analysis

April 21, 2024

[1]:
```
!pip install geopandas
!pip install pygris
!pip install seaborn
```

```
Requirement already satisfied: geopandas in /opt/conda/lib/python3.11/site-
packages (0.14.3)
Requirement already satisfied: fiona>=1.8.21 in /opt/conda/lib/python3.11/site-
packages (from geopandas) (1.9.6)
Requirement already satisfied: packaging in /opt/conda/lib/python3.11/site-
packages (from geopandas) (23.2)
Requirement already satisfied: pandas>=1.4.0 in /opt/conda/lib/python3.11/site-
packages (from geopandas) (2.2.0)
Requirement already satisfied: pyproj>=3.3.0 in /opt/conda/lib/python3.11/site-
packages (from geopandas) (3.6.1)
Requirement already satisfied: shapely>=1.8.0 in /opt/conda/lib/python3.11/site-
packages (from geopandas) (2.0.4)
Requirement already satisfied: attrs>=19.2.0 in /opt/conda/lib/python3.11/site-
packages (from fiona>=1.8.21->geopandas) (23.1.0)
Requirement already satisfied: certifi in /opt/conda/lib/python3.11/site-
packages (from fiona>=1.8.21->geopandas) (2023.11.17)
Requirement already satisfied: click~=8.0 in /opt/conda/lib/python3.11/site-
packages (from fiona>=1.8.21->geopandas) (8.1.7)
Requirement already satisfied: click-plugins>=1.0 in
/opt/conda/lib/python3.11/site-packages (from fiona>=1.8.21->geopandas) (1.1.1)
Requirement already satisfied: cligj>=0.5 in /opt/conda/lib/python3.11/site-
packages (from fiona>=1.8.21->geopandas) (0.7.2)
Requirement already satisfied: six in /opt/conda/lib/python3.11/site-packages
(from fiona>=1.8.21->geopandas) (1.16.0)
Requirement already satisfied: numpy<2,>=1.23.2 in
/opt/conda/lib/python3.11/site-packages (from pandas>=1.4.0->geopandas) (1.26.3)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.11/site-packages (from pandas>=1.4.0->geopandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.11/site-
packages (from pandas>=1.4.0->geopandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.11/site-
packages (from pandas>=1.4.0->geopandas) (2023.4)
Requirement already satisfied: pygris in /opt/conda/lib/python3.11/site-packages
(0.1.6)
```

```
Requirement already satisfied: geopandas>=0.9 in /opt/conda/lib/python3.11/site-
packages (from pygris) (0.14.3)
Requirement already satisfied: fiona in /opt/conda/lib/python3.11/site-packages
(from pygris) (1.9.6)
Requirement already satisfied: pandas in /opt/conda/lib/python3.11/site-packages
(from pygris) (2.2.0)
Requirement already satisfied: shapely in /opt/conda/lib/python3.11/site-
packages (from pygris) (2.0.4)
Requirement already satisfied: requests in /opt/conda/lib/python3.11/site-
packages (from pygris) (2.31.0)
Requirement already satisfied: appdirs in /opt/conda/lib/python3.11/site-
packages (from pygris) (1.4.4)
Requirement already satisfied: pip in /opt/conda/lib/python3.11/site-packages
(from pygris) (23.3)
Requirement already satisfied: numpy in /opt/conda/lib/python3.11/site-packages
(from pygris) (1.26.3)
Requirement already satisfied: rtree in /opt/conda/lib/python3.11/site-packages
(from pygris) (1.2.0)
Requirement already satisfied: packaging in /opt/conda/lib/python3.11/site-
packages (from geopandas>=0.9->pygris) (23.2)
Requirement already satisfied: pyproj>=3.3.0 in /opt/conda/lib/python3.11/site-
packages (from geopandas>=0.9->pygris) (3.6.1)
Requirement already satisfied: attrs>=19.2.0 in /opt/conda/lib/python3.11/site-
packages (from fiona->pygris) (23.1.0)
Requirement already satisfied: certifi in /opt/conda/lib/python3.11/site-
packages (from fiona->pygris) (2023.11.17)
Requirement already satisfied: click~=8.0 in /opt/conda/lib/python3.11/site-
packages (from fiona->pygris) (8.1.7)
Requirement already satisfied: click-plugins>=1.0 in
/opt/conda/lib/python3.11/site-packages (from fiona->pygris) (1.1.1)
Requirement already satisfied: cligj>=0.5 in /opt/conda/lib/python3.11/site-
packages (from fiona->pygris) (0.7.2)
Requirement already satisfied: six in /opt/conda/lib/python3.11/site-packages
(from fiona->pygris) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.11/site-packages (from pandas->pygris) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.11/site-
packages (from pandas->pygris) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.11/site-
packages (from pandas->pygris) (2023.4)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/conda/lib/python3.11/site-packages (from requests->pygris) (3.3.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.11/site-
packages (from requests->pygris) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/conda/lib/python3.11/site-packages (from requests->pygris) (2.0.7)
Requirement already satisfied: seaborn in /opt/conda/lib/python3.11/site-
packages (0.13.2)
```

```
Requirement already satisfied: numpy!=1.24.0,>=1.20 in
/opt/conda/lib/python3.11/site-packages (from seaborn) (1.26.3)
Requirement already satisfied: pandas>=1.2 in /opt/conda/lib/python3.11/site-
packages (from seaborn) (2.2.0)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in
/opt/conda/lib/python3.11/site-packages (from seaborn) (3.8.2)
Requirement already satisfied: contourpy>=1.0.1 in
/opt/conda/lib/python3.11/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(1.2.0)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.11/site-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/conda/lib/python3.11/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(4.47.2)
Requirement already satisfied: kiwisolver>=1.3.1 in
/opt/conda/lib/python3.11/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(1.4.5)
Requirement already satisfied: packaging>=20.0 in
/opt/conda/lib/python3.11/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(23.2)
Requirement already satisfied: pillow>=8 in /opt/conda/lib/python3.11/site-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/opt/conda/lib/python3.11/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/lib/python3.11/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.11/site-
packages (from pandas>=1.2->seaborn) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.11/site-
packages (from pandas>=1.2->seaborn) (2023.4)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.11/site-
packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
```

[2]:
```
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
import seaborn as sb
```

```
/tmp/ipykernel_4491/3691917747.py:1: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of
pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better
interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
```

```
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

1 calculate ian score

look through florida first

[3]:
```
fl_risk = pd.read_csv("risk_index/NRI_Table_CensusTracts_FL_short.csv")
fl_fin = pd.read_csv("census/financial/FL_fin.csv")
fl_housing = pd.read_csv("census/housing/FL_housing.csv")
fl_income = pd.read_csv("census/income/FL_income.csv")
fl_pov = pd.read_csv("census/poverty/FL_pov.csv")
```

[4]:
```
fl_risk.head()
```

[4]:
```
   Unnamed: 0   OID_        NRI_ID    STATE STATEABBRV  STATEFIPS   COUNTY  \
0           0  16098  T12001000201  Florida         FL         12  Alachua
1           1  16099  T12001000202  Florida         FL         12  Alachua
2           2  16100  T12001000301  Florida         FL         12  Alachua
3           3  16101  T12001000302  Florida         FL         12  Alachua
4           4  16102  T12001000400  Florida         FL         12  Alachua

  COUNTYTYPE  COUNTYFIPS  STCOFIPS  …  HRCN_EALS            HRCN_EALR  \
0     County           1     12001  …  85.012892      Relatively High
1     County           1     12001  …  81.036877  Relatively Moderate
2     County           1     12001  …  85.807790      Relatively High
3     County           1     12001  …  84.843538  Relatively Moderate
4     County           1     12001  …  87.688998      Relatively High

  HRCN_ALRB     HRCN_ALRP  HRCN_ALRA  HRCN_ALR_NPCTL    HRCN_RISKV  \
0  0.001033  4.528248e-07   0.000000       84.230200  5.856124e+05
1  0.001033  4.528666e-07   0.000000       84.227149  4.329441e+05
2  0.001033  4.528666e-07   0.000000       84.286652  8.268655e+05
3  0.001033  4.528666e-07   0.009639       84.265291  7.187622e+05
4  0.001033  4.528666e-07   0.009639       84.246983  1.267773e+06

   HRCN_RISKS           HRCN_RISKR NRI_VER
0   84.638979  Relatively Moderate  Mar-23
1   82.072096  Relatively Moderate  Mar-23
2   87.171781      Relatively High  Mar-23
3   86.081205      Relatively High  Mar-23
4   90.914442      Relatively High  Mar-23

[5 rows x 91 columns]
```

[ ]:
```

```

[5]:
```
fl_risk["NRI_ID"] = fl_risk["NRI_ID"].str[1:]
```

[6]:
```
# making the id in geography the same in all
fl_pov["geography"] = fl_pov["geography"].str[9:]
fl_fin["geography"] = fl_fin["geography"].str[9:]
fl_housing["geography"] = fl_housing["geography"].str[9:]
fl_income["geography"] = fl_income["geography"].str[9:]
```

[7]:
```
fl_risk = fl_risk.rename(columns={"NRI_ID": "geography"})
```

[8]:
```
mixed = pd.merge(fl_risk, fl_pov, on="geography",how="inner")
```

[9]:
```
fl_risk[["geography"]].loc[5113]
```

[9]:
```
geography    12133970303
Name: 5113, dtype: object
```

[10]:
```
fl_pov[["geography", "percent_below_poverty_level"]].isna().sum()
```

[10]:
```
geography                     0
percent_below_poverty_level    0
dtype: int64
```

[11]:
```
flrando = [fl_pov["geography"].value_counts()]
print(flrando)
```

```
[geography
12001000201    1
12095016904    1
12095016746    1
12095016745    1
12095016744    1
              ..
12057013924    1
12057013923    1
12057013922    1
12057013919    1
12133970303    1
Name: count, Length: 5160, dtype: int64]
```

[12]:
```
fl_risk_additional = fl_risk[~fl_risk['geography'].
  ↪isin(fl_pov['geography'])]['geography'].unique()
additional_values = fl_pov[~fl_pov['geography'].
  ↪isin(fl_risk['geography'])]['geography'].unique()
print(additional_values)
print(fl_risk_additional)
```

```
['12005990000' '12009990000' '12011990000' '12015990000' '12017990000'
 '12021990000' '12029990000' '12031990000' '12033990000' '12035990000'
 '12037990000' '12037990100' '12043990000' '12045990000' '12051990000'
 '12053990000' '12057990000' '12057990100' '12061990000' '12065990000'
 '12071990000' '12075990000' '12081990000' '12085990000' '12085990100'
 '12086990000' '12087990000' '12089990000' '12091990100' '12091990200'
 '12093990000' '12095990000' '12099990000' '12099990100' '12101990000'
 '12103990000' '12103990100' '12109990100' '12109990200' '12111990000'
 '12113990000' '12115990000' '12123990000' '12127990000' '12129990000'
 '12131990000']
[]
```

fl_pov has additional 46 geographical sections that are not in the fl_risk. I will check if they belong in the same trect. If they don't, the extra data can be discarded, but if they do then I need to take them into account.

[13]:
```
fl_risk["TRACT"].head()
```

[13]:
```
0    201
1    202
2    301
3    302
4    400
Name: TRACT, dtype: int64
```

[14]:
```
# Filter rows in fl_risk and fl_pov for the additional_values
fl_risk_additional = fl_risk[fl_risk['geography'].isin(additional_values)]
fl_pov_additional = fl_pov[fl_pov['geography'].isin(additional_values)]

# Check the values in the "trect" column for each dataframe

print(fl_risk_additional['TRACT'].unique())


print(fl_pov_additional['geographic_area_name'].unique())
```

```
[]
['Census Tract 9900; Bay County; Florida'
 'Census Tract 9900; Brevard County; Florida'
 'Census Tract 9900; Broward County; Florida'
 'Census Tract 9900; Charlotte County; Florida'
 'Census Tract 9900; Citrus County; Florida'
 'Census Tract 9900; Collier County; Florida'
 'Census Tract 9900; Dixie County; Florida'
 'Census Tract 9900; Duval County; Florida'
 'Census Tract 9900; Escambia County; Florida'
 'Census Tract 9900; Flagler County; Florida'
 'Census Tract 9900; Franklin County; Florida'
 'Census Tract 9901; Franklin County; Florida'
```

```
 'Census Tract 9900; Glades County; Florida'
 'Census Tract 9900; Gulf County; Florida'
 'Census Tract 9900; Hendry County; Florida'
 'Census Tract 9900; Hernando County; Florida'
 'Census Tract 9900; Hillsborough County; Florida'
 'Census Tract 9901; Hillsborough County; Florida'
 'Census Tract 9900; Indian River County; Florida'
 'Census Tract 9900; Jefferson County; Florida'
 'Census Tract 9900; Lee County; Florida'
 'Census Tract 9900; Levy County; Florida'
 'Census Tract 9900; Manatee County; Florida'
 'Census Tract 9900; Martin County; Florida'
 'Census Tract 9901; Martin County; Florida'
 'Census Tract 9900; Miami-Dade County; Florida'
 'Census Tract 9900; Monroe County; Florida'
 'Census Tract 9900; Nassau County; Florida'
 'Census Tract 9901; Okaloosa County; Florida'
 'Census Tract 9902; Okaloosa County; Florida'
 'Census Tract 9900; Okeechobee County; Florida'
 'Census Tract 9900; Orange County; Florida'
 'Census Tract 9900; Palm Beach County; Florida'
 'Census Tract 9901; Palm Beach County; Florida'
 'Census Tract 9900; Pasco County; Florida'
 'Census Tract 9900; Pinellas County; Florida'
 'Census Tract 9901; Pinellas County; Florida'
 'Census Tract 9901; St. Johns County; Florida'
 'Census Tract 9902; St. Johns County; Florida'
 'Census Tract 9900; St. Lucie County; Florida'
 'Census Tract 9900; Santa Rosa County; Florida'
 'Census Tract 9900; Sarasota County; Florida'
 'Census Tract 9900; Taylor County; Florida'
 'Census Tract 9900; Volusia County; Florida'
 'Census Tract 9900; Wakulla County; Florida'
 'Census Tract 9900; Walton County; Florida']
```

I will be disregarding the additional data because they cannot be calculated in the functions

1.1 perform the ian function to get the ian score

- first get the median percent_below_poverty_level for every county
- get CFLD_RISKS for every county
- get the ian score for every county
- sort them by highest to lowest and get the highest

[15]:
```
non_numeric_values = mixed[~mixed['percent_below_poverty_level'].str.match(r'^-?
↪\d+\.?\d*$')]
print(non_numeric_values['percent_below_poverty_level'])
```

```
12      -
```

```
230     -
262     -
263     -
679     -
1401    -
1402    -
1530    -
1729    -
1730    -
1731    -
1733    -
1734    -
1735    -
1786    -
2104    -
2111    -
2125    -
2199    -
2389    -
2390    -
2823    -
3125    -
3126    -
3128    -
3130    -
3132    -
3133    -
3134    -
3136    -
3165    -
3384    -
3812    -
3919    -
3932    -
3938    -
3939    -
3940    -
3941    -
3942    -
4519    -
4536    -
4652    -
4926    -
4927    -
Name: percent_below_poverty_level, dtype: object
```

[16]:
```
county_values = non_numeric_values['COUNTY']
print(county_values)
```

```
12           Alachua
230          Brevard
262          Brevard
263          Brevard
679          Broward
1401       Highlands
1402       Highlands
1530    Hillsborough
1729    Hillsborough
1730    Hillsborough
1731    Hillsborough
1733    Hillsborough
1734    Hillsborough
1735    Hillsborough
1786    Indian River
2104             Lee
2111           Leon
2125           Leon
2199         Liberty
2389          Marion
2390          Marion
2823      Miami-Dade
3125      Miami-Dade
3126      Miami-Dade
3128      Miami-Dade
3130      Miami-Dade
3132      Miami-Dade
3133      Miami-Dade
3134      Miami-Dade
3136      Miami-Dade
3165          Monroe
3384          Orange
3812      Palm Beach
3919      Palm Beach
3932      Palm Beach
3938      Palm Beach
3939      Palm Beach
3940      Palm Beach
3941      Palm Beach
3942      Palm Beach
4519            Polk
4536          Putnam
4652       St. Lucie
4926          Sumter
```

```
4927          Sumter
Name: COUNTY, dtype: object
```

1.2 screw it, I will just nuke everything that doesnt have data

[17]:
```
# Convert the "percent_below_poverty_level" column to numeric, coercing errors␣
 ↪to NaN
mixed['percent_below_poverty_level'] = pd.
 ↪to_numeric(mixed['percent_below_poverty_level'], errors='coerce')

# Drop rows with missing values in the "percent_below_poverty_level" column
mixed_numeric = mixed.dropna(subset=['percent_below_poverty_level'])

# Convert the column to float type
mixed_numeric['percent_below_poverty_level'] =␣
 ↪mixed_numeric['percent_below_poverty_level'].astype(float)

mixed_numeric = mixed_numeric.reset_index(drop=True)


mixed_cleaned = mixed_numeric
mixed_cleaned.shape
```

```
/tmp/ipykernel_4491/945283858.py:8: 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
  mixed_numeric['percent_below_poverty_level'] =
mixed_numeric['percent_below_poverty_level'].astype(float)
```

[17]:
```
(5069, 118)
```

[18]:
```
# Group the merged dataset by the "county" column and calculate the median of␣
 ↪"percent_below_poverty_level" for each group
county_median_poverty = mixed_cleaned.
 ↪groupby("COUNTY")["percent_below_poverty_level"].median()
len(county_median_poverty)
```

[18]:
```
67
```

risk

[19]:
```
# Convert the "percent_below_poverty_level" column to numeric, coercing errors␣
 ↪to NaN
mixed_cleaned['CFLD_RISKS'] = pd.to_numeric(mixed_cleaned['CFLD_RISKS'],␣
 ↪errors='coerce')
```

```
# Drop rows with missing values in the "percent_below_poverty_level" column
mixed_numeric = mixed_cleaned.dropna(subset=['CFLD_RISKS'])

# Convert the column to float type
mixed_numeric['CFLD_RISKS'] = mixed_numeric['CFLD_RISKS'].astype(float)


mixed_numeric = mixed_numeric.reset_index(drop=True)

mixed_cleaned = mixed_numeric
mixed_cleaned.shape
```

```
/tmp/ipykernel_4491/225427614.py:8: 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
  mixed_numeric['CFLD_RISKS'] = mixed_numeric['CFLD_RISKS'].astype(float)
```

[19]:
```
(4744, 118)
```

[20]:
```
# Group the merged dataset by the "county" column and calculate the median of␣
 ↪"percent_below_poverty_level" for each group
county_median_risk = mixed_cleaned.groupby("COUNTY")["CFLD_RISKS"].median()
county_median_risk
a = county_median_risk.sort_values(ascending=False)
```

1.3 cleaning the social vulnerability score

[21]:
```
# cleaning the soviet score
mixed_cleaned['SOVI_SCORE'] = pd.to_numeric(mixed_cleaned['SOVI_SCORE'],␣
 ↪errors='coerce')
mixed_numeric = mixed_cleaned.dropna(subset=['SOVI_SCORE'])
mixed_numeric['SOVI_SCORE'] = mixed_numeric['SOVI_SCORE'].astype(float)
mixed_numeric = mixed_numeric.reset_index(drop=True)
mixed_cleaned = mixed_numeric
```

1.4 cleaning the resiliance score

[22]:
```
# cleaning the soviet score
mixed_cleaned['RESL_SCORE'] = pd.to_numeric(mixed_cleaned['RESL_SCORE'],␣
 ↪errors='coerce')
mixed_numeric = mixed_cleaned.dropna(subset=['RESL_SCORE'])
mixed_numeric['RESL_SCORE'] = mixed_numeric['RESL_SCORE'].astype(float)
```

```
mixed_numeric = mixed_numeric.reset_index(drop=True)
mixed_cleaned = mixed_numeric
```

1.5 Making the ian score tract model

GETTING THE IAN SCORE

[23]:
```
# defining the variables for the tracts model
tract_risk = mixed_cleaned[["TRACT", "CFLD_RISKS"]]
tract_percent_below_poverty_level = mixed_cleaned[["TRACT",␣
 ↪"percent_below_poverty_level"]]
tract_sovi = mixed_cleaned[["TRACT", "SOVI_SCORE"]]
tract_resl = mixed_cleaned[["TRACT", "RESL_SCORE"]]


tract_ianscore = 0.17 *␣
 ↪tract_percent_below_poverty_level["percent_below_poverty_level"] + 0.16 *␣
 ↪tract_sovi["SOVI_SCORE"] + 0.5 * tract_risk["CFLD_RISKS"]+ (0.17␣
 ↪*(100-tract_resl["RESL_SCORE"]))

# Concatenate the Ian scores and tracts into a single DataFrame
ianscores_with_tracts = pd.concat([tract_ianscore, mixed_cleaned["TRACT"],␣
 ↪mixed_cleaned["geographic_area_name"], mixed_cleaned["geography"]], axis=1)


# Sort the DataFrame based on Ian scores
ianscores_sorted = ianscores_with_tracts.sort_values(by=0, ascending=False)
ianscores_sorted
```

[23]:

	0	TRACT	geographic_area_name	\
944	83.375391	970201	Census Tract 9702.01; Dixie County; Florida	
4193	81.833197	951300	Census Tract 9513; Putnam County; Florida	
4188	81.785894	950800	Census Tract 9508; Putnam County; Florida	
2505	81.731014	3001	Census Tract 30.01; Miami-Dade County; Florida	
1396	81.258072	4300	Census Tract 43; Hillsborough County; Florida	
…	…	…	…	
2082	6.676100	2418	Census Tract 24.18; Leon County; Florida	
2084	6.586300	2420	Census Tract 24.20; Leon County; Florida	
2032	6.519100	501	Census Tract 5.01; Leon County; Florida	
2051	6.488700	1701	Census Tract 17.01; Leon County; Florida	
2078	6.346500	2413	Census Tract 24.13; Leon County; Florida	

	geography
944	12029970201
4193	12107951300
4188	12107950800
2505	12086003001

```
1396  12057004300
…           …
2082  12073002418
2084  12073002420
2032  12073000501
2051  12073001701
2078  12073002413

[4744 rows x 4 columns]
```

[24]:
```
# Filter mixed_cleaned DataFrame to include only tracts in Dixie County
dixie_tracts = mixed_cleaned[mixed_cleaned['COUNTY'] == 'Dixie']

# Calculate the IAN score for Dixie County tracts
dixie_tract_ianscore = (
    0.17 * dixie_tracts["percent_below_poverty_level"] +
    0.16 * dixie_tracts["SOVI_SCORE"] +
    0.5 * dixie_tracts["CFLD_RISKS"] +
    (0.17 *(100-dixie_tracts["RESL_SCORE"]))
)

# Concatenate the IAN scores and tracts into a single DataFrame
dixie_ianscores_with_tracts = pd.concat([dixie_tract_ianscore,␣
 ↪dixie_tracts["TRACT"], dixie_tracts["geographic_area_name"],␣
 ↪dixie_tracts["geography"]], axis=1)
dixie_ianscores_with_tracts = dixie_ianscores_with_tracts.rename(columns={0:␣
 ↪"ianscore"})

# Sort the DataFrame based on IAN scores
dixie_ianscores_sorted = dixie_ianscores_with_tracts.sort_values(by="ianscore",␣
 ↪ascending=False)
dixie_ianscores_sorted
```

[24]:
```
      ianscore   TRACT                         geographic_area_name  \
944  83.375391  970201  Census Tract 9702.01; Dixie County; Florida
945  75.028273  970202  Census Tract 9702.02; Dixie County; Florida
941  72.426687  970101  Census Tract 9701.01; Dixie County; Florida
942  70.147638  970103  Census Tract 9701.03; Dixie County; Florida
943  63.325255  970104  Census Tract 9701.04; Dixie County; Florida

       geography
944  12029970201
945  12029970202
941  12029970101
942  12029970103
943  12029970104
```

2 We chose to relocate tract 970201 in dixie FL

2.1 will now get the heatmap of the ian sc

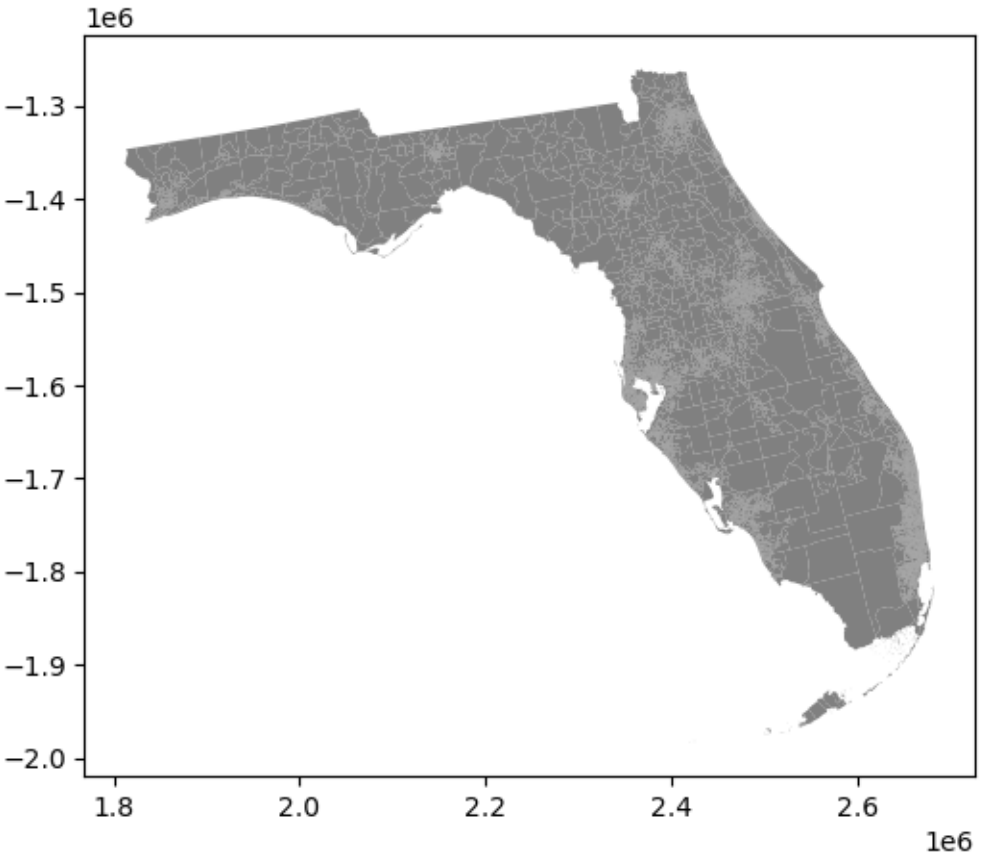
[25]:
```
import geopandas as gp
from pygris import tracts
```

[26]:
```
# Coordinate system
fl_tracts = tracts("FL", cb = True,
                   year = 2021,
                   cache = True).to_crs(6571)

heatmap_tracts = fl_tracts.rename(columns={"GEOID": "geography"})

# merge heatMap tracts with the ianscore data
ianscore_heatmap = heatmap_tracts.merge(ianscores_with_tracts, on="geography",␣
 ↪how='inner')
ianscore_dixie_heatmap = heatmap_tracts.merge(dixie_ianscores_with_tracts,␣
 ↪on="geography", how='inner')
```

```
Using FIPS code '12' for input 'FL'
```

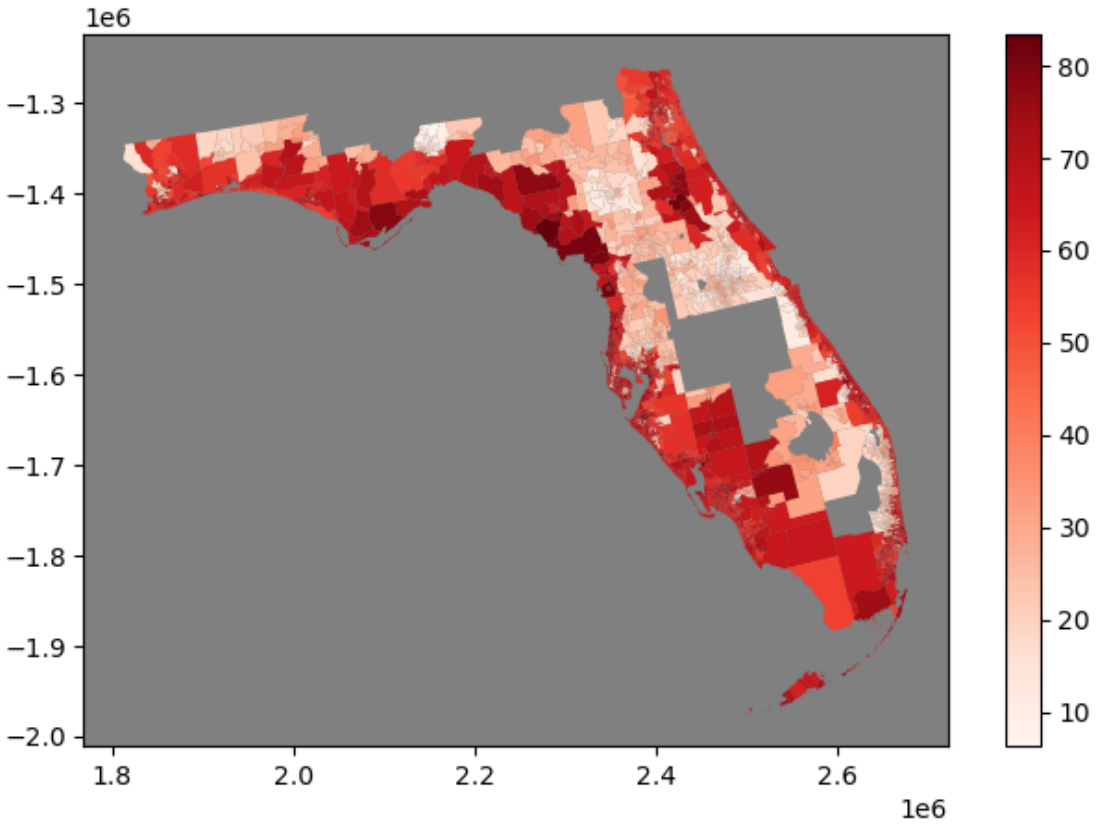
[27]:
```
fig, ax = plt.subplots(figsize = (8, 5))

fl_tracts.plot(ax = ax, color = "grey")
```

[27]:
```
<Axes: >
```



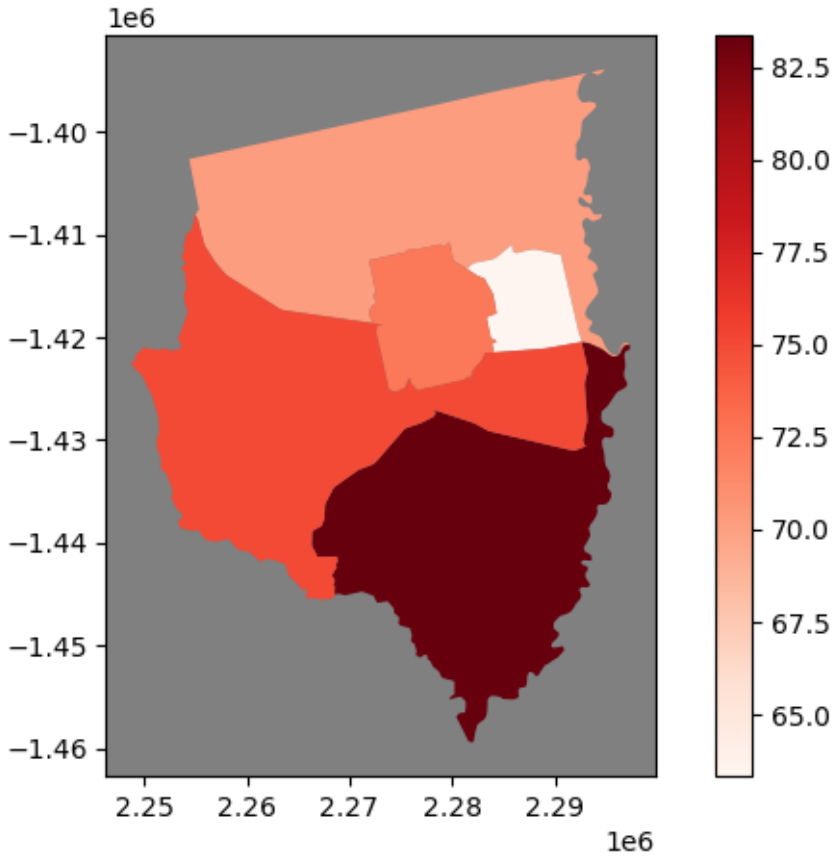
[28]:
```
fig, ax = plt.subplots(figsize = (8, 5))
ax.set_facecolor('grey')

ianscore_heatmap.plot(column=0, cmap='Reds', legend=True, ax=ax)
```

[28]: <Axes: >



[29]:
```
fig, ax = plt.subplots(figsize = (8, 5))
ax.set_facecolor('grey')

ianscore_dixie_heatmap.plot(column='ianscore', cmap='Reds', legend=True, ax=ax)
```

[29]: <Axes: >



2.2 the cleaned data will be called bryanscore

3 GET THE relocation SCORE

[ ]:
```
# bryanscore = fl_risk.merge(fl_pov, on="geography", how='inner')
# bryanscore = bryanscore.merge(fl_income, on="geography", how='inner')
# bryanscore = bryanscore.merge(fl_fin, on="geography", how='inner')
# bryanscore = bryanscore.merge(fl_housing, on="geography", how='inner')


# Merge DataFrames with custom suffixes
bryanscore = fl_risk.merge(fl_pov, on="geography", how='inner',␣
 ↪suffixes=('_risk', '_pov'))
bryanscore = bryanscore.merge(fl_income, on="geography", how='inner',␣
 ↪suffixes=('_pov', '_income'))
bryanscore = bryanscore.merge(fl_fin, on="geography", how='inner',␣
 ↪suffixes=('_income', '_fin'))
bryanscore = bryanscore.merge(fl_housing, on="geography", how='inner',␣
 ↪suffixes=('_fin', '_housing'))
```

```
# Drop duplicated columns if they exist
if 'geographic_area_name_x' in bryanscore.columns and 'geographic_area_name_y'␣
 ↪in bryanscore.columns:
    bryanscore.drop(columns=['geographic_area_name_x',␣
 ↪'geographic_area_name_y'], inplace=True)
bryanscore
```

[ ]:
```
# cleaning the families_median_income
bryanscore['families_median_income'] = pd.
 ↪to_numeric(bryanscore['families_median_income'], errors='coerce')
bryanscore = bryanscore.dropna(subset=['families_median_income'])
bryanscore['families_median_income'] = bryanscore['families_median_income'].
 ↪astype(float)
bryanscore = bryanscore.reset_index(drop=True)

# cleaning the EAL_SCORE
bryanscore['EAL_SCORE'] = pd.to_numeric(bryanscore['EAL_SCORE'],␣
 ↪errors='coerce')
bryanscore = bryanscore.dropna(subset=['EAL_SCORE'])
bryanscore['EAL_SCORE'] = bryanscore['EAL_SCORE'].astype(float)
bryanscore = bryanscore.reset_index(drop=True)

# cleaning the total_housing_units
bryanscore['total_housing_units'] = pd.
 ↪to_numeric(bryanscore['total_housing_units'], errors='coerce')
bryanscore = bryanscore.dropna(subset=['total_housing_units'])
bryanscore['total_housing_units'] = bryanscore['total_housing_units'].
 ↪astype(float)
bryanscore = bryanscore.reset_index(drop=True)

# cleaning the vacancy_housing_units
bryanscore['vacant_housing_units'] = pd.
 ↪to_numeric(bryanscore['vacant_housing_units'], errors='coerce')
bryanscore = bryanscore.dropna(subset=['vacant_housing_units'])
bryanscore['vacant_housing_units'] = bryanscore['vacant_housing_units'].
 ↪astype(float)
bryanscore = bryanscore.reset_index(drop=True)

# cleaning the risk
bryanscore['CFLD_RISKS'] = pd.to_numeric(bryanscore['CFLD_RISKS'],␣
 ↪errors='coerce')
bryanscore = bryanscore.dropna(subset=['CFLD_RISKS'])
bryanscore['CFLD_RISKS'] = bryanscore['CFLD_RISKS'].astype(float)
bryanscore = bryanscore.reset_index(drop=True)
bryanscore
```

[ ]:
```
#define variables
chosen_tract = bryanscore[bryanscore["geography"] == "12029970201"]
chosen_tract
```

[ ]:
```
d_tract= fl_tracts[fl_tracts["GEOID"]=="12029970201"]
d_tract
```

[ ]:
```
# tract_centroids = fl_tracts.centroid
# dist = d_tract.centroid.geometry.apply(lambda g: tract_centroids.distance(g,␣
 ↪align = False))


tract_centroids = fl_tracts.centroid
distances = tract_centroids.distance(d_tract.centroid.iloc[0], align=False)

# Create a DataFrame with distances and desired columns
distances_df = pd.DataFrame({
    'geography': fl_tracts["GEOID"],
    'COUNTY': fl_tracts["NAMELSADCO"],
    'Distance': distances
})

# Display the resulting DataFrame
distances_df
s = distances_df.sort_values(by="Distance", ascending=True)
s
```

[ ]:
```
# Merge bryanscore DataFrame with distances_df
bryanscore = bryanscore.merge(distances_df, on="geography", how='inner')

# Drop duplicated columns manually
columns_to_drop = [col for col in bryanscore.columns if '_fin' in col and '_df'␣
 ↪in col]
bryanscore.drop(columns=columns_to_drop, inplace=True)

# Check for and drop any remaining duplicate columns
duplicate_columns = bryanscore.columns[bryanscore.columns.duplicated()]
bryanscore.drop(columns=duplicate_columns, inplace=True)
bryanscore
```

[ ]:
```
# Extract necessary columns
tract_families_median_income = bryanscore[["TRACT", "families_median_income"]]
tract_EAL_SCORE = bryanscore[["TRACT", "EAL_SCORE"]]
tract_vacant_housing_units = bryanscore[["TRACT", "vacant_housing_units"]]
tract_total_housing_units = bryanscore[["TRACT", "total_housing_units"]]
tract_risk2 = bryanscore[["TRACT", "CFLD_RISKS"]]
tract_distance = bryanscore[["TRACT", "Distance"]]
```

```
# Calculate the relocationscore for each tract
tract_relocationscore = (
    0.20 * abs((bryanscore["families_median_income"] -␣
 ↪tract_families_median_income["families_median_income"]) /␣
 ↪bryanscore["families_median_income"]) +
    0.05 * abs((bryanscore["EAL_SCORE"] - tract_EAL_SCORE["EAL_SCORE"]) /␣
 ↪bryanscore["EAL_SCORE"]) +
    0.20 * (tract_vacant_housing_units["vacant_housing_units"] /␣
 ↪tract_total_housing_units["total_housing_units"]) +
    0.3 * tract_risk2["CFLD_RISKS"] +
    0.25 * (tract_distance["Distance"]/590137.928567)
)

# Concatenate the relocationscores and tracts into a single DataFrame
relocationscore_with_tracts = pd.concat([tract_relocationscore,␣
 ↪bryanscore["TRACT"], bryanscore["COUNTY_x"], bryanscore["geography"]],␣
 ↪axis=1)

# Rename the columns
relocationscore_with_tracts.columns = ["relocationscore", "TRACT", "COUNTY_x",␣
 ↪"geography"]

# Sort the DataFrame based on relocationscores
relocationscores_sorted = relocationscore_with_tracts.
 ↪sort_values(by="relocationscore", ascending=True)
relocationscores_sorted
```

[ ]:
```
relocationscore_heatmap = heatmap_tracts.merge(relocationscore_with_tracts,␣
 ↪on="geography", how='inner')
```

[ ]:
```
# Create the figure and axis
fig, ax = plt.subplots(figsize=(10, 10))
ax.set_facecolor('grey')

# Plot the heatmap with inverted colormap
relocationscore_heatmap.plot(column="relocationscore", cmap='Reds_r',␣
 ↪legend=True, ax=ax)

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

[ ]:
```
relocationscore_heatmap_less =␣
 ↪relocationscore_heatmap[relocationscore_heatmap["relocationscore"] <0.1]

# Create the figure and axis
```

```
fig, ax = plt.subplots(figsize=(10, 10))
ax.set_facecolor('grey')

# Plot the heatmap with inverted colormap
relocationscore_heatmap_less.plot(column="relocationscore", cmap='Reds_r',␣
 ↪legend=True, ax=ax)

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

[ ]:
```
relocationscore_heatmap_scope =␣
 ↪relocationscore_heatmap[relocationscore_heatmap["relocationscore"] <0.06]

# Create the figure and axis
fig, ax = plt.subplots(figsize=(10, 10))
ax.set_facecolor('grey')

# Plot the heatmap with inverted colormap
relocationscore_heatmap_scope.plot(column="relocationscore", cmap='Reds_r',␣
 ↪legend=True, ax=ax)

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

[ ]:
```

```