**NAME: ASIF ERFAN KHAN**

**ROLL NUMBER: 546**

**COURSE: MSc CS**

**SUBJECT: FUNDAMENTALS OF DATA SCIENCE**

**PRACTICAL: 1-8**

**PRACTICAL 1**

Data Collection: Data collection is defined as the procedure of collecting, measuring and analyzing accurate insights for research using standard validated techniques. A researcher can evaluate their hypothesis on the basis of collected data. In most cases, data collection is the primary and most important step for research, irrespective of the field of research. The approach of data collection is different for different fields of study, depending on the required information. The most critical objective of data collection is ensuring that information-rich and reliable data is collected for statistical analysis so that data-driven decisions can be made for research.

Data Collection and Datasets

From .csv Files From Excel Files From SQL Files

my\_dict={'Name':["a","b","c","d","e","f","g"], 'age':[20,27,35,45,55,43,35], 'designation':["VP","CEO","CFO","VP","VP","CEO","MD"]}import pandas as pdimport numpy as npdf=pd.DataFrame(my\_dict)df

Name age designation0 a 20 VP1 b 27 CEO2 c 35 CFO3 d 45 VP4 e 55 VP5 f 43 CEO6 g 35 MD

df.to\_csv('Csv example')df

Name age designation0 a 20 VP1 b 27 CEO2 c 35 CFO3 d 45 VP4 e 55 VP5 f 43 CEO6 g 35 MD

df\_csv=pd.read\_csv('Csv example')df\_csv

Unnamed: 0 Name age designation0 0 a 20 VP1 1 b 27 CEO2 2 c 35 CFO3 3 d 45 VP4 4 e 55 VP5 5 f 43 CEO6 6 g 35 MD

df.to\_csv('CSV Ex',index=False)df\_csv=pd.read\_csv('CSV Ex')df\_csv

Name age designation0 a 20 VP1 b 27 CEO2 c 35 CFO3 d 45 VP4 e 55 VP5 f 43 CEO6 g 35 MD

import pandas as pdLocation = "/content/drive/MyDrive/Colab Notebooks/student-mat.csv"df = pd.read\_csv(Location, header=None)df.head()

0 1 2 3 4 5 6 7 8 9 \0 school sex age address famsize Pstatus Medu Fedu Mjob Fjob 1 GP F 18 U GT3 A 4 4 at\_home teacher 2 GP F 17 U GT3 T 1 1 at\_home other 3 GP F 15 U LE3 T 1 1 at\_home other 4 GP F 15 U GT3 T 4 2 health services  ... 23 24 25 26 27 28 29 30 31 32 0 ... famrel freetime goout Dalc Walc health absences G1 G2 G3 1 ... 4 3 4 1 1 3 6 5 6 6 2 ... 5 3 3 1 1 3 4 5 5 6 3 ... 4 3 2 2 3 3 10 7 8 10 4 ... 3 2 2 1 1 5 2 15 14 15 [5 rows x 33 columns]

import pandas as pdLocation = "/content/drive/MyDrive/Colab Notebooks/student-mat.csv"df = pd.read\_csv(Location)df.head()

school sex age address famsize Pstatus Medu Fedu Mjob Fjob ... \0 GP F 18 U GT3 A 4 4 at\_home teacher ... 1 GP F 17 U GT3 T 1 1 at\_home other ... 2 GP F 15 U LE3 T 1 1 at\_home other ... 3 GP F 15 U GT3 T 4 2 health services ... 4 GP F 16 U GT3 T 3 3 other other ...  famrel freetime goout Dalc Walc health absences G1 G2 G3 0 4 3 4 1 1 3 6 5 6 6 1 5 3 3 1 1 3 4 5 5 6 2 4 3 2 2 3 3 10 7 8 10 3 3 2 2 1 1 5 2 15 14 15 4 4 3 2 1 2 5 4 6 10 10 [5 rows x 33 columns]

import pandas as pdLocation = "/content/drive/MyDrive/Colab Notebooks/student-mat.csv" *# To add headers as we load the data...*df = pd.read\_csv(Location, names=['RollNo','Names','Grades']) *# To add headers to a dataframe*df.columns = ['RollNo','Names','Grades']df.head()

RollNo \school sex age address famsize Pstatus Medu Fedu Mjob Fjob reason guardian traveltime studytime failures schoolsup famsup paid activities nursery higher internet romantic famrel freetime goout Dalc Walc health absences G1 GP F 18 U GT3 A 4 4 at\_home teacher course mother 2 2 0 yes no no no yes yes no no 4 3 4 1 1 3 6 5  17 U GT3 T 1 1 at\_home other course father 1 2 0 no yes no no no yes yes no 5 3 3 1 1 3 4 5  15 U LE3 T 1 1 at\_home other other mother 1 2 3 yes no yes no yes yes yes no 4 3 2 2 3 3 10 7  GT3 T 4 2 health services home mother 1 3 0 no yes yes yes yes yes yes yes 3 2 2 1 1 5 2 15  Names \school sex age address famsize Pstatus Medu Fedu Mjob Fjob reason guardian traveltime studytime failures schoolsup famsup paid activities nursery higher internet romantic famrel freetime goout Dalc Walc health absences G2 GP F 18 U GT3 A 4 4 at\_home teacher course mother 2 2 0 yes no no no yes yes no no 4 3 4 1 1 3 6 6  17 U GT3 T 1 1 at\_home other course father 1 2 0 no yes no no no yes yes no 5 3 3 1 1 3 4 5  15 U LE3 T 1 1 at\_home other other mother 1 2 3 yes no yes no yes yes yes no 4 3 2 2 3 3 10 8  GT3 T 4 2 health services home mother 1 3 0 no yes yes yes yes yes yes yes 3 2 2 1 1 5 2 14  Grades school sex age address famsize Pstatus Medu Fedu Mjob Fjob reason guardian traveltime studytime failures schoolsup famsup paid activities nursery higher internet romantic famrel freetime goout Dalc Walc health absences G3 GP F 18 U GT3 A 4 4 at\_home teacher course mother 2 2 0 yes no no no yes yes no no 4 3 4 1 1 3 6 6  17 U GT3 T 1 1 at\_home other course father 1 2 0 no yes no no no yes yes no 5 3 3 1 1 3 4 6  15 U LE3 T 1 1 at\_home other other mother 1 2 3 yes no yes no yes yes yes no 4 3 2 2 3 3 10 10  GT3 T 4 2 health services home mother 1 3 0 no yes yes yes yes yes yes yes 3 2 2 1 1 5 2 15

import pandas as pdnames = ['Bob','Jessica','Mary','John','Mel']grades = [76,95,77,78,99]bsdegrees = [1,1,0,0,1]msdegrees = [2,1,0,0,0]phddegrees = [0,1,0,0,0]Degrees = zip(names,grades,bsdegrees,msdegrees,phddegrees)columns = ['Names','Grades','BS','MS','PhD']df = pd.DataFrame(data = Degrees, columns=columns)df

Names Grades BS MS PhD0 Bob 76 1 2 01 Jessica 95 1 1 12 Mary 77 0 0 03 John 78 0 0 04 Mel 99 1 0 0

import pandas as pdLocation = "/content/drive/MyDrive/Colab Notebooks/gradedata.xlsx"df = pd.read\_excel(Location) *#Changing column Names*df.columns = ['first','last','sex','age','exer','hrs','grd','addr']df.head()

first last sex age exer hrs grd \0 Marcia Pugh female 17 3 10 82.4 1 Kadeem Morrison male 18 4 4 78.2 2 Nash Powell male 18 5 9 79.3 3 Noelani Wagner female 14 2 7 83.2 4 Noelani Cherry female 18 4 15 87.4  addr 0 7379 Highland Rd. , Dublin, GA 31021 1 8 Bayport St. , Honolulu, HI 96815 2 Encino, CA 91316, 3 Lilac Street 3 Riverview, FL 33569, 9998 North Smith Dr. 4 97 SE. Ocean Street , Bethlehem, PA 18015

pip install xlsxwriter

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/Collecting xlsxwriter Downloading XlsxWriter-3.0.3-py3-none-any.whl (149 kB)

import pandas as pdnames = ['Bob','Jessica','Mary','John','Mel']grades = [76,95,77,78,99]GradeList = zip(names,grades)df = pd.DataFrame(data = GradeList,columns=['Names','Grades'])writer = pd.ExcelWriter('dataframe.xlsx', engine='xlsxwriter')df.to\_excel(writer, sheet\_name='Sheet1')writer.save()

import sqlite3con = sqlite3.connect("/content/drive/MyDrive/Colab Notebooks/portal\_mammals.sqlite")cur = con.cursor()**for** row **in** cur.execute('SELECT \* FROM species;'): print(row) con.close()

('AB', 'Amphispiza', 'bilineata', 'Bird')('AH', 'Ammospermophilus', 'harrisi', 'Rodent')('AS', 'Ammodramus', 'savannarum', 'Bird')('BA', 'Baiomys', 'taylori', 'Rodent')('CB', 'Campylorhynchus', 'brunneicapillus', 'Bird')('CM', 'Calamospiza', 'melanocorys', 'Bird')('CQ', 'Callipepla', 'squamata', 'Bird')('CS', 'Crotalus', 'scutalatus', 'Reptile')('CT', 'Cnemidophorus', 'tigris', 'Reptile')('CU', 'Cnemidophorus', 'uniparens', 'Reptile')('CV', 'Crotalus', 'viridis', 'Reptile')('DM', 'Dipodomys', 'merriami', 'Rodent')('DO', 'Dipodomys', 'ordii', 'Rodent')('DS', 'Dipodomys', 'spectabilis', 'Rodent')('DX', 'Dipodomys', 'sp.', 'Rodent')('EO', 'Eumeces', 'obsoletus', 'Reptile')('GS', 'Gambelia', 'silus', 'Reptile')('NL', 'Neotoma', 'albigula', 'Rodent')('NX', 'Neotoma', 'sp.', 'Rodent')('OL', 'Onychomys', 'leucogaster', 'Rodent')('OT', 'Onychomys', 'torridus', 'Rodent')('OX', 'Onychomys', 'sp.', 'Rodent')('PB', 'Chaetodipus', 'baileyi', 'Rodent')('PC', 'Pipilo', 'chlorurus', 'Bird')('PE', 'Peromyscus', 'eremicus', 'Rodent')('PF', 'Perognathus', 'flavus', 'Rodent')('PG', 'Pooecetes', 'gramineus', 'Bird')('PH', 'Perognathus', 'hispidus', 'Rodent')('PI', 'Chaetodipus', 'intermedius', 'Rodent')('PL', 'Peromyscus', 'leucopus', 'Rodent')('PM', 'Peromyscus', 'maniculatus', 'Rodent')('PP', 'Chaetodipus', 'penicillatus', 'Rodent')('PU', 'Pipilo', 'fuscus', 'Bird')('PX', 'Chaetodipus', 'sp.', 'Rodent')('RF', 'Reithrodontomys', 'fulvescens', 'Rodent')('RM', 'Reithrodontomys', 'megalotis', 'Rodent')('RO', 'Reithrodontomys', 'montanus', 'Rodent')('RX', 'Reithrodontomys', 'sp.', 'Rodent')('SA', 'Sylvilagus', 'audubonii', 'Rabbit')('SB', 'Spizella', 'breweri', 'Bird')('SC', 'Sceloporus', 'clarki', 'Reptile')('SF', 'Sigmodon', 'fulviventer', 'Rodent')('SH', 'Sigmodon', 'hispidus', 'Rodent')('SO', 'Sigmodon', 'ochrognathus', 'Rodent')('SS', 'Spermophilus', 'spilosoma', 'Rodent')('ST', 'Spermophilus', 'tereticaudus', 'Rodent')('SU', 'Sceloporus', 'undulatus', 'Reptile')('SX', 'Sigmodon', 'sp.', 'Rodent')('UL', 'Lizard', 'sp.', 'Reptile')('UP', 'Pipilo', 'sp.', 'Bird')('UR', 'Rodent', 'sp.', 'Rodent')('US', 'Sparrow', 'sp.', 'Bird')('ZL', 'Zonotrichia', 'leucophrys', 'Bird')('ZM', 'Zenaida', 'macroura', 'Bird')

import sqlite3 *# Create a SQL connection to our SQLite database*con = sqlite3.connect("/content/drive/MyDrive/Colab Notebooks/portal\_mammals.sqlite")cur = con.cursor() *# Return all results of query*cur.execute('SELECT plot\_id FROM plots WHERE plot\_type="Control"')print(cur.fetchall()) *# Return first result of query*cur.execute('SELECT species FROM species WHERE taxa="Bird"')print(cur.fetchone()) *# Be sure to close the connection*con.close()

[(2,), (4,), (8,), (11,), (12,), (14,), (17,), (22,)]('bilineata',)

import pandas as pdimport sqlite3 *# Read sqlite query results into a pandas DataFrame*con = sqlite3.connect("/content/drive/MyDrive/Colab Notebooks/portal\_mammals.sqlite")df = pd.read\_sql\_query("SELECT \* from surveys", con) *# Verify that result of SQL query is stored in the dataframe*print(df.head())con.close()

record\_id month day year plot\_id species\_id sex hindfoot\_length \0 1 7 16 1977 2 NL M 32.0 1 2 7 16 1977 3 NL M 33.0 2 3 7 16 1977 2 DM F 37.0 3 4 7 16 1977 7 DM M 36.0 4 5 7 16 1977 3 DM M 35.0  weight 0 NaN 1 NaN 2 NaN 3 NaN 4 NaN

from pandas import DataFrameCars={'Brand':['Honda Civic','Toyota Corolla','Ford Focus','Audi A4'], 'Price':[22000,25000,27000,35000] }df=DataFrame(Cars,columns=['Brand','Price'])print(df)

