Sampling Distributions

November 29, 2017

0.0.1 Sampling Distributions Introduction

In order to gain a bit more comfort with this idea of sampling distributions, let's do some practice in python.

Below is an array that represents the students we saw in the previous videos, where 1 represents the students that drink coffee, and 0 represents the students that do not drink coffee.

```
In [1]: import numpy as np
        np.random.seed(42)

        students = np.array([1,0,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,0])
```

1. Find the proportion of students who drink coffee in the above array. Store this value in a variable **p**.

```
In [22]: p = (students == 1).mean()
         p

Out[22]: 0.7142857142857143
```

2. Use numpy's **random.choice** to simulate 5 draws from the `students` array. What is proportion of your sample drink coffee?

```
In [23]: np.random.choice(students, size=5).mean()

Out[23]: 0.80000000000000004
```

3. Repeat the above to obtain 10,000 additional proportions, where each sample was of size 5. Store these in a variable called `sample_props`.

```
In [24]: sample_props = []
         for _ in range(100000):
             sample_props.append(np.random.choice(students, size=5).mean())
```

4. What is the mean proportion of all 10,000 of these proportions? This is often called **the mean of the sampling distribution**.

```
In [25]: np.mean(sample_props)

Out[25]: 0.71425399999999994
```

5. What are the variance and standard deviation for the original 21 data values?

```
In [26]: np.var(students), np.std(students)
```

```
Out[26]: (0.20408163265306126, 0.45175395145262565)
```

6. What are the variance and standard deviation for the 10,000 proportions you created?

```
In [27]: np.var(sample_props), np.std(sample_props)
```

```
Out[27]: (0.04079202348400001, 0.20197035298280788)
```

7. Compute p(1-p), which of your answers does this most closely match?

```
In [29]: p*(1-p)
```

```
Out[29]: 0.20408163265306123
```

this is similar to the variance of the student population

8. Compute p(1-p)/n, which of your answers does this most closely match?

```
In [30]: n = 5
         p*(1-p)/n
```

```
Out[30]: 0.040816326530612249
```

this is similar to the variance of the large sample base

9. Notice that your answer to 8. is commonly called the **variance of the sampling distribution**. If you were to change your first sample to be 20, what would this do for the variance of the sampling distribution? Simulate and calculate the new answers in 6. and 8. to check that the consistency you found before still holds.

```
In [31]: n2 = 20
         p2 = np.random.choice(students, size=n2).mean()
         p2
```

```
Out[31]: 0.69999999999999996
```

```
In [32]: p2*(1-p2)/n2
```

```
Out[32]: 0.010500000000000001
```

```
In [33]: sp2 = []
         for _ in range(100000):
             sp2.append(np.random.choice(students, size=n2).mean())
```

```
In [34]: np.var(sp2), np.std(sp2)
```

```
Out[34]: (0.010197795547750001, 0.10098413512898945)
```

