Week 11: Simulation-Based Estimation

ResEcon 703: Topics in Advanced Econometrics

Matt Woerman
University of Massachusetts Amherst

Agenda

Last week

- Mixed logit model

This week

- Simulated choice probabilities
- Simulation-based estimators
- Properties of simulation-based estimators
- Simulation details
- Simulation-based estimation R example

This week's reading

- Train textbook, chapter 10

Simulated Choice Probabilities

Mixed Logit Choice Probability

$$P_{ni} = \int L_{ni}(\beta) f(\beta \mid \theta) d\beta$$

$$L_{ni}(\beta) = \frac{e^{V_{ni}(\beta)}}{\sum_{j=1}^{J} e^{V_{nj}(\beta)}}$$

The mixed logit choice probability is a weighted average of logit choice probabilities

- The logit choice probabilities are evaluated at different values of β
- Each logit choice probability is weighted by the density $f(\beta \mid \theta)$

This choice probability does not have a closed-form solution, so we cannot calculate it directly

- But we can approximate it using numerical simulation

Simulated Choice Probability Intuition

$$P_{ni} = \int L_{ni}(\beta) f(\beta \mid \theta) d\beta$$

$$L_{ni}(\beta) = \frac{e^{V_{ni}(\beta)}}{\sum_{j=1}^{J} e^{V_{nj}(\beta)}}$$

We effectively want to do the following:

1. Calculate the logit choice probability at every possible β, $L_{ni}(\beta)$
2. Weight each of these values by the likelihood of observing that β in the population, $f(\beta \mid \theta)$
3. Sum these weighted logit choice probabilities

But β is usually continuous, so it takes on infinitely many values

- We will approximate this procedure by considering only a finite random sample of all possible values of β

Simulated Choice Probability

$$P_{ni} = \int L_{ni}(\beta) f(\beta \mid \theta) d\beta$$

$$L_{ni}(\beta) = \frac{e^{V_{ni}(\beta)}}{\sum_{j=1}^{J} e^{V_{nj}(\beta)}}$$

Given a set of parameters, θ, that defines the random coefficient distributions, $f(\beta \mid \theta)$, we can simulate this mixed logit choice probability

1. Draw R random vectors from $f(\beta \mid \theta)$, denoted $\{\beta^1, \beta^2, \ldots, \beta^R\}$
2. For each random vector, β^r, calculate the conditional logit choice probability, $L_{ni}(\beta^r)$
3. Average over these R conditional logit choice probabilities

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

Simulated Choice Probability Details

Step 1: Draw R random vectors from $f(\beta \mid \theta)$, denoted $\{\beta^1, \beta^2, \ldots, \beta^R\}$

- If β contains K random coefficients, then we need $R \times K$ total random variables
- We are incorporating the "weighting" into our simulation by drawing these coefficients from $f(\beta \mid \theta)$
  - We are more likely to draw coefficients with a greater probability density or "weighting"
- See chapter 9 of the Train textbook for more details on drawing random variables

Simulated Choice Probability Details

Step 2: For each random vector, β^r, calculate the conditional logit choice probability, $L_{ni}(\beta^r)$

$$L_{ni}(\beta^r) = \frac{e^{V_{ni}(\beta^r)}}{\sum_{j=1}^{J} e^{V_{nj}(\beta^r)}}$$

- We will have R conditional logit choice probabilities for a single alternative for one decision maker
  - We would have $R \times N \times J$ total conditional choice probabilities if we calculated a conditional choice probability for every random coefficient vector draw for each decision maker for every alternative

Simulated Choice Probability Details

Step 3: Average over these R conditional logit choice probabilities

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

- This simulated integral converges almost surely to the actual integral with the number of random draws, R, so long as the random draws are from $f(\beta \mid \theta)$

$$\frac{1}{R} \sum_{r=1}^{R} L_{ni}(\beta^r) \overset{a.s.}{\to} \int L_{ni}(\beta) f(\beta \mid \theta) d\beta$$

Simulation-Based Estimators

Simulation-Based Estimation

Simulation-based estimators are roughly equivalent to their traditional analogs

- We replace terms that are difficult or impossible to calculate with their simulated counterparts
  - Example: Mixed logit choice probabilities include an integral and do not have a closed-form expression, so we replace them with simulated choice probabilities
- Simulation can potentially introduce bias or noise into the estimation, which we need to consider when using simulation-based estimators

Maximum Simulated Likelihood Estimation

Maximum simulated likelihood (MSL) estimation, or simulated maximum likelihood estimation, is the simulation analog of maximum likelihood estimation

The maximum simulated likelihood estimator is the set of parameters that maximizes the simulated log-likelihood

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \ln \check{L}(\boldsymbol{\theta} \mid \boldsymbol{y}, \boldsymbol{X})$$

where $\ln \check{L}(\boldsymbol{\theta} \mid \boldsymbol{y}, \boldsymbol{X})$ is the log of the simulated likelihood

$$\ln \check{L}(\boldsymbol{\theta} \mid \boldsymbol{y}, \boldsymbol{X}) = \sum_{n=1}^{N} \ln \check{f}(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta})$$

and $\check{f}(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta})$ is a simulated density function

Maximum Simulated Likelihood with Discrete Choice

For discrete choice applications, the log of simulated likelihood is a function of simulated choice probabilities

$$\ln \check{L}(\boldsymbol{\theta} \mid \boldsymbol{y}, \boldsymbol{X}) = \sum_{n=1}^{N} \sum_{i=1}^{J} y_{ni} \ln \check{P}_{ni}(\boldsymbol{x}_n, \boldsymbol{\theta})$$

so the MSL estimator for a discrete choice model is

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^{N} \sum_{i=1}^{J} y_{ni} \ln \check{P}_{ni}(\boldsymbol{x}_n, \boldsymbol{\theta})$$

which gives first-order conditions equivalent to

$$\sum_{n=1}^{N} \sum_{i=1}^{J} y_{ni} \frac{\partial \ln \check{P}_{ni}(\boldsymbol{x}_n, \widehat{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} = \mathbf{0}$$

Method of Simulated Moments

Method of simulated moments (MSM), or simulated method of moments, is the simulation analog of generalized method of moments

The method of simulated moments estimator is the set of parameters that "solves" a set of simulated moments

$$\frac{1}{N}\sum_{n=1}^{N}\check{\boldsymbol{m}}(y_n, \boldsymbol{x}_n, \boldsymbol{z}_n, \widehat{\boldsymbol{\theta}}) = \mathbf{0}$$

where $\check{\boldsymbol{m}}(y_n, \boldsymbol{x}_n, \boldsymbol{z}_n, \widehat{\boldsymbol{\theta}})$ are the simulated sample moments that are the simulated empirical analogs of population moment conditions

$$E[\boldsymbol{m}(y_n, \boldsymbol{x}_n, \boldsymbol{z}_n, \boldsymbol{\theta})] = \mathbf{0}$$

With more moments than parameters, we cannot solve this system of equations, so we instead minimize the weighted sum of squared simulated sample moments

Method of Simulated Moments with Discrete Choice

For discrete choice applications, the population moments result from the econometric residuals being orthogonal to a set of exogenous instruments

$$E\left[(y_{ni} - P_{ni}(\boldsymbol{x}_n, \boldsymbol{\theta}))\boldsymbol{z}_{ni}\right] = \mathbf{0}$$

so the MSM estimator for a discrete choice model "solves" the simulated sample moments

$$\frac{1}{NJ}\sum_{n=1}^{N}\sum_{i=1}^{J}\left(y_{ni} - \check{P}_{ni}(\boldsymbol{x}_n, \widehat{\boldsymbol{\theta}})\right)\boldsymbol{z}_{ni} = \mathbf{0}$$

or minimizes the weighted sum of squared simulated sample moments

Properties of Simulation-Based Estimators

Traditional Estimators

Traditional estimators are the set of parameters that solves a specific system of functions that can be expressed as sample means

$$\boldsymbol{g}(\widehat{\boldsymbol{\theta}}) = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{g}_n(\widehat{\boldsymbol{\theta}}) = \mathbf{0}$$

ML estimation of a discrete choice model uses the functions

$$\boldsymbol{g}_n(\boldsymbol{\theta}) = \sum_{i=1}^{J} y_{ni} \frac{\partial \ln P_{ni}(\boldsymbol{x}_n, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

GMM estimation of a discrete choice model uses the functions

$$\boldsymbol{g}_n(\boldsymbol{\theta}) = \frac{1}{J}\sum_{i=1}^{J} (y_{ni} - P_{ni}(\boldsymbol{x}_n, \boldsymbol{\theta}))\boldsymbol{z}_{ni}$$

When certain assumptions are met, these estimators yield consistent estimates of the true set of parameters, $\boldsymbol{\theta}_0$

Simulation-Based Estimators

Simulation-based estimators are the set of parameters that solves the simulation-based analog of these functions

$$\check{\boldsymbol{g}}(\widehat{\theta}) = \frac{1}{N}\sum_{n=1}^{N}\check{\boldsymbol{g}}_n(\widehat{\theta}) = \mathbf{0}$$

where $\check{\boldsymbol{g}}_n(\boldsymbol{\theta})$ is the simulation-based analog of $\boldsymbol{g}_n(\boldsymbol{\theta})$

We can express the simulation-based functions as

$$\begin{aligned}\check{\boldsymbol{g}}(\boldsymbol{\theta}) &= \check{\boldsymbol{g}}(\boldsymbol{\theta}) + (\boldsymbol{g}(\boldsymbol{\theta}) - \boldsymbol{g}(\boldsymbol{\theta})) + (E_r[\check{\boldsymbol{g}}(\boldsymbol{\theta})] - E_r[\check{\boldsymbol{g}}(\boldsymbol{\theta})]) \\ &= \boldsymbol{g}(\boldsymbol{\theta}) + (E_r[\check{\boldsymbol{g}}(\boldsymbol{\theta})] - \boldsymbol{g}(\boldsymbol{\theta})) + (\check{\boldsymbol{g}}(\boldsymbol{\theta}) - E_r[\check{\boldsymbol{g}}(\boldsymbol{\theta})])\end{aligned}$$

where $E_r[\check{\boldsymbol{g}}(\boldsymbol{\theta})]$ is the expectation of $\check{\boldsymbol{g}}(\boldsymbol{\theta})$ over the simulation draws

Simulation Bias and Noise

We can express the simulation-based sample mean functions as

$$\check{\boldsymbol{g}}(\boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta}) + (E_r[\check{\boldsymbol{g}}(\boldsymbol{\theta})] - \boldsymbol{g}(\boldsymbol{\theta})) + (\check{\boldsymbol{g}}(\boldsymbol{\theta}) - E_r[\check{\boldsymbol{g}}(\boldsymbol{\theta})])$$

A simulation-based estimator can differ from a traditional estimator for two reasons

- Simulation bias: $E_r[\check{\boldsymbol{g}}(\boldsymbol{\theta})] - \boldsymbol{g}(\boldsymbol{\theta})$
- Simulation noise: $\check{\boldsymbol{g}}(\boldsymbol{\theta}) - E_r[\check{\boldsymbol{g}}(\boldsymbol{\theta})]$

How do we reduce the simulation bias and noise in a simulation-based estimator?

- To reduce simulation bias, increase the number of simulation draws, R
- To reduce simulation noise, increase the sample size, N

Properties of Simulation-Based Estimators

With a sufficient number of simulation draws and a sufficient sample size, a simulation-based estimator is:

- Consistent
- Asymptotically normal
- Sometimes equivalent to (or converging to) its traditional analog

The specifics depend on the estimator in question

- See the Train textbook for details of each simulation-based estimator

Simulation Details

Dependence of Simulated Choice Probabilities

The simulated choice probability that decision maker n chooses alternative i is

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

We need to simulate choice probabilities for every alternative for each decision maker

- For a given decision maker, use the same set of β^r random draws for every alternative in order to maintain dependence between the alternatives
  - That is, we use a single set of R draws to calculate all J alternatives for a decision maker
- Use a different set of β^r draws for each decision maker in order to maintain independence between decision makers
  - That is, we need N different sets of R draws

Numerical Optimization with Simulation

We use these simulated choice probabilities within a numerical optimization procedure to find the estimator, $\widehat{\theta}$

- We want to use the "same" set of β^r random draws for a given decision maker throughout the numerical optimization procedure
- If we use different random draws for each iteration of the procedure, we introduce additional noise that impedes convergence

In order to avoid this additional noise

1. Draw many $(K \times N \times R)$ random variables from a standard normal distribution before starting the numerical optimization
2. Transform this same set of standard normal random variables in each iteration of the optimization algorithm to represent $f(\beta \mid \theta)$ for the set of parameters, θ, of that iteration

Transforming a Standard Normal Random Variable

We can transform a standard normal random variable (or a vector of standard normals) into many other distributions

1. Draw K standard normal random variables, $\omega \sim \mathcal{N}(0, 1)$, where K is the number of random parameters
2. Transform these standard normals into the desired distributions
   - Normal: $\beta = \mu + \sigma\omega$ gives $\beta \sim \mathcal{N}(\mu, \sigma^2)$
   - Log-normal: $\beta = e^{\mu+\sigma\omega}$ gives $\ln \beta \sim \mathcal{N}(\mu, \sigma^2)$
   - Multivariate normal: $\boldsymbol{\beta} = \boldsymbol{\mu} + \boldsymbol{L\omega}$ gives $\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\boldsymbol{\beta}$, $\boldsymbol{\mu}$, and $\boldsymbol{\omega}$ are each a vector of length equal to the number of multivariate normal random variables, $\boldsymbol{\Sigma}$ is the variance-covariance matrix of these variables, and $\boldsymbol{L}$ is the Choleski factor of $\boldsymbol{\Sigma}$
   - Comparable transformations exist for most distributions

See chapter 9 in the Train textbook for more on drawing and transforming random variables

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, $\boldsymbol{\theta}^0$
   2. Simulate choice probabilities for the current set of parameters, $\check{P}_{ni}(\boldsymbol{\theta}^s)$
      1. Transform each set of K standard normals using $\boldsymbol{\theta}^s$ 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 $\check{P}_{ni}(\boldsymbol{\theta}^s)$
   3. Use these simulated choice probabilities to calculate simulated log-likelihood, simulated moments, etc.
   4. Step to a better set of parameters, $\boldsymbol{\theta}^{s+1}$
   5. Repeat steps (2)–(4) until the algorithm converges to a set of parameters that is your simulation-based estimator

Simulation-Based Estimation R Example

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 (real) 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

- `ec`: electric central
- `ecc`: electric central with AC
- `er`: electric room
- `erc`: electric room with AC
- `gc`: gas central
- `gcc`: gas central with AC
- `hpc`: heat pump with AC

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

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

Dataset

```
## Look at dataset
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

```
## Pivot into a long dataset
hvac_long <- HC %>%
  mutate(id = 1:n()) %>%
  pivot_longer(c(starts_with('ich.'), starts_with('och.')),
               names_to = c('cost', 'alt'), names_sep = '[.]',
               values_to = 'value') %>%
  pivot_wider(names_from = cost, values_from = value) %>%
  mutate(choice = (depvar == alt)) %>%
  select(-depvar)
```

Dataset in a Long Format

```
## Look at long dataset
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  27.3  2.95     20     1 gcc    9.7   2.26 FALSE
##  2  27.3  2.95     20     1 ecc    7.86  4.09 FALSE
##  3  27.3  2.95     20     1 erc    8.79  3.85 TRUE
##  4  27.3  2.95     20     1 hpc   11.4   1.73 FALSE
##  5  27.3  2.95     20     1 gc    24.1   2.26 FALSE
##  6  27.3  2.95     20     1 ec    24.5   4.09 FALSE
##  7  27.3  2.95     20     1 er     7.37  3.85 FALSE
##  8  26.5  1.63     50     2 gcc    8.77  2.3  FALSE
##  9  26.5  1.63     50     2 ecc    8.69  2.69 FALSE
## 10  26.5  1.63     50     2 erc    7.09  3.45 FALSE
## # ... with 1,740 more rows
```

Clean Dataset

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

Cleaned Dataset

```
## Look at cleaned dataset
tibble(hvac_clean)
## # A tibble: 1,750 x 7
##       id alt   choice    ac    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
##  8     2 ec    FALSE      0 32.7   2.69     50
##  9     2 ecc   FALSE      1 35.2   4.32     50
## 10     2 er    FALSE      0  9.33  3.45     50
## # ... with 1,740 more rows
```

Random Utility Model of HVAC System Choice

We model the utility to household n of installing HVAC system j as

$$U_{nj} = V_{nj} + \varepsilon_{nj}$$

where V_{nj} depends on the data about alternative j and household n

What might affect the utility of the different HVAC systems?

- Installation cost
- Annual operating cost
- HVAC system technology
  - Does the system have air conditioning or not?
- Anything else?

What if the effects of these attributes on utility vary throughout the population?

- We can use a mixed logit model with random coefficients

Representative Utility of HVAC System Choice

We model the representative utility to household n of installing HVAC system j as

$$V_{nj} = \beta_{1n} AC_j + \beta_{2n} IC_{nj} + \beta_{3n} OC_{nj}$$

where the random coefficients are normally distributed

$$\beta_{1n} \sim \mathcal{N}(\mu_1, \sigma_1^2)$$
$$\beta_{2n} \sim \mathcal{N}(\mu_2, \sigma_2^2)$$
$$\beta_{3n} \sim \mathcal{N}(\mu_3, \sigma_3^2)$$

We will estimate the six parameters that define the distributions of these random coefficients

$$\boldsymbol{\theta} = \{\mu_1,\ \sigma_1^2,\ \mu_2,\ \sigma_2^2,\ \mu_3,\ \sigma_3^2\}$$

Choice Probabilities of HVAC System Choice

The mixed logit choice probabilities for this model are

$$P_{ni} = \int L_{ni}(\beta) f(\beta \mid \theta) d\beta$$

where $L_{ni}(\beta)$ is the logit probability at a given set of coefficients, β

$$L_{ni}(\beta) = \frac{e^{\beta_1 AC_i + \beta_2 IC_{ni} + \beta_3 OC_{nj}}}{\sum_{j=1}^{J} e^{\beta_1 AC_j + \beta_2 IC_{nj} + \beta_3 OC_{nj}}}$$

These choice probabilities do not have a closed-form expression, so we will simulate them

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

and estimate the parameters of the model using maximum simulated likelihood

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, $\boldsymbol{\theta}^0$
   2. Simulate choice probabilities for the current set of parameters, $\check{P}_{ni}(\boldsymbol{\theta}^s)$
      1. Transform each set of K standard normals using $\boldsymbol{\theta}^s$ 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 $\check{P}_{ni}(\boldsymbol{\theta}^s)$
   3. Use these simulated choice probabilities to calculate simulated log-likelihood, simulated moments, etc.
   4. Step to a better set of parameters, $\boldsymbol{\theta}^{s+1}$
   5. Repeat steps (2)–(4) until the algorithm converges to a set of parameters that is your simulation-based estimator

map() Function in R

We will use the map() and map2() 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

```
## 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

```
## Random draws without setting a seed
rnorm(5)
## [1] -1.50664274 -0.08802471 -0.61032925 -1.74517516 -0.03536937

rnorm(5)
## [1]  0.07561777 -1.60810834 -0.14475906 -1.24398939  1.29974032

## 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 household

```
## Draw standard normal random variables for each household
draws_hh <- map(1:250, ~ tibble(ac_draw = rnorm(100),
                                ic_draw = rnorm(100),
                                oc_draw = rnorm(100)))

draws_hh[[1]]
## # A tibble: 100 x 3
##    ac_draw ic_draw oc_draw
##      <dbl>   <dbl>   <dbl>
##  1  -1.31    0.107   0.945
##  2   0.865  -0.935   1.37
##  3  -1.25   -0.304   0.322
##  4   0.599   0.160  -0.268
##  5  -1.22   -1.09   -0.923
##  6  -0.231   0.105   1.27
##  7  -0.708   0.708   0.125
##  8   0.444  -1.55    2.05
##  9  -1.47   -0.467   0.525
## 10  -0.347   0.967  -0.202
## # ... with 90 more rows
```

Step 1.5: Organize Data

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

```
## Split data into list by household
data_hh <- hvac_clean %>%
  group_by(id) %>%
  group_split()
data_hh[[1]]
## # A tibble: 7 x 7
##      id alt   choice    ac    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 MSL Estimator

```
## 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 other arguments 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 ML, 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 MSL, 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
## Function to simulate choice probabilities for an individual household
sim_probs_ind <- function(params, draws_ind, data_ind){
  ## Select relevant variables and convert into a matrix [J x K]
  data_matrix <- data_ind %>%
    select(ac, ic, oc) %>%
    as.matrix()
  ## Transform random coefficients based on parameters [R x K]
  coef_matrix <- draws_ind %>%
    mutate(ac_coef = params[1] + params[4] * ac_draw,
           ic_coef = params[2] + params[5] * ic_draw,
           oc_coef = params[3] + params[6] * oc_draw) %>%
    select(ac_coef, ic_coef, oc_coef) %>%
    as.matrix()
  ## Calculate representative utility for each alternative in each draw [R x J]
  utility <- (coef_matrix %*% t(data_matrix)) %>%
    pmin(700) %>%
    pmax(-700)
  ## Sum the exponential of utility over alternatives [R x 1]
  prob_denom <- utility %>%
    exp() %>%
    rowSums()
  ## Calculate the conditional probability for each alternative and draw [R x J]
  cond_prob <- exp(utility) / prob_denom
  ## Calculate simulated choice probabilities as means over all draws [1 x J]
  sim_prob <- colMeans(cond_prob)
  ## Add simulated probability to initial dataset
  data_ind_out <- data_ind %>%
    mutate(prob = sim_prob)
  ## Return initial dataset with simulated probability variable
  return(data_ind_out)
}
```

Calculate Simulated Log-Likelihood

```
## Function to calculate simulated log-likelihood
sim_ll_fn <- function(params, draws_list, data_list){
  ## Simulate probabilities for each individual household
  data_sim_ind <- map2(.x = draws_list, .y = data_list,
                       .f = ~ sim_probs_ind(params = params,
                                            draws_ind = .x,
                                            data_ind = .y))
  ## Combine individual datasets into one
  data_sim <- data_sim_ind %>%
    bind_rows()
  ## Calculate log of simulated probability for the chosen alternative
  data_sim <- data_sim %>%
    filter(choice == TRUE) %>%
    mutate(log_prob = log(prob))
  ## Calculate the simulated log-likelihood
  sim_ll <- sum(data_sim$log_prob)
  ## Return the negative of simulated log-likelihood
  return(-sim_ll)
}
```

Maximize Simulated Log-Likelihood

```
## Maximize the log-likelihood function
model <- optim(par = c(6.53, -0.17, -1.04, 0, 0, 0), fn = sim_ll_fn,
               draws_list = draws_hh, data_list = data_hh,
               method = 'BFGS', hessian = TRUE,
               control = list(trace = 1, REPORT = 5))
## initial  value 330.051626
## iter   5 value 329.842440
## iter  10 value 329.559384
## iter  15 value 326.652634
## iter  20 value 325.961965
## iter  25 value 325.960937
## final  value 325.959722
## converged
```

MSL Optimization Results

```
## Report model results
model
## $par
## [1] 11.0917037025 -0.2164020694 -1.1278947803  4.5934515313  0.0002048493  0.0057092340
##
## $value
## [1] 325.9597
##
## $counts
## function gradient
##       94       28
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##             [,1]        [,2]        [,3]        [,4]        [,5]        [,6]
## [1,]  4.0451882   95.551233    3.536574  -4.1442779    3.285546   0.2552387
## [2,] 95.5512329 4130.546546 -215.794416 -92.6246226  122.374461   9.2682826
## [3,]  3.5365739 -215.794416  144.136869  -3.9213534   -5.159820   1.3656793
## [4,] -4.1442779  -92.624623   -3.921353   4.4151970   -7.345430  -0.2337583
## [5,]  3.2855456  122.374461   -5.159820  -7.3454298 2686.692261 -15.9361930
## [6,]  0.2552387    9.268283    1.365679  -0.2337583  -15.936193  42.8194368
```

MSL Parameters and Standard Errors

```
## Show MSL parameters
model$par
## [1] 11.0917037025 -0.2164020694 -1.1278947803  4.5934515313  0.0002048493  0.0057092340

## Calculate MSL standard errors
model_se <- model$hessian %>%
  solve() %>%
  diag() %>%
  sqrt()
model_se
## [1] 3.06288544 0.03011049 0.10472838 2.62472527 0.01974123 0.15308970

## Calculate parameter z-stats
model_zstat <- model$par / model_se
model_zstat
## [1]   3.62132503  -7.18693314 -10.76971448   1.75006946   0.01037672   0.03729339

## Calculate parameter p-values
model_pvalue <- 2 * pnorm(q = -abs(model_zstat))
model_pvalue
## [1] 2.930980e-04 6.626284e-13 4.784827e-27 8.010633e-02 9.917207e-01 9.702511e-01
```

Mixed Logit Elasticities

The public utility commission is considering a subsidy on the installation cost (`ic`) of heat pump (`hpc`) systems to incentivize households to switch to this most efficient HVAC system

- What is the elasticity of each HVAC system with respect to the installation cost of a heat pump?

The elasticities from the mixed logit model are

$$\text{Own: } E_{iz_{ni}} = \frac{z_{ni}}{P_{ni}} \int \beta_z L_{ni}(\beta)[1 - L_{ni}(\beta)] f(\beta \mid \boldsymbol{\theta}) d\beta$$

$$\text{Cross: } E_{iz_{nj}} = -\frac{z_{nj}}{P_{ni}} \int \beta_z L_{ni}(\beta) L_{nj}(\beta) f(\beta \mid \boldsymbol{\theta}) d\beta$$

These integrals do not have closed-form expressions, so we will have to simulate them

Steps to Simulate Mixed Logit Elasticities

$$\text{Own: } E_{iz_{ni}} = \frac{z_{ni}}{P_{ni}} \int \beta_z L_{ni}(\beta)[1 - L_{ni}(\beta)] f(\beta \mid \boldsymbol{\theta}) d\beta$$

$$\text{Cross: } E_{iz_{nj}} = -\frac{z_{nj}}{P_{ni}} \int \beta_z L_{ni}(\beta) L_{nj}(\beta) f(\beta \mid \boldsymbol{\theta}) d\beta$$

To simulate these elasticities at our MSL estimator, $\widehat{\boldsymbol{\theta}}$, for one household

1. Draw R sets of coefficients, β^r, from the density $f(\beta \mid \widehat{\boldsymbol{\theta}})$
   - Or use our existing standard normal draws and transform them using $\widehat{\boldsymbol{\theta}}$
2. Calculate the representative utility for every alternative for each draw
3. Calculate the conditional choice probability, $L_{ni}(\beta^r)$, for every alternative for each draw
4. Calculate the simulated choice probability, $\check{P}_{ni}$, for every alternative as the mean over all draws
5. Calculate the integrand for every alternative for each draw
6. Simulate the integral for every alternative
7. Calculate the elasticities using the simulated integrals

Simulate Elasticities for One Household

$$\text{Own: } E_{iz_{ni}} = \frac{z_{ni}}{P_{ni}} \int \beta_z L_{ni}(\beta)[1 - L_{ni}(\beta)] f(\beta \mid \theta) d\beta$$

$$\text{Cross: } E_{iz_{nj}} = -\frac{z_{nj}}{P_{ni}} \int \beta_z L_{ni}(\beta) L_{nj}(\beta) f(\beta \mid \theta) d\beta$$

```r
## Function to simulate elasticities for one household
sim_elas_ind <- function(params, draws_ind, data_ind){
  ## Select relevant variables and convert into a matrix [J x K]
  data_matrix <- data_ind %>%
    select(ac, ic, oc) %>%
    as.matrix()
  ## Transform random coefficients based on parameters [R x K]
  coef_matrix <- draws_ind %>%
    mutate(ac_coef = params[1] + params[4] * ac_draw,
           ic_coef = params[2] + params[5] * ic_draw,
           oc_coef = params[3] + params[6] * oc_draw) %>%
    select(ac_coef, ic_coef, oc_coef) %>%
    as.matrix()
  ## Calculate representative utility for each alternative in each draw [R x J]
  utility <- (coef_matrix %*% t(data_matrix)) %>%
    pmin(700) %>%
    pmax(-700)
  ## Sum the exponential of utility over alternatives [R x 1]
  prob_denom <- utility %>%
    exp() %>%
    rowSums()
  ## Calculate the conditional probability for each alternative and draw [R x J]
  cond_prob <- exp(utility) / prob_denom
  ## Calculate simulated choice probabilities as means over all draws [1 x J]
  sim_prob <- colMeans(cond_prob)
  ## Calculate simulated integral for own elasticity [1 x 1]
```

Simulate Elasticities for One Household

$$\text{Own: } E_{iz_{ni}} = \frac{z_{ni}}{P_{ni}} \int \beta_z L_{ni}(\beta)[1 - L_{ni}(\beta)] f(\beta \mid \boldsymbol{\theta}) d\beta$$

$$\text{Cross: } E_{iz_{nj}} = -\frac{z_{nj}}{P_{ni}} \int \beta_z L_{ni}(\beta) L_{nj}(\beta) f(\beta \mid \boldsymbol{\theta}) d\beta$$

```
  ## Calculate the conditional probability for each alternative and draw [R x J]
  cond_prob <- exp(utility) / prob_denom
  ## Calculate simulated choice probabilities as means over all draws [1 x J]
  sim_prob <- colMeans(cond_prob)
  ## Calculate simulated integral for own elasticity [1 x 1]
  sim_int_own_elas <- mean(coef_matrix[, 2] *
                             cond_prob[, 7] * (1 - cond_prob[, 7]))
  ## Calculate simulated integral for cross elasticities [1 x (J - 1)]
  sim_int_cross_elas <- colMeans(coef_matrix[, 2] *
                                   cond_prob[, 7] * cond_prob[, -7])
  ## Combine elasticity simulated integrals into one vector [1 x J]
  sim_int_elas <- c(sim_int_cross_elas, sim_int_own_elas)
  ## Calculate cross-price and own-price simulated elasticities [1 x J]
  sim_elas <- c(rep(-1, 6), 1) * data_ind$ic[7] / sim_prob * sim_int_elas
  ## Add simulated elasticities to initial dataset
  data_ind_out <- data_ind %>%
    mutate(elasticity = sim_elas)
  ## Return initial dataset with simulated elasticity variable
  return(data_ind_out)
}
```

Simulated Elasticities

```
## Simulate elasticities for each household
data_ind <- map2(.x = draws_hh, .y = data_hh,
                 .f = ~ sim_elas_ind(params = model$par,
                                     draws_ind = .x,
                                     data_ind = .y))
## Combine list of data into one tibble
data <- data_ind %>%
  bind_rows()
## Calculate average elasticity with respect to ic of hpc
data %>%
  group_by(alt) %>%
  summarize(elasticity = mean(elasticity), .groups = 'drop')
## # A tibble: 7 x 2
##   alt   elasticity
##   <chr>      <dbl>
## 1 ec          1.30
## 2 ecc         3.66
## 3 er          1.30
## 4 erc         3.66
## 5 gc          1.30
## 6 gcc         3.66
## 7 hpc        -4.12
```