

seaborn_tutorial

January 7, 2025

1 Introduction to Seaborn

Seaborn is a Python visualisation library built on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. This tutorial will guide you through some of the fundamental plotting functions used in exploratory data analysis.

For more information you can check out the Seaborn documentation: <https://seaborn.pydata.org/index.html>

You will need to make sure that Seaborn is installed before importing. Here we follow the convention of importing Seaborn as sns.

```
[1]: import seaborn as sns
```

1.1 Load the dataset

First we need to load a dataset. For this tutorial, we will use Seaborn to load one of the datasets that it makes available for testing and practice. You can see the full set of Seaborn datasets here: <https://github.com/mwaskom/seaborn-data>

Here we'll go with the penguins dataset.

```
[2]: penguins_df = sns.load_dataset("penguins")
```

This gives us a regular Pandas DataFrame.

```
[3]: penguins_df.sample(10)
```

```
[3]:
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	\
64	Adelie	Biscoe	36.4	17.1	184.0	
7	Adelie	Torgersen	39.2	19.6	195.0	
238	Gentoo	Biscoe	46.2	14.5	209.0	
275	Gentoo	Biscoe	45.0	15.4	220.0	
215	Chinstrap	Dream	55.8	19.8	207.0	
213	Chinstrap	Dream	46.8	16.5	189.0	
142	Adelie	Dream	32.1	15.5	188.0	
59	Adelie	Biscoe	37.6	19.1	194.0	
33	Adelie	Dream	40.9	18.9	184.0	
210	Chinstrap	Dream	50.2	18.8	202.0	

	body_mass_g	sex
64	2850.0	Female
7	4675.0	Male
238	4800.0	Female
275	5050.0	Male
215	4000.0	Male
213	3650.0	Female
142	3050.0	Female
59	3750.0	Male
33	3900.0	Male
210	3800.0	Male

```
[4]: penguins_df.describe().T
```

```
[4]:
```

	count	mean	std	min	25%	50% \
bill_length_mm	342.0	43.921930	5.459584	32.1	39.225	44.45
bill_depth_mm	342.0	17.151170	1.974793	13.1	15.600	17.30
flipper_length_mm	342.0	200.915205	14.061714	172.0	190.000	197.00
body_mass_g	342.0	4201.754386	801.954536	2700.0	3550.000	4050.00

	75%	max
bill_length_mm	48.5	59.6
bill_depth_mm	18.7	21.5
flipper_length_mm	213.0	231.0
body_mass_g	4750.0	6300.0

```
[5]: penguins_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
#   Column              Non-Null Count  Dtype
---  -
0   species             344 non-null    object
1   island              344 non-null    object
2   bill_length_mm      342 non-null    float64
3   bill_depth_mm       342 non-null    float64
4   flipper_length_mm   342 non-null    float64
5   body_mass_g         342 non-null    float64
6   sex                 333 non-null    object
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

1.2 Visualise our data

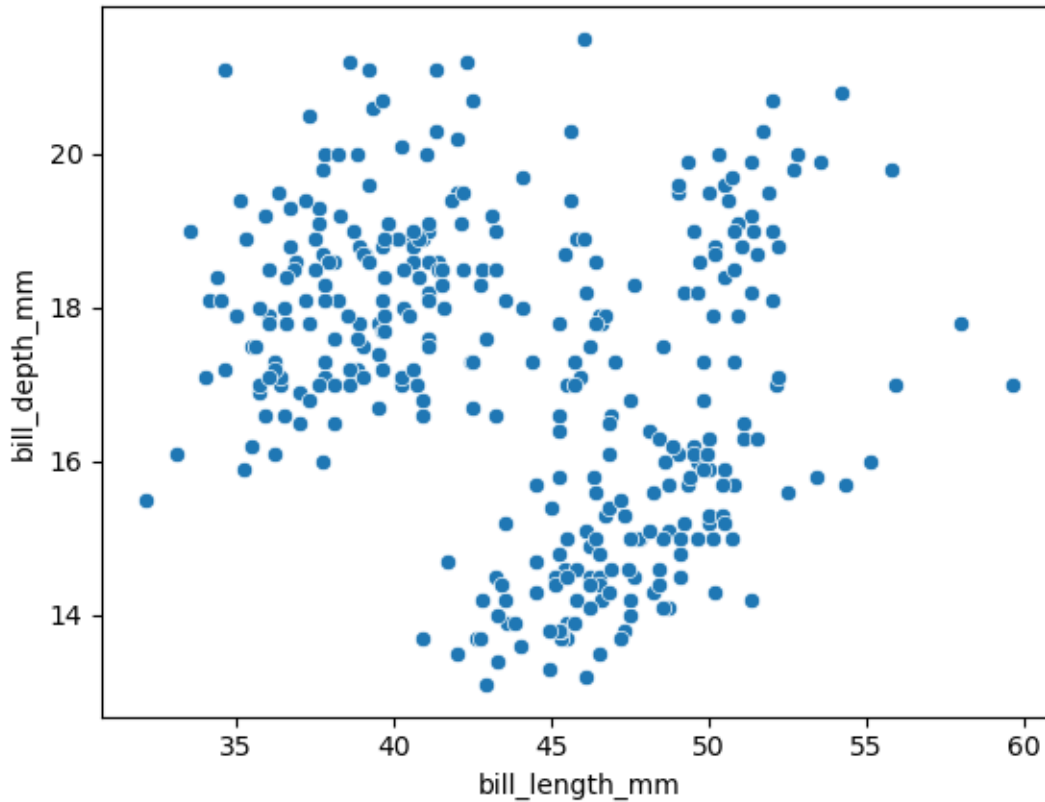
1.2.1 Scatter plots

With a scatter plot, we can plot 2D points. In the example below we plot bill length against bill depth.

For more info on how to customise a scatterplot see: <https://seaborn.pydata.org/generated/seaborn.scatterplot.html>

```
[6]: sns.scatterplot(data=penguins_df,  
                    x="bill_length_mm",  
                    y="bill_depth_mm")
```

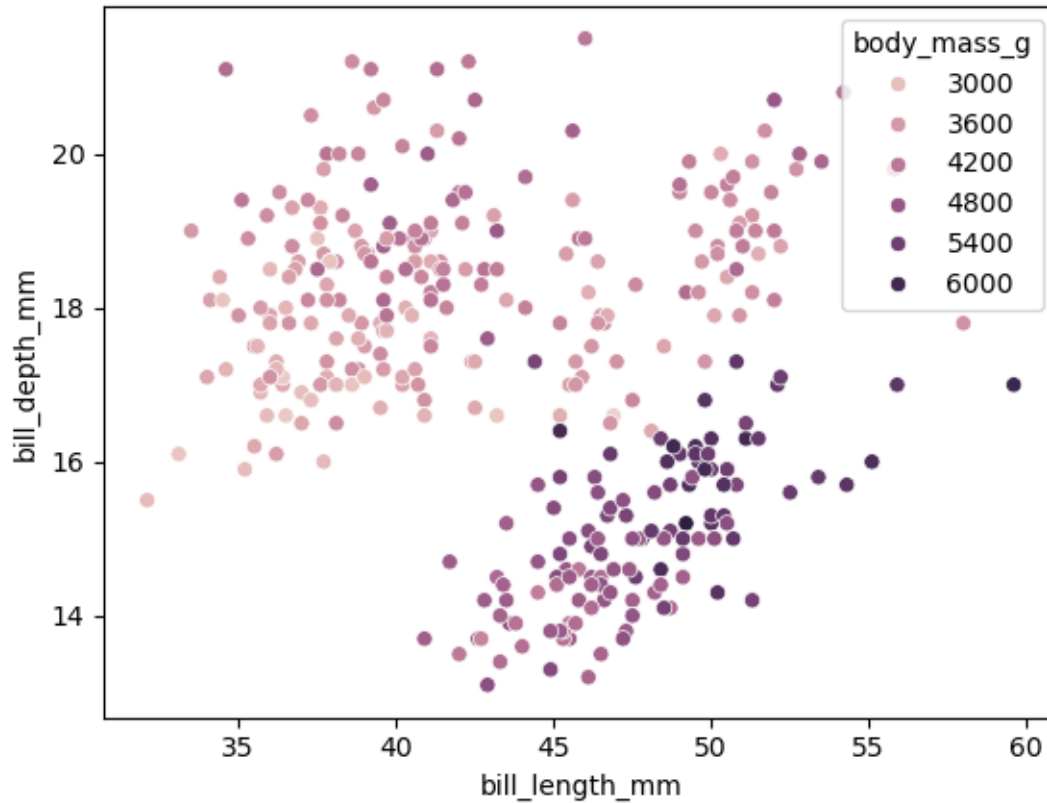
```
[6]: <Axes: xlabel='bill_length_mm', ylabel='bill_depth_mm'>
```



Using the hue argument, we can represent a third numeric or categorical variable. Here we use hue to show the body mass.

```
[7]: sns.scatterplot(data=penguins_df,  
                    x="bill_length_mm",  
                    y="bill_depth_mm",  
                    hue="body_mass_g")
```

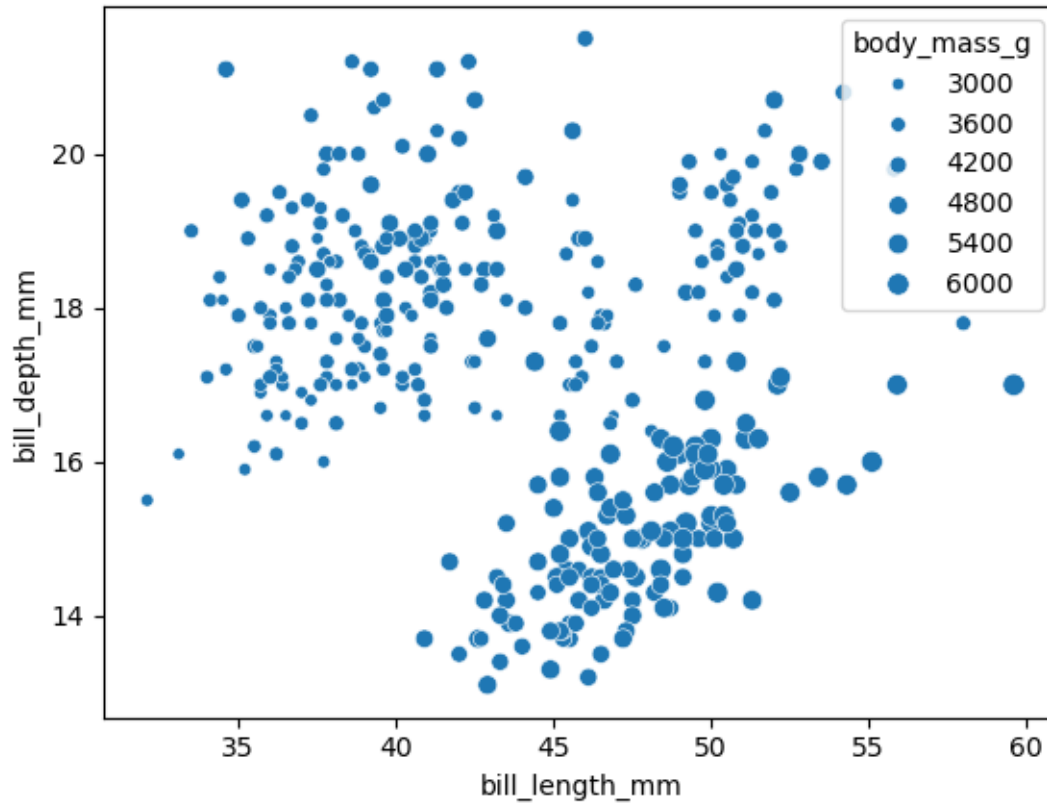
```
[7]: <Axes: xlabel='bill_length_mm', ylabel='bill_depth_mm'>
```



We can also use the size argument to represent another variable. Here we represent body mass with different sized points rather than different coloured points.

```
[8]: sns.scatterplot(data=penguins_df,  
                    x="bill_length_mm",  
                    y="bill_depth_mm",  
                    size="body_mass_g")
```

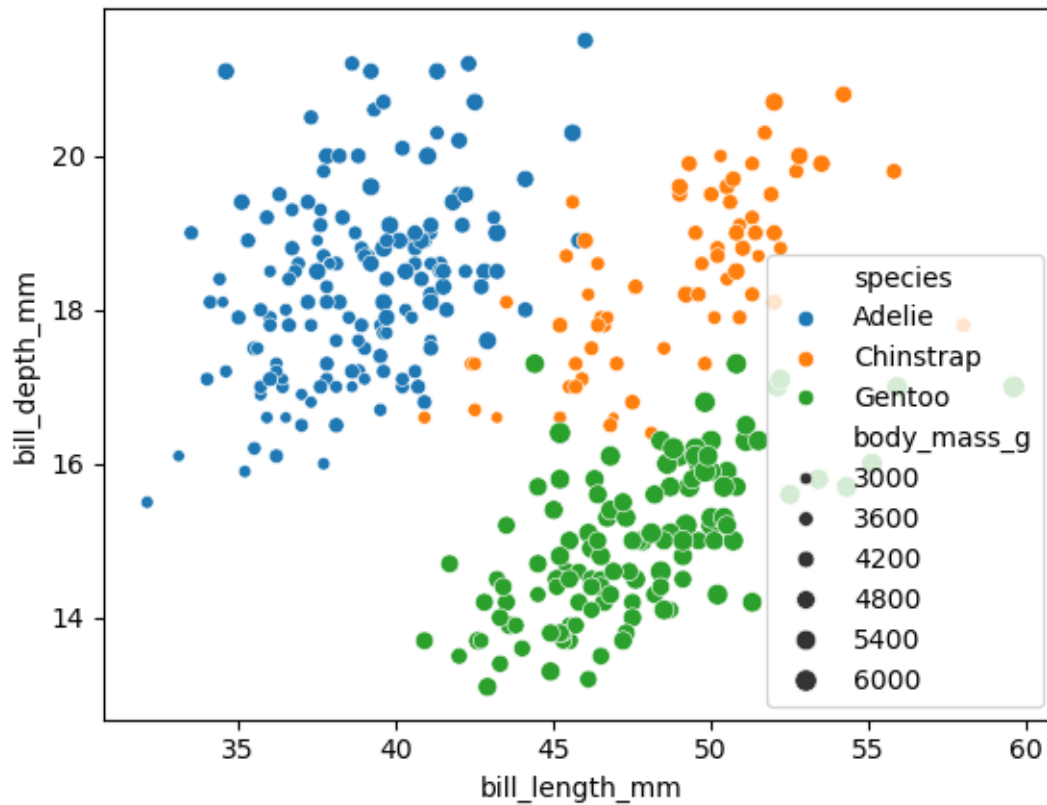
```
[8]: <Axes: xlabel='bill_length_mm', ylabel='bill_depth_mm'>
```



We can use both hue and size. In the example below we show four different variables.

```
[9]: sns.scatterplot(data=penguins_df,  
                    x="bill_length_mm",  
                    y="bill_depth_mm",  
                    hue="species",  
                    size="body_mass_g")
```

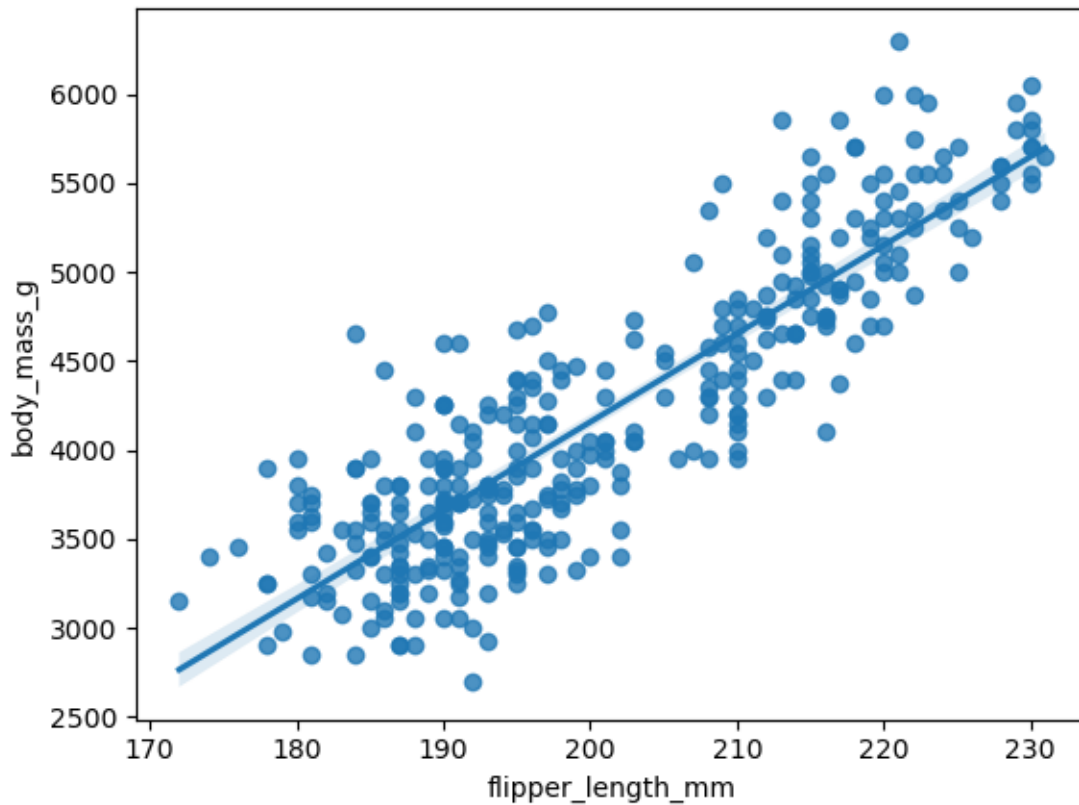
```
[9]: <Axes: xlabel='bill_length_mm', ylabel='bill_depth_mm'>
```



Adding a regression line Here we switch to `regplot` in order to show a regression line. See the following for more information on `regplot`: <https://seaborn.pydata.org/generated/seaborn.regplot.html>

```
[10]: sns.regplot(data=penguins_df,
                  x="flipper_length_mm",
                  y="body_mass_g")
```

```
[10]: <Axes: xlabel='flipper_length_mm', ylabel='body_mass_g'>
```



