Problem Set 4

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

Solutions

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 all group members can submit the same code and output; indicate in your writeup who you worked with. You must submit a unique writeup that answers the problems below. You can discuss answers with your fellow group members, but your writeup must be in your own words. Problem 1 allows you to use R's "canned routines," while Problem 2 requires you to code your own estimators and write your own simulation code. Each problem will indicate which R function to use.

Data

Download the file ps4_dataset.zip from the course website (github.com/woerman/ResEcon703). This zipped file contains the dataset, phones.csv, that you will use for this problem set. The dataset contains simulated data from 1000 customers on the purchases or pre-orders of highly-anticipated phone models recently released by Apple and Google: iPhone 11 and Pixel 4. See the file data_descriptions.txt for descriptions of the variables in the dataset.

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

## - Attaching packages ------------------ tidyverse 1.2.1 -
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## - Conflicts ------------------- tidyverse_conflicts() -
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(mlogit)

## Loading required package:  Formula
## Loading required package:  zoo
##
## Attaching package:  'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
## Loading required package:  lmtest
```

Problem 1: Mixed Logit Model

a. We are again interested in understanding how consumers value two phone characteristics: internal storage and screen size. As we saw in the last problem set, consumers appear to have brand preferences, which we are also interested in valuing. Model the purchase of a phone as a mixed logit model. Include an Apple brand dummy, the amount of storage, the screen size, and the price of each phone as explanatory variables with random coefficients. That is, the representative utility for alternative j is

$$V_{nj} = \beta_0 Apple_j + \beta_1 GB_j + \beta_2 SS_j + \beta_3 p_j$$

where $Apple_j$ is a dummy variable if alternative j is from Apple, GB_j is the internal storage of alternative j, SS_j is the diagonal screen size of alternative j, and p_j is the price of alternative j. Model all four random coefficients as having a normal distribution. Estimate this mixed logit model using the `mlogit()` function in R; use 100 draws for simulation (`R = 100`) and set a seed of 703 for replication (`seed = 703`).

   i. Report your parameter estimates, standard errors, z-stats, and p-values. Briefly interpret these results.

   ii. For each coefficient, calculate the proportion of the population with negative coefficients and the proportion of the population with positive coefficients. Describe whether these coefficient distributions match economic intuition.

```r
### Part a
## Load dataset
phones <- read_csv('phones.csv')

## Parsed with column specification:
## cols(
##   customer_id = col_double(),
##   phone_id = col_double(),
##   purchase = col_double(),
##   phone = col_character(),
##   storage = col_double(),
##   screen = col_double(),
##   price = col_double()
## )

## Create dummy variable for Apple
data_1 <- phones %>%
  mutate(apple = 1 * (phone_id %in% 5:10))
## Convert dataset to mlogit format
```

```
data_1a <- data_1 %>%
  mlogit.data(shape = 'long', choice = 'purchase', alt.var = 'phone_id')

## Warning:  Setting row names on a tibble is deprecated.

## Model phone purchase as a mixed logit with normal coefficients
model_1a <- data_1a %>%
  mlogit(purchase ~ apple + storage + screen + price | 0 | 0, data = .,
         rpar = c(apple = 'n', storage = 'n', screen = 'n', price = 'n'),
         R = 100, seed = 703)
## Summarize model results
model_1a %>%
  summary()

##
## Call:
## mlogit(formula = purchase ~ apple + storage + screen + price |
##     0 | 0, data = ., rpar = c(apple = "n", storage = "n", screen = "n",
##     price = "n"), R = 100, seed = 703)
##
## Frequencies of alternatives:
##     1     2     3     4     5     6     7     8     9    10
## 0.073 0.008 0.078 0.005 0.208 0.355 0.009 0.081 0.014 0.169
##
## bfgs method
## 34 iterations, 0h:0m:39s
## g'(-H)^-1g = 2.33E-07
## gradient close to zero
##
## Coefficients :
##                Estimate  Std. Error  z-value  Pr(>|z|)
## apple      -0.27432806  0.18745159  -1.4635    0.1433
## storage     0.01575859  0.00110029  14.3222 < 2.2e-16 ***
## screen      3.32983065  0.37194887   8.9524 < 2.2e-16 ***
## price      -0.01364835  0.00099822 -13.6727 < 2.2e-16 ***
## sd.apple   -0.39293068  0.65101653  -0.6036    0.5461
## sd.storage  0.01030173  0.00120933   8.5185 < 2.2e-16 ***
## sd.screen   4.38053637  0.77092870   5.6822  1.33e-08 ***
## sd.price   -0.00102265  0.00130528  -0.7835    0.4334
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log-Likelihood: -1795.8
##
## random coefficients
##          Min.     1st Qu.      Median        Mean      3rd Qu. Max.
## apple    -Inf -0.539355773 -0.27432806 -0.27432806 -0.009300343  Inf
```

