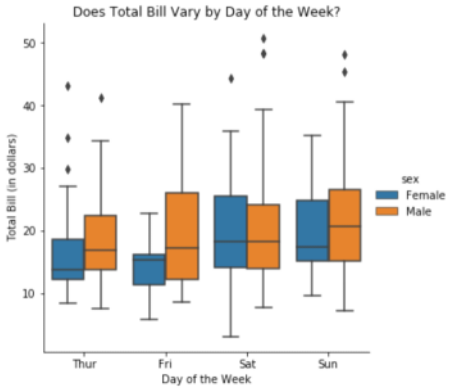
Introduction to Data Visualization with Seaborn

Tuesday, 6 October 2020 5:00 PM

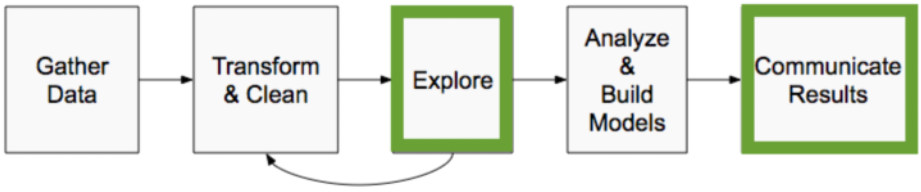
Introduction to Seaborn

What is Seaborn?

- Python data visualization library
- Easily create the most common types of plots



Why is Seaborn useful?

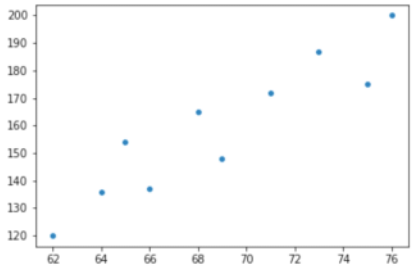


Advantages of Seaborn

- Easy to use
- Works well with `pandas` data structures
- Built on top of `matplotlib`

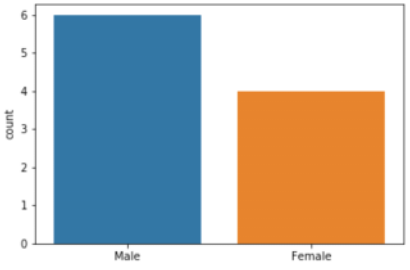
Example 1: Scatter plot

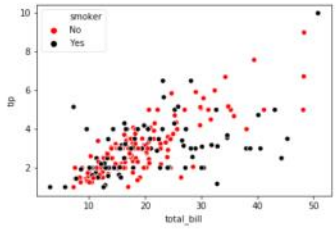
```
import seaborn as sns
import matplotlib.pyplot as plt
height = [62, 64, 69, 75, 66,
          68, 65, 71, 76, 73]
weight = [120, 136, 148, 175, 137,
          165, 154, 172, 200, 187]
sns.scatterplot(x=height, y=weight)
plt.show()
```

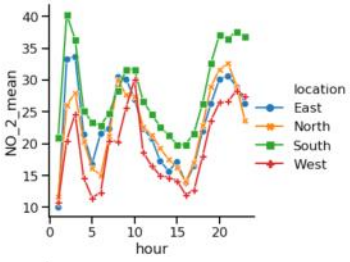


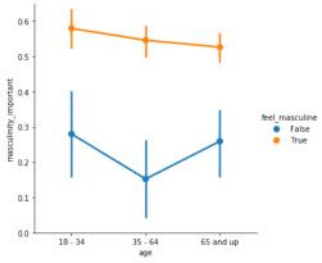
Example 2: Create a count plot

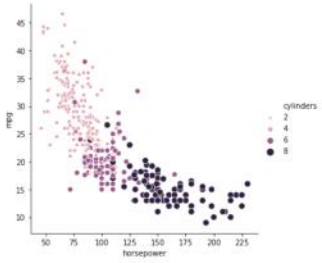
```
import seaborn as sns
import matplotlib.pyplot as plt
gender = ["Female", "Female",
          "Female", "Female",
          "Male", "Male", "Male",
          "Male", "Male", "Male"]
sns.countplot(x=gender)
plt.show()
```









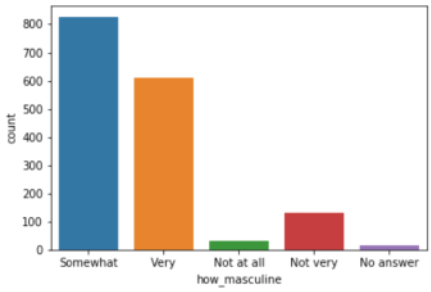


What is pandas?

- Python library for data analysis
- Easily read datasets from csv, txt, and other types of files
- Datasets take the form of `DataFrame` objects

Using DataFrames with countplot()

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv("masculinity.csv")
sns.countplot(x="how_masculine",
              data=df)
plt.show()
```



"Tidy data" means that each observation has its own row and each variable has its own column.

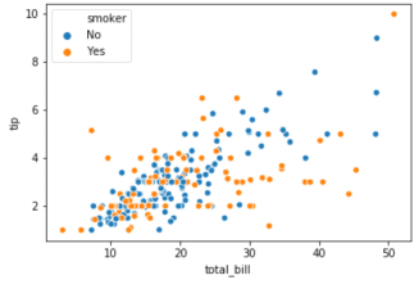
"Tidy data" means that each observation has its own row and each variable has its own column.

	AMONG ADULT MEN	Unnamed: 1	Adult Men	Age	Unnamed: 4	Unnamed: 5
0				18 - 34	35 - 64	65 and up
1	In general, how masculine or "manly" do you feel?					
2		Very masculine	37%	29%	42%	37%
3		Somewhat masculine	46%	47%	46%	47%
4		Not very masculine	11%	13%	9%	13%
5		Not at all masculine	5%	10%	2%	3%
6		No answer	1%	0%	1%	1%
7	How important is it to you that others see you as masculine?					
8		Very important	16%	18%	17%	13%
9		Somewhat important	37%	38%	37%	32%
10		Not too important	28%	18%	31%	37%
11		Not at all important	18%	26%	15%	18%
12		No answer	0%	0%	1%	0%

In contrast, here is an example of an "untidy" DataFrame made from the same survey on masculinity.

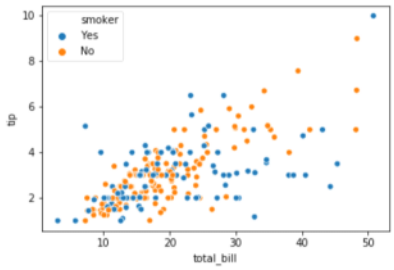
A scatter plot with hue

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

sns.scatterplot(x="total_bill",
                y="tip",
                data=tips,
                hue="smoker")
plt.show()
```



Setting hue order

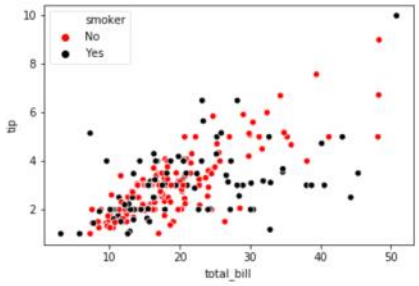
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.scatterplot(x="total_bill",
                y="tip",
                data=tips,
                hue="smoker",
                hue_order=["Yes",
                           "No"])
plt.show()
```



The "hue order" parameter takes in a list of values and will set the order of the values in the plot accordingly.

Specifying hue colors

```
import matplotlib.pyplot as plt
import seaborn as sns
hue_colors = {"Yes": "black",
              "No": "red"}
sns.scatterplot(x="total_bill",
                y="tip",
                data=tips,
                hue="smoker",
                palette=hue_colors)
```

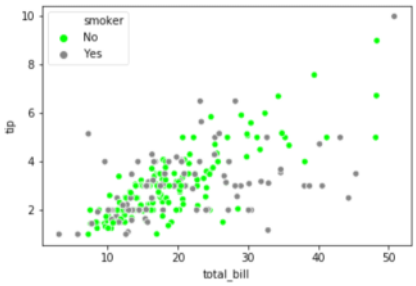


	Color	Matplotlib name	Matplotlib abbreviation	HTML color code (hex)
	blue	"blue"	"b"	#0000ff
	green	"green"	"g"	#008000
	red	"red"	"r"	#ff0000
	green/blue	"cyan"	"c"	#00bfbf
	purple	"magenta"	"m"	#bf00bf
	yellow	"yellow"	"y"	#bfbf00
	black	"black"	"k"	#000000
	white	"white"	"w"	#ffffff

Here is the list of Matplotlib colors and their names.

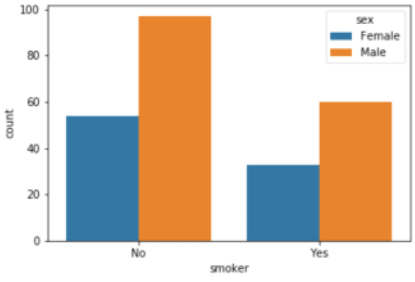
Using HTML hex color codes with hue

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

hue_colors = {"Yes": "#808080",
              "No": "#00FF00"}

sns.scatterplot(x="total_bill",
                y="tip",
                data=tips,
                hue="smoker",
                palette=hue_colors)
plt.show()
```



Using hue with count plots

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

sns.countplot(x="smoker",
              data=tips,
              hue="sex")
plt.show()
```



Visualizing Two Quantitative Variables

Introducing relplot()

- Create "relational plots": scatter plots or line plots

Why use `relplot()` instead of `scatterplot()` ?

- `relplot()` lets you create subplots in a single figure

scatterplot() vs. relplot()

Using `scatterplot()`

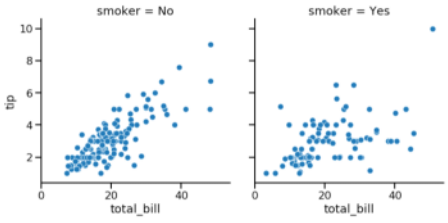
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.scatterplot(x="total_bill",
                y="tip",
                data=tips)

plt.show()
```

Using `relplot()`

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

sns.relplot(x="total_bill",
            y="tip",
            data=tips,
            kind="scatter")

plt.show()
```

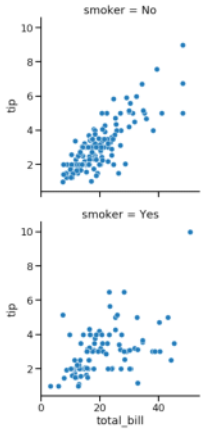
Subplots in columns

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

sns.relplot(x="total_bill",
            y="tip",
            data=tips,
            kind="scatter",
            col="smoker")

plt.show()
```



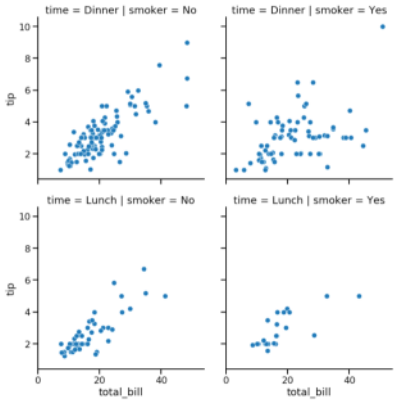
Subplots in rows

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

sns.relplot(x="total_bill",
            y="tip",
            data=tips,
            kind="scatter",
            row="smoker")

plt.show()
```

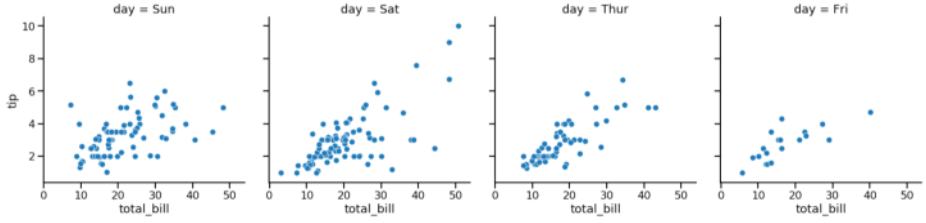


Subplots in rows and columns

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

sns.relplot(x="total_bill",
            y="tip",
            data=tips,
            kind="scatter",
            col="smoker",
            row="time")

plt.show()
```

