



HyperionDev

Data Visualisation

August 2024

Data Science Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: [**Questions**](#)

Data Science Session Housekeeping cont.

- For all **non-academic questions**, please submit a query:
www.hyperiondev.com/support
- Report a **safeguarding** incident:
www.hyperiondev.com/safeguardreporting
- We would love your **feedback** on lectures: [Feedback on Lectures](#)

Learning Outcomes

- ❖ **Describe** the different types of data (discrete, continuous, time series, categorical) and the appropriate basic visualisations for each.
- ❖ **Create** advanced data visualisations using Seaborn, including kernel density plots, box plots, violin plots, heatmaps, and pairplots.

Learning Outcomes

- ❖ **Apply** Matplotlib functions to create basic data visualisations, including bar plots, pie charts, scatterplots, and line graphs.

Problem Statement

In data science, effectively communicating insights from data is just as important as the analysis itself. Different types of data require different visualisation techniques to accurately and clearly convey information. While basic visualisations like bar plots and line graphs are essential tools, more complex datasets and relationships often require advanced visualisations to reveal deeper insights.

- ❖ How can we visualise data in our code that allows us to generate insights into our data?

Lecture Overview

- Types of Data
- Basic Visualisations
- Matplotlib
- Advanced Visualisations
 - ↳ Seaborn



Data Visualisation



Key Goals of Data Visualisation

- ❖ Expose **underlying trends, outliers, and relationships within data**
- ❖ Support **intuitive exploration and rapid insight generation**
- ❖ Depending on what **information we want** from the dataset, we use the data to **answer the questions through visualisations**
- ❖ Create **appropriate, easily read visual narratives** that resonate with your audience.

Types of Data



Types of Data

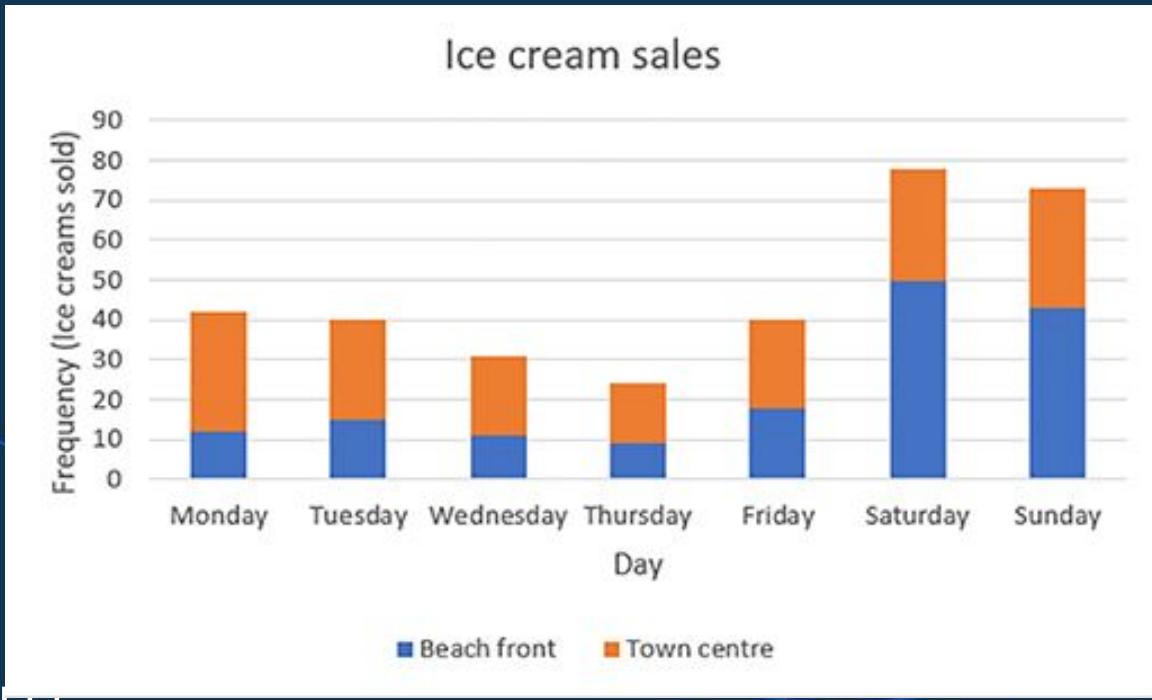
- ❖ **Discrete data:** Can take specific values, with an infinite range, e.g : [1, 2, 3, 4, 5, ...]
- ❖ **Categorical data:** Can take specific values, with a limited range, e.g. [Dog, Cat, Hamster, Fish]
- ❖ **Continuous data:** Available values in a spectrum, there are an infinite number of values, e.g. temperature or distance
- ❖ **Time series data:** Data changes along with some form of time-related progression

Basic Visualisations



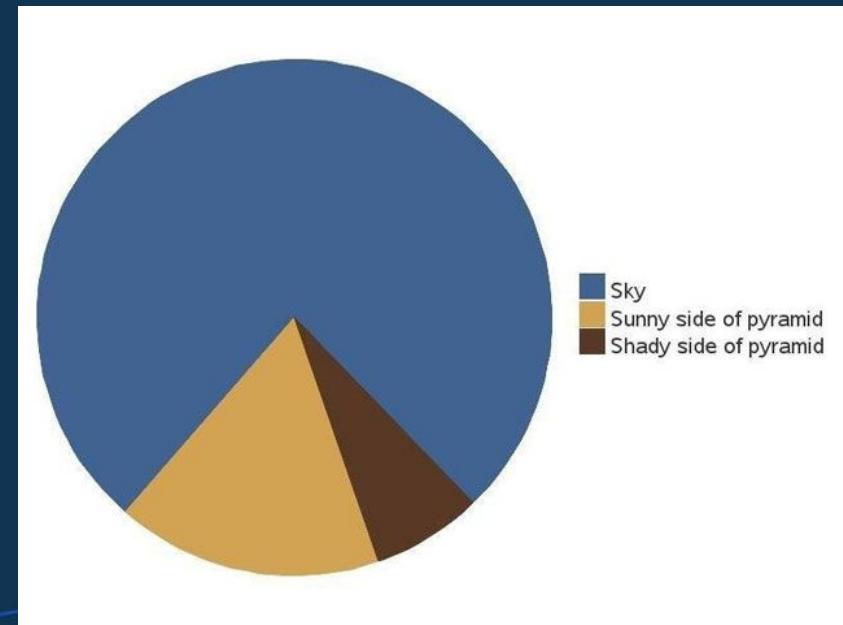
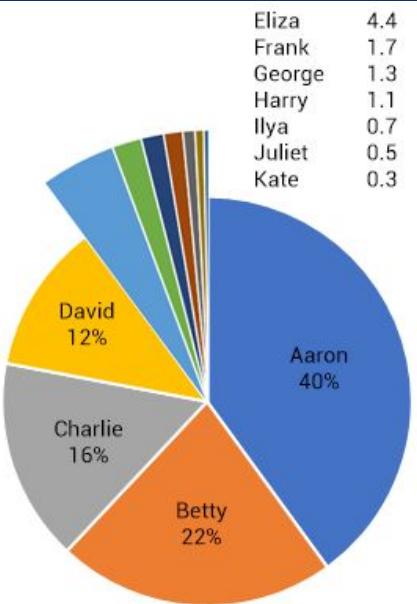
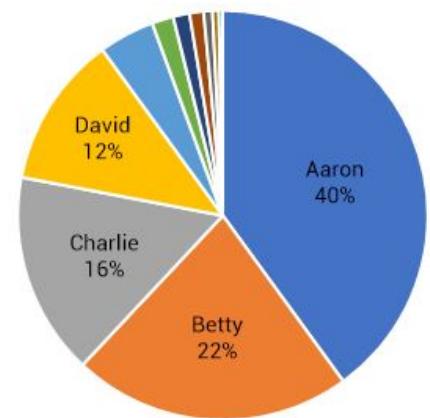
Barplot (Bar chart)

Good for plotting Categorical vs. Continuous /Discrete data



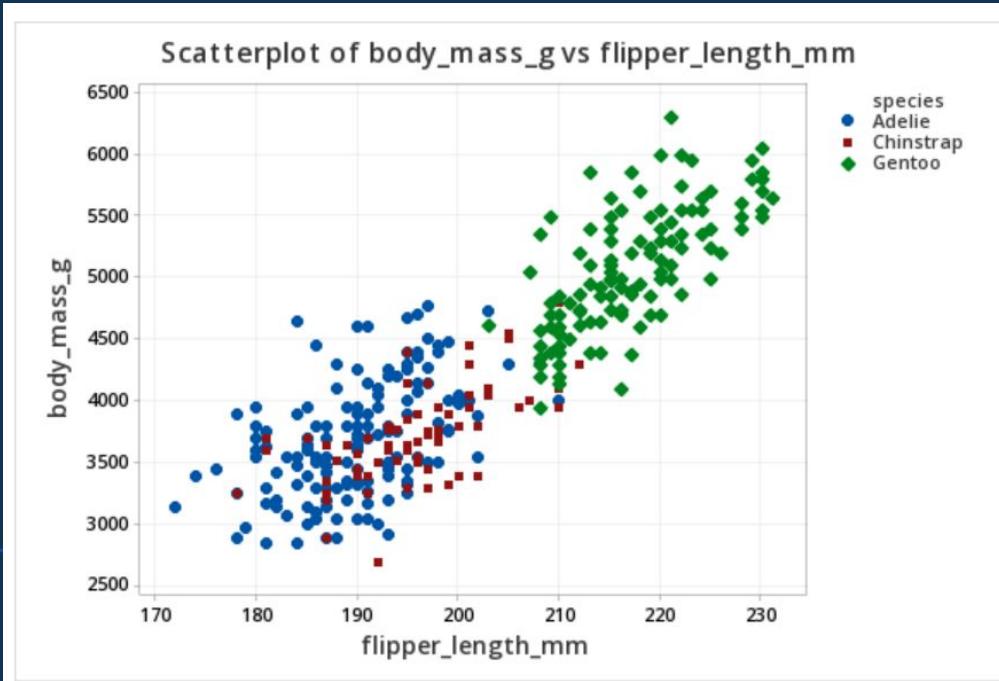
Pie chart

Good for plotting Categorical vs. Discrete data. Also great for getting a sense of proportions



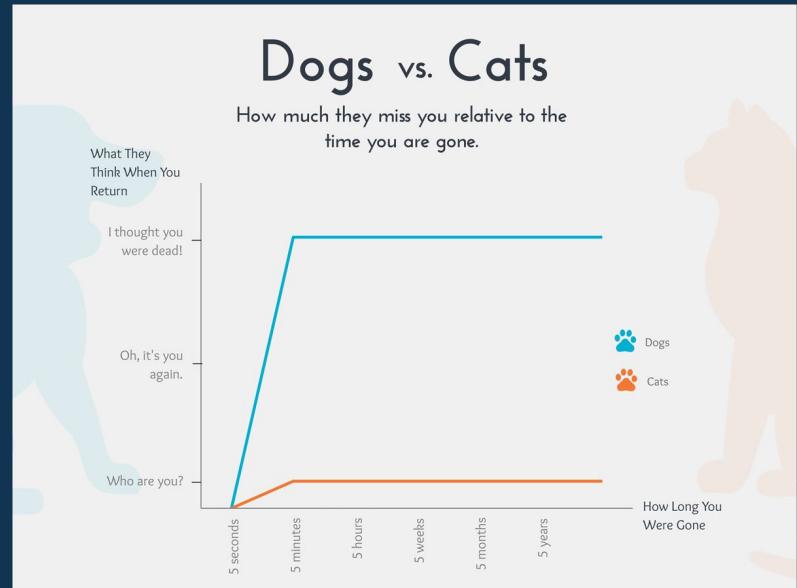
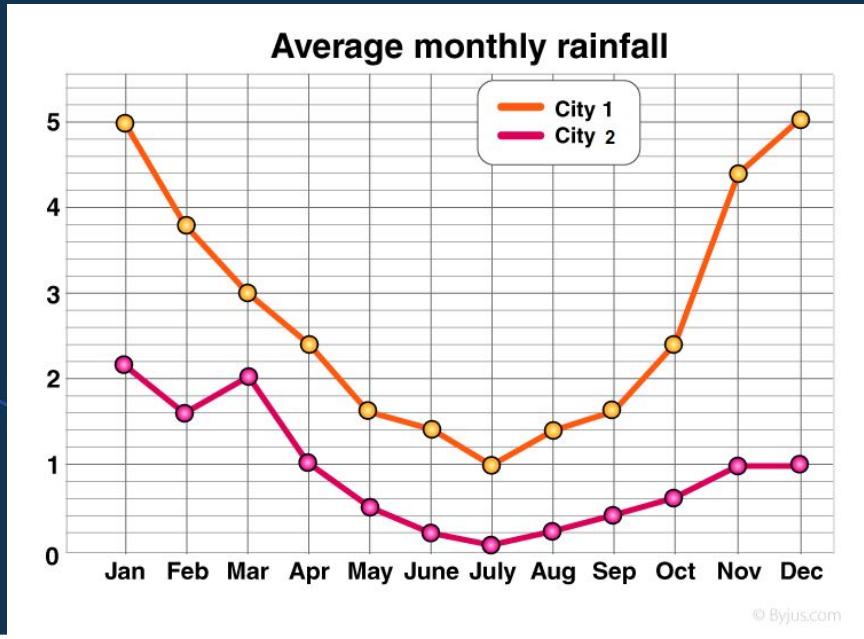
Scatterplots

Great for plotting Discrete vs. Discrete data or Continuous vs. Continuous data. Useful for finding relationships between variables.



Line graphs

Good for plotting Discrete / Continuous vs. Discrete / Continuous data or comparing with a Time Series.



Matplotlib



Matplotlib

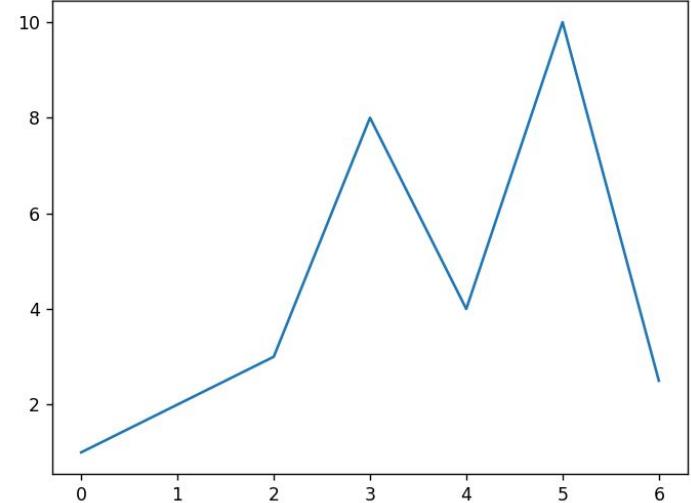
Installation

```
pip install matplotlib
```

