EC1410-Fall 2026
Problem Set 2.AI

(Updated 31 August 2026)

Matt Turner

(100 points possible)

This problem set asks you to use U.S. Census data to trace fifty years of population density gradients for Providence.

If you get stuck programming, copy the full error message into an LLM chat window and ask for help.

1. Get NHGIS API Access. Census tract data back to 1970 is available through the **National Historical Geographic Information System (NHGIS)**. Complete this step first.

   (a) Go to `https://account.ipums.org/users/register` and create a free IPUMS account using your university email address.

   (b) Once logged in, navigate to `https://account.ipums.org/api_keys` and generate an API key.

   (c) Store it as a permanent environment variable. Ask an LLM: *"How do I permanently set an environment variable called `IPUMS_API_KEY` to [your key] on [Mac / Windows]?"* and follow the instructions.

   (d) Close and reopen your terminal, then confirm:

Mac / Linux	`echo $IPUMS_API_KEY`
Windows	`echo %IPUMS_API_KEY%`

   You should see your key printed back. If you see nothing, revisit this step.

2. To complete this exercise, your program needs to read U.S. Census maps extract tract areas and centroid coordinates. This requires the ability to process a geospatial data format called a 'shapefile', which is basically a line drawing of a map. These files are tricky to work with, and the most common python toolbox for doing so is called `Geopandas`. `Geopandas` is famously fussy to install, and contains more tools than we need. Instead,we're going to rely on `pyshp`, a much smaller toolbox. This means that you need to tell the agent to use `pyshp`. It will use `Geopandas` otherwise.

   Open an Anaconda Prompt (Windows) or Terminal (Mac) and install the `pyshp` package into your existing python environment:

   ```
   conda install -n urbanecon -c conda-forge pyshp -y
   ```

3. In the same Anaconda Prompt or Terminal, navigate to your `echw` folder and create a folder for this problem set:

```
cd Desktop/echw
mkdir pset2
cd pset2
```

4. Activate your python environment and call/execute the agent.

```
conda activate urbanecon
claude
```

   You should see the agent's welcome prompt.

5. Give the agent the following prompt.

> *I want to plot population density gradients for Providence, RI for the decennial census years 1970, 1980, 1990, 2000, 2010, and 2020.*
>
> ***Population data.*** *Use the NHGIS API (documented at* https://developer.ipums.org/docs/v2/apiref/*) with the key in* `IPUMS_API_KEY` *to download total population for all census tracts in Rhode Island for each of the six years.*
>
> ***Geometry.*** *For each year, download the corresponding NHGIS census tract shapefile for Rhode Island directly from the NHGIS website using* `requests`*. Read the shapefile using the* `shapefile` *module (*`import shapefile`*; installed as* `pyshp`*). Do not use* `geopandas`*. For each tract polygon, compute the centroid using the standard area-weighted centroid formula on the raw projected coordinates, and compute the land area in square meters using the shoelace formula. Both computations should use the polygon vertex coordinates read directly from the* `.shp` *file; do not rely on pre-computed attribute fields such as* `INTPTLAT` *or* `ALAND`*, which may be absent in the earlier-year files. NHGIS shapefiles are projected in Albers Equal Area (meters), so the shoelace formula gives area in square meters directly.*
>
> ***Distance.*** *Convert the Albers centroid coordinates to latitude/longitude using the standard inverse Albers projection. Downtown Providence is at 41.8240° N, 71.4128° W. Compute the straight-line distance from each tract centroid to this point using the haversine formula. Retain only tracts whose centroid is within 30 km of downtown. Express distances in kilometers and density in persons per square kilometer.*
>
> ***Figure.*** *Produce a single figure with one LOWESS smoother per census year (six curves total), plotting log population density on the y-axis and distance to CBD in kilometers on the x-axis. Use a distinct color for each decade, label the axes fully, and include a legend. Save the figure as* `gradients.png`*.*
>
> *After producing the figure, print a summary table with columns: Year, Tract count, Median density (persons per km²), Min distance (km), Max distance (km). Save the table as* `gradient_summary.csv`*.*

This task involves more API calls and file downloads than Problem Set 1 and will take longer. The agent will explore the NHGIS API structure, download tabular data for

six decades, and fetch shapefiles. Let it run. If it gets stuck on an error, copy the message into a chat window and ask for help.

6. When you ask computers to do something, they do something. The trick is figuring out whether they did what you intended. This is true for all programming, but with agents, it's worse.

   One way to check that your program is doing what you intend is to construct test or synthetic data such that, if the code does what you want, you know the answer. To do this, give the agent a prompt like this,

   > *We want to construct a synthetic dataset to test the gradient-plotting pipeline.*
   >
   > *Copy the census map from 1970 to a test file and relabel this year as 1900. From here out, 1900 will be a test year. For every tract in the 1900 map, replace population density with a fake number as follows. For any tract with distance between 0 and 1 mile to the CBD, assign the tract population density 1. For any tract with distance between 1 and 2 to the CBD, assign population density 2. For and tract with distance to the CBD between 2 and three assign density 3. For any other distance, assign density 5.*
   >
   > *Run this synthetic dataset through the same LOWESS gradient-plotting function you wrote above. On the same axes, plot the synthetic data gradient* $\log D(x)$ *as a dashed black line. Save the figure as* `synthetic_test.png`.

7. When you ask computers to do something, they do something. The trick is figuring out whether they did what you intended. This is true for all programming, but with agents, it's worse.

   One way to check that your program is doing what you intend is to actually read the program. Sometimes you will want to do this, but an easier way is to ask an agent to read the code for you.

   If you look in your code directory, somewhere in the pset2 folder, you should see the python program that the agent wrote to make your plots. It should have an intuitive name like 'pop_gradients.py' or 'ps2.py'. Copy this file into the chat window of a different LLM and ask it to read the program and tell you what it does.

8. When you ask computers to do something, they do something. The trick is figuring out whether they did what you intended. This is true for all programming, but with agents, it's worse.

   An easy way to get in trouble is if the data is not what you think it is. This prompt suggests a few checks for census data.

   > *Using the tract-level dataset assembled in the previous step, carry out the validation checks below. Save a plain-text report of the findings as* `validation_report.txt`.

Check 1 — Zero and missing population. *For each census year, count how many tracts have population recorded as zero or as missing. Report both the raw count and the share of all tracts in the 30-km window for that year.*

Check 2 — Implausible density. *For each census year, flag tracts whose population densities take implausible values, either very low or very high.*

Check 3 — Inconsistent coverage across years.

Identify tract identifiers that appear in the dataset for some census years but not others. Report: (a) how many identifiers appear in all six years; (b) how many appear in fewer than six years; (c) for each such identifier, which years it is present and which it is absent. Note that census tract boundaries are redrawn between censuses, so some churn is expected.

For each check, state clearly whether the flagged observations were dropped or retained in the gradient plot produced in Step 4, and why.

Check 4 — heat maps.

Make heat maps showing population density for each year of the data.

We would like you to hand in a short summary of what you have found. This summary should show,

1. (25) `gradients.png` and, in a few sentences, describe how the density gradient changes between 1970 and 2020. What does this imply about where population growth (or decline) was concentrated? Does this confirm the monocentric city model?
2. (25) the plot of synthetic data. Explain whether this confirms that your code did what you intended.
3. (25) the response of the second LLM when you gave it your program and asked what it does. Does this confirm that your program is doing what you intended?
4. (25) Finally, describe the results of your checks on the census data and show at least one heat map. Does this confirm that the data are what you think they should be?