**Week 1 - Introduction to Data Visualization Tools**

* Introduction to Data Visualization
  + Why?
    - For exploratory data analysis
    - Communicate data clearly
    - Share unbiased representation of data
    - Support recommendations to different stakeholders.
  + Best Practices
    - Darkhorse Analytics
      * Less is more effective
      * Less is more attractive
      * Less is more impactive
* Introduction to Matplotlib
  + History
    - Developed by John hunter
    - EEG/ECoG Visualization Tool
    - Analogous to Matlab scripting interface.
  + Architecture
    - Backend Layer
      * FigureCancas
        + **matplotlib.backend\_bases.FigureCanvas**

Encompasses the area onto which the figure is drawn.

* + - * Renderer
        + **matplotlib.backend\_bases.Renderer**

knows how to draw on the FigureCanvas

* + - * Event
        + **matplotlib.backend\_bases.Event**

Handles user inputs such as keyboard strokes and mouse clicks.

* + - Artist Layer
      * Artist
        + Knows how to use the renderer to draw on the canvas.
        + Title, lines, tick labels, and images all correspond to individual artist instances.
        + Two type of artist object

Primitive

Line2D, Rectangle, Circle, and Text.

Composite

Axis, Tick, Axes, and Figure

* + - * Text

        Description automatically generatedUse
    - Scripting Layer (More for scientist not programmers.)
      * Pyplot
        + A scripting interface that is lighter that the Artist layer.
      * Graphical user interface, text

        Description automatically generated with medium confidenceGenerating the same graph as the code above.
* Basic Plotting with Matplotlib
  + Using Jupyter Notebook
    - **import matplotlib.pyplot as plt**
    - Use **%matplotlib inline** to ensure the plot stays in the same window. Note doing this makes it impossible to modify the plot further.
    - Instead use **%matplotlib notebook** which still allows the plot to be modified and manipulated.
    - Using plot function
      * Example:
        + **Plt.plot(5,5,’0’)**
        + **Plt.show()**
    - Pandas has built in Matplotlib
    - **https://pandas.pydata.org/?utm\_medium=Exinfluencer&utm\_source=Exinfluencer&utm\_content=000026UJ&utm\_term=10006555&utm\_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork20297740-2021-01-01**
      * **Name\_df.plot(kind = ”line”)**
      * **Name\_df[“Column Name”].plot(kind = “hist”)**
* Dataset on Immigration to Canada
  + We will import everything as a pandas dataframe.
    - Code:
      * **Import numpy as np** \*For computation
      * **Import pandas as pd** \*for Data Structure library
      * **From \_\_future\_\_ import print\_function** \*Compatibility to python 2
      * **!pip install xlrd**
      * **Print(‘xlrd installed!’)**
      * **df\_can = pd.read.excel(‘filepath’, sheetname = “Sheet name”, skiprows = range(number of rows to skip), skip\_footer = 2)**
* Line Plots
  + Chart, line chart

    Description automatically generatedA line plot is a type of plot which displays information as a series of data points called ‘marker’ connected by straight line segments.
  + Code
    - **Import matplotlib as mpl**
    - **Import matplotlib.pyplot as plt**
    - **Year = list(map(str, range(1980, 2014)))**
    - **Df\_canada.loc[‘Haiti’, years].plot(kind = ‘line’)**
    - **Plt.title(‘Immigration from Haiti’)**
    - **Plt.ylabel(‘Number of immigrats’)**
    - **Plt.xlabel(‘Years’)**
    - **Plt.show()**
* Lab: Introduction to Matplotlib and Line Plots
  + Useful function
    - **.head()**
    - **.tail()**
    - **.info**
    - **.columns**
    - **.index**
    - **.tolist()**
      * To get the index and columns as lists, we can use the tolist() method. Since the data type of the columns and rows are not lists.
    - **.shape**
      * Returns size of the dataframe.
    - **.drop** (to drop columns or data)
      * **# in pandas axis=0 represents rows (default) and axis=1 represents columns.**
      * **df\_can.drop(['AREA','REG','DEV','Type','Coverage'], axis=1, inplace=True)**
    - **.rename**
      * **df\_can.rename(columns={'OdName':'Country', 'AreaName':'Continent', 'RegName':'Region'}, inplace=True)**
    - Adding a column
      * Example
        + We will also add a 'Total' column that sums up the total immigrants by country over the entire period 1980 - 2013, as follows:
        + **df\_can['Total'] = df\_can.sum(axis=1)**
    - Filtering
      * **df\_can.Country** 
        + # returns a series for all countries and all years
      * **df\_can[['Country', 1980, 1981, 1982, 1983, 1984, 1985]]**
        + filtering on the list of countries ('Country') and the data for years: 1980 - 1985.
    - Locating Data
      * **df.loc[label**] # filters by the labels of the index/column
      * **df.iloc[index]** # filters by the positions of the index/column
    - Setting a column as the index
      * **df\_can.set\_index('Country', inplace=True)**
    - Remove the name of the index.
      * **df\_can.index.name = None**
    - Getting full row of data
      * **df\_can.loc['Japan'] or df\_can.iloc[87] or df\_can[df\_can.index == 'Japan']**
    - Getting data with a where clause
      * **df\_can.loc['Japan', 2013] or df\_can.iloc[87, 36] or df\_can.loc['Japan', [1980, 1981, 1982, 1983, 1984, 1984]]**
    - Converting data column types
      * **df\_can.columns = list(map(str, df\_can.columns))**
    - Filtering base on criteria
      * # 1. create the condition boolean series
        + **condition = df\_can['Continent'] == 'Asia'**
        + **print(condition)**
      * # 2. pass this condition into the dataFrame
        + **df\_can[condition]**
      * # 3. # we can pass multiple criteria in the same line.# let's filter for AreaNAme = Asia and RegName = Southern Asia
        + **df\_can[(df\_can['Continent']=='Asia') & (df\_can['Region']=='Southern Asia')]**
        + # note: When using 'and' and 'or' operators, pandas requires we use '&' and '|' instead of 'and' and 'or'
        + # don't forget to enclose the two conditions in parentheses
    - Sorting
      * **DataFrame.sort\_values(*by*, *axis=0*, *ascending=True*, *inplace=False*, *kind='quicksort'*, *na\_position='last'*, *ignore\_index=False*, *key=None*)**
    - Plotting
      * Code:
      * Initial Plots
        + **haiti = df\_can.loc['Haiti', years] # passing in years 1980 - 2013 to exclude the 'total' column**
        + **haiti.head()**
        + **haiti.plot()**
      * Adding Title
        + **haiti.index = haiti.index.map(int)** # let's change the index values of Haiti to type integer for plotting
        + **haiti.plot(kind='line')**
        + **plt.title('Immigration from Haiti')**
        + **plt.ylabel('Number of immigrants')**
        + **plt.xlabel('Years')**
        + **plt.show()** # need this line to show the updates made to the figure
      * Adding Text Annotation
        + **plt.text(2000, 6000, '2010 Earthquake')** # see note below
      * **Swapping columns for rows**
        + **Use .transpose()**
    - Lab Assignment
      * Step 1: Get the data set for China and India, and display the dataframe.
        + **df\_CI = df\_can.loc[['China','India'], years]**
      * Step 2: Plot graph. We will explicitly specify line plot by passing in kind parameter to plot().
        + **cn\_in\_plot = df\_CI.plot(kind = 'line')**
        + **cn\_in\_plot.plot()**
      * Step 3: we must first transpose the dataframe using transpose() method to swap the row and columns.
        + **df\_CI = df\_CI.transpose()**
        + **df\_CI.head()**
      * Step 4: pandas will auomatically graph the two countries on the same graph. Go ahead and plot the new transposed dataframe. Make sure to add a title to the plot and label the axes.
        + **cn\_in\_plot = df\_CI.plot(kind = 'line')**
        + **plt.title('Immingration of People from China and India')**
        + **plt.ylabel('Number of People')**
        + **plt.xlabel('Years')**
        + **plt.show()**

\*Note we don’t call .title, and axis name with the plot object.

#Function below simply shorts the data using pandas

**df\_can.sort\_values(by = 'Total', axis = 0 , ascending = False, inplace = True, kind = 'quicksort')**

#Getting top 5 only

**top5\_countries = df\_can.head(5)**

#top5\_countries.plot(kind = 'line')

\*This displays data being used in the wrong axis.

**top5\_countries = top5\_countries[years].transpose()**

\*Transposing years to the X axis.

**top5\_countries.plot(kind = 'line')**

**plt.title('Top 5 Countries by Number of Immigrants')**

**plt.ylabel('Number of People')**

**plt.xlabel('Years')**

**plt.show()**

Quiz: Introduction to Data Visualization Tools