basic_plot_3

February 1, 2023

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

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

[ ]:
```
# clean up the dataset to remove unnecessary columns (eg. REG)
df_can.drop(['AREA', 'REG', 'DEV', 'Type', 'Coverage'], axis=1, inplace = True)

# let's rename the columns so that they make sense
df_can.rename(columns={'OdName':'Country', 'AreaName':'Continent','RegName':
 ↪'Region'}, inplace=True)

# for sake of consistency, let's also make all column labels of type string
df_can.columns = list(map(str, df_can.columns))

# set the country name as index - useful for quickly looking up countries using␣
 ↪.loc method
df_can.set_index('Country', inplace = True)

# years that we will be using in this lesson - useful for plotting later on
years = list(map(str, range(1980, 2014)))

# add a new column with the total immigration
df_tot = df_can[years].sum(axis=1)
df_can['Total'] = df_tot

print('data dimensions:', df_can.shape)
```

[ ]:
```
print (df_can['Total'])
```

Generate Pie Plot to show the immigrants by continents We will use the *pandas* ***groupby*** to summarize the immigration numbers by *Continent*

The general process of using *groupby* involves * Split: Splitting the data into groups based on some criteria * Apply: Applying a function to each group independently like - sum(), count(), mean(), std() * Combine: Combining the results into a data structure

[ ]:
```
# group countries by Continents and apply the sum() function
df_continents = df_can.groupby ('Continent', axis = 0).sum()
df_continents.head()
```

[ ]:
```
# draw the pie chart
df_continents['Total'].plot(kind = 'pie', figsize = (5,6), startangle = 90,␣
 ↪labels = None)

plt.title ('Immigration by Continents')
plt.axis ('equal')
plt.legend (labels = df_continents.index, loc = 'upper right')
plt.show()
```

Box Plots

- Q1 = First Quartile 25% below this value
- Q2 = Median or 50% below this value
- Q3 = Third Quartile or 75% below this value
- IQR = Inter Quartile Range = Q3 - Q1
- Minimum = Q1 - 1.5 * IQR
- Maximum = Q3 + 1.5 * IQR

Make a box plot of all Japanese immigrants

[ ]:
```
# get the data
years = list(map(str, range(1980, 2014)))
df_japan = df_can.loc[['Japan'], years].transpose()
df_japan.head()
```

[ ]:
```
# create the box plot
df_japan.plot(kind = 'box', figsize = (8, 6))
plt.title ('Box Plot of Japanese Immigrants')
plt.ylabel ('Number of Immingrants')
plt.show()
```

[ ]:
```
# view actual numbers
df_japan.describe()
```

Scatter Plots

[ ]:
```
# get total immigration per year
df_tot = pd.DataFrame(df_can[years].sum(axis = 0))
df_tot.head()
```

[ ]:
```
# change the years to int
df_tot.index = map(int, df_tot.index)

# reset the index
df_tot.reset_index (inplace = True)

# rename columns
df_tot.columns = ['year', 'total']

# view the final dataframe
df_tot.head()
```

[ ]:
```
# plot data as scatter plot
df_tot.plot (kind = 'scatter', x = 'year', y = 'total', figsize = (10, 6),␣
 ↪color = 'darkblue')

plt.title ('Total Immigration to Canada')
plt.xlabel ('Year')
plt.ylabel ('Number of Immigrants')

plt.show()
```

Get Line of Best Fit

[ ]:
```
x = df_tot['year']
y = df_tot['total']
fit = np.polyfit (x, y, deg = 1)
fit
```

[ ]:
```
# print the regression line
df_tot.plot(kind = 'scatter', x = 'year', y = 'total', figsize = (10,6), color␣
 ↪= 'darkblue')

plt.title ('Total Immigration to Canada')
plt.xlabel ('Year')
plt.ylabel ('Number of Immigrants')

# plot the line of best fit
plt.plot (x, fit[0] * x + fit[1], color = 'red')
plt.annotate ('y={0:.0f} x +{1:.0f}'.format (fit[0], fit[1]), xy = (2000,␣
 ↪150000))

plt.show()
```

Bubble Plots

[ ]:
```
# transposed data frame
df_can_t = df_can[years].transpose()

# change years to type int
df_can_t.index = map (int, df_can_t.index)

# label the index Year
df_can_t.index.name = 'Year'

# reset the index
df_can_t.reset_index (inplace = True)

# view the changes
df_can_t.head()
```

Normalized Weights

X' = (X - Xmin) / (Xmax - Xmin)
X' has a max value of 1 and a min value of 0

[ ]:
```
# normalize Brazil data
x_min = df_can_t['Brazil'].min()
x_max = df_can_t['Brazil'].max()
norm_brazil = (df_can_t['Brazil'] - x_min) / (x_max - x_min)

# normalize Argentina data
x_min = df_can_t['Argentina'].min()
x_max = df_can_t['Argentina'].max()
norm_argentina = (df_can_t['Argentina'] - x_min) / (x_max - x_min)
```

[ ]:
```
# scale the norm to get the weights
# weight = 2000 * norm + 10

# Brazil
ax0 = df_can_t.plot (kind = 'scatter', x = 'Year', y = 'Brazil',
                     figsize = (14, 8), alpha = 0.5,
                     color = 'green',
                     s = norm_brazil * 2000 + 10,
                     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')
ax0.legend (['Brazil', 'Argentina'], loc ='upper left', fontsize = 'x-large')
```