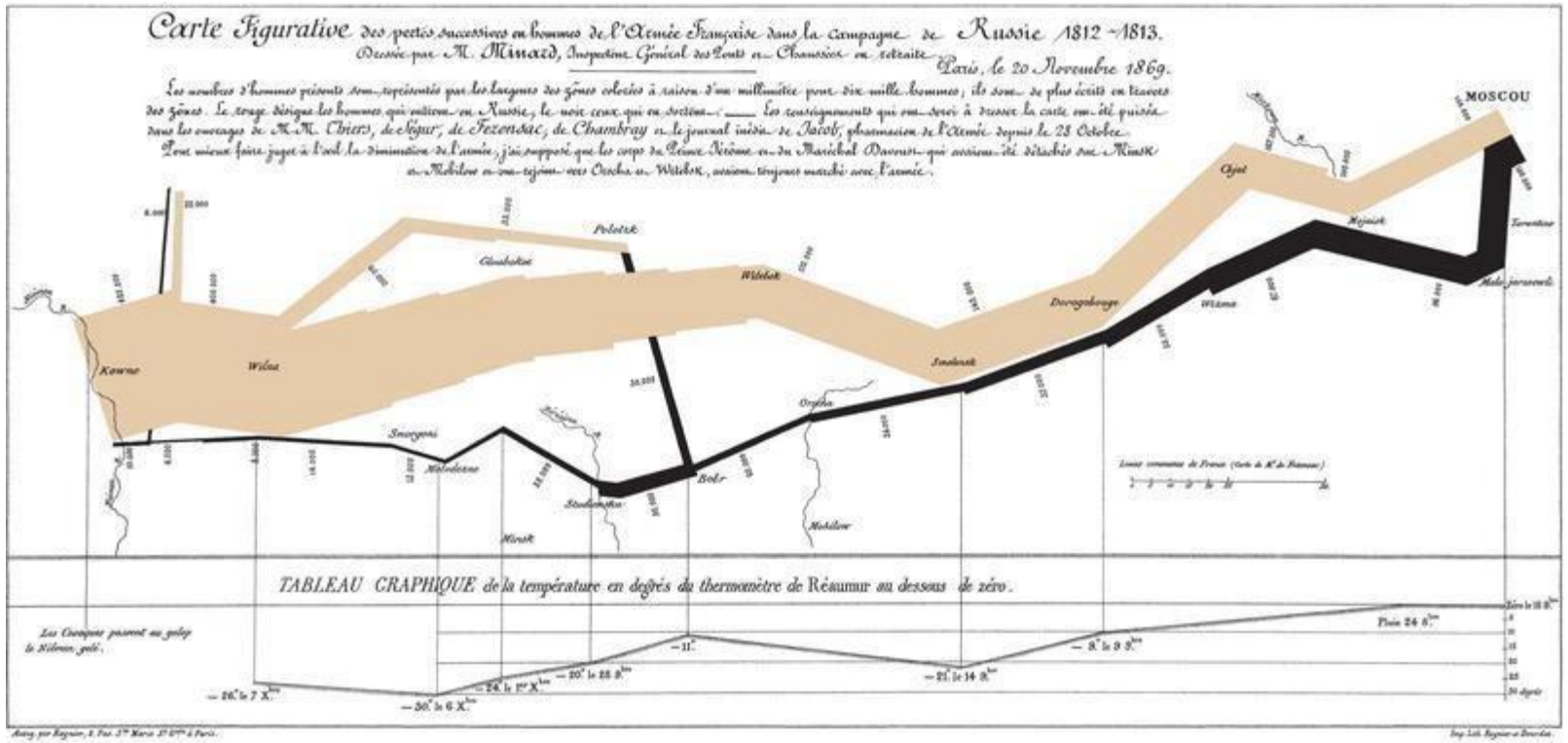
Week 3

Visualization with Python

Data Science Academy

The map of Napoleon's Russian campaign

Minard is best known for his cartographic depiction of numerical data on a map of Napoleon's disastrous losses suffered during the Russian campaign of 1812.



Data visualization

- EDA & Data Understanding
- Data Manipulation
- Visual Data Mining
- Models validation
- Analytics & reporting

Python Libraries for Data Science

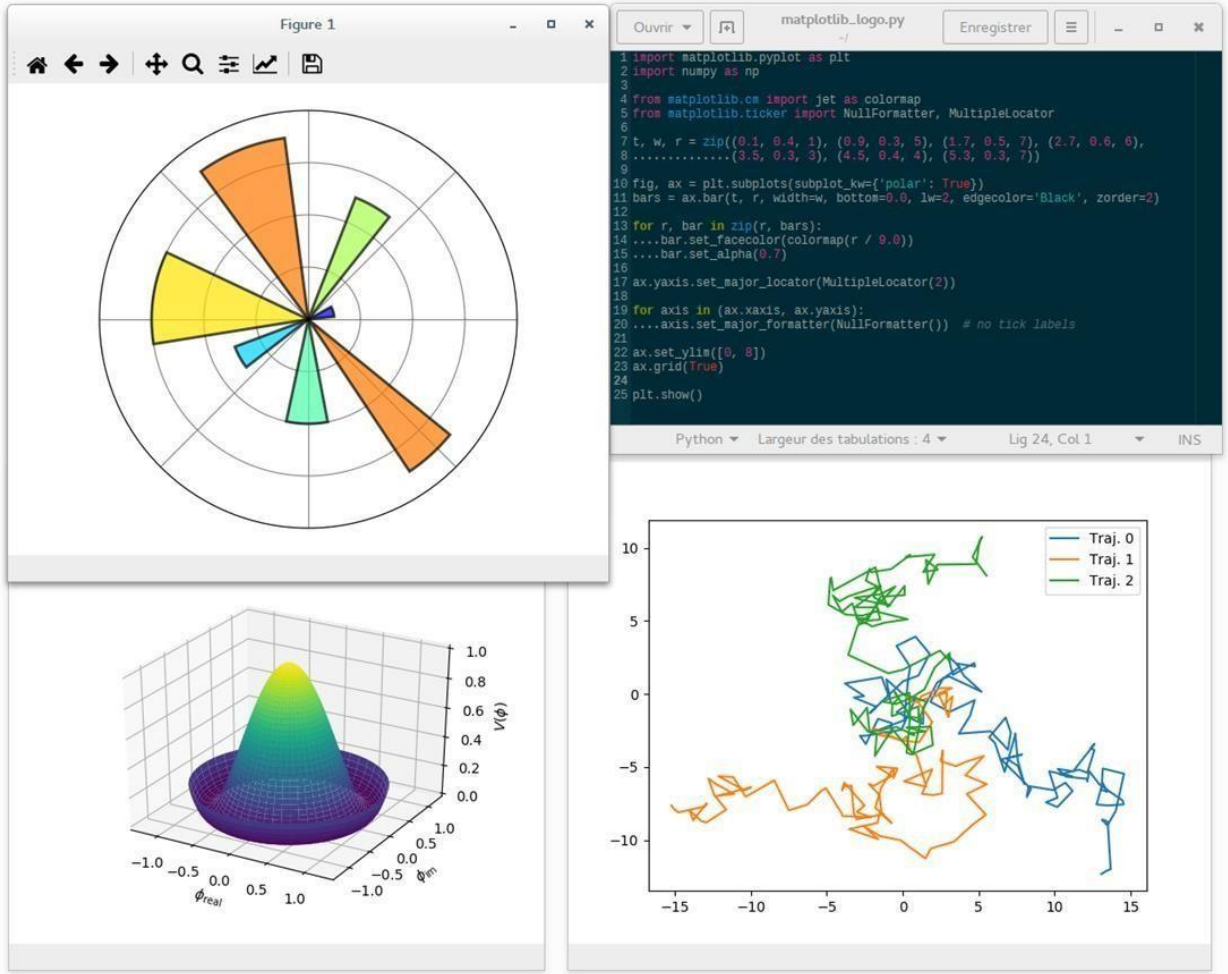
matplotlib:

- python 2D plotting library which produces publication quality figures in a variety of hardcopy formats
- a set of functionalities similar to those of MATLAB
- line plots, scatter plots, barcharts, histograms, pie charts etc.
- relatively low-level; some effort needed to create advanced visualization

Link: https://matplotlib.org/

Matplotlib

#pip install matplotlib

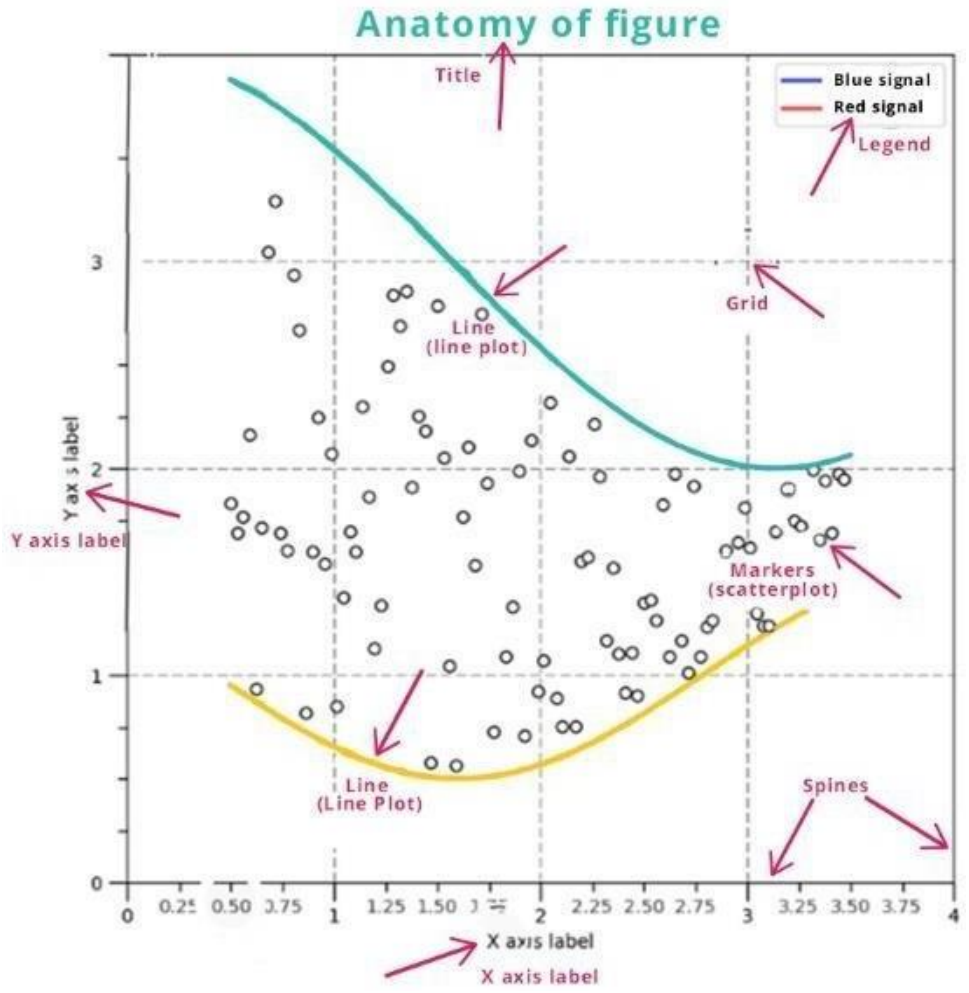


Basic visualization rules

1. Select plot type gives insight about case.
2. Label axis.
3. Add title.
4. Add labels for each categories.
5. Optionally text or arrow can be added at interesting data points.
6. Use sizes and colors of the data to make plot more appealing.

Parts of a Plot

- There are different parts of a plot, are denoted in the following figure:

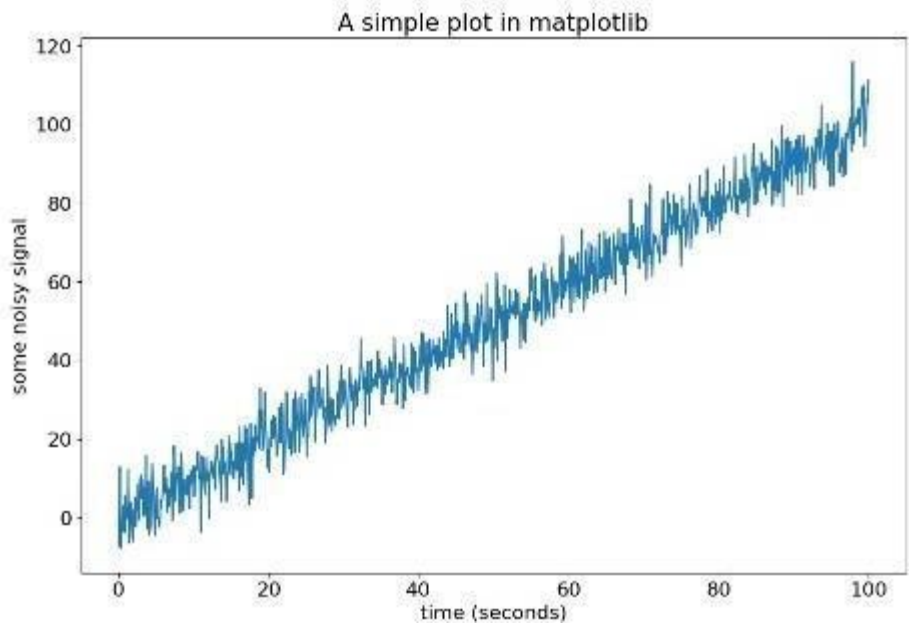


Structure for plotting

- plt.plot(x, y) *#plot x and y using default line style and color*
- plt.xlabel('comment') *#set a label that will be displayed in the legend*
- plt.ylabel('comment') *#set a label that will be displayed in the legend*
- plt.title("comment") *#available titles are positioned above the axes in the center*
- plt.legend() *#elements to be added to the legend are automatically determined*
- plt.show() *#display a figure*

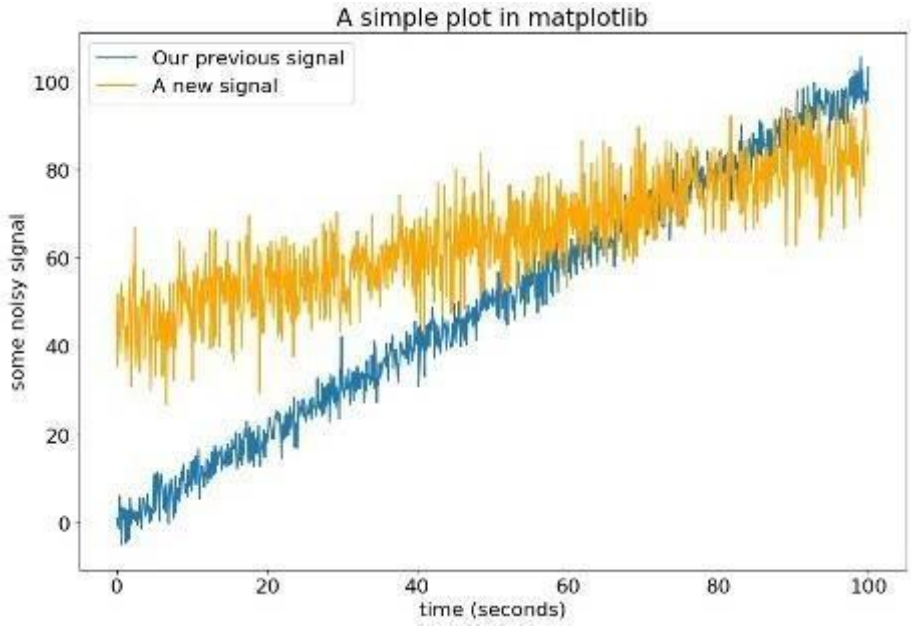
Example of Plot

- import numpy
- from matplotlib import pyplot
- x = numpy.linspace(0., 100., 1001)
- y = x + numpy.random.randn(1001) * 5
- pyplot.plot(x, y)
- pyplot.xlabel("time (seconds)")
- pyplot.ylabel("some noisy signal")
- pyplot.title("A simple plot in matplotlib")



Example of Plot

- import numpy
- from matplotlib import pyplot
- x = numpy.linspace(0., 100., 1001)
- y1 = x + numpy.random.randn(1001) * 3
- y2 = 45 + x * .4 + numpy.random.randn(1001) * 7
- pyplot.plot(x, y1, label='Our previous signal')
- pyplot.plot(x, y2, color='orange', label='A new signal')
- pyplot.xlabel('time (seconds)')
- pyplot.ylabel('some noisy signal')
- pyplot.title('A simple plot in matplotlib')
- pyplot.legend()

