



Programming with *AI at Your Side*

AI at Your Side · Lecture 4
Sebastian Galiani · Raul A. Sosa

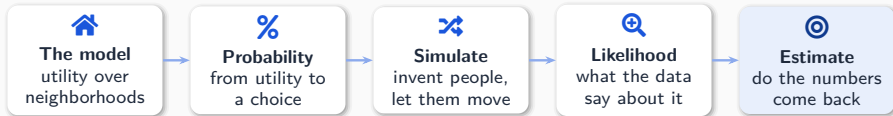
Kyiv School of Economics



Book website

Today: One Model, From English to Estimates

About ten minutes on how to work, and then we build one thing for the rest of the lecture



🎯 We are going to invent a small world, write down what people want, and then throw away the answer and try to get it back from the choices alone. If the numbers we made up come back out, we have good evidence that the code is right.

💡 Today we follow **one real session with a coding agent**, from a folder with one data file, neighborhoods.csv (slide 9). Every line in a dark terminal window comes from it, cut to fit and stamped with the model and the date.

One Rule, Before Any Code

You no longer need to know how to write every line of code.
You still need to know what every important line is doing.

✓ What changed

Syntax, boilerplate, which library does what, and the afternoon you used to lose to a misplaced bracket. The model handles all of it now.

✗ What did not

Whether the thing you built is the thing you meant. Only you can check that.

🧠 **The uncomfortable part:** code that is wrong in an interesting way usually runs. It returns a number with the right number of digits, and nothing says stop.

Words You Will Need Today

You do not need to know them already. Each one is a question for the tutor of Lecture 3

Coding agent. An AI that works in a folder on your computer: it reads the files, writes code, runs it and reads what comes back. Here, Claude Code.

Python, numpy, scipy. The language of the code, numpy for arrays and math, and scipy, which has ready-made optimizers.

Seed. The number that fixes the random draws, so running it again gives the same data.

Overflow, underflow, inf, nan. e^{800} is too big to store and comes back as inf. e^{-800} rounds to 0, and $\log 0$ is $-\text{inf}$. nan, “not a number,” is what arithmetic returns once it breaks.

1e-12. One times ten to the minus twelve.

Turn. One request and the agent’s answer to it. Today’s session has eight.

Finite differences. A numerical check of a derivative: move one input a little and see how the output moves.

Log-likelihood, NLL. How probable the observed choices are under given coefficients, in logs. The NLL, the negative log-likelihood, is the same number with the sign flipped, so minimizing it maximizes the likelihood.

Optimizer, start. The routine that searches for the coefficients with the smallest NLL, beginning from a starting guess.

Gradient, Hessian. The slope and the curvature of the NLL. At the answer the gradient is about zero, and the square roots of the diagonal of the Hessian’s inverse are the standard errors.

Eigenvalues. Of the Hessian: how curved the NLL is along each of its main directions. One near zero means the NLL is flat that way (slide 28).

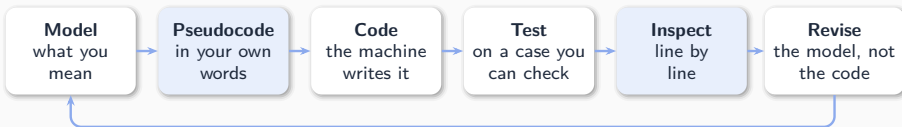
Newton-Raphson, step-halving. An optimizer that picks each step from the gradient and the Hessian. Step-halving halves a step while it makes the fit worse.

Block 1 · How to Work

A short block, and then we build something

The Loop

Six steps, and the two that people skip are the second and the fifth



 **Pseudocode is the step that pays.** Writing what you want in plain sentences, before anyone writes code, is where you find out that you did not actually know. If you cannot say it in four sentences, the code you get back will solve something else, and it will run.

 And inspect is not reading. Reading is what you do when the code already looks right. Inspecting is picking a line and asking it to justify itself. Slide 14 is about that.

Three Ways Code Goes Wrong Without Telling You

Announced now, and all three come back later in this lecture with the numbers

1 · Silent errors

The calculation breaks and nothing raises an error. No exception, no stop, a gray warning line nobody reads, and a result comes back anyway. You find out because the answer is strange, or you never find out.

2 · Wrong object

You reach for something with the right name and the right shape, and it is not the thing you wanted. The units are plausible, the table prints, and the number means something else.

3 · Right code, unanswerable question

Every line does exactly what it says. The model asks the data a question they cannot answer, the table prints anyway, and no amount of debugging finds that.

🕒 We will meet all three today, in that order. The agent steps around or catches each one on its own, and we will see what each would have cost by taking away what saved us.

Block 2 · One Model, End to End

Where people live, and what it takes to find out why

Six Neighborhoods

Invented, so we know the answer in advance



Rent per month, minutes to downtown, school index from 1 to 10

Neighborhood	Rent	Commute	Schools
Old Town	\$2,600	12	5.5
The Flats	\$1,750	28	4.0
Hillside	\$3,900	22	8.5
University	\$3,100	8	6.0
Riverside	\$2,200	35	7.0
Eastgate	\$1,900	45	6.5

 **Not one dominates another**, which is what makes this a choice and not a ranking. And they are invented on purpose: with real neighborhoods you would spend the hour arguing about the rent. Here we **make up the answer ourselves**, hide it, and see whether we get it back. You cannot do that with real data, because nobody knows the answer.

Where a Model Like This Comes From

If you do not know the literature, start as in Lectures 1 and 2: five fields, and ask for the map



Claude Sonnet 5 · clean session · Oct 7, 2026

condensed

Context: I'm a second-year economics student. I know intermediate micro and OLS, and I know nothing about how economists study where people choose to live.

Task: tell me how the literature models the choice of where to live, and then take me from that literature to the simplest model I could simulate and then estimate myself.

Constraints: one city, a handful of neighborhoods, and only things about them I could actually measure.

Output: [...] **Verification:** write UNSURE next to any paper, year or journal you are not sure of. [...]



Claude

[...] 1. **Monocentric city / bid-rent models** [...] 2. **Discrete choice / random utility models** [...] McFadden (1978)
[...] UNSURE on exact publication details [...] 3. **Equilibrium sorting models** [...] Bayer, Ferreira & McMillan (2007)
[...] 4. **Hedonic / spatial equilibrium** [...] The right starting point is **McFadden's discrete choice model, stripped down to a conditional logit** [...]

$U_{ij} = \beta_1 \text{Rent}_j + \beta_2 \text{Distance}_{ij} + \beta_3 \text{Amenity}_j + \varepsilon_{ij}$ [...] 6. Check that $\hat{\beta}$ is close to the true β you picked [...]



It lands on the logit of the next slide, and on this lecture's test: check that the betas come back. All eight works it names are real. Ours: rent as a share of income, and schools that matter more with children.

Say It in English First

The sentence comes before the equation, always, and the equation is just the sentence

“A household likes a neighborhood **less** when the rent eats a bigger share of what it earns, **less** when the commute is long, **more** when the schools are good, and **much more** when the schools are good *and* there are children in the house.”

$$V_{ij} = \beta_1 \frac{\text{rent}_j}{\text{income}_i} + \beta_2 \text{commute}_j + \beta_3 \text{schools}_j + \beta_4 (\text{schools}_j \times \text{kids}_i)$$

V_{ij} , the utility we can measure, scores how much household i likes neighborhood j . Four numbers, and every one of them is a sentence you just heard.

☉ The signs are a prediction. We expect the first two negative and the last two positive. If the estimates come back with the wrong sign, something is wrong *before* we start interpreting magnitudes.

Three Kinds of Variable, and They Behave Differently

Where a variable varies decides what it can teach you

Only j

Commute and schools. The same number for everyone, because it is a fact about the place.

Both i and j

The rent burden. The rent is the neighborhood's, the income is the household's, and the ratio belongs to the pair.

An interaction

Schools times children. It exists so that two households can look at the same school and want it differently.

 **Two points here.** Only the last two kinds can explain why two *households* made different choices. And nothing at all can be learned from a variable that is identical across the six *neighborhoods*. Hold on to the second one. It is the reason one of today's three failures happens.

From What People Want to What People Do

One assumption about what we cannot see, and a formula falls out

V_{ij} is what we can write down. But people also care about things we did not measure, like a friend two blocks away. Call that ε_{ij} . A household picks the neighborhood with the highest $V + \varepsilon$.

✓ Assume those unmeasured bits are independent draws from one and the same **Type I extreme value** (Gumbel) distribution, and the probability of picking j comes out in closed form, **the logit**:

$$P_{ij} = \frac{e^{V_{ij}}}{\sum_k e^{V_{ik}}}$$

Why not normal draws? They give the probit, just as sensible, but with no formula: each probability is an integral the computer has to approximate. **The formula has a price, known as IIA:** the odds between any two neighborhoods ignore all the others.

⚠ **A fact and a convention.** Only *differences* in V matter: add the same number to every alternative and nothing changes, so no level can be estimated. And the scale of ε is fixed at one by convention, so only ratios of coefficients mean something. The first is Problem Set 3 (slide 32).

Four Questions, Every Time Code Comes Back

Ideally every line. Fewer if that takes too long, but never zero

What is this line doing?

In one sentence, without using the words in the line itself.

Why do we need it?

Which part of the model asked for it. If nothing did, ask harder.

What if I deleted it?

The best of the four. If the answer is nothing, you have found something.

How could I test it?

A case where you already know the answer. There is always one.

🔍 The more lines you ask about, the more of the code is yours. Ask the agent in the same session. It answers all four well. Explaining a line is a separate job from writing it, so the explanation can catch what the writing missed. We use them three times today: on the unmeasured part, the line nobody asked for, and the standard errors.

The First Request

The five fields from Lecture 1, with the pseudocode inside the Task

Claude Code · turn 1 of 8 condensed · reply on slides 16 and 17

> **Context:** I'm an economist. I know the logit model on paper and can read numpy, but I check every line. [...] I expect households with children to choose Hillside more often.

Task: Simulate where 2,000 households choose to live. My pseudocode:

- Build 2,000 households with an income and a yes-or-no on children.
 - For each household and each neighborhood, work out V .
 - Add the unmeasured part.
 - Whichever total is highest, that is where they live.
- $$V = -5 \text{ rent/income} - 0.045 \text{ commute} + 0.20 \text{ schools} + 0.40 \text{ schools} \times \text{kids}$$

Constraints: numpy. [...] Do NOT use a probability formula: draw the unmeasured part and take the largest, the way I wrote it.

Output: households.csv with income, kids and chosen, and the share choosing each neighborhood, overall and split by kids.

Verification: flag every number you had to choose yourself. I will check the Hillside split.

 **Context** states your own frontier (Lecture 3): your level and your hunch, so you can audit the code.

 **Task** is four sentences of pseudocode, and its last line is the answer we are going to hide.

 **No probability formula**, so the simulation shares as little code as possible with the estimator.

 **Verification** asks for every number it invents. It pays off two slides later.

Simulating the Choices

Three lines that came back, and two of the four questions asked of them

Claude Code · Sonnet 5 · Sep 23, 2026 · turn 1 of 8

condensed · simulate.py

```
40 eps = rng.gumbel(loc=0.0, scale=GUMBEL_SCALE, size=(N, J))
41 U = V + eps
42 chosen_idx = U.argmax(axis=1) # (N,)
```

Claude Code · Sonnet 5 · Sep 23, 2026 · turn 2 of 8

condensed

- > Two lines I want to understand before I trust this. The epsilon line: why Gumbel and not a normal? And the argmax line: what does axis=1 do, and what would break if it said axis=0?
- [...] If each household-neighborhood pair gets an independent Gumbel (type I extreme value) draw, and households pick the highest $V + \text{eps}$, then the probability of picking neighborhood j is exactly $\exp(V_j) / \sum_k \exp(V_k)$. That is McFadden's result, [...] With normal draws you'd get a valid choice model, but it's probit, not logit. [...]
- [...] For each household it returns the column index of that household's largest total, so the result has shape $(2000,)$ with values 0 to 5. [...]
- [...] The script would most likely crash rather than give quiet wrong numbers. [...] The quiet version of this bug would appear if the matrix were square, say 6 households and 6 neighborhoods. [...]

Look at It Before You Estimate Anything

Two minutes here saves the next twenty

Claude Code · Sonnet 5 · Sep 23, 2026 · turn 1 of 8 condensed

neighborhood	overall	kids=1	kids=0	diff
Old Town	22.1%	16.4%	25.3%	-8.9%
The Flats	13.6%	7.2%	17.0%	-9.8%
Hillside	15.4%	23.6%	11.0%	+12.6%
University	19.6%	19.5%	19.6%	-0.1%
Riverside	17.2%	20.5%	15.5%	+5.0%
Eastgate	12.0%	12.9%	11.6%	+1.2%

● [...] Numbers I had to choose

1. **Income units and distribution [C]**. This is the one that matters most. [...] I assumed monthly rent and monthly income. [...] If your rents are meant to be monthly and incomes annual, the rent term nearly vanishes, and Hillside's price stops mattering.
2. **Share with kids [B]**. I set $P(\text{kids}) = 0.35$ [...]
4. **Seed [A]**. The seed is 20260923, which is arbitrary. [...]

✓ The check to run first.

Hillside has the best schools and the highest rent, so households with children should pick it more. They do: **23.6 against 11.0**. Had that come out at zero, we would have a bug. We built that gap in, so the check tests the code.

⚠ And what it chose for us.

Verification asked, so it listed its own choices, the most important first: from the income units, which matter most, to the seed, which is arbitrary. The letters label them in its code.

If You Have Never Seen a Likelihood

The next slide writes one. This is the question to take to a new chat first



Claude Sonnet 5 · clean session · Oct 7, 2026

condensed

Explain likelihood to me intuitively, assuming I know no econometrics. Why do we maximize it? And why does the computer minimize the negative log-likelihood instead? Use the example of a model where households choose among six neighborhoods, and keep it short.



Claude

[...] **Likelihood** = if your model's parameters were true, how probable is it that we'd see the exact choices the data actually shows? For each household, plug in the probability your model assigned to the neighborhood *they actually chose*. Multiply these probabilities across all 1,000 households. That product is the likelihood.

Why maximize it? [...] **which parameter values make the data we observed look most probable?** [...]

Why minimize the negative log-likelihood instead? [...] 1. **Underflow**: multiplying 1,000 small probabilities [...] gives an absurdly tiny number computers can't represent accurately. Taking logs turns multiplication into addition [...]

2. **Optimizers minimize by convention** [...] The parameters that maximize likelihood are identical to those that minimize negative log-likelihood.



All of it is right, including the answer to the next slide's minus sign. Any word on slide 4 can go to a new chat the same way.

The Likelihood, and the Part That Is Not Economics

Nobody asked for the row maximum, and it is the thing to interrogate

Claude Code · Sonnet 5 · Sep 23, 2026 · turn 3 of 8

condensed · loglik.py

> Good. Now write the log-likelihood for this model, as a function of the four betas, using households.csv. Return the NEGATIVE, so I can minimize it. Then explain, in words, what the optimizer is maximizing.

```
34 def neg_loglik(beta):
35     [...]
36     V = X @ beta                                # (N, J)
37     # log P(chosen) = V_chosen - log sum_j exp(V_j); subtract the row max first so exp cannot overflow
38     m = V.max(axis=1, keepdims=True)            # (N, 1)
39     logsumexp = m[:, 0] + np.log(np.exp(V - m).sum(axis=1)) # (N,)
40     V_chosen = V[np.arange(N), chosen]          # (N,)
41     return -(V_chosen - logsumexp).sum()
```

● [...] I subtract each row's maximum before exp so nothing overflows. The result is identical. [...]

✓ **The row maximum, in lines 38 and 39, has no economics in it.** Line 37 says why: e^V for a large V is too big to store. It also keeps the sum from rounding to zero when all V are very negative (slides 24, 25).

Tested Where the Answer Is Known

The fourth question of slide 14, and the agent asked it of itself

Claude Code · Sonnet 5 · Sep 23, 2026 · turn 3 of 8

condensed

● [...] Checks I ran

- **Loop version:** A plain loop written straight from the formula gives 3413.5637, the same as the vectorised function at the true betas.
- **Zero betas:** At $\beta = 0$ every neighborhood has probability $1/6$, so the result must equal $N \cdot \ln 6$. It does: 3583.5189.
- **Perturbations:** Moving any one beta 50% away from the truth, in either direction, raises the negative log-likelihood (by 11 to 98). So the true betas sit in a minimum along each axis.

● [...] What the optimizer is maximizing

[...] The optimizer searches for the betas that make the observed choices as probable as possible. [...]

💬 **And the sentence most people skip.** Asked what the optimizer is maximizing, it answered with the last line above. That is maximum likelihood, and it fits in a sentence.

Do the Numbers Come Back?

Same session, one more request

Claude Code · Sonnet 5 · Sep 23, 2026 · turn 4 of 8

condensed

- > Now the estimation. Load households.csv and neighborhoods.csv, minimize the negative log-likelihood, and report each estimate with a standard error and a 95% interval, next to the true values. Then tell me which lines I should check.
- The estimates are all close to the true values, and all four true values fall inside the 95% intervals. [...]
- | | True | Estimate | s.e. | 95% interval | Covers truth? |
|----------------|--------|----------|--------|--------------------|---------------|
| rent/income | -5.000 | -5.088 | 0.412 | [-5.895, -4.281] | yes |
| commute | -0.045 | -0.0483 | 0.0033 | [-0.0549, -0.0418] | yes |
| schools | 0.200 | 0.230 | 0.034 | [0.163, 0.298] | yes |
| schools × kids | 0.400 | 0.387 | 0.037 | [0.313, 0.460] | yes |
- [...] **Caveat on coverage:** Four out of four intervals covering the truth is one sample of 2,000. It is reassuring but is not proof that the standard errors are calibrated. To test that you'd repeat the simulation and estimation many times and count how often each interval covers. [...]

- ✓ Four for four, and none equals the truth, nor should it: four exact matches would point to a bug.

Four for Four Proves Less Than It Looks

So we did what it said: the same model, three hundred times



Our Monte Carlo: 300 samples of 2,000 households, each one estimated

	True	Mean estimate	SD of estimates	Mean SE	Covered
rent burden	-5.000	-4.997	0.408	0.421	96.0%
commute	-0.045	-0.0449	0.0034	0.0035	96.0%
schools	0.200	0.1995	0.033	0.035	97.3%
schools $\times$ kids	0.400	0.3996	0.036	0.038	95.3%



The estimator passes

The estimates center on the truth, their spread matches the standard errors, and each interval covers about 95% of the time.



And four for four fails

In **32 of the 300** samples, at least one interval missed. With code that is right, about one lecture in nine would not have shown you four for four.

🔍 Run by us with the lecture's model on our own simulated data, not by the agent.

No scipy, So It Wrote Its Own

The sixth line on its list can fail silently: swap two columns, and two estimates trade labels

Claude Code · Sonnet 5 · Sep 23, 2026 · turn 4 of 8

condensed

- [...] scipy isn't installed, and you asked for numpy, so I wrote the optimizer myself. It is Newton-Raphson with step-halving, using the analytic gradient and Hessian [...]
- **Checks I ran**
 - **Gradient at the optimum:** The gradient is about $1e-12$, so it is a genuine stationary point.
 - **Finite differences:** At an arbitrary test point, my analytic gradient and Hessian match finite differences of `neg_loglik` (differences of $2e-7$ and $2e-5$).
- [...] Lines you should check
 1. **grad_hess, the grad line:** [...] Check the sign and that `xbar` uses `P`.
 2. **grad_hess, the second and hess lines:** [...] The finite-difference check tests this, but only against my own `neg_loglik`.
 3. **probs:** the `V.max` subtraction and the normalisation. [...]
 4. **Newton update:** [...] The sign has to be a minus [...]
 5. **se = sqrt(diag(inv(H))):** [...]
 6. **loglik.py, the X stack:** The order of the regressors must match `beta`. If you swap two columns, the code still runs and every estimate lands under the wrong label. [...]

Gradient: which way the NLL goes up. **Hessian:** how curved it is there. **Newton-Raphson** uses both to step down.

The Error That Stops Nothing

The third question, on the line nobody asked for

Claude Code · Sonnet 5 · Sep 23, 2026 · turn 5 of 8

condensed

- > In the likelihood, the line that subtracts the row maximum: I never asked for it. What would happen if I deleted it? [...] show me what comes back.
- Deleting the line changes nothing in your estimation. It does matter in the failure cases I tested, and there it can return a wrong answer with no error.
- [...] The reason is that at the estimate V only ranges from -15.7 to 3.5 . $\exp$ overflows above about 709 and underflows below about -745 , so nowhere near either limit. [...]

beta	with max	without max
[...]		
commute coef -100	2,973,733.67	-inf
- [...] V becomes very negative for every neighborhood, every $\exp$ underflows to 0 , and $\log(0) = -\text{inf}$. The function then reports a negative log-likelihood of $-\text{inf}$, the best possible score [...]. A minimizer would move towards those betas, and nothing would raise an error.
- [...] In this run you could delete it safely, but I'd keep it, because the failure would be silent. [...]

⚠ So what, in real data, pushes a search that far out?

The Silent Error, Where It Bites

Income in thousands, as in many datasets: without the line, a bad starting guess scores best

Claude Code · Sonnet 5 · Sep 23, 2026 · turn 6 of 8

condensed

- > Suppose income had come in thousands of dollars, as it does in many datasets. Run the estimation both ways on that, with the line and without it, and show me what comes back, including what the optimizer reports.
- With a start at zero, income in thousands changes nothing about the fit [...]
- Start at the dollar-income betas (-5, -0.045, 0.20, 0.40): both fail [...] Here they put V near -2,000, and the two versions break differently:

	with max	without max
NLL at the start	1,370,821.69 (correct)	-inf (wrong)
Iteration 0	NLL 3.0e10, after 27 step-halvings	NLL nan
What my loop reports	CRASHED: LinAlgError: Singular matrix [...]	STOPPED: NLL is nan [...]
- [...] Without the max line: every exp underflows to 0 and the function reports -inf, the best possible score, at a point where the true value is 1.4 million. [...] This is the silent failure from the last test, now reached by a plausible input. [...]
- [...] A flaw in my earlier code The step-halving loop in estimate.py is [...] It exits when it reaches the 1e-8 floor without checking that the step actually decreased the NLL. It then accepts the step. [...] I should have listed it among the lines to check last time. [...]

The Object That Is Not the Object

Right name, right shape, and it depends on the road the search took

Claude Code · Sonnet 5 · Sep 23, 2026 · turn 7 of 8

condensed

```
> [...] Where did your standard errors come from? Most people would have called scipy's
  minimize here, which also hands back res.hess_inv. What is it [...]? Install scipy if you
  need it.
● [...] With the default method, BFGS, hess_inv is not the inverse Hessian at the optimum. It
  is BFGS's running approximation to it, starting from the identity matrix. [...]
● Estimates are the same; standard errors are not [...]
```

Run	rent/income	commute	schools	schools × kids	success
My exact Hessian	0.4116	0.0033	0.0344	0.0374	n/a
minimize(f, zeros) (default)	1.0187	0.0102	0.1260	0.0081	False
Different start (-1, -0.01, 0.1, 0.1)	0.0090	0.0018	0.0200	0.0301	False
With analytic gradient	0.4098	0.0033	0.0342	0.0345	True
Analytic gradient, gtol=1e-9	0.4066	0.0033	0.0342	0.0361	False
[...]					

⚠ **hess_inv is a fact about the search, not about your data.** The default call makes schools look insignificant, $[-0.017, 0.477]$ ($0.230 \pm 1.96 \times 0.126$), and three of four scipy runs said success=False with good estimates.

A Commute That Never Changes

3 · Right code, unanswerable question

We built a second city's files too: the same columns, and every neighborhood 25 minutes away

Claude Code · Sonnet 5 · Sep 23, 2026 · turn 8 of 8

condensed

> A colleague ran the same survey in another city and sent me the two files, same columns.
[...] Run the same estimation on them and give me the same table.

	true	estimate	s.e.	95% interval	covers?
commute	-0.045	0.0885	13107.2000	[-25690.0235, 25690.2005]	yes

- The output above has one problem. The commute row is meaningless [...]
- My "covers? yes" was also vacuous, because an interval that wide covers anything. [...]
- **Is 25 real?** All six neighborhoods having the identical commute suggests a placeholder [...]

🔍 This is slide 12's second point: a variable that is identical across the six neighborhoods teaches nothing.

Why No Code Can Fix It

3 · Right code, unanswerable question

The likelihood is flat in one direction, and that is a fact about the data

✖ **Commute, scipy's minimize, run by us**

start	estimate	log-likelihood
0	0.0000	-3276.8048758
5	5.0000	-3276.8048758
100	100.0000	-3276.8048758

The estimate is wherever it started, and hess_inv reports 1.0000, the identity it began with.

🔍 **The code was right, and so was the model.** These data cannot answer the question. The Hessian is the curvature of the NLL (slide 4). Its smallest eigenvalue over its largest, with the regressors on comparable scales, says how flat the flattest direction is: about 3 in 100 in the first city, and here about 10^{-12} or less, even negative, which is zero plus rounding error. Along that direction the likelihood does not move, so the data cannot pick a value.

🔍 **Two numbers to ask for, always:** the gradient at the answer, which says the search stopped at an optimum, and this ratio, which says that optimum is a single point and not a flat ridge.

How Much Data Is Enough?

Run by us: the same model on three fresh samples, with twenty times the data from top to bottom

 Estimates, with the standard error in parentheses

Households	burden	commute	schools	sch × kids
<i>truth</i>	−5.000	−0.045	0.200	0.400
100	−4.18 (2.13)	−0.044 (0.016)	0.039 (0.167)	0.471 (0.163)
500	−5.85 (0.90)	−0.049 (0.007)	0.308 (0.075)	0.445 (0.078)
2,000	−5.08 (0.43)	−0.050 (0.004)	0.237 (0.036)	0.389 (0.038)

 With 100 households the school coefficient comes back at **0.039** against a truth of **0.20**. Its interval, −0.29 to 0.37, covers the truth and zero, so it cannot say whether schools matter.

 Twenty times the data should shrink each error by $\sqrt{20} = 4.47$. Error at 100 over error at 2,000, unrounded: **4.98, 4.47, 4.60, 4.35**.

Change the Model, Not the Code

This is where AI coding stops being coding and starts being economics



The option to stay

Nobody has to move. Add staying as a seventh option and the six are measured against it. Raise the rent in all six by the same amount and, for the first time, something happens: people stay.



A cost of moving

Make staying mean staying where you live now, and charge a fixed amount to leave. Now where you already live changes what you choose, and the model needs a past.



Time worth more

Rent is divided by income and the commute is not, so a minute is already worth more dollars to a richer household, in exact proportion. Let the data say how fast it rises.

🎯 None of the three is a coding problem. Every one is a sentence about people, and the code follows.

Becker Has Been Here Since Slide 11

What a household would pay in rent to save one minute of commute

Becker's point is that an hour is worth more to someone who earns more. Rent enters as a share of income and the commute does not. So the rent a household would give up for one minute less of commute is

$$\frac{\beta_{\text{commute}}}{\beta_{\text{burden}}} \times \text{income} = \frac{0.045}{5} \times \text{income} = 0.009 \times \text{income per month} \quad (\beta_2 \text{ and } \beta_1 \text{ on slide 11})$$

🕒 At the numbers we hid

About **\$59 a month** at the median income of the session's households (\$6,503 a month), and **\$117** at twice the median.

✗ The obvious move is the wrong one

Divide the commute by income too, and it sounds like Becker. It cuts the link to income instead: the ratio becomes $\beta_{\text{commute}}/\beta_{\text{burden}}$, the same dollars for everybody.

➔ **The extension:** estimate how fast the value of time rises with income.

Problem Set 3: The Option to Stay

The first extension, on your own laptop, with an agent

 You get a new data file where a good number of households did not move at all. Add the option to stay, estimate it, and say whether the numbers came back. Along the way part of the model stops being identified, and finding out which part, and why, is the assignment.

1. Write the model down.

No AI.

2. Look before you estimate.

No AI for the reading.

3. Estimate it.

With AI.

4. Break it.

With AI.

5. The four questions.

With AI.

6. Half a page.

No AI.

 Individually, about six hours. Due before Lecture 5, as a PDF, by email to aiatyourside.00@gmail.com. The files are in `data_lecture4.zip`. It needs an agent that runs Python on your own laptop: the installation guide comes with the course materials.

What You Take With You Today

The loop

Model, pseudocode, code, test, inspect, revise. The pseudocode is ninety seconds and it saves the afternoon.

The four questions

What is it doing, why do we need it, what if I deleted it, how could I test it. Ask them of as many lines as you can, and never of none.

The check that works

Make up the answer, hide it, and see whether you get it back. It works on almost any estimator, unless the simulation and the estimator share the mistake.

⚠ And the three failures. The agent stepped around or caught all three today, and we saw each one by taking away what saved us: a calculation that breaks and raises no error, an object with the right name and the wrong meaning, and a question the data cannot answer that still prints a table.

You did not write the code. You are still answerable for what it says.

Next lecture: **the same skill, pointed at a decision**

Problem Set 3: add the option to stay, before Lecture 5



Book website