PSTAT160A F21 Python HW 1_sol

October 6, 2021

Python Homework 1 Due date: October 5, 11:59 p.m. via GauchoSpace

Instruction: Please upload your pdf or html file with your code and result on GauchoSpace with filename "PythonHW1_YOURPERMNUMBER".

The purpose of this Python Homework is to practice with sampling from a distribution with the **NumPy Package**.

Attention: Don't forget to import the necessary packages!

[1]:
```
import numpy as np
import numpy.random as npr
import matplotlib
from matplotlib import pyplot
```

0.1 Problem 1 (6 Points)

1. Simulate 100,000 realizations from the binomial distribution with N=2500 trails and success probability p=0.45.

[2]:
```
npr.seed(160)
sample = npr.binomial(2500,0.45,100000)
sample[0:5]
```

[2]:
```
array([1136, 1129, 1117, 1108, 1151])
```

[3]:
```
# after using the seed, every time the result will be the same
npr.seed(seed=160)
sample = npr.binomial(2500,0.45,100000)
sample[0:5]
```

[3]:
```
array([1136, 1129, 1117, 1108, 1151])
```

2. Compute the empirical mean and the empricial standard deviation of your sample and compare these values with the theoretical values.

[4]:
```
# theoretical:
Mean_the = 2500 * 0.45
Std_the  = np.sqrt(2500 * 0.45 * (1-0.45))
# empirical
```

```
Mean_emp = sample.mean()
Std_emp  = sample.std()
```

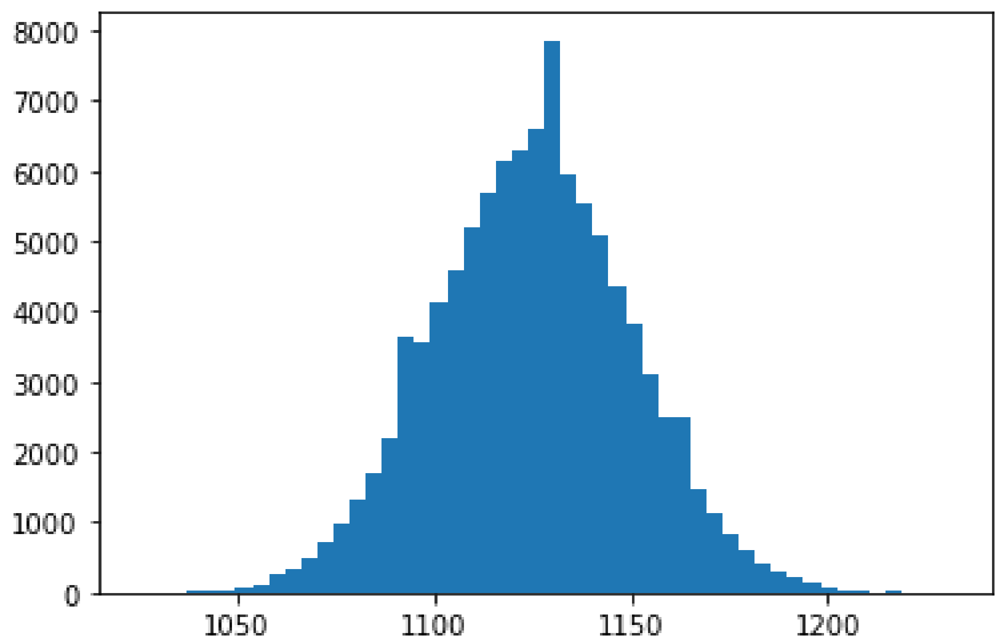
[5]:
```
Mean_the, Mean_emp, Std_the, Std_emp
```

[5]: (1125.0, 1125.05449, 24.8746859276655, 24.88304163159922)

3. Plot a histogram of your sample with the absolute number of counts for each bin. Choose 50 bins.

For details on pyplot.hist: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html

[6]:
```
a, b, c = pyplot.hist(x = sample, bins = 50, density = False)
```



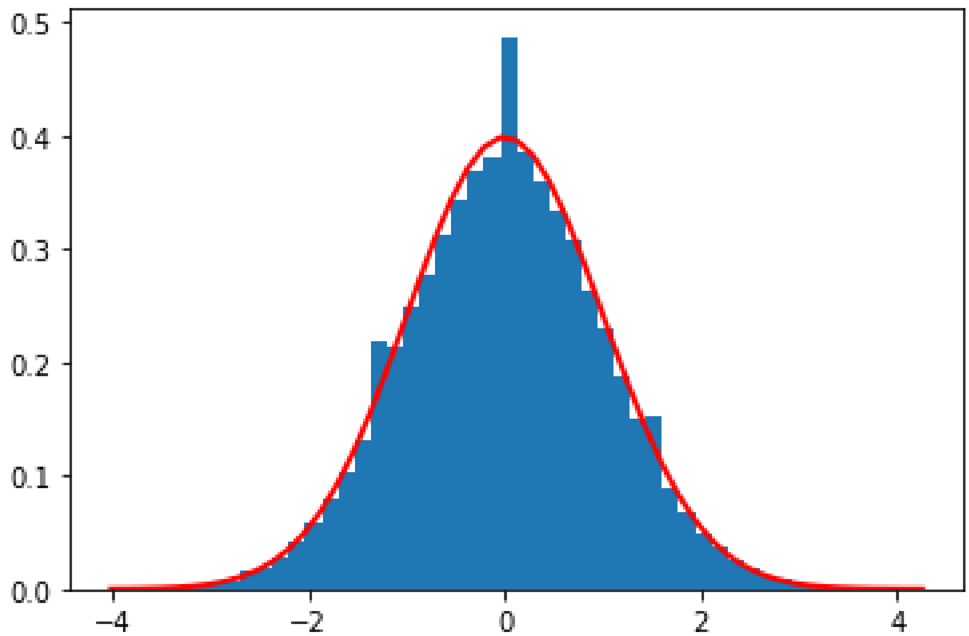
4. Standardize your sample, that is, subtract the emprical mean and divide by the empricial standard deviation.

[7]:
```
sample_standard = (sample - Mean_emp) / Std_emp
```

5. Plot a histogram of your standardized sample with the counts normalized to form a probability density. Choose again 50 bins. Compare your histrogram with the density of the standard normal distribution by inserting its density into the histogram plot.

[8]:
```
from scipy.stats import norm

#Get the histogram
count, bins, patches = pyplot.hist(sample_standard, 50, density = True)
```

```
#Standard Normal Density Curve
pyplot.plot(bins,norm.pdf(bins,0,1), linewidth=2, color='r')
pyplot.show()
```



0.2 Problem 2 (4 Points)

1. Implement the simulation of a biased 6-sided die which takes the values 1,2,3,4,5,6 with probabilities 1/8,1/12,1/8,1/12,1/4,1/3.

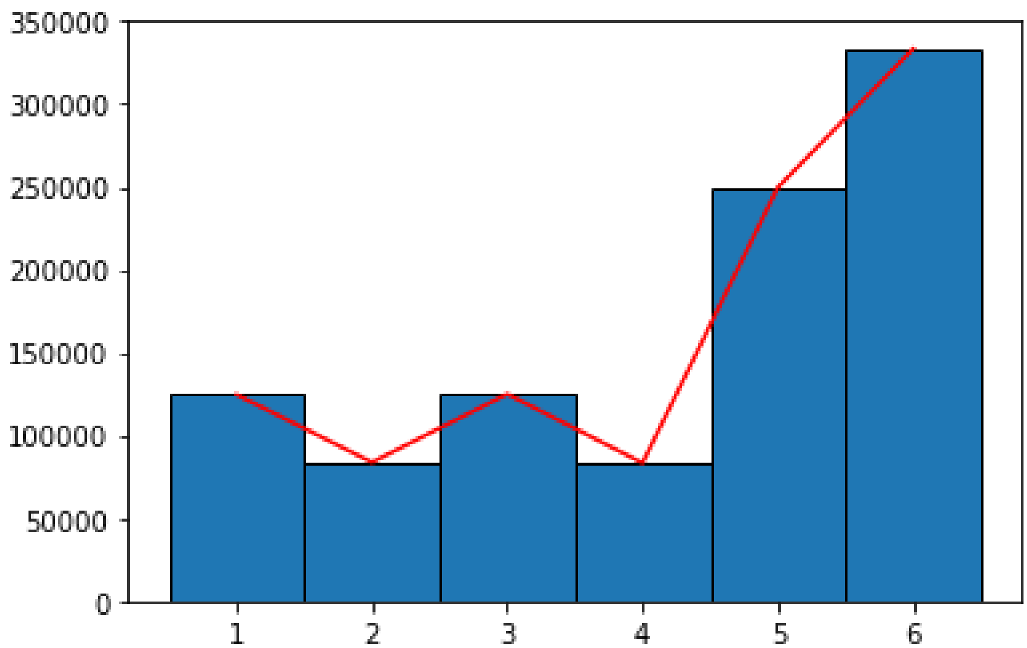
[9]:
```
npr.seed(160)
die = [1, 2, 3, 4, 5, 6]
probs = [1/8, 1/12, 1/8, 1/12, 1/4, 1/3]

#Simulation
npr.choice(die, 100, p=probs)
```

[9]:
```
array([6, 3, 4, 3, 5, 6, 2, 4, 3, 6, 6, 6, 4, 1, 1, 6, 5, 5, 3, 5, 6, 6,
       6, 5, 6, 6, 6, 5, 5, 4, 5, 5, 2, 5, 6, 5, 6, 1, 6, 2, 5, 2, 3, 5,
       6, 6, 5, 1, 1, 1, 6, 4, 3, 3, 6, 5, 5, 6, 5, 6, 6, 5, 5, 2, 2, 3,
       2, 5, 6, 6, 6, 1, 4, 2, 4, 5, 5, 5, 6, 5, 6, 5, 6, 6, 3, 3, 2, 6,
       6, 6, 6, 3, 6, 1, 1, 5, 6, 6, 6, 3])
```

2. Plot a histrogramm with 1,000,000 simulations to check if the relative counts of each number is approximately equal to the corresponding specified probabilities.

Remark: Specify the bins of your histogram correctly.

[11]:
```
npr.seed(160)
sim = npr.choice(die, 1000000, p=probs)
cc, bb, pp = pyplot.hist(sim, die+[7], align='left', ec='black')
pyplot.plot(bb[:-1], cc, color='r')
pyplot.show()
print('sample counts: ', cc)
print('theoretical values: ', [float('%.3f' % (probs[i] * 1000000)) for i in␣
↪range(len(probs))])
print('differences are very small: \n', [float('%.3f' % (cc[i] - probs[i] *␣
↪1000000)) for i in range(len(cc))])
```




```
sample counts:  [124676.  83749. 125039.  83342. 249780. 333414.]
theoretical values:  [125000.0, 83333.333, 125000.0, 83333.333, 250000.0,
333333.333]
differences are very small:
 [-324.0, 415.667, 39.0, 8.667, -220.0, 80.667]
```