ECON 6120I Topics in Empirical Industrial Organization

Kohei Kawaguchi

Last updated: 2019-05-03

Contents

Chapter 1

Syllabus

1.1 Instructor Information

- Instructor:
  - Name: Kohei Kawaguchi.
  - Office: LSK6070, Monday 11:00-12:00.
- All questions related to the class have to be publicly asked on the discussion board of canvas rather than being privately asked in e-mail. The instructor usually does not reply in the evening, weekends, and holidays.

1.2 General Information

1.2.1 Class Time

- Date: Monday.
- Time: 13:30-17:20.
- Venue: CYTG001.

1.2.2 Description

- This is a PhD-level course for empirical industrial organization. This course covers various econometric methods used in industrial organization that is often referred to as the structural estimation approach. These methods have been gradually developed since 1980s in parallel with the modernization of industrial organization based on the game theory and now widely applied in antitrust policy, business strategy, and neighboring fields such as labor economics and international economics.
- This course presumes a good understanding of PhD-level microeconomics and microeconometrics. Participants are expected to understand at least UG-level industrial organization. This course requires participants to write programs mostly in R and sometimes in C++ to implement various econometric methods. In particular, all assignments will involve such a non-trivial programming task. Even though the understanding of these programming languages is not a prerequisite, a sharp learning curve will be required. Some experience in other programming languages will help. Audit without a credit is not admitted for students.

1.2.3 Expectation and Goals

- The goal of this course is to learn and practice econometric methods for empirical industrial organization. The lecture covers the econometric methods that have been developed between 80s and 00s to estimate primitive parameters governing imperfect competition among firms, such as production and cost function estimation, demand function estimation, merger simulation, entry and exit analysis, and dynamic decision models. The lecture also covers various new methods to recover model primitives in certain mechanisms such as auction, matching, network, and bargaining. The emphasis is put on the former group of methods, because they are the basis for other new methods. Participants will not only understand the theoretical background of the methods but also become able to implement these methods from scratches by writing their own programs. I will briefly discuss the latter class of new methods through reading recent papers. The participants will become able to understand and use these new methods.

1.3 Required Environment

- Participants should bring their laptop to the class. We have enough extension codes for students. The laptop should have sufficient RAM (at least $\geq$ 8GB, $\geq$ 16GB is recommended) and CPU power (at least Core i5 grade, Core i7 grade is recommended). Participants are fully responsible for their hardware issues. Operating System can be arbitrary. The instructor mainly uses OSX High Siera with iMac (Retina 5K, 27-inch, Late 2015) and Macbook Pro (Retina, 15-inch, Early 2017).
- Please install the following software by the first lecture. Technical issues related to the installment should be resolved by yourself, because it depends on your local environment. If you had an error, copy and paste the error message on a search engine, and find a solution. This solves 99.9% of the problems.
  - R: https://www.r-project.org/
  - RStudio: https://www.rstudio.com/
  - LaTeX:
    - MixTex https://miktex.org/
    - TeXLive https://www.tug.org/texlive/
    - MacTeX http://www.tug.org/mactex/

1.4 Evaluation

- Assignments (80): In total 8 homework are assigned. Each homework involves the implementation of the methods covered in the class. Each homework has 10 points. The working hour for each homework will be around 10-20 hours.
- Participation (10): Every time a participant asks a question in the class, after the class, during the office hour, or in the canvas. the participant gets one point, up to 10 points. The participant who asked the question writes the name, ID number, his/her question, and my answer in a discussion board on the course website to claim a point.
- Referee report (10): Toward the end of the semester, a paper in industrial organization is randomly assigned to each participant. Each participant writes a critical referee report of the assigned paper in A4 2 pages that consists of the summary, contribution, strong and weak points of the paper.
- Grading is based on the absolute scores: A+ with more than 80 points, A with more than 70 points, A- with more than 60 points, B+ with more than 50 points, B with more than 40 points, B- with more than 30 points and C otherwise.

1.5 Academic Integrity

Without academic integrity, there is no serious learning. Thus you are expected to hold the highest standard of academic integrity in the course. You are encouraged to study and do homework in groups. However, no cheating, plagiarism will be tolerated. Anyone caught cheating, plagiarism will fail the course. Please make sure adhere to the HKUST Academic Honor Code at all time (see http://www.ust.hk/vpaao/integrity/).

1.6 Schedule

- Introduction to structural estimation, R and RStudio
- Production function estimation I
- Production function estimation II
- Demand function estimation I
- Demand function estimation II
- Merger Analysis
- Entry and Exit
- Single-Agent Dynamics I
- Single-Agent Dynamics II, I change date due to my business trip
- Dynamic Game I
- Dynamic Game II
- Auction
- Other Mechanisms

1.7 Course Materials

1.7.1 R and RStudio

- Grolemund, G., 2014, Hands-On Programming with R, O'Reilly.
  - Free online version is available: https://rstudio-education.github.io/hopr/.
- Wickham, H., & Grolemund, G., 2017, R for Data Science, O'Reilly.
  - Free online version is available: https://r4ds.had.co.nz/.
- Boswell, D., & Foucher, T., 2011, The Art of Readable Code: Simple and Practical Techniques for Writing Better Code, O'Reilly.

1.7.2 Handbook Chapters

- Ackerberg, D., Benkard, C., Berry, S., & Pakes, A. (2007). "Econometric tools for analyzing market outcomes". Handbook of econometrics, 6, 4171-4276.
- Athey, S., & Haile, P. A. (2007). "Nonparametric approaches to auctions". Handbook of Econometrics, 6, 3847-3965.
- Berry, S., & Reiss, P. (2007). "Empirical models of entry and market structure". Handbook of Industrial Organization, 3, 1845-1886.
- Bresnahan, T. F. (1989). "Empirical studies of industries with market power". Handbook of Industrial Organization, 2, 1011-1057.
- Hendricks, K., & Porter, R. H. (2007). "An empirical perspective on auctions". Handbook of Industrial Organization, 3, 2073-2143.
- Matzkin, R. L. (2007). "Nonparametric identification". Handbook of Econometrics, 6, 5307-5368.
- Newey, W. K., & McFadden, D. (1994). "Large sample estimation and hypothesis testing". Handbook of Econometrics, 4, 2111-2245.

- Reiss, P. C., & Wolak, F. A. (2007). "Structural econometric modeling: Rationales and examples from industrial organization". Handbook of Econometrics, 6, 4277-4415.

1.7.3 Books

- Train, K. E. (2009). Discrete Choice Methods with Simulation, Cambridge university press.
- Davis, P., & Garces, E. (2010). Quantitative Techniques for Competition and Antitrust Analysis, Princeton University Press.
- Tirole, J. (1988). The Theory of Industrial Organization, The MIT Press.

1.7.4 Papers

- The list of important papers are occasionally given during the course.

Chapter 2

Introduction

2.1 Structural Estimation and Counterfactual Analysis

2.1.1 Example

- Igami [2017] "Estimating the Innovator's Dilemma: Structural Analysis of Creative Destruction in the Hard Disk Drive Industry, 1981-1998".
- **Question**:
- Does "Innovator's Dilemma" [Christensen, 1997] or the delay of innovation among incumbents exist?
- Christensen argued that old winners tend to lag behind entrants even when introducing a new technology is not too difficult, with a case study of the HDD industry.
  - Apple's smartphone vs. Nokia's feature phones.
  - Amazon vs. Borders.
  - Kodak's digital camera.
- If it exists, what is the reason for that?
- How do we empirically answer this question?
- **Hypotheses**:
- Identify potentially competing hypotheses to explain the phenomenon.
  1. Cannibalization: Because of cannibalization, the benefits of introducing a new product are smaller for incumbents than for entrants.
  2. Different costs: The incumbents may have higher costs for innovation due to the organizational inertia, but at the same time they may have some cost advantage due to accumulated R&D and better financial access.
  3. Preemption: The incumbents have additional incentive for innovation to preempt potential rivals.
  4. Institutional environment: The impacts of the three components differ across different institutional contexts such as the rules governing patents and market size.
- Casual empiricists pick up their favorite factors to make up a story.
- Serious empiricists should try to separate the contributions of each factor from data.
- To do so, the author develops an economic model that explicitly incorporates the above mentioned factors, while keeping the model parameters flexible enough to let the data tell the sign and size of the effects of each factor on innovation.

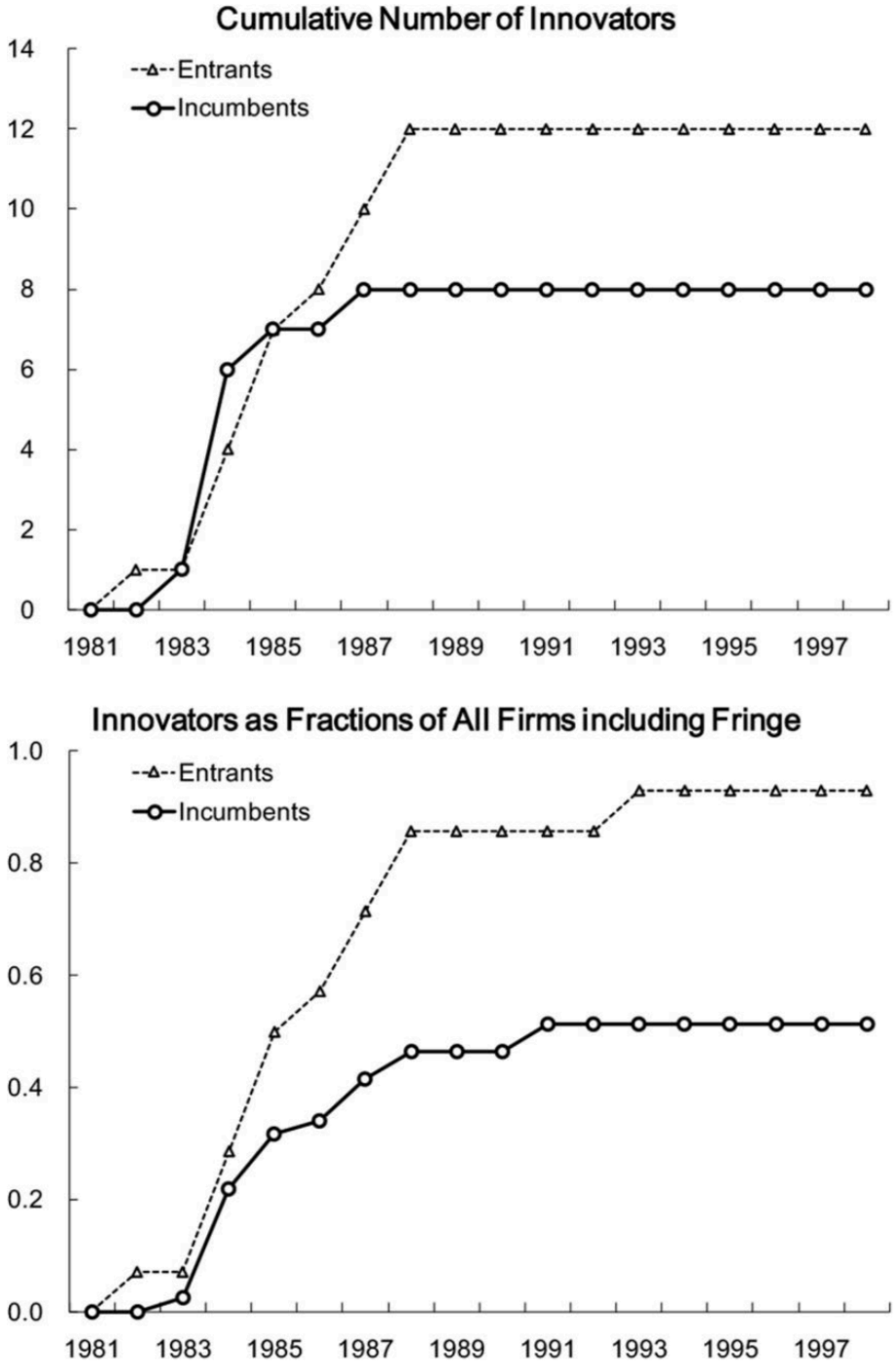


FIG. 1.—The incumbent-entrant innovation gap. The top panel plots the timing of the initial shipment of 3.5-inch HDDs separately for incumbents (i.e., firms already active in the 5.25-inch generation) and entrants (i.e., firms that appeared for the first time as producers of 3.5-inch HDDs). The bottom panel expresses similar numbers in terms of the fractions of all firms that could have innovated. See the text for details.

Figure 2.1: Figure 1 of Igam (2017)

- **Economic model**:
- The time is discrete with finite horizon $t = 1, \cdots, T$.
- In each year, there is a finite number of firms indexed by i.
- Each firm is in one of the technological states:

$$s_{it} \in \{\text{old only, both, new only, potential entrant}\}, \tag{2.1}$$

  where the first two states are for incumbents (stick to the old technology or start using the new technology) and the last two states are for actual and potential entrants (enter with the new technology or stay outside the market).

- In each year:
  - Pre-innovation incumbent (s_{it} = old): exit or innovate by paying a sunk cost κ^{inc} (to be $s_{i,t+1}$ = both).
  - Post-innovation incumbent (s_{it} = both): exit or stay to be both.
  - Potential entrant (s_{it} = potential entrant): give up entry or enter with the new technology by paying a sunk cost κ^{net} (to be $s_{i,t+1}$ = new).
  - Actual entrant (s_{it} = new): exit or stay to be new.
- Given the industry state $s_t = \{s_{it}\}_i$, the product market competition opens and the profit of firm i, $\pi_t(s_{it}, s_{-it})$, is realized for each active firm.
- As the product market competition closes:
  - Pre-innovation incumbents draw private cost shocks and make decisions: a_t^{pre}.
  - Observing this, post-innovation incumbents draw private cost shocks and make decisions: a_t^{post}.
  - Observing this, actual entrants draw private cost shocks and make decisions: a_t^{act}.
  - Observing this, potential entrants draw private cost shocks and make decisions: a_t^{pot}.
- This is a dynamic game. The equilibrium is defined by the concept of **Markov-perfect equilibrium** [Maskin and Tirole, 1988a].
- The representation of the competing theories in the model:
  - The existence of cannibalization is represented by the assumption that an incumbent maximizes the joint profits of old and new technology products.
  - The size of cannibalization is captured by the shape of profit function.
  - The difference in the cost of innovation is captured by the difference in the sunk costs of innovation.
  - The preemptive incentive for incumbents are embodied in the dynamic optimization problem for each incumbent.
- **Econometric model**:
- The author then turns the economic model into an econometric model.
- This amounts to specify which part of the economic model is observed/known and which part is unobserved/unknown.
- The author collects the data set of the HDD industry during 1977-99.
- Based on the data, the author specify the identities of active firms and their products and the technologies embodied in the products in each year to code their **state variables**.
- Moreover, by tracking the change in the state, the author code their **action variables**.
- Thus, the state and action variables, s_t and a_t. These are the **observables**.
- The author does not observe:

  - Profit function $\pi_t(\cdot)$.
  - Sunk cost of innovation for pre-innovation incumbents κ^{inc}.
  - Sunk cost of entry for potential entrants κ^{net}.
  - Private cost shocks.

- These are the **unobservables**.
- Among the unobservables, the profit function and sunk costs are the **parameter of interets** and the private cost shocks are **nuissance parameters** in the sense only the knowledge about the distribution of the latter is demanded.
- **Identification**:
- Can we infer the unobservables from the observables and the restrictions on the distribution of observable by the economic theory?
- The profit function is identified from estimating the demand function for each firm's product, and estimating the cost function for each firm from using their price setting behavior.
- The sunk costs of innovation are identified from the conditional probability of innovation across various states. If the cost is low, the probability should be high.
- **Estimation**:
- The identification established that in principle we can uncover the parameters of interests from observables under the restrictions of economic theory.
- Finally, we apply a statistical method to the econometric model and infer the parameters of interest.
- **Counterfactual analysis**:
- If we can uncover the parameters of interest, we can conduct **comparative statics**: study the change in the endogenous variables when the exogenous variables including the model parameters are set different. In the current framework, this exercise is often called the **counterfactual analysis**.
- What if there was no cannibalization?:
  - An incumbents separately maximizes the profit from old technology and new technology instead of jointly maximizing the profits. Solve the model under this new assumption everything else being equal.
  - Free of cannibalization concerns, 8.95 incumbents start producing new HDDs in the first 10 years, compared with 6.30 in the baseline.
  - The cumulative numbers of innovators among incumbents and entrants differ only by 2.8 compared with 6.45 in the baseline.
  - Thus cannibalization can explain a significant part of the incumbent-entrant innovation gap.
- What if there was no preemption?:
  - A potential entrant ignores the incumbents' innovations upon making entry decisions.
  - Without the preemptive motives, only 6.02 incumbents would innovate in the first 10 7ears, compared with 6.30 in the baseline.
  - The cumulative incumbent-entrant innovation gap widen to 8.91 compared with 6.45 in the baseline.
- The sunk cost of entry is smaller for incumbents than for entrants in the baseline.
- **Interpretations and policy/managerial implication**:
- Despite the cost advantage and the preemptive motives, the speed of innovation is slower among incumbents due to the strong cannibalization effect.
- Incumbents that attempt to avoid the "innovator's dilemma" should separate the decision makings between old and new sections inside the organization so that it can avoid the concern for cannibalization.

2.1.2 Recap

- The structural approach in empirical industrial organization consists of the following components:

1. Research question.
2. Competing hypotheses.
3. Economic model.
4. Econometric model
5. Identification.
6. Data collection.
7. Data cleaning.
8. Estimation.
9. Counterfactual analysis.
10. Coding.
11. Interpretations and policy/managerial implications.

- The goal of this course is to be familiar with the standard methodology to complete this process.
- The methodology covered in this class is mostly developed to analyze the standard framework to dynamic or oligopoly competition.
- The policy implications are centered around competition policies.
- But the basic idea can be extend to different class of situations such as auction, matching, voting, contract, marketing, and so on.
- Note that the depth of the research question and the relevance of the policy/managerial implications are the most important part of the research.
- Focusing on the methodology in this class is to minimize the time to allocate to less important issues and maximize the attention and time to the most valuable part in the future research.
- Given a research question, what kind of data is necessary to answer the question?
- Given data, what kind of research questions can you address? Which question can be credibly answered? Which question can be an over-stretch?
- Given a research question and data, what is the best way to answer the question? What type of problem can you avoid using the method? What is the limitation of your approach? How will you defend the possible referee comments?
- Given a result, what kinds of interpretation can you credibly derive? What kinds of interpretation can be contested by potential opponents? What kinds of contribution can you claim?
- To address these issues is **necessary** to publish a paper and it is **necessary** to be familiar with the methodology to do so.

2.1.3 Historical Remark

- The words **reduced-form** and **structural-form** date back to the literature of estimation of simultaneous equations in macroeconomics [Hsiao, 1983].
- Let y be the vector of observed endogenous variables, x be the vector of observed exogenous variables, and ϵ be the vector of unobserved exogenous variables.
- The equilibrium condition for y on x and ϵ is often written as:

$$Ay + Bx = \Sigma\epsilon. \tag{2.2}$$

- These equations **implicitly** determine the vector of endogenous variables y .
- If A is invertible, we can solve the equations for y to obtain:

$$y = -A^{-1}Bx + A^{-1}\Sigma\epsilon. \tag{2.3}$$

- These equations **explicitly** determine the vector of endogenous variables y.
- Equation (2.2) is the **structural-form** and (2.3) is the **reduced-form**.

- If y and x are observed and x is of full column rank, then $A^{-1}B$ and $A^{-1}\Sigma A^{-1}$ will be estimated by regression for (2.3). But this does not mean that A, B and Σ are separately estimated.
- This was the traditional identification problems.
- Thus, reduced-form does not mean either of:
  - Regression analysis;
  - Statistical analysis free from economic assumptions.
- Recent development in this line of literature of identification is found in Matzkin [2007].
- In econometrics, the idea of imposing restrictions from economic theories seems to have been formalized by the work of Manski [1994] and Matzkin [1994].

2.2 Setting Up The Environment

- Assume that R, RStudio and LaTex are all installed in the local computer.

2.2.1 RStudio Project

- The assignments should be conducted inside a project folder for this course.
- `File > New Project...> New Directory > New Directory > R Package using RcppEigen`.
- Name the directory `ECON6120I` and place in your favorite location.
- You can open this project from the upper right menu of RStudio or by double clicking the `ECON6120I.Rproj` file in the `ECON6120I` directory.
- This navigates you to the root directory of the project.
- In the root directory, make folders named:
  - `assignment`.
  - `input`.
  - `output`.
  - `figuretable`.
- We will store R functions in `R` folder, C/C++ functions in `src` folder, and data in `input` folder, data generated from the code in `output`, and figures and tables in `figurtable` folder.
- Open `src/Makevars` and erase the content. Then, write: `PKG_CPPFLAGS = -w -std=c++11 -O3`
- Open `src/Makevars.win` and erase the content. Then, write: `PKG_CPPFLAGS = -w -std=c++11`

2.2.2 Basic Programming in R

- `File > New File > R Script` to open `Untitled` file.
- `Ctrl (Cmd) + S` to save it with `test.R` in `assignment` folder.
- In the console, type and push enter:

```
1 + 1
```

```
## [1] 2
```

```
100:130
```

```
##  [1] 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116
## [18] 117 118 119 120 121 122 123 124 125 126 127 128 129 130
```

- This is the interactive way of using R functionalities.
- In `test.R`, write:

```
1 + 1
```

- Then, save the file and push `Run`.
- Alternatively, place the mouse over the `1 + 1` line in `test.R` file.

- Then, `Ctrl (Cmd) + Enter` to run the line.
- In this way, we can write procedures in the file and send to the console to run.
- There are functions to conduct basic calculations:

```
1 + 2
```

```
## [1] 3
```

```
2 * 3
```

```
## [1] 6
```

```
4 - 1
```

```
## [1] 3
```

```
6 / 2
```

```
## [1] 3
```

```
2^3
```

```
## [1] 8
```

- We can define objects and assign values to them.

```
a <- 1
a
```

```
## [1] 1
```

```
a + 2
```

```
## [1] 3
```

- In addition to scalar object, we can define a vector by:

```
2:10
```

```
## [1]  2  3  4  5  6  7  8  9 10
```

```
3:20
```

```
##  [1]  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
c(2, 3, 5, 9, 10)
```

```
## [1]  2  3  5  9 10
```

```
seq(1, 10, 2)
```

```
## [1] 1 3 5 7 9
```

- `seq` is a function with initial value, end values, and the increment value.
- By typing `seq` in the `help`, we can read the manual page of the function.
- `seq {base}` means that this function is named `seq` and is contained in the library called `base`.
- Some libraries are automatically called when the R is launched, but some are not.
- Some libraries are even not installed.
- We can install a library from a repository called `CRAN`.

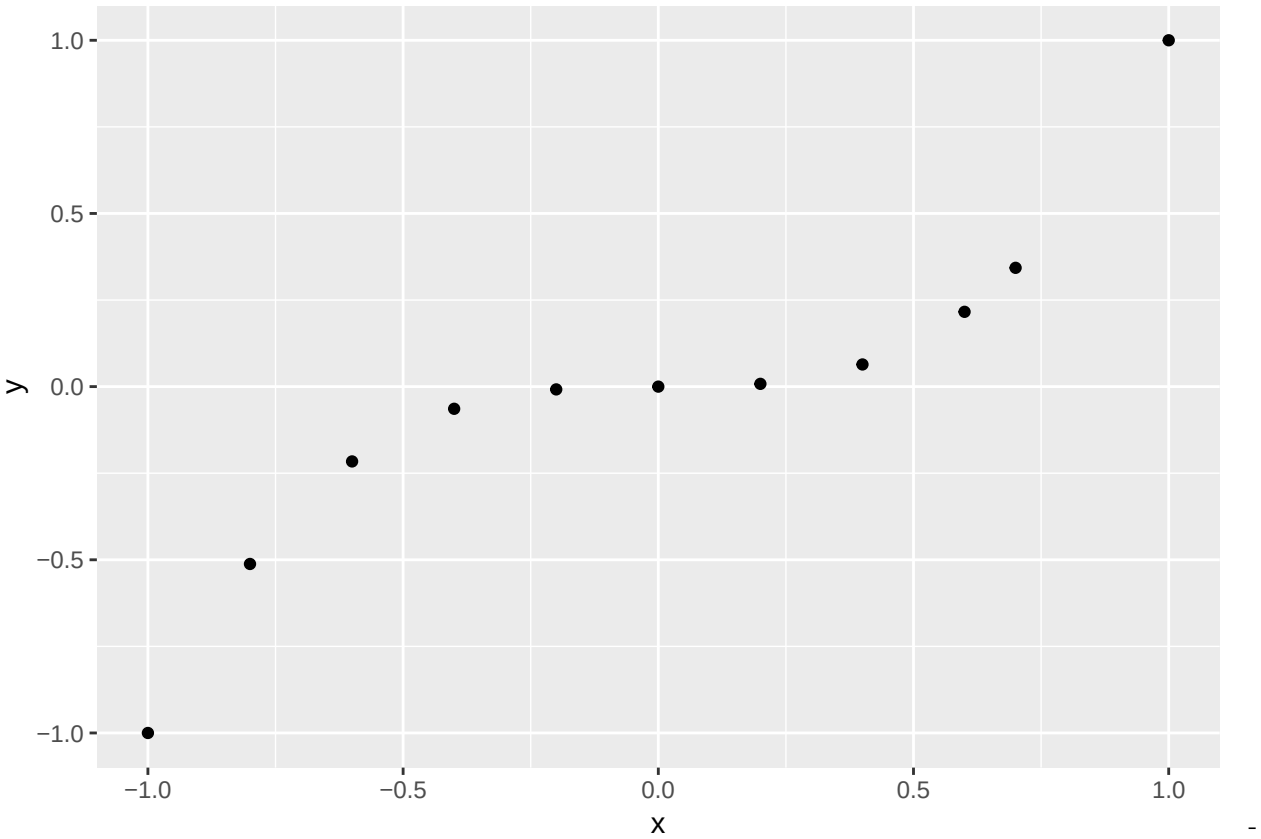
```
install.packages("ggplot2")
```

- To use the package, we have to load by:

```
library(ggplot2)
```

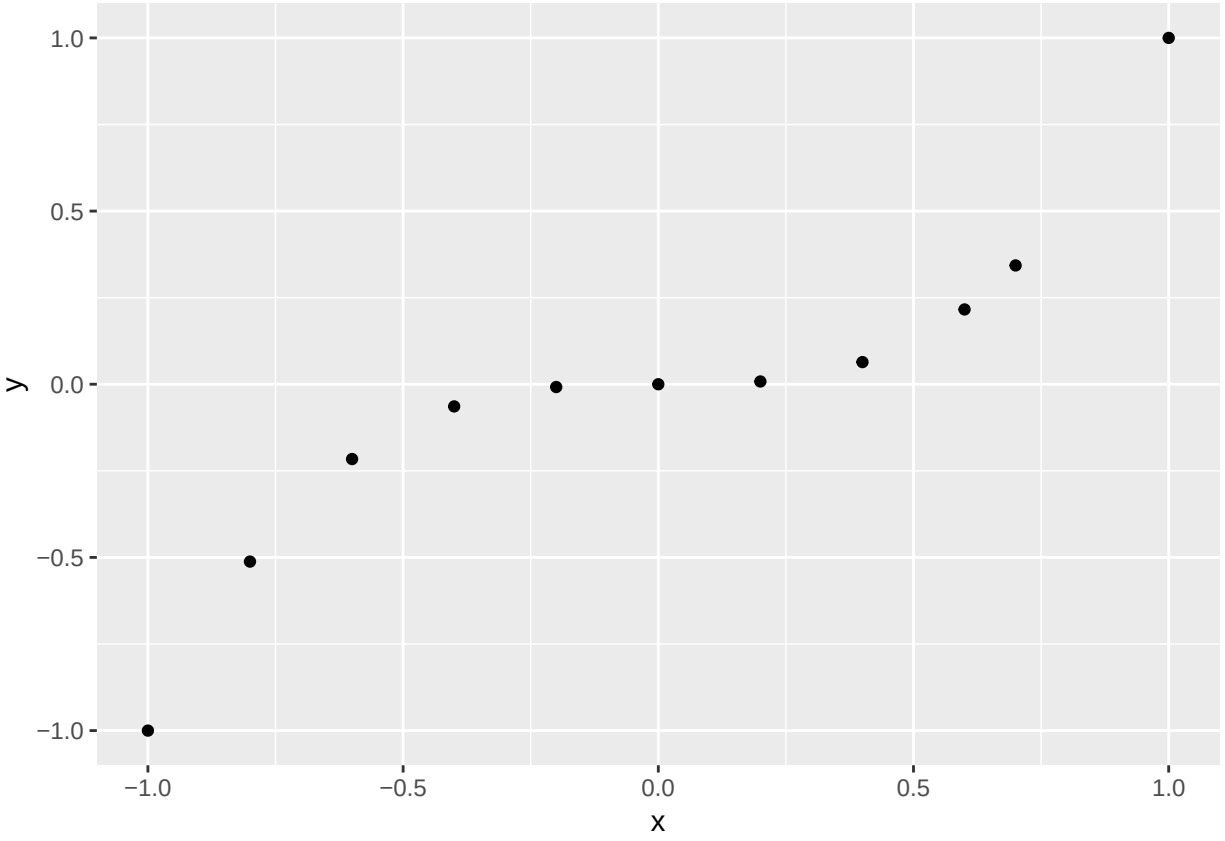
- Use `qplot` function in `ggplot2` library to draw a scatter plot.

```
x <- c(-1, -0.8, -0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6, 0.7, 1)
y <- x^3
qplot(x, y)
```



Instead of loading a package by `library`, you can directly call it as:

```
ggplot2::qplot(x, y)
```



- We can write own functions.

```
roll <- function(n) {
  die <- 1:6
  dice <- sample(die, size = n, replace = TRUE)
  y <- sum(dice)
  return(y)
}
roll(1)
```

```
## [1] 4
```

```
roll(2)
```

```
## [1] 5
```

```
roll(10)
```

```
## [1] 42
```

```
roll(10)
```

```
## [1] 39
```

- We can `set.seed` to obtain the same realization of random variables.

```
set.seed(1)
roll(10)
```

```
## [1] 38
```

```
set.seed(1)
roll(10)
```

```
## [1] 38
```

- When a variable used in a function is not given as its argument, the function calls the variable in the global environment:

```
y <- 1
plus_1 <- function(x) {
  return(x + y)
}
plus_1(1)
```

```
## [1] 2
```

```
plus_1(2)
```

```
## [1] 3
```

- However, you should NOT do this. All variables used in a function should be given as its arguments:

```
y <- 1
plus_2 <- function(x, y) {
  return(x + y)
}
plus_2(2, 3)
```

```
## [1] 5
```

```
plus_2(2, 4)
```

```
## [1] 6
```

- The best practice is to use `findGlobals` function in `codetools` to check global variables in a funciton:

```
library(codetools)
findGlobals(plus_1)
```

```
## [1] "{"      "+"      "return" "y"
```

```
findGlobals(plus_2)
```

```
## [1] "{"      "+"      "return"
```

- This function returns the list of global variables used in a function. If this returns a global variable other than the system global gariables, you should include it as the argument of the function.
- You can write the functions in the files with executing codes.
- But I recommend you to separate files for writing functions and executing codes.
- `File > New File > R Rcript` and name it as `functions.R` and save to `R` folder.
- Cut the function you wrote and paste it in `functions.R`.
- There are two ways of calling a function in `functions.R` from `test.R`.
- One way is to use `source` function.

```
source("R/functions.R")
```

- When this line is read, the codes in the file are executed.
- The other way is to bundle functions as a package and load it.

- Choose `Build > Clean and Rebuild`.
- This compiles files in `src` folder and bundle functions in `R` folder and build a package named `ECON6120I`.
- Now, the functions in `R` folder and `src` folder can be used by loading the package by:

```
library(ECON6120I)
```

- Best practice:
  1. Write functions in the scratch file.
  2. As the functions are tested, move them to `R/functions.R`.
  3. Clean and rebuild and load them as a package.

2.2.3 Reproducible Reports using Rmarkdown

- Reporting in empirical studies involves:

1. Writing texts;
2. Writing formulas;
3. Writing and implementing programs;
4. Demonstrating the results with figures and tables.

- Moreover, this has to be done in a **reproducible** manner: Whoever can reproduce the output from the scratch.
- "Whoever" includes yourself in the future. Because the revision process of structural papers is usually lengthy, you often have to remember the content few weeks or few months later. It is inefficient if you cannot recall what you have done.
- We use `Rmarkdown` to achieve this goal.
- This assumes that you have LaTex installed.
- Install package `Rmarkdown`:

```
install.packages("rmarkdown")
```

- `File > New File > R Markdown... > HTML` with title `Test`.
- Save it in `assignment` folder with name `test.Rmd`.
- From `Knit` tab, choose `Knit to HTML`.
- This outputs the content to html file.
- You can also choose `Knit to PDF` from `Knit` tab to obtain output in pdf file.
- Reports should be knit to pdf to submit.
- But you can use html output while writing a report because html is lighter to compile.
- Refer to the help page for further information.

Chapter 3

Production and Cost Function Estimation

3.1 Motivations

- Estimating **production and cost functions** of producers is the cornerstone of economic analysis.
- Estimating the functions includes to separate the contribution of observed inputs and the other factors, which is often referred to as the **productivity**.
- "What determines productivity?" [Syverson, 2011]-type research questions naturally follow.
- The methods covered in this chapter are widely used across different fields.
- Some of them are variants from the standard methods.

3.1.1 IO

- Olley and Pakes [1996]:
  - How much did the deregulation in the U.S. telecommunication industry, in particular the divestiture of AT&T in 1984, spurred the productivity growth of the incumbent, facilitated entries, and increased the aggregate productivity?
  - To do so, the authors estimate the plant-level production functions and productivity in the telecommunication industry.
- Doraszelski and Jaumandreu [2013]:
  - What is the role of R&D in determining the differences in productivity across firms and the evolution of firm-level productivity over time?
  - To do so, the authors estimate the firm-level production functions and productivity of Spanish manufacturing firms during 1990s in which the transition probability of a productivity is a function of the R&D activities.

3.1.2 Development

- Hsieh and Klenow [2009]:
  - How large is the misallocation of inputs across manufacturing firms in China and India compared to the U.S? How will the aggregate productivity of China and India change if the degree of misallocation is reduced to the U.S. level?
  - To do so, the authors measure the revenue productivity of firms, which should be the same across firms within an industry if there were no distortion, and the measurement of the revenue productivity requires to estimate the production function.

- Gennaioli et al. [2013]:
  - What are the determinants of regional growth? Do geographic, institutional, cultural, and human capital factors explain the difference across regions?
  - To do so, the authors construct the data set that covers 74% of the world's surface and 97% of its GDP and estimate the production function in which the above mentioned factors could affect the productivity.

3.1.3 Trade

- Haskel et al. [2007]:
  - Are there spillovers from FDI to domestic firms?
  - To do so, the authors estimate the plant-level production function of the U.K. manufacturing firms during 1973 and 1992 and study how the foreign presence in the U.K. affected the productivity.
- De Loecker [2011]:
  - Does the removal of trade barriers induces efficiency gain for producers?
  - To do so, the author estimate the production functions of Belgian textile industry during 1994-2002 in which the degree of trade protection can affect the productivity level.

3.1.4 Management

- Bloom and Van Reenen [2007]:
  - How do management practices affect the firm productivity?
  - To do so, the authors first estimate the production function and productivity of manufacturing firms in developed countries, and then study how the independently measured management practices of the firms affect the estimated productivity.
- Braguinsky et al. [2015]:
  - How do changes in ownership affect the productivity and profitability of firms?
  - To do so, the authors estimate the production function for various outputs including the physical output, return on capital and labor, and the utilization rate, price level, using the cotton spinners data in Japan during 1896 and 1920.

3.1.5 Education

- Cunha et al. [2010]:
  - How do childhood and schooling interventions "produce" the cognitive and non-cognitive skills of children?
  - To do so, the authors estimate the mapping from childhood and schooling interventions to children's cognitive and non-cognitive skills, the "production function" of childhood environment and education.

3.2 Analyzing Producer Behaviors

- There are several levels of parameters that govern the behavior of firms:
- **Production function**
  - Add factor market structure.
  - Add cost minimization.
- → **Cost function**
  - Add product market structure.

  - Add profit maximization.
- → **Supply function (Pricing function)**
  - Combine cost and supply (pricing) functions.
- → **Profit function**
- Which parameter to identify?
- Primitive enough to be invariant to relevant policy changes.
  - e.g. If you conduct a policy experiment that changes the factor market structure, identifying cost functions is not enough.
- As reduced-form as possible among such specifications.
  - A reduced-form parameter usually can be rationalized by a class of underlying structural parameters and institutional assumptions. Thus, the analysis becomes robust to some misspecifications.
  - e.g. A non-parametric function $C(q, w)$ can represent a cost function of a producer who is not necessarily minimizing the cost. If we derive a cost function from a production function and a factor market structure, then the cost function cannot represent such a non-optimization behavior.

3.3 Production Function Estimation

3.3.1 Cobb-Douglas Specification as a Benchmark

- Most of the following argument carries over to a general model.
- For firm $j = 1, \cdots, J$ and time $t = 1, \cdots, T$, we observe output Y_{jt}, labor L_{jt}, and capital K_{jt}.
- We consider an asymptotic of $J \to \infty$ for a fixed T.
- Assume Cobb-Douglas production function:

$$Y_{jt} = A_{jt} L_{jt}^{\beta_l} K_{jt}^{\beta_k}, \tag{3.1}$$

  where A_{jt} is firm j and time t specific unobserved heterogeneity in the model.
- Taking the logarithm gives:

$$y_{jt} = \beta_0 + \beta_l l_{jt} + \beta_k k_{jt} + \epsilon_{jt}, \tag{3.2}$$

  where lowercase symbols represent natural logs of variables and $\ln(A_{jt}) = \beta_0 + \epsilon_{jt}$.
- This can be regarded as a first-order log-linear approximation of a production function.
- Linear regression model! May OLS work?

3.3.2 Potential Bias I: Endogeneity

- ϵ_{jt} contains everything that cannot be explained by the observed inputs: better capital may be employed, a worker may have obtained better skills, etc.
- When the manager of a firm makes an input choice, she should have some information about the realization of ϵ_{jt}.
- Thus, the input choice can be correlated with ϵ_{jt}; for example under static optimization of L_{jt} given K_{jt}:

$$L_{jt} = \left[\frac{p_{jt}}{w_{jt}} \beta_l \exp^{\beta_0 + \epsilon_{jt}} K_{jt}^{\beta_k} \right]^{\frac{1}{1-\beta_l}}. \tag{3.3}$$

- In this case, OLS estimator for β_l is *positively* biased, because when ϵ_{jt} is high, l_{jt} is high and thus the increase in output caused by ϵ_{jt} is captured as if caused by the increase in labor input.

- The endogeneity problem was already recognized by Marschak and Andrews [1944].

3.3.3 Potential Bias II: Selection

- Firms freely enter and exit market.
- Therefore, a firm that had low ϵ_{jt} is likely to exit.
- However, if firms have high capital K_{jt}, it can stay in the market even if the realization of ϵ_{jt} is very low.
- Therefore, conditional on being in the market, there is a *negative* correlation between the capital K_{jt} and ϵ_{jt}.
- This problem occurs even if the choice of K_{jt} itself is not a function of ϵ_{jt}.

3.3.4 How to Resolve Endogeneity Bias?

- Temporarily abstract away from entry and exit.
- The data is balanced.

1. Panel data.
2. First-order condition for inputs.
3. Instrumental variable.
4. Olley-Pakes approach and its followers/critics.

- Griliches and Mairesse [1998] is a good survey of the history up to Olley-Pakes approach.
- Ackerberg et al. [2015] also offer a good survey and clarify problems and implicit assumptions in Olley-Pakes approach.

3.3.5 Panel Data

- Assume that $\epsilon_{jt} = \mu_j + \eta_{jt}$, where η_{jt} is uncorrelated with input choices up to period t:

$$y_{jt} = \beta_0 + \beta_l l_{jt} + \beta_k k_{jt} + \mu_j + \eta_{jt}. \tag{3.4}$$

- Then, by differentiating period t and $t-1$ equations, we get:

$$y_{jt} - y_{j,t-1} = \beta_l(l_{jt} - l_{j,t-1}) + \beta_k(k_{jt} - k_{j,t-1}) + (\eta_{jt} - \eta_{j,t-1}). \tag{3.5}$$

- Then, because $\eta_{jt} - \eta_{j,t-1}$ is uncorrelated either with $l_{jt} - l_{j,t-1}$ or $k_{jt} - k_{j,t-1}$, we can identify the parameter.
- Problem:
  - Restrictive heterogeneity.
  - When there are measurement errors, fixed-effect estimator can generate higher biases than OLS estimator, because measurement errors more likely to survive first-difference and within-transformation.

3.3.6 First-Order Condition for Inputs

- Use the first-order condition for inputs as the moment condition [McElroy, 1987].
- Closely related to the cost function estimation literature.
- Need to specify the factor market structure and the nature of the optimization problem for a firm.
- Recently being center of attention again as one of the solutions to the "collinearity problem" discussed below.

3.3.7 Instrumental Variable

- Borrow the idea from the first-order condition approach that the input choices are affected by some exogenous variables.
- If we have instrumental variables that affect inputs but are uncorrelated with errors ϵ_{jt}, then we can identify the parameter by an instrumental variable method.
- One candidate for the instrumental variables: **input prices**.
- Input price affect input decision.
- Input price is not correlated with ϵ_{jt} if the factor product market is competitive and ϵ_{jt} is an idiosyncratic shock to a firm.
- Problems:
  - Input prices often lack cross-sectional variation.
  - Cross-sectional variation is often due to unobserved input quality.
- Another candidate for the instrumental variables: **lagged inputs**.
- If ϵ_{jt} does not have auto-correlation, lagged inputs are not correlated with the current shock.
- If there are adjustment costs for inputs, then lagged inputs are correlated with the current inputs.
- Problem:
  - If ϵ_{jt} has auto-correlation, all lagged inputs are correlated with the errors: For example, if ϵ_{jt} is AR(1), $\epsilon_{jt} = \alpha\epsilon_{j,t-1} + \nu_{j,t-1} = \cdots \alpha^l \epsilon_{j,t-l} + \nu_{j,t-1} + \cdots, \alpha^{l-1}\nu_{j,t-l}$ for any l.

3.3.8 Olley-Pakes Approach

- Exploit restrictions from the economic theory [Olley and Pakes, 1996].
- Write $\epsilon_{jt} = \omega_{jt} + \eta_{jt}$, where ω_{jt} is an anticipated shock and η_{jt} is an ex-post shock.
- Inputs are correlated with ω_{jt} but not with η_{jt}
- The model is written as:

$$y_{jt} = \beta_0 + \beta_l l_{jt} + \beta_k k_{jt} + \omega_{jt} + \eta_{jt}. \tag{3.6}$$

- OP use economic theory to derive a valid proxy for the anticipated shock ω_{jt}.

3.3.9 Assumption I: Information Set

- The firm's information set at t, I_{jt}, includes current and past productivity shocks $\{\omega_{j\tau}\}_{\tau=0}^{t}$ but does not include future productivity shocks $\{\omega_{j\tau}\}_{\tau=t+1}^{\infty}$.
- The transitory shocks η_{jt} satisfy $\mathbb{E}\{\eta_{jt}|I_{jt}\} = 0$.

3.3.10 Assumption II: First Order Markov

- Productivity shocks evolve according to the distribution:

$$p(\omega_{j,t+1}|I_{jt}) = p(\omega_{j,t+1}|\omega_{jt}), \tag{3.7}$$

  and the distribution is known to firms and stochastically increasing in ω_{jt}.

- Then:

$$\omega_{jt} = \mathbb{E}\{\omega_{jt}|\omega_{j,t-1}\} + \nu_{jt}, \tag{3.8}$$

and:

$$\mathbb{E}\{\nu_{jt}|I_{j,t-1}\} = 0, \tag{3.9}$$

by construction.

3.3.11 Assumption III: Timing of Input Choices

- Firms accumulate capital according to:

$$k_{jt} = \kappa(k_{j,t-1}, i_{j,t-1}), \tag{3.10}$$

where investment $i_{j,t-1}$ is chosen in period $t-1$.
- Labor input l_{jt} is non-dynamic and chosen at t.
- This assumption characterizes and distinguishes labor and capital.
- Intuitively, it takes a full period for new capital to be ordered, delivered, and installed.

3.3.12 Assumption IV: Scalar Unobservable

- Firms' investment decisions are given by:

$$i_{jt} = f_t(k_{jt}, \omega_{jt}). \tag{3.11}$$

- This assumption places strong implicit restrictions on additional firm-specific unobservables.
  - No **across firm** unobserved heterogeneity in adjustment cost of capital, in demand and labor market conditions, or in other parts of the production function.
  - Okay with **across time** unobserved heterogeneity.

3.3.13 Assumption IV: Strict Monotonicity

- The investment policy function $f_t(k_{jt}, \omega_{jt})$ is strictly increasing in ω_{jt}.
- This holds if the realization of higher ω_{jt} implies higher expectation for future productivity (Assumption III) and if the marginal product of capital is increasing in the expectation for future productivity.
- To verify the latter condition in a given game is often not easy.

3.3.14 Two-step Approach: The First Step

- Insert $\omega_{jt} = h(k_{jt}, i_{jt})$ to the original equation to get:

$$\begin{aligned} y_{jt} &= \beta_l l_{jt} + \underbrace{\beta_0 + \beta_k k_{jt} + h(k_{jt}, i_{jt})}_{\text{unknown function of } k_{jt} \text{ and } i_{jt}} + \eta_{jt} \\ &\equiv \beta_l l_{jt} + \phi(k_{jt}, i_{jt}) + \eta_{jt}. \end{aligned} \tag{3.12}$$

- This is a **partially linear model**: see Ichimura and Todd [2007] for reference.

- Because l_{jt}, k_{jt} and i_{jt} are uncorrelated with η_{jt}, we can identify β_l and $\phi(\cdot)$ by exploiting the moment condition:

$$\begin{aligned} &\mathbb{E}\{\eta_{jt}|l_{jt}, k_{jt}, i_{jt}\} = 0 \\ &\Leftrightarrow \mathbb{E}\{y_{jt} - \beta_l l_{jt} - \phi(k_{jt}, i_{jt})|l_{jt}, k_{jt}, i_{jt}\} = 0. \end{aligned} \tag{3.13}$$

  if there is enough variation in l_{jt}, k_{jt} and i_{jt}.
- This "if there is enough variation" part is actually problematic. Discuss later.
- Let β_l^0 and ϕ^0 be the identified true parameters.

3.3.15 Two-step Approach: The Second Step

- Note that:

$$\omega_{jt} \equiv \phi(k_{jt}, i_{jt}) - \beta_0 - \beta_k k_{jt}. \tag{3.14}$$

- Therefore, we have:

$$\begin{aligned} &y_{jt} - \beta_l^0 l_{jt} \\ &= \beta_0 + \beta_k k_{jt} + \omega_{jt} + \eta_{jt} \\ &= \beta_0 + \beta_k k_{jt} + g(\omega_{j,t-1}) + \nu_{jt} + \eta_{jt} \\ &= \beta_0 + \beta_k k_{jt} + g[\phi^0(k_{j,t-1}, i_{j,t-1}) - (\beta_0 + \beta_k k_{j,t-1})] + \nu_{jt} + \eta_{jt}. \end{aligned} \tag{3.15}$$

- ν_{jt} and η_{jt} are independent of the covariates.
- This is a **multiple-index model** with indices $\beta_0 + \beta_1 k_{jt}$ and $\beta_0 + \beta_1 k_{j,t-1}$ where parameters of two indices are restricted to be the same: see Ichimura and Todd [2007] for reference.
- We can identify β_0, β_k and g by exploiting the moment condition:

$$\begin{aligned} &\mathbb{E}\{\nu_{jt} + \eta_{jt}|k_{jt}, k_{j,t-1}, i_{j,t-1}\} = 0 \\ &\Leftrightarrow \mathbb{E}\{\nu_{jt} + \eta_{jt}|k_{jt}, k_{j,t-1}, i_{j,t-1}\} = 0. \end{aligned} \tag{3.16}$$

3.3.16 Identification of the Anticipated Shocks

- If ϕ, β_0, β_k are identified, then ω_{jt} is also identified by:

$$\omega_{jt} \equiv \phi(k_{jt}, i_{jt}) - \beta_0 - \beta_k k_{jt}. \tag{3.17}$$

3.3.17 Two-Step Estimation of Olley and Pakes [1996].

- **First step**: Estimate β_L and ϕ in :

$$y_{jt} = \beta_l l_{jt} + \phi(k_{jt}, i_{jt}) + \eta_{jt}. \tag{3.18}$$

  by approximating ϕ with some basis functions, say, polynomials or splines:

$$\begin{aligned} y_{jt} &= \beta_l l_{jt} + \sum_{p=1}^{P} \gamma_p \phi_p(k_{jt}, i_{jt}) + \left[\phi(k_{jt}, i_{jt}) - \sum_{n=1}^{N} \gamma_n \phi_n(k_{jt}, i_{jt})\right] + \eta_{jt} \\ &= \beta_l l_{jt} + \sum_{p=1}^{P} \gamma_p \phi_p(k_{jt}, i_{jt}) + \tilde{\eta}_{jt} \end{aligned} \tag{3.19}$$

where $P \to \infty$ when the sample size goes to infinity.
- e.g. second-order polynomial approximation:

$$\begin{aligned} &\phi_1(k_{jt}, i_{jt}) = k_{jt}, \phi_2(k_{jt}, i_{jt}) = i_{jt} \\ &\phi_3(k_{jt}, i_{jt}) = k_{jt}^2, \phi_4(k_{jt}, i_{jt}) = i_{jt}^2 \\ &\phi_5(k_{jt}, i_{jt}) = k_{jt} i_{jt}. \end{aligned} \tag{3.20}$$

- Once the basis functions are fixed, estimation is the same as the linear model.
- But the inference (the computation of the standard deviation) is difference, because of the approximation error.
- See Chen [2007] for reference.
- Let $\hat{\beta}_l$ and $\hat{\phi}$ be the estimates from the first step.
- **Second step**: Estimate β_0, β_k, and g in:

$$\begin{aligned} y_{jt} - \hat{\beta}_l l_{jt} &= \beta_0 + \beta_k k_{jt} + g[\hat{\phi}(k_{j,t-1}, i_{j,t-1}) - (\beta_0 + \beta_k k_{j,t-1})] + \nu_{jt} + \eta_{jt} \\ &\quad + [\beta_l - \hat{\beta}_l] l_{jt} \\ &\quad + \Big\{ g[\phi(k_{j,t-1}, i_{j,t-1}) - (\beta_0 + \beta_k k_{j,t-1})] - g[\hat{\phi}(k_{j,t-1}, i_{j,t-1}) - (\beta_0 + \beta_k k_{j,t-1})] \Big\} \\ &= \beta_0 + \beta_k k_{jt} + g[\hat{\phi}(k_{j,t-1}, i_{j,t-1}) - (\beta_0 + \beta_k k_{j,t-1})] + \nu_{jt} + \tilde{\eta}_{jt} \end{aligned} \tag{3.21}$$

by approximating g by some basis functions, say, polynomials or splines.

3.3.18 From An Economic Models to An Econometric Model

- Starting from economic model with some unobserved heterogeneity, we reach some reduced-form model.
- If the resulting model belongs to a class of econometric models whose identification and estimation are established, we can simply apply the existing methods.

3.3.19 How to Resolve Selection Bias

- Use propensity score to correct selection bias: Ahn and Powell [1993].
- At the beginning of period t, after observing ω_{jt}, firm j decides whether to continue the business ($\chi_{jt} = 1$) or exit ($\chi_{jt} = 0$).
- Assume that the difference between continuation and exit values is strictly increasing in ω_{jt}.
- Then, there is a threshold $\underline{\omega}(k_{jt})$ such that:

$$\chi_{jt} = \begin{cases} 1 & \text{if } \omega_{jt} \geq \underline{\omega}(k_{jt}) \\ 0 & \text{otherwise.} \end{cases} \tag{3.22}$$

- We can only observe firms that satisfy $\chi_{jt} = 1$.

3.3.20 Correction in the First Step

- In the first step, we need no correction because:

$$\begin{aligned} &\mathbb{E}\{y_{jt} | l_{jt}, k_{jt}, i_{jt}, \chi_{jt} = 1\} \\ &= \beta_l l_{jt} + \phi(k_{jt}, i_{jt}) + \mathbb{E}\{\eta_{jt} | \chi_{jt} = 1\} \\ &= \beta_l l_{jt} + \phi(k_{jt}, i_{jt}). \end{aligned} \tag{3.23}$$

- Ex-post shock η_{jt} is independent of continuation/exit decision. Therefore, we can identify β_l and $\phi(\cdot)$ as in the previous case.

3.3.21 Correction in the Second Step I: The Source of Bias

- One the other hand, we need correction in the second step, because:

$$\begin{aligned}
&\mathbb{E}\{y_{jt} - \beta_l^0 l_{jt} | k_{jt}, i_{jt}, k_{j,t-1}, l_{j,t-1}, \chi_{jt} = 1\} \\
&= \beta_0 + \beta_k k_{jt} + g[\phi^0(k_{jt}, i_{jt}) - (\beta_0 + \beta_k k_{jt})] \\
&+ \mathbb{E}\{\nu_{jt} + \eta_{jt} | k_{jt}, i_{jt}, k_{j,t-1}, l_{j,t-1}, \chi_{jt} = 1\} \\
&= \beta_0 + \beta_k k_{jt} + g[\phi^0(k_{j,t-1}, i_{j,t-1}) - (\beta_0 + \beta_k k_{j,t-1})] \\
&+ \mathbb{E}\{\nu_{jt} | k_{jt}, i_{jt}, k_{j,t-1}, l_{j,t-1}, \chi_{jt} = 1\}.
\end{aligned} \tag{3.24}$$

and

$$\mathbb{E}\{\nu_{jt} | k_{jt}, i_{jt}, k_{j,t-1}, l_{j,t-1}, \chi_{jt} = 1\} \neq 0, \tag{3.25}$$

since anticipated shock matters continuation/exit decision in period t.

3.3.22 Correction in the Second Step II: Conditional Exit Probability

- Let's see that the conditional expectation:

$$\begin{aligned}
&\mathbb{E}\{\omega_{jt} | k_{jt}, i_{jt}, k_{j,t-1}, l_{j,t-1}, \chi_{jt} = 1\} \\
&= \mathbb{E}\{\omega_{jt} | k_{jt}, i_{jt}, k_{j,t-1}, l_{j,t-1}, \omega_{jt} \geq \underline{\omega}(k_{jt})\} \\
&= \int_{\underline{\omega}(k_{jt})} \omega_{jt} \frac{p(\omega_{jt} | \omega_{j,t-1})}{\int_{\underline{\omega}(k_{jt})} p(\omega | \omega_{j,t-1}) d\omega} d\omega_{jt} \\
&\equiv \tilde{g}(\omega_{j,t-1}, \underline{\omega}(k_{jt})),
\end{aligned} \tag{3.26}$$

is a function of $\omega_{j,t-1}$ and $\underline{\omega}(k_{jt})$.

3.3.23 Correction in the Second Step III: Invertibility in Threshold

- The propensity of continuation conditional on observed information up to period $t-1$:

$$\begin{aligned}
P_{jt} &\equiv \mathbb{P}\{\chi_{jt} = 1 | \mathcal{I}_{j,t-1}\} \\
&= \mathbb{P}\{\omega_{jt} \geq \underline{\omega}(k_{jt}) | \mathcal{I}_{j,t-1}\} \\
&= \mathbb{P}\{g(\omega_{j,t-1}) + \nu_{jt} \geq \underline{\omega}[(1-\delta)k_{j,t-1} + i_{j,t-1}] | \mathcal{I}_{j,t-1}\} \\
&= \mathbb{P}\{\chi_{jt} = 1 | i_{j,t-1}, k_{j,t-1}\}.
\end{aligned} \tag{3.27}$$

- $\rightarrow$ It suffices to condition on $i_{j,t-1}, k_{j,t-1}$.
- We also have:

$$P_{jt} = \mathbb{P}\{\chi_{jt} = 1 | \omega_{j,t-1}, \underline{\omega}(k_{jt})\}, \tag{3.28}$$

and it is invertible in $\underline{\omega}(k_{jt})$, that is,

$$\underline{\omega}(k_{jt}) \equiv \psi(P_{jt}, \omega_{j,t-1}). \tag{3.29}$$

3.3.24 Correction in the Second Step IV: Controlling the Threshold

- Now, he have:

$$\begin{aligned}
&\mathbb{E}\{y_{jt} - \beta_l^0 l_{jt} | k_{jt}, i_{jt}, k_{j,t-1}, l_{j,t-1}, \chi_{jt} = 1\} \\
&= \beta_0 + \beta_k k_{jt} + \mathbb{E}\{\omega_{jt} | k_{jt}, i_{jt}, k_{j,t-1}, l_{j,t-1}, \chi_{jt} = 1\} \\
&= \beta_0 + \beta_k k_{jt} + \tilde{g}(\omega_{j,t-1}, \underline{\omega}(k_{jt})) \\
&= \beta_0 + \beta_k k_{jt} + \tilde{g}(\omega_{j,t-1}, \psi(P_{jt}, \omega_{j,t-1})) \\
&\equiv \beta_0 + \beta_k k_{jt} + \tilde{\tilde{g}}(\omega_{j,t-1}, P_{jt}) \\
&= \beta_0 + \beta_k k_{jt} + \tilde{\tilde{g}}[\phi^0(k_{j,t-1}, i_{j,t-1}) - (\beta_0 + \beta_k k_{j,t-1}), P_{jt}].
\end{aligned} \tag{3.30}$$

- At the end, the only difference is to include P_{jt} as a covariate.
- P_{jt} is a **known** function of $i_{j,t-1}$ and $k_{j,t-1}$.
- Even if we condition on $P_{jt} = p$, there are still many combinations of $i_{j,t-1}$ and $k_{j,t-1}$ that gives $P_{jt} = p$.
- With this remaining variation, we can identify β_0, β_k, and $\tilde{\tilde{g}}$ by the same argument as the case without selection, for each $P_{jt} = p$.

3.3.25 Three Step Estimation of Olley and Pakes [1996]

- **Zero step**: Estimate the propensity score:

$$P_{jt} = 1\{\chi_{jt} = 1 | i_{j,t-1}, k_{j,t-1}\}, \tag{3.31}$$

  by a kernel estimator.
- Insert the resulting estimates $\widehat{P}_{jt}$ into the first and second steps.

3.3.26 Zero Investment Problem

- One of the key assumptions in OP method was invertibility between anticipated shock and investment:

$$\omega_{jt} = i^{-1}(k_{jt}, i_{jt}) \equiv h(k_{jt}, i_{jt}). \tag{3.32}$$

- However, in micro data, zero investment is a rule rather than exceptions.
- Then, the invertibility does not hold globally: there are some region of the anticipated shock in which the investment takes value zero.

3.3.27 Tackle Zero Investment Problem I: Discard Some Data

- Discard a data (j,t) such that $i_{j,t-1} = 0$.
- Use a data (j,t) such that $i_{j,t-1} > 0$.
- Then, invertibility recovers on this selected sample.
- This **does not** cause bias in the estimator because ν_{jt} in :

$$\beta_0 + \beta_l k_{jt} + g[\phi^0(k_{j,t-1}, i_{j,t-1}) - (\beta_0 + \beta_k k_{j,t-1})] + \nu_{jt} + \eta_{jt}, \tag{3.33}$$

  is independent of the event up to $t-1$, including $i_{j,t-1}$.
- However, this **does** cause information loss. The loss is high if the proportion of the sample such that $i_{j,t-1} = 0$ is high.

3.3.28 Tackle Zero Investment Problem II: Use Another Proxy

- Investment is just a possible proxy for the anticipated shock.
- Intermediate inputs can be used as proxies as well [Levinsohn and Petrin, 2003].
- The problem is that these intermediate inputs are included in the gross production function, whereas investment is excluded.
- Let m_{jt} be the log material input, and assume that the production function takes the form of:

$$y_{jt} = \beta_0 + \beta_l l_{jt} + \beta_k k_{jt} + \beta_m m_{jt} + \omega_{jt} + \eta_{jt}. \tag{3.34}$$

- In addition, assume that the **optimal policy function** for m_{jt} is strictly monotonic in the ex-ante shock, and hence is invertible:

$$m_{jt} = m(k_{jt}, \omega_{jt}) \Leftrightarrow \omega_{jt} = m^{-1}(m_{jt}, k_{jt}) \equiv h(m_{jt}, k_{jt}). \tag{3.35}$$

- **First step**:

$$\begin{aligned} y_{jt} &= \beta_0 + \beta_l l_{jt} + \beta_k k_{jt} + \beta_m m_{jt} + h(m_{jt}, k_{jt}) + \eta_{jt} \\ &= \beta_l l_{jt} + \phi(m_{jt}, k_{jt}) + \eta_{jt}. \end{aligned} \tag{3.36}$$

- We can identify β_l and ϕ by exploiting the moment condition:

$$\begin{aligned} &\mathbb{E}\{\eta_{jt} | l_{jt}, m_{jt}, k_{jt}, i_{jt}\} = 0 \\ &\Leftrightarrow \mathbb{E}\{y_{jt} - \beta_0 - \beta_l l_{jt} - \phi(m_{jt}, k_{jt}) | l_{jt}, m_{jt}, k_{jt}\} = 0, \end{aligned} \tag{3.37}$$

  if **there is enough variation** in l_{jt}, m_{jt}, k_{jt}.
- **Second step**:

$$\begin{aligned} &y_{jt} - \beta_l^0 l_{jt} \\ &= \beta_k k_{jt} + \beta_m m_{jt} + g[\phi^0(m_{j,t-1}, k_{j,t-1}) - \beta_k k_{j,t-1} - \beta_m m_{j,t-1}] \\ &+ \nu_{jt} + \eta_{jt}. \end{aligned} \tag{3.38}$$

- We can identify β_k, β_m, and g by exploiting the moment condition:

$$\mathbb{E}\{\nu_{jt} + \eta_{jt} | k_{jt}, m_{j,t-1}, k_{j,t-1}\} = 0. \tag{3.39}$$

- Because m_{jt} is correlated with ν_{jt}, the moment should not condition on m_{jt}.
- The identification of β_m comes from $\beta_m m_{j,t-1}$.

3.3.29 One-step Estimation of Olley and Pakes [1996] and Levinsohn and Petrin [2003]

- Levinsohn and Petrin [2003] can be estimated in the similar two-step method.
- We can jointly estimate the parameters in first and second steps to improve the efficiency [Wooldridge, 2009].
- We estimate under the assumptions of Olley and Pakes [1996]:

$$y_{jt} = \beta_0 + \beta_1 l_{jt} + \beta_k k_{jt} + \omega_{jt} + \eta_{jt}. \tag{3.40}$$

- The first step exploits the following moment:

$$\mathbb{E}\{\eta_{jt} | l_{jt}, k_{jt}, i_{jt}\} = 0, \tag{3.41}$$

that is:

$$\mathbb{E}\{y_{jt} - \beta_1 l_{jt} - \beta_0 - \beta_k k_{jt} - \omega(k_{jt}, i_{jt})|l_{jt}, k_{jt}, i_{jt}\} = 0. \tag{3.42}$$

- We can reinforce the moment condition as:

$$\mathbb{E}\{\eta_{jt}|l_{jt}, k_{jt}, i_{jt}, \cdots, l_{j1}, k_{j1}, i_{j1}\} = 0 \tag{3.43}$$

if we assume that lagged inputs are correlated with the current inputs and η_{jt} is independent.
- The second step exploits the following moment:

$$\mathbb{E}\{\nu_{jt}|k_{jt}, i_{j,t-1}, l_{j,t-1}\} = 0, \tag{3.44}$$

that is:

$$\mathbb{E}\{y_{jt} - \beta_0 - \beta_1 l_{jt} - \beta_k k_{jt} - g[\omega(k_{j,t-1}, i_{j,t-1})]|k_{jt}, i_{j,t-1}, l_{j,t-1}\} = 0. \tag{3.45}$$

- We can reinforce the moment condition as:

$$\mathbb{E}\{\nu_{jt}|k_{jt}, i_{j,t-1}, l_{j,t-1}, \cdots, k_{j1}, i_{j1}, l_{j1}\} = 0, \tag{3.46}$$

if we assume that lagged input are correlated with the current inputs and $\nu_{jt} + \eta_{jt}$ are independent.
- We can construct a GMM estimator based on equations (3.42) and (3.45).
- The one-step estimator can be more efficient but can be computationally heavier than the two-step estimator.

3.3.30 Scalar Unobservable Problem: Finite-order Markov Process

- Borrow the idea of using the first-order condition to resolve the collinearity problem [Gandhi et al., 2017].
- We have assumed that anticipated shocks follow a first-order Markov process:

$$\omega_{jt} = g(\omega_{j,t-1}) + \nu_{jt}. \tag{3.47}$$

- However, it may be true that it has more than one lags, for example:

$$\omega_{jt} = g(\omega_{j,t-1}, \omega_{j,t-2}) + \nu_{jt}. \tag{3.48}$$

- Then, we need proxies as many as the number of unobservables:

$$\begin{pmatrix} i_{jt} \\ m_{jt} \end{pmatrix} = \Gamma(k_{jt}, \omega_{jt}, \omega_{j,t-1}), \tag{3.49}$$

such that the policy function for the proxies is a bijection in $(\omega_{jt}, \omega_{j,t-1})$.
- Then, we can have:

$$\omega_{jt} = \Gamma_1^{-1}(k_{jt}, i_{jt}, m_{jt}). \tag{3.50}$$

- The reminder goes as in the standard OP method.

3.3.31 Scalar Unobservable Problem: Demand and Productivity Shocks

- There may be a demand shock μ_{jt} that also follows first-order Markov process.
- Then, the policy function depend both on μ_{jt} and ω_{jt}.
- We again need proxies as many as the number of unobservable.
- Suppose that we can observe the price of the firm p_{jt}.

- Inverting the policy function:

$$\begin{pmatrix} i_{jt} \\ p_{jt} \end{pmatrix} = \Gamma(k_{jt}, \omega_{jt}, \mu_{jt}). \tag{3.51}$$

yields:

$$\omega_{jt} = \Gamma_1^{-1}(k_{jt}, i_{jt}, p_{jt}). \tag{3.52}$$

- If ω_{jt} only depends on $\omega_{j,t-1}$ but not on $\mu_{j,t-1}$, then the second step of the modified OP method is to estimate:

$$\begin{aligned} y_{jt} - \hat{\beta}_l l_{jt} &= \beta_0 + \beta_k k_{jt} \\ &\quad + g(\omega_{j,t-1}) + \nu_{jt} + \eta_{jt} \\ &= \beta_0 + \beta_k k_{jt} \\ &\quad + g(\hat{\phi}_{j,t-1} - \beta_0 - \beta_k k_{j,t-1}) + \nu_{jt} + \eta_{jt}. \end{aligned} \tag{3.53}$$

- It goes as in the standard OP method.
- If ω_{jt} depends both on $\omega_{j,t-1}$ and $\mu_{j,t-1}$, the second step regression equation will be:

$$\begin{aligned} y_{jt} - \hat{\beta}_l l_{jt} &= \beta_0 + \beta_k k_{jt} \\ &\quad + g(\omega_{j,t-1}, \mu_{j,t-1}) + \nu_{jt} + \eta_{jt} \\ &= \beta_0 + \beta_k k_{jt} \\ &\quad + g(\hat{\phi}_{j,t-1} - \beta_0 - \beta_k k_{j,t-1}, \mu_{j,t-1}) + \nu_{jt} + \eta_{jt}. \end{aligned} \tag{3.54}$$

- We still have to control $\mu_{j,t-1}$ in the second step.
- Invert the policy function for $\mu_{j,t-1}$ to get:

$$\mu_{j,t-1} = \Gamma_2^{-1}(k_{j,t-1}, i_{j,t-1}, p_{j,t-1}), \tag{3.55}$$

and plug it into the second step regression equation to get:

$$\begin{aligned} &y_{jt} - \hat{\beta}_l l_{jt} \\ &= \beta_0 + \beta_k k_{jt} \\ &+ g(\hat{\phi}_{j,t-1} - \beta_0 - \beta_k k_{j,t-1}, \Gamma_2^{-1}(k_{j,t-1}, i_{j,t-1}, p_{j,t-1})) + \nu_{jt} + \eta_{jt}. \end{aligned} \tag{3.56}$$

- The parameters β_0 and β_k **cannot** be identified only with this observation, because Γ_2^{-1} is **unknown non-parametric** function: it can mean any function of $(k_{j,t-1}, i_{j,t-1}, p_{j,t-1})$.
- To estimate such a model, we jointly estimate the demand function along with the production function.
- At this point, we do not investigate it further because we have not yet learned how to estimate the demand function.
- For now just keep in mind that:
  - There has to be as many proxies as the dimension of the unobservable state variables.
  - It is okay that the unobservable state variable includes a demand shock.
  - It can be problematic when the unobservable demand shock affect the evolution of the anticipated productivity shock.

3.3.32 Collinearity Problem

- The collinearity problem is formally pointed out by Ackerberg et al. [2015].
- This paper is finally published in 2015, but has been circulated since 2005.

- We assumed that k_{jt} and ω_{jt} are state variables.
- Then the policy function for labor input should take the form of:

$$l_{jt} = l(k_{jt}, \omega_{jt}). \tag{3.57}$$

- However, because $\omega_{jt} = h(i_{jt}, k_{jt})$, we have:

$$l_{jt} = l[k_{jt}, h(i_{jt}, k_{jt})] = \tilde{l}(i_{jt}, k_{jt}). \tag{3.58}$$

- Therefore, in the first stage, we encounter a multicollinearity problem:

$$\begin{aligned} y_{jt} &= \beta_0 + \beta_l \tilde{l}(i_{jt}, k_{jt}) + \phi(i_{jt}, k_{jt}) + \eta_{jt} \\ &\equiv \tilde{\phi}(i_{jt}, k_{jt}). \end{aligned} \tag{3.59}$$

- Thus, β_l cannot be identified in the first step.
- The second step becomes:

$$y_{jt} = \beta_0 + \beta_l l_{jt} + \beta_k k_{jt} + g[\tilde{\phi}(i_{j,t-1}, k_{j,t-1}) - \beta_0 - \beta_l l_{j,t-1} - \beta_k k_{jt}] + \nu_{jt} + \eta_{jt} \tag{3.60}$$

- Because l_{jt} is correlated with ν_{jt}, moment can only condition on $l_{j,t-1}$.
- However, conditioning on $k_{j,t-1}$ and $i_{j,t-1}$, again there is no remaining variation in $l_{j,t-1}$.
- Therefore, β_l cannot be identified either in the second step.
- β_l **cannot be identified!**

3.3.33 Tackle Collinearity Problem: Peculiar Assumptions

- To make Olley-Pakes/Levinsohn-Petrin approach workable, we need peculiar data generating process for l_{jt}.
- Consider Levinsohn-Petrin framework.

1. There is an optimization error in l_{jt}.
   - If it is not i.i.d over time, it becomes a state variable and enters to the policy for m_{jt}, violating the scalar unobserved heterogeneity assumption of m_{jt}.
   - If there is an optimization error for m_{jt}, this again violates the scalar unobserved heterogeneity assumption.
2. k_{jt} is realized, ω_{jt} is observed, m_{jt} and i_{jt} are determined, a new i.i.d. unexpected shock is observed, l_{jt} is determined, and η_{jt} is observed.
   - If it is not i.i.d over time, it becomes a state variable and enters to the policy for m_{jt}, violating the scalar unobserved heterogeneity assumption.
3. k_{jt} is realized, an unexpected shock is observed, l_{jt} is determined, ω_{jt} is observed, m_{jt} and i_{jt} are determined, and η_{jt} is observed (Ackerberg [2016] recommends this assumption).
   - In this case, the unexpected shock can be serially correlated, because it suffices to know k_{jt}, i_{jt}, l_{jt} to decide m_{jt}. It does not have to predict the future unexpected shock based on the realization of the current shock because m_{jt} is a static decision.
   - This changes the optimal policy function of m_{jt} (3.35) to:

$$m_{jt} = m(k_{jt}, \omega_{jt}, l_{jt}). \tag{3.61}$$

   - The first step:

$$\begin{aligned} y_{jt} &= \beta_0 + \beta_l l_{jt} + \beta_k k_{jt} + h(k_{jt}, m_{jt}, l_{jt}) + \eta_{jt} \\ &= \psi(k_{jt}, m_{jt}, l_{jt}) + \eta_{jt}. \\ \Rightarrow &\mathbb{E}\{y_{jt} - \psi(k_{jt}, m_{jt}, l_{jt}) | k_{jt}, m_{jt}, l_{jt}\} = 0. \end{aligned} \tag{3.62}$$

- The second step:

$$\begin{aligned} &y_{jt} = \beta_0 + \beta_l l_{jt} + \beta_k k_{jt} + g[\psi(k_{j,t-1}, m_{j,t-1}, l_{j,t-1}) - \beta_0 - \beta_l l_{j,t-1} - \beta_k k_{j,t-1}] + \nu_{jt} + \eta_{jt} \\ &\Rightarrow \mathbb{E}\{y_{jt} - \beta_0 - \beta_l l_{jt} - \beta_k k_{jt} - g[\psi(k_{j,t-1}, m_{j,t-1}, l_{j,t-1}) - \beta_0 - \beta_l l_{j,t-1} - \beta_k k_{j,t-1}] | k_{j,t-1}, i_{j,t-1}, l_{j,t-1}, m_{j,t-1}\} \end{aligned} \tag{3.63}$$

- m_{jt} has to be excluded from the production function, i.e., it has to be a value-added production function. Otherwise, $\beta_m m_{jt}$ and $\beta_m m_{j,t-1}$ appear in the second step. Because m_{jt} is correlated with ν_{jt}, the only hope is to vary $m_{j,t-1}$. But there is no additional variation in $m_{j,t-1}$ conditional on $k_{j,t-1}$, $i_{j,t-1}$, and $l_{j,t-1}$.

3.3.34 Tackle Collinearity Problem: Share Regression

- How to avoid the peculiar assumptions on shocks and timing of decisions?
- How to identify gross production function avoiding the third assumption by Ackerberg et al. [2015]?
- Return to the old literature using the first-order condition.
- Let w_t be wage and p_t be the product price.
- Assume that the factor market is competitive.
- Then, the first-order condition for profit maximization with respect to L_{jt} is:

$$\begin{aligned} &P_t F_L(L_{jt}, K_{jt}) e^{\omega_{jt}} \mathbb{E} e^{\eta_{jt}} = w_t \\ \Leftrightarrow\ &\frac{P_t F_L(L_{jt}, K_{jt}) e^{\omega_{jt}} \mathbb{E} e^{\eta_{jt}}}{F(L_{jt}, K_{jt})} = \frac{w_t}{F(L_{jt}, K_{jt})} \\ \Leftrightarrow\ &\frac{F_L(L_{jt}, K_{jt}) L_{jt}}{F(L_{jt}, K_{jt}) e^{\eta_{jt}}} = \frac{w_t L_{jt}}{P_t \underbrace{F(L_{jt}, K_{jt}) e^{\omega_{jt}} e^{\eta_{jt}}}_{Y_{jt}}}, \end{aligned} \tag{3.64}$$

  where the right hand side is expenditure share to the labor, which is observed.
- Furthermore, on the left hand side, we only have η_{jt}, which is independent of inputs.
- Let s_{jt} be the log of expenditure share to the labor, and take a log of the previous equation gives:

$$\begin{aligned} s_{jt} &= \log[F_L(L_{jt}, K_{jt}) L_{jt} \mathbb{E} e^{\eta_{jt}} / F(L_{jt}, K_{jt})] - \eta_{jt} \\ &= \log(\beta_l) + \ln \mathbb{E} e^{\eta_{jt}} - \eta_{jt}. \end{aligned} \tag{3.65}$$

- Remember that the coefficient in the Cobb-Douglas function is equal to the expenditure share.
- In general, share regression provides additional variation to identify the elasticity of anticipated production with respect to the labor. Then we can follow the standard OP method to recover other parameters.

3.4 Cost Function Estimation

3.4.1 Cost Function: Duality

- Given a function $y = F(x)$ such that:

  - Add factor market structure.
  - Add cost minimization.

- $\rightarrow$ There exists a unique **cost function** $c = C(y, p)$:

  - **Positivity**: positive for positive input prices and a positive.
  - **Homogeneity**: homogeneous of degree one in the input prices.
  - **Monotonicity**: increasing in the input prices and in the level of output.
  - **Concavity**: concave in the input prices.

- Given a function $c = C(y, p)$ such that:

  - **Positivity**: positive for positive input prices and a positive.
  - **Homogeneity**: homogeneous of degree one in the input prices.
  - **Monotonicity**: increasing in the input prices and in the level of output.
  - **Concavity**: concave in the input prices.

- $\rightarrow$ There exists a unique production function $F(x)$ that yields $C(y, p)$ as a solution to the cost minimization problem:

$$C(y, p) = \min_x p'x \text{ s.t. } F(x) \geq y. \tag{3.66}$$

- If the latter condition holds, the function C is said to be **integrable**.
- It is rare that you can find a closed-form cost function of a production function.
- It makes sense to start from cost function.
- The duality ensures that there is a one-to-one mapping between a class of cost function and a class of production function.
- If you accept competitive factor markets and cost minimization, identifying a cost function is equivalent to identifying a production function.
- We used this idea in the last slides to identify the parameters regarding static decision variables.
- See Jorgenson [1986] for the literature in this topic up to the mid 80s.

3.4.2 Translog Cost Function

- One of the popular specifications:

$$\begin{aligned} \ln c = \alpha_0 + \alpha_p' \ln p + \alpha_y \ln y + \frac{1}{2} \ln p' B_{pp} \ln p \\ + \ln p' \beta_{py} \ln y + \frac{1}{2} \beta_{yy} (\ln y)^2. \end{aligned} \tag{3.67}$$

- It assumes that the first and second order elasticities are constant.
- A second-order (log) Taylor approximation of a general cost function.

3.4.3 Translog Cost Function: Integrability

- Translog cost function is known to be integrable if the following conditions hold:
- **Homogeneity**: the cost shares and the cost flexibility are homogeneity of degree zero: $B_{pp}1 = 0$, $\beta_{py}'1 = 0$.
- **Cost exhaustion**: the sum of cost shares is equal to unity: $\alpha_p'1 = 1$, $B_{pp}'1 = 0$, $\beta_{py}'1 = 0$.

- **Symmetry**: the matrix of share elasticities, biases of scale, and the cost flexibility elasticity is symmetric:

$$\begin{pmatrix} B_{pp} & \beta_{py} \\ \beta'_{py} & \beta_{yy} \end{pmatrix} = \begin{pmatrix} B_{pp} & \beta_{py} \\ \beta'_{py} & \beta_{yy} \end{pmatrix}'. \tag{3.68}$$

- **Monotonicity**: The matrix of share elasticities $B_{pp} + vv' - diag(v)$ is positive semi-definite.

3.4.4 Two Approaches

1. Cost data approach.
   - Use accounting cost data.
   - It does not depend on behavioral assumption.
   - One can impose restrictions of assuming cost minimization.
   - The accounting cost data may not represent economic cost.
2. Revealed preference approach.
   - Assume decision problem for firms.
   - Assume profit maximization.
   - Reveal the costs from firm's equilibrium strategy.
   - It depends on structural assumptions.
   - It reveals the cost as perceived by firms.

3.4.5 Cost Data Approach

- Estimating a cost function using cost data from accounting data.
- McElroy [1987] is one of the most flexible and robust frameworks.
- The approach is somewhat getting less popular in IO researchers.
- Recently, the approach is not popular among IO researchers.
- I *conjecture* one of the reasons for this is that IO researchers believe cost data taken from accounting information does not capture all the costs firms face.
- However, it is good to know the classical literature because it sometimes gives a new insight.
- cf. Byrne et al. [2015] : Propose a novel method to combine accounting cost data to estimate demand and cost function jointly without using instrumental variable approach.

3.4.6 Revealed Preference Approach

- Another approach is to **reveal** the marginal cost from firm's price/quantity setting behavior assuming it is maximizing profit.
  - A parameter affects economic agent's action.
  - Therefore, economic agent's action **reveals** the information about the parameter.
  - See Bresnahan [1981] and Bresnahan [1989] for reference.
- We have shown that the assumption on the factor market and cost function minimization gives restriction on the cost parameters.
- We may further assume the product market structure and profit maximization to identify cost parameters.
- Example: In a competitive market, the equilibrium price is equal to the marginal cost. Therefore, the marginal cost is identified from prices.
- What if the competition is imperfect?

3.4.7 Single-product Monopolist

- This approach requires researcher to specify the decision problem of a firm.
- Assume that the firm is a single-product monopolist.
- Let $D(p)$ be the demand function.
- Let $C(q)$ be the cost function.
- Temporarily, assume that we **know** the demand function.
- We learn how to estimate demand functions in coming weeks.
- The only unknown parameter is the cost function.
- The monopolist solves:

$$\max_p D(p)p - C(D(p)). \tag{3.69}$$

- The first-order condition w.r.t. p for profit maximization is:

$$\begin{aligned} & D(p) + pD'(p) - C'(D(p))D'(p) = 0. \\ \Leftrightarrow\ & C'(D(p)) = \underbrace{\frac{D(p) + pD'(p)}{D'(p)}}_{p \text{ is observed and } D(p) \text{ is known.}} \end{aligned} \tag{3.70}$$

- This identifies the marginal cost *at the equilibrium quantity*.
- To trace out the entire marginal cost function, you need a demand shifter Z that changes the equilibrium: $D(p, Z)$.

$$C'(D(p, z)) = \frac{D(p, z) + pD'(p, z)}{D'(p, z)} \tag{3.71}$$

- This identifies the marginal cost function *at the equilibrium quantity when* $Z = z$.
- If the equilibrium quantities cover the domain of the marginal cost function when the demand shifter Z moves around, then it identifies the entire marginal cost function.

3.4.8 Unobserved Heterogeneity in the Cost Function

- Previously we did not consider any unobserved heterogeneity in the cost function.
- Now suppose that the cost function is given by:

$$C(q) = \tilde{C}(q) + q\epsilon + \mu, \tag{3.72}$$

  and ϵ and μ are not observed.
- Moreover, because it includes anticipated shocks, it is likely to be correlated with input decisions and hence the output.

- The first-order condition w.r.t. p for profit maximization is:

$$\begin{aligned} & D(p,z) + pD'(p,z) - [\tilde{C}'(D(p,z)) + \epsilon]D'(p,z) = 0. \\ & \Leftrightarrow \tilde{C}'(D(p,z)) = \frac{D(p,z) + pD'(p,z)}{D'(p,z)} - \epsilon. \end{aligned} \tag{3.73}$$

- Take the expectation conditional on $Z = z$:

$$\tilde{C}'(D(p,z)) = \frac{D(p,z) + pD'(p,z)}{D'(p,z)} - \mathbb{E}\{\epsilon | Z = z\}. \tag{3.74}$$

- If Z and ϵ is independent, then the last term becomes zero and we can follow the same argument as before to trace out the marginal cost function.

3.4.9 Multi-product Monopolist Case

- Demand for good j is $D_j(p)$ given a price vector p.
- Cost for producing a vector of good q is $C(q)$.
- Demand function is **known** but cost function is not known.
- The monopolist solves:

$$\max_p \sum_{j=1}^{J} p_j D_j(p) - C(D_1(p), \cdots, D_J(p)). \tag{3.75}$$

- The first-order condition w.r.t. p_i for profit maximization is:

$$\begin{aligned} & D_i(p) + \sum_{j=1}^{J} p_j \frac{\partial D_j(p)}{\partial p_i} = \sum_{j=1}^{J} \frac{\partial C(D_1(p), \cdots, D_J(p))}{\partial q_j} \frac{\partial D_j(p)}{\partial p_i}. \\ & = \begin{pmatrix} \frac{\partial D_1(p)}{\partial p_i} & \cdots & \frac{\partial D_J(p)}{\partial p_i} \end{pmatrix} \begin{pmatrix} \frac{\partial C(D_1(p), \cdots, D_J(p))}{\partial q_1} \\ \vdots \\ \frac{\partial C(D_1(p), \cdots, D_J(p))}{\partial q_J} \end{pmatrix} \end{aligned} \tag{3.76}$$

- Summing up, the first-order condition w.r.t. p is summarized as:

$$\begin{aligned} & \begin{pmatrix} D_1(p) + \sum_{j=1}^{J} p_j \frac{\partial D_j(p)}{\partial p_1} \\ \vdots \\ D_J(p) + \sum_{j=1}^{J} p_j \frac{\partial D_j(p)}{\partial p_J} \end{pmatrix} = \begin{pmatrix} \frac{\partial D_1(p)}{\partial p_1} & \cdots & \frac{\partial D_J(p)}{\partial p_1} \\ \vdots & & \\ \frac{\partial D_1(p)}{\partial p_J} & \cdots & \frac{\partial D_J(p)}{\partial p_J} \end{pmatrix} \begin{pmatrix} \frac{\partial C(D_1(p), \cdots, D_J(p))}{\partial q_1} \\ \vdots \\ \frac{\partial C(D_1(p), \cdots, D_J(p))}{\partial q_J} \end{pmatrix} \\ & \Leftrightarrow \begin{pmatrix} \frac{\partial C(D_1(p), \cdots, D_J(p))}{\partial q_1} \\ \vdots \\ \frac{\partial C(D_1(p), \cdots, D_J(p))}{\partial q_J} \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{\partial D_1(p)}{\partial p_1} & \cdots & \frac{\partial D_J(p)}{\partial p_1} \\ \vdots & & \\ \frac{\partial D_1(p)}{\partial p_J} & \cdots & \frac{\partial D_J(p)}{\partial p_J} \end{pmatrix}^{-1} \begin{pmatrix} D_1(p) + \sum_{j=1}^{J} p_j \frac{\partial D_j(p)}{\partial p_1} \\ \vdots \\ D_J(p) + \sum_{j=1}^{J} p_j \frac{\partial D_j(p)}{\partial p_J} \end{pmatrix}}_{p \text{ is observed and } D(p)\text{s are known.}}. \end{aligned} \tag{3.77}$$

- Hence, the cost function is identified.
- Including unobserved heterogeneity in the cost function causes the same problem as in the previous case.

3.4.10 Oligopoly

- There are firm $j = 1, \cdots, J$ and they sell product $j = 1, \cdots, J$, that is, firm = product (for simplicity).
- Consider a price setting game. When the price vector is p, demand for product j is given by $D_j(p)$.
- The cost function for firm j is $C_j(q_j)$.
- Given other firms' price p_{-j}, firm j solves:

$$\max_{p_j} D_j(p)p_j - C_j(D_j(p)). \tag{3.78}$$

- The first-order condition w.r.t. p_j for profit maximization is:

$$\begin{aligned} &D_j(p) + \frac{\partial D_j(p)}{\partial p_j} p_j = \frac{\partial C_j(D_j(p))}{\partial q_j} \frac{\partial D_j(p)}{\partial p_j}. \\ &\frac{\partial C_j(D_j(p))}{\partial q_j} = \underbrace{\frac{\partial D_j(p)}{\partial p_j}^{-1} [D_j(p) + \frac{\partial D_j(p)}{\partial p_j} p_j]}_{p \text{ is observed and } D_j(p) \text{ is known}}. \end{aligned} \tag{3.79}$$

- In Nash equilibrium, these equations jointly hold for all firms $j = 1, \cdots, J$.]
- Including unobserved heterogeneity in the cost function causes the same problem as in the previous case.

Chapter 4

Demand Function Estimation

4.1 Motivations

- From demand function and utility maximization assumption, we can reveal the preference of the decision maker.
- Thus, estimating demand function is necessary for **evaluating the consumer welfare**.
- In IO, estimating the **price elasticity of demand** is specifically important, because it determines the **market power** of a monopolist and the size of the dead-weight loss.
- In macroeconomics, estimating demand is in important to determine the **price level**, because the price level is the minimum expenditure for a consumer to achieve the certain level of utility.
- In marketing, estimating demand is necessary to design the optimal pricing, advertising, and all the other marketing interventions.
- In principle, the theory can be applied to whatever decisions other than the consumer choice.
- Nevo [2000]:
  - How do the hypothetical mergers in the ready-to-eat cereal industry affect the market price, markup, and consumer surplus?
  - To do so, the authors estimate the demand for ready-to-eat cereals and the cost functions for each product. Then, the authors conduct counterfactual simulations of mergers to quantify the effects.
- Chung and Alcácer [2002]:
  - To what extent do firms go abroad to access technology available in other locations?
  - To study this issue, the authors estimate the firms' locational choice when going abroad.
- Rysman [2004]:
  - In Yellow pages, how do consumers evaluate the advertisement on it, and how do advertisers value consumer usage?
  - To study this, the author simultaneously estimate the consumer demand for usage of a directory, advertiser demand for advertising, and a publisher's first-order condition.
- Gentzow [2004]:
  - Are online and print newspapers substitutes or complements?
  - To study this, the author estimate a demand function in which online and print newspapers can be either substitutes or complements.
- Bayer et al. [2007]:

  - How is the preference of people for schools and neighborhoods? How is this capitalized into housing prices?
  - To do so, the authors estimate the discrete choice of residents over locations. To deal with the endogeneity between the neighborhood and the unobserved attributes of the location, the authors use the discontinuity at the school attendance zone.

- Archak et al. [2011]:
  - How does the information embedded in product reviews the consumer choice?
  - To study this, the authors estimate the discrete choice model of consumers in which the text information from the product reviews are included as the product attributes.

- Holmes [2011]:
  - Wal-Mart maintain high store density. How large is the economy of density and the sales cannibalization?
  - To study this, the author first estimate the demand function across neighborhood Wal-Mart to capture the sales cannibalization, and then estimate the cost structure from their entry and exit behaviors.

- Handbury and Weinstein [2014]:
  - Urban and rural areas differ in available products. How does the price difference change if the heterogeneity in the product availability is incorporated?
  - To do so, the authors estimate the demand function at each location, and the construct the spatial price index based on the available products at each location.

4.2 Analyzing Consumer Behaviors

- **Alternative set**.
- **Utility function**.
  - Add system of **choice sets**.
  - Aad utility maximization.
- $\rightarrow$ **Demand function**.
- In case of producer behavior, there was a chance to directly observe the output of the most primitive function, the production function.
- In case of consumer behavior, we never directly observe the output of the most primitive function, the utility function.
- We can at most identify demand functions.
- **Revealed preference theory**:
  - Samuelson [1938], Houthakker [1950], Richter [1966], Afriat [1967], Varian [1982].
  - If the demand function is derived from a preference by maximizing the preference, the demand function should satisfy some restrictions.
  - If the assumption is true, we can recover part of the preference from the demand function.

4.3 Continuous Choice

- The alternative set $\mathcal{X}$ is a subset of $\mathbb{R}^J$.
- The utility function u is rational, monotone, and continuous on $\mathcal{X}$.

- The choice sets are given by a system of **linear budget set**:

$$\mathcal{B}(p, w) = \{q \in \mathcal{X} : p \cdot q \leq w\}.$$

- If choice sets are non-linear, the following duality approach needs to be modified.

4.3.1 Duality between Utility and Expenditure Functions

- It is rather a special case that we can derive a closed form solution to a utility maximization problem.
- We can use the first-order conditions as moment conditions for identification.

$$\frac{\partial u(q)}{\partial q_i} = \lambda p_i, i = 1, \cdots, J. \tag{4.1}$$

- The derivation of a demand function from the identified utility function in general require a numerical simulation, which can be bothering.
- As well as the duality between production and cost functions, we have the same duality theorem for utility and expenditure functions.
- There is a one-to-one mapping between a class of utility functions and a class of expenditure functions.
- Therefore, it is okay to start from an expenditure function.
- It is rare that we can recover the utility function associated with an expenditure function in a closed form. But it is not often required for analysis.
- Moreover, we can easily derive other important functions from the expenditure functions.
- Let p be the price vector and u be the target utility level.
- Let $u(q)$ be a utility function.
- An expenditure function associated with the utility function is defined by:

$$e(u, p) = \min_q p \cdot q, u(q) \geq u. \tag{4.2}$$

- Let x be the total expenditure such that:

$$x = e(u, p). \tag{4.3}$$

- We can start the analysis by specifying this function instead of the utility function.

4.3.2 Deriving Other Functions

- It is easy to derive other functions from an expenditure function.
- **Indirect utility function**: invert the expenditure function to get:

$$u = e^{-1}(p, x) \equiv v(p, x). \tag{4.4}$$

- **Hicksian demand function**: apply Shepard's lemma:

$$q_i = \frac{\partial e(u, p)}{\partial p_i} \equiv h_i(u, p). \tag{4.5}$$

- **Marshallian demand function**: insert Hicksian demand function to the expenditure function:

$$q_i = h_i(v(p, x), p) \equiv d_i(p, x). \tag{4.6}$$

4.3.3 Starting from an Indirect Utility Function

- It is almost equivalent to start from an indirect utility function.
- An indirect utility function with the utility function is defined by:

$$v(p,x) \equiv \max_q u(q), p'q \leq x. \tag{4.7}$$

- We can derive Marshallian demand function by Roy's identity:

$$q_i = \frac{-\partial v(p,x)/\partial p_i}{\partial v(p,x)/\partial x} \equiv d_i(p,x). \tag{4.8}$$

4.3.4 Expenditure Share Equation

- Let's start from an expenditure function $e(p,x)$.
- By Shepard's lemma, we have:

$$\frac{\partial \ln e(u,p)}{\partial \ln p_i} = \frac{\partial e(u,p)}{\partial p_i}\frac{p_i}{e(u,p)} = \frac{p_i q_i}{x} \equiv w_i. \tag{4.9}$$

- We call this an expenditure share equation.
- The estimation is based on the share equations.

4.3.5 Almost Ideal Demand System (AIDS)

- Based on Deaton and Muellbauer [1980].
- See Angus Deaton and John Muellbauer [1980] for further reference.
- Consider an expenditure function that satisfies the following useful conditions:
  - It allows aggregation (this motivation is less important in recent days).
  - It gives an arbitrary first-order approximation to any demand system.
  - It can satisfy the restrictions of utility maximization.
  - It can be used to test the restrictions of utility maximization.

4.3.6 PIGLOG Class

- PIGLOG (price-independent generalized logarithmic) class [Muellbauer, 1976].

$$\ln e(u,p) = (1-u)\ln a(p) + u \ln b(p), \tag{4.10}$$

  where $a(p)$ and $b(p)$ are arbitrary linear homogeneous concave functions.
- Consider households that differ in total income.
- PIGLOC form ensures that the aggregate demand can be written in the same form where the total income is replaced with the sum of household total income.
- The derivatives should be given free parameters for the model to be an arbitrary first-order approximation to any demand system.
- In AIDS, we specify $a(p)$ and $b(p)$ as:

$$\begin{aligned} \ln a(p) &\equiv a_0 + \sum_k \alpha_k \ln p_k + \frac{1}{2}\sum_k \sum_j \gamma^*_{kj} \ln p_k \ln p_j \\ \ln b(p) &\equiv \ln a(p) + \beta_0 \prod_k p_k^{\beta_k}. \end{aligned} \tag{4.11}$$

4.3.7 Derive the Share Equation I

- By Roy's identify, we can derive the associated share equation:

$$w_i \equiv \frac{\partial \ln e(u,p)}{\partial \ln p_i} = \alpha_i + \sum_j \gamma_{ij} \log p_j + \beta_i u \beta_0 \prod_k p_k^{\beta_k}, \tag{4.12}$$

where

$$\gamma_{ij} = \frac{1}{2}(\gamma_{ij}^* + \gamma_{ji}^*). \tag{4.13}$$

4.3.8 Derive the Share Equation II

- Insert indirect utility function $u = v(p, x)$ to this to get:

$$w_i = \alpha_i + \sum_j \gamma_{ij} \ln p_j + \beta_i \ln \frac{x}{P}, \tag{4.14}$$

where

$$\ln P \equiv \alpha_0 + \sum_k \alpha_k \ln p_k + \frac{1}{2} \sum_j \sum_k \gamma_{kj} \ln p_k \ln p_j. \tag{4.15}$$

- P is a price index associated with the given preference.
- With the specification of Richard Stone [1954], it becomes $\ln P = \sum_j x_j \ln p_j$.
- It can be used as an approximation.

4.3.9 Specify the Detail II

- It can satisfy the restrictions of utility maximization.
- It can be used to test the restrictions of utility maximization.
  - $\sum_j x_j = 1$:

$$\sum_j \alpha_j = 1, \sum_j \gamma_{jk} = 0, \sum_j \beta_j = 0. \tag{4.16}$$

  - $e(u, p)$ is linear homogeneous in p:

$$\sum_j \gamma_{ij} = 0. \tag{4.17}$$

  - Symmetry:

$$\gamma_{ij} = \gamma_{ji}. \tag{4.18}$$

4.3.10 Estimation

- We can estimate parameters based on the share equations.
- If we use aggregate data, the aggregate error term is correlated with the price vector.
- Therefore, we need at least as many instrumental variables as the dimension of the price vector.
- With valid instrumental variables, we can estimate the model with GMM.
- If we use household-level data, the household-specific errors controlling for aggregate errors will not be correlated with the price vector if the price is determined in a competitive market.

4.3.11 From Product Space Approach to Characteristics Space Approach

- The framework up to here is called **product space approach** because the utility has been defined over a product space.
- When there are J goods, there are J^2 parameters for prices.
- One way to resolve this issue is to introduce a priori knowledge about the preference.
- For example, we can introduce a priori segmentation with separability.
- It is hard to evaluate the effect of introducing new product.
- Again, we have to a priori decide which segment/product is similar to the new product.
- This leads us to the **characteristics space approach** [Lancaster, 1966, Muth, 1966]:
  - Consumption is an activity in which goods are inputs and in which the output is a collection of characteristics.
  - Utility ranks collections of characteristics and only to rank collections of goods indirectly through the characteristics that they possesses.
  - There are $k = 1, \cdots, K$ activities.
  - The activity y requires to consume $x = Ay$ products.
  - The activity y generates $z = By$ characteristics.
  - The budget constraint is $p \cdot x \leq 1$.
  - The utility is defined over the characteristics $u(z)$.
  - The consumer's problem is:

$$\max_{y} u(z)$$

    s.t.

$$p \cdot x \leq 1, x = Ay, z = By, x, y, z \geq 0.$$

- Then, only the dimension of characteristics matters and the value of new products can be evaluated by the contribution to the production of characteristics.
- The early application includes Rosen [1974], Muellbauer [1974], Gorman [1980].
- The nonparametric analysis based on the reveals preference is Blow et al. [2008].

4.3.12 From Continuous Choice Approach to Discrete Choice Approach

- The aggregate demand is a collection of choice across consumers and within consumers over time.
- It makes sense to model individual choices and then aggregate rather than directly modeling the aggregate demand.
- The resulting aggregate demand will satisfy restrictions that are consistent with the underlying consumer choice model.
- If there is an interaction across choices, the aggregation is not trivial.
- This is especially true when aggregating choices within consumers.
- For now, assume that each choice is independent.

4.4 Discrete Choice

4.4.1 Discrete Choice Approach

- Let $u(q, z_i)$ be the utility of a consumer over $J + 1$ dimensional consumption bundle q characterized by consumer characteristics z_i.
- The consumer solves:

$$V(p, y_i, z_i) = \max_{q} u(q, z_i), \text{ s.t. } p'q \leq y_i. \tag{4.19}$$

- Alternative 0 is an **outside good**.

- Normalize $p_0 = 1$.
- We call alternatives $j = 1, \cdots, J$ **inside goods**.
- The choice space is restricted on:

$$\begin{aligned} Q = \{q :& q_0 \in [0, M], q_j \in \{0, 1\}, j = 1, \cdots, J, \\ & q_j q_k = 0, \forall j \neq k, j, k > 0, M < \infty\}. \end{aligned} \tag{4.20}$$

4.4.2 Discrete Choice Approach

- The budget constraint reduces to:

$$\begin{cases} q_0 + p_j q_j = y & \text{if } q_j = 1, j > 0 \\ q_0 = y & \text{otherwise.} \end{cases} \tag{4.21}$$

- Hence,

$$q_0 = y - \sum_{j=1}^{J} p_j q_j. \tag{4.22}$$

4.4.3 Discrete Choice Approach

- The utility maximization problem can be written as:

$$V(p, y_i, z_i) = \max_{j=0,1,\cdots,j} v_j(p_j, y_i, z_i), \tag{4.23}$$

where

$$\begin{aligned} & v_j(p_j, y_i, z_i) \\ & = \begin{cases} u(y_i - p_j, 0, \cdots, \underbrace{1}_{q_j}, \cdots, 0, z_i) & \text{if } j > 0, \\ u(y_i, 0, \cdots, 0, z_i) & \text{if } j = 0, \end{cases} \end{aligned} \tag{4.24}$$

is called the **choice-specific indirect utility**.

4.4.4 Characteristics Space Approach

- Preference is defined over the characteristics of alternatives, x_j:
- Car: vehicle, engine power, model-year, car maker, etc.
- PC: CPU power, number of cores, memory, HDD volume, etc.
- The choice-specific indirect utility is a function of the characteristics of the alternative:

$$\begin{aligned} v_j(p_j, y_i, z_i) &= u(y_i - p_j, 0, \cdots, \underbrace{1}_{q_j}, \cdots, 0, z_i) \\ &= u^*(y_i - p_j, x_j, z_i) \\ &\equiv v(p_j, x_j, y_i, z_i). \end{aligned} \tag{4.25}$$

4.4.5 Weak Separability and Income Effect

- We usually focus on a particular product category such as cars, PCs, cereals, detergents, and so on.
- Assume that the preference is separable between the category in question (**inside goods**) and other categories (**outside goods**).
- $u(q) = u[q_I, v(q_O)]$:
  - q_I: the consumption vector of inside goods.
  - q_O: the consumption vector of outside goods.
  - Increasing in $v_O = v(q_O)$.
- $p = (p_I, p_O)$.
  - p_I: the price vector of inside goods.
  - p_O: the price vector of outside goods.
- When y_O is left for the outside goods, the conditional demand for the outside goods $q_O(y_O, p_O)$ exists.
- Inserting this into the utility function gives:

$$u\{q_I, v[q_O(y_O, p_O)]\} \equiv \tilde{u}(q_I, y_O; p_O). \tag{4.26}$$

4.4.6 Weak Separability and Income Effect

- Thus, how the preference for the outside good is modeled determines how the individual income affects the choice.

$$\begin{aligned} u(y_i - p_j, x_j, z_i) &= \tilde{u}(x_j, z_i) + \alpha(y_i - p_j). \\ u(y_i - p_j, x_j, z_i) &= \tilde{u}(x_j, z_i) + \alpha \ln(y_i - p_j). \end{aligned} \tag{4.27}$$

- In the first example, the income level does not affect the choice because the term αy_i is common and constant across choices (there is no income effect).
- We often do not observe income of a consumer, y_i.
- Remember that the price of a product enters because we here consider **indirect** utility function.

4.4.7 Utility Function Normalization

- The **location** of utility function is often normalized by setting:

$$u(y^*, 0, \cdots, 0, z^*) = 0, \tag{4.28}$$

  for certain choice of (y^*, z^*).

4.4.8 Aggregation of the Individual Demand

- Let $q(p, x, y_i, z_i) = \{q_j(p, x, y_i, z_i)\}_{j=0,\cdots,J}$ be the demand function of consumer i, that is:

$$q_j(p, x, y_i, z_i) = 1 \Leftrightarrow j = \text{argmax}_{j=0,1,\cdots,J} v(p_j, x_j, y_i, z_i). \tag{4.29}$$

- Let $f(y, z)$ be the joint distribution of the income and other consumer characteristics.
- The aggregate demand for good j is:

$$\sigma_j(p, x) \equiv N \int q_j(p, x, y, z) f(y, z) dy dz, \tag{4.30}$$

  where N is the population.

4.4.9 Horizontal Product Differentiation

- **horizontal product differentiation**: consumers do not agree on the ranking of the choices.
- There are two convenience stores $j = 1, 2$ on a street $[0, 1]$.
- Let z_i be the location of consumer i and x_j be the location of the choice on a street $[0, 1]$ with $x_1 < x_2$.
- A consumer has a preference such that:

$$v_{ij} \equiv v(p_j, x_j, y_i, z_i) \equiv s - t|z_i - x_j| - p_j. \tag{4.31}$$

4.4.10 Horizontal Product Differentiation

- Suppose that the prices are low enough that entire consumers on the street are willing to buy either from the stores.
- Consumer i buys from store 1 if and only if:

$$\begin{aligned} & v(p_1, x_1, y_i, z_i) \geq v(p_2, x_2, y_i, z_i) \\ & \Leftrightarrow s - t|z_i - x_1| - p_1 \geq s - t|z_i - x_2| - p_2 \\ & \Leftrightarrow z_i \leq \frac{p_2 - p_1}{2t} + \frac{x_1 + x_2}{2} \equiv \overline{z}_1(p_1, p_2). \end{aligned} \tag{4.32}$$

- Let $f(z_i)$ be $U[0, 1]$. Then, the aggregate demand for store 1 is:

$$\sigma_1(p, x) = N \int_0^{\overline{z}_1(p_1, p_2)} dz_i = N\overline{z}_1(p_1, p_2). \tag{4.33}$$

4.4.11 Vertical Product Differentiation

- **Vertical product differentiation**: Consumers agree on the ranking of the choices. Consumers can have different willingness to pay.
- Bresnahan [1987] analyzed automobile demand with this framework.
- There are J goods and consumer i has a utility such as:

$$v_{ij} \equiv v(p_j, x_j, y_i, z_i) = z_i x_j - p_j, \tag{4.34}$$

  where x_j is a quality of product j and z_i is the consumer's willingness to pay for the quality with $x_j < x_{j+1}$.
- Consumers' problem is:

$$\max\{0, z_i x_1 - p_1, \cdots, z_i x_J - p_J\}. \tag{4.35}$$

4.4.12 Vertical Product Differentiation

- Consumer i prefers good $j + 1$ to good j if and only if:

$$\begin{aligned} & v(p_{j+1}, wxj + 1, y_i, z_i) \geq v(p_j, x_j, y_i, z_i) \\ & \Leftrightarrow z_i x_{j+1} - p_{j+1} \geq z_i x_j - p_j \\ & \Leftrightarrow z_i \geq \frac{p_{j+1} - p_j}{x_{j+1} - x_j} \equiv \Delta_j. \end{aligned} \tag{4.36}$$

- So consumer i purchases good j if and only if $z_i \in [\Delta_{j-1}, \Delta_j)$ and buys nothing if:

$$z_i \leq \Delta_0 \equiv \min\{p_1/x_1, \cdots p_J/x_j\}. \tag{4.37}$$

- Letting $F(z)$ be the distribution function of z, the aggregate demand for good j is:

$$\sigma_j(p, x, z) = N[F(\Delta_j) - F(\Delta_{j-1})]. \tag{4.38}$$

4.4.13 Econometric Models

- So far there was no econometrics.
- Next we define what are observable and unobservable, and what are known and unknown.
- Then consider how to identify and estimate the model.

4.4.14 Multinomial Logit Model: Preference Shock

- This originates at Mcfadden [1974].
- See Train [2009] for reference.
- Suppose that there is some unobservable component in consumer characteristics.
- In reality, consumers choice change somewhat randomly.
- Let's capture such a **preference shock** by consider the following model:

$$v(p_j, x_j, y_i, z_i) + \epsilon_{ij}, \tag{4.39}$$

  with some random vector:

$$\epsilon_i \equiv (\epsilon_{i0}, \cdots, \epsilon_{iJ})' \sim G. \tag{4.40}$$

- At this point, G can be any distribution and the shocks can be dependent across j within i.
- p, x, y_i, z_i are **observed** but ϵ_{ij} are **unobserved**.
- When the realization of the preference shock is given, the consumer choice is:

$$q_j(p, x, y_i, z_i, \epsilon_i) \equiv 1\{j = \text{argmax}_{k=0,\cdots,J} v(p_k, x_k, y_i, z_i) + \epsilon_{ik}\}$$

  for $k = 0, \cdots, J$.
- The **choice probability** as observed by econometrician is:

$$\sigma_j(p, x, y_i, z_i) \equiv \int q_j(p, x, y_i, z_i, \epsilon_i) dG(\epsilon_i).$$

4.4.15 Multinomial Logit Model: Distributional Assumption

- Now assume the followings:
- ϵ_{ij} are independent across j: $G(\epsilon_i) = \prod_{j=0,\cdots,J} G_j(\epsilon_{ij})$.
- ϵ_{ij} are identical across j: $G_j(\epsilon_{ij}) = \overline{G}(\epsilon_{ij})$.
- $\overline{G}$ is a type-I extreme value.
- $\rightarrow$ The density $g(\epsilon_{ij}) = \exp[-\exp(-\epsilon_{ij}) - \epsilon_{ij}]$.
- This is called the (homoskedastic) **multinomial logit model**.
- Setting the variance of ϵ_{ij} at 1 for some j is a **scale** normalization.
- By dropping some of the assumptions, we can have heteroskedastic multinomial logit model, generalized extreme value model, and so on.

- Another popular distribution assumption is to assume a multivariate normal distribution of ϵ_i. This case is called the **multinomial probit model**.

4.4.16 Multinomial Logit Model: Choice Probability

- The **choice probability** of consumer i of good j is:

$$\begin{aligned}\sigma_j(p,x,y_i,z_i) &\equiv \mathbb{P}\{j = \text{argmax}_{k=0,1,\cdots,J} v(p_k,x_k,y_i,z_i)+\epsilon_{ik}\}\\ &= \mathbb{P}\{v(p_j,x_j,y_i,z_i)-v(p_k,x_k,y_i,z_i)\geq \epsilon_{ik}-\epsilon_{ij}, \forall k\neq j\}\\ &= ...\text{after some algebra: leave as an exercise...}\\ &= \frac{\exp[v(p_j,x_j,y_i,z_i)]}{\sum_{k=0}^{J}\exp[v(p_k,x_k,y_i,z_i)]}.\end{aligned} \tag{4.41}$$

- For example, if:

$$v(p_k,x_k,y_i,z_i) = \beta_i' x_k + \alpha_i(y_i - p_k), \tag{4.42}$$

$$\begin{pmatrix}\beta_i\\ \alpha_i\end{pmatrix} = \begin{pmatrix}\beta_0\\ \alpha_0\end{pmatrix} + \begin{pmatrix}\Gamma\\ \pi'\end{pmatrix} z_i. \tag{4.43}$$

- Then, we have:

$$\begin{aligned}\sigma_j(p,x,y_i,z_i) &= \frac{\exp[\beta_i' x_j + \alpha_i(y_i-p_j)]}{\sum_{k=0}^{J}\exp[\beta_i' x_k + \alpha_i(y_i-p_k)]}\\ &= \frac{\exp[\beta_i' x_j - \alpha_i p_j]}{\sum_{k=0}^{J}\exp[\beta_i' x_k - \alpha_i p_k]}\end{aligned} \tag{4.44}$$

- If we normalize the characteristics vector so that $w_0 = 0$ holds for the outside option, it becomes:

$$\sigma_j(p,x,y_i,z_i) = \frac{\exp[\beta_i' x_j - \alpha_i p_j]}{1+\sum_{k=1}^{J}\exp[\beta_i' x_k - \alpha_i p_k]}$$

4.4.17 Multinomial Logit Model: Inclusive Value

- The expected utility for consumer i before the preference shocks are drawn under multinomial logit model is given by:

$$\begin{aligned}&\mathbb{E}\{\max_{j=0,\cdots,J} v(p_j,x_j,y_i,z_i)+\epsilon_{ij}\}\\ &= ...\text{after some algebra: leave as an exercise...}\\ &= \ln\left\{\sum_{j=0}^{J}\exp[v(p_j,x_j,y_i,z_i)]\right\} + constant.\end{aligned} \tag{4.45}$$

- This is sometimes called the **inclusive value** of the choice set.

4.4.18 Maximum Likelihood Estimation of Multinomial Logit Model

- Suppose we observe a sequence of income y_i, consumer characteristics z_i, choice q_i, product characteristics x_j and price p_j.

- $q_i = (q_{i0}, \cdots, q_{iJ})'$ and $q_{ij} = 1$ if j is chosen and 0 otherwise.
- The parameter of interest is the mean indirect utility function v.
- Then the log likelihood of $\{q_i\}_{i=1}^N$ conditional on $\{y_i, z_i\}_{i=1}^N$ and $\{x_j, p_j\}_{j=1}^J$ is:

$$\begin{aligned} l(v; q, y, z, w) &= \sum_{i=1}^{N} \ln \mathbb{P}\{q_i = q(p, x, y_i, z_i) | p, x, y_i, z_i\} \\ &= \sum_{i=1}^{N} \log \left\{ \prod_{j=0}^{J} \sigma_j(p, x, y_i, z_i)^{q_{ij}} \right\} \\ &= \sum_{i=1}^{N} \sum_{j=0}^{J} \log \sigma_j(p, x, y_i, z_i)^{q_{ij}}. \end{aligned} \tag{4.46}$$

- We can estimate the parameters by finding parameters that maximize the log likelihood.

4.4.19 Nonlinear Least Square Estimation of Multinomial Logit Model

- The multinomial logit model can be estimated by nonlinear least square method as well.
- Suppose that the share of product j among consumers with characteristics z and income y was:

$$\sigma_j(p, x, y, z). \tag{4.47}$$

- Note that:

$$\begin{aligned} \ln \sigma_j(p, x, y, z) &= \ln \left\{ \frac{\exp[v(p_j, x_j, y, z)]}{\sum_{k=0}^{J} \exp[v(p_k, x_k, y, z)]} \right\} \\ &= v(p_j, x_j, y, z) - \ln \left\{ \sum_{k=0}^{J} \exp[v(p_k, x_k, y, z)] \right\}. \end{aligned} \tag{4.48}$$

- Moreover, because of the location normalization of the utility function,

$$\sigma_0(p, x, y, z) = \frac{1}{\sum_{k=0}^{J} \exp[v(p_j, x_k, y, z)]}. \tag{4.49}$$

- Hence,

$$\ln \sigma_j(p, x, y, z) - \ln \sigma_0(p, x, y, z) = v(p, x_j, y, z). \tag{4.50}$$

- The left-hand variables are observed in the data.
- Let $s_j(y, z)$ be the share of product j among consumers with characteristics z and income y **in the data**.
- This can be calculated from the consumer-level data.
- More importantly, if there is the total sales data for each demographic, we can use this approach.
- Then, we can estimate the parameter by NLLS such that:

$$\min \sum_{(y,z)} \sum_{j=1}^{J} \{\ln[s_j(y, z)/s_0(y, z)] - v(p_j, x_j, y, z)\}^2. \tag{4.51}$$

- If v is linear in parameter, it is the ordinal least squares:

$$v(p_j, x_j, y_m) = \beta_i' x_j - \alpha_i p_j. \tag{4.52}$$

$$\ln[s_j(y, z)/s_0(y, z)] = \beta_i' x_j - \alpha_i p_j. \tag{4.53}$$

4.4.20 IIA Problem

- Multinomial logit problem is intuitive and easy to implement.
- However, there are several problems in the model.
- The most important problem is the **independence of irrelevant alternatives (IIA)** problem.
- Notice that:

$$\frac{\sigma_j(p,x,y,z)}{\sigma_k(p,x,y,z)} = \frac{\exp[v(p_j,x_j,y,z)]}{\exp[v(p_k,x_k,y,z)]}. \tag{4.54}$$

- The ratio of choice probabilities between two alternatives depend only on the mean indirect utility of these two alternatives and **independent of irrelevant alternatives (IIA)**.
- Why is this a problem?

4.4.21 Blue Bus and Red Bus Problem

- Suppose that you can go to a town by bus or by train.
- Half of commuters use a bus and the other half use a train.
- The existing bus was blue. Now, the county introduced a red bus, which is identical to the existing blue bus.
- No one take care of the color of bus. So the mean indirect utility of blue bus and red bus are equal.
- What is the new share across blue bus, red bus, and train?
- IIA $\rightarrow$ share of blue bus = share of train.
- Buses are identical $\rightarrow$ share of blue bus = share of red bus.
- Therefore, shares have to be 1/3, respectively.
- But shouldn't it be that train keeps half share and bus have half share in total?

4.4.22 Restrictive Price Elasticity

- IIA property restrict price elasticities in an unfavorable manner.
- This is a serious problem because the main purpose for us to estimate demand functions is to identify the price elasticity.
- Let $v(p_j,x_j,y,z) = \beta_z' x_j - \alpha_z p_j$. Then, we have:

$$e_{jk} = \begin{cases} -\alpha p_j(1-\sigma_j(p,x,y,z)) & \text{if } k=j \\ \alpha p_k \sigma_k(p_k,x_k,y,z) & \text{if } k \neq j. \end{cases} \tag{4.55}$$

- The price elasticity is completely determined by the existing choice probabilities of the relevant alternatives.
- Suppose that there are coca cola, Pepsi cola, and a coffee.
- The shares were 1/2, 1/6, 1/3, respectively.
- Suppose that the price of coca cola increased.
- We expect that they instead purchase Pepsi cola because Pepsi cola is more similar to coca cola than coffee.
- However, according to the previous result, twice more consumers substitute to coffee rather than to Pepsi cola.

4.4.23 Monotonic Inclusive Value

- Suppose that there is a good whose mean indirect utility is v.
- The inclusive value for this choice set is $\ln[1+\exp(v)]$.
- Suppose that we put J same goods on the shelf and consumer can choose any of them.

- The inclusive value is $\ln[1 + J\exp(v)]$.
- We just added the same goods. But the expected utility of consumer increases monotonically in the number of alternatives.

4.4.24 The Source of the Problem

- **The source of the problem is that there is no correlation in the preference shock across products**.
- When the preference shock to coca cola is high, the preference shock to Pepsi cola should be high, while the preference shock to coffee should be relatively independent.
- Because the expected value of the maximum of the preference shocks increases according to the number of alternatives, the inclusive value becomes increasing in the number of alternatives.
- However, the preference shocks should be the same for the same good. Then, the the expected value of the maximum of the preference shock should not increase even if we add the same products on the shelf.

4.4.25 Correlation in Preference Shocks

- Therefore, the preference shock should be such that: preference shocks between two alternative should be more correlated when they are closer in the characteristics space.
- So we have to allow the covariance matrix of the preference shock to be free parameters.
- If we allow flexible covariance matrix, the curse of dimensionality in the number of alternatives comes back: The dimensionality of the covariance matrix is J^2.
- Another way is to remove ϵ_{ij}: it is called a **pure characteristics model** [Berry and Pakes, 2007].
- But the pure characteristics model is computationally not straightforward.
- We explore the way of introducing mild correlation across similar products in the preference shocks.

4.4.26 Observed and Unobserved Consumer Heterogeneity

- Consider beverage demand and let $x_j = \text{carbonated}_j$ and $z_i = \text{teenager}_i$.
- Suppose that the mean indirect utility is:

$$v(p_j, x_j, y_i, z_i) = \beta_i(\text{carbonated})_j - \alpha_i p_j, \tag{4.56}$$

$$\beta_i = 0.1 + 0.2 \cdot (\text{teenager})_i. \tag{4.57}$$

- The mean utility of a carbonated drink for a teenager is 0.3 but only 0.1 for others.
- When coca cola was not available, teenager will substitute more to Pepsi cola than non-teenagers.
- IIA holds at the market-segment level but not at the market level.
- How to avoid IIA at the market-segment level?: Introduce unobserved consumer heterogeneity.
- Suppose that the mean indirect utility is:

$$\beta_i = 0.1 + 0.2 \cdot (\text{teenager})_i + \nu_i. \tag{4.58}$$

- Consumers with high ν_i values carbonated drinks more than those with low ν_i values.
- When coca cola was not available, consumers with high ν_i will substitute more to Pepsi cola than those with low ν_i values.
- IIA holds at the market-segment-ν level but not at the market-segment level.

- In the above example, "$0.2 \cdot (\text{carbonated})_i$" captures the consumer heterogeneity by observed characteristics and "ν_i" by unobserved characteristics.

4.4.27 Mixed Logit Model

- Suppose that the mean indirect utility is:

$$v(p_j, x_j, y_i, z_i, \beta_i, \alpha_i) = \beta_i' x_j - \alpha_i p_j, \tag{4.59}$$

with

$$(\beta_i, \alpha_i) \sim f(\beta_i, \alpha_i | y_i, z_i). \tag{4.60}$$

- If ϵ_{ij} is drawn i.i.d. from type-I extreme value distribution, the choice probability of good j by consumer i conditional on p, x, y_i, z_i is:

$$\sigma_j(p, x, y_i, z_i) = \int_{\beta_i, \alpha_i} \frac{\exp[v(p_j, x_j, y_i, z_i, \beta_i, \alpha_i)]}{\sum_{k=0}^{J} \exp[v(p_j, x_j, y_i, z_i, \beta_i, \alpha_i)]} f(\beta_i, \alpha_i | y_i, z_i) d\beta_i d\alpha_i. \tag{4.61}$$

- This is called the **mixed-logit model**.
- If the distribution of ϵ_{ij} is different, it is no longer mixed logit.
- Conditional on (β_i, α_i) the choice probability is written in the same way with the multinomial logit model.
- β_i, α_i are marginal out, because econometrician does not observe them.

4.4.28 Mixed Logit Model : Parametric Assumptions

- It is often assumed that:

$$v(p_j, x_j, y_i, z_i, \beta_i, \alpha_i) = \beta_i' x_j - \alpha_i p_j. \tag{4.62}$$

- McFadden and Train [2000] showed that any discrete choice models that are consistent with the random utility maximization can be arbitrarily closely approximated by this class of mixed-logit model.
- The distribution of β_i and α_i is often assumed to be:

$$\begin{aligned} \beta_i &= \beta_0 + \Gamma z_i + \Sigma \nu_i, \\ \alpha_i &= \alpha_0 + \pi' z_i + \omega \upsilon_i, \end{aligned} \tag{4.63}$$

where ν_i and υ_i are i.i.d. standard normal random vectors.

4.4.29 Mixed Logit Model: IIA

- There is no IIA at the market-segment level:

$$\frac{\sigma_j(p, x, y, z)}{\sigma_l(p, x, y_i, z_i)} = \frac{\int_{\beta_i, \alpha_i} \frac{\exp[v(p_j, x_j, y_i, z_i, \beta_i, \alpha_i)]}{\sum_{k=0}^{J} \exp[v(p_k, x_k, y_i, z_i, \beta_i, \alpha_i)]} f(\beta_i, \alpha_i | y_i) d\beta_i d\alpha_i}{\int_{\beta_i, \alpha_i} \frac{\exp[v(p_l, x_l, y_i, z_i, \beta_i, \alpha_i)]}{\sum_{k=0}^{J} \exp[v(p_k, x_k, y_i, z_i, \beta_i, \alpha_i)]} f(\beta_i, \alpha_i | y_i) d\beta_i d\alpha_i}. \tag{4.64}$$

- The share ratio depends on the price and characteristics of all the other products.

4.4.30 Mixed Logit Moel: Price Elasticities

- Let:

$$v(p_j, x_j, y_i, z_i, \beta_i, \alpha_i) = \beta_i' x_j - \alpha_i p_j. \tag{4.65}$$

- The price elasticities of the choice probabilities conditional on p, x, y_i, z_i is:

$$e_{jk} = \begin{cases} -\frac{p_j}{\sigma_j} \int \alpha_i \sigma_{ij}(1-\sigma_{ij}) f(\beta_i, \alpha_i | y_i, z_i) d\beta_i d\alpha_i & \text{if } j = k \\ \frac{p_k}{\sigma_j} \int \alpha_i \sigma_{ij} \sigma_{ik} f(\beta_i, \alpha_i | y_i, z_i) d\beta_i d\alpha_i & \text{otherwise,} \end{cases} \tag{4.66}$$

  where

$$\sigma_{ij} = \frac{\exp(\beta_i' x_j - \alpha_i p_j)}{\sum_{k=0}^{J} \exp(\beta_i' x_k - \alpha_i p_k)}. \tag{4.67}$$

- The price elasticity depends on the density of unobserved consumer types.

4.4.31 Simulated Maximum Likelihood Estimation of the Mixed Logit Model

- The choice probability of the mixed logit model is an integration of the multinomial logit choice probability.
- This is not derived analytically in general.
- We can use simulation to evaluate the choice probability:
- Draw R values of β and α, $\{\beta^r, \alpha^r\}_{r=1}^R$.
- Compute the multinomial choice probabilities associated with (β^r, α^r) for each $r = 1, \cdots, R$.
- Approximate the choice probability with the mean of the simulated multinomial choice share:

$$\sigma_j(p, x, y_i, z_i) \approx \hat{\sigma}_j(p, x, y_i, z_i) \equiv \frac{1}{R} \sum_{r=1}^{R} \frac{\exp[v(p_j, x_j, y_i, z_i, \beta^r, \alpha^r)]}{\sum_{k=0}^{J} \exp[v(p_k, x_k, y_i, z_i, \beta^r, \alpha^r)]}. \tag{4.68}$$

- This is one of the numerical integration: **Monte Carlo integration**.
- Another approach is to use **quadrature**. See Judd [1998] for reference.

4.4.32 Simulated Maximum Likelihood Estimation of the Mixed Logit Model

- There are $t = 1, \cdots, T$ markets and there $i = 1, \cdots, N$ consumers in each market.
- Let $\mathcal{J}_t$ be the set of products that are available in market t.
- Suppose that we observe income y_{it}, characteristics z_{it}, and choice q_{it} for each consumer in a market.
- Suppose that we observe product characteristics x_{jt} and price p_{jt} of each product in each market.
- The simulated conditional log likelihood is:

$$\begin{aligned} &\sum_{i=1}^{N} \sum_{t=1}^{T} \ln \mathbb{P}\{q_{it} = q(p_t, x_t, y_{it}, z_{it}) | p_t, x_t, y_{it}, z_{it}\} \\ &\approx \sum_{i=1}^{N} \ln \left\{ \prod_{j \in \mathcal{J}_t \cup \{0\}} \hat{\sigma}_j(p_t, x_t, y_{it}, z_{it})^{q_{itj}} \right\}. \end{aligned} \tag{4.69}$$

- We find parameters that maximize the simulated conditional log likelihood.

4.4.33 Simulated Non-linear Least Square Estimation of the Mixed Logit Model

- Suppose that we only know the sales or share at the market-segment level.
- That is, we only observe the share of product j in market t among consumers of characteristics z and income y, $s_{jt}(y,z)$.
- Then we can estimate the parameter by:

$$\min \sum_{t=1}^{T} \sum_{j \in \mathcal{J}_t \cup \{0\}} \sum_{(y,z) \in \mathcal{Y} \times \mathcal{Z}} \{s_{jt}(y,z) - \hat{\sigma}_j(p_t, x_t, y, z)\}^2. \tag{4.70}$$

4.4.34 Nested Logit Model: A Special Case of Mixed Logit Model

- Let $w_{j1}, \cdots, w_{jG}$ be the indicator of product category, i.e., w_{jg} takes value 1 if good j belong to category g and 0 otherwise.
- e.g., car category = {Sports, Luxury, Large, Midsize, Small}.
- We have:

$$v(p, x_j, y_i, z_i) = \beta' x_j - \alpha_i p_j + \sum_{g=1}^{G} \zeta_{ig} w_{jg} + \epsilon_{ij}. \tag{4.71}$$

- If ζ_{ig} takes high value, the consumer attaches higher value to the category.
- When a product in category g was not available, consumers with high ζ_{ig} will substitute more to the other products in the same category than consumers with low ζ_{ig}.

4.4.35 Nested Logit Model: Distributional Assumption

- Let

$$\varepsilon_{ij} \equiv \sum_{g=1}^{G} \zeta_{ig} w_{jg} + \epsilon_{ij}. \tag{4.72}$$

- Under certain distributional assumption on ζ_{ig} and ϵ_{ij}, the term ε_{ij} have a cumulative distribution [Cardell, 1997]:

$$F(\varepsilon_i) = \exp\left\{ -\sum_{g=1}^{G} \left(\sum_{j \in \text{ category } g} \exp[-\varepsilon_{ij}/\lambda_g] \right)^{\lambda_g} \right\}. \tag{4.73}$$

4.4.36 Nested Logit Model: Choice Probability

- Under this distributional assumption, the choice probability is:

$$\sigma_j(p, x, y_i, z_i) = \frac{\exp[v(p, x_j, y_i, z_i)/\lambda_g] \left(\sum_{k \in \text{ category } g} \exp[v(p, x_k, y_i, z_i)/\lambda_g] \right)^{\lambda_g - 1}}{\sum_{g=1}^{G} \left(\sum_{k \in \text{ category } g} \exp[v(p, x_k, y_i, z_i)/\lambda_g] \right)^{\lambda_g}}, \tag{4.74}$$

  if good j belongs to category g.
- The higher $\lambda_g \in [0,1]$ implies lower correlation within category g.
- $\lambda_g = 1$ for all g coincides with the multinomial logit model.

4.4.37 Nested Logit Model: Decomposition of the Choice Probability

- The choice probability can be decomposed into two parts:

$$\sigma_j(p,x,y_i,z_i) = \frac{\exp[v(p,x_j,y_i,z_i)/\lambda_g]}{\sum_{k\in \text{ category } g}\exp[v(p,x_k,y_i,z_i)/\lambda_g]} \frac{\sum_{k\in \text{ category } g}\exp[v(p,x_k,y_i,z_i)/\lambda_g]^{\lambda_g}}{\sum_{g=1}^{G}\left(\sum_{k\in \text{ category } g}\exp[v(p,x_k,y_i,z_i)/\lambda_g]\right)^{\lambda_g}}. \tag{4.75}$$

- Letting:

$$I_g(p,x,y_i,z_i) \equiv \log \sum_{k\in \text{ category } g} \exp[v(p,x_k,y_i,z_i)/\lambda_g],$$

  we have:

$$\sigma_j(p,x,y_i,z_i) = \frac{\exp[v(p,x_j,y_i,z_i)/\lambda_g]}{\sum_{k\in \text{ category } g}\exp[v(p,x_k,y_i,z_i)/\lambda_g]} \frac{\exp[\lambda_g I_g(p,x,y_i,z_i)]}{\sum_{g=1}^{G}\exp[\lambda_g I_g(p,x,y_i,z_i)]}. \tag{4.76}$$

- The second first term can be interpreted as the probability of choosing product j conditional on choosing category g and the second term as the probability of choosing category g.

4.4.38 Discrete Choice Model with Unobserved Fixed Effects

- We have assumed that good j is characterized by a vector of observed characteristics x_j.
- Can econometrician observe all the relevant characteristics of the products in the choice set? Maybe no. For example, econometrician may not observe brand values that are created by advertisement and recognized by consumers.
- Such unobserved product characteristics is likely to be correlated with the price.
- This can cause **endogeneity problems**.
- In the following, we consider the situation where only market-segment level share data is available.
- Because we can construct the market-share level data from individual choice level data, all the arguments should go through with the individual choice level data.

4.4.39 Unobserved Fixed Effects in Multinomial Logit Model

- To fix the idea, let's revisit the multinomial logit model.
- For now, we do not consider either observed or unobserved consumer heterogeneity.
- Including observed heterogeneity is straightforward.
- We discuss how to include unobserved heterogeneity in the subsequent sections.
- Suppose that the indirect utility function of good j for consumer i in market t is:

$$\beta' x_{jt} - \alpha p_{jt} - \xi_{jt} + \epsilon_{ik}, \tag{4.77}$$

- ϵ_{ik} is i.i.d. Type-I extreme value.
- ξ_{jt} is the *unobserved product-market-specific fixed effect* of product j in market t, which can be correlated with p_{jt}.
- We hold the assumption that x_{jt} is uncorrelated with ξ_{jt}.
- The choice probability of good j for this consumer and hence the choice share in this market is:

$$\sigma_j(p_t,x_t,\xi_t) = \frac{\exp(\beta' x_j - \alpha p_{jt} + \xi_{jt})}{1+\sum_{k=1}^{J}\exp(\beta' x_k - \alpha p_{kt} + \xi_{kt})}. \tag{4.78}$$

- How to deal with the endogeneity between p_{jt} and ξ_{jt}?

4.4.40 Instrumental Variables and Inversion

- Suppose that we have a vector of instrumental variables w_{jt} such that:

$$\mathbb{E}\{\xi_{jt}|w_{jt}\} = 0. \tag{4.79}$$

- In a liner model, we **invert** the model for the unobserved fixed effects:

$$\xi_{jt} = y_{jt} - \beta' x_{jt}, \tag{4.80}$$

- Notice that the unobserved fixed effect is written as a function of parameters and data.
- Then we exploit the moment condition by:

$$\begin{aligned} &\mathbb{E}\{\xi_{jt}|w_{jt}\} = 0, \\ &\Rightarrow \mathbb{E}\{\xi_{jt} w_{jt}\} = 0, \\ &\Leftrightarrow \mathbb{E}\{(y_{jt} - \beta' x_{jt}) w_{jt}\} = 0 \end{aligned} \tag{4.81}$$

- We can estimate β by finding the value that makes the sample analogue of the above expectation zero.

4.4.41 Inversion in Multinomial Logit Model

- Can we invert the multinomial model for ξ_{jt}?
- We have:

$$\begin{aligned} &\ln[\sigma_{jt}(p_t, x_t, \xi_t)/\sigma_{0t}(p_t, x_t, \xi_t)] = \beta' x_j - \alpha p_{jt} + \xi_{jt} \\ &\Leftrightarrow \xi_{jt} = \ln[\sigma_j(p_t, x_t, \xi_t)/\sigma_0(p_t, x_t, \xi_t)] - [\beta' x_j - \alpha p_{jt}]. \end{aligned} \tag{4.82}$$

- Therefore, the moment condition can be written as:

$$\begin{aligned} &\mathbb{E}\{\xi_{jt}|w_{jt}\} = 0, \\ &\Rightarrow \mathbb{E}\{\xi_{jt} w_{jt}\} = 0, \\ &\Leftrightarrow \mathbb{E}\{(\ln[\sigma_{jt}(p_t, x_t, \xi_t)/\sigma_{0t}(p_t, x_t, \xi_t)] - [\beta' x_j - \alpha p_{jt}]) w_{jt}\} = 0. \end{aligned} \tag{4.83}$$

- We can evaluate the sample analogue of the expectation by replacing the theoretical choice probability σ with the observed share s.
- At the end, it is no different from the linear model where the dependent variable is $\ln s_{jt}/s_{0t}$.

4.4.42 Market-invariant Product-specific Fixed Effects

- Furthermore, if you can assume $\xi_{jt} = \xi_j$, then

$$\ln[\sigma_j(p_t, x_t, \xi_t)/\sigma_0(p_t, x_t, \xi_t)] = \beta' x_{jt} - \alpha p_{jt} + \xi_j. \tag{4.84}$$

- This is nothing but a linear regression on x_j and p_{jt} with product-specific unobserved fixed effect.
- This can be estimated by a within-estimator.
- This specification is a good starting point: we better start with the simplest specification and use the estimate as the initial guess for the following specifications.

4.4.43 Unobserved Consumer Heterogeneity and Unobserved Fixed Effects in Mixed-logit Model

- So far we abstracted away from the unobserved consumer heterogeneity.
- Next, suppose that the indirect utility function of good j for consumer i in market t is:

$$\beta_i' x_{jt} - \alpha_i p_{jt} - \xi_{jt} + \epsilon_{ik}, \tag{4.85}$$

  where ϵ_{ik} is i.i.d. Type-I extreme value.
- The coefficient are drawn according to:

$$\begin{aligned} \beta_{it} &= \beta_0 + \Sigma \nu_{it}, \\ \alpha_{it} &= \alpha_0 + \Omega \upsilon_{it}, \end{aligned} \tag{4.86}$$

- ν_i are i.i.d. standard normal random variables.
- Then the indirect utility of good j for consumer i in market t is written as:

$$\underbrace{\beta_0' x_{jt} - \alpha_0 p_{jt} + \xi_{jt}}_{\text{(conditional) mean}} + \underbrace{\nu_{it}' \Sigma x_{jt} - \upsilon_{it}' \Omega p_{jt}}_{\text{deviation from the mean}} \tag{4.87}$$

- We refer to β_0, α_0 as **linear parameters** and Σ, Ω as **non-linear parameters**, because of the reason I explain in the subsequent section.
- Let θ_1 be the linear parameters and θ_2 the non-linear parameters and let $\theta = (\theta_1', \theta_2')'$.

4.4.44 Unobserved Fixed Effects in Mixed-logit Model

- The choice share of good j in market t is:

$$\begin{aligned} &\sigma_j(p_t, x_t, \xi_t; \theta) \\ &= \int \frac{\exp[\beta_0' x_{jt} - \alpha_0 p_{jt} + \xi_{jt} + \nu_{it}' \Sigma x_{jt} - \upsilon_{it}' \Omega p_{jt}]}{1 + \sum_{k \in \mathcal{J}_t} \exp[\exp[\beta_0' x_{kt} - \alpha_0 p_{kt} + \xi_{kt} + \nu_{it}' \Sigma x_{kt} - \upsilon_{it}' \Omega p_{kt}]]} f(\nu, \upsilon) d\nu d\upsilon. \end{aligned} \tag{4.88}$$

- How can we represent ξ_{jt} as a function of parameters of interest to exploit the moment condition?

4.4.45 Representing ξ_{jt} as a Function of Parameters of Interest

- Let s_{jt} be the share of product j in market t.
- The following system of equations implicitly determines ξ_{jt} as a function of parameters of interest:

$$s_{jt} = \sigma_j(p_t, x_t, \xi_t; \theta). \tag{4.89}$$

- Let $\xi_{jt}(\theta)$ is the solution to the system of equations above given parameter θ.
- If it exists, it is the unobserved heterogeneity as a function of parameters and data.
- Does this solution exist?
- Is it unique?
- Is there efficient method to find the solution?

4.4.46 Summarizing the Conditional Mean Term

- Now, let δ_{jt} be the conditional mean term in the indirect utility:

$$\delta_{jt} \equiv \beta_0' x_{jt} - \alpha_0 p_{jt} + \xi_{jt}. \tag{4.90}$$

- I call it the average utility of the product in the market.
- Then, the choice share of product j in market t is written as:

$$\begin{aligned}&\sigma_{jt}(\delta_t, \theta_2) \\ &\equiv \int \frac{\exp\left(\delta_{jt} + \nu'\Sigma x_{jt} - \upsilon'\Omega p_{jt}\right)}{1 + \sum_{k\in\mathcal{J}_t}\exp\left(\delta_{kt} + \nu'\Sigma x_{kt} - \upsilon'\Omega p_{kt}\right)} f(\nu, \upsilon) d\nu d\upsilon,\end{aligned} \tag{4.91}$$

for $j = 1, \cdots, J, t = 1, \cdots, T$.

4.4.47 Contraction Mapping for δ_t.

- Now, fix θ_2 and define an operator T such that:

$$T_t(\delta_t) = \delta_t + \ln \underbrace{s_{jt}}_{\text{data}} - \ln \underbrace{\sigma_{jt}(\delta_t, \theta_2)}_{\text{model}}. \tag{4.92}$$

- Let $\delta_t^{(0)} = (\delta_{1t}^{(0)}, \cdots, \delta_{Jt}^{(0)})'$ be an arbitrary starting vector of average utility of products in a market.
- Using the operator above, we update $\delta_t^{(r)}$ by:

$$\delta_t^{(r+1)} = T_t(\delta_t^{(r)}) = \delta_t^{(r)} + \ln s_{jt} - \ln \sigma_{jt}(\delta_t^{(r)}, \theta_2), \tag{4.93}$$

for $r = 0, 1, \cdots$.
- Berry et al. [1995] proved that T_t as specified above is a **contraction mapping with modulus less than one**.
- This means that:
  1. T_t has a unique fixed point;
  2. For arbitrary $\delta_t^{(r)}$, $\lim_{r\to\infty} T_t^r(\delta_t^{(0)})$ is the unique fixed point.
- The fixed point of T_t is δ_t^* such that $\delta_t^* = T_t(\delta_t^*)$, i.e.,

$$\begin{aligned}&\delta_t^* = \delta_t^* + \ln s_{jt} - \ln \sigma_{jt}(\delta_t^*, \theta_2), \\ &\Leftrightarrow s_{jt} = \sigma_{jt}(\delta_t^*, \theta_2).\end{aligned} \tag{4.94}$$

- So, the fixed point δ_t^* is the conditional mean indirect utility that solves the equality given non-linear parameter θ_2.
- Moreover, the solution is unique.
- Moreover, it can be found by iterating the operator.
- Let $\delta_t(\theta_2)$ be the solution to this equation, i.e., the limit of this operation.
- The above result is useful because it ensures the inversion and provides the algorithm to find the solution.
- The invertibility itself holds under more general settings [Berry et al., 2013].

4.4.48 Solving for $\xi_{jt}(\theta)$

- We defined the average utility as:

$$\delta_{jt} = \beta_0' x_{jt} - \alpha_0 p_{jt} + \xi_{jt}. \tag{4.95}$$

- Hence, if we set:

$$\xi_{jt}(\theta) \equiv \delta_t(\theta_2) - \left[\sum_{l=1}^{L} \beta_l x_{jl} + \alpha_0 p_{jt}\right], \tag{4.96}$$

the $\xi_{jt}(\theta)$ solves the equality:

$$s_{jt} = \sigma_j(p_t, x, \xi_t; \theta). \tag{4.97}$$

4.4.49 Solving for $\xi_{jt}(\theta)$: Summary

- In summary, ξ_{jt} that solves the equality exists and unique, and can be computed by:
- Fix $\theta = \{\theta_1, \theta_2\}$.
- Fix arbitrary starting value $\delta_t^{(0)}$ for $t = 1, \cdots, T$.
- Let $\delta_t(\theta_2)$ be the limit of $T_t^r(\delta_t^{(0)})$ for $r = 0, 1, \cdots$ for each $t = 1, \cdots, T$.
- Stop the iteration if $|\delta_t(\theta_2)^{(r+1)} - \delta_t(\theta_2)^{(r)}|$ is below a threshold.
- Let $\xi_{jt}(\theta)$ be such that:

$$\xi_{jt}(\theta) = \delta_{jt}(\theta_2) - \beta_0' x_{jt} - \alpha_0 p_{jt}. \tag{4.98}$$

- Then we can evaluate the moment at θ by:

$$\mathbb{E}\{\xi_{jt}(\theta)|w_{jt}\} = 0. \tag{4.99}$$

- We run this algorithm every time we evaluate the moment condition at a parameter value.

4.4.50 GMM Objective Function

- Find θ that solves:

$$\min_{\theta} \xi(\theta)' W \Phi^{-1} W' \xi(\theta), \tag{4.100}$$

where Φ is a weight matrix,

$$\xi(\theta) = \begin{pmatrix} \xi_{11}(\theta) \\ \vdots \\ \xi_{J_1 1}(\theta) \\ \vdots \\ \xi_{1T} \\ \vdots \\ \xi_{J_T T} \end{pmatrix}, W = \begin{pmatrix} w_{11}' \\ \vdots \\ w_{J_1 1}' \\ \vdots \\ w_{1T}' \\ \vdots \\ w_{J_T T}' \end{pmatrix}. \tag{4.101}$$

- There are $J \to \infty$ and $T \to \infty$ asymptotics. Either is fine to consistently estimate the parameters.
- $w_{jt} = (x_{jt}', w_{jt}^*)'$ where w_{jt}^* is an excluded instrument that is relevant to p_{jt}.

4.4.51 Estimating Linear Parameters

- The first-order condition for θ_1 is:

$$\theta_1 = (X_1' W \Phi^{-1} W' X_1)^{-1} X_1' W \Phi^{-1} W' \delta(\theta_2), \tag{4.102}$$

where

$$X_1 = \begin{pmatrix} x'_{11} & -p_{11} \\ \vdots & \vdots \\ x'_{J_1 1} & -p_{J_1 1} \\ \vdots & \vdots \\ x'_{1T} & -p_{1T} \\ \vdots & \vdots \\ x_{J_T T} & -p_{J_T T} \end{pmatrix}, \delta(\theta_2) = \begin{pmatrix} \delta_1(\theta_2) \\ \vdots \\ \delta_T(\theta_2) \end{pmatrix} \tag{4.103}$$

.

- If θ_2 is given, the optimal θ_1 is computed by the above formula.
- $\rightarrow$ We only have to search over θ_2.
- This is the reason why we called θ_1 linear parameters and θ_2 non-linear parameters.

4.4.52 BLP Algorithm

- Find θ_2 that maximizes the GMM objective function.
- To do so:

1. Pick up θ_2.
2. Compute $\delta(\theta_2)$ by the fixed-point algorithm.
3. Compute associated θ_1 by the formula:

$$\theta_1 = (X_1' W \Phi^{-1} W' X_1)^{-1} X_1' W \Phi^{-1} W' \delta(\theta_2), \tag{4.104}$$

4. Compute $\xi(\theta)$ from the above $\delta(\theta_2)$ and θ_1.
5. Evaluate the GMM objective function with the $\xi(\theta)$.

4.4.53 Mathematical Program with Equilibrium Constraints (MPEC)

- In the BLP algorithm, for each parameter θ, find $\xi(\theta)$ that solve:

$$s = \sigma(p, x, \xi; \theta) \tag{4.105}$$

  by the fixed-point algorithm and then evaluate the GMM objective function.
- This inner loop takes time if the stopping criterion is tight.
- If the stopping criterion is loose, the loop may stop earlier but the error may be unacceptably large.
- Dubé et al. [2012] suggest to minimize the GMM objective function with the above equation as the constraints.

$$\min_{\theta} \xi(\theta)' W \Phi^{-1} W' \xi(\theta) \text{ s.t. } s = \sigma(p, x, \xi; \theta). \tag{4.106}$$

- To enjoy the benefit of this approach, we have to analytically derive the gradient and hessian of the objective function and the constraints, which are anyway needed if we estimate the standard error with the plug-in method.
- If the problem is of small scale, BLP algorithm will be fast enough and easier to implement.
- If the problem is of large scale, you may better use the MPEC approach.

4.4.54 Instrumental Variables

- The remaining problem is how to choose the excluded instrumental variable w_{jt}^* for each product/market.
- **Cost shifters**:

  - Traditional instruments.
- **Hausman-type IV** [Hausman et al., 1994]:
  - Assume that demand shocks are independent across markets, whereas the cost shocks are correlated.
  - The latter will be true if the product is produced by the same manufacturer.
  - Then, the price of the same product in the other markets $p_{j,-t}$ will be valid instruments for the price of the product in a given market, p_{jt}.
- **BLP-type IV** [Berry et al., 1995]:
  - In oligopoly, the price of a good in a market depends on the market structure, i.e., what kind of products are available in the market.
  - For example, if there are similar products in the market, the price will tend to be lower.
  - Then, the product characteristics of other products in the market , will be valid instrument for the price of goods in a given market, p_{jt}.
  - If there are multi-product firms, whether the other good is owned by the same company will also affect the price.
  - Specifically, Berry et al. [1995] use:

$$\sum_{k \neq j \in \mathcal{J}_t \cap \mathcal{F}_f} x_{kt}, \tag{4.107}$$

$$\sum_{k \neq j \in \mathcal{J}_t \setminus \mathcal{F}_f} x_{kt}. \tag{4.108}$$

- f is the firm that owns product j and $\mathcal{F}_f$ is the set of products firm f owns.
- **Differentiation IV** [Gandhi and Houde, 2015]:
  - Let $d_{jkt} = d(x_{jt}, x_{kt})$ be some distance between product characteristics.
  - They showed that under certain conditions the optimal BLP-type IV is a function $d_{-jt}\{d_{jkt}\}_{k \neq j \in \mathcal{J}_t}$.
  - The suggest to use the moments of d_{-jt} as the excluded instrument variables.
- Weak instruments problem of BLP-type IV:
  - Armstrong [2016] argued that estimates based on BLP-type IV may be inconsistent when $J \times \infty$ asymptotics is considered, because then the market approaches the competitive market and the correlation between the markup and the product characteristics of the rivals disappear.
  - Specifically, the estimator is inconsistent if all of the following conditions are met:

1. $J \to \infty$ but T is fixed;
2. The demand/cost functions are such that the correlation between markups and characteristics of other products decreases quickly enough as $J \to \infty$.
3. There is no cost instruments or other sources of identification.

Chapter 5

Merger Analysis

5.1 Motivations

- Measuring the market power of firms and predicting the possible consequence of horizontal merger cases is one of the primary goal of empirical industrial organization.
- This is important for antitrust authority to review merger cases.
- To do so, we integrate the product/cost function estimation and demand function estimation techniques.
- We introduce the last piece of parameters that characterize the market competition, **conduct parameter**, and discuss its identification.
- Then, we conduct the first kind of **counterfactual analysis**, the **merger simularion**.
- In this exercise, we predict the market response when the ownership structure of product is changed due to a hypothetical merger.
- Every market institution needs its own model for merger simulation:
- Gowrisankaran et al. [2015]:
  - In the U.S. hospitals and managed care organizations (MGO) negotiate the hospital prices and the coinsurance rates.
  - What if hospitals are merged? How much do the hospital prices and the coinsurance rates increase?
- Smith [2004]:
  - Sometimes the same service is sold through multiple stores such as in the supermarket industry.
  - How does this multi-store nature affect the merger effects?
- Ivaldi and Verboven [2005] reviews the cases in the European Commission.

5.2 Identification of Conduct

5.2.1 Identification of Conduct

- So far we have been concerned with the two types of parameters:
  - Production and/or cost function.
  - Demand function.
- To identify the marginal cost by the revealed preference approach, we have assumed that firms are engaging in a price competition.
- The mode of competition is another parameter of interest.
- Can we infer the mode of competition instead of assuming it?

5.2.2 Marginal Revenue Function

- To be specific, consider firms producing homogeneous product.
- Under what conditions can we distinguish across Bertrand competition, Cournot competition, and collusion? [Bresnahan, 1982].
- Consider the following marginal revenue function:

$$MR(Q) \equiv \lambda Q P'(Q) + P(Q), \tag{5.1}$$

  where Q is the aggregate quantity, $P(Q)$ is the inverse demand function.
- This formula nests Bertrand, Cournot, and collusion:
- Bertrand:
  - In Bertrand, a firm cannot change the market price.
  - If a firm increases the production by one unit, whose revenue increases by $P(Q)$.
  - Therefore, $\lambda = 0$.
- Cournot:
  - In Cournot, the marginal revenue of firm f is:

$$q_f P'(Q) + P(Q).$$

  - Therefore, $\lambda = s_f$, the quantity share of the firm f.
- Collusion:
  - Under collusion, firms behave like a single monopoly.
  - Then, the marginal revenue is:

$$QP'(Q) + P(Q).$$

  - Therefore, $\lambda = 1$.
- The identification of the mode of competition in this context is equivalent to the identification of λ, the **conduct parameter**.

5.2.3 First-Order Condition

- Let $MC(q_f)$ be the marginal cost of firm f.
- Given the previous general marginal revenue function, the first-order condition for profit maximization for firm f is written as:

$$\lambda Q P'(Q) + P(Q) = MC(q_f). \tag{5.2}$$

- The system of equations for $f = 1, \cdots, F$ determine the market equilibrium.

5.2.4 Linear Model

- To be simple, consider a linear inverse demand function:

$$P^D(Q) = \frac{\alpha_0}{\alpha_1} + \frac{1}{\alpha_1} Q + \frac{\alpha_2}{\alpha_1} X + \frac{1}{\alpha_1} u^D, \tag{5.3}$$

  where X_t is a vector of observed demand sifters.
- Consider a linear marginal cost function:

$$MC(q_f) = \beta_0 + \beta_1 q_f + \beta_2 W + u^S, \tag{5.4}$$

  where W is a vector of observed cost sifters.

5.2.5 Pricing Equation

- Inserting the inverse demand function and marginal cost function to the optimality condition:

$$\frac{\lambda}{\alpha_1}Q + P^S(Q) = \beta_0 + \beta_1 q_f + \beta_2 W + u^S \tag{5.5}$$

- Summing them up and dividing by the number of firms N:

$$\frac{\lambda}{\alpha_1}Q + P^S(Q) = \beta_0 + \frac{\beta_1}{N}Q + \beta_2 W_t + u^S, \tag{5.6}$$

- This determines the aggregate pricing equation:

$$\begin{aligned} P^S(Q) &= \beta_0 + (\frac{\beta_1}{N} - \frac{\lambda}{\alpha_1})Q + \beta_2 W + u^S \\ &= \beta_0 + \gamma Q + \beta_2 W + u^S. \end{aligned} \tag{5.7}$$

- The key parameter is:

$$\gamma \equiv \frac{\beta_1}{N} - \frac{\lambda}{\alpha_1}. \tag{5.8}$$

5.2.6 Identification of Inverse Demand Function and Pricing Equation

- We have two systems of **reduced-form** equations:

$$\begin{aligned} P^D(Q) &= \frac{\alpha_0}{\alpha_1} + \frac{1}{\alpha_1}Q + \frac{\alpha_2}{\alpha_1}X + \frac{1}{\alpha_1}u^D, \\ P^S(Q) &= \beta_0 + \gamma Q + \beta_2 W + u^S. \end{aligned} \tag{5.9}$$

- If we observe a **demand shifter** X, then it can be used as an instrument for Q in the pricing equation to identify the parameters in the pricing equation.
- If we observe a **cost shifter** W_t, then it can be used as an instrument for Q in the inverse demand function to identify the parameters in the pricing equation.
- Thus, we can identify the **reduced-form parameters** $(\alpha_0, \alpha_1, \alpha_2)$ and $(\beta_0, \gamma, \beta_2)$ if we observe a demand shifter X and a cost shifter W.
- However, this is not enough to separately identify the **structural-form parameters** β_1 and λ in γ.

5.2.7 The Conduct Parameter is Unidentified

- Even if the demand function and pricing equation (supply function) are identified, we still cannot identify the conduct parameter λ.
- The price at a quantity may be high either because of the high marginal cost or because of the high markup.
- Remember that the identification of γ and α_1 do not determine the value of λ and β_1 in:

$$\gamma = \frac{\beta_1}{N} - \frac{\lambda}{\alpha_1}. \tag{5.10}$$

- β_1 is the derivative of the marginal cost.

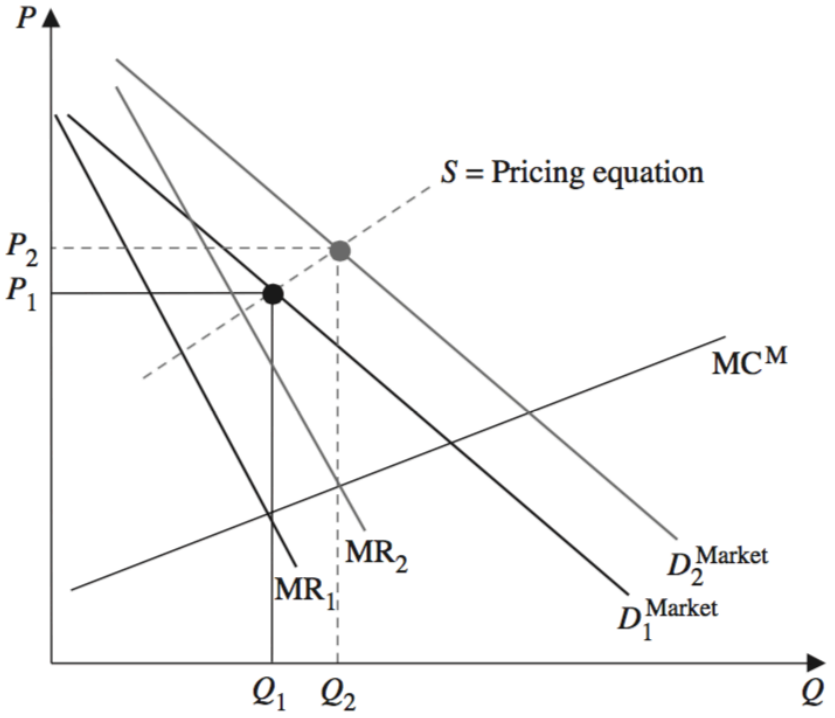


Figure 6.2. Demand shifts do not identify conduct.
Source: Authors' rendition of figure 1 in Bresnahan (1982).

Figure 5.1: Figure 6.2 of Davis (2006)

5.2.8 Identification of the Conduct Parameter: When Cost Data is Available

- If there is reliable cost data, we can directly identify the marginal cost function:

$$MC(q_f) = \beta_0 + \beta_1 q_f + \beta_2 W + u^S, \tag{5.11}$$

  and so β_1.
- Then, in a combination with the identification of inverse demand function and pricing equation, λ is identified as:

$$\lambda = \alpha_1 \left(\frac{\beta_1}{N} - \gamma \right). \tag{5.12}$$

5.2.9 Identification of the Conduct Parameter: When Cost Data is Not Available

- Remember the first-order condition:

$$\lambda Q P^{D\prime}(Q) + P^S(Q) = MC(q_f), \tag{5.13}$$

  where we can identify $P^D(Q)$ and $P^S(Q)$ if we have demand and cost shifters.
- It is clear from this expression that we need a variation in $P^{D\prime}(Q)$ with a fixed Q to identify λ, i.e., something that rotates the inverse demand function.
- Intuition: If demand becomes more elastic, prices will decrease and quantity will increase in a market with a high degree of market power.

5.2.10 Identification of the Conduct Parameter: Demand Rotater is Available

- Let's formalize the idea.
- To identify the conduct parameter, we needed a demand rotater:

$$P^D(Q) = \frac{\alpha_0}{\alpha_1} + \frac{1}{\alpha_1} Q + \frac{\alpha_2}{\alpha_1} X + \frac{\alpha_3}{\alpha_1} Q \underbrace{Z}_{\text{demand rotater}} + \frac{1}{\alpha_1} u^D. \tag{5.14}$$

- Inserting this into the first-order condition yields:

$$\frac{\lambda}{\alpha} Q_t (1 + \alpha Z_t) + P^S(Q) = \beta_0 + \frac{\beta_1}{N} Q + \beta_2 W + u^s, \tag{5.15}$$

- This determines the pricing equation:

$$\begin{aligned} P^S(Q) &= \beta_0 - \frac{\lambda}{\alpha_1} Q(1 + \alpha_3 Z_t) + \frac{\beta_1}{N} Q_t + \beta_2 W + u^S \\ &\equiv \beta_0 + \gamma_1 Q + \gamma_2 Z Q + \beta_2 W + u^S, \end{aligned} \tag{5.16}$$

  where:

$$\gamma_1 \equiv \frac{\beta_1}{N} - \frac{\lambda}{\alpha_1}, \gamma_2 \equiv -\frac{\lambda \alpha_3}{\alpha_1}. \tag{5.17}$$

5.2.11 Identification of the Conduct Parameters: Demand Rotater is Available

- If we have cost shifters W, it can be used as instruments for Q in the inverse demand function to identify the demand parameters $(\alpha_0, \alpha_1, \alpha_2, \alpha_3)$.
- If we have demand shifters X, it can be used as instruments for Q in the pricing equation to identify the supply parameters $(\beta_0, \gamma_1, \gamma_2, \beta_2)$.
- Now we can identify the **reduced-form** parameters α_1, α_3 and γ_2.
- Then, we can not identify the conduct parameter:

$$\lambda = -\frac{\gamma_2 \alpha_1}{\alpha_3}. \tag{5.18}$$

5.2.12 Identification of Conduct in Differentiated Product Market

- Consider the identification of conduct when there are two differentiated substitutable products and companies compete in price [Nevo, 1998].
- Are the prices determined independently or jointly?
- The general first-order condition is:

$$\begin{aligned} (p_1 - c_1)\frac{\partial Q_1(p)}{\partial p_1} + Q_1^S(p) + \Delta_{12}(p_2 - c_2)\frac{\partial Q_2(p)}{\partial p_1} = 0 \\ \Delta_{21}(p_1 - c_1)\frac{\partial Q_1(p)}{\partial p_2} + Q_2^S(p) + (p_2 - c_2)\frac{\partial Q_2(p)}{\partial p_2} = 0, \end{aligned} \tag{5.19}$$

where

- $\Delta_{12} = \Delta_{21} = 0$ if prices are determined independently.
- $\Delta_{12} = \Delta_{21} = 1$ if price are determined jointly.
- Under what conditions can we identify Δ_{12} and Δ_{21}?
- We will have to rotate $\frac{\partial Q_1(p)}{\partial p_2}$ and $\frac{\partial Q_2(p)}{\partial p_1}$ while keeping the other variables at the same values.
- Miller and Weinberg [2017] infers Δ after the MillerCoors, a joint venture of SABMiller PLC and Molson Coors Brewing, is formed.
  - The unobserved year-specific and region-specific cost shocks are identified from the outsiders and the unobserved product-specific cost shocks are assumed to be the same before and after the merger.

5.3 Merger Simulation

5.3.1 Unilateral and Coordinated Effects of a Horizontal Merger

- There are two effects associated with a merge episode:

1. **Unilateral effect**:
   - The new merged firm usually have a unilateral incentive to raise prices above their pre-merger level.
   - This unilateral effect may lead to the other firms to have an unilateral incentive to raise price, and the reaction continues to reach the new equilibrium.

- The latter chain reaction is sometimes called the **multi-lateral effect**.
- In economic theory, it is the price change when the same mode of competition (say, the Bertrand-Nash equilibrium) is played.
- Δ is either 0 or 1: firms internalize the profits from owned product but do not internalize the other products.

2. **Coordinated effect**:
   - After the merger, the mode of competition may change.
   - For example, the tacit collusion becomes easier and it can happen.
   - This effect, caused by the change in the mode of competition, is called the coordinated effect of a merger.
   - In this case, the conduct parameters Δ may take positive values for products owned by the rival firms.

5.3.2 Merger Simulation

- In merger simulation, we compute the equilibrium under different ownership structure.
- This amounts to run a counterfactual simulation hypothetically changing the conduct parameter Δ.
- The idea stems back to Farrell and Shapiro [1990], Werden and Frobe [1993], Hausman et al. [1994].
- Before running the simulation, you have to be very careful about the model assumptions:
  - In music industry, firms compete not in price but in advertisement.
  - If technological diffusion is important, firms may set dynamic pricing.
- This is the first **counterfactual analysis** we study in this lecture.
- The results are valid only if the modeling assumptions are correct.

5.3.3 Quantifying the Unilateral Effect

- See Nevo [2000] and Nevo [2001].
- There are J products, $\mathcal{J} = \{1, \cdots, J\}$.
- We can have multiple markets but we suppress the market index.
- Firm f produces a set of products which we denote $\mathcal{J}_f \subset \mathcal{J}$.
- Let mc_j be the constant marginal cost of producing good j.
- Thus assuming a separable cost function.
  - We can relax this assumption for the estimation.
  - However, once you admit that the costs can be non-separable, you will start to wonder what happens to the cost function if two firms merged and started to produce the two products that were previously produced by separate firms.
- Let $D_j(p)$ be the demand for product j when the price vector is p.
- The problem for firm f given the price of other firms p_{-f} is:

$$\max_{p_f} \sum_{j \in \mathcal{J}_f} \Pi_j(p_f, p_{-f}) = \sum_{j \in \mathcal{J}_f} (p_j - mc_j) D_j(p_f, p_{-f}), \tag{5.20}$$

  where $p_f = \{p_j\}_{j \in \mathcal{J}_f}$, $p_{-f} = \{p_j\}_{j \in \mathcal{J} \setminus \mathcal{J}_f}$.

5.3.4 Pre-Merger Equilibrium

- The first-order condition for firm f is:

$$D_k(p) + \sum_{j \in \mathcal{J}_f} (p_j - mc_j) \frac{\partial D_j(p)}{\partial p_k} = 0, \forall k \in \mathcal{J}_f. \tag{5.21}$$

- Let Δ_{jk}^{pre} takes 1 if same firm produces j and k and 0 otherwise before the merger.

- The first-order condition can be written as:

$$D_k(p) + \sum_{j \in \mathcal{J}} \Delta_{jk}^{pre}(p_j - mc_j)\frac{\partial D_j(p)}{\partial p_k} = 0, \forall k \in \mathcal{J}_f. \tag{5.22}$$

- In terms of the product share:

$$s_k(p) + \sum_{j \in \mathcal{J}} \Delta_{jk}^{pre}(p_j - mc_j)\frac{\partial s_j(p)}{\partial p_k} = 0, \forall k \in \mathcal{J}_f. \tag{5.23}$$

- Let Δ^{pre} be a $J \times J$ matrix whose (j,k)-element is Δ_{jk}.
- At the end of the day, performing a merger simulation is to recompute the equilibrium with different ownership structure encoded in Δ.

5.3.5 Post-Merger Equilibrium

- Let $\Omega^{pre}(p)$ is a matrix whose (j,k)-element is:

$$-\frac{\partial s_j(p)}{\partial p_k}\Delta_{jk}^{pre}. \tag{5.24}$$

- Then, by the first-order condition, the marginal cost should be:

$$mc = p - \Omega^{pre}(p)^{-1}s(p). \tag{5.25}$$

- If the ownership structure Δ^{pre} is changed to Δ^{post}, and Ω^{pre} changed to Ω^{post}, the post-merger price is determined by solving the non-linear equation:

$$p^{post} = mc + \Omega^{post}(p^{post})^{-1}s(p^{post}). \tag{5.26}$$

- The post-merger share is given by:

$$s^{post} = s(p^{post}). \tag{5.27}$$

5.3.6 Consumer Surplus

- Suppose that the demand function is based on the mixed-logit model such that the indirect utility is:

$$u_{ijt} = x_{jt}\beta_i + \alpha_i p_{jt} + \xi_j + \xi_t + \Delta\xi_{jt} + \epsilon_{ijt}, \tag{5.28}$$

  with ϵ_{ijt} is drawn from i.i.d. Type-I extreme value distribution and the consumer-level heterogeneity:

$$\begin{pmatrix}\alpha_i \\ \beta_i\end{pmatrix} = \begin{pmatrix}\alpha \\ \beta\end{pmatrix} + \Pi z_i + \Sigma\nu_i, \nu_i \sim N(0, I_{K+1}). \tag{5.29}$$

- Then, the compensated variation due to the price change for consumer i is:

$$CV_{it} = \frac{\ln(\sum_{j=0}^{J}\exp(V_{ijt}^{post})) - \ln(\sum_{j=0}^{J}\exp(V_{ijt}^{pre}))}{\alpha_i}, \tag{5.30}$$

  where V_{ijt}^{post} and V_{ijt}^{pre} are indirect utility for consumer i to purchase good j at the prices after and before the merger.
- This formula holds only if the price enters linearly in the indirect utility (no income effect).
- For general case, see Small and Rosen [1981] and Mcfadden et al. [1995].

5.3.7 Quantifying the Coordinated Effect

- The repeated-game theory suggests that a collusion is sustainable if and only if it it incentive compatible: the collusion profit is no less than the deviation profit for each member of the collusion.
- The theory provides a check list that affects the incentive compatibility such as the market share, cost asymmetry, and demand dynamics.
- But it is often hard to judge the coordinated effects from these qualitative information, because mergers simultaneously change many factors and the factors may encourage or hinder collusion.
- Miller and Weinberg [2017] **retrospectively** studies the coordinated effect of a merger.
- Is **prospective** analysis of coordinated effects possible as well as the analysis of unilateral effects?
- If we can identify the demand and cost functions, we can calculate the collusion profits and deviation profits.
- If we specify the collusion strategy, we can write down the incentive compatibility.
- We can check how the incentive compatibility change when a hypothetical merger happens.
- The problem is the identification of conduct: to identify the cost function, we need to know the conduct.
- Thus, the stated strategy will work only if we have a data during which we are sure that there was no collusion, or there was a particular type of collusion.
- Igami et al. [2018] use the detailed information of vitamin C cartel case and apply this approach.

Chapter 6

Entry and Exit

6.1 Motivations

- By studying the entry decisions of firms, we can identify the profit function including the entry cost of firms.
- A profit function is the reduced-form parameter of the underlying demand function, cost function, and conduct parameters.
- The profit function can be identified without assuming a particular conduct.
- The parameter is informative enough to answer questions regarding the market structure and producer surplus.
- In the last chapter, we discussed the identification of conduct, in which we learned that the exogenous change in the number of firms in a market gives some information about the conduct.
- In the entry/exit analysis, we study the relationship between the market size, which exogenously changes the equilibrium number of firms, and the change in the market structure to infer the conduct.
- Entry and exit is not all about firms.
- The decision of launching a product is a sort of entry decision and the decision of abolishing a product is a sort of exit decision.
- The framework in this chapter can be applied to a wider class of problems.
- This chapter is mostly based on Berry and Tamer [2006].

6.1.1 Entry Cost, Mode of Competition, and Market Structure

- Fixed and sunk entry costs and mode of competition are key determinants for market structure [Sutton, 2007].
- The tougher the mode of competition, the less firms can earn enough profit to compensate the entry cost.
- Therefore, the tougher the mode of competition, the number of firms in the market in the equilibrium cannot grow when the market size increases.

6.1.2 Exogenous and Endogenous Entry Cost

- **Exogenous fixed and sunk entry cost**:
- The cost of entry is the same across modes of entry.
- **Endogenous fixed and sunk entry cost**:
- The cost differs across modes of entry.

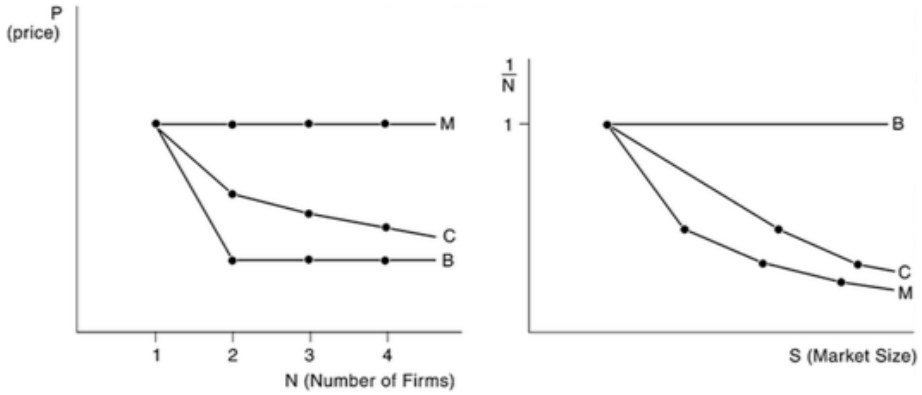


Figure 35.1. Equilibrium price as a function of the number of entrants, N, and equilibrium concentration $(1/N)$ as a function of market size, for three simple examples (B = Bertrand, C = Cournot, M = joint profit maximization).

Figure 6.1: Figure 1 of Sutton (2007)

- For example, firms decide the quality of the product upon entering the market and the the cost of entry is increasing in the quality choice.
- If endogenous fixed and sunk entry cost is relevant, the entry cost to compete with the incumbent and have a positive profit will increase as the the number of incumbent firms increases.
- Therefore, the equilibrium firm number will be small and firm size will be large when endogenous fixed and sunk entry cost is relevant.

6.2 Monopoly Entry

6.2.1 Variable Profits and Fixed Costs

- Consider a cross-section of markets, with one potential entrant in each market.
- Profits in market i are given by:
$$\pi(x_i, F_i) \equiv v(x_i) - F_i.$$
- $v(x)$ is deterministic and F is random.
- Typically, $v(x)$ is interpreted as the variable profits and F as the fixed costs.
- The potential entrant will enter market i if and only if:
$$F_i \leq v(x_i).$$
- The parameters of interest are v and the distribution of F, Φ.
- In general, F in a market can be correlated with x.

6.2.2 Non-Identification

- Assume that F is independent of x.

- Notice that a monotonic transformation of both sides of:
$$F_i \leq v(x_i).$$
will not change the entry probability.
- Therefore, without further restrictions, v and Ψ are at best identified only up to a monotonic transformation.

6.2.3 Restrictions for Identification.

- Keep assuming that F is independent of x.
- Let $p(x)$ is the observed entry probability at x.

1. v is known: Because v is the variable profit function, we can identify it directly from the demand function and cost function.
   - Let $z = v(x)$. Then Ψ is identified at z by:
$$\Psi(z) = \mathbb{P}\{F \leq z\} = \mathbb{P}\{F \leq v[v^{-1}(z)]\} = p[v^{-1}(z)].$$
2. Ψ is known: Normalize the distribution of F.
   - Then, v is identified at x by:
$$v(x) = \Psi^{-1}[p(x)].$$
3. Impose shape restrictions on v [Matzkin, 1992]:
   - $v(x)$ is homogeneous of degree 1 and there exists x_0 such that $v(x_0) = 1$. Then the functions v and Ψ are identified.
   - Homogeneous of degree 1: $v(z \cdot x) = zv(x)$.
   - Let $p(z, x_0)$ be the probability of entry at z and x_0.
   - Then, Ψ is identified at z by:
$$\Psi(z) = \mathbb{P}\{F \leq z\} = \mathbb{P}\{F \leq zv(x_0)\} = p(z, x_0).$$
   - Then, v is identified by the second argument.

6.2.4 Homogeneous of Degree 1 Variable Profits

- Sufficient condition for the variable profit function to satisfy the stated condition:
  - Demand is proportional to the population.
  - Marginal cost is constant.
  - Then, the variable profit is the market size z times the per-capita profit.
  - We can normalize v at a market of size $z = 1$ by choosing arbitrary x_0.

6.3 Complete-Information Homogeneous Oligopoly Entry

6.3.1 Variable and Fixed Costs

- Bresnahan and Reiss [1991] pioneered the analysis of oligopoly entry models.
- We observe a cross-section of markets in which we observe the number of homogeneous firms and other market-specific characteristics.
- Let y_m be the number of firms in market m.
- Let $v_{y_m}(x_m)$ is the variable profit per firm in a market with number of firms y_m and the market characteristics x_m.
- Let F_m be the market-specific fixed costs that are i.i.d. across markets with unknown distribution Ψ.
- If there are y_m firms in market m, the profit per firm in the market is:
$$\pi(y_m, x_m, F_m) \equiv v_{y_m}(x_m) - F_m.$$

6.3.2 Equilibrium Condition

- The unique Nash equilibrium in the number of firms in market m is determined by the following equilibrium condition:

$$v_{y_m}(x_m) \geq F_m.$$

$$v_{y_m+1}(x_m) < F_m.$$

- Under this equilibrium, the probability of observing y firms in a market of type x is:

$$\mathbb{P}\{y = 0|x\} = 1 - \Psi[v_1(x)].$$

$$\mathbb{P}\{y = 1|x\} = \Psi[v_1(x)] - \Psi[v_2(x)].$$

$$\mathbb{P}\{y = 2|x\} = \Psi[v_2(x)] - \Psi[v_3(x)].$$

$$\dots$$

- In other words, the probability of observing at least y firms in a market of type x is:

$$\mathbb{P}\{y \geq 1|x\} = \Psi[v_1(x)].$$

$$\mathbb{P}\{y \geq 2|x\} = \Psi[v_2(x)].$$

$$\mathbb{P}\{y \geq 3|x\} = \Psi[v_3(x)].$$

$$\dots$$

6.3.3 Identification under a Shape Restriction

- The identification argument for each y is the same as the monopoly entry case.
- For example, assume $v_y(x) = zv_y(\tilde{x})$ and $v_1(\tilde{x}_0) = 1$.
- Let $P_y(z, \tilde{x})$ be the observed probability that the number of firms is no less than y in market of type $z, \tilde{x}$.
- Then Ψ is identified at z by:

$$\Psi(z) = \mathbb{P}\{F \leq z\} = \mathbb{P}\{F \leq zv_1(\tilde{x}_0)\} = P_1(z, \tilde{x}_0).$$

- Then, the identification of v_y follows from:

$$v_y(\tilde{x}) = \frac{\Psi^{-1}[P_y(z, \tilde{x})]}{z}.$$

6.3.4 Log Likelihood Function

- The log likelihood of observing $\{y_m\}_{m=1}^M$ given $\{x_m\}_{m=1}^M$ is:

$$l(v, \Psi|\{y_m\}_{m=1}^M, \{x_m\}_{m=1}^M) = \sum_{m=1}^{M} \log\{\Psi[v_{y_m}(x_m)] - \Psi[v_{y_m+1}(x_m)]\}.$$

- This is a **ordered** model in y_m.
- If Ψ is a normal distribution, it is called an **ordered probit model**.

6.4 Complete-Information Heterogeneous Oligopoly Entry

6.4.1 Bivartiate Game with Heterogeneous Profits

- We observe a cross-section of markets in which there are two potential entrants.
- Let the profit of firm i in market m be:

$$\begin{aligned}\pi_{im}(x_{im}, y_{jm}, f_{im}) &\equiv v(y_{jm}, x_{im}) - f_{im} \\ &= v_{0i}(x_{im}) + y_{jm} v_{1i}(x_{im}) - f_{im}.\end{aligned}$$

- y_{im} and y_{jm} are the indicators of entry of firm i and j in market m.
- x_{im} and x_{jm} are firm i and j's characteristics in market m.
- f_{im} and f_{jm} are the fixed costs of firm i and j in market m.
- The second equation is without loss of generality because y_{jm} is a binary variable.
- The competitive effect of firm j on i and the effect o firm i on j are asymmetric.
- The parameters of interest are $v_{0i}, v_{0j}, v_{1i}, v_{1j}$ and the joint distribution of f_{im}, f_{jm} conditional on x_{im} and x_{jm}.
- Firms **observe** both f_{im}, f_{jm} when they make decisions, but econometrician does not.

6.4.2 Sampling Assumption and Observations

- We have a random sample of observations on markets where every observation is an observable realization of an equilibrium game played between firm i and j.
- Thus, we can observe:
  - $\mathbb{P}\{0,0|x\}$: the probability that a market of type x has no firm.
  - $\mathbb{P}\{1,0|x\}$: the probability that a market of type x has firm i but not firm j.
  - $\mathbb{P}\{0,1|x\}$: the probability that a market of type x has firm j but not firm i.
  - $\mathbb{P}\{1,1|x\}$: the probability that a market of type x has both firms.

6.4.3 Identification Assuming Pure-Strategy Equilibrium

- Tamer [2003] considers the identification when there are only two potential entrants and the pure-strategy Nash equilibrium is assumed.
- Assume that the data is from a pure-strategy equilibrium.
- Then, the probabilities $\mathbb{P}\{0,0|x\}$ and $\mathbb{P}\{1,1|x\}$ are written as:

$$\mathbb{P}\{0,0|x\} = \mathbb{P}\{f_{im} \geq v_{0i}(x_{im}), f_{jm} \geq v_{0j}(x_{jm})|x_{im}, x_{jm}\}.$$

$$\mathbb{P}\{1,1|x\} = \mathbb{P}\{f_{im} \leq v_{0i}(x_{im}) + v_{1i}(x_{im}), f_{jm} \leq v_{0j}(x_{jm}) + v_{1j}(x_{jm})|x_{im}, x_{jm}\}.$$

- Assume that (f_{im}, f_{jm}) are distributed independently of (x_{im}, x_{jm}) with a joint distribution F.
- Assume that $v_{0i}(x_{im}) = z_{im} v_0(\tilde{x}_{im})$ and $v_{0j}(x_{jm}) = z_{jm} v_0(\tilde{x}_{jm})$.
- Assume that $v_{0i}(\tilde{x}_0) = v_{0j}(\tilde{x}_0) = 1$.
- Assume that v_{1i} and v_{1j} are non-positive.
- Assume that $z_{im}|z_{jm}, \tilde{x}_{im}, \tilde{x}_{jm}$ has a distribution with support on $\mathbb{R}$ and similar for z_{jm}.
- Then, F is identified by:

$$\begin{aligned}\mathbb{P}\{f_{im} \geq z_{im}, f_{jm} \geq z_{jm}\} &= \mathbb{P}\{f_{im} \geq z_{im} v_0(\tilde{x}_0), f_{jm} \geq z_{jm} v_0(\tilde{x}_0)\} \\ &= \mathbb{P}\{0,0|z_{im}, \tilde{x}_0, z_{jm}, \tilde{x}_0\}.\end{aligned}$$

- Then, push $z_{jm} \to -\infty$ to get:

$$\begin{aligned}\mathbb{P}\{f_{im} \geq z_{im} v_0(\tilde{x}_{im})\} &= \lim_{z_{jm} \to -\infty} \mathbb{P}\{f_{im} \geq z_{im} v_0(x_{im}), f_{jm} \geq z_{jm} v_0(x_{jm})\} \\ &= \lim_{z_{jm} \to -\infty} \mathbb{P}\{0, 0 | z_{im}, \tilde{x}_{im}, z_{jm}, \tilde{x}_{jm}\}\end{aligned}$$

- Hence, v_0 is identified by:

$$v_0(\tilde{x}_{im}) = \frac{F^{-1}[\lim_{z_{jm} \to -\infty} \mathbb{P}\{0, 0 | z_{im}, \tilde{x}_{im}, z_{jm}, \tilde{x}_{jm}\}]}{z_{im}}.$$

- The identification of v_{1i} and v_{1j} are similar.

6.4.4 Heterogeneous Independent Fixed Costs

- Berry [1992] considers several extensions to the homogeneous oligopoly models.
- Assume that the fixed costs are heterogeneous and independent across firms.
  - c.f. In homogeneous oligopoly model, the fixed costs were perfectly correlated across firms in a market.
- Assume that the characteristics can be firm-specific (that can include market-specific characteristics).
- Assume that the variable profit function is homogeneous:

$$\pi_y(x_m, F_{im}) = v_y(x_{im}) - F_{im}.$$

- Assume that F is independent of x.
- Suppose that we observe the **number of potential entrants** in each market.
- For example, in the airline industry, market is a city pair.
- The potential entrants into an airline city pair were those with some service out of at least one of the endpoints of the city pair.
- The variation in the number of potential entrants can be used to identify the model.
- Define:

$$\mu(x) = \mathbb{P}\{F_{im} < v_1(x)\}.$$

$$\delta(x) = \mathbb{P}\{F_{im} < v_2(x)\}.$$

- Suppose that we **know** that there only two potential entrants into a market.
- Among such markets, we have:

$$\mathbb{P}\{0, 0 | x_1, x_2\} = [1 - \mu(x_1)][1 - \mu(x_2)].$$

$$\mathbb{P}\{1, 1 | x_1, x_2\} = \delta(x_1)\delta(x_2).$$

- If we set $x_1 = x_2 = x$, then we have:

$$\mathbb{P}\{0, 0 | x, x\} = [1 - \mu(x)]^2.$$

$$\mathbb{P}\{1, 1 | x, x\} = \delta(x)^2.$$

- They identify μ and δ at x.
- Under shape restrictions, we can identify v and Ψ as well.

6.4.5 Heterogenous and Market-Level Correlated Fixed Costs

- Let ϵ_m be the market-specific shock that affects the entry into market m.
- Conditional on ϵ_m, F_{im} are still independent across firms in a market.
- Let μ be:
$$\mu(x, \epsilon_m) = \mathbb{P}\{F_{im} < v_1(x, \epsilon_m)\}.$$
- When there is a market in which we **know** that there are only two potential entrants, the probability of observing $(0, 0)$ becomes:
$$\mathbb{P}\{0, 0|x_1, x_2\} = \int [1 - \mu(x_1, \epsilon_m)][1 - \mu(x_2, \epsilon_m)]d\Gamma(\epsilon_m).$$
- Specifically, assume that:
$$\epsilon_m = \begin{cases} 0 & \text{with probability } \lambda \\ 1 & \text{with probability } 1 - \lambda. \end{cases}$$
- Then, we in a market in which we **know** that there are only two potential entrants, the probability of observing $(0, 0)$ becomes:
$$\mathbb{P}\{0, 0|x_1, x_2\} = \lambda[1 - \mu(x_1, 0)][1 - \mu(x_2, 0)] + (1 - \lambda)[1 - \mu(x_1, 1)][1 - \mu(x_2, 1)].$$
- In general, in a market in which we **know** that there are only K potential entrants such that $x_1 = \cdots x_K = x$, the probability of observing $(0, 0)$ becomes:
$$\mathbb{P}\{0, 0|x, \cdots x\} = \lambda[1 - \mu(x, 0)]^K + (1 - \lambda)[1 - \mu(x, 1)]^K.$$
- If there markets with $K = 1, \cdots, \overline{K}$, we can construct $\overline{K}$ equations with three unknowns λ, $\mu(x, 0)$, and $\mu(x, 1)$.
- Thus, the knowledge and the variation in the number of potential entrants help the identification.

6.4.6 Inference Based On a Unique Prediction

- Berry [1992] develops a more general model built on the ideas above.
- The key for his analysis is that he ensures that **there is a unique number of equilibrium entrants**.
- This enables him to calculate the likelihood of observing a sequence of number of entrants, but at the cost of the generality of the underlying model.

6.4.7 Entry in the Airline Industry: One-shot Game

- Based on Berry [1992].
- A market = a city pair market at a single point in time.
- Consider a one-shot entry game that yields a network structure.
- At the beginning of the period, each firm takes its overall network structure as given and decides whether to operate in a given city pair **independently** across markets.

6.4.8 Entry in the Airline Industry: Profit Function

- There are K_m potential entrants in market m.
- Let y_m be a strategy profile.
- $y_m = (y_{1m}, \cdots, y_{K_m m})'$, $y_{im} \in \{0, 1\}$.
- The profit function for firm i in market m:

$$\pi_{im}(y_m, f_{im}) = v_m(N_m) - f_{im}. \tag{6.1}$$

- $N_m = \sum_{i=1}^{K_m} y_{im}$.
- v_m is strictly decreasing in N_m.

6.4.9 Entry in the Airline Industry: Profit Function

- The common term is assumed to be:

$$v_m(N) = x_m'\beta + h(\delta, N_m) + \rho u_m,$$

- x_m is the observed market characteristics, $h(\cdot)$ is a function that is decreasing in N_m, say, $-\delta \ln(N_m)$.
- u_m is the market characteristics that is observed by firms but not by econometrician.
- The firm-specific term:

$$f_{im} = z_{im}'\alpha + \sigma u_{im},$$

- z_{im} is the observed firm characteristics.
- A scale normalization: $\sigma = \sqrt{1-\rho^2} \Rightarrow var(\rho u_m + \sigma u_{im}) = 1$.

6.4.10 Entry in the Airline Industry: Likelihood Function

- The observed part:

$$r_{im}(N) = x_m'\beta - \delta \ln(N_m) + z_{im}'\alpha.$$

- The unobserved part:

$$\epsilon_{im} = \sqrt{1-\rho^2} u_{im} + \rho u_m.$$

6.4.11 Is the Equilibrium Number of Firms Unique?

- Either of the following conditions are sufficient:
- No firm-level unobserved heterogeneity: $\rho = 1$.
- No market-level unobserved heterogeneity: $\rho = 0$.
- The order of entry is predetermined, for example, the most profitable firms enter first.
- The incumbent firms enter first.
- Under either of the above assumptions, simulate the equilibrium number of firms in each market and match with the data.

6.5 Multiple Prediction

- If we further generalize the model, we will suffer from the problem of **multiple prediction**.
- First, even in Berry [1992]'s framework, the identity of entrants were not uniquely predicted.
- Second, if we allows for the asymmetric competitive effects, we will not have the unique number of equilibrium entrants.
- Third, the uniqueness of the equilibria may not hold once we allow for the mixed-strategy Nash equilibria.

- If we do not have the unique prediction on the endogenous variables, we cannot write down the likelihood function.

6.5.1 Multiple Equilibria in Bivariate Game with Heterogenous Profits

- Return to Tamer [2003]'s bivariate game with heterogeneous profits.
- Consider the pure-strategy Nash equilibrium when f_{im}, f_{jm} are realized.
- $(0,0)$ is a pure-strategy Nash equilibrium if:

$$f_{im} \geq v_{0i}(x_{im});$$

$$f_{jm} \geq v_{0j}(x_{jm}).$$

- $(1,1)$ is a pure-strategy Nash equilibrium if:

$$f_{im} \leq v_{0i}(x_{im}) + v_{1i}(x_{im});$$

$$f_{jm} \leq v_{0j}(x_{jm}) + v_{1j}(x_{jm}).$$

- $(0,1)$ is a pure-strategy Nash equilibrium if:

$$f_{im} \geq v_{0i}(x_{im}) + v_{1i}(x_{im});$$

$$f_{jm} \leq v_{0j}(x_{jm}).$$

- $(1,0)$ is a pure-strategy Nash equilibrium if:

$$f_{im} \leq v_{0i}(x_{im});$$

$$f_{jm} \geq v_{0j}(x_{jm}) + v_{1j}(x_{jm}).$$

- In a certain region of f_{im}, f_{jm}, there are multiple equilibria.

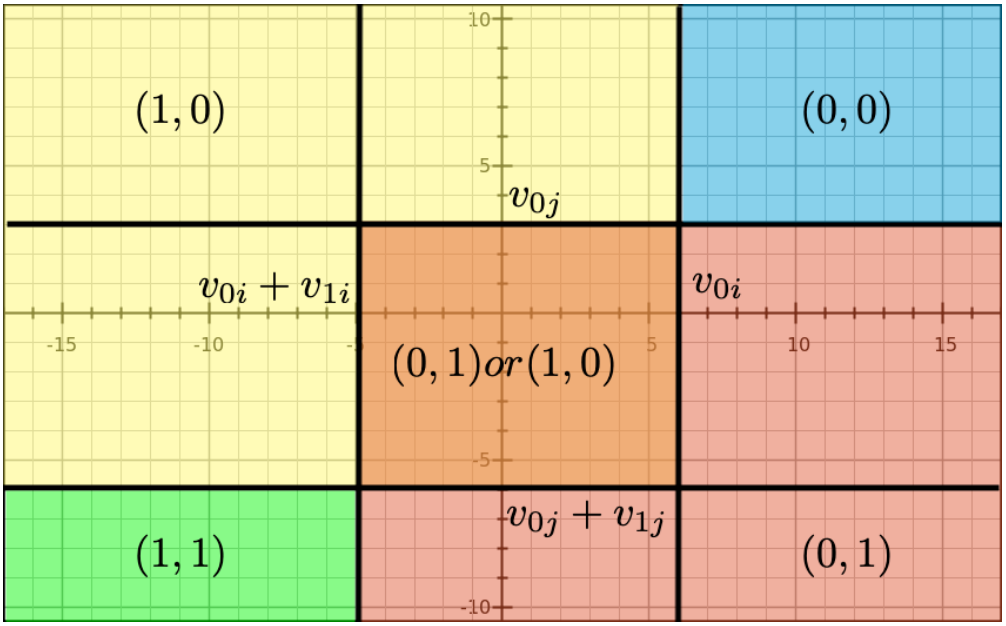
6.5.2 Multiple Prediction in Bivariate Game with Heterogenous Profits

- In the orange region, we have multiple pure-strategy equilibria.
- If we allow for mixed-strategy equilibria, the region of f_{im}, f_{jm} on which there are multiple equilibria will increase.
- In this example, we can write the likelihood of the number of equilibrium entrants:
  - 2: green area.
  - 1: yellow, orange, and red areas.
  - 0: blue area.
- However, we cannot write the likelihood of the strategy profile.
  - $(1,1)$: green area.
  - $(0,1)$: $\geq$ red area, $\leq$ orange and red area.
  - $(1,0)$: $\geq$ yellow area, $\leq$ orange and yellow area.
  - $(0,0)$: blue area.

6.5.3 Inference Based on Moment Inequalities

- According to the theory, we have:

$$\mathbb{P}\{0,0|x\} = \mathbb{P}\{f_i \geq v_{0i}(x_i), f_j \geq v_{0j}(x_j)\} \equiv H(0,0|x).$$



$$\mathbb{P}\{1,1|x\} = \mathbb{P}\{f_i \leq v_{0i}(x_i) + v_{1i}(x_i), f_j \leq v_{0j}(x_j) + v_{1j}(x_j)\} \equiv H(1,1|x).$$

$$\mathbb{P}\{0,1|x\} \geq \mathbb{P}\{f_i \geq v_{0i}, f_j \leq v_{0j}\} + \mathbb{P}\{f_i \geq v_{0i} + v_{1i}, f_j \leq v_{0j} + v_{1j}\} \equiv \underline{H}(0,1|x).$$

$$\mathbb{P}\{0,1|x\} \leq \underline{H}(0,1|x) + \mathbb{P}\{v_{0i} + v_{1i} \leq f_i \leq v_{0i}, v_{0j} + v_{1j} \leq f_j \leq f_{0j}\} \equiv \overline{H}(0,1|x).$$

$$\mathbb{P}\{1,0|x\} \geq \mathbb{P}\{f_j \geq v_{0j}, f_i \leq v_{0i}\} + \mathbb{P}\{f_j \geq v_{0j} + v_{1j}, f_i \leq v_{0i} + v_{1i}\} \equiv \underline{H}(1,0|x).$$

$$\mathbb{P}\{1,0|x\} \leq \underline{H}(0,1|x) + \mathbb{P}\{v_0 + v_{1i} \leq f_i \leq v_{0i}, v_{0j} + v_{1j} \leq f_j \leq f_{0j}\} \equiv \overline{H}(1,0|x).$$

- The parameters should satisfy the moment conditions:

$$\mathbb{P}\{0,0|x\} - H(0,0|x) = 0;$$

$$\mathbb{P}\{1,1|x\} - H(1,1|x) = 0;$$

$$\min\{\mathbb{P}\{0,1|x\} - \underline{H}(0,1|x), 0\} = 0;$$

$$\max\left\{\mathbb{P}\{0,1|x\} - \overline{H}(0,1|x), 0\right\} = 0;$$

$$\min\{\mathbb{P}\{1,0|x\} - \underline{H}(1,0|x), 0\} = 0;$$

$$\max\left\{\mathbb{P}\{1,0|x\} - \overline{H}(1,0|x), 0\right\} = 0;$$

- We can estimate the parameters with the GMM method using the above modified moment conditions.
- Inference based on the moment inequalities are found in Andrews and Soares [2010].
- Ciliberto and Tamer [2009] study the entry exit of airlines when there are heterogeneous competitive effects using the above approach.

6.6 Incompelete-Information Heterogenous Oligopoly Entry

6.6.1 Bivariate Case

- There are two potential entrants.
- Let the profit of firm i in market m be:

$$\begin{aligned}\pi_{im}(x_{im}, y_{jm}, f_{im}) &\equiv v(y_{jm}, x_{im}) - f_{im} \\ &= v_{0i}(x_{im}) + y_{jm} v_{1i}(x_{im}) - f_{im}.\end{aligned}$$

- y_{im} and y_{jm} are the indicators of entry of firm i and j in market m.
- x_{im} and x_{jm} are firm i and j's characteristics in market m.
- f_{im} and f_{jm} are the fixed costs of firm i and j in market m.
- The second equation is without loss of generality because y_{jm} is a binary variable.
- The competitive effect of firm j on i and the effect o firm i on j are asymmetric.
- The parameters of interest are $v_{0i}, v_{0j}, v_{1i}, v_{1j}$ and the joint distribution of f_{im}, f_{jm} conditional on x_{im} and x_{jm}.
- Firm i **observe** both f_{im} but **not** f_{jm} when it makes decision.
- Firm j **observe** both f_{jm} but **not** f_{im} when it makes decision.
- Econometrican does not observe either of them.
- The joint distribution of f_{im}, f_{jm} is F (independent of x).

6.6.2 The Equilibrium Strategy and Belief

- The equilibrium strategy becomes a step function that decreases in a threshold:

$$y_{im} = 1\{f_{im} \leq t_{im}\}.$$

$$y_{jm} = 1\{f_{jm} \leq t_{jm}\}.$$

- Suppose that firm i believes that f_{jm} has a distribution of G_{jm}^{im} and firm j believes that f_{im} has a distribution of G_{im}^{jm}.
- Usually, we assume a common and objective prior: $G_{jm}^{im}(\epsilon_{jm}) = F(\epsilon_{jm}|\epsilon_{im})$ and $G_{im}^{jm}(\epsilon_{im}) = F(\epsilon_{im}|\epsilon_{jm})$.
- When firm j follows strategy t_{jm}, the expected payoff for firm i to enter is:

$$[v_{0i}(x_{im}) - f_{im}][1 - G_{jm}^{im}(t_{jm})] + [v_{0i}(x_{im}) + v_{1i}(x_{im} - f_{im})]G_{jm}^{im}(t_{jm}).$$

- Thus, the threshold t_{im} is determined by:

$$[v_{0i}(x_{im}) - t_{im}][1 - G_{jm}^{im}(t_{jm})] + [v_{0i}(x_{im}) + v_{1i}(x_{im} - t_{im})]G_{jm}^{im}(t_{jm}) = 0.$$

- In the same way, the threshold t_{jm} is determined by:

$$[v_{0j}(x_{jm}) - t_{jm}][1 - G_{im}^{jm}(t_{im})] + [v_{0j}(x_{jm}) + v_{1j}(x_{jm} - t_{jm})]G_{im}^{jm}(t_{im}) = 0.$$

- This system of equations can have multiple solution.

6.6.3 Equilibrium Selection and Likelihood

- If we specify the equilibrium selection rule $\mathbb{P}(t_{im}, t_{jm}|x_{im}, x_{jm}, f_{im}, f_{jm})$, then we can specify the likelihood of observing $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$.
- If we do not, we will only have bounds on the likelihood of observing $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$.

- Another way is to assume that **the same equilibrium** $t^*_{im}(x_{im}, x_{jm}), t^*_{jm}(x_{im}, x_{jm})$ **holds across markets**.
- Then, the across market relative frequency of entries conditional on x_{im}, x_{jm} gives the estimates of the entry probabilities:

$$\widehat{G}_{im}(x_{im}, x_{jm}) \approx G^{jm}_{im}[t^*_{im}(x_{im}, x_{jm})],$$
$$\widehat{G}_{jm}(x_{im}, x_{jm}) \approx G^{im}_{jm}[t^*_{jm}(x_{im}, x_{jm})],$$

  for each x_{im}, x_{jm}.
- The estimated distribution has to solve:

$$\widehat{G}_{im}(x_{im}, x_{jm}) = \mathbb{P}\{[v_{0i}(x_{im}) - f_{im}][1 - \widehat{G}_{jm}(x_{im}, x_{jm})] + [v_{0i}(x_{im}) + v_{1i}(x_{im}) - f_{im}]\widehat{G}_{jm}(x_{im}, x_{jm}) > 0\}.$$

$$\widehat{G}_{jm}(x_{im}, x_{jm}) = \mathbb{P}\{[v_{0j}(x_{jm}) - f_{jm}][1 - \widehat{G}_{im}(x_{im}, x_{jm})] + [v_{0j}(x_{jm}) + v_{1j}(x_{jm}) - f_{jm}]\widehat{G}_{im}(x_{im}, x_{jm}) > 0\}.$$

- We find parameters $v_{0i}, v_{0j}, v_{1i}, v_{1j}, F$ that solve this system of equations.
- The "same equilibrium" assumption is hardly justified, but is often used in the empirical studies.

Chapter 7

Dynamic Decision Model

7.1 Motivations

- The model is **dynamic** if there is an **endogenous state variable**, a state variable that is affected by an action of a player in the past.
- There are many cases where the decision makers have to take into account the dynamic effects of their actions.
- **Payoff linkages**:
  - Storable good: Next week's demand for a detergent depends on how many bottles consumers purchase and consume this week. The latter depends on this week's price and the price schedule of the detergent. The stock of the detergent consumers hold is an endogenous state variable.
  - Learning by doing: The productivity of a firm next year can be higher if the firm produced more this year because the firm can learn from the experience. The cumulative production level is an endogenous state variable.
- **Information linkages**:
  - Uncertainty about the quality: If a consumer is uncertain about the quality of a product but can learn from experiencing the product, next season's demand for the product depends on how many consumers purchase and experience the product this season. The latter depends on this season's price and the price schedule of the product.
- **Strategic linkages**:
  - Tacit collusion: If a firm deviates from the collusive price, the price war will start. Then, the history of the prices is the endogenous state variable.

7.2 Single-Agent Model

7.2.1 Setting

- This model originates at Rust [1987], while the setting and the notation follows Pesendorfer and Schmidt-Dengler [2008].
- We start from a simple set-up:
- Single agent.
- Infinite-horizon discrete time.
  - Time is $t = 1, 2, \cdots, \infty$.
- Finitely many choices.
  - There are $K + 1$ actions $A = \{0, 1, \cdots, K\}$.
- Finite state space.

  - There are L states $S = \{1, \cdots, L\}$.
- Markovian state transition.

7.2.2 Timing of the Model

- At period t:
- State $s_t \in S$ is publicly observed.
- Choice-specific profitability shocks $\epsilon_t \in \mathbb{R}^{K+1}$ are realized according to $F(\cdot|s_t)$ and privately observed.
- Choice $a_t \in A$ is made.
- State evolves according to a transition probability:

$$g(a, s, s') := \mathbb{P}\{s_{t+1} = s' | s_t = s, a_t = a\}, \tag{7.1}$$

- Thus, the transition law only depends on today's state and action, but not on the past history.

$$G := \begin{pmatrix} g(0,1,1) & \cdots & g(0,1,L) \\ \vdots & & \vdots \\ g(K,1,1) & \cdots & g(K,1,L) \\ & \vdots & \\ g(0,L,1) & \cdots & g(0,L,L) \\ \vdots & & \vdots \\ g(K,L,1) & \cdots & g(K,L,L) \end{pmatrix}. \tag{7.2}$$

7.2.3 Period Payoff

- When the state is s_t, action is a_t, and the profitability shocks are ϵ_t, the period payoff is:

$$\pi(a_t, s_t) + \sum_{k=1}^{K} \epsilon_{tk} 1\{a_t = k\}, \tag{7.3}$$

- $\pi(a_t, s_t)$ is the mean **choice-specific period payoff**.
- ϵ_t is assumed to be i.i.d. across times and is drawn from F.
- Let:

$$\epsilon_{ta_t} := \sum_{k=1}^{K} \epsilon_{tk} 1\{a_t = k\},$$

  be the choice-specific profitability shock.
- Let Π summarize the choice-specific period payoffs at each state:

$$\Pi = \begin{pmatrix} \pi(0,1) \\ \vdots \\ \pi(K,1) \\ \vdots \\ \pi(0,L) \\ \vdots \\ \pi(K,L) \end{pmatrix}. \tag{7.4}$$

- The **payoff** is the discounted sum of future payoffs with discount factor $\beta < 1$.
- Π is one of the parameters of interest.

7.2.4 Markovian Framework

- The strategy is in general a mapping from the **entire** history to the action set.
- We restrict the set of possible strategies to **Markovian strategies** $a(\epsilon_t, s_t)$ that only depends on the latest realization of states ϵ_t, s_t, i.e., the behavior does not depend on the past states, conditional on today's states.
- The Markovian strategy is introduced by Maskin and Tirole [1988b]. But their model is slightly different from the current mode. They considered an oligopoly model in which only one firm can move at one time and his/her action depends only on the rival's latest move. The current single-agent model can be regarded as a version of Maskin and Tirole [1988b]'s model such that the rival is replaced with the nature.

7.2.5 Belief

- When a play makes a decision, s/he should have some belief about the future ϵ_t, s_t, and a_t.
- We usually assume the **rational expectation**: the play knows the equilibrium distribution of these future variables and use it as his/her belief.
- The distribution of ϵ_t and s_t is believed to follow F and G.
- Let $\sigma(a|s)$ be the player's belief about the possibility of taking a when the realized state is s, which may or may not coincide with the equilibrium probability.
- Let σ stack them up as:

$$\sigma = \begin{pmatrix} \sigma(0|1) \\ \vdots \\ \sigma(K|1) \\ \vdots \\ \sigma(0|L) \\ \vdots \\ \sigma(K|L) \end{pmatrix}. \tag{7.5}$$

7.2.6 Decision Problem

- The agent chooses strategy $a(\cdot,\cdot)$ such that:

$$\begin{aligned} &\max_{a(\cdot,\cdot)} \pi[a(\epsilon_0, s_0), s_0] + \epsilon_{0a(\epsilon_0,s_0)} \\ &\quad + \mathbb{E}\left\{ \sum_{t=1}^{\infty} \beta^t \left[\pi(a_t, s_t) + \epsilon_{ta(\epsilon_t,s_t)} \right] \middle| s_0, a(\epsilon_0, s_0) \right\} \end{aligned} \tag{7.6}$$

- The expectation is taken with respect to his/her belief.

7.2.7 Value Function and Ex-ante Value Function

- When the belief about the future behavior is σ, then the value function associated with the belief is defined as:

$$\begin{aligned} &V(\sigma, s_0, \epsilon_0) \\ &= \sum_{a\in A} \sigma(a|s_0) \left\{ \pi(a, s_0) + \epsilon_{0a} + \mathbb{E}\left[\sum_{t=1}^{\infty} \beta^t \sum_{a\in A} \sigma(a|s_t) \left(\pi(a, s_t) + \epsilon_{ta} \right) \middle| s_0, a \right] \right\}. \end{aligned} \tag{7.7}$$

- It has the recursive structure as:

$$\begin{aligned} & V(\sigma, s_0, \epsilon_0) \\ &= \sum_{a \in A} \sigma(a|s_0) \left\{ \pi(a, s_0) + \epsilon_{0a} + \beta \mathbb{E}\left[V(\sigma, s_1, \epsilon_1) \middle| s_0, a \right] \right\} \\ &= \sum_{a \in A} \sigma(a|s_0) \left\{ \pi(a, s_0) + \epsilon_{0a} + \beta \sum_{s_1 \in S} V(\sigma, s_1, \epsilon_1) g(a, s_0, s_1) \right\}. \end{aligned} \tag{7.8}$$

- $V(\sigma, s, \epsilon)$ is the value function after the profitablity shock ϵ is realized.
- On the other hand, we define the **ex-ante value function** under belief σ as:

$$V(\sigma, s) = \mathbb{E}\{V(\sigma, s, \epsilon)|s\}. \tag{7.9}$$

7.2.8 Choice-specific Value Function

- When the current state and profitability shocks are s and ϵ and the belief about the future behavior is σ, we define the **choice-specific** value function for an agent in a period as follows:

$$\begin{aligned} V(\sigma, a, s, \epsilon) &= \pi(a, s) + \epsilon_a + \beta \sum_{s' \in S} V(\sigma, s') g(a, s, s') \\ &= \underbrace{\pi(a, s) + \beta \sum_{s' \in S} V(\sigma, s') g(a, s, s')}_{v(\sigma, a, s)} + \epsilon_a. \end{aligned} \tag{7.10}$$

- We call $v(\sigma, a, s)$ be the **choice-specific** mean value function with belief σ.
- $V(\sigma, s, \epsilon)$, $V(\sigma, s)$, $V(\sigma, a, s, \epsilon)$, and $v(\sigma, a, s)$ are all different, abusing the notation.

7.2.9 Optimality Condition

- When the state and profitability shocks are s and ϵ, a is the optimal choice if and only if:

$$v(\sigma, a, s) + \epsilon_a \geq v(\sigma, a', s) + \epsilon_{a'}, \forall a' \in A. \tag{7.11}$$

- This condition looks similar to the optimality condition in the static discrete choice model.
- The only difference from the static discrete choice model is that the mean indirect utility is the sum of the choice-specific mean profit and the discounted continuation value.
- Thus, as long as the mean choice-specific value function for given parameters can be computed, the following simulation and estimation procedure will be similar to the static discrete choice model.

7.2.10 Optimal Conditional Choice Probability

- From the previous optimality condition, we can define the **optimal conditional choice probability** with belief σ as:

$$\begin{aligned} p(a|s) &:= \mathbb{P}\{v(\sigma, a, s) + \epsilon_a \geq v(\sigma, a', s) + \epsilon_{a'}, \forall a' \in A\} \\ &= \int \prod_{a' \neq a} 1\{v(\sigma, a, s) + \epsilon_a \geq v(\sigma, a', s) + \epsilon_{a'}\} dF \\ &:= \Psi(\sigma, a, s). \end{aligned} \tag{7.12}$$

- $\Psi(\sigma, a, s)$ maps the tuple of action, state and belief to the optimal conditional choice probability of the action given the state and the belief.

7.2.11 Optimality Condition

- Let p and Ψ be:

$$p = \begin{pmatrix} p(0|1) \\ \vdots \\ p(K|1) \\ \vdots \\ p(0|L) \\ \vdots \\ p(K|L) \end{pmatrix}, \tag{7.13}$$

$$\Psi(\sigma) = \begin{pmatrix} \Psi(\sigma, 0, 1) \\ \vdots \\ \Psi(\sigma, K, 1) \\ \vdots \\ \Psi(\sigma, 0, L) \\ \vdots \\ \Psi(\sigma, K, L) \end{pmatrix}. \tag{7.14}$$

- The optimality condition with respect to the conditional choice probabilities given the belief is written as:

$$p = \Psi(\sigma). \tag{7.15}$$

- The rational expectation hypothesis requires that the belief about the future behavior coincides with the optimal conditional choice probability, i.e.:

$$p = \Psi(p). \tag{7.16}$$

- The optimal conditional choice probability under the rational expectation hypothesis is characterized as a fixed point of mapping Ψ.

7.2.12 Mapping from a Conditional Choice Probability to an Ex-ante Value Function

- Inserting $\sigma = p$, we obtain a mapping from an optimal conditional choice probability to an ex-ante value function such as:

$$\begin{aligned} V(s) &= \mathbb{E}\{V(s, \epsilon)|s\} \\ &= \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t \sum_{a \in A} p(a|s_t) \left[\pi(a, s_t) + \epsilon_{ta}\right] \middle| s_0, a\right] \\ &:= \varphi(p, s). \end{aligned} \tag{7.17}$$

7.2.13 Mapping from an Ex-ante Value Function to an Optimal Conditional Choice Probability

- On the other hand, we can derive a mapping from an ex-ante value function to an optimal conditional choice probability such as:

$$\begin{aligned} p(a|s) = \mathbb{P}\Bigg\{ & \pi(a,s) + \beta \sum_{s' \in S} V(s') g(a,s,s') + \epsilon_a \geq \\ & \pi(a',s) + \beta \sum_{s' \in S} V(s') g(a',s,s') + \epsilon_{a'}, \forall a' \in A \Bigg\} \\ & := \Lambda(V,a,s). \end{aligned} \tag{7.18}$$

7.2.14 The Optimality Conditions

- Composing these two mappings, we can write down the optimality condition as the fixed-point for ex-ante value functions:

$$V = \varphi(p) = \varphi[\Lambda(V)] := \Phi(V). \tag{7.19}$$

- Or as the fixed-point for the optimal conditional choice probabilities:

$$p = \Lambda(V) = \Lambda[\varphi(p)] := \Psi(p).$$

7.2.15 Fixed-point Algorithm

- If ϵ is drawn from an i.i.d. type-I extreme value distribution, we can derive the mapping from the ex-ante value function to the conditional choice probability in the closed form:

$$\begin{aligned} \Lambda(V,a,s) = \mathbb{P}\{ & \pi(a,s) + \beta \sum_{s' \in S} V(s') g(a,s,s') + \epsilon_a \geq \\ & \pi(a',s) + \beta \sum_{s' \in S} V(s') g(a',s,s') + \epsilon_{a'}, \forall a' \in A\} \\ & = \frac{\exp[\pi(a,s) + \beta \sum_{s' \in S} V(s') g(a,s,s')]}{\sum_{a' \in A} \exp[\pi(a',s) + \beta \sum_{s' \in S} V(s') g(a',s,s')]}. \end{aligned} \tag{7.20}$$

- Moreover, we can also derive the mapping from the ex-ante value function to the ex-ante value function as follows:

$$\begin{aligned} \Phi(V) &= \mathbb{E}\{\max_{a \in A} \pi(a,s) + \beta \sum_{s' \in S} V(s') g(a,s,s') + \epsilon_a\} \\ &= \log\left\{ \sum_{a \in A} \exp[\pi(a,s) + \beta \sum_{s' \in S} V(s') g(a,s,s')] \right\} + \gamma, \end{aligned} \tag{7.21}$$

  where γ is Euler's constant.
- Φ is shown to be a contraction mapping as long as $\beta < 1$.
- Thus, we can solve the model by starting from an arbitrary ex-ante value function V and by iterating the mapping Φ until the change in the ex-ante value functions is below a certain threshold.
- Rust [1987] gives the result for more general distributional assumption for ϵ.

7.3 Identification

7.3.1 Unidentification Result

- The identification of the single-agent dynamic decision model is studied in Magnac and Thesmar [2002].
- The model primitives are (Π, F, β, G).
- The transition probability G is directly identified from the data because a, s, s' are observed by econometrician.
- In the same manner, we can directly identify the optimal conditional choice probability p because a and s are observed by econometrician.
- It is known that the model is in general not identified.
- The discount factor β is hard to identify.
- It determines the weights between the current and future profits.
- Suppose that a firm makes a large investment.
- This may be because the firm overweights the future (high β) or because the investment cost is low (π is such that the investment cost is low).
- We cannot distinguish between these two possibilities.
- To identify it, you need some instruments that changes the future return to the investment but does not affect today's payoff.

7.3.2 Identification when β and F is Known

- We often fix β and assume the distribution F and only consider the identification of Π.
- Note that the optimal conditional choice probability is directly identified from the data because s and a are observed.
- Then the optimality condition under the rational expectation hypothesis gives the following KL system of equations:

$$p = \Psi(p).$$

- On the other hand, the dimension of parameter Π is in general $(K+1)L$ (the mean profit at a state and an action).
- One possible restriction is to assume that $\pi(0, s)$ are known for any s. For example, assume that $a = 0$ means that the firm is inactive and so $\pi(0, s) = 0$.

7.3.3 Crucial Assumptions for the Argument

- The following assumptions are crucial for the above argument.

1. **Conditional i.i.d. Unobservable**: The profit shocks that are unobservable to econometrician are i.i.d. conditional on the observable state.
2. **Conditional Independence of Future Observable State**:

$$\mathbb{P}\{s_{t+1}|s_t, a_t, \epsilon_t\} = \mathbb{P}\{s_{t+1}|s_t, a_t\}. \tag{7.22}$$

- If the first assumption is violated, the choice probability cannot be written as a function only of the observable state of the period. If ϵ_t is serially correlated, to integrate over ϵ_{t+1} we have to condition on ϵ_t. Then, we may not be able to identify the optimal conditional choice probability from the data/
- If the second assumption is violated, for the same reason, we may not be able to identify the state transition law.
- Kasahara and Shimotsu [2009] proves the identification when the first assumption is violated: there is a player-specific fixed effect that has a finite-mixture structure.

7.4 Estimation by Nested Fixed-point Algorithm

7.4.1 Nested Fixed-Point Algorithm

- A straightforward way of estimating the single-agent dynamic model is to solve the optimal conditional choice probability by a fixed-point algorithm for each parameter and evaluate the likelihood function using the optimal conditional choice probability.
- Because a fixed-point algorithm is nested in the parameter search, Rust [1987] named it the **nested fixed-point algorithm**.

7.4.2 Solving for the Ex-ante Value Function

- Let θ_1 be the parameters that determine Π, θ_2 be the parameters that determine G, and $\theta = (\theta_1', \theta_2')'$.
- The ex-ante value function is a fixed point of a contraction mapping Φ^θ such that:

$$\Phi^\theta(p) = \log\left\{\sum_{a\in A}\exp[\pi^{\theta_1}(a,s) + \beta\sum_{s'\in S}V(s')g^{\theta_2}(a,s,s')]\right\} + \gamma, \tag{7.23}$$

- Let $V^{\theta(0)}$ be an arbitrary initial function and define $V^{\theta(r+1)}$ by:

$$V^{\theta(r+1)} = \Phi^\theta(V^{\theta(r)}).$$

- We iterate this until $|V^{\theta(r+1)} - V^{\theta(r)}|$ is below a certain threshold.
- Let $V^{\theta(*)}$ be the solution to the fixed-point algorithm.
- Then, we can derive the optimal choice probability by:

$$p^{\theta(*)} = \Lambda\left[V^{\theta(*)}\right]. \tag{7.24}$$

- These are the ex-ante value function and optimal conditional choice probabilities under parameters θ.

7.4.3 Estimation by Nested Fixed-Point Algorithm

- The previous algorithm allows us to derive the optimal choice probability given parameters.
- Then it is straight forward to evaluate the likelihood function given observations $\{a_t, s_t\}_{t=1}^T$ by:

$$L(\theta; \{a_t, s_t\}_{t=1}^T) = \prod_{t=1}^T\prod_{a_t=0}^1 p^{\theta(*)}(a_t|s_t)^{a_t} g^{\theta_2}(s_{t+1}|s_t, a_t). \tag{7.25}$$

and so the log likelihood function is:

$$\begin{aligned} &l(\theta; \{a_t, s_t\}_{t=1}^T) \\ &= \sum_{t=1}^T\sum_{a_t=0}^1 a_t \log[p^{\theta(*)}(a_t|s_t)] + \sum_{t=1}^T \log[g^{\theta_2}(s_{t+1}|s_t, a_t)]. \end{aligned} \tag{7.26}$$

7.4.4 Full and Partial Likelihood

- We can find θ that maximizes the full log likelihood $l(\theta; \{a_t, s_t\}_{t=1}^T)$ to estimate the model.
- However, the convergence takes longer as the number of parameters are larger.
- Parameters that govern the state transition is estimated by finding θ_2 that maximizes the partial likelihood:

$$\hat{\theta}_2 = \text{argmax}_{\theta_2} \sum_{t=1}^{T} \log g^{\theta_2}(s_{t+1}|s_t, a_t). \tag{7.27}$$

- Then we can estimate θ_1 by finding θ_1 that maximizes the partial likelihood:

$$\hat{\theta}_1 = \text{argmax}_{\theta_1} \sum_{t=1}^{T} \sum_{a_t=0}^{1} a_t \log[p^{(\theta_1, \hat{\theta}_2)(*)}(a_t|s_t)]. \tag{7.28}$$

- This causes some efficiency loss but speeds up the computation, because we can estimate θ_2 without solving the fixed-point.

7.5 Estimation by Conditional Choice Probability (CCP) Approach

7.5.1 CCP Approach

- **Conditional Choice Probability (CCP)** approach suggested by Hotz and Miller [1993] significantly reduces the computation time at the cost of some efficiency.
- This approach can be applied to many other settings.
- The idea is:
  - We can identify the optimal conditional choice probability p^θ directly from the data. This is a reduced-form parameter (cf. θ is the structural parameters) of the model.
  - The optimality condition $p^\theta = \Psi^\theta(p^\theta)$ can be regarded as a moment condition.
- In the nested fixed-point algorithm, we find p^θ that solves the optimality condition given θ to compute the likelihood.
- In CCP approach, we find θ that solves the optimality condition given p^θ that is identified directly from the data.

7.5.2 First Step: Estimating CCP

- The first step of the CCP approach is to estimate the conditional choice probability and transition probability.
- If everything is discrete, it is nothing but the empirical distribution:

$$\begin{aligned} \hat{p}(a|s) &= \frac{\sum_{i=1}^N \sum_{t=1}^T 1\{a_{it} = a, s_{it} = s\}}{\sum_{i=1}^N \sum_{t=1} 1\{s_{it} = s\}}, \\ \hat{g}(s'|s, a) &= \frac{\sum_{i=1}^N \sum_{t=1}^T 1\{s_{i,t+1} = s', s_{it} = s, a_{it} = a\}}{\sum_{i=1}^N \sum_{t=1} 1\{s_{it} = s, a_{it} = a\}}. \end{aligned} \tag{7.29}$$

- We can of course use a parametric model.

- For example, we may estimate the conditional choice probability with a multinomial logit models:

$$\hat{p}(a|s) = \frac{\exp[\hat{\beta}a + \hat{\gamma}s)]}{\sum_{a' \in A} \exp[\hat{\beta}a' + \hat{\gamma}s)]}. \tag{7.30}$$

7.5.3 First Step: Estimating CCP

- What is the estimated CCP $\hat{p}$?
- This is the optimal conditional choice probability **at a particular equilibrium** under a true parameter.
- If parameter changes, then the equilibrium changes. Then, the conditional choice probability also changes.
- The reduced-form parameter $\hat{p}$ embodies the information about behaviors under the actual equilibrium but does not tell anything about behaviors under hypothetical equilibria.
- Therefore, $\hat{p}$ is not sufficient to make counterfactual prediction.

7.5.4 Second Step: Estimating Structural Parameters

- Among structural parameters θ, parameters in the transition probability θ_2 is already identified from the data in the first step.
- How do we identify θ_1, parameters in the profit function π?
- If we fix θ_1, in theory, we can compute:

$$\hat{V}^{(\theta_1,\hat{\theta}_2)}(s) = \varphi^{(\theta_1,\hat{\theta}_2)}(\hat{p}, s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t \sum_{a \in A} \hat{p}(a|s_t)\left(\pi^{\theta_1}(a, s_t) + \epsilon_{ta}\right)\middle| s\right], \tag{7.31}$$

although the expectation may not have a closed form solution.

7.5.5 Second Step: Estimating Structural Parameters

- In addition, if we fix θ_1, in theory, we can compute:

$$\begin{aligned}
&\Lambda^{(\theta_1,\hat{\theta}_2)}(\hat{V}^{(\theta_1,\hat{\theta}_2)}, a, s) \\
&:= \mathbb{P}\Bigg\{\pi^{\theta_1}(a, s) + \beta \sum_{s' \in S} \hat{V}^{(\theta_1,\hat{\theta}_2)}(s') g^{\hat{\theta}_2}(a, s, s') + \epsilon_a \\
&\geq \pi^{\theta_1}(a', s) + \beta \sum_{s' \in S} \hat{V}^{(\theta_1,\hat{\theta}_2)}(s') g^{\hat{\theta}_2}(a', s, s') + \epsilon_{a'}, \forall a' \in A\Bigg\}
\end{aligned} \tag{7.32}$$

- Combining these two mappings, we can compute:

$$\Psi^{(\theta_1,\hat{\theta_2})}(\hat{p}) = \Lambda^{(\theta_1,\hat{\theta}_2)}[\varphi^{(\theta_1,\hat{\theta}_2)}(\hat{p})]. \tag{7.33}$$

- Then, we can find θ_1 that minimizes the distance between $\hat{p}$ and $\Psi^{(\theta_1,\hat{\theta_2})}(\hat{p})$ to find θ_1 that is consistent with the observed conditional choice probabilities.

7.5.6 Type-I Extreme Value Distribution

- Λ and φ do no have closed form expressions in general.
- Exception is the case where the profitability shocks ϵ_{ta} is drawn from i.i.d. type-I extreme value distribution.
- First, we know that Λ can be written as:

$$\begin{aligned}
&\Lambda^{(\theta_1,\hat{\theta}_2)}(\hat{V}^{(\theta_1,\hat{\theta}_2)}, a, s) \\
&:= \mathbb{P}\Bigg\{\pi^{\theta_1}(a,s) + \beta \sum_{s' \in S} \hat{V}^{(\theta_1,\hat{\theta}_2)}(s') g^{\hat{\theta}_2}(a,s,s') + \epsilon_a \geq \\
&\pi^{\theta_1}(a',s) + \beta \sum_{s' \in S} \hat{V}^{(\theta_1,\hat{\theta}_2)}(s') g^{\hat{\theta}_2}(a',s,s') + \epsilon_{a'}, \forall a' \in A\Bigg\} \\
&= \frac{\exp\left[\pi^{\theta_1}(a,s) + \beta \sum_{s' \in S} \hat{V}^{(\theta_1,\hat{\theta}_2)}(s') g^{\hat{\theta}_2}(a,s,s')\right]}{\sum_{a'=0}^{K} \exp\left[\pi^{\theta_1}(a',s) + \beta \sum_{s' \in S} \hat{V}^{(\theta_1,\hat{\theta}_2)}(s') g^{\hat{\theta}_2}(a',s,s')\right]}.
\end{aligned} \tag{7.34}$$

- Second, we can show that φ has a closed form solution:

$$\begin{aligned}
&\varphi^{(\theta_1,\hat{\theta}_2)}(p,s) \\
&:= \mathbb{E}\{V(s,\epsilon)|s\} \\
&= \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t \sum_{a \in A} \hat{p}(a|s_t)\left(\pi^{\theta_1}(a,s_t) + \epsilon_{ta}\right)\middle| s\right] \\
&= \mathbb{E}\left[\sum_{a \in A} \hat{p}(a|s)\left(\pi^{\theta_1}(a,s) + \epsilon_a + \beta \sum_{s' \in S} \mathbb{E}\{\hat{V}^{(\theta_1,\hat{\theta}_2)}(s,\epsilon)|s'\} g^{\hat{\theta}_2}(a,s,s')\right)\middle| s\right] \\
&= \mathbb{E}\left[\sum_{a \in A} \hat{p}(a|s)\left(\pi^{\theta_1}(a,s) + \epsilon_a + \beta \sum_{s' \in S} \varphi^{(\theta_1,\hat{\theta}_2)}(\hat{p},s') g^{\hat{\theta}_2}(a,s,s')\right)\middle| s\right] \\
&= \sum_{a \in A} \hat{p}(a|s)\left(\pi^{\theta_1}(a,s) + \mathbb{E}[\epsilon_a|s,a] + \beta \sum_{s' \in S} \varphi^{(\theta_1,\hat{\theta}_2)}(\hat{p},s') g^{\hat{\theta}_2}(a,s,s')\right)
\end{aligned} \tag{7.35}$$

- We need closed form expression of $\mathbb{E}[\epsilon_a|s,a]$: the expected value of choice$-a$ specific profitability shock **conditional on that state is s and a is optimal** ($\neq$ unconditional mean of ϵ_a).

7.5.7 Type-I Extreme Value Distribution

- If ϵ_a is drawn from i.i.d. type-I extreme value distribution, it can be shown that:

$$\begin{aligned}
\mathbb{E}[\epsilon_a|s,a] = \hat{p}(a|s)^{-1} \int \epsilon_a 1\Bigg\{&\pi^{\theta_1}(a,s) + \beta \sum_{s' \in S} \hat{V}^{(\theta_1,\hat{\theta}_2)}(s') g^{\hat{\theta}_2}(a,s,s') + \epsilon_a \geq \\
&\pi^{\theta_1}(a',s) + \beta \sum_{s' \in S} \hat{V}^{(\theta_1,\hat{\theta}_2)}(s') g^{\hat{\theta}_2}(a',s,s') + \epsilon_{a'}, \forall a' \in A\Bigg\} dF(e) \\
&= \gamma - \ln \hat{p}(a|s),
\end{aligned} \tag{7.36}$$

where γ is Euler's constant:

$$\gamma := \lim_{n\to\infty}\left(\sum_{k=1}^{n}\frac{1}{k} - \ln(n)\right) \approx 0.57721... \tag{7.37}$$

7.5.8 Type-I Extreme Value Distribution

- Inserting this into the previous expression, we get:

$$\varphi^{(\theta_1,\hat{\theta}_2)}(\hat{p},s) = \sum_{a\in A}\hat{p}(a|s)\left(\pi^{\theta_1}(a,s) + \gamma - \ln\hat{p}(a|s) + \beta\sum_{s'\in S}\varphi^{(\theta_1,\hat{\theta}_2)}(\hat{p},s')g^{\hat{\theta}_2}(a,s,s')\right). \tag{7.38}$$

7.5.9 Type-I Extreme Value Distribution

- Write the continuation value in a matrix form:

$$\begin{aligned}
&\sum_{s'\in S}\varphi^{(\theta_1,\hat{\theta}_2)}(p,s')g^{\hat{\theta}_2}(a,s,s') \\
&= [g^{\hat{\theta}_2}(a,s,1),\cdots,g^{\hat{\theta}_2}(a,s,L)]\underbrace{\begin{bmatrix}\varphi^{(\theta_1,\hat{\theta}_2)}(p,1)\\ \vdots \\ \varphi^{(\theta_1,\hat{\theta}_2)}(p,L).\end{bmatrix}}_{:=\varphi^{(\theta_1,\hat{\theta}_2)}(p)}
\end{aligned} \tag{7.39}$$

7.5.10 Type-I Extreme Value Distribution

- Write the ex-ante value function in a matrix form:

$$\begin{aligned}
&\varphi^{(\theta_1,\hat{\theta}_2)}(p,s) \\
&= \underbrace{[p(0|s),\cdots,p(K|s)]}_{:=p(s)'} \\
&\times\left[\underbrace{\begin{bmatrix}\pi^{\theta_1}(0,s)\\ \vdots \\ \pi^{\theta_1}(K,s)\end{bmatrix}}_{:=\pi^{\theta_1}(s)} + \gamma - \underbrace{\begin{bmatrix}\ln p(0|s)\\ \vdots \\ \ln p(K|s)\end{bmatrix}}_{:=\ln p(s)} + \beta\underbrace{\begin{bmatrix}g^{\hat{\theta}_2}(0,s,1),\cdots,g^{\hat{\theta}_2}(0,s,L)\\ \vdots \\ g^{\hat{\theta}_2}(K,s,1),\cdots,g^{\hat{\theta}_2}(K,s,L)\end{bmatrix}}_{:=G^{\hat{\theta}_2}(s)}\varphi^{(\theta_1,\hat{\theta}_2)}(p)\right] \\
&= p(s)'[\pi^{\theta_1}(s) + \gamma - \ln p(s)] + \beta p(s)'G^{\hat{\theta}_2}(s)\varphi^{(\theta_1,\hat{\theta}_2)}(p)
\end{aligned} \tag{7.40}$$

- Stacking up for s, we get:

$$\varphi^{(\theta_1,\hat{\theta}_2)}(p) = \begin{bmatrix} p(1)'[\pi^{\theta_1}(1) + \gamma - \ln p(1)] \\ \vdots \\ p(L)'[\pi^{\theta_1}(L) + \gamma - \ln p(L)] \end{bmatrix} + \beta \begin{bmatrix} p(1)'G^{\hat{\theta}_2}(1) \\ \vdots \\ p(L)'G^{\hat{\theta}_2}(L) \end{bmatrix} \varphi^{(\theta_1,\hat{\theta}_2)}(p)$$

$$\Leftrightarrow \tag{7.41}$$

$$\varphi^{(\theta_1,\hat{\theta}_2)}(p) = \left[I - \beta \begin{bmatrix} p(1)'G^{\hat{\theta}_2}(1) \\ \vdots \\ p(L)'G^{\hat{\theta}_2}(L) \end{bmatrix} \right]^{-1} \begin{bmatrix} p(1)'[\pi^{\theta_1}(1) + \gamma - \ln p(1)] \\ \vdots \\ p(L)'[\pi^{\theta_1}(L) + \gamma - \ln p(L)] \end{bmatrix}.$$

- Note that you can get this expression even if the profitability shocks are not type-I extreme value, although you need numerical integration for $\mathbb{E}\{\epsilon_a|s,a\}$ instead of the analytical solution $\gamma - \ln p(a|s)$.
- Let:

  $$\Sigma(p) = \begin{pmatrix} p(1)' & & \\ & \ddots & \\ & & p(L)' \end{pmatrix}$$

  and:

  $$E(p) = \gamma - \ln p,$$

  we can have a matrix representation:

  $$\varphi^{(\theta_1,\hat{\theta}_2)}(p) = [I - \beta\Sigma(p)G]^{-1}\Sigma(p)[\Pi + E(p)].$$

7.5.11 General Distribution

- If the profitability shock ϵ_a is not an i.i.d. type-I extreme value random variable, you may need to compute $\mathbb{E}\{\epsilon_a|s,a\}$ and $\Lambda^{(\theta_1,\hat{\theta}_2)}(V)$ numerically.
- This may or may not feasible.

7.6 Unobserved Heterogeneity

7.6.1 Dynamic Decision Model with a Finite Mixture

- Decision makers such as a firm, a worker, and a consumer can be different in an unobserved manner.
- Finite mixture models is restrictive yet flexible enough modeling framework of unobserved heterogeneity.
- Kasahara and Shimotsu [2009] consider a dynamic decision model with a finite mixture and provide a sufficient condition for identification.

7.6.2 Setting

- Each period, each player makes a choice a_t from a discrete and finite set A conditioning on $(x_t, x_{t-1}, a_{t-1}) \in X \times X \times A$ (being allowed to depend on x_{t-1} and a_{t-1}).
- x_t is observable individual characteristics that can change over time.
- Each player belongs to one of M types.
- For example, parameters are different across types.
- The probability of belonging to type m is denoted by π^m such that $\sum_{m=1}^{M} \pi^m = 1$.

- Type m's conditional choice probability: $P_t^m(a_t|x_t, x_{t-1}, a_{t-1})$.
- Type m's initial probability of (x_1, a_1): $p^{*m}(x_1, a_a)$.
- Type m's transition probability of x_t: $f_t^m(x_t|\{x_\tau, a_\tau\}_{\tau=1}^{t-1})$ (being allowed to depend on the entire history).

7.6.3 Observation

- We have a panel data set with time-deimension equal to T.
- Each player's observation $w_i = \{a_{it}, x_{it}\}_{t=1}^T$ is drawn randomly from an M-term mixture distribution such as:

$$\begin{aligned} P(\{a_t, x_t\}_{t=1}^T) &= \sum_{m=1}^M \pi^m p^{*m}(x_1, a_1) \prod_{t=2}^T f_t^m(x_t|\{x_\tau, a_\tau\}_{\tau=1}^{t-1}) P_t^m(a_t|x_t, \{x_\tau, a_\tau\}_{\tau=1}^{t-1}) \\ &= \sum_{m=1}^M \pi^m p^{*m}(x_1, a_1) \prod_{t=2}^T f_t^m(x_t|\{x_\tau, a_\tau\}_{\tau=1}^{t-1}) P_t^m(a_t|x_t, x_{t-1}, a_{t-1}), \end{aligned}$$

  where the second equality uses the Markovian assumption on the conditional choice probability.

7.6.4 Further Assumptions

- We start from a model with the following simplifying assumptions, which are often imposed in an applied work.

1. The choice probability of a_t does not depend on time: $P_t^m(a_t|x_t, x_{t-1}, a_{t-1}) = P^m(a_t|x_t, x_{t-1}, a_{t-1})$. for all t.
2. The choicd probability of a_t does not depend on x_{t-1}, a_{t-1}: $P^m(a_t|x_t, x_{t-1}, a_{t-1}) = P^m(a_t|x_t)$.
3. $f_t^m(x_t|\{x_\tau, a_\tau\}_{\tau=1}^{t-1}) > 0$ for all $(x_t, \{x_\tau, a_\tau\}_{\tau=1}^{t-1})$ and all m.
4. The transition function is common across types: $f_t^m(x_t|\{x_\tau, a_\tau\}_{\tau=1}^{t-1}) = f_t(x_t|\{x_\tau, a_\tau\}_{\tau=1}^{t-1})$ for all m.
5. The transition function does not depend on time: $f_t(x_t|\{x_\tau, a_\tau\}_{\tau=1}^{t-1}) = f(x_t|x_{t-1}, a_{t-1})$ for all t.

- Then the probability of an observation becomes:

$$P(\{a_t, x_t\}_{t=1}^T) = \sum_{m=1}^M \pi^m p^{*m}(x_1, a_1) \prod_{t=2}^T f(x_t|x_{t-1}, a_{t-1}) P^m(a_t|x_t).$$

7.6.5 Lower-dimensional Submodels

- Because $f(x_t|x_{t-1}, a_{t-1})$ is non-parametrically identified from data, we are only concerned with identification of type probabilities and conditional choice probabilities.
- Transform the previous equation as:

$$\begin{aligned} \widetilde{P}(\{a_t, x_t\}_{t=1}^T) &:= \frac{P(\{a_t, x_t\}_{t=1}^T)}{\prod_{t=2}^T f(x_t|x_{t-1}, a_{t-1})} \\ &= \sum_{m=1}^M \pi^m p^{*m}(x_1, a_1) \prod_{t=2}^T P^m(a_t|x_t). \end{aligned}$$

- Let $\mathcal{I} := \{i_1, \cdots, i_l\} \subset \{1, \cdots, T\}$ be a subset of time indices.
- We define a **lower-dimensional submodels** given $\mathcal{I}$ as:

$$\widetilde{P}(\{a_{i_s}, x_{i_s}\}_{i_s \in \mathcal{I}}) = \sum_{m=1}^M \pi^m p^{*m}(x_1, a_1) \prod_{s=2}^l P^m(a_t|x_t),$$

if $1 \in \mathcal{I}$ and:

$$\widetilde{P}(\{a_{i_s}, x_{i_s}\}_{i_s \in \mathcal{I}}) = \sum_{m=1}^{M} \pi^m \prod_{s=2}^{l} P^m(a_t|x_t),$$

if $1 \notin \mathcal{I}$.

- Under each different value of $(x_1, \cdots, x_T)$, above equations imply different restrictions on the type probabilities and conditional choice probabilities.
- There are the order of $|X|^T$ variations in $(x_1, \cdots, x_T)$, whereas the number of parameters $\{\pi^m, p^{*m}(a, x), P^m(a|X)\}_{m=1}^M$ is of a order of $|X|$.

7.6.6 Notations

- For notational simplicity, consider a case with $A = \{0, 1\}$.
- Define, for $\xi \in X$:

$$\lambda_\xi^{*m} := p^{*m}[(a_1, x_1) = (1, \xi)],$$
$$\lambda_\xi^m := P^m(a = 1|x = \xi).$$

7.6.7 Notations for Parameters to be Identified

- Let $\xi_j, j = 1, \cdots, M-1$ be elements of X and k be an element of X.
- Define a matrix of type-specific distribution functions and type probabilities as:

$$L := \begin{pmatrix} 1 & \lambda_{\xi_1}^1 & \cdots & \lambda_{\xi_{M-1}^1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_{\xi_1}^M & \cdots & \lambda_{\xi_{M-1}^M} \end{pmatrix},$$

$$D_k^* := \begin{pmatrix} \lambda_k^{*1} & & \\ & \ddots & \\ & & \lambda_k^{*M} \end{pmatrix},$$

and

$$V := \begin{pmatrix} \pi^1 & & \\ & \ddots & \\ & & \pi^M \end{pmatrix}.$$

7.6.8 Notations for Observables

- Define for every (x_1, x_2, x_3):

$$F_{x_1,x_2,x_3}^* := \widetilde{P}(\{1, x_t\}_{t=1}^3) = \sum_{m=1}^{M} \pi^m \lambda_{x_1}^{*m} \lambda_{x_2}^m \lambda_{x_3}^m.$$

- Define for every (x_2, x_3):

$$F_{x_2,x_3} := \widetilde{P}(\{1, x_t\}_{t=2}^3) = \sum_{m=1}^{M} \pi^m \lambda_{x_2}^m \lambda_{x_3}^m.$$

- In the same way, define F_{x_1,x_2}^* and f_{x_1,x_3}^*.
- Define for every x_1:

$$F_{x_1}^* := \widetilde{P}(\{1, x_1\}) = \sum_{m=1}^{M} \pi^m \lambda_{x_1}^{*m}.$$

- In the same way, define F_{x_2} and F_{x_3}.
- In the notations above, F^* involves (a_1, x_1) and F does not.
- Summing up probabilities that do not involve x_1 as:

$$P := \begin{pmatrix} 1 & F_{\xi_1} & \cdots & F_{\xi_{M-1}} \\ F_{\xi_1} & F_{\xi_1,\xi_1} & \cdots & F_{\xi_1,\xi_{M-1}} \\ \vdots & \vdots & \ddots & \vdots \\ F_{\xi_{M-1}} & F_{\xi_{M-1},\xi_1} & \cdots & F_{\xi_{M-1},\xi_{M-1}} \end{pmatrix}$$

- Summing up probabilities that involve x_1 as:

$$P^* := \begin{pmatrix} k & F^*_{k,\xi_1} & \cdots & F^*_{k,\xi_{M-1}} \\ F^*_{k,\xi_1} & F^*_{k,\xi_1,\xi_1} & \cdots & F^*_{k,\xi_1,\xi_{M-1}} \\ \vdots & \vdots & \ddots & \vdots \\ F^*_{k,\xi_{M-1}} & F^*_{k,\xi_{M-1},\xi_1} & \cdots & F^*_{k,\xi_{M-1},\xi_{M-1}} \end{pmatrix}.$$

7.6.9 Sufficient Condition for Identification

- **Identification theorem**:
- Suppose that assumptions in 7.6.4 hold.
- $T \geq 3$.
- There exist some $\{\xi_1, \cdots, \xi_{M-1}\} \in X^{M-1}$ such that L is non-singular.
- There exists $k \in X$ such that $\lambda_k^{*m} > 0$ for all $m = 1, \cdots, M$ and $\lambda_k^{*m} \neq \lambda_k^{*n}$ for any $m \neq n$.
- Then, $\{\pi^m, \{\lambda_\xi^{*m}, \lambda_\xi^m\}_{\xi \in X}\}_{m=1}^M$ is uniquely identified from $\{\widetilde{P}(\{a_t, x_t\}_{t=1}^3)\}$.
- Because the assumptions of the above theorem refer to model parameters, the authors also derive sufficient conditions based on observables.
- **Corollary**:
- Suppose that assumptions in 7.6.4 hold.
- $T \geq 3$.
- There exist some $\{\xi_1, \cdots, \xi_{M-1}\} \in X^{M-1}$ and $k \in X$ such that P is of full rank and that all the eigenvalues of $P^{-1}P_k^*$ take distinct values.
- Then, $\{\pi^m, \{\lambda_\xi^{*m}, \lambda_\xi^m\}_{\xi \in X}\}_{m=1}^M$ is uniquely identified from $\{\widetilde{P}(\{a_t, x_t\}_{t=1}^3)\}$.

7.6.10 Remarks on the Theorem

- The condition says that L is non-singular.
  - This implies that all columns in L must be linearly independent.
  - In other words, the changes in covariate x must induce sufficient heterogeneous variations in the conditional choice probabilities across types.
- The conditions says $\lambda_k^{*m} > 0$ for all m.
  - If there is some m that is $\lambda_k^m = 0$ for any k, such type never shows up in the data.
- The condition says $\lambda_k^{*m} \neq \lambda_k^{*n}$ for any $m \neq n$.
  - This condition is satisfied if initial distribution is different across types.
  - If this condition fails, the identification becomes severe.
  - Actually, $T = 3$ is not enough and $T \geq 4$ becomes necessary.
- The identification only requires one set of $M - 1$ points $\xi_1, \cdots, \xi_{M-1}$ that satisfy the condition.
  - Information from other points provide overidentifying restrictions.

7.6.11 Factorization Equations

- Parameters L, V, D_k^* and data P, P_k^* are related through the following **factorization equations** (check manually):
$$P = L'VL,$$
$$P_k^* = L'D_k^*VL.$$
- Note that $(1,1)$-th element of $P = L'VL$ is $\sum_{m=1}^{M} \pi^m = 1$ and give no information.

7.6.12 Sketch of the Proof

- Suppose that P is invertible or equivalently L is invertible.
- Then, we have:
$$P^{-1} = L^{-1}V^{-1}L^{'-1}.$$
- Therefore, we have:
$$\begin{aligned} P^{-1}P_k^* &= L^{-1}V^{-1}L^{'-1}L'D_k^*VL \\ &= L^{-1}V^{-1}D_k^*VL \\ &= L^{-1}D_k^*L, \end{aligned}$$
where the third equality is because V and D_k^* are diagonal matrices.
- This equation means that D_k^* is identified as the matrix of eigenvalues of $P^{-1}P_k^*$.
- Moreover, the columns of L^{-1} are identified as the eigen vectors of $P^{-1}P_k^*$.
- Finally, $V = L^{'-1}PL^{-1}$ is identified because the right-hand side is now known.

7.6.13 Constructive Estimation

- According to the above identification argument, we can consider a following constructive estimation procedure.

1. Estimate P and P_k^* non-parametrically.
2. By applying an eigenvalue decomposition algorithm to $P^{-1}P_k^*$, identify D_k^* and L.
3. Then identify V.
4. Once conditional choice probability is identified, we can use the standard estimation method based on the CCP approach to each type.

7.6.14 Estimation by an EM Algorithm

- Arcidiacono and Miller [2011] suggest to use an EM algorithm to estimate a dynamic decision model with unobserved heterogeneity.
- Return to our original notation, and suppose that state s_t is partitioned into $s_t := (x_t, w_t)$, where x_t is observed but w_t is unobserved to an econometrician.
- Suppose that the transition probability is such that:
$$\begin{aligned} \mathbb{P}\{s_{t+1}|s_t, a_t\} &= \mathbb{P}\{x_{t+1}|x_t, w_t, a_t\}\mathbb{P}\{w_{t+1}|w_t\} \\ &:= g(x_{t+1}|x_t, w_t, a_t)h(w_{t+1}|w_t). \end{aligned}$$
- The initial distribution of w_1 is $h(w_1|x_1)$.
- We **assume** that the model is identified and only consider estimation.
- For example, in the previous finite-mixture model, we considered a case with $h(w_{t+1} = w'|w_t = w) = 1\{w' = w\}$ and provided a sufficient condition for identification.
- Let θ_1 be the parameters in π and θ_2 be the parameters in g.
- Let $\theta = (\theta_1, \theta_2)$.

7.6.15 Idea

- Let $q_{it}(w)$ be the probability that firm i is in unobserved state w in time t.
- The idea of the EM algorithm is:

1. **Expectation step**: Given a parameter $\theta^{(r)}$, a conditional choice probability $p^{(r)}(a|x,w)$, an initial distribution of the unobserved state variable $h^{(r)}(w_1|x_1)$, and a transition probability of the unobserved state variable $h^{(r)}(w'|w)$, update the probability that firm i is in unobserved state w in time t to $q_{it}^{(r+1)}(w)$, update the initial distribution of the unobserved state variable to $h^{(r+1)}(w|x_1)$, update the transition probability of the unobserved state variable to $h^{(r+1)}(w'|w)$, and update the conditional choice probability to $p^{(r+1)}(a|x,w)$.
2. **Maximization step**: Given the probability that firm i is in unobserved state w in time t to $q_{it}^{(r+1)}(w)$, the initial distribution of the unobserved state variable to $h^{(r+1)}(w|x_1)$, the transition probability of the unobserved state variable to $h^{(r+1)}(w'|w)$, and the conditional choice probability to $p^{(r+1)}(a|x,w)$, update the parameters to $\theta^{(r+1)}$.

- And continue this until a convergence.
- By doing so, we avoid integrating out the unobserved state variables to evaluate the likelihood function.

7.6.16 Optimal Conditional Choice Probability

- We can still define the optimal conditional choice probability given a choice probability in the future as $\varphi^{\theta,h}(p)$ because we just assumed that some of the state is not observed to an econometrician.
- We just divided parameters in G into θ_2 and h.

7.6.17 Likelihood Functions

- The following likelihood functions can be calculated if we know θ, h, and p.
- The likelihood of observing $\{a_{it}, x_{i,t+1}\}$ conditional on x_{it} and w_{it} is:

$$L(a_{it}, x_{i,t+1}|x_{it}, w_{it}; \theta, h, p) := \varphi^{\theta,h}(p)(a_{it}|x_{it}, w_{it}) g^{\theta_2}(x_{i,t+1}|x_{it}, w_{it}, a_{it}).$$

- The likelihood of observing $\{a_{it}, x_{it}\}_{t=1}^T$ conditional on x_{i1} is:

$$\begin{aligned} L(\{a_{it}, x_{it}\}_{t=1}^T|x_{i1}, \theta, h, p) := &\sum_{w_{i1}=1}^W \cdots \sum_{w_{iT}=1}^W h(w_{i1}|x_{i1}) L(a_{i1}, x_{i,2}|x_{i1}, w_{i1}; \theta, h, p) \\ &\times \prod_{t=2}^T h(w_{i,t+1}|w_{it}) L(a_{it}, x_{i,t+1}|x_{it}, w_{it}; \theta, h, p). \end{aligned}$$

- The likelihood of having w_{it} in period t and having $\{a_{it}, x_{it}\}_{t=1}^T$ conditional on x_{i1} is:

$$\begin{aligned} L(w_{it}, \{a_{it}, x_{it}\}_{t=1}^T|x_{1t}, \theta, h, p) := &\sum_{w_{i1}=1}^W \cdots \sum_{w_{i,t-1}=1}^W \sum_{w_{i,t+1}=1}^W \cdots \sum_{w_{iT}=1}^W h(w_{i1}|x_{i1}) L(a_{i1}, x_{i,2}|x_{i1}, w_{i1}; \theta, h, p) \\ &\times \prod_{t=2}^T h(w_{i,t+1}|w_{it}) L(a_{it}, x_{i,t+1}|x_{it}, w_{it}; \theta, h, p). \end{aligned}$$

- The likelihood of having w_{it} in period t conditional on $\{a_{it}, x_{it}\}_{t=1}^T$ is:

$$L(w_{it}|\{a_{it}, x_{it}\}_{t=1}^T, \theta, h, p) := \frac{L(w_{it}, \{a_{it}, x_{it}\}_{t=1}^T|x_{1t}, \theta, h, p)}{L(\{a_{it}, x_{it}\}_{t=1}^T|x_{1t}, \theta, h, p)}.$$

7.6.18 Expectation Step

- We have a parameter $\theta^{(r)}$, a conditional choice probability $p^{(r)}(a|x,w)$, an initial distribution of the unobserved state variable $h^{(r)}(s_1|x_1)$, and a transition probability of the unobserved state variable $h^{(r)}(w'|w)$.

1. Update the probability that firm i is in unobserved state w in time t to $q_{it}^{(r+1)}(w)$:

$$q_{it}^{(r+1)}(w) := L(w|\{a_{it}, x_{it}\}_{t=1}^{T}, \theta^{(r)}, h^{(r)}, p^{(r)}).$$

2. Update the initial distribution of the unobserved state variable to $h^{(r+1)}(w|x_1)$:

$$h^{(r+1)}(w|x_1) := \frac{\sum_{i=1}^{N} 1\{x_{i1} = x_1\} q_{i1}^{(r+1)}(w)}{\sum_{i=1}^{N} 1\{x_{i1} = x_1\}}.$$

3. Update the transition probability of the unobserved state variable to $h^{(r+1)}(w'|w)$:

$$h^{(r+1)}(w'|w) := \frac{\sum_{i=1}^{N}\sum_{t=2}^{T} q_{i,t-1}^{(r+1)}(w) q_{it}^{(r+1)}(w')}{\sum_{i=1}^{N}\sum_{t=2}^{T} q_{i,t-1}^{(r+1)}(w)}.$$

4. Update the conditional choice probability to $p^{(r+1)}(a|x,w)$:

$$p^{(r+1)}(a|x,w) := \frac{\sum_{i=1}^{N}\sum_{t=1}^{T} q_{it}^{(r+1)}(w) 1\{x_{it} = x\} 1\{a_{it} = a\}}{\sum_{i=1}^{N}\sum_{t=1}^{T} q_{it}^{(r+1)}(w) 1\{x_{it} = x\}}.$$

7.6.19 Maximization Step

- We have the probability that firm i is in unobserved state w in time t to $q_{it}^{(r+1)}(w)$, the initial distribution of the unobserved state variable to $h^{(r+1)}(w|x_1)$, the transition probability of the unobserved state variable to $h^{(r+1)}(w'|w)$, and the conditional choice probability to $p^{(r+1)}(a|x,w)$.
- Update parameters to $\theta^{(r+1)}$:

$$\theta^{(r+1)} := \text{argmax}_{\theta} \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{w=1}^{W} q_{it}^{(r+1)}(w) \ln L(a_{it}, x_{i,t+1}|x_{it}, w_{it}; \theta, h^{(r+1)}, p^{(r+1)}).$$

- The maximization step can be either of a nested-fixed point algorithm or CCP approach.

Chapter 8

Dynamic Game

8.1 Multiple-Agent Model

- There are multiple players in an industry.
- There is a strategic interaction across players.

8.1.1 Setting

- There is $i = 1, \cdots, N$ players.
- Time is discrete $t = 1, \cdots, \infty$.
- There are $K + 1$ actions for each player $A_i = \{0, 1, \cdots, K\}$.
- $A = \prod_{i=1}^{N} A_i$. $m_a = (K + 1)^N$.
- There are L states for each player $S_i = \{1, \cdots, L\}$.
- $S = \prod_{i=1}^{N} S_i$. $m_s = L^N$.

8.1.2 Timing of the Model

- At period t:
- State $s_t = (s_{t1}, \cdots, s_{tN})' \in S$ is publicly observed.
- Choice-specific profitability shocks $\epsilon_{ti} \in \mathbb{R}^{K+1}$ are drawn from $F(\cdot|s_{ti,t,-i})$ i.i.d. and privately observed by player i for each $i = 1, \cdots, N$.
- ϵ_{ti} is independent across i.
- Actions $a_{ti} \in A_i, i = 1, \cdots, N$ are chosen simultaneously.
- The state evolves according to a transition probability:

$$g(a, s, s') := \mathbb{P}\{s_{t+1} = s' | s_t = s, a_t = a\}, \tag{8.1}$$

$$G := \begin{pmatrix} g(1,1,1) & \cdots & g(1,1,m_s) \\ \vdots & & \vdots \\ g(m_a,1,1) & \cdots & g(m_a,1,m_s) \\ & \vdots & \\ g(1,m_s,1) & \cdots & g(1,m_s,m_s) \\ \vdots & & \vdots \\ g(m_a,m_s,1) & \cdots & g(m_a,m_s,m_s) \end{pmatrix}. \tag{8.2}$$

8.1.3 Period Profit

- When the state is s_t, action profile is a_t, and the profitability shocks are ϵ_t, the period payoff of player i is:

$$\pi_i(a_t, s_t) + \sum_{k=0}^{K} \epsilon_{tik} 1\{a_{ti} = k\}, i = 1, \cdots, N, \tag{8.3}$$

  where $\pi_i(a_t, s_t)$ is the mean period profit of player i.

- Let:

$$\epsilon_{tia_{ti}} = \sum_{k=0}^{K} \epsilon_{tik} 1\{a_{ti} = k\}$$

  be the choice-specific profitability shock.

- Let Π_i summarize the choice-specific mean period payoffs at each state:

$$\Pi_i := \begin{pmatrix} \pi_i(1,1) \\ \vdots \\ \pi_i(m_a, 1) \\ \vdots \\ \pi_i(1, m_s) \\ \vdots \\ \pi_i(m_a, m_s) \end{pmatrix}. \tag{8.4}$$

- The profit of player i is the discounted sum of future payoffs with discount factor $\beta < 1$.

8.1.4 Belief

- Let $\sigma_i(a|s)$ be the player i's belief about the possibility of having action profile a when the realized state is s, which may or may not coincide with the equilibrium probability.
- Let σ_i stack them up as:

$$\sigma_i(s) = \begin{pmatrix} \sigma_i(1|s) \\ \vdots \\ \sigma_i(m_a|s), \end{pmatrix}$$

$$\sigma_i = \begin{pmatrix} \sigma_i(1) \\ \vdots \\ \sigma_i(m_s) \end{pmatrix},$$

  and let σ be:

$$\sigma = \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_N \end{pmatrix}.$$

8.1.5 Markov Perfect Equilibrium

- A collection $(a, \sigma) = (a_1, \cdots, a_N, \sigma_1, \cdots, \sigma_N)$ is a pure-strategy **Markov perfect equilibrium** if:
  - All players use Markovian strategies;
  - For all i, a_i is a best response to a_{-i} given the belief σ_i at all state $s \in S$;
  - For all i, the belief σ_i is consistent with the strategy a.

8.1.6 Ex-ante Value Function

- When the belief of player i about the future behavior is σ_i, then the ex-ante value function associated with this belief is:

$$\begin{aligned} V_i(\sigma_i, s) = &\sum_{a \in A} \sigma_i(a|s)[\pi_i(a, s) + \beta \sum_{s' \in S} g(a, s, s') V_i(\sigma_i, s')] \\ &+ \sum_{k=0}^{K} \mathbb{E}\{\epsilon_i^k | a_i = k, s\} \sigma_i(a_i = k|s). \end{aligned} \tag{8.5}$$

- By stacking up over the state, we obtain a matrix representation:

$$V_i(\sigma_i) = \Sigma_i(\sigma_i)\Pi_i + \Sigma_i(\sigma_i)E_i(\sigma_i) + \beta\Sigma_i(\sigma_i)GV_i(\sigma_i). \tag{8.6}$$

where

$$\Sigma_i(\sigma_i) = \begin{pmatrix} \sigma_i(1)' & & \\ & \ddots & \\ & & \sigma_i(m_s)' \end{pmatrix}, \tag{8.7}$$

and

$$D_i(\sigma_i) = \begin{pmatrix} \sum_{k=0}^{K} \mathbb{E}\{\epsilon_i^k | a_i = k, 1\} \sigma_i(a_i = k|1) \\ \vdots \\ \sum_{k=0}^{K} \mathbb{E}\{\epsilon_i^k | a_i = k, m_s\} \sigma_i(a_i = k|m_s) \end{pmatrix}. \tag{8.8}$$

- If $I - \beta\Sigma_i(\sigma_i)G$ is invertible, we have:

$$V_i(\sigma_i) = [I - \beta\Sigma_i(\sigma_i)G]^{-1}[\Sigma_i(\sigma_i)\Pi_i + D_i(\sigma_i)].$$

8.1.7 Choice-specific Value Function

- When the state is s and the belief of player i is σ_i, then the mean value of choosing action a_i is:

$$\begin{aligned} v_i(\sigma_i, a_i, s) = &\sum_{a_{-i} \in A_{-i}} \sigma_i(a_{-i}|s)[\pi_i(a_i, a_{-i}, s) \\ &+ \beta \sum_{s' \in S} g(a_i, a_{-i}, s, s') V_i(\sigma_i, s')]. \end{aligned} \tag{8.9}$$

- This is the choice-specific mean value function given belief σ_i.

8.1.8 Optimality Condition

- When the state is s and profitability shock is ϵ_i, a_i is optimal for player i under belief σ_i if and only if:

$$v_i(\sigma_i, a_i, s) + \epsilon_{i,a_i} \geq v_i(\sigma_i, a_i', s) + \epsilon_{i,a_i'}, \forall a_i' \in A_i. \tag{8.10}$$

- This condition is similar to the optimality condition in single-agent dynamic models.
- Only difference is that the belief is now about the others' actions in addition to about own future behaviors.

8.1.9 Optimal Conditional Choice Probability

- Therefore, the optimal conditional choice probability of player i with belief σ_i is:

$$\begin{aligned} p(a_i|s) &= \mathbb{P}\{v_i(\sigma_i, a_i, s) + \epsilon_{i,a_i} \geq v_i(\sigma_i, a_i', s) + \epsilon_{i,a_i'}, \forall a_i' \in A_i\} \\ &= \int \prod_{a_i' \neq a_i} 1\{v_i(\sigma_i, a_i, s) + \epsilon_{i,a_i} \geq v_i(\sigma_i, a_i', s) + \epsilon_{i,a_i'}\} dF \\ &:= \Psi(\sigma_i, a_i, s). \end{aligned} \tag{8.11}$$

8.1.10 Equilibrium Condition

- Let:

$$p(a|s) = \prod_{i=1}^{N} p(a_i|s).$$

and

$$\Psi(\sigma, a, s) = \prod_{i=1}^{N} \Psi(\sigma_i, a_i, s).$$

- Stacking them up as:

$$p(s) = \begin{pmatrix} p(1|s) \\ \vdots \\ p(m_a|s) \end{pmatrix},$$

$$p = \begin{pmatrix} p(1) \\ \vdots \\ p(m_s) \end{pmatrix},$$

$$\Psi(\sigma, s) = \begin{pmatrix} \Psi(\sigma, 1, s) \\ \vdots \\ \Psi(\sigma, m_a, s) \end{pmatrix},$$

$$\Psi(\sigma) = \begin{pmatrix} \Psi(\sigma, 1) \vdots \\ \Psi(\sigma, m_s) \end{pmatrix}.$$

- Pesendorfer and Schmidt-Dengler [2008] prove the following statement:

$$p = \Psi(p). \tag{8.12}$$

- In any Markov perfect equilibrium, the optimal conditional choice probability vector p satisfies the above equality.
- Conversely, any p that satisfies the equilibrium condition above can be derived as the optimal conditional choice probability of a Markov perfect equilibrium strategy profile.

8.1.11 Equilibrium Condition

- Pesendorfer and Schmidt-Dengler [2008] also show the following statements.
- By Brouwer's fixed-point theorem, a Markov perfect equilibrium exists.
- But Markov perfect equillibria need not be unique.
- Therefore, we may not be able to derive the likelihood.

- If the payoff function and transition probability is symmetric across players, then **symmetric** Markov perfect equilibrium exists.
- This significantly reduces the dimensionality of the problem.

8.1.12 Single Path of Play

- The estimation of dynamic games is mostly based on time series data of an industry.
- We often assume that the data is generated from a single path of play.
- Under Markov perfect equilibrium assumption, this means that players make same choices over time at the same state (s, ϵ_i).
- This assumption rules out equilibrium switches over time.
- If we use cross-sectional data, we have to keep this point in our mind, because then we observe data generated from multiple paths across markets.
- The assumption that the same equilibrium is played across markets may need further justification.

8.1.13 Identification

- The parameters of the model is $(\Pi_1, \cdots, \Pi_N, F, \beta, G)$.
- G is directly identified from data
- β and F are fixed.
- $(\Pi_1, \cdots, \Pi_N)$ contains $m_a \times m_s \times N$ unknown parameters.
- On the other hand, the equilibrium condition only has $K \times m_s \times N$ restrictions.
- This time, the normalization $\pi_i(0, a_{-i}|s) = 0$ is not sufficient.
- It is often assumed that the payoff of a player i is not directly influence by the state of other players:

$$\pi_i(a, s_i, s_{-i}) = \pi_i(a, s_i, s'_{-i}), \forall a \in A, s_i \in S_i, s_{-i}, s'_{-i} \in S_{-i}. \tag{8.13}$$

- Under these restrictions, the number of unknown parameters is $K(K+1)^{N-1} \times L \times N$.
- So, if $L \geq K + 1$, i.e., the number of states is larger than the number of actions, at least the order condition is satisfied.

8.1.14 Estimation: CCP Approach

- Because dynamic games often have multiple equillibria, nested-fixed point algorithm fails to pin down the likelihood of data.
- Instead we use CCP approach exploiting the equilibrium condition:

$$p = \Psi^{\theta}(p) = \Psi^{(\theta_1, \theta_2)}(p), \tag{8.14}$$

  where θ_1 is the parameters in the mean profit function, and θ_2 is the parameters in the transition law.
- In the first step, we estimate θ_2 and p directly from the data. Let $\hat{\theta}_2$ and $\hat{p}$ be the estimates.
- Then, we can estimate θ_1 by solving:

$$\min_{\theta_1} [\hat{p} - \Psi^{(\theta_1, \hat{\theta}_2)}(\hat{p})]' W [\hat{p} - \Psi^{(\theta_1, \hat{\theta}_2)}(\hat{p})], \tag{8.15}$$

  where W is some weighting matrix.

8.1.15 Computing Ψ

- Mapping from conditional choice probabilities to ex-ante value function:

$$V_i(\hat{p}) = [I - \beta \Sigma_i(\hat{p}) G]^{-1} [\Sigma_i(\hat{p}) \Pi_i + D_i(\hat{p})].$$

- Mapping from ex-ante value function *and conditional choice probabilities* to conditional choice probability:

$$p_i(a_i|s) = \mathbb{P}\{v_i(\tilde{p}, a_i, s) + \epsilon_{ia_i} \geq v_i(\tilde{p}, a_i', s) + \epsilon_{ia_i'}, \forall a_i' \in A_i\}, \tag{8.16}$$

where

$$v_i(\tilde{p}, a_i, s) = \sum_{a_{-i} \in A_{-i}} \tilde{p}(a_{-i}|s)[\pi_i(a_i, a_{-i}, s) + \beta \sum_{s' \in S} g(a_i, a_{-i}, s, s')V_i(\hat{p}, s')]. \tag{8.17}$$

- If ϵ_{ia} follows an i.i.d. type-I extreme value distribution, both $D_i(\hat{p})$ and the second equation have closed-forms.

8.2 With Continuous Dynamic Choice

8.2.1 Continuous Control

- So far we have assumed that players' actions are discrete.
- However, continuous controls such as investment prevails in the real world.
- One way to address this issues is to discretize the data and model everything in a discrete way.
- How to model continuous controls without discretization?
- This argument applies to the single-agent models as well.
- Let $a_{ti} = (a_{tid}, a_{tic}) \in A_i = A_{id} \times A_{ic}$ be a generic action of player i in period t, where $A_{id} = \{0, 1, \cdots, K\}$ and $A_{ic} \subset \mathbb{R}$.
- Let $s_{ti} \in S_i$ be a generic state of player i in period t, where $S_i \subset \mathbb{R}^L$.
- Let $\epsilon_{ti} \in \mathbb{R}^M$ be profitability shocks for player i in period t, which are independent across players.

8.2.2 Period Payoff

- Let $\pi_i(a, s, \epsilon_{ti})$ be the period payoff of player i in period t including the profitability shocks when the action profile is a, the state is s, and the profitability shocks are ϵ_{ti}.
- In the discrete choice framework, we assumed additive separability:

$$\pi_i(a, s, \epsilon_{ti}) = \pi_i(a, s) + \sum_{k=0}^{K} \epsilon_{tik} 1\{a_{ti} = k\}. \tag{8.18}$$

- For continuous control, we may want to impose a restriction that ensures the monotonicity of the policy function in the profitability shocks, for example:

$$\pi_i(a, s, \epsilon_{ti}) = \pi_i(a, s) + \sum_{k=0}^{K} \epsilon_{tik} 1\{a_{tid} = k\} + \epsilon_{tic} a_{tic}. \tag{8.19}$$

8.2.3 Policy Function Estimation with Continuous Control

- Let $a_{ic}(s, \epsilon_i) = a_{ic}(s, \epsilon_{ic})$ be the policy function of firm i and suppose that it is increasing in ϵ_{ic}.
- Moreover, assume that we know the distribution of ϵ_{ic}, say, $F_{\epsilon_c}(\cdot|s)$.
- We can identify the distribution of continuous control a_{ic} conditional on s directly from the data. Let $F_{a_{ic}}(a_{ic}|s)$ be the identified distribution function of a_{ic} conditional on s.

- Then, we can identify the policy function for the continuous control:

$$\begin{aligned} F_{a_{ic}}(a|s) &= \mathbb{P}\{a_{ic} \leq a|s\} \\ &= \mathbb{P}\{a_{ic}(s,\epsilon_{ic}) \leq a\} \\ &= \mathbb{P}\{\epsilon_{ic} \leq a_{ic}^{-1}(s,a)\} \\ &= F_c[a_{ic}^{-1}(s,a)|s] \\ &\Leftrightarrow a = F_{a_{ic}}^{-1}\{F_c[a_{ic}^{-1}(s,a)|s]|s\}, \end{aligned} \tag{8.20}$$

- Inserting $a = a_{ic}(s,\epsilon_{ic})$ to obtain:

$$\begin{aligned} a_{ic}(s,\epsilon_{ic}) &= F_{a_{ic}}^{-1}\{F_c[\underbrace{a_{ic}^{-1}(s,a_{ic}(s,\epsilon_{ic}))}_{\epsilon_{ic}}|s]|s\} \\ &= F_{a_{ic}}^{-1}[F_c^{-1}(\epsilon_{ic}|s)|s]. \end{aligned} \tag{8.21}$$

8.2.4 Converting to the Standard Discrete Choice Model

- Inserting the identified policy for the continuous controls, we have:

$$\begin{aligned} &\pi_i[a_{id}, a_{ic}(s,\epsilon_{ic}), a_{-i}, s, \epsilon_d] \\ &= \pi_i[a_{id}, a_{ic}(s,\epsilon_{ic}), a_{-i}, s] + \sum_{k=0}^{K} \epsilon_{ik} 1\{a_{id} = k\} + \epsilon_{ic} a_{ic}(s,\epsilon_{ic}) \end{aligned} \tag{8.22}$$

- Then, we can consider a choice-specific mean value function w.r.t. the discrete control a_{id} by integrating out ϵ_{ic}^t and others strategies and future behaviors.
- Therefore, we can identify the optimal conditional choice probability in a reduced-form and from which we can identify the policy function for the discrete choice as well.
- Then, we can apply the CCP approach to estimate the structural parameters.

8.2.5 Value Function Based CCP Approach

- Bajari et al. [2007] proposes an alternative estimation method.
- The key is to characterize the equilibrium as follows.
- Let $V_i^{(a_i,a_{-i})}(s)$ be the ex-ante value function associated with strategy profile (a_i, a_{-i}).
- The observed (a_i^*, a_{-i}^*) is an equilibrium if:

$$V_i^{(a_i^*,a_{-i}^*)}(s) \geq V_i^{(a_i',a_{-i}^*)}(s), \forall a_i' \in A_i, \tag{8.23}$$

  for all $s \in S, i = 1, \cdots, N$.
- Then, we can consider an objective function such that:

$$Q(\theta) = \sum_{s=1}^{m_s} \sum_{i=1}^{N} \sum_{a_i'} \min\{V_i^{(a_i^*,a_{-i}^*)}(s) - V_i^{(a_i',a_{-i}^*)}(s), 0\}^2, \tag{8.24}$$

  which penalizes the parameter violating the inequality condition of the equilibrium.
- There is a discretion about the choice of alternative strategies to construct the objective function.
- Thus, the summation over continuous control does not cover the entire range.

8.2.6 Approximating the Ex-ante Value Function

- Once we obtain the equilibrium policy functions, then in principle, we can compute the associated ex-ante value function, $V_i^{(a_i,a_{-i})}$, in the equilibrium.
- We often use a numerical method to compute the ex-ante value function when continuous controls are relevant.
- Draw forward profitability shocks $\epsilon_{ti}^{(r)}$ for $t = \cdots, \tau$ for $r = 1, \cdots, R$.
- At each period, sample $a_{tid}^{(r)}$ from the estimated conditional choice probability.
- Also, compute $a_{tic}^{(r)} = a_{tic}(s_t^{(r)}, \epsilon_{tic}^{(r)})$ using the estimated policy function for the continuous control and let $a_{ti}^{(r)} = (a_{tid}^{(r)}, a_{tic}^{(r)})$.
- Draw next period states $s_{t+1}^{(r)}$ from $g(a_t^{(r)}, s_t^{(r)}, s')$.
- Continue this until $t = \tau$.
- Compute:

$$V_i^{(a_i^*,a_{-i}^*),R\tau}(s) = \frac{1}{R}\sum_{r=1}^{R}\sum_{t=1}^{\tau}\beta^t \pi_i[a^{(r)}, s^{(r)}, \epsilon_i^{(r)}] \tag{8.25}$$

  to approximate $V_i^{(a_i^*,a_{-i}^*)}(s)$.
- It is a τ-period forward simulation of the ex-ante value functions.
- For arbitrary a_i', we can approximate $V_i^{(a_i',a_{-i}^*)}(s)$ by following the same steps using the perturbed policy a_i'.

8.3 Dynamic Oligopoly Model

- In the application of the dynamic game estimation to the dynamic oligopoly model, we often jointly consider entry and exit decisions and investment decisions of firms.
- The entry and exit decisions are the discrete decisions and the investment decision is the continuous controls.
- The formal models and their characterization are found in Ericson and Pakes [1995] and Doraszelski and Satterthwaite [2010].
- Igami et al. [2018]', the paper referred to in the introduction, falls into this category.

8.3.1 Ryan [2012]

- Ryan [2012] is one of the earliest applications.
- In 1990, the U.S. congress passed Amendments to the Clean Air act, adding new categories of regulated SO_2 and NO_x emissions and requiring plants to undergo an environmental certification process.
- How costly is this regulation to the economy?
- The cost analysis is typically an engineering estimates of the expenditures on control and monitoring equipment necessary to bring a plant into compliance with the new regulation.
- However, the regulation changes both the sunk cost of entry as well as the investment cost. If the sunk cost is higher, the less firm will be active in the market, which reinforces the market power of active firms.
- This affects the equilibrium market structure.
- The engineering based cost analysis misses this important welfare loss from the changes in the market structure.
- This paper quantifies the welfare cost based on an empirical dynamic oligopoly model.

8.3.2 Setting

- There are several regional cement markets in the U.S.
- Each market is described by $\overline{N} \times 1$ state vector s_t, where s_{it} is the capacity of the ith firm at time t, and $\overline{N}$ is an exogenously imposed maximal number of active firms.
- $s_{it} = 0$ means the firm is inactive.

8.3.3 Timing of the Game

- Incumbent firms ($s_{it} > 0$) receives a private draw from the distribution of scrap values.
- Incumbent firms decide whether to exit.
- Potential entrants receive a private draw of both investment and entry costs, while incumbent firms receive a private draw of both investment and disinvestment costs.
- All firms simultaneously make entry and investment decisions.
- Incumbent firms compete in the product market.
- Firms entry, exit, and investment are realized.
- The economy moves to the next period.

8.3.4 Demand Function

- In each market m, firms face a constant elasticity of demand curves:

$$\ln Q_m(\alpha) = \alpha_{0m} + \alpha_1 \ln P_m, \tag{8.26}$$

  where Q_m is the aggregate market quantity, P_m is price, α_{0m} is market-specific intercept, and α_1 is the elasticity of demand.

8.3.5 Production Cost Function

- The cost of output, q_i, is given by:

$$C_i(q_i; \delta) = \delta_0 + \delta_1 q_1 + \delta_2 1\{q_i > \epsilon s_i\}(q_i - \epsilon s_i)^2, \tag{8.27}$$

  where q_i is the output of firm i, δ_0 is the fixed cost of production, δ_1 is the linear variable cost, and the last term is a quadratic cost that matters only when the output is sufficiently close to the capacity.

8.3.6 Investment Cost

- The cost of investment and disinvestment is:

$$\begin{aligned} \Gamma(x_i; \gamma) = {} & 1\{x_i > 0\}(\gamma_{i1} + \gamma_2 x_i + \gamma_3 x_i^2) \\ & + 1\{x_i < 0\}(\gamma_{i4} + \gamma_5 x_i + \gamma_6 x_i^2), \end{aligned} \tag{8.28}$$

where x_i is the investment of firm i, and fixed investment and disinvestment costs γ_{i1} and γ_{i4} are drawn from distributions F_γ and G_γ. - Let μ_γ^+ and σ_γ^+ be the mean and standard deviation of F_γ and μ_γ^- and σ_γ^- be the mean and standard deviation of G_γ.

8.3.7 Entry and Exit Costs

- If the firm is a new entrant, it draws entry cost $-\kappa_i$ from F_κ.
- IF the firm is an incumbent, it draws scrap values ϕ_i from F_ϕ.
- Then entry and exit cost is:

$$\Phi_i(a_i, \kappa_i, \phi_i) = \begin{cases} -\kappa_i & \text{if the firm enters} \\ \phi_i & \text{if the firm exits.} \end{cases} \tag{8.29}$$

8.3.8 Period Profit Function

- As a consequence, the period profit function of firm i is:

$$\pi_i(s, a; \alpha, \delta, \gamma_i, \kappa_i, \phi_i) = \tilde{\pi}_i(s; \alpha, \delta) - \Gamma(x_i; \gamma_i) + \Phi_i(a_i; \kappa_i, \phi_i). \tag{8.30}$$

- Let θ summarize the parameter regarding the period profit.

8.3.9 Transition

- The transition is deterministic.
- The investment changes the capacity s_i.
- As a firm exits, the capacity moves to 0.
- As a firm enters, the capacity moves to the initial investment level.

8.3.10 Equilibrium

- Let ϵ_i represent the firm's private information about the cost of entry, exit, investment, and disinvestment.
- Each firm's Markovian strategy $\sigma_i(s, \epsilon_i)$ is a mapping from states and shocks to actions.

8.3.11 Value Function for Incumbents

- The value function for a firm with $s_i > 0$ at the time of the exit decision under strategy profile σ is:

$$\begin{aligned} &V_i(s; \sigma(s), \theta, \epsilon_i) \\ &= \tilde{\pi}_i(s; \theta) + \max \Bigg\{ \phi_i, \\ &\mathbb{E}_{\epsilon_i} \max_{x_i^* \geq 0} \left[-\gamma_{i1} - \gamma_2 x_i^* - \gamma_3 x_i^{*2} + \beta \int \mathbb{E}_{\epsilon_i} V_i(s'; \sigma(s'), \theta, \epsilon_i) dP(s_i + x^*, s'_{-i}; s, \sigma(s)) \right], \\ &\max_{x_i^* < 0} \left[-\gamma_{i4} - \gamma_5 x_i^* - \gamma_6 x_i^{*2} + \beta \int \mathbb{E}_{\epsilon_i} V_i(s'; \sigma(s'), \theta, \epsilon_i) dP(s_i + x^*, s'_{-i}; s, \sigma(s)) \right] \Bigg\}, \end{aligned} \tag{8.31}$$

where each term in the blanket represents the value of exit, staying and making investment, and staying and making disinvestment. - At this timing the incumbent only observes ϕ_i, although it is a bit confusing because the value function is written as a function of ϵ_i.

8.3.12 Value Function for Entrants

- The value function for a firm with $s_i = 0$ at the time of entry decision under strategy profile σ at the time of entry and investment decisions are:

$$\begin{aligned} & V_i(s; \sigma(s), \theta, \epsilon_i) \\ & = \max \Bigg\{ 0, \\ & \max_{x_i^* > 0} \Bigg[-\gamma_{1i} - \gamma_2 x_i^* - \gamma_3 x_i^2 + \beta \int \mathbb{E}_{\epsilon_i} V_i(s'; \sigma(s'), \theta, \epsilon_i) dP(s_i + x_i^*, s'_{-i}; s, \sigma(s)) \Bigg] - \kappa_i \Bigg\}. \end{aligned} \tag{8.32}$$

8.3.13 Markov Perfect Equilibrium

- The strategy profile σ^* is a Markov perfect equilibrium if:

$$V_i(s; \sigma_i^*(s), \sigma_{-i}^*(s), \theta, \epsilon_i) \geq V_i(s; \sigma_i'(s), \sigma_{-i}^*(s), \theta, \epsilon_i), \forall \sigma_i', \tag{8.33}$$

  for all s, ϵ_i, and i.
- To apply the CCP approach, we first have to assume that the same equilibrium is played across markets.
- This assumption is not innocuous as we saw in the previous section.
- It is also important to notice that we have to fix the expectation about the regulation regime by firms.
- We assume that firms assume that the regulatory environment is permanent.

8.3.14 Demand and Production Cost Estimation

- To estimate the demand function, Ryan uses supply-side cost shifters as the instrumental variables: coal prices, gas prices, electricity rates, and wage rates.
- Given the demand function, Ryan estimate the cost function from the FOC for the quantity competition.
- You should already know how to do this.

8.3.15 Investment Policy Function Estimation

- With fixed costs of investment and disinvestment, investment policy function follows a so-called (s, S) rule, in which firms investment only when the current capacity is below a certain lower bound and makes disinvestment only when the current capacity is above a certain upper bound.
- The lower and upper bounds, and the adjustment level are chosen optimal as a function of the state.

8.3.16 Investment Policy Function Estimation

- Let s_{it}^* be the target level of capacity, which is parameterize as:

$$\ln s_{it}^* = \lambda_1' bs(s_{it}) + \lambda_2' bs\left(\sum_{j \neq i} s_{jt} \right) + u_{it}^*, \tag{8.34}$$

  where $bs(\cdot)$ is finite-dimensional piece-wise cubic b-splines.
- The upper and lower bounds are parameterized as:

$$\begin{aligned}\overline{s}_{it} &= s^*_{it} + \exp\left(\lambda_3' bs_1(s_{it}) + \lambda_4' bs_2\left(\sum_{j\neq i} s_{jt}\right) + \overline{u}^b_{it}\right),\\ \underline{s}_{it} &= s^*_{it} - \exp\left(\lambda_3' bs_1(s_{it}) + \lambda_4' bs_2\left(\sum_{j\neq i} s_{jt}\right) + \underline{u}^b_{it}\right).\end{aligned} \tag{8.35}$$

8.3.17 Entry and Exit Policy Functions

- The probability of entry is parameterized as:

$$\mathbb{P}\{\chi_i = 1; s_i = 0, s\} = \Phi\left(\psi_1 + \psi_2\left(\sum_{j\neq i}\right) + \psi_3 1(t > 1990)\right), \tag{8.36}$$

  and the probability of exit is parameterized as:

$$\mathbb{P}\{\chi_i = 0; s_i > 0, s\} = \Phi\left(\psi_4 + \psi_5 s_{it} + \psi_6\left(\sum_{j\neq i}\right) + \psi_7 1(t > 1990)\right), \tag{8.37}$$

  where Ψ is the cumulative distribution function of the standard normal random variable.

8.3.18 BBL Estimation

- Now by comparing the value function under the identified policies with value functions under alternative policies, one can estimate the structural parameters.
- Alternative policies can be obtained by perturbing the upper and lower bounds and the target level of investment and the thresholds for entry and exit around the equilibrium policies.
- How much to perturb is a practical issue.
- Then, Ryan finds structural parameters that minimizes the BBL objective function.

8.3.19 Structural Parameters Estimation Results

8.3.20 Counterfactual Simulation

- To asses the welfare cost of the amendment, Ryan solves the model under two sets of parameters: the actual parameters and the parameters before the regulation.
- In the simulation, we can set whatever initial state.
- Ryan consider two types of initial state:
  - No incumbent firms and 4 potential entrants.
  - Two incumbent firms and two potential entrants.

8.3.21 Counterfactual Simulation Results

8.3.22 Counterfactual Simulation Estimation Results

8.4 Dynamic Oligopoly with Many Firms

- Solving a dynamic game becomes quickly intractable when the number of firms increases and hence the dimension of the state profile increases.

TABLE IX

DYNAMIC PARAMETERS[a]

	Before 1990		After 1990		Difference	
	Mean	SE	Mean	SE	Mean	SE
Parameter						
Investment cost	230	85	238	51	−8	19
Investment cost squared	0	0	0	0	0	0
Divestment cost	−123	34	−282	56	−155	35
Divestment cost squared	3932	1166	5282	1130	1294	591
Investment Fixed Costs						
Mean (μ_γ^+)	621	345	1253	722	653	477
Standard deviation (σ_γ^+)	113	72	234	145	120	97
Divestment Fixed Costs						
Mean (μ_γ^-)	297,609	84,155	307,385	62,351	12,665	34,694
Standard deviation (σ_γ^-)	144,303	41,360	142,547	29,036	109	17,494
Scrap Values						
Mean (μ_ϕ)	−62,554	33,773	−53,344	28,093	9833	21,788
Standard deviation (σ_ϕ)	75,603	26,773	69,778	27,186	−6054	11,702
Entry Costs						
Mean (μ_κ)	182,585	36,888	223,326	45,910	43,654	21,243
Standard deviation (σ_κ)	101,867	22,845	97,395	14,102	−6401	12,916

TABLE X

COUNTERFACTUAL POLICY EXPERIMENTS[a]

	Low Entry Costs (Pre-1990)		High Entry Costs (Post-1990)		Difference	
	Mean	Standard Error	Mean	Standard Error	Mean	Standard Error
De Novo Market						
Total producer profit ($ in NPV[b])	43,936.11	(7796.98)	33,356.87	(7767.22)	−11,182.04	(7885.20)
Profit firm 1 ($ in NPV)	45,126.30	(10,304.87)	34,321.61	(9520.93)	−11,965.22	(11,684.96)
Total net consumer surplus ($ in NPV)	1,928,985.09	(62,750.34)	1,848,872.52	(75,729.17)	−66,337.44	(58,404.32)
Total welfare ($ in NPV)	2,116,810.12	(74,265.74)	1,992,937.65	(96,634.83)	−119,771.39	(49,423.06)
Periods with no firms (periods)	1.29	(0.08)	1.32	(0.09)	0.04	(0.08)
Periods with one firm (periods)	1.51	(0.37)	2.60	(0.86)	1.05	(0.78)
Periods with two firms (periods)	8.17	(4.68)	21.43	(9.92)	12.26	(9.99)
Periods with three firms (periods)	54.71	(20.22)	91.35	(21.27)	33.38	(18.85)
Periods with four firms (periods)	135.91	(24.64)	84.03	(32.67)	−46.73	(25.04)
Average size of active firm (tons)	980.71	(76.18)	1054.65	(85.17)	73.42	(74.01)
Average market capacity (tons)	3467.85	(188.21)	3352.23	(208.94)	−112.75	(107.84)
Average market quantity (tons)	3094.23	(161.57)	2987.61	(177.58)	−105.69	(89.41)
Average market price	66.66	(1.90)	68.12	(2.11)	1.47	(1.14)

(*Continues*)

TABLE X—*Continued*

	Low Entry Costs (Pre-1990)		High Entry Costs (Post-1990)		Difference	
	Mean	Standard Error	Mean	Standard Error	Mean	Standard Error
Mature Market						
Total producer profit ($ in NPV)	223,292.75	(4831.95)	231,568.23	(5830.42)	9551.01	(5465.77)
Profit firm 1 ($ in NPV)	549,179.30	(14,138.37)	579,742.32	(20,446.75)	32,968.00	(19,161.33)
Total net consumer surplus ($ in NPV)	2,281,584.08	(52,663.88)	2,208,573.20	(62,906.14)	−62,974.37	(32,662.05)
Total welfare ($ in NPV)	3,178,504.60	(60,267.34)	3,141,916.43	(62,618.02)	−30,099.56	(18,078.21)
Periods with no firms (periods)	0.00	(0.00)	0.00	(0.00)	0.00	(0.00)
Periods with one firm (periods)	0.00	(0.00)	0.00	(0.00)	0.00	(0.00)
Periods with two firms (periods)	8.63	(3.57)	23.20	(10.05)	14.13	(10.00)
Periods with three firms (periods)	61.32	(16.83)	98.37	(21.49)	35.73	(20.16)
Periods with four firms (periods)	131.52	(20.10)	78.38	(31.99)	−50.00	(27.39)
Average size of active firm (tons)	989.33	(44.45)	1059.31	(63.41)	73.48	(54.64)
Average market capacity (tons)	3502.49	(171.20)	3371.03	(191.87)	−117.56	(73.19)
Average market quantity (tons)	3123.42	(150.66)	3001.98	(165.51)	−108.16	(69.48)
Average market price	66.82	(1.64)	68.36	(1.91)	1.44	(0.85)

[a]Industry distributions were simulated along 25,000 paths of length 200 each. All values are present values denominated in thousands of dollars. The new market initially has no firms and four potential entrants. The incumbent market starts with one 750,000 TPY incumbent and one 1.5M TPY incumbent and two potential entrants. Counts of active firms may not sum to 200 due to rounding off. Means and standard deviations were calculated by subsampling.

[b]Where NPV denotes Net Present Value.

- Weintraub et al. [2008] propose a new equilibrium concept called **oblivious equilibrium** that approximates a Markov perfect equilibrium when there are many firms and do not suffer from the aforementioned computational problem.

8.4.1 Setting

- The setting is effectively the same as Ericson and Pakes [1995] and Doraszelski and Satterthwaite [2010].
- The time is discrete and infinite $t \in \mathbb{N}$.
- The information set at time t is $\mathcal{F}_t$.
- Each firm that enters the industry is assigned a positive integer value index $x_{it} \in \mathbb{N}$.
- It can be regarded as a quality index, but can be whatever.
- Let S_t be the set of indices of incumbent firms at time t.
- Let n_t be the number of incumbent firms at time t.
- The **industry state** s_t is a vector of counts over possible values of x_{it} among $i \in S_t$.
- The state space is $\overline{\mathcal{S}} := \{s \in N^\infty : \sum_{x=0}^{\infty} s(x) < \infty\}$.
- For expositional reason, consider an extended state space too $\mathcal{S} := \{s \in \mathbb{R}_+^\infty : \sum_{x=0}^{\infty} s(x) < \infty\}$
- For each $i \in S_t$, define the state of the **competitors** by $s_{-i,t}(x) := s_t(x) - 1\{x_{it} = x\}$.
- The expected period profit of firm i is $\pi(x_{it}, s_{-i,t})$.
- In each period, each incumbent draws a positive real-valued sell-off value ϕ_{it}, and exits from the market if the sell-off value exceeds the continuation value.
- If an incumbent firm stays in the market, the firm chooses an investment $\iota_{it} \in \mathbb{R}_+$.
- Then, the state of the firm is updated according to a rule:

$$x_{i,t+1} = \max\{0, x_{it} + w(\iota_{it}, \zeta_{i,t+1})\},$$

- where w captures the impact of investment and $\zeta_{i,t+1}$ is a shock to the investment process.
- The unit cost of investment is d.
- In each period, new firms can enter the industry by paying a setup cost κ.
- Entrants do not earn profits in the period that they enter.
- They appear in the following period at state $x^e \in \mathbb{N}$.

8.4.2 Timing of the Game

1. Each incumbent firm observes its sell-off value, and then makes exit and investment decisions.
2. The number of entering firms is determined and each entrant pays an entry cost κ.
3. Incumbent firms compete in the spot market and receive profits.
4. Exiting firms exit and receive their sell-off values.
5. Investment outcomes are determined, new entrants enter, and the industry takes on a new state s_{t+1}.

8.4.3 Assumptions

- An industry state $s \in \mathcal{S}$ is said to dominate $s' \in \mathcal{S}$ if for all $x \in \mathbb{N}$ $\sum_{z \geq x} s(z) \geq \sum_{z \geq x} s'(z)$, and the relation is denoted by $s \succeq s'$.
- **Assumptions on** π:

1. For all $s \in \mathcal{S}$, $\pi(x, s)$ is increasing in x.
2. For all $x \in \mathbb{N}$ and $s, s' \in \mathcal{S}$, if $s \succeq s'$, then $\pi(x, s) \leq \pi(x, s')$.
3. For all $x \in \mathbb{N}$ and $s \in \mathcal{S}$, $\pi(x, s) > 0$ and $\sup_{x,z} \pi(x, s) < \infty$.
4. For all $x \in \mathbb{N}$, the function $\ln \pi(x, \cdot) : \mathcal{S} \to \mathbb{R}_+$ is "smooth" enough (for the rigorous assumption, refer to the paper).

- **Assumptions on** w **and shocks**:

1. The random variables $\{\phi_{it}|t \geq 0, i \geq 1\}$ are i.i.d. and have finite expectations and well-defined density functions with support $\mathbb{R}_+$.
2. The random variables $\{\zeta_{it}|t \geq 0, i \geq 1\}$ are i.i.d. and independent of $\{\phi_{it}|t \geq 0, i \geq 1\}$.
3. For all ζ, $w(\iota, \zeta)$ is non-decreasing in ι.
4. For all $\iota > 0$, $\mathbb{P}\{w(\iota, \zeta_{i,t+1} > 0\} > 0$.
5. There exists a positive constant $\overline{w} \in \mathbb{N}$ such that $|w(\iota, \zeta)| \leq \overline{w}$ for all (ι, ζ).
6. There exist a positive constant $\overline{\iota}$ such that $\iota_{it} < \overline{\iota}$, $\forall i, t$.
7. For all $k \in \{-\overline{w}, \cdots, \overline{w}\}$, $\mathbb{P}\{w(\iota, \zeta_{i,t+1} = k\}$ is continuous in ι.
8. The transitions generated by $w(\iota, \zeta)$ are **unique investment choice admissible**.

- The notion of unique investment choice admissibility is introduced by @Doraszelski and Satterthwaite [2010] and means that under the investment function firms' investment decision problem has a unique solution.
- Without this assumption, the existence of pure-strategy equilibrium is not guaranteed and the computation can become complicated.
- **Assumptions on potential entrants**:
- When there are a large number of potential entrants who play a symmetric mixed entry strategy, the equilibrium number of entrants is well-approximated by a Poisson distribution.
- We assume this for simplicity.

1. The number of firms entering during period t is a Poisson random variable that is independent of $\{\phi_{it}|t \geq 0, i \geq 1\