CP Activity 1: Plotting and Basic Data Analysis

PHY2004W **KDSMIL001**

17 Feb 2020

Contents

1 Introduction

This Computational Activity was created in order to reintroduce us to Python and to give us a first look at Python modules such as numpy, scipy, and matplotlib. To perform the tasks set for us by the activity we used a laptop which had Python 3.7 installed, as well as the numpy, scipy, and matplotlib modules.

2 Process

The first task set to us (CP1a) was to use numpy and scipy to calculate the mean value and the variance of a set of data points given to us in the document Activity1Data.txt. We expected to see the same results for each method of computing the values. Below is the code used to compute those values [Figure 5] as well as the results from each method [Figure 1].

```
Numpy Mean: 40.08308
Numpy Variance: 3.62912
Scipy Mean: 40.08308
Scipy Variance: 3.69063
```

Figure 1: CP1a Results

As can be seen in Figure 2, we saw a difference of about 0.06 between the values of variance computed by numpy and scipy. After looking through the documentation of each module we found that the var() function in the numpy module uses a biased formula to find the variance, whereas the stats.describe() function from scipy uses an unbiased formula to calculate variance. Note that if we had used the var() function from scipy, we would have observed the same biased result as with the numpy var() function.

The next part of the activity (CP1b) was an introduction to the module matplotlib, which is used to produce publication quality plots. We were given a set of points with error bounds for each and were asked to reproduce (exactly!) a plot provided to us. Below is the code we used to produce the plot [Figure 6] as well as the plot itself [Figure 2]. The final question of the activity asked us to plot the line of best fit for this plot, which we have included in Figure 2.

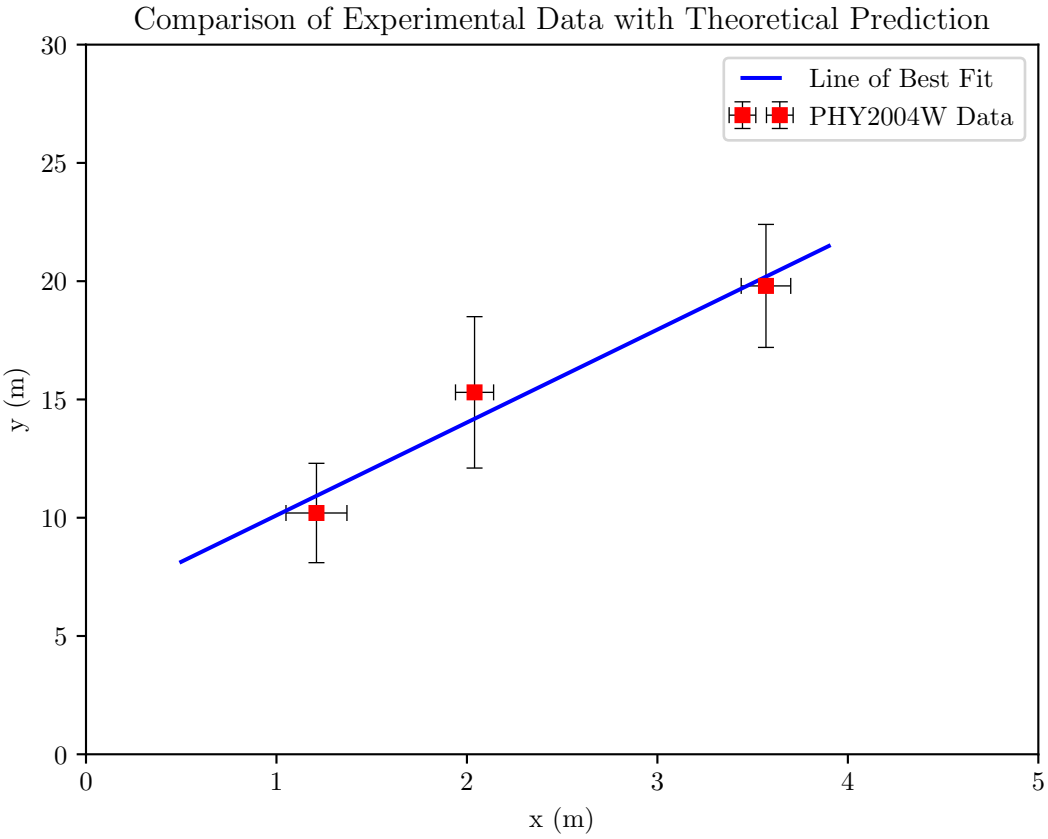

Figure 2: CP1b Plot

The third part of the activity (CP1c) was to write a program that would calculate m and c, as well as the uncertainties for both, from a set of data provided in the document LinearNoErrors.txt. The code for this section can be found in Figure 7 & 8 and the results are below [Figure 3]. The formulas used to find these values were provided in the activity sheet.

```
m: 0.59024
u(m): 0.02642
c: 1.07091
u(c): 0.19447
```

Figure 3: CP1c Results

The final part of the activity (CP1d) was split into three parts. Firstly we were asked to write another program to calculate the mean and variance of the original set of data, ACtivity1Data.txt, using the given equations for each rather than relying on the modules to do all the work. We decided to calculate both a biased and unbiased variance in order to compare them to the values obtained in CP1a. What we saw was exactly what we expected; the numpy variance was exactly what we obtained using the biased variance formula, and the scipy variance was exactly what we had obtained using the unbiased variance formula. These results can be seen below [Figure 4] and the code for this program can be found in Figure 9. Do note that all values obtained from these programs have been rounded to 5 decimal places.

```
Mean: 40.08308
Biased Variance: 3.62912
Unbiased Variance: 3.69063
```

Figure 4: CP1d Part 1 Results

Secondly, we were asked to comment on the measurement of x. The data provided to us was an array of readings of x. What we did to the readings, calculating the mean and the variance, provided us with measurement for x, namely 40.08 ± 3.69 if we use the biased variance, which we should use as this is only a sample of the data, not the entire population. This variance acts as the uncertainty of our measurement. In other words, the variance is our confidence interval as it shows the interval around our measurement that the actual value could lie it, given the readings that we have to consider.

The final part of the activity was to plot the line of best fit for the data we used in CP1b, which can be seen in Figure 2.

The code for all of these questions can be found in the Appendix on the next page.

3 Appendix

```python
import numpy as np
import scipy.stats as stats

#creates a numpy array to hold the data
data = np.zeros(60)
i = 0

#reads the data and puts it into the array
f = open('PHY2004W Computational\CP1\Activity1Data.txt', 'r')
header = f.readline()
for line in f:
    line = line.strip()
    columns = line.split()
    data[i] = float(columns[1])
    # print(i, data[i])
    i += 1

f.close()

#computing the mean and variance using numpy
npMean = np.mean(data)
npVariance = np.var(data)

#computing using scipy
n, (xmin, xmax), spMean, spVariance, s, k = stats.describe(data)

#printing out the results
print("Numpy Mean:", round(npMean, 5))
print("Numpy Variance:", round(npVariance, 5))
print("Scipy Mean:", round(spMean, 5))
print("Scipy Variance:", round(spVariance, 5))
```

Figure 5: CP1a Code

```python
from matplotlib import pyplot as plt
import numpy

x = [1.21, 2.04, 3.57]
xUn = [0.16, 0.10, 0.13]
y = [10.2, 15.3, 19.8]
yUn = [2.1, 3.2, 2.6]

N = 3

plt.errorbar(x, y, yUn, xUn, 'rs', 'black', 0.5, capsize=3, label="PHY2004W Data", \
    markersize=5, capthick=0.5)

plt.xlabel("x (m)")
plt.ylabel("y (m)")
plt.title("Comparison of Experimental Data with Theoretical Prediction")

plt.xlim(0,5)
plt.ylim(0,30)

xy = []
for c in range(N):
    xy.append(round(x[c]*y[c], 3))

x2 = []
for t in range(N):
    x2.append(round(x[t]**2, 3))

m = ((N*sum(xy)) - sum(x)*sum(y))/((N*sum(x2))-(sum(x))**2)
c = ((sum(x2)*sum(y))-(sum(xy)*sum(x)))/((N*sum(x2))-(sum(x)**2))

xLine = numpy.arange(0.5, 4, 0.1)
yLine = []
for i in xLine:
    yLine.append((m*i)+c)

plt.plot(xLine, yLine, color='blue', label="Line of Best Fit")

plt.legend(numpoints=2)
plt.savefig('CP1b Plot.pdf')
```

Figure 6: CP1b Code

```python
import numpy as np
import scipy.stats as stats
from math import sqrt

f = open('PHY2004W Computational\CP1\LinearNoErrors.txt', 'r')
header = f.readline()
N = 12
data = np.zeros([2, N])
i = 0

for line in f:
    data[0, i] = line.split()[0]
    data[1, i] = line.split()[1]
    i += 1

xy = []
for c in range(N):
    xy.append(round(data[0, c]*data[1,c], 3))

x2 = []
for t in range(N):
    x2.append(round(data[0,t]**2, 3))
```

Figure 7: CP1c Code 1

```python
x = data[0]
y = data[1]
d = []
d2 = []

m = ((N*sum(xy)) - sum(x)*sum(y))/((N*sum(x2))-(sum(x))**2)
c = ((sum(x2)*sum(y))-(sum(xy)*sum(x)))/((N*sum(x2))-(sum(x)**2))

for r in range(N):
    d.append(y[r] - ((m*x[r]) + c))
    d2.append(d[r]**2)

um = sqrt(((sum(d2)/((N*sum(x2))-(sum(x)**2)))*(N/(N-2))))
uc = sqrt((((sum(d2)*sum(x2))/(N*((N*sum(x2))-(sum(x)**2))))*(N/(N-2))))

print("m:", m)
print("um:", um)
print("c:", c)
print("uc:", uc)

f.close()
```

Figure 8: CP1c Code 2

```python
import numpy as np
import scipy.stats as stats

#creates a numpy array to hold the data
N = 60
data = np.zeros(N)
i = 0

#reads the data and puts it into the array
f = open('PHY2004W Computational\CP1\Activity1Data.txt', 'r')
header = f.readline()
for line in f:
    line = line.strip()
    columns = line.split()
    data[i] = float(columns[1])
    i += 1

f.close()

#calculating the mean and variance from their definitions
mean = sum(data)/N

data2 = np.zeros(N)
for i in range(N):
    data2[i] = (data[i]-mean)**2

varianceB = sum(data2)/(N)
varianceU = sum(data2)/(N-1)

print("Mean:", round(mean, 5))
print("Biased Variance:", round(varianceB, 5))
print("Unbiased Variance:", round(varianceU, 5))
```

Figure 9: CP1d Part 1 Code