**Introduction to Seaborn**

Hello! Welcome to this introductory course on Seaborn! My name is Erin Case, and I'll be your instructor.

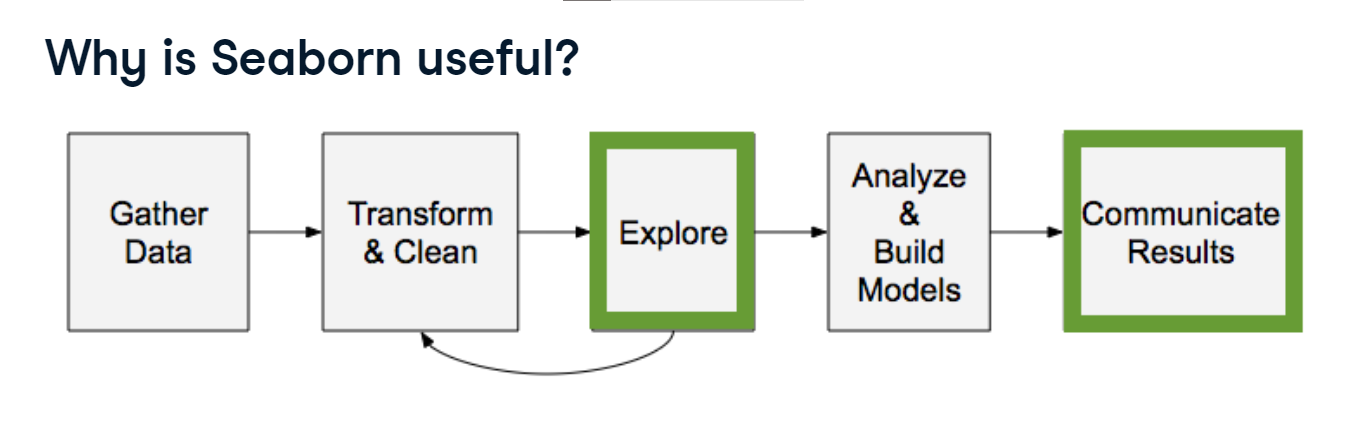
**What is Seaborn?**

So what is Seaborn? Seaborn is a powerful Python library for creating data visualizations. It was developed in order to make it easy to create the most common types of plots. The plot shown here can be created with just a few lines of Seaborn code.

1. 1 Waskom, M. L. (2021). seaborn: statistical data visualization. https://seaborn.pydata.org/

**Why is Seaborn useful?**

This is a picture of a typical data analysis workflow. Data visualization is often a huge component of both the data exploration phase and the communication of results, so Seaborn will be very useful there.



**Advantages of Seaborn**

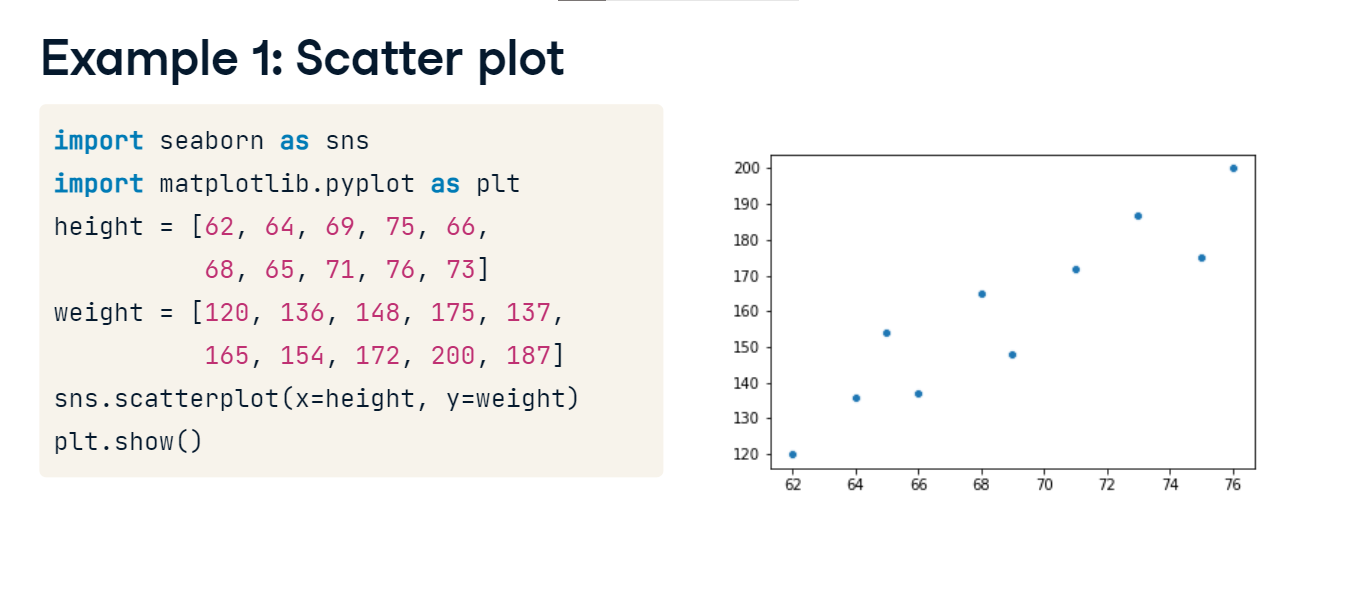
There are several tools that can be used for data visualization, but Seaborn offers several advantages. First, Seaborn's main purpose is to make data visualization easy. It was built to automatically handle a lot of complexity behind the scenes. Second, Seaborn works extremely well with pandas data structures. pandas is a Python library that is widely used for data analysis. Finally, it's built on top of Matplotlib, which is another Python visualization library. Matplotlib is extremely flexible. Seaborn allows you to take advantage of this flexibility when you need it, while avoiding the complexity that Matplotlib's flexibility can introduce.

**Getting started**

To get started, we'll need to import the Seaborn library. The line "import seaborn as sns" will import Seaborn as the conventionally used alias "sns". Why "sns"? The Seaborn library was apparently named after a character named Samuel Norman Seaborn from the television show "The West Wing" - thus, the standard alias is the character's initials ("sns"). We also need to import Matplotlib, which is the library that Seaborn is built on top of. We do this by typing "import matplotlib.pyplot as plt". "plt" is the alias that most people use to refer to Matplotlib, so we'll use that here as well.

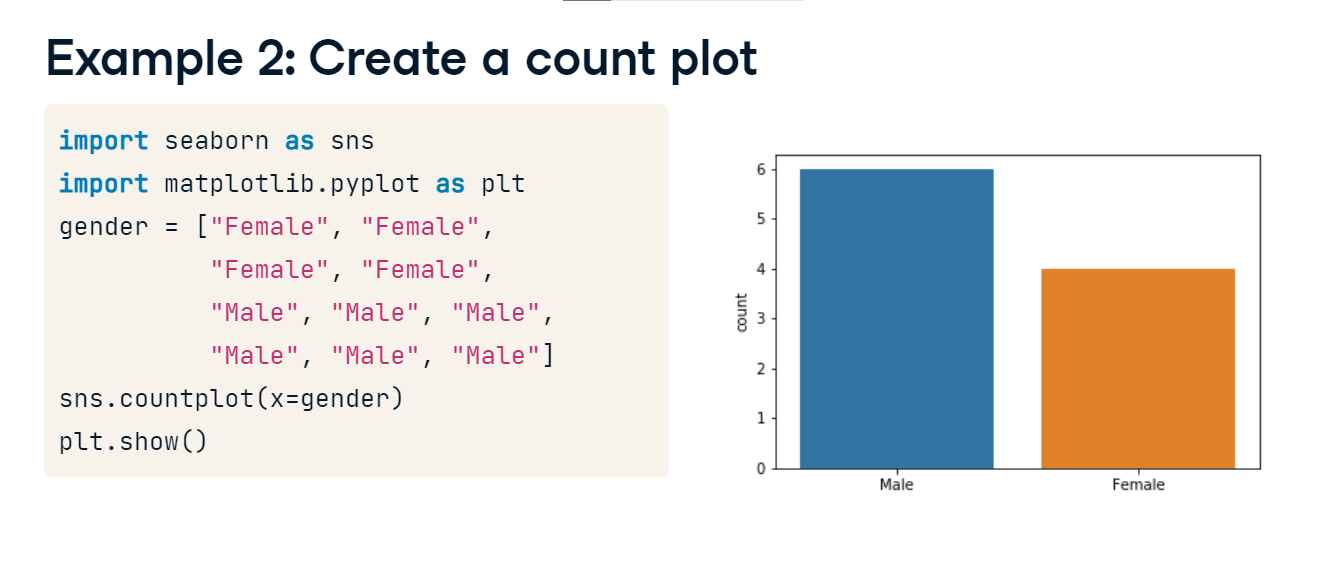
**Example 1: Scatter plot**

Let's now dive into an example to illustrate how easily you can create visualizations using Seaborn. Here, we have data for 10 people consisting of lists of their heights in inches and their weights in pounds. Do taller people tend to weigh more? You can visualize this using a type of plot known as a scatter plot, which you'll learn more about later in the course. Use "sns dot scatterplot()" to call the scatterplot function from the Seaborn library. Then, specify what to put on the x-axis and y-axis. Finally, call the "plt dot show()" function from Matplotlib to show the scatterplot. This plot shows us that taller people tend to have a higher weight.



