EC1410-Fall 2026
Problem Set 3.AI

(Updated 19 November 2026)

Matt Turner

(100 points possible)

This problem set asks you to calibrate a spatial model of Providence and then use it to evaluate the effects of a counterfactual transit corridor (https://www.providenceri.gov/enhanced-transit-corridor-downtown-providence/) proposed by the (past) Mayor of Providence, Jorge Elorza. The problem set has two main questions. In Problem 1 you use a coding agent to calibrate a spatial model of Providence. In Problem 2, with a coding agent, you use the calibrated model to evaluate the proposed transit corridor.

The zip file that accompanies this problem set contains six files.

1. `Providence_tracts_travel_time_matrix_baseline.csv` for Problem 1. This gives travel times in minutes between each of the 42 census tracts that lie in both municipal Providence and Providence county. To make the problem easier, this file describes an area that is quite a bit smaller than the actual extent of the city or MSA.
2. `Providence_tracts_travel_time_matrix_rail.csv` for Problem 2. This is a stylized description of travel times after the transit corridor is built. It is calculated by adjusting the observed times so that any commuter can move from the statehouse to the hospitals instantaneously without any time lost to travel mode changes.
3. `inputs.csv` reports the wage w_i and the rent R_j of each of the 42 tracts. Wages come from the 2019 LEHD Origin-Destination Employment Statistics (LODES) Workplace Area Characteristics file, which reports jobs in three earnings bins by workplace tract. Rents are median gross rents from the American Community Survey (ACS) five-year estimates for 2017–2021.
4. `counts_matrix.csv` reports the number of households that work in tract i (rows) and live in tract j (columns). The counts come from the 2019 LODES origin-destination file. They count jobs by workplace tract and residence tract, and we treat each job as one household.
5. `tracts.geojson` and `water.geojson` is a map file reporting tract boundaries along with the shoreline and major rivers. This plays no part in the analysis, but makes heat maps easier to read.

Part 0: Set up your session. You installed Python and the agent in Problem Set 1. Do the steps below at the start of every session. If a step fails, look at Problem Set 1, or copy the error message into an LLM chat window and ask for help.

1. Unzip the file that came with this problem set. Open an Anaconda Prompt (Windows) or Terminal (Mac). Make a folder for this problem set:

```
cd Desktop/echw
mkdir pset3
cd pset3
```

2. Copy the six data files into the `pset3` folder. List the folder to check that all six are there (type `dir` on Windows, `ls` on Mac).

3. Activate your python environment and start the agent:

```
conda activate urbanecon
claude
```

   You should see the agent's welcome prompt. You do not need to install anything new. Your `urbanecon` environment has the packages these steps use. If the agent offers to install `geopandas`, say no and remind it to use only pandas, numpy, scipy, and matplotlib.

4. Switch the agent to the Sonnet model. Sonnet handles this work well and costs less than the larger models. At the agent's prompt, type the following and press Enter:

```
/model sonnet
```

   Check the model name in the header at the top of the agent's window. It should say Sonnet. Do this at the start of every session.

Always start the agent from inside your `pset3` folder. The agent can reach other files on your computer. Starting inside `pset3` keeps its work in one place. Watch the agent as it works. If it loops or runs for many minutes on one step, press Esc to stop it.

How the steps work. Each step below has the same four parts. First, you read a short piece of the model, and sometimes you do a derivation by hand. Second, you give the agent the prompt in the gray box. Third, you check what the agent made. Fourth, you hand in what the *Hand in* list asks for. Copy the prompt into the agent's window. Copy-paste from a PDF can add stray spaces, so read the prompt in the agent's window before you press Enter, and do not change the numbers in it.

Each step saves its results to files in `pset3`, so the next step reads them. If you close the agent and come back later, start from the step you stopped at.

We need to specify the model, and then determine which of the model's variables we know, and which we need to solve for. Once we have solved for, i.e., calibrated all of the unknown parameters of the model, we will know *all* of the model's variables.

This will allow us to change the transportation costs and solve the model for new equilibrium levels of tract employment, residence and commute times.

The model is close to the one we discussed in lecture. A closed city has N tracts and $\nu \in [0,L]$ households. Each household ν picks a workplace i and a residence j, where i and j each run from 1 to N. Rent is collected by absentee landlords.

