Week 3: Data Visualizations

Dr Giuseppe Brandi

Northeastern University London

Outline

- Definition of Data Visualization
- Why do we need visualization?
- Scientific vs Information
- Data Visualization in AI and ML
- Data Visualization Process
- Principles of Data Visualization
- Data Visualization with Python
- Conclusion

Definition of Data Visualization

"**Data visualisation** is the representation of data through use of **common graphics**, such as **charts, plots, infographics, and even animations.** These visual displays of information communicate **complex data relationships and data-driven insights** in a way that is easy to understand.

Data visualisation can be utilised for **a variety of purposes**, and it's important to note that it is not only reserved for use by data teams. Management also leverages it to convey **organisational structure and hierarchy** while data analysts and data scientists use it to **discover and explain patterns and trends**." (IBM.com, 2021)

Four main purposes of data visualisation: **(a) idea generation, (b) idea illustration, (c) visual discovery and (d) everyday data visualisation.**

Data Visualisation and Big Data

The volume of data in modern systems is increasing at an unprecedented rate, necessitating advanced data visualization techniques to handle the complexity and scale.

Implementing effective data visualisation solutions for Big Data requires consideration of various factors:

- Real-time changes
- Extreme variety of the sources
- Different levels of data structuring

Simultaneous usage of several visualisation techniques to better illustrate relationships among a large amount of data.

Why do we need visualization?

- Idea generation
- Idea illustration
- Visual discovery
- Everyday data visualisation

Data visualisation serves not just data scientists but is also utilized extensively by management to structure and present data and organizational hierarchies. It plays a critical role in decision-making processes and in explaining patterns and trends that are not immediately obvious.

Scientific vs Information Visualisation

Visualisation consists of converting raw data to a form that is viewable and understandable to humans. There are two main types:

- **Scientific Visualisation:** Specifically concerned with data that has a well-defined representation in 2D or 3D space (e.g., from simulation mesh or scanner).
- **Information Visualisation:** Concerned with data that does not have a well-defined representation in 2D or 3D space (i.e., "abstract data").

Data Visualisation in AI and ML

Data visualisation is an important skill in applied statistics and machine learning:

- Statistics focus on quantitative descriptions and estimations of data. Data visualisation provides an important suite of tools for gaining a qualitative understanding.
- Helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more.
- With a little domain knowledge, data visualisations can be used to express and demonstrate key relationships in plots and charts.

Key Plots for Basic Data Visualisation

Understanding these five key plots is crucial for basic data visualization in AI and ML:

- **Line Plot:** Useful for showing trends over time.
- **Bar Chart:** Effective for comparing quantities among different groups.
- **Histogram Plot:** Ideal for depicting the distribution of data.
- **Box and Whisker Plot:** Helps visualize the distribution of data in terms of quartiles and outliers.
- **Scatter Plot:** Great for observing relationships and distributions between two variables.

With a knowledge of these plots, you can quickly get a qualitative understanding of most data that you come across.

Data Visualisation: Process

The visualisation process involves several key steps:

- **Filtering and Processing:** Transforming raw data into refined information suitable for analysis.
- **Translation and Visual Representation:** Creating visual representations that convey the data's narrative to the audience effectively.
- **Visualisation and Interpretation:** Ensuring the visualisations make a cognitive impact, aiding in knowledge construction and decision-making.

Principles of Data Visualisation

The purpose of seeing, *aka visualising*, the data is to help us understand something they do not easily reveal.

It is also a way of telling stories and research results as a data analysis and testing platform. Help us to create meaning as well as easy-to-remember reports, infographics, and dashboards.

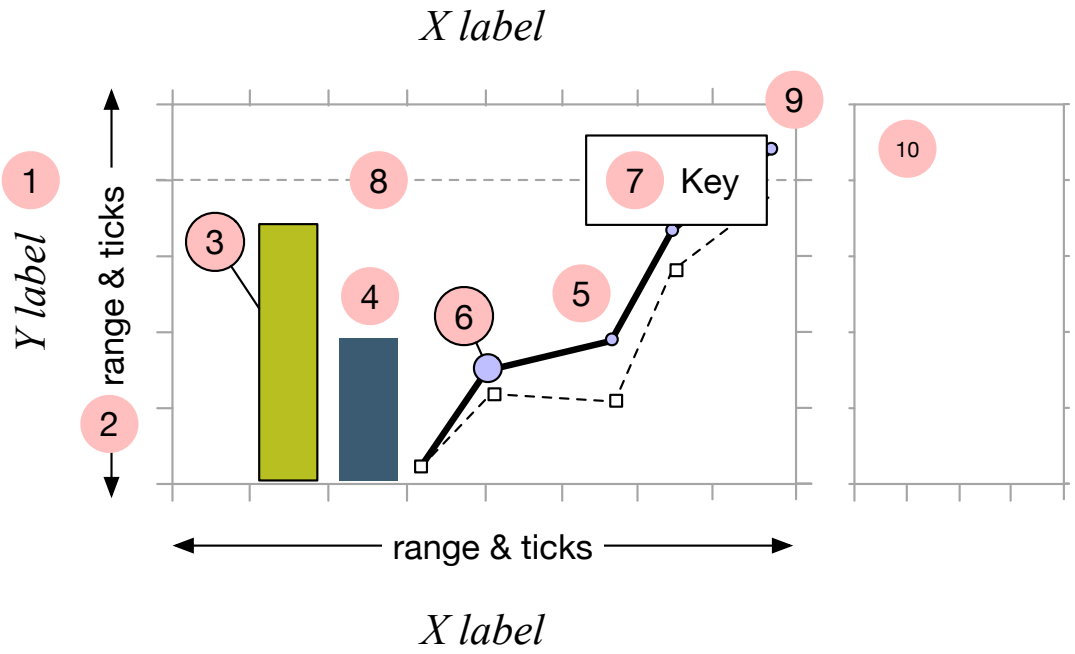
Creating the right perspective helps us to solve problems and analyse subject material in detail.

Principles of Data Visualisation

Key principles include:

- **Preview:** Viewers have more data comprehension, as their starting point for checking. The first step involves giving to the users a visual summary of different types of data, describing their relationship at the same time. This strategy helps us to visualise the process of data, in all its different levels, simultaneously.
- **Zoom and Filter:** The second step involves inserting the first so that viewers can understand the data basement. Zoom in / out enables us to select available data subsets that meet certain methods while maintaining the concept of position and context.
- **Highly Needed Data:** Third step makes it possible to choose a small subset of data, enabling the user to participate with information and apply filters by hovering or clicking data for more details.

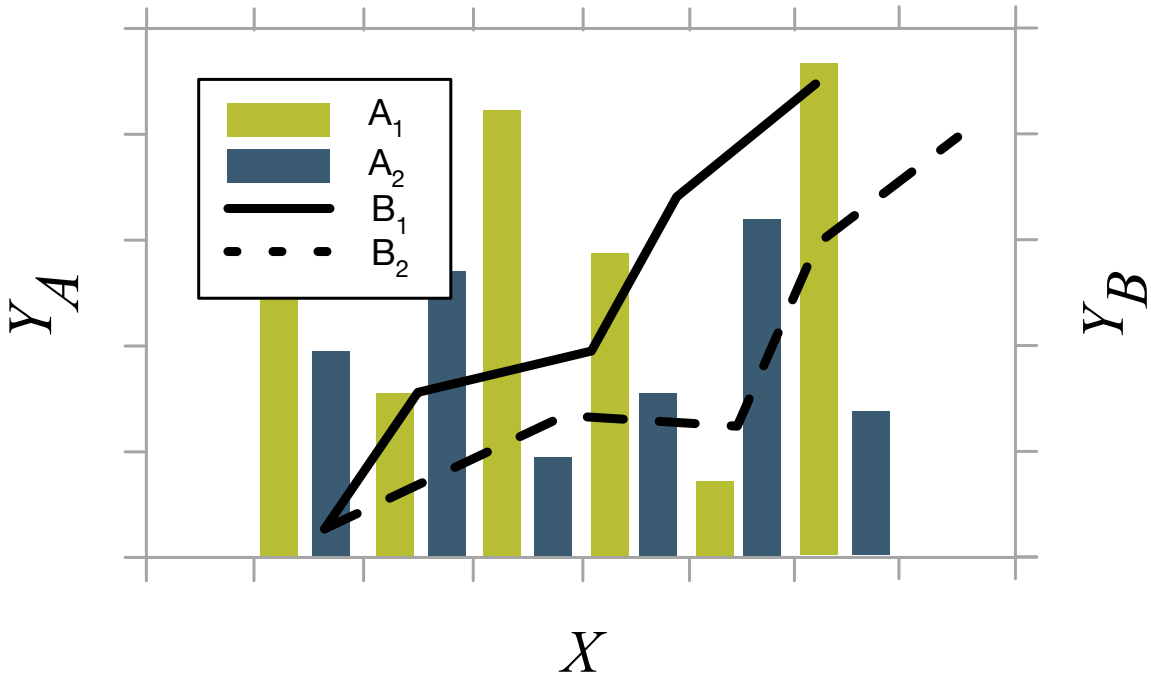
Learn to control basic plotting parameters



1. Axes labels
2. Value ranges and tick marks
3. Boxes — width, stroke, fill
4. Plot two data sets
5. Lines — thickness, line types
6. Lines points — size, type, fill
7. Key or legend position
8. Grids; vertical & horizontal lines
9. Plot area, and aspect ratio
10. Subplots

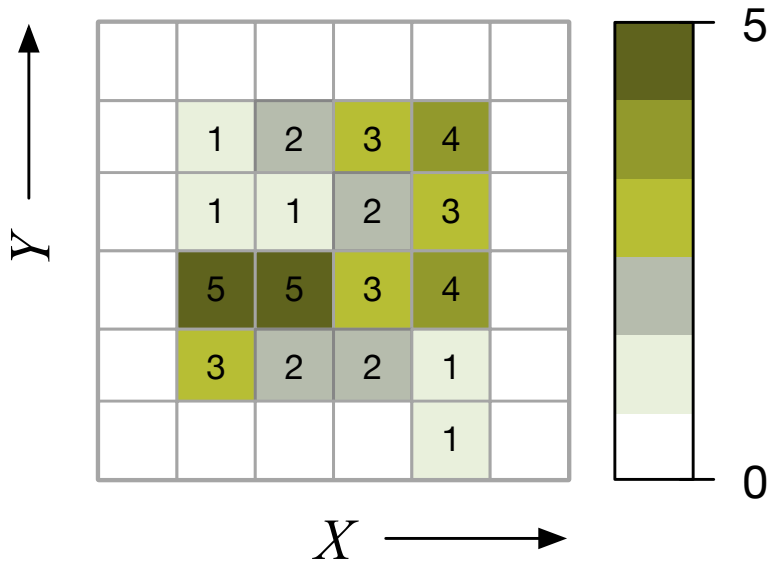
Data visualisation principles & tips

1. A plot is **not** a thousand words. It must convey a **single message** clearly.



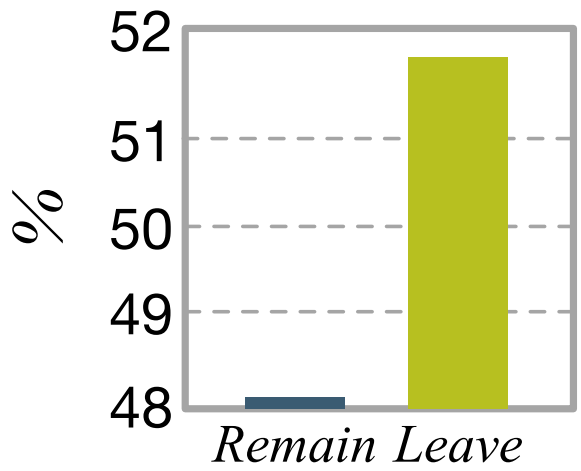
Data visualisation principles & tips

2. Bar charts are better than pie charts
3. Try encoding a 3rd variable in 2D before resorting to 3D
   - Try (x, y) plus size, or plus colour (contour plots)



Data visualisation principles & tips

4. Start from 0, or an appropriate minimum value
   - E.g., visualise 48.1 vs. 51.9%

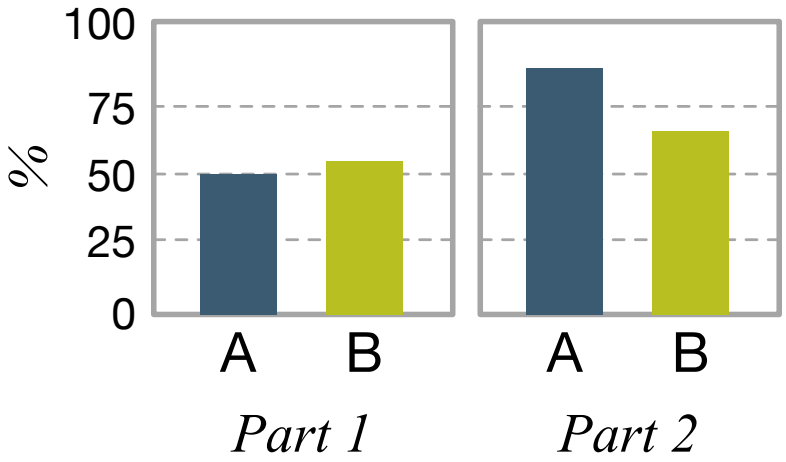


Just a 4% difference made look many-fold!

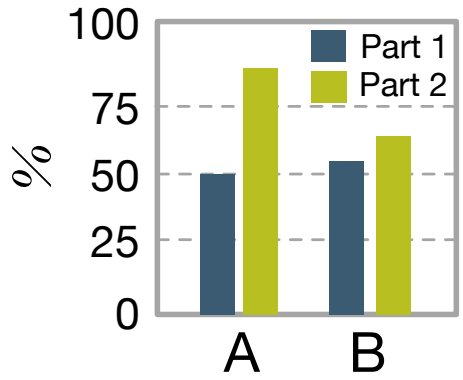
Data visualisation principles & tips

5. Make comparisons easy by using common axes

   - Just remember to use same range

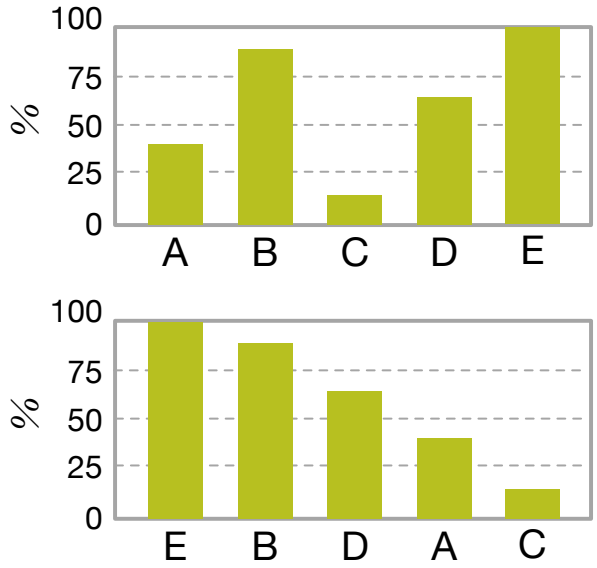


Alternatively,



Data visualisation principles & tips

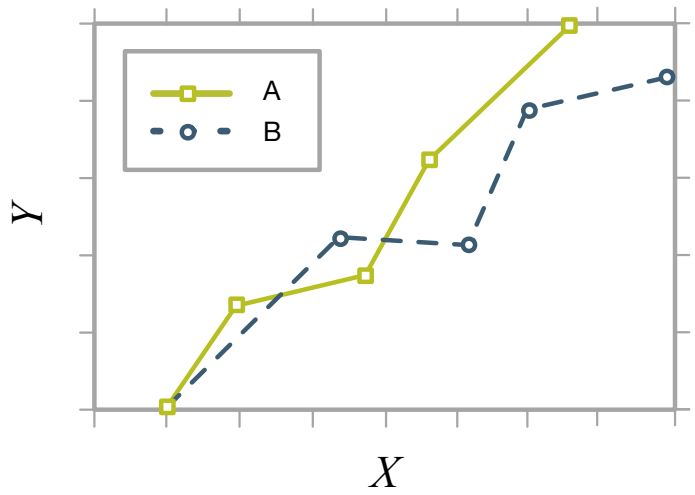
6. When ordering does not matter, order by value, not key
   - But consistency supersedes ordering



Data visualisation principles & tips

7. Use colour; and use it consistently

   - Don't differentiate just with color, try different line types, point markers, patterns etc.



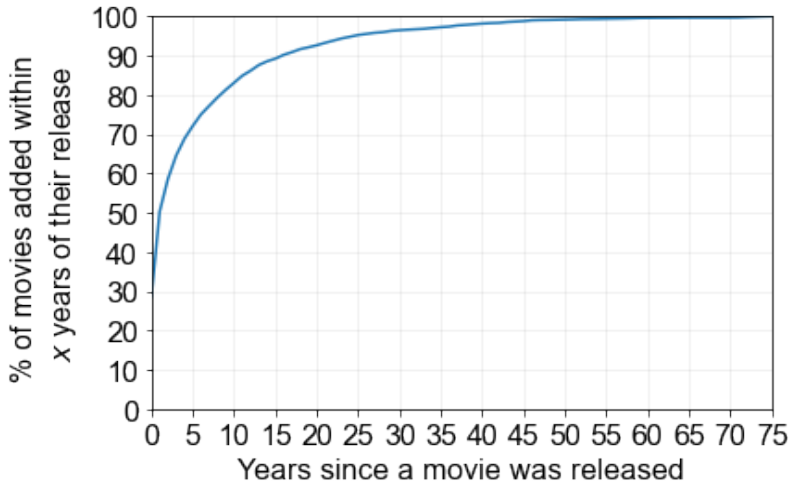
8. Finally, use the same theme throughout your report

How to present & write your results

1. Start with a 1-sentence summary of the key outcome of your result – what is the audience going to learn?
2. Explain briefly the methodology to arrive at this data view
3. Explain the x and y-axis of your plot
4. Explain the data shown on the plot
5. Conclude or summarise, if appropriate

An example

How recent are the recently added movies on Netflix?



(1) Most movies on Netfix are recent, but there are a few old gems. (2) The figure above shows the cummulative distribution of all movies in the dataset with respect to number of years passed from the time of their release until eventually added to the Netflix database. (3) The y-axis is cummulative – it shows the percentage of movies added within 0 years (i.e., immediately), within 1 year or less, 2 years or less, and so on, since released. (4) Almost half the movies are added within a year of their release. The oldest movie is 75 years old.

Data Visualisation Tools in Python

- Overview of tools like Matplotlib, and Seaborn.
- Demonstrations of basic to complex visualization techniques, including line charts, bar charts, histograms, and scatter plots.
- Discussion on the integration of these tools with Python programming to enhance data analysis and visual representation capabilities.

Matplotlib

Matplotlib is the foundational library for creating static, animated, and interactive visualizations in Python:

- Basic yet powerful, serving as the base for many other visualization libraries.
- Charts in Matplotlib consist of two main elements:
  - **Axes:** Lines that border the chart area, define where data is plotted on the X-axis and Y-axis.
  - **Elements:** Graphical representations of data, like lines, bars, and markers.

Seaborn

Seaborn enhances Matplotlib's capabilities, focusing on the aesthetics and usability for statistical graphics:

- Built on top of Matplotlib and closely integrated with pandas data structures.
- Facilitates the creation of complex visualizations such as heat maps, time series, and violin plots with minimal code.
- Known for its ability to produce aesthetically pleasing designs and complex visualizations effortlessly.

Generating Data for Various Plot Types

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")
import numpy as np
import pandas as pd

# For line plots
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.sin(x) * np.cos(x)

# For bar chart
categories = ['Category A', 'Category B', 'Category C']
values = np.random.randint(10, 50, size=3)

# For histogram
data_hist = np.random.randn(1000)

# For box and whisker plot
data_box = [np.random.normal(0, std, 100) for std in range(1, 4)]

# For scatter plot
x_scatter = np.random.rand(50)
y_scatter = 1 + 5 * x_scatter + 0.5 * np.random.rand(50)
```

Line Plot Example

```
plt.figure(figsize=(10, 5))
plt.plot(x, y1, label='sin(x)', color='blue', linestyle='-', linewidth=2)
plt.plot(x, y2, label='cos(x)', color='red', linestyle='--', linewidth=2)
plt.plot(x, y3, label='sin(x)*cos(x)', color='green', linestyle=':',
         linewidth=2)
plt.title('Multiple Line Plot')
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.legend()
plt.grid(True)
plt.show()
```

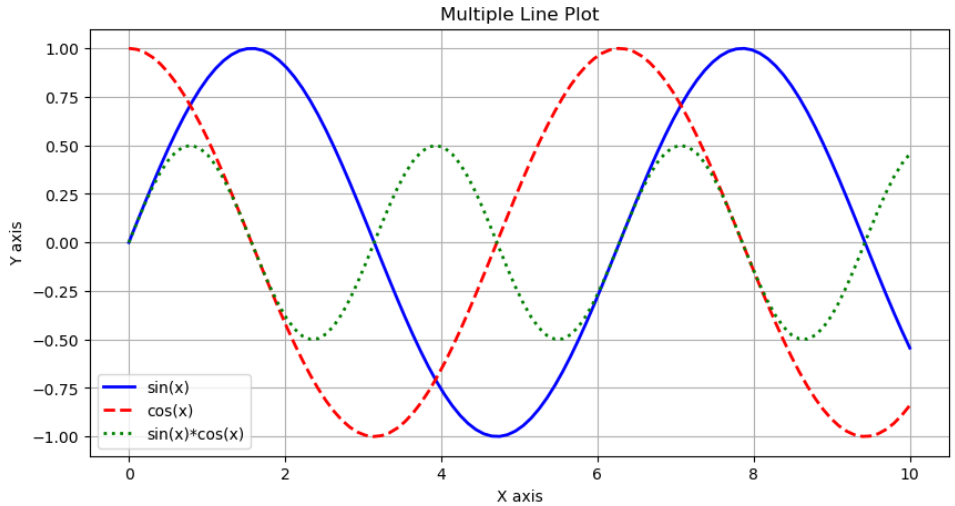


Figure: Line Plot of Sine Wave

Bar Chart Example

```
plt.figure(figsize=(7, 4))
plt.bar(categories, values, color=['blue', 'green', 'red'])
plt.title('Bar Chart Example')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.show()
```

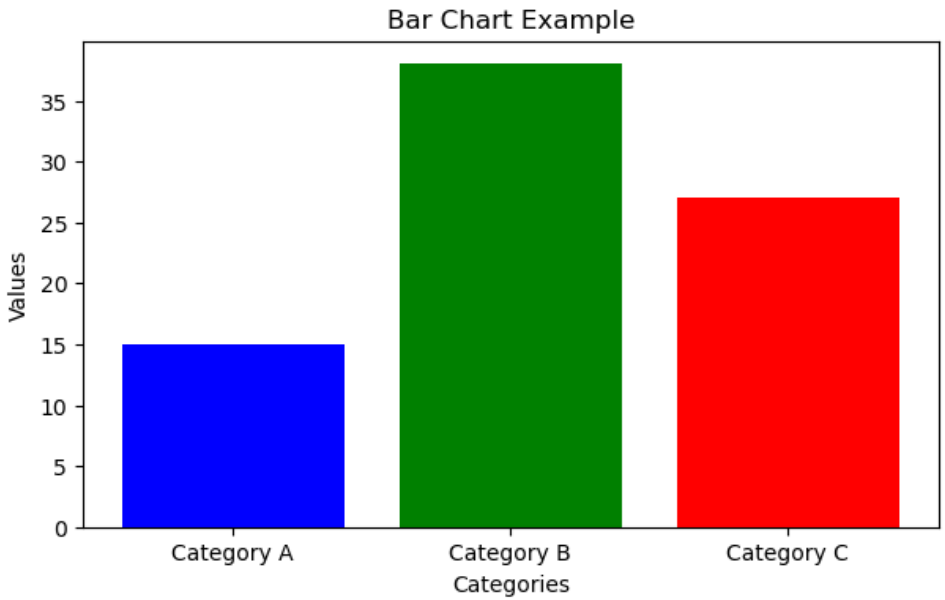


Figure: Bar Chart of Random Values

Histogram Example

```
plt.figure(figsize=(7, 4))
plt.hist(data_hist, bins=30, color='gray')
plt.title('Histogram Example')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.show()
```

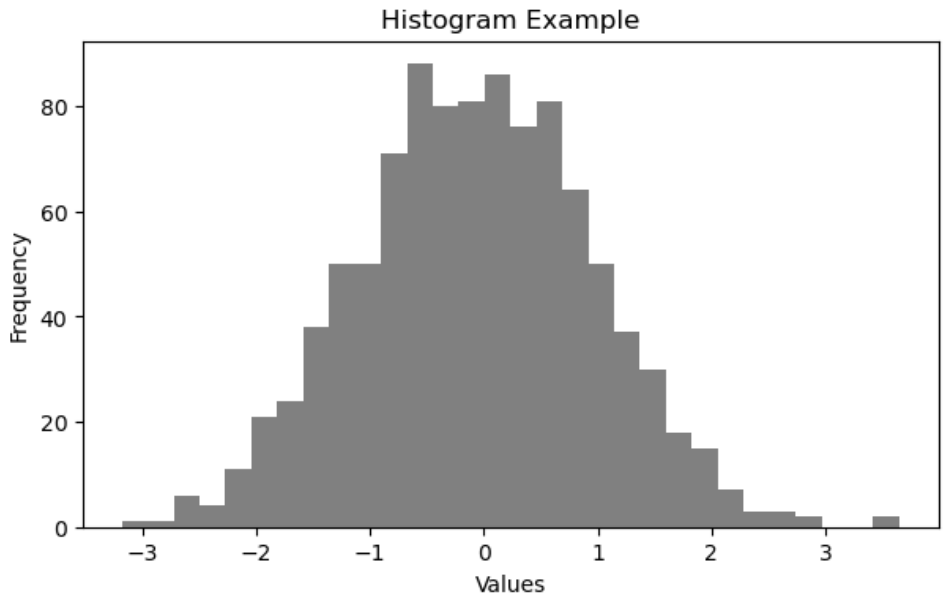


Figure: Histogram of Normally Distributed Data

Box and Whisker Plot Example

```
plt.figure(figsize=(7, 4))
plt.boxplot(data_box)
plt.title('Box and Whisker Plot Example')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.show()
```

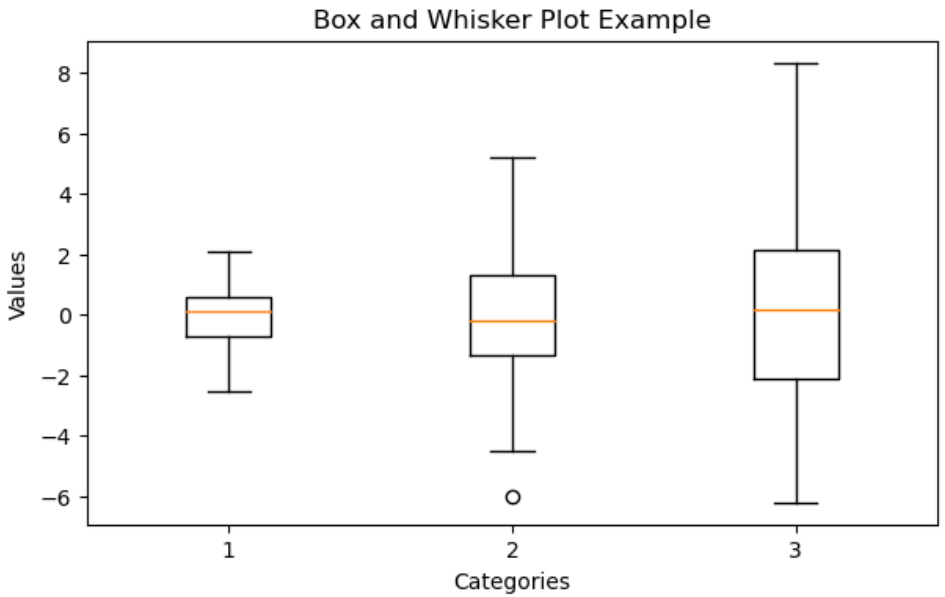


Figure: Box and Whisker Plot of Normally Distributed Data

Scatter Plot Example

```
plt.figure(figsize=(7, 4))
plt.scatter(x_scatter, y_scatter, color='purple')
plt.title('Scatter Plot Example')
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.show()
```

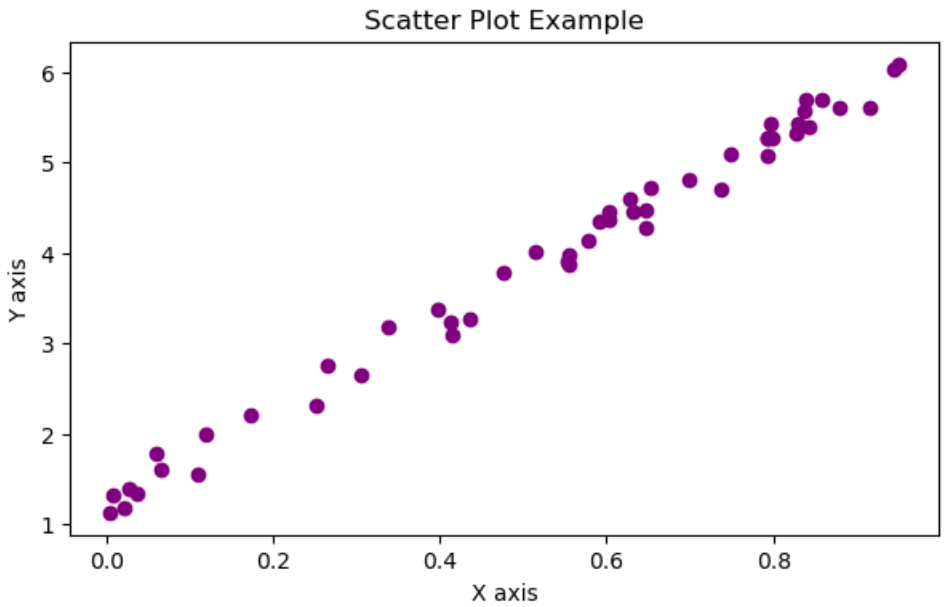


Figure: Scatter Plot of Random Points

Advanced Line Plot with Seaborn

```
plt.figure(figsize=(10, 6))
sns.lineplot(x=x, y=y1, label='sin(x)', color='blue', linestyle='-',
             linewidth=2)
sns.lineplot(x=x, y=y2, label='cos(x)', color='red', linestyle='--',
             linewidth=2)
sns.lineplot(x=x, y=y3, label='sin(x)*cos(x)', color='green', linestyle=':',
             linewidth=2)
plt.title('Multiple Line Plot')
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.legend()
plt.grid(True)
plt.show()
```

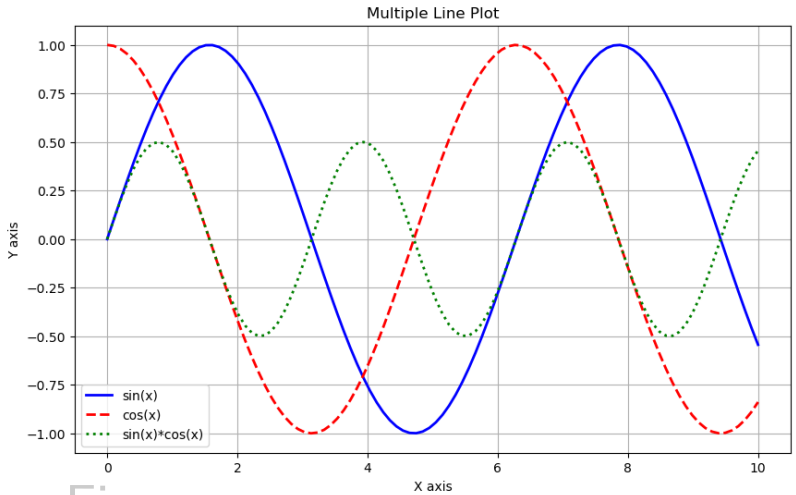


Figure: Advanced Line Plot with Seaborn

Seaborn Bar Chart Example

```
plt.figure(figsize=(7, 4))
sns.barplot(x=categories, y=values, palette='bright')
plt.title('Bar Chart Example')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.show()
```

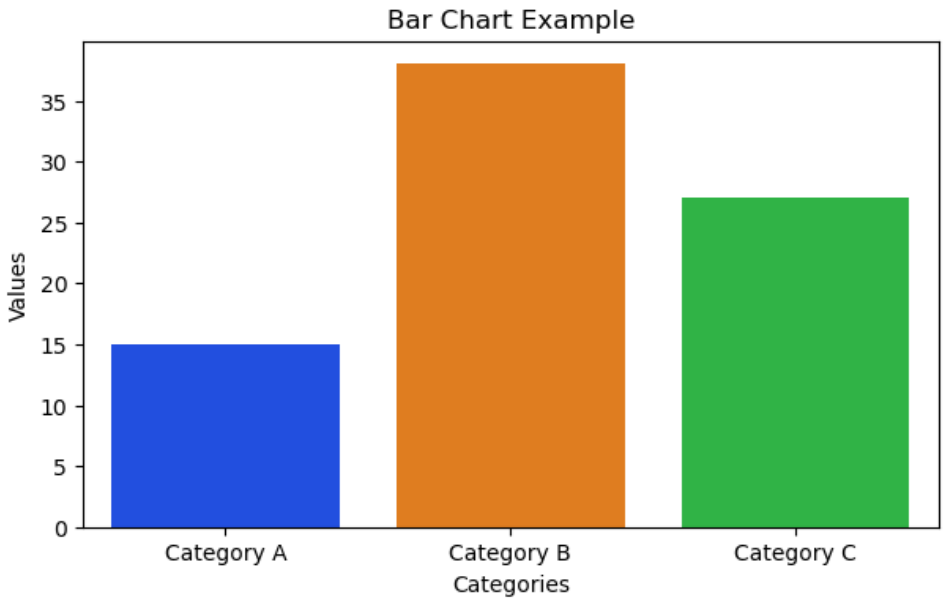


Figure: Seaborn Bar Chart Example

Seaborn Histogram Example

```
sns.set(style="whitegrid")
plt.figure(figsize=(7, 4))
sns.histplot(data_hist, kde=True, color='grey')
plt.title('Seaborn Histogram Example')
plt.xlabel('Values')
plt.ylabel('Density')
plt.show()
```

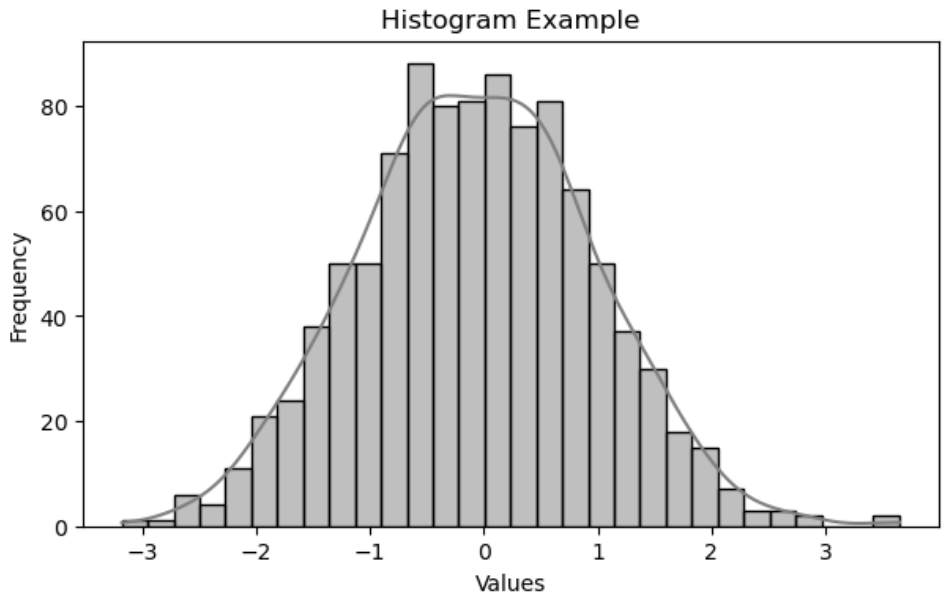


Figure: Seaborn Histogram Example with KDE

Seaborn Box and Whisker Plot Example

```
plt.figure(figsize=(7, 4))
sns.boxplot(data=np.array(data_box).T)
plt.title('Box and Whisker Plot Example')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.show()
```

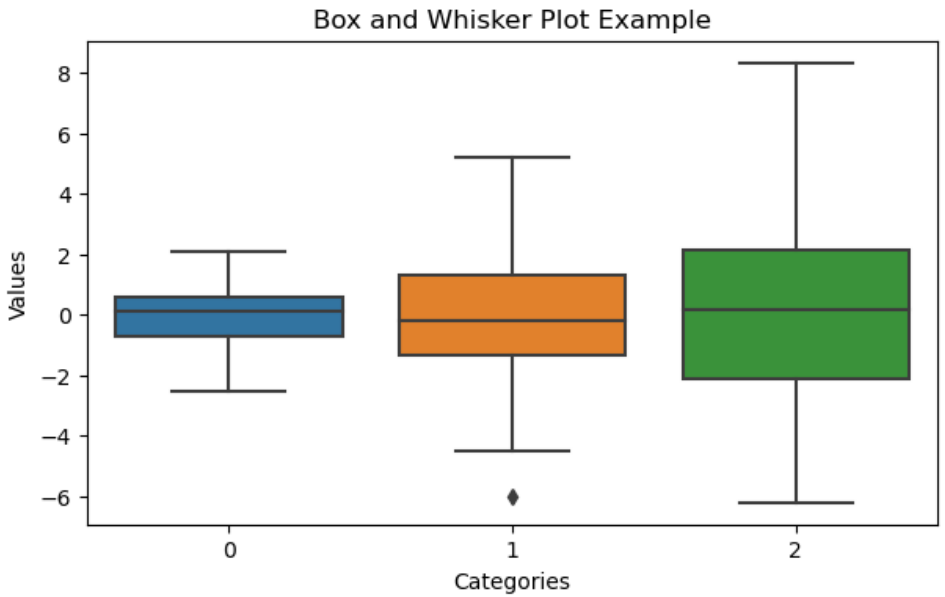


Figure: Seaborn Box and Whisker Plot Example

Seaborn Scatter Plot Example

```
plt.figure(figsize=(7, 4))
sns.scatterplot(x=x_scatter, y=y_scatter, color='purple')
plt.title('Scatter Plot Example')
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.show()
```

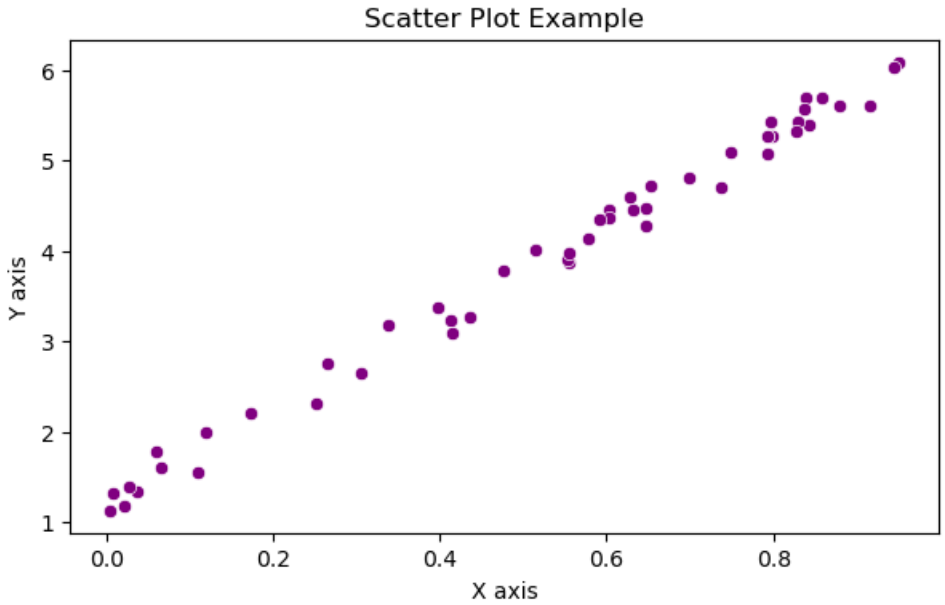


Figure: Seaborn Scatter Plot Example

Summary

- Think data applications ***end to end***
- A good data scientist understands the ***source*** and ***destination*** of data
- A good data scientist presents results clearly