yes, it seems it basically still holds

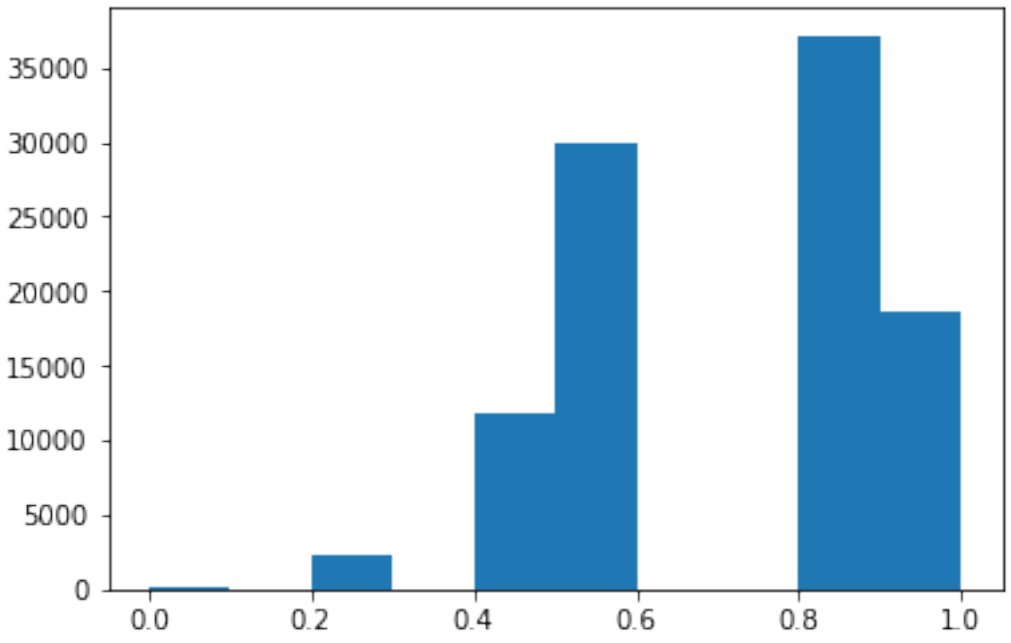
```
In [ ]: ##Simulate your 20 draws
```

```
In [ ]: ##Compare your variance values as computed in 6 and 8,
        ##but with your sample of 20 values
```

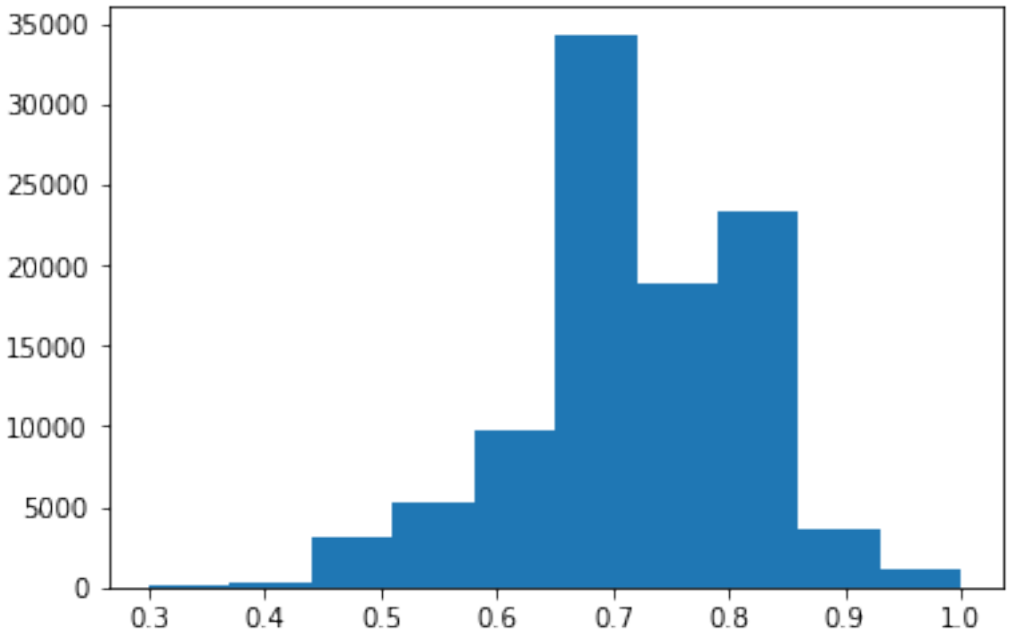
10. Finally, plot a histgram of the 10,000 draws from both the proportions with a sample size of 5 and the proportions with a sample size of 20. Each of these distributions is a sampling distribution. One is for the proportions of sample size 5 and the other a sampling distribution for proportions with sample size 20.

```
In [37]: import matplotlib.pyplot as plt
         %matplotlib inline
```

```
In [38]: plt.hist(sample_props);
```




```
In [39]: plt.hist(sp2);
```


35000
30000
25000
20000
15000
10000
5000
0
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0