**1. Introduction to bootstrapping**

So far, we've mostly focused on the idea of sampling without replacement.

**2. With or without**

00:05 - 00:31

Sampling without replacement is like dealing a pack of cards. When we deal the ace of spades to one player, we can't then deal the ace of spades to another player. Sampling with replacement is like rolling dice. If we roll a six, we can still get a six on the next roll. Sampling with replacement is sometimes called resampling. We'll use the terms interchangeably.

**3. Simple random sampling without replacement**

00:31 - 00:41

If we take a simple random sample without replacement, each row of the dataset, or each type of coffee, can only appear once in the sample.

**4. Simple random sampling with replacement**

00:41 - 00:49

If we sample with replacement, it means that each row of the dataset, or each coffee, can be sampled multiple times.

**5. Why sample with replacement?**

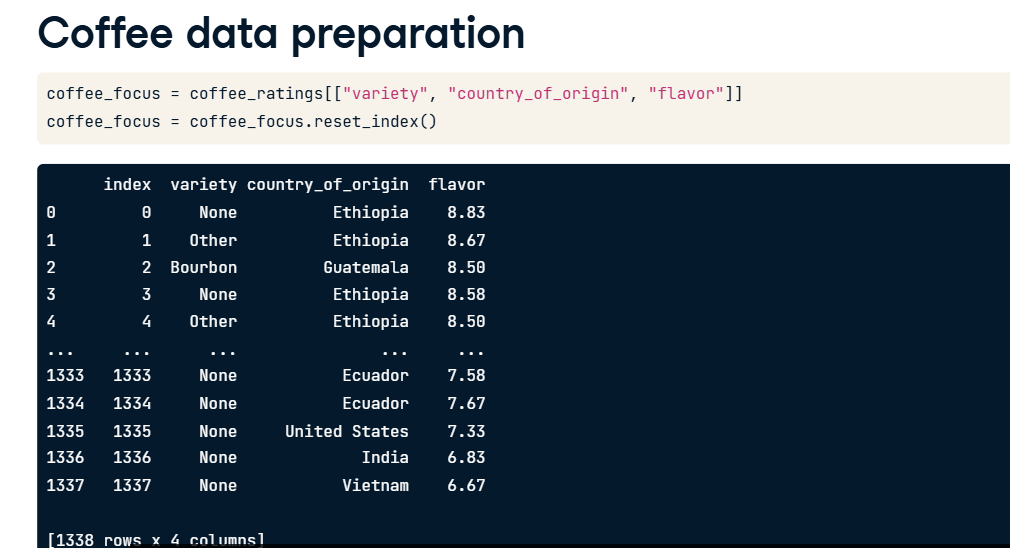
00:49 - 01:26

So far, we've been treating the coffee\_ratings dataset as the population of all coffees. Of course, it doesn't include every coffee in the world, so we could treat the coffee dataset as just being a big sample of coffees. To imagine what the whole population is like, we need to approximate the other coffees that aren't in the dataset. Each of the coffees in the sample dataset will have properties that are representative of the coffees that we don't have. Resampling lets us use the existing coffees to approximate those other theoretical coffees.

**6. Coffee data preparation**

01:26 - 01:40

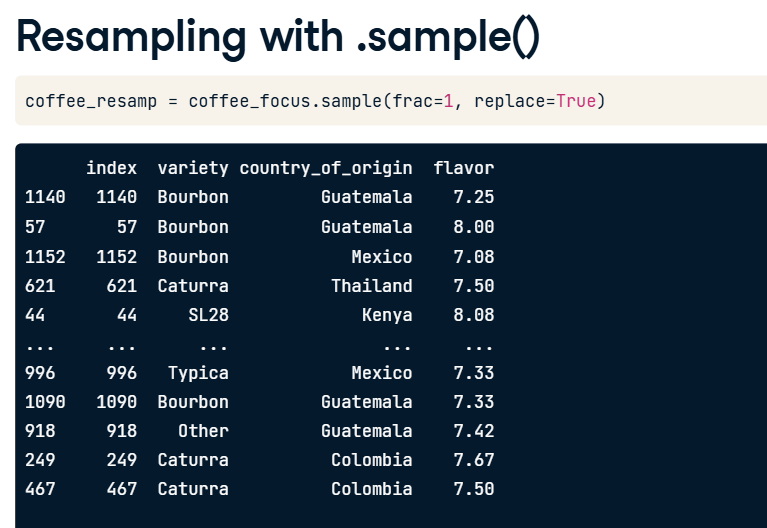
To keep it simple, let's focus on three columns of the coffee dataset. To make it easier to see which rows ended up in the sample, we'll add a row index column called index using the reset\_index method.



**7. Resampling with .sample()**

01:40 - 01:52

To sample with replacement, we call sample as usual but set the replace argument to True. Setting frac to 1 produces a sample of the same size as the original dataset.



**8. Repeated coffees**

01:52 - 02:03

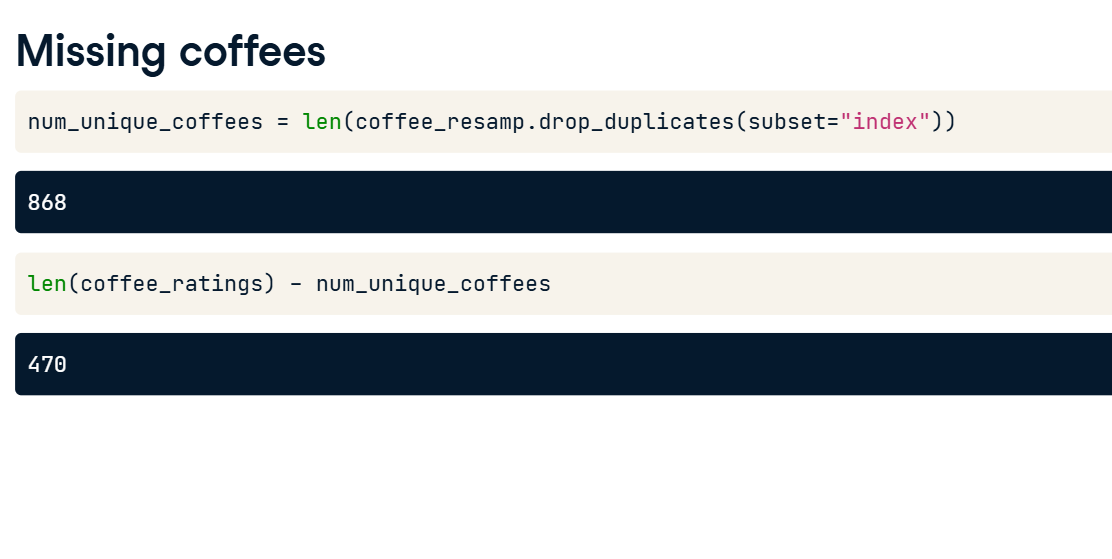
Counting the values of the index column shows how many times each coffee ended up in the resampled dataset. Some coffees were sampled four or five times.



**9. Missing coffees**

02:03 - 02:27

That means that some coffees didn't end up in the resample. By taking the number of distinct index values in the resampled dataset, using len on drop\_duplicates, we see that eight hundred and sixty-eight different coffees were included. By comparing this number with the total number of coffees, we can see that four hundred and seventy coffees weren't included in the resample.



**10. Bootstrapping**

02:27 - 03:03

We're going to use resampling for a technique called bootstrapping. In some sense, bootstrapping is the opposite of sampling from a population. With sampling, we treat the dataset as the population and move to a smaller sample. With bootstrapping, we treat the dataset as a sample and use it to build up a theoretical population. A use case of bootstrapping is to try to understand the variability due to sampling. This is important in cases where we aren't able to sample the population multiple times to create a sampling distribution.

**11. Bootstrapping process**

03:03 - 03:49

The bootstrapping process has three steps. First, randomly sample with replacement to get a resample the same size as the original dataset. Then, calculate a statistic, such as a mean of one of the columns. Note that the mean isn't always the choice here and bootstrapping allows for complex statistics to be computed, too. Then, replicate this many times to get lots of these bootstrap statistics. Earlier in the course, we did something similar. We took a simple random sample, then calculated a summary statistic, then repeated those two steps to form a sampling distribution. This time, when we've used resampling instead of sampling, we get a bootstrap distribution.

**12. Bootstrapping coffee mean flavor**

03:49 - 04:14

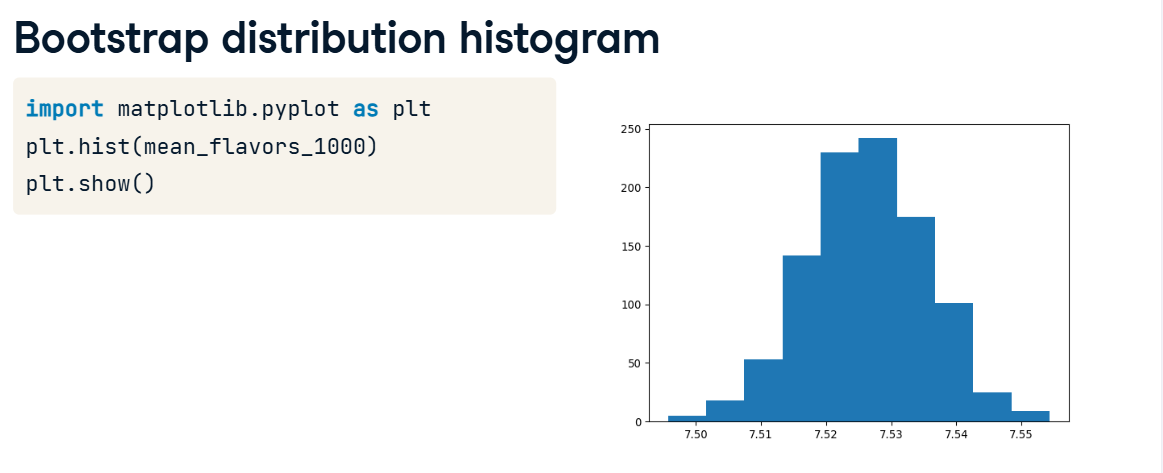
The resampling step uses the code we just saw: calling sample with frac set to one and replace set to True. Calculating a bootstrap statistic can be done with mean from NumPy. In this case, we're calculating the mean flavor score. To repeat steps one and two one thousand times, we can wrap the code in a for loop and append the statistics to a list.



**13. Bootstrap distribution histogram**

04:14 - 04:22

Here's a histogram of the bootstrap distribution of the sample mean. Notice that it is close to following a normal distribution.

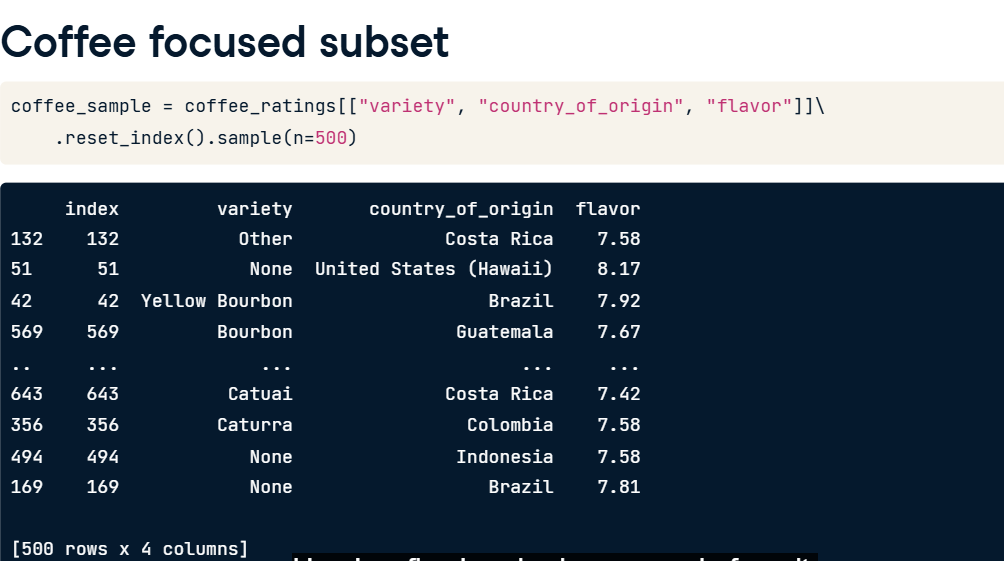


**Comparing sampling and bootstrap distributions**

we took a focused subset of the coffee dataset. Here's a five hundred row sample from it.

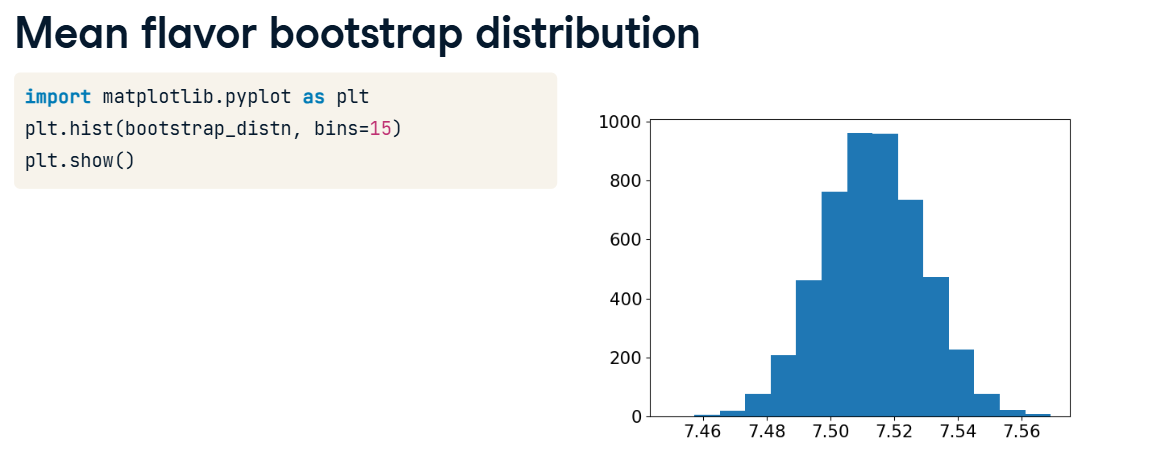
**3. The bootstrap of mean coffee flavors**

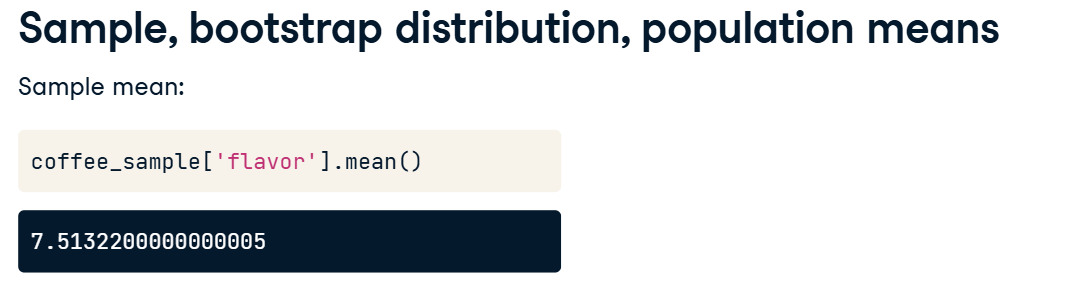
Here, we generate a bootstrap distribution of the mean coffee flavor scores from that sample. dot-sample generates a resample, np-dot-mean calculates the statistic, and the for loop with dot-append repeats these steps to produce a distribution of bootstrap statistics.



**4. Mean flavor bootstrap distribution**

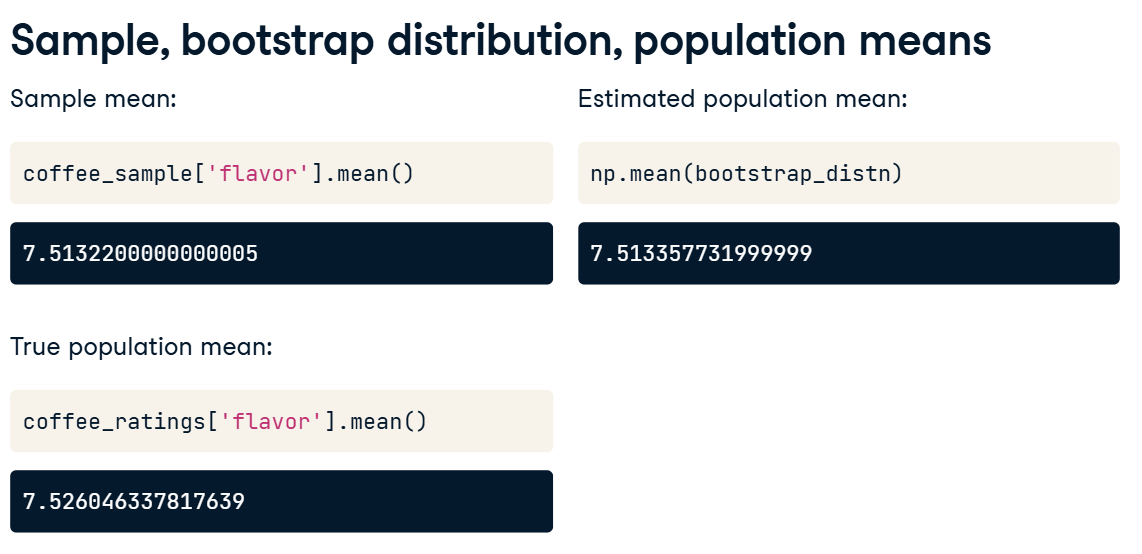
Here's the histogram of the bootstrap distribution, which is close to a normal distribution.





**5. Sample, bootstrap distribution, population means**

Here's the mean flavor score from the original sample. In the bootstrap distribution, each value is an estimate of the mean flavor score. Recall that each of these values corresponds to one potential sample mean from the theoretical population. If we take the mean of those means, we get our best guess of the population mean. The two values are really close. However, there's a problem. The true population mean is actually a little different.

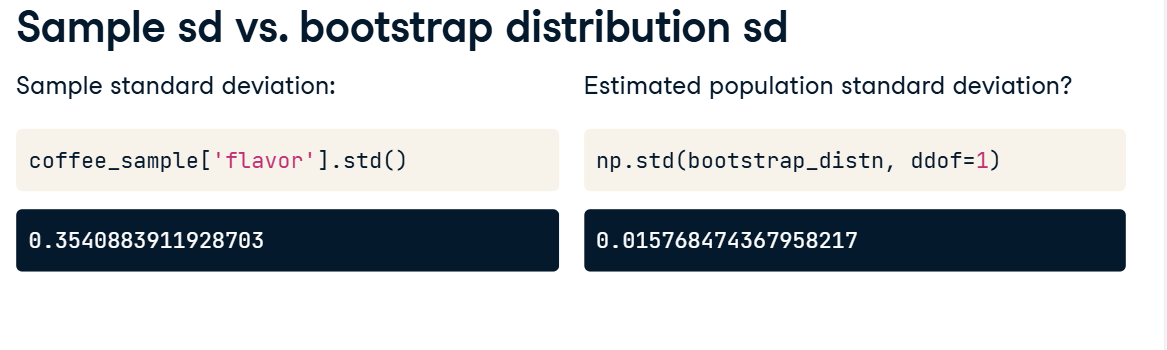


**6. Interpreting the means**

The behavior that you just saw is typical. The bootstrap distribution mean is usually almost identical to the original sample mean. However, that is not often a good thing. If the original sample wasn't closely representative of the population, then the bootstrap distribution mean won't be a good estimate of the population mean. Bootstrapping cannot correct any potential biases due to differences between the sample and the population.

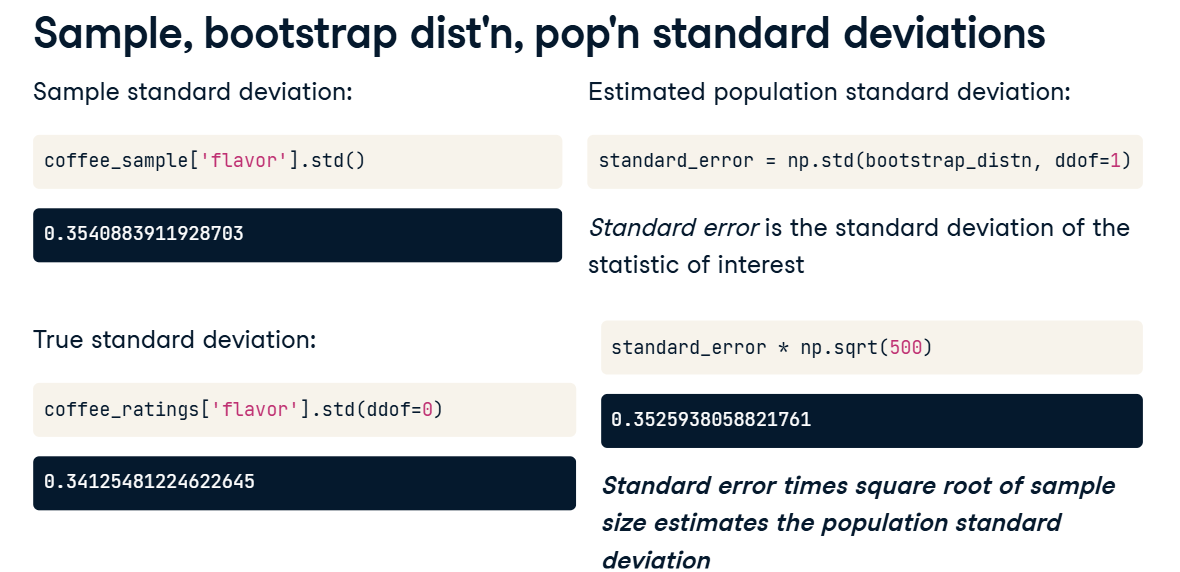
**7. Sample sd vs. bootstrap distribution sd**

While we do have that limitation in estimating the population mean, one great thing about distributions is that we can also quantify variation. The standard deviation of the sample flavors is around zero-point-three-five-four. Recall that pandas dot-std calculates a sample standard deviation by default. If we calculate the standard deviation of the bootstrap distribution, specifying a ddof of one, then we get a completely different number. So what's going on here?



**8. Sample, bootstrap dist'n, pop'n standard deviations**

Remember that one goal of bootstrapping is to quantify what variability we might expect in our sample statistic as we go from one sample to another. Recall that this quantity is called the standard error as measured by the standard deviation of the sampling distribution of that statistic. The standard deviation of the bootstrap means can be used as a way to estimate this measure of uncertainty. If we multiply that standard error by the square root of the sample size, we get an estimate of the standard deviation in the original population. Our estimate of the standard deviation is around point-three-five-three. The true standard deviation is around point-three-four-one, so our estimate is pretty close. In fact, it is closer than just using the sample standard deviation alone.



**9. Interpreting the standard errors**

To recap, the estimated standard error is the standard deviation of the bootstrap distribution values for our statistic of interest. This estimated standard error times the square root of the sample size gives a really good estimate of the standard deviation of the population. That is, although bootstrapping was poor at estimating the population mean, it is generally great for estimating the population standard deviation.