In data analysis and statistics, an ideal dataset is one that contains measurements and results from every possible outcome, condition, or consideration. These datasets are known as a **population** **dataset** and contain all possible elements of a study or experiment.

Often, such an exhaustive dataset is prohibitively expensive or logistically impossible to generate. In this case, we must use a **sample** or subset of the population dataset, where not all elements of a study or experiment are collected or measured.

**CAUTION**

In data science, the concept of sample versus population does not strictly apply to people or animals. Any comprehensive dataset is considered a population, and any dataset that is a subset of a larger dataset is considered a sample.

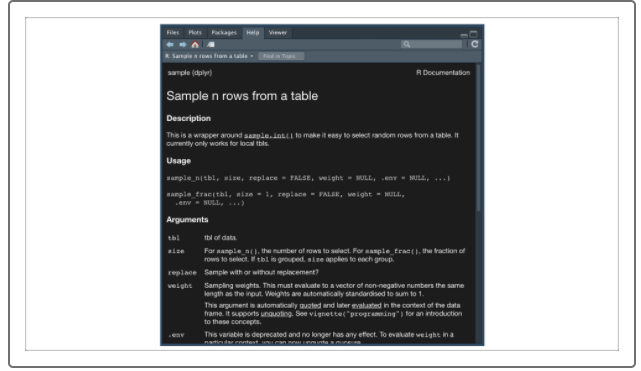
Since a sample dataset is just that—a sample—we must be clear about how a sample dataset represents the corresponding population data. One of the most straightforward ways to characterize a sample versus its population data is to compare the mean and standard deviation of both datasets. Ideally, a sample dataset will have a similar distribution to the population data, and therefore the mean and standard deviation would be about equal.

To produce a sample dataset that has a similar distribution to the population data, most statisticians suggest using random sampling. **Random sampling** is a technique in data science in which every subject or data point has an equal chance of being included in the sample. This technique increases the likelihood that even a small sample size will include individuals from each "group" within the population.

If performed using functions such as the built-in sample()function in R, or sample\_n() function from dplyr, the resulting sample distributions should be similar to the input population data. When selecting sample data from a numerical vector, we should use the built-in sample() function. However, in most cases we will want to use the sample\_n() function to select sample data from a two-dimensional data object.

Type the following code into the R console to look at the sample\_n() documentation in the Help pane, listed under the subhead "Usage" in the image below:

>?sample\_n()



Using the sample\_n()function only requires two arguments:

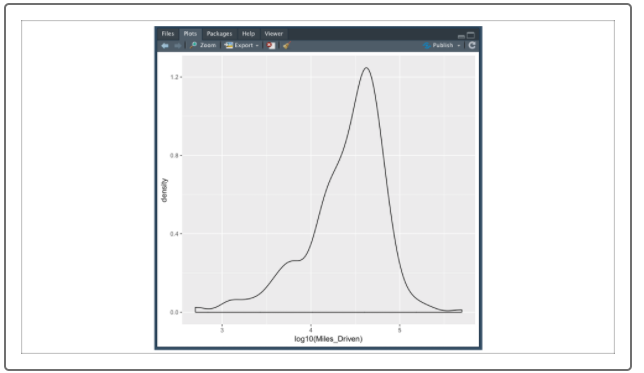
* **tbl** is the name of the input table, which is typically the name of a data frame. Optionally, we can use a dplyr pipe (%>%) to provide the data frame object directly, in which case, this argument is optional.
* **size** is the number of rows to return. As noted in the documentation, if we are providing a data frame that was grouped using the group\_by()function, the **size** argument is the number of groups to return.

To practice generating samples using random sampling, download the [used vehicle dataset](https://2u-data-curriculum-team.s3.amazonaws.com/dataviz-online/module_15/used_car_data.csv) dataset contains market data on more than 300 used vehicles. If we want to visualize the distribution of driven miles for our entire population dataset, we can use the geom\_density()function from ggplot2:

> population\_table <- read.csv('used\_car\_data.csv',check.names = F,stringsAsFactors = F) #import used car dataset

> plt <- ggplot(population\_table,aes(x=log10(Miles\_Driven))) #import dataset into ggplot2

> plt + geom\_density() #visualize distribution using density plot



**IMPORTANT**

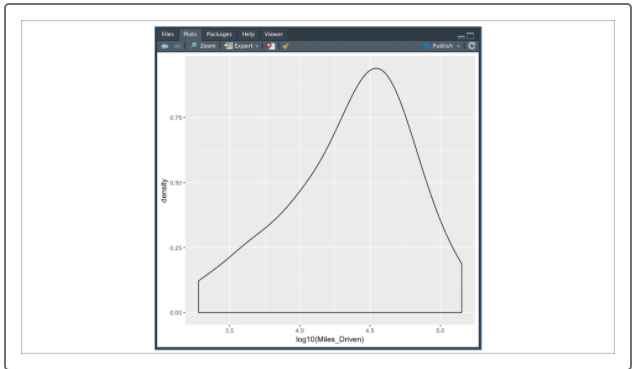
In this example, we want to transform our miles driven using a log10 transformation. This is because the distribution of raw mileage is right skewed—a few used vehicles have more than 100,000 miles, while the majority of used vehicles have less than 50,000 miles. The log10 transformation makes our mileage data more normal.

Now that we characterized our population data using our density plot, we'll create a sample dataset using dplyr's sample\_n()function. Type the following code in the R console:

> sample\_table <- population\_table %>% sample\_n(50) #randomly sample 50 data points

> plt <- ggplot(sample\_table,aes(x=log10(Miles\_Driven))) #import dataset into ggplot2

> plt + geom\_density() #visualize distribution using density plot



By using dplyr's sample\_n() function, we can create a random sample dataset from our population data that contains minimal bias and (ideally) represents the population data.

Depending on the size of the population data, we may need to also adjust the size argument in our sample\_n() function to ensure that our sample data is representative of the underlying population data. There are two basic ways to check that our sample data is representative of the underlying population: a qualitative assessment of each density plot or a quantitative statistical test such as the one-sample t-test.