Chapter 1

Generalized Block Shapes

Ryan Shiroma, Kristel Lenk, 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 on the errors changing over time (heteroskedasticity), a weighted least squares regression approach can be used.

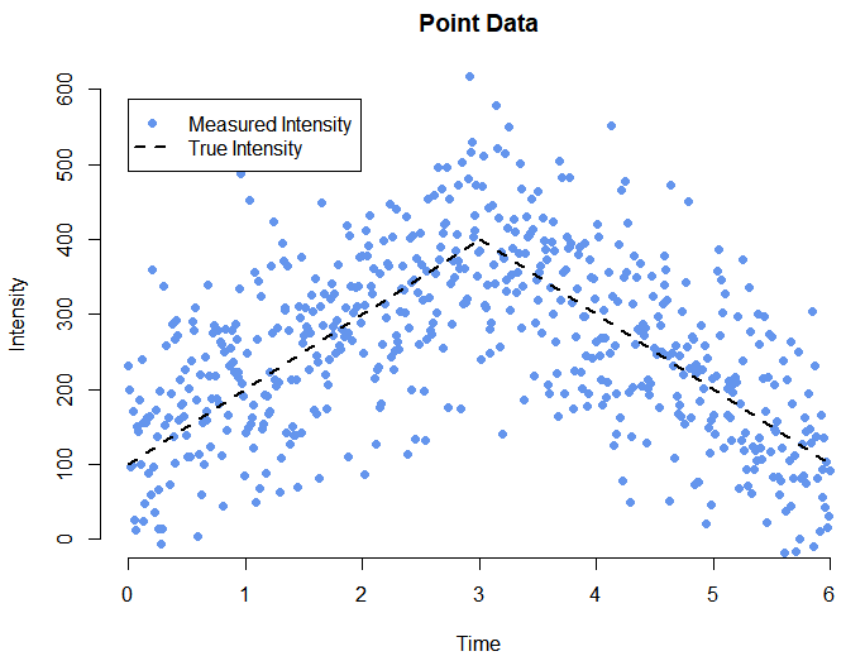

Figure 6.1: An example of point data. Measurements here are recorded every 0.1 time units.

Binned Data

The second data mode we use is **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 to estimate what this true intensity function might be.

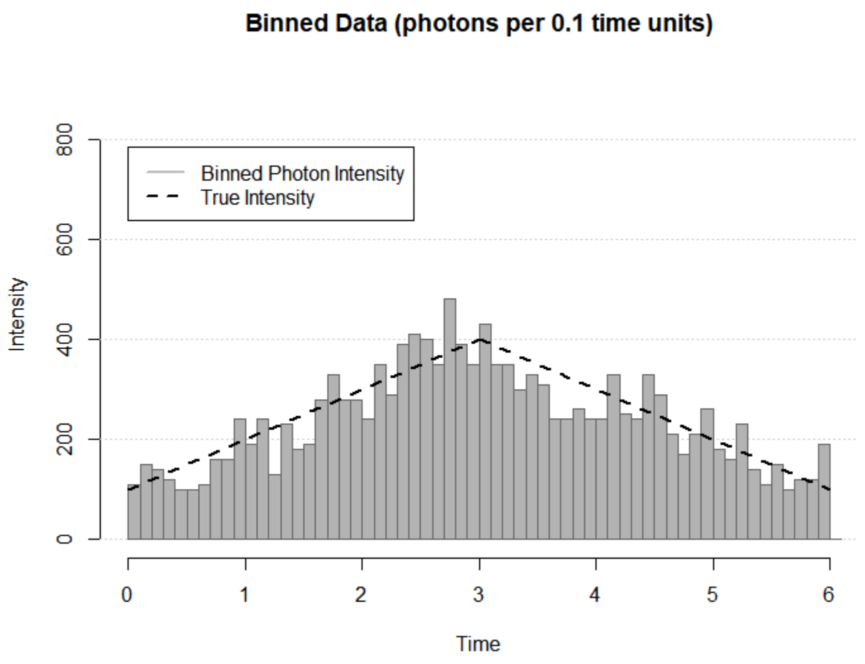

Figure 6.2: An example of binned data

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. Since each point represents a single photon, it has no idea of intensity associated with it. Therefore we assign an estimated "intensity" corresponding to each photon by taking the reciprical of the length of time between arriving photons, which is called the **inter-arrival time**. This type of data can be assumed to follow some form of piecewise non-homogeneous Poisson processes where the intensity, λ represents the true intensity at time, t. We will discuss in more detail how we model this intensity in 6.2. We can also think of this as time-to-spill data where $k = 1$, or binned data where each bin has at most one photon.

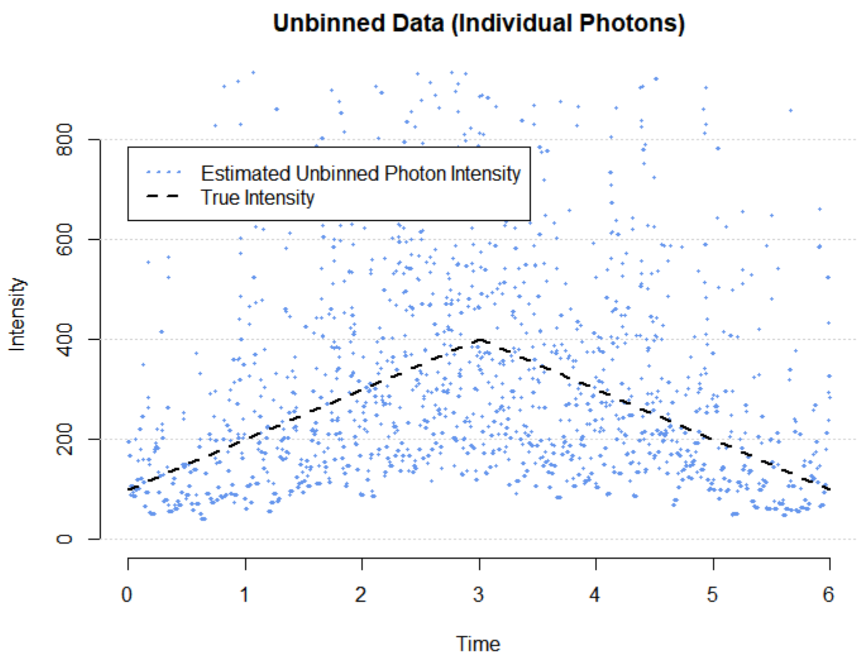

Figure 6.3: An example of unbinned data. Note the difference between unbinned data and point data. Standard least squares regression techniques will not work on raw Poisson process data.

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. The total number of photons in a bin follows a Poisson distribution where the parameter λ is the intensity for that bin's 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 using one method of fitting. Or in general, all Poisson based data can be fitted in this way. To skip to the final model fit cost formula, skip to equation 6.15.

