Lab Manual for Tools and Technologies for Data Science

Lab-05

Data Visualization with python

Table of Contents

Lab 5: Data Visualization with python

1. Objective

- Data Wrangling
- Data Cleanup and its usage
- Unnecessary Columns dealing
- Manipulating DataFrame in Python
- Dealing with Missing Values
- Discretizing and Binning
- Aggregation and Grouping in DataFrame
- Outliers Detection
- Outliers removal

2. Data Visualization

In this lab, we will take an in-depth look at the Matplotlib tool for visualization in Python.

3. Plotting and Visualization

Making plots and static or interactive visualizations is one of the most important tasks in data analysis. It may be a part of the exploratory process; for example, helping identify outliers, needed data transformations, or coming up with ideas for models.

4. Matplotlib Introduction

Two of the most commonly used graphical libraries are:

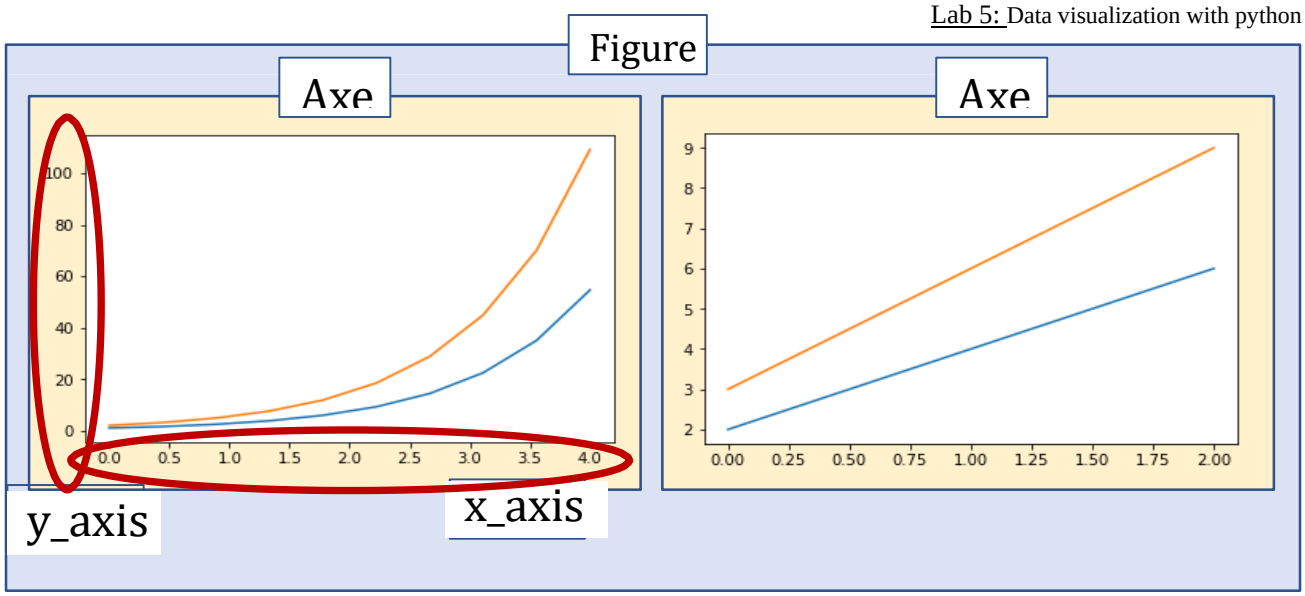
Matplotlib, Seaborn (data visualization library based on Matplotlib)

Before we dive into the details of creating visualizations with Matplotlib, there are a few useful things you should know about using the package.

Importing matplotlib

Just as we use the np shorthand for NumPy and the pd shorthand for Pandas, we will use some standard shorthands for Matplotlib imports:

```
import matplotlib as mpl
import matplotlib.pyplot as plt
```




```
fig, ax = plt.subplots(figsize=(5, 3))
plt.show()
```

- Subplots returns a new **Figure** and its **Axes** object
- **figsize** specifies the figure size (width, height) in inches
- By default ax is a single Axes object (Figure with 1 single)

```
fig, ax = plt.subplots(2, 3, figsize=(5, 3))
plt.tight_layout()
plt.show()
```

- The first two parameters of subplots specify to create a figure with **2 rows, 3 columns** (6 Axes objects)
- **tight_layout()** is necessary at the end to let the subplots fit the frame size without blank spaces at the borders

Drawing a Line plot (Single Axes Object)

```
fig, ax = plt.subplots(figsize=(3, 2))
ax.plot([0,1,2],[2,4,6])
ax.plot([0,1,2],[3,6,9])
plt.show()
```

The plot method of a specific Axes takes as input two lists (or Numpy arrays): x, y coordinates of the points
The default style draws segments passing through the specified coordinates
Subsequent calls of plot add new line to the same Figure.

Drawing a Line plot (Multi Axes Object)

```
fig, ax = plt.subplots(1, 2,
figsize=(3, 2))
ax[0].plot([0,1,2],[2,4,6])
ax[1].plot([0,1,2],[3,6,9])
plt.tight_layout()
plt.show()
```

- The ax object is a **Numpy array** with the created Axes objects
- It has **shape = (n,)** if the figure has 1 row and n columns

4.1 Plot Types

With Matplotlib you can design different plot types
The most common are:

- Line plot
- Scatter plot
- Bar chart

Line Plots:

- Allows displaying a sequence of points/segments that **share the same properties**
  - E.g. same size, color, width, ...

```
x = np.linspace(0, 5, 20)
y = np.exp(x)
fig, ax = plt.subplots(figsize=(3, 2))
ax.plot(x, y, c='red', linestyle='', marker='*')
ax.plot(x, 2*y, c='green', linestyle='--')
plt.show()
```

Line Styles:

```
x = np.linspace(0, 5, 20)
plt.plot(x, x + 0, linestyle='solid')
plt.plot(x, x + 1, linestyle='dashed')
plt.plot(x, x + 2, linestyle='dashdot')
plt.plot(x, x + 3, linestyle='dotted');
plt.show()
```

```
Labels:
plt.title("A Simple Line")
plt.xlabel("x")
plt.ylabel("y");

Or

ax.plot(x, y, c='red', linestyle='', marker='*', label =
'curve1')
ax.plot(x, 2*y, c='green', linestyle='--',label = 'curve 2')
ax.legend(loc=(0, 0.5))
```

- **linestyle** specifies the type of line
  - Examples: '-', '--' (or 'dashed'), ':' (or 'dotted')
- **marker** specifies the type of points to be drawn
  - Examples: 'o', '*', '+', '^'
- **c** specifies the color to be applied to markers and segments
  - Examples: 'red', 'orange', 'grey'
  - Examples: '#0F0F6B' (RGB)
  - Examples: (0.5, 1, 0.8, 0.8) (RGBA tuple)

4.2 Scatter Plot

Allows displaying a set of points and assign them custom properties
E.g. different color, size

Another commonly used plot type is the simple scatter plot, a close cousin of the line plot. Instead of points being joined by line segments, here the points are represented individually with a dot, circle, or other shape.

```
x = np.linspace(0, 10, 30)
y = np.sin(x)
plt.plot(x, y, 'o', color='black')
plt.show()
```

```
x = np.random.rand(20)
y = np.random.rand(20)
colors = x + y
fig, ax = plt.subplots(figsize=(3, 2))
ax.scatter(x, y, c=colors)
plt.show()
```

I. Case-Study

```
Install xlrd package.

df = pd.read_excel('C:/Users/Sheri/Desktop/canada.xlsx',
                   sheet_name='Canada by Citizenship',
                   skiprows=range(20),
                   skipfooter=2)

print('Data read into a dataframe!')
print(df.head())
```

II. Rename columns to make sense

```
df.rename(columns={'OdName':'Country',
'AreaName':'Continent','RegName':'Region'}, inplace=True)
```

III. Column type into string

```
print(all(isinstance(column, str) for column in fd.columns))

df.columns= df.columns.astype(str)
```

.

IV. Make Index column for better selection

```
df.set_index('Country', inplace=True)

Now in last, let's create a list of years from 1980 - 2013
this will come in handy when we start plotting the data

years = list(map(str, range(1980, 2014)))
```

4.3

Case 1

In 2010, Haiti suffered a catastrophic magnitude 7.0 earthquake. The quake caused widespread devastation and loss of life and about three million people were affected by this natural disaster. As part of Canada's humanitarian effort, the Government of Canada stepped up its effort in accepting refugees from Haiti.

Task: Plot a line graph of immigration from Haiti

```
 haiti = df.loc['Haiti', years]
 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()
```

We can clearly notice how number of immigrants from Haiti spiked up from 2010 as Canada stepped up its efforts to accept refugees from Haiti. Let's annotate this spike in the plot by using the plt.text() method.

```
plt.text(2000, 6000, '2010 Earthquake')
plt.show()
```

Task: Let's compare the number of immigrants from Pakistan and China from 1980 to 2013.

```
df_PC = df.loc[['Pakistan', 'China'], years]
df_PC.plot()
plt.show()
```

pandas plots the indices on the x-axis and the columns as individual lines on the y-axis. Since `our` dataframe have `country` as the index and `years` as the columns, we must first transpose the dataframe using `transpose()` method to swap the row and columns.

```
df_PC = df_PC.transpose()
df_PC.plot()
plt.show()
```

Task: Compare the trend of top 5 countries that contributed the most to immigration to Canada.

`First,` **add a 'Total' column that sums up the total immigrants by country over the entire period 1980 – 2013**

```
df['Total'] = df.sum(axis=1)
```

//Get top 5 values

```
df_top5 = df.sort_values(by='Total', ascending=False).head(5)

df_top5 = df_top5[years].transpose()
df_top5.plot(kind='line')
plt.title('Immigrants from top 5 Countries')
plt.xlabel('Years')
```

```
plt.ylabel('Numbers of Immigrants')
plt.show()
```

5. Area Plot

Till far, we created a line plot that visualized the top 5 countries that contributed the most immigrants to Canada from 1980 to 2013. With a little modification to the code, we can visualize this plot as a cumulative plot, also knows as a Stacked Line Plot or Area plot..

```
df_top5.plot(kind='area',
             stacked=False,
             figsize=(20, 10), # pass a tuple (x, y) size
             )
```

The unstacked plot has a default transparency (alpha value) at 0.5. We can modify this value by passing in the `alpha` parameter.

```
df_top5.plot(kind='area',
             alpha = 0.25,
             stacked=False,
             figsize=(20, 10), # pass a tuple (x, y) size
             )
```

Task: **create an unstacked area plot of the 5 countries that contributed the least to immigration to Canada from 1980 to 2013..**

```
df_low5 = df.sort_values(by='Total', ascending=True).head(5)
df_low5 = df_low5[years].transpose()
df_low5.plot(kind='area',
             stacked=False,
             figsize=(20, 10), # pass a tuple (x, y) size
             )
plt.title('Immigrants from Low 5 Countries')
plt.xlabel('Years')
plt.ylabel('Numbers of Immigrants')
plt.show()
```

6. Histograms

A histogram is a way of representing the frequency distribution of numeric dataset. The way it works is it partitions the x-axis into bins, assigns each data point in our dataset to a bin, and then counts the number of data points that have been assigned to each bin. So the y-axis is the frequency or the number of data points in each bin. Note that we can change the bin size and usually one needs to tweak it so that the distribution is displayed nicely.

Task: What is the frequency distribution of the number (population) of new immigrants from the various countries to Canada in 2013?

Before we proceed with creating the histogram plot, let's first examine the data split into intervals. To do this, we will use NumPy's histogram method to get the bin ranges and frequency counts**.**

```
count, bin_edges = np.histogram(df['2013'])

print(count) # frequency count
print(bin_edges) # bin ranges, default = 10 bins
```

By default, the `histrogram` **method breaks up the dataset into 10 bins.**

```
df['2013'].plot(kind='hist', figsize=(8, 5))

plt.title('Histogram of Immigration from 195 Countries in 2013')
# add a title to the histogram
plt.ylabel('Number of Countries') # add y-label
plt.xlabel('Number of Immigrants') # add x-label

plt.show()
```

In the above plot, the x-axis represents the population range of immigrants in intervals of 3412.9. The y-axis represents the number of countries that contributed to the aforementioned population.

Note: Notice that the x-axis labels do not match with the bin size. This can be fixed by passing in a `xticks` **keyword that contains the list of the bin sizes**

```
df['2013'].plot(kind='hist', figsize=(8, 5), xticks=bin_edges)
```

Task: What is the immigration distribution for Denmark, Norway, and Sweden for years 1980 - 2013?

```
df.loc[['Denmark', 'Norway', 'Sweden'], years].plot.hist()
plt.show()
```

Don't worry, you'll often come across situations like this when creating plots. The solution often lies in how the underlying dataset is structured.

Instead of plotting the population frequency distribution of the population for the 3 countries, pandas instead plotted the population frequency distribution for the years.

This can be easily fixed by first transposing the dataset, and then plotting as shown below.

```
df_t = df.loc[['Denmark', 'Norway', 'Sweden'],
years].transpose()
df_t.plot(kind='hist', figsize=(10, 6))

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()
```

Let's make a few modifications to improve the impact and aesthetics of the previous plot:

- increase the bin size to 15 by passing in bins parameter
- set transparency to 60% by passing in alpha paramemter
- label the x-axis by passing in x-label paramater
- change the colors of the plots by passing in color parameter

```
count, bin_edges = np.histogram(df_t, 15)
print(bin_edges)
# un-stacked histogram
df_t.plot(kind ='hist',
          figsize=(10, 6),
          bins=15,
          alpha=0.6,
          xticks=bin_edges,
          color=['coral', 'darkslateblue', 'mediumseagreen']
         )

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()
```

Tip: For a full listing of colors available in Matplotlib, run the following code in your python shell:

```
import matplotlib
for name, hex in matplotlib.colors.cnames.items():
    print(name, hex)
```

6.1 Bar Chart

A bar plot is a way of representing data where the length of the bars represents the magnitude/size of the feature/variable. Bar graphs usually represent numerical and categorical variables grouped in intervals.
To create a bar plot, we can pass one of two arguments via `kind` parameter in `plot()`:

- `kind=bar` **creates a vertical bar plot**
- `kind=barh` **creates a horizontal bar plot**

Vertical bar plot

In vertical bar graphs, the x-axis is used for labelling, and the length of bars on the y-axis corresponds to the magnitude of the variable being measured. Vertical bar graphs are particularly useful in analyzing time series data. One disadvantage is that they lack space for text labelling at the foot of each bar.

Task: Let's compare the number of Icelandic immigrants (country = 'Iceland') to Canada from year 1980 to 2013.

```
df_iceland = df.loc['Iceland', years]
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.show()
```

Horizontal Bar Plot

Sometimes it is more practical to represent the data horizontally, especially if you need more room for labelling the bars. In horizontal bar graphs, the y-axis is used for labelling, and the length of bars on the x-axis corresponds to the magnitude of the variable being measured. As you will see, there is more room on the y-axis to label categorical variables.

Task: Create a horizontal bar plot showing the total number of immigrants to Canada from the top 15 countries, for the period 1980 - 2013.

Get the data pertaining to the top 15 countries.

```
df_top15=df.sort_values(by='Total',ascending=True).tail(15)['Total']
```

1. **Use** `kind='barh'` **to generate a bar chart with horizontal bars.**
2. **Make sure to choose a good size for the plot and to label your axes and to give the plot a title.**
3. **Loop through the countries and annotate the immigrant population using the anotate function of the scripting interface.**

```
df_top15.plot(kind='barh', figsize=(10,10))
plt.title('Immigrants from Top 15 Countries')
plt.xlabel('Number of Immigrants')
plt.ylabel('Countries')
plt.show()
```

6.2 Pie Chart

A pie chart is a circular graphic that displays numeric proportions by dividing a circle (or pie) into proportional slices. You are most likely already familiar with pie charts as it is widely used in business and media. We can create pie charts in Matplotlib by passing in the kind=pie keyword.

Let's use a pie chart to explore the proportion (percentage) of new immigrants grouped by continents for the entire time period from 1980 to 2013.

Step 1: Gather data.

We will use pandas group by method to summarize the immigration data by Continent.

df_continents = df.groupby('**Continent**', axis=0).sum()

Step 2: Plot the data. We will pass in kind = 'pie' keyword, along with the following additional parameters:

- autopct - is a string or function used to label the wedges with their numeric value. The label will be placed inside the wedge. If it is a format string, the label will be fmt%pct.
- startangle - rotates the start of the pie chart by angle degrees counterclockwise from the x-axis.
- shadow - Draws a shadow beneath the pie (to give a 3D feel).

```
df_continents['Total'].plot(kind='pie',
                            figsize=(5, 6),
                            autopct='%1.1f%%', # add in
percentages
                            startangle=90,     # start angle 90°
(Africa)
                            shadow=True,       # add shadow
                            )

plt.title('Immigration to Canada by Continent [1980 - 2013]')
plt.axis('equal') # Sets the pie chart to look like a circle.

plt.show()
```

The above visual is not very clear, the numbers and text overlap in some instances. Let's make a few modifications to improve the visuals:

- **Remove the text labels on the pie chart by passing in** `legend` **and add it as a seperate legend using** `plt.legend()`**.**
- **Push out the percentages to sit just outside the pie chart by passing in** `pctdistance` **parameter.**

Add these values:

```
labels=None,          # turn off labels on pie chart
pctdistance=1.12,
```

7. Evaluation Task (Unseen) [Expected time = 30mins for tasks]

7.1 Practice Task 1

Find the immigrants from Pakistan from 2010 to 2013

7.2 Practice Task 2

Select first sheet in excel file and show region by citizenship analysis using different charts.

7.3 Practice Task 3

Find year in which Canada have maximum immigrants

7.4 Practice Task 4

Compare India and Pakistan from 1980 to 2013 w.r.t to Immigrants numbers

7.5 Practice Task 5

Highlight each country per year that have lowest immigrants to Canada.

8. Evaluation criteria

The evaluation criteria for this lab will be based on the completion of the following tasks. Each task is assigned the marks percentage which will be evaluated by the instructor in the lab whether the student has finished the complete/partial task(s).

Table 3: Evaluation of the Labs

Sr. No.	Description	Marks
1	Problem Modeling	20
2	Procedures and Tools	10
3	Practice tasks and Testing	35
4	Evaluation Tasks (Unseen)	20
5	Comments	5
6	Good Programming Practices	10