**. Simple random and systematic sampling**

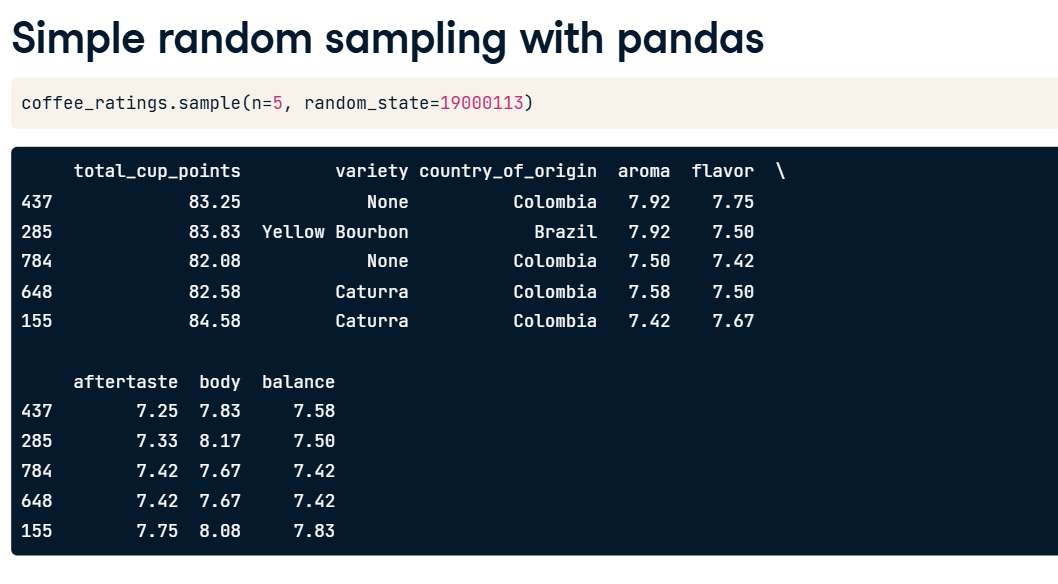
00:00 - 00:09

There are several methods of sampling from a population. In this video, we'll look at simple random sampling and systematic random sampling.

**2. Simple random sampling**

00:09 - 00:21

Simple random sampling works like a raffle or lottery. We start with our population of raffle tickets or lottery balls and randomly pick them out one at a time.



**3. Simple random sampling of coffees**

00:21 - 00:53

In our coffee ratings dataset, instead of raffle tickets or lottery balls, the population consists of coffee varieties. To perform simple random sampling, we take some at random, one at a time. Each coffee has the same chance as any other of being picked. When using this technique, sometimes we might end up with two coffees that were next to each other in the dataset, and sometimes we might end up with large areas of the dataset that were not selected from at all.

**4. Simple random sampling with pandas**

00:53 - 01:22

We've already seen how to do simple random sampling with pandas. We call dot-sample and set n to the size of the sample. We can also set the seed using the random\_state argument to generate reproducible results, just like we did pseudo-random number generation. Previously, by not setting random\_state when sampling, our code would generate a different random sample each time it was run.

**5. Systematic sampling**

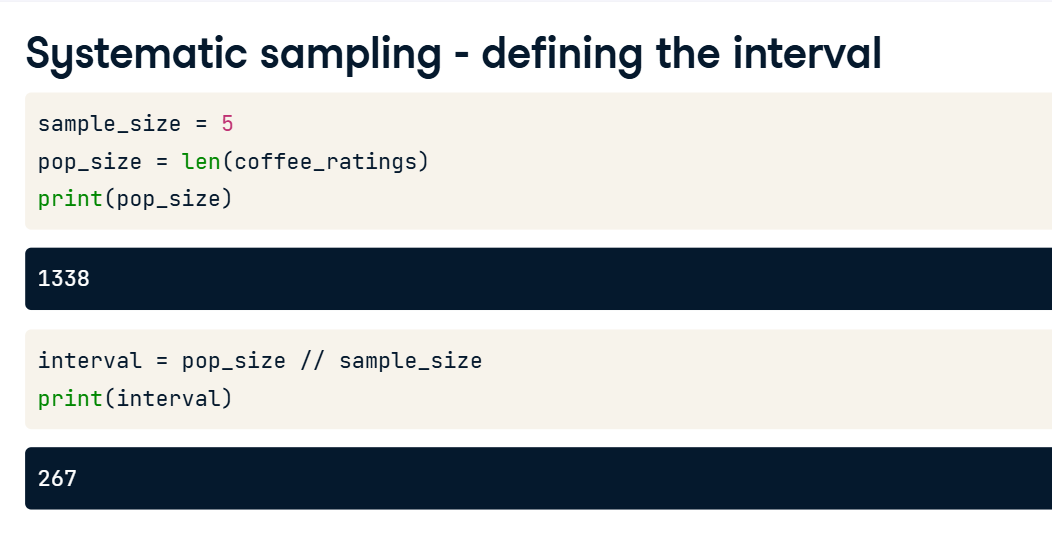
01:22 - 01:37

Another sampling method is known as systematic sampling. This samples the population at regular intervals. Here, looking from top to bottom and left to right within each row, every fifth coffee is sampled.

**6. Systematic sampling - defining the interval**

01:37 - 02:39

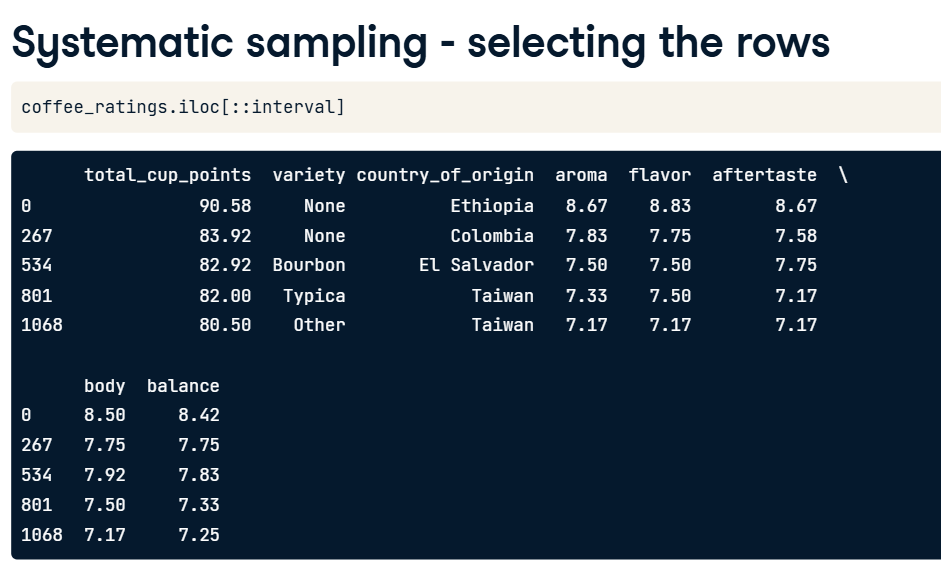
Systematic sampling with pandas is slightly trickier than simple random sampling. The tricky part is determining how big the interval between each row should be for a given sample size. Suppose we want a sample size of five coffees. The population size is the number of rows in the whole dataset, and in this case, it's one thousand three hundred and thirty-eight. The interval is the population size divided by the sample size, but because we want the answer to be an integer, we perform integer division with two forward slashes. This is like standard division but discards any fractional part. One-three-three-eight divided by five is actually two hundred and sixty-seven-point-six, and discarding the fractional part leaves two hundred and sixty-seven. Thus, to get a systematic sample of five coffees, we will select every two hundred sixty-seventh coffee in the dataset.



**7. Systematic sampling - selecting the rows**

02:39 - 03:02

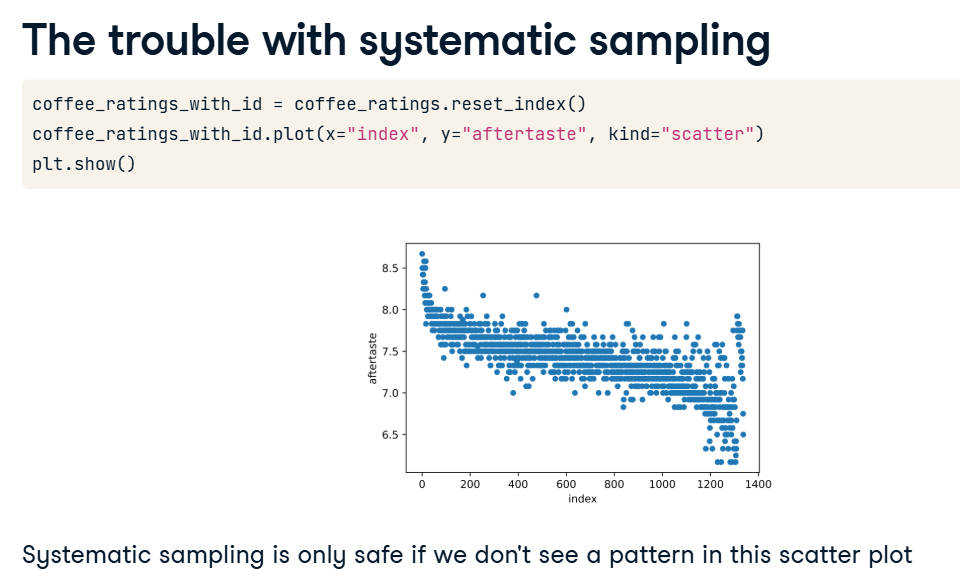
To select every two hundred and sixty-seventh row, we call dot-iloc on coffee\_ratings and pass double-colons and the interval, which is 267 in this case. Double-colon interval tells pandas to select every two hundred and sixty-seventh row from zero to the end of the DataFrame.



**8. The trouble with systematic sampling**

03:02 - 03:43

There is a problem with systematic sampling, though. Suppose we are interested in statistics about the aftertaste attribute of the coffees. To examine this, first, we use reset\_index to create a column of index values in our DataFrame that we can plot. Plotting aftertaste against index shows a pattern. Earlier rows generally have higher aftertaste scores than later rows. This introduces bias into the statistics that we calculate. In general, it is only safe to use systematic sampling if a plot like this has no pattern; that is, it just looks like noise.



**9. Making systematic sampling safe**

03:43 - 04:45

To ensure that systematic sampling is safe, we can randomize the row order before sampling. dot-sample has an argument named frac that lets us specify the proportion of the dataset to return in the sample, rather than the absolute number of rows that n specifies. Setting frac to one randomly samples the whole dataset. In effect, this randomly shuffles the rows. Next, the indices need to be reset so that they go in order from zero again. Specifying drop equals True clears the previous row indexes, and chaining to another reset\_index call creates a column containing these new indexes. Redrawing the plot with the shuffled dataset shows no pattern between aftertaste and index. This is great, but note that once we've shuffled the rows, systematic sampling is essentially the same as simple random sampling.

