PER SCHOLAS

Explore Matplotlib's Charts and Plots

Introduction

In this Lesson, we will introduce you to the Matplotlib library and the Pyplot module. This will be followed by an exploration and demonstration of graph or chart styling, and you will learn how to create professional-looking line charts.

Prerequisites

Learners need to be proficient in Python programming and Python Pandas. Learners must also have Python and Jupyter Notebook installed and configured on their machines, or must be able to access a cloud-based notebook.

PER SCHOLAS

Table of Contents

Section One

Matplotlib Pyplot: Scatter Plot

Learning Objectives

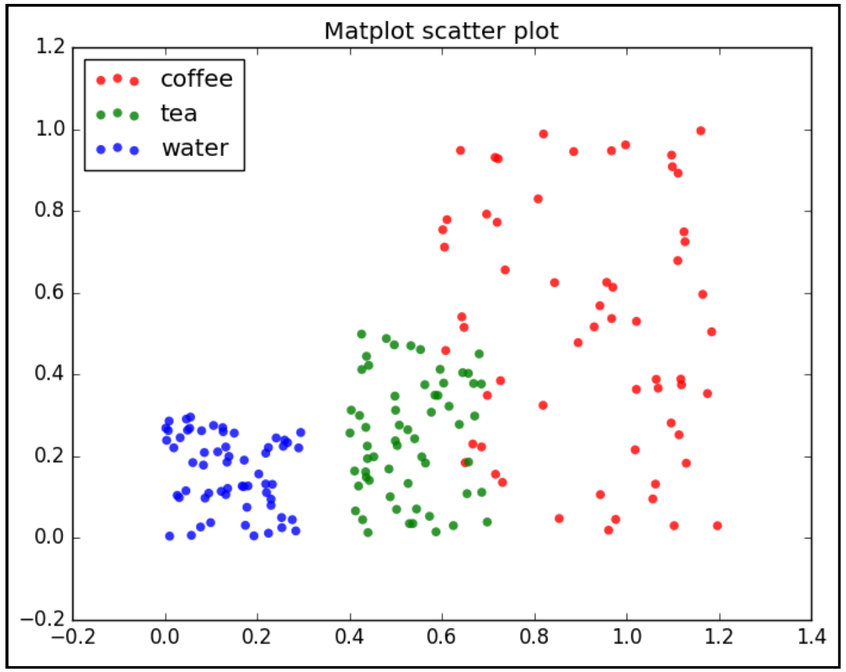
In this section, participants will dive into the world of scatter plots using Matplotlib. They will start by learning the fundamentals of scatter plots and how to create them using Matplotlib.

By the end of this section, participants will be able to:

- Create scatter plots using Matplotlib.
- Describe how to visualize relationships between variables in their datasets effectively.

Matplotlib Pyplot: Scatter Plot

- A scatter plot is a type of data visualization that displays individual data points as dots (circle, or other shape) on a two-dimensional plane, with one variable plotted along the x-axis and another variable plotted along the y-axis.
- Scatter plots are particularly useful for visually examining the relationship between two numerical variables and identifying patterns, trends, and correlations in the data.
- We can create a scatter plot by using **plt.scatter() function:**
- The **plt.scatter()** function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis:



Matplotlib Pyplot: Syntax - Scatter Plot

In Matplotlib Pyplot, scatter plots can be created using the **plt.scatter()** function, which allows users to specify the x and y coordinates of each data point. Additionally, scatter plots in Matplotlib can be customized with various parameters such as **colors, markers, sizes,** and **transparency** to enhance their appearance and interpretability.

```
plt.scatter(x, y, color="red", s='', c='',cmap='',alpha='')
```

Here's a summary of key points to remember about the main input parameters:

- x and y: These parameters represent two variables we want to show the relationship.
- s: Defines the marker size.
- c: Represents the marker color.
- marker: Customizes the shape of the marker.
- cmap: Selects the mapping between values and colors.
- alpha: This parameter is a float number and represents the transparency of the markers.

PER SCHOLAS

Example - Scatter Plot

Click here for the Scatter plot examples - Guided LAB - 344.3.1 - Example Scatter Plots

PER SCHOLAS

Use Cases for Scatter Plot

Scatter plots are commonly used in exploratory data analysis and are especially effective for:

- Visualizing the relationship between two continuous variables: Scatter plots allow analysts to observe how changes in one variable affect the other variable, helping to identify trends and patterns in the data.

- Identifying clusters or groupings within the data: By examining the distribution of data points in a scatter plot, analysts can detect clusters or groupings that may indicate distinct patterns or subpopulations within the dataset.

- Assessing the strength and direction of correlation: Scatter plots provide a visual representation of the correlation between two variables, allowing analysts to determine whether the relationship is positive, negative, or neutral, as well as the strength of the association.

Summary

- Overall, scatter plots are a fundamental tool in data visualization and are widely used across various domains, including statistics, machine learning, and exploratory data analysis, to gain insights into the relationships between variables in a dataset.
- **scatter()** is used to create scatter plots.

