Interactive Visualization With Plotly - Seaborn - 1

One should look for what is and not what he thinks should be. (Albert Einstein)

Seaborn: Topic introduction

In this part of the course, we will cover the following concepts:

- Introduce seaborn package and it’s capabilities
- Organize and visualize data with seaborn

Warm-up

- Have you ever wondered, “how do eggs get their shape?”
- Take a look at this Science Mag website with cool interactive visualizations ***here***
  - Did you find the visualizations effective? What was the most interesting?

Module completion checklist

Objective	Complete
Introduce Seaborn plotting library and create univariate plots	
Plot bivariate charts, heatmaps and format plots in Seaborn	

What is `seaborn`?

- **seaborn** is a python data visualization library based on ***matplotlib***
- It provides a high-level interface for drawing attractive and informative statistical graphics
- It integrates closely with pandas dataframes
- It has a variety of sample datasets available for experimenting with plots
- It uses the `rcParams` structure to control graph elements, like matplotlib
- ***Click here*** to learn more about `seaborn`

What can you do with `seaborn`?

- Like matplotlib, `seaborn` has a beautiful gallery of different types of plots
- Check out the `seaborn` gallery ***here***
- Examples include:
  - Univariate plots: Histograms, Box plots, Bar charts
  - Bivariate plots: Scatter plots, Line plots, Residual plots
  - Heatmaps

Introduce the penguins dataset

- This is an `seaborn` inbuilt dataset
- The data was collected and made available by Dr. Kristen Gorman and the Palmer Station, Antarctica LTER, a member of the Long Term Ecological Research Network

Data description of penguin dataset

- Let’s look at the detailed data description of penguin dataset
  - **species**: a factor denoting penguin species (Adélie, Chinstrap and Gentoo)
  - **island**: a factor denoting island in Palmer Archipelago, Antarctica (Biscoe, Dream or Torgersen)
  - **bill_length_mm**: a number denoting bill length (millimeters)
  - **bill_depth_mm**: a number denoting bill depth (millimeters)
  - **flipper_length_mm**: an integer denoting flipper length (millimeters)
  - **body_mass_g**: an integer denoting body mass (grams)
  - **sex**: a factor denoting penguin sex (female, male)

Loading the dataset and libraries

- Let’s import required libraries

```
# import the libraries
import seaborn as sns
from matplotlib import pyplot as plt
```

- Load the penguin dataset from `seaborn`

```
# Load the dataset
penguins = sns.load_dataset("penguins")

# Top 5 entries of dataset
penguins.head()
```

```
  species     island  bill_length_mm  ...  flipper_length_mm  body_mass_g     sex
0  Adelie  Torgersen            39.1  ...              181.0       3750.0    Male
1  Adelie  Torgersen            39.5  ...              186.0       3800.0  Female
2  Adelie  Torgersen            40.3  ...              195.0       3250.0  Female
3  Adelie  Torgersen             NaN  ...                NaN          NaN     NaN
4  Adelie  Torgersen            36.7  ...              193.0       3450.0  Female

[5 rows x 7 columns]
```

Data preprocessing

- Let’s check for NA’s in penguin dataset and remove them

```
# Check for null values
penguins.isna().sum()
```

```
species               0
island                0
bill_length_mm        2
bill_depth_mm         2
flipper_length_mm     2
body_mass_g           2
sex                  11
dtype: int64
```

