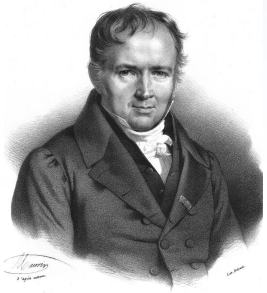
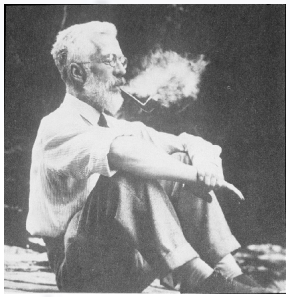
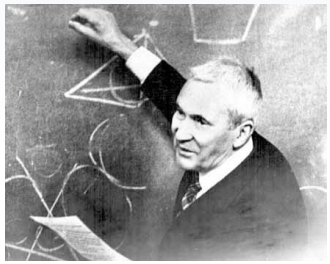
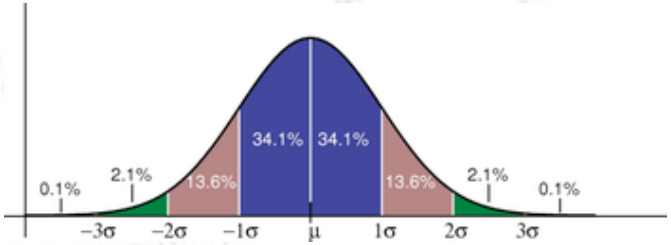
Applied Statistics

Notes on binning



Troels C. Petersen (NBI)



"Statistics is merely a quantisation of common sense"

Binning is a (small) art

When producing a histogram, you **have to consider the binning**, i.e. range and number of bins. Unfortunately, **matplotlib** does not force you to make any choices. Make sure you don't make that mistake…

Data can be very tricky, as several features may make your life hard. Things you have to consider are:

- Is the data discrete or not? This may seem trivial, but sometimes it is not.
- If integer values only, then make sure you make bins from i-0.5 to i+0.5. Why?
- If discrete, make sure you understood the discreteness and plot accordingly.
- Consider, if you need to include all data or just a range?
- Consider, if you can put several categories of data into the same histogram?
- Ensure that your binning width is a "reasonable" size (0.1 and not 1/7), and…
  - Consider the amount of statistics.
  - Consider the size of features in the data.
- Should the y-axis be linear or logarithmic? Should the x-axis?
- Will an insert plot be worthwhile? Should you transform the data?
- And… the style/choice of colours/legend/axis/titles/etc. is still to consider!

An example…

Consider this data…

```
# Choose data paramters:
Ndata = 1000000
Npeak = 250
mu = 3.95
sigma = 0.004
byte = 1024.0

# Generate data:
ideal_data = np.concatenate(( 1.0/r.uniform(size=Ndata)-1.0, np.full(Npeak, mu, dtype=float)), axis=0)
smearing = r.normal(0.0, sigma, size=Ndata+Npeak)
data = np.around(byte*(ideal_data + smearing))/byte
```

