Homework 2

March 2, 2023

1(a)A sample distribution and sampling distribution are the same thing. Ans: False

(b) We use sample parameters to estimate population statistics. Ans: True

(c) Sampling with replacement means every time we sample one observation from the population, we return that observation so we can (in principle, at least) sample it again. Ans: True

(d) A natural experiment is a kind of observational study. Ans:True

(e)The Law of Large Numbers can be violated under certain sample sizes. Ans:True

[91]:
```
#Q2
import numpy as np

lst = np.array([11, 4, 1, 1, 2, 8, 12, 60])

mean = np.mean(lst)
median = np.median(lst)
std = np.std(lst)
print("Mean:", mean)
print("Median:", median)
print("Standard deviation:", std)

lst[-1] = 6
mean = np.mean(lst)
median = np.median(lst)
std = np.std(lst)
print("Mean:", mean)
print("Median:", median)
print("Standard deviation:", std)


#e
def multiplier(lst):
    new_lst = lst*3
    return new_lst
print(multiplier(lst))

#f
for i in range(0,len(lst)):
```

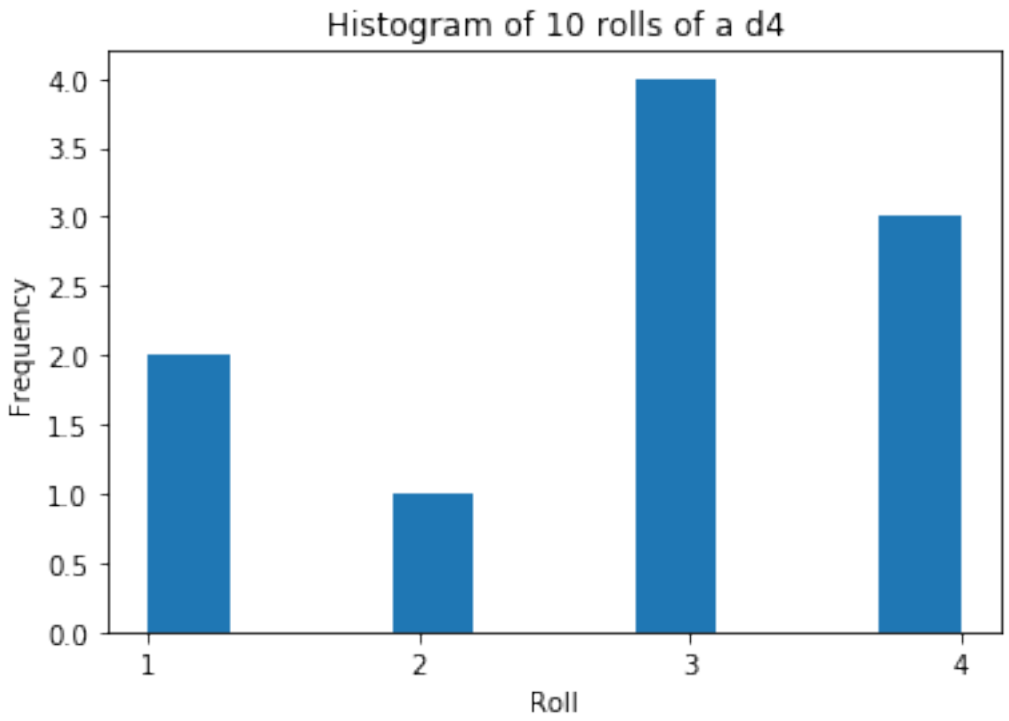
```
    lst[i] = lst[i]*3

print(lst)
```

```
Mean: 12.375
Median: 6.0
Standard deviation: 18.45899171135845
Mean: 5.625
Median: 5.0
Standard deviation: 4.090767042988393
[33 12  3  3  6 24 36 18]
[33 12  3  3  6 24 36 18]
```

(2c)The relative robustness of median is better than the mean.The mean deviates a lot from the mean of the false data.However,the median does not as it is the middle number of the sorted list and mean is the average of the data sets.