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

$$\begin{aligned}
\mathrm{P}(a|\lambda(t)) &= \prod_{b\in U}\mathrm{P}(b|\lambda(t)) && (6.4)\\
&= \prod_{b\in S}\mathrm{P}(b|\lambda(t))\prod_{b\in U\setminus S}\mathrm{P}(b|\lambda(t)) && (6.5)\\
&= \prod_{b\in S}\big(\lambda(t)dt\big)e^{-\lambda(t)dt}\prod_{b\in U\setminus S}e^{-\lambda(t)dt} && (6.6)\\
\log(\mathrm{P}(a|\lambda(t))) &= \sum_{b\in S}\log\Big[\big(\lambda(t)dt\big)e^{-\lambda(t)dt}\Big]-\sum_{b\in U\setminus S}\lambda(t)dt && (6.7)\\
&= \sum_{b\in S}\Big[\log\big(\lambda(t)\big)+\log(dt)-(\lambda(t)dt)\Big]-\sum_{b\in U\setminus S}\lambda(t)dt && (6.8)\\
&= \sum_{b\in S}\log\big(\lambda(t)\big)+n\log(dt)-\sum_{b\in S}\lambda(t)dt && (6.9)
\end{aligned}$$

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

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

Therefore for any k blocks in the entire dataset, the total log probability for all blocks is represented as,

$$\begin{aligned}
\sum_{i=1}^{k}\log(\mathrm{P}(a_i|\lambda_i(t))) &= \sum_{i=1}^{k}\Big[\sum_{b\in S_i}\log\big(\lambda_i(t)\big)+\tau_i-\int_0^{T_i}\lambda_i(t)\mathrm{d}t\Big] && (6.11)\\
&= \sum_{i=1}^{k}\Big(\sum_{b\in S_i}\log\big(\lambda_i(t)\big)\Big)+\sum_{i=1}^{k}\tau_i-\sum_{i=1}^{k}\int_0^{T_i}\lambda_i(t)\mathrm{d}t && (6.12)
\end{aligned}$$

Let $\ell(\theta_i) = \log(\mathrm{P}(a_i|\lambda_i(t))$ θ_i *are the intensity function parameters of the ith block*

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

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

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

$$\underset{\theta_1 \ldots \theta_k}{\operatorname{argmax}} \sum_{i=1}^{k} \ell^*(\theta_i) \tag{6.14}$$

However, without any regularization, maximizing this quantity will result in $k = N$, ie. one data point per block. Since the goal is to find *blocks* of photons, we want $k << N$. So to penalize small block sizes, we incorporate a penalization parameter, ncp_{prior} for each additional block[14].

$$\underset{\theta_1 \ldots \theta_k}{\operatorname{argmax}} \sum_{i=1}^{k} \left(\ell^*(\theta_i) - ncp_{prior} \right) \tag{6.15}$$

This is equivalent to adding a prior distribution on the number of points per block. Scargle et al. derived this penalized likelihood function using a prior distribution on n, the number of points in the block. Now that we have a function to maximize, we'll need to find the optimal combination of intensity functions for the given data. Our approach is to restrict the choices of intensity function "shapes" and then calculate the maximum likelihood estimates for all combinations of blocks and shapes. Obviously calculating every possible combination would involve an extremely large number of cost computations. We therefore use the dynamic programming optimal partitioning algorithm[7] combined with the PELT algorithm[11] to find the optimal combination of shapes in $\mathcal{O}(sN)$ time where s represents the number of shapes to be considered. The next section deals with the calculation of the maximum likelihood estimates for some of the relevant shapes for gamma ray burst modeling.

6.2 Relevant block shape MLE derivations

Using this model fit cost function, we can try and fit any intensity function to the data. Whichever intensity function given the optimal parameters maximizes that cost equation will be chosen as the best fit intensity function for some given subset of the data. Any function can be fit however we have identified four common functions (shapes) that are relevant to gamma ray burst modeling.

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

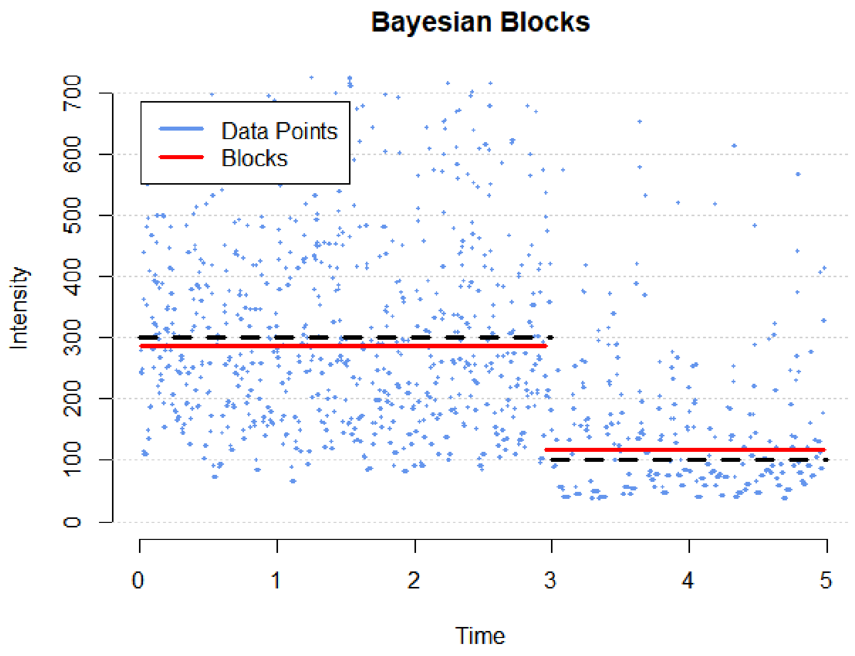

Figure 6.4: An example of a piecewise constant model

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

$$L(a) = \sum_{i=1}^{n} \log(a) - \int_0^T (a)dt \tag{6.16}$$

$$= n\log(a) - aT \tag{6.17}$$

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

$$\frac{\partial L(a)}{\partial a} = \frac{n}{a} - T \overset{\text{set}}{=} 0 \tag{6.18}$$

$$\rightarrow \hat{a}_{\text{MLE}} = \frac{n}{T} \tag{6.19}$$

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

$$L(a) = n\log\left(\frac{n}{T}\right) - \frac{n}{T}T \tag{6.20}$$

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

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

$$L_{only_constant}(a) = n\log\left(\frac{n}{T}\right) \tag{6.22}$$

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 the 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.

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

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

Newton's method is used to solve for the converged values of the estimates of a and b. Figure 6.5 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.

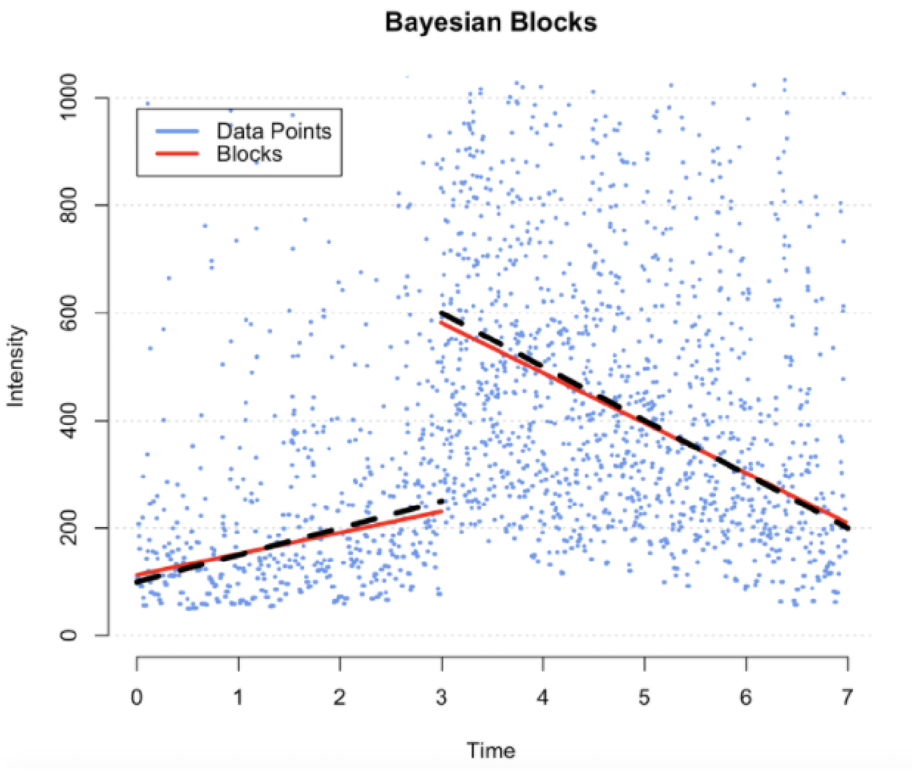

Figure 6.5: An example of a linear model.

