Draft version October 3, 2022

AST326 Lab-01 Report
Basic Photon Statistics with Python

Raina Irons[1]

[1] *University of Toronto*

ABSTRACT

In this report we introduce some basic functions of Python and it's applications in data science. We discuss the differences of displaying data as scattered data versus using histograms to determine whether or not they follow a Poisson or Gaussian distribution. Two files of photon count rates with different integration times were used to test this purpose, with $\mu = 3.268$ and $\mu = 1137.97$ for the small and large integration times, and it was demonstrated for large μ the Poisson distribution follows a Gaussian. A Normal distribution was also fit to the histograms, fitting much better to the photon count rates with a larger integration time than the smaller. Finally, the Hubble constant was calculated for several galaxies with receding radial velocities and distances given. Creating a model in order to fit the linear data and find the error resulted in Hubble constant $66 kms^{-1}Mpc^{-1} \pm 1.27$ with intercept $632 Mpckms^{-1} \pm 843$.

1. INTRODUCTION

The Introduction in this lab detail some mathematical arguments and support for Section 2 and the process. We begin by noting the importance of examining the distribution by comparing the distribution to a histogram of a set of data.

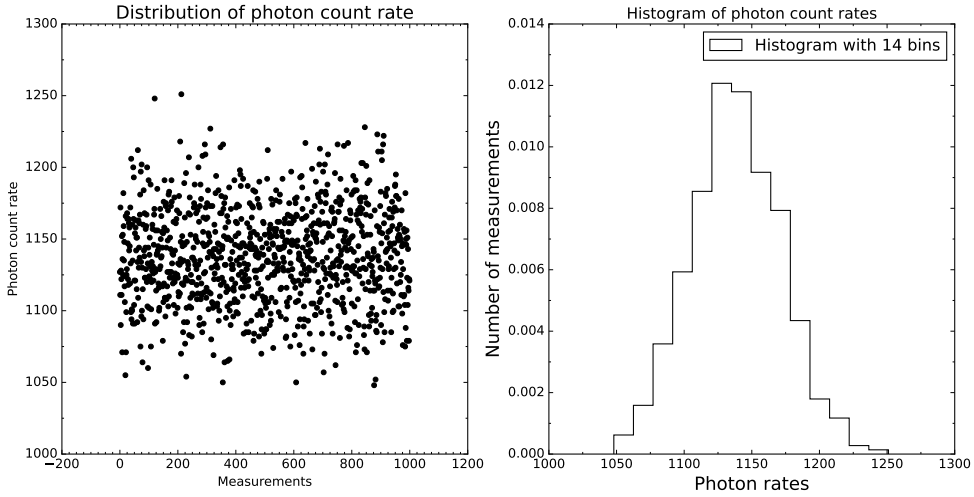

Figure 1: *Left*: The distribution of a data set, here we see it is not immediate the key features and distribution we are looking at. *Right*: The histogram of the same data set. The mean and shape of the data is much more clear. Note that this figure is an altered plot from Section 2.3 to demonstrate the motivation of displaying data in histogram format.

Two key distributions (and most common in the field of astronomy) is the Poisson,

$$P(x;\mu) = \frac{\mu^x}{x!}e^{-\mu}, \tag{1}$$

and the Gaussian (or Normal) distribution,

$$P(x;\mu;\sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp\Big[-\frac{1}{2}\Big(\frac{x-\mu}{\mu}\Big)^2\Big]. \tag{2}$$

Where μ is the mean and σ is the standard deviation (or spread) of the data. A property of this distribution is its discrete behaviour which measures the probability of x events occurring within a given time interval μ with a skewed symmetry about the mean. We will demonstrate in Section 2.3 how as the mean of the distribution increases, we approach a Normal distribution, which can be thought of as the continuous counterpart. The Poisson has the unique property

$$\sigma = \sqrt{\mu}, \tag{3}$$

Which allows it to be parameterized by only the mean of the distribution. In comparison, the Gaussian distribution is symmetric about its mean and has 6.3% of its probability within

$$\mu \pm \sigma, \tag{4}$$

with the standard deviation defined as

$$\sigma = \sqrt{\frac{\sum(x_i-\mu)^2}{N}}, \tag{5}$$

where N is the number of measurements.

In Section 2.5, we will be fitting a linear set of data and finding the errors for that fit. The equation of a line follows

$$y = mx + c. \tag{6}$$

In Section 2.5, we will be looking at bodies moving with velocities, and examining their speed (m) and initial position (c). We can solve for these two parameters through matrix multiplication,

$$\begin{pmatrix} m \\ c \end{pmatrix} = \begin{pmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & N \end{pmatrix}^{-1} \begin{pmatrix} \sum x_i y_i \\ \sum y_i \end{pmatrix}. \tag{7}$$

Which in Python are trivial to compute. The deviations from the fit are

$$\sigma^2 = \frac{1}{N-2}\sum_i [y_i - (mx_i + c)]^2, \tag{8}$$

with two unknowns for the line there are $N-2$ degrees of freedom. The error for the slope and intercept of the best fit are

$$\sigma_m^2 = \frac{N\sigma^2}{N\sum_i x_i^2 - (\sum_i x_i)^2}, \tag{9}$$

and

$$\sigma_c^2 = \frac{\sigma^2 \sum_i x_i^2}{N\sum_i x_i^2 - (\sum_i x_i)^2}, \tag{10}$$

respectively.

2. RESULTS AND METHODS

2.1. *Reproducing Fig.3 in the Lab Manual*

We begin by examining the distribution of photon count rates for a given set of data with $\mu = 12$. We plot the histogram of the count rates as in Figure 1 (recreating Figure 3 in the lab manual), and overlay the Poisson distribution overtop.

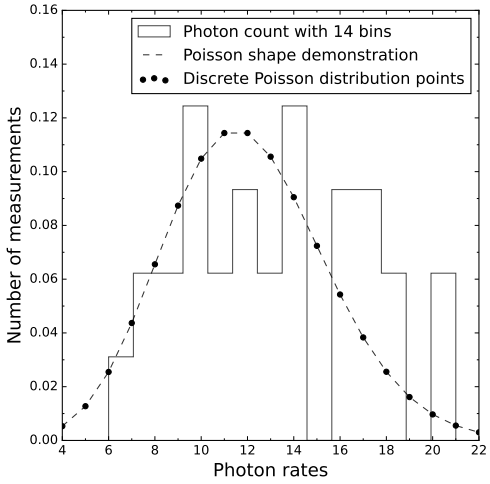

Figure 2: The photon count rates follow a Poisson as evident from the overlayed dashed lines indicating a Poisson distribution. Note the Poisson is a discrete distribution, and the dashed line is only there for demonstration of the shape, not to signify continuity. The histogram of the count rate is normalized.

Here it is clear from the peak of the distribution what the mean is. The Poisson was fit using the *scipy.stats.poisson.pmf*() function and generating random variables given $\mu = 12$.

2.2. *Mean and standard deviation of small and large data sets*

We introduce two new sets of data, *small_data* and *large_data*, which are 1000 measurements of photon count rates from a star, where *large_data* has a much longer integration time than *small_data*. We can find the mean and standard deviation of these count rates by utilizing the *Numpy* library, specifically the *numpy.mean*() and *numpy.std*() function. *numpy.std*() work much like Equation 5 in its implementation, while *numpy.mean*() is the sum of the elements along the axis divided by the number of elements, or

$$\mu = \frac{1}{N} \sum_i x_i. \tag{11}$$

Implementing these functions, we get the following results.

Data Set	Mean	Standard Deviation	Variance
Small Data Set	3.268	1.785	1.807
Large Data Set	1137.97	33.21	33.73

We can see the standard deviations of both sets of date are very close to their variance, implying that the Poisson distribution would be a good fit for both.

2.3. *Examining distributions of large and small data sets with Poisson*

The photon count rates can be displayed much like in Figure 1 to examine their distributions.

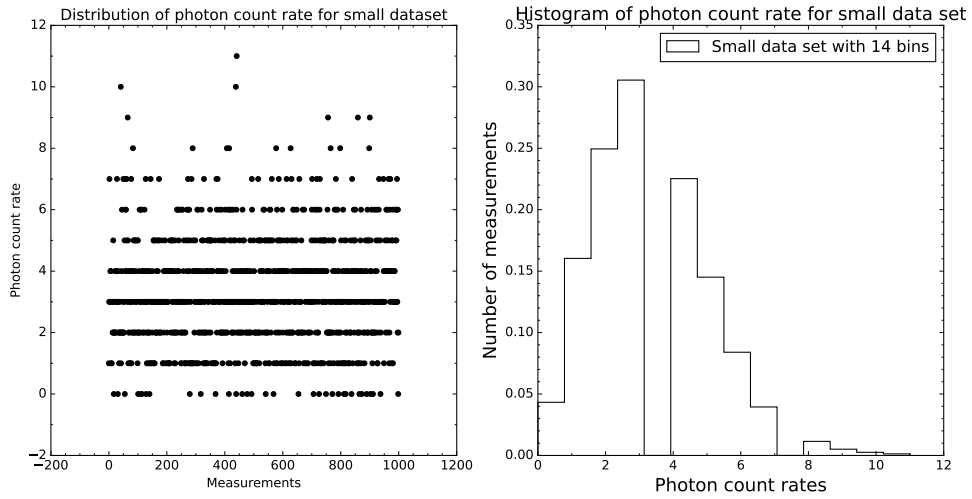

Figure 3: *Left*: The distribution of the photon count rates for the small data set do not have any immediate features we can pick out. *Right*: Displaying a histogram in order to see the distribution of the data is motivated with Figure 1 in Section 1. Here we can clearly see the mean of the data as the peak of the normalized histogram.

From Figure 3, we can estimate the mean from a glance and determine it must be around 3 as evident of the peak. We can repeat this process for the larger data set.

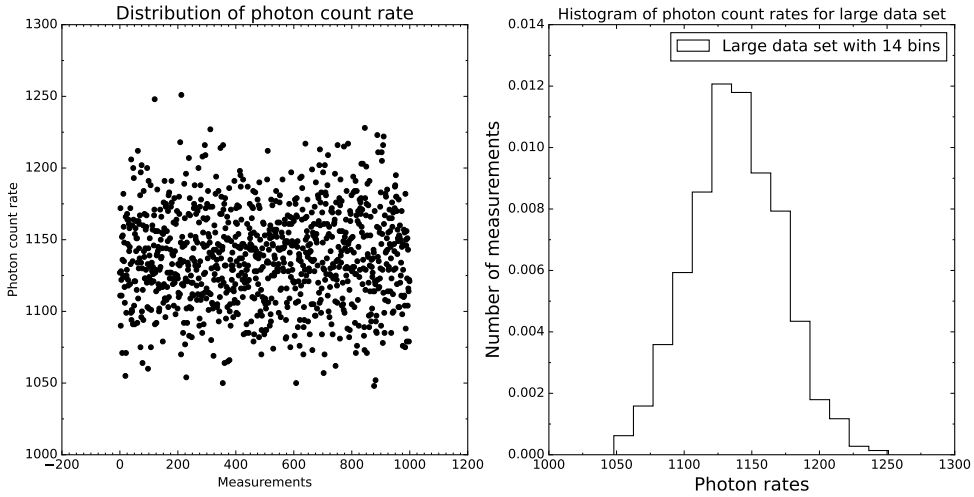

Figure 4: *Left*: The distribution of the large data set of photon count rates. *Right*: The histogram of the photon count rates supply more information than simply plotting the distribution. Note that this is the unaltered form of Figure 1 from the Introduction.

From Figure 4, we can once again estimate the mean at a glance, appearing to be somewhere around 1140. The small data set can be compared with a Poisson distribution much like in Figure 2 by overlaying the distribution. The calculated mean from Table 2.2 was used.

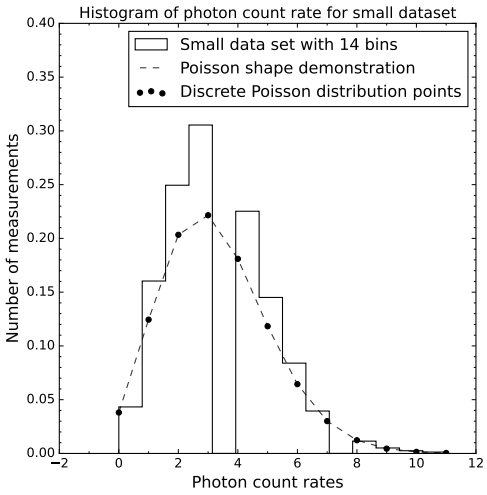

Figure 5: The Poisson distribution acts as a good fit for the small data file. Note that while Figure 2 had its Poisson distribution with extended domain past the histogram for more accurate fitting, we do not do that here as negative photon counts make no physical sense.

Figure 5 demonstrates just how well the Poisson distribution fits the data. In Section 4.3, we will explore how for large means the Poisson distribution approaches the Normal.

2.4. *Small and large data set with Normal distribution*

We can fit our distributions now with a Normal probability density function and compare how this differs from the Poisson.

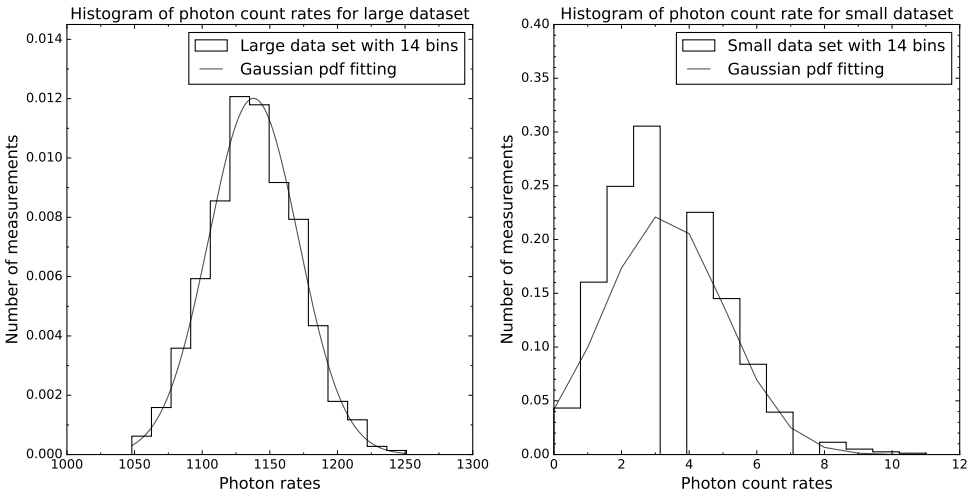

Figure 6: *Left*: The Normal fitting for the large photon count over the histogram from calculated mean and standard deviation from Table 2.2. It is clear here the Gaussian looks much more similar to the large data set than the small one. *Right*: The Gaussian appears to have a less accurate fit as opposed to the Poisson in Figure 5. The mean and standard deviation used were taken from Table 2.2.

2.5. *Calculating the Hubble constant*

We will introduce a new set of data containing the radial receding velocities and distances of several galaxies. Our goal for this section is to develop a linear fit to this data and determine errors of the fitting. Plotting the data is quite simple, and only requires separating out our distances and velocities into separate arrays. Recall from Equation 7 how the gradient and $y-$intercept can be calculated through matrix multiplication. Details on code and practical implementation can be found in Section 4.4. Additionally, the deviations from the slope can be found using Equation 8 to calculate the error, keeping in mind to take the square root at the end of the operation.

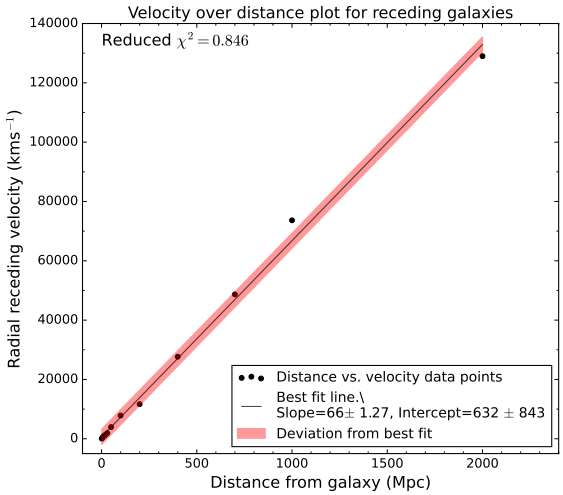

Figure 7: The velocity vs distance plot illustrates the calculation of the Hubble constant. By plotting velocity (kms^{-1}) against distance (Mpc), we get a s^{-1} unit for slope, which is the Hubble constant. The inverse of this is the age of the universe. The deviations from the slope are shown in the error bars on the data points, which directly correlate to the reduced chi-square. The data set for the large part remains inside the deviations from the fit, and any outliers play a role in straying the chi-square from 1.

From Figure 7, we found the slope and intercept of our best fit line to be $66kms^{-1}Mpc^{-1}\pm1.27$ and $632Mpckms^{-1}\pm843$ respectively. For details on the code used to implement the fitting and errors along with the chi-square, refer to Section 4.4.

3. DISCUSSIONS AND CONCLUSIONS

This report demonstrates many important functions in data analysis methods for astronomy. We begin in the Introduction by describing some mathematical background and support for the Results and Methods, Section 2, and motivate what we are doing. The majority is looking at distributions of data and seeing that in order to understand the distribution being analyzed, a histogram is an excellent option. This is demonstrated in Figure 3, where it is not immediate what is mean is in the distribution when we plot the data as is. Figure 2 shows the initial photon count rate distribution we look at, and the Poisson probability mass function overlayed. Here we begin to see the necessity of examining the distribution of our data. Using different data, two separate files with short and long integration time were read into the file and examined as in Figure 2. First, Table 2.2 summarizes the mean, standard deviation, and the variance of both distributions using basic *Numpy* functions that implement Equations 5 and 11. Using the calculated mean, we can fit a Poisson distribution to the smaller data set, visualized in Figure 2.

The second distribution we examined was the Normal, or Gaussian distribution. Defined in Equation 2, takes in both the mean and standard deviation of the photon count rates, and is visualized in Figure 6 overtop both the large and small data sets. This continuous distribution shows a much better fitting for the large integration time than the smaller, as opposed to fitting the Poisson to the smaller data set. An additional note is in Section 4.3, where we

demonstrate how for large means, the Poisson distribution approaches a Gaussian shape.

Lastly, we looked at the set of receding radial velocities and distances of several galaxies, and used Equation 7 to fit a linear best fit model to our data. The errors were calculated using Equation 8 -10, and implementation of these equations can be found in Section 4.4. From the fit, we were able to extract a line with slope $66 kms^{-1} Mpc^{-1} \pm 1.27$ with intercept $632 Mpckms^{-1} \pm 843$. The slope of our line is the Hubble constant, whose inverse is the age of the universe.

4. APPENDICES

4.1. *Reproducing Fig.3 additional comments and Python code*

The photon count rate used in Figure 2 can be represented as an array of data for the plotting.

```
photon_count_rate = np.array([13,17,18,14,11,8,21,18,9,12,9,17,14,6,10,16,16,11,10,12
    ,8,20,14,10,14,17,13,16,12,10])
N_measurements = len(photon_count_rate)
mean = 12
```

Where we also specify the length of the data for the domain and mean. To implement the Poisson distribution, we use *scipy.stats.poisson.pmf*() for the probability mass function of the poisson distribution. We chose the domain specific to the range of values in the photon count rate.

```
# call poisson function
x = np.arange(4,23,1)
P_dist = ss.poisson.pmf(x,mu=mean)
```

From here we plot P_{dist} against x and the histogram for the photon count rate and achieve Figure 2.

4.2. *Importing files into Python and code*

The *Numpy* library was utilized for reading in our small and large files. The data can be read in simply with

```
# for small data set
small_data = np.loadtxt('Irons-ironsrai-Small.txt')
small_mean = np.mean(small_data)
small_stddev = np.std(small_data)

large_data = np.loadtxt('Irons-ironsrai-Large.txt')
large_mean = np.mean(large_data)
large_stddev = np.std(large_data)
```

along with calculating the means and standard deviations as supplied in Table 2.2. Note that *np* is short for the *Numpy* library.

4.3. *Poisson distributions with large means*

In parallel with Figure 5, we can fit the Poisson distribution with calculated mean from Table 2.2. In addition, we can fit a Normal distribution and see that for large means, the Poisson and Gaussian are very similar. We implement the Gaussian as follows.

```
large_domain4 = np.arange(1048,1252,1)
Norm_largedata = ss.norm.pdf(large_domain4,loc=1137.97,scale=33.21468)
```

Where *large_domain*4 represents the domain of the large photon count data and *ss.norm.pdf*() is short for *scipy.stats.norm.pdf*(). This domain is slightly extended to receive a better visual fit with the histogram. Using this and previous methods for Poisson pmf fittings, we create the folowing plot.

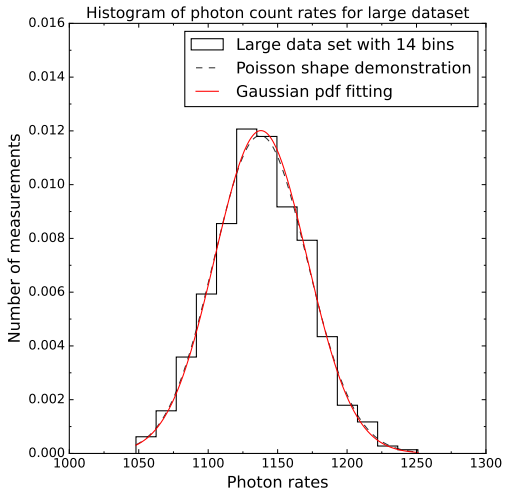

Figure 8: The dashed line here represents the shape of the Poisson distribution for $\mu = 1137.97$, with $\sigma = 33.21$. We see that it is very similar to the Normal distribution (solid red line), which demonstrates for large means, the two distributions are similar in nature.

4.4. *Fitting linear data and determining errors*

The fitting parameters for the line of best fit are written mathematically in Equation 7, in code, this is written as

```
# creating the arrays for curve fitting
N = len(distance)
M1 = [[np.sum(distance**2) , np.sum(distance)] , [np.sum(distance) , N]]
M1_inverse = np.linalg.inv(M1)
M2 = [[np.sum(distance*velocity)] , [np.sum(velocity)]]
[m,c] = np.dot(M1_inverse,M2)
print('The fitting parameters are',m,c)
```

which output fitting parameters 66.14711197 for the slope, and 632.69381533 for the $y-$intercept. The associated errors, Equations 9 and 10 are the method for calculating the errors.

```
m_std = (N*std_squared)/(N*np.sum(distance**2) - np.sum(distance)**2) # error for the
slope
m_std = np.sqrt(m_std)
c_std = (std_squared*np.sum(distance**2))/(N*np.sum(distance**2) - np.sum(distance)**2)
# error for the intercept
c_std = np.sqrt(c_std)
```

Which returned an error for the slope at 1.27343 and error for the $y-$int as 843.5326. The chi-square functions used to test the fitting of the best fit line with the data are defined as

```
# defining chi square function to test fits for question 6
def chi_square(y_measured, y_expected,errors):
    return np.sum( np.power((y_measured - y_expected),2) / np.power(errors,2) )
# define chi square reduced
def chi_square_reduced(y_measured,y_expected,errors,number_parameters):
    return chi_square(y_measured,y_expected,errors)/(len(y_measured - number_parameters))
```

Where the reduced chi-square calls on the chi-square function and corrects for the degrees of freedom for a line. In our case, we have 2 degrees of freedom for the slope and intercept. The following lines are the practical implementation of calling the functions.

```
# finding the chi square and chi square reduced for the fit
chisq = chi_square(velocity, m[0]*distance + c[0],std)
chisqreduced = chi_square_reduced(velocity,m[0]*distance + c[0],std,2)
```

Where we plotted our reduced chi-square on Figure 7.