In [17]:
```
import pandas as pd
import os
import numpy as np
import geopandas as gpd
import matplotlib.pyplot as plt
import folium
```

In [18]:
```
project_path = "/Volumes/T7/Water Project"
```

In [19]:
```
os.chdir("/Volumes/T7/Water Project")
```

In [20]:
```
ACS_folder = os.open("State, County Level ACS data", os.O_RDONLY)
```

In [21]:
```
folder_list = list(list(os.walk("State, County Level ACS data"))[0])
```

In [22]:
```
files = folder_list[2]
```

In [23]:
```
files1 = []
for i in files:
    if i[0] == "A":
        files1.append(i) ## Selects all files from ACS
```

In [44]:
```
os.chdir(project_path+"/State, County Level ACS data")
dict_dat = {}
years = [2010,2014,2016,2020,2022]
for i in files1:
        dict_dat[years[files1.index(i)]]  = pd.read_csv(i, skiprows =
```

In [45]:
```
for i in years:
    ##Calculates % for year i
    dict_dat[i]["Percent Of Homes Lacking"] = (dict_dat[i].iloc[:,6]/d
    ##Drop unnamed column
    dict_dat[i] = dict_dat[i].drop('Unnamed: 8', axis = 1)
    #Standard Column Names
    dict_dat[i].columns = ["Geography", "Geographic Area Name",
                           "Estimate Total", "Estimate Total MoE",
                           "Estimate Complete Plumbing", "Estimate Com
                          "Estimate Lacking Plumbing", "Estimate Lacki
    #Adds Year Suffix to the data
    dict_dat[i] = dict_dat[i].add_suffix("_{0}".format(i))
```

In [46]:
```
state_data = dict()
for i in years:
    state_data[i] = dict_dat[i].iloc[:52,:]
```

In [47]:
```
os.chdir(project_path)
```

In [48]:
```
gdf = gpd.read_file('cb_2018_us_state_500k')
```

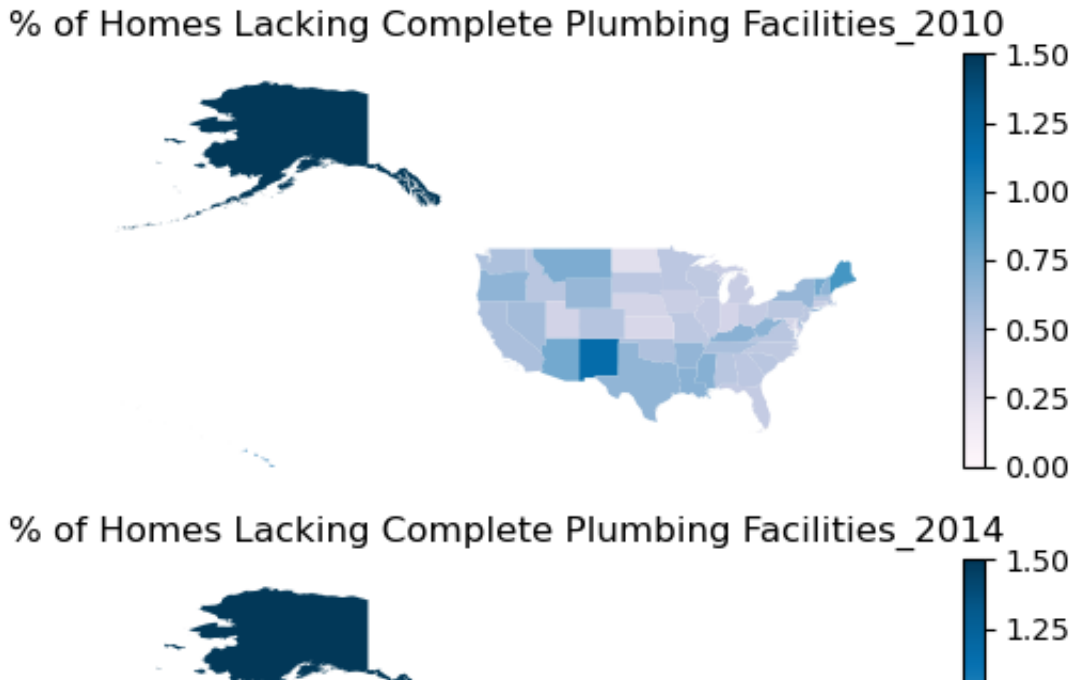
In [49]:
```
State_geo = gdf
```

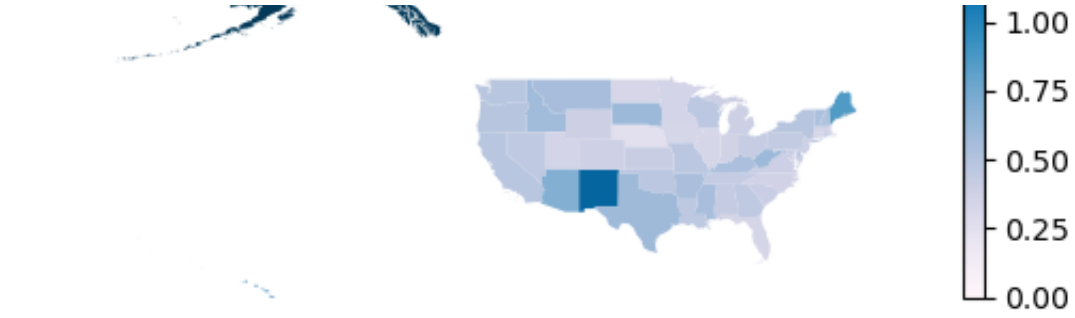
In [50]:
```
for i in years:
    State_geo = State_geo.merge(state_data[i], how = 'outer',
                                left_on = "NAME", right_on = "Geograph
```

In [51]:
```
State_geo.to_csv("State_ACS_Data.csv")
```

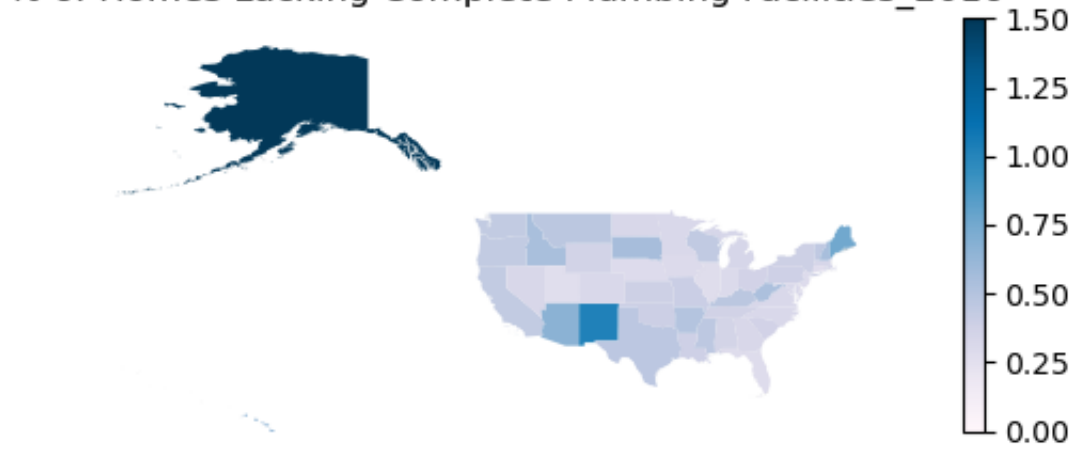
In [53]:
```
fig, axs = plt.subplots(5,1, figsize=(6,12))
for i in years:
    State_geo.plot(column="Percent of Homes Lacking_{0}".format(i),
                   cmap='PuBu', ax=axs[years.index(i)], legend=True, v
    axs[years.index(i)].set_title("% of Homes Lacking Complete Plumbin
    axs[years.index(i)].axis('off')
    axs[years.index(i)].set_xlim(-180,-60)  # Set the x-axis limits ba
    axs[years.index(i)].set_ylim(20,75)  # Set the y-axis limits based

plt.tight_layout()
plt.show()
```

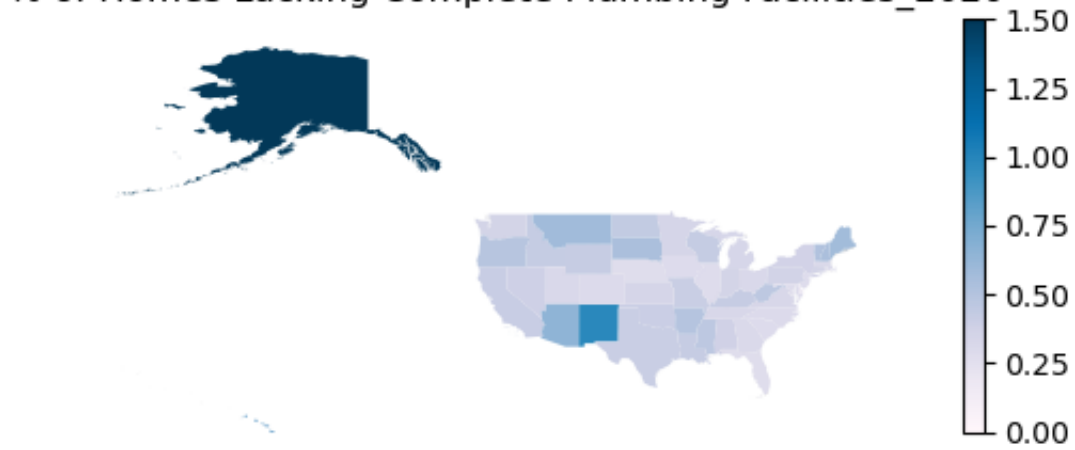




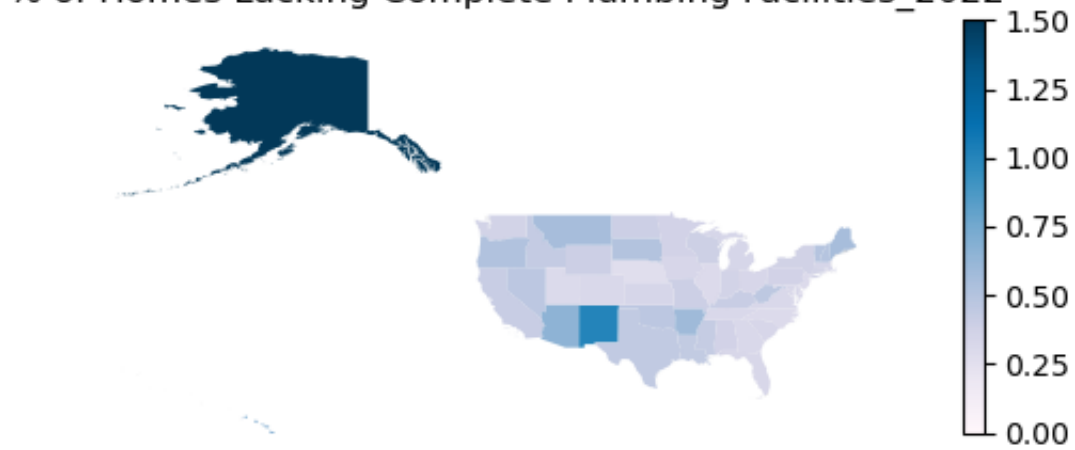
% of Homes Lacking Complete Plumbing Facilities_2016



% of Homes Lacking Complete Plumbing Facilities_2020

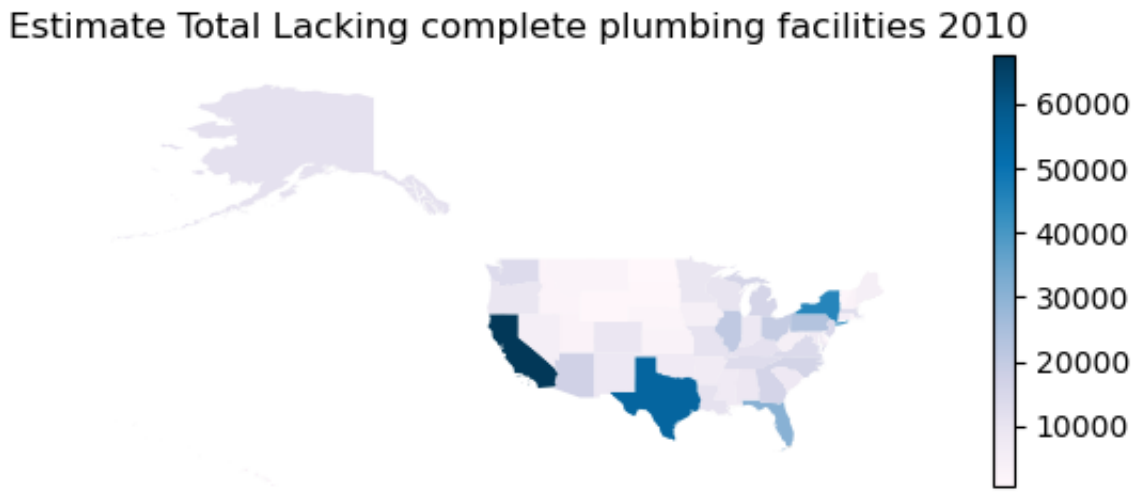


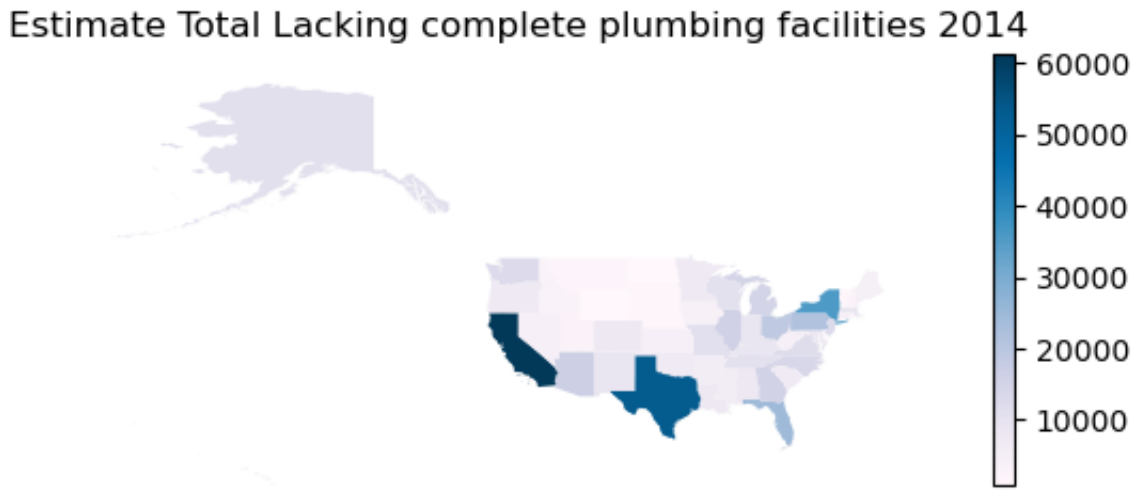
% of Homes Lacking Complete Plumbing Facilities_2022

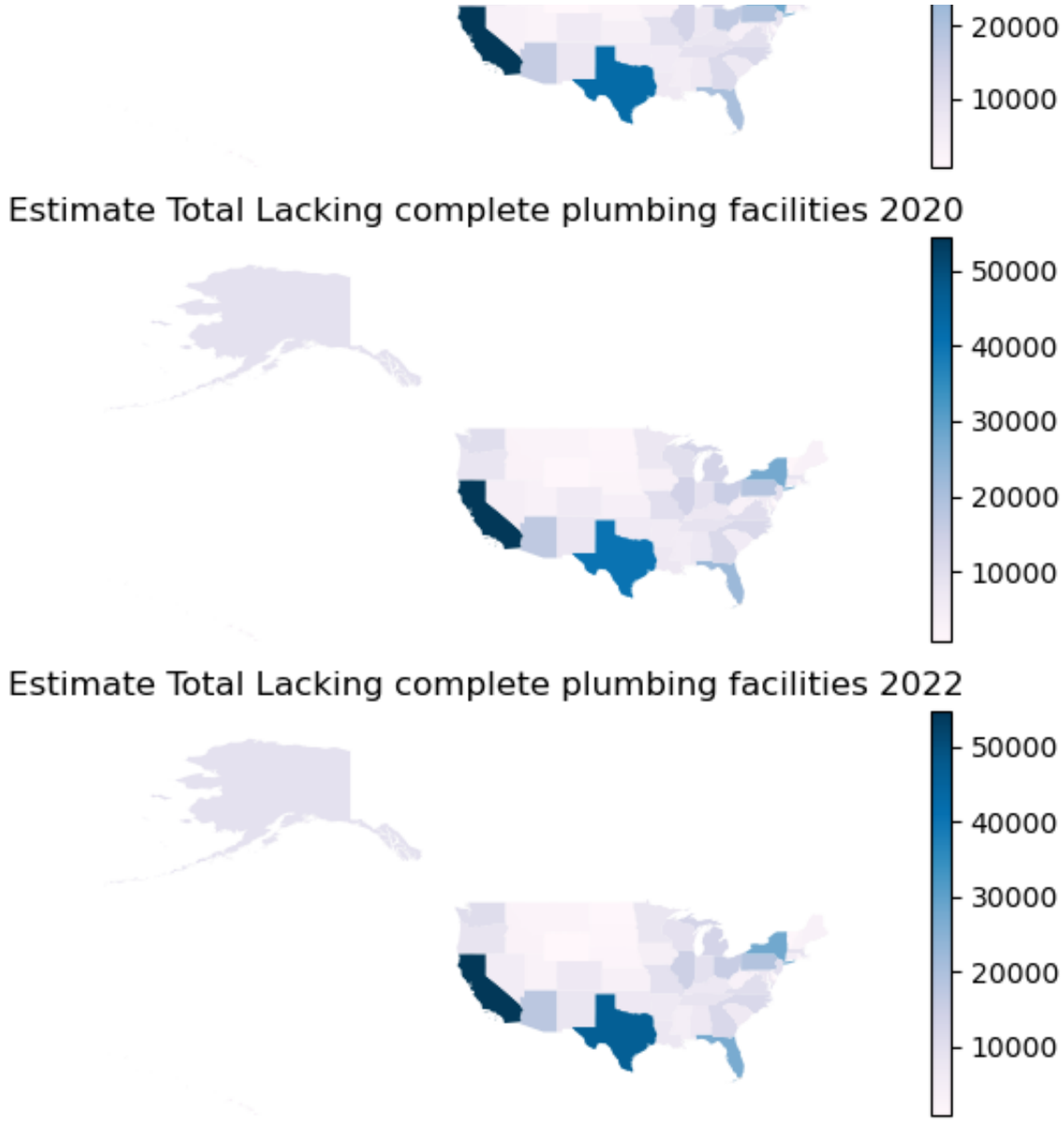


In [54]:
```
fig, axs = plt.subplots(5,1, figsize=(6,12))
for i in years:
    State_geo.plot(column="Estimate Lacking Plumbing_{0}".format(i),
                   cmap='PuBu', ax=axs[years.index(i)], legend=True)
    axs[years.index(i)].set_title("Estimate Total Lacking complete plu
    axs[years.index(i)].axis('off')
    axs[years.index(i)].set_xlim(-180,-60)  # Set the x-axis limits ba
    axs[years.index(i)].set_ylim(20,75)  # Set the y-axis limits based

plt.tight_layout()
plt.show()
```







In [115]:
```
m=folium.Map()
State_geo.explore(m = m,column='Percent Of Homes Lacking_2010',
              linewidth=0.8, edgecolor='0.8', legend = True, name = '2
                    tooltip=["NAME",'Percent Of Homes Lacking_2010'],
                        overlay = True , vmin = 0, vmax = 1.5)
for i in years[1:]:
    State_geo.explore(m = m, column='Percent Of Homes Lacking_{0}'.for
              linewidth=0.8, edgecolor='0.8', legend = False, name = s
                    tooltip=["NAME",'Percent Of Homes Lacking_{0}'.for
                        overlay = True , vmin = 0, vmax = 1.5)
folium.LayerControl().add_to(m)
```

Out[115]: <folium.map.LayerControl at 0x7fd5d4f4e640>

In [117]:
```
m.save("ACS_States.html")
```

In [56]:
```
gdcounties = gpd.read_file(os.getcwd()+'/cb_2018_us_county_5m',
                           dtype = {'COUNTYFP' : str, 'STATEFP'  : str
```

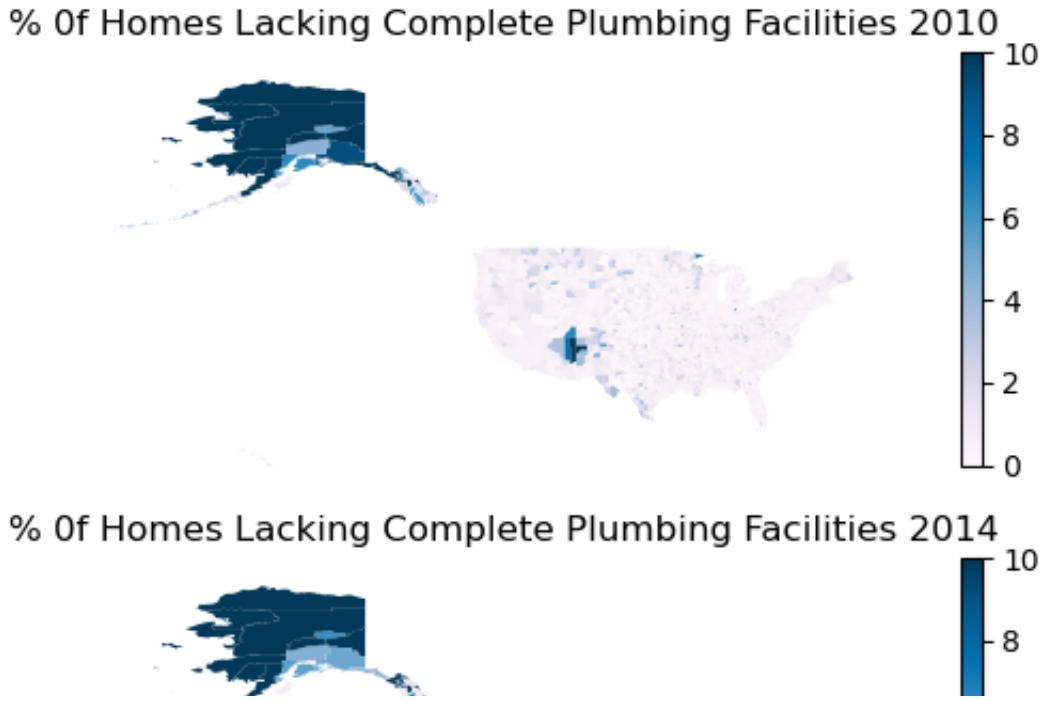
In [57]:
```
county_geo = gdcounties
```

In [58]:
```
county_data = dict()
for i in years:
    county_data[i] = dict_dat[i][dict_dat[i]['Geography_{}'.format(i)]
```

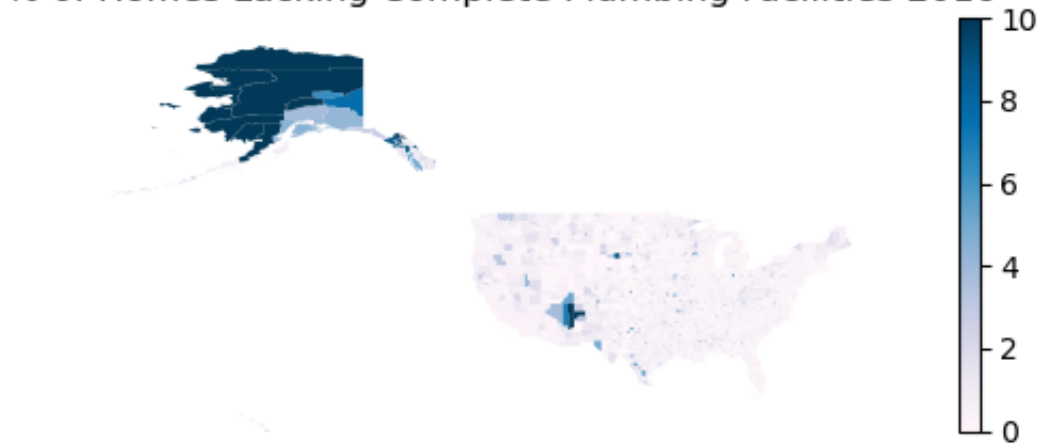
In [59]:
```
for i in years:
    county_geo = county_geo.merge(county_data[i], how = 'outer',
                                  left_on = "AFFGEOID", right_on = "Geog
```

In [61]:
```
fig, axs = plt.subplots(5,1, figsize=(6,12))
for i in years:
    county_geo.plot(column="Percent of Homes Lacking_{0}".format(i),
                    cmap='PuBu', ax=axs[years.index(i)], legend=True,vm
    axs[years.index(i)].set_title("% 0f Homes Lacking Complete Plumbir
    axs[years.index(i)].axis('off')
    axs[years.index(i)].set_xlim(-180,-60)  # Set the x-axis limits ba
    axs[years.index(i)].set_ylim(20,75)  # Set the y-axis limits based

plt.tight_layout()
plt.show()
```



% 0f Homes Lacking Complete Plumbing Facilities 2016

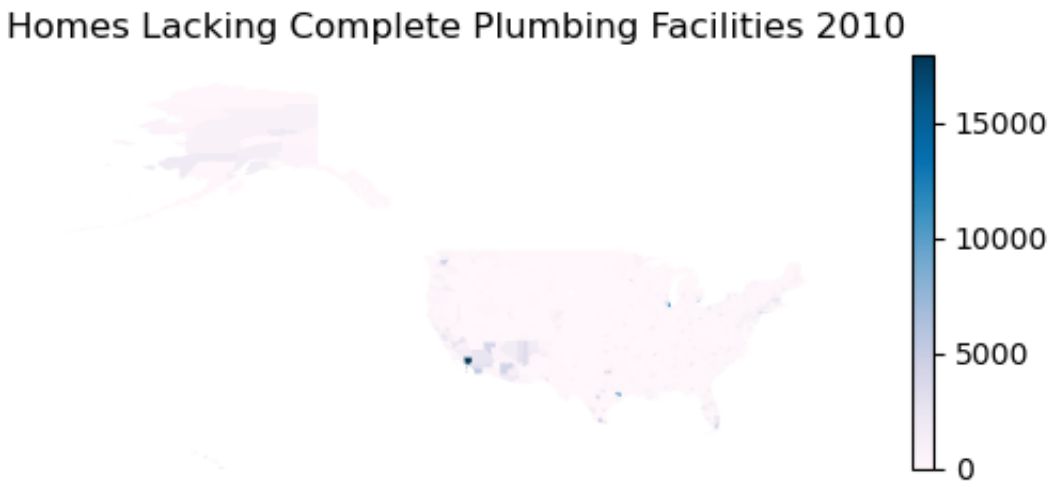


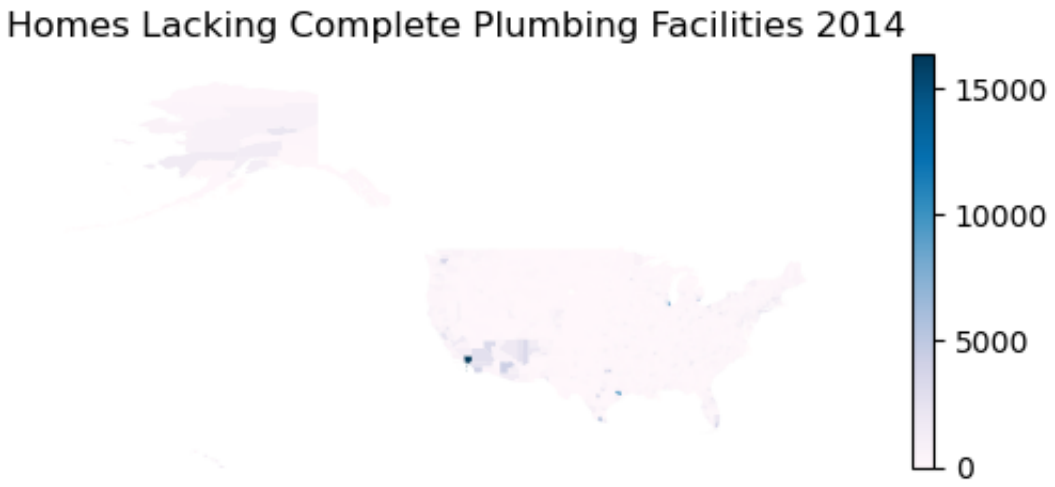
% 0f Homes Lacking Complete Plumbing Facilities 2020

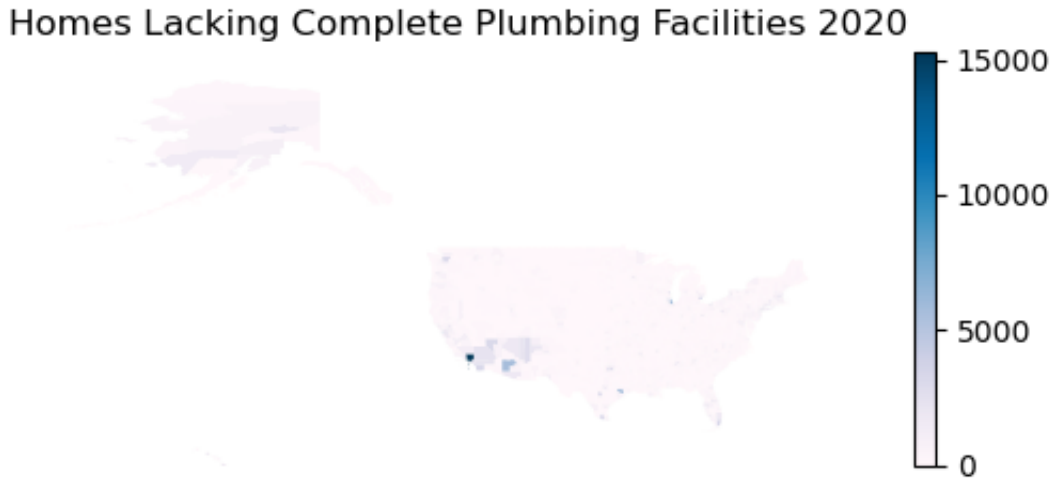
% 0f Homes Lacking Complete Plumbing Facilities 2022

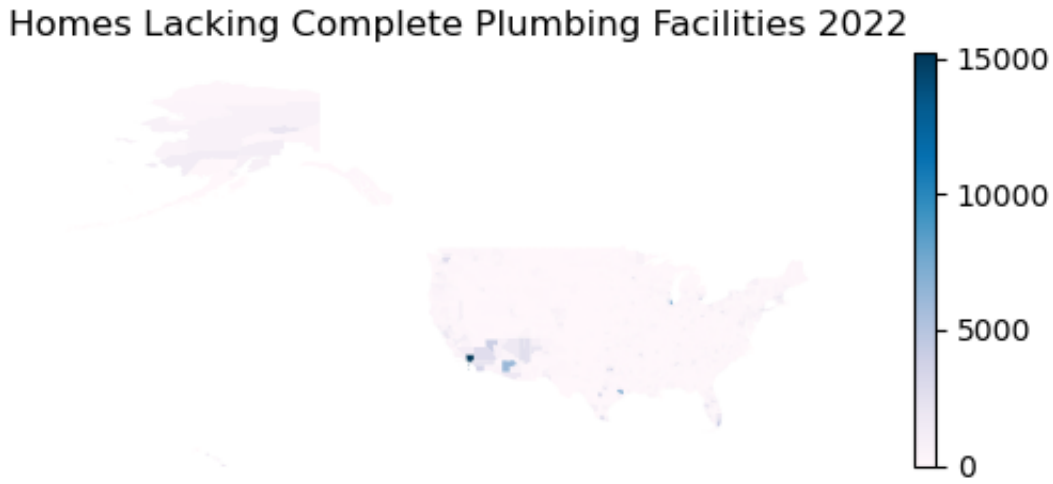
In [64]:
```
fig, axs = plt.subplots(5,1, figsize=(6,12))
for i in years:
    county_geo.plot(column="Estimate Lacking Plumbing_{0}".format(i),
                    cmap='PuBu', ax=axs[years.index(i)], legend=True)
    axs[years.index(i)].set_title("Homes Lacking Complete Plumbing Fac
    axs[years.index(i)].axis('off')
    axs[years.index(i)].set_xlim(-180,-60)  # Set the x-axis limits ba
    axs[years.index(i)].set_ylim(20,75)  # Set the y-axis limits based

plt.tight_layout()
plt.show()
```









In [67]:
```
county_map = county_geo.explore(column="Percent of Homes Lacking_2010"
                cmap = 'RdPu', legend=True,vmin = 0, vmax = 10,
                   tooltip=["NAME",'Percent of Homes Lacking_2010'])
```

In [179]:
```
county_map.save("County Map Example 2010.html")
```

In [69]:
```
county_map_2 = county_geo.explore(column="Estimate Lacking Plumbing_20
                cmap = 'RdPu', legend=True,
                   tooltip=["NAME",'Estimate Lacking Plumbing_2010'])
```

In [70]:
```
county_map_2
```

Out[70]:
Make this Notebook Trusted to load map: File -> Trust Notebook

In [216]:
```
AI = dict()
for i in years:
    AI[i] = dict_dat[i][dict_dat[i]['Geography_{}'.format(i)].str.cont
```

In [220]:
```
AI_geos = gpd.read_file('/Volumes/T7/Water Project/cb_2018_us_aiannh_5
```

In [218]:
```
for i in years:
    AI[i]['ID_{0}'.format(i)] = AI[i]['Geography_{0}'.format(i)].str[1
```

```
/var/folders/fj/58nmvrz11g517ghvh__5bmb40000gn/T/ipykernel_19644/1817
986057.py: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/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  AI[i]['ID_{0}'.format(i)] = AI[i]['Geography_{0}'.format(i)].str[1
1:-1]
/var/folders/fj/58nmvrz11g517ghvh__5bmb40000gn/T/ipykernel_19644/1817
986057.py: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/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  AI[i]['ID_{0}'.format(i)] = AI[i]['Geography_{0}'.format(i)].str[1
1:-1]
/var/folders/fj/58nmvrz11g517ghvh__5bmb40000gn/T/ipykernel_19644/1817
986057.py: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/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  AI[i]['ID_{0}'.format(i)] = AI[i]['Geography_{0}'.format(i)].str[1
1:-1]
/var/folders/fj/58nmvrz11g517ghvh__5bmb40000gn/T/ipykernel_19644/1817
986057.py: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/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  AI[i]['ID_{0}'.format(i)] = AI[i]['Geography_{0}'.format(i)].str[1
1:-1]
/var/folders/fj/58nmvrz11g517ghvh__5bmb40000gn/T/ipykernel_19644/1817
986057.py: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/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  AI[i]['ID_{0}'.format(i)] = AI[i]['Geography_{0}'.format(i)].str[1
1:-1]
```

In [221]:

```
for i in years:
    AI_geos = AI_geos.merge(AI[i], how = 'outer',
                            left_on = "GEOID", right_on = "ID_{0}"
```

In [233]:

```
AI_map = AI_geos.explore(column = 'Percent Of Homes Lacking_2014', leg
                  tooltip=["NAME",'Percent Of Homes Lacking_2014'] )
```

In [234]:
```
AI_map.save("AI Trial Map.html")
```

In [194]:
```
unique_first_4_chars
```

Out[194]:
```
array(['0400000US', '0500000US', '2830000US', '8600000US'], dtype=obj
ect)
```

In [235]:
```
ZCTAs = dict()
for i in years:
    ZCTAs[i] = dict_dat[i][dict_dat[i]['Geography_{}'.format(i)].str.c
```

In [247]:
```
for i in years:
    ZCTAs[i]['ID_{0}'.format(i)] = ZCTAs[i]['Geography_{0}'.format(i)]
```

```
/var/folders/fj/58nmvrz11g517ghvh__5bmb40000gn/T/ipykernel_19644/7469
56582.py: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/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  ZCTAs[i]['ID_{0}'.format(i)] = ZCTAs[i]['Geography_{0}'.format(i)].
str[-5:]
/var/folders/fj/58nmvrz11g517ghvh__5bmb40000gn/T/ipykernel_19644/7469
56582.py: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/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  ZCTAs[i]['ID_{0}'.format(i)] = ZCTAs[i]['Geography_{0}'.format(i)].
```

In [239]:
```
ZCTA2020 = gpd.read_file("/Volumes/T7/Water Project/tl_2020_us_zcta520
```

In [250]:
```
ZCTA_geos = ZCTA2020
```

In [252]:
```
for i in years:
    ZCTA_geos = ZCTA_geos.merge(ZCTAs[i], how = 'outer',
                                left_on = "GEOID20", right_on = "ID_{0
```

In [255]:
```
ZCTA_map = ZCTA_geos.explore(column = 'Percent Of Homes Lacking_2014',
                  tooltip=["NAME",'Percent Of Homes Lacking_2014'] )
```

```
---------------------------------------------------------------------
```