basic_plot_2

February 3, 2023

```
[ ]: # import libraries
     import matplotlib as mpl
     import matplotlib.pyplot as plt
     import numpy as np
     import pandas as pd
```

```
[ ]: # read data file
     df_can = pd.read_excel ('Canada.xlsx', sheet_name = 'Canada by Citizenship',␣
      ↪skiprows = 20, skipfooter = 2)
```

```
[ ]: df_can.head()
```

```
[ ]: df_can.tail()
```

```
[ ]: print (df_can.shape)
```

```
[ ]: # clean up data
     df_can.drop(['AREA', 'REG', 'DEV', 'Type', 'Coverage'], axis = 1, inplace =␣
      ↪True)
```

```
[ ]: df_can.head()
```

```
[ ]: # rename some of the columns
     df_can.rename(columns={'OdName':'Country', 'AreaName':'Continent', 'RegName':
      ↪'Region'}, inplace = True)
```

```
[ ]: df_can.head()
```

```
[ ]: # change column labels to strings
     df_can.columns = list(map(str, df_can.columns))
```

```
[ ]: df_can.set_index('Country', inplace = True)
```

```
[ ]: df_can.head()
```

```
[ ]: # add a total column
     df_can['Total'] = df_can.sum(numeric_only = True, axis=1)
```

```
df_can.head()
```

[ ]:
```
# check dimension
df_can.shape
```

[ ]:
```
# create a list of years
years = list (map (str, range (1980, 2014)))
print (years)
```

Stacked Line Plot or Area Plot

[ ]:
```
df_can.sort_values(['Total'], ascending = False, axis = 0, inplace = True)
```

[ ]:
```
# get top 5 entries
df_top5 = df_can.head(5)
print (df_top5)
```

[ ]:
```
# transpose the data frame
df_top5 = df_top5[years].transpose()
df_top5.head()
```

[ ]:
```
# change the index values of df_top5 to integer for plotting
df_top5.index = df_top5.index.map(int)
df_top5.plot (kind = 'area', stacked = False, figsize = (20,10))
plt.title ('Immigration Trend of Top 5 Countries')
plt.xlabel ('Years')
plt.ylabel ('Number of Immigrants')
plt.show()
```

[ ]:
```
# Plot of stacked data
df_top5.plot (kind = 'area', stacked = True, figsize = (20,10))
plt.title ('Immigration Trend of Top 5 Countries')
plt.xlabel ('Years')
plt.ylabel ('Number of Immigrants')
plt.show()
```

[ ]:
```
# plotting with the Artist layer
ax = df_top5.plot (kind = 'area', alpha = 0.15, figsize = (20, 10))
ax.set_title ('Immigration Trend of Top 5 Countries')
ax.set_xlabel ('Years')
ax.set_ylabel ('Number of Immigrants')
```

[ ]:
```
# 5 Countries that contributed the least to the immigration to Canada
df_can.sort_values(['Total'], ascending = True, axis = 0, inplace = True)
df_bot5 = df_can.head(5)
df_bot5
```

[ ]:
```
# transpose the dataframe
df_bot5 = df_bot5[years].transpose()
df_bot5
```

[ ]:
```
# change the index values of df_bot5 to integer for plotting
df_bot5.index = df_bot5.index.map(int)
df_bot5.plot (kind = 'area', stacked = False, figsize = (20,10))
plt.title ('Immigration Trend of Bottom 5 Countries')
plt.xlabel ('Years')
plt.ylabel ('Number of Immigrants')
plt.show()
```

[ ]:
```
# stacked area plot
df_bot5.plot (kind = 'area', stacked = True, figsize = (20,10))
plt.title ('Immigration Trend of Bottom 5 Countries')
plt.xlabel ('Years')
plt.ylabel ('Number of Immigrants')
plt.show()
```

[ ]:
```
# plotting with the Artist layer
ax = df_bot5.plot (kind = 'area', alpha = 0.35, figsize = (20, 10))
ax.set_title ('Immigration Trend of Bottom 5 Countries')
ax.set_xlabel ('Years')
ax.set_ylabel ('Number of Immigrants')
```

Frequency distribution of the number (population) of new immigrants from the various countries to Canada in 2013

[ ]:
```
# get the 2013 data
df_can['2013'].head()
```

[ ]:
```
# np.histogram returns 2 values
count, bin_edges = np.histogram(df_can['2013'])
# frequency count
print (count)
# bin ranges, default = 10 bins
print (bin_edges)
```

[ ]:
```
# plot the histogram
df_can['2013'].plot(kind = 'hist', figsize = (8,5))

# add title to histogram
plt.title ('Histogram of Immigration from 195 Countries in 2013')

# add x label
plt.xlabel ('Number of Immigrants')
```

```
# add y label
plt.ylabel ('Number of Countries')

plt.show()
```

[ ]:
```
# get list of bin intervals as 'bin_edges'
count, bin_edges = np.histogram (df_can['2013'])

# plot the histogram
df_can['2013'].plot.hist (figsize = (8,5), xticks = bin_edges)

# add title to histogram
plt.title ('Histogram of Immigration from 195 Countries in 2013')

# add x label
plt.xlabel ('Number of Immigrants')

# add y label
plt.ylabel ('Number of Countries')

plt.show()
```

