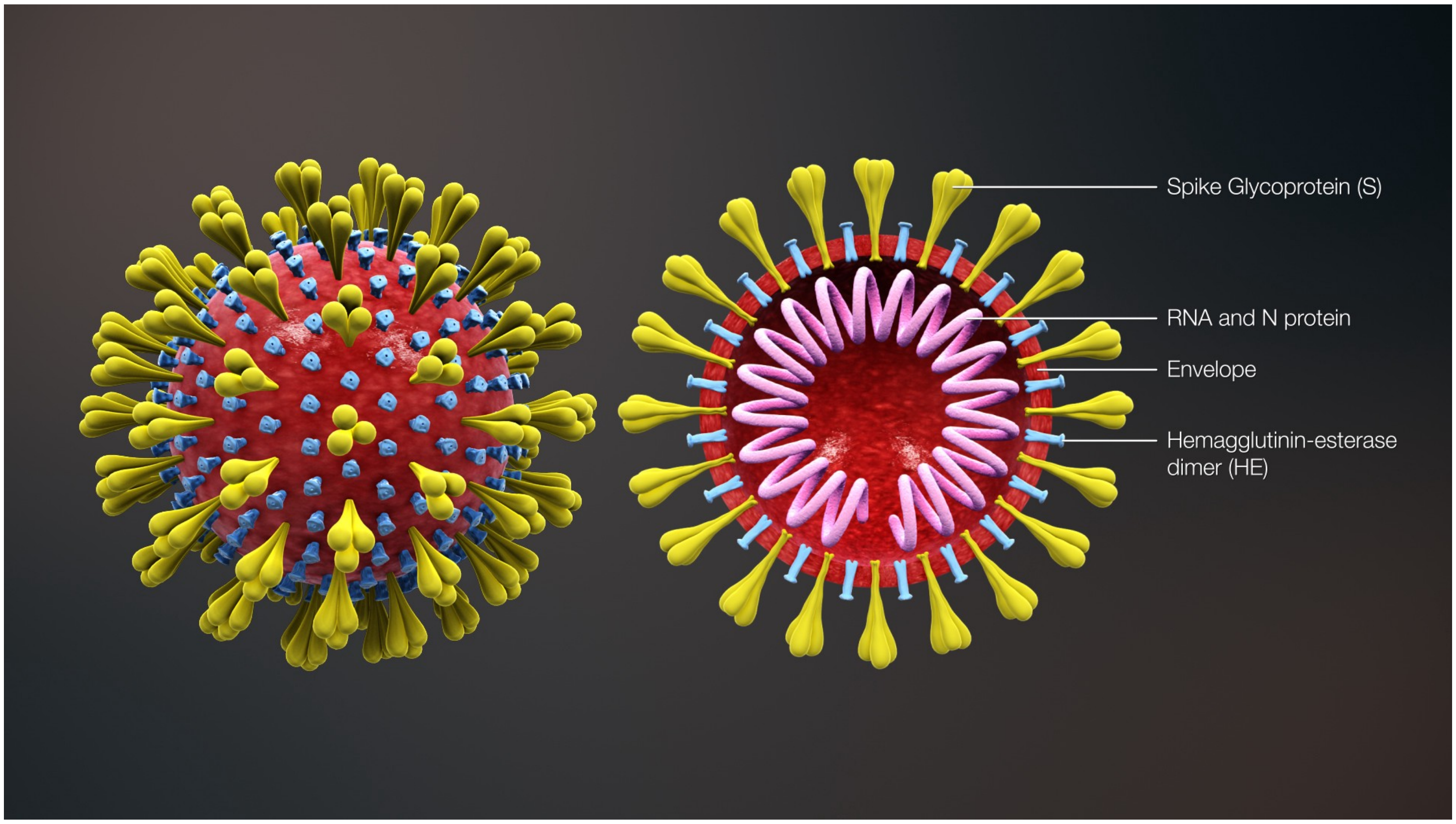
Exploring the Coronavirus Dataset

Exploratory Data Analysis of the Novel Coronavirus 2019 Dataset

Sadrach Pierre, Ph.D. Follow
Mar 5 · 4 min read ★



Source

In this post we will perform simple exploratory data analysis of the Novel Corona Virus 2019 Dataset. The data set can be found on Kaggle. The data has daily information on the number of cases, deaths, and recovery in 2019. The data set, 'covid_19_data.csv', contains the following columns:

1. Sno — Serial Number

2. ObservationDate — Date of the observation in MM/DD/YYYY
3. Province/State — Province or state of the observation
4. Country/Region — Country of observation
5. Last Update — Time in UTC at which the row is updated for the given province or county
6. Confirmed — Cumulative number of confirmed cases until that date
7. Deaths — Cumulative number of deaths until that date
8. Recovered — Cumulative number of recovered cases until that date

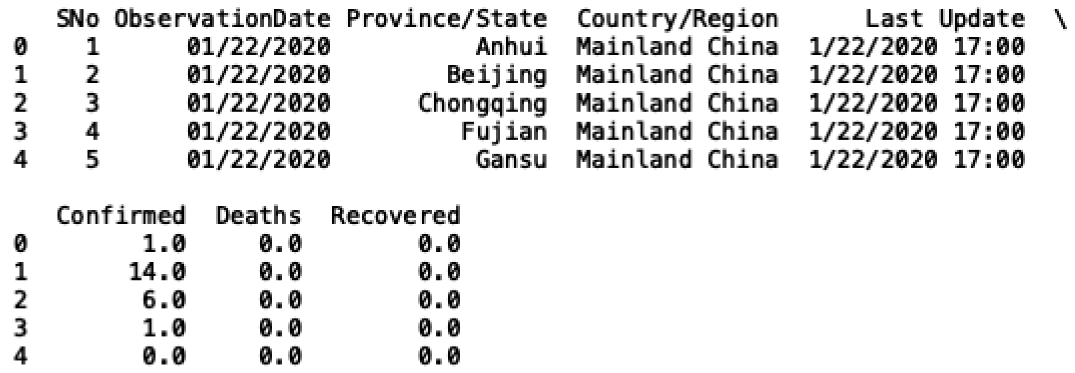
Let's read the data into a Pandas data frame:

```
import pandas as pd

df = pd.read_csv("covid_19_data.csv")
```

Let's print the first five rows of data:

```
print(df.head())
```

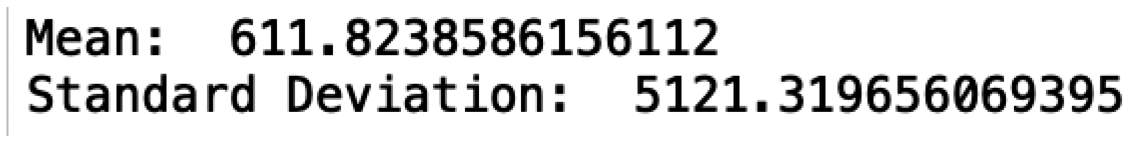


```
   SNo ObservationDate Province/State  Country/Region      Last Update  \
0    1      01/22/2020          Anhui  Mainland China  1/22/2020 17:00
1    2      01/22/2020        Beijing  Mainland China  1/22/2020 17:00
2    3      01/22/2020      Chongqing  Mainland China  1/22/2020 17:00
3    4      01/22/2020         Fujian  Mainland China  1/22/2020 17:00
4    5      01/22/2020          Gansu  Mainland China  1/22/2020 17:00

   Confirmed  Deaths  Recovered
0        1.0     0.0        0.0
1       14.0     0.0        0.0
2        6.0     0.0        0.0
3        1.0     0.0        0.0
4        0.0     0.0        0.0
```

The first thing we can do is generate some statistics on 'Confirmed' column. Let's take a look at the mean and standard deviation in confirmed cases:

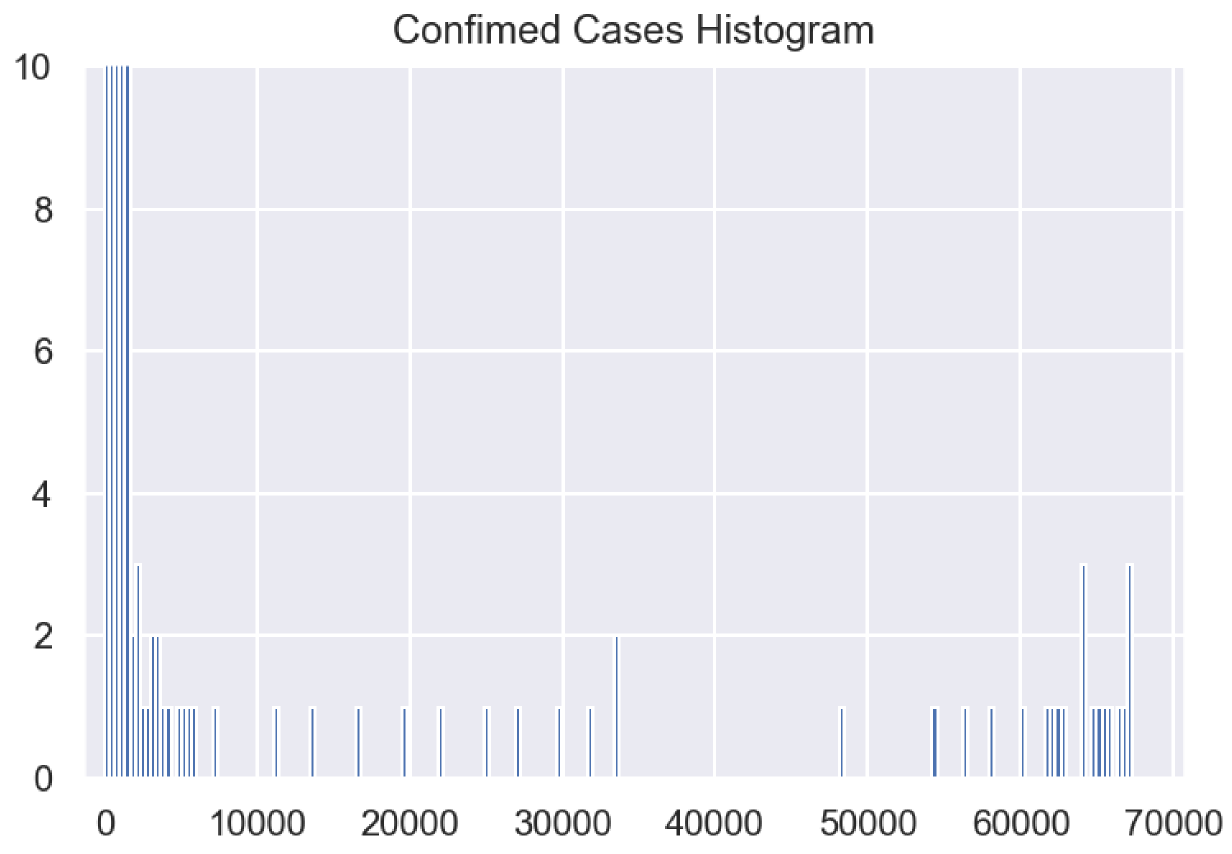
```
print("Mean: ", df['Confirmed'].mean())
print("Standard Deviation: ", df['Confirmed'].std())
```



```
Mean:  611.8238586156112
Standard Deviation:  5121.319656069395
```

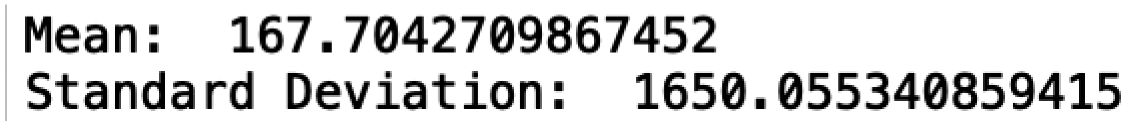
We can also generate a histogram of confirmed cumulative cases:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set()
plt.title("Confimed Cases Histogram")
df['Confirmed'].hist(bins = 10)
```



We can generate similar statistics for recovered cases:

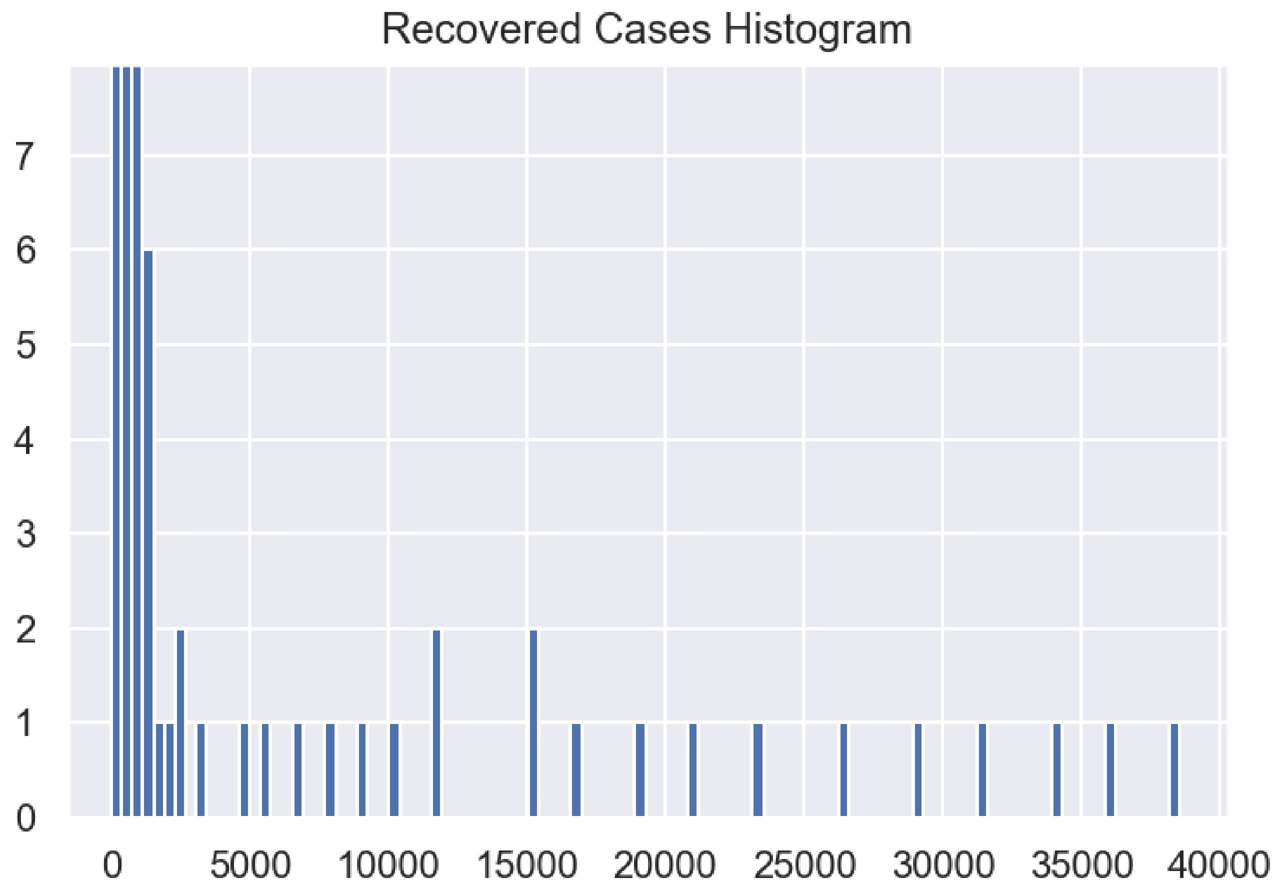
```
print("Mean: ", df['Recovered'].mean())
print("Standard Deviation: ", df['Recovered'].std())
```



```
Mean:  167.7042709867452
Standard Deviation:  1650.055340859415
```

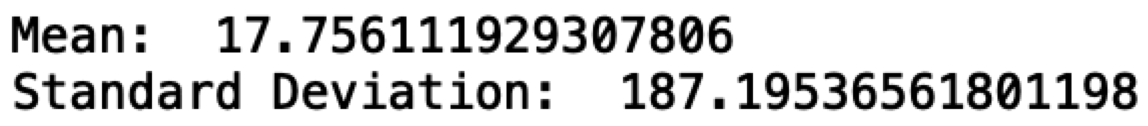
And plot the histogram:

```
plt.title("Recovered Cases Histogram")
sns.set()
df['Recovered'].hist(bins = 200)
```



And finally we can look at the sstatistics for deaths:

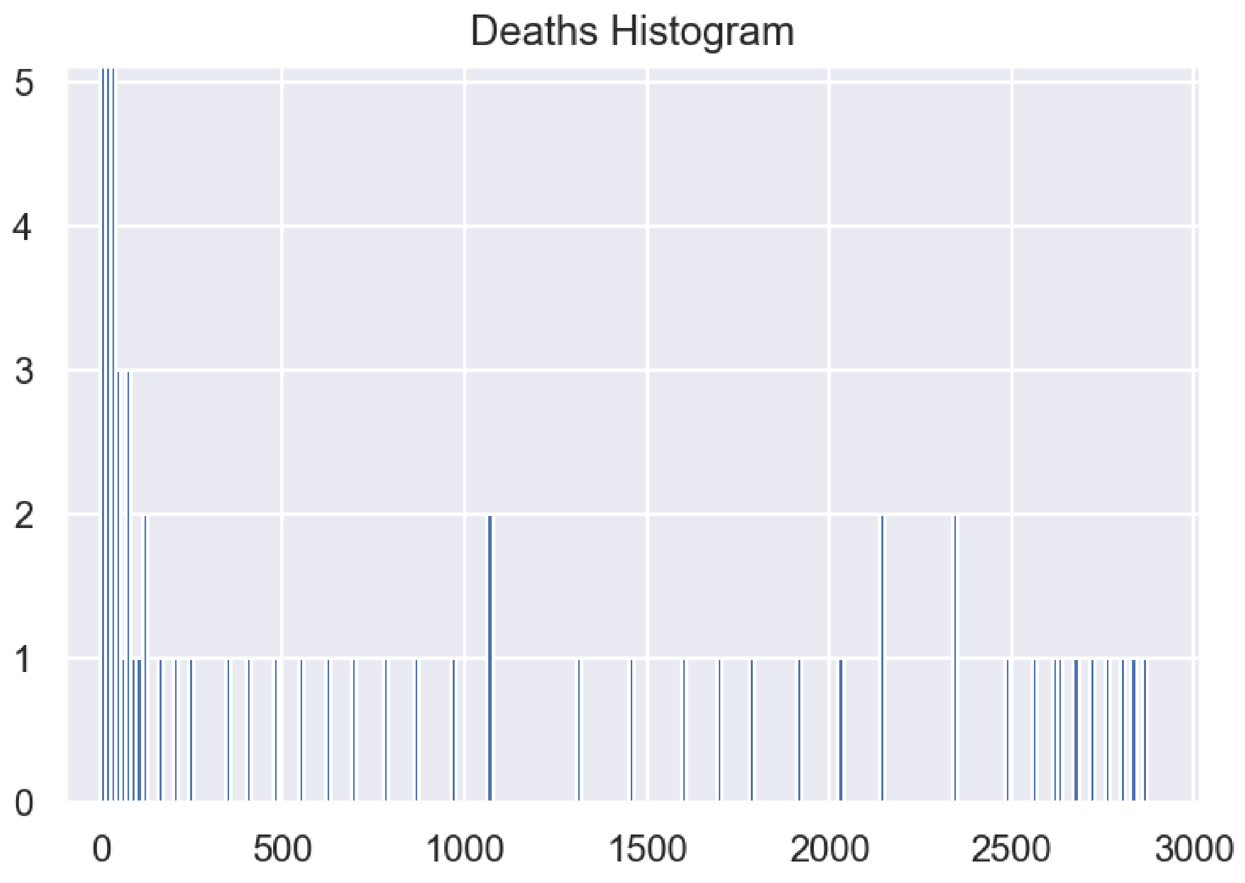
```
print("Mean: ", df['Deaths'].mean())
print("Standard Deviation: ", df['Deaths'].std())
```



```
Mean:  17.756111929307806
Standard Deviation:  187.19536561801198
```

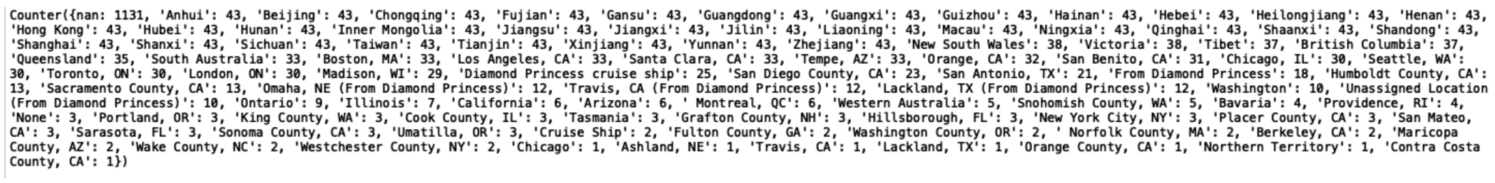
As well as the histogram:

```
plt.title("Deaths Histogram")
sns.set()
df['Deaths'].hist(bins = 200)
```



We can also look at a the frequency in Province/State using the 'Counter' method from the collections module:

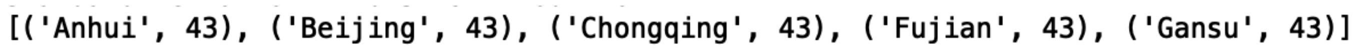
```
from collections import Counter
print(Counter(df['Province/State'].values))
```



```
Counter({nan: 1131, 'Anhui': 43, 'Beijing': 43, 'Chongqing': 43, 'Fujian': 43, 'Gansu': 43, 'Guangdong': 43, 'Guangxi': 43, 'Guizhou': 43, 'Hainan': 43, 'Hebei': 43, 'Heilongjiang': 43, 'Henan': 43,
'Hong Kong': 43, 'Hubei': 43, 'Hunan': 43, 'Inner Mongolia': 43, 'Jiangsu': 43, 'Jiangxi': 43, 'Jilin': 43, 'Liaoning': 43, 'Macau': 43, 'Ningxia': 43, 'Qinghai': 43, 'Shaanxi': 43, 'Shandong': 43,
'Shanghai': 43, 'Shanxi': 43, 'Sichuan': 43, 'Taiwan': 43, 'Tianjin': 43, 'Xinjiang': 43, 'Yunnan': 43, 'Zhejiang': 43, 'New South Wales': 38, 'Victoria': 38, 'Tibet': 37, 'British Columbia': 37,
'Queensland': 35, 'South Australia': 33, 'Boston, MA': 33, 'Los Angeles, CA': 33, 'Santa Clara, CA': 33, 'Tempe, AZ': 33, 'Orange, CA': 32, 'San Benito, CA': 31, 'Chicago, IL': 30, 'Seattle, WA':
30, 'Toronto, ON': 30, 'London, ON': 30, 'Madison, WI': 29, 'Diamond Princess cruise ship': 25, 'San Diego County, CA': 23, 'San Antonio, TX': 21, 'From Diamond Princess': 18, 'Humboldt County, CA':
13, 'Sacramento County, CA': 13, 'Omaha, NE (From Diamond Princess)': 12, 'Travis, CA (From Diamond Princess)': 12, 'Lackland, TX (From Diamond Princess)': 12, 'Washington': 10, 'Unassigned Location
(From Diamond Princess)': 10, 'Ontario': 9, 'Illinois': 7, 'California': 6, 'Arizona': 6, ' Montreal, QC': 6, 'Western Australia': 5, 'Snohomish County, WA': 5, 'Bavaria': 4, 'Providence, RI': 4,
'None': 3, 'Portland, OR': 3, 'King County, WA': 3, 'Cook County, IL': 3, 'Tasmania': 3, 'Grafton County, NH': 3, 'Hillsborough, FL': 3, 'New York City, NY': 3, 'Placer County, CA': 3, 'San Mateo,
CA': 3, 'Sarasota, FL': 3, 'Sonoma County, CA': 3, 'Umatilla, OR': 3, 'Cruise Ship': 2, 'Fulton County, GA': 2, 'Washington County, OR': 2, ' Norfolk County, MA': 2, 'Berkeley, CA': 2, 'Maricopa
County, AZ': 2, 'Wake County, NC': 2, 'Westchester County, NY': 2, 'Chicago': 1, 'Ashland, NE': 1, 'Travis, CA': 1, 'Lackland, TX': 1, 'Orange County, CA': 1, 'Northern Territory': 1, 'Contra Costa
County, CA': 1})
```

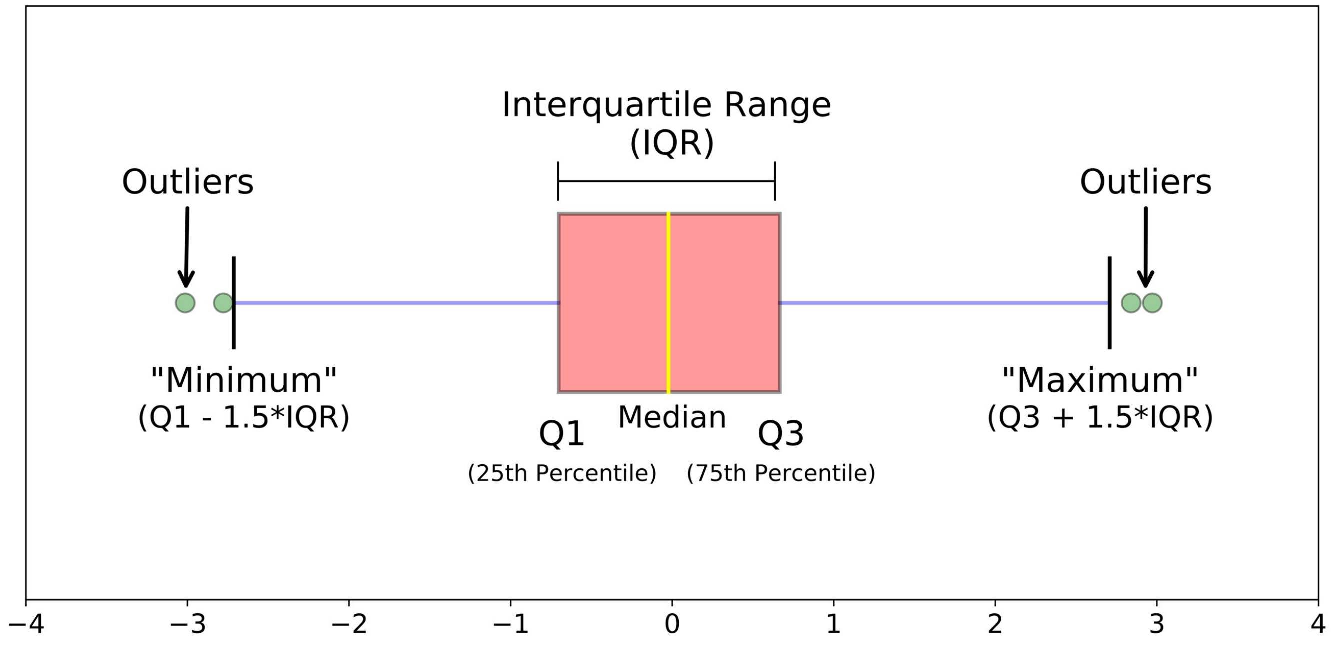
Let's drop the missing values and limit the counter to output only the five most common Provinces:

```
df.dropna(inplace=True)
print(Counter(df['Province/State'].values).most_common(5))
```



```
[('Anhui', 43), ('Beijing', 43), ('Chongqing', 43), ('Fujian', 43), ('Gansu', 43)]
```

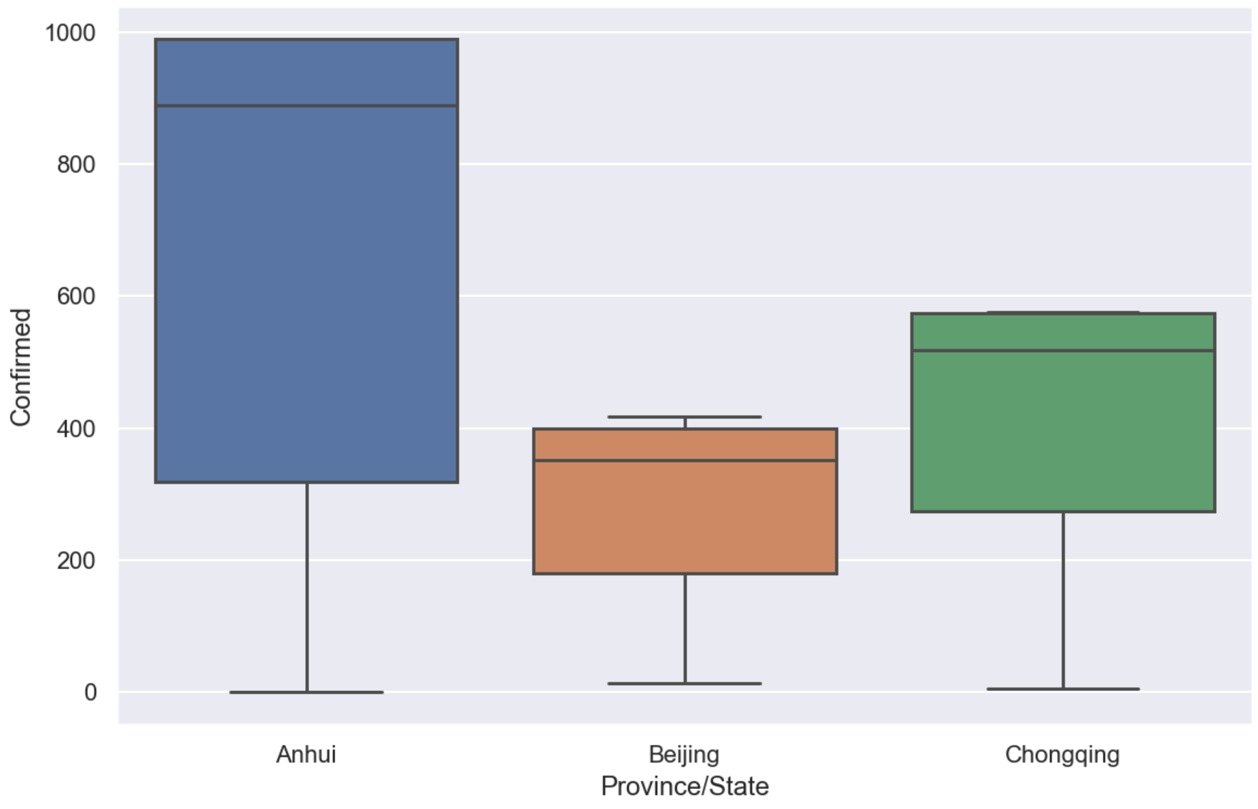
We can also use box plots to visualize the distribution in numeric values based on the minimum, maximum, median, first quartile, and third quartile. If you are unfamiliar with them take a look at the article Understanding Boxplots.



Source

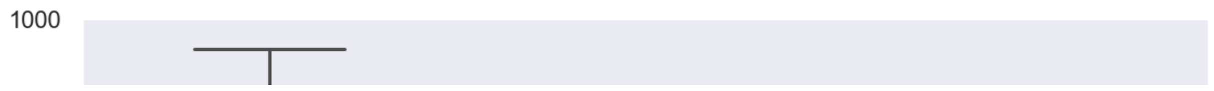
For example, let's plot the distribution in confirmed cases for 'Anhui', 'Beijing', and 'Chongqing':

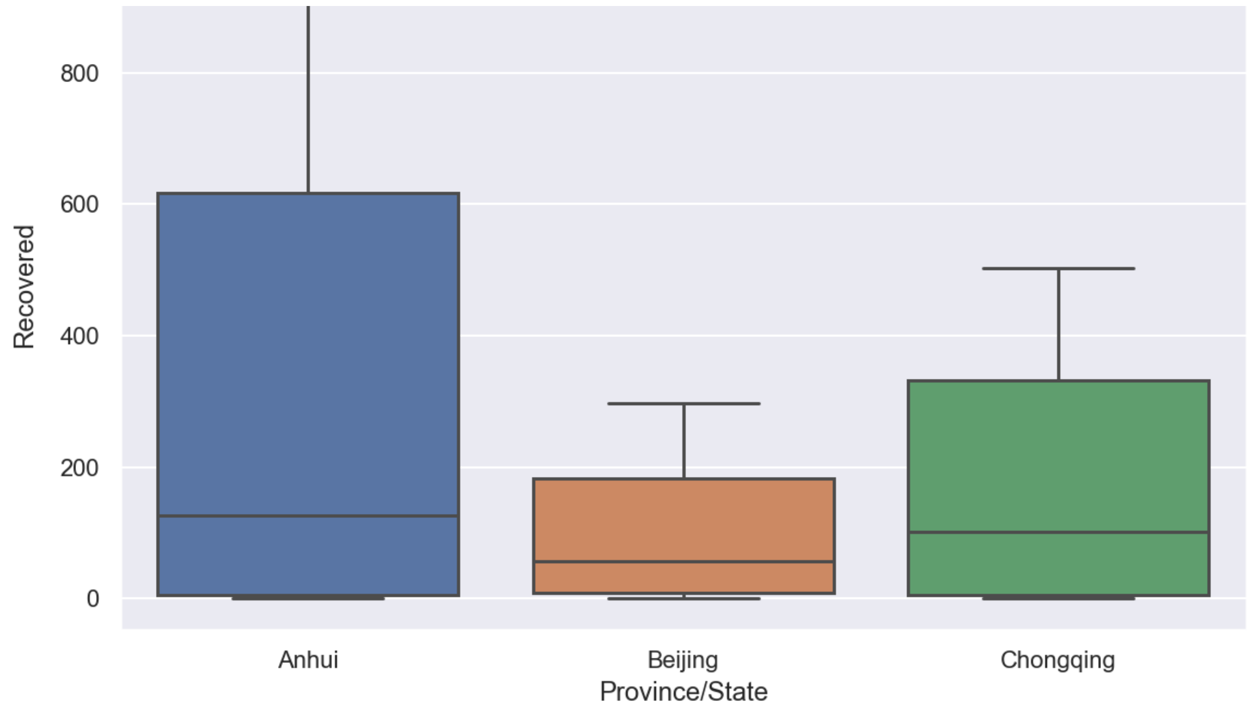
```
df = df[df['Province/State'].isin(['Anhui', 'Beijing',
'Chongqing'])]
sns.boxplot(x= df['Province/State'], y = df['Confirmed'])
plt.show()
```



We can do the same for recovered cases:

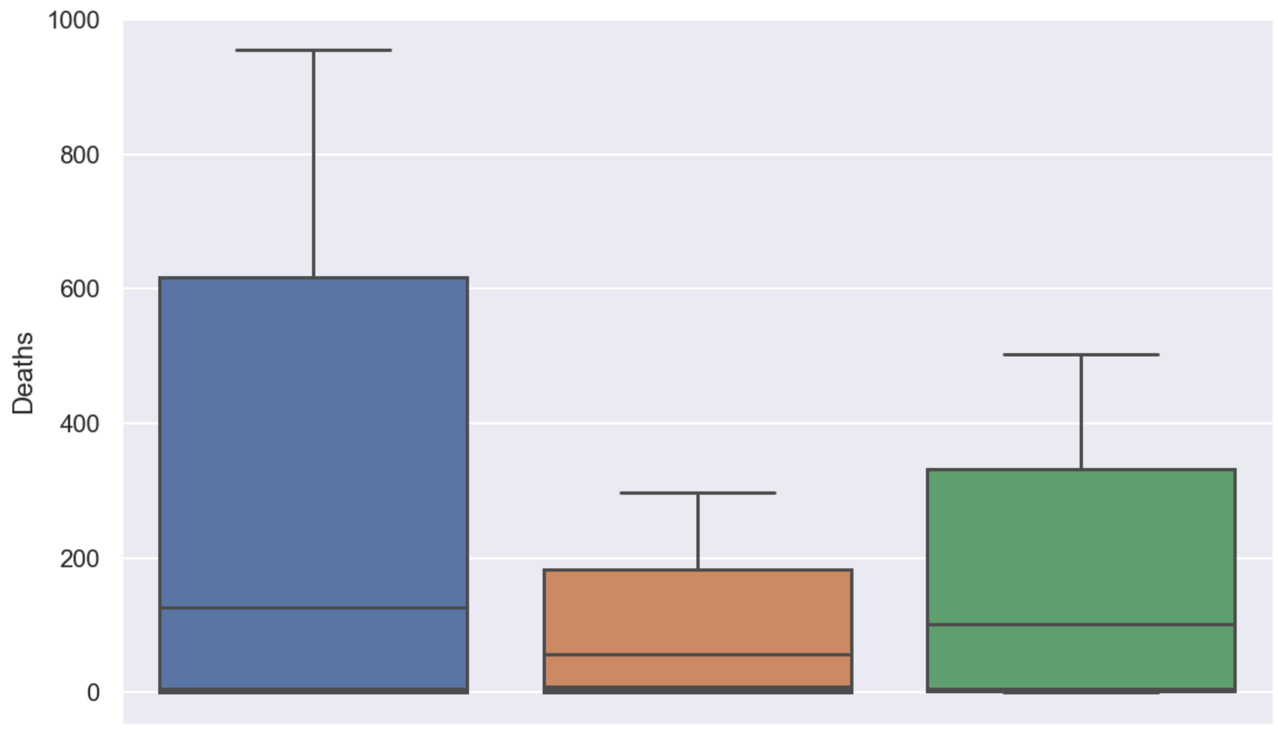
```
df = df[df['Province/State'].isin(['Anhui', 'Beijing',
'Chongqing'])]
sns.boxplot(x= df['Province/State'], y = df['Recovered'])
plt.show()
```

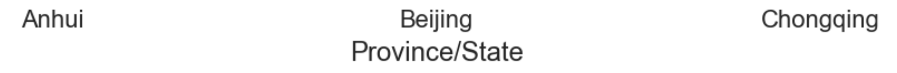




And for deaths:

```
df = df[df['Province/State'].isin(['Anhui', 'Beijing',
'Chongqing'])]
sns.boxplot(x= df['Province/State'], y = df['Deaths'])
plt.show()
```





I’ll stop here but feel free to play around with the data and code yourself. The code from this post is available on GitHub. Thank you for reading!

Public Health Data Science Python Programming Software Development