Section Two

Matplotlib Pyplot: Bar Chart

Learning Objectives

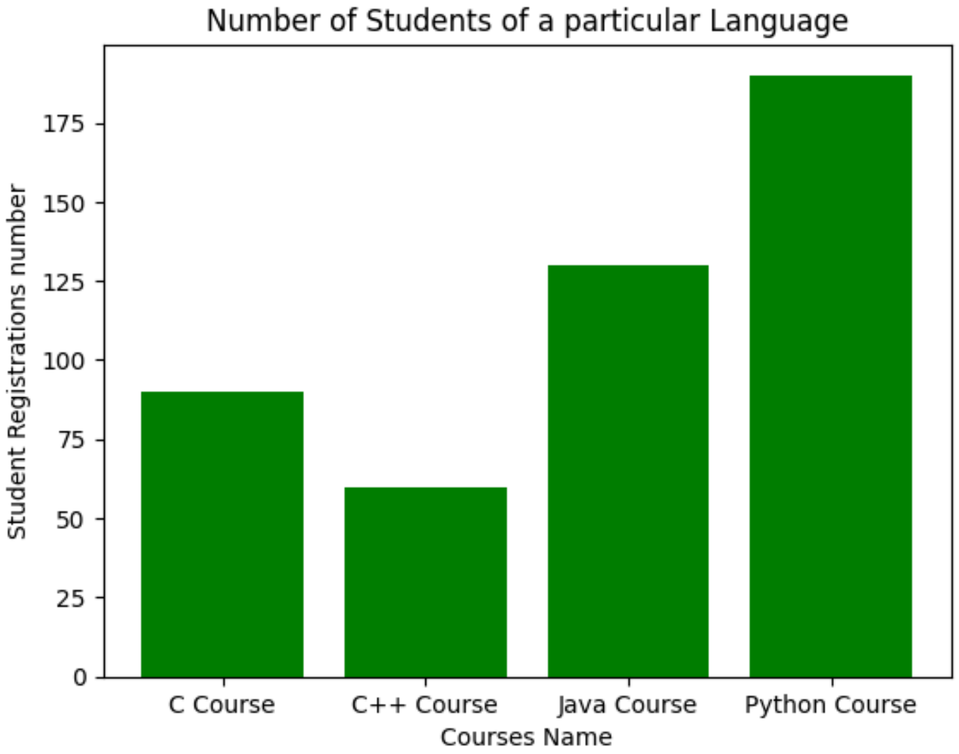
In this section, participants will dive into the world of Bar Chart using Matplotlib. They will start by learning the fundamentals of Bar Chart and how to create them using Matplotlib.

By the end of this section, participants will be able to:

- Create Bar Chart using Matplotlib.
- Describe how to visualize relationships between variables in their datasets effectively.
- Use Matplotlib to create informative and visually appealing bar charts for data visualization tasks.

Matplotlib Pyplot: Bar Chart

- A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars, with heights or lengths proportional to the values that they represent. The bars can be oriented either vertically or horizontally. Typically, the horizontal axis of a bar chart represents categories, while the vertical axis represents their corresponding values. A bar chart is used to compare discrete data, such as occurrences or proportions.



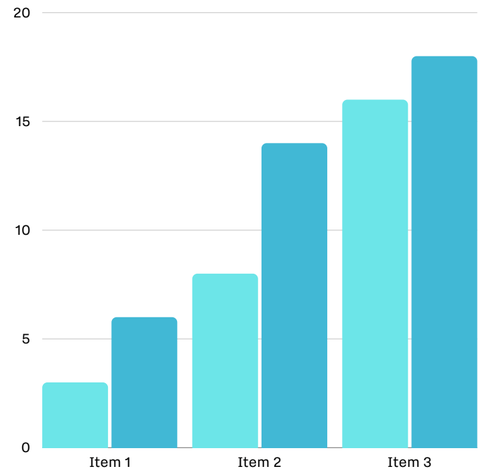
Matplotlib Pyplot: Bar Chart Syntax

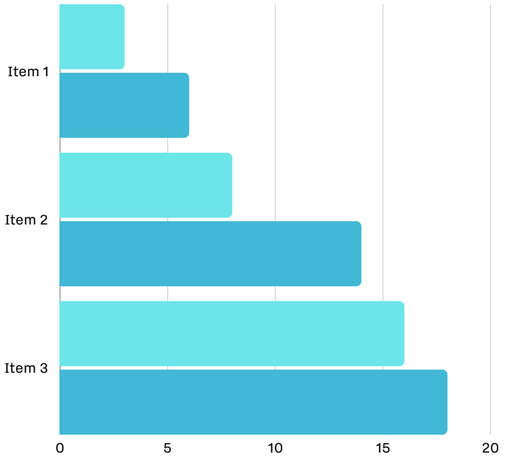
With Pyplot, you can use:

- plt.bar() function to draw bar chart vertically.
- plt.barh() function to draw bar chart horizontally.

The syntax for this function is as follows:

- plt.bar(x, height, width, bottom, align)
- x is a category
- height is the corresponding value.
- width is how wide you want your bars(default value is 0.8).
- bottom is the base of the y-coordinate; in other words, it is the point where your bars start. The (default is 0).
- align is where you want to place your category names. By default, they are positioned at the bar center.





