Count plots and bar plots

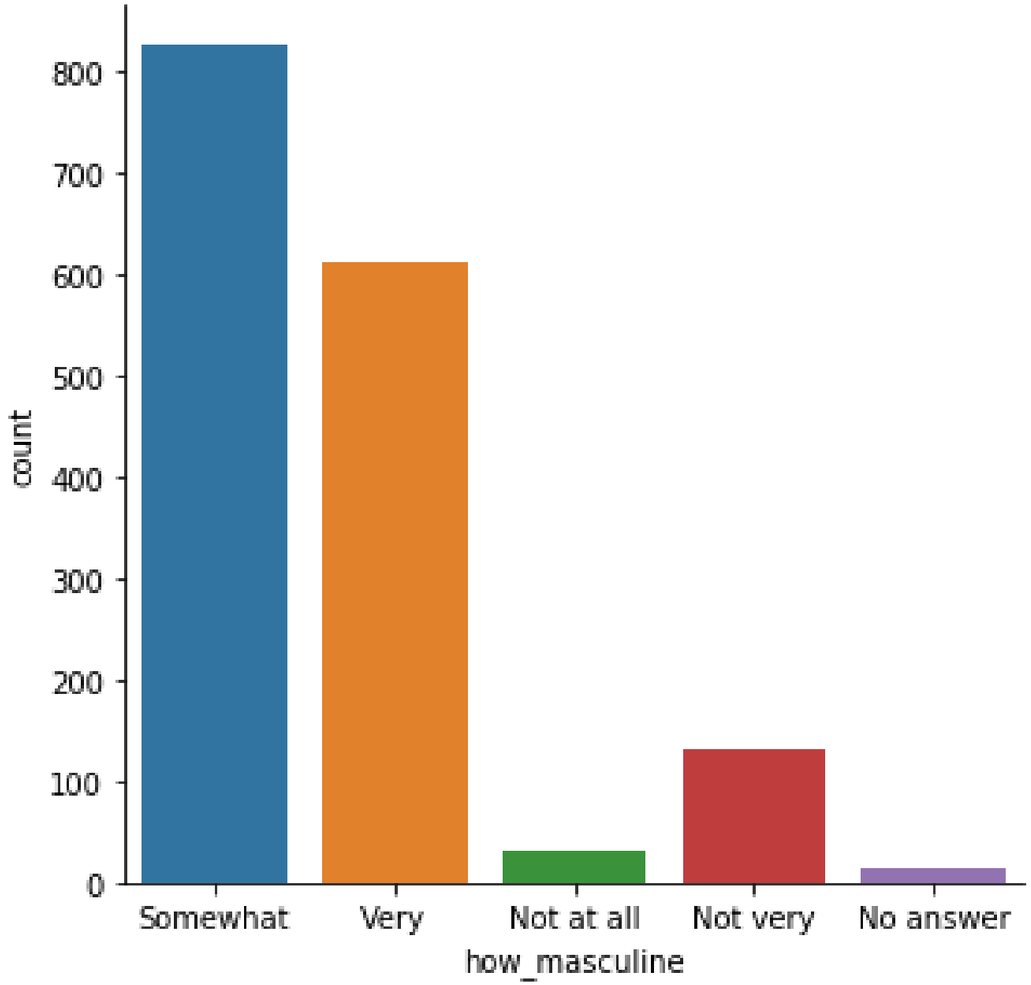
INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Erin Case
Data Scientist

Categorical plots

- Examples: count plots, bar plots
- Involve a categorical variable
- Comparisons between groups

2. Categorical plots
Count plots and bar plots are two types of visualizations that Seaborn calls "categorical plots". Categorical plots involve a categorical variable, which is a variable that consists of a fixed, typically small number of possible values, or categories. These types of plots are commonly used when we want to make comparisons between different groups. As a reminder, a count plot displays the number of observations in each category.



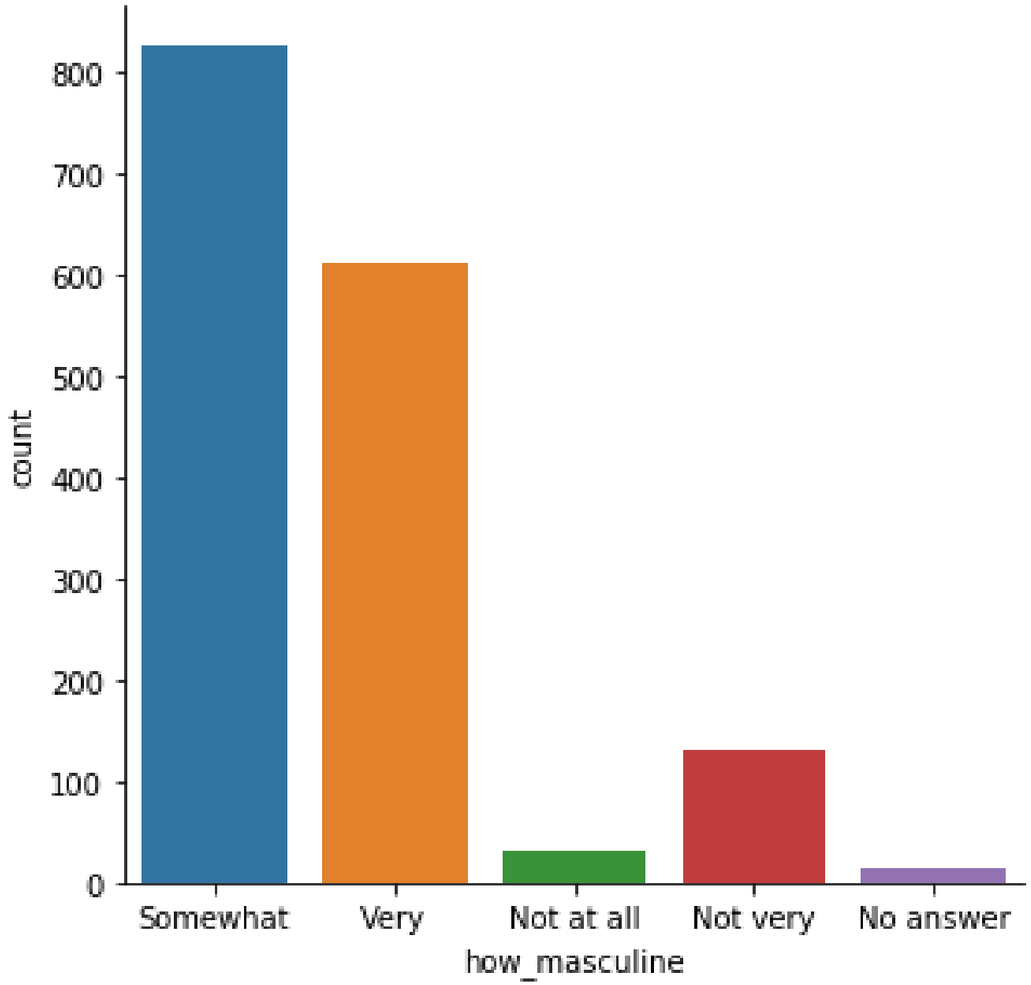
catplot()

- Used to create categorical plots
- Same advantages of `relplot()`
- Easily create subplots with `col=` and `row=`

3. catplot()
Just like we used "relplot()" to create different types of relational plots, in this chapter we'll be using "catplot()" to create different types of categorical plots. "catplot()" offers the same flexibility that "relplot()" does, which means it will be easy to create subplots if we need to using the same "col" and "row" parameters.

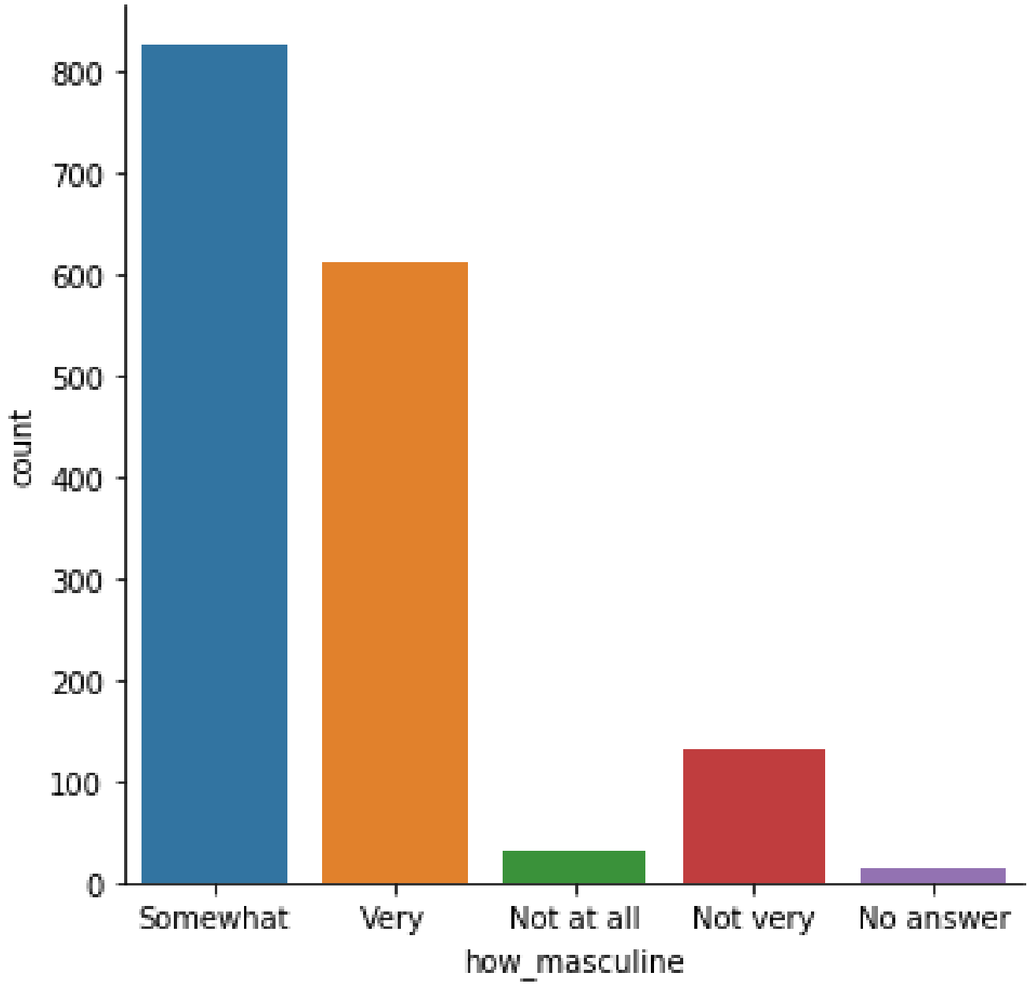
countplot() vs. catplot()

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.countplot(x="how_masculine",
              data=masculinity_data)

plt.show()
```



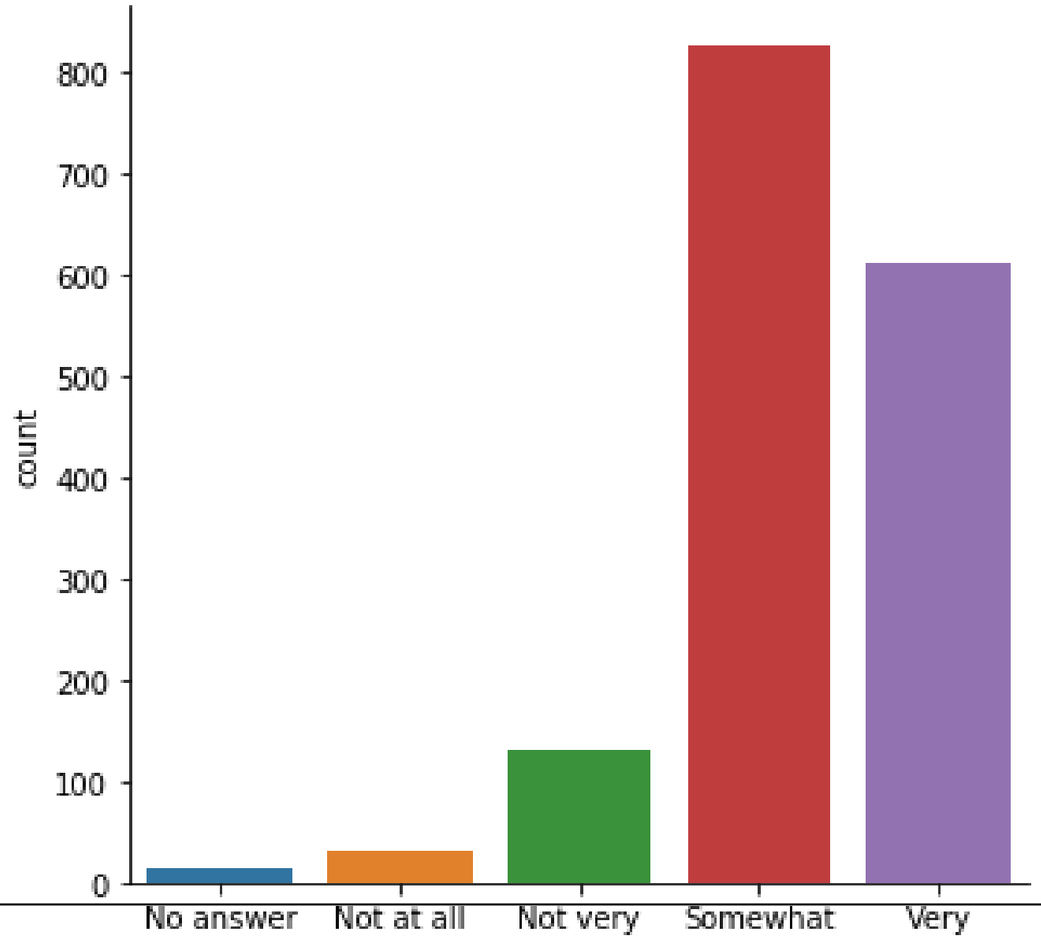
countplot() vs. catplot()

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.catplot(x="how_masculine",
            data=masculinity_data,
            kind="count")

plt.show()
```



Changing the order

```
import matplotlib.pyplot as plt
import seaborn as sns
category_order = ["No answer",
                  "Not at all",
                  "Not very",
                  "Somewhat",
                  "Very"]
sns.catplot(x="how_masculine",
            data=masculinity_data,
            kind="count",
            order=category_order)
plt.show()
```

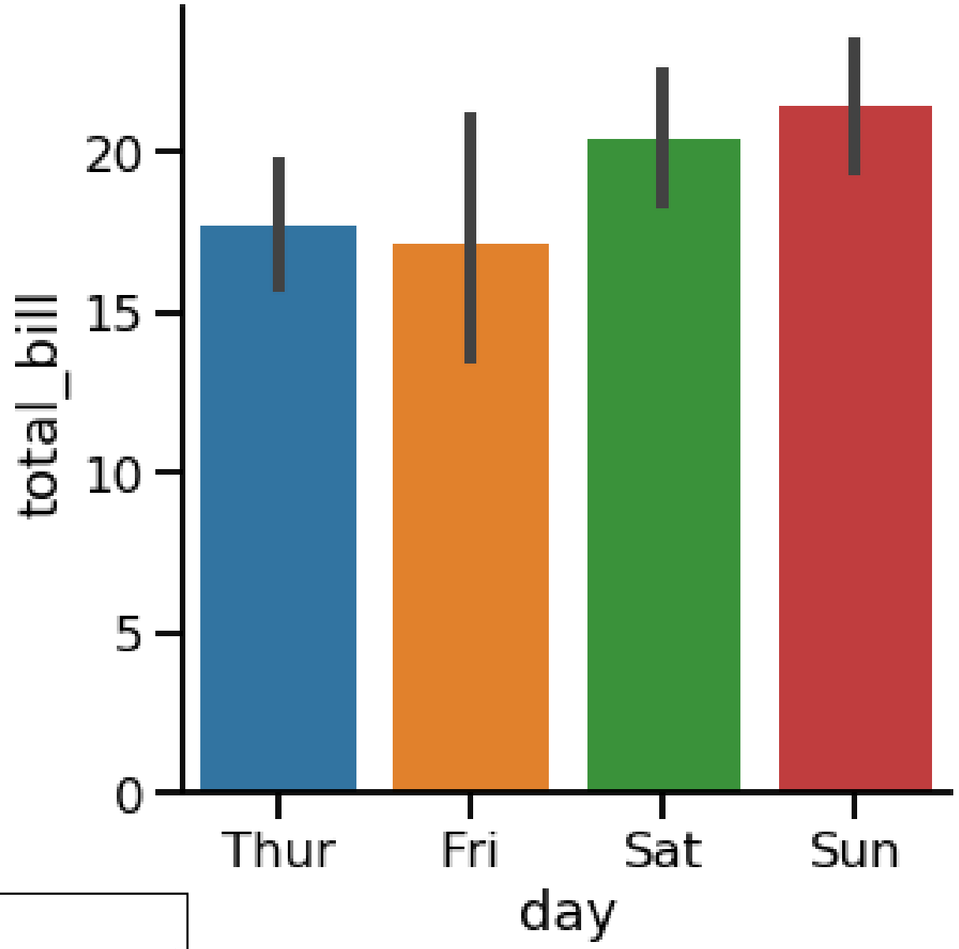


6. Changing the order
Sometimes there is a specific ordering of categories that makes sense for these plots. In this case, it makes more sense for the categories to be in order from not masculine to very masculine. To change the order of the categories, create a list of category values in the order that you want them to appear, and then use the "order" parameter. This works for all types of categorical plots, not just count plots.

Bar plots

Displays mean of quantitative variable per category

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.catplot(x="day",
            y="total_bill",
            data=tips,
            kind="bar")

plt.show()
```



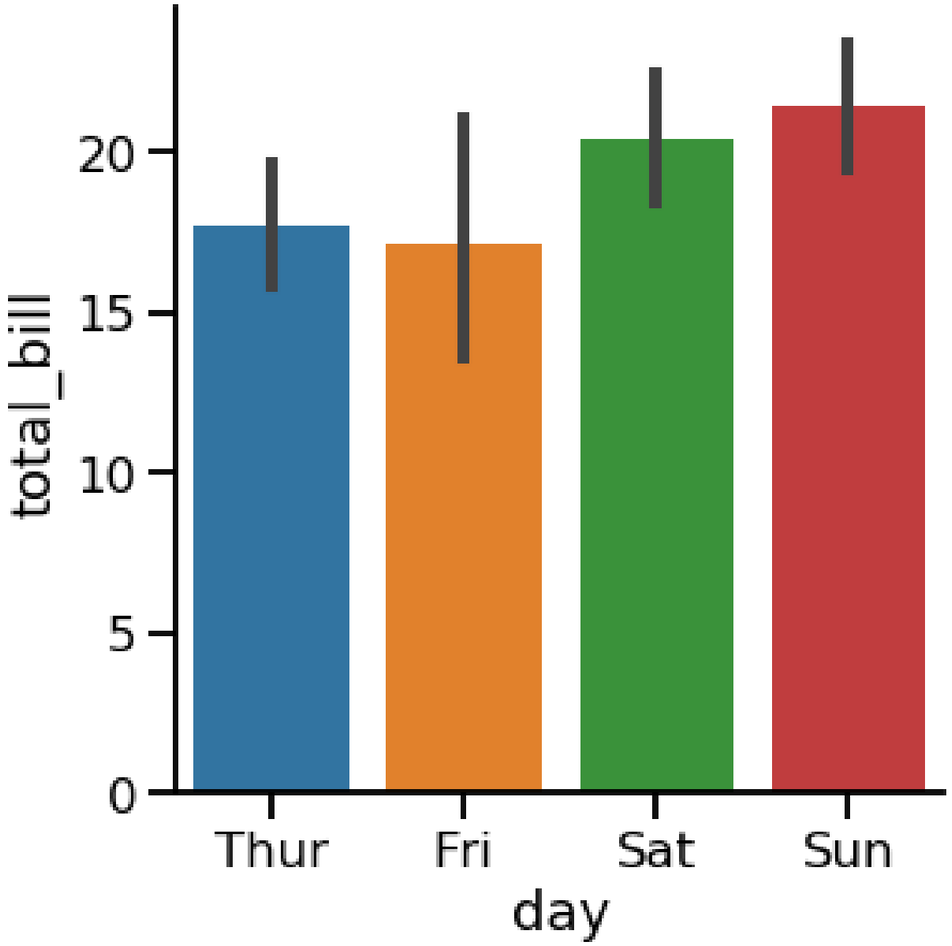
7. Bar plots

Bar plots look similar to count plots, but instead of the count of observations in each category, they show the **mean** of a quantitative variable among observations in each category.

Confidence intervals

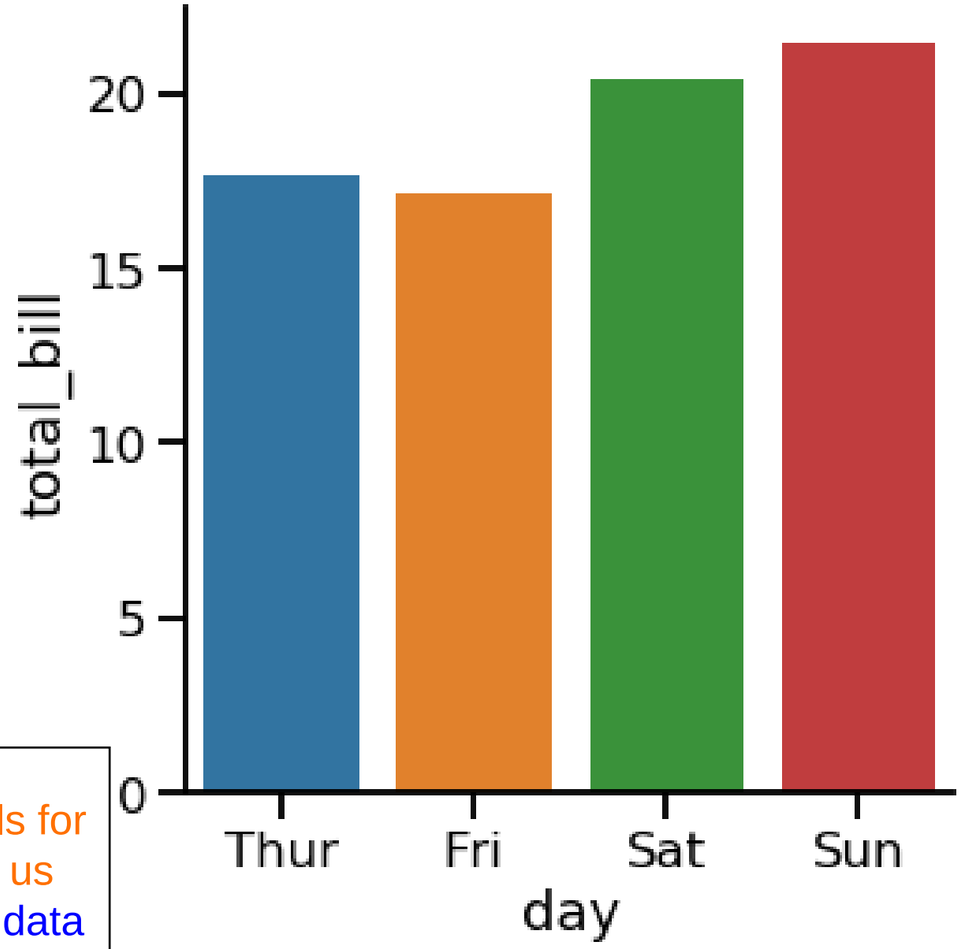
- Lines show 95% confidence intervals for the mean
- Shows uncertainty about our estimate
- Assumes our data is a random sample

7. Bar plots
Bar plots look similar to count plots, but instead of the count of observations in each category, they show the mean of a quantitative variable among observations in each category. This bar plot uses the tips dataset and shows the average bill paid among people who visited the restaurant on each day of the week. From this, we can see that the average bill is slightly higher on the weekends. To create this bar plot, we use "catplot". Specify the categorical variable "day" on the x-axis, the quantitative variable "total bill" on the y-axis, and set the "kind" parameter equal to "bar".



Turning off confidence intervals

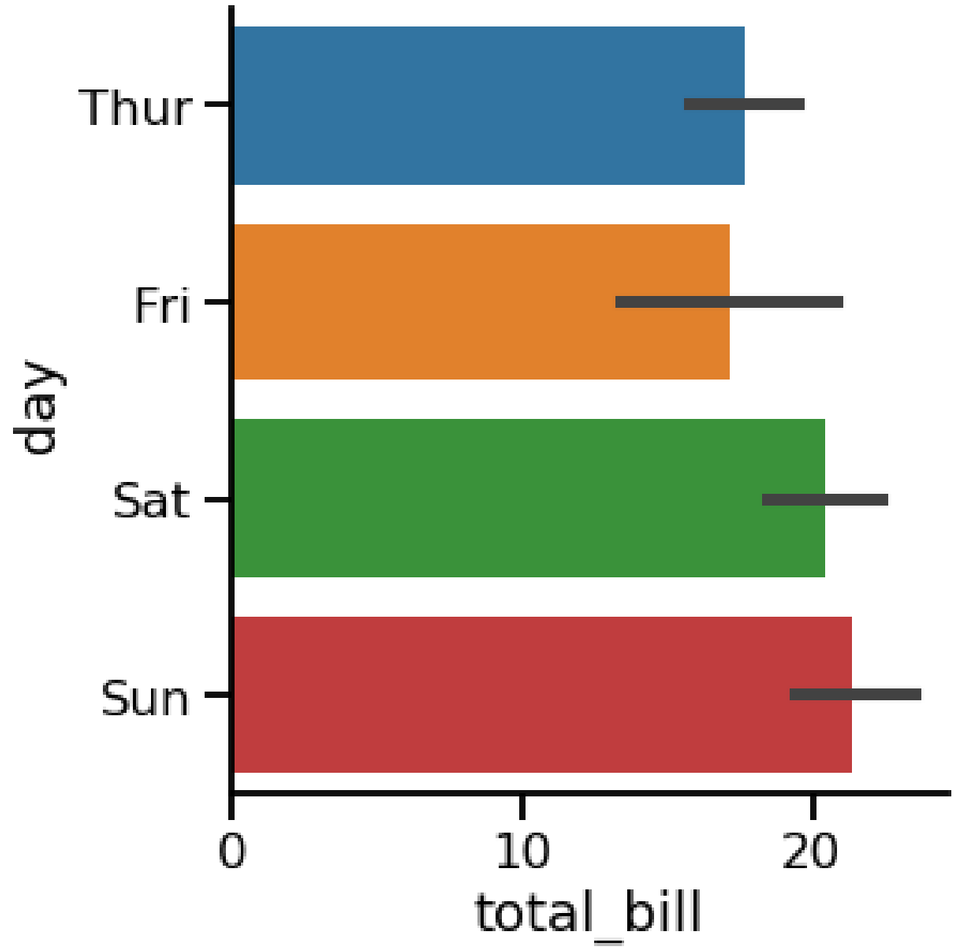
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.catplot(x="day",
            y="total_bill",
            data=tips,
            kind="bar",
            ci=None)
plt.show()
```



8. Confidence intervals
Notice also that Seaborn automatically shows 95% confidence intervals for these means. Just like with line plots, these confidence intervals show us the level of uncertainty we have about these estimates. Assuming our data is a random sample of some population, we can be 95% sure that the true population mean in each group lies within the confidence interval shown.

Changing the orientation

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.catplot(x="total_bill",
            y="day",
            data=tips,
            kind="bar")
plt.show()
```



Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Creating a box plot

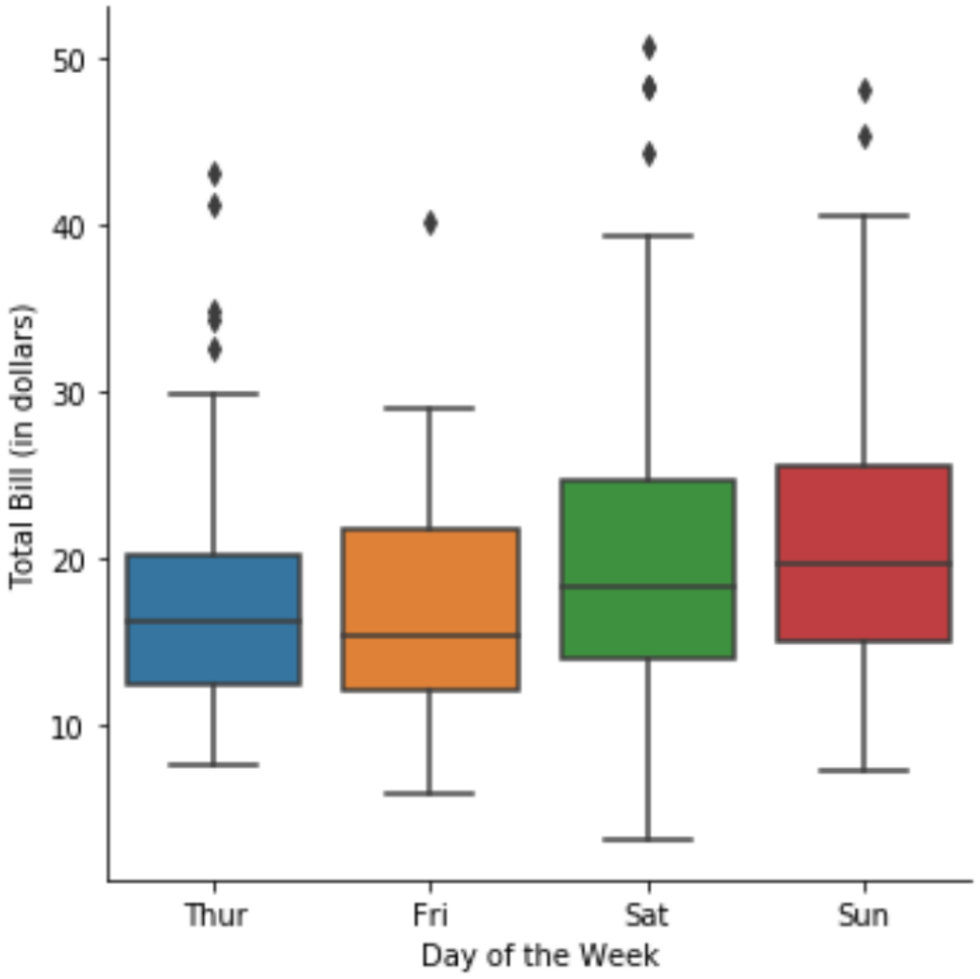
INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Erin Case
Data Scientist

datacamp

What is a box plot?

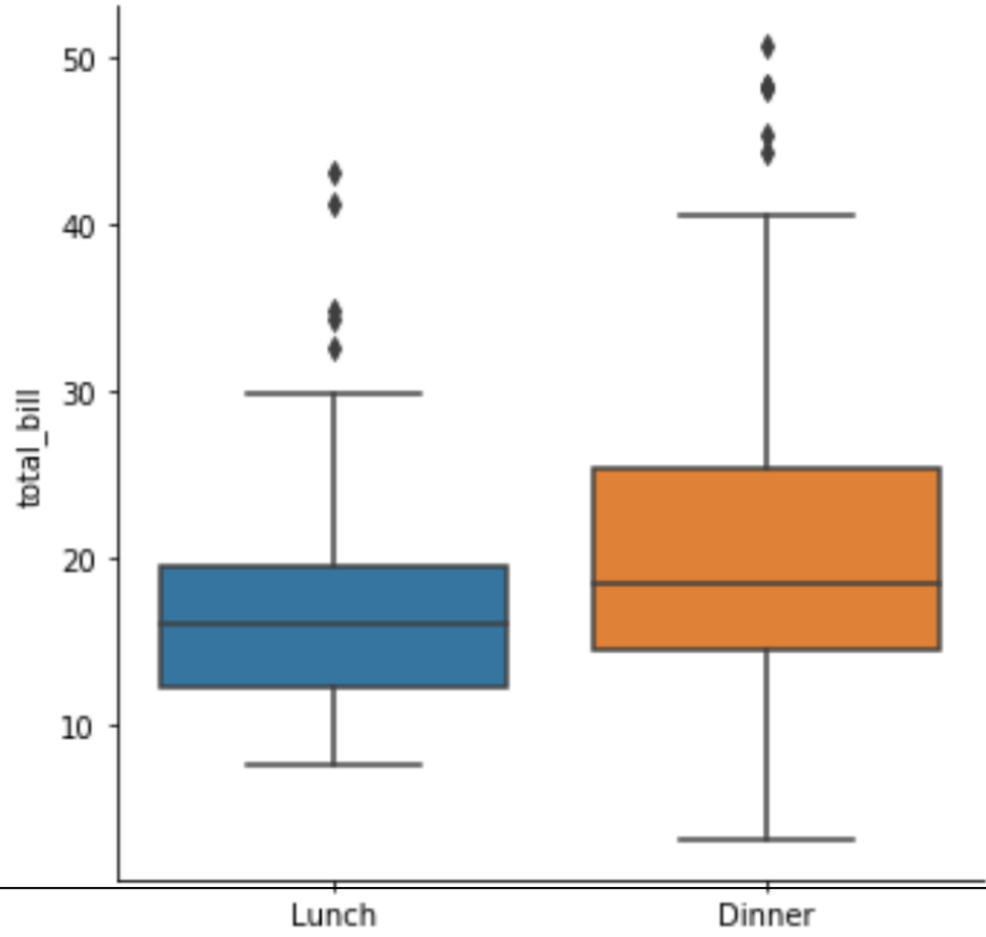
- Shows the distribution of quantitative data
- See median, spread, skewness, and outliers
- Facilitates comparisons between groups



2. What is a box plot?
A box plot shows the distribution of quantitative data. The colored box represents the 25th to 75th percentile, and the line in the middle of the box represents the median. The whiskers give a sense of the spread of the distribution, and the floating points represent outliers. Box plots are commonly used as a way to compare the distribution of a quantitative variable across different groups of a categorical variable. To see this, let's look at this example. The box plot shown here uses the tips dataset and compares the distribution of the total bill paid per table across the different days of the week. From this box plot we can quickly see that the median bill is higher on Saturday and Sunday, but the spread of the distribution is also larger. This comparison would be much harder to do with other types of visualizations.

How to create a box plot

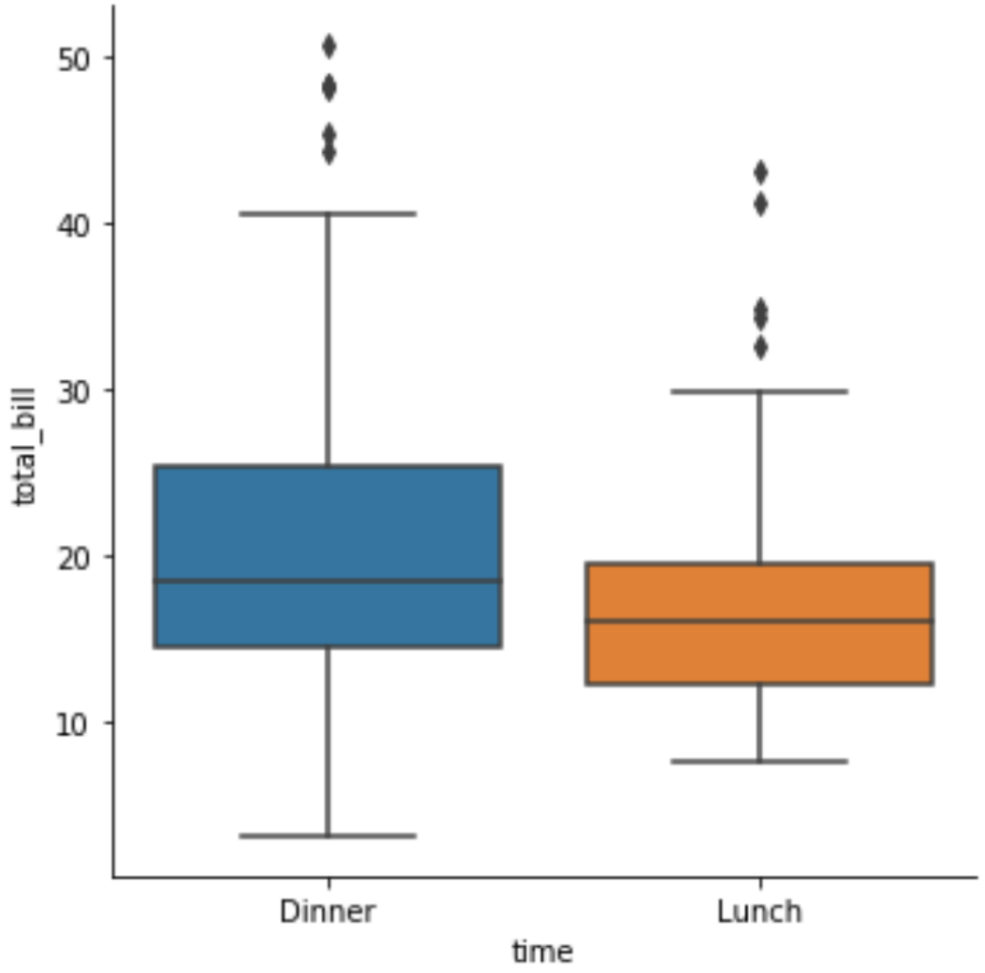
```
import matplotlib.pyplot as plt
import seaborn as sns

g = sns.catplot(x="time",
                y="total_bill",
                data=tips,
                kind="box")
plt.show()
```



3. How to create a box plot
Now let's look at how to create a box plot in Seaborn. While Seaborn does have a "boxplot()" function, we'll be using the "catplot()" function that we introduced in an earlier lesson because it makes it easy to create subplots using the "col" and "row" parameters. We'll put the categorical variable "time" on the x-axis and the quantitative variable "total bill" on the y-axis. Here, we want box plots, so we'll specify kind="box". That's it! We have a nice looking box plot. Next, we'll look at different ways to customize this plot.

Change the order of categories

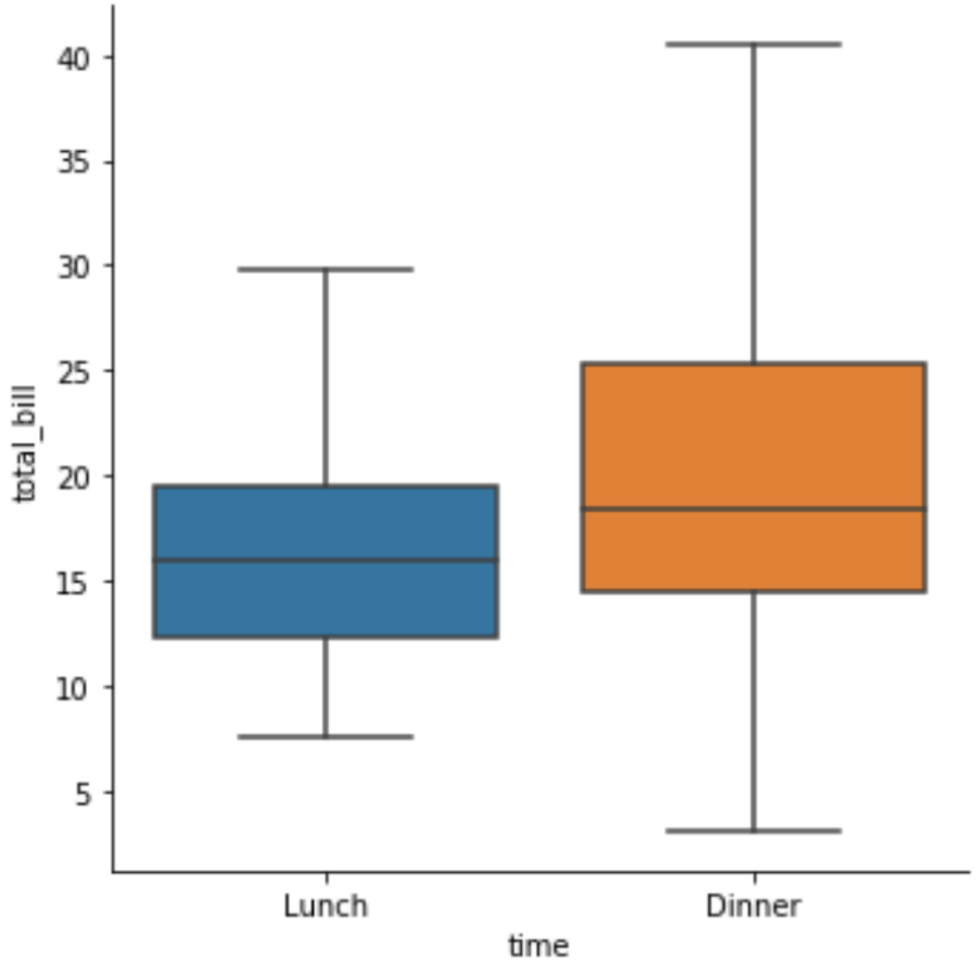
```
import matplotlib.pyplot as plt
import seaborn as sns

g = sns.catplot(x="time",
                y="total_bill",
                data=tips,
                kind="box",
                order=["Dinner",
                       "Lunch"])
plt.show()
```



4. Change the order of categories
As a reminder, "catplot" allows you to change the order of the categories using the "order" parameter. Here, we specified that "dinner" should be shown before "lunch".

Omitting the outliers using `sym`

```
import matplotlib.pyplot as plt
import seaborn as sns

g = sns.catplot(x="time",
                y="total_bill",
                data=tips,
                kind="box",
                sym="")
plt.show()
```



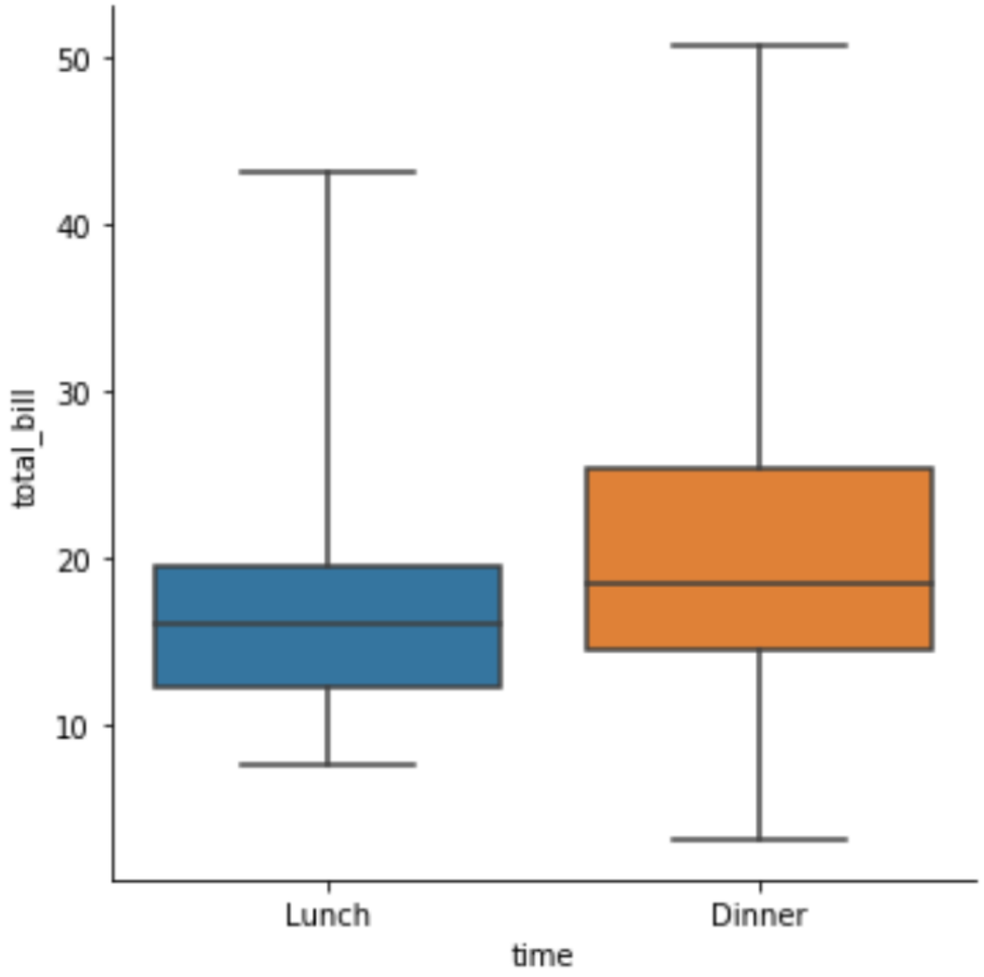
Changing the whiskers using `whis`

- By default, the whiskers extend to 1.5 * the interquartile range
- Make them extend to 2.0 * IQR: `whis=2.0`
- Show the 5th and 95th percentiles: `whis=[5, 95]`
- Show min and max values: `whis=[0, 100]`

6. Changing the whiskers using `whis`
By default, the whiskers extend to 1 point 5 times the interquartile range, or "IQR". The IQR is the 25th to the 75th percentile of a distribution of data. If you want to change the way the whiskers in your box plot are defined, you can do this using the "whis" parameter. There are several options for changing the whiskers. You can change the range of the whiskers from 1 point 5 times the IQR (which is the default) to 2 times the IQR by setting "whis" equal to 2 point 0. Alternatively, you can have the whiskers define specific lower and upper percentiles by passing in a list of the lower and upper values. In this example, passing in "[5, 95]" will result in the lower whisker being drawn at the 5th percentile and the upper whisker being drawn at the 95th percentile. Finally, you may just want to draw the whiskers at the min and max values. You can do this by specifying the lower percentile as 0 and the upper percentile as 100.

Changing the whiskers using `whis`

```
import matplotlib.pyplot as plt
import seaborn as sns

g = sns.catplot(x="time",
                y="total_bill",
                data=tips,
                kind="box",
                whis=[0, 100])
plt.show()
```



Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Point plots

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

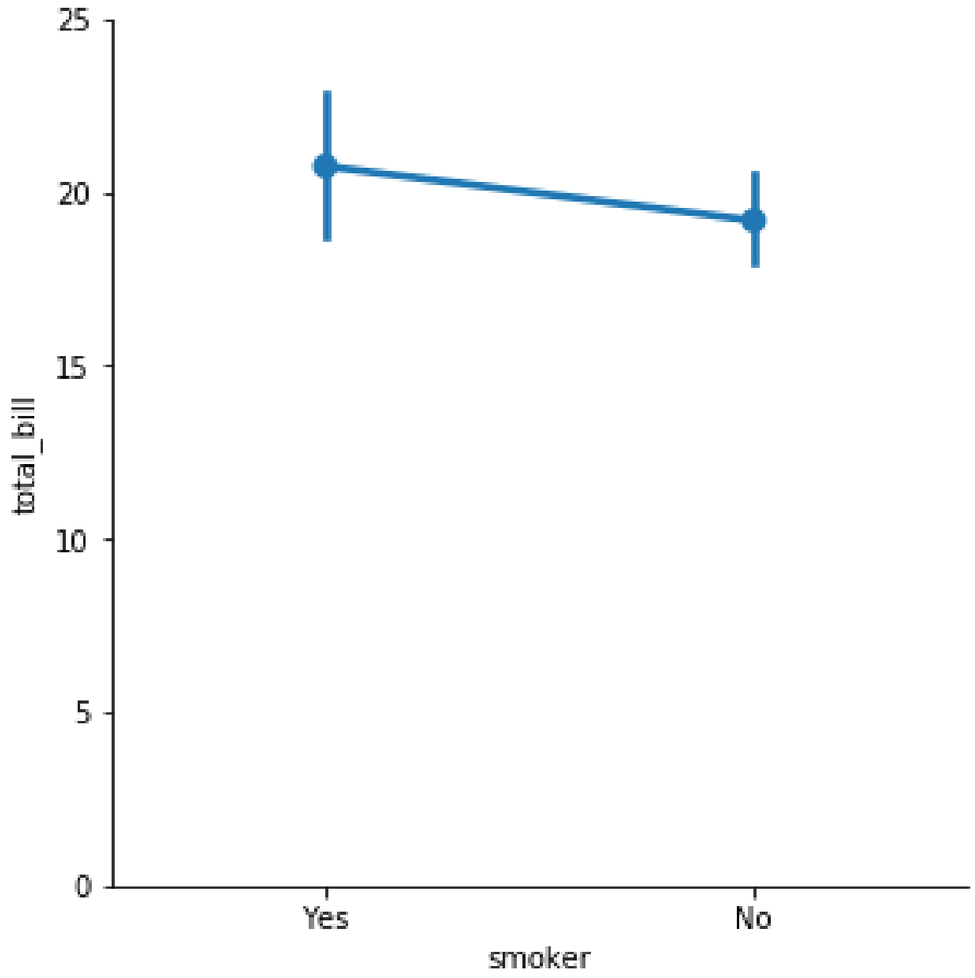
Erin Case
Data Scientist

datacamp

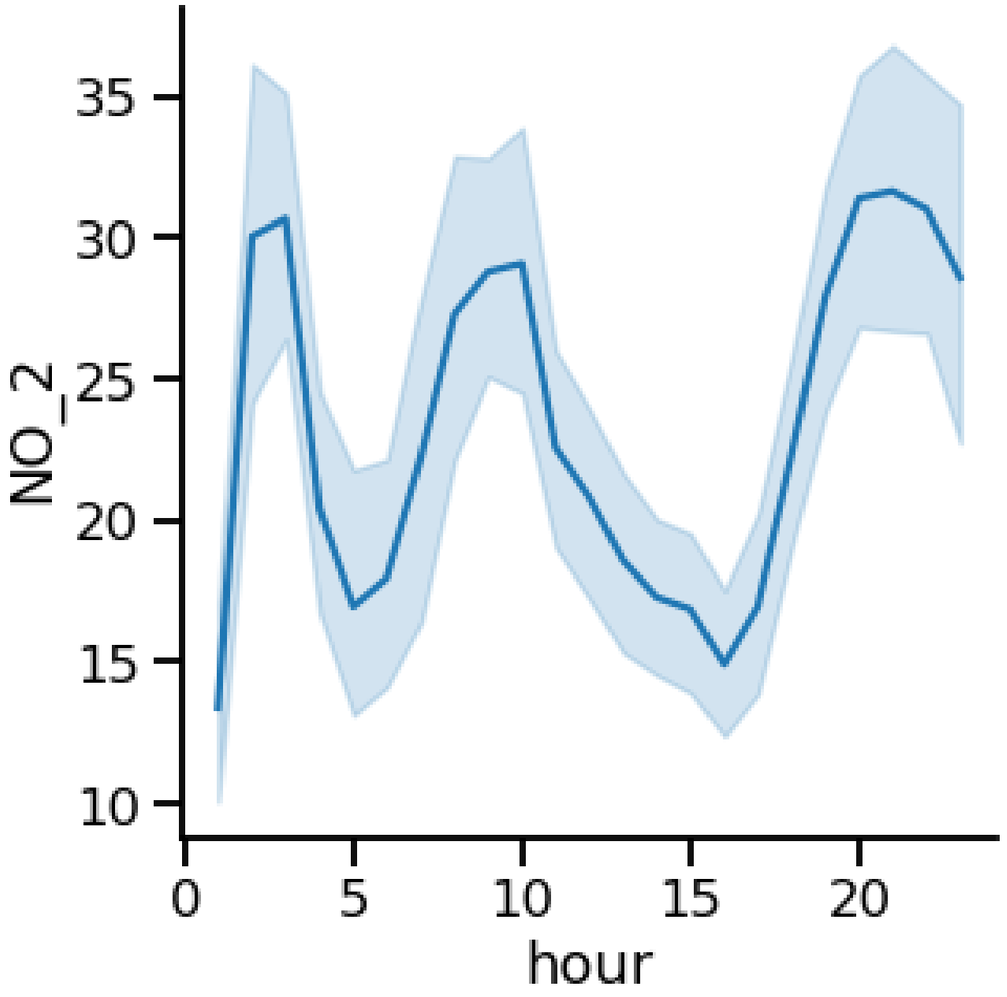
What are point plots?

- Points show mean of quantitative variable
- Vertical lines show 95% confidence intervals

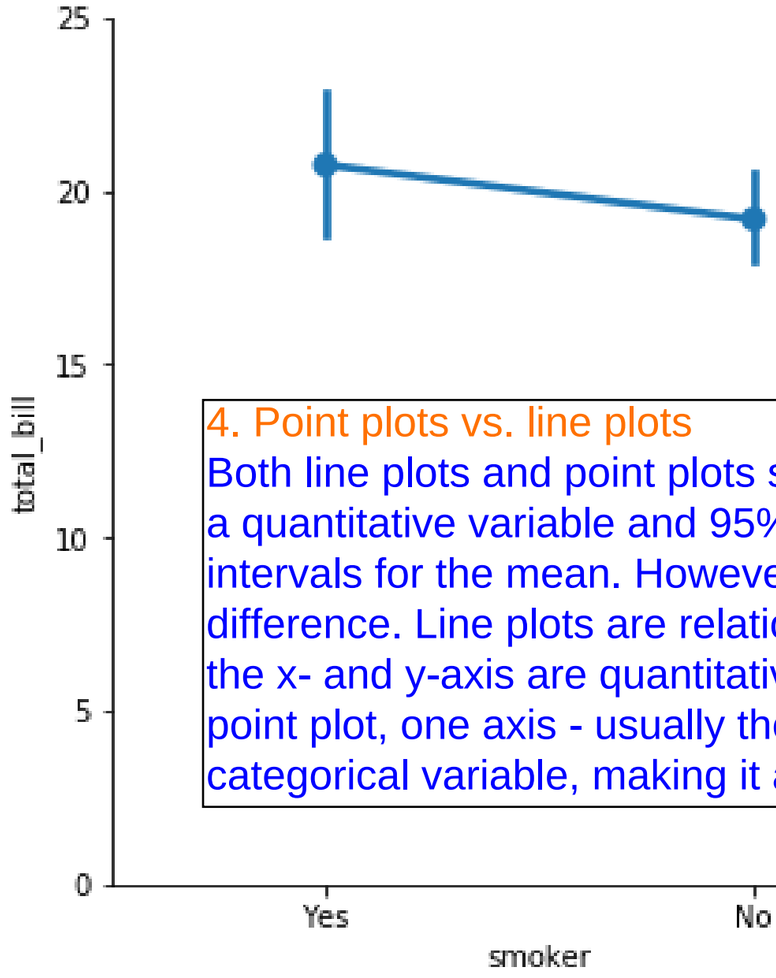
2. What are point plots?
Point plots show the mean of a quantitative variable for the observations in each category, plotted as a single point. This point plot uses the tips dataset and shows the average bill among smokers versus non-smokers. The vertical bars extending above and below the mean represent the 95% confidence intervals for that mean. Just like the confidence intervals we saw in line plots and bar plots, these confidence intervals show us the level of uncertainty we have about these mean estimates. Assuming our data is a random sample of some population, we can be 95% sure that the true population mean in each group lies within the confidence interval shown.



Line plot: average level of nitrogen dioxide over time



Point plot: average restaurant bill, smokers vs. non-smokers



4. Point plots vs. line plots

Both line plots and point plots show the mean of a quantitative variable and 95% confidence intervals for the mean. However, there is a key difference. Line plots are relational plots, so both the x- and y-axis are quantitative variables. In a point plot, one axis - usually the x-axis - is a categorical variable, making it a categorical plot.

Point plots vs. line plots

Both show:

- Mean of quantitative variable
- 95% confidence intervals for the mean

Differences:

- Line plot has **quantitative** variable (usually time) on x-axis
- Point plot has **categorical** variable on x-axis

5. Point plots vs. bar plots
You may also be thinking: point plots seem to show the same information as bar plots. For each category, both show the mean of a quantitative variable and the confidence intervals for those means. When should we use one over the other? Let's look at an example using data from the masculinity survey that we've seen in prior lessons.

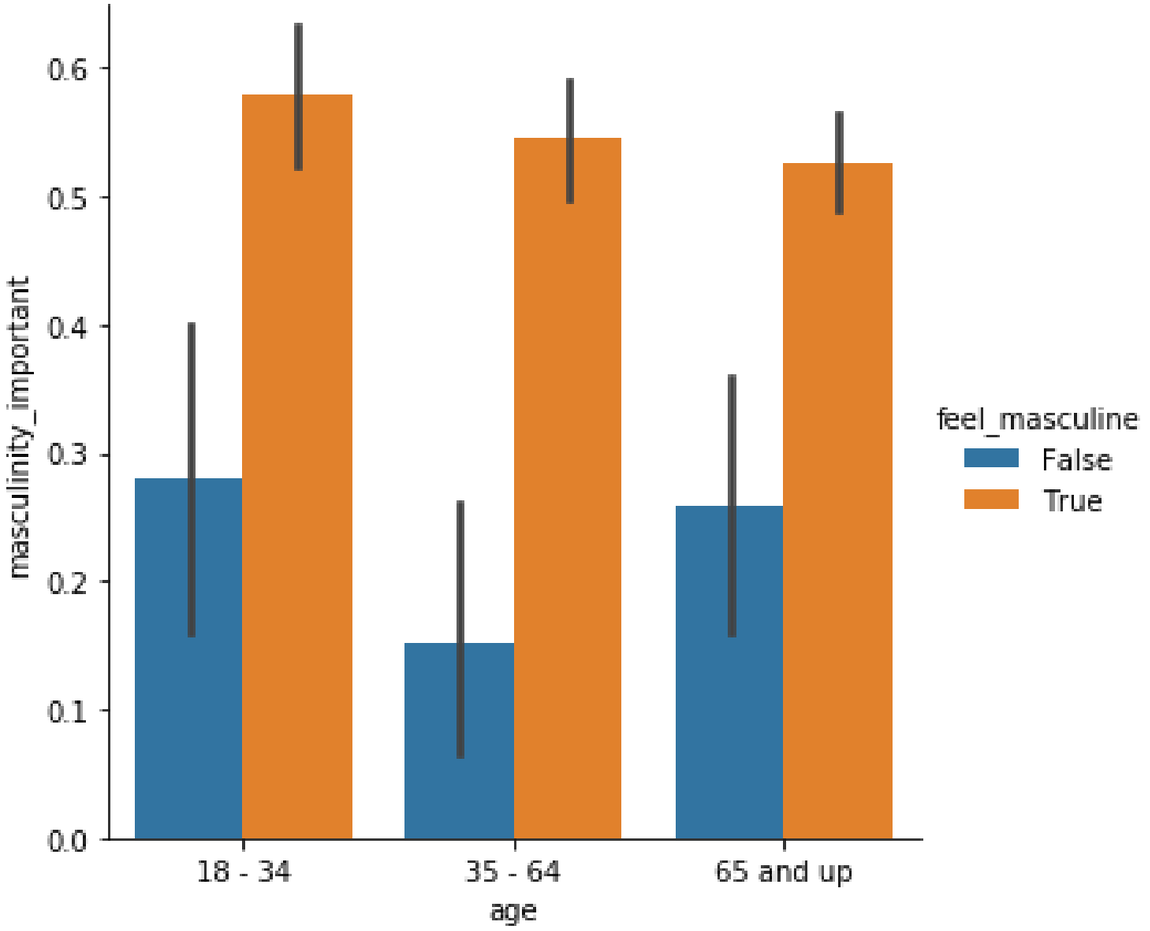
Point plots vs. bar plots

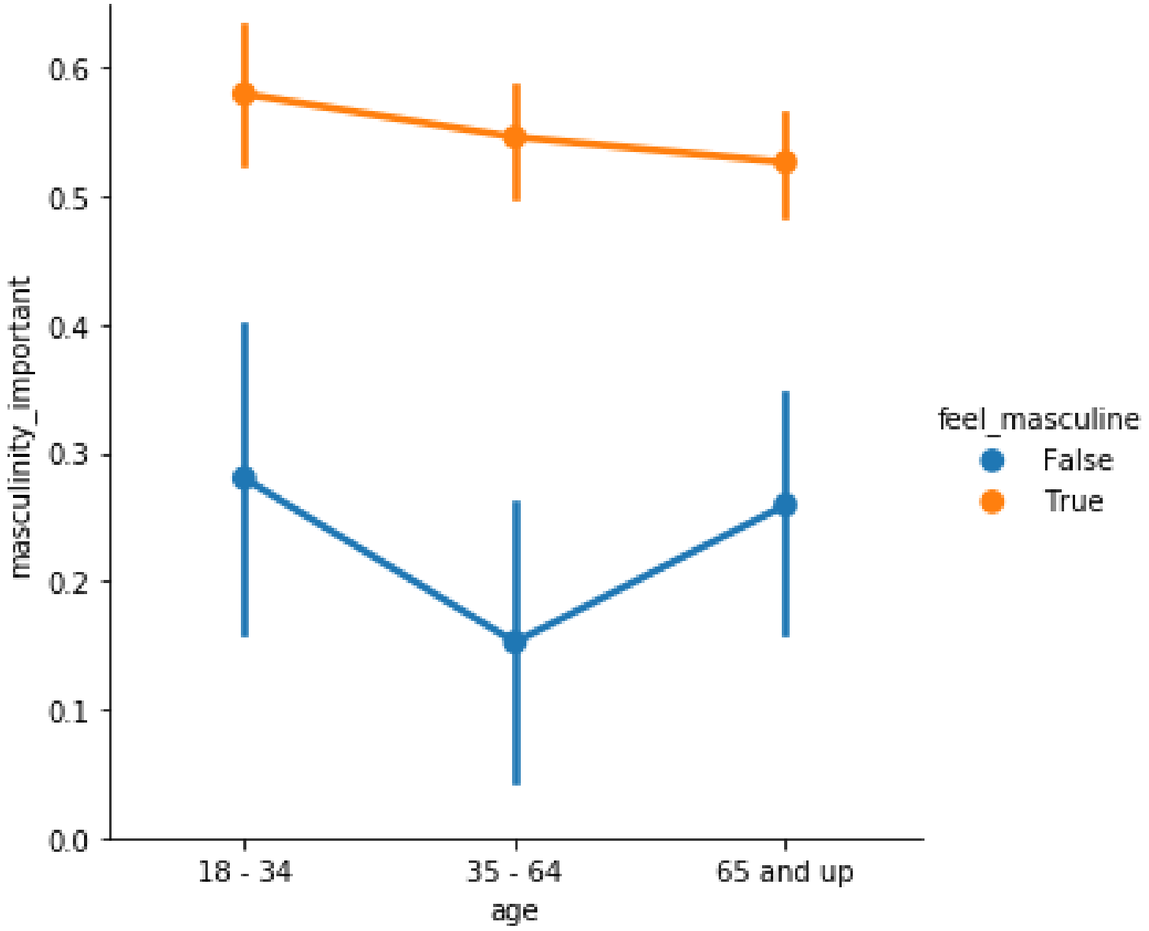
Both show:

- Mean of quantitative variable
- 95% confidence intervals for the mean

6. Point plots vs. bar plots
This is a bar plot of the percent of men per age group surveyed who report thinking that it's important that others see them as masculine, with subgroups based on whether they report feeling masculine or not. This is the same information, represented as a point plot. In the point plot, it's easier to compare the heights of the subgroup points when they're stacked above each other. In the point plot, it's also easier to look at the differences in slope between the categories than it is to compare the heights of the bars between them.

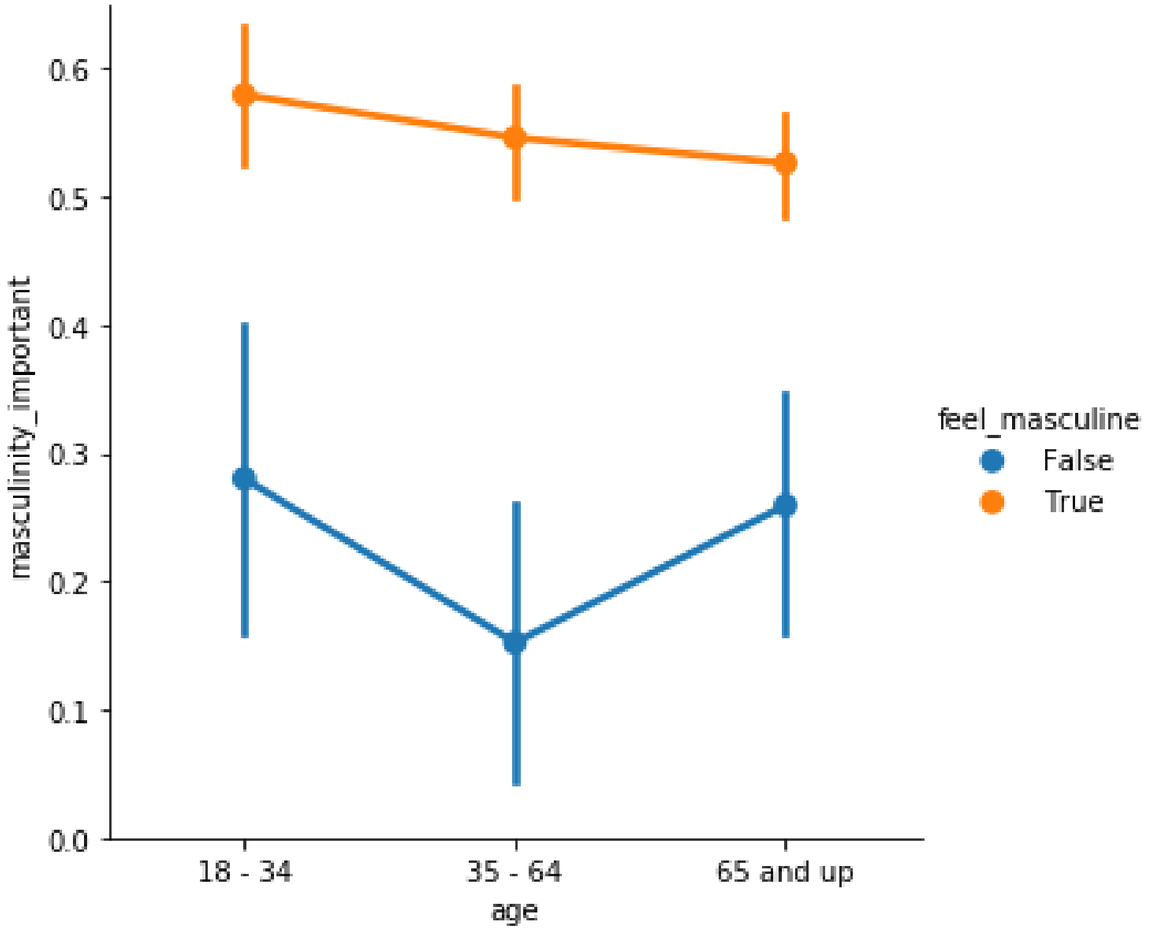
Point plots vs. bar plots





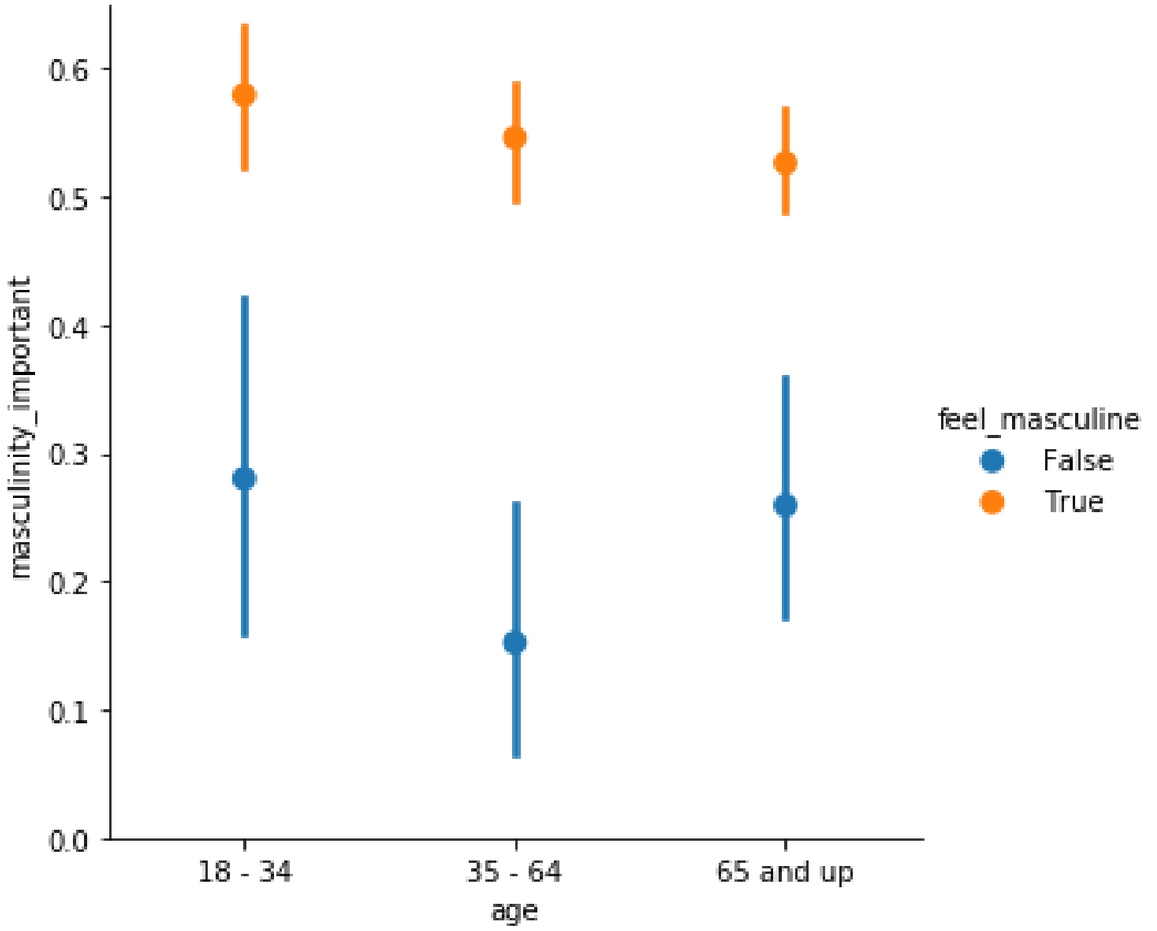
Creating a point plot

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.catplot(x="age",
            y="masculinity_important",
            data=masculinity_data,
            hue="feel_masculine",
            kind="point")

plt.show()
```



Disconnecting the points

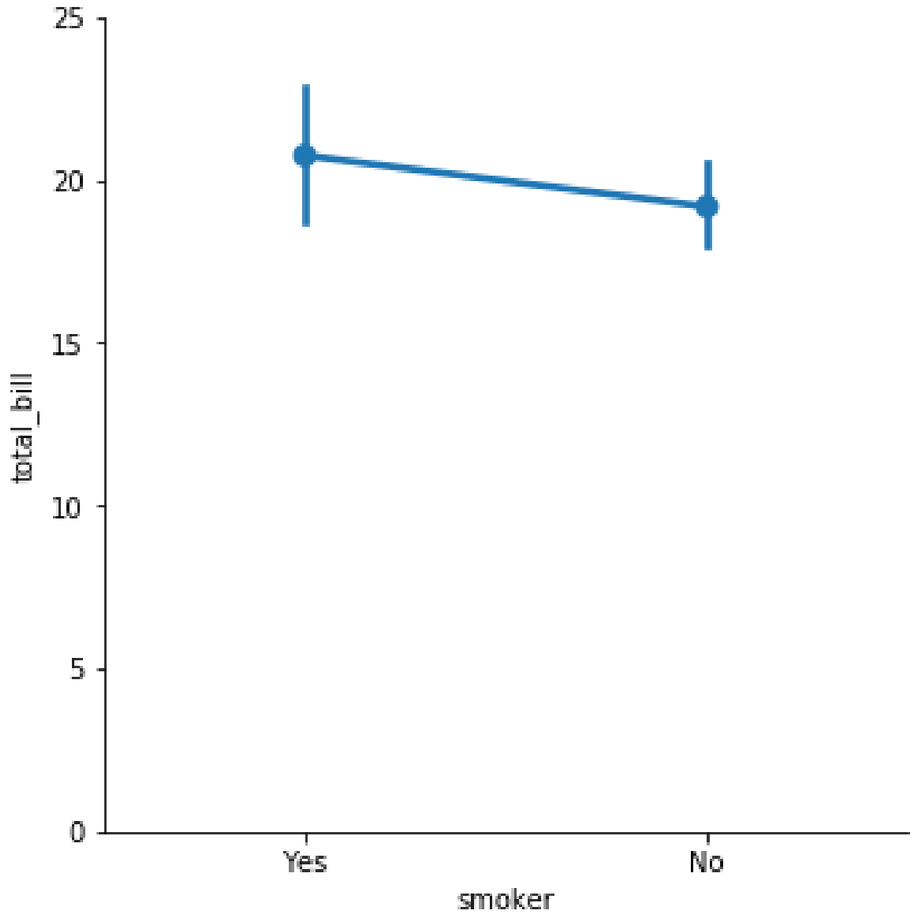
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.catplot(x="age",
            y="masculinity_important",
            data=masculinity_data,
            hue="feel_masculine",
            kind="point",
            join=False)
plt.show()
```



Displaying the median

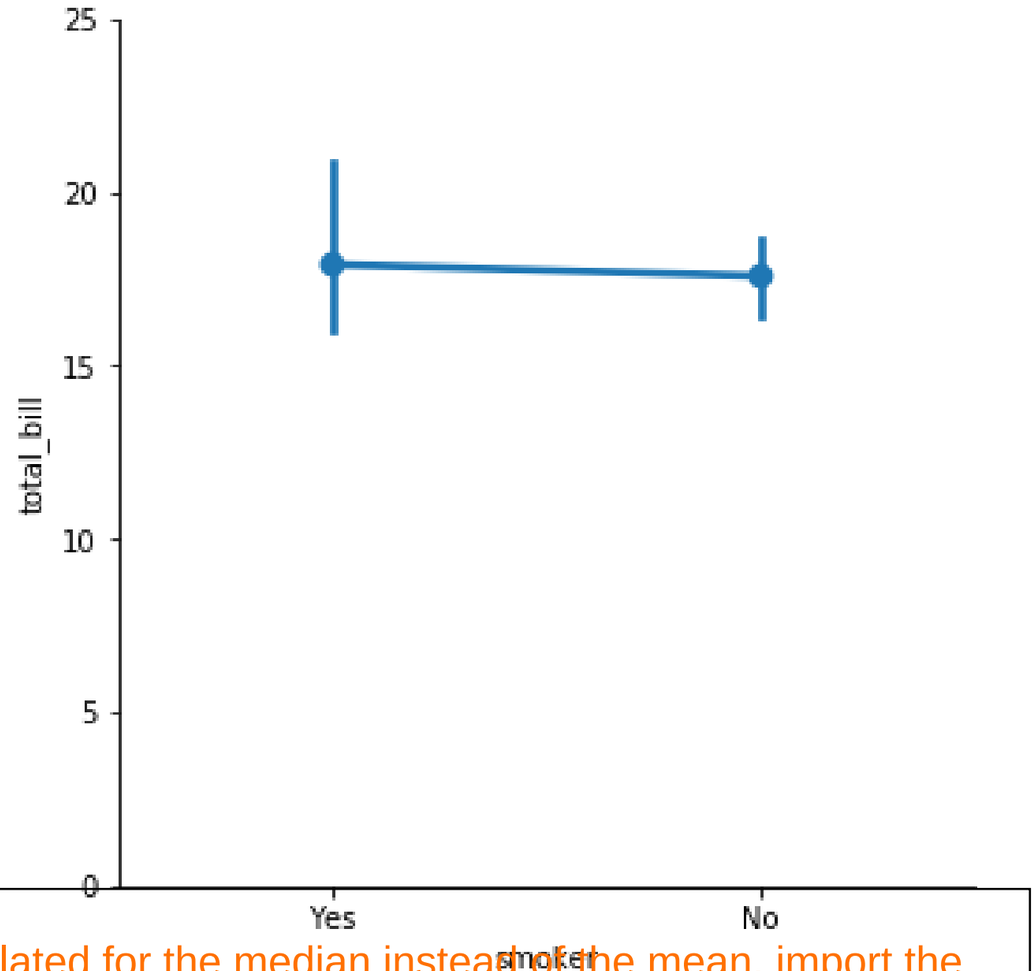
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.catplot(x="smoker",
            y="total_bill",
            data=tips,
            kind="point")

plt.show()
```

9. Displaying the median
Let's return to the point plot using the tips dataset and go over a few more ways to customize your point plots. Here is the point plot of average bill comparing smokers to non-smokers.



Displaying the median

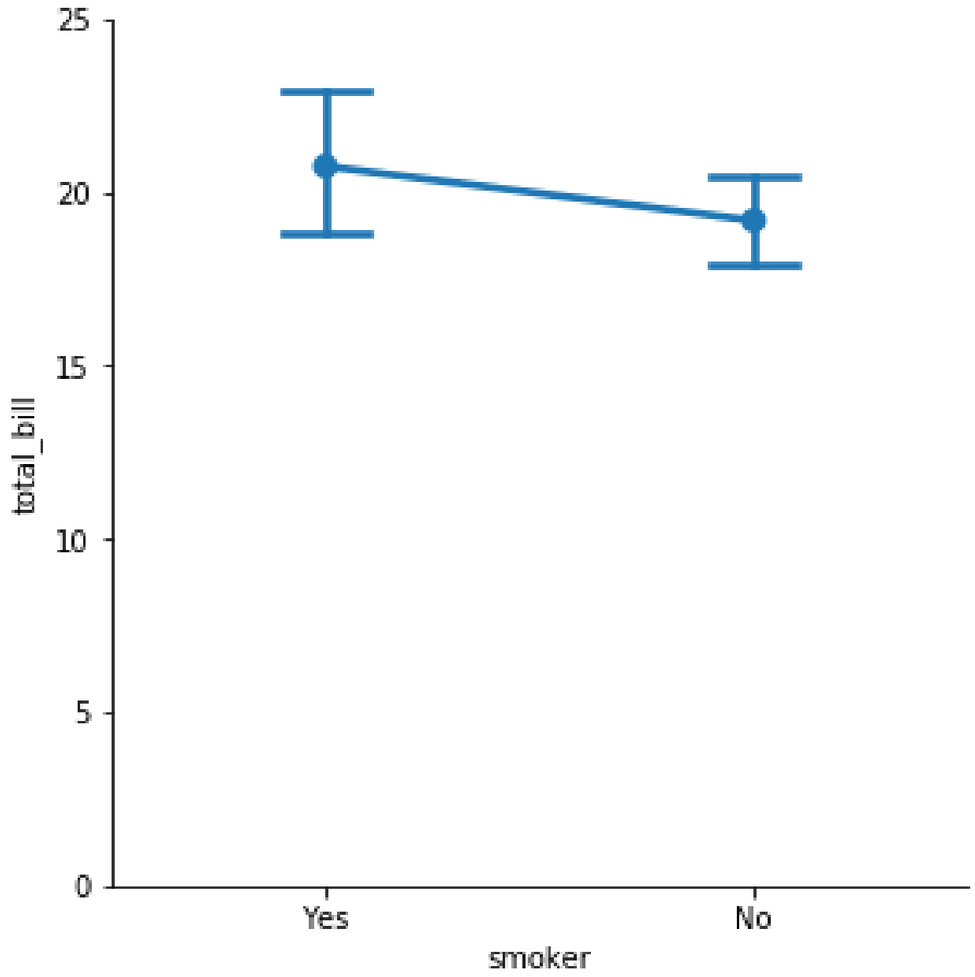
```
import matplotlib.pyplot as plt
import seaborn as sns
from numpy import median

sns.catplot(x="smoker",
            y="total_bill",
            data=tips,
            kind="point",
            estimator=median)

plt.show()
```



10. Displaying the median
To have the points and confidence intervals be calculated for the median instead of the mean, import the median function from the numpy library and set "estimator" equal to the numpy median function. Why might you want to use the median instead of the mean? The median is more robust to outliers, so if your dataset has a lot of outliers, the median may be a better statistic to use.

Customizing the confidence intervals

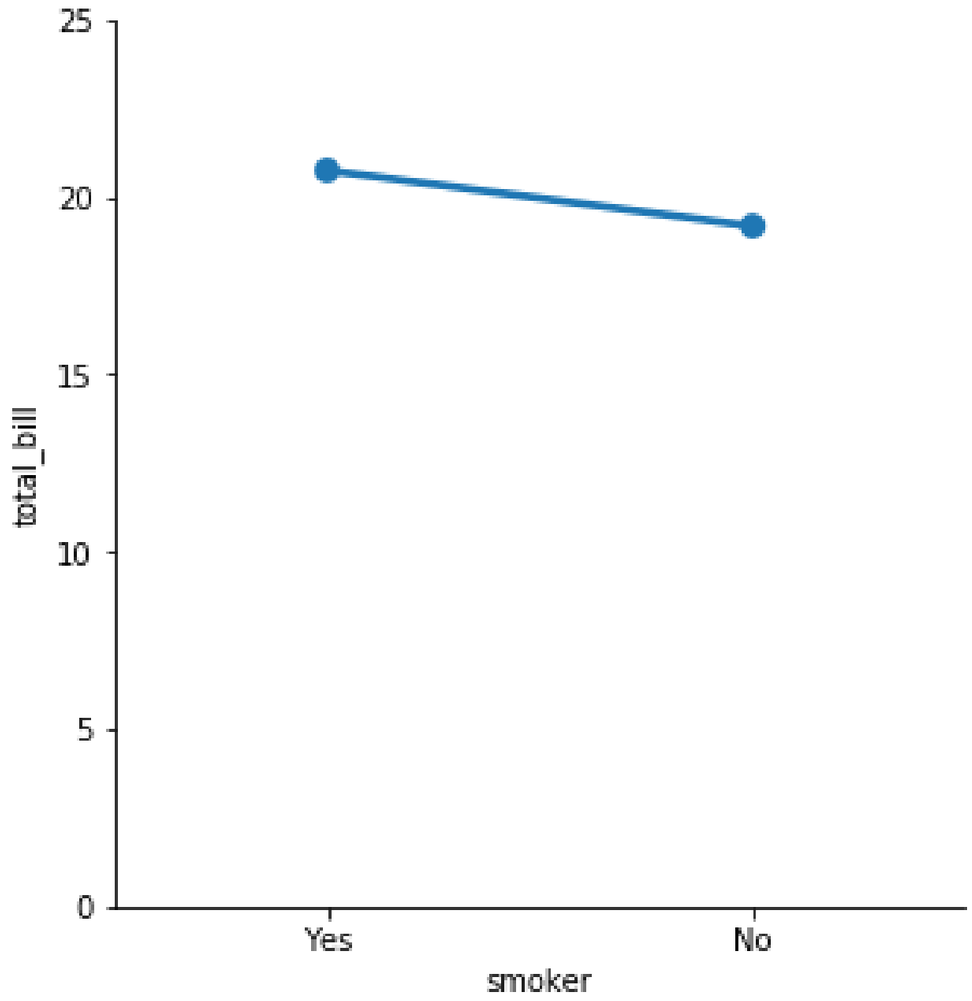
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.catplot(x="smoker",
            y="total_bill",
            data=tips,
            kind="point",
            capsize=0.2)
plt.show()
```



11. Customizing the confidence intervals
You can also customize the way that the confidence intervals are displayed. To add “caps” to the end of the confidence intervals, set the “capsize” parameter equal to the desired width of the caps. In this case, we chose a width of 0.2.

Turning off confidence intervals

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.catplot(x="smoker",
            y="total_bill",
            data=tips,
            kind="point",
            ci=None)
plt.show()
```



Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN