Welcome to the CoGrammar Data Visualisation

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

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. **(Fundamental British Values: Mutual Respect and Tolerance)**
- 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**

Skills Bootcamp
8-Week Progression Overview

Fulfil 4 Criteria to Graduation

✅ Criterion 1: Initial Requirements

Timeframe: First 2 Weeks
Guided Learning Hours (GLH): Minimum of 15 hours
Task Completion: First four tasks

Due Date: 24 March 2024

✅ Criterion 2: Mid-Course Progress

60 Guided Learning Hours

Data Science - **13 tasks**
Software Engineering - **13 tasks**
Web Development - **13 tasks**

Due Date: 28 April 2024

CoGrammar

Skills Bootcamp Progression Overview

✅ Criterion 3: Course Progress

Completion: All mandatory tasks, including Build Your Brand and resubmissions by study period end
Interview Invitation: Within 4 weeks post-course
Guided Learning Hours: Minimum of 112 hours by support end date (10.5 hours average, each week)

✅ Criterion 4: Demonstrating Employability

Final Job or Apprenticeship Outcome: Document within 12 weeks post-graduation
Relevance: Progression to employment or related opportunity


SKILLS FOR LIFE
SKILLS BOOTCAMPS
Department for Education
CoGrammar
Data Visualisation
April 2024

Learning objectives

- Importance of data visualisation
- Using the Matplotlib library
- Using the Seaborn library
- Analyse and interpret visualisations

Data Visualisation

CoGrammar

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

CoGrammar

Types of Data

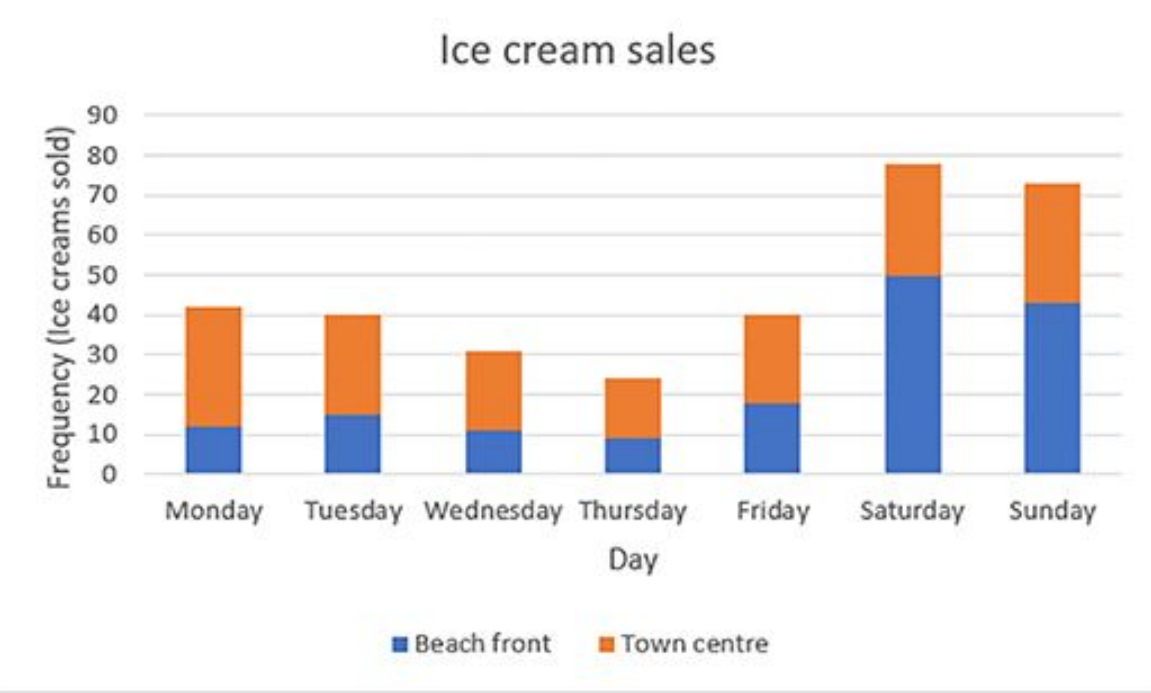
- **Discrete data:** Can take specific values, with an infinite range, e.g : [1, 2, 3, 4, 5, ...]
- **Categorical data:** Can take specific values, with a limited range, e.g. [Dog, Cat, Hamster, Fish]
- **Continuous data:** Available values in a spectrum, there are an infinite number of values, e.g.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

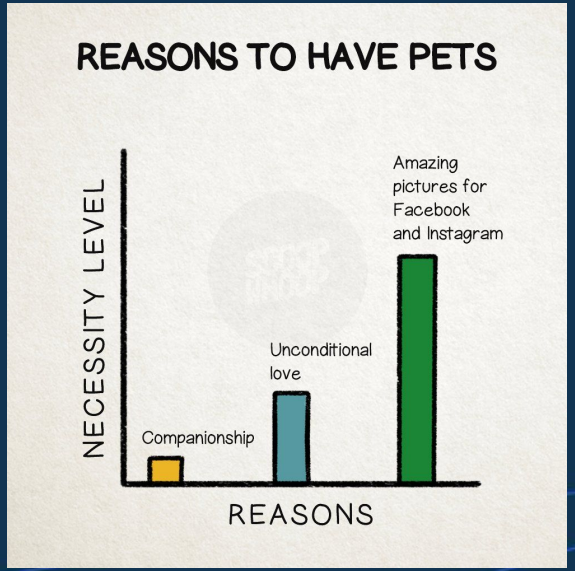
Basic Visualisations

CoGrammar

Barplot (Bar chart)

Good for plotting Categorical vs. Continuous /Discrete data

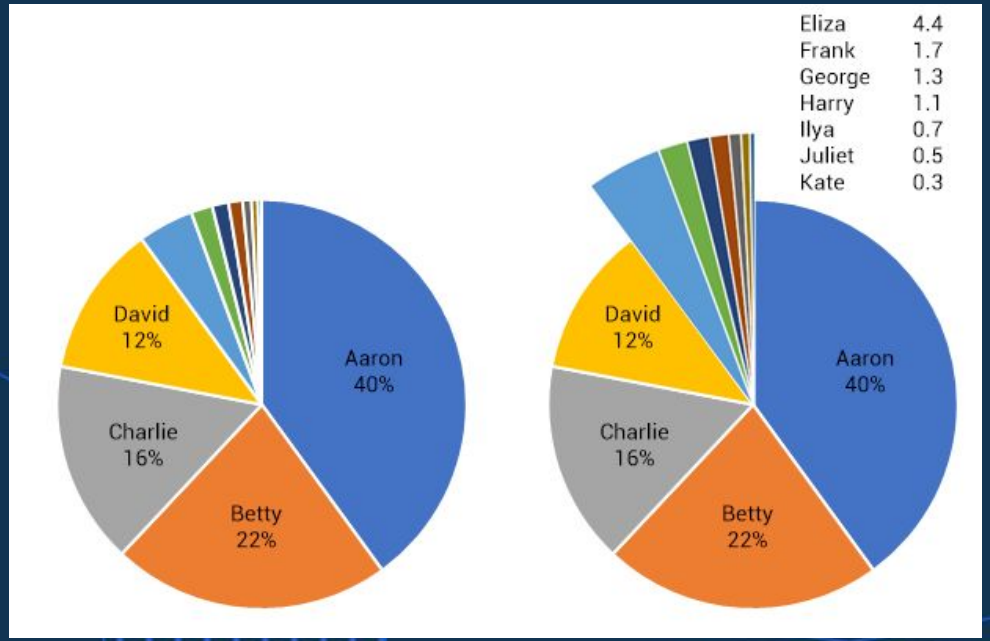


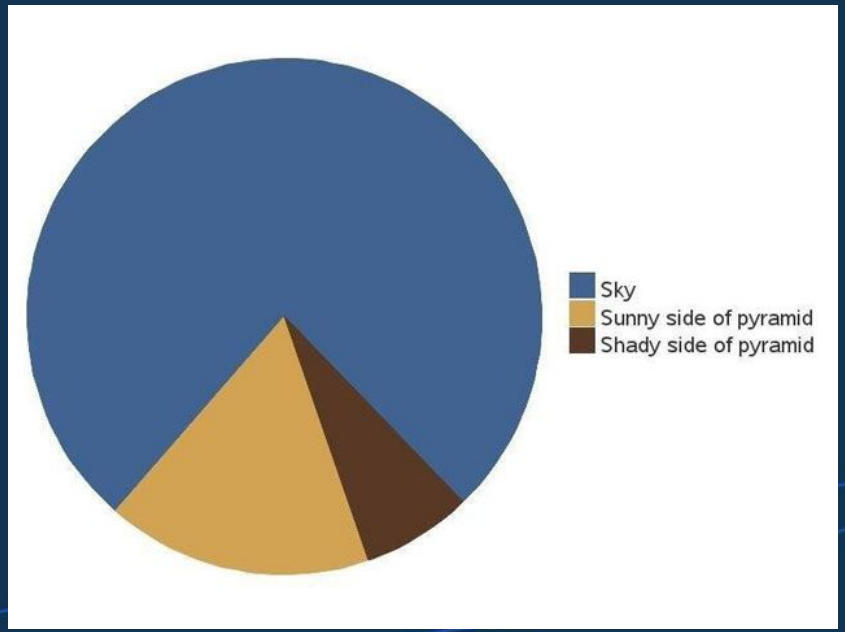


CoGrammar

Pie chart

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





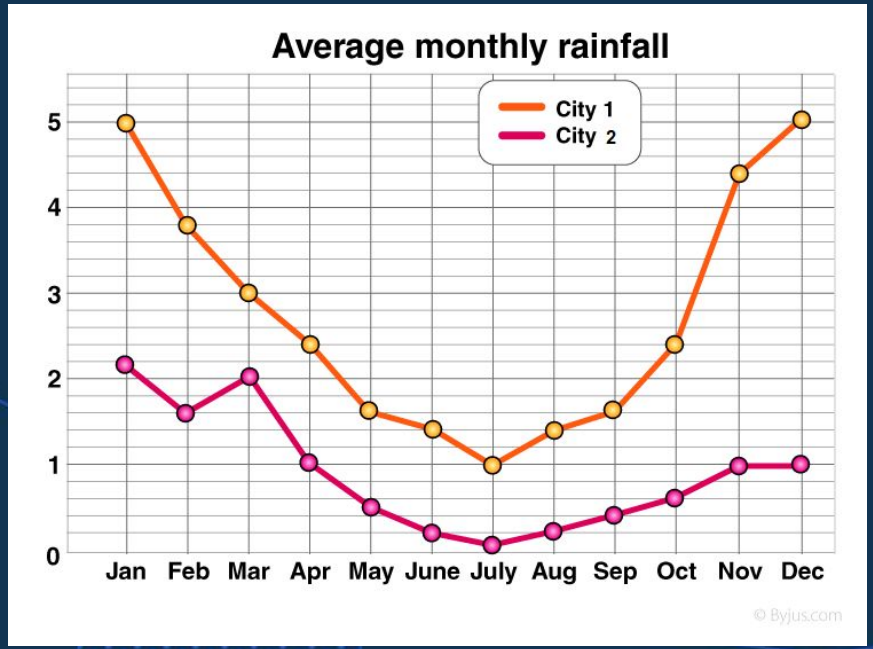
Scatterplots

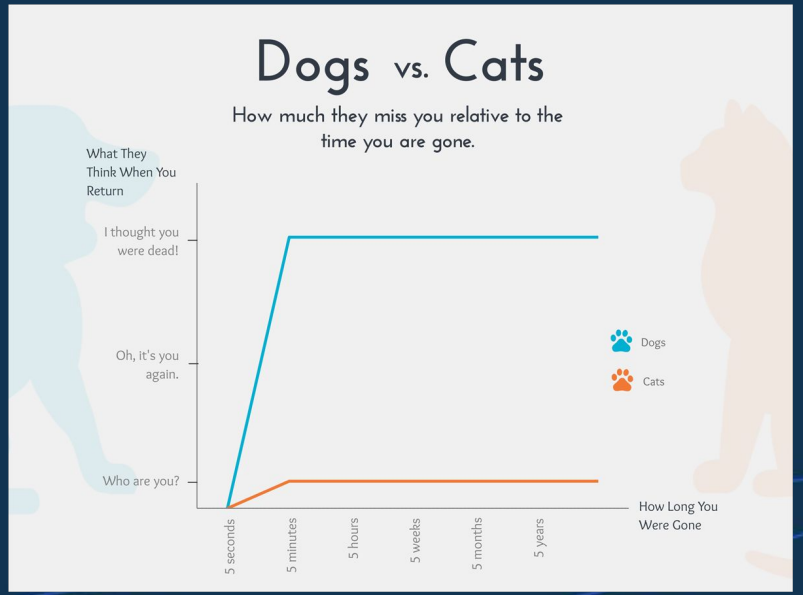
Great for plotting Discrete vs. Discrete data or Continuous vs. Continuous data. Useful for finding relationships between variables.

CoGrammar

Line graphs

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





CoGrammar

Matplotlib

CoGrammar

Matplotlib

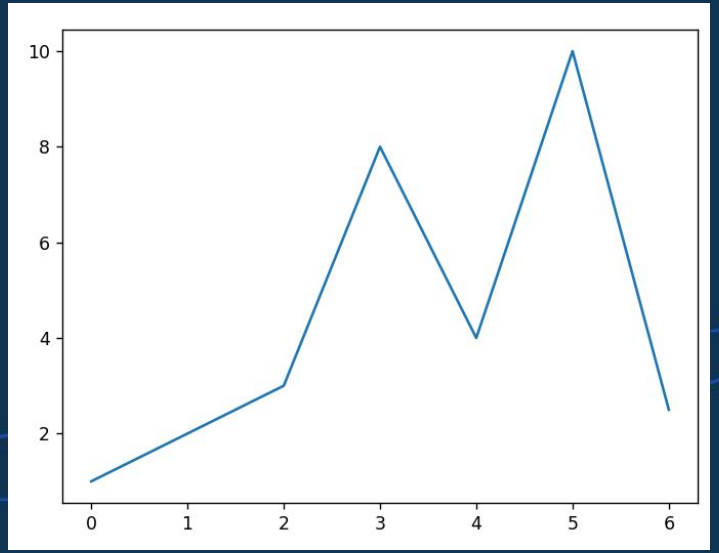
Installation

```
pip install matplotlib
```

Importing

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

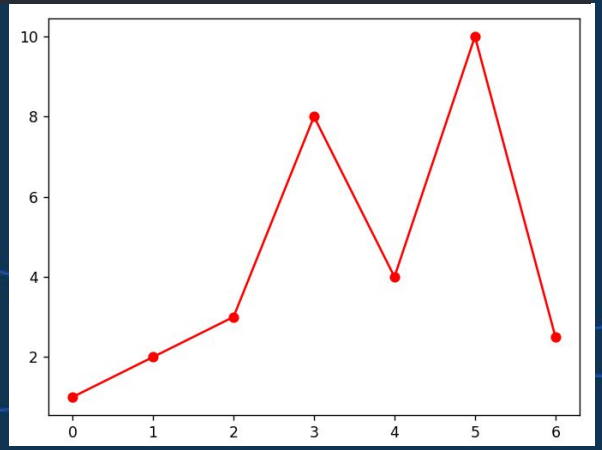
```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([1, 2, 3, 8, 4, 10, 2.5])

plt.plot(ypoints)
plt.show()
```



CoGrammar

Matplotlib

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

ypoints = np.array([1, 2, 3, 8, 4, 10, 2.5])

plt.plot(ypoints, 'o-r')

plt.show()
```



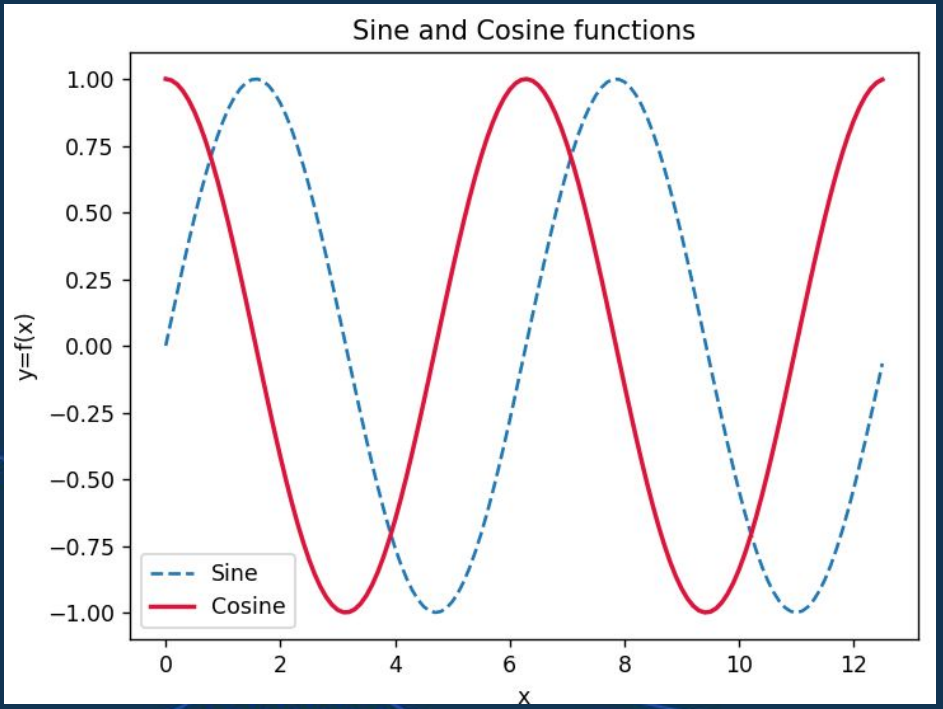
Markers
o = Circle, * = Star,
. = Point, x = Cross,
s = Square, d = diamond

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

CoGrammar

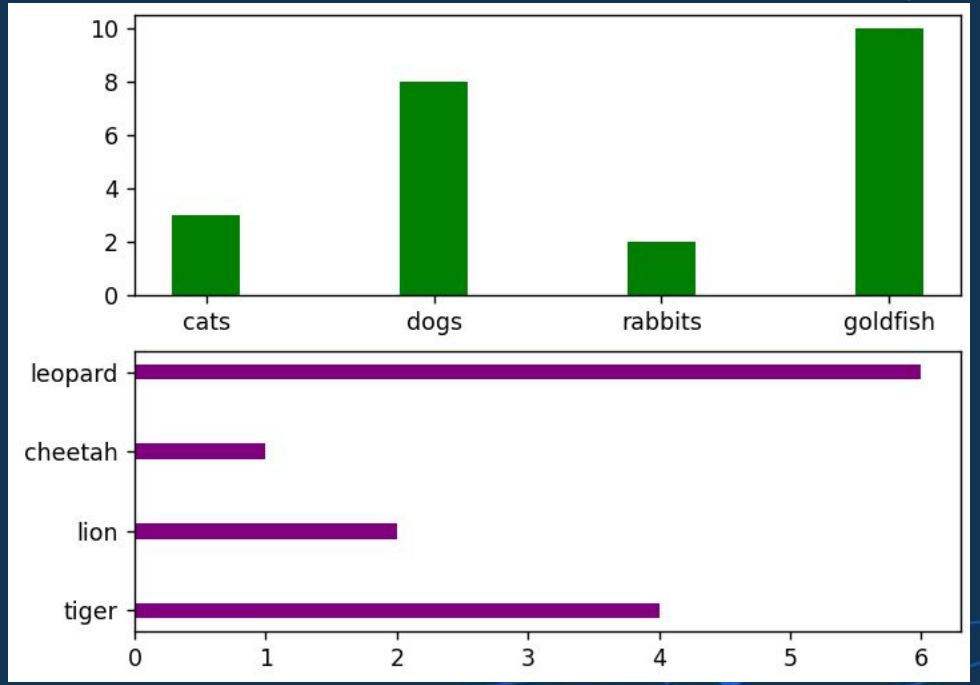
Matplotlib




```python
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')
```

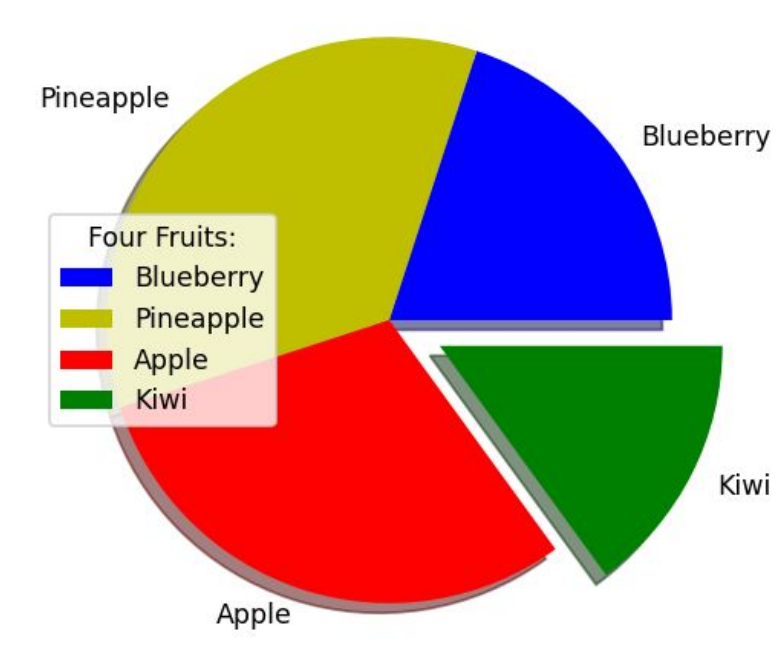
Matplotlib: Barplot

```
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.barh(x2, y2, color = '#800080', height=0.2)
plt.show()
```



CoGrammar

Matplotlib: Pie chart



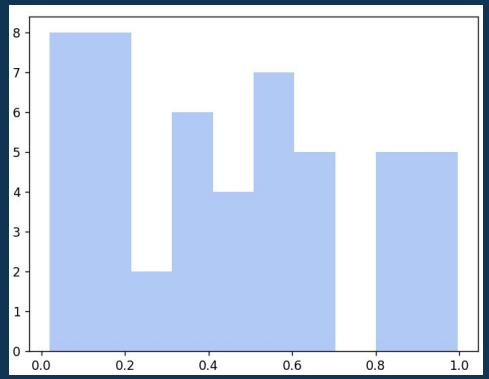

```python
import matplotlib.pyplot as plt
import numpy as np

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()
```

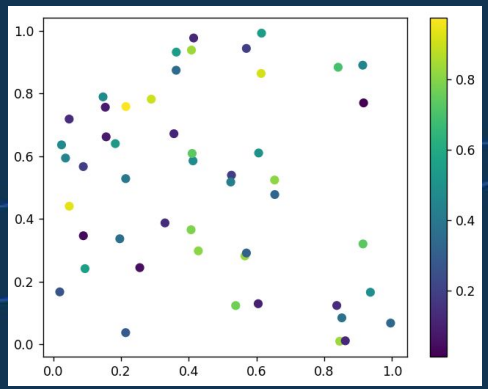
CoGrammar

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()
```





CoGrammar

Advanced Visualisations

CoGrammar

Why advanced visualisation?

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

Seaborn

CoGrammar

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`

Importing `import seaborn as sns`

```
# 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()
```

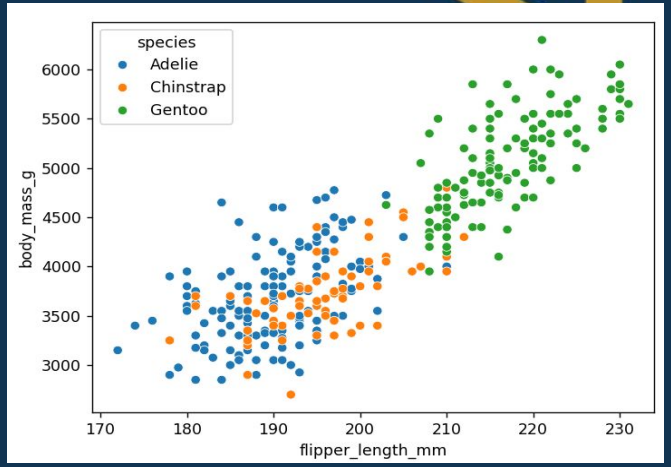
- load_dataset() is like pd.read_csv()

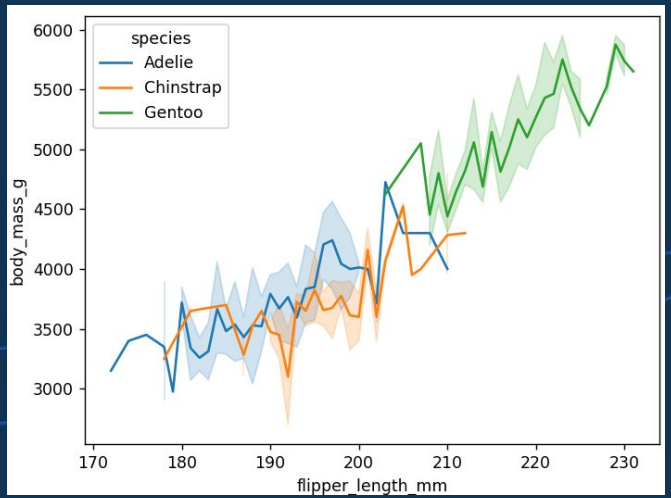
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   species            344 non-null    object
 1   island             344 non-null    object
 2   bill_length_mm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





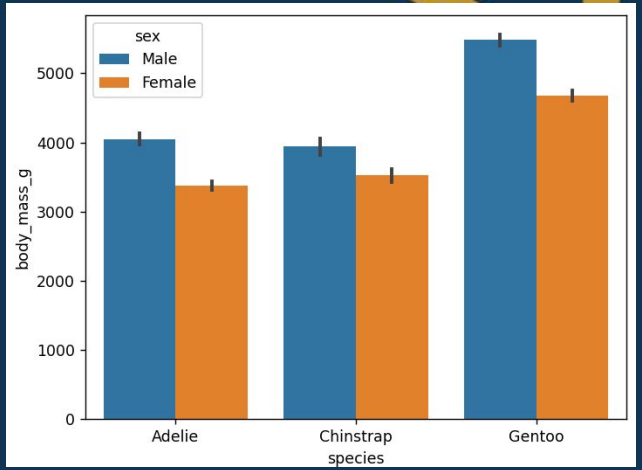
- If not in Jupyter or IPython notebook, explicitly call matplotlib.pyplot for displaying the plot

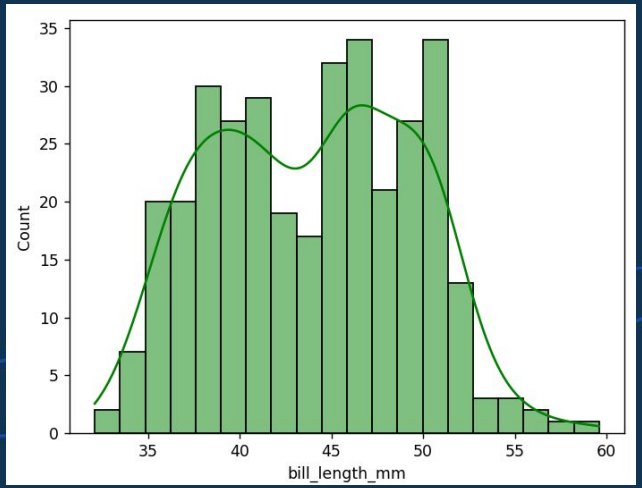
CoGrammar

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()
```





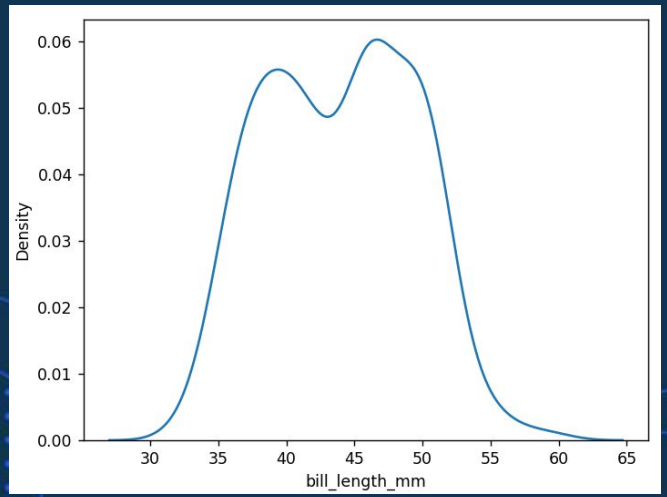
CoGrammar

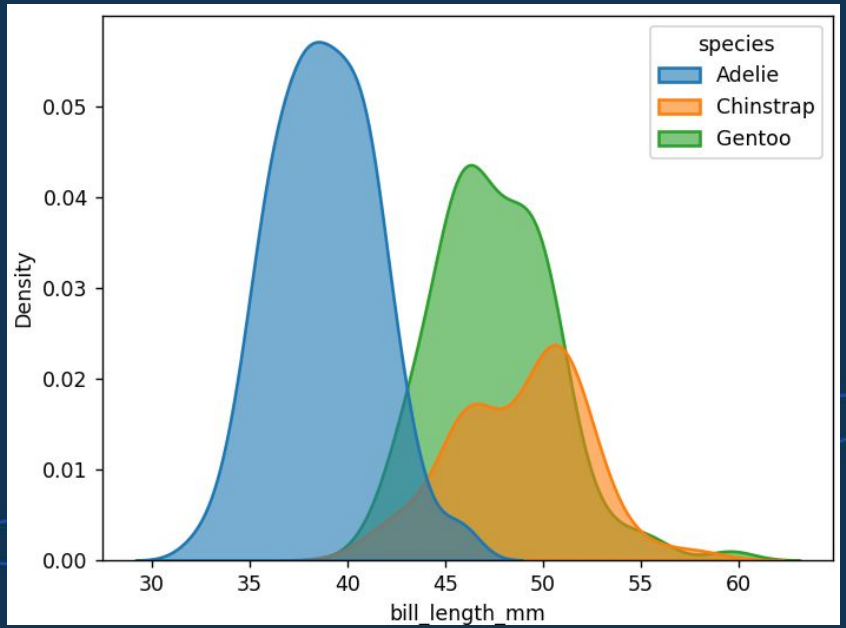
Seaborn plots

```
#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.

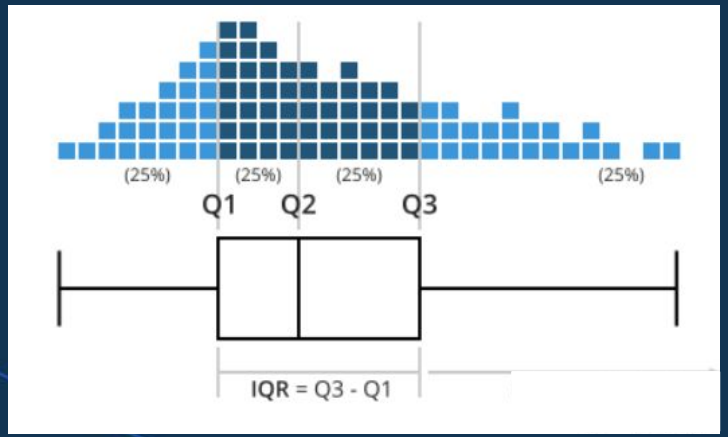


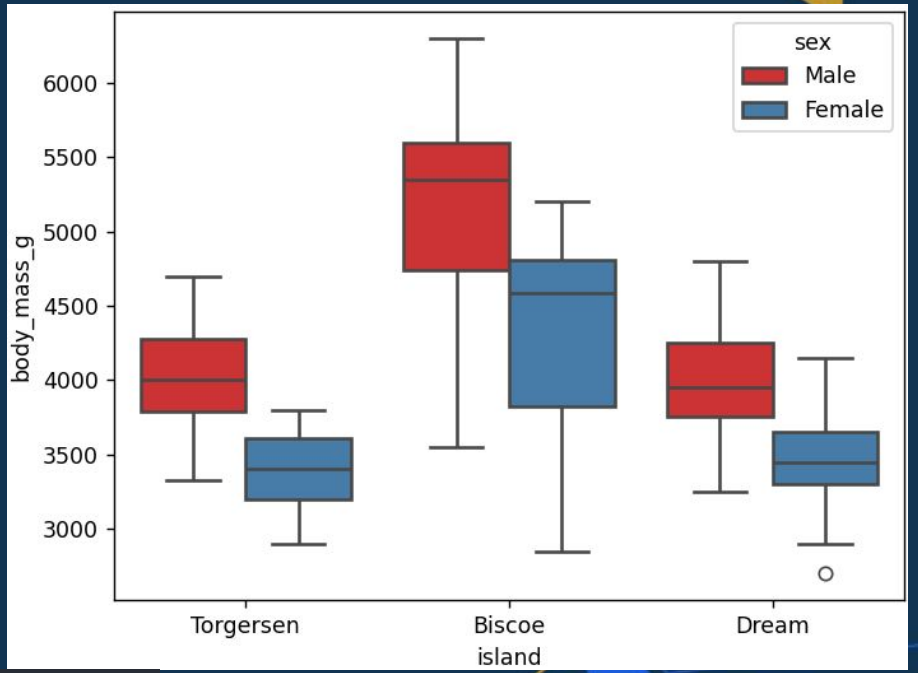


CoGrammar

Seaborn plots

Boxplot



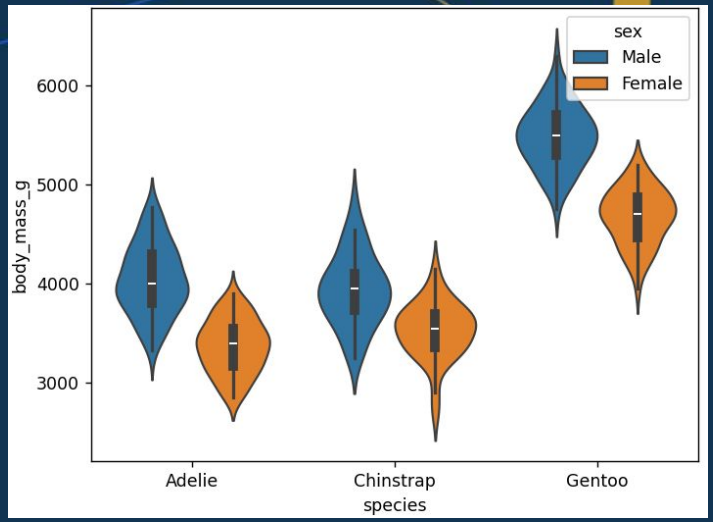



```
#Boxplot
sns.boxplot(x="island", y="body_mass_g", hue="sex",
            data=peng_df, palette="Set1", linewidth=1.5)
plt.show()
```

Seaborn plots

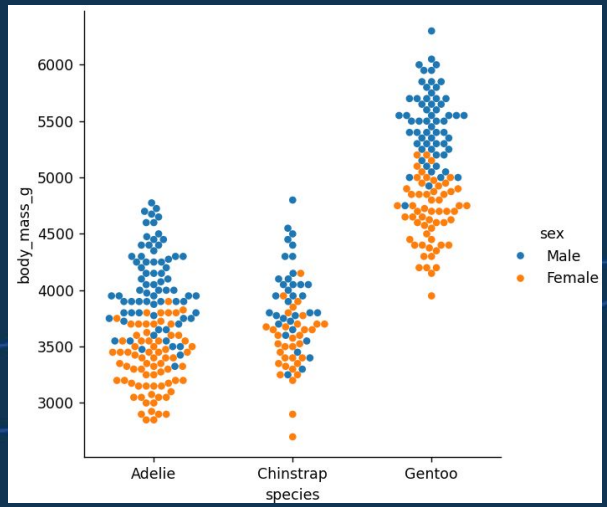
Violinplot: Combine aspects of KDEs and boxplots, ideal for showing density alongside summary statistics.

```
#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()
```

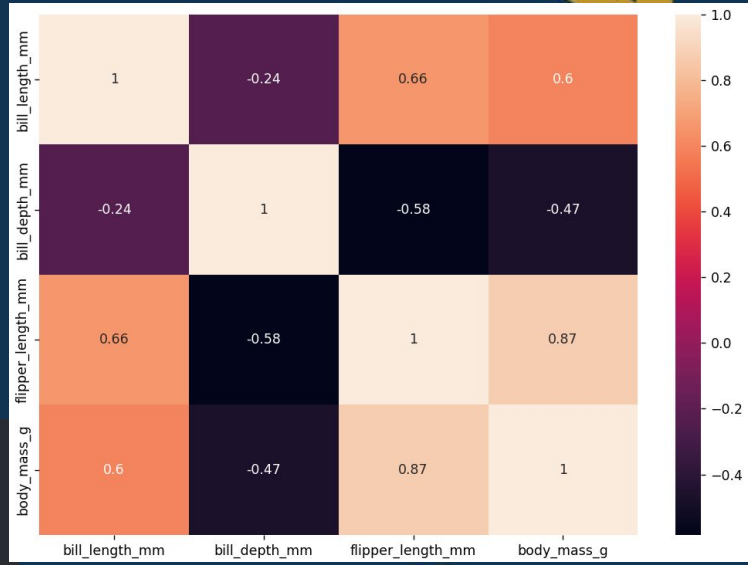


Seaborn plots

Heatmap

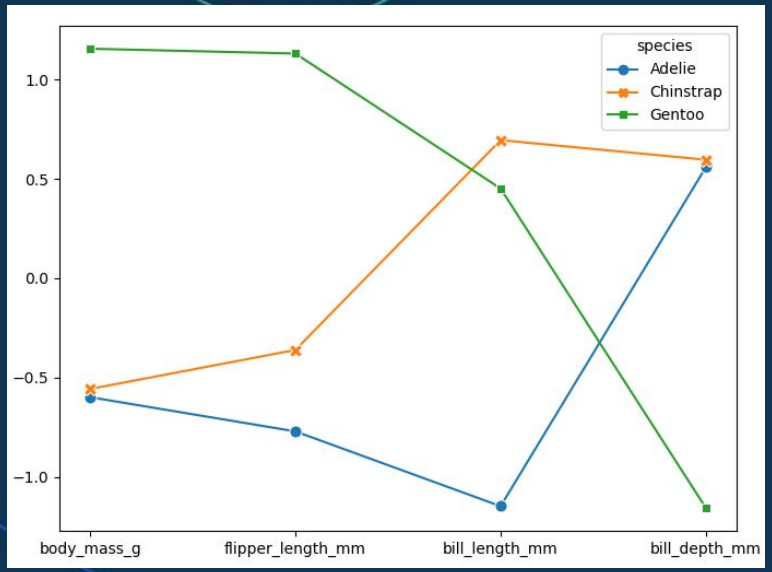
Color-coded matrices excellent for **revealing structure, highlighting correlations, and identifying clusters**.

```
#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



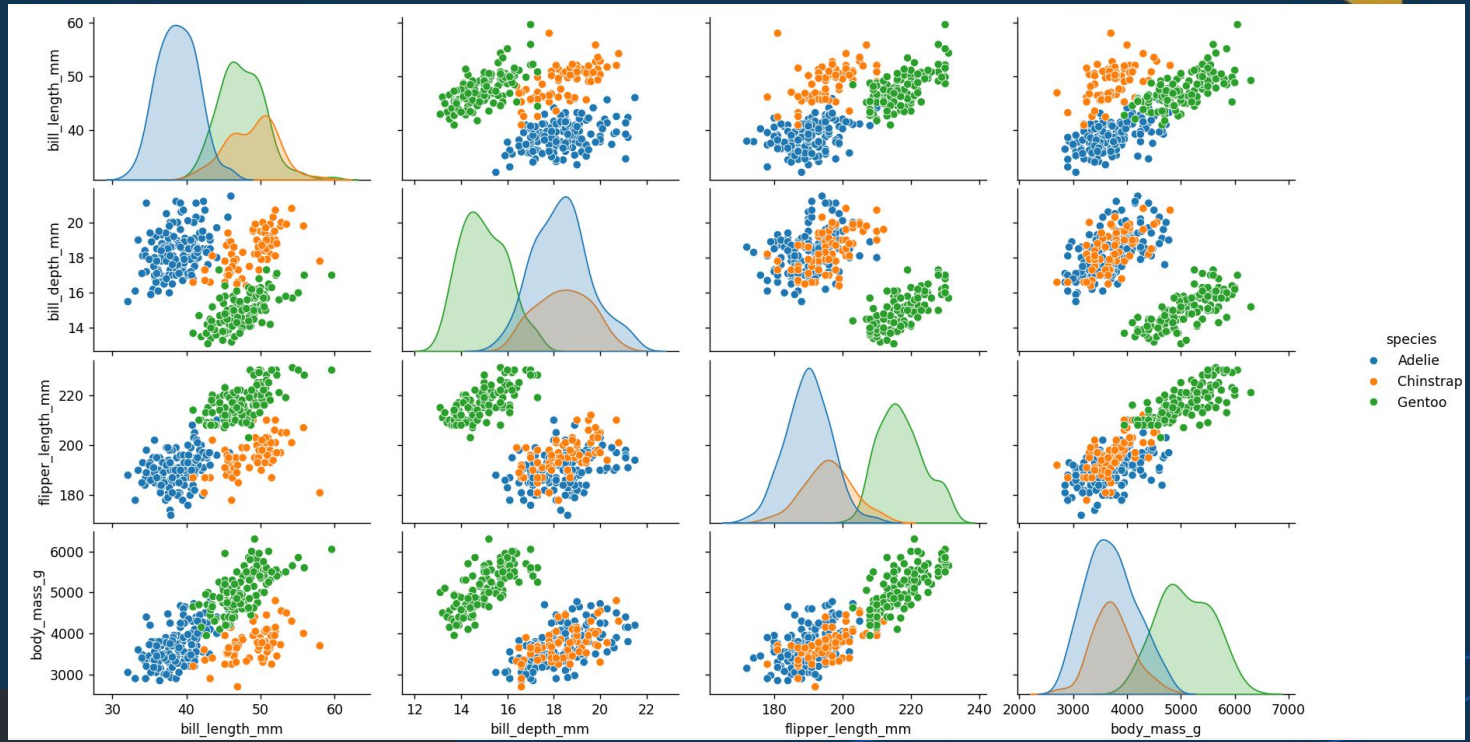
- Show many dimensions on the same plot
- Each feature has a vertical axis, data points become lines crossing them.
- Ideal when you have many interrelated variables and need to spot outliers or group characteristics.

```
#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()
```

CoGrammar

Seaborn plots

Pairplot



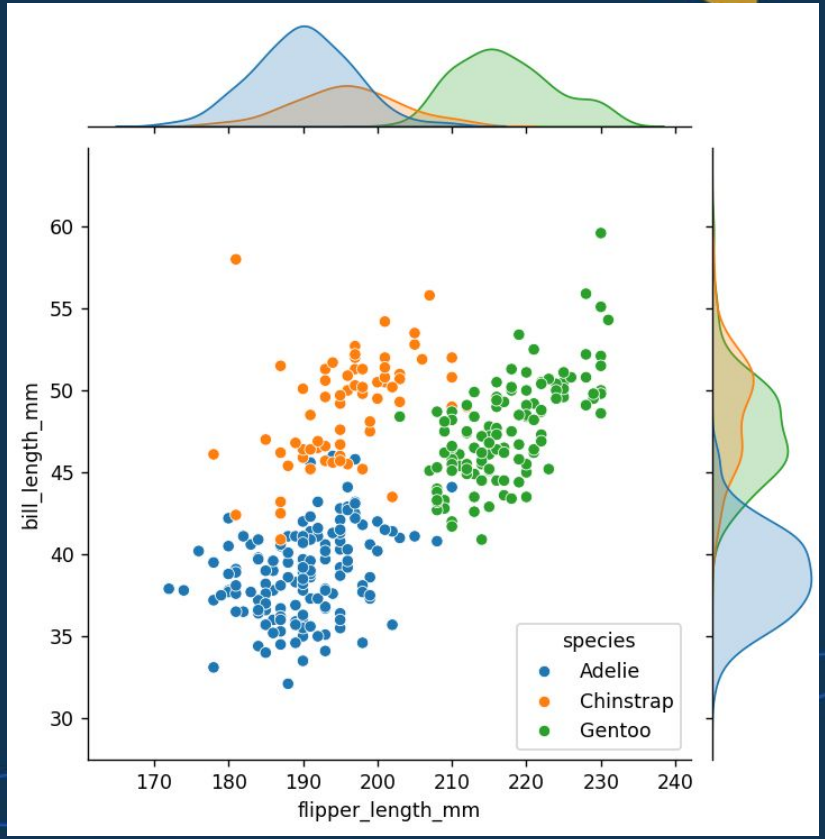

```
#Pairplot
sns.pairplot(peng_df, hue="species")
plt.show()
```

Compact way to depict pairwise relationships between several variables simultaneously.

Seaborn plots

Jointplot

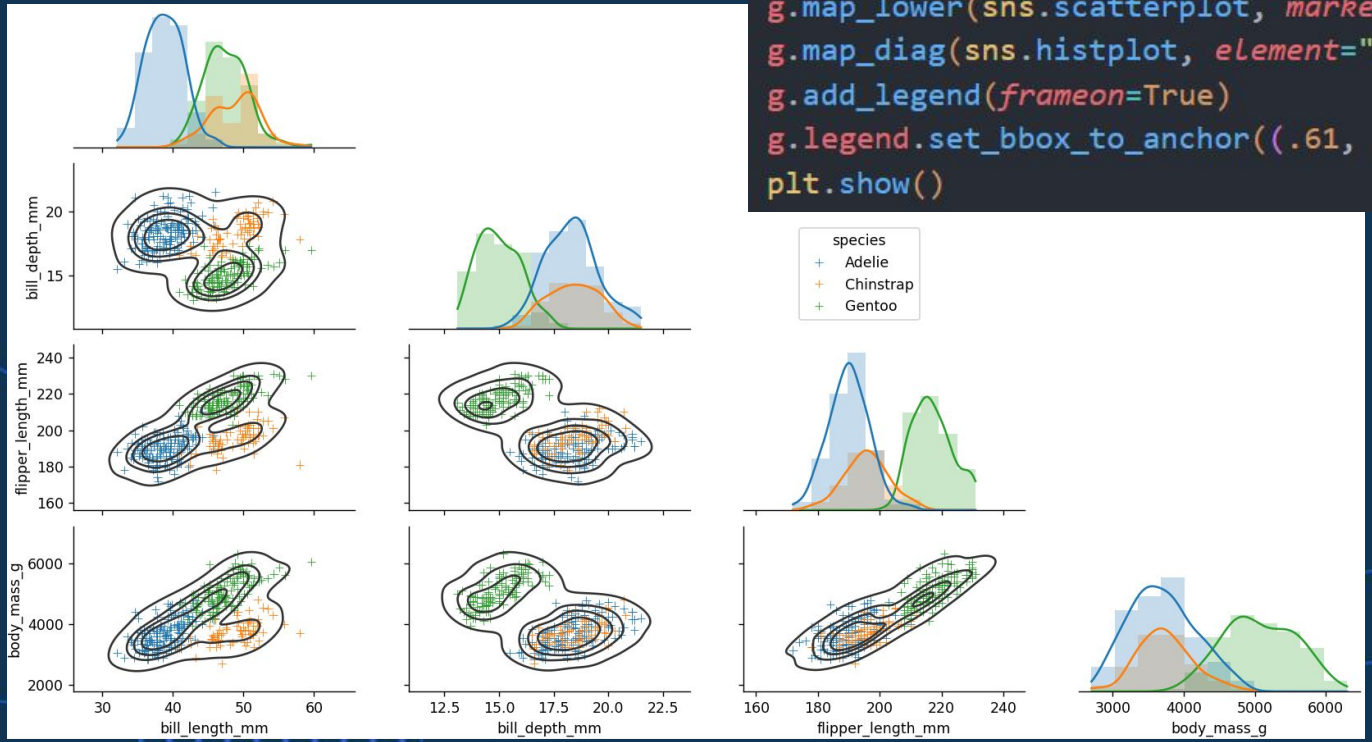
```
#Jointplot
sns.jointplot(data=peng_df, x="flipper_length_mm",
              y="bill_length_mm", hue="species")
plt.show()
```



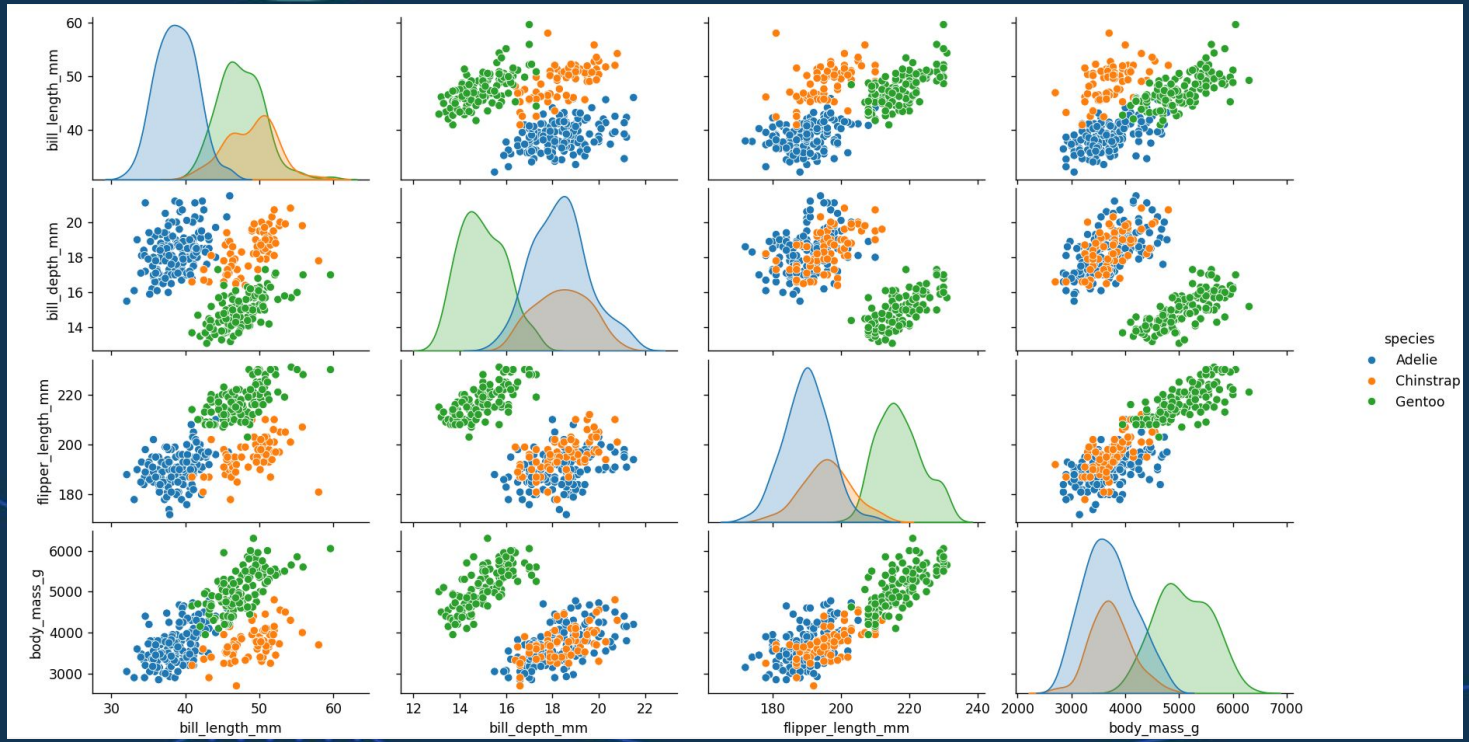
CoGrammar

Combination plots

```python
#Combination plots
g = sns.PairGrid(peng_df, hue="species", corner=True)
g.map_lower(sns.kdeplot, hue=None, levels=5, color=".2")
g.map_lower(sns.scatterplot, marker="+")
g.map_diag(sns.histplot, element="step", linewidth=0, kde=True)
g.add_legend(frameon=True)
g.legend.set_bbox_to_anchor((.61, .6))
plt.show()
```



Visualisation analysis



Gentoo, on average

- Higher body mass, flipper length
- Lower bill depth

Adelie has lower bill length on an average

CoGrammar

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**

Which type of plot is ideal for comparing distributions of several groups using smooth curves representing density of data points?

1. Scatterplot
2. Violin plot
3. Heatmap
4. Kernel density plot

Which type of plot is ideal for comparing distributions of several groups using smooth curves representing density of data points?

1. Scatterplot
2. Violin plot
3. Heatmap
4. **Kernel density plot**

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar