Sampling with Python

Throughout the following exercises, we will learn to use Python to simulate random sampling and make a point estimate of a population mean based on our sample data.

Sampling in statistics is the process of selecting a subset of individuals or items from a larger population to estimate characteristics of the whole population. It's a crucial method because it allows researchers to make inferences about a population without having to study every individual or item within it.

As we move forward, we can find instructions on how to install required libraries as they arise in this notebook. Before we begin with the exercises and analyzing the data, we need to import all libraries and extensions required for this programming exercise. Throughout the course, we will be using numpy, pandas, scipy stats, and statsmodels for operations, and matplotlib for plotting.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
import statsmodels.api as sm
```

In [2]:
```
education_districtwise = pd.read_csv('education_districtwise.csv')
education_districtwise = education_districtwise.dropna()
```

We'll continue with our previous scenario, in which I am a data professional working for the Department of Education of a large nation. Recall that we are analyzing data on the literacy rate for each district.

Now imagine that we are asked to *collect* the data on district literacy rates, and that we have limited time to do so. We can only survey 50 randomly chosen districts, instead of the 634 districts included in our original dataset. The goal of the research study is to estimate the mean literacy rate for *all* 634 districts based on our sample of 50 districts.

Simulate random sampling

We can use Python to simulate taking a random sample of 50 districts from your dataset. To do this, use `pandas.DataFrame.sample()` . The following arguments in the `sample()` function will help us simulate random sampling:

- `n` : Refers to the desired sample size
- `replace` : Indicates whether we are sampling with or without replacement
- `random_state` : Refers to the seed of the random number

Reference: [https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sample.html (https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sample.html)](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sample.html).

Note: A **random seed** is a starting point for generating random numbers. We can use any arbitrary number to fix the random seed, and give the random number generator a starting point. Also, going forward, we can use the same random seed to generate the same set of numbers.

Now we're ready to write the code. First, name a new variable `sampled_data` . Then, set the arguments for the `sample()` function:

- `n` : We're sampling from 50 districts, so our sample size is `50` .
- `replace` : For the purpose of our example, we'll sample *with* replacement. `True` indicates sampling with replacement.
- `random_state` : Choose an arbitrary number for the random seed. Say, `31208` .

In [3]:
```
sampled_data = education_districtwise.sample(n=50, replace=True, random_state=
sampled_data
```

Out[3]:

	DISTNAME	STATNAME	BLOCKS	VILLAGES	CLUSTERS	TOTPOPULAT	OVERALL_LI
661	DISTRICT528	STATE6	9	112	89	1863174.0	92.14
216	DISTRICT291	STATE28	14	1188	165	3273127.0	52.49
367	DISTRICT66	STATE23	12	1169	116	1042304.0	62.14
254	DISTRICT458	STATE3	3	157	19	82839.0	76.33
286	DISTRICT636	STATE35	3	187	44	514683.0	86.70
369	DISTRICT512	STATE23	6	589	30	717169.0	68.35
258	DISTRICT156	STATE3	6	80	9	35289.0	59.94
10	DISTRICT412	STATE1	11	187	95	476820.0	68.69
512	DISTRICT277	STATE9	10	558	179	2298934.0	84.31
144	DISTRICT133	STATE21	14	1672	136	3673849.0	69.61
325	DISTRICT1	STATE33	4	534	98	957853.0	69.37
227	DISTRICT159	STATE28	18	870	134	2954367.0	66.23
86	DISTRICT667	STATE25	5	396	75	896129.0	82.23
425	DISTRICT144	STATE31	7	1064	108	2662077.0	71.59
260	DISTRICT305	STATE3	2	62	6	145538.0	69.88
281	DISTRICT385	STATE35	6	531	30	354972.0	75.00
262	DISTRICT552	STATE3	3	103	4	111997.0	52.23
253	DISTRICT168	STATE3	5	312	16	176385.0	82.14
301	DISTRICT551	STATE14	9	103	63	693281.0	88.29
356	DISTRICT494	STATE34	25	2179	223	3596292.0	70.95
165	DISTRICT196	STATE21	10	1354	119	1795092.0	77.52
565	DISTRICT308	STATE17	8	721	144	848868.0	86.54
388	DISTRICT281	STATE23	6	392	58	949159.0	73.92
461	DISTRICT619	STATE22	5	859	57	1064989.0	68.36
384	DISTRICT455	STATE23	9	1217	55	1063458.0	68.85
590	DISTRICT70	STATE20	7	427	84	1846993.0	80.30
343	DISTRICT354	STATE33	2	192	46	1260419.0	88.66
539	DISTRICT440	STATE17	15	1465	167	2887826.0	88.23
459	DISTRICT431	STATE22	9	1778	143	2363744.0	73.42
667	DISTRICT123	STATE11	3	80	16	237586.0	88.49
387	DISTRICT231	STATE23	6	657	63	530299.0	64.51
306	DISTRICT37	STATE4	7	1083	92	642923.0	68.38
213	DISTRICT347	STATE28	11	623	94	2228397.0	59.65
97	DISTRICT22	STATE2	7	182	7	2531583.0	87.12
78	DISTRICT247	STATE25	7	314	60	1332042.0	72.73
394	DISTRICT640	STATE24	17	1857	191	1802777.0	69.00

	DISTNAME	STATNAME	BLOCKS	VILLAGES	CLUSTERS	TOTPOPULAT	OVERALL_LI
184	DISTRICT596	STATE21	11	1281	108	2149066.0	51.76
147	DISTRICT335	STATE21	17	1945	138	4380793.0	69.44
542	DISTRICT489	STATE17	7	749	63	1198810.0	85.14
105	DISTRICT157	STATE13	14	1994	508	3671999.0	71.68
254	DISTRICT458	STATE3	3	157	19	82839.0	76.33
109	DISTRICT158	STATE13	6	769	211	1338114.0	66.19
609	DISTRICT17	STATE20	4	359	59	9588910.0	88.48
53	DISTRICT126	STATE26	3	197	21	596294.0	68.90
81	DISTRICT45	STATE25	9	351	130	1742815.0	73.24
516	DISTRICT300	STATE9	5	651	84	590379.0	73.29
641	DISTRICT484	STATE6	15	333	83	1721179.0	74.92
650	DISTRICT145	STATE6	11	489	100	1614069.0	84.09
70	DISTRICT99	STATE25	4	279	43	558890.0	83.44
163	DISTRICT366	STATE21	9	1330	86	1579160.0	79.99

The output shows 50 districts selected randomly from our dataset. Each has a different literacy rate, but note that row 254 was sampled twice, which is possible because we sampled with replacement.

Compute the sample mean

Now that we have our random sample, use the mean function to compute the sample mean. First, name a new variable `estimate1`. Next, use `mean()` to compute the mean for our sample data.

In [4]:
```
estimate1 = sampled_data['OVERALL_LI'].mean()
estimate1
```

Out[4]:
```
74.22359999999999
```

The sample mean for district literacy rate is about 74.22%. This is a point estimate of the population mean based on our random sample of 50 districts. Remember that the population mean is the literacy rate for *all* districts. Due to sampling variability, the sample mean is usually not exactly the same as the population mean.

Next, let's find out what will happen if we compute the sample mean based on another random sample of 50 districts.

To generate another random sample, name a new variable `estimate2`. Then, we set the arguments for the sample function. Once again, `n` is `50` and `replace` is "True." This time, choose a different number for the random seed to generate a different sample: 56,810. Finally, add `mean()` at the end of the line of code to compute the sample mean.

In [5]:
```
estimate2 = education_districtwise['OVERALL_LI'].sample(n=50, replace=True, ran
estimate2
```

Out[5]:
```
74.24780000000001
```

For our second estimate, the sample mean for district literacy rate is about 74.25%.

Due to sampling variability, this sample mean is different from the sample mean of our previous estimate, 74.22% – but they're really close.

The central limit theorem

Recall that the **central limit theorem** tells us that when the sample size is large enough, the sample mean approaches a normal distribution. And, as we sample more observations from a population, the sample mean gets closer to the population mean. The larger your sample size, the more accurate our estimate of the population mean is likely to be.

In this case, the population mean is the overall literacy rate for *all* districts in the nation. Earlier, we found that the population mean literacy rate is 73.39%. Based on sampling, we first estimated sample mean was 74.22%, and our second estimate was 74.24%. Each estimate is relatively close to the population mean.

Compute the mean of a sampling distribution with 10,000 samples

Now, imagine we repeat the study 10,000 times and obtain 10,000 point estimates of the mean. In other words, we take 10,000 random samples of 50 districts, and compute the mean for each sample. According to the central limit theorem, the mean of our sampling distribution will be roughly equal to the population mean.

We can use Python to compute the mean of the sampling distribution with 10,000 samples.

Let's go over the code step by step:

1. Create an empty list to store the sample mean from each sample. Name this `estimate_list`.
2. Set up a for-loop with the `range()` function. The `range()` function generates a sequence of numbers from 1 to 10,000. The loop will run 10,000 times, and iterate over each number in the sequence.
3. Specify what you want to do in each iteration of the loop. The `sample()` function tells the computer to take a random sample of 50 districts with replacement–the argument `n` equals `50`, and the argument `replace` equals `True`. The `append()` function adds a single item to an existing list. In this case, it appends the value of the sample mean to each item in the list. Your code generates a list of 10,000 values, each of which is the sample mean from a random sample.
4. Create a new data frame for your list of 10,000 estimates. Name a new variable `estimate_df` to store your data frame.

In [6]:
```
estimate_list = []
for i in range(10000):
    estimate_list.append(education_districtwise['OVERALL_LI'].sample(n=50, rep
estimate_df = pd.DataFrame(data={'estimate': estimate_list})
```

Note that, because we didn't specify a random seed for each loop iteration, by default the rows sampled will be different each time.

Now, name a new variable `mean_sample_means` and compute the mean for our sampling distribution of 10,000 random samples.

In [7]:
```
mean_sample_means = estimate_df['estimate'].mean()
mean_sample_means
```

Out[7]:
```
73.37863700000047
```

The mean of our sampling distribution is about 73.4%.

Compare this with the population mean of our complete dataset:

In [8]:
```
population_mean = education_districtwise['OVERALL_LI'].mean()
population_mean
```

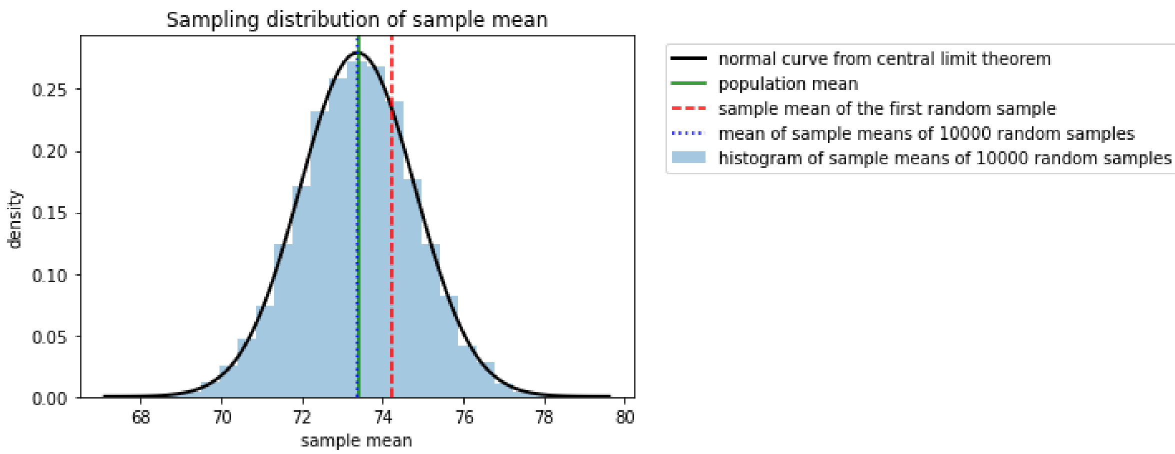
Out[8]:
```
73.39518927444797
```

The mean of our sampling distribution is essentially identical to the population mean, which is also about 73.4%!

Visualize your data

To visualize the relationship between our sampling distribution of 10,000 estimates and the normal distribution, we can plot both at the same time.

In [9]:
```
plt.hist(estimate_df['estimate'], bins=25, density=True, alpha=0.4, label = "hi
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100) # generate a grid of 100 values from xmin to x
p = stats.norm.pdf(x, mean_sample_means, stats.tstd(estimate_df['estimate']))
plt.plot(x, p,'k', linewidth=2, label = 'normal curve from central limit theore
plt.axvline(x=population_mean, color='g', linestyle = 'solid', label = 'popula
plt.axvline(x=estimate1, color='r', linestyle = '--', label = 'sample mean of t
plt.axvline(x=mean_sample_means, color='b', linestyle = ':', label = 'mean of s
plt.title("Sampling distribution of sample mean")
plt.xlabel('sample mean')
plt.ylabel('density')
plt.legend(bbox_to_anchor=(1.04,1))
plt.show()
```



There are three key takeaways from this graph:

1. As the central limit theorem predicts, the histogram of the sampling distribution is well approximated by the normal distribution. The outline of the histogram closely follows the normal curve.
2. The mean of the sampling distribution, the blue dotted line, overlaps with the population mean, the green solid line. This shows that the two means are essentially equal to each other.
3. The sample mean of your first estimate of 50 districts, the red dashed line, is farther away from the center. This is due to sampling variability.

The central limit theorem shows that as we increase the sample size, our estimate becomes more accurate. For a large enough sample, the sample mean closely follows a normal distribution.

Your first sample of 50 districts estimated the mean district literacy rate as 74.22%, which is relatively close to the population mean of 73.4%.

To ensure our estimate will be useful to the government, we can compare the nation’s literacy rate to other benchmarks, such as the global literacy rate, or the literacy rate of peer nations. If the nation’s literacy rate is below these benchmarks, this may help convince the government to devote more resources to improving literacy across the country.