Importing

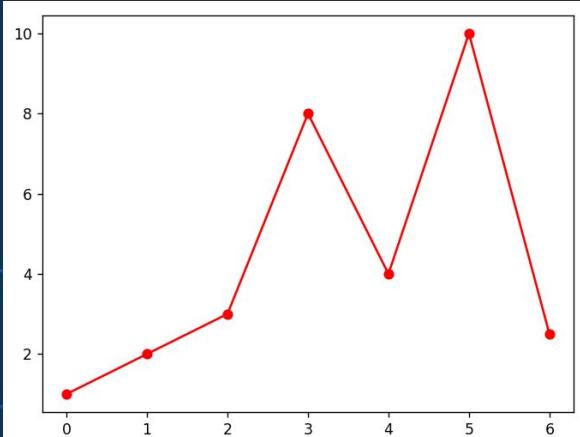
```
import matplotlib.pyplot as plt
```

```
import matplotlib.pyplot as plt  
import numpy as np  
  
y whole points = np.array([1, 2, 3, 8, 4, 10, 2.5])  
  
plt.plot(y whole points)  
plt.show()
```



Matplotlib

```
import matplotlib.pyplot as plt  
import numpy as np  
  
y whole points = np.array([1, 2, 3, 8, 4, 10, 2.5])  
  
plt.plot(y whole points, 'o-r')  
  
plt.show()
```



Markers

o = Circle, * = Star,
. = Point, x = Cross,
s = Square, d = diamond

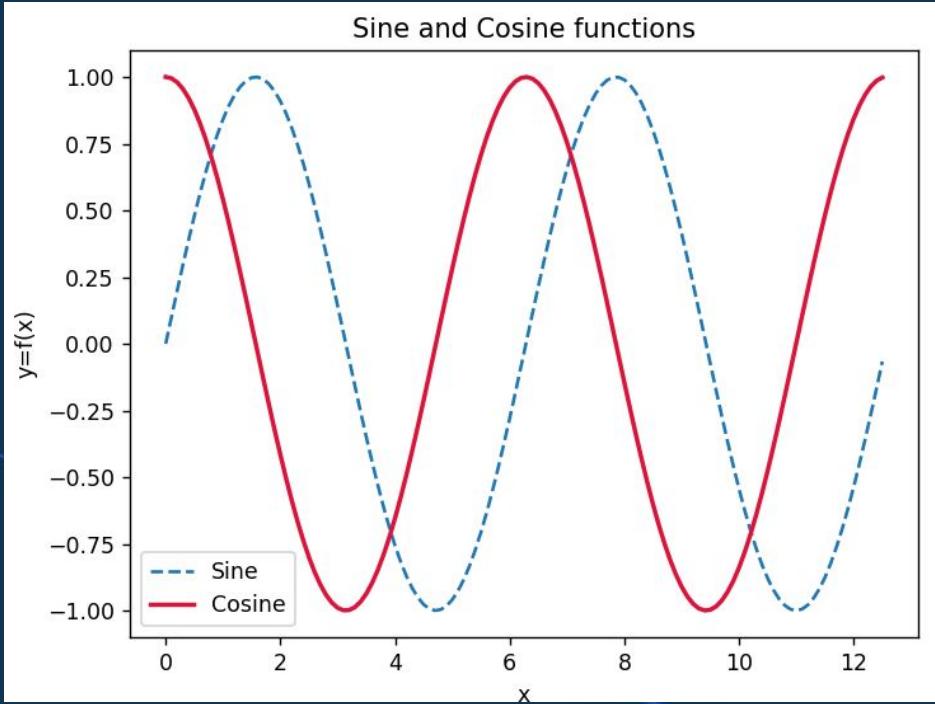
Linestyle ls

- Solid line
- : Dotted line
- Dashed line
- . Dashed/dotted line

Colour

List of colours
b = Blue, r = Red, g = Green,
c = Cyan, m = Magenta, y =
Yellow, k = Black, w = White

Matplotlib



```
import matplotlib.pyplot as plt
import numpy as np

pi = np.pi
# Creating x axis with range
#and y axis with sine/cosine
x = np.arange(0, 4*pi, 0.1)
y1 = np.sin(x)
y2 = np.cos(x)

#Plotting sine/cosine on the same plot
plt.plot(x,y1, label='Sine', ls='--')
plt.plot(x,y2, label='Cosine', lw=2, c='crimson')

#x-axis, y-axis Label, and plot title
plt.xlabel('x')
plt.ylabel('y=f(x)')
plt.title('Sine and Cosine functions')

#Legend for the two curves
plt.legend()
plt.show() #plt.savefig('sine.png')
```

