### **Statistics Fundamentals**

1. **Sampling**

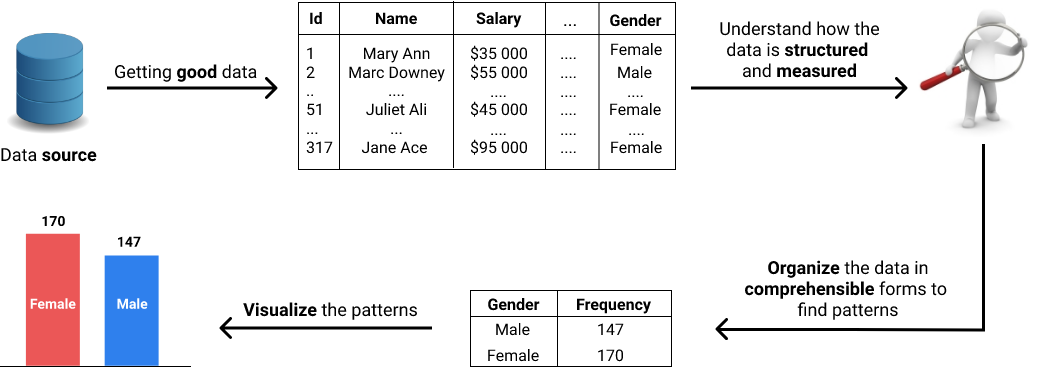
**Mission 1: Introduction**

In previous courses, we learned to perform basic data analysis and data visualization. We learned about some fundamental statistical metrics like the mean or the median, and we plotted bar graphs or Pie plots.

In this step's courses, we'll build on that knowledge, and we'll learn how to do better data analysis. First, we'll go much deeper into the theory behind what we've already learned. Second, we'll learn new and more powerful statistical techniques and metrics like standard deviation, z-scores, confidence intervals, probability estimation, and hypothesis testing (including A/B testing).

In this first course, we begin with discussing the details around getting data for analysis, and continue with trying to understand the intricacies around how data is structured and measured. We'll then move on with learning techniques to organize and visualize relatively large amounts of data, which will make the process of finding patterns considerably less difficult.

Below is a diagram describing the workflow we'll be focusing on throughout this first course.

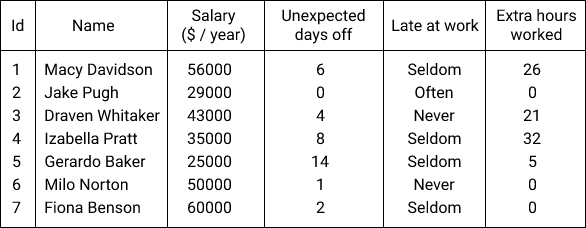


In this first mission, our focus will be on the details around getting data for analysis. As usual, we'll work with a real world data set. Before we dive into the technical details and start playing with the data, we begin with getting a sense about what statistics is.

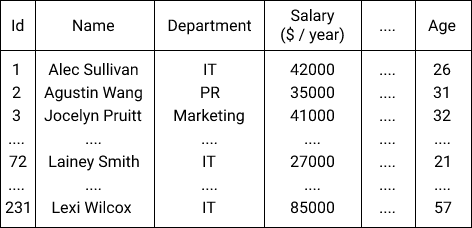
At this stage in our learning journey, a one-sentence definition of statistics would probably sound dull and be difficult to grasp. We'll avoid defining statistics that way, and we'll discuss instead what sort of problems can be solved with statistics. Understanding what challenges we can overcome using statistics should give us a good sense about what statistics is.

**Mission 2: Solving Problems with Statistics**

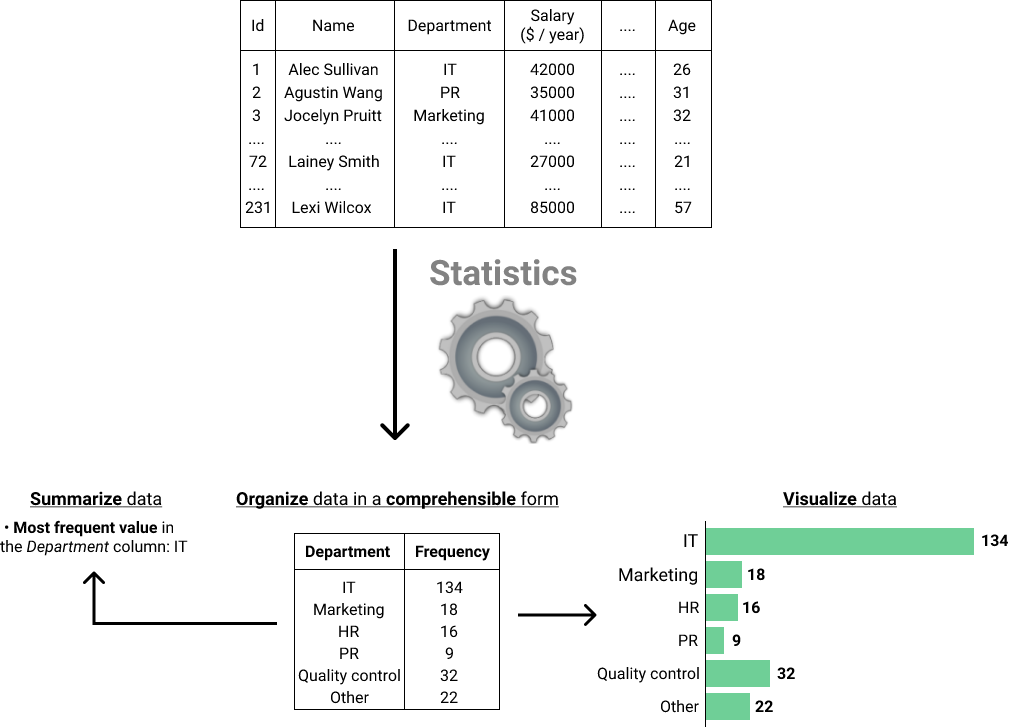
Imagine you're managing a small tech company with 7 employees. At the end of the year, you piece together some data about your employees, with the intention of understanding better the state of your company. The data you collected is straightforward, and you can quickly make a few conclusions just by using a bit of arithmetics and logic.



But years have gone by, and your business has grown into a successful company with 231 employees. You still want to get insights from data, but now you have so much of it that analyzing it has become difficult and inconvenient. As you continue to scale your company, analyzing data slowly gears toward becoming an impossible task.



This is an example of a problem we can solve with statistics. Using statistical techniques, we can *organize*, *summarize*, and *visualize* large amounts of data to find patterns that otherwise would remain hidden.

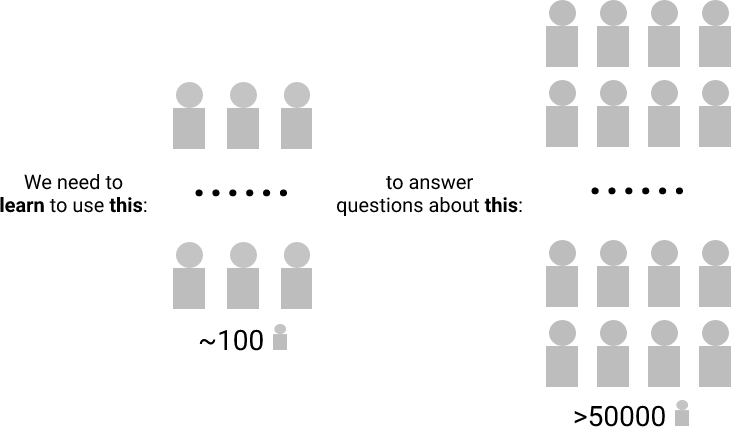


More years have gone by, and now you run an international company with over 5,0000 employees. You've recently made a company-wide change which resulted in making the work of your employees more demanding. Now you want to determine whether the employees have been impacted negatively in any significant way. If this is true, then the change may backfire in the long run, and it'd be a good decision to revert the process while it's still possible.

You reach out to your data analyst and ask for her opinion. She says that she can do a survey to collect data, and answer your question. Surveying over 50000 employees would be time-consuming and expensive, so you're being told that 100 people or so will be enough to survey to get an answer to your question.

One week later, the analysis shows that people generally report they are less satisfied with their work compared to last year (when the change hadn't been yet implemented). Also, the inability to balance work and personal life is the main reported cause of dissatisfaction. Your analyst also tells you that the decrease in satisfaction is *significant*, meaning that it's very unlikely to simply have happened by chance. Something must have caused the decrease, and that something is probably the major change you've done recently.

This sort of scenario is very common in practice. As a data analyst, you'll often need to use a small set of data to answer questions about a much larger set of data.



We'll learn ourselves throughout the statistics courses how to use a small set of data to answer questions about a much larger set of data.

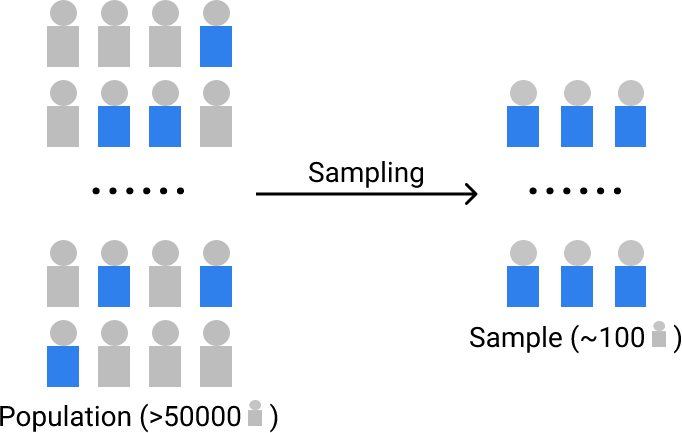
Now we begin with discussing the details around collecting data, which is what the data analyst in our story did when she surveyed employees.

**Mission 3: Populations and Sizes**

The data analyst in our previous example tried to answer whether people in the company are less satisfied at work compared to the previous year. Her question was about *all* the people inside the company. Yet she only selected a small group to answer the question.

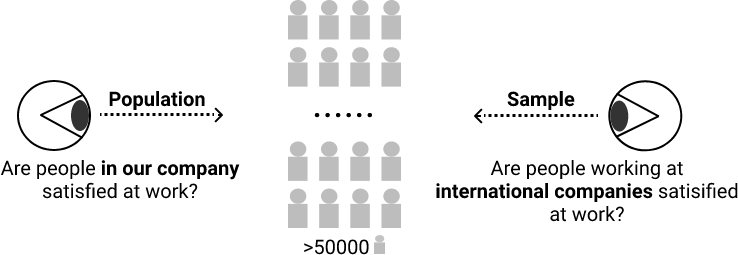
In statistics, the set of *all* individuals relevant to a particular statistical question is called a population. For our analyst's question, all the people inside the company were relevant. So the population in this case consisted from all the people in the company.

A smaller group selected from a population is called a sample. When we select a smaller group from a population we do sampling. In our example, the data analyst took a sample of approximately 100 people from a population of over 50000 people.



Whether a set of data is a sample or a population depends on the question we're trying to answer. For our analyst's question, the population consisted of all the company members. But if we change the question, the same group of individuals can become a sample.

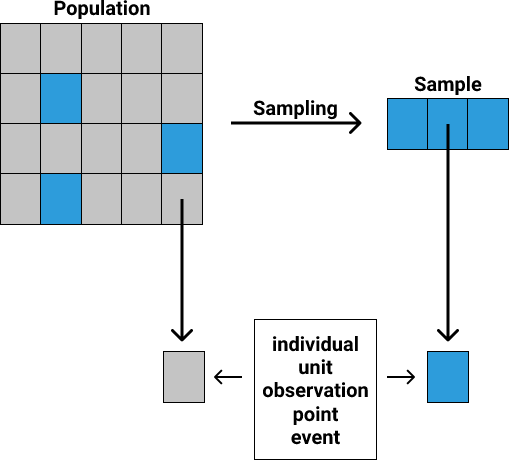
For instance, if we tried to find out whether people at international companies are satisfied at work, then our group formed by over 50000 employees would become a sample. There are a lot of international companies out there, and ours is just one of them. The population (the set of *all* individuals relevant to this question) is made up of all the people working in all the international companies.



Populations do not necessarily consist of people. Behavioral scientists, for instance, often try to answer questions about populations of monkeys, rats or other lab animals. In a similar way, other people try to answer questions about countries, companies, vegetables, soils, pieces of equipment produced in a factory, etc.

The individual elements of a population or a sample go under many names. You'll often see the elements of a population referred to as *individuals, units, events, observations*. These are all used interchangeably and refer to the same thing: the individual parts of a population. When we use the term "population individuals", the population is not necessarily composed of people. "Individuals" here is a general term that could refer to people, needles, frogs, stars, etc.

In the case of a sample, you'll often see this terminology used interchangeably: *sample unit, sample point, sample individual, or sample observation*.



Instructions

Now it's our turn to play the data analyst. We collected data about the salary of *all* the individuals in the company working in IT roles. Based on this data, we want to answer a series of questions. Depending on the question, our data is either a sample or a population. Identify which is the case, and assign to the corresponding variable the string 'population' or 'sample'. Here are the questions we need to answer:

1. What's the average salary of the individuals in our company working in IT roles? (Assign either 'population' or 'sample' to the variable question1.)
2. What's the proportion of individuals in the IT department having salaries under $60000? (Assign either 'population' or 'sample' to the variable question2.)
3. What's the minimum salary in the entire company? (Assign either 'population' or 'sample' to the variable question3.)
4. What's the minimum salary in the IT department of our company? (Assign either 'population' or 'sample' to the variable question4.)
5. What's the proportion of salaries under $20000 in the entire company? (Assign either 'population' or 'sample' to the variable question5.)

**Mission 4: Sampling Errors**

For every statistical question we want to answer, we should try to use the population. In practice, that's not always possible because the populations of interest usually vary from large to extremely large. Also, getting data is generally not an easy task, so small populations often pose problems too.

These problems can be solved by sampling from the population that interests us. Although not as good as working with the entire population, working with a sample is the next best thing we can do.

When we sample, the data we get might be more or less similar to the data in the population. For instance, let's say we know that the average salary in our company is $34500, and the proportion of women is 60%. We take two samples and find these results:



As you can see, the metrics of the two samples are different than the metrics of the population. A sample is by definition an incomplete set of data for the question we're trying to answer. For this reason, there's almost always some difference between the metrics of a population and the metrics of a sample. This difference can be seen as an error, and because it's the result of sampling, it's called sampling error.

A metric specific to a population is called a parameter, while one specific to a sample is called a statistic. In our example above, the average salary of *all the employees* is a parameter because it's a metric that describes the entire population. The average salaries from our two samples are examples of statistics because they only describe the samples.

Another way to think of the concept of the sampling error is as the difference between a parameter and a statistic:

sampling error=parameter−statistic

At this point in the mission, we'll move from the tech company example to working with a real world data set. Our first challenge will be to measure sampling error using this data set.

The data set is about basketball players in WNBA (Women's National Basketball Association), and contains general information about players, along with their metrics for the season 2016-2017. The data set was put together by Thomas De Jonghe, and can be downloaded from [Kaggle](https://www.kaggle.com/jinxbe/wnba-player-stats-2017), where you can also find useful documentation for the data set.

Instructions

Get familiar with the data set.

* Print the first five rows using DataFrame.head() and the last five rows with DataFrame.tail().
* Find the number of rows and columns using DataFrame.shape.
* Learn about each column from the [documentation](https://www.kaggle.com/jinxbe/wnba-player-stats-2017). You can also find useful documentation in this [glossary](https://www.basketball-reference.com/about/glossary.html) and on [WNBA's official page](http://www.wnba.com/stats/player-stats/).

Take one measure of the sampling error.

* Use the Games Played column to find the maximum number of games played by a player in the season 2016-2017. The data set contains all the players that had at least one game, so it's a population relative to our question. Find this *parameter*, and assign the result to a variable named parameter.
* Using the [Series.sample() method](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.sample.html?highlight=sample#pandas.Series.sample), sample randomly 30 players from the population, and assign the result to a variable named sample.
  + When calling Series.sample(), use the the argument random\_state = 1. This makes your results reproducible and helps us with the answer checking (we'll discuss more about this in the next screen).
* Find the maximum number of games using the sample, and assign the result to a variable named statistic.
* Measure the sampling error, and assign the result to a variable named sampling\_error.

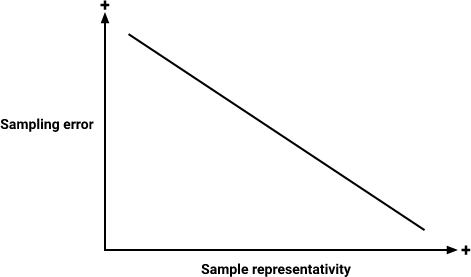
**Mission 5: Simple Random Sampling**

When we sample we want to minimize the sampling error as much as possible. We want our sample to mirror the population as closely as possible.

If we sampled to measure the mean height of adults in the US, we'd like our sample *statistic* (sample mean height) to get as close as possible to the population's *parameter* (population mean height). For this to happen, we need the individuals in our sample to form a group that is similar in structure with the group forming the population.

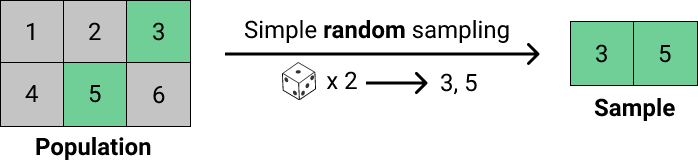
The US adult population is diverse, made of people of various heights. If we sampled 100 individuals from various basketball teams, then we'd almost certainly get a sample whose structure is significantly different than that of the population. As a consequence, we should expect a large sampling error (a large discrepancy between our sample's statistic (sample mean height) and the population's parameter (population mean height)).

In statistical terms, we want our samples to be representative of their corresponding populations. If a sample is representative, then the sampling error is low. The more representative a sample is, the smaller the sampling error. The less representative a sample is, the greater the sampling error.



To make our samples representative, we can try to give every individual in the population an equal chance to be selected in our samples. We want a very tall individual to have the same chance as being selected as an individual having a medium or short height. To give every individual an equal chance of being picked, we need to sample randomly.

One way to perform random sampling is to generate random numbers and use them to select a few sample units from the population. In statistics, this sampling method is called simple random sampling, and it's often abbreviated as SRS.



In our previous exercise, we used Series.sample() to sample. This method performs simple random sampling by generating an array of random numbers, and then using those numbers to select values from a Series at the indices corresponding to those random numbers. The method can be also extended for DataFrame [objects](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.sample.html?highlight=sample#pandas.DataFrame.sample), where random rows or columns can be sampled.

When we use the random\_state parameter, like we did in the previous exercise with Series.sample(30, random\_state = 1), we make the generation of random numbers predictable. This is because Series.sample() uses a [pseudorandom number generator](https://en.wikipedia.org/wiki/Pseudorandom_number_generator) under the hood. A pseudorandom number generator uses an *initial value* to generate a sequence of numbers that has properties similar to those of a sequence that is truly random. With random\_state we specify that initial value used by the pseudorandom number generator.

If we want to generate a sequence of five numbers using a pseudorandom generator, and begin from an initial value of 1, we'll get the same five numbers no matter how many times we run the code. If we ran wnba['Games Played'].sample(5, random\_state = 1) we'd get the same sample every time we run the code.

Pseudorandom number generators are of great use in scientific research where [reproducible work](https://en.wikipedia.org/wiki/Reproducibility) is necessary. In our case, pseudorandom number generators allow us to work with the same samples as you do in the exercises, which allows in turn for a meaningful answer checking.

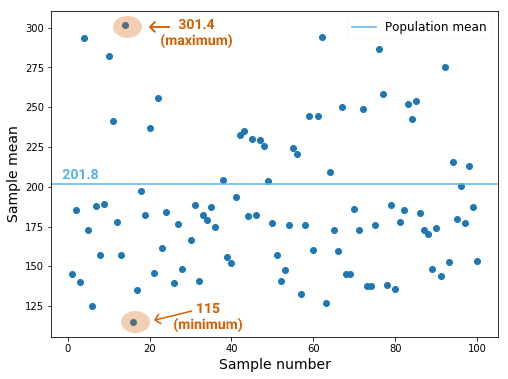
Instructions

Let's visualize the discrepancy between a parameter and its corresponding statistics in the case of simple random sampling.

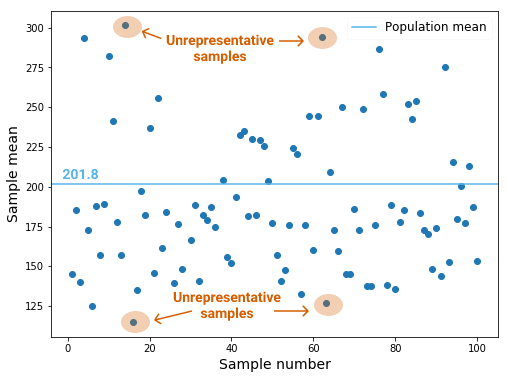
* Using simple random sampling, take 100 samples from our WNBA data set, and for each sample measure the average points scored by a player during the 2016-2017 season. For *each* of the 100 iterations of a for loop:
  + Sample 10 values from the PTS column.
  + Compute the mean of this sample made of 10 values from the PTS column, and append the result to a list.
  + To make your results reproducible, vary the random\_state parameter of the sample() method with values between 0 and 99. For the first iteration of the for loop, random\_state should equal 0, for the second iteration should equal 1, for the third should equal 2, and so on.
* Display the discrepancy between the *parameter* of interest (the mean of the PTS column) and the *statistics* obtained in the previous step.
  + Using plt.scatter(), display all the 100 sample means using a scatter plot. For the x-axis, use integers from 1 to 100 to designate the sample number. Use the y-axis for the sample means.
  + Using plt.axhline(), draw a horizontal line that represents the average number of points in the population.

**Mission 6: The Importance of Sample Size**

From the scatter plot in the last screen, we can notice that the sample means vary a lot around the population mean. With a minimum sample mean of 115 points, a maximum of 301.4, and a population mean of roughly 201.8, we can tell that the sampling error is quite large for some of the cases.