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.

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

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

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

Newton's method is used to solve for the converged values of the estimates of a, b, and c. Figure 6.6 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.

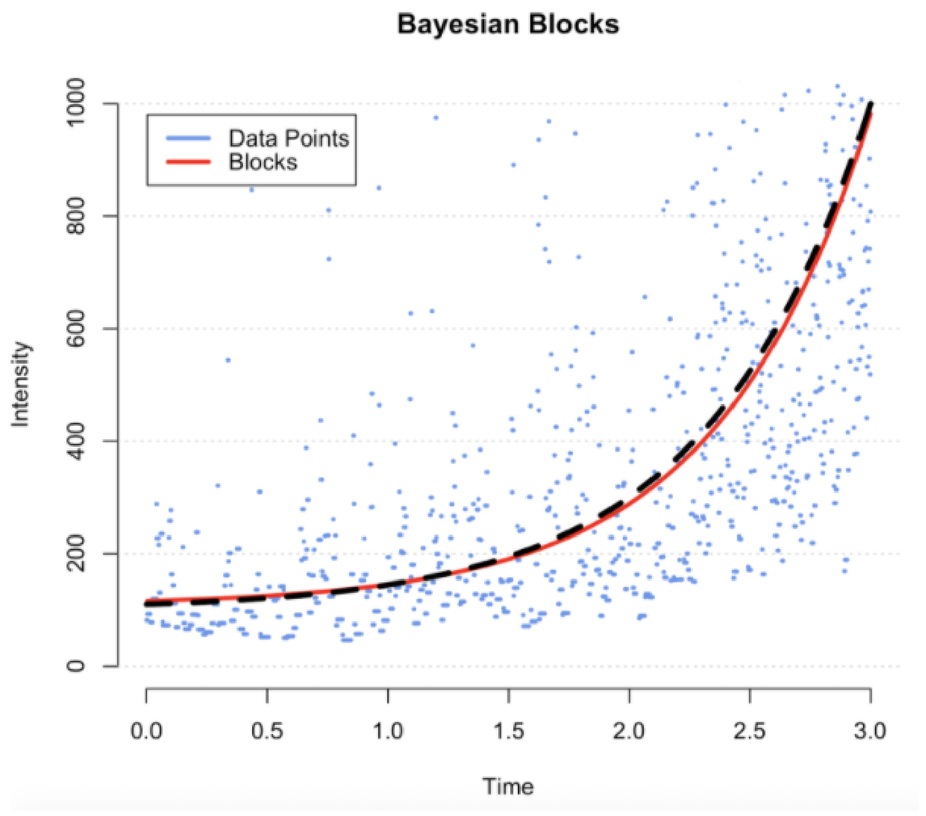

Figure 6.6: An example of an exponential model.

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.

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

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

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

Newton's method is used to solve for the converged values of the estimates of a, b, and c. Figure 6.7 illustrates an example of a power block model, where the dashed line is the true intensity of the simulated data and the solid line is the fitted block.

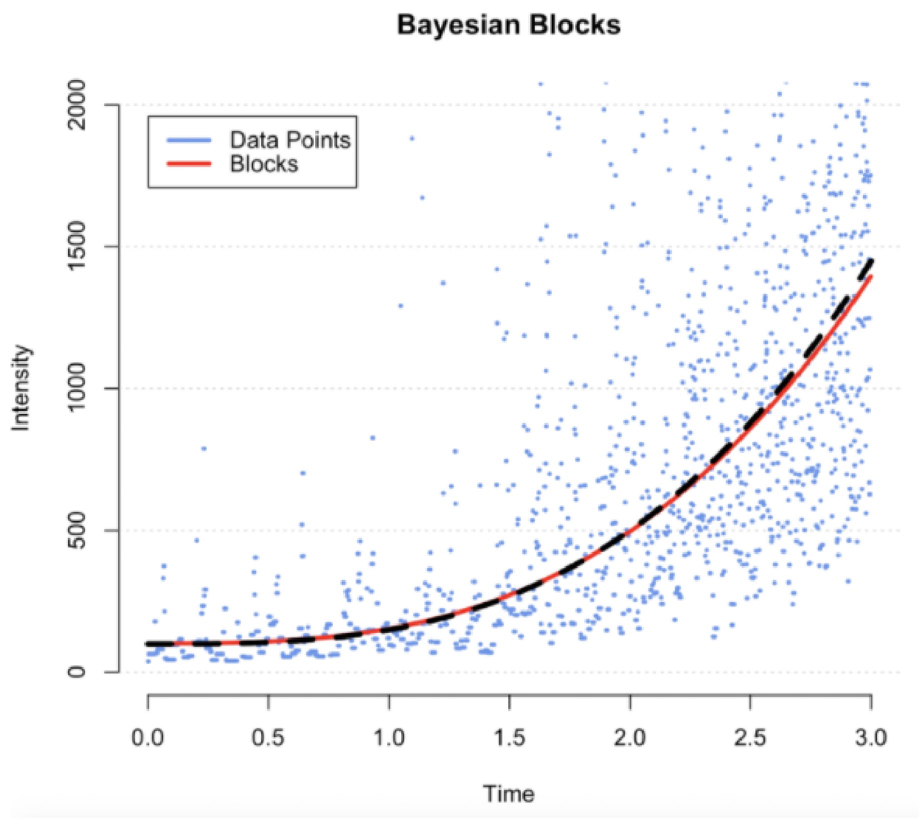

Figure 6.7: An example of a power model.

6.3 Comparing and Combining Different Block Shapes

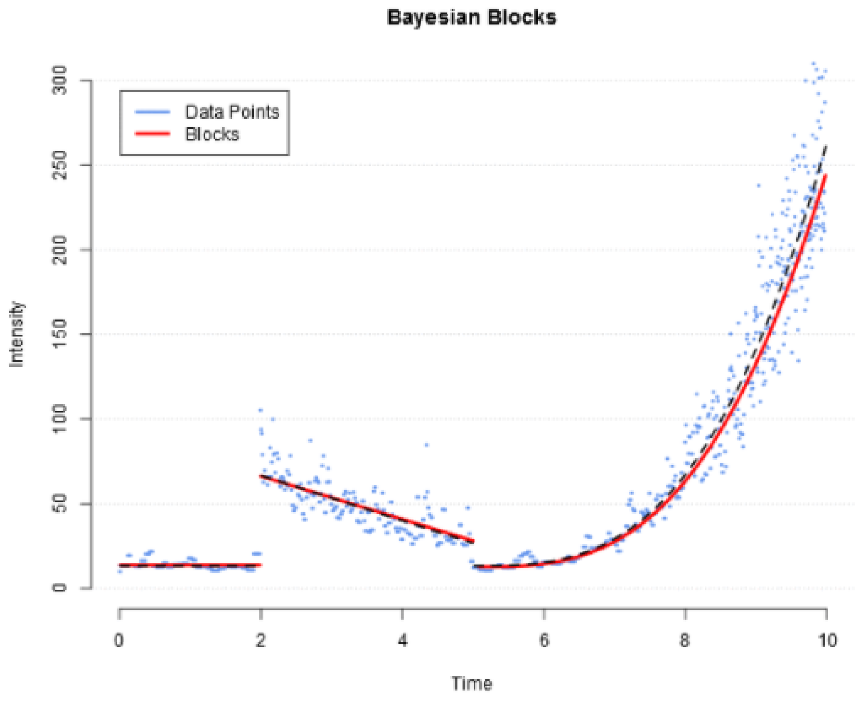

Figure 6.8: A combination of constant, linear and exponential models

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.

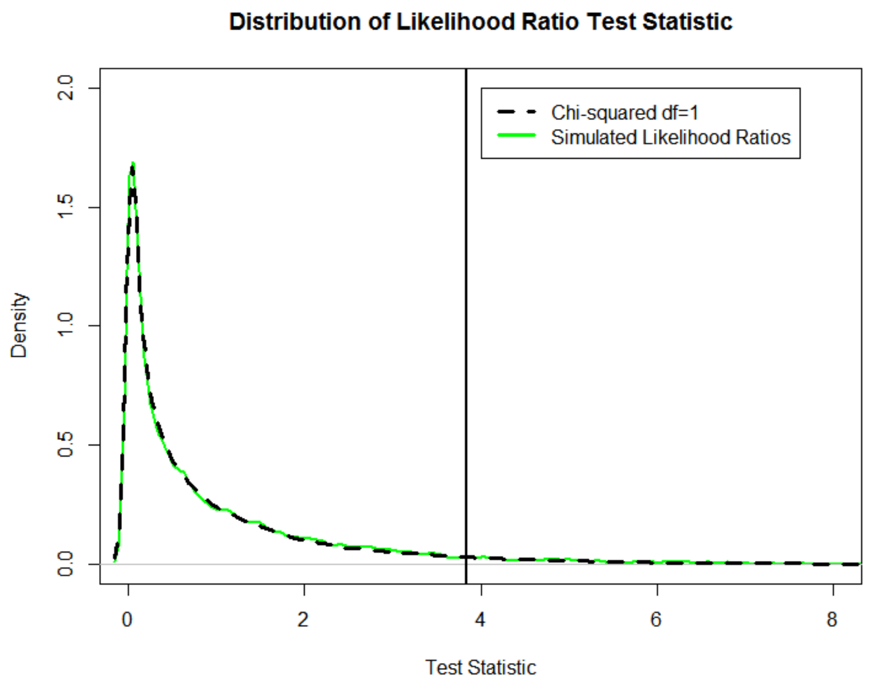

Figure 6.9: 10,000 simulations of test-statistics with n=20

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 fairly computationally expensive and thus makes this block shapes generalization somewhat impractical for large datasets. This was especially true for shapes with greater than two parameters. 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

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

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 6.10 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 25th and 75th 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 6.11 and 6.12 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 95th and 5th 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.

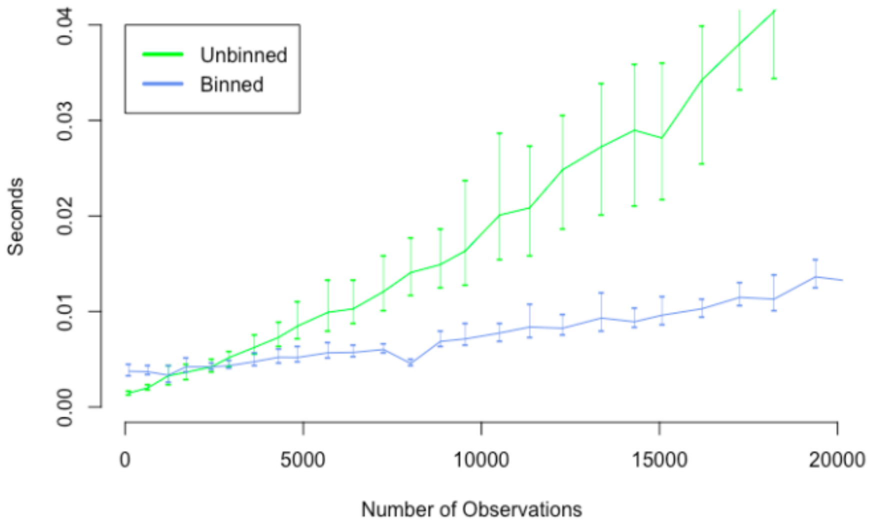

Figure 6.10: Comparing the Speed of Parameter Estimation

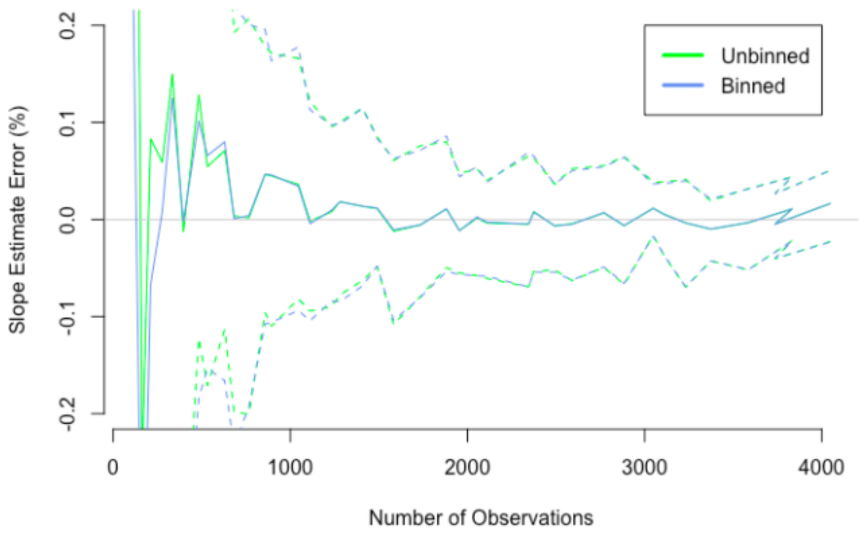

Figure 6.11: Comparing the Accuracy of Slope Estimates

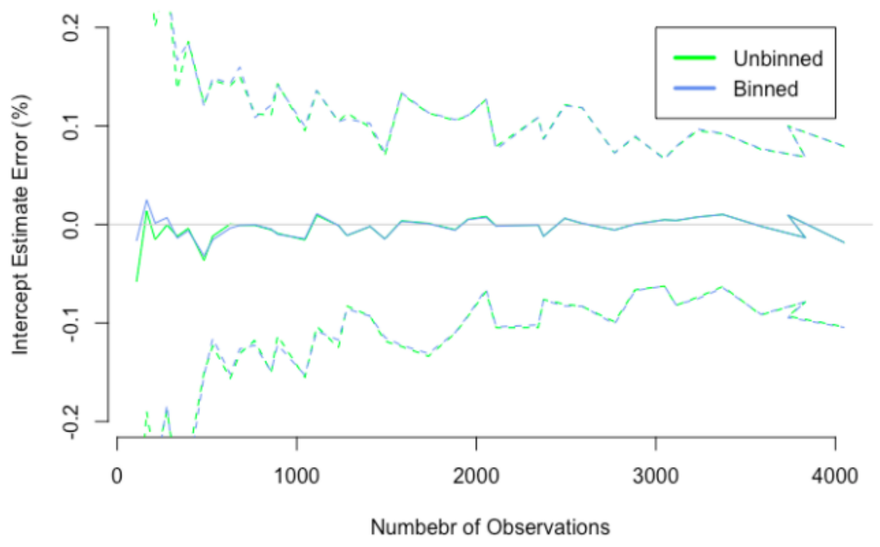

Figure 6.12: Comparing the Accuracy of Intercept Estimates

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)

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

Of course any function, linear or non-linear, can be substituted here with a weighted least squared error approach using some optimization method.

