The Intersection of Computer Science and Biology
Computational Biology

Uploading pandas and csv
-Rocky Mountain Data

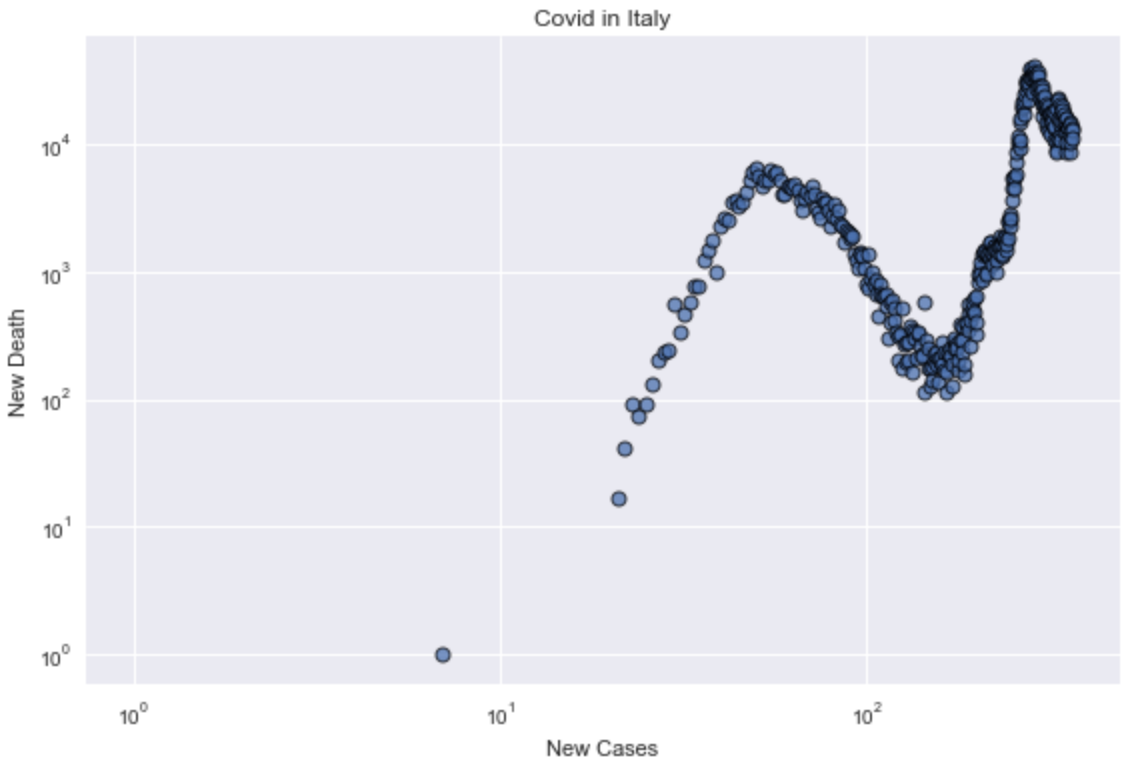
Covid Data:
Using Matplotlib
Uploading and Scrubbing CSV file
Converting to date time format
#date column to datetime
df['date']=pd.to_datetime(df['date'])
dfdata=pd.DataFrame(df,columns=['date'])
dataTypeSeries=df data.types
dataTypeSeries
df

]:

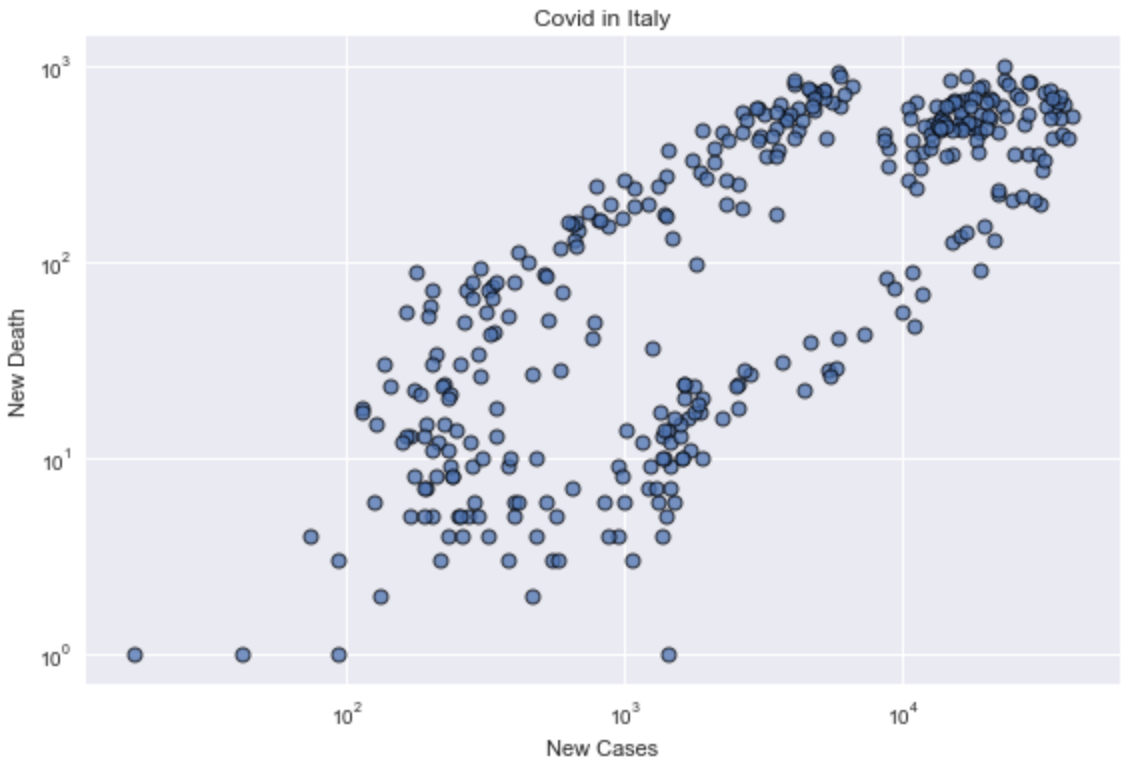
	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	tota
0	AFG	Asia	Afghanistan	2020-02-24	5.0	5.0	NaN	
1	AFG	Asia	Afghanistan	2020-02-25	5.0	0.0	NaN	
2	AFG	Asia	Afghanistan	2020-02-26	5.0	0.0	NaN	
3	AFG	Asia	Afghanistan	2020-02-27	5.0	0.0	NaN	
4	AFG	Asia	Afghanistan	2020-02-28	5.0	0.0	NaN	
...	...	...	...	...	...	...	...	
163782	ZWE	Africa	Zimbabwe	2022-02-18	233030.0	432.0	259.429	
163783	ZWE	Africa	Zimbabwe	2022-02-19	233224.0	194.0	275.000	
163784	ZWE	Africa	Zimbabwe	2022-02-20	233352.0	128.0	281.571	
163785	ZWE	Africa	Zimbabwe	2022-02-21	233571.0	219.0	281.143	
163786	ZWE	Africa	Zimbabwe	2022-02-22	233980.0	409.0	339.571	

ScatterPlot:

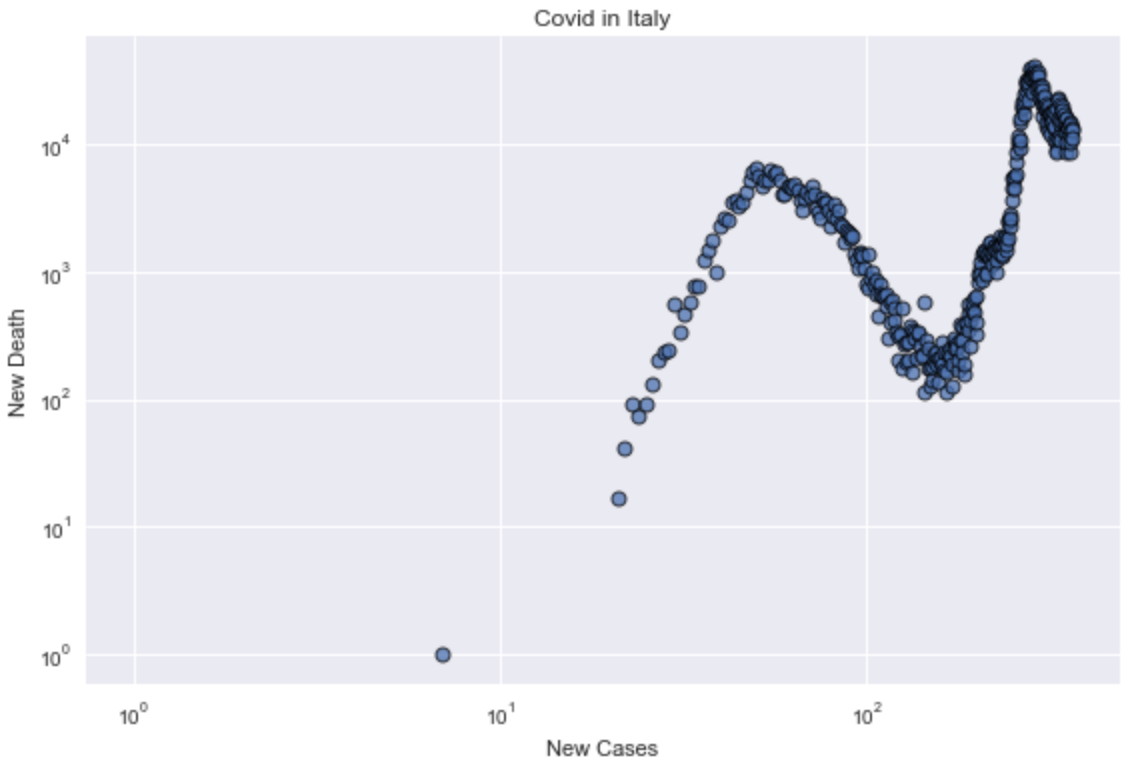
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df=pd.read_csv('mcoviddata.csv')
integer_location = np.where(df.index == 74844)[0][0]
start = max(0, integer_location - 367)
end = max(1, integer_location)
df = df.iloc[start:end]


plt.style.use('seaborn')
view_count=df['population_density']
date=df['date']
case=df['new_cases']
death=df['new_deaths']

plt.scatter(date, case, edgecolor='black', linewidth=1,alpha=0.75)

plt.xscale('log')
plt.yscale('log')

plt.title('Covid in Italy')

plt.xlabel('New Cases')

plt.ylabel('New Death')

plt.tight_layout()

plt.show()
```




```
plt.style.use('seaborn')
view_count=df['population_density']
date=df['date']
case=df['new_cases']
death=df['new_deaths']

plt.scatter(case,death, edgecolor='black', linewidth=1,alpha=0.75)

plt.xscale('log')
plt.yscale('log')

plt.title('Covid in Italy')

plt.xlabel('New Cases')

plt.ylabel('New Death')

plt.tight_layout()

plt.show()
```

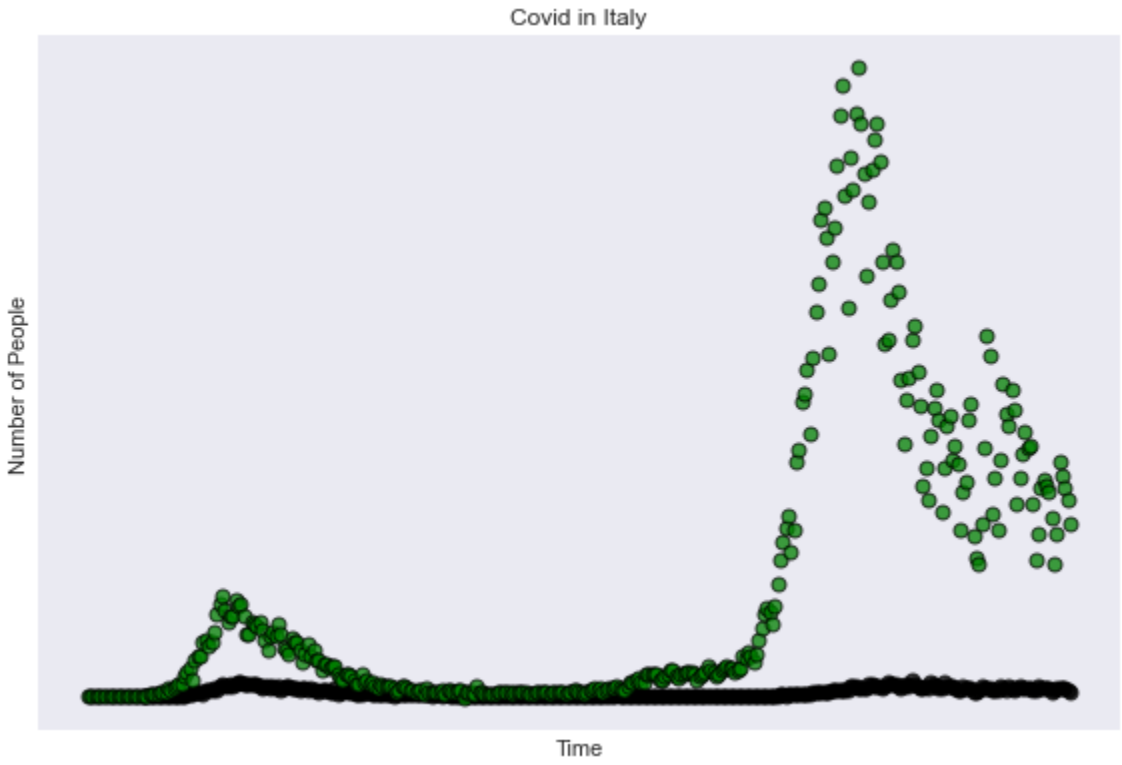



```
plt.style.use('seaborn')
view_count=df['population_density']
date=df['date']
case=df['new_cases']
death=df['new_deaths']

plt.scatter(date, case, edgecolor='black', linewidth=1,alpha=0.75)

plt.xscale('log')
plt.yscale('log')

plt.title('Covid in Italy')

plt.xlabel('New Cases')

plt.ylabel('New Death')

plt.tight_layout()

plt.show()
```



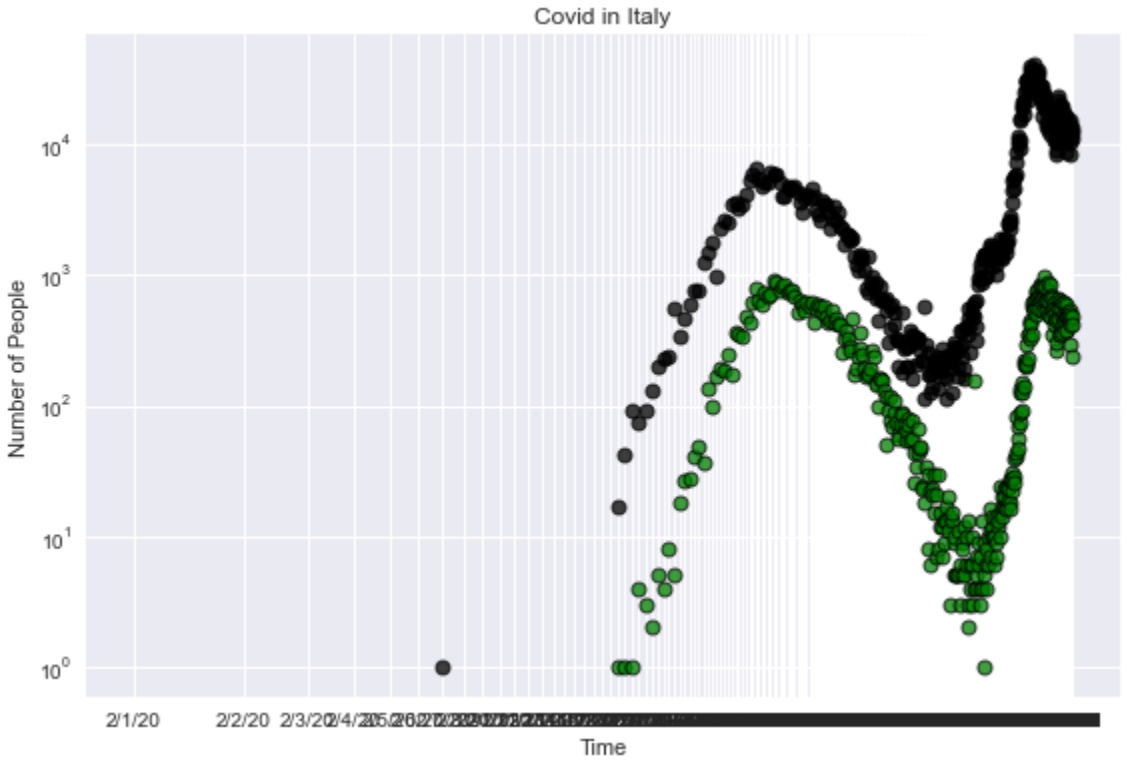

```
plt.style.use('seaborn')
view_count=df['population_density']
date=df['date']
case=df['new_cases']
death=df['new_deaths']


X=date
Y1=death
Y2=case
#plt.xscale('log')
#plt.yscale('log')

plt.xticks([])
plt.yticks([])

plt.title('Covid in Italy')

plt.xlabel('Time')

plt.ylabel('Number of People')

plt.scatter(X,Y1, color='k',edgecolor='black', linewidth=1,alpha=0.75)
```

```
plt.scatter(X,Y2,color='g',edgecolor='black', linewidth=1,alpha=0.75)
plt.tight_layout()
plt.show()
```




```
plt.style.use('seaborn')
view_count=df['population_density']
date=df['date']
case=df['new_cases']
death=df['new_deaths']

X=date
Y1=case
Y2=death
plt.xscale('log')
plt.yscale('log')



plt.title('Covid in Italy')

plt.xlabel('Time')
```

```
plt.ylabel('Number of People')

plt.scatter(X,Y1, color='k',edgecolor='black', linewidth=1,alpha=0.75)
plt.scatter(X,Y2,color='g',edgecolor='black', linewidth=1,alpha=0.75)
plt.tight_layout()
plt.show()
```



Histogram 1:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
df=pd.read_csv('mcoviddata.csv', usecols=['total_cases','date','location'])

integer_location = np.where(df.index == 74844)[0][0]
start = max(0, integer_location - 367)
end = max(1, integer_location)
dfRange = df.iloc[start:end]

print(dfRange)
```

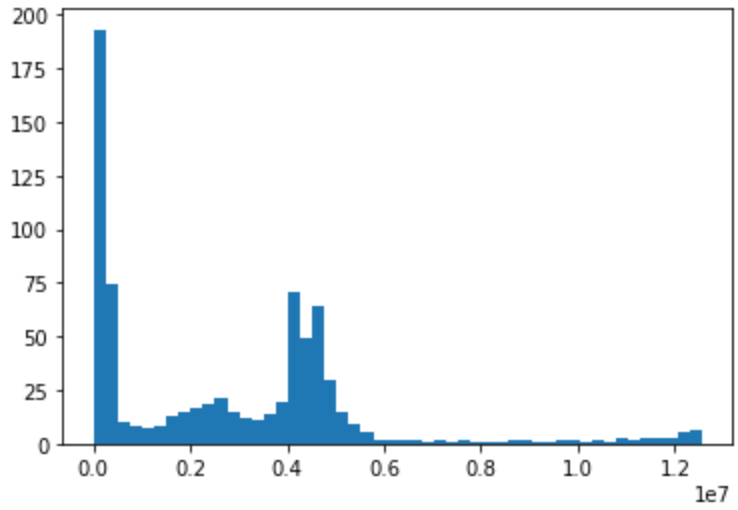
	location	date	total_cases
74477	Italy	1/31/20	2.0
74478	Italy	2/1/20	2.0
74479	Italy	2/2/20	2.0
74480	Italy	2/3/20	2.0
74481	Italy	2/4/20	2.0
...	...	...	...
74839	Italy	1/27/21	2501147.0
74840	Italy	1/28/21	2515507.0
74841	Italy	1/29/21	2529070.0
74842	Italy	1/30/21	2541783.0
74843	Italy	1/31/21	2553032.0

[367 rows x 3 columns]

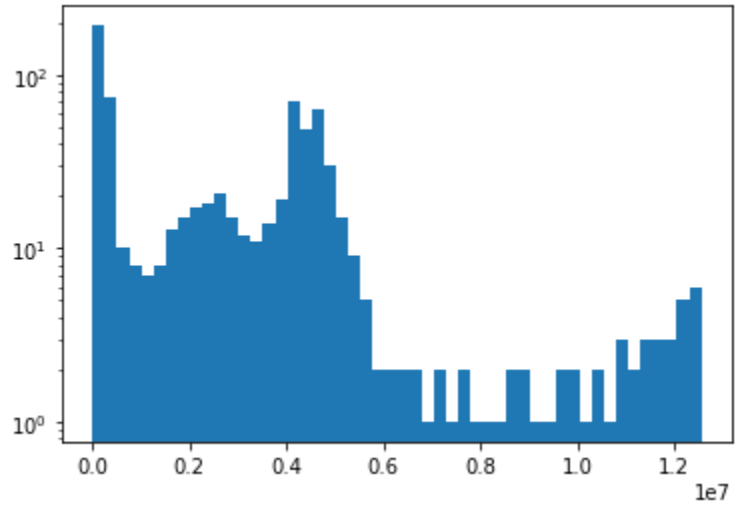
dfRange.describe()

	total_cases
count	3.670000e+02
mean	6.109483e+05
std	7.498863e+05
min	2.000000e+00
25%	2.083780e+05
50%	2.478320e+05
75%	6.943825e+05
max	2.553032e+06

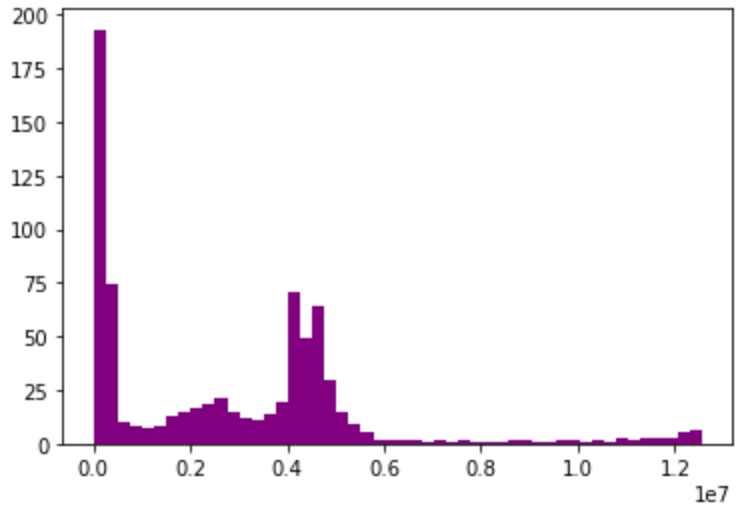
plt.hist(dfRange['total_cases'],bins=50)



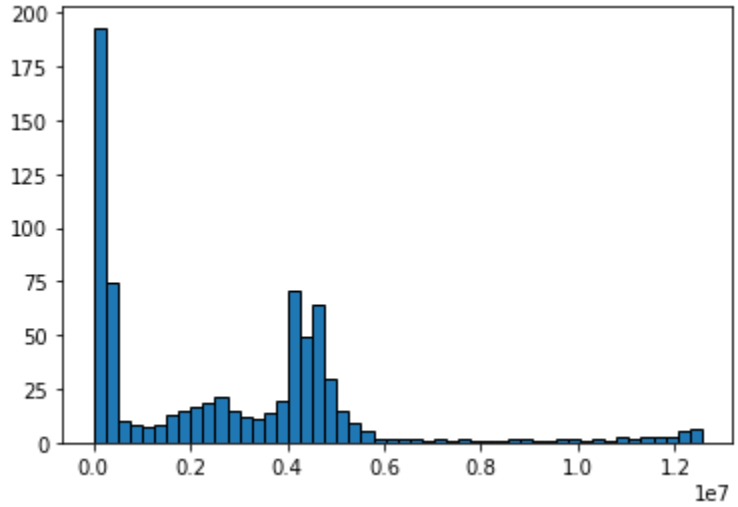
plt.hist(dfRange['total_cases'], bins=50,log=True)



plt.hist(dfRange['total_cases'], bins=50, color='purple')



plt.hist(dfRange['total_cases'], bins=50, edgecolor='black')



Histogram 2:
Italy: New Covid cases in Italy vs Date
Source:https://towardsdatascience.com/histograms-with-pythons-matplotlib-b8b768da9305
Isolating the Covid Data to Italy:

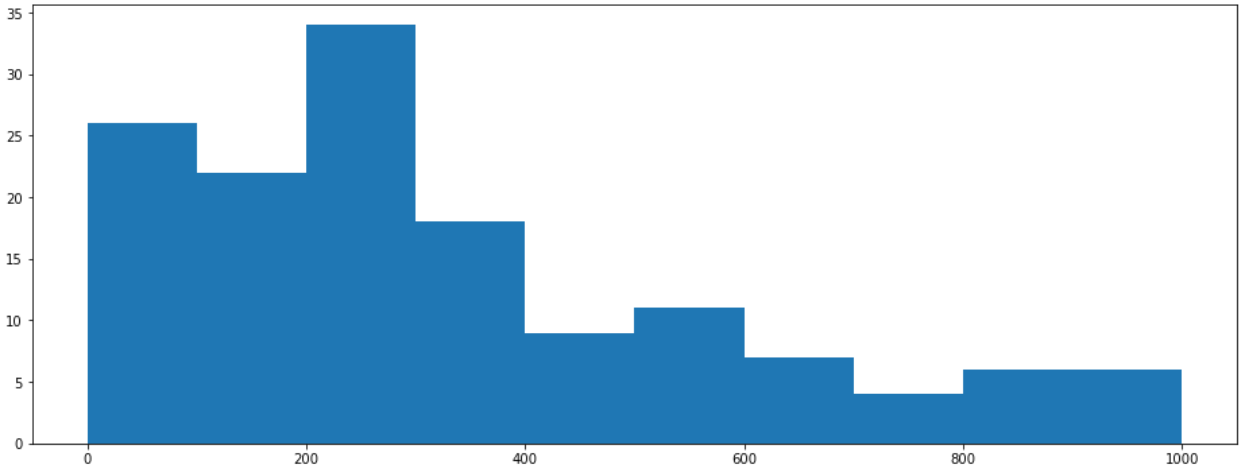
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
from matplotlib.gridspec import GridSpec
from matplotlib.ticker import AutoMinorLocator


df=pd.read_csv('mcoviddata.csv')
integer_location = np.where(df.index == 74844)[0][0]
start = max(0, integer_location - 367)
end = max(1, integer_location)
df = df.iloc[start:end]

#gathers data from 2020-2021 January 31 to January 31

df['date']=pd.to_datetime(df['date'])
dfdata=pd.DataFrame(df,columns=['date'])
dataTypeSeries=of data.types
dataTypeSeries
df

#Convert the date column to the workable datetime format
```

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	...	population	pc
74477	ITA	Europe	Italy	2020-01-31	2.0	2.0	NaN	NaN	NaN	NaN	...	60367471.0	
74478	ITA	Europe	Italy	2020-02-01	2.0	0.0	NaN	NaN	NaN	NaN	...	60367471.0	
74479	ITA	Europe	Italy	2020-02-02	2.0	0.0	NaN	NaN	NaN	NaN	...	60367471.0	
74480	ITA	Europe	Italy	2020-02-03	2.0	0.0	NaN	NaN	NaN	NaN	...	60367471.0	
74481	ITA	Europe	Italy	2020-02-04	2.0	0.0	NaN	NaN	NaN	NaN	...	60367471.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
74839	ITA	Europe	Italy	2021-01-27	2501147.0	15191.0	12425.857	86889.0	467.0	458.286	...	60367471.0	
74840	ITA	Europe	Italy	2021-01-28	2515507.0	14360.0	12469.429	87381.0	492.0	454.143	...	60367471.0	
74841	ITA	Europe	Italy	2021-01-29	2529070.0	13563.0	12459.429	87858.0	477.0	454.857	...	60367471.0	
74842	ITA	Europe	Italy	2021-01-30	2541783.0	12713.0	12371.143	88279.0	421.0	445.286	...	60367471.0	
74843	ITA	Europe	Italy	2021-01-31	2553032.0	11249.0	12317.000	88516.0	237.0	436.429	...	60367471.0	

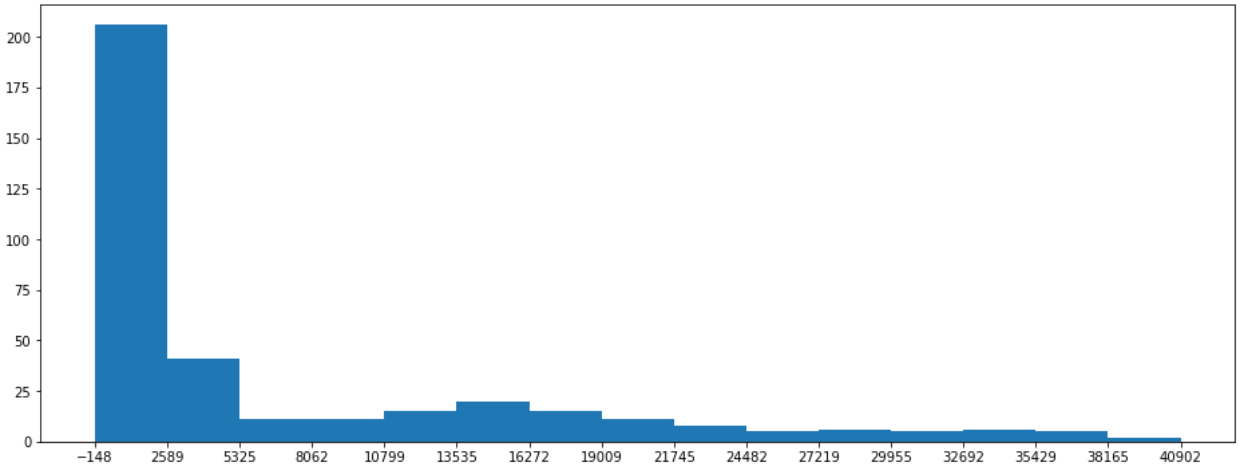
367 rows × 23 columns

```
fig=plt.figure(figsize=(16,6))
n,bins,patches=plt.hist(df.new_cases,range=[0, 1000])
#plt.xticks(bins)
plt.show()
```



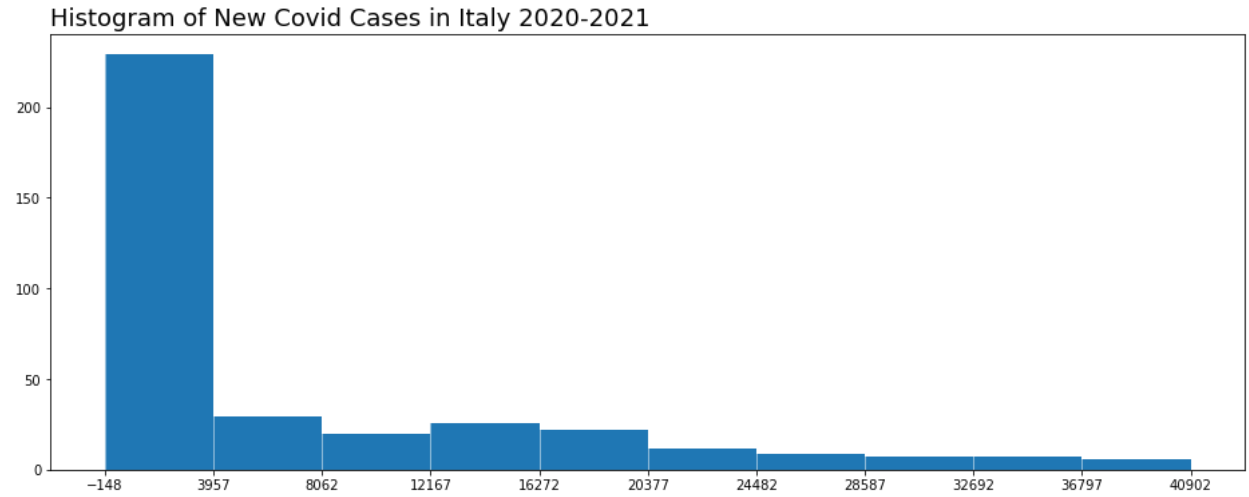
changing bin and plot size

```
fig = plt.figure(figsize=(16,6))
n, bins, patches = plt.hist(df.new_cases, bins='rice')
plt.xticks(bins)
plt.show()
```



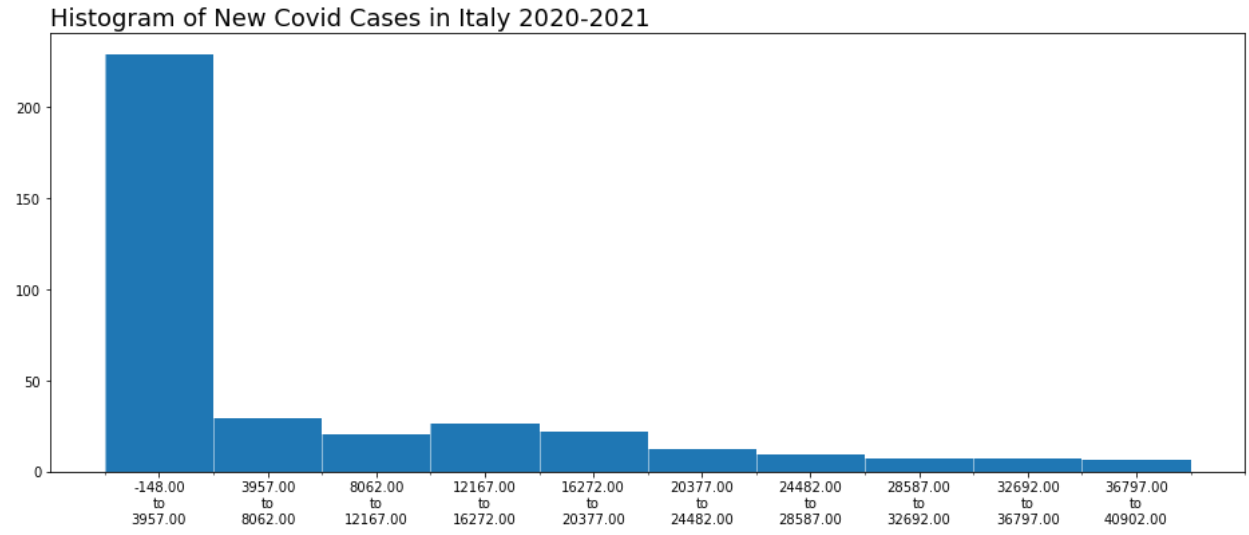

```
fig = plt.figure(figsize=(16,6))
n, bins, patches = plt.hist(df.new_cases)
plt.xticks(bins)
plt.grid(color='white', lw = 0.5, axis='x')
plt.title('Histogram of New Covid Cases in Italy 2020-2021', loc = 'left', font size = 18)
plt.show()
```

#Adding title and changing colors



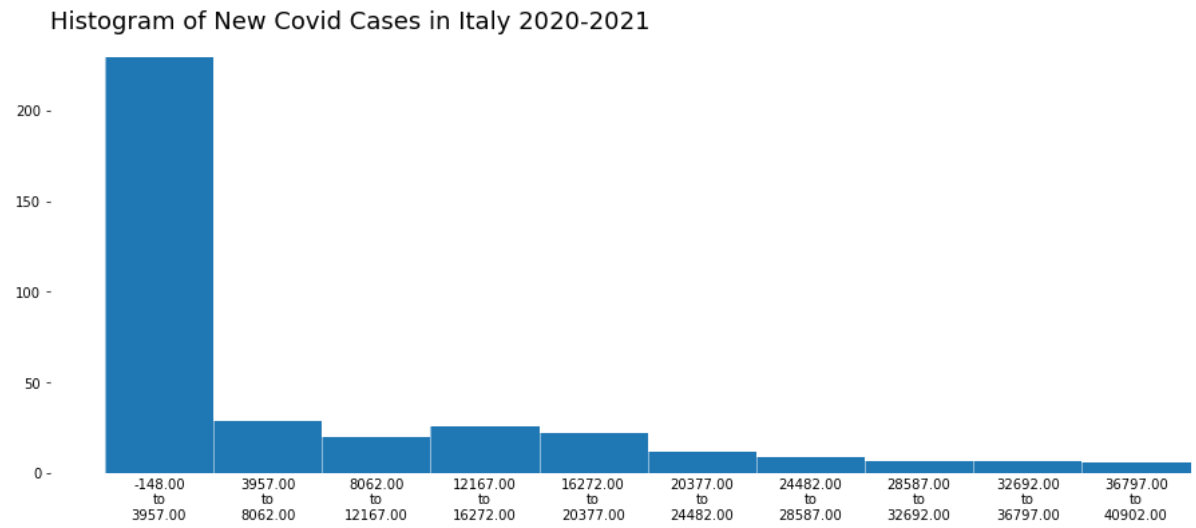

```
fig = plt.figure(figsize=(16,6))
n, bins, patches = plt.hist(df.new_cases)
# define minor ticks and draw a grid with them
minor_locator = AutoMinorLocator(2)
plt.gca().xaxis.set_minor_locator(minor_locator)
plt.grid(which='minor', color='white', lw = 0.5)
# x ticks
xticks = [(bins[idx+1] + value)/2 for idx, value in enumerate(bins[:-1])]
xticks_labels = [ "{:.2f}\nto\n{:.2f}".format(value, bins[idx+1]) for idx, value in
enumerate(bins[:-1])]
plt.xticks(xticks, labels = xticks_labels)
plt.title('Histogram of New Covid Cases in Italy 2020-2021', loc = 'left', font size = 18)
```

#putting it all together



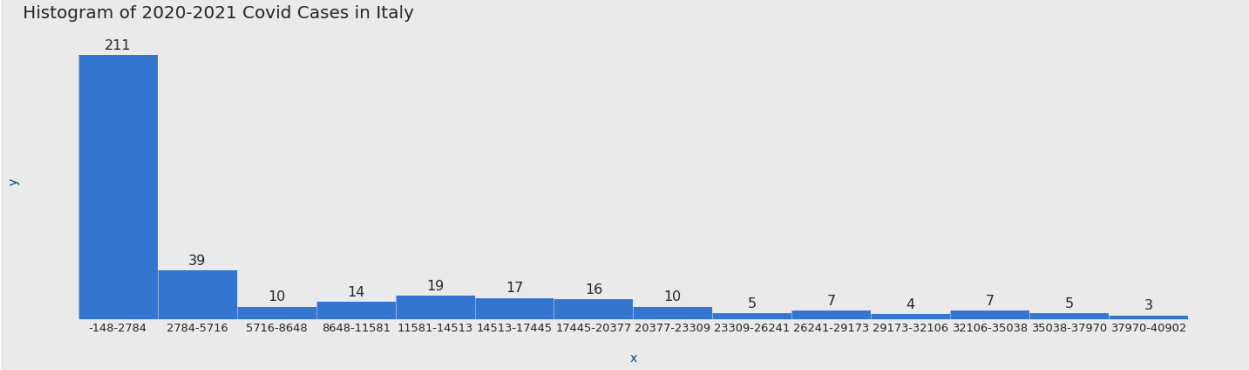

```
fig, ax = plt.subplots(1, figsize=(16,6))
n, bins, patches = plt.hist(df.new_cases)
```

```
# define minor ticks and draw a grid with them
minor_locator = AutoMinorLocator(2)
plt.gca().xaxis.set_minor_locator(minor_locator)
plt.grid(which='minor', color='white', lw = 0.5)
# x ticks
xticks = [(bins[idx+1] + value)/2 for idx, value in enumerate(bins[:-1])]
xticks_labels = [ "{:.2f}\nto\n{:.2f}".format(value, bins[idx+1]) for idx, value in
enumerate(bins[:-1])]
plt.xticks(xticks, labels = xticks_labels)
# remove major and minor ticks from the x axis, but keep the labels
ax.tick_params(axis='x', which='both',length=0)
# Hide the right and top spines
ax.spines['bottom'].set_visible(False)
ax.spines['left'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
plt.title('Histogram of New Covid Cases in Italy 2020-2021', loc = 'left', font size = 18)
```




```
color_bars = '#3475D0'
txt_color1 = '#252525'
txt_color2 = '#004C74'
fig, ax = plt.subplots(1, figsize=(20,6), facecolor=facecolor)
ax.set_facecolor(facecolor)
n, bins, patches = plt.hist(df.new_cases, color=color_bars, bins='doane')
#grid
minor_locator = AutoMinorLocator(2)
```

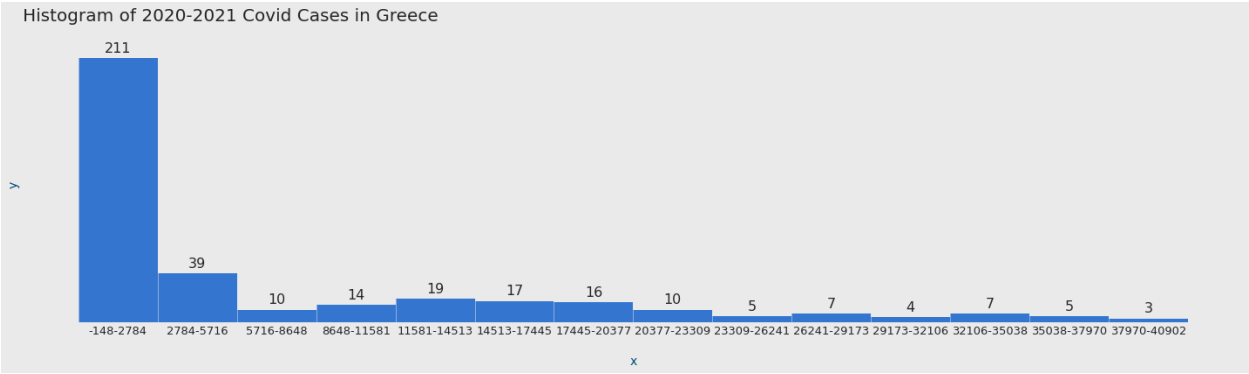
```
plt.gca().xaxis.set_minor_locator(minor_locator)
plt.grid(which='minor', color=facecolor, lw = 0.5)
xticks = [(bins[idx+1] + value)/2 for idx, value in enumerate(bins[:-1])]
xticks_labels = [ "{:.0f}-{:.0f}".format(value, bins[idx+1]) for idx, value in enumerate(bins[:-1])]
plt.xticks(xticks, labels=xticks_labels, c=txt_color1, font size=13)
# remove major and minor ticks from the x axis, but keep the labels
ax.tick_params(axis='x', which='both',length=0)
# remove y ticks
plt.yticks([])
# Hide the right and top spines
ax.spines['bottom'].set_visible(False)
ax.spines['left'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
for idx, value in enumerate(n):
    if value > 0:
        plt.text(xticks[idx], value+5, int(value), ha='center', fontsize=16, c=txt_color1)
plt.title('Histogram of 2020-2021 Covid Cases in Italy\n', loc = 'left', fontsize = 20, c=txt_color1)
plt.xlabel('\nx', c=txt_color2, fontsize=14)
plt.ylabel('y', c=txt_color2, fontsize=14)
plt.tight_layout()
plt.savefig('costs.png', facecolor=facecolor)
```



Greece:

```
#date column to datetime
df['date']=pd.to_datetime(df['date'])
dfdata=pd.DataFrame(df,columns=['date'])
dataTypeSeries=of data.types
dataTypeSeries
Df
```

```
facecolor = '#EAEAEA'
color_bars = '#3475D0'
txt_color1 = '#252525'
txt_color2 = '#004C74'
fig, ax = plt.subplots(1, figsize=(20,6), facecolor=facecolor)
ax.set_facecolor(facecolor)
n, bins, patches = plt.hist(df.new_cases, color=color_bars, bins='doane')
#grid
minor_locator = AutoMinorLocator(2)
plt.gca().xaxis.set_minor_locator(minor_locator)
plt.grid(which='minor', color=facecolor, lw = 0.5)
xticks = [(bins[idx+1] + value)/2 for idx, value in enumerate(bins[:-1])]
xticks_labels = [ "{:.0f}-{:.0f}".format(value, bins[idx+1]) for idx, value in enumerate(bins[:-1])]
plt.xticks(xticks, labels=xticks_labels, c=txt_color1, font size=13)
# remove major and minor ticks from the x axis, but keep the labels
ax.tick_params(axis='x', which='both',length=0)
# remove y ticks
plt.yticks([])
# Hide the right and top spines
ax.spines['bottom'].set_visible(False)
ax.spines['left'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
for idx, value in enumerate(n):
    if value > 0:
        plt.text(xticks[idx], value+5, int(value), ha='center', fontsize=16, c=txt_color1)
plt.title('Histogram of 2020-2021 Covid Cases in Greece\n', loc = 'left', fontsize = 20,
c=txt_color1)
plt.xlabel('\nx', c=txt_color2, fontsize=14)
plt.ylabel('y', c=txt_color2, fontsize=14)
plt.tight_layout()
plt.savefig('greece.png', facecolor=facecolor)
```



Vaccination rates
Histogram

Scatterplot

Bioinformatics:
1000 Year Journey

- Input: A DNA string genome
- Output: The location of ori in the genome
    - Origin of replication
- Input a string and output a count

Hidden messages in the replication origin

- How doe the bacterial cell know to begin replication
    - Initiation is mediated by DnaA
        - Protein that binds to a short segment with the ori (DnaA box)
- Problem: find the hidden message in the replication origin
    - Input: string (replication origin of the genome)
    - Output: a hidden message text
- K-mer: a string of length k and Count number of times that the pattern appears in a substring
- Text goes through indexing where the k-mer pattern begins
- Count(text,pattern)

```
PatternCount (text, pattern)
        count=0
        For i in 0 to |text|-|pattern|
                If Text(i, |Pattern|)=Pattern
                        count+=1
        Return count
```

-Bio python
        -bio.seq

```
import Bio
from Bio.Seq import Seq

my_seq=Seq("CAT TAGATAG")

print("seq %s is %i bases long" % (my_seq, len(my_seq)))
print("reverse complement is %s" % my_seq.reverse_complement())
```

print("protein translation is %s" % my_seq.translate())

-Burnt Pancake problem