Click here for Bar Charts examples: GLAB - 344.3.2 - Creating Bar Charts

PER SCHOLAS

Practice Task: Analyzing Course Enrollment Data

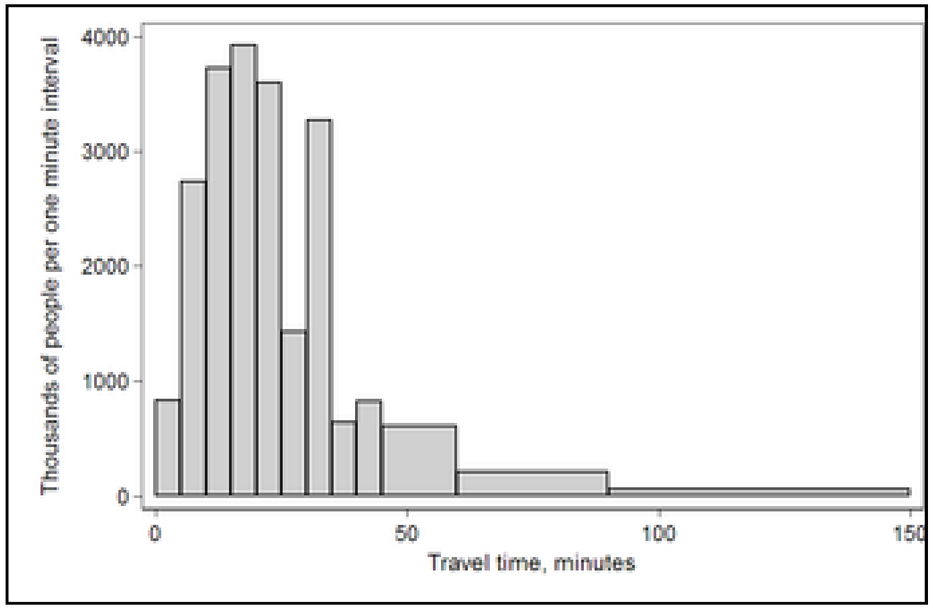
Click here for Challenge task for the practice: PA - 344.3.1 -Practice Activity: Analyzing Monthly Expenses

Section Three

Matplotlib Pyplot: Histogram

Matplotlib Pyplot: Histogram Chart

- A histogram is a graphical display of data that organizes groups of data points into ranges, represented by bars. While it resembles a bar chart, the key difference lies in its usage: a bar chart represents categorical data, whereas a histogram is used exclusively for numerical data. Examples include age groups or test scores. Instead of displaying every single age, a histogram groups data into intervals, such as 20-25, 25-30, and so on.
- In other words, A histogram is an accurate representation of the distribution of numerical data. It differs from a bar graph in the sense that a bar graph relates two variables, but a histogram relates only one.



img src: wikipedia.org

- **When to use it?**
  - When estimating the probability distribution of a continuous variable (quantitative variable).
  - When organizing large amounts of data and producing a visualization quickly, using a single dimension.

PER SCHOLAS

Matplotlib Pyplot: Syntax - Histogram Chart

The **plt.hist()** function allows for the creation of histograms in Python, using matplotlib based on the elements in a given vector (usually numerical data) as follows:

- By setting the orientation argument to "horizontal," you can flip the axes to create a horizontal histogram.
- The **hist()** function also allows the creation of cumulative histograms by setting the argument cumulative to **True**.

Examples: Visit the following labs:

- Guided LAB - 344.3.3 - Creating Histogram.
- Guided LAB - 343.3.4 - Exploring Data Distributions with Histogram.

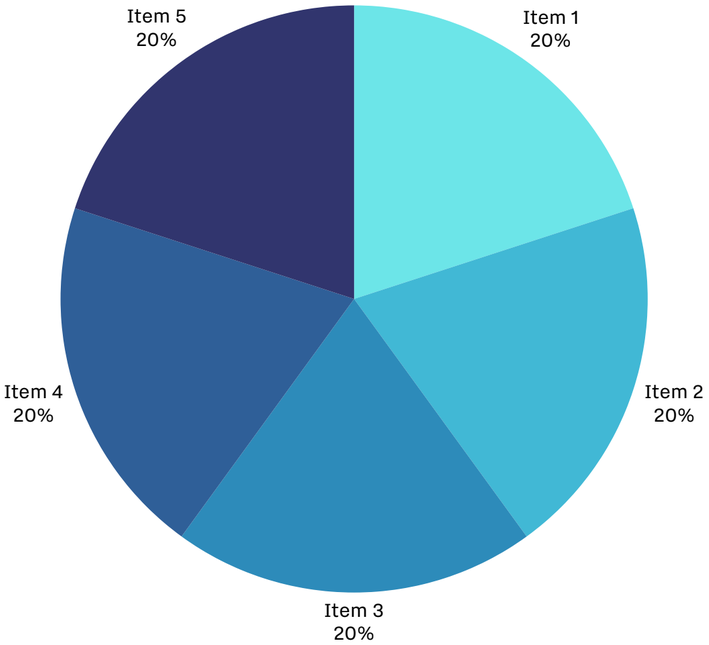
Section Four

Matplotlib Pyplot: Pie Chart

PER SCHOLAS

Matplotlib Pyplot: Pie Chart

- A pie chart is a circular statistical graphic used to illustrate numerical proportions or percentages. (Try to use positive values only.)
- The pie chart total area of the circle is the data percentage. **The area of slices (also called wedges)** represents the percentage of data parts and shows the relation between them.
- Pie charts are good when we need to compare parts of a whole.
- Pie charts are useful for visualizing the relative sizes of different categories or parts of a whole. They are commonly used to represent categorical data and are particularly effective when there are only a few categories to display. Pie charts make it easy to compare the proportions of different categories at a glance.
- They are widely used in business presentations, reports, and academic research to communicate data in a visually appealing and easy-to-understand manner.



PER SCHOLAS

Matplotlib Pyplot: Syntax - Pie Chart

The **plt.pie(data)** function allows for creating a pie chart in Python when using matplotlib.

There are many other optional parameters that can pass to plt.pie() function. Some include:

- **explode** allows separating slices of a pie chart.
- **labels** is a list of strings that sets the label of each corresponding slice.
- **labeldistance** determines the radial distance at which pie labels are drawn (default is 1.1).
- **color** colors slices.
- **shadow** creates the shadow of a slice.
- **startangle** lets you choose the starting angle of a plot.
- **wedgeprops** lets you tune various slice parameters.
- **radius** sets the radius of the circle (by default, it is 1).

Examples: Guided Lab - 344.3.5 - Creating Pie Chart

Section Five

Matplotlib Pyplot: Box plot

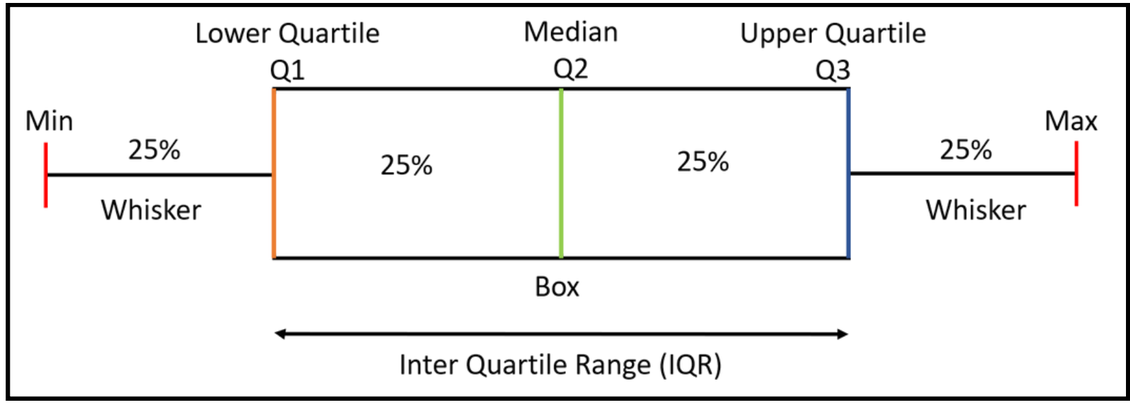
PER SCHOLAS

Matplotlib Pyplot: Box Plot

- The term **“box plot”** refers to an outlier box plot. This plot is also called a box-and-whisker plot, and is a convenient way to visualize the distributions of numerical data using quartiles. Box plots are widespread in descriptive statistics and will allow you to quickly explore one or more datasets.
- A box plot gives a five-number summary of a set of data, which includes:
  - **Minimum** – It is the minimum value in the dataset, excluding the outliers.
  - **First Quartile (Q1)** – 25% of the data lies below the first (lower) Quartile.
  - **Median (Q2)** – The midpoint of the dataset, where half of the values lie below and half above.
  - **Third Quartile (Q3)** – 75% of the data lies below the third (upper) Quartile.
  - **Maximum** – the maximum value in the dataset, excluding the outliers.

Note: The box plot shown in the diagram is a perfect plot with no skewness. The plots can have skewness, and the median might not be at the center of the box.
The area inside the box (50% of the data) is known as the Inter Quartile Range (IQR). The IQR is calculated as – IQR = Q3-Q1

- **Click here to learn more about Box Plots**.



PER SCHOLAS

Matplotlib Pyplot: Syntax - Box Plot

The **plt.boxplot()** function allows for creating box plot charts in Python when using matplotlib.

There is a great number of optional arguments. Here are some of them:

- **vert** if False, produces a horizontal box plot.
- **labels** is a sequence of strings that sets a label for each dataset.
- **showmeans** if True, displays the mean values as a triangle on the box.
- **meanline** if True, alongside with showmeans = True displays the mean as a line.
- **boxprops** medianprops, meansprops, whiskerprops, capprops, and flierprops allow us to change the properties of the box, median, mean, whiskers, caps, and outliers, respectively.
- **patch_artist** accepts Boolean values -- either true or false -- and this is an optional parameter.

Examples: Guided LAB - 344.3.6 - Example Box Plot.ipynb

PER SCHOLAS

Hands-On Activity

Complete the following practice assignment. You can find this assignment on Canvas under the Assignment section.

- **PA - Practice Activity -344.3.2 - Pie-Charts-Box-Plots-Scatter-Plots-and-Box-Plots.**

Note: This assignment covers some advanced properties and attributes of the plots, which may require additional research to fully understand and utilize effectively.

Section Six

Matplotlib Pyplot: Area plot

Learning Objectives

In this section, we will explore the concept of area plots, also known as stacked line plots, and learn how to create them using Matplotlib.

The objective of this lesson is to gain proficiency in creating area plots using Matplotlib. By the end of this lesson, learners will be able to:

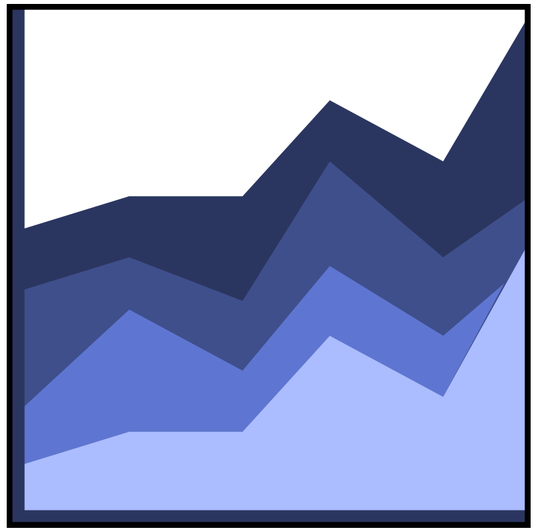
- Describe the concept and purpose of area plots.
- Visualize data with area plots to interpret trends effectively.
- Create basic area plots using Matplotlib.
- Explore advanced features like filling between lines to highlight specific areas.
- Analyze trends and patterns in area plots to derive meaningful insights from data.

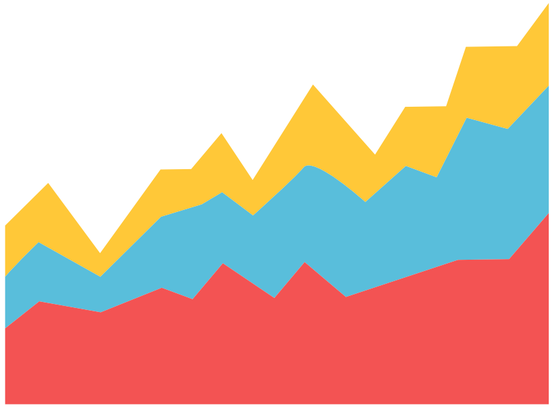
PER SCHOLAS

Matplotlib Pyplot: Area Plot

- An Area plot, also known as a **filled line plot**, is a type of plot in matplotlib that displays the magnitude of a variable over a range of values. It is similar to a line plot, but the area between the plotted line and the x-axis is filled with color, making it easier to visualize the cumulative effect of the data.
- Area plots are particularly useful for illustrating trends and changes over time or across different categories. They can show how the total magnitude of variable changes as different components contribute to it.
- Overall, area plots provide a visually appealing way to represent data, and are commonly used in data analysis and visualization tasks.

Area Plot





Area Plot

PER SCHOLAS

Matplotlib Pyplot: Syntax - Area Plot

In Matplotlib, you can create an Area plot using the **fill_between()** function, specifying the x-values, y-values, and optionally, the baseline value.

pyplot.fill_between(x, y1, y2=0, where=None, interpolate=False, step=None, *, data=None, **kwargs)

Fill the area between two horizontal curves. The curves are defined by the points (x, y1) and (x, y2). This creates one or multiple polygons describing the filled area.

- Click here for more details for **fill_between()** function

Examples: Visit following lab:

Guided LAB - 344.3.7- Example Box Plot.ipynb

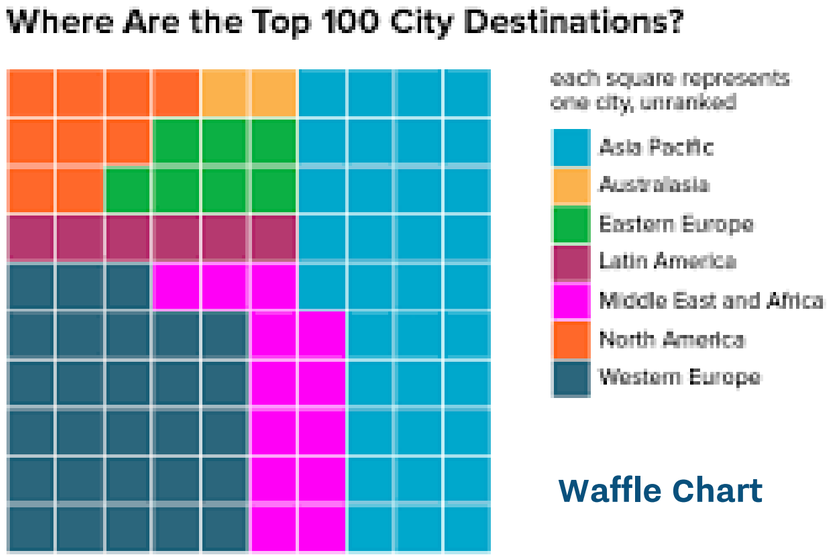
Section Seven

Matplotlib Pyplot: Waffle Chart

PER SCHOLAS

Matplotlib Pyplot: Waffle Chart

- A Waffle Chart, also known as a square pie chart, is a visualization technique that represents data using a grid of small squares or rectangles, each of which represents a proportion or percentage of the whole.
- In a Waffle Chart, the squares are filled or shaded based on the proportion or percentage of each category, allowing viewers to quickly grasp the relative sizes of different categories visually.



Waffle Chart

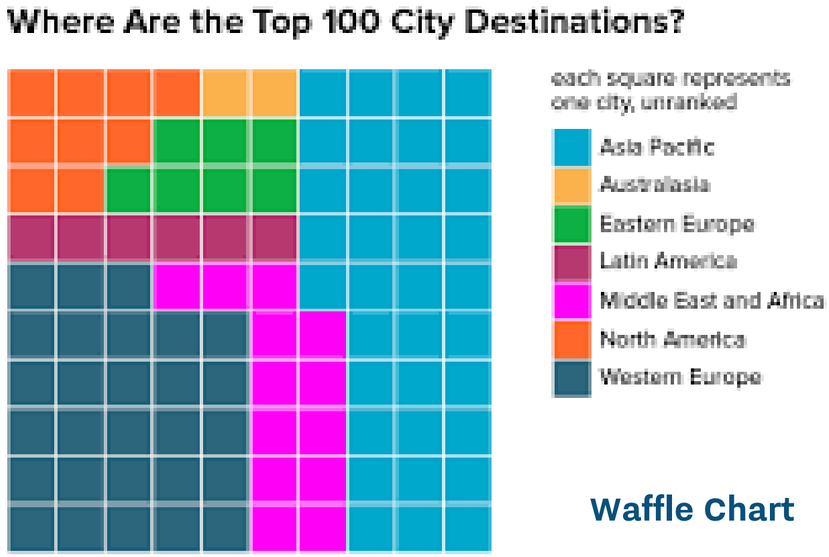
- Waffle Charts are particularly useful when you want to convey proportions or percentages in a visually engaging and easy-to-understand manner.

PER SCHOLAS

Matplotlib Pyplot: Waffle Chart Use Cases

You can use Waffle Charts for:

- Market share analysis.
  - Visualize market share data.
- Demographic representation.
  - Display demographic data.
- Project progress tracking.
  - Represent completion status of tasks or milestones.
- Budget allocation.
  - Demonstrate allocation of budgetary resources.
- Survey responses.
  - Summarize survey responses.
- Election results.
  - Provide a clear visualization of voting outcomes.
- Product sales analysis.
  - Illustrate product sales .



Waffle Chart

Matplotlib Pyplot: PyWafflelibrary

- Python enables the creation of waffle charts through the **PyWafflelibrary**.
- The very first thing to do is to install the library by using following command:

```
pip install pywaffle
```

- Then we can import the pywaffle library by using the command:

```
from pywaffle import Waffle
```

Examples: Visit following lab:

Guided LAB - 344.3.8 - Example Waffle Chart.ipynb

PER SCHOLAS

Knowledge Check

- What is a scatter plot?
- What is a pie chart?
- Why do we use Area Plots?
- Why do we use Box Plots?

Section Eight

Matplotlib Pyplot: Maps

Learning Objectives

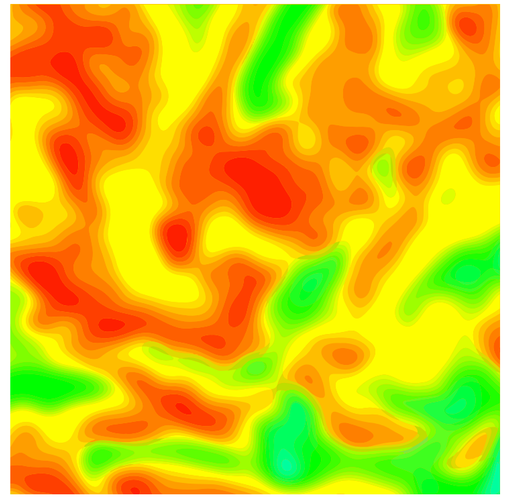
In this section, we will explore the concept of Map plots or charts. We will also demonstrate how to use the Folium library and choropleth map.

The main objective of this lesson is to create heat plots and choropleth maps using Folium. By the end of this lesson, learners will be able to:

- Describe the concept and purpose of Geographical data plots.
- Create heat maps using Matplotlib.
- Utilize the Folium Library for visualizing geospatial data.
- Create a choropleth map using the Folium library.
- Describe the GeoJSON file.

Matplotlib Pyplot: Heatmaps

- Heatmaps represent data using colors or gradients on a continuous scale, often superimposed on either a geographic or non-geographic grid.
- Heatmaps depict relationships between variables using colors rather than numerical values.
    - Heatmaps illustrate the density or intensity of data points across a space, with colors indicating areas of higher or lower concentration.
    - There are two main types of heatmaps:
        i. **Cluster heatmaps**, which display a cell-matrix of different colors.
        ii. **Spatial heatmaps,** where the variable variation is considered continuous.



Heat map

Matplotlib Pyplot: Syntax - Heatmaps

- In matplotlib, you can create heatmaps using **pyplot.imshow()** function.
- Syntax: **imshow(X, cmap=None, alpha=None)**
  - X :- input data matrix, which is to be displayed.
  - cmap :- colormap we use to dispay the heatmap.
  - alpha :- specifies the opacity or transpiracy of the heatmap.
  - aspect: {'equal', 'auto'} or float,(default: 'equal'). The aspect ratio of the plot axes.
  - interpolation: str, (default: 'antialiased') - the interpolation method used.

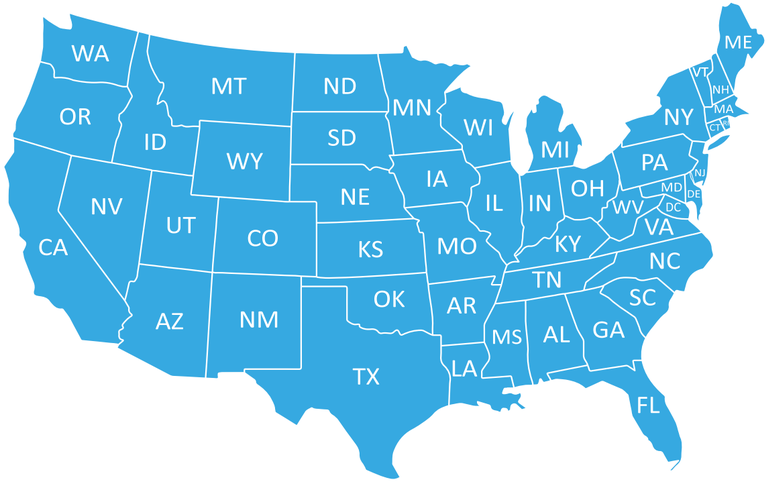
Examples: Visit following lab:

Guided LAB - 344.3.9 - Example Heatmap chart

PER SCHOLAS

Matplotlib Pyplot: Choropleth Maps

- The **Choropleth map** is one such representation of geospatial data. It can be used to analyze the distribution of data in geographical regions such as countries, states, or regions, based on the value of a particular variable, population density, GDP per capita of countries.
- Each area is typically represented as a distinct polygon, and the color intensity or shading represents the magnitude of the variable being measured within that area.
- Python has a list of libraries (geopandas, bokeh, cartopy, **folium**, ipyleaflet, etc.) that let us create choropleth maps.



Choropleth map

PER SCHOLAS

Folium Library for Visualize Geospatial Data (1 of 2)

- Folium is a powerful data visualization library in Python that was built primarily to help people visualize geospatial data.
- With Folium, you can create a map of any location in the world using latitude and longitude values. You can also create a map and superimpose markers and clusters on top of the map for interesting visualizations.
- With Folium, we can create various types of maps, including:
  - **Static maps:** Simple maps that display geographical data without any interactivity.
  - **Interactive maps:** Maps that allow users to interact with the data by zooming in, panning, clicking on markers, and more.
  - **Choropleth maps:** Maps that visualize statistical data by shading or coloring different geographical areas based on their values.
- The Folium library provides a simple and intuitive way to generate maps using data from various sources, including GeoJSON file, Pandas' DataFrames, and NumPy arrays.

Folium Library for visualize geospatial data (2 of 2)

Download and Installation

- Python enables the creation of geo maps through the folium.
- The very first thing to do is to install the library by using following command:

```
pip install folium
```

- This command will download and install the Folium package and its dependencies from the Python Package Index (PyPI). Once the installation is complete, you can import Folium in your Python scripts or Notebooks and start using it for creating interactive maps.

Verification: By using below code, we can verify the folium installation:

```
import folium
print('folium ver: ', folium.__version__)
```

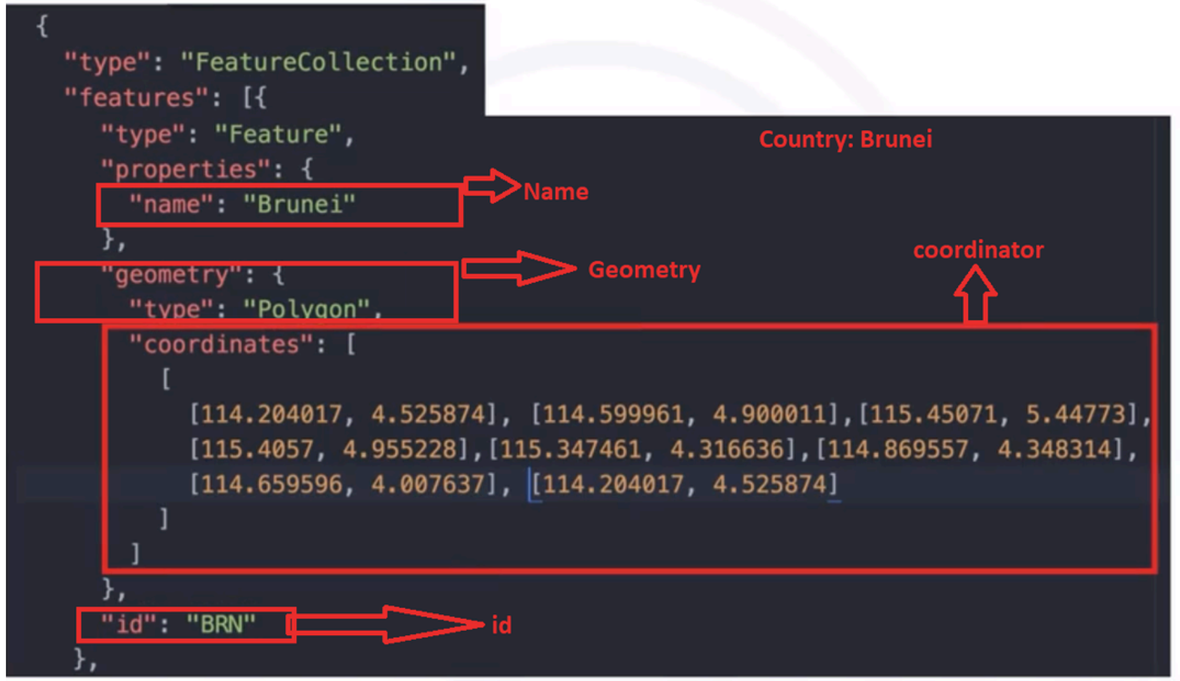
Folium Library - Syntax: folium.Map()

- In Folium, folium.Map() is a class constructor used to create map objects. It is used to initialize a map with specified parameters such as location, zoom level, and choice of tiles. When called, it returns an instance of the Map class, which represents the map object that can then be customized and displayed.
- There are several parameters, but at first you only need to know following five basic parameter
  - location (tuple or list, default:None) : Latitude & Longitude of Map
  - width & heigth (int, string, default :'100%'): int is pixel, str is percentage (100, or '100%')
  - min_zoom (int, default:0): Minimum allowed zoom level for the tile layer that is created.
  - max_zoom (int, default:18): Maximum allowed zoom level for the tile layer that is created.
  - zoom_start(int, default:10): Initial zoom level for the map (Map scale)
  - tiles : style of map (default=OpenStreetMap, Stamen Terrain, Stamen Toner, Mapbox Bright, Mapbox Control Room, etc.)

What is GeoJSON file

- GeoJSON files are written in the same format as JSON files. They are commonly used for representing geographic features such as points, lines, and polygons, along with their associated attributes.
- To create a choropleth map of a region of interest, Folium requires a GeoJSON file that includes geospatial data for the region.
- For a choropleth map of the world, we would need a GeoJSON file that lists each country and provides geospatial data to define its borders and boundaries.

- As shown in the screenshot, Here is an example of what a GeoJSON file would include about each country.
  - The example here pertains to the country **Brunei**.
  - As you can see, the file consists of its name, geometry, ID and coordinates(shape) that define its borders and boundaries.



PER SCHOLAS

Example - Visualize geospatial data with Folium

- In the below lab, we will demonstrate how to use Folium and how to visualize geospatial data with Folium.
  - **Guided LAB - 344.3.10 - How to use Folium for generating Maps with Python**
  - **Additional resource for the practice**.

- The below lab aims to create a **choropleth map** that visualizes geospatial data. This lab utilizes GeoJSON data for country boundaries and pandas DataFrame for immigration data.
  - **Guided LAB - 344.4.11 - How to use Folium for generating Choropleth Maps**.

Summary

- Matplotlib is a versatile plotting library that offers a flexible interface for creating various types of plots.
- The pyplot module of matplotlib contains a collection of functions that can be used to work on a plot. The plot() function of the pyplot module is used to create a figure. A figure is the overall window where the outputs of pyplot functions are plotted.
- A scatter plot displays values pertaining to typically two variables against each other.
- The process of creating a scatter plot involves importing Matplotlib to visualize a large set of data.
- A pie chart is a circular statistical graphic, divided into segments, to illustrate numerical proportion.
- The process of creating a pie chart involves importing Matplotlib to represent a large set of data over a period of time.
- A box plot is a way of statistically representing given data distribution through five main dimensions.
- The five main dimensions are minimum, first quartile, median, third quartile, and maximum.
- You can create a box plot using Matplotlib.

Summary

- Folium is a data visualization library in Python that helps people visualize geospatial data.
- With Folium, you can create maps of different styles, such as street-level maps, stamen maps, and more.
- A feature of Folium is that you can create different map styles using the tiles parameter.
- With Folium, you can easily add markers on maps.
- The ‘location’ parameter specifies the latitude and longitude coordinates of the center point of the map.
- Markers play a vital role in enhancing interactivity and adding context to maps.
- The folium.Marker() function specifies location parameters.
- The popup parameter provides a label upon being clicked.
- Markers can be created using “feature group.”
- A choropleth map is a thematic map in which areas are shaded or patterned in proportion to the measurement of the statistical variable.
- When creating a choropleth map, Folium requires a GeoJson file that includes geospatial data of the region.

Summary

In summary, while both heat maps and choropleth maps are used for spatial visualization, choropleth maps focus on aggregating data by geographic areas and coloring them accordingly, whereas heat maps represent the density or intensity of data across a continuous space.

Heatmaps illustrate the density or intensity of data points across a space with colors indicating areas of higher or lower concentration. In contrast to choropleth maps, heatmaps do not depend on predefined geographic boundaries. They can visualize data at a more granular level such as individual points on a map or pixels in an image.

References

https://www.analyticsvidhya.com/blog/2020/06/guide-geospatial-analysis-folium-python/

https://medium.com/@sawsanyusuf/data-visualization-with-python-10-choropleth-maps-df7ab3118c3a

PER SCHOLAS

Questions?

PER SCHOLAS

UNLOCKING POTENTIAL

CHANGING THE FACE OF TECH