Lab #1: Basic Photon Statistics with Python

Lab report is due at 4pm on Oct 3 (Monday). Submission on Quercus.

1. Overview and Goals

This document provides a description of the activities that we will explore in the 1st lab. Because we use the 1st lab to introduce many new skills, which include Python programming (potentially on Linux or JupyterHub platform) for data analysis and visualization as well as document writing using LaTex, this document is a straightforward guide that you can follow sequentially.

The primary goal of the 1st lab is to learn how to conduct simple statistical analysis and data visualization using Python programming language. (Check the calendar of the TA tutorials/office hours for tutorials on how to start Python programming.) **This simple lab will serve as a stepping-stone to more advanced analyses in the forthcoming labs.** *(For those students who already have Python programming experiences, this will be a very easy assignment. More challenging and advanced assignments will come later.)*

Schedule: The report due is 4pm on Oct 3 on Quercus. See §5 for detailed steps.

References (available online and on the course website as well as in TA tutorials)

- Python
  In the class web page on Quercus, you can find "*Introductory Document for Computing Resources*" by Programmer Supporter TA. The following web pages can be very useful to learning Python:
  https://docs.scipy.org/doc/scipy/getting_started.html
  https://numpy.org/ (on numerical computing in Python)
  https://matplotlib.org/ (on visualization in Python)
- LaTex document writing: You are required to write your lab reports using LaTex in this course. The Overleaf web page at https://www.overleaf.com/ provides the service that you can use free. Note that LaTex is a default editor for astronomy and many fields in sciences and engineering.
- Linux: For those who use Python on Linux (including JupyterHub), it is important to understand basic Linux commands.

2. Basic Statistical Parameters and Data Visualization

Let's assume that you have conducted a series of astronomical observations to estimate the distance to a nearby star. As below, you repeated the same observations 30 times and obtained the following results – the numbers are distances (in parsec) to the star obtained in each trial, whereas the numbers in the parentheses are the 1-σ measurement uncertainty (= uncertainty at the confidence level of 68.3% assuming that it follows a Gaussian distribution).

Distance Measurements:

38.91 (1.41)	37.14 (0.36)	38.19 (0.69)	41.03 (3.53)	34.86 (2.64)
37.33 (0.17)	35.16 (2.34)	37.96 (0.46)	36.93 (0.57)	40.41 (2.91)
29.50 (8.00)	37.33 (0.17)	41.84 (4.34)	37.53 (0.03)	34.12 (3.38)
34.11 (3.39)	37.94 (0.44)	34.43 (3.07)	36.68 (0.82)	41.31 (3.81)
39.61 (2.11)	35.48 (2.02)	34.98 (2.52)	39.05 (1.55)	39.62 (2.12)
37.96 (0.46)	39.02 (1.52)	37.47 (0.03)	33.76 (3.74)	36.51 (0.99)

The first thing you want to do with the data is to examine the distribution of the measurements. The left panel in Figure 1 is a plot comparing all the 30 measurements in sequence, while the right panel shows a histogram of the data. Do you see the difference? Which is better?

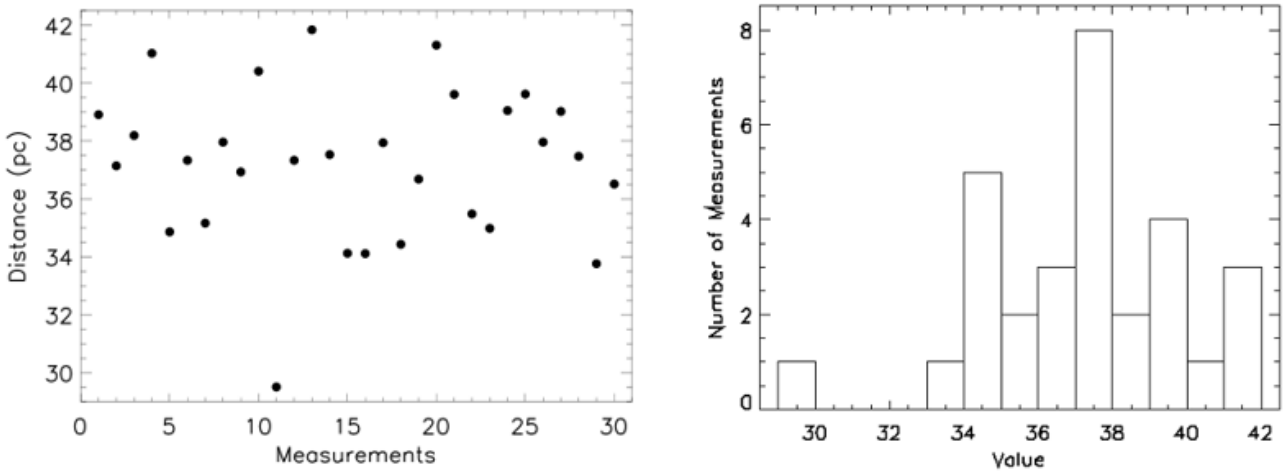

Figure 1. (Left) Distribution of the measured distances. (Right) Histogram of the measure distances.

The next step is to calculate a statistical parameter that represents the measured distance best. We often use mean for this and standard deviation (= STDDEV) for its 1-σ uncertainty. From the data above, I have the following mean and STDDEV:

Mean ± STDDEV = 37.21 ± 2.65 (pc)

using the following definition of STDDEV (= σ)

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

where x_i, μ, and N represent measured value, the mean, and the number of measurements, respectively.

However, each measurement has a different uncertainty, and the measurements with a smaller uncertainty should be more reliable than those with a large one. Using weighted mean statistics (in which the measurements with a smaller uncertainty are given a more weight), now I have the following (weighted) mean and STDDEV:

(Weighted) Mean ± STDDEV = 37.50 ± 0.02 (pc)

You can confirm that the weighted mean has a much smaller (or reliable) uncertainty.

3. Poisson Distribution and Gaussian Distribution

Often astronomical signals (e.g., photons detected with an astronomical detector based on photoelectric effect) follow the Poisson probability distribution as below

$$P(x;\mu) = \frac{\mu^x}{x!}e^{-\mu} \quad \text{(Poisson distribution)}$$

where *P(x;μ)* is the probability *x* events will be measured within an interval given a mean of μ events. Note that a Poisson distribution is a discrete (as opposed to continuous) distribution and cannot take a negative mean value. As seen in Figure 2 below, the Poisson distribution is an asymmetric distribution when the mean value is small, but it becomes more symmetric for larger mean values.

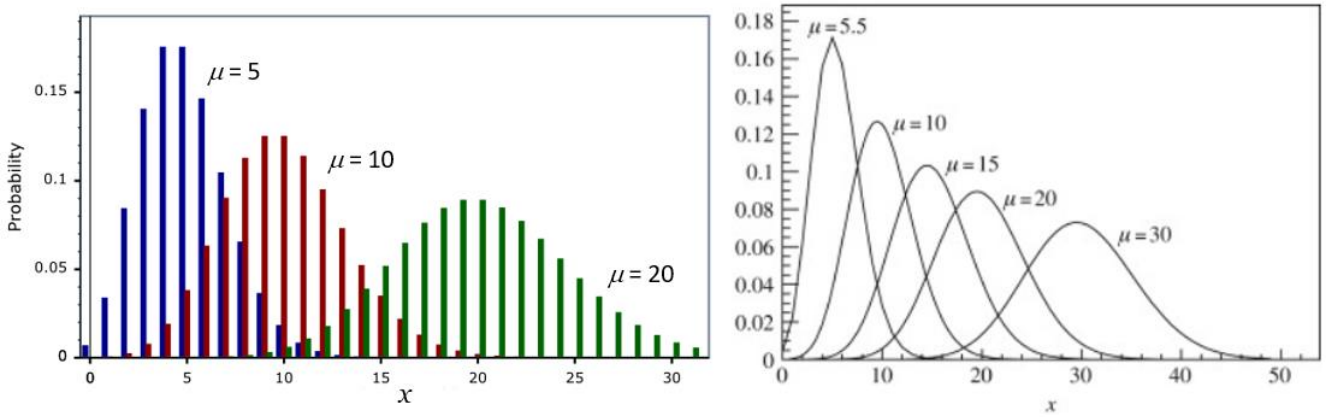

Figure 2. Poisson probability distribution for different mean values.

The Poisson distribution has the following unique property

$$\text{STDDEV} = \sqrt{(\text{Mean})} \text{ (for Poisson distribution)}$$

in which standard deviation is determined by mean value.

Now let's assume that you measured the number of incoming photons from an astronomical source per second 30 consecutive times as below. The expected mean value is 12 photons per second.

Measured Photon Count Rates (= Number of incoming photons per second)

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

First, let's calculate the mean and standard deviation as follows:

$$\text{Mean} \pm \text{STDDEV} = 13.2 \pm 3.8 \text{ (counts/second)},$$

and, since $\sqrt{13.2} \approx 3.6$, you can see that the above relation of $\text{STDDEV} = \sqrt{(\text{Mean})}$ for a Poisson distribution (approximately) applies in these measurements.

Figure 3 below compares the histogram (solid line) of the 30 measured photon count rates and the expected Poisson distribution (dashed line) with $\mu = 12$ calculated using the Poisson distribution formula above. Do you think the data follow the Poisson distribution of $\mu = 12$? (Note that the dashed line in the figure was calculated by normalizing probabilities from a Poission distribution of $\mu = 12$ and then multiplying the number of measurements to be compared with the measurement histogram.)

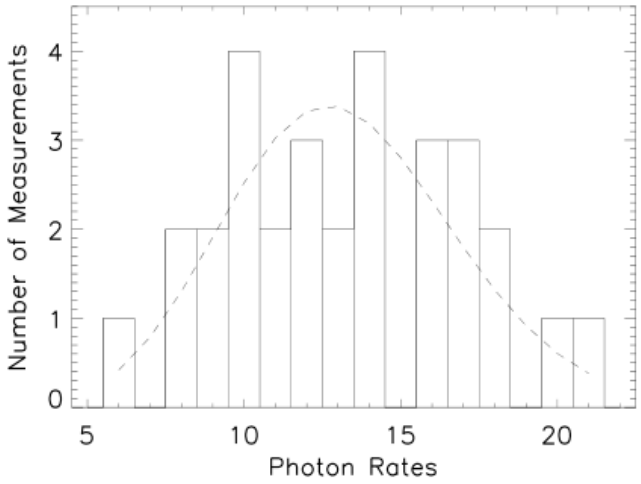

Figure 3. (Solid line) Histogram of the photon count rates; (Dashed line) Expected Poisson distribution with μ = 12.

In comparison, the Gaussian probability distribution, which is often called "normal distribution" and we are more familiar with, takes the following form

$$P(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

and it is a continuous distribution whose mean can be a negative value. As in Figure 4 below, the Gaussian distribution is a symmetric distribution wherein 68.3 % of the entire probability lies within (Mean) ± (STDDEV). Random variables (e.g., random noise) follow the Gaussian probability distribution. In contrast to the Poisson distribution, the standard deviation cannot be determined by the mean value in a Gaussian distribution.

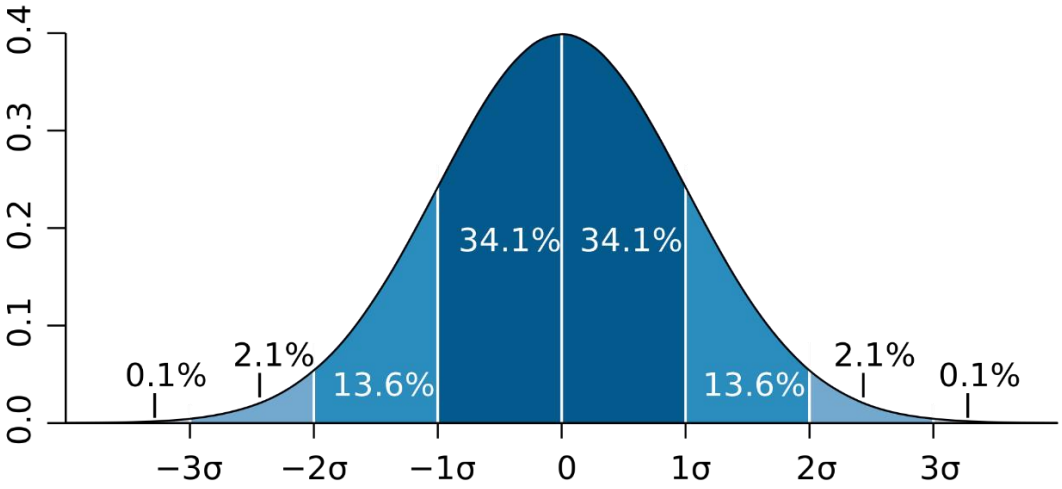

Figure 4. Gaussian probability distribution. (Wikipedia)

Note that for a large mean value, a Poisson distribution becomes very similar to a Gaussian distribution (see Figure 2).

4. Linear Least Squares Fitting

One of the primary skills we need to learn is "linear least squares fitting". Often observations and experimental measurements are undetermined, which means that they are limited by the number of observations or sampling to calculate an undetermined parameter space. To correct for this, we use an equation to model a set of data and compare the difference between the observed values to the fitted values from the model. This difference is referred to as residuals. The term "least-squares" refers to minimizing the *square of the residuals* to determine the best-fit model to the observed data set.

In this lab, we will first focus on linear-least squares where the model is a straight line, but we will generalize the least squares method to other non-linear functions. It is important that in this lab you do not use a canned least-squares routine and you write your own least-square routine.

4.1 Fitting a Straight Line

Suppose that we have a set of N observations (x_i, y_i) where we believe that the measured value, y, depends linearly on x, i.e.,

$$y = mx + c.$$

For example, suppose a body is moving with constant velocity, what is the speed (m) and initial (c) position of the object?

Given our data, what is the best estimate of m and c? Assume that the independent variable, x_i, is known exactly, and the dependent variable, y_i, is drawn from a Gaussian probability distribution function with constant standard deviation σ_i = const. Under these circumstances the most likely values of m and c are those corresponding to the straight line with the total minimum square deviation, i.e., the quantity

$$\chi^2 = \sum_i \left[y_i - (mx_i + c) \right]^2$$

is minimized when m and c have their most likely values. Figure 5 shows a typical deviation.

The best values of m and c are found by solving the simultaneous equations,

$$\frac{\partial}{\partial m}\chi^2 = 0, \quad \frac{\partial}{\partial c}\chi^2 = 0$$

Evaluating the derivatives yields

$$\frac{\partial}{\partial m}\chi^2 = \frac{\partial}{\partial m}\sum_i \left[y_i - (mx_i + c) \right]^2 = 2m\sum_i x_i^2 + 2c\sum_i x_i - 2\sum_i x_i y_i = 0$$

$$\frac{\partial}{\partial c}\chi^2 = \frac{\partial}{\partial c}\sum_i \left[y_i - (mx_i + c) \right]^2 = 2m\sum_i x_i + 2cN - 2\sum_i y_i = 0.$$

This set of equations can conveniently be expressed compactly in matrix form,

$$\begin{pmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & N \end{pmatrix} \begin{pmatrix} m \\ c \end{pmatrix} = \begin{pmatrix} \sum x_i y_i \\ \sum y_i \end{pmatrix}$$

and then solved by multiplying both sides by the inverse,

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

The inverse can be computed analytically, or in Python it is trivial to compute the inverse numerically, as follows.

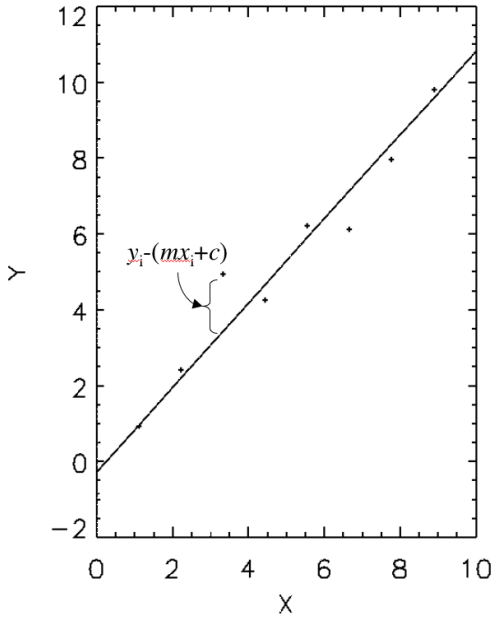

Figure 5: Example data with a least squares fit to a straight line. A typical deviation from the straight line is illustrated.

4.2 Example Python Script

```
# Test least squares fitting by simulating some data.
import numpy as np
import matplotlib.pyplot as plt

nx = 20          # Number of data points
m = 1.0          # Gradient
c = 0.0          # Intercept

x = np.arange(nx,dtype=float)        # Independent variable
y = m * x + c                                # dependent variable

# Generate Gaussian errors
sigma = 1.0                                  # Measurement error
np.random.seed(1)                                    # init random no. generator
errors = sigma*np.random.randn(nx)  # Gaussian distributed errors
```

```
ye = y + errors                 # Add the noise

plt.plot(x,ye,'o',label='data')
plt.xlabel('x')
plt.ylabel('y')

# Construct the matrices
ma = np.array([ [np.sum(x**2), np.sum(x)],[np.sum(x), nx ] ]  )
mc = np.array([ [np.sum(x*ye)],[np.sum(ye)]])

# Compute the gradient and intercept
mai = np.linalg.inv(ma)
print 'Test matrix inversion gives identity',np.dot(mai,ma)
md = np.dot(mai,mc)     # matrix multiply is dot

# Overplot the best fit
mfit = md[0,0]
cfit = md[1,0]
plt.plot(x, mfit*x + cfit)
plt.axis('scaled')
plt.text(5,15,'m = {:.3f}\nc = {:.3f}'.format(mfit,cfit))
plt.savefig('lsq1.png')
```

See the figure below for the output of this program.

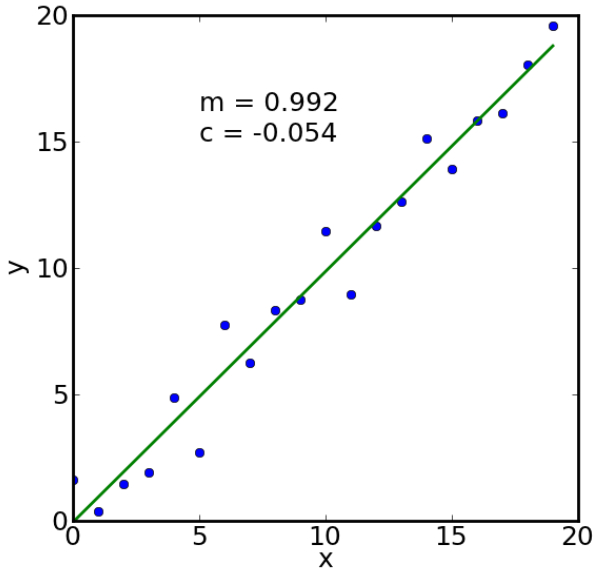

Figure 5. Least squares straight line fit. The true values are *m* = 1 and *c* = 0.

4.3 Error Propagation

What are the uncertainties in the slope and the intercept? To begin the process of error propagation we need the inverse matrix

$$\begin{pmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & N \end{pmatrix}^{-1} = \begin{pmatrix} N\Big/\left[N\sum x_i^2 - \left(\sum x_i\right)^2\right] & \sum x_i\Big/\left[N\sum x_i^2 - \left(\sum x_i\right)^2\right] \\ \sum x_i\Big/\left[\left(\sum x_i\right)^2 - N\sum x_i^2\right] & \sum x_i\Big/\left[N\sum x_i^2 - \left(\sum x_i\right)^2\right] \end{pmatrix},$$

so that we can compute analytic expressions for *m* and *c*,

$$\begin{pmatrix} m \\ c \end{pmatrix} = \begin{pmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & N \end{pmatrix}^{-1} \begin{pmatrix} \sum x_i y_i \\ \sum y_i \end{pmatrix} = \begin{pmatrix} \dfrac{\sum x_i \sum y_i - N\sum x_i y_i}{\left(\sum x_i\right)^2 - N\sum x_i^2} \\ \dfrac{\sum x_i \sum x_i y_i - \sum x_i^2 \sum y_i}{\left(\sum x_i\right)^2 - N\sum x_i^2} \end{pmatrix}.$$

The analysis of error propagation shows that if $z = z(x_1, x_2, .. x_N)$ and the individual measurements x_i are uncorrelated (they have zero covariance) then the standard deviation of the quantity *z* is

$$\sigma_z^2 = \sum_i \left(\partial z/\partial x_i\right)^2 \sigma_i^2 .$$

If the data points were correlated, then we would have a covariance matrix. The diagonal elements of this matrix are the standard deviations σ_{ii}^2 and of the off-diagonal elements $\sigma_{ij}^2 = \langle x_i x_j \rangle - \langle x_i \rangle \langle x_j \rangle$.

Thus, ignoring any possible covariance

$$\sigma_m^2 = \sum_j \left(\partial m/\partial y_j\right)^2 \sigma_j^2 \quad \text{and} \quad \sigma_c^2 = \sum_j \left(\partial c/\partial y_j\right)^2 \sigma_j^2 .$$

The expression for the derivative of the gradient, *m*, is

$$\frac{\partial m}{\partial y_j} = \frac{\partial}{\partial y_j}\left(\frac{\sum x_i \sum y_i - N\sum x_i y_i}{\left(\sum x_i\right)^2 - N\sum x_i^2}\right) = \frac{\sum x_i - N x_j}{\left(\sum x_i\right)^2 - N\sum x_i^2}$$

because $\left(\partial y_i/\partial y_j\right) = \delta_{ij}$, where δ is the Kroneker. If we assume that the measurement error is the same for each measurement, then

$$\sigma_m^2 = \sigma^2 \sum_j \left(\frac{\sum x_i - N x_j}{\left(\sum x_i\right)^2 - N \sum x_i^2} \right)^2$$

$$= \frac{\sigma^2}{\left[\left(\sum x_i\right)^2 - N \sum x_i^2\right]^2} \sum_j \left[\left(\sum x_i\right)^2 - 2N x_j \sum x_i + N^2 x_j^2 \right]$$

$$= \frac{\sigma^2}{\left[\left(\sum x_i\right)^2 - N \sum x_i^2\right]^2} \left[N\left(\sum x_i\right)^2 - 2N\left(\sum x_i\right)^2 + N^2 \sum x_i^2 \right]$$

$$= \frac{N\sigma^2}{N \sum x_i^2 - \left(\sum x_i\right)^2}$$

Similarly, for the intercept, *c*,

$$\frac{\partial c}{\partial y_j} = \frac{\partial}{\partial y_j} \left(\frac{\sum x_i \sum x_i y_i - \sum x_i^2 \sum y_i}{\left(\sum x_i\right)^2 - N \sum x_i^2} \right) = \frac{x_j \sum x_i - \sum x_i^2}{\left(\sum x_i\right)^2 - N \sum x_i^2}$$

and hence

$$\sigma_c^2 = \sigma^2 \sum_j \left(\frac{x_j \sum x_i - \sum x_i^2}{\left(\sum x_i\right)^2 - N \sum x_i^2} \right)^2$$

$$= \frac{\sigma^2}{\left[\left(\sum x_i\right)^2 - N \sum x_i^2\right]^2} \sum_j \left[x_j^2 \left(\sum x_i\right)^2 - 2 x_j \sum x_i \sum x_i^2 + \left(\sum x_i^2\right)^2 \right]$$

$$= \frac{\sigma^2}{\left[\left(\sum x_i\right)^2 - N \sum x_i^2\right]^2} \left[\sum x_i^2 \left(\sum x_i\right)^2 - 2\left(\sum x_i\right)^2 \sum x_i^2 + N\left(\sum x_i^2\right)^2 \right]$$

$$== \frac{\sigma^2 \sum x_i^2}{N \sum x_i^2 - \left(\sum x_i\right)^2}.$$

If we do not know standard deviation σ a priori, the best estimate is derived from the deviations from the fit, i.e.,

$$\sigma^2 = \frac{1}{N-2} \sum_i \left[y_i - (m x_i + c) \right]^2.$$

Previously, when we compute the standard deviation, the mean is unknown and we have to estimate it from the data themselves; hence, the Bessel correction factor of 1/(*N*-1), because there are *N*-1 degrees of freedom. In the case of the straight line fit there are two unknowns and there are *N*-2 degrees of freedom.

5. Lab #1 Assignment

Below is the list of steps that you need to follow for Lab #1 assignment. (Before you jump to them, however, I recommend you repeat calculations and plots above using your own Python codes for your own exercise.) *Using basic built-in functions in Python (e.g., numpy.mean, numpy.std, etc) is allowed it not specified otherwise.*

1. **[10 pts]** Reproduce Figure 3 using your own Python code. Your plot does not have to be exactly the same as Figure 3 in every details but must provide a histogram compared with an expected Poisson distribution.

2. In the following website,

   https://drive.google.com/drive/folders/1OxiihoBAsvf05nLN-ZRv6D9RFJZFkqKn?usp=sharing

   you can download two files that are assigned to you. (The file names have your last name and Login ID.) One file is named "*Small.txt", while the other is "*Large.txt." Each file has 1000 measurements of photon count rates from a star. The difference between "Small" and "Large" is the integration time wherein the latter has a much longer integration time than the former, so larger count rates in "Large" files than "Small." *(If there is no file for you, you need to use the file named "Noname-noname*.")*

3. **[10 pts]** Read the two files in your Python code and calculate the mean and standard deviation of the measurements recorded in each file (so you will have values for "Small" and "Large" files respectively). Are they consistent with what you expect from a Poisson distribution?

4. **[15 pts]** Plot the measurements to examine their distributions. First, you can simply plot the measurements in sequence and then using a histogram (like Figure 1 above). These analyses and plots need to be done separately for "Small" and "Large" measurements. By glancing the plots, can you roughly estimate the mean and standard deviation of the two measurements? Compare the histogram that you created for the "Small" data with what is expected from a Poisson distribution as Figure 3 above.

5. **[15 pts]** We learn in the class when the expected average is large, a Poisson distribution becomes very similar to a Gaussian distribution. Let's confirm this. First, between the histograms that you created for the "Small" and "Large" measurements, do you think the "Large" measurement looks like a Gaussian distribution more than the "Small" measurement (or vice versa)? Next, let's overplot the Gaussian distribution expected from the mean and standard deviation that you calculated in Step 3 on the histograms that you created in Step 4. Does the Gaussian distribution for the "Large" measurements look more similar to the data than the "Small" measurements?

6. **[30 pts]** Calculation of the Hubble constant using the straight-line linear least squares fitting. In the folder linked below

   https://drive.google.com/drive/folders/1khjxHLR_PM0RaWpy-JfNtXQCuRI0Zlh9?usp=sharing

you can find a file named as "[Your Last Name]-[Your Login ID]-Hubble.txt." (If there is no file for you, you need to use "Noname-noname-Hubble.txt.") The file has two columns: the first column is a list of the distances (in unit of mega-parsec [Mpc]) of 13 galaxies in the range of 1–2000 Mpc; the second column gives the observed receding velocities of the galaxies in unit of km s^{-1}. Using the data in the file, calculate the Hubble constant with its uncertainty by applying the linear least squares fitting method. **You need to use your own codes for this assignment; in other words, you are NOT allowed to use a built-in Python library for the fitting.**

7. **[20 pts for the quality of your report]** Prepare your report in LaTex and submit your report in PDF format. The Python codes that you developed for the analyses and plots need to be attached to your report in the same PDF file. Clearly explain your results and show plots in your report. You also need to specify the names of the data files that you use in your report. This is a practical assignment which always takes longer than you expect. So, **start early!**