Data Visualization and Plotting with Python

Jupyter notebook (Anaconda)

Biomedical Data Science Initiative class @ NIA/NIH

Dr. Showkat Dar

PostDoc

Computational Genomics Unit (LGG)

Class Times: 12:30-2:00 pm Tuesday, June 11th and Thursday, June 13th, 2024

Class link: https://nih.zoomgov.com/j/1613089725?pwd=dE8reWkzRnBSaTRWNXB4Sk5XWElDQT09

Our Schedule -> Hands-On :: 2-way Learning

Day-I:

1. Background - Rules for better figures
2. Hands-on – Introduction to Python's plotting libraries (Matplotlib and Seaborn)
3. Basic plotting techniques (line graphs, bar charts, and histograms) and saving plots in different formats.

Day-II:

4. Advanced visualizations (heatmaps, pair plots, and time series visualization)
5. Best practices in data visualization.

Day-I

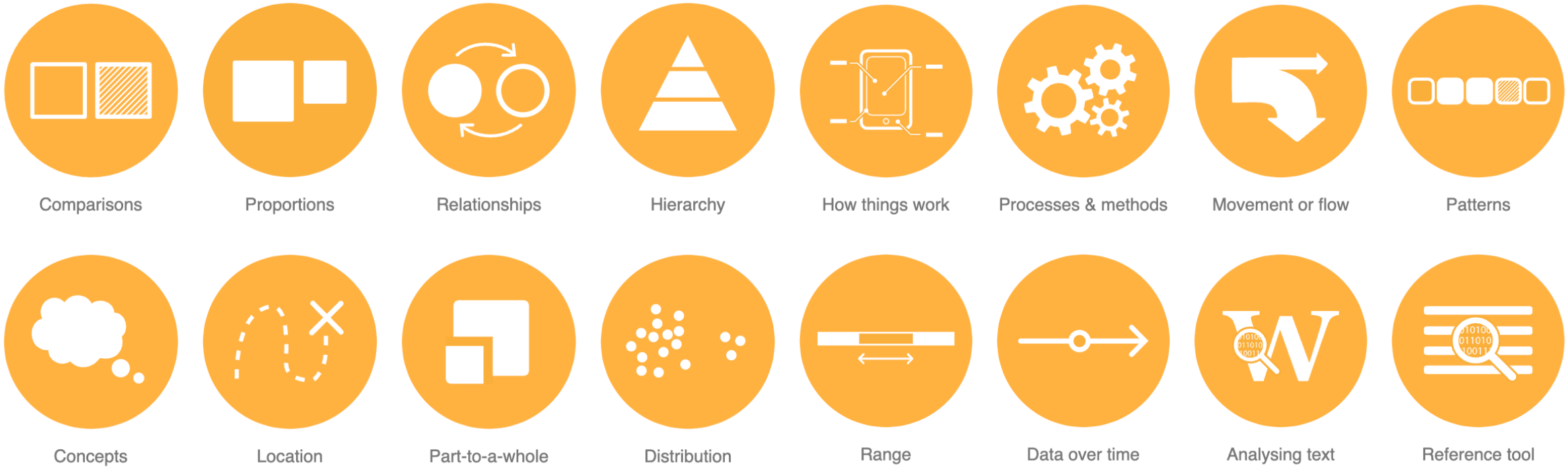
Background - Rules for better figures

Definition - data visualization?

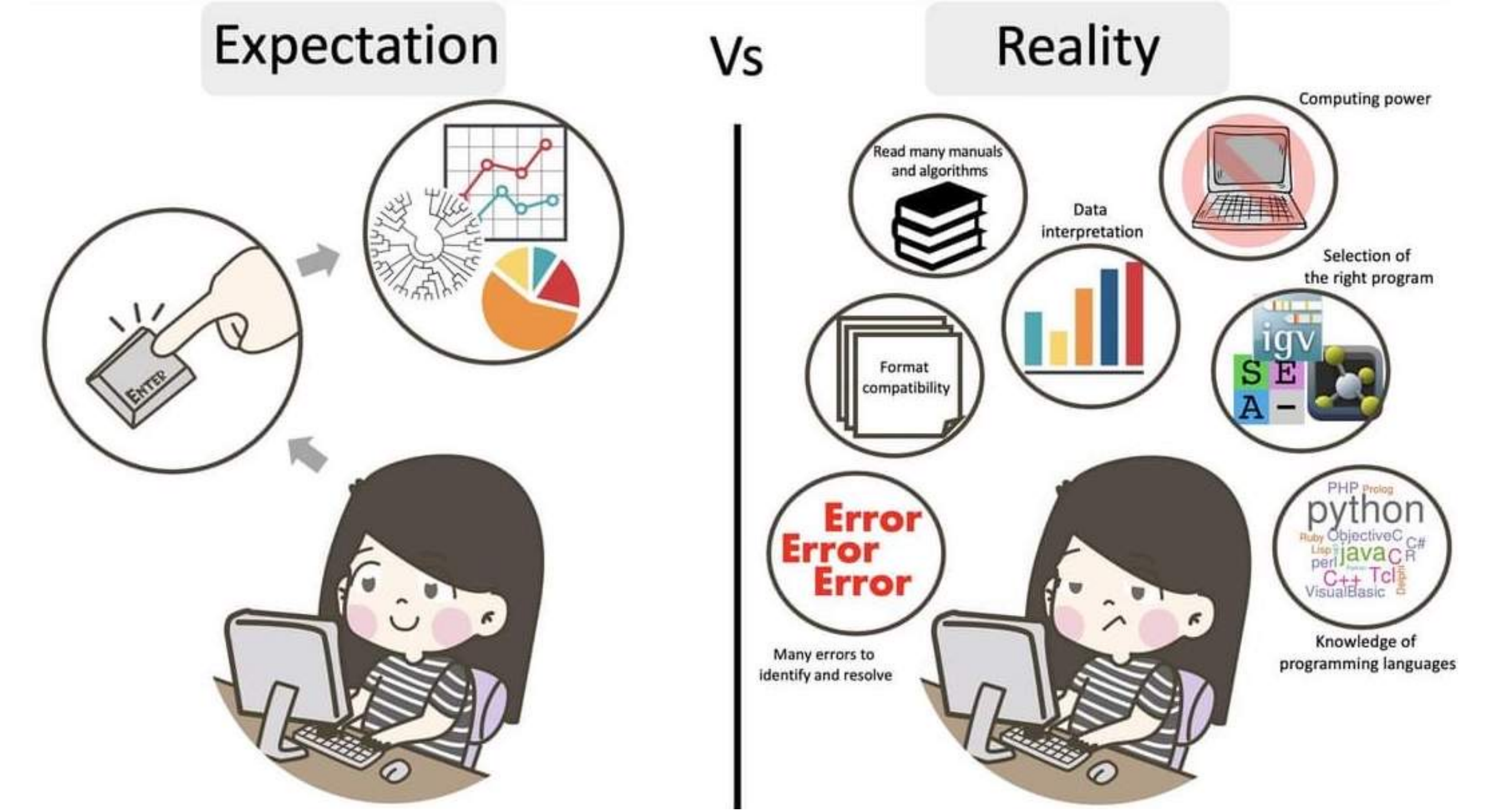
- Data visualization is the **graphical representation** of information and data.
- By using **visual elements like charts, graphs, and maps**, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.
- Excellent way to present data to non-technical audiences without confusion.
- Data visualization tools and technologies are essential to analyze massive amounts of information (**BIG-DATA**) and make data-driven decisions.

https://www.tableau.com/learn/articles/data-visualization#definition

What do you want to show?



https://datavizcatalogue.com/search.html



Adopted from https://training.galaxyproject.org/

Humans process visual data better.

60,000

The human brain processes images 60,000 times faster than text

50 milliseconds

How long it takes to form a 1st impression

90%

90 percent of information transmitted to the brain is visual

Examples...

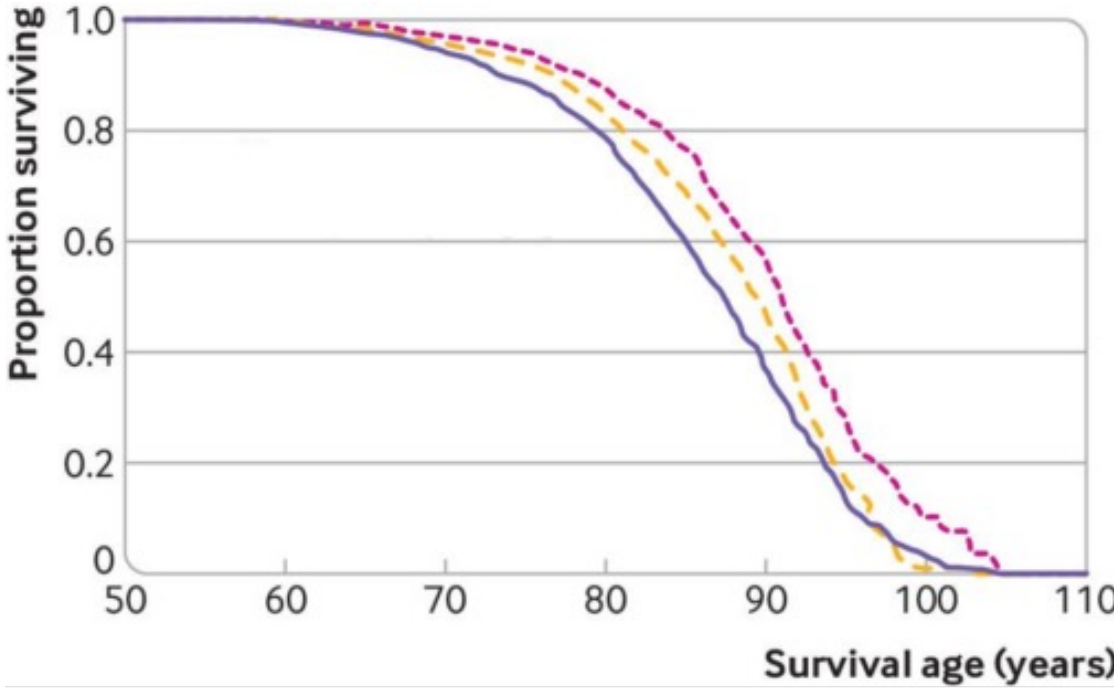
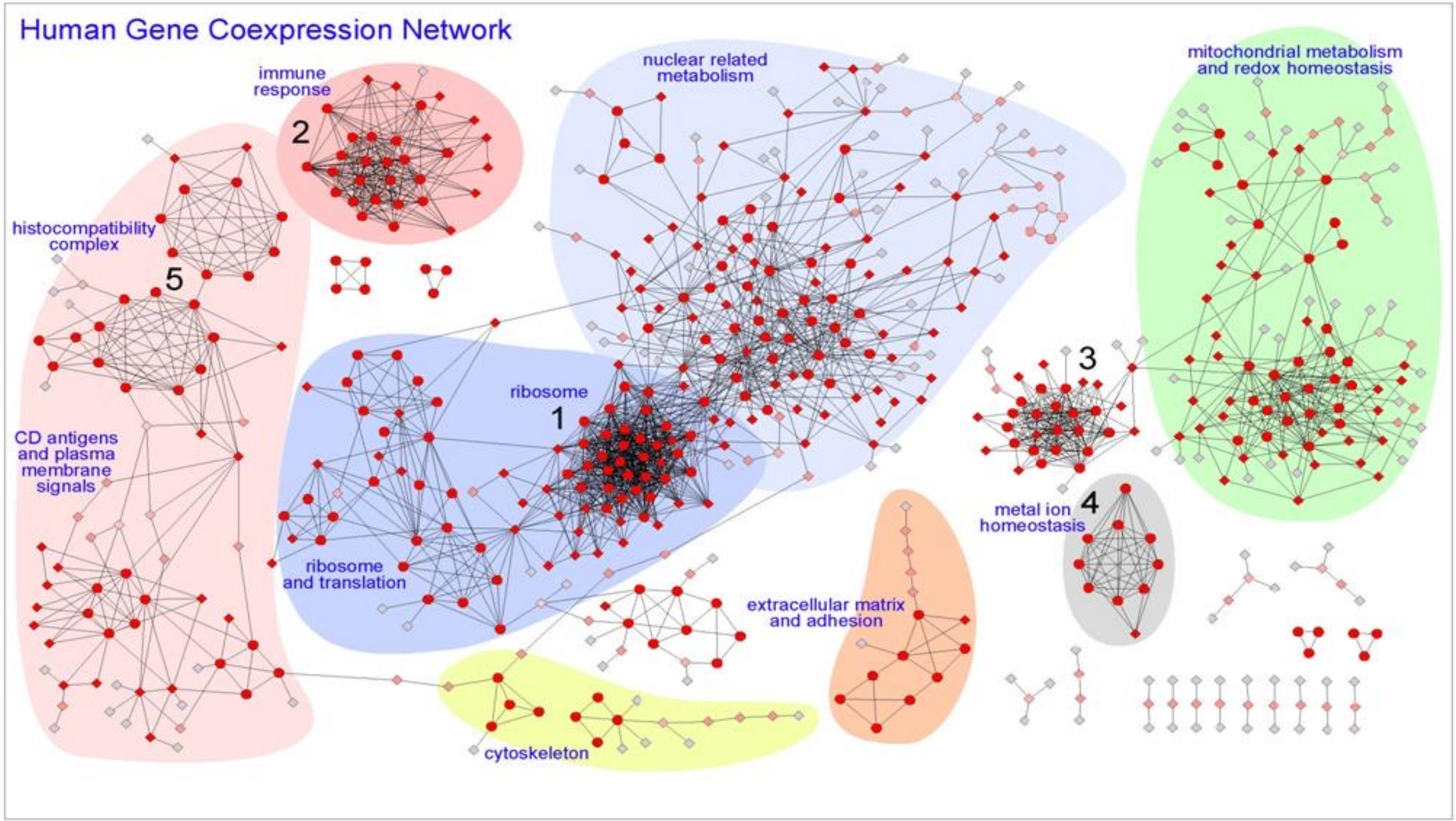


Fig. 1. Arts engagement had a protective association with longevity in older adults. Adjusted for demographic, socioeconomic, health related, behavioural, and social confounding factors. Solid blue line represents adults who never engaged with arts activities; yellow dashed line represents those who infrequently engaged with arts activities; pink dashed line represents those who frequently engaged with arts activities.



http://bioinfow.dep.usal.es/pages/coexpression/index.html

Ten Simple Rules for Better Figures

PLOS COMPUTATIONAL BIOLOGY

OPEN ACCESS

EDITORIAL

Ten Simple Rules for Better Figures

Nicolas P. Rougier, Michael Droettboom, Philip E. Bourne

Published: September 11, 2014 • https://doi.org/10.1371/journal.pcbi.1003833

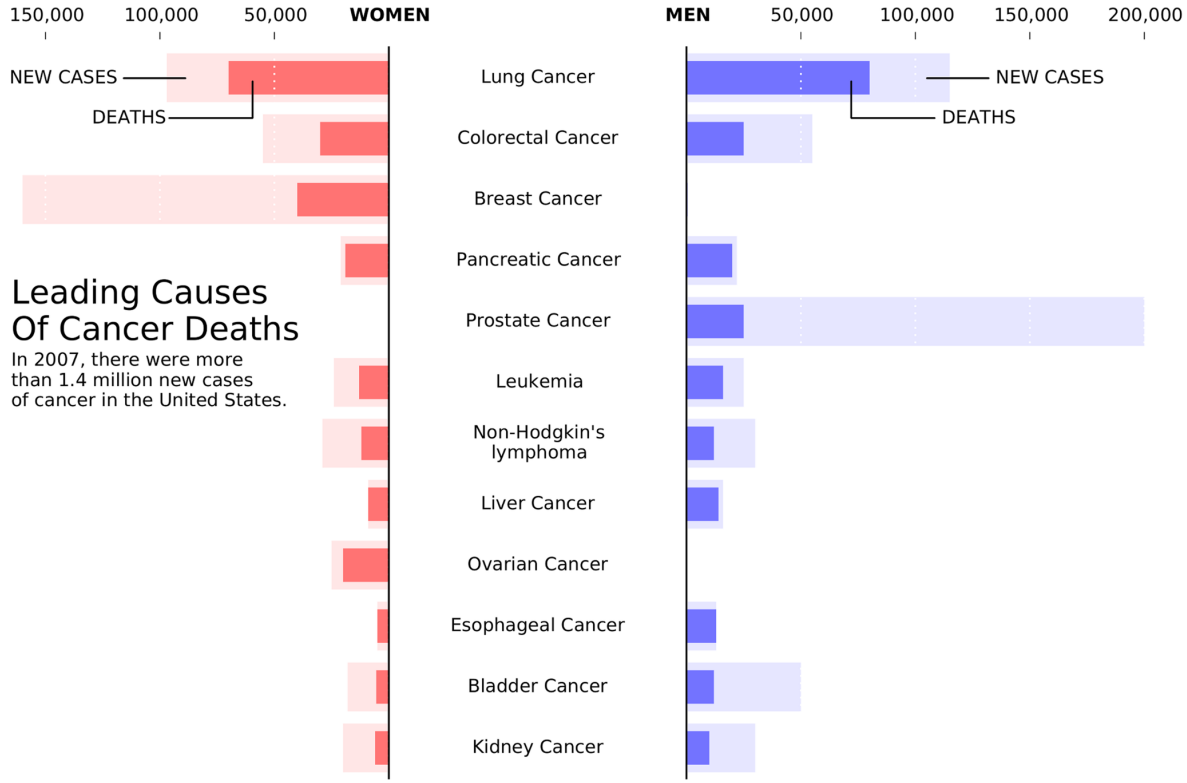
https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003833

Rule 1: Know Your Audience

- Understand the audience and the message.
- Design figures based on audience needs.
- Simple figures for the general public; detailed figures for scientific journals.
- Add extra information for students to ensure understanding.

Rule 2: Identify Your Message

- Clearly define the message of the figure.
- Design figures to express the message effectively.
- A clear message increases the impact of the figure.

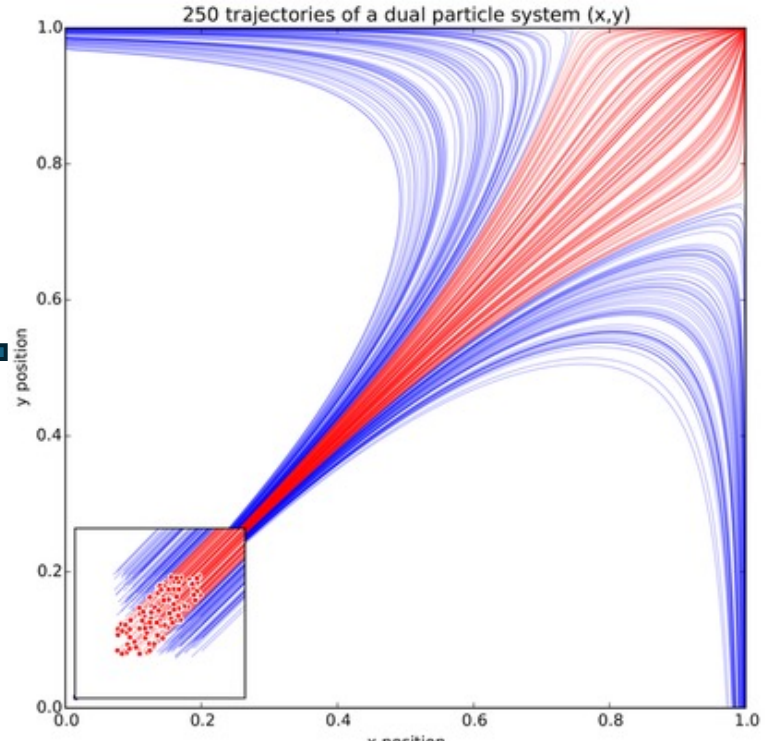


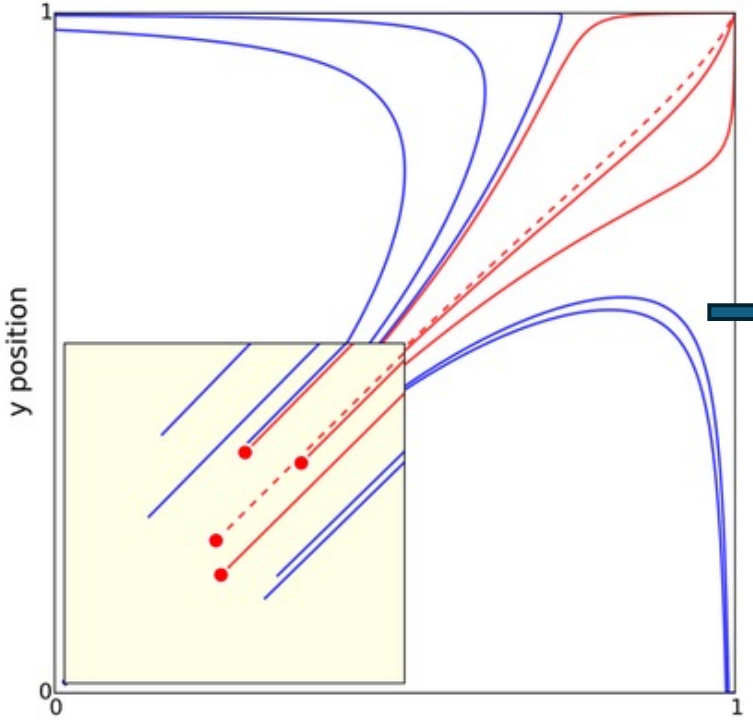
This is a remake of a figure that was originally published in the New York Times (NYT) in 2007. This new figure was made with matplotlib using approximated data. The data is made of four series (men deaths/cases, women deaths/cases) that could have been displayed using classical double column (deaths/cases) bar plots. However, the layout used here is better for the intended audience.

Rule 3: Adapt the Figure to the Support Medium

- Tailor figures to the display medium (poster, screen, paper).
- Keep figures simple and clear for oral presentations.
- Include more details in figures for journal articles.

Manuscript





Oral Presentation

Simulation of the trajectories of a dual-particle system, where each particle interacts with the other. Depending on the initial conditions, the system may end up in three different states

Rule 4: Captions Are Not Optional

- Use captions to explain the figure.
- Provide details that can't be visually represented.
- Anticipate questions and include necessary information.

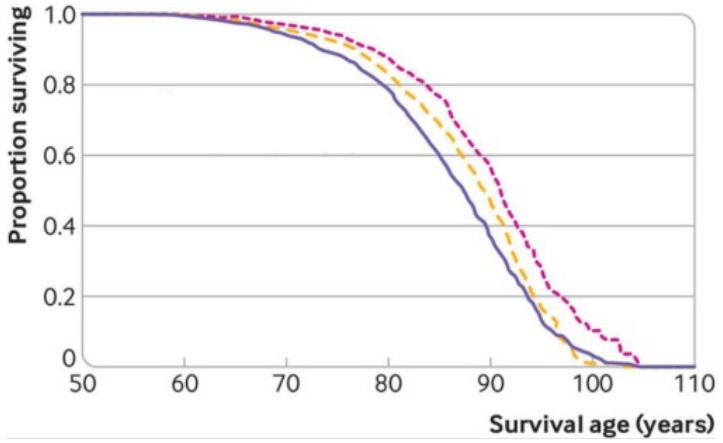


Fig. 1. Arts engagement had a protective association with longevity in older adults. Adjusted for demographic, socioeconomic, health related, behavioural, and social confounding factors. Solid blue line represents adults who never engaged with arts activities; yellow dashed line represents those who infrequently engaged with arts activities; pink dashed line represents those who frequently engaged with arts activities.

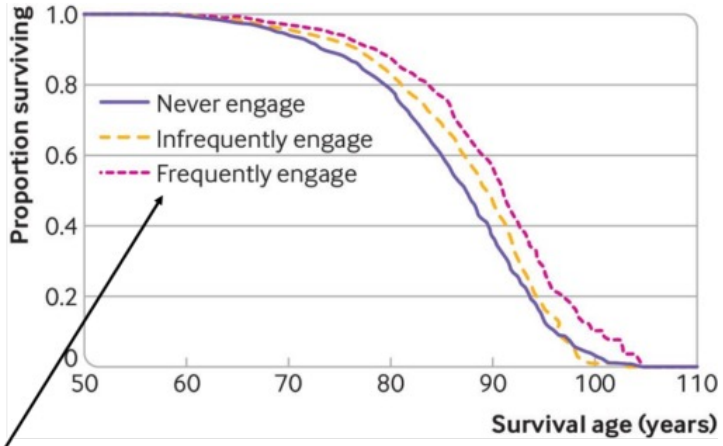
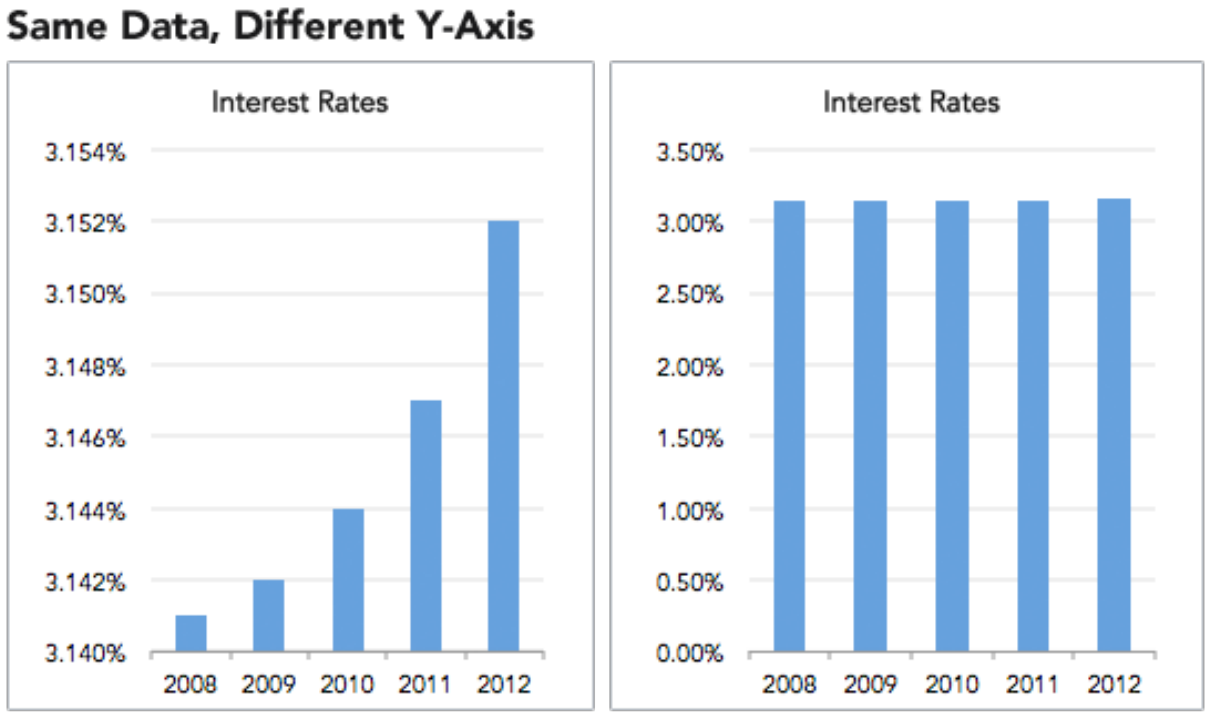


Fig. 1. Arts engagement had a protective association with longevity in older adults. Adjusted for demographic, socioeconomic, health related, behavioural, and social confounding factors.

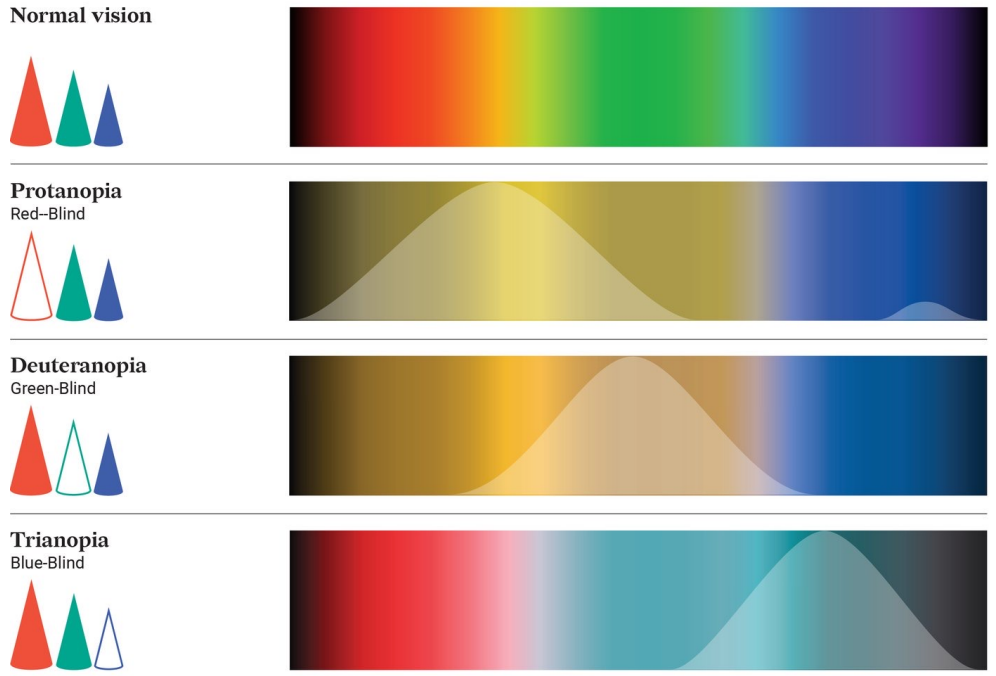
Rule 5: Do Not Trust the Defaults

- Default settings may not suit your figure's needs.
- Manually adjust settings for clarity and precision.
- Customize colors, fonts, and styles to enhance the message.



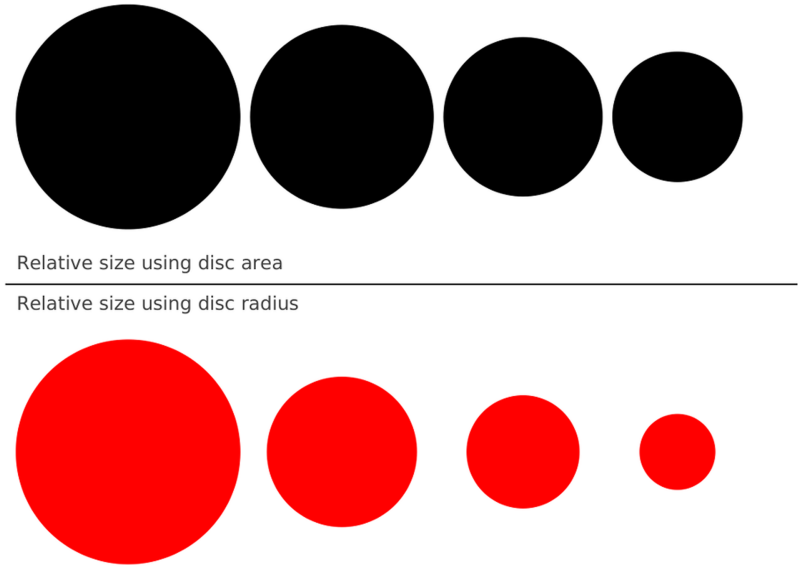
Rule 6: Use Color Effectively

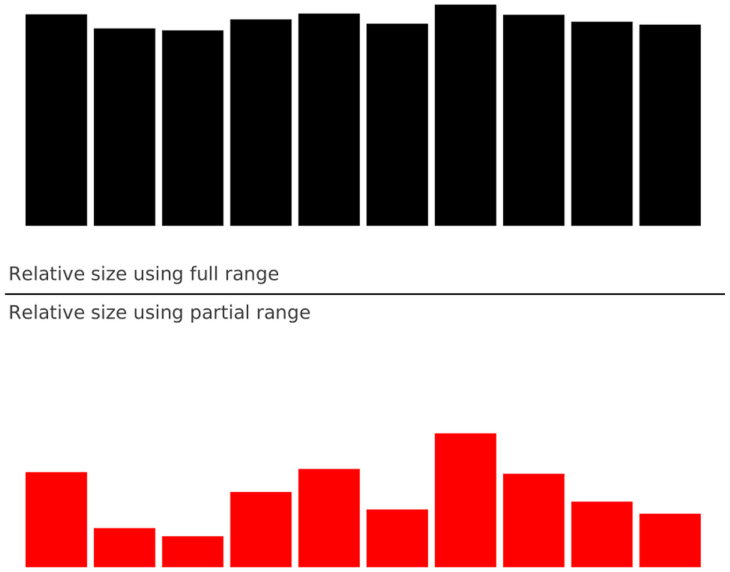
- Use color to highlight important elements.
- Avoid unnecessary colors.
- Choose the appropriate colormap for your data type.
- Consider color blindness in your color choices.



Rule 7: Do Not Mislead the Reader

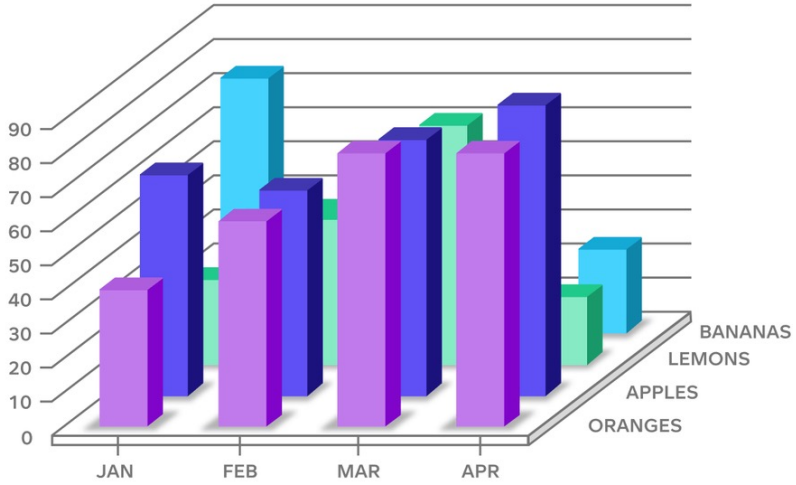
- Represent data accurately and objectively.
- Avoid visual distortions and misleading scales.
- Use simple plots and appropriate labels.

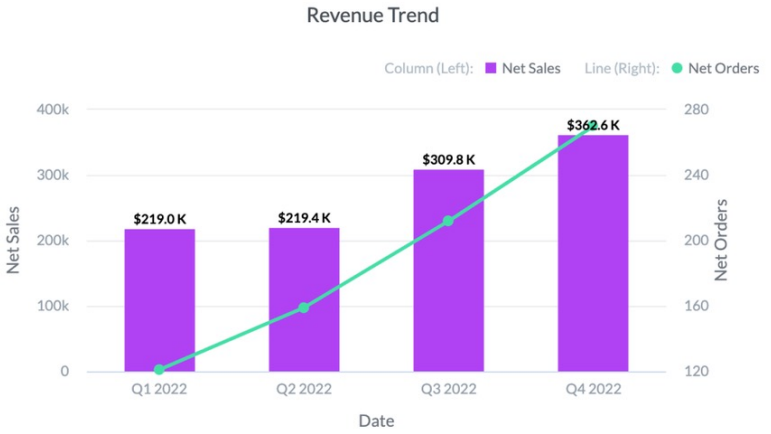




Rule 8: Avoid “Chartjunk”

- Eliminate unnecessary visual elements.
- Keep the design clean and focused on the data.
- Justify any additional elements to avoid clutter.

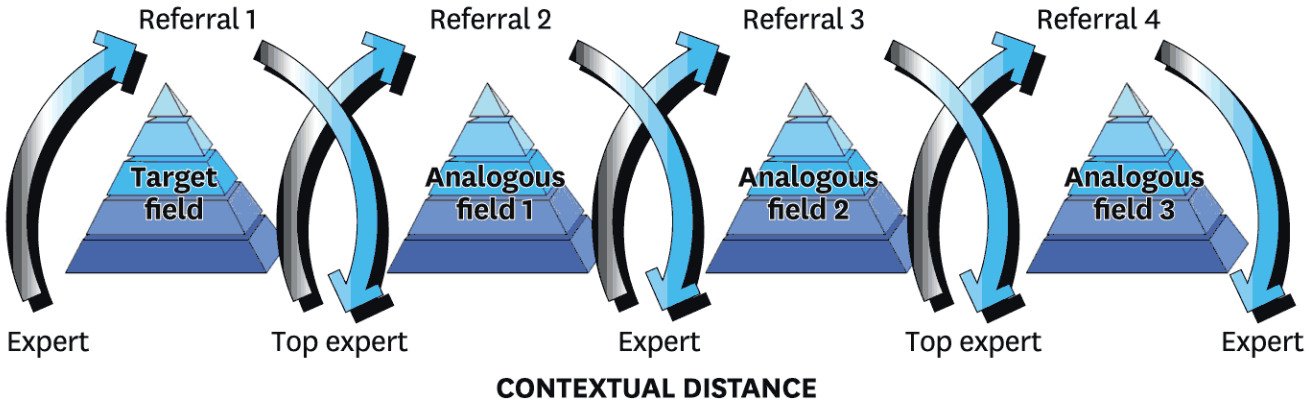


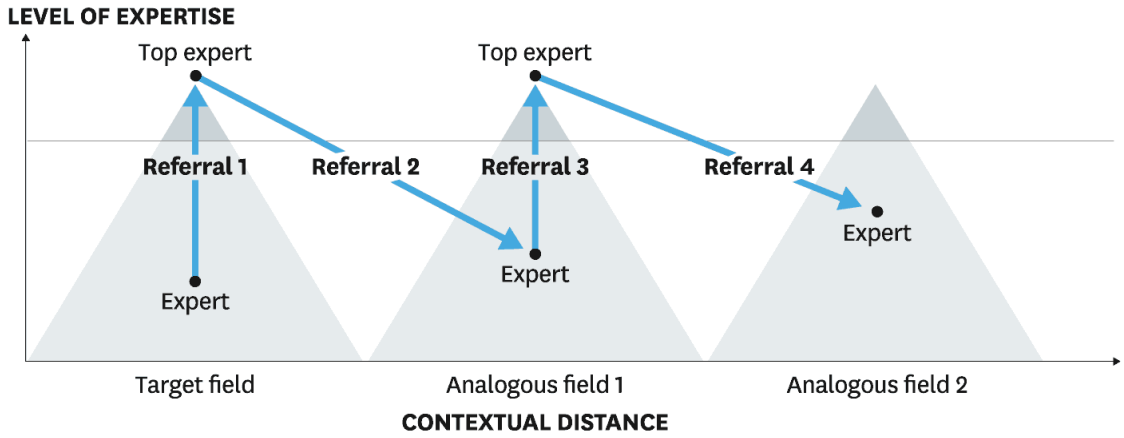


Rule 9: Message Trumps Beauty

- Prioritize clarity and readability over aesthetics.
- Follow scientific best practices for figure design.
- Avoid purely aesthetic designs that compromise data integrity.

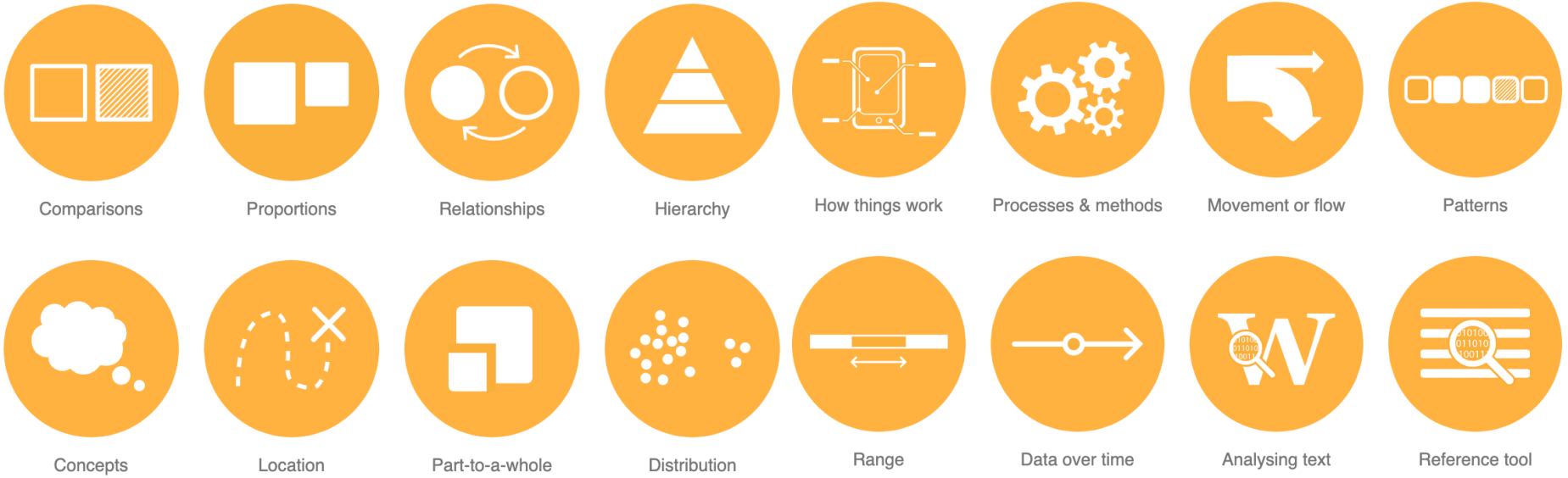
How does a pyramid search work?





Rule 10: Get the Right Tool

- Use appropriate tools for different types of figures.
- Explore open-source tools for flexibility and control.
- Familiarize yourself with specialized software for better results.



https://datavizcatalogue.com/search.html

Break !

2. Hands-on

Introduction to Python's plotting libraries : Matplotlib and Seaborn

Requirements:

- System installed Jupyter notebook (Anaconda) with packages python (), matplotlib and seaborn.
- Internet connection

Python Data Visualization Libraries

- **Matplotlib :**

A foundational library for creating static, animated, and interactive visualizations in Python. Matplotlib is versatile, supporting a wide range of plot types and customization options. It integrates well with Numpy for handling complex numerical data and allows exporting visualizations in various formats.
URL:: **https://matplotlib.org/**

- **Seaborn :**

Built on top of Matplotlib, Seaborn simplifies the creation of beautiful, informative statistical graphics. It offers enhanced support for themes and color palettes and integrates seamlessly with Pandas for structured data visualization. Seaborn is ideal for users seeking attractive default styles and color schemes.

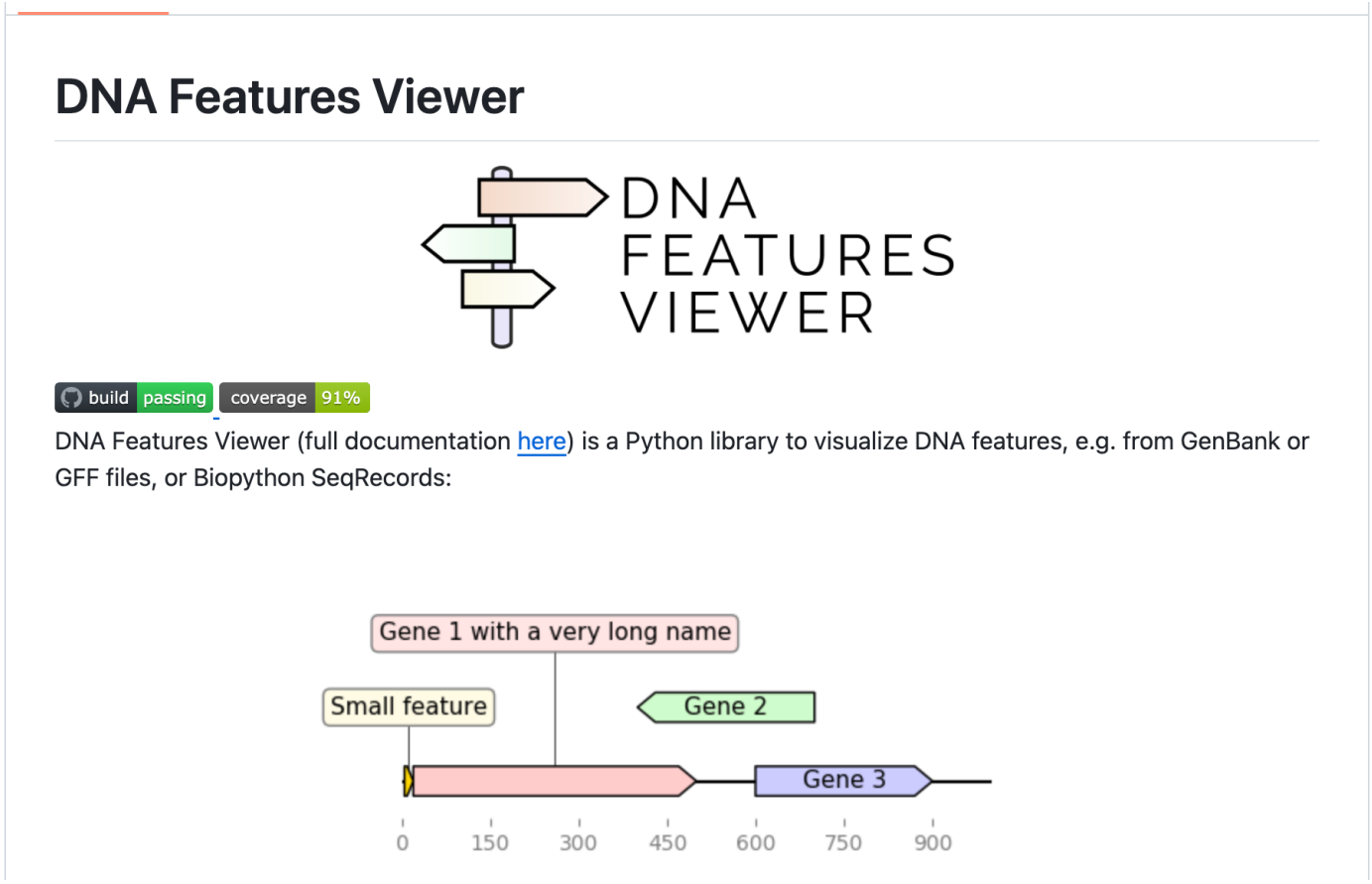
URL:: **https://seaborn.pydata.org/**

Python Data Visualization Libraries (others)

- **Plotly :** A library for making interactive, publication-quality graphs online. Plotly's strength lies in its ability to create complex, interactive plots that are web-friendly.
- **Altair :**Focused on declarative statistical visualization, Altair is built on Vega-Lite and allows for concise, intuitive plot creation.
- **Bokeh :**Targets web browsers for output, offering interactive, web-ready visualizations. Bokeh is suitable for creating interactive plots, dashboards, and data applications.
- **GeoPandas:** Extends Pandas for working with geospatial data, integrating with libraries like Shapely and Fiona for geometric operations and file access, respectively.
- **Python Data Visualization with Hex:** Hex is a popular polyglot development environment that allows you to write code in multiple languages in the same environment.

DNA Features Viewer

DNA Features Viewer is a Python library to visualize DNA features, e.g. from GenBank or Gff files, or Biopython SeqRecords.



https://github.com/Edinburgh-Genome-Foundry/DnaFeaturesViewer

Python 3rd-party and user-contributed packages

A list of packages that extend Matplotlib. These are maintained and distributed independently from Matplotlib so go to the website listed for instructions.

- https://matplotlib.org/mpl-third-party/

What we need for visualization?

Step 1. Data in proper format like table, no-zeros, NA's, missing values, empty values.

Step 2. Create the environment for the plotting like

python→ matplotlib, seaborn (in Jupyter notebook).

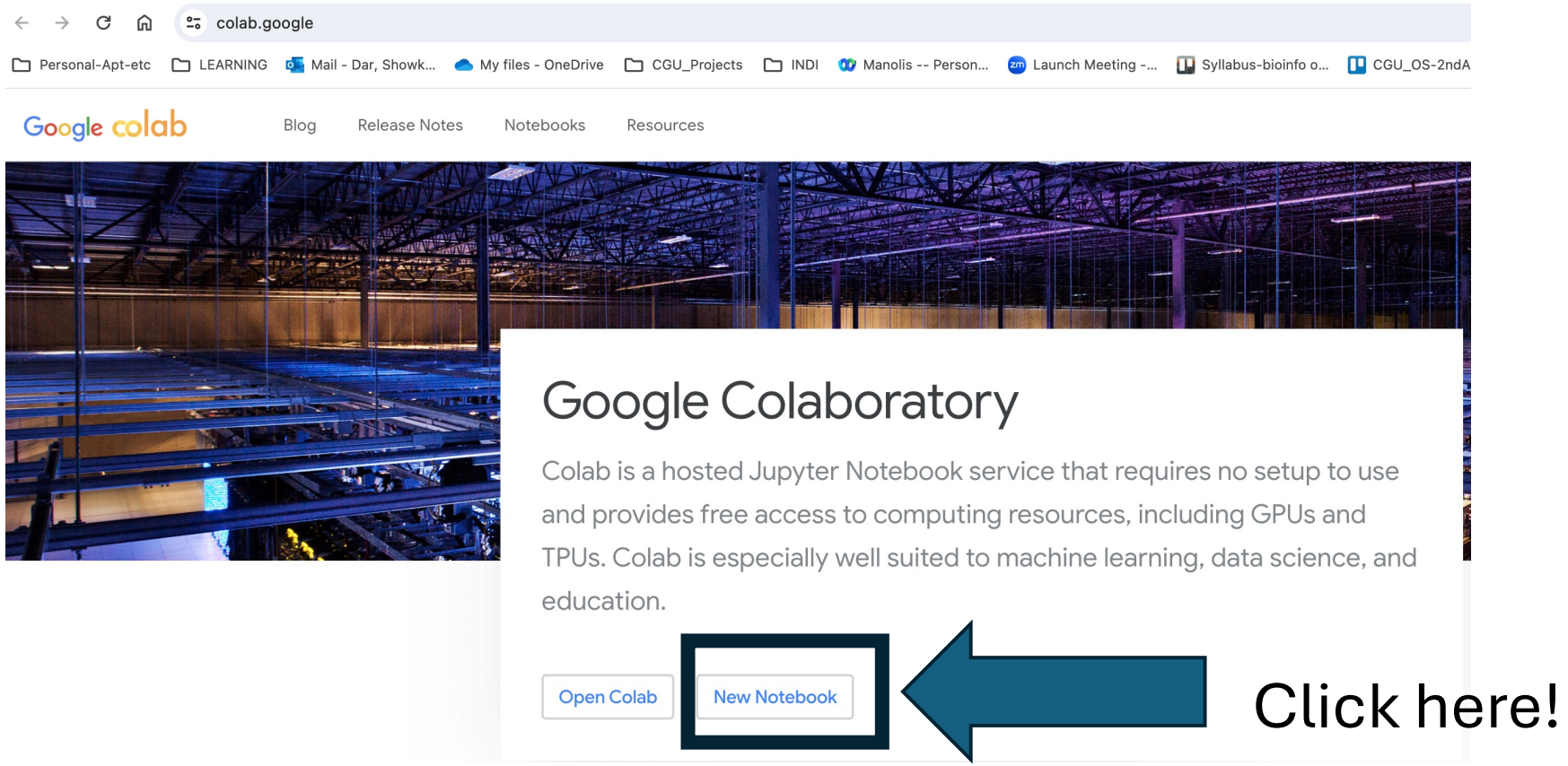
Step 3. Loading the libraries and data.

Step 4. Creating the required plots.

Step 5. Saving the plot (and further modification in Photoshop/Affinity etc).

Online jupyter notebook options.

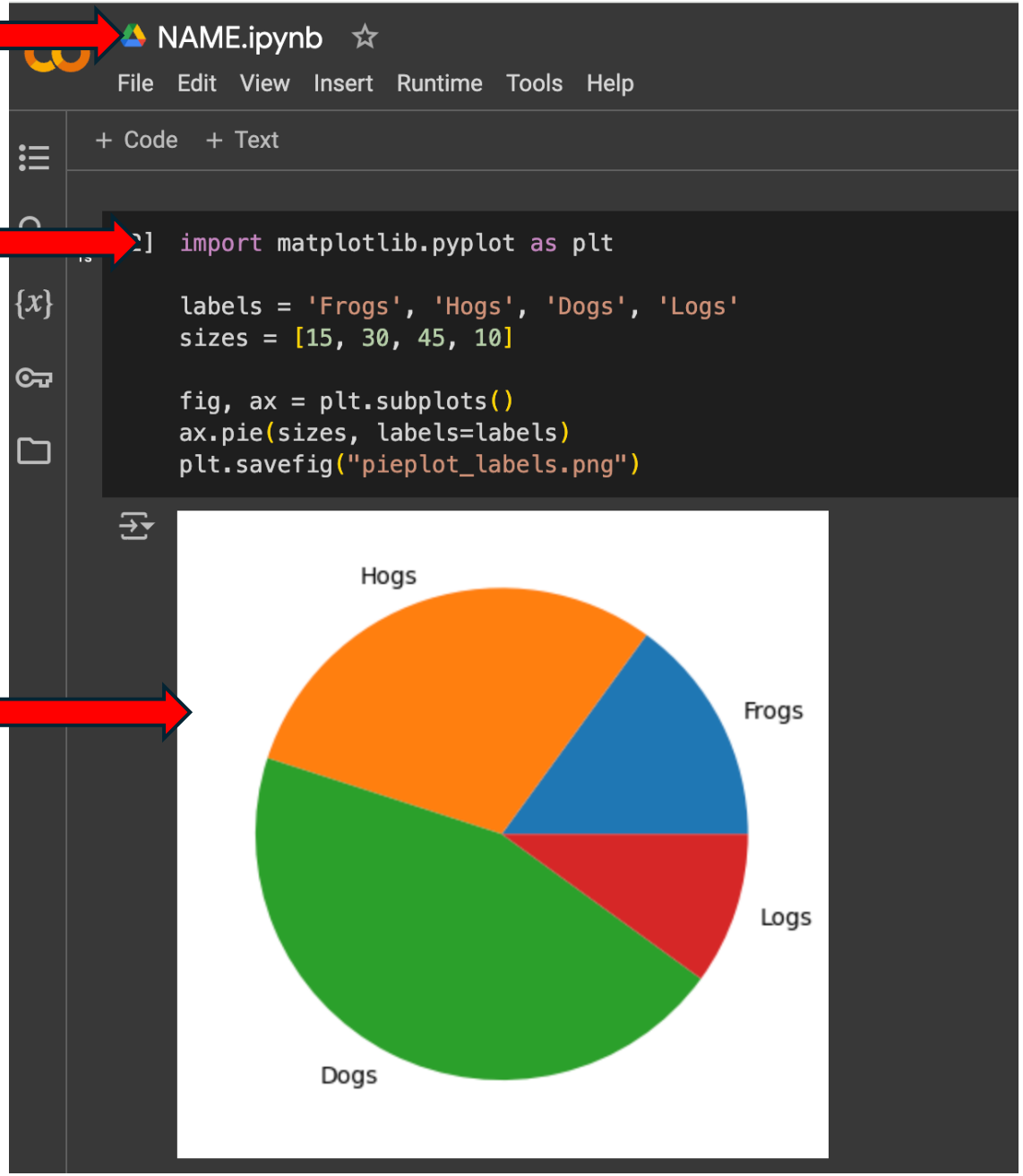
- Go to https://colab.google/



Rename the notebook

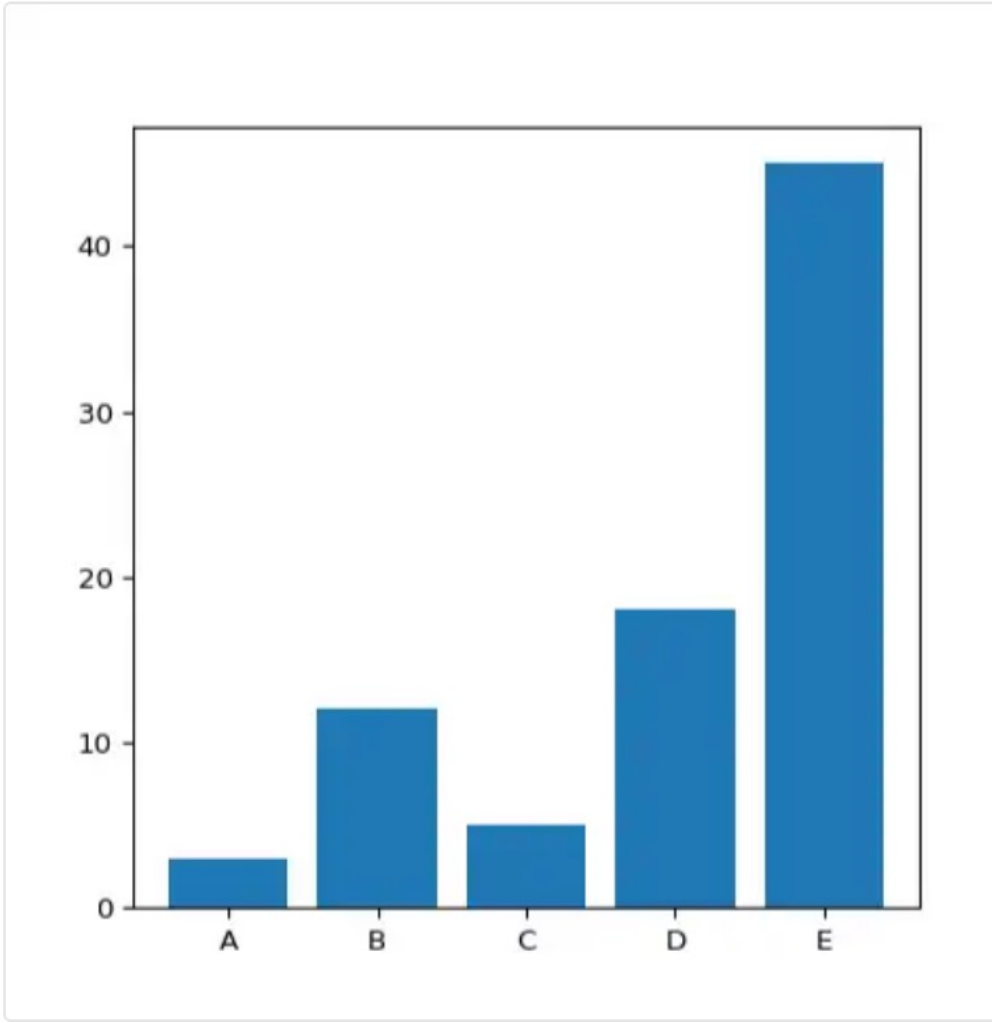
Write and Run the code

Generate plots and save them



Matplotlib : Barplot

```
import matplotlib.pyplot as plt

# Make a random dataset:
height = [3, 12, 5, 18, 45]
bars = ['A', 'B', 'C', 'D', 'E']

# Create bars
plt.bar(bars, height)

# Show graphic
plt.show()

# Save figure in different formats
plt.savefig('bar_plot.png')  # Save as PNG
plt.savefig('bar_plot.pdf')  # Save as PDF
plt.savefig('bar_plot.svg')  # Save as SVG
```



https://python-graph-gallery.com/barplot/

3. Basic plotting techniques

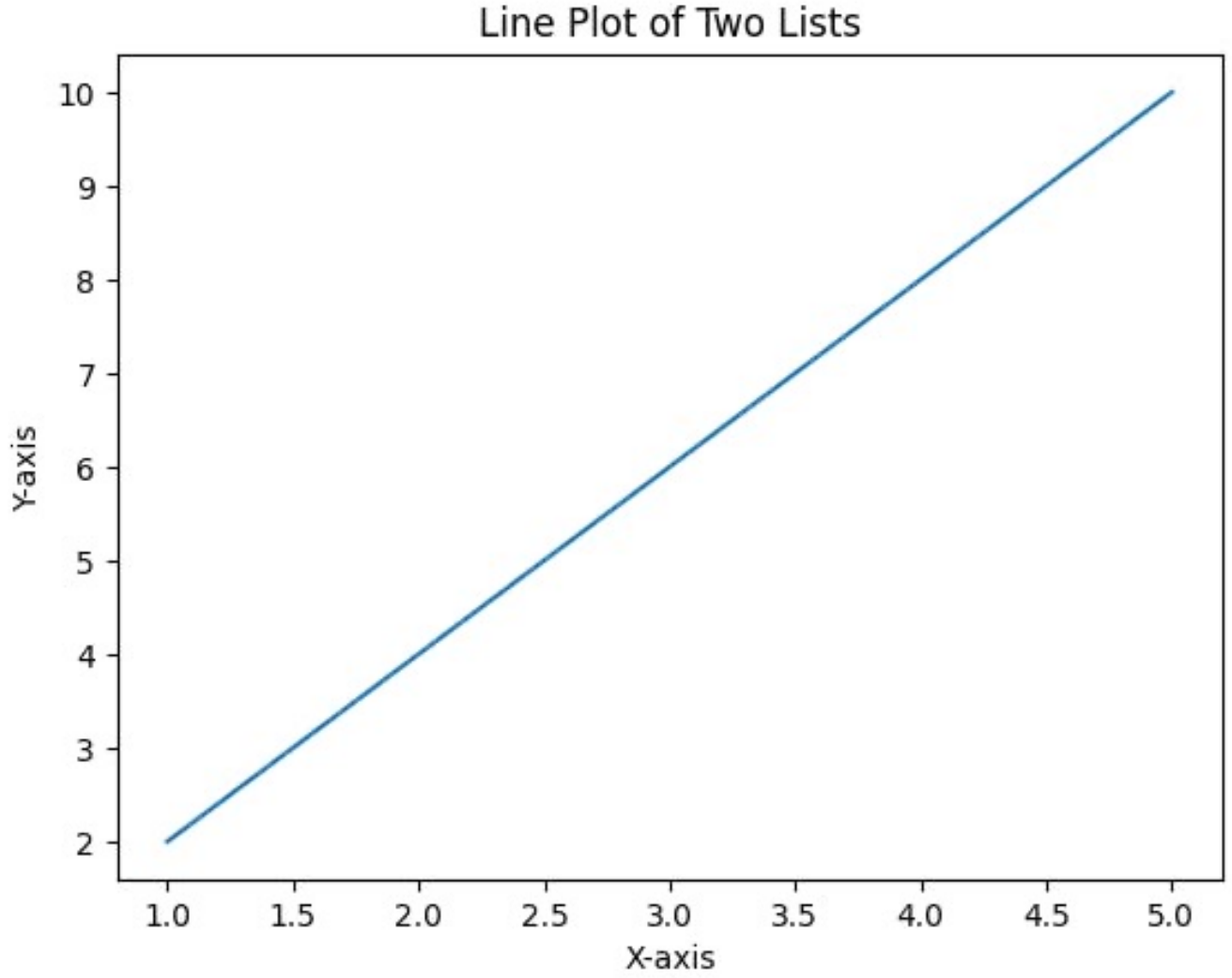
- Line graphs, bar charts, and histograms) and more …
- Saving plots in different formats

Matplotlib – Examples/Gallery

- https://matplotlib.org/stable/gallery/

Matplotlib : Plotting categorical variables

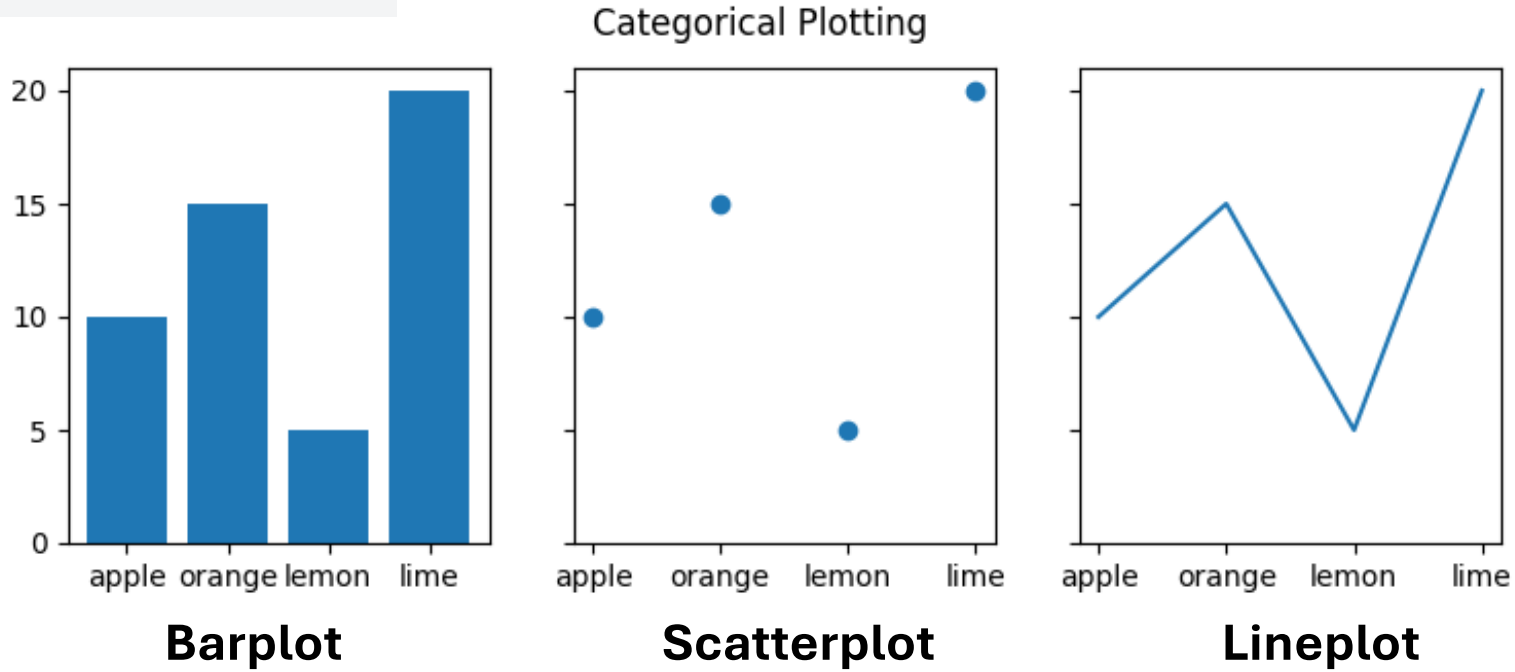
```
import matplotlib.pyplot as plt
# Sample data
listOne = [1, 2, 3, 4, 5]
listTwo = [2, 4, 6, 8, 10]

# Create a line plot
plt.plot(listOne, listTwo)

# Add labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Line Plot of Two Lists')

# Show the plot
plt.show()
plt.savefig('Lineplot.tiff')
```



Lineplot

Matplotlib : Plotting categorical variables

```
import matplotlib.pyplot as plt

data = {'apple': 10, 'orange': 15, 'lemon': 5, 'lime': 20}
names = list(data.keys())
values = list(data.values())

fig, axs = plt.subplots(1, 3, figsize=(9, 3), sharey=True)
axs[0].bar(names, values)
axs[1].scatter(names, values)
axs[2].plot(names, values)
fig.suptitle('Categorical Plotting')
```

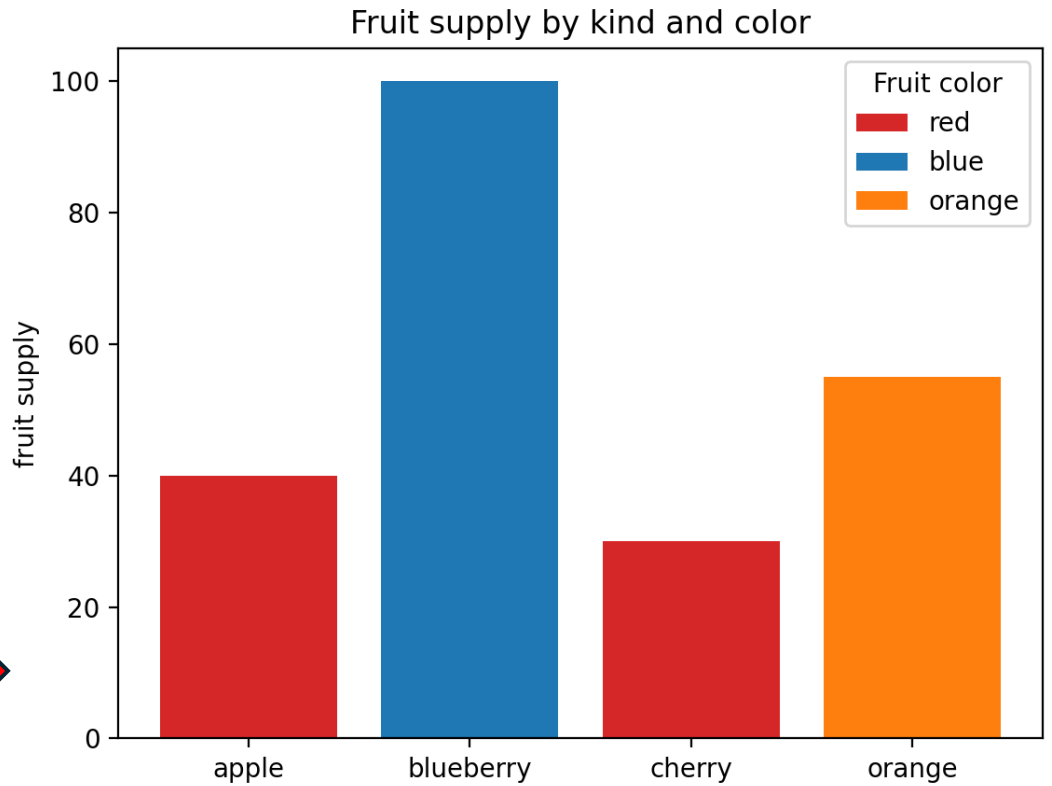


Barplot **Scatterplot** **Lineplot**

https://matplotlib.org/stable/gallery/lines_bars_and_markers/categorical_variables.html#sphx-glr-gallery-lines-bars-and-markers-categorical-variables-py

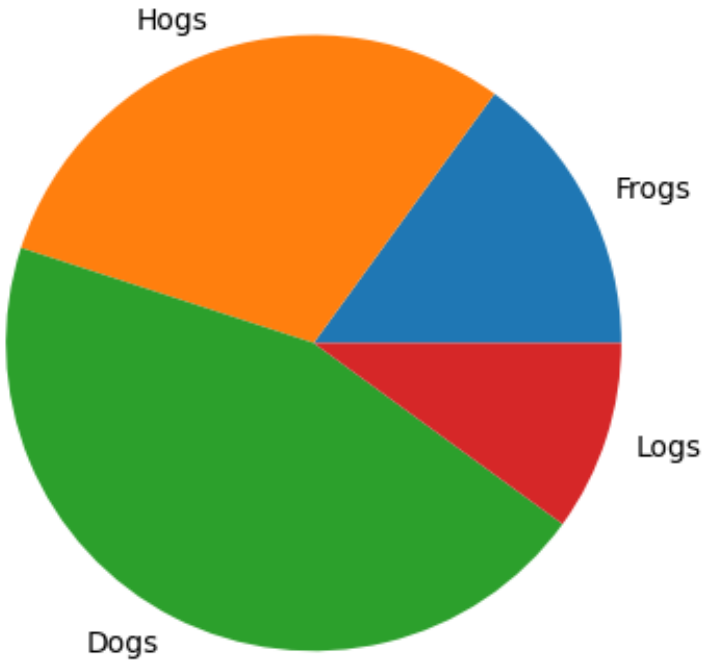
Matplotlib : Lines-bars-and-markers-bar-colors

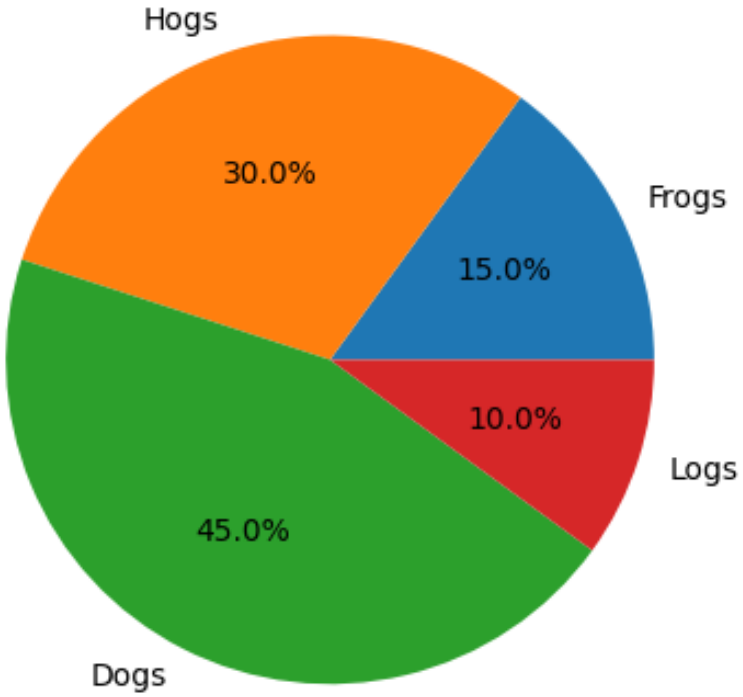
```python
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

fruits = ['apple', 'blueberry', 'cherry', 'orange']
counts = [40, 100, 30, 55]
bar_labels = ['red', 'blue', '_red', 'orange']
bar_colors = ['tab:red', 'tab:blue', 'tab:red', 'tab:orange']

ax.bar(fruits, counts, label=bar_labels, color=bar_colors)

ax.set_ylabel('fruit supply')
ax.set_title('Fruit supply by kind and color')
ax.legend(title='Fruit color')

plt.show()
```

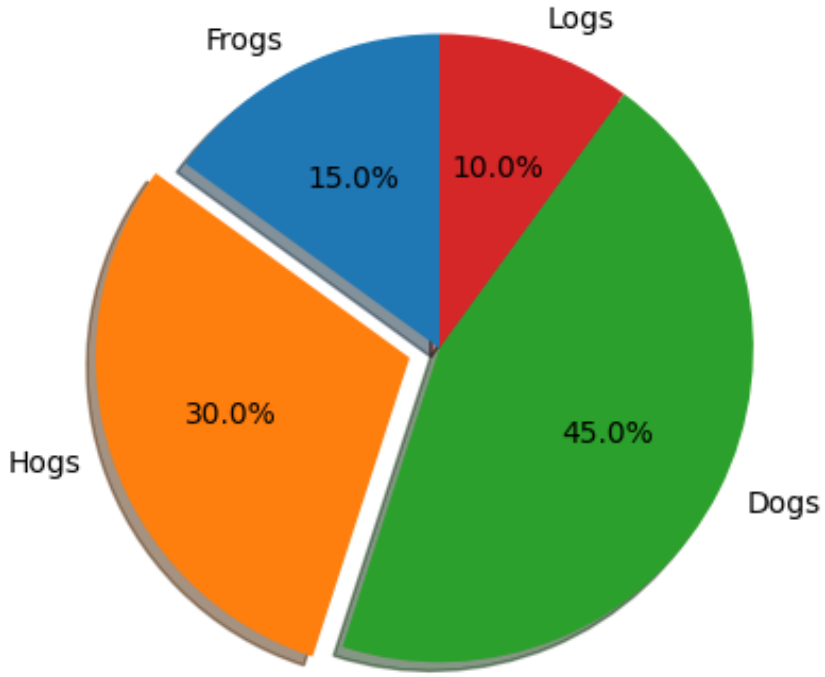


https://matplotlib.org/stable/gallery/lines_bars_and_markers/bar_colors.html#sphx-glr-gallery-lines-bars-and-markers-bar-colors-py

Matplotlib : Pie charts







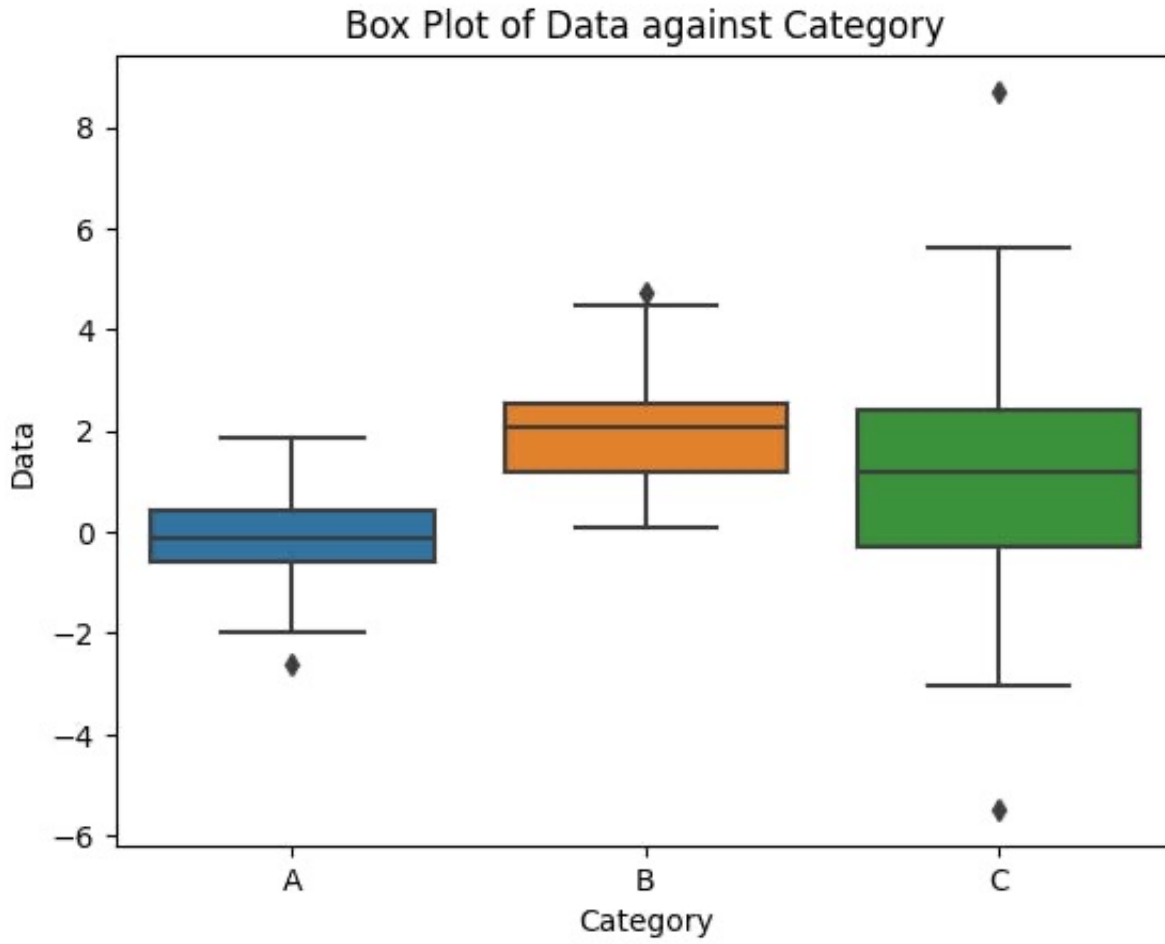
https://matplotlib.org/stable/gallery/pie_and_polar_charts/pie_features.html#sphx-glr-gallery-pie-and-polar-charts-pie-features-py

Seaborn – Examples/Gallery

- https://seaborn.pydata.org/examples/index.html

seaborn : Boxplot

```
import seaborn as sns
import numpy as np
import pandas as pd

# Generate random data
np.random.seed(42)
dataOne = np.random.normal(0, 1, 100)
dataTwo = np.random.normal(2, 1, 100)
dataThree = np.random.normal(1, 2, 100)

# Combine the data into a DataFrame
data = np.concatenate([dataOne, dataTwo, dataThree])
categories = np.repeat(['A', 'B', 'C'], 100)
df = pd.DataFrame({'Category': categories, 'Data': data})

# Create a figure with specific dimensions (width, height)
plt.figure(figsize=(10, 6))

# Create a box plot
sns.boxplot(x='Category', y='Data', data=df)
# Add title
plt.title('Box Plot of Data against Category')
# Show the plot
plt.show()
```



seaborn : Boxplot - code-decoded -I

```
import seaborn as sns
import numpy as np
import pandas as pd
```

•**seaborn (sns):** A Python visualization library based on matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics.

•**numpy (np):** A fundamental package for scientific computing with Python, used for creating and manipulating numerical data.

•**pandas (pd):** A data manipulation and analysis library for Python, which provides data structures like DataFrame to work with structured data.

seaborn : Boxplot - code-decoded -I

```
# Generate random data
np.random.seed(42)
dataOne = np.random.normal(0, 1, 100)
dataTwo = np.random.normal(2, 1, 100)
dataThree = np.random.normal(1, 2, 100)
```

np.random.seed(42): Sets the seed for the random number generator to ensure reproducibility. The seed value 42 ensures that the same random numbers are generated each time the code is run.

dataOne = np.random.normal(0, 1, 100): Generates 100 random numbers from a normal distribution with a mean of 0 and a standard deviation of 1.

dataTwo = np.random.normal(2, 1, 100): Generates 100 random numbers from a normal distribution with a mean of 2 and a standard deviation of 1.

dataThree = np.random.normal(1, 2, 100): Generates 100 random numbers from a normal distribution with a mean of 1 and a standard deviation of 2.

seaborn : Boxplot - code-decoded -III

```
data = np.concatenate([dataOne, dataTwo, dataThree])
categories = np.repeat(['A', 'B', 'C'], 100)
df = pd.DataFrame({'Category': categories, 'Data': data})
```

data = np.concatenate([dataOne, dataTwo, dataThree]): Combines the three datasets into a single array.

categories = np.repeat(['A', 'B', 'C'], 100): Creates an array of category labels ('A', 'B', 'C'), each repeated 100 times to match the data points.

df = pd.DataFrame({'Category': categories, 'Data': data}): Creates a DataFrame with two columns, 'Category' and 'Data', where each row represents a data point and its corresponding category.

seaborn : Boxplot - code-decoded -IV

 # Create a figure with specific dimensions (width, height)
plt.figure(figsize=(10, 6))
Example: 10 inches wide, 6 inches high

 # Create a box plot
sns.boxplot(x='Category', y='Data', data=df)
Creates a box plot using Seaborn. The x-axis represents the 'Category', and the y-axis represents the 'Data'. The box plot visualizes the distribution of the data for each category, showing the median, quartiles, and potential outliers.

plt.title('Box Plot of Data against Category')
Adds a title to the box plot.

seaborn : Boxplot - code-decoded -IV

> Save the plot

```
# Save as PNG
plt.savefig('box_plot.png’)

# Save as PDF
plt.savefig('box_plot.pdf’)

# Save as SVG
plt.savefig('box_plot.svg')
```

End Of the Day -I

- Enjoy the data Visualization.

Thank you!

Reach me @

- Showkat.dar@nih.gov
- Showkat.dar19@gmail.com
- Mob: 443-248-8491
- B –Wing, 10th Floor, LGG, NIA/NIH.