

Studies in Astronomical Time Series Analysis.
VI. Optimal Segmentation: Blocks, Triggers, and Histograms

Jeffrey D. Scargle

Space Science Division, NASA Ames Research Center

Jay Norris

Laboratory for High Energy Astrophysics
Code 661, NASA Goddard Space Flight Center

Brad Jackson

San José State University, Department of Mathematics and
Computer Science,
The Center for Applied Mathematics and Computer Science

Draft of July 11, 2006. **Warning:** this document is under construction. Notation in various sections may not yet be consistent.

Abstract

This paper addresses the problem of detecting and characterizing local variability in time series. Since such data are always corrupted by observational errors, the goal is to find statistically any significant variations and ignore the inevitable random noise fluctuations. We present a simple nonparametric modeling technique and an algorithm implementing it—an improved and generalized version of *Bayesian Blocks* [Scargle 1998]—that finds the optimal partitioning of the observation interval. The structure of the algorithm allows it to be used in either a real-time, *trigger* mode, or a *retrospective* mode. The necessary maximum likelihood or marginal posterior functions to measure model fitness are presented for points, binned counts, and measurements at arbitrary times with a known error distribution. The same algorithm can also be used to compute histograms.

Contents

1	Introduction: Block Segmentation	4
2	The Model: Piecewise Constant	5
3	Optimum Partition of an Interval	7
3.1	Data Cells	7
3.2	Blocks of Cells	9
3.3	Partitions	10
3.4	Fitness of a Partition	10
3.5	Changepoints	11
3.6	A Lemma on Subpartitions	12
3.7	The Algorithm	13
4	Block Fitness Functions for Sequential Data	15
4.1	Event Data	17
4.1.1	Poisson Distributed Event Data	19
4.1.2	0-1 Event Data: Duplicate Time Tags Forbidden	23
4.1.3	Time-to-Spill Data	25
4.2	Binned Data	26
4.3	Measurements with Normal Errors	27
4.4	Distributed Measurements	31
4.5	Gaps and Mixed Data Modes	34
4.6	Prior for Number of Blocks	35
5	Examples	38
5.1	Determination of the Parameter γ	38
5.2	Dynamic Range	39
5.3	Point Data Time Series	40

5.4	Binned Data	42
5.5	Maximum Likelihood Histograms	46
5.6	Measurements with Normal Errors	49
5.7	Real Time Analysis: Triggers	50
6	Appendix A: MatLab Code	53
6.1	Main Program	53
6.2	Construct Data Cells	56
6.3	Global Optimum	59
6.4	Load TTE Data	62
6.5	Logarithm of the Cost Function	63
6.6	Plot partitions	66
6.7	Plot TTE partitions	66
6.8	Reverse (from <i>WaveLab</i>)	66
7	Bibliography	68

1 Introduction: Block Segmentation

The goal is to detect local signals in noisy time series data. The term *local* excludes global structures, such as periodic signals that extend over all or a large part of the observation interval. Instead we target signal features that are confined to a limited interval of time, and either do not repeat or repeat at random times.

A key goal is to impose as few prior conditions on the signal as possible. In particular, we wish to avoid smoothness or shape assumptions that place *a priori* limitations on scales and resolution. The algorithm should handle arbitrary sampling (*i.e.*, not be limited to gapless, evenly spaced data) and large dynamic ranges in amplitude and scale. For scientific data mining applications and for objectivity, the method should be automatic. It should eliminate noise as much as possible, while conserving most of the valid information in the data. It should be applicable to multivariate problems. Incorporation of auxiliary, extrinsic data, such as spectral or color information, and variable exposure, should be possible. It should be able to operate both retrospectively (optimal model of all the data after it is collected) and in a real-time fashion that triggers on the first significant variation of the signal from its initial value.

Our algorithm achieves these desiderata in a simple computational framework that is easy to use and represents the structure in the signal in a form handy for further analysis and the estimation of physically meaningful quantities. It includes an automatic penalty for model complexity, thus solving the vexing problem often called *determining the order of the model*. It is exact, not a *greedy approximation*¹ as in [Scargle 1998].

¹An iteration making an optimal improvement at each step, but not guaranteed of an optimal overall solution.

Its limitations include that it detects local, rather than global, structure, and while much faster than an explicit search of the exponentially large parameter space, the runtime of the simple algorithm is $O(N^2)$. This computational complexity is considered prohibitive in some large problems, but an effective way to reduce the time to $\sim N \log N$ is in development.

These desiderata suggest the use of the most generic possible nonparametric data model, and have motivated our development of data segmentation and Bayesian changepoint methods [Ó Ruanaidh and Fitzgerald 1996]. It is remarkable that a very simple idea – *fitting of piecewise constant models to the data* – achieves essentially all of the above desiderata. This approach yields a step-function, or segmented, representation of the signal in which the range of the independent variable (*e.g.* time) is automatically divided into unequal subintervals, in each of which the dependent variable (*e.g.* intensity) is modeled as constant.

2 The Model: Piecewise Constant

As just indicated we are led to employ a very simple model, in which the data interval is partitioned into segments (here called *blocks*) and the signal is taken to be constant within each segment. The model has three parameters for each block: the start time, the duration, and the signal amplitude. In the model of the full data interval the first two are not independent, since one block begins where another leaves off. Specifically, we represent these parameters in terms of a finite set of changepoints, essentially one per block. For event data the signal amplitude is more specifically the *Poisson rate parameter*.

This representation is in the spirit of a nonparametric approxima-

tion, and not meant to imply that we believe the signal is actually discontinuous. The crude, blocky appearance of our discontinuous model may be a liability in the context of visualization, but for our interests in deriving physically meaningful quantities we have not found it so. Blocky models are useful in broad signal processing contexts [Donoho 1994], and have several motivations. Their simplicity allows exact treatment of the likelihood. We can optimize or marginalize the rate parameters exactly, giving simple formulas for the fitness function. And we regard the estimated model itself as less important than quantities derived from it. For example, while smoothed plots of pulses within gamma-ray bursts make pretty pictures, one is really interested in pulse locations, lags, amplitudes, widths, rise and decay times, *etc.* These quantities can be accurately determined directly from the locations, heights and widths of the blocks.

Especially for applications in measuring similarity among time series and pattern matching, piecewise linear models are often used (*cf* the work of Heikki Mannila and Eamonn Keogh). Such models may have a better visual appearance, but in our experience the improved flexibility is largely offset by added complexity of the model and its interpretation. Note further that if continuity is imposed at the changepoints, a piecewise linear model has essentially the same number of parameters, or degrees of freedom, as does the simpler piecewise constant model.

Below §3 discusses partitions of the data interval, a convenient data representation scheme, and the new algorithm for computing optimal partitions. Then §4 exhibits the computation of cost functions for a variety of data modes, followed by numerical simulations and other examples in §5.

3 Optimum Partition of an Interval

Our algorithm works on any sequential data. We introduce it in a somewhat abstract setting because it can be used for other partitioning problems beyond time series analysis. In a special case it implements Bayesian blocks or other 1D segmentation ideas for any model fitness function that satisfies a simple additivity condition. It improves on our previous approximate segmentation algorithms by achieving a rigorous solution of the multiple changepoint problem, and is guaranteed to find the global maximum, not just a local one. This is made possible by reducing the infinite optimization search space to a finite set of partitions consisting of blocks containing discrete data cells, as we now demonstrate.

3.1 Data Cells

The set of possible values of the independent variable is called the *data space*. For the one dimensional case treated here, the data space is usually an interval, such as the time over which observations have been made. The measured quantity can be almost anything. Most commonly it is either a physical variable or the density of discrete events.

Consider observational data comprising N sequential elements

$$\mathbf{x}_n, \quad n = 1, 2, \dots, N. \quad (1)$$

The specific meaning of the quantities \mathbf{x}_n is left vague because almost any of a wide variety of data types can be treated within this formalism. Simple examples are: points, counts of points in bins, and measurements – correspondingly, the array \mathbf{x} would contain point coordinates; counts, bin sizes and locations; and measured values and

their uncertainties, respectively. The only requirement is that the data be ordered (*i.e.*, *sequential*), meaning that each \mathbf{x}_n is associated with a time t_n , such that the latter are ordered and contained in some time interval I :

$$\min(I) \leq t_1 < t_2 < \dots < t_N \leq \max(I) \quad . \quad (2)$$

In general t_n specifies the time of measurement, be it a point or an interval. For event data (also called point data), t_n is just the time of event n . Although times are often represented as real numbers, the finite accuracy of measurement means that one is really specifying an integer multiple of some small unit of time (typically on the order of milliseconds to microseconds in high energy astrophysics). For cases such as binned counts or measurements averaged over finite time intervals, the time interval must be specified, either explicitly (as in an array giving the lengths of a series of unequal time bins) or implicitly (*e.g.* through specification of bin size and time of the first bin).

It is convenient to represent sequential data with a data structure consisting of a set of N *data cells*

$$C_n \equiv \{\mathbf{x}_n, t_n\} \quad , \quad (3)$$

derived from the raw data. They form an ordered sequence with respect to the independent variable t , can be grouped into blocks (§3.2) forming partitions of I (§3.3), and contain whatever data quantities are necessary to evaluate the fitness (§3.4) of an arbitrary partition. In some cases two or more data elements are combined into a single cell (see *e.g.* the discussion of duplicate time tags in §4.1), but for the most part data cells correspond one-to-one with data elements. In some cases (*e.g.* time-tagged event data) t_n is contained in \mathbf{x}_n and need not be separately specified. Figure 1 is a cartoon of typical data

cells.

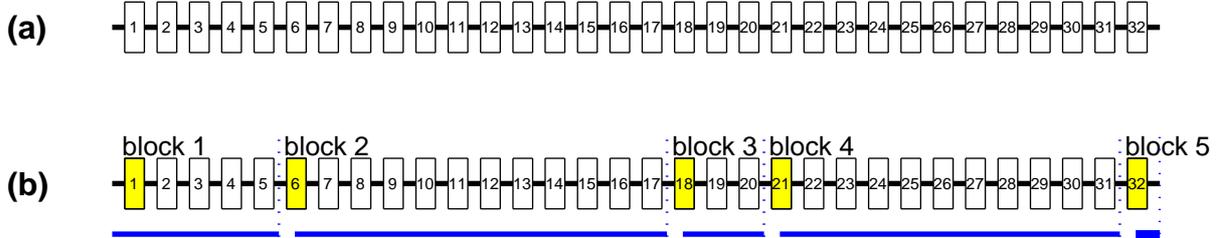


Figure 1: Pictorial representation of data cells and the blocks made from them. The horizontal axis represents the independent variable (often, but not necessarily time), with respect to which the data are ordered. The sequential order depicted in Panel (a) is the only essential requirement for data to be analyzable with our block algorithm. Panel (b) exemplifies the partition of the set of data cells into blocks. The shaded cells are changepoints marking the beginnings of the blocks.

3.2 Blocks of Cells

A block is a set of adjacent cells. Panel (b) of Figure 1 shows a sequence of 32 data cells divided into five blocks. The following notation for blocks is useful:

$$\mathbf{B}(n, m) \equiv \{C_n, C_{n+1}, \dots, C_m\}, \quad (4)$$

that is $m - n + 1$ cells in sequence. The case $m = n$ represents a block consisting of just one cell, as in the last block of the partition in Figure 1(b). The model of the time series data is segmented into blocks, meaning that any model parameters are constant within each block but undergo discrete jumps at the changepoints (§3.5) marking the edges of the blocks. The fitness of a block is of elementary importance, because

the fitness of a partition (§3.4) is the sum of the fitness of the blocks comprising it.

3.3 Partitions

A *partition* of the interval I is simply a set of non-overlapping blocks that together add up to the whole interval.² A partition can be defined by specifying the number of blocks (the elements of the partition) N_{blocks} , and the block edges n_k :

$$\mathcal{P}(I) \equiv \{N_{blocks}, n_k, k = 1, 2, 3, \dots, N_{blocks}\} . \quad (5)$$

There are one fewer changepoints than blocks, since by convention the first block begins at the first data cell – $n_1 \equiv 1$ is implicit – and the last block terminates with the last data cell. As described in §3.4 we will seek the partition that maximizes a given function over all possible partitions. How big is this search space if there are N cells? Establish a 1-1 mapping between partitions and binary numbers of length N , by setting the k -th digit to 1 if cell k is a changepoint, 0 otherwise. Remembering that the first cell is always a changepoint, the number of partitions is then

$$N_{partitions} = 2^{N-1} \quad (6)$$

Except for short time series this number is too large for an exhaustive search, but our algorithm nevertheless finds the optimum over this space in a time that scales as only N^2 .

3.4 Fitness of a Partition

Since our goal is to model data, we maximize³ a quantity measuring the fitness of models in a specified class. We take as this model class

²Formally a partition of I is a set of blocks satisfying $I = \bigcup_k B_k$ and $B_j \cap B_k = \emptyset$ (the null set), for $j \neq k$.

³Alternatively, one can minimize an error measure. Both are called *optimization*.

all partitions of the interval, with a given statistical model for each block of the partition. If the observational errors at different times are independent, as is often the case, fitness is additive over blocks:

$$F[\mathcal{P}(I)] = \sum_{k=1}^{N_{blocks}} f(B_k) \quad , \quad (7)$$

where $F[\mathcal{P}(I)]$ is the total fitness and $f(B_k)$ is the fitness of block k . Our algorithm depends explicitly on this additivity.

Specific examples and details of fitness functions are given below in §4. What is important here is that we marginalize, or otherwise eliminate, all parameters of the block models except the times defining the beginning and end of the block (Paper V). Then the total fitness depends on only $\mathcal{P}(I)$. The best model is found by maximizing F over all partitions. As an example, the fitness function we adopt for count data does not depend on the Poisson rate parameters – they can be computed in an almost trivial way, once the changepoints of the optimum partition are determined.

3.5 Changepoints

We call the time separating two blocks a *changepoint*⁴. In principle a changepoint could be anywhere in the interval, but we restrict them to occur at the times corresponding to the data cells. The reasoning is that moving a changepoint lying between two data cells to a new location between the same cells does not sensibly change the model's representation of the data. This simplification reduces the search over an infinite space to a finite optimization problem.

In some applications it might be useful to assign a data cell that is

⁴In statistics, a changepoint in a time series is a point at which the statistical model undergoes an abrupt transition, usually by one or more of its parameters jumping to a new value

a changepoint to be in *both* the subsequent and previous blocks, but here we assign it to only one – with the convention that a changepoint is the first cell in the subsequent block (rather than the last cell of the previous block). Correspondingly, since the smallest partition consists of a single block containing all data cells, the first data cell is always a changepoint. If the last cell is a changepoint, it demarcates a block consisting of that one cell, as in panel (b) of Figure 1, where the five changepoints dividing the data cells into five blocks are shaded.

3.6 A Lemma on Subpartitions

We define a *subpartition* of a given partition $\mathcal{P}(I)$ to be a partition (of a subset of I) consisting of a subset of the blocks of $\mathcal{P}(I)$. Although not a necessary condition for the lemma to be true, in all cases of interest here the blocks in the subpartition are contiguous, and thus form a partition of a subinterval of I . Below we will make use of this simple result on subpartitions of optimal partitions:

Lemma: A subpartition of an optimal partition is an optimal partition of the subset it covers.

Let \mathcal{P}' be the subpartition and I' the subset of I that it covers. If there were a partition of I' , different from and fitter than \mathcal{P}' , then combining it with the blocks of \mathcal{P} not in \mathcal{P}' would, by the block additivity condition, yield a partition of I fitter than \mathcal{P} , contrary to the optimality of \mathcal{P} . ■

Corollary: removing the last block of an optimal partition leaves an optimal partition.

3.7 The Algorithm

We have assembled the definitions and results needed to state our procedure and prove that it finds a global optimum partition. This algorithm is in the spirit of dynamic programming [Hubert, Arabie, and Meulman 2001]. It begins with the first data cell, adding one more at each step until the whole interval has been treated. This feature makes the algorithm suitable for real-time applications (see §5.7).

The proof is by mathematical induction: if a theorem is true for $R = 1$, and one can show that, if it is true for R then it is true for $R+1$, then the theorem holds for all R . At step R the algorithm finds the optimum partition of the interval comprised of data cells $I_R \equiv \{C_1, C_2, \dots, C_R\}$. To analyze all the data, take $R = 1, 2, \dots, N$. The case $R = 1$ is trivial: there is only one cell, and the only partition possible is the optimum one.

Now suppose we have completed step R , having obtained the optimal partition $\mathcal{P}^{opt}[I_R]$, hereafter abbreviated $\mathcal{P}^{opt}(R)$, and are now at step $R + 1$ and wish to find the optimal partition $\mathcal{P}^{opt}(R+1)$. Assume further that we have kept a running record of the fitness of the optimum partition obtained at each previous step (call this array **best**) and the location of the last changepoint in that partition (call this array **last**). It is straightforward to compute

$$M(r) \equiv f[\mathbf{B}(r, R + 1)] \quad (r = 1, 2, \dots, R + 1) \quad (8)$$

that is, the fitness of a putative last block starting at r and extending to the end of the current interval. For example $M(1)$ is the fitness of the whole interval currently in play, namely the cells from 1 through $R + 1$.

Using the block additivity of fitness, Eq. (7), the fitness of the partition of I_{R+1} consisting of the optimum partition $\mathcal{P}^{opt}[I_{r-1}]$ followed by a single block $\mathbf{B}(r, R+1)$ is:

$$A(r) = M(r) + \begin{cases} 0 & r = 1 \\ \mathbf{best}(r-1), & r = 2, 3, \dots, R+1 \end{cases}, \quad (9)$$

Now comes the key reasoning step. While we don't yet know what it is, the new optimum partition $\mathcal{P}^{opt}(R+1)$ must exist and must have a *last changepoint*, say r^* .⁵ From its definition $A(r^*)$ is the fitness of $\mathcal{P}^{opt}(R+1)$. In particular, $\mathbf{best}(r^*-1)$ is the fitness of the optimal subpartition consisting of all but the last block of $\mathcal{P}^{opt}(R+1)$, and $M(r^*)$ is the fitness of said last block. Further, any partition with its last changepoint at some other $r \neq r^*$ must have fitness not greater than that of $\mathcal{P}^{opt}(R+1)$, so we have

$$A(r) \leq A(r^*) \quad \text{for } r \neq r^*. \quad (10)$$

In other words, the maximum of $A(r)$ occurs at r^* :

$$r^* = \operatorname{argmax}[A(r)], \quad (11)$$

so finding the fitness and last changepoint of $\mathcal{P}^{opt}(R+1)$ is just a matter of finding the maximum of the array A and the index r at which this maximum occurs.

At the end of the computation, it only remains to find the locations of the optimal changepoints. The needed information is contained in the array $\mathbf{last}(r)$ in which we have stored the index r^* at each step. Using the corollary of the subpartition lemma, it is a simple matter to use the last value in this array to determine the last changepoint in $\mathcal{P}^{opt}(N)$, peel off the end section of \mathbf{last} corresponding to this last block, and repeat. That is to say, the values

⁵Any finite combinatorial optimization problem has at least one solution. Also, all partitions have at least one changepoint.

- (1) $cp_1 = \text{last}(N)$
- (2) $cp_2 = \text{last}(cp_1 - 1)$
- (3) $cp_3 = \text{last}(cp_2 - 1)$
- ...

are the index values giving the locations of the changepoints, in reverse order. The positions of the changepoints are not necessarily fixed until the very last iteration, although in practice it turns out that they become more or less “frozen” once a few succeeding changepoints have been detected.

The MatLab code for the algorithm in Appendix XX indicates how all of these computations are implemented.

4 Block Fitness Functions for Sequential Data

Here we outline the computation of *model fitness*. The *fitness function* for a fixed block of data numerically evaluates how well a constant signal strength represents whatever data lie in that block. The resulting quantities for all blocks in the observation interval are combined to form a fitness measure for the complete (piecewise constant) model.

For our algorithm to work, in addition to being block-additive [§3.4, Eq. (7)], the total model fitness must depend on only parameters which specify the locations of the block edges, *i.e.* the changepoints (§3). We must account for and eliminate all other parameters. The only ones in our model are the block signal strengths, which can be eliminated by taking block fitness to be the likelihood either maximized or marginalized with respect to signal strength. In both cases the result is a quantity assessing alternative models for the data, not an absolute goodness-of-fit.

Computation of fitness functions varies with the data mode, but the following features are common to all cases considered in this paper. The fitness function always depends on only the parameters describing the error distribution of whatever measurements lie in the block. For event data governed by the Poisson distribution (§§4.1, 4.2), there are exactly two such *sufficient statistics*: N , the number of events in the block, and M , the length of the block. In other cases (*e.g.* §4.3) the number of parameters depends on the form of the distribution. If the sufficient statistics for a block are the sums of those for its cells (as in all cases treated here), the computations are simplified; however this condition is not essential.

It is interesting to note some things that do not matter, because they do not change the sufficient statistics. For example, the actual locations of the data cells within their assigned blocks do not matter. The cells in a block need not even be contiguous. This allows a very simple treatment of data gaps. Or the cells near the beginning and the end of the interval might be assigned to the same block. For example, the pre-burst and post-burst data from a gamma ray burst could combine into a single block representing a constant background. An algorithm explicitly allowing wraparound would be a natural way to deal with this case. These extensions are of most interest for higher dimensional data, and will be further discussed in a future paper.

Finally, there are two types of factors in a fitness function that can be ignored, for different reasons. First, a factor in the likelihood for each data cell that does not depend on the rate parameter yields a simple constant factor for the whole time series (namely the product of the factor over all the data cells), independent of both the rate parameter and where the changepoints lie. Such a factor cancels out in

any explicit model comparison, and is irrelevant as well for the implicit model comparison that takes place in our optimization algorithm. Second, there are factors in the fitness function for each block that are independent of the rate parameter. These factors do matter, but they contribute to the log of the fitness function a term proportional to the number of blocks, and as such can be absorbed into the parameter derived from the prior on the number of blocks (*cf.* §4.6).

Many of the data modes discussed in the following subsections are part of the Burst and Transient Source Experiment (BATSE) experiment on board the now-defunct NASA Compton Gamma Ray Observatory (GRO). However, they are relevant to a wide range of astronomical data acquisition systems, especially in high energy astrophysics.

4.1 Event Data

Sometimes the physical process, or perhaps the way it is recorded, takes the form a sequence of discrete events, each yielding a point in the data space. (In practice, the coordinates of the points are integer multiples of some small but finite unit—and are thus discrete, not continuous. This fact is important for the computations below.) The quantity of ultimate interest is the *distribution function* of the points, interpretable as the intensity or probability density of some physical variable. Accordingly the terms *density estimation* and *rate estimation* are sometimes used. A key example is the case where the events are the detection of individual photons, the corresponding points are the measured detection times, and the quantity of interest is the radiation intensity as a function of time.

For point data, it is natural to associate one cell with each event. However, if the detector can detect two (or more) events that are si-

multaneous to within its timing accuracy, such pairs would be assigned to the same cell. Since data cells must contain whatever information is necessary to compute the fitness function of a block containing the cells (§3.1), the data structure representing the cells must contain the number of events assigned to the cell (most often 1) and the length of the interval associated with the event.

There is more than one way to make such an association between sequential events and intervals. Perhaps most natural is to assign to a point *all times closer to it than to any other data point*. This resulting intervals join the midpoints between successive events. This concept generalizes to data spaces of any dimensions (where it is called the *Voronoi tessellation* of the data points, [Okabe, Boots, Sugihara and Chiu 2000, Scargle 2001a, Scargle 2001c]), allowing finite partitions which adequately approximate the infinite set of arbitrary partitions.

Alternatively, one can use the intervals between successive data points—assigning half of an event to the interval immediately to its left and half to the one immediately to its right. This choice may handle the onset of a steep gradient in the underlying density slightly better, and is also easily generalized to higher dimension where it is known as the *Delaunay triangulation* [Okabe, Boots, Sugihara and Chiu 2000]. The algorithm described below allows use of either of these interval schemes.

The analysis in §2.2.1 of [Scargle 1998] can be carried over largely unchanged to the cell-based approach described here. But we offer several extensions of that work. First, we develop a new class of fitness functions based on maximizing the likelihood with respect to the rate parameter, in contrast to marginalizing it as in computing the Bayesian

posterior. In addition, for the case where we compute the posterior we consider a prior with a finite range, as opposed to the flat prior over an infinite range. And we include variable bin size and exposure factors.

As mentioned above, and detailed in §2.2.1 of [Scargle 1998], assume that there is an elementary quantum of time—a *tick*—set by the measurement system. This is the finest time resolution the measurement apparatus is capable of recording. Let n_m be the number of events (*e.g.* photons) detected in tick m . We consider two data modes. In mode 1 the number of events in a given tick is presumed to follow a Poisson distribution. Mode 2 corresponds to situations where detection of more than one event at a given time is not possible, typically due to the *deadtime* of the detector, so that the number of events in a tick can be only 0 or 1. An example is time-tagged event (TTE) data in which duplicate time tags are not allowed. The fitness functions for the two modes, while similar, are different enough that the appropriate one should be used in practice.

4.1.1 Poisson Distributed Event Data

For mode 1, the likelihood for tick m is, from the Poisson distribution

$$L_m = \frac{\lambda^{n_m} e^{-\lambda}}{n_m!} . \quad (12)$$

The block likelihood is the usual product

$$L^{(k)} = \prod_{m=1}^{M^k} \frac{\lambda^{n_m} e^{-\lambda}}{n_m!} . \quad (13)$$

where M^k is the number of ticks in block k . Simplifying and collecting the factors for ticks with the same number of events, we have

$$L^{(k)} = e^{-\lambda M^k} \prod_{n=0}^{\infty} \left(\frac{\lambda^n}{n!} \right)^{H(n)} , \quad (14)$$

where $H(n)$ is the number of ticks in the block with n events. The factor resulting from the factorial in the denominator is a constant, independent of the model, and therefore irrelevant for model comparison. Dropping this factor, and noting that $\sum_{n=0}^{\infty} nH(n) = N^k$, we have

$$L^{(k)} = \lambda^{N^k} e^{-\lambda M^k} \quad (15)$$

In this context it is often suggested that one should employ the intervals between successive events, since they in some sense carry the rate information more directly than do the actual times. We will now show that the likelihood based on intervals is essentially equivalent to the one above. It is a classic result [Papoulis 1965] that intervals between independent events distributed uniformly in time with a constant rate λ is exponential:

$$P(dt) = \lambda e^{-\lambda dt} U(dt), \quad (16)$$

where $U(x)$ is the unit step function:

$$\begin{aligned} U(x) &= 1 & x \geq 0 \\ &= 0 & x < 0 \end{aligned}$$

Pretend that the data consists of the inter-event intervals, and we do not even know the absolute times. The likelihood of our constant-rate Poisson model for interval $dt_n \geq 0$ is

$$L_n = \lambda e^{-\lambda dt_n}, \quad (17)$$

so the block likelihood is

$$L^{(k)} = \prod_{n=1}^{N^k} \lambda e^{-\lambda dt_n} = \lambda^{N^k} e^{-\lambda M^k}, \quad (18)$$

This likelihood is the same as that in Eq. (15).

There are two ways to proceed. The first is to find the maximum of this likelihood as a function of λ , which is at $\lambda = \frac{N^k}{M^k}$, so we have

$$L_{max} = \left(\frac{N^k}{M^k}\right)^{N^k} e^{-N^k} \quad (19)$$

The log of this expression,

$$\boxed{\log L_{max} = N^k \left(\log \frac{N^k}{M^k} - 1\right)}, \quad (20)$$

is the maximum likelihood fitness function for event data following a Poisson distribution.

In the other approach, we marginalize the likelihood in Eq. (15) with the finite-range constant prior, giving

$$P = \frac{1}{\lambda_2 - \lambda_1} \int_{\lambda_1}^{\lambda_2} \lambda^{N^k} e^{-\lambda M^k} d\lambda \quad (21)$$

yielding

$$P = \frac{1}{\lambda_2 - \lambda_1} \frac{1}{(M^k)^{N^k+1}} \int_{z_1}^{z_2} z^{N^k} e^{-z} dz \quad (22)$$

where $z_{1,2} = M^k \lambda_{1,2}$. In terms of the incomplete gamma function

$$\gamma(a, x) \equiv \int_0^x z^{a-1} e^{-z} dz \quad (23)$$

this is

$$\boxed{P = \frac{1}{\lambda_2 - \lambda_1} \frac{1}{(M^k)^{N^k+1}} \left[\gamma(N^k + 1, z_2) - \gamma(N^k + 1, z_1) \right]}. \quad (24)$$

The unnormalizable flat prior that extends to infinity gives

$$\boxed{P = \frac{1}{(M^k)^{N^k+1}} \Gamma(N^k + 1)}, \quad (25)$$

differing slightly from Eq. (29) of [Scargle 1998] only because of different priors for λ .

Another commonly used prior is the so-called conjugate Poisson distribution

$$P(\lambda) = C \lambda^{\alpha-1} e^{-\beta\lambda} . \quad (26)$$

As noted by [Gelman] this “prior density is, in some sense, equivalent to a total count of $\alpha-1$ in β prior observations” a relation that might be useful in some circumstances. The normalization constant $C = \frac{\beta^\alpha}{\Gamma(\alpha)}$ will be ignored. With this prior the marginalized posterior probability is

$$P = \int_0^\infty \lambda^{N^{(k)}+\alpha-1} e^{-\lambda(M^{(k)}+\beta)} d\lambda , \quad (27)$$

or

$$P = \frac{\Gamma(N^{(k)}+\alpha)}{(M^{(k)}+\beta)^{N^{(k)}+\alpha}} \quad (28)$$

Note that this prior and posterior reduce to those in Eqs. (28) and (29) of [Scargle 1998] for $\alpha = 1, \beta = 1$.

Equations (20), (24), (25) and (28) are the forms to be used whenever the counts in each tick follow the Poisson distribution. This includes both time-tagged data where duplicate tags are permitted and, as we will see below in §4.2, binned data.

Recently [Prahl 1996] has derived a statistic for event clustering in Poisson process data that tests departures from the known interval distribution (see the discussion above), by evaluating the likelihood over a restricted interval range. Prahl’s statistic is

$$M_N = \frac{1}{N} \sum_{\Delta T_i < C^*} \left(1 - \frac{\Delta T_i}{C^*}\right) , \quad (29)$$

where ΔT_i is the interval between events i and $i + 1$, and

$$C^* \equiv \frac{1}{N} \sum \Delta T_i \quad (30)$$

is the empirical mean interval. In other settings, the fact that this statistic is a global measure of departure of the distribution (used here only locally, over one block) may be useful in the detection of periodic, and other global, signals in event data. Results using the Prahla statistic are given below in §5.

4.1.2 0-1 Event Data: Duplicate Time Tags Forbidden

In this mode duplicate time tags are not allowed, the number of events detected at a given tick is 0 or 1, and the corresponding tick likelihood is:

$$L_m(\lambda) = e^{-\lambda} = 1 - p \quad n_m = 0 \quad (31)$$

$$= 1 - e^{-\lambda} = p \quad n_m = 1 \quad (32)$$

where λ is the model event rate. From the Poisson distribution $p = 1 - e^{-\lambda}$ is the probability of an event, $1 - p = e^{-\lambda}$ that of no event. We can therefore use p or λ interchangeably to specify the event rate. Since independent probabilities multiply, the block likelihood is the product of the tick likelihoods:

$$L^{(k)} = \prod_{m=1}^{M^k} L_m = p^{N^k} (1 - p)^{M^k - N^k} \quad (33)$$

where M^k is the number of ticks in block k and N^k is the number of events in the block.

There are again two ways to proceed. The maximum of this likelihood occurs at $p = \frac{N^k}{M^k}$ and is

$$L_{max} = \left(\frac{N^k}{M^k}\right)^{N^k} \left(1 - \frac{N^k}{M^k}\right)^{M^k - N^k} \quad (34)$$

Using the logarithm of the maximum likelihood,

$$\boxed{\log(L_{max}) = N^k \log\left(\frac{N^k}{M^k}\right) + (M^k - N^k) \log\left(1 - \frac{N^k}{M^k}\right)} \quad (35)$$

yields the additivity needed for our cost function.

An alternative way to quantify the fitness of the class of constant models to marginalize the rate parameter. That is to say, we remove this parameter by integrating it out:

$$P(B^k) = \int L^{(k)} P(\lambda) d\lambda, \quad (36)$$

where $P(\lambda)$ is the prior probability distribution for the rate parameter. Here we adopt a generic prior that is consistent with not having any particular prior information about the event rate, except that it must be positive. In [Scargle 1998] we used p as the independent variable, and chose a prior flat (constant) as a function of p . Here, we use a prior flat as a function of the rate parameter:

$$P(\lambda) = \frac{1}{\lambda_2 - \lambda_1} \quad \lambda_1 \leq \lambda \leq \lambda_2 \quad (37)$$

$$= 0 \quad \text{otherwise} \quad (38)$$

The posterior, marginalized over λ is then:

$$P = \frac{1}{\lambda_2 - \lambda_1} \int_{\lambda_1}^{\lambda_2} (1 - e^{-\lambda})^{N^k} (e^{-\lambda})^{M^k - N^k} d\lambda. \quad (39)$$

Changing variables to $p = 1 - e^{-\lambda}$, with $dp = e^{-\lambda} d\lambda$, this integral becomes

$$P = \frac{1}{\lambda_2 - \lambda_1} \int_{p_1}^{p_2} p^{N^k} (1 - p)^{M^k - N^k - 1} dp, \quad (40)$$

with $p_1 = 1 - e^{-\lambda_1}$ and $p_2 = 1 - e^{-\lambda_2}$, expressible in terms of the *incomplete beta function*

$$B(z; a, b) = \int_0^z u^{a-1} (1 - u)^{b-1} du \quad (41)$$

as follows:

$$\boxed{P = \frac{1}{\lambda_2 - \lambda_1} [B(p_2; N^k + 1, M^k - N^k) - B(p_1; N^k + 1, M^k - N^k)]}. \quad (42)$$

The incomplete beta function reduces to the ordinary *beta function* for $z = 1$, so for the infinite range case $\lambda_1 = 0, \lambda_2 = \infty; p_1 = 0, p_2 = 1$ we have

$$\boxed{P_\infty = B(N^k + 1, M^k - N^k)}, \quad (43)$$

differing from Eq. (21) of [Scargle 1998] by one in the second argument, due to the difference between a prior flat in p and one flat in λ . All of the equations (35), (42), and (43), in their logarithmic form, can be used as fitness functions in the global optimization algorithm, and will be demonstrated below.

4.1.3 Time-to-Spill Data

As discussed in §2.2.3 of [Scargle 1998], reduction of the necessary telemetry rate is sometimes accomplished by recording only the time of detection of every S th photon, e.g. with $S=64$ for the BATSE time-to-spill mode. This data mode has the attractive feature that its time resolution is greater when the source is brighter (and possibly more active, so that more time resolution is useful). The likelihood in Eq. (32) of [Scargle 1998] simplifies, with slightly revised notation and using the fourth comment at the beginning of this section, to

$$L_{TTS}^{(k)} = \lambda^{SN_{spills}} e^{-\lambda M} \quad (44)$$

where N_{spills} is the number of spill events in the block, and M is as usual the length of the block. With $N = N_{spills}S$ this is identical to the Poisson likelihood in Eq.(15), and in particular the maximum likelihood is at $\lambda = \frac{N_{spills}S}{M}$ and the corresponding cost function is

$$\log L_{max,TTS}^{(k)} = SN_{spills} \left(\log \frac{N_{spills}S}{M} - 1 \right) \quad (45)$$

just as in Eq. (20) with $N = SN_{spills}$, and with the same property that the unit in which block lengths are expressed is irrelevant.

4.2 Binned Data

One of the most common data modes consists of counts in bins. The bins are typically predefined intervals in the measured variable. The count N_n in bin n is simply the number of values in it. The data cells are simply the bins and their associated counts:

$$\mathbf{Cell\ n} \equiv \{\text{bin } n, N_n\}, n = 1, 2, \dots, N_{total}. \quad (46)$$

Absent correlation effects, such as dead time, the probability distribution for the number of events of a bin is Poisson, and this data mode is equivalent to that discussed above in §4.1.1, with the bins taking the role of the ticks of that section. Here we generalize these results (and those in [Scargle 1998]) in two ways, allowing unequal bin sizes and a variable efficiency factor. The latter, sometimes called *exposure*, refers to anything that affects the count (*e.g.* instrumental sensitivity, dwell time, or uncorrected atmospheric effects). Assume that, whatever the nature of the effect, it can be represented by an efficiency factor between 0 and 1, such that the effective Poisson event rate is E times the actual (observed or modeled) event rate. Because of the nature of our piece-wise constant Poisson model, these two effects—bin size and bin efficiency—are equivalent in simply altering the local event rate, and can be represented with a single parameter equal to the product of the bin’s width and efficiency.

The likelihood for bin n is found from the Poisson distribution:

$$L_n = \frac{(\lambda E_n W_n)^{N_n} e^{-\lambda E_n W_n}}{N_n!} \quad (47)$$

where λ is the actual event rate, in counts per unit time, and N_n is the number of events in the bin. The bin width W_n is expressed in the same units as λ^{-1} . The efficiency factor E_n is averaged over the bin. The

product $W_n E_n$ can be replaced with a single quantity, $w_n \equiv W_n E_n$, expressing relative bin efficiencies.

The likelihood for block k is the product of the likelihoods of all its bins:

$$L^{(k)} = \prod_{n=1}^{M^{(k)}} L_n = \lambda^{N^{(k)}} e^{-\lambda w^{(k)}}. \quad (48)$$

Here $M^{(k)}$ is the number of bins in block k ,

$$w^{(k)} = \sum_{n=1}^{M^{(k)}} w_n \quad (49)$$

is the sum of the bin efficiencies in the block, and

$$N^{(k)} = \sum_{n=1}^{M^{(k)}} N_n \quad (50)$$

is the total event count in the block. We have discarded the factor $(E_n W_n)^{N_n} / N_n!$ in Eq. (47) because, when multiplied out over all blocks in any model it produces a model-independent factor—its product over all bins. Any such common factor is irrelevant for model comparison.

Note that the block likelihood is essentially the same as that of Eq. (15). The only difference is that what we called a tick is now called a bin, and we have allowed for a bin efficiency factor (which in principle could be applied to ticks). Hence the maximum likelihood and marginal posterior cost functions to be used here are the same as those of Equations (20), (24), (25) and (28), with $M^{(k)}$ interpreted as the block-sum of the w_n instead of just the number of ticks in the block.

4.3 Measurements with Normal Errors

Here is a very common signal processing scenario: in order to estimate a signal embedded in noise, one makes measurements at a sequence

of times. For example, if the noise is additive one has this nearly ubiquitous model for the time series observations:

$$x_n \equiv x(t_n) = f(t_n) + z_n \quad n = 1, 2, \dots, N, \quad (51)$$

where f is the unknown signal, z is the noise, and the observation times t_n may be evenly spaced or otherwise. We here consider the case where the noise is assumed to be normally distributed and with a known variance:

$$P(z_n | \sigma_n) = \frac{1}{\sigma_n \sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{z_n}{\sigma_n}\right)^2} \quad (52)$$

The data cell then is denoted

$$\mathbf{X}_n = \left\{ x_n, t_n, \sigma_n \right\} \quad n = 1, 2, \dots, N, \quad (53)$$

where x_n is the value measured at time t_n , and σ_n is the standard deviation of the noise.

In a block where the true signal is λ , the likelihood of measurement n is then

$$L_n = \frac{1}{\sigma_n \sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{x_n - \lambda}{\sigma_n}\right)^2} \quad (54)$$

and the entire likelihood for block k is

$$L^{(k)} = \prod_n \frac{1}{\sigma_n \sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{x_n - \lambda}{\sigma_n}\right)^2} \quad (55)$$

where the product is over all n such that t_n falls within the block. The exponential is the only factor that matters, since the rest contributes to the total posterior probability the constant factor

$$\frac{(2\pi)^{-\frac{N}{2}}}{\prod_{n=1}^N \sigma_n}, \quad (56)$$

where here the product is over all N data points. Hence the block likelihood can be written

$$L^{(k)} = e^{-\frac{1}{2} \sum_n \left(\frac{x_n - \lambda}{\sigma_n}\right)^2} \quad (57)$$

The maximum of this likelihood is found as follows: Clearly we can just as well minimize the quantity

$$Q(\lambda) = \frac{1}{2} \sum_n \left(\frac{x_n - \lambda}{\sigma_n}\right)^2, \quad (58)$$

which can be done by setting its derivative to zero:

$$\frac{dQ(\lambda)}{d\lambda} = - \sum_n \left(\frac{x_n - \lambda}{\sigma_n^2}\right) \quad (59)$$

so that

$$\lambda_{max} = \frac{\sum_n \left(\frac{x_n}{\sigma_n^2}\right)}{\sum_n \left(\frac{1}{\sigma_n^2}\right)} \quad (60)$$

Letting $\rho_n = \frac{1}{\sigma_n^2}$ be the weight corresponding to the variance of measurement n , and putting the resulting expression

$$\lambda_{max} = \frac{\sum_n \rho_n x_n}{\sum_n \rho_n} \quad (61)$$

into the log of Eq. (57), we have

$$\log P = -\frac{1}{2} \sum_n \rho_n \left(x_n - \frac{\sum_n \rho_n x_n}{\sum_n \rho_n}\right)^2 \quad (62)$$

$$\log P = -\frac{1}{2} \sum_n \rho_n \left[x_n^2 - 2x_n \frac{\sum_n \rho_n x_n}{\sum_n \rho_n} + \left(\frac{\sum_n \rho_n x_n}{\sum_n \rho_n}\right)^2\right] \quad (63)$$

$$\log P = -\frac{1}{2} \left[\sum_n \rho_n x_n^2 - 2 \frac{(\sum_n \rho_n x_n)^2}{\sum_n \rho_n} + \frac{(\sum_n \rho_n x_n)^2}{\sum_n \rho_n}\right] \quad (64)$$

$$\log P = -\frac{1}{2} \left[\sum_n \rho_n x_n^2 - \frac{(\sum_n \rho_n x_n)^2}{\sum_n \rho_n} \right] \quad (65)$$

$$\boxed{\log P = -\frac{1}{2} \left[\bar{x}^2 - \frac{\bar{x}^2}{\sum_n \rho_n} \right]} \quad (66)$$

This expression is related to the *weighted variance*, although there seems to be no universal choice for how to define same. But it makes sense that the block cost function is this variance: the best constant model for the block should have minimum variance.

As in the other cases, we can alternatively marginalize λ , by choosing the flat, unnormalizable prior

$$P(\lambda) = \text{constant} \quad (67)$$

yielding for the marginal posterior

$$P(B_k) = \int_{-\infty}^{\infty} e^{-\frac{1}{2} \sum_n \left(\frac{x_n - \lambda}{\sigma_n} \right)^2} d\lambda \quad (68)$$

Setting

$$a_k = \frac{1}{2} \sum_n \frac{1}{\sigma_n^2} \quad (69)$$

$$b_k = - \sum_n \frac{x_n}{\sigma_n^2} \quad (70)$$

and

$$c_k = \frac{1}{2} \sum_n \frac{x_n^2}{\sigma_n^2} \quad (71)$$

we have

$$P(B_k) = \int_{-\infty}^{\infty} e^{-\frac{1}{2}(a_k \lambda^2 + b_k \lambda + c_k)} d\lambda \quad (72)$$

$$= \sqrt{\frac{\pi}{a_k}} e^{\left(\frac{b_k^2}{4a_k} \right) - c_k} \quad (73)$$

The total posterior is of course

$$P = \prod_k P(B_k) \quad (74)$$

or, in terms of the additive log-posterior, we have

$$\boxed{\log P = \sum_k \log P(B_k) = \sum_k \left[-\frac{1}{2} \log(a_k) + \left(\frac{b_k^2}{4a_k} \right) - c_k \right]} \quad (75)$$

where the sum is over all blocks, k .

As with the other data modes, either of equations (66) or (75) can be used for normally distributed data.

4.4 Distributed Measurements

The data can also consist of measurements of a quantity, averaged over a range of values of t – not at discrete point, as in the previous section. A good example is the spatial power spectra computed from measurements of the cosmic microwave background radiation [refs.], where the different experiments have widely different *window functions* (the term used to describe sensitivity as a function of the independent variable – *i.e.*, spatial harmonic number in the CMB case). In this case the data array could consist of the structure in Equation (53) augmented by the inclusion of a window function, indicating the variation of the instrumental sensitivity:

$$x = \{x_n, t_n, w_n(t - t_n)\} \quad n = 1, 2, \dots, N, \quad (76)$$

where $w_n(t)$ describes, for the value reported as X_n , the relative weights assigned to times near t_n , and all other quantities are as in Eq. (53).

This is a nontrivial complication if the window functions overlap, but can nevertheless be handled with the same technique.

We assume the standard piece-wise constant model of the underlying signal, that is, a set of contiguous blocks:

$$B(x) = \sum_{j=1}^{N_b} B^{(j)}(x) \quad (77)$$

where each block is represented as a *boxcar* function:

$$B^{(k)}(x) = \begin{cases} B_j & \zeta_j \leq x \leq \zeta_{j+1} \\ 0 & \text{otherwise} \end{cases} \quad (78)$$

the ζ_j are the changepoints, satisfying

$$\min(x_n) \leq \zeta_1 \leq \zeta_2 \leq \dots \leq \zeta_j \leq \zeta_{j+1} \leq \dots \leq \zeta_{N_b} \leq \max(x_n) \quad (79)$$

and the B_j are the heights of the blocks.

The value of the observed quantity, y_n , at x_n , under this model is

$$\begin{aligned} \hat{y}_n &= \int w_n(x) B(x) dx \\ &= \int w_n(x) \sum_{j=1}^{N_b} B^{(j)}(x) dx \\ &= \sum_{j=1}^{N_b} \int w_n(x) B^{(j)}(x) dx \\ &= \sum_{j=1}^{N_b} B_j \int_{\zeta_j}^{\zeta_{j+1}} w_n(x) dx \end{aligned} \quad (80)$$

so we can write

$$\hat{y}_n = \sum_{j=1}^{N_b} B_j G_j(n) \quad (81)$$

where

$$G_j(n) \equiv \int_{\zeta_j}^{\zeta_{j+1}} w_n(x) dx \quad (82)$$

is the inner product of the n -th weight function with the support of the j -th block. The analysis in [Bretthorst 1988] shows how to deal with the non-orthogonality that is generally the case here.⁶

⁶If the weighting functions are delta functions, it is easy to see that $G_j(n)$ is non-zero if and only if x_n lies in block j , and since the blocks do not overlap the product $G_j(n)G_k(n)$ is zero for $j \neq k$, yielding orthogonality, $\sum_N G_j(n)G_k(n) = \delta_{j,k}$. And of course there can be some orthogonal blocks, for which there happens to be no “spill over”, but these are exceptions.

[Note: the following repeats some of the above, and therefore needs to be rewritten.]

The averaging process in this data model induces dependence among the blocks. The likelihood, written as a product of likelihoods of the assumed independent data samples, is

$$P(\text{Data}|\text{Model}) = \prod_{n=1}^N P(y_n|\text{Model}) \quad (83)$$

$$= \prod_{n=1}^N \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{1}{2}\left(\frac{y_n - \hat{y}_n}{\sigma_n}\right)^2} \quad (84)$$

$$= \prod_{n=1}^N \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{1}{2}\left(\frac{y_n - \sum_{j=1}^{N_b} B_j G_j(n)}{\sigma_n}\right)^2} \quad (85)$$

$$= Q e^{-\frac{1}{2}\left(\frac{y_n - \sum_{j=1}^{N_b} B_j G_j(n)}{\sigma_n}\right)^2}, \quad (86)$$

where

$$Q \equiv \prod_{n=1}^N \frac{1}{\sqrt{2\pi\sigma_n^2}}. \quad (87)$$

After more algebra and adopting a new notation, symbolized by

$$\frac{y_n}{\sigma_n^2} \rightarrow y_n \quad (88)$$

and

$$\frac{G_k(n)}{\sigma_n^2} \rightarrow G_k(n), \quad (89)$$

we arrive at

$$\log P(\{y_n\}|B) = Q e^{-\frac{H}{2}}, \quad (90)$$

where

$$H \equiv \sum_{n=1}^N y_n^2 - 2 \sum_{j=1}^{N_b} B_j \sum_{n=1}^N y_n G_j(n) + \sum_{j=1}^{N_b} \sum_{k=1}^{N_b} B_j B_k \sum_{n=1}^N G_j(n) G_k(n). \quad (91)$$

The last two equations are equivalent to Eqs. (3.2) and (3.3) of [Bretthorst 1988], so that the orthogonalization of the basis functions and the final expressions follow exactly as in that reference.

4.5 Gaps and Mixed Data Modes

In some cases there are subintervals over which no events are possible (*e.g.* gaps due to failures in the detector system). What matters is the “live time” during the block, and this is simply the sum of the cell lengths. Thus data gaps can be handled by ignoring them! The only subtlety lies in interpreting what the model implies if a block extends across a gap. For each block the procedure yields the optimum rate parameter for whatever data lies in the block, ignoring any gaps. At the end of the procedure, for display purposes the gaps can be restored and plotted, preferably with some indication that rates within gaps are more uncertain.

Only if the fitness function depends on the total length of the block, and not just the live time, do the lengths of the overlap between the block and these gaps need to be included. The only example of this we have encountered results from the adoption of a prior distribution of block width.

Furthermore, one can even mix data modes. *E.g.*, bins of arbitrary sizes can be combined with point data. As with gaps the only burden for doing this is placed on the fitness function, which in this case would have to include a provision for data of mixed modes falling within the block. An example of this would be the analysis of both binned and time-tagged event (TTE) data for gamma-ray bursts observed by BATSE.

Which of the several posteriors above should be used? Should a

new fitness function be constructed, based on ones understanding of the data and potential signals? If the conjugate prior is used, what values of its two parameters should be used? The answers depend on what is known about the data and its errors, as well as what one wants to assume about the signal. To aid in making such choices, §5.4 has relevant examples.

4.6 Prior for Number of Blocks

In earlier work [Scargle 1998] no explicit prior probability distribution was assigned the number of blocks, *i.e.* the parameter N_{blocks} . This omission amounts to using a flat prior, but in many contexts it is unreasonable to assign the same prior probability to all values. In particular, in most settings $N_{blocks} \ll N$ is *a priori* much more likely than $N_{blocks} \approx N$. 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* [Coram 2002]:

$$P(N_{blocks}) = P_0 \gamma^{-N_{blocks}} \quad (92)$$

for $0 \leq N_{blocks} \leq N$, and zero otherwise since N_{blocks} cannot be negative nor larger than the number of data cells. The normalization constant – irrelevant for model comparison – is easily obtained, giving

$$P(N_{blocks}) = \frac{1 - \frac{1}{\gamma}}{1 - (\frac{1}{\gamma})^{N+1}} \gamma^{-N_{blocks}} \quad (93)$$

Through the dependence of this prior on N_{blocks} , the value of γ affects the number of blocks in the optimal representation—a number of some importance since it affects the visual appearance of the representation and to a lesser extent the values of quantities derived from it. To favor

smaller numbers of blocks, γ must be > 1 ; as it increases beyond 1 there are fewer and fewer blocks in the optimal representation. Thus, while it is not explicitly a smoothing parameter, its effect can be mistaken for such.

It is of some use to compute from eq. (92) the expected number of blocks:

$$\langle N_{blocks} \rangle = P_0 \sum_{N_{blocks}=0}^N N_{blocks} \gamma^{-N_{blocks}} \quad (94)$$

The sum can be evaluated to give:

$$\langle N_{blocks} \rangle = \frac{1}{\gamma - 1} + \frac{N + 1}{1 - \gamma^{N+1}} \quad (95)$$

The form in Eq. (92) is not the only prior possible, but it is very convenient to implement, since with the fitness equal to the log of the posterior, one only needs to subtract the constant $\log \gamma$ from the fitness of each block. A few examples will now show how the value of γ can be determined, and demonstrate that, especially with good signal-to-noise, the block representation is not very sensitive to the precise value adopted.

Figure 2 is the result of one such simulation study of BATSE TTE data, using all 523 bursts with at least 1,000 photons. An ordinary histogram of all photons in a burst, with 1024 evenly spaced bins, was taken as the *true signal* for that burst. Then 10 random subsamples of $\frac{1}{8}$ -th as many photons were put through the algorithm, using the maximum likelihood cost function, Eq. (20). The resulting block representation was evaluated at the same 1024 time points as the true signal; the RMS difference between the two was taken as the measure of error. This operation was done for the 32 values of $\ln \gamma$ shown in the figure, and for 10 realizations of the random eightfold downsampling.

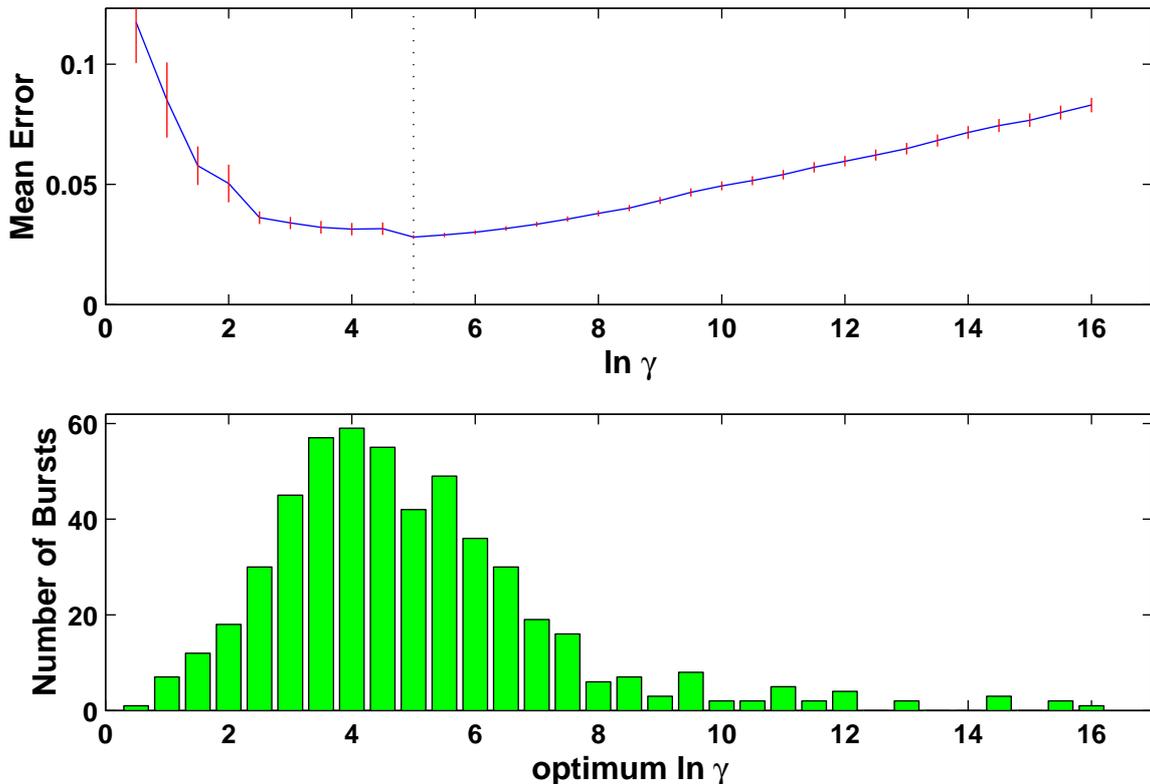


Figure 2: Simulation study for the parameter γ . Top panel: Error, averaged over 10 resamplings of 104 BATSE TTE bursts, *vs.* $\ln \gamma$ (3σ error bars). Bottom panel: Distribution of values of $\ln \gamma$ giving minimum error for the individual bursts.

The error curve (upper panel) is relatively flat, nearly constant for $\ln \gamma$ in the approximate range 2-8, with a nominal optimum at 5. Since the optimal values of γ vary somewhat widely from burst to burst (bottom panel), the flatness of the error curve means that the errors in the block representation of the light curves will not be greatly in error if one adopts a single value of γ for all bursts. Some of this scatter is no doubt due to the dependence of the optimal γ on the number of data cells, but the latter does not vary over enough of a range to allow us to study this effect here.

The effect⁷ of $\log \gamma$ in this and other simulations seems to level off

⁷A large value of this parameter naturally has the effect of reducing the number of blocks, producing a block representation that has less structure – giving a smoother visual appearance. But the parameter is not explicitly

at around 6. We have adopted the value 8 in the examples shown here. A simple argument, due to Mike Nowak, yields $\gamma \approx N$, where N is the number of data points.

These results are given not as a universal result for γ but because the general shape of the curve in Fig. 2 does seem characteristic of a wide variety of situations. We recommend that persons using the algorithm carry out simulations of this kind to study the behavior of the algorithm as a function of γ for their application.

5 Examples

This section presents results using the algorithms given in the Appendix on various sample data sets.

5.1 Determination of the Parameter γ

In applications, one must specify the prior for the number of blocks. The convenient geometric prior described in §4.6 amounts to the assumption that the prior probability of $k + 1$ blocks is a constant factor, namely $\frac{1}{\gamma}$, times that for k blocks. Values of $\gamma > 1$ express the notion that a small number of blocks is *a priori* more likely than a large number.

In principle, the value of γ depends on one's prior knowledge of the number of blocks, but in applications it is rare that one can express this knowledge simply. In this section we perform block analysis of synthetic data where, knowing the correct answer, we can determine the best value.

a smoothing parameter.

5.2 Dynamic Range

One of the goals listed in §1 was that the algorithm have a large dynamic range. Here we give an example meant to demonstrate the dynamic range in both time and amplitude. The synthetic signal is a single block superimposed on a constant background, and the data are a set of points drawn from a distribution with the corresponding shape. The value $\log(\gamma) = 8$ was used, and we adjusted the number of

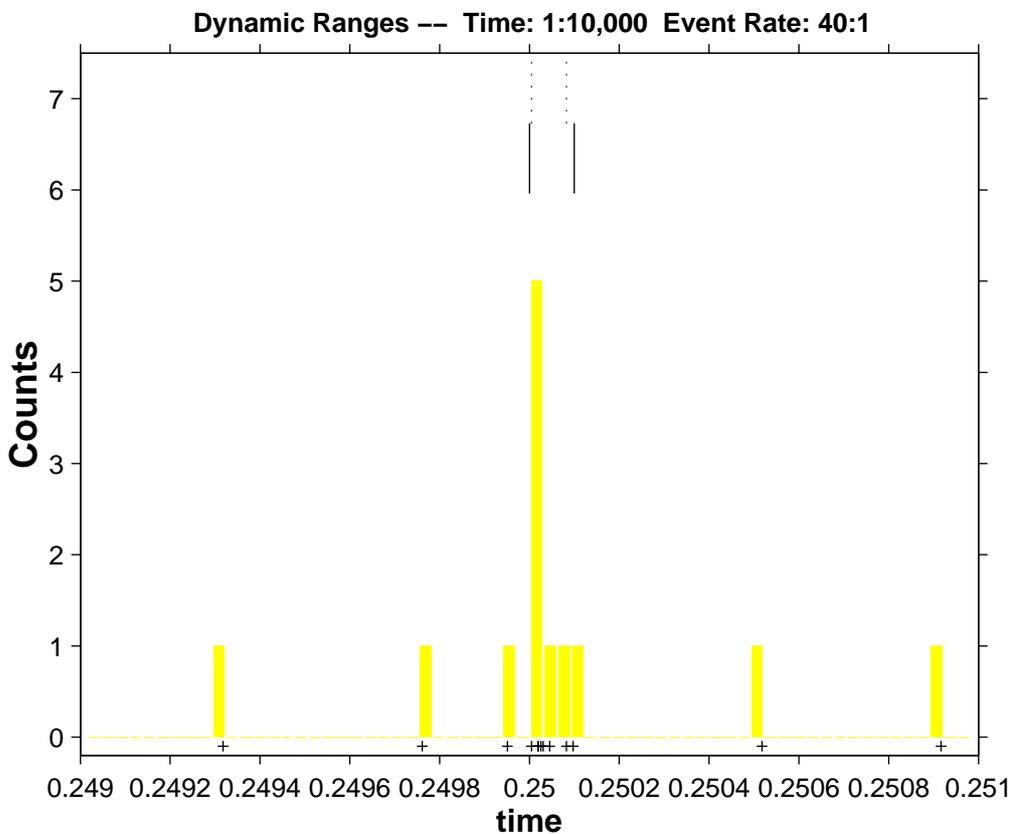


Figure 3: Maximum likelihood segmentation of a synthetic spike: 8 events in .0001 second on a background of 2000 events over the unit interval, only a small fraction of which is plotted. The solid lines at the top of the figure indicate the edges of the actual block; the dotted lines are the two changepoints of the optimal segmentation. The actual points are shown just below the histogram of the raw counts.

events in the spike to be as small as possible and still detect the spike. The errors in the block edges (-4 and +17 microseconds) are just per-

ceptible in the figure. For as few as 4 events the spike was detectable only by making $\log(\gamma) = 4$, and with larger errors.

The next figure depicts a segmentation analysis meant to demonstrate the ability of the algorithm to handle a signal that has a large dynamic range in amplitude. The signal consists of three adjacent blocks on a small, constant background. The middle block has a much smaller amplitude, and the goal is to see if the near presence of large spikes on either side affects its edges. The rates in the spikes are roughly a million times the background rate and several thousand times the rate of the central satellite block. The dotted lines near the top signify the estimated block edges, or changepoints, whereas the solid lines denote the actual edge locations. The errors in the four edge locations are all less than 10^{-8} seconds. Our method is essentially impervious to large amplitude differences within a signal. In fact, increasing the number of counts in the main spikes in this example would only enhance the determination of the edges.

5.3 Point Data Time Series

Figure 6 shows the optimal block decompositions of data for a γ -ray burst based on the point data comprising the TTE data for BATSE trigger 0551 (reference). The value $\log(\gamma) = 8$ was used for the parameter in the prior. This analysis is based on the first 14,000 photon time tags for this burst. The full data set consists of 28,904 photons, but the last half is essentially background. Since the data are time tagged events, we used the form of the posterior given in Equation TBD. Need to compare duplicates allowed with not allowed.

Figure 7 shows the TTE data summed over all four energy channels, analyzed with four different values of the prior parameter $\log \gamma$. The

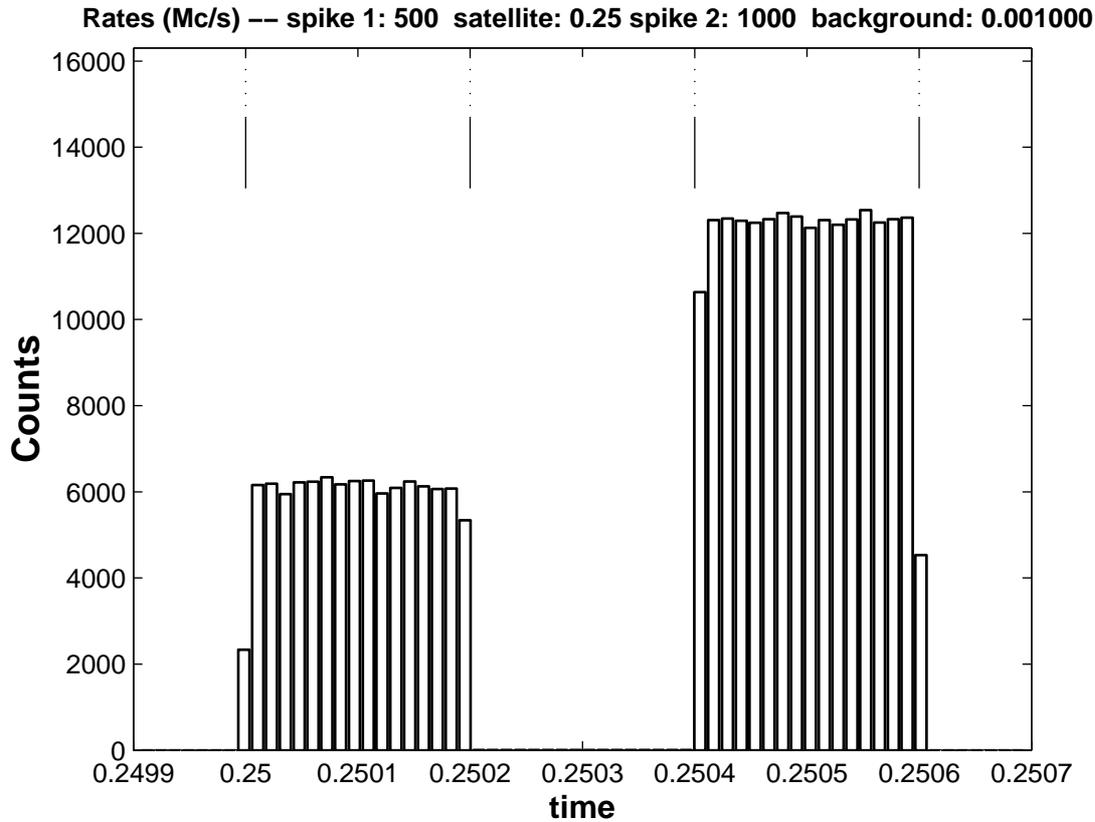


Figure 4: Maximum likelihood segmentation of a set of block with a large range of amplitudes. Each block has a width of 200 microseconds, with 100,000, 50, and 200,000 events, respectively, while the background consists of 1,000 events over the full 1 second interval analyzed. The central block and background are almost imperceptible on the scale of the figure. The analysis parameters are the same as in Fig. 3

first panel corresponds to a flat prior, giving too much prior probability to large numbers of changepoints. The obvious symptom is the appearance of many short spikes, corresponding to narrow intervals in which statistical fluctuations are elevated by the inappropriate prior into apparent significance. These putative features that are probably not real, and—while they represent a small amount of fluence (intensity \times duration), they are cosmetically obnoxious and confound, for example, procedures for automated identification of pulses (local maxima).

The second panel, with a prior that gives lower weight to large

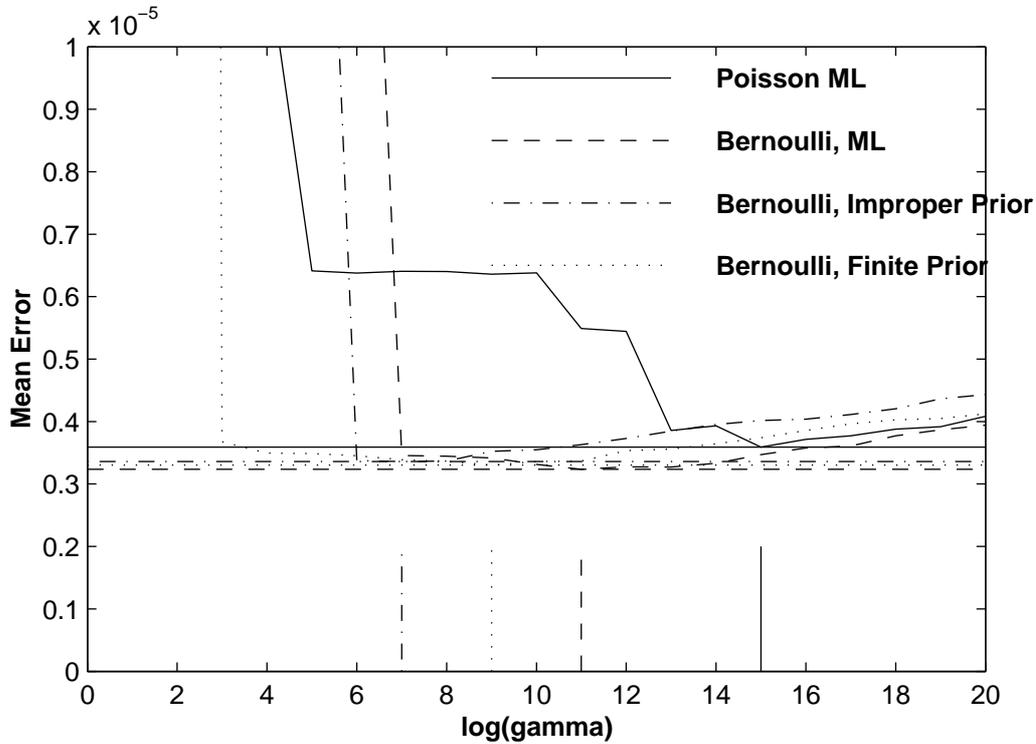


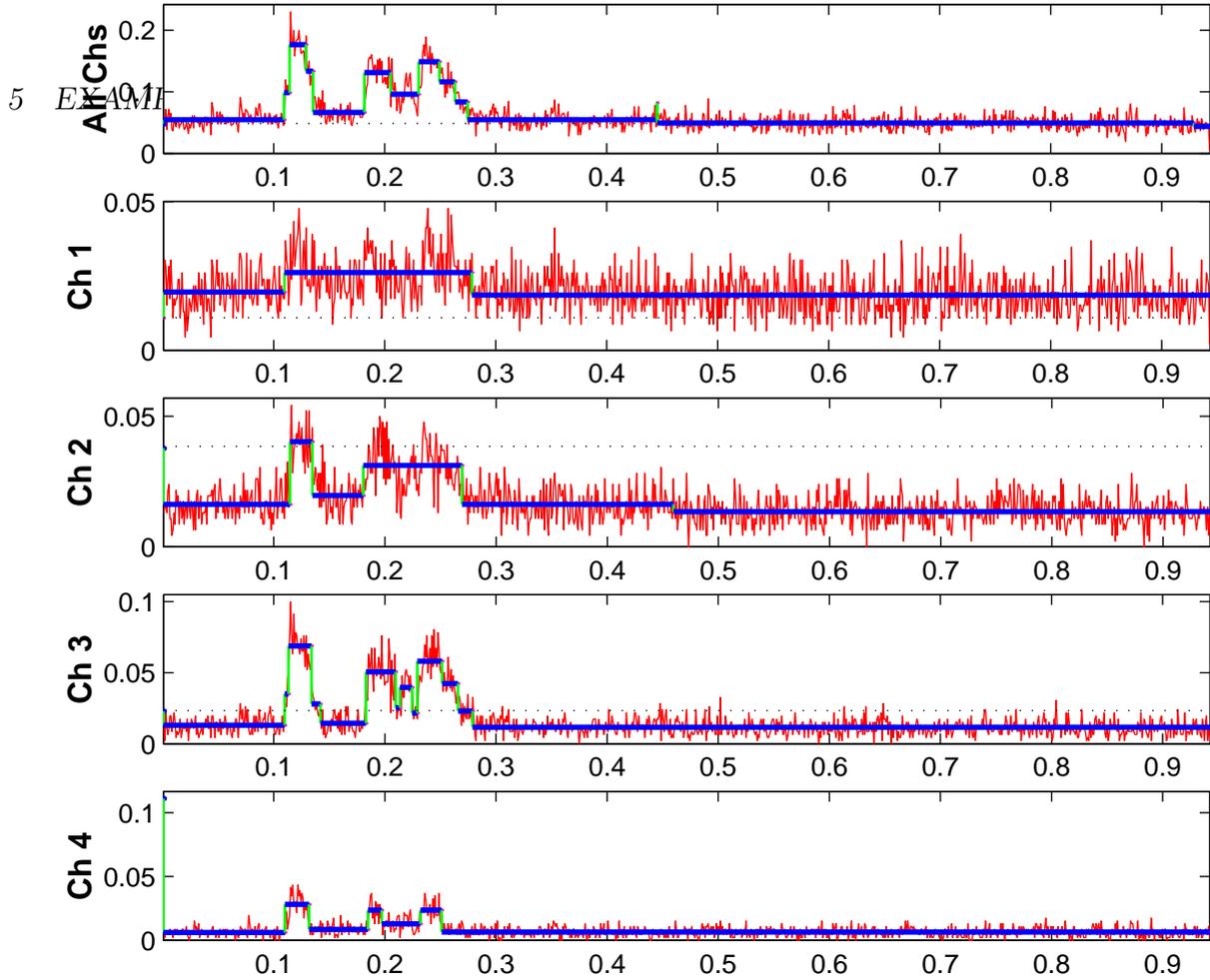
Figure 5: Simulation study, to find optimum value of the parameter $\log \gamma$.

numbers of changepoints has fewer spikes. By the time one reaches $\log \gamma = 4$, there is little change in the representation (*cf.* Figure 12). This result is not necessarily universal, but the figures shown here indicate that the value $\log \gamma = 8$ is quite reasonable and that values somewhat lower or higher would not make any real difference in the final representation.

5.4 Binned Data

Figure 5 shows the block representation for a portion of the light curve of the first burst in the BATSE catalog, observed on April 21, 1991,

Trigger 0551



43

Figure 6: Optimal partitions of BATSE TTE data for Trigger 0551. All photons were used in the top panel; the others are based on the smaller number of photons detected in each of the four BATSE energy channels.

Trigger 0105. These data are available [BATSE www site]⁸ in binned format, with larger bins at the beginning, transitioning to smaller bins at the fiducial trigger time.

The three panels in the figure are for different values of the prior parameter $\log \gamma$. The first case, $\log \gamma = 0$, corresponds to a flat prior. With this rather strong encouragement for a large number of blocks, it is seen that the block representation is identical to the raw binned

⁸BATSE continuously recorded data in time bins 1.024 seconds long, and the time series posted on the web has 116 seconds of such low-resolution data pre-pended to the 16 times higher (64 millisecond bins) resolution data starting at the fiducial trigger time. To make the bins equal, the numbers given on the web site apportion the counts in each large bin into 16 small bins. Since our analysis can handle unequal bins, we have undone this, and reconstructed the actual integer counts in the larger bins.

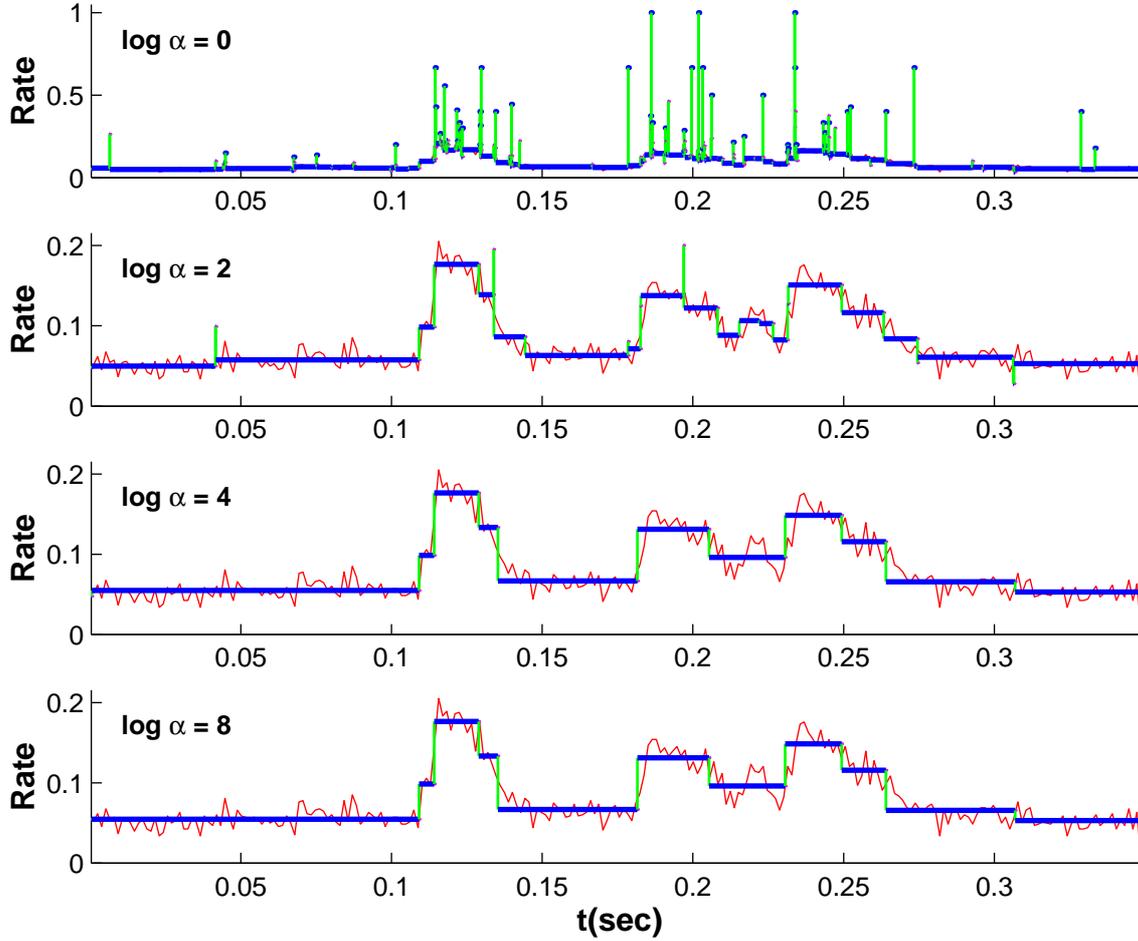


Figure 7: Optimal partitions of BATSE TTE data for Trigger 0551. Same as the first panel of Figure 3, except that four different values of $\log \gamma$ were used: 0, 2, 4 and 8

data. Even the coarse pre-trigger bins that seem to be combined into large blocks because their event rates are so similar, are represented as separate blocks.

The second panel, $\log \gamma = 8$, corresponds to the best choice for the parameter, and can here be taken as the best block representation of these data. The last panel, $\log \gamma = 16$, corresponds to too much of a penalty against a large number of blocks. One notes that the most intense peak, which is resolved into two peaks in the other panels, is here a single peak.

Finally, for comparison in Figure 6, we show analyses of the same data, unbinned, binned, and time-to-spill, for BATSE Trigger 0551. This figure was created with the MatLab code included in the Appendix

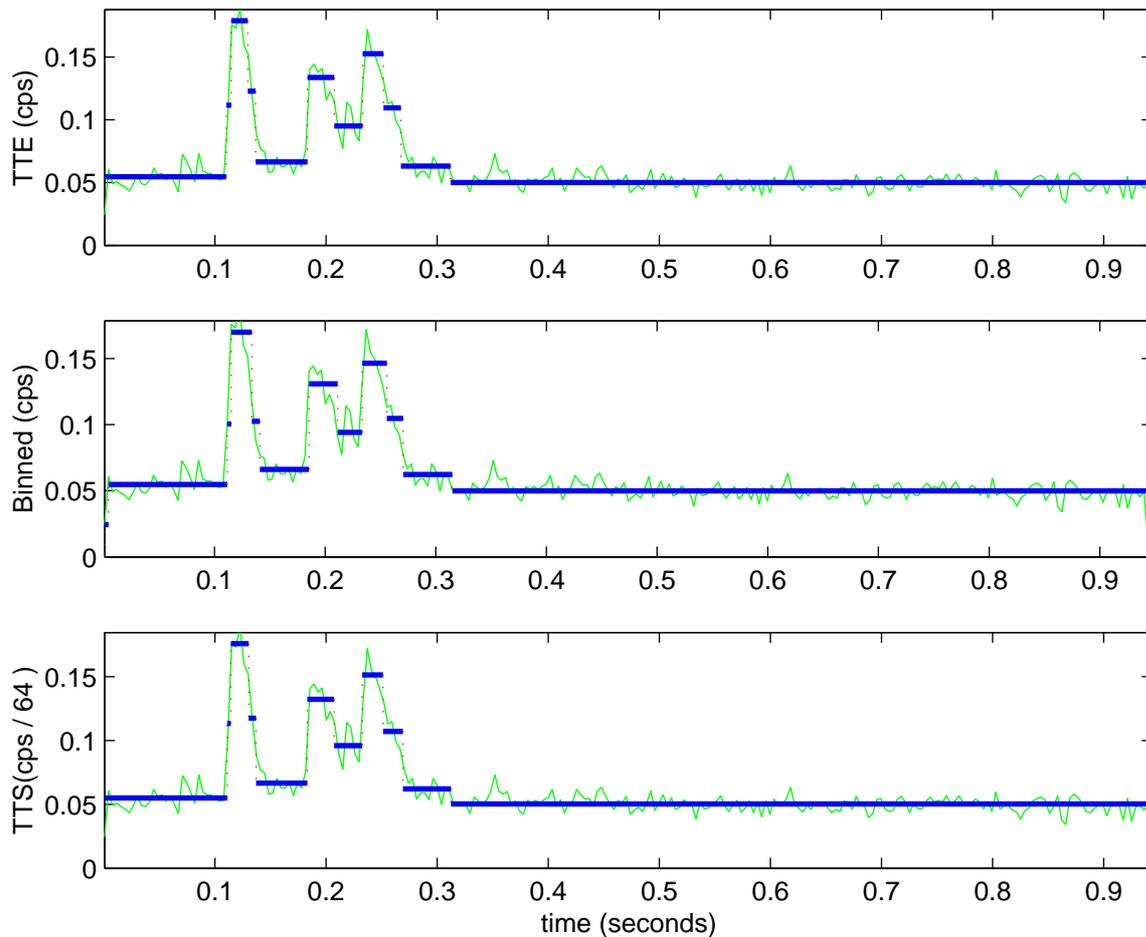


Figure 8: BATSE Trigger 0551. Top: TTE data. Middle: same data, binned into 256 bins. Bottom: same data, converted to time-to-spill data with $S = 64$.

and available electronically. Note that the results for the three different data modes are nearly identical, except for details of the first pulse.

5.5 Maximum Likelihood Histograms

One may obtain data in order to study the probability distribution of the measured variable. The histogram [Silverman 1999, Scott 1992] is perhaps the simplest estimator for the underlying distribution function,⁹ but is known to be sensitive to the choice of bins. Procedures for choosing the number of bins, such as Sturges' rule or Kevin Knuth's entropy-based approach [Knuth 2005], and for eliminating sensitivity to bin locations, such as the *average shifted histogram*, have been designed to achieve various goals [Scott 1992].

Almost always the bins are taken, somewhat arbitrarily, to be equal in size. A simple way to achieve greater resolution where the data warrant it, sometimes called the *equi-depth histogram*, is to make each bin contain the same number of points – leaving the problem of choosing this number.

Note that if the measurements are arranged in increasing order, one can think of the underlying probability density as a signal, and use segmentation ideas to model and estimate it. In particular, a straightforward piecewise constant density estimate provides a data-adaptive histogram in which the number and location of the bins are determined by the data themselves. The fitness functions derived above for independently distributed point data are appropriate for any histogram estimation, since counts of events in intervals are governed by the Poisson distribution. If one has little or no information about the measurement errors or the smallest measurable difference (the *quantum of the measurement*, cf. §4.1), as is often the case, the maximum likelihood fitness function in Eq. (20) is appropriate. Its invariance

⁹However, the *empirical cumulative distribution function* is also very useful and can be evaluated almost trivially and nonparametrically [Scott 1992].

property makes specification of scale of the measured variable or its quantum unnecessary.

It remains to specify the prior on the number of blocks. Figure (9) depicts the results of a Monte Carlo study of maximum likelihood histograms of synthetic data from a Poisson process which changes rate at known locations. The error of the representation was computed

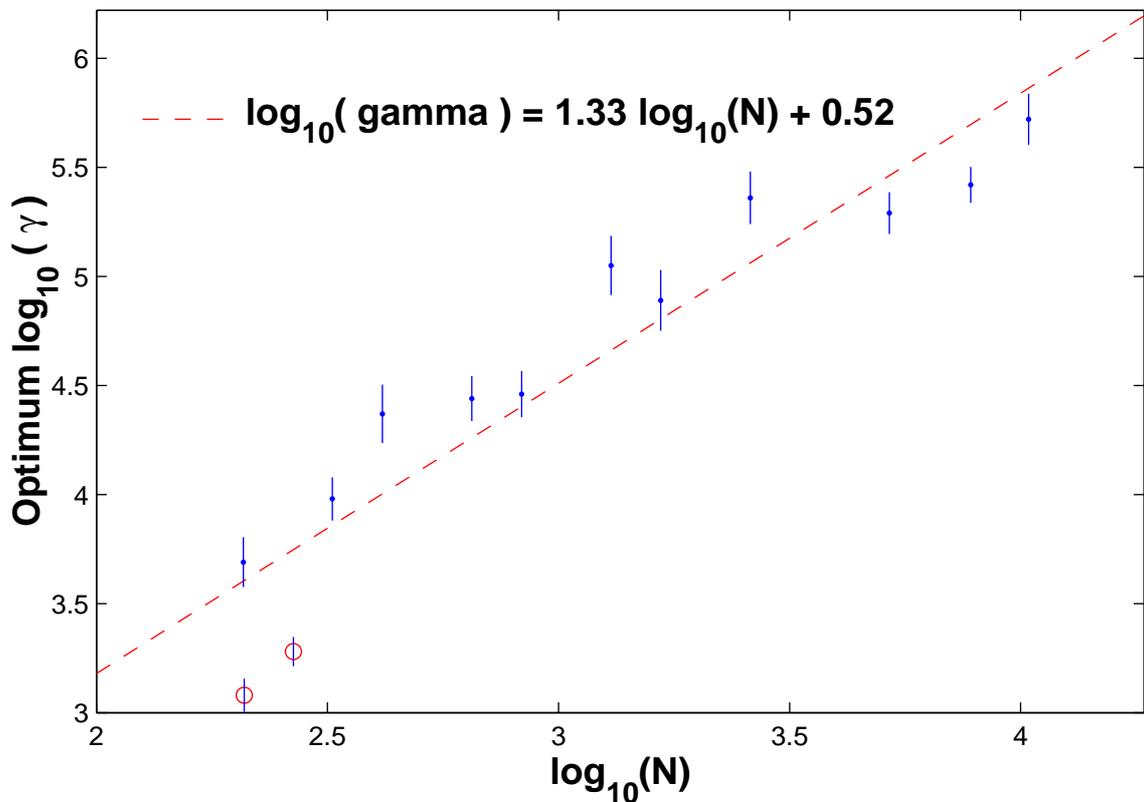


Figure 9: Simulation study for histograms.

by comparing the number and location of the actual and estimated changepoints, with the result that the optimum value of the parameter $\log(\gamma)$ in the geometric prior can be determined. This kind of study is only valid for the specific problem simulated, but it is reasonable to assume that the results would not be drastically different for other situations. It is natural that the optimum parameter depend on the

number of data points, and the figure shows an empirical relationship.

Figure 10 histograms of readily available data on the durations of eruptions of the Old Faithful geyser in Yellowstone National Park. The top part of the figure is based on 107 measurements from [Weisberg 1980], also found in Table 2.2 of [Silverman 1999]. The latter author uses these data to demonstrate a number of existing density estimation methods, including conventional fix-bin histograms, the *naive estimator*, kernel estimators with kernels of various widths, the *nearest neighbor method*, variable kernels, orthogonal series estimators, maximum penalized likelihood estimators, *general weight function estimators*, and others. It is dangerous to use data from a phenomenon that is not understood in detail. In particular, here we do not know what the true distribution is. But it may be instructive to compare our segmentation results with those in textbooks such as [Silverman 1999]. The main difference in the upper panel of Figure 10 is that the maximum likelihood histogram is consistent with a bimodal distribution consisting of two rather flat components; with the ordinary histogram, while the bimodality may be pretty secure, the shapes of the components are ambiguous. The histograms in the lower panel, with more than twice as much data, more or less confirm the correctness of our description based on the data in the upper panel.

5.6 Measurements with Normal Errors

First consider a simulation consisting of measurements at arbitrary times in an interval. These variates are taken to be zero-mean-normal, except over an unknown sub-interval where the mean is instead an unknown nonzero constant. Figure 11 shows synthetic data for three simulated realizations with different values for this constant. The solid line is the Bayesian blocks representation, using the posterior in Eq. (75). For the smallest amplitude (first panel), no changepoints are found and so the signal is completely missed. In the next panel, the solution is correct except that the second changepoint is one point too early, while the solution in the third panel gets both changepoints correct.

In this experiment the points are evenly spaced, but only their order matters, so the results would be the same for arbitrary spacing of the data points.

A recent paper [Arias-Castro, Donoho and Huo 2003] on multiscale methods discusses essentially the same problem and develops several theorems for the asymptotic behavior of optimal detectors of such signals. To quote these authors, “In short, we can efficiently and reliably detect intervals of amplitude roughly $\sqrt{2\log N}$, but not smaller.” Figure 4 reports some results of detection of the same normally distributed step-function process shown in Figure 11. The solid lines show the root-mean-square residuals from the true function, while the dashed line

This figure generally confirms this theoretical result, since the errors (both a measure of the errors in the number and location of the changepoints) are

5.7 Real Time Analysis: Triggers

Because of its incremental structure, our algorithm is well suited for real-time analysis. Starting with a small amount of data, the algorithm typically finds no changepoints. But at each step, by determining the optimal partition up to and including the most recently added data cell, the algorithm effectively tests for the presence of the first changepoint¹⁰. If the real time mode is selected, our algorithm halts at the first significant changepoint and reports its location. Of course other halting conditions are possible, *e.g.* that a changepoint be found and its position remain stable for a fixed number of steps.

The real time mode can detect the presence of a time-dependent signal rising significantly above a slowly varying background. For example, in a photon stream the resulting *trigger* indicates the presence of a new bursting or transient source.

The usual way to approach this and similar problems is to report a detection if and when the actual event rate, averaged over some interval, exceeds one or more pre-set thresholds. See [Band 2002] for an extensive discussion, as well as [Fenimore *et al.* 2001, McLean *et al.* 2003, Schmidt 1999] for other applications in high energy astrophysics. One must consider a wide range of configurations: “BAT uses about 800 different criteria to detect GRBs, each defined by a large number of commandable parameters. [McLean *et al.* 2003]”. Both the size and locations of the intervals over which the signal is averaged affect the result, and therefore one must consider many different values of the corresponding parameters. The idea is to minimize the chances of missing a signal because, for example, its duration is poorly matched to the interval size chosen. If the background is determined dynamically, by

¹⁰It is rare, but not impossible, that this first indication of change yields more than one changepoint.

averaging over an interval in which it is presumed there is no signal, similar considerations apply to this interval.

In principle, our segmentation algorithm greatly simplifies the above considerations, since predefined bin sizes and locations are not needed, and the background is automatically determined in real time. In practice, the situation is not so simple. If one lets the data stream accumulate continuously, the N^2 dependence of the compute time eventually makes the computations unfeasible, so in practice it is necessary to adopt a finite window size, and only analyze the data in this sliding window. But slow variations in the background in many cases would mandate something like a sliding window.

Because of additional complexities, such as accounting for background variability and the Pandora's box that spectral resolution opens [Band 2002], we will defer a serious treatment of triggers to a future publication.

We end this discussion with a brief discussion of a simple topic, of an issue that can be relatively easily discussed, namely the *false alarm* (also called *false positive*) – no signal is present but a noise fluctuation passes the algorithm's detection criteria. Unavoidably a detection procedure embodies a trade-off between sensitivity and false alarms. Other things being equal, making an algorithm more able to detect weak signals renders it more sensitive to noise fluctuations. Making an algorithm avoid noise fluctuations renders it insensitive to weak signals as well. In applications one typically chooses the balance of these competing factors based on some notion of the relative importance of avoiding false positives and not missing weak signals. Hence there can be no universal prescription.

Identification of a good algorithm may include adopting criteria that

are relatively more sensitive to signal and less to noise, perhaps making use of prior information about the nature of both target signals and observational noise. And typically a given algorithm contains one or more parameters that can be adjusted, empirically or otherwise, to achieve improved selectivity. A simple example: detection occurs when the data exceed a threshold, the value of which is the adjustable parameter.

The corresponding parameter in our algorithm is $\log(\gamma)$. In the real-time mode, this parameter has a simple interpretation: it fixes the value of the prior odds ratio for triggering (vs. not triggering)

$$\log \frac{P(\text{two blocks})}{P(\text{one block})} = -\log(\gamma) \quad (96)$$

Figure 13 quantifies the false alarm rate as a function of this parameter, from analysis of signal-free Poisson noise¹¹. The maximum likelihood cost function was used, and its scale independence means that only the number of random photons in each input interval, here denoted N , matters, and not the photon rates—as approximately confirmed in this figure. The increased scatter for large values of $\log \gamma$ is due to the small number of tabulated false positives. The figure can be used to set the value of $\ln(\gamma)$ to achieve a desired maximum false alarm rate.

¹¹The rates plotted are the number of detections divided by the number of photons analyzed. This denominator is less than the number of simulated photons, due to the termination of the algorithm upon triggering.

6 Appendix A: MatLab Code

This section contains MatLab¹² code for the analysis tools. The function `fit` evaluates the natural logarithm of the fitness function, and `reverse` reverses the order of an array. The quantity `eps` is the smallest number representable on the current machine. All other constructs and functions are standard MatLab.

6.1 Main Program

This program computes two different segmented representations of BATSE data for a gamma-ray burst. These code listings can be used to recreate Figure 8, as well as verifying all of the other code modules. The time-tags of the photon detections are the raw data, and in (2) and (3) these same times are binned and converted to time-to-spill, respectively. The array `cell_sizes` contains the widths of each bin, which need not be equal as they are in this example.

```
% test_global.m
%-----
% Optimal segmentation, for three data modes:
% (1) TTE data (2) binned data (3) TTS data
%-----

    first = 0;          % Retrospective (not real-time) mode
tick2sec = .000002; % convert 2 microsecond ticks to seconds

%-----
% Load BATSE TTE Data; make histogram
%-----

[ tt, channels, detectors ] = load_ttedata( 'tteascii.00551' );
min_tt = min( tt ); max_tt = max( tt );
bins = linspace( min_tt, max_tt, 256 );
dt_bins = bins(2) - bins(1); % bin width
xx = hist( tt, bins );      % make binned data
```

¹²© the Mathworks, Inc.

```

%-----
%   Optimal segmentation: raw TTE Data
%-----

max_delt = 1; % Max separation of time tags
data_type = 1;
bin_size = 2; % 2 microsecond ticks
data_cells = make_cells( tt', data_type, max_delt, bin_size );
ncp_prior = 8;
cpv_1 = global_optimum( data_cells, data_type, ncp_prior, first );

subplot(3,1,1)
plot( tick2sec * bins, xx/dt_bins, '-g' ) % plot binned data for reference
hold on
[ ii_pulses, count_vec ] = plot_tte( tt, tt( cpv_1 ), 1 );
v = axis;
v(1) = tick2sec*min_tt;
v(2) = tick2sec*max_tt;
axis(v)
ylabel('TTE (cps)')

%-----
%   Optimal segmentation: binned version of same data
%-----

cell_sizes = dt_bins * ones( size( xx ) );
data_cells = [ cell_sizes; xx; ]';
cost_type = 2;
cpv_2 = global_optimum( data_cells, cost_type, ncp_prior, first );

subplot(3,1,2)
cell_begin = min_tt + cumsum( cell_sizes ) - cell_sizes(1);
min_height = plot_partition( cpv_2, xx, cell_sizes, cell_begin, xx );
ylabel('Binned (cps)')

%-----
%   Optimal segmentation: TTS version of same data
%-----

ss_tts = 64;
tts_data = tt( 1:ss_tts: length( tt ) );
data_cells = make_cells( tts_data', data_type, max_delt, bin_size );
data_cells( :, 2 ) = ss_tts * ones( size( tts_data ) );
ncp_prior = 8;
cpv_3 = global_optimum( data_cells, data_type, ncp_prior, first );

```

```
subplot(3,1,3)
plot( tick2sec * bins, xx/(ss_tts*dt_bins), '-g') % plot binned data for reference
hold on
[ ii_pulses, count_vec ] = plot_tte( tts_data, tts_data( cpv_3 ), 1 );
v = axis;v(1) = tick2sec*min_tt;v(2) = tick2sec*max_tt;axis(v)
ylabel('TTS(cps / 64 )');xlabel('time (seconds)')

set(gcf,'PaperPosition', [ 0 0 7 4.5 ] )
print -depsc2 C:\global_paper\ttebin3.epsc2
```

6.2 Construct Data Cells

This routine constructs data cells, taking into account time tags that are identical or so close that they are to be assigned to the same cell, and similar details. For binned data, the routine simply constructs the matrix of the proper data elements.

```

function data_cells = make_cells( tt, data_type, max_delt, bin_size )
%-----
% Make data cells from data - for input to global optimization
%
% Input:          tt -- array of time tags or bin counts
%
%          max_delt -- maximum separation of times
%                   dt <= max_delt: in same cell
%                   dt > max_delt: in different cells
%
%          data_type -- cell type: 1: midpoints (~Voronoi)
%                           2: intervals (~Delaunay)
%                           3: bins
%
%          (dt = 0 corresponds to duplicate time tags)
%
% Output: cell_pops -- array of cell populations
%
%          cell_sizes -- array of cell sizes
%
% NB: length of data_type 2 output is one smaller than of data_type 1
%-----

if data_type == 3

    % binned data

    [ aa, bb ] = size( tt );
    if aa > bb; tt = tt'; end
    [ bin_flag, num_data ] = size( tt ); % force row vectors

    if bin_flag == 1
        cell_pops = tt;
        cell_sizes = bin_size * ones(size(tt));
    elseif bin_flag == 2
        cell_pops = tt(1,:); % bin populations
        cell_sizes = tt(2,:); % bins sizes
        disp('2')
    else

```

```

        error('Incorrect matrix dimensions in global_optimum.m ...')
    end

else

    % TTE data; interpret bin_size as time quantum ("tick")

    tt = fix( tt / bin_size );
    cell_pops = ones( size( tt ) ); % Initial: one datum per cell

    %-----
    % Find clumps of points closer together than max_delt
    %-----

    ii_close = find( diff( tt ) < max_delt );

    while ~isempty( ii_close )

        ii_start = ii_close(1); % Beginning of clump

        %-----
        % Index of end of the clump:
        % all ii_close-indices up to but not including
        % ii_beyond are in clump
        %-----

        ii_beyond = find( diff( ii_close ) > 1 );

        if isempty( ii_beyond )
            % All remaining close points are in the clump
            ii_end = ii_start + length( ii_close );
        else
            ii_end = ii_start + ii_beyond(1);
        end

        ii_clump = ii_start:ii_end;

        clump_pop = sum( cell_pops( ii_clump ) );
        clump_tt = mean( tt( ii_clump ) );

        % put members of the clump in one cell:
        cell_pops( ii_start ) = clump_pop;
        tt( ii_start ) = clump_tt;

        % null the cells evacuated by this operation:
        for ind = ii_end:-1:ii_start + 1

```

```

        cell_pops( ind ) = [];
            tt( ind ) = [];

    end

    ii_close = find( diff( tt ) < max_delt );

end

if data_type == 1

    %-----
    % midpoints
    %-----

    dt = diff( tt );
    ndt = length( dt );

    cell_sizes = 0.5 * ( dt(1:ndt-1) + dt(2:ndt) );

    dt_left = dt(1);
    dt_rite = dt(ndt);

    cell_sizes = [ dt_left cell_sizes dt_rite ];

elseif data_type == 2

    %-----
    % intervals
    %-----

    cell_pops( length( cell_pops ) ) = []; % last datum can't define cp!
    cell_sizes = diff( tt );

end

end % if data_type

data_cells = [ cell_sizes; cell_pops ]';

```

6.3 Global Optimum

This function implements the dynamic programming procedure which is the heart of the segmentation process. The input variable `data_cells` is an ordered array containing the sequential data. In many cases it comprises two arrays (the sufficient statistics): the numbers of events in (often 1) and the sizes of the data cells. The integer `cost_id` identifies which cost function is to be used, as detailed in the comments for the function `log_cost`. The parameter `lgam` is a real number, typically approximately 8, expressing prior information about the number of changepoints likely to occur. Finally, the input parameter `first` is simply a flag to indicate whether the routine should return when it first encounters a changepoint as it sweeps through the data in sequence. If this trigger mode is not to be used, the parameter and relevant `if` statement can be removed.

The main output is the array `cpt` giving the index values (in the input array `data_cells`) at which the changepoints occur. The convention is that the values in `cpt` give the data cell which starts a block, so that the previous block ends at this value minus 1. The arrays `last` and `best` are of value only for debugging or diagnostic purposes. The index `R` is of use only in the trigger mode, and indicates the segment of the input data array that was processed when the (first, and in this case only) changepoint was detected.

For interactive operation with large data arrays, it is sometimes useful to insert an output statement indicating progress within the `R` loop.

```

function [ cpt, last, best, R ] = global_optimum( data_cells, cost_id, lgam, first )
%=====
%   Find the optimum partition of sequential data
%-----
%
% Input: data_cells -- sequential data array, a N x M array:
%               * column index: the N data cells (each row is one cell)
%               *   row index: the M parameters to compute the cost function
%
%               cost_id -- indicates cost function (see log_cost.m)
%
%               lgam -- log of parameter in geometric prior for number of changepoints
%
%               first -- 0 --> normal "retrospective" mode; analyze all data
%                       >0 --> trigger mode; return at first sign of a change
%
%-----
%
% Output: cpt -- array of change points (index values for input array)
%
%         last -- working array of indices\
%                |-- for diagnostic purposes only
%         best -- working array of optima /
%         R -- for the realtime mode, this is how much data was processed
%-----

[ num_cells, num_parameters ] = size( data_cells );

best = []; % "best(R)" is the value of the optimum at iteration R
last = []; % "last(R)" is the index at which this optimum occurs

%-----
%   Start with the first datum (R=1);
%   add the next one at each iteration
%-----

for R = 1:num_cells

    if R == 1
        qq = data_cells(R:-1:1, :);
    else
        qq = cumsum( data_cells(R:-1:1, :) );
    end
end

```

```
[ best(R), last(R) ] = max( [0 best] + ...
    reverse( log_cost( qq, lgam, cost_id )' ) );

if first > 0 & last(R) > first
    % Trigger on first significant changepoint
    cpt = last(R);
    return
end

end

%-----
% Find the optimum partition
%-----

index = last( num_cells ); cpt = [];

while index > 1

    cpt = [ index cpt ];
    index = last( index - 1 );

end

end
```

6.4 Load TTE Data

This routine reads the data from the BATSE files as posted at the data archive at NASA's High Energy Astrophysics Science Archive Research Center (HEASARC): ftp://coss.c.gsfc.nasa.gov/pub/data/batse/ascii_data/batse_tte/.

```
function [ times, channels, detectors, trig_time ] = load_ttedata( file_name )
% Open and read data from BATSE TTE files

[ fid message ] = fopen( file_name, 'r' );

if fid == -1
    fprintf(1, [ 'Error opening file ' file_name '\n' ] ); return
else
    fprintf(1, [ 'Successfully OPENED file ' file_name '\n' ] )
end

%-----
% Read the File Header (5 lines)
%-----
format1 = '%s';
for ijk = 1:5
    aa(ijk).line = fgetl( fid );
end

trig_time = aa( 1 ).line;
trig_time = eval( trig_time( 38: length( trig_time ) ) );

npts = aa( 2 ).line;
npts = eval( npts( 9: length( npts ) ) );

%-----
% Now read the data: times, channels, detectors
%-----

[ times, count_times ] = fscanf(fid, '%f', npts);
[ channels, count_channels ] = fscanf(fid, '%f', npts);
[ detectors, count_detectors ] = fscanf(fid, '%f', inf);

fclose( fid ); % close the file
```

6.5 Logarithm of the Cost Function

This code segment computes any of the cost functions discussed above. To maintain the block additivity of the cost function used by the optimization algorithm, the logarithm is computed.

```

function cost = log_cost( cell_data, ncp_prior, cost_type )
%-----
% Log of cost function for various data types
%
% Input: cell_data -- MatLab structure containing these arrays:
%           cell_sizes -- size of each cell
%           cell_pops  -- number of events in each cell
%           ncp_prior  -- complexity parameter (from prior on number of changepoints)
%           cost_type  -- 1 for TTE data; 2 for binned data, etc.
%
% Output: cost -- array of corresponding logarithmic cost function
%-----

global lam_11 lam_22 p_11 p_22 log_lam
global alpha_0 beta_0

if cost_type == 1

    cell_sizes = cell_data( :, 1 );
    cell_pops  = cell_data( :, 2 );

    %-----
    %   TTE data
    %-----
    arg = cell_sizes - cell_pops + 1;

    ii = find( arg > 0 );
    num_bad = length( cell_sizes ) - length( ii );

    if num_bad == 0
        cost = gammaln( cell_pops + 1 ) + gammaln( arg ) ...
              - gammaln( cell_sizes + 2 );
    else

        cost = eps * ones( size( cell_pops ) ); % eps is smallest number

        cost(ii) = gammaln( cell_pops(ii) + 1 ) + gammaln( arg(ii) ) ...
                  - gammaln( cell_sizes(ii) + 2 );
    end
end

```

```

end

elseif cost_type == 2 % Binned data, infinite prior

    cell_sizes = cell_data( :, 1 ); % (number of bins in the block)
    cell_pops  = cell_data( :, 2 ) + 1; % Note offset!
    cost = gammaln( cell_pops ) - cell_pops .* log( cell_sizes );

elseif cost_type == 3

    % Normally distributed measurements

    sum_x_2    = cell_data( : , 1 ); % sum( x.^2 / sig^2 )
    sum_x_1    = cell_data( : , 2 ); % sum( x    / sig^2 )
    sum_x_0    = cell_data( : , 3 ); % sum[ 1    / sig^2 )

    cost = 0.5 * log( pi ) ... % this can be absorbed into log_prior
           - 0.5 * log( sum_x_0 ) ...
           + ( ( sum_x_1 ) .^ 2 ) ./ ( 4 * sum_x_0 ) ...
           - sum_x_2;

elseif cost_type == 4 % Binned data, conjugate prior

    cell_sizes = cell_data( :, 1 ); % (number of bins in the block)
    cell_pops  = cell_data( :, 2 ) + alpha_0; % Note offset!
    cost = gammaln( cell_pops ) - cell_pops .* log( cell_sizes + beta_0 );

elseif cost_type == 5 % Binned data, finite prior

    cell_sizes = cell_data( :, 1 ); % (number of bins in the block)
    cell_pops  = cell_data( :, 2 ) + 1; % Note offset!
    term_2 = gammainc( lam_22 * cell_sizes , cell_pops );
    term_1 = gammainc( lam_11 * cell_sizes , cell_pops );

    cost = gammaln( cell_pops ) + ...
           log( term_2 - term_1 ) - cell_pops .* log( cell_sizes ) ...
           - log_lam;

elseif cost_type == 10

    cell_sizes = cell_data( :, 1 );

```

```

cell_pops = cell_data( :, 2 ) + 1; % Note offset!

%-----
% TTE data, multiple hits allowed, finite prior
%-----

gam_term = gammainc( lam_22 * cell_sizes, cell_pops ) - ...
           gammainc( lam_11 * cell_sizes, cell_pops );

cost = log( gam_term ) - cell_pops .* log( cell_sizes ) - log_lam;

elseif cost_type == 11

    cell_sizes = cell_data( :, 1 );
    cell_pops = cell_data( :, 2 ) + 1;

    %-----
    % TTE data; finite prior
    %-----
    arg = cell_sizes - cell_pops + 2; % NB 2, not 1
    ig = find( arg > 0 );
    qq_21 = eps * ones( size( arg ) );

    qq_21(ig) = betainc( p_22, cell_pops(ig), arg(ig) ) - ...
               betainc( p_11, cell_pops(ig), arg(ig) );
    ii_bad = find( qq_21 == 0 );
    qq_21( ii_bad ) = eps*ones( size(ii_bad) ); % overflow

    term_1 = eps * ones( size( arg ) );
    term_1(ig) = betaln( cell_pops(ig), arg(ig) );
    cost = term_1 + log( qq_21 ) - log_lam;

elseif cost_type == 12

    %-----
    % TTE data; max likelihood
    % instead of marginalization
    %-----
    small = eps; % 2.2e-16
    cell_sizes = cell_data( :, 1 );
    cell_pops = cell_data( :, 2 );
    prob = cell_pops ./ cell_sizes;

```

```

    cost = cell_pops .* log( prob + small ) ...
           + ( cell_sizes - cell_pops ) .* log( 1 - prob + small );

elseif cost_type == 13

    %-----
    % Maximum Likelihood
    % Any Poisson data: duplicate tags ok
    %-----

    small = eps; % smallest number
    cell_sizes = cell_data( :, 1 );
    cell_pops = cell_data( :, 2 );

    cost = cell_pops .* ...
           ( log( cell_pops + eps ) - log( cell_sizes ) - 1 );

end

cost = cost - ncp_prior; % prior on number of changepoints

```

6.6 Plot partitions

6.7 Plot TTE partitions

6.8 Reverse (from *WaveLab*)

```

function r = reverse(x)
% reverse -- Reverse order of elements in 1-d signal
% Usage
%   r = reverse(x)
% Inputs
%   x    1-d signal
% Outputs
%   r    1-d time-reversed signal
%
% See Also
%   flipud, fliplr
%
    r = x(length(x):-1:1);

%
% Copyright (c) 1993. David L. Donoho
%

```

```
%  
% Part of WaveLab Version .701  
% Built Tuesday, January 30, 1996 8:25:59 PM  
% This is Copyrighted Material  
% For Copying permissions see COPYING.m  
% Comments? e-mail wavelab@playfair.stanford.edu  
%
```

7 Bibliography

References

- [Arias-Castro, Donoho and Huo 2003] Arias-Castro, E., , Donoho, D., and Huo, X. 2003, 'Near-Optimal Detection of Geometric Objects by Fast Multiscale Methods,' preprint.
- [Band 2002] Band, D. (2002), "A Gamma-Ray Burst Trigger Toolkit," *Astrophys.J.*, **578**, 806-811 (arxiv.org/abs/astro-ph/0205548)
- [Bretthorst 1988] Bretthorst, G. Larry (1988), *Bayesian Spectrum Analysis and Parameter Estimation*, Lecture Notes in Statistics, Springer-Verlag. <http://bayes.wustl.edu/>
- [Coram 2002] Coram, Marc, (2002), personal communication and Ph. D. thesis, Nonparametric Bayesian Classification, www-stat.stanford.edu/~mcoram/
- [Donoho 1994] Donoho, D.L., (1994), Smooth Wavelet Decompositions with Blocky Coefficient Kernels, in *Recent Advances in Wavelet Analysis*, L Schumaker and G. Webb, eds., Academic Press, pp. 259-308.
- [Fenimore *et al.* 2001] Fenimore, E., Palmer, D., Galassi, M., Tavenner, T., Barthelmy, S., Gehrels, N., Parsons, A., Tueller, J. (2001), "The Trigger Algorithm for the Burst Alert Telescope on Swift," in *Gamma-Ray Burst and Afterglow Astronomy 2001*, Ricker and Vanderspek (eds), AIP, **662**, 491, astro-ph/0408514
- [Gelman] Book by Gelman.
- [Hubert, Arabie, and Meulman 2001] Hubert, L., Arabie, P., and Meulman, J., 2001, *Combinatorial Data Analysis: Optimization by Dynamic Programming*, SIAM: Philadelphia
- [Howell 2005] http://www.uvm.edu/~dhowell/StatPages/More_Stuff/OldFaithful.html
- [Knuth 2005] Preprint.
- [Kolaczyk 1996] Kolaczyk, Eric D., (1996), "Estimation of Intensities of Inhomogeneous Poisson Processes Using Haar Wavelets," Technical Report 436, Department of Statistics, The University of Chicago, Chicago, to be submitted to *Journal of the Royal Statistical Society, Series B.*
- [Kolaczyk 1999] Kolaczyk, E.D. (1999). Bayesian Multi-Scale Models for Poisson Processes. *Journal of the American Statistical Association*, 94, 920-933.
- [Kolaczyk 2000] Kolaczyk, E.D. and Dixon, D.D. (2000). Nonparametric estimation of intensity maps using Haar wavelets and Poisson noise characteristics. *The Astrophysical Journal*, 534:1, 490-505.
- [Kolaczyk 1998] Kolaczyk, E.D. (1998) Wavelet Shrinkage Estimation of Certain Poisson Intensity Signals Using Corrected Thresholds. *Statistica Sinica*, 9, 119-135.
- [Kolaczyk 1998] Kolaczyk, E.D. (1997) Non-Parametric Estimation of Gamma-Ray Burst Intensities Using Haar Wavelets. *The Astrophysical Journal*, Vol. 483, 340-349.

- [Kolaczyk and Nowak 2002] Kolaczyk, Eric D. and Nowak, Robert D., (2002), “Multiscale Statistical Models,” Penn State Statistical Challenges 3
- [Kolaczyk and Nowak 2002] Kolaczyk, Eric D. and Nowak, Robert D., (2002), “A Multiresolution Analysis for Likelihoods: Theory and Methods,” preprint
- [McLean *et al.* 2003] McLean, K., Fenimore, E., Palmer, D., Barthelmy, S., Gehrels, N., Krimm, H., Markwardt, C., and Parsons, A. (2003), “Setting the Triggering Threshold on Swift, in proceedings of the *Gamma-Ray Bursts: 30 Years of Discovery* conference in Sante Fe NM, Fenimore and Galassi (eds), AIP, astro-ph/0408512
- [Nowak and Figueiredo 2002] Nowak, Robert D., and Figueiredo, Mario A. T., “Unsupervised Progressive Parsing of Poisson Fields Using Minimum Description Length Criteria,” preprint
- [Nowak and Figueiredo 2002] Nowak, Robert D., and Figueiredo, Mario A. T. (2002), “Unsupervised Segmentation of Poisson Data,” preprint
- [Okabe, Boots, Sugihara and Chiu 2000] Okabe, A., Boots, B., Sugihara, K., and Chiu, S. N. (2000), *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, John Wiley and Sons, Ltd., New York, Second Edition
- [Ò Ruanaidh and Fitzgerald 1996] Ò Ruanaidh, J. J. & Fitzgerald, W. J., 1996, *Numerical Bayesian Methods Applied to Signal Processing*, Springer: New York.
- [Papoulis 1965] Papoulis, A, 1965, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill: New York.
- [Prahl 1996] Prahl, J., “A fast unbinned test on event clustering in Poisson processes,” astro-ph/9909399.
- [Scargle 1981] Scargle, J. (1981), Studies in astronomical time series analysis. I: Modeling random processes in the time domain. *Ap. J. Supp.*, **45**, 1-71.
- [Scargle 1998] Scargle, J., 1998, “Studies in Astronomical Time Series Analysis. V. Bayesian Blocks, A New Method to Analyze Structure in Photon Counting Data”, *Astrophysical Journal*, **504**, p. 405-418, Paper V. <http://xxx.lanl.gov/abs/astro-ph/9711233>
- [Scargle 2001a] Scargle, J. D., (2001), Bayesian Blocks: Divide and Conquer, MCMC, and Cell Coalescence Approaches, in *Bayesian Inference and Maximum Entropy Methods in Science and Engineering*, 19th International Workshop, Boise, Idaho, 2-5 August, 1999. Eds. Josh Rychert, Gary Erickson and Ray Smith, AIP Conference Proceedings, Vol. 567, p. 245-256.
- [Scargle 2001b] Scargle, J. D., (2001a), “Bayesian Estimation of Time Series Lags and Structure,” Contribution to **Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering (MAXENT 2001)**, held at Johns Hopkins University, Baltimore, MD USA on August 4-9, 2001.
- [Scargle 2001c] Scargle, J. D., (2001), “Bayesian Blocks in Two or More Dimensions: Image Segmentation and Cluster Analysis,” Contribution to **Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering (MAXENT 2001)**, held at Johns Hopkins University, Baltimore, MD USA on August 4-9, 2001.

[Scargle and Babu 2002] Scargle, J. D., and Babu, G. J. (2002), “Point Processes in Astronomy: Exciting Events in the Universe,” Chapter 20 of Handbook of Statistics: Stochastic Processes: Modeling and Simulation, 2002, Elsevier Science.

[Scott 1992] Scott, D. W. (1992), *Multivariate Density Estimation*, John Wiley & Sons, Inc.: New York

[Schmidt 1999] Schmidt, M. (1999), “Derivation of a Sample of Gamma-Ray Bursts from BATSE DISCLA Data,” in Proc. of the 5th Huntsville Gamma Ray Burst Symposium, Oct. 1999, ed. R.M. Kippen, AIP astro-ph/0001122

[Silverman 1999] Silverman, B. W. (1999), *Density Estimation for Statistics and Data Analysis*, Chapman & Hall/CRC: New York

[BATSE www site] ftp://coss.c.gsfc.nasa.gov/compton/data/batse/ascii_data/64ms/trig00000/cat64

[Weisberg 1980] Weisberg, S. (1980), *Applied Linear Regression*, John Wiley & Sons, Inc.: New York

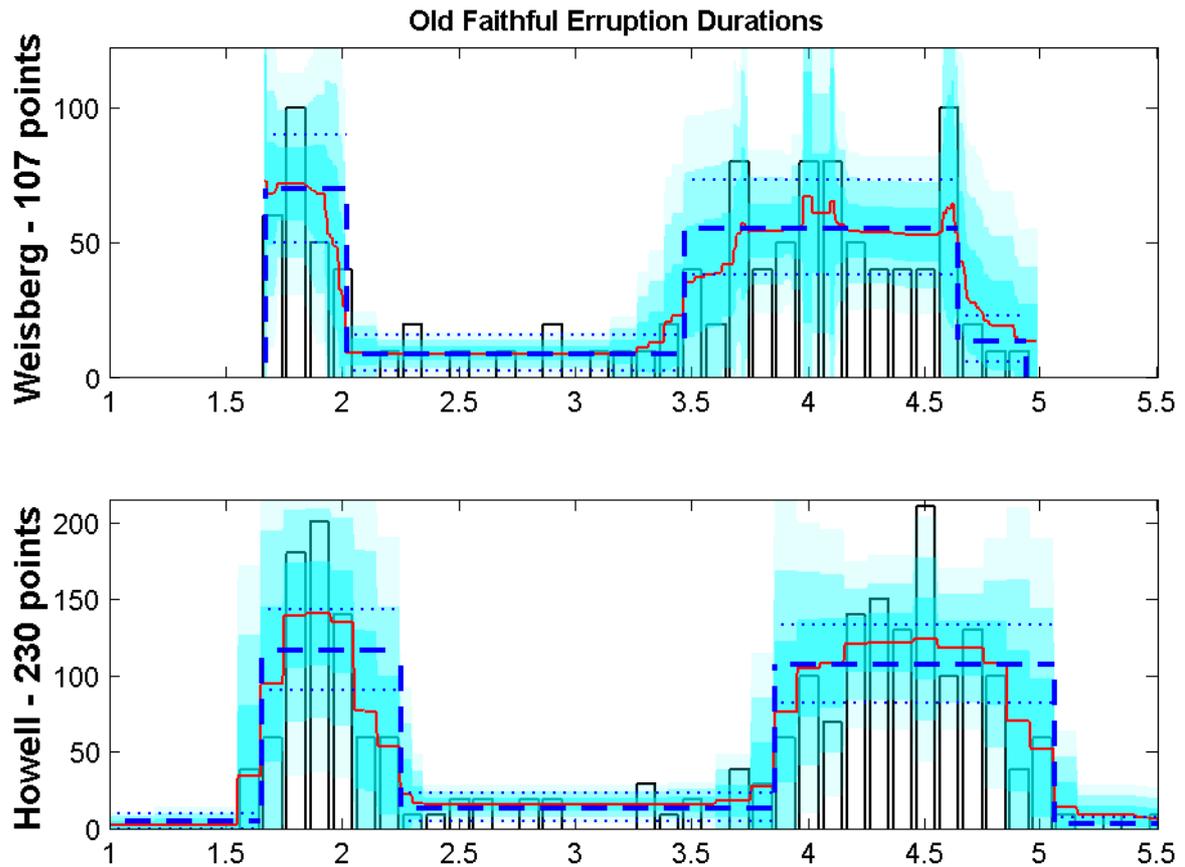


Figure 10: Histogram estimation of Old Faithful eruption durations. The black bars are an equal width, fixed-bin histogram similar to those in [Silverman 1999]. The blue dashed line shows the adaptively binned histogram determined with the methods described in this paper, with the maximum likelihood fitness function. The blue dotted lines depict the rates where the probability in the Poisson distribution drops to 0.05 times that at the peak of the Poisson distribution with parameter equal the its maximum likelihood value. The solid red line is the mean of 1000 bootstrap samples, and the cyan shaded regions delineate 1, 2, and 3 time the bootstrap variance about the mean. Upper panel: 107 values from Table 2.2 of [Silverman 1999]. Lower panel: augmented sample of 230 points [Howell 2005].

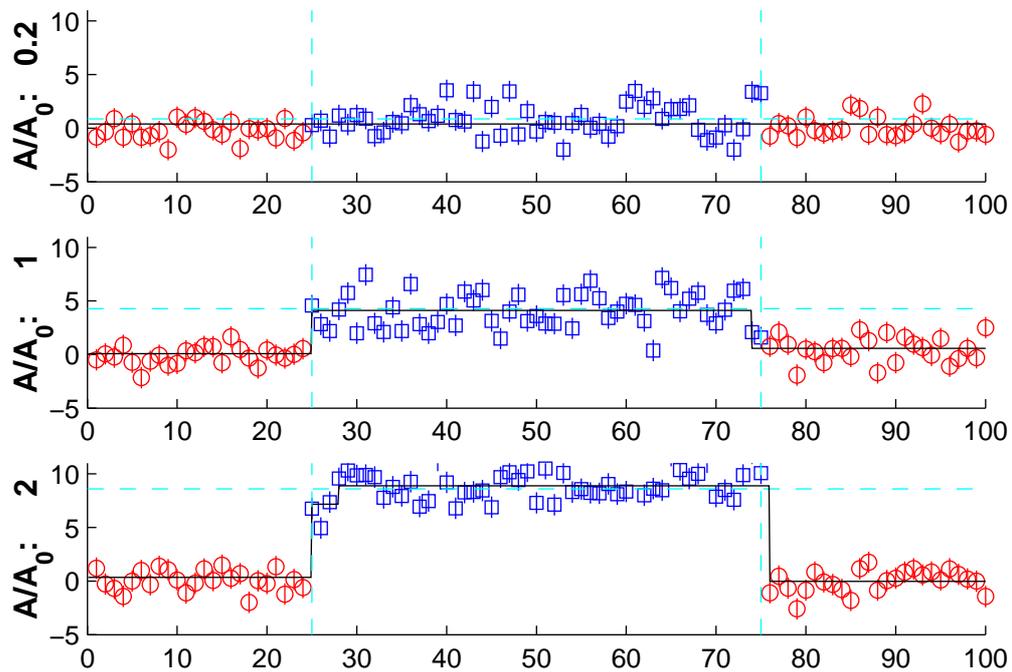


Figure 11: One hundred normally distributed measurements – zero-mean (circles) except for points 25-75 (squares), where the means are 0.2, 1.0 and 2.0 in units of the Arias-Castro *et al.* threshold $\sqrt{2 \log N}$. The dashed lines indicate the true changepoints and block amplitudes, and the solid lines are the Bayesian block representations.

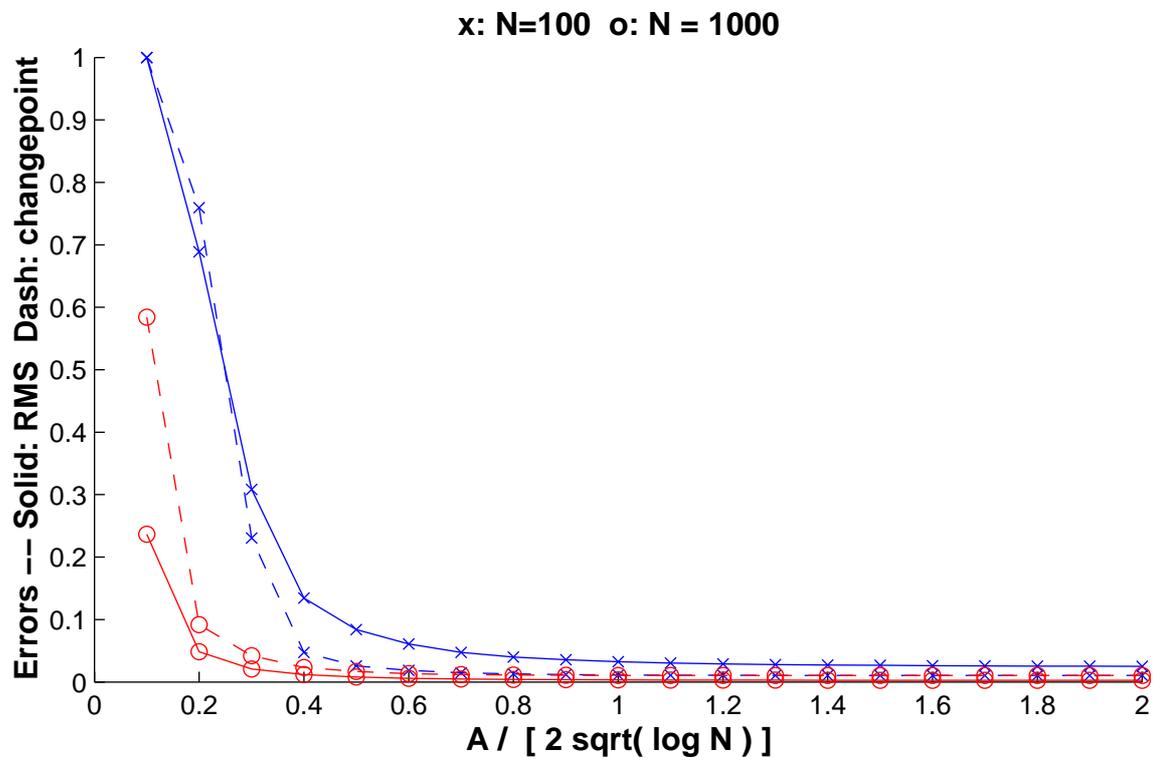


Figure 12: Relative error in finding a single block. **Abscissa:** True block amplitude in units of Arias-Castro *et al.*'s threshold amplitude. **Ordinate:** Error measures described in the text.

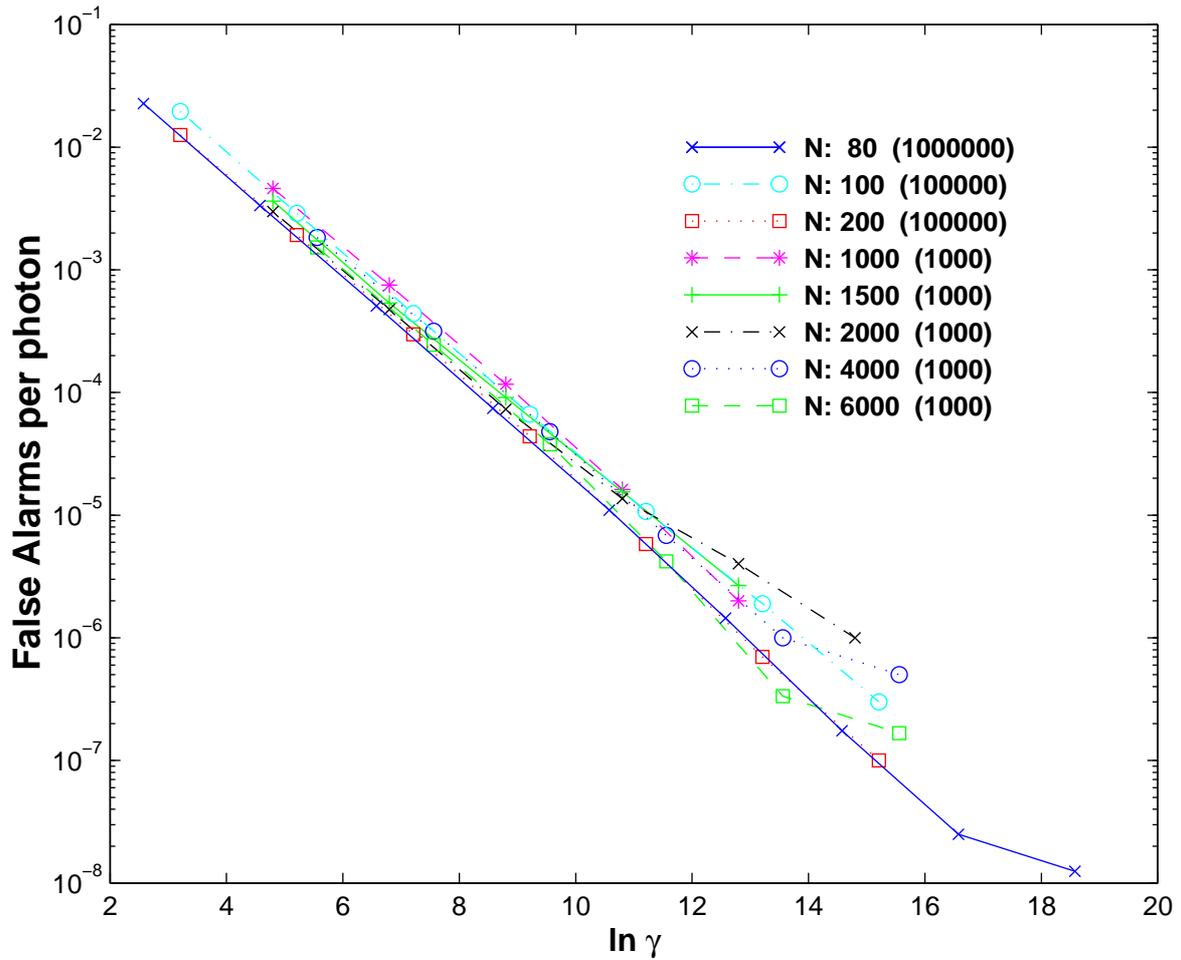


Figure 13: False alarm rates per photon *vs.* $\ln \gamma$. The number of photons per interval, N , and in parentheses the number of averaged simulations, are indicated next to the line-style legend. A linear fit for the false alarm rate is $\sim 0.085 \gamma^{-0.86 \pm 0.08}$ triggers per photon.