Covid-19_Combined

June 9, 2021

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import numpy as np
import plotly
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

Below is population data by sex and single year of age for July 1, 2019. It is calld 'Single Year of Age and Sex Population Estimates: April 1, 2010 to July 1, 2019 - CIVILIAN (SC-EST2019-AGESEX-CIV)'. This will be used to merge with the COVID death data to calculate death rates.

[2]:
```
pop = pd.read_csv('https://www2.census.gov/programs-surveys/popest/tables/
↪2010-2019/state/asrh/sc-est2019-agesex-civ.csv')
pop = pop[pop.SEX == 0]
pop = pop[['NAME','AGE','POPEST2019_CIV']]
popkeep = pop
pop.columns = ['NAME','AGE','Population']
pop['AgeGroup'] = pd.cut(pop['AGE'], bins = [-1, 24, 34, 44, 54, 64, 74, 84,␣
↪85, 9999],
       labels =␣
↪['0-24','25-34','35-44','45-54','55-64','65-74','75-84','85+','All Ages'])
population = pop.groupby(['NAME','AgeGroup'])['Population'].sum().reset_index()
```

[3]:
```
Biggest = population[population.AgeGroup == 'All Ages'].
↪sort_values('Population', ascending = False)['NAME']
def get_biggest_states(number = 10):
    number = number + 1
    return Biggest[:number]
```

[4]:
```
center = pd.read_csv("https://www2.census.gov/geo/docs/reference/cenpop2010/
↪CenPop2010_Mean_ST.txt")
also = pd.read_csv("https://www2.census.gov/geo/docs/reference/cenpop2010/
↪CenPop2010_Mean_US.txt")
also['STNAME'] = 'United States'
center = center.append(also)
center = center[['STNAME','LATITUDE','LONGITUDE']]
```

```
center
```

[4]:

	STNAME	LATITUDE	LONGITUDE
0	Alabama	33.008097	-86.756826
1	Alaska	61.399882	-148.873973
2	Arizona	33.368266	-111.864310
3	Arkansas	35.142580	-92.655243
4	California	35.463595	-119.325359
5	Colorado	39.513420	-105.208056
6	Connecticut	41.497001	-72.870342
7	Delaware	39.358946	-75.556835
8	District of Columbia	38.910270	-77.014468
9	Florida	27.822726	-81.634654
10	Georgia	33.376825	-83.882712
11	Hawaii	21.115289	-157.484404
12	Idaho	44.218532	-115.178681
13	Illinois	41.286759	-88.390334
14	Indiana	40.149246	-86.259514
15	Iowa	41.946066	-93.036629
16	Kansas	38.464949	-96.462812
17	Kentucky	37.824499	-85.248467
18	Louisiana	30.722814	-91.508833
19	Maine	44.299950	-69.736482
20	Maryland	39.140769	-76.797763
21	Massachusetts	42.272291	-71.363370
22	Michigan	42.873187	-84.203434
23	Minnesota	45.203555	-93.571903
24	Mississippi	32.590954	-89.579514
25	Missouri	38.423798	-92.198469
26	Montana	46.787064	-111.296213
27	Nebraska	41.174300	-97.315578
28	Nevada	37.043481	-116.191720
29	New Hampshire	43.154858	-71.461974
30	New Jersey	40.431810	-74.432208
31	New Mexico	34.632744	-106.354349
32	New York	41.501299	-74.620909
33	North Carolina	35.543075	-79.658232
34	North Dakota	47.348468	-99.309504
35	Ohio	40.455191	-82.773339
36	Oklahoma	35.598464	-96.836786
37	Oregon	44.743243	-122.585214
38	Pennsylvania	40.456756	-77.009680
39	Rhode Island	41.753609	-71.450869
40	South Carolina	34.025176	-81.011022
41	South Dakota	44.014397	-99.002355
42	Tennessee	35.808090	-86.359136
43	Texas	30.905244	-97.365594

```
44                  Utah  40.401359 -111.927035
45               Vermont  44.094874  -72.816417
46              Virginia  37.810313  -77.811160
47            Washington  47.330750 -121.619994
48         West Virginia  38.795594  -80.731308
49             Wisconsin  43.721933  -89.018997
50               Wyoming  42.697026 -107.019126
51           Puerto Rico  18.280422  -66.346513
0          United States  37.517534  -92.173096
```

1 1 Data

This is the 'Conditions Contributing to COVID-19 Deaths, by State and Age' dataset from NCHS. This is the updated on June 2, 2021 edition. This dataset shows health conditions and contributing causes mentioned in conjunction with deaths involving coronavirus disease 2019 (COVID-19) by age group and jurisdiction of occurrence.

Footnote from website:

Number of conditions reported in this table are tabulated from deaths received and coded as of the date of analysis and do not represent all deaths that occurred in that period. Data during this period are incomplete because of the lag in time between when the death occurred and when the death certificate is completed, submitted to NCHS and processed for reporting purposes. This delay can range from 1 week to 8 weeks or more. Conditions contributing to the death were identified using the International Classification of Diseases, Tenth Revision (ICD-10). Deaths involving more than one condition (e.g., deaths involving both diabetes and respiratory arrest) were counted in both totals. To avoid counting the same death multiple times, the numbers for different conditions should not be summated. Deaths with confirmed or presumed COVID-19, coded to ICD–10 code U07.1. "COVID-19 Deaths" represents the number of deaths that mention one or more of the conditions indicated. The "Number of Mentions" column represents the number of total conditions mentioned for each age group.

[5]:
```
df = pd.read_csv('Conditions_Contributing_to_COVID-19_Deaths__by_State_and_Age.
↪csv')
```

[6]:
```
df.shape
```

[6]:
```
(248400, 14)
```

[7]:
```
df.head()
```

[7]:

	Data As Of	Start Date	End Date	Group	Year	Month	State	\
0	05/30/2021	01/01/2020	05/29/2021	By Total	NaN	NaN	United States	
1	05/30/2021	01/01/2020	05/29/2021	By Total	NaN	NaN	United States	
2	05/30/2021	01/01/2020	05/29/2021	By Total	NaN	NaN	United States	
3	05/30/2021	01/01/2020	05/29/2021	By Total	NaN	NaN	United States	
4	05/30/2021	01/01/2020	05/29/2021	By Total	NaN	NaN	United States	

```
        Condition Group                Condition ICD10_codes Age Group  \
0  Respiratory diseases  Influenza and pneumonia     J09-J18      0-24
1  Respiratory diseases  Influenza and pneumonia     J09-J18     25-34
2  Respiratory diseases  Influenza and pneumonia     J09-J18     35-44
3  Respiratory diseases  Influenza and pneumonia     J09-J18     45-54
4  Respiratory diseases  Influenza and pneumonia     J09-J18     55-64

   COVID-19 Deaths  Number of Mentions Flag
0            409.0               426.0  NaN
1           1865.0              1911.0  NaN
2           4961.0              5090.0  NaN
3          14705.0             15124.0  NaN
4          37388.0             38398.0  NaN
```

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 248400 entries, 0 to 248399
Data columns (total 14 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   Data As Of          248400 non-null  object
 1   Start Date          248400 non-null  object
 2   End Date            248400 non-null  object
 3   Group               248400 non-null  object
 4   Year                235980 non-null  float64
 5   Month               211140 non-null  float64
 6   State               248400 non-null  object
 7   Condition Group     248400 non-null  object
 8   Condition           248400 non-null  object
 9   ICD10_codes         248400 non-null  object
 10  Age Group           248400 non-null  object
 11  COVID-19 Deaths     183722 non-null  float64
 12  Number of Mentions  185590 non-null  float64
 13  Flag                64678 non-null   object
dtypes: float64(4), object(10)
memory usage: 26.5+ MB
```

[9]:
```
df.describe()
```

[9]:
```
                Year          Month  COVID-19 Deaths  Number of Mentions
count  235980.000000  211140.000000    183722.000000       185590.000000
mean     2020.315789       5.470588       142.274828          151.303179
std         0.464831       3.397544      2556.148244         2705.100248
min      2020.000000       1.000000         0.000000            0.000000
25%      2020.000000       3.000000         0.000000            0.000000
50%      2020.000000       5.000000         0.000000            0.000000
75%      2021.000000       8.000000        22.000000           24.000000
```

```
max       2021.000000       12.000000     581832.000000       581832.000000
```

The `Condition Group` variable completely overlaps with the `Condition` variable. It is possible to create the `Condition Group` variable from the `Condition` variable alone as seen by function condition_to_group below. The `Condition` variable simply creates more categories for respiratory diseases and circulatory diseases.

The `ICD10_codes` variable has a one-to-one relationship with `Condition`, so the two variables are actually interchangable.

[10]:
```
def condition_to_group(condition):
    condition = condition.replace(['Influenza and pneumonia','Chronic lower␣
↪respiratory diseases','Adult respiratory distress syndrome',
                        'Respiratory failure','Respiratory arrest','Other␣
↪diseases of the respiratory system'],'Respiratory diseases')
    condition = condition.replace(['Hypertensive diseases','Ischemic heart␣
↪disease','Cardiac arrest','Cardiac arrhythmia',
                                   'Heart failure','Cerebrovascular␣
↪diseases','Other diseases of the circulatory system'],'Circulatory diseases')
    return condition
def condition_to_ICD10(condition):
    codes = ['J09-J18','J40-J47','J80','J96','R09.2','J00-J06, J20-J39,␣
↪J60-J70, J81-J86, J90-J95, J97-J99, U04',
            'I10-I15','I20-I25','I46','I44, I45, I47-I49','I50','I60-I69',
            'I00-I09, I26-I43, I51, I52,␣
↪I70-I99','A40-A41','C00-C97','E10-E14','E65-E68','G30',
             'F01, F03','N17-N19',
             'S00-T98, V01-X59, X60-X84, X85-Y09, Y10-Y36, Y40-Y89, U01-U03',
             'A00-A39, A42-B99, D00-E07, E15-E64, E70-E90, F00, F02, F04-G26,␣
↪G31-H95, K00-K93, L00-M99, N00-N16, N20-N98, O00-O99, P00-P96, Q00-Q99,␣
↪R00-R08, R09.0, R09.1, R09.3, R09.8, R10-R99',
            'U071']
    conditions = ['Influenza and pneumonia','Chronic lower respiratory␣
↪diseases',
                  'Adult respiratory distress syndrome','Respiratory␣
↪failure','Respiratory arrest',
                'Other diseases of the respiratory system','Hypertensive␣
↪diseases','Ischemic heart disease',
                'Cardiac arrest','Cardiac arrhythmia','Heart␣
↪failure','Cerebrovascular diseases',
                'Other diseases of the circulatory system','Sepsis','Malignant␣
↪neoplasms','Diabetes',
                  'Obesity','Alzheimer disease','Vascular and unspecified␣
↪dementia',
                  'Renal failure',
                  'Intentional and unintentional injury, poisoning, and other␣
↪adverse events',
```

```
                  'All other conditions and causes (residual)','COVID-19']
    condition = condition.replace(conditions, codes)
    return condition
```

Below is creating a new DataFrame, where all the deaths and mentions from January 1, 2020 through April 30, 2021 is combined. This is done because the death data total that includes May 2021 is still provisional and the death data for that month is mostly incomplete. Therefore, May 2021 needs to be excluded from the DataFrame. Afterwards, the DataFrame is merged with the population data. Then the death rate can be calculated.

Also, immediately remove New York City, which is a subset of New York. Remove Puerto Rico as it is not a state and not of interest here.

[11]:
```
df = df[(df.State != 'Puerto Rico') & (df.State != 'New York City')]
filt = df[(df['Start Date'] != '05/01/2021') & (df['Group'] == 'By Month')]
by_may1 = filt.groupby(['State','Age Group','Condition'])[['COVID-19␣
 ↪Deaths','Number of Mentions']].sum().reset_index()
by_may1['Condition Group'] = condition_to_group(by_may1['Condition'])
by_may1 = by_may1.merge(population, how='left', left_on=['State','Age Group'],␣
 ↪right_on=['NAME','AgeGroup'])
by_may1 = by_may1.drop(['NAME','AgeGroup'], axis = 1)
by_may1['Death Rate'] = by_may1['COVID-19 Deaths']/by_may1['Population']*100
by_may1['Mention Rate'] = by_may1['Number of Mentions']/
 ↪by_may1['Population']*100
```

Drop the `Flag` for being not useful for analysis. All it does is a footnote whether data is suppressed if there isn't enough data. `Data As Of` is not useful it is all just equal to 05/30/2021. `Start Date`, `End Date`, and `ICD10_codes` are extraneous information that can be found in other columns.

[12]:
```
df = df.merge(population, how = 'left', left_on=['State','Age Group'], right_on␣
 ↪= ['NAME','AgeGroup'])
df = df.drop(['NAME','AgeGroup'], axis = 1)
df['Death Rate'] = df['COVID-19 Deaths']/df['Population']*100
df['Mention Rate'] = df['Number of Mentions']/df['Population']*100
df = df[(df['Start Date'] != '05/01/2021')]
df = df.drop(['Flag','Data As Of','Start Date','End Date','ICD10_codes'], axis␣
 ↪= 1)
df = df[(df['Age Group'] != 'Not stated')]
```

Below is the data for provisional COVID-19 deaths by Sex and Age. Deaths involving coronavirus disease 2019 (COVID-19), pneumonia, and influenza are reported to NCHS by sex, age group, and jurisdiction of occurrence.

[13]:
```
number2 = pd.read_csv('Provisional_COVID-19_Deaths_by_Sex_and_Age.csv')
number2 = number2.drop(['Footnote'], axis = 1)
number2 = number2[(number2['Age Group'] != '0-17 years')]
keep = df['Age Group'].unique()
number2 = number2.replace(['Under 1 year','1-4 years','5-14 years','15-24␣
 ↪years'], '0-24')
```

```
number2 = number2.replace(['25-34 years','35-44 years','45-54 years','55-64␣
↪years','65-74 years','75-84 years','85 years and over'],
               ['25-34','35-44','45-54','55-64','65-74','75-84','85+'])
number2 = number2[(number2['Age Group'].isin(keep))]
number2 = number2.groupby(['Data As Of','Start Date','End␣
↪Date','Group','Year','Month','State','Sex','Age Group'], dropna = False).
↪sum().reset_index()
number2 = number2[(number2['Start Date'] != '05/01/2021')]
number2 = number2.drop(['Data As Of','Start Date','End Date'], axis = 1)
number2 = number2[(number2['Sex'] == 'All Sexes')].drop(['Sex'], axis = 1)
```

Below is Life Tables by State (2018) from the CDC. This is used to calculate the death rate, average age, and average age of death for all age groups.

[14]:
```
states = df['State'].unique()
states = np.delete(states, 0)
statematrix = pd.DataFrame({'State':[], 'Age':[], 'Death Rate':[]})
for state in states:
    s = state.replace(' ','-')
    url = 'https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/
↪70-01/'+s+'-1-Total.xlsx'
    print(url)
    A = pd.read_excel(url, skiprows = 2)
    A['Age'] = A.index
    A['Death Rate'] = A['qx']
    A['State'] = state
    statematrix = statematrix.append(A[['State','Age','Death Rate']])
```

```
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Alabama-1
-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Alaska-1-
Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Arizona-1
-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Arkansas-
1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Californi
a-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Colorado-
1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Connectic
ut-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Delaware-
1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/District-
of-Columbia-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Florida-1
-Total.xlsx
```

https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Georgia-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Hawaii-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Idaho-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Illinois-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Indiana-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Iowa-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Kansas-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Kentucky-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Louisiana-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Maine-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Maryland-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Massachusetts-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Michigan-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Minnesota-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Mississippi-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Missouri-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Montana-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Nebraska-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Nevada-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/New-Hampshire-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/New-Jersey-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/New-Mexico-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/New-York-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/North-Carolina-1-Total.xlsx

```
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/North-
Dakota-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Ohio-1-To
tal.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Oklahoma-
1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Oregon-1-
Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Pennsylva
nia-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Rhode-
Island-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/South-
Carolina-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/South-
Dakota-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Tennessee
-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Texas-1-T
otal.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Utah-1-To
tal.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Vermont-1
-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Virginia-
1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Washingto
n-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/West-
Virginia-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Wisconsin
-1-Total.xlsx
https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/70-01/Wyoming-1
-Total.xlsx
```

[15]:
```
url = 'https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/69-12/
 ↪Table01.xlsx'
A = pd.read_excel(url, skiprows = 2)
A['Age'] = A.index
A['Death Rate'] = A['qx']
A['State'] = 'United States'
statematrix = statematrix.append(A[['State','Age','Death Rate']])
```

[16]:
```
popkeep2 = popkeep.merge(statematrix, how = 'left', left_on = ['NAME','AGE'],␣
 ↪right_on = ['State','Age'])
popkeep2 = popkeep2.drop(['State','Age'], axis = 1)
popkeep2['Deaths'] = popkeep2['Death Rate'] * popkeep2['Population']
```

```
AverageAge = popkeep2.groupby(['NAME','AgeGroup']).apply(lambda x:␣
↪(x['Deaths']*x['AGE']).sum()/x['Deaths'].sum()).reset_index()
AverageAgeA = popkeep2.groupby(['NAME','AgeGroup']).apply(lambda x:␣
↪(x['Population']*x['AGE']).sum()/x['Population'].sum()).reset_index()
Drate = popkeep2.groupby(['NAME','AgeGroup']).apply(lambda x: (x['Death Rate']).
↪mean()).reset_index()
Drate.columns = ['State','Age Group','Tables Death Rate']
AverageAge.columns = ['State','Age Group','Average Age of Death']
AverageAgeA.columns = ['State','Age Group','Average Age']
```

```
<ipython-input-16-988918d745ca>:4: RuntimeWarning: invalid value encountered in
double_scalars
  AverageAge = popkeep2.groupby(['NAME','AgeGroup']).apply(lambda x:
(x['Deaths']*x['AGE']).sum()/x['Deaths'].sum()).reset_index()
```

[17]:
```
base_url_cdd = 'https://ftp.cpc.ncep.noaa.gov/htdocs/products/
↪analysis_monitoring/cdus/degree_days/archives/Cooling%20Degree%20Days/
↪monthly%20cooling%20degree%20days%20state/'
base_url_hdd =  'https://ftp.cpc.ncep.noaa.gov/htdocs/products/
↪analysis_monitoring/cdus/degree_days/archives/Heating%20degree%20Days/
↪monthly%20states/'
month_list = ['2020/Jan 2020','2020/Feb 2020','2020/Mar 2020','2020/Apr 2020',
              '2020/May 2020','2020/Jun 2020','2020/Jul 2020','2020/Aug 2020',
              '2020/Sep 2020','2020/Oct 2020','2020/Nov 2020','2020/Dec␣
↪2020','2021/Jan 2021',
              '2021/Feb 2021','2021/Mar 2021','2021/Apr 2021']

month_list = [s.replace(' ','%20') for s in month_list]
degreematrix = pd.DataFrame({'State':[],'Month?':[],'CDD':[],'HDD':[]})
for month in month_list:
    url = base_url_cdd + month + ".txt"
    url2 = base_url_hdd + month + ".txt"
    print(url, url2)
    A = pd.read_fwf(url, widths = [17,6,5,8,6,6,6,6,6])
    A = A[14:]
    A['Month?'] = month
    A['State'] = A['Unnamed: 0']
    A['CDD'] = A['Unnamed: 1']
    B = pd.read_fwf(url2, widths = [17,6,5,8,6,6,6,6,6])
    B = B[14:]
    A['HDD'] = B['Unnamed: 1']
    degreematrix = degreematrix.append(A[['State','Month?','CDD','HDD']])
```

```
https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitoring/cdus/degree_da
ys/archives/Cooling%20Degree%20Days/monthly%20cooling%20degree%20days%20state/20
20/Jan%202020.txt https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitor
ing/cdus/degree_days/archives/Heating%20degree%20Days/monthly%20states/2020/Jan%
202020.txt
```

https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitoring/cdus/degree_da
ys/archives/Cooling%20Degree%20Days/monthly%20cooling%20degree%20days%20state/20
20/Feb%202020.txt https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitor
ing/cdus/degree_days/archives/Heating%20degree%20Days/monthly%20states/2020/Feb%
202020.txt
https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitoring/cdus/degree_da
ys/archives/Cooling%20Degree%20Days/monthly%20cooling%20degree%20days%20state/20
20/Mar%202020.txt https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitor
ing/cdus/degree_days/archives/Heating%20degree%20Days/monthly%20states/2020/Mar%
202020.txt
https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitoring/cdus/degree_da
ys/archives/Cooling%20Degree%20Days/monthly%20cooling%20degree%20days%20state/20
20/Apr%202020.txt https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitor
ing/cdus/degree_days/archives/Heating%20degree%20Days/monthly%20states/2020/Apr%
202020.txt
https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitoring/cdus/degree_da
ys/archives/Cooling%20Degree%20Days/monthly%20cooling%20degree%20days%20state/20
20/May%202020.txt https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitor
ing/cdus/degree_days/archives/Heating%20degree%20Days/monthly%20states/2020/May%
202020.txt
https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitoring/cdus/degree_da
ys/archives/Cooling%20Degree%20Days/monthly%20cooling%20degree%20days%20state/20
20/Jun%202020.txt https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitor
ing/cdus/degree_days/archives/Heating%20degree%20Days/monthly%20states/2020/Jun%
202020.txt
https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitoring/cdus/degree_da
ys/archives/Cooling%20Degree%20Days/monthly%20cooling%20degree%20days%20state/20
20/Jul%202020.txt https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitor
ing/cdus/degree_days/archives/Heating%20degree%20Days/monthly%20states/2020/Jul%
202020.txt
https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitoring/cdus/degree_da
ys/archives/Cooling%20Degree%20Days/monthly%20cooling%20degree%20days%20state/20
20/Aug%202020.txt https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitor
ing/cdus/degree_days/archives/Heating%20degree%20Days/monthly%20states/2020/Aug%
202020.txt
https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitoring/cdus/degree_da
ys/archives/Cooling%20Degree%20Days/monthly%20cooling%20degree%20days%20state/20
20/Sep%202020.txt https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitor
ing/cdus/degree_days/archives/Heating%20degree%20Days/monthly%20states/2020/Sep%
202020.txt
https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitoring/cdus/degree_da
ys/archives/Cooling%20Degree%20Days/monthly%20cooling%20degree%20days%20state/20
20/Oct%202020.txt https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitor
ing/cdus/degree_days/archives/Heating%20degree%20Days/monthly%20states/2020/Oct%
202020.txt
https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitoring/cdus/degree_da
ys/archives/Cooling%20Degree%20Days/monthly%20cooling%20degree%20days%20state/20
20/Nov%202020.txt https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitor

```
ing/cdus/degree_days/archives/Heating%20degree%20Days/monthly%20states/2020/Nov%
202020.txt
https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitoring/cdus/degree_da
ys/archives/Cooling%20Degree%20Days/monthly%20cooling%20degree%20days%20state/20
20/Dec%202020.txt https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitor
ing/cdus/degree_days/archives/Heating%20degree%20Days/monthly%20states/2020/Dec%
202020.txt
https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitoring/cdus/degree_da
ys/archives/Cooling%20Degree%20Days/monthly%20cooling%20degree%20days%20state/20
21/Jan%202021.txt https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitor
ing/cdus/degree_days/archives/Heating%20degree%20Days/monthly%20states/2021/Jan%
202021.txt
https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitoring/cdus/degree_da
ys/archives/Cooling%20Degree%20Days/monthly%20cooling%20degree%20days%20state/20
21/Feb%202021.txt https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitor
ing/cdus/degree_days/archives/Heating%20degree%20Days/monthly%20states/2021/Feb%
202021.txt
https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitoring/cdus/degree_da
ys/archives/Cooling%20Degree%20Days/monthly%20cooling%20degree%20days%20state/20
21/Mar%202021.txt https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitor
ing/cdus/degree_days/archives/Heating%20degree%20Days/monthly%20states/2021/Mar%
202021.txt
https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitoring/cdus/degree_da
ys/archives/Cooling%20Degree%20Days/monthly%20cooling%20degree%20days%20state/20
21/Apr%202021.txt https://ftp.cpc.ncep.noaa.gov/htdocs/products/analysis_monitor
ing/cdus/degree_days/archives/Heating%20degree%20Days/monthly%20states/2021/Apr%
202021.txt
```

[18]:
```
Z = degreematrix['Month?'].str.split('/|%', expand = True).drop(2, axis = 1)
degreematrix['Year'] = Z[0]
degreematrix['Month'] = Z[1]
degreematrix = degreematrix.
 ↪replace(['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'],␣
 ↪[1,2,3,4,5,6,7,8,9,10,11,12])
degreematrix['State'] = degreematrix['State'].str.title()
degreematrix = degreematrix.replace(['Distrct Columbia'], ['District of␣
 ↪Columbia'])
degreematrix = degreematrix.drop('Month?', axis = 1)
degreematrix['Month'] = degreematrix['Month'] + 0.00
degreematrix['Year'] = degreematrix['Year'].astype(float)
```

1.0.1 EDA(incomplete)

[19]:
```
def get_percent_of_deathsage(frame, inde =␣
 ↪['State','Year','Month','Condition'], col = 'Age Group', val = 'COVID-19␣
 ↪Deaths'):
    piv = frame.pivot(index = inde, columns = col, values = val)
```

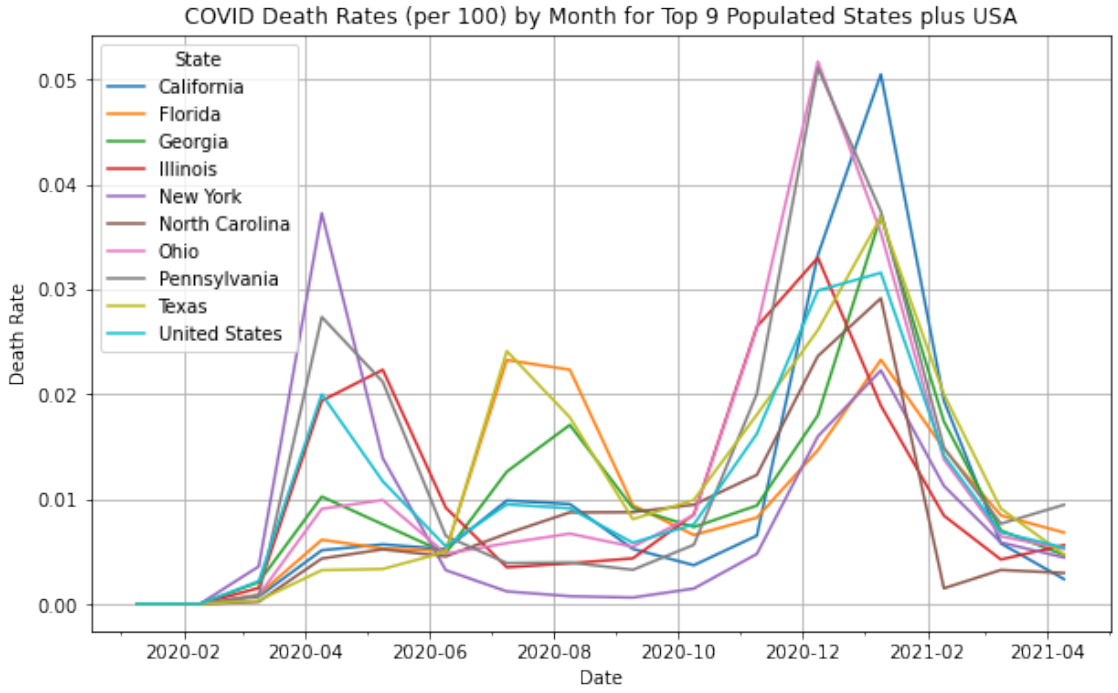
```
    pov = piv.copy()
    keep = piv.columns
    for col in keep:
        pov[col] = piv[col]/piv['All Ages']
    return pd.melt(pov.reset_index(), value_vars=keep, id_vars=inde)
# Gets percent of deaths with certain condition. Note that if 0 COVID-19␣
 ↪deaths, then a NaN results since divide by 0 error
def get_percent_of_deaths(frame, inde = ['State','Year','Month','Age Group'],␣
 ↪col = 'Condition', val = 'Death Rate'):
    piv = frame.pivot(index = inde, columns = col, values = val)
    pov = piv.copy()
    keep = piv.columns
    for col in keep:
        pov[col] = piv[col]/piv['COVID-19']
    return pd.melt(pov.reset_index(), value_vars=keep, id_vars=inde)
# Converts the Year/Month into a single Year/Month using ISO format
def yearmonth(frame):
    frame['Year/Month'] = frame['Year'].astype(int).astype(str) + "/" +␣
 ↪frame['Month'].astype(int).astype(str).str.zfill(2)
    return frame
# Cutyearmonth cuts by month: Note that 1 is 2020/1, 12 is 2020/12, and 16 is␣
 ↪2021/4
# This function allows grouping months.
def cutyearmonth(frame, cuts = [0,9,17], names = ['First','Second']):
    frame['Peak'] = (frame['Year'].astype(int)-2020)*12 + frame['Month'].
 ↪astype(int)
    frame['Peak'] = pd.cut(frame['Peak'], bins = cuts, labels = names)
    return frame
explore = df.query('Group == "By Month"')
explore = yearmonth(explore)
explore = cutyearmonth(explore)
explore = explore.merge(center, how='left', left_on=['State'],␣
 ↪right_on=['STNAME'])
explore = explore.drop(['STNAME'], axis = 1)
```

```
<ipython-input-19-a4a636352767>:18: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  frame['Year/Month'] = frame['Year'].astype(int).astype(str) + "/" +
frame['Month'].astype(int).astype(str).str.zfill(2)
<ipython-input-19-a4a636352767>:23: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

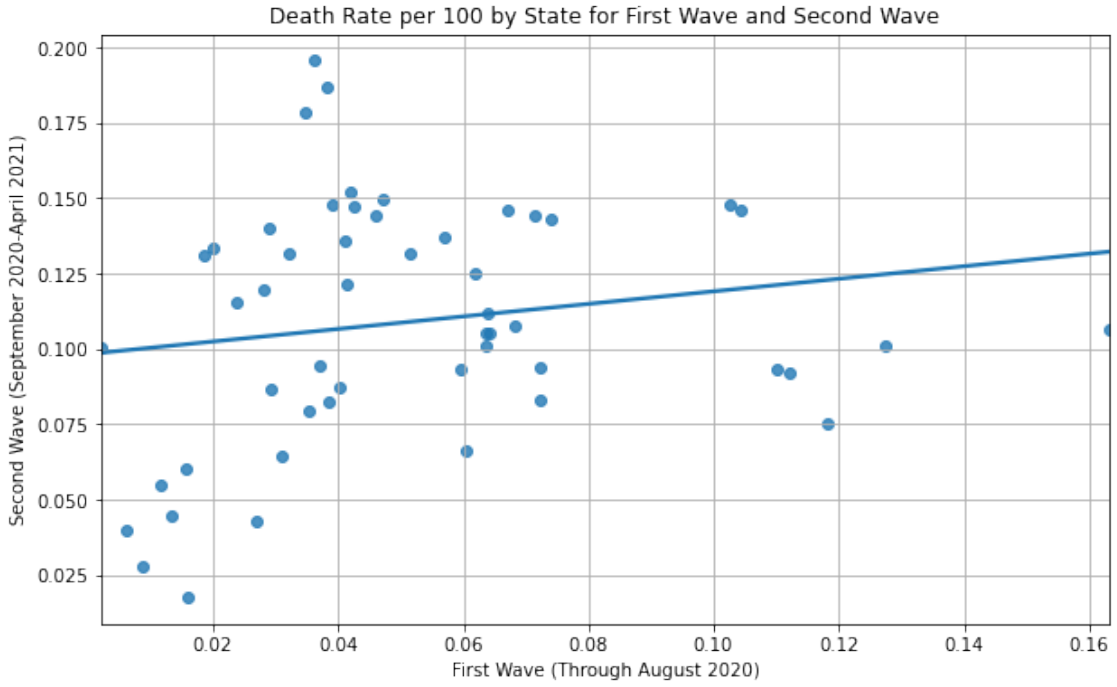
```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  frame['Peak'] = (frame['Year'].astype(int)-2020)*12 +
frame['Month'].astype(int)
<ipython-input-19-a4a636352767>:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  frame['Peak'] = pd.cut(frame['Peak'], bins = cuts, labels = names)
```

[20]:
```
import seaborn as sns
import matplotlib.dates
```

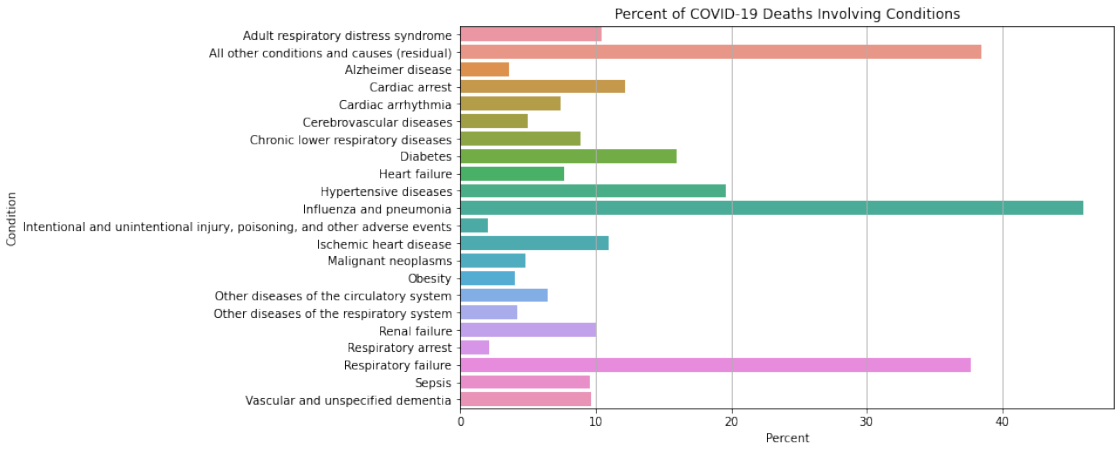
[21]:
```
temp = explore.groupby(['Year/Month','State','Condition','Age Group'])[['Death␣
 ↪Rate','Mention Rate']].sum().reset_index()
temp = temp.merge(center, how='left', left_on=['State'], right_on=['STNAME']).
 ↪drop(['STNAME'], axis = 1)
allages = temp[(temp.Condition == 'COVID-19') & (temp['Age Group'] == 'All␣
 ↪Ages')]
allages = allages[allages.State.isin(get_biggest_states(9))]
fig, ax = plt.subplots(figsize = (10,6))
allages['Date'] = matplotlib.dates.datestr2num(allages['Year/Month'])
ax.xaxis.set_major_formatter(matplotlib.dates.DateFormatter('%Y-%m'))
fmt_year = matplotlib.dates.MonthLocator(interval=2)
ax.xaxis.set_major_locator(fmt_year)
fmt_month = matplotlib.dates.MonthLocator()
ax.xaxis.set_minor_locator(fmt_month)
ax.set_title("COVID Death Rates (per 100) by Month for Top 9 Populated States␣
 ↪plus USA")
plt.grid()
plot = sns.lineplot(data=allages, x="Date",y="Death Rate",hue="State",ax=ax)
plt.savefig("DeathRateTop10.png", dpi = 300)
```