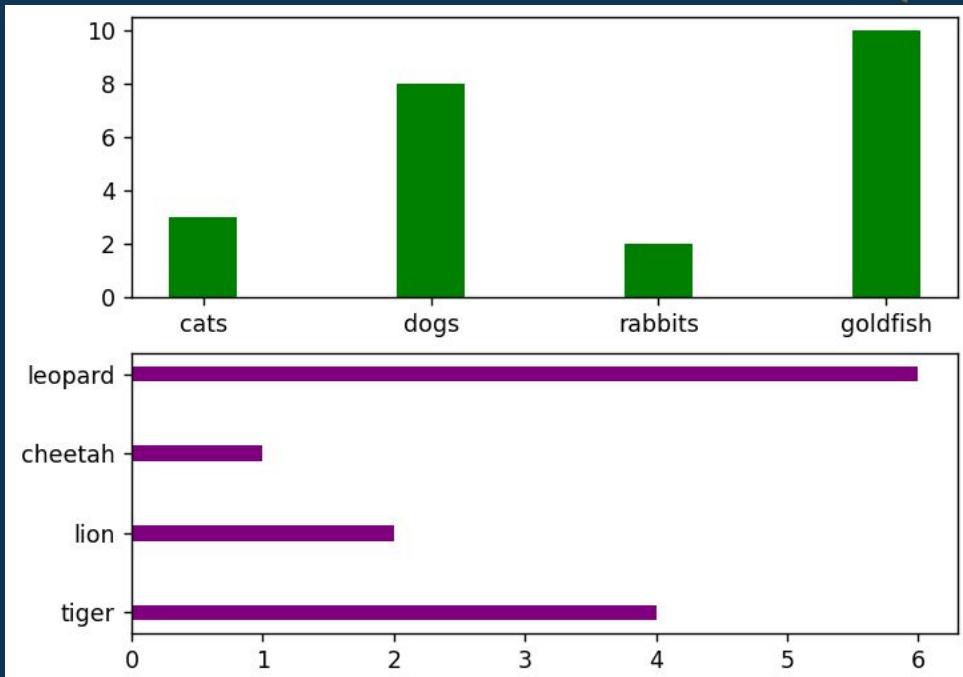


Matplotlib: Barplot

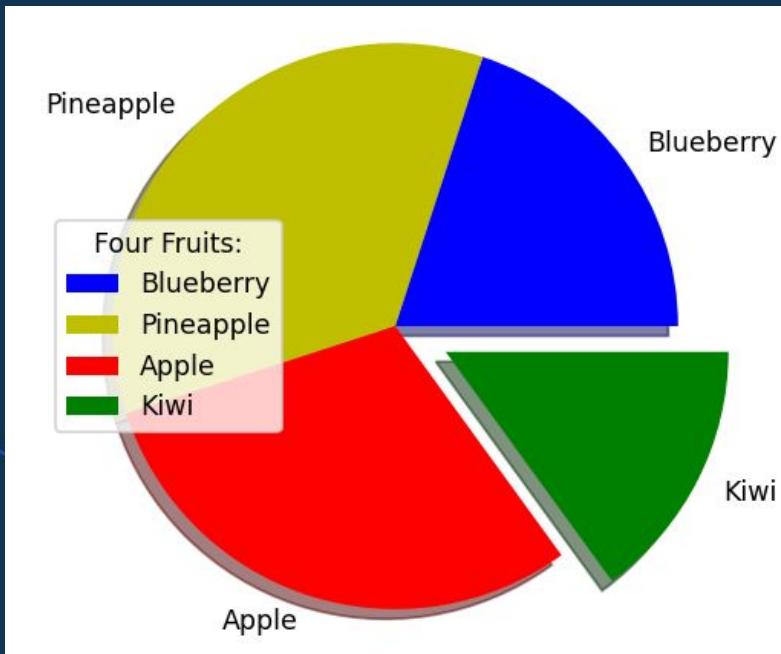
```
import matplotlib.pyplot as plt
import numpy as np

#x,y for 1st and 2nd barplots
x1 = np.array(["cats", "dogs", "rabbits", "goldfish"])
y1 = np.array([3, 8, 2, 10])
x2 = np.array(["tiger", "lion", "cheetah", "leopard"])
y2 = np.array([4, 2, 1, 6])

#SubPlot, parameters(rows, columns, index of current plot)
plt.subplot(2,1,1)
plt.bar(x1, y1, color = 'g', width=0.3)
plt.subplot(2,1,2)
plt.bart(x2, y2, color = '#800080', height=0.2)
plt.show()
```



Matplotlib: Pie chart



```
import matplotlib.pyplot as plt
import numpy as np

y = np.array([20, 35, 30, 15])
mylabels = ['Blueberry', 'Pineapple', 'Apple', 'Kiwi']
mycolors = ['b', 'y', 'r', 'g']
myexplode = [0, 0, 0, 0.2]

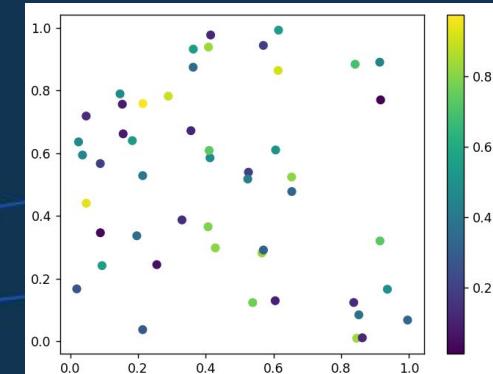
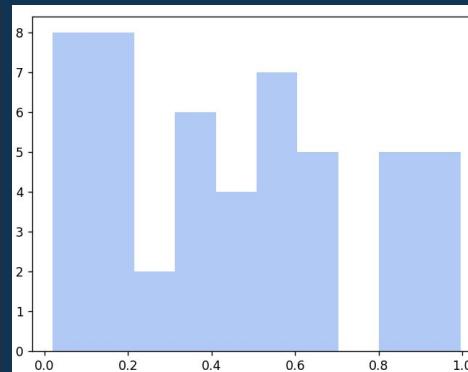
plt.pie(y, labels = mylabels, colors = mycolors,
        explode = myexplode, shadow = True)
plt.legend(loc='center left', title = "Four Fruits:")
plt.show()
```

Histograms & Scatterplots

A graph showing frequency distributions, the number of observations within each given interval.

```
N = 50  
x = np.random.rand(N)  
y = np.random.rand(N)  
colors = np.random.rand(N)  
  
plt.hist(x, color='cornflowerblue', alpha=0.5)  
plt.show()
```

```
plt.scatter(x, y, c=colors)  
plt.colorbar()  
plt.show()
```



Advanced Visualisations



HyperionDev

Why advanced visualisation?

- ❖ Complex data demands **more sophisticated and flexible visual analysis** methods
- ❖ Answering **specific questions** sometimes requires **going beyond standard plots**.

Seaborn

Seaborn

- ❖ **Matplotlib** is a low-level plotting library, create highly customizable visualizations.
- ❖ **Seaborn**, built on top of Matplotlib, is a high-level interface for creating statistical graphics, with a range of built-in statistical functions for complex statistical analyses with the visualizations.
- ❖ Designed to work with **Pandas** dataframes

Seaborn

Installation

```
pip install seaborn
```

```
# Import seaborn, matplotlib
import seaborn as sns
import matplotlib.pyplot as plt

#Check available datasets
print(sns.get_dataset_names())

# Load an example dataset
peng_df = sns.load_dataset("penguins")
#Check the dataset
peng_df.info()
```

Importing

```
import seaborn as sns
```

- ❖ `load_dataset()` is like `pd.read_csv()`

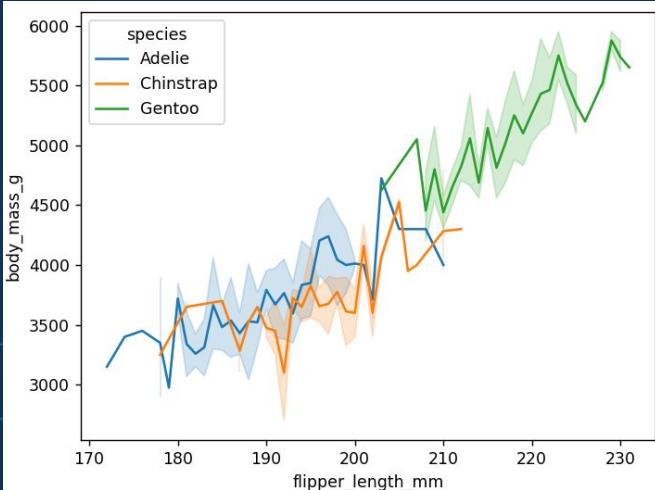
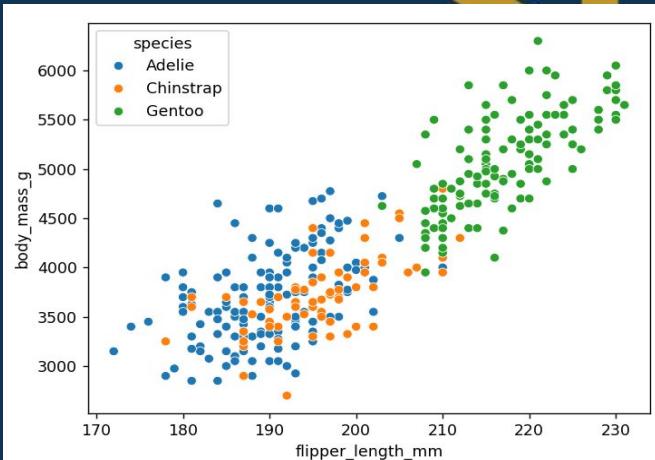
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   species          344 non-null    object 
 1   island            344 non-null    object 
 2   bill_length_mm    342 non-null    float64
 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 
```

Seaborn plots

Scatter and Line plots

```
#Scatterplot
sns.scatterplot(x="flipper_length_mm", y="body_mass_g",
                 data=peng_df, hue="species")
plt.show()
```

```
#Lineplot
sns.lineplot(x="flipper_length_mm", y="body_mass_g",
              data=peng_df, hue="species")
plt.show()
```



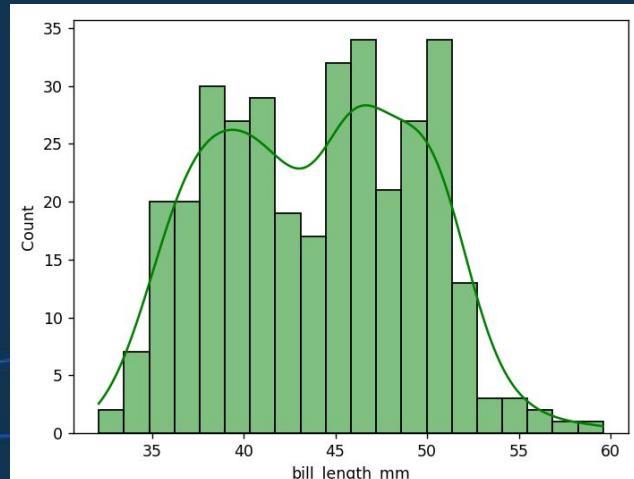
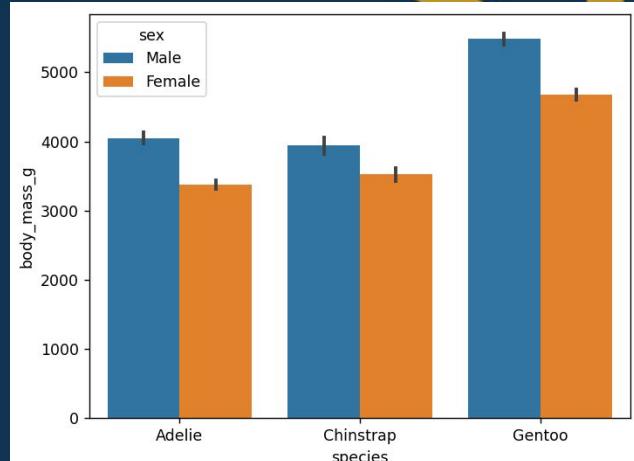
- ❖ If not in Jupyter or IPython notebook, explicitly call `matplotlib.pyplot` for displaying the plot

Seaborn plots

Bar plots and histogram

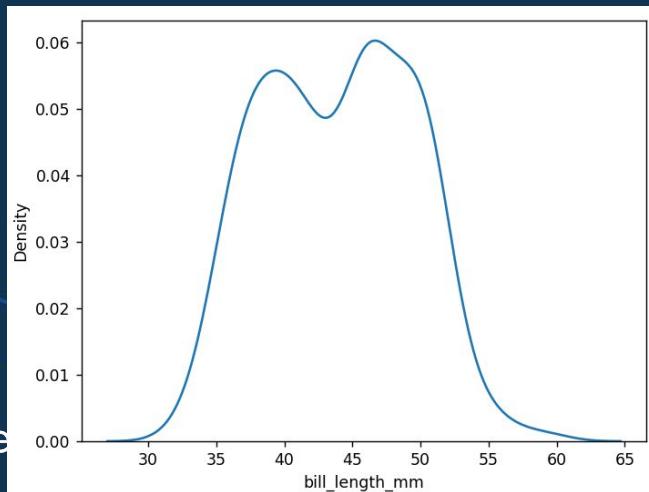
```
#BarPlot  
sns.barplot(x="species", y="body_mass_g",  
             hue="sex", data=peng_df)  
plt.show()
```

```
#Histogram  
sns.histplot(x="bill_length_mm", data=peng_df,  
             bins=20, kde=True, color="green")  
plt.show()
```



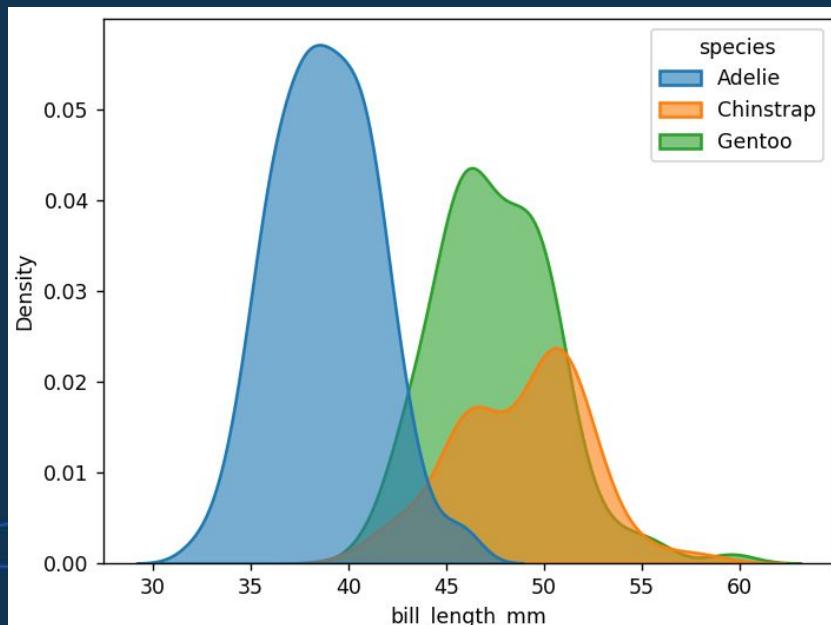
Seaborn plots

```
#Kernel density plot  
sns.kdeplot(data=peng_df, x="bill_length_mm")  
plt.show()  
sns.kdeplot(data=peng_df, x="bill_length_mm",  
             hue="species", fill=True, alpha=0.6,  
             linewidth=1.5)  
  
plt.show()
```



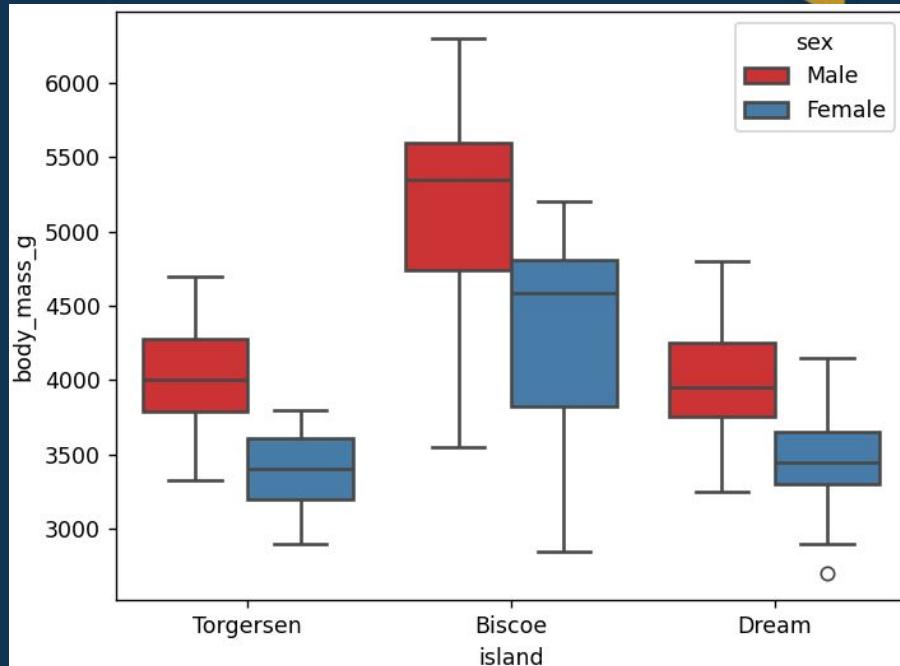
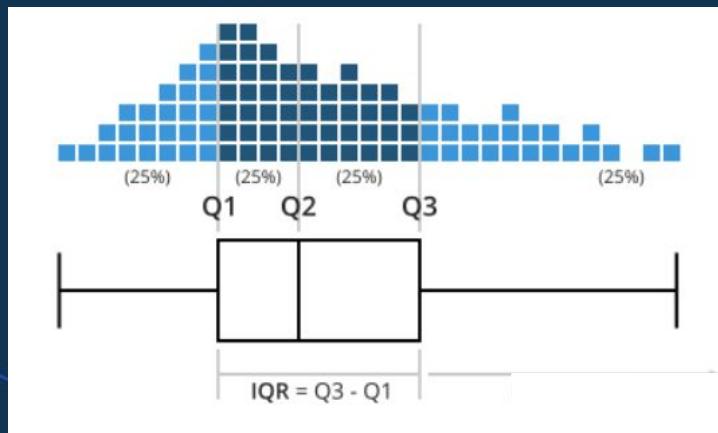
Kernel density plots:

Smooth curves representing density of data points, great for comparing distributions of several groups.



Seaborn plots

Box plot

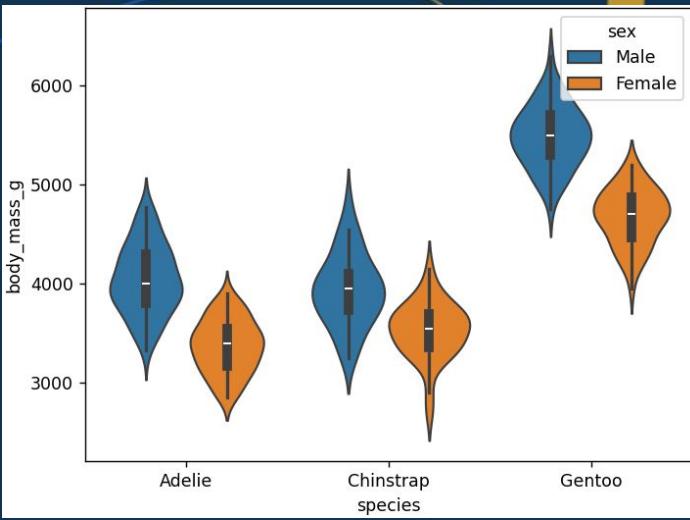


```
#BoxPlot
sns.boxplot(x="island", y="body_mass_g", hue="sex",
             data=peng_df, palette="Set1", Linewidth=1.5)
plt.show()
```

Seaborn plots

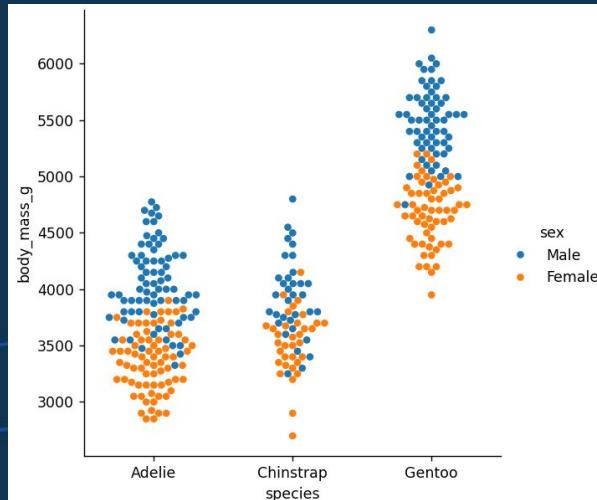
Violin plot: Combine aspects of KDEs and boxplots, ideal for showing density alongside summary statistics.

```
#Violinplot  
sns.violinplot(x="species", y="body_mass_g",  
hue="sex", data=peng_df)  
plt.show()
```



Categorical plot

```
#Categorical plot  
sns.catplot(data=peng_df, kind="swarm",  
x="species", y="body_mass_g", hue="sex")  
plt.show()
```



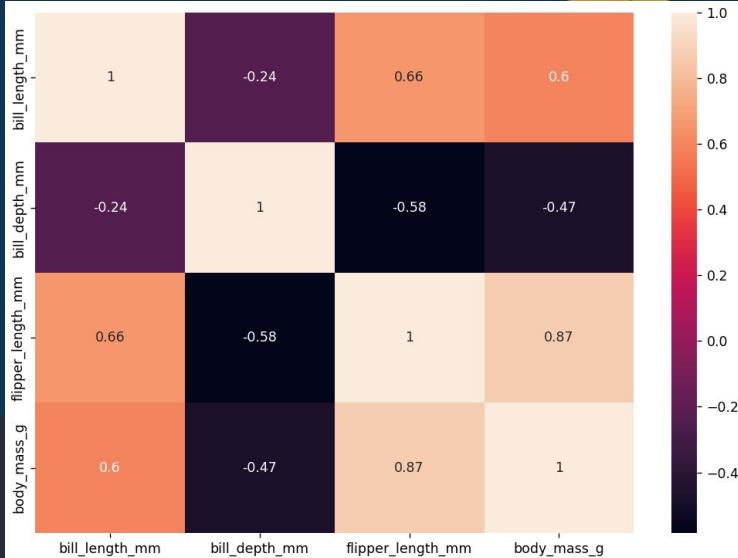
Seaborn plots

Heatmap

Color-coded matrices excellent for **revealing structure, highlighting correlations, and identifying clusters.**

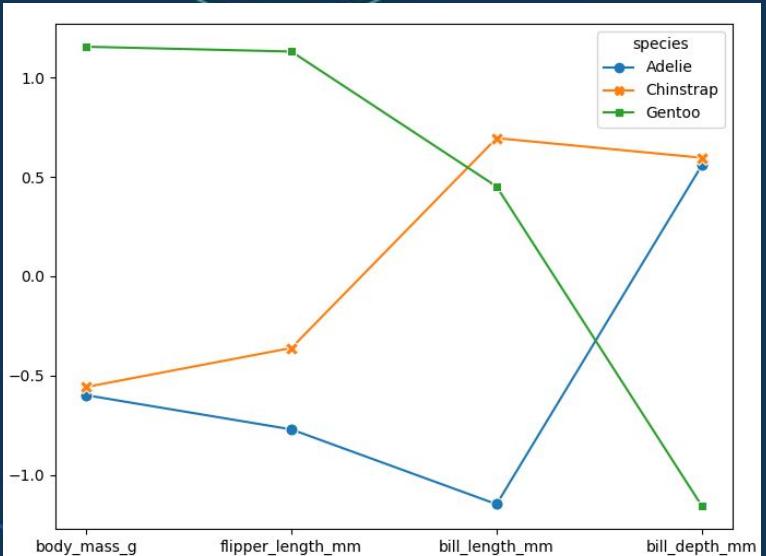
```
#Heatmap
# Create correlation between variables (floats only)
print(peng_df.dtypes)
peng_float = peng_df.select_dtypes(include=[np.float64])
corr = peng_float.corr()

#Plot
plt.figure(figsize=(10,7))
sns.heatmap(corr, annot=True)
plt.show()
```



Customization in heatmaps is key for optimal interpretation.

Parallel coordinates



- ❖ Show many dimensions on the same plot
- ❖ Each feature has a vertical axis, data points become lines crossing them.
- ❖ Ideal when you have many interrelated variables and need to spot outliers or group characteristics.

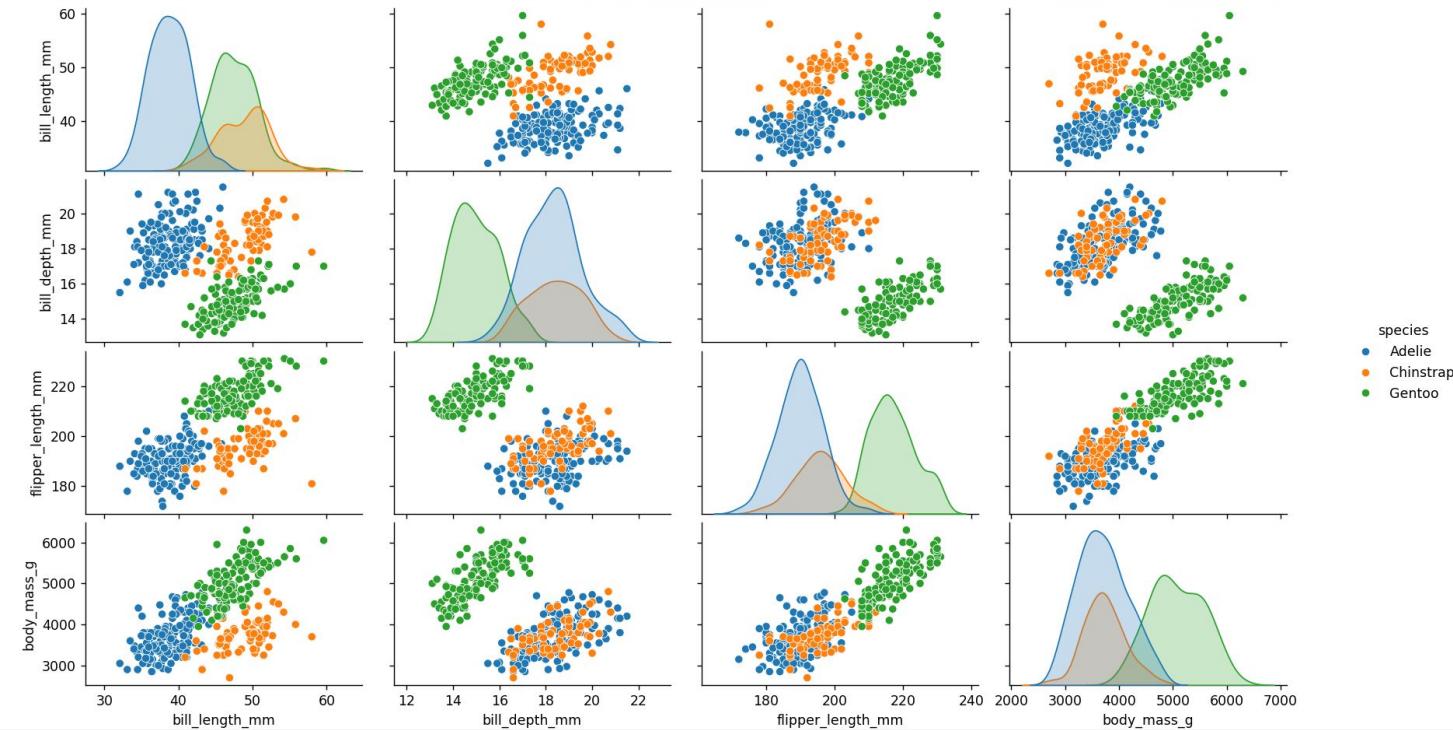
```
#Parallel coordinates
# Calculate the average values for each continent
average_data = peng_df.groupby('species')[['body_mass_g', 'flipper_length_mm',
                                             'bill_length_mm','bill_depth_mm']].mean()

# Normalize the data for better visualization
normalized_data = (average_data - average_data.mean()) / average_data.std()

