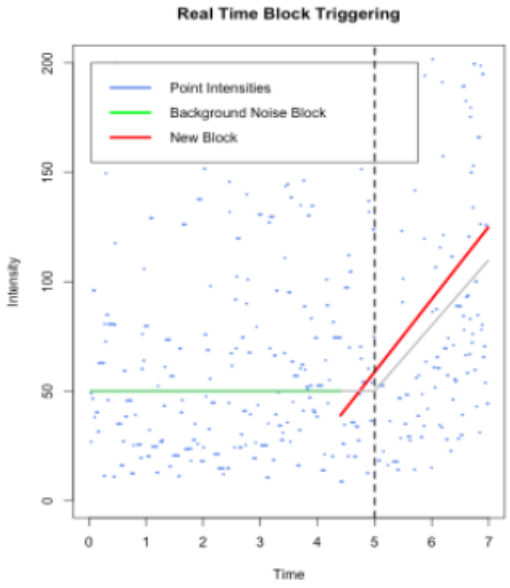


Figure 7.1: Real Time Block Triggering

7.3 Comparing Bayesian Blocks and Thresholding

Before the comparison of Bayesian block v.s. thresholding algorithms starts, some definitions has to be clarified: The data point at which real change of intensity had taken place, we call it ***True change-point*** (Tr); The algorithm's best guess on where the change-point is, we call it ***Estimated change-point*** (E); The data point at which the algorithm start to realize that a change of intensity had occurred, we call it ***Trigger-point*** (T).

Although Bayesian Block algorithm is ideal for transient detection theoretically, still people wants to know how far the BB algorithm would superior than thresholding in practice. In order to compare the two, some numerical values that can measure how effective each method was is on demand. For the data where we know the true change-point, we have two metrics: Accuracy and Reaction time. ***Accuracy*** (A) is the closeness between the estimated change-point to the true change-point. For example, in the plot (Figure 1.5), the accuracy is very high since the two blocks are separated (estimated

change-point) right where the dashed line (the true change-point) is. ***Reaction time*** (R) is measuring the distance from the trigger-point to the true change-point. So in the same plot (Figure 1.5), the distance from the dashed line to the purple line representing the reaction time for this specific case. Moreover, Instead of using time to measure the distance in between two distinct data points, we adopted the number of points that occurred in between. Because the frequency of data points can vary widely crossing different datasets. So, how many data points far away from the true change-point becomes more intuitive.

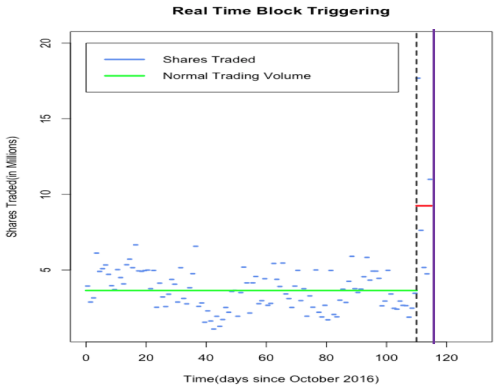


Figure 7.2: Illustration of Accuracy and Reaction time

7.3.1 Compare BB and Threshold by Accuracy

Once we have the metrics, first crucial thing we have to face is to pre-set some "fair" parameters, p,w for the Threshold algorithm as well as c for the BB algorithm.
The procedure of our experiment on simulated datasets is: First, randomly select 20 simulated datasets with same pattern. Same pattern at here means all datasets has a change from a lower constant density to a higher constant density (one could also assign a random number for the length of each potential blocks).
Then, select the candidate parameters, c in BB/ p,w in Threshold for each datasets by maximize the Accuracy (A) within a big range. In our experiment, c is selected from 100

values that are uniformly distributed in a range of (1, 200); w is selected from 10 values that are uniformly distributed in a range of (1,100); p is selected from 10 values that are uniformly distributed in a range of (0.1, 0.9). So, for each datasets, the candidate pair (w,p) for Threshold algorithm is selected from total 100 pairs. That is the same rate as c for BB algorithm.

Next, average those candidates and set it to be our "best guess" of parameters for each algorithm. Since each simulated datasets gives us same information about how good the parameters affect A (no extra weight could be added here). So, we use moment method of estimation, that's c and (w, p) for BB and Thresholding respectively.

Finally, use those "best guess" for each algorithm, test other 5 new simulated datasets with 5 replicated runs. This 5 new simulated datasets, basically was generated simultaneously with those 20 simulated datasets which has set at the beginning. So, they would hold same pattern which had a change from lower intensity to higher intensity for each datasets.

Note: Accuracy is defined as the inverse of the distance between estimated change-point to the true change-point. If the estimated change-point is exactly the true change-point, we assign the maximum value 1 to accuracy in order to avoid the existence of zero denominator.

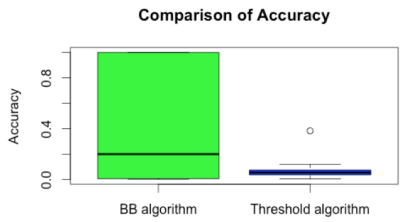


Figure 7.3: Comparison of BB v.s Threshold by Accuracy

Result:

The result shows: there are 32% of the time Accuracy can reach the value of 1 for the BB algorithm. That means some time or very often the change-point estimated by

BB algorithm can exactly hit the true change-point. In the contrast, for the Threshold algorithm, the maximum value of accuracy in the test of 5 simulated datasets with 5 replications still less than 0.2. That means, the most accurate estimation of change-point calculated by Threshold is at least 5 points away from the true change-point. Unfortunately, the accuracy calculated by BB algorithm is not always high (close or equal to 1), but some times could be very low as 0.001 (estimated change-point 1000 points away from the true change-point). As the box plot (Figure 1.6) shows, the accuracy calculated by BB algorithm could vary widely from 0 to 1, whereas the accuracy calculated by Threshold algorithm is constricted within 0 to 0.2. Overall, even the variance of accuracy is big when using the BB algorithm, still on average BB algorithm is more accurate than Threshold algorithm.

7.3.2 Compare BB and Threshold by Reaction Time

Beside Accuracy, the superior algorithm should be more sensitive, reacting faster. Thus, we present Reaction time which could welly illustrate the sensitivity for each algorithm. The procedure of this part of design is as below:

First, randomly select 20 simulated datasets with same pattern. Same pattern at here means all datasets has a change from a lower constant density to a higher constant density. We also assigned a random number for the length of each potential blocks for each simulated datasets, see how reaction speed was.

Then, select the candidate parameters, c in BB/ p,w in Threshold for each datasets by minimize the Reaction time(R) within a big range. In our experiment, c is selected from 100 values that are uniformly distributed in a range of (1, 200); w is selected from 10 values that are uniformly distributed in a range of (1,100); p is selected from 10 values that are uniformly distributed in a range of (0.1, 0.9). So, for each datasets, the candidate pair (w,p)'s for Threshold algorithm is selected from total 100 pairs. That is as same rate as c for BB algorithm.

Next, average those candidates and set it to be our "best guess" of parameters for each algorithm. Since each simulated datasets gives us same information about how good the parameters affect R (no extra weight could be added here). So, we use moment method

of estimation, that's c and (w, p) for BB and Thresholding respectively.
Finally, use those "best guess" for each algorithm, test other 5 new simulated datasets with 5 replicated runs. This 5 new simulated datasets, basically was generated simultaneously with those 20 simulated datasets which has set at the beginning. So, they would hold same pattern, had a change from lower intensity to higher intensity.
Note: Reaction time is defined as the subtraction from trigger point to true change-point. If Trigger point comes before True change-point (i.e. it was triggered by noise), reassign the length of the dataset (biggest distance between two data points) to reaction time (R). Since this maximum R value could represent the worst situation when the algorithm was too sensitive that falsely report a changing of intensity had occurred.

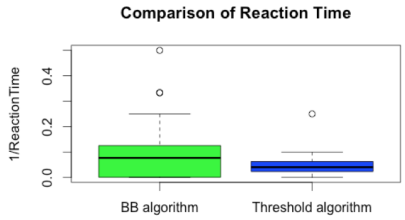


Figure 7.4: Comparison of BB v.s Threshold by Reaction time

Result:
As the result shows, the trigger-point could be as fast as 2 points away from the true change-point in BB algorithm, whereas in Thresholding algorithm the best situation within 5 testing datasets with 5 simulations is 4 points away. In addition, from the information revealed in the box-plot (Figure 7), we know that on average BB algorithm reacting more faster than Thresholding algorithm. Although the variance of R is relatively big for BB algorithm compare to Thresholding. But, since the vertical axis measures 1/R (Higher means faster), so the concentration of 1/R in Thresholding means reaction time basically all higher than some value (10 points away in this case), whereas the dispersion of 1/R in BB algorithm Actually presenting the sensitivity were sometimes better sometimes worse. Moreover, there are 10 negative R values were produced

by BB algorithm (that's 40% of the time), and only 2 negative R values were produced by Threshold algorithm (that's 8%). Many negative values of R values will drag my box plot close to 0. Without consider those falsely reported values, the 1/R in BB algorithm would be more concentrated in a higher position (performs better). But we can not discard those important information. Actually, less negative values in Threshold algorithm means by carefully select the value of parameters (w,p) could effectively reduce the rate of falsely report. On the other hand, by average those best c's in BB algorithm may dangerously lead the system become too sensitive.

7.4 Applications of Bayesian Block Method to Real Datasets

The main idea of this part is to employ generalized Bayesian blocks algorithm to model real-time data with the ultimate goal being to detect the change points of real world events intensities. Those intensities come in the form of occurrence of events. We would like to fit the shape of those intensities over time and capture the change points. The generalized Bayesian blocks algorithms can model differently shaped blocks. Generally, we can capture patterns of piecewise constant blocks, constant plus linear blocks, and gamma ray bursts in real datasets.

7.4.1 Piecewise Constant Blocks

In this case, we assume that the intensity of events follows some underlying pattern that varies over time. And we might expect to see the intensity start off constant as background noise, then a new constant block of intensity at around the real change point, when the intensity change in fact.

Dataset: United Airlines, Inc. (UAL) daily historical volumes
Range: From 10/20/2016 to 5/15/2017

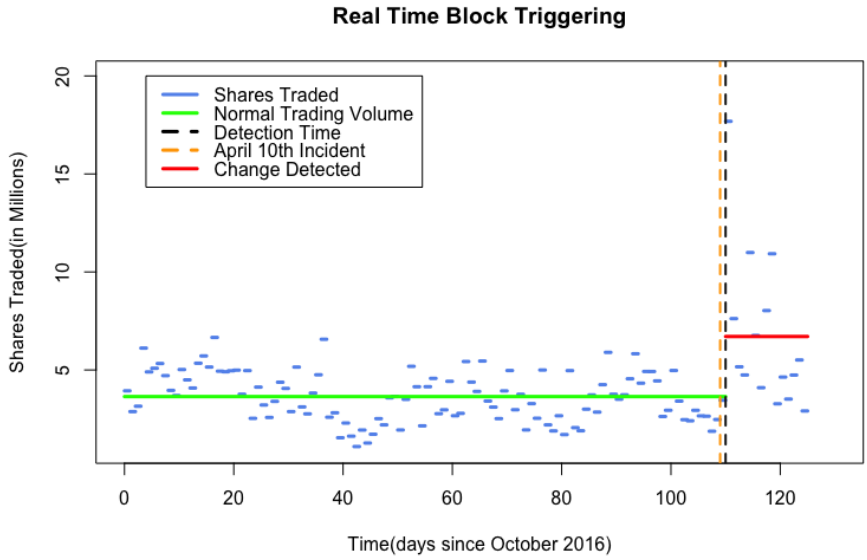


Figure 7.5: United Airline Daily Historical Volumes

This dataset describe United Airlines, Inc. (UAL) daily historical volumes. Data collected is from October 20 2016 to May 15 2017.

On April 10 2017, it went beyond the typical nightmares of travelers on a United Airline overbooked flight. And United Airlines complied with rules regarding overbooking. The video of the physical confrontation and the passenger's anguished protests spread rapidly as people criticized the airline's tactics. The shocking scene raised questions about the common practice of overbooking and how far airlines will go to sell all of their seats. Particularly annoying was that the UA was looking for extra seats for some of its employees. And this made UA stock become aggressive and high in portfolio risk. So share holders wanted to sell what they had, and meanwhile, some agency bought them at low price. This is why the daily volume went up.

As We know the fact, some actual change point occurred, we would like to test the Bayesian Block algorithm on it and see the performance. At previous days, the volumes

did not show a lot fluctuation, as UA is a steady and growing company. And there was a dramatic jump around the orange dot line, April 10 2017. Generalized Bayesian Block algorithm detected the change point at the black dot line and regarded the data after April 10 2017 as a new constant block. Therefore, it might be worthy to apply generalized Bayesian Block algorithm to more real stock market data to detect peaks and troughs.

7.4.2 Piecewise Constant plus Linear Blocks

Dataset: Synchronized Brainwave Recordings with a Audio-Visual Stimulus(from UC Berkeley)

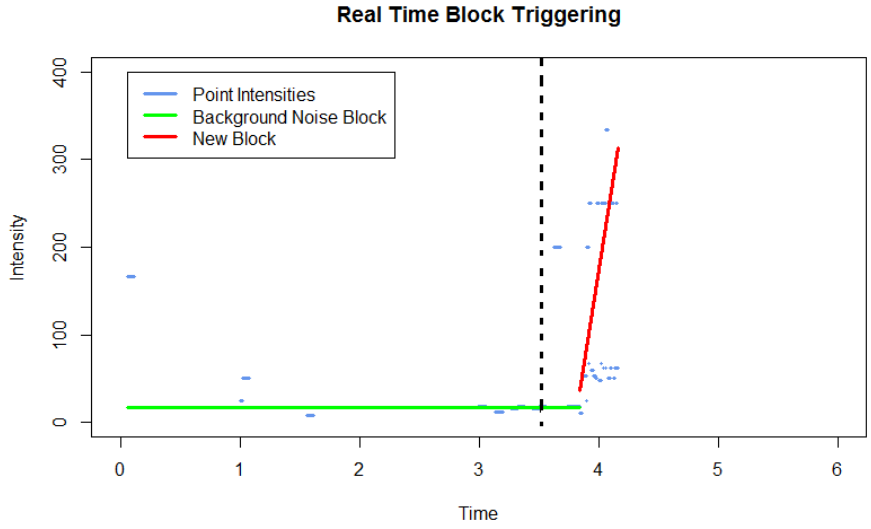


Figure 7.6: Brain Wave Signal Frequency

The data is from one UC Berkeley study, the Synchronized Brainwave Recordings with an Audio-Visual Stimulus. Each data point indicates the timestamp of each Neuron signal received.

The researcher showed some videos to a group of students and recorded student's brain activities. We focused on the brainwave signal frequency for one of the students. At first, the brainwave signal frequency remained flat at a low level. Then the student became interested in the stimulus, and brain activity increased linearly. The Bayesian Block algorithm captured that linear change and fitted it as one linear block. Also the change point, which algorithm detected, is close to the real trigger point.

7.4.3 Gamma Ray Bursts

Dataset: Gamma Ray Burst – BATSE Trigger 551

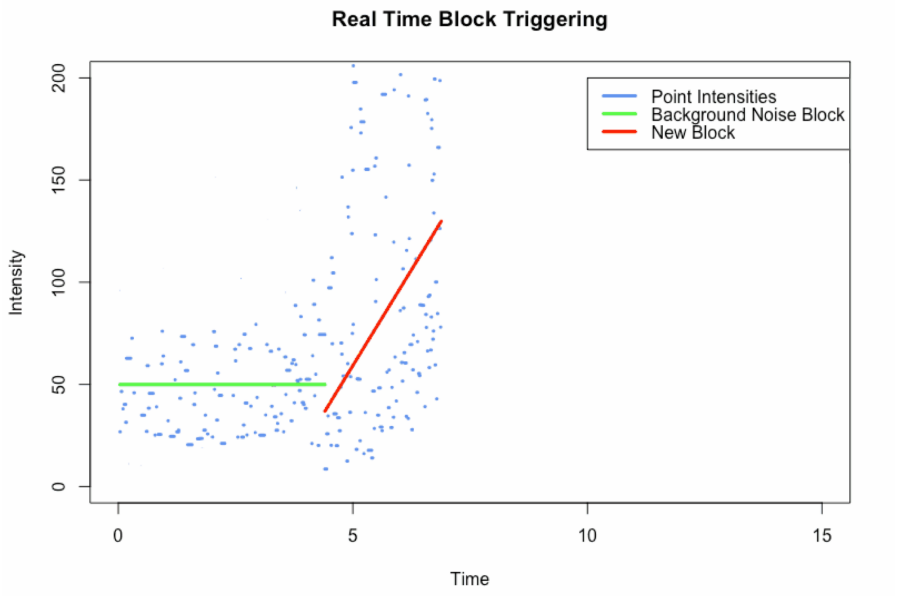


Figure 7.7: Brain Wave Signal Frequency

Data we used is one sequence of Gamma Ray Bursts data, BATSE Trigger 551. And the intensity increases dramatically when a gamma ray burst signal is detected.

The Bayesian Blocks algorithm detected the background noise as one constant block. Then once the first intensity spike occurred, at around the 4th decisecond here, the Bayesian Blocks algorithm determined that the optimal partition would have all new data in a new linear block. For future work, we would like to apply exponential shaped fitting method to capture the patterns and change point in real Gamma Ray Bursts data.

Bibliography

[1] B Alberts, A Johnson, J Lewis, M Raff, K Roberts, and P Walter. Molecular biology of the cell (6th ed.). In *International Conference on Geometric Modeling and Processing*, pages Chapter 4: DNA, Chromosomes and Genomes. Garland. ISBN 9780815344322, 2014.

[2] et al. Bachman, Ammie N. Altered methylation in gene-specific and gc-rich regions of dna is progressive and nonrandom during promotion of skin tumorigenesis. *Toxicological Sciences*, 91(2):406–418, 2006.

[3] Gary A. Churchill. Hidden markov chains and the analysis of genome structure. *Computers & Chemistry*, 16(2):107–115, 1992.

[4] Mark de Berg, Otfried Cheong, Marc van Krevald, and Mark Overmas. Computational geometry algorithms and application. In *Third Edition*, 2008.

[5] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. Biological sequence analysis. In *Eleventh Edition*, 2006.

[6] A. W. F. Edwards and L. L. Cavalli-Sforza. A method for cluster analysis. *Biometrics*, 21:362–375, 1965.

[7] Brad Jackson, Jeffrey D Scargle, David Barnes, Sundararajan Arabhi, Alina Alt, Peter Gioumousis, Elyus Gwin, Paungkaew Sangtrakulcharoen, Linda Tan, and Tun Tao Tsai. An algorithm for optimal partitioning of data on an interval. *IEEE Signal Processing Letters*, 12(2):105–108, 2005.

[8] Peter A. Jones and Peter W. Laird. Cancer-epigenetics comes of age. *Nature genetics*, 21(2):163–167, 1999.

[9] Kees Jong, Elena Marchiori, Aad van der Vaart, Bauke Ylstra, Marjan Weiss, and Gerrit Meijer. Chromosomal breakpoint detection in human cancer. *Applications of Evolutionary Computing*, 2611(2003):54–65, 2003.

[10] Rebecca Killick and Idris Eckley. changepoint: An r package for changepoint analysis. *Journal of Statistical Software*, 58(3):1–19, 2014.

[11] Rebecca Killick, Paul Fearnhead, and Idris A Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012.

[12] William A Massey, Geraldine A Parker, and Ward Whitt. Estimating the parameters of a nonhomogeneous poisson process with linear rate. *Telecommunication Systems*, 5(2):361–388, 1996.

[13] J Norris, J Bonnell, D Kazanas, J Scargle, J Hakkila, and T Giblins. Long-lag, wide-pulse gamma ray bursts. *The Astrophysical Journal*, 627(July 1):324–245, 2005.

[14] Jeffrey D Scargle, Jay P Norris, Brad Jackson, and James Chiang. Studies in astronomical time series analysis. vi. bayesian block representations. *The Astrophysical Journal*, 764(2):167, 2013.

[15] A. J. Scott and M. Knott. A cluster analysis method for grouping means in the analysis of variance. *Biometrics*, 30:507–512, 1974.

[16] William Fraser Tompkins. Applications of likelihood analysis in gamma-ray astrophysics. *arXiv preprint astro-ph/0202141*, 2002.

[17] S.S. Wilks. The large-sample distribution of the likelihood ratio for testing composite hypotheses. *The Annals of Mathematical Statistics*, 9(1):60–62, 1938.

[18] Dong-Ming Yan, Wenping Wang, Bruno Lévy, and Yang Liu. Efficient computation of 3d clipped voronoi diagram. In *International Conference on Geometric Modeling and Processing*, pages 269–282. Springer, 2010.