Homework 2 - Suvir Wadhwa

March 6, 2022

[53]:
```
import numpy as np
import random
import pandas as pd
import matplotlib.pyplot as plt
```

Question 1

[54]:
```
print("Sample (a)")
data = np.array([1, 1, 2, 4, 5, 6, 7, 18])
mean = data.mean()
median = np.median(data)
sd = data.std()
print("The mean is {:.2}".format(mean))
print("The median is {:.2}".format(median))
print("The standard deviation is {:.2}".format(sd))
```

```
Sample (a)
The mean is 5.5
The median is 4.5
The standard deviation is 5.2
```

[56]:
```
print("Sample (b)")
data = np.array([100, 1, 2, 4, 5, 6, 7, 18]) #Array with the first element␣
↪replaced to be 100
mean = data.mean()
median = np.median(data)
sd = data.std()
print("The mean is {:.2f}".format(mean))
print("The median is {:.2}".format(median))
```

```
Sample (b)
The mean is 17.88
The median is 5.5
```

(c) The value for the mean isn't robust as an outlier impacted the total final value significantly. However, the value for the median changed a little but not as much, making it more robust than the mean

Question 2

[58]:
```
# Question 2, part (a)
def sampler(array_in, replace_in): #Function to create a randon sample from a␣
↪given array, with/without replacement
    sample = np.random.choice(array_in, size = len(array_in), replace =␣
↪replace_in)
    return sample
```

[59]:
```
# Question 2, part (b)
print("With Replacement:\n")
data2 = np.array([1, 1, 2, 4, 5, 6, 7, 18])
for i in range(3):
    sam = sampler(data2, True)
    print("Mean for the {}st sample is {}".format(i, sam.mean()))
```

```
With Replacement:

Mean for the 0st sample is 3.625
Mean for the 1st sample is 3.625
Mean for the 2st sample is 3.875
```

[60]:
```
# Question 2, part (c)
print("Without Replacement:\n")
data2 = np.array([1, 1, 2, 4, 5, 6, 7, 18])
for i in range(3):
    sam = sampler(data2, False)
    print("Mean for the {}st sample is {}".format(i, sam.mean()))
```

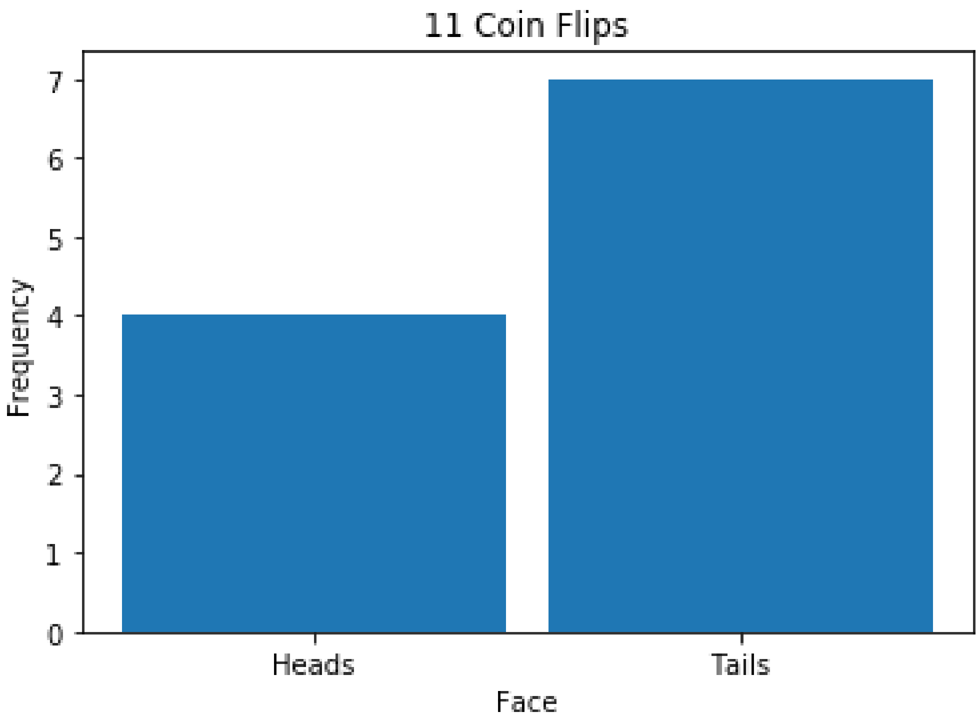
```
Without Replacement:

Mean for the 0st sample is 5.5
Mean for the 1st sample is 5.5
Mean for the 2st sample is 5.5
```