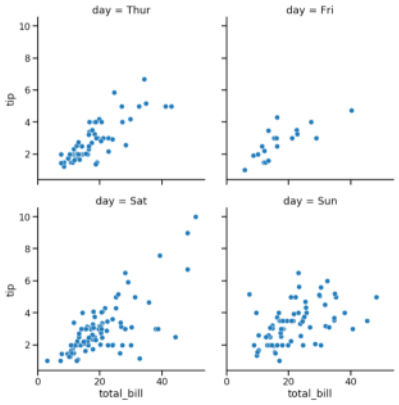


Subgroups for days of the week



Ordering columns

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

sns.relplot(x="total_bill",
            y="tip",
            data=tips,
            kind="scatter",
            col="day",
            col_wrap=2,
            col_order=["Thur",
                       "Fri",
                       "Sat",
                       "Sun"])

plt.show()
```



Scatter plot overview

Show relationship between two quantitative variables

We've seen:

- Subplots (`col` and `row`)
- Subgroups with color (`hue`)

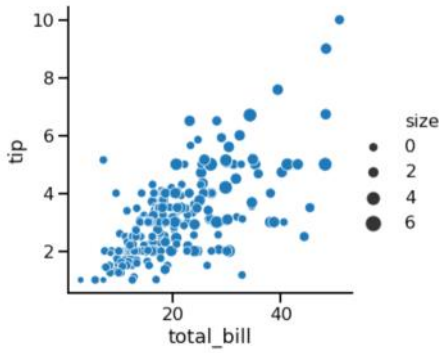
New Customizations:

- Subgroups with point size and style
- Changing point transparency

Use with both `scatterplot()` and `relplot()`

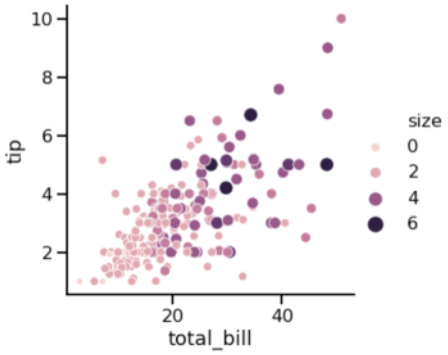
Subgroups with point size

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

sns.relplot(x="total_bill",
            y="tip",
            data=tips,
            kind="scatter",
            size="size")
plt.show()
```



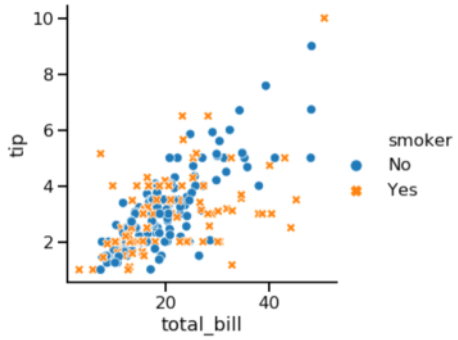
Point size and hue

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

sns.relplot(x="total_bill",
            y="tip",
            data=tips,
            kind="scatter",
            size="size",
            hue="size")
plt.show()
```



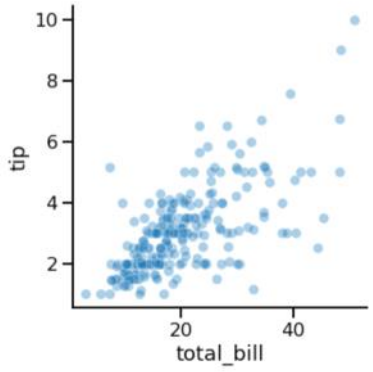
Subgroups with point style

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

sns.relplot(x="total_bill",
            y="tip",
            data=tips,
            kind="scatter",
            hue="smoker",
            style="smoker")
plt.show()
```



Changing point transparency

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

# Set alpha to be between 0 and 1
sns.relplot(x="total_bill",
            y="tip",
            data=tips,
            kind="scatter",
            alpha=0.4)
plt.show()
```



What are line plots?

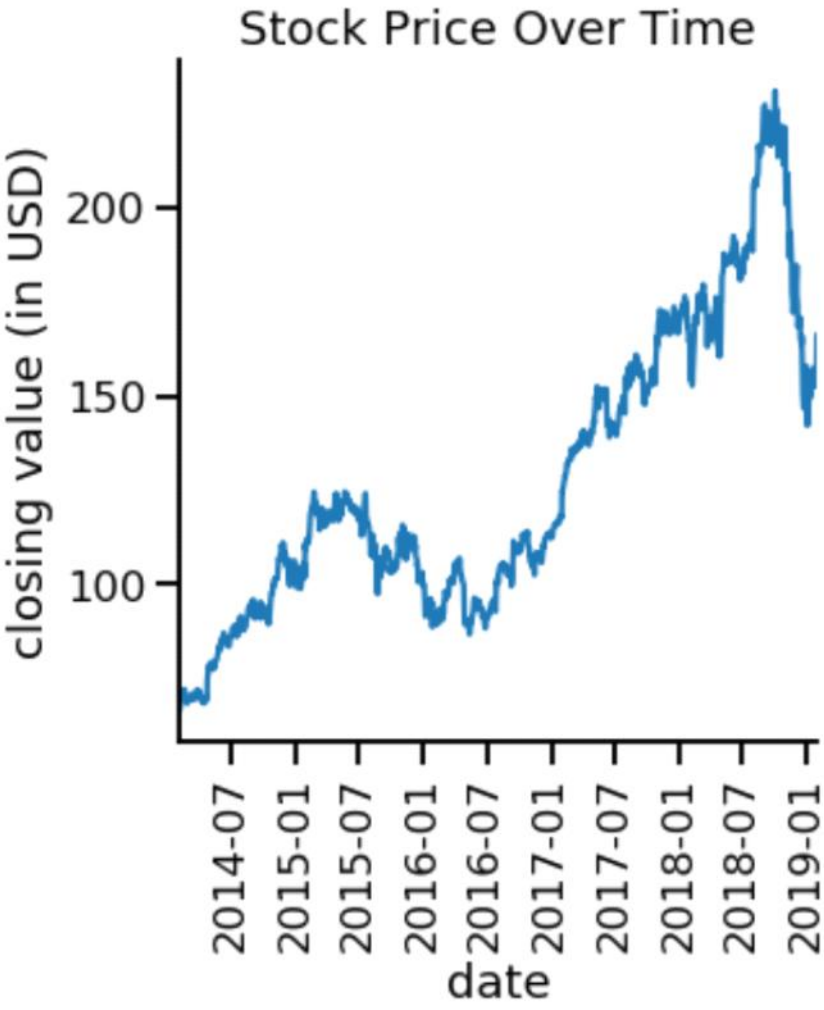
Two types of relational plots: scatter plots and line plots

Scatter plots

- Each plot point is an independent observation

Line plots

- Each plot point represents the same "thing", typically tracked over time



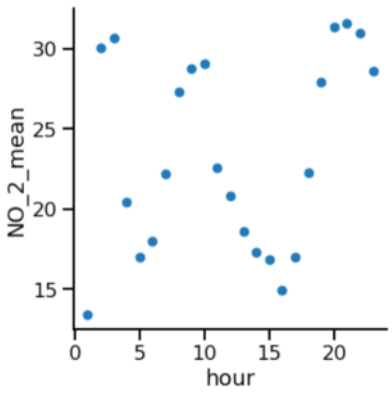
Air pollution data

- Collection stations throughout city
- Air samples of nitrogen dioxide levels

	hour	NO_2_mean
0	1	13.375000
1	2	30.041667
2	3	30.666667
3	4	20.416667
4	5	16.958333

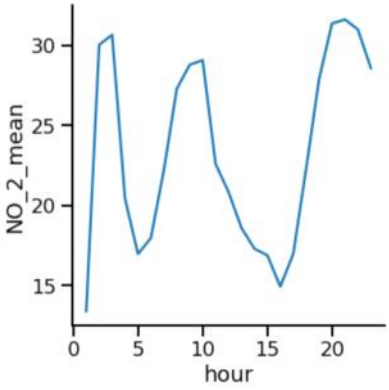
Scatter plot

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

sns.relplot(x="hour", y="NO_2_mean",
            data=air_df_mean,
            kind="scatter")

plt.show()
```



