PS4

May 1, 2023

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
```

0.1 1. Explore Central Limit Theorem

[2]:
```
def makehist(S):
    num_bins = math.ceil(math.sqrt(R))
    print(num_bins)
    np.random.seed(1)
    mean = []
    for i in range(R):
        mean.append(np.mean(np.random.randint(0,2, size=S)*2-1))
    plt.hist(mean, bins= 30, edgecolor="k")
    plt.show()
    s_mean = np.mean(mean)
    print("Sample Mean: " + str(s_mean))
    s_var = np.var(mean)
    print("Sample Variance: " + str(s_var) + "\n")
    print("Expected Value stays the same at " + str(ex_v))
    print("Variance of Pair Means is VarX * 1/S, which is " + str(var/S) + "\n")
    print("Difference between Expected Value & Sample Mean: " + str(ex_v -␣
↪s_mean))
    print("Difference between Variance & Sample Variance: " + str(var/S -␣
↪s_var))
```

1.1. What is Random Variable (RV)? What makes X a RV? RV is a phenomenon or experiment with well-defined properties and a rule how to assign numeric labels to the events in that phenonmenon. In this case X is a RV because it assigns the values -1 and 1 respectively to the events of flipping a Tails or a Head.

1.2. Calculate the expected value and variance of this random variable. Explain what is the difference between expected value and the sample mean.

[3]:
```
ex_v = -1*0.5 + 0.5*1
print("Expected Value: " + str(ex_v))
```