(d) In the case we sample with replacement, the means of differnet samples differ from one another without any given pattern. This is because the values are repeated in some cases. However, in the case we do it without replacing, our sample includes all samples from the input array only once. In this case, the mean will always be the same.

Question 3

[61]:
```
# Question 3, part (a)
results = np.random.choice(['Heads', 'Tails'], size= 11) #Simulate 11 coin flips
plt.hist(results,
         bins=[-.5, .5, 1.5], rwidth=.9) # make the bars take up 90% of the␣
↪width to add some white space
plt.title('11 Coin Flips') # title
plt.xlabel('Face') # x axis label
```

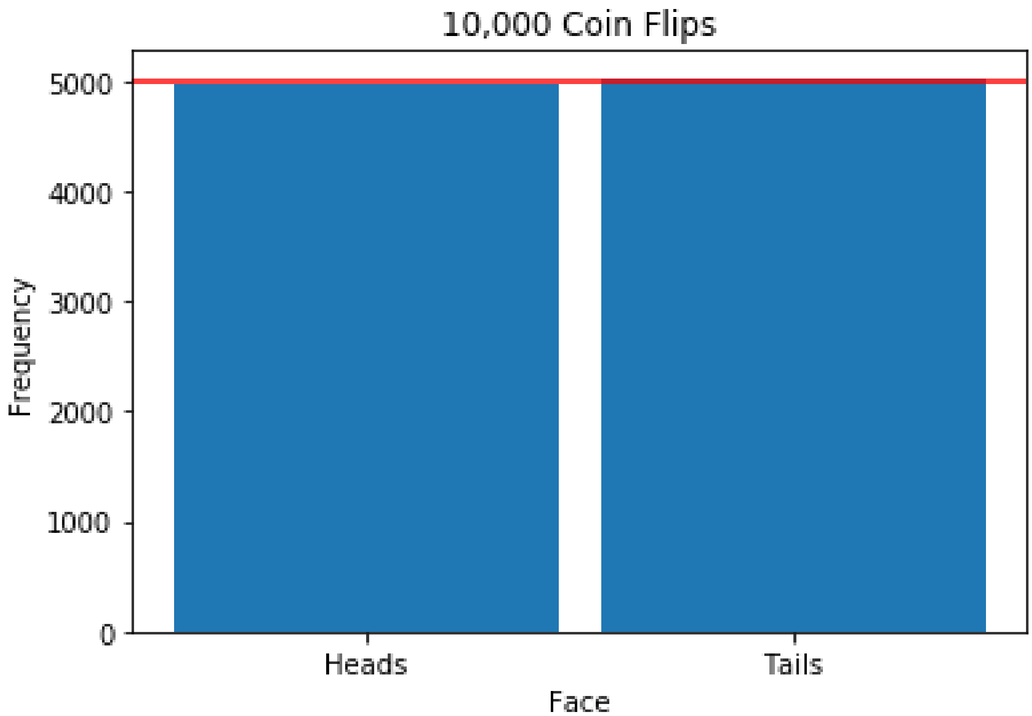
```
plt.ylabel('Frequency') # y axis label
plt.show()
```



[63]:
```
# Question 3, part (b)
print("The number of coin flips that showed up as heads were: 4")
```

```
The number of coin flips that showed up as heads were: 4
```

[65]:
```
# Question 3, part (c)
results = np.random.choice(['Heads', 'Tails'], size= 10000) #Simulate 10,000␣
 ↪coin flips
plt.hist(results,
         bins=[-.5, .5, 1.5], rwidth=.9) # make the bars take up 90% of the␣
 ↪width to add some white space
plt.title('10,000 Coin Flips') # title
plt.axhline((1/2)*10000, color='red', label='Expected distribution (LLN)')␣
 ↪#Line to represent LLN Expectation
plt.xlabel('Face') # x axis label
plt.ylabel('Frequency') # y axis label
plt.show()
```



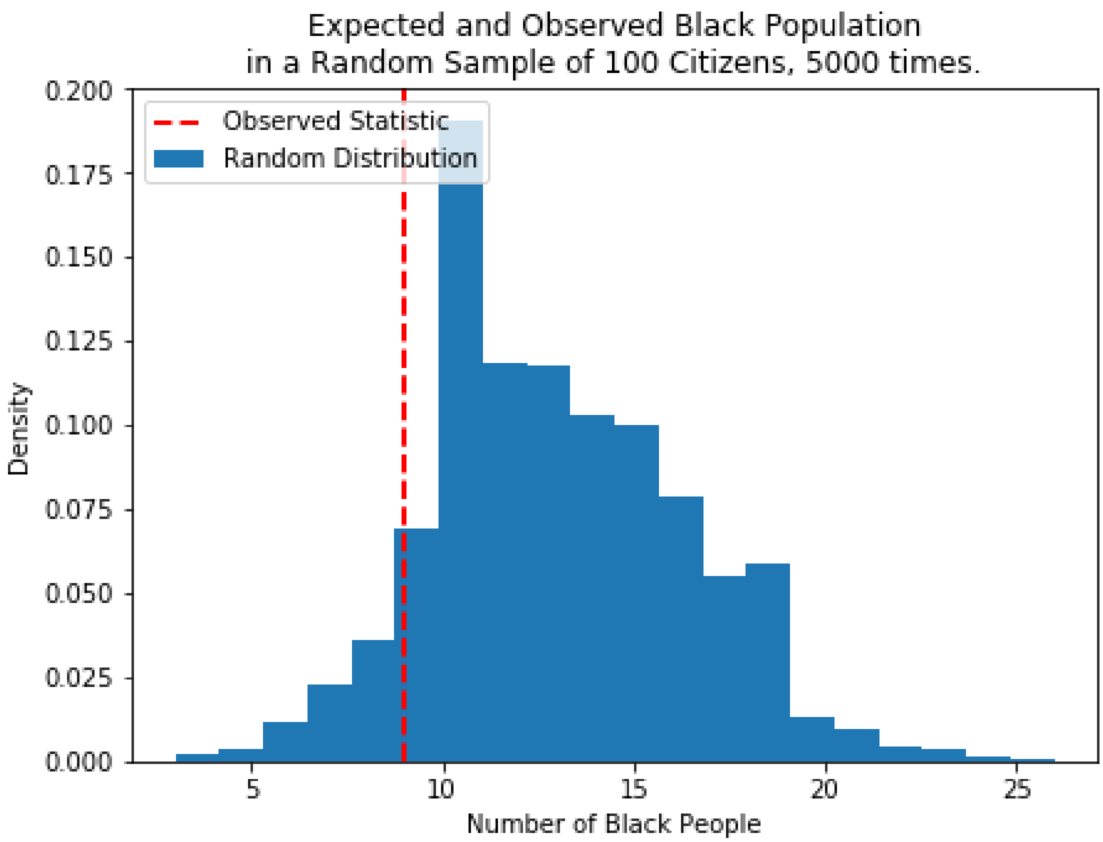
(d) The larger sample has values that are closer to what we would expect. As the sample size grew, the probability that the the sample statistic is close to the true statistic goes to 1.

Question 4

[66]:
```
# Question 4, part (a)
def black_panel(samples): #Function to predict number of black people in each␣
↪sample given probability.
    counts = np.zeros(samples)
    for i in range(samples):
        panel = np.random.choice([1, 0, 0, 0, 0], # possible outcomes
                                 size=100, # size of panel
                                 p=[.13,.48,.25,.11,.03])
        total = sum(panel)
        counts[i] = total
    return counts
```

[67]:
```
# Question 4, part (b)
sample = black_panel(5000) #Predict number of black people in 5000 samples
plt.figure(figsize = (7, 5))
plt.hist(sample,
         bins=20, # specify number, but not exact position of bins
         weights=np.ones(len(sample))/len(sample),
```

```
         label='Random Distribution')

plt.axvline(x=9,
          color='red', ls='--', lw=2,
          label = 'Observed Statistic')

plt.xlabel('Number of Black People')
plt.ylabel('Density')
plt.title('Expected and Observed Black Population\nin a Random Sample of 100␣
↪Citizens, 5000 times.')
plt.legend(loc='upper left')

plt.show()
```



(c) The number of black members in the observed statistic was 9. The mode of the data is 11 and hence 9 isn't too far from it. Hence, it is highly plausible that the jury was picked at random.

[68]:
```
# Question 4, part (d)
```

```
p = len(np.where(sample <= 9)[0]) / len(sample) #Measure to calculate p-value␣
↪for less than or equal to 9.
print("The estimated p-value for 9 or fewer black panelists is:", p)
```

```
The estimated p-value for 9 or fewer black panelists is: 0.145
```

(e) Out of a sample of a 100 people, the probability of picking a black person is 0.13 or 13%. Hence, we can assume that with any average sample, we can expect around 13 black people. Hence, as the p-value is greater, **I would fail to reject the Null Hypothesis**.

(f) A type 2 Error.

Question 5

[69]:
```
# Question 5, part (a)
df = pd.read_csv('horses.csv') #Load csv file into a pandas dataframe
df.head() #First 5 rows
```

[69]:
```
                name    price      sex       height     color  \
0   Always Bar Time  18500.0  Gelding  16.1 hands     Brown
1             Sonny   1000.0  Gelding  14.2 hands       Bay
2            Norman   5800.0  Gelding  15.2 hands    Sorrel
3  Twist Kitty Chex   2000.0     Mare         NaN  Palomino
4    Drews Approval   6500.0  Gelding  16.2 hands  Palomino

                  location                     markings       weight foaldate  \
0  Arthur, Ontario, Canada                          NaN  1100 pounds    4-Mar
1       Carbondale, Kansas                          NaN          NaN    1-Jan
2          Hale, Michigan  3 white socks and a blaze          NaN    5-Apr
3     Bloomfield, Nebraska                          NaN   600 pounds   12-May
4          Oglesby, Texas                          NaN  1100 pounds   Apr-00

                                      registrations  \
0  AQHA - American Quarter Horse Association (453…
1                                               NaN
2  AQHA - American Quarter Horse Association (491…
3  AQHA - American Quarter Horse Association (549…
4                ApHC - Appaloosa Horse Club (617728)

                                        disciplines  temperament  \
0  Hunter Under Saddle (Champion) Equitation (Cha…            2
1  Youth/4-H Horse (Prospect) Trail Horse (Traine…            2
2  Halter (Competed or Shown) Horsemanship (Compe…            3
3  Cutting (Prospect) Ranch Sorting (Prospect) Re…            1
4  Showmanship (Competed or Shown) Youth/4-H Hors…            4

                                               link
0  http://www.equine.com/horses-for-sale/horse-ad…
1  http://www.equine.com/horses-for-sale/horse-ad…
```

```
2   http://www.equine.com/horses-for-sale/horse-ad…
3   http://www.equine.com/horses-for-sale/horse-ad…
4   http://www.equine.com/horses-for-sale/horse-ad…
```

[70]:
```python
# Question 5, part (b)
lst = df['price'].tolist()
print("Number of Observations = {}".format(len(lst)))
print("Mean of Data = {:.2f}".format(np.mean(lst)))
print("Median of Data = {:.2f}".format(np.median(lst)))
print("Standard Deviation of Data = {:.2f}".format(np.std(lst)))
```

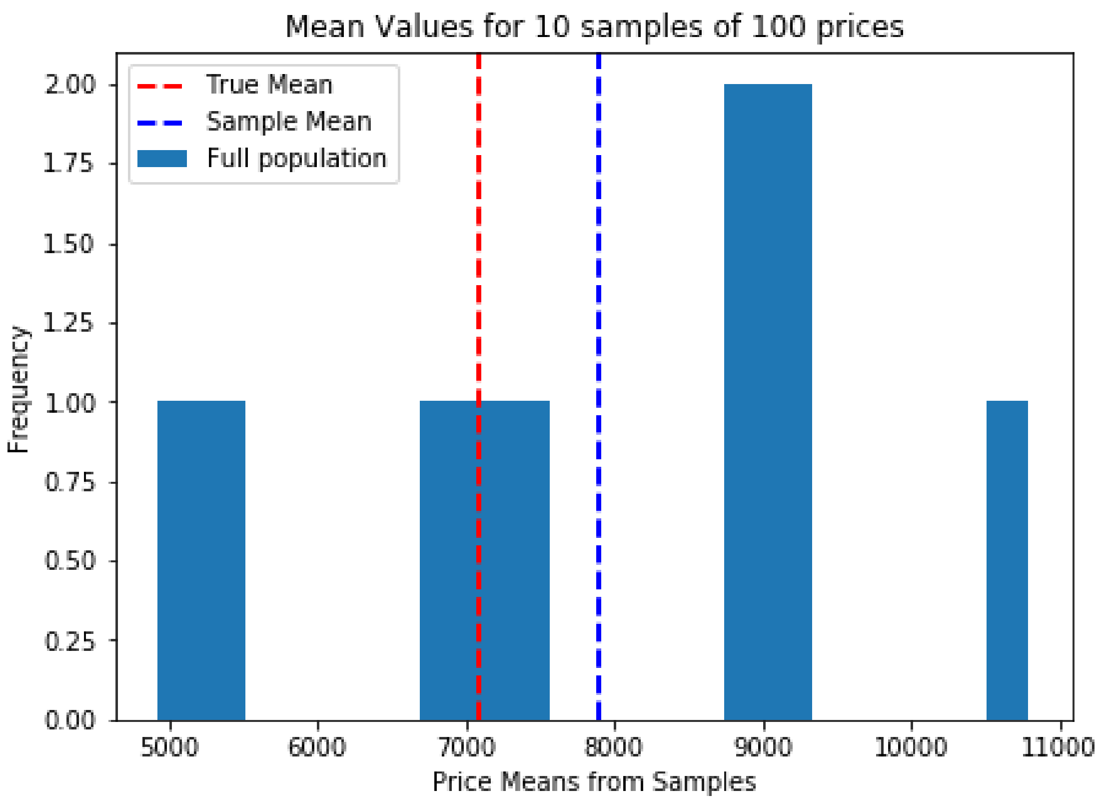
```
Number of Observations = 1080
Mean of Data = 7084.90
Median of Data = 3350.00
Standard Deviation of Data = 12690.62
```

[71]:
```python
# Question 5, part (c)
sample = np.random.choice(lst, 100)
print("Number of Observations = {}".format(len(sample)))
print("Mean of Data = {:.2f}".format(np.mean(sample)))
print("Median of Data = {:.2f}".format(np.median(sample)))
print("Standard Deviation of Data = {:.2f}".format(np.std(sample)))
```

```
Number of Observations = 100
Mean of Data = 7575.50
Median of Data = 3000.00
Standard Deviation of Data = 18212.05
```

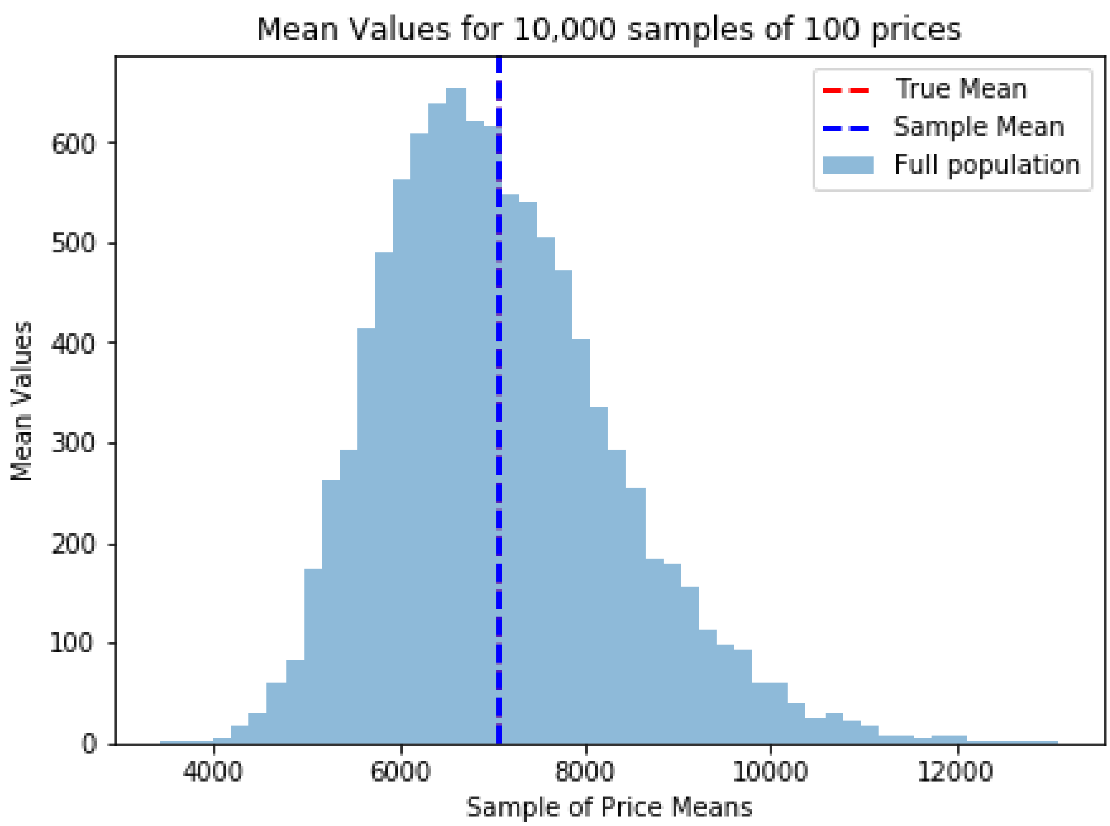
[72]:
```python
# Question 5, part (d)
mean_list = []
for i in range(10):
    sample = np.random.choice(lst, 100)
    mean_list.append(np.mean(sample))

plt.figure(figsize=(7,5))

# plot true distribution
plt.hist(mean_list, bins=20,
              label='Full population') # alpha makes transparent!

    # labels etc

plt.axvline(x= np.mean(lst) ,
          color='red', ls='--', lw=2,
          label = 'True Mean')

plt.axvline(x= np.mean(mean_list) ,
```

```
            color='blue', ls='--', lw=2,
            label = 'Sample Mean')


plt.xlabel('Price Means from Samples')
plt.ylabel('Frequency')
plt.title('Mean Values for 10 samples of 100 prices')
plt.legend()
plt.show()
```



[73]:
```
# Question 5, part (e)
mean_list = []
for i in range(10000):
    sample = np.random.choice(lst, 100)
    mean_list.append(np.mean(sample))

plt.figure(figsize=(7,5))

# plot true distribution
plt.hist(mean_list, bins=50,
             alpha=0.5, label='Full population')
```

```
    # labels etc

plt.axvline(x= np.mean(lst),
          color='red', ls='--', lw=2,
          label = 'True Mean')

plt.axvline(x= np.mean(mean_list) ,
          color='blue', ls='--', lw=2,
          label = 'Sample Mean')


plt.xlabel('Sample of Price Means')
plt.ylabel('Mean Values')
plt.title('Mean Values for 10,000 samples of 100 prices')
plt.legend()
plt.show()
```



(f) The mean of the sampling distribution approaches the true mean as the number of samples increases.

(g) The shape of the sampling distribution becomes more symmetric and centered as the number of samples increases.

(h) The central limit Theorem

<**End of Assignment**>

```
[ ]:
```