Histogram Binning with Bayesian Blocks

Brian Pollack, Northwestern University

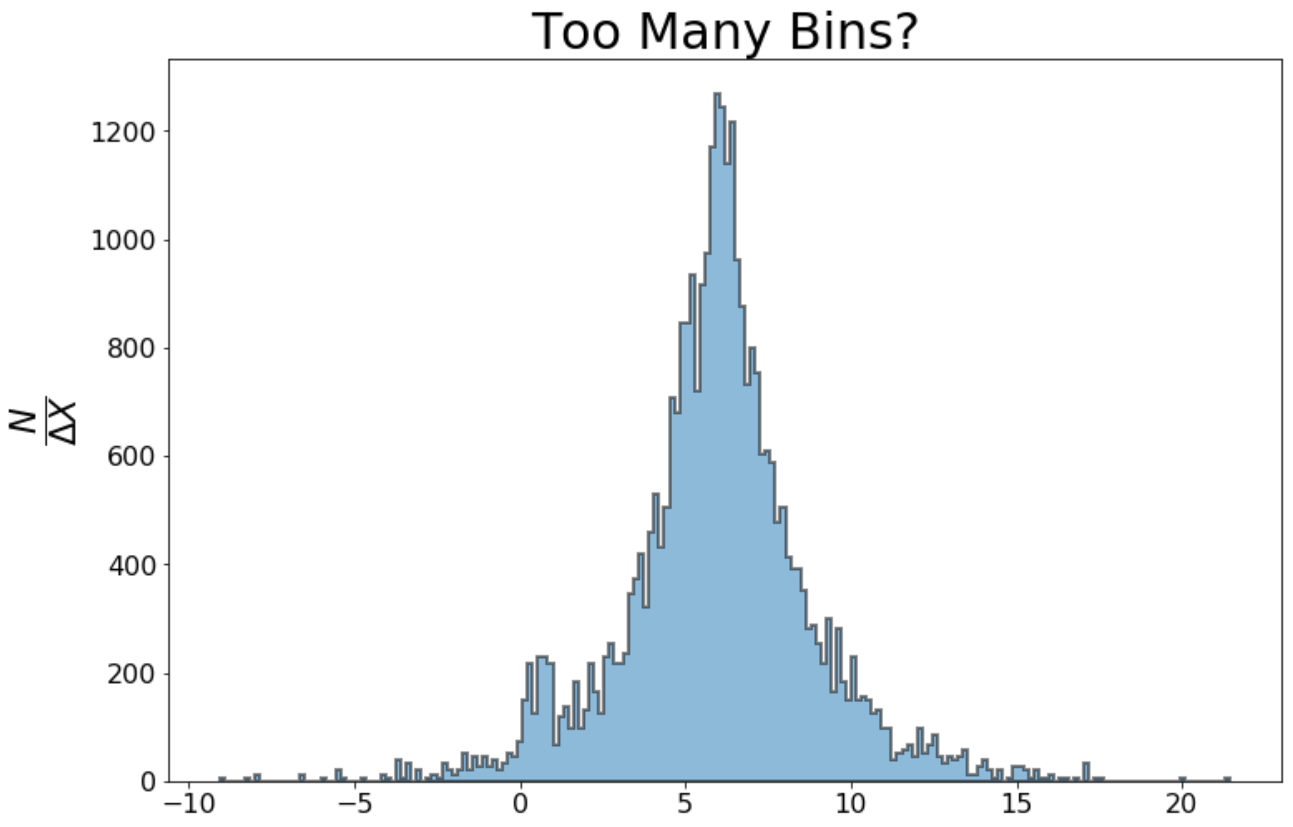
8/3/17

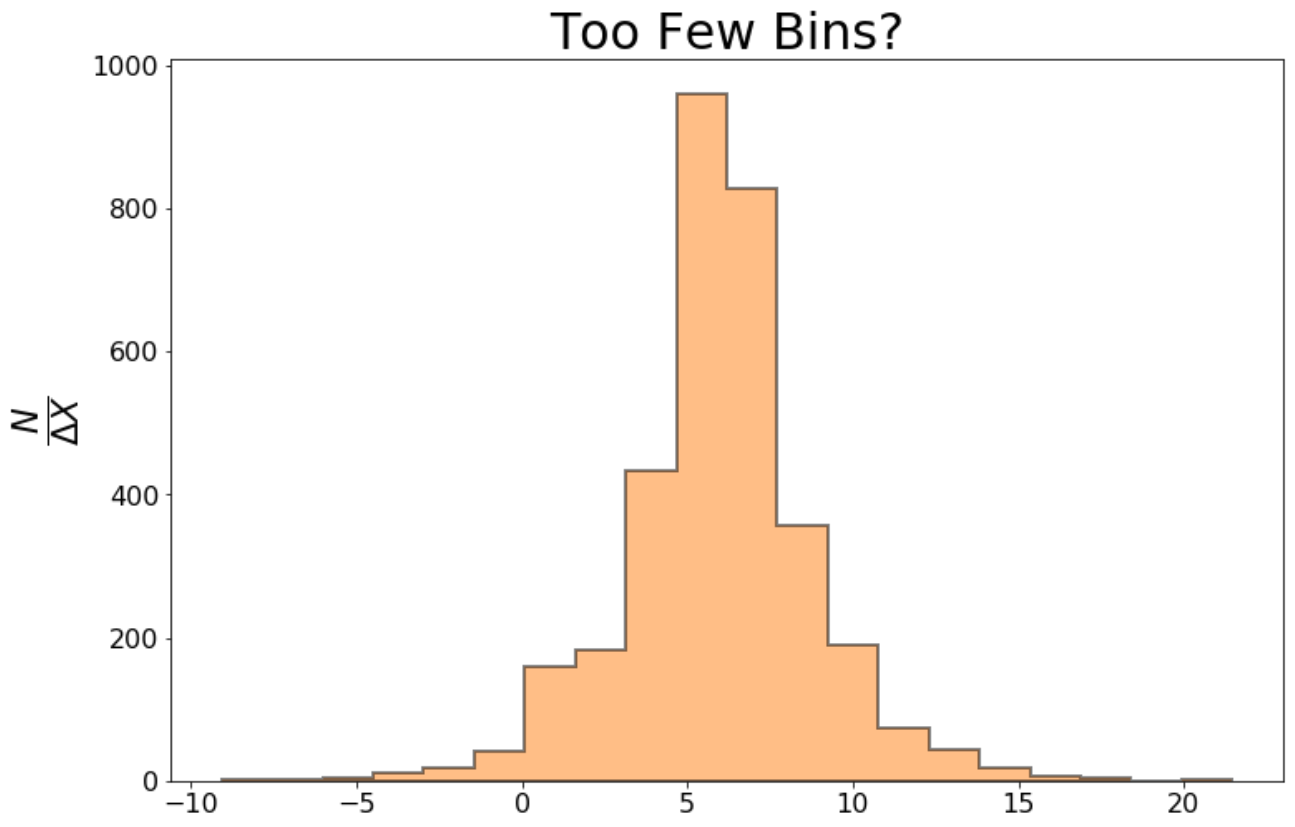
Coauthors: Sapta Bhattacharya, Michael Schmitt

arXiv: [1708.00810](https://arxiv.org/abs/1708.00810)

How Do We Bin?

- ★ **Histogram binning is usually *arbitrary.***
  - Number of bins → Whatever seems to look reasonable.
  - Too many bins → Statistical fluctuations obscure structure.
  - Too few bins → Small structures are swallowed by background.
- ★ **Bayesian Blocks (BB) chooses 'best' number of blocks (bins), and 'best' choice for bin edges.**



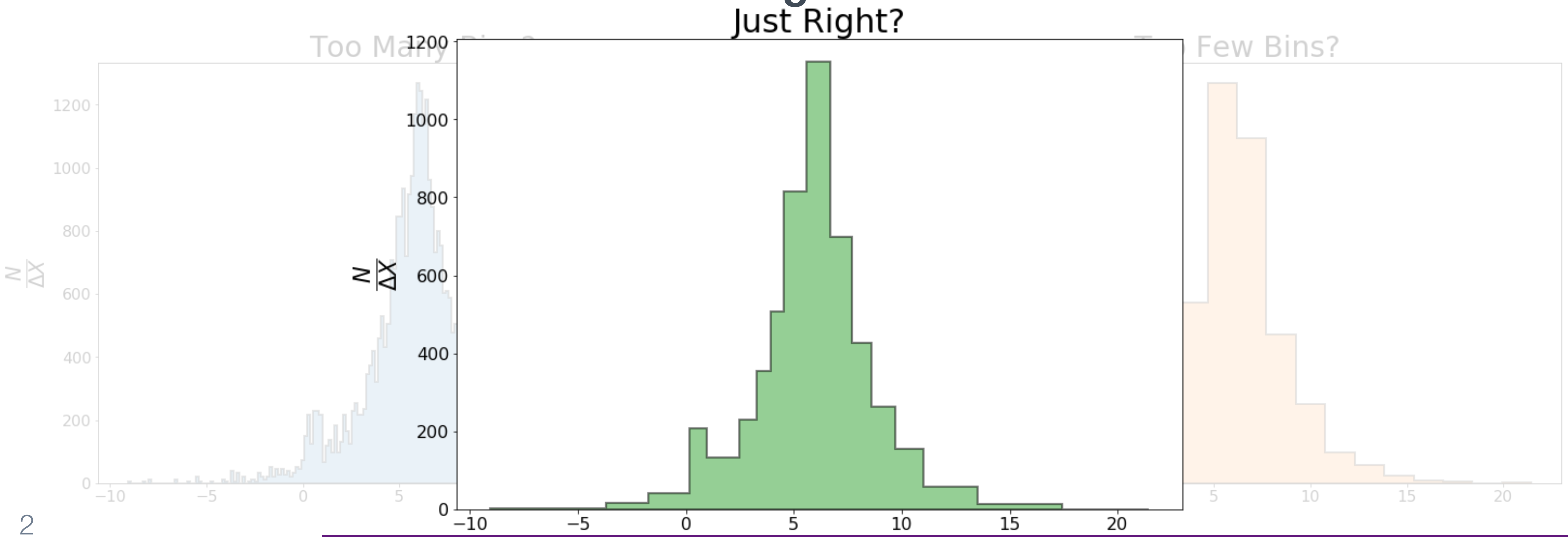


How Do We Bin?

★ **Histogram binning is usually *arbitrary.***

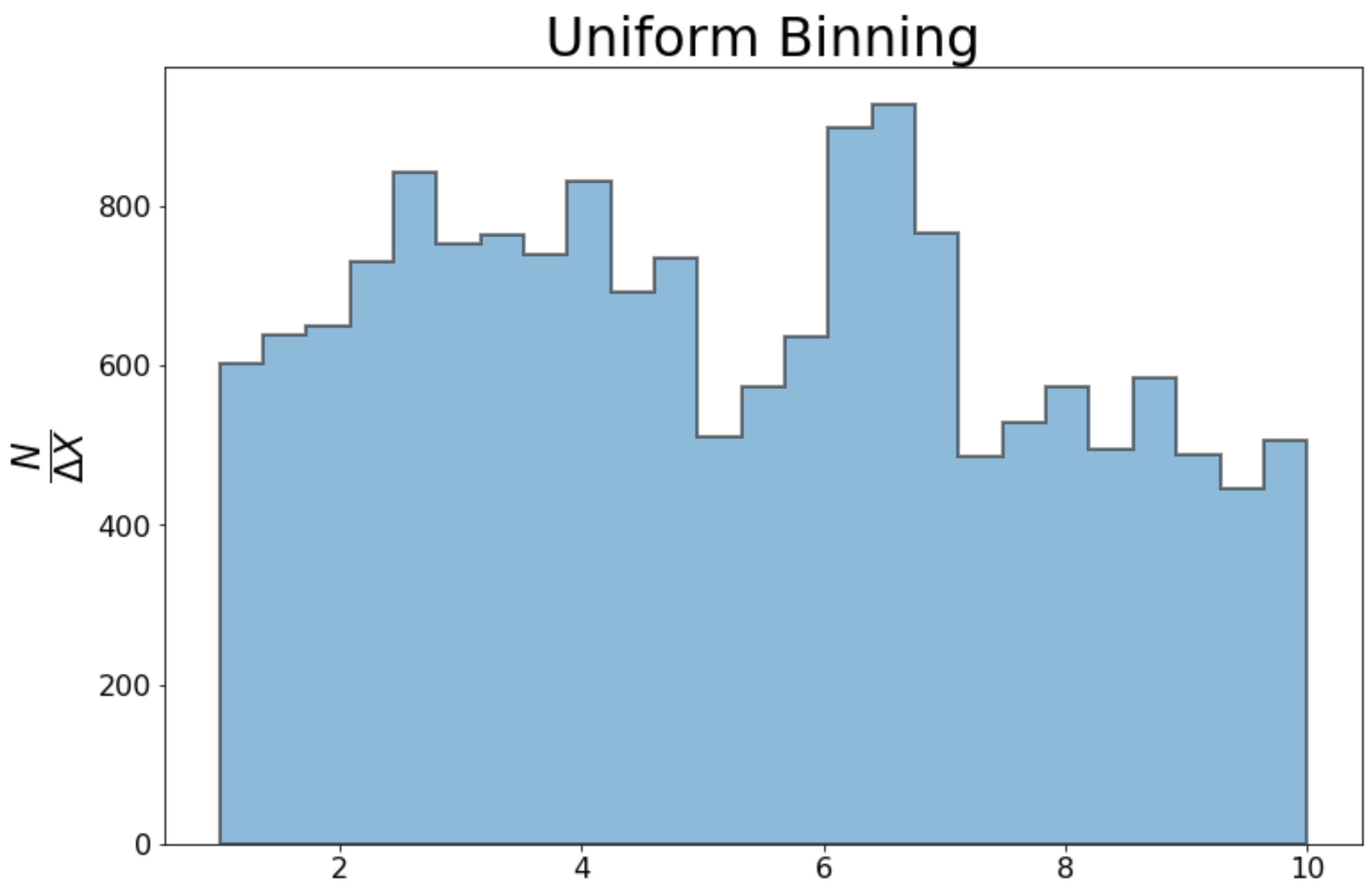
- Number of bins → Whatever seems to look reasonable.
- Too many bins → Statistical fluctuations obscure structure.
- Too few bins → Small structures are swallowed by background.

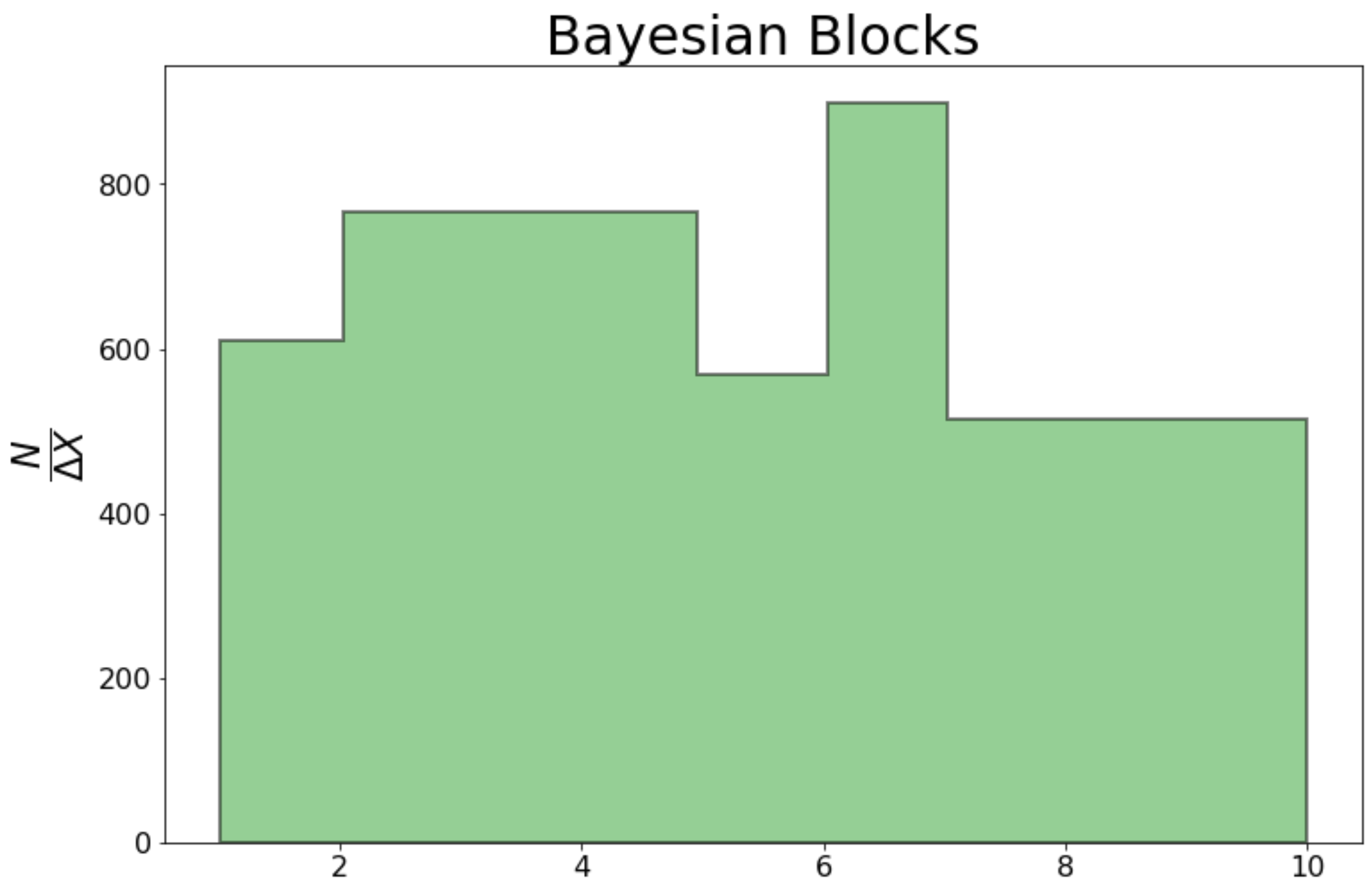
★ **Bayesian Blocks (BB) chooses 'best' number of blocks (bins), and 'best' choice for bin edges.**



Bayesian Blocks

- ★ **Input:**
  - Data
  - False-positive rate (tuning parameter)
- ★ **Output:**
  - Bin Edges
- ★ **Each edge is statistically significant**
  - New edge → change in underlying pdf

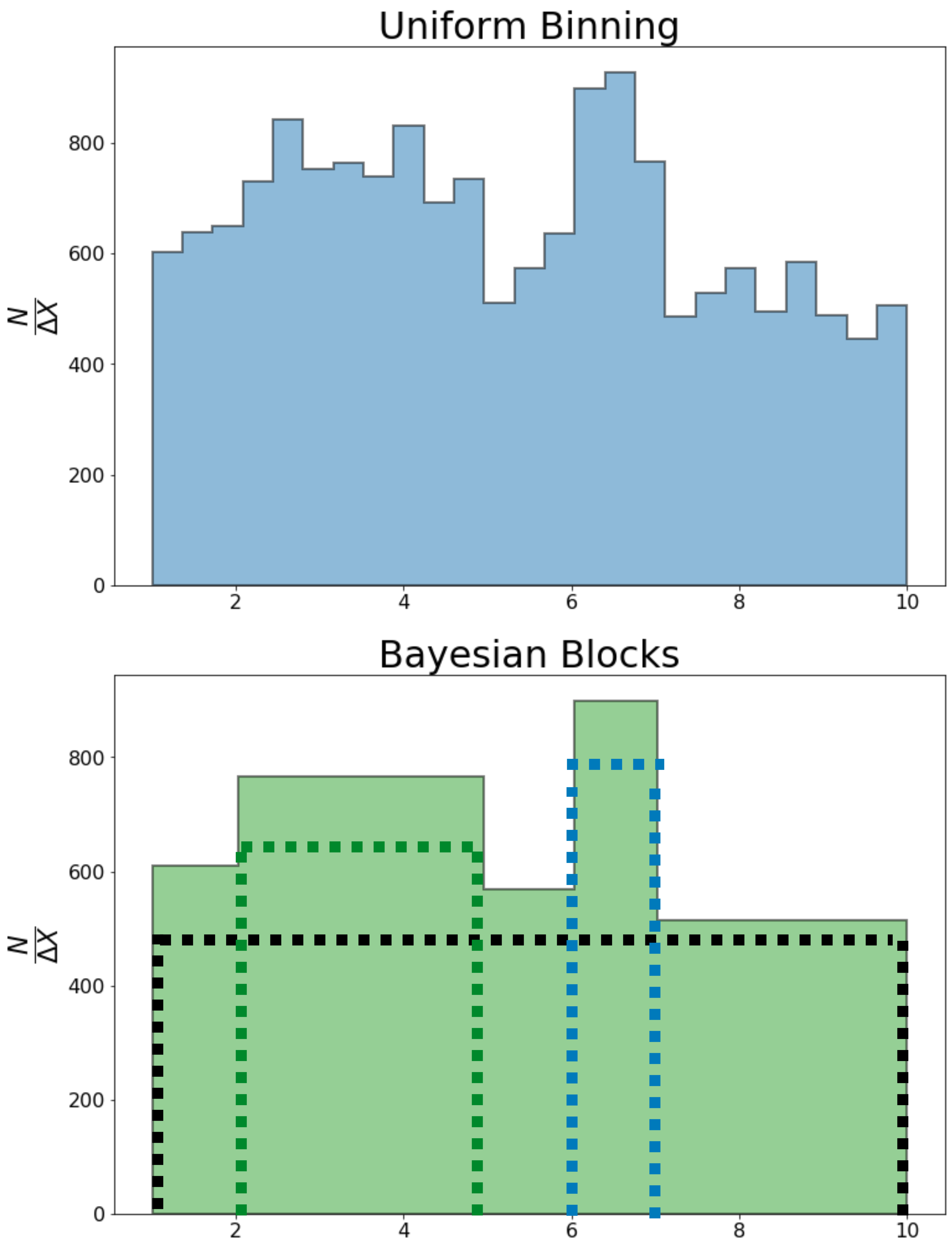




Underlying pdfs: 3 Uniform distributions

Bayesian Blocks

- ★ **Input:**
  - Data
  - False-positive rate (tuning parameter)
- ★ **Output:**
  - Bin Edges
- ★ **Each edge is statistically significant**
  - New edge → change in underlying pdf



Underlying pdfs: 3 Uniform distributions

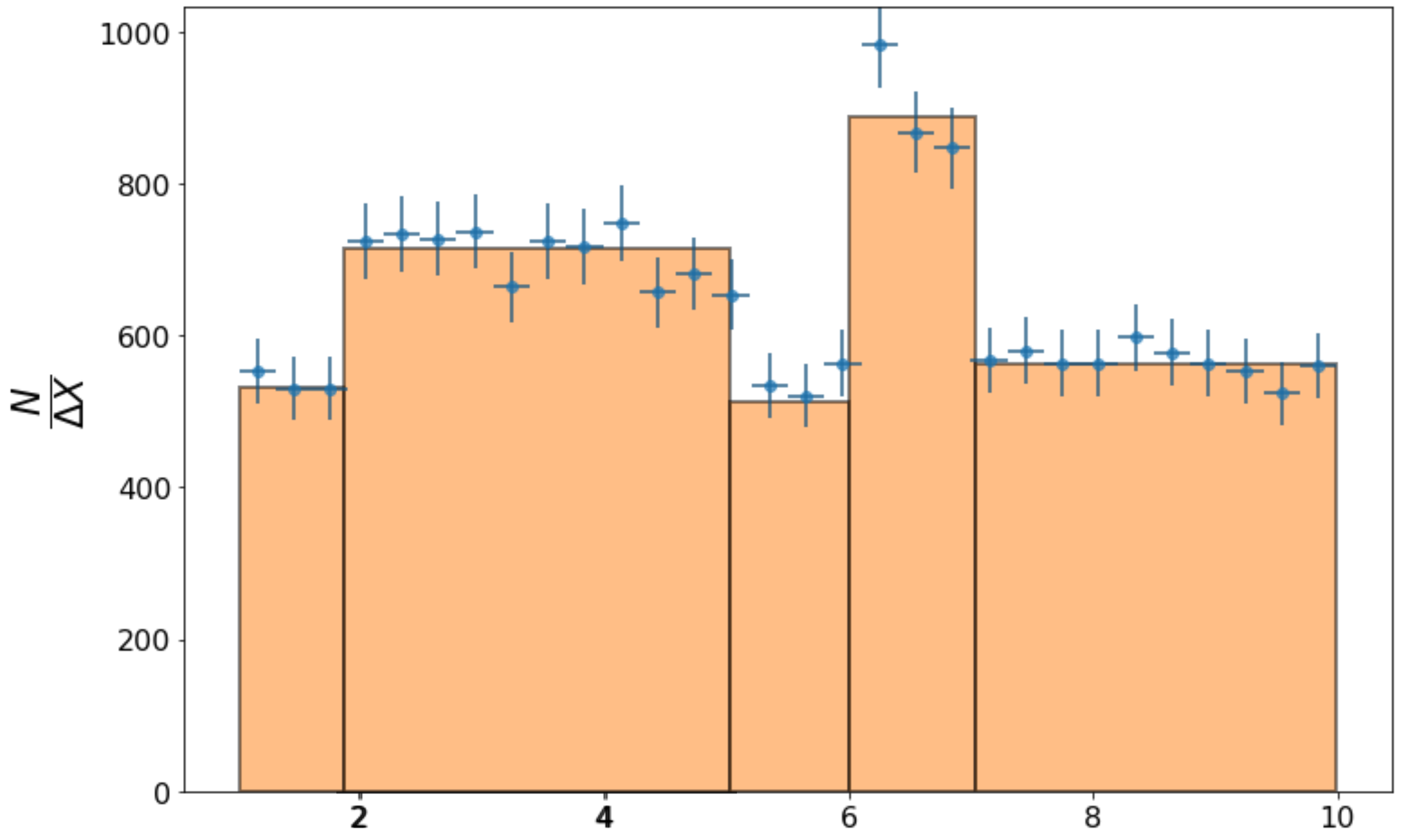
Bayesian Blocks

- ★ **Developed by J. D. Scargle et. al.*, for use with time-series data in astronomy.**
- ★ **Goal: characterize statistically significant variations in data.**
  - Accomplish via <u>optimal segmentation</u> using non-parametric modeling.
    - ✦ Each segment treated as histogram bin (bins have variable widths).
    - ✦ Each segment associated with uniform distribution.
    - ✦ Combination of data and uniform distributions → calculation of **<u>fitness function</u>**.
- ★ **Finding maximal fitness function requires clever programming, not feasible to use naive (brute force) methods.**
  - For N data points, 2^N possible binnings → untenable for large N

*STUDIES IN ASTRONOMICAL TIME SERIES ANALYSIS. VI. BAYESIAN BLOCK REPRESENTATIONS

The Fitness Function

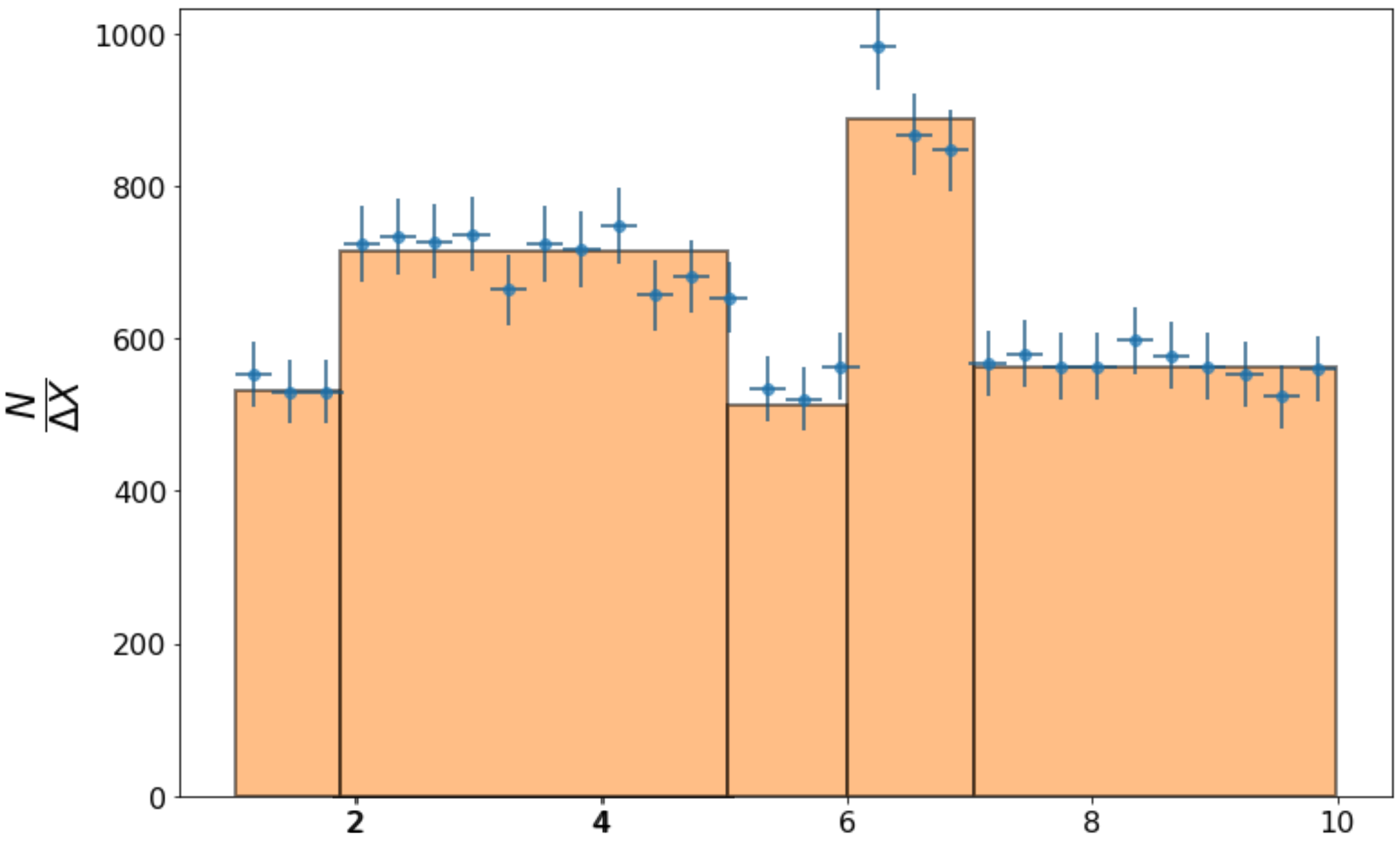
- The **Fitness Function** is a quantity that is maximized when the optimal segmentation of a dataset is achieved.



The Fitness Function

- ★ **The Fitness Function is a quantity that is maximized when the optimal segmentation of a dataset is achieved.**
- ★ **For *K* bins, the total fitness, *F*$_{total}$, can be defined as the sum of the fitnesses of each bin, *f(B*$_i$*)*:**

$$F_{total} = \sum_{i=0}^{K} f(B_i)$$

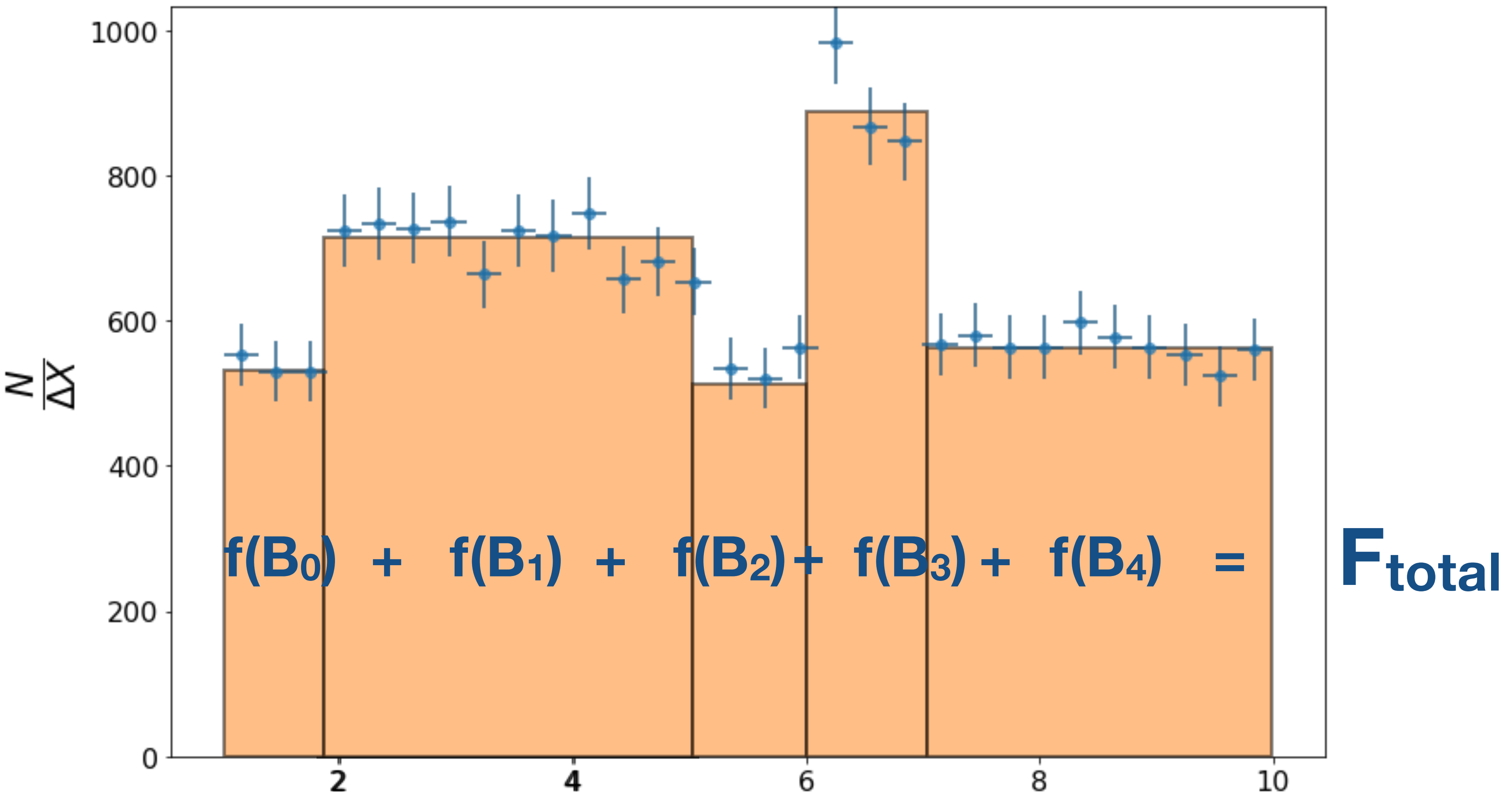


The Fitness Function

★ **The <u>Fitness Function</u> is a quantity that is maximized when the optimal segmentation of a dataset is achieved.**

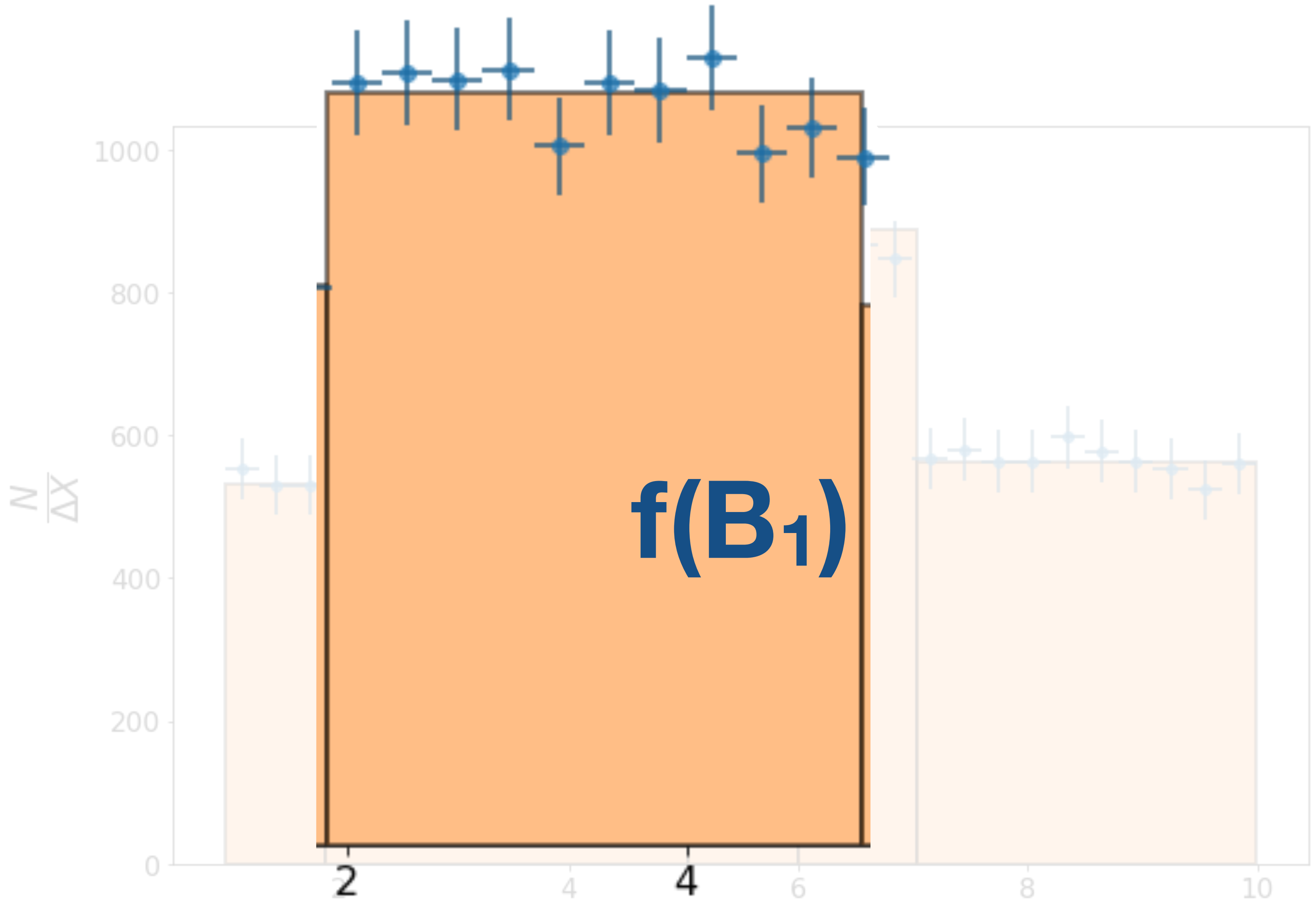
★ **For *K* bins, the total fitness, *F_{total}*, can be defined as the sum of the fitnesses of each bin, *$f(B_i)$*:**

$$F_{total} = \sum_{i=0}^{K} f(B_i)$$



The Fitness Function

The fitness, *f(B$_i$)*, of <u>each bin</u> can be treated as a log-likelihood, assuming the events in each bin follow a Poisson distribution.



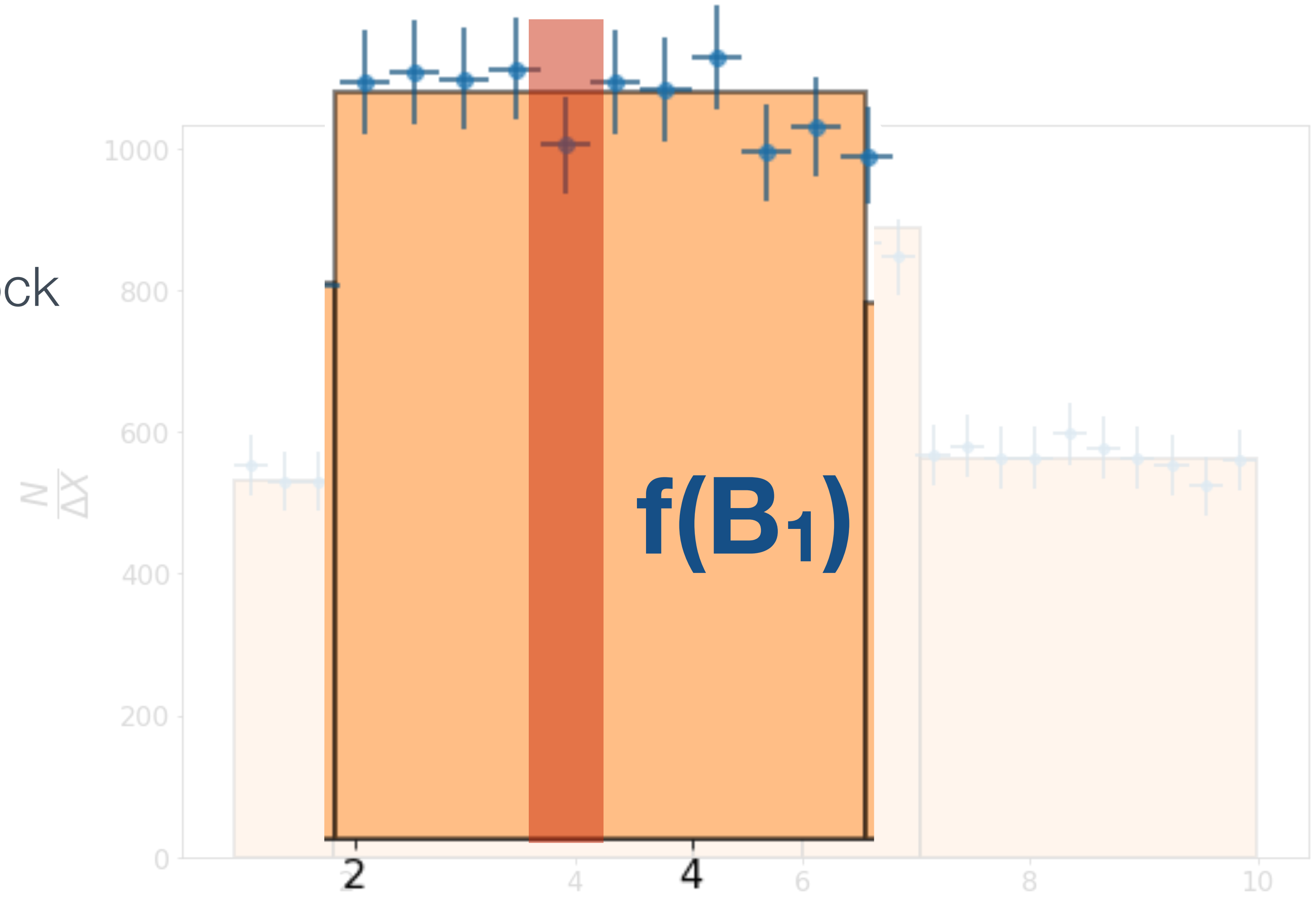
The Fitness Function

The fitness, *f(B$_i$)*, of <u>each bin</u> can be treated as a log-likelihood, assuming the events in each bin follow a Poisson distribution.

$P_{dx} = \lambda(x)dx \times e^{-\lambda(x)dx}$ → probability for an infinitesimal bin.

λ: amplitude

x: width of block



The Fitness Function

The fitness, *f(B$_i$)*, of <u>each bin</u> can be treated as a log-likelihood, assuming the events in each bin follow a Poisson distribution.

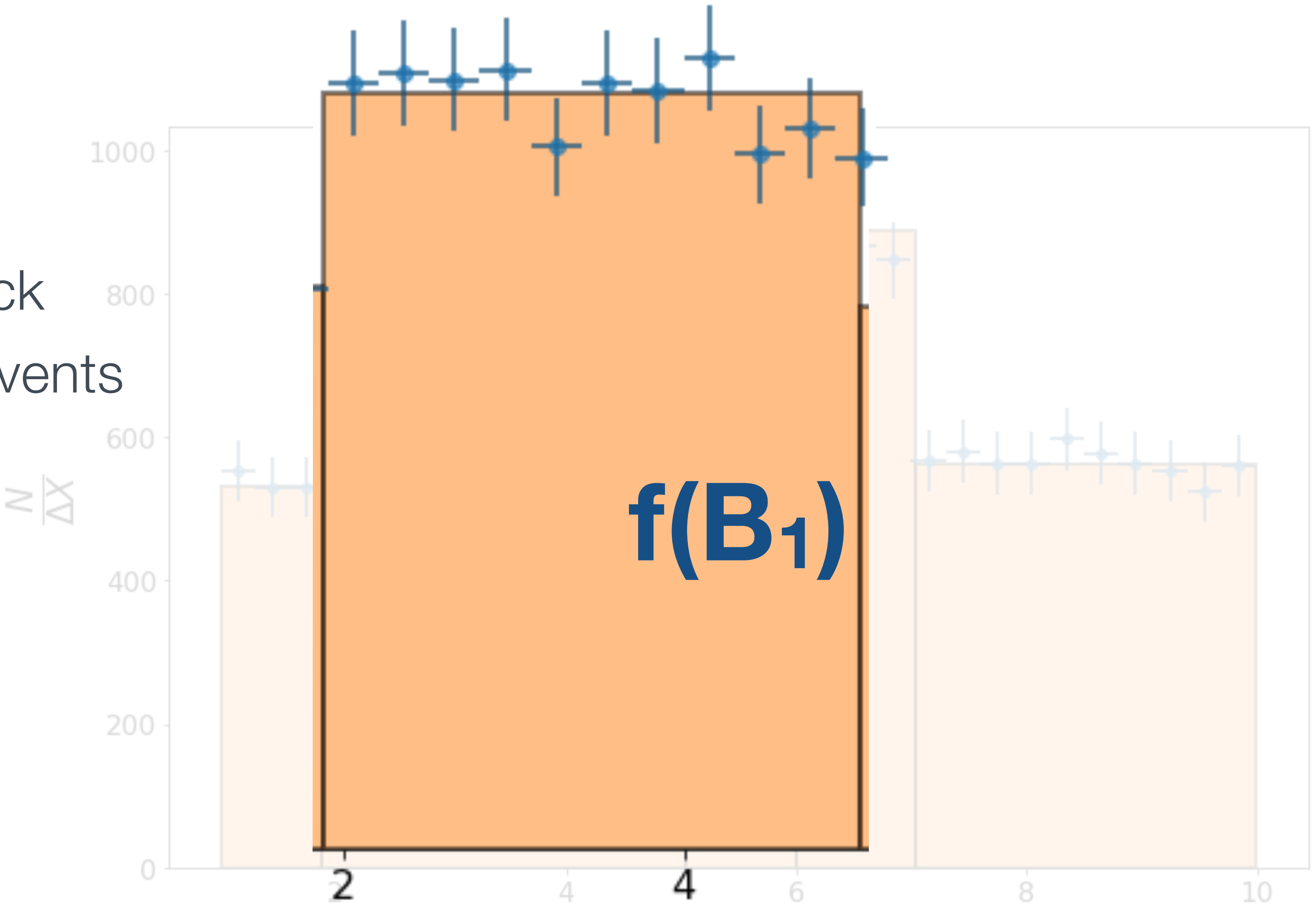
$$P_{dx} = \lambda(x)dx \times e^{-\lambda(x)dx}$$ → probability for an infinitesimal bin.

$$\ln L_B = \sum^{n} \ln \lambda(x) + \sum^{n} \ln dx - \int \lambda(x)dx$$ → log-likelihood for an entire bin.

λ: amplitude

x: width of block

n: number of events in a bin



The Fitness Function

The fitness, *f(B$_i$)*, of <u>each bin</u> can be treated as a log-likelihood, assuming the events in each bin follow a Poisson distribution.

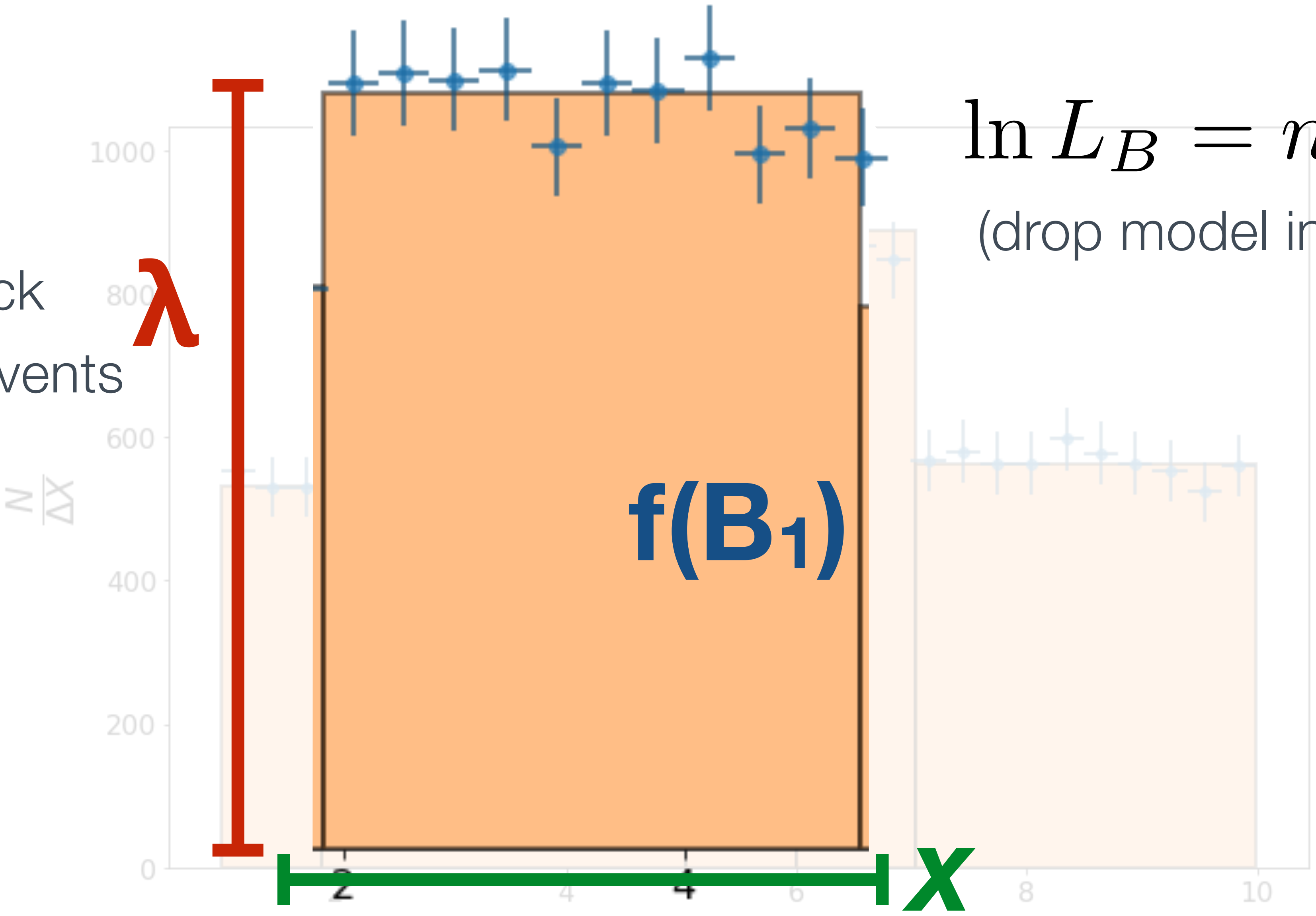
$P_{dx} = \lambda(x)dx \times e^{-\lambda(x)dx}$ → probability for an infinitesimal bin.

$\ln L_B = \sum^{n} \ln\lambda(x) + \sum^{n} \ln dx - \int \lambda(x)dx$ → log-likelihood for an entire bin.

λ: amplitude

x: width of block

n: number of events in a bin



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

(drop model independent terms)

The Fitness Function

The fitness, *f(B$_i$)*, of <u>each bin</u> can be treated as a log-likelihood, assuming the events in each bin follow a Poisson distribution.

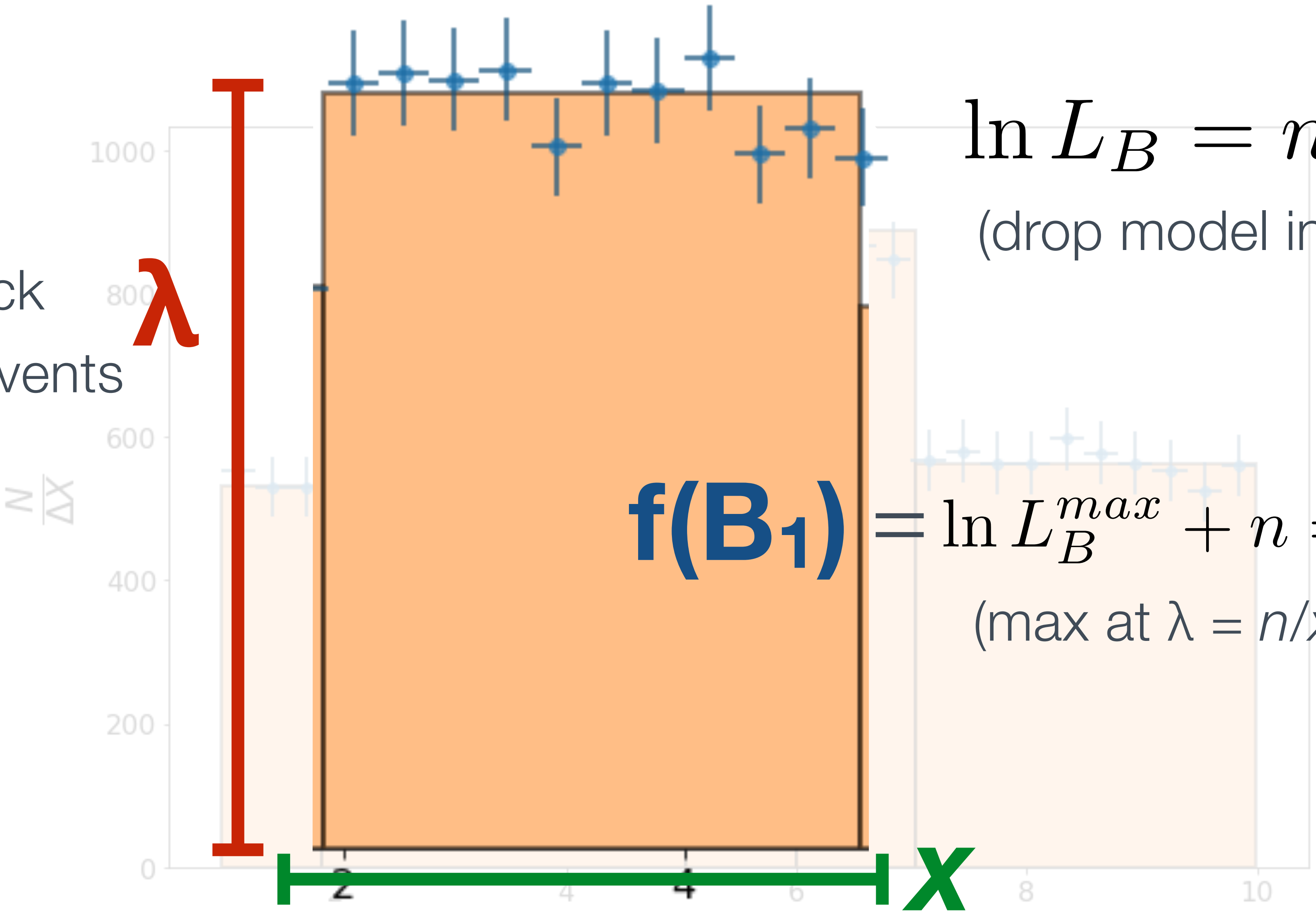
$P_{dx} = \lambda(x)dx \times e^{-\lambda(x)dx}$ → probability for an infinitesimal bin.

$\ln L_B = \sum^n \ln \lambda(x) + \sum^n \ln dx - \int \lambda(x)dx$ → log-likelihood for an entire bin.

λ: amplitude

x: width of block

n: number of events in a bin



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

(drop model independent terms)

$$f(B_1) = \ln L_B^{max} + n = \boxed{n(\ln n - \ln x)}$$

(max at λ = *n*/*x*)

Penalty Term

- ★ **Given the previous definitions, the total fitness, F_{total}, will be maximal when the number of bins, K, is equal to the number of data points.**
  - This is not desirable!
- ★ **A penalty term, $g(K)$, is introduced such that:**

$$F_{total} = \sum_{i=0}^{K} f(B_i) \rightarrow \sum_{i=0}^{K} f(B_i) - g(K)$$

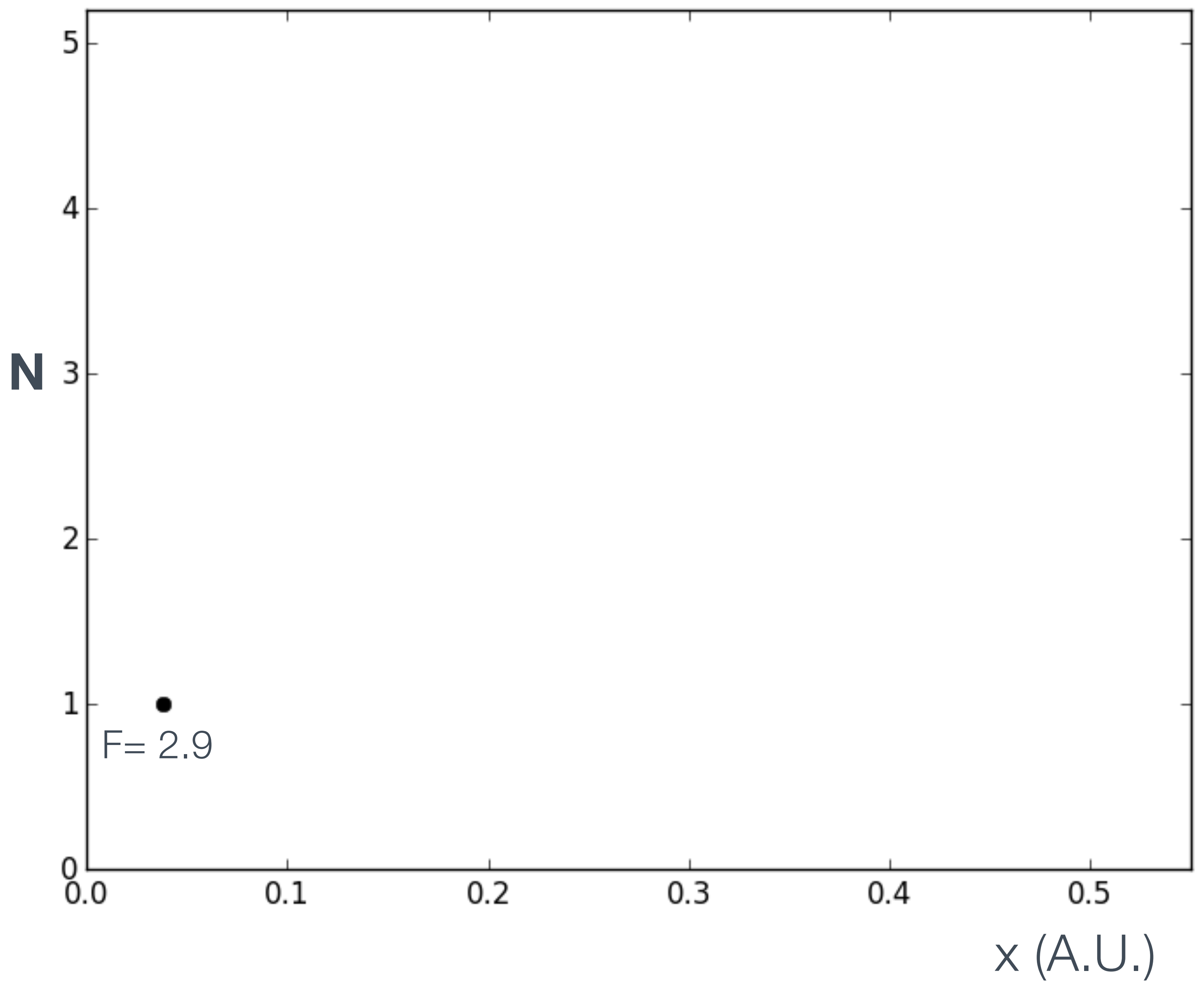
- ★ **Term reduces F_{total} as K increases.**
- ★ **This term is user defined, and should be tuned on signal-free data.**

Algorithm Overview

- **For N data points, there are 2^N total bin combinations.**
- **BB algo finds optimal binning in $O(N^2)$.**
  - Start: Ordered, unbinned data.
  - Iterate over data:
    - Calculate fitness for all new potential bins *("New bins" = set of all bins that include newest data point).*
    - Determine current maximum total fitness *(Use cached results of previous iterations with new best bin).*
  - Finish iteration, return bin edges associated with max fitness.

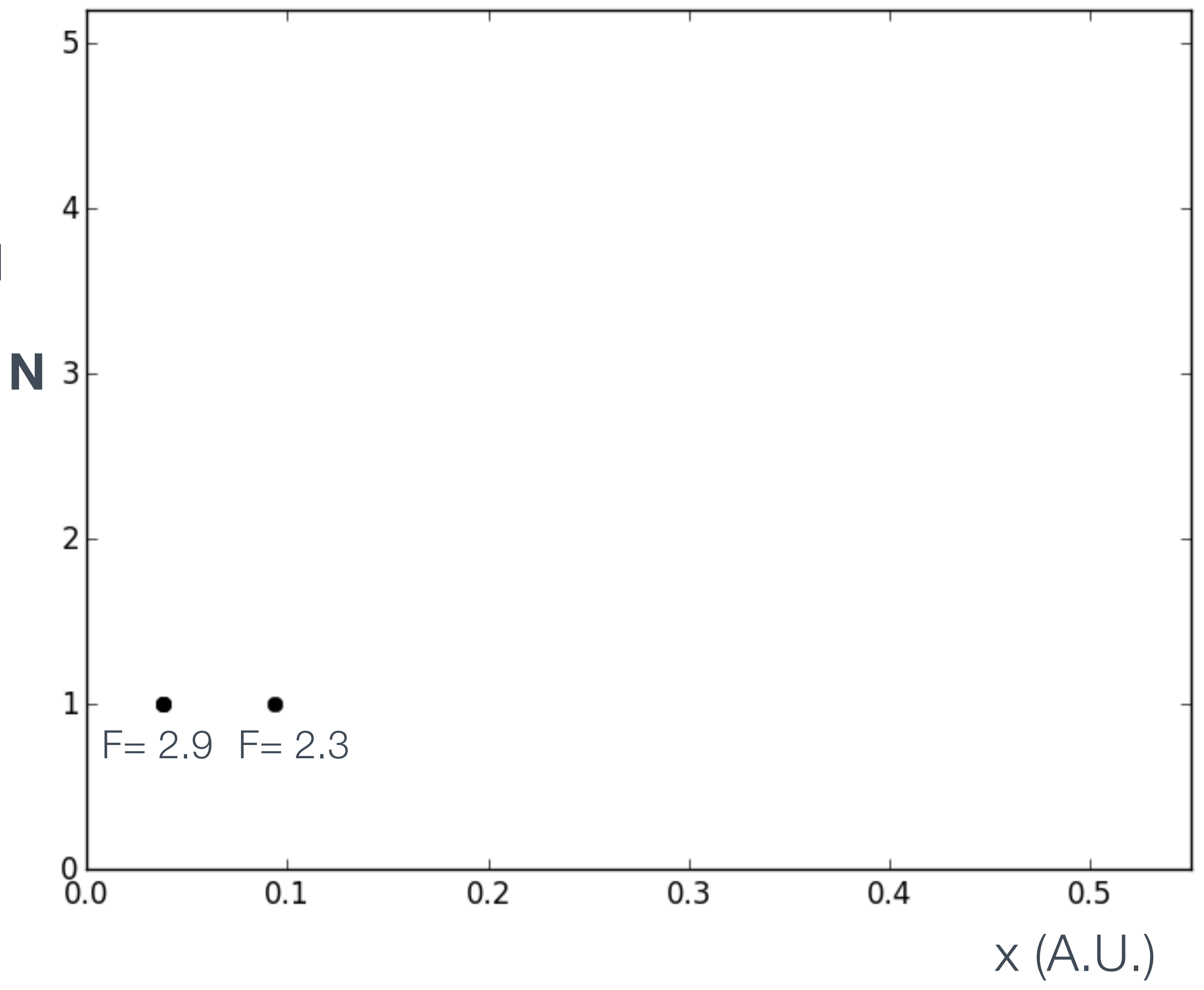
Algorithm Example

- First data point added.
- Fitness Function (F) is trivial, only one point considered.



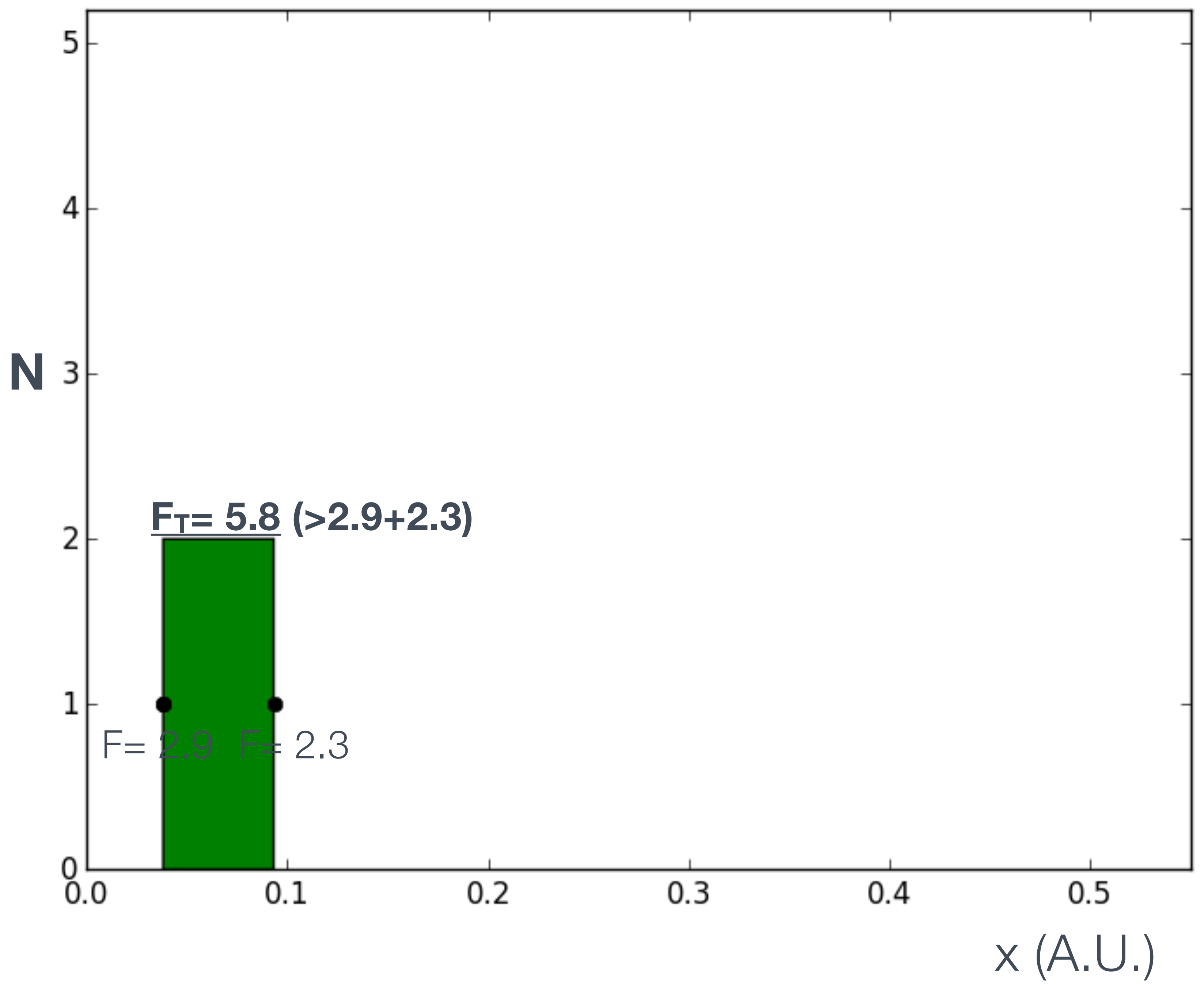
Algorithm Example

- Second data point added.
- Total fitness calculated (F_T is sum of the fitness of all potential blocks)
- For 2 bins, $F_T = 5.2$



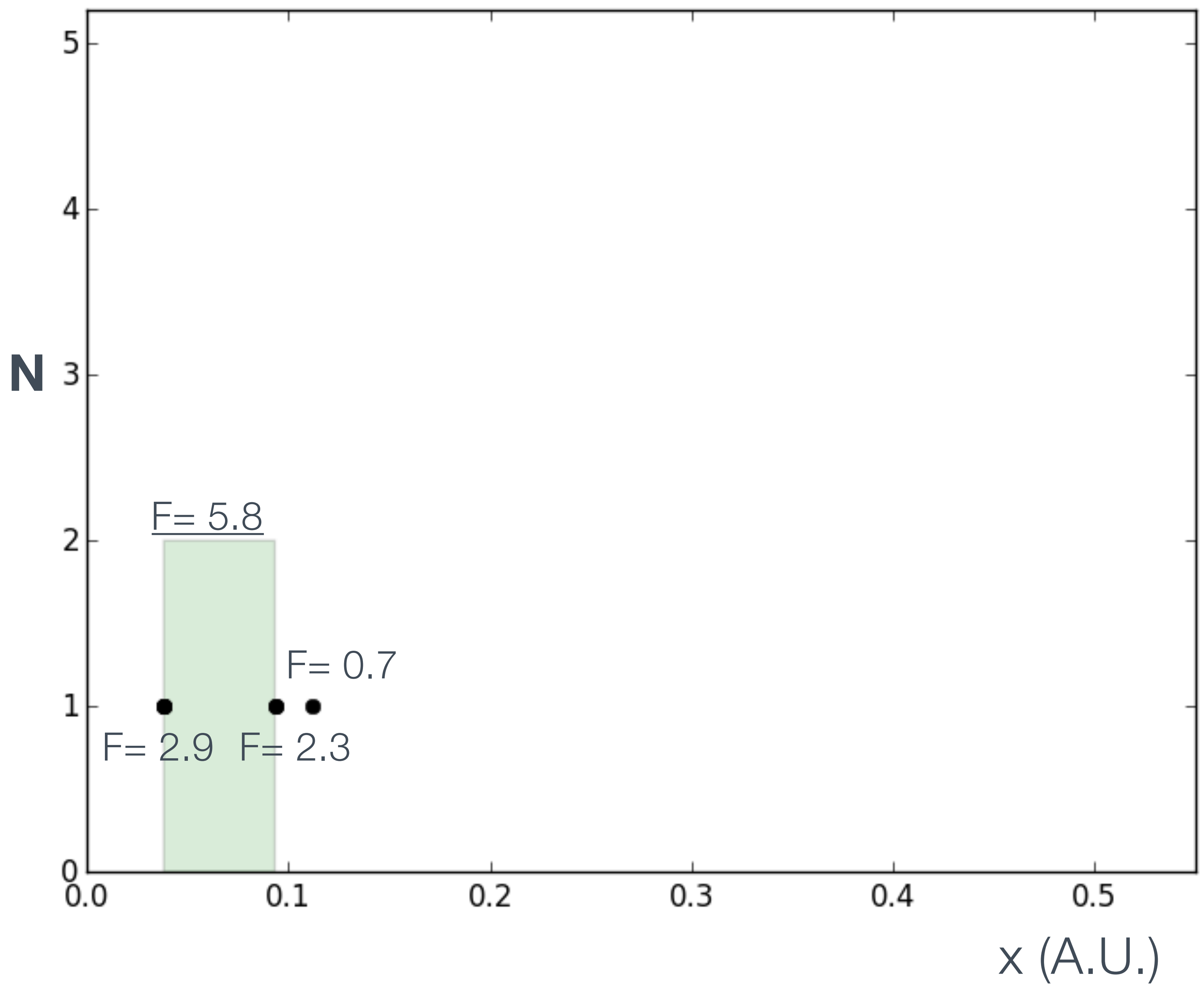
Algorithm Example

- F_T of single bin > F_T of two bins.
- Single bin is chosen.



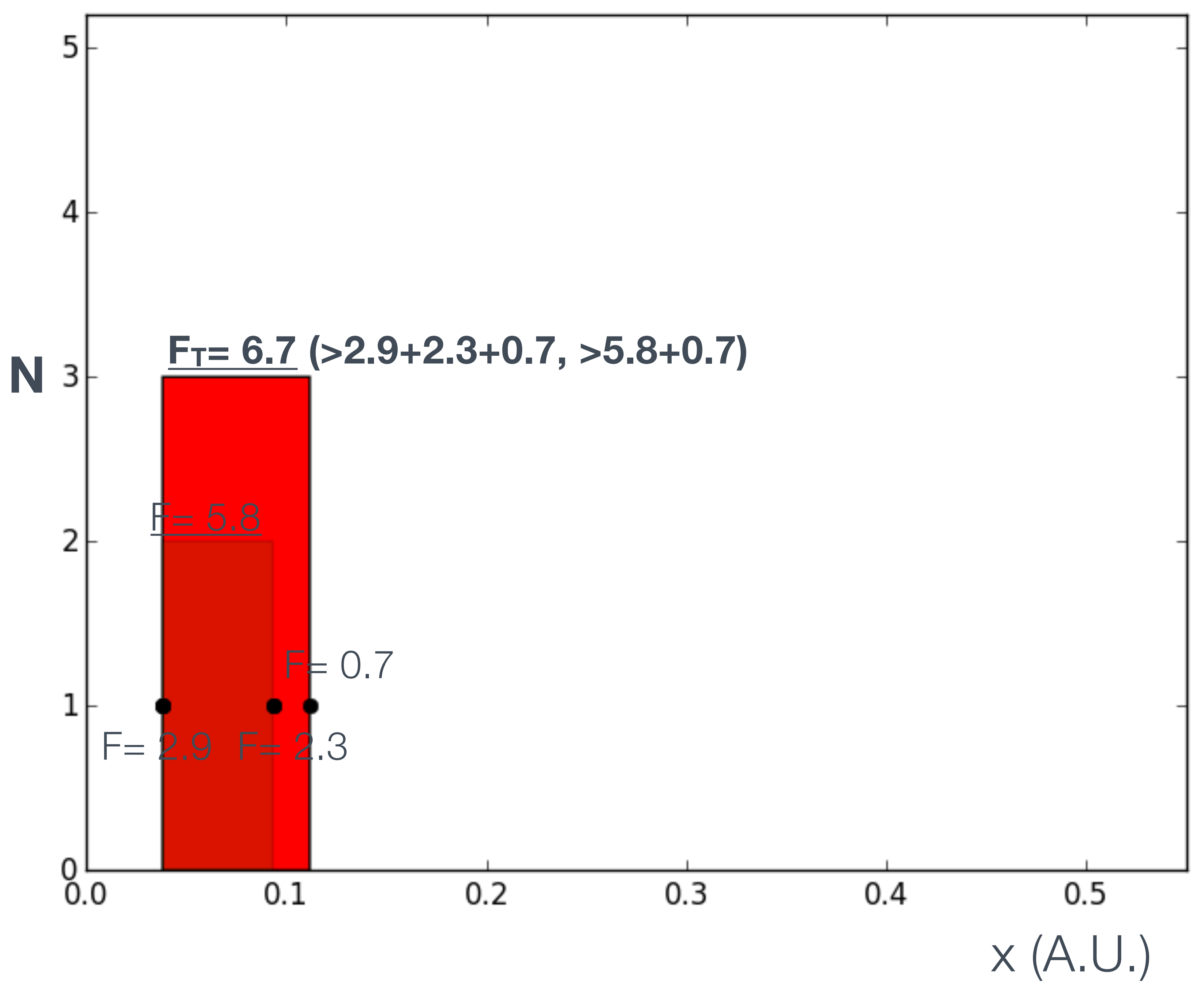
Algorithm Example

- Third data point added



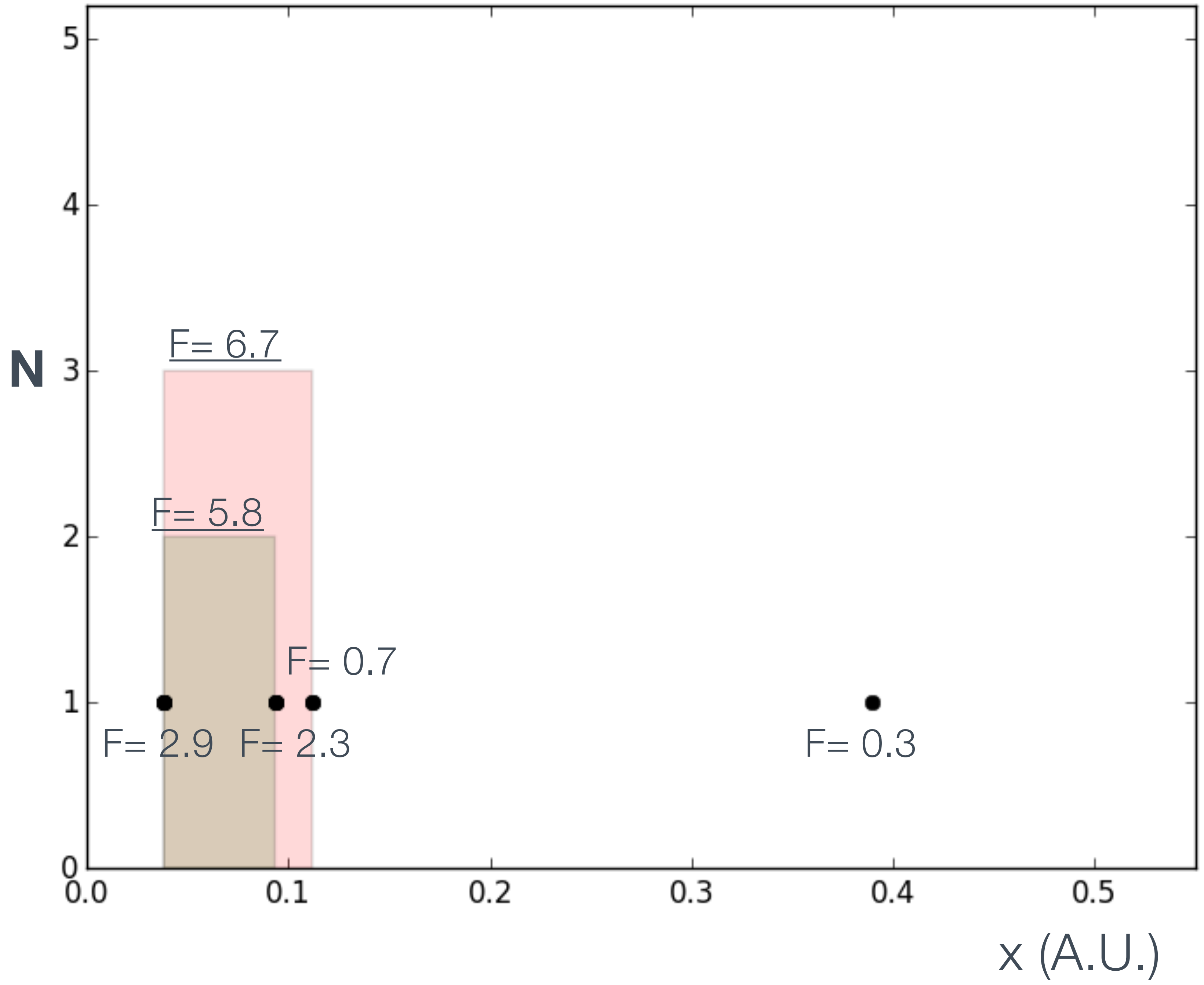
Algorithm Example

- F_T of single bin > F_T of all other combos (using stored F values from previous iterations)



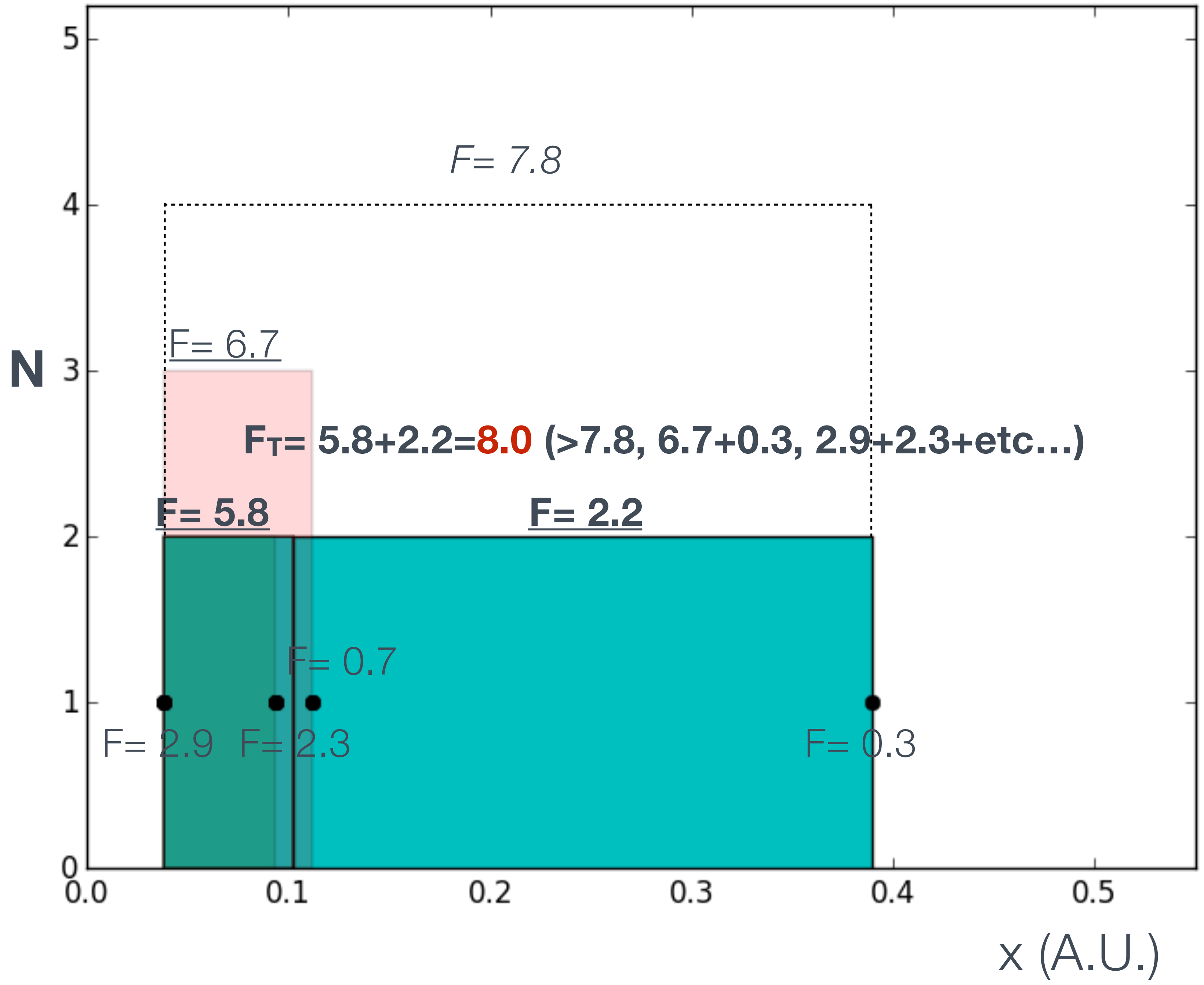
Algorithm Example

- Fourth data point added



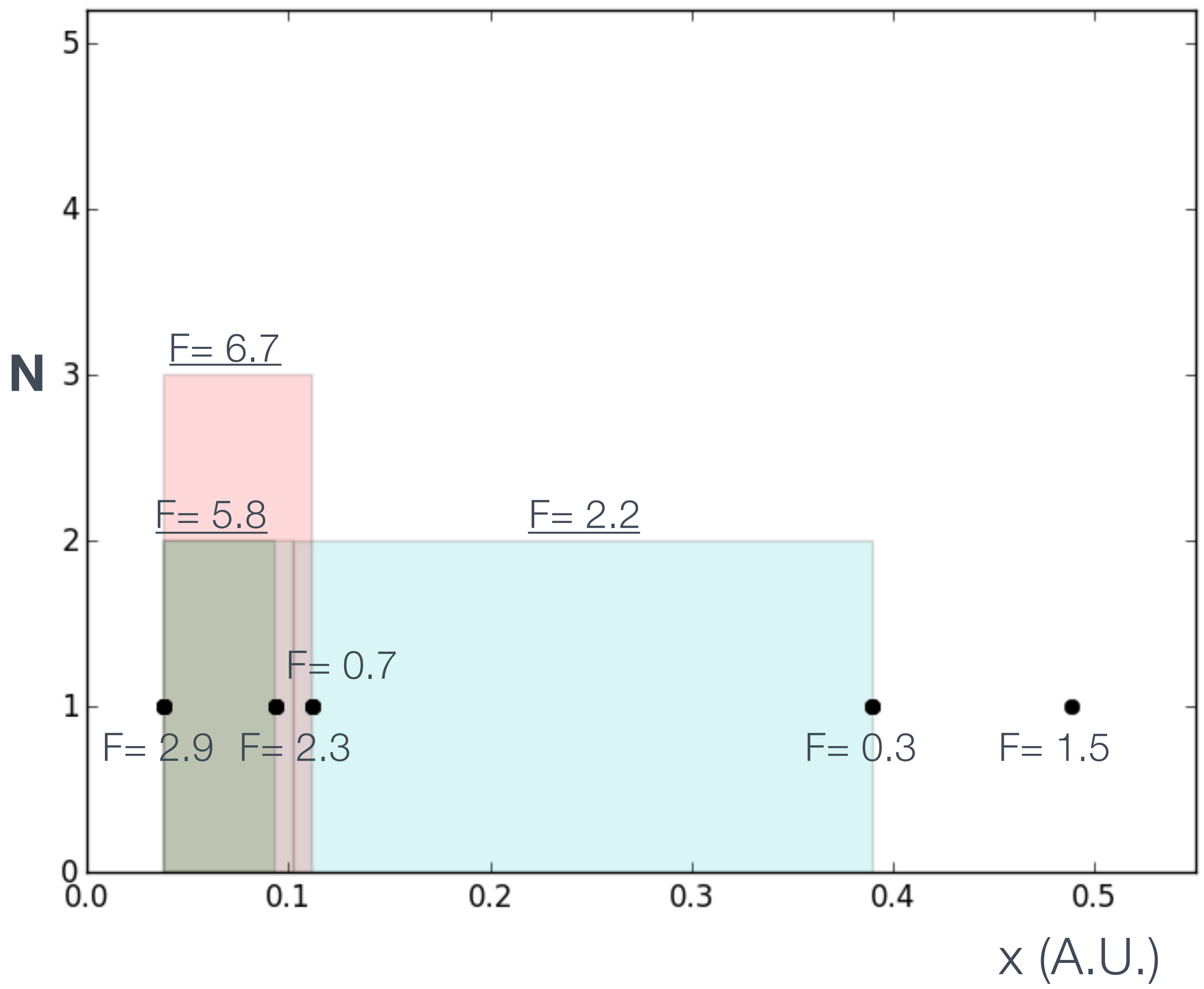
Algorithm Example

- Maximum F_T is for 2 bins
  - ✴*F value of first bin was stored from previous iteration*
- New change-point is determined between pts 2 and 3
- Change-point is saved along with F_T value



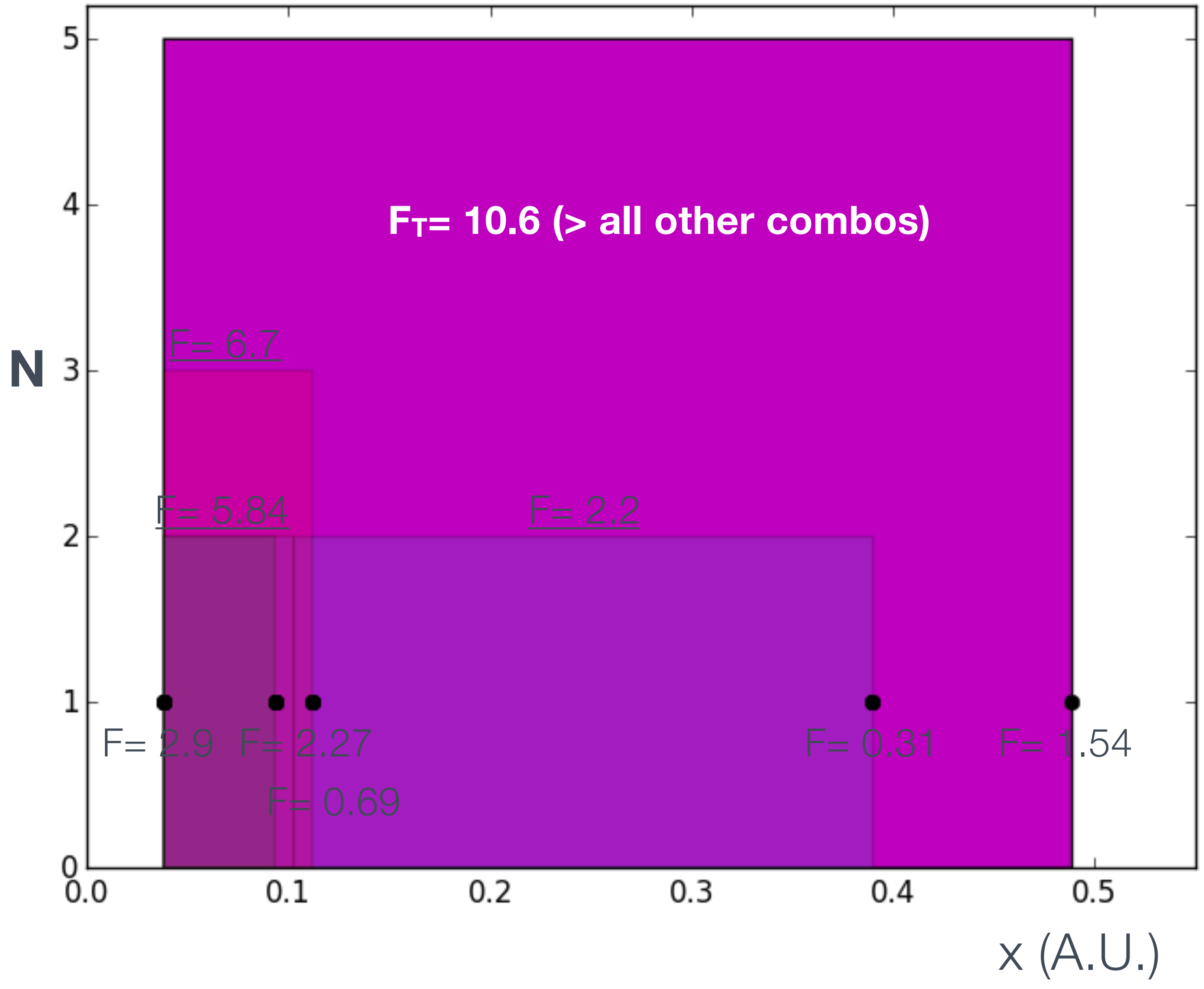
Algorithm Example

- Final data point added

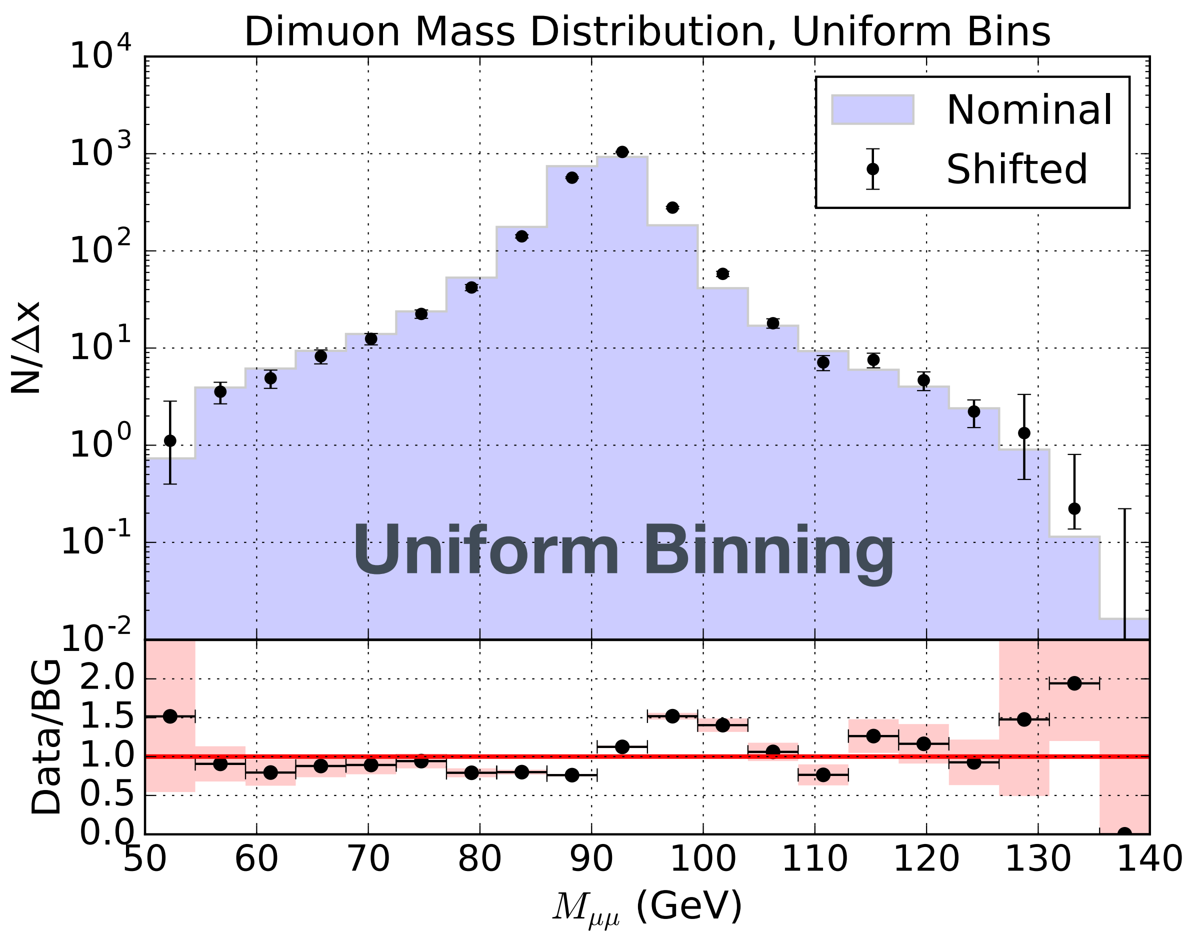


Algorithm Example

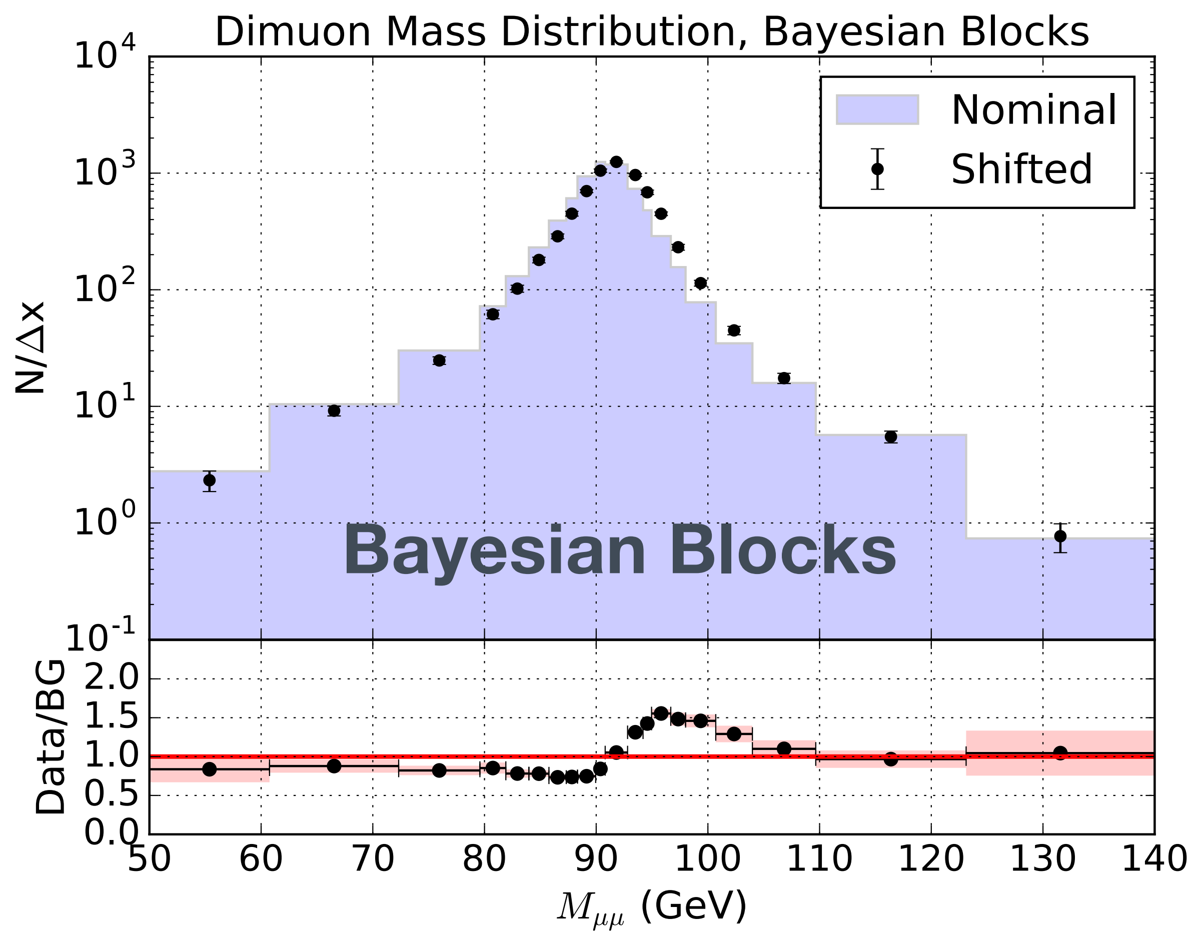
- Maximum F_T is determined to be single bin
- Previous change-point is ignored because of sub-optimal value
- Final result yields bin edges at [1,5]



Visual Impact



(a) Fixed-width binning.



(b) BB binning.

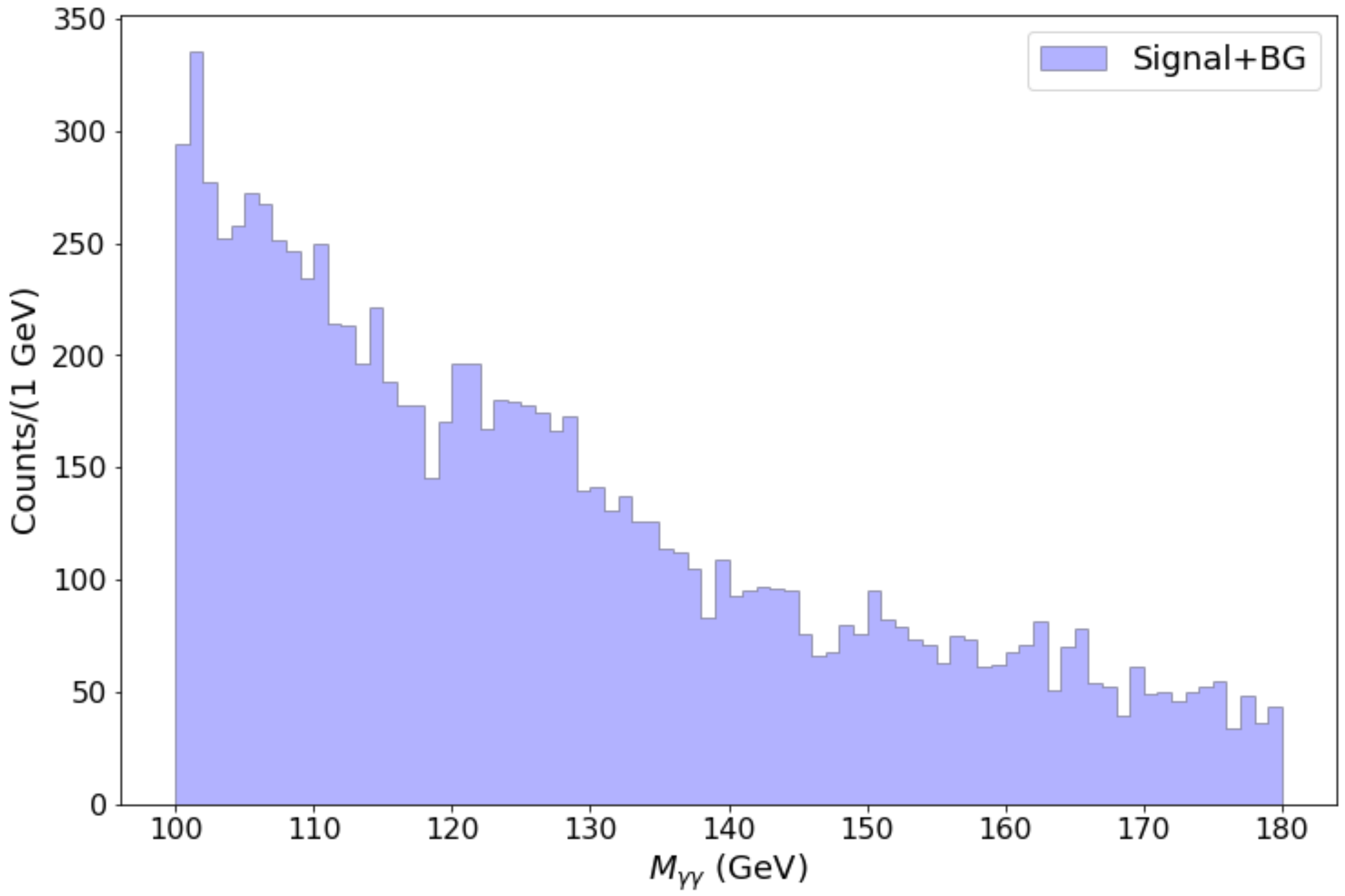
- ★ **Simulated Z→μμ example.**
  - One distribution is slightly shifted w.r.t. other → typical HEP scenario before muon scale corrections are applied.
- ★ **Bayesian Blocks example shows more detail in peak, smooths out statistical fluctuation in tails.**

Bump Hunting

★ **The bin edges determined by Bayesian Blocks are statistically significant.**

- Can they assist with analyses, outside of purely visual?

★ **Consider the H→γγ discovery (simulated):**



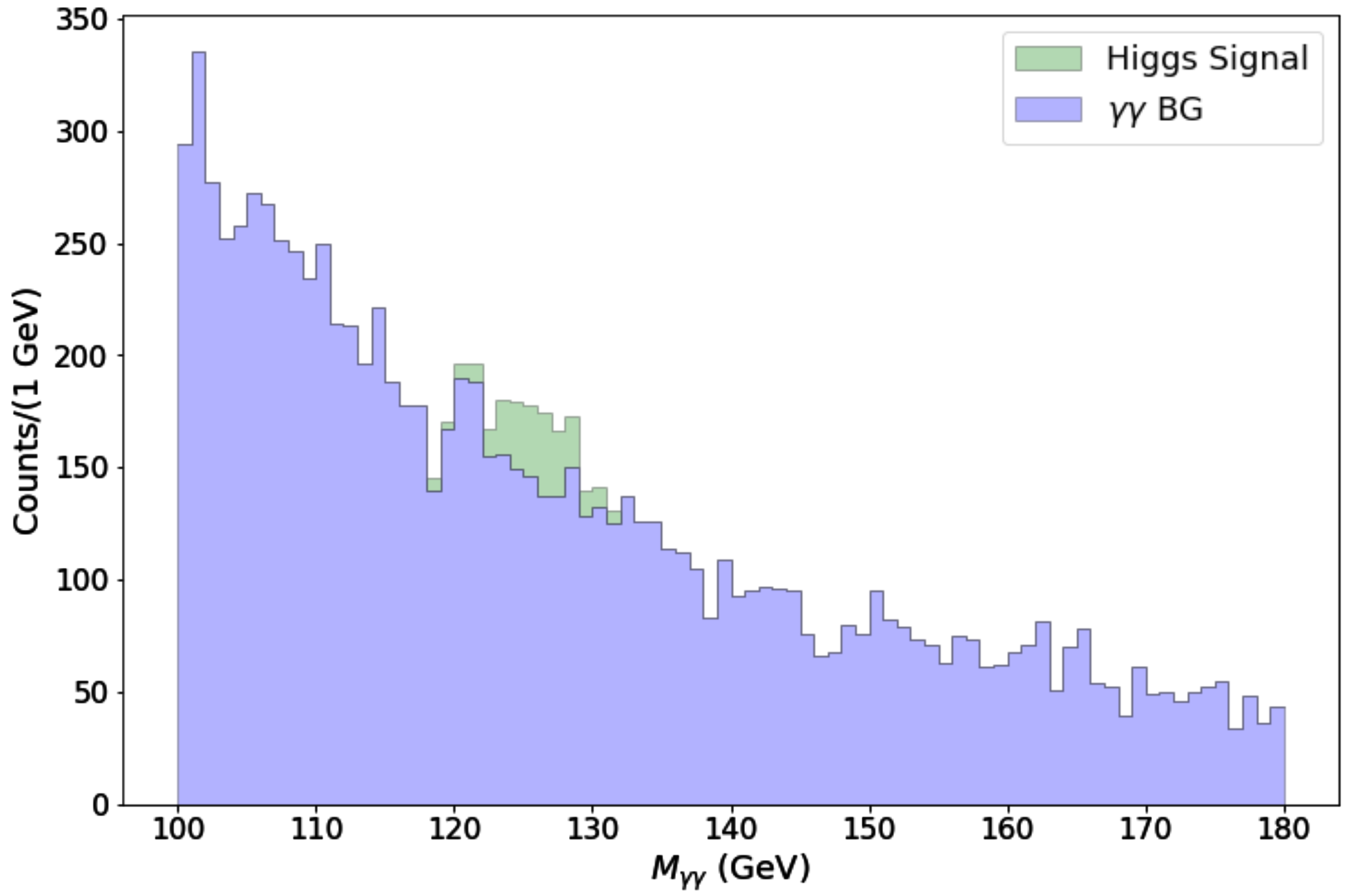
- Falling diphoton BG, ~10k events.
- ~230 Higgs signal events at $M_{\gamma\gamma}$=125 GeV (~5 σ excess)

Bump Hunting

★ **The bin edges determined by Bayesian Blocks are statistically significant.**

- Can they assist with analyses, outside of purely visual?

★ **Consider the H→γγ discovery (simulated):**

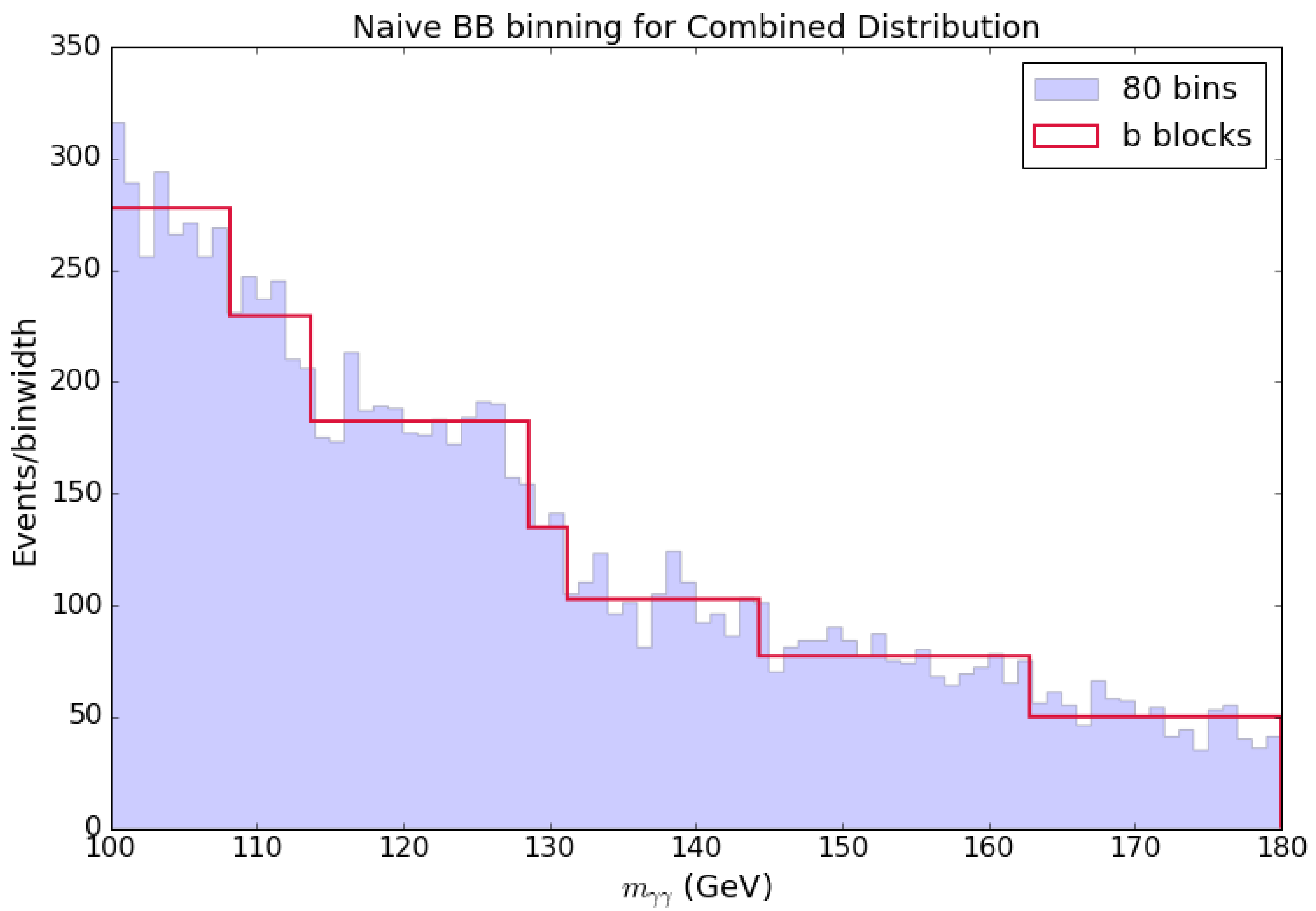


- Falling diphoton BG, ~10k events.
- ~230 Higgs signal events at $M_{\gamma\gamma}$=125 GeV (~5 σ excess)

Significant excess, difficult to discern by eye.

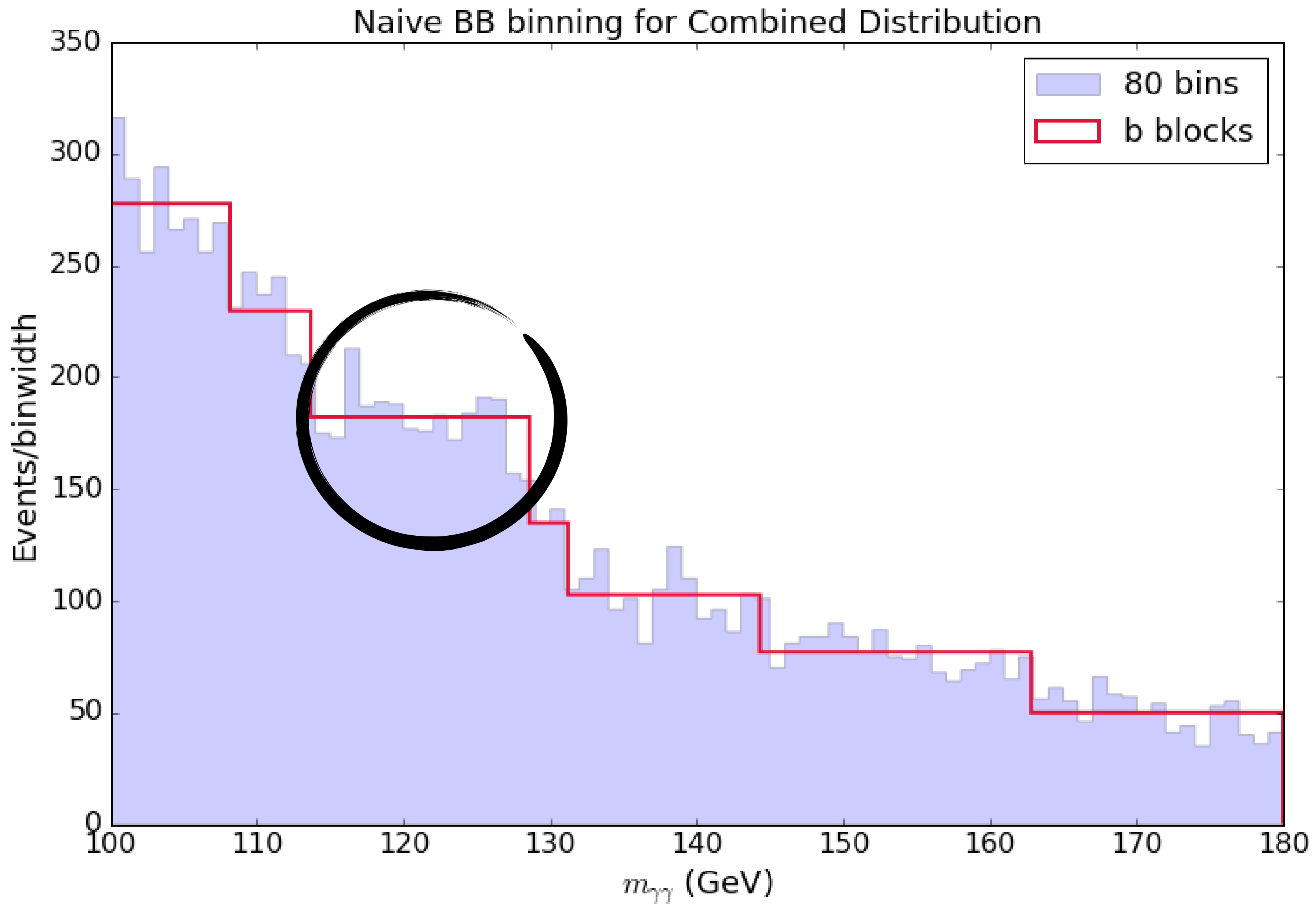
Bump Hunting

First try, naive binning of signal+background:



Bump Hunting

First try, naive binning of signal+background:



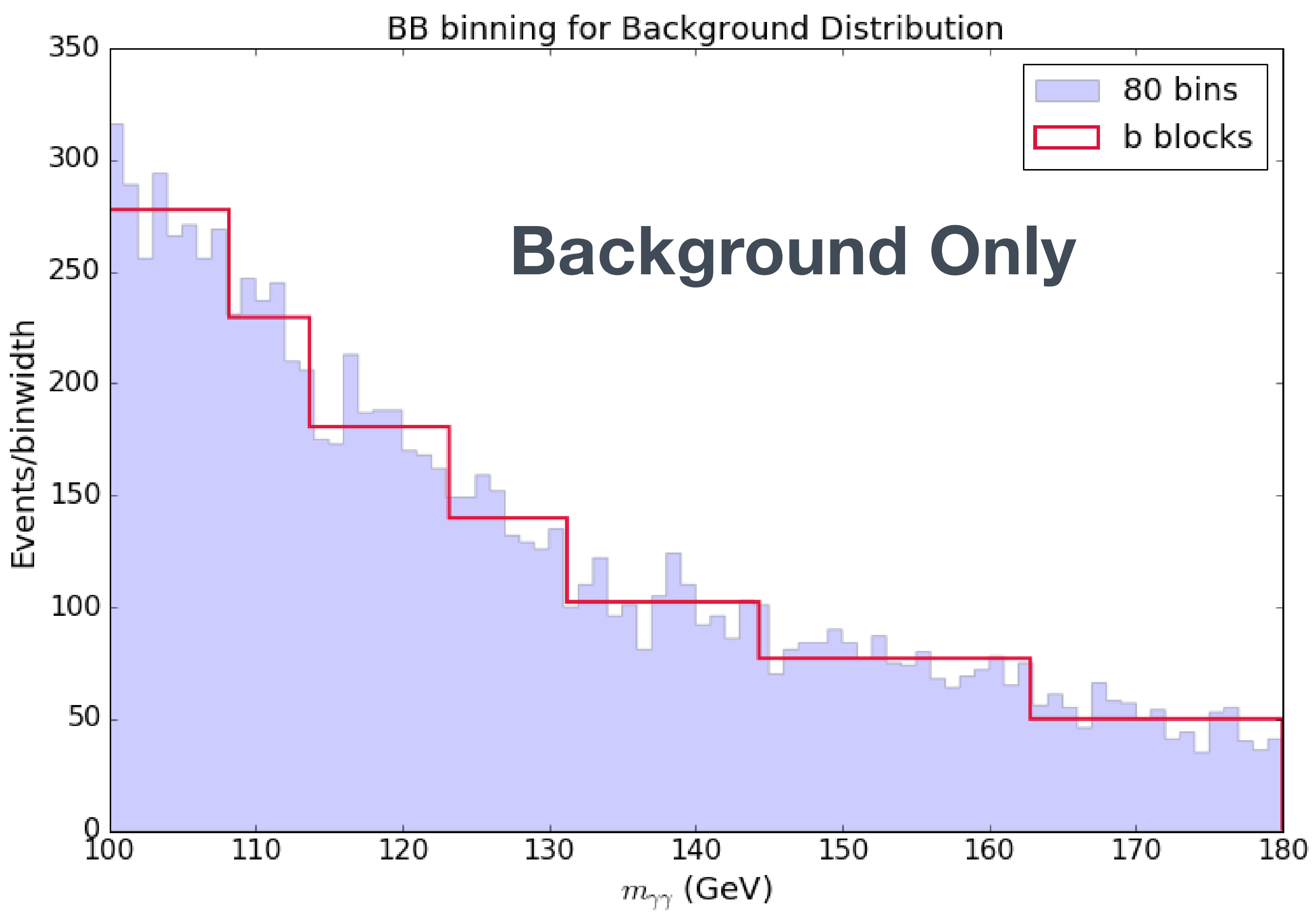
Results not great.

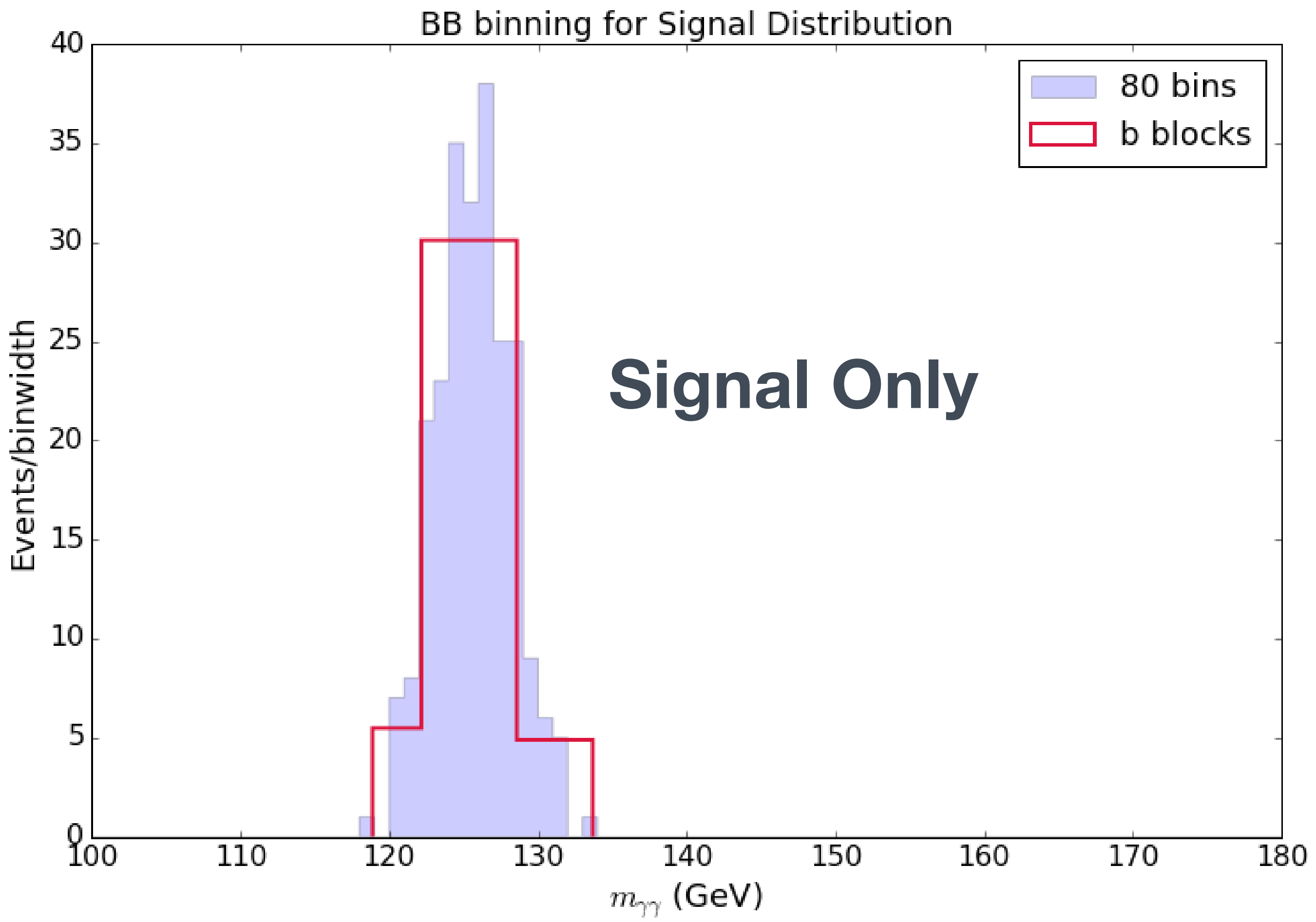
Falling background + rising signal = one large bin.

Bump Hunting

★ **Generate a "hybrid" binning, leveraging knowledge of signal shape:**

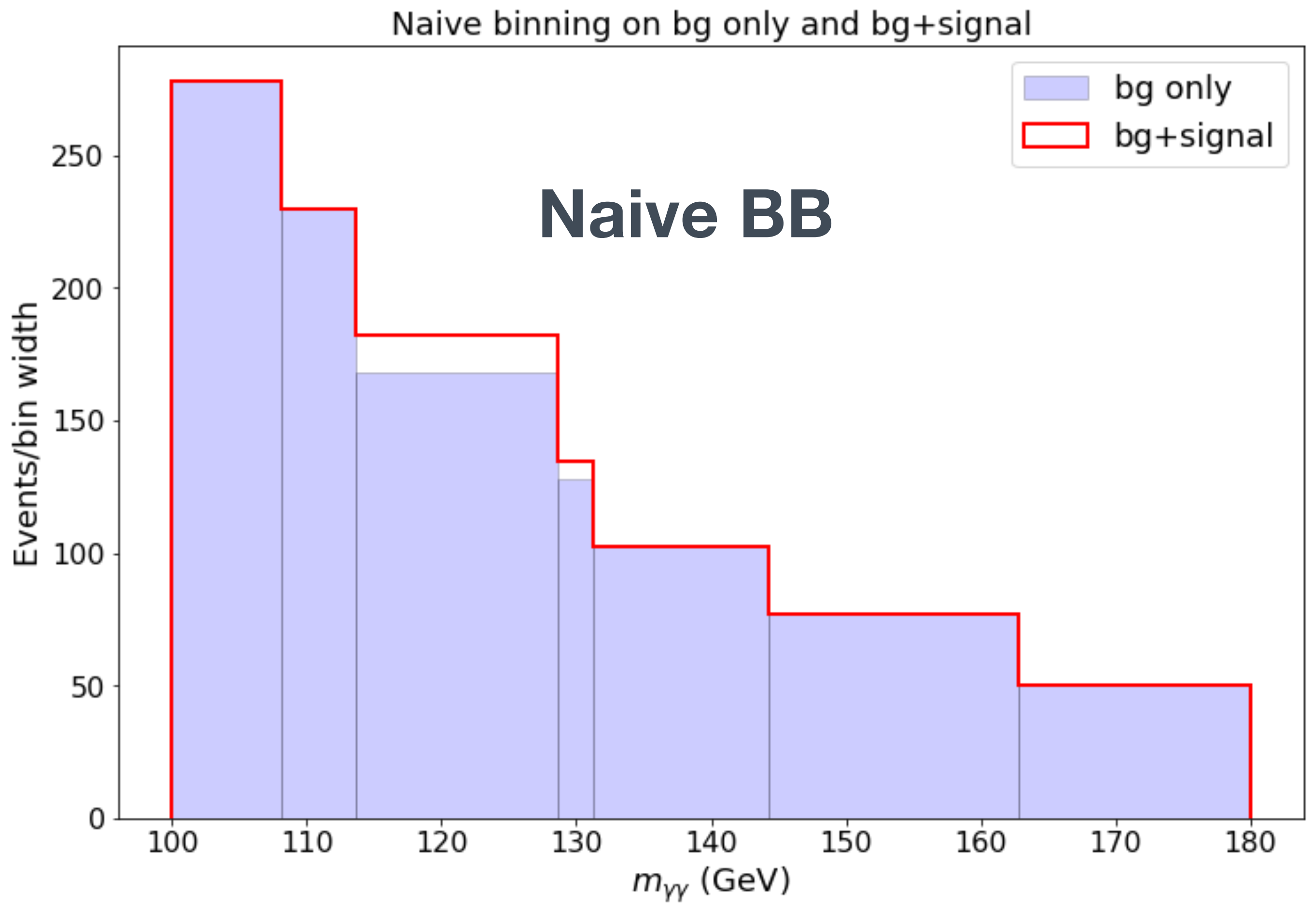
- Use Bayesian Blocks on simulated signal and background templates.
- Combine the bin edges (background bin edges in signal region replaced by signal bin edges)

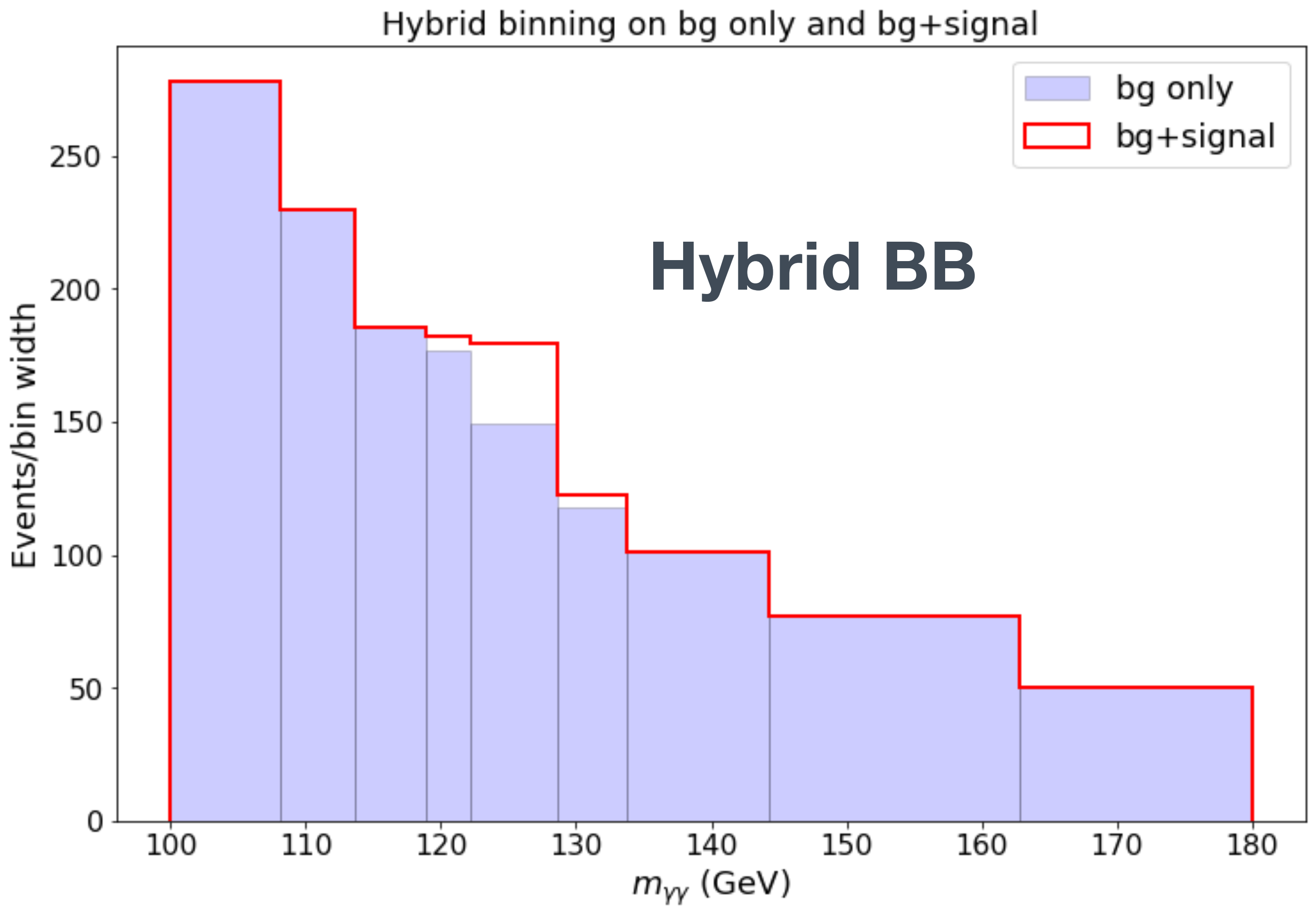




Bump Hunting

★ **Signal excess much more apparent with hybrid binning:**



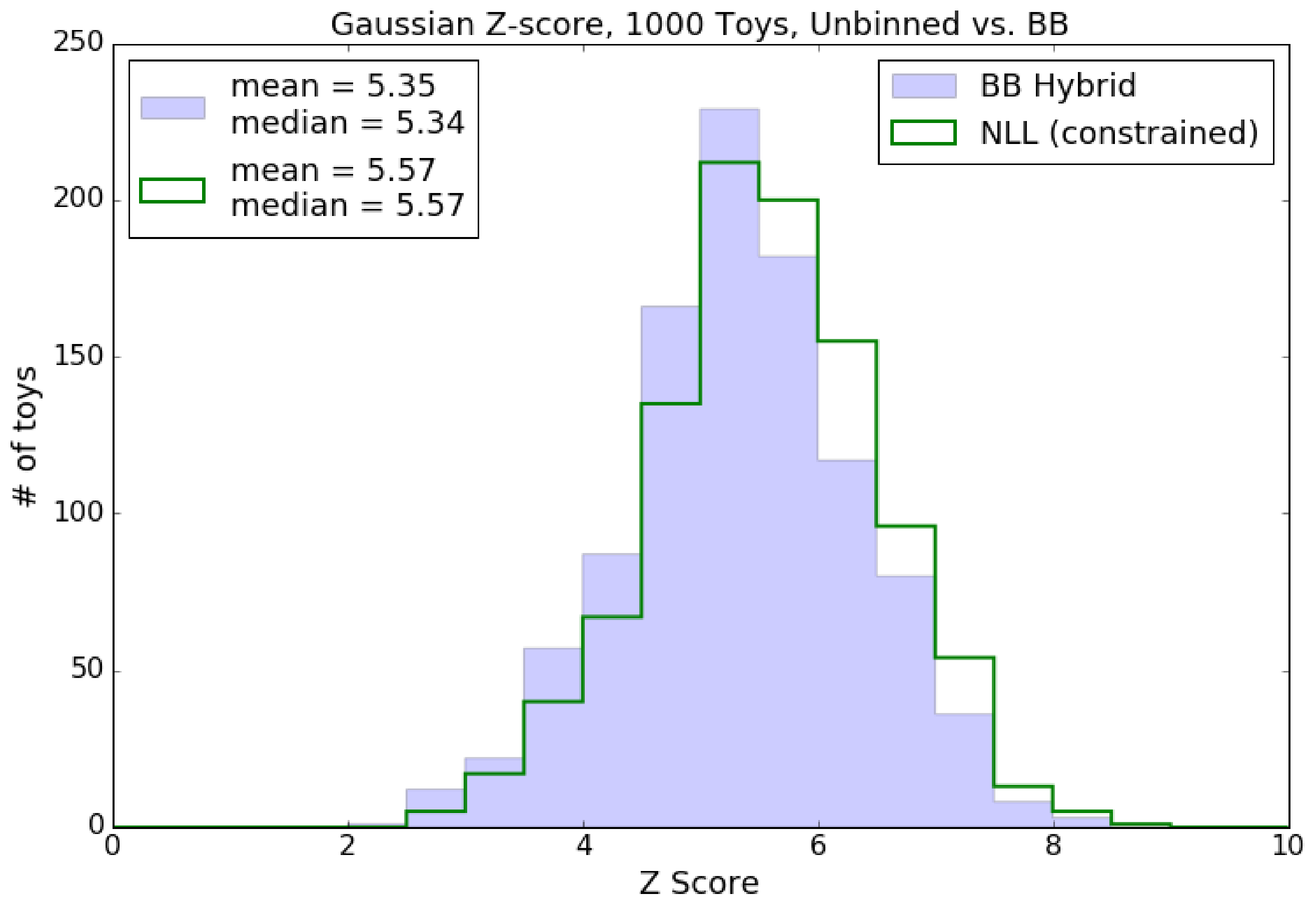


No parametric models used to generate binning, completely MC dependent.

What is the sensitivity of this excess?

Bump Hunting

- **Calculate Gaussian Z-score (# of σ excess) for 1000 simulations, and compare to unbinned likelihood from known underlying pdfs.**
  - Z-score from unbinned likelihood are the <u>upper-bound</u>.



Mean Z-scores:

Bayesian Blocks Template: 5.35 σ

Unbinned likelihood: 5.57 σ

Hybrid binning is only slightly less sensitive than unbinned pdf, and is completely non-parametric!

Software

★ **Python histogramming package developed for HEP:**

- Wraps matplotlib, adds automatic error bars, scaling, Bayesian Blocks binning, and more!

★ **Install with pip:**

- `$ pip install histogram_plus`

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from histogram_plus.hist_funcs import hist

        %matplotlib inline
        plt.rcParams['figure.figsize'] = (12,8)
```

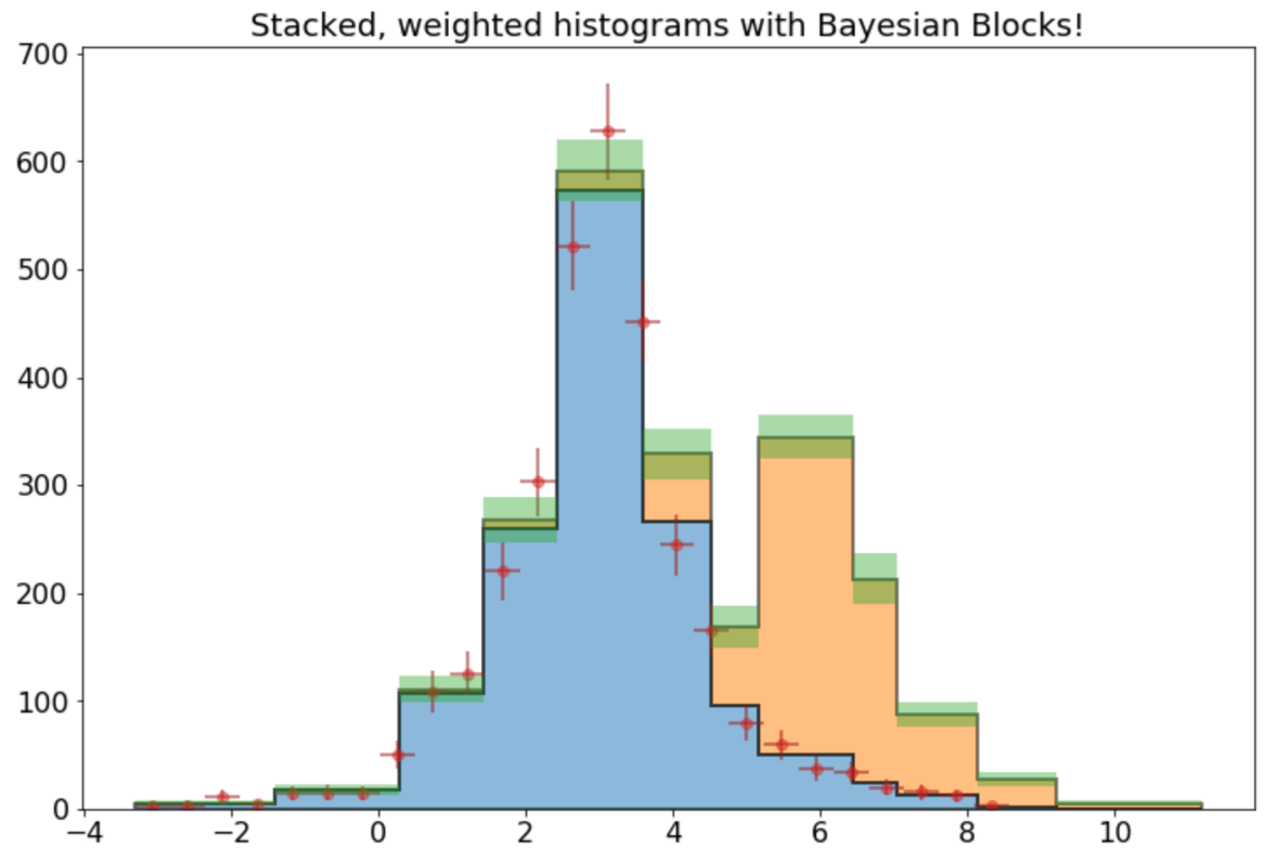
```
In [2]: np.random.seed(8008)
        data = np.random.laplace(3, 1, size=1000)
        data2 = np.random.laplace(6, 1, size=500)
        weights = np.random.uniform(1,2, size=1000)
        weights2 = np.random.uniform(1,2, size=500)
```

```
In [3]: stacked_hists = hist([data,data2], weights=[weights, weights2], bins='blocks',
                             stacked=True, errorbars=True, scale='binwidth', p0=0.01)

        marker_hists = hist(data, bins=25, weights=weights, histtype='marker',
                            errorbars=True, scale='binwidth')

        plt.title('Stacked, weighted histograms with Bayesian Blocks!')

Out[3]: <matplotlib.text.Text at 0x113b93b50>
```



Documentation (in progress):

https://brovercleveland.github.io/histogram_plus/

Summary

- ★ **The Bayesian Blocks algorithm is a data-driven, model-independent method for binning.**
  - Bins are variable-width, edges represent statistically significant changes in data.
  - Improves visualization of distributions, even with dense peaks and sparse tails.
- ★ **Bayesian Blocks can also assist in template-based analyses.**
  - Provides a non-parametric way of modeling distributions in histograms, with minimal loss in sensitivity when compared to unbinned methods.
- ★ **New paper on HEP application for Bayesian Blocks:**
  - https://arxiv.org/abs/1708.00810