Problem Set 4

Topics in Advanced Econometrics (ResEcon 703)
University of Massachusetts Amherst

Due: November 19, 11:30 am ET

Rules

Email a single .pdf file of your problem set writeup, code, and output to `mwoerman@umass.edu` by the date and time above. You may work in groups of up to three and submit one writeup for the group, and I strongly encourage you to do so. This problem set requires you to code your own estimators, rather than using R's "canned" routines (e.g., `glm()` and `mlogit()`), except where indicated in problem 2.

Data

Download the file `camping_dataset.zip` from the course website. This zipped file contains the dataset `camping.csv`, which you will use for this problem set. This dataset contains simulated data on the state park choice of 1000 visitors who camped at one of five Massachusetts State Parks. See the file `camping_description.txt` for a description of the variables in the dataset.

```
### Load packages for problem set
library(tidyverse)

## - Attaching packages ----------------- tidyverse 1.3.1 -
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
## - Conflicts -------------------- tidyverse_conflicts() -
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
## Load dataset
data_camping <- read_csv('camping.csv')

## Rows:  5000 Columns:  8
## - Column specification -------------------------
## Delimiter:  ","
## chr (1):  park
## dbl (7):  camper_id, park_id, visit, mountain, beach, cost, time
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Problem 1: Simulation-Based Estimation

We are again estimating a mixed logit model of state park choice among campers—as in problem 2 of problem set 3—but we are now coding our own estimator to better understand the simulation-based estimation method.

a. Model the camping park choice as a mixed logit model. Express the representative utility of each alternative as a linear function of its cost, time, and setting—mountain or beach—with a fixed coefficient on cost and random coefficients on time and mountain. That is, the representative utility to camper n from park j is

$$V_{nj} = \beta_1 C_{nj} + \beta_{2n} T_{nj} + \beta_{3n} M_j$$

where C_{nj} is the cost to camper n of traveling to and camping at park j, T_{nj} is the time for camper n to travel to park j, M_j is a binary indicator if park j is in the mountains. Model β_1 as a fixed coefficient and β_2 and β_3 as random with a normal distribution:

$$\beta_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$$
$$\beta_3 \sim \mathcal{N}(\mu_3, \sigma_3^2)$$

Estimate the parameters of this model by maximum simulated likelihood estimation; use 100 draws for your simulation and set a seed of 703 for replication. The following steps can provide a rough guide to creating your own maximum simulated likelihood estimator:

I. Set a seed of 703 for replication.

II. Draw 200,000 standard normal random variables (2 random coefficients $\times$ 100 draws $\times$ 1000 campers).

III. Create a function to simulate choice probabilities for one camper:
   i. The function should take a set of parameters, the random draws for one camper, and the data for one camper as inputs: `function(parameters, draws, data)`.
   ii. Transform the standard normal draws into the correct distributions using the distribution parameters.
   iii. Calculate the representative utility for every alternative for each draw.
   iv. Calculate the conditional choice probability for every alternative for each draw.
   v. Calculate the simulated choice probability for every alternative as the mean over all draws.

IV. Create a function to calculate simulated log-likelihood:
   i. The function should take a set of parameters, the random draws for all campers, and the data for all campers as inputs: `function(parameters, draws, data)`.
   ii. Simulate choice probabilities for every alternative for each camper—that is, call your previous function for each camper.
   iii. Sum the log of the simulated choice probability for each camper's chosen alternative.
   iv. Return the negative of the log of simulated likelihood.

V. Maximize the simulated log-likelihood (by minimizing its negative) using `optim()`. Your call of the `optim()` function may look something like:

```
optim(par = your_starting_guesses, fn = your_second_function,
      data = your_data, draws = your_draws,
      method = 'BFGS', hessian = TRUE)
```

A few additional notes on using the `optim()` function to estimate this mixed logit model:

- This estimation could potentially take hours to converge. To speed up convergence, we can start close to our expected parameter estimates. I recommend using starting guesses that are approximately equal to the parameters we estimated in problem 2(b) from problem set 3.
- To see the progress of the optimization algorithm and receive an update on the value of the objective function at every iteration, add the argument `control = list(trace = 1, REPORT = 1)` to your call of the `optim()` function.

Report the estimated parameters and standard errors from this model. Briefly interpret these results. For example, what does each parameter mean?

```
## Set seed for replication
set.seed(703)
## Draw standard normal random variables and split into list
draws_camper <- map(1:1000, ~ tibble(time_draw = rnorm(100),
                                     mountain_draw = rnorm(100)))
## Split data into list by camper
data_camper <- data_camping %>%
  group_by(camper_id) %>%
  group_split()
## Function to simulate choice probabilities for one individual
sim_probs_ind <- function(params, draws_ind, data_ind){
  ## Select relevant variables and convert into a matrix
  data_matrix <- data_ind %>%
    select(cost, time, mountain) %>%
    as.matrix()
  ## Transform random draws into coefficients based on parameters
  coef_matrix <- draws_ind %>%
    mutate(cost_coef = params[1],
           time_coef = params[2] + params[4] * time_draw,
           mountain_coef = params[3] + params[5] * mountain_draw) %>%
    select(cost_coef, time_coef, mountain_coef) %>%
    as.matrix()
  ## Calculate representative utility for each alternative in each draw
  utility <- (coef_matrix %*% t(data_matrix)) %>%
    pmin(700) %>%
    pmax(-700)
  ## Sum the exponential of utility over alternatives
  prob_denom <- utility %>%
    exp() %>%
    rowSums()
  ## Calculate the conditional probability for each alternative in each draw
  cond_prob <- exp(utility) / prob_denom
  ## Calculate simulated choice probabilities as the means over all draws
  sim_prob <- colMeans(cond_prob)
  ## Add simulated choice probability to initial dataset
  data_ind_out <- data_ind %>%
```

```
    mutate(prob = sim_prob)
  ## Return initial dataset with simulated probability variable
  return(data_ind_out)
}
## Function to calculate simulated log-likelihood
sim_ll_fn <- function(params, draws_list, data_list){
  ## Simulate probabilities for each individual
  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 the log of simulated probability for the chosen alternative
  data_sim <- data_sim %>%
    filter(visit == 1) %>%
    mutate(log_prob = log(prob))
  ## Calculate the simulated log-likelihood
  sim_ll <- sum(data_sim$log_prob)
  ## Return the negative of simulated log-likelihood
  return(-sim_ll)
}
## Maximize the log-likelihood function
model_1a <- optim(par = c(-0.02, -0.005, -0.8, 0.002, 5), fn = sim_ll_fn,
                  draws_list = draws_camper, data_list = data_camper,
                  method = 'BFGS', hessian = TRUE)
## Function to summarize MLE model results
summarize_mle <- function(model, names){
  ## Extract model parameter estimates
  parameters <- model$par
  ## Calculate parameters standard errors
  std_errors <- model$hessian %>%
    solve() %>%
    diag() %>%
    sqrt()
  ## Calculate parameter z-stats
  z_stats <- parameters / std_errors
  ## Calculate parameter p-values
  p_values <- 2 * pnorm(-abs(z_stats))
  ## Summarize results in a list
  model_summary <- tibble(names = names,
                          parameters = parameters,
                          std_errors = std_errors,
                          z_stats = z_stats,
                          p_values = p_values)
  ## Return model_summary object
```

```
  return(model_summary)
}
## Summarize model results
summarize_mle(model_1a,
              c('cost', 'time', 'mountain', 'sd.time', 'sd.mountain'))

## # A tibble: 5 x 5
##   names       parameters std_errors z_stats     p_values
##   <chr>            <dbl>      <dbl>   <dbl>        <dbl>
## 1 cost          -0.0215     0.00416   -5.16 0.000000244
## 2 time          -0.00620    0.00115   -5.40 0.0000000668
## 3 mountain      -0.879      0.322     -2.73 0.00628
## 4 sd.time        0.00375    0.00144    2.61 0.00894
## 5 sd.mountain    5.79       1.28       4.51 0.00000661
```

These five parameters are interpreted as they were in problem 2(b) of problem set 3, and the values are roughly comparable. The cost coefficient, β_1, is fixed at a value of -0.022, indicating this value is the marginal utility of cost for all campers. The time coefficient parameters, μ_2 and σ_2^2, indicate that the marginal utility of time traveling to camp is normally distributed with a mean of -0.0062 and a standard deviation of 0.0038. The mountain coefficient parameters, μ_3 and σ_3^2, indicate that the utility obtained by camping in the mountains (relative to camping at the beach), *ceteris paribus*, is normally distributed with a mean of -0.88 and a standard deviation of 5.8.

b. As in problem set 3, the Massachusetts Department of Conservation and Recreation (DCR) is considering an increase to the camping fee at Mount Greylock due to the cost of maintaining those camp sites, and they want to know how this change would affect park visitation patterns. Use the model and simulation draws in part (a) to simulate the elasticity of choosing each park with respect to the cost of camping at Mount Greylock (`park_id == 1`) for each camper; that is, 5 alternatives $\times$ 1000 campers $=$ 5000 elasticities. Then, for each park, calculate the mean of its elasticity with respect to the cost of camping at Mount Greylock. The following steps can provide a rough guide to simulating elasticities:

    I. Create a function to simulate elasticities for one camper:
        i. The function should take a set of parameters, the random draws for one camper, and the data for one camper as inputs: `function(parameters, draws, data)`.
        ii. Transform the standard normal draws into the correct distributions using the distribution parameters.
        iii. Calculate the representative utility for every alternative for each draw.
        iv. Calculate the conditional choice probability for every alternative for each draw.
        v. Calculate the simulated choice probability for every alternative as the mean over all draws.
        vi. Calculate the term inside the integral of the elasticity formula for every alternative for each draw by taking products of conditional choice probabilities and the cost coefficient.
        vii. Simulate the integral in the elasticity formula by taking the mean of the previous values over all draws for every alternative.
        viii. Calculate the elasticities by multiplying these simulated integrals by the cost of camping at Mount Greylock and dividing by the simulated choice probability of the respective alternative.

II. Simulate elasticities for each camper—that is, call this function for each camper—using your parameter estimates from part (a).

III. Average the simulated elasticity of each park over all campers.

Report these five mean elasticities. Briefly interpret these results.

```
## Function to simulate elasticities for one individual
sim_elas_ind <- function(params, draws_ind, data_ind){
  ## Select relevant variables and convert into a matrix
  data_matrix <- data_ind %>%
    select(cost, time, mountain) %>%
    as.matrix()
  ## Transform random draws into coefficients based on parameters
  coef_matrix <- draws_ind %>%
    mutate(cost_coef = params[1],
           time_coef = params[2] + params[4] * time_draw,
           mountain_coef = params[3] + params[5] * mountain_draw) %>%
    select(cost_coef, time_coef, mountain_coef) %>%
    as.matrix()
  ## Calculate representative utility for each alternative in each draw
  utility <- (coef_matrix %*% t(data_matrix)) %>%
    pmin(700) %>%
    pmax(-700)
  ## Sum the exponential of utility over alternatives
  prob_denom <- utility %>%
    exp() %>%
    rowSums()
  ## Calculate the conditional probability for each alternative in each draw
  cond_prob <- exp(utility) / prob_denom
  ## Calculate simulated choice probabilities as the means over all draws
  sim_prob <- colMeans(cond_prob)
  ## Calculate simulated integral for own elasticity
  sim_int_own_elas <- params[1] *
    mean(cond_prob[, 1] * (1 - cond_prob[, 1]))
  ## Calculate simulated integral for cross elasticities
  sim_int_cross_elas <- params[1] *
    colMeans(cond_prob[, 1] * cond_prob[, -1])
  ## Combine elasticity simulated integrals into one vector
  sim_int_elas <- c(sim_int_own_elas, sim_int_cross_elas)
  ## Calculate own-price and cross-price simulated elasticities
  sim_elas <- c(1, rep(-1, 4)) * data_ind$cost[1] / 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)
}
## Simulate elasticities for each individual
```

```
data_1b_ind <- map2(.x = draws_camper, .y = data_camper,
                    .f = ~ sim_elas_ind(params = model_1a$par,
                                        draws_ind = .x,
                                        data_ind = .y))
## Combine list of data into one tibble
data_1b <- data_1b_ind %>%
  bind_rows()
## Calculate average elasticity with respect to cost of alternative 1
data_1b %>%
  group_by(park_id, park) %>%
  summarize(elasticity = mean(elasticity), .groups = 'drop') %>%
  arrange(park_id)
```

```
## # A tibble: 5 x 3
##   park_id park             elasticity
##     <dbl> <chr>                 <dbl>
## 1       1 Mount Greylock      -0.655
## 2       2 October Mountain     0.602
## 3       3 Horseneck Beach      0.0893
## 4       4 Salisbury Beach      0.0896
## 5       5 Scusset Beach        0.0889
```

The mean elasticity of camping at Mount Greylock with respect to its cost is -0.655; the mean elasticity of camping at October Mountain with respect to the cost of camping at Mount Greylock is 0.602; and the mean elasticity of camping at any of beach parks with respect to the cost of camping at Mount Greylock is approximately 0.089, as reported above. This model implies that campers will substitute to October Mountain in much greater proportion than to the beach parks.

Problem 2: Individual-Level Coefficients

In problem set 3, we calculated the distribution of the dollar value that a camper places on each hour spent traveling and the distribution of the dollar value that a camper places on camping in the mountains (relative to camping at the beach). DCR is also interested in understanding these valuations for the campers who visit each of the five state parks in our dataset.

a. Use the model in problem 1(a) to calculate the mean coefficients for campers at each of the five parks. Use the same simulation draws as in problem 1 to simulate the mean coefficients for each camper, and then average over the campers who visit each park. The following steps can provide a rough guide to simulating mean coefficients:

    I. Create a function to simulate mean coefficients for one camper:
        i. The function should take a set of parameters, the random draws for one camper, and the data for one camper as inputs: `function(parameters, draws, data)`.
        ii. Transform the standard normal draws into the correct distributions using the distribution parameters.
        iii. Calculate the representative utility for every alternative for each draw.
        iv. Calculate the conditional choice probability of the chosen alternative for each draw.

v. Calculate weights for each simulation draw using the conditional choice probability of the chosen alternative.

vi. Calculate the weighted average for each coefficient using these simulation draw weights.

II. Simulate mean coefficients for each camper—that is, call this function for each camper—using your parameter estimates from problem 1(a).

III. For each of the five parks, average the simulated mean coefficients for all campers at that park.

```r
## Function to simulate individual coefficients for one individual
calc_mean_coefs <- function(params, draws_ind, data_ind){
  ## Select relevant variables and convert into a matrix
  data_matrix <- data_ind %>%
    select(cost, time, mountain) %>%
    as.matrix()
  ## Transform random draws into coefficients based on parameters
  coef <- draws_ind %>%
    mutate(cost_coef = params[1],
           time_coef = params[2] + params[4] * time_draw,
           mountain_coef = params[3] + params[5] * mountain_draw) %>%
    select(cost_coef, time_coef, mountain_coef)
  ## Convert coefficients tibble to a matrix
  coef_matrix <- as.matrix(coef)
  ## Calculate representative utility for each alternative in each draw
  utility <- (coef_matrix %*% t(data_matrix)) %>%
    pmin(700) %>%
    pmax(-700)
  ## Sum the exponential of utility over alternatives
  prob_denom <- utility %>%
    exp() %>%
    rowSums()
  ## Calculate the conditional probability for each alternative in each draw
  cond_prob <- exp(utility) / prob_denom
  ## Calculate the numerator of the draw weights as probability of chosen alt
  weights_num <- c(cond_prob %*% data_ind$visit)
  ## Calculate the draw weights
  weights <- weights_num / sum(weights_num)
  ## Add draw weights to dataset of coefficients
  coef <- coef %>%
    mutate(weight = weights)
  ## Calculate weighted mean for each coefficient
  coef_means <- coef %>%
    summarize(cost_coef = sum(cost_coef * weight),
              time_coef_mean = sum(time_coef * weight),
              mountain_coef_mean = sum(mountain_coef * weight))
  ## Add individual coefficient means to initial dataset
  data_ind_out <- data_ind %>%
    bind_cols(coef_means)
  ## Return initial dataset with simulated probability variable
```

```
  return(data_ind_out)
}
## Calculate mean coefficients for each individual
data_2a_ind <- map2(.x = draws_camper, .y = data_camper,
                    .f = ~ calc_mean_coefs(params = model_1a$par,
                                           draws_ind = .x,
                                           data_ind = .y))
## Combine list of data into one tibble
data_2a <- data_2a_ind %>%
  bind_rows()
```

If you are not able to calculate these mean coefficients "by hand," you can instead use the "canned" routine. First, estimate the model in problem 1(a) using the `mlogit()` function; this model is identical to problem 2(b) from problem set 3. Then use the `fitted()` function with argument `type = 'parameters'` to calculate the mean coefficients for each camper.

```
## Load mlogit package
library(mlogit)

## Loading required package: dfidx

##
## Attaching package: 'dfidx'

## The following object is masked from 'package:stats':
##
##     filter

## Convert dataset to dfidx format
data_dfidx <- dfidx(data = data_camping, shape = 'long',
                    choice = 'visit', idx = c('camper_id', 'park_id'))
## Model camping park visit as a mixed logit with fixed cost coefficient
model_2a_alt <- mlogit(formula = visit ~ cost + time + mountain | 0 | 0,
                       data = data_dfidx,
                       rpar = c(time = 'n', mountain = 'n'),
                       R = 100, seed = 703)
## Calculate mean coefficients for each individual
model_2a_alt_params_ind <- fitted(model_2a_alt, type = 'parameters')
## Add mean coefficients to dataset
data_2a_alt <- data_camping %>%
  filter(visit == 1) %>%
  mutate(cost_coef = coef(model_2a_alt)[1],
         time_coef_mean = model_2a_alt_params_ind[, 1],
         mountain_coef_mean = model_2a_alt_params_ind[, 2])
```

i. Report the mean coefficients for each park; that is, 3 variables $\times$ 5 alternatives $=$ 15 coefficients.

```
## Calculate average coefficients for each camping park
coef_park_2a <- data_2a %>%
  filter(visit == 1) %>%
  group_by(park_id, park) %>%
  summarize(cost_coef = mean(cost_coef),
            time_coef_mean = mean(time_coef_mean),
            mountain_coef_mean = mean(mountain_coef_mean),
            .groups = 'drop')
coef_park_2a
```

```
## # A tibble: 5 x 5
##   park_id park             cost_coef time_coef_mean mountain_coef_mean
##     <dbl> <chr>                <dbl>          <dbl>              <dbl>
## 1       1 Mount Greylock     -0.0215       -0.00608               3.56
## 2       2 October Mountain   -0.0215       -0.00634               3.51
## 3       3 Horseneck Beach    -0.0215       -0.00620              -4.99
## 4       4 Salisbury Beach    -0.0215       -0.00626              -4.85
## 5       5 Scusset Beach      -0.0215       -0.00616              -4.88
```

The cost coefficient, β_1, is fixed and, hence, it is the same for campers at all parks. The time coefficient, β_2, is random, but it is approximately equal on average for campers at all parks. The mountain coefficient, β_3, however, varies substantially across parks. This coefficient is positive on average at each of the mountain parks and negative on average at each of the beach parks.

```
## Calculate average coefficients for each camping park
coef_park_2a_alt <- data_2a_alt %>%
  group_by(park_id, park) %>%
  summarize(cost_coef = mean(cost_coef),
            time_coef_mean = mean(time_coef_mean),
            mountain_coef_mean = mean(mountain_coef_mean),
            .groups = 'drop')
coef_park_2a_alt
```

```
## # A tibble: 5 x 5
##   park_id park             cost_coef time_coef_mean mountain_coef_mean
##     <dbl> <chr>                <dbl>          <dbl>              <dbl>
## 1       1 Mount Greylock     -0.0187       -0.00486               2.77
## 2       2 October Mountain   -0.0187       -0.00493               2.71
## 3       3 Horseneck Beach    -0.0187       -0.00485              -3.99
## 4       4 Salisbury Beach    -0.0187       -0.00487              -3.95
## 5       5 Scusset Beach      -0.0187       -0.00487              -3.88
```

Using the "canned" routine, the patterns observed in the mean coefficients are the same.

ii. Calculate the mean dollar value that a camper at each park places on each hour spent traveling and the mean dollar value that a camper at each park places on camping in the mountains (relative to camping at the beach); that is, 2 variables $\times$ 5 alternatives $=$ 10 dollar values. Briefly interpret these results.

```
## Calculate mean values for each camping park
coef_park_2a %>%
  mutate(time_value = time_coef_mean / cost_coef * 60,
         mountain_value = mountain_coef_mean / -cost_coef) %>%
  select(park_id, park, time_value, mountain_value)

## # A tibble: 5 x 4
##   park_id park             time_value mountain_value
##     <dbl> <chr>                 <dbl>          <dbl>
## 1       1 Mount Greylock         17.0           166.
## 2       2 October Mountain       17.7           164.
## 3       3 Horseneck Beach        17.3          -232.
## 4       4 Salisbury Beach        17.5          -226.
## 5       5 Scusset Beach          17.2          -227.
```

The mean dollar value that a camper places on each hour spent traveling is approximately equal at all five parks. The mean dollar value that a camper at each park places on camping in the mountains (relative to camping at the beach) differs substantially when comparing the mountain parks and the beach parks. These results indicate that the campers who choose to camp in the mountains tend to place a positive value on camping in the mountains. Conversely, the campers who choose to camp at the beach tend to place a negative value on camping in the mountains, or tend to place a positive value on camping at the beach. In other words, the campers who choose a particular state park prefer the attributes of that park, which is intuitive.

```
## Calculate mean values for each camping park
coef_park_2a_alt %>%
  mutate(time_value = time_coef_mean / cost_coef * 60,
         mountain_value = mountain_coef_mean / -cost_coef) %>%
  select(park_id, park, time_value, mountain_value)

## # A tibble: 5 x 4
##   park_id park             time_value mountain_value
##     <dbl> <chr>                 <dbl>          <dbl>
## 1       1 Mount Greylock         15.6           148.
## 2       2 October Mountain       15.8           145.
## 3       3 Horseneck Beach        15.5          -213.
## 4       4 Salisbury Beach        15.6          -211.
## 5       5 Scusset Beach          15.6          -207.
```

Using the "canned" routine, the patterns observed in the mean dollar values are the same.