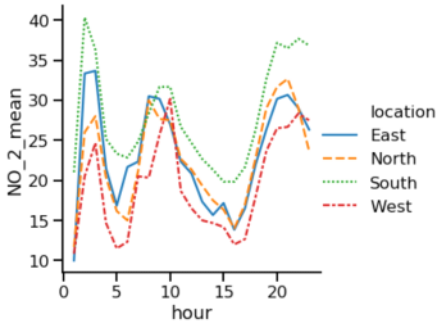
Line plot

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

sns.relplot(x="hour", y="NO_2_mean",
            data=air_df_mean,
            kind="line")

plt.show()
```



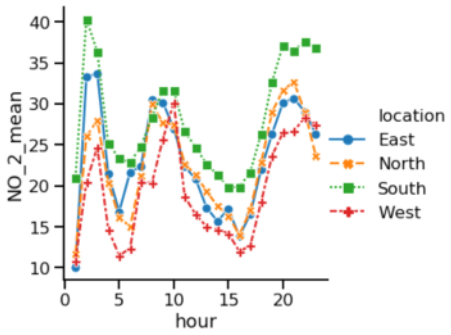
Subgroups by location

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

sns.relplot(x="hour", y="NO_2_mean",
            data=air_df_loc_mean,
            kind="line",
            style="location",
            hue="location")

plt.show()
```



Adding markers

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

sns.relplot(x="hour", y="NO_2_mean",
            data=air_df_loc_mean,
            kind="line",
            style="location",
            hue="location",
            markers=True)
plt.show()
```



The marker will vary based on the subgroup you've set using the "style" parameter.

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

Multiple observations per x-value

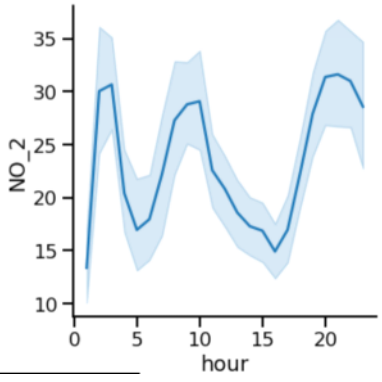
	hour	NO_2	station	location
0	1	15.0	28079004	South
1	1	33.0	28079008	South
2	1	11.0	28079011	South
3	1	12.0	28079016	South
4	1	23.0	28079017	South

Notice that Seaborn will automatically calculate a confidence interval for the mean, displayed by the shaded region.

Multiple observations per x-value

Line plot

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

sns.relplot(x="hour", y="NO_2",
            data=air_df,
            kind="line")

plt.show()
```

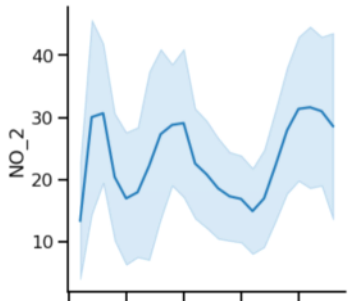


Instead of visualizing a confidence interval, we may want to see how varied the

measurements of nitrogen dioxide are across the different collection stations at a given point in time.

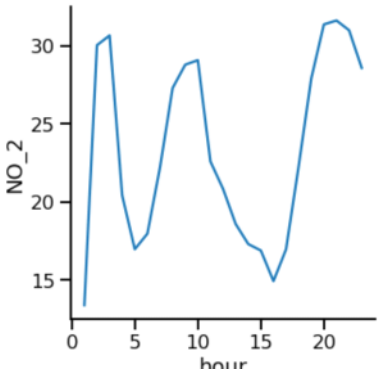
Replacing confidence interval with standard deviation

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

sns.relplot(x="hour", y="NO_2",
            data=air_df,
            kind="line",
            ci="sd")

plt.show()
```



Turning off confidence interval

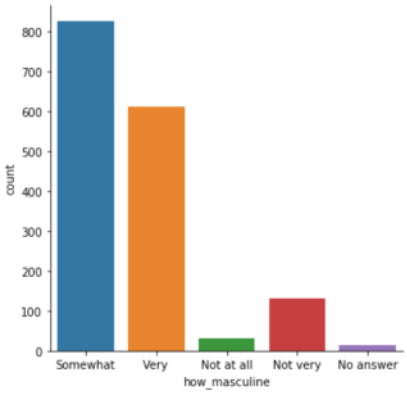
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.relplot(x="hour", y="NO_2",
            data=air_df,
            kind="line",
            ci=None)

plt.show()
```



Visualizing a Categorical and a Quantitative Variable

Categorical plots

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

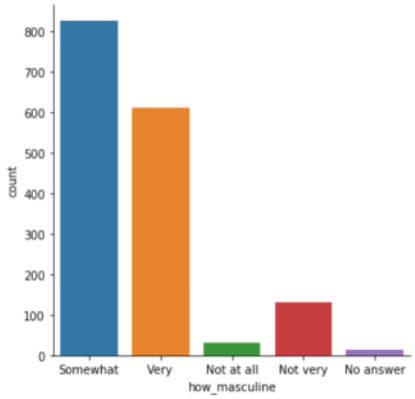


catplot()

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

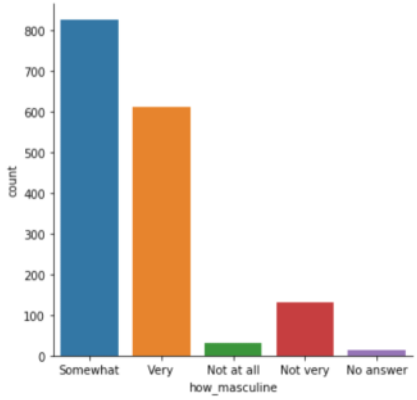
countplot() vs. catplot()

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

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

plt.show()
```



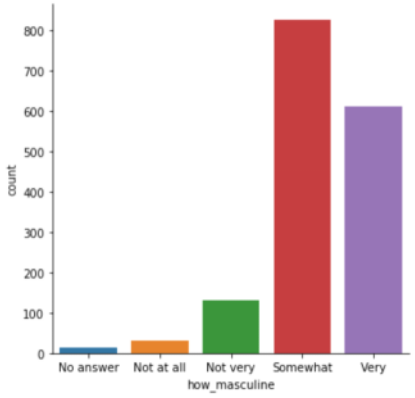
countplot() vs. catplot()

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

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

plt.show()
```



Changing the order

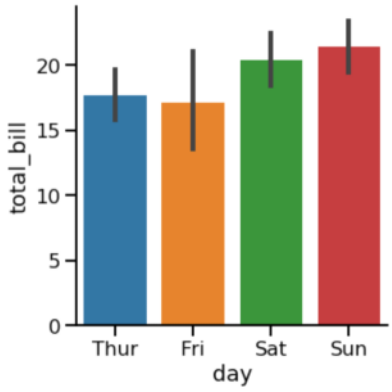
```
import matplotlib.pyplot as plt
import seaborn as sns
category_order = ["No answer",
                  "Not at all",
                  "Not very",
                  "Somewhat",
                  "Very"]
sns.catplot(x="how_masculine",
            data=masculinity_data,
            kind="count",
 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



Bar plots

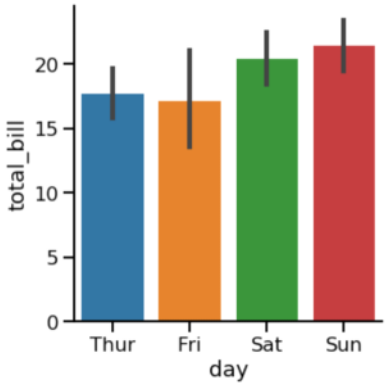
Displays mean of quantitative variable per category

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

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

plt.show()
```



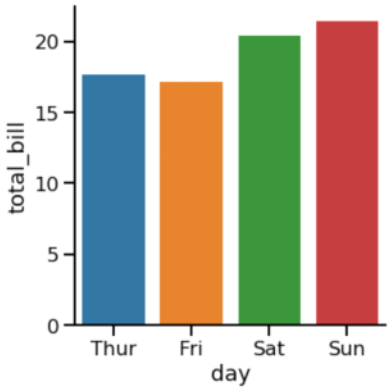
Confidence intervals

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



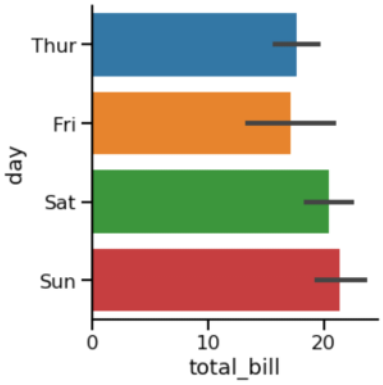
Turning off confidence intervals

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

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



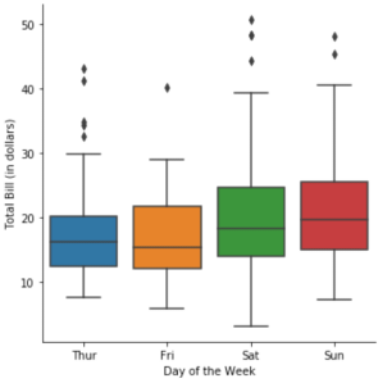
Changing the orientation

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

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



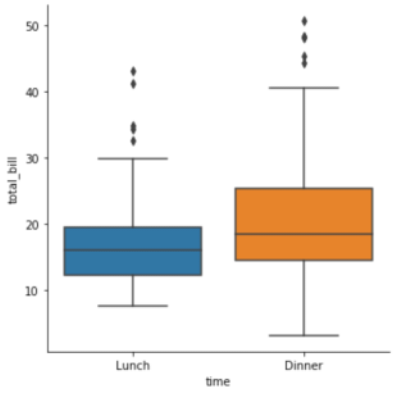
What is a box plot?

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



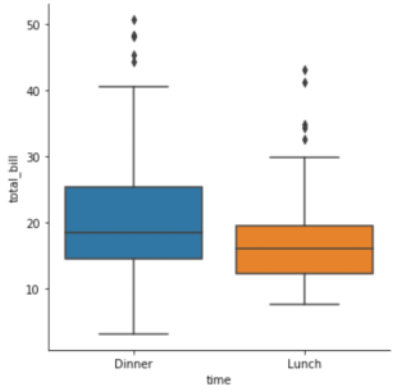
How to create a box plot

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

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

plt.show()
```



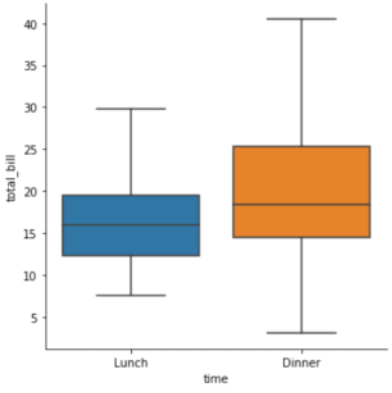
Change the order of categories

```
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()
```



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()
```

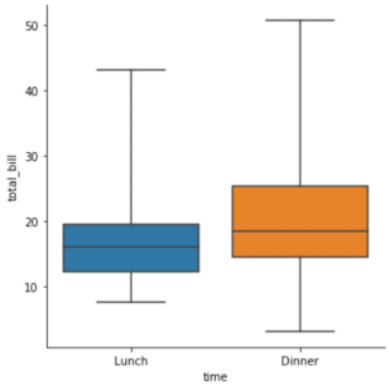


Changing the whiskers using `whis`

- 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]`

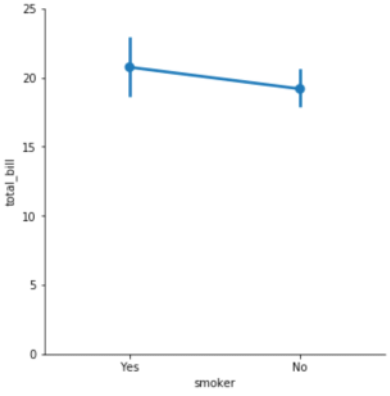
Changing the whiskers using `whis`

```
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()
```

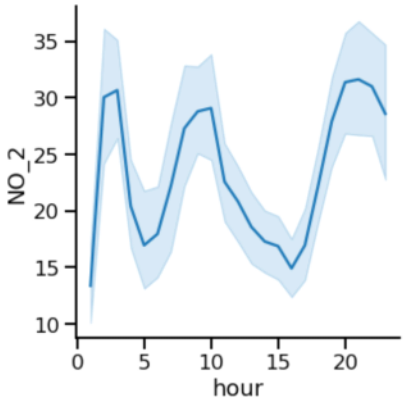


What are point plots?

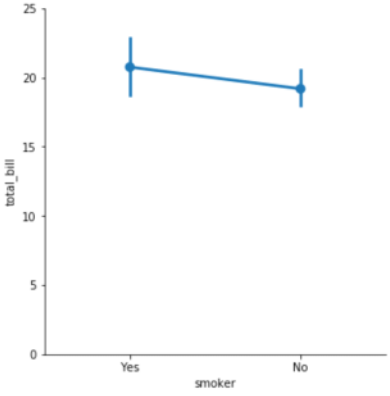
- Points show mean of quantitative variable
- Vertical lines show 95% confidence intervals



Line plot: average level of nitrogen dioxide over time



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



Point plots vs. line plots

Both show:

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

Differences:

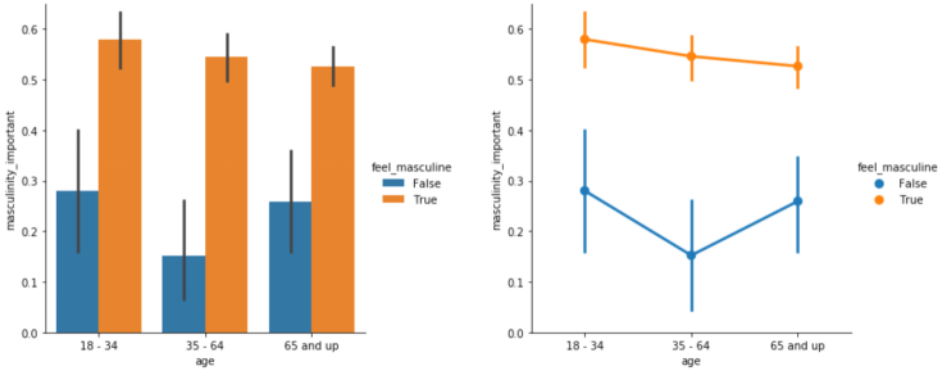
- Line plot has **quantitative** variable (usually time) on x-axis
- Point plot has **categorical** variable on x-axis

Point plots vs. bar plots

Both show:

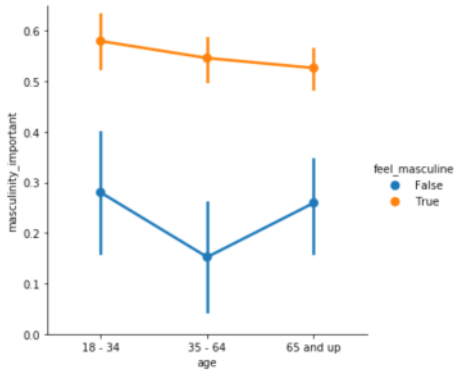
- Mean of quantitative variable
- 95% confidence intervals for the mean

Point plots vs. bar plots



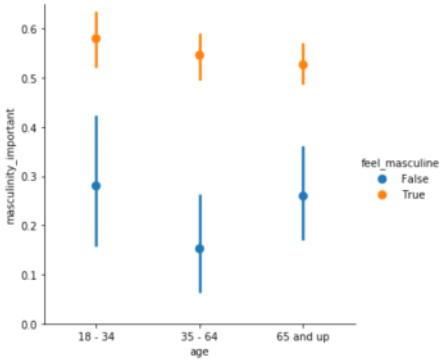
Creating a point plot

```
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()
```



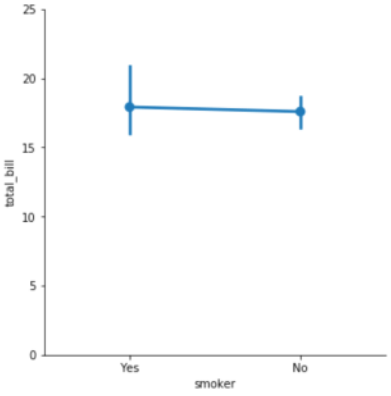
Disconnecting the points

```
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()
```



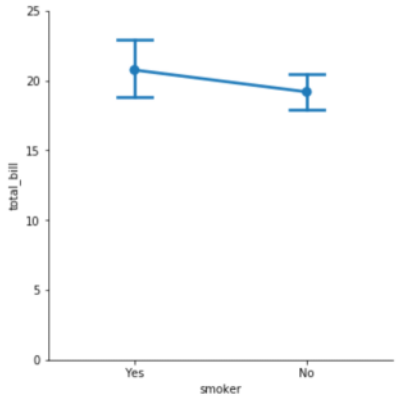
Displaying the median

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

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

plt.show()
```



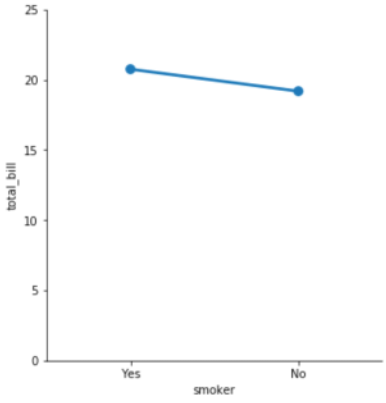
Customizing the confidence intervals

```
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()
```



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()
```



Customizing Seaborn Plots

Why customize?

Reasons to change style:

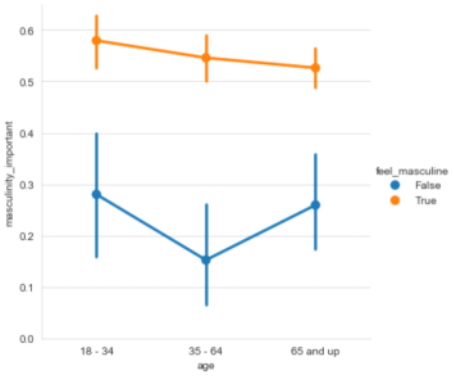
- Personal preference
- Improve readability
- Guide interpretation

Changing the figure style

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

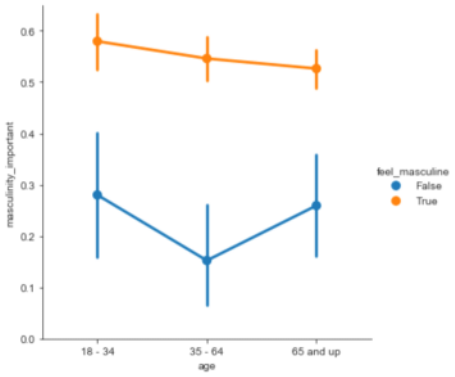
Figure style: "whitegrid"

```
sns.set_style("whitegrid")

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

plt.show()
```



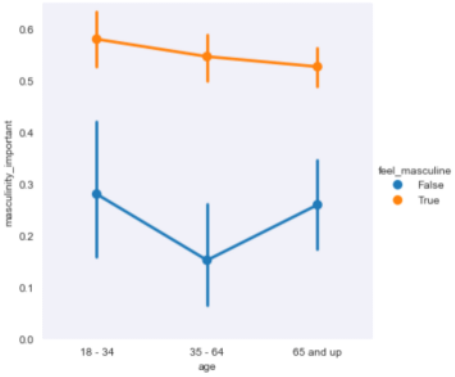
Other styles

```
sns.set_style("ticks")

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

plt.show()
```



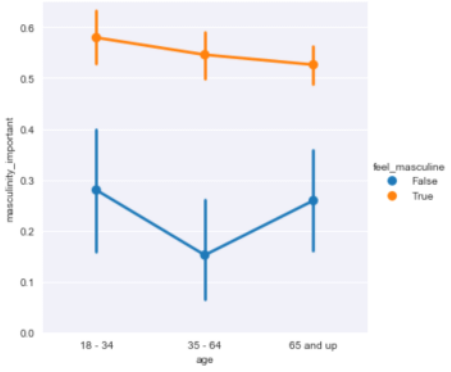
Other styles

```
sns.set_style("dark")

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

plt.show()
```



Other styles

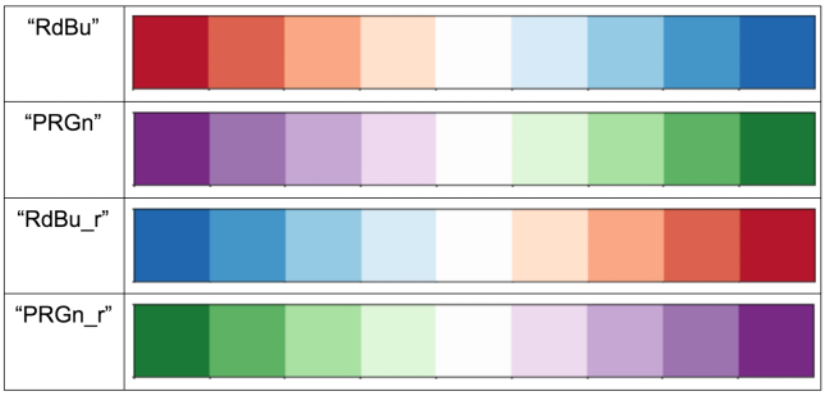
```
sns.set_style("darkgrid")

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

plt.show()
```



Changing the palette

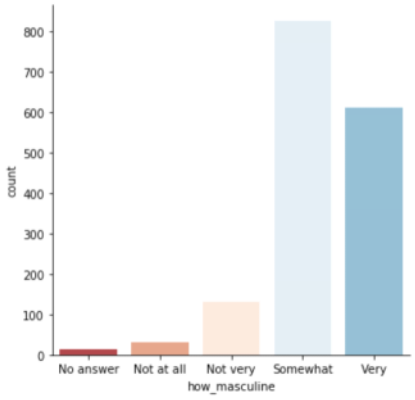
- Figure "palette" changes the color of the main elements of the plot
- `sns.set_palette()`
- Use preset palettes or create a custom palette

Diverging palettes

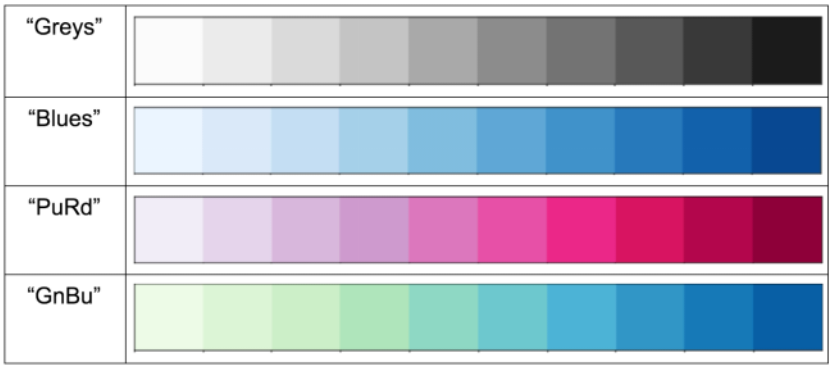


Example (diverging palette)

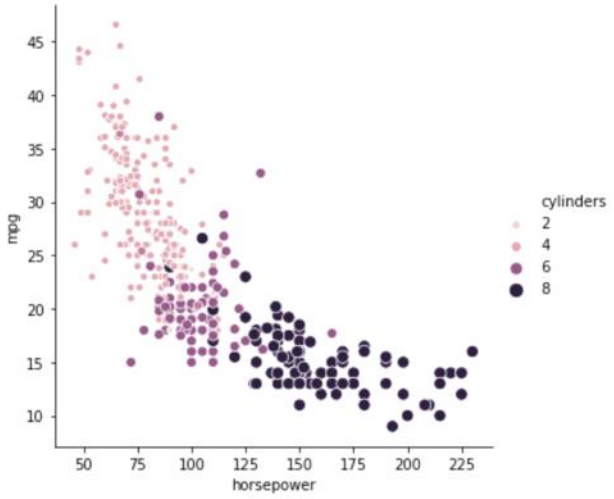
```
sns.set_palette("RdBu")

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()
```



Sequential palettes



Sequential palette example



Custom palettes

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

sns.set_palette(custom_palette)
```

Custom palettes

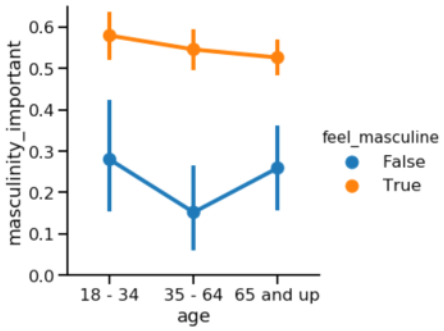
```
custom_palette = ['#FBB4AE', '#B3CDE3', '#CCEBC5',
                  '#DECBE4', '#FED9A6', '#FFFFCC',
                  '#E5D8BD', '#FDDAEC', '#F2F2F2']

sns.set_palette(custom_palette)
```

Changing the scale

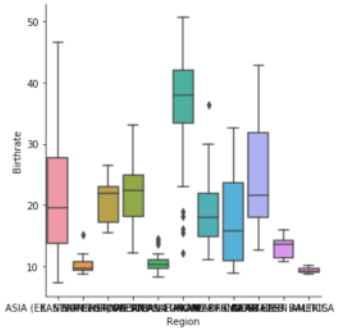
- Figure "context" changes the scale of the plot elements and labels
- `sns.set_context()`
- Smallest to largest: "paper", "notebook", "talk", "poster"

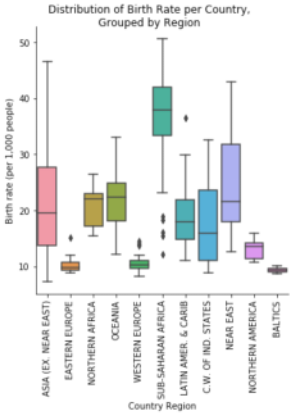
Larger context: "talk"

```
sns.set_context("talk")

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

plt.show()
```



Creating informative visualizations





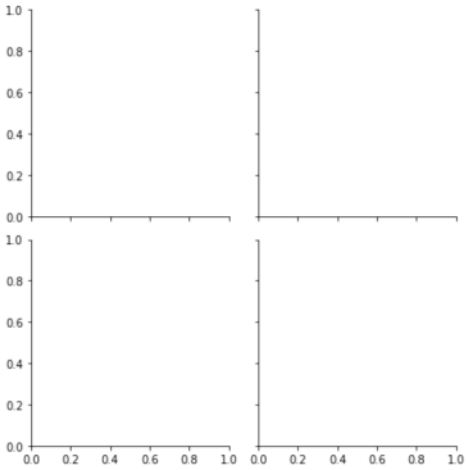
FacetGrid vs. AxesSubplot objects

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

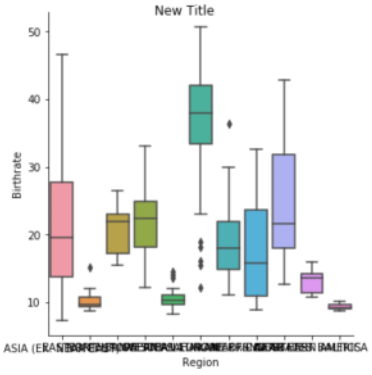


FacetGrid vs. AxesSubplot objects

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

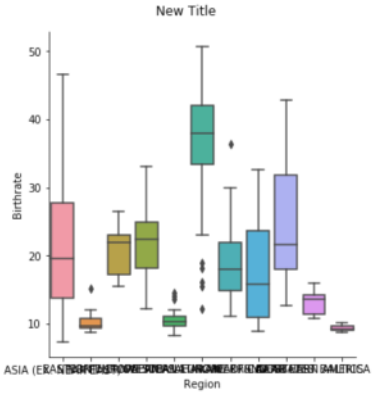
Adding a title to FacetGrid

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



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()
```



Adding a title to AxesSubplot

FacetGrid

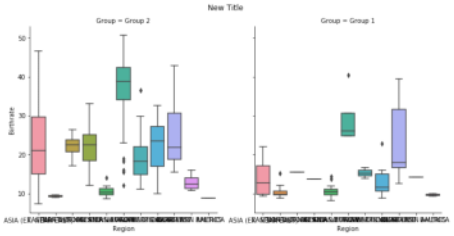
```
g = sns.catplot(x="Region",
                y="Birthrate",
                data=gdp_data,
                kind="box")

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)
```

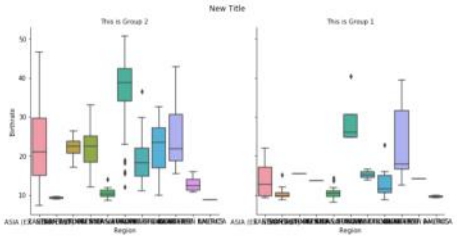
Titles for subplots

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

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



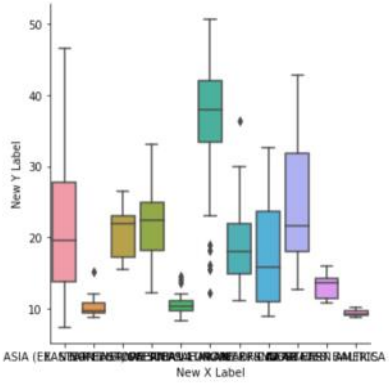
Titles for subplots

```
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}")
```



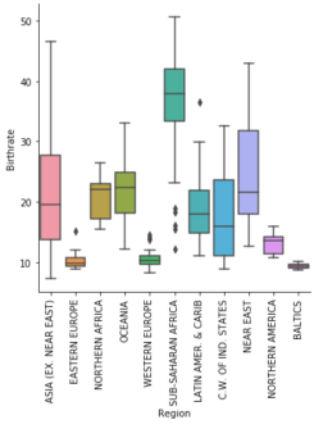
Adding axis labels

```
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()
```



Getting started

To import Seaborn:

```
import seaborn as sns
```

To import Matplotlib:

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

To show a plot:

```
plt.show()
```

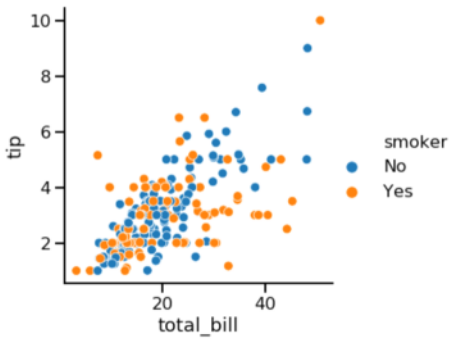
Relational plots

- 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")
```

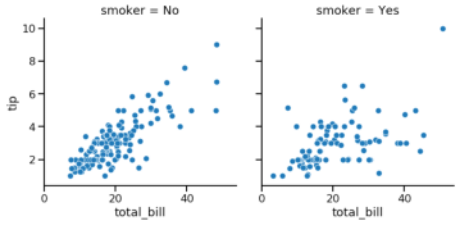
Adding a third variable (hue)

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)
```