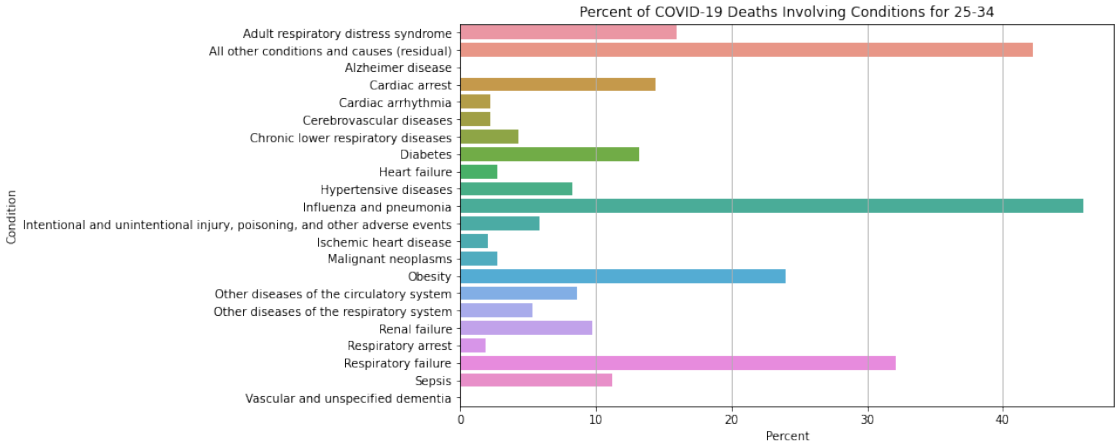
[22]:
```
temp2 = explore.groupby(['Peak','State','Condition','Age Group'])[['Death␣
 ↪Rate','Mention Rate']].sum().reset_index()
temp2.merge(center, how='left', left_on=['State'], right_on=['STNAME']).
 ↪drop(['STNAME'], axis = 1)
period = temp2[(temp2.Condition == 'COVID-19') & (temp2['Age Group'] == 'All␣
 ↪Ages')]
period = period.pivot(index = 'State', columns = 'Peak', values = 'Death Rate')
fig, ax = plt.subplots(figsize = (10,6))
plot = sns.regplot(data=period, x="First",y="Second", ax=ax, ci = None)
ax.set_xlabel("First Wave (Through August 2020)")
ax.set_ylabel("Second Wave (September 2020-April 2021)")
plt.grid()
ax.set_title("Death Rate per 100 by State for First Wave and Second Wave")
plt.savefig("FirstSecondWave.png", dpi = 300)
```



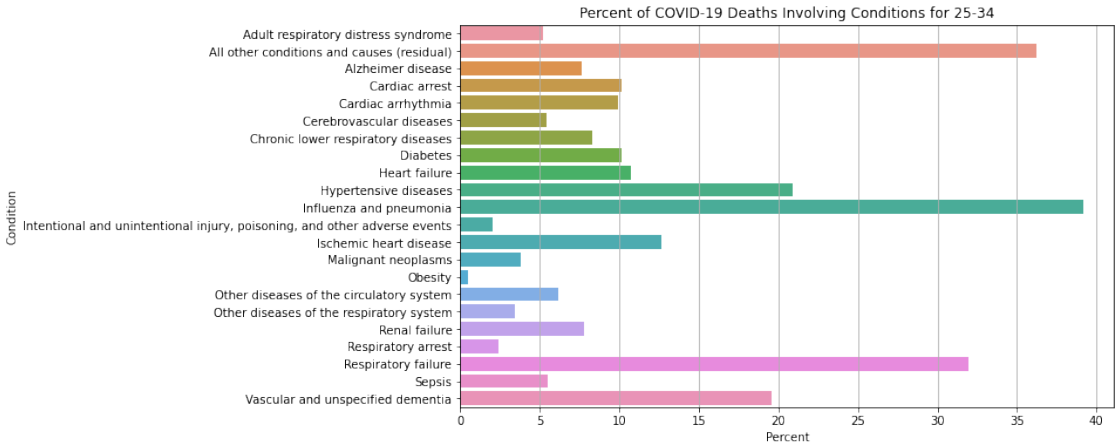
[23]:
```
totald = df.query('Group == "By Total"')
totald = get_percent_of_deaths(totald)
totald['value'] = totald['value']*100
temp2 = totald[(totald['Age Group'] == 'All Ages') & (totald['State'] ==␣
 ↪'United States') & (totald['Condition'] != 'COVID-19')]
fig, ax = plt.subplots(figsize = (10,6))
sns.barplot(data=temp2, y="Condition",x="value",ax=ax)
plt.grid(axis='x')
ax.set_xlabel('Percent')
ax.set_title('Percent of COVID-19 Deaths Involving Conditions')
plt.savefig('PercentConditions.png', dpi = 300)
```



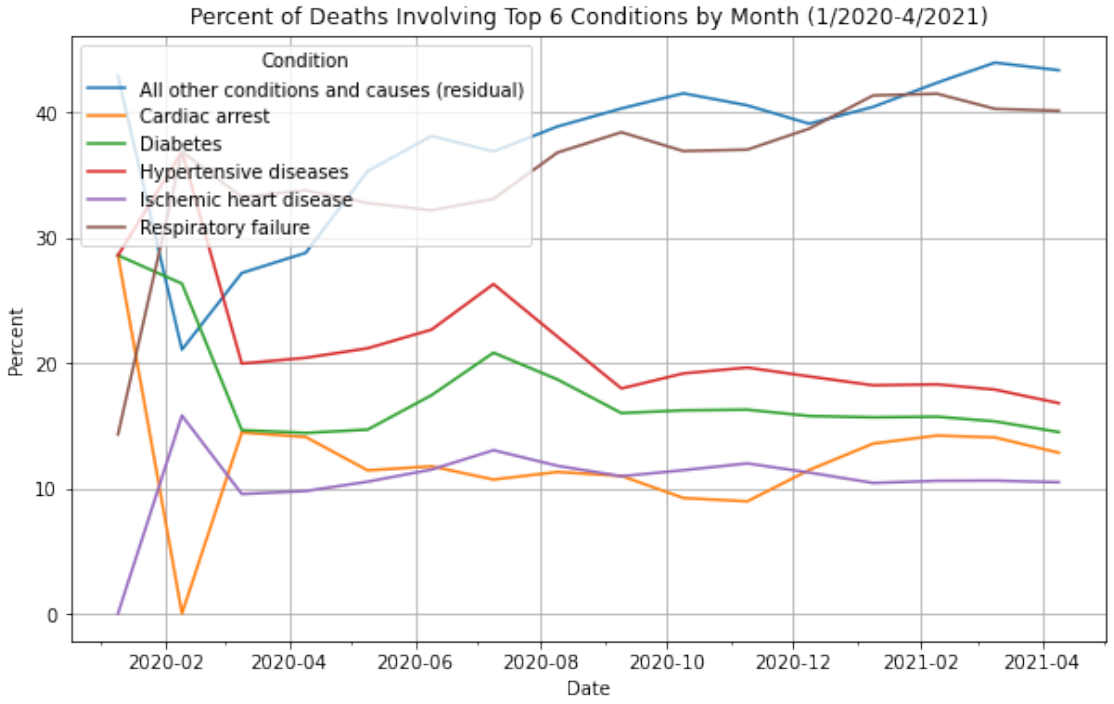
[24]:
```
totald = df.query('Group == "By Total"')
totald = get_percent_of_deaths(totald)
totald['value'] = totald['value']*100
temp2 = totald[(totald['Age Group'] == '25-34') & (totald['State'] == 'United␣
 ↪States') & (totald['Condition'] != 'COVID-19')]
fig, ax = plt.subplots(figsize = (10,6))
sns.barplot(data=temp2, y="Condition",x="value",ax=ax)
plt.grid(axis='x')
ax.set_xlabel('Percent')
ax.set_title('Percent of COVID-19 Deaths Involving Conditions for 25-34')
plt.savefig('PercentConditions25-34.png', dpi = 300)
```



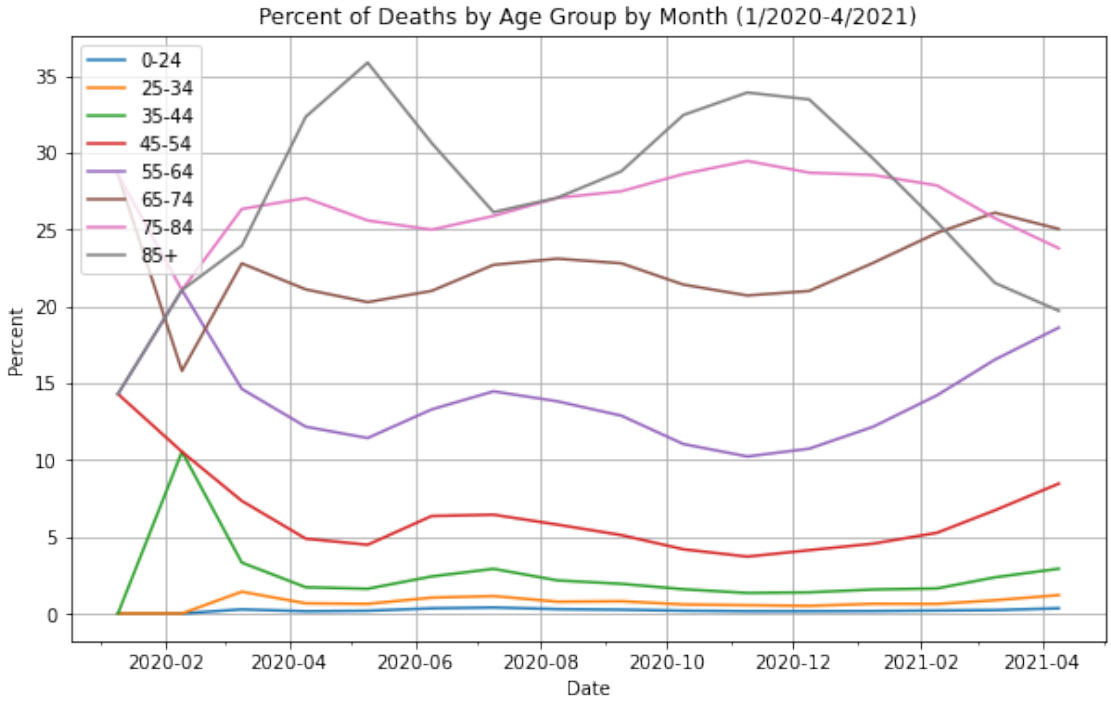
[25]:
```
totald = df.query('Group == "By Total"')
totald = get_percent_of_deaths(totald)
totald['value'] = totald['value']*100
temp2 = totald[(totald['Age Group'] == '85+') & (totald['State'] == 'United␣
 ↪States') & (totald['Condition'] != 'COVID-19')]
fig, ax = plt.subplots(figsize = (10,6))
sns.barplot(data=temp2, y="Condition",x="value",ax=ax)
plt.grid(axis='x')
ax.set_xlabel('Percent')
ax.set_title('Percent of COVID-19 Deaths Involving Conditions for 25-34')
plt.savefig('PercentConditions85+.png', dpi = 300)
```



[26]:
```
totald = df.query('Group == "By Month"')
totald = totald.query('State == "United States"')
totald = get_percent_of_deaths(totald)
totald = totald[(totald['Age Group'] == 'All Ages')]
totald = totald[(totald['Condition'] == 'Influenza and Pnemonia') |␣
↪(totald['Condition'] == 'Respiratory failure') |
                (totald['Condition'] == 'All other conditions and causes␣
↪(residual)') | (totald['Condition'] == 'Hypertensive diseases') |
                (totald['Condition'] == 'Diabetes') | (totald['Condition'] ==␣
↪'Ischemic heart disease') | (totald['Condition'] == 'Cardiac arrest')]
totald = yearmonth(totald)
totald['value'] = totald['value']*100
fig, ax = plt.subplots(figsize = (10,6))
totald['Date'] = matplotlib.dates.datestr2num(totald['Year/Month'])
ax.xaxis.set_major_formatter(matplotlib.dates.DateFormatter('%Y-%m'))
fmt_year = matplotlib.dates.MonthLocator(interval=2)
ax.xaxis.set_major_locator(fmt_year)
fmt_month = matplotlib.dates.MonthLocator()
ax.xaxis.set_minor_locator(fmt_month)
ax.set_title("Percent of Deaths Involving Top 6 Conditions by Month (1/2020-4/
↪2021)")
plt.grid()
plot = sns.lineplot(data=totald, x="Date",y="value",hue="Condition",ax=ax)
ax.set_ylabel('Percent')
plt.savefig("ConditionsByMonth.png", dpi = 300)
```



[27]:
```
totald = df.query('Group == "By Month"')
totald = totald.query('State == "United States"')
totald = get_percent_of_deathsage(totald)
totald = totald[(totald['Condition'] == 'COVID-19') & (totald['Age Group'] !=␣
↪'All Ages')]
totald = yearmonth(totald)
totald['value'] = totald['value']*100
fig, ax = plt.subplots(figsize = (10,6))
totald['Date'] = matplotlib.dates.datestr2num(totald['Year/Month'])
ax.xaxis.set_major_formatter(matplotlib.dates.DateFormatter('%Y-%m'))
fmt_year = matplotlib.dates.MonthLocator(interval=2)
ax.xaxis.set_major_locator(fmt_year)
fmt_month = matplotlib.dates.MonthLocator()
ax.xaxis.set_minor_locator(fmt_month)
ax.set_title("Percent of Deaths by Age Group by Month (1/2020-4/2021)")
plt.grid()
plot = sns.lineplot(data=totald, x="Date",y="value",hue="Age Group",ax=ax)
ax.legend(loc = "upper left")
ax.set_ylabel('Percent')
plt.savefig("AgeGroupByMonth.png", dpi = 300)
```



1.1 Data Preprocessing

[28]:
```
df_new = df.drop(['Death Rate','Mention Rate','Number of Mentions','Condition␣
 ↪Group','Population'], axis = 1)
df_new = df_new.pivot(index = ['Group','Year','Month','State','Age Group'],␣
 ↪columns = 'Condition', values = 'COVID-19 Deaths').reset_index()
df_new = df_new.merge(number2, how = 'left', left_on =␣
 ↪['Group','Year','Month','State', 'Age Group'], right_on =␣
 ↪['Group','Year','Month','State','Age Group'])
df_new = df_new.drop(['COVID-19'], axis = 1)
df_new = df_new.merge(Drate, how = 'left', left_on = ['State','Age Group'],␣
 ↪right_on = ['State','Age Group'])
df_new = df_new.merge(AverageAge, how = 'left', left_on = ['State','Age␣
 ↪Group'], right_on = ['State','Age Group'])
df_new = df_new.merge(AverageAgeA, how = 'left', left_on = ['State','Age␣
 ↪Group'], right_on = ['State','Age Group'])
df_new = df_new.merge(degreematrix, how = 'left', left_on =␣
 ↪['State','Year','Month'], right_on = ['State','Year','Month'])
df_new = df_new.merge(population, how='left', left_on=['State','Age Group'],␣
 ↪right_on=['NAME','AgeGroup'])
df_new = df_new.merge(center, how='left', left_on=['State'],␣
 ↪right_on=['STNAME'])
df_new = df_new.drop(['NAME','STNAME','AgeGroup'], axis = 1)
```

[29]:
```
df_cut = cutyearmonth(df_new[(df_new['Group'] == 'By Month')])
df_cut[(df_cut['Age Group'] == '0-24') & (df_cut['State'] == 'Alabama')]
df_cut = df_cut.replace(np.nan, 0)
df_cut['CDD'] = df_cut['CDD'].astype(int)
df_cut['HDD'] = df_cut['HDD'].astype(int)
temp = df_cut.groupby(['State','Age Group', 'Peak']).sum().reset_index()
temp = temp.drop(['Year','Month'], axis = 1)
peak_one = temp[(temp['Peak'] == 'First') & (temp['Age Group'] != 'All Ages')]
OriginalPeakOne = peak_one
peak_one.to_csv('OriginalPeakOne.csv')
peak_one = peak_one.drop(['State','Age Group','Peak'], axis = 1)
peak_two = temp[(temp['Peak'] == 'Second') & (temp['Age Group'] != 'All Ages')]
OriginalPeakTwo = peak_two
peak_two.to_csv('OriginalPeakTwo.csv')
peak_two = peak_two.drop(['State','Age Group','Peak'], axis = 1)
death_one = peak_one['Total Deaths'].copy()
death_two = peak_two['Total Deaths'].copy()
for i in range(28):
    peak_one[peak_one.columns[i]] = peak_one[peak_one.columns[i]]/death_one
    peak_two[peak_two.columns[i]] = peak_two[peak_two.columns[i]]/death_two
for i in range(28,34):
    peak_one[peak_one.columns[i]] = peak_one[peak_one.columns[i]]/9
    peak_two[peak_two.columns[i]] = peak_two[peak_two.columns[i]]/7
peak_one = peak_one.drop(['Population','Total Deaths'], axis = 1)
peak_two = peak_two.drop(['Population','Total Deaths'], axis = 1)
```

```
<ipython-input-19-a4a636352767>:23: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  frame['Peak'] = (frame['Year'].astype(int)-2020)*12 +
frame['Month'].astype(int)
<ipython-input-19-a4a636352767>:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  frame['Peak'] = pd.cut(frame['Peak'], bins = cuts, labels = names)
```

[30]:
```
from sklearn.preprocessing import StandardScaler

peak_one = peak_one.fillna(0)
peak_two = peak_two.fillna(0)
transform_one = StandardScaler().fit_transform(peak_one)
transform_two = StandardScaler().fit_transform(peak_two)
```

[31]:
```
PeakOneNoPCAScale = peak_one
PeakTwoNoPCAScale = peak_two
peak_one.to_csv('PeakOneNo-PCA-Scale.csv')
peak_two.to_csv('PeakTwoNo-PCA-Scale.csv')
```

1.2 Principal Components Analysis

[32]:
```
from pca import pca

pca1 = pca()
result_one = pca1.fit_transform(transform_one)
result_one['explained_var']
```

```
[pca] >Column labels are auto-completed.
[pca] >Row labels are auto-completed.
[pca] >The PCA reduction is performed to capture [95.0%] explained variance
using the [34] columns of the input data.
[pca] >Fitting using PCA..
[pca] >Computing loadings and PCs..
[pca] >Computing explained variance..
[pca] >Number of components is [13] that covers the [95.00%] explained variance.
[pca] >Outlier detection using Hotelling T2 test with alpha=[0.05] and
n_components=[13]
[pca] >Outlier detection using SPE/DmodX with n_std=[2]
```

[32]:
```
array([0.51031497, 0.61531457, 0.6882964 , 0.75768038, 0.8019259 ,
       0.83322461, 0.86113476, 0.88283693, 0.90259822, 0.91850608,
       0.93244629, 0.94240352, 0.95088663, 0.95673277, 0.96215778,
       0.96683291, 0.97115003, 0.97519875, 0.97880819, 0.98185218,
       0.98458502, 0.98695163, 0.98927303, 0.99135678, 0.99322785,
       0.99485192, 0.99628902, 0.99757109, 0.99846944, 0.99927887,
       0.99981705, 0.9999778 , 0.9999959 , 1.        ])
```

[33]:
```
round(result_one['loadings'],3)
```

[33]:

	1	2	3	4	5	6	7	8	9	10	\
PC1	0.121	0.222	0.137	0.169	0.203	0.208	0.206	0.185	0.197	0.179	
PC2	0.215	0.034	-0.268	0.081	-0.178	-0.076	-0.082	0.196	-0.204	0.105	
PC3	0.226	-0.013	0.045	-0.190	-0.119	-0.167	-0.066	0.095	-0.073	0.185	
PC4	-0.250	-0.064	0.228	-0.172	0.134	0.041	0.046	-0.134	0.129	-0.062	
PC5	0.327	-0.132	0.126	-0.196	0.095	0.041	0.126	0.194	0.078	0.376	
PC6	-0.179	-0.055	-0.085	-0.317	0.049	0.110	0.192	0.247	-0.007	0.190	
PC7	0.103	-0.073	0.013	0.074	0.047	0.021	-0.005	0.059	-0.005	0.079	
PC8	-0.187	0.120	-0.015	-0.167	0.011	0.104	0.173	0.101	0.138	0.020	
PC9	0.067	0.043	0.442	0.089	-0.037	-0.237	-0.213	-0.036	0.116	-0.016	
PC10	-0.097	-0.087	-0.275	0.019	-0.065	-0.079	-0.048	0.011	-0.166	-0.005	
PC11	0.021	-0.039	-0.053	-0.095	-0.120	-0.215	0.158	0.082	0.095	0.023	
PC12	-0.232	-0.054	0.168	0.338	-0.069	-0.078	0.290	0.124	-0.063	0.120	

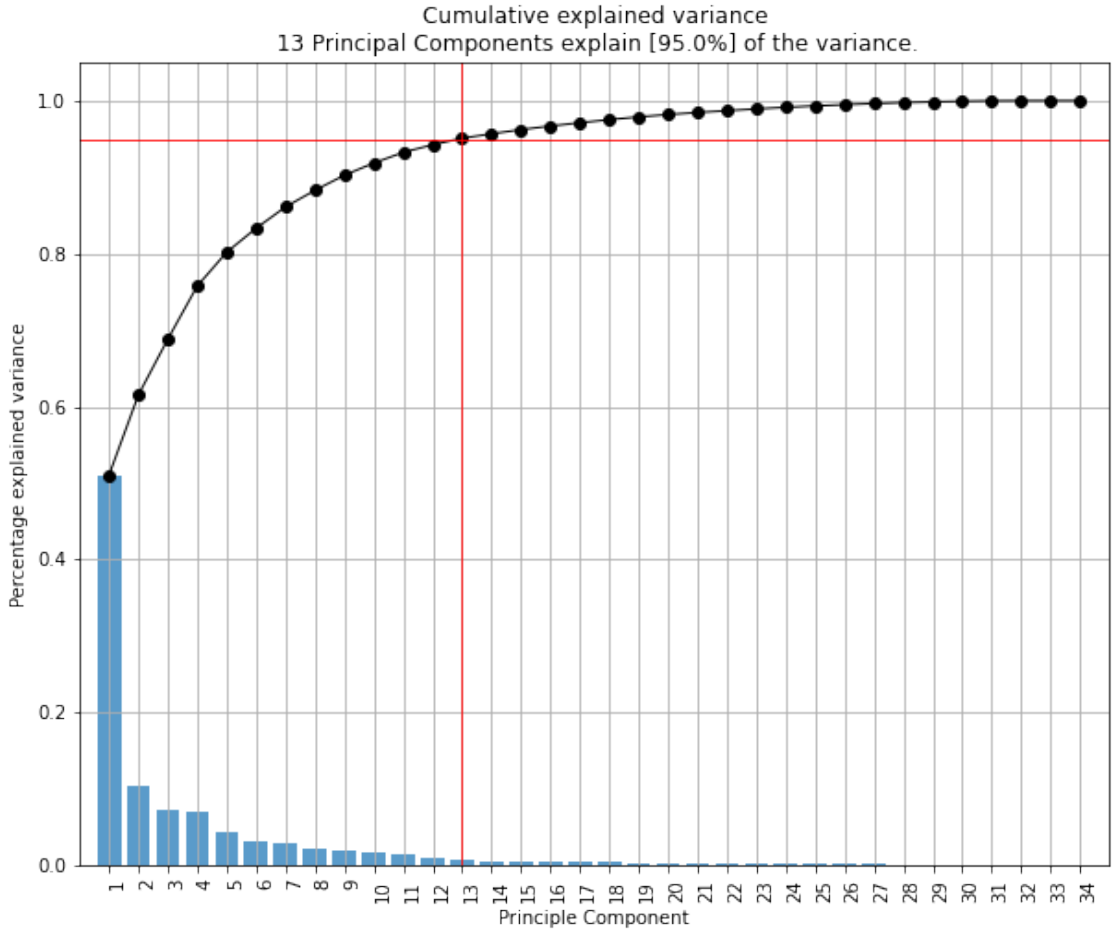
```
PC13 -0.180  0.281  0.010 -0.081 -0.046 -0.090  0.241 -0.126 -0.061 -0.186
PC14 -0.222  0.178 -0.099 -0.466  0.114  0.045 -0.208  0.209  0.145  0.147
PC15  0.100 -0.026 -0.335  0.022  0.187  0.314 -0.157 -0.306  0.033 -0.216
PC16  0.181 -0.060  0.383 -0.127 -0.095 -0.045 -0.013  0.056 -0.105 -0.054
PC17  0.100 -0.100 -0.292 -0.117  0.109  0.110  0.066 -0.006 -0.159  0.122
PC18  0.055 -0.152  0.313 -0.180  0.246  0.294 -0.070 -0.267  0.014 -0.169
PC19 -0.160 -0.268 -0.043  0.283  0.228  0.108 -0.261  0.129  0.076  0.175
PC20  0.039 -0.031  0.081 -0.280 -0.120 -0.090  0.195 -0.160 -0.400 -0.153
PC21 -0.069 -0.124  0.056 -0.183  0.156  0.138 -0.059  0.034  0.097  0.070
PC22  0.132 -0.104 -0.136 -0.195 -0.324  0.080  0.276 -0.206  0.325 -0.066
PC23 -0.089 -0.358 -0.121 -0.027 -0.188 -0.322  0.014 -0.164  0.546  0.025
PC24 -0.202 -0.003  0.102 -0.173 -0.105 -0.079 -0.274 -0.043 -0.273  0.101
PC25  0.112  0.175 -0.004 -0.056  0.157  0.013  0.212 -0.037 -0.069 -0.003
PC26  0.046  0.153 -0.108  0.068  0.493 -0.352  0.326  0.003  0.038 -0.148
PC27  0.211  0.356  0.053  0.124 -0.250  0.343  0.047 -0.045  0.183  0.062
PC28 -0.047 -0.399  0.085  0.109 -0.192  0.349  0.322 -0.079 -0.206  0.130
PC29  0.135 -0.319  0.004 -0.005  0.330 -0.182  0.166 -0.164 -0.119  0.136
PC30 -0.470 -0.042  0.037  0.094 -0.082  0.119  0.149  0.176 -0.057 -0.077
PC31  0.187 -0.268  0.016 -0.058 -0.001  0.082  0.006  0.595  0.059 -0.655
PC32 -0.035  0.041 -0.004 -0.011 -0.011  0.005  0.012 -0.008 -0.023  0.026
PC33 -0.010  0.007 -0.004  0.012  0.003  0.011 -0.037 -0.002  0.017 -0.005
PC34  0.001 -0.002  0.000 -0.001  0.005 -0.002  0.000 -0.003 -0.002  0.000

      …     25     26     27     28     29     30     31     32     33  \
PC1   …  0.205  0.101  0.216  0.115  0.146  0.146  0.059 -0.073 -0.058
PC2   …  0.171  0.047  0.037 -0.288 -0.194 -0.197  0.317 -0.323 -0.303
PC3   …  0.148 -0.012  0.178  0.315  0.366  0.366  0.059 -0.035 -0.035
PC4   … -0.140  0.027 -0.047  0.160  0.004  0.008  0.456 -0.440 -0.476
PC5   …  0.102 -0.300 -0.137 -0.083 -0.251 -0.248  0.017  0.016  0.029
PC6   … -0.101  0.517 -0.048 -0.074  0.035  0.033 -0.049  0.119  0.107
PC7   …  0.051 -0.185 -0.106 -0.009 -0.010 -0.009  0.105  0.054  0.090
PC8   … -0.067 -0.558  0.000  0.014  0.026  0.025  0.033  0.058  0.039
PC9   …  0.092  0.031  0.011  0.102 -0.171 -0.166 -0.042  0.060  0.113
PC10  … -0.085 -0.395  0.070  0.119  0.214  0.212  0.097  0.054 -0.027
PC11  … -0.066  0.273  0.112 -0.138  0.010  0.007  0.003 -0.012 -0.022
PC12  … -0.051 -0.008  0.000  0.020 -0.014 -0.014 -0.001  0.013 -0.021
PC13  …  0.064 -0.120  0.291 -0.465 -0.041 -0.051  0.030  0.059  0.044
PC14  … -0.073 -0.075  0.123 -0.066 -0.075 -0.075 -0.014 -0.006  0.044
PC15  …  0.056  0.070  0.094  0.048 -0.042 -0.040  0.061  0.032 -0.007
PC16  … -0.097 -0.061  0.025 -0.435  0.253  0.236  0.018  0.022 -0.088
PC17  …  0.078  0.041 -0.099 -0.128  0.089  0.084  0.222  0.115  0.122
PC18  …  0.175  0.019 -0.030 -0.250  0.091  0.080  0.118  0.104  0.064
PC19  … -0.088 -0.102  0.237 -0.305  0.032  0.021 -0.384 -0.171 -0.178
PC20  …  0.007 -0.054  0.206  0.222 -0.222 -0.211 -0.144 -0.107  0.061
PC21  …  0.055 -0.025  0.087  0.197 -0.111 -0.104 -0.098 -0.149  0.114
PC22  …  0.130 -0.038 -0.126 -0.053  0.096  0.085 -0.492 -0.340 -0.160
PC23  …  0.061 -0.008  0.121 -0.077 -0.041 -0.036  0.314  0.274  0.056
```

```
PC24  …  0.089  0.018 -0.086 -0.102  0.075  0.067 -0.001 -0.096  0.097
PC25  …  0.007 -0.012 -0.065  0.049 -0.047 -0.040 -0.113  0.516 -0.604
PC26  …  0.063 -0.038 -0.151 -0.089  0.082  0.072  0.039 -0.314  0.295
PC27  … -0.255  0.021  0.166  0.010 -0.023 -0.018  0.193 -0.074  0.218
PC28  … -0.016 -0.037  0.028 -0.026 -0.011 -0.010  0.134 -0.029  0.151
PC29  … -0.165  0.037  0.284  0.078 -0.025 -0.031 -0.026  0.013 -0.021
PC30  …  0.311  0.014 -0.077  0.086 -0.008 -0.004  0.030  0.065 -0.027
PC31  … -0.046 -0.022  0.097  0.061 -0.021 -0.017  0.037  0.020  0.005
PC32  …  0.131  0.004  0.672  0.042 -0.017 -0.023  0.002  0.011 -0.004
PC33  …  0.725  0.004  0.021  0.004  0.004 -0.005  0.005  0.002  0.001
PC34  …  0.006 -0.001  0.003 -0.019 -0.698  0.716 -0.005 -0.006  0.002
```

```
        34
PC1   0.055
PC2   0.005
PC3  -0.088
PC4  -0.070
PC5   0.239
PC6  -0.090
PC7  -0.915
PC8   0.000
PC9  -0.030
PC10  0.182
PC11 -0.028
PC12  0.003
PC13 -0.111
PC14 -0.067
PC15 -0.036
PC16  0.028
PC17  0.020
PC18  0.075
PC19 -0.055
PC20 -0.059
PC21  0.001
PC22 -0.082
PC23  0.036
PC24  0.011
PC25 -0.040
PC26  0.048
PC27 -0.002
PC28  0.021
PC29 -0.043
PC30  0.016
PC31  0.001
PC32 -0.002
PC33  0.002
PC34  0.000
```

[34 rows x 34 columns]

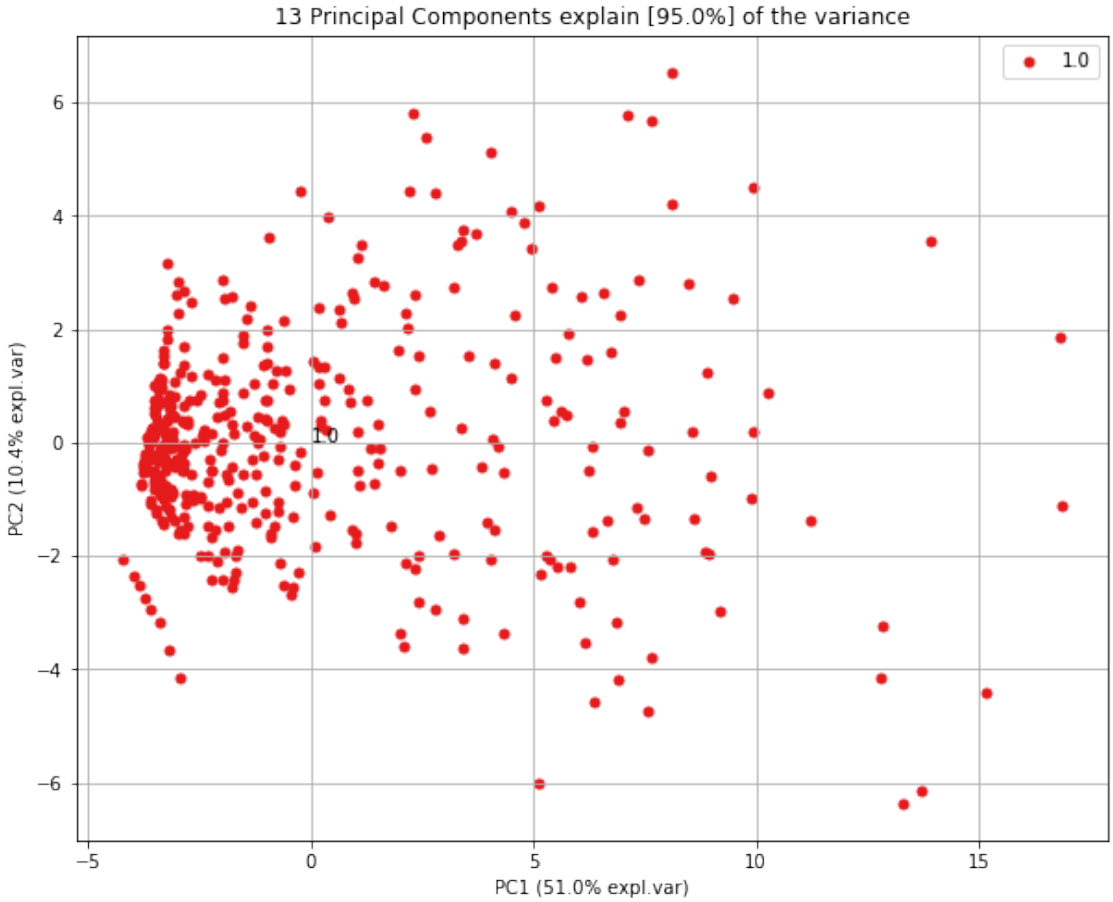
[218]:
```
pca1.plot()
```



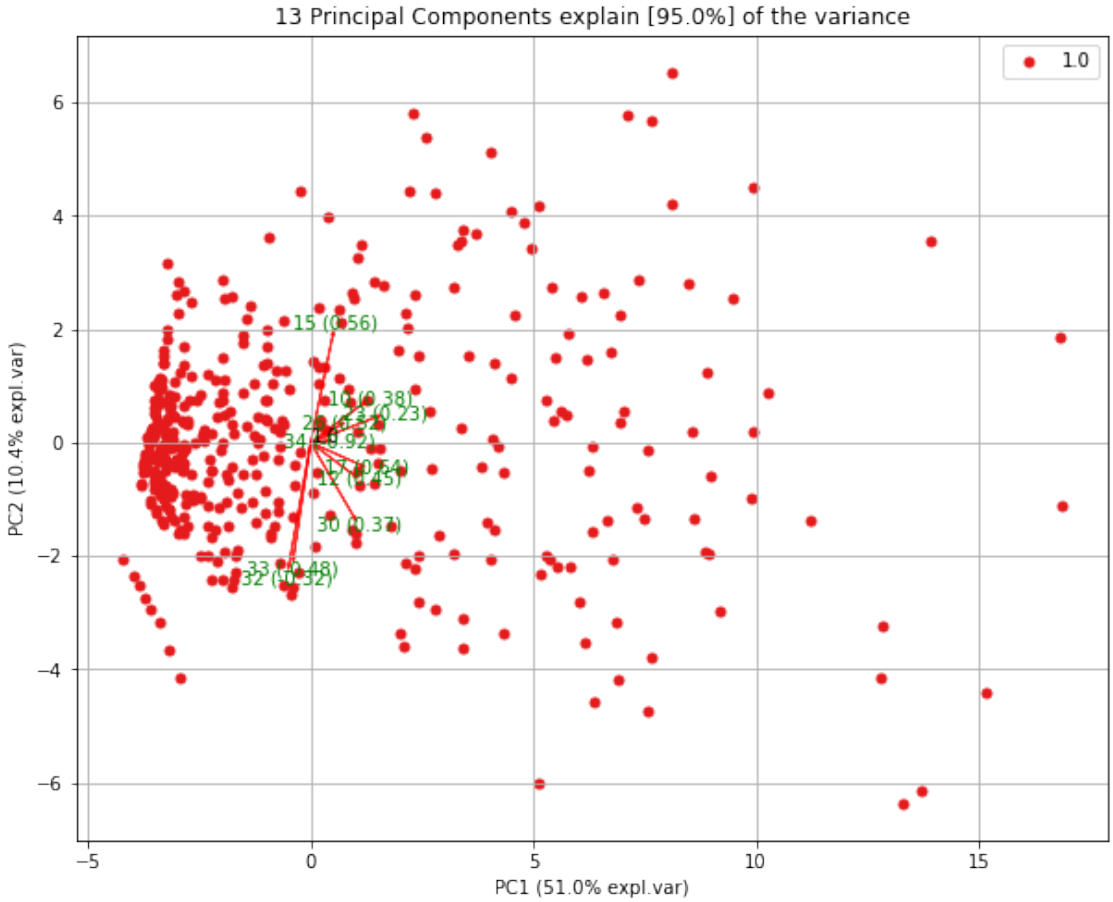
<Figure size 432x288 with 0 Axes>

[35]:
```
pca1.scatter()
```

[35]:
```
(<Figure size 720x576 with 1 Axes>,
 <AxesSubplot:title={'center':'13 Principal Components explain [95.0%] of the
variance'}, xlabel='PC1 (51.0% expl.var)', ylabel='PC2 (10.4% expl.var)'>)
```



[36]:
```
pca1.biplot(n_feat=10)
```



[36]:
```
(<Figure size 720x576 with 1 Axes>,
 <AxesSubplot:title={'center':'13 Principal Components explain [95.0%] of the
variance'}, xlabel='PC1 (51.0% expl.var)', ylabel='PC2 (10.4% expl.var)'>)
```

[37]:
```
pca_transform_one = pca1.transform(transform_one)
pca_transform_one
```

```
[pca] >Column labels are auto-completed.
[pca] >Row labels are auto-completed.
```

[37]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	\
1.0	-3.292124	1.517491	-1.457907	1.845980	1.076299	-0.563544	-0.272162	
1.0	-2.939893	1.249270	-0.767865	1.825723	0.624506	-0.533552	-0.340818	
1.0	-2.297535	1.214677	-0.044416	1.688675	0.195339	-0.652316	-0.529986	
1.0	-0.975263	1.411792	0.541269	1.287771	-0.561943	-1.171058	-0.785404	
1.0	1.966947	1.621573	-0.972956	0.357727	-2.369732	-2.087097	-0.550462	
..	…	…	…	…	…	…	…	
1.0	-3.268245	-1.103020	-0.412937	-0.851150	0.060963	0.212598	0.764965	
1.0	-3.130316	-1.304727	-0.063660	-0.825773	-0.160319	0.232276	0.755885	

```
1.0 -2.922118 -1.552744  0.381684 -0.776911 -0.422178  0.223112  0.720136
1.0 -2.483724 -1.988948  1.271168 -0.643234 -0.827157  0.152944  0.638518
1.0 -2.246967 -2.430993  1.857209 -0.451627 -1.067055  0.075380  0.611717

          PC8       PC9      PC10  …      PC25      PC26      PC27  \
1.0 -0.099489 -0.077273 -0.301191  … -0.079729  0.096672  0.037110
1.0 -0.081871 -0.332738  0.050688  … -0.150005  0.164561  0.057741
1.0 -0.080562 -0.392753  0.322466  … -0.286825 -0.066616  0.232755
1.0  0.209514 -0.208613  0.197257  … -0.173355 -0.167039  0.019493
1.0 -2.677389 -0.715190  0.199660  …  0.144898  0.066181  0.441501
..        …       …        …  …       …        …       …
1.0  0.131586 -0.153008  0.103790  …  0.396374 -0.124038 -0.242773
1.0  0.154633 -0.282505  0.299056  …  0.365867 -0.070534 -0.259087
1.0  0.179678 -0.380354  0.528151  …  0.332534 -0.075166 -0.233188
1.0  0.162062 -0.430833  0.870294  …  0.383595 -0.115894 -0.228300
1.0  0.189911 -0.389650  1.127690  …  0.416564 -0.200669 -0.210535

         PC28      PC29      PC30      PC31      PC32      PC33      PC34
1.0  0.026970 -0.069565 -0.018250  0.019291 -0.022332  0.006024 -0.007115
1.0  0.064913 -0.083425 -0.092666  0.017518  0.091080  0.004628 -0.012825
1.0  0.085900  0.347972 -0.022655  0.101888 -0.185631 -0.010758 -0.005010
1.0  0.043950  0.216515 -0.326222 -0.090993 -0.049849 -0.018462 -0.000256
1.0 -0.207816 -0.092532  0.025432 -0.085825  0.016253 -0.013969 -0.000189
..        …       …        …       …       …       …       …
1.0 -0.126351 -0.054123  0.023038 -0.047204 -0.060096 -0.002893 -0.003443
1.0 -0.138704 -0.068098  0.029087 -0.055650 -0.071903 -0.002936  0.013661
1.0 -0.149809  0.003439  0.035896 -0.026942  0.085435  0.002766  0.011258
1.0 -0.064618 -0.076879  0.007085  0.020255 -0.013254 -0.011374 -0.006060
1.0 -0.096337  0.027823  0.098980  0.094389  0.089212 -0.005908  0.003014

[416 rows x 34 columns]
```

[38]:
```
pca2 = pca()
result_two = pca2.fit_transform(transform_two)
result_two['explained_var']
```

```
[pca] >Column labels are auto-completed.
[pca] >Row labels are auto-completed.
[pca] >The PCA reduction is performed to capture [95.0%] explained variance
using the [34] columns of the input data.
[pca] >Fitting using PCA..
[pca] >Computing loadings and PCs..
[pca] >Computing explained variance..
[pca] >Number of components is [13] that covers the [95.00%] explained variance.
[pca] >Outlier detection using Hotelling T2 test with alpha=[0.05] and
n_components=[13]
[pca] >Outlier detection using SPE/DmodX with n_std=[2]
```

[38]:
```
array([0.54063528, 0.65155854, 0.72078523, 0.77314315, 0.80802971,
       0.83904424, 0.86573044, 0.88640509, 0.90472609, 0.92100427,
       0.9351236 , 0.94337427, 0.95116638, 0.95788408, 0.96352674,
       0.96893475, 0.97332021, 0.97717444, 0.98040776, 0.9830532 ,
       0.98547002, 0.98772192, 0.98972347, 0.99158081, 0.99336714,
       0.99488708, 0.9962468 , 0.99740641, 0.99845201, 0.99932434,
       0.99987723, 0.99997787, 0.99999607, 1.        ])
```

[39]:
```
round(result_two['loadings'], 3)
```

[39]:

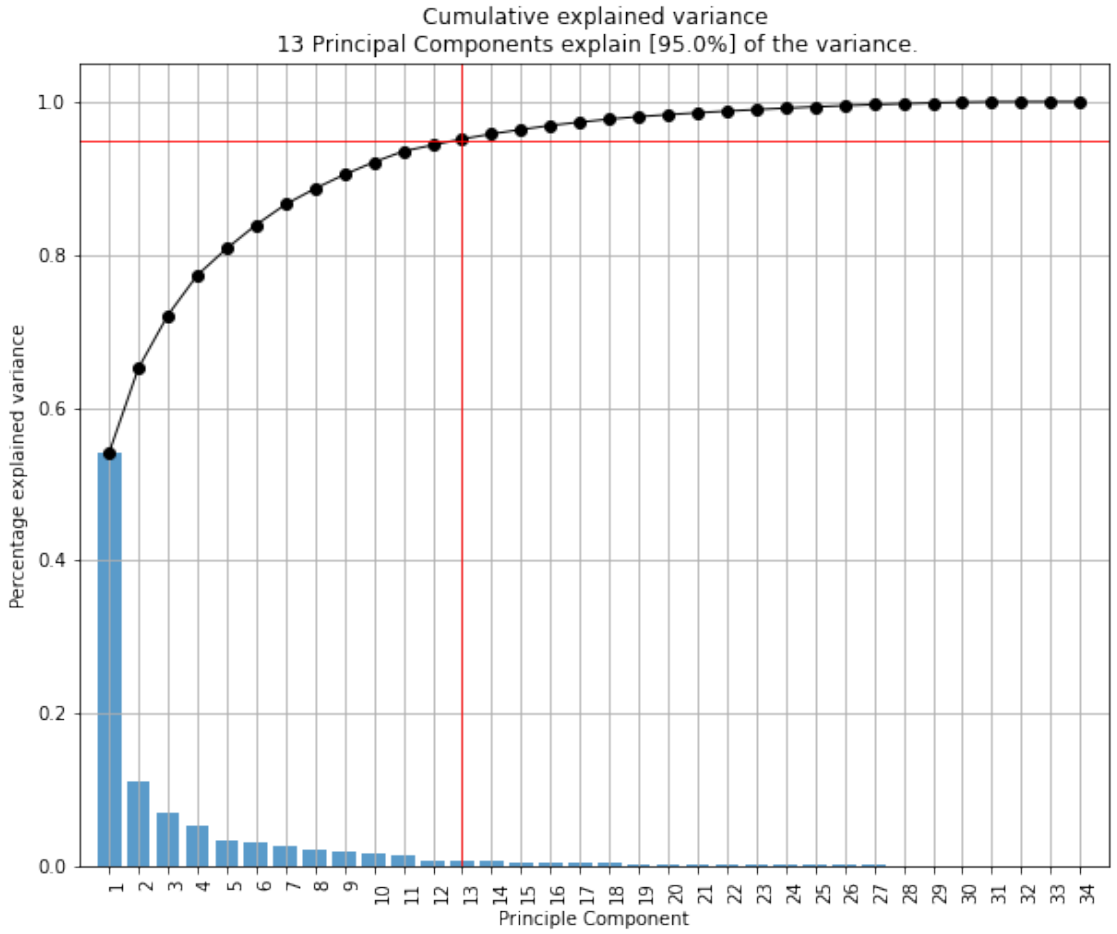
	1	2	3	4	5	6	7	8	9	10	\
PC1	0.121	0.214	0.133	0.148	0.200	0.203	0.202	0.194	0.196	0.192	
PC2	-0.258	-0.021	0.316	-0.217	0.200	0.124	0.102	-0.152	0.221	0.010	
PC3	-0.141	-0.125	0.138	0.001	0.114	0.109	-0.031	-0.106	0.104	-0.004	
PC4	-0.165	-0.103	-0.025	0.106	0.093	0.181	0.049	0.053	0.063	0.002	
PC5	0.022	0.014	0.079	-0.182	0.014	0.052	0.088	0.173	0.017	0.127	
PC6	0.086	-0.054	0.006	0.373	-0.051	-0.064	-0.192	-0.208	-0.051	-0.233	
PC7	-0.004	0.043	0.025	-0.291	-0.044	-0.066	-0.021	0.046	-0.031	0.119	
PC8	0.286	-0.057	0.402	0.185	0.000	-0.018	-0.238	0.103	0.084	0.242	
PC9	0.533	-0.054	-0.006	0.020	0.066	0.080	0.181	0.038	0.077	0.175	
PC10	-0.356	0.187	0.162	-0.173	-0.032	-0.031	0.190	0.117	0.047	-0.007	
PC11	0.053	-0.212	-0.223	0.001	-0.063	-0.017	-0.042	0.297	-0.102	0.497	
PC12	0.346	0.037	0.201	0.023	0.009	0.078	-0.029	-0.133	0.097	-0.188	
PC13	0.058	-0.017	-0.055	-0.321	0.034	0.027	-0.003	-0.057	0.035	-0.003	
PC14	-0.310	-0.158	-0.113	0.301	0.084	-0.134	0.090	0.109	0.040	0.176	
PC15	0.107	0.133	-0.482	0.182	0.199	-0.134	0.241	-0.147	-0.054	-0.173	
PC16	0.017	0.008	0.171	0.080	-0.047	-0.163	0.061	-0.030	0.157	0.043	
PC17	0.201	0.053	-0.305	-0.481	0.160	-0.127	0.025	0.021	-0.040	-0.073	
PC18	-0.051	0.216	-0.030	0.067	-0.133	-0.123	0.099	0.199	-0.204	0.294	
PC19	0.101	0.229	0.190	0.089	-0.013	-0.092	0.339	0.043	-0.155	-0.061	
PC20	0.120	-0.006	0.182	-0.007	-0.204	-0.550	0.452	0.036	0.198	-0.109	
PC21	-0.031	-0.250	0.049	0.017	-0.112	0.450	0.393	-0.032	-0.184	-0.129	
PC22	0.026	-0.113	-0.109	0.212	0.112	-0.025	-0.021	0.079	0.055	-0.040	
PC23	-0.045	0.140	-0.100	0.159	-0.127	0.283	0.377	-0.141	-0.075	0.061	
PC24	0.042	0.100	-0.292	0.056	-0.336	0.243	0.015	0.030	0.537	0.125	
PC25	-0.107	-0.180	0.025	0.057	-0.464	-0.218	-0.037	-0.138	-0.100	0.038	
PC26	0.048	-0.190	0.061	-0.059	-0.317	0.122	0.015	0.529	-0.239	-0.362	
PC27	0.018	0.249	0.117	0.137	0.330	0.022	-0.054	0.176	-0.417	-0.021	
PC28	0.149	0.173	0.074	-0.117	-0.345	0.200	-0.057	-0.390	-0.315	0.212	
PC29	-0.088	-0.128	0.039	0.000	0.103	-0.144	0.141	-0.351	-0.214	0.344	
PC30	-0.139	0.634	-0.042	0.067	-0.194	0.018	-0.213	0.064	0.052	-0.038	
PC31	-0.039	0.062	0.029	0.065	-0.052	0.036	-0.011	-0.010	0.039	-0.043	
PC32	-0.002	0.037	-0.001	0.001	-0.016	-0.000	0.012	-0.002	-0.005	0.002	
PC33	-0.002	-0.016	0.009	0.000	0.004	-0.004	-0.038	-0.025	-0.011	-0.005	
PC34	-0.001	-0.006	-0.000	0.000	0.005	-0.001	0.004	-0.003	0.001	-0.000	

…	25	26	27	28	29	30	31	32	33	\

```
PC1   …  0.209  0.027  0.214  0.154  0.186  0.187 -0.010 -0.034 -0.037
PC2   … -0.150 -0.057 -0.043  0.281  0.109  0.113 -0.157  0.270  0.248
PC3   … -0.110  0.047 -0.095  0.128 -0.039 -0.035  0.498 -0.487 -0.493
PC4   … -0.149  0.281 -0.193 -0.214 -0.294 -0.294 -0.218  0.092  0.101
PC5   … -0.024  0.154 -0.053 -0.069 -0.033 -0.033  0.260  0.180  0.184
PC6   …  0.024  0.503  0.020  0.127  0.133  0.133 -0.014  0.083  0.185
PC7   …  0.029  0.718  0.094  0.079  0.102  0.102 -0.033 -0.074 -0.138
PC8   …  0.071 -0.128  0.004  0.057 -0.256 -0.250 -0.257 -0.083 -0.061
PC9   …  0.061  0.224 -0.115 -0.105 -0.182 -0.180 -0.009 -0.025 -0.026
PC10  …  0.035  0.176  0.126 -0.215 -0.196 -0.197  0.006  0.019 -0.101
PC11  … -0.141 -0.036 -0.024  0.095  0.213  0.211  0.032  0.008  0.061
PC12  … -0.154  0.062 -0.137  0.091  0.099  0.100  0.163  0.121  0.000
PC13  …  0.021  0.034 -0.003  0.223 -0.072 -0.068 -0.432 -0.222 -0.079
PC14  … -0.024  0.109 -0.116  0.283 -0.013 -0.008 -0.143 -0.150  0.058
PC15  … -0.002  0.004 -0.015  0.214 -0.165 -0.157  0.008  0.087 -0.115
PC16  …  0.092  0.009  0.019  0.008 -0.133 -0.125  0.422  0.296  0.128
PC17  …  0.010 -0.047 -0.001  0.044 -0.055 -0.052  0.077  0.092  0.020
PC18  … -0.075  0.022 -0.096  0.111 -0.093 -0.081  0.208  0.184  0.008
PC19  … -0.248 -0.010  0.126 -0.244  0.143  0.129 -0.214  0.089 -0.344
PC20  …  0.005 -0.039 -0.171  0.141  0.064  0.064 -0.074 -0.222  0.106
PC21  …  0.209 -0.025 -0.156  0.192  0.030  0.031 -0.025  0.269 -0.242
PC22  … -0.100  0.020  0.097 -0.418  0.148  0.129 -0.058  0.073 -0.131
PC23  … -0.004  0.000  0.153 -0.082 -0.053 -0.055  0.090 -0.398  0.432
PC24  … -0.078  0.001 -0.002  0.113  0.002  0.006 -0.062  0.167 -0.227
PC25  …  0.000 -0.003  0.249  0.198 -0.139 -0.125 -0.034  0.207 -0.210
PC26  …  0.054  0.010 -0.117 -0.008  0.030  0.029 -0.009 -0.106  0.067
PC27  … -0.085  0.002  0.041  0.350 -0.091 -0.076 -0.006  0.079 -0.090
PC28  … -0.163 -0.017 -0.006  0.046 -0.009 -0.003 -0.012 -0.100  0.097
PC29  …  0.220  0.003 -0.261 -0.254  0.115  0.096 -0.075  0.092 -0.141
PC30  …  0.203  0.002 -0.359 -0.024  0.070  0.072 -0.069 -0.026 -0.073
PC31  … -0.208 -0.007 -0.014 -0.034 -0.027 -0.023 -0.056  0.054 -0.056
PC32  …  0.052 -0.001  0.664  0.020 -0.011 -0.018 -0.001  0.010 -0.007
PC33  …  0.732 -0.001  0.065 -0.016  0.017 -0.016 -0.003  0.005 -0.008
PC34  …  0.019 -0.000  0.004 -0.023 -0.696  0.717 -0.005 -0.002 -0.002

         34
PC1   0.011
PC2   0.044
PC3  -0.093
PC4   0.210
PC5  -0.749
PC6  -0.278
PC7   0.334
PC8  -0.019
PC9  -0.042
PC10 -0.088
PC11  0.067
```

```
PC12  0.130
PC13 -0.252
PC14 -0.119
PC15 -0.033
PC16  0.192
PC17  0.018
PC18  0.116
PC19 -0.109
PC20  0.036
PC21  0.046
PC22 -0.034
PC23  0.055
PC24 -0.062
PC25 -0.081
PC26  0.010
PC27 -0.004
PC28 -0.025
PC29 -0.060
PC30 -0.027
PC31 -0.016
PC32  0.002
PC33 -0.001
PC34 -0.002

[34 rows x 34 columns]
```

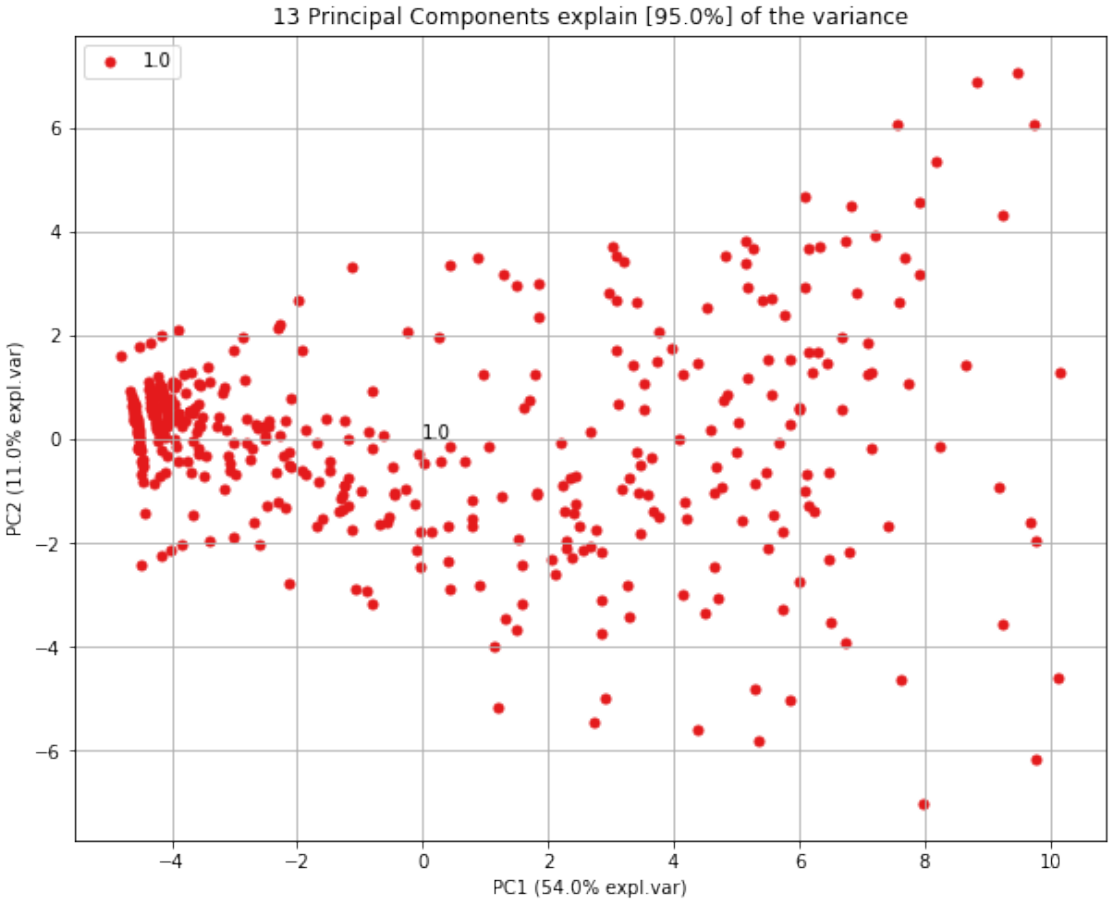
[149]:
```
pca2.plot()
plt.savefig('CumulativeVar (Second Wave).png', dpi = 300)
```



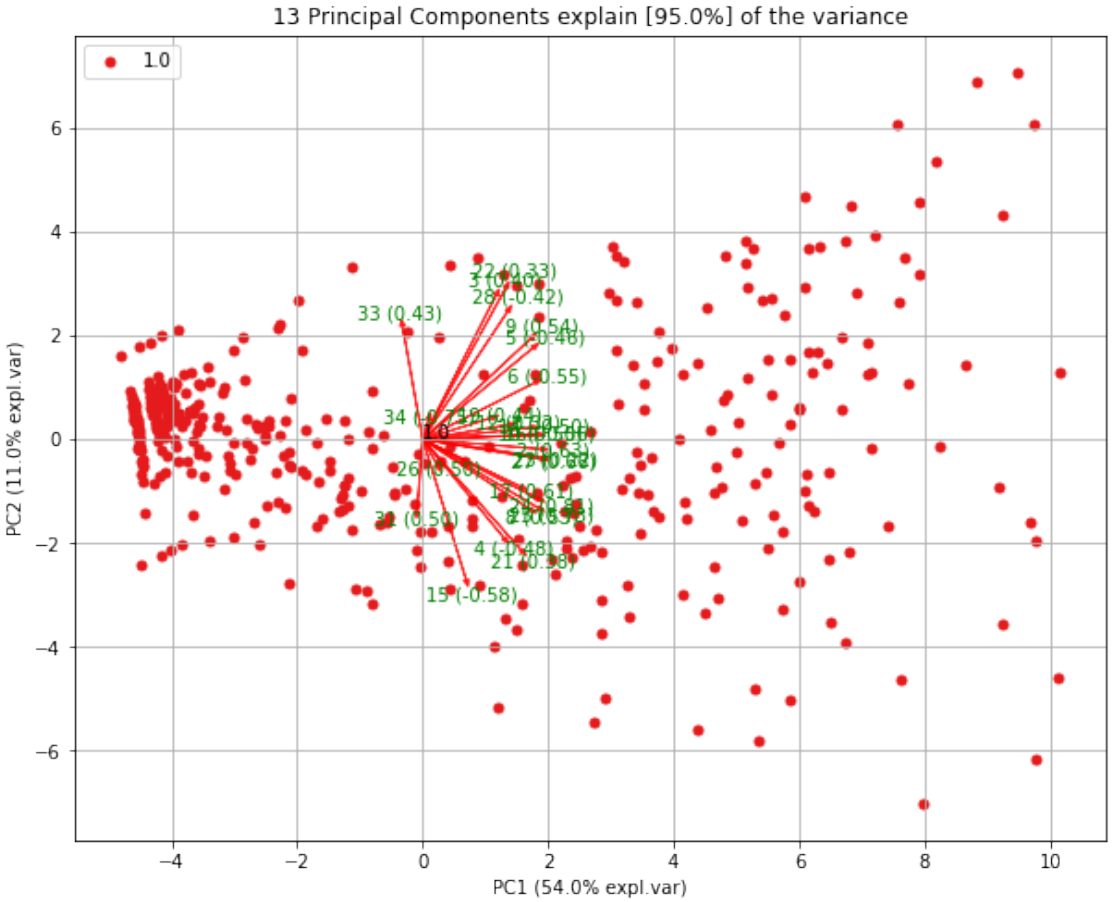

```
<Figure size 432x288 with 0 Axes>
```

[41]:
```
pca2.scatter()
```

[41]:
```
(<Figure size 720x576 with 1 Axes>,
 <AxesSubplot:title={'center':'13 Principal Components explain [95.0%] of the
variance'}, xlabel='PC1 (54.0% expl.var)', ylabel='PC2 (11.0% expl.var)'>)
```



[42]:
```
pca2.biplot()
```



[42]:
```
(<Figure size 720x576 with 1 Axes>,
 <AxesSubplot:title={'center':'13 Principal Components explain [95.0%] of the
variance'}, xlabel='PC1 (54.0% expl.var)', ylabel='PC2 (11.0% expl.var)'>)
```

[43]:
```
pca_transform_two = pca2.transform(transform_two)
pca_transform_two
```

```
[pca] >Column labels are auto-completed.
[pca] >Row labels are auto-completed.
```

[43]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	\
1.0	-4.465615	-0.479673	1.727604	0.909328	-0.580360	-0.777211	-0.038866	
1.0	-4.066622	-0.327372	1.624965	0.364906	-0.660415	-0.564859	0.163685	
1.0	-2.999341	-0.676840	1.239417	-0.345442	-1.002823	-0.071887	0.179834	
1.0	-0.687889	-1.642158	0.817860	-0.658948	-1.704584	0.987245	-0.534296	
1.0	2.297525	-1.938118	1.385482	0.679495	-2.748818	2.763935	-2.385940	
..	…	…	…	…	…	…	…	
1.0	-4.002388	1.046091	-0.590665	0.228896	0.739283	0.489239	-0.459534	
1.0	-3.160035	1.022456	-0.926332	-0.560759	0.590190	0.628599	-0.131755	

```
1.0 -1.249000  0.360488 -1.764548 -2.082792  0.291478  0.905976  0.332040
1.0  0.956402  1.245553 -1.967754 -2.833930  0.530159  0.171877  0.742480
1.0  1.846982  2.997161 -0.951849 -2.771204  0.437652  0.311179  0.895055

          PC8       PC9      PC10  …      PC25      PC26      PC27  \
1.0  0.569504  0.569833  0.274403  …  0.030639  0.075650  0.086654
1.0  0.203689  0.235255  0.047496  … -0.028060  0.061575 -0.005563
1.0  0.158361 -0.118949  0.187478  …  0.118234 -0.120218  0.111715
1.0  0.018831 -0.317256  1.150437  …  0.078425  0.101644 -0.309212
1.0 -0.002695 -0.600377  2.872270  …  0.092858  0.025628 -0.210302
..         …       …       …  …       …         …         …
1.0 -0.389852 -0.080792 -0.404283  … -0.072930  0.001772 -0.073452
1.0 -0.585455 -0.503057 -0.181880  …  0.238935 -0.312480  0.125784
1.0 -0.533909 -0.772014  0.288934  …  0.367326 -0.492034  0.141082
1.0 -1.137337 -0.417177  0.984648  …  0.268607 -0.678449  0.181773
1.0 -0.242297 -0.863992  0.016239  … -0.554236 -0.642043  0.995269

          PC28      PC29      PC30      PC31      PC32      PC33      PC34
1.0  0.019905 -0.054130  0.088359 -0.007338 -0.006911  0.001831 -0.006049
1.0  0.014020 -0.036840  0.020867 -0.113686  0.145458  0.018249 -0.008320
1.0  0.055112 -0.215895 -0.194080 -0.280222  0.012949 -0.004675 -0.000534
1.0 -0.128381 -0.396429 -0.242730 -0.157164 -0.073119 -0.003853 -0.005061
1.0  0.194376 -0.103900  0.273902 -0.098940 -0.001964 -0.001842  0.002718
..         …       …       …       …       …       …       …
1.0 -0.059066  0.360824  0.193763  0.003307 -0.039279  0.002049  0.001516
1.0  0.019502 -0.110694  0.044013 -0.048707 -0.031063 -0.014133  0.012934
1.0 -0.295792  0.262602  0.912461 -0.272782 -0.042512  0.082527  0.008617
1.0  0.317761  1.039918 -0.032898 -0.092785 -0.050557  0.041301  0.000898
1.0  0.234518  0.068097  0.018090  0.096087  0.038608  0.048120  0.002503

[416 rows x 34 columns]
```

[44]:
```
final_one = pca_transform_one.iloc[:,:13]
final_two = pca_transform_two.iloc[:,:13]
```

[45]:
```
final_one.to_csv('PeakOnePCA-Top13.csv')
final_two.to_csv('PeakTwoPCA-Top13.csv')
```

1.3 K-Means Clustering

[46]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import numpy as np
import plotly
import plotly.graph_objects as go
```

```
from plotly.subplots import make_subplots
import seaborn as sns
```

[47]:
```
peak_one_pca = final_one
peak_two_pca = final_two
```

[48]:
```
peak_one_pca.head()
```

[48]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	\
1.0	-3.292124	1.517491	-1.457907	1.845980	1.076299	-0.563544	-0.272162	
1.0	-2.939893	1.249270	-0.767865	1.825723	0.624506	-0.533552	-0.340818	
1.0	-2.297535	1.214677	-0.044416	1.688675	0.195339	-0.652316	-0.529986	
1.0	-0.975263	1.411792	0.541269	1.287771	-0.561943	-1.171058	-0.785404	
1.0	1.966947	1.621573	-0.972956	0.357727	-2.369732	-2.087097	-0.550462	

	PC8	PC9	PC10	PC11	PC12	PC13
1.0	-0.099489	-0.077273	-0.301191	-0.053856	0.050206	-0.265575
1.0	-0.081871	-0.332738	0.050688	-0.000013	0.031194	-0.193634
1.0	-0.080562	-0.392753	0.322466	0.186709	-0.001983	0.328689
1.0	0.209514	-0.208613	0.197257	0.250275	-0.181944	1.314325
1.0	-2.677389	-0.715190	0.199660	1.603276	2.561807	-0.926886

[49]:
```
peak_two_pca.head()
```

[49]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	\
1.0	-4.465615	-0.479673	1.727604	0.909328	-0.580360	-0.777211	-0.038866	
1.0	-4.066622	-0.327372	1.624965	0.364906	-0.660415	-0.564859	0.163685	
1.0	-2.999341	-0.676840	1.239417	-0.345442	-1.002823	-0.071887	0.179834	
1.0	-0.687889	-1.642158	0.817860	-0.658948	-1.704584	0.987245	-0.534296	
1.0	2.297525	-1.938118	1.385482	0.679495	-2.748818	2.763935	-2.385940	

	PC8	PC9	PC10	PC11	PC12	PC13
1.0	0.569504	0.569833	0.274403	-0.325902	-0.160779	0.338461
1.0	0.203689	0.235255	0.047496	-0.027572	-0.082015	0.242718
1.0	0.158361	-0.118949	0.187478	-0.252371	-0.388349	-0.074527
1.0	0.018831	-0.317256	1.150437	-0.008968	-0.301541	-0.076990
1.0	-0.002695	-0.600377	2.872270	2.205336	0.226075	0.072092

[50]:
```
peak_one_data = peak_one_pca.values
peak_two_data = peak_two_pca.values
```

[51]:
```
from sklearn.cluster import KMeans
```

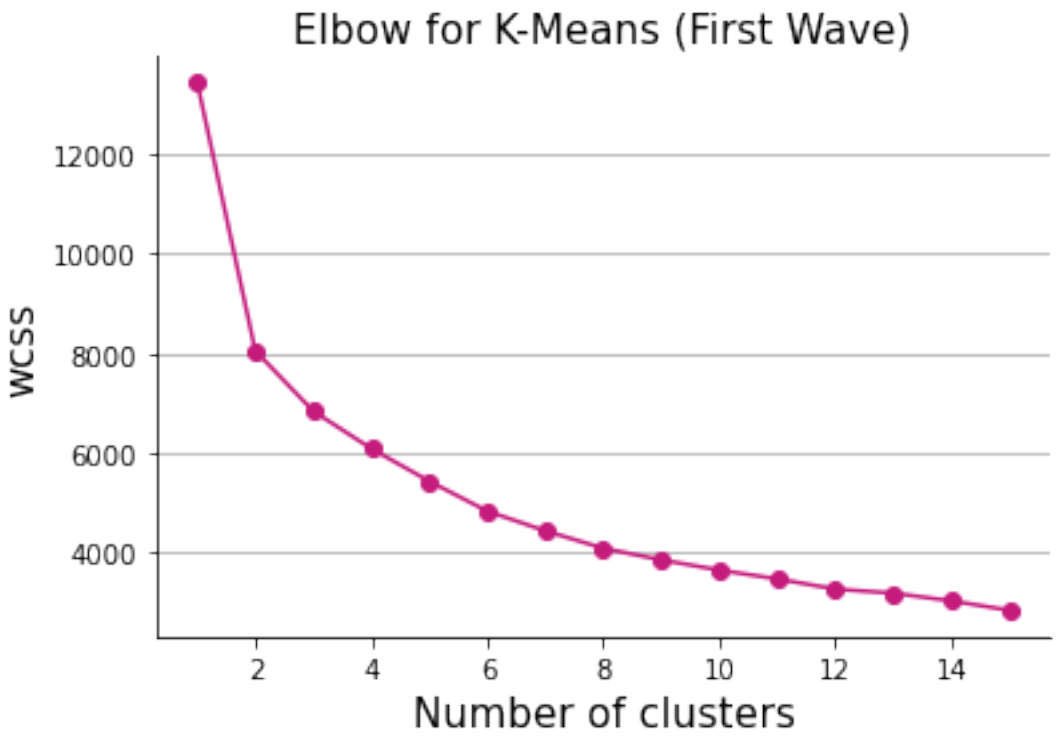
[219]:
```
def elbow_method(data_cl, title, pngname):
    # determine the number of clustersu
    #The Elbow method looks at how the total WSS(within cluster sum of squares)␣
 ↪varies with the number of clusters.
```

```
    wcss = []
    for i in range(1,16):
        km = KMeans(n_clusters=i,init='k-means++', max_iter=300, n_init=10,␣
↪random_state=0)
        km.fit(data_cl)
        wcss.append(km.inertia_)
    plt.plot(range(1,16),wcss, c="#c51b7d", marker = 'o')
    plt.gca().spines["top"].set_visible(False)
    plt.gca().spines["right"].set_visible(False)
    plt.title(title, size=15)
    plt.xlabel('Number of clusters', size=15)
    plt.grid(axis='y')
    plt.ylabel('wcss', size=15)
    plt.savefig(pngname, dpi = 300)
    plt.show()
```

[220]:
```
def train_KMeans(num_clusters, data_cl, title, pngname):
    # number of clusters
    km = KMeans(n_clusters=num_clusters)
    km.fit(data_cl)

    labels = km.predict(data_cl)
    #print(clusters)
    plt.scatter(data_cl[:, 0], data_cl[:, 1], c=labels,
            s=50, cmap='viridis')
    plt.xlabel("PCA 1")
    plt.ylabel("PCA 2")
    plt.title(title)
    plt.grid()
    plt.savefig(pngname, dpi = 300)
    return km, labels
```

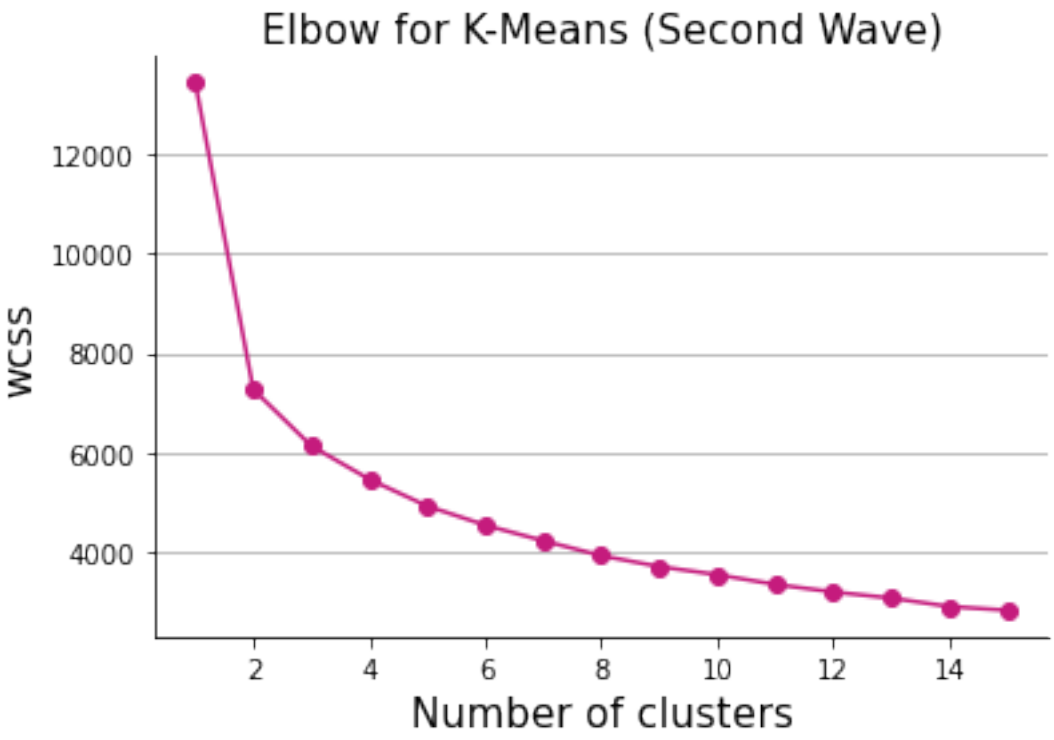
[221]:
```
# Peak One
elbow_method(peak_one_data,"Elbow for K-Means (First Wave)", "ElbowOne.png")
# optimal = 4
```

```
C:\Users\williamshih\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=2.
  warnings.warn(
```



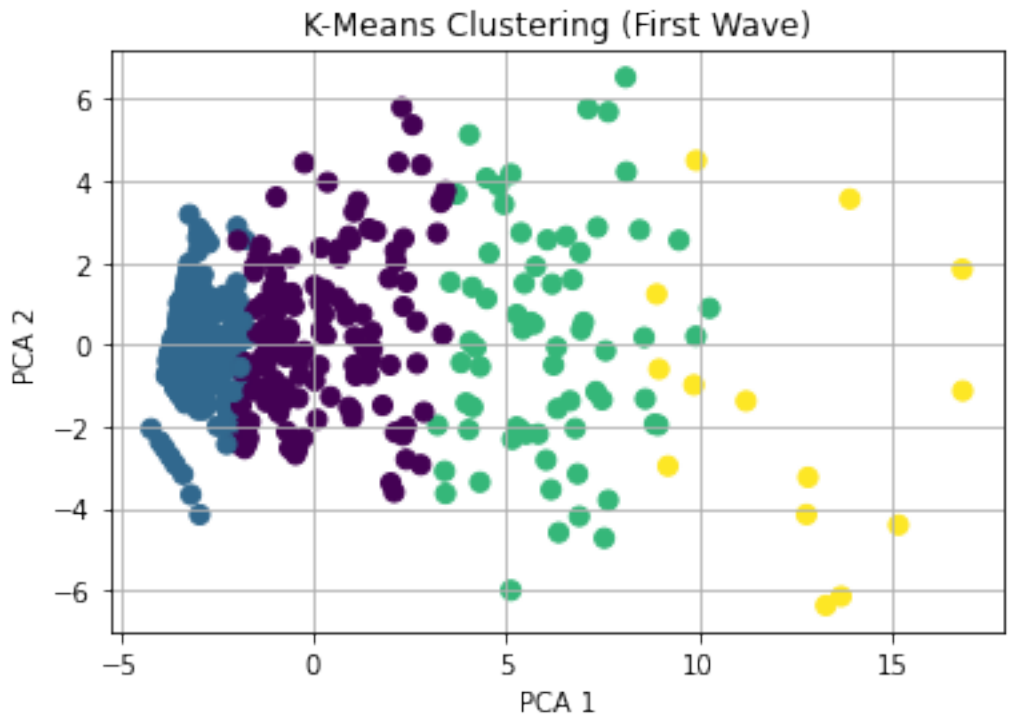
[222]:
```
elbow_method(peak_two_data,"Elbow for K-Means (Second Wave)","ElbowTwo.png")
# optimal = 4
```

```
C:\Users\williamshih\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=2.
  warnings.warn(
```

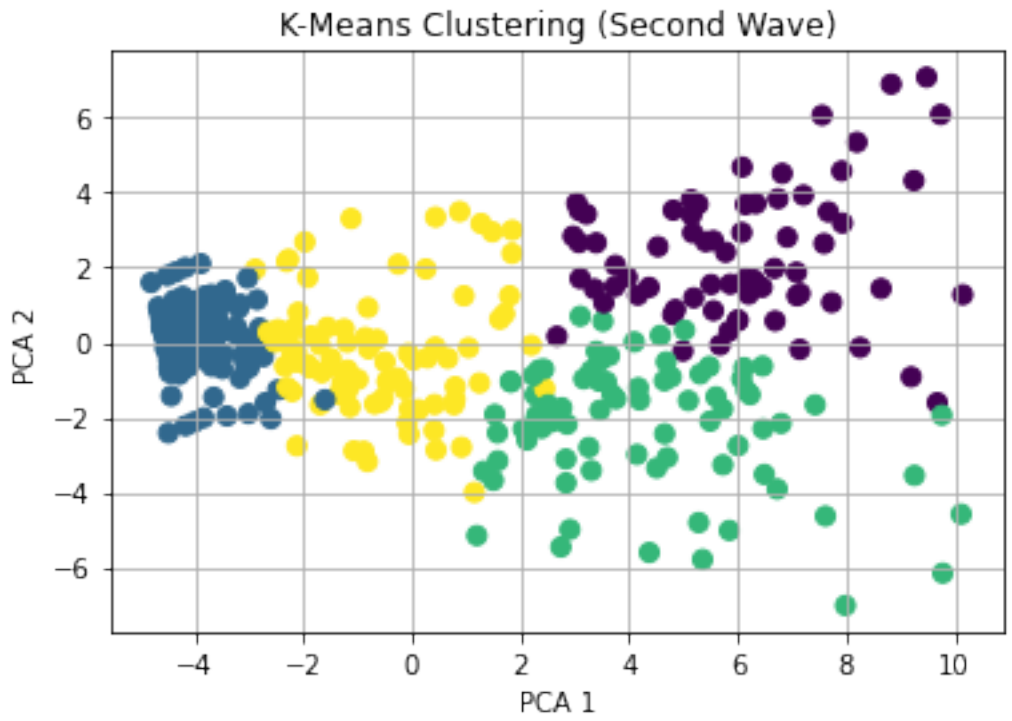


[56]:
```
opt1 = 4
opt2 = 4
```

[57]:
```
# peak one kmeans clustering
km_one, labels_one = train_KMeans(opt1, peak_one_data, "K-Means Clustering␣
↪(First Wave)","KmeansOne.png")
# 3 clusters seem to be overlapping, one cluster deviates from other clusters␣
↪but with few data points
```



[58]:
```
# peak two kmeans clustering
km_two, labels_two = train_KMeans(opt2, peak_two_data, "K-Means Clustering␣
↪(Second Wave)", "KmeansTwo.png")
# peak two clusters are more distributed evenly
```



[59]:
```
#sihouette scores for KMeans
from sklearn.metrics import silhouette_score
def calculate_silhouette_score(X, km):
    score = silhouette_score(X, km.labels_, metric='euclidean')
    print('Silhouetter Score: %.3f' % score)
```

1.3.1 Analyze data based on cluster

[60]:
```
# get centroids and corresponding index in the original dataset
def group_data_clusters(data, centroids):
    cluster_dict = {}
    for i in range(len(centroids)):
        lab = centroids[i]
        if lab not in cluster_dict:
            cluster_dict[lab] = []
        tmp = cluster_dict[lab]
        tmp.append(data[i])
        cluster_dict[lab] = tmp
    return cluster_dict
```

[61]:
```
df_one = OriginalPeakOne
```

[62]:
```
df_two = OriginalPeakTwo
```

[63]:
```
df_cond_one = PeakOneNoPCAScale
df_cond_two = PeakTwoNoPCAScale
```

[64]:
```
general_cols = ['State', 'Age Group', 'Tables Death Rate', 'Population']
cond_cols = ['Adult respiratory distress syndrome',
       'All other conditions and causes (residual)', 'Alzheimer disease',
       'Cardiac arrest', 'Cardiac arrhythmia', 'Cerebrovascular diseases',
       'Chronic lower respiratory diseases', 'Diabetes', 'Heart failure',
       'Hypertensive diseases', 'Influenza and pneumonia',
       'Intentional and unintentional injury, poisoning, and other adverse␣
 ↪events',
       'Ischemic heart disease', 'Malignant neoplasms', 'Obesity',
       'Other diseases of the circulatory system',
       'Other diseases of the respiratory system', 'Renal failure',
       'Respiratory arrest', 'Respiratory failure', 'Sepsis',
       'Vascular and unspecified dementia', 'COVID-19 Deaths',
       'Pneumonia Deaths', 'Pneumonia and COVID-19 Deaths', 'Influenza Deaths',
       'Pneumonia, Influenza, or COVID-19 Deaths']
```

[65]:
```
df_one_general = df_one[general_cols]
df_one_condition_death = df_cond_one[cond_cols]
```

[66]:
```
df_two_general = df_two[general_cols]
df_two_condition_death = df_cond_two[cond_cols]
```

[67]:
```
data_one_general = df_one_general.values
data_two_general = df_two_general.values
```

[68]:
```
data_one_cond = df_one_condition_death.values
data_two_cond = df_two_condition_death.values
```

[69]:
```
cluster_dict_one_general = group_data_clusters(data_one_general , labels_one)
cluster_dict_one_cond = group_data_clusters(data_one_cond  , labels_one)
```

[70]:
```
cluster_dict_two_general = group_data_clusters(data_two_general , labels_two)
cluster_dict_two_cond = group_data_clusters(data_two_cond  , labels_two)
```

[71]:
```
def array_to_list(cluster_dict, i):
    cluster_data = [l.tolist() for l in cluster_dict[i]]
    return cluster_data
```

[72]:
```
def get_different_clusters_data(cluster_dict,num_lst):
    cluster_lst_dict = {}
    for num in num_lst:
        data_cl = array_to_list(cluster_dict, num)
        cluster_lst_dict[num] = data_cl
    return cluster_lst_dict
```

[73]:
```
# get original data and normalized death conditions based on different clusters␣
↪in Peak One
general_one_clusters_dict =␣
↪get_different_clusters_data(cluster_dict_one_general,[0, 1, 2, 3])
cond_one_clusters_dict = get_different_clusters_data(cluster_dict_one_cond,[0,␣
↪1, 2, 3])
```

[74]:
```
# get original data and normalized death conditions based on different clusters␣
↪in Peak Two
general_two_clusters_dict =␣
↪get_different_clusters_data(cluster_dict_two_general,[0, 1, 2, 3])
cond_two_clusters_dict = get_different_clusters_data(cluster_dict_two_cond,[0,␣
↪1, 2, 3])
```

Peak One Data Clusters Pattern

[75]:
```
df_gen1_cl1 = pd.DataFrame(general_one_clusters_dict[0], columns = general_cols)
df_gen1_cl2 = pd.DataFrame(general_one_clusters_dict[1], columns = general_cols)
df_gen1_cl3 = pd.DataFrame(general_one_clusters_dict[2], columns = general_cols)
df_gen1_cl4 = pd.DataFrame(general_one_clusters_dict[3], columns = general_cols)
```

[76]:
```
df_cond_cl1 = pd.DataFrame(cond_one_clusters_dict[0], columns = cond_cols)
df_cond_cl2 = pd.DataFrame(cond_one_clusters_dict[1], columns = cond_cols)
df_cond_cl3 = pd.DataFrame(cond_one_clusters_dict[2], columns = cond_cols)
df_cond_cl4 = pd.DataFrame(cond_one_clusters_dict[3], columns = cond_cols)
```

[77]:
```
def plot_top_groups(df, top_n, x_name, title):
    f1 = sns.catplot(x= x_name, kind="count", palette="ch:.25",␣
↪data=df[[x_name]], order = df[x_name].value_counts()[:top_n].index)
    f1.set_xticklabels(rotation=40)
    plt.title(title)
    plt.savefig(title +".png", dpi = 300)

def plot_all_clusters_group(df_lst, top_n, x_name, typ):
    # fig, ax = plt.subplots(2,2, figsize=(12,10))
    title = "Top " + str(top_n) + " " + x_name + "s for "
    typ = " (" + typ + ")"
    plot_top_groups(df_lst[0], top_n, x_name, title + "Cluster One" + typ)
    plot_top_groups(df_lst[1], top_n, x_name, title + "Cluster Two" + typ)
    plot_top_groups(df_lst[2], top_n, x_name, title + "Cluster Three" + typ)
    plot_top_groups(df_lst[3], top_n, x_name, title + "Cluster Four" + typ)
```

[193]:
```
def violin_plot(col_name1, col_name2, data, title):
    ax = sns.violinplot(x=col_name1, y=col_name2, data = data)
    plt.title(title)
    plt.savefig(title + ".png", dpi = 300)
    plt.show()
```

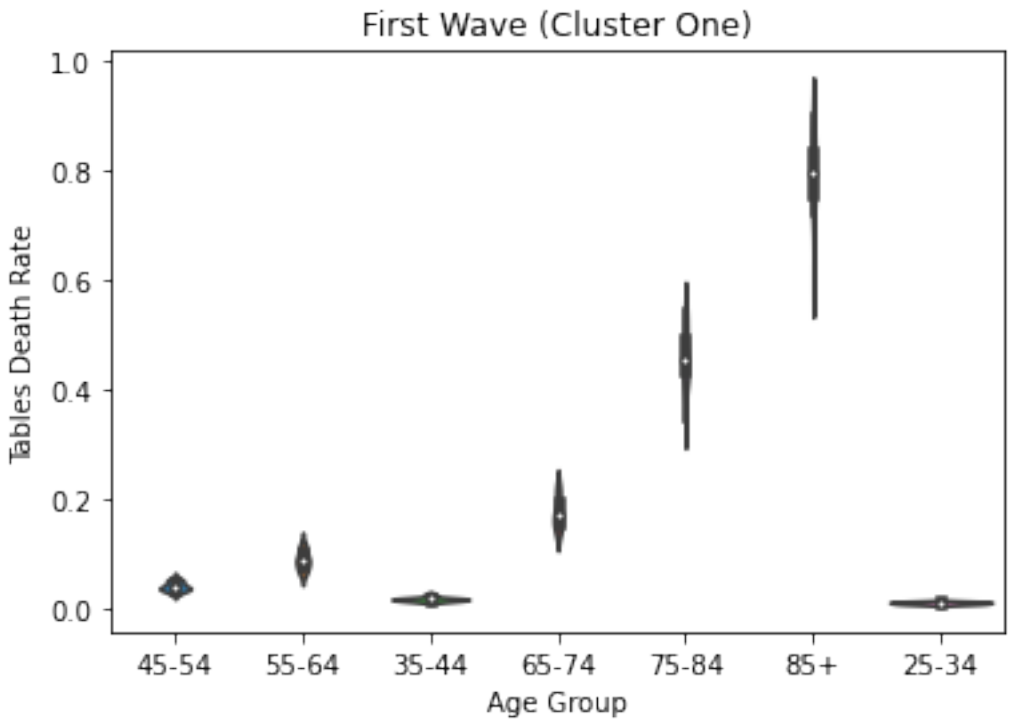
```
def plot_death_rate_by_group(lst_df, col_name1, col_name2, title):
    # fig = plt.figure()
    violin_plot(col_name1,col_name2, lst_df[0], title + " (Cluster One)")
    violin_plot(col_name1,col_name2, lst_df[1], title + " (Cluster Two)")
    violin_plot(col_name1,col_name2, lst_df[2], title + " (Cluster Three)")
    violin_plot(col_name1,col_name2, lst_df[3], title + " (Cluster Four)")
```

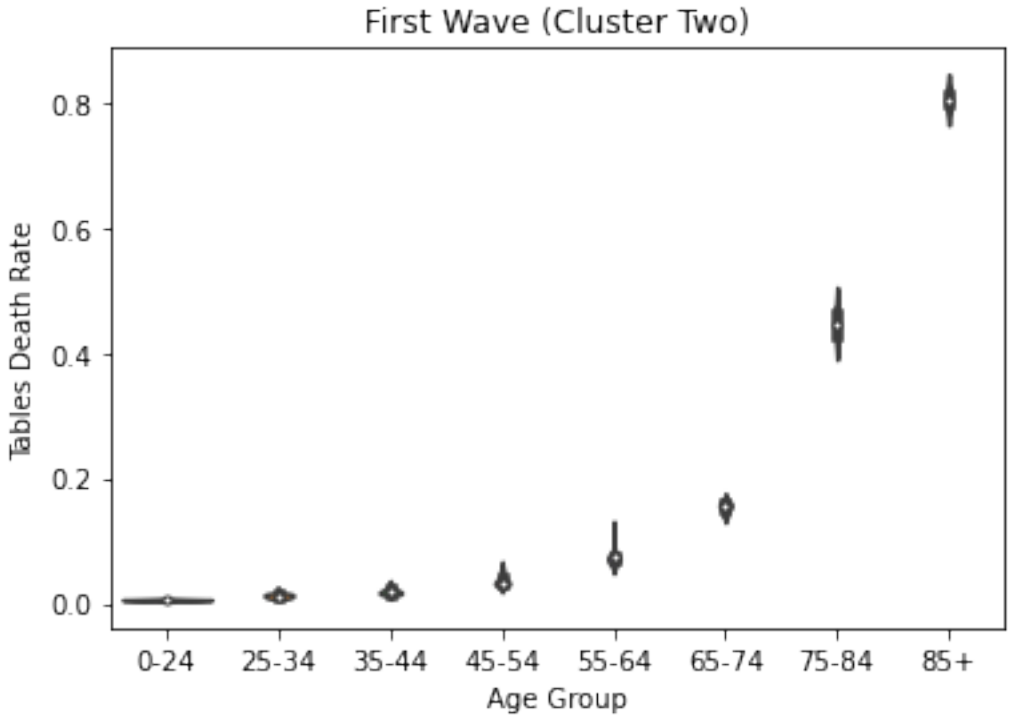
[79]:
```
def get_top_death_rate_cond(df_cond_cl, top_n):
    # select top n columns based on average death rate for each column
    df1 = df_cond_cl.describe()
    df_t = df1.T
    df_top = df_t.nlargest(top_n, 'mean')
    df_res = df_top.T
    cols = [col for col in df_res.columns]
    df_final = df_cond_cl[cols]
    return df_final
```

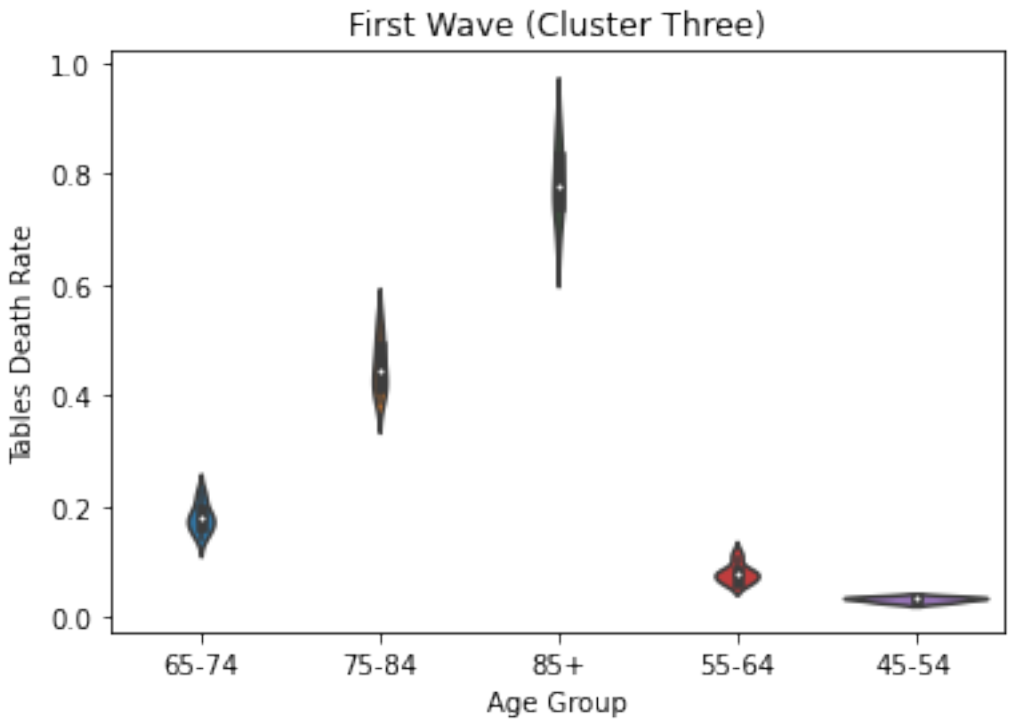
[194]:
```
def hist_top_cond(df, title):
    for col in df.columns:
        hist = df[col].hist(bins=10)
        print("Plotting for column {}".format(col))
        plt.title(title + " " + col)
        plt.savefig(title + " " + col + ".png", dpi = 300)
        plt.show()

def plot_death_rate_cond(lst_df, top_n, title):
    print("Cluster1:")
    top_data1 = get_top_death_rate_cond(lst_df[0], top_n)
    hist_top_cond(top_data1, title + " (Cluster 1)")
    print("Cluster2:")
    top_data2 = get_top_death_rate_cond(lst_df[1], top_n)
    hist_top_cond(top_data2, title + " (Cluster 2)")
    print("Cluster3:")
    top_data3 = get_top_death_rate_cond(lst_df[2], top_n)
    hist_top_cond(top_data3, title + " (Cluster 3)")
    print("Cluster4:")
    top_data4 = get_top_death_rate_cond(lst_df[3], top_n)
    hist_top_cond(top_data4, title + " (Cluster 4)")
```

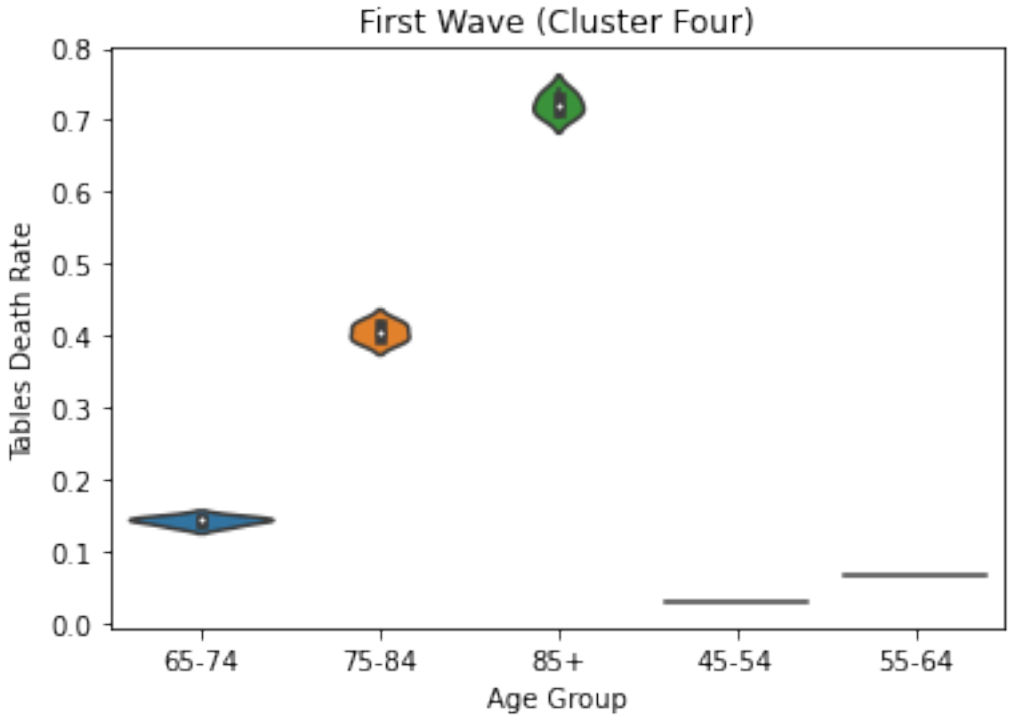
[195]:
```
x_name0 = "State"
df__gen1_cl_lst = [df_gen1_cl1,df_gen1_cl2, df_gen1_cl3, df_gen1_cl4 ]
```

[196]:
```
# death rate by age groups
plot_death_rate_by_group(df__gen1_cl_lst, "Age Group", 'Tables Death Rate',␣
 ↪"First Wave")
```


First Wave (Cluster One)
Tables Death Rate
1.0
0.8
0.6
0.4
0.2
0.0
45-54
55-64
35-44
65-74
75-84
85+
25-34
Age Group


First Wave (Cluster Two)
Tables Death Rate
0.8
0.6
0.4
0.2
0.0
0-24
25-34
35-44
45-54
55-64
65-74
75-84
85+
Age Group


First Wave (Cluster Three)
Tables Death Rate
1.0
0.8
0.6
0.4
0.2
0.0
65-74
75-84
85+
55-64
45-54
Age Group


First Wave (Cluster Four)
Tables Death Rate
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
65-74
75-84
85+
45-54
55-64
Age Group

[197]:
```
df_gen1_cl1
```

[197]:

	State	Age Group	Tables	Death Rate	Population
0	Alabama	45-54		0.051329	5537511
1	Alabama	55-64		0.111217	5917887
2	Arizona	35-44		0.017627	8031015
3	Arizona	45-54		0.036836	7662006
4	Arkansas	55-64		0.109074	3473631
..	…	…		…	…
125	West Virginia	75-84		0.541764	985968
126	West Virginia	85+		0.887698	361224
127	Wisconsin	65-74		0.156507	5352732
128	Wisconsin	75-84		0.434646	2647674
129	Wisconsin	85+		0.762085	1154781

[130 rows x 4 columns]

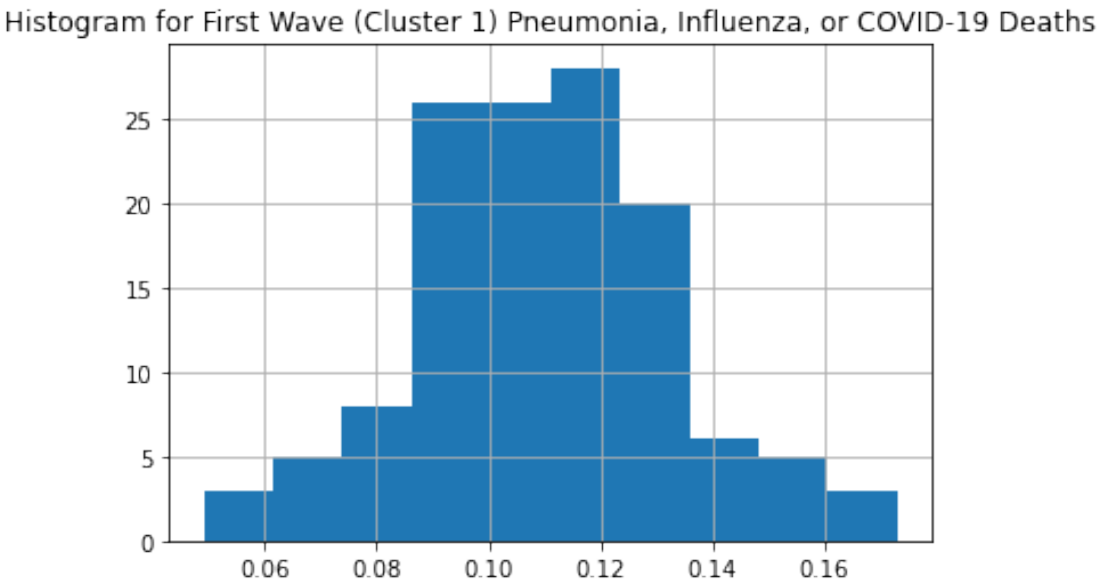
[198]:
```
df_cond_cl_lst = [df_cond_cl1, df_cond_cl2, df_cond_cl3, df_cond_cl4]
```

[199]:
```
df_cond_cl1.columns
```

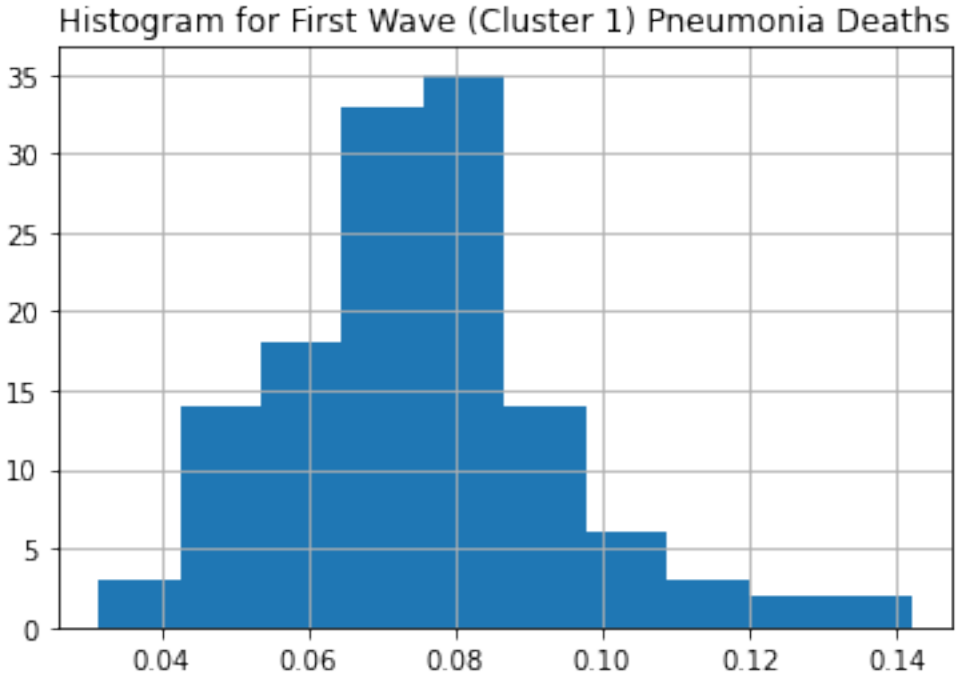
[199]:
```
Index(['Adult respiratory distress syndrome',
       'All other conditions and causes (residual)', 'Alzheimer disease',
       'Cardiac arrest', 'Cardiac arrhythmia', 'Cerebrovascular diseases',
       'Chronic lower respiratory diseases', 'Diabetes', 'Heart failure',
       'Hypertensive diseases', 'Influenza and pneumonia',
       'Intentional and unintentional injury, poisoning, and other adverse
events',
       'Ischemic heart disease', 'Malignant neoplasms', 'Obesity',
       'Other diseases of the circulatory system',
       'Other diseases of the respiratory system', 'Renal failure',
       'Respiratory arrest', 'Respiratory failure', 'Sepsis',
       'Vascular and unspecified dementia', 'COVID-19 Deaths',
       'Pneumonia Deaths', 'Pneumonia and COVID-19 Deaths', 'Influenza Deaths',
       'Pneumonia, Influenza, or COVID-19 Deaths'],
      dtype='object')
```

[200]:
```
# death condition with top 3 highest death rate in each cluster
plot_death_rate_cond(df_cond_cl_lst, 3, "Histogram for First Wave")
```

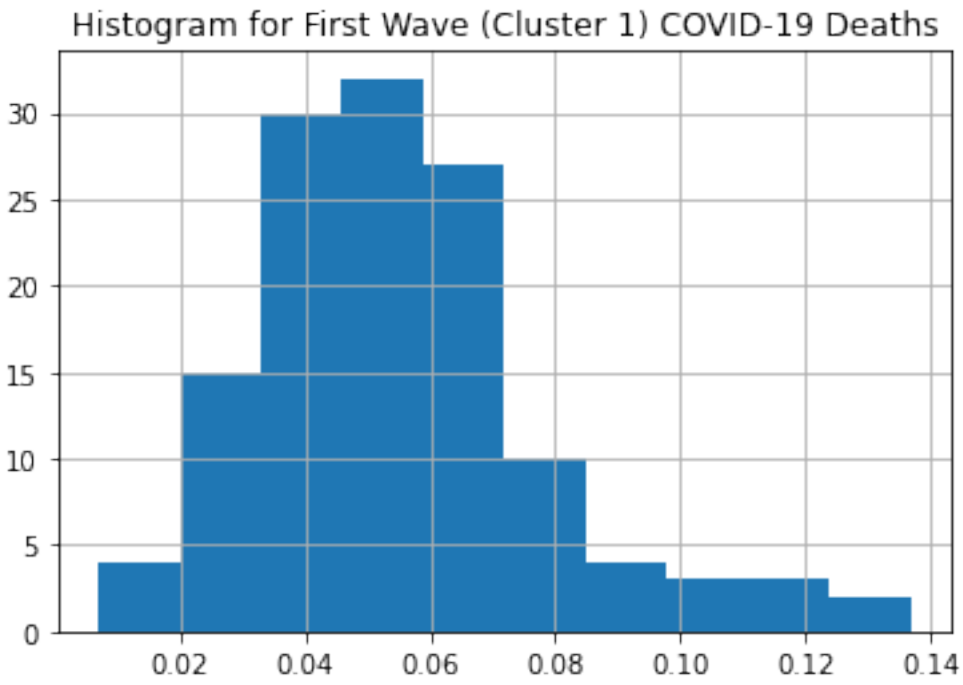
```
Cluster1:
Plotting for column Pneumonia, Influenza, or COVID-19 Deaths
```



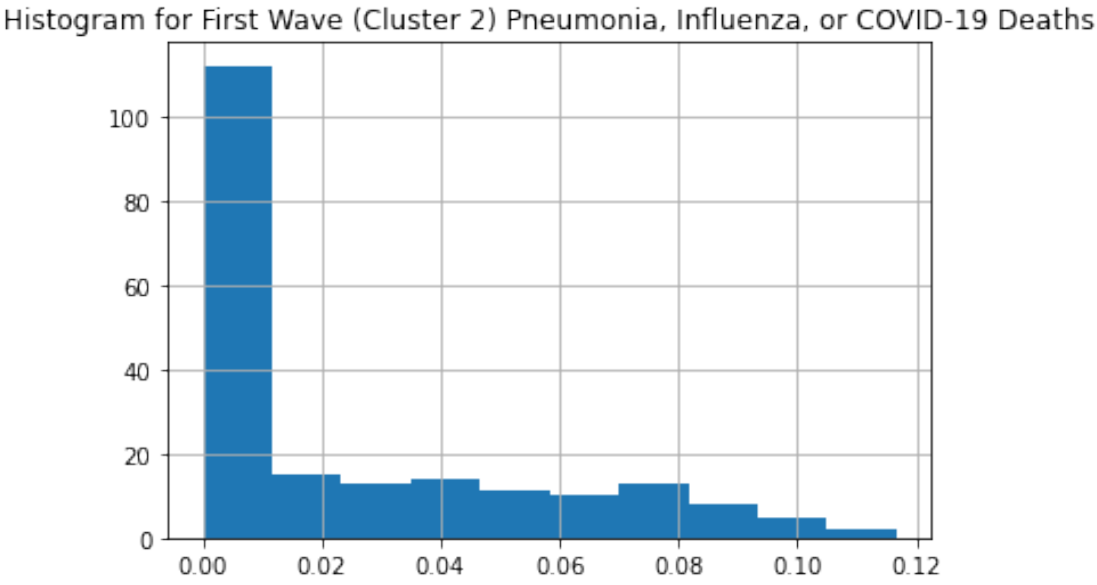
Plotting for column Pneumonia Deaths



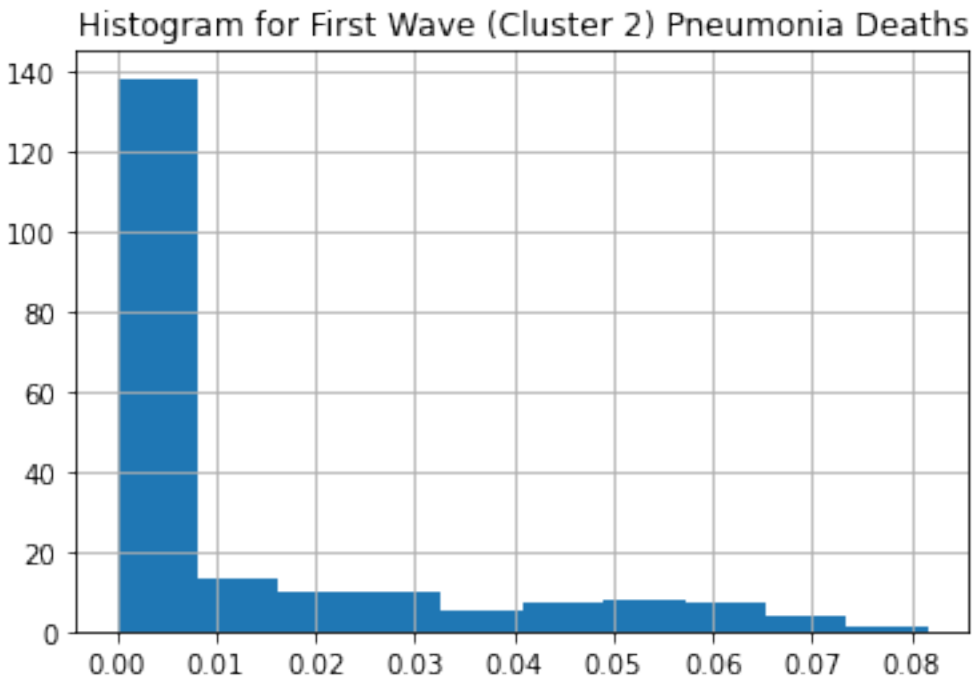
Plotting for column COVID-19 Deaths



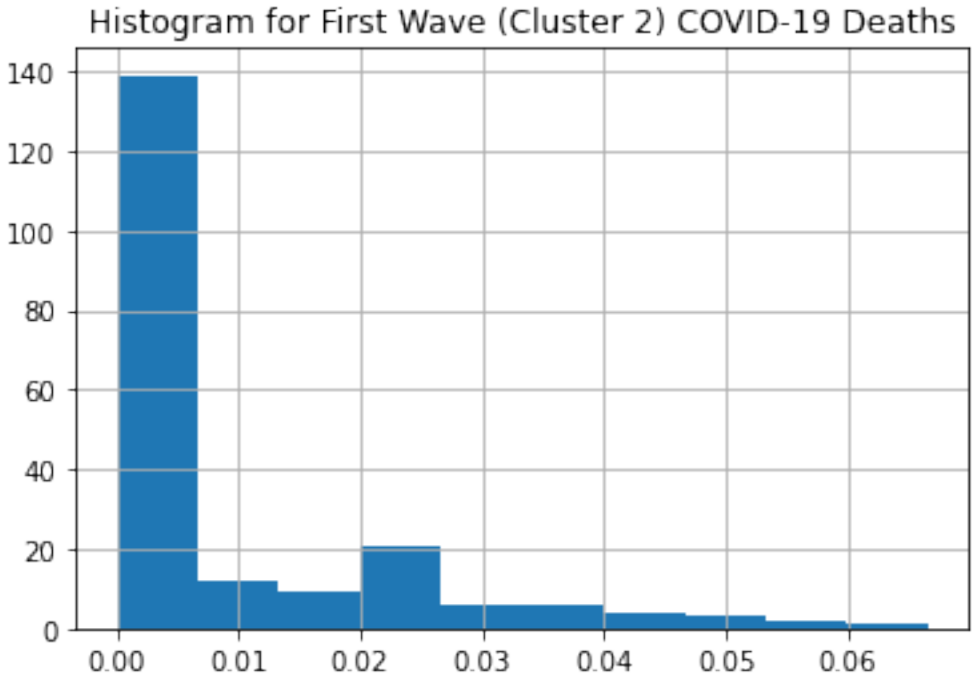
Cluster2:
Plotting for column Pneumonia, Influenza, or COVID-19 Deaths



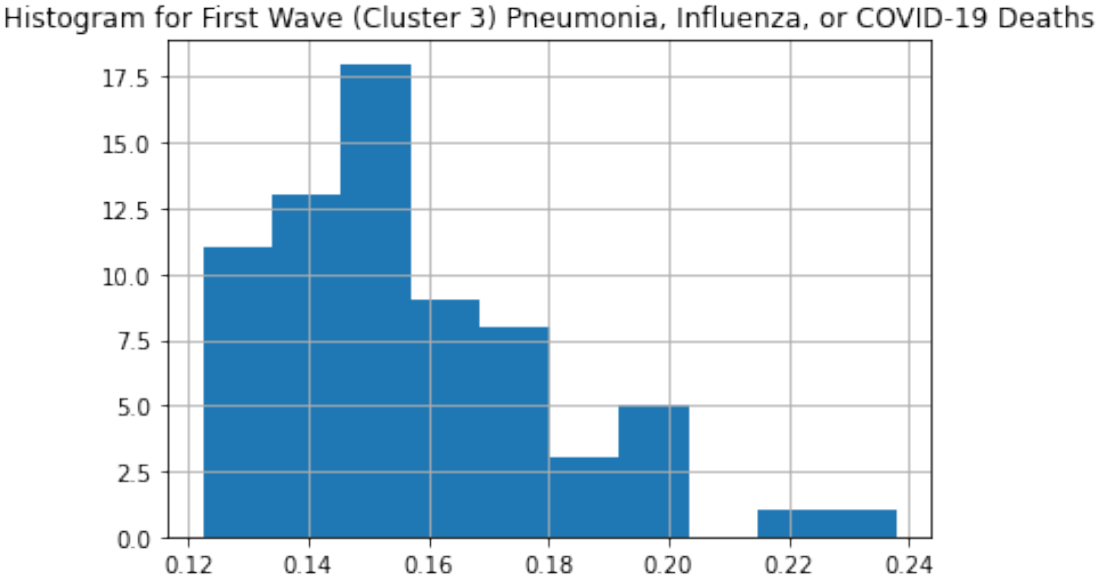
Plotting for column Pneumonia Deaths



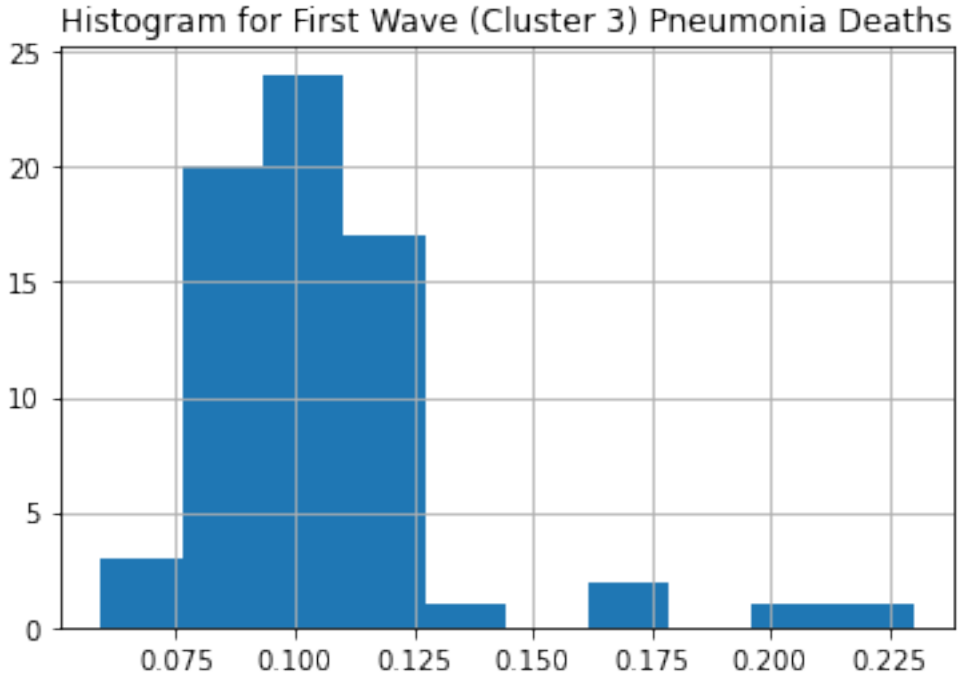

