**Introduction to relational plots and subplots**

Many questions in data science are centered around describing the relationship between two quantitative variables. Seaborn calls plots that visualize this relationship "relational plots".

**Questions about quantitative variables**

So far we've seen several examples of questions about the relationship between two quantitative variables, and we answered them with scatter plots. These examples include: "do taller people tend to weigh more?"

**Questions about quantitative variables**

"what's the relationship between the number of absences a student has and their final grade?"

**Questions about quantitative variables**

and "how does a country's GDP relate to the percent of the population that can read and write?" Because they look at the relationship between two quantitative variables, these scatter plots are all considered relational plots.

**Visualizing subgroups**

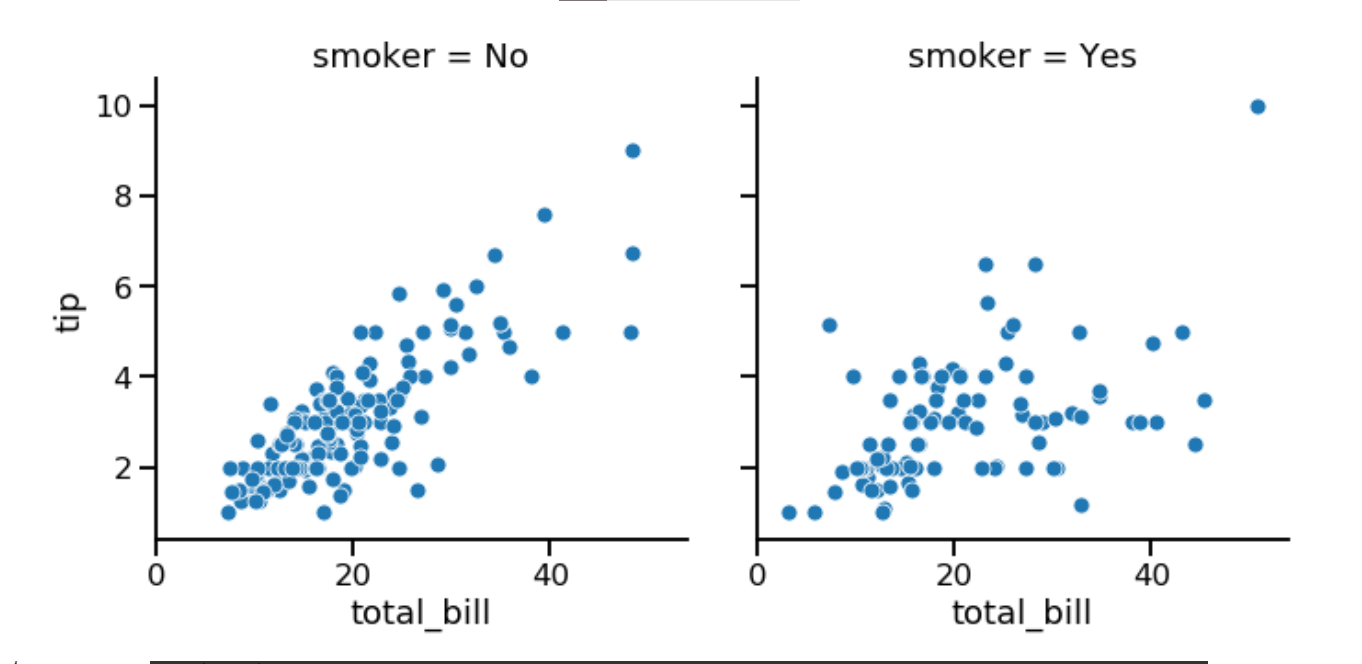
While looking at a relationship between two variables at a high level is often informative, sometimes we suspect that the relationship may be different within certain subgroups. In the last chapter, we started to look at subgroups by using the "hue" parameter to visualize each subgroup using a different color on the same plot.

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

**Visualizing subgroups**

In this lesson, we'll try out a different method: creating a separate plot per subgroup.

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

****

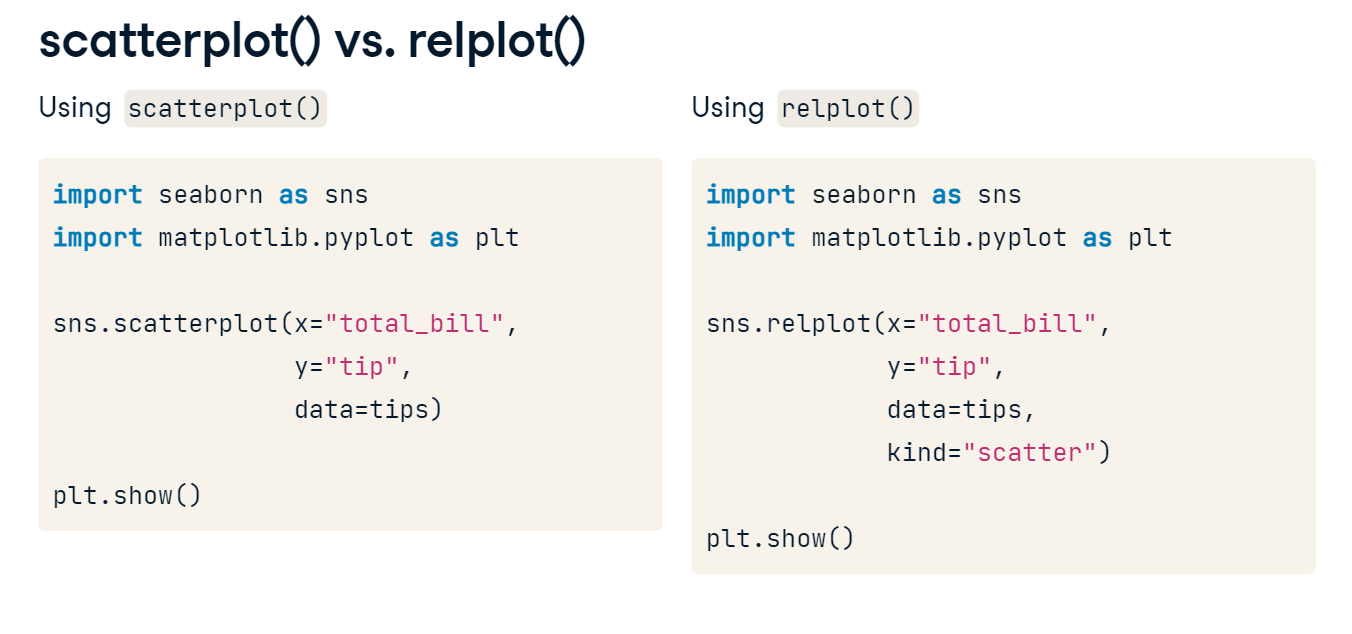
**Introducing relplot()**

To do this, we're going to introduce a new Seaborn function: "relplot()". "relplot()" stands for "relational plot" and enables you to visualize the relationship between two quantitative variables using either scatter plots or line plots. You've already seen scatter plots, and you'll learn about line plots later in this chapter. Using "relplot()" gives us a big advantage: the ability to create subplots in a single figure. Because of this advantage, we'll be using "relplot()" instead of "scatterplot()" for the rest of the course.

**scatterplot() vs. relplot()**

Let's return to our scatter plot of total bill versus tip amount from the tips dataset. On the left, we see how to create a scatter plot with the "scatterplot" function. To make it with "relplot()" instead, we change the function name to "relplot()" and use the "kind" parameter to specify what kind of relational plot to use - scatter plot or line plot. In this case, we'll set kind equal to the word "scatter".

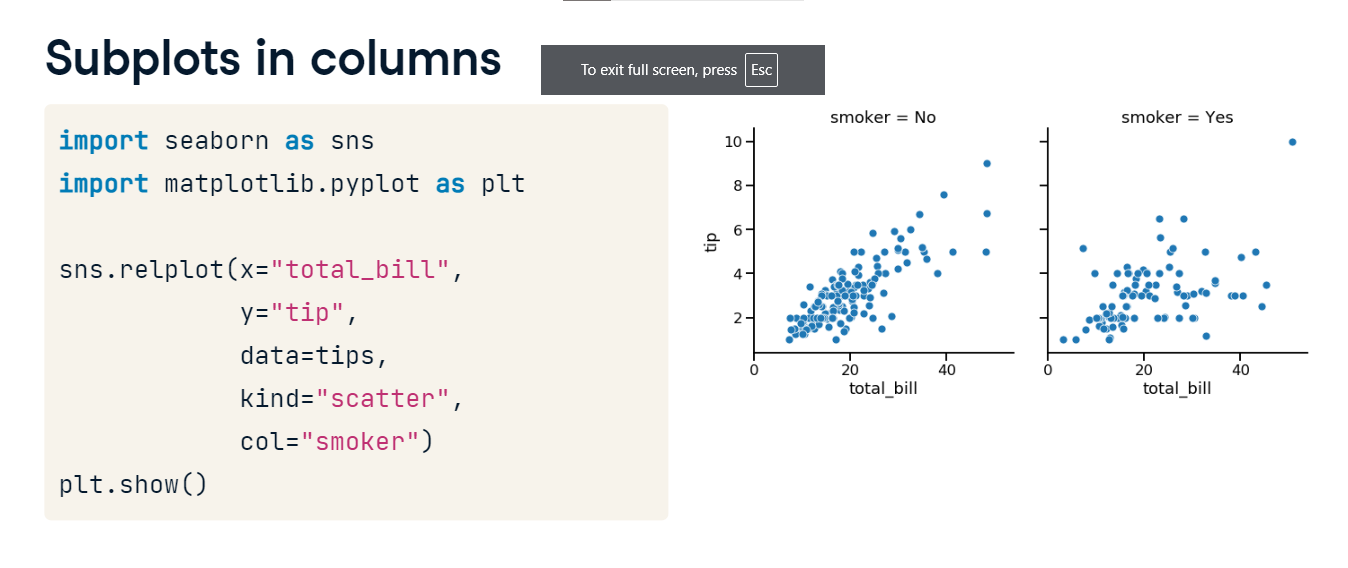
1. 1 Waskom, M. L. (2021). seaborn: statistical data visualization. https://seaborn.pydata.org/

****

**Subplots in columns**

By setting "col" equal to "smoker", we get a separate scatter plot for smokers and non-smokers, arranged horizontally in columns.

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

****

**Subplots in rows**

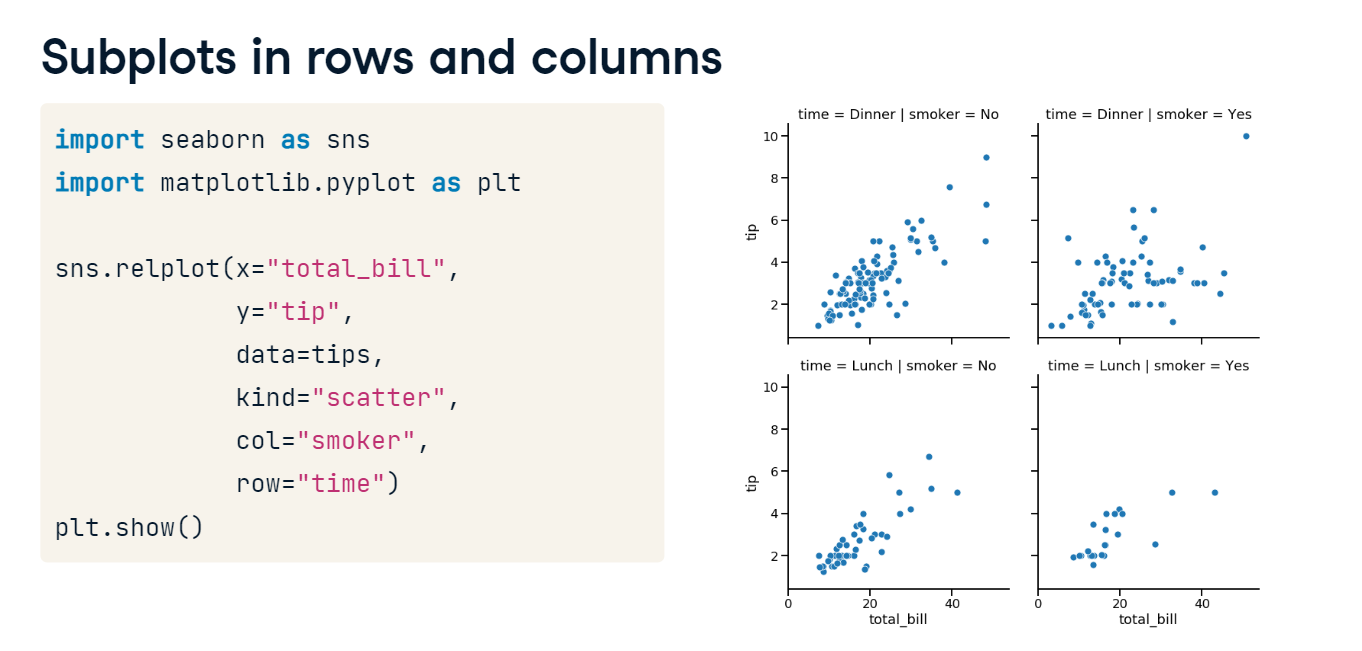
If you want to arrange these vertically in rows instead, you can use the "row" parameter instead of "col".

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

**Subplots in rows and columns**

It is possible to use both "col" and "row" at the same time. Here, we set "col" equal to smoking status and "row" equal to the time of day (lunch or dinner). Now we have a subplot for each combination of these two categorical variables.

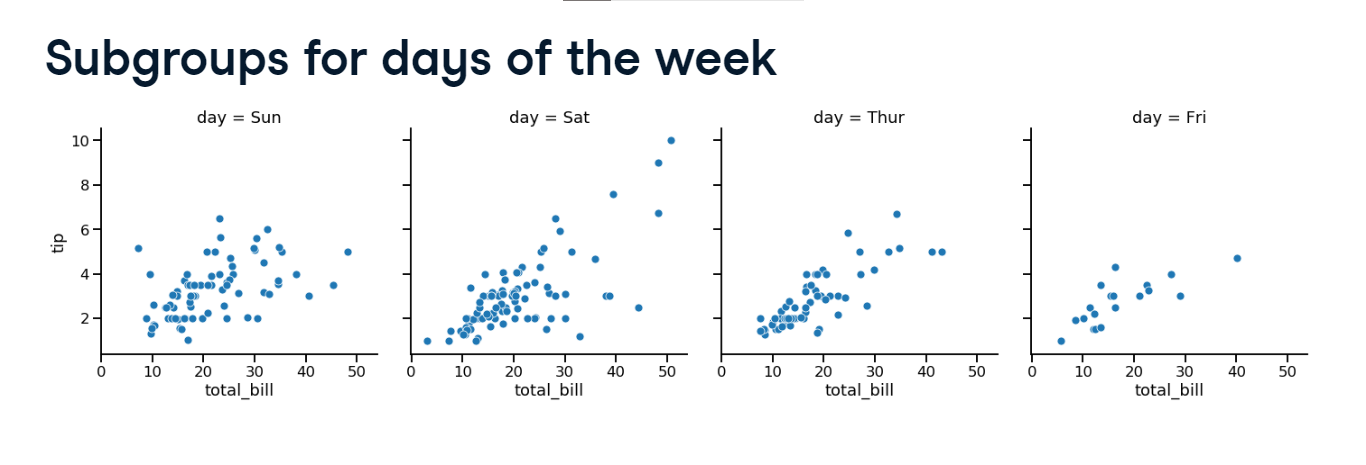
1. 1 Waskom, M. L. (2021). seaborn: statistical data visualization. https://seaborn.pydata.org/

****

**Subgroups for days of the week**

As another example, let's look at subgroups based on day of the week. There are four subplots here, which can be a lot to show in a single row. To address this, you can use the "col\_wrap" parameter to specify how many subplots you want per row.

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



**Wrapping columns**

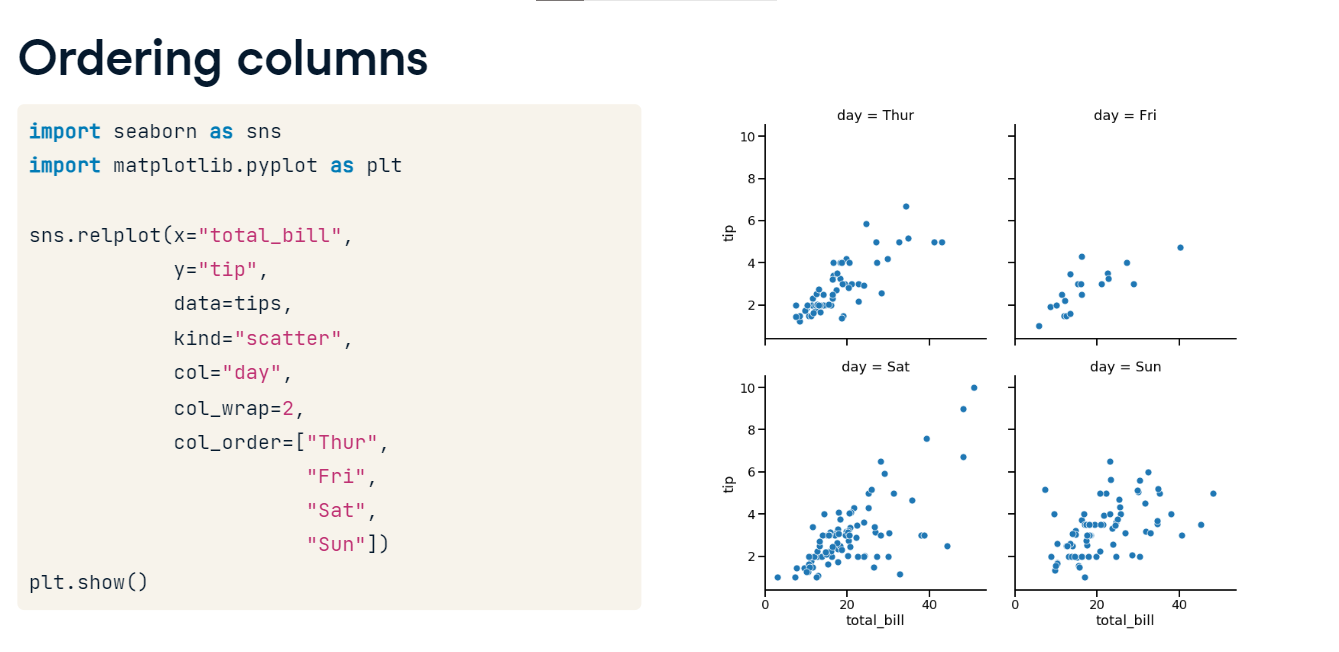
Here, we set "col\_wrap" equal to two plots per row.

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

**Ordering columns**

We can also change the order of the subplots by using the "col\_order" and "row\_order" parameters and giving it a list of ordered values.

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



**Customizing scatter plots**

So far, we've only scratched the surface of what we're able to do with scatter plots in Seaborn.

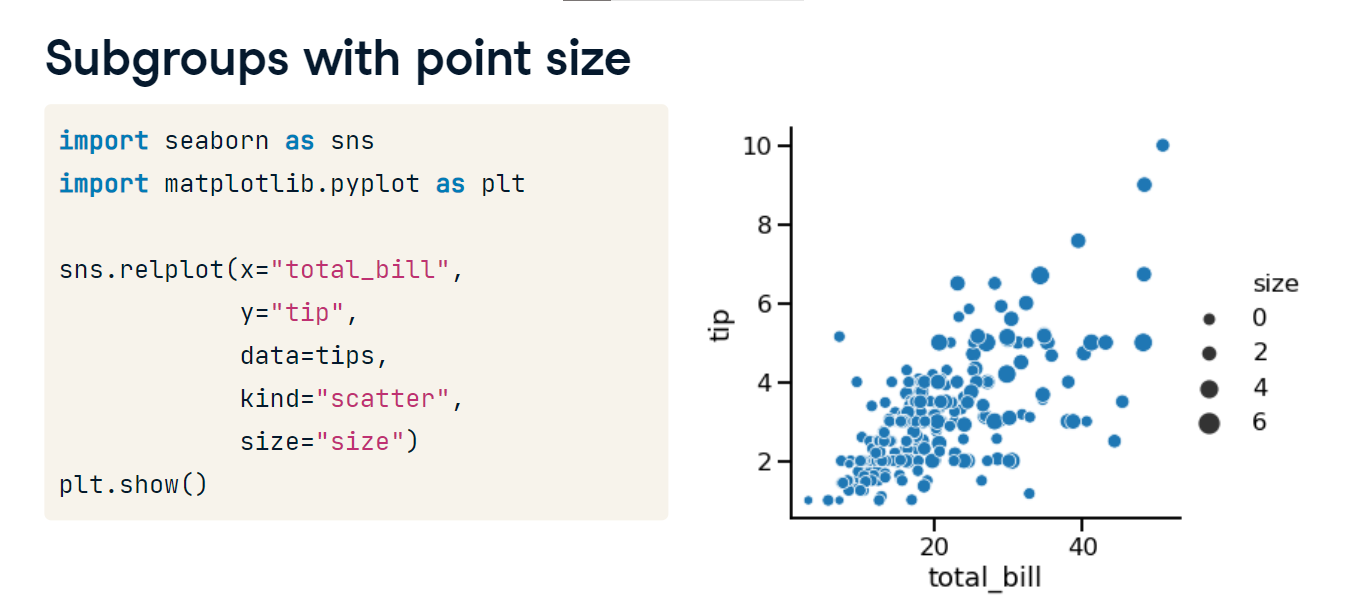
**Scatter plot overview**

As a reminder, scatter plots are a great tool for visualizing the relationship between two quantitative variables. We've seen a few ways to add more information to them as well, by creating subplots or plotting subgroups with different colored points. In addition to these, Seaborn allows you to add more information to scatter plots by varying the size, the style, and the transparency of the points. All of these options can be used in both the "scatterplot()" and "relplot()" functions, but we'll continue to use "relplot()" for the rest of the course since it's more flexible and allows us to create subplots. For the rest of this lesson, we'll use the tips dataset to learn how to use each customization and cover best practices for deciding which customizations to use.

**Subgroups with point size**

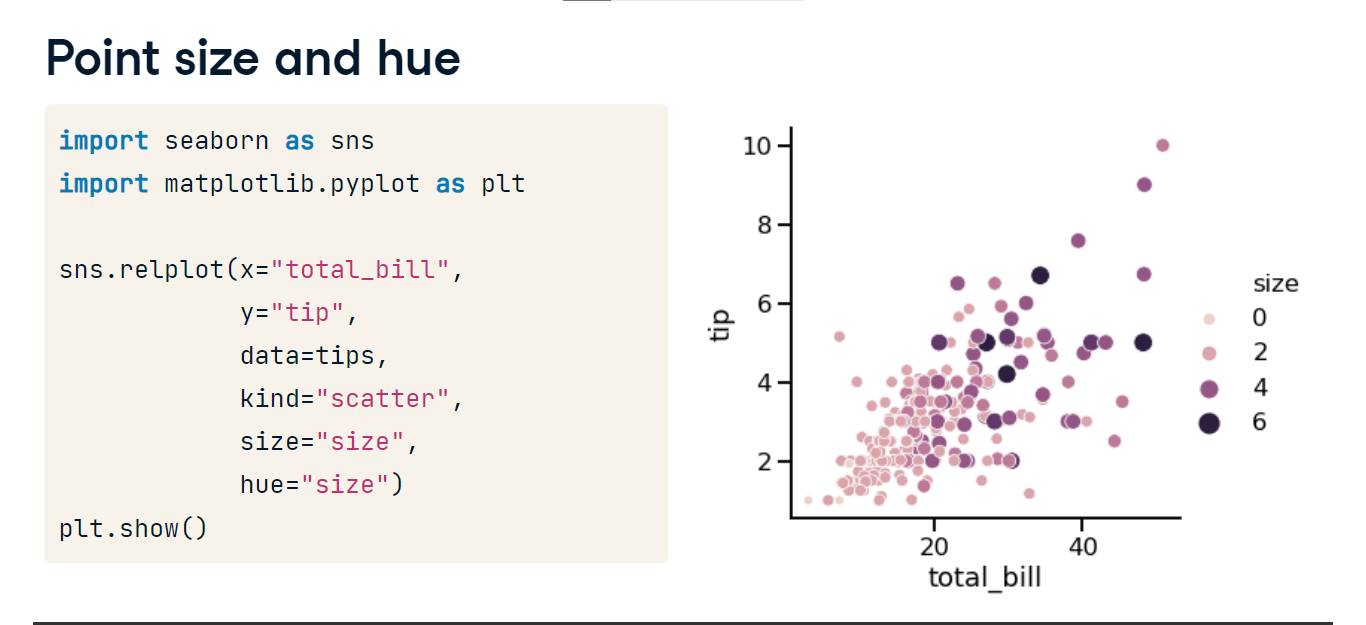
The first customization we'll talk about is point size. Here, we're creating a scatter plot of total bill versus tip amount. We want each point on the scatter plot to be sized based on the number of people in the group, with larger groups having bigger points on the plot. To do this, we'll set the "size" parameter equal to the variable name "size" from our dataset. As this example demonstrates, varying point size is best used if the variable is either a quantitative variable or a categorical variable that represents different levels of something, like "small", "medium", and "large". This plot is a bit hard to read because all of the points are of the same color.

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



**Point size and hue**

We can make it easier by using the "size" parameter in combination with the "hue" parameter. To do this, set "hue" equal to the variable name "size". Notice that because "size" is a quantitative variable, Seaborn will automatically color the points different shades of the same color instead of different colors per category value like we saw in previous plots. Now larger groups have both larger and darker points, which provides better contrast and makes the plot easier to read.

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

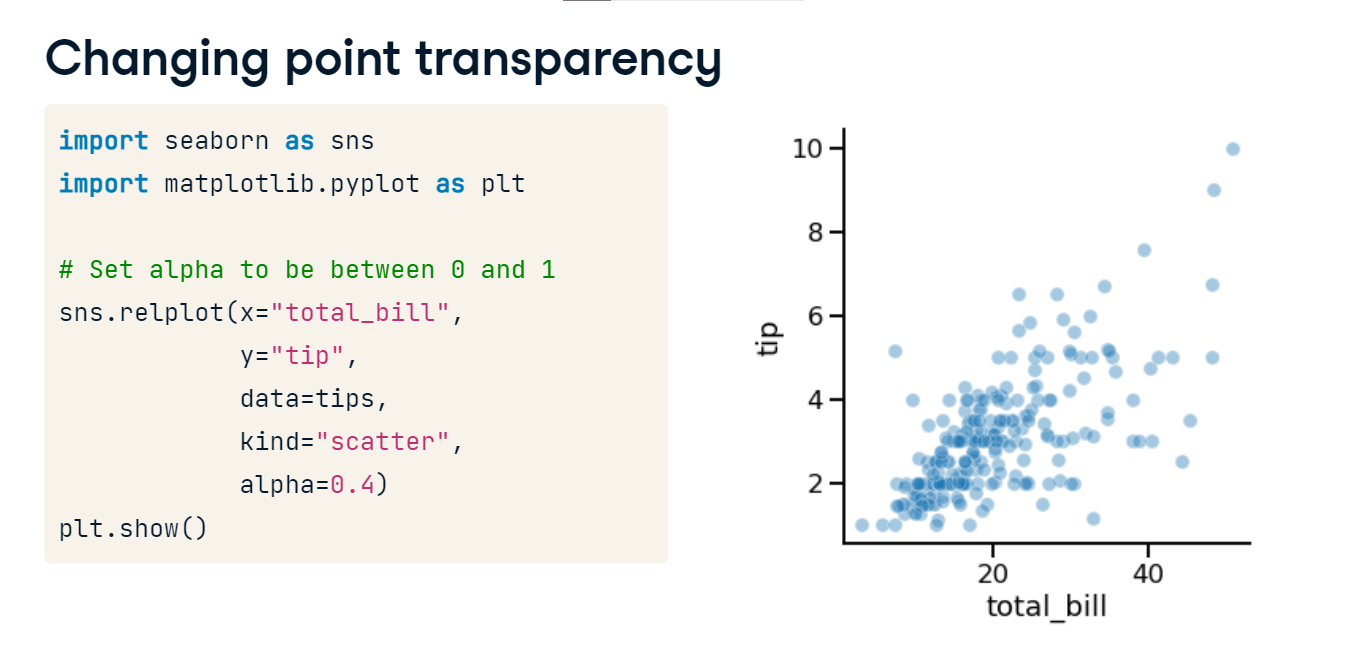
**Subgroups with point style**

The next customization we'll look at is the point style. Setting the "style" parameter to a variable name will use different point styles for each value of the variable. Here's a scatter plot we've seen before, where we use "hue" to create different colored points based on smoking status. Setting "style" equal to "smoker" allows us to better distinguish these subgroups by plotting smokers with a different point style in addition to a different color.

**Changing point transparency**

The last customization we'll look at is point transparency. Setting the "alpha" parameter to a value between 0 and 1 will vary the transparency of the points in the plot, with 0 being completely transparent and 1 being completely non-transparent. Here, we've set "alpha" equal to 0.4. This customization can be useful when you have many overlapping points on the scatter plot, so you can see which areas of the plot have more or less observations.

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



**Introduction to line plots**

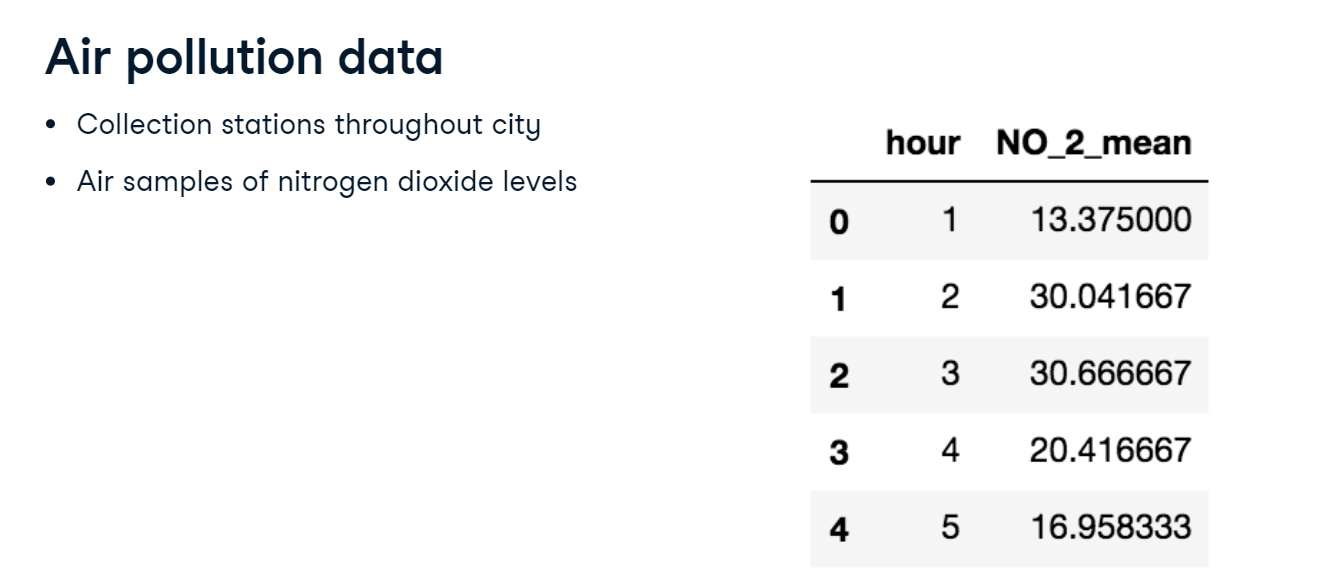
Hello! In this video we'll dive into a new type of relational plot: line plots.

**What are line plots?**

In Seaborn, we have two types of relational plots: scatter plots and line plots. While each point in a scatter plot is assumed to be an independent observation, line plots are the visualization of choice when we need to track the same thing over time. A common example is tracking the value of a company's stock over time, as shown here.

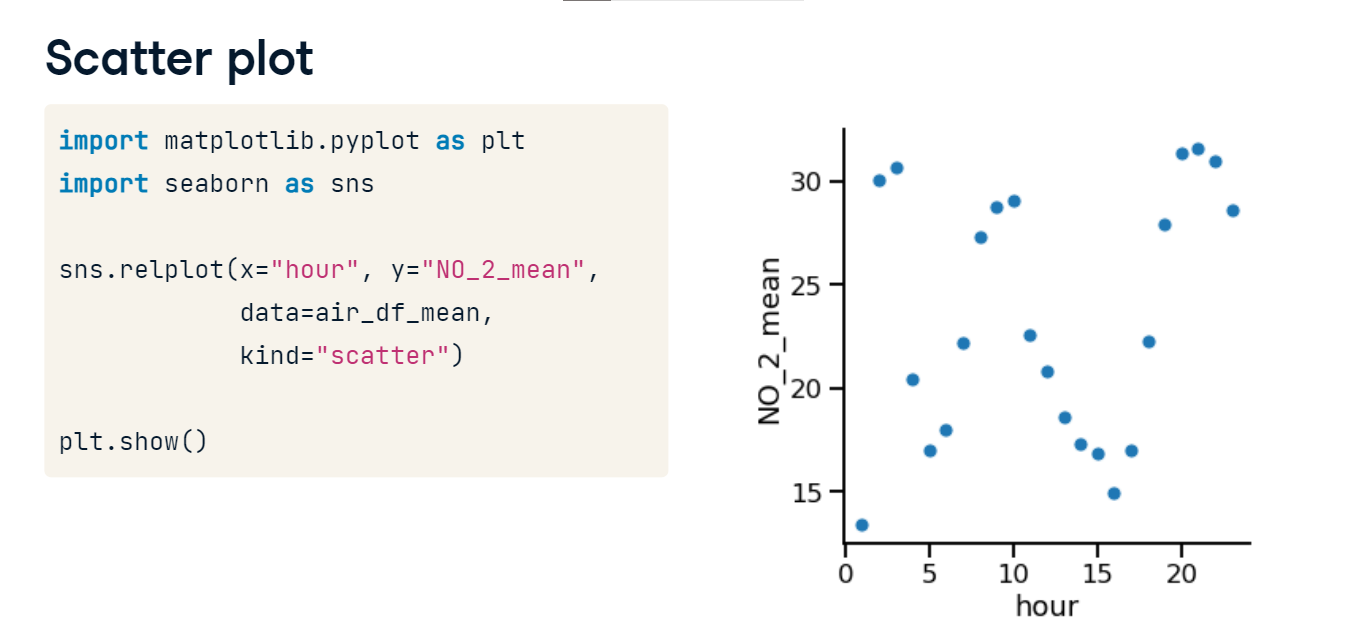
**Air pollution data**

In this video, we'll be using data on the levels of air pollution in a city. There are many air collection stations around the city, each measuring the nitrogen dioxide level every hour for a single day. Long-term exposure to high levels of nitrogen dioxide can cause chronic lung diseases. Let's begin with the simple case where we have one data point per x-value. Here we have one row per hour over the course of the day with the average nitrogen dioxide level across all the stations in a column called "NO\_2\_mean".



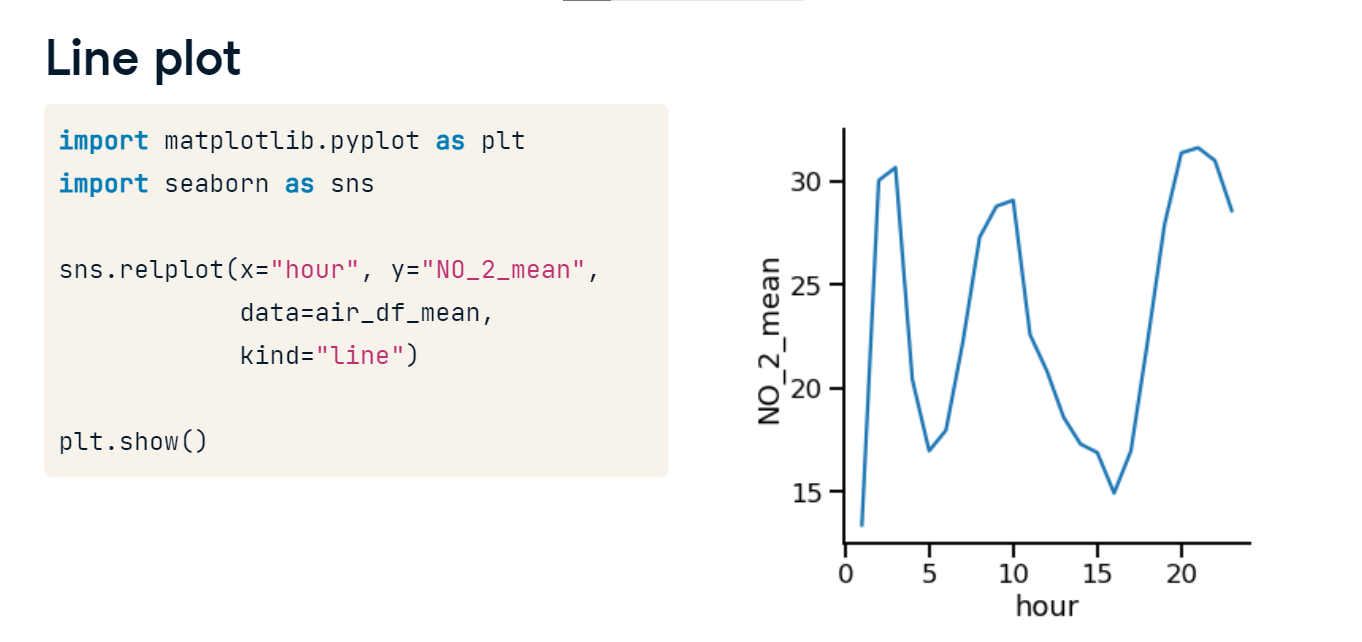
**Scatter plot**

This is a scatter plot with the average nitrogen dioxide level on the y-axis and the hour of the day on the x-axis. We're tracking the same thing over time, so a line plot would be a better choice.



**Line plot**

By specifying "kind" equals "line", we can create a line plot and more easily see how the average nitrogen dioxide level fluctuates throughout the day.



**Subgroups by location**

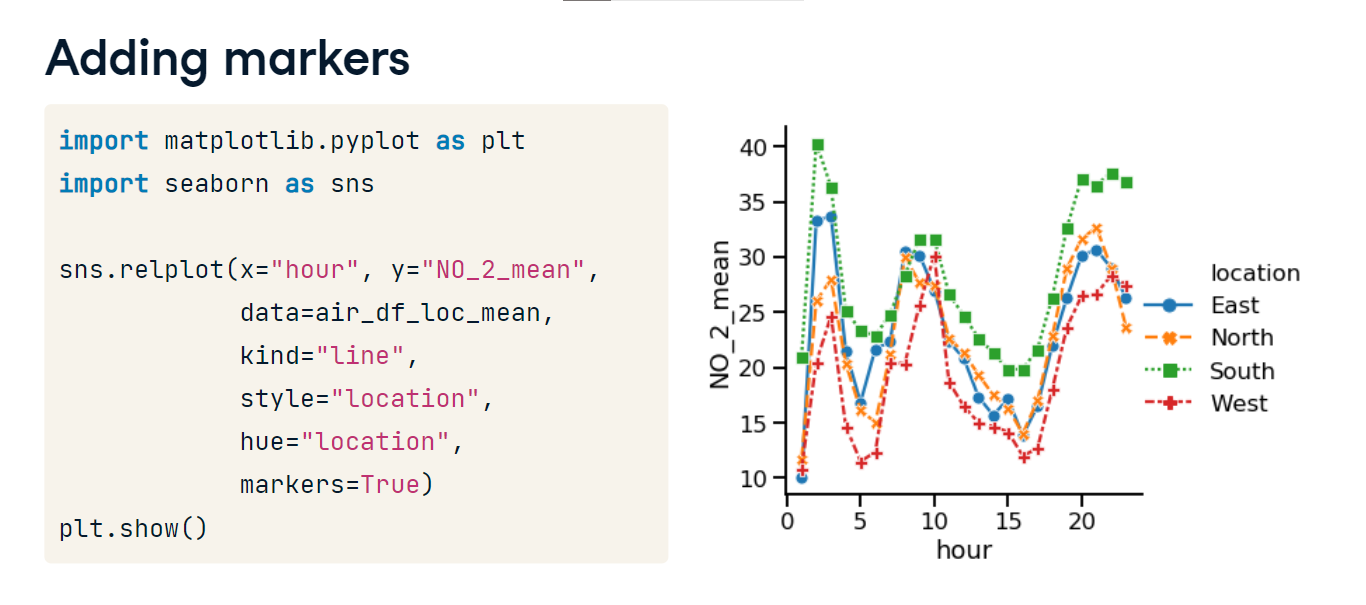
We can also track subgroups over time with line plots. Here we have the average nitrogen dioxide level for each region (North, South, East, and West) for each hour in the day.

**Subgroups by location**

Setting the "style" and "hue" parameters equal to the variable name "location" creates different lines for each region that vary in both line style and color. Here, we can see that the South region tends to have slightly higher average nitrogen dioxide levels compared to the other regions.

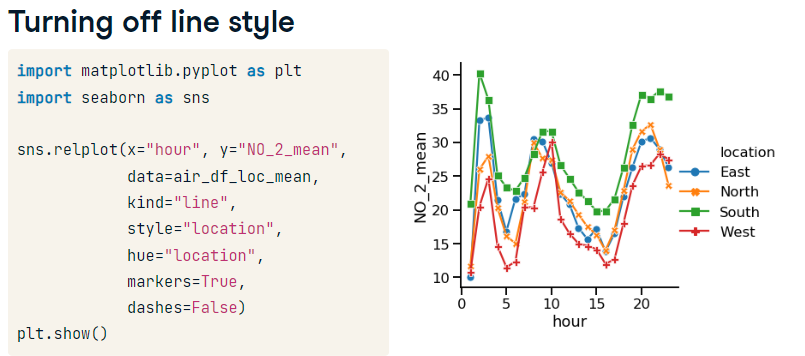
**Adding markers**

Setting the "markers" parameter equal to "True" will display a marker for each data point. The marker will vary based on the subgroup you've set using the "style" parameter.



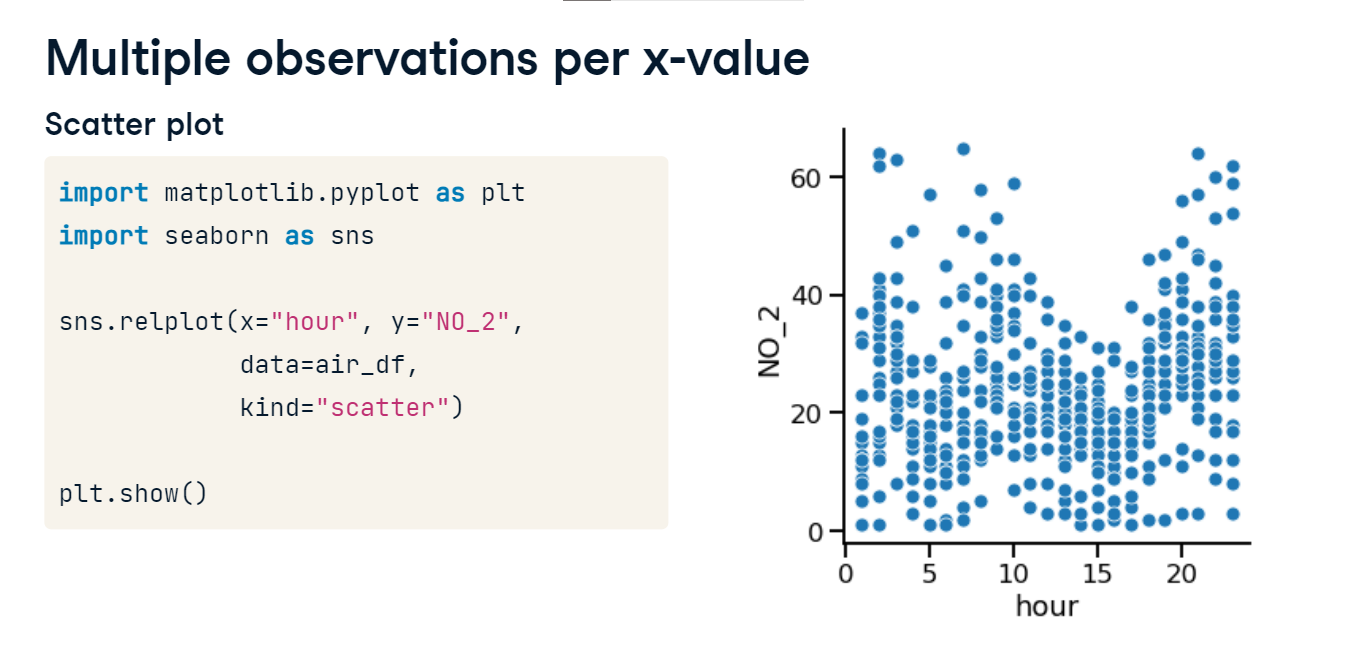
**Turning off line style**

If you don't want the line styles to vary by subgroup, set the "dashes" parameter equal to "False".



**Multiple observations per x-value**

Line plots can also be used when you have more than one observation per x-value. This dataset has a row for each station that is taking a measurement every hour.

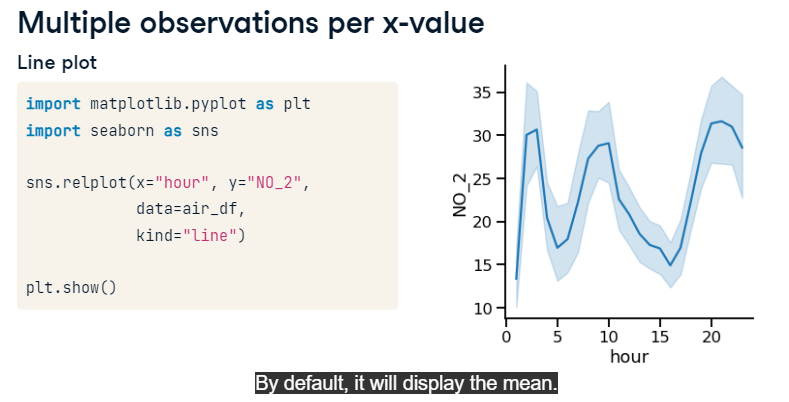


**Multiple observations per x-value**

This is the scatter plot, displaying one point per observation.

**Multiple observations per x-value**

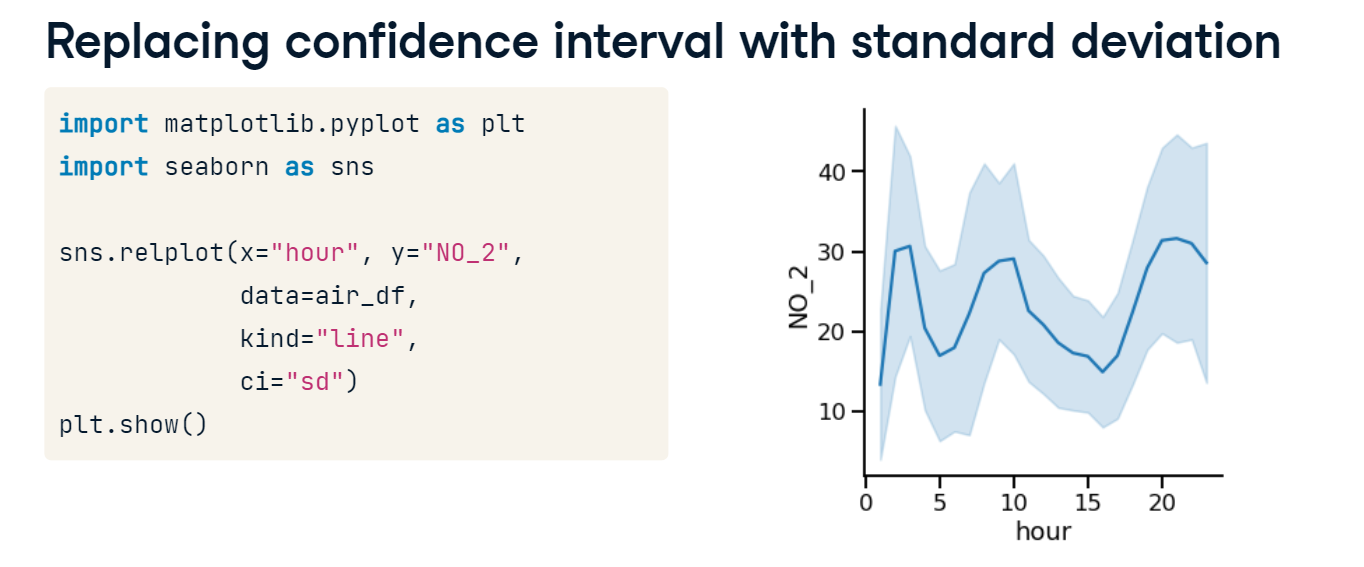
This is the line plot. If a line plot is given multiple observations per x-value, it will aggregate them into a single summary measure. By default, it will display the mean.

**Multiple observations per x-value**

Notice that Seaborn will automatically calculate a confidence interval for the mean, displayed by the shaded region. Assuming the air collection stations were randomly placed throughout the city, this dataset is a random sample of the nitrogen dioxide levels across the whole city. This confidence interval tells us that based on our sample, we can be 95% confident that the average nitrogen dioxide level for the whole city is within this range. Confidence intervals indicate the uncertainty we have about what the true mean is for the whole city. To learn more about confidence intervals, you can check out DataCamp's statistics courses.

**Replacing confidence interval with standard deviation**

Instead of visualizing a confidence interval, we may want to see how varied the measurements of nitrogen dioxide are across the different collection stations at a given point in time. To visualize this, set the "ci" parameter equal to the string "sd" to make the shaded area represent the standard deviation, which shows the spread of the distribution of observations at each x value.



**Turning off confidence interval**

We can also turn off the confidence interval by setting the "ci" parameter equal to "None".