**Data Visualization using Python**

**What is Data Visualization?**

Data visualization 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 visualization can be utilized for a variety of purposes, and it’s important to note that is not only reserved for use by data teams. Management also leverages it to convey organizational structure and hierarchy while data analysts and data scientists use it to discover and explain patterns and trends.

Building visuals convey a clear description of the data to disclose findings to the audience. It helps in presenting results from exploratory data analysis, clear communication of data finding, sharing unbiased representation and in supporting recommendations to stakeholders.

But how do you decide what should your data represent? When you analyze the data, you need to ask yourself a few questions that will help you decide how your analysis will help the audience at large:

* Who is your target audience?
* What is the main message you’re trying to convey?
* What do you want your audience to do about it?
* \*Be as specific as possible when exploring these questions.\*

**Matplotlib**

Matplotlib is a cross-platform, data visualization and graphical plotting library (histograms, scatter plots, bar charts, etc) for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib’s APIs (Application Programming Interfaces) to embed plots in GUI applications.

A Python matplotlib script is structured so that a few lines of code are all that is required in most instances to generate a visual data plot. The matplotlib scripting layer overlays two APIs:

* The pyplot API is a hierarchy of Python code objects topped by matplotlib.pyplot
* An OO (Object-Oriented) API collection of objects that can be assembled with greater flexibility than pyplot. This API provides direct access to Matplotlib’s backend layers.

Take a look at the image below to understand the architecture of Matplotlib.

A screenshot of a computer

Description automatically generated

Let's understand these layers in a little more detail.

Backend Layer: Defines and encompasses the area on which the figure is drawn

Renderer: Knows how to draw on the Figure Canvas

Event: User input (like key strokes, etc)

Artistic Layer: Artist knows how to use the Renderer and draw on the canvas. Primitive object consists of a line, rectangle, circle or text. The composite object consists of figure or axes. Each composite artist may contain other composite artists as well as primitive ones. The artist layer is syntactically heavy.

Scripting Layer: This layer involves the matplotlib.pyplot interface. It automates the process of defining a canvas and defining a figure artist.

**Basic Plotting**

Apart from Matplotlib, basic plotting can be done using different packages or algorithms in Python. Some of these are listed below:

* Jupyter Notebook: Jupyter notebook is an open source web app that allows one to create and share documents that contain live code visualizations and explanatory text
* Pandas: Pandas has a built-in implementation of Matplotlib
* Plot function: Can be used to create all visualization tools such as histograms, charts, box plots etc.
* Notebook backend:  If a plt function is called, it checks to see if active figures exist and applies them. If no active ones exist, it renders a new one.
* Magic Function:  % Matplotlib; limitation is that you cannot modify a figure once it is rendered

**Let's Practice**

In the past lesson, we learnt about Matplotlib, its applications and the architecture of it. We also learnt about different functions that can be used in Python for basic plotting. In this lesson, we will practice basic plotting using matplotlib.

**How to:**

You can use the Python editor of your choice but it is highly encouraged that you use Google Colab.

The exercise sheets and the datasheet for this exercises are attached below:

A screenshot of a computer

Description automatically generated

**Basic Visualization Tools**

Area Plot: An area plot depicts accumulated totals using numbers or percentages over time. It is based on the line plot and is commonly used when comparing 2+ quantities. An area chart is typically used with multiple lines to make a comparison between groups (aka series) or to show how a whole is divided into component parts. You can have an overlapping area chart or a stacked area chart.

Histograms: Histograms are a way of representing the frequency distribution of a numeric dataset. It separates data into bins; assigns each datapoint to a bin; counts the number of datapoints that have been assigned to each bin. It is used when the data is numerical and you want to see the shape of the data’s distribution, especially when determining whether the output of a process is distributed approximately normally.

Bar Charts: It is arguably most popular visualization tool. Bar charts aka bar graph is a chart where the length of each bar is proportionate the item it represents. Bar charts are mainly used to visualize discontinuous (or discrete) data or to show the relationship between a part to a whole. For multidimensional data, you can choose a Stacked Bar chart, Grouped Bar chart, or Percent Bar chart.

**Specialized Visualization Tools**

Pie Charts: Pie chart is a circular statistical graphic divided into slices to illustrate numerical proportions. Pie charts can be effective in showing the contributions of data segments as a percentage of a whole.

Box Plots: Box Plots represent data through 5 dimensions: minimum, first quartile, median, third quartile, and maximum. It is similar to a histogram but is usually better for showing several simultaneous comparisons such as data grouped by month, etc.

Scatter Plots: Scatter plots display values pertaining to two variables against each other: dependent vs. independent. It also determines whether or not a correlation exists. Researchers tend to use scatter graphs when they are faced with very large sets of data. Scatter graphs allow researchers to identify anomalies in the data more easily as well as the overall trend and relationship between the variables.

**Let's Practice**

In the past lesson, we learnt about the basic and specialized visualization tools and their applications. You also reflected on the real use case scenarios for these. In this lesson, we will practice plotting those charts.

Provided below is a weblink to several datasets. Try plotting the visualizations discussed above using any 3 sets of data per your scrum group. Try to implement using various libraries/packages; matplotlib, seaborn, plotly. Explain how your group approached your implementation and what you are able to gather about the data from your visualizations.

<https://courses.washington.edu/b517/Datasets/datasets.html>

**Reflect**

What was your rationale to choose a particular visualization too when trying to plot the datasets? Why? Is this tool the most appropriate for representing/analyzing the dataset you selected?

**Advanced Visualization Tools**

Waffle charts: Waffle charts are square or rectangular displays made up of smaller squares in a grid pattern. Most commonly, it is a 10 x 10 grid, but they can be any dimension you want them to be, and this will depend on the data you are looking to display. Each square within the grid is coloured based on a category and represents a portion of the whole. From these plots, we can see contributions of individual categories or display progress towards a goal.

Word Cloud: Word clouds or tag clouds are graphical representations of word frequency that give greater prominence to words that appear more frequently in a source text. Word Clouds  can assist evaluators with exploratory textual analysis by identifying words that frequently appear in a set of interviews, documents, or other text. It can also be used for communicating the most salient points or themes in the reporting stage.

**What is Seaborn?**

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

With seaborn, you can achieve multifaceted visualizations based on the parameters you choose in your code and the best part, it helps you utilize 5 times less code.

**Geospatial Data**

As kids, we all were fascinated by looking at colorful interactive informational maps that provided information on Population, weather, street addresses. As a Data Scientist, you can now actually create these maps. Let's find out how:

**Using Folium to visualize Geospatial Data**

Folium is a powerful Python library that helps you create several types of Leaflet maps. By default, Folium creates a map in a separate HTML file. Since Folium results are interactive, this library is very useful for dashboard building. You can also create inline Jupyter maps in Folium.

Folium builds on the data wrangling strengths of the Python ecosystem and the mapping strengths of the Leaflet.js library. Using Folium, you can manipulate your data in Python, then visualize it in a Leaflet map.

Folium enables you to generate a base map of specified width and height with either default tilesets (i.e., map styles) or a custom tileset URL. OpenStreetMap, Mapbox Bright, Mapbox Control Room, Stamen (incl. Terrain, Toner, and Watercolor), Cloudmade , Mapbox, and CartoDB (incl. positron and dark\_matter) are some of the tilesets available by default with Folium.

Folium is a great library to build interactive maps and allows your visualizations to speak for themselves. If you want to have more fun creating different maps (including heat maps with markers), read through this [article(opens in a new tab)](https://medium.com/datasciencearth/map-visualization-with-folium-d1403771717) and try all the codes mentioned.

**Choropleth Maps**

Choropleth maps are built over Folium. These are thematic maps where areas are shaded or patterned in proportion to the measurement of the statistical variable displayed on the map like population or income. The higher the measurement , the darker is the color.

To create choropleth map using Folium, a Geo JSON file is needed. The steps are as follows:

* First, create a map using Folium
* Then, to convert it to a choropleth map, define a variable that points to the JSON file
* Apply the choropleth function identifying the columns you want to highlight as well as the country names

The following image is a good recent example of a choropleth map that shows the risk of Wildfires all across Canada. (Image borrowed to be used a reference from Natural Resources Canada and is available for use with citation credits)

**Let's Practice**

In the past lesson, you learnt about Seaborn library and how it can be used for plotting meaningful and interactive visualizations. You also learnt about plotting Geospatial data using Folium and Choropleth. In this lesson, we will practice the same.

**How to:**

You can choose any Python editor of your choice, but it is highly recommended that you use Google Colab. The instructions for the practice exercises are listed below:

Using the same datasets that you selected in Lesson 1, choose one or two charts from the list below that might be better for visualizing your chosen dataset or would help tell the story hidden in your sample.

* Waffle charts
* Word Clouds
* Seaborn & Regression Plots
* Visualizing Geospatial Data - Folium, maps with markers & Choropleth maps

The datasets used in Lesson 1 are attached below for your reference.

<https://courses.washington.edu/b517/Datasets/datasets.html>

**What is Plotly?**

The plotly Python library is an interactive, open-source plotting library that supports over 40 unique chart types covering a wide range of statistical, financial, geographic, scientific, and 3-dimensional use-cases.

Built on top of the Plotly JavaScript library (plotly.js), plotly enables Python users to create beautiful interactive web-based visualizations that can be displayed in Jupyter notebooks, saved to standalone HTML files, or served as part of pure Python-built web applications using Dash. The plotly Python library is sometimes referred to as "plotly.py" to differentiate it from the JavaScript library.

Plotly express can be used to create basic charts, statistical charts, scientific charts, financial charts, maps and 3-D charts.

**What is Dash?**

Dash is a python framework created by plotly for creating interactive web applications. Dash is open source and using this framework the application build is viewed on the web browser.

Dash applications are made up of 2 building blocks :

* Layout: Layout describes the look and feel of the app, it defines the elements such as graphs, dropdowns etc and the placement, size, color etc of these elements. We can use dash html component and dash core components for this layout.
* Callbacks: It is used to bring interactivity to the dash applications. These are the functions using which, for example, we can define the activity that would happen on clicking a button or a dropdown.

Dash has two components:

* Dash HTML components: we can create and style HTML content such as headings, paragraph, lists.
* Dash Core components: we can create graphs, dropdowns, sliders.

Using Dash in Python requires the import of a few packages and the need to initialize the package when executing the code. To make the dashboard interactive, we make the use of the "callback" function. For example, we can write, click and select some data and based on that our dashboard can be changed.

**Let's Practice**

In the past lesson, we learnt about creating visualizations using Plotly and Dash. We learnt about Dash components and the Callback functions that help create interactive dashboards. We will practice creating these visualizations in this lesson.

**How to?**

You can choose any Python editor of your choice, but it is highly recommended for you to use Google Colab.

Using the same datasets that you selected in Lesson 2 and 3 of this course, implement a simple dashboard. The datasets are attached below for your reference.

<https://courses.washington.edu/b517/Datasets/datasets.html>