6.4.2 Parallelization of computations

One additional method to speed of computation is to parallelize the algorithm as much as possible. In the Jackson et al.[7] paper it is shown that with constant block shapes a simple closed form solution exists for the maximum likelihood estimate. This makes MLE calculations easily vectorized. Unfortunately, with other more complicated shapes, a closed form solution may not exist. For all four new shapes shown in this paper, MLE's were found using optimization methods. This cannot be vectorized but it can be parallelized onto separate processors. This kind of parallelization is also known as "embarrassingly parallel" due to its simplicity.

A second form of parallelization changes the way the algorithm run. Clearly, due to the sequential nature of the optimal partitioning algorithm, we can only process points one at a time. However, when pruning is used, it is possible to work from both the beginning and the end of the dataset at the same time on two independent processors. Once the total set of pruned points from both running algorithms equals the set of all points, the optimal partition has been found. Since a pruned point is guaranteed to never change upon continuing the algorithm, we can be sure all points have been set. The best case scenario would be if all points get pruned shortly after the two running algorithms meet in the middle. This would give us a speedup of just under 2x. The worst case scenario is when very few points are pruned and both running algorithms process all points independently. This might be the case when there are very few changepoints in the dataset. In this case, there would be no speed up in the algorithm. The only case where this kind of parallelization wouldn't work is where multiple optimal partitionings are possible in the dataset. This might cause the algorithm to find suboptimal partitionings, albeit unlikely with large time-to-event datasets.

6.5 Final Algorithm

In conclusion, our final algorithm is based off of the optimal partitioning algorithm with PELT implemented. The main differences are:

- Instead of calculating only the cost associated with constant blocks, we also compute the cost of all other shapes dersired.
- Each shape type cost is additionally penalized depending on the number of parameters in the shape.
- The shape chosen for the model is the shape that had the highest cost overall despite shape penalties.
- The threshold for PELT pruning is set to be ncp_{prior} plus the maximum value of the shape penalty constants included in the model.

6.6 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 function fitting methods to improve overall 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. For example, we might assume that block shapes that begin close to the end of the previous block is more likely.
- Run simulations on the parallelized algorithm

Chapter 2

Real-time Analysis

Stefanie Deo, Fangzhou Shen, Jiaxing Ren, Lu Liu

7.1 Applications to Real-time analysis: An Introduction

In many cases, it is sufficient to analyze existing data, i.e. data that has already been collected in the past. However, in other cases, researchers are interested in detecting activity in real time. This is because there may be some important action to be taken in reaction to the specific activity, which is only useful if it happens while the activity is occurring. This could be something as simple as alerting a sales manager of a spike in online sales, in which case the resulting action would be a commensurate adjustment in orders to a supplier. On the other hand, it could also mean the reallocation of precious resources, like focusing a high-powered specialized instrument, e.g. a telescope, to a specific area of interest. For example, in astronomy, a scientist may be observing several parts of the sky which are "blank fields", where there are no objects 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. When a transient is detected in a blank field, it is likely researchers will want to pay more attention to that area of space.

Figure 7.1 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.

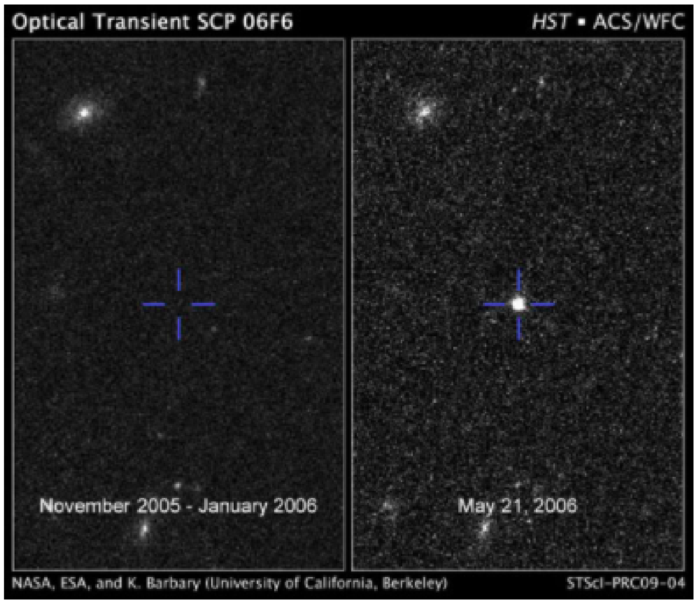
Optical Transient SCP 06F6
HST • ACS/WFC
November 2005 - January 2006
May 21, 2006
NASA, ESA, and K. Barbary (University of California, Berkeley)
STScI-PRC09-04

Figure 7.1: Optical Transient SCP 06F6

The left part of Figure 7.1, specifically the section marked by the blue reticle, shows a blank field. For this situation, we would expect only noise in the intensity data, something similar to the left area of the plot below (Figure 7.2). When the transient appears, the signal might look something like the burst of intensity as shown in the middle of the plot.

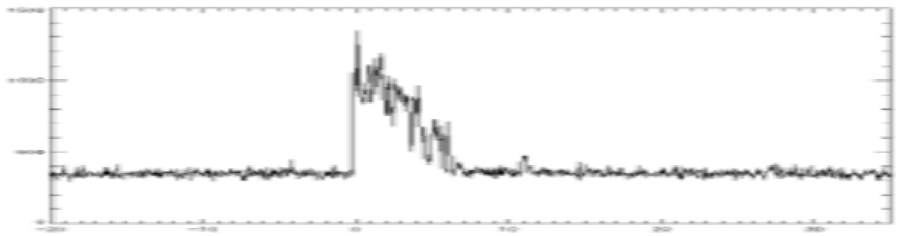

Figure 7.2: Gamma Ray Burst

7.2 Bayesian Block Triggers vs Thresholding Triggers

7.2.1 Thresholding

The current method of detecting transients is by setting a threshold value. The telescope will usually detect a low and relatively steady incoming signal stream, which is just noise. 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. Furthermore, one also needs to specify a window of time over which an average is drawn. Figure 7.3 illustrates how the thresholding algorithm works. The idea here is that we have a left window (the green box), and we have another window (the red box) that "slides" to the right whenever new data comes in. The average of the current sliding window is compared to the green window. The horizontal dashed line represents the threshold, and the horizontal blue line represents the mean of the intensity data encased by the red window.

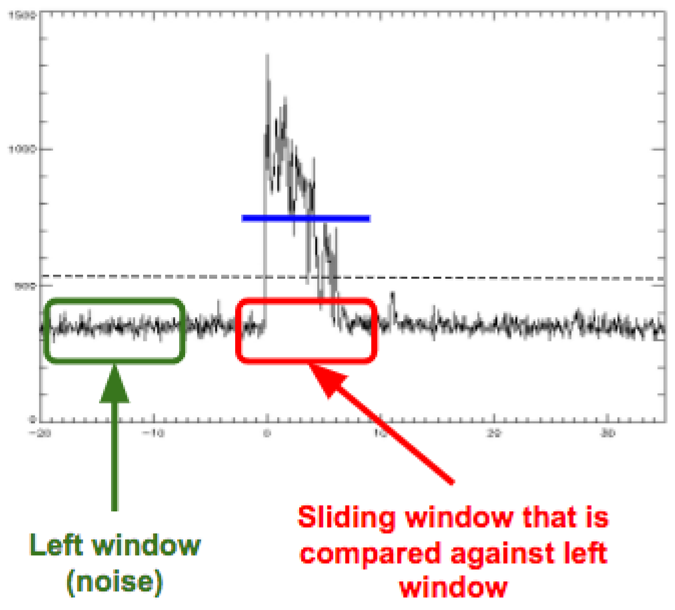

Figure 7.3: Sliding Window

Here, we can clearly see that the blue line is above the threshold, so in this case, the thresholding algorithm would have signaled that a transient had been detected.

