

Data Science

Chapter 4: Visualization basics

Chapter outline

This chapter covers

- Introduction to Matplotlib : its architecture
- Line plots
- Area plots
- Histograms
- KDE
- Bar charts
- Pie charts
- Box plots
- Scatter plot
- Bubble plot
- Subplots

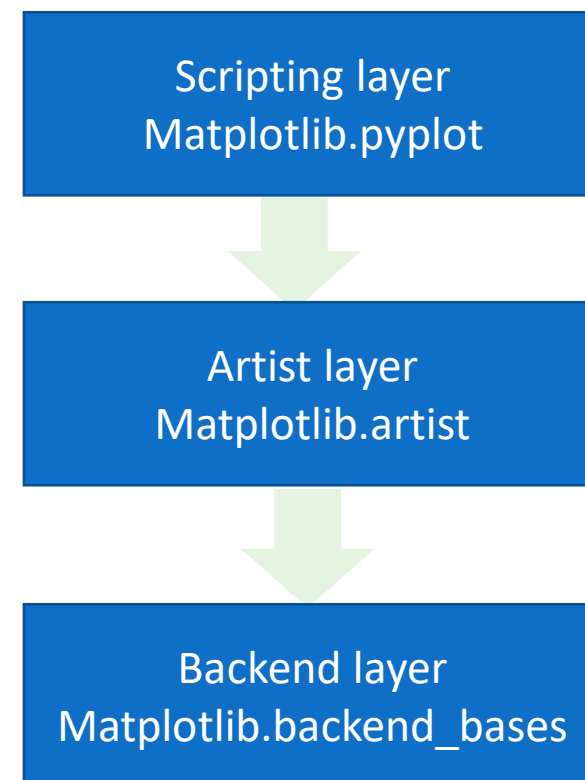
I.

Matplotlib

Matplotlib is the oldest Python plotting library, and it's still the most popular.

The Matplotlib architecture is composed of three main layers:

- ❑ Backend Layer - Handles all the heavy works
- ❑ Artist Layer - Allows full control and fine-tuning of the Matplotlib figure
- ❑ Scripting Layer - The lightest scripting interface among the three layers, designed to make Matplotlib work like MATLAB script.



I.

Matplotlib

The backend layer handles all the heavy works via communicating to the toolkits like wxPython or drawing languages like PostScript in your machine. It is the most complex layer of the Matplotlib library.

It has three main built-in abstract interface classes:

- ❑ **FigureCanvas** - `matplotlib.backend_bases.FigureCanvasBase` The canvas the figure renders into.
- ❑ **Renderer** - `matplotlib.backend_bases.RendererBase` An abstract base class to handle drawing/rendering operations. Responsible to draw in the FigureCanvas.
- ❑ **Event** - `matplotlib.backend_bases.` Event Handles user inputs such as keyboard and mouse clicks.

A regular user hardly has to deal with this layer, which is why we won't go into more detail.

I.

Matplotlib

The artist layer allows control and fine-tune as many elements as possible in the figure just like an artist paints on the canvas.

- ❑ This layer is comprised of one main object, Artist, that uses the Renderer to draw on the canvas. It allows you to do more customization compare to the Scripting layer and is more convenient for advanced plots. Especially when handling multiple figures/axes, you will not get confused as to which one is currently active since every subplot is assigned to an Artist object.
- ❑ That's why the Artist Layer is sometimes called object-based plotting.
- ❑ Everything visible on a Matplotlib figure is an artist instance: The title, the lines, the tick labels, the images, and so on, all correspond to an individual artist.
- ❑ There are two types of Artist objects. The first type is the primitive type, such as a Line2D, Rectangle, Circle, and Text. And the second type is the composite type, such as the Axis, Tick, Axes and Figure.
- ❑ It is important to notice that each composite artist may contain other composite artists as well as primitive artists. For example a figure artist can contain an axis artist as well as a rectangle or text artist.

I.

Matplotlib

The scripting layer is the top layer, designed to make Matplotlib work like MATLAB script.

- ❑ This is why many Matplotlib tutorials prefer to introduce from this layer. It is the easiest part to start with and use, you add up objects — e.g. line, text, rectangle- on top of the figure.
- ❑ Scripting layer plotting is sometimes also called procedural plotting.
- ❑ Matplotlib's scripting layer is essentially the `matplotlib.pyplot` interface, which automates the process of defining a canvas and defining a figure artist instance and connecting them.

II. Matplotlib - Pandas

- We can plot easily a dataframe

```
df.plot(kind='line')
```

- We can plot an histogram of an attribute

```
df["column_name"].plot(kind="hist")
```

III.

Line plots

➤ Definition

A line plot is a type of plot which displays information as a series of data points called 'markers' connected by straight line segments.

➤ When to use lineplots

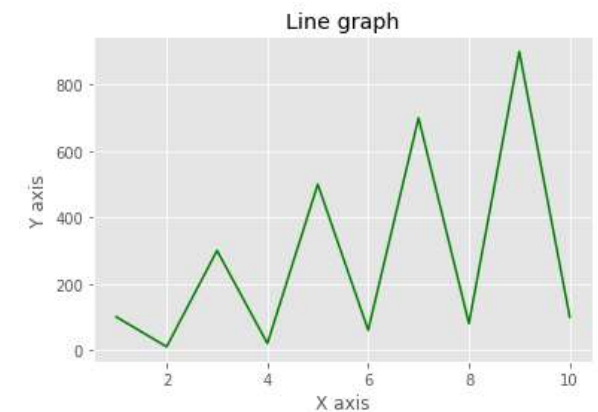
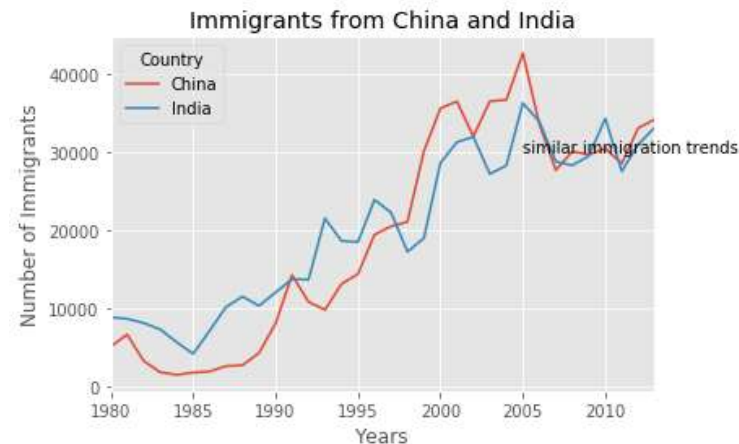
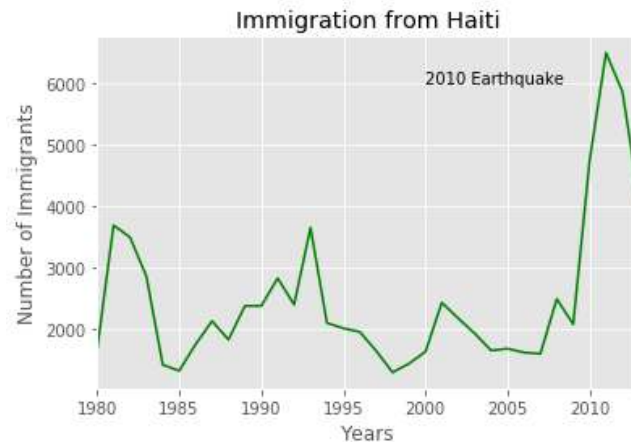
The best use case for a line plot is when you have a continuous dataset and you're interested in visualizing the data over a period of time. As an example, we're interested in the trend of immigrants from Haiti to Canada.

Typical examples of the types of data that can be presented using line graphs are monthly rainfall and annual unemployment rates.

III.

Line plots in python

- Using Pandas : ex: `df_china_india.plot(color=['green', 'red'])`
 - You can plot a pandas Series : the keys are the x and the values are the y. Example of immigration from Haiti.
 - You can plot a pandas Dataframe : the index is the x and the columns are the variables to plot. That's why you have to transpose sometimes the dataframe. Example of immigration from China and India – the data of China & India are rows so you transpose the data to have it as columns and the years as index.
- Plot a numpy array (Specify X and Y)



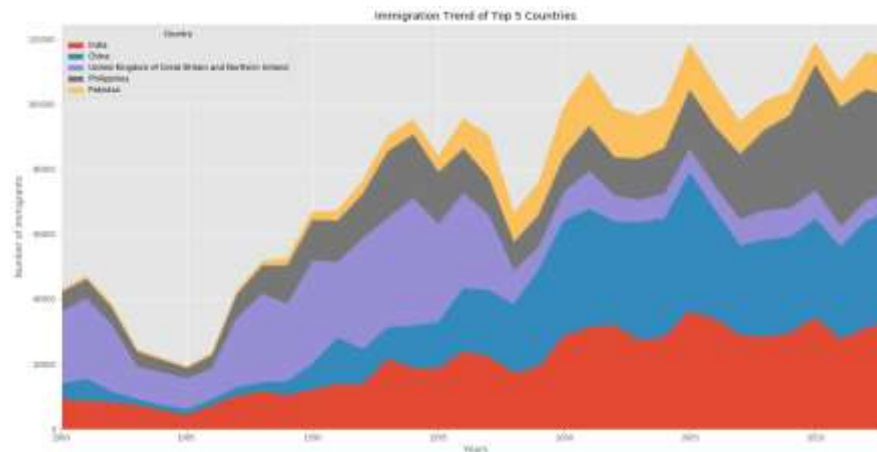
IV. Area plot

➤ Definition

Area charts are great to show how a total developed over time. They are not easy to read, so there are many cases when a line chart or a stacked column chart is a better choice.

➤ When to use Areaplots

Area charts are perfect when communicating the **overall trend**, as opposed to the individual values.



IV.

Area plots types

- **Stacked** Area Chart: it's like several area charts stacked on top of one another. Each data series representation begins where the previous data series ends (they do not overlap). The final figure indicates the sum of all the data represented.
- **Overlapped** Area charts : it shows a filled area below the quantity line. This fill can obscure other data series that are on the same chart. One way to solve this occlusion problem is with transparency of the areas. Transparency works well with two or three different areas, but if used with too many areas, it turns into a muddy mess.
- **Percentage** Area charts: : An area plot drawn to plot variables their summation =1.

IV.

How to calculate the percentage ?

	China	Inda	Pakistan	SUM
2000	250	340	245	835
2001	300	350	500	1150

	China	Inda	Pakistan	SUM =1
2000	250/835	340/835	245/835	1
2001	300/1150	350/1150	500/1150	1

```
normalizedDataFrame = df_top5.div(df_top5.sum(axis=1), axis=0);
```

- sum the number for each row on all columns → you will obtain the SUM in the first table
- divide the dataframe by column SUM. (take each columns and divide it)

IV.

Area plots in python

- Using Pandas (same as line): Ex: `df_top5.plot(kind='area',figsize=(20, 10), color= ['yellow','green','yellow','green','yellow'])`
- You can plot a pandas Series : the keys are the x and the values are the y. Example of immigration from Haiti.
- You can plot a pandas Dataframe : the index is the x and the columns are the variables to plot. That's why you have to transpose sometimes the dataframe. Example of immigration from China and India – the data of China & India are rows so you transpose the data to have it as columns and the years as index.
- Plot a numpy array (Specify X and Y)

V.

Histogram

➤ Definition

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.

➤ Where to use an histogram

The data are numerical. You want to see the shape of the data's distribution, especially when determining whether the output of a process is distributed approximately normally.

V.

Histogram plots in python

- Using Pandas
 - You can plot a pandas Series : the keys are the x and the values are the y. Example of immigration from Haiti.
 - You can plot a pandas Dataframe : the index is the x and the columns are the variables to plot. That's why you have to transpose sometimes the dataframe. Example of immigration from China and India – the data of China & India are rows so you transpose the data to have it as columns and the years as index.
- Plot a numpy array (example: `np.histogram([1, 1, 2, 2, 1], bins=[0, 1, 2, 3])`)
- Some parameters: bins, alpha, stacked, xticks

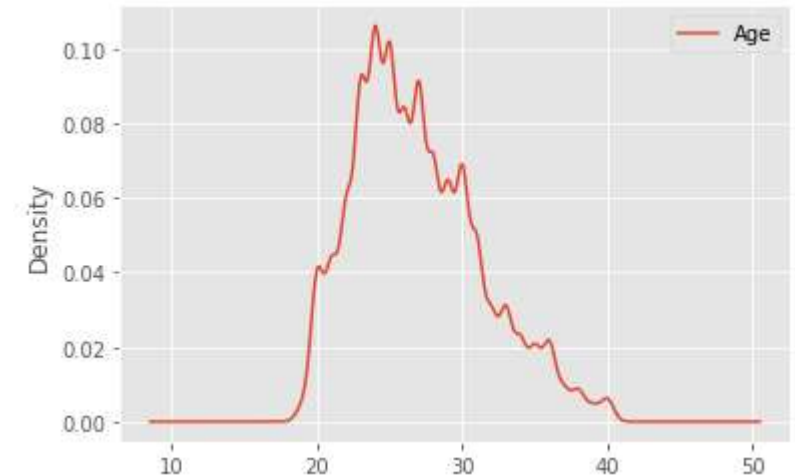
VI. KDE

➤ Definition

KDE Plot described as Kernel Density Estimate is used for visualizing the Probability Density of a continuous variable. It depicts the probability density at different values in a continuous variable. We can also plot a single graph for multiple samples which helps in more efficient data visualization.

➤ Where to use KDE

When you want to visualize the probability density for one or more variables



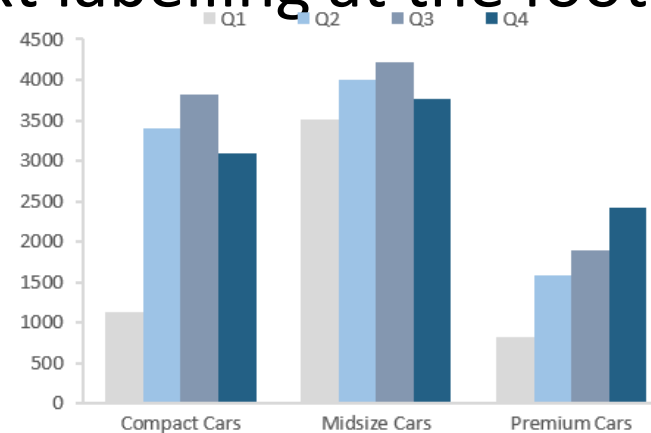
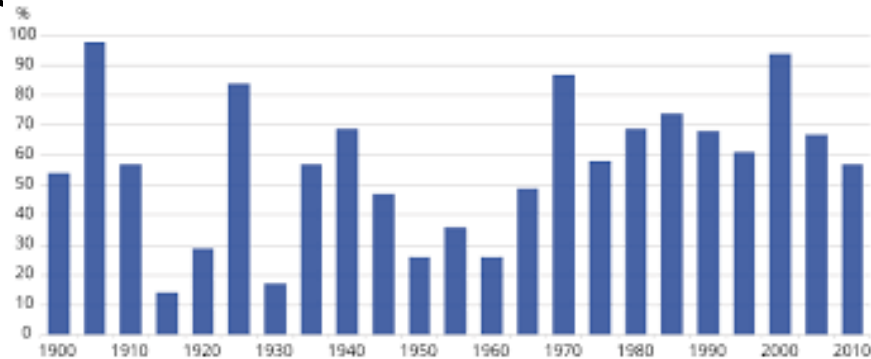
VII. Bar Chart

➤ Definition

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.

➤ Where to use a Bar chart

Vertical bar graphs are particularly useful in analyzing time series data- to track changes over time or to compare things between different groups. One disadvantage is that they lack space for text labelling at the foot of each bar.



➤ Definition

A pie chart (or a circle chart) is a circular statistical graphic, which is divided into slices to illustrate numerical proportion. In a pie chart, the arc length of each slice (and consequently its central angle and area), is proportional to the quantity it represents.

➤ Where to use Pie Chart

A pie chart is best used when trying to work out the composition of something. If you have categorical data then using a pie chart would work really well as each slice can represent a different category.

Pie charts are good for displaying data for around **6 categories** or fewer.

IX.

Box Plot

➤ Definition

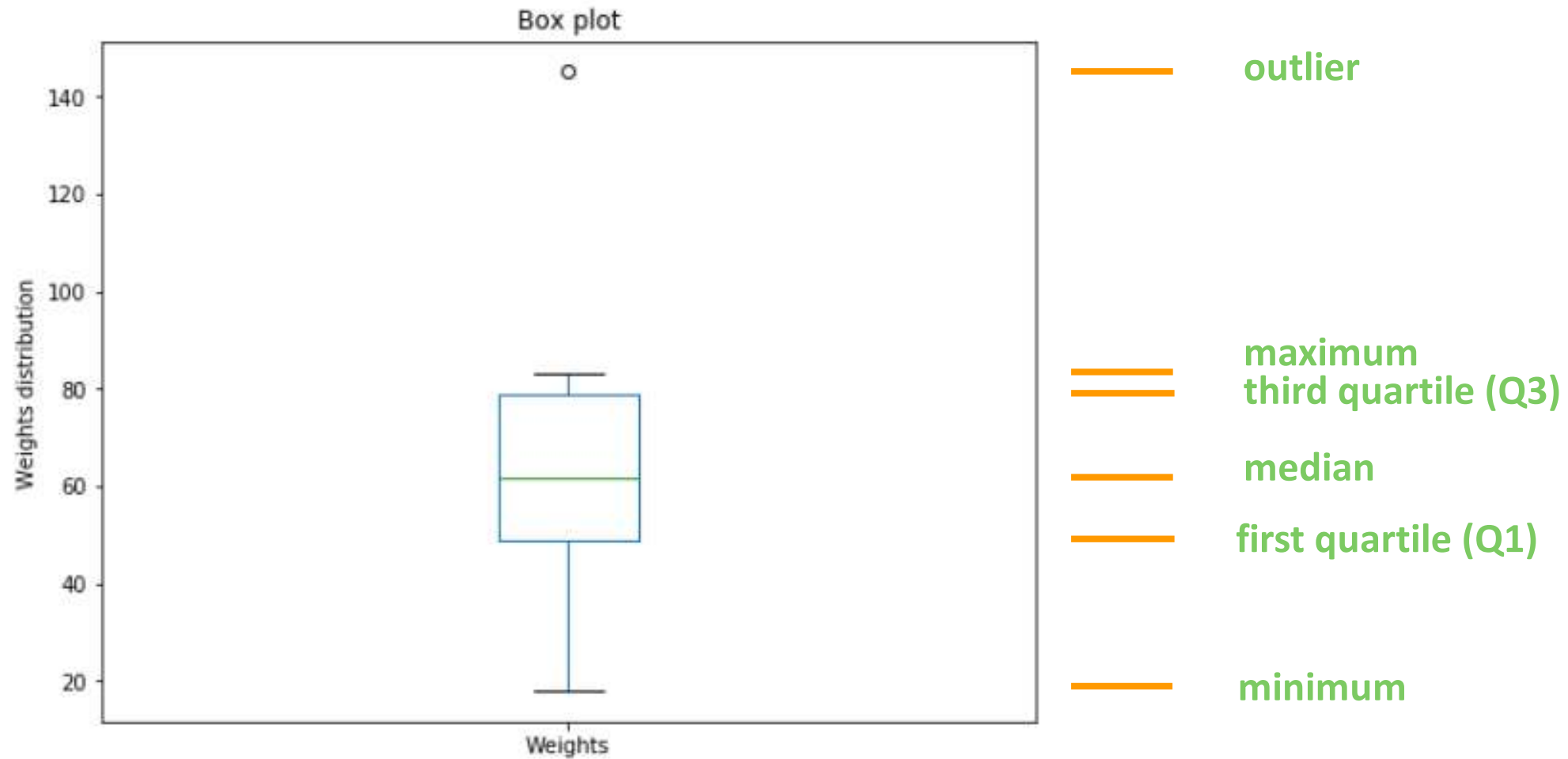
A box and whisker plot—also called a box plot—displays the five-number summary of a set of data. The five-number summary is the minimum, first quartile, median, third quartile, and maximum.

➤ Where to use Box plot

Box plots are useful as they provide a visual summary of the data enabling researchers to quickly identify mean values, the dispersion of the data set, and signs of skewness.

IX.

Box plot



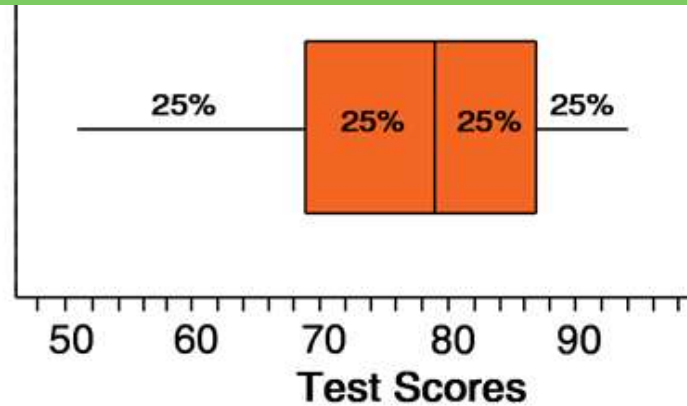
IX.

Box plot

- Weights = 79 47 55 83 145 44 61 18 78 62
- Step 1: order the list → 18 44 47 55 61 62 78 79 83 145
- Step 2: Median = $(61+62)/2 = 61.5$
- Step 3: $Q1 = \text{median}(18, 44, 47, 55, 61) = 47$
- Step 4: $Q3 = \text{median}(62, 78, 79, 83, 145) = 79$
- Step 5: $IQR = Q3 - Q1 = 79 - 47 = 32$
- Step 6: $MWL = IQR \times 1.5 = 32 \times 1.5 = 48$
- Step 7: minimum = $Q_1 - MWL = 47 - 48 = 1$ (Since there are no data values below the lower bound, the minimum is 18).
- Step 8: maximum = $Q_3 + MWL = 79 + 48 = 127$ (145 is above the upper fence. Thus 145 is a possible outlier and 83 is the maximum)

IX.

Box plot



- **Inter-quartile range**
The middle “box” represents the middle 50% of scores for the group. The range of scores from lower to upper quartile is referred to as the inter-quartile range. The middle 50% of scores fall within the inter-quartile range.
- **Upper quartile**
Seventy-five percent of the scores fall below the upper quartile.
- **Lower quartile**
Twenty-five percent of scores fall below the lower quartile.
- **Whiskers**
The upper and lower whiskers represent scores outside the middle 50%. Whiskers often (but not always) stretch over a wider range of scores than the middle quartile groups.

X.

Scatter plot

➤ Definition

A scatter plot is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data.

➤ When to use scatter plots

- When you have paired numerical data
- When trying to determine whether the two variables are related, such as: When trying to identify potential root causes of problems

XI. Bubble plot

➤ Definition

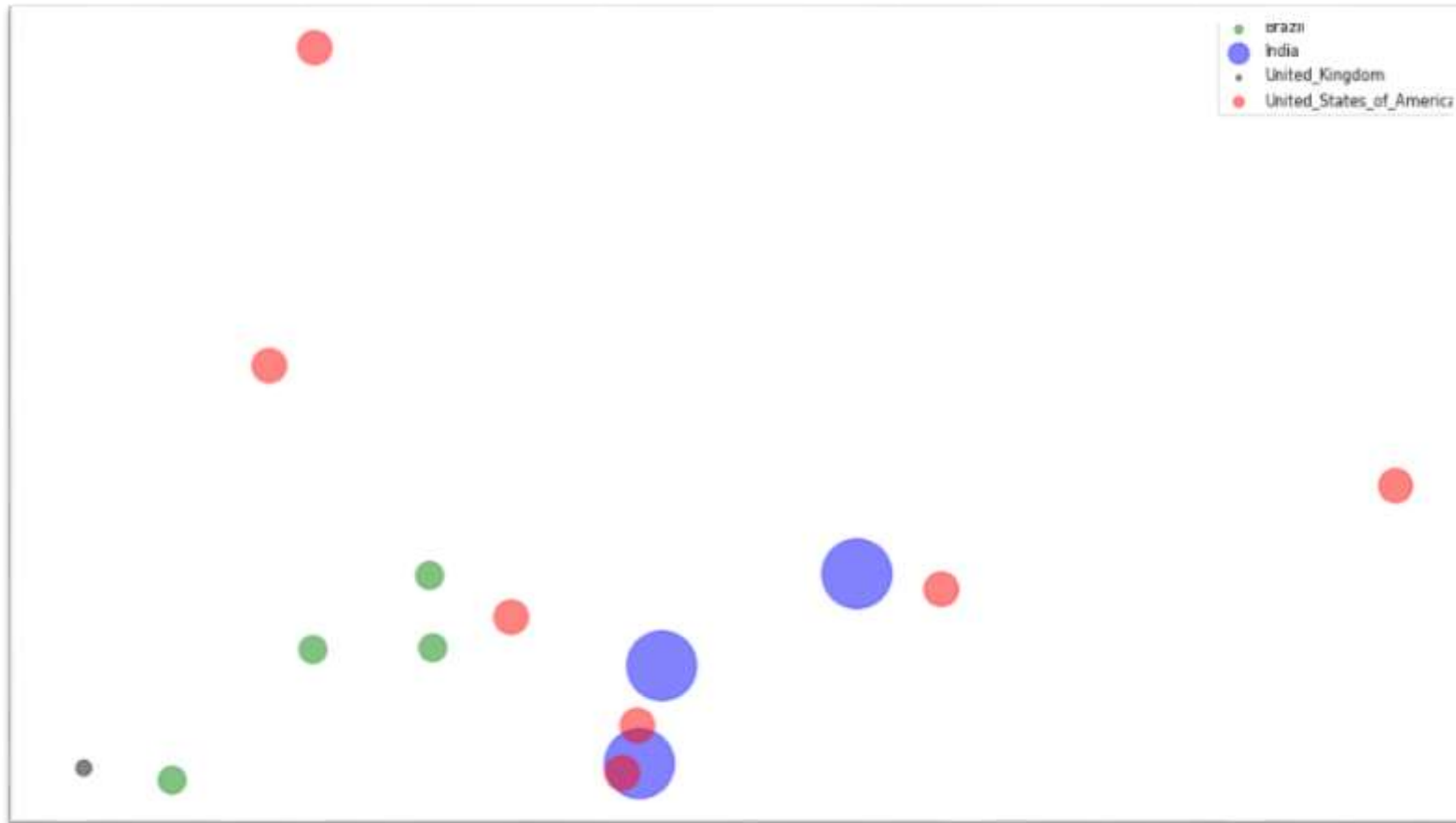
A bubble chart is a data visualization that displays multiple circles (bubbles) in a two-dimensional plot. It is a generalization of the scatter plot, replacing the dots with bubbles. Most commonly, a bubble chart displays the values of three numeric variables, where each observation's data is shown by a circle ("bubble"), while the horizontal and vertical positions of the bubble show the values of two other variables.

➤ When to use scatter plots

You can **use a bubble chart** instead of a scatter **chart** if your data has three data variables or four.

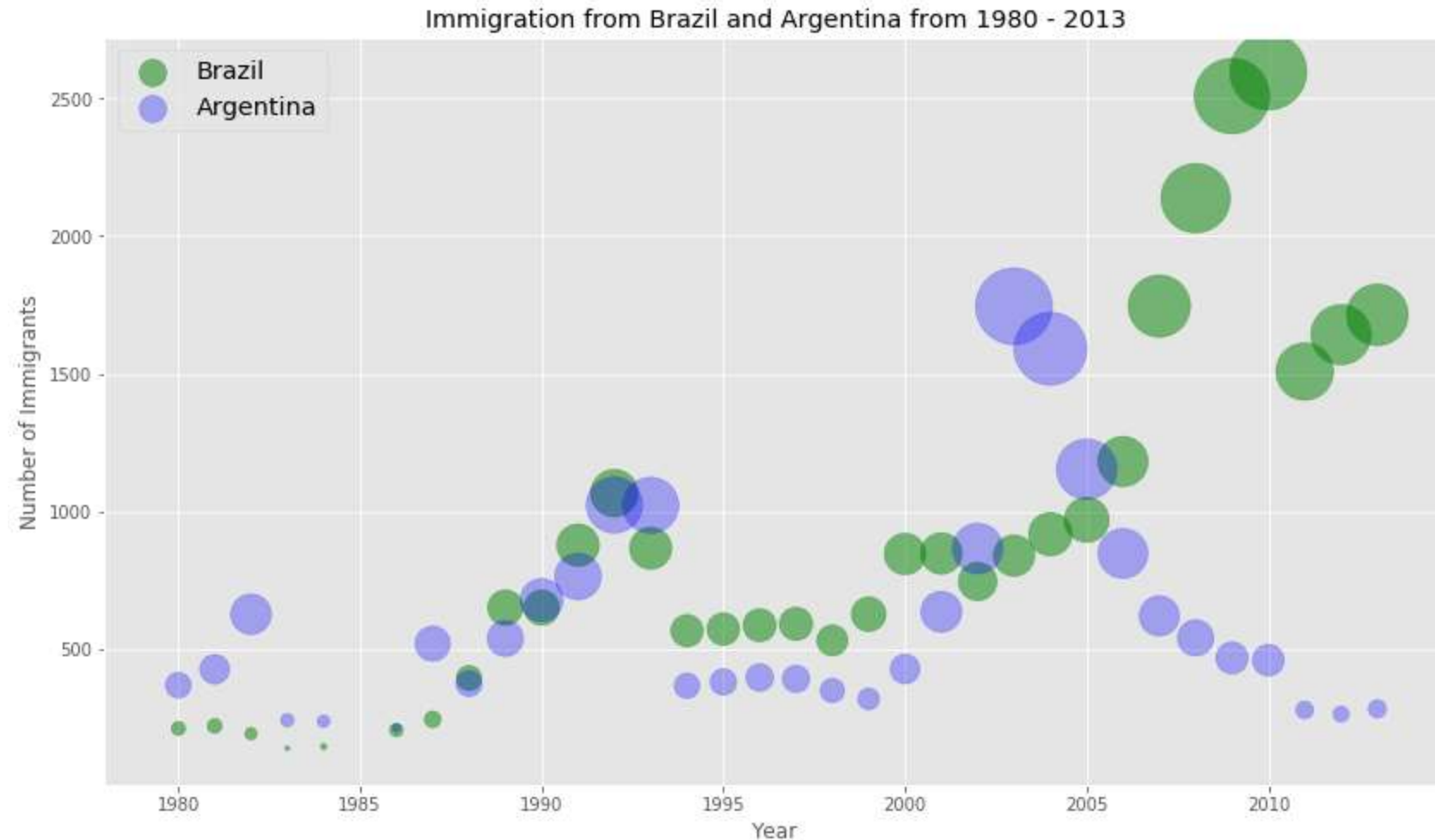
XI. Bubble plot example

- Example: Plot many variables for each data row



XI. Bubble plot example

- Example: compare two variables, based on 3 variables



XII. Subplots

- Often times we might want to plot multiple plots within the same figure. For example, we might want to perform a side by side comparison of the box plot with the line plot of China and India's immigration.
- 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.
- Typical syntax is :
 - `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.
- For more: <https://matplotlib.org/3.1.0/tutorials/intermediate/gridspec.html>