```
## storage -Inf  0.008810186  0.01575859  0.01575859  0.022707003  Inf
## screen  -Inf  0.375203765  3.32983065  3.32983065  6.284457530  Inf
## price   -Inf -0.014338119 -0.01364835 -0.01364835 -0.012958582  Inf

## Calculate fraction of population with negative coefficients
pnorm(0, model_1a$coefficients[1:4], abs(model_1a$coefficients[5:8]))

##      apple    storage     screen      price
## 0.75746109 0.06304496 0.22358479 1.00000000

## Calculate fraction of population with positive coefficients
1 - pnorm(0, model_1a$coefficients[1:4], abs(model_1a$coefficients[5:8]))

##     apple   storage    screen     price
## 0.2425389 0.9369550 0.7764152 0.0000000
```

Neither parameter for the β_0 distribution is significant, suggesting no brand preference in this model. Parameters for β_1 and β_2 are significant, indicating that preferences for internal storage and screen size vary throughout the population. The mean parameter for β_3 is significant but not the standard deviation, suggesting no variation in the marginal utility of income. The distributions of these coefficients is mostly intuitive. Coefficients β_0 and β_2 have substantial mass on both the positive and negative sides; some consumers prefer Apple phones and others prefer Google, and some consumers prefer a larger screen size and others prefer a more compact phone. Coefficient β_1 is positive for a large proportion of the population, but not for everyone, which is not intuitive because there is no clear reason to prefer less storage *ceteris paribus*. Coefficient β_3 is technically negative for some consumers, but only a negligible amount, which is intuitive because we generally think everyone has a positive marginal utility of income.

b. The mixed logit model of (a) is not be the best model for this setting if we think some coefficients should always be positive or some coefficient should always be negative. Again model this purchase as a mixed logit model with the same underlying utility model as in (a). That is, the representative utility for alternative j is

$$V_{nj} = \beta_0 Apple_j + \beta_1 GB_j + \beta_2 SS_j + \beta_3 p_j$$

where $Apple_j$ is a dummy variable if alternative j is from Apple, GB_j is the internal storage of alternative j, SS_j is the diagonal screen size of alternative j, and p_j is the price of alternative j. Model β_0 and β_2 as having a normal distribution and β_1 and β_3 as having a log-normal distribution. Estimate this mixed logit model using the `mlogit()` function in R; use 100 draws for simulation (`R = 100`) and set a seed of 703 for replication (`seed = 703`).

   i. Report your parameter estimates, standard errors, z-stats, and p-values. Briefly interpret these results.

   ii. For each coefficient, calculate the proportion of the population with negative coefficients and the proportion of the population with positive coefficients. Describe whether these coefficient distributions match economic intuition.

```r
### Part b
## Convert dataset to mlogit format with negative prices
data_1b <- data_1 %>%
  mlogit.data(shape = 'long', choice = 'purchase', alt.var = 'phone_id',
              opposite = 'price')

## Warning:  Setting row names on a tibble is deprecated.

## Model phone purchase as a mixed logit with random coefficients
model_1b <- data_1b %>%
  mlogit(purchase ~ apple + storage + screen + price | 0 | 0, data = .,
         rpar = c(apple = 'n', storage = 'ln', screen = 'n', price = 'ln'),
         R = 100, seed = 703)
## Summarize model results
model_1b %>%
  summary()

##
## Call:
## mlogit(formula = purchase ~ apple + storage + screen + price |
##     0 | 0, data = ., rpar = c(apple = "n", storage = "ln", screen = "n",
##     price = "ln"), R = 100, seed = 703)
##
## Frequencies of alternatives:
##     1     2     3     4     5     6     7     8     9    10
## 0.073 0.008 0.078 0.005 0.208 0.355 0.009 0.081 0.014 0.169
##
## bfgs method
## 26 iterations, 0h:0m:27s
## g'(-H)^-1g = 8.69E-08
## gradient close to zero
##
## Coefficients :
##             Estimate Std. Error  z-value  Pr(>|z|)
## apple      -0.180224   0.228987  -0.7870    0.4313
## storage    -4.224395   0.072971 -57.8916 < 2.2e-16 ***
## screen      3.482946   0.396939   8.7745 < 2.2e-16 ***
## price      -4.283510   0.075627 -56.6398 < 2.2e-16 ***
## sd.apple   -0.545821   0.686927  -0.7946    0.4269
## sd.storage  0.610595   0.061004  10.0091 < 2.2e-16 ***
## sd.screen   4.734797   0.787636   6.0114 1.839e-09 ***
## sd.price    0.069882   0.102294   0.6832    0.4945
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log-Likelihood: -1799.5
##
```

```
## random coefficients
##           Min.      1st Qu.      Median        Mean    3rd Qu. Max.
## apple   -Inf -0.548375347 -0.18022435 -0.18022435 0.18792664  Inf
## storage    0  0.009694127  0.01463419  0.01763303 0.02209167  Inf
## screen  -Inf  0.289374140  3.48294624  3.48294624 6.67651834  Inf
## price      0  0.013159054  0.01379415  0.01382788 0.01445991  Inf

## Calculate fraction of population with negative coefficients
c(pnorm(0, model_1b$coefficients[1], abs(model_1b$coefficients[5])),
  plnorm(0, model_1b$coefficients[2], abs(model_1b$coefficients[6])),
  pnorm(0, model_1b$coefficients[3], abs(model_1b$coefficients[7])),
  plnorm(0, model_1b$coefficients[4], abs(model_1b$coefficients[8]))) %>%
  setNames(c('apple', 'storage', 'screen', 'price'))

##     apple   storage    screen     price
## 0.6293715 0.0000000 0.2309852 0.0000000

## Calculate fraction of population with positive coefficients
1 - c(pnorm(0, model_1b$coefficients[1], abs(model_1b$coefficients[5])),
      plnorm(0, model_1b$coefficients[2], abs(model_1b$coefficients[6])),
      pnorm(0, model_1b$coefficients[3], abs(model_1b$coefficients[7])),
      plnorm(0, model_1b$coefficients[4], abs(model_1b$coefficients[8]))) %>%
  setNames(c('apple', 'storage', 'screen', 'price'))

##     apple   storage    screen     price
## 0.3706285 1.0000000 0.7690148 1.0000000
```

The interpretation of the parameter estimates for this model is roughly the same as in (a). We now model β_1 and β_3 as log-normal distributions, however, so any significant variation in these coefficients occurs only on the positive side. Coefficients β_0 and β_2 still have substantial mass on both the positive and negative sides, which has intuitive appeal for the reasons described above. Since we model β_1 and β_3 using log-normal distributions, these coefficients are now positive for everyone, which is more intuitive and corrects the issues described above. Technically, however, we can interpret coefficient β_3 as negative for the entire population because we transformed the price data to be negative, making the positive coefficients effectively negative coefficients.

c. We could use the model in (b) to calculate how consumers value the Apple brand, storage, and screen size, but it would require taking a ratio of distributions. To make these calculations easier, we can model price as having a fixed coefficient. Again model this purchase as a mixed logit model with the same underlying utility model as in (a) and (b). That is, the representative utility for alternative j is

$$V_{nj} = \beta_0 Apple_j + \beta_1 GB_j + \beta_2 SS_j + \beta_3 p_j$$

where $Apple_j$ is a dummy variable if alternative j is from Apple, GB_j is the internal storage of alternative j, SS_j is the diagonal screen size of alternative j, and p_j is the price of alternative j. Model β_0 and β_2 as having a normal distribution, β_1 as having a log-normal distribution, and β_3 as a fixed coefficient. Estimate this mixed logit model using the `mlogit()` function in R; use 100 draws for simulation (`R = 100`) and set a seed of 703 for replication (`seed = 703`).

i. Report your parameter estimates, standard errors, z-stats, and p-values. Briefly interpret these results.

ii. Calculate the value consumers place on the Apple brand, each gigabyte of internal storage, and each 0.1 inch of diagonal screen size. Because we have distributions for β_0, β_1, and β_2, these values will also be distributions. Report the parameters that define these distributions. That is, for the value of the Apple brand and the value of each 0.1 inch of diagonal screen size, report the mean and standard deviation of the value; for the value of each gigabyte of internal storage, report the mean and standard deviation of the underlying normal distribution from which the log-normal distribution is derived.

```r
### Part c
## Convert dataset to mlogit format
data_1c <- data_1 %>%
  mlogit.data(shape = 'long', choice = 'purchase', alt.var = 'phone_id')

## Warning:  Setting row names on a tibble is deprecated.

## Model phone purchase as a mixed logit with random coefficients but
## fixed price coefficient
model_1c <- data_1c %>%
  mlogit(purchase ~ apple + storage + screen + price | 0 | 0, data = .,
         rpar = c(apple = 'n', storage = 'ln', screen = 'n'),
         R = 100, seed = 703)
## Summarize model results
model_1c %>%
  summary()

##
## Call:
## mlogit(formula = purchase ~ apple + storage + screen + price |
##     0 | 0, data = ., rpar = c(apple = "n", storage = "ln", screen = "n"),
##     R = 100, seed = 703)
##
## Frequencies of alternatives:
##     1     2     3     4     5     6     7     8     9    10
## 0.073 0.008 0.078 0.005 0.208 0.355 0.009 0.081 0.014 0.169
##
## bfgs method
## 24 iterations, 0h:0m:20s
## g'(-H)^-1g = 8.38E-07
## gradient close to zero
##
## Coefficients :
##              Estimate Std. Error  z-value  Pr(>|z|)
## apple      -0.1566701  0.2380459  -0.6582    0.5104
## storage    -4.2061132  0.0747870 -56.2412 < 2.2e-16 ***
## screen      3.6675156  0.4142715   8.8529 < 2.2e-16 ***
```

```
## price      -0.0141172  0.0010967 -12.8727 < 2.2e-16 ***
## sd.apple    0.6787496  0.6674151   1.0170    0.3092
## sd.storage  0.6085495  0.0580973  10.4747 < 2.2e-16 ***
## sd.screen  -5.0153374  0.8154579  -6.1503 7.732e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log-Likelihood: -1801.5
##
## random coefficients
##         Min.      1st Qu.      Median        Mean    3rd Qu. Max.
## apple   -Inf -0.614479794 -0.15667012 -0.15667012 0.30113956  Inf
## storage    0  0.009886614  0.01490419  0.01793597 0.02246824  Inf
## screen  -Inf  0.284721895  3.66751558  3.66751558 7.05030927  Inf
```

```
## Calculate distirbuiton of the value of brand preference
c(model_1c$coefficients[1] / -model_1c$coefficients[4],
  abs(model_1c$coefficients[5]) / -model_1c$coefficients[4]) %>%
  setNames(c('apple', 'sd.apple'))
```

```
##     apple  sd.apple
## -11.09782  48.07962
```

```
## Calculate distirbuiton of the value of storage
c(model_1c$coefficients[2] - log(-model_1c$coefficients[4]),
  abs(model_1c$coefficients[6])) %>%
  setNames(c('storage', 'sd.storage'))
```

```
##    storage sd.storage
## 0.05424812 0.60854953
```

```
## Calculate distirbuiton of the value of screen size
c(model_1c$coefficients[3] / -model_1c$coefficients[4] * 0.1,
  abs(model_1c$coefficients[7]) / -model_1c$coefficients[4] * 0.1) %>%
  setNames(c('screen', 'sd.screen'))
```

```
##    screen sd.screen
##  25.97906  35.52643
```

The general interpretation of these parameter estimates is the same as in (b) because the standard deviation of β_3 was not previously significant and no other parameter estimates changed substantially. The value consumers place on an Apple phone, relative to a Google phone, is distributed normally with a mean of $-\$11$ and a standard deviation of $\$48$. The value consumers place on each gigabyte of internal storage is distributed log-normally such that its underlying normal distribution has a mean of $\$0.05$ and a standard deviation of $\$0.61$. The value consumers place on each 0.1 inch of diagonal screen size is distributed normally with a mean of $\$26$ and a standard deviation of $\$3.6$.

d. Conduct a likelihood ratio test to compare the models in (b) and (c). Write down the null hypothesis that you are testing and describe this hypothesis in words. Conduct this likelihood ratio test using the function `lrtest()` in R. Do you reject your null hypothesis? What is the p-value of the test?

```
### Part d
## Conduct likelihood ratio test of the models in parts b and c
lrtest(model_1c, model_1b)

## Likelihood ratio test
##
## Model 1: purchase ~ apple + storage + screen + price | 0 | 0
## Model 2: purchase ~ apple + storage + screen + price | 0 | 0
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   7 -1801.5
## 2   8 -1799.5  1 3.9614    0.04656 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The null hypothesis we are testing is

$$H_0\colon\ \sigma_3 = 0$$

where σ_3 is the standard deviation of the underlying normal distribution for coefficient β_3. That is, we are testing if β_3 is a fixed coefficient or a random coefficient. We reject this null hypothesis; the likelihood ratio test has a p-value of 0.047. Thus, we conclude that, even though this parameter was not significant in (b), allowing β_3 to be log-normally distributed statistically improves the fit of the model.

e. Using the model in (c), calculate the mean coefficients for purchasers of each of the four Google phones. Use the function `fitted(type = 'parameters')` in R to calculate mean coefficients for each consumer. Report these 12 mean coefficients (3 random coefficients $\times$ 4 Google phones). Describe and briefly interpret the patterns you observe in these coefficients.

```
### Part e
## Calculate individual-level coefficients
coefficients_1e <- model_1c %>%
  fitted(type = 'parameters')
## Calculate average coefficients for Google phone consumers
data_1 %>%
  filter(purchase == 1) %>%
  select(customer_id, phone, storage) %>%
  rename(storage_data = storage) %>%
  cbind(coefficients_1e) %>%
  group_by(phone, storage_data) %>%
  summarize(apple = mean(apple),
            storage = mean(storage),
            screen = mean(screen)) %>%
  ungroup() %>%
  slice(1:4) %>%
```

```r
  rename(apple_coef = apple,
         storage_coef = storage,
         screen_coef = screen,
         storage = storage_data)
```

```
## # A tibble: 4 x 5
##   phone               storage apple_coef storage_coef screen_coef
##   <chr>                 <dbl>      <dbl>        <dbl>       <dbl>
## 1 Google Pixel 4           64     -0.430       0.0112       -2.20
## 2 Google Pixel 4          128     -0.433       0.0132       -1.79
## 3 Google Pixel 4 XL        64     -0.465       0.0105        6.94
## 4 Google Pixel 4 XL       128     -0.490       0.0115        6.97
```

These mean coefficients for consumers of Google phones have intuitive appeal. Consumers who purchase Google phones have a preference for Google phones; that is, the coefficient on the Apple brand variable, β_0, is negative. Consumers who purchase phones with 64 GB of internal storage place a lower value on internal storage than the consumers who purchase phones with 128 GB of internal storage. Consumers who purchase the Google Pixel 4, which has a smaller screen size, place a negative value on screen size; consumers who purchase the larger Google Pixel 4 XL place a positive value on screen size.

Problem 2: Simulation-Based Estimation

a. Model the purchase of a phone as in (c) of problem 1. That is, the representative utility for alternative j is
$$V_{nj} = \beta_0 Apple_j + \beta_1 GB_j + \beta_2 SS_j + \beta_3 p_j$$
where $Apple_j$ is a dummy variable if alternative j is from Apple, GB_j is the internal storage of alternative j, SS_j is the diagonal screen size of alternative j, and p_j is the price of alternative j. Model β_0 and β_2 as having a normal distribution, β_1 as having a log-normal distribution, and β_3 as a fixed coefficient. 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 300,000 standard normal random variables (3 random coefficients $\times$ 100 draws $\times$ 1000 consumers).
   III. Create a function to simulate choice probabilities for one consumer:
      i. The function should take a set of parameters, the random draws for one consumer, and the data for one consumer 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 each alternative for each draw.
      iv. Calculate the conditional choice probability for each alternative for each draw.
      v. Calculate the simulated choice probability for each 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 consumers, and the data for all consumers as inputs: `function(parameters, draws, data)`.
ii. Simulate choice probabilities for each alternative for each consumer (call your previous function for each consumer).
iii. Sum the log of the simulated choice probability for each consumer's chosen alternative.
iv. Return the negative of the log of simulated likelihood.

V. Maximize the simulated log-likelihood (or minimize its negative) using `optim()`. Use the parameters from (c) in problem 1 as your starting guesses to speed up convergence. Your call of the `optim()` function may look something like:

```
optim(par = c(model_1c$coefficients), fn = your_second_function,
      data = your_data, draws = your_draws,
      method = 'BFGS', hessian = TRUE)
```

Report your parameter estimates, standard errors, z-stats, and p-values. Briefly interpret these results.

```r
### Part a
## Set seed for replication
set.seed(703)
## Draw standard normal random variables and split into list
draws_2_list <- 1:1000 %>%
  map(., ~ tibble(apple_coef = rnorm(100),
                  storage_coef = rnorm(100),
                  screen_coef = rnorm(100)))
## Split data into list by customer
data_2_list <- data_1 %>%
  group_by(customer_id) %>%
  group_split()
## Function to simulate choice probabilities for one individual
simulate_probabilities <- function(parameters, draws, data){
  ## Select relevant variables and convert into a matrix
  data_matrix <- data %>%
    select(apple, storage, screen, price) %>%
    as.matrix()
  ## Transform random coefficients based on parameters
  coefficients <- draws %>%
    mutate(apple_coef = parameters[1] + parameters[5] * apple_coef,
           storage_coef = exp(parameters[2] + parameters[6] * storage_coef),
           screen_coef = parameters[3] + parameters[7] * screen_coef,
           price_coef = parameters[4])
  ## Calculate utility for each alternative in each draw
  utility <- (as.matrix(coefficients) %*% t(data_matrix)) %>%
    pmin(700) %>%
    pmax(-700)
  ## Sum the exponential of utility over alternatives
  summed_utility <- utility %>%
```

```r
    exp() %>%
    rowSums()
  ## Calculate the conditional probability for each alternative in each draw
  conditional_probability <- exp(utility) / summed_utility
  ## Average conditional probabilities over all draws
  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)
}
## Function to calculate simulated log-likelihood
simulate_log_likelihood <- function(parameters, draws_list, data_list){
  ## Simulate probabilities for each individual
  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(purchase == 1) %>%
    mutate(log_probability = log(probability))
  ## Calculate the simulated log-likelihood
  simulated_log_likelihood <- sum(data$log_probability)
  ## Return the negative of simulated log-likelihood
  return(-simulated_log_likelihood)
}
## Maximize the log-likelihood function
model_2a <- optim(par = c(model_1c$coefficients), fn = simulate_log_likelihood,
                  draws_list = draws_2_list, data_list = data_2_list,
                  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
```

```r
  p_values <- 2 * pnorm(-abs(z_stats))
  ## Summarize results in a list
  model_summary <- list(names = names,
                        parameters = parameters,
                        std_errors = std_errors,
                        z_stats = z_stats,
                        p_values = p_values)
  ## Return model_summary object
  return(model_summary)
}
summarize_mle(model_2a, names(model_2a$par))
```

```
## $names
## [1] "apple"      "storage"    "screen"     "price"      "sd.apple"
## [6] "sd.storage" "sd.screen"
##
## $parameters
##        apple      storage       screen         price     sd.apple
##  17.72907047  -3.08390631  10.49272675  -0.04632928 -23.30666552
##   sd.storage    sd.screen
##   0.41970114 -31.48374229
##
## $std_errors
##       apple     storage      screen       price    sd.apple  sd.storage
## 2.602764067 0.051008739 1.071194680 0.001999222 2.730096682 0.032545145
##   sd.screen
## 0.510014775
##
## $z_stats
##      apple    storage     screen      price   sd.apple sd.storage  sd.screen
##   6.811632 -60.458392   9.795350 -23.173651  -8.536938  12.895968 -61.731040
##
## $p_values
##         apple       storage        screen         price      sd.apple
##  9.649783e-12  0.000000e+00  1.178885e-22 8.397354e-119  1.378257e-17
##    sd.storage     sd.screen
##  4.742727e-38  0.000000e+00
```

Coefficients β_0 and β_2 are normally distributed with positive means and large standard deviations, indicating high variability in these coefficients throughout the population; that is, the utility generated by the Apple brand and the marginal utility of screen size are positive on average but negative for many individuals. Coefficient β_1 is log-normally distributed, with the distribution defined by the parameters above, forcing everyone to have a positive marginal utility of internal storage. Coefficient β_3 is fixed and negative, indicating a positive marginal utility of income.

b. Using the model in (a), calculate the mean coefficients for purchasers of each of the four Google phones. Use the same simulation draws as in (a) to simulate the mean coefficients for each consumer, and then average over all purchasers for each Google phone. The following steps can provide a rough guide to simulating coefficients:

I. Create a function to simulate mean coefficients for one consumer:
   i. The function should take a set of parameters, the random draws for one consumer, and the data for one consumer 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 each alternative for each draw.
   iv. Calculate the conditional choice probability of the chosen alternative for each draw.
   v. Calculate the weighted average for each coefficient with weights equal to the conditional choice probability of the chosen alternative for that simulation draw.

II. Simulate mean coefficients for each consumer (call your previous function for each consumer).

III. For each of the four Google phones, average the simulated mean coefficients for all purchasers of that phone.

Report these 12 mean coefficients (3 random coefficients $\times$ 4 Google phones). Describe and briefly interpret the patterns you observe in these coefficients.

```
### Part b
## Function to simulate individual coefficients for one individual
simulate_coefficients <- function(parameters, draws, data){
  ## Select relevant variables and convert into a matrix
  data_matrix <- data %>%
    select(apple, storage, screen, price) %>%
    as.matrix()
  ## Transform random coefficients based on parameters
  coefficients <- draws %>%
    mutate(apple_coef = parameters[1] + parameters[5] * apple_coef,
           storage_coef = exp(parameters[2] + parameters[6] * storage_coef),
           screen_coef = parameters[3] + parameters[7] * screen_coef,
           price_coef = parameters[4]) %>%
    select(apple_coef, storage_coef, screen_coef, price_coef)
  ## Calculate utility for each alternative in each draw
  utility <- (as.matrix(coefficients) %*% t(data_matrix)) %>%
    pmin(700) %>%
    pmax(-700)
  ## Sum the exponential of utility over alternatives
  summed_utility <- utility %>%
    exp() %>%
    rowSums()
  ## Calculate the conditional probability for each alternative in each draw
  conditional_probability <- exp(utility) / summed_utility
  ## Extract conditional probabilities of chosen alternative for each draw
  probability_draw <- conditional_probability %*% data$purchase
  ## Add draw probability to dataset of coefficients
  coefficients <- coefficients %>%
```

```
    mutate(probability = c(probability_draw))
  ## Calculate weighted average for each coefficient
  coefficients_weighted <- coefficients %>%
    summarize(apple_coef = sum(apple_coef * probability),
              storage_coef = sum(storage_coef * probability),
              screen_coef = sum(screen_coef * probability),
              probability = sum(probability)) %>%
    mutate(apple_coef = apple_coef / probability,
           storage_coef = storage_coef / probability,
           screen_coef = screen_coef / probability) %>%
    select(-probability)
  ## Add individual coefficients to initial dataset
  data_out <- data %>%
    mutate(apple_coef = coefficients_weighted$apple_coef,
           storage_coef = coefficients_weighted$storage_coef,
           screen_coef = coefficients_weighted$screen_coef)
  ## Return initial dataset with simulated probability variable
  return(data_out)
}
## Calculate individual coefficients for each individual
data_2b_list <- map2(.x = draws_2_list, .y = data_2_list,
                     .f = ~ simulate_coefficients(parameters = model_2a$par,
                                                  draws = .x,
                                                  data = .y))
## Combine list of data into one tibble
data_2b <- data_2b_list %>%
  bind_rows()
## Calculate average coefficients for Google phone consumers
data_2b %>%
  filter(purchase == 1) %>%
  group_by(phone, storage) %>%
  summarize(apple_coef = mean(apple_coef),
            storage_coef = mean(storage_coef),
            screen_coef = mean(screen_coef)) %>%
  ungroup() %>%
  slice(1:4)
```

```
## # A tibble: 4 x 5
##   phone             storage apple_coef storage_coef screen_coef
##   <chr>               <dbl>      <dbl>        <dbl>       <dbl>
## 1 Google Pixel 4         64      -12.6       0.0414       -23.1
## 2 Google Pixel 4        128      -14.7       0.0540       -23.9
## 3 Google Pixel 4 XL      64      -17.4       0.0399        33.8
## 4 Google Pixel 4 XL     128      -21.7       0.0583        33.3
```

These mean coefficients for consumers of Google phones differ from those calculated in part (e) of problem 1, but they again have intuitive appeal. Consumers who purchase Google phones have a preference for Google phones; that is, the coefficient on the Apple brand variable, β_0, is negative. Consumers who purchase phones with 64 GB of internal storage place a lower value on internal storage than the consumers who purchase phones with 128 GB of internal storage. Consumers who purchase the Google Pixel 4, which has a smaller screen size, place a negative value on screen size; consumers who purchase the larger Google Pixel 4 XL place a positive value on screen size.

c. As in the previous problem set, Apple is still interested in raising the price of the iPhone 11 with 256 GB. Using the model in (a), simulate the elasticity of each other phone with respect to the price of the iPhone 11 with 256 GB. The following steps can provide a rough guide to simulating elasticities:

   I. Create a function to simulate elasticities for one consumer:

      i. The function should take a set of parameters, the random draws for one consumer, and the data for one consumer 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 each alternative for each draw.
      iv. Calculate the conditional choice probability for each alternative for each draw.
      v. Calculate the simulated choice probability for each alternative as the mean over all draws.
      vi. Calculate the term inside the integral of the elasticity formula for each alternative for each draw by taking products of conditional choice probabilities and the price coefficient.
      vii. Simulate the integral in the elasticity formula by taking the mean of the previous values over all draws for each alternative.
      viii. Calculate the elasticities by multiplying these simulated integrals by the price of the iPhone 11 with 256 GB and dividing by the simulated choice probability of the respective alternative.

   II. Simulate elasticities for each consumer (call your previous function for each consumer).

   III. Average each simulated elasticity over all consumers.

   Report these 10 elasticities. Briefly interpret these results.

```r
### Part c
## Function to simulate elasticities for one individual
simulate_elasticities <- function(parameters, draws, data){
  ## Select relevant variables and convert into a matrix
  data_matrix <- data %>%
    select(apple, storage, screen, price) %>%
    as.matrix()
  ## Transform random coefficients based on parameters
  coefficients <- draws %>%
    mutate(apple_coef = parameters[1] + parameters[5] * apple_coef,
           storage_coef = exp(parameters[2] + parameters[6] * storage_coef),
           screen_coef = parameters[3] + parameters[7] * screen_coef,
           price_coef = parameters[4])
  ## Calculate utility for each alternative in each draw
  utility <- (as.matrix(coefficients) %*% t(data_matrix)) %>%
    pmin(700) %>%
    pmax(-700)
```

```r
  ## Sum the exponential of utility over alternatives
  summed_utility <- utility %>%
    exp() %>%
    rowSums()
  ## Calculate the conditional probability for each alternative in each draw
  conditional_probability <- exp(utility) / summed_utility
  ## Calculate simulated choice probabilities
  simulated_probability <- colMeans(conditional_probability)
  ## Calculate mean product of conditional probability for own elasticity
  own_elasticity_integral <- mean(conditional_probability[, 6] *
                                    (1 - conditional_probability[, 6])) *
    unname(model_2a$par[4])
  ## Calculate mean product of conditional probability for cross elasticities
  cross_elasticity_integral <- colMeans(conditional_probability[, 6] *
                                          conditional_probability[, -6]) *
    model_2a$par[4]
  ## Combine elasticity integrals into one vector
  elasticity_integral <- c(cross_elasticity_integral[1:5],
                           own_elasticity_integral,
                           cross_elasticity_integral[6:9])
  ## Calculate own-price and cross-price simulated elasticities
  simulated_elasticity <- c(rep(-1, 5), 1, rep(-1, 4)) * data$price[6] /
    simulated_probability * elasticity_integral
  ## Add simulated elasticities to initial dataset
  data_out <- data %>%
    mutate(elasticity = simulated_elasticity)
  ## Return initial dataset with simulated probability variable
  return(data_out)
}
## Simulate elasticities for each individual
data_2c_list <- map2(.x = draws_2_list, .y = data_2_list,
                     .f = ~ simulate_elasticities(parameters = model_2a$par,
                                                  draws = .x,
                                                  data = .y))
## Combine list of data into one tibble
data_2c <- data_2c_list %>%
  bind_rows()
## Calculate average elasticity with respect to price of iPhone 11 with 256 GB
data_2c %>%
  group_by(phone, storage) %>%
  summarize(elasticity = mean(elasticity)) %>%
  ungroup()

## # A tibble: 10 x 3
##    phone             storage elasticity
##    <chr>               <dbl>      <dbl>
##  1 Google Pixel 4         64       1.10
```

```
##  2 Google Pixel 4        128       1.29
##  3 Google Pixel 4 XL      64       1.08
##  4 Google Pixel 4 XL     128       1.34
##  5 iPhone 11              64      12.4
##  6 iPhone 11             256      -9.95
##  7 iPhone 11 Pro          64       5.44
##  8 iPhone 11 Pro         512       4.91
##  9 iPhone 11 Pro Max      64       1.55
## 10 iPhone 11 Pro Max     512       3.02
```

The own-price elasticity of the iPhone 11 with 256 GB is nearly -10, indicating that consumers are highly price-responsive and many would substitute away from this phone if the price increases substantially. The cross-price elasticities of the Google phones with respect to the price of the iPhone 11 with 256 GB are all in the range of 1.08–1.34, suggesting few consumers would substitute across brands to the Google phones if Apple raises the price of the iPhone 11 with 256 GB. The largest cross-price elasticity is for the iPhone 11 with 64 GB, with an elasticity of 12.4, indicating many iPhone 11 consumers would substitute within model to less storage if Apple increases the price of the iPhone 11 with 256 GB. The remaining Apple phones have cross-price elasticities with respect to the price of the iPhone 11 with 256 GB of 1.55–5.44, larger than the Google phones but smaller than the iPhone 11 with 64 GB. These substitution patterns seem reasonable if consumers have strong brand and screen size preferences, as estimated by this model.