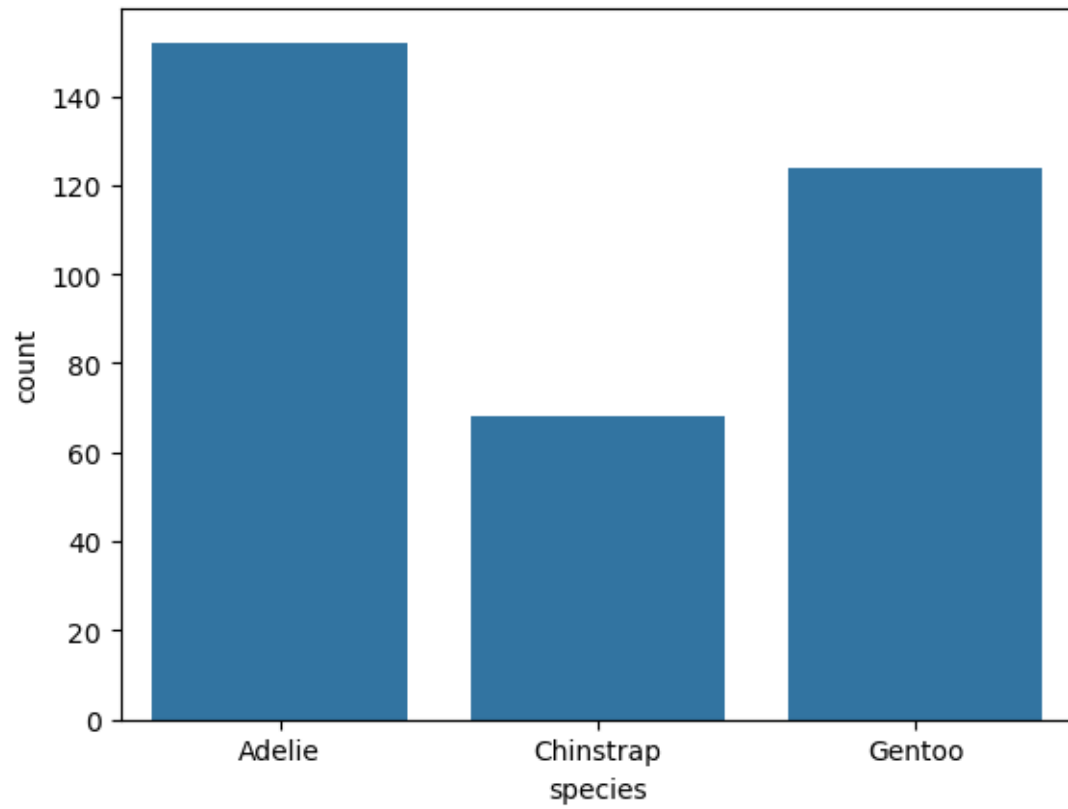
1.2.2 Count plots

Count plots “show the counts of observations in each categorical bin using bars” - <https://seaborn.pydata.org/generated/seaborn.countplot.html>.

Here we show the counts of penguins of different species.

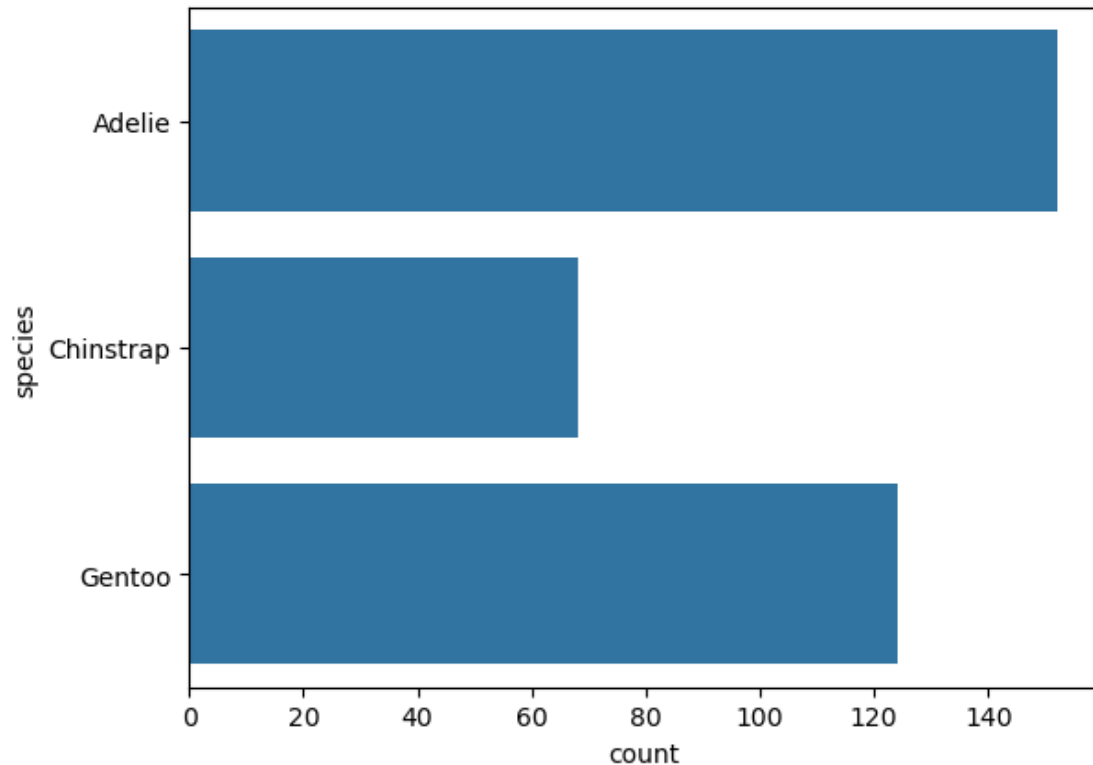
```
[11]: sns.countplot(data=penguins_df,  
                    x="species")
```

```
[11]: <Axes: xlabel='species', ylabel='count'>
```



```
[12]: sns.countplot(data=penguins_df,  
                    y="species")
```

```
[12]: <Axes: xlabel='count', ylabel='species'>
```

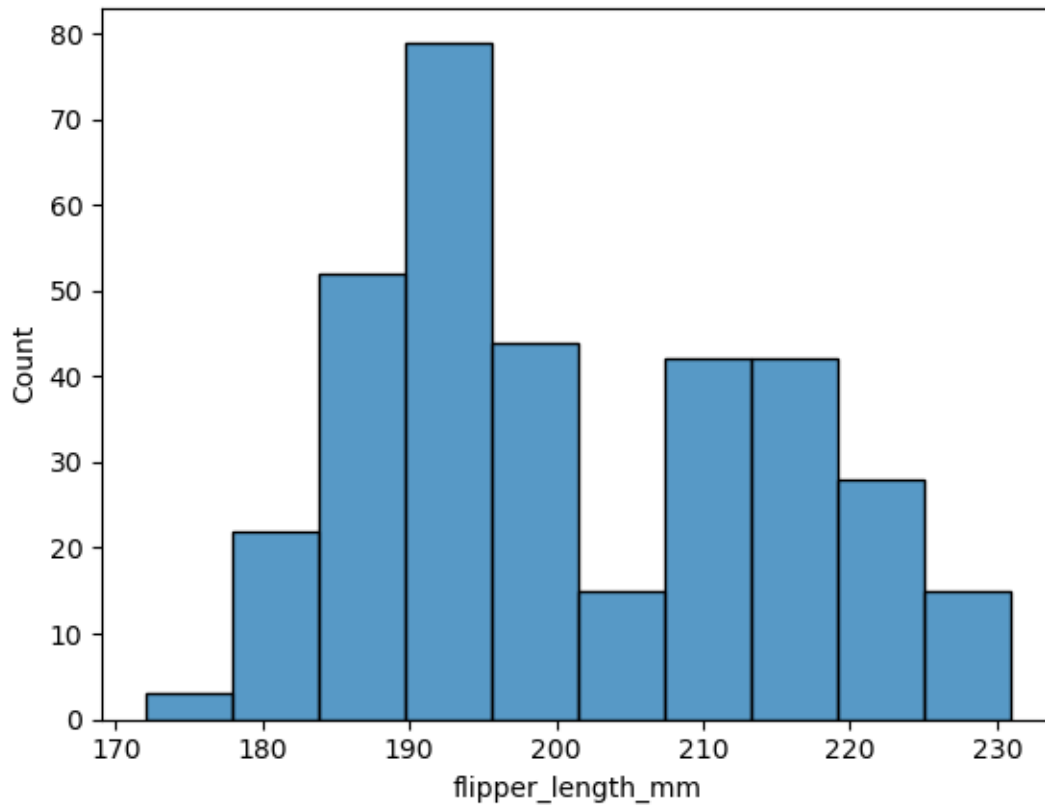
1.2.3 Histograms

“A histogram is a classic visualization tool that represents the distribution of one or more variables by counting the number of observations that fall within discrete bins.” - <https://seaborn.pydata.org/generated/seaborn.histplot.html>

Here we use histplot to show the distribution of the flipper_length_mm values.

```
[13]: sns.histplot(data=penguins_df,  
                  x="flipper_length_mm")
```

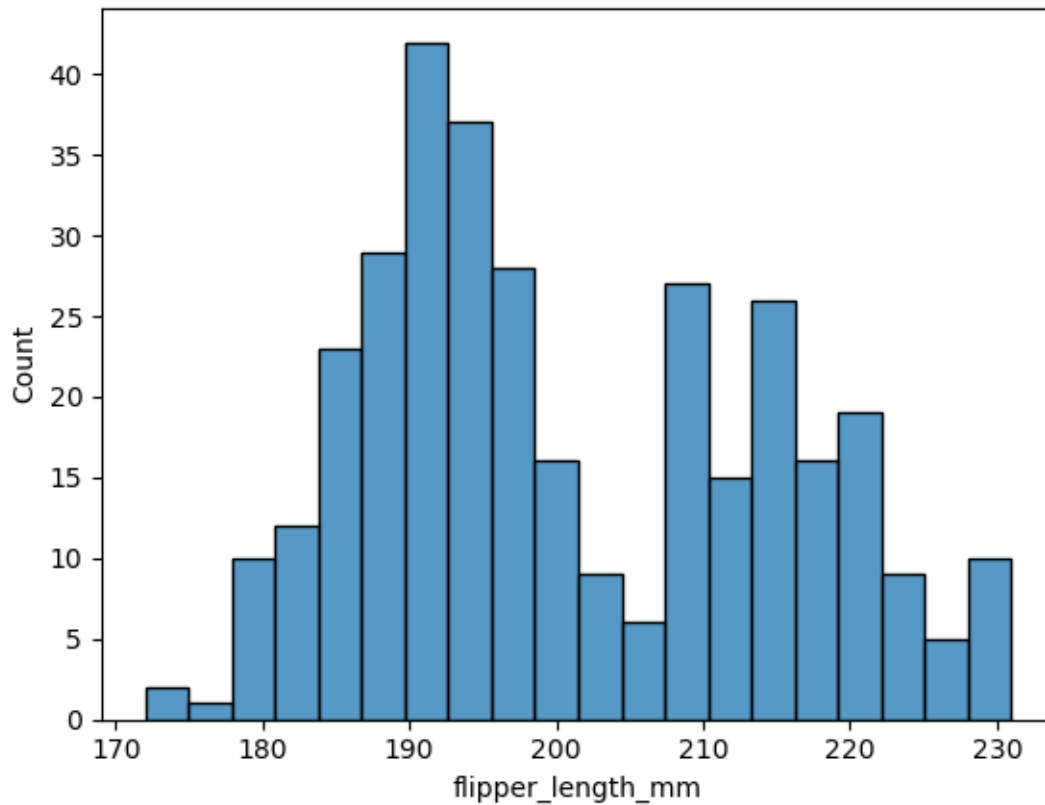
```
[13]: <Axes: xlabel='flipper_length_mm', ylabel='Count'>
```



There are many different ways to customise and configure the histogram as documented in the link above. For example, here we change the number of bins to 20.

```
[14]: sns.histplot(data=penguins_df,  
                  x="flipper_length_mm",  
                  bins=20)
```

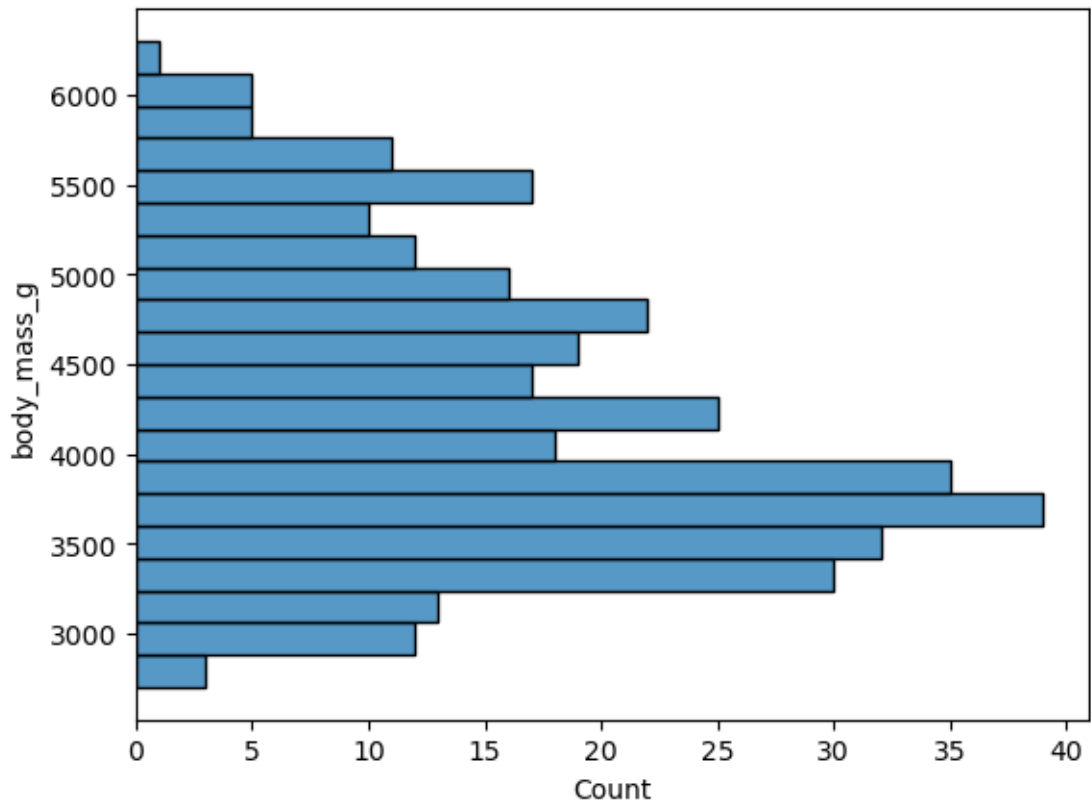
```
[14]: <Axes: xlabel='flipper_length_mm', ylabel='Count'>
```



And here we show a horizontal histogram by setting y rather x. In this case we're showing the distribution of body mass.

```
[15]: sns.histplot(data=penguins_df,  
                  y="body_mass_g",  
                  bins=20)
```

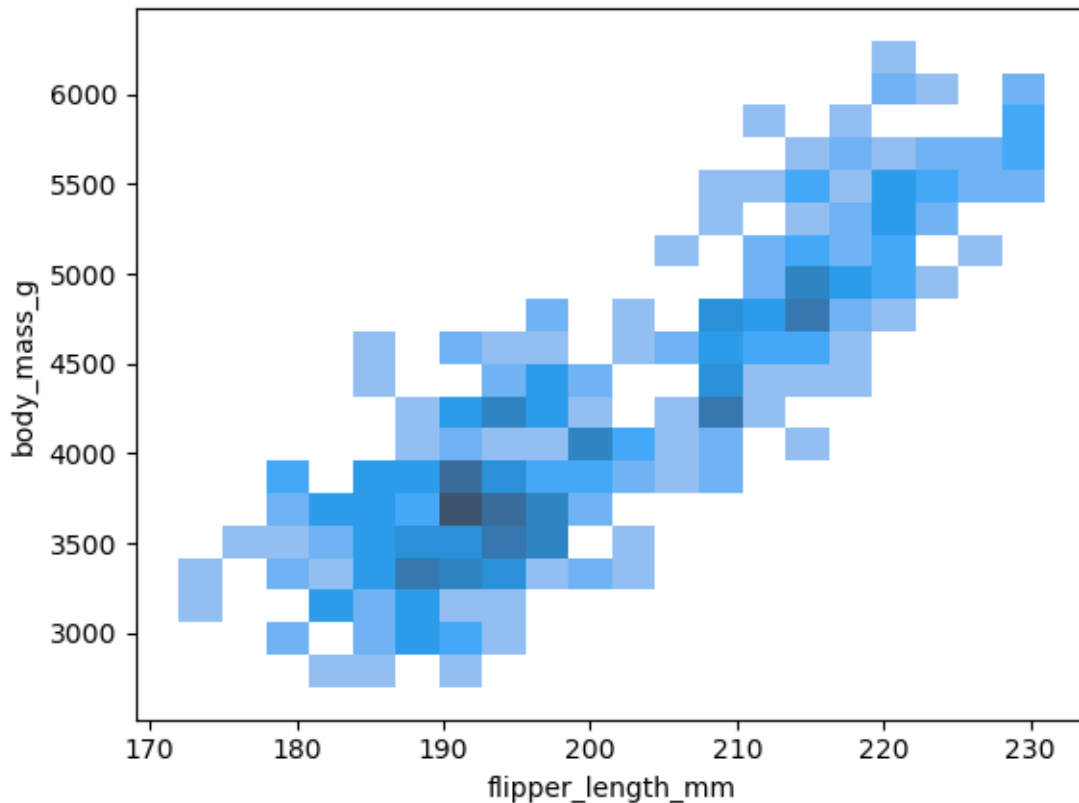
```
[15]: <Axes: xlabel='Count', ylabel='body_mass_g'>
```



We can also create a bivariate histogram, where the data is grouped into 2D bins, by setting both x and y.

```
[16]: sns.histplot(data=penguins_df,  
                  x="flipper_length_mm",  
                  y="body_mass_g",  
                  bins=20)
```

```
[16]: <Axes: xlabel='flipper_length_mm', ylabel='body_mass_g'>
```



1.2.4 Joint plots

Joint plots allow us to draw a plot of two variables with bivariate and univariate graphs.

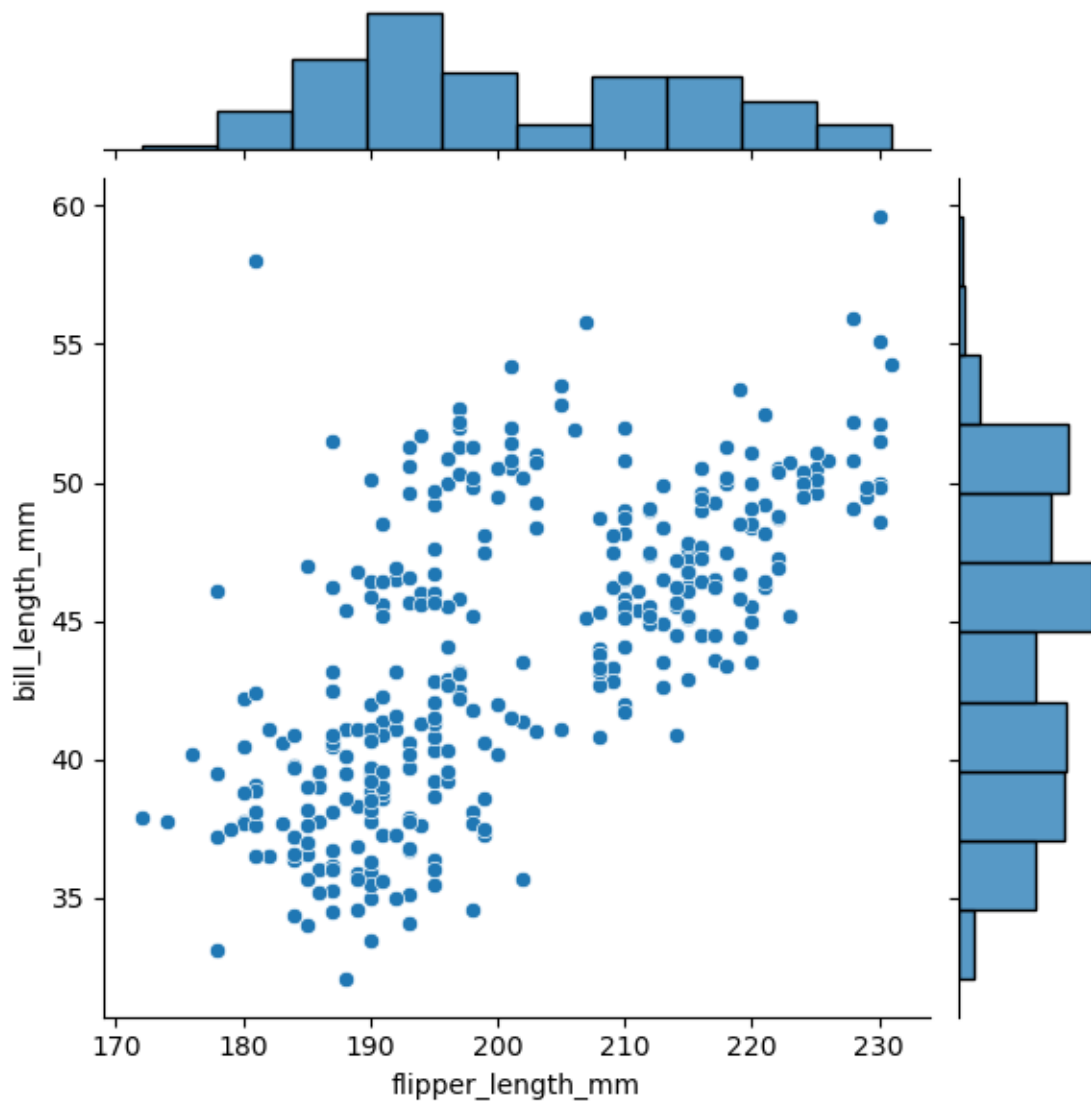
There are many other options for jointplot that will give different visualisations of the data, see <https://seaborn.pydata.org/generated/seaborn.jointplot.html>

See also JointGrid: <https://seaborn.pydata.org/generated/seaborn.JointGrid.html#seaborn.JointGrid>

Here we plot flipper length against bill length. Notice that we get a scatter plot and two histograms.

```
[17]: sns.jointplot(data=penguins_df,
                  x="flipper_length_mm",
                  y="bill_length_mm")
```

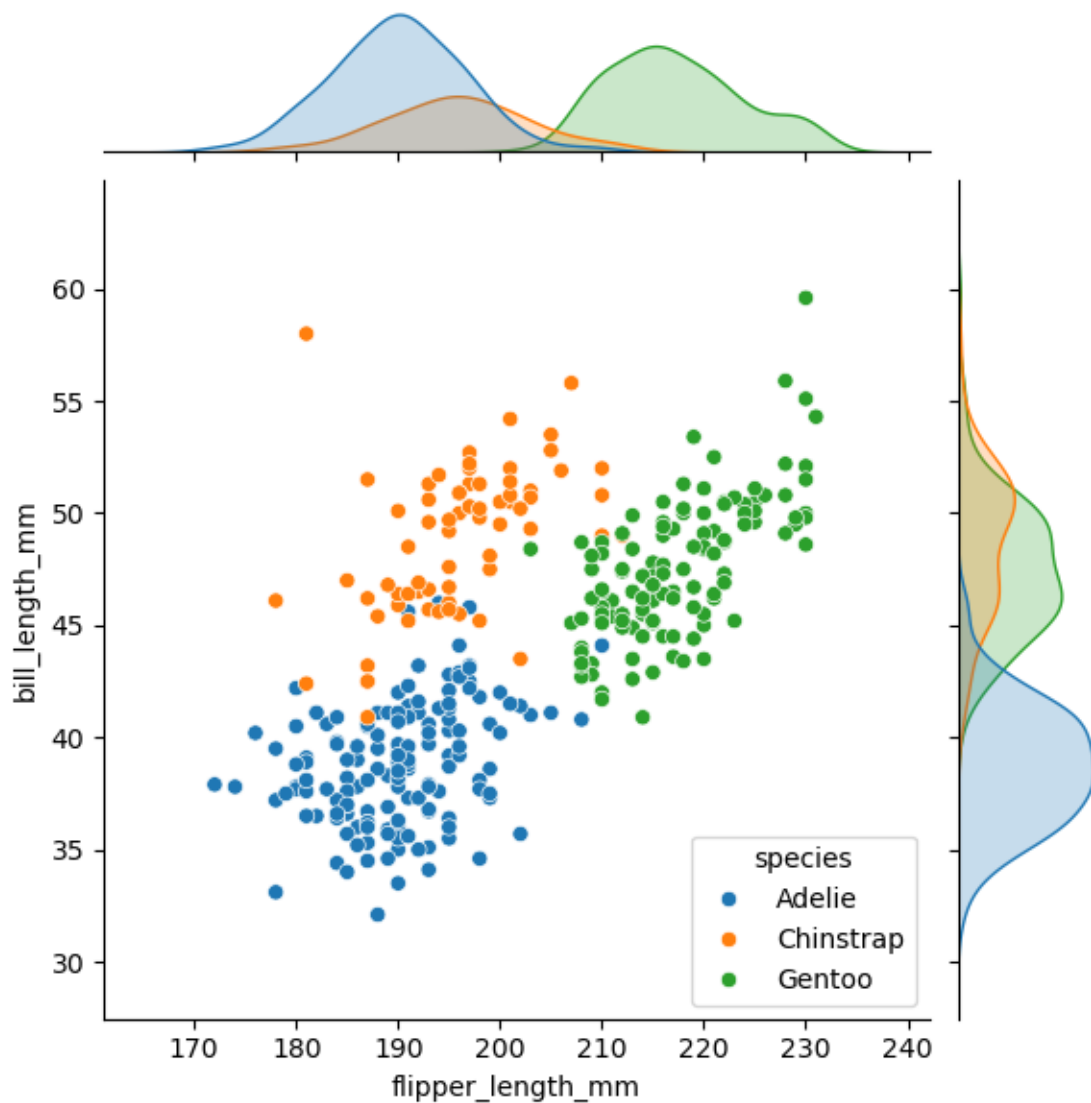
```
[17]: <seaborn.axisgrid.JointGrid at 0x1e96dcea7b0>
```



We can again use the hue argument to represent a third variable, either categorical or numeric, with colour. Here we show the penguin species.

```
[18]: sns.jointplot(data=penguins_df,  
                  x="flipper_length_mm",  
                  y="bill_length_mm",  
                  hue="species")
```

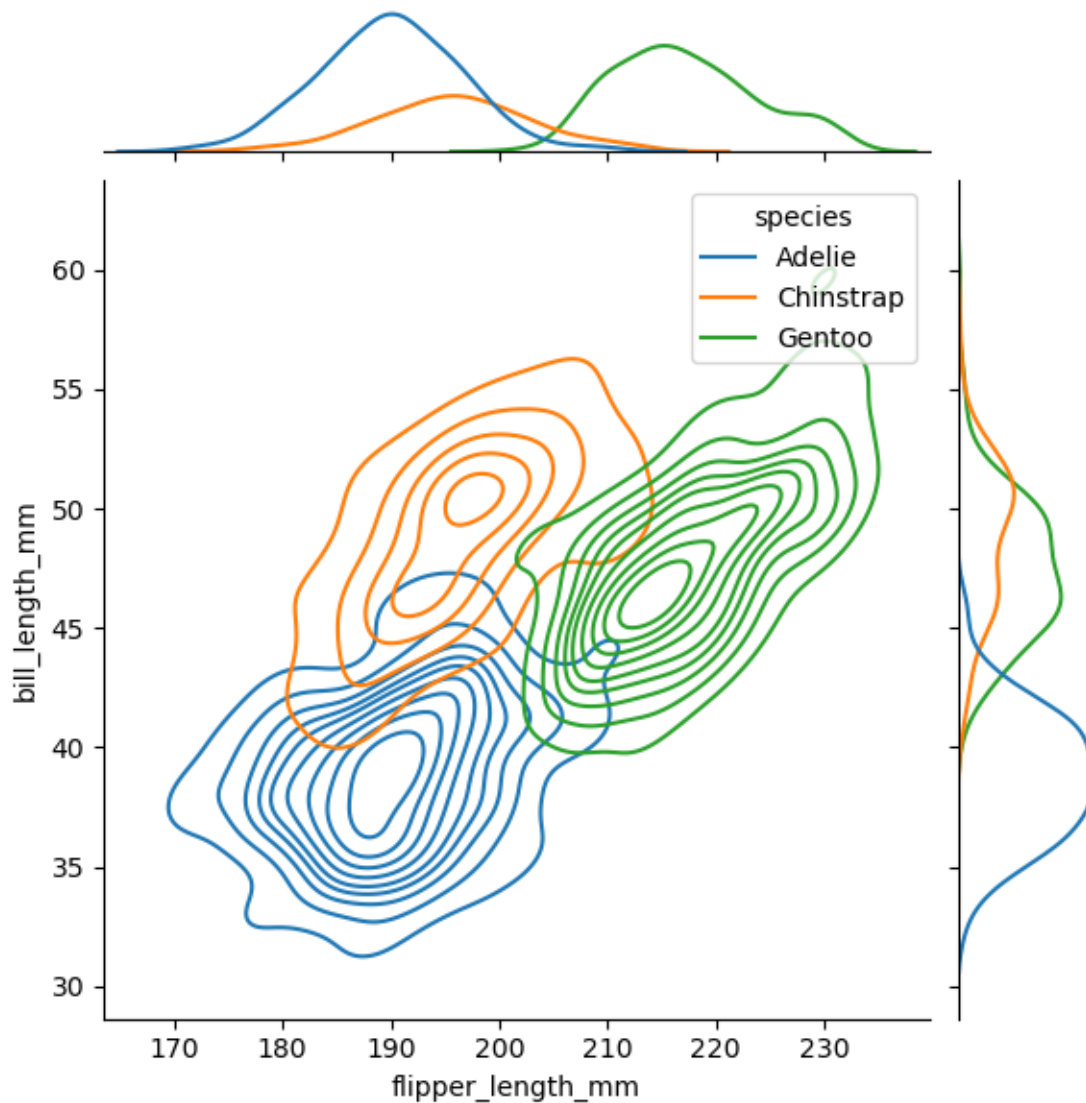
```
[18]: <seaborn.axisgrid.JointGrid at 0x1e96de75b80>
```



There are various kinds of jointplot. Here we set kind="kde" (kernel density estimate) to show density contours.

```
[19]: sns.jointplot(data=penguins_df,  
                 x="flipper_length_mm",  
                 y="bill_length_mm",  
                 hue="species",  
                 kind="kde")
```

```
[19]: <seaborn.axisgrid.JointGrid at 0x1e96fb1f500>
```

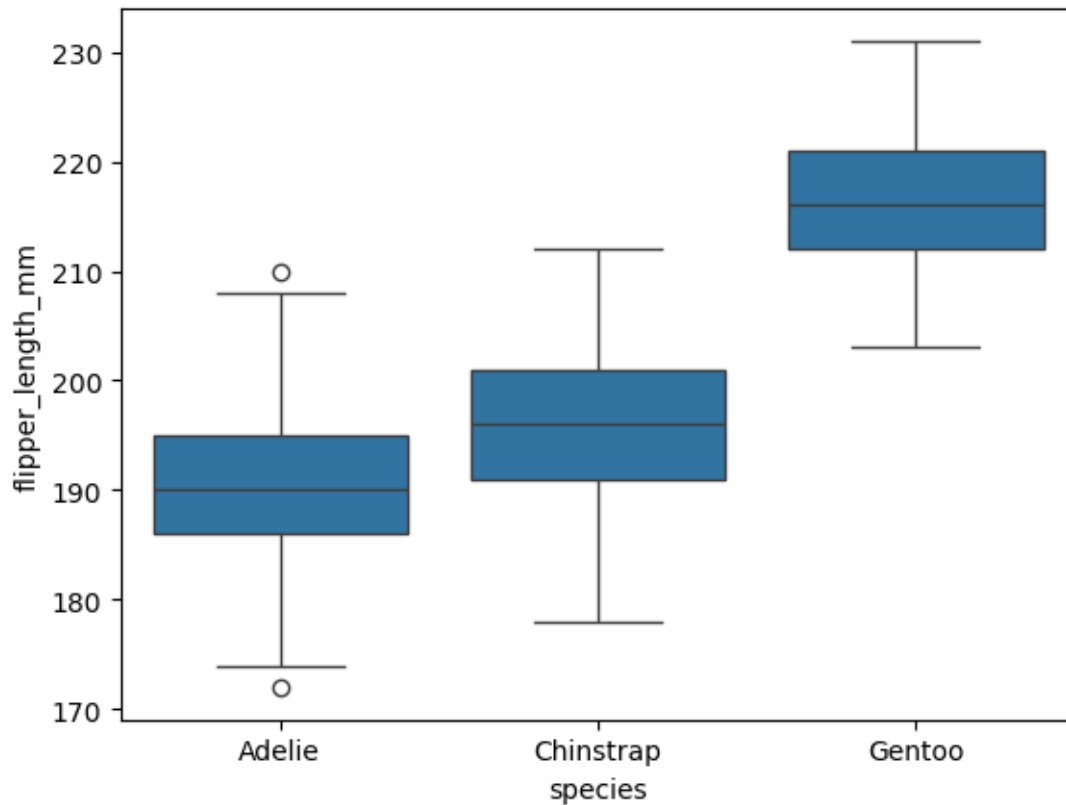


1.2.5 Boxplots

A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable. The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution, except for points that are determined to be “outliers” using a method that is a function of the inter-quartile range. - <https://seaborn.pydata.org/generated/seaborn.boxplot.html#seaborn.boxplot>

```
[20]: sns.boxplot(data=penguins_df,
                  x="species",
                  y="flipper_length_mm")
```

```
[20]: <Axes: xlabel='species', ylabel='flipper_length_mm'>
```

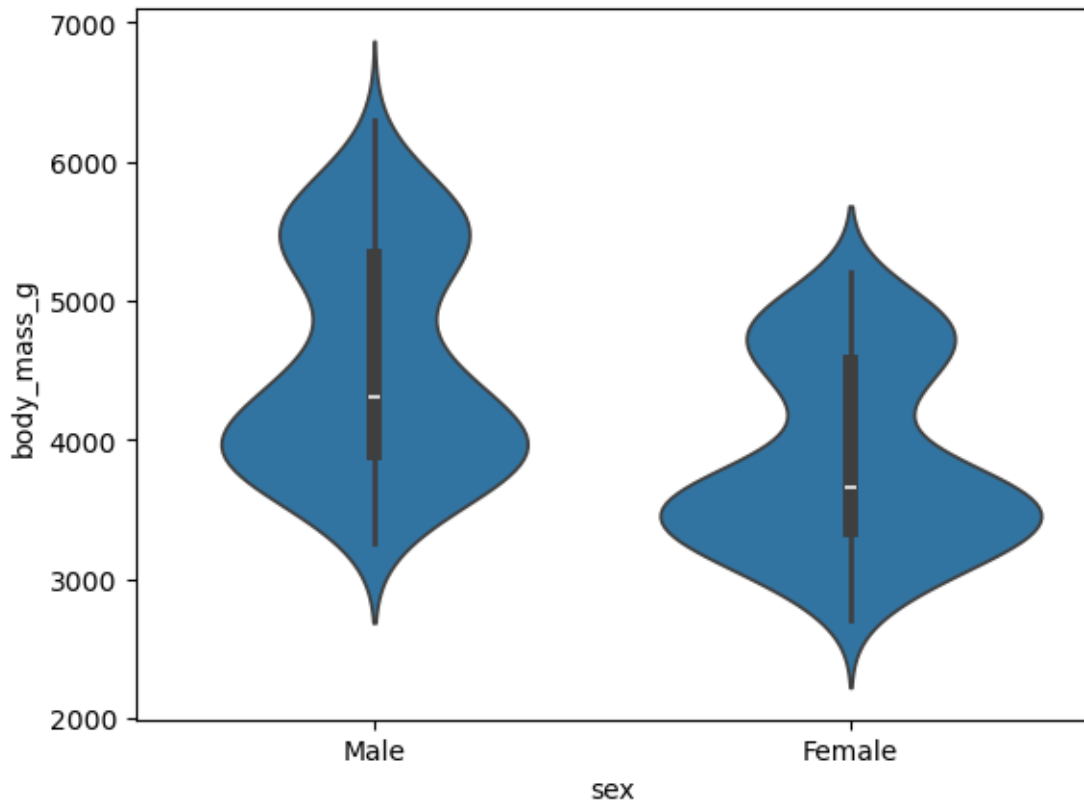
1.2.6 Violin plots

“A violin plot plays a similar role as a box-and-whisker plot. It shows the distribution of data points after grouping by one (or more) variables. Unlike a box plot, each violin is drawn using a kernel density estimate of the underlying distribution.” - <https://seaborn.pydata.org/generated/seaborn.violinplot.html>

In this example we show the body mass grouped by sex.

```
[21]: sns.violinplot(data=penguins_df,
                    x="sex",
                    y="body_mass_g")
```

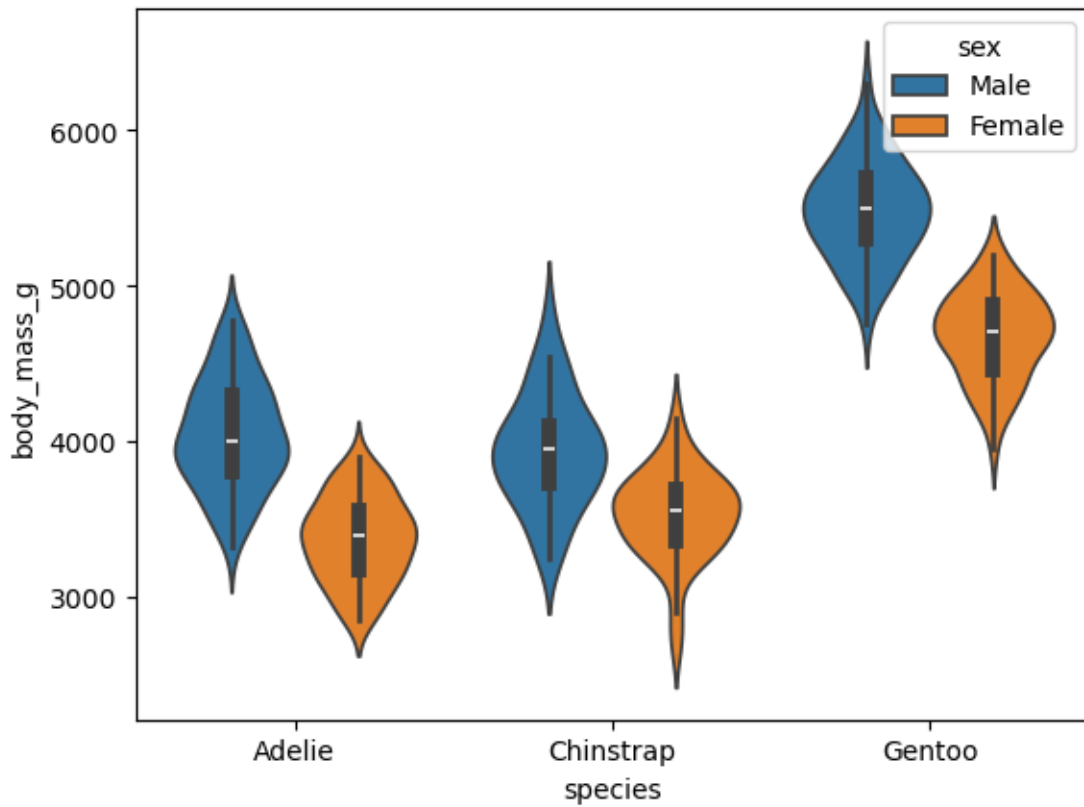
```
[21]: <Axes: xlabel='sex', ylabel='body_mass_g'>
```



Here we show the distributions of body mass broken down by species and sex.

```
[22]: sns.violinplot(data=penguins_df,  
                    x="species",  
                    y="body_mass_g",  
                    hue='sex')
```

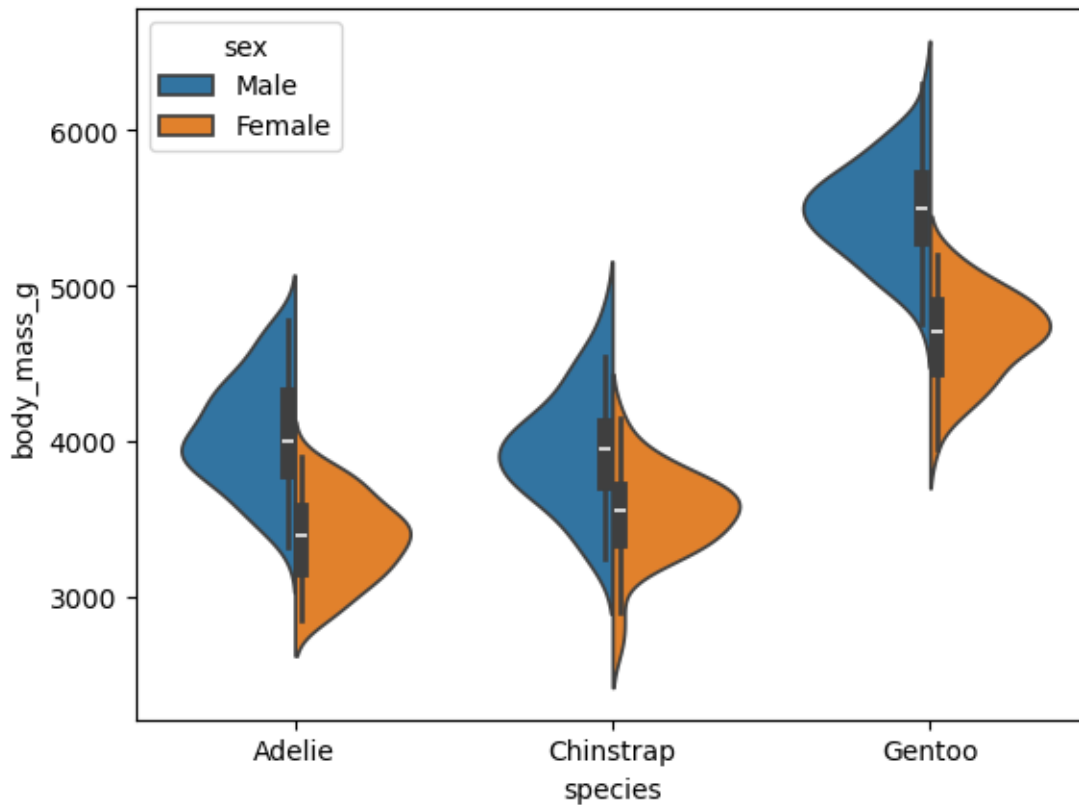
```
[22]: <Axes: xlabel='species', ylabel='body_mass_g'>
```



This next example is the same as the one above, except that we've set `split=True`.

```
[23]: sns.violinplot(data=penguins_df,  
                    x="species",  
                    y="body_mass_g",  
                    hue='sex',  
                    split=True)
```

```
[23]: <Axes: xlabel='species', ylabel='body_mass_g'>
```



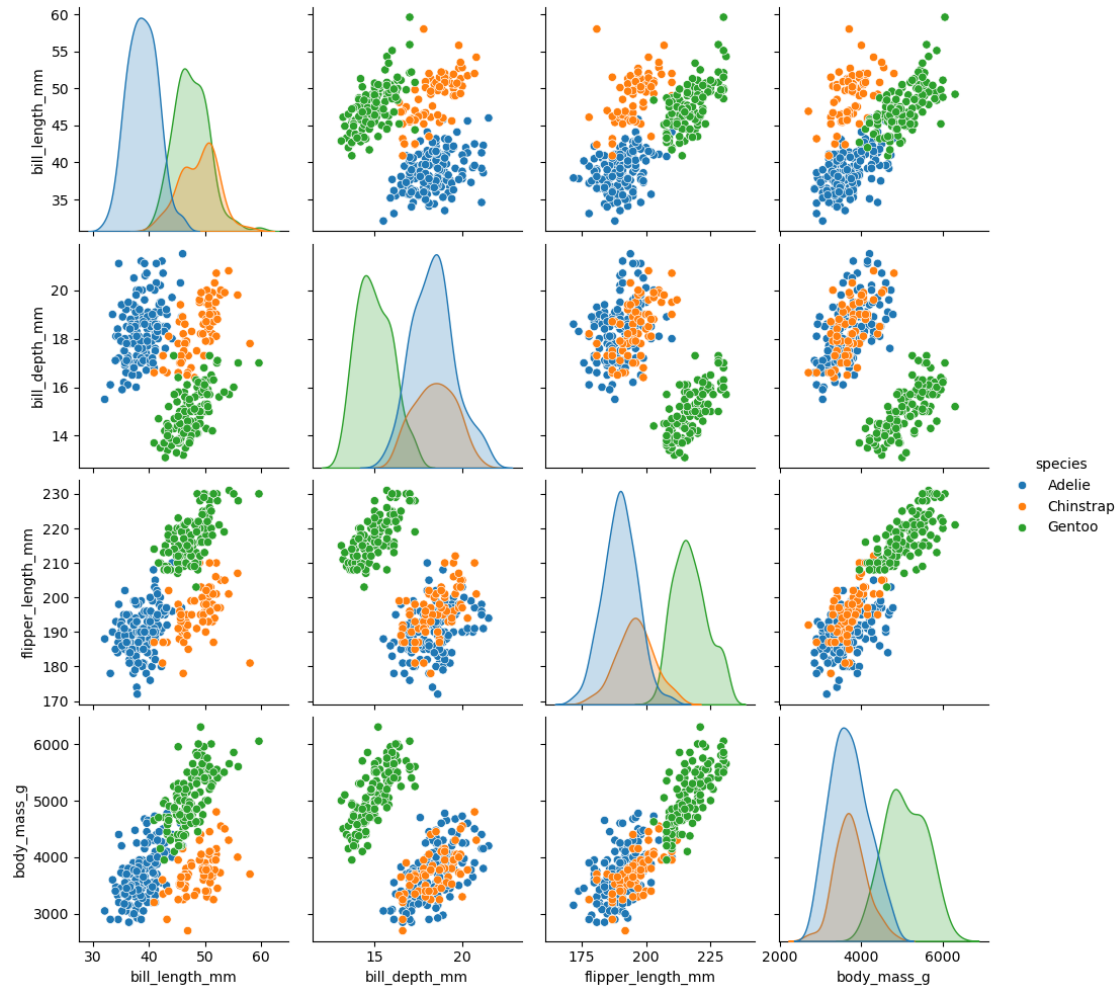
1.2.7 Pair plots

Pair plots allow us to get a great overview our data. In this example we see scatter plots for every pair of numeric variables, along the distributions of each numeric variable. We're also using hue to highlight species.

For more info on customising pairplots see <https://seaborn.pydata.org/generated/seaborn.pairplot.html>

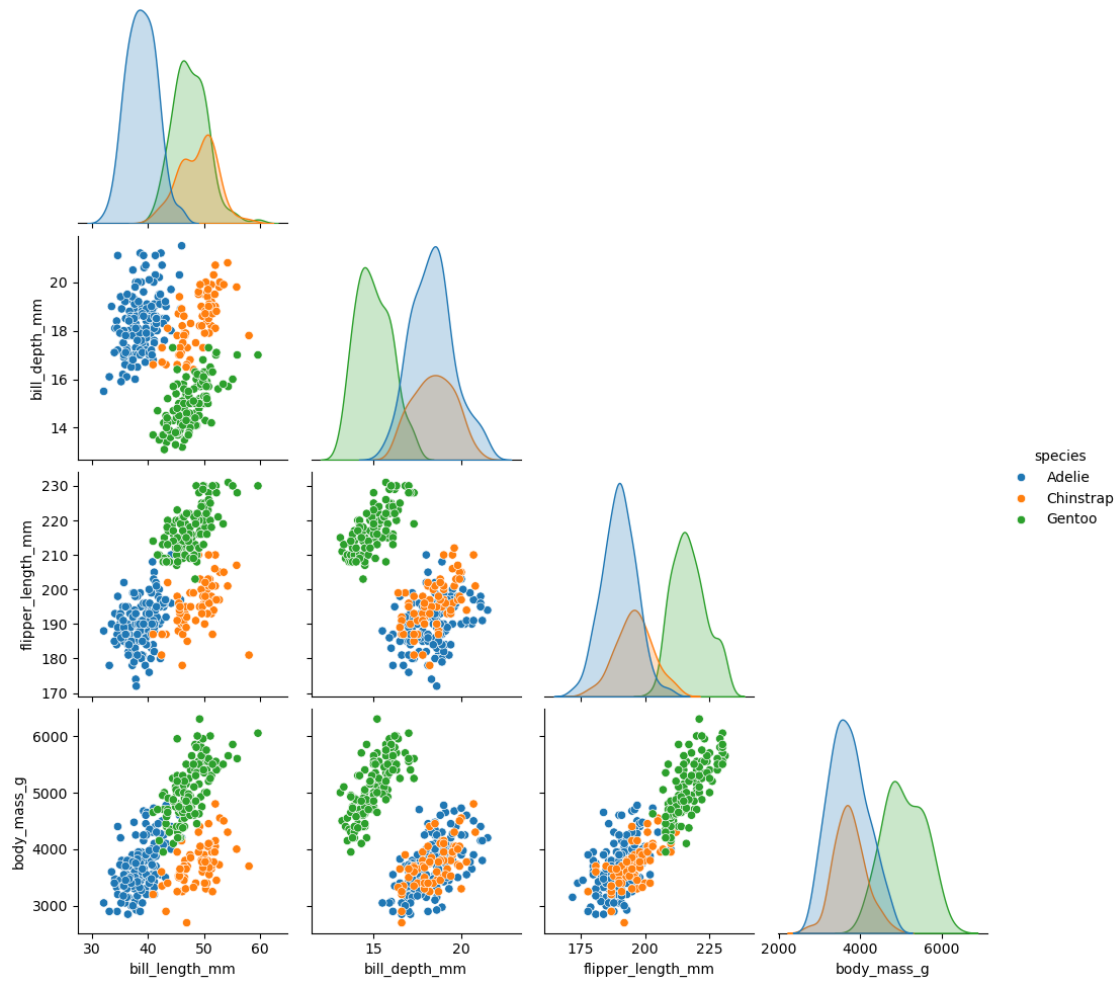
```
[24]: sns.pairplot(data=penguins_df,  
                  hue="species")
```

```
[24]: <seaborn.axisgrid.PairGrid at 0x1e970412630>
```



```
[25]: sns.pairplot(data=penguins_df,
                  hue="species",
                  corner=True)
```

```
[25]: <seaborn.axisgrid.PairGrid at 0x1e9721ccc20>
```



1.2.8 Heatmaps

We can use a heatmap to visualise a grid of numeric values. In this example we will visualise the linear correlations between the numeric variables.

First we will need to work out the correlations. We can do this using Pandas with a little help from NumPy.

```
[26]: import numpy as np

# Select only numeric columns for correlation
numeric_columns = penguins_df.select_dtypes(include=[np.number])

numeric_columns
```

```
[26]:      bill_length_mm  bill_depth_mm  flipper_length_mm  body_mass_g
0              39.1           18.7           181.0         3750.0
```

1	39.5	17.4	186.0	3800.0
2	40.3	18.0	195.0	3250.0
3	NaN	NaN	NaN	NaN
4	36.7	19.3	193.0	3450.0
..
339	NaN	NaN	NaN	NaN
340	46.8	14.3	215.0	4850.0
341	50.4	15.7	222.0	5750.0
342	45.2	14.8	212.0	5200.0
343	49.9	16.1	213.0	5400.0

[344 rows x 4 columns]

```
[27]: # Compute the correlation matrix using the corr DataFrame method
correlations = numeric_columns.corr()

correlations
```

```
[27]:
```

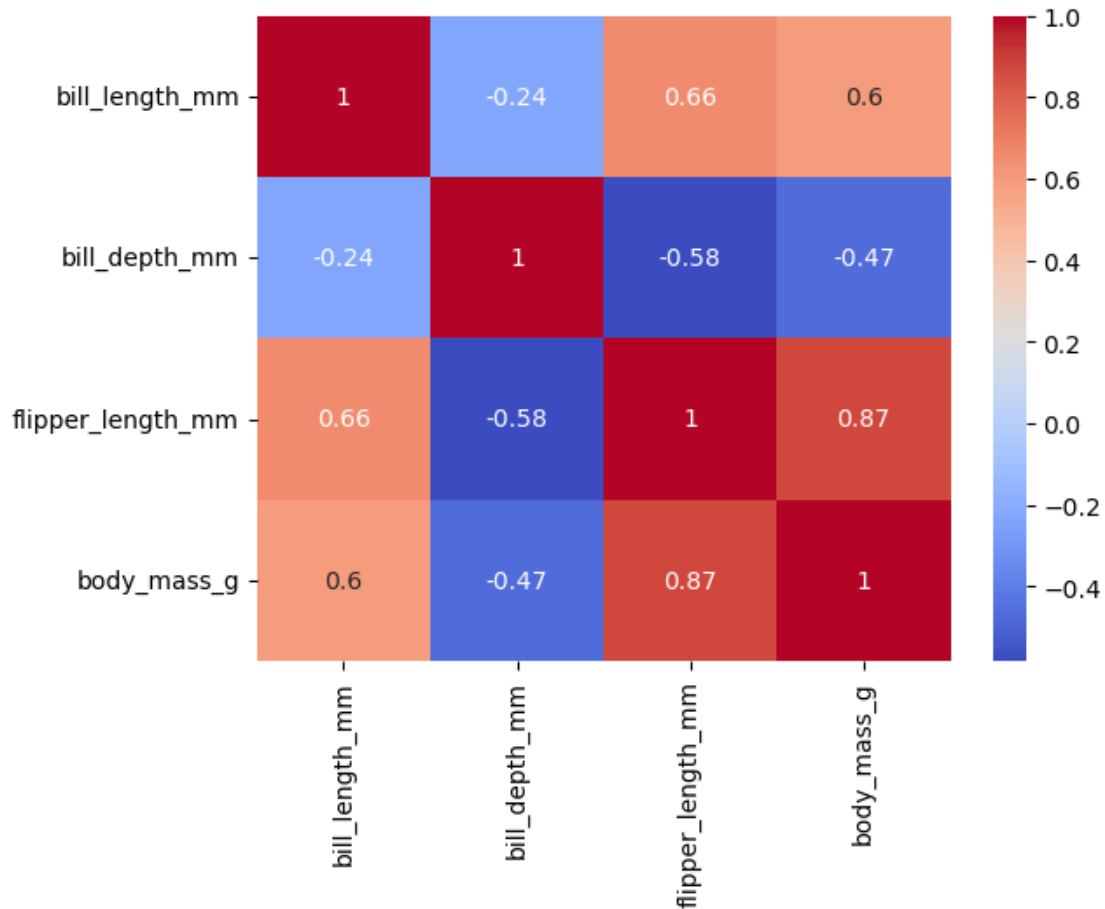
	bill_length_mm	bill_depth_mm	flipper_length_mm	\
bill_length_mm	1.000000	-0.235053	0.656181	
bill_depth_mm	-0.235053	1.000000	-0.583851	
flipper_length_mm	0.656181	-0.583851	1.000000	
body_mass_g	0.595110	-0.471916	0.871202	

	body_mass_g
bill_length_mm	0.595110
bill_depth_mm	-0.471916
flipper_length_mm	0.871202
body_mass_g	1.000000

We can now use a heatmap to make the larger positive and negative correlations easier to spot.

```
[28]: sns.heatmap(data=correlations,
                  annot=True,
                  cmap="coolwarm")
```

```
[28]: <Axes: >
```



For more on heatmaps see <https://seaborn.pydata.org/generated/seaborn.heatmap.html>

1.3 Using Matplotlib alongside Seaborn

Seaborn is a high-level interface to Matplotlib. This means that we can use both of them together. Seaborn allows us to create a wide range of visualisations quickly and easily. Matplotlib gives us detailed control over our visualisations when we need it.

```
[29]: import matplotlib.pyplot as plt
```

```
[30]: # Create heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(data=correlations, annot=True, cmap="coolwarm", fmt=".2f")

# Add a title and labels
plt.title("Correlation Heatmap of Penguin Attributes", fontsize=14, pad=15)
plt.xlabel("Attributes", fontsize=12)
plt.ylabel("Attributes", fontsize=12)
```



```
# Rotate x-axis labels for better readability
plt.xticks(rotation=45, ha="right")

# Show the heatmap
plt.tight_layout()
plt.show()
```

