```
import pandas as pd
import numpy as np
import statistics as stat
import matplotlib.pyplot as plt
import plotly.graph_objects as go
from streamlit import columns

df_pop = pd.read_csv('data/covid_county_population_usafacts.csv')
df_deaths = pd.read_csv('data/covid_deaths_usafacts.csv')
df_confirmed = pd.read_csv('data/covid_confirmed_usafacts.csv')

df_pop.info()
df_deaths.info()
df_confirmed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3195 entries, 0 to 3194
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   countyFIPS   3195 non-null   int64 
 1   County Name  3195 non-null   object
 2   State        3195 non-null   object
 3   population   3195 non-null   int64 
dtypes: int64(2), object(2)
memory usage: 100.0+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3193 entries, 0 to 3192
Columns: 1269 entries, countyFIPS to 2023-07-23
dtypes: int64(1267), object(2)
memory usage: 30.9+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3193 entries, 0 to 3192
Columns: 1269 entries, countyFIPS to 2023-07-23
dtypes: int64(1267), object(2)
memory usage: 30.9+ MB
```

```
df_deaths[df_deaths['State']=='CA']
```

```
     countyFIPS                      County Name State  StateFIPS  \
190           0          Statewide Unallocated    CA          6   
191        6001                Alameda County     CA          6   
192        6003                  Alpine County     CA          6   
193        6005                  Amador County     CA          6   
194        6007                   Butte County     CA          6   
195        6009                Calaveras County     CA          6   
196        6011                   Colusa County     CA          6   
197        6013             Contra Costa County     CA          6   
198        6015                Del Norte County     CA          6   
199        6017                El Dorado County     CA          6   
```

200	6019	Fresno County	CA	6
201	6021	Glenn County	CA	6
202	6023	Humboldt County	CA	6
203	6025	Imperial County	CA	6
204	6027	Inyo County	CA	6
205	6029	Kern County	CA	6
206	6031	Kings County	CA	6
207	6033	Lake County	CA	6
208	6035	Lassen County	CA	6
209	6037	Los Angeles County	CA	6
210	6039	Madera County	CA	6
211	6041	Marin County	CA	6
212	6043	Mariposa County	CA	6
213	6045	Mendocino County	CA	6
214	6047	Merced County	CA	6
215	6049	Modoc County	CA	6
216	6051	Mono County	CA	6
217	6053	Monterey County	CA	6
218	6055	Napa County	CA	6
219	6057	Nevada County	CA	6
220	6059	Orange County	CA	6
221	6061	Placer County	CA	6
222	6063	Plumas County	CA	6
223	6065	Riverside County	CA	6
224	6067	Sacramento County	CA	6
225	6069	San Benito County	CA	6
226	6071	San Bernardino County	CA	6
227	6073	San Diego County	CA	6
228	6075	City and County of San Francisco	CA	6
229	6077	San Joaquin County	CA	6
230	6079	San Luis Obispo County	CA	6
231	6081	San Mateo County	CA	6
232	6083	Santa Barbara County	CA	6
233	6085	Santa Clara County	CA	6
234	6087	Santa Cruz County	CA	6
235	6089	Shasta County	CA	6
236	6091	Sierra County	CA	6
237	6093	Siskiyou County	CA	6
238	6095	Solano County	CA	6
239	6097	Sonoma County	CA	6
240	6099	Stanislaus County	CA	6
241	6101	Sutter County	CA	6
242	6103	Tehama County	CA	6
243	6105	Trinity County	CA	6
244	6107	Tulare County	CA	6
245	6109	Tuolumne County	CA	6
246	6111	Ventura County	CA	6
247	6113	Yolo County	CA	6
248	6115	Yuba County	CA	6

```
     2020-01-22  2020-01-23  2020-01-24  2020-01-25  2020-01-26  2020-
01-27  \
190           0           0           0           0           0
0
191           0           0           0           0           0
0
192           0           0           0           0           0
0
193           0           0           0           0           0
0
194           0           0           0           0           0
0
195           0           0           0           0           0
0
196           0           0           0           0           0
0
197           0           0           0           0           0
0
198           0           0           0           0           0
0
199           0           0           0           0           0
0
200           0           0           0           0           0
0
201           0           0           0           0           0
0
202           0           0           0           0           0
0
203           0           0           0           0           0
0
204           0           0           0           0           0
0
205           0           0           0           0           0
0
206           0           0           0           0           0
0
207           0           0           0           0           0
0
208           0           0           0           0           0
0
209           0           0           0           0           0
0
210           0           0           0           0           0
0
211           0           0           0           0           0
0
212           0           0           0           0           0
0
```

213	0	0	0	0	0
0					
214	0	0	0	0	0
0					
215	0	0	0	0	0
0					
216	0	0	0	0	0
0					
217	0	0	0	0	0
0					
218	0	0	0	0	0
0					
219	0	0	0	0	0
0					
220	0	0	0	0	0
0					
221	0	0	0	0	0
0					
222	0	0	0	0	0
0					
223	0	0	0	0	0
0					
224	0	0	0	0	0
0					
225	0	0	0	0	0
0					
226	0	0	0	0	0
0					
227	0	0	0	0	0
0					
228	0	0	0	0	0
0					
229	0	0	0	0	0
0					
230	0	0	0	0	0
0					
231	0	0	0	0	0
0					
232	0	0	0	0	0
0					
233	0	0	0	0	0
0					
234	0	0	0	0	0
0					
235	0	0	0	0	0
0					
236	0	0	0	0	0
0					
237	0	0	0	0	0

```
0
238           0           0           0           0           0
0
239           0           0           0           0           0
0
240           0           0           0           0           0
0
241           0           0           0           0           0
0
242           0           0           0           0           0
0
243           0           0           0           0           0
0
244           0           0           0           0           0
0
245           0           0           0           0           0
0
246           0           0           0           0           0
0
247           0           0           0           0           0
0
248           0           0           0           0           0
0

     ...  2023-07-14  2023-07-15  2023-07-16  2023-07-17  2023-07-
18  \
190  ...           0           0           0           0           0

191  ...        2172        2172        2172        2172        2172

192  ...           0           0           0           0           0

193  ...         100         100         100         100         100

194  ...         503         503         503         503         503

195  ...         144         144         144         144         144

196  ...          24          24          24          24          24

197  ...        1604        1604        1604        1604        1604

198  ...          62          62          62          62          62

199  ...         248         248         248         248         248

200  ...        3031        3031        3031        3031        3031

201  ...          56          56          56          56          56
```

202	...	172	172	172	172	172
203	...	987	987	987	987	987
204	...	63	63	63	63	63
205	...	2502	2502	2502	2502	2502
206	...	487	487	487	487	487
207	...	165	165	165	165	165
208	...	65	65	65	65	65
209	...	36317	36317	36317	36317	36317
210	...	377	377	377	377	377
211	...	260	260	260	260	260
212	...	32	32	32	32	32
213	...	146	146	146	146	146
214	...	904	904	904	904	904
215	...	11	11	11	11	11
216	...	8	8	8	8	8
217	...	817	817	817	817	817
218	...	182	182	182	182	182
219	...	139	139	139	139	139
220	...	8235	8235	8235	8235	8235
221	...	687	687	687	687	687
222	...	15	15	15	15	15
223	...	6863	6863	6863	6863	6863
224	...	3627	3627	3627	3627	3627
225	...	117	117	117	117	117
226	...	8144	8144	8144	8144	8144
227	...	5900	5900	5900	5900	5900

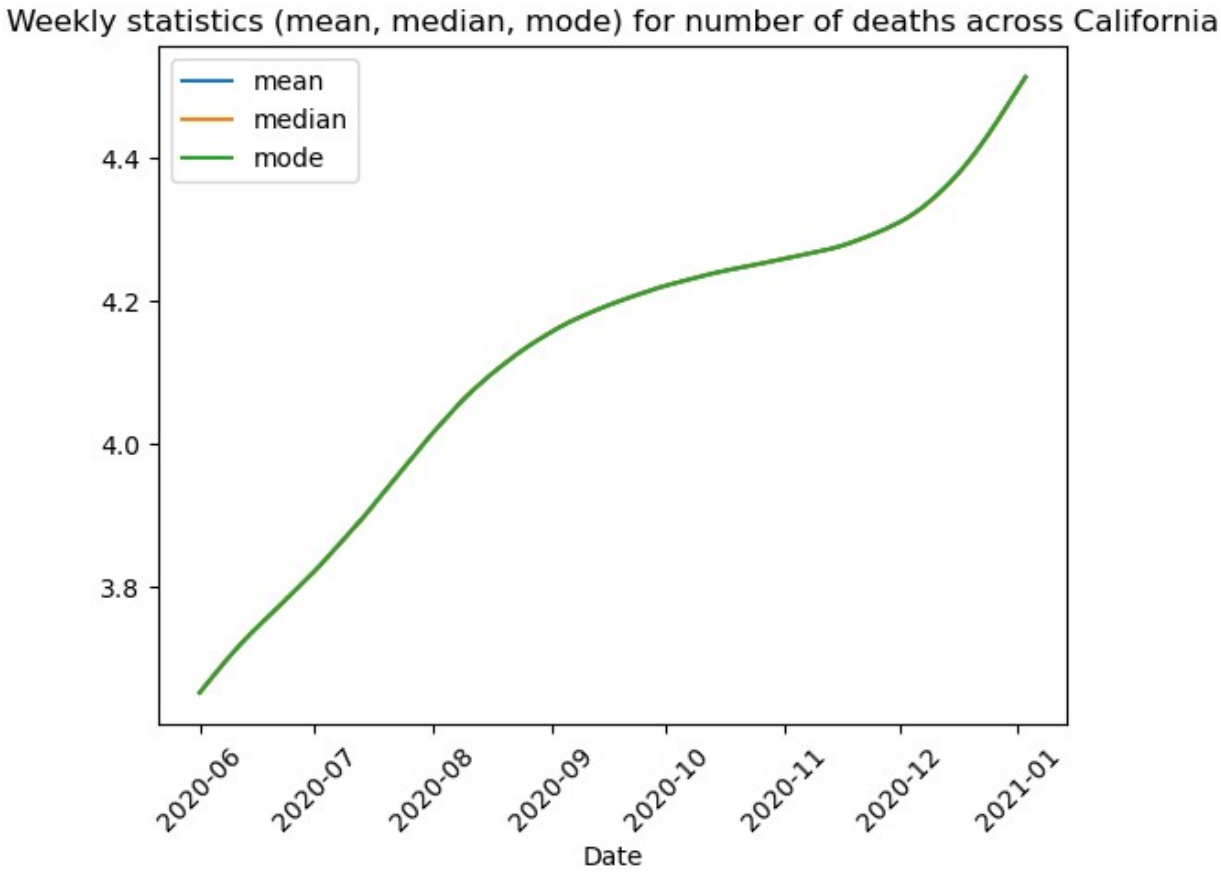
228	...	1235	1235	1235	1235	1235
229	...	2463	2463	2463	2463	2463
230	...	586	586	586	586	586
231	...	747	747	747	747	747
232	...	777	777	777	777	777
233	...	2844	2844	2844	2844	2844
234	...	277	277	277	277	277
235	...	612	612	612	612	612
236	...	5	5	5	5	5
237	...	97	97	97	97	97
238	...	476	476	476	476	476
239	...	578	578	578	578	578
240	...	1869	1869	1869	1869	1869
241	...	242	242	242	242	242
242	...	241	241	241	241	241
243	...	25	25	25	25	25
244	...	1603	1603	1603	1603	1603
245	...	212	212	212	212	212
246	...	1708	1708	1708	1708	1708
247	...	459	459	459	459	459
248	...	134	134	134	134	134

	2023-07-19	2023-07-20	2023-07-21	2023-07-22	2023-07-23
190	0	0	0	0	0
191	2172	2172	2172	2172	2172
192	0	0	0	0	0
193	100	100	100	100	100
194	503	503	503	503	503
195	144	144	144	144	144
196	24	24	24	24	24

197	1604	1604	1604	1604	1604
198	62	62	62	62	62
199	248	248	248	248	248
200	3031	3031	3031	3031	3031
201	56	56	56	56	56
202	172	172	172	172	172
203	987	987	987	987	987
204	63	63	63	63	63
205	2502	2502	2502	2502	2502
206	487	487	487	487	487
207	165	165	165	165	165
208	65	65	65	65	65
209	36317	36317	36317	36317	36317
210	377	377	377	377	377
211	260	260	260	260	260
212	32	32	32	32	32
213	146	146	146	146	146
214	904	904	904	904	904
215	11	11	11	11	11
216	8	8	8	8	8
217	817	817	817	817	817
218	182	182	182	182	182
219	139	139	139	139	139
220	8235	8235	8235	8235	8235
221	687	687	687	687	687
222	15	15	15	15	15
223	6863	6863	6863	6863	6863
224	3627	3627	3627	3627	3627
225	117	117	117	117	117
226	8144	8144	8144	8144	8144
227	5900	5900	5900	5900	5900
228	1235	1235	1235	1235	1235
229	2463	2463	2463	2463	2463
230	586	586	586	586	586
231	747	747	747	747	747
232	777	777	777	777	777
233	2844	2844	2844	2844	2844
234	277	277	277	277	277
235	612	612	612	612	612
236	5	5	5	5	5
237	97	97	97	97	97
238	476	476	476	476	476
239	578	578	578	578	578
240	1869	1869	1869	1869	1869
241	242	242	242	242	242
242	241	241	241	241	241
243	25	25	25	25	25
244	1603	1603	1603	1603	1603
245	212	212	212	212	212
246	1708	1708	1708	1708	1708

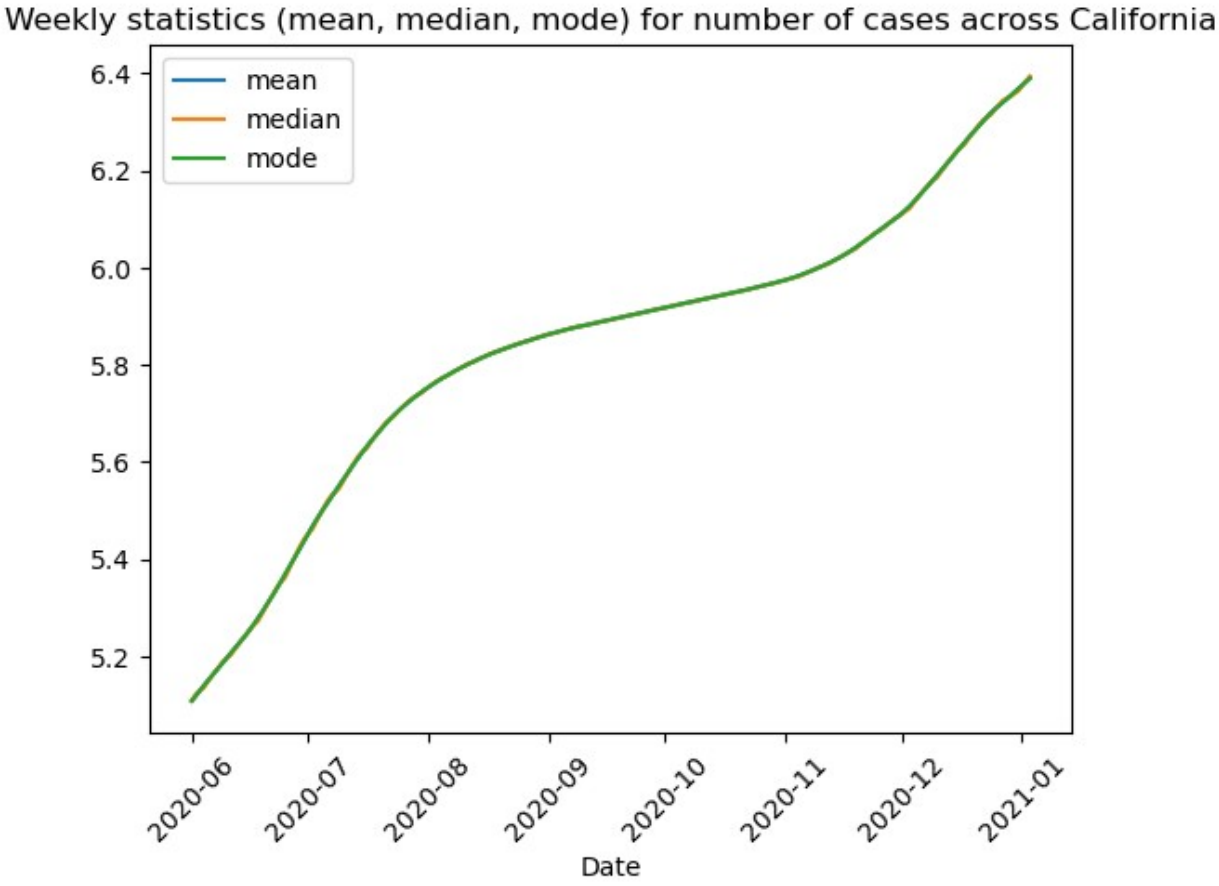
```
247         459         459         459         459         459
248         134         134         134         134         134

[59 rows x 1269 columns]
```

```
def trim_dates(df):
    return df.drop(df.loc[:,:'2020-05-24'],
axis=1).drop(df.loc[:,'2021-01-04':], axis=1)
series = trim_dates(df_deaths[df_deaths['State']=='CA']).sum()

sr_mean = np.round(series.rolling(window=7).mean())
sr_mean.drop(sr_mean.index[0:7], inplace=True)
sr_median = series.rolling(window=7).median()
sr_median.drop(sr_median.index[0:7], inplace=True)
def get_mode(window):
    val_count = window.value_counts()
    mode = val_count.index[val_count == val_count.max()]
    return np.round(stat.fmean(mode.tolist())) if len(mode) > 0 else
mode[0]
sr_mode = series.rolling(window=7).apply(get_mode)
sr_mode.drop(sr_mode.index[0:7], inplace=True)

def add_to_plot(sr, name):
    x = sr.index.values
    x = np.asarray(x, dtype='datetime64[s]')
    y = np.log10(sr.values)
    plt.plot(x, y, label = name)
add_to_plot(sr_mean, 'mean')
add_to_plot(sr_median, 'median')
add_to_plot(sr_mode, 'mode')
plt.title("Weekly statistics (mean, median, mode) for number of deaths
across California")
plt.xlabel("Date")
plt.xticks(rotation=45)
plt.ylabel("")
plt.legend()
plt.show()
```



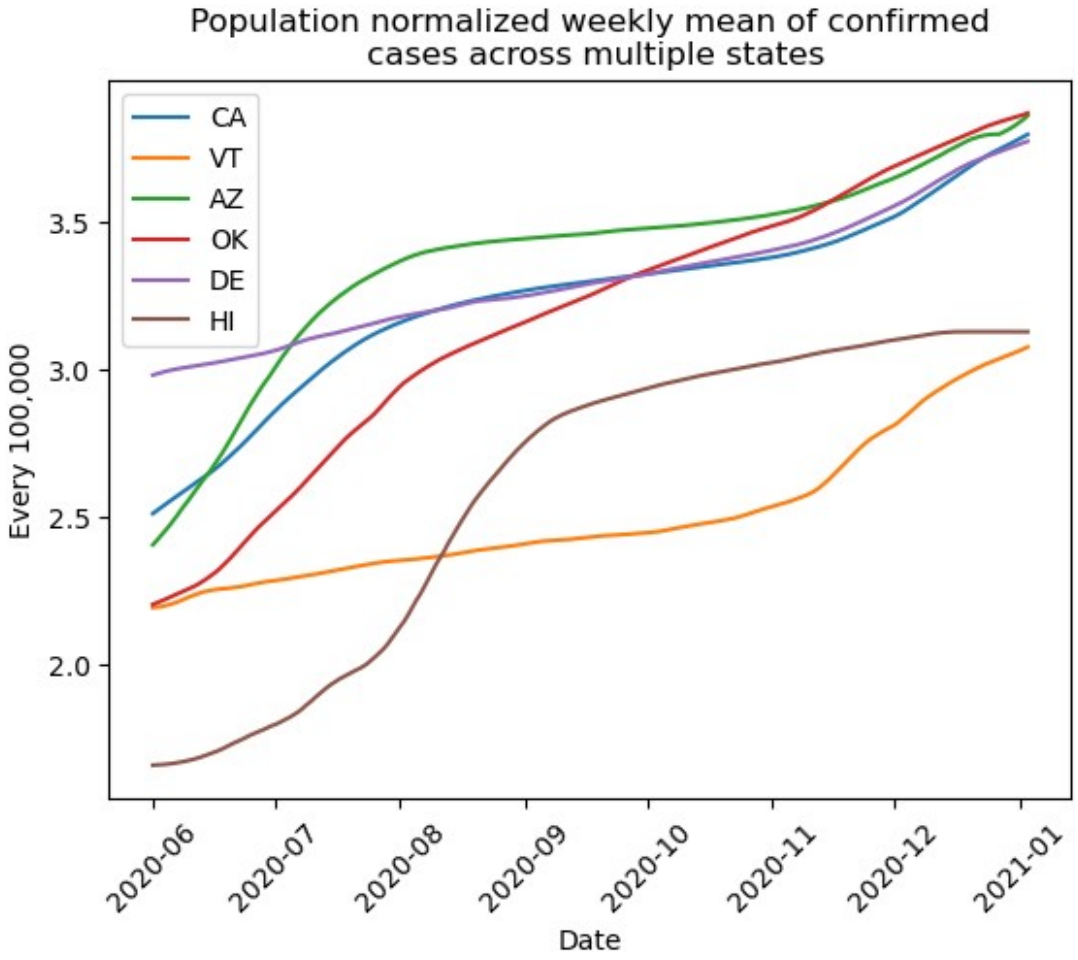

```
series = trim_dates(df_confirmed[df_confirmed['State']=='CA']).sum()
sr_mean = np.round(series.rolling(window=7).mean())
sr_mean.drop(sr_mean.index[0:7], inplace=True)
sr_median = series.rolling(window=7).median()
sr_median.drop(sr_median.index[0:7], inplace=True)
def get_mode(window):
    val_count = window.value_counts()
    mode = val_count.index[val_count == val_count.max()]
    return np.round(stat.fmean(mode.tolist())) if len(mode) > 0 else
mode[0]
sr_mode = series.rolling(window=7).apply(get_mode)
sr_mode.drop(sr_mode.index[0:7], inplace=True)
def add_to_plot(sr, name):
    x = sr.index.values
    x = np.asarray(x, dtype='datetime64[s]')
    y = np.log10(sr.values)
    plt.plot(x, y, label = name)
add_to_plot(sr_mean, 'mean')
add_to_plot(sr_median, 'median')
add_to_plot(sr_mode, 'mode')
plt.title("Weekly statistics (mean, median, mode) for number of cases
across California")
```

```
plt.xlabel("Date")
plt.xticks(rotation=45)
plt.ylabel("")
plt.legend()
plt.show()
```



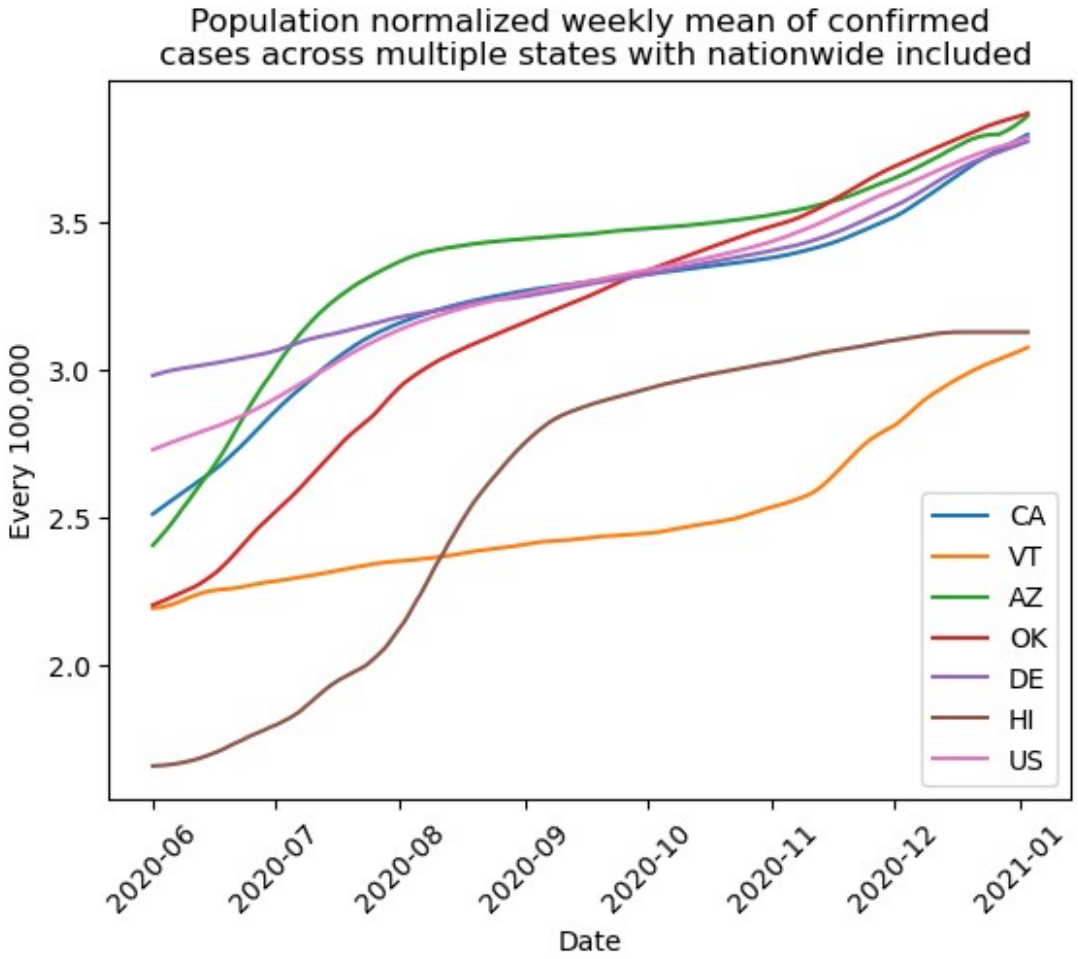

```
def get_pop(state):
    return df_pop[df_pop['State']==state].iloc[:,3].sum()

def add_state_to_plot(*states):
    for state in states:
        sr =
trim_dates(df_confirmed[df_confirmed['State']==state]).sum()
        sr_mean = np.round(sr.rolling(window=7).mean())
        sr_mean.drop(sr_mean.index[0:7], inplace=True)
        x = sr_mean.index.values
        x = np.asarray(x, dtype='datetime64[s]')
        y = np.log10(sr_mean.values / get_pop(state) * 100000)
        plt.plot(x, y, label = state)
add_state_to_plot('CA','VT','AZ','OK','DE','HI')
plt.title("Population normalized weekly mean of confirmed\n cases
across multiple states")
```

```
plt.xlabel("Date")
plt.xticks(rotation=45)
plt.ylabel("Every 100,000")
plt.legend()
plt.show()
```




```
add_state_to_plot('CA','VT','AZ','OK','DE','HI')
df_us = pd.read_csv('data/us_confirmed_mean.csv')
x = df_us.iloc[:, 0].values
x = np.asarray(x, dtype='datetime64[s]')
y = np.log10(df_us.iloc[:, 1].values / df_pop.iloc[:,3].sum() *
100000)
plt.plot(x, y, label = "US")
plt.title("Population normalized weekly mean of confirmed\n cases
across multiple states with nationwide included")
plt.xlabel("Date")
plt.xticks(rotation=45)
plt.ylabel("Every 100,000")
```

```
plt.legend()
plt.show()
```



From the graph we can observe that DE, AZ, and CA mostly followed the nationwide trend of population normalized confirmed cases. The delay in cases gain in AZ, OK, and HI can be attributed to their geographical remoteness, especially to HI. VT being particularly resistant to cases rising can be partially attributed to its remoteness and population sparsity.

```
df_ca_pop = df_pop[df_pop['State'] ==
'CA'].drop([192]).reset_index(drop=True)
df_ca_pop
```

	countyFIPS	County Name	State	population
0	0	Statewide Unallocated	CA	0
1	6001	Alameda County	CA	1671329
2	6003	Alpine County	CA	1129
3	6005	Amador County	CA	39752
4	6007	Butte County	CA	219186
5	6009	Calaveras County	CA	45905
6	6011	Colusa County	CA	21547

7	6013	Contra Costa County	CA	1153526
8	6015	Del Norte County	CA	27812
9	6017	El Dorado County	CA	192843
10	6019	Fresno County	CA	999101
11	6021	Glenn County	CA	28393
12	6023	Humboldt County	CA	135558
13	6025	Imperial County	CA	181215
14	6027	Inyo County	CA	18039
15	6029	Kern County	CA	900202
16	6031	Kings County	CA	152940
17	6033	Lake County	CA	64386
18	6035	Lassen County	CA	30573
19	6037	Los Angeles County	CA	10039107
20	6039	Madera County	CA	157327
21	6041	Marin County	CA	258826
22	6043	Mariposa County	CA	17203
23	6045	Mendocino County	CA	86749
24	6047	Merced County	CA	277680
25	6049	Modoc County	CA	8841
26	6051	Mono County	CA	14444
27	6053	Monterey County	CA	434061
28	6055	Napa County	CA	137744
29	6057	Nevada County	CA	99755
30	6059	Orange County	CA	3175692
31	6061	Placer County	CA	398329
32	6063	Plumas County	CA	18807
33	6065	Riverside County	CA	2470546
34	6067	Sacramento County	CA	1552058
35	6069	San Benito County	CA	62808
36	6071	San Bernardino County	CA	2180085
37	6073	San Diego County	CA	3338330
38	6075	San Francisco County	CA	881549
39	6077	San Joaquin County	CA	762148
40	6079	San Luis Obispo County	CA	283111
41	6081	San Mateo County	CA	766573
42	6083	Santa Barbara County	CA	446499
43	6085	Santa Clara County	CA	1927852
44	6087	Santa Cruz County	CA	273213
45	6089	Shasta County	CA	180080
46	6091	Sierra County	CA	3005
47	6093	Siskiyou County	CA	43539
48	6095	Solano County	CA	447643
49	6097	Sonoma County	CA	494336
50	6099	Stanislaus County	CA	550660
51	6101	Sutter County	CA	96971
52	6103	Tehama County	CA	65084
53	6105	Trinity County	CA	12285
54	6107	Tulare County	CA	466195
55	6109	Tuolumne County	CA	54478

```
56        6111          Ventura County    CA      846006
57        6113             Yolo County    CA      220500
58        6115             Yuba County    CA       78668
```

```
df_ca = df_confirmed[df_confirmed['State'] == 'CA']
df_ca = df_ca[['County Name', '2021-01-03']].reset_index(drop=True)
df_ca_pop = df_pop[df_pop['State'] ==
'CA'].drop([192]).reset_index(drop=True)
df_norm_counties = pd.DataFrame({'County Name': df_ca['County Name'],
'Normalized': df_ca['2021-01-03']/df_pop['population']}, index =
df_ca.index)
df_norm_counties.sort_values('Normalized', ascending=False,
inplace=True)
fig = go.Figure(

data=[go.Bar(y=df_norm_counties['Normalized'].iloc[1:6],x=df_norm_coun
ties['County Name'].iloc[1:6])],
    layout_title_text="Population normalized of top 5 counties as of
2021-01-03"
)
fig.show()
```

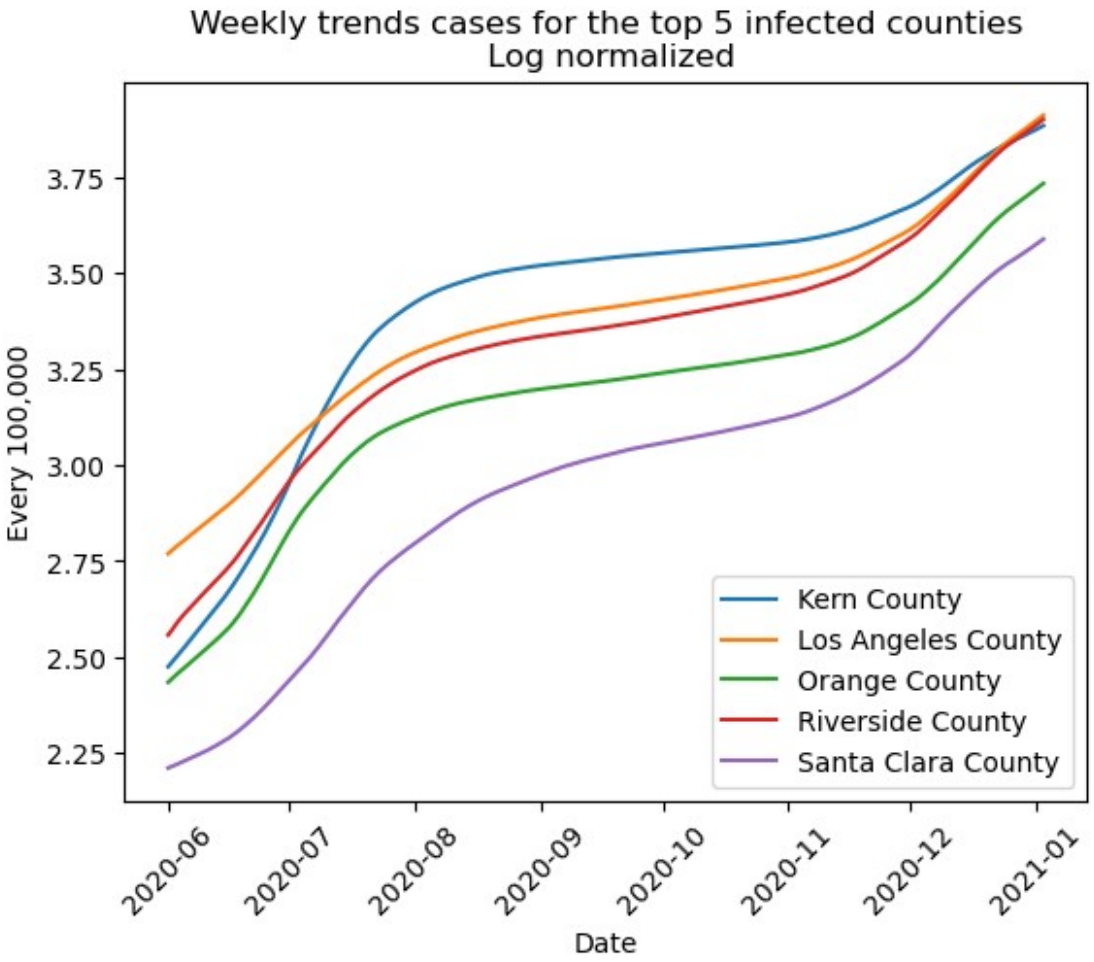

```
{"config":{"plotlyServerURL":"https://plot.ly"},"data":
[{"type":"bar","x":["Los Angeles County ","Riverside County ","Santa
Clara County ","Orange County ","Kern County "],"y":
[80.51195723529963,14.059108593270084,8.063849475632326,5.759230916395
638,4.784775318578135]}],"layout":{"template":{"data":{"candlestick":
[{"decreasing":{"line":{"color":"#000033"}},"increasing":{"line":
{"color":"#000032"}},"type":"candlestick"}],"contour":[{"colorscale":
[[0.0,"#000011"],[0.1111111111111111,"#000012"],
[0.2222222222222222,"#000013"],[0.3333333333333333,"#000014"],
[0.4444444444444444,"#000015"],[0.5555555555555556,"#000016"],
[0.6666666666666666,"#000017"],[0.7777777777777778,"#000018"],
[0.8888888888888888,"#000019"],
[1.0,"#000020"]],"type":"contour"}],"contourcarpet":[{"colorscale":
[[0.0,"#000011"],[0.1111111111111111,"#000012"],
[0.2222222222222222,"#000013"],[0.3333333333333333,"#000014"],
[0.4444444444444444,"#000015"],[0.5555555555555556,"#000016"],
[0.6666666666666666,"#000017"],[0.7777777777777778,"#000018"],
[0.8888888888888888,"#000019"],
[1.0,"#000020"]],"type":"contourcarpet"}],"heatmap":[{"colorscale":
[[0.0,"#000011"],[0.1111111111111111,"#000012"],
[0.2222222222222222,"#000013"],[0.3333333333333333,"#000014"],
[0.4444444444444444,"#000015"],[0.5555555555555556,"#000016"],
[0.6666666666666666,"#000017"],[0.7777777777777778,"#000018"],
[0.8888888888888888,"#000019"],
[1.0,"#000020"]],"type":"heatmap"}],"histogram2d":[{"colorscale":
[[0.0,"#000011"],[0.1111111111111111,"#000012"],
[0.2222222222222222,"#000013"],[0.3333333333333333,"#000014"],
[0.4444444444444444,"#000015"],[0.5555555555555556,"#000016"],
```

```
[0.6666666666666666,"#000017"],[0.7777777777777778,"#000018"],
[0.8888888888888888,"#000019"],
[1.0,"#000020"]],"type":"histogram2d"}],"icicle":[{"textfont":
{"color":"white"},"type":"icicle"}],"sankey":[{"textfont":
{"color":"#000036"},"type":"sankey"}],"scatter":[{"marker":{"line":
{"width":0}},"type":"scatter"}],"table":[{"cells":{"fill":
{"color":"#000038"},"font":{"color":"#000037"},"line":
{"color":"#000039"}},"header":{"fill":{"color":"#000040"},"font":
{"color":"#000036"},"line":
{"color":"#000039"}},"type":"table"}],"waterfall":[{"connector":
{"line":{"color":"#000036","width":2}},"decreasing":{"marker":
{"color":"#000033"}},"increasing":{"marker":
{"color":"#000032"}},"totals":{"marker":
{"color":"#000034"}},"type":"waterfall"}]},"layout":{"coloraxis":
{"colorscale":[[0.0,"#000011"],[0.1111111111111111,"#000012"],
[0.2222222222222222,"#000013"],[0.3333333333333333,"#000014"],
[0.4444444444444444,"#000015"],[0.5555555555555556,"#000016"],
[0.6666666666666666,"#000017"],[0.7777777777777778,"#000018"],
[0.8888888888888888,"#000019"],[1.0,"#000020"]]},"colorscale":
{"diverging":[[0.0,"#000021"],[0.1,"#000022"],[0.2,"#000023"],
[0.3,"#000024"],[0.4,"#000025"],[0.5,"#000026"],[0.6,"#000027"],
[0.7,"#000028"],[0.8,"#000029"],[0.9,"#000030"],
[1.0,"#000031"]],"sequential":[[0.0,"#000011"],
[0.1111111111111111,"#000012"],[0.2222222222222222,"#000013"],
[0.3333333333333333,"#000014"],[0.4444444444444444,"#000015"],
[0.5555555555555556,"#000016"],[0.6666666666666666,"#000017"],
[0.7777777777777778,"#000018"],[0.8888888888888888,"#000019"],
[1.0,"#000020"]],"sequentialminus":[[0.0,"#000011"],
[0.1111111111111111,"#000012"],[0.2222222222222222,"#000013"],
[0.3333333333333333,"#000014"],[0.4444444444444444,"#000015"],
[0.5555555555555556,"#000016"],[0.6666666666666666,"#000017"],
[0.7777777777777778,"#000018"],[0.8888888888888888,"#000019"],
[1.0,"#000020"]]},"colorway":
["#000001","#000002","#000003","#000004","#000005","#000006","#000007"
,"#000008","#000009","#000010"]}},"title":{"text":"Population
normalized of top 5 counties as of 2021-01-03"}}}

df_ca = df_deaths[df_deaths['State'] == 'CA']
df_ca = df_ca[['County Name', '2021-01-03']].reset_index(drop=True)
df_ca_pop = df_pop[df_pop['State'] ==
'CA'].drop([192]).reset_index(drop=True)
df_norm_counties = pd.DataFrame({'County Name': df_ca['County Name'],
'Normalized': df_ca['2021-01-03']/df_pop['population']}, index =
df_ca.index)
df_norm_counties.sort_values('Normalized', ascending=False,
inplace=True)
fig = go.Figure(

data=[go.Bar(y=df_norm_counties['Normalized'].iloc[0:5],x=df_norm_coun
ties['County Name'].iloc[0:5])],
```

```
    layout_title_text="Population normalized deaths of top 5 counties
as of 2021-01-03"
)
fig.show()
```


```
{"config":{"plotlyServerURL":"https://plot.ly"},"data":
[{"type":"bar","x":["Los Angeles County ","Riverside County ","Santa
Clara County ","Orange County ","Sacramento County "],"y":
[1.2191690893744724,0.1740495529315405,9.551717047090273e-
2,8.956699190102672e-2,6.271432723045627e-2]}],"layout":{"template":
{"data":{"candlestick":[{"decreasing":{"line":
{"color":"#000033"}},"increasing":{"line":
{"color":"#000032"}},"type":"candlestick"}],"contour":[{"colorscale":
[[0.0,"#000011"],[0.1111111111111111,"#000012"],
[0.2222222222222222,"#000013"],[0.3333333333333333,"#000014"],
[0.4444444444444444,"#000015"],[0.5555555555555556,"#000016"],
[0.6666666666666666,"#000017"],[0.7777777777777778,"#000018"],
[0.8888888888888888,"#000019"],
[1.0,"#000020"]],"type":"contour"}],"contourcarpet":[{"colorscale":
[[0.0,"#000011"],[0.1111111111111111,"#000012"],
[0.2222222222222222,"#000013"],[0.3333333333333333,"#000014"],
[0.4444444444444444,"#000015"],[0.5555555555555556,"#000016"],
[0.6666666666666666,"#000017"],[0.7777777777777778,"#000018"],
[0.8888888888888888,"#000019"],
[1.0,"#000020"]],"type":"contourcarpet"}],"heatmap":[{"colorscale":
[[0.0,"#000011"],[0.1111111111111111,"#000012"],
[0.2222222222222222,"#000013"],[0.3333333333333333,"#000014"],
[0.4444444444444444,"#000015"],[0.5555555555555556,"#000016"],
[0.6666666666666666,"#000017"],[0.7777777777777778,"#000018"],
[0.8888888888888888,"#000019"],
[1.0,"#000020"]],"type":"heatmap"}],"histogram2d":[{"colorscale":
[[0.0,"#000011"],[0.1111111111111111,"#000012"],
[0.2222222222222222,"#000013"],[0.3333333333333333,"#000014"],
[0.4444444444444444,"#000015"],[0.5555555555555556,"#000016"],
[0.6666666666666666,"#000017"],[0.7777777777777778,"#000018"],
[0.8888888888888888,"#000019"],
[1.0,"#000020"]],"type":"histogram2d"}],"icicle":[{"textfont":
{"color":"white"},"type":"icicle"}],"sankey":[{"textfont":
{"color":"#000036"},"type":"sankey"}],"scatter":[{"marker":{"line":
{"width":0}},"type":"scatter"}],"table":[{"cells":{"fill":
{"color":"#000038"},"font":{"color":"#000037"},"line":
{"color":"#000039"}},"header":{"fill":{"color":"#000040"},"font":
{"color":"#000036"},"line":
{"color":"#000039"}},"type":"table"}],"waterfall":[{"connector":
{"line":{"color":"#000036","width":2}},"decreasing":{"marker":
{"color":"#000033"}},"increasing":{"marker":
{"color":"#000032"}},"totals":{"marker":
{"color":"#000034"}},"type":"waterfall"}]},"layout":{"coloraxis":
{"colorscale":[[0.0,"#000011"],[0.1111111111111111,"#000012"],
[0.2222222222222222,"#000013"],[0.3333333333333333,"#000014"],
```

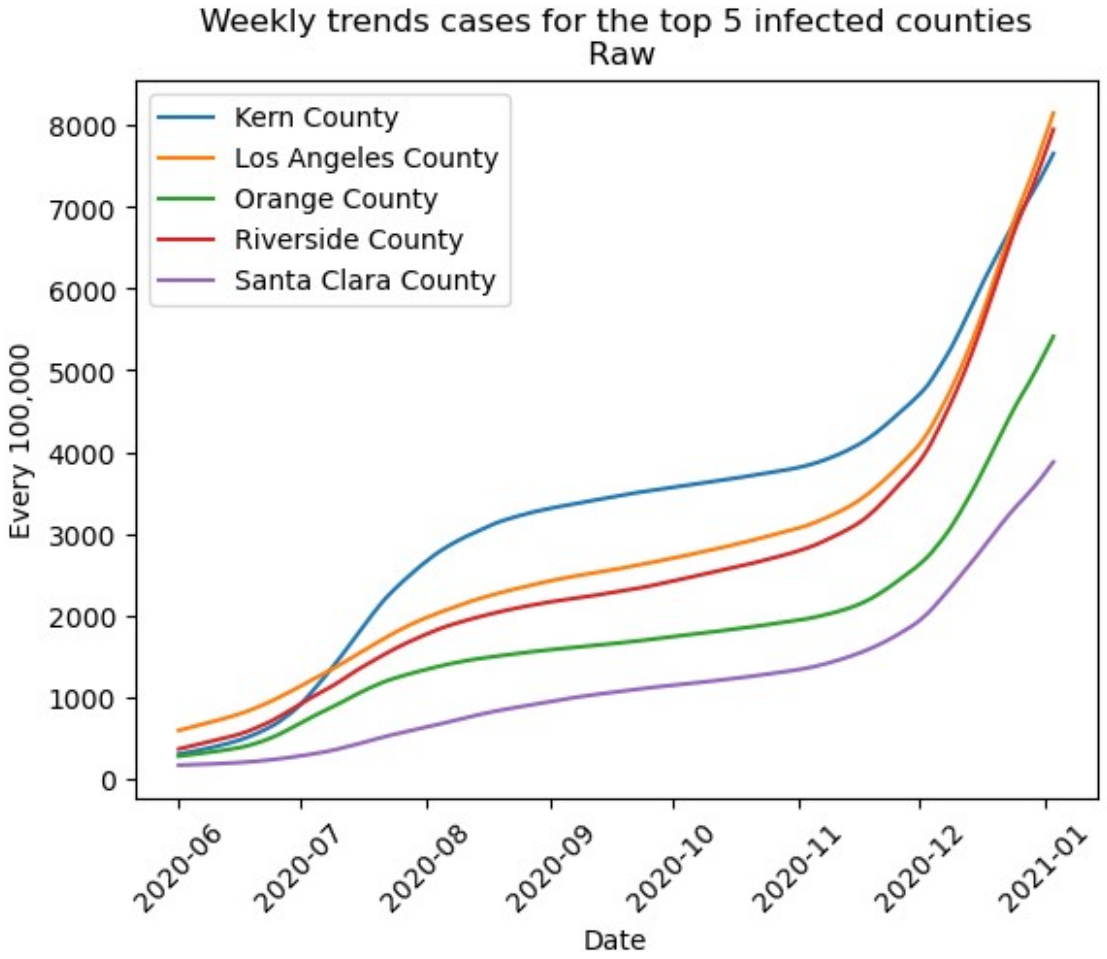
```
[0.4444444444444444,"#000015"],[0.5555555555555556,"#000016"],
[0.6666666666666666,"#000017"],[0.7777777777777778,"#000018"],
[0.8888888888888888,"#000019"],[1.0,"#000020"]]},"colorscale":
{"diverging":[[0.0,"#000021"],[0.1,"#000022"],[0.2,"#000023"],
[0.3,"#000024"],[0.4,"#000025"],[0.5,"#000026"],[0.6,"#000027"],
[0.7,"#000028"],[0.8,"#000029"],[0.9,"#000030"],
[1.0,"#000031"]],"sequential":[[0.0,"#000011"],
[0.1111111111111111,"#000012"],[0.2222222222222222,"#000013"],
[0.3333333333333333,"#000014"],[0.4444444444444444,"#000015"],
[0.5555555555555556,"#000016"],[0.6666666666666666,"#000017"],
[0.7777777777777778,"#000018"],[0.8888888888888888,"#000019"],
[1.0,"#000020"]],"sequentialminus":[[0.0,"#000011"],
[0.1111111111111111,"#000012"],[0.2222222222222222,"#000013"],
[0.3333333333333333,"#000014"],[0.4444444444444444,"#000015"],
[0.5555555555555556,"#000016"],[0.6666666666666666,"#000017"],
[0.7777777777777778,"#000018"],[0.8888888888888888,"#000019"],
[1.0,"#000020"]]},"colorway":
["#000001","#000002","#000003","#000004","#000005","#000006","#000007"
,"#000008","#000009","#000010"]}},"title":{"text":"Population
normalized deaths of top 5 counties as of 2021-01-03"}}}

df_confirmed = pd.read_csv('data/covid_confirmed_usafacts.csv')

def get_pop_county(county_name):
    return df_pop[df_pop['County
Name']==county_name].iloc[:,3].values[0]

def add_county_to_plot(*counties):
    for county in counties:
        county_name = df_pop[df_pop['countyFIPS']==county].iloc[0,1]
        sr =
trim_dates(df_confirmed.loc[df_confirmed['countyFIPS']==county,:]).sum
()
        sr_mean = np.round(sr.rolling(window=7).mean())
        sr_mean.drop(sr_mean.index[0:7], inplace=True)
        x = sr_mean.index.values
        x = np.asarray(x, dtype='datetime64[s]')
        y = np.log10(sr_mean.values / get_pop_county(county_name) *
100000)
        plt.plot(x, y, label = county_name)
add_county_to_plot(6029,6037,6059,6065,6085)
plt.title("Weekly trends cases for the top 5 infected counties\n Log
normalized")
plt.xlabel("Date")
plt.xticks(rotation=45)
plt.ylabel("Every 100,000")
plt.legend()
plt.show()
```



Had to use countyFIPS here. pandas refuse to give me any return if I search with county name directly.

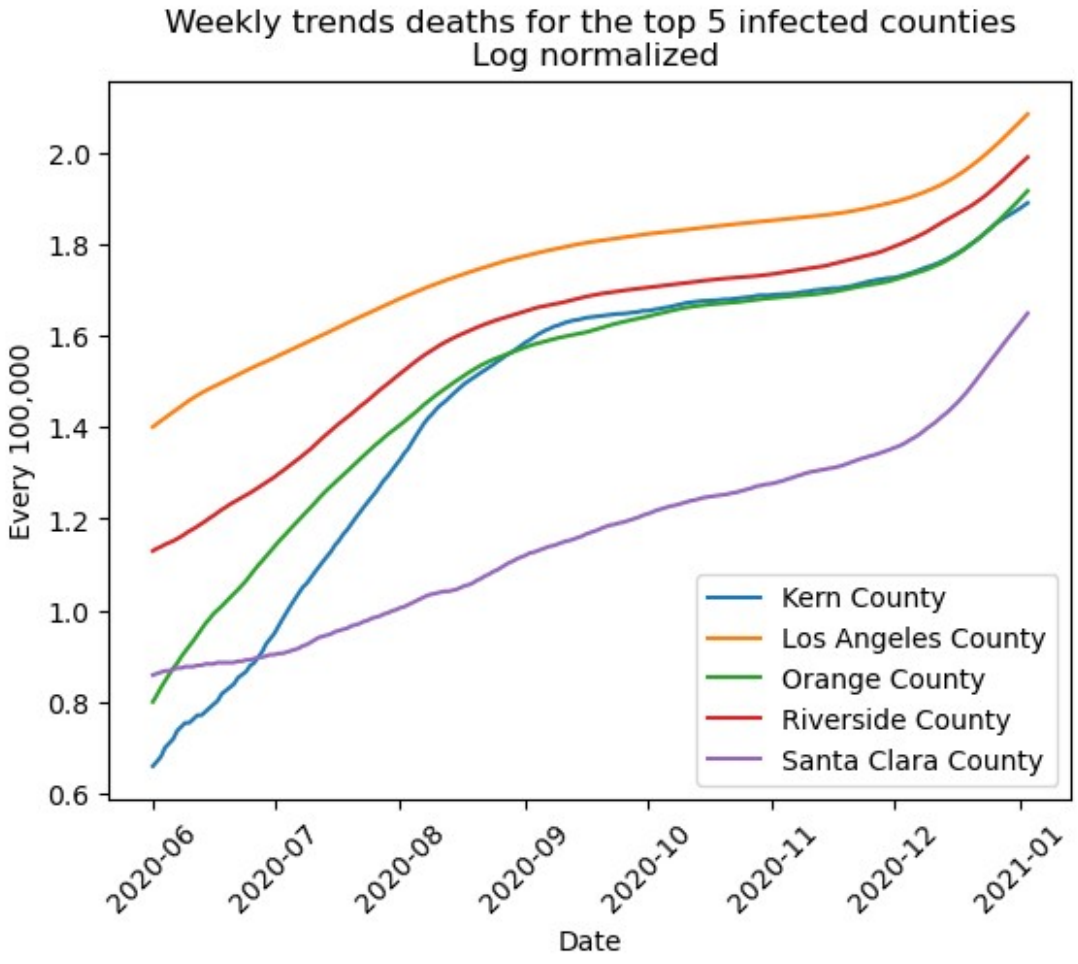
```
df_confirmed = pd.read_csv('data/covid_confirmed_usafacts.csv')

def get_pop_county(county_name):
    return df_pop[df_pop['County
Name']==county_name].iloc[:,3].values[0]

def add_county_to_plot(*counties):
    for county in counties:
        county_name = df_pop[df_pop['countyFIPS']==county].iloc[0,1]
        sr =
trim_dates(df_confirmed.loc[df_confirmed['countyFIPS']==county,:]).sum
()
        sr_mean = np.round(sr.rolling(window=7).mean())
        sr_mean.drop(sr_mean.index[0:7], inplace=True)
        x = sr_mean.index.values
        x = np.asarray(x, dtype='datetime64[s]')
        y = sr_mean.values / get_pop_county(county_name) * 100000
```

```
        plt.plot(x, y, label = county_name)
add_county_to_plot(6029,6037,6059,6065,6085)
plt.title("Weekly trends cases for the top 5 infected counties\n Raw")
plt.xlabel("Date")
plt.xticks(rotation=45)
plt.ylabel("Every 100,000")
plt.legend()
plt.show()
```



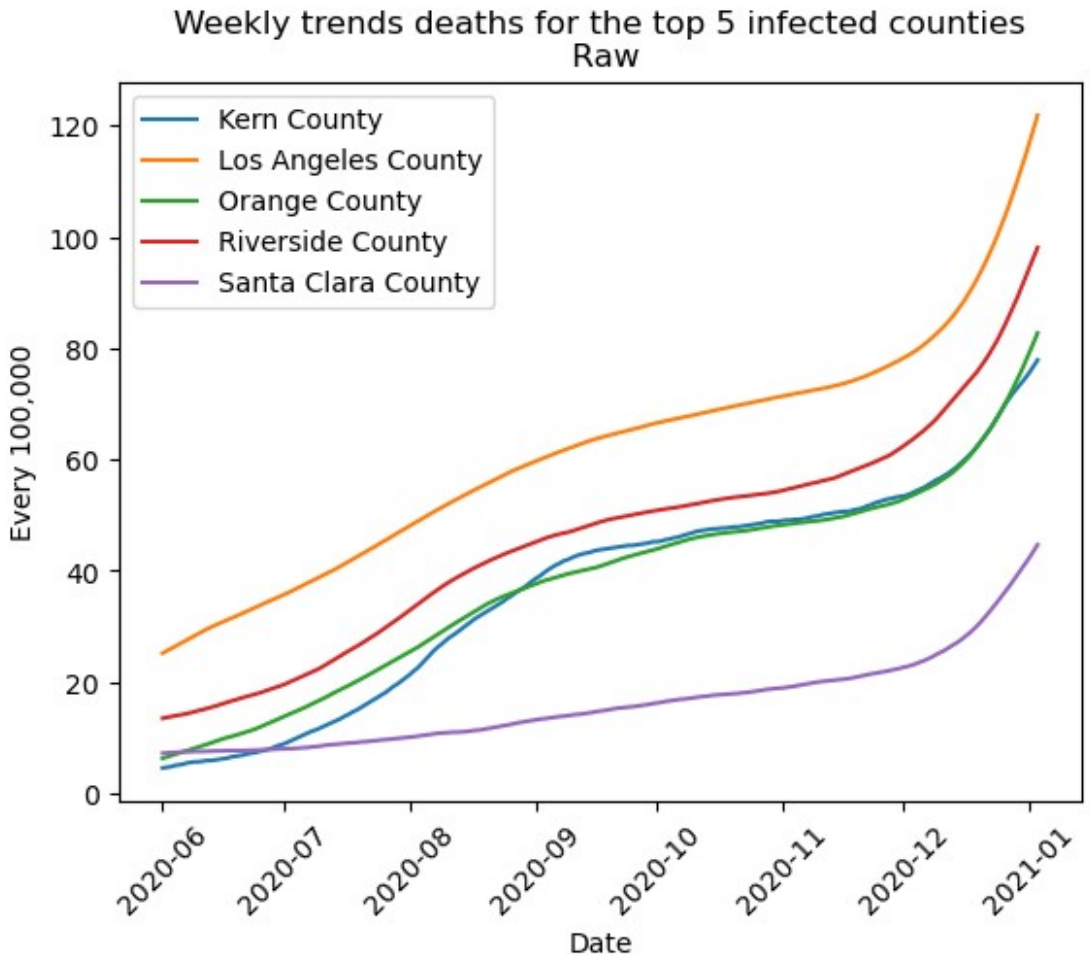

```
df_deaths = pd.read_csv('data/covid_deaths_usafacts.csv')

def get_pop_county(county_name):
    return df_pop[df_pop['County
Name']==county_name].iloc[:,3].values[0]

def add_county_to_plot(*counties):
    for county in counties:
        county_name = df_pop[df_pop['countyFIPS']==county].iloc[0,1]
        sr =
```

```
trim_dates(df_deaths.loc[df_deaths['countyFIPS']==county,:]).sum()
        sr_mean = np.round(sr.rolling(window=7).mean())
        sr_mean.drop(sr_mean.index[0:7], inplace=True)
        x = sr_mean.index.values
        x = np.asarray(x, dtype='datetime64[s]')
        y = np.log10(sr_mean.values / get_pop_county(county_name) *
100000)
        plt.plot(x, y, label = county_name)
add_county_to_plot(6029,6037,6059,6065,6085)
plt.title("Weekly trends deaths for the top 5 infected counties\n Log
normalized")
plt.xlabel("Date")
plt.xticks(rotation=45)
plt.ylabel("Every 100,000")
plt.legend()
plt.show()
```



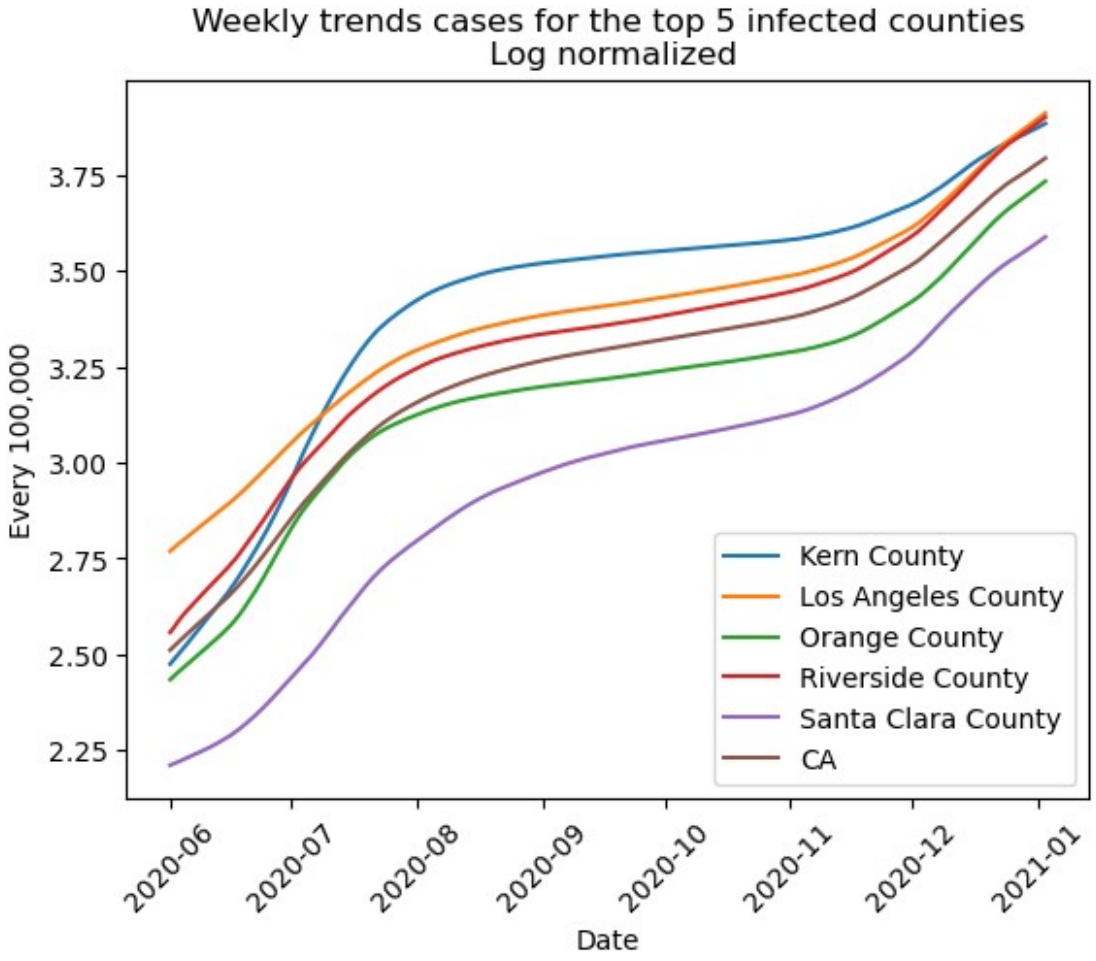

```
df_deaths = pd.read_csv('data/covid_deaths_usafacts.csv')
```

```
def get_pop_county(county_name):
    return df_pop[df_pop['County
Name']==county_name].iloc[:,3].values[0]

def add_county_to_plot(*counties):
    for county in counties:
        county_name = df_pop[df_pop['countyFIPS']==county].iloc[0,1]
        sr =
trim_dates(df_deaths.loc[df_deaths['countyFIPS']==county,:]).sum()
        sr_mean = np.round(sr.rolling(window=7).mean())
        sr_mean.drop(sr_mean.index[0:7], inplace=True)
        x = sr_mean.index.values
        x = np.asarray(x, dtype='datetime64[s]')
        y = sr_mean.values / get_pop_county(county_name) * 100000
        plt.plot(x, y, label = county_name)
add_county_to_plot(6029,6037,6059,6065,6085)
plt.title("Weekly trends deaths for the top 5 infected counties\n
Raw")
plt.xlabel("Date")
plt.xticks(rotation=45)
plt.ylabel("Every 100,000")
plt.legend()
plt.show()
```




```
df_confirmed = pd.read_csv('data/covid_confirmed_usafacts.csv')

def get_pop_county(county_name):
    return df_pop[df_pop['County
Name']==county_name].iloc[:,3].values[0]

def add_county_to_plot(*counties):
    for county in counties:
        county_name = df_pop[df_pop['countyFIPS']==county].iloc[0,1]
        sr =
trim_dates(df_confirmed.loc[df_confirmed['countyFIPS']==county,:]).sum
()
        sr_mean = np.round(sr.rolling(window=7).mean())
        sr_mean.drop(sr_mean.index[0:7], inplace=True)
        x = sr_mean.index.values
        x = np.asarray(x, dtype='datetime64[s]')
        y = np.log10(sr_mean.values / get_pop_county(county_name) *
100000)
        plt.plot(x, y, label = county_name)
add_county_to_plot(6029,6037,6059,6065,6085)
```

```
add_state_to_plot('CA')
plt.title("Weekly trends cases for the top 5 infected counties\n Log
normalized")
plt.xlabel("Date")
plt.xticks(rotation=45)
plt.ylabel("Every 100,000")
plt.legend()
plt.show()
```



From the plot I can identify two periods of sharp rises in case, one being first week of July and the other is middle of December. It may be possible to attribute the first rise to high heat in Summer and increased outdoor activities. The second rise in December may be attributed to increased travel during holiday season. And lastly, yes the counties somewhat follows state pattern with the exception of Kern County.