

Plotly: Data interpretation and Visualization



Name: Patrick Delaney

T-Number: T00144648

Module: Advanced Scripting

Lecturer: Owen Mackessy

SubmissionDate: 02/05/19

# **Introduction:**

For this project the basis is to look at functional API’s or libraries that can be used as a means of improving or developing better practices as a system administrator. One such action is to retrieve user entered data and present it in a visual manner to provide ease of understanding of the complexity or severity of the data presented. Simplifying and Automating such developments can prove useful for a company.

# **Technology:**

The technology to be used to present this simplification of data manipulation and visualization is with the use of the API Plotly. Plotly is an API that is designed to take in values that are assigned to storage containers to be transcribed to visual representations of the values provided. With the use of file manipulation API Panda and the calculation API Numpy values and titles can be assigned as appropriately needed to construct a diagram as needed.

One of the main requirements of this however is data needs to be constructed in a format easily read by the API, the one most commonly used is comma separated value file(csv). The format of this file is rows and columns of data presented in a way that values can be associated with an index value or a title.

# **Goals:**

The goal of the usage of the API Plotly is to gain a foundational understanding of the construction of complex data files and the ease at which an API can interpret the values given.

Once looked at Plotly can determined to be used in certain environment and a foundational course of learned can be undertaken to show the complexity of the features available to it and the differing visual aids that can be created through its use.

To further develop on the foundation of construction a practical research into the construction of data files required by the API should be undertaken to personalize the process of data creation and storage of information with the API web structure.

A final development of the interaction between the creation of a variable data set and a custom visual aid to prove an understanding of the functions in usage throughout this research of data manipulation.

# **What Is Plotly:**

Plotly is described as an “interactive , browser based charting library” working as an open source graphing library using frameworks.(Bauer and Traunmüller, 2016) Plotly is not only set as set as an API for python but it has libraries for multiple different languages like Matlab, Javascript and ‘R’.(Mesquita, 2018) The structure of Plotly has three main components to its structure the ‘Data’, the ‘Layout’ and the ‘Figure’. The data part defines what will be displayed within the charts structure and is represented by the specifications set on the data known as a trace by Plotly. The layout is all the other components not related to the data, such as the titles of the axis or the type of chart that is going to be constructed. The figure it the constructed chart object that is created at the end.(Mesquita, 2018)

To understand how to use Plotly the best way to breakdown how to use it is to go through the base tutorial. For a better breakdown Jupyter notebook is the python application of choice to use. The first step described is usage of pip installations to add the packages Plotly, NumPy, SciPy and Pandas for one to interact with their functions.(plotly, 2019) Before going through that the user must create an account with plotly to gain a login username an API access key to make use of the service.

The first thing to learn is the usage of panda to import a dataset file to make use of for the functions of plotly. Panda.read\_csv(variable) command takes a url or system location and reads the data within as a csv format. By then piping the data into a variable for the first example an iplot, an excel like format will be displayed in the output window of jupyter, as well as piped to the users plotly storage container for files.

Panda can then breakdown the structure of the data set into columns based on the titles given so they can be piped into the data access of plotly x and y axis. When importing the capability to create charts plotly must call to the graph\_objs to access the current library of chart type available. This allows for defining the data as a Bar, Scatter, or Pie chart and when a variable is set as one of these inside the function called one can set constraints on how the data is displayed, colorize the data and define which heading are pulled to which axis.

After data is defined the Layout can be called to apply the formatting touches to the description of the chart being created, and contextually have its colour defined.

The advanced capabilities of this API is full animated plotted charts changing visually based on the users input, which in turn can change the values in the data being accessed by the plotly chart.

# **Creating a DataSet File:**

When it comes to working with data the conventional way is to ask for a user to enter the values one by one into separate variables then must reference them each time they are needed to be used, this results in excess of coding and repetition of issuing commands. The creation of a single file to be referenced in future allows for the data to be categorised more easily and accessed by many other applications.

The file structure with the best layout for data manipulation would be a comma separated value (csv) file. This is due to its ability to be a storage set for multiple lists or dictionaries and allows for an API like pandas to predetermine the headings of each group of values.

Before one can use Panda, it would be best to go through the steps of setting up a file for use. By importing csv, we are declaring that we want to use the system functions associated with csv file to initiate our creation of the csv file. For our example we will be taking in the values of ‘EmployeeName’ and ‘Income’ to a list and then nested with a group of lists to create the rows and columns of the data set. By asking the user to input the number of rows they require for the dataset we can iterate through asking for the input for the values for the heading and use the append function to add the values (the sublist) to the main list to be used for generating our file.

