APPLIED DATA SCIENCE WITH PYTHON

MODULE 5 – VISUALISATION

OVERVIEW

- Introduction
- Matplotlib
- Pandas
- Seaborn

INTRODUCTION

Visualisation tools can provide us, as Data Scientist a visual insight into the data that we are working with.

In this module we will explore a few libraries that can help us visualise our data.

MATPLOTLIB

Matplotlib is one of the staple tools used for visualising data.

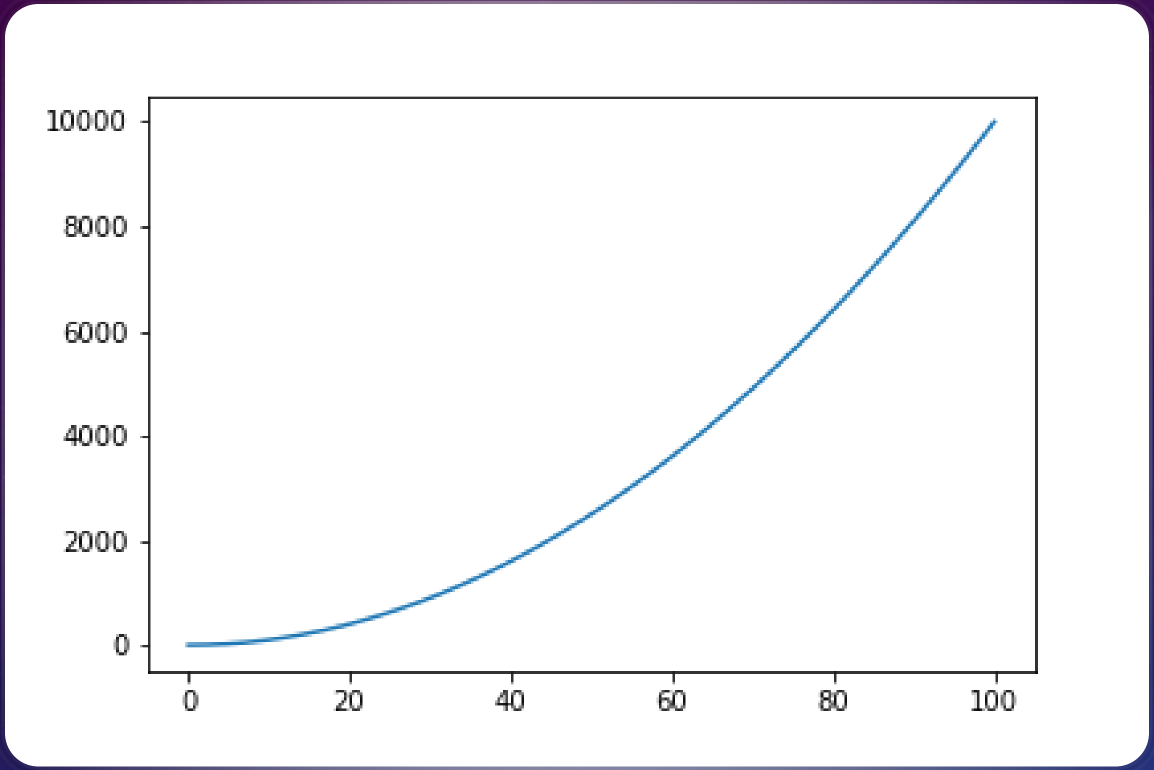
We can install matplotlib via as follows:

```
pip install matplotlib
```

Or via conda as follows:

```
conda install matplotlib
```

CREATING A BASIC PLOT

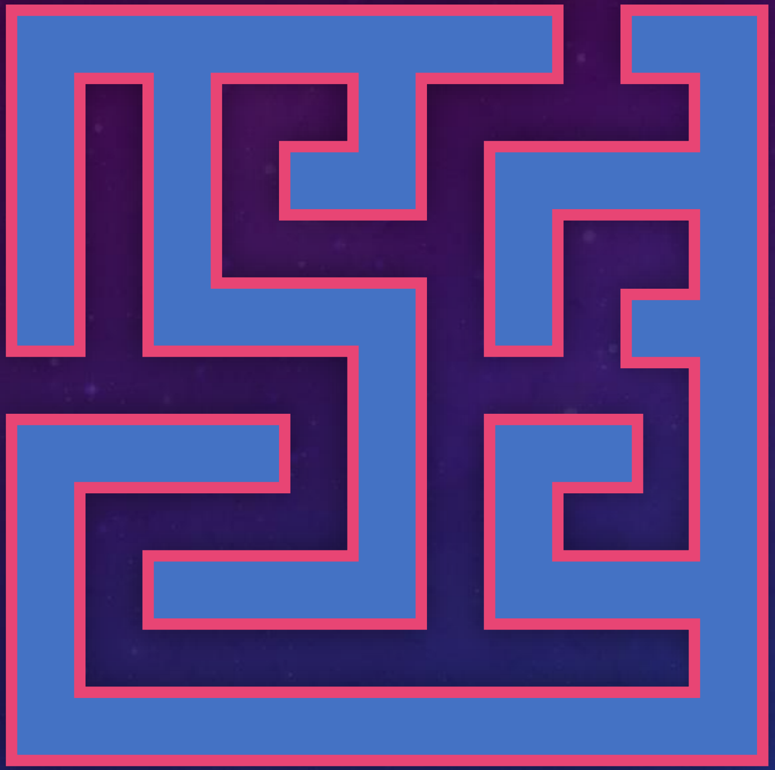


A basic plot can be created by using the following method:

- plt.plot (x, y)

Using the following values below, we get the corresponding plot diagram.

- x = np.linspace(0, 100, 1000)
- y = np.power(x, 2)



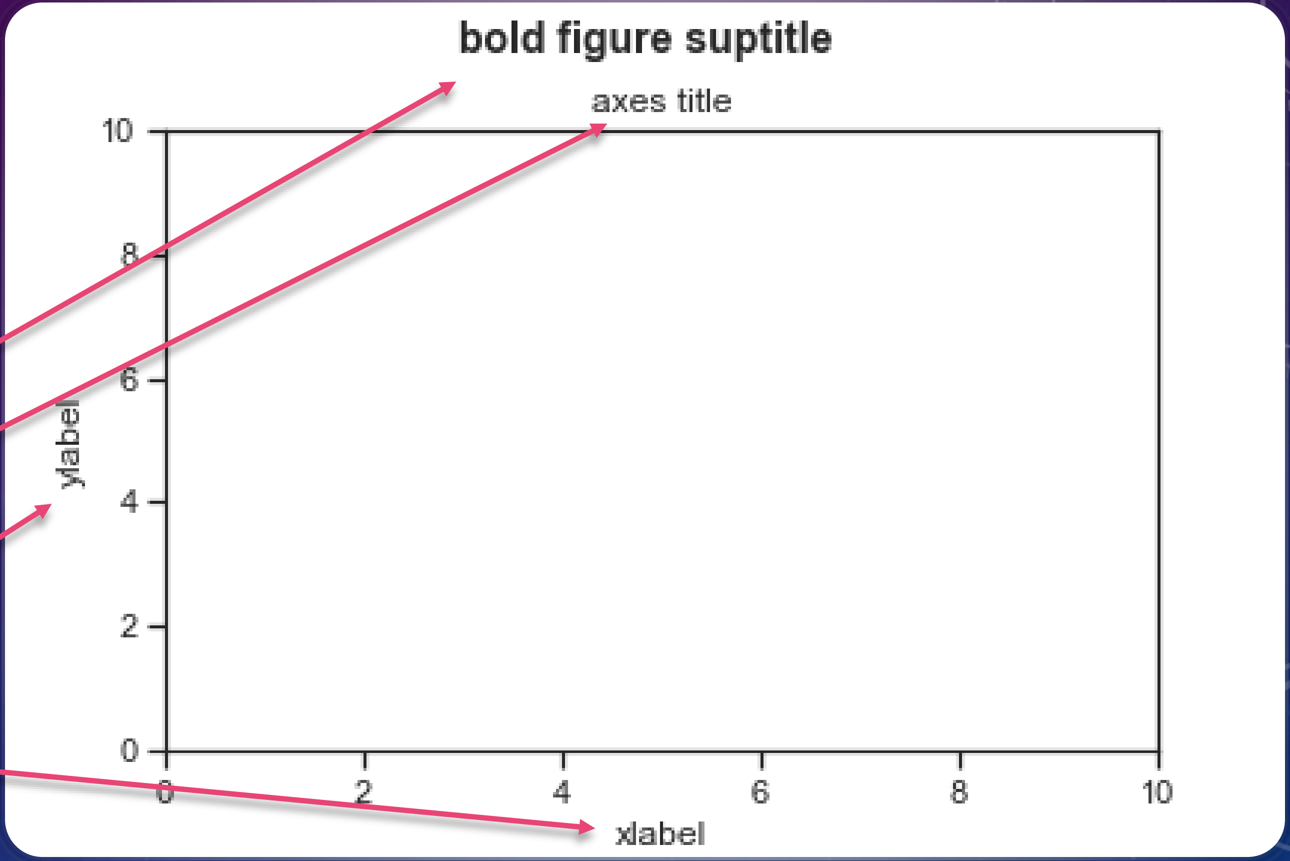
LEGEND'S AND ANNOTATION

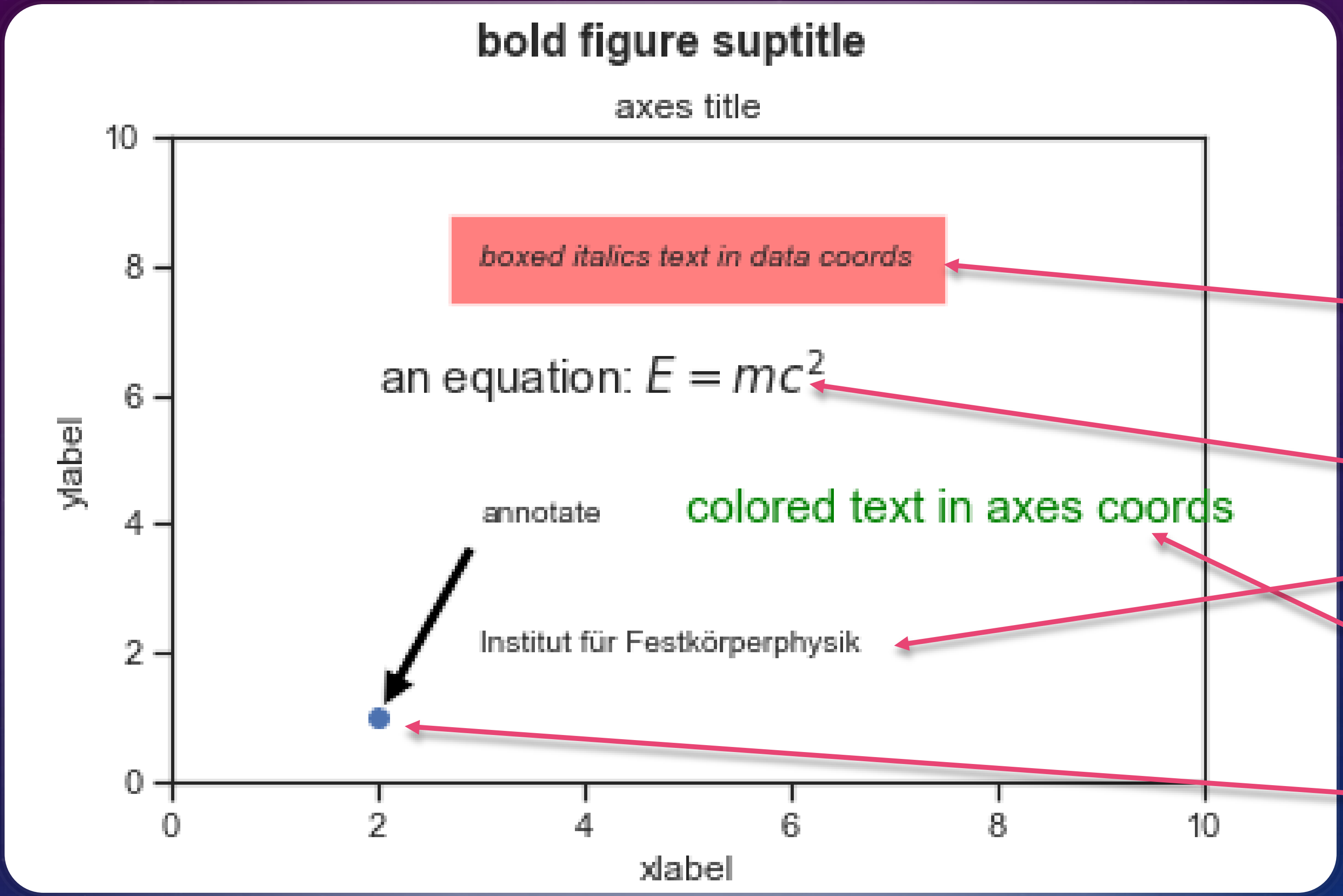
Legends and annotations are effective tools to display information required to comprehend a plot in a glance. A typical plot will have the following additional information elements:

- A legend describing the various data series in the plot. This is provided by invoking the matplotlib **legend()** function and supplying the labels for each data series.
- Annotations for important points in the plot. The matplotlib **annotate()** function can be used for this purpose. A matplotlib annotation consists of a label and an arrow. This function has many parameters describing the label and arrow style and position, so you may need to call **help(annotate)** for a detailed description.
- Labels on the horizontal and vertical axes. These labels can be drawn by the **xlabel()** and **ylabel()** functions. We need to give these functions the text of the labels as a string and optional parameters such as the font size of the label.
- A descriptive title for the graph with the matplotlib **title()** function. Typically, we will only give this function a string representing the title.
- A grid is also nice to have in order to localize points easily. The matplotlib **grid()** function turns the plot grid on and off.

TITLES AND LABELS

- In this image example we are using the following functions:
- suptitle('bold figure suptitle', fontsize=14, fontweight='bold')
- set_title('axes title')
- set_xlabel("xlabel")
- set_ylabel("ylabel")



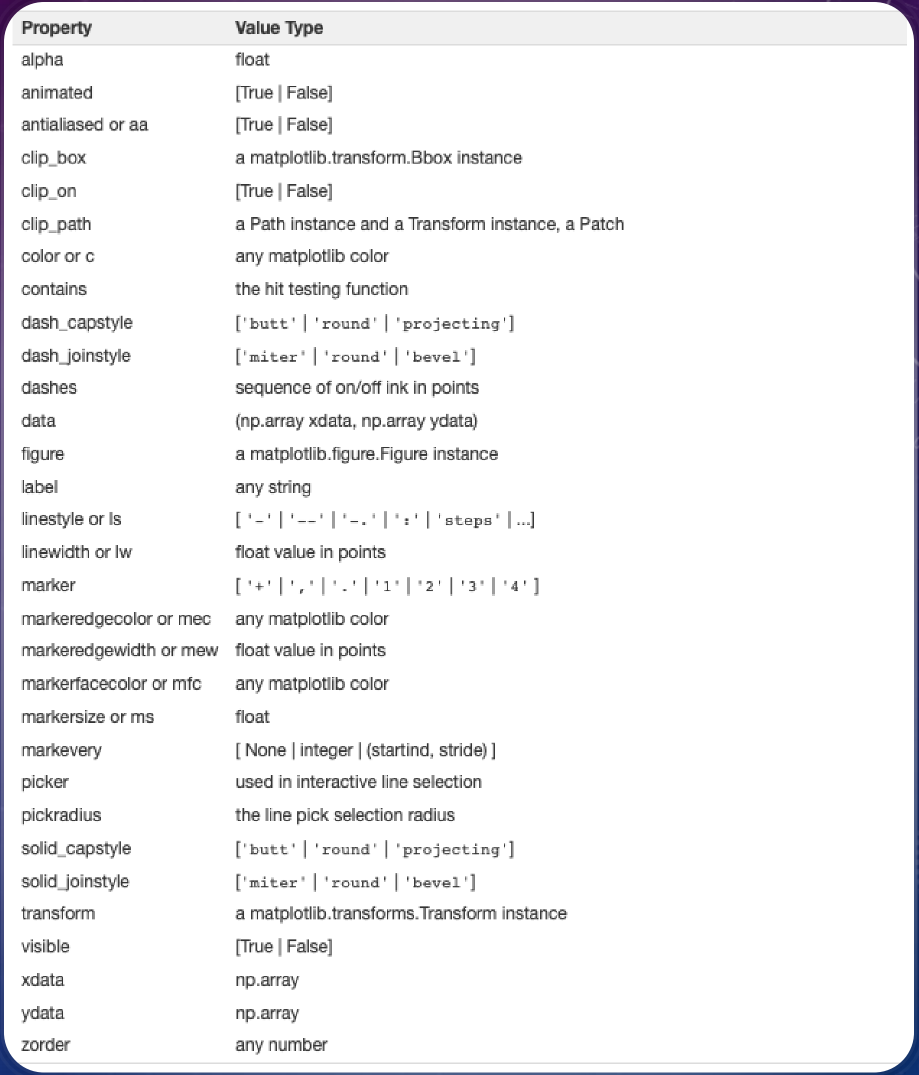


MORE INFORMATIVE INFORMATION

- In this image example we are using the following functions:
- Insert a boxed text
  - text(3, 8, 'boxed italics text in data coords', style='italic', bbox={'facecolor':'red', 'alpha':0.5, 'pad':10})
- Insert a latex equation
  - text(2, 6, r'an equation: $E=mc^2$', fontsize=15)
- Add unicodes
  - text(3, 2, u'Institut für Festkörperphysik')
- Colour Text
  - text(5, 4, 'colored text in axes coords',color='green', fontsize=15)
- Annotate a point
  - plot([2], [1], 'o')
  - annotate('annotate', xy=(2, 1), xytext=(3, 4),arrowprops=dict(facecolor='black', shrink=0.05))

CONTROLLING LINE PROPERTIES

There are many proprty items that can be used to control a line in a plot. This is a comprehensive list and we will make use of many of the propterties moving forward in our activities.

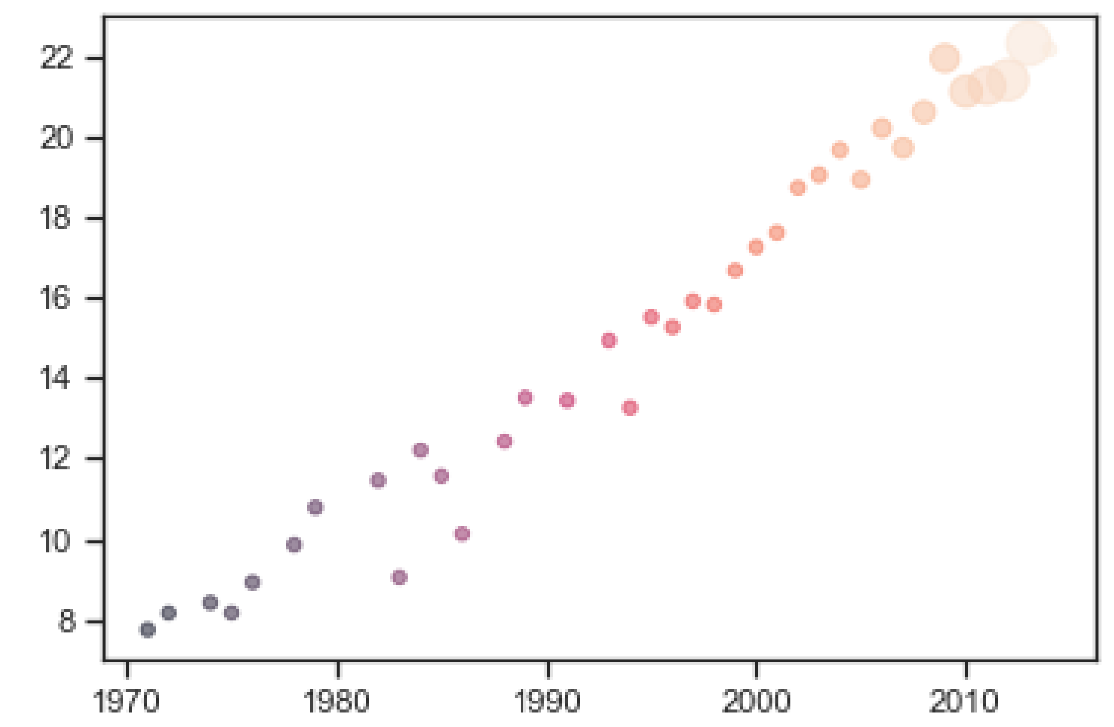


Property	Value Type
alpha	float
animated	[True | False]
antialiased or aa	[True | False]
clip_box	a matplotlib.transform.Bbox instance
clip_on	[True | False]
clip_path	a Path instance and a Transform instance, a Patch
color or c	any matplotlib color
contains	the hit testing function
dash_capstyle	['butt' | 'round' | 'projecting']
dash_joinstyle	['miter' | 'round' | 'bevel']
dashes	sequence of on/off ink in points
data	(np.array xdata, np.array ydata)
figure	a matplotlib.figure.Figure instance
label	any string
linestyle or ls	['-' | '--' | '-.' | ':' | 'steps' | ...]
linewidth or lw	float value in points
marker	['+' | ',' | '.' | '1' | '2' | '3' | '4']
markeredgecolor or mec	any matplotlib color
markeredgewidth or mew	float value in points
markerfacecolor or mfc	any matplotlib color
markersize or ms	float
markevery	[None | integer | (startind, stride)]
picker	used in interactive line selection
pickradius	the line pick selection radius
solid_capstyle	['butt' | 'round' | 'projecting']
solid_joinstyle	['miter' | 'round' | 'bevel']
transform	a matplotlib.transforms.Transform instance
visible	[True | False]
xdata	np.array
ydata	np.array
zorder	any number

SCATTER PLOT 2D

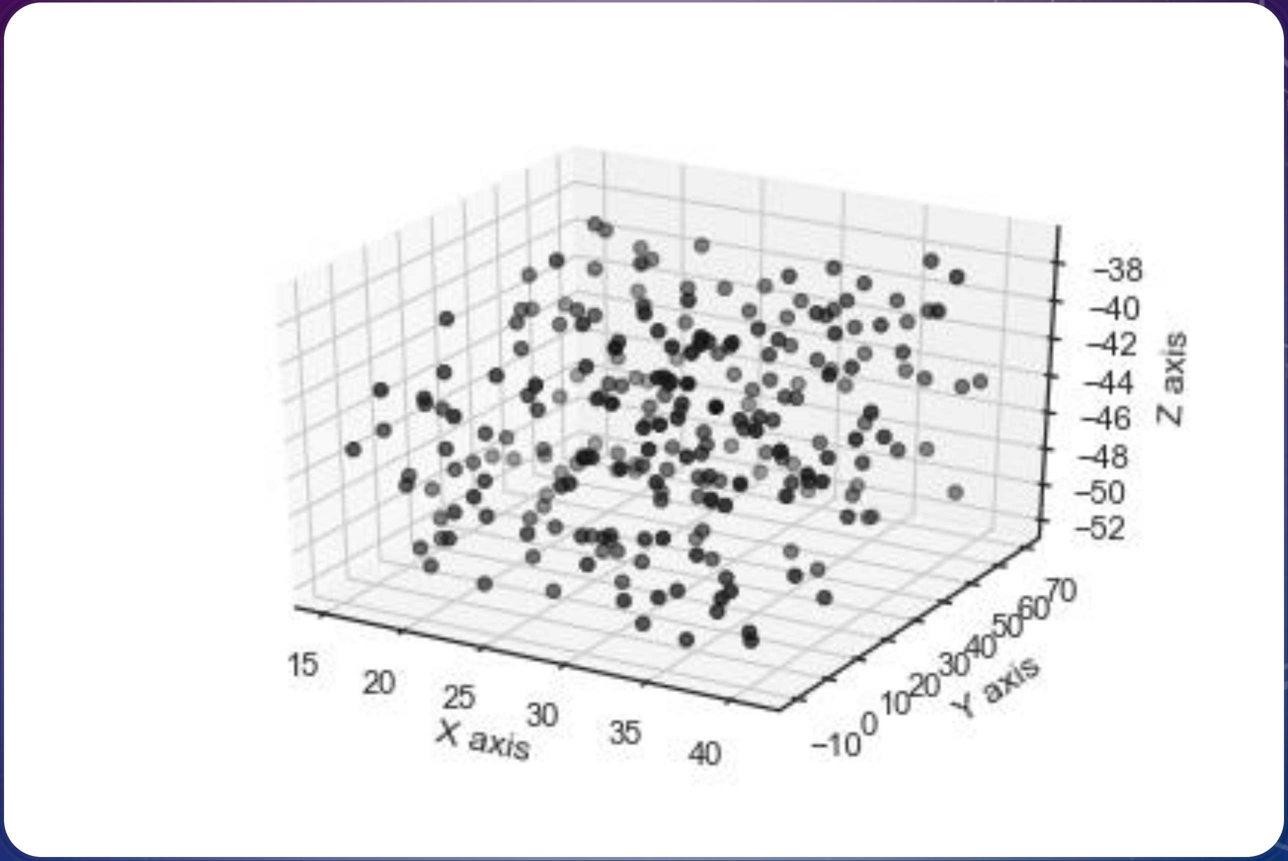
- A scatter plot shows the relationship between two variables in a Cartesian coordinate system.
- The position of each data point is determined by the values of these two variables.
- The scatter plot can provide hints for any correlation between the variables under study.
- An upward trending pattern suggests positive correlation.

- A bubble chart is an extension of the scatter plot. In a bubble chart, the value of a third variable is relatively represented by the size of the bubble surrounding a data point, hence the name.



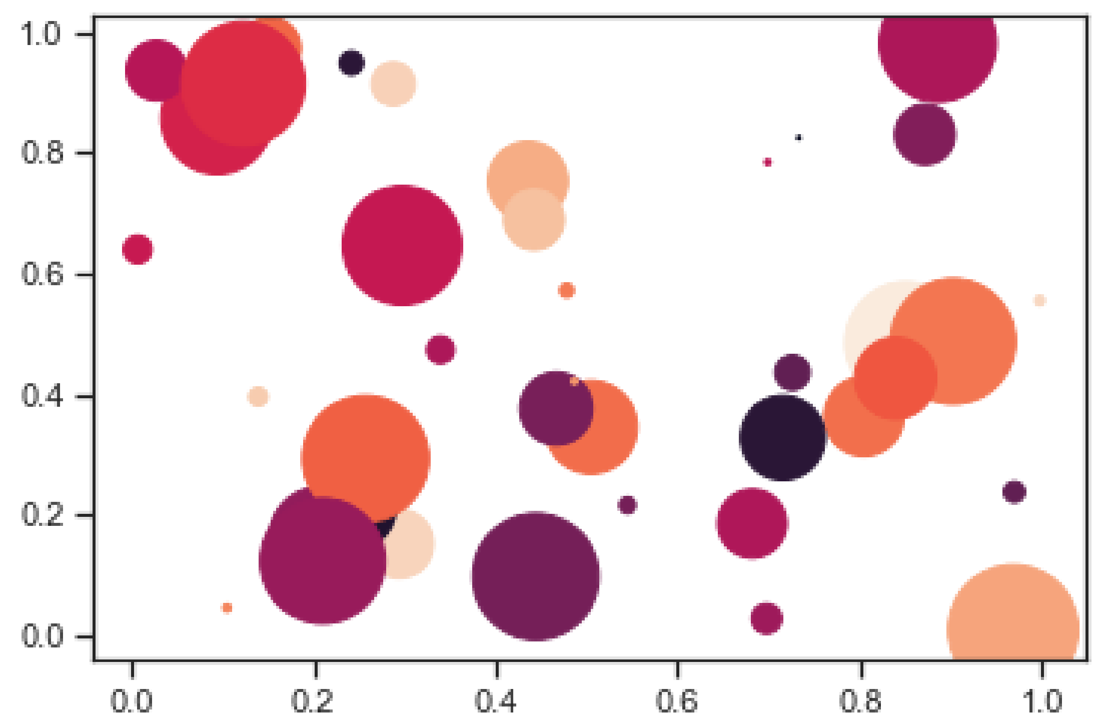
SCATTER PLOT 3D

- The relationships between quantitative variables can be represented by using a scatter plot. A version of this graph is represented by the three-dimensional scatter plots that are used to show the relationships between three variables.



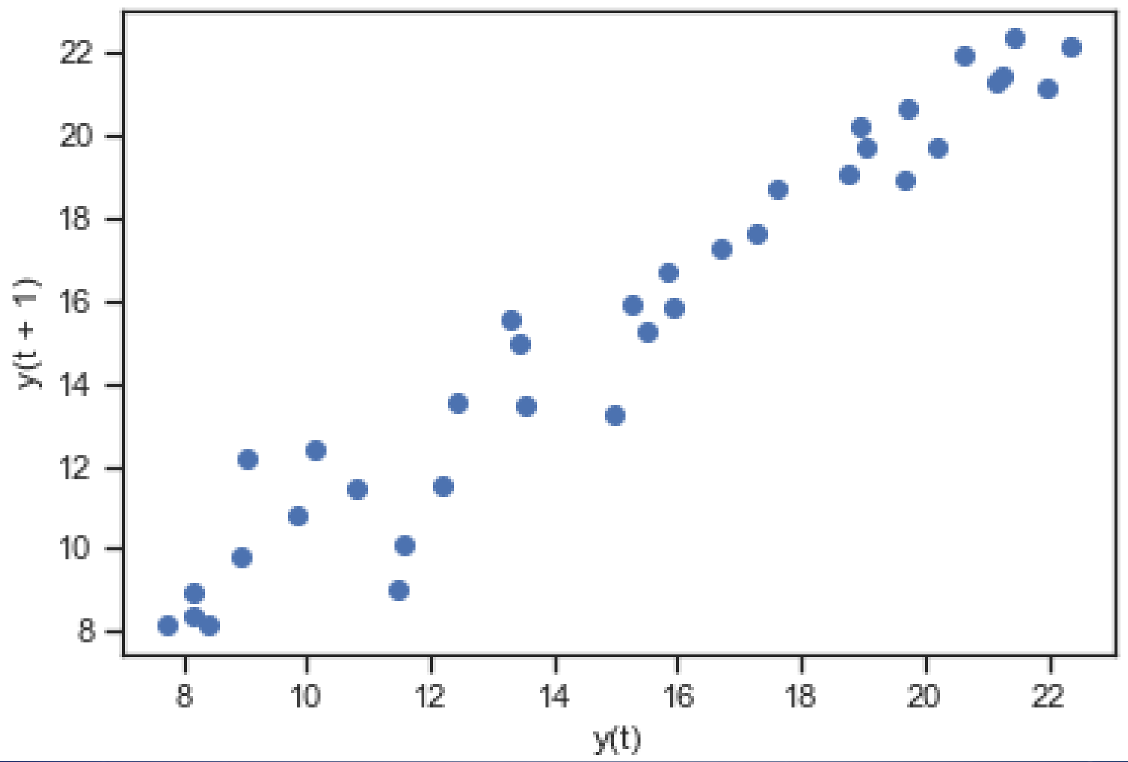
PLOTTING BUBBLES

- A bubble plot is a type of chart in which each represented entity is defined in terms of three distinct numerical parameters. The first two parameters are used as values of the two Cartesian axes, while the third is used to find the radius of the bubble. Bubble plots are used to describe relationships in various scientific fields.



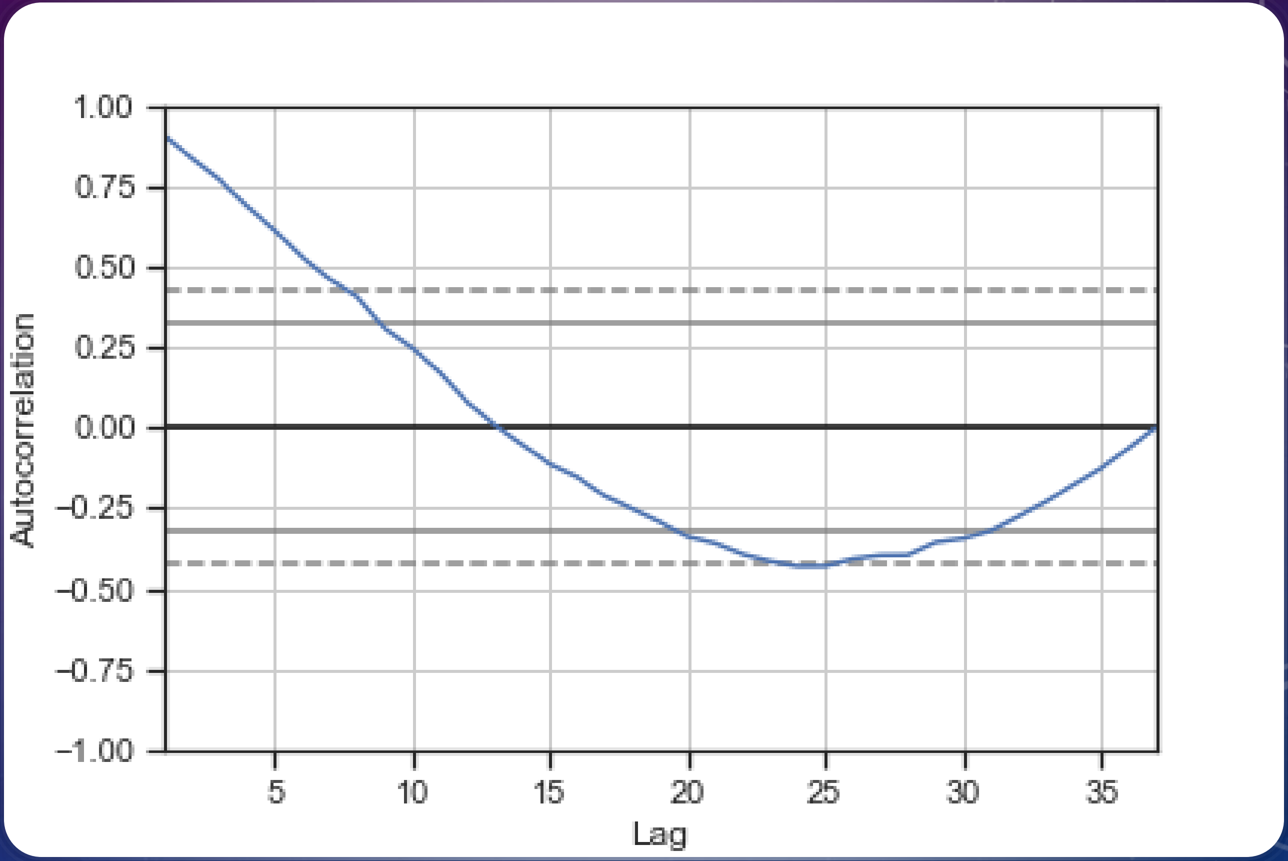
LAG PLOTS

- A lag plot is a scatter plot for a time series and the same data lagged. With such a plot, we can check whether there is a possible correlation between CPU transistor counts this year and the previous year, for instance.
  The **lag_plot()** pandas function in **pandas.plotting** can draw a lag plot.



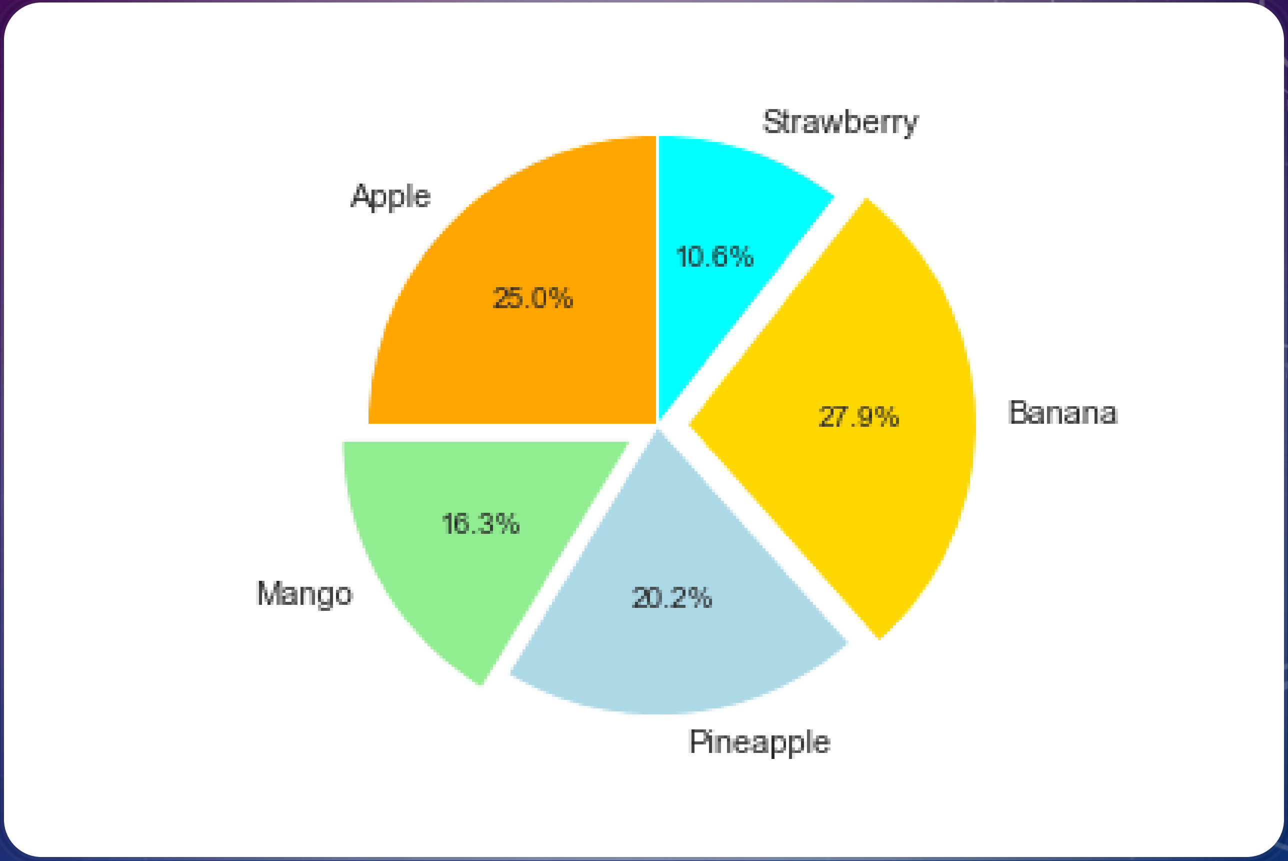
AUTOCORRELATION PLOTS

- Autocorrelation plots
- Autocorrelation plots graph autocorrelations of time series data for different lags. Autocorrelation is the correlation of a time series with the same time series lagged. The **autocorrelation_plot()** pandas function in **pandas.tools.plotting** can draw an autocorrelation plot



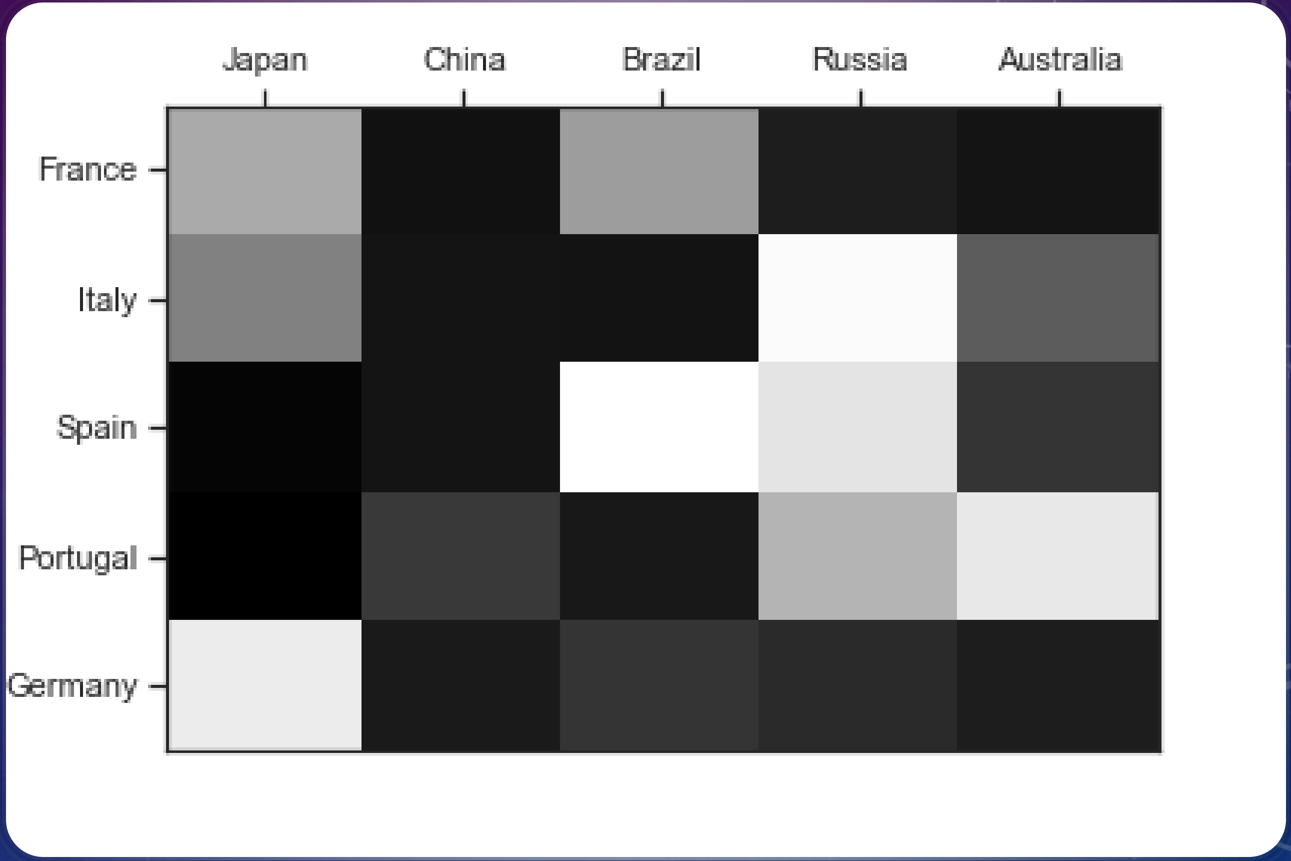
PIE CHARTS

- The circular diagram, often referred to as a pie chart, is a method used in descriptive statistics for graphical representations of quantitative variables measured on classes of categories (nominal values), in order to avoid establishing, even unintentionally, an order that does not exist in the categories.
- A pie chart is constructed by dividing a circle into slices whose angular amplitudes are proportional to the frequency classes. The areas identified by the slices are proportional to the frequencies. To make the diagram clearer, the various slices are filled with different colors.
- The pie chart is useful for displaying the market shares of products and brands, or the percentages taken by political parties in elections. The chart does not work with large percentage differences or with too many elements, as this would make the pie too jagged



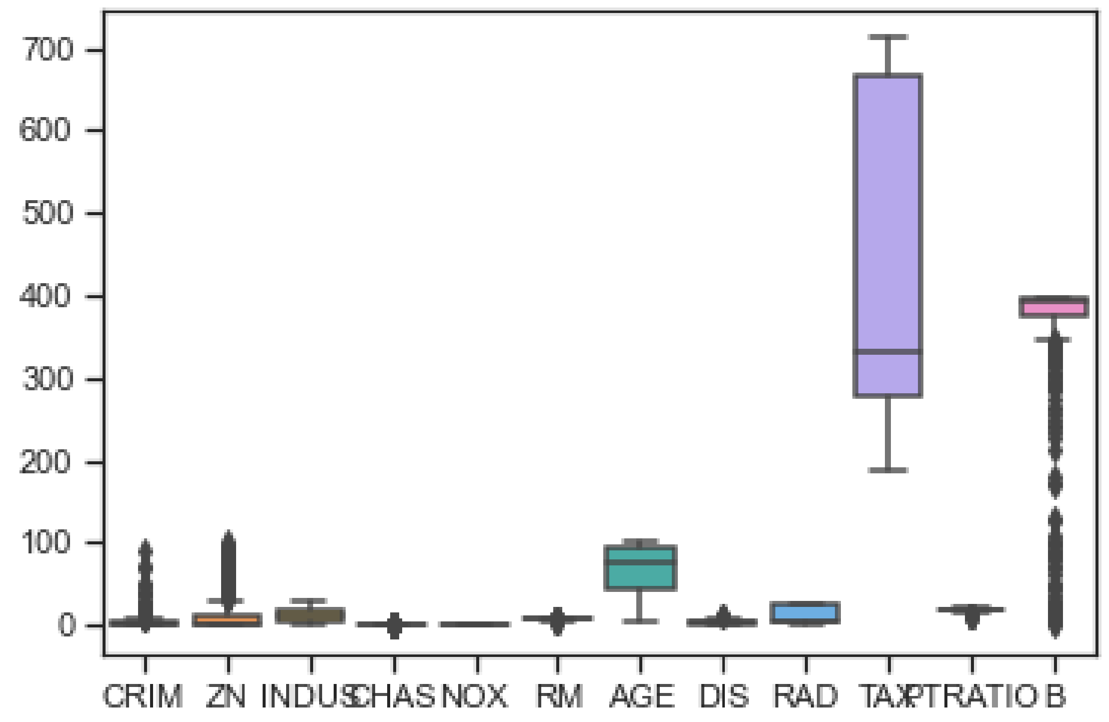
HEAT MAP

- The heat map is a graph where the individual values contained in a matrix are represented through gradations of colors. Both fractal maps and tree maps often use the same color-coding systems to represent the hierarchy of a variable. For example, if we measure the number of clicks on a web page or the areas where the mouse pointer passes the most often, we will obtain a heat map with certain areas highlighted by warm colors, that is, those that most attract our attention.
- We already stated that a heat map is a type of graphical representation of data, where the values are expressed through a different color range. Its representation usually includes the following:
    - Warm colors: Red, orange, and yellow, in the areas of greatest interest
    - Cold colors: Green or blue, for the areas with worse results
- Generally, heat maps require a large sample of data, and their main objective is to obtain useful data on the trend of a particular variable. This means of analysis allows us to evaluate the distribution of the variable of interest in the analyzed area. In this way, the concentration of warm colors in particular areas will highlight the areas in which the variable assumes the highest values.



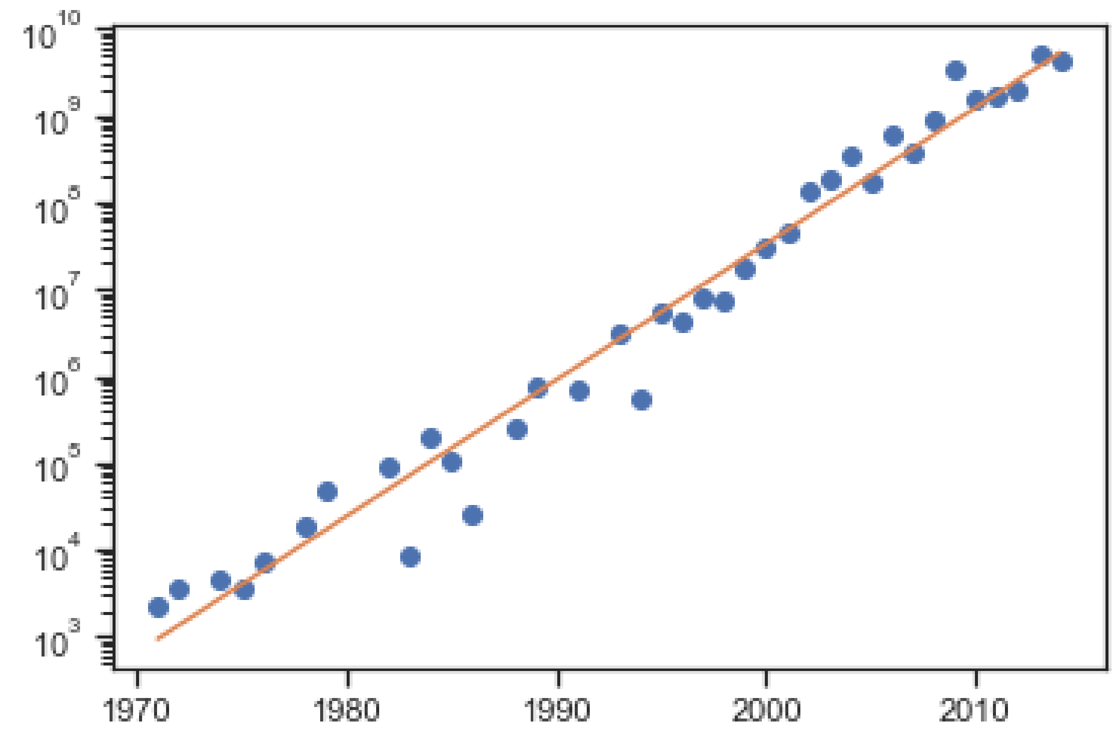
BOX PLOT

- A box plot, also called a whiskers chart, is a graph that uses simple dispersion and position indexes to describe the distribution of a sample. A box plot can be depicted either horizontally or vertically, by means of a rectangular partition divided by two segments. The rectangle (box) is delimited by the first quartile (the 25th percentile) and the third quartile (the 75th percentile), and is divided by the median (the 50th percentile).



LOGARITHMIC PLOT

- Logarithmic plots (or log plots) are plots that use a logarithmic scale. A logarithmic scale shows the value of a variable which uses intervals that match orders of magnitude, instead of a regular linear scale. There are two types of logarithmic plots. The log-log plot employs logarithmic scaling on both axes and is represented in matplotlib by the **matplotlib.pyplot.loglog()** function. The semi-log plots use linear scaling on one axis and logarithmic scaling on the other axis. These plots are represented in the matplotlib API by the **semilogx()** and **semilogy()** functions. On log-log plots, power laws appear as straight lines. On semi-log plots, straight lines represent exponential laws.



10^{10}
10^{9}
10^{8}
10^{7}
10^{6}
10^{5}
10^{4}
10^{3}
1970
1980
1990
2000
2010