**Week 2 - Basic and Specialized Visualization Tools**

**For all code**

**Import matplotlib as mpl**

**Import matplotlib.pyplot as plt**

* Area Plots
  + - Area chart or area graph
    - Used to present cumulated totals using numbers or percentages over time
    - Based on the line plot.
    - Code
      * **Df\_top5.plot(kind=’area’)**
    - **Lab Example (alpha value is the transparency 0-1)**
      * **df\_top5.index = df\_top5.index.map(int)**
      * **df\_top5.plot(kind='area',**
      * **alpha = 0.25,**
      * **stacked=False,**
      * **figsize=(20, 10)) # pass a tuple (x, y) size**
      * **plt.title('Immigration Trend of Top 5 Countries')**
      * **plt.ylabel('Number of Immigrants')**
      * **plt.xlabel('Years')**
      * **plt.show()**
* Histograms
  + - A histogram is a way of representing the frequency distribution of a variable.
    - Code: (IMPORTANT TO ADD THE XTICKS IN THE .PLOT FUNCTION)
      * **Df\_canada[‘2013’].plot(kind=’hist’)**
      * To align bar with the graph
        + **Import numpy as np**
        + **Count, bin\_edges = np.historgram(df\_canada[‘2013’])**
        + **Df\_canada[‘2013’].plot(kind=’hist’, xticks = bin\_edges)**
    - **Lab Example**
      * **df\_can['2013'].head()**
      * **count, bin\_edges = np.histogram(df\_can['2013'])**
      * **print(count) # frequency count**
      * **print(bin\_edges) # bin ranges, default = 10 bins**
    - **Full Example**
      * **count, bin\_edges = np.histogram(df\_t, 15)**
      * **xmin = bin\_edges[0] - 10 # first bin value is 31.0, adding buffer of 10 for aesthetic purposes**
      * **xmax = bin\_edges[-1] + 10 # last bin value is 308.0, adding buffer of 10 for aesthetic purposes**
      * **# stacked Histogram**
      * **df\_t.plot(kind='hist',**
      * **figsize=(10, 6),**
      * **bins=15,**
      * **xticks=bin\_edges,**
      * **color=['coral', 'darkslateblue', 'mediumseagreen'],**
      * **stacked=True,**
      * **xlim=(xmin, xmax)**
      * **)**
      * **plt.title('Histogram of Immigration from Denmark, Norway, and Sweden from 1980 - 2013')**
      * **plt.ylabel('Number of Years')**
      * **plt.xlabel('Number of Immigrants')**
      * **plt.show()**
* Bar Charts
  + - Bar chart commonly used to compare the values of a variable at a given point in time.
      * **Df\_iceland.plot(kind=’bar’)**
    - Two types
      * **Kind = bar (Vertical Bar)**
      * **Kind = barh (Horizontal Bar)**
    - **Lab Example**
      * **df\_iceland = df\_can.loc['Iceland', years]**
      * **df\_iceland.head()**
      * **df\_iceland.plot(kind='bar', figsize=(10, 6))**
      * **plt.xlabel('Year')** # add to x-label to the plot
      * **plt.ylabel('Number of immigrants')** # add y-label to the plot
      * **plt.title('Icelandic immigrants to Canada from 1980 to 2013')** # add title to the plot
      * **plt.annotate('',** # s: str. Will leave it blank for no text

**xy=(32, 70),** # place head of the arrow at point **(year 2012 , pop 70)**

**xytext=(28, 20),** # place base of the arrow at **point (year 2008 , pop 20)**

**xycoords='data',** # will use the coordinate **system of the object being annotated**

**arrowprops=dict(arrowstyle='->', connectionstyle='arc3', color='blue', lw=2)**

**)**

* + - * **plt.show()**
    - **To add Text to the annotation**
      * **plt.annotate('2008 - 2011 Financial Crisis',** # text to display

**xy=(28, 30),** # start the text at at point **(year 2008 , pop 30)**

**rotation=72.5,** # based on trial and error to match the arrow

**va='bottom',** # want the text to be vertically 'bottom' aligned

**ha='left',** # want the text to be horizontally 'left' algned.

**)**

* Lab: Basic Visualization Tools
  + - Useful code
      * Change all to string
        + **df\_can.columns = list(map(str, df\_can.columns))**
      * To check if all string
        + **all(isinstance(column, str) for column in df\_can.columns)**
      * Creating a list with a range
        + **years = list(map(str, range(1980, 2014)))**
      * Transposing dataframe index to (in this case years)
        + **df\_top5 = df\_top5[years].transpose()**
      * Setting index as an int **(UNSURE YOU DO THIS FOR PLOTTING)**
        + **df\_least5.index = df\_least5.index.map(int)**
      * For annotating
        + s: str, the text of annotation.
        + xy: Tuple specifying the (x,y) point to annotate (in this case, end point of arrow).
        + xytext: Tuple specifying the (x,y) point to place the text (in this case, start point of arrow).
        + xycoords: The coordinate system that xy is given in - 'data' uses the coordinate system of the object being annotated (default).
        + arrowprops: Takes a dictionary of properties to draw the arrow:

arrowstyle: Specifies the arrow style, '->' is standard arrow.