# Create parallel plot
plt.figure(figsize=(8, 6))
parallel_plot = sns.lineplot(data=normalized_data.transpose(),
                             dashes=False,
                             markers=True,
                             markerSize=8)
plt.show()
```

Seaborn plots

Pairplot



#Pairplot

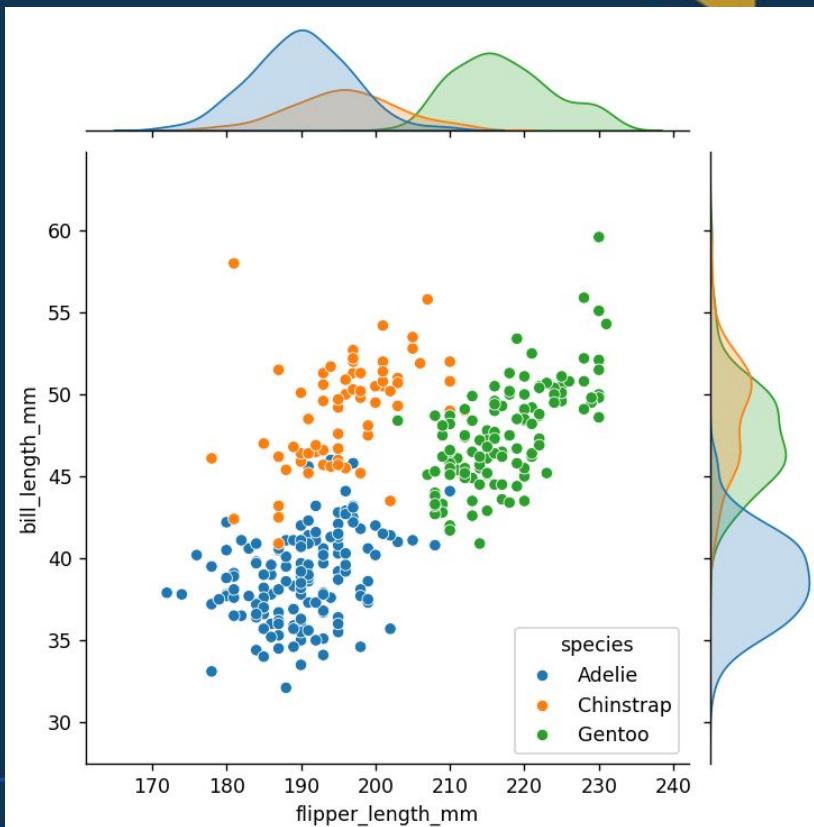
```
sns.pairplot(peng_df, hue="species")  
plt.show()
```

Compact way to depict pairwise relationships between several variables simultaneously.

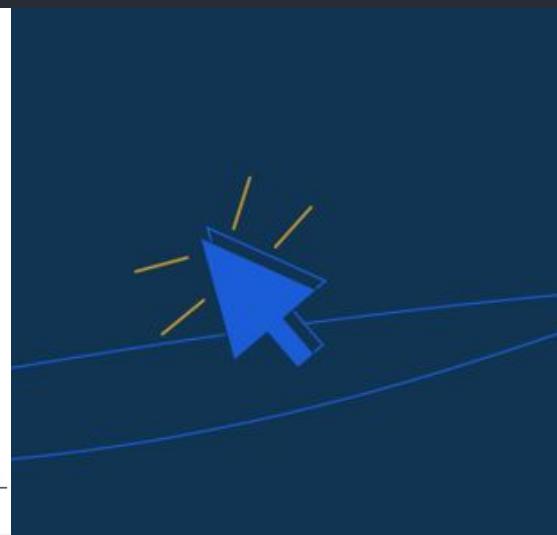
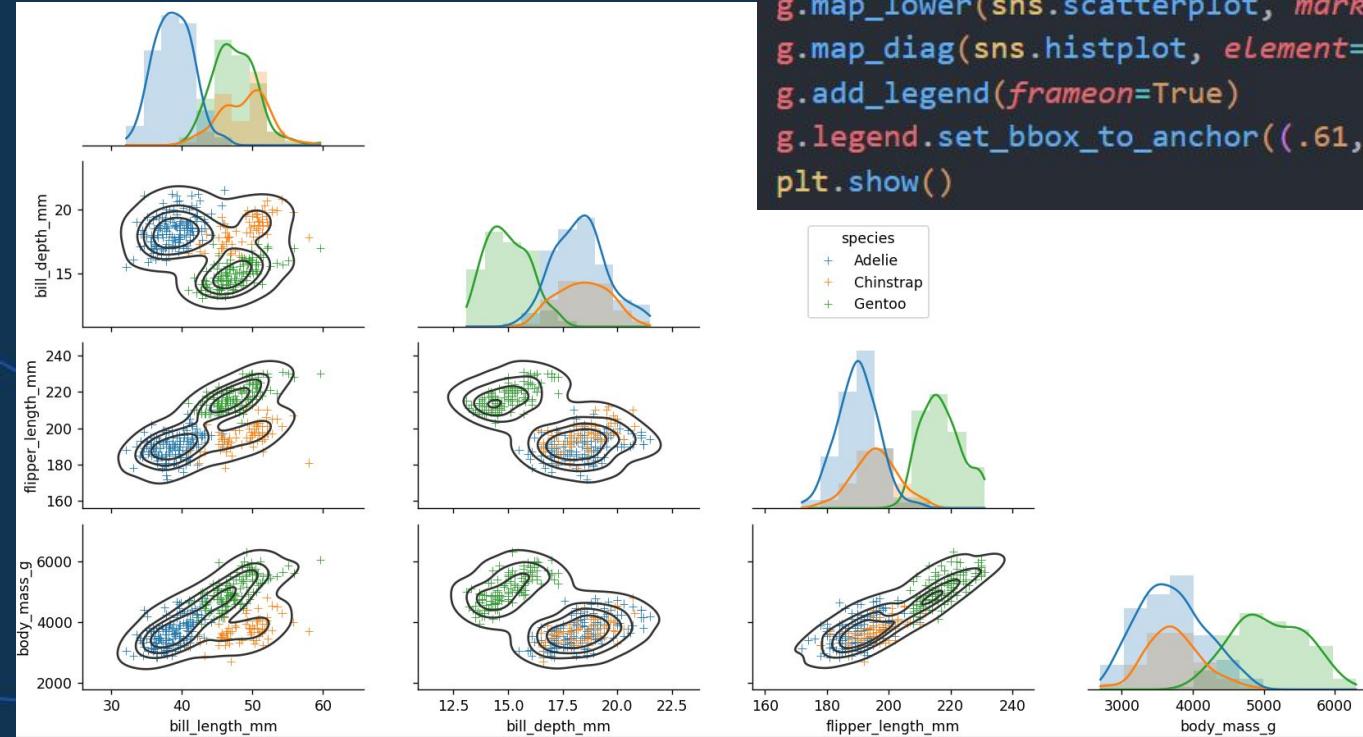
Seaborn plots

Jointplot

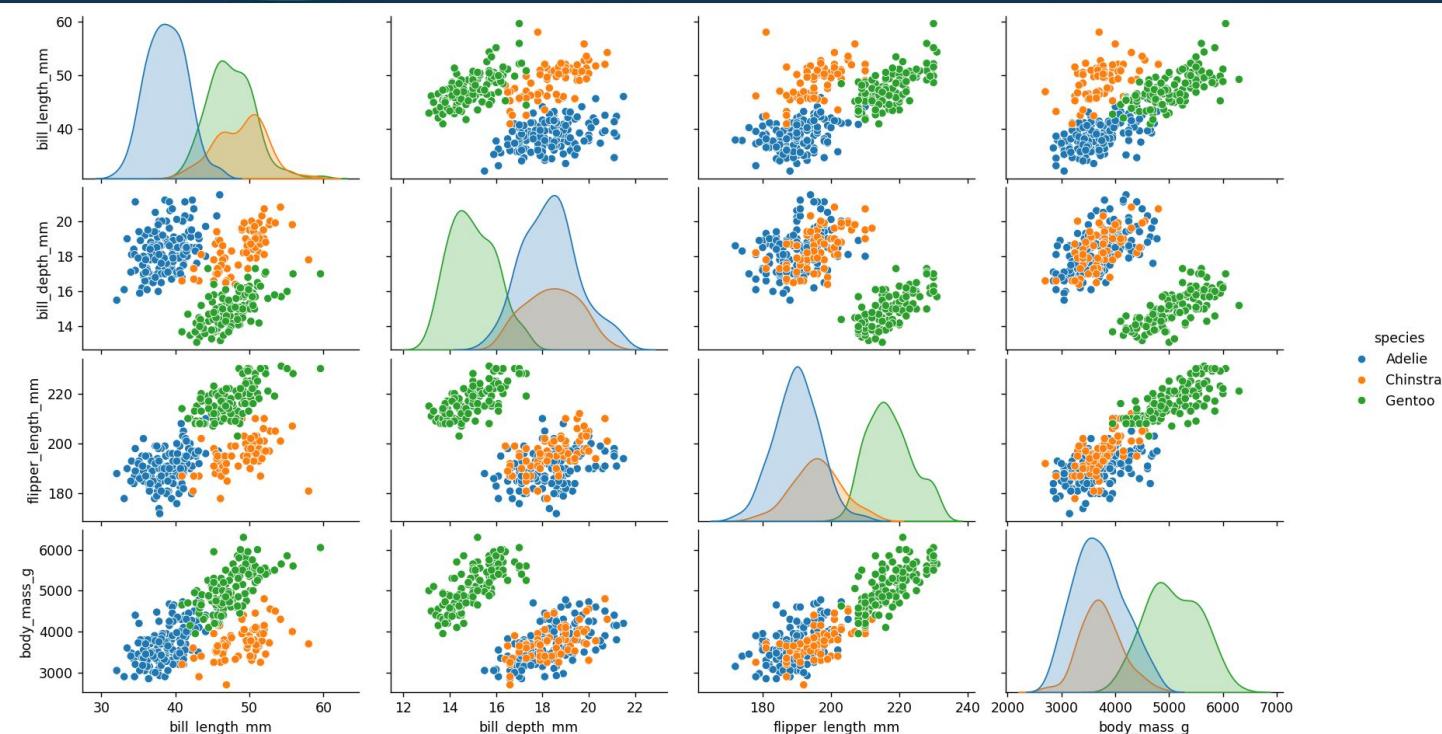
```
#Jointplot  
sns.jointplot(data=peng_df, x="flipper_length_mm",  
                y="bill_length_mm", hue="species")  
plt.show()
```



Combination plots



Visualisation analysis



Gentoo, on average

- ❖ Higher body mass, flipper length
- ❖ Lower bill depth

Adelie has lower bill length on an average



Summary

- ❖ **Visual perception**
 - **Pre-attentive attributes** (color, position, size, shape) are quickly processed by our brains. **Use them deliberately for emphasis.**
 - **Our brains group visual elements automatically**; leverage this fact by using **spatial arrangement, similarity, and clear layout** to reinforce the **key takeaways** of your visualization.
- ❖ **Less is More**
 - **Clear labels**, thoughtful **color choice**, and **minimal clutter** improve the impact of your visual message. Unnecessary embellishment creates a distraction.

Summary

- ❖ **Chart Choice vs. Your Question**
 - Comparing **individual values**? **Bar charts**, or **dot plots**.
 - Analyzing **trends over time**? **Line charts**.
 - Interested in **proportions of a whole**? **Pie charts**, **treemaps**
 - Focus on **relationships between variables**? **Scatter plots** (and their extensions) and **heatmaps**.

Resources

- ❖ Matplotlib:
https://matplotlib.org/stable/gallery/color/named_colors.html
https://matplotlib.org/stable/api/pyplot_summary.html
- ❖ Seaborn examples:
<https://seaborn.pydata.org/examples/index.html>
- ❖ If SSL error while getting Seaborn built-in data, download the data csv file from here - <https://github.com/mwaskom/seaborn-data>
- ❖ Resources to be installed for tutorial: **Python, NumPy, Pandas, Jupyter Notebook, Matplotlib, Seaborn**

Questions and Answers



Thank you for attending