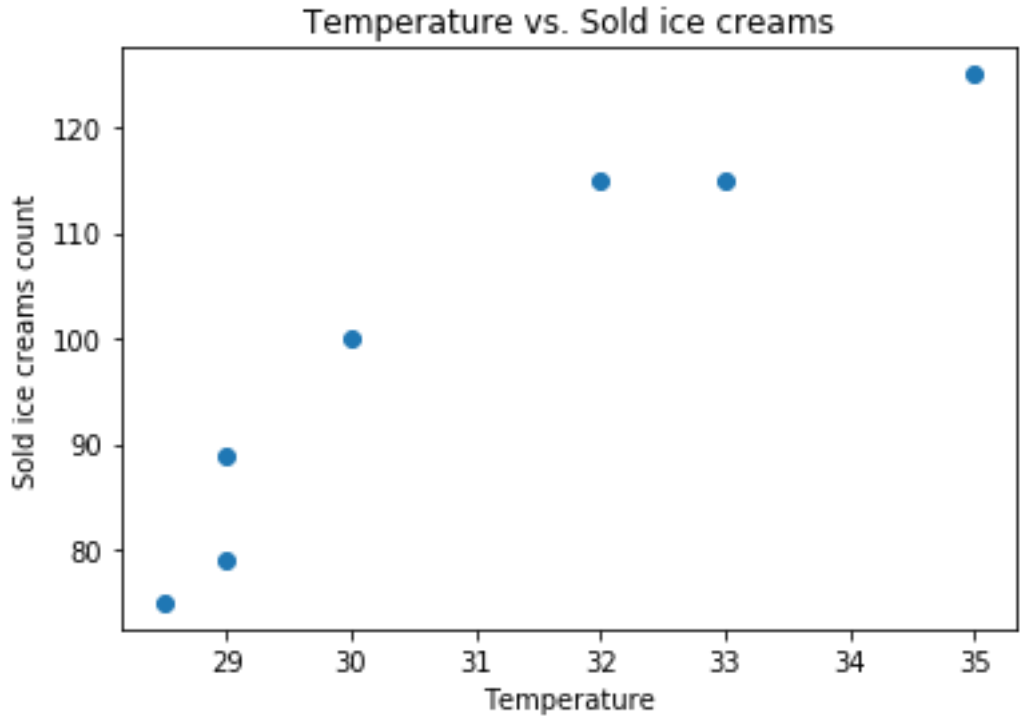


Scatter

- Scatter shows all individual data points, they aren't connected with lines.
- Each data point has the value of the x-axis value and the value from the y-axis values.
- Scatter can be used to display trends or correlations.
- In data science, it shows how 2 variables compare.
- To make a scatter plot with Matplotlib, we can use the *plt.scatter()* function.
- Again, the first argument is used for the data on the horizontal axis, and the second - for the vertical axis.

Example Scatter Plot

- import matplotlib.pyplot as plt
- temp = [30, 32, 33, 28.5, 35, 29, 29]
- ice_creams_count = [100, 115, 115, 75, 125, 79,89]
- plt.scatter(temp, ice_creams_count)
- plt.title("Temperature vs. Sold ice creams")
- plt.xlabel("Temperature")
- plt.ylabel("Sold ice creams count")
- plt.show()

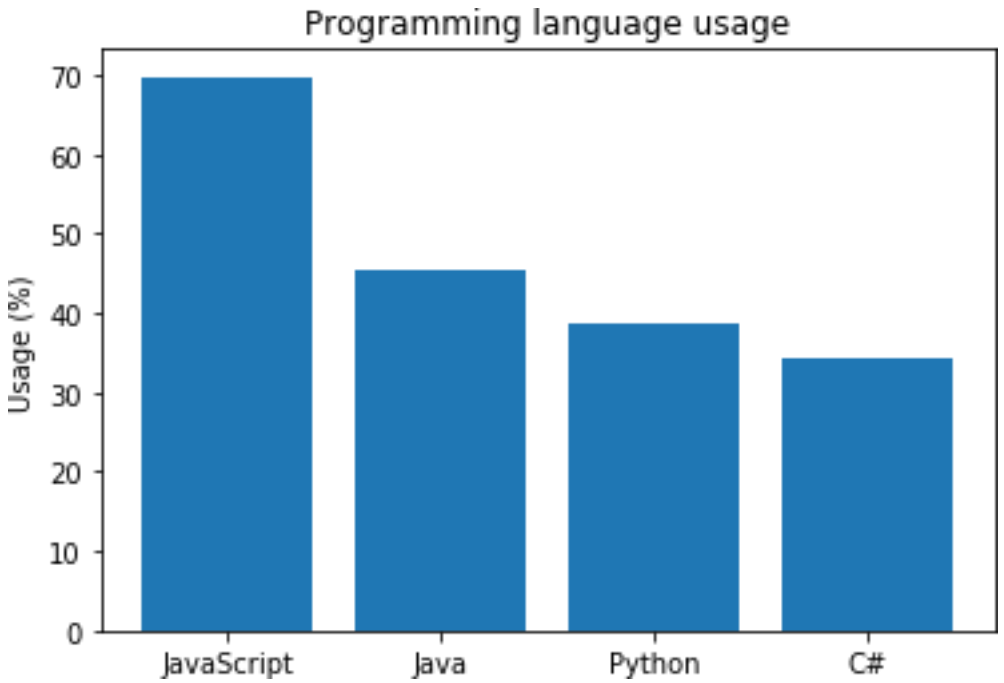


Bar chart

- Represents categorical data with rectangular bars.
- Each bar has a height corresponds to the value it represents.
- It's useful when we want to compare a given numeric value on different categories.
- It can also be used with 2 data series.
- To make a bar chart with Maplotlib, we'll need the *plt.bar()* function.

Example Bar chart

- import matplotlib.pyplot as plt
- labels = ["JavaScript", "Java", "Python", "C#"]
- usage = [69.8, 45.3, 38.8, 34.4]
- y_positions =range(len(labels))
- plt.bar(y_positions, usage)
- plt.xticks(y_positions,labels)
- plt.ylabel("Usage (%)")
- plt.title("Programming language usage")
- plt.show()

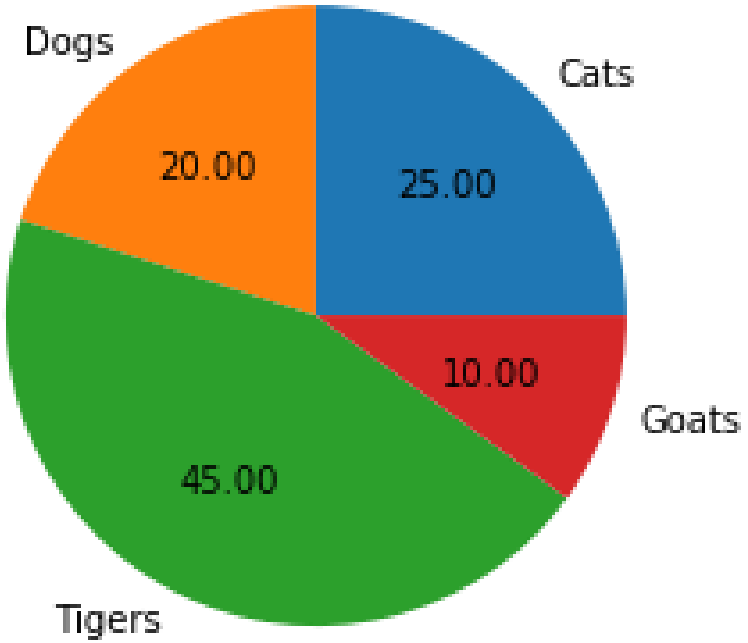


Pie chart

- Pie chart a circular plot, divided into slices to show numerical proportion.
- They are widely used in the business world. However, many experts recommend to avoid them.
- The main reason is that it's difficult to compare the sections of a given pie chart.
- Also, it's difficult to compare data across multiple pie charts.
- They can be replaced by a bar chart.

Example Pie chart

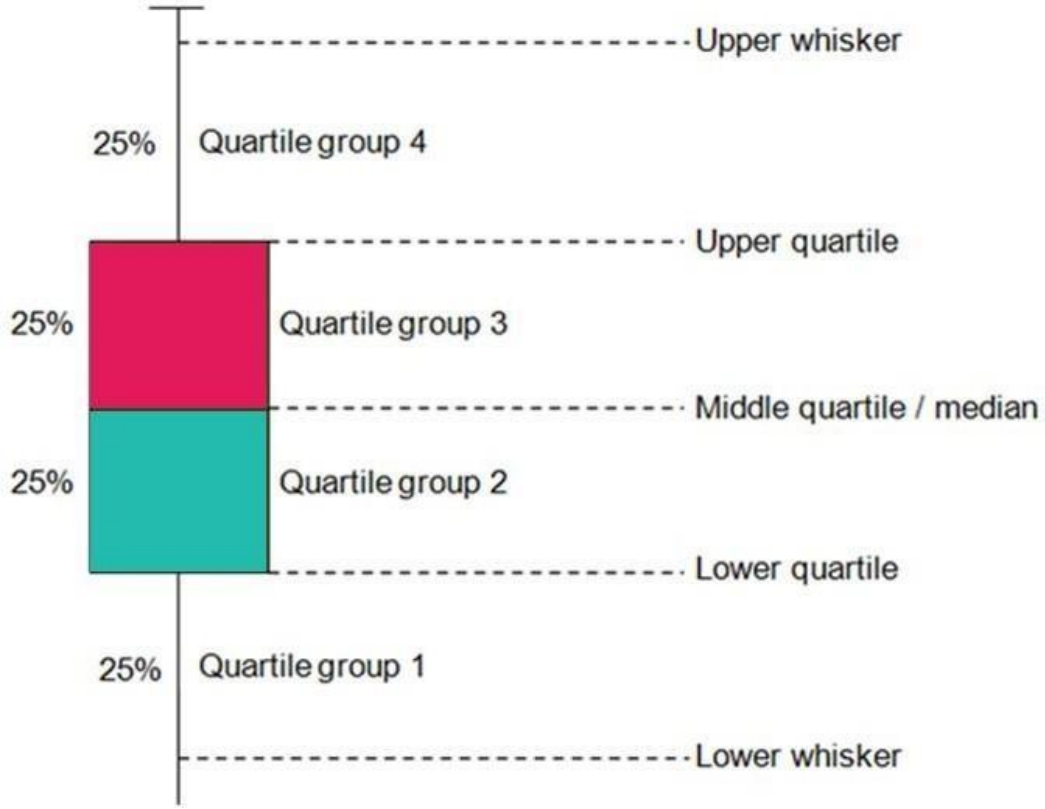
- import matplotlib.pyplot as plt
- sizes = [25, 20, 45, 10]
- labels = ["Cats", "Dogs", "Tigers", "Goats"]
- plt.pie(sizes, labels = labels, autopct = "%.2f")
- #float and persentage value
- plt.axes().set_aspect("equal") #auto #num #aspect ratio
- plt.show()



Box plot

- A **box plot** is a graphical rendition of statistical data based on the minimum, first quartile, median, third quartile, and maximum.

- The term "**box plot**" comes from the fact that the graph looks like a rectangle with lines extending from the top and bottom.

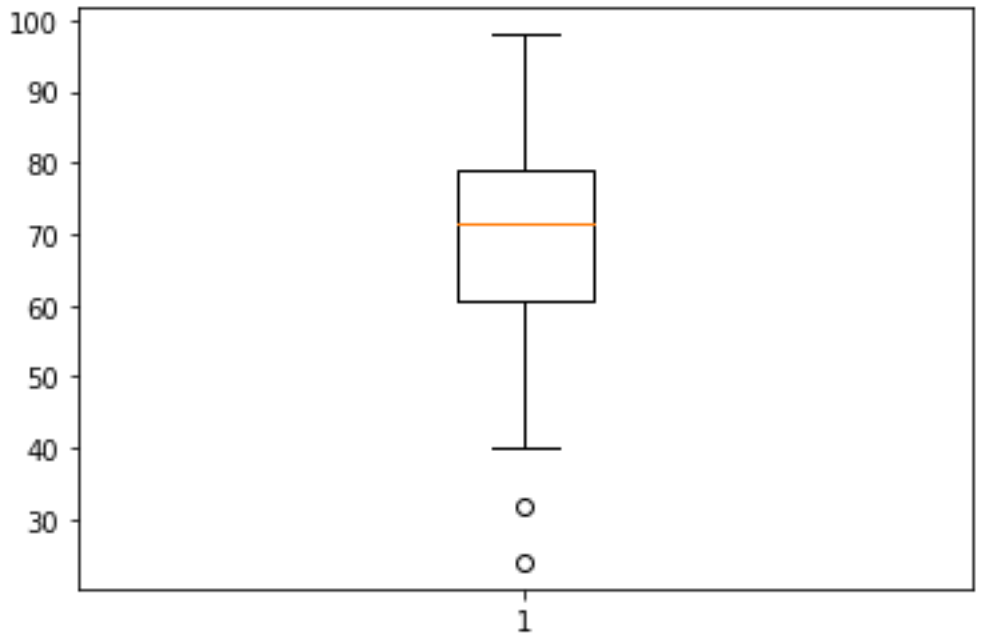


Example of Box plot

```
import matplotlib.pyplot as plt

value1 = [82,76,24,40,67,62,75,78,71,32,98,89,78,67,72,82,87,66,56,52]

plt.boxplot(value1)
plt.show()
```



Python Libraries for Data Science

Seaborn:

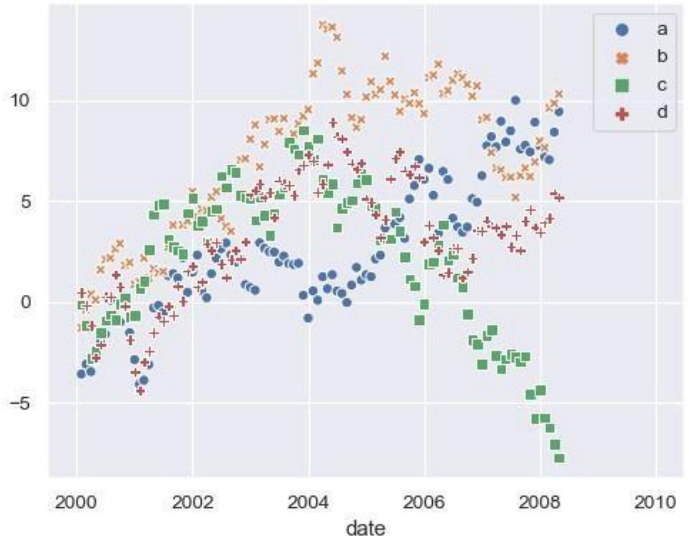
- Data Analysis and Visualization using Python Sept 2017 seaborn - harnesses the power of matplotlib to create beautiful charts in a few lines of code.
- based on matplotlib
- provides high level interface for drawing attractive statistical graphics
- Similar (in style) to the popular ggplot2 library in R

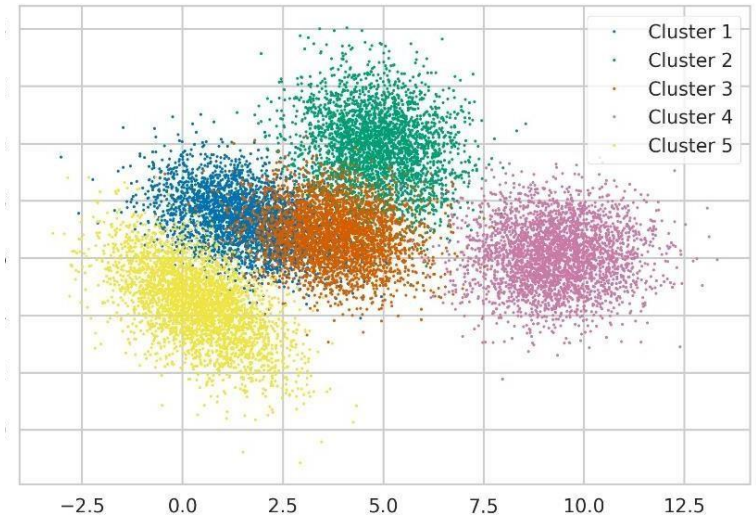
Link: https://seaborn.pydata.org/

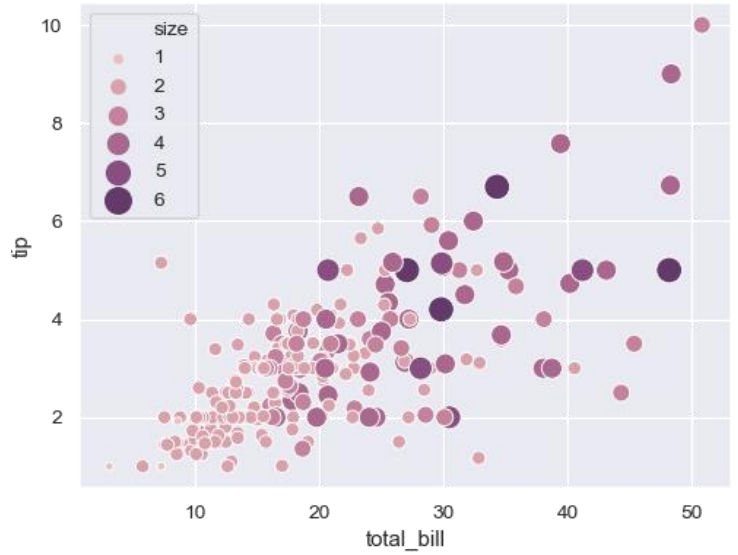
Seaborn plots

Graphics	Description
distplot	histogram
barplot	estimate of central tendency for a numeric variable
violinplot	similar to boxplot, also shows the probability density of the data
jointplot	Scatterplot
regplot	Regression plot
pairplot	Pairplot
boxplot	boxplot
swarmplot	categorical scatterplot
factorplot	General categorical plot

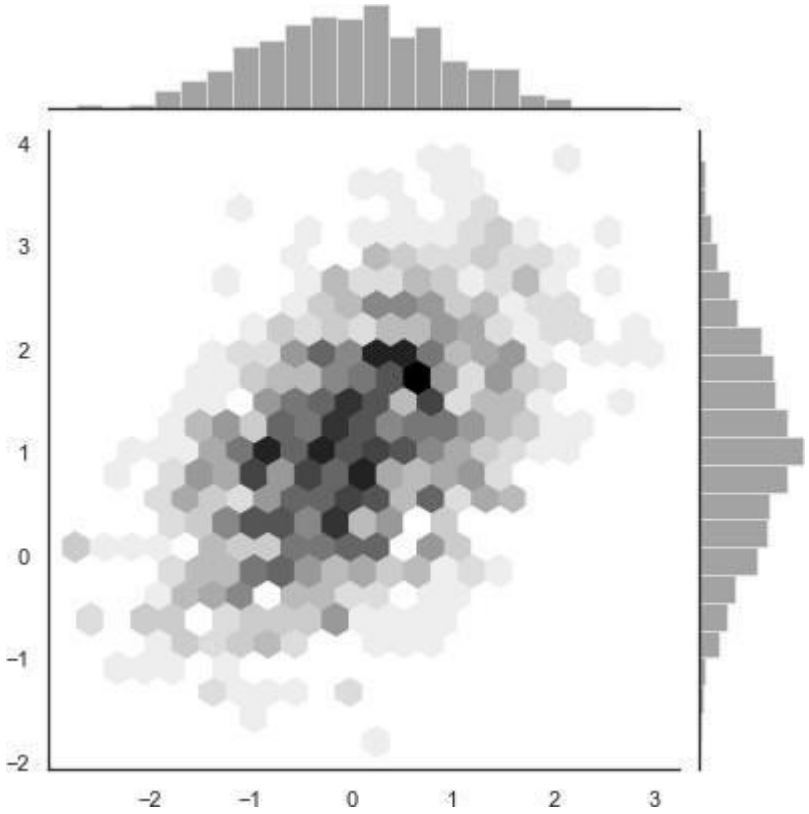
Example of Scatter Plot

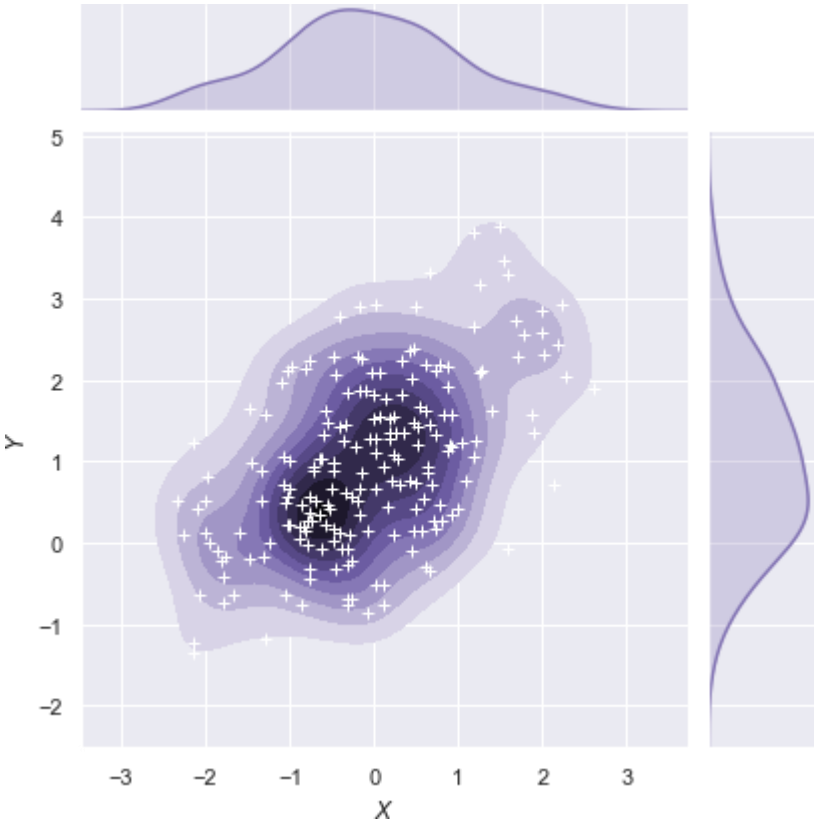


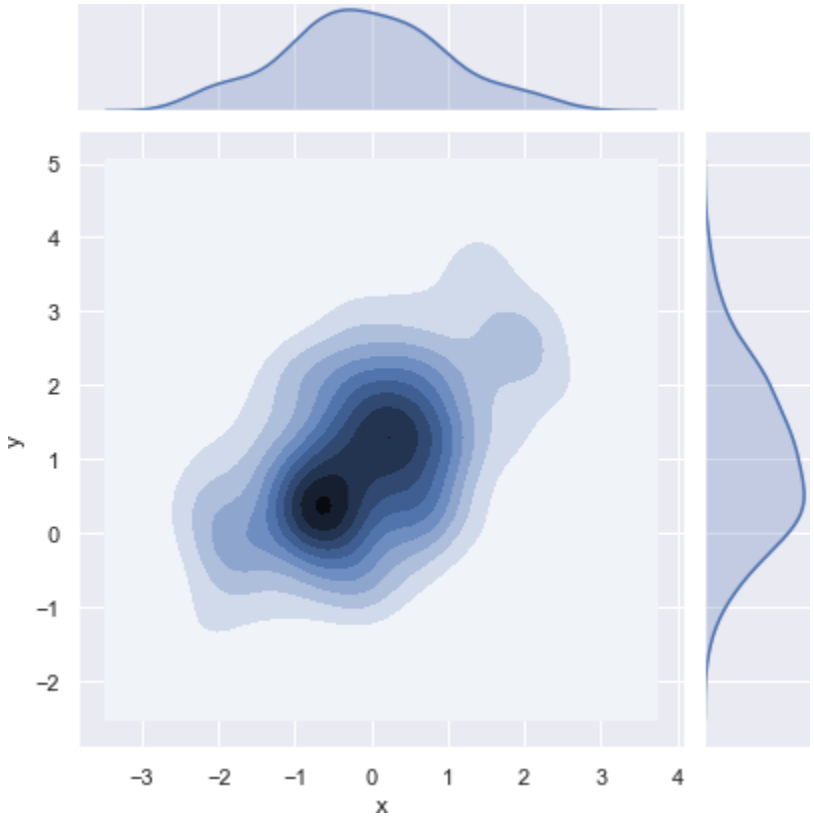




Example of Density Plot







Lambda functions

- A lambda function can take any number of arguments, but can only have one expression.

- It allows you to write quick throw away functions without naming them ,means we don't have to use functions such as **def.** and **return** the output

```
lambda arguments: expression
```

Arguments

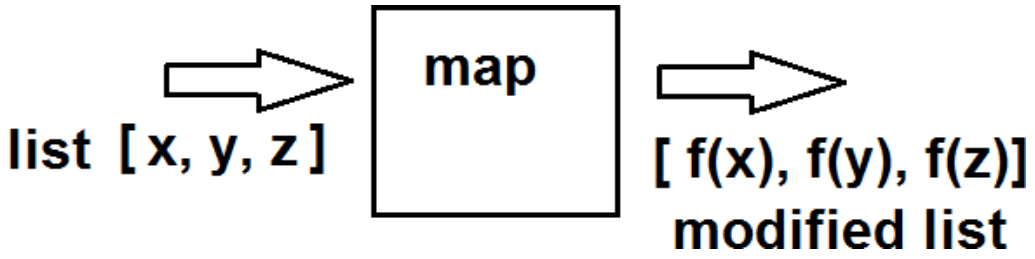
Contains a list of values passed to the function

Expression

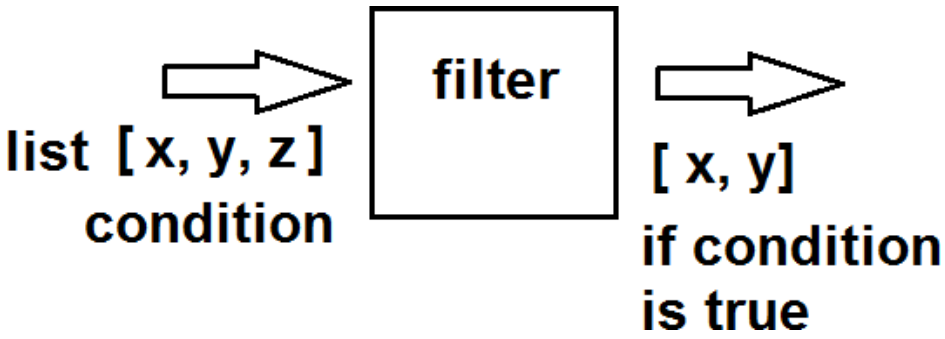
The expression is executed when the function is called

Functional operators: map and filter

- Map: *map* functions expect a function object and any number of iterables, such as list, dictionary, etc. It executes the function_object for each element in the sequence and returns a list of the elements modified by the function object.



- Filter: The filter function expects two arguments: function_object and an iterable. function_object returns a boolean value and is called for each element of the iterable. Filter returns only those elements for which the function_object returns true.



Functional operators: zip and reduce

- Zip() function returns a zip object, which is an iterator of tuples where the first item in each passed iterator is paired together, and then the second item in each passed iterator are paired together etc.

- Reduce() function is basically a function that performs the same operation to an items of a sequence, using the result of each operation as the first parameter of the next operation. A reduce function, in the end, returns an item, not a list, unlike map() and filter() functions.