Stage2 Member

March 13, 2023

[24]:
```
import pandas as pd
import os
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import datetime as dt
from  scipy.signal import find_peaks
```

[2]:
```
#checking if the path exists for the given link
import os
#"C:\Users\ashdh\Documents\GitHub\Ferocious5_CS605\DATASETS\ENRICHMENT␣
 ↪DATASETS\ACSDP1Y2021.DP04-DataHousing.csv"
relative_path = r"..\..\..\DATASETS\COVID DATASETS\covid_confirmed_usafacts.csv"

if os.path.exists(relative_path):
    print("The file exists at the specified path.")
else:
    print("The file does not exist at the specified path.")
```

```
The file exists at the specified path.
```

0.0.1 Part 1 - Generate weekly statistics (mean, median, mode) for number of new cases and deaths across a specific state.

[3]:
```
'''Reading data from files and initializing for the state of New Jersey'''
#reading the data for the confirmed number of COVID cases and displaying them
confirmedCases = pd.read_csv(r"..\..\..\DATASETS\COVID␣
 ↪DATASETS\covid_confirmed_usafacts.csv")

# reading the confirmed number of COVID Deaths and displaying them
confirmedDeaths = pd.read_csv(r"..\..\..\DATASETS\COVID␣
 ↪DATASETS\covid_deaths_usafacts.csv")

#Extracting the last week of data from the given set(number of confirmed cases)
confirmedCasesNJ = confirmedCases[confirmedCases["State"] == "NJ"]
#confirmedCasesNJ

#Extracting the last week of data from the given set(number of deaths)
```

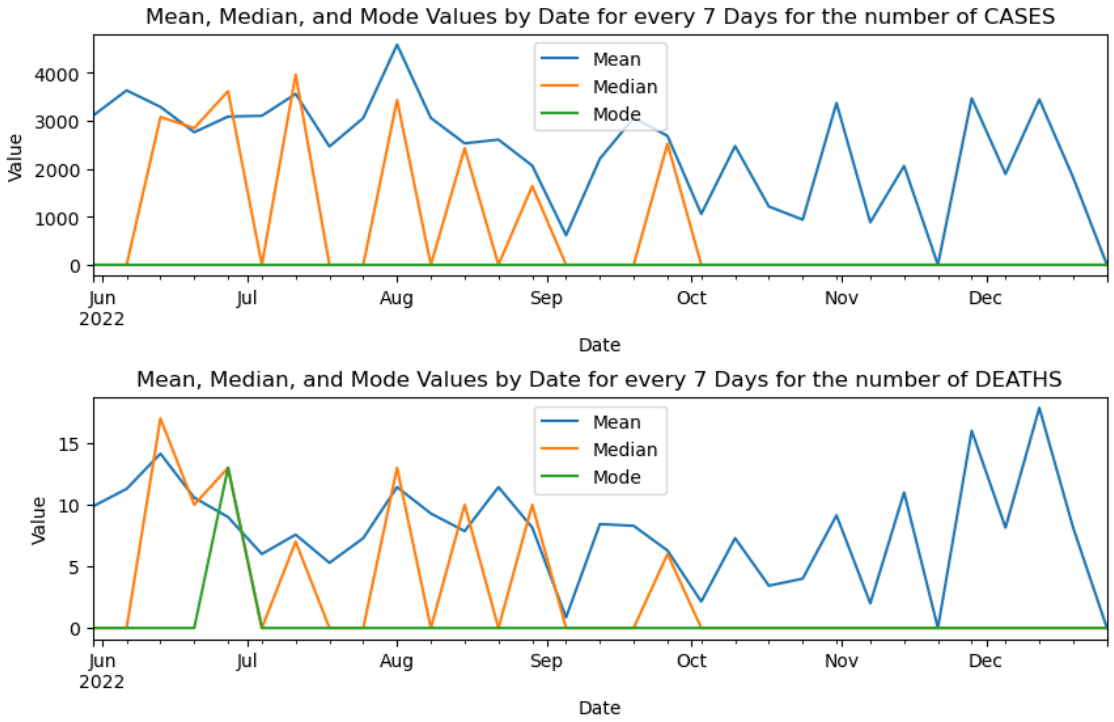
```
confirmedDeathsNJ = confirmedDeaths[confirmedDeaths["State"] == "NJ"]
```

[4]:
```
'''FOR NUMBER OF CASES'''
x = confirmedCasesNJ.iloc[:, 4:].fillna(0)
confirmedCasesNJ_integral = x.diff(axis = 1).drop(index = 1804)
#confirmedCasesNJ_integral

confirmendCasesNJNew1 = confirmedCasesNJ.iloc[:, :3]

# Delete columns outside the date range of June 1st to December 31st
confirmedCasesNJnew2 =  confirmedCasesNJ_integral.loc[:, '2022-05-30':
 ↪'2023-01-01']
confirmedCasesNJnew2
confirmendCasesNJNew = pd.concat([confirmendCasesNJNew1, confirmedCasesNJnew2],␣
 ↪axis=1).drop(index = 1804)
#confirmendCasesNJNew

ccnj =  confirmedCasesNJ_integral.loc[:, '2022-05-30':'2023-01-01']
ccnjSum = ccnj.sum()
#ccnjSum
#ccnjSum.to_csv(r"C:
 ↪\Users\ashdh\Documents\GitHub\Ferocious5_CS605\Member\AshritaD\STAGE␣
 ↪2\CCNJSum.csv")

'''FOR NUMBER OF DEATHS'''

y = confirmedDeathsNJ.iloc[:, 4:].fillna(0)
confirmedDeathsNJ_integral = y.diff(axis = 1).drop(index = 1804)
#confirmedDeathsNJ_integral.sum(axis =1 )

confirmedDeathsNJNew1 = confirmedDeathsNJ.iloc[:, :3]

# Delete columns outside the date range of June 1st to December 31st
confirmedDeathsNJnew2 =  confirmedDeathsNJ_integral.loc[:, '2022-05-30':
 ↪'2023-01-01']
confirmedDeathsNJnew2

confirmedDeathsNJNew = pd.concat([confirmedDeathsNJNew1,␣
 ↪confirmedDeathsNJnew2], axis=1).drop(index = 1804)
#confirmedDeathsNJNew

cdnj =  confirmedDeathsNJ_integral.loc[:, '2022-05-30':'2023-01-01']
cdnjSum = cdnj.sum()
```

[5]:
```
'''FOR PLOTTING THE SUBMER OF CASES AND THE NUMBER OF DEATHS IN GRAPH BELOW␣
 ↪EACH OTHER USING SUBPLOTS CONCEPT'''
```

```
#NUMBER OF CASES
date_range = pd.date_range(start='2022-05-30', end='2023-01-01')
ccnjSum.index = date_range

ccnjSum.index = pd.to_datetime(ccnjSum.index)
#ccnj1.index = ccnj1.index.strftime('%Y-%m-%d')
ccnj_resampled = ccnjSum.resample('7D')

ccnj_resampled_mean = ccnj_resampled.mean()
ccnj_resampled_mean.columns = ['Mean']
ccnj_resampled_median = ccnj_resampled.median()
ccnj_resampled_median.columns = ['Median']
ccnj_resampled_mode = ccnj_resampled.apply(lambda mnj: mnj.mode().iloc[0])


#NUMBER OF DEATHS
date_range = pd.date_range(start='2022-05-30', end='2023-01-01')
cdnjSum.index = date_range

#cdnjSum.index = cdnjSum.index.strftime('%Y-%m-%d')
cdnjSum.index = pd.to_datetime(cdnjSum.index)

cdnj_resampled = cdnjSum.resample('7D')

cdnj_resampled_mean = cdnj_resampled.mean()
cdnj_resampled_mean.columns = ['Mean']
cdnj_resampled_median = cdnj_resampled.median()
cdnj_resampled_median.columns = ['Median']
cdnj_resampled_mode = cdnj_resampled.apply(lambda mdnj: mdnj.mode().iloc[0])

#PLOTTING THE DATA
# Create a new figure and axis object
fig, ax = plt.subplots(2, 1, figsize=(10, 6))

# Plot the mean,median and mode for the number of cases ax[0] data as a line␣
 ↪chart
ccnj_resampled_mean.plot(kind='line', ax=ax[0], label='Mean')
ccnj_resampled_median.plot(kind='line', ax=ax[0], label='Median')
ccnj_resampled_mode.plot(kind='line', ax=ax[0], label='Mode')

# Plot the mean,median and mode for the number of deaths ax[1] data as a line␣
 ↪chart
cdnj_resampled_mean.plot(kind='line', ax=ax[1], label='Mean')
cdnj_resampled_median.plot(kind='line', ax=ax[1], label='Median')
cdnj_resampled_mode.plot(kind='line', ax=ax[1], label='Mode')
```

```
# Set the axis labels and title
ax[0].set_xlabel('Date')
ax[0].set_ylabel('Value')
ax[0].set_title('Mean, Median, and Mode Values by Date for every 7 Days for the␣
 ↪number of CASES')

ax[1].set_xlabel('Date')
ax[1].set_ylabel('Value')
ax[1].set_title('Mean, Median, and Mode Values by Date for every 7 Days for the␣
 ↪number of DEATHS')

#adding padding between the graphs
fig.subplots_adjust(hspace=0.5)

# Add a legend to the plot
ax[0].legend()
ax[1].legend()
# Show the plot
plt.show()
```



0.0.2 Part 2 - Compare the data against 3 other states. Normalize by population, use a normalization factor which is able to identify cases and deaths, for example try per 10,000 or 100,000 (this depends on the population). Plot the values across the weeks in a line plot for the 3 states in a single graph.

[6]:
```
'''FOR CONFIRMED CASES'''
#reading the data for the confirmed number of COVID cases and displaying them
confirmedCases = pd.read_csv(r"..\..\..\DATASETS\COVID␣
 ↪DATASETS\covid_confirmed_usafacts.csv")

#Extracting the last week of data from the given set(number of confirmed cases)
confirmedCasesMD = confirmedCases[confirmedCases["State"] == "MD"] #MARYLAND
confirmedCasesNH = confirmedCases[confirmedCases["State"] == "NH"] #NEW␣
 ↪HAMSHIRE
confirmedCasesVT = confirmedCases[confirmedCases["State"] == "VT"] #VERMONT
#confirmedCasesNJ

'''FOR CONFIRMED DEATHS'''
# reading the confirmed number of COVID Deaths and displaying them
confirmedDeaths = pd.read_csv(r"..\..\..\DATASETS\COVID␣
 ↪DATASETS\covid_deaths_usafacts.csv")

#Extracting the last week of data from the given set(number of deaths)
confirmedDeathsMD = confirmedDeaths[confirmedDeaths["State"] == "MD"]
confirmedDeathsNH = confirmedDeaths[confirmedDeaths["State"] == "NH"]
confirmedDeathsVT = confirmedDeaths[confirmedDeaths["State"] == "VT"]
#confirmedCasesMD
```

[7]:
```
'''FOR CONFIRMED CASES'''

md = confirmedCasesMD.iloc[:, 4:].fillna(0)
confirmedCasesMD_integral = md.diff(axis = 1).drop(index = 1213)

nh = confirmedCasesNH.iloc[:, 4:].fillna(0)
confirmedCasesNH_integral = nh.diff(axis = 1).drop(index = 1793)

vt = confirmedCasesVT.iloc[:, 4:].fillna(0)
confirmedCasesVT_integral = vt.diff(axis = 1).drop(index = 2851)


'''FOR CONFIRMED DEATHS'''
mdD = confirmedDeathsMD.iloc[:, 4:].fillna(0)
confirmedDeathsMD_integral = mdD.diff(axis = 1).drop(index = 1213)

nhD = confirmedDeathsNH.iloc[:, 4:].fillna(0)
confirmedDeathsNH_integral = nhD.diff(axis = 1).drop(index = 1793)
```

```
vtD = confirmedDeathsVT.iloc[:, 4:].fillna(0)
confirmedDeathsVT_integral = vtD.diff(axis = 1).drop(index = 2851)
#confirmedDeathsVT_integral
```

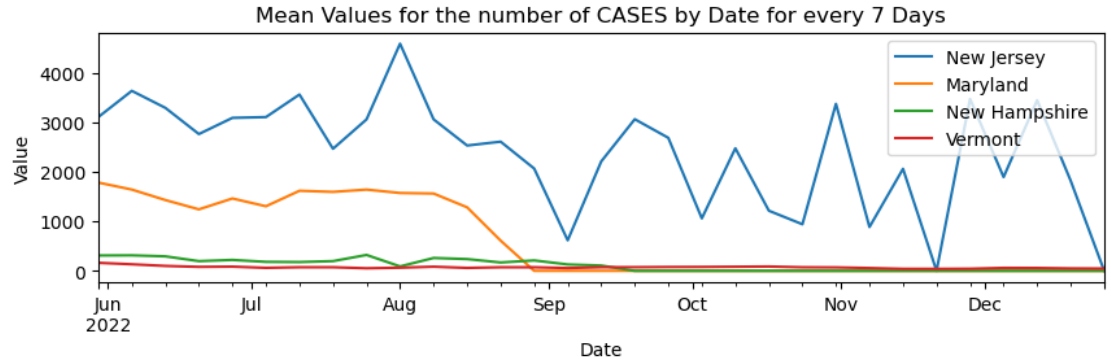
[8]:
```
'''FOR CONFIRMED CASES'''
#Maine
confirmendCasesMDNew1 = confirmedCasesMD.iloc[:, :3]
# Delete columns outside the date range of June 1st to December 31st
confirmedCasesMDnew2 =  confirmedCasesMD_integral.loc[:, '2022-05-30':
 ↪'2023-01-01']
confirmedCasesMDnew2
confirmendCasesMDNew = pd.concat([confirmendCasesMDNew1, confirmedCasesMDnew2],␣
 ↪axis=1).drop(index = 1213)
confirmendCasesMDNew

#New Hampshire
confirmendCasesNHNew1 = confirmedCasesNH.iloc[:, :3]
# Delete columns outside the date range of June 1st to December 31st
confirmedCasesNHnew2 =  confirmedCasesNH_integral.loc[:, '2022-05-30':
 ↪'2023-01-01']
confirmedCasesNHnew2
confirmendCasesNHNew = pd.concat([confirmendCasesNHNew1, confirmedCasesNHnew2],␣
 ↪axis=1).drop(index = 1793)
confirmendCasesNHNew

#Vermont
confirmendCasesVTNew1 = confirmedCasesVT.iloc[:, :3]
# Delete columns outside the date range of June 1st to December 31st
confirmedCasesVTnew2 =  confirmedCasesVT_integral.loc[:, '2022-05-30':
 ↪'2023-01-01']
confirmedCasesVTnew2
confirmendCasesVTNew = pd.concat([confirmendCasesVTNew1, confirmedCasesVTnew2],␣
 ↪axis=1).drop(index = 2851)
#confirmendCasesVTNew

'''FOR CONFIRMED DEATHS'''
#Maine
confirmendDeathsMDNew1 = confirmedDeathsMD.iloc[:, :3]
# Delete columns outside the date range of June 1st to December 31st
confirmedDeathsMDnew2 =  confirmedDeathsMD_integral.loc[:, '2022-05-30':
 ↪'2023-01-01']
confirmedDeathsMDnew2
confirmendDeathsMDNew = pd.concat([confirmendDeathsMDNew1,␣
 ↪confirmedDeathsMDnew2], axis=1).drop(index = 1213)
confirmendDeathsMDNew
```

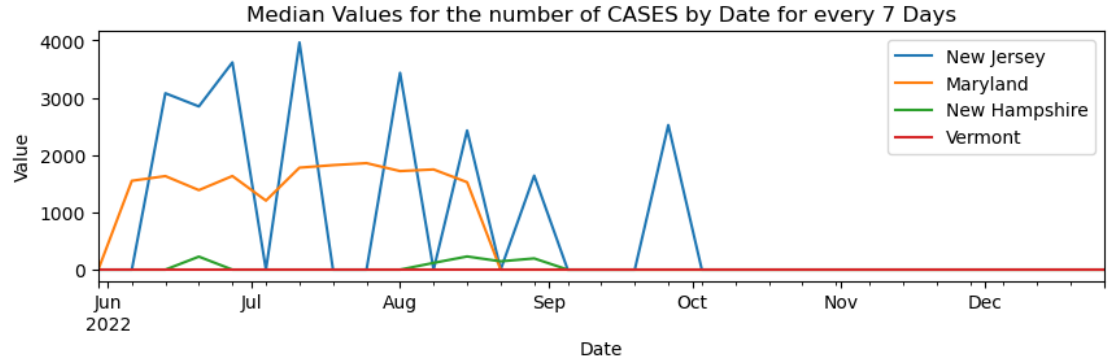
```
#New Hampshire
confirmendDeathsNHNew1 = confirmedDeathsNH.iloc[:, :3]
# Delete columns outside the date range of June 1st to December 31st
confirmedDeathsNHnew2 =  confirmedDeathsNH_integral.loc[:, '2022-05-30':
 ↪'2023-01-01']
confirmedDeathsNHnew2
confirmendDeathsNHNew = pd.concat([confirmendDeathsNHNew1,␣
 ↪confirmedDeathsNHnew2], axis=1).drop(index = 1793)
confirmendDeathsNHNew

#Vermont
confirmendDeathsVTNew1 = confirmedDeathsVT.iloc[:, :3]
# Delete columns outside the date range of June 1st to December 31st
confirmedDeathsVTnew2 =  confirmedDeathsVT_integral.loc[:, '2022-05-30':
 ↪'2023-01-01']
confirmedDeathsVTnew2
confirmendDeathsVTNew = pd.concat([confirmendDeathsVTNew1,␣
 ↪confirmedDeathsVTnew2], axis=1).drop(index = 2851)
```

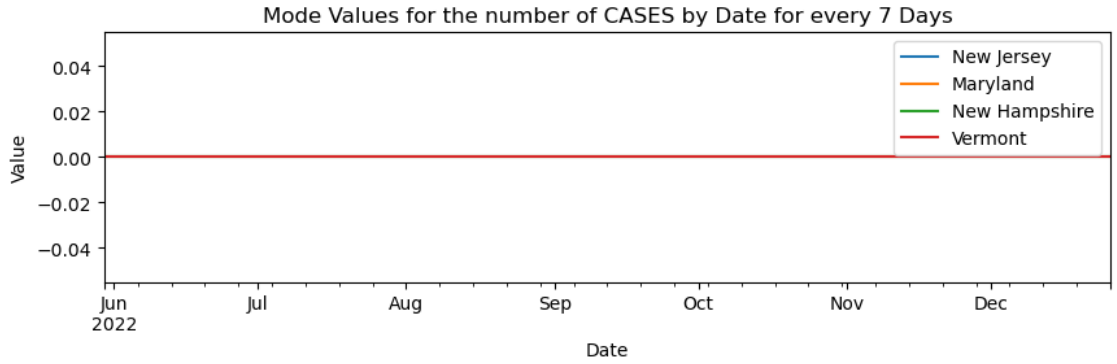
[9]:
```
'''COMPARING THE MEAN, MEDIAN AND MODE FOR THE STATES OF NEW JERSEY, MAINE, NEW␣
 ↪HAMPSHIRE, VERMONT (NUMBER OF CASES)'''

date_range = pd.date_range(start='2022-05-30', end='2023-01-01')
'''#NEW JERSEY NUMBER OF CASES
ccnj =  confirmedCasesNJ_integral.loc[:, '2022-05-30':'2023-01-01']
ccnjSum = ccnj.sum()
ccnjSum.index = date_range
ccnjSum.index = pd.to_datetime(ccnjSum.index)
ccnj_resampled = ccnjSum.resample('7D')
ccnj_resampled_mean = ccnj_resampled.mean()
ccnj_resampled_mean.columns = ['Mean']
ccnj_resampled_median = ccnj_resampled.median()
ccnj_resampled_median.columns = ['Median']
ccnj_resampled_mode = ccnj_resampled.apply(lambda mcnj: mcnj.mode().iloc[0])'''

#MARYLAND NUMBER OF CASES
ccmd =  confirmedCasesMD_integral.loc[:, '2022-05-30':'2023-01-01']
ccmdSum = ccmd.sum()
ccmdSum.index = date_range
ccmdSum.index = pd.to_datetime(ccmdSum.index)
ccmd_resampled = ccmdSum.resample('7D')
ccmd_resampled_mean = ccmd_resampled.mean()
ccmd_resampled_mean.columns = ['Mean']
ccmd_resampled_median = ccmd_resampled.median()
ccmd_resampled_median.columns = ['Median']
ccmd_resampled_mode = ccmd_resampled.apply(lambda mcmD: mcmD.mode().iloc[0])
#print(ccmdSum)
```

```
#print(ccmD_resampled.apply(lambda mcme: mcme.mode().iloc[0]))

#NEW HAMPSHIRE NUMBER OF CASES
ccnh =  confirmedCasesNH_integral.loc[:, '2022-05-30':'2023-01-01']
ccnhSum = ccnh.sum()
ccnhSum.index = date_range
ccnhSum.index = pd.to_datetime(ccnhSum.index)
ccnh_resampled = ccnhSum.resample('7D')
ccnh_resampled_mean = ccnh_resampled.mean()
ccnh_resampled_mean.columns = ['Mean']
ccnh_resampled_median = ccnh_resampled.median()
ccnh_resampled_median.columns = ['Median']
ccnh_resampled_mode = ccnh_resampled.apply(lambda mcnh: mcnh.mode().iloc[0])

#VERMONT NUMBER OF CASES
ccvt =  confirmedCasesVT_integral.loc[:, '2022-05-30':'2023-01-01']
ccvtSum = ccvt.sum()
ccvtSum.index = date_range
ccvtSum.index = pd.to_datetime(ccvtSum.index)
ccvt_resampled = ccvtSum.resample('7D')
ccvt_resampled_mean = ccvt_resampled.mean()
ccvt_resampled_mean.columns = ['Mean']
ccvt_resampled_median = ccvt_resampled.median()
ccvt_resampled_median.columns = ['Median']
ccvt_resampled_mode = ccvt_resampled.apply(lambda mcvt: mcvt.mode().iloc[0])

#PLOTTING THE MEAN, MEDIAN AND MODE FOR THE STATES FOR COMPARISION FOR THE␣
 ↪NUMBER OF CASES
# Create a new figure and axis object
fig, ax = plt.subplots(3, 1, figsize=(10, 10))
# Plot the mean data as a line chart
ccnj_resampled_mean.plot(kind='line', ax=ax[0], label='New Jersey')
ccmd_resampled_mean.plot(kind='line', ax=ax[0], label='Maryland')
ccnh_resampled_mean.plot(kind='line', ax=ax[0], label='New Hampshire')
ccvt_resampled_mean.plot(kind='line', ax=ax[0], label='Vermont')

# Plot the median data as a line chart
ccnj_resampled_median.plot(kind='line', ax=ax[1], label='New Jersey')
ccmd_resampled_median.plot(kind='line', ax=ax[1], label='Maryland')
ccnh_resampled_median.plot(kind='line', ax=ax[1], label='New Hampshire')
ccvt_resampled_median.plot(kind='line', ax=ax[1], label='Vermont')


# Plot the mode data as a line chart
ccnj_resampled_mode.plot(kind='line', ax=ax[2], label='New Jersey')
ccmd_resampled_mode.plot(kind='line', ax=ax[2], label='Maryland')
ccnh_resampled_mode.plot(kind='line', ax=ax[2], label='New Hampshire')
```

```
ccvt_resampled_mode.plot(kind='line', ax=ax[2], label='Vermont')


# Set the axis labels and title
ax[0].set_xlabel('Date')
ax[0].set_ylabel('Value')
ax[0].set_title('Mean Values for the number of CASES by Date for every 7 Days')

ax[1].set_xlabel('Date')
ax[1].set_ylabel('Value')
ax[1].set_title('Median Values for the number of CASES by Date for every 7␣
 ↪Days')

ax[2].set_xlabel('Date')
ax[2].set_ylabel('Value')
ax[2].set_title('Mode Values for the number of CASES by Date for every 7 Days')

# Add a legend to the plot
ax[0].legend()
ax[1].legend()
ax[2].legend()

#adding padding between the graphs
fig.subplots_adjust(hspace=0.5)

# Show the plot
plt.show()
```



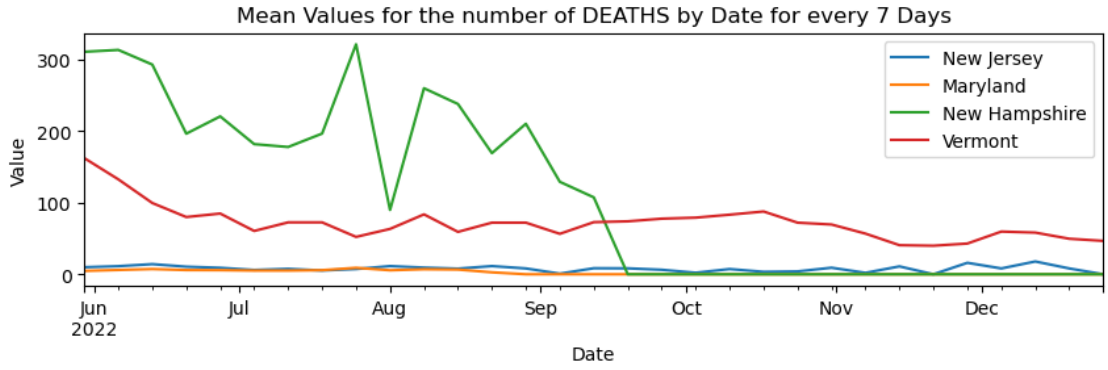


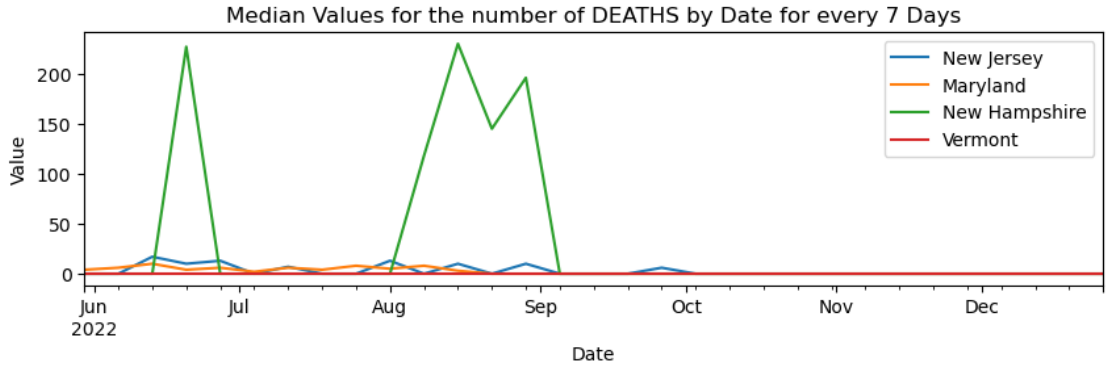


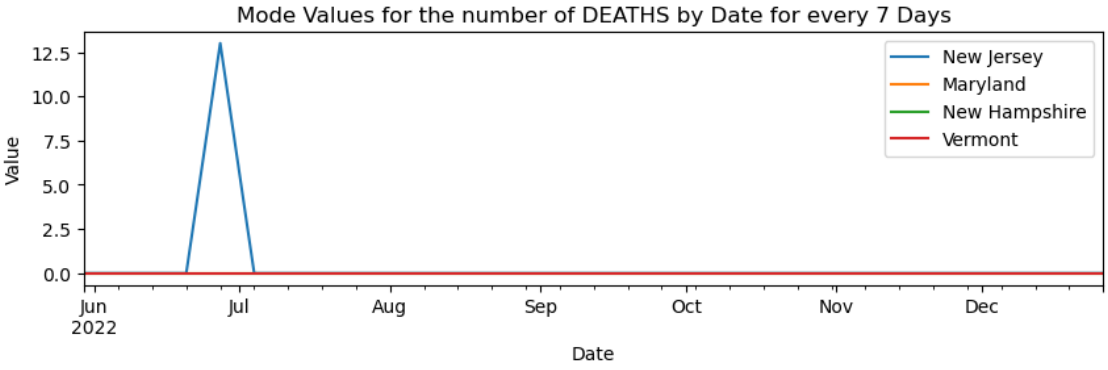
[10]:
```
'''COMPARING THE MEAN, MEDIAN AND MODE FOR THE STATES OF NEW JERSEY, MAINE, NEW␣
 ↪HAMPSHIRE, VERMONT (NUMBER OF DEATHS)'''
date_range = pd.date_range(start='2022-05-30', end='2023-01-01')

'''#NEW JERSEY NUMBER OF CASES
cdnj =  confirmedDEATHSNJ_integral.loc[:, '2022-05-30':'2023-01-01']
cdnjSum = ccnj.sum()
cdnjSum.index = date_range
cdnjSum.index = pd.to_datetime(cdnjSum.index)
cdnj_resampled = cdnjSum.resample('7D')
cdnj_resampled_mean = cdnj_resampled.mean()
cdnj_resampled_mean.columns = ['Mean']
```

```
cdnj_resampled_median = cdnj_resampled.median()
cdnj_resampled_median.columns = ['Median']
cdnj_resampled_mode = cdnj_resampled.apply(lambda mdnj: mdnj.mode().iloc[0])'''

#MARYLAND NUMBER OF CASES
cdmd =  confirmedDeathsMD_integral.loc[:, '2022-05-30':'2023-01-01']
cdmdSum = cdmd.sum()
cdmdSum.index = date_range
cdmdSum.index = pd.to_datetime(cdmdSum.index)
cdmd_resampled = cdmdSum.resample('7D')
cdmd_resampled_mean = cdmd_resampled.mean()
cdmd_resampled_mean.columns = ['Mean']
cdmd_resampled_median = cdmd_resampled.median()
cdmd_resampled_median.columns = ['Median']
cdmd_resampled_mode = cdmd_resampled.apply(lambda mdmd: mdmd.mode().iloc[0])
#print(ccmdSum)
#print(ccmD_resampled.apply(lambda mcme: mcme.mode().iloc[0]))

#NEW HAMPSHIRE NUMBER OF CASES
cdnh =  confirmedDeathsNH_integral.loc[:, '2022-05-30':'2023-01-01']
cdnhSum = cdnh.sum()
cdnhSum.index = date_range
cdnhSum.index = pd.to_datetime(cdnhSum.index)
cdnh_resampled = ccnhSum.resample('7D')
cdnh_resampled_mean = cdnh_resampled.mean()
cdnh_resampled_mean.columns = ['Mean']
cdnh_resampled_median = cdnh_resampled.median()
cdnh_resampled_median.columns = ['Median']
cdnh_resampled_mode = cdnh_resampled.apply(lambda mdnh: mdnh.mode().iloc[0])

#VERMONT NUMBER OF CASES
cdvt =  confirmedDeathsVT_integral.loc[:, '2022-05-30':'2023-01-01']
cdvtSum = ccvt.sum()
cdvtSum.index = date_range
cdvtSum.index = pd.to_datetime(cdvtSum.index)
cdvt_resampled = cdvtSum.resample('7D')
cdvt_resampled_mean = cdvt_resampled.mean()
cdvt_resampled_mean.columns = ['Mean']
cdvt_resampled_median = cdvt_resampled.median()
cdvt_resampled_median.columns = ['Median']
cdvt_resampled_mode = cdvt_resampled.apply(lambda mdvt: mdvt.mode().iloc[0])


#PLOTTING THE MEAN, MEDIAN AND MODE FOR THE STATES FOR COMPARISION FOR THE␣
 ↪NUMBER OF DEATHS
# Create a new figure and axis object
fig, ax = plt.subplots(3, 1, figsize=(10, 10))
```

```
# Plot the mean data as a line chart
cdnj_resampled_mean.plot(kind='line', ax=ax[0], label='New Jersey')
cdmd_resampled_mean.plot(kind='line', ax=ax[0], label='Maryland')
cdnh_resampled_mean.plot(kind='line', ax=ax[0], label='New Hampshire')
cdvt_resampled_mean.plot(kind='line', ax=ax[0], label='Vermont')

# Plot the median data as a line chart
cdnj_resampled_median.plot(kind='line', ax=ax[1], label='New Jersey')
cdmd_resampled_median.plot(kind='line', ax=ax[1], label='Maryland')
cdnh_resampled_median.plot(kind='line', ax=ax[1], label='New Hampshire')
cdvt_resampled_median.plot(kind='line', ax=ax[1], label='Vermont')


# Plot the mode data as a line chart
cdnj_resampled_mode.plot(kind='line', ax=ax[2], label='New Jersey')
cdmd_resampled_mode.plot(kind='line', ax=ax[2], label='Maryland')
cdnh_resampled_mode.plot(kind='line', ax=ax[2], label='New Hampshire')
cdvt_resampled_mode.plot(kind='line', ax=ax[2], label='Vermont')


# Set the axis labels and title
ax[0].set_xlabel('Date')
ax[0].set_ylabel('Value')
ax[0].set_title('Mean Values for the number of DEATHS by Date for every 7 Days')

ax[1].set_xlabel('Date')
ax[1].set_ylabel('Value')
ax[1].set_title('Median Values for the number of DEATHS by Date for every 7␣
 ↪Days')

ax[2].set_xlabel('Date')
ax[2].set_ylabel('Value')
ax[2].set_title('Mode Values for the number of DEATHS by Date for every 7 Days')

# Add a legend to the plot
ax[0].legend()
ax[1].legend()
ax[2].legend()

#adding padding between the graphs
fig.subplots_adjust(hspace=0.5)

# Show the plot
plt.show()
```





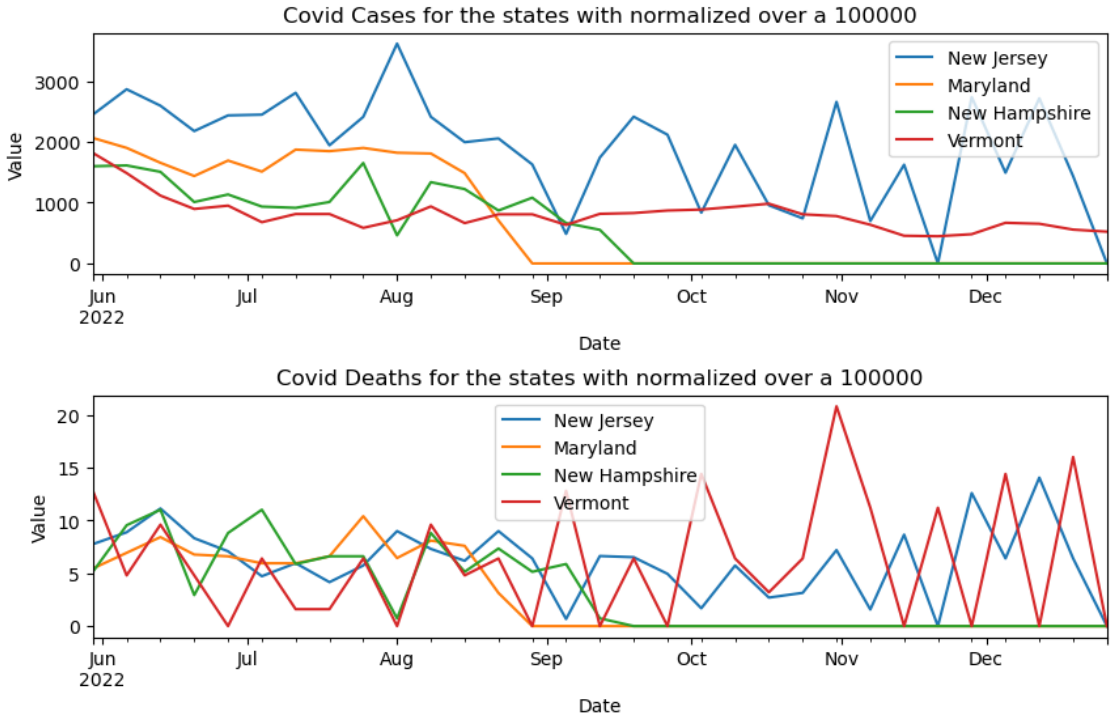


[11]:
```
'''FOR NORMALIZING THE DATA'''

#reading the data for the population of number of people and displaying them
population = pd.read_csv(r"..\..\..\DATASETS\COVID␣
 ↪DATASETS\covid_county_population_usafacts.csv")


#Extracting data from the given set of population for that county
populationNJ = population[population["State"] == "NJ"].drop(index = 1805) #NEW␣
 ↪JERSEY
populationMD = population[population["State"] == "MD"].drop(index = 1214)␣
 ↪#MARYLAND
```

```
populationNH = population[population["State"] == "NH"].drop(index = 1794) #NEW␣
 ↪HAMSHIRE
populationVT = population[population["State"] == "VT"].drop(index = 2853)␣
 ↪#VERMONT


#finding the normalized factor for each state per 10000
normFactorNJ = 1000000 / populationNJ.sum()['population']
normFactorMD = 1000000 / populationMD.sum()['population']
normFactorNH = 1000000 / populationNH.sum()['population']
normFactorVT = 1000000 / populationVT.sum()['population']

#multiplying the normalizing factor with the for the number of cases
normValueNJ = confirmedCasesNJ_integral.loc[:, '2022-05-30':'2023-01-01']␣
 ↪*normFactorNJ
normValueMD = confirmedCasesMD_integral.loc[:, '2022-05-30':'2023-01-01']␣
 ↪*normFactorMD
normValueNH = confirmedCasesNH_integral.loc[:, '2022-05-30':'2023-01-01']␣
 ↪*normFactorNH
normValueVT = confirmedCasesVT_integral.loc[:, '2022-05-30':'2023-01-01']␣
 ↪*normFactorVT


#multiplying the normalizing factor with the for the number of deaths
normValueDeathsNJ = confirmedDeathsNJ_integral.loc[:, '2022-05-30':
 ↪'2023-01-01'] *normFactorNJ
normValueDeathsMD = confirmedDeathsMD_integral.loc[:, '2022-05-30':
 ↪'2023-01-01'] *normFactorMD
normValueDeathsNH = confirmedDeathsNH_integral.loc[:, '2022-05-30':
 ↪'2023-01-01'] *normFactorNH
normValueDeathsVT = confirmedDeathsVT_integral.loc[:, '2022-05-30':
 ↪'2023-01-01'] *normFactorVT

normValueSumNJ = normValueNJ.sum()
normValueSumMD = normValueMD.sum()
normValueSumNH = normValueNH.sum()
normValueSumVT = normValueVT.sum()

normValueDeathsSumNJ = normValueDeathsNJ.sum()
normValueDeathsSumMD = normValueDeathsMD.sum()
normValueDeathsSumNH = normValueDeathsNH.sum()
normValueDeathsSumVT = normValueDeathsVT.sum()

#INDEXING AND CREATING A SAMPLE FOR 7DAYS

'''FOR THE NUMBER OF CONFIRMED CASES'''
```

```
date_range = pd.date_range(start='2022-05-30', end='2023-01-01')
normValueSumNJ.index = date_range
normValueSumMD.index = date_range
normValueSumNH.index = date_range
normValueSumVT.index = date_range

#cdnjSum.index = cdnjSum.index.strftime('%Y-%m-%d')
normValueSumNJ.index = pd.to_datetime(normValueSumNJ.index)
normValueSumMD.index = pd.to_datetime(normValueSumMD.index)
normValueSumNH.index = pd.to_datetime(normValueSumNH.index)
normValueSumVT.index = pd.to_datetime(normValueSumVT.index)

normValueSumNJ_resampled = normValueSumNJ.resample('7D')
normValueSumMD_resampled = normValueSumMD.resample('7D')
normValueSumNH_resampled = normValueSumNH.resample('7D')
normValueSumVT_resampled = normValueSumVT.resample('7D')

'''FOR THE NUMBER OF CONFIRMED DEATHS'''
date_range = pd.date_range(start='2022-05-30', end='2023-01-01')
normValueDeathsSumNJ.index = date_range
normValueDeathsSumMD.index = date_range
normValueDeathsSumNH.index = date_range
normValueDeathsSumVT.index = date_range

#cdnjSum.index = cdnjSum.index.strftime('%Y-%m-%d')
normValueDeathsSumNJ.index = pd.to_datetime(normValueDeathsSumNJ.index)
normValueDeathsSumMD.index = pd.to_datetime(normValueDeathsSumMD.index)
normValueDeathsSumNH.index = pd.to_datetime(normValueDeathsSumNH.index)
normValueDeathsSumVT.index = pd.to_datetime(normValueDeathsSumVT.index)

normValueDeathsSumNJ_resampled = normValueDeathsSumNJ.resample('7D')
normValueDeathsSumMD_resampled = normValueDeathsSumMD.resample('7D')
normValueDeathsSumNH_resampled = normValueDeathsSumNH.resample('7D')
normValueDeathsSumVT_resampled = normValueDeathsSumVT.resample('7D')

#Plotting graphs for the normalized values for the states that were compare␣
 ↪above

'''FOR THE NUMBER OF CONFIRMED CASES'''
# Create a new figure and axis object
fig, ax = plt.subplots(2, 1 , figsize=(10, 6))

# Plot the mean data as a line chart
normValueSumNJ_resampled.sum().plot(kind='line', ax=ax[0] ,label='New Jersey')
normValueSumMD_resampled.sum().plot(kind='line', ax=ax[0] ,label='Maryland' )
normValueSumNH_resampled.sum().plot(kind='line', ax=ax[0] ,label='New␣
 ↪Hampshire')
```

```
normValueSumVT_resampled.sum().plot(kind='line', ax=ax[0] ,label='Vermont')
ax[0].set_xlabel('Date')
ax[0].set_ylabel('Value')
ax[0].set_title('Covid Cases for the states with normalized over a 100000')

# Add a legend to the plot
ax[0].legend()


'''FOR THE NUMBER OF CONFIRMED DEATHS'''
# Plot the mean data as a line chart
normValueDeathsSumNJ_resampled.sum().plot(kind='line', ax=ax[1] ,label='New␣
 ↪Jersey')
normValueDeathsSumMD_resampled.sum().plot(kind='line', ax=ax[1]␣
 ↪,label='Maryland' )
normValueDeathsSumNH_resampled.sum().plot(kind='line', ax=ax[1] ,label='New␣
 ↪Hampshire')
normValueDeathsSumVT_resampled.sum().plot(kind='line', ax=ax[1]␣
 ↪,label='Vermont')
ax[1].set_xlabel('Date')
ax[1].set_ylabel('Value')
ax[1].set_title('Covid Deaths for the states with normalized over a 100000')

# Add a legend to the plot
ax[1].legend()

#adding padding between the graphs
fig.subplots_adjust(hspace=0.5)

# Show the plot
plt.show()
```



- Describe why the rates differ across these states in the notebook.

There may be any number of reasons for the difference in rates accross the states: a. Because the population varies in these states, the number of cases and deaths definitely vary among them. b. There may be different demographics of people who are more susceptible to catch covid, because of this the cases and may vary. c. There may be a different rates in states where the density of these population crowd. d. Because of migration of the population masses.

[22]:
```
#FOR FINDING THE PEAK WEEKS FOR THE NUMBER OF CASES AND DEATHS

'''NUMBER OF CASES'''
print('\033[1m' +"The peak week for the cases in the US for the chosen states:")
dict = {'State' : ['New Jersey', 'Maryland', 'New Hampshire', 'Vermont'],
        'Date' : [ccnj_resampled.sum().sort_values(ascending = False).head(1).
 ↪index[0].strftime("%Y-%m-%d"),
                  ccmd_resampled.sum().sort_values(ascending = False).head(1).
 ↪index[0].strftime("%Y-%m-%d"),
                  ccnh_resampled.sum().sort_values(ascending = False).head(1).
 ↪index[0].strftime("%Y-%m-%d"),
                  ccvt_resampled.sum().sort_values(ascending = False).head(1).
 ↪index[0].strftime("%Y-%m-%d")],
        'cases' : [ccnj_resampled.sum().sort_values(ascending = False).
 ↪head(1)[0],
```

```
                    int(ccmd_resampled.sum().sort_values(ascending = False).
 ↪head(1)[0]),
                    int(ccnh_resampled.sum().sort_values(ascending = False).
 ↪head(1)[0]),
                    int(ccvt_resampled.sum().sort_values(ascending = False).
 ↪head(1)[0])]}


peakWeekCases = pd.DataFrame(dict)
display(peakWeekCases)

print("")

'''NUMBER OF DEATHS'''

print('\033[1m' +"The peak week for the deaths in the US for the chosen states:
 ↪")
dict = {'State' : ['New Jersey', 'Maryland', 'New Hampshire', 'Vermont'],
        'Date' : [cdnj_resampled.sum().sort_values(ascending = False).head(1).
 ↪index[0].strftime("%Y-%m-%d"),
                  cdmd_resampled.sum().sort_values(ascending = False).head(1).
 ↪index[0].strftime("%Y-%m-%d"),
                  cdnh_resampled.sum().sort_values(ascending = False).head(1).
 ↪index[0].strftime("%Y-%m-%d"),
                  cdvt_resampled.sum().sort_values(ascending = False).head(1).
 ↪index[0].strftime("%Y-%m-%d")],
        'cases' : [cdnj_resampled.sum().sort_values(ascending = False).
 ↪head(1)[0],
                   int(cdmd_resampled.sum().sort_values(ascending = False).
 ↪head(1)[0]),
                   int(cdnh_resampled.sum().sort_values(ascending = False).
 ↪head(1)[0]),
                   int(cdvt_resampled.sum().sort_values(ascending = False).
 ↪head(1)[0])]}


peakWeekCases = pd.DataFrame(dict)
display(peakWeekCases)
```

The peak week for the cases in the US for the chosen states:

	State	Date	cases
0	New Jersey	2022-08-01	32110
1	Maryland	2022-05-30	12479
2	New Hampshire	2022-07-25	2246
3	Vermont	2022-05-30	1134

The peak week for the deaths in the US for the chosen states:

	State	Date	cases
0	New Jersey	2022-12-12	125
1	Maryland	2022-07-25	63
2	New Hampshire	2022-07-25	2246
3	Vermont	2022-05-30	1134

- Here for the 4 states I have shown which for each state which weak starting from that date has the highest number of cases and number of deaths for the total given data date range
- The peaks aren't really consistent with the Country because there may be varying factors for the states when compared to the whole of the country.

0.0.3 Part 3 - Identify 3 counties within a state of your choice with high cases and death rates.

References: https://stackoverflow.com/questions/8924173/how-can-i-print-bold-text-in-python - for printing the statement in bold

[12]:
```
#ccnjNewVal = confirmendCasesNJNew.iloc[:, :3]
ccnjDesList=[]
cdnjDesList = []
confirmendCasesNJNew_sum = confirmendCasesNJNew.iloc[:, 3:].sum(axis = 1)
#ccnjNewVal = pd.concat(confirmendCasesNJNew.iloc[:, :
 ↪3],confirmendCasesNJNew_sum )
#ccnjNewVal['Sum'] = confirmendCasesNJNew_sum
#print(confirmendCasesNJNew)
if 'Sum' not in confirmendCasesNJNew:
    confirmendCasesNJNew.insert(loc=3, column='Sum',␣
 ↪value=confirmendCasesNJNew_sum)

ccnjDes = confirmendCasesNJNew.sort_values('Sum',ascending = False).
 ↪head(3)#["County Name"]
#print(ccnjDes)
print('\033[1m' +"The top 3 counties within the State of New Jersey with high␣
 ↪number of cases:")
for i in ccnjDes["County Name"]:
    ccnjDesList.append(i)
    print(i)
print('')

#cdnjNewVal = confirmedDeathsNJNew.iloc[:, :3]
confirmendDeathsNJNew_sum = confirmedDeathsNJNew.iloc[:, 3:].sum(axis = 1)
#ccnjNewVal = pd.concat(confirmendCasesNJNew.iloc[:, :
 ↪3],confirmendCasesNJNew_sum )
#cdnjNewVal['Sum'] = confirmendDeathsNJNew_sum
if 'Sum' not in confirmedDeathsNJNew:
```

```
    confirmedDeathsNJNew.insert(loc=3, column='Sum',␣
 ↪value=confirmendDeathsNJNew_sum)

cdnjDes = confirmedDeathsNJNew.sort_values('Sum',ascending = False).
 ↪head(3)#["County Name"]
#print(ccnjDes)
print( '\033[1m' + "The top 3 counties within the State of New Jersey with high␣
 ↪number of deaths:" )
for j in cdnjDes["County Name"]:
    cdnjDesList.append(j)
    print(j)

#confirmendCasesNJNew
```

The top 3 counties within the State of New Jersey with high number of cases:

Bergen County

Middlesex County

Essex County

The top 3 counties within the State of New Jersey with high number of

deaths:

Ocean County

Bergen County

Monmouth County

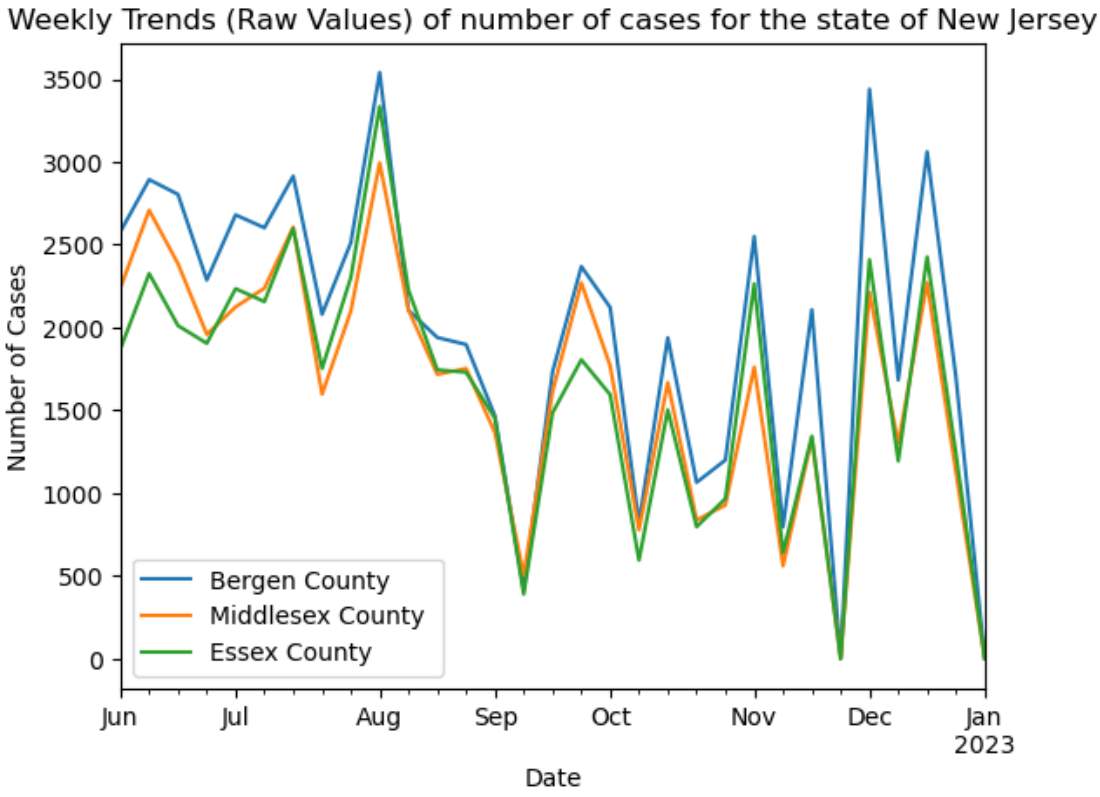
[ ]:

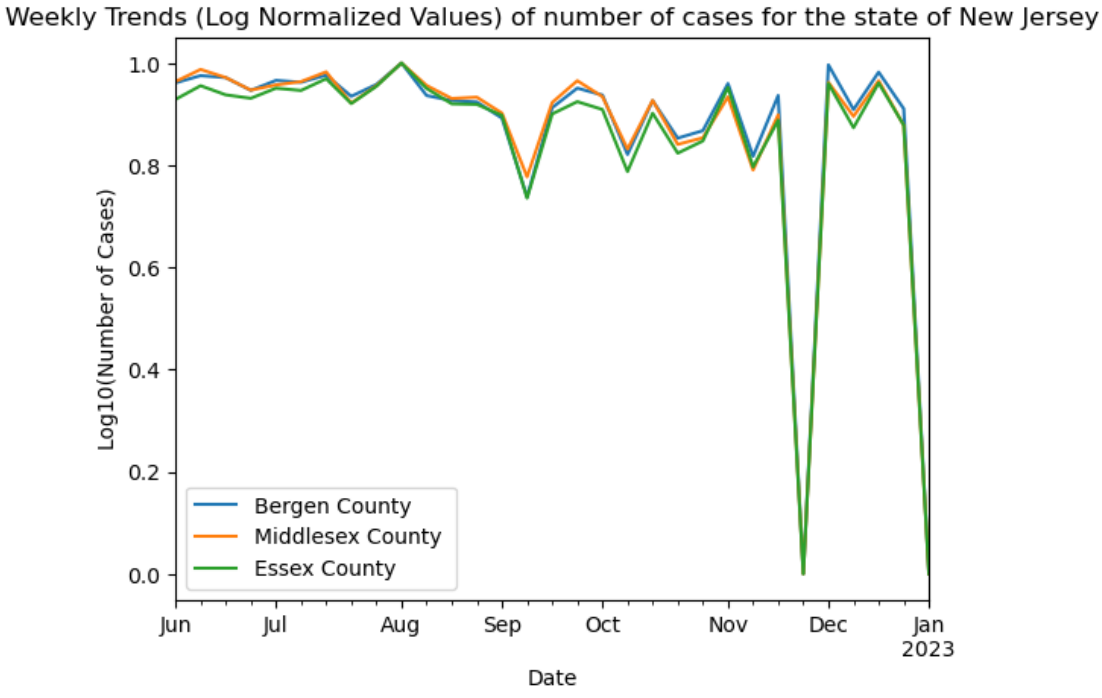
0.0.4 Part 4 - Plot weekly trends (new cases and deaths) for the top 3 infected counties. Show plots by raw values and log normalized values. Describe what is causing them and what were the peaks. Do the counties follow state pattern.

Reference: https://www.geeksforgeeks.org/data-normalization-with-pandas/ : used this to get a better understanding on what normalization of data means and how to use it, and what output should i expect

