Learning Outcome

Data analytics using Python

(15 Hours)

## 

# Multidimensional data handling using Pandas Library

## What is PANDAS

PANDAS (PANel Data) is a high level data manipulation tool used for analysis data. It is vary easy to import and export data using the Pandas library which has a very rich set of functions. It give us a single, convenient place to do most of our data analysis and visualization work.

Pandas have three important data structures, namely- Series, DataFrame, and Panel to make the process of analyzing data organized, effective and efficient.

* It is a package useful for data analysis and manipulation.
* Pandas provide an easy way to create, manipulate and wrangle the data.
* Pandas provide powerful and easy-to-use data structures, as well as the means to quickly perform operations on these structures.

Data scientists use Pandas for its following advantages:

* Easily handles missing data.
* It uses Series for one-dimensional data structure and DataFrame for multi-dimensional data structure.
* It provides an efficient way to slice the data.
* It provides a flexible way to merge, concatenate or reshape the data.



Image : What is PANDAS

Reference: <https://cdn.educba.com/academy/wp-content/uploads/2019/04/What-is-Pandas-1.jpg>

## Data Structure in Pandas

A data structure is a collection of data values and operations that can be applied to that data. It enables efficient storage, retrieval and modification to the data.

Pandas deals with 3 data structure

1. Series

2. Data Frame

3. Panel

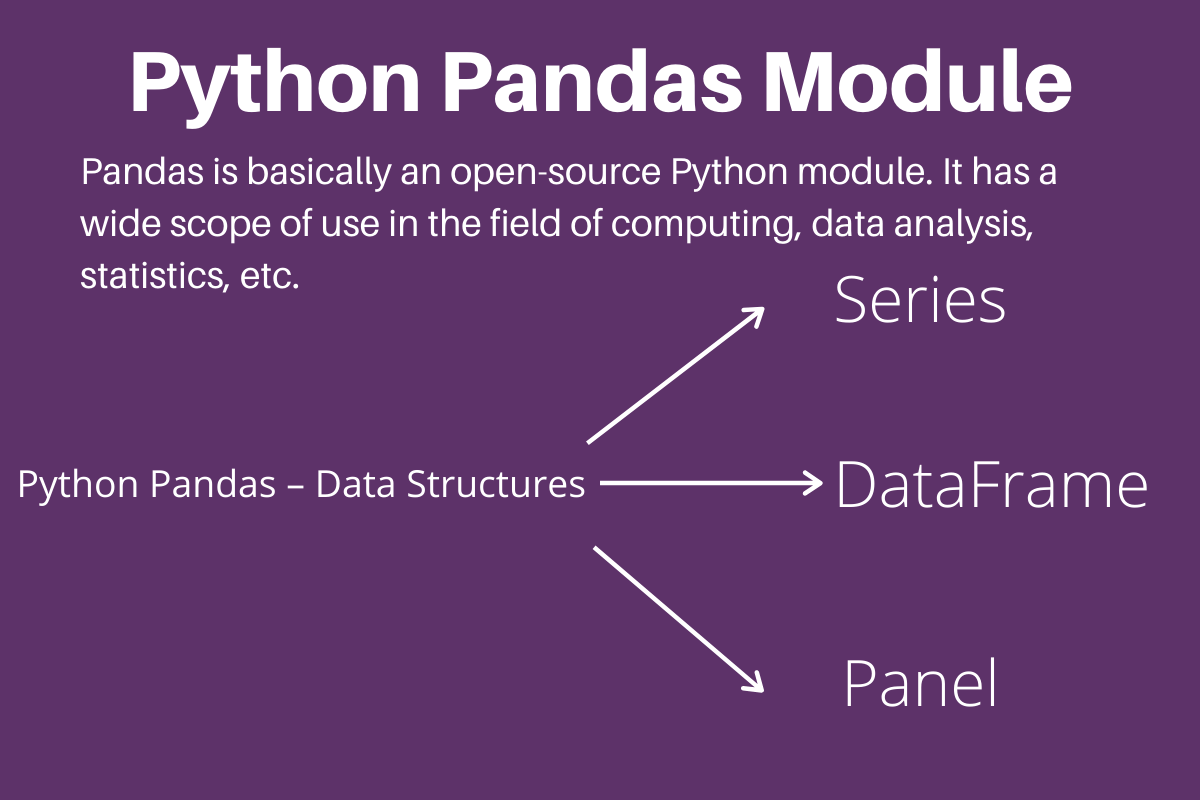


Image: Data Structure in PANDAS

Reference: : <https://www.askpython.com/wp-content/uploads/2020/02/Python-Pandas-Module.png>

## Series

Series is a one-dimensional array like structure with homogeneous data, which can be used to handle and manipulate data. What makes it special is its index attribute, which has incredible functionality and is heavily mutable.

It has two parts

1. Data part (An array of actual data)

2. Associated index with data (associated array of indexes or data labels)

* We can say that Series is a labeled one-dimensional array which can hold any type of data.
* Data of Series is always mutable, means it can be changed.
* But the size of Data of Series is always immutable, means it cannot be changed.
* Series may be considered as a Data Structure with two arrays out which one array works as Index (Labels) and the second array works as original Data.
* Row Labels in Series are called Index.

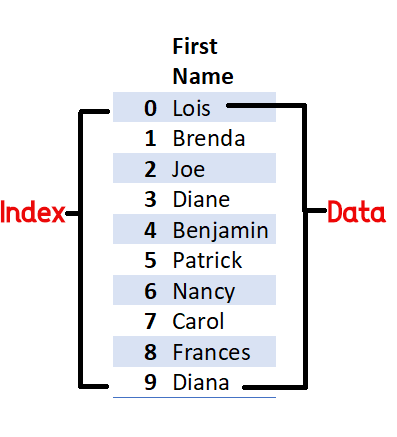


Image: Series

Reference: <https://1.bp.blogspot.com/-mYFbQb6dNWo/Xyq3MiVrWMI/AAAAAAAAsB8/ayRGqtckDEIyctuow7M65ezNtMIdqeH7ACPcBGAYYCw/s437/Pandas%2Bseries.png>

Example of a series containing names of students is given below:

Index Value

0 Arnab

1 Samridhi

2 Ramit

3 Divyam

4 Kritika

## Creation of Series

There are different ways in which a series can be created in Pandas. To create or use series, we first need to import the Pandas library.

### Creation of Series from Scalar Values

A Series can be created using scalar values as shown in

the example below:

>>> import pandas as pd #import Pandas with alias pd

>>> series1 = pd.Series([10,20,30]) #create a Series

>>> print(series1) #Display the series

Output:

0 10

1 20

2 30

dtype: int64

### Creation of Series from NumPy Arrays

We can create a series from a one-dimensional (1D) NumPy array, as shown below:

>>> import numpy as np # import NumPy with alias np

>>> import pandas as pd

>>> array1 = np.array([1,2,3,4])

>>> series3 = pd.Series(array1)

>>> print(series3)

Output:

0 1

1 2

2 3

3 4

dtype: int32

### Creation of Series from Dictionary

Recall that Python dictionary has key: value pairs and a value can be quickly retrieved when its key is known.

Dictionary keys can be used to construct an index for a Series, as shown in the following example. Here, keys of the dictionary dict1 become indices in the series.

>>> dict1 = {'India': 'NewDelhi', 'UK': 'London', 'Japan': 'Tokyo'}

>>> print(dict1) #Display the dictionary {'India': 'NewDelhi', 'UK': 'London', 'Japan': 'Tokyo'}

>>> series8 = pd.Series(dict1)

>>> print(series8) #Display the series

India NewDelhi

UK London

Japan Tokyo

dtype: object

## Accessing Elements of a Series

There are two common ways for accessing the elements of a series: Indexing and Slicing.

### Indexing

Indexing in Series is similar to that for NumPy arrays, and is used to access elements in a series. Indexes are of two types: positional index and labelled index. Positional index takes an integer value that corresponds to its position in the series starting from 0, whereas labelled index takes any user-defined label as index.Following example shows usage of the positional index for accessing a value from a Series.

>>> seriesNum = pd.Series([10,20,30])

>>> seriesNum[2]

30

Here, the value 30 is displayed for the positional index 2.

When labels are specified, we can use labels as indices while selecting values from a Series, as shown below. Here, the value 3 is displayed for the labelled index Mar.

>>> seriesMnths = pd.Series([2,3,4],index=["Feb ","Mar","Apr"])

>>> seriesMnths["Mar"]

3

### Slicing

Sometimes, we may need to extract a part of a series. This can be done through slicing. This is similar to slicing used with NumPy arrays. We can define which part of the series is to be sliced by specifying the start and end parameters [start :end] with the series name. When we use positional indices for slicing, the value at the endindex position is excluded, i.e., only (end - start) number of data values of the series are extracted.

Consider the following series seriesCapCntry:

>>> seriesCapCntry = pd.Series(['NewDelhi', 'WashingtonDC', 'London', 'Paris'], index=['India', 'USA', 'UK', 'France'])

>>> seriesCapCntry[1:3] #excludes the value at index position 3

USA WashingtonDC

UK London

dtype: object

# Data Visualization using Matplotlib

Data Visualization is an important part of business activities as organizations nowadays collect a huge amount of data. Sensors all over the world are collecting climate data, user data through clicks, car data for prediction of steering wheels etc. All of these data collected hold key insights for businesses and visualizations make these insights easy to interpret.

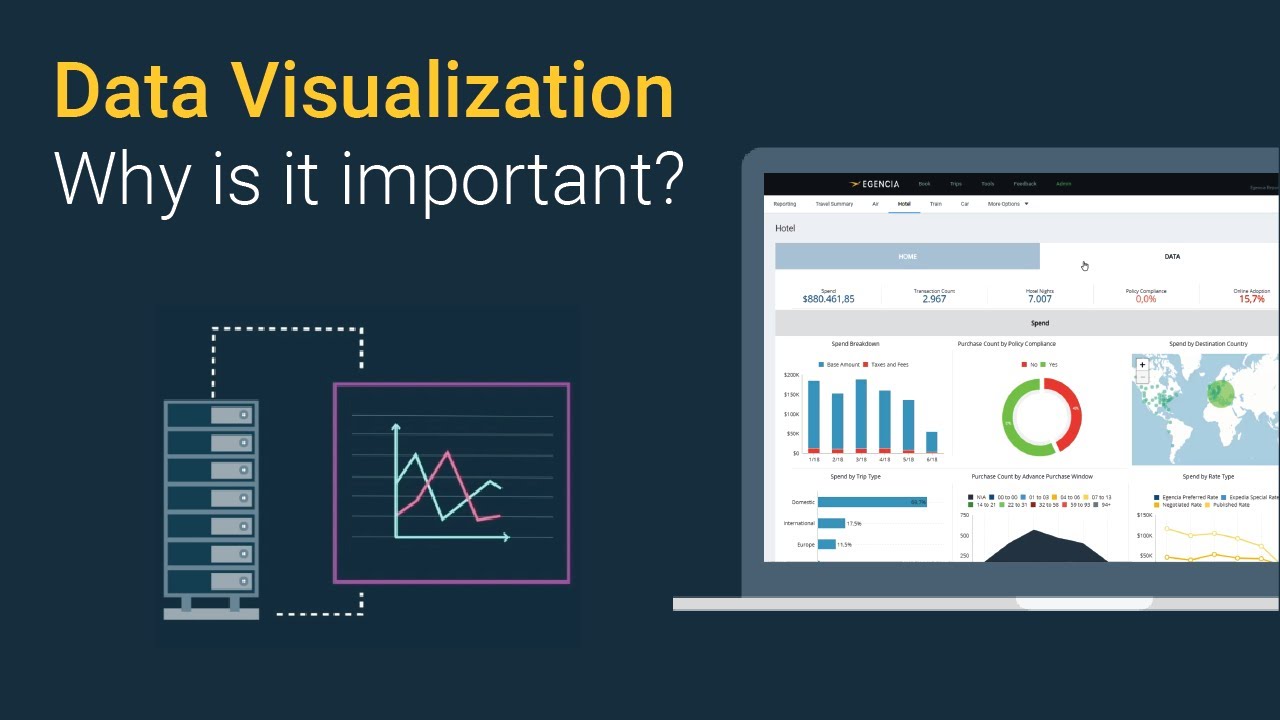


Image 1: Data visualization using Matplotlib

Reference: <https://i.ytimg.com/vi/VyhLRJVoIrI/maxresdefault.jpg>

## Why are visualizations important?

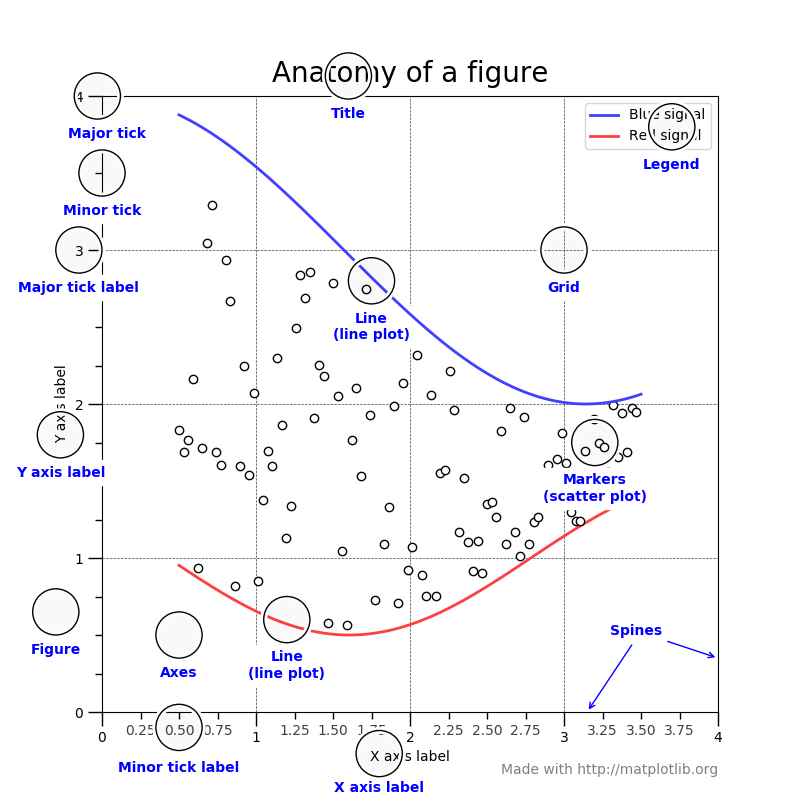
Visualizations are the easiest way to analyze and absorb information. Visuals help to easily understand the complex problem. They help in identifying patterns, relationships, and outliers in data. It helps in understanding business problems better and quickly. It helps to build a compelling story based on visuals. Insights gathered from the visuals help in building strategies for businesses. It is also a precursor to many high-level data analysis for Exploratory Data Analysis(EDA) and Machine Learning(ML).

Data visualizations in python can be done via many packages. We’ll be discussing of matplotlib package. It can be used in Python scripts, Jupyter notebook, and web application servers.

## Matplotlib

Matplotlib is a 2-D plotting library that helps in visualizing figures. Matplotlib emulates Matlab like graphs and visualizations. Matlab is not free, is difficult to scale and as a programming language is tedious. So, matplotlib in Python is used as it is a robust, free and easy library for data visualization.

### Anatomy of Matplotlib Figure



Image– Anatomy of Matplotlib figure

Reference: <https://matplotlib.org/3.1.1/_images/sphx_glr_anatomy_001.png>

The figure contains the overall window where plotting happens, contained within the figure are where actual graphs are plotted. Every Axes has an x-axis and y-axis for plotting. And contained within the axes are titles, ticks, labels associated with each axis. An essential figure of matplotlib is that we can more than axes in a figure which helps in building multiple plots, as shown below. In matplotlib, pyplot is used to create figures and change the characteristics of figures.

## Installing Matplotlib

Type !pip install matplotlib in the Jupyter Notebook or if it doesn’t work in cmd type conda install -c conda-forge matplotlib . This should work in most cases.

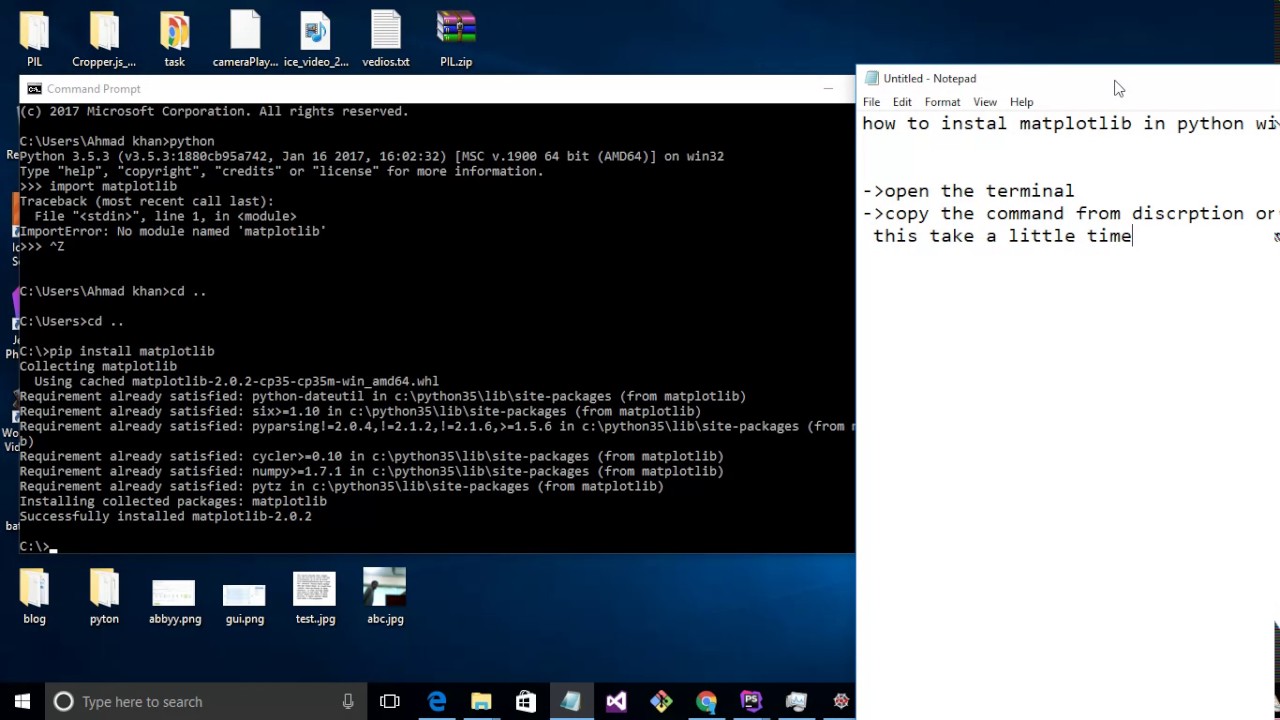


Image: Installing Matplotlib

Reference: <https://i.ytimg.com/vi/Iq9f2bQJOPg/maxresdefault.jpg>

## Things to follow

Plotting of Matplotlib is quite easy. Generally, while plotting they follow the same steps in each and every plot. Matplotlib has a module called pyplot which aids in plotting figure. The Jupyter notebook is used for running the plots. We import matplotlib.pyplot as plt for making it call the package module.

* Importing required libraries and dataset to plot using Pandas pd.read\_csv()
* Extracting important parts for plots using conditions on Pandas Dataframes.
* plt.plot()for plotting line chart similarly in place of plot other functions are used for plotting. All plotting functions require data and it is provided in the function through parameters.
* plot.xlabel , plt.ylabel for labeling x and y-axis respectively.
* plt.xticks , plt.yticks for labeling x and y-axis observation tick points respectively.
* plt.legend() for signifying the observation variables.
* plt.title() for setting the title of the plot.
* plot.show() for displaying the plot.

## Histogram

A histogram takes in a series of data and divides the data into a number of bins. It then plots the frequency data points in each bin (i.e. the interval of points). It is useful in understanding the count of data ranges.

When to use: We should use histogram when we need the count of the variable in a plot.

eg: Number of particular games sold in a store.

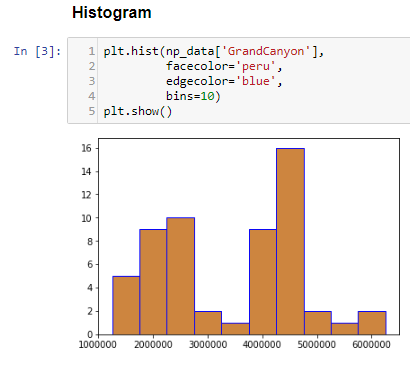


Image : Histogram

Reference: [https://miro.medium.com/max/820/1\*\_1qvF8jD\_1amVM0SMJ3cjA.png](https://miro.medium.com/max/820/1*_1qvF8jD_1amVM0SMJ3cjA.png)

From above we can see the histogram for GrandCanyon visitors in years. plt.hist() takes the first argument as numeric data in the horizontal axis i.e GrandCanyon visitor.bins=10 is used to create 10 bins between values of visitors in GrandCanyon.

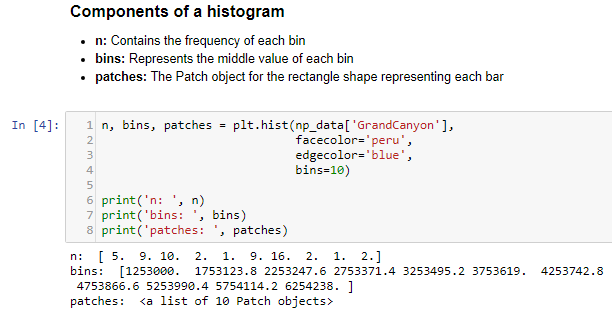


Image : Histogram

Reference: [https://miro.medium.com/max/1224/1\*5a17l7yG4VLKjGLVouyycw.png](https://miro.medium.com/max/1224/1*5a17l7yG4VLKjGLVouyycw.png)

From above, we can see the components that make a histogram, n as the max values in each bin of histogram i.e 5,9, and so on.

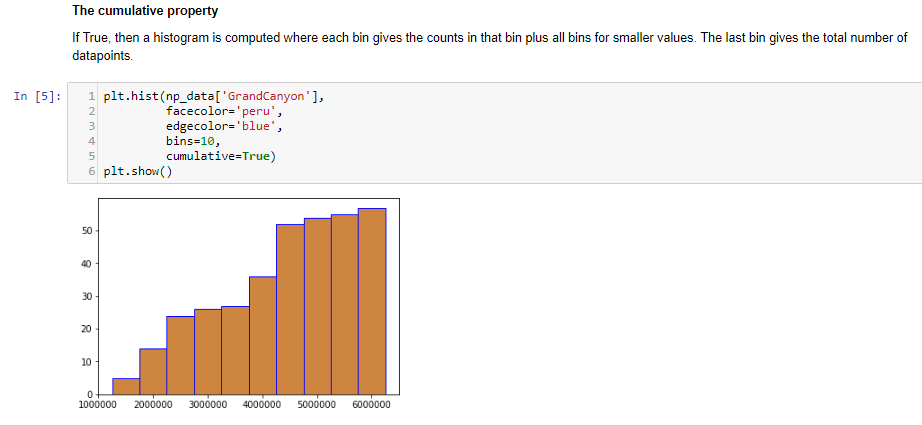


Image: Histogram

Reference: [https://miro.medium.com/max/1400/1\*TVUmVjh\_BVjQe2wRwwnOzw.png](https://miro.medium.com/max/1400/1*TVUmVjh_BVjQe2wRwwnOzw.png)

The cumulative property gives us the end added value and helps us understand the increase in value at each bin.

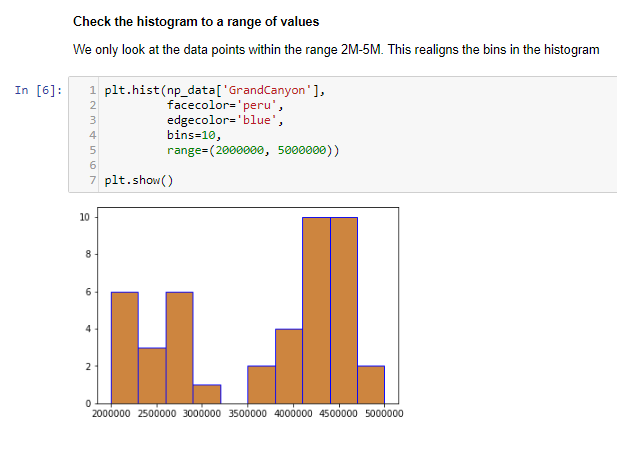


Image: Histogram

Reference: [https://miro.medium.com/max/1234/1\*RryYV9wgjBdscZR\_8OhRdg.png](https://miro.medium.com/max/1234/1*RryYV9wgjBdscZR_8OhRdg.png)

Range helps us in understanding value distribution between specified values.

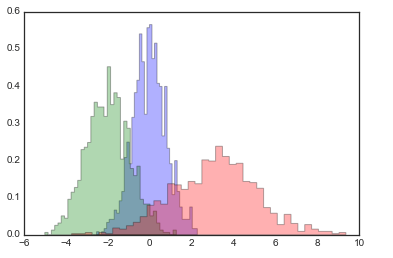


Image: Histogram

Reference: <https://i.stack.imgur.com/Ji8I6.png>

Multiple histograms are useful in understanding the distribution between 2 entity variables. We can see that GrandCanyon has comparably more visitors than BryceCanyon.

## Implementation: Histogram

### Pie Chart

It is a circular plot which is divided into slices to illustrate numerical proportion. The slice of a pie chart is to show the proportion of parts out of a whole.

When to use: Pie chart should be used seldom used as It is difficult to compare sections of the chart. Bar plot is used instead as comparing sections is easy.

eg: Market share in Films.

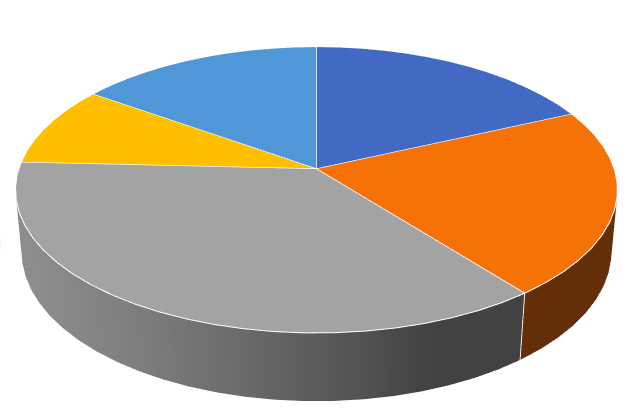


Image: Histogram

Reference: [https://miro.medium.com/max/629/1\*rzNBRBt1RCr4NsN4Lb7V1Q.png](https://miro.medium.com/max/629/1*rzNBRBt1RCr4NsN4Lb7V1Q.png)

Above, plt.pie() takes the numeric data as 1st argument i.e Percentage and labels to display as second argument i.e Sector. Ultimately, it shows the distribution of data in proportion to the pie.

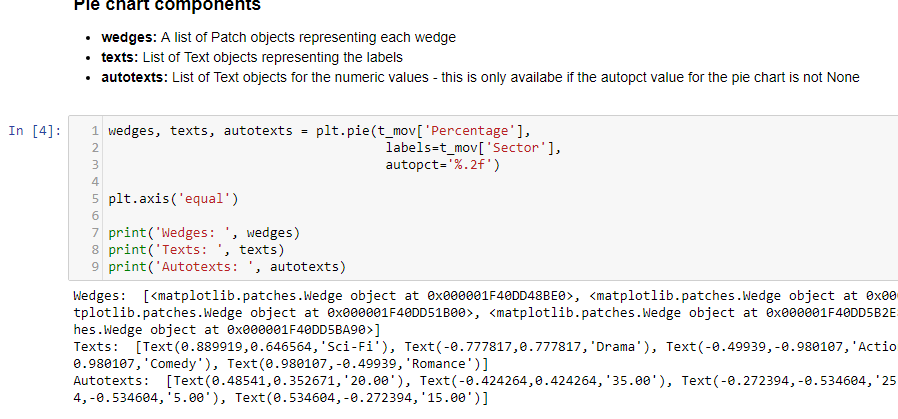


Image: Histogram

Reference: [https://miro.medium.com/max/1400/1\*yrThnLG3Ub5By5IwzaC1yQ.png](https://miro.medium.com/max/1400/1*yrThnLG3Ub5By5IwzaC1yQ.png)

From above we can the components that make a pie chart and it returns wedge object, text in labels and so on.

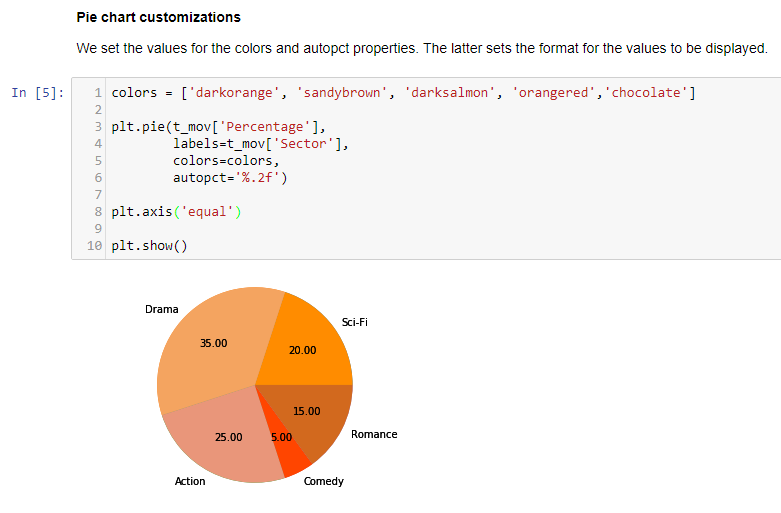


Image: Histogram

Reference: [https://miro.medium.com/max/1400/1\*vR0jlrqSeVkBf8gmHi79eA.png](https://miro.medium.com/max/1400/1*vR0jlrqSeVkBf8gmHi79eA.png)

A pie chart can be easily customized and from above color and label values are formatted.

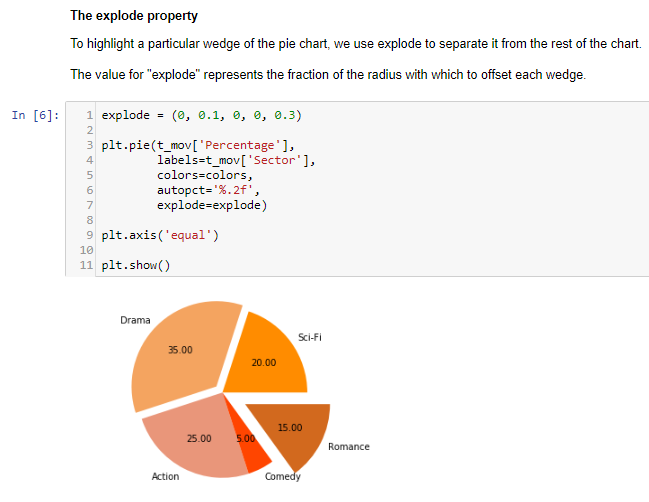


Image: Histogram

Reference: [https://miro.medium.com/max/1298/1\*5qWJfIY-cBCBgZlTzerXsA.png](https://miro.medium.com/max/1298/1*5qWJfIY-cBCBgZlTzerXsA.png)

From above explode is used to separate out points from the pie. Similar to a pizza piece being cut.

## Time Series by line plot

Time series is a line plot and it is basically connecting data points with a straight line. It is useful in understanding the trend over time. It can explain the correlation between points by the trend. An upward trend means positive correlation and downward trend means a negative correlation. It mostly used in forecasting, monitoring models.

### When to use: Time Series should be used when single or multiple variables are to be plotted over time.

eg: Stock Market Analysis of Companies, Weather Forecasting.

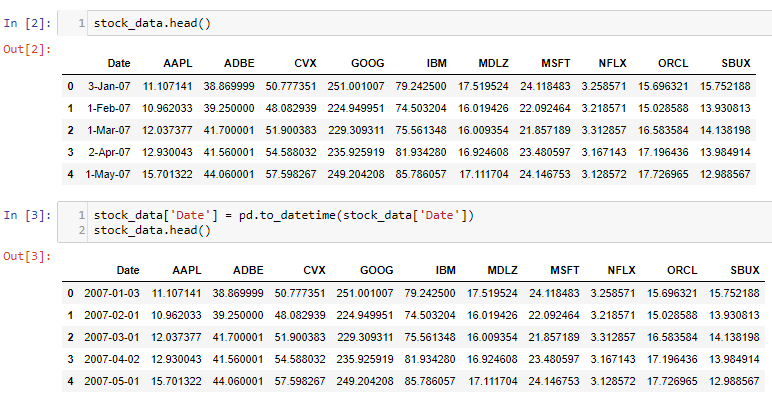


Image: Time Series by line plot

Reference: [https://miro.medium.com/max/1400/1\*3HWpcekG8MUwpQAtmW1Oow.png](https://miro.medium.com/max/1400/1*3HWpcekG8MUwpQAtmW1Oow.png)

First, Convert Date to pandas DateTime for easier plotting of data.



Image: Time Series by line plot

Reference: [https://miro.medium.com/max/822/1\*X3q-\_Cl\_eMuncoVG-DoQCQ.png](https://miro.medium.com/max/822/1*X3q-_Cl_eMuncoVG-DoQCQ.png)

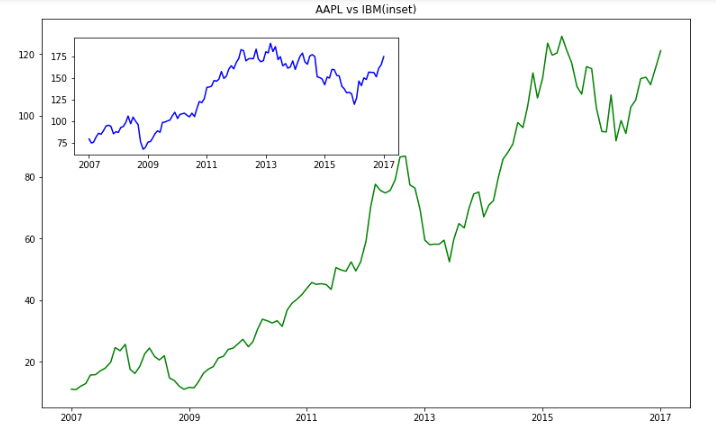


Image: Time Series by line plot

Reference: [https://miro.medium.com/max/1400/1\*aYgw-zGwdRn\_RE3h5KcIHQ.png](https://miro.medium.com/max/1400/1*aYgw-zGwdRn_RE3h5KcIHQ.png)

From above, fig.add\_axes is used for plotting the canvas. Check this What are the differences between add\_axes and add\_subplot? to understand axes and subplots. plt.plot() takes the 1st argument as numeric data i.e Date and 2nd argument is to numeric stock data. AAPL Stock is considered as ax1 which is the outer figure and on ax2 IBM Stock is considered for plotting which is inset.

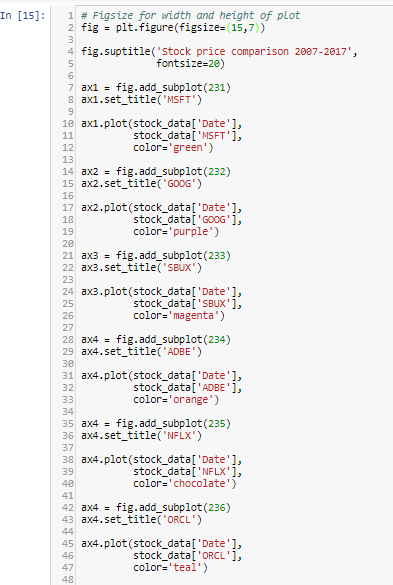


Image: Time Series by line plot

Reference: [https://miro.medium.com/max/786/1\*klsAZMknOQT6B2yd-My-xA.png](https://miro.medium.com/max/786/1*klsAZMknOQT6B2yd-My-xA.png)

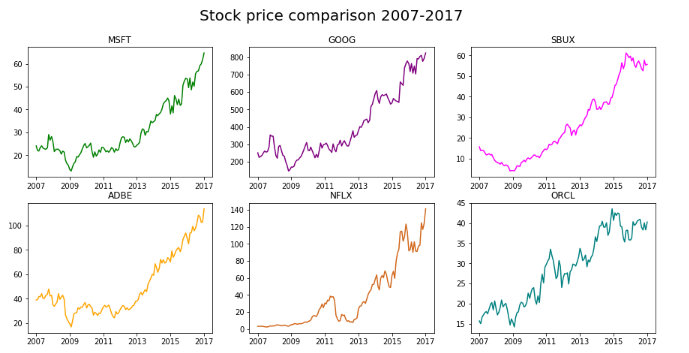


Image: Time Series by line plot

Reference: [https://miro.medium.com/max/1378/1\*26SwjgcLr\_j4\_xY6EJ2Jfw.png](https://miro.medium.com/max/1378/1*26SwjgcLr_j4_xY6EJ2Jfw.png)

In the earlier figure,add\_axes is used to used to add an axes to a figure whereas from above add\_subplot adds multiple subplots to a figure. fig.add\_subplot(237) cannot be done as there are only 6 subplots possible.

We can see that the tech company stocks are following an upward trend showing positive results for traders to invest in stocks.

## Boxplot and Violinplot

### Boxplot

Boxplot gives a nice summary of the data. It helps in understanding our distribution better.

#### When to use: It should be used when we require to use the overall statistical information on the distribution of the data. It can be used to detect outliers in the data.

eg: Credit Score of Customer. We can get the max, min and much more information about the mark.

## Understanding Boxplot

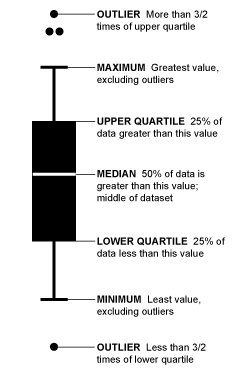


Image: Boxplot

Reference: [https://miro.medium.com/max/482/1\*fCE\_5juz235c6cmaOP\_PDQ.png](https://miro.medium.com/max/482/1*fCE_5juz235c6cmaOP_PDQ.png)

From the above diagram, the line that divides the box into 2 parts represents the median of the data. The end of the box shows the upper quartile(75%)and the start of the box represents the lower quartile(25%). Upper Quartile is also called 3rd quartile and similarly, Lower Quartile is also called as 1st quartile. The region between lower quartile and the upper quartile is called as Inter Quartile Range(IQR) and it is used to approximate the 50% spread in the middle data(75–25=50%). The maximum is the highest value in data, similarly minimum is the lowest value in data, it is also called as caps. The points outside the boxes and between the maximum and maximum are called as whiskers, they show the range of values in data. The extreme points are outliers to the data. A commonly used rule is that a value is an outlier if it’s less than lower quartile-1.5 \* IQR or high than the upper quartile + 1.5\* IQR.

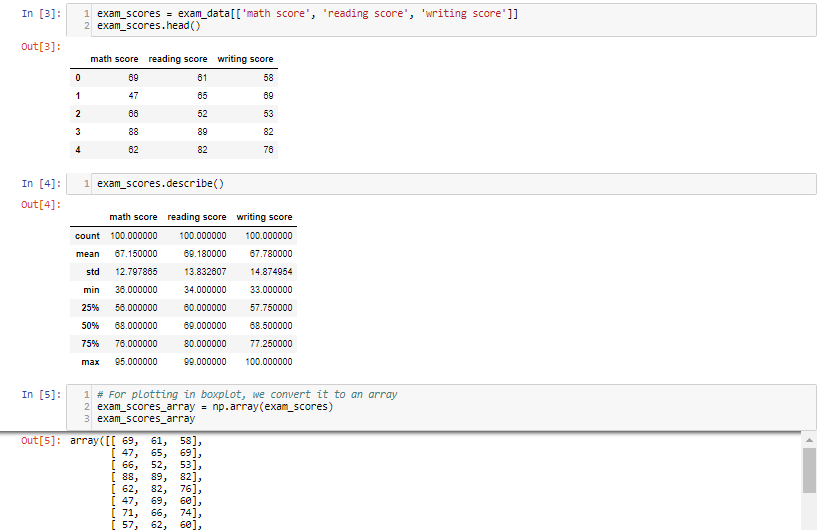


Image: Boxplot

Reference: [https://miro.medium.com/max/1400/1\*ad0xiuRs1X-CvlO-XSCKoA.png](https://miro.medium.com/max/1400/1*ad0xiuRs1X-CvlO-XSCKoA.png)

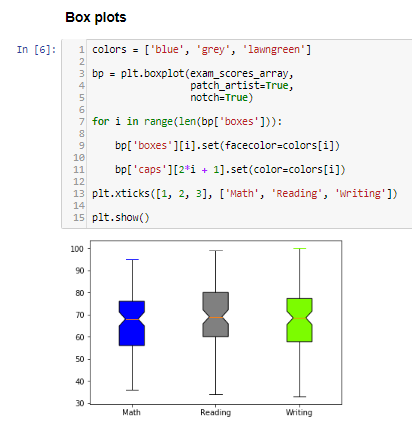


Image: Boxplot

Reference: [https://miro.medium.com/max/824/1\*g12e9wIZOtKe8IHWGDNVBw.png](https://miro.medium.com/max/824/1*g12e9wIZOtKe8IHWGDNVBw.png)

bp contains the boxplot components like boxes, whiskers, medians, caps. Seaborn another plotting library makes it easier to build custom plots than matplotlib. patch\_artist makes the customization possible. notchmakes the median look more prominent.

A caveat of using boxplot is the number of observations in the unique value is not defined, Jitter Plot in Seaborn can overcome this caveat or Violinplot is also useful

## Violin plot

Violin plot is a better chart than boxplot as it gives a much broader understanding of the distribution. It resembles a violin and dense areas point the more distribution of data otherwise hidden by box plots

### When to use: Its an extension to boxplot. It should be used when we require a better intuitive understanding of data.

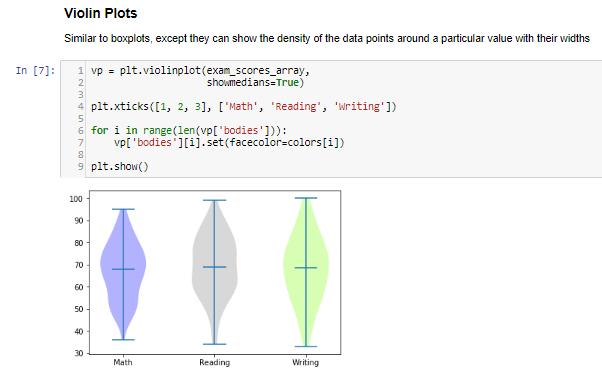


Image 1:

Reference: [https://miro.medium.com/max/1204/1\*J9OnuX8f5BjlB3XZiyHkVA.png](https://miro.medium.com/max/1204/1*J9OnuX8f5BjlB3XZiyHkVA.png)

The density of points in the middle seems more as students tend to score around average mostly in the subjects.

## TwinAxis

TwinAxis helps in visualizing plotting 2 plots w.r.t to the y-axis and same x-axis.

### When to use: It should when we require 2 plots or grouped data in the same direction.

Eg: Population, GDP data in the same x-axis (Date).

### Plotting 2 Plots w.r.t the y-axis and same x-axis

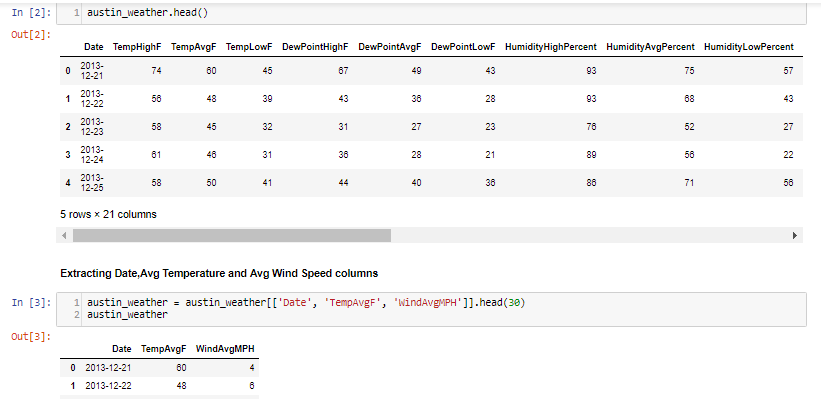


Image : TwinAxis

Reference: [https://miro.medium.com/max/1400/1\*dHwnU6ySte5aqNZYezQAgQ.png](https://miro.medium.com/max/1400/1*dHwnU6ySte5aqNZYezQAgQ.png)

Extracting important details i.e Date for the x-axis, TempAvgF, and WindAvgMPH for the different y-axis.

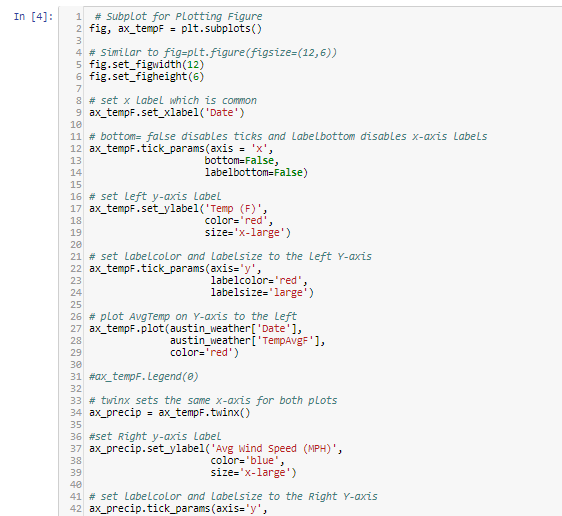


Image: TwinAxis

Reference: [https://miro.medium.com/max/1124/1\*d0qOyBrqqoXjgybphi\_RLQ.png](https://miro.medium.com/max/1124/1*d0qOyBrqqoXjgybphi_RLQ.png)

As we can there is only 1 axis,twinx() is used for twinning the x-axis and left y-axis is used for Temp and the right y-axis is used for WindMPH.

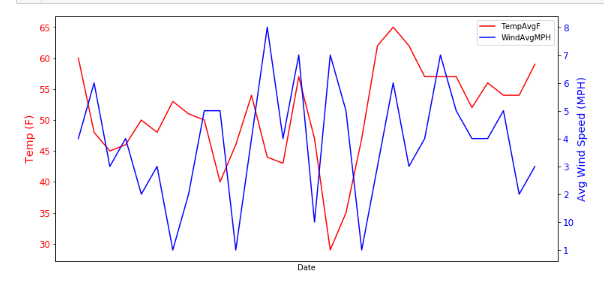


Image: TwinAxis

Reference: [https://miro.medium.com/max/1208/1\*r8oHw5OPQhMcl4aUcYA2-w.png](https://miro.medium.com/max/1208/1*r8oHw5OPQhMcl4aUcYA2-w.png)

### Plotting the same data in different units and the same x-axis

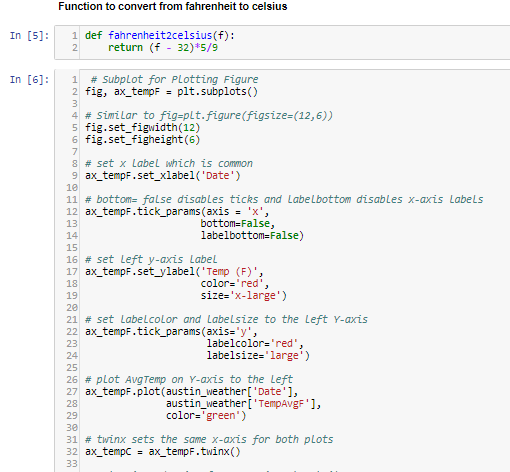


Image: Plotting the same data in different units and the same x-axis

Reference: [https://miro.medium.com/max/1020/1\*5ScqG3lYngJ1-eHUWaNbhA.png](https://miro.medium.com/max/1020/1*5ScqG3lYngJ1-eHUWaNbhA.png)

The function is defined for calculating different unit of data i.e convert from Fahrenheit to Celsius.



Image: Plotting the same data in different units and the same x-axis

Reference: [https://miro.medium.com/max/1222/1\*lbNAVBW3\_4x2aEsmcgem0A.png](https://miro.medium.com/max/1222/1*lbNAVBW3_4x2aEsmcgem0A.png)

We can see that to the left y-axis Temp in Fahrenheit is plotted and to the right x-axis Temp in Celsius is plotted.

## Stack Plot and Stem Plot

### Stack Plot

Stack plot visualizes data in stacks and shows the distribution of data over time.

#### When to use: It is used for checking multiple variable area plots in a single plot.

Eg: It is useful in understanding the change of distribution in multiple variables over an interval.

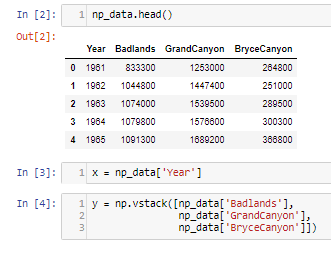


Image: Stack Plot

Reference: [https://miro.medium.com/max/662/1\*BxpiEY02pFavicN3H4-VIg.png](https://miro.medium.com/max/662/1*BxpiEY02pFavicN3H4-VIg.png)

As stack plot requires stacking, it is done in using np.vstack()



Image: Stack Plot

Reference: [https://miro.medium.com/max/684/1\*Vco9sE3BfPzN8Wm4G4hojg.png](https://miro.medium.com/max/684/1*Vco9sE3BfPzN8Wm4G4hojg.png)

plt.stackplot takes in 1st argument numeric data i.e year and 2nd argument the vertically stacked data i.e the Nationalparks.

#### Percentage Stacked plot

Similar to stack plot but each data is converted into a percentage of distribution it holds.

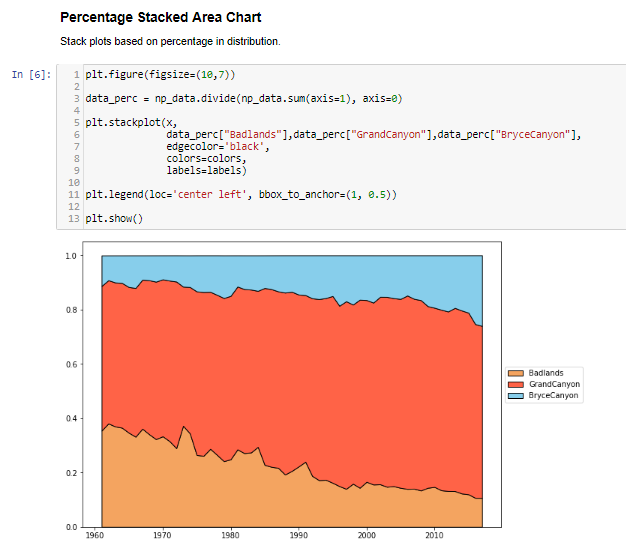


Image: Stack Plot

Reference: [https://miro.medium.com/max/1252/1\*-7K1U1plketjCnpk8fVTXw.png](https://miro.medium.com/max/1252/1*-7K1U1plketjCnpk8fVTXw.png)

data\_prec is used to divide the overall percentage into individual percentage distributions. s= np\_data.sum(axis=1) calculates sum along columns, np\_data.divide(s,axis=0) divides data along rows.

### Stem Plot

Stemplot even takes negative values, so the difference is taken of data and is plotted over time.

When to use:

It is similar to a stack plot but the difference helps in comparing the data points.

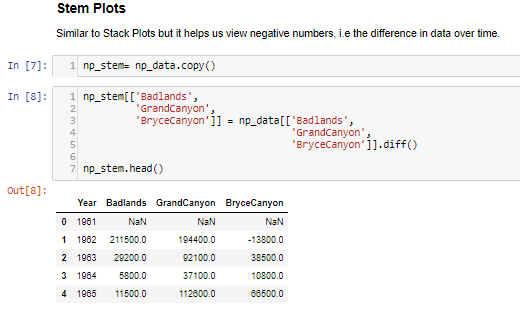


Image: Stem Plot

Reference: [https://miro.medium.com/max/1040/1\*hKw-F9uSY-5oPhW7ofdamg.png](https://miro.medium.com/max/1040/1*hKw-F9uSY-5oPhW7ofdamg.png)

diff() is used to find the difference between previous data and is stored in another copy of the data. The first data point is NaN (Not a Number) as it doesn’t contain any previous data for calculating the difference.



Image: Stem Plot

Reference: [https://miro.medium.com/max/1400/1\*SQsTcZvyGCu3nWhoReBWyw.png](https://miro.medium.com/max/1400/1*SQsTcZvyGCu3nWhoReBWyw.png)

(31n)Subplots are created to accommodate 3 rows 1 column subplots in the figure. plt.stem() takes the 1st argument as numeric data i.e year and 2nd argument as numeric data of the National Park visitors.

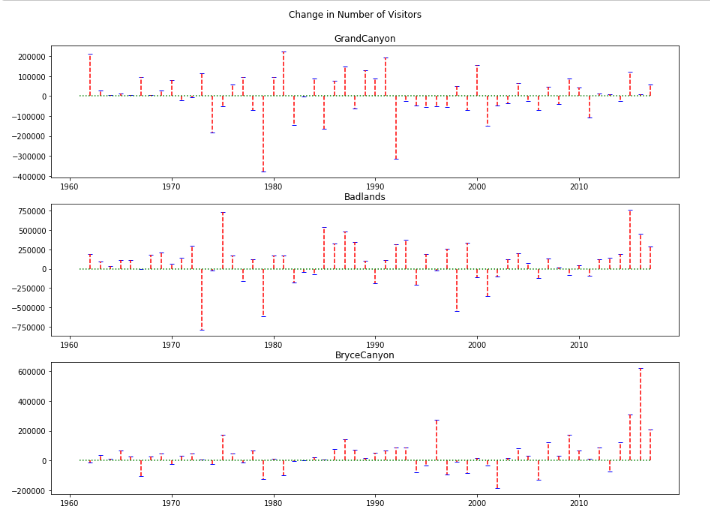


Image: Stem Plot

Reference: [https://miro.medium.com/max/1400/1\*AlIkWpXoCSdzut5xwJ4jEw.png](https://miro.medium.com/max/1400/1*AlIkWpXoCSdzut5xwJ4jEw.png)

### Bar Plot

Bar Plot shows the distribution of data over several groups. It is commonly confused with a histogram which only takes numerical data for plotting. It helps in comparing multiple numeric values.

When to use:

It is used when to compare between several groups.

Eg: Student marks in an exam.

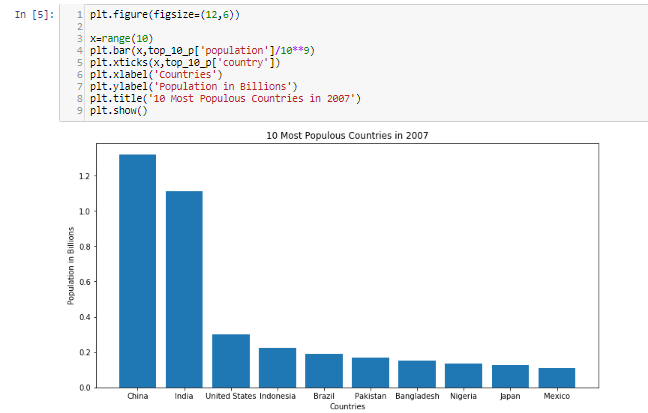


Image: Bar Plot

Reference: [https://miro.medium.com/max/1296/1\*xKfvb2z9GY3MS0Tp9yrytg.png](https://miro.medium.com/max/1296/1*xKfvb2z9GY3MS0Tp9yrytg.png)

plt.bar() takes the 1st argument as labels in numeric format and 2nd argument for the value it represents w.r.t to the plots.

### Scatter Plot

Scatter plot helps in visualizing 2 numeric variables. It helps in identifying the relationship of the data with each variable i.e correlation or trend patterns. It also helps in detecting outliers in the plot.

When to use:

It is used in Machine learning concepts like regression, where x and y are continuous variables. It is also used in clustering scatters or outlier detection.

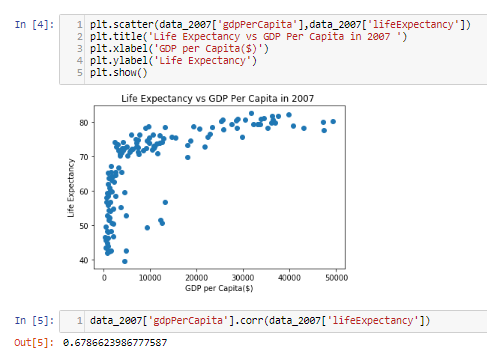


Image: Scatter Plot

Reference: [https://miro.medium.com/max/974/1\*u2qXX6T1h4oRppTe\_y0d4g.png](https://miro.medium.com/max/974/1*u2qXX6T1h4oRppTe_y0d4g.png)

plt.scatter() takes 2 numeric arguments for scattering data points in the plot. It is similar to line plot except without the connected straight lines. By corr we mean correlation and it means that how correlated GDP is with life expectancy, as we can see that it is positive it means as GDP of a country increases, life expectancy too increases.

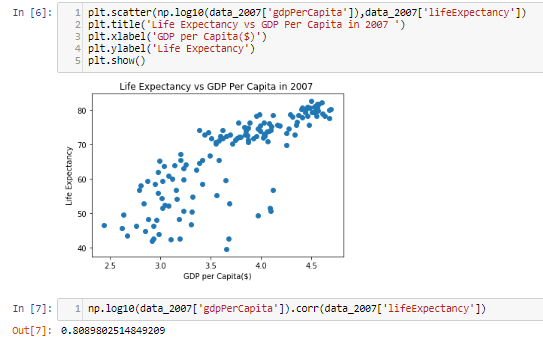


Image: Scatter Plot

Reference: [https://miro.medium.com/max/1086/1\*o8g6tfbhafhcq333IG1vIw.png](https://miro.medium.com/max/1086/1*o8g6tfbhafhcq333IG1vIw.png)

By taking the log of GDP, we can there is a much better correlation as we can fit points better, it converts GDP in log scale i.e log($1000)=3.

### 3D Scatterplot

3D Scatterplot helps in visualizing 3 numerical variables in a three- dimensional plot.

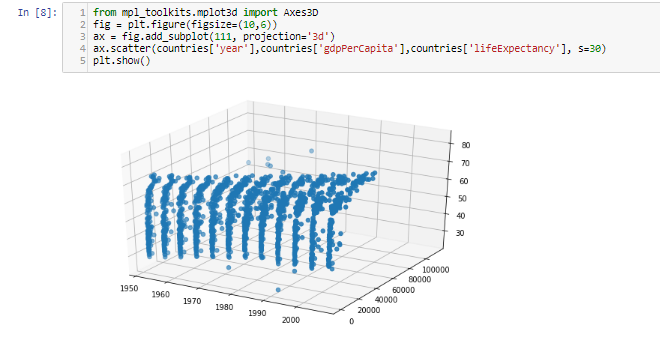


Image: 3D Scatterplot

Reference: [https://miro.medium.com/max/1320/1\*1vqIWOwAZXcl9RJ6Iutp6g.png](https://miro.medium.com/max/1320/1*1vqIWOwAZXcl9RJ6Iutp6g.png)

It is similar to scatter except we add 3 numerical variables this time. By looking at the plot we can make an inference that as the year and GDP increases, life expectancy too increases.

# Advanced data visualization using seaborn

Data visualization occupies a special place at the heart of all data-related professions. Nothing is more satisfying for a data scientist than to take a large set of random numbers and turn it into a beautiful visual.

The majority of data visuals created by data scientists are created with Python and its twin visualization libraries: Matplotlib and Seaborn. Matplotlib and Seaborn are widely used to create graphs that enable individuals and companies to make sense of terabytes of data.

## What is Seaborn?

So, what are these two libraries, exactly?

Matplotlib is the king of Python data visualization libraries and makes it a breeze to explore tabular data visually.

Seaborn is another Python data visualization library built on top of Matplotlib that introduces some features that weren’t previously available, and, in this tutorial, we’ll use Seaborn.

To follow along with this project, you’ll also need to know about Pandas, a powerful library that manipulates and analyzes tabular data.

In this blog post, we’ll learn how to perform data analysis through visualizations created with Seaborn. You will be introduced to histograms, KDEs, bar charts, and more. By the end, you’ll have a solid understanding of how to visualize data.

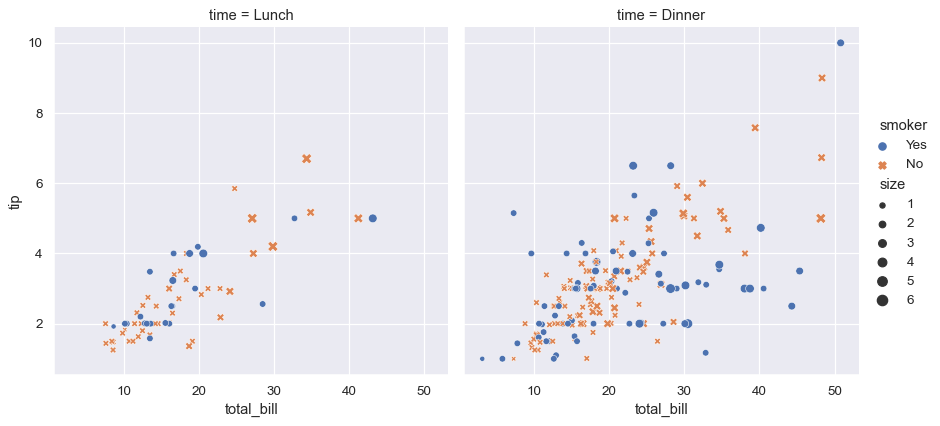


Image: What is Seaborn?

Reference: <https://seaborn.pydata.org/_images/introduction_1_0.png>

## Installing the libraries and loading the data

We will start by installing the libraries and importing our data. Running the below command will install the Pandas, Matplotlib, and Seaborn libraries for data visualization:

pip install pandas matplotlib seaborn

Now, let’s import the libraries under their standard aliases:

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

Next, load in the data to be analyzed. The dataset contains physical measurements of 54,000 diamonds and their prices. You can download the original dataset as a CSV file from here on Kaggle, but we will be using a shortcut:

diamonds = sns.load\_dataset("diamonds")

Because the dataset is already built into Seaborn, we can load it as pandas.DataFrame using the load\_dataset function.

>>> type(diamonds)

pandas.core.frame.DataFrame

## Exploring the dataset

Before we dive head-first into visuals, let’s ensure we have a high-level understanding of our dataset:

>>> diamonds.head()

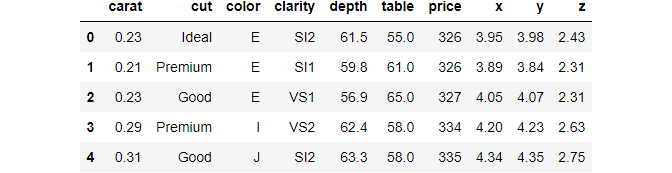


Image: Installing the libraries and loading the data

Reference: <https://blog.logrocket.com/wp-content/uploads/2021/11/dataset-graph.png>

We have used the handy head function of Pandas that prints out the first five rows of the data frame. head should be the first function you use when you load a dataset into your environment for the first time.

Notice the dataset has ten variables  — three categorical and seven numeric.

* Carat: weight of a diamond
* Cut: the cut quality with five possible values in increasing order: Fair, Good, Very Good, Premium, Ideal
* Color: the color of a diamond with color codes from D (the best) to J (the worst)
* Clarity: the clarity of a diamond with eight clarity codes
* X: length of a diamond (mm)
* Y: the height of a diamond (mm)
* Z: depth of a diamond (mm)
* Depth: total depth percentage calculated as Z / average(X, Y)
* Table: the ratio of the height of a diamond to its widest point
* Price: diamond price in dollars

Instead of counting all variables one by one, we can use the shape attribute of the data frame:

>>> diamonds.shape

(53940, 10)

There are 53,940 diamonds recorded, along with their ten different features. Now, let’s print a five-number summary of the dataset:

>>> diamonds.describe()



Image: Exploring the Dataset

Reference: <https://blog.logrocket.com/wp-content/uploads/2021/11/dataset-summary.png>

The describe function displays some critical metrics of each numeric variable in a data frame. Here are some observations from the above output:

* The cheapest diamond in the dataset costs $326, while the most expensive costs almost 60 times more , $18,823
* The minimum weight of a diamond is 0.2 carats, while the max is 5.01. The average weight is ~0.8
* Looking at the mean of X and Y features, we see that diamonds, on average, have the same height and width

Now that we are comfortable with the features in our dataset, we can start plotting them to uncover more insights.

## Performing univariate analysis with Seaborn

In the previous section, we started something called “Exploratory Data Analysis” (EDA), which is the basis for any data-related project.

The goal of EDA is simple  —  get to know your dataset at the deepest level possible. Becoming intimate with the data and learning its relationships between its variables is an absolute must.

Completing a successful and thorough EDA lays the groundwork for future stages of your data project.

We have already performed the first stage of EDA, which was a simple “get acquainted” step. Now, let’s go deeper, starting with univariate analysis.

As the name suggests, we’ll explore variables one at a time, not the relationships between them just yet. Before we start plotting, we take a small dataset sample because 54,000 is more than we need, and we can learn about the data set pretty well with just 3,000 and to prevent overplotting.

sample = diamonds.sample(3000)

To take a sample, we use the sample function of pandas, passing in the number of random data points to include in a sample.

## Creating histograms in Seaborn

Now, we create our first plot, which is a histogram:

sns.histplot(x=sample["price"])

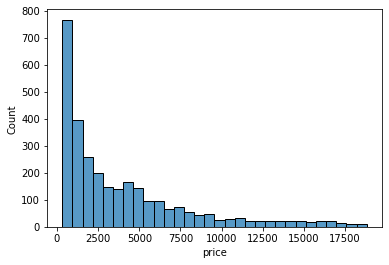


Image: Creating histograms in Seaborn

Reference: <https://blog.logrocket.com/wp-content/uploads/2021/11/histogram-count.png>

Histograms only work on numeric variables. They divide the data into an arbitrary number of equal-sized bins and display how many diamonds go into each bin. Here, we can approximate that nearly 800 diamonds are priced between 0 and 1000.

Each bin contains the count of diamonds. Instead, we might want to see what percentage of the diamonds falls into each bin. For that, we will set the stat argument of the histplot function to percent:

>>> sns.histplot(sample["price"], stat="percent")

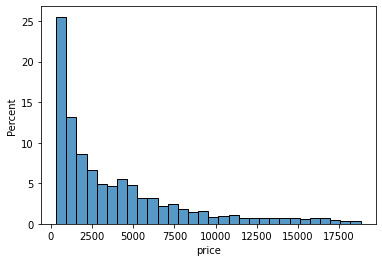


Image: Creating histograms in Seaborn

Reference: <https://blog.logrocket.com/wp-content/uploads/2021/11/histogram.png>

Now, the height of each bar/bin shows the percentage of the diamonds. Let’s do the same for the carat of the diamonds:

sns.histplot(sample["carat"], stat="percent")

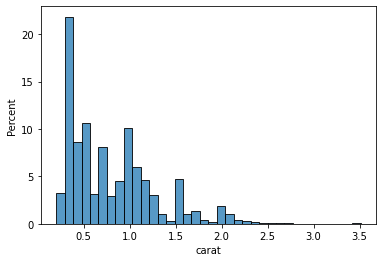


Image: Creating histograms in Seaborn

Reference: <https://blog.logrocket.com/wp-content/uploads/2021/11/diamod-histogram.png>

Looking at the first few bars, we can conclude that the majority of the diamonds weigh less than 0.5 carats. Histograms aim to take a numeric variable and show what its shape generally looks like. Statisticians look at the distribution of a variable.

However, histograms aren’t the only plots that do the job. There is also a plot called KDE Plot (Kernel Density Estimate), which uses some fancy math under the hood to draw curves like this:

sns.kdeplot(sample["table"])

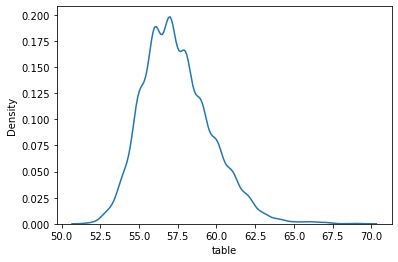


Image: Creating histograms in Seaborn

Reference: <https://blog.logrocket.com/wp-content/uploads/2021/11/density-table.png>

Creating the KDE plot of the table variable shows us that the majority of diamonds measure between 55.0 and 60.0. At this point, I will leave it to you to plot the KDEs and histograms of other numeric variables because we have to move on to categorical features.

## Creating count plots in Seaborn

The most common plot for categorical features is a countplot. Passing the name of a categorical feature in our dataset to Seaborn’s countplot draws a bar chart, with each bar height representing the number of diamonds in each category. Below is a countplot of diamond cuts:

sns.countplot(sample["cut"])

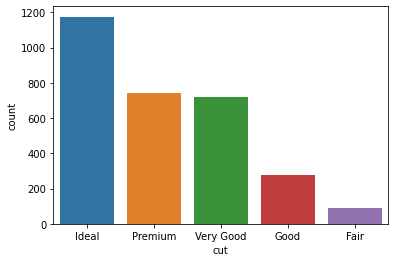


Image: Creating count plots in Seaborn

Reference: <https://blog.logrocket.com/wp-content/uploads/2021/11/dataset-color.png>

We can see that our dataset consists of much more ideal diamonds than premium or very good diamonds. Here is a countplot of colors for the interested:

sns.countplot(sample["color"])

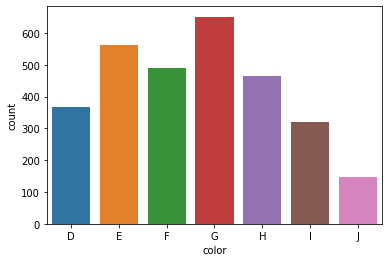


Image: Creating count plots in Seaborn

Reference: <https://blog.logrocket.com/wp-content/uploads/2021/11/countplot-colors.png>

This concludes the univariate analysis section of the EDA.

## Performing bivariate analysis with Seaborn

Now, let’s look at the relationships between two variables at a time. Let’s start with the connection between diamond carats and price.

### Creating scatterplots

We already know that diamonds with higher carats cost more. Let’s see if we can visually capture this trend:

sns.scatterplot(x=sample["carat"], y=sample["price"])

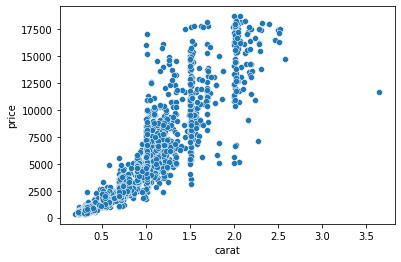


Image: Creating scatterplots

Reference: <https://blog.logrocket.com/wp-content/uploads/2021/11/carat-count.png>

Here, we are using another Seaborn function that plots a scatter plot. Scatterplots are one of the most widely-used charts because they accurately show the relationships between two variables by using a cloud of dots.

Above, each dot represents a single diamond. The dots’ positions are determined by their carat and price measurements, which we passed to X and Y parameters of the scatterplot function.

The plot confirms our assumptions — heavier diamonds tend to be more expensive. We are drawing this conclusion based on the curvy upward trend of the dots.

sns.scatterplot(x=sample["depth"], y=sample["table"])



Image: Creating scatterplots

Reference: <https://blog.logrocket.com/wp-content/uploads/2021/11/scatterplot.png>

Let’s try plotting depth against the table. Frankly, this scatterplot is disappointing because we can’t draw a tangible conclusion as we did with the previous one.

### Building boxplots

Another typical bivariate plot is a boxplot, which plots the distribution of a variable against another based on their five-number summary:

sns.boxplot(x=sample["color"], y=sample["price"])

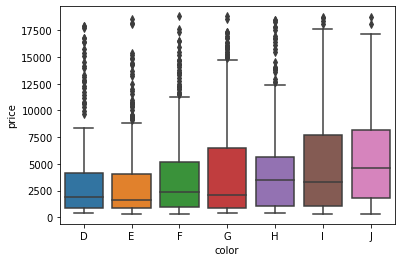


Image: Creating scatterplots

Reference: <https://blog.logrocket.com/wp-content/uploads/2021/11/boxplaot.png>

The boxplot above shows the relationship between each color category and their respective prices. The horizontal vertices at the bottom and top of each vertical line of a box represent that category’s minimum and maximum values. The edges of the boxes, specifically the bottom and top edges, represent the 25th and 75th percentiles.

In other words, the bottom edge of the first box tells us that 25% of D-colored diamonds cost less than about $1,250, while the top edge says that 75% of diamonds cost less than about $4,500. The little horizontal line in the middle denotes the median ,  the 50% mark.

The dark dots above are outliers. Let’s plot a boxplot of diamond clarities and their relationship with carat:

sns.boxplot(diamonds["clarity"], diamonds["carat"])

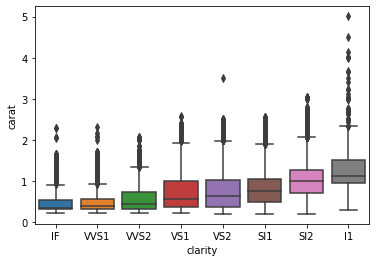


Image: Building boxplots

Reference: <https://blog.logrocket.com/wp-content/uploads/2021/11/diamond-clarity.png>

# Pandas profiling for report generation

## What is Pandas Profiling

Pandas Profiling is an open-source python library, which allows you to do your EDA very quickly. By the way, it also generates an interactive HTML report, which you can show to anyone. Imagine going to your boss, who doesn’t code, with an interactive description of the company’s data. Great for your branding, right?

These are some of the things you get in your report:

* Type inference: detect the types of columns in a Data Frame.
* Essentials: type, unique values, missing values.
* Quantile statistics like minimum value, Q1, median, Q3, maximum, range, interquartile range.
* Descriptive statistics like mean, mode, standard deviation, sum, median absolute deviation, coefficient of variation, kurtosis, skewness.
* Most frequent values.
* Histogram.
* Correlations highlighting of highly correlated variables, Spearman, Pearson and Kendall matrices.
* Missing values matrix, count, heat-map and dendrogram of missing values.
* Text analysis learns about categories (Uppercase, Space), scripts (Latin, Cyrillic) and blocks (ASCII) of text data.
* File and Image analysis extract file sizes, creation dates and dimensions and scan for truncated images or those containing EXIF information.

Given this, let’s get going.

## How to install Pandas Profiling

First of all, you need to install the package.

#installing Pandas Profiling

!pip install https://github.com/pandas-profiling/pandas-profiling/archive/master.zip -q

Now, let’s import both pandas and panda\_profiling.

#importing modules

from pandas\_profiling import ProfileReport

import pandas as pd

We will be using the Titanic dataset to complete our analysis, let’s import it:

#linking df to our dataset

df = pd.read\_csv("https://raw.githubusercontent.com/datasciencedojo/

datasets/master/titanic.csv")

After you import it, you should always take a look at your dataset and then merely link report to it:

report = ProfileReport(df)

Now you simply have “to tell” Pandas Profiling to make a report out of your dataset.

report.to\_notebook\_iframe()



Image: How to install Pandas Profiling

Reference: [https://miro.medium.com/max/1400/0\*LtaSDCg7z7BMMkJ5](https://miro.medium.com/max/1400/0*LtaSDCg7z7BMMkJ5)

If you use a Jupyter Notebook, your report is embedded in it. However, you may want to use it in other places, and Pandas Profiling also allows you to do that. Just type this to save your report as an HTML file:

report.to\_file('file\_name')

If you want the HTML source “code” (don’t kill me for calling it code), which would be quite rare, however possible, just type:

report.to\_html()

It will return the whole HTML source code.



Image: How to install Pandas Profiling

Reference: [https://miro.medium.com/max/1400/1\*qeXNHD0ZQA1IYJMF1tTbxw.png](https://miro.medium.com/max/1400/1*qeXNHD0ZQA1IYJMF1tTbxw.png)

You can even save it as a JSON file:

# As a string

json\_data = profile.to\_json()

# As a file

profile.to\_file("your\_report.json")

# Need for data visualization

## What is Data Visualization?

With so much information being collected through data analysis in the business world today, we must have a way to paint a picture of that data so we can interpret it. Data visualization gives us a clear idea of what the information means by giving it visual context through maps or graphs. This makes the data more natural for the human mind to comprehend and therefore makes it easier to identify trends, patterns, and outliers within large data sets.

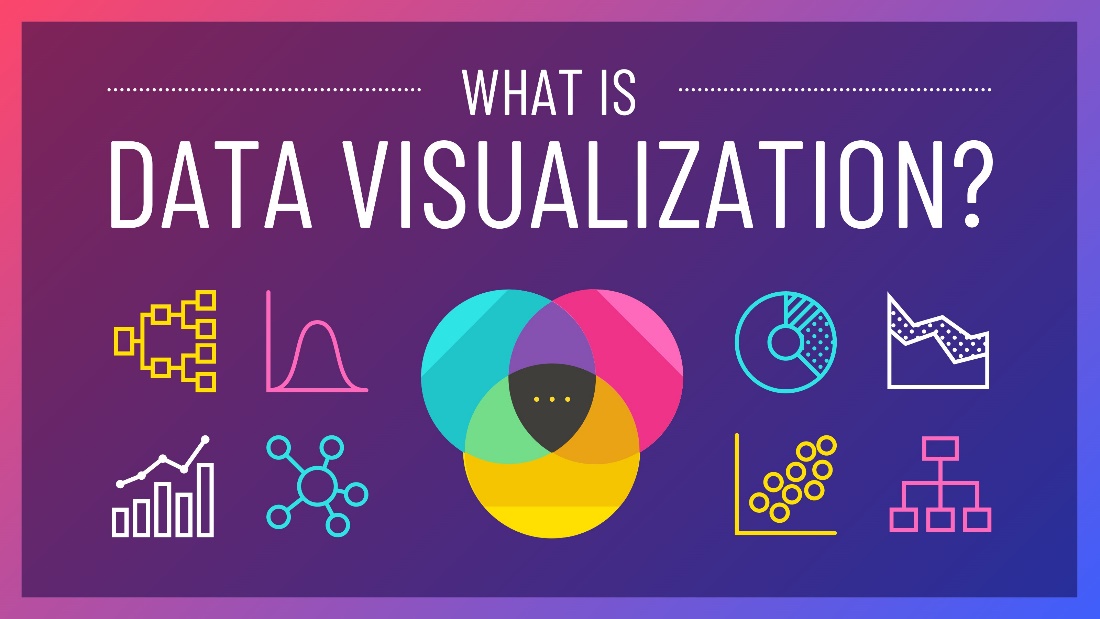


Image: What is Data Visualization?

Reference: <https://venngage-wordpress.s3.amazonaws.com/uploads/2020/06/What-is-Data-Visualization-Blog-Header.jpg>

## Why is Data Visualization Important?

No matter what business or career you’ve chosen, data visualization can help by delivering data in the most efficient way possible. As one of the essential steps in the business intelligence process, data visualization takes the raw data, models it, and delivers the data so that conclusions can be reached. In advanced analytics, data scientists are creating machine learning algorithms to better compile essential data into visualizations that are easier to understand and interpret.

Specifically, data visualization uses visual data to communicate information in a manner that is universal, fast, and effective. This practice can help companies identify which areas need to be improved, which factors affect customer satisfaction and dissatisfaction, and what to do with specific products (where should they go and who should they be sold to). Visualized data gives stakeholders, business owners, and decision-makers a better prediction of sales volumes and future growth.

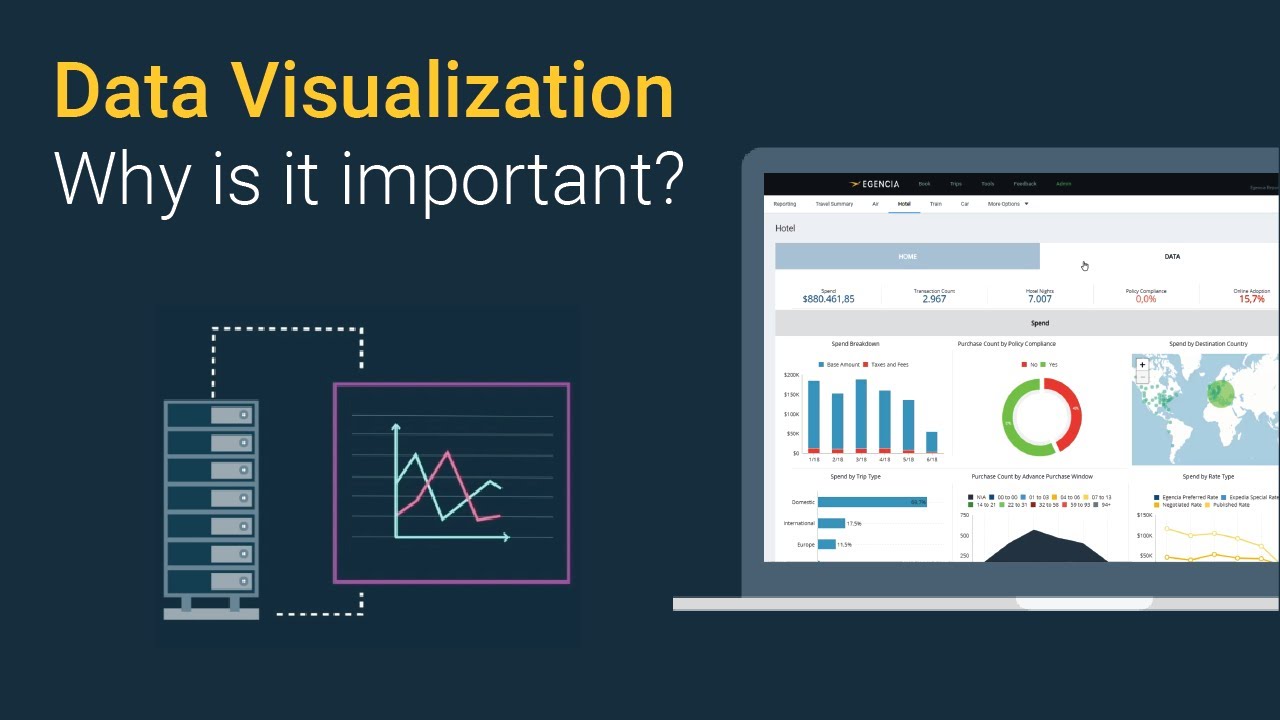


Image: Why is Data Visualization Important?

Reference: <https://i.ytimg.com/vi/VyhLRJVoIrI/maxresdefault.jpg>

## What Are The Benefits of Data Visualization?

Data visualization positively affects an organization’s decision-making process with interactive visual representations of data. Businesses can now recognize patterns more quickly because they can interpret data in graphical or pictorial forms. Here are some more specific ways that data visualization can benefit an organization:

* Correlations in Relationships: Without data visualization, it is challenging to identify the correlations between the relationship of independent variables. By making sense of those independent variables, we can make better business decisions.
* Trends Over Time: While this seems like an obvious use of data visualization, it is also one of the most valuable applications. It’s impossible to make predictions without having the necessary information from the past and present. Trends over time tell us where we were and where we can potentially go.
* Frequency: Closely related to trends over time is frequency. By examining the rate, or how often, customers purchase and when they buy gives us a better feel for how potential new customers might act and react to different marketing and customer acquisition strategies.
* Examining the Market: Data visualization takes the information from different markets to give you insights into which audiences to focus your attention on and which ones to stay away from. We get a clearer picture of the opportunities within those markets by displaying this data on various charts and graphs.
* Risk and Reward: Looking at value and risk metrics requires expertise because, without data visualization, we must interpret complicated spreadsheets and numbers. Once information is visualized, we can then pinpoint areas that may or may not require action.
* Reacting to the Market: The ability to obtain information quickly and easily with data displayed clearly on a functional dashboard allows businesses to act and respond to findings swiftly and helps to avoid making mistakes.

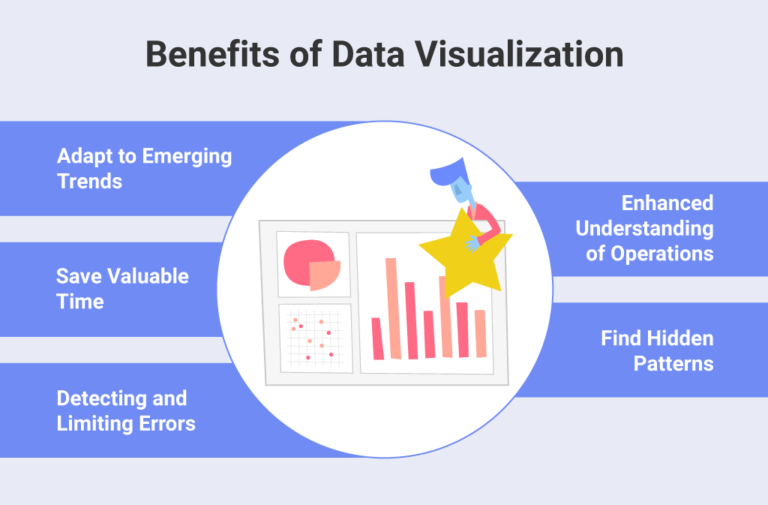


Image: Benefits of Data Visualization

Reference: <https://cdn.slingshotapp.io/wp-content/uploads/2021/09/slingshot-Benefits-of-Data-Visualization-768x505.png>

## Which Data Visualization Techniques are Used?

There are many different methods of putting together information in a way that the data can be visualized. Depending on the data being modeled, and what its intended purpose is, a variety of different graphs and tables may be utilized to create an easy to interpret dashboard. Some visualizations are manually created, while others are automated. Either way, there are many types to meet your visualization needs.

* Infographics: Unlike a single data visualization, infographics take an extensive collection of information and gives you a comprehensive visual representation. An infographic is excellent for exploring complex and highly-subjective topics.
* Heatmap Visualization: This method uses a graph with numerical data points highlighted in light or warm colors to indicate whether the data is a high-value or a low-value point. Psychologically, this data visualization method helps the viewer to identify the information because studies have shown that humans interpret colors much better than numbers and letters.
* Fever Charts: A fever chart shows changing data over a period of time. As a marketing tool, we could take the performance from the previous year and compare that to the prior year to get an accurate projection of next year. This can help decision-makers easily interpret wide and varying data sources.
* Area Chart (or Graph): Area charts are excellent for visualizing the data’s time-series relationship. Whether you’re looking at the earnings for individual departments on a month to month basis or the popularity of a product since the 1980s, area charts can visualize this relationship.
* Histogram: Rather than looking at the trends over time, histograms are measuring frequencies instead. These graphs show the distribution of numerical data using an automated data visualization formula to display a range of values that can be easily interpreted.

## Who Uses Data Visualization?

Data visualization is used across all industries to increase sales with existing customers and target new markets and demographics for potential customers. The World Advertising and Research Center (WARC) predicts that in 2020 half of the world’s advertising dollars will be spent online, which means companies everywhere have discovered the importance of web data. As a crucial step in data analytics, data visualization gives companies critical insights into untapped information and messages that would otherwise be lost. The days of scouring through thousands of rows of spreadsheets are over, as now we have a visual summary of data to identify trends and patterns.

References

1. <https://mycstutorial.in/data-handling-using-pandas-i/>
2. <http://python.mykvs.in/presentation/presentation2021/class%20xii/informatics%20practices/Python%20Pandas.pdf>
3. <https://towardsdatascience.com/data-handling-using-pandas-cleaning-and-processing-3aa657dc9418>
4. <https://matplotlib.org/>
5. <https://www.mygreatlearning.com/blog/matplotlib-tutorial-for-data-visualisation/>
6. <https://datacarpentry.org/python-socialsci/13-matplotlib/index.html>
7. <https://www.geeksforgeeks.org/data-visualization-with-python-seaborn/>
8. <https://datalore.jetbrains.com/view/notebook/v8mLoENq8XTfmStTCLNMV6>
9. <https://www.edureka.co/blog/python-seaborn-tutorial/>
10. <https://towardsdatascience.com/exploratory-data-analysis-with-pandas-profiling-de3aae2ddff3>
11. <https://www.thiscodeworks.com/generate-pandas-profiling-report-and-save-it-to-html-python/61f0b8384faf03001546b098>
12. <https://www.geeksforgeeks.org/what-is-data-visualization-and-why-is-it-important/>
13. <https://splashbi.com/importance-purpose-benefit-of-data-visualization-tools/>
14. <https://www.edureka.co/blog/needs-and-benefits-of-data-visualization/>
15. <https://piktochart.com/blog/what-is-data-visualization/>