**Example 2: Create a count plot**

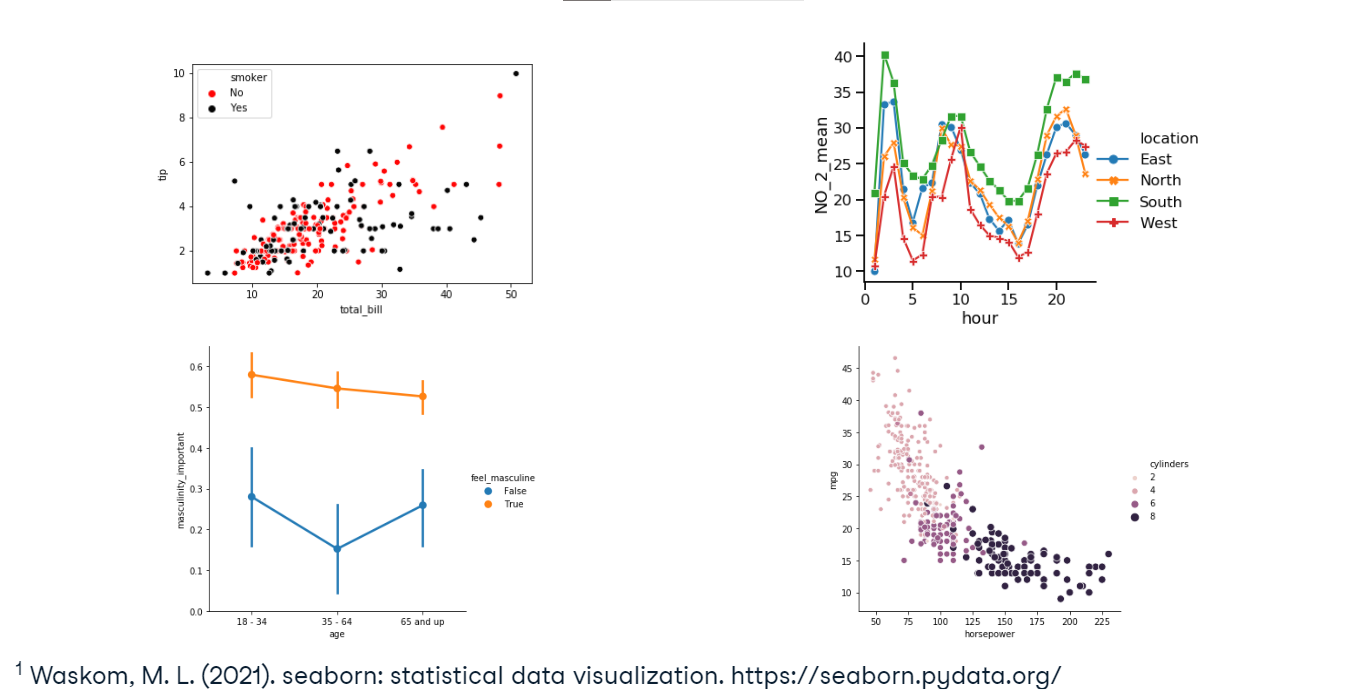
How many of our observations of heights and weights came from males vs. females? You can use another type of plot - the count plot - to investigate this. Count plots take in a categorical list and return bars that represent the number of list entries per category. Use the "countplot()" function and provide the list of every person's gender. This count plot shows that out of the 10 observations we had in our height and weight scatter plot, 6 were male and 4 were female.



**Course Preview**

Now, those were a couple of simple examples. Throughout this course, you'll learn to make more complex visualizations such as those pictured here. More importantly, you'll learn when to use each type of visualization in order to most effectively extract and communicate insights using data.

1. 1 Waskom, M. L. (2021). seaborn: statistical data visualization. https://seaborn.pydata.org/



**Using pandas with Seaborn**

Data scientists commonly use pandas to perform data analysis, so it's a huge advantage that Seaborn works extremely well with pandas data structures. Let's see how this works!

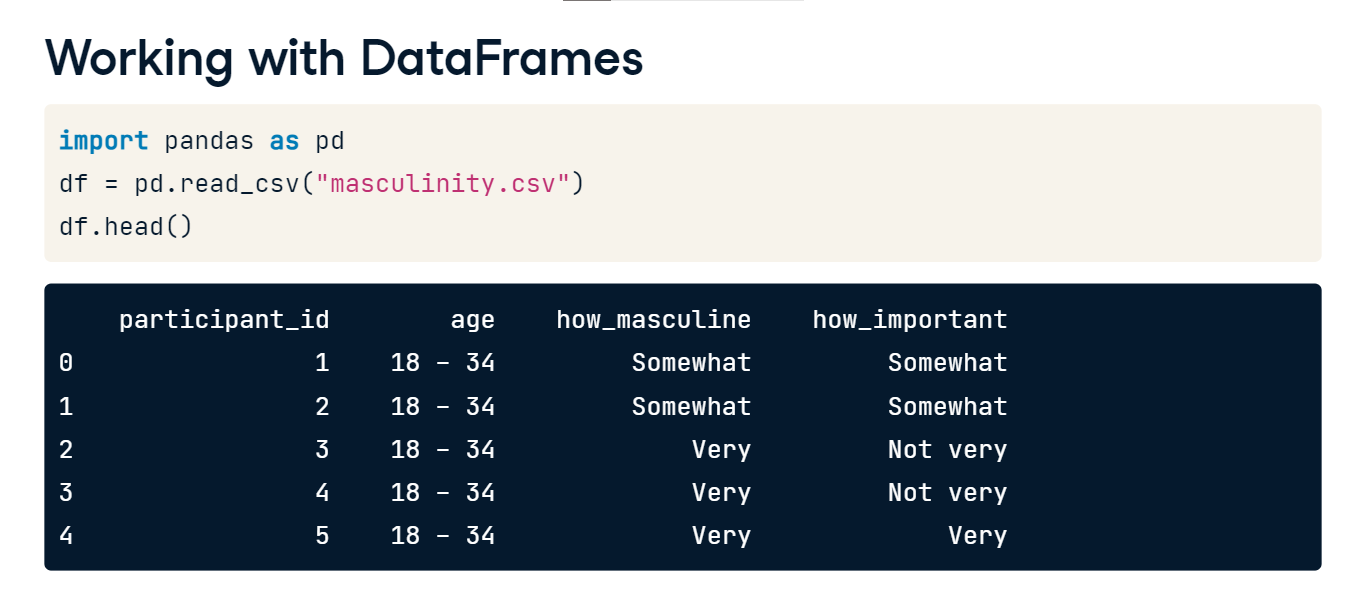
**What is pandas?**

pandas is a python library for data analysis. It can easily read datasets from many types of files including csv and txt files. pandas supports several types of data structures, but the most common one is the DataFrame object. When you read in a dataset with pandas, you will create a DataFrame.

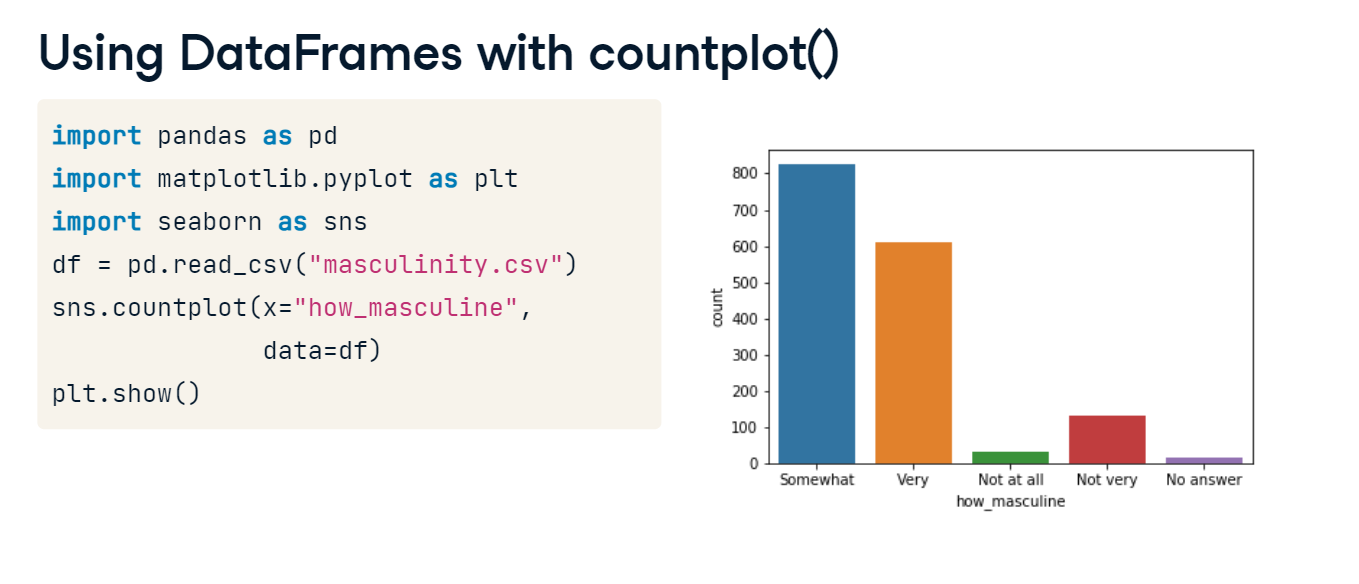
**Working with DataFrames**

Let's look at an example. First, import the pandas library as "pd". Then, use the "read\_csv" function to read the csv file named "masculinity dot csv" and create a pandas DataFrame called "df". Calling "head" on the DataFrame will show us its first five rows. This dataset contains the results of a survey of adult men. We can see that it has four columns: "participant\_id"; "age"; "how\_masculine", which is that person's response to the question "how masculine or 'manly' do you feel?"; and "how\_important", which is the response to the question "how important is it to you that others see you as masculine?"

**Using DataFrames**

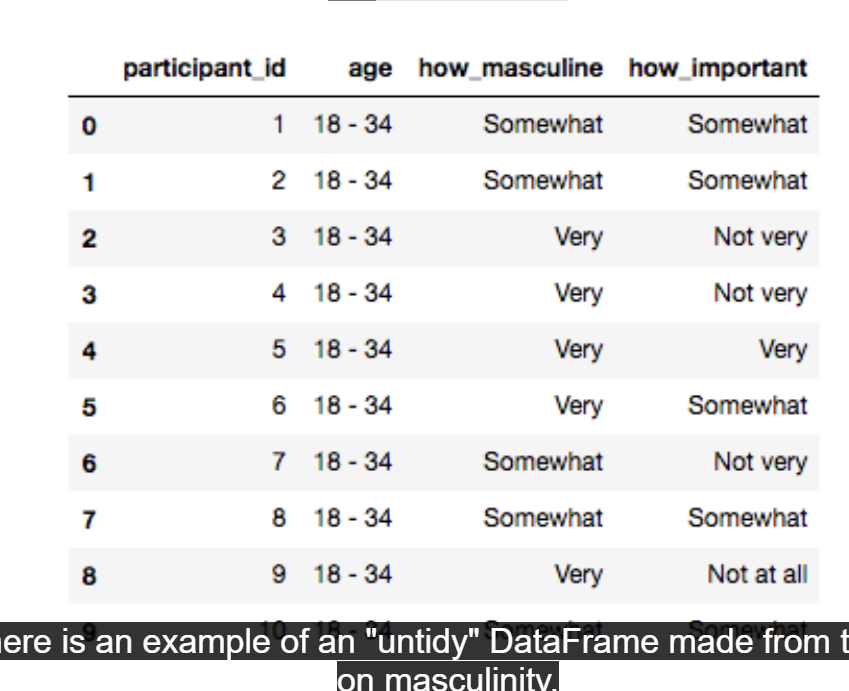
**with countplot()**

Now let's look at how to make a count plot with a DataFrame instead of a list of data. The first thing we'll do is import pandas, Matplotlib and Seaborn as we have in past examples. Then, we'll create a pandas DataFrame called "df" from the masculinity csv file. To create a count plot with a pandas DataFrame column instead of a list of data, set x equal to the name of the column in the DataFrame - in this case, we'll use the "how\_masculine" column. Then, we'll set the data parameter equal to our DataFrame, "df". After calling "plt dot show", we can see that we have a nice count plot of the values in the "how\_masculine" column of our data. This plot shows us that the most common response to the question "how masculine or 'manly' do you feel?" is "somewhat", with "very" being the second most common response. Note also that because we're using a named column in the DataFrame, Seaborn automatically adds the name of the column as the x-axis label at the bottom.



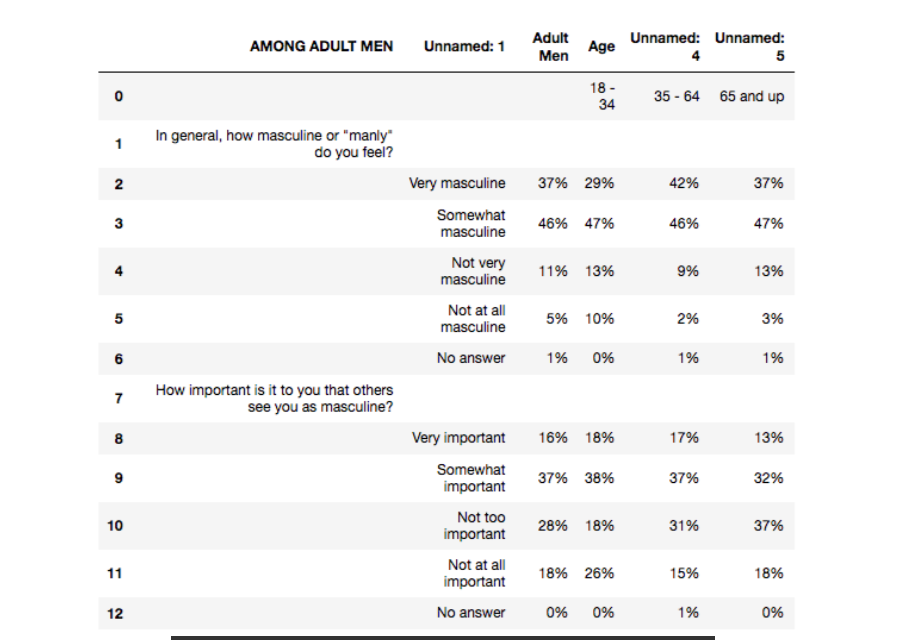
**"Tidy" data**

Let's pause for an important note here. Seaborn works great with pandas DataFrames, but only if the DataFrame is "tidy". "Tidy data" means that each observation has its own row and each variable has its own column. The "masculinity" DataFrame shown here is tidy because each row is a survey response with one answer to each survey question in each column. Making a count plot with the "how masculine" column works just like passing in a list of that column's values.



**"Untidy" data**

In contrast, here is an example of an "untidy" DataFrame made from the same survey on masculinity. In this untidy DataFrame, notice how each row doesn't contain the same information. Row 0 contains the age categories, rows 1 and 7 contain the question text, and the other rows contain summary data of the responses. This will not work well with Seaborn. Unlike the tidy DataFrame, values in the "Age" column don't look like a list of age categories for each observation. Transforming untidy DataFrames into tidy ones is possible, but it's not in scope for this course. There are other DataCamp courses that can teach you how to do this.



**Adding a third variable with hue**

We saw in the last lesson that a really nice advantage of Seaborn is that it works well with pandas DataFrames. In this lesson, we'll see another big advantage that Seaborn offers: the ability to quickly add a third variable to your plots by adding color.

**Tips dataset**

To showcase this cool feature in Seaborn, we'll be using Seaborn's built-in tips dataset. You can access it by using the "load dataset" function in Seaborn and passing in the name of the dataset. These are the first five rows of the tips dataset. This dataset contains one row for each table served at a restaurant and has information about things like the bill amount, how many people were at the table, and when the table was served. Let's explore the relationship between the "total\_bill" and "tip" columns using a scatter plot.

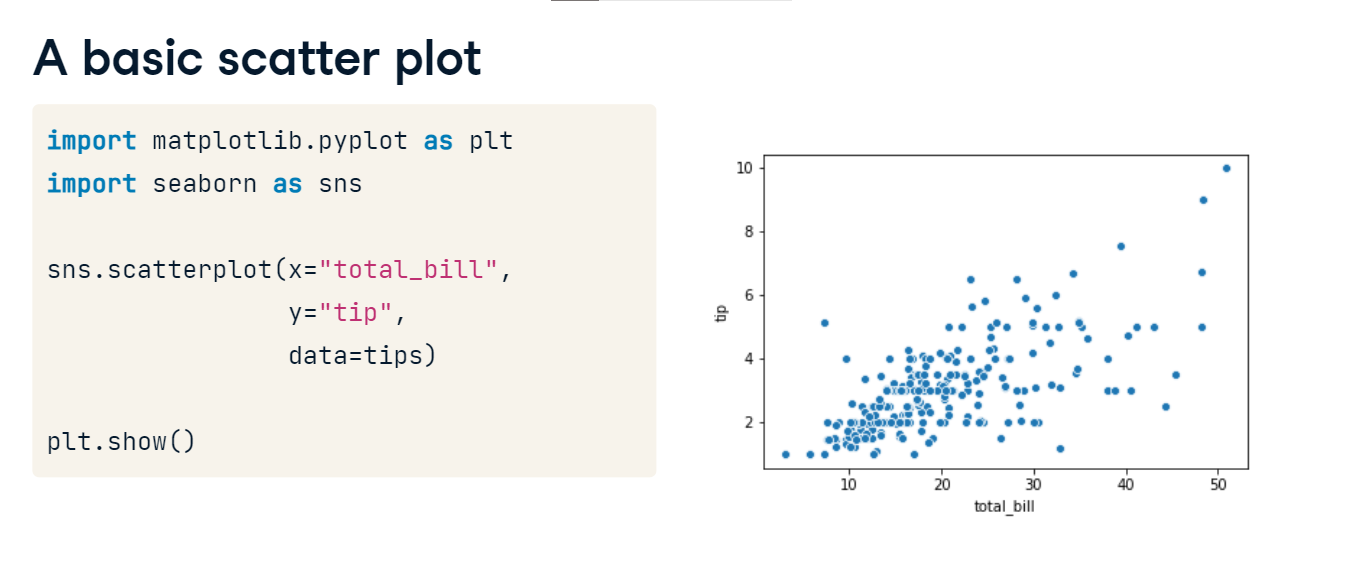
1. 1 Waskom, M. L. (2021). seaborn: statistical data visualization. https://seaborn.pydata.org/

****

**A basic scatter plot**

Here is the code to generate it. The total bill per table (in dollars) is on the x-axis, and the total tip (in dollars) is on the y-axis. We can see from this plot that larger bills are associated with larger tips. What if we want to see which of the data points are smokers versus non-smokers? Seaborn makes this super easy.

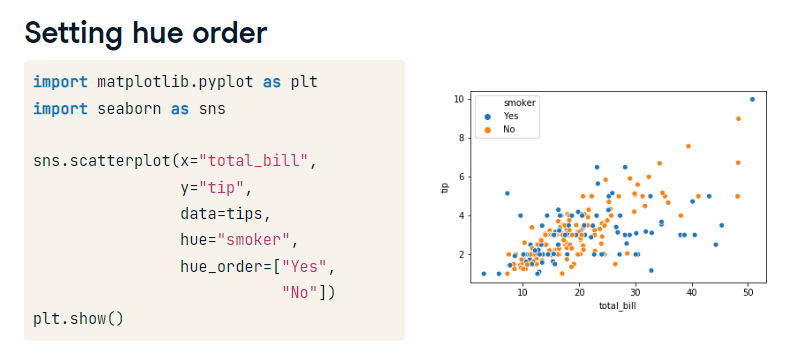
1. 1 Waskom, M. L. (2021). seaborn: statistical data visualization. https://seaborn.pydata.org/

****

**A scatter plot with hue**

You can set the "hue" parameter equal to the DataFrame column name "smoker" and then Seaborn will automatically color each point by whether they are a smoker. Plus, it will add a legend to the plot automatically! If you don't want to use pandas, you can set it equal to a list of values instead of a column name.

1. 1 Waskom, M. L. (2021). seaborn: statistical data visualization. https://seaborn.pydata.org/

****

**Setting hue order**

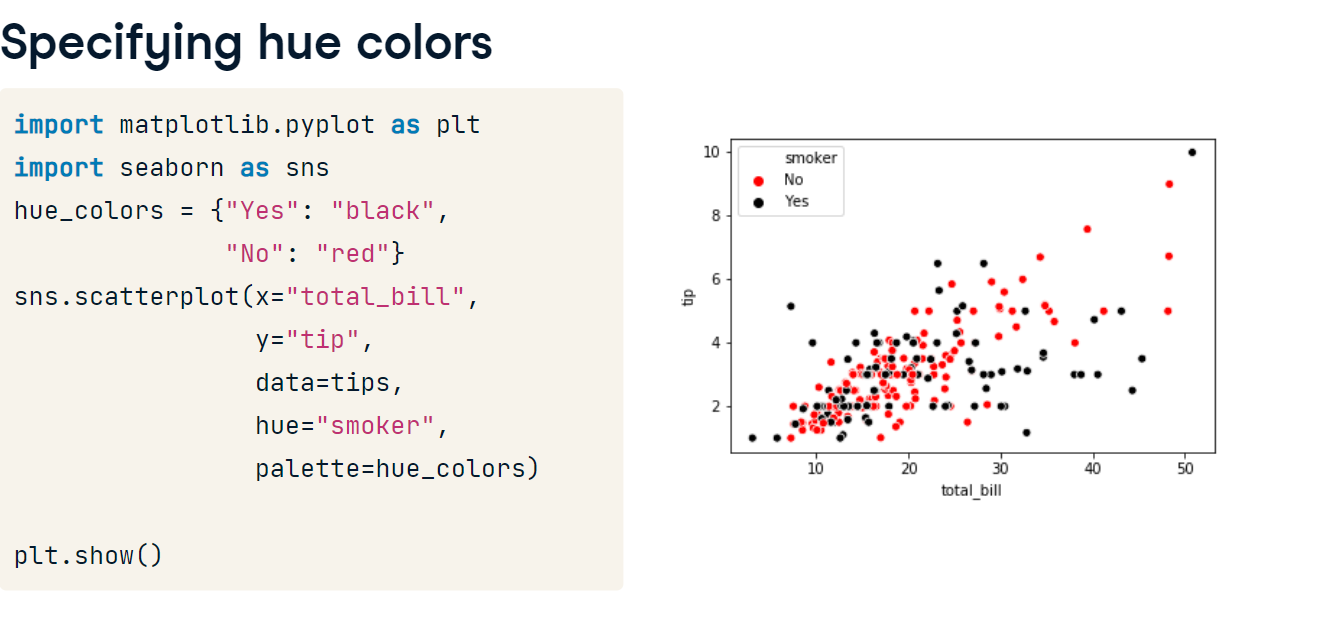
Hue also allows you to assert more control over the ordering and coloring of each value. The "hue order" parameter takes in a list of values and will set the order of the values in the plot accordingly. Notice how the legend for smoker now lists "yes" before "no".

1. 1 Waskom, M. L. (2021). seaborn: statistical data visualization. https://seaborn.pydata.org/

**Specifying hue colors**

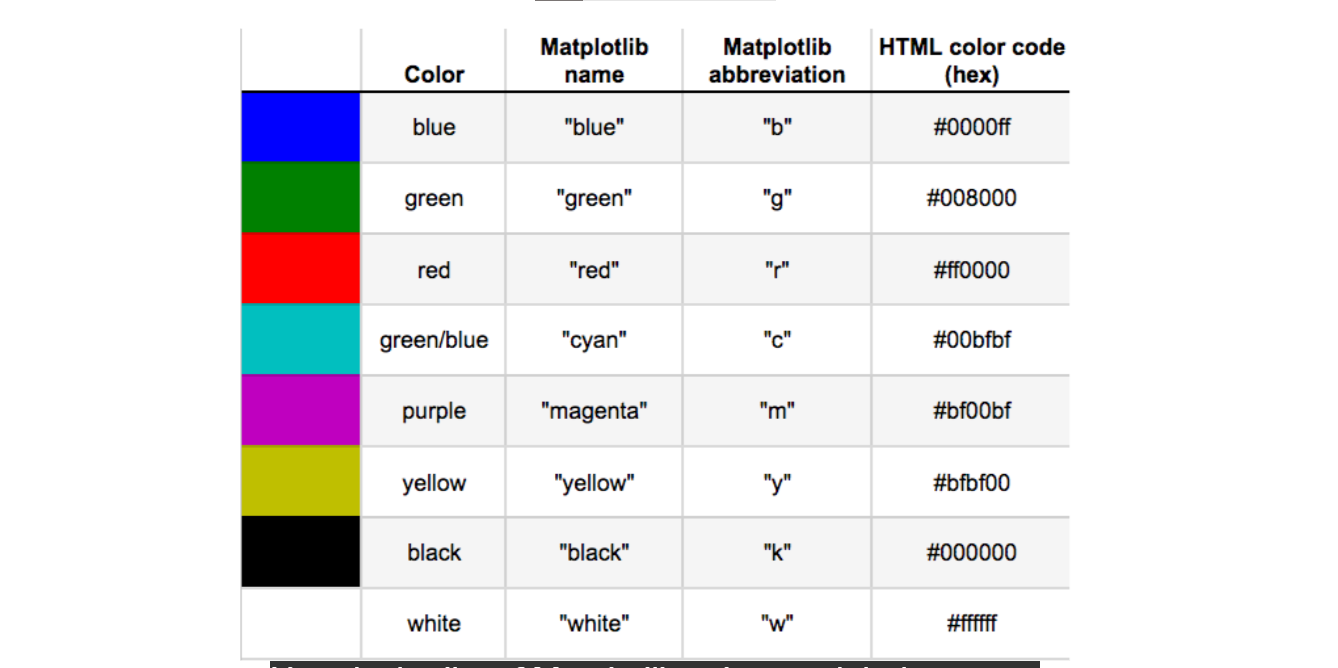
You can also control the colors assigned to each value using the "palette" parameter. This parameter takes in a dictionary, which is a data structure that has key-value pairs. This dictionary should map the variable values to the colors you want to represent the value. Here, we create a dictionary called "hue colors" that maps the value "Yes" to the color black and the value "No" to the color red. When we set hue equal to "smoker" and the palette parameter equal to this dictionary, we have a scatter plot where smokers are represented with black dots and non-smokers are represented with red dots.

1. 1 Waskom, M. L. (2021). seaborn: statistical data visualization. https://seaborn.pydata.org/

****

**Color options**

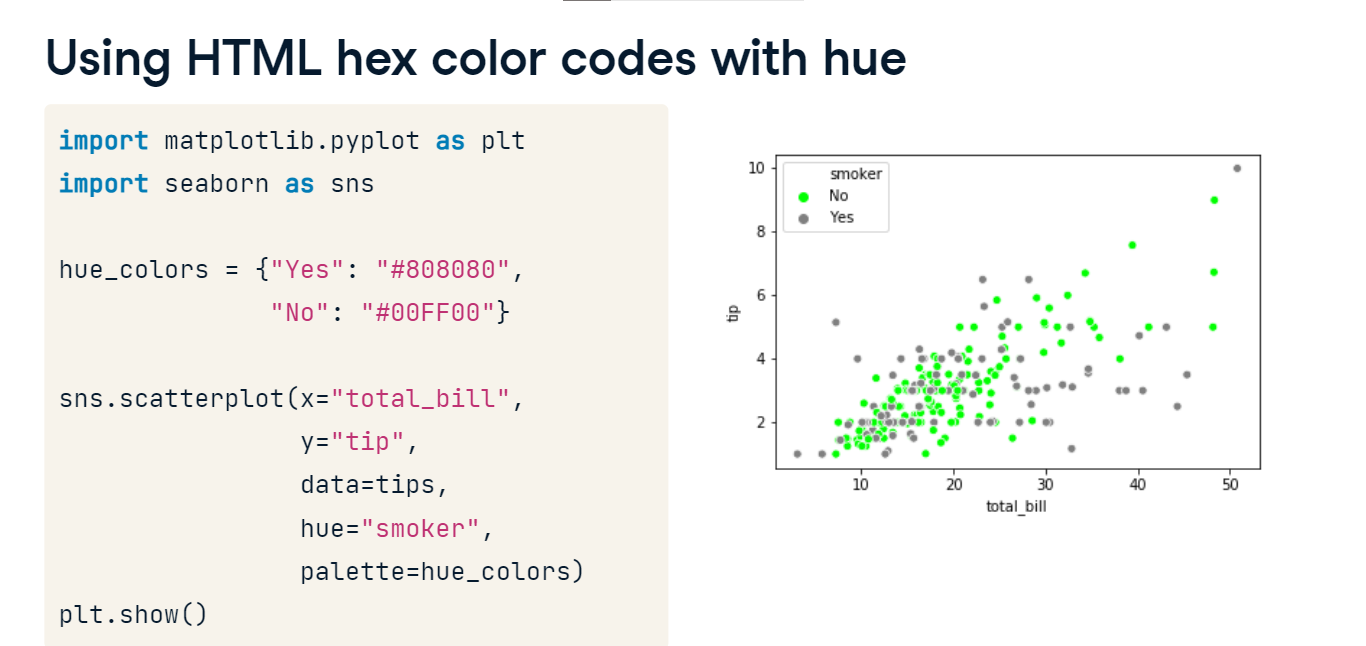
In the last example, we used the words "black" and "red" to define what the hue colors should be. This only works for a small set of color names that are defined by Matplotlib. Here is the list of Matplotlib colors and their names. Note that you can use a single-letter Matplotlib abbreviation instead of the full name. You can also use an HTML color hex code instead of these Matplotlib color names, which allows you to choose any color you want to.

****

**Using HTML hex color codes with hue**

Here's an example using HTML hex codes. Make sure you put the hex codes in quotes with a pound sign at the beginning.

1. 1 Waskom, M. L. (2021). seaborn: statistical data visualization. https://seaborn.pydata.org/

****

**Using hue with count plots**

As a final note, hue is available in most of Seaborn's plot types. For example, this count plot shows the number of observations we have for smokers versus non-smokers, and setting "hue" equal to "sex" divides these bars into subgroups of males versus females. From this plot, we can see that males outnumber females among both smokers and non-smokers in this dataset.

1. 1 Waskom, M. L. (2021). seaborn: statistical data visualization. https://seaborn.pydata.org/