Since there are two parameters that need to be chosen for the thresholding algorithm, the method can be quite sensitive to wrong combinations of the threshold value and window size. Some aspects of this issue can be avoided by adopting several different thresholds, but this can be problematic.

7.2.2 Bayesian Block algorithm

In contrast to the thresholding algorithm, the Bayesian Block algorithm does not rely on any pre-set threshold or pre-set window. This makes it ideal, in theory, for transient detection. When new data comes in, the algorithm recomputes the "best fit" for the data. If the algorithm determines that the intensity has not changed a lot and the best fit is still one constant block, no change point is declared. But if the algorithm determines that the intensity changed significantly and the best fit is now two blocks (in which the first block contains the noise and the second block contains the higher-intensity data), then a change point is declared.

To illustrate, in Figure 7.4 the green line represents the first block which is the best fit block for the background noise. The Bayesian Block algorithm detected that the intensity had shifted significantly at around the 5th second, increasing linearly from the level of background noise. Thus, the new best fit would involve a second linear block, represented by the red line.

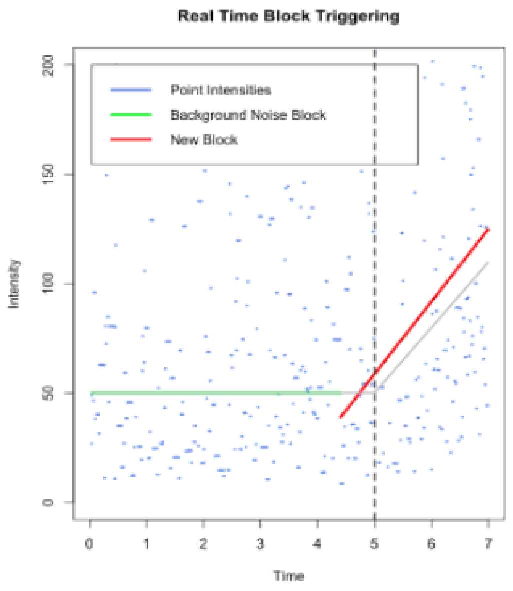

Figure 7.4: Real Time Block Triggering

7.3 Comparing Bayesian Blocks and Thresholding

We have established that the Bayesian Block algorithm should theoretically be superior to the thresholding algorithm for detecting transients. In order to put this theory to the test, we need to compare the performance of the two methods, which therefore means that we need to define some measures of performance.

Let's first clarify some definitions. The data point at which the real change of intensity takes place is called the ***true changepoint*** (Tr). The algorithm's best guess for where the changepoint occurred is referred to as the ***estimated changepoint*** (E). Finally, the data point at which the algorithm detected that a change of intensity had occurred is called the ***trigger point*** (T).

For data where we know the true changepoint, we came up with two metrics: Accuracy and Reaction Time. ***Accuracy*** (A) is the closeness of the estimated changepoint to the true changepoint. An example of high accuracy is shown in Figure 7.5. The dashed line represents the true changepoint, and the estimated changepoint is located at the point of separation between the two blocks. Here, the estimated changepoint is located right where the true changepoint is, thus this means that the algorithm was highly accurate in this case.

The second metric, ***Reaction Time*** (R), is defined as the distance from the trigger point to the true changepoint. In Figure 7.5, the horizontal purple line represents the trigger point. Thus, the distance from the dashed line to the purple line represents the reaction time for this specific case. In order to measure this distance, we used the number of points that occurred in between the two points instead of using the elapsed time between the two points. This is because the frequency of data points can vary widely across different datasets. One dataset might have several points every millisecond, while another dataset might have several points every few seconds. Therefore, using the number of data points instead of the elapsed time was a fairer way to measure distance.

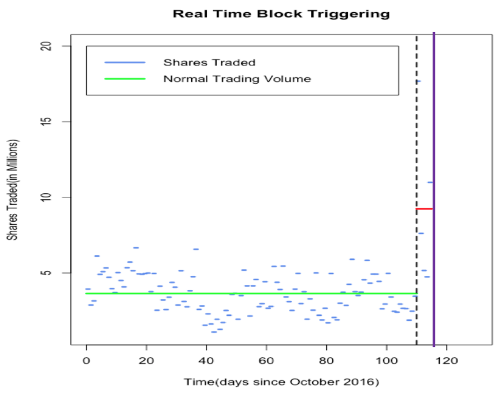

Figure 7.5: Illustration of Accuracy and Reaction time

7.3.1 Comparing the BB and Threshold Algorithms by Accuracy

To begin comparing the two algorithms, we would need to pre-set some “fair” parameters (p,w) for the Threshold algorithm as well as the c parameter for the BB algorithm. The procedure of our experiment was as follows:

First, we randomly selected 20 simulated datasets with the same pattern. Here, having the ”same pattern” meant that all the datasets had a change from a lower constant intensity to a higher constant intensity. (Note: one could also assign a random number for the length of each potential block; this could be an area of future research.)

Second, we selected the candidate parameters (c in BB; p,w in Threshold) for each dataset by maximizing the Accuracy (A) within a big range. In our experiment, c was selected from 100 values that are uniformly distributed in the range of (1, 200), w was selected from 10 values that are uniformly distributed in the range of (1, 100), and p was selected from 10 values that are uniformly distributed in the range of (0.1, 0.9). So, for each dataset, the candidate pair (w,p) for the Threshold algorithm was selected from a total of 100 pairs. As the c for the BB algorithm was selected from 100 values, these two situations were meant to simulate a real-life situation in which parameters needed to be selected for 100 different situations.

Third, we averaged the candidate parameters and set it to be our “best guess” of parameters for each algorithm. Since each simulated dataset gives us the same information about how much the parameters affect A, no extra weight needs to be added here. Thus we simply used the average, i.e. $\overline{c}$ and $(\overline{w}, \overline{p})$ for BB and Thresholding respectively.

Finally, we used those “best guesses” for each algorithm, and tested them on 5 simulated datasets with 5 replicated runs. These 5 ”new” simulated datasets were actually generated along with the 20 simulated datasets generated earlier for finding the ”best guess” parameters, and were set aside as the testing dataset group. Therefore, the 5 ”new” simulated datasets would have the same pattern, in which there was a change from a lower intensity to a higher intensity for each dataset.

Note: Accuracy is defined as the inverse of the distance between the estimated changepoint to the true changepoint in order to achieve an intuitive metric where a higher score would be better. That is,

$Accuracy = \frac{1}{|E-Tr|}$

If the estimated changepoint is exactly the true changepoint, we set accuracy to the maximum value of 1 in order to avoid the existence of a zero denominator.

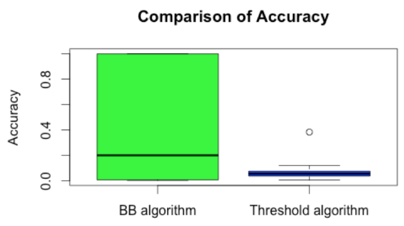

Figure 7.6: Comparison of BB v.s Threshold by Accuracy

Result:
The results in Figure 7.6 show that for the BB algorithm, Accuracy can reach the value of 1, i.e. attain perfect accuracy, 32% of the time. By contrast, for the Threshold algorithm, the maximum value of accuracy in the test of 5 simulated datasets with 5 replications was still less than 0.2. In effect, the most accurate estimation of the changepoint calculated by the Threshold algorithm is at least 5 points away from the true changepoint. Unfortunately, the accuracy calculated by the BB algorithm is not always high (i.e. it is not close or equal to 1 all the time); sometimes it can be as low as 0.001 (where the estimated changepoint is 1000 points away from the true changepoint). As the box plot in Figure 7.6 shows, the accuracy calculated by the BB algorithm could vary widely from 0 to 1, whereas the accuracy calculated by the Threshold algorithm is confined within 0 to 0.2. Overall, even if the variance of accuracy is big when using the BB algorithm, on average the BB algorithm is still more accurate than the Threshold algorithm.

7.3.2 Comparing the BB and Threshold Algorithms by Reaction Time

Aside from Accuracy, the ideal algorithm should be adequately sensitive, reacting quickly enough that the detection is useful to a researcher. The procedure of this part of the experiment is as follows:

First, we randomly selected 20 simulated datasets with the same pattern, the same as the experiment for Accuracy. We also assigned a random number for the length of each block for each simulated dataset, to test the reaction speed under varying circumstances.

Then, we selected the candidate parameters, c in BB/(p,w) in Threshold, for each dataset by minimizing the Reaction time (R) within a big range. In our experiment, c was selected from 100 values that are uniformly distributed in a range of (1, 200); w was selected from 10 values that are uniformly distributed in a range of (1,100); p was selected from 10 values that are uniformly distributed in a range of (0.1, 0.9). So, for each dataset, the candidate pairs of (w,p) for the Threshold algorithm was selected from a total of 100 pairs. Similarly, c was selected from 100 values for the BB algorithm.

Next, we averaged those candidates and set the averages to be our "best guess" of parameters for each algorithm. Since each simulated dataset gives the same information about how much the parameters affect R, no extra weight should be added here. Thus we simply used the average, i.e. $\overline{c}$ and $(\overline{w}, \overline{p})$ for BB and Thresholding respectively.

Finally, using those "best guesses" for each algorithm, we tested them on 5 new simulated datasets with 5 replicated runs. These 5 new simulated datasets were generated simultaneously with the 20 simulated datasets used earlier to find the best guesses. Thus, they would have same pattern of a lower intensity changing to higher intensity.

Note: Reaction time is defined as the number of points between the trigger point and the true change-point. That is,

Reaction Time = T - Tr

If the Trigger point comes before the True changepoint (i.e. it was falsely triggered by noise), we assigned the length of the dataset (the biggest distance between two data points) to Reaction Time (R). This maximum R value thus represents the worst case situation when the algorithm was either 1) too sensitive, falsely reporting that a change of intensity had occurred, or 2) not sensitive at all, in that it failed to detect any change in intensity.

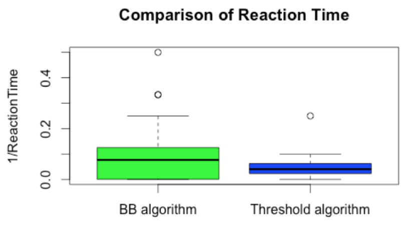

Figure 7.7: Comparison of BB vs Threshold by Reaction time

Result:

As Figure 7.7 shows, the trigger point could be as fast as 2 points away from the true change point for the BB algorithm, whereas for the Thresholding algorithm, the best situation among the 5 test datasets with 5 simulations is 4 points away. In addition, we know that on average the BB algorithm reacts faster than the Thresholding algorithm, though the variance of R is relatively larger for the BB algorithm compared to Thresholding. However, since the vertical axis measures 1/R (i.e. a higher value means a faster reaction time), the narrow band of 1/R for Thresholding means that the values for Reaction Time were basically all higher than some value (10 points away in this case), whereas the wider green band of 1/R for the BB algorithm implies that the sensitivity was sometimes better than Thresholding, but also sometimes worse. Moreover, 10 negative R values were produced by the BB algorithm (that's 40% of the time that the algorithm was triggered by noise), and only 2 negative R values were produced by the Threshold algorithm (that's 8%). Many negative values of R values will drag the box plot close to 0. Without considering 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 this important information. The fact that there are fewer negative values for the Threshold algorithm implies that carefully selecting the value of parameters (w,p) could effectively reduce the rate of false triggers. On the other hand, the averaging of the best c's for the BB algorithm may dangerously lead the system to become too sensitive.

7.3.3 Details on How the Algorithms Differ

This section will focus on differences in how the Bayesian Block and Thresholding algorithms perform.

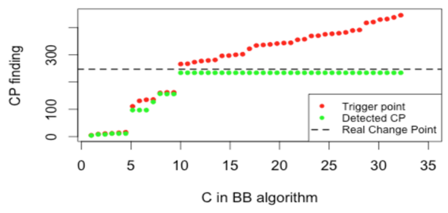

Figure 7.8: Example of Bayesian Block Performance per value of C

Figure 7.8 shows a typical example of how the Bayesian Block algorithm performance is affected by the choice of C. For low values of C, note that the green and red dots are underneath the horizontal dashed line; this means that the algorithm is too sensitive and is triggered before the real changepoint (i.e. it is triggered by noise). However, for values of C that are greater than or equal to 10, the BB algorithm is able to correctly estimate the true changepoint, as shown by the green dots that lie almost exactly on top of the horizontal dashed line. The only difference is that as the value of C grows higher than 10, it takes the algorithm longer to detect that a changepoint has occurred. This is shown by the red dots that increase linearly compared to the horizontal dashed line.

The Thresholding algorithm behaves differently. Since there are two parameters that need to be preset, the algorithm will only perform well if both w and p are close to optimal.

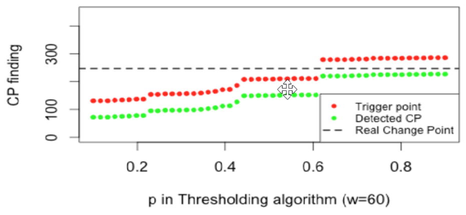

Figure 7.9: Example of Thresholding Performance per value of p given a "good" choice of w

Figure 7.9 shows an example of how the Thresholding algorithm performs for each value of p given a reasonably good choice of w. Note that values of p less than 0.6 result in the algorithm being too sensitive and being triggered by noise, but values higher than 0.6 result in accurate estimations and very fast reaction times. However, when the choice of w is suboptimal, as shown in Figure 7.10, the Thresholding algorithm is way too sensitive and is only ever triggered by noise; there is no value of p for which it makes an accurate estimation for the true changepoint.

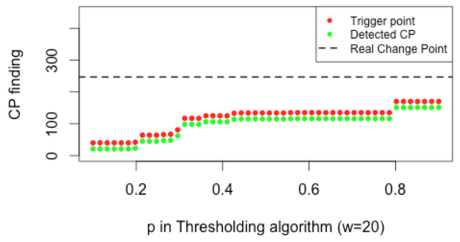

Figure 7.10: Example of Thresholding Performance per value of p given a "bad" choice of w

These differences in the behavior of the two algorithms provide some insight into why the Bayesian Block algorithm performed better on average in the experiments for Accuracy and Reaction Time.

7.4 Applications of Bayesian Block Method to Real Datasets

The main idea is to use the generalized Bayesian Blocks algorithm to detect important real world events by detecting changes of intensity in the real-time data. The generalized Bayesian Block algorithm can model different shapes of 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. We might expect to see the intensity start off constant as background noise, then a new constant block of intensity begins near the real changepoint.

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

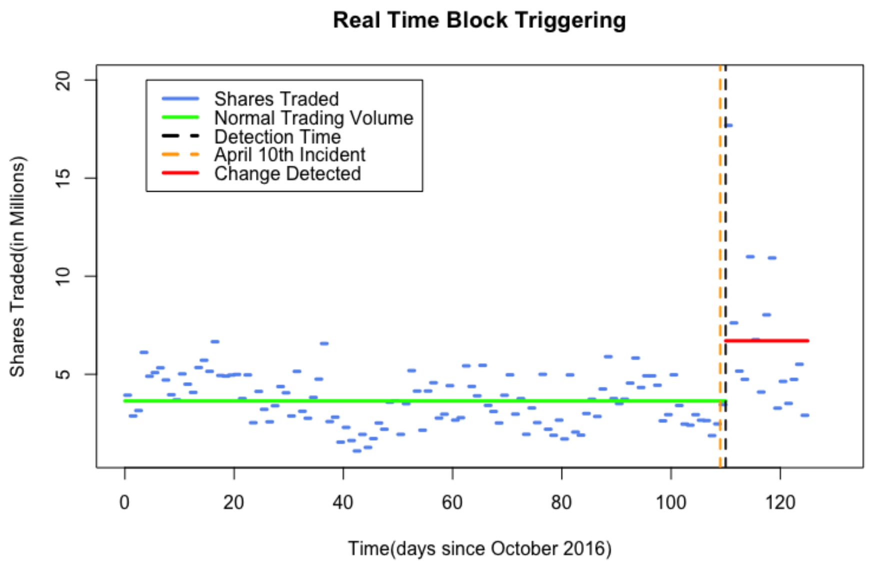

Figure 7.11: United Airline Daily Historical Volumes

Figure 7.11 shows United Airlines, Inc. (UAL) daily historical trading volumes. Data collected is from October 20 2016 to May 15 2017. On April 10 2017, United Airlines suffered a public relations nightmare as a result of an overbooked flight. The airline had asked several passengers to give up their seats for United employees who needed to be transferred ASAP, but one man refused to comply. He was forcefully removed from his seat by security personnel and ended up suffering a concussion, a broken nose, and also lost two of his front teeth. A video of the event went viral online. This caused some to deem United Airlines stock as a high portfolio risk, so many share holders decided to sell their shares. The effect of this sudden mass selling meant that the price of the share dropped, and so other people started buying the shares to capitalize on the low price. Overall, this resulted in a huge jump in the daily trading volume for the company.

In this situation, we know that some actual change point occurred. Before April 10, the volume of trade did not show a lot fluctuation. Then, there was a dramatic jump around the orange dotted line, which marks April 10, 2017. The Generalized Bayesian

Block algorithm effectively detected the change point at the black dotted line and treated the data after April 10, 2017 as a new constant block. Therefore, it could be worthwhile to apply the 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)

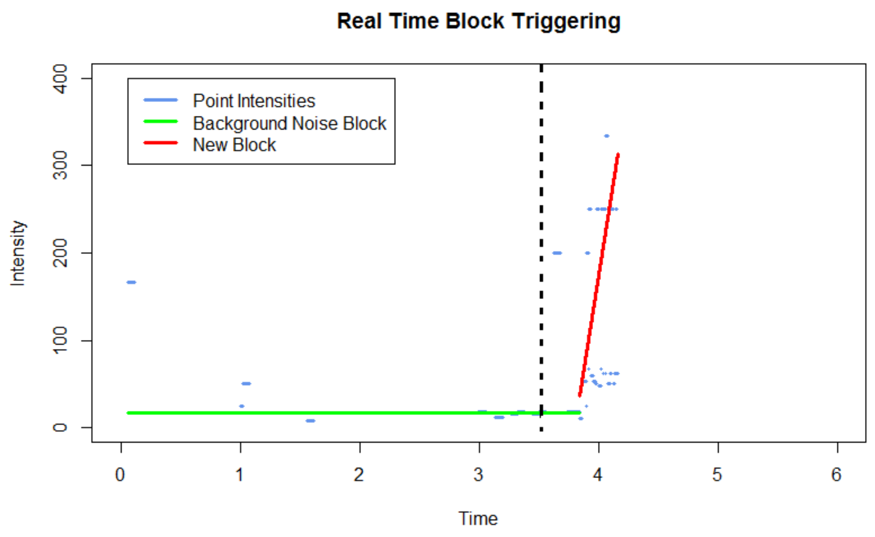

Figure 7.12: Brain Wave Signal Frequency

Figure 7.12 shows data from a UC Berkeley study, the Synchronized Brainwave Recordings with an Audio-Visual Stimulus. Each data point indicates the timestamp of each neuron signal received.

The experiment involved the researcher showing some videos to a group of students and subsequently recording the students' 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. Here, the change point estimated by the algorithm happens to be close to the trigger point.

7.4.3 Gamma Ray Bursts

Dataset: Gamma Ray Burst – BATSE Trigger 551

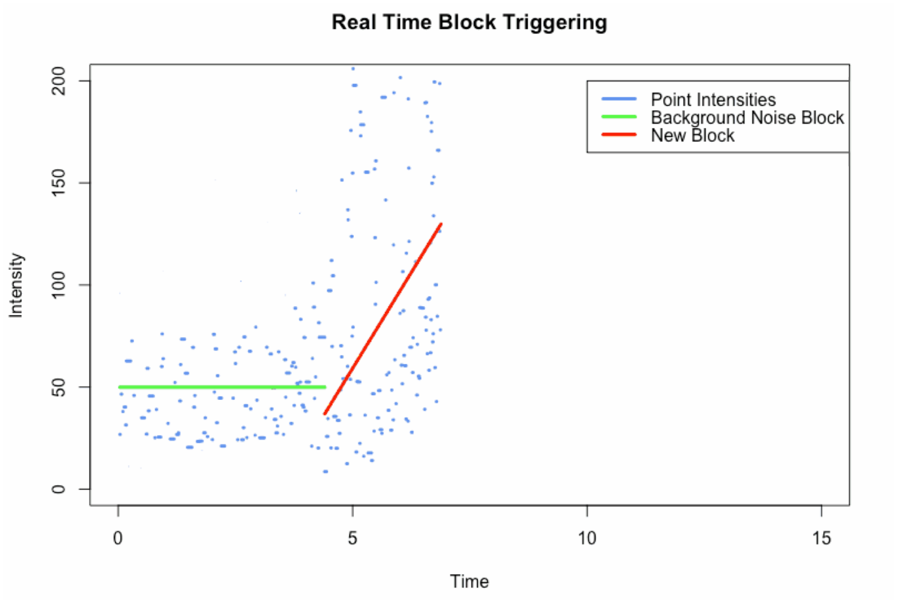

Figure 7.13: Gamma Ray Burst

Figure 7.13 shows one sequence of Gamma Ray Burst data, BATSE Trigger 551. The intensity increases dramatically when a gamma ray burst occurs. The Bayesian Block 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.

7.5 Future Work

This work could be extended by modifying the real-time algorithm to recognize exponential changes in intensity, which are more ideal for modeling real Gamma Ray Burst data. More simulations can always be run using different simulated datasets to compare the performance of the Bayesian Block algorithm to the Thresholding algorithm.

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.