Brand Price0 Honda Civic 220001 Toyota Corolla 250002 Ford Focus 270003 Audi A4 35000

import sqlite3conn=sqlite3.connect('TestDB1.db')c=conn.cursor()

c.execute('CREATE TABLE CARS2(Brand text, Price number)')conn.commit()

df.to\_sql('CARS2',conn,if\_exists='replace',index=False)df

Brand Price0 Honda Civic 220001 Toyota Corolla 250002 Ford Focus 270003 Audi A4 35000

c.execute('''SELECT Brand,max(Price) from CARS2''')

<sqlite3.Cursor at 0x7f39bd9e1ce0>

df=DataFrame(c.fetchall(),columns=['Brand','Price'])df

Brand Price0 Audi A4 35000

**Example1**

import pandas as pd import osimport sqlite3 as litefrom sqlalchemy import create\_engine

studentId=["rj101","rj150","rj134","rj70"]SName=["Saurabh","Giftson","Vikas","Radha"]LName=["Chavan","Paul","Bisoi","Rai"]Department=["Bms","Bcom","BscCS","BScIT"]Email=["100rabh@gmail.com","gift01@gmail.com","vik21@gmail.com","rad01@gmail.com"]

studata = zip(studentId,SName,LName,Department,Email)

df = pd.DataFrame(data =studata, columns=['StudentId','SName','LName','Department','Email'])df

StudentId SName LName Department Email0 rj101 Saurabh Chavan Bms 100rabh@gmail.com1 rj150 Giftson Paul Bcom gift01@gmail.com2 rj134 Vikas Bisoi BscCS vik21@gmail.com3 rj70 Radha Rai BScIT rad01@gmail.com

df1=df.to\_csv('studentdata.csv',index=False,header=True)df1

df2=df.to\_excel('studentdata2.xlsx',index=False,header=True)

df2

db\_filename = r'studentdata.db'con = lite.connect(db\_filename)df.to\_sql('student',con,schema=None,if\_exists='replace',index=True,index\_label=None,chunksize=None,dtype=None)con.close()db\_file = r'studentdata.db'engine = create\_engine(r"sqlite:///{}" .format(db\_file))sql = 'SELECT \* from student 'studf = pd.read\_sql(sql, engine)studf

index StudentId SName LName Department Email0 0 rj101 Saurabh Chavan Bms 100rabh@gmail.com1 1 rj150 Giftson Paul Bcom gift01@gmail.com2 2 rj134 Vikas Bisoi BscCS vik21@gmail.com3 3 rj70 Radha Rai BScIT rad01@gmail.com

import numpy as npimport pandas as pd

state=pd.read\_csv("/content/drive/MyDrive/Colab Notebooks/US\_violent\_crime.csv")state.head()

State Murder Assault UrbanPop Rape0 Alabama 13.2 236 58 21.21 Alaska 10.0 263 48 44.52 Arizona 8.1 294 80 31.03 Arkansas 8.8 190 50 19.54 California 9.0 276 91 40.6

**def** some\_func(x): **return** x\*2state.apply(some\_func) *#update each entry of dataframe without any loop*state.apply(**lambda** n: n\*2) *#lambda also works the same*

State Murder Assault UrbanPop Rape0 AlabamaAlabama 26.4 472 116 42.41 AlaskaAlaska 20.0 526 96 89.02 ArizonaArizona 16.2 588 160 62.03 ArkansasArkansas 17.6 380 100 39.04 CaliforniaCalifornia 18.0 552 182 81.25 ColoradoColorado 15.8 408 156 77.46 ConnecticutConnecticut 6.6 220 154 22.27 DelawareDelaware 11.8 476 144 31.68 FloridaFlorida 30.8 670 160 63.89 GeorgiaGeorgia 34.8 422 120 51.610 HawaiiHawaii 10.6 92 166 40.411 IdahoIdaho 5.2 240 108 28.412 IllinoisIllinois 20.8 498 166 48.013 IndianaIndiana 14.4 226 130 42.014 IowaIowa 4.4 112 114 22.615 KansasKansas 12.0 230 132 36.016 KentuckyKentucky 19.4 218 104 32.617 LouisianaLouisiana 30.8 498 132 44.418 MaineMaine 4.2 166 102 15.619 MarylandMaryland 22.6 600 134 55.620 MassachusettsMassachusetts 8.8 298 170 32.621 MichiganMichigan 24.2 510 148 70.222 MinnesotaMinnesota 5.4 144 132 29.823 MississippiMississippi 32.2 518 88 34.224 MissouriMissouri 18.0 356 140 56.425 MontanaMontana 12.0 218 106 32.826 NebraskaNebraska 8.6 204 124 33.027 NevadaNevada 24.4 504 162 92.028 New HampshireNew Hampshire 4.2 114 112 19.029 New JerseyNew Jersey 14.8 318 178 37.630 New MexicoNew Mexico 22.8 570 140 64.231 New YorkNew York 22.2 508 172 52.232 North CarolinaNorth Carolina 26.0 674 90 32.233 North DakotaNorth Dakota 1.6 90 88 14.634 OhioOhio 14.6 240 150 42.835 OklahomaOklahoma 13.2 302 136 40.036 OregonOregon 9.8 318 134 58.637 PennsylvaniaPennsylvania 12.6 212 144 29.838 Rhode IslandRhode Island 6.8 348 174 16.639 South CarolinaSouth Carolina 28.8 558 96 45.040 South DakotaSouth Dakota 7.6 172 90 25.641 TennesseeTennessee 26.4 376 118 53.842 TexasTexas 25.4 402 160 51.043 UtahUtah 6.4 240 160 45.844 VermontVermont 4.4 96 64 22.445 VirginiaVirginia 17.0 312 126 41.446 WashingtonWashington 8.0 290 146 52.447 West VirginiaWest Virginia 11.4 162 78 18.648 WisconsinWisconsin 5.2 106 132 21.649 WyomingWyoming 13.6 322 120 31.2

state.transform(func = **lambda** x : x \* 10)

State Murder Assault \0 AlabamaAlabamaAlabamaAlabamaAlabamaAlabamaAlab... 132.0 2360 1 AlaskaAlaskaAlaskaAlaskaAlaskaAlaskaAlaskaAlas... 100.0 2630 2 ArizonaArizonaArizonaArizonaArizonaArizonaAriz... 81.0 2940 3 ArkansasArkansasArkansasArkansasArkansasArkans... 88.0 1900 4 CaliforniaCaliforniaCaliforniaCaliforniaCalifo... 90.0 2760 5 ColoradoColoradoColoradoColoradoColoradoColora... 79.0 2040 6 ConnecticutConnecticutConnecticutConnecticutCo... 33.0 1100 7 DelawareDelawareDelawareDelawareDelawareDelawa... 59.0 2380 8 FloridaFloridaFloridaFloridaFloridaFloridaFlor... 154.0 3350 9 GeorgiaGeorgiaGeorgiaGeorgiaGeorgiaGeorgiaGeor... 174.0 2110 10 HawaiiHawaiiHawaiiHawaiiHawaiiHawaiiHawaiiHawa... 53.0 460 11 IdahoIdahoIdahoIdahoIdahoIdahoIdahoIdahoIdahoI... 26.0 1200 12 IllinoisIllinoisIllinoisIllinoisIllinoisIllino... 104.0 2490 13 IndianaIndianaIndianaIndianaIndianaIndianaIndi... 72.0 1130 14 IowaIowaIowaIowaIowaIowaIowaIowaIowaIowa 22.0 560 15 KansasKansasKansasKansasKansasKansasKansasKans... 60.0 1150 16 KentuckyKentuckyKentuckyKentuckyKentuckyKentuc... 97.0 1090 17 LouisianaLouisianaLouisianaLouisianaLouisianaL... 154.0 2490 18 MaineMaineMaineMaineMaineMaineMaineMaineMaineM... 21.0 830 19 MarylandMarylandMarylandMarylandMarylandMaryla... 113.0 3000 20 MassachusettsMassachusettsMassachusettsMassach... 44.0 1490 21 MichiganMichiganMichiganMichiganMichiganMichig... 121.0 2550 22 MinnesotaMinnesotaMinnesotaMinnesotaMinnesotaM... 27.0 720 23 MississippiMississippiMississippiMississippiMi... 161.0 2590 24 MissouriMissouriMissouriMissouriMissouriMissou... 90.0 1780 25 MontanaMontanaMontanaMontanaMontanaMontanaMont... 60.0 1090 26 NebraskaNebraskaNebraskaNebraskaNebraskaNebras... 43.0 1020 27 NevadaNevadaNevadaNevadaNevadaNevadaNevadaNeva... 122.0 2520 28 New HampshireNew HampshireNew HampshireNew Ham... 21.0 570 29 New JerseyNew JerseyNew JerseyNew JerseyNew Je... 74.0 1590 30 New MexicoNew MexicoNew MexicoNew MexicoNew Me... 114.0 2850 31 New YorkNew YorkNew YorkNew YorkNew YorkNew Yo... 111.0 2540 32 North CarolinaNorth CarolinaNorth CarolinaNort... 130.0 3370 33 North DakotaNorth DakotaNorth DakotaNorth Dako... 8.0 450 34 OhioOhioOhioOhioOhioOhioOhioOhioOhioOhio 73.0 1200 35 OklahomaOklahomaOklahomaOklahomaOklahomaOklaho... 66.0 1510 36 OregonOregonOregonOregonOregonOregonOregonOreg... 49.0 1590 37 PennsylvaniaPennsylvaniaPennsylvaniaPennsylvan... 63.0 1060 38 Rhode IslandRhode IslandRhode IslandRhode Isla... 34.0 1740 39 South CarolinaSouth CarolinaSouth CarolinaSout... 144.0 2790 40 South DakotaSouth DakotaSouth DakotaSouth Dako... 38.0 860 41 TennesseeTennesseeTennesseeTennesseeTennesseeT... 132.0 1880 42 TexasTexasTexasTexasTexasTexasTexasTexasTexasT... 127.0 2010 43 UtahUtahUtahUtahUtahUtahUtahUtahUtahUtah 32.0 1200 44 VermontVermontVermontVermontVermontVermontVerm... 22.0 480 45 VirginiaVirginiaVirginiaVirginiaVirginiaVirgin... 85.0 1560 46 WashingtonWashingtonWashingtonWashingtonWashin... 40.0 1450 47 West VirginiaWest VirginiaWest VirginiaWest Vi... 57.0 810 48 WisconsinWisconsinWisconsinWisconsinWisconsinW... 26.0 530 49 WyomingWyomingWyomingWyomingWyomingWyomingWyom... 68.0 1610  UrbanPop Rape 0 580 212.0 1 480 445.0 2 800 310.0 3 500 195.0 4 910 406.0 5 780 387.0 6 770 111.0 7 720 158.0 8 800 319.0 9 600 258.0 10 830 202.0 11 540 142.0 12 830 240.0 13 650 210.0 14 570 113.0 15 660 180.0 16 520 163.0 17 660 222.0 18 510 78.0 19 670 278.0 20 850 163.0 21 740 351.0 22 660 149.0 23 440 171.0 24 700 282.0 25 530 164.0 26 620 165.0 27 810 460.0 28 560 95.0 29 890 188.0 30 700 321.0 31 860 261.0 32 450 161.0 33 440 73.0 34 750 214.0 35 680 200.0 36 670 293.0 37 720 149.0 38 870 83.0 39 480 225.0 40 450 128.0 41 590 269.0 42 800 255.0 43 800 229.0 44 320 112.0 45 630 207.0 46 730 262.0 47 390 93.0 48 660 108.0 49 600 156.0

*#usinggroupby*mean\_purchase =state.groupby('State')["Murder"].mean().rename("User\_mean").reset\_index() print(mean\_purchase)

State User\_mean0 Alabama 13.21 Alaska 10.02 Arizona 8.13 Arkansas 8.84 California 9.05 Colorado 7.96 Connecticut 3.37 Delaware 5.98 Florida 15.49 Georgia 17.410 Hawaii 5.311 Idaho 2.612 Illinois 10.413 Indiana 7.214 Iowa 2.215 Kansas 6.016 Kentucky 9.717 Louisiana 15.418 Maine 2.119 Maryland 11.320 Massachusetts 4.421 Michigan 12.122 Minnesota 2.723 Mississippi 16.124 Missouri 9.025 Montana 6.026 Nebraska 4.327 Nevada 12.228 New Hampshire 2.129 New Jersey 7.430 New Mexico 11.431 New York 11.132 North Carolina 13.033 North Dakota 0.834 Ohio 7.335 Oklahoma 6.636 Oregon 4.937 Pennsylvania 6.338 Rhode Island 3.439 South Carolina 14.440 South Dakota 3.841 Tennessee 13.242 Texas 12.743 Utah 3.244 Vermont 2.245 Virginia 8.546 Washington 4.047 West Virginia 5.748 Wisconsin 2.649 Wyoming 6.8

mer=state.merge(mean\_purchase)mer

State Murder Assault UrbanPop Rape User\_mean0 Alabama 13.2 236 58 21.2 13.21 Alaska 10.0 263 48 44.5 10.02 Arizona 8.1 294 80 31.0 8.13 Arkansas 8.8 190 50 19.5 8.84 California 9.0 276 91 40.6 9.05 Colorado 7.9 204 78 38.7 7.96 Connecticut 3.3 110 77 11.1 3.37 Delaware 5.9 238 72 15.8 5.98 Florida 15.4 335 80 31.9 15.49 Georgia 17.4 211 60 25.8 17.410 Hawaii 5.3 46 83 20.2 5.311 Idaho 2.6 120 54 14.2 2.612 Illinois 10.4 249 83 24.0 10.413 Indiana 7.2 113 65 21.0 7.214 Iowa 2.2 56 57 11.3 2.215 Kansas 6.0 115 66 18.0 6.016 Kentucky 9.7 109 52 16.3 9.717 Louisiana 15.4 249 66 22.2 15.418 Maine 2.1 83 51 7.8 2.119 Maryland 11.3 300 67 27.8 11.320 Massachusetts 4.4 149 85 16.3 4.421 Michigan 12.1 255 74 35.1 12.122 Minnesota 2.7 72 66 14.9 2.723 Mississippi 16.1 259 44 17.1 16.124 Missouri 9.0 178 70 28.2 9.025 Montana 6.0 109 53 16.4 6.026 Nebraska 4.3 102 62 16.5 4.327 Nevada 12.2 252 81 46.0 12.228 New Hampshire 2.1 57 56 9.5 2.129 New Jersey 7.4 159 89 18.8 7.430 New Mexico 11.4 285 70 32.1 11.431 New York 11.1 254 86 26.1 11.132 North Carolina 13.0 337 45 16.1 13.033 North Dakota 0.8 45 44 7.3 0.834 Ohio 7.3 120 75 21.4 7.335 Oklahoma 6.6 151 68 20.0 6.636 Oregon 4.9 159 67 29.3 4.937 Pennsylvania 6.3 106 72 14.9 6.338 Rhode Island 3.4 174 87 8.3 3.439 South Carolina 14.4 279 48 22.5 14.440 South Dakota 3.8 86 45 12.8 3.841 Tennessee 13.2 188 59 26.9 13.242 Texas 12.7 201 80 25.5 12.743 Utah 3.2 120 80 22.9 3.244 Vermont 2.2 48 32 11.2 2.245 Virginia 8.5 156 63 20.7 8.546 Washington 4.0 145 73 26.2 4.047 West Virginia 5.7 81 39 9.3 5.748 Wisconsin 2.6 53 66 10.8 2.649 Wyoming 6.8 161 60 15.6 6.8

*#checking for missing values*print(state.isnull().sum())

State 0Murder 0Assault 0UrbanPop 0Rape 0dtype: int64

**EXAMPLE2**

import pandas as pdimport numpy as npcols=['col0', 'col1', 'col2', 'col3', 'col4']rows=['row0', 'row1', 'row2', 'row3', 'row4']data=np.random.randint(0, 100, size=(5,5))df=pd.DataFrame(data, columns=cols, index=rows)df.head()

col0 col1 col2 col3 col4row0 23 19 47 30 65row1 85 4 34 64 33row2 98 14 4 40 11row3 34 12 42 22 28row4 46 52 57 64 9

df.iloc[4,2]

57

df.iloc[3, 3]=0df.iloc[1, 2]=np.nandf.iloc[4, 0]=np.nandf['col5']=0df['col6']=np.nandf.head()

col0 col1 col2 col3 col4 col5 col6row0 23.0 19 47.0 30 65 0 NaNrow1 85.0 4 NaN 64 33 0 NaNrow2 98.0 14 4.0 40 11 0 NaNrow3 34.0 12 42.0 0 28 0 NaNrow4 NaN 52 57.0 64 9 0 NaN

df.loc[:,df.all()]

col0 col1 col2 col4 col6row0 23.0 19 47.0 65 NaNrow1 85.0 4 NaN 33 NaNrow2 98.0 14 4.0 11 NaNrow3 34.0 12 42.0 28 NaNrow4 NaN 52 57.0 9 NaN

df.loc[:,df.any()]

col0 col1 col2 col3 col4row0 23.0 19 47.0 30 65row1 85.0 4 NaN 64 33row2 98.0 14 4.0 40 11row3 34.0 12 42.0 0 28row4 NaN 52 57.0 64 9

df.loc[:,df.isnull().any()]

col0 col2 col6row0 23.0 47.0 NaNrow1 85.0 NaN NaNrow2 98.0 4.0 NaNrow3 34.0 42.0 NaNrow4 NaN 57.0 NaN

df.loc[:,df.notnull().all()]

col1 col3 col4 col5row0 19 30 65 0row1 4 64 33 0row2 14 40 11 0row3 12 0 28 0row4 52 64 9 0

df.dropna(how="all",axis=0)

col0 col1 col2 col3 col4 col5 col6row0 23.0 19 47.0 30 65 0 NaNrow1 85.0 4 NaN 64 33 0 NaNrow2 98.0 14 4.0 40 11 0 NaNrow3 34.0 12 42.0 0 28 0 NaNrow4 NaN 52 57.0 64 9 0 NaN

df.fillna(df.sum())

col0 col1 col2 col3 col4 col5 col6row0 23.0 19 47.0 30 65 0 0.0row1 85.0 4 150.0 64 33 0 0.0row2 98.0 14 4.0 40 11 0 0.0row3 34.0 12 42.0 0 28 0 0.0row4 240.0 52 57.0 64 9 0 0.0

*#Demonstrate transfomr function using pandas in python*import pandas as pdimport numpy as npimport randomdata = pd.DataFrame({ 'C' : [random.choice(('a','b','c')) **for** i **in** range(1000000)], 'A' : [random.randint(1,10) **for** i **in** range(1000000)], 'B' : [random.randint(1,10) **for** i **in** range(1000000)]})data

C A B0 c 4 41 a 7 102 b 2 43 a 10 74 a 8 2... .. .. ..999995 a 1 9999996 a 7 9999997 a 3 4999998 c 5 9999999 a 9 9[1000000 rows x 3 columns]

v=data.groupby('C')["A"].meanv

<bound method GroupBy.mean of <pandas.core.groupby.generic.SeriesGroupBy object at 0x7f39ba052b90>>

mean=data.groupby('C')["A"].mean().rename("D").reset\_index()mean

C D0 a 5.4994081 b 5.4957392 c 5.498086

df\_1=data.merge(mean)df\_1

C A B D0 c 4 4 5.4980861 c 3 4 5.4980862 c 5 10 5.4980863 c 3 3 5.4980864 c 9 6 5.498086... .. .. .. ...999995 b 2 3 5.495739999996 b 3 8 5.495739999997 b 10 10 5.495739999998 b 10 6 5.495739999999 b 10 5 5.495739[1000000 rows x 4 columns]

**PRACTICAL 2-3**

Data visualization allows us to quickly interpret the data and adjust different variables to see their effect

•Technology is increasingly making it easier for us to do so

Why visualize data?

o Observe the patterns

o Identify extreme values that could be anomalies

o Easy interpretation Popular plotting libraries in Python Python offers multiple graphing libraries that offers diverse features

• 1) matplotlib --> to create 2D graphs and plots •

2) pandas visualization --> easy to use interface, built on Matplotlib •

3) seaborn --> provides a high level interface for drawing attractive and informative statistical graphics •

4) ggplot --> based on R’s ggplot2, uses Grammar of Graphics •

5) plotly --> can create interactive plots

Scatter Plot What is a scatter plot? A scatter plot is a set of points that represents the values obtained for two different variables plotted on a horizontal and vertical axes

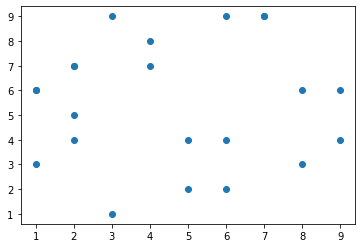
When to use scatter plots?

Scatter plots are used to convey the relationship between two numerical variables

Scatter plots are sometimes called correlation plots because they show how two variables are correlated

import matplotlib.pyplot as plt *# create a figure and axis*fig, ax = plt.subplots()x = [2, 4, 6, 6, 9, 2, 7, 2, 6, 1, 8, 4, 5, 9, 1, 2, 3, 7, 5, 8, 1, 3]y = [7, 8, 2, 4, 6, 4, 9, 5, 9, 3, 6, 7, 2, 4, 6, 7, 1, 9, 4, 3, 6, 9]ax.scatter(x, y)

<matplotlib.collections.PathCollection at 0x7fb57e1d3e10>

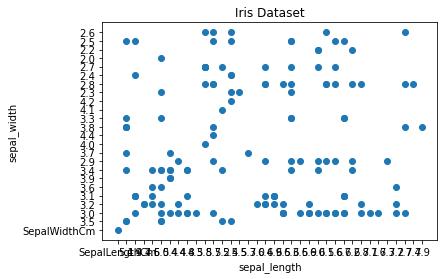


import pandas as pdiris = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/Iris.csv', names=['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width', 'class'])print(iris.head())

sepal\_length sepal\_width petal\_length petal\_width classId SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm Species1 5.1 3.5 1.4 0.2 Iris-setosa2 4.9 3.0 1.4 0.2 Iris-setosa3 4.7 3.2 1.3 0.2 Iris-setosa4 4.6 3.1 1.5 0.2 Iris-setosa

import matplotlib.pyplot as plt *# create a figure and axis*fig, ax = plt.subplots() *# scatter the sepal\_length against the sepal\_width*ax.scatter(iris['sepal\_length'], iris['sepal\_width']) *# set a title and labels*ax.set\_title('Iris Dataset')ax.set\_xlabel('sepal\_length')ax.set\_ylabel('sepal\_width')

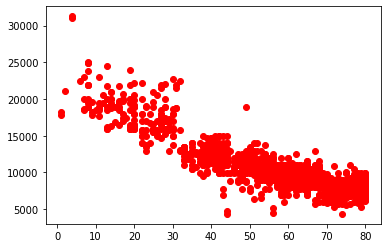
Text(0, 0.5, 'sepal\_width')



import pandas as pdcars\_data=pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/Toyota.csv',index\_col=0)cars\_data.head()

Price Age KM FuelType HP MetColor Automatic CC Doors Weight0 13500 23.0 46986 Diesel 90 1.0 0 2000 three 11651 13750 23.0 72937 Diesel 90 1.0 0 2000 3 11652 13950 24.0 41711 Diesel 90 NaN 0 2000 3 11653 14950 26.0 48000 Diesel 90 0.0 0 2000 3 11654 13750 30.0 38500 Diesel 90 0.0 0 2000 3 1170

import matplotlib.pyplot as pltplt.scatter(cars\_data['Age'],cars\_data['Price'], c='red')plt.show()

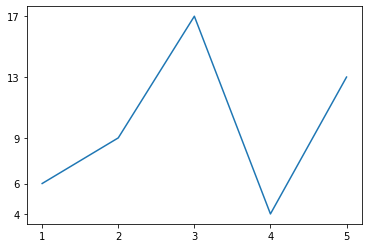


**Line Chart**

In Matplotlib we can create a line chart by calling the plot method. We can also plot multiple columns in one graph, by looping through the columns we want and plotting each column on the same axis.

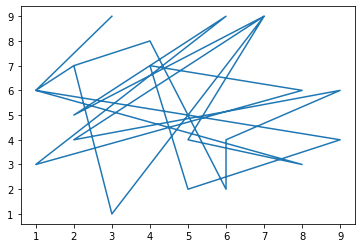
import pandas as pdimport numpy as npfrom matplotlib import pyplot as pltx=range(1,6)y=np.random.randint(1,20,5)plt.plot(x,y)plt.xticks(x)plt.yticks(y)

([<matplotlib.axis.YTick at 0x7fb576e52f90>, <matplotlib.axis.YTick at 0x7fb576e52850>, <matplotlib.axis.YTick at 0x7fb576e99ad0>, <matplotlib.axis.YTick at 0x7fb576e78b10>, <matplotlib.axis.YTick at 0x7fb576e78d90>], <a list of 5 Text major ticklabel objects>)



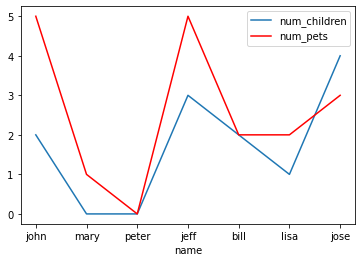
import matplotlib.pyplot as plt *# create a figure and axis*fig, ax = plt.subplots()x = [2, 4, 6, 6, 9, 2, 7, 2, 6, 1, 8, 4, 5, 9, 1, 2, 3, 7, 5, 8, 1, 3]y = [7, 8, 2, 4, 6, 4, 9, 5, 9, 3, 6, 7, 2, 4, 6, 7, 1, 9, 4, 3, 6, 9]ax.plot(x,y)

[<matplotlib.lines.Line2D at 0x7fb5755ca110>]



import pandas as pddf = pd.DataFrame({ 'name':['john','mary','peter','jeff','bill','lisa','jose'], 'age':[23,78,22,19,45,33,20], 'gender':['M','F','M','M','M','F','M'], 'state':['california','dc','california','dc','california','texas','texas'], 'num\_children':[2,0,0,3,2,1,4], 'num\_pets':[5,1,0,5,2,2,3]}) *# From pandas to plot multiple plots on same figure  
# gca stands for 'get current axis'*ax = plt.gca()df.plot(kind='line',x='name',y='num\_children',ax=ax)df.plot(kind='line',x='name',y='num\_pets', color='red',ax=ax)

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fb5755f8f90>

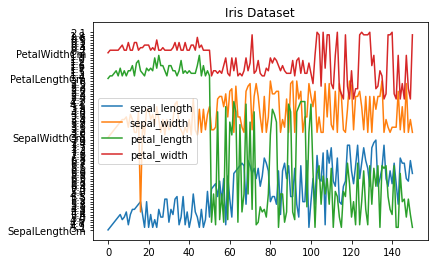


import pandas as pdiris = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/Iris.csv', names=['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width', 'class'])print(iris.head())

sepal\_length sepal\_width petal\_length petal\_width classId SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm Species1 5.1 3.5 1.4 0.2 Iris-setosa2 4.9 3.0 1.4 0.2 Iris-setosa3 4.7 3.2 1.3 0.2 Iris-setosa4 4.6 3.1 1.5 0.2 Iris-setosa

*# get columns to plot*columns = iris.columns.drop(['class']) *# create x data*x\_data = range(0, iris.shape[0]) *# create figure and axis*fig, ax = plt.subplots() *# plot each column***for** column **in** columns: ax.plot(x\_data, iris[column], label=column) *# set title and legend*ax.set\_title('Iris Dataset')ax.legend()

<matplotlib.legend.Legend at 0x7fb575538590>



**Histogram**

In Matplotlib we can create a Histogram using the hist method. If we pass it categorical data like the points column from the wine-review dataset it will automatically calculate how often each class occurs.

*# create figure and axis*fig, ax = plt.subplots() *# plot histogram*ax.hist(iris['sepal\_length']) *# set title and labels*ax.set\_title('iris')ax.set\_xlabel('sepal\_length')ax.set\_ylabel('Frequency')

Text(0, 0.5, 'Frequency')



**Bar Chart**

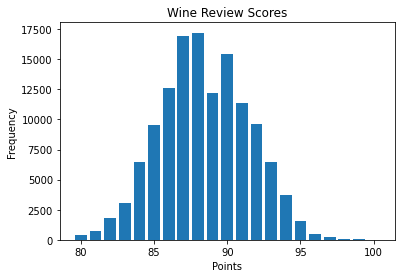
A bar chart can be created using the bar method. The bar-chart isn’t automatically calculating the frequency of a category so we are going to use pandas value\_counts function to do this. The bar-chart is useful for categorical data that doesn’t have a lot of different categories (less than 30) because else it can get quite messy.

wine\_reviews = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/winemag-data-130k-v2.csv', index\_col=0)wine\_reviews.head()

country description \0 Italy Aromas include tropical fruit, broom, brimston... 1 Portugal This is ripe and fruity, a wine that is smooth... 2 US Tart and snappy, the flavors of lime flesh and... 3 US Pineapple rind, lemon pith and orange blossom ... 4 US Much like the regular bottling from 2012, this...  designation points price province \0 Vulkà Bianco 87 NaN Sicily & Sardinia 1 Avidagos 87 15.0 Douro 2 NaN 87 14.0 Oregon 3 Reserve Late Harvest 87 13.0 Michigan 4 Vintner's Reserve Wild Child Block 87 65.0 Oregon  region\_1 region\_2 taster\_name \0 Etna NaN Kerin O’Keefe 1 NaN NaN Roger Voss 2 Willamette Valley Willamette Valley Paul Gregutt 3 Lake Michigan Shore NaN Alexander Peartree 4 Willamette Valley Willamette Valley Paul Gregutt  taster\_twitter\_handle title \0 @kerinokeefe Nicosia 2013 Vulkà Bianco (Etna) 1 @vossroger Quinta dos Avidagos 2011 Avidagos Red (Douro) 2 @paulgwine  Rainstorm 2013 Pinot Gris (Willamette Valley) 3 NaN St. Julian 2013 Reserve Late Harvest Riesling ... 4 @paulgwine  Sweet Cheeks 2012 Vintner's Reserve Wild Child...  variety winery 0 White Blend Nicosia 1 Portuguese Red Quinta dos Avidagos 2 Pinot Gris Rainstorm 3 Riesling St. Julian 4 Pinot Noir Sweet Cheeks

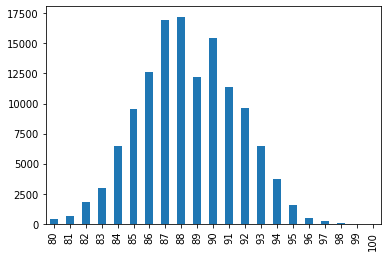
*#Bar Chart  
# create a figure and axis*fig, ax = plt.subplots()  *# count the occurrence of each class*data = wine\_reviews['points'].value\_counts()  *# get x and y data*points = data.index frequency = data.values  *# create bar chart*ax.bar(points, frequency)  *# set title and labels*ax.set\_title('Wine Review Scores') ax.set\_xlabel('Points') ax.set\_ylabel('Frequency')

Text(0, 0.5, 'Frequency')



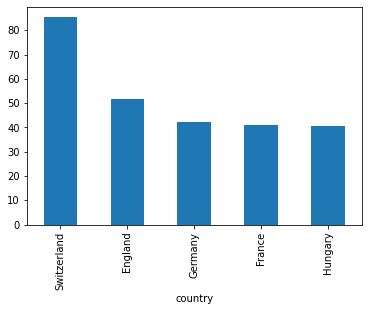
wine\_reviews['points'].value\_counts().sort\_index().plot.bar()

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f629781eb10>



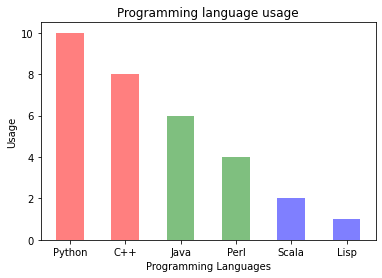
wine\_reviews.groupby("country").price.mean().sort\_values(ascending=False)[:5].plot.bar()

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f629781e290>

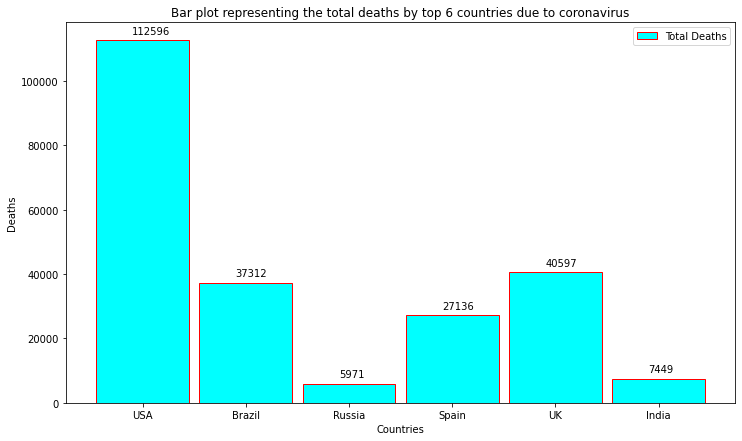


import numpy as npimport matplotlib.pyplot as pltobjects = ('Python', 'C++', 'Java', 'Perl', 'Scala', 'Lisp')y\_pos = np.arange(len(objects))performance = [10,8,6,4,2,1] *# Bar Chart  
# X Axis positions as first parameter list, it can be floating point numbers also  
# Y Values as 2nd parameter list  
# Alpha is transparency,   
# Align can be center or edge  
# Color can be single value or a list of color codes, one for each bar.*plt.bar(y\_pos, performance, width=0.5, align='center', alpha=0.5, color=['r', 'r', 'g', 'g', 'b', 'b']) *# To define labels for x axis values.*plt.xticks(y\_pos, objects)plt.ylabel('Usage')plt.xlabel('Programming Languages')plt.title('Programming language usage')

Text(0.5, 1.0, 'Programming language usage')



*# Importing the matplotlib library*import matplotlib.pyplot as plt *# Declaring the figure or the plot (y, x) or (width, height)*plt.figure(figsize = (12,7)) *# Categorical data: Country names*countries = ['USA', 'Brazil', 'Russia', 'Spain', 'UK', 'India'] *# Integer value interms of death counts*totalDeaths = [112596, 37312, 5971, 27136, 40597, 7449] *# Passing the parameters to the bar function, this is the main function which creates the bar plot*plt.bar(countries, totalDeaths, width= 0.9, align='center',color='cyan', edgecolor = 'red') *# This is the location for the annotated text*i = 1.0j = 2000 *# Annotating the bar plot with the values (total death count)***for** i **in** range(len(countries)): plt.annotate(totalDeaths[i], (-0.1 + i, totalDeaths[i] + j))  *# Creating the legend of the bars in the plot*plt.legend(labels = ['Total Deaths']) *# Giving the tilte for the plot*plt.title("Bar plot representing the total deaths by top 6 countries due to coronavirus") *# Namimg the x and y axis*plt.xlabel('Countries')plt.ylabel('Deaths') *# Saving the plot as a 'png'*plt.savefig('1BarPlot.png') *# Displaying the bar plot*plt.show()

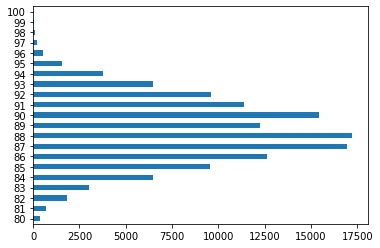


**Horizontal bar plot**

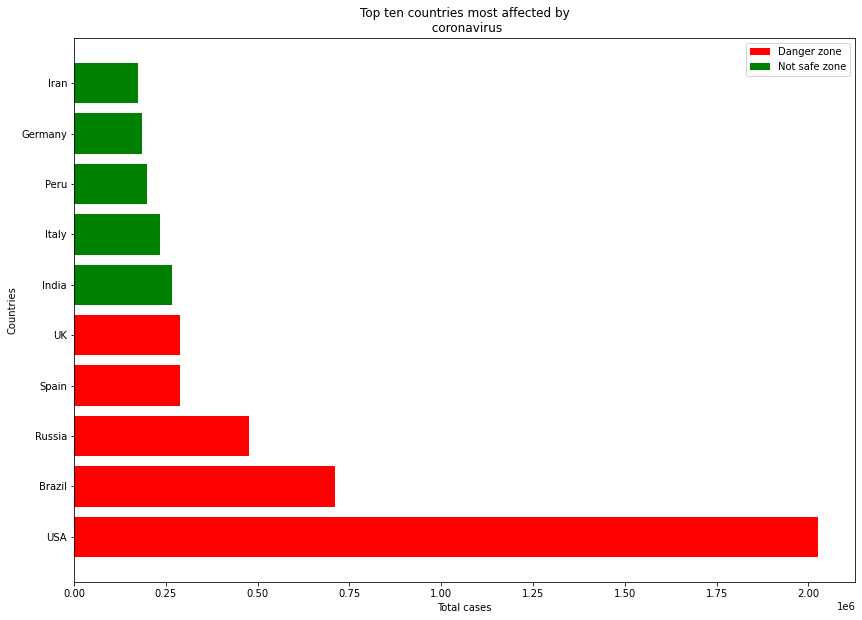
It’s also really simple to make a horizontal bar-chart using the plot.barh() method. By adding one extra character ‘h’, we can align the bars horizontally. Also, we can represent the bars in two or more different colors, this will increase the readability of the plots.

wine\_reviews['points'].value\_counts().sort\_index().plot.barh()

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f62975f6d50>



*# Importing the matplotlib library*import matplotlib.pyplot as plt *# Declaring the figure or the plot (y, x) or (width, height)*plt.figure(figsize=[14, 10]) *# Passing the parameters to the bar function, this is the main function which creates the bar plot  
# For creating the horizontal make sure that you append 'h' to the bar function name*plt.barh(['USA', 'Brazil', 'Russia', 'Spain', 'UK'], [2026493, 710887, 476658, 288797, 287399], label = "Danger zone", color = 'r')plt.barh(['India', 'Italy', 'Peru', 'Germany', 'Iran'], [265928, 235278, 199696, 186205, 173832], label = "Not safe zone", color = 'g') *# Creating the legend of the bars in the plot*plt.legend() *# Namimg the x and y axis*plt.xlabel('Total cases')plt.ylabel('Countries') *# Giving the tilte for the plot*plt.title('Top ten countries most affected by\n coronavirus') *# Saving the plot as a 'png'*plt.savefig('2BarPlot.png') *# Displaying the bar plot*plt.show()

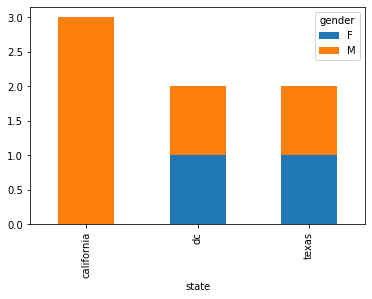


**Stacking two bar plots on top of each other**

At times you might want to stack two or more bar plots on top of each other. With the help of this, you can differentiate two separate quantities visually. To do this just follow.

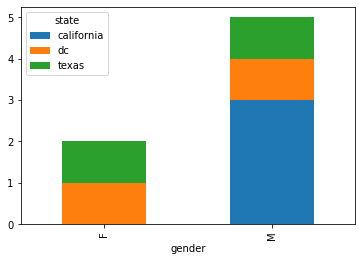
import pandas as pddf = pd.DataFrame({ 'name':['john','mary','peter','jeff','bill','lisa','jose'], 'age':[23,78,22,19,45,33,20], 'gender':['M','F','M','M','M','F','M'], 'state':['california','dc','california','dc','california','texas','texas'], 'num\_children':[2,0,0,3,2,1,4], 'num\_pets':[5,1,0,5,2,2,3]}) *# From pandas to plot multiple plots on same figure*df.groupby(['state','gender']).size().unstack().plot(kind='bar', stacked=True)

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f62974a7a50>

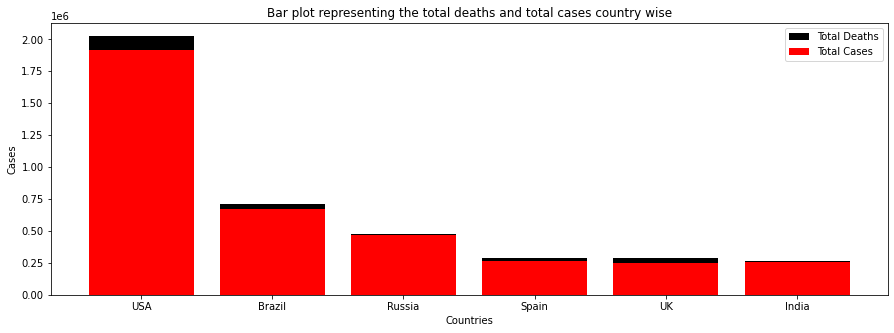


df.groupby(['gender','state']).size().unstack().plot(kind='bar',stacked=True)

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f629748ed10>



*# Importing the matplotlib library*import matplotlib.pyplot as plt *# Declaring the figure or the plot (y, x) or (width, height)*plt.figure(figsize=[15, 5]) *# Categorical data: Country names*countries = ['USA', 'Brazil', 'Russia', 'Spain', 'UK', 'India'] *# Integer value interms of total cases*totalCases = (2026493, 710887, 476658, 288797, 287399, 265928) *# Integer value interms of death counts*totalDeaths = (113055, 37312, 5971, 27136, 40597, 7473) *# Plotting both the total death and the total cases in a single plot. Formula total cases - total deaths***for** i **in** range(len(countries)):  plt.bar(countries[i], totalDeaths[i], bottom = totalCases[i] - totalDeaths[i], color='black') plt.bar(countries[i], totalCases[i] - totalDeaths[i], color='red')  *# Creating the legend of the bars in the plot*plt.legend(labels = ['Total Deaths','Total Cases']) *# Giving the tilte for the plot*plt.title("Bar plot representing the total deaths and total cases country wise") *# Namimg the x and y axis*plt.xlabel('Countries')plt.ylabel('Cases') *# Saving the plot as a 'png'*plt.savefig('3BarPlot.png') *# Displaying the bar plot*plt.show()

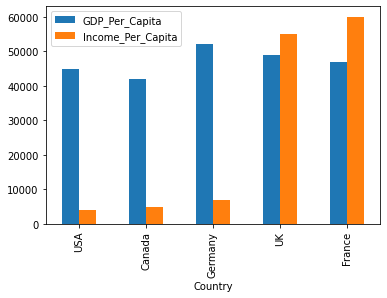


**Plotting two or bar plot next to another (Grouping)**

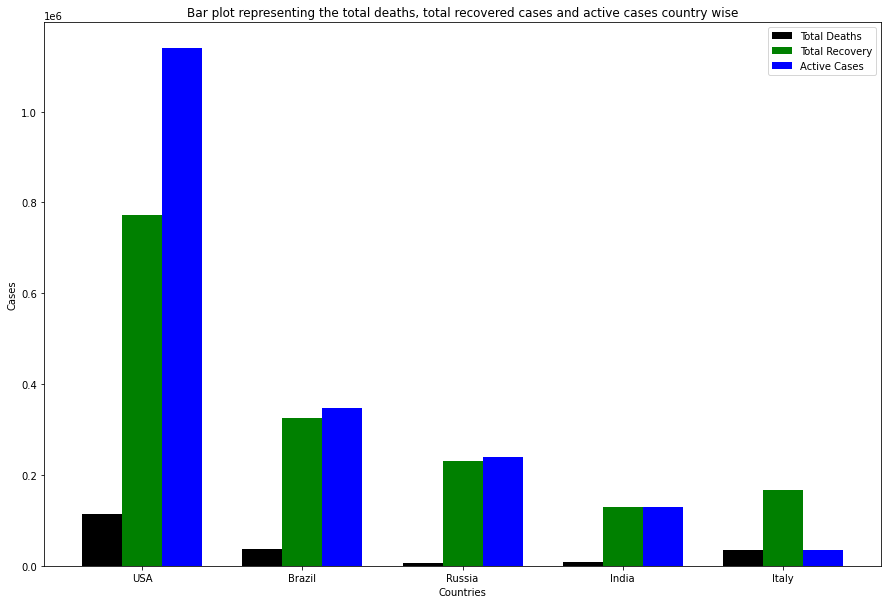
Often many-a-times you might want to group two or more plots just to represent two or more different quantities or whatever. Also in the below code, you can learn to override the name of the x-axis with the name of your choice.

import pandas as pdfrom matplotlib import pyplot as pltData = {'Country': ['USA','Canada','Germany','UK','France'], 'GDP\_Per\_Capita': [45000,42000,52000,49000,47000], 'Income\_Per\_Capita': [4000,5000,7000,55000,60000] } df = pd.DataFrame(Data) *# Multiple metrics in same chart*df.plot(x ='Country', y=['GDP\_Per\_Capita', 'Income\_Per\_Capita'], kind = 'bar')

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f62973d7550>



*# Importing the matplotlib library*import numpy as npimport matplotlib.pyplot as plt *# Declaring the figure or the plot (y, x) or (width, height)*plt.figure(figsize=[15, 10]) *# Data to be plotted*totalDeath = [113055, 37312, 5971, 7473, 33964]totalRecovery = [773480, 325602, 230688, 129095, 166584]activeCases = [1139958, 347973, 239999, 129360, 34730]country = ['USA', 'Brazil', 'Russia', 'India', 'Italy'] *# Using numpy to group 3 different data with bars*X = np.arange(len(totalDeath)) *# Passing the parameters to the bar function, this is the main function which creates the bar plot  
# Using X now to align the bars side by side*plt.bar(X, totalDeath, color = 'black', width = 0.25)plt.bar(X + 0.25, totalRecovery, color = 'g', width = 0.25)plt.bar(X + 0.5, activeCases, color = 'b', width = 0.25) *# Creating the legend of the bars in the plot*plt.legend(['Total Deaths', 'Total Recovery', 'Active Cases']) *# Overiding the x axis with the country names*plt.xticks([i + 0.25 **for** i **in** range(5)], country) *# Giving the tilte for the plot*plt.title("Bar plot representing the total deaths, total recovered cases and active cases country wise") *# Namimg the x and y axis*plt.xlabel('Countries')plt.ylabel('Cases') *# Saving the plot as a 'png'*plt.savefig('4BarPlot.png') *# Displaying the bar plot*plt.show()

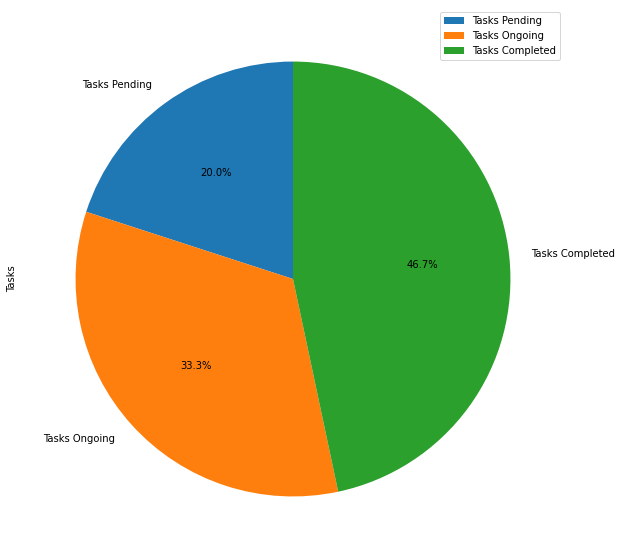


**Pie chart**

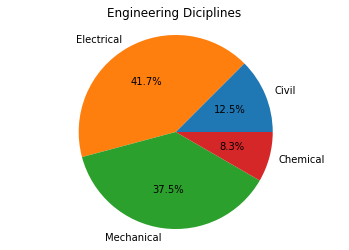
A pie chart is a type of data visualization that is used to illustrate numerical proportions in data.

*# Data Frame plotting*from pandas import DataFrameimport matplotlib.pyplot as pltData = {'Tasks': [300,500,700], 'Task Type' : ['Tasks Pending','Tasks Ongoing','Tasks Completed'] }df = DataFrame(Data)df.set\_index('Task Type', inplace=True) *# autopct has extra % at the end as escape, as % is interpreted as formatting string begin by default.  
# Only pie chart needs labels to be data frame index*df.plot.pie(y='Tasks', figsize=(10,10),autopct='%1.1f%%', startangle=90)

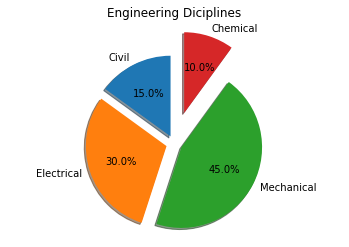
<matplotlib.axes.\_subplots.AxesSubplot at 0x7f6297541150>



import numpy as npimport matplotlib.pyplot as plt *# if using a Jupyter notebook, include:*%matplotlib inline *# Pie chart, where the slices will be ordered and plotted counter-clockwise:*labels = ['Civil', 'Electrical', 'Mechanical', 'Chemical']sizes = [15, 50, 45, 10]fig, ax = plt.subplots()ax.pie(sizes, labels=labels, autopct='%1.1f%%')ax.axis('equal') *# Equal aspect ratio ensures the pie chart is circular.*ax.set\_title('Engineering Diciplines')plt.show()

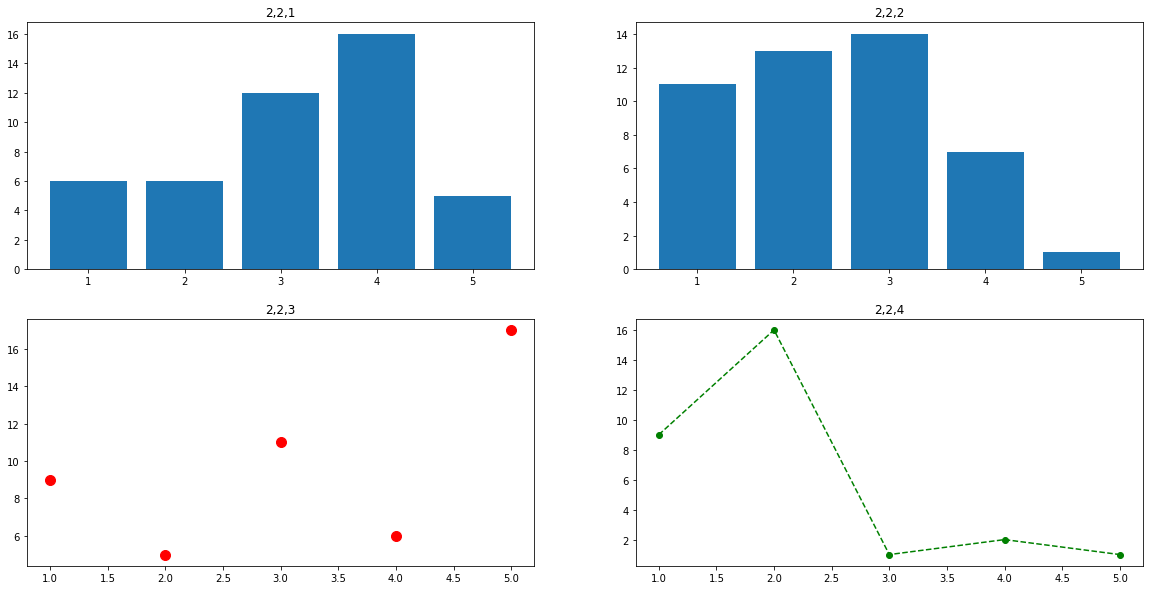


import numpy as npimport matplotlib.pyplot as plt *# if using a Jupyter notebook, include:*%matplotlib inline *# Pie chart, where the slices will be ordered and plotted counter-clockwise*labels = ['Civil', 'Electrical', 'Mechanical', 'Chemical']sizes = [15, 30, 45, 10] *# Explode out the 'Chemical' pie piece by offsetting it a greater amount*explode = (0.1, 0.1, 0.1, 0.4)fig, ax = plt.subplots()ax.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', shadow=True, startangle=90)ax.axis('equal') *# Equal aspect ratio ensures the pie chart is circular.*ax.set\_title('Engineering Diciplines')plt.show()



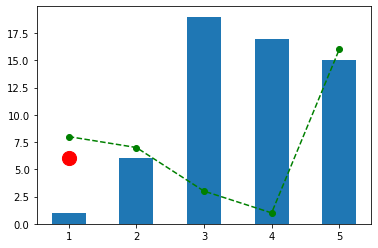
plt.figure(figsize=(20,10))plt.subplot(2,2,1)plt.bar(range(1,6), np.random.randint(1,20,5))plt.title("2,2,1")plt.subplot(2,2,2)plt.bar(range(1,6), np.random.randint(1,20,5))plt.title("2,2,2")plt.subplot(2,2,3) *# s is the size of dot*plt.scatter(range(1,6), np.random.randint(1,20,5), s=100, color="r")plt.title("2,2,3")plt.subplot(2,2,4)plt.plot(range(1,6), np.random.randint(1,20,5), marker='o', color='g', linestyle='--')plt.title("2,2,4")

Text(0.5, 1.0, '2,2,4')



plt.bar(range(1,6), np.random.randint(1,20,5), width=0.5)plt.scatter(range(1,6), np.random.randint(1,20,5), s=200, color="r")plt.plot(range(1,6), np.random.randint(1,20,5), marker='o', color='g', linestyle='--')

[<matplotlib.lines.Line2D at 0x7f6296fa1310>]



**Seaborn**

• Seaborn is a Python data visualization library based on matplotlib • It provides a high level interface for drawing attractive and informative statistical graphics

import pandas as pdimport numpy as npimport matplotlib.pyplot as pltimport seaborn as snsimport os

os.chdir('/content/drive/MyDrive/Colab Notebooks')cars\_data=pd.read\_csv('Toyota.csv',index\_col=0,na\_values=["??","????"])cars\_data.size

14360

cars\_data.dropna(axis=0,inplace=True)cars\_data.size

10960

cars\_data=pd.read\_csv('Toyota.csv')cars\_data.head()

Unnamed: 0 Price Age KM FuelType HP MetColor Automatic CC \0 0 13500 23.0 46986 Diesel 90 1.0 0 2000 1 1 13750 23.0 72937 Diesel 90 1.0 0 2000 2 2 13950 24.0 41711 Diesel 90 NaN 0 2000 3 3 14950 26.0 48000 Diesel 90 0.0 0 2000 4 4 13750 30.0 38500 Diesel 90 0.0 0 2000  Doors Weight 0 three 1165 1 3 1165 2 3 1165 3 3 1165 4 3 1170

cars\_data=pd.read\_csv('Toyota.csv',index\_col=0)cars\_data.head()

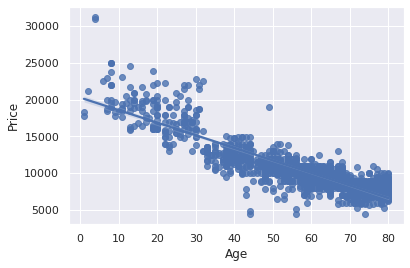
Price Age KM FuelType HP MetColor Automatic CC Doors Weight0 13500 23.0 46986 Diesel 90 1.0 0 2000 three 11651 13750 23.0 72937 Diesel 90 1.0 0 2000 3 11652 13950 24.0 41711 Diesel 90 NaN 0 2000 3 11653 14950 26.0 48000 Diesel 90 0.0 0 2000 3 11654 13750 30.0 38500 Diesel 90 0.0 0 2000 3 1170

**Scatter plot**

Scatter plot of Price vs Age with default arguments

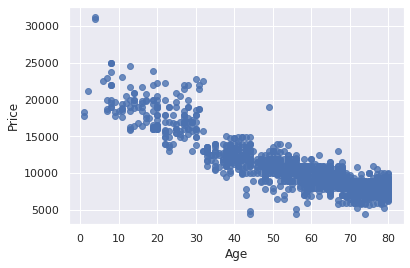
sns.set(style="darkgrid")sns.regplot(x=cars\_data['Age'],y=cars\_data['Price']) *#It estimates and plots a regression model relating the x and y variables*

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f628ae52cd0>



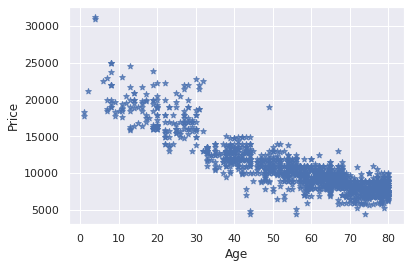
*#Scatter plot of Price vs Age without the regression fit line*sns.regplot(x=cars\_data['Age'],y=cars\_data['Price'],fit\_reg=False)

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f628ad38190>



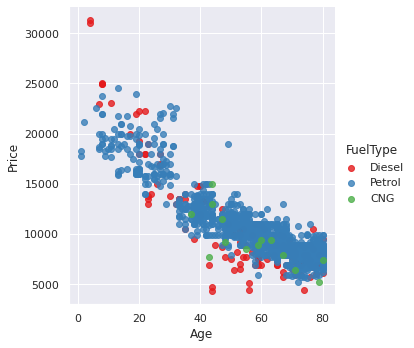
*#Scatter plot of Price vs Age by customizing the appearance of markers*sns.regplot(x=cars\_data['Age'], y=cars\_data['Price'], marker="\*", fit\_reg=False)

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f628ad22310>



*# Scatter plot of Price vs Age by FuelType  
  
#Using hue parameter, including another variable to show the fuel types categories with different colors*sns.lmplot(x='Age', y='Price', data=cars\_data, fit\_reg=False, hue='FuelType', legend=True, palette="Set1")

<seaborn.axisgrid.FacetGrid at 0x7f628ac8d210>



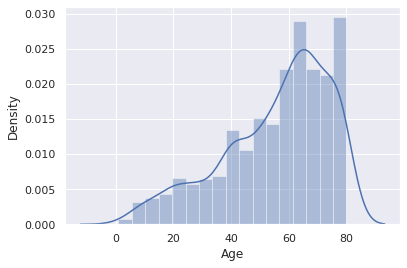
**Histogram**

Histogram with default kernel density estimate

sns.distplot(cars\_data['Age'])

/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms). warnings.warn(msg, FutureWarning)

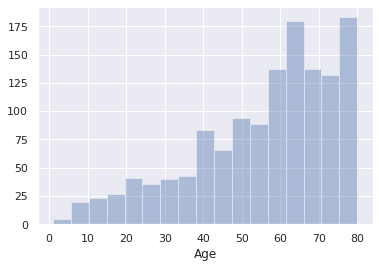
<matplotlib.axes.\_subplots.AxesSubplot at 0x7f62883e5a10>



*#Histogram without kernel density estimate*sns.distplot(cars\_data['Age'],kde=False)

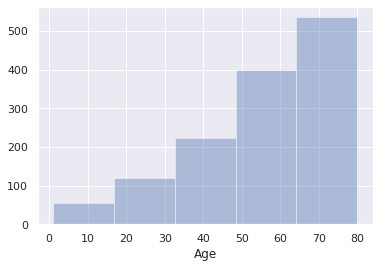
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms). warnings.warn(msg, FutureWarning)

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f62882ffd10>



*#Histogram with fixed no. of bins*sns.distplot(cars\_data['Age'],kde=False, bins=5)

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f62882a6c50>

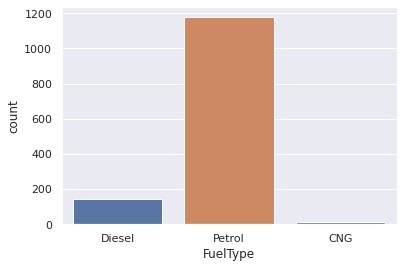


**Bar plot**

Frequency distribution of fuel type of the cars

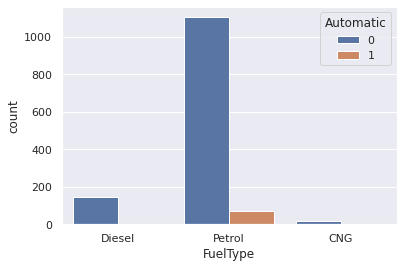
sns.countplot(x="FuelType", data=cars\_data)

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f6288222150>



*###Grouped bar plot  
#Grouped bar plot of FuelType and Automatic*sns.countplot(x="FuelType", data=cars\_data, hue="Automatic")

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f62881968d0>



pd.crosstab(index=cars\_data['Automatic'], columns=cars\_data['FuelType'],dropna=True)

FuelType CNG Diesel PetrolAutomatic 0 15 144 11041 0 0 73

**Box and whiskers plot**

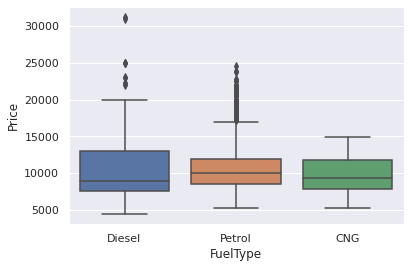
 Box and whiskers plot for numerical vs categorical variable

A Box Plot is also known as Whisker plot is created to display the summary of the set of data values having properties like minimum, first quartile, median, third quartile and maximum. In the box plot, a box is created from the first quartile to the third quartile, a vertical line is also there which goes through the box at the median. Here x-axis denotes the data to be plotted while the y-axis shows the frequency distribution.

 Price of the cars for various fuel types

sns.boxplot(x=cars\_data['FuelType'],y=cars\_data["Price"])

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f6288105bd0>

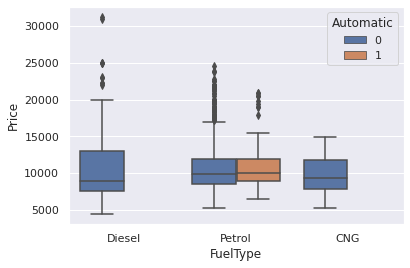


**Grouped box and whiskers plot**

 Grouped box and whiskers plot of Price vs FuelType and Automatic

sns.boxplot(x="FuelType", y=cars\_data["Price"],hue="Automatic",data=cars\_data)

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f62880a7690>



**Box**

whiskers plot and Histogram

 Let’s plot box whiskers plot and histogram on the same window

 Split the plotting window into 2 parts

f,(ax\_box,ax\_hist)=plt.subplots(2,gridspec\_kw={"height\_ratios": (.15, .85)})

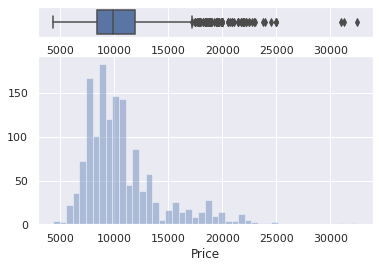
---------------------------------------------------------------------------NameError Traceback (most recent call last)<ipython-input-1-515edc54e648> in <module>----> 1 f,(ax\_box,ax\_hist)=plt.subplots(2,gridspec\_kw={"height\_ratios": (.15, .85)})NameError: name 'plt' is not defined

**Now, add create two plots**

f,(ax\_box,ax\_hist)=plt.subplots(2,gridspec\_kw={"height\_ratios": (.15, .85)})sns.boxplot(cars\_data['Price'],ax=ax\_box)sns.distplot(cars\_data['Price'],ax=ax\_hist,kde=False)

/usr/local/lib/python3.7/dist-packages/seaborn/\_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation. FutureWarning/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms). warnings.warn(msg, FutureWarning)

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f6287f71050>



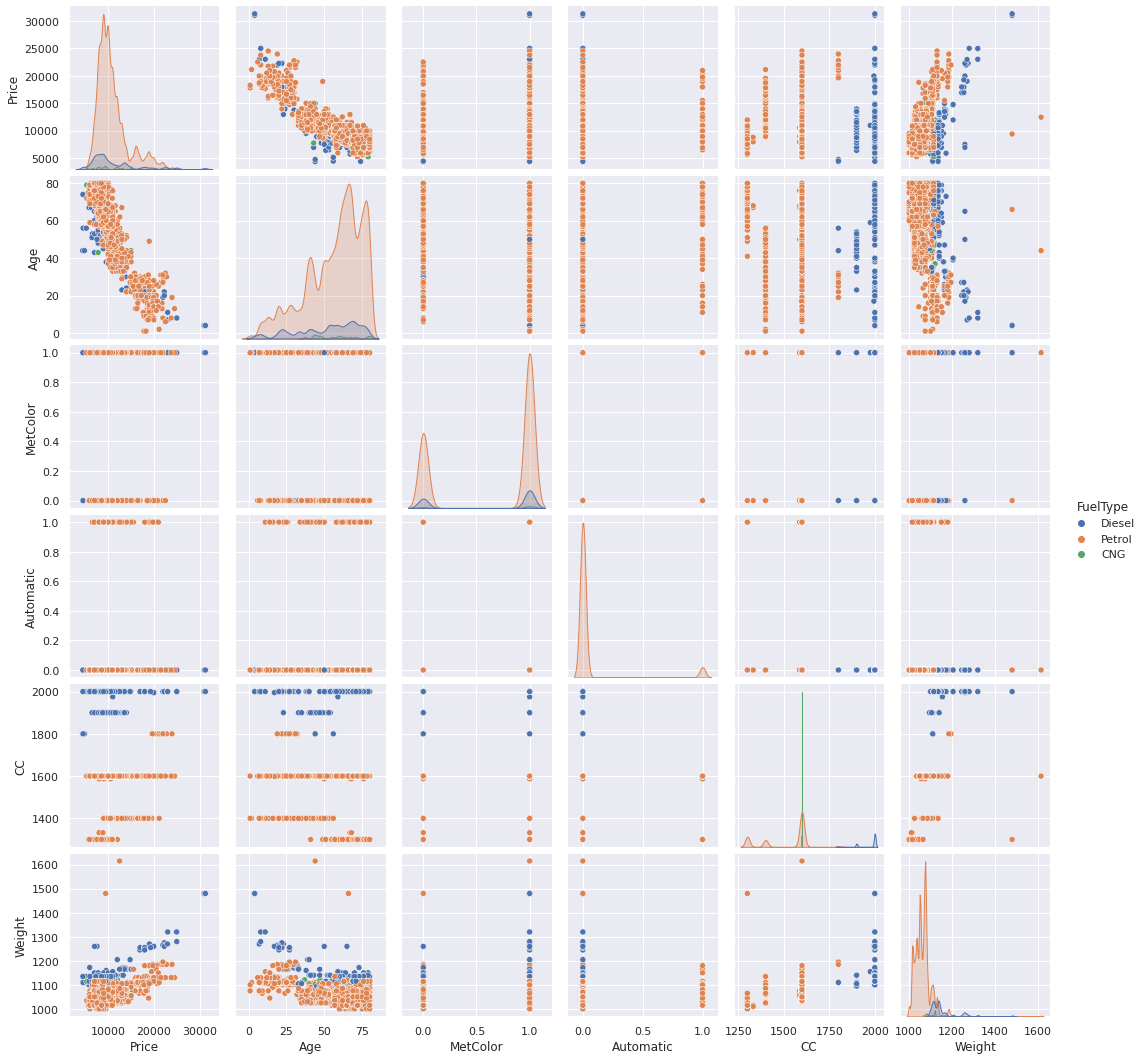
**Pairwise plots**

 It is used to plot pairwise relationships in a dataset

 Creates scatterplots for joint relationships and histograms for univariate distributions

sns.pairplot(cars\_data,kind="scatter",hue="FuelType",diag\_kws={'bw': 0.1})plt.show()

/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:1699: FutureWarning: The `bw` parameter is deprecated in favor of `bw\_method` and `bw\_adjust`. Using 0.1 for `bw\_method`, but please see the docs for the new parameters and update your code. warnings.warn(msg, FutureWarning)/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:1699: FutureWarning: The `bw` parameter is deprecated in favor of `bw\_method` and `bw\_adjust`. Using 0.1 for `bw\_method`, but please see the docs for the new parameters and update your code. warnings.warn(msg, FutureWarning)/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:1699: FutureWarning: The `bw` parameter is deprecated in favor of `bw\_method` and `bw\_adjust`. Using 0.1 for `bw\_method`, but please see the docs for the new parameters and update your code. warnings.warn(msg, FutureWarning)/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:1699: FutureWarning: The `bw` parameter is deprecated in favor of `bw\_method` and `bw\_adjust`. Using 0.1 for `bw\_method`, but please see the docs for the new parameters and update your code. warnings.warn(msg, FutureWarning)/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:1699: FutureWarning: The `bw` parameter is deprecated in favor of `bw\_method` and `bw\_adjust`. Using 0.1 for `bw\_method`, but please see the docs for the new parameters and update your code. warnings.warn(msg, FutureWarning)/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:1699: FutureWarning: The `bw` parameter is deprecated in favor of `bw\_method` and `bw\_adjust`. Using 0.1 for `bw\_method`, but please see the docs for the new parameters and update your code. warnings.warn(msg, FutureWarning)



**Heatmap**

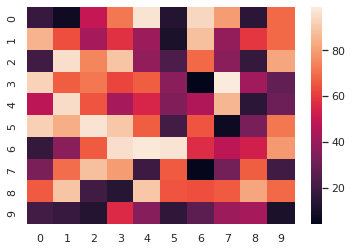
Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. I

import numpy as npimport pandas as pdimport seaborn as snsimport matplotlib.pyplot as pltimport os

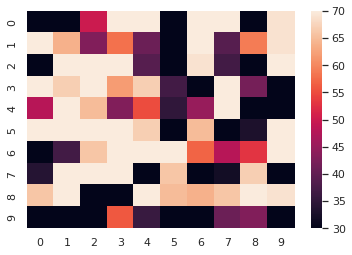
data=np.random.randint(1,100,size=(10,10))print("The data to be plotted: \n")print(data)

The data to be plotted: [[18 8 50 72 97 13 94 80 15 69] [85 64 43 58 41 11 88 39 59 69] [20 96 75 89 39 23 69 37 17 82] [93 67 72 62 67 37 4 99 42 28] [48 95 65 43 55 35 45 86 15 30] [92 84 97 90 67 21 65 7 33 72] [17 37 66 96 98 97 57 48 53 79] [34 70 88 80 19 66 5 32 67 20] [66 89 20 14 90 65 64 66 82 69] [21 18 13 56 36 16 27 41 43 11]]

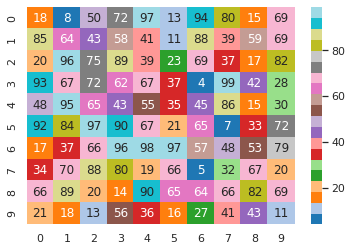
*#Plotting Heatmap*hm=sns.heatmap(data=data)plt.show()



hm = sns.heatmap(data=data, vmin='30', vmax='70')plt.show()



*# setting the parameter values*cmap = "tab20"center = 0  *# setting the parameter values*annot = True *# plotting the heatmap*hm = sns.heatmap(data=data, cmap=cmap, annot=annot)  *# displaying the plotted heatmap*plt.show()



**PRACTICAL 4**

**Probability**

Definition : Probability is a measure of the likelihood of an event to occur. Many events cannot be predicted with total certainty. We can predict only the chance of an event to occur i.e. how likely they are to happen, using it. Probability can range in from 0 to 1, where 0 means the event to be an impossible one and 1 indicates a certain event.The probability of all the events in a sample space adds up to 1.

Formula for Probability

The probability formula is defined as the possibility of an event to happen is equal to the ratio of the number of favourable outcomes and the total number of outcomes.

Probability of event to happen P(E) = Number of favourable outcomes/Total Number of outcomes

Eg 1

*# probability of getting 3 when a die is rolled*  
ns = 6 *#n(S) = {1,2,3,4,5,6}*  
na = 1 *#n(A) = {3}*  
pa = na/ns *# P(A)*  
print("probability of getting 3 is:",pa)

probability of getting 3 is: 0.16666666666666666

*# probability of atleast getting one head when a coin is tossed thrice*  
ns = 8 *#n(S) = {HHH, HHT, HTH, THH, TTH, THT, HTT, TTT}*  
na = 7 *#n(A) = {HHH, HHT, HTH, THH, TTH, THT, HTT}*  
pa = na/ns *# P(A)*  
print("probability of getting atleast one head is:",pa)

probability of getting atleast one head is: 0.875

*# A glass jar contains 5 red, 3 blue and 2 green jelly beans. If a jelly bean is chosen at random from the jar,*   
*# mwhat is the probability that it is not blue?*  
ns = 10 *#n(S) = {5red,3blue,2green}*  
na = 7 *#n(A) = {5red, 2green}*  
pa = na/ns *# P(A)*  
print("probability of getting not blue jellybean is:",pa)

probability of getting not blue jellybean is: 0.7

**Independent and Dependent Events**

If the occurrence of any event is completely unaffected by the occurrence of any other event, such events are known as an independent event in probability and the events which are affected by other events are known as dependent events.

eg.1

*# If the probability that person A will be alive in 20 years*  
*#is 0.7 and the probability that person B will be alive in*  
*# 20 years is 0.5, what is the probability that they will*   
*#both be alive in 20 years?*  
  
*#These are independent events, so*  
P = 0.7\*0.5  
print("probability that they will be alive after 20 years is:",P)

probability that they will be alive after 20 years is: 0.35

**def** event\_probability(n,s):  
 **return** n/s

*#A fair die is tossed twice. Find the probability of getting a 4 or 5 on the first toss and a 1,2, or 3 in the second toss.*  
pa = event\_probability(2,6) *# probability of getting a 4 or 5 on the first toss*  
pb = event\_probability(3,6) *# probability of getting 1,2,3 in second toss*  
P = pa\*pb   
print("probability of getting a 4 or 5 on the first toss and a 1,2, or 3 in the second toss is:",P)

probability of getting a 4 or 5 on the first toss and a 1,2, or 3 in the second toss is: 0.16666666666666666

*# A bag contains 5 white marbles, 3 black marbles and 2 green marbles. In each draw, a marble is drawn from the bag*  
*# and not replaced. In three draws, find the probability of obtaining white, black and green in that order.*  
pw = event\_probability(5,10)  
pb = event\_probability(3,9)  
pg = event\_probability(2,8)  
print("the probability of obtaining white, black and green in that order is ",(pw\*pb\*pg))

the probability of obtaining white, black and green in that order is 0.041666666666666664

*# Sample Space*  
cards = 52  
  
*# Calculate the probability of drawing a heart or a club*  
hearts = 13  
clubs = 13  
heart\_or\_club = event\_probability(hearts, cards) + event\_probability(clubs, cards)  
print(heart\_or\_club )

0.5

*# Calculate the probability of drawing an ace, king, or a queen*  
aces = 4  
kings = 4  
queens = 4  
ace\_king\_or\_queen = event\_probability(aces, cards) + event\_probability(kings, cards) + event\_probability(queens, cards)  
  
print(heart\_or\_club)  
print(ace\_king\_or\_queen)

0.5  
0.23076923076923078

*# Calculate the probability of drawing a heart or an ace*  
hearts = 13  
aces = 4  
ace\_of\_hearts = 1  
heart\_or\_ace = event\_probability(hearts, cards) + event\_probability(aces, cards) - event\_probability(ace\_of\_hearts, cards)  
print(round(heart\_or\_ace, 1))

0.3

red\_cards = 26  
face\_cards = 12  
red\_face\_cards = 6  
red\_or\_face\_cards = event\_probability(red\_cards, cards) + event\_probability(face\_cards, cards) - event\_probability(red\_face\_cards, cards)  
  
print(round(heart\_or\_ace, 1))  
print(round(red\_or\_face\_cards, 1))

0.3  
0.6

**Complementary Events**

For any event E1 there exists another event E1‘ which represents the remaining elements of the sample space S.

E1 = S − E1‘

If a dice is rolled then the sample space S is given as S = {1 , 2 , 3 , 4 , 5 , 6 }. If event E1 represents all the outcomes which is greater than 4, then E1 = {5, 6} and E1‘ = {1, 2, 3, 4}.

Thus E1‘ is the complement of the event E1.

Similarly, the complement of E1, E2, E3……….En will be represented as E1‘, E2‘, E3‘……….En‘

eg.1

*#probabiltiy of not getting 5 when a fair die is rolled*  
ns = 6 *#n(S) = {1,2,3,4,5,6}*  
na = 1 *#n(A) = {5}*  
pa = na/ns *# P(A)*  
print("probabilty of not getting 5 is:",1-pa)

probabilty of not getting 5 is: 0.8333333333333334

**Conditional Probability**

The formula for conditional probability is

P(A|B) = P(A OR B) / P(B).

The parts: P(A|B) = probability of A occurring, given B occurs P(A â© B) = probability of both A and B occurring P(B) = probability of B occurring

Calculate the probability a student gets an A (80%+) in math, given they miss 10 or more classes.

import pandas as pd  
import numpy as np  
df = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/student-mat.csv')  
df.head(3)

school sex age address famsize Pstatus Medu Fedu Mjob Fjob ... \  
0 GP F 18 U GT3 A 4 4 at\_home teacher ...   
1 GP F 17 U GT3 T 1 1 at\_home other ...   
2 GP F 15 U LE3 T 1 1 at\_home other ...   
  
 famrel freetime goout Dalc Walc health absences G1 G2 G3   
0 4 3 4 1 1 3 6 5 6 6   
1 5 3 3 1 1 3 4 5 5 6   
2 4 3 2 2 3 3 10 7 8 10   
  
[3 rows x 33 columns]

len(df)

395

df['grade\_A'] = np.where(df['G3']\*5 >= 80, 1, 0)

df['high\_absenses'] = np.where(df['absences'] >= 10, 1, 0)

df['count'] = 1

df = df[['grade\_A','high\_absenses','count']]  
df.head()

grade\_A high\_absenses count  
0 0 0 1  
1 0 0 1  
2 0 1 1  
3 0 0 1  
4 0 0 1

final= pd.pivot\_table(  
 df,   
 values='count',   
 index=['grade\_A'],   
 columns=['high\_absenses'],   
 aggfunc=np.size,   
 fill\_value=0  
)

print(final)

high\_absenses 0 1  
grade\_A   
0 277 78  
1 35 5

We now have all the data we need to do our calculation. Lets start by calculating each individual part in the formula.

In our case: P(A) is the probability of a grade of 80% or greater. P(B) is the probability of missing 10 or more classes. P(A|B) is the probability of a 80%+ grade, given missing 10 or more classes.

Calculations of parts: P(A) = (35 + 5) / (35 + 5 + 277 + 78) = 0.10126582278481013 P(B) = (78 + 5) / (35 + 5 + 277 + 78) = 0.21012658227848102 P(A OR B) = 5 / (35 + 5 + 277 + 78) = 0.012658227848101266

And per the formula, P(A|B) = P(A Or B) / P(B), put it together.

P(A|B) = 0.012658227848101266/ 0.21012658227848102= 0.06

There we have it. The probability of getting at least an 80% final grade, given missing 10 or more classes is 6%. Conclusion

While the learning from our specific example is clear - go to class if you want good grades

**PRACTICAL 5**

*# for inline plots in jupyter*  
%matplotlib inline  
*# import matplotlib*  
import matplotlib.pyplot as plt  
*# for latex equations*  
from IPython.display import Math, Latex  
*# for displaying images*  
from IPython.core.display import Image  
import numpy as np

*# import seaborn*  
import seaborn as sns  
*# settings for seaborn plotting style*  
sns.set(color\_codes=True)  
*# settings for seaborn plot sizes*  
sns.set(rc={'figure.figsize':(5,5)})

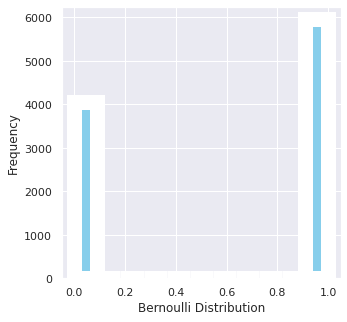
**Bernoulli Distribution**

from scipy.stats import bernoulli  
data\_bern = bernoulli.rvs(size=10000,p=0.6)

ax= sns.distplot(data\_bern,  
 kde=False,  
 color="skyblue",  
 hist\_kws={"linewidth": 15,'alpha':1})  
ax.set(xlabel='Bernoulli Distribution', ylabel='Frequency')

/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)

[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Bernoulli Distribution')]

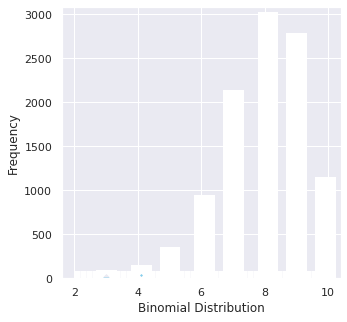


**BINOMINAL DISTRIBUTION**

from scipy.stats import binom  
data\_binom = binom.rvs(n=10,p=0.8,size=10000)

ax = sns.distplot(data\_binom,  
 kde=False,  
 color='skyblue',  
 hist\_kws={"linewidth": 15,'alpha':1})  
ax.set(xlabel='Binomial Distribution', ylabel='Frequency')

[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Binomial Distribution')]



**Poisson Distribution**

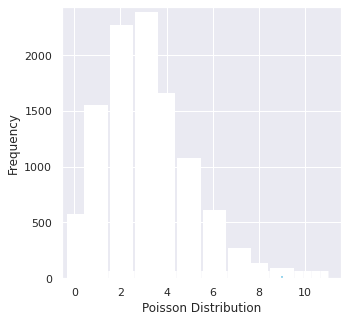
Poisson random variable is typically used to model the number of times an event happened in a time interval

from scipy.stats import poisson  
data\_poisson = poisson.rvs(mu=3, size=10000)

You can generate a poisson distributed discrete random variable using scipy.stats module's poisson.rvs() method which takes μ as a shape parameter and is nothing but the λ in the equation. To shift distribution use the loc parameter. size decides the number of random variates in the distribution. If you want to maintain reproducibility, include a random\_state argument assigned to a number.

ax = sns.distplot(data\_poisson,  
 bins=30,  
 kde=False,  
 color='skyblue',  
 hist\_kws={"linewidth": 15,'alpha':1})  
ax.set(xlabel='Poisson Distribution', ylabel='Frequency')

[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Poisson Distribution')]



**PRACTICAL 6**

**CONTINOUS DISTRIBUTION**

*# for inline plots in jupyter*  
%matplotlib inline  
*# import matplotlib*  
import matplotlib.pyplot as plt  
*# for latex equations*  
from IPython.display import Math, Latex  
*# for displaying images*  
from IPython.core.display import Image  
import numpy as np

*# import seaborn*  
import seaborn as sns  
*# settings for seaborn plotting style*  
sns.set(color\_codes=True)  
*# settings for seaborn plot sizes*  
sns.set(rc={'figure.figsize':(5,5)})

**UNIFORM DISTRIBUTION**

You can visualize uniform distribution in python with the help of a random number generator acting over an interval of numbers (a,b). You need to import the uniform function from scipy.stats module.

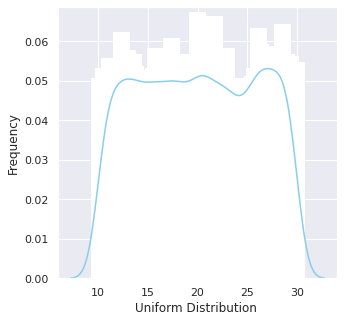
*# import uniform distribution*  
from scipy.stats import uniform

*# random numbers from uniform distribution*  
n = 10000  
start = 10  
width = 20  
data\_uniform = uniform.rvs(size=n, loc = start, scale=width)

ax = sns.distplot(data\_uniform,  
 bins=100,  
 kde=True,  
 color='skyblue',  
 hist\_kws={"linewidth": 15,'alpha':1})  
ax.set(xlabel='Uniform Distribution ', ylabel='Frequency')

/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)

[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Uniform Distribution ')]



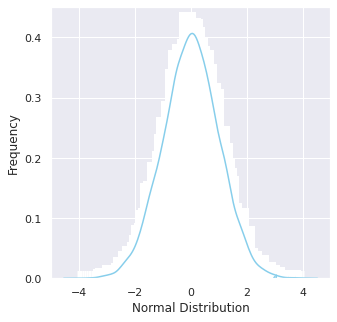
**NORMAL DISTRIBUTION**

from scipy.stats import norm  
*# generate random numbers from N(0,1)*  
data\_normal = norm.rvs(size=10000,loc=0,scale=1)

ax = sns.distplot(data\_normal,  
 bins=100,  
 kde=True,  
 color='skyblue',  
 hist\_kws={"linewidth": 15,'alpha':1})  
ax.set(xlabel='Normal Distribution', ylabel='Frequency')

/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)

[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Normal Distribution')]



**Exponential Distribution**

The exponential distribution describes the time between events in a Poisson point process, i.e., a process in which events occur continuously and independently at a constant average rate. It has a parameter λ

called rate parameter, and its equation is described as :

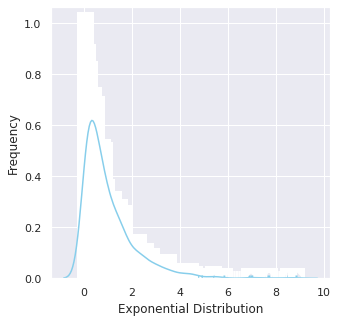
A decreasing exponential distribution looks like :

from scipy.stats import expon  
data\_expon = expon.rvs(scale=1,loc=0,size=1000)

ax = sns.distplot(data\_expon,  
 kde=True,  
 bins=100,  
 color='skyblue',  
 hist\_kws={"linewidth": 15,'alpha':1})  
ax.set(xlabel='Exponential Distribution', ylabel='Frequency')

/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)

[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Exponential Distribution')]



**Chi Square Distribution**

Chi Square distribution is used as a basis to verify the hypothesis.

It has two parameters:

df - (degree of freedom).

size - The shape of the returned array.

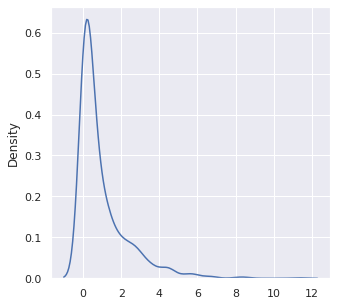
Draw out a sample for chi squared distribution with degree of freedom 2 with size 2x3:

from numpy import random  
  
x = random.chisquare(df=2, size=(2, 3))  
  
print(x)

[[0.04103389 1.57798989 1.85507302]  
 [5.82944896 1.46579974 0.8402198 ]]

from numpy import random  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.distplot(random.chisquare(df=1, size=1000), hist=False)  
  
plt.show()

/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).  
 warnings.warn(msg, FutureWarning)

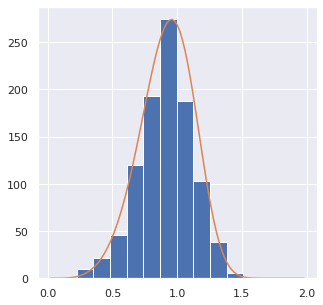


**Weibull Distribution**

a = 5. *# shape*  
  
s = np.random.weibull(a, 1000)

*#Display the histogram of the samples, along with the probability density function:*  
import matplotlib.pyplot as plt  
  
x = np.arange(1,100.)/50.  
  
**def** weib(x,n,a):  
  
 **return** (a / n) \* (x / n)\*\*(a - 1) \* np.exp(-(x / n)\*\*a)

count, bins, ignored = plt.hist(np.random.weibull(5.,1000))  
  
x = np.arange(1,100.)/50.  
  
scale = count.max()/weib(x, 1., 5.).max()  
  
plt.plot(x, weib(x, 1., 5.)\*scale)  
  
plt.show()



**PRACTICAL 7**

import pandas as pd

df=pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/stats.csv')

df

Name Salary Country  
0 Dan 40000 USA  
1 Elizabeth 32000 Brazil  
2 Jon 45000 Italy  
3 Maria 54000 USA  
4 Mark 72000 USA  
5 Bill 62000 Brazil  
6 Jess 92000 Italy  
7 Julia 55000 USA  
8 Jeff 35000 Italy  
9 Ben 48000 Brazil

# Measure of Central Tendancy

# Mean Salary  
mean1=df['Salary'].mean()  
mean1

53500.0

#Sum of Salaries  
sum1=df['Salary'].sum()  
sum1

535000

#Maximum Salary  
max1=df['Salary'].max()  
max1

92000

#Minimum Salary  
min1=df['Salary'].min()  
min1

32000

#Total count  
  
count1=df['Salary'].count()  
count1

10

#Median  
median=df['Salary'].median()  
median

51000.0

#Mode  
mode1=df['Salary'].mode()  
mode1

0 32000  
1 35000  
2 40000  
3 45000  
4 48000  
5 54000  
6 55000  
7 62000  
8 72000  
9 92000  
dtype: int64

countrywise\_sum=df.groupby(['Country'])['Salary'].sum()  
countrywise\_sum

Country  
Brazil 142000  
Italy 172000  
USA 221000  
Name: Salary, dtype: int64

countrywise\_count=df.groupby(['Country']).count()  
countrywise\_count

Name Salary  
Country   
Brazil 3 3  
Italy 3 3  
USA 4 4

# Measure of variability

#variance of salaries  
var1=df['Salary'].var()  
var1

332055555.5555556

#standard deviation  
std1=df['Salary'].std()  
std1

18222.391598128816

# Measure of Symmetry

skew1=df.skew(axis=0, skipna=True)  
skew1

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric\_only=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.  
 """Entry point for launching an IPython kernel.

Salary 1.021551  
dtype: float64

#The skewness is positive so x will have right side tail.  
df.describe()

Salary  
count 10.000000  
mean 53500.000000  
std 18222.391598  
min 32000.000000  
25% 41250.000000  
50% 51000.000000  
75% 60250.000000  
max 92000.000000

**PRACTICAL 8**

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
import scipy.stats as stats  
from scipy.stats import ttest\_1samp  
from statsmodels.stats.power import tt\_ind\_solve\_power

T test A t test is inferntial statistics which is used to determine if there is a significant difference betweenthe means of two groups which may be related in certain features

T-test has 2 types: 1) One sampled t test 2) Two sampled t test

t= (sample mean - population mean) / standard error

ages=[10,20,35,50,28,40,55,18,16,55,30,25,43,18,30,28,14,24,16,17,32,35,26,27,65,18,43,23,21,20,19,70]

ages\_mean=np.mean(ages)  
print(ages\_mean)

30.34375

*#Lets take sample*  
sample\_size=10  
age\_sample=np.random.choice(ages,sample\_size)  
age\_sample

array([28, 16, 16, 43, 35, 27, 24, 10, 18, 16])

from scipy.stats import ttest\_1samp

ttest,p\_value=ttest\_1samp(age\_sample,30)

print(p\_value)

0.0663276542607543

**if** p\_value < 0.05:  
 print("We are rejecting null hypothesis")  
**else**:  
 print("We are accepting null hypothesis")

We are accepting null hypothesis