- Drop the NA values

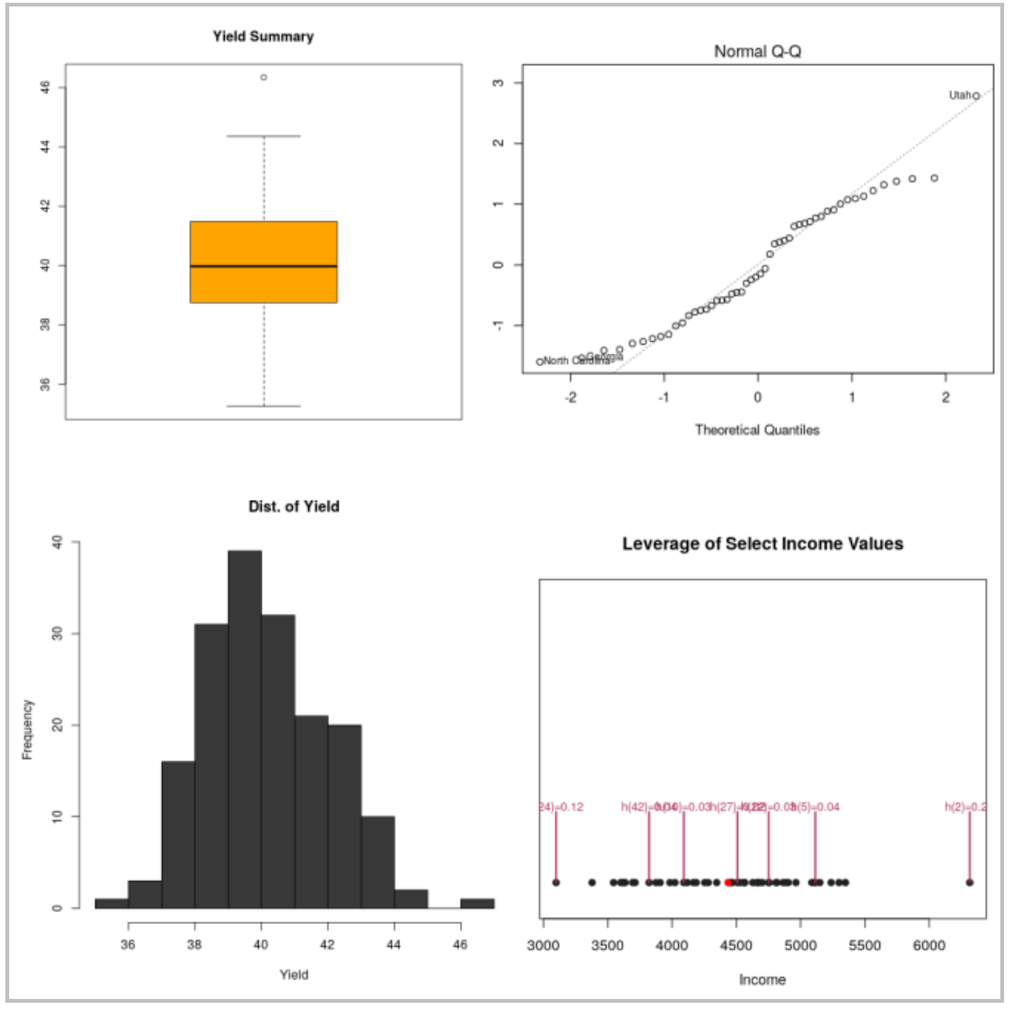
```
# Drop NA's and reassign clean data
penguins = penguins.dropna()
```

- Now our data is ready, let’s create plots out of it using `seaborn`

Univariate plots

- Univariate plots are used to visualize distribution of a **single variable**
- They are used primarily in the initial stages of EDA to learn more about individual variables in our data
- They are also used in combination with other univariate plots to compare data distributions of different variables
- Univariate plots include the following popular graphs: boxplot, histogram, density curve, dot plot, QQ plot, and bar plot

- Different univariate plots



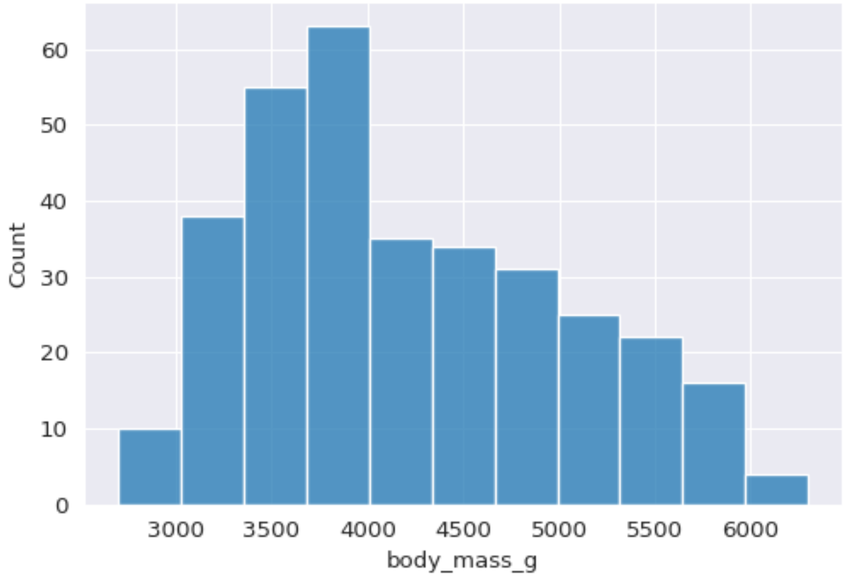
Univariate plots: histogram

- A histogram represents the **distribution of numerical data**
  - The height of each bar has been calculated as the number of observations in that range
  - `histplot()` produces a basic histogram of any *numeric* variable

Plot a histogram

- Let’s create a histogram to visualize the penguin data
- Here we will look at the distribution of the body mass in grams of penguins

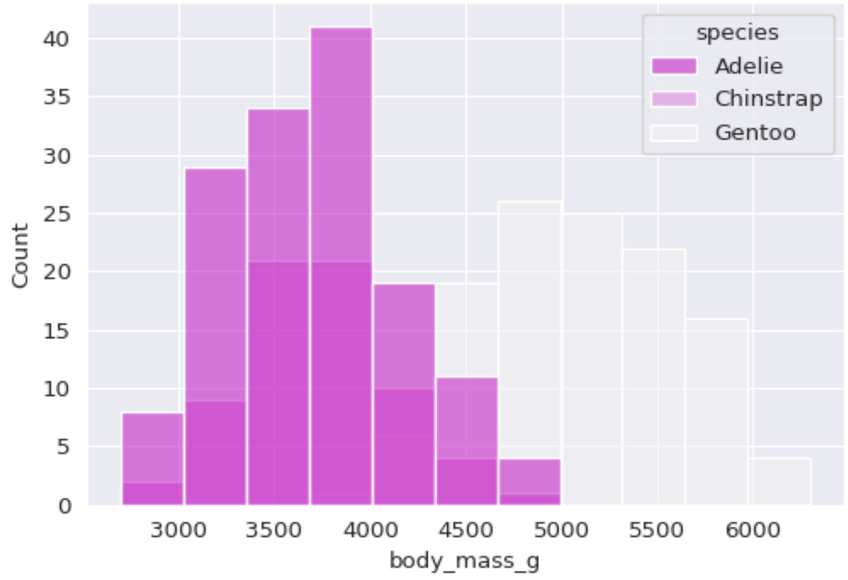
```
plt.figure(figsize=(6,4))       #<- set the figure size

# Create a simple histogram
sns.histplot(data = penguins,      #<- dataset
        x = "body_mass_g")         #<- x_axis variable
```



Plot a layered histogram

- We have a visual of body mass of all penguins, but what if we want to compare the distributions by species?
- We can change this simple histogram into a layered histogram by just adding two parameters, `hue` and `palette`
- `hue` describes the grouping mechanism for our histogram, and `palette` tells Seaborn what colors to use
- You can see the list of possible palettes ***here***

```
plt.figure(figsize=(6,4))      #<- set the figure
size

sns.histplot(data = penguins,      #<- set data
      x = "body_mass_g",           #<- set x
variable
      hue = "species",             #<- grouping
parameter
      palette = "light:m_r")       #<- set color
```



Univariate plots: box plot

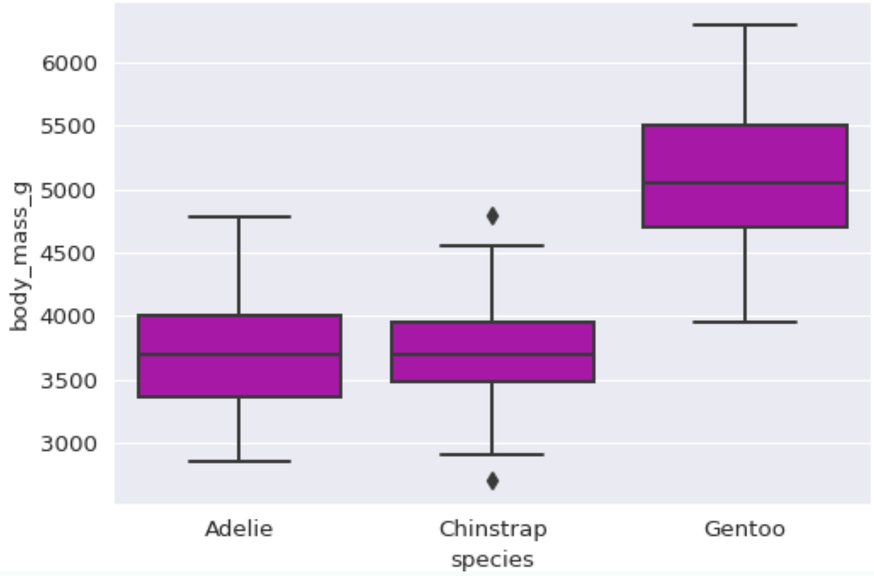
- Histograms aren't the only type of plot used to illustrate the distribution of a variable. We can also use box plots.
- Instead of showing the shape of the distribution, **box plots show the quartiles of your data** and unlike a histogram, **box plots show outliers in our data**
- We will use the same `penguins` dataset and `body_mass_g` variable:

```
plt.figure(figsize=(5,3))      #<- set the figure size

# Create a simple boxplot
sns.boxplot(data = penguins,       #<- set data
        y = "body_mass_g",         #<- set y variable
        palette = ["m"])           #<- set color
```

Plot a box plot

- Like a histogram we can compare distributions of different variables on a single box plot by adding a single parameter `x` and set the appropriate color palette using `palette` parameter

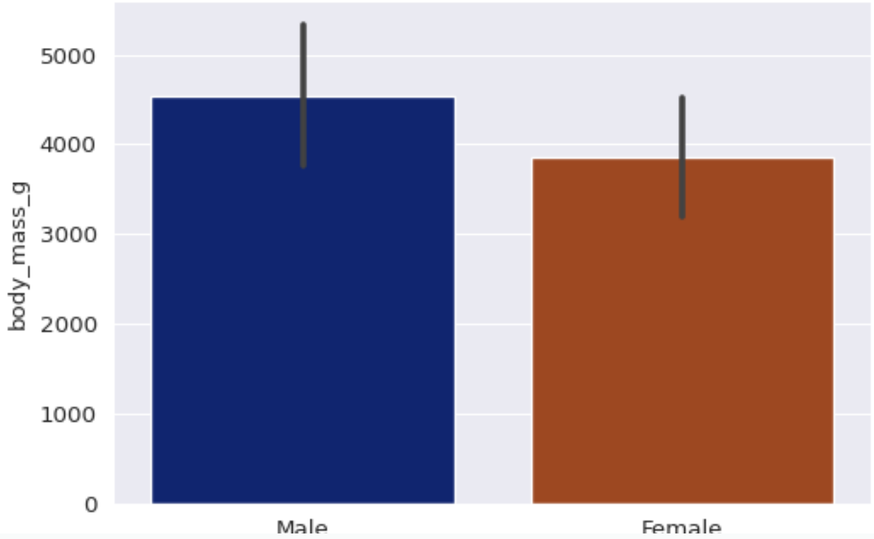
```
plt.figure(figsize=(6,4))     #<- set the figure size
sns.boxplot(data = penguins,     #<- set data
        x = "species",           #<- set x variable
        y = "body_mass_g",       #<- set y variable
        palette = ["m"])         #<- set color
```