connectionstyle: Specifies the connection type. arc3 is a straight line.

color: Specifies color of arrow.

lw: Specifies the line width.

We will use *pandas* groupby method to summarize the immigration data by Continent. The general process of groupby involves the following steps:

* **Split:** Splitting the data into groups based on some criteria.
* **Apply:** Applying a function to each group independently: .sum() .count() .mean() .std() .aggregate() .apply() .etc..
* **Combine:** Combining the results into a data structure.
  + - Two methods of plotting
      * **Procedural Method (Scripting Layer)**
        + **df\_last5 = df\_can.tail(5)**
        + **df\_last5 = df\_last5[years].transpose()**
        + **df\_last5.index = df\_last5.index.map(int)**
        + **df\_last5.plot(kind='area', stacked = True, alpha = 0.45, figsize = (20,10))**
        + **plt.title('The 5 Countries that Contributed least to Immingration')**
        + **plt.ylabel('Number of Immigrants')**
        + **plt.xlabel('Years')**

**Text

Description automatically generated with medium confidence**

* + - * **Object Oriented Method (Artist Layer)**
        + **df\_last5 = df\_can.tail(5)**
        + **df\_last5 = df\_last5[years].transpose()**
        + **df\_last5.index = df\_last5.index.map(int)**
        + **ax = df\_top5.plot(kind = 'area', alpha = 0.55, figsize = (20,10))**
        + **ax.set\_title('The 5 Countries that Contributed least to Immingration')**
        + **ax.set\_ylabel('Number of Immigrants')**
        + **ax.set\_xlabel('Years')**

**Text

Description automatically generated with medium confidence**

**Lab Example Answer**

**Question**: Use the scripting layer to display the immigration distribution for Greece, Albania, and Bulgaria for years 1980 - 2013? Use an overlapping plot with 15 bins and a transparency value of 0.35.

**df\_GAB = df\_can.loc[['Greece', 'Albania', 'Bulgaria'], years]**

**df\_GAB = df\_GAB.transpose()**

**count, bin\_edges = np.histogram(df\_GAB, 15)**

**df\_GAB.plot(kind = 'hist', bins = 15, xticks = bin\_edges, alpha = 0.35, figsize = (20,10), color=['coral', 'darkslateblue', 'mediumseagreen'])**

**plt.title('Distribution for Greece, Albania, and Bulgaria for years 1980 - 2013')**

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

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

* Pie Charts
  + - How to create a pie chart
      * First group data
        + **Df\_continent = df\_canada.groupby(‘Continent’, axis = 0).sum()**
        + **Text

          Description automatically generatedDf\_continents[‘Total’].plot(kind = ‘pie’)**
      * **Graphical user interface, application

        Description automatically generatedProfessional Looking Pie Charts**
* Box Plots
  + - Displays information in 5 dimensions
      * Outliers
      * Maximum
      * Third Quartile
      * Median
      * First Quartile
      * Minimum
    - Code
      * Diagram

        Description automatically generated with low confidence
* Scatter Plots
  + - Table

      Description automatically generatedChart, scatter chart

      Description automatically generatedDisplays values comparing two variables

Graphical user interface, text, application, email

Description automatically generated

\*Code for Horizontal Box Plots

**df\_CI.plot(kind='box', figsize=(10, 7), color='blue', vert=False)**

**plt.title('Box plots of Immigrants from China and India (1980 - 2013)')**

**plt.xlabel('Number of Immigrants')**

**plt.show()**

* Subplots
  + To visualize multiple plots together, we can create a **figure** (overall canvas) and divide it into **subplots**, each containing a plot. With **subplots**, we usually work with the **artist layer** instead of the **scripting layer**.
  + Syntax for creating a Figure
    - fig **=** plt.figure() *# create figure*
    - ax **=** fig.add\_subplot(nrows, ncols, plot\_number) *# create subplots*
  + Where
    - nrows and ncols are used to notionally split the figure into (nrows \* ncols) sub-axes,
    - plot\_number is used to identify the particular subplot that this function is to create within the notional grid. plot\_number starts at 1, increments across rows first and has a maximum of nrows \* ncols as shown below.

Diagram

Description automatically generated

* + Chart, histogram, box and whisker chart

    Description automatically generatedGraphical user interface, text, application

    Description automatically generatedCode for Subplots

Question 1

**Question:** Create a box plot to visualize the distribution of the top 15 countries (based on total immigration) grouped by the decades 1980s, 1990s, and 2000s.

Step 1: Get the dataset. Get the top 15 countries based on Total immigrant population. Name the dataframe **df\_top15**.

**df\_top15 = df\_can.sort\_values(['Total'], ascending =False, axis = 0).head(15).transpose()**

**df\_top15**

Step 2: Create a new dataframe which contains the aggregate for each decade. One way to do that:

1. Create a list of all years in decades 80's, 90's, and 00's.
2. Slice the original dataframe df\_can to create a series for each decade and sum across all years for each country.
3. Merge the three series into a new data frame. Call your dataframe **new\_df**.

**year80s = list(map(str, range(1980, 1990)))**

**year90s = list(map(str, range(1990, 2000)))**

**year20s = list(map(str, range(2000, 2010)))**

**df\_80s = df\_top15.loc[year80s].sum(axis=1)**

**df\_90s = df\_top15.loc[year90s].sum(axis=1)**

**df\_20s = df\_top15.loc[year20s].sum(axis=1)**

**df\_combined = pd.DataFrame({'1980s' : df\_80s, '1990s' : df\_90s, '2000s': df\_20s})**

**df\_combined**

Step 3: let's check how many entries fall above the outlier threshold

**df\_combined.describe()**

**df\_combined=df\_combined.reset\_index()**

**df\_combined[df\_combined['2000s']> 209611.5]**

* Scatter Plots
  + - A scatter plot (2D) is a useful method of comparing variables against each other.
  + Step 1: Get the dataset. Since we are expecting to use the relationship between years and total population, we will convert years to int type.

# we can use the sum() method to get the total population per year

**df\_tot = pd.DataFrame(df\_can[years].sum(axis=0))**

# change the years to type int (useful for regression later on)

**df\_tot.index = map(int, df\_tot.index)**

# reset the index to put in back in as a column in the df\_tot dataframe

**df\_tot.reset\_index(inplace = True)**

# rename columns

**df\_tot.columns = ['year', 'total']**

# view the final dataframe

**df\_tot.head()**

* + Step 2: Plot the data. In Matplotlib, we can create a scatter plot set by passing in kind='scatter' as plot argument. We will also need to pass in x and y keywords to specify the columns that go on the x- and the y-axis.

**df\_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6), color='darkblue')**

**plt.title('Total Immigration to Canada from 1980 - 2013')**

**plt.xlabel('Year')**

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

**Chart, scatter chart

Description automatically generatedplt.show()**

* Step 1: Get the equation of line of best fit. We will use **Numpy**'s polyfit() method by passing in the following:
  + x: x-coordinates of the data.
  + y: y-coordinates of the data.
  + deg: Degree of fitting polynomial. 1 = linear, 2 = quadratic, and so on.
    - **x = df\_tot['year'] # year on x-axis**
    - **y = df\_tot['total'] # total on y-axis**
    - **fit = np.polyfit(x, y, deg=1)**
    - **fit**
    - array([ 5.56709228e+03, -1.09261952e+07])
* Step 2: Plot the regression line on the scatter plot.

**df\_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6), color='darkblue')**

**plt.title('Total Immigration to Canada from 1980 - 2013')**

**plt.xlabel('Year')**

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

# plot line of best fit

**plt.plot(x, fit[0] \* x + fit[1], color='red') # recall that x is the Years**

**plt.annotate('y={0:.0f} x + {1:.0f}'.format(fit[0], fit[1]), xy=(2000, 150000))**

**plt.show()**

# print out the line of best fit

**'No. Immigrants = {0:.0f} \* Year + {1:.0f}'.format(fit[0], fit[1])**

* Bubble Plot
  + A bubble plot is a variation of the scatter plot that displays three dimensions of data (x, y, z). The data points are replaced with bubbles, and the size of the bubble is determined by the third variable z, also known as the weight. In maplotlib, we can pass in an array or scalar to the parameter s to plot(), that contains the weight of each point.
  + Example
  + **Step 1**: Get the data for Brazil and Argentina. Like in the previous example, we will convert the Years to type int and include it in the dataframe.

# transposed dataframe

**df\_can\_t = df\_can[years].transpose()**

# cast the Years (the index) to type int

**df\_can\_t.index = map(int, df\_can\_t.index)**

# let's label the index. This will automatically be the column name when we reset the index

**df\_can\_t.index.name = 'Year'**

# reset index to bring the Year in as a column

**df\_can\_t.reset\_index(inplace=True)**

# view the changes

**df\_can\_t.head()**

* + **Step 2**: Create the normalized weights.
    - **A picture containing text

      Description automatically generated**There are several methods of normalizations in statistics, each with its own use. In this case, we will use [feature scaling](https://en.wikipedia.org/wiki/Feature_scaling?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork20297740-2021-01-01) to bring all values into the range [0, 1]. The general formula is:
    - where 𝑋 is the original value, 𝑋′ is the corresponding normalized value. The formula sets the max value in the dataset to 1, and sets the min value to 0. The rest of the data points are scaled to a value between 0-1 accordingly.

# normalize Brazil data

**norm\_brazil = (df\_can\_t['Brazil'] - df\_can\_t['Brazil'].min()) / (df\_can\_t['Brazil'].max() - df\_can\_t['Brazil'].min())**

# normalize Argentina data

**norm\_argentina = (df\_can\_t['Argentina'] - df\_can\_t['Argentina'].min()) / (df\_can\_t['Argentina'].max() - df\_can\_t['Argentina'].min())**

* + **Step 3**: Plot the data.
* To plot two different scatter plots in one plot, we can include the axes one plot into the other by passing it via the ax parameter.
* We will also pass in the weights using the s parameter. Given that the normalized weights are between 0-1, they won't be visible on the plot. Therefore, we will:
  + multiply weights by 2000 to scale it up on the graph, and,
  + add 10 to compensate for the min value (which has a 0 weight and therefore scale with ×2000×2000).

# Brazil

**ax0 = df\_can\_t.plot(kind='scatter',**

**x='Year',**

**y='Brazil',**

**figsize=(14, 8),**

**alpha=0.5, # transparency**

**color='green',**

**s=norm\_brazil \* 2000 + 10, # pass in weights**

**xlim=(1975, 2015)**

**)**

# Argentina

**ax1 = df\_can\_t.plot(kind='scatter',**

**x='Year',**

**y='Argentina',**

**alpha=0.5,**

**color="blue",**

**s=norm\_argentina \* 2000 + 10,**

**ax=ax0**

**)**

**ax0.set\_ylabel('Number of Immigrants')**

**ax0.set\_title('Immigration from Brazil and Argentina from 1980 to 2013')**

**ax0.legend(['Brazil', 'Argentina'], loc='upper left', fontsize='x-large')**

Another Example

**Question**: Previously in this lab, we created box plots to compare immigration from China and India to Canada. Create bubble plots of immigration from China and India to visualize any differences with time from 1980 to 2013. You can use **df\_can\_t** that we defined and used in the previous example.

Step 1: Normalize the data pertaining to China and India.

**# normalized Chinese data**

**norm\_china = (df\_can\_t['China'] - df\_can\_t['China'].min()) / (df\_can\_t['China'].max() - df\_can\_t['China'].min())**

**# normalized Indian data**

**norm\_india = (df\_can\_t['India'] - df\_can\_t['India'].min()) / (df\_can\_t['India'].max() - df\_can\_t['India'].min())**

**#norm\_china, norm\_india**

Step 2: Generate the bubble plots.

**ax0 = df\_can\_t.plot(kind = 'scatter',**

**x = 'Year',**

**y= 'China' ,**

**figsize = (14,8) ,**

**alpha = 0.5,**

**color = 'red',**

**s = norm\_china \* 2000 + 10,**

**xlim=(1975, 2015))**

**ax1 = df\_can\_t.plot(kind = 'scatter',**

**x = 'Year',**

**y= 'India' ,**

**figsize = (14,8) ,**

**alpha = 0.5,**

**color = 'blue',**

**s = norm\_india \* 2000 + 10,**

**ax=ax0 #Note use this to combine the plots**

**)**

**ax1.set\_title('Immigration from China and India to Canada')**

**ax1.set\_ylabel('Number of Immigrants')**

**ax1.legend(['China', 'India'], loc='upper left', fontsize='x-large')**

**Chart, bubble chart

Description automatically generated**

* Lab: Specialized Visualization Tools
* Quiz: Basic Visualization Tools
* Quiz: Specialized Visualization Tools