Bayesian Blocks: an algorithm for histogram representation

Project for Advanced Statistics for Physics Analysis,
Angelica Foroni, 5/10/2020

01 BAYESIAN BLOCK TECHNIQUE

02 FITNESS FUNCTION

03 HISTOGRAM APPLICATION

04 ALGORITHM

05 R IMPLEMENTATION

06 ENERGY SPECTRUM ANALYSIS

01

Bayesian Block representation

- Non-parametric modelling technique
- Goal: to properly detect and characterize local variability of sequential data to find the optimal segmentation

CHANGE POINT DETECTION ANALYSIS

Final model given by the parameters:

N_{cp} = number of change points → $N_{blocks} = N_{cp} + 1$

$t_{cp}(k)$ = change point starting block *k*

X(k) = signal amplitude of block *k*

01 Bayesian Block representation

- Based on Bayes statistics: the best model (the optimal partition) is chosen maximizing the posterior probability

$M \equiv \mathcal{P}(I) \equiv \{N_{blocks}; n_k, k = 1, 2, 3, \ldots N_{blocks}\}$

$$P(M|D) \propto P(D|M)\, P(M)$$

Likelihood: how well the partition fits the given data

Prior: prior probability distribution for the number of blocks

02

Fitness Function

- Quantity measuring the fitness for the class of piecewise constant models

- Block-additive (assuming independence of the observational errors)

$$F[P(T)] = \sum_{k=1}^{N_{blocks}} f(B_k)$$

- Both for binned and unbinned data the resulting fitness function is:

$$\log L_k(\lambda) = N_k \log(\lambda) - \lambda T_k$$

Maximum for $\lambda = N_k/T_k$

$$F \equiv \log L_{k\,max} + N_k = N_k\,(\log(N_k) - \log(T_k))$$

02

Prior Function

- It is referred to N_{blocks}
- Omitted in the original algorithm [1]

  ↓

  Uniform prior is assumed

- In most settings it is much more likely a priori that $N_{blocks} << N$ than that $N_{blocks} \approx N$, *implementation as a function of N*

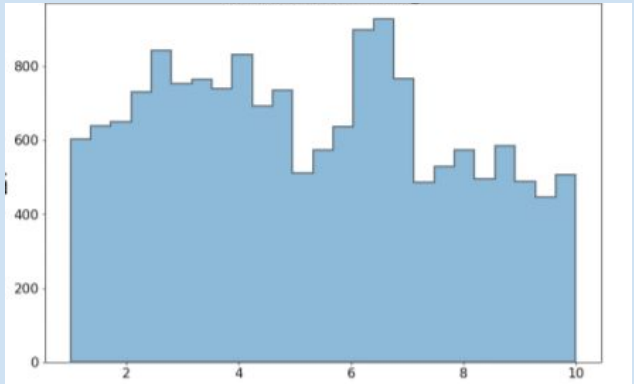
$$ncp_{prior} = 4 - 73.53\, p_0\, N^{(-0.478)}$$

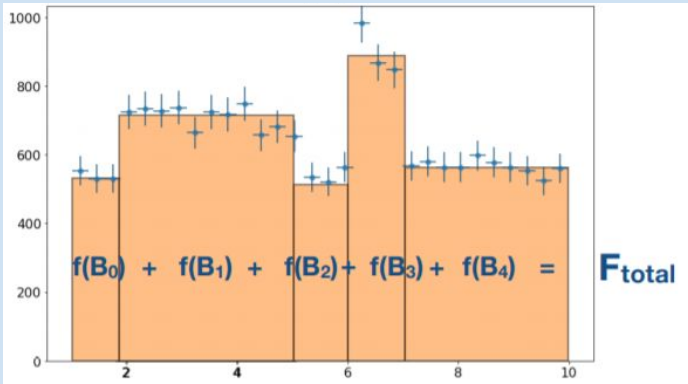
(p_0 = frequency with which the algorithm correctly rejects the presence of a change point with no signal)

03

Histogram Application

- Bayesian Blocks chooses the optimum number of blocks (bins) bin edges
- Blocks treated independently and each edge is statistically significant: new edge → change in underlying pdf
- Input: data, data mode, p_0 = false-positive rate
  Output: Bin Edges

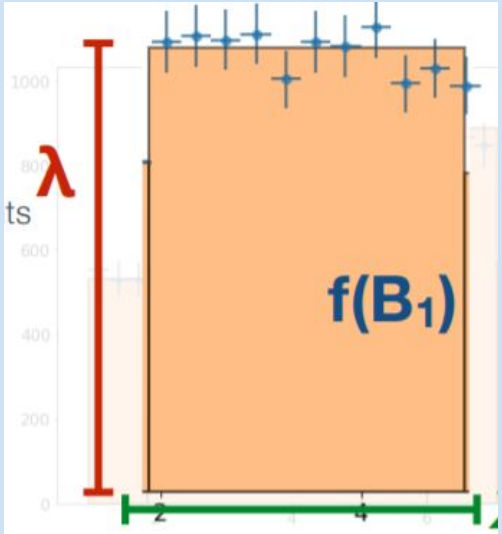




03

Histogram Application

- The fitness, f(Bi), of each bin can be treated as a log-likelihood, assuming the events in each (infinitesimal) bin follow a Poisson distribution



λ = amplitude
x = width of block
n = number of events in a bin

$$\ln(L_B) = n \ln(\lambda) - \lambda x$$

$$f(B_1) \equiv \ln(L_{B\,max}) + n = n\,(\log(n) - \log(x))$$

04

The Algorithm

First version by Jeffrey D. Scargle in 1998 [1] :

1. Change point function: function to find the optimum single change point of an interval corresponding to the smallest odds ratio (Bayes factor):

$$O_{21} = \frac{J(M_2, D)}{J(M_1, D)}$$

J = joint probability for M_i and the data D
M_1 = unsegmented model
M_2 = segmented model

2. Iterative procedure to find the optimal multiple change point function to find the optimum single change point of an interval: accept the segmentation of each subinterval until $O_{21}>1$

R implementation

See jupyter notebook : http://localhost:8888/notebooks/Downloads/ADV_STAT/project/BayesianBlocks.ipynb

04

The Algorithm

Second version developed by Scargle et. al. in 2013 [2]

1. Function to find the optimal number of blocks: beginning with the first data cell and at each step one more cell is added and computed

$\mathcal{P}_{opt}(R + 1)$ = *optimal partition for the first R+1 cells where last block starts with cell r (and by definition ends at R + 1)*

$$A(r) = F(r) + best(r - 1)$$

$$r_{opt} = argmax[A(r)]$$

04

The Algorithm

2. Function to detect the optimal partition with r_{opt} blocks

The needed information is contained in the array last in which we have stored the index ropt at each step

1

c_{p1} = last(N)

2

c_{p2} = last(c_{p1} − 1)

3

c_{p3} = last(c_{p2} − 1)

. . .

Index values giving the locations of the change points, in reverse order

05

R implementation

1. Preprocessing data

```r
log.prior <-  function (p0, N) log(73.53 * p0 * N*p0^(-0.478)) - 4

loglk <- function(N, M) N*(log(N) - log(M))

Bayesian.blocks <- function (x, p0=0.003, model="Gamma",
                             params = c(gamma=0.01, p=0.01),
                             input.type = "TTE") {

    stopifnot(model %in% c("Normal", "Gamma", "Exponential", "Poisson"))
    stopifnot(input.type %in% c("TTE", "TTS", "binned", "histogram"))

    if (input.type == "TTE" | input.type == "TTS" | input.type == "TTS"){

        x_unique <- sort(x)
        N <- length(x_unique)
        nn.vec <- rep(1, N)
        N.ev  <- sum(x_weight)
        dx <- diff(x_unique)
        edges <- c(x_unique[1],
                   0.5*(x_unique[2:N] + x_unique[1:N-1]),
                   x_unique[N])
        block.length <- edges[N + 1] - edges
    }

    else if (input.type == "histogram"){

        N <- length(x)
        x_unique <- 1 : N
        nn.vec <- x
        N.ev  <- sum(nn.vec)
        edges <- c(x_unique[1],
                   0.5*(x_unique[2:N] + x_unique[1:N-1]),
                   x_unique[N])
        block.length <- edges[N + 1] - edges
    }
```

False positive rate, probability that the change is not real default: 0.003 i.e. 3 sigma

2. Detect optimal Nblocks

```r
best <- rep(0, N)
last <- rep(0, N)

for (k in 1:N) {

    width <- block.length[1 : k] - block.length[k + 1]
    width[which(width <= 0)] <- Inf
    count_vec <- rev(cumsum(rev(nn.vec[1 : k])))

    ncp.prior <- log.prior(p0, N)
    F <- loglk(count_vec, width) + ncp.prior
    A <- F[2 : length(F)] + best[1 : k - 1]
    A <- c(F[1], A)

    maxA <- max(A)[1]
    i.max <- which(A == maxA)[1]

    best[k] <- maxA
    last[k] <- i.max
}
```

Store the value and index of the point that maximises the fit function

Start with first data cell; add one cell at each iteration

Compute the width and count of the final bin for all possible locations of the Kth changepoint

Evaluate F:= fitness function + prior for these possibilities and find the maximum value corresponding to the K-th changepoint

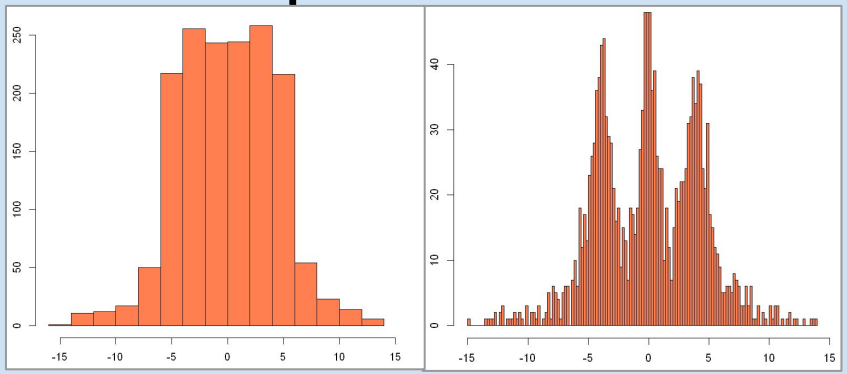
3. Detect change points

```r
    change.points <-  rep(0,N)
    i.cp <- N
    ind <- N

    while (TRUE) {
        i.cp <- i.cp - 1
        change.points[i.cp] <- ind

        if (ind <= 1){
            break
        }

        ind <- last[ind - 1]
    }

    n.cp <- length(change.points)
    change.points <- change.points[i.cp:n.cp]

    return (c(edges[c(change.points)]))
}
```

Recover changepoints by iteratively peeling off the last block

05 Application example

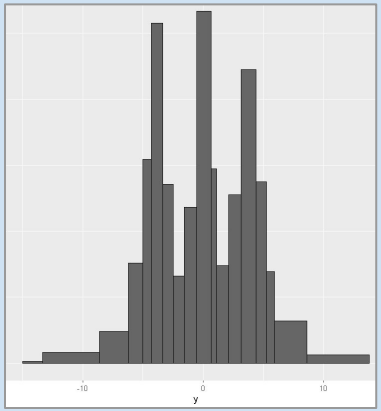
```r
y <- Filter( function(x) (x > -15) && (x < 15),
             c(rcauchy(2000, location=-2, scale=0.8),
               rcauchy(1000, location=2, scale=0.8),
               rcauchy(2000, location=5, scale=1.5)))

hist(y, main="", col= "coral")

hist(y, breaks=200, main="", col= "coral")
```



Automatic and manual setting of number of bins

"subjective"

```r
bb <- Bayesian.blocks(y, input.type="TTE")
library("ggplot2")
dfy <- data.frame(y)
  ggplot(dfy, aes(y)) +
          geom_histogram(aes(y=..density..),
                 color="black", fill="grey40", breaks=bb)
```



Histogram with Bayesian Blocks

"objective"

06

Energy Spectrum Analysis

- Energy spectrum collected by a Germanium Detector of a source given by a combination of Am241, Co60 and Cs137

- 1-D dataset

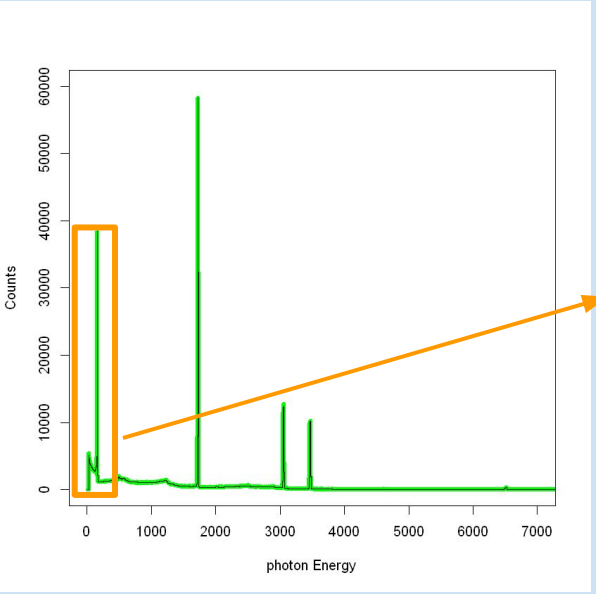
1. Direct modelling of data without preprocessing them into frequencies

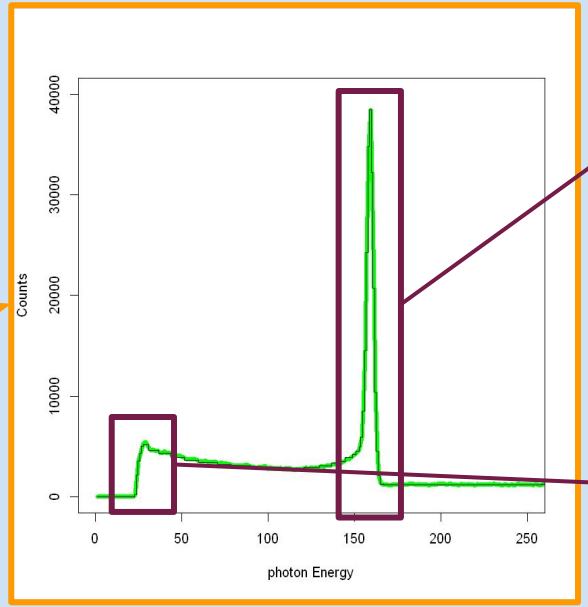
Output = binned data

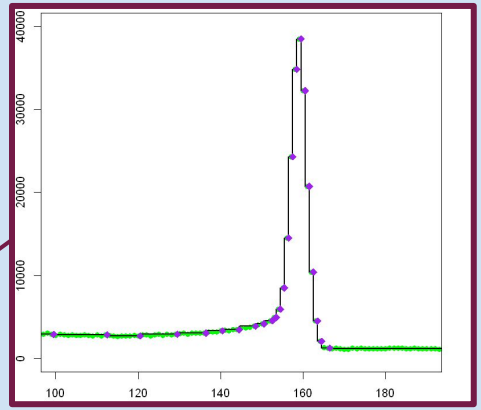
2. Sorted and translated into an histogram with infinitesimal bins (as a TTE dataset)

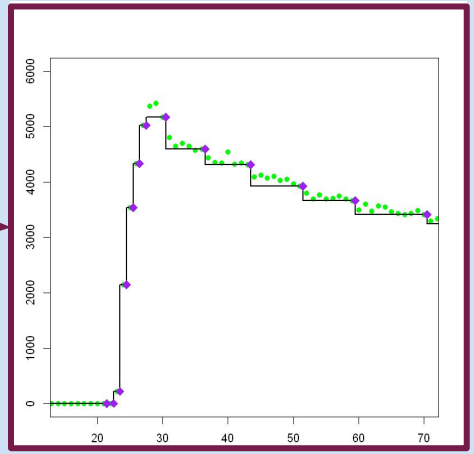
Output = binned histogram of data

06 Energy Spectrum Analysis

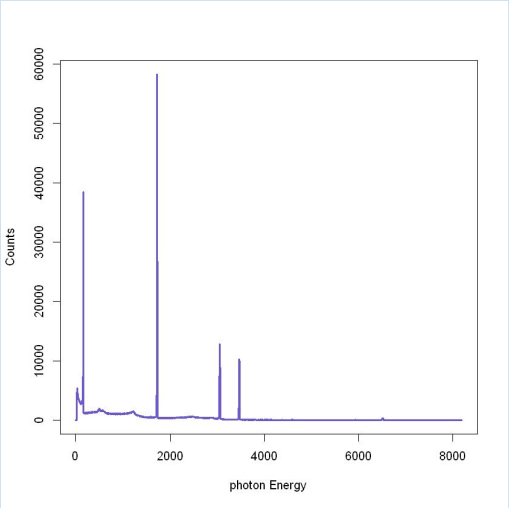


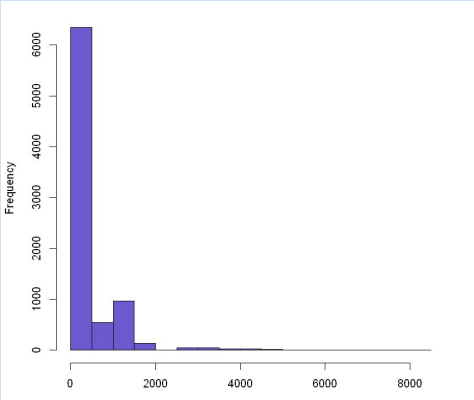


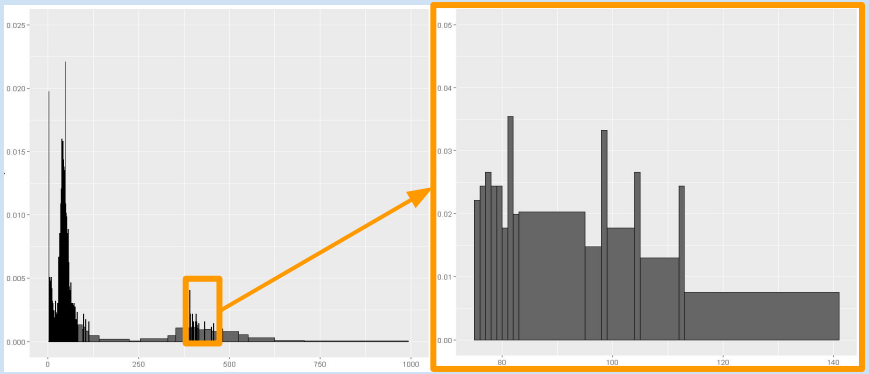




06 Energy Spectrum Analysis







REFERENCES

[1] Scargle, J. 1998, *ApJ*, 504, 405
https://iopscience.iop.org/0004-637X/504/1/405

[2] B. Pollack et al., arXiv:1708.0081
https://arxiv.org/pdf/1708.00810.pdf

[3] Scargle et. al. 2013, ApJ. 764, 2

https://iopscience.iop.org/article/10.1088/0004-637X/764/2/167#references