Immigration Distribution for Denmark, Norway, and Sweden for 1980 - 2013

[ ]:
```
# get dataset
df_nord = df_can.loc[['Denmark', 'Norway', 'Sweden'], years]
df_nord = df_nord.transpose()

# generate histogram
df_nord.plot (kind = 'hist', figsize = (10, 6))

# add title to histogram
plt.title ('Histogram of Immigration from Denmark, Norway, Sweden, from 1980 to␣
 ↪2013')

# add x label
plt.xlabel ('Number of Immigrants')

# add y label
plt.ylabel ('Number of Years')

plt.show()
```

[ ]:
```
# increase bin size to 15
count, bin_edges = np.histogram(df_nord, 15)

# un-stacked histogram
```

```
df_nord.plot (kind = 'hist', figsize = (10, 6), bins = 15,
              alpha = 0.6, xticks = bin_edges,
              color = ['coral', 'darkslateblue', 'mediumseagreen'])

# add title to histogram
plt.title ('Histogram of Immigration from Denmark, Norway, Sweden, from 1980 to␣
↪2013')

# add x label
plt.xlabel ('Number of Immigrants')

# add y label
plt.ylabel ('Number of Years')

plt.show()
```

Full Listing of Colors in MatPlotLib

```
[ ]: '''
     for name, hex in mpl.colors.cnames.items():
         print (name, hex)
     '''
```

Stacked Histogram - No Overlap

```
[ ]: count, bin_edges = np.histogram (df_nord, 15)

     # first bin value is 31.0, add a buffer of 10
     xmin = bin_edges[0] - 10

     # last bin value is 308, add a buffer of 10
     xmax = bin_edges[-1] + 10

     # define ymin
     ymin = 0

     # define ymax
     ymax = 22

     # stacked histogram
     df_nord.plot (kind = 'hist', figsize = (10, 6), bins = 15,
                   xticks = bin_edges,
                   color = ['coral', 'darkslateblue', 'mediumseagreen'],
                   stacked = True, xlim = (xmin, xmax),
                   ylim = (ymin, ymax))

     # add title to histogram
```

```
plt.title ('Histogram of Immigration from Denmark, Norway, Sweden, from 1980 to␣
 ↪2013')

# add x label
plt.xlabel ('Number of Immigrants')

# add y label
plt.ylabel ('Number of Years')

plt.show()
```

Analyzing Iceland's Financial Crisis using Bar Graphs

[ ]:
```
# get immigration numbers for Iceland
df_ice = df_can.loc ['Iceland', years]
df_ice.head()
```

[ ]:
```
# plot bar graph
df_ice.plot (kind = 'bar', figsize = (10, 6))

plt.title ('Immigration from Iceland')
plt.xlabel ('Year')
plt.ylabel ('Number of Immigrants')

plt.show()
```

Annotate the Bar Graph

- s: str the text for annotation
- xy: tuple specifying the (x,y) point to annotate - end of the arrow
- xytext: tuple specifying the (x,y) point to place the text - start of the arrow
- xycoords: coordinate system that xy is given in
- arrowprops: takes a dictionary of properties to draw the arrow
  - arrowstyle: specifies the arrow style
  - connectionstyle: specifies the connection type
  - color: specifies color of arrow
  - lw: specifies the line width
- annotate text
  - rotation: rotation angle of text in degrees counterclockwise
  - va: vertical alignment of text [‘center’|‘top’|‘bottom’|‘baseline’]
  - ha: horizontal alignment of text [‘center’|‘right’|‘left’]

[ ]:
```
# plot bar graph
df_ice.plot (kind = 'bar', figsize = (10, 6), rot = 90)

plt.title ('Immigration from Iceland')
plt.xlabel ('Year')
plt.ylabel ('Number of Immigrants')

# annotate arrow
plt.annotate ('', # s is blank for now
              xy = (32, 70), # place head of arrow (year 2012, pop 70)
              xytext = (28, 20), # place base of arrow (year 2008, pop 20)
              xycoords = 'data', # use coordinate system of the object
              arrowprops = dict(arrowstyle = '->', connectionstyle = 'arc3',␣
 ↪color = 'red', lw = 2)
             )

# annotate text
plt.annotate ('2008 - 2011 Financial Crisis', # text to display
              xy = (28, 30), # start the text at (year 2008, pop 30)
              rotation = 72.5, # based on trial and error
              va = 'bottom', # vertically bottom aligned
              ha = 'left', # horizontally left aligned
             )

plt.show()
```

Horizontal Bar Plot

```
[ ]: # get the data
     df_can.sort_values(['Total'], ascending = False, axis = 0, inplace = True)

     # get the top 15 countries
     df_top15 = df_can.head(15)

     # display the top 15 countries
     df_top15
```

```
[ ]: # get the countries and the total immigrants
     df_tot15 = df_top15['Total']

     # show the total immigrants
     df_tot15

     # plot the data
     df_tot15.plot (kind = 'barh', figsize = (10,6), color = 'yellow')

     plt.title ('Total Immigration Numbers for Top 15 Countries')
     plt.xlabel ('Number of Immigrants')
     plt.ylabel ('Country')

     # annotate value labels to each country
     for index, value in enumerate (df_tot15):
```

```
    label = format (int(value), ',')

    # place text at the end of bar
    plt.annotate (label, xy=(value - 47000, index - 0.10), color = 'black')

plt.show()
```