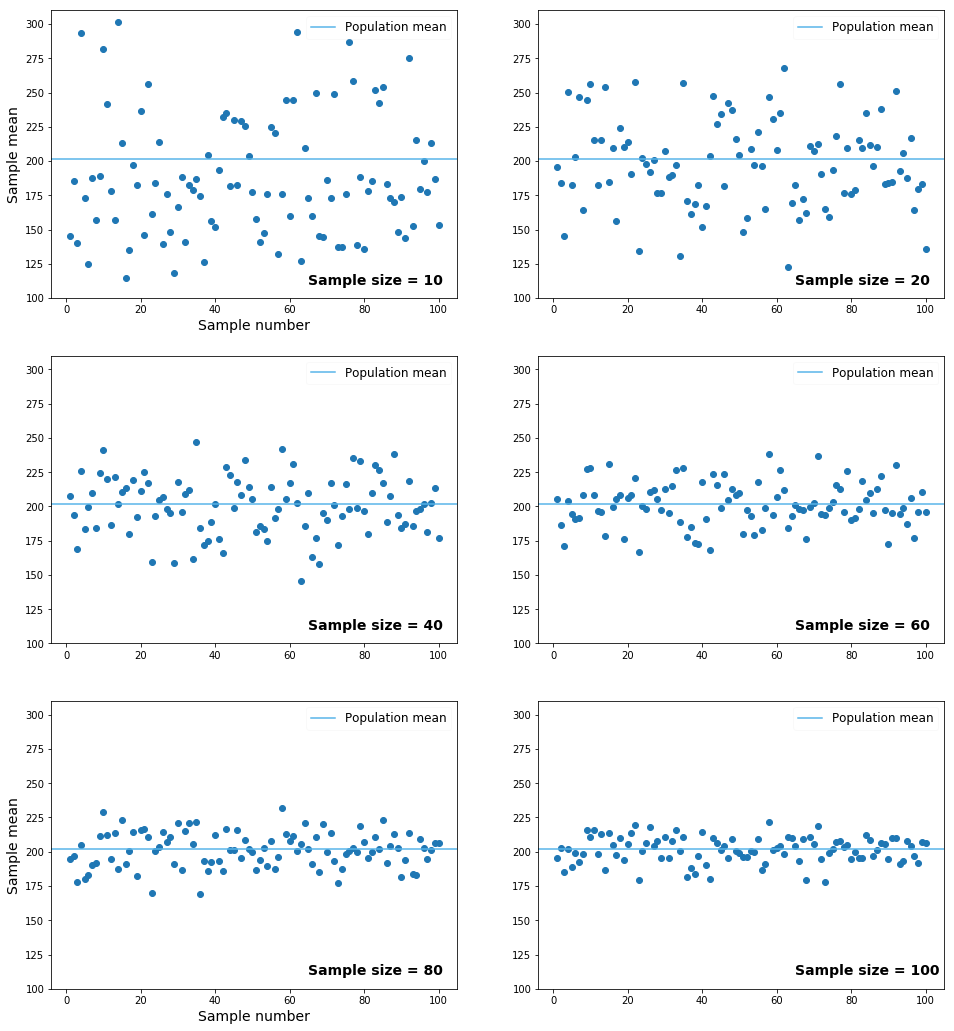


Because sample means vary a lot around the population mean, there's a good chance we get a sample that is not representative of the population:



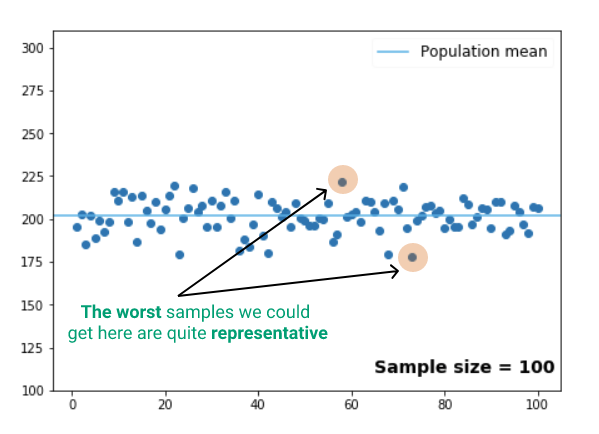
This problem can be solved by increasing the sample size. As we increase the sample size, the sample means vary less around the population mean, and the chances of getting an unrepresentative sample decrease.

In our last exercise we took 100 samples, and each had a sample size of 10 units. This is what happens when we repeat the procedure, but increase the size of the samples:



We can easily see how sample means tend to vary less and less around the population mean as we increase the sample size. From this observation we can make two conclusions:

* Simple random sampling is not a reliable sampling method when the sample size is small. Because sample means vary a lot around the population mean, there's a good chance we'll get an unrepresentative sample.
* When we do simple random sampling, we should try to get a sample that is as large as possible. A large sample decreases the variability of the sampling process, which in turn decreases the chances that we'll get an unrepresentative sample.



**Mission 7: Stratified Sampling**

Because simple random sampling is entirely random, it can leave out certain population individuals that are of great interest to some of the questions we may have.

For example, players in basketball play in different positions on the court. The metrics of a player (number of points, number of assists, etc.) depend on their position, and we might want to analyze the patterns for each individual position. If we perform simple random sampling, there's a chance that some categories won't be included in our sample. In other words, it's not guaranteed that we'll have a representative sample that has observations for every position we want to analyze.

There are five unique positions in our data set:

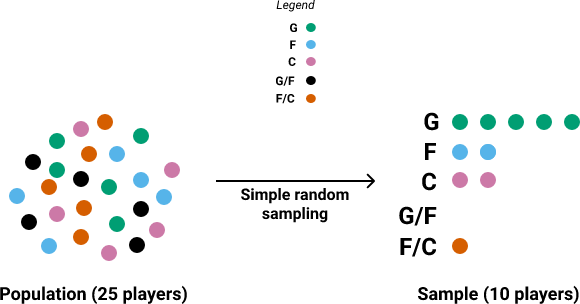
print(wnba['Pos'].unique())

array(['F', 'G/F', 'G', 'C', 'F/C'], dtype=object)

Let's decipher quickly each abbreviation:

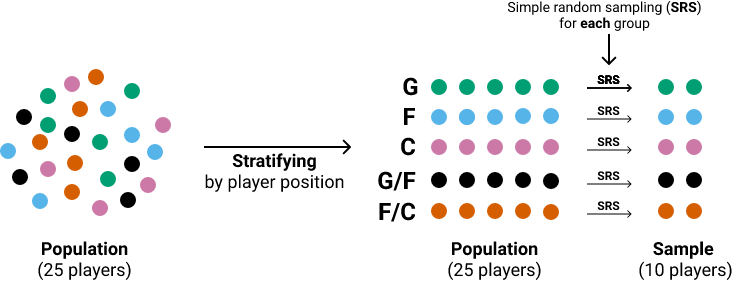
|  |  |
| --- | --- |
| Abbreviation | Full name |
| F | Forward |
| G | Guard |
| C | Center |
| G/F | Guard/Forward |
| F/C | Forward/Center |

The downside of simple random sampling is that it can leave out individuals playing in a certain position on the field. Visually, and on a smaller scale, this is what could happen:



To ensure we end up with a sample that has observations for all the categories of interest, we can change the sampling method. We can organize our data set into different groups, and then do simple random sampling for every group. We can group our data set *by player position*, and then sample randomly from each group.

Visually, and on a smaller scale, we need to do this:



This sampling method is called stratified sampling, and each stratified group is also known as a stratum.

Instructions

Perform stratified sampling: stratify the data set by player position, and then do simple random sampling on every stratum. At the end, use the sample to find which position has the greatest number of points per game.

* Create a new column which describes the number of points a player scored per game during the season. The number of total points a player scored the entire season is stored in the PTS column, and the number of games played in the Games Played column. Give the new column a relevant name.
* Stratify the wnba data set by player position. The Pos column describes a player's position on the field. Assign each stratum to a different variable.
* Loop through the strata, and for *each* stratum:
  + Sample 10 observations using simple random sampling (set random\_state = 0).
  + Find the mean points per game using the sample. Use the new column you've created earlier.
  + Find a way to store the mean along with its corresponding position. You can use a dictionary.
* Find the position that has the greatest number of points per game, and assign its name to a variable named position\_most\_points.
  + To find the dictionary key that has the greatest dictionary value, you can use [this technique](https://stackoverflow.com/questions/268272/getting-key-with-maximum-value-in-dictionary#280156).

**Mission 8: Proportional Stratified Sampling**

Earlier in this mission we performed simple random sampling 100 times on the original data set, and for each sample we computed the mean number of *total* points a player scores in a season. The problem is that the number of total points is influenced by the number of games played, which ranges from 2 to 32:

print(wnba['Games Played'].min())

2

print(wnba['Games Played'].max())

32

Approximately 72.7% of the players had more than 23 games for the 2016-2017 season, which means that the mean of the total points is probably influenced by this category of players who played a lot of games. Let's take a look at the other percentages too:

print(wnba['Games Played'].value\_counts(bins = 3, normalize = True) \* 100)

(22.0, 32.0] 72.727273

(12.0, 22.0] 18.181818

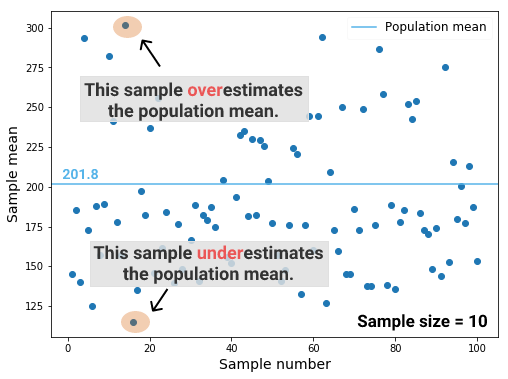
(1.969, 12.0] 9.090909

Name: Games Played, dtype: float64

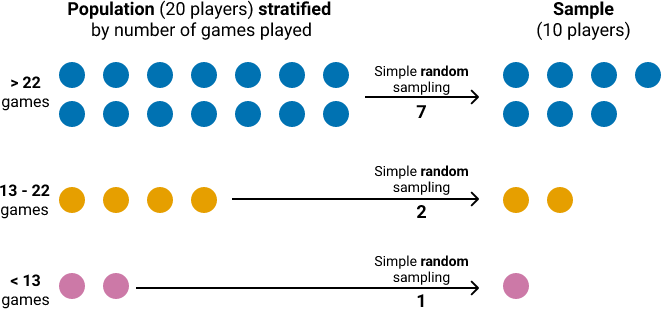
As a side note on the output above, (1.969, 12.0], (12.0, 22.0] and (22.0, 32.0] are number intervals. The ( character indicates that the beginning of the interval is not included, and the ] indicates that the endpoint is included. For example, (22.0, 32.0] means that 22.0 isn't included, while 32.0 is, and the interval contains this array of numbers: [23, 24, 25, 26, 27, 28, 29, 30, 31, 32].

Getting back to our discussion, when we compute the mean of the total points using the population (the entire data set), the mean will probably be signficantly influenced by those 72.7% players who played more than 23 games. However, when we sample randomly, we can end up with a sample where the proportions are different than in the population.

For instance, we might end up with a sample where only 2% of the players played more than 23 games. This will result in a sample mean which *underestimates* the population mean. Or we could have a sample where more than 95% of the players had 23 games in the 2016-2017 season. This will result in *overestimating* the population mean. This scenario of under or over estimation is common for small samples.



One solution to this problem is to use stratified sampling while being mindful of the proportions in the population. We can stratify our data set by the number of games played, and then sample randomly from each stratum a *proportional*number of observations.



In the diagram above, we can see that from a population of 20 individuals:

* 14 individuals played more than 22 games.
* 4 individuals played between 13 and 22 games.
* 2 individuals played below 13 games.

Transforming these figures to percentages, 70% of the individuals played more than 22 games, 20% played between 13 and 22 games, and 10% played below 13 games. Because we sampled proportionally, the same percentages (70%, 20%, 10%) are preserved in the sample (even though the absolute values are different): 70% played more than 22 games, 20% played between 13 and 22 games, and 10% played below 13 games.

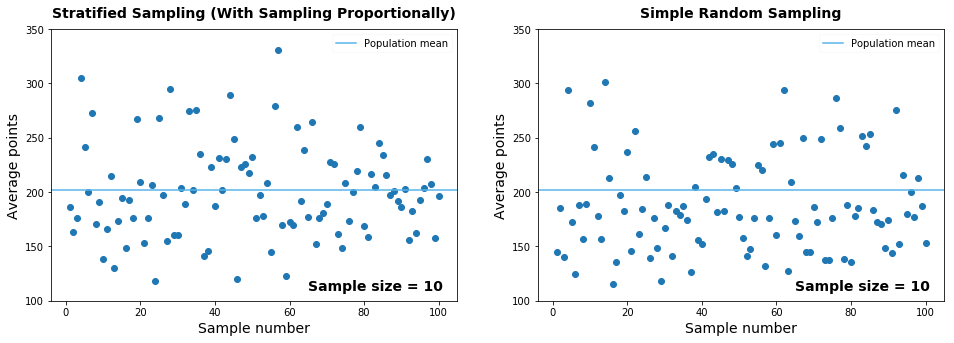
Instructions

Perform stratified sampling on the data set 100 times, and sample strata proportionally.

* Stratify the data set by the number of games played in the following way: the first strata should be composed of players that played 12 games or less; the second of players that played more than 12 games, but up to 22 (included); and the third of players that played more than 22 games (22 not included).
* Perform stratified sampling 100 times. For each of the 100 iterations of a forloop:
  + Sample each stratum proportionally. Sample at random: one sample observation from the first stratum, two sample observations from the second, and seven sample observations from the third stratum.
  + random\_state should vary from 0 to 99: 0 for the first iteration, 99 for the last iteration.
  + Once you're done with the sampling for the current iteration of the loop, concatenate all the sample observations into one final sample. You can use pd.concat().
  + Compute the mean of the final sample, and append it to a list defined outside the loop. The mean should be for the PTS column.
* Display the entire sampling process.
  + Using plt.scatter(), display the sampling means on a scatter plot. Place the means on the y-axis, and the sample numbers on the x-axis (the numbers should range from 1 to 100 - both endpoints included).
  + Using plt.axhline(), display the population mean for the total points in the form of a horizontal line.

**Mission 9: Choosing the Right Strata**

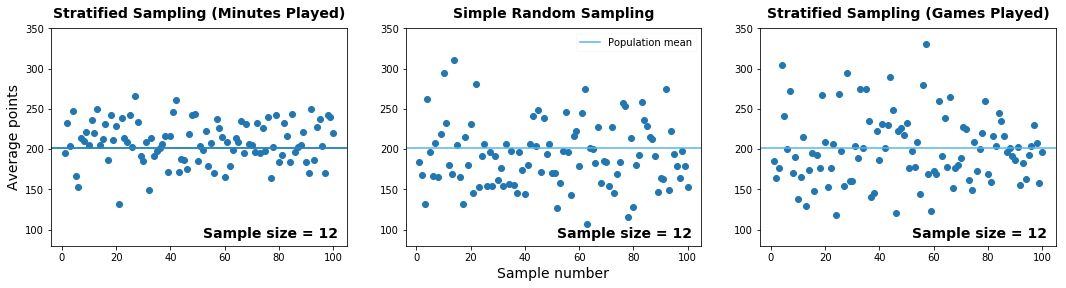
You might not have been very impressed by what we've just got with sampling proportionally. The variability of the sampling was quite large, and many sample means were unrepresentative, being far from the population mean. In fact, this sampling method doesn't seem to perform better than simple random sampling:



The poor performance is caused by a bad choice of strata. We stratified the data by the number of games played, but this isn't a good approach. A player is considered as having played one game even if she only played for one or two minutes. But others play 30 or 40 minutes, and they're still considered as having played one game.

It makes more sense to stratify the data by number of minutes played, rather than by number of games played. The minutes played are a much better indicator of how much a player scored in a season than the number of games played.

Our data set contains the total amount of minutes a player had for the entire season. If we make strata based on minutes played, and then sample proportionally using stratified sampling, we get something visibly better than simple random sampling (especially in terms of variability):

We increased the sample size to 12 so that we can do a better proportional sampling for the strata organized by minutes played.

Here are a few guidelines for choosing good strata:

1. Minimize the variability within each stratum.

For instance, avoid having in the same stratum a player that has scored 10 points and a player that has scored 500. If the variability is high, it might be a sign that you either need a more granular stratification (need more strata), or you need to change the criterion of stratification (an example of criterion is minutes played).

2. Maximize the variability between strata.

Good strata are different from one another. If you have strata that are similar to one another with respect to what you want to measure, you might need a more granular stratification, or to change the stratification criterion. In the previous screen, stratifying the data by games played resulted in strata that weren't too different from each other with respect to the distribution of the total points. We managed to increase the variability between strata by changing the criterion of stratification to minutes played.

3. The stratification criterion should be strongly correlated with the property you're trying to measure.

For instance, the column describing minutes played (the criterion) should be strongly correlated with the number of total points (property we want to measure). We've covered briefly the concept of correlation in the pandas courses, and we'll cover it again later in these statistics courses, so don't worry if the concept of correlation doesn't make much sense to you now.

We've left the code editor open for you to try to experiment with the different sampling methods we've learned so far. One thing you can try is to replicate the last graph above. You can then play with sample sizes, and try to get insights into how variability and sampling error change.

**Mission 10: Cluster Sampling**

The data set we've been working with was scraped from the [WNBA's website](http://www.wnba.com/stats/player-stats/#?Season=2017&SeasonType=Regular%20Season&PerMode=Totals). The website centralizes data on basketball games and players in the WNBA. Let's suppose for a moment that such a site didn't exist, and the data were instead scattered across each individual team's website. There are twelve unique teams in our data set, which means we'd have to scrape twelve different websites, each requiring its own scraping script.

This scenario is quite common in the data science workflow: you want to answer some questions about a population, but the data is scattered in such a way that data collection is either time-consuming or close to impossible. For instance, let's say you want to analyze how people review and rate movies as a function of movie budget. There are a lot of websites out there that can help with data collection, but how can you go about it so that you can spend one day or two on getting the data you need, rather than one month or two?

One way is to list all the data sources you can find, and then randomly pick only a few of them to collect data from. Then you can sample individually each of the sources you've randomly picked. This sampling method is called cluster sampling, and each of the individual data sources is called a cluster.



In our case, we'd first list all the possible data sources. Assuming that all the teams in our data set have a website where we can take data from, we end up with this list of clusters (each team's *website* is considered a cluster) :

print(wnba['Team'].unique())

array(['DAL', 'LA', 'CON', 'SAN', 'MIN', 'SEA', 'PHO', 'CHI', 'WAS', 'NY',

'ATL', 'IND'], dtype=object)

Then we need to find a way to pick randomly a few clusters from our listing. There are many ways to do that, but the important thing to keep in mind is that we should avoid picking a cluster twice. Here's one way to sample four clusters randomly:

print(pd.Series(wnba['Team'].unique()).sample(4, random\_state = 0))

6 PHO

11 IND

4 MIN

10 ATL

dtype: object

Once we pick the clusters, we move to collecting the data. We can collect all the data from each cluster, but we can also perform sampling on each. It's actually possible to use different sampling methods for different clusters. For instance, we can use stratified sampling on the first two clusters, and simple random sampling on the other two.

Instructions

Let's simulate a cluster sampling on our data set.

* Pick four team clusters randomly using the technique we've learned (use random\_state = 0).
* Collect the data from each cluster without sampling the clusters. Create a new DataFrame object that stores the data collected from all clusters.
* Use the data collected to estimate the mean for the following player attributes:
  + Height;
  + Age;
  + BMI;
  + Total points.
* Finally, measure the sampling error of your estimates, and assign the errors to the following variables: sampling\_error\_height, sampling\_error\_age, sampling\_error\_BMI, sampling\_error\_points.

**Mission 11: Sampling in Data Science Practice**

So far, we've explored a few scenarios where sampling can be useful. There are more situations, however, where a data scientist can use sampling, and we discuss a few in this section.

Let's say you work for an e-commerce company that has a table in a database with more than 10 million rows of online transactions. The marketing team asks you to analyze the data and find categories of customers with a low buying rate, so that they can target their marketing campaigns at the right people. Instead of working with more than 10 million rows at each step of your analysis, you can save a lot of code running time by sampling several hundred rows, and perform your analysis on the sample. You can do a simple random sampling, but if you're interested in some categories beforehand, it might be a good idea to use stratified sampling.

Let's consider a different situation. It could be that you need to collect data from an API that either has usage limit, or is not free. In this case, you are more or less forced to sample. Knowing *how* and *what* to sample can be of great use.

Another common use case of sampling is when the data is scattered across different locations (different websites, different databases, different companies, etc.). As we've discussed in the previous screen, cluster sampling would be a great choice in such a scenario.

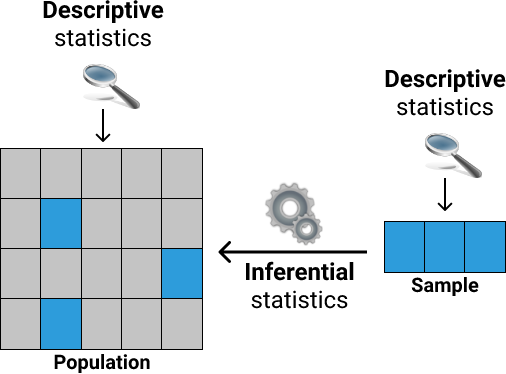
Sampling is a vast topic in statistics, and there are other sampling methods besides what we've discussed so far in our course. [Here's](https://en.wikipedia.org/wiki/Sampling_(statistics)#Sampling_methods) a good starting point to read about other potentially useful sampling methods.

**Mission 12: Descriptive and Inferential Statistics**

Practical statistical analysis revolves entirely around the distinction between a population and a sample. When we're doing statistics in practice, our goal is either to describe a sample or a population, or to use a sample to draw conclusions about the population to which it belongs (or a mix of these two goals).

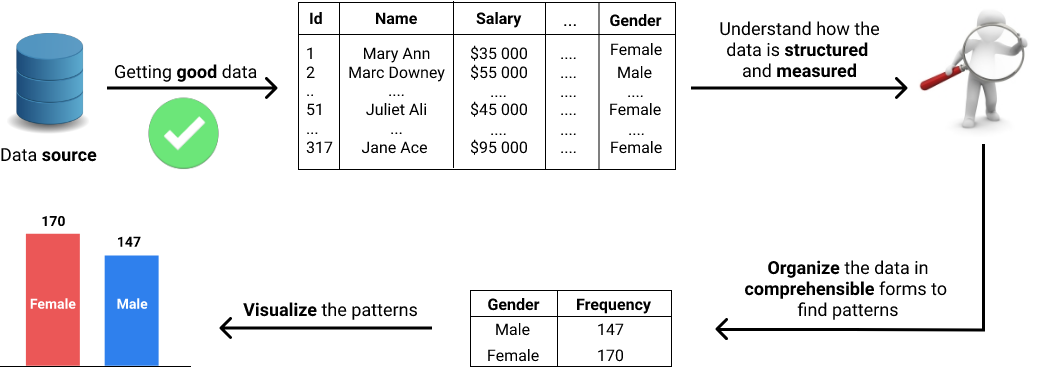
When we describe a sample or a population (by measuring averages, proportions, and other metrics; by visualizing properties of the data through graphs; etc.), we do descriptive statistics.

When we try to use a sample to draw conclusions about a population, we do inferential statistics (we *infer* information from the sample about the population).



**Mission 13: Next Steps**

Throughout this mission, we discussed the details around collecting data for our analysis, and completed the first part of the workflow we'll cover in this course.



Next in our statistics journey, we'll learn what variables are in statistics, how data sets are structured as collections of variables, and how each variable is measured.

**Mission 14: Takeaways**

Syntax

* Sampling randomly a Series object:

### Sampling 10 sample points ###

sample\_10 = Series.sample(10)

### Sampling 500 sample points ###

sample\_500 = Series.sample(500)

* Making the generation of random numbers predictable using the random\_stateparameter:

### Sampling 10 sample points in a reproducible way ###

sample\_10 = Series.sample(10, random\_state = 1)

### Using a different value for `random\_state` ###

sample\_10\_different = Series.sample(10, random\_state = 2)

## Concepts

* The set of *all* individuals relevant to a particular statistical question is called a population. A smaller group selected from a population is called a sample. When we select a smaller group from a population, we do sampling.
* A parameter is a metric specific to a population, and a statistic is a metric specific to a sample. The difference between a statistic and its corresponding parameter is called sampling error. If the sampling error is low, then the sample is representative.
* To make our samples representative we can try different sampling methods:
  + Simple random sampling.
  + Stratified sampling.
  + Cluster sampling.
* When we describe a sample or a population, we do descriptive statistics. When we try to use a sample to draw conclusions about a population, we do inferential statistics (we *infer* information from the sample about the population)