Univariate plots: bar plot

- The final type of univariate plot we are going to discuss is a **bar plot**
- The bar plot is used to **compare categories within a dataset**

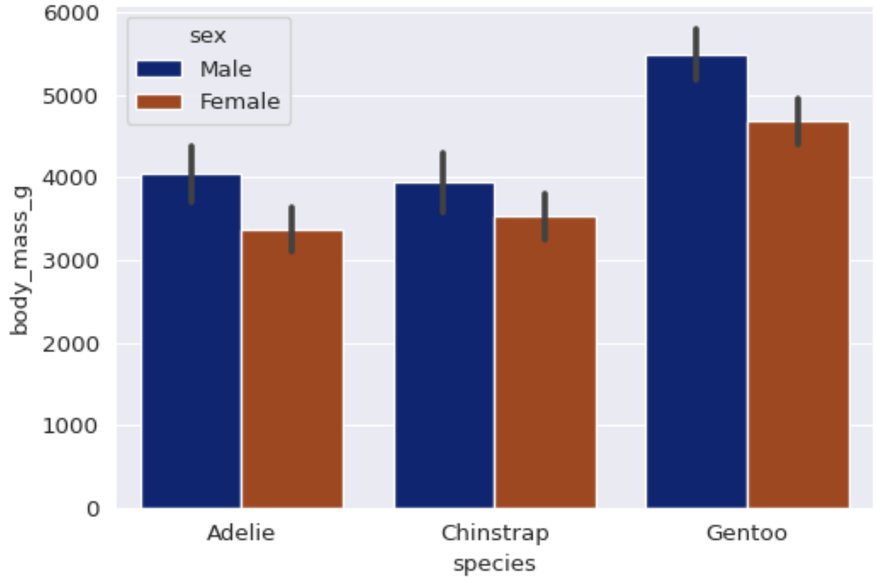
```
plt.figure(figsize=(6,4))        #<- set the figure size

# Create a barplot
sns.barplot(data = penguins,      #<- set the data
        x = "sex",                #<- set x variable
        y = "body_mass_g",        #<- set y variable
        ci = "sd",
        palette = "dark")         #<- set color
```



Plot a bar plot

- Like box plots and histograms, bar plots can compare groups by adding an extra parameter, `hue`

```
plt.figure(figsize=(6,4))          #<- set the figure size
sns.barplot(data = penguins,       #<- set the data
        x = "species",             #<- set x variable
        y = "body_mass_g",         #<- set y variable
        hue = "sex",ci="sd",       #<- set grouping variable
        palette = "dark")          #<- set color
```



Module completion checklist

Objective	Complete
Introduce Seaborn plotting library and create univariate plots	✔
Plot bivariate charts, heatmaps and format plots in Seaborn	

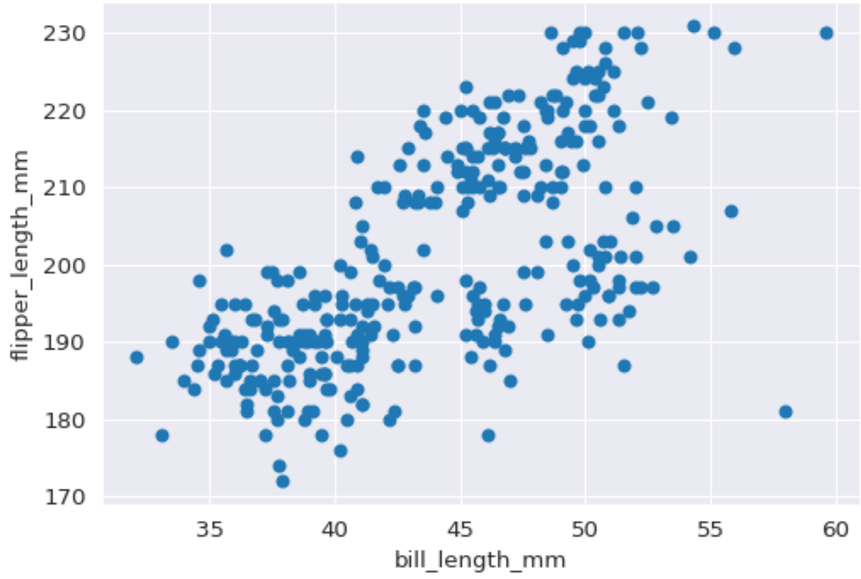
Bivariate plots

- Bivariate plots are used to visualize data distribution and relationships between **two variables**
- They are used to a great extent throughout different stages of EDA to learn more about how **one variable relates to another**
- They are also used in combination with other bivariate plots to compare relationships between **different pairs of variables**
- Bivariate plots include `scatterplots` and `line graphs`

Bivariate plots: scatter plot

- **Scatter plots** are an excellent way to see the relationship between two variables

```
plt.figure(figsize=(6,4))          #<- set the figure size

# Create a scatterplot
sns.scatterplot(data = penguins,        #<- set the data
        x = "bill_length_mm",           #<- set x variable
        y = "flipper_length_mm",         #<- set y variable
        linewidth = 0)                  #<- set line width
```



Bivariate plots: line plot

- After we see a linear relationship between two variables, we may want to plot that relationship, that is where a line plot comes in

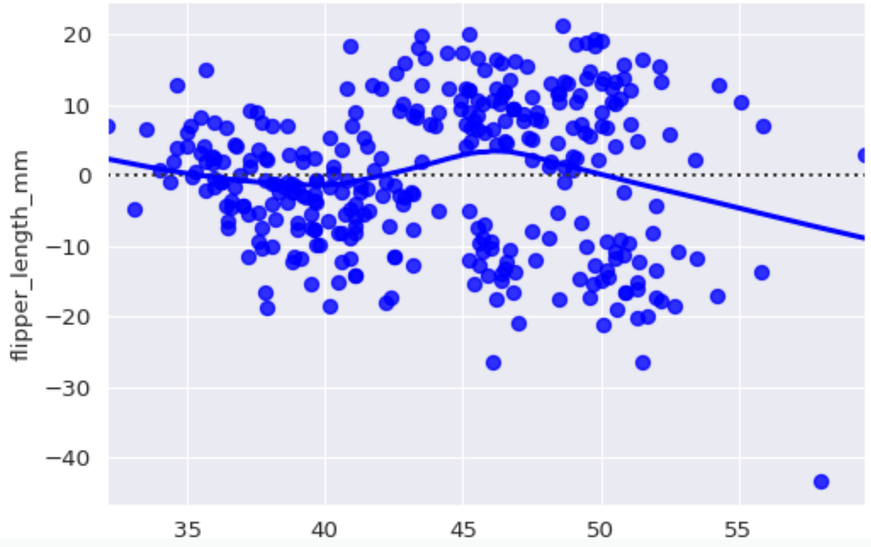
```
plt.figure(figsize=(5,4))       #<- set the figure size

sns.lmplot(data = penguins,           #<- set the data
        x = "bill_length_mm",         #<- set x variable
        y = "flipper_length_mm",      #<- set y variable
        fit_reg = True)
```

```
<seaborn.axisgrid.FacetGrid object at 0x7f3c557d5bd0>
```

Bivariate plots: residual plot

- One way we can decide if there is a **linear relationship** between those two variables is to look at a **plot of the residuals**
- Seaborn allows you to plot those residuals with one simple line of code

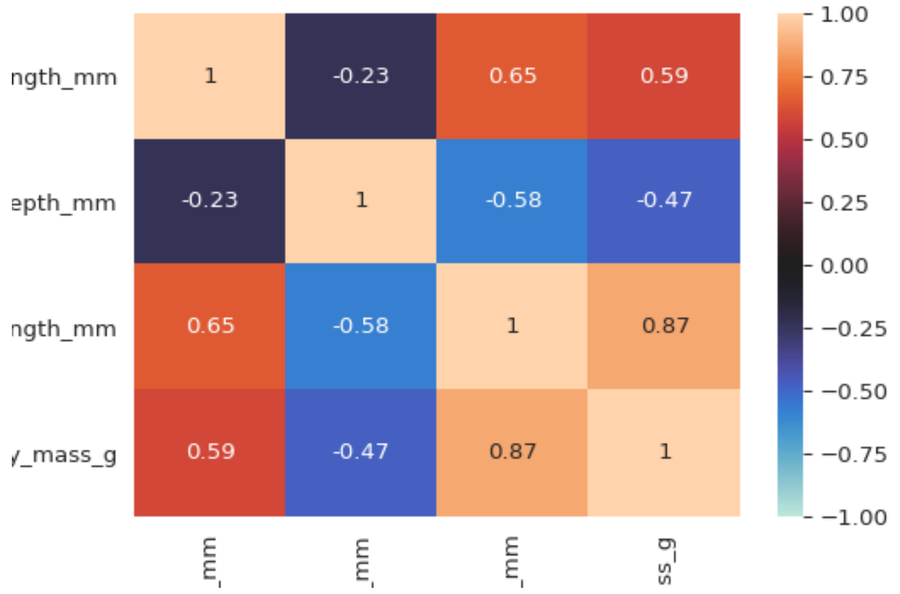
```
plt.figure(figsize=(6,4))           #<- set the figure size
sns.residplot(data = penguins,
        x = 'bill_length_mm',               #<- set x variable
        y = 'flipper_length_mm',            #<- set y variable
        lowess = True,
        color="b")
```



Multivariate plots: Heatmaps

- One way we can check to see what variables we might want to use in a model is to check the correlations of those variables
- An easy way to see those correlations is a **heatmap**
- Which variables are correlated with each other?

```
plt.figure(figsize=(6,4))                #<- set the figure
size
sns.heatmap(penguins.corr(),             #<- set the data to
find out the correlation
        vmin = -1,
        vmax = 1,
        center = 0,
        annot = True)
```



Format plots in seaborn

- Our Seaborn plots come out readable if variables are named well
- What if we haven't named them well or want to change the names on the axes and title of our plot?
- `seaborn` allows easy formatting of things like the title or the axes labels using the `set` function

Format plots in seaborn (cont'd)

```
plt.figure(figsize=(5,4))              #<- set the figure size
g = sns.lmplot(data = penguins,
        x = "bill_length_mm",                    #<- set x variable
        y = "flipper_length_mm",                 #<- set x variable
        height = 4,                              #<- set the height
        fit_reg = True)
g.set(title = 'Bill Length versus Flipper Length',           #<- set the title
        xlabel = 'Bill Length (mm)',                         #<- set label for x variable
        ylabel = 'Flipper Length (mm)')                      #<- set label for y variable
```

```
<seaborn.axisgrid.FacetGrid object at 0x7f3c4bb771d0>
```

Knowledge check

Link: ***Click here to complete the knowledge check***

Exercise

Module completion checklist

Objective	Complete
Introduce Seaborn plotting library and describe univariate plots in Seaborn	✔
Describe bivariate plots and heatmaps and format plots in Seaborn	✔

Seaborn: Topic summary

In this part of the course, we have covered:

- Introduce seaborn package and it’s capabilities
- Organize and visualize data with seaborn

Congratulations on completing this module!