In-Class Activity: Model Fit and Counterfactuals

ECON 6343: Econometrics III
Prof. Tyler Ransom
University of Oklahoma

Overview

Today we'll practice the structural estimation workflow: assess model fit, run counterfactual simulations, and compute confidence intervals. This builds directly on Problem Set 4.

Required Packages

```
using Random, LinearAlgebra, Statistics, Optim
using DataFrames, CSV, HTTP, ForwardDiff
using FreqTables, Distributions
```

1 Setup (10 minutes)

1.1 Load Data and Estimates

Use your code from PS4 to load data and define the likelihood function:

```
Load data (same as PS4)
url = "https://raw.githubusercontent.com/OU-PhD-Econometrics/
fall-2020/master/ProblemSets/PS4-mixture/nlsw88t.csv"
df = CSV.read(HTTP.get(url).body, DataFrame)
X = [df.age df.white df.collgrad]
Z = hcat(df.elnwage1, df.elnwage2, df.elnwage3, df.elnwage4,
df.elnwage5, df.elnwage6, df.elnwage7, df.elnwage8)
y = df.occ_code
Include your likelihood function from PS4
include("mlogit_with_Z.jl")
Use your PS4 estimates as starting values
theta_start = [.0403744; .2439942; -1.57132; .0433254; .1468556;
-2.959103; .1020574; .7473086; -4.12005; .0375628;
.6884899; -3.65577; .0204543; -.3584007; -4.376929;
.1074636; -.5263738; -6.199197; .1168824; -.2870554;
-5.322248; 1.307477]
```

1.2 Estimate the Model

```
Estimate (same procedure as PS4)
td = TwiceDifferentiable(b -> mlogit_with_Z(b, X, Z, y),
theta_start; autodiff = :forward)
theta_optim = optimize(td, theta_start, LBFGS(),
Optim.Options(g_tol = 1e-5, iterations=100_000))
theta_mle = theta_optim.minimizer
H = Optim.hessian!(td, theta_mle)
println("Wage coefficient (gamma): ", round(theta_mle[end], digits=4))
```

2 Model Fit (15 minutes)

2.1 Define Prediction Function

```
function plogit(theta, X, Z, J)
alpha = theta[1:end-1]
gamma = theta[end]
K = size(X, 2)
bigalpha = [reshape(alpha, K, J-1) zeros(K)]
P = exp.(Xbigalpha .+ Zgamma) ./ sum.(eachrow(exp.(Xbigalpha .+ Zgamma)))
return P
end
```

2.2 Compare Model vs Data

```
Compute predicted probabilities
J = length(unique(y))
P = plogit(theta_mle, X, Z, J)
Create comparison table
modelfit_df = DataFrame(
occupation = 1:J,
data_pct = 100 * convert(Array, prop(freqtable(df, :occ_code))),
model_pct = 100 * vec(mean(P', dims=2))
)
modelfit_df.difference = modelfit_df.model_pct .- modelfit_df.data_pct
println("\nModel Fit:")
println(modelfit_df)
println("\nMean Absolute Error: ", round(mean(abs.(modelfit_df.difference)), digits=4))
```

Discussion: How well does the model fit? Which occupations fit best/worst?

3 Counterfactual Simulations (20 minutes)

Now we'll run three policy experiments:

3.1 Counterfactual 1: No Wage Effects

What if people didn't care about wages?

```
Set gamma = 0
theta_cfl1 = copy(theta_mle)
theta_cfl1[end] = 0
P_cfl1 = plogit(theta_cfl1, X, Z, J)
modelfit_df.cfl1_pct = 100 * vec(mean(P_cfl1', dims=2))
modelfit_df.cfl1_effect = modelfit_df.cfl1_pct .- modelfit_df.model_pct
modelfit_df.avg_wage = vec(mean(Z', dims=2))
```

Task: Which occupations gain/lose workers? How does this relate to wages?

3.2 Counterfactual 2: 10% Wage Increase

Universal wage increase of 10%

```
Increase all wages by 10%
Z_cfl2 = Z .* 1.10
P_cfl2 = plogit(theta_mle, X, Z_cfl2, J)
modelfit_df.cfl2_pct = 100 * vec(mean(P_cfl2', dims=2))
modelfit_df.cfl2_effect = modelfit_df.cfl2_pct .- modelfit_df.model_pct
```

Task: What happens? Why might effects be small?

3.3 Counterfactual 3: Targeted Wage Subsidy

20% wage increase for low-wage occupations (5-8)

```
Target low-wage occupations
Z_cfl3 = copy(Z)
Z_cfl3[:, 5:8] = Z[:, 5:8] .* 1.20
P_cfl3 = plogit(theta_mle, X, Z_cfl3, J)
modelfit_df.cfl3_pct = 100 * vec(mean(P_cfl3', dims=2))
modelfit_df.cfl3_effect = modelfit_df.cfl3_pct .- modelfit_df.model_pct
println("\nCounterfactual Results:")
println(modelfit_df)
```

Discussion: Compare all three counterfactuals. Which is most effective? Why?

4 Bootstrap Confidence Intervals (15 minutes)

Quantify uncertainty for Counterfactual 1:

```
Parametric bootstrap
Random.seed!(1234)
invH = inv(H)
invH_sym = (invH + invH') / 2
d = MvNormal(theta_mle, invH_sym)
B = 1000
cfl_bs = zeros(J, B)
println("Running bootstrap...")
for b = 1:B
theta_draw = rand(d)
# Baseline
P_b = plogit(theta_draw, X, Z, J)
P_base = 100 * vec(mean(P_b', dims=2))

# Counterfactual
theta_draw[end] = 0
P_cfl_b = plogit(theta_draw, X, Z, J)
P_cfl_draw = 100 * vec(mean(P_cfl_b', dims=2))

cfl_bs[:, b] = P_cfl_draw .- P_base

if b % 200 == 0
    println("  $b/$B complete")
end
end
Compute 95% CIs
modelfit_df.ci_lower = [quantile(cfl_bs[j,:], 0.025) for j=1:J]
modelfit_df.ci_upper = [quantile(cfl_bs[j,:], 0.975) for j=1:J]
println("\nResults with 95% CIs:")
println(modelfit_df[:, [:occupation, :cfl1_effect, :ci_lower, :ci_upper]])
```

Task: Which effects are statistically significant? What does this mean for policy?

5 Wrap-Up (5 minutes)

5.1 Key Takeaways

1. **Model fit** validates your model before counterfactuals
2. **Counterfactuals** answer policy questions by changing parameters
3. **Bootstrap** quantifies uncertainty in predictions
4. **Assumptions matter**: We assume parameter invariance and partial equilibrium

5.2 Critical Assumptions

When running counterfactuals, we assume:

- Other parameters don't change (e.g., β's stay fixed)
- No general equilibrium effects (wages don't adjust)
- Model structure remains valid under policy change

5.3 Extensions

Try on your own:

- Bootstrap CIs for other counterfactuals
- Education policy: set everyone as college graduate
- Holdout validation: 80/20 train/test split
- Sensitivity to starting values