

Covid-19 Vaccination World – EDA

About the Data Set

The data set is about the vaccination details of the current Covid-19 pandemic all over the world.

The attributes in the data set include the daily count of vaccinations as well as the percentage from the beginning of the year 2021 (Some Dec 2020 also). It also includes the type of vaccine used by each country and that will be really useful to analyze and visualize to have a good idea about the vaccines recommended most. Some of the other attributes are the dates and source of each feature. Here in this project I cleaned the data for the exploratory data analysis of my dataset and then visualized them to highlight some general features of the data set.

The data set for this project was collected from <https://www.kaggle.com/datasets>.

Initial Plan for the Exploration

Key Features to be used

- Data cleaning and feature engineering
- Key findings and insights
- Visualizing in a useful manner
- Hypothesis testing
- Next moves

Exploring the data

```
import numpy as np
import pandas as pd
import seaborn as sb
from scipy import stats
import matplotlib.pyplot as plt

%matplotlib inline

vd = pd.read_csv("country_vaccinations.csv")
```



Let's navigate to the top five rows of the dataframe.

```
vd.head()
```

Out[230]:

| | country | iso_code | date | total_vaccinations | people_vaccinated | people_fully_vaccinated | daily_vaccinations_raw | daily_vaccinations | total_vaccinations_per_ |
|---|-------------|----------|------------|--------------------|-------------------|-------------------------|------------------------|--------------------|-------------------------|
| 0 | Afghanistan | AFG | 2021-02-22 | 0.0 | 0.0 | NaN | NaN | NaN | |
| 1 | Afghanistan | AFG | 2021-02-23 | NaN | NaN | NaN | NaN | 1367.0 | |
| 2 | Afghanistan | AFG | 2021-02-24 | NaN | NaN | NaN | NaN | 1367.0 | |
| 3 | Afghanistan | AFG | 2021-02-25 | NaN | NaN | NaN | NaN | 1367.0 | |
| 4 | Afghanistan | AFG | 2021-02-26 | NaN | NaN | NaN | NaN | 1367.0 | |

Here we have to select the columns we need for the analysis. The useless columns should be dropped from the DataFrame. So Let us drop these columns in order to clean our dataset. Below mentioned data is only sufficient for our analysis.

1. iso_code
2. date
3. total_vaccinations
4. daily_vaccinations
5. total_vaccinations_per_hundred
6. daily_vaccinations_per_million
7. vaccines

We can rename the data in the attribute 'date' as follows.

```
vd['date'] = vd.date.str.replace('2021-', '')
vd['date'] = vd.date.str.replace('2020-', '')
vd.head()
```



| | country | iso_code | date | total_vaccinations | people_vaccinated | people_fully_vaccinated | daily_vaccinations_raw | daily_vaccinations | total_vaccinations_per_hundred |
|---|-------------|----------|-------|--------------------|-------------------|-------------------------|------------------------|--------------------|--------------------------------|
| 0 | Afghanistan | AFG | 02-22 | 0.0 | 0.0 | NaN | NaN | NaN | NaN |
| 1 | Afghanistan | AFG | 02-23 | NaN | NaN | NaN | NaN | NaN | 1367.0 |
| 2 | Afghanistan | AFG | 02-24 | NaN | NaN | NaN | NaN | NaN | 1367.0 |
| 3 | Afghanistan | AFG | 02-25 | NaN | NaN | NaN | NaN | NaN | 1367.0 |
| 4 | Afghanistan | AFG | 02-26 | NaN | NaN | NaN | NaN | NaN | 1367.0 |

```
vd = vd.rename(columns={'iso_code': 'countryCode', 'total_vaccinations': 'totalCount', 'daily_vaccinations': 'dailyCount', 'total_vaccinations_per_hundred': 'totalPercentage', 'daily_vaccinations_per_million': 'dailyCountPerMillion'})
```

```
vd.head()
```

| | countryCode | date | totalCount | totalPercentage | people_vaccinated_per_hundred | people_fully_vaccinated_per_hundred | dailyCountPerMillion | vaccines | source_name |
|---|-------------|-------|------------|-----------------|-------------------------------|-------------------------------------|----------------------|--------------------|--------------------------|
| 0 | AFG | 02-22 | 0.0 | 0.0 | NaN | NaN | NaN | Oxford/AstraZeneca | Governmer of Afghanistan |
| 1 | AFG | 02-23 | NaN | NaN | NaN | NaN | 35.0 | Oxford/AstraZeneca | Governmer of Afghanistan |
| 2 | AFG | 02-24 | NaN | NaN | NaN | NaN | 35.0 | Oxford/AstraZeneca | Governmer of Afghanistan |
| 3 | AFG | 02-25 | NaN | NaN | NaN | NaN | 35.0 | Oxford/AstraZeneca | Governmer of Afghanistan |
| 4 | AFG | 02-26 | NaN | NaN | NaN | NaN | 35.0 | Oxford/AstraZeneca | Governmer of Afghanistan |

Now let's have a description about the data set

```
print('Number of rows: ' + str(vd.shape[0]))
print('Column names: ' + str(vd.columns.tolist()))
print('Number of countries: ' + str(len(vd['countryCode'].unique())))
print('Number of missing values: \n' + str(vd.isnull().sum()))
```



```

Number of rows: 15666
Column names: ['country', 'countryCode', 'date', 'totalCount', 'people_vaccinated', 'people_fully_vaccinated', 'daily_vaccinations_raw', 'dailyCount', 'totalPercentage', 'people_vaccinated_per_hundred', 'people_fully_vaccinated_per_hundred', 'dailyCountPerMillion', 'vaccines', 'source_name', 'source_website']
Number of countries: 196
Number of missing values:
country                0
countryCode            0
date                   0
totalCount              6229
people_vaccinated      6912
people_fully_vaccinated 9164
daily_vaccinations_raw 7738
dailyCount              201
totalPercentage        6229
people_vaccinated_per_hundred 6912
people_fully_vaccinated_per_hundred 9164
dailyCountPerMillion   201
vaccines                0
source_name             0
source_website          0
dtype: int64

```

In [116]:

```

sufficient_fields = ['countryCode', 'date', 'dailyCount', 'dailyCountPerMillion', 'vaccines']
vd = vd[sufficient_fields]
vd.head()

```

Out[116]:

| | countryCode | date | dailyCount | dailyCountPerMillion | vaccines |
|---|-------------|-------|------------|----------------------|--------------------|
| 0 | AFG | 02-22 | NaN | NaN | Oxford/AstraZeneca |
| 1 | AFG | 02-23 | 1367.0 | 35.0 | Oxford/AstraZeneca |
| 2 | AFG | 02-24 | 1367.0 | 35.0 | Oxford/AstraZeneca |
| 3 | AFG | 02-25 | 1367.0 | 35.0 | Oxford/AstraZeneca |
| 4 | AFG | 02-26 | 1367.0 | 35.0 | Oxford/AstraZeneca |



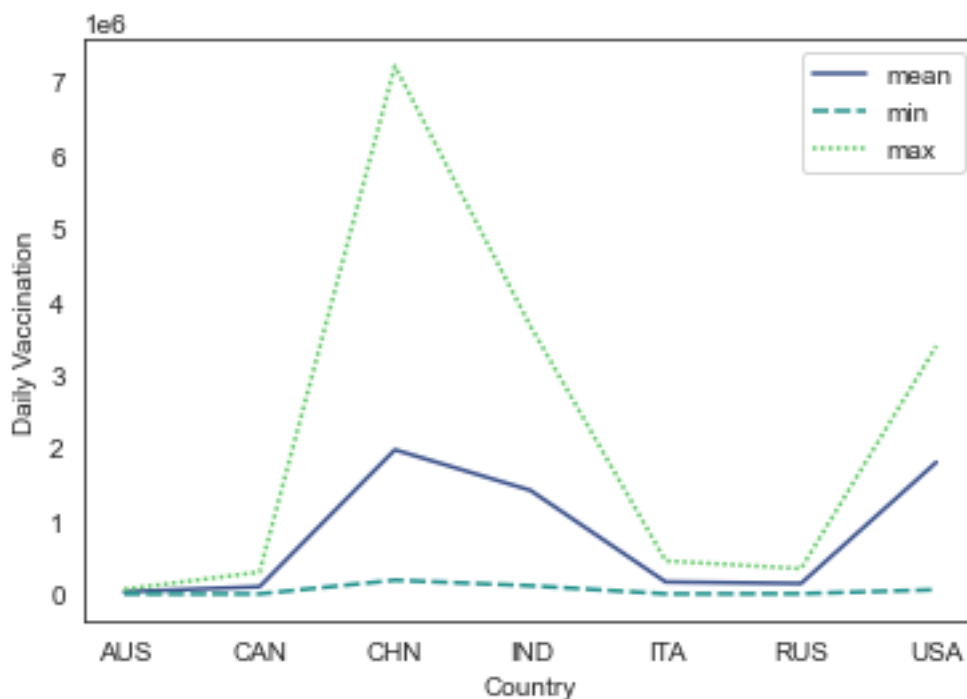
Below is the graph of the mean, min and max of daily vaccination count of some specified countries:

```
spec = vd[(vd['countryCode'] == 'ITA') | (vd['countryCode'] == 'USA') | (vd['countryCode'] == 'IND') |
(vd['countryCode'] == 'CHN') | (vd['countryCode'] == 'RUS') | (vd['countryCode'] == 'UK') | (vd['countryCode'] ==
'AUS') | (vd['countryCode'] == 'CAN')]

df = spec.groupby('countryCode')['dailyCount'].agg(['mean', 'min', 'max'])

ax = sb.lineplot(data=df, palette="viridis")

ax.set(xlabel='Country', ylabel='Daily Vaccination')
```



Let's visualize the vaccines used by above mentioned countries.

```
lst_col = 'vaccines'

x = vd.assign(**{lst_col:vd[lst_col].str.split(', ')})

vd2 = pd.DataFrame({col:np.repeat(x[col].values, x[lst_col].str.len())

for col in
x.columns.difference([lst_col])}).assign(**{lst_col:np.concatenate(x[lst_col].values)})[x.columns.tolist()]

vd2.vaccines.value_counts()
```



| | |
|--------------------|-------|
| Oxford/AstraZeneca | 11702 |
| Pfizer/BioNTech | 9907 |
| Moderna | 4903 |
| Sinopharm/Beijing | 2866 |
| Sputnik V | 2437 |
| Sinovac | 2147 |
| Johnson&Johnson | 1749 |
| Sinopharm/Wuhan | 265 |
| CanSino | 226 |
| EpiVacCorona | 143 |
| Covaxin | 112 |

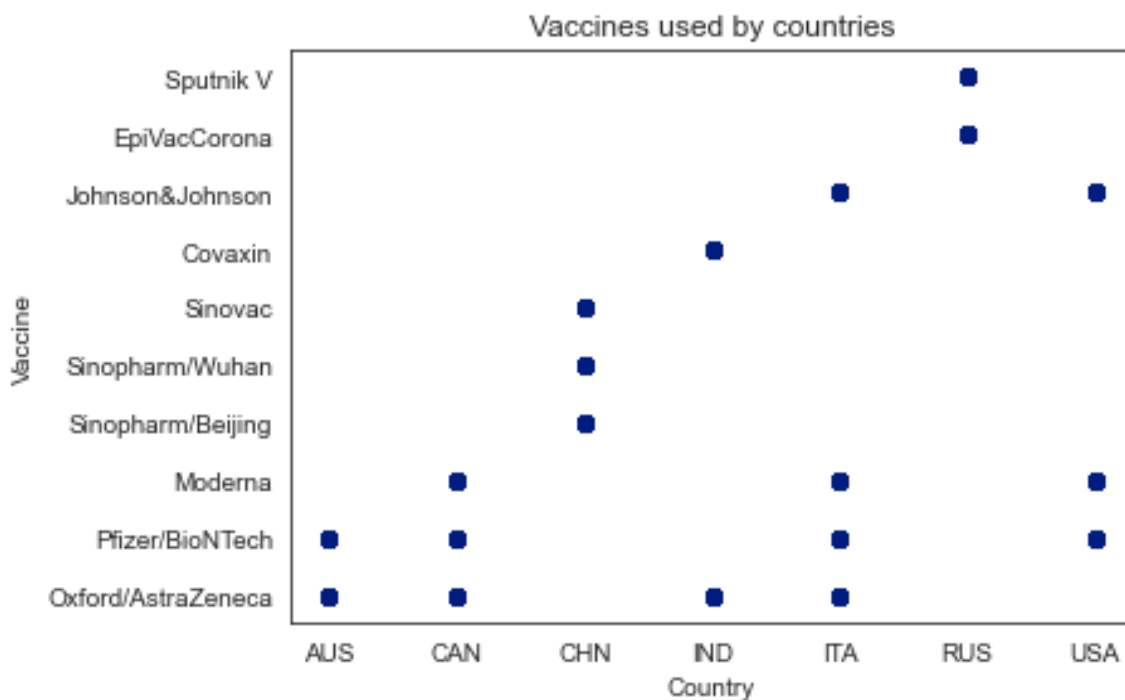
Name: vaccines, dtype: int64

```
spec = vd2[(vd2['countryCode'] == 'ITA') | (vd2['countryCode'] == 'USA') | (vd2['countryCode'] == 'IND') |
(vd2['countryCode'] == 'CHN') | (vd2['countryCode'] == 'RUS') | (vd2['countryCode'] == 'UK') |
(vd2['countryCode'] == 'AUS') | (vd2['countryCode'] == 'CAN')]

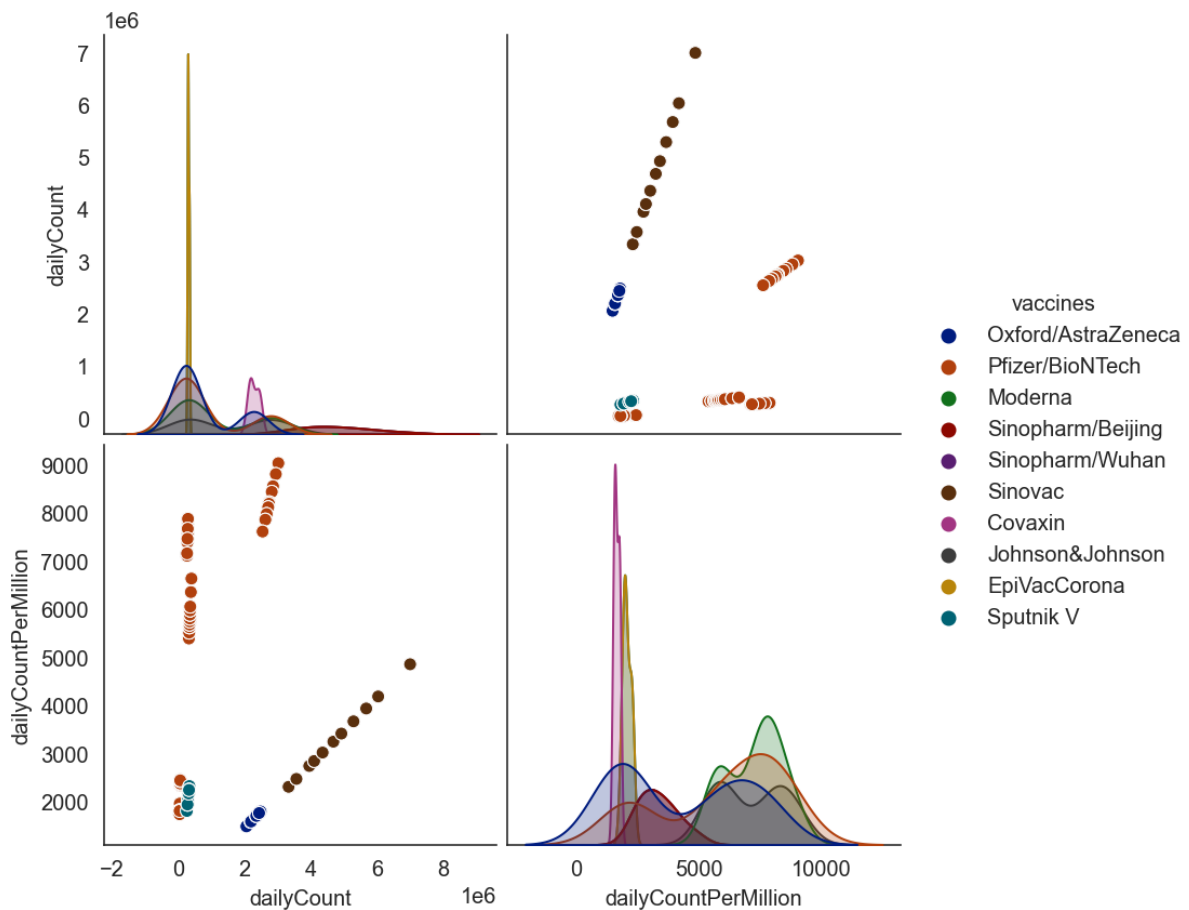
ax = plt.axes()

ax.scatter(spec.countryCode, spec.vaccines)

ax.set(xlabel='Country',
       ylabel='Vaccine',
       title='Vaccines used by countries');
```



```
sb.pairplot(spec, hue='vaccines', height=3)
```



Feature Engineering

```
vd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15666 entries, 0 to 15665
Data columns (total 5 columns):
#   Column                Non-Null Count  Dtype
---  -
0   countryCode            15666 non-null  object
1   date                   15666 non-null  object
2   dailyCount             15465 non-null  float64
3   dailyCountPerMillion   15465 non-null  float64
4   vaccines               15666 non-null  object
dtypes: float64(2), object(3)
memory usage: 612.1+ KB
```



It shows that our data set has only two missing values. So there is no need of dropping any column.

```
num_cols = vd.select_dtypes('number').columns
num_cols
```

```
Index(['dailyCount', 'dailyCountPerMillion'], dtype='object')
```

```
skew_limit = 0.75
skew_vals = vd[num_cols].skew()
skew_vals
```

```
dailyCount          9.954500
dailyCountPerMillion 7.729351
dtype: float64
```

```
skew_cols = skew_vals[abs(skew_vals > skew_limit)].sort_values(ascending=False)
skew_cols
```

```
dailyCount          9.954500
dailyCountPerMillion 7.729351
dtype: float64
```

Applying logarithmic transformation

Let's select most skewed field, 'dailyCount'

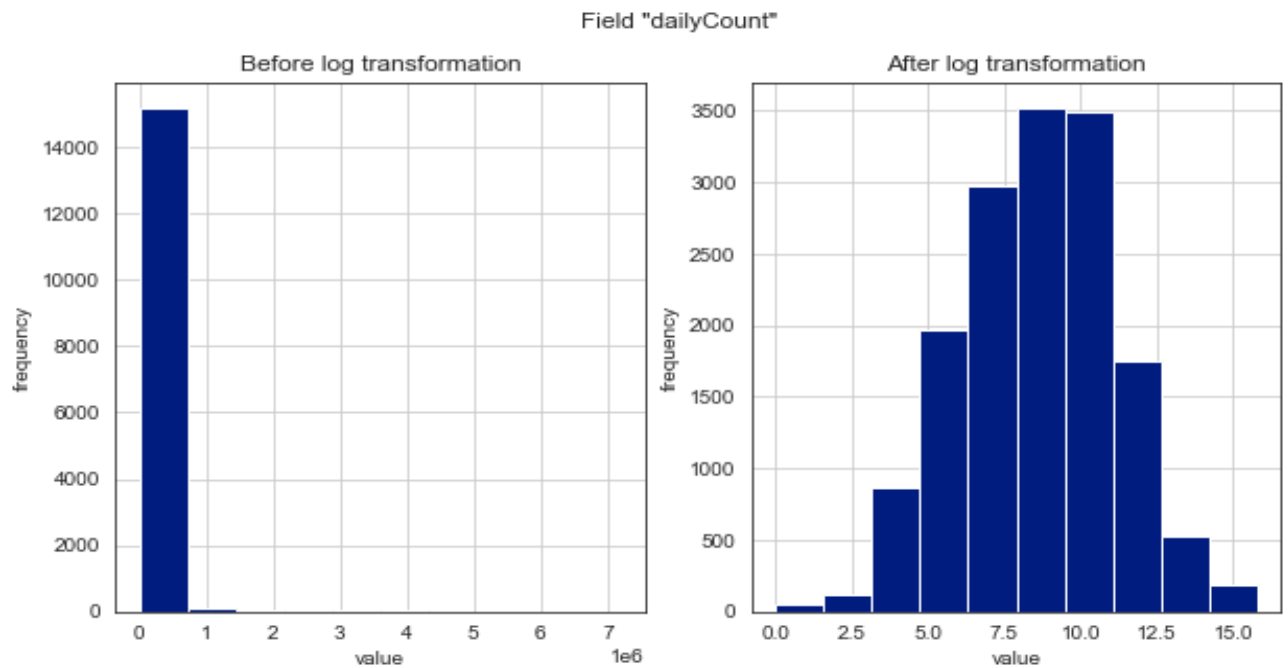
```
field = "dailyCount"

fig, (ax_before, ax_after) = plt.subplots(1, 2, figsize=(10, 5))
vd[field].hist(ax=ax_before)
vd[field].apply(np.log1p).hist(ax=ax_after)
```



Covid-19 Vaccination 2021 – EDA – Januka Fernando

```
ax_before.set(title='Before log transformation', ylabel='frequency', xlabel='value')
ax_after.set(title='After log transformation', ylabel='frequency', xlabel='value')
fig.suptitle('Field "{}".format(field));
```



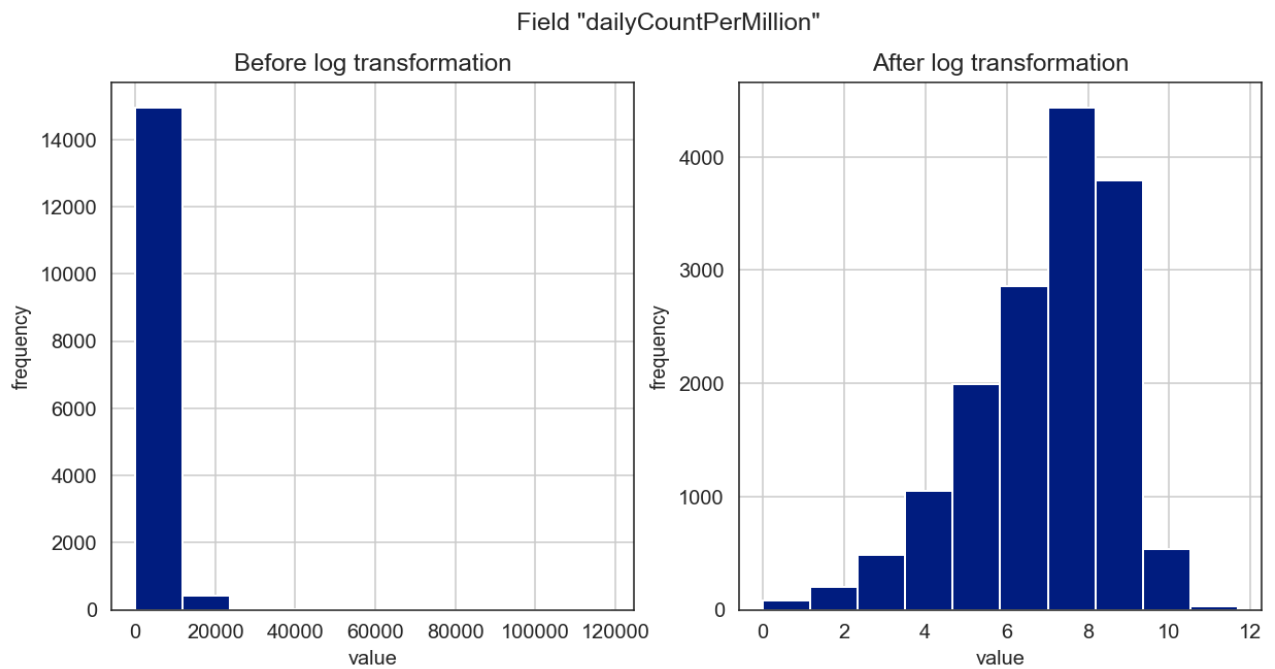
Let's do the same to the dailyCountPerMillion field

```
%config InlineBackend.figure_formats = ['retina']
field = "dailyCountPerMillion"

fig, (ax_before, ax_after) = plt.subplots(1, 2, figsize=(10, 5))
vd[field].hist(ax=ax_before)
vd[field].apply(np.log1p).hist(ax=ax_after)

ax_before.set(title='Before log transformation', ylabel='frequency', xlabel='value')
ax_after.set(title='After log transformation', ylabel='frequency', xlabel='value')
fig.suptitle('Field "{}".format(field));
```





Hypothesis Testing

Let's take the vaccination details from 11/04 to 20/4 and 21/04 to 30/04

```
spec_main = vd[(vd['countryCode'] == 'IND')]  
spec_main
```



| | countryCode | date | dailyCount | dailyCountPerMillion | vaccines |
|------|-------------|-------|------------|----------------------|-----------------------------|
| 6394 | IND | 01-15 | NaN | NaN | Covaxin, Oxford/AstraZeneca |
| 6395 | IND | 01-16 | 191181.0 | 139.0 | Covaxin, Oxford/AstraZeneca |
| 6396 | IND | 01-17 | 112150.0 | 81.0 | Covaxin, Oxford/AstraZeneca |
| 6397 | IND | 01-18 | 151350.0 | 110.0 | Covaxin, Oxford/AstraZeneca |
| 6398 | IND | 01-19 | 168709.0 | 122.0 | Covaxin, Oxford/AstraZeneca |
| ... | ... | ... | ... | ... | ... |
| 6501 | IND | 05-02 | 2146620.0 | 1556.0 | Covaxin, Oxford/AstraZeneca |
| 6502 | IND | 05-03 | 1936741.0 | 1403.0 | Covaxin, Oxford/AstraZeneca |
| 6503 | IND | 05-04 | 1838788.0 | 1332.0 | Covaxin, Oxford/AstraZeneca |
| 6504 | IND | 05-05 | 1839692.0 | 1333.0 | Covaxin, Oxford/AstraZeneca |
| 6505 | IND | 05-06 | 1904976.0 | 1380.0 | Covaxin, Oxford/AstraZeneca |

112 rows × 5 columns

Hypothesis: The mean value of the mean daily vaccination count of India during the month of March is as same as the month of April

Alternative Hypothesis: There is no relationship between number of vaccinations done in given In two months

Results

As the number of Covid-19 patients increasing and decreasing time by time, the number of vaccinations can be seen as not connected to each other.

Next Moves



Covid-19 Vaccination 2021 – EDA – Januka Fernando

After the EDA part, we can either analyse the data further or we can use the data set for future ML model to do a prediction that how many vaccines must be consumed by each country in future or to predict which of the above vaccines going to be used much or less etc.

