

Changing plot style and color

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



Erin Case
Data Scientist

Why customize?

Reasons to change style:

- Personal preference
- Improve readability
- Guide interpretation

Changing the figure style

- Figure "style" includes background and axes
- Preset options: "white", "dark", "whitegrid", "darkgrid", "ticks"
- `sns.set_style()`

3. Changing the figure style

Seaborn has five preset figure styles which change the background and axes of the plot. You can refer to them by name: "white", "dark", "whitegrid", "darkgrid", and "ticks". To set one of these as the global style for all of your plots, use the "set style" function.

Default figure style ("white")

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

4. Default figure style ("white")

This is a plot we've seen before, showing the percentage of men reporting that masculinity was important to them, stratified by their age and whether or not they feel masculine. The default style is called "white" and provides clean axes with a solid white background. If we only care about the comparisons between groups or the general trend across age groups instead of the specific values, this is a good choice.

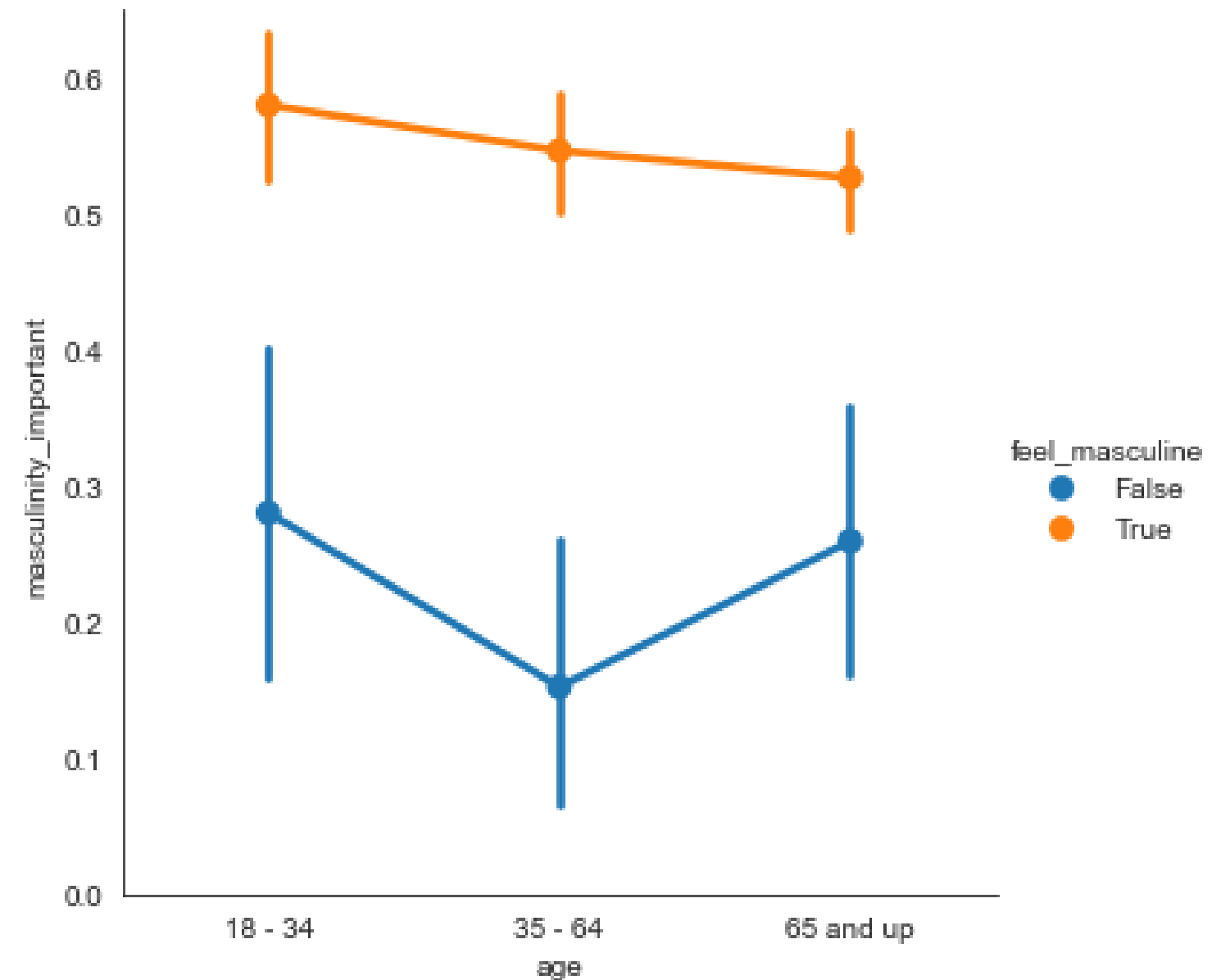


Figure style: "whitegrid"

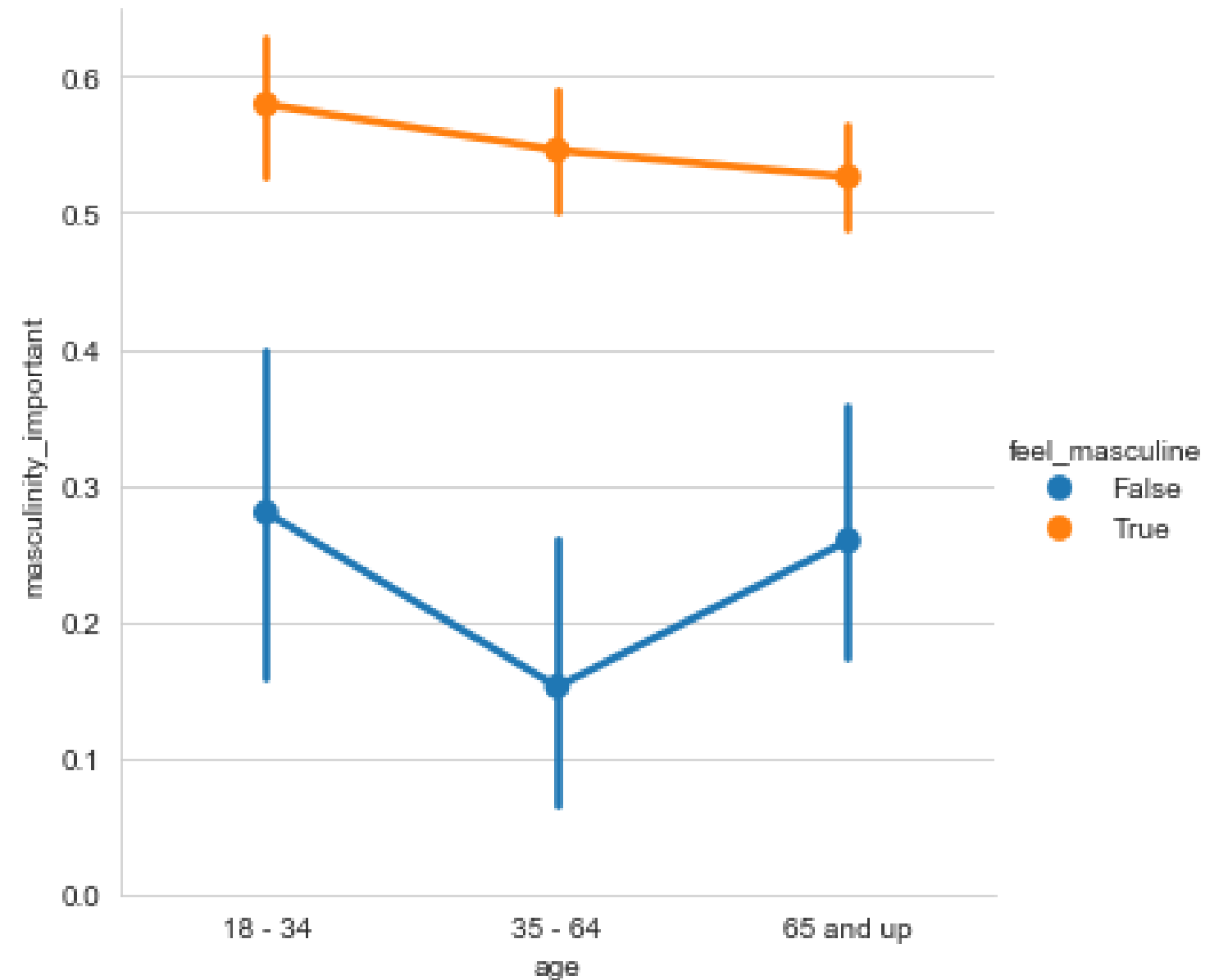
```
sns.set_style("whitegrid")

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

plt.show()
```

5. Figure style: "whitegrid"

Changing the style to "whitegrid" will add a gray grid in the background. This is useful if you want your audience to be able to determine the specific values of the plotted points instead of making higher level observations.



Other styles

```
sns.set_style("ticks")

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

plt.show()
```

6. Other styles

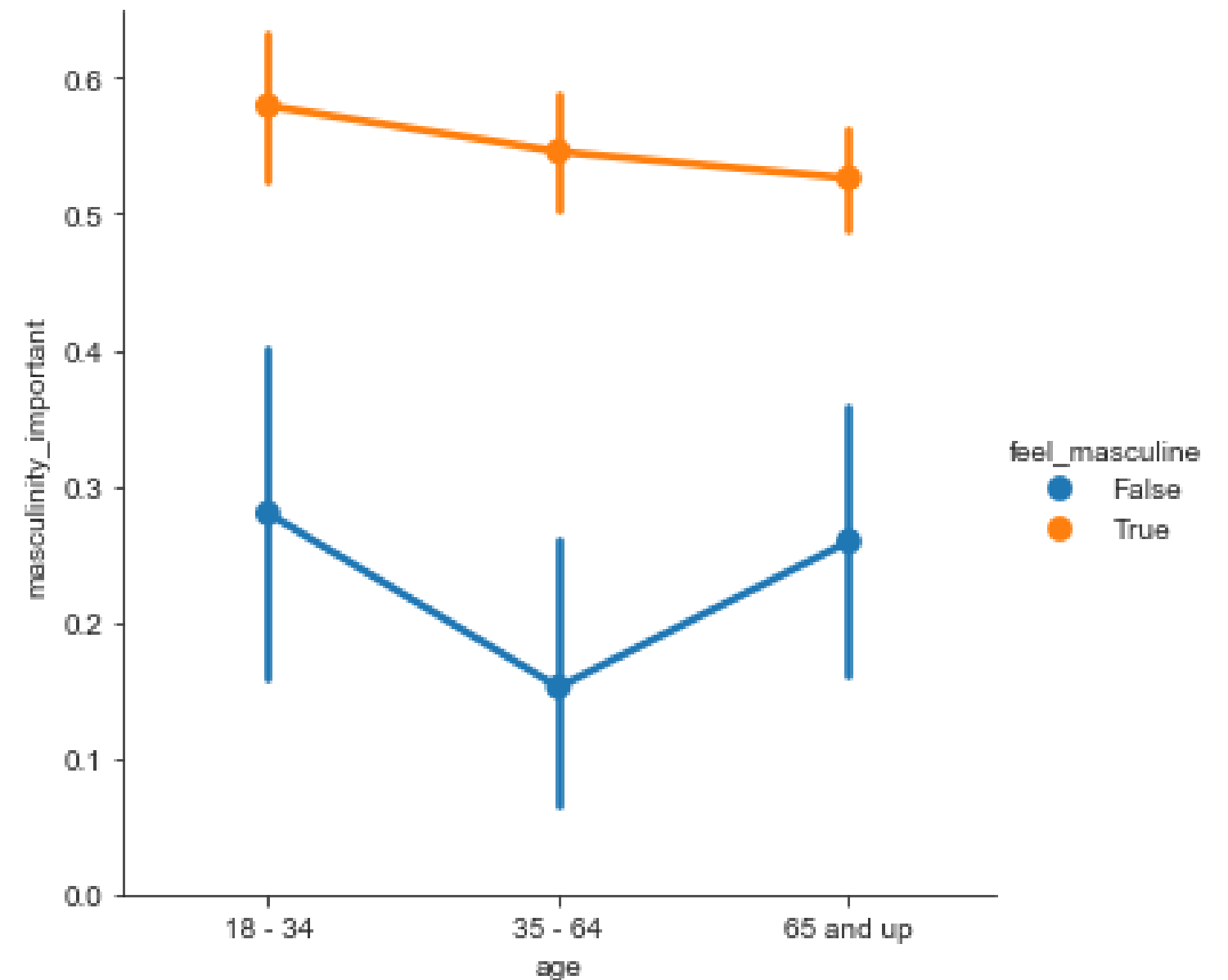
The other styles are variants on these. "ticks" is similar to "white", but adds small tick marks to the x- and y-axes.

7. Other styles

"dark" provides a gray background,

8. Other styles

and "darkgrid" provides a gray background with a white grid.

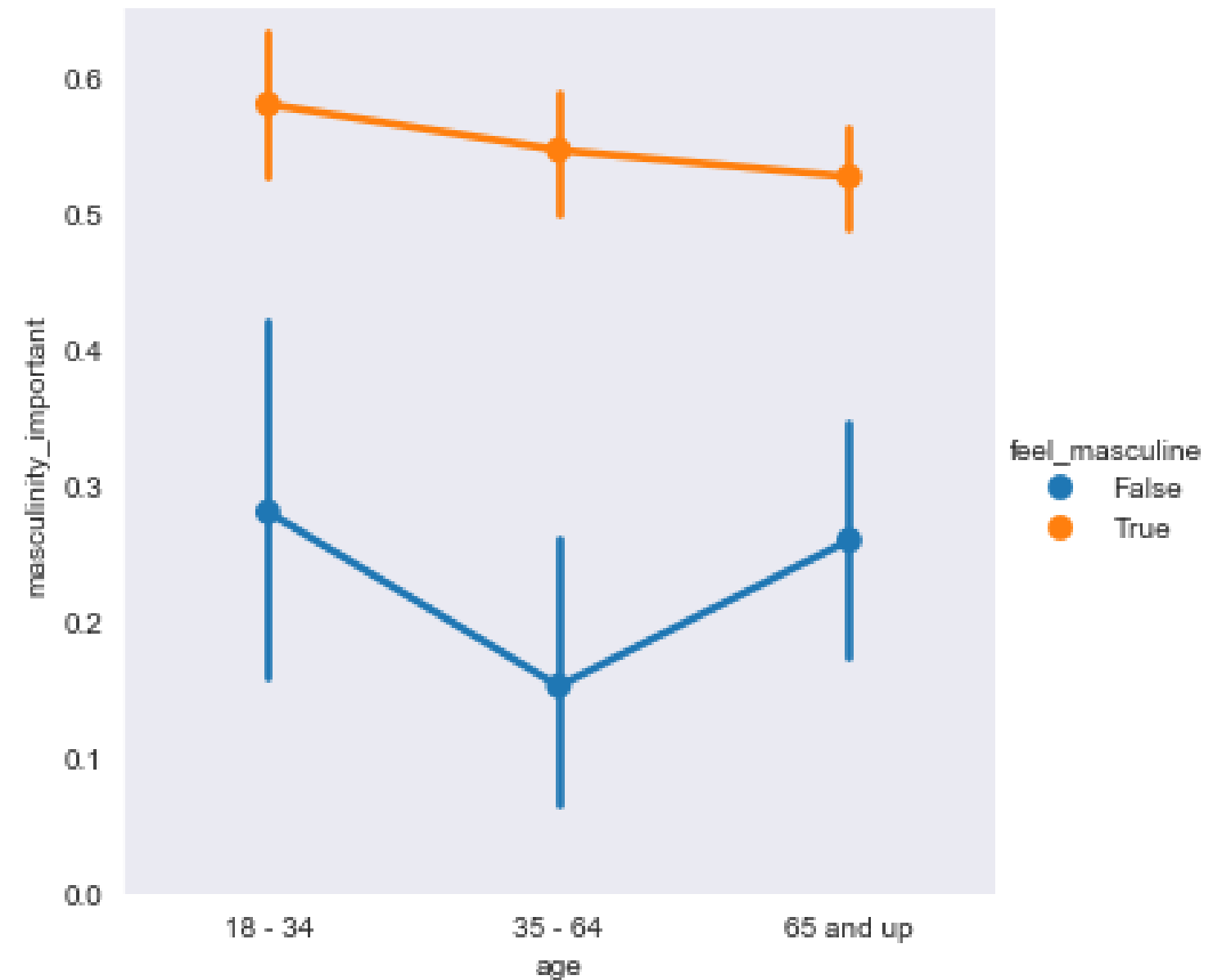


Other styles

```
sns.set_style("dark")

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

plt.show()
```

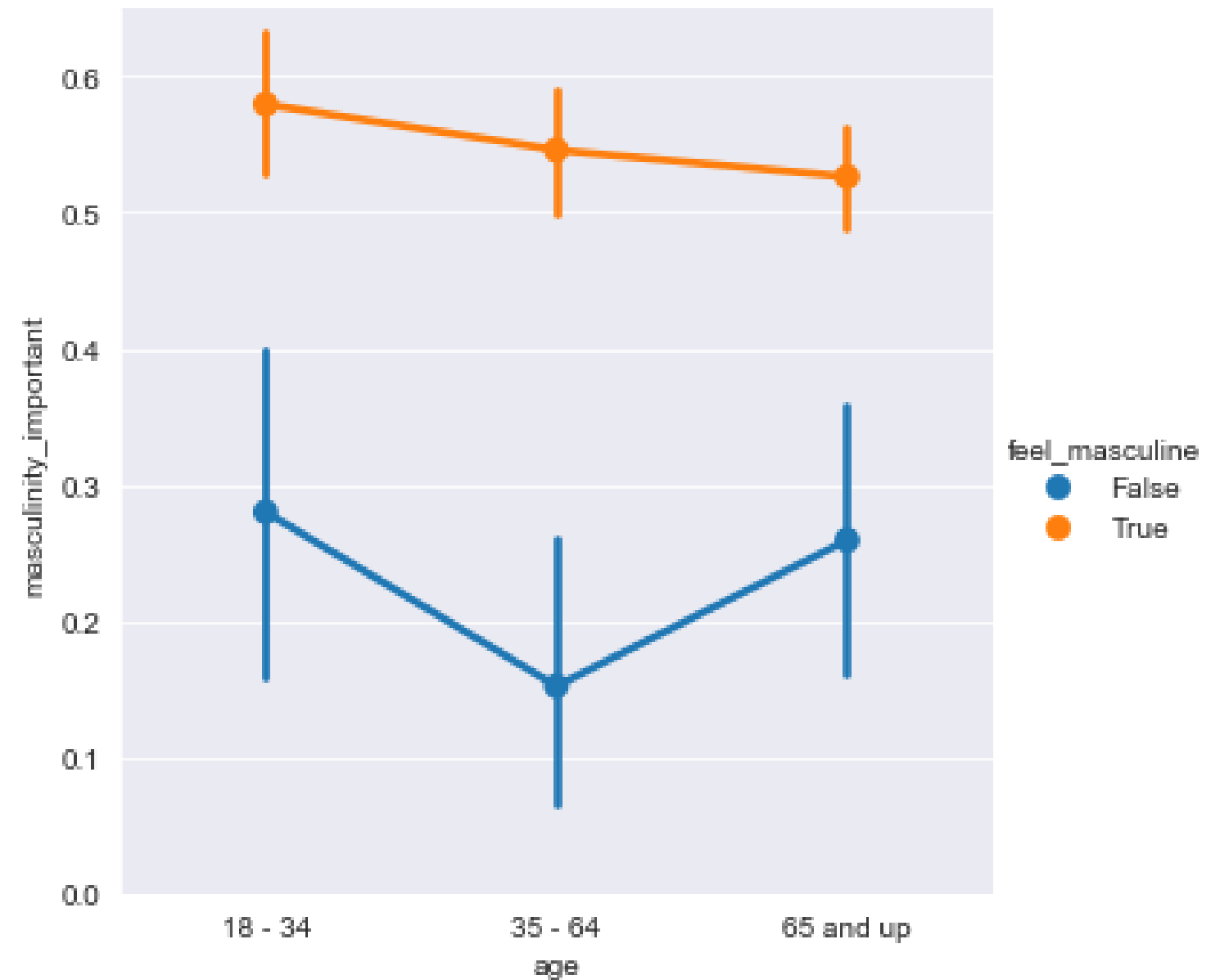


Other styles

```
sns.set_style("darkgrid")

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

plt.show()
```







Changing the palette

- Figure "palette" changes the color of the main elements of the plot
- `sns.set_palette()`
- Use preset palettes or create a custom palette

9. Changing the palette

You can change the color of the main elements of the plot with Seaborn's "set palette" function. Seaborn has many preset color palettes that you can refer to by name, or you can create your own custom palette. Let's see an example.

Diverging palettes

| | |
|----------|--|
| "RdBu" |  |
| "PRGn" |  |
| "RdBu_r" |  |
| "PRGn_r" |  |

10. Diverging palettes

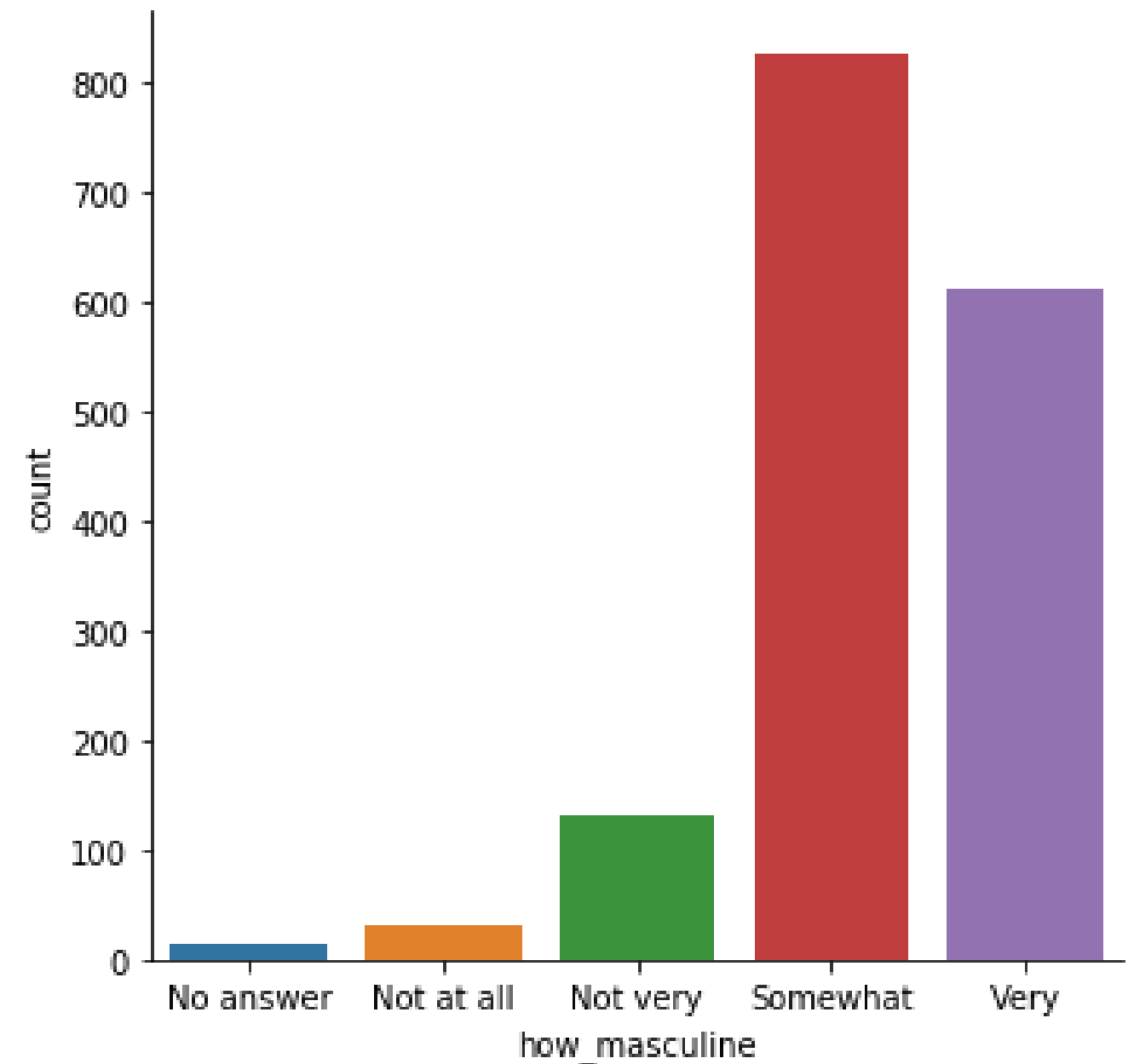
Seaborn has a group of preset palettes called diverging palettes that are great to use if your visualization deals with a scale where the two ends of the scale are opposites and there is a neutral midpoint. Here are some examples of diverging palettes - red/blue and purple/green. Note that if you append the palette name with "_r", you can reverse the palette.

Example (default palette)

```
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()
```

11. Example (default palette)

To see this in action, let's return to a count plot we've seen before of the responses of men reporting how masculine they feel.



Example (diverging palette)

```
sns.set_palette("RdBu")

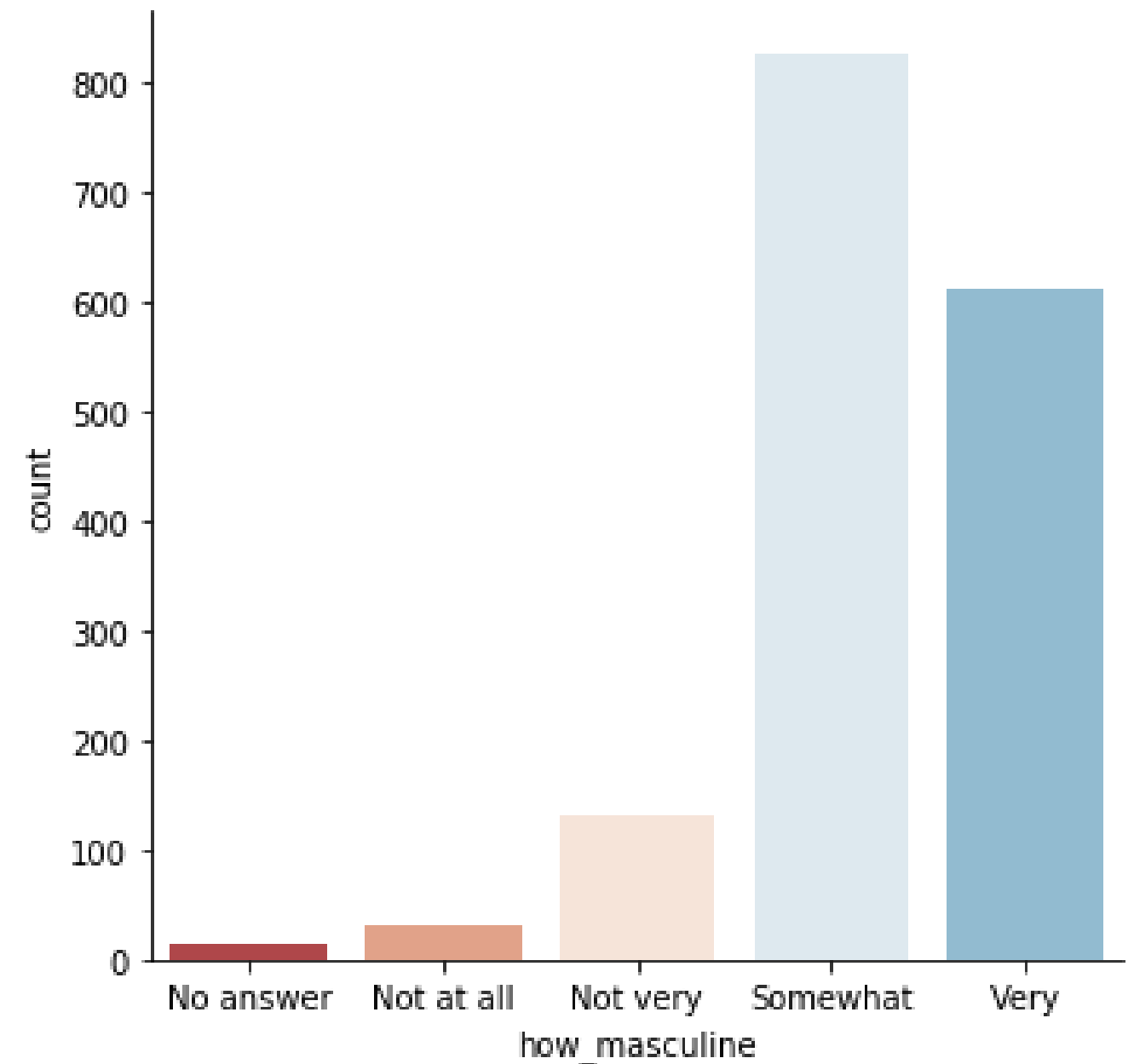
category_order = ["No answer",
                  "Not at all",
                  "Not very",
                  "Somewhat",
                  "Very"]

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





plt.show()
```

12. Example (diverging palette)

Setting this plot's palette to red/blue diverging provides a clearer contrast between the men who do not feel masculine and the men who do.



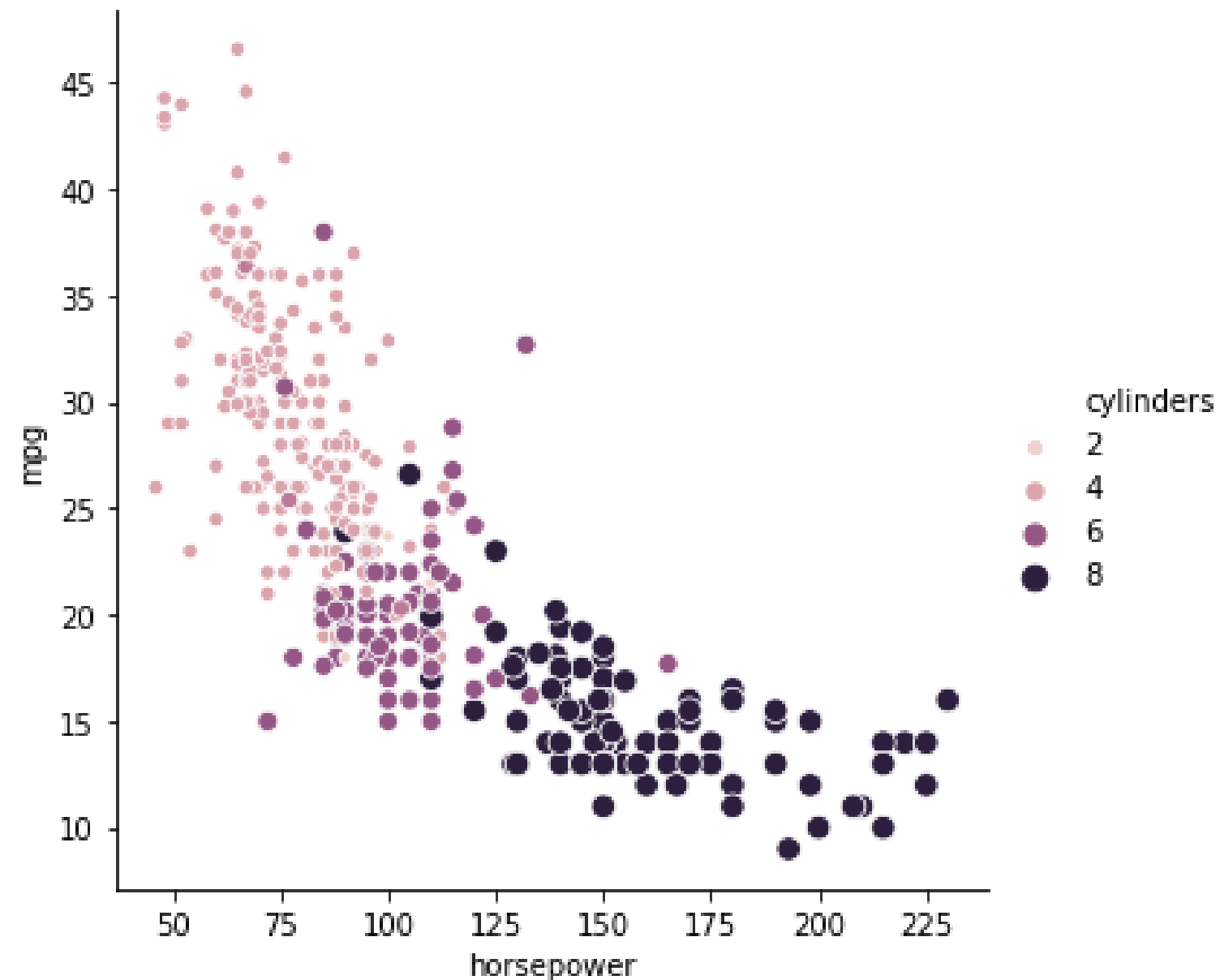
Sequential palettes

| | |
|---------|--|
| "Greys" |  |
| "Blues" |  |
| "PuRd" |  |
| "GnBu" |  |

13. Sequential palettes

Another group of palettes are called sequential palettes. These are a single color (or two colors blended) moving from light to dark values.

Sequential palette example

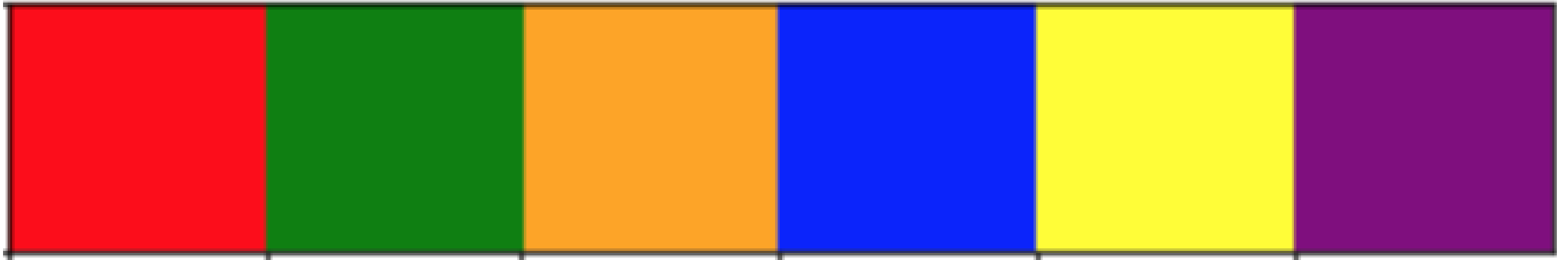


14. Sequential palette example
Sequential palettes are great for emphasizing a variable on a continuous scale. One example is this plot depicting the relationship between a car's horsepower and its miles per gallon, where points grow larger and darker when the car has more cylinders.

Custom palettes

```
custom_palette = ["red", "green", "orange", "blue",  
                  "yellow", "purple"]
```

```
sns.set_palette(custom_palette)
```



15. Custom palettes

You can also create your own custom palettes by passing in a list of color names...

Custom palettes

```
custom_palette = ['#FBB4AE', '#B3CDE3', '#CCEBC5',  
                  '#DECBE4', '#FED9A6', '#FFFFCC',  
                  '#E5D8BD', '#FDDAEC', '#F2F2F2']
```

```
sns.set_palette(custom_palette)
```



16. Custom palettes
or a list of hex color codes.

Changing the scale

- Figure "context" changes the scale of the plot elements and labels
- `sns.set_context()`
- Smallest to largest: "paper", "notebook", "talk", "poster"

17. Changing the scale

Finally, you can change the scale of your plot by using the "set context" function. The scale options from smallest to largest are "paper", "notebook", "talk", and "poster".

Default context: "paper"

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

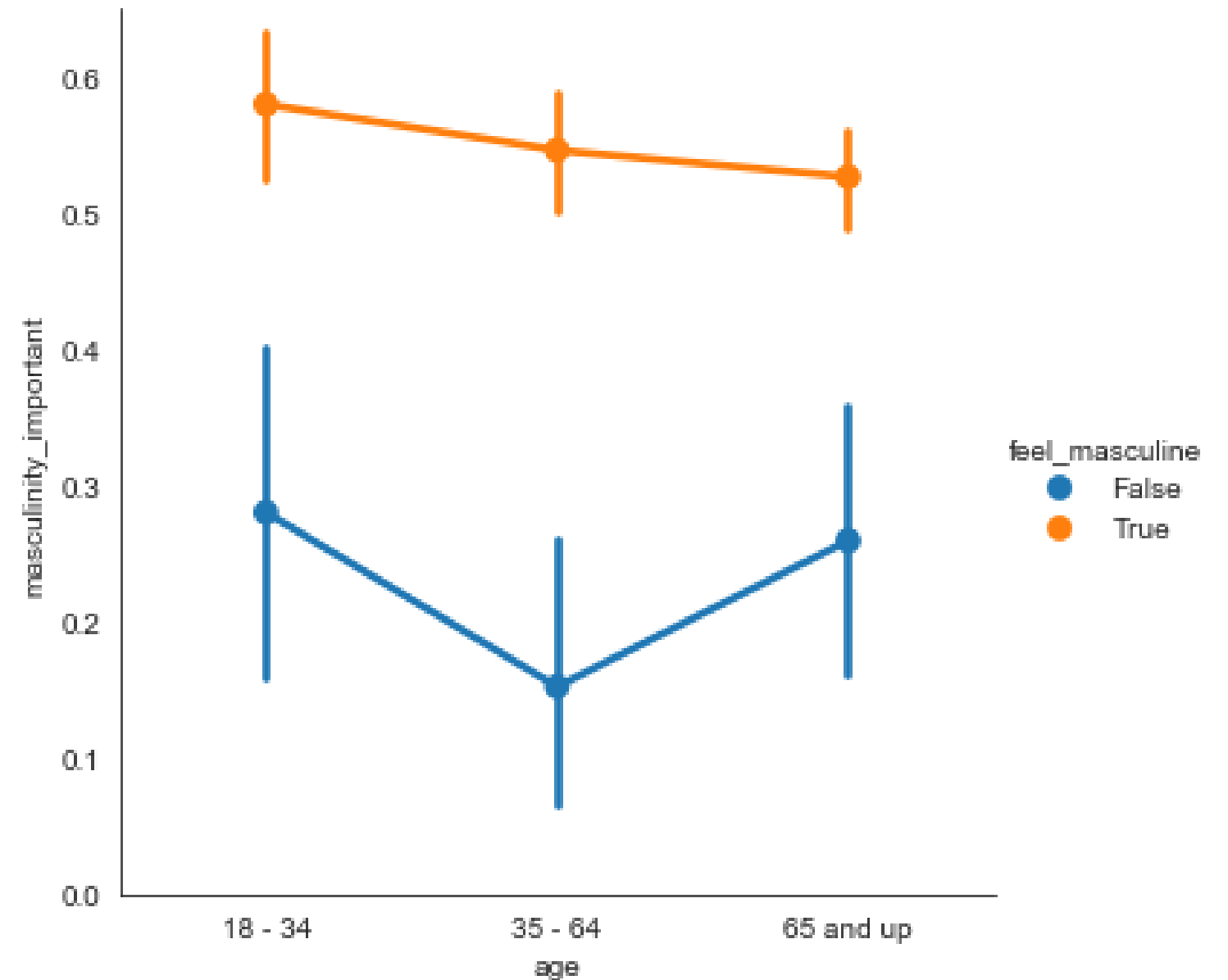
```
plt.show()
```

18. Default context: "paper"

The default context is "paper".

19. Larger context: "talk"

You'll want to choose a larger scale like "talk" for posters or presentations where the audience is further away from the plot.

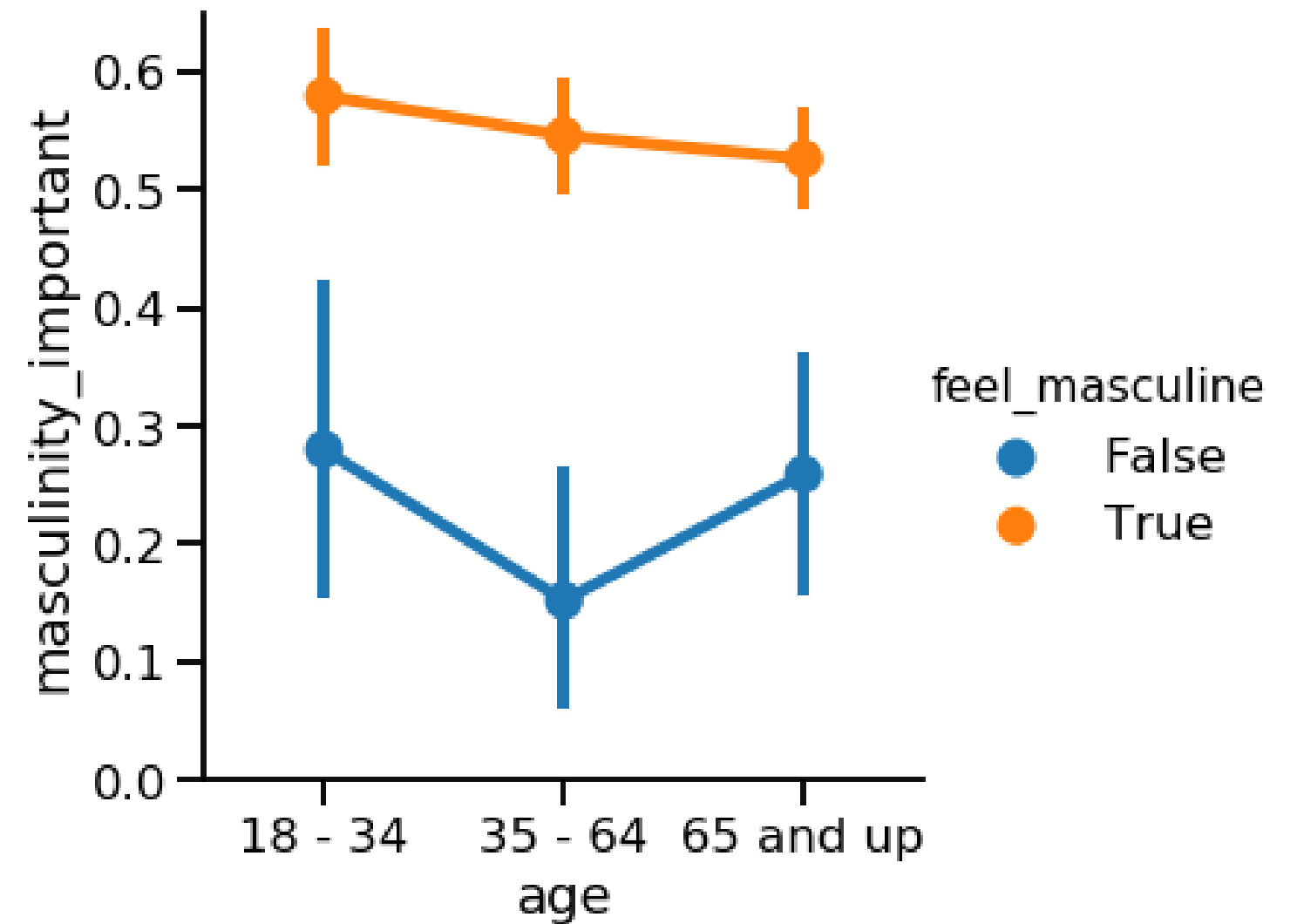


Larger context: "talk"

```
sns.set_context("talk")

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

plt.show()
```



Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

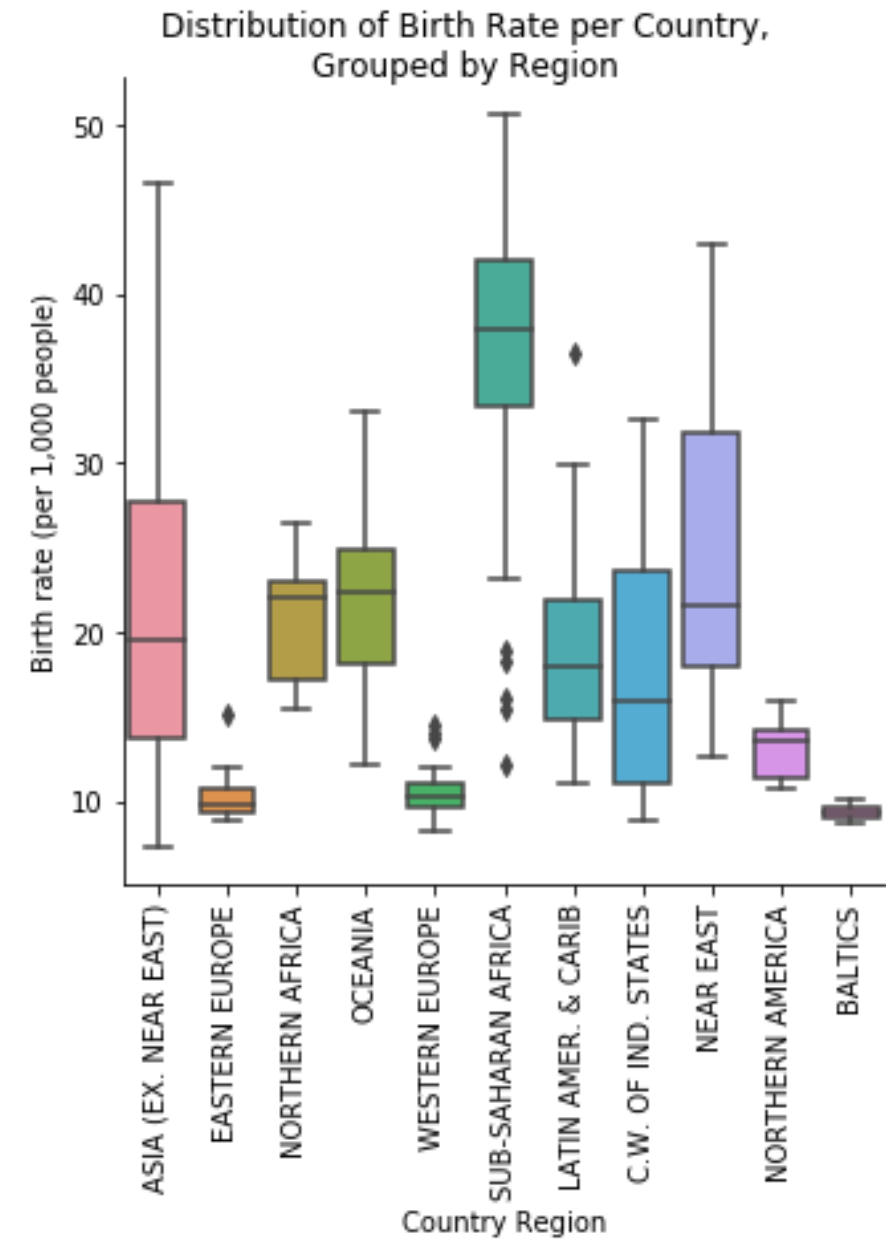
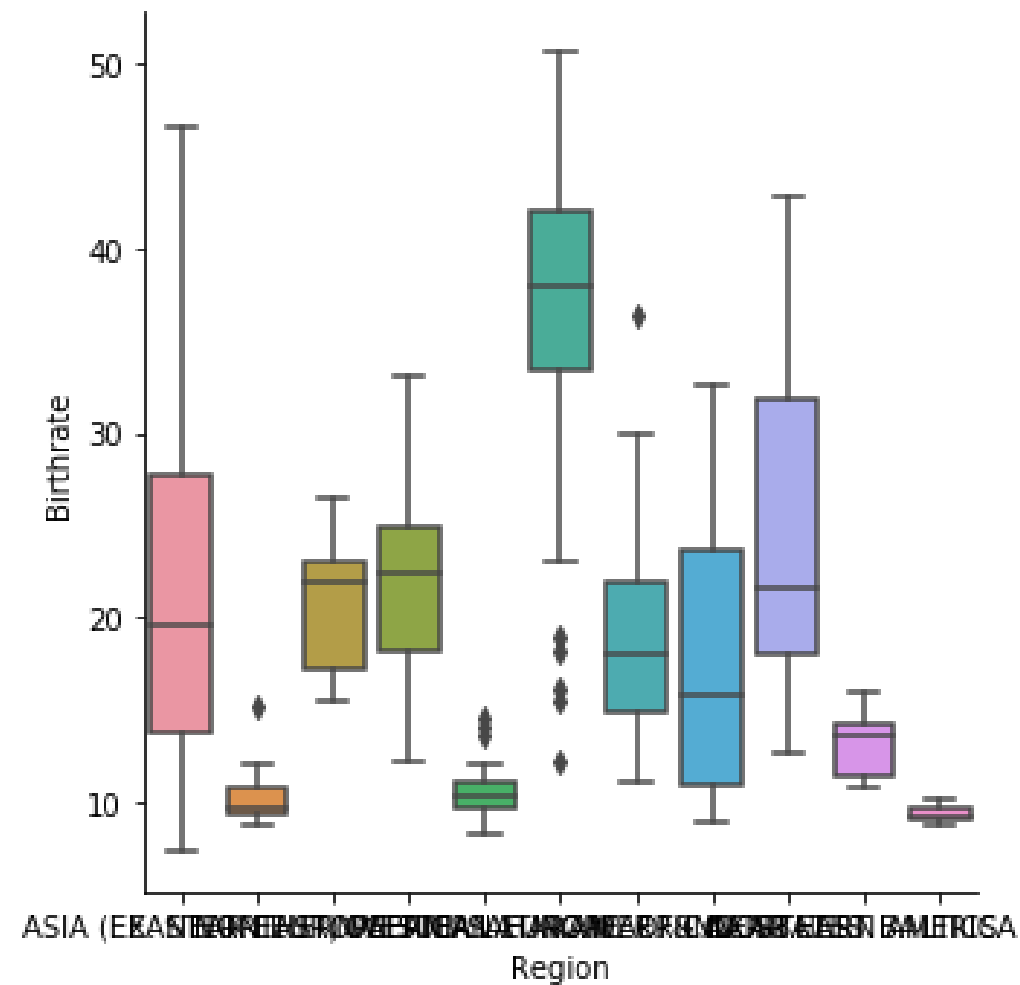
Adding titles and labels: Part 1

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



Erin Case
Data Scientist

Creating informative visualizations



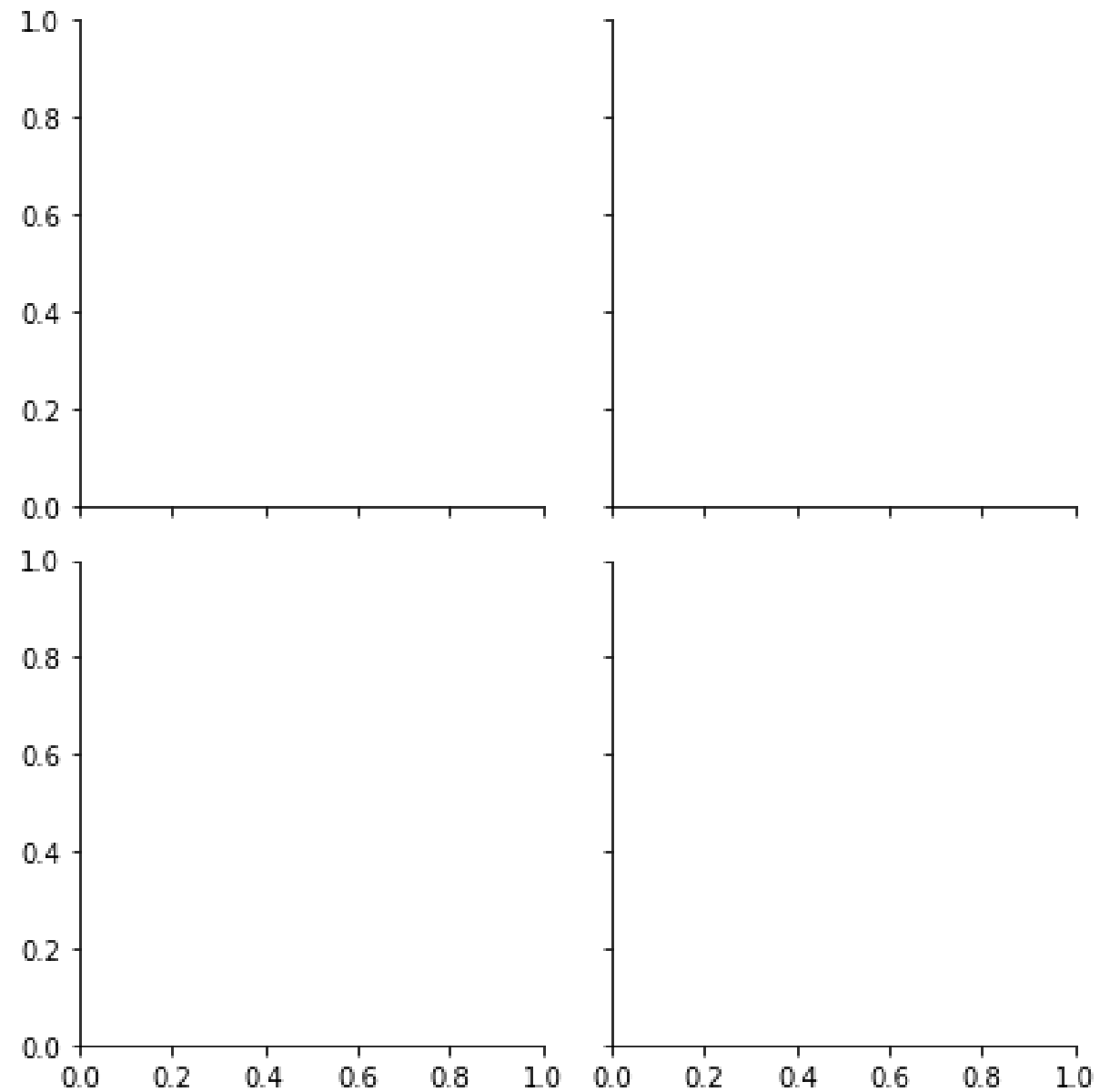
FacetGrid vs. AxesSubplot objects

Seaborn plots create two different types of objects: `FacetGrid` and `AxesSubplot`

```
g = sns.scatterplot(x="height", y="weight", data=df)
type(g)
```

```
> matplotlib.axes._subplots.AxesSubplot
```

An Empty FacetGrid



FacetGrid vs. AxesSubplot objects

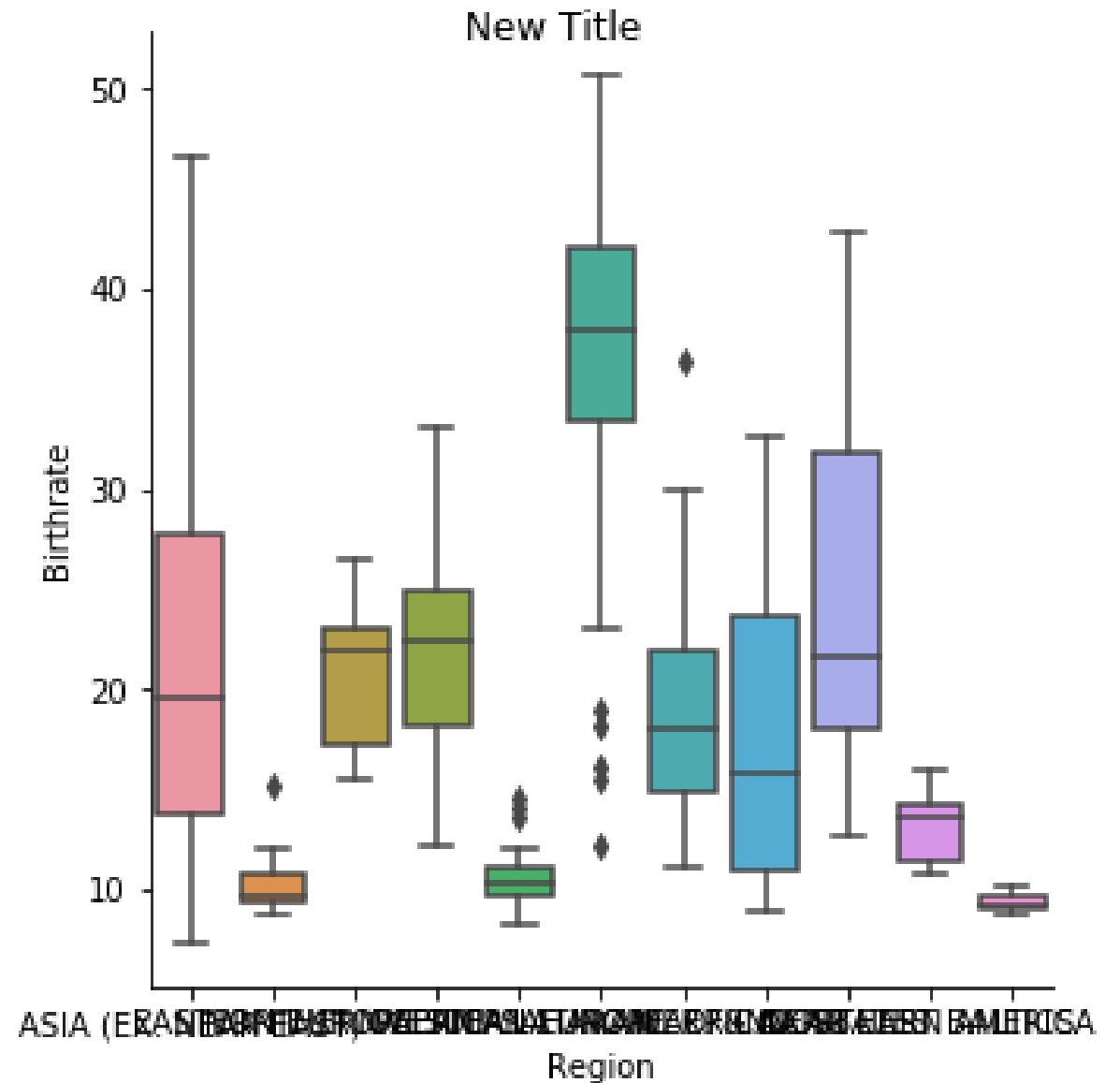
| Object Type | Plot Types | Characteristics |
|-------------|------------------------------------|----------------------------|
| FacetGrid | relplot() , catplot() | Can create subplots |
| AxesSubplot | scatterplot() , countplot() , etc. | Only creates a single plot |

Adding a title to FacetGrid

```
g = sns.catplot(x="Region",
                 y="Birthrate",
                 data=gdp_data,
                 kind="box")
g.fig.suptitle("New Title")
plt.show()
```

Let's return to our messy plot from the beginning. Recall that "catplot()" enables subplots, so it returns a FacetGrid object. To add a title to a FacetGrid object, first assign the plot to the variable "g". After you assign the plot to "g", you can set the title using "g dot fig dot suptitle". This tells Seaborn you want to set a title for the figure as a whole.

Let's return to our messy plot from the beginning. Recall that "catplot()" enables subplots, so it returns a FacetGrid object. To add a title to a FacetGrid object, first assign the plot to the variable "g". After you assign the plot to "g", you can set the title using "g dot fig dot suptitle". This tells Seaborn you want to set a title for the figure as a whole.

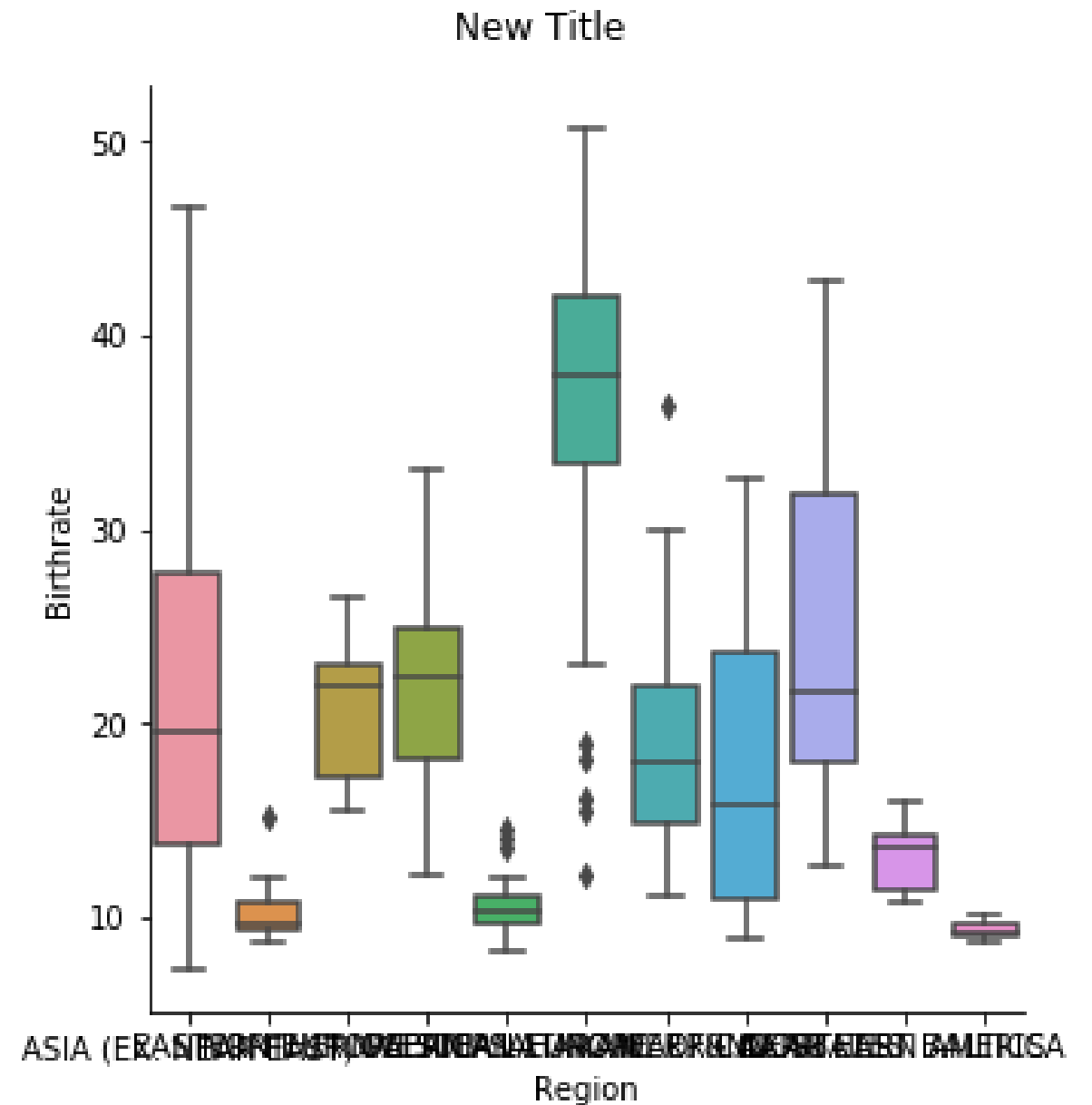


Adjusting height of title in FacetGrid

```
g = sns.catplot(x="Region",  
                y="Birthrate",  
                data=gdp_data,  
                kind="box")  
  
g.fig.suptitle("New Title",  
               y=1.03)  
  
plt.show()
```

7. Adjusting height of title in FacetGrid

Note that by default, the figure title might be a little low. To adjust the height of the title, you can use the "y" parameter. The default value is 1, so setting it to 1 point 03 will make it a little higher than the default.



Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Adding titles and labels: Part 2

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



Erin Case
Data Scientist

Adding a title to AxesSubplot

FacetGrid

```
g = sns.catplot(x="Region",  
                y="Birthrate",  
                data=gdp_data,  
                kind="box")
```

```
g.fig.suptitle("New Title",  
               y=1.03)
```

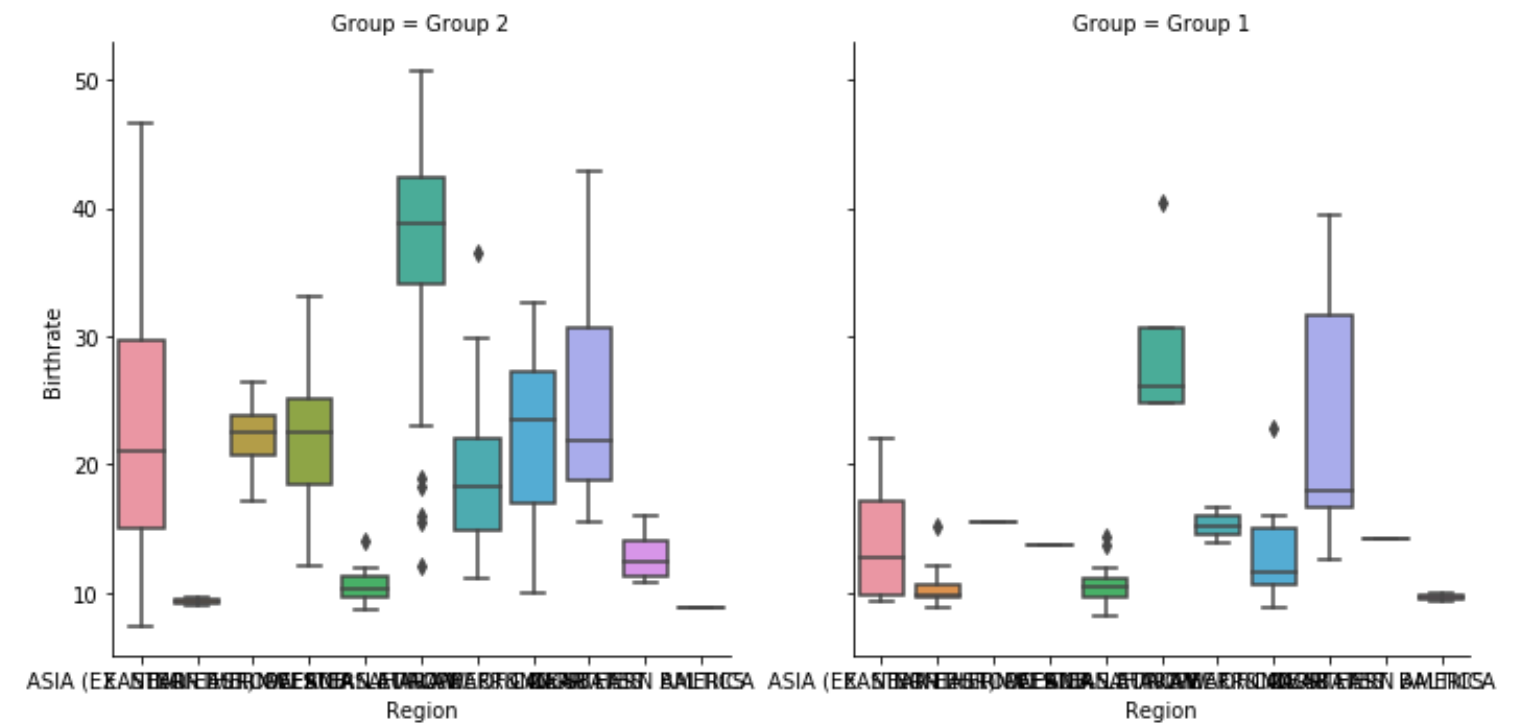
AxesSubplot

```
g = sns.boxplot(x="Region",  
                y="Birthrate",  
                data=gdp_data)
```

```
g.set_title("New Title",  
            y=1.03)
```

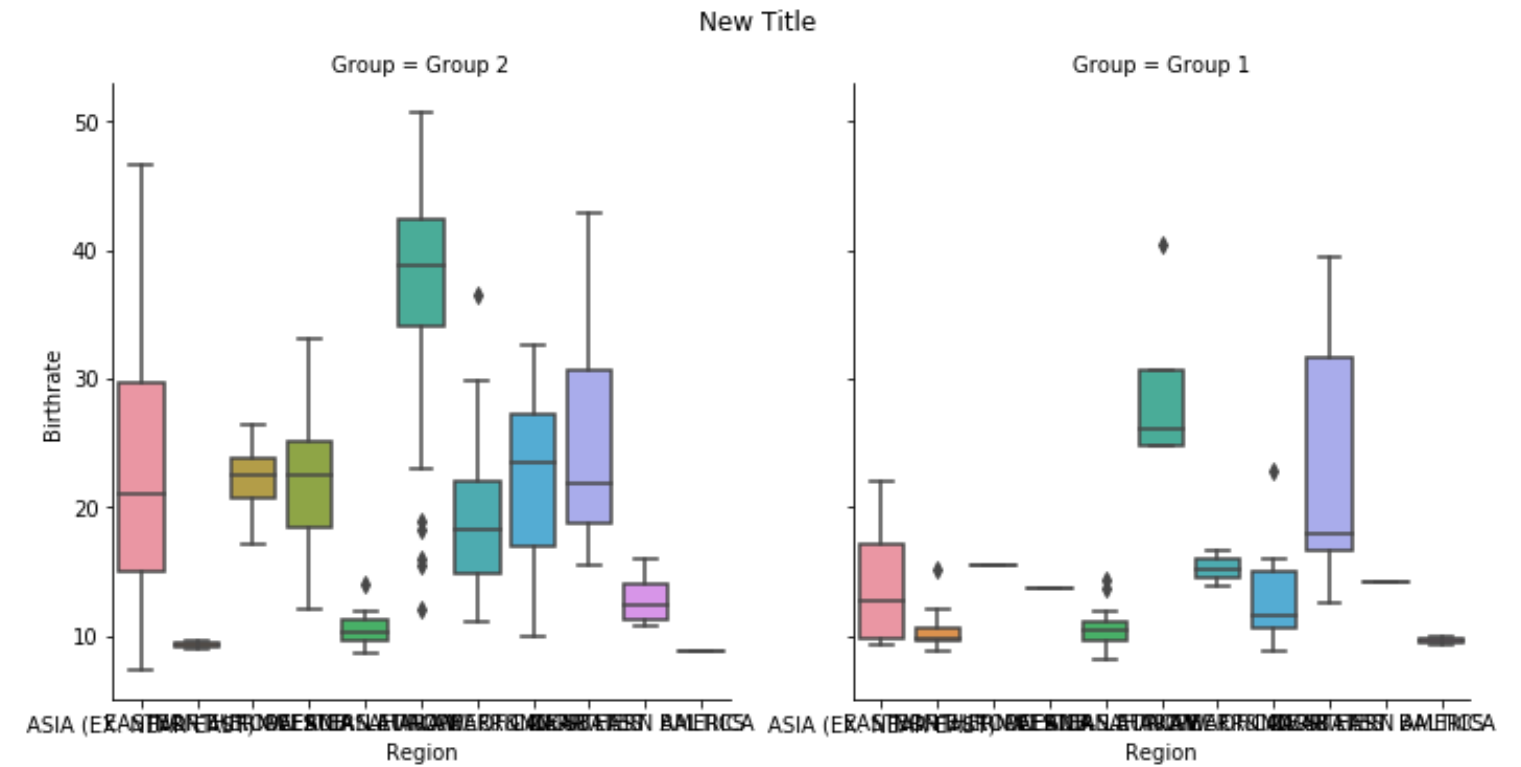
Titles for subplots

```
g = sns.catplot(x="Region",  
                y="Birthrate",  
                data=gdp_data,  
                kind="box",  
                col="Group")
```



Titles for subplots

```
g = sns.catplot(x="Region",  
                y="Birthrate",  
                data=gdp_data,  
                kind="box",  
                col="Group")  
  
g.fig.suptitle("New Title",  
               y=1.03)
```

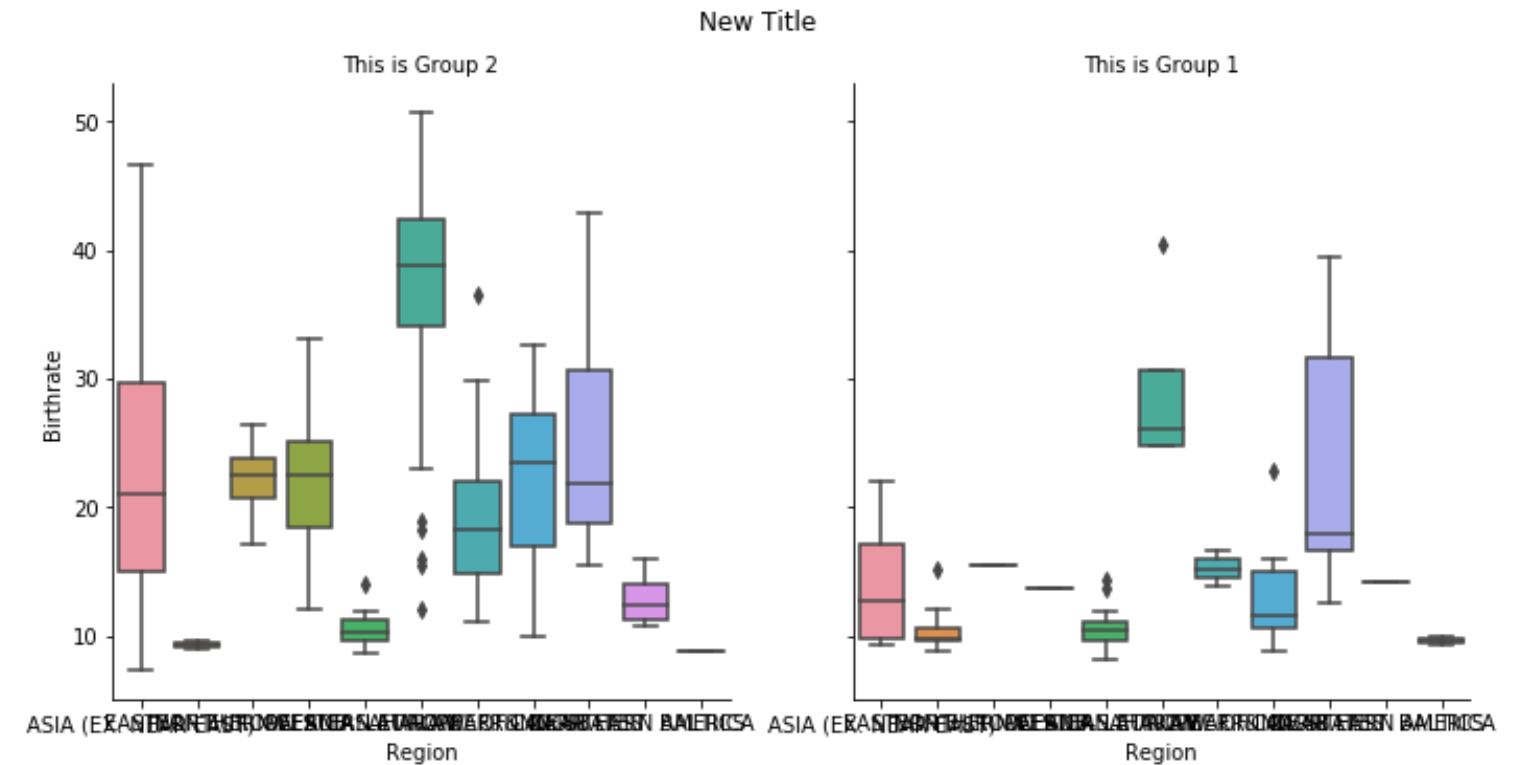


Titles for subplots

```
g = sns.catplot(x="Region",
                y="Birthrate",
                data=gdp_data,
                kind="box",
                col="Group")

g.fig.suptitle("New Title",
              y=1.03)

g.set_titles("This is {col_name}")
```

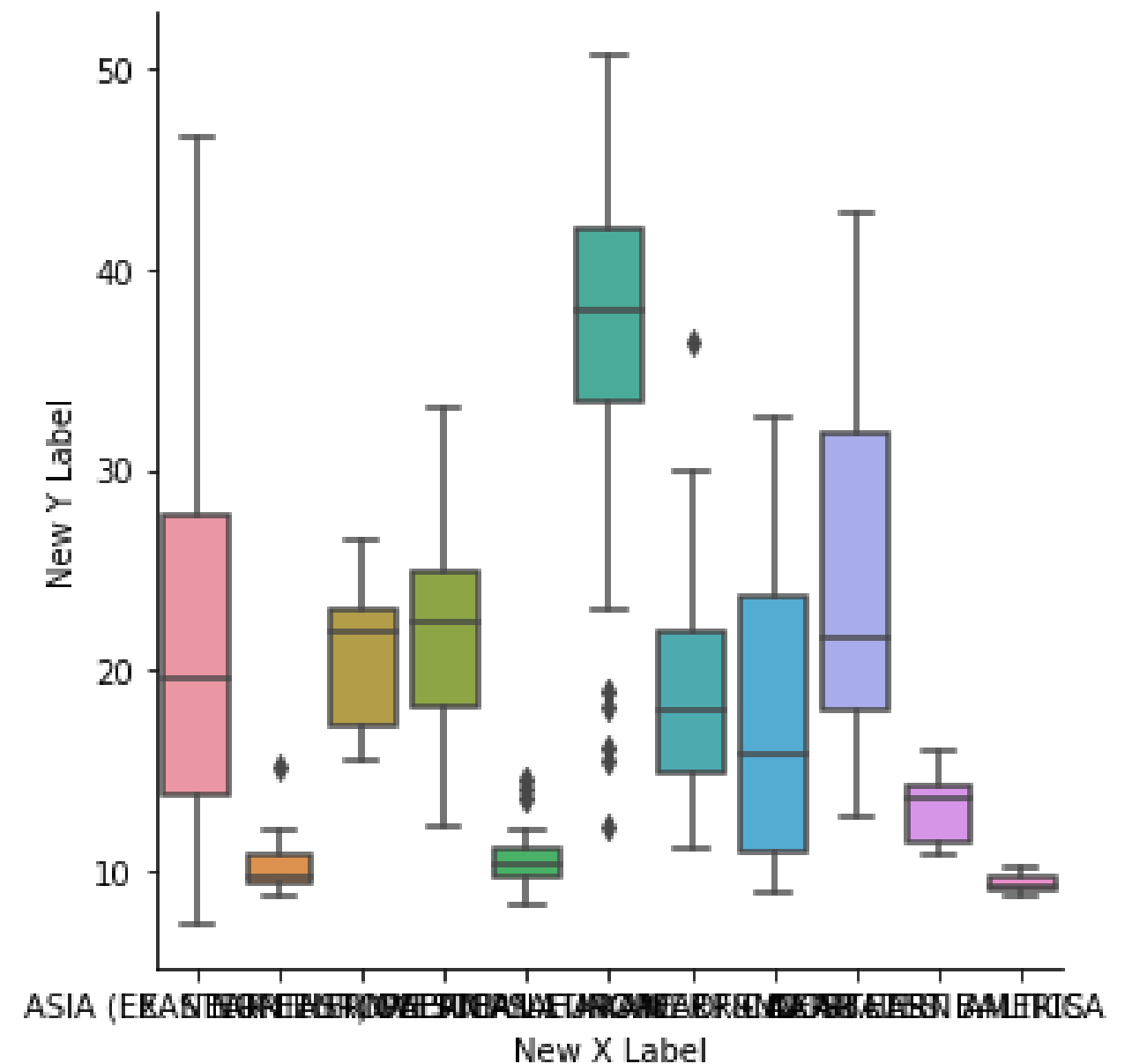


Adding axis labels

```
g = sns.catplot(x="Region",  
                y="Birthrate",  
                data=gdp_data,  
                kind="box")
```

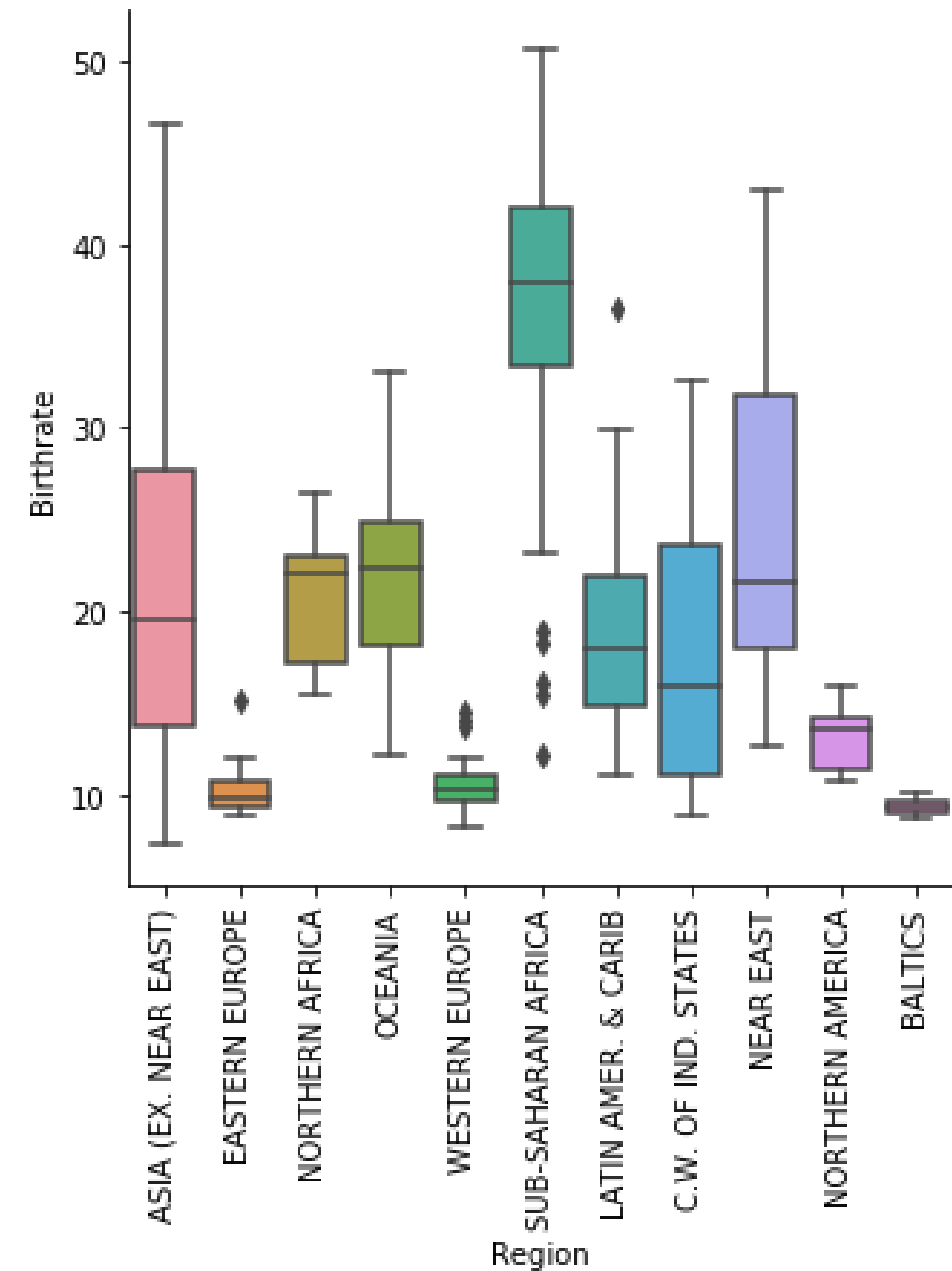
```
g.set(xlabel="New X Label",  
      ylabel="New Y Label")
```

```
plt.show()
```



Rotating x-axis tick labels

```
g = sns.catplot(x="Region",  
                y="Birthrate",  
                data=gdp_data,  
                kind="box")  
  
plt.xticks(rotation=90)  
plt.show()
```



Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

Putting it all together

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



Erin Case
Data Scientist

Getting started

To import Seaborn:

```
import seaborn as sns
```

To import Matplotlib:

```
import matplotlib.pyplot as plt
```

To show a plot:

```
plt.show()
```

Relational plots

- Show the relationship between two quantitative variables
- Examples: scatter plots, line plots

```
sns.relplot(x="x_variable_name",  
            y="y_variable_name",  
            data=pandas_df,  
            kind="scatter")
```

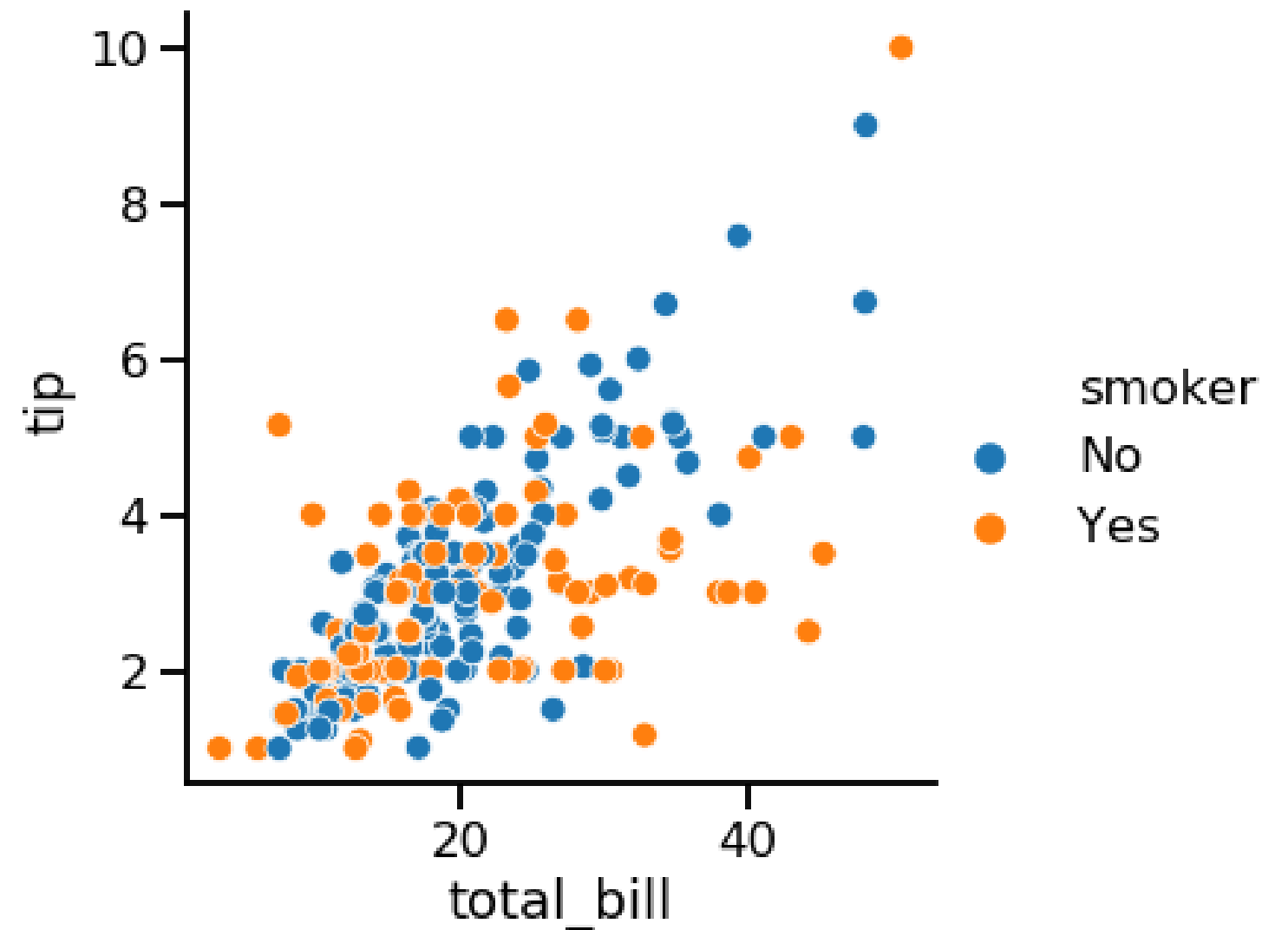
Categorical plots

- Show the distribution of a quantitative variable within categories defined by a categorical variable
- Examples: bar plots, count plots, box plots, point plots

```
sns.catplot(x="x_variable_name",  
            y="y_variable_name",  
            data=pandas_df,  
            kind="bar")
```

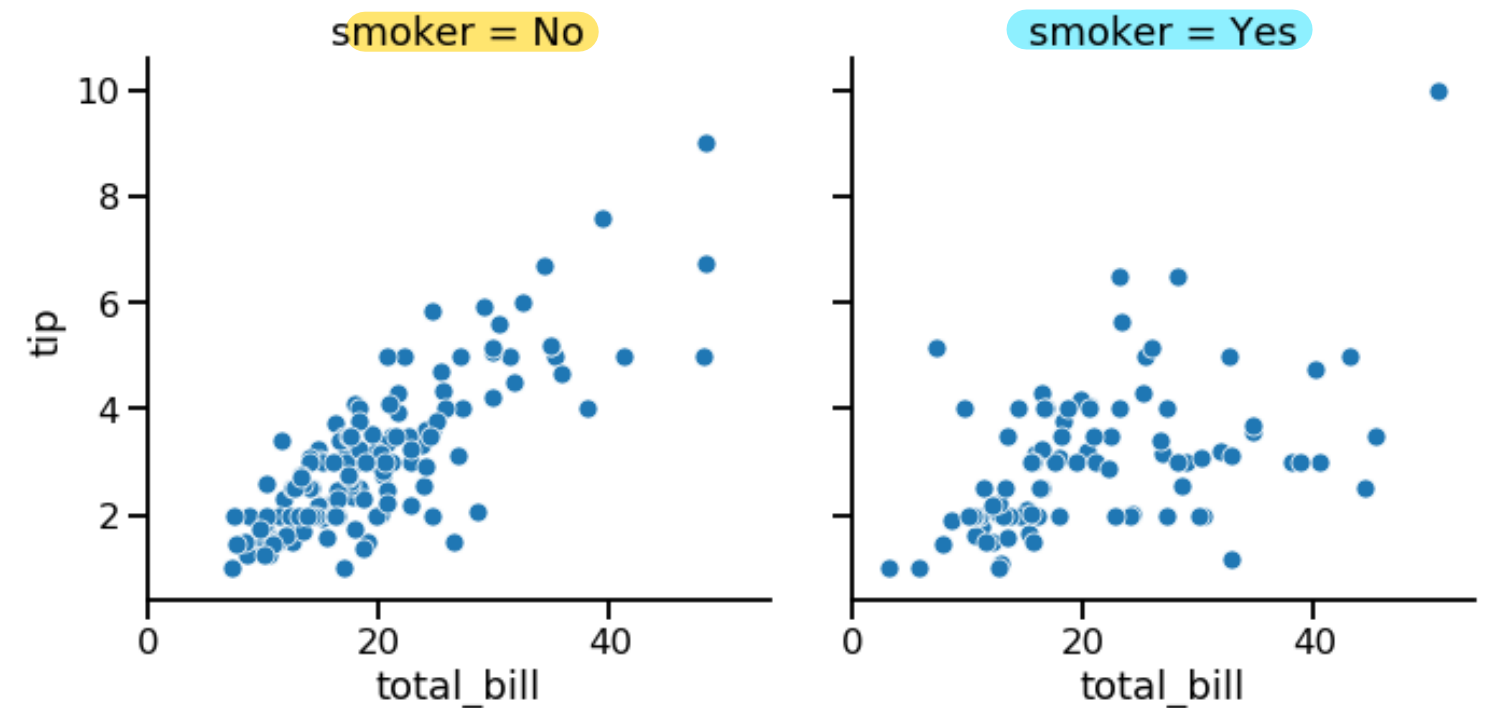

Adding a third variable (hue)

Setting `hue` will create subgroups that are displayed as different colors on a single plot.



Adding a third variable (row/col)

Setting `row` and/or `col` in `relplot()` or `catplot()` will create subgroups that are displayed on separate subplots.



Customization

- Change the background: `sns.set_style()`
- Change the main element colors: `sns.set_palette()`
- Change the scale: `sns.set_context()`

Adding a title

| Object Type | Plot Types | How to Add Title |
|-------------|------------------------------------|------------------|
| FacetGrid | relplot() , catplot() | g.fig.suptitle() |
| AxesSubplot | scatterplot() , countplot() , etc. | g.set_title() |

Final touches

Add x- and y-axis labels:

```
g.set(xlabel="new x-axis label",  
      ylabel="new y-axis label")
```

Rotate x-tick labels:

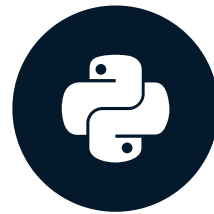
```
plt.xticks(rotation=90)
```

Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

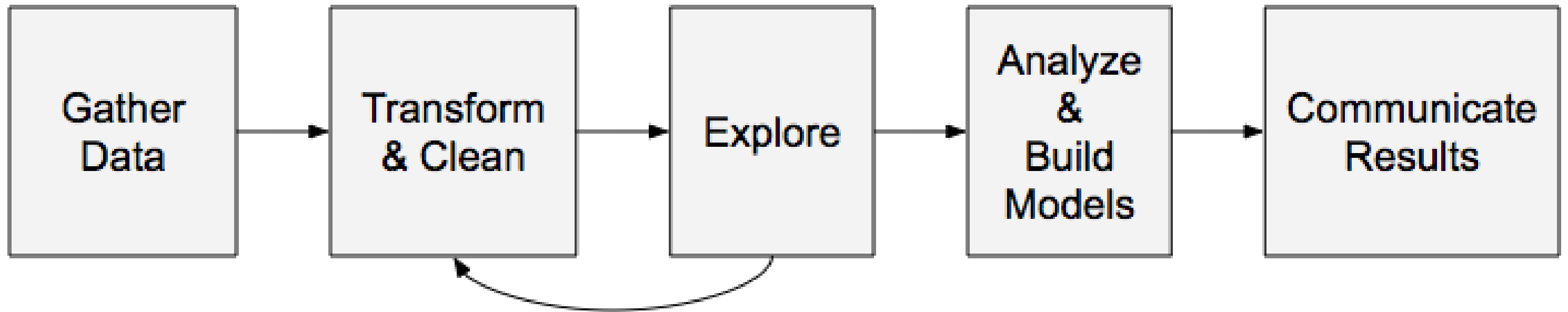
Well done! What's next?

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN

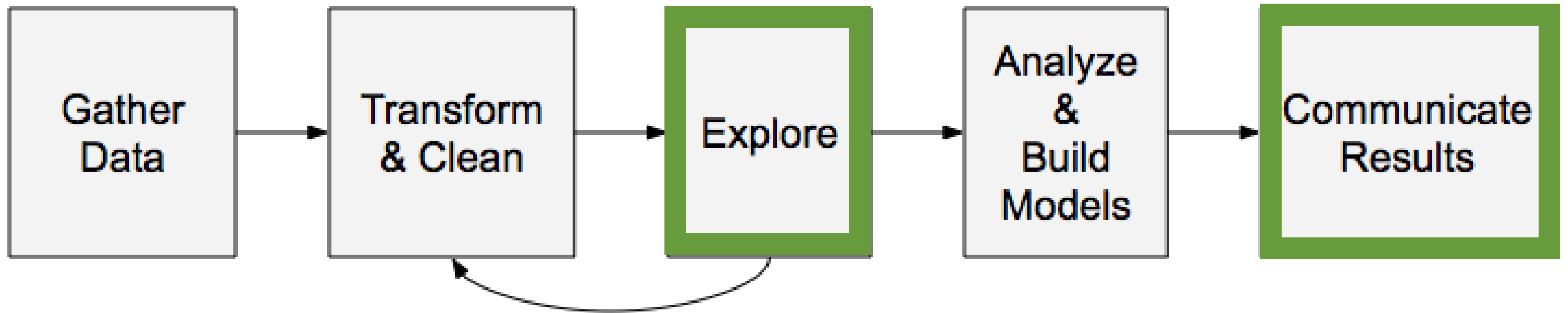


Erin Case
Data Scientist

Where does Seaborn fit in?



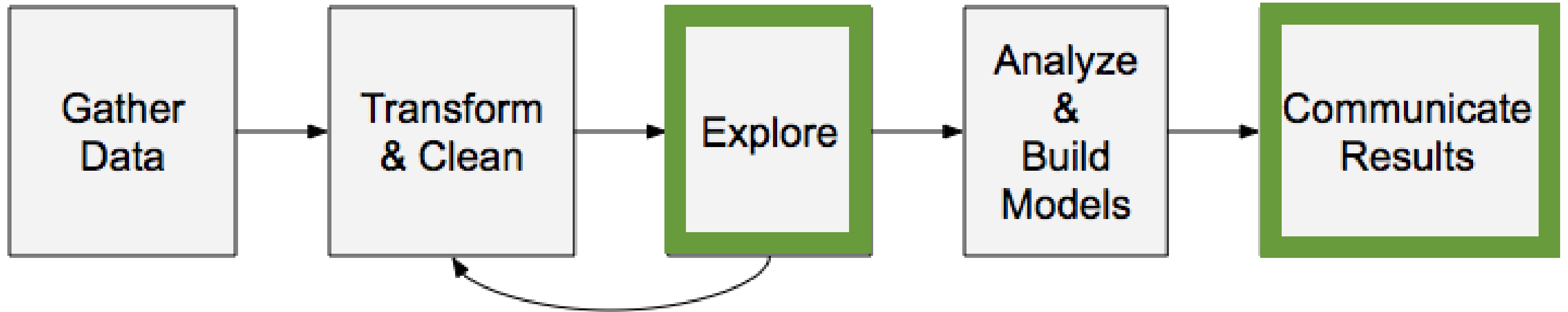
Where does Seaborn fit in?



3. Where does Seaborn fit in?

As we've seen in our examples, Seaborn is great for both the initial exploration of your data and communicating the results at the end of your data analysis.

Next Steps: Explore and communicate results



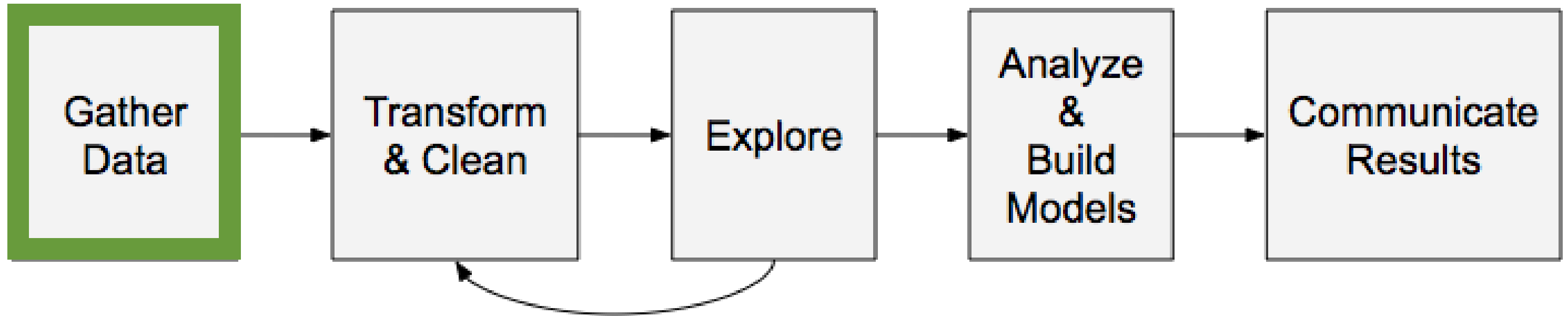
Next steps:

- Seaborn advanced visualizations
- Matplotlib advanced customizations

4. Next Steps: Explore and communicate results

In this course, we've covered the most common data visualizations used for data exploration. DataCamp has other visualization courses if you want to learn even more. For example, Seaborn also supports more advanced visualizations and analyses like linear regressions. We also learned that Seaborn was built on top of Matplotlib and practiced how to use some Matplotlib functions to customize Seaborn plots. Here, too, there are many more customizations that Matplotlib supports if you wish to learn more.

Next steps: Gather data



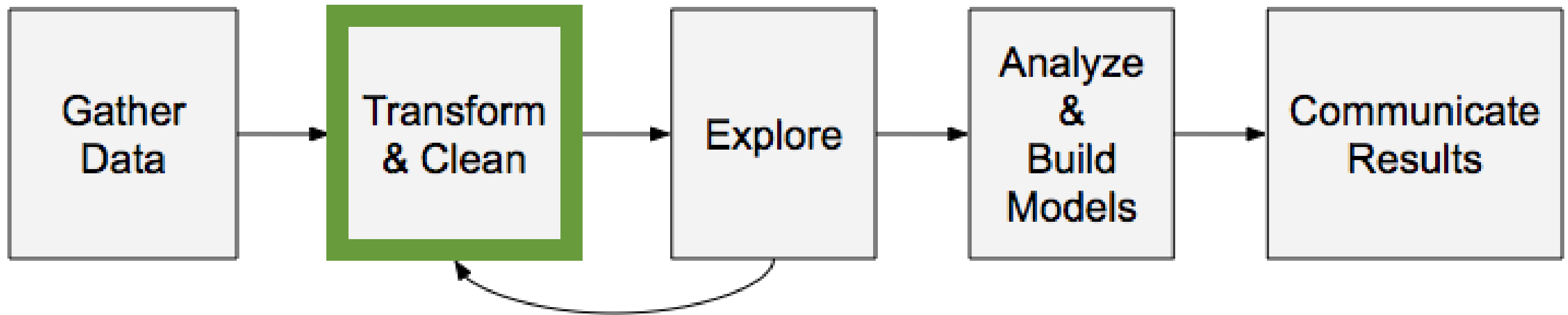
Next steps:

- Python
- SQL

5. Next steps: Gather data

You can also learn more about the other steps of the data analysis workflow. If you wish to learn more about how to gather your data, explore courses on importing data in Python and SQL.

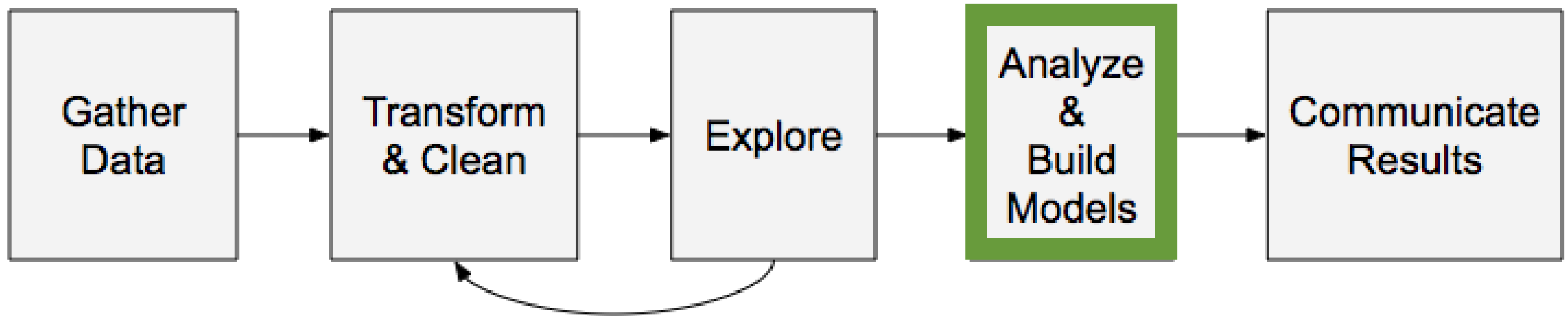
Next steps: Transform and clean



Next steps:

- Getting data into Pandas DataFrames
- Cleaning data
- Transforming into tidy format

Next steps: Analyze and build models



Next steps:

- Statistical analysis
- Calculating and interpreting confidence intervals

6. Next steps: Transform and clean
In this course, we learned that Seaborn works extremely well with tidy Pandas DataFrames. There is more to learn here about how to get your data into Pandas DataFrames, clean it, and transform it into a tidy format.

7. Next steps: Analyze and build models
Finally, I encourage you to learn more about statistical analysis. For example, for bar plots, Seaborn automatically calculates confidence intervals for each bar value. There is a lot to learn here about how these confidence intervals are calculated and how to interpret them.

Congratulations!

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN