Bayesian Blocks: an algorithm for histogram representation

Alice Pagano
(Dated: July 16, 2020)

The Bayesian Block algorithm can be used to improve the binning of histograms by finding the optimal segmentation of the data. The visual improvement can be dramatic, but, more importantly, this algorithm produces histograms that accurately represent the underlying distribution while being robust to statistical fluctuations. In this project, the Bayesian Blocks algorithm is implemented in R and its performances are tested with different sets of data.

I. INTRODUCTION

A. Representation of data: histograms

The data analysis goal is to identify and characterize statistically significant variations and features in observed data. In particular, histograms are used to represent distributions of data. Most of the time, a subjectively natural range and bin width is chosen, motivated mainly by obtaining a nice looking plot. Objective methods have been proposed that determine binning according to some optimization procedure such as Scott's Rule optimal for random samples of normally distributed data. Some methods takes the structure of the distribution into account but uses bins of fixed width. Others requires that each bin have similar numbers of entries, and thus the bin widths may vary, but the location of the bin edges is still chosen arbitrarily. The Bayesian Block algorithm, in contrast, allows the bin widths to vary and determines the bin edges based on the structure of the distribution.

B. Introduction to Bayesian Blocks method

The Bayesian Block algorithm was originally developed by Jeffrey D. Scargle[1,3] for applications in astronomy to addresses the problem of detecting and characterizing local variability in time series. Although it was developed for time series data, the algorithm is applicable to other forms of sequential data, or any other other independent variable. In particular, it can be used to improve the representations of histograms into ones in which the bins are not fixed and are free to be unequal in size as determined by the data.

The Bayesian Blocks algorithm is a non-parametric modeling technique for determining the optimal segmentation of a given set of univariate random variables into blocks, with each block containing consecutive data elements satisfying some well defined criterion. It operate in a bayesian framework. The goal is to separate statistically significant features from the ever-present random observational errors and thus to discover local structures in background data exploiting the full information brought by the data themself. These concepts and methods can be applied in general higher dimensional contexts. Here, however, we concentrate on one-dimensional data.

II. MAIN ASSUMPTIONS OF BAYESIAN BLOCKS METHOD

To recap, the mean idea of Bayesian Blocks is the segmentation of the data interval into variable-sized blocks, each containing consecutive data satisfying some well defined criteria. The optimal segmentation is the one that maximizes some quantification of this criterion. Now, let us discuss the main assumptions of this model.

A. Piecewise Constant Model

For this implementation of Bayesian Blocks algorithm, the range of the independent variable is divided into subintervals (called **blocks**) generally unequal in size, in which the dependent variable is modeled as constant within errors. In particular, the segmentation of the whole observation interval is described by the following parameters:

- N_{cp}, the number of change-points;
- t_k^{cp}, the change-point starting block k;
- X_k, the signal amplitude in block k;

for $k = 1, 2, \ldots, N_{blocks}$, where the number of blocks is $N_{blocks} = N_{cp} + 1$.

The set of blocks is **gapless** and **non-overlapping**, where the first block edge is defined by the first data point, and the last block edge is defined by the last data point. Even though the first data cell always starts the first block, our convention is that it is not considered a change-point. A block can contain between 1 and N data points, where the sum of the contents of all the blocks must equal N. Each block is effectively defined by two parameters: the first represents the signal amplitude, and is treated as a nuisance parameter to be determined after the change-points have been located, while the second parameter is the length of the interval spanned by the block.

This segmented representation is in the spirit of non-parametric approximation and not meant to imply that we believe the signal is actually discontinuous. The sometimes crude and blocky appearance of this model may be awkward in visualization contexts, but for deriving physically meaningful quantities it is not.

B. Fitness of Blocks

The number and the edges of the blocks are determined through optimization of a "fitness function", which is essentially a goodness-of-fit statistic dependent only on the input data and a regularization parameter. The algorithm relies on the fitness being **block-additive**, i.e.

$$F_{total} = \sum_{k=1}^{N_{blocks}} f(B_k) \tag{1}$$

where F_{total} is the total fitness of the partition for a given dataset, and $f(B_k)$ is the fitness of block k. The latter can be any convenient measure of how well a constant signal represents the data within the block. Hence, the **key idea** is that the *blocks can be treated independently*, in the sense that a block's fitness depends on its data only. The best model is found by maximizing F_{total} over all possible such partitions and there is a considerable freedom in choosing the fitness function (rely on sufficient statistics).

For instance, let us consider **event data** (i.e. series of discrete events) for which it is natural to associate one data cell with each event. The fitness function is easily obtained starting with the unbinned likelihood known as the **Cash statistics**. With a model $M(t, \theta)$, the unbinned log-likelihood reads:

$$\log L(\theta) = \sum_n \log M(t_n, \theta) - \int M(t, \theta) \mathrm{d}t \tag{2}$$

where the sum is over the events and θ represents the model parameters. Our block model is constant with a single parameter, $M(t, \lambda) = \lambda$, so for block k:

$$\log L^{(k)}(\lambda) = N^{(k)} \log \lambda - \lambda T^{(k)} \tag{3}$$

where $N^{(k)}$ is the number of events in block k and $T^{(k)}$ is the length of the block. Now, we maximize with respect to the nuisance parameter λ (height of the block):

$$\log L^{(k)}_{max} + N^{(k)} = N^{(k)}(\log N^{(k)} - \log T^{(k)}) \tag{4}$$

where the term $N^{(k)}$ is taken to the left side because its sum over the blocks is a constant (N, the total number of events) that is model independent and therefore irrelevant. Moreover, note that this fitness function is *scale invariant* ($T \to \alpha T$). The fitness of the entire partition will be then:

$$\log L = \sum_k \log L^{(k)}_{max} \tag{5}$$

C. Prior distribution for the number of blocks

The fitness described above must be modified by a penalty term for the number of blocks. We influence the number of blocks by defining a prior distribution for the number of blocks. Adjusting a parameter controlling the steepness of this prior establishes relative probabilities of smaller or larger numbers of blocks. In the usual fashion for Bayesian model selection in cases with high signal-to-noise, N_{blocks} is determined by the structure of the signal; with lower signal-to-noise, the prior becomes more and more important. In short, we are regulating not smoothness but complexity.

Let us discuss some common priors for the number of blocks:

- **Flat Prior**. Without explicitly adding any additional parameter to the fitness function, there is an implicit assumption of a uniform prior on the number of blocks between 0 and N.

- **Geometric Prior**. However, a flat prior on the number of blocks is unreasonable. Indeed, in most settings, it is much more likely a priori that $N_{blocks} \ll N$ than that $N_{blocks} \sim N$. For this reason, it is desirable to impose a prior that assigns smaller probability to a large number of blocks, and we adopt this geometric prior:

  $$P(N_{blocks}) = P_0 \gamma^{N_{blocks}} \tag{6}$$

  where P_0 is the normalization constant defined as:

  $$P_0 = \frac{1-\gamma}{1-\gamma^{N+1}} \tag{7}$$

  for $0 \leq N_{blocks} \leq N$, and zero otherwise since N_{blocks} cannot be negative or larger than the number of data cells. This form for the distribution dictates that, finding $k+1$ blocks is less likely by the constant factor γ than is finding k blocks. Thus, in almost all applications γ will be chosen < 1 to express that a smaller number of blocks is a priori more likely than a larger number.

- **Rigorous calibrated prior** (for event data). A prior calibration following a rigorous approach for event data yields:

  $$\log P(N, p_0) = 4 - \log\big(73.53 p_0 N^{-0.478}\big) \tag{8}$$

1. *Fixing the Parameter in the prior distribution*

Now, we briefly discuss the idea of the approaches for calibrating the prior. For instance, let us consider the geometric prior. An overly conservative value will suppress the detection of true change-points, while a too liberal value will lead to spurious change points (eventually reaching the limit of $N_{blocks} = N$). Hence, an optimal choice is a tradeoff between a conservative choice and a liberal choice.

A **general rule** is running the algorithm with a few different values can be enough. In general, the number of change-points is insensitive to a large range of reasonable values of your "steepness" parameter.

However, a **rigorous approach** is calibrating the prior as a function of the number of data points N and the correct detection rate p_0 on toy pure-noise experiments. This objective procedure is based on relating this parameter to the *false positive probability*. This is the relative frequency with which the algorithm falsely reports detection of a change-point in data with no signal present. It is convenient to use the complementary quantity:

$$p_0 \equiv 1 - (\text{false positive probability})$$

This number is called **correct detection rate** for single change-points, because it is the frequency with which the algorithm correctly rejects the presence of a change-point in pure noise.

III. THE ALGORITHM

For a given dataset of dimension N, the number of possible partitions (i.e. the number of ways N cells can be arranged in blocks) is 2^N. This number is exponentially large, rendering an explicit exhaustive search of partition space utterly impossible for all but very small N.

A **dynamic programming approach** (following the spirit of mathematical induction) is used to overcome the problem: beginning with the first data cell, at each step one more cell is added. More specifically:

- sort the data and start from the first one, the only possible partition is trivially optimal;
- the optimal partition is updated at each step using the information from the previous ones. For example, during iteration n (where data point n is being evaluated), the potential total fitnesses are calculated as:

$$F_{total}(n, m) = F_m + f(B_m^n) \tag{9}$$

  with $m = 1, 2, \ldots, n-1$ where F_m is the optimal fitness as determined during iteration m, and $f(B_m^n)$ is the fitness of the block bound between data points n and m;
- this potential total fitness is calculated $n-1$ times at each iteration, and the maximum of those fitnesses along with the relevant change-points are stored and used during the subsequent iterations;
- the iterations normally continue until the whole interval has been analyzed. After the final iteration, N, the change-points associated with the maximum total fitness are returned.

The beauty of the Bayesian Blocks algorithm is that it finds the optimum among all partitions without an exhaustive explicit search, which is obviously impossible for almost any value of N arising in practice. Indeed, this method guarantees that the global maximum fitness is obtained in $O(N^2)$, which is much more efficient than an exhaustive search of all 2^N potential configurations.

1. *Test computational time as a function of N*

As said, the computational time required by the Bayesian Blocks algorithm scales as $O(N^2)$. Now, we check this scaling time law by giving to the algorithm a growing number of data of normal distributed random variables with μ=0 and σ=1.

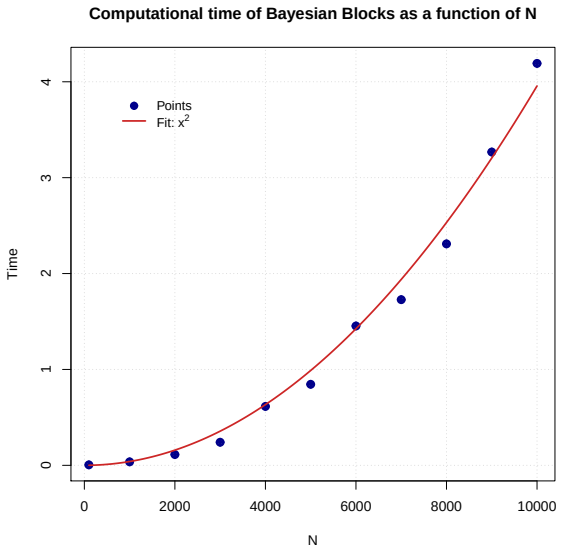

FIG. 1. Plot of computational time required to execute the algorithm as a function of N, the number of total counts in the histogram. In blue the experimental points are illustrated, while the red lines represent the second order fit. The algorithm is tested on a set of normal distributed variables with μ=0 and σ=1 with a growing N.

We make a second order fit of the time data as a function of N to verify the scaling law. It is illustrated in Fig.1. The result is consistent with what expected: the computational time required scales approximately as $O(N^2)$. Moreover, we note that for $N \sim 10^5$ the algorithm requires a lot of time. To handle the problem, for a large number of N, we can use a trick adopted for high statistics: we bin the data and treat each bin as a data point of multiplicity, which correspond to bincontent in the algorithm. This solutions is easily obtainable with the implemented code. We just need to make an histogram of the data with a wanted fixed number of edges and pass this object as data of the algorithm function.

IV. APPLICATIONS: BINNING HISTOGRAMS

Now, we test the Bayesian Blocks algorithm with different sets of data in order to evaluate its performance.

First of all, we sample a set of normal distributed random data which dimension is $N \sim 10^4$. We make an histogram of data with equal bin size and with a number of edges chosen arbitrarly. Then, we use the Bayesian

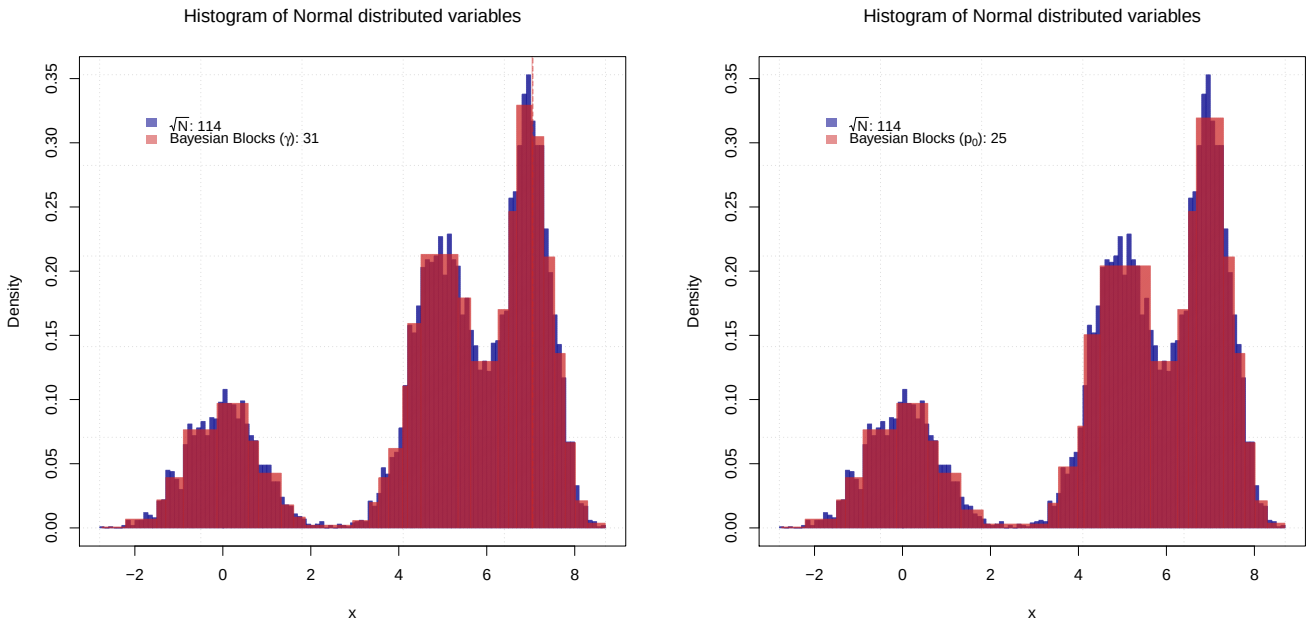

FIG. 2. Histogram of normal distributed random variables. The binning is obtained with standard rules as $\sqrt{N}$, where N is the number of total counts in the histogram, and Bayesian Blocks algorithm. In the legend, the number of edges for each method is reported.

Blocks algorithm with the two different priors (defined in Eq.6 and Eq.8). In Fig.2, we can see that the adaptive-width bins lead to a very clean representation of the important features in the data. Moreover, if we tune appropriately the prior parameter, we cannot note any difference between the results. In practice, as the length of the data becomes higher, the used prior influence less the representation of the data itself.

Now, we test the algorithm on a dataset made by a uniform noise background and three gaussian signals. In particular, we want to see if the algorithm is capable to recognize both tight and wide peaks. In Fig.3, we see that the algorithm is able to recognize all the important information of the dataset. It recognize very well the tight peak. It also recognize both the two gaussian wider peaks by distinguishing well the uniform background. In this plot we can exploit the simplicity of this algorithm: it returns with few number of edges the most important features of the dataset.

Then, we test the Bayesian Blocks algorithm with an energy spectrum dataset collected with an HPGe detector. The source is a combination of Am241, Co60 and Cs137. It was sealed up and only photons could escape from the source to be detected. Indeed, the alpha particles were absorbed by the sealing. The energy spectrum is illustrated on the left of Fig.4. We make the histogram of the data either by using a fixed number of edges and by using Bayesian Blocks algorithm with γ prior. On the right of Fig.4, we represent the histogram adding a logarithmic scale on y for a better visualization. The Bayesian Blocks algorithm highlight the main features

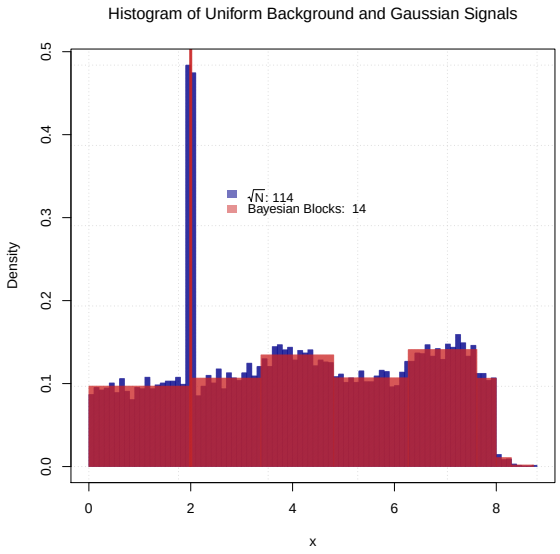

FIG. 3. Histogram of three gaussian signals in an uniform background. The binning is obtained with $\sqrt{N}$, where N is the number of total counts in the histogram, and Bayesian Blocks algorithm. In the legend, the number of edges for each method is reported.

of the energy spectrum by recognizing the main spikes of the spectrum. We note also that varying the parameter of the prior, or changing the prior function itself, does not change the result of the algorithm. It is due to

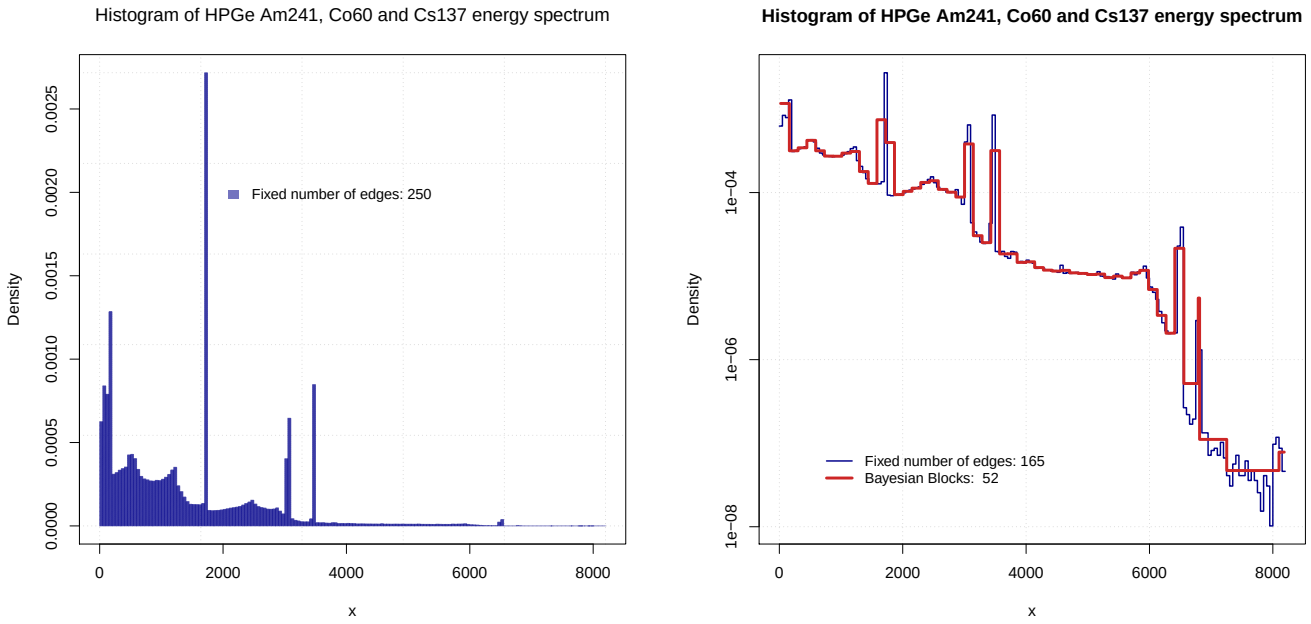

FIG. 4. On the left, plot of HPGE Am241, Co60 and Cs137 energy spectrum. On the right, it is illustrated the histogram of the energy spectrum, in logaritmic scale on y axis, obtained both with a fixed number of edges and with Bayesian Blocks method. In the legend, the number of edges for each method is reported.

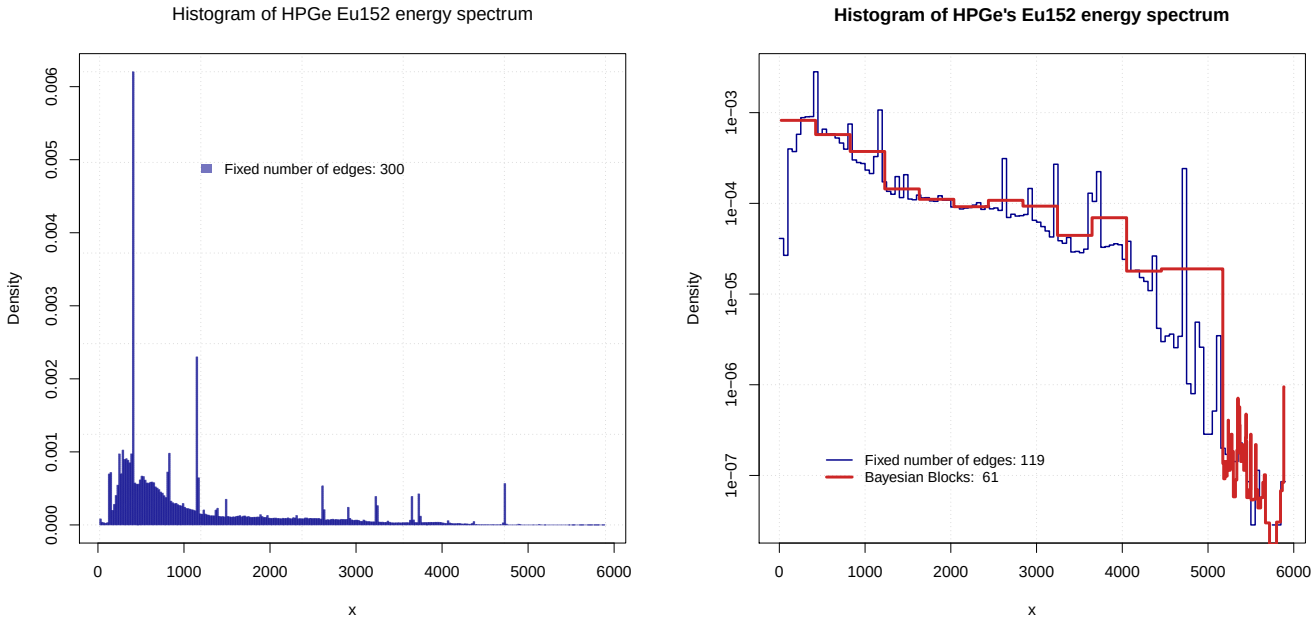

FIG. 5. On the left, histogram of HPGE Eu152 energy spectrum. On the right, it is illustrated the histogram of the energy spectrum, in logaritmic scale on y axis, obtained both with a fixed number of edges and with Bayesian Blocks method. In the legend, the number of edges for each method is reported.

the high number of data in the histogram. As said, for high statistics the prior becomes less important in the spirit of Bayesian analysis. Altought the large number of events, the algorithm is very fast because it execute the loop only for unique values of x (see implemented code). Hence, in this case we are considering only 8192 integer values with their relative weights. However, if we analyze a float dataset, the number of unique values of x

could be very large. In this case, we should use the trick previously discussed for high statistics.

Moreover, let us test the Bayesian Blocks algorithm with an energy spectrum dataset collected with a HPGe detector with a Eu152 source. The histogram of the energy spectrum for a fixed number of edges is illustrated on the left of Fig.5. We note that for this spectrum we have less statistics with respect to the previously HPGe energy spectrum dataset. However, let us test the Bayesian Blocks algorithm. Unfortunately, the algorithm does not recognize any spike for any implemented prior as we can see on the right of Fig.5 (we observe that the prior does not influence the number of change points or their position). This result is a bad representation of the Eu152 spectrum. The algorithm could fails because of the large number of spikes on this energy spectrum. Indeed, they are a lot and they are also not well distinguishable from the background. We recall that a lower signal-to-noise ratio can influence a lot the performance of the Bayesian Blocks algorithm. Moreover, the peaks are very tight, but this should not be a problem because, as we have seen in the previous examples, the algorithm is able to recognize also very tight spikes.

V. CONCLUSIONS

In conclusion, the Bayesian Blocks representation provides an objective way to enlighten the key features of a data set by imposing few preconditions as possible.

An advantage of Bayesian Blocks is that, most of the times, the adaptive-width bins lead to a very clean representation of the important features in the data. Moreover, the algorithm can center the peak of a signal in a bin, so that it will be as clear as possible compared to the background.

However, Bayesian blocks are better if you are interested in peaks than tails, as tails have few entries and therefore do not change much and have few change points. For the same reason, Bayesian blocks work better the more data you have, as the changes can be too small to make change points in small data sets. Thus, we need large statistics to correct recognize the important features of a histogram.

[1] J. D. Scargle et al., Astrophys. J. 764 (2013) 167
[2] L. Pertoldi, The Bayesian Blocks algorithm from time series analysis to histogram representation, GERDA meeting presentation, 2018.
[3] J. D. Scargle et al., Astrophys. J. 504 (1998) 405
[4] B. Pollack et al., arXiv:1708.00810