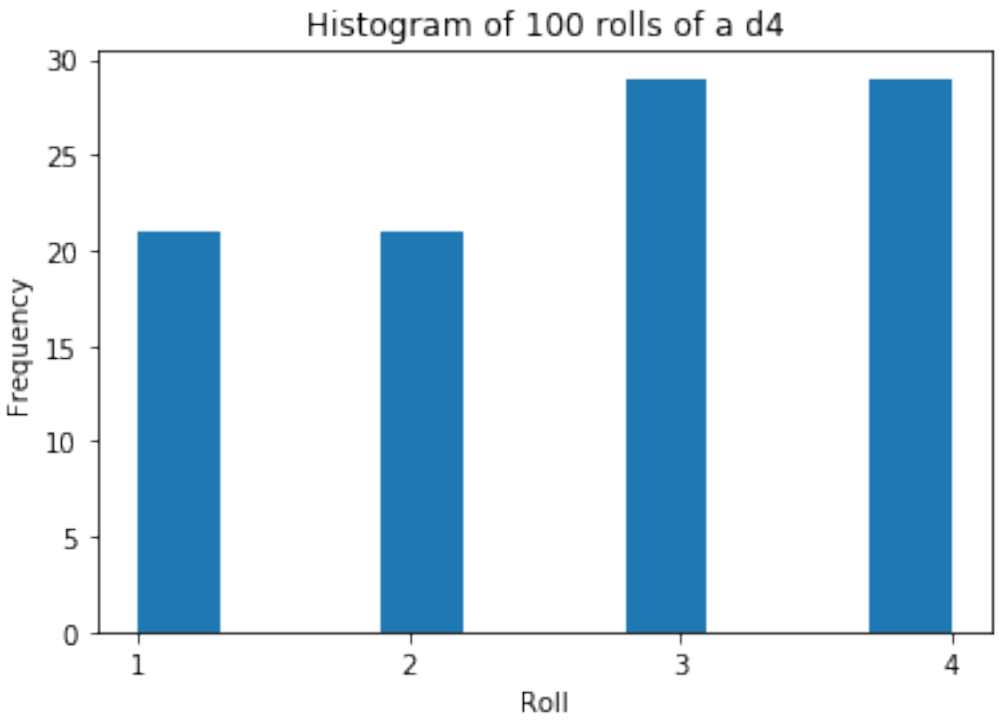
(2d) The false data set had a number which was larger than the corrected data set. The false data set as a result had a higher spike which resulted in a higher standard deviation.

[28]:
```
#3
import numpy as np
import matplotlib.pyplot as plt



# Roll the d4 10 times
rolls = np.random.choice([1, 2, 3, 4], size=10)

# Plot the histogram
plt.hist(rolls)
plt.xticks([1,2,3,4])
plt.xlabel('Roll')
plt.ylabel('Frequency')
plt.title('Histogram of 10 rolls of a d4')
plt.show()
```

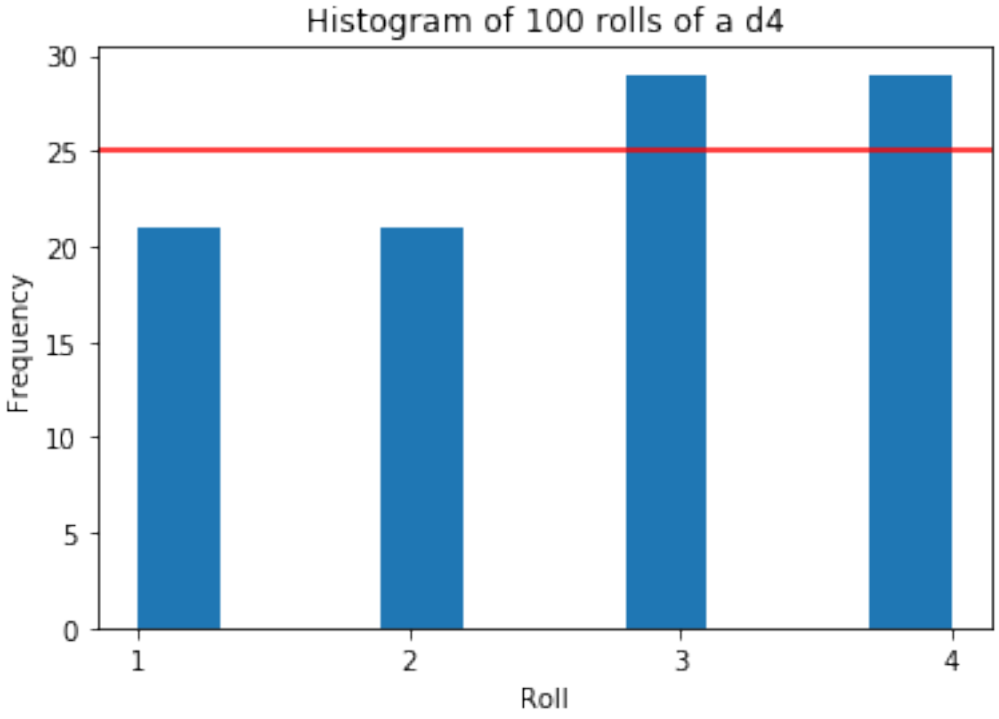


(3b) Two of the ten flips turned out to be 4. Theoretically, you would expect 4 to come up 2.5 times(i.e 2 or 3 times)

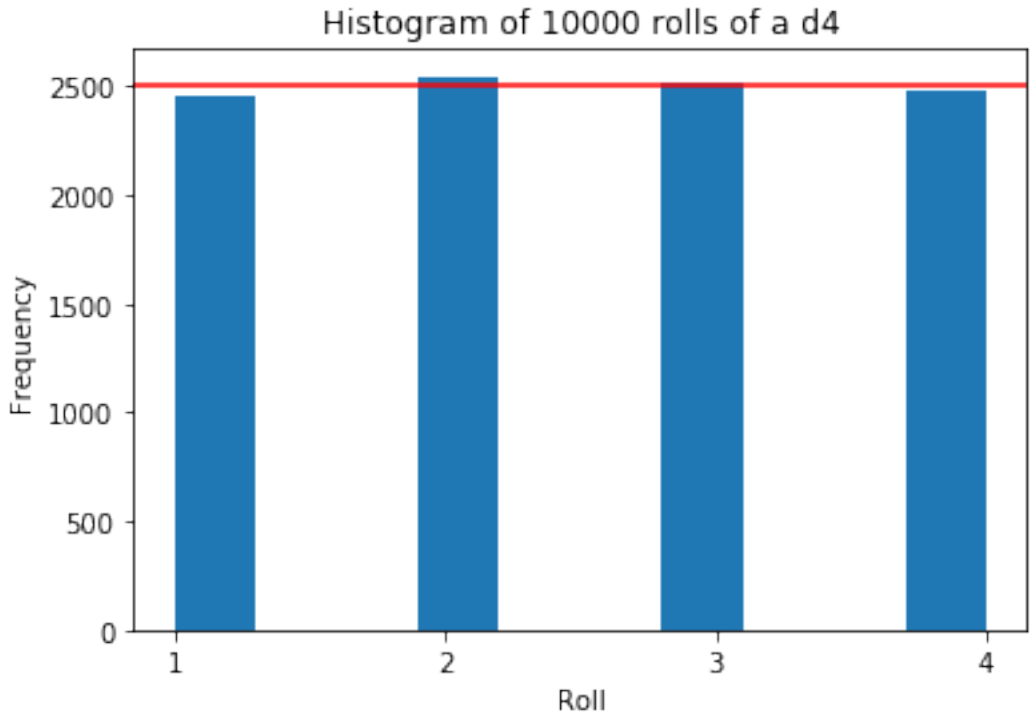
[33]:
```
rolls = np.random.choice([1, 2, 3, 4], size=100)
plt.hist(rolls)
plt.xticks([1,2,3,4])
plt.xlabel('Roll')
plt.ylabel('Frequency')
plt.title('Histogram of 100 rolls of a d4')
plt.show()
plt.hist(rolls)
plt.xticks([1,2,3,4])
plt.xlabel('Roll')
plt.ylabel('Frequency')
plt.title('Histogram of 100 rolls of a d4')
plt.axhline(y=25,color = 'r')
```



[33]: <matplotlib.lines.Line2D at 0x7f64d190bbd0>



[35]:
```
rolls = np.random.choice([1, 2, 3, 4], size=10000)
plt.hist(rolls)
plt.xticks([1,2,3,4])
plt.xlabel('Roll')
plt.ylabel('Frequency')
plt.title('Histogram of 10000 rolls of a d4')
plt.axhline(y=2500,color = 'r')

plt.show()
```



(3f) By performing the same experiment a larger number of times will yield in more data and less flukes whic as a result would be closer to the theoretical yield. And as there are more rolls the average and precision will also increase due to the law of large numbers.

[24]:
```
import pandas as pd
data= pd.read_csv("population_data.csv",squeeze=True)
```

[24]:
```
0     4.340970
1     1.837203
2     3.518534
3     0.305815
4    16.428887
```

```
Name: 0, dtype: float64
```

[13]:
```
#Q4(a)
from random import sample
mean_data=data.mean()
print("mean is",mean_data)
median_data=data.median()
print("median is",median_data)
std_data=data.std()
print("standard deviation is",std_data)
```

```
mean is 8.330765833333718
median is 7.047814401943995
standard deviation is 6.291949735580204
```

[38]:
```
sp = data.sample(n=25)
mean_data_1 = sp.mean()
print("mean is",mean_data_1)
median_data_1=sp.median()
print("median is",median_data_1)
std_data_1=sp.std()
print("standard deviation is",std_data_1)
```

```
mean is 8.57819707111611
median is 7.4245393219825555
standard deviation is 6.052358728000899
```

[74]:
```
sp_1 = data.sample(n=1000)
mean_data_2 = sp_1.mean()
print("mean is",mean_data_2)
median_data_2=sp_1.median()
print("median is",median_data_2)
std_data_2=sp_1.std()
print("standard deviation is",std_data_2)
```

```
mean is 0    8.530862
dtype: float64
median is 0    7.12553
dtype: float64
standard deviation is 0    6.396053
dtype: float64
```

This happens due to the law of large numbers. The more sampling you do the closer to the actual value you get.

[48]:
```
def complex_statistic(data):
    y_1 = np.power(data, 0.25)
    y_2 = -np.sqrt(y_1)
```

```
    y_3 = np.exp(y_2)
    y_4 = np.sin(y_3)

    percentile_80 = np.percentile(y_4, 80)

    return percentile_80
print(complex_statistic(data))
sp_2 = data.sample(n=100)
print(complex_statistic(sp_2))
sp_3 = data.sample(n=1000)
print(complex_statistic(sp_3))
```

```
0.3177257855396434
0.3122940705103871
0.31853828020243713
```

The sample with more data works better. The law of large numnbers does work still works well with complex statistics

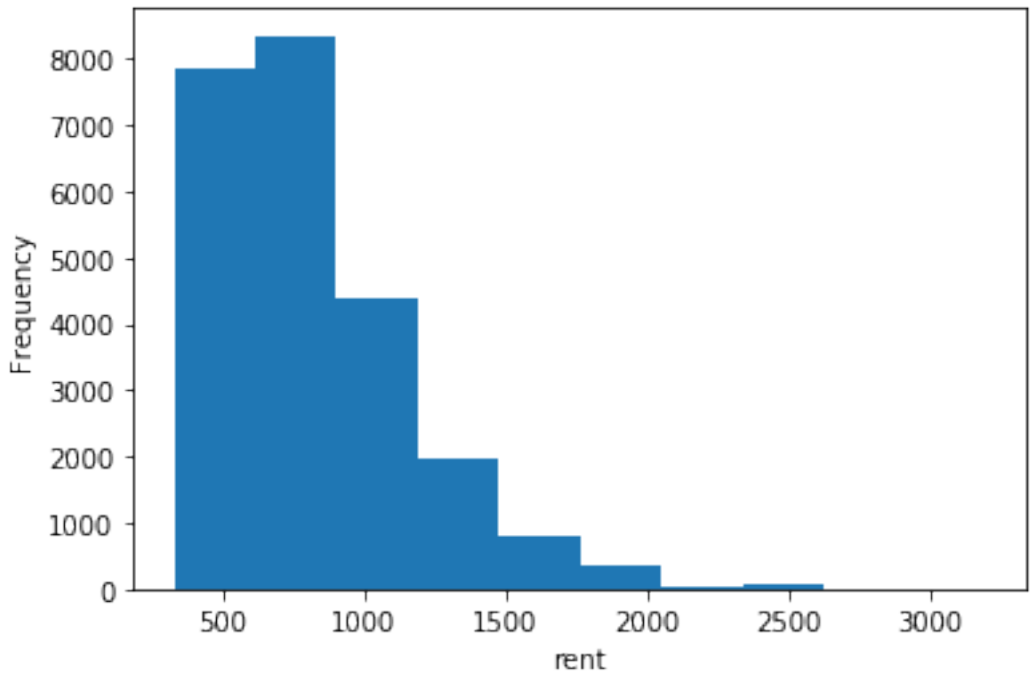
[92]:
```
data_1= pd.read_csv("rents_county.csv",squeeze=True)
data_1.head(5)
```

[92]:

	year	fips	state	county	bedrooms	rent
0	2009	100199999	1	1	0	580
1	2009	100399999	1	3	0	524
2	2009	100599999	1	5	0	453
3	2009	100799999	1	7	0	590
4	2009	100999999	1	9	0	590

[63]:
```
plt.hist(data_1['rent'])
plt.ylabel('Frequency')
plt.xlabel("rent")

plt.show
```

[63]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



Its skewed right. This means the peak of the graph lies towards the left of the histogram(i.e the lower end)

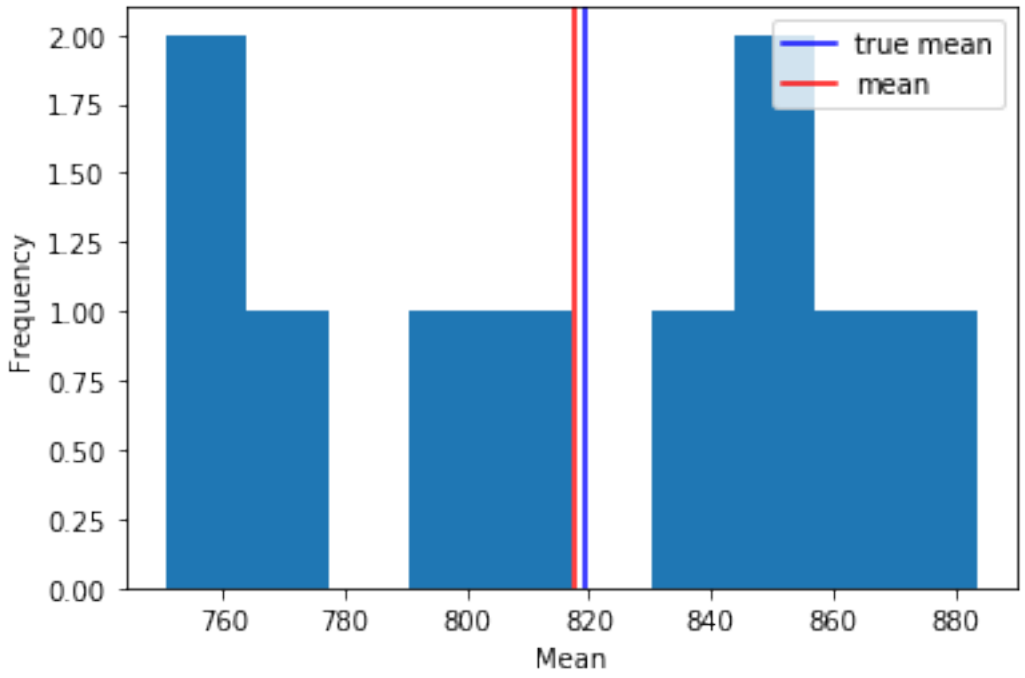
[73]:
```
sp_3 = data_1['rent']

count_sp3 = len(sp_3)
print(count_sp3)


mean_sp3 = round(sp_3.mean(), 1)
print(mean_sp3)


max_sp3 = round(max(sp_3), 1)
print(max_sp3)

std_sp3 = round(sp_3.std(), 1)
print(std_sp3)
```

```
23815
819.2
3198
343.3
```

[88]:
```
sp_4 = sp_3.sample(n=100)
mean_sp4 = sp_4.mean()

print(mean_sp4)
lst = []
for i in range(10):
    sp_4 = sp_3.sample(n=100)
    lst.append(sp_4.mean())
print(lst)
plt.hist(lst)
plt.ylabel('Frequency')
plt.xlabel('Mean of 10 times')
arr = np.array(lst)
plt.axvline(x = sp_3.mean(), color = 'b')
plt.axvline(x=arr.mean(),color = 'r')
plt.legend(('true mean','mean'),loc = 'upper right')
```

```
787.9
[844.9, 810.39, 750.56, 848.03, 883.77, 802.93, 860.22, 760.66, 771.25, 841.39]
```

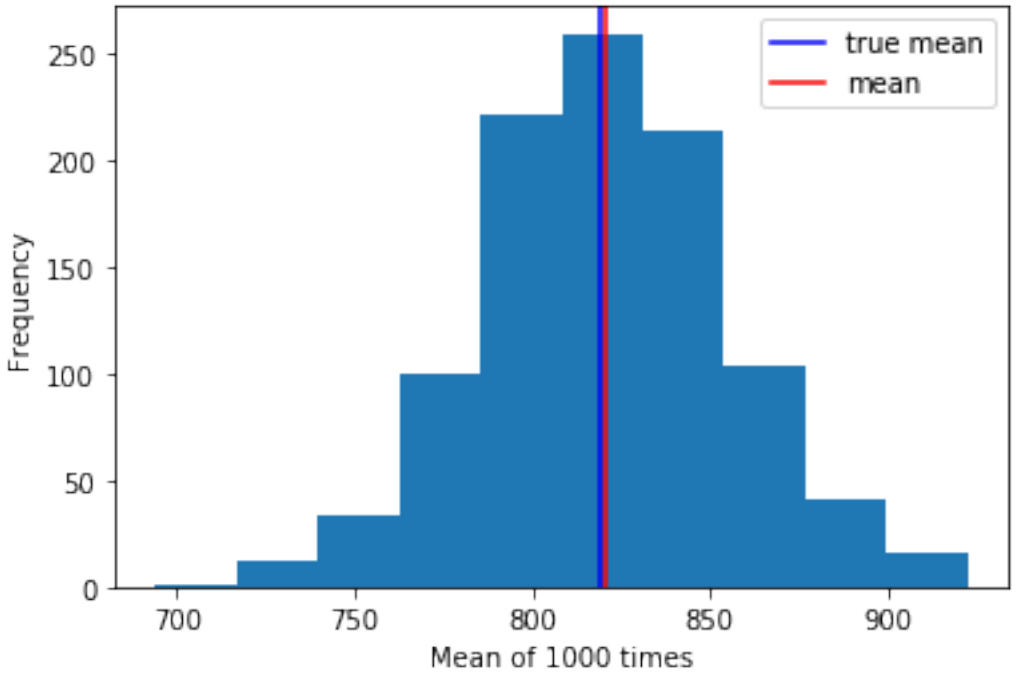
[88]:
```
<matplotlib.legend.Legend at 0x7f64d1af1f90>
```



[90]:
```
lst = []
for i in range(1000):
    sp_4 = sp_3.sample(n=100)
```

```
    lst.append(sp_4.mean())

plt.hist(lst)
plt.ylabel('Frequency')
plt.xlabel('Mean of 1000 times')
arr = np.array(lst)
plt.axvline(x = sp_3.mean(), color = 'b')
plt.axvline(x=arr.mean(),color = 'r')
plt.legend(('true mean','mean'),loc = 'upper right')
```

[90]: <matplotlib.legend.Legend at 0x7f64d1b17b10>



5)(g)As the number of sample increases we get a normal distribution. The true mean and the mean of the sample get closer to each other.

5)(h) The shape of the histogram changes to a normal distribution curve(as a bell curve.

5)(i)The central limit theorem predicts this outcome