```
var = (0.5 * 1) + (0.5 *1) - (0**2)
print("Variance: " + str(var))
```

```
Expected Value: 0.0
Variance: 1.0
```

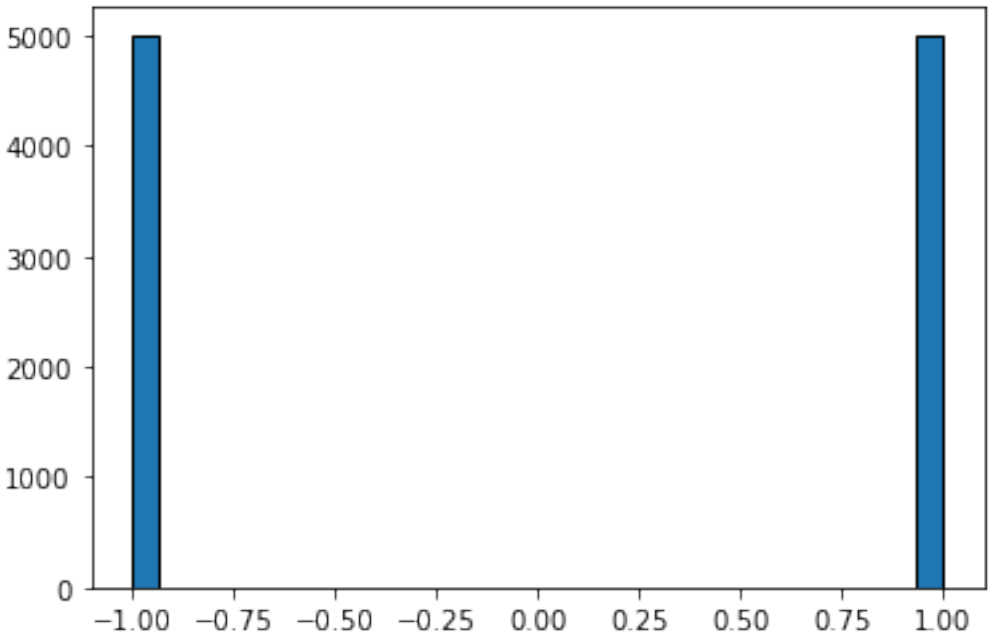
The difference between expected value and the sample mean is that the expected value is a theoretical value calculated using the probability and the denoted values, while sample mean is just the average of a realization. Key difference is just one is theoretical and the other one is calulated.

1.3. Choose your number of repetitions R. 1000 is a good number but you can also take 10,000 or 100,000 to get smoother histograms.

[4]:
```
R = 10000
```

1.4. Create a vector of R random realizations of X. Make a histogram of those. Comment the shape of the histogram.

[5]:
```
S = 1
np.random.seed(1)
realizations = []
for i in range(R):
    realizations.append(np.random.randint(0,2, size=S)[0]*2-1)
plt.hist(realizations, bins=30, edgecolor="k")
plt.show()
```



The histogram just has two bars, one for each value stated in the RV.

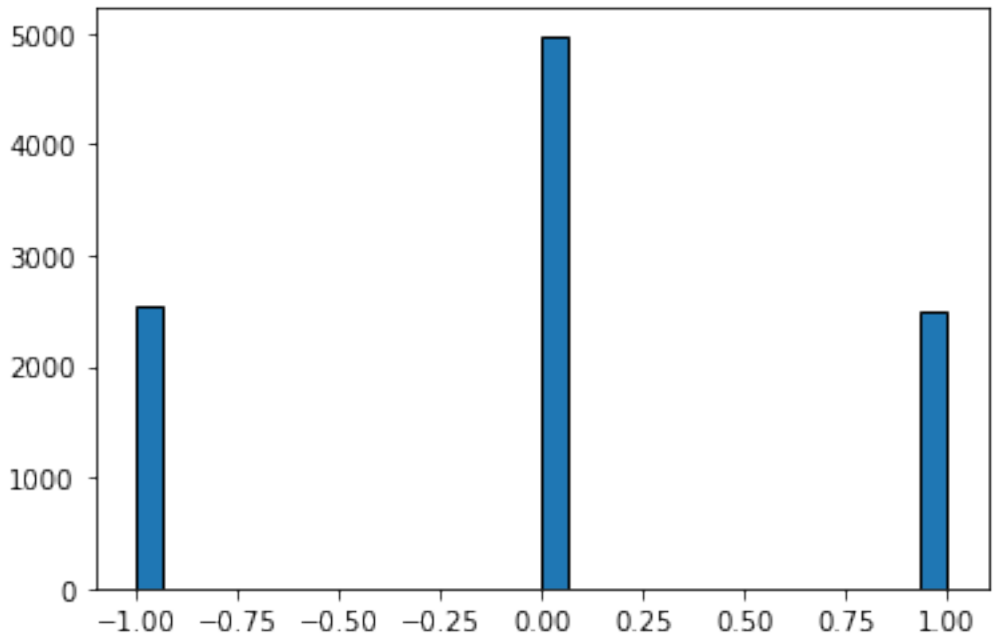
1.5. Compute and report mean and variance of the sample you created (just use np.mean and np.var). NB! Here we talk about sample mean and sample variance. Compare these numbers with the theoretical values computed in question 2 above.

[6]:
```
s_mean = np.mean(realizations)
print("Sample Mean: " + str(s_mean))
s_var = np.var(realizations)
print("Sample Variance: " + str(s_var) + "\n")
print("Difference between Expected Value & Sample Mean: " + str(ex_v - s_mean))
print("Difference between Variance & Sample Variance: " + str(var - s_var))
```

```
Sample Mean: -0.0012
Sample Variance: 0.9999985600000002

Difference between Expected Value & Sample Mean: 0.0012
Difference between Variance & Sample Variance: 1.439999999797159e-06
```

1.6. Now create R pairs of random realizations of X (i.e. sample size S = 2). For each pair, compute its mean. You should have R mean values. Make the histogram. How does this look like?

[7]:
```
S = 2
np.random.seed(1)
pair_means = []
for i in range(R):
    pair_means.append(np.mean(np.random.randint(0,2, size=S)*2-1))
plt.hist(pair_means, bins=30, edgecolor="k")
plt.show()
```



The shape looks like a normal symmetrical distribution.

1.7. Compute and report mean of the R pair means, and variance of the means. NB! we talk about sample mean and sample variance again, where sample is your sample of R pair means.

[8]:
```
p_mean = np.mean(pair_means)
print("Sample Mean: " + str(p_mean))
p_var = np.var(pair_means)
print("Sample Variance: " + str(p_var) + "\n")
```