Tract i pays wage w_i. Tract j offers amenity B_j and a fixed housing stock H_j. A trip between i and j takes t_{ij} minutes. Each minute costs a share of income, so the iceberg cost equals

$$\tau_{ij} = e^{\kappa t_{ij}}, \tag{1}$$

where κ sets the cost per minute. The after-commute wage equals w_i/τ_{ij}. A household spends the share $\beta = 0.3$ of its after-commute wage on housing at rent R_j per unit, and the rest on consumption. Its utility from the workplace-residence pair (i,j) equals $V_{ij}(\nu) = \varepsilon_{ij} u_{ij}$, with

$$u_{ij} = \frac{B_j \, w_i}{\tau_{ij} \, R_j^{\beta}}. \tag{2}$$

The taste draw ε_{ij} varies across households and workplace-residence pairs. It follows a Fréchet distribution, $F(\varepsilon) = e^{-T\varepsilon^{-\theta}}$, with $T = 1$ and dispersion $\theta = 8.0$. Each household picks its best pair. By the Discrete Choice Theorem, the share of households choosing (i,j) equals

$$s_{ij} = \frac{u_{ij}^{\theta}}{\sum_{k=1}^{N} \sum_{l=1}^{N} u_{kl}^{\theta}}. \tag{3}$$

Multiplying every B_j by the same number leaves all shares unchanged, so the scale of the amenities is arbitrary. Fix the scale by requiring

$$\sum_{k=1}^{N} \sum_{l=1}^{N} u_{kl}^{\theta} = 1. \tag{4}$$

Then the denominator in Equation (3) equals one, and in the baseline city

$$s_{ij} = u_{ij}^{\theta}. \tag{5}$$

The normalization applies to the baseline only. After a policy changes the travel times or the rents, the double sum no longer equals one.

Three sums of the baseline shares matter:

$$r_j = \sum_{i=1}^{N} s_{ij}, \tag{6}$$

$$e_i = \sum_{j=1}^{N} s_{ij}, \tag{7}$$

$$q_{ij} = \frac{s_{ij}}{r_j}. \tag{8}$$

The residence share r_j is the fraction of households that live in tract j. The employment share e_i is the fraction that work in tract i. The commute share q_{ij} is the fraction of the residents of tract j who work in tract i.

These are shares in the model. The data give observed counterparts, which we write with a superscript d. The observed share s_{ij}^d is the number of households that work in i and live in j, divided by L. Its sums are $r_j^d = \sum_{i=1}^N s_{ij}^d$ and $e_i^d = \sum_{j=1}^N s_{ij}^d$. Calibration makes the model match some of these and not others. We choose κ and B_j so that the model matches the mean commute time and the residence shares, $r_j = r_j^d$. The model does not match the cells s_{ij}^d or the employment shares e_i^d, so the model shares s_{ij} differ from the observed shares s_{ij}^d.

The commute share does not depend on the amenities or the rents. From Equation (2), $u_{ij}^\theta = B_j^\theta R_j^{-\beta\theta} (w_i/\tau_{ij})^\theta$. In Equation (8), s_{ij} and $r_j = \sum_{i=1}^N s_{ij}$ both carry the denominator of Equation (3), and every term of r_j carries the factor $B_j^\theta R_j^{-\beta\theta}$. All of these cancel in the ratio, so the commute share depends on θ, κ, and the wages only:

$$q_{ij} = \frac{(w_i/\tau_{ij})^\theta}{\sum_{k=1}^{N} (w_k/\tau_{kj})^\theta}. \tag{9}$$

Residents of tract j spend the share β of their after-commute wages on housing. Rent adjusts until that spending equals the value of the stock:

$$R_j H_j = \beta L \sum_{i=1}^{N} s_{ij} \frac{w_i}{\tau_{ij}}. \tag{10}$$

Equations (3) and (10) define the equilibrium. Equation (3) is the household side: given rents, each household picks its best pair. Equation (10) is the market side: rent is the only price that adjusts, and it moves until housing demand equals the fixed stock in every tract. Wages, amenities, and housing stocks do not adjust. Given rents, households choose. Given those choices, rents must clear the market. Together the two conditions fix the shares and the rents.

Problem 1: Calibrate the model to Providence (50 points). We want to calibrate the model. This is a big algebra problem where we use the data and the model equations to solve for the quantities we don't observe. To start, inventory what we observe. The city has $N = 42$ tracts. The data give the travel times t_{ij}, the wages w_i and rents R_j, total household count L and the observed shares s_{ij}^d.

Summing the observed shares gives r_j^d and e_i^d. We assume $\theta = 8$ and $\beta = 0.3$. We need to solve for the cost per minute κ, the amenities B_j, and the housing stocks H_j, for $j = 1, \ldots, N$.

(a) Find κ (20 points). The travel-time matrix gives minutes between population-weighted tract centroids at 8 a.m. Given the observed residence shares r_j^d, the mean commute time in the model is

$$\sum_{j=1}^{N} r_j^d \sum_{i=1}^{N} q_{ij}\, t_{ij}. \tag{11}$$

The mean commute time in the data is

$$\sum_{k=1}^{N} \sum_{l=1}^{N} s_{kl}^d\, t_{kl}. \tag{12}$$

The model's mean falls as κ rises, and the data value does not depend on κ, so the two cross once. The crossing gives κ. Give the agent this prompt.

```
I am working in the current folder. Use Python with pandas, numpy, scipy and
    matplotlib. Do not use geopandas. Save your code in kappa.py.

Files in this folder:
- Providence_tracts_travel_time_matrix_baseline.csv: travel time t[i,j] in minutes
    between 42 census tracts. The first row and first column hold the tract labels.
- inputs.csv: one row per tract, in the same order as the travel-time matrix. Columns
    : tract_label, w (wage), rent. Ignore any other columns.
- counts_matrix.csv: 42 by 42 table of household counts. Rows are the tract where
    people work (i). Columns are the tract where they live (j). The first row and
    first column hold the tract labels.
Check that the tract labels and their order agree in all three files. If any tract
    has a missing wage or rent, stop and tell me which tracts. Do not fill in numbers
    .

Task:
1. N = 42. L = sum of all entries in counts. Observed share s_d[i,j] = counts[i,j] /
    L. Observed residence share r_d[j] = sum over i of s_d[i,j].
2. Data mean commute time D = sum over all i and j of s_d[i,j] * t[i,j]. Print D and
    L.
3. For each kappa on a grid from 0 to 0.2 in steps of 0.001:
   tau[i,j] = exp(kappa * t[i,j])
   q[i,j] = (w[i]/tau[i,j])**8 / (sum over k of (w[k]/tau[k,j])**8)
   M(kappa) = sum over j of r_d[j] * (sum over i of q[i,j] * t[i,j])
4. Plot M(kappa) as a curve against kappa, and D as a horizontal line. Label the axes
    , with units (minutes). Find the kappa where M(kappa) = D by linear interpolation
     between grid points. Mark it on the graph. Save the graph as kappa_graph.png.
```

```
5. Print kappa to four decimals. Save it in kappa.txt (one number, nothing else).
6. Print 100*(1 - exp(-kappa)), the percent of income lost per minute of commuting.
If M(kappa) never crosses D on the grid, tell me and stop.
```

Open `kappa_graph.png`. The model curve should fall as κ rises and cross the flat line once. If it does not, tell the agent what you see.

Hand in.

- (10) The graph, with the crossing marked.
- (5) The value of κ to three decimals, and the percent of income lost per minute of commuting.
- (5) In two sentences, explain why the model's mean commute time falls as κ rises.

(b) Amenities and housing (12 points). *Derivation of B_j.* At baseline, Equation (5) gives $s_{ij} = u_{ij}^{\theta}$, so the model residence share is $r_j = \sum_{i=1}^{N} u_{ij}^{\theta}$. Substitute u_{ij} from Equation (2). Then $u_{ij}^{\theta} = B_j^{\theta} R_j^{-\beta\theta} (w_i/\tau_{ij})^{\theta}$. The factor $B_j^{\theta} R_j^{-\beta\theta}$ does not depend on i, so it leaves the sum. Calibration sets $r_j = r_j^d$, so

$$r_j^d = B_j^{\theta} R_j^{-\beta\theta} \sum_{i=1}^{N} (w_i/\tau_{ij})^{\theta},$$

$$B_j^{\theta} = R_j^{\beta\theta} \frac{r_j^d}{\sum_{i=1}^{N} (w_i/\tau_{ij})^{\theta}}.$$

Take the power $1/\theta$ of both sides:

$$B_j = R_j^{\beta} \left[\frac{r_j^d}{\sum_{i=1}^{N} (w_i/\tau_{ij})^{\theta}} \right]^{1/\theta}. \tag{13}$$

Derive H_j. Rent clears the housing market in tract j, Equation (10), with the model shares s_{ij}. Divide both sides by R_j. You should find

$$H_j = \frac{\beta L}{R_j} \sum_{i=1}^{N} s_{ij} \frac{w_i}{\tau_{ij}}. \tag{14}$$

Now give the agent this prompt. It uses the κ from part (a).

```
Same folder and rules as before: Python with pandas, numpy, scipy and matplotlib, no
    geopandas. Save your code in calibrate.py.

Files: Providence_tracts_travel_time_matrix_baseline.csv, inputs.csv, counts_matrix.
    csv (all as before) and kappa.txt (one number, from the last step). Set up t, w,
    R (the rent column of inputs.csv), counts, L, s_d and r_d as before. theta = 8
    and beta = 0.3. Check that the tract order agrees in all files.
```

```
Task:
1. tau[i,j] = exp(kappa * t[i,j]).
2. Amenity: B[j] = R[j]**beta * ( r_d[j] / (sum over i of (w[i]/tau[i,j])**theta) )
    **(1/theta).
3. Model share: u[i,j] = B[j] * w[i] / (tau[i,j] * R[j]**beta), and s[i,j] = u[i,j]**
    theta.
4. Housing stock: H[j] = (beta * L / R[j]) * (sum over i of s[i,j] * w[i] / tau[i,j])
    . Use the model shares s, not the observed shares s_d.
5. Save calibration.csv with columns tract_label, w, R, B, H (one row per tract).
    Save params.csv with columns kappa, L, theta, beta (one row).
```

Hand in.

- (6) Your derivation of H_j. Show the steps from Equation (10) to the result.
- (6) In two sentences, explain why Equation (14) uses the model shares s_{ij} and not the observed shares s^d_{ij}.

(c) Heat maps (12 points). The mean commute time of the residents of tract j is

$$m_j = \sum_{i=1}^{N} q_{ij}\, t_{ij}. \tag{15}$$

Give the agent this prompt.

```
Same folder and rules as before: Python with pandas, numpy, scipy and matplotlib. Do
    not use geopandas. Save your code in maps.py.

Files: calibration.csv, params.csv, Providence_tracts_travel_time_matrix_baseline.csv
    , counts_matrix.csv, tracts.geojson, water.geojson.
- tracts.geojson has one polygon (or multipolygon) per tract, with a property
    tract_label that matches the tract labels in the travel-time matrix. Match labels
     as text.
- water.geojson holds the shoreline and major rivers. Each feature has a property
    kind: "area" for a water polygon, "line" for a river line.
Read both geojson files with the json module and draw them with matplotlib. Fill
    tract polygons by value. Draw area water as a filled light blue and line water as
     blue lines on top of the tracts.

Task:
1. Read calibration.csv and params.csv. tau[i,j] = exp(kappa * t[i,j]). u[i,j] = B[j]
     * w[i] / (tau[i,j] * R[j]**beta). Model share s[i,j] = u[i,j]**theta.
2. Model employment share e[i] = sum over j of s[i,j]. Model residence share r[j] =
    sum over i of s[i,j]. Model mean commute time of the residents of j: m[j] = sum
    over i of (s[i,j] / r[j]) * t[i,j]. Observed residence share r_d[j] = (column sum
     of counts_matrix.csv for j) / L.
3. Write a function draw_map(values, title, colorbar_label, vmin, vmax, filename) in
    maps.py. It shades each tract by its value, adds a color bar with the label,
    draws the water on top, and uses the same map extent (the bounds of tracts.
    geojson) every time. I will reuse this function later.
```

```
4. Draw and save four maps: map_model_employment.png (e), map_model_residence.png (r)
    , map_model_commute.png (m, in minutes), map_observed_residence.png (r_d). The
    two residence maps must use the same color scale.
5. Save e, r, m, and r_d by tract in baseline_values.csv, with a tract_label column.
```

Hand in.

- (8) The four maps.
- (2) In two or three sentences, say where jobs and residents concentrate, and where commutes run long.
- (2) Does the model residence map r_j match the observed map r_j^d? Why or why not?

(d) Check the code (6 points). The agent did what you asked, but did it do what you meant? Find the file `calibrate.py` in your `pset3` folder. Copy it into the chat window of a different LLM and ask: *"Read this program. Tell me, step by step, what it computes."*

Hand in.

- (3) The other LLM's description of `calibrate.py`.
- (3) Does the description match the steps in your prompt in part (b)? Name any difference.

Problem 2: The light rail line (50 points). You evaluate a rail line that links the State House to the hospitals, using the model you calibrated in Problem 1. The line joins tract 800, which holds the State House, to tract 600, which holds Rhode Island Hospital. Travel between the two stations takes no time, and switching between car and train costs nothing. Both stations sit at their tract centroids. The file `Providence_tracts_travel_time_matrix_rail.csv` gives the new travel times t'_{ij}, and these give new iceberg costs $\tau'_{ij} = e^{\kappa t'_{ij}}$.

For the counterfactual case, we know t'_{ij}. We also know amenities B_j, the housing stocks H_j (we just solved for them). Wages w_i, the household count L, the cost per minute κ, and the assumed values $\theta = 8$ and $\beta = 0.3$ are assumed not to change in the counterfactual. If we want to solve for the new rents R'_j, for $j = 1, \ldots, N$, then the new shares s'_{ij} are easy to calculate, and everything else follows.

(a) New rents (12 points). *Derivation.* After the policy the double sum in Equation (4) no longer equals one, so the denominator of Equation (3) returns. Take any vector of rents $R = (R_1, \ldots, R_N)$. The shares after the policy are $(u'_{ij})^\theta$ divided by the double

sum. Recalling that $u'_{ij} = B_j w_i / (\tau'_{ij} R_j^{\beta})$, we can write the shares s'_{ij} as

$$s'_{ij}(R) = \frac{B_j^{\theta} \left(w_i / \tau'_{ij}\right)^{\theta} R_j^{-\beta\theta}}{\sum_{k=1}^{N} \sum_{l=1}^{N} B_l^{\theta} \left(w_k / \tau'_{kl}\right)^{\theta} R_l^{-\beta\theta}}. \tag{16}$$

The new rents clear the housing market in every tract, as in Equation (10):

$$R'_j H_j = \beta L \sum_{i=1}^{N} s'_{ij}(R') \frac{w_i}{\tau'_{ij}} \qquad \text{for } j = 1, \ldots, N. \tag{17}$$

Substitute Equation (16) into Equation (17) and divide by H_j. This gives

$$R'_j = \frac{\beta L}{H_j} \sum_{i=1}^{N} \frac{B_j^{\theta} \left(w_i / \tau'_{ij}\right)^{\theta} R_j^{-\beta\theta}}{\sum_{k=1}^{N} \sum_{l=1}^{N} B_l^{\theta} \left(w_k / \tau'_{kl}\right)^{\theta} R_l^{-\beta\theta}} \frac{w_i}{\tau'_{ij}}. \tag{18}$$

for all $j = 1, \ldots, N$.

Equations (16) and (17) restate the equilibrium conditions we gave earlier. Equation (16) gives the choices of households at any rents R. Equation (17) says that the new rents clear the housing market for given household choices. Substituting equation (16) into equation (17) forces both conditions to hold, so a rent vector that solves (18) is an equilibrium: every household picks its best pair, and every housing market clears.

Solving (18) is tricky. Rents appear on both sides. The left side holds R'_j. The right side depends on the whole vector R': through $R_j^{-\beta\theta}$ in the numerator, and through $R_l^{-\beta\theta}$ for every tract l in the denominator. This type of problem is called a fixed point problem and it has no closed-form solution. You must use a numerical solver to find R'.

Guidance for the solver.

- Solve for the log rents $x_j = \log R'_j$. This keeps every rent positive.
- Let $G_j(R)$ be the right side of Equation (18) at rents R. Find the x that sets the 42 residuals $\log G_j(e^x) - x_j$ to zero.
- Start from the baseline rents, $x_j = \log R_j$.
- Use a root finder, for example `scipy.optimize.root`, and stop when the largest residual falls below 10^{-6}. Do not loop on $R \leftarrow G(R)$. That loop can diverge.

The prompt below passes this guidance to the agent. It also asks for a test. If you run the solver with the baseline travel times in place of the rail times, nothing has changed, so the solver must return the baseline rents.

```
Same folder and rules as before: Python with pandas, numpy, scipy and matplotlib, no
    geopandas. Save your code in counterfactual.py.

Files: calibration.csv (columns tract_label, w, R, B, H), params.csv (kappa, L, theta
    , beta), Providence_tracts_travel_time_matrix_baseline.csv, and
    Providence_tracts_travel_time_matrix_rail.csv (new travel times, same layout as
    the baseline matrix). Check that the tract order agrees in all files.

For a travel-time matrix T and a vector of rents Rv, define:
  tau[i,j] = exp(kappa * T[i,j])
  num[i,j] = B[j]**theta * (w[i]/tau[i,j])**theta * Rv[j]**(-beta*theta)
  share[i,j] = num[i,j] / (sum of num over all i and j)
  G[j] = (beta * L / H[j]) * (sum over i of share[i,j] * w[i] / tau[i,j])
We want rents Rnew such that Rnew[j] = G[j] when Rv = Rnew, for all 42 tracts.

How to solve: let x[j] = log(Rnew[j]). Find x such that log(G[j](exp(x))) - x[j] = 0
    for all j. Use scipy.optimize.root. Do NOT iterate Rv <- G(Rv); that loop
    diverges. After solving, compute the largest absolute residual. It must be below
    1e-6. If it is not, try another method (for example method="lm") and tell me.

Task:
1. Test. Use T = the baseline travel times. Start the solver from the baseline rents
    R multiplied by a random factor between 0.8 and 1.2 for each tract (fixed random
    seed). The solution must equal R. Print the largest value of |Rnew - R| / R. It
    should be below 1e-6. If it is not, stop and tell me.
2. Now use T = the rail travel times. Start the solver from the baseline rents R.
    Solve. Print whether the solver succeeded and the largest residual.
3. Compute the counterfactual shares share[i,j] at Rnew. Save them as
    counterfactual_shares.csv (42 by 42, first row and column hold tract labels).
    Save counterfactual.csv with columns tract_label, R_new, pct_change_rent = 100 *
    (R_new / R - 1).
4. Print the five tracts with the largest rent increase and the five with the largest
     rent decrease, in percent, with labels.
5. Aggregate land rent. Print the sum over j of R[j]*H[j] at baseline and the sum
    over j of Rnew[j]*H[j] after the line, and the percent change.
```

Hand in.

- (4) Your derivation: show the steps from Equations (16) and (17) to Equation (18).
- (4) The result of the test in step 1 of the prompt. In one sentence, say why the baseline rents must come back.
- (4) Whether the solver succeeded, and the largest residual. The five largest rent increases and the five largest decreases.

(b) Heat maps (8 points). Use s'_{ij} from part (a) to compute the residence share $r'_j = \sum_{i=1}^{N} s'_{ij}$, the employment share $e'_i = \sum_{j=1}^{N} s'_{ij}$, and the mean commute time $m'_j = \sum_{i=1}^{N} (s'_{ij}/r'_j)\, t'_{ij}$ of the residents of tract j. Give the agent this prompt.

```
Same folder and rules as before: Python with pandas, numpy, scipy and matplotlib, no
    geopandas. Save your code in counterfactual_maps.py. Reuse the function draw_map
    in maps.py. You may change it so that it can draw into a given matplotlib axis.

Files: counterfactual_shares.csv, baseline_values.csv (columns tract_label, e, r, m,
    r_d), Providence_tracts_travel_time_matrix_rail.csv, params.csv, tracts.geojson,
    water.geojson.

Task:
1. s_new[i,j] comes from counterfactual_shares.csv. r_new[j] = sum over i of s_new[i,
    j]. e_new[i] = sum over j of s_new[i,j]. m_new[j] = sum over i of (s_new[i,j] /
    r_new[j]) * t_new[i,j], where t_new is the rail travel-time matrix.
2. Make three figures. Each figure has two panels side by side. The left panel is the
     model baseline map (r, e, or m from baseline_values.csv). The right panel is the
     counterfactual map (r_new, e_new, or m_new). Do not use the observed values r_d.
     In each figure, use the same map extent and the same color scale for both panels
    . Add a color bar with units and draw the shoreline and rivers. Save them as
    pair_residence.png, pair_employment.png, pair_commute.png.
3. Print the five tracts with the largest increase and the five with the largest
    decrease in the number of residents, L * (r_new - r), in households, with labels.
```

Hand in.

- (6) The three figures.
- (2) In two or three sentences, say what the line does to the geography of residence, jobs, and commuting.

The next three questions need no new prompt. Use your results.

(c) Observed H_j (10 points). It is easy to observe a measure of the amount of housing in a tract H_j. What would happen if we also used observed values of H_j to calibrate the model?

(d) Patterns (10 points). Describe the patterns you see in your heat maps in one paragraph. What areas benefit from new transit corridor, and which areas are harmed? Does this seem sensible?

(e) Land rent (10 points). What happens to aggregate land rent? Is this consistent with your intuition from the monocentric city model? Use the aggregate land rent that the agent printed in step 5 of the prompt in part (a).