Once we are satisfied that our nested list is the appropriate length and has the number of values, we can use the csv library to first create a csv file with the ‘with open’ command set to write mode and pipe it into a variable. We then initialise the writer function of the csv file to interact with our created csv file by using the writerows function to iterate through our nested list to pipe into the blank csv file we created. Finally, we close the file we created to save the contents.

To test to see if this works by using a the plotly tutorial and change what file its referencing to our csv file to generate a gird table to see if the heading match to the data. If done correctly the visuals should be something like this:

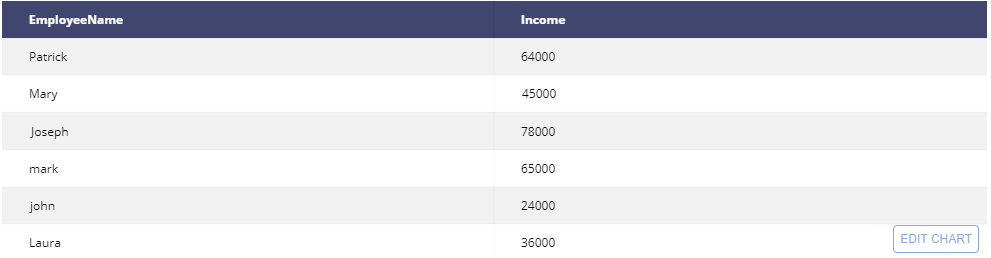


Figure csv file output

# **Manipulating Data into Visual Aid:**

By building the foundation of using Plotly as well as building a foundation on creating the data files it will need to access, one can move on to first create a variating creation setup for the csv file and then create a customized Plotly graph that shows the users income plus also the average income line of the users entered.

To begin with, the file will need to ask the user to enter the values for the heading of our file each of them separated by a comma to allow us to calculate the number of columns we have. By using the split function on our headings, we can divide the heading up and input them into a list which can then be added to dataset list. Then by asking the number of rows to be created we can use a combination of for and while loop to iterate through first by the number of column values required ,we can use a sub-value list that resets after each row is added, the while loop will keep asking for column value inputs until column value is equal to the length of the headings list. The range of rows will continue looping until we have reached the appropriately designate number of rows required by the user. After this the Dataset list is piped into an employee details csv file for to be used by our Plotly visual graphic.

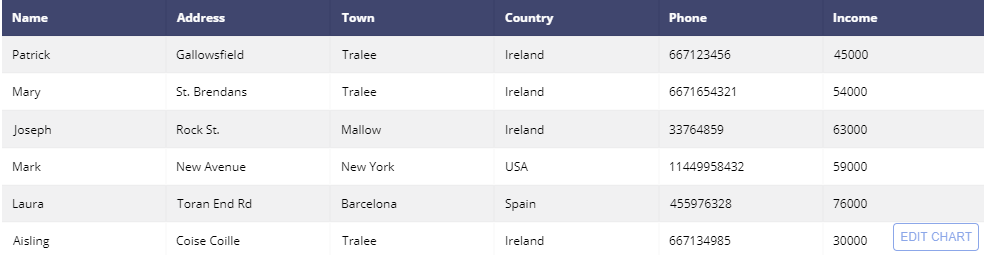


Figure Detailed dataset list of employees

For our Plotly code we will import the Plotly graphics library and the Panda and NumPy library for our usage. Using Panda to read the file we created our dataset will be loaded into the background. For creating our average line, it will be needed to define an average value list as we need to put the average value on our graph at each point for it to display as a straight line. We can use Pandas index location (iloc)function (Lynn, 2018) to select the income column of our dataset and define it as a list and place it into our income values. Using a simple while loops we can then use the NumPy average function to take our list and calculate the average of the values as an integer.(SciPy.org, 2018) To make sure that our graphic can use it correctly we duplicate this value into a list equal in length to the rows created in our data set.

We then define our data values we will be using in each value display starting with the total incomes of our employees. We want our x axis to access values from the Name and the Address of the employee, the y axis to access the values from our Income, We use the marker definition to define a colour coded side bar indicating the level of quality a person income has from red to green, the scale needs to be set from 0 to 1 using floating points to develop a gradient of the coloration of the vertical bars representing the income of the employees as well as a hover bar for the overall range of values of income. The type setting determines the type of chart to be used while the sources point to the user account created on plotly to access data stored on the cloud when we create the graphic.

The average data set is used to define the line along the secondary x-axis pulling y-axis values we set as the average income of the employees and the graphic draws it by defining the chart type as a scatter chart. Once this is defined, we pipe both tchart traces into the data variable to be referenced when we compile the code into the figure.

The layout is then setup for the chart using definitions that are similar to using html formatting to define what will be visible on the chart. We set the chart to autoscale based on the monitor being used to display the data, we define the spaces between the bars on the chart we enable a hovermode to display the values of the bar when the mouse is set over the chart plus define its font size.

We also define in the layout the headings of the overall graphic and the x-axis and y-axis define font sizes, determine if a legend is needed need to set to false to remove it, set the color of the plot background, axis line color and thickness and enable the grid lines to show base increments of a certain value.

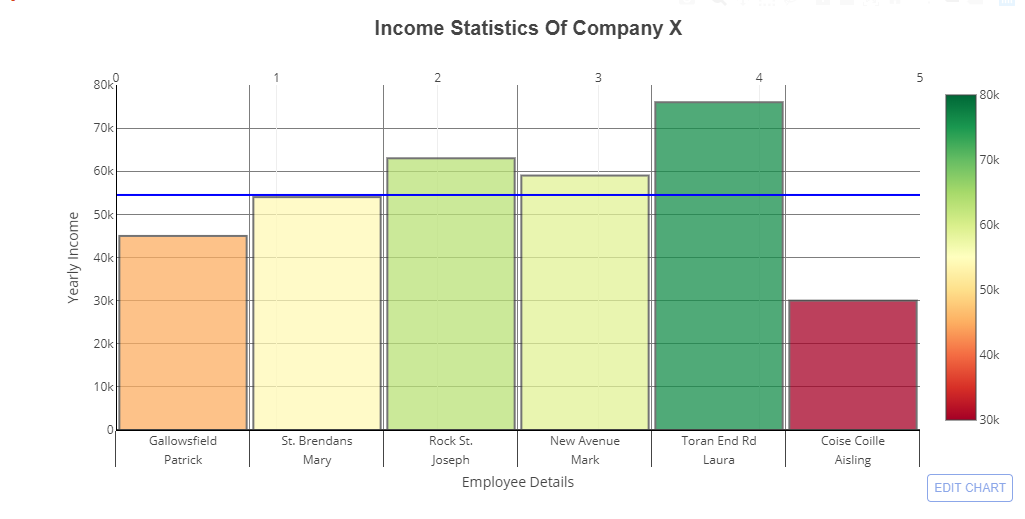
Once both data and layout are defined we can pipe them into the plotly figure function that can then be uploaded to the plotly user files account, giving the file a name and determining if it is public or private.

Figure Custom graphic display of data

# **Conclusion:**

As a derivative of looking into Plotly as a functional API one can discover that it is a handy library if the user requires the manipulation and visualization of the data being presented, in a more understandable format rather than displaying data in a tabled format. The usefulness of such a library is the transference of knowledge from the development of website visual formatting as the context of colouring and sizing uses similar terminology to the syntax used by Plotly for defining the structure of its chart development.

Using libraries to create and manipulate files and data provides a foundation for expanding on simplifying code structures to minimize cluttering of code with redundant or wasteful coding structure, by the creation of external files containing variables and data rather than typing the values manually the libraries in use can pull and change data as needed by the user.

# **Future work:**

As a baseline of the foundation created in learning to manipulate data and values the expanded knowledge of this topic would be to introduce and access data from a database rather than a local file create manually, this would showcase a development of interacting with corporate information structures and have a means of developing complex code or database jargon into a format that can be presented to business personal who have no knowledge of coding or computer related languages developing an ease of communication.

# **Links:**

Plotly main website: <https://plot.ly/#/>

Pandas main website: <https://pandas.pydata.org/>

Python documentation: <https://www.programiz.com/python-programming/>

Github Repository: <https://github.com/Breadus/PlotlyTesting.git>

**References:**

Bauer, P. C. and Traunmüller, R. (2016) *Interactive Data Visualization (2nd Day)*, *bookdown.org*. Available at: https://bookdown.org/paulcbauer/idv2/ (Accessed: 2 May 2019).

Lynn, S. (2018) *iloc, loc, and ix for data selection in Python Pandas | Shane Lynn*, *shanelynn.ie*. Available at: https://www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/ (Accessed: 2 May 2019).

Mesquita, D. (2018) *How and why I used Plotly (instead of D3) to visualize my Lollapalooza data*, *freecodecamp*. Available at: https://medium.freecodecamp.org/how-and-why-i-used-plotly-instead-of-d3-to-visualize-my-lollapalooza-data-d48345e2ca68 (Accessed: 2 May 2019).

plotly (2019) *Jupyter Notebook Tutorial | plotly*, *Plotly*. Available at: https://plot.ly/python/ipython-notebook-tutorial/#import-data (Accessed: 2 May 2019).

SciPy.org (2018) *numpy.average — NumPy v1.14 Manual*, *SciPy.org*. Available at: https://docs.scipy.org/doc/numpy-1.14.5/reference/generated/numpy.average.html (Accessed: 2 May 2019).