```
Sample Mean: -0.005
Sample Variance: 0.502175
```

1.8. Compute the expected value and variance of the pair means, i.e. the theoretical concepts. This mirrors what you did in 2.

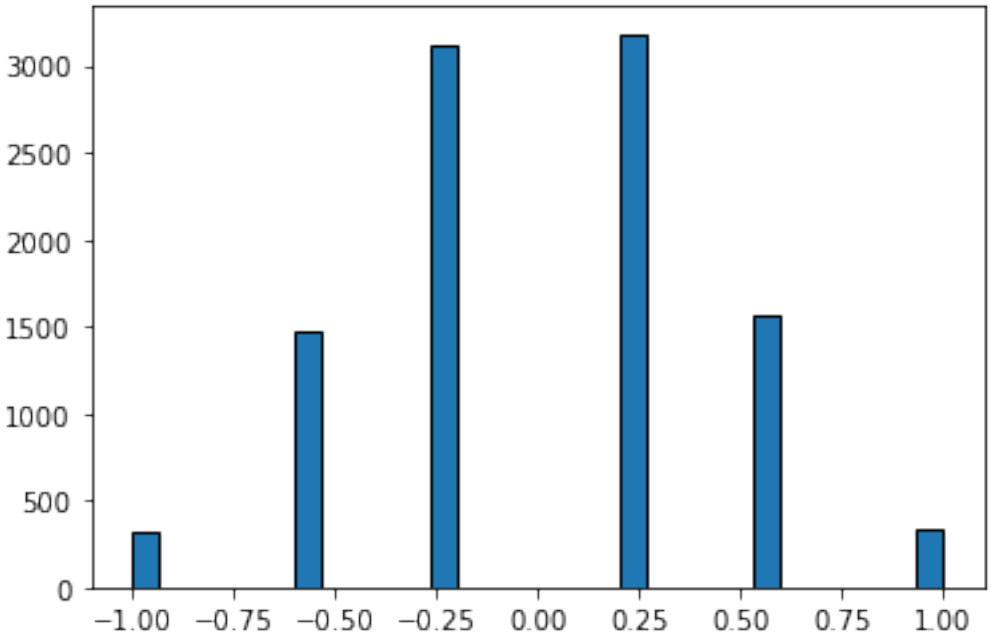
[9]:
```
print("Expected Value stays the same at " + str(ex_v))
print("Variance of Pair Means is VarX * 1/S, which is " + str(var/2))
```

```
Expected Value stays the same at 0.0
Variance of Pair Means is VarX * 1/S, which is 0.5
```

1.9. Now instead of pairs of random numbers, repeat this with 5-tuples of random numbers (i.e. S = 5 random numbers per one repetition, and still the same R = 1000 or whatever you chose repetitions in total). Compare the theoretical and sample version of mean and variance of 5-tuples. Are they similar? Do you spot any noticeable differences in the histogram compared to your previous histogram?

[10]:
```
makehist(5)
```

```
100
```



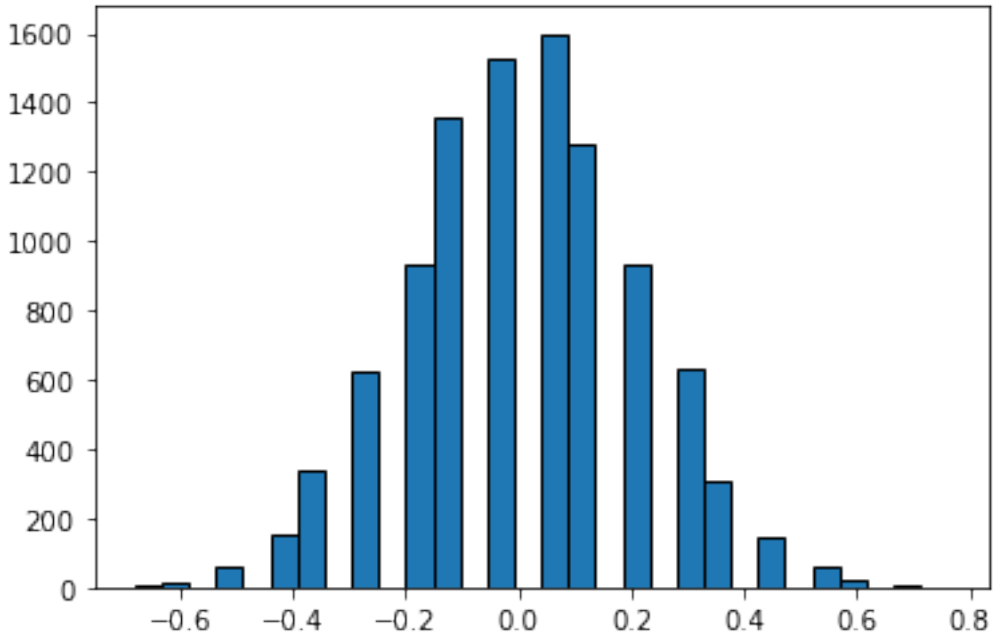

```
Sample Mean: 0.007519999999999999
Sample Variance: 0.2002634496

Expected Value stays the same at 0.0
Variance of Pair Means is VarX * 1/S, which is 0.2

Difference between Expected Value & Sample Mean: -0.007519999999999999
Difference between Variance & Sample Variance: -0.00026344959999999307
```

The histogram is still normal and symmetrical, but there are more possible values compared to when sample size is 2.

1.10. Repeat with 25-tuples... (Also compute the expectation and theoretical variance, and compare those with sample mean, sample variance)

[11]:
```
makehist(25)
```

```
100
```

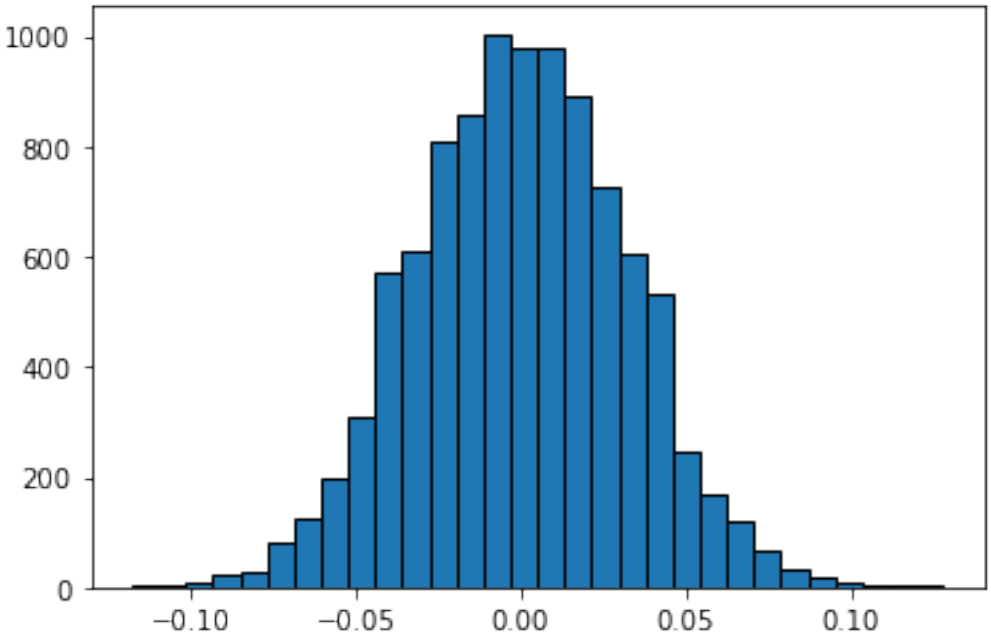



```
Sample Mean: -0.0016720000000000003
Sample Variance: 0.040694804416000006

Expected Value stays the same at 0.0
Variance of Pair Means is VarX * 1/S, which is 0.04

Difference between Expected Value & Sample Mean: 0.0016720000000000003
Difference between Variance & Sample Variance: -0.0006948044160000055
```

1.11. and with 1000-tuples. Do not forget to compare with theoretical results.

[12]:
```
makehist(1000)
```

```
100
```




```
Sample Mean: 3.9400000000000056e-05
Sample Variance: 0.00100089564764

Expected Value stays the same at 0.0
Variance of Pair Means is VarX * 1/S, which is 0.001

Difference between Expected Value & Sample Mean: -3.9400000000000056e-05
Difference between Variance & Sample Variance: -8.956476399998945e-07
```

1.12. Comment on the tuple size, and how the shape of the histogram changes when the tuple size increases. The histogram is still normal and symmetrical, but there are more possible values compared to when sample size is smaller, hence there being more bars in the histogram.

1.13. Explain why do the histograms resemble normal distribution as S grows. In particular, explain what happens when we move from single values S = 1 to pairs S = 2. Why did two equal peaks turn into a " "-shaped histogram? According to Central Limit Theorem, it is more likely for the results to be near the middle number than the edge numbers, which means that the values will pile up in the center as it is less likely for all of the values to be 0 or all of them to be 1. At S = 1, there is an equal chance of the toss being heads as well as tails, thats why it's two peaks, but at S = 2, it is more likely that the tosses would be 1 head and 1 tail, rather than both heads or both tails, which is why it turns into a " "-shaped histogram.

1.14. Explain what is the difference between R and S. How do changing these values affect the histograms? R would is the number of realizations we coduct while S is the size of each realization. S effects the shape of the histogram, as S increases the histogram becomes more

normally distributed. As R increases, we would get a smoother histogram as we get more data to plot.

0.2 2. Is povery in Azraq refugee camp falling?

0.2.1 2.1. Background

2.1.1. What was the abject poverty in Azraq camp in Q1 and Q2 2022 (when including all assistance)? Lets call these variables p1 and p2.

[13]:
```
p1 = 0.66
p2 = 0.51
```

For Q1 of 2022 the abject poverty line is 66%, while for Q2, the abject poverty line is 51%

2.1.2. How many households were surveyed in the camp? Call this sample size S.

[14]:
```
S = 325
```

There are 325 households surveyed in the Azraq Camp.

0.2.2 2.2. Simulations

2.2.1. Create a random sample using the correct values of S and p2 you found in 2.1 above.

[15]:
```
np.random.seed(1)
sample = np.random.binomial(1, p2, size=S)
```

2.2.2 Compute the sample mean and compare it with p1 and p2 above. How close is it to these figures?

[16]:
```
mean = np.mean(sample)
print("Sample Mean: " + str(mean))
print("Difference from p1: " + str(p1-mean))
print("Difference from p2: " + str(p2-mean))
```

```
Sample Mean: 0.47384615384615386
Difference from p1: 0.18615384615384617
Difference from p2: 0.03615384615384615
```

The sample mean is closer to P2

2.2.3. Pick your number of replications R (something like 1000 or 10,000 are good numbers).

[17]:
```
R = 10000
```

2.2.4. Repeat the points 1 and 2 for R times: create the sample, compute the average, but also store the average in an array. You should have R averages now.

[18]:
```
means = []
np.random.seed(1)
for i in range(R):
    means.append(np.mean(np.random.binomial(1, p2, size=S)))
```

2.2.5. What is the average of the averages? Which probability from 1 does it resemble? Why?

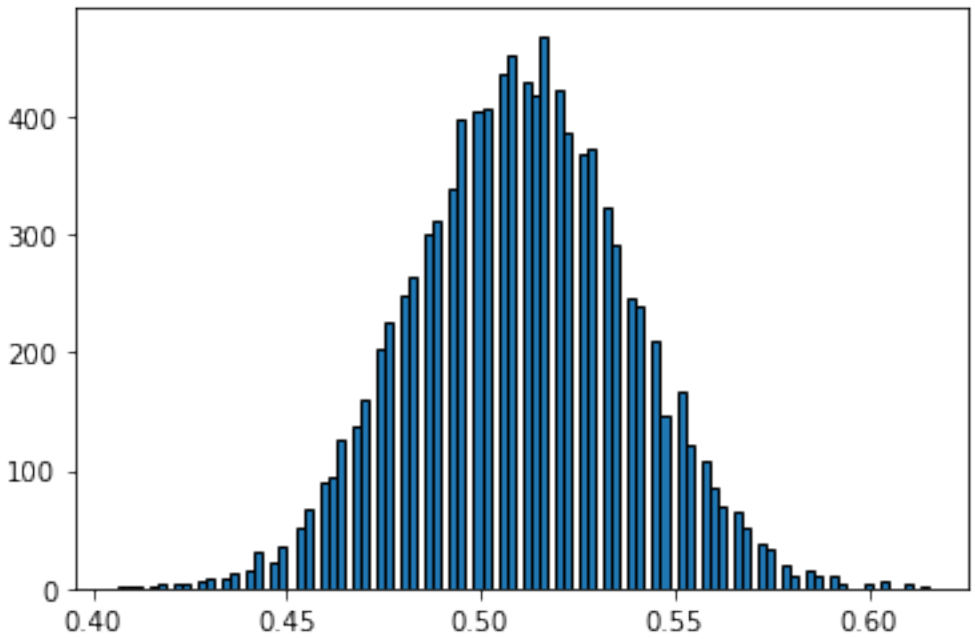
[19]:
```
m_mean = np.mean(means)
print("Mean: " + str(m_mean))
print("Difference from p1: " + str(p1-m_mean))
print("Difference from p2: " + str(p2-m_mean))
```

```
Mean: 0.5101381538461538
Difference from p1: 0.14986184615384623
Difference from p2: -0.0001381538461537879
```

It is closer to p2, this is because we used the probability of p2 as the probability of the binomial that has the value of 1 and p1 as the probability of the event with value of 0, which means that the expected value would be p2.

2.2.6. Plot a histogram of the averages. Which distribution does it resemble? What do you say, by just eyeballing the plot, what are the largest and smallest values that are "reasonably" common?

[20]:
```
plt.hist(means, bins = 100, edgecolor = "k")
plt.show()
```



The smallest reasonably common value would be around 0.48 while the largest would be around 0.53.

2.2.7. Finally, compute 2.5th and 97.5th percentile and the 95% confidence intervals. Does the Q1 poverty value fall into this interval?

[21]:
```
p_25, p_975 = np.percentile(means, (2.5, 97.5))
print("Confidence Interval: " + str([p_25, p_975]))
```

```
Confidence Interval: [0.4553846153846154, 0.5661538461538461]
```

The Q1 Value does not fall into this interval.

0.2.3 2.3 Theoretical CI

2.3.1. Compute variance of your sample of X. You can use the Bernoulli variance formula Var X = p(1 − p). You can also use the sample you created in 2.2.1, or create a new sample, and find the sample variance.

[22]:
```
var_x = p2*(1 - p2)
print("Variance: " + str(var_x))
```

```
Variance: 0.2499
```

2.3.2. But this was variance of X (or sample variance if that was what you computed). What we need is variance of sample mean. What does CLT tell about relationship b/w sample variance and variance of the sample mean? Based on CLT, variance of the sample mean = 1/s * sample variance

[23]:
```
var_smean = var_x/S
print("Variance of Sample Mean: " + str(var_smean))
```

```
Variance of Sample Mean: 0.000768923076923077
```

2.3.3. Compute the standard deviation of sample mean using CLT Standard deviation is the sqrt of the variance of the sample mean

[24]:
```
print("Standard Deviation of Sample Mean (CLT): " + str(math.sqrt(var_smean)))
```

```
Standard Deviation of Sample Mean (CLT): 0.02772946225448804
```

2.3.4. Compare the standard deviation you got here with the standard deviation of the sample of averages you computed in 2.2.4.

[25]:
```
std_smean = np.std(means)
print("Standard Deviation of Sample Mean (np.var): " + str(std_smean))
print("Difference between CLT and np.var: " + str(math.sqrt(var_smean) -␣
 ↪std_smean))
```

```
Standard Deviation of Sample Mean (np.var): 0.02803481081899439
Difference between CLT and np.var: -0.00030534856450634834
```

2.3.5. Use this standard deviation to compute the confidence interval. Compare it to what you got in 2.2.7. Does p1 falls inside or outside of this interval?

[27]:
```
q_25 = 0.51-1.96*math.sqrt(var_smean)
q_975 = 0.51+1.96*math.sqrt(var_smean)
print("Confidence Interval: " + str([q_25, q_975]))
```

```
Confidence Interval: [0.45565025398120346, 0.5643497460187966]
```

No, P1 does not fall inside this interval.