Lecture 18: Simulation-Based Estimation II

ResEcon 703: Topics in Advanced Econometrics

Matt Woerman
University of Massachusetts Amherst

Agenda

Last time

- Simulation-Based Estimation

Today

- Simulation-Based Estimation Example in R

Upcoming

- Reading for next time
  - Train textbook, Chapter 11
- Problem sets
  - Problem Set 4 is posted, due November 21

Maximum Simulated Likelihood Estimation

To estimate a mixed logit model, we have to use maximum simulated likelihood estimation (MSLE), or another simulation-based estimator, because mixed logit choice probabilities do not have a closed-form solution

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \sum_{n=1}^{N} \sum_{j=1}^{J} y_{nj} \ln \check{P}_{nj}(\theta)$$

where $\check{P}_{nj}(\theta)$ is the simulated choice probability

$$\check{P}_{ni} = \frac{1}{R} \sum_{r=1}^{R} L_{ni}(\beta^r)$$

To simulate this choice probability for a given set of parameters, θ,

1. Draw a set of coefficients, β^r, from the density $f(\beta \mid \theta)$
2. Calculate the conditional probability, $L_{ni}(\beta^r)$, for each alternative
3. Repeat steps 1 and 2 for a total of R draws from $f(\beta \mid \theta)$
4. Average over these R draws to get $\check{P}_{ni}$ for each alternative

Simulation-Based Estimation Example in R

Maximum Simulated Likelihood Estimation Example

We are again studying how consumers make choices about expensive and highly energy-consuming systems in their homes. We have data on 250 households in California and the type of HVAC (heating, ventilation, and air conditioning) system in their home. Each household has the following choice set, and we observe the following data

Choice set

- GCC: gas central with AC
- ECC: electric central with AC
- ERC: electric room with AC
- HPC: heat pump with AC
- GC: gas central
- EC: electric central
- ER: electric room

Alternative-specific data

- ICH: installation cost for heat
- ICCA: installation cost for AC
- OCH: operating cost for heat
- OCCA: operating cost for AC

Household demographic data

- income: annual income

Load Dataset

```r
### Load and look at dataset
## Load tidyverse and mlogit
library(tidyverse)
library(mlogit)
## Load dataset from mlogit package
data('HC', package = 'mlogit')
```

Dataset

```
## Look at dataset
as_tibble(HC)
## # A tibble: 250 x 18
##    depvar ich.gcc ich.ecc ich.erc ich.hpc ich.gc ich.ec ich.er  icca
##    <fct>    <dbl>   <dbl>   <dbl>   <dbl>  <dbl>  <dbl>  <dbl> <dbl>
##  1 erc       9.7     7.86    8.79   11.4    24.1   24.5   7.37  27.3
##  2 hpc       8.77    8.69    7.09    9.37   28     32.7   9.33  26.5
##  3 gcc       7.43    8.86    6.94   11.7    25.7   31.7   8.14  22.6
##  4 gcc       9.18    8.93    7.22   12.1    29.7   26.7   8.04  25.3
##  5 gcc       8.05    7.02    8.44   10.5    23.9   28.4   7.15  25.4
##  6 gcc       9.32    8.03    6.22   12.6    27.0   21.4   8.6   19.9
##  7 gc        7.11    8.78    7.36   12.4    22.9   28.6   6.41  27.0
##  8 hpc       9.38    7.48    6.72    8.93   26.2   27.9   7.3   18.1
##  9 gcc       8.08    7.39    8.79   11.2    23.0   22.6   7.85  22.6
## 10 gcc       6.24    4.88    7.46    8.28   19.8   27.5   6.88  25.8
## # ... with 240 more rows, and 9 more variables: och.gcc <dbl>,
## #   och.ecc <dbl>, och.erc <dbl>, och.hpc <dbl>, och.gc <dbl>,
## #   och.ec <dbl>, och.er <dbl>, occa <dbl>, income <dbl>
```

Format Dataset in a Long Format

```r
### Format dataset
## Gather into a long dataset
hvac_long <- HC %>%
  mutate(id = 1:n()) %>%
  gather(key, value, starts_with('ich.'), starts_with('och.')) %>%
  separate(key, c('cost', 'alt')) %>%
  spread(cost, value) %>%
  mutate(choice = (depvar == alt)) %>%
  select(-depvar)
```

Dataset in a Long Format

```
## Look at long dataset
as_tibble(hvac_long)
## # A tibble: 1,750 x 8
##      icca  occa income    id alt     ich   och choice
##     <dbl> <dbl>  <dbl> <int> <chr> <dbl> <dbl> <lgl>
##  1  17    2.79     60   133 ec    20.3   4.52 FALSE
##  2  17    2.79     60   133 ecc    8.46  4.52 FALSE
##  3  17    2.79     60   133 er     7.7   4.32 FALSE
##  4  17    2.79     60   133 erc    8.16  4.32 FALSE
##  5  17    2.79     60   133 gc    25.3   2.26 FALSE
##  6  17    2.79     60   133 gcc    6.33  2.26 TRUE
##  7  17    2.79     60   133 hpc   11.1   1.63 FALSE
##  8  18.1  2.55     50    14 ec    25.6   5.21 FALSE
##  9  18.1  2.55     50    14 ecc   11.2   5.21 FALSE
## 10  18.1  2.55     50    14 er     9.3   3.8  FALSE
## # ... with 1,740 more rows
```

Clean Dataset

```r
## Combine heating and cooling costs into one variable
hvac_clean <- hvac_long %>%
  mutate(cooling = (nchar(alt) == 3),
         ic = if_else(cooling, ich + icca, ich),
         oc = if_else(cooling, och + occa, och)) %>%
  mutate(cooling = 1 * cooling,
         ic = -ic,
         oc = -oc) %>%
  select(id, alt, choice, cooling, ic, oc, income)
```

Cleaned Dataset

```
## Look at cleaned dataset
as_tibble(hvac_clean)
## # A tibble: 1,750 x 7
##       id alt   choice cooling    ic    oc income
##    <int> <chr> <lgl>    <dbl> <dbl> <dbl>  <dbl>
##  1   133 ec    FALSE        0 -20.3 -4.52     60
##  2   133 ecc   FALSE        1 -25.5 -7.31     60
##  3   133 er    FALSE        0  -7.7 -4.32     60
##  4   133 erc   FALSE        1 -25.2 -7.11     60
##  5   133 gc    FALSE        0 -25.3 -2.26     60
##  6   133 gcc   TRUE         1 -23.3 -5.05     60
##  7   133 hpc   FALSE        1 -28.1 -4.42     60
##  8    14 ec    FALSE        0 -25.6 -5.21     50
##  9    14 ecc   FALSE        1 -29.2 -7.76     50
## 10    14 er    FALSE        0  -9.3 -3.8      50
## # ... with 1,740 more rows
```

Mixed Logit Model to Estimate with MSLE

The representative utility of each alternative is

$$V_{nj} = \alpha AC_j + \beta_1 IC_{nj} + \beta_2 OC_{nj}$$

with

$$\ln \beta_1 \sim N(\mu_1, \sigma_1^2)$$
$$\ln \beta_2 \sim N(\mu_2, \sigma_2^2)$$

What are the MSLE parameters for this model?

What is the elasticity of each alternative with respect to the installation cost (IC) of a central gas system with AC (GCC)?

Steps for Simulation-Based Estimation

1. Draw $K \times N \times R$ standard normal random variables
   - K random coefficients for each of
   - N different decision makers for each of
   - R different simulation draws
2. Find the set of parameters that maximizes or minimizes the objective function of a simulation-based estimator
   1. Start with some set of parameters, θ^0
   2. Simulate choice probabilities for this set of parameters, $\check{P}_{ni}(\theta^t)$
      1. Transform each set of K standard normals using θ^t to get a set of β_n^r
      2. Calculate the choice probabilities for each individual and draw, $L_{ni}(\beta_n^r)$
      3. Average over all R simulation draws to get $\check{P}_{ni}(\theta^t)$
   3. Use these simulated choice probabilities to calculate simulated log-likelihood, simulated moments, etc.
   4. Step to a better set of parameters, θ^{t+1}
   5. Repeat steps 2 and 3 until the algorithm converges to a set of parameters that is your simulation-based estimator

map() Function in R

We will use the map() function, and related functions, to help with our simulation

- map() applies a function to each element of a vector or list
- map2() applies a function to elements from two vectors or lists
- pmap() is similar but allows for any number of vectors or lists

```
### Map function in R
## List to pass to the map function
list(1:5, 6:10)
## [[1]]
## [1] 1 2 3 4 5
##
## [[2]]
## [1]  6  7  8  9 10

## Take mean of each list element
list(1:5, 6:10) %>%
  map(~ mean(.x))
## [[1]]
## [1] 3
##
## [[2]]
## [1] 8
```

Step 0: Set a Seed for Replication

We first set a seed so we can replicate our random simulation draws

```
### Set a seed for replication
## Random draws without setting a seed
rnorm(5)
## [1] -1.02642786  1.97342317  0.56796636  0.58073292  0.02490352

rnorm(5)
## [1] -0.7259931  0.5011211  0.5906434 -0.2886113  0.8112558

## Random draws with the same seed
set.seed(703)
rnorm(5)
## [1] -1.313404  0.865439 -1.247334  0.598521 -1.224091

set.seed(703)
rnorm(5)
## [1] -1.313404  0.865439 -1.247334  0.598521 -1.224091

## Set seed for replication
set.seed(703)
```

Step 1: Draw Random Variables

Draw $K \times N \times R$ standard normal random variables and organize into a list with each element corresponding to one individual

```
### Draw random variable for random coefficients
## Draw standard normal random variables and split into list
draws_list <- 1:250 %>%
  map(., ~ tibble(ic_coef = rnorm(100),
                  oc_coef = rnorm(100)))
draws_list[1]
## [[1]]
## # A tibble: 100 x 2
##    ic_coef oc_coef
##      <dbl>   <dbl>
##  1  -1.31    0.107
##  2   0.865  -0.935
##  3  -1.25   -0.304
##  4   0.599   0.160
##  5  -1.22   -1.09
##  6  -0.231   0.105
##  7  -0.708   0.708
##  8   0.444  -1.55
##  9  -1.47   -0.467
## 10  -0.347   0.967
## # ... with 90 more rows
```

Step 1.5: Organize Data

Organize data into a list with each element corresponding to one individual to be compatible with random draws

```
### Organize data for MSLE optimization
## Split data into list by household
data_list <- hvac_clean %>%
  arrange(id, alt) %>%
  group_by(id) %>%
  group_split()
data_list[1]
## [[1]]
## # A tibble: 7 x 7
##        id alt   choice cooling     ic    oc income
##     <int> <chr> <lgl>    <dbl>  <dbl> <dbl>  <dbl>
## 1       1 ec    FALSE        0 -24.5  -4.09     20
## 2       1 ecc   FALSE        1 -35.1  -7.04     20
## 3       1 er    FALSE        0  -7.37 -3.85     20
## 4       1 erc   TRUE         1 -36.1  -6.8      20
## 5       1 gc    FALSE        0 -24.1  -2.26     20
## 6       1 gcc   FALSE        1 -37.0  -5.21     20
## 7       1 hpc   FALSE        1 -38.6  -4.68     20
```

Step 2: Find the MSLE

```r
### Optimization in R
## Help file for the optimization function, optim
?optim
## Arguments for optim function
optim(par, fn, gr, ..., method, lower, upper, control, hessian)
```

`optim()` requires that you create a function, `fn`, that

1. Takes a set of parameters and data as inputs
2. Calculates your objective function given those parameters
3. Returns this value of the objective function

Some `control` arguments may be useful when doing optimization that takes longer to converge

- `trace`: 1 will report progress of convergence
- `REPORT`: How often (number of iterations) to report on convergence

Step 2.2–2.3: Choice Probabilities and Log-Likelihood

To estimate a multinomial logit model using MLE, we created a single function that calculated choice probabilities and then used them to calculate the log-likelihood

To estimate a mixed logit model using MSLE, we will create two separate functions

- Function 1: Simulate choice probabilities for a single decision maker (household)
- Function 2: Use simulated choice probabilities to calculate simulated log-likelihood

We do not have to split this process into two functions, but it makes the code more transparent (and probably slower...)

Simulate Choice Probabilities for One Household

```r
### Find MSLE estimator for mixed logit with random coefficients
## Function to simulate choice probabilities for one household
simulate_probabilities <- function(parameters, draws, data){
  ## Select relevant variables and convert into a matrix [J * K]
  data_matrix <- data %>%
    select(cooling, ic, oc) %>%
    as.matrix()
  ## Transform random coefficients based on parameters [R * K]
  coefficients <- draws %>%
    mutate(cooling_coef = parameters[1],
           ic_coef = exp(parameters[2] + parameters[4] * ic_coef),
           oc_coef = exp(parameters[3] + parameters[5] * oc_coef)) %>%
    select(cooling_coef, ic_coef, oc_coef)
  ## Calculate utility for each alternative in each draw [R * J]
  utility <- (as.matrix(coefficients) %*% t(data_matrix)) %>%
    pmin(700) %>%
    pmax(-700)
  ## Sum the exponential of utility over alternatives [R * 1]
  summed_utility <- utility %>%
    exp() %>%
    rowSums()
  ## Calculate the conditional probability for each alternative and draw [R * J]
  conditional_probability <- exp(utility) / summed_utility
  ## Average conditional probabilities over all draws [1 * J]
  simulated_probability <- colMeans(conditional_probability)
  ## Add simulated probability to initial dataset
  data_out <- data %>%
    mutate(probability = simulated_probability)
  ## Return initial dataset with simulated probability variable
  return(data_out)
}
```

Calculate Simulated Log-Likelihood

```r
## Function to calculate simulated log-likelihood
simulate_log_likelihood <- function(parameters, draws_list, data_list){
  ## Simulate probabilities for each household
  data <- map2(.x = draws_list, .y = data_list,
               .f = ~ simulate_probabilities(parameters = parameters,
                                             draws = .x,
                                             data = .y))
  ## Combine individual datasets into one
  data <- data %>%
    bind_rows()
  ## Calcule the log of simulated probability for the chosen alternative
  data <- data %>%
    filter(choice == TRUE) %>%
    mutate(log_probability = log(probability))
  ## Calculate the simulated log-likelihood
  simulated_log_likelihood <- sum(data$log_probability)
  ## Return the negative of simulated log-likelihood
  return(-simulated_log_likelihood)
}
```

Maximize Simulated Log-Likelihood

```r
## Maximize the log-likelihood function
model <- optim(par = c(6.53, log(0.174), log(1.04), 0, 0),
               fn = simulate_log_likelihood,
               draws_list = draws_list, data_list = data_list,
               method = 'BFGS', hessian = TRUE,
               control = list(trace = 1, REPORT = 5))
## initial  value 329.883895
## iter   5 value 329.564176
## iter  10 value 324.533793
## iter  15 value 316.547576
## iter  20 value 312.568560
## iter  25 value 310.847759
## iter  30 value 310.768625
## final  value 310.768592
## converged
```

MSLE Optimization Results

```
## Report model results
model
## $par
## [1] 21.394424175 -0.443357857  0.491331484  0.001841538 -1.032088604
##
## $value
## [1] 310.7686
##
## $counts
## function gradient
##       93       31
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##             [,1]       [,2]       [,3]       [,4]       [,5]
## [1,]    7.268596 -143.48065  -5.512424  -2.307161   3.485463
## [2,] -143.480651 2951.97793  31.175840  42.295369 -11.598448
## [3,]   -5.512424   31.17584 107.424808   2.646411 -46.423371
## [4,]   -2.307161   42.29537   2.646411 332.456394   5.786799
## [5,]    3.485463  -11.59845 -46.423371   5.786799 103.157399
```

MSLE Parameters and Standard Errors

```r
## Report parameter estimates and standard errors
model$par
## [1] 21.394424175 -0.443357857  0.491331484  0.001841538 -1.032088604

model$hessian %>%
  solve() %>%
  diag() %>%
  sqrt()
## [1] 2.79310027 0.13590931 0.13829674 0.05499237 0.11696267
```

Mixed Logit Elasticities

The elasticity of alternative i with respect to the mth attribute of alternative j is

$$\text{Own: } E_{nix_{ni}^m} = \frac{x_{ni}^m}{P_{ni}} \int \beta^m L_{ni}(\beta)[1 - L_{ni}(\beta)] f(\beta \mid \theta) d\beta$$

$$\text{Cross: } E_{nix_{nj}^m} = -\frac{x_{nj}^m}{P_{ni}} \int \beta^m L_{ni}(\beta) L_{nj}(\beta) f(\beta \mid \theta) d\beta$$

To simulate these elasticities at our MSLE, $\hat{\theta}$,

1. Draw a set of coefficients, β^r, from the density $f(\beta \mid \hat{\theta})$
2. Calculate the conditional probability, $L_{ni}(\beta^r)$, for each alternative
3. Calculate the term inside the integral for each elasticity
4. Repeat steps 1–3 for a total of R draws from $f(\beta \mid \theta)$
5. Average over these R draws to simulate the integral for each elasticity
6. Multiply each integral by the ratio in front

Simulate Elasticities for One Household

$$\text{Own: } E_{nix^m_{ni}} = \frac{x^m_{ni}}{P_{ni}} \int \beta^m L_{ni}(\beta)[1 - L_{ni}(\beta)] f(\beta \mid \theta) d\beta$$

$$\text{Cross: } E_{nix^m_{nj}} = -\frac{x^m_{nj}}{P_{ni}} \int \beta^m L_{ni}(\beta) L_{nj}(\beta) f(\beta \mid \theta) d\beta$$

```r
### Find elasticities with respect to the installation cost (ic) of
### gas central with AC (gcc)
## Function to simulate elasticities for one household
simulate_elasticities <- function(parameters, draws, data){
  ## Select relevant variables and convert into a matrix [J * K]
  data_matrix <- data %>%
    select(cooling, ic, oc) %>%
    as.matrix()
  ## Transform random coefficients based on parameters [R * K]
  coefficients <- draws %>%
    mutate(cooling_coef = parameters[1],
           ic_coef = exp(parameters[2] + parameters[4] * ic_coef),
           oc_coef = exp(parameters[3] + parameters[5] * oc_coef)) %>%
    select(cooling_coef, ic_coef, oc_coef)
  ## Calculate utility for each alternative in each draw [R * J]
  utility <- (as.matrix(coefficients) %*% t(data_matrix)) %>%
    pmin(700) %>%
    pmax(-700)
  ## Sum the exponential of utility over alternatives [R * 1]
  summed_utility <- utility %>%
    exp() %>%
    rowSums()
```

Simulate Elasticities for One Household

$$\text{Own: } E_{nix^m_{ni}} = \frac{x^m_{ni}}{P_{ni}} \int \beta^m L_{ni}(\beta)[1 - L_{ni}(\beta)] f(\beta \mid \theta) d\beta$$

$$\text{Cross: } E_{nix^m_{nj}} = -\frac{x^m_{nj}}{P_{ni}} \int \beta^m L_{ni}(\beta) L_{nj}(\beta) f(\beta \mid \theta) d\beta$$

```
## Calculate the conditional probability for each alternative and draw [R * J]
conditional_probability <- exp(utility) / summed_utility
## Average conditional probabilities over all draws [1 * J]
simulated_probability <- colMeans(conditional_probability)
## Calculate integral term for own elasticity for each draw [R * 1]
own_elasticity_integral <- coefficients$ic_coef *
  conditional_probability[, 6] *
  (1 - conditional_probability[, 6])
## Calculate integral term for cross elasticities for each draw [R * (J - 1)]
cross_elasticity_integral <- coefficients$ic_coef *
  conditional_probability[, 6] *
  conditional_probability[, -6]
## Combine previous terms in correct order [R * J]
elasticity_integral <- cross_elasticity_integral[, 1:5] %>%
  cbind(own_elasticity_integral) %>%
  cbind(cross_elasticity_integral[, 6]) %>%
  unname()
## Average elasticity integral terms to simulate integral [1 * J]
simulated_integral <- colMeans(elasticity_integral)
## Calculate own-price and cross-price simulated elasticities [1 * J]
simulated_elasticity <- c(rep(-1, 5), 1, -1) * data$ic[6] /
  simulated_probability * simulated_integral
```

Simulate Elasticities for One Household

$$\text{Own: } E_{nix^m_{ni}} = \frac{x^m_{ni}}{P_{ni}} \int \beta^m L_{ni}(\beta)[1 - L_{ni}(\beta)] f(\beta \mid \theta) d\beta$$

$$\text{Cross: } E_{nix^m_{nj}} = -\frac{x^m_{nj}}{P_{ni}} \int \beta^m L_{ni}(\beta) L_{nj}(\beta) f(\beta \mid \theta) d\beta$$

```
## Add simulated elasticities to initial dataset
  data_out <- data %>%
    mutate(elasticity = simulated_elasticity)
  ## Return initial dataset with simulated probability variable
  return(data_out)
}
```

Simulated Elasticities

```
## Simulate elasticities for each household
data_list <- map2(.x = draws_list, .y = data_list,
                  .f = ~ simulate_elasticities(parameters = model$par,
                                               draws = .x,
                                               data = .y))
## Combine list of data into one tibble
data <- data_list %>%
  bind_rows()
## Calculate average elasticity with respect to ic of gcc
data %>%
  group_by(alt) %>%
  summarize(elasticity = mean(elasticity)) %>%
  ungroup()
## # A tibble: 7 x 2
##   alt   elasticity
##   <chr>      <dbl>
## 1 ec         7.87
## 2 ecc       10.1
## 3 er         8.14
## 4 erc       10.0
## 5 gc         0.319
## 6 gcc      -10.9
## 7 hpc        7.83
```

Announcements

Reading for next time

- Train textbook, Chapter 11

Upcoming

- Problem Set 4 is posted, due November 21