Victor_Cruz_Ramos-Homework04

March 29, 2021

Victor Cruz Ramos
Section 101

1 Homework 4: File I/O, Statistics

Submit this notebook to bCourses to receive a grade for this Homework.

Please complete homework activities in code cells in this iPython notebook. Be sure to comment your code well so that anyone who reads it can follow it and use it. Enter your name in the cell at the top of the notebook. When you are ready to submit it, you should download it as a python notebook (click "File", "Download as", "Notebook (.ipynb)") and upload it on bCourses under the Assignments tab. Please upload the PDF file also.

For questions that ask you to make an interpretation, please write a sentence or two explaining your response. You can use "Markdown" language to create a cell (like this one) which is ordinary text instead of using code. To do this, create a new cell, and then look at the bar above which has a picture of a floppy disk. On the right side is a drop down menu that allows you change whether a cell is "Markdown" or "Code".

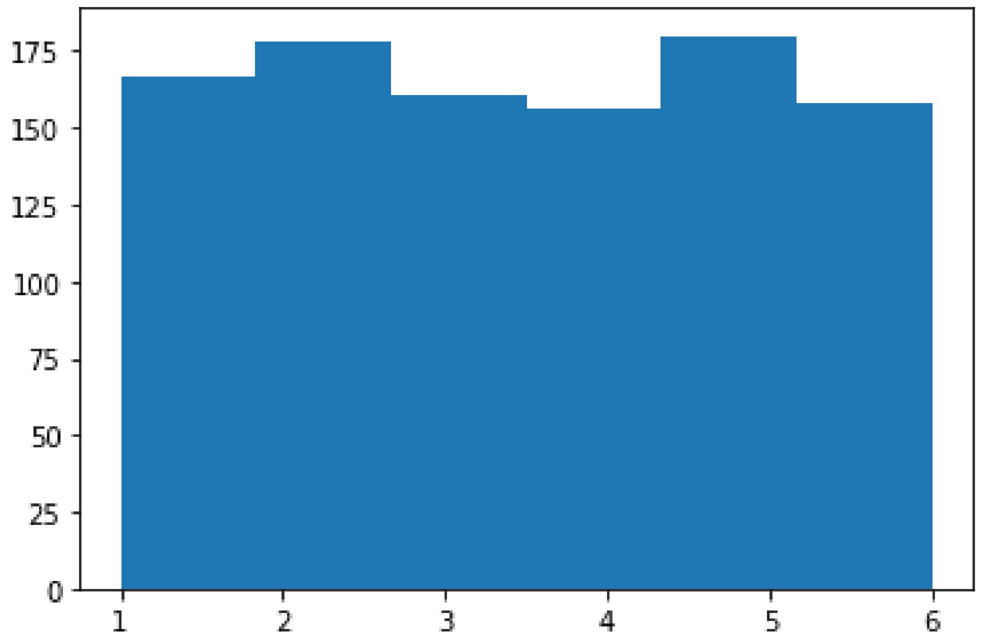
1.1 Problem 1: Central Limit Theorem

Here we will verify the Central Limit Theorem and reproduced a plot I showed in class (https://en.wikipedia.org/wiki/Central_limit_theorem#/media/File:Dice_sum_central_limit_theorem.svg)

1. Write a function that returns n integer random numbers, uniformly disributed between 1 and 6, inclusively. This represents n throws of a fair 6-sided die. The value that comes up at each throw will be called the "score".
2. Generate a distribution of 1000 dice throws and plot it as a histogram normalized to unit area. Compute the mean μ_1 and standard deviation σ_1 of this distribution. Compare your numerical result to the analytical calculation.
3. Generate 1000 sets of throws of $N = 2, 3, 4, 5, 10$ dice, computing the total sum of dice scores for each set. For each value of N, plot the distribution of total scores, and compute the mean μ_N and standard deviation σ_N of each distribution. This should be similar to the plot at the link above.
4. Plot the standard deviation σ_N as a function of N. Does it follow the Central Limit Theorem?

[63]:
```python
import numpy as np
import random
import matplotlib.pyplot as plt
```

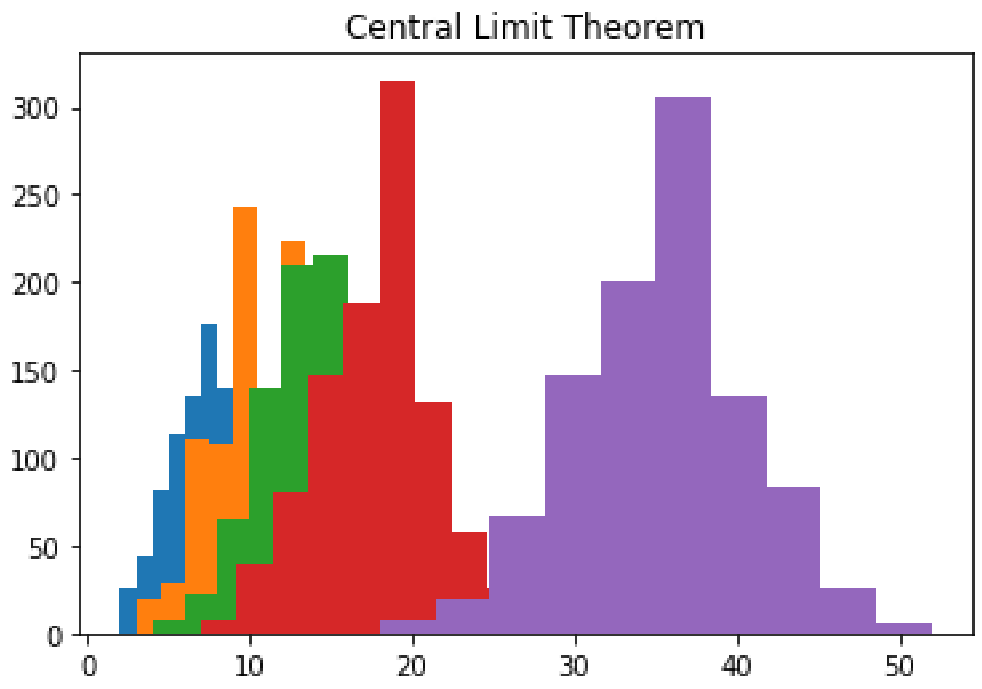
```python
def random_number(min, max):
    val = random.randint(min, max)
    return val
def N_rolls(m):
    values = []
    for i in range(m):
        start = random_number(1, 6)
        values.append(start)
    return values
die_roll = N_rolls(1000)
n, bins, patches = plt.hist(die_roll,6)
plt.show()
print(np.mean(die_roll))
print(np.std(die_roll))
```



```
3.478
1.707488213722133
```

[68]:
```python
Nvals = [2,3,4,5,10]
N_vals = []

#intial = N_rolls(1000)
#for N in Nvals:
    #v = N_rolls(1000)
    #N_vals.append(v)
```

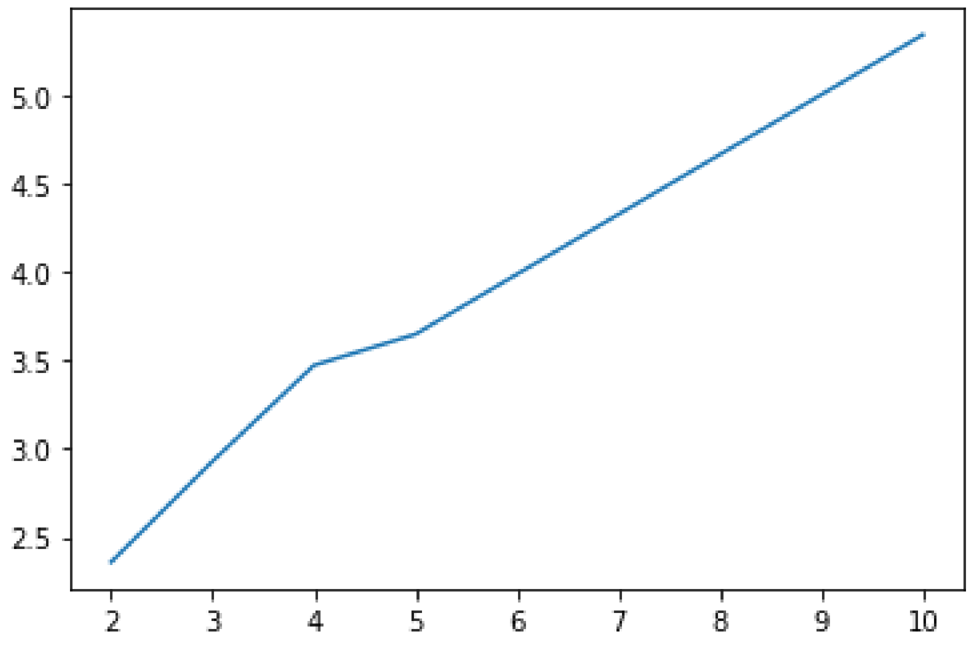
```
final = []
for val in Nvals:
    initial = np.array(N_rolls(1000))
    for i in range(val-1):
        v = np.array(N_rolls(1000))
        initial += v
    final.append(initial)
plt.title("Central Limit Theorem")
plt.hist(final[0])
plt.hist(final[1])
plt.hist(final[2])
plt.hist(final[3])
plt.hist(final[4])
plt.show()


print(np.mean(final[0]), np.std(final[0]))
print(np.mean(final[1]), np.std(final[1]))
print(np.mean(final[2]), np.std(final[2]))
print(np.mean(final[3]), np.std(final[3]))
print(np.mean(final[4]), np.std(final[4]))



#plt.hist(final[4])
#print(final[3])
#print(N_rolls(1000))
```



```
7.065 2.360248927549804
10.435 2.9308317931945527
13.982 3.473280293900854
17.703 3.6486697575965956
35.16 5.341198367407824
```

[70]:
```
std_values = [np.std(final[0]), np.std(final[1]), np.std(final[2]), np.
 ↪std(final[3]), np.std(final[4])]
plt.plot(Nvals, std_values)
'''I think the histograms do follow the central limit theorem '''
```

[70]:
```
'I think the histograms do follow the central limit theorem '
```



1.2 Problem 2: Parity-Violating Asymmetry

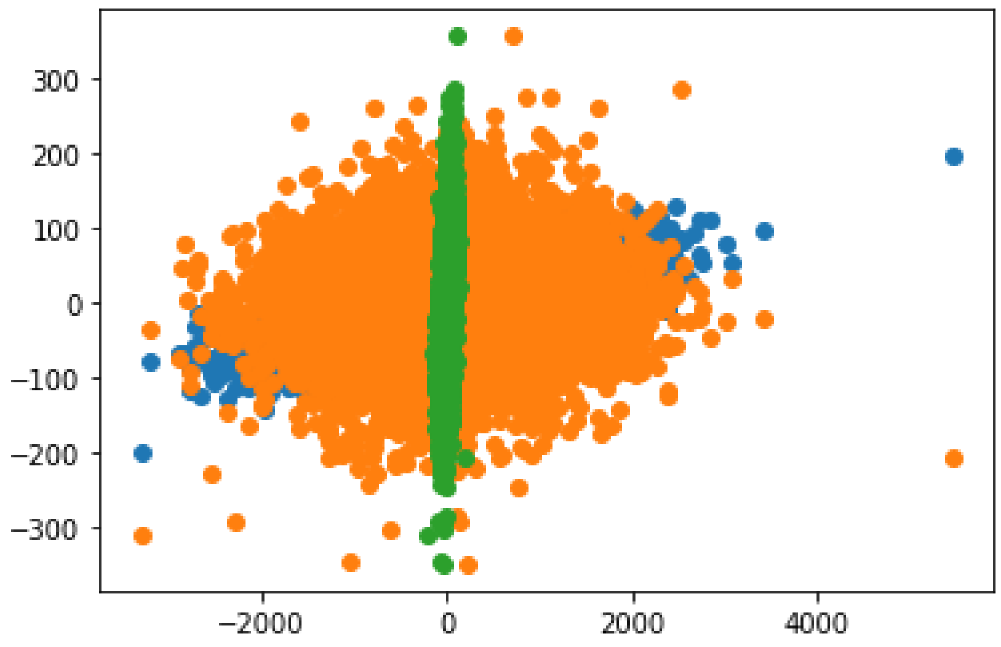
The data sample for this problem comes from the E158 experiment at SLAC (a national lab near that Junior university across the Bay). E158 measured a parity-violating asymmetry in Møller (electron-electron) scattering. This was a fixed-target experiment, which scattered longitudinally-polarized electrons off atomic (unpolarized) electrons in the 1.5m liquid hydrogen target. The data below contains a snapshot of 10,000 “events” from this experiment (overall, the experiment collected almost 400 million such events over the course of about 4 months). Each event actually records a pair of pulses: one for the right-handed electron (spin pointing along momentum) and one for the left-handed electron. For each event, we record 4 variables:

- Counter: event index
- Asymmetry: "raw" cross section asymmetry A_{raw} from one of the detector channels (there are 50 of these overall). The cross section asymmetry is defined as $A_{raw} = \frac{\sigma_R - \sigma_L}{\sigma_R + \sigma_L}$ The asymmetry is recorded in units of PPM (parts per million). It is called "raw" because corrections due to the difference in beam properties at the target are not yet applied.
- DeltaX: difference in beam position $\Delta X = X_R - X_L$ at the target in X direction in microns (with the convention that the beam is traveling along Z)
- DeltaY: difference in beam position $\Delta Y = Y_R - Y_L$ at the target in Y direction in microns

The data sample is provided in plain text format as the file asymdata.txt. Questions for this analysis:

1. Read the data from the file, and plot distributions of A_{raw}, ΔX, and ΔY.
2. Compute the mean of the raw asymmetry distribution and its statistical uncertainty.
3. Plot A_{raw} vs ΔX, A_{raw} vs ΔY, and ΔX vs ΔY as scatter plots.
4. Compute the correlation coefficients Corr(Asym,DeltaX), Corr(Asym,DeltaY), and Corr(DeltaX,DeltaY). See lecture notes, Workshop05.ipynb and Workshop05_optional.ipynb or https://en.wikipedia.org/wiki/Pearson_correlation_coefficient for additional help understanding correlation coefficients. Which variables are approximately independent of each other ?

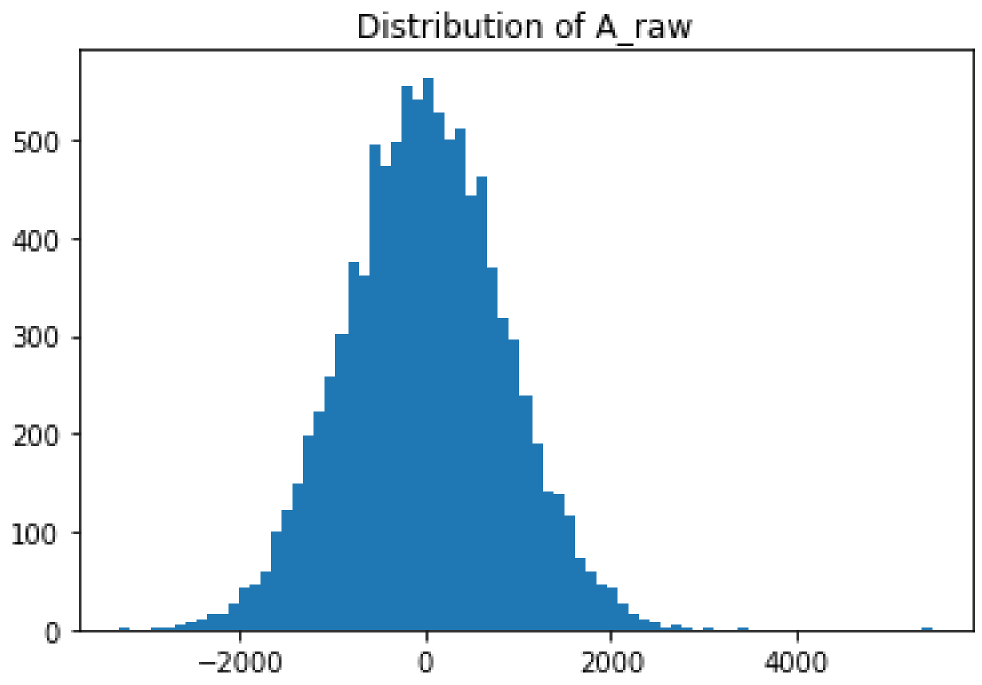
[98]:
```python
x = np.loadtxt('asymdata.txt')

A = []
Dy = []
Dx = []

for i in range(len(x)):
    row = x[i]
    A.append(row[1])
    Dy.append(row[2])
    Dx.append(row[3])

plt.scatter(A, Dx)
plt.scatter(A, Dy)
plt.scatter(Dx, Dy)
plt.show()

n = len(A)
mean = sum(A)/n
sum_squares = 0
for i in A:
    sum_squares = sum_squares + (i - mean)**2
uncertainty = (sum_squares/(n-1))**(1/2)
print('uncertainty :', uncertainty )
print('mean:', mean)
```

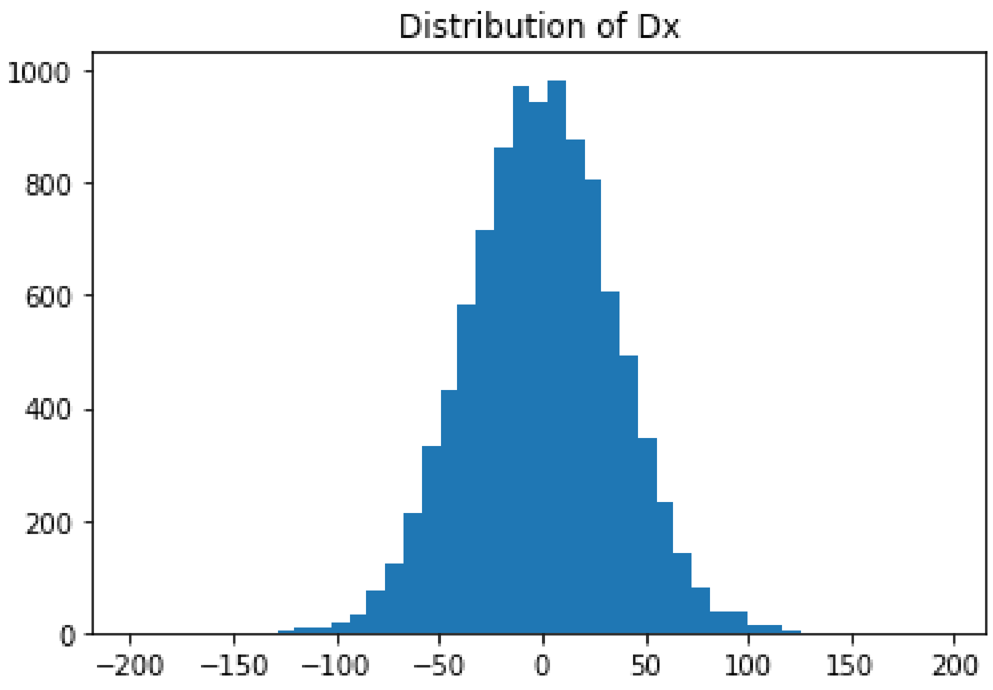


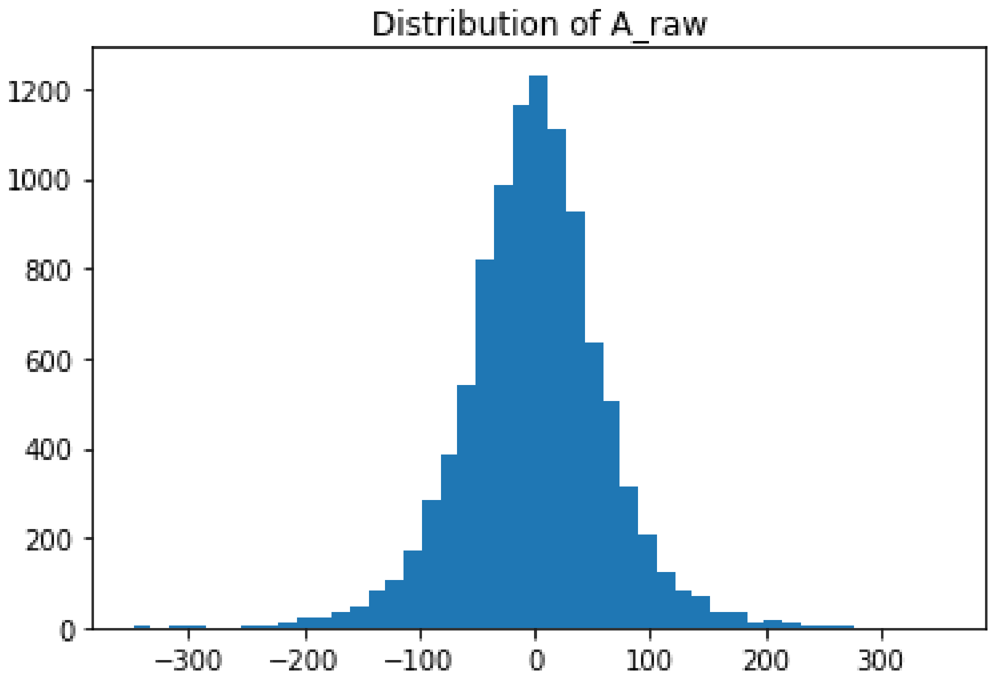
```
uncertainty : 848.8957761173451
mean: 0.4430145300000137
```

[107]:
```
plt.hist(A, bins = 75)
plt.title('Distribution of A_raw')
plt.show()

plt.hist(Dx, bins = 45)
plt.title('Distribution of Dx')
plt.show()

plt.hist(Dy, bins = 45)
plt.title('Distribution of A_raw')
plt.show()

def correlation(x, y):
    mxy = 0;
    mxx = 0
    myy = 0
    xmean = np.mean(x)
    ymean = np.mean(y)

    for i in range (0, len(x)):
        mxy += (x[i] - xmean) * (y[i] - ymean)
        mxx += (x[i] - xmean) * (x[i] - xmean)
        myy += (y[i] - ymean) * (y[i] - ymean)
```

```python
    return mxy/(np.sqrt(mxx*myy))

print(correlation(A, Dx))
print(correlation(A, Dy))
print(correlation(Dx, Dy))
```







```
0.6475434676821308
0.056392764237038845
```

```
0.40212859128196377
```

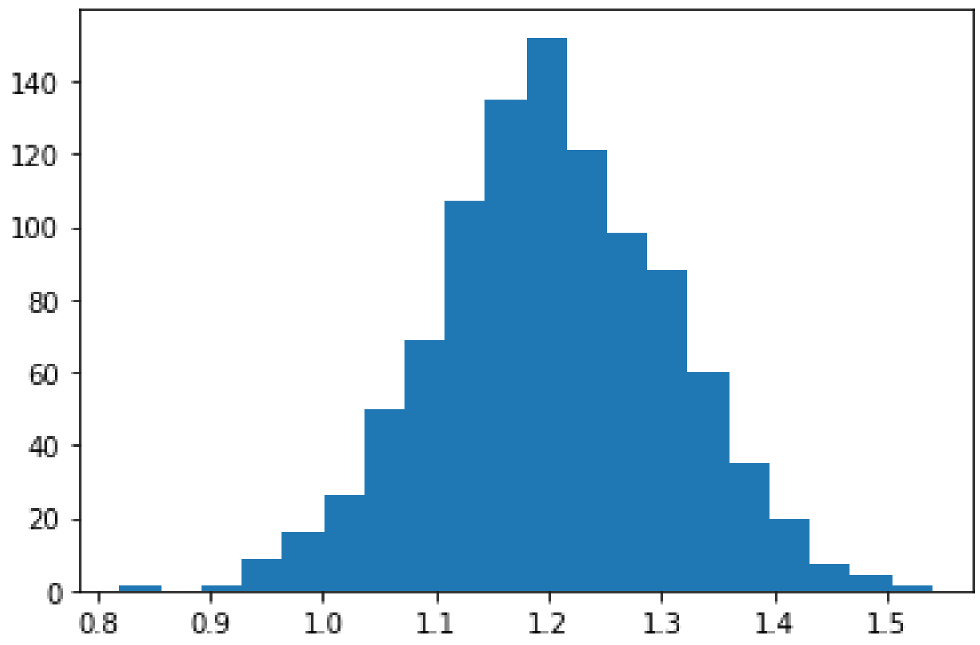
1.3 Problem 3: Gamma-ray peak

[Some of you may recognize this problem from Advanced Lab's Error Analysis Exercise. That's not an accident.]

You are given a dataset (`peak.dat`) from a gamma-ray experiment consisting of ~1000 gamma-ray hits. Each line in the file corresponds to one recorded gamma-ray event, and stores the the measured energy of the gamma-ray (in MeV). We will assume that the energies are randomly distributed about a common mean, and that each event is uncorrelated to others. Read the dataset from the enclosed file and: 1. Produce a histogram of the distribution of energies. Choose the number of bins wisely, i.e. so that the width of each bin is smaller than the width of the peak, and at the same time so that the number of entries in the most populated bin is relatively large. Since this plot represents randomly-collected data, plotting error bars would be appropriate. 1. Compute the mean and standard deviation of the distribution of energies and their statistical uncertainties. Assume the distribution is Gaussian and see the lecture notes for the formulas for the mean and variance of the sample and the formulas for the errors on these quantities. 1. Compute the fraction of events contained within $\pm 1\sigma$ of the mean, $\pm 2\sigma$ of the mean, and $\pm 3\sigma$ of the mean (where σ is the standard deviation you computed in Part 2). Compare these fractions with the quantiles of the Gaussian distribution (see lecture notes) ?

[110]:
```python
data = np.loadtxt('peak.dat')
plt.hist(data, 20)



n = len(data)
mean = sum(data)/n
sum_squares = 0
for i in data:
    sum_squares = sum_squares + (i - mean)**2
uncertainty = (sum_squares/(n-1))**(1/2)
print('uncertainty :', uncertainty )
print('mean:', mean)
```

```
uncertainty : 0.10383705612019474
mean: 1.2026802650000004
```


0
20
40
60
80
100
120
140
0.8
0.9
1.0
1.1
1.2
1.3
1.4
1.5

```
[ ]:
```