The data is discrete, has a long tail, is smeared, has a small peak "somewhere"…

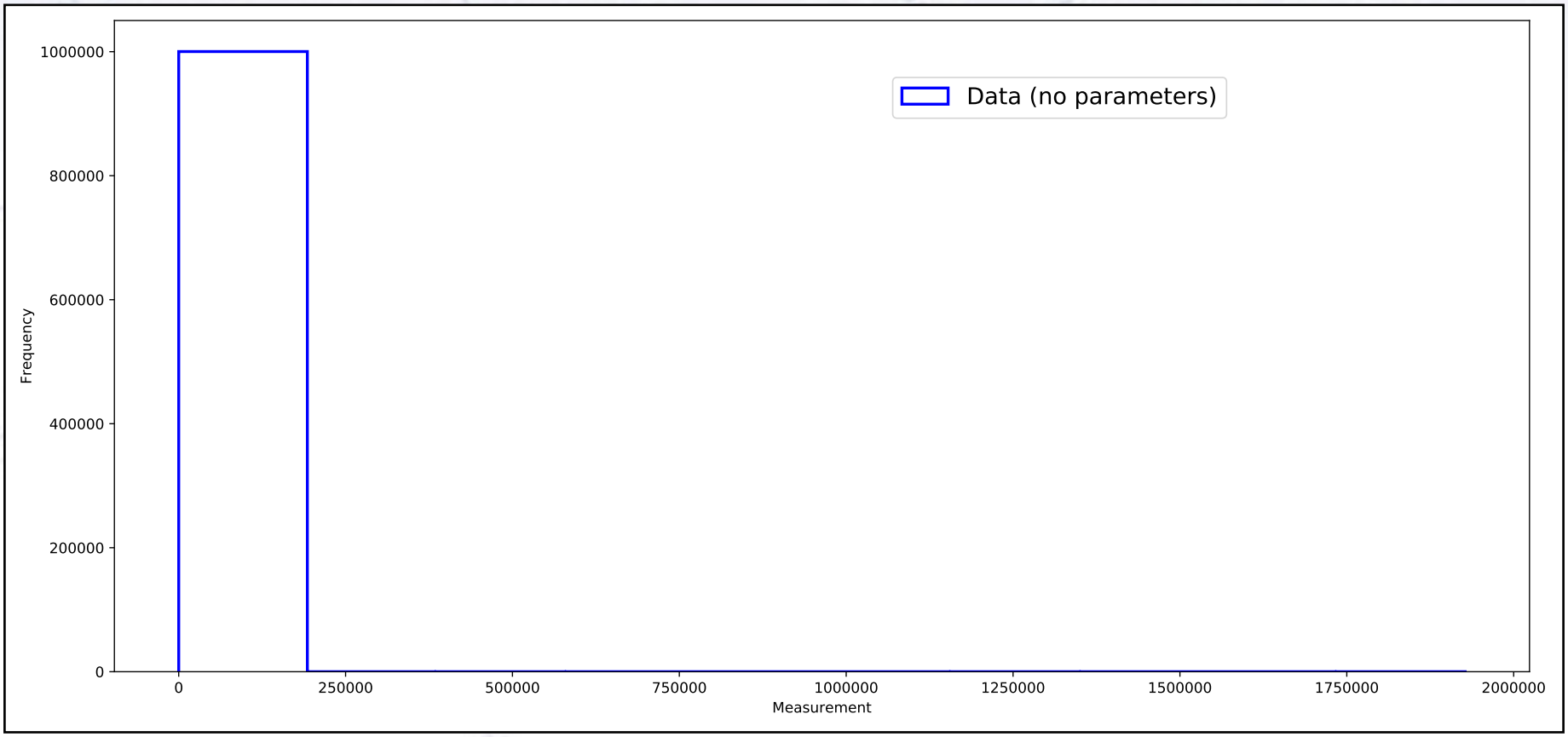
…in summary: A nightmare

But let us dive into putting this into a histogram, and see where it goes.

Histogram 1

Let Python decide the range and binning!

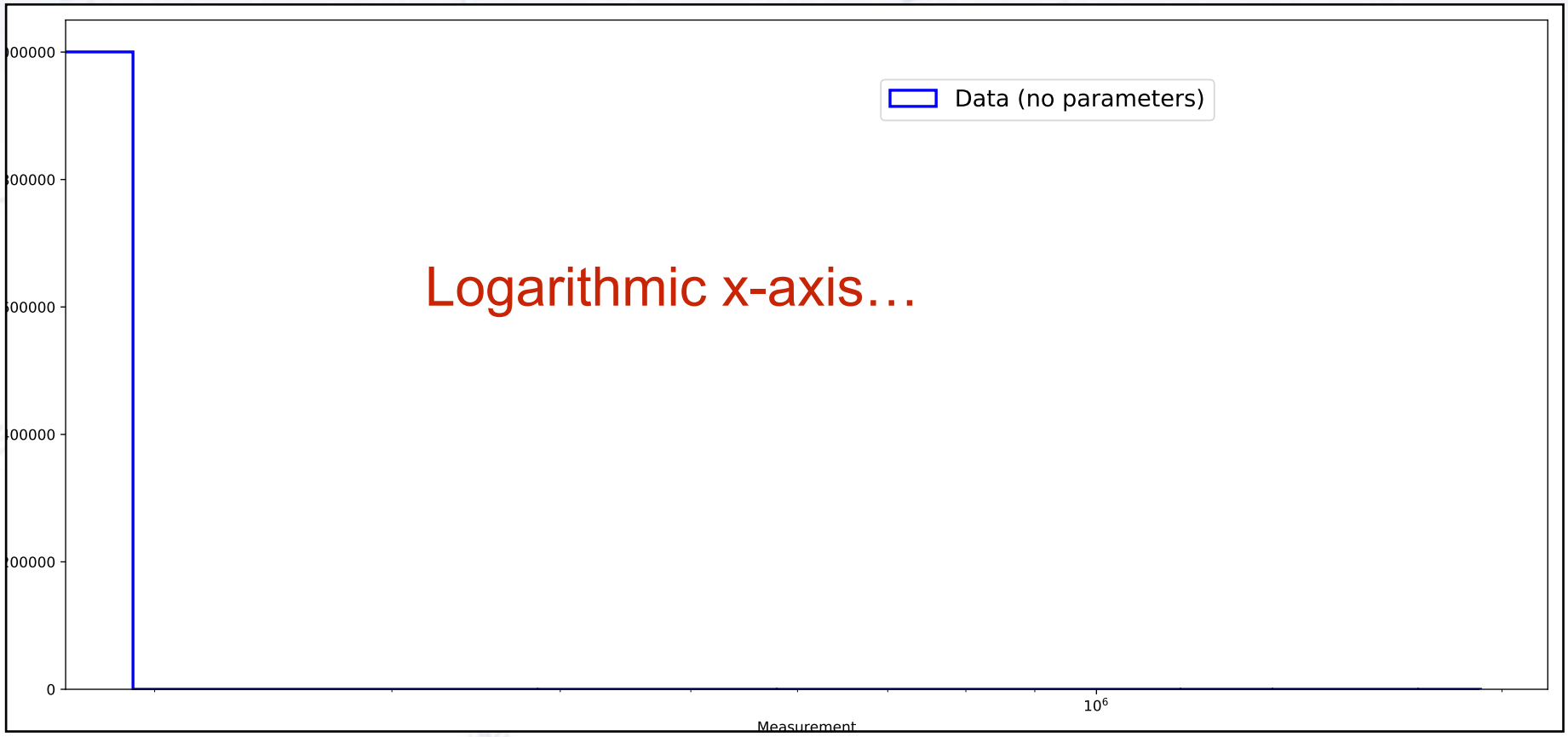
…a straight disaster. But maybe we can fix it by making the axis logarithmic.



Histogram 1

Let Python decide the range and binning!

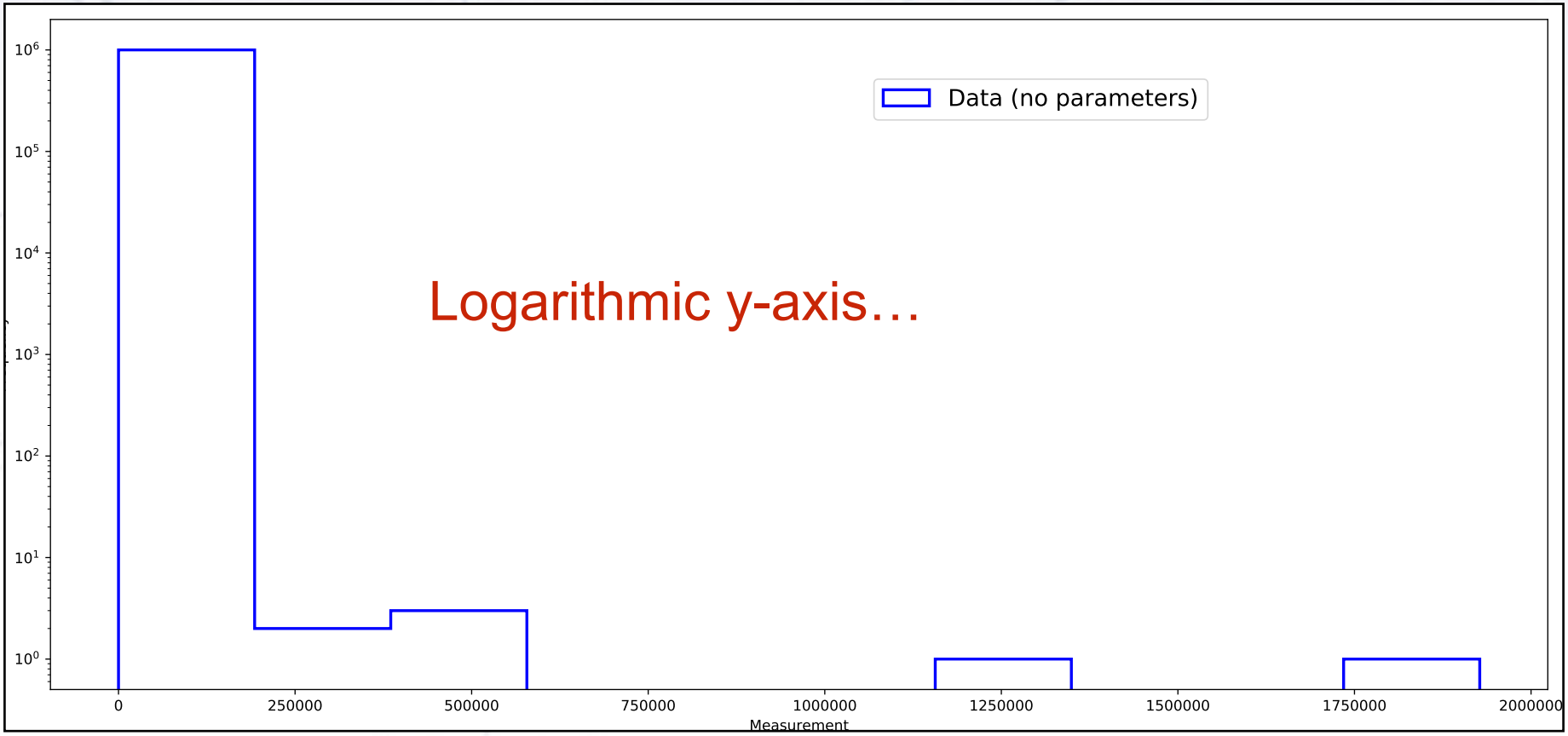
…a straight disaster. But maybe we can fix it by making the axis logarithmic.



Histogram 1

Let Python decide the range and binning!

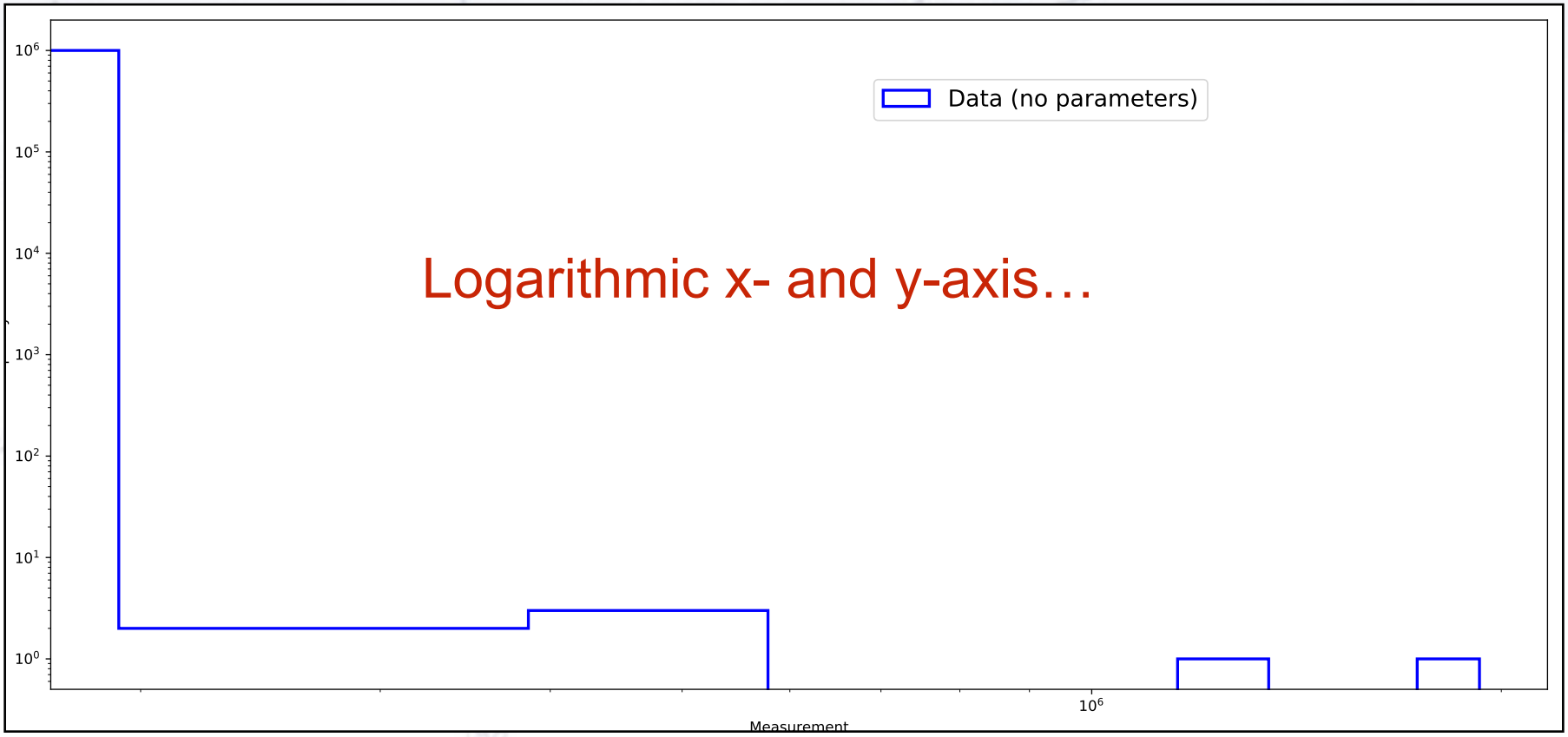
…a straight disaster. But maybe we can fix it by making the axis logarithmic.



Histogram 1

Let Python decide the range and binning!

…a straight disaster. But maybe we can fix it by making the axis logarithmic.

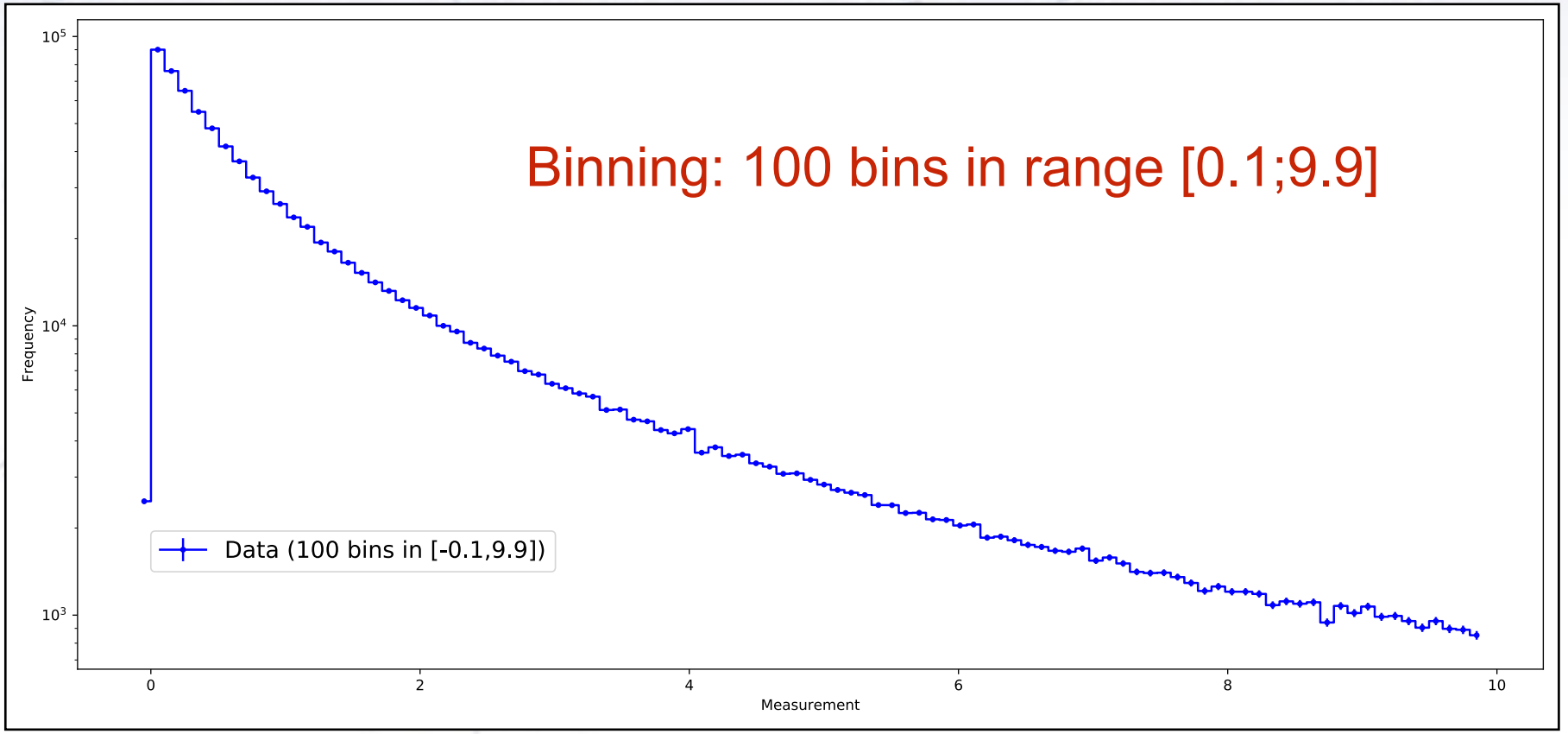


Status 1

Nothing worked… the plots are downright USELESS!

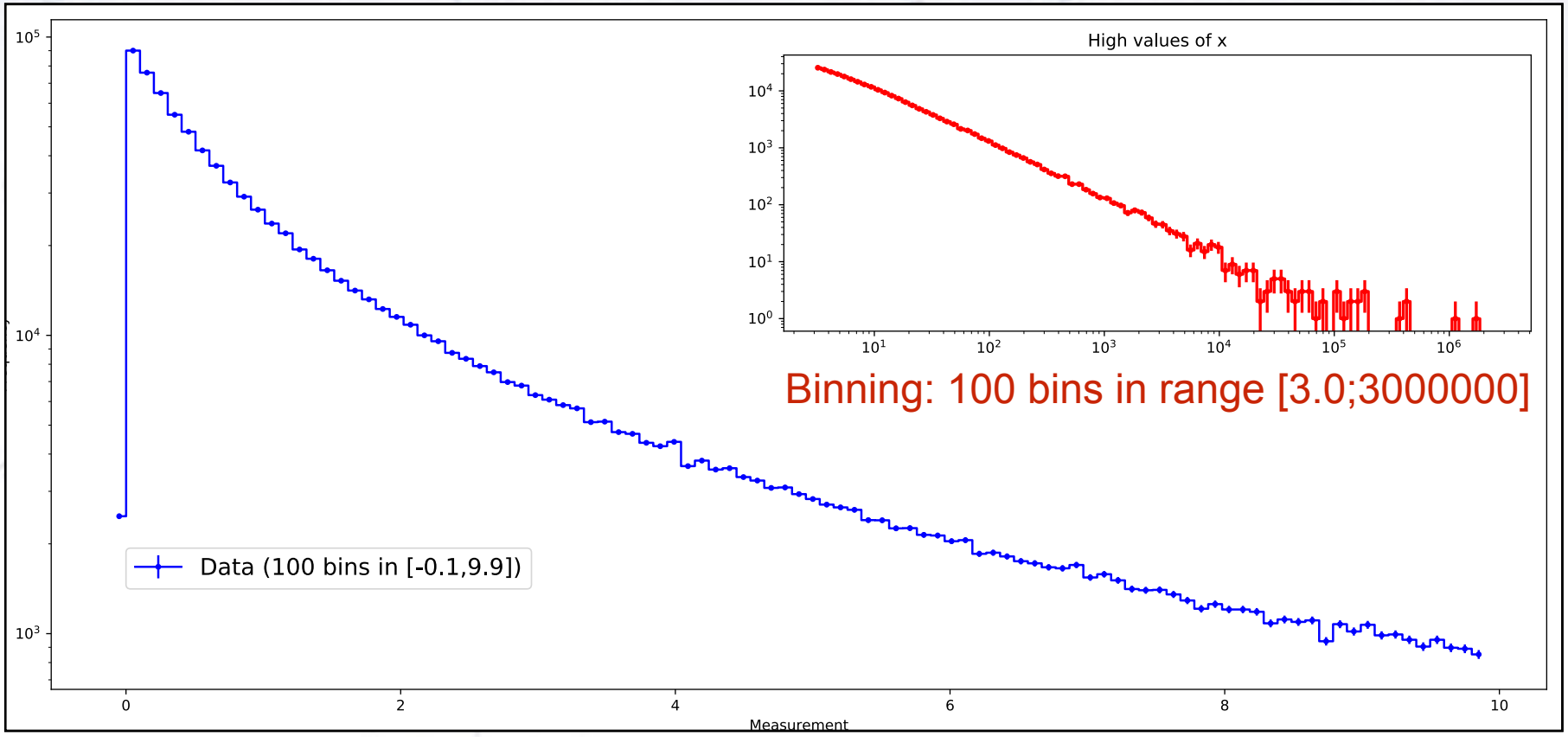
Histogram 2

Since the low values dominates the data, perhaps a way forward is to start plotting this:



Histogram 2

Since the low values dominates the data, perhaps a way forward is to start plotting this.
Not only does this give something sensible, but also makes the log-log-plot work:



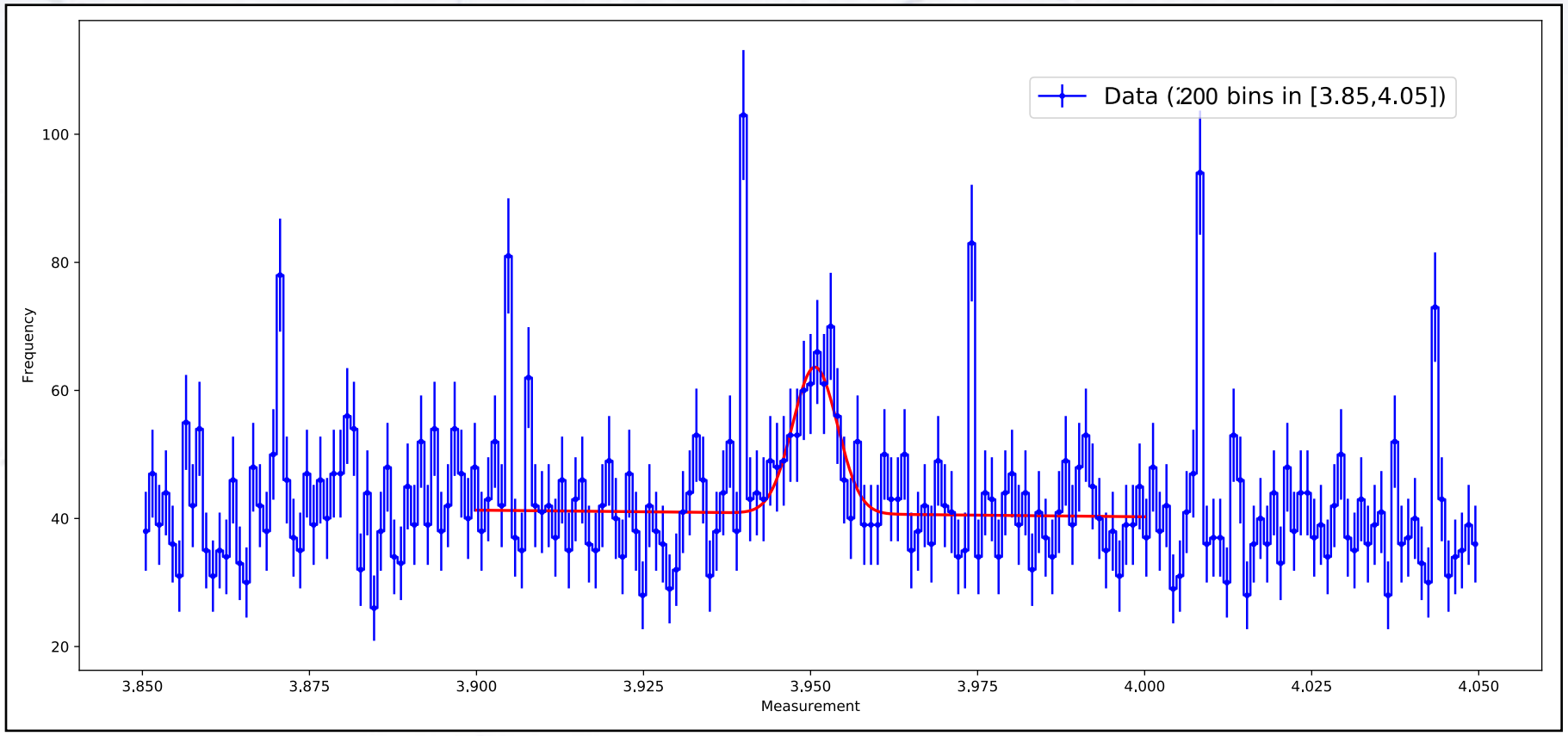
Status 2

Choosing the binning range solves part of the problem.
Now we can see the general distributions of the data.

…but what was that small peak in the middle?

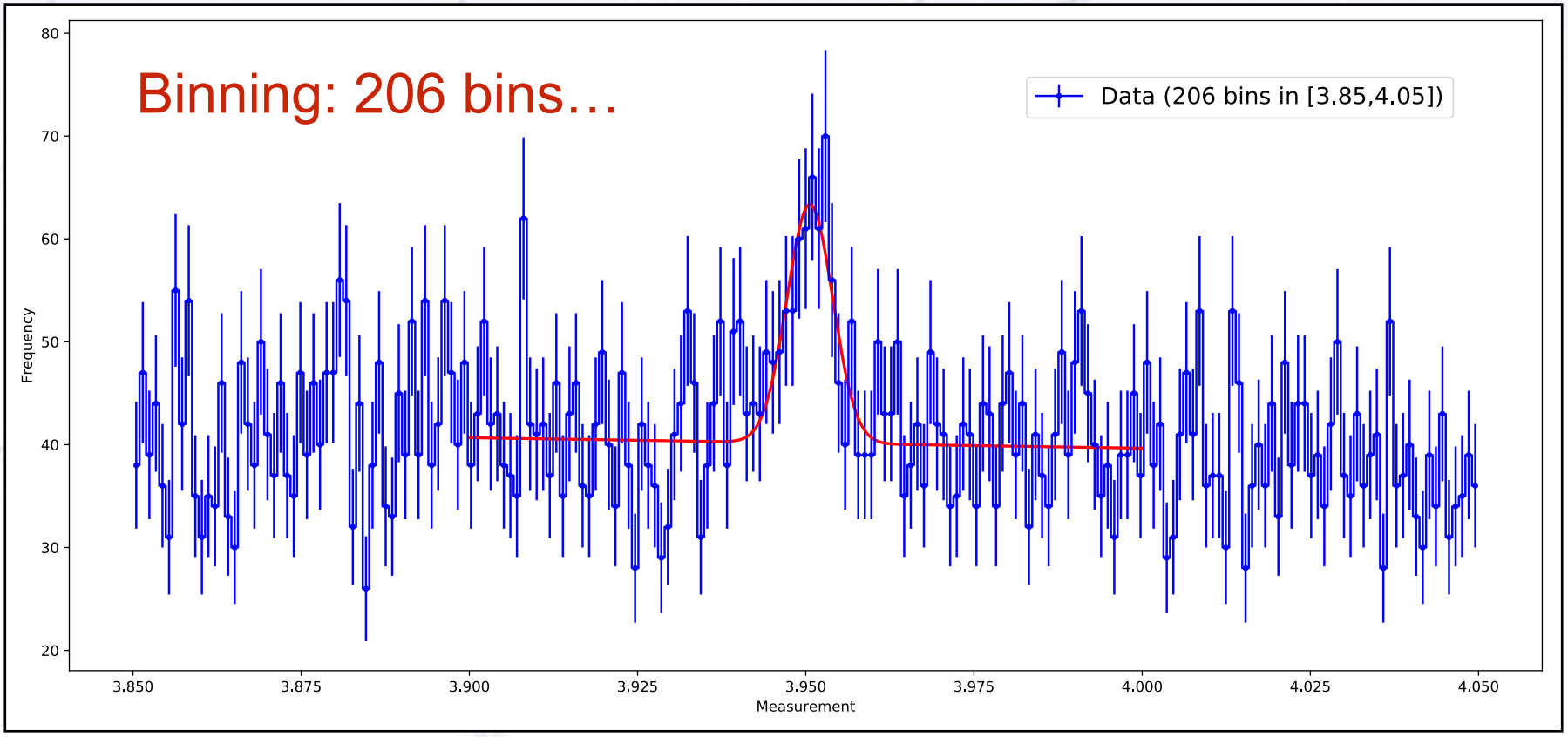
Histogram 3

When plotting a small range, one can see a small but significant Gaussian peak, but also six “spikes”. What is that?
Well, those a bins, where two values fall in one bin, resulting in a double value!



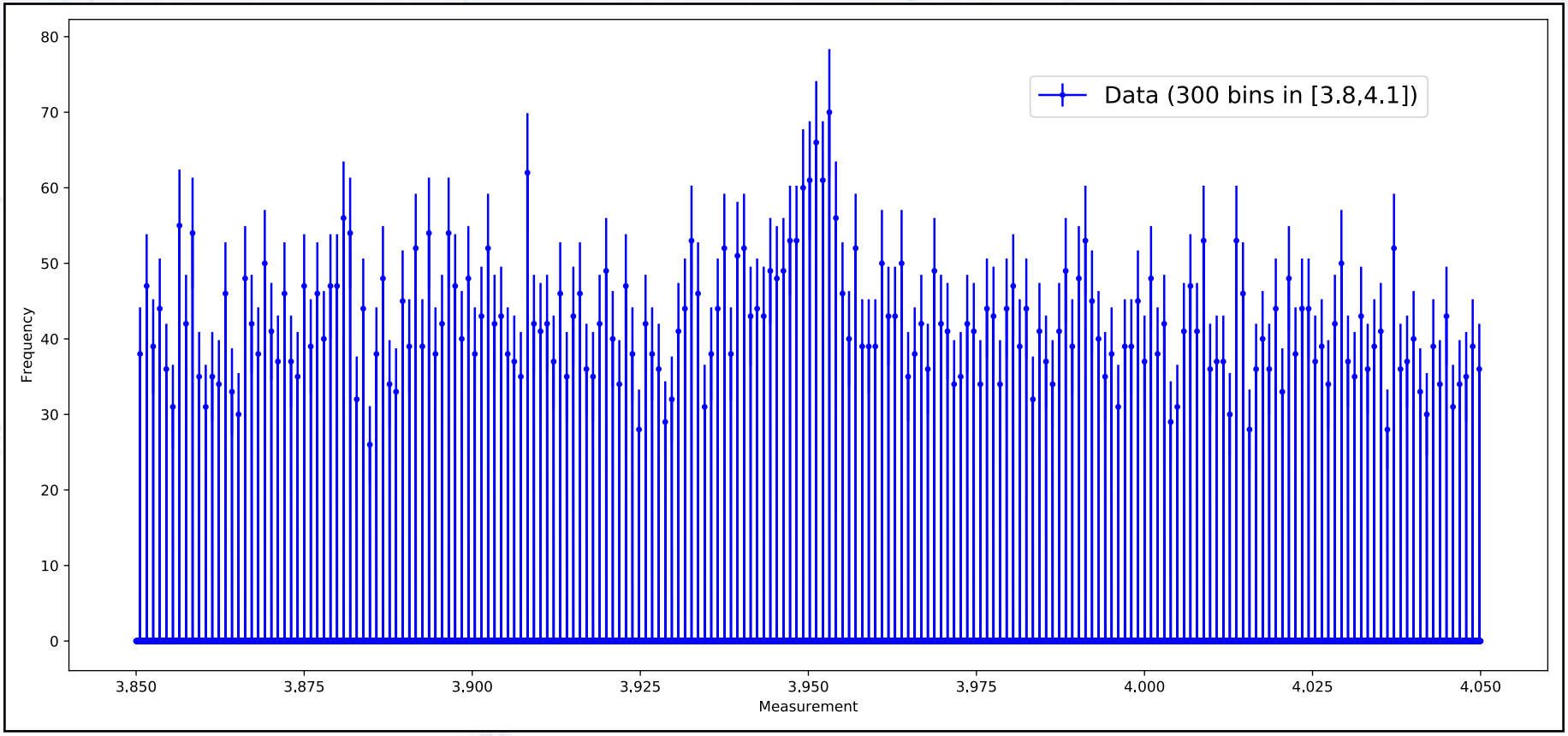
Histogram 3

When plotting a small range, one can see a small but significant Gaussian peak, but also six “spikes”. What is that?
Well, those a bins, where two values fall in one bin, resulting in a double value!



206? How could you know?

If you make a plot with a lot of bins - 20000 here - then you can see how many UNIQUE bins there are, and use this as a guideline.



Conclusions

When producing a histogram, you **have to consider the binning**, i.e. range and number of bins. Unfortunately, **matplotlib** does not force you to make any choices. Make sure you don't make that mistake…

…I hope that the above words now have a little more depth, and that you have binning in mind, whenever you make a histogram from now on.