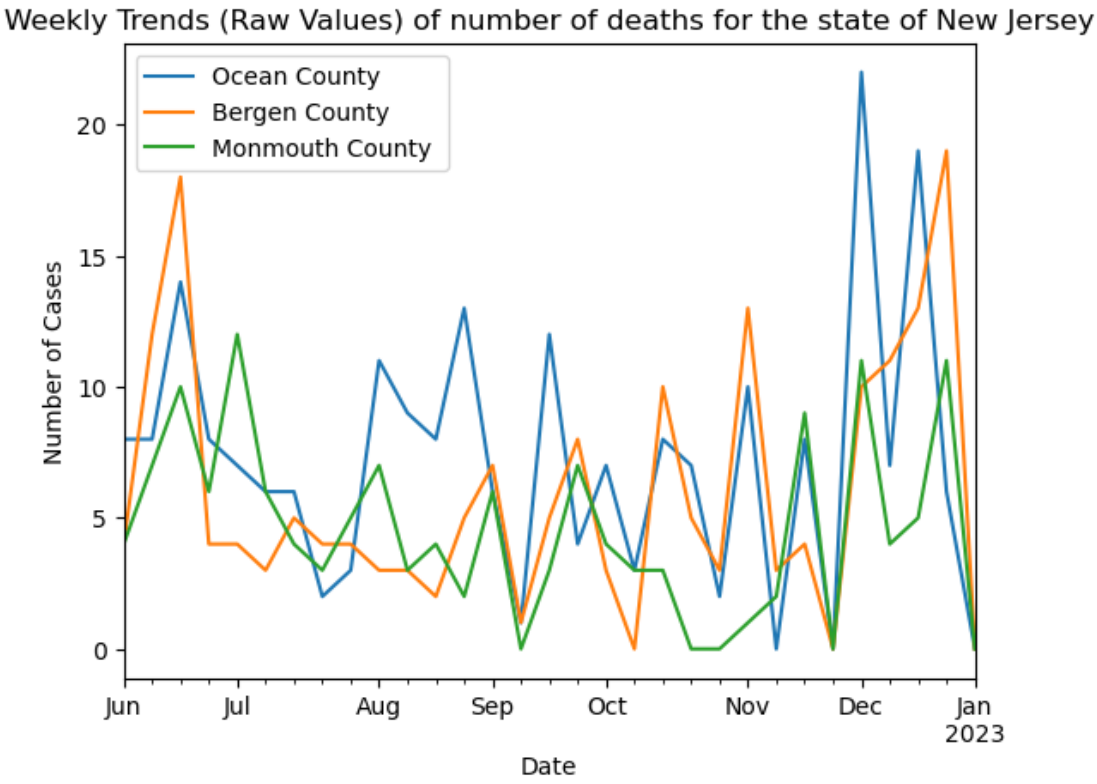
[13]:
```
ccnjDesT = ccnjDes.iloc[:, 4:].transpose()
ccnjDesT
ccnjDesT.index = pd.to_datetime(ccnjDesT.index)
ccnjDesT_resampled = ccnjDesT.resample('7D')
#print(ccnjDesT)
ccnjDesT.index = pd.to_datetime(ccnjDesT.index)

top_counties = ccnjDesT.columns
#print(top_counties)
```

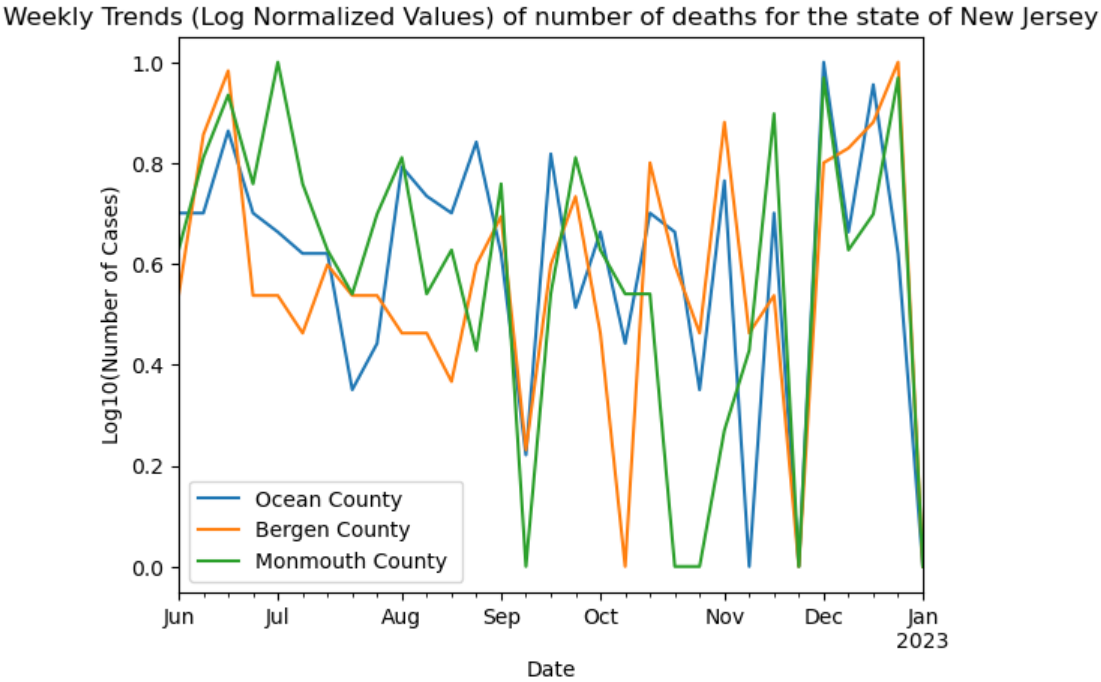
```
ccnjDesT_res = ccnjDesT.resample('W').sum()
#print('x',len(ccnjDesT_res))


ccnjDesT_res.plot( title="Weekly Trends (Raw Values) of number of cases for the␣
 ↪state of New Jersey", label ='raw' )
plt.xlabel("Date")
plt.ylabel("Number of Cases")
plt.legend(ccnjDesList)
plt.show()

df_max_scaled = ccnjDesT_res.copy()

for county in top_counties:
    df_max_scaled[county] = np.log10(df_max_scaled[county] + 1) / np.
 ↪log10(df_max_scaled[county].max() + 1)

#weeklyLogNorm = df_max_scaled.resample("W").mean()
df_max_scaled.plot(title="Weekly Trends (Log Normalized Values) of number of␣
 ↪cases for the state of New Jersey")
plt.xlabel("Date")
plt.ylabel("Log10(Number of Cases)")
plt.legend(ccnjDesList)
plt.show()
```


Weekly Trends (Raw Values) of number of cases for the state of New Jersey
Number of Cases
3500
3000
2500
2000
1500
1000
500
0
Bergen County
Middlesex County
Essex County
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
2023
Date


Weekly Trends (Log Normalized Values) of number of cases for the state of New Jersey
Log10(Number of Cases)
1.0
0.8
0.6
0.4
0.2
0.0
Bergen County
Middlesex County
Essex County
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
2023
Date

[14]:
```
cdnjDesT = cdnjDes.iloc[:, 4:].transpose()
cdnjDesT
cdnjDesT.index = pd.to_datetime(cdnjDesT.index)
cdnjDesT_resampled = cdnjDesT.resample('7D')
#print(ccnjDesT)
cdnjDesT.index = pd.to_datetime(cdnjDesT.index)

top_countiesd = cdnjDesT.columns
#print(top_counties)

cdnjDesT_res = cdnjDesT.resample('W').sum()
#print('x',len(ccnjDesT_res))


cdnjDesT_res.plot( title="Weekly Trends (Raw Values) of number of deaths for␣
 ↪the state of New Jersey", label ='raw' )
plt.xlabel("Date")
plt.ylabel("Number of Cases")
plt.legend(cdnjDesList)
plt.show()

df_max_scaled_deaths = cdnjDesT_res.copy()

for countyd in top_countiesd:
    df_max_scaled_deaths[countyd] = np.log(df_max_scaled_deaths[countyd] + 1) /␣
 ↪np.log(df_max_scaled_deaths[countyd].max() + 1)

#weeklyLogNormDeaths = df_max_scaled_deaths.resample("W").mean()
df_max_scaled_deaths.plot(title="Weekly Trends (Log Normalized Values) of␣
 ↪number of deaths for the state of New Jersey")
plt.xlabel("Date")
plt.ylabel("Log10(Number of Cases)")
plt.legend(cdnjDesList)
plt.show()
```


Weekly Trends (Raw Values) of number of deaths for the state of New Jersey
Ocean County
Bergen County
Monmouth County
Number of Cases
0
5
10
15
20
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
2023
Date


Weekly Trends (Log Normalized Values) of number of deaths for the state of New Jersey
Log10(Number of Cases)
0.0
0.2
0.4
0.6
0.8
1.0
Ocean County
Bergen County
Monmouth County
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
2023
Date

Describe what is causing them and what were the peaks. Do the counties follow state pattern.

On the first glance we can see that the number of cases rised rapidly and suddenly during the last week of July and the first week of August, after some research I found that there were state fairs conducted during this time across th state of New Jersey. Since it was a state fair many people attended this event, and if huge groups of people gather at a small place we know that COVID-19 would spread. This is also the case when the cases were fluctuating during the monthds of Novemeber and December as it was the holiday season and people would meet and gather, which would also cause the rise in COVID

Yes, the trends of the counties do follow the state pattern for the given time frame.

[ ]: