In [82]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly
import random
from gcmap import GCMapper, Gradient
from matplotlib.colors import Normalize, LinearSegmentedColormap, PowerNorm
from mpl_toolkits.basemap import Basemap
```

In [83]:

```
# Prints all of dataframe (may take a while)
pd.options.display.max_columns = None
pd.options.display.max_rows = None
```

In [84]:

```
# Read in Data for Domestic Flights
flight_data = pd.read_csv("Domestic_Flights_MarApr2019.csv")
flight_data.head()
```

Out[84]:

	PASSENGERS	DISTANCE	ORIGIN_CITY_NAME	ORIGIN_STATE_ABR	ORIGIN_STATE_NM	DEST_C
0	1	134	Birmingham, AL	AL	Alabama	
1	1	826	Birmingham, AL	AL	Alabama	Ha
2	1	822	Birmingham, AL	AL	Alabama	Ro
3	2	906	Birmingham, AL	AL	Alabama	Midlaı
4	2	744	Birmingham, AL	AL	Alabama	San

In [85]:

```
def flights(numberOfPassengers, dailyAvgNOP, totalDailyPassengers, origin_s

    for index, row in flight_data.iterrows():

        stateIndex = 0
        while(stateIndex < 100):

            state = origin_state[int(stateIndex/2)]
            if((row["ORIGIN_STATE_ABR"] == state or row["DEST_STATE_ABR"] =
                numberOfPassengers[stateIndex] += row["PASSENGERS"]
            stateIndex += 1
            if((row["ORIGIN_STATE_ABR"] == state or row["DEST_STATE_ABR"] =
                numberOfPassengers[stateIndex] += row["PASSENGERS"]
            stateIndex += 1


    #Average Daily Passengers leaving/entering each state (31 days in March
    for i in range(len(numberOfPassengers)):
        if (i % 2 == 0):
            dailyAvgNOP[i] = numberOfPassengers[i]/31
        if (i % 2 != 0):
            dailyAvgNOP[i] = numberOfPassengers[i]/30

    #Total Number of Passengers travelling each day (March and April)
    for i in range(len(numberOfPassengers)):
        if (i % 2 == 0):
            totalDailyPassengers[0] += dailyAvgNOP[i]
        if (i % 2 != 0):
            totalDailyPassengers[1] += dailyAvgNOP[i]

    return(numberOfPassengers, dailyAvgNOP, totalDailyPassengers)
```

In [107]:

```
US_Infection = pd.read_csv("us-states.csv")
US_Infection
```

Out[107]:

	Date	State	cases	Population	PercentInfected
0	3/19/20	AL	78	4903185	0.000016
1	3/19/20	AK	12	731545	0.000016
2	3/19/20	AZ	47	7278717	0.000006
3	3/19/20	AR	62	3017804	0.000021
4	3/19/20	CA	1067	39512223	0.000027
5	3/19/20	CO	278	5758736	0.000048
6	3/19/20	CT	159	3565287	0.000045
7	3/19/20	DE	30	973764	0.000031
8	3/19/20	FL	434	21477737	0.000020
9	3/19/20	GA	282	10617423	0.000027
10	3/19/20	HI	26	1415872	0.000018

In [109]:

```
# Drop row for Deleware
US_Infection.drop(US_Infection[US_Infection['State'] == "DE"].index, inplac
US_Infection.reset_index(inplace=True)
US_Infection.drop(['index'], axis=1,inplace=True)
US_Infection
```

Out[109]:

	Date	State	cases	Population	PercentInfected
0	3/19/20	AL	78	4903185	0.000016
1	3/19/20	AK	12	731545	0.000016
2	3/19/20	AZ	47	7278717	0.000006
3	3/19/20	AR	62	3017804	0.000021
4	3/19/20	CA	1067	39512223	0.000027
5	3/19/20	CO	278	5758736	0.000048
6	3/19/20	CT	159	3565287	0.000045
7	3/19/20	FL	434	21477737	0.000020
8	3/19/20	GA	282	10617423	0.000027
9	3/19/20	HI	26	1415872	0.000018
10	3/19/20	ID	23	1787065	0.000013

In [87]:

```
# Using Domestic Flight Data, calculate number of passengers flying betweer
def detailedFlights(states):

    # Dataframes storing daily number of passengers flying between each sta
    detailed_flightsMarch = pd.DataFrame(0, index = states, columns = state
    detailed_flightsApril = pd.DataFrame(0, index = states, columns = state

    for index, row in flight_data.iterrows():
            for i in range(50):
                if(row["ORIGIN_STATE_ABR"] == states[i] and row["MONTH"] ==
                    for j in range(50):
                        if(row["DEST_STATE_ABR"] == states[j]):
                            detailed_flightsMarch.iat[i,j] += row["PASSENGE

            for i in range(50):
                if(row["ORIGIN_STATE_ABR"] == states[i] and row["MONTH"] ==
                    for j in range(50):
                        if(row["DEST_STATE_ABR"] == states[j]):
                            detailed_flightsApril.iat[i,j] += row["PASSENGE

    return(detailed_flightsMarch, detailed_flightsApril)
```

```
In [111]: def simulation(detailed_flightsMarch, states):

              #Initialize key variables
              percentInfected = US_Infection['PercentInfected']
              state_Populations = US_Infection['Population']
              state_Infections = US_Infection['cases']

              #Store the initial number of infections in eac state
              original_StateInfections = state_Infections

              #Create a dictionary that will store the infected travelers for each da
              infectedTravelers_Collection = {}

              # Infected Travelers = daily passengers flying between each state * per
              # is infected
              infectedTravelers = detailed_flightsMarch.multiply(percentInfected.sque

              # update number of infected citizens of each state
              for i in range(13):

                  #Add the number of infected people leavijng the state from state in
                  state_Infections = -1*state_Infections.rsub(infectedTravelers.sum(a

                  #Add the number of infected people entering the state to state infe
                  state_Infections = state_Infections.add(infectedTravelers.sum(axis=

                  percentNotInfected = 1 - (state_Infections/state_Populations)

                  # Each person is likely to infect between 2 and 2.5 people
                  state_Infections = state_Infections * random.uniform(2.0,2.5) * per


                  percentInfected = state_Infections/state_Populations

                  infectedTravelers_Collection[i] = pd.DataFrame(infectedTravelers.sq
                  infectedTravelers = detailed_flightsMarch.multiply(percentInfected.


              # transform into arrays so that they can later be stored into dataframe
              Original_StateInfections = original_StateInfections.squeeze().values
              state_Infections = state_Infections.squeeze().values

              # initialize and fill indataframe where we will store initial and final
              original_StateInfectionsDF = pd.DataFrame(0, index = states, columns = 
              state_InfectionsDF = pd.DataFrame(0, index = states, columns = ["# of I

              for i in range(49):
                  original_StateInfectionsDF.iat[i,0] = original_StateInfections[i]
                  state_InfectionsDF.iat[i,0] = state_Infections[i]

              # change in infected citizens over the course of the 13 days
              changeInInfectedCitizens = original_StateInfectionsDF.rsub(state_Infect

              changeInInfectedCitizens.columns = ['Change in # of Infected Citizens']
              state_InfectionsDF.columns = ['# of Infected Citizens March 31st']
              original_StateInfectionsDF.columns = ['# of Infected Citizens March 19t
```

```
    return(changeInInfectedCitizens, state_InfectionsDF, original_StateInfe
```

In [89]:

```
#Read in Data for Stay-at-Home Order Dates+Times
stayAtHome_Dates = pd.read_csv("StayAtHome_Dates.csv")
stayAtHome_Dates
```

Out[89]:

	States	State_Number	StayAtHome_Date	DaysAfterFirst	Time
0	AL	0	4/4/20	16.0	17.0
1	AK	2	3/28/20	9.0	17.0
2	AZ	4	3/31/20	12.0	17.0
3	AR	6	NaN	NaN	NaN
4	CA	8	3/19/20	0.0	0.0
5	CO	10	3/26/20	7.0	6.0
6	CT	12	3/23/20	4.0	20.0
7	DE	14	3/24/20	5.0	8.0
8	FL	16	4/3/20	15.0	0.0
9	GA	18	4/3/20	15.0	0.0
10	HI	20	3/25/20	6.0	0.0

In [90]:

```
# Drop row for Deleware
stayAtHome_Dates.drop(stayAtHome_Dates[stayAtHome_Dates['States'] == "DE"].
stayAtHome_Dates.reset_index(inplace=True)
stayAtHome_Dates.drop(['index'], axis=1,inplace=True)
stayAtHome_Dates
```

Out[90]:

	States	State_Number	StayAtHome_Date	DaysAfterFirst	Time
0	AL	0	4/4/20	16.0	17.0
1	AK	2	3/28/20	9.0	17.0
2	AZ	4	3/31/20	12.0	17.0
3	AR	6	NaN	NaN	NaN
4	CA	8	3/19/20	0.0	0.0
5	CO	10	3/26/20	7.0	6.0
6	CT	12	3/23/20	4.0	20.0
7	FL	16	4/3/20	15.0	0.0
8	GA	18	4/3/20	15.0	0.0
9	HI	20	3/25/20	6.0	0.0
10	ID	22	3/25/20	6.0	13.5

In [91]:

```
def effectOnAirlines(dailyAvgNOP_state):

    latestOrder = 0
    for index, row in stayAtHome_Dates.iterrows():
        if(row["DaysAfterFirst"] > latestOrder):
            latestOrder = row["DaysAfterFirst"]


    lossOfPassengersMarch = np.zeros(13)
    lossOfPassengersApril = np.zeros(int(latestOrder-12))

    totalPassengersLostMarch = 0
    totalPassengersLostApril = 0

    for i in range(int(latestOrder+1)):
            for index, row in stayAtHome_Dates.iterrows():
                if(i <= 12 and row["DaysAfterFirst"] == i):
                    totalPassengersLostMarch += dailyAvgNOP_state[row["Stat
                    totalPassengersLostApril += dailyAvgNOP_state[row["Stat
                    lossOfPassengersMarch[i] = totalPassengersLostMarch

                if(i > 12 and row["DaysAfterFirst"] == i):
                    totalPassengersLostApril += dailyAvgNOP_state[row["Stat
                    lossOfPassengersApril[i-13] = totalPassengersLostApril

                if (i <= 12):
                    lossOfPassengersMarch[i] = totalPassengersLostMarch

                if (i > 12):
                    lossOfPassengersApril[i-13] = totalPassengersLostApril


    return(lossOfPassengersMarch, lossOfPassengersApril)
```

Part 1:

Calculating average daily passengers by state, total daily passengers across the US and total number of passengers by state for the months of March and April

In [92]:

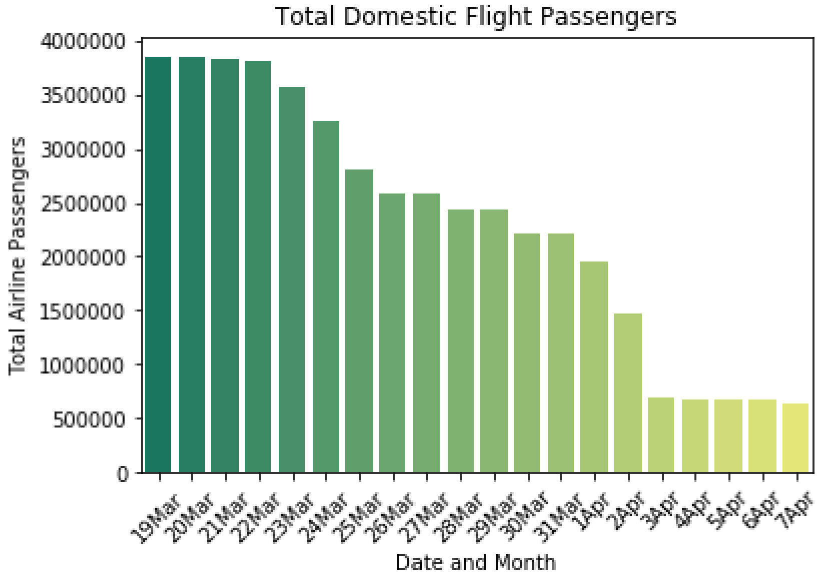
```
numberOfPassengers_state = np.zeros(100)
dailyAvgNOP_state = np.zeros(100)
totalDailyPassengers = np.zeros(2)
origin_state = flight_data.ORIGIN_STATE_ABR.unique().tolist()
origin_state.sort()

numberOfPassengers_state, dailyAvgNOP_state, totalDailyPassengers = flights
```

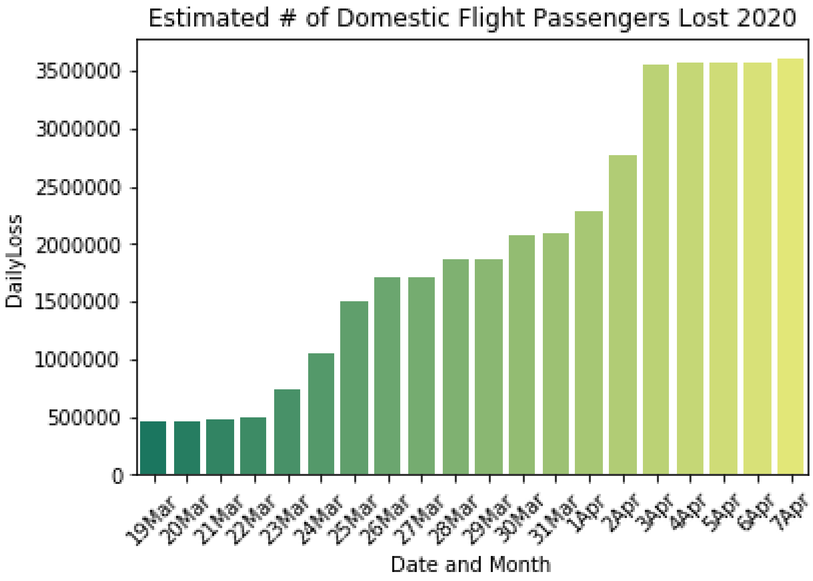
Part 2:

Part 2:
Estimation of loss of domestic flights as a function of days since the first state, California, released stay-at-home order

In [141]:

```
lossOfPassengersMarch, lossOfPassengersApril = effectOnAirlines(dailyAvgNOP
lossOfPassengers = list(lossOfPassengersMarch) + list(lossOfPassengersApril

totalDailyPassengersMarchCOVID = np.zeros(13)
totalDailyPassengersAprilCOVID = np.zeros(7)

for i in range(13):
    totalDailyPassengersMarchCOVID[i] = totalDailyPassengers[0] - lossOfPas

for i in range(7):
    totalDailyPassengersAprilCOVID[i] = totalDailyPassengers[1] - lossOfPas


totalDailyPassengersCOVID = list(totalDailyPassengersMarchCOVID) + list(tot

x = list(range(19,32))
x += list(range(1,8))

data = {'Total Airline Passengers':totalDailyPassengersCOVID,'DailyLoss':lo

df = pd.DataFrame(data)

df['Month'] = ['Mar' if x >= 19 else 'Apr' for x in df['Date']]
df['Date'] = df['Date'].apply(str)
df['Date and Month'] = df[['Date', 'Month']].apply(lambda x: ''.join(x), ax
plt.title('Total Domestic Flight Passengers')
chart = sns.barplot(x='Date and Month',y='Total Airline Passengers',data=df
chart.set_xticklabels(chart.get_xticklabels(), rotation=45)
plt.savefig('TotalDomseticFlightPassengers.png')
plt.show()


plt.title("Estimated # of Domestic Flight Passengers Lost 2020")
chart2 = sns.barplot(x='Date and Month',y='DailyLoss',data=df,palette='summ
chart2.set_xticklabels(chart2.get_xticklabels(), rotation=45)
plt.savefig('Estimated#ofDomesticFlightPassengersLost.png')
plt.show()
```





Part 3:

Dataframe of daily number of passengers travelling between each of the 50 states (March and April)

In [94]:
```
detailed_flightsMarch, detailed_flightsApril = detailedFlights(origin_state
```

In [95]:
```
detailed_flightsMarch
```

Out[95]:

	AK	AL	AR	AZ	CA	CO	CT	DE	FL	GA	HI	IA	ID	IL	
AK	6367	0	0	98	145	134	0	0	0	0	273	0	0	31	
AL	0	0	1	6	8	315	0	0	611	2708	0	0	0	494	
AR	0	0	0	103	111	252	0	0	168	1117	0	0	0	530	
AZ	102	5	95	2081	16698	4717	0	0	2293	2147	1231	832	497	5074	
CA	137	2	117	17064	64979	12754	68	0	7291	7220	12981	4	1353	12429	
CO	132	317	282	4850	13237	3534	252	0	6044	2955	774	631	732	4550	
CT	0	0	0	1	76	239	0	0	2965	841	0	0	0	760	
DE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
FL	0	616	169	2138	7066	5900	3061	0	10766	27330	0	630	0	16363	3
GA	0	2687	1134	2105	7244	2828	874	0	27441	4426	280	548	0	4053	2
HI	294	0	0	1229	12613	746	0	0	0	304	17941	0	0	728	

```
In [96]: detailed_flightsApril
```

```
Out[96]:
        AK    AL    AR     AZ     CA     CO    CT  DE     FL     GA     HI   IA    ID     IL
  AK  6498     0     0     69    125     96     0   0      0      0    142    0     0    127
  AL     0     0     0      0      3    324     0   0    524   2708      0    0     0    499
  AR     0     0     0     83    131    276     0   0    143   1151      0    0     0    623
  AZ    89     1    72   2199  16526   4470     0   0   2057   2037    780  615   469   4570
  CA   131     2   148  15642  68870  13034    22   0   6760   7374  12854    4  1188  12966
  CO   109   323   272   4310  13169   2974   349   0   4786   2367    864  619   743   3994
  CT     0     3     0     12     10    326     0   0   2999    915      0    0     0   1041
  DE     0     0     0      0      0      0     0   0      0      0      0    0     0      0
  FL     0   521   141   2003   6872   5042  3154   0   9840  26953      0  356     0  14263  3
  GA     0  2668  1134   1945   7412   2364   918   0  25054   4893    243  585     0   4449  2
  HI   186     0     0    843  13898    917     0   0      0    262  17156    0     0    534
```

```
In [97]: columns = ['Airport Name', 'City', 'State', 'Latitude', 'Longitude']
         airports = pd.read_csv('airports.csv', names=columns)
         airports
```

```
Out[97]:
                                        Airport Name  City  State   Latitude    Longitude
   0          Ted Stevens Anchorage International Airport   ANC     AK  61.174400  -149.996002
   1     Birmingham-Shuttlesworth International Airport   BHM     AL  33.562901   -86.753502
   2      Bill & Hillary Clinton National Airport/Adams ...   LIT     AR  34.729401   -92.224297
   3                Phoenix Sky Harbor International Airport   PHX     AZ  33.434299  -112.012001
   4                    Los Angeles International Airport   LAX     CA  33.942501  -118.407997
   5                         Denver International Airport   DEN     CO  39.861698  -104.672996
   6                        Bradley International Airport   BDL     CT  41.938900   -72.683197
   7                        Orlando International Airport   MCO     FL  28.429399   -81.308998
   8      Hartsfield Jackson Atlanta International Airport   ATL     GA  33.636700   -84.428101
   9                 Daniel K Inouye International Airport   HNL     HI  21.320620  -157.924228
  10                      Des Moines International Airport   DSM     IA  41.534000   -93.663101
```

```
In [98]: columns = ['dep_lat', 'dep_lon', 'arr_lat', 'arr_lon', 'nb_passengers']
         index = range(1,2450)
         origin_state = flight_data.ORIGIN_STATE_ABR.unique().tolist()
         origin_state.sort()
         origin_state.pop(7)

         map_visualization = pd.DataFrame(0.0, index = index, columns = columns)
```

In [99]:

```
for i in range (49):
    for j in range(49):
        map_visualization.iat[(i*50)+j,0] = float(airports['Latitude'][i])
        map_visualization.iat[(i*50)+j,1] = airports['Longitude'][i]
        map_visualization.iat[(i*50)+j,2] = airports['Latitude'][j]
        map_visualization.iat[(i*50)+j,3] = airports['Longitude'][j]
        map_visualization.iat[(i*50)+j,4] = detailed_flightsMarch.iat[i,j]

map_visualization
```

Out[99]:

	dep_lat	dep_lon	arr_lat	arr_lon	nb_passengers
1	61.174400	-149.996002	61.174400	-149.996002	6367.0
2	61.174400	-149.996002	33.562901	-86.753502	0.0
3	61.174400	-149.996002	34.729401	-92.224297	0.0
4	61.174400	-149.996002	33.434299	-112.012001	98.0
5	61.174400	-149.996002	33.942501	-118.407997	145.0
6	61.174400	-149.996002	39.861698	-104.672996	134.0
7	61.174400	-149.996002	41.938900	-72.683197	0.0
8	61.174400	-149.996002	28.429399	-81.308998	0.0
9	61.174400	-149.996002	33.636700	-84.428101	0.0
10	61.174400	-149.996002	21.320620	-157.924228	0.0
11	61.174400	-149.996002	41.534000	-93.663101	273.0

In [139]:

```
# create gradient to color the routes according to the number of flights
grad = Gradient(((0, 0, 0, 0), (0.5, 204, 0, 153), (1, 255, 204, 230)))
# initialize GCMapper and set data
gcm = GCMapper(cols=grad, height=2000, width=4000)
gcm.set_data(map_visualization['dep_lon'], map_visualization['dep_lat'], ma
             map_visualization['arr_lat'], map_visualization['nb_passengers

img = gcm.draw()
img.save('flights_map_gcmap.png')

order_State = []

for i in range(49):
    for index, row in stayAtHome_Dates.iterrows():
        if(row["DaysAfterFirst"] == i):
            order_State.append(row["States"])

order_Latitude = []
for i in range(len(order_State)):
    for index, row in airports.iterrows():
        if(row["State"] == order_State[i]):
            order_Latitude.append(row["Latitude"])

df_filtered = map_visualization

for i in range(len(order_Latitude)):
    df_filteredTemp = df_filtered[df_filtered['dep_lat'] != order_Latitude[
    df_filteredTemp2 = df_filteredTemp[df_filteredTemp['arr_lat'] != order_
    df_filtered = df_filteredTemp2
    # create gradient to color the routes according to the number of flight
    grad = Gradient(((0, 0, 0, 0), (0.5, 204, 0, 153), (1, 255, 204, 230)))
    # initialize GCMapper and set data
    gcm = GCMapper(cols=grad, height=2000, width=4000)
    gcm.set_data(df_filtered['dep_lon'], df_filtered['dep_lat'], df_filtere
             df_filtered['arr_lat'], df_filtered['nb_passengers'])

    img = gcm.draw()
    img.save('flights_map_gcmap' + str(i) + ".png")
```

Part 4:

Spread of the Virus if none of the states had gone on lockdown (March19 - March31)

Number of Infections in Each State March 19th

In [105]:
```
detailed_flightsMarch.drop("DE",axis=0,inplace = True)
detailed_flightsMarch.drop(['DE'],axis=1,inplace = True)
detailed_flightsMarch
```

Out[105]:

	AK	AL	AR	AZ	CA	CO	CT	FL	GA	HI	IA	ID	IL	IN
AK	6367	0	0	98	145	134	0	0	0	273	0	0	31	0
AL	0	0	1	6	8	315	0	611	2708	0	0	0	494	0
AR	0	0	0	103	111	252	0	168	1117	0	0	0	530	0
AZ	102	5	95	2081	16698	4717	0	2293	2147	1231	832	497	5074	632
CA	137	2	117	17064	64979	12754	68	7291	7220	12981	4	1353	12429	700
CO	132	317	282	4850	13237	3534	252	6044	2955	774	631	732	4550	626
CT	0	0	0	1	76	239	0	2965	841	0	0	0	760	3
FL	0	616	169	2138	7066	5900	3061	10766	27330	0	630	0	16363	3667
GA	0	2687	1134	2105	7244	2828	874	27441	4426	280	548	0	4053	2057
HI	294	0	0	1229	12613	746	0	0	304	17941	0	0	728	0
IA	0	0	0	782	1	613	0	612	520	0	6	0	1231	4

In [112]:
```
changeInInfectedCitizens, state_InfectionsDF, original_StateInfectionsDF, i
original_StateInfectionsDF
```

Out[112]:

	# of Infected Citizens March 19th
AK	78
AL	12
AR	47
AZ	62
CA	1067
CO	278
CT	159
FL	434
GA	282
HI	26
IA	23

Number of Infections in Each State March 31st

In [113]:

```
state_InfectionsDF
```

Out[113]:

	# of Infected Citizens March 31st
AK	1790050
AL	277619
AR	1459944
AZ	1211909
CA	19213019
CO	3203592
CT	1955830
FL	9300274
GA	5195839
HI	494077
IA	593392

Change in Number of Infections in Each State

In [114]:

```
changeInInfectedCitizens
```

Out[114]:

	Change in # of Infected Citizens
AK	1789972
AL	277607
AR	1459897
AZ	1211847
CA	19211952
CO	3203314
CT	1955671
FL	9299840
GA	5195557
HI	494051
IA	593369

```
In [115]: print("Dataframes of Infected Travelers Traveling Between States")
          for i in range(13):
              print("March", i+19)
              print(infectedTravelers_Collection[i])
              print("")
```

```
Dataframes of Infected Travelers Traveling Between States
March 19
          AK        AL        AR        AZ        CA        CO        CT
\
AK  0.101286  0.000000  0.000000  0.002013  0.003916  0.006469  0.000000
AL  0.000000  0.000000  0.000006  0.000123  0.000216  0.015206  0.000000
AR  0.000000  0.000000  0.000000  0.002116  0.002997  0.012165  0.000000
AZ  0.001623  0.000082  0.000613  0.042754  0.450918  0.227711  0.000000
CA  0.002179  0.000033  0.000755  0.350575  1.754712  0.615693  0.003033
CO  0.002100  0.005200  0.001821  0.099642  0.357456  0.170602  0.011238
CT  0.000000  0.000000  0.000000  0.000021  0.002052  0.011538  0.000000
FL  0.000000  0.010105  0.001091  0.043925  0.190812  0.284820  0.136510
GA  0.000000  0.044076  0.007322  0.043247  0.195619  0.136520  0.038978
HI  0.004677  0.000000  0.000000  0.025249  0.340605  0.036013  0.000000
IA  0.000000  0.000000  0.000000  0.016066  0.000027  0.029592  0.000000
ID  0.000000  0.000000  0.000000  0.010190  0.037023  0.036930  0.000000
IL  0.000620  0.008251  0.003377  0.102436  0.333368  0.219842  0.033135
IN  0.000000  0.000000  0.000000  0.014196  0.019308  0.030896  0.000000
KS  0.000000  0.000000  0.000000  0.003431  0.000081  0.015400  0.000000
```

Dataframes of Infected Travelers Traveling Between States

```
In [116]: for i in range(13):
              print("March", i+19)
              print(infectedTravelers_Collection[i])
              print("")
```

```
March 19
          AK        AL        AR        AZ        CA        CO        CT
\
AK  0.101286  0.000000  0.000000  0.002013  0.003916  0.006469  0.000000
AL  0.000000  0.000000  0.000006  0.000123  0.000216  0.015206  0.000000
AR  0.000000  0.000000  0.000000  0.002116  0.002997  0.012165  0.000000
AZ  0.001623  0.000082  0.000613  0.042754  0.450918  0.227711  0.000000
CA  0.002179  0.000033  0.000755  0.350575  1.754712  0.615693  0.003033
CO  0.002100  0.005200  0.001821  0.099642  0.357456  0.170602  0.011238
CT  0.000000  0.000000  0.000000  0.000021  0.002052  0.011538  0.000000
FL  0.000000  0.010105  0.001091  0.043925  0.190812  0.284820  0.136510
GA  0.000000  0.044076  0.007322  0.043247  0.195619  0.136520  0.038978
HI  0.004677  0.000000  0.000000  0.025249  0.340605  0.036013  0.000000
IA  0.000000  0.000000  0.000000  0.016066  0.000027  0.029592  0.000000
ID  0.000000  0.000000  0.000000  0.010190  0.037023  0.036930  0.000000
IL  0.000620  0.008251  0.003377  0.102436  0.333368  0.219842  0.033135
IN  0.000000  0.000000  0.000000  0.014196  0.019308  0.030896  0.000000
KS  0.000000  0.000000  0.000000  0.003431  0.000081  0.015400  0.000000
KY  0.000000  0.000000  0.000000  0.011176  0.017094  0.040068  0.001561
```