**COVID-19 VACCINATION ANALYSIS**

Team Members

SS Amesha-961721104012

Gowri Suresh-961721104009

Devichandana AS-961721104006

Nima P-961721104010



Datasets link - <https://www.kaggle.com/gpreda/covid-world-vaccination-progress>

# COVID-19 Vaccination Data Analysis

This project is about "**COVID-19 World Vaccination Progress**" Data Analysis with Python. Collected this Dataset from "Kaggle" which is the world's largest data science community with powerful tools and resources.

COVID-19 Vaccination image

This dataset contains 35310 rows and 15 columns which is really informaive to analysis. In this project,an attempt has been made to analyze various information of COVID-19 World Vaccination Progress such as country, total\_Vaccinations, people\_vaccinated, daily\_vaccinations total\_vaccinations\_per\_hundred, people\_vaccinated\_per\_hundred, people\_fully\_vaccinated\_per\_hundred, vaccines and many more.

Library Used:

* pandas
* matplotlib
* seaborn

## Data Preparation and Cleaning

* Load the dataset into a data frame using Pandas
* Explore the number of rows & columns, ranges of values etc.
* Handle missing, incorrect and invalid data

import pandas as pd

vaccinations\_df = pd.read\_csv('../input/covid-world-vaccination-progress/country\_vaccinations.csv')

vaccinations\_df

country iso\_code date total\_vaccinations \  
0 Afghanistan AFG 2021-02-22 0.0   
1 Afghanistan AFG 2021-02-23 NaN   
2 Afghanistan AFG 2021-02-24 NaN   
3 Afghanistan AFG 2021-02-25 NaN   
4 Afghanistan AFG 2021-02-26 NaN   
... ... ... ... ...   
86507 Zimbabwe ZWE 2022-03-25 8691642.0   
86508 Zimbabwe ZWE 2022-03-26 8791728.0   
86509 Zimbabwe ZWE 2022-03-27 8845039.0   
86510 Zimbabwe ZWE 2022-03-28 8934360.0   
86511 Zimbabwe ZWE 2022-03-29 9039729.0   
  
 people\_vaccinated people\_fully\_vaccinated daily\_vaccinations\_raw \  
0 0.0 NaN NaN   
1 NaN NaN NaN   
2 NaN NaN NaN   
3 NaN NaN NaN   
4 NaN NaN NaN   
... ... ... ...   
86507 4814582.0 3473523.0 139213.0   
86508 4886242.0 3487962.0 100086.0   
86509 4918147.0 3493763.0 53311.0   
86510 4975433.0 3501493.0 89321.0   
86511 5053114.0 3510256.0 105369.0   
  
 daily\_vaccinations total\_vaccinations\_per\_hundred \  
0 NaN 0.00   
1 1367.0 NaN   
2 1367.0 NaN   
3 1367.0 NaN   
4 1367.0 NaN   
... ... ...   
86507 69579.0 57.59   
86508 83429.0 58.25   
86509 90629.0 58.61   
86510 100614.0 59.20   
86511 103751.0 59.90   
  
 people\_vaccinated\_per\_hundred people\_fully\_vaccinated\_per\_hundred \  
0 0.00 NaN   
1 NaN NaN   
2 NaN NaN   
3 NaN NaN   
4 NaN NaN   
... ... ...   
86507 31.90 23.02   
86508 32.38 23.11   
86509 32.59 23.15   
86510 32.97 23.20   
86511 33.48 23.26   
  
 daily\_vaccinations\_per\_million \  
0 NaN   
1 34.0   
2 34.0   
3 34.0   
4 34.0   
... ...   
86507 4610.0   
86508 5528.0   
86509 6005.0   
86510 6667.0   
86511 6874.0   
  
 vaccines \  
0 Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...   
1 Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...   
2 Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...   
3 Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...   
4 Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...   
... ...   
86507 Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...   
86508 Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...   
86509 Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...   
86510 Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...   
86511 Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...   
  
 source\_name \  
0 World Health Organization   
1 World Health Organization   
2 World Health Organization   
3 World Health Organization   
4 World Health Organization   
... ...   
86507 Ministry of Health   
86508 Ministry of Health   
86509 Ministry of Health   
86510 Ministry of Health   
86511 Ministry of Health   
  
 source\_website   
0 https://covid19.who.int/   
1 https://covid19.who.int/   
2 https://covid19.who.int/   
3 https://covid19.who.int/   
4 https://covid19.who.int/   
... ...   
86507 https://www.arcgis.com/home/webmap/viewer.html...   
86508 https://www.arcgis.com/home/webmap/viewer.html...   
86509 https://www.arcgis.com/home/webmap/viewer.html...   
86510 https://www.arcgis.com/home/webmap/viewer.html...   
86511 https://www.arcgis.com/home/webmap/viewer.html...

vaccinations\_df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 86512 entries, 0 to 86511  
Data columns (total 15 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 country 86512 non-null object   
 1 iso\_code 86512 non-null object   
 2 date 86512 non-null object   
 3 total\_vaccinations 43607 non-null float64  
 4 people\_vaccinated 41294 non-null float64  
 5 people\_fully\_vaccinated 38802 non-null float64  
 6 daily\_vaccinations\_raw 35362 non-null float64  
 7 daily\_vaccinations 86213 non-null float64  
 8 total\_vaccinations\_per\_hundred 43607 non-null float64  
 9 people\_vaccinated\_per\_hundred 41294 non-null float64  
 10 people\_fully\_vaccinated\_per\_hundred 38802 non-null float64  
 11 daily\_vaccinations\_per\_million 86213 non-null float64  
 12 vaccines 86512 non-null object   
 13 source\_name 86512 non-null object   
 14 source\_website 86512 non-null object   
dtypes: float64(9), object(6)  
memory usage: 9.9+ MB

vaccinations\_df.columns

Index(['country', 'iso\_code', 'date', 'total\_vaccinations',  
 'people\_vaccinated', 'people\_fully\_vaccinated',  
 'daily\_vaccinations\_raw', 'daily\_vaccinations',  
 'total\_vaccinations\_per\_hundred', 'people\_vaccinated\_per\_hundred',  
 'people\_fully\_vaccinated\_per\_hundred', 'daily\_vaccinations\_per\_million',  
 'vaccines', 'source\_name', 'source\_website'],  
 dtype='object')

vaccinations\_df.shape

(86512, 15)

vaccinations\_df.describe()

total\_vaccinations people\_vaccinated people\_fully\_vaccinated \  
count 4.360700e+04 4.129400e+04 3.880200e+04   
mean 4.592964e+07 1.770508e+07 1.413830e+07   
std 2.246004e+08 7.078731e+07 5.713920e+07   
min 0.000000e+00 0.000000e+00 1.000000e+00   
25% 5.264100e+05 3.494642e+05 2.439622e+05   
50% 3.590096e+06 2.187310e+06 1.722140e+06   
75% 1.701230e+07 9.152520e+06 7.559870e+06   
max 3.263129e+09 1.275541e+09 1.240777e+09   
  
 daily\_vaccinations\_raw daily\_vaccinations \  
count 3.536200e+04 8.621300e+04   
mean 2.705996e+05 1.313055e+05   
std 1.212427e+06 7.682388e+05   
min 0.000000e+00 0.000000e+00   
25% 4.668000e+03 9.000000e+02   
50% 2.530900e+04 7.343000e+03   
75% 1.234925e+05 4.409800e+04   
max 2.474100e+07 2.242429e+07   
  
 total\_vaccinations\_per\_hundred people\_vaccinated\_per\_hundred \  
count 43607.000000 41294.000000   
mean 80.188543 40.927317   
std 67.913577 29.290759   
min 0.000000 0.000000   
25% 16.050000 11.370000   
50% 67.520000 41.435000   
75% 132.735000 67.910000   
max 345.370000 124.760000   
  
 people\_fully\_vaccinated\_per\_hundred daily\_vaccinations\_per\_million   
count 38802.000000 86213.000000   
mean 35.523243 3257.049157   
std 28.376252 3934.312440   
min 0.000000 0.000000   
25% 7.020000 636.000000   
50% 31.750000 2050.000000   
75% 62.080000 4682.000000   
max 122.370000 117497.000000

vaccinations\_df.isnull().sum()

country 0  
iso\_code 0  
date 0  
total\_vaccinations 42905  
people\_vaccinated 45218  
people\_fully\_vaccinated 47710  
daily\_vaccinations\_raw 51150  
daily\_vaccinations 299  
total\_vaccinations\_per\_hundred 42905  
people\_vaccinated\_per\_hundred 45218  
people\_fully\_vaccinated\_per\_hundred 47710  
daily\_vaccinations\_per\_million 299  
vaccines 0  
source\_name 0  
source\_website 0  
dtype: int64

vaccinations\_df.fillna(value=0, inplace=True)  
date = vaccinations\_df.date.str.split('-', expand=True)  
date

0 1 2  
0 2021 02 22  
1 2021 02 23  
2 2021 02 24  
3 2021 02 25  
4 2021 02 26  
... ... .. ..  
86507 2022 03 25  
86508 2022 03 26  
86509 2022 03 27  
86510 2022 03 28  
86511 2022 03 29  
  
[86512 rows x 3 columns]

vaccinations\_df['year'] = date[0]  
vaccinations\_df['month'] = date[1]  
vaccinations\_df['day'] = date[2]  
  
vaccinations\_df.year = pd.to\_numeric(vaccinations\_df.year)  
vaccinations\_df.month = pd.to\_numeric(vaccinations\_df.month)  
vaccinations\_df.day = pd.to\_numeric(vaccinations\_df.day)  
  
vaccinations\_df.date = pd.to\_datetime(vaccinations\_df.date)  
  
vaccinations\_df.head()

country iso\_code date total\_vaccinations people\_vaccinated \  
0 Afghanistan AFG 2021-02-22 0.0 0.0   
1 Afghanistan AFG 2021-02-23 0.0 0.0   
2 Afghanistan AFG 2021-02-24 0.0 0.0   
3 Afghanistan AFG 2021-02-25 0.0 0.0   
4 Afghanistan AFG 2021-02-26 0.0 0.0   
  
 people\_fully\_vaccinated daily\_vaccinations\_raw daily\_vaccinations \  
0 0.0 0.0 0.0   
1 0.0 0.0 1367.0   
2 0.0 0.0 1367.0   
3 0.0 0.0 1367.0   
4 0.0 0.0 1367.0   
  
 total\_vaccinations\_per\_hundred people\_vaccinated\_per\_hundred \  
0 0.0 0.0   
1 0.0 0.0   
2 0.0 0.0   
3 0.0 0.0   
4 0.0 0.0   
  
 people\_fully\_vaccinated\_per\_hundred daily\_vaccinations\_per\_million \  
0 0.0 0.0   
1 0.0 34.0   
2 0.0 34.0   
3 0.0 34.0   
4 0.0 34.0   
  
 vaccines \  
0 Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...   
1 Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...   
2 Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...   
3 Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...   
4 Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...   
  
 source\_name source\_website year month day   
0 World Health Organization https://covid19.who.int/ 2021 2 22   
1 World Health Organization https://covid19.who.int/ 2021 2 23   
2 World Health Organization https://covid19.who.int/ 2021 2 24   
3 World Health Organization https://covid19.who.int/ 2021 2 25   
4 World Health Organization https://covid19.who.int/ 2021 2 26

vaccinations\_df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 86512 entries, 0 to 86511  
Data columns (total 18 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 country 86512 non-null object   
 1 iso\_code 86512 non-null object   
 2 date 86512 non-null datetime64[ns]  
 3 total\_vaccinations 86512 non-null float64   
 4 people\_vaccinated 86512 non-null float64   
 5 people\_fully\_vaccinated 86512 non-null float64   
 6 daily\_vaccinations\_raw 86512 non-null float64   
 7 daily\_vaccinations 86512 non-null float64   
 8 total\_vaccinations\_per\_hundred 86512 non-null float64   
 9 people\_vaccinated\_per\_hundred 86512 non-null float64   
 10 people\_fully\_vaccinated\_per\_hundred 86512 non-null float64   
 11 daily\_vaccinations\_per\_million 86512 non-null float64   
 12 vaccines 86512 non-null object   
 13 source\_name 86512 non-null object   
 14 source\_website 86512 non-null object   
 15 year 86512 non-null int64   
 16 month 86512 non-null int64   
 17 day 86512 non-null int64   
dtypes: datetime64[ns](1), float64(9), int64(3), object(5)  
memory usage: 11.9+ MB

## Exploratory Analysis and Visualization

Let's begin by importingmatplotlib.pyplot and seaborn.

import seaborn as sns  
import matplotlib  
import matplotlib.pyplot as plt  
%matplotlib inline  
  
sns.set\_style('darkgrid')  
matplotlib.rcParams['font.size'] = 14  
matplotlib.rcParams['figure.figsize'] = (9, 5)  
matplotlib.rcParams['figure.facecolor'] = '#00000000'

Explore the mean, min, max

vaccinations\_df.mean()

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:1: FutureWarning: DataFrame.mean and DataFrame.median with numeric\_only=None will include datetime64 and datetime64tz columns in a future version.  
 """Entry point for launching an IPython kernel.

total\_vaccinations 2.315117e+07  
people\_vaccinated 8.451007e+06  
people\_fully\_vaccinated 6.341251e+06  
daily\_vaccinations\_raw 1.106083e+05  
daily\_vaccinations 1.308517e+05  
total\_vaccinations\_per\_hundred 4.041962e+01  
people\_vaccinated\_per\_hundred 1.953547e+01  
people\_fully\_vaccinated\_per\_hundred 1.593274e+01  
daily\_vaccinations\_per\_million 3.245792e+03  
year 2.021199e+03  
month 6.165711e+00  
day 1.571936e+01  
dtype: float64

vaccinations\_df.min()

country Afghanistan  
iso\_code ABW  
date 2020-12-02 00:00:00  
total\_vaccinations 0.0  
people\_vaccinated 0.0  
people\_fully\_vaccinated 0.0  
daily\_vaccinations\_raw 0.0  
daily\_vaccinations 0.0  
total\_vaccinations\_per\_hundred 0.0  
people\_vaccinated\_per\_hundred 0.0  
people\_fully\_vaccinated\_per\_hundred 0.0  
daily\_vaccinations\_per\_million 0.0  
vaccines Abdala, Johnson&Johnson, Oxford/AstraZeneca, P...  
source\_name Africa Centres for Disease Control and Prevention  
source\_website http://103.247.238.92/webportal/pages/covid19-...  
year 2020  
month 1  
day 1  
dtype: object

vaccinations\_df.max()

country Zimbabwe  
iso\_code ZWE  
date 2022-03-29 00:00:00  
total\_vaccinations 3263129000.0  
people\_vaccinated 1275541000.0  
people\_fully\_vaccinated 1240777000.0  
daily\_vaccinations\_raw 24741000.0  
daily\_vaccinations 22424286.0  
total\_vaccinations\_per\_hundred 345.37  
people\_vaccinated\_per\_hundred 124.76  
people\_fully\_vaccinated\_per\_hundred 122.37  
daily\_vaccinations\_per\_million 117497.0  
vaccines Sinopharm/Beijing, Sputnik V  
source\_name World Health Organization  
source\_website https://www.ssm.gov.mo/docs/19164/19164\_dd2dfe...  
year 2022  
month 12  
day 31  
dtype: object

Explore the country Coloum

vaccinations\_df.country.value\_counts()

Norway 482  
Latvia 480  
Denmark 476  
United States 471  
Canada 470  
 ...   
Bonaire Sint Eustatius and Saba 146  
Tokelau 114  
Saint Helena 92  
Pitcairn 85  
Falkland Islands 67  
Name: country, Length: 223, dtype: int64

vaccinations\_df.country

0 Afghanistan  
1 Afghanistan  
2 Afghanistan  
3 Afghanistan  
4 Afghanistan  
 ...   
86507 Zimbabwe  
86508 Zimbabwe  
86509 Zimbabwe  
86510 Zimbabwe  
86511 Zimbabwe  
Name: country, Length: 86512, dtype: object

vaccinations\_df.country.nunique()

223

Explore the min and max of fully vacnated people.

vaccinations\_df.people\_fully\_vaccinated.min()

0.0

vaccinations\_df.people\_fully\_vaccinated.max()

1240777000.0

Explore the min and max date.

vaccinations\_df.date.min()

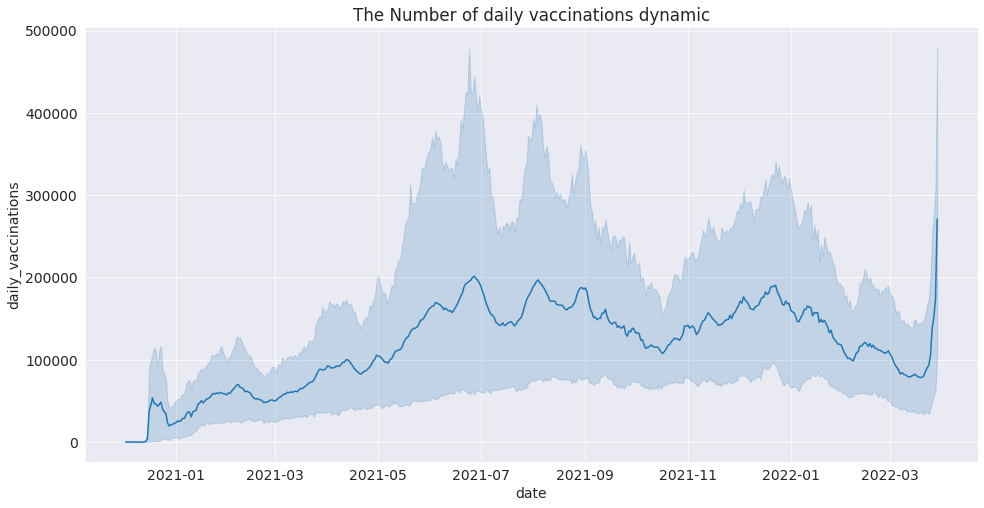
Timestamp('2020-12-02 00:00:00')

vaccinations\_df.date.max()

Timestamp('2022-03-29 00:00:00')

Explore The Number of daily vaccinations dynamic

plt.figure(figsize=(16,8))  
sns.lineplot(x=vaccinations\_df.date, y=vaccinations\_df.daily\_vaccinations)  
plt.title('The Number of daily vaccinations dynamic')  
plt.show()



Explore the Vaccination procedure go on rapidly from which date.

countries = vaccinations\_df.groupby('country')['total\_vaccinations'].max().sort\_values(ascending= False)[:5].index  
  
top\_countries = pd.DataFrame(columns= vaccinations\_df.columns)  
for country in countries:  
 top\_countries = top\_countries.append(vaccinations\_df.loc[vaccinations\_df['country'] == country])

plt.figure(figsize=(20,8))  
sns.lineplot(top\_countries['date'], top\_countries['daily\_vaccinations\_per\_million'], hue= top\_countries['country'], ci= False)  
plt.title('Vaccination procedure go on rapidly');

/opt/conda/lib/python3.7/site-packages/seaborn/\_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. 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



## Asking and Answering Questions

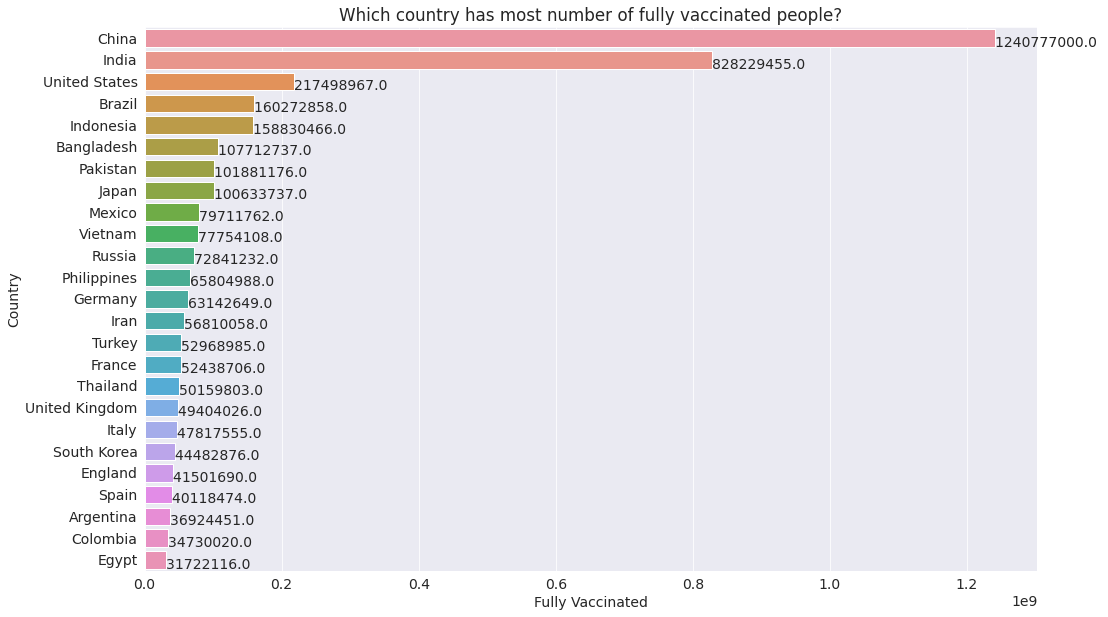
#### Q1: Which country has most number of fully vaccinated people?

fully\_vaccinated = vaccinations\_df.groupby("country")["people\_fully\_vaccinated"].max().sort\_values(ascending= False).head(25)

fully\_vaccinated.reset\_index()

country people\_fully\_vaccinated  
0 China 1.240777e+09  
1 India 8.282295e+08  
2 United States 2.174990e+08  
3 Brazil 1.602729e+08  
4 Indonesia 1.588305e+08  
5 Bangladesh 1.077127e+08  
6 Pakistan 1.018812e+08  
7 Japan 1.006337e+08  
8 Mexico 7.971176e+07  
9 Vietnam 7.775411e+07  
10 Russia 7.284123e+07  
11 Philippines 6.580499e+07  
12 Germany 6.314265e+07  
13 Iran 5.681006e+07  
14 Turkey 5.296898e+07  
15 France 5.243871e+07  
16 Thailand 5.015980e+07  
17 United Kingdom 4.940403e+07  
18 Italy 4.781756e+07  
19 South Korea 4.448288e+07  
20 England 4.150169e+07  
21 Spain 4.011847e+07  
22 Argentina 3.692445e+07  
23 Colombia 3.473002e+07  
24 Egypt 3.172212e+07

plt.figure(figsize=(16,10))  
ax = sns.barplot(x=fully\_vaccinated, y=fully\_vaccinated.index)  
plt.xlabel("Fully Vaccinated")  
plt.ylabel("Country");  
plt.title('Which country has most number of fully vaccinated people?');  
  
for patch in ax.patches:  
 width = patch.get\_width()  
 height = patch.get\_height()  
 x = patch.get\_x()  
 y = patch.get\_y()  
   
 plt.text(width + x, height + y, '{:.1f} '.format(width))



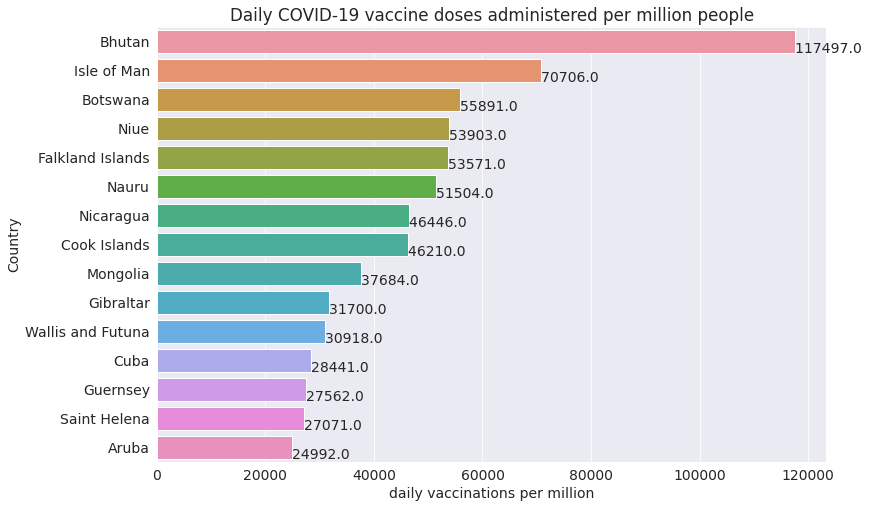
#### Q2: Daily COVID-19 vaccine doses administered per million people.

daily\_vaccinations\_per\_million = vaccinations\_df.groupby("country")["daily\_vaccinations\_per\_million"].max().sort\_values(ascending= False).head(15)

daily\_vaccinations\_per\_million.reset\_index()

country daily\_vaccinations\_per\_million  
0 Bhutan 117497.0  
1 Isle of Man 70706.0  
2 Botswana 55891.0  
3 Niue 53903.0  
4 Falkland Islands 53571.0  
5 Nauru 51504.0  
6 Nicaragua 46446.0  
7 Cook Islands 46210.0  
8 Mongolia 37684.0  
9 Gibraltar 31700.0  
10 Wallis and Futuna 30918.0  
11 Cuba 28441.0  
12 Guernsey 27562.0  
13 Saint Helena 27071.0  
14 Aruba 24992.0

plt.figure(figsize=(12,8))  
ax = sns.barplot(x=daily\_vaccinations\_per\_million, y=daily\_vaccinations\_per\_million.index )  
plt.xlabel("daily vaccinations per million")  
plt.ylabel("Country")  
plt.title("Daily COVID-19 vaccine doses administered per million people");  
  
for patch in ax.patches:  
 width = patch.get\_width()  
 height = patch.get\_height()  
 x = patch.get\_x()  
 y = patch.get\_y()  
   
 plt.text(width + x, height + y, '{:.1f} '.format(width))



#### Q3: How many people daily vaccinated in Bangladesh?

bangladesh\_df = vaccinations\_df[vaccinations\_df['country'] == 'Bangladesh']  
bangladesh\_df

country iso\_code date total\_vaccinations people\_vaccinated \  
6133 Bangladesh BGD 2021-01-26 0.0 0.0   
6134 Bangladesh BGD 2021-01-27 26.0 26.0   
6135 Bangladesh BGD 2021-01-28 567.0 567.0   
6136 Bangladesh BGD 2021-01-29 0.0 0.0   
6137 Bangladesh BGD 2021-01-30 0.0 0.0   
... ... ... ... ... ...   
6556 Bangladesh BGD 2022-03-25 0.0 0.0   
6557 Bangladesh BGD 2022-03-26 229789298.0 127084404.0   
6558 Bangladesh BGD 2022-03-27 231420654.0 127169172.0   
6559 Bangladesh BGD 2022-03-28 238459012.0 127365973.0   
6560 Bangladesh BGD 2022-03-29 243642749.0 127544055.0   
  
 people\_fully\_vaccinated daily\_vaccinations\_raw daily\_vaccinations \  
6133 0.0 0.0 0.0   
6134 0.0 26.0 26.0   
6135 0.0 541.0 284.0   
6136 0.0 0.0 1209.0   
6137 0.0 0.0 1671.0   
... ... ... ...   
6556 0.0 0.0 911880.0   
6557 95424031.0 0.0 918926.0   
6558 96671169.0 1631356.0 1003343.0   
6559 103130478.0 7038358.0 1854299.0   
6560 107712737.0 5183737.0 2411925.0   
  
 total\_vaccinations\_per\_hundred people\_vaccinated\_per\_hundred \  
6133 0.00 0.00   
6134 0.00 0.00   
6135 0.00 0.00   
6136 0.00 0.00   
6137 0.00 0.00   
... ... ...   
6556 0.00 0.00   
6557 138.17 76.42   
6558 139.16 76.47   
6559 143.39 76.59   
6560 146.50 76.69   
  
 people\_fully\_vaccinated\_per\_hundred daily\_vaccinations\_per\_million \  
6133 0.00 0.0   
6134 0.00 0.0   
6135 0.00 2.0   
6136 0.00 7.0   
6137 0.00 10.0   
... ... ...   
6556 0.00 5483.0   
6557 57.38 5526.0   
6558 58.13 6033.0   
6559 62.01 11150.0   
6560 64.77 14503.0   
  
 vaccines \  
6133 Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...   
6134 Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...   
6135 Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...   
6136 Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...   
6137 Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...   
... ...   
6556 Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...   
6557 Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...   
6558 Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...   
6559 Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...   
6560 Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...   
  
 source\_name \  
6133 Directorate General of Health Services   
6134 Directorate General of Health Services   
6135 Directorate General of Health Services   
6136 Directorate General of Health Services   
6137 Directorate General of Health Services   
... ...   
6556 Directorate General of Health Services   
6557 Directorate General of Health Services   
6558 Directorate General of Health Services   
6559 Directorate General of Health Services   
6560 Directorate General of Health Services   
  
 source\_website year month day   
6133 http://103.247.238.92/webportal/pages/covid19-... 2021 1 26   
6134 http://103.247.238.92/webportal/pages/covid19-... 2021 1 27   
6135 http://103.247.238.92/webportal/pages/covid19-... 2021 1 28   
6136 http://103.247.238.92/webportal/pages/covid19-... 2021 1 29   
6137 http://103.247.238.92/webportal/pages/covid19-... 2021 1 30   
... ... ... ... ...   
6556 http://103.247.238.92/webportal/pages/covid19-... 2022 3 25   
6557 http://103.247.238.92/webportal/pages/covid19-... 2022 3 26   
6558 http://103.247.238.92/webportal/pages/covid19-... 2022 3 27   
6559 http://103.247.238.92/webportal/pages/covid19-... 2022 3 28   
6560 http://103.247.238.92/webportal/pages/covid19-... 2022 3 29   
  
[428 rows x 18 columns]

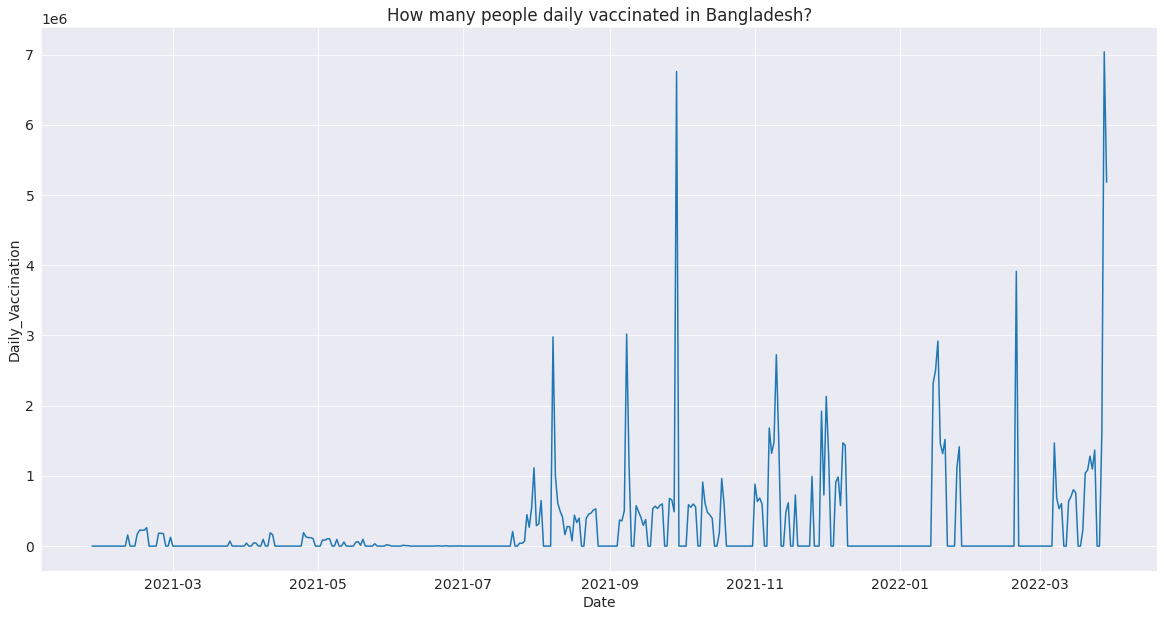
bangladesh\_df.info()

<class 'pandas.core.frame.DataFrame'>  
Int64Index: 428 entries, 6133 to 6560  
Data columns (total 18 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 country 428 non-null object   
 1 iso\_code 428 non-null object   
 2 date 428 non-null datetime64[ns]  
 3 total\_vaccinations 428 non-null float64   
 4 people\_vaccinated 428 non-null float64   
 5 people\_fully\_vaccinated 428 non-null float64   
 6 daily\_vaccinations\_raw 428 non-null float64   
 7 daily\_vaccinations 428 non-null float64   
 8 total\_vaccinations\_per\_hundred 428 non-null float64   
 9 people\_vaccinated\_per\_hundred 428 non-null float64   
 10 people\_fully\_vaccinated\_per\_hundred 428 non-null float64   
 11 daily\_vaccinations\_per\_million 428 non-null float64   
 12 vaccines 428 non-null object   
 13 source\_name 428 non-null object   
 14 source\_website 428 non-null object   
 15 year 428 non-null int64   
 16 month 428 non-null int64   
 17 day 428 non-null int64   
dtypes: datetime64[ns](1), float64(9), int64(3), object(5)  
memory usage: 63.5+ KB

bangladesh\_df.daily\_vaccinations\_raw.sum()

114220220.0

plt.figure(figsize=(20,10))  
sns.lineplot(x=bangladesh\_df.date, y=bangladesh\_df.daily\_vaccinations\_raw)  
plt.xlabel("Date")  
plt.ylabel("Daily\_Vaccination")  
plt.title('How many people daily vaccinated in Bangladesh?');



#### Q4: How many people take at least one dose of vaccine in Bangladesh?

total\_vaccinated\_bd = bangladesh\_df.total\_vaccinations.max()/1000000

print("Total doses given in Bangladesh: {0:.2f} M".format(total\_vaccinated\_bd))

Total doses given in Bangladesh: 243.64 M

#### Q5: How many people total fully vaccinated in Bangladesh?

fully\_vaccinated\_bd = bangladesh\_df.people\_fully\_vaccinated.max()/1000000

print("Total fully vaccinated people in Bangladesh: {0:.2f}M".format(fully\_vaccinated\_bd))

Total fully vaccinated people in Bangladesh: 107.71M

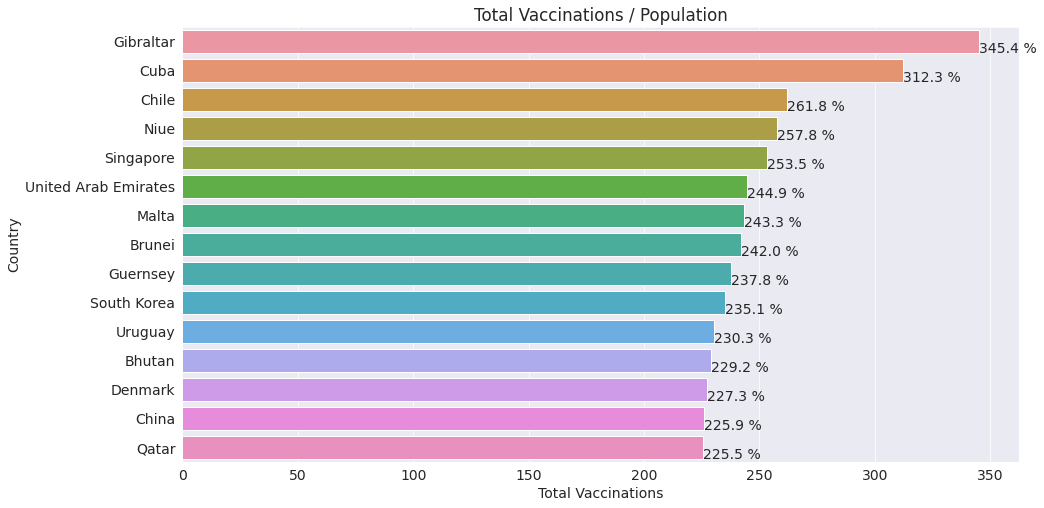
#### Q6: What is the country that vaccinated completely most of the population?

population\_country=vaccinations\_df.groupby('country')['total\_vaccinations\_per\_hundred'].max().sort\_values(ascending=False).head(15)

population\_country.reset\_index()

country total\_vaccinations\_per\_hundred  
0 Gibraltar 345.37  
1 Cuba 312.28  
2 Chile 261.82  
3 Niue 257.81  
4 Singapore 253.51  
5 United Arab Emirates 244.86  
6 Malta 243.33  
7 Brunei 241.99  
8 Guernsey 237.79  
9 South Korea 235.07  
10 Uruguay 230.34  
11 Bhutan 229.25  
12 Denmark 227.27  
13 China 225.94  
14 Qatar 225.46

plt.figure(figsize= (15, 8))  
ax = sns.barplot(x=population\_country, y=population\_country.index)  
plt.title('Total Vaccinations / Population')  
plt.xlabel('Total Vaccinations')  
plt.ylabel('Country')  
  
for patch in ax.patches:  
 width = patch.get\_width()  
 height = patch.get\_height()  
 x = patch.get\_x()  
 y = patch.get\_y()  
   
 plt.text(width + x, height + y, '{:.1f} %'.format(width))



# Data Preparation & Cleaning

We read the data file and aggregate the data on a few fields (country, iso\_code, and vaccines — that is the vaccination scheme used in a certain country). Data Cleaning is the most crucial step towards a successful data analysis project. In most of the cases, the dataset has few “NaN”(not a number) values, some empty rows(having value 0) as well as redundant columns which could be removed using and configuring drop function and changing NaN values to 0 or removing the entire row as per need.

### Initializing Dataset

Importing Dataset on Covid-19 India case time series

data = pd.read\_csv('case\_time\_series.csv')



case\_time\_series.csv dataset has 7 column. We will be collecting Daily Confirmed  Daily Recovered and Daily Deceased in variables as array.

Y = data.iloc[61:,1].values #Stores Daily Confirmed

R = data.iloc[61:,3].values #Stores Daily Recovered

D = data.iloc[61:,5].values #Stores Daily Deceased

X = data.iloc[61:,0] #Stores Date

‘*Y’*variable stores the ‘Daily Confirmed’ corona virus cases

‘*R’*variable stores the ‘Daily Recovered’ corona virus cases

‘D*’*variable stores the ‘Daily Deceased’ corona virus cases

And ‘X’ variable stores the ‘Date’ column

## Plotting Simple Plot

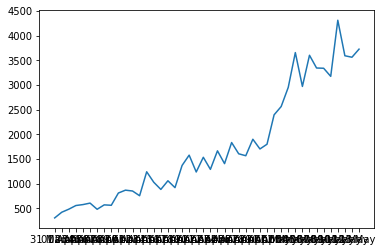
We’ll be following the object-oriented method for plotting. The plot function takes two arguments that are X-axis values and Y-axis values plot. In this case, we will pass the ‘X’ variable which has ‘Dates’ and ‘Y’ variable which has ‘Daily Confirmed’ to plot.

**Example:**

* Python3

|  |
| --- |
| **import** numpy as np  **import** pandas as pd  **import** matplotlib.pyplot as plt      data **=** pd.read\_csv('case\_time\_series.csv')    Y **=** data.iloc[61:,1].values  R **=** data.iloc[61:,3].values  D **=** data.iloc[61:,5].values  X **=** data.iloc[61:,0]    plt.plot(X,Y) |

**Output:**



We get a Simple Plot on the execution of above code which looks like this where X-axis has Dates and Y-axis has Number of Confirmed cases. But this is not the best representation for large dataset values along the axes as you can see the ‘Dates’ are overlapping on X-axis. To overcome this challenge we will introduce some new functions to take more control over the aesthetics of the graph.

### Aesthetics

To have control over the aesthetics of the graph such as labels, titles, color and size we shall apply more functions as shown below.

plt.figure(figsize=(25,8))

This creates a canvas for the graph where the first value ‘25’ is the width argument position and ‘8’ is the height argument position of the graph.

ax = plt.axes()

Let’s create an object of the axes of the graph as ‘ax’ so it becomes easier to implement functions.

ax.grid(linewidth=0.4, color='#8f8f8f')

‘.grid’ function lets you create grid lines across the graph. The width of the grid lines can be adjusted by simply passing an argument ‘linewidth’ and changing its color by passing ‘color’ argument.

ax.set\_facecolor("black")

ax.set\_xlabel('\nDate',size=25,

color='#4bb4f2')

ax.set\_ylabel('Number of Confirmed Cases\n',

size=25,color='#4bb4f2')

‘.set\_facecolor’ lets you set the background color of the graph which over here is black. ‘.set\_xlabel’ and ‘.set\_ylabel’ lets you set the label along both axes whose size and color can be altered .

ax.plot(X,Y,

color='#1F77B4',

marker='o',

linewidth=4,

markersize=15,

markeredgecolor='#035E9B')

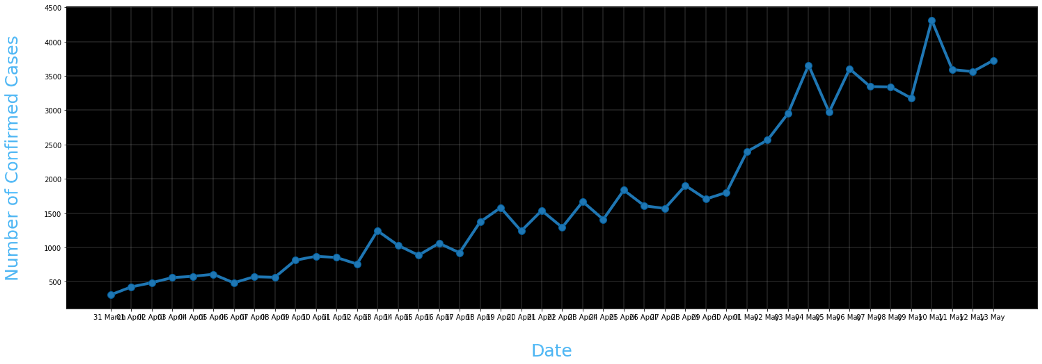
Now we plot the graph again with ‘X’ as Dates and ‘Y’ as Daily Confirmed by calling plot function. The plotting line , marker and color can be altered by passing color, linewidth?—?to change color and adjust the width of plotting line and marker, markersize, markeredgecolor?—?to create marker which is the circle in this case, adjust the size of the marker and define marker’s edge color.

**Complete Code:**

* Python3

|  |
| --- |
| **import** numpy as np  **import** pandas as pd  **import** matplotlib.pyplot as plt    data **=** pd.read\_csv('case\_time\_series.csv')    Y **=** data.iloc[61:,1].values  R **=** data.iloc[61:,3].values  D **=** data.iloc[61:,5].values  X **=** data.iloc[61:,0]    plt.figure(figsize**=**(25,8))    ax **=** plt.axes()  ax.grid(linewidth**=**0.4, color**=**'#8f8f8f')    ax.set\_facecolor("black")  ax.set\_xlabel('\nDate',size**=**25,color**=**'#4bb4f2')  ax.set\_ylabel('Number of Confirmed Cases\n',                size**=**25,color**=**'#4bb4f2')    ax.plot(X,Y,          color**=**'#1F77B4',          marker**=**'o',          linewidth**=**4,          markersize**=**15,          markeredgecolor**=**'#035E9B') |

**Output:**



Still as we can see the dates are overlapping and the labels along axes are not clear enough. So now we are going to change the ticks of the axes and also annotate the plots.

plt.xticks(rotation='vertical',size='20',color='white')

plt.yticks(size=20,color='white')

plt.tick\_params(size=20,color='white')

‘.xticks’ and ‘.yticks’ lets you alter the Dates and Daily Confirmed font. For the Dates to not overlap with each other we are going to represent them vertically by passing ‘rotation = vertical’. To make the ticks easily readable we change the font color to white and size to 20.

‘.tick\_params’ lets you alter the size and color of the dashes which look act like a bridge between the ticks and graph.

for i,j in zip(X,Y):

ax.annotate(str(j),xy=(i,j+100),

color='white',size='13')

‘.annotate’ lets you annotate on the graph. Over here we have written a code to annotate the plotted points by running a for loop which plots at the plotted points. The str(j) holds the ‘Y’ variable which is Daily Confirmed. Any string passed will be plotted. The XY is the coordinates where the string should be plotted. And finally the color and size can be defined. Note we have added +100 to j in XY coordinates so that the string doesn’t overlap with the marker and it is at the distance of 100 units on Y?—?axis.

ax.annotate('Second Lockdown 15th April',

xy=(15.2, 860),

xytext=(19.9,500),

color='white',

size='25',

arrowprops=dict(color='white',

linewidth=0.025))

To annotate an arrow pointing at a position in graph and its tail holding the string we can define ‘arrowprops’ argument along with its tail coordinates defined by ‘xytext’. Note that ‘arrowprops’ alteration can be done using a dictionary.

plt.title("COVID-19 IN : Daily Confirmed\n",

size=50,color='#28a9ff')

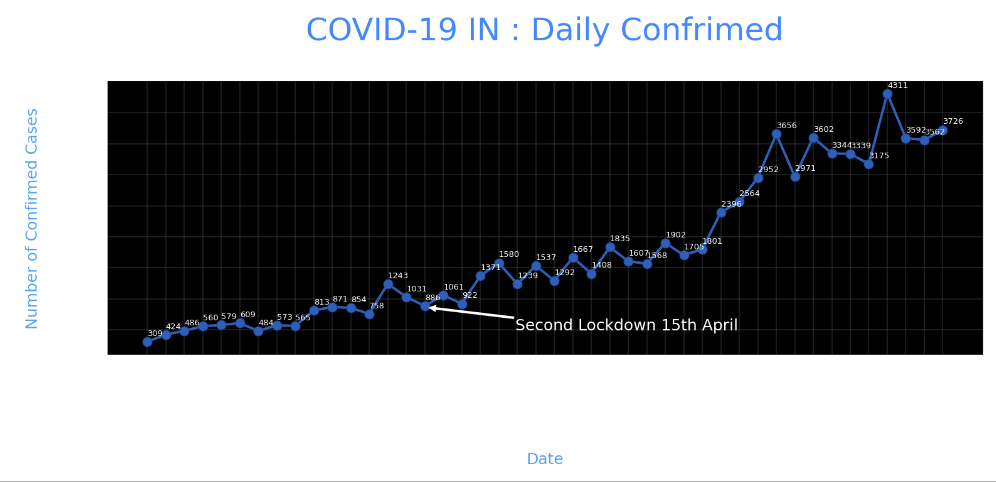
Finally we define a title by ‘.title’ and passing string ,its size and color.

**Complete Code:**

* Python3

|  |
| --- |
| **import** numpy as np  **import** pandas as pd  **import** matplotlib.pyplot as plt    data **=** pd.read\_csv('case\_time\_series.csv')    Y **=** data.iloc[61:,1].values  R **=** data.iloc[61:,3].values  D **=** data.iloc[61:,5].values  X **=** data.iloc[61:,0]    plt.figure(figsize**=**(25,8))    ax **=** plt.axes()  ax.grid(linewidth**=**0.4, color**=**'#8f8f8f')    ax.set\_facecolor("black")  ax.set\_xlabel('\nDate',size**=**25,color**=**'#4bb4f2')  ax.set\_ylabel('Number of Confirmed Cases\n',                size**=**25,color**=**'#4bb4f2')    plt.xticks(rotation**=**'vertical',size**=**'20',color**=**'white')  plt.yticks(size**=**20,color**=**'white')  plt.tick\_params(size**=**20,color**=**'white')    **for** i,j **in** zip(X,Y):      ax.annotate(str(j),xy**=**(i,j**+**100),color**=**'white',size**=**'13')    ax.annotate('Second Lockdown 15th April',              xy**=**(15.2, 860),              xytext**=**(19.9,500),              color**=**'white',              size**=**'25',              arrowprops**=**dict(color**=**'white',                              linewidth**=**0.025))    plt.title("COVID-19 IN : Daily Confirmed\n",            size**=**50,color**=**'#28a9ff')    ax.plot(X,Y,          color**=**'#1F77B4',          marker**=**'o',          linewidth**=**4,          markersize**=**15,          markeredgecolor**=**'#035E9B') |

**Output:**



## Pie Chart

We’ll be plotting the Transmission Pie Chart to understand the how the virus is spreading based on Travel, Place Visit and Unknown reason.

### Initializing Dataset

slices = [62, 142, 195]

activities = ['Travel', 'Place Visit', 'Unknown']

So we have created list slices based on which our Pie Chart will be divided and the corresponding activities are it’s valued.

### Plotting Pie Chart

cols=['#4C8BE2','#00e061','#fe073a']

exp = [0.2,0.02,0.02]

plt.pie(slices,

labels=activities,

textprops=dict(size=25,color='black'),

radius=3,

colors=cols,

autopct='%2.2f%%',

explode=exp,

shadow=True,

startangle=90)

plt.title('Transmission\n\n\n\n',color='#4fb4f2',size=40)

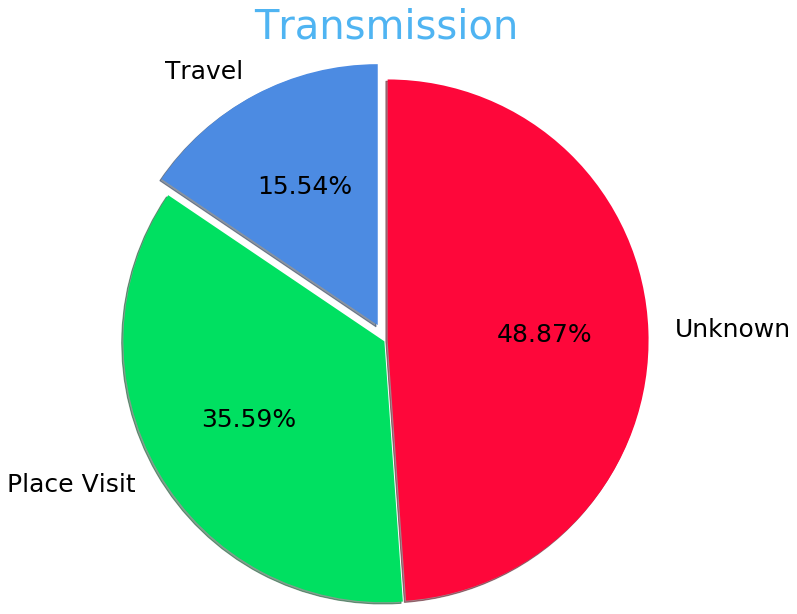
To plot a Pie Chart we call ‘.pie’ function which takes x values which is ‘slices’ over here based on it the pie is divided followed by labels which have the corresponding string the values it represents. These string values can be altered by ‘textprops’. To change the radius or size of Pie we call ‘radius’. For the aesthetics we call ‘shadow’ as True and ‘startangle ’= 90. We can define colors to assign by passing a list of corresponding colors. To space out each piece of Pie we can pass on the list of corresponding values to ‘explode’. The ‘autopct’ defines the number of positions that are allowed to be shown. In this case, autopct allows 2 positions before and after the decimal place.

**Complete** **Code:**

* Python3

|  |
| --- |
| slices **=** [62, 142, 195]  activities **=** ['Travel', 'Place Visit', 'Unknown']    cols**=**['#4C8BE2','#00e061','#fe073a']  exp **=** [0.2,0.02,0.02]    plt.pie(slices,labels**=**activities,          textprops**=**dict(size**=**25,color**=**'black'),          radius**=**3,          colors**=**cols,          autopct**=**'%2.2f%%',          explode**=**exp,          shadow**=**True,          startangle**=**90)    plt.title('Transmission\n\n\n\n',color**=**'#4fb4f2',size**=**40) |

**Output:**



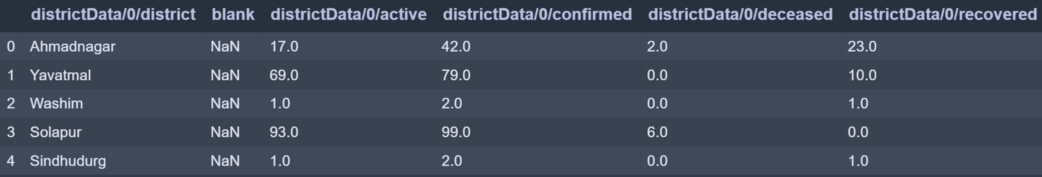
## Bar Plot

Now we are going to plot the most common type of graph which is a bar plot. Over here we will be plotting the district wise coronavirus cases for a state.

### Initializing Dataset

data = pd.read\_csv('district.csv')

data.head()



re=data.iloc[:30,5].values

de=data.iloc[:30,4].values

co=data.iloc[:30,3].values

x=list(data.iloc[:30,0])

‘re’ stores Recovered corona patients count for all districts.

‘de’ stores Deceased corona patients count for all districts.

‘co’ stores Confirmed corona patients count for all districts.

‘x’ stored District names.

### Plotting Bar Chart

plt.figure(figsize=(25,10))

ax=plt.axes()

ax.set\_facecolor('black')

ax.grid(linewidth=0.4, color='#8f8f8f')

plt.xticks(rotation='vertical',

size='20',

color='white')#ticks of X

plt.yticks(size='20',color='white')

ax.set\_xlabel('\nDistrict',size=25,

color='#4bb4f2')

ax.set\_ylabel('No. of cases\n',size=25,

color='#4bb4f2')

plt.tick\_params(size=20,color='white')

ax.set\_title('Maharashtra District wise breakdown\n',

size=50,color='#28a9ff')

The code for aesthetics will be the same as we saw in earlier plot. The Only thing that will change is calling for the bar function.

plt.bar(x,co,label='re')

plt.bar(x,re,label='re',color='green')

plt.bar(x,de,label='re',color='red')

for i,j in zip(x,co):

ax.annotate(str(int(j)),

xy=(i,j+3),

color='white',

size='15')

plt.legend(['Confirmed','Recovered','Deceased'],

fontsize=20)

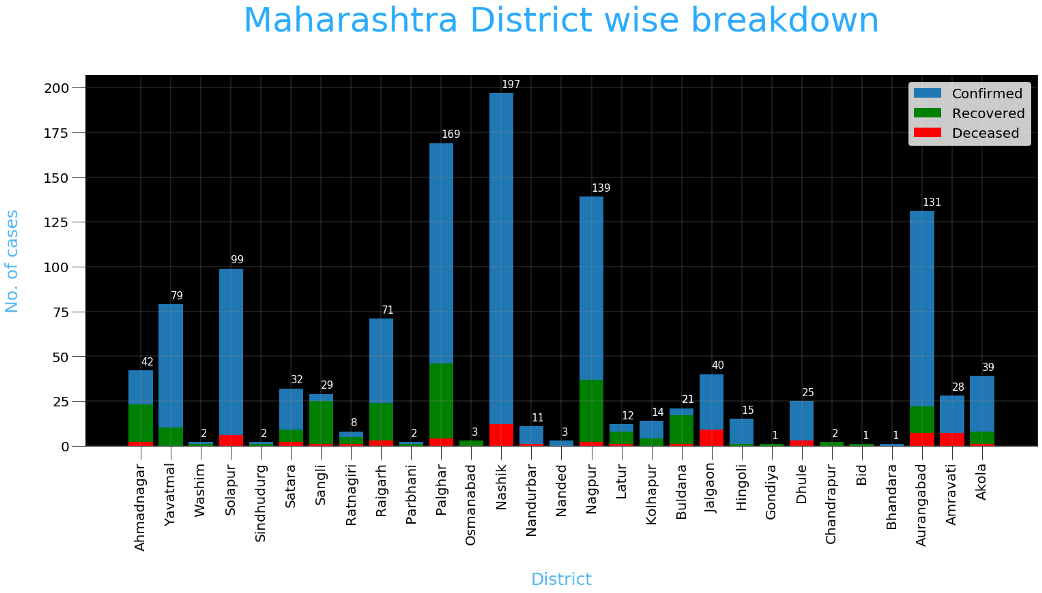
To plot a bar graph we call ‘.bar’ and pass it x-axis and y-axis values. Over here we called the plot function three times for all three cases (i.e Recovered , Confirmed, Deceased) and all three values are plotted with respect to y-axis and x-axis being common for all three which is district names. As you can see we annotate only for Confirmed numbers by iterating over Confirmed cases value. Also, we have mentioned ‘.legend’ to indicate legends of the graph.

**Complete Code:**

* Python3

|  |
| --- |
| data **=** pd.read\_csv('district.csv')  data.head()    re**=**data.iloc[:30,5].values  de**=**data.iloc[:30,4].values  co**=**data.iloc[:30,3].values  x**=**list(data.iloc[:30,0])    plt.figure(figsize**=**(25,10))  ax**=**plt.axes()    ax.set\_facecolor('black')  ax.grid(linewidth**=**0.4, color**=**'#8f8f8f')      plt.xticks(rotation**=**'vertical',             size**=**'20',             color**=**'white')#ticks of X    plt.yticks(size**=**'20',color**=**'white')      ax.set\_xlabel('\nDistrict',size**=**25,                color**=**'#4bb4f2')  ax.set\_ylabel('No. of cases\n',size**=**25,                color**=**'#4bb4f2')      plt.tick\_params(size**=**20,color**=**'white')      ax.set\_title('Maharashtra District wise breakdown\n',               size**=**50,color**=**'#28a9ff')    plt.bar(x,co,label**=**'re')  plt.bar(x,re,label**=**'re',color**=**'green')  plt.bar(x,de,label**=**'re',color**=**'red')    **for** i,j **in** zip(x,co):      ax.annotate(str(int(j)),                  xy**=**(i,j**+**3),                  color**=**'white',                  size**=**'15')    plt.legend(['Confirmed','Recovered','Deceased'],             fontsize**=**20) |

**Output:**



## Conclusion

Here is the analysis of the covid-19 vaccinations data. In future we work more analysis on this data.