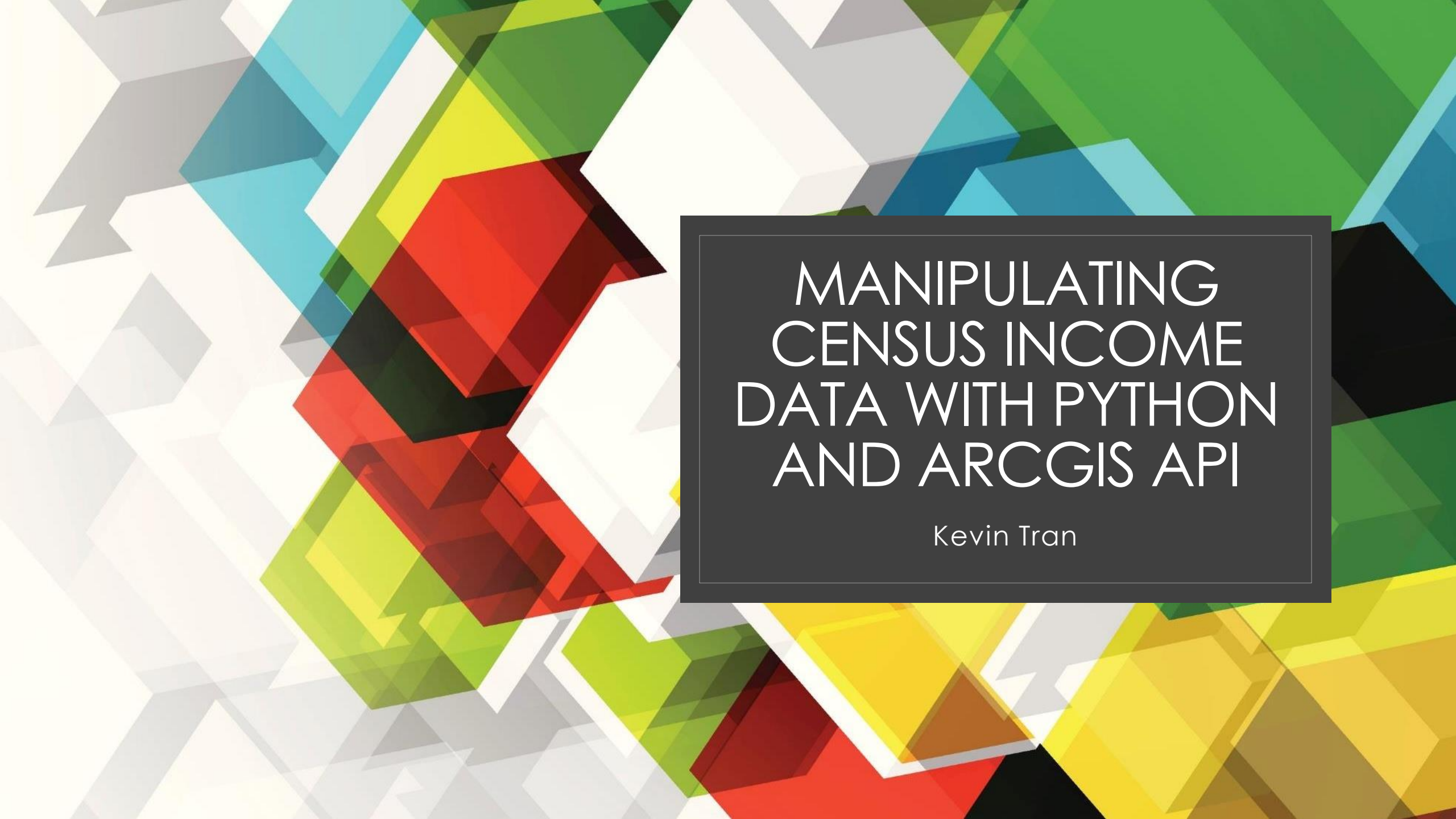
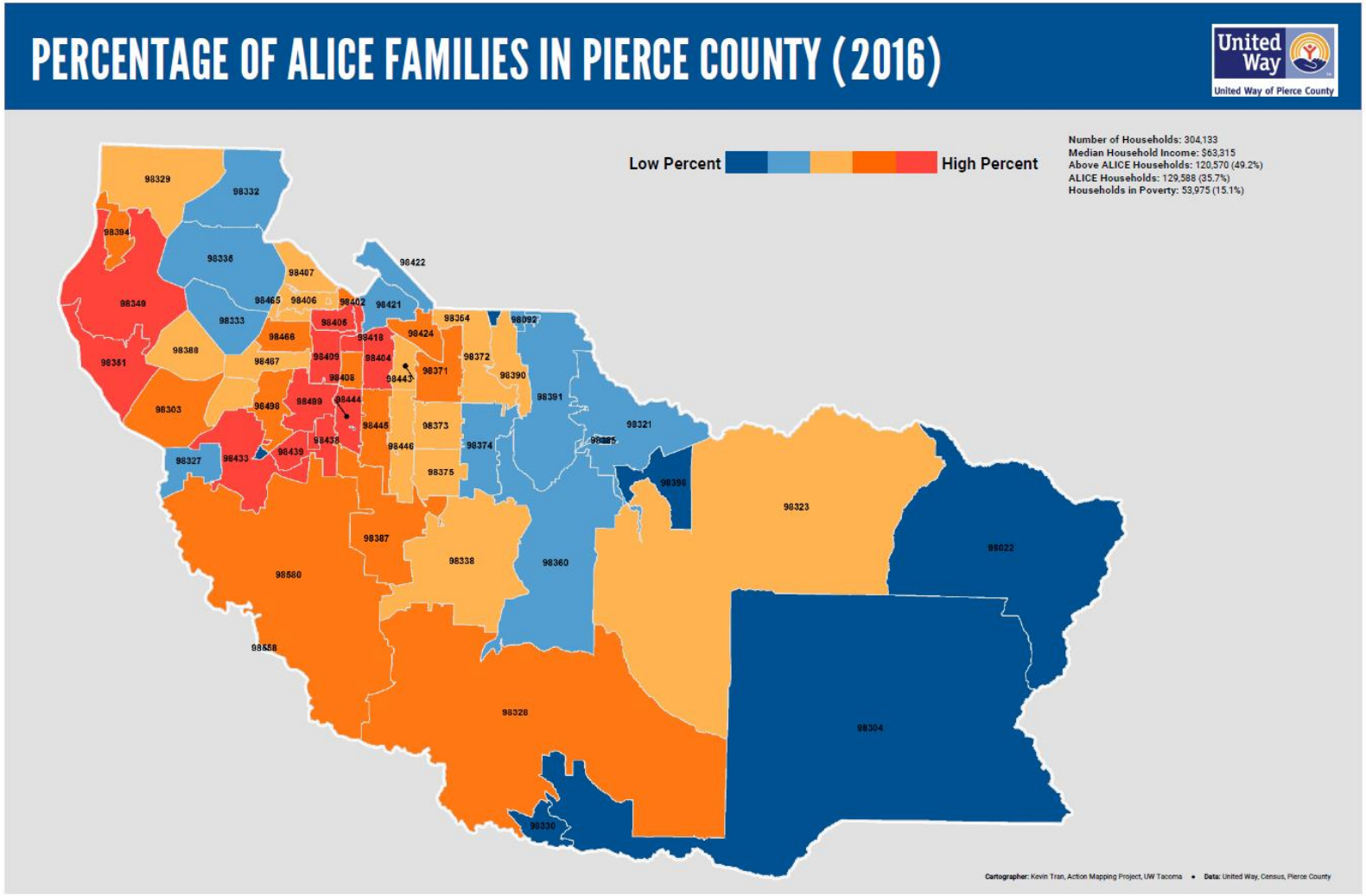
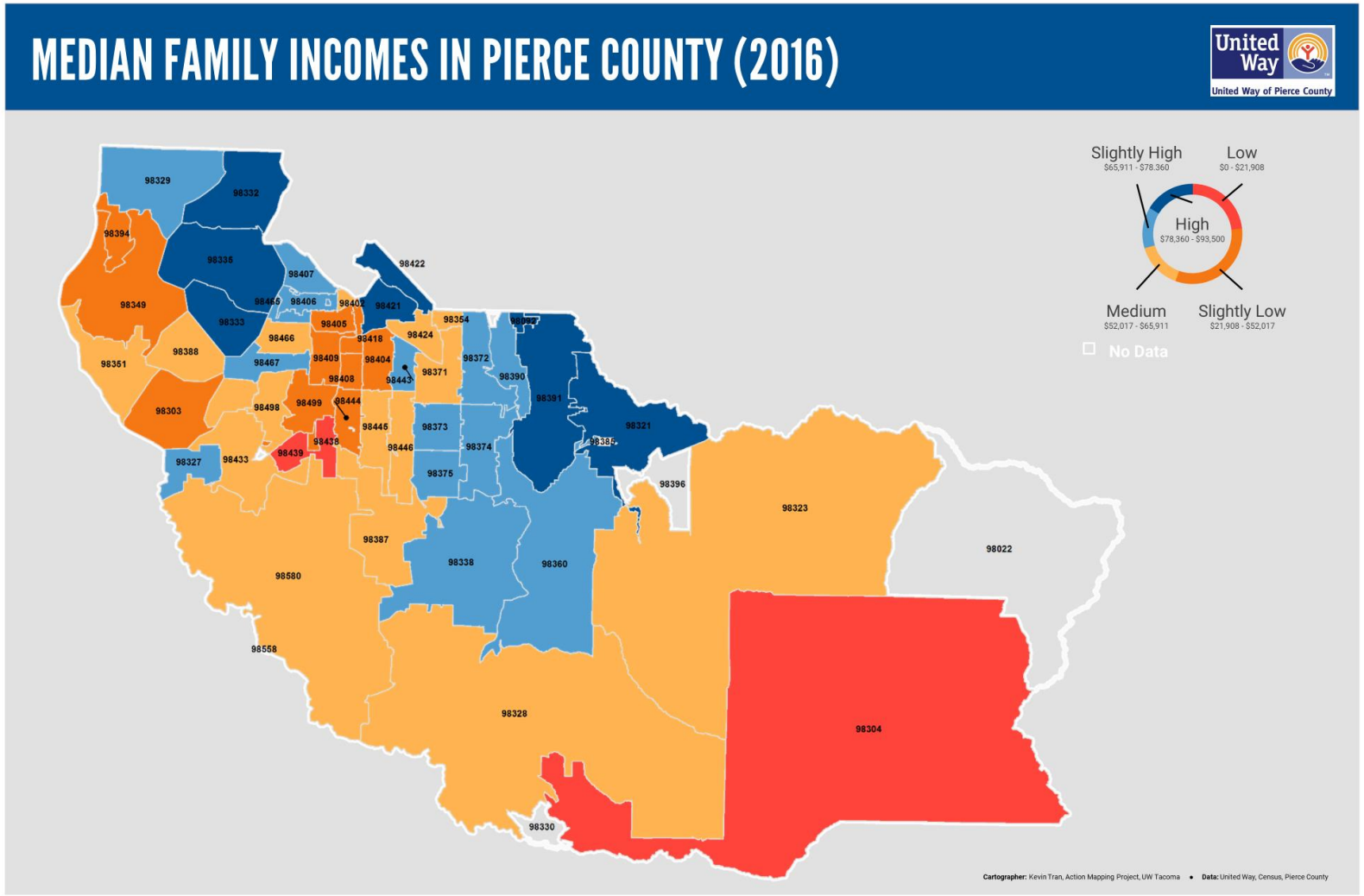

MANIPULATING CENSUS INCOME DATA WITH PYTHON AND ARCGIS API
Kevin Tran



Background

As a project administrator for Action Mapping Project (A.M.P.), I was tasked with designing maps for United Way. My capstone is inspired by the work I did.

Question: Is there a way to automate data manipulation and visualization of Census data?



Background

Python libraries:

- Pandas
- Geopandas
- Zipfile
- ArcGIS API

Materials:

Census Data with:

- Columns reflecting income brackets
- It must be a CSV
- It must have a column head that matches the column head of a shapefile and similar values if able from each.

TIGER/LINE Shapefiles

How does it work?

Step 1: Unzip your files.

```
# STEP 1

import pandas
from zipfile import ZipFile
# specifying the zip file name
file_name = "block_group_data.zip"

# opening the zip file in READ mode
with ZipFile(file_name, 'r') as zip:
    # printing all the contents of the zip file
    zip.printdir()

    # extracting all the files
    print('Extracting all the files now...')
    zip.extractall()
    print('Done!')

file_name = "block_groups.zip"

# opening the zip file in READ mode
with ZipFile(file_name, 'r') as zip:
    # printing all the contents of the zip file
    zip.printdir()

    # extracting all the files
    print('Extracting all the files now...')
    zip.extractall()
    print('Done!')
```

Step 2: Manipulate Data via Pandas.

```
# STEP 2

x = pandas.read_csv('ACS_16_5YR_B19101_with_ann.csv', header=1)

x_drop = x.drop(x.columns[[4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36]], axis = 1)

x_drop['Poverty_Count'] = x_drop.iloc[:,4:8].sum(axis=1)

x_drop['ALICE_Count'] = x_drop.iloc[:,8:15].sum(axis=1)

x_drop['ALICE_Beyond'] = x_drop.iloc[:,15:20].sum(axis=1)

new_table = x_drop[['Id', 'Id2', 'Geography', 'Poverty_Count', 'ALICE_Count', 'ALICE_Beyond']]

new_table.rename(columns = {'Id2':'GEOID10'}, inplace = True)

database = pandas.DataFrame(new_table)

database.GEOID10.astype('int64')
```

How does it work?

Step 3: Manipulate Data via GeoPandas then zip up files .

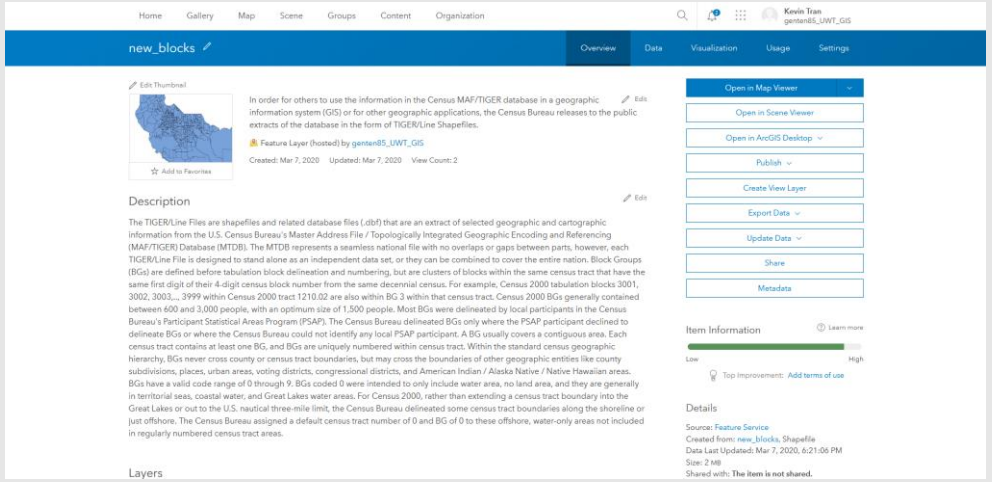
```
# STEP 3

gdf = geopandas.read_file('tl_2010_53053_bg10.shp')

shapefile = pandas.DataFrame(gdf)

shapefile['GEOID10']=shapefile['GEOID10'].astype('int64')

newgeo = shapefile.merge(database, on='GEOID10')

ALICE = geopandas.GeoDataFrame(newgeo, geometry='geometry')

# https://geopandas.org/io.html
ALICE.to_file('tl_2010_53053_bg10.shp',driver ='ESRI Shapefile')

# //ZIP FILE//

def main():

    file_name = 'new_blocks.zip'

    # opening the 'Zip' in writing mode
    with zipfile.ZipFile(file_name, 'a') as file:
        # append mode adds files to the 'Zip'
        # you have to create the files which you have to add to the 'Zip'
        file.write('tl_2010_53053_bg10.cpg')
        file.write('tl_2010_53053_bg10.dbf')
        file.write('tl_2010_53053_bg10.shp')
        file.write('tl_2010_53053_bg10.shp.xml')
        file.write('tl_2010_53053_bg10.shx')
        print('Files added to the Zip')

    # opening the 'Zip' in reading mode to check
    with zipfile.ZipFile(file_name, 'r') as file:
        print(file.namelist())

if __name__ == '__main__': main()
```

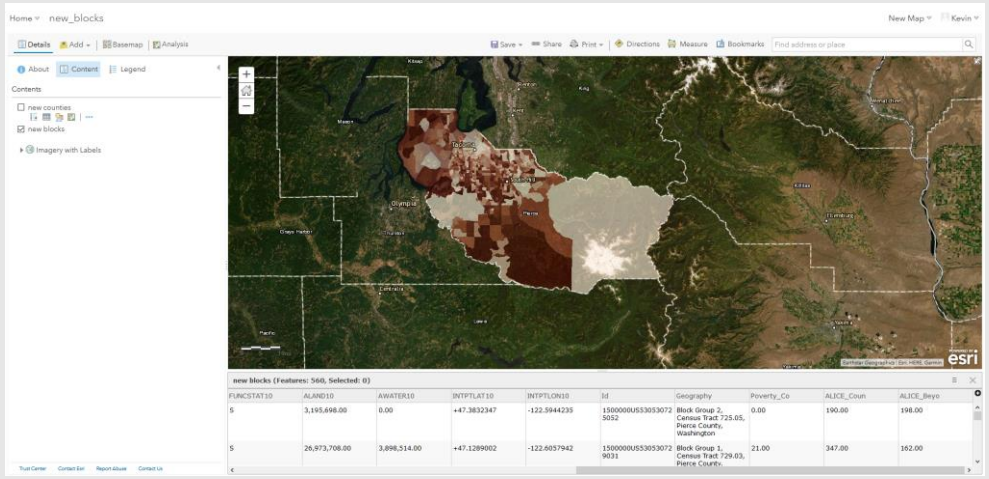
Step 4: Publish shapefiles using ArcGIS Python API

```
from arcgis.gis import arcgis

gis = arcgis.GIS("https://www.arcgis.com", username="ENTER_USERNAME", password="ENTER_PASSWORD")

data = "new_blocks.zip"
shpfile = gis.content.add({}, data)
published_service = shpfile.publish()
display(published_service)
```

Outcomes:

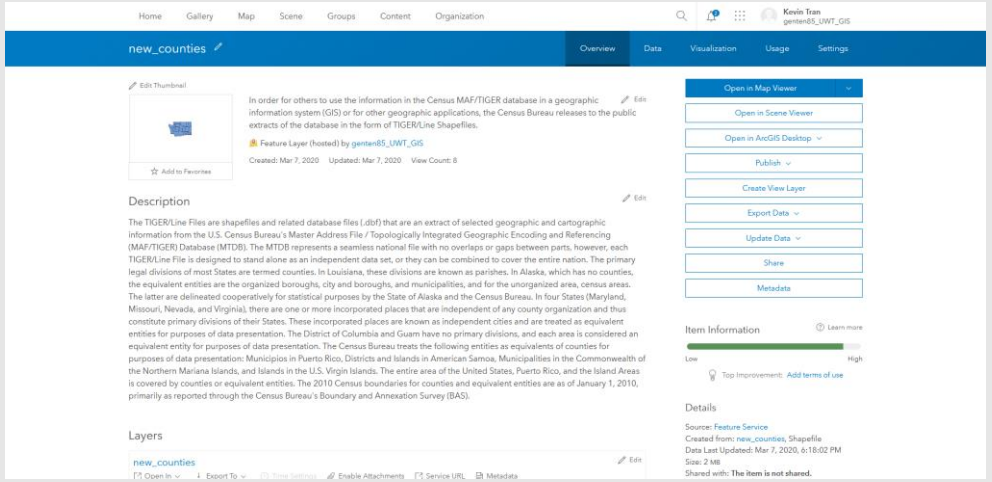
Block Group Feature Layer

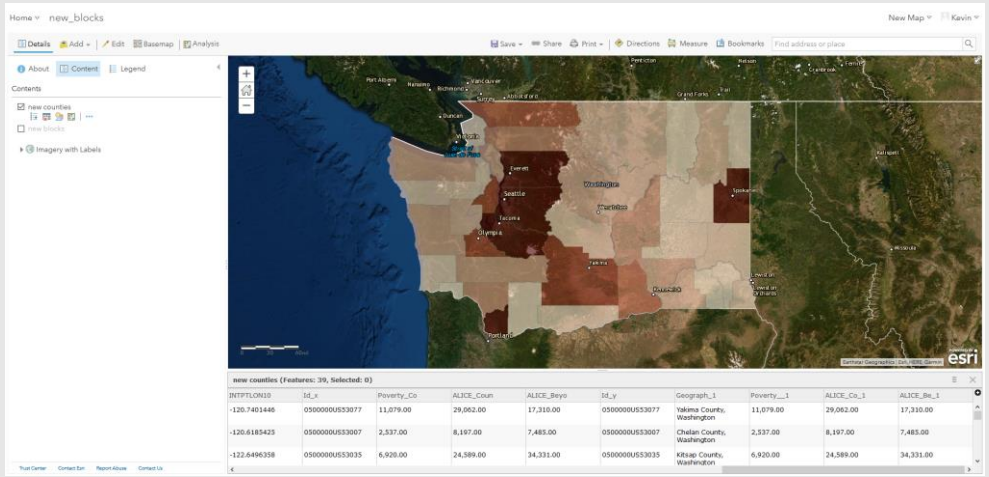




Outcomes:

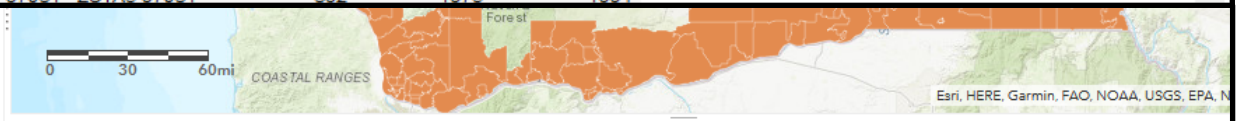
County-Lv Feature Layer





```
x_drop['ALICE_Beyond'] = x_drop.iloc[:,15:20].sum(axis=1)

new_table = x_drop[['Id', 'Id2', 'Geography', 'Poverty_Count', 'ALICE_Count', 'ALICE_Beyond']]

new_table.rename(columns = {'Id2':'GEOID10'}, inplace = True)

database = pandas.DataFrame(new_table)

database.GEOID10.astype('int64')

database
```

	Id	GEOID10	Geography	Poverty_Count	ALICE_Count	ALICE_Beyond
0	8600000US83501	83501	ZCTA5 83501	1279	4537	3393
1	8600000US83822	83822	ZCTA5 83822	83	254	120
2	8600000US83824	83824	ZCTA5 83824	27	46	17
3	8600000US83832	83832	ZCTA5 83832	37	170	216
4	8600000US83851	83851	ZCTA5 83851	104	222	81
5	8600000US83855	83855	ZCTA5 83855	85	375	160
6	8600000US83870	83870	ZCTA5 83870	18	37	17
7	8600000US83872	83872	ZCTA5 83872	7	14	145
8	8600000US97031	97031	ZCTA5 97031	392	1975	1994



zip boundaries (Features: 598, Selected: 0)

STATEFP10	ZCTA5CE10	GEOID10	CLASSFP10	MTFCC10	FUNCSTAT10
53	98822	5398822	B5	G6350	S
53	98821	5398821	B5	G6350	S
53	98357	5398357	B5	G6350	S
53	98663	5398663	B5	G6350	S
53	98684	5398684	B5	G6350	S
53	98848	5398848	B5	G6350	S
53	99166	5399166	B5	G6350	S
53	98851	5398851	B5	G6350	S
53	99103	5399103	B5	G6350	S

Trust Center Contact Esri Report Abuse
Contact Us

Strengths and Limitations

Strengths

- Simple and effective way to generate shapefiles and visualize data.
- Gets away with using ArcPy package.
- It is possible in theory when customizing code to manipulate and visualize other Census data. However, one must know what data and tools to mess with.

Limitations

- Not so much a limitation, but the code is roughly written. There is yet a way to store zip file names in a dictionary to use for re-zipping files.
- For this specific code, it presumes that column headers for shapefiles used do not change.

< One of the column headers need to change. This is manipulating zip boundary data.

THANKS FOR VIEWING!!!

Any questions?