```
Plotting for column COVID-19 Deaths
```



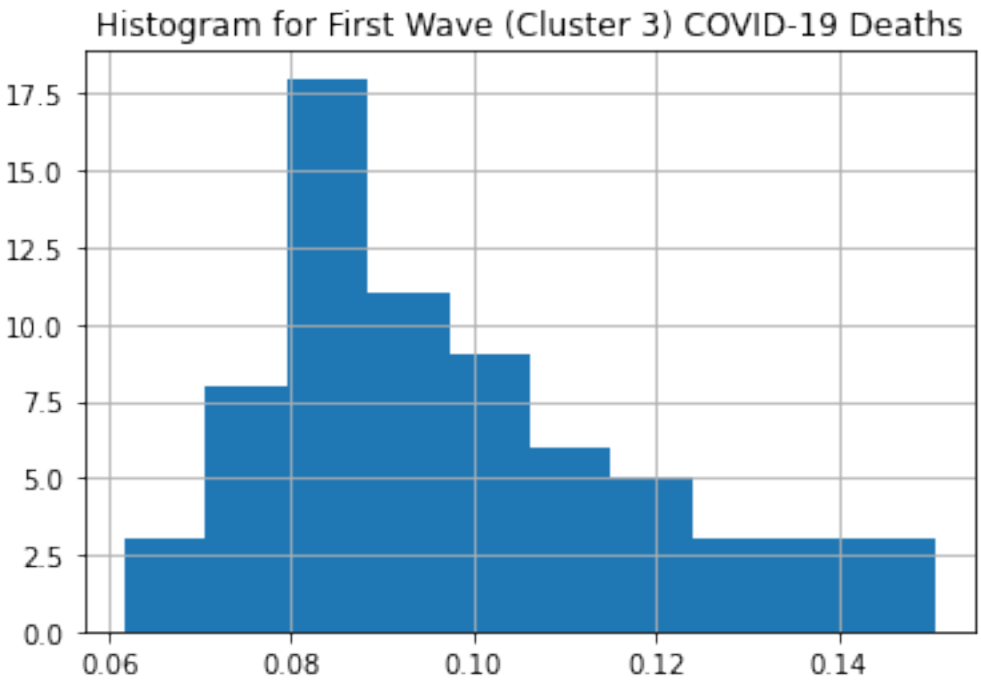
Cluster3:
Plotting for column Pneumonia, Influenza, or COVID-19 Deaths



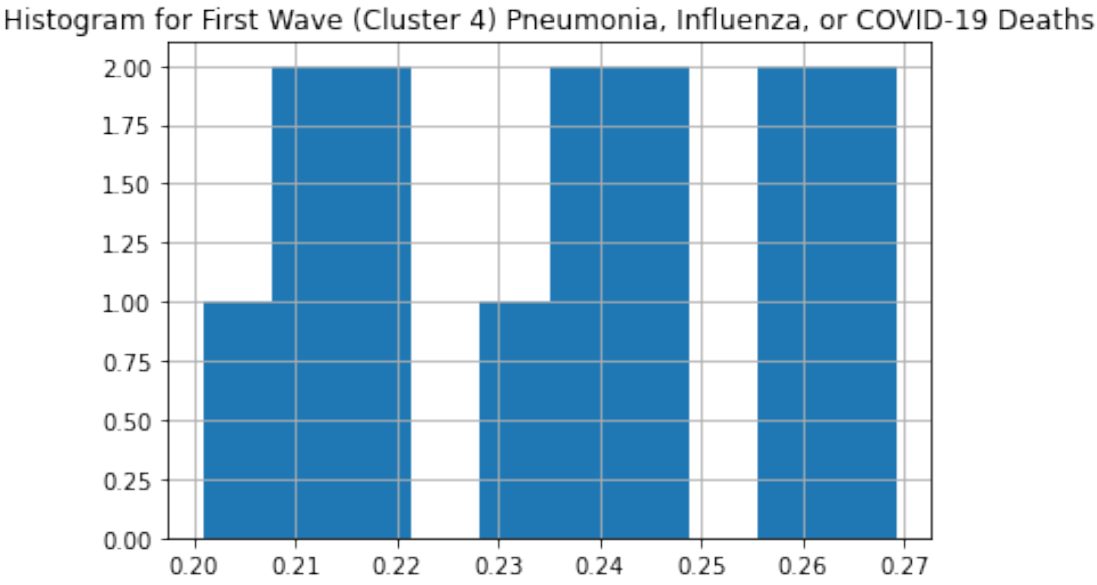
Plotting for column Pneumonia Deaths



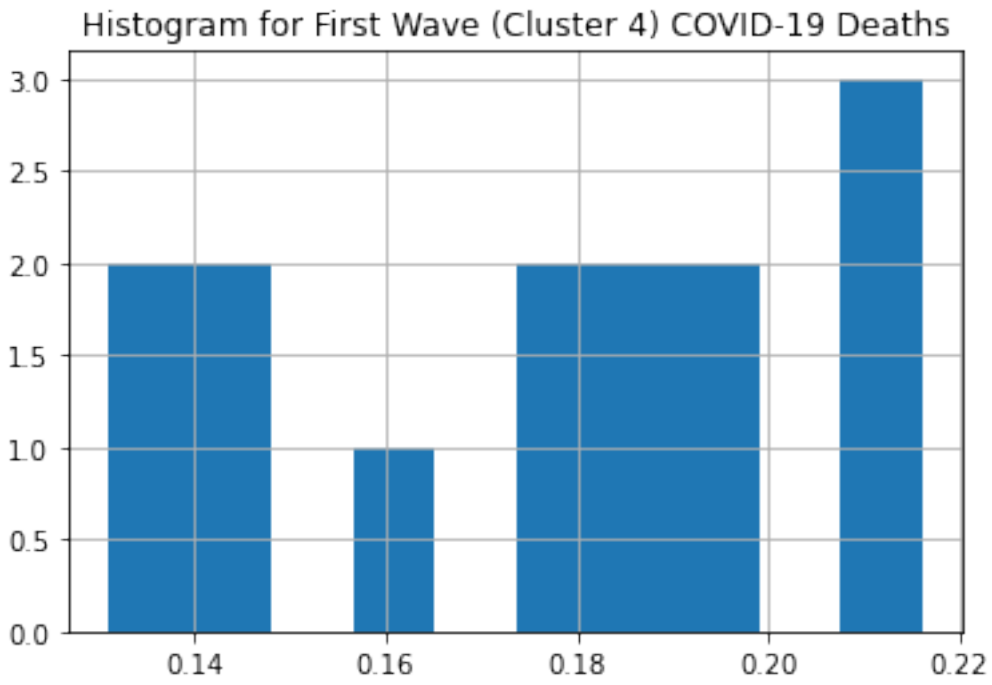
Plotting for column COVID-19 Deaths



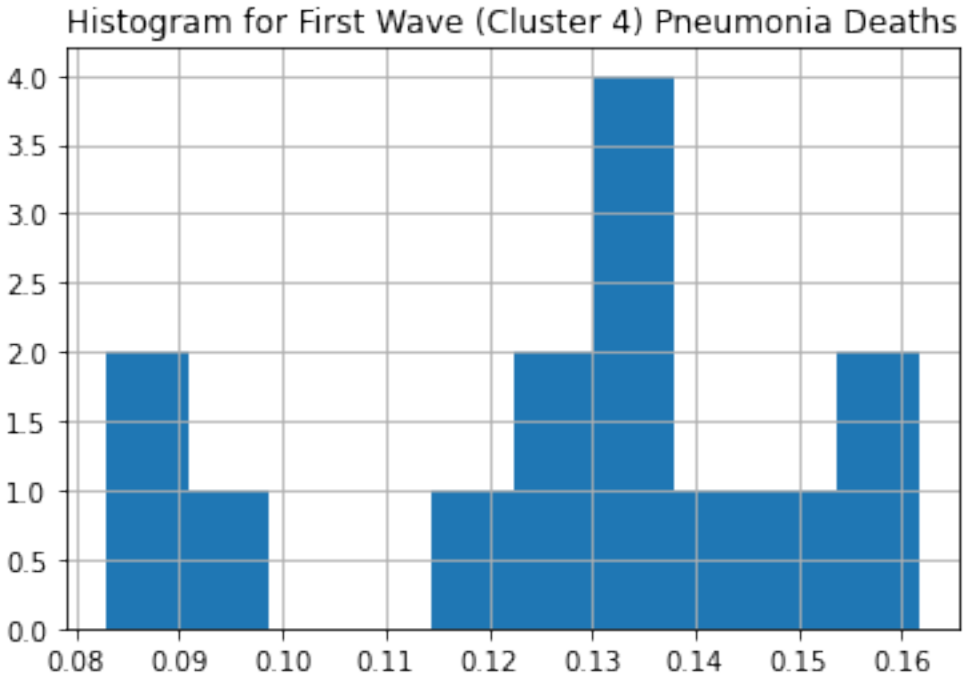
Cluster4:
Plotting for column Pneumonia, Influenza, or COVID-19 Deaths



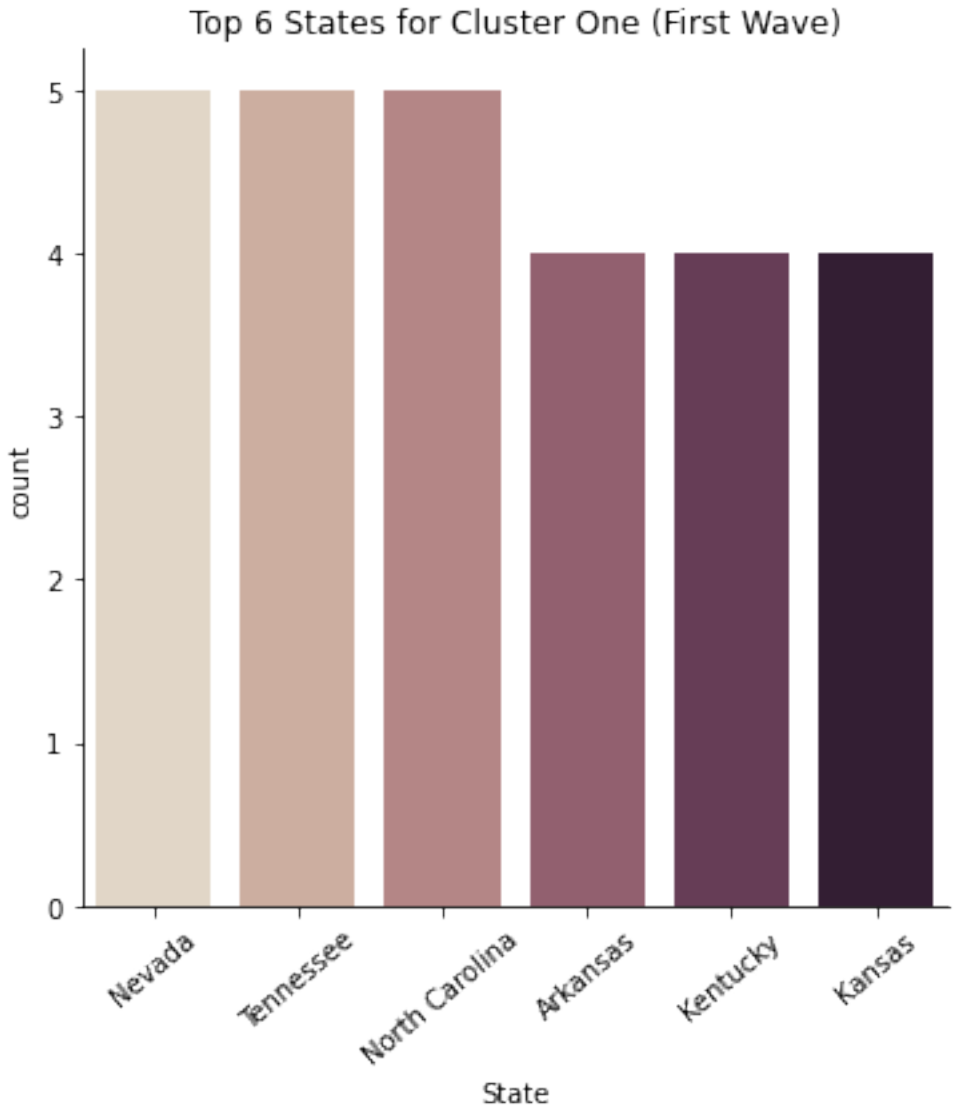
Plotting for column COVID-19 Deaths

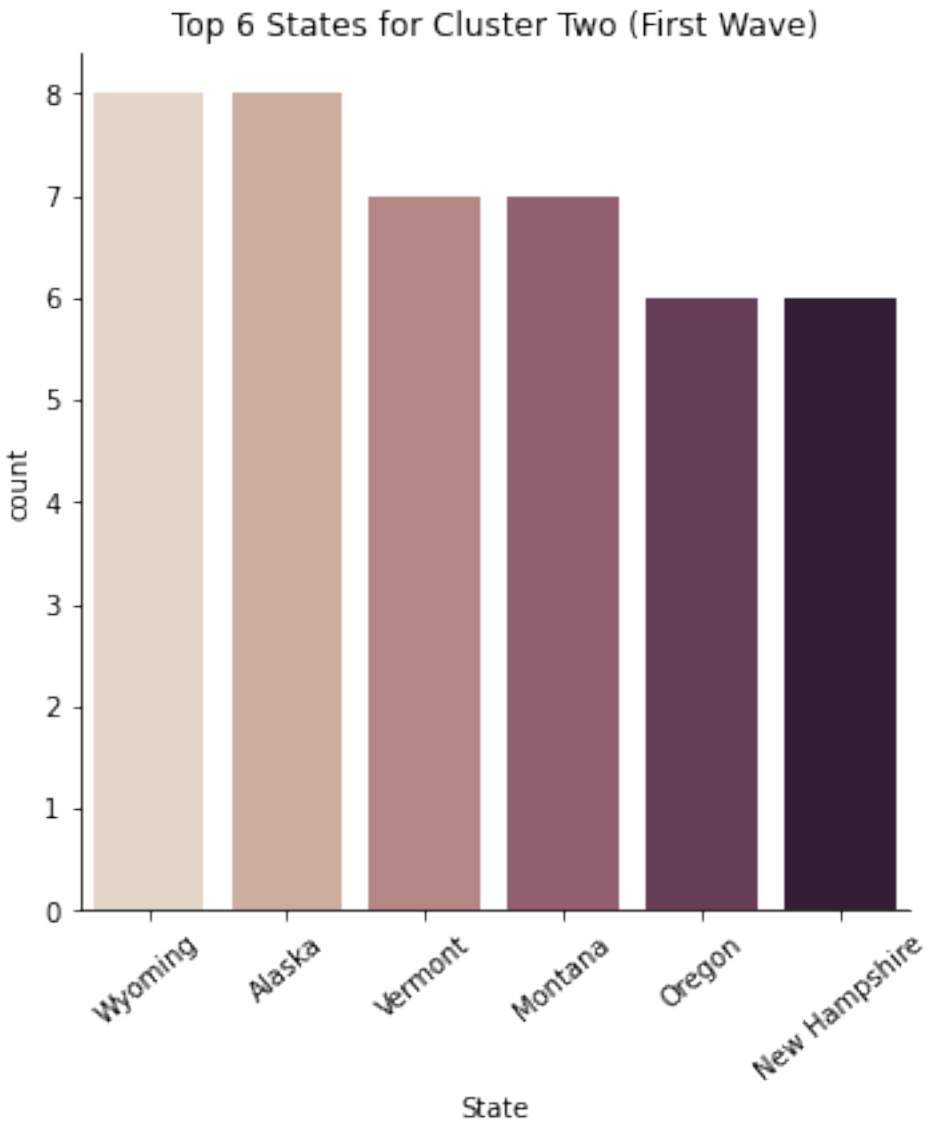


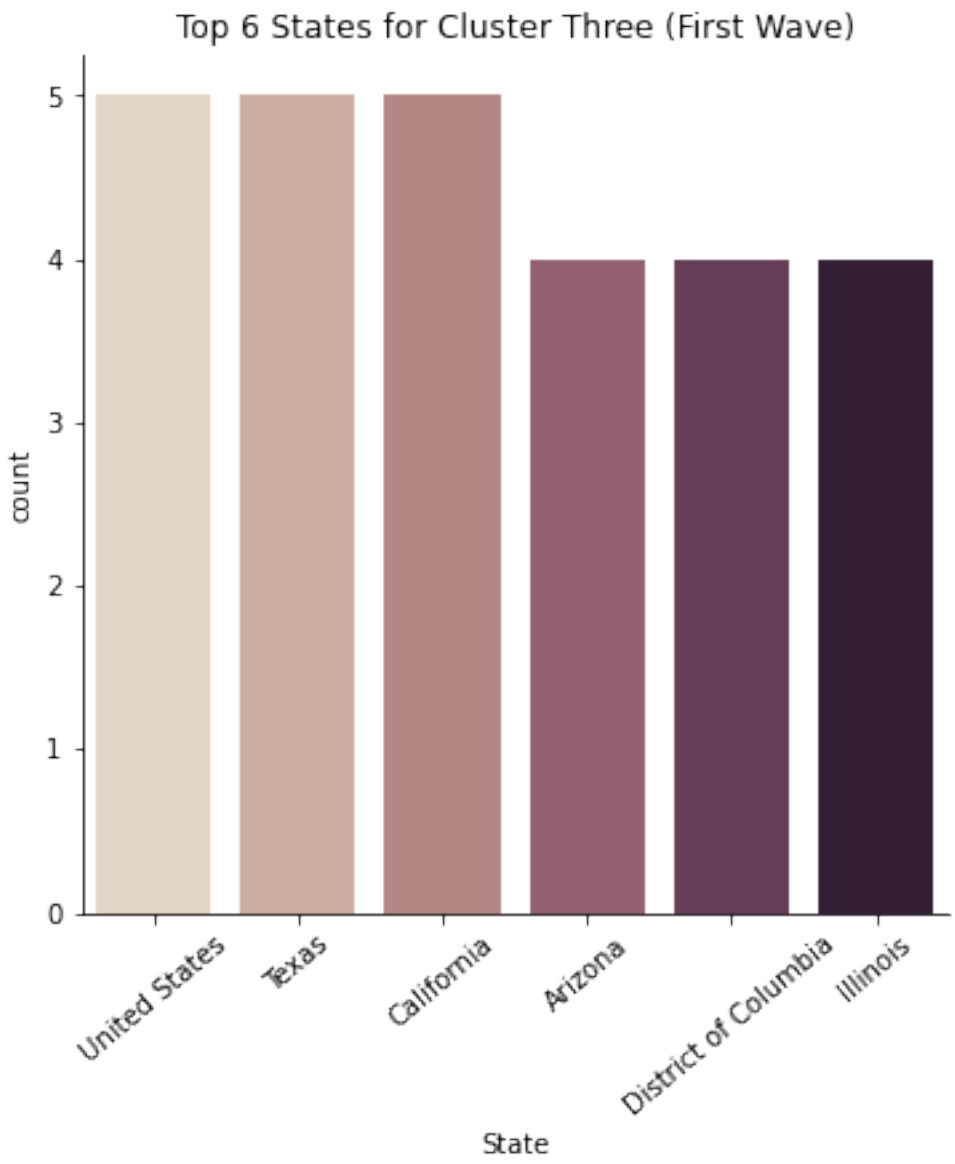
Plotting for column Pneumonia Deaths

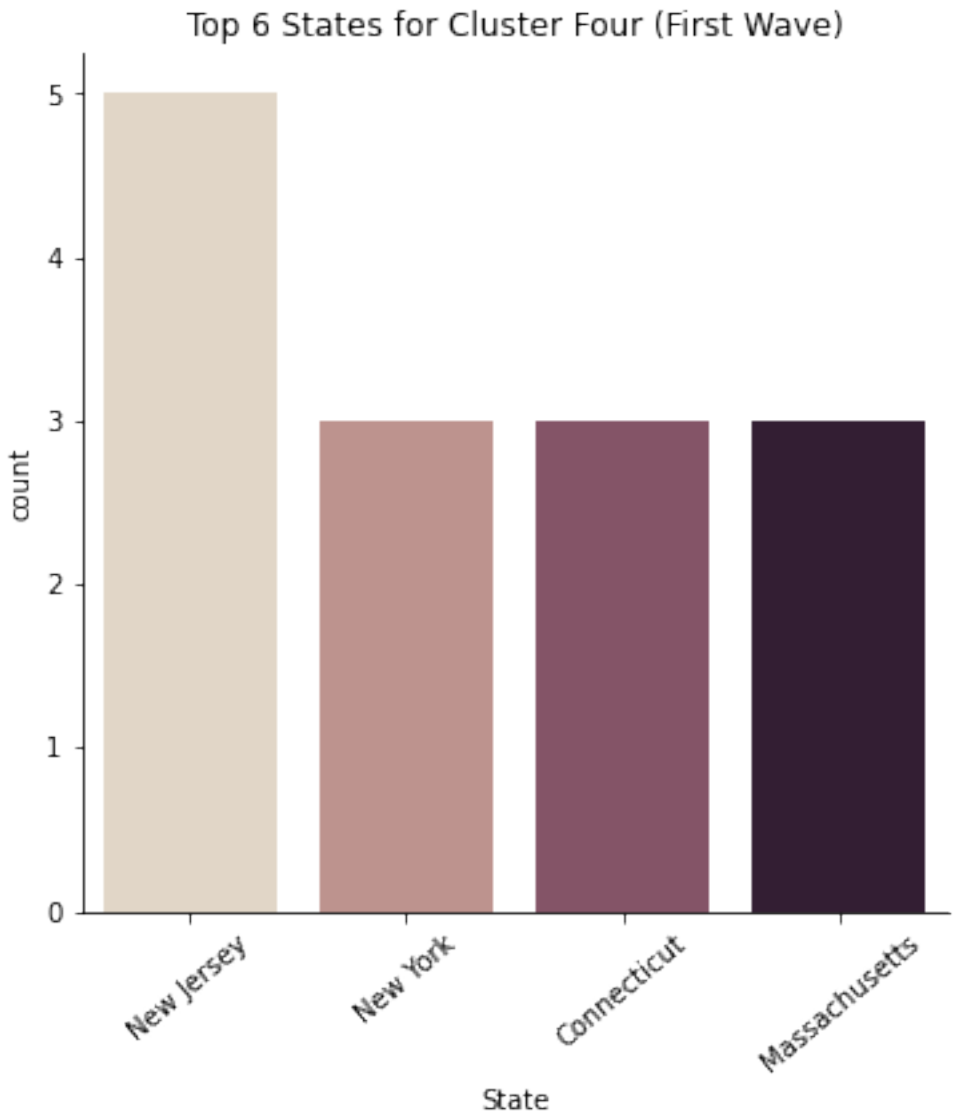


[201]:
```
# plot top 3 state groups
plot_all_clusters_group(df__gen1_cl_lst, 6, x_name0, "First Wave")
```

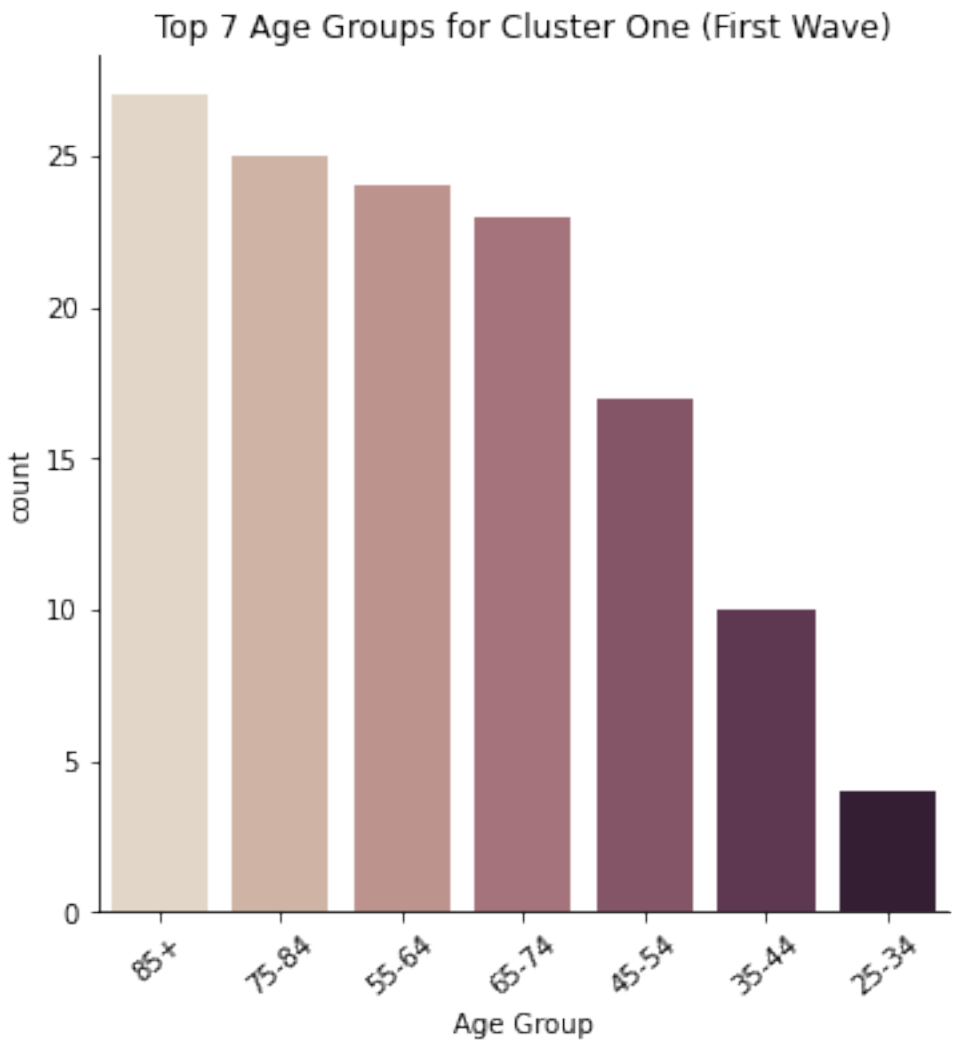


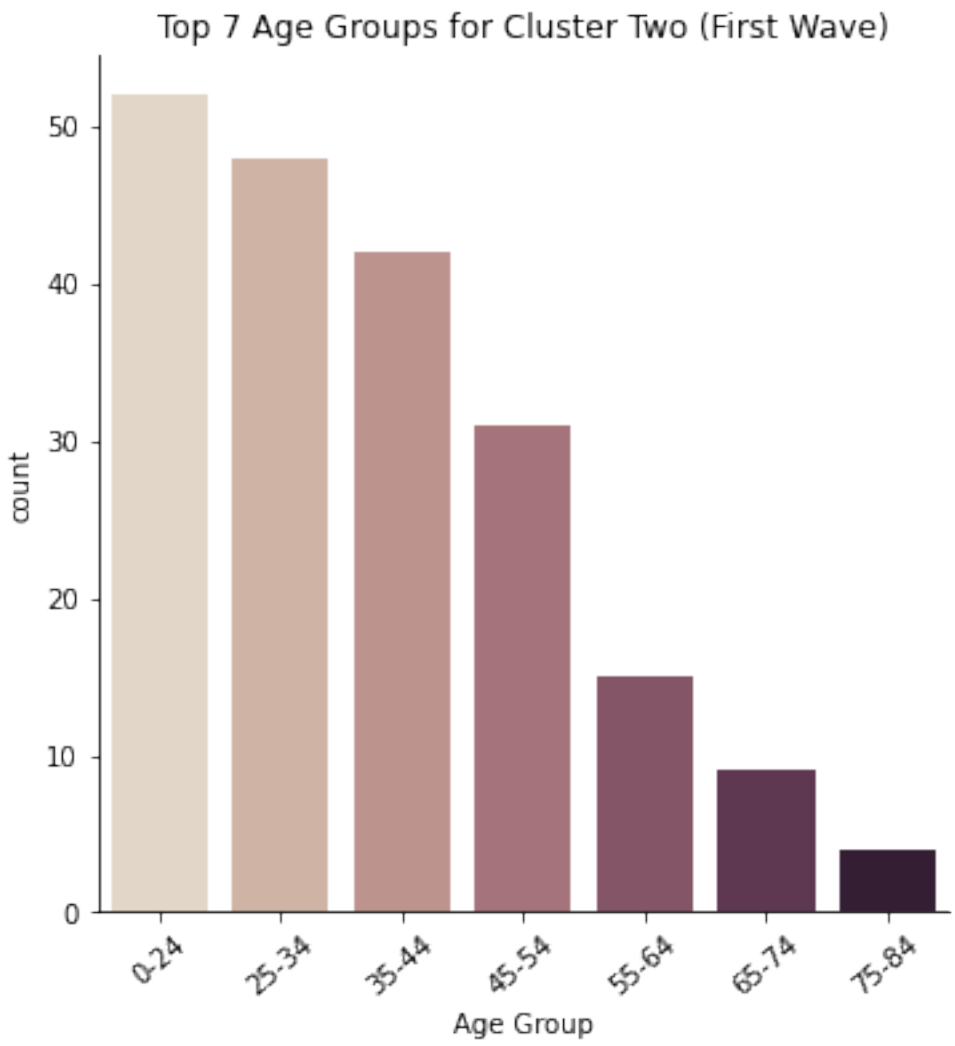

Top 6 States for Cluster Two (First Wave)
count
0
1
2
3
4
5
6
7
8
Wyoming
Alaska
Vermont
Montana
Oregon
New Hampshire
State

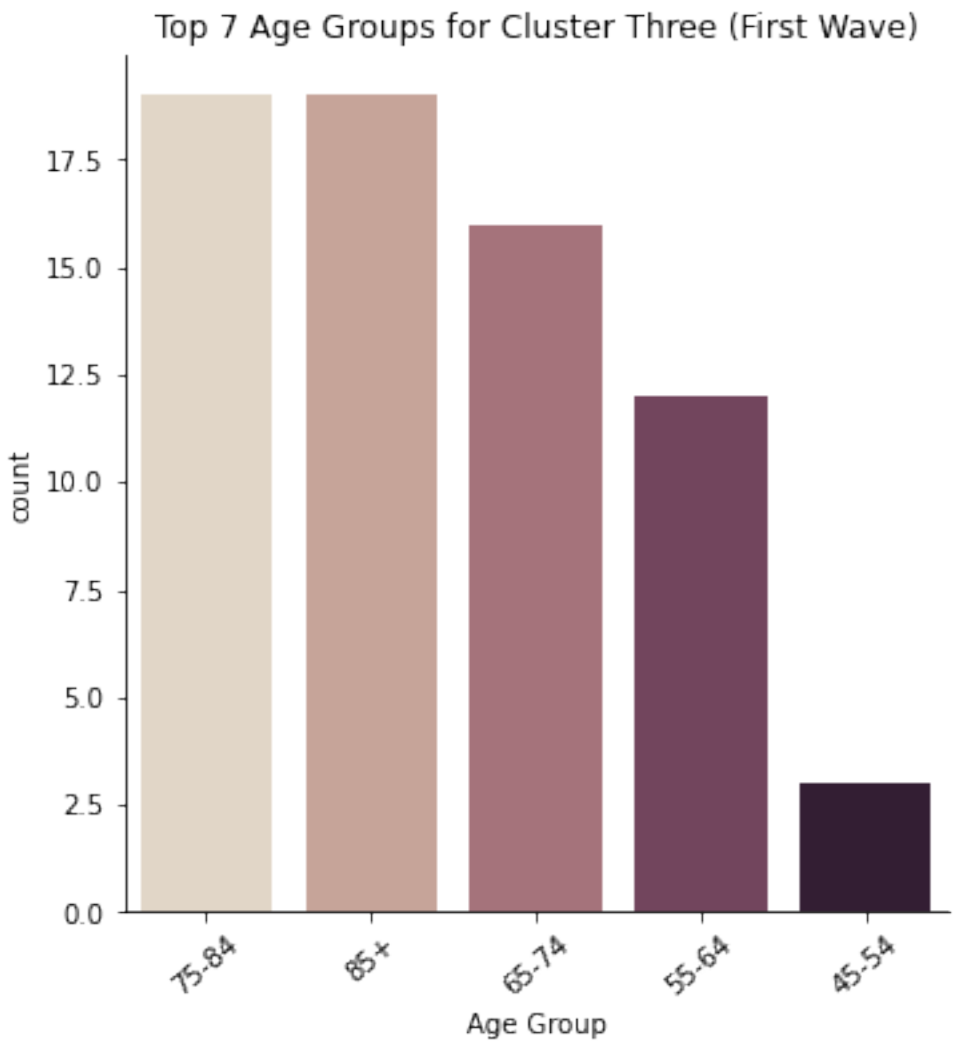

Top 6 States for Cluster Three (First Wave)
count
0
1
2
3
4
5
United States
Texas
California
Arizona
District of Columbia
Illinois
State

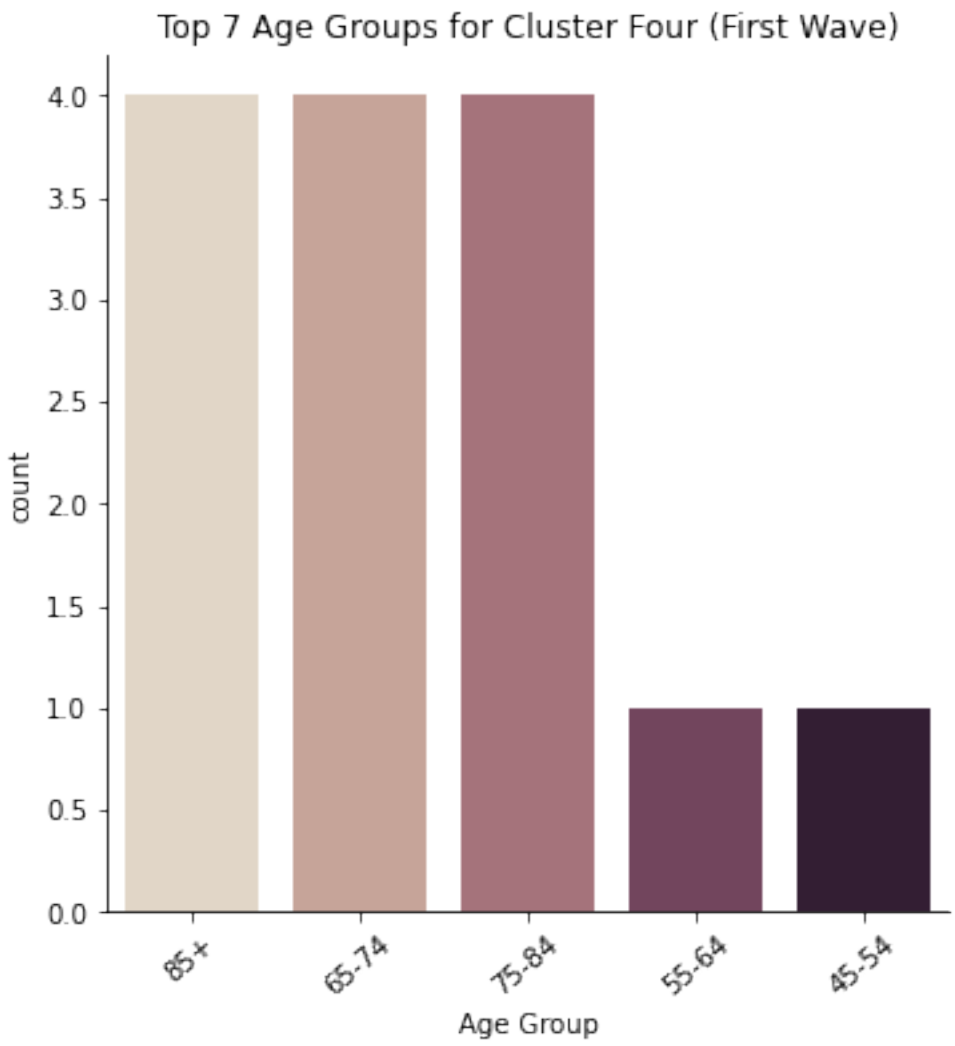


[202]:
```
# plot top age group
x_name1 = "Age Group"
df__gen1_cl_lst = [df_gen1_cl1,df_gen1_cl2, df_gen1_cl3, df_gen1_cl4 ]
plot_all_clusters_group(df__gen1_cl_lst, 7, x_name1, "First Wave")
```


Top 7 Age Groups for Cluster One (First Wave)
count
0
5
10
15
20
25
85+
75-84
55-64
65-74
45-54
35-44
25-34
Age Group


Top 7 Age Groups for Cluster Two (First Wave)
count
50
40
30
20
10
0
0-24
25-34
35-44
45-54
55-64
65-74
75-84
Age Group


Top 7 Age Groups for Cluster Three (First Wave)
count
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
75-84
85+
65-74
55-64
45-54
Age Group

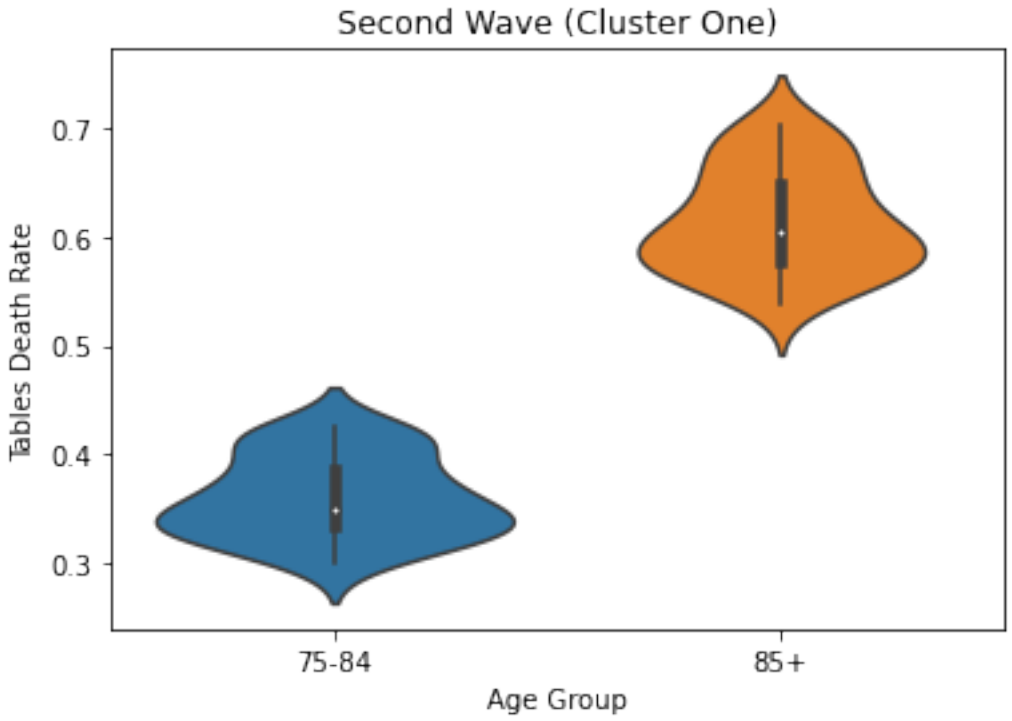


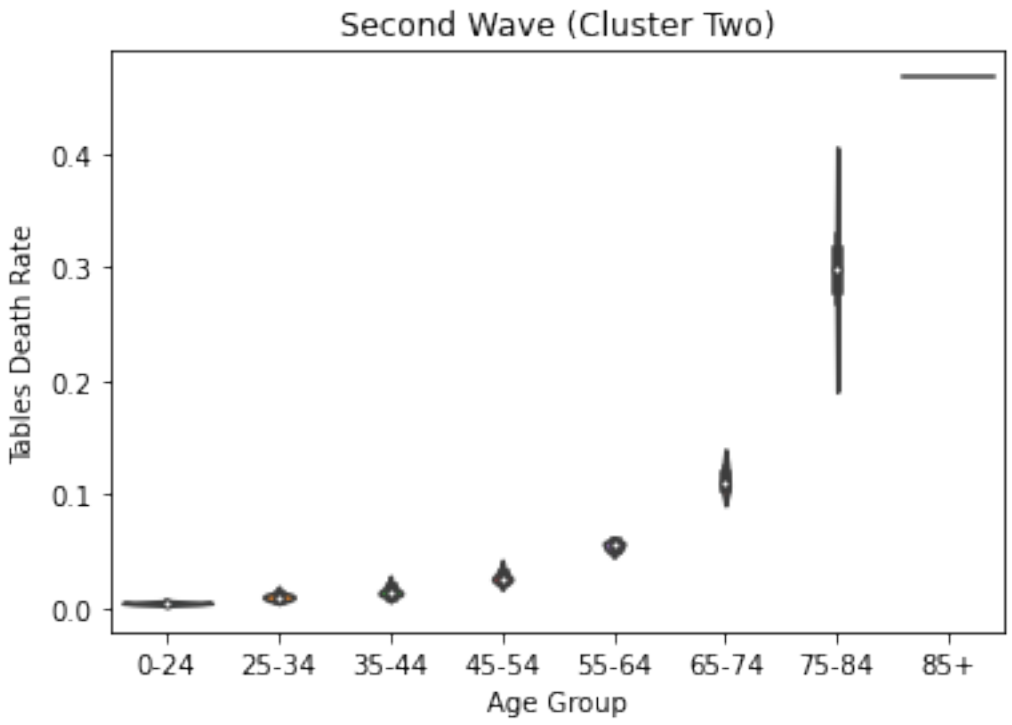
Peak Two Data Clusters Pattern

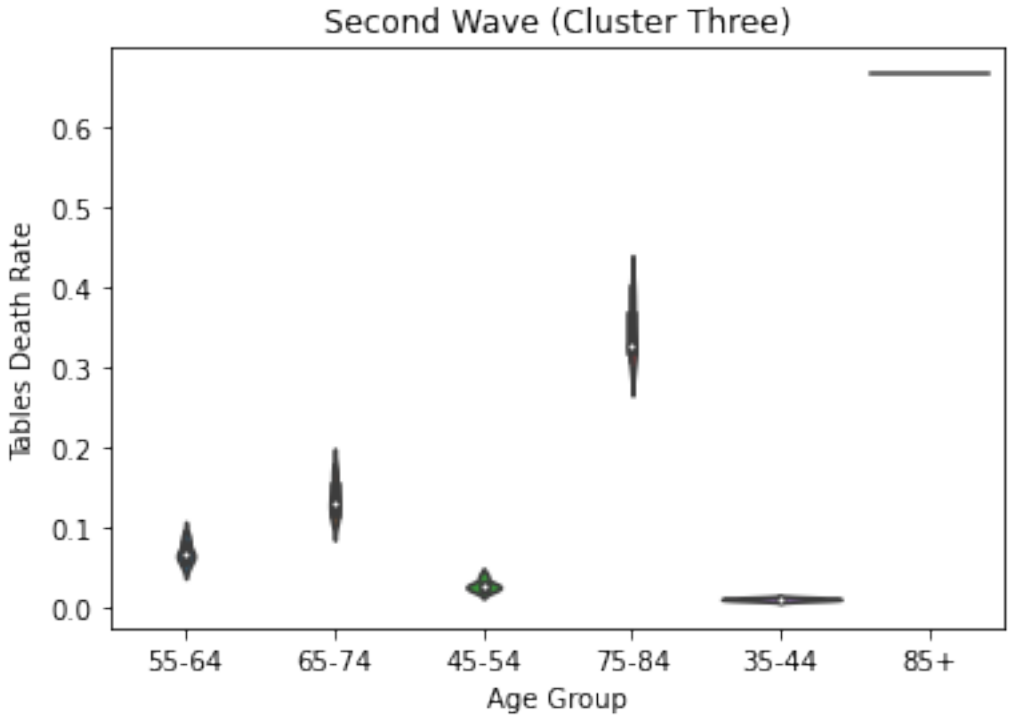
[203]:
```
df_gen2_cl1 = pd.DataFrame(general_two_clusters_dict[0], columns = general_cols)
df_gen2_cl2 = pd.DataFrame(general_two_clusters_dict[1], columns = general_cols)
df_gen2_cl3 = pd.DataFrame(general_two_clusters_dict[2], columns = general_cols)
df_gen2_cl4 = pd.DataFrame(general_two_clusters_dict[3], columns = general_cols)

df_cond2_cl1 = pd.DataFrame(cond_two_clusters_dict[0], columns = cond_cols)
df_cond2_cl2 = pd.DataFrame(cond_two_clusters_dict[1], columns = cond_cols)
df_cond2_cl3 = pd.DataFrame(cond_two_clusters_dict[2], columns = cond_cols)
df_cond2_cl4 = pd.DataFrame(cond_two_clusters_dict[3], columns = cond_cols)
```

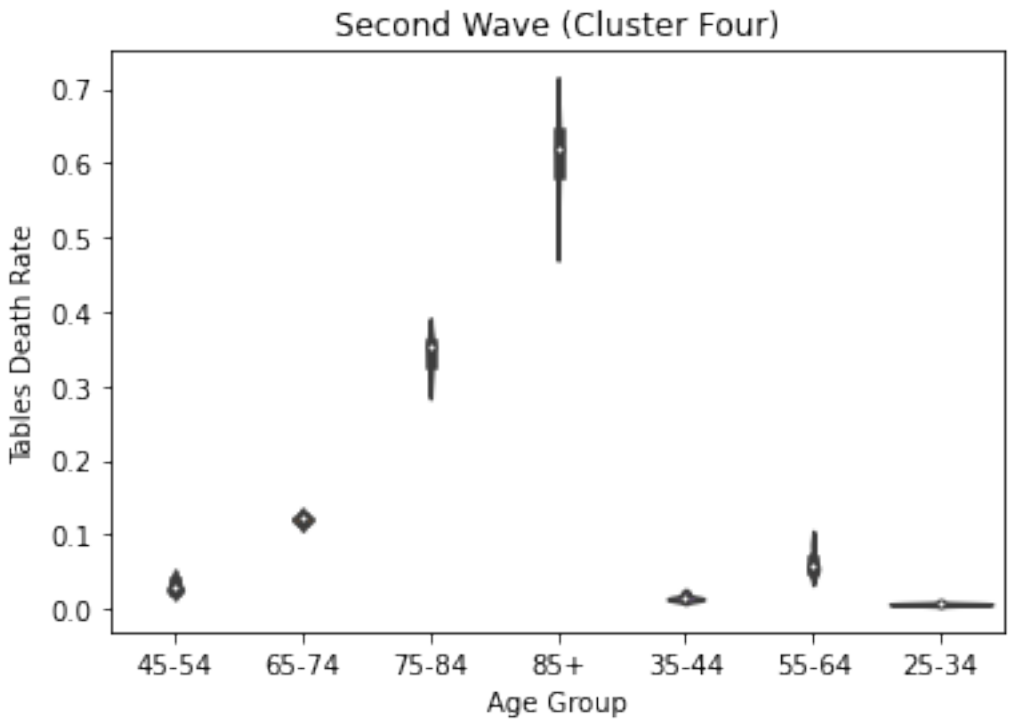
[204]:
```
df__gen2_cl_lst = [df_gen2_cl1,df_gen2_cl2, df_gen2_cl3, df_gen2_cl4 ]
```

[205]:
```
# death rate by age groups
plot_death_rate_by_group(df__gen2_cl_lst, "Age Group", 'Tables Death Rate',␣
↪"Second Wave")
```




Second Wave (Cluster Two)
Tables Death Rate
0.0
0.1
0.2
0.3
0.4
0-24
25-34
35-44
45-54
55-64
65-74
75-84
85+
Age Group

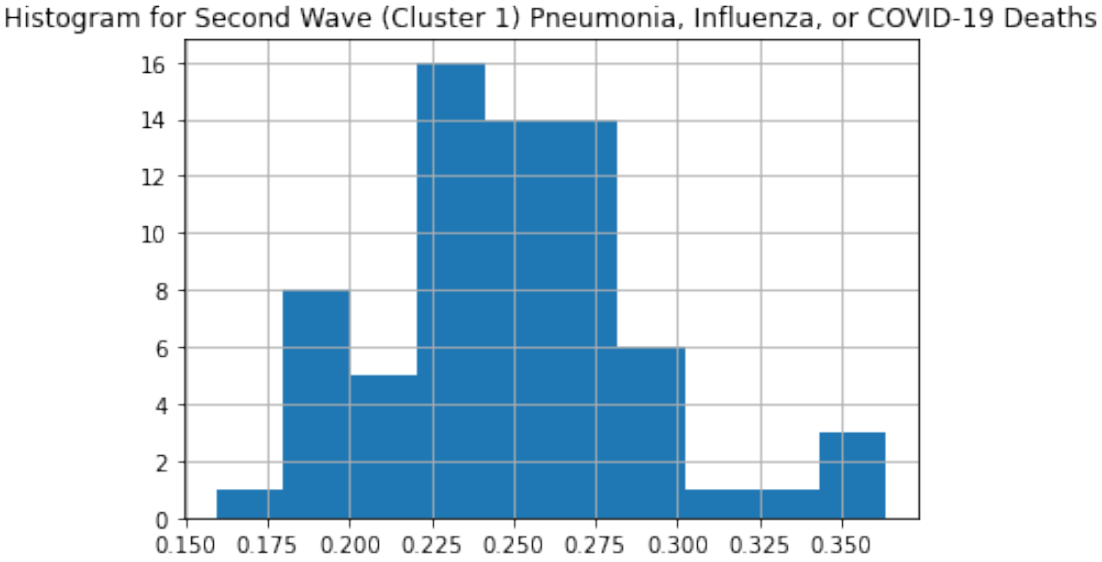

Second Wave (Cluster Three)
Tables Death Rate
0.0
0.1
0.2
0.3
0.4
0.5
0.6
55-64
65-74
45-54
75-84
35-44
85+
Age Group



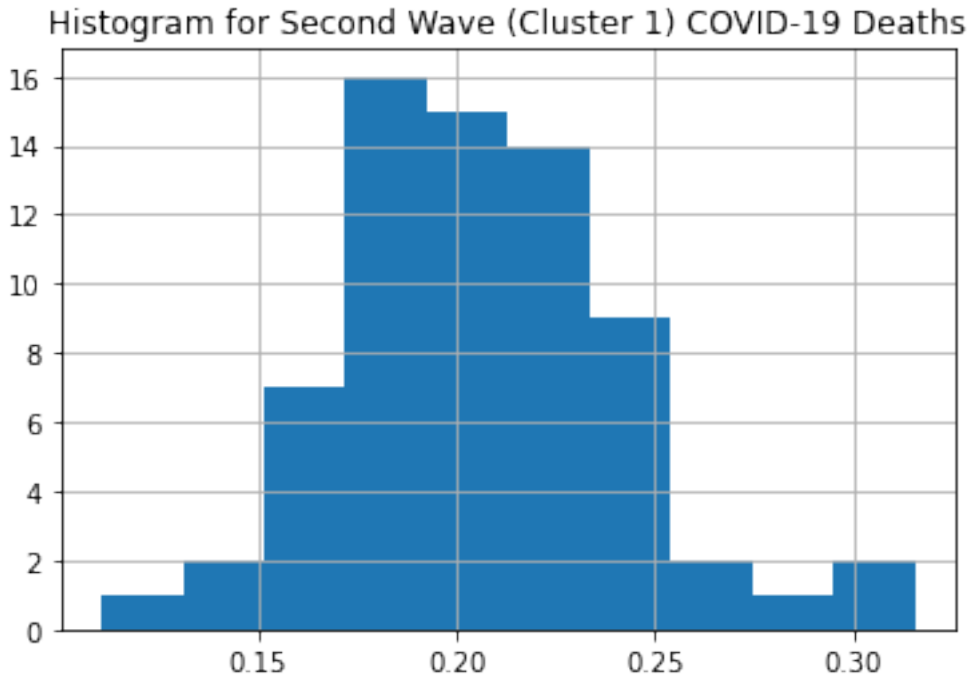
[206]:
```
df_cond2_cl_lst = [df_cond2_cl1, df_cond2_cl2, df_cond2_cl3, df_cond2_cl4]
```

[207]:
```
# death condition with top 3 highest death rate in each cluster
plot_death_rate_cond(df_cond2_cl_lst, 3, "Histogram for Second Wave")
```

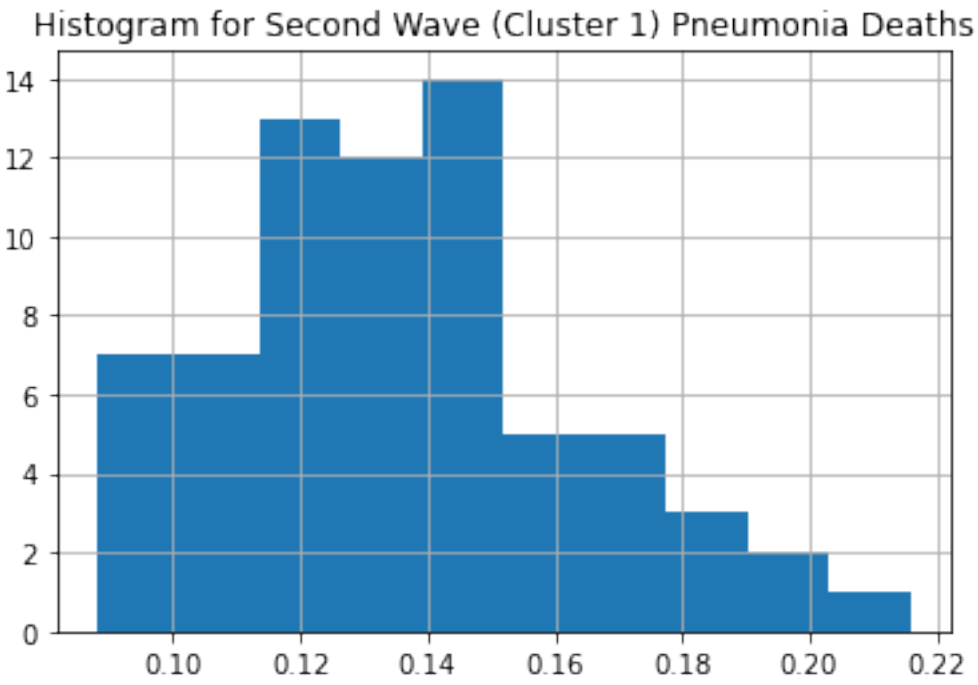
```
Cluster1:
Plotting for column Pneumonia, Influenza, or COVID-19 Deaths
```



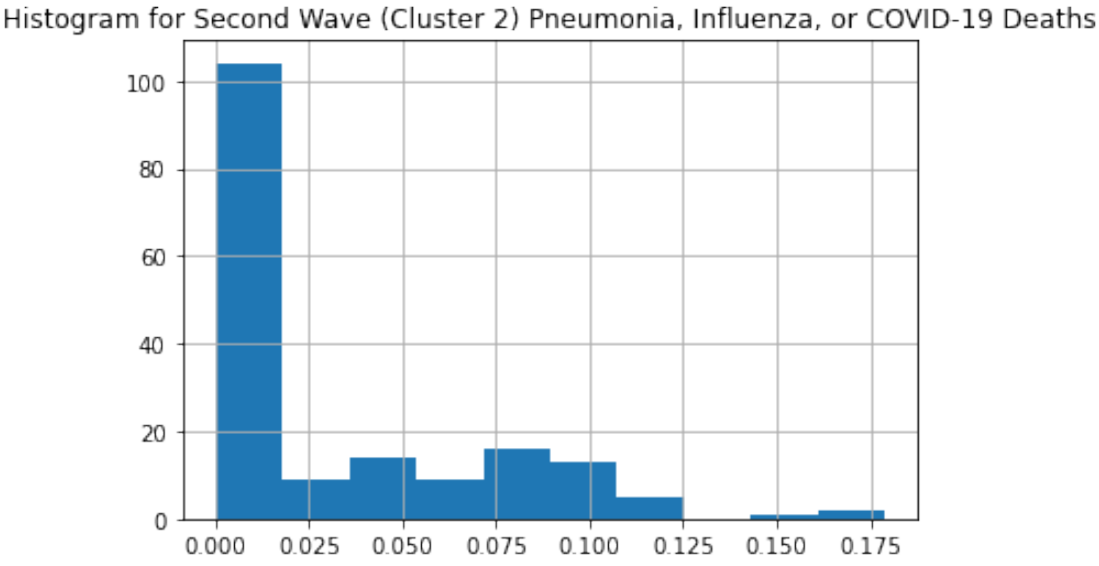
```
Plotting for column COVID-19 Deaths
```



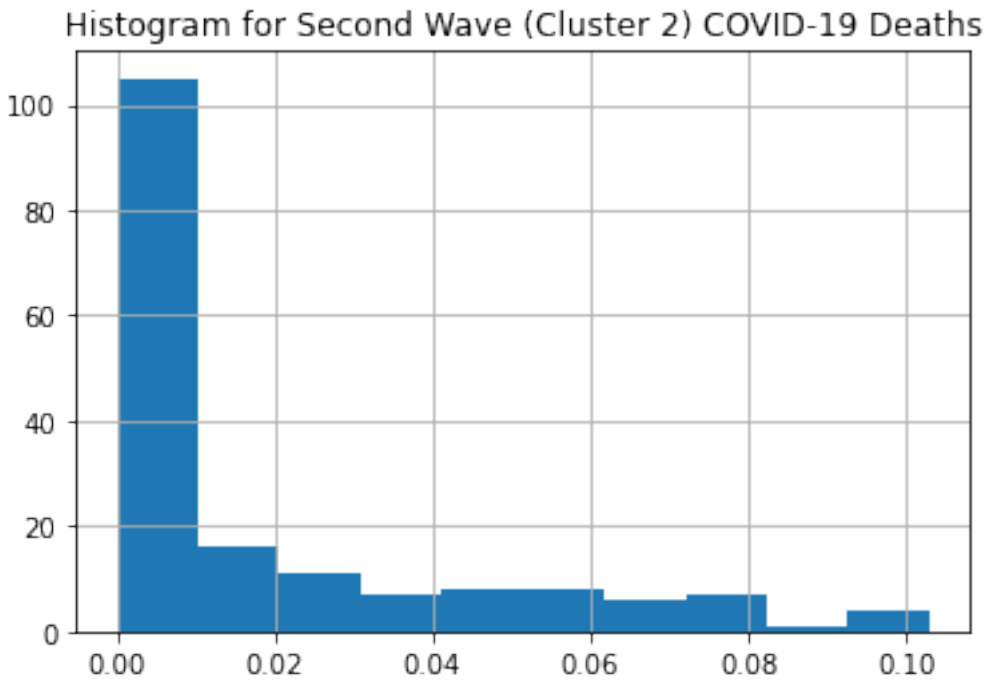

```
Plotting for column Pneumonia Deaths
```



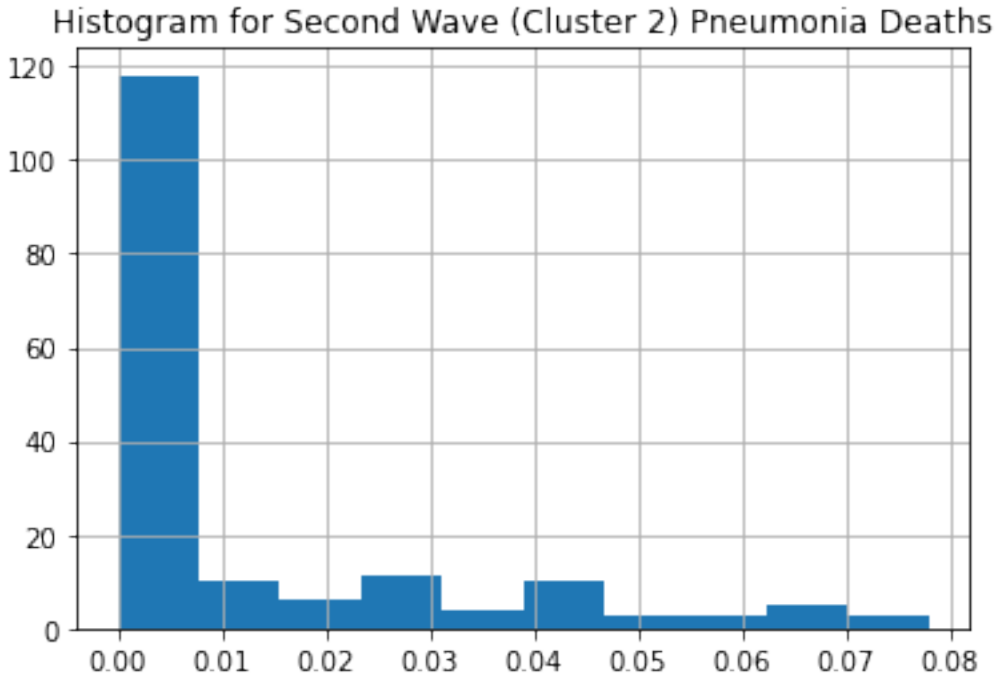
Cluster2:
Plotting for column Pneumonia, Influenza, or COVID-19 Deaths



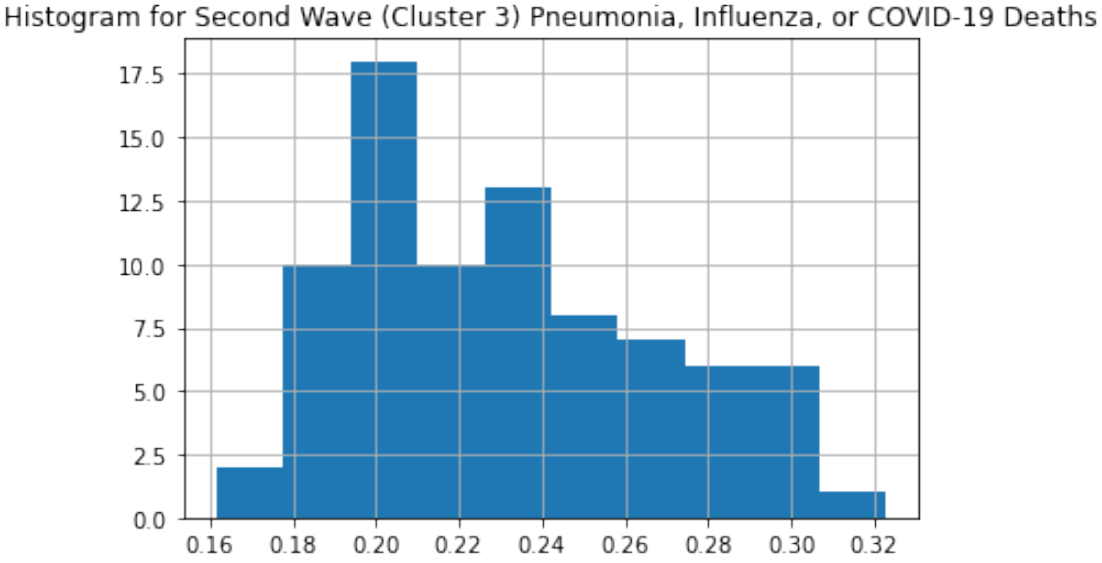
Plotting for column COVID-19 Deaths



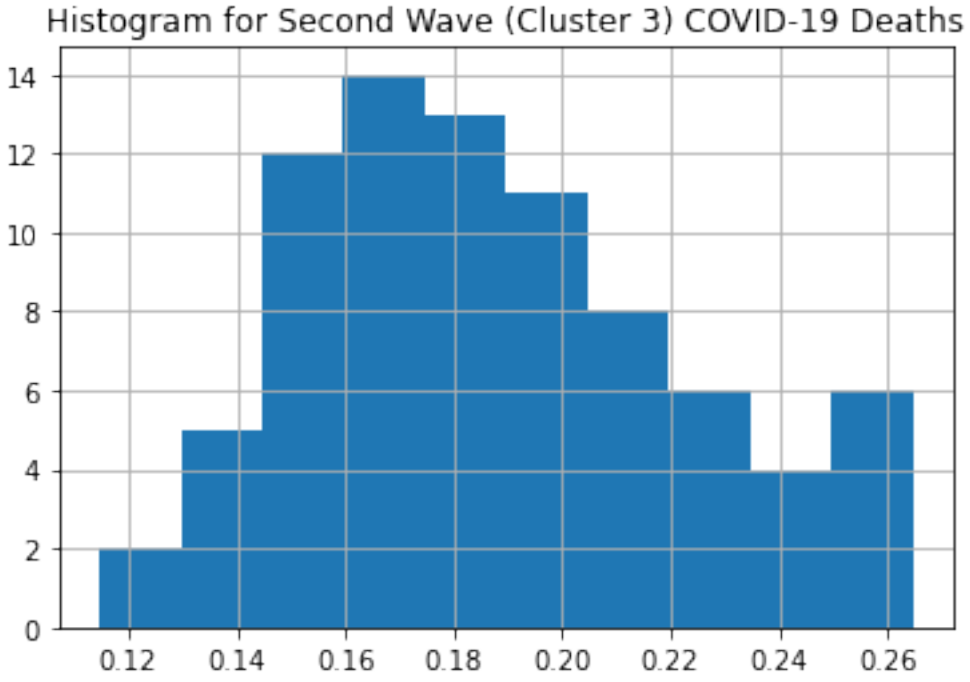
Plotting for column Pneumonia Deaths



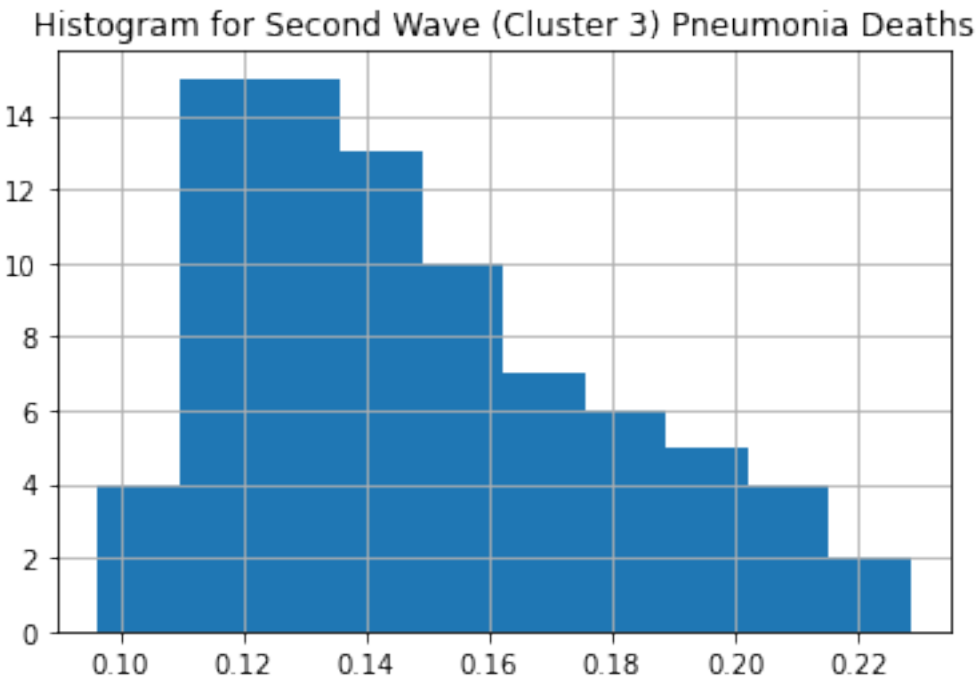
Cluster3:
Plotting for column Pneumonia, Influenza, or COVID-19 Deaths



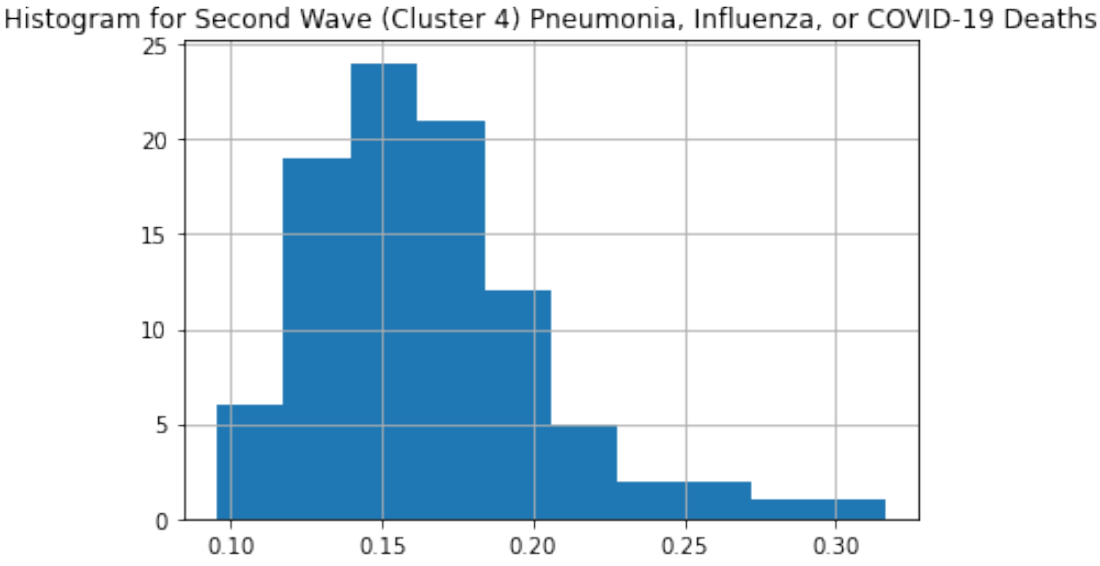
Plotting for column COVID-19 Deaths



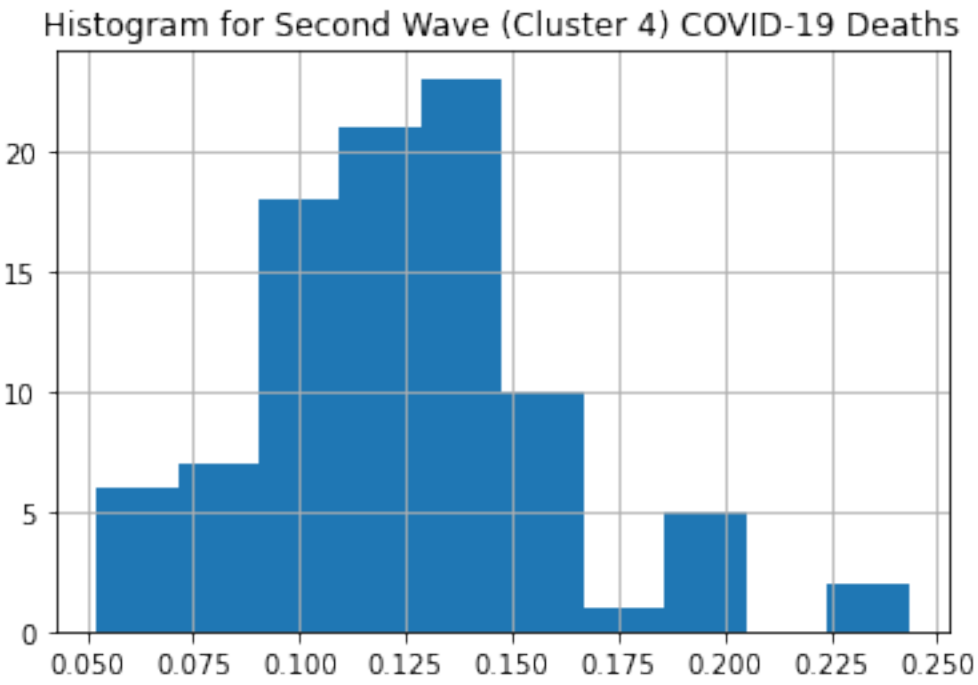
Plotting for column Pneumonia Deaths



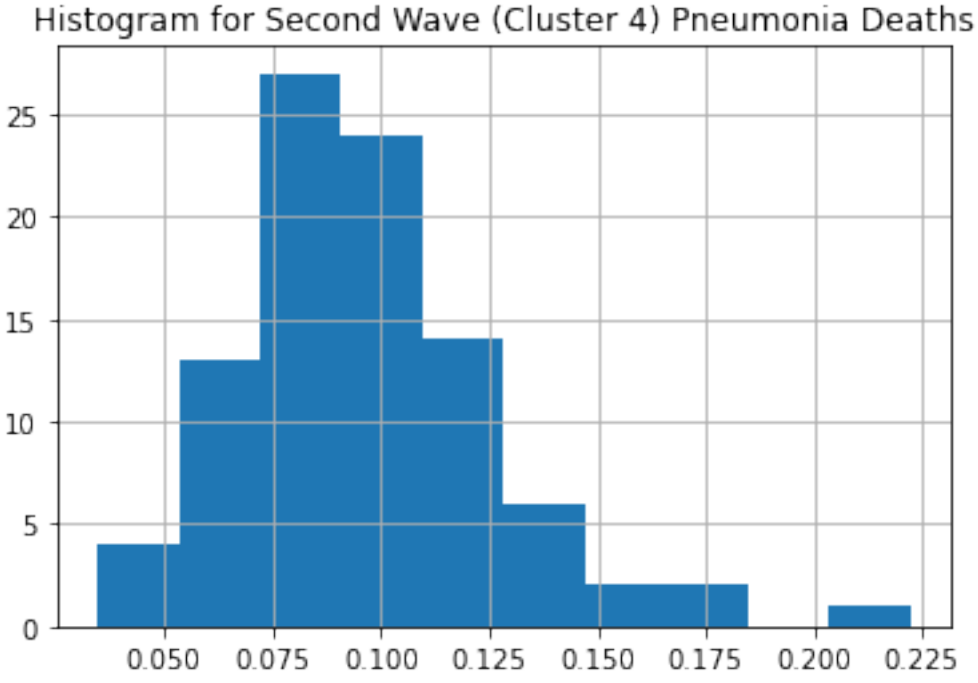
Cluster4:
Plotting for column Pneumonia, Influenza, or COVID-19 Deaths



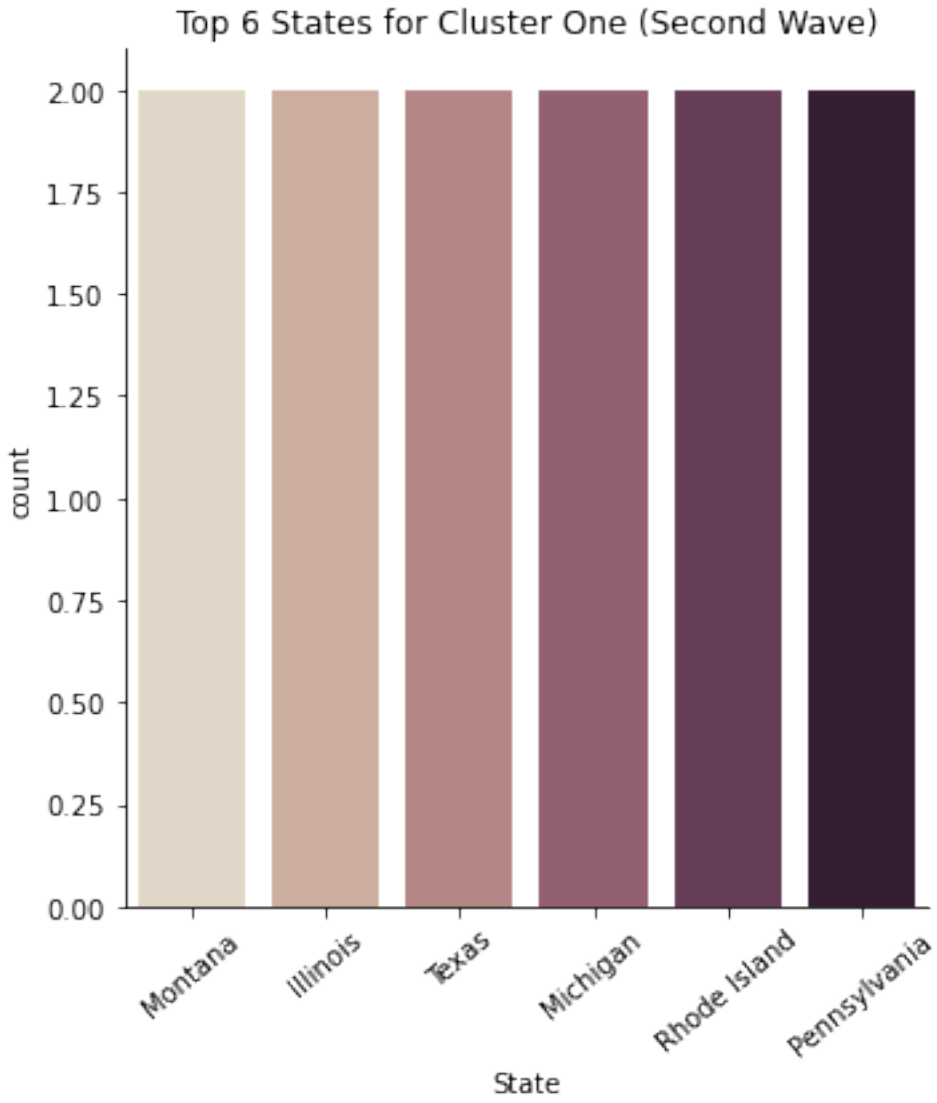
Plotting for column COVID-19 Deaths

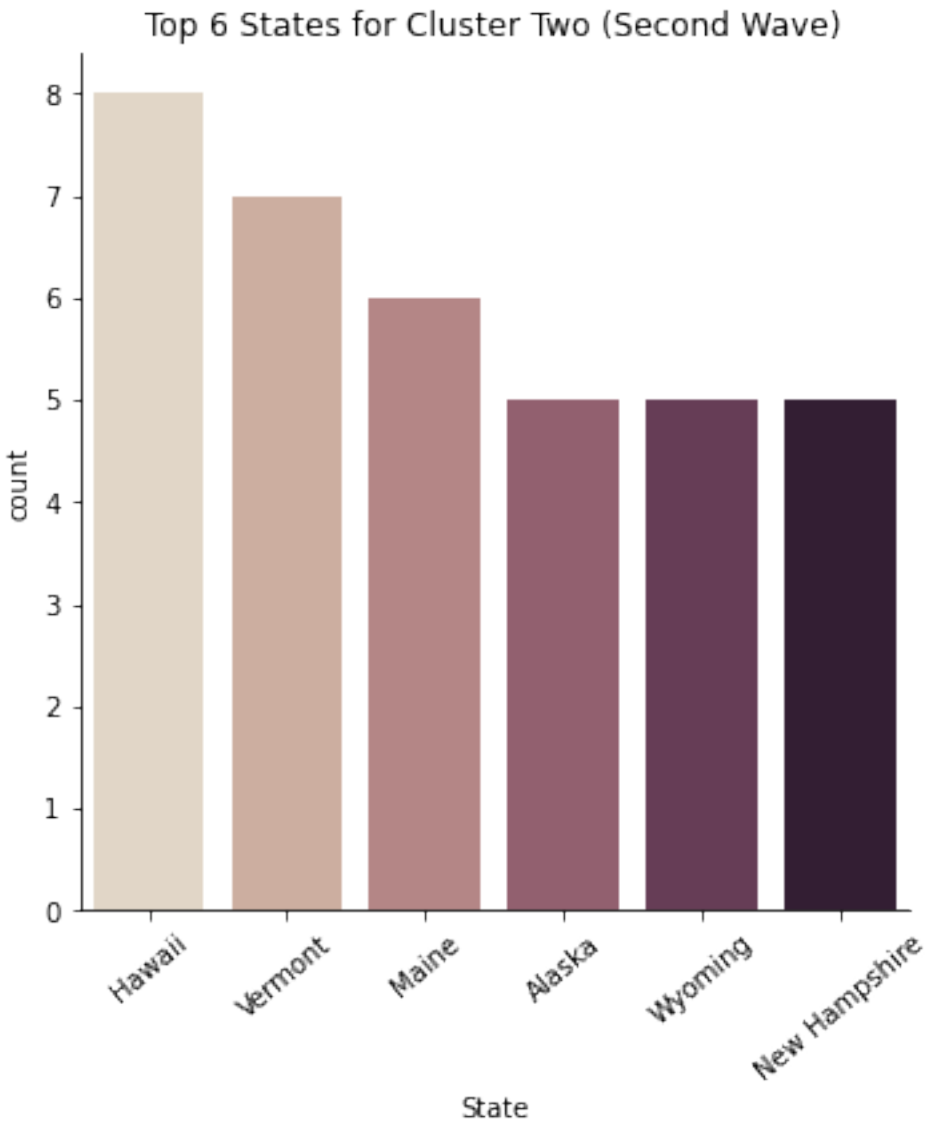


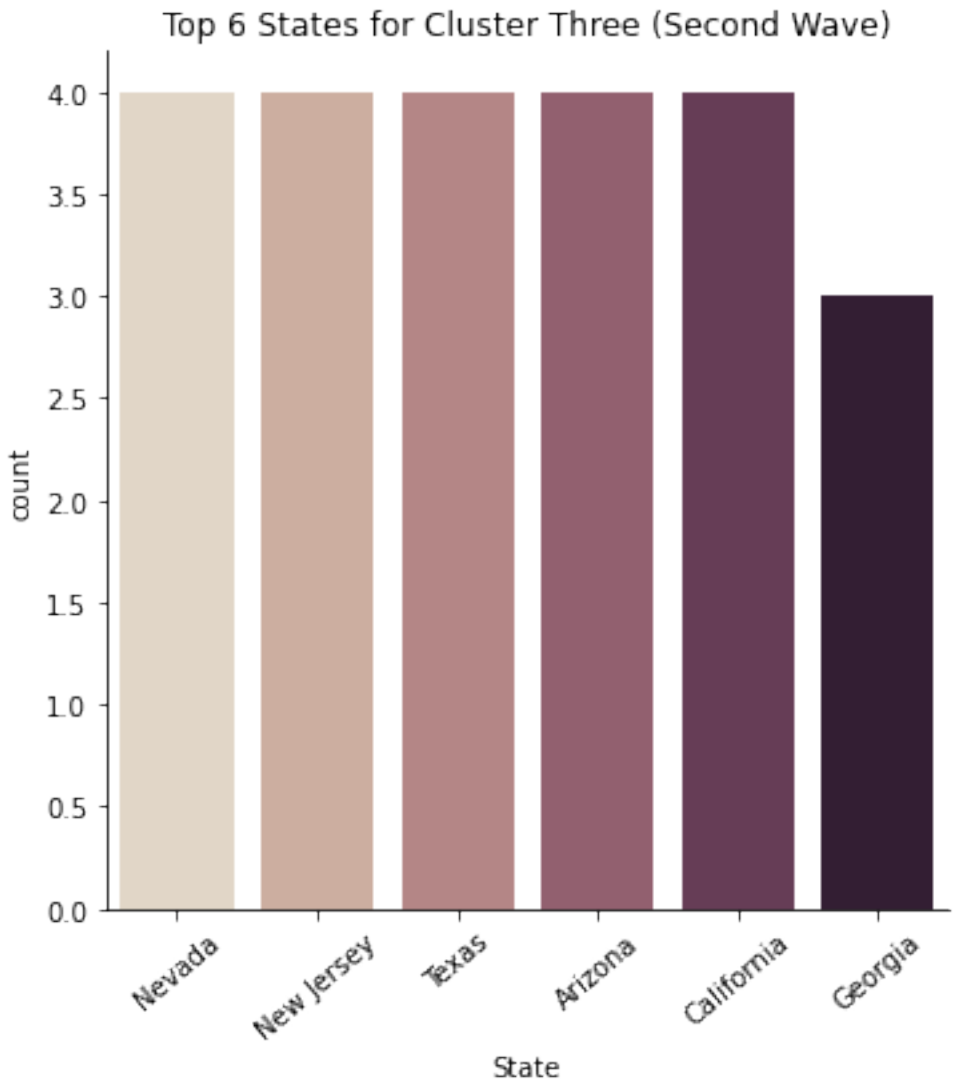
Plotting for column Pneumonia Deaths

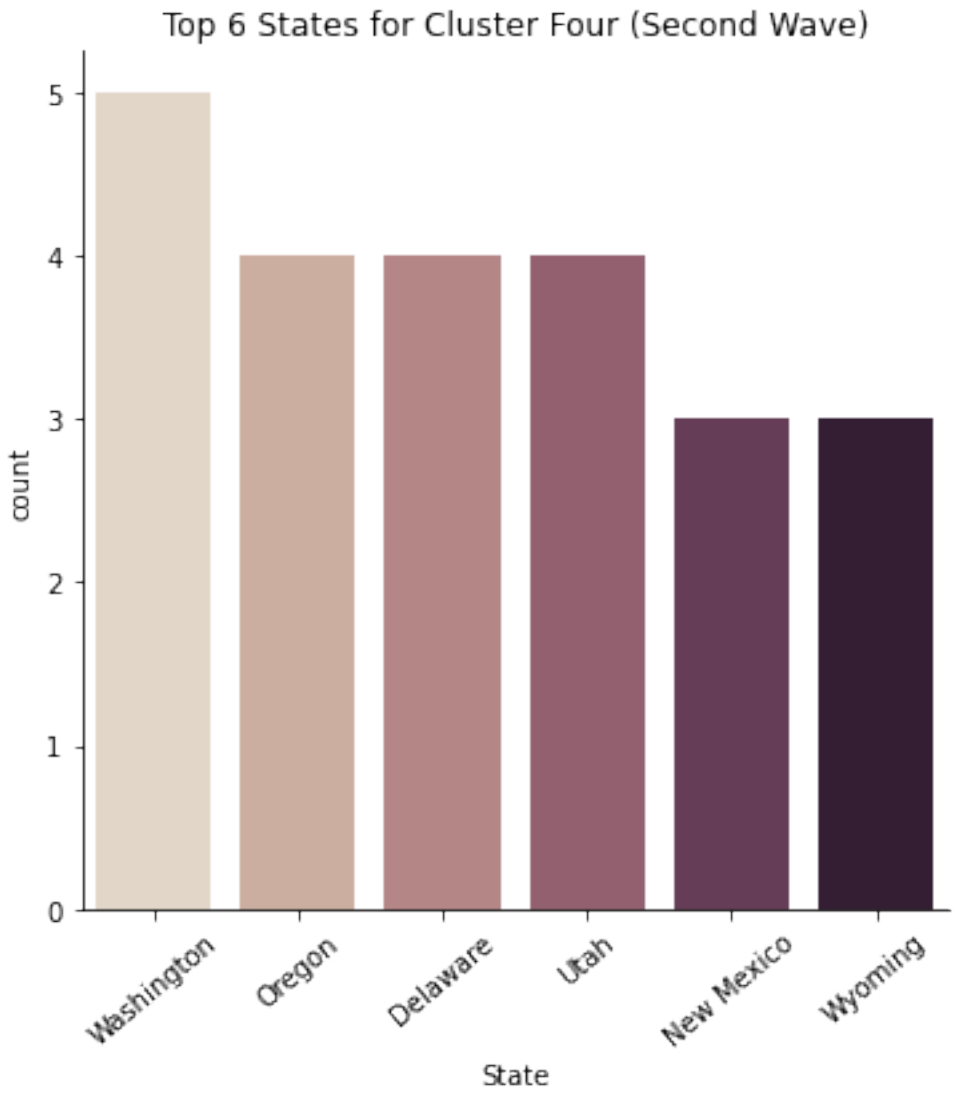


[208]:
```
# plot top 3 states
plot_all_clusters_group(df__gen2_cl_lst, 6, x_name0, "Second Wave")
```

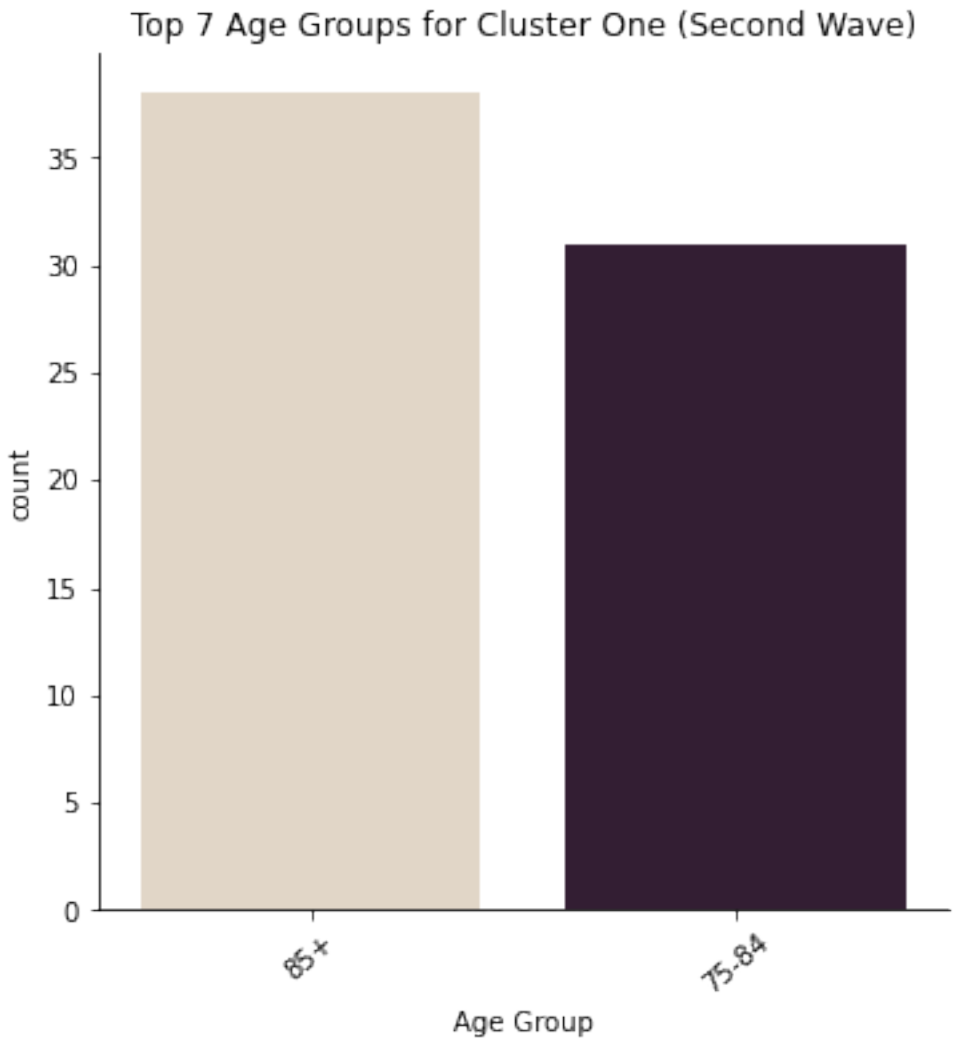


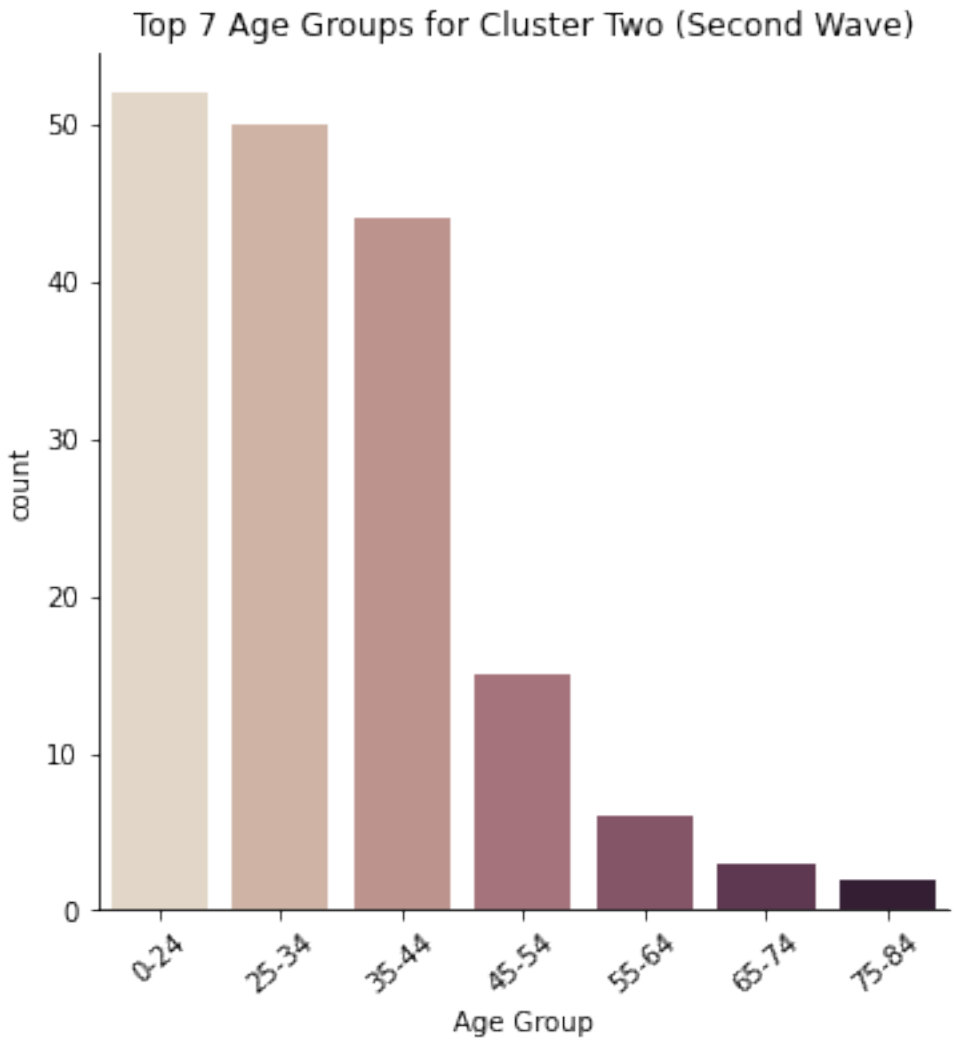

Top 6 States for Cluster Two (Second Wave)
count
0
1
2
3
4
5
6
7
8
Hawaii
Vermont
Maine
Alaska
Wyoming
New Hampshire
State

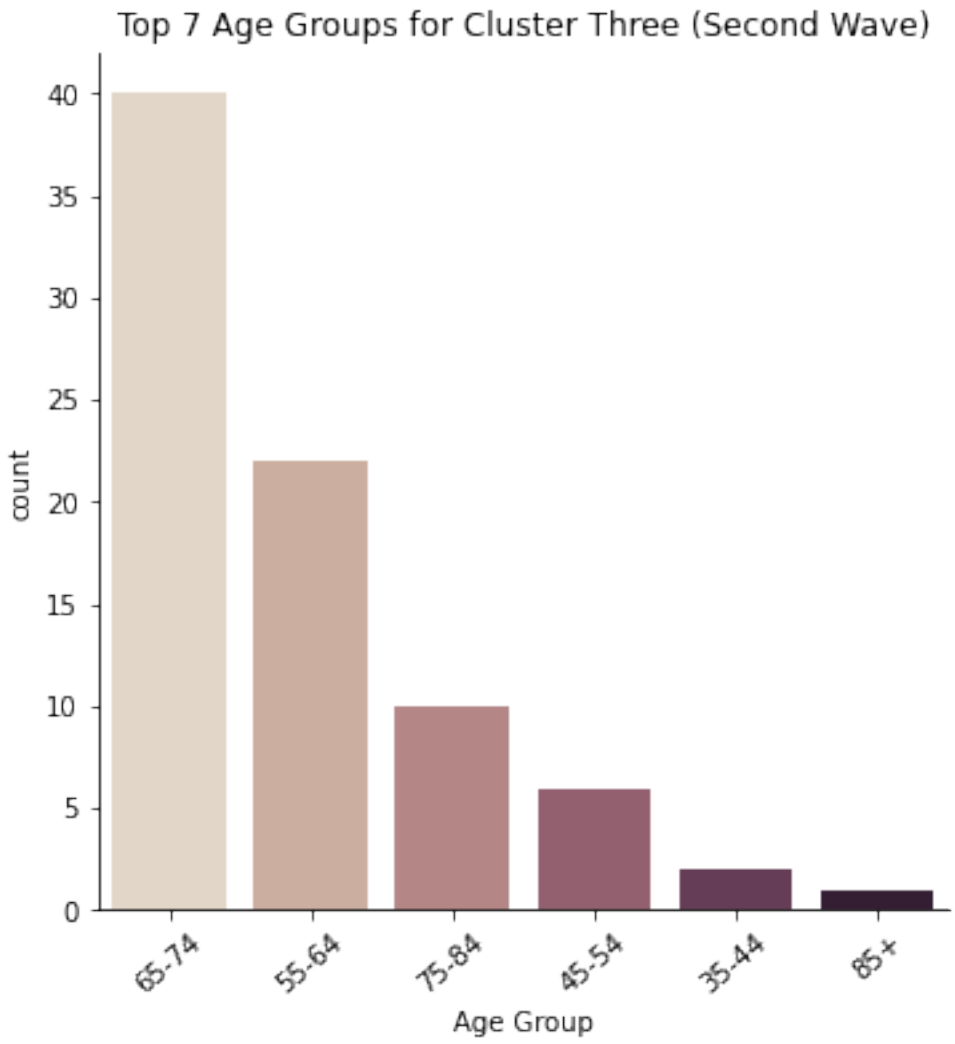

Top 6 States for Cluster Three (Second Wave)
count
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Nevada
New Jersey
Texas
Arizona
California
Georgia
State

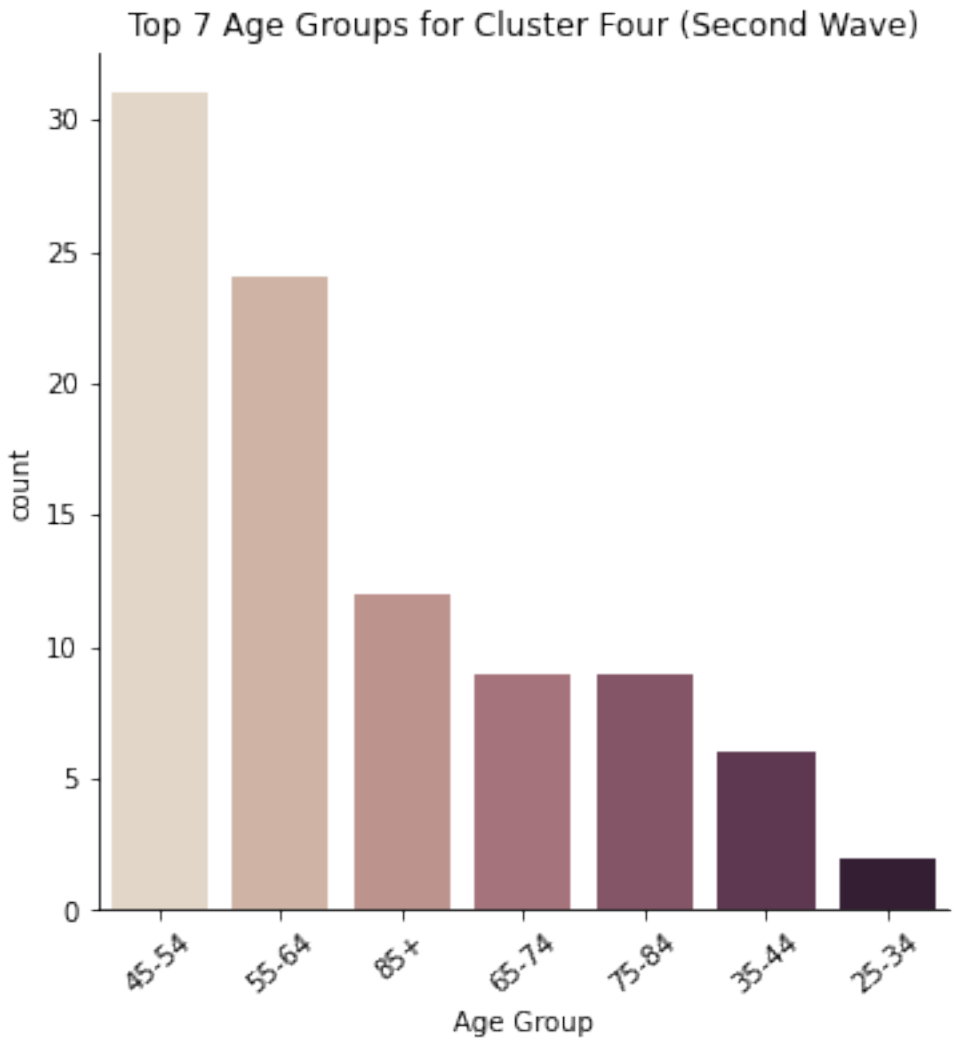


[209]:
```
# plot top age group
plot_all_clusters_group(df__gen2_cl_lst, 7, x_name1, "Second Wave")
```


Top 7 Age Groups for Cluster One (Second Wave)
count
0
5
10
15
20
25
30
35
85+
75-84
Age Group


Top 7 Age Groups for Cluster Two (Second Wave)
count
50
40
30
20
10
0
0-24
25-34
35-44
45-54
55-64
65-74
75-84
Age Group


Top 7 Age Groups for Cluster Three (Second Wave)
count
0
5
10
15
20
25
30
35
40
65-74
55-64
75-84
45-54
35-44
85+
Age Group



[210]:
```
OriginalPeakOne['Cluster'] = labels_one + 1
OriginalPeakTwo['Cluster'] = labels_two + 1
OriginalPeakOne['Cluster'] = OriginalPeakOne['Cluster'].astype(str)
OriginalPeakTwo['Cluster'] = OriginalPeakTwo['Cluster'].astype(str)
```

```
<ipython-input-210-07838acd3c0d>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  OriginalPeakOne['Cluster'] = labels_one + 1
<ipython-input-210-07838acd3c0d>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  OriginalPeakTwo['Cluster'] = labels_two + 1
<ipython-input-210-07838acd3c0d>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  OriginalPeakOne['Cluster'] = OriginalPeakOne['Cluster'].astype(str)
<ipython-input-210-07838acd3c0d>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  OriginalPeakTwo['Cluster'] = OriginalPeakTwo['Cluster'].astype(str)
```

[211]:
```
OriginalPeakOne[['Tables Death Rate', 'Average Age of Death', 'Average Age',␣
↪'LATITUDE',
       'LONGITUDE']] = PeakOneNoPCAScale[['Tables Death Rate', 'Average Age of␣
↪Death', 'Average Age', 'LATITUDE',
       'LONGITUDE']]
OriginalPeakTwo[['Tables Death Rate', 'Average Age of Death', 'Average Age',␣
↪'LATITUDE',
       'LONGITUDE']] = PeakTwoNoPCAScale[['Tables Death Rate', 'Average Age of␣
↪Death', 'Average Age', 'LATITUDE',
       'LONGITUDE']]
OriginalPeakOne[['Tables Death Rate', 'Average Age of Death', 'Average Age',␣
↪'LATITUDE',
       'LONGITUDE']] = OriginalPeakOne[['Tables Death Rate', 'Average Age of␣
↪Death', 'Average Age', 'LATITUDE',
       'LONGITUDE']]/9
OriginalPeakTwo[['Tables Death Rate', 'Average Age of Death', 'Average Age',␣
↪'LATITUDE',
       'LONGITUDE']]= OriginalPeakTwo[['Tables Death Rate', 'Average Age of␣
↪Death', 'Average Age', 'LATITUDE',
       'LONGITUDE']]/7
```

```
C:\Users\williamshih\anaconda3\lib\site-packages\pandas\core\frame.py:3065:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
        self[k1] = value[k2]
```

[212]:
```
def plot_two_by_group(fix, axis1, axis2, title, filt, filtval):
    if filt != None:
        fix = fix[(fix[filt] == filtval)]
    sns.scatterplot(data = fix, x=axis1, y=axis2, hue="Cluster", hue_order =␣
 ↪['1','2','3','4'])
    if filt != None:
        title = title + " by " + filt + " (" + filtval + ")"
    plt.title(title)
    plt.savefig(title + ".png", dpi = 300)
    plt.clf()
```

[225]:
```
plot_two_by_group(OriginalPeakOne, "LONGITUDE", "LATITUDE", "K-Means for First␣
 ↪Wave by State", "Age Group", "0-24")
plot_two_by_group(OriginalPeakOne, "LONGITUDE", "LATITUDE", "K-Means for First␣
 ↪Wave by State", "Age Group", "25-34")
plot_two_by_group(OriginalPeakOne, "LONGITUDE", "LATITUDE", "K-Means for First␣
 ↪Wave by State", "Age Group", "35-44")
plot_two_by_group(OriginalPeakOne, "LONGITUDE", "LATITUDE", "K-Means for First␣
 ↪Wave by State", "Age Group", "45-54")
plot_two_by_group(OriginalPeakOne, "LONGITUDE", "LATITUDE", "K-Means for First␣
 ↪Wave by State", "Age Group", "55-64")
plot_two_by_group(OriginalPeakOne, "LONGITUDE", "LATITUDE", "K-Means for First␣
 ↪Wave by State", "Age Group", "65-74")
plot_two_by_group(OriginalPeakOne, "LONGITUDE", "LATITUDE", "K-Means for First␣
 ↪Wave by State", "Age Group", "75-84")
plot_two_by_group(OriginalPeakOne, "LONGITUDE", "LATITUDE", "K-Means for First␣
 ↪Wave by State", "Age Group", "85+")
```

```
<Figure size 432x288 with 0 Axes>
```

[226]:
```
plot_two_by_group(OriginalPeakTwo, "LONGITUDE", "LATITUDE", "K-Means for Second␣
 ↪Wave by State", "Age Group", "0-24")
plot_two_by_group(OriginalPeakTwo, "LONGITUDE", "LATITUDE", "K-Means for Second␣
 ↪Wave by State", "Age Group", "25-34")
plot_two_by_group(OriginalPeakTwo, "LONGITUDE", "LATITUDE", "K-Means for Second␣
 ↪Wave by State", "Age Group", "35-44")
plot_two_by_group(OriginalPeakTwo, "LONGITUDE", "LATITUDE", "K-Means for Second␣
 ↪Wave by State", "Age Group", "45-54")
plot_two_by_group(OriginalPeakTwo, "LONGITUDE", "LATITUDE", "K-Means for Second␣
 ↪Wave by State", "Age Group", "55-64")
plot_two_by_group(OriginalPeakTwo, "LONGITUDE", "LATITUDE", "K-Means for Second␣
 ↪Wave by State", "Age Group", "65-74")
plot_two_by_group(OriginalPeakTwo, "LONGITUDE", "LATITUDE", "K-Means for Second␣
 ↪Wave by State", "Age Group", "75-84")
```

```
plot_two_by_group(OriginalPeakTwo, "LONGITUDE", "LATITUDE", "K-Means for Second␣
 ↪Wave by State", "Age Group", "85+")
```

```
<Figure size 432x288 with 0 Axes>
```

[215]:
```
plot_two_by_group(OriginalPeakOne, "CDD", "HDD", "K-Means for First Wave by␣
 ↪CDD,HDD", "Age Group", "0-24")
plot_two_by_group(OriginalPeakOne, "CDD", "HDD", "K-Means for First Wave by␣
 ↪CDD,HDD", "Age Group", "25-34")
plot_two_by_group(OriginalPeakOne, "CDD", "HDD", "K-Means for First Wave by␣
 ↪CDD,HDD", "Age Group", "35-44")
plot_two_by_group(OriginalPeakOne, "CDD", "HDD", "K-Means for First Wave by␣
 ↪CDD,HDD", "Age Group", "45-54")
plot_two_by_group(OriginalPeakOne, "CDD", "HDD", "K-Means for First Wave by␣
 ↪CDD,HDD", "Age Group", "55-64")
plot_two_by_group(OriginalPeakOne, "CDD", "HDD", "K-Means for First Wave by␣
 ↪CDD,HDD", "Age Group", "65-74")
plot_two_by_group(OriginalPeakOne, "CDD", "HDD", "K-Means for First Wave by␣
 ↪CDD,HDD", "Age Group", "75-84")
plot_two_by_group(OriginalPeakOne, "CDD", "HDD", "K-Means for First Wave by␣
 ↪CDD,HDD", "Age Group", "85+")
```

```
<Figure size 432x288 with 0 Axes>
```

[216]:
```
plot_two_by_group(OriginalPeakTwo, "CDD", "HDD", "K-Means for Second Wave by␣
 ↪CDD,HDD", "Age Group", "0-24")
plot_two_by_group(OriginalPeakTwo, "CDD", "HDD", "K-Means for Second Wave by␣
 ↪CDD,HDD", "Age Group", "25-34")
plot_two_by_group(OriginalPeakTwo, "CDD", "HDD", "K-Means for Second Wave by␣
 ↪CDD,HDD", "Age Group", "35-44")
plot_two_by_group(OriginalPeakTwo, "CDD", "HDD", "K-Means for Second Wave by␣
 ↪CDD,HDD", "Age Group", "45-54")
plot_two_by_group(OriginalPeakTwo, "CDD", "HDD", "K-Means for Second Wave by␣
 ↪CDD,HDD", "Age Group", "55-64")
plot_two_by_group(OriginalPeakTwo, "CDD", "HDD", "K-Means for Second Wave by␣
 ↪CDD,HDD", "Age Group", "65-74")
plot_two_by_group(OriginalPeakTwo, "CDD", "HDD", "K-Means for Second Wave by␣
 ↪CDD,HDD", "Age Group", "75-84")
plot_two_by_group(OriginalPeakTwo, "CDD", "HDD", "K-Means for Second Wave by␣
 ↪CDD,HDD", "Age Group", "85+")
```

```
<Figure size 432x288 with 0 Axes>
```

[185]:
```
PeakOneNoPCAScale['Cluster'] = OriginalPeakOne['Cluster']
PeakTwoNoPCAScale['Cluster'] = OriginalPeakTwo['Cluster']
```

[186]:
```
plot_two_by_group(PeakOneNoPCAScale,"COVID-19 Deaths","Tables Death␣
↪Rate","K-Means for First Wave by COVID-19 Deaths (Ratio over Total Deaths)␣
↪vs Overall Death Rate",None,None)
plot_two_by_group(PeakTwoNoPCAScale,"COVID-19 Deaths","Tables Death␣
↪Rate","K-Means for Second Wave by COVID-19 Deaths (Ratio over Total Deaths)␣
↪vs Overall Death Rate",None,None)
```

```
<Figure size 432x288 with 0 Axes>
```

[187]:
```
plot_two_by_group(PeakOneNoPCAScale,"HDD", "COVID-19 Deaths","K-Means for First␣
↪Wave by HDD vs COVID-19 Deaths (Ratio over Total Deaths)",None,None)
plot_two_by_group(PeakTwoNoPCAScale,"HDD", "COVID-19 Deaths","K-Means for␣
↪Second Wave by HDD vs COVID-19 Deaths (Ratio over Total Deaths)",None,None)
```

```
<Figure size 432x288 with 0 Axes>
```

[188]:
```
PeakOneNoPCAScale
```

[188]:

```
     Adult respiratory distress syndrome  \
0                                0.000000
2                                0.000000
4                                0.000000
6                                0.000000
8                                0.001457
..                                    …
924                              0.000000
926                              0.000000
928                              0.000000
930                              0.000000
932                              0.000000

     All other conditions and causes (residual)  Alzheimer disease  \
0                                      0.000000                0.0
2                                      0.000000                0.0
4                                      0.000000                0.0
6                                      0.022304                0.0
8                                      0.023452                0.0
..                                          …                   …
924                                    0.000000                0.0
926                                    0.000000                0.0
928                                    0.000000                0.0
930                                    0.000000                0.0
932                                    0.000000                0.0

     Cardiac arrest  Cardiac arrhythmia  Cerebrovascular diseases  \
0          0.000000                 0.0                       0.0
2          0.000000                 0.0                       0.0
4          0.000000                 0.0                       0.0
```

```
6          0.000000                 0.0                       0.0
8          0.013547                 0.0                       0.0
..              …                   …                         …
924        0.000000                 0.0                       0.0
926        0.000000                 0.0                       0.0
928        0.000000                 0.0                       0.0
930        0.000000                 0.0                       0.0
932        0.000000                 0.0                       0.0

     Chronic lower respiratory diseases  Diabetes  Heart failure  \
0                              0.000000  0.000000            0.0
2                              0.000000  0.000000            0.0
4                              0.000000  0.000000            0.0
6                              0.000000  0.000000            0.0
8                              0.002039  0.007866            0.0
..                                  …       …              …
924                            0.000000  0.000000            0.0
926                            0.000000  0.000000            0.0
928                            0.000000  0.000000            0.0
930                            0.000000  0.000000            0.0
932                            0.000000  0.000000            0.0

     Hypertensive diseases  …  Pneumonia, Influenza, or COVID-19 Deaths  \
0                 0.000000  …                                  0.000000
2                 0.000000  …                                  0.025151
4                 0.000000  …                                  0.077358
6                 0.000000  …                                  0.117177
8                 0.006264  …                                  0.134596
..                     …  …                                     …
924               0.000000  …                                  0.000000
926               0.000000  …                                  0.000000
928               0.000000  …                                  0.015625
930               0.000000  …                                  0.032538
932               0.000000  …                                  0.039252

     Tables Death Rate  Average Age of Death  Average Age         CDD  \
0             0.000826             11.728497    12.221907  217.444444
2             0.001795             29.763483    29.389798  217.444444
4             0.003077             39.836973    39.447028  217.444444
6             0.005703             50.168650    49.552707  217.444444
8             0.012357             60.010934    59.434400  217.444444
..                 …                     …          …          …
924           0.004547             50.209237    49.501436   49.666667
926           0.008267             60.073045    59.609790   49.666667
928           0.018421             69.715637    68.981680   49.666667
930           0.052393             79.572537    78.733930   49.666667
932           0.088160             85.000000    85.000000   49.666667
```

```
             HDD    LATITUDE   LONGITUDE  Cluster  Pneumonia/Influenza Only
0    139.777778  297.072873 -780.811434        2                  0.000000
2    139.777778  297.072873 -780.811434        2                  0.025151
4    139.777778  297.072873 -780.811434        2                  0.037736
6    139.777778  297.072873 -780.811434        1                  0.046272
8    139.777778  297.072873 -780.811434        1                  0.062345
..          …         …         …      …                        …
924  556.666667  384.273234 -963.172134        2                  0.000000
926  556.666667  384.273234 -963.172134        2                  0.000000
928  556.666667  384.273234 -963.172134        2                  0.015625
930  556.666667  384.273234 -963.172134        2                  0.032538
932  556.666667  384.273234 -963.172134        2                  0.039252

[416 rows x 36 columns]
```

[189]:
```
PeakOneNoPCAScale['Pneumonia/Influenza Only'] = PeakOneNoPCAScale['Pneumonia,␣
↪Influenza, or COVID-19 Deaths'] - PeakOneNoPCAScale['COVID-19 Deaths']
PeakTwoNoPCAScale['Pneumonia/Influenza Only'] = PeakTwoNoPCAScale['Pneumonia,␣
↪Influenza, or COVID-19 Deaths'] - PeakTwoNoPCAScale['COVID-19 Deaths']
```

[190]:
```
plot_two_by_group(PeakOneNoPCAScale,"Pneumonia/Influenza Only", "COVID-19␣
↪Deaths","K-Means for First Wave by Pneumonia,Influenza vs COVID-19 Deaths␣
↪(Ratio over Total Deaths)",None,None)
plot_two_by_group(PeakTwoNoPCAScale,"Pneumonia/Influenza Only", "COVID-19␣
↪Deaths","K-Means for Second Wave by Pneumonia,Influenza vs COVID-19 Deaths␣
↪(Ratio over Total Deaths)",None,None)
```

```
<Figure size 432x288 with 0 Axes>
```

[191]:
```
PeakOneNoPCAScale
```

[191]:
```
     Adult respiratory distress syndrome  \
0                                0.000000
2                                0.000000
4                                0.000000
6                                0.000000
8                                0.001457
..                                    …
924                              0.000000
926                              0.000000
928                              0.000000
930                              0.000000
932                              0.000000

     All other conditions and causes (residual)  Alzheimer disease  \
0                                      0.000000               0.0
2                                      0.000000               0.0
```

```
4                                    0.000000                0.0
6                                    0.022304                0.0
8                                    0.023452                0.0
..                                        …                   …
924                                  0.000000                0.0
926                                  0.000000                0.0
928                                  0.000000                0.0
930                                  0.000000                0.0
932                                  0.000000                0.0

     Cardiac arrest  Cardiac arrhythmia  Cerebrovascular diseases  \
0          0.000000                 0.0                       0.0
2          0.000000                 0.0                       0.0
4          0.000000                 0.0                       0.0
6          0.000000                 0.0                       0.0
8          0.013547                 0.0                       0.0
..              …                   …                         …
924        0.000000                 0.0                       0.0
926        0.000000                 0.0                       0.0
928        0.000000                 0.0                       0.0
930        0.000000                 0.0                       0.0
932        0.000000                 0.0                       0.0

     Chronic lower respiratory diseases  Diabetes  Heart failure  \
0                              0.000000  0.000000            0.0
2                              0.000000  0.000000            0.0
4                              0.000000  0.000000            0.0
6                              0.000000  0.000000            0.0
8                              0.002039  0.007866            0.0
..                                  …       …              …
924                            0.000000  0.000000            0.0
926                            0.000000  0.000000            0.0
928                            0.000000  0.000000            0.0
930                            0.000000  0.000000            0.0
932                            0.000000  0.000000            0.0

     Hypertensive diseases  …  Pneumonia, Influenza, or COVID-19 Deaths  \
0                 0.000000  …                                   0.000000
2                 0.000000  …                                   0.025151
4                 0.000000  …                                   0.077358
6                 0.000000  …                                   0.117177
8                 0.006264  …                                   0.134596
..                     …  …                                      …
924               0.000000  …                                   0.000000
926               0.000000  …                                   0.000000
928               0.000000  …                                   0.015625
930               0.000000  …                                   0.032538
```

```
932              0.000000  …                                    0.039252

     Tables Death Rate  Average Age of Death  Average Age         CDD  \
0              0.000826             11.728497    12.221907  217.444444
2              0.001795             29.763483    29.389798  217.444444
4              0.003077             39.836973    39.447028  217.444444
6              0.005703             50.168650    49.552707  217.444444
8              0.012357             60.010934    59.434400  217.444444
..                  …                     …           …          …
924            0.004547             50.209237    49.501436   49.666667
926            0.008267             60.073045    59.609790   49.666667
928            0.018421             69.715637    68.981680   49.666667
930            0.052393             79.572537    78.733930   49.666667
932            0.088160             85.000000    85.000000   49.666667

             HDD    LATITUDE   LONGITUDE  Cluster  Pneumonia/Influenza Only
0     139.777778  297.072873 -780.811434        2                  0.000000
2     139.777778  297.072873 -780.811434        2                  0.025151
4     139.777778  297.072873 -780.811434        2                  0.037736
6     139.777778  297.072873 -780.811434        1                  0.046272
8     139.777778  297.072873 -780.811434        1                  0.062345
..           …         …         …      …                          …
924   556.666667  384.273234 -963.172134        2                  0.000000
926   556.666667  384.273234 -963.172134        2                  0.000000
928   556.666667  384.273234 -963.172134        2                  0.015625
930   556.666667  384.273234 -963.172134        2                  0.032538
932   556.666667  384.273234 -963.172134        2                  0.039252

[416 rows x 36 columns]
```

[192]:
```
plot_two_by_group(PeakOneNoPCAScale,"Obesity", "Respiratory failure","K-Means␣
 ↪for First Wave by Obesity vs Respiratory failure (Ratio over Total␣
 ↪Deaths)",None,None)
plot_two_by_group(PeakTwoNoPCAScale,"Obesity", "Respiratory failure","K-Means␣
 ↪for Second Wave by Obesity vs Respiratory failure (Ratio over Total␣
 ↪Deaths)",None,None)
```

```
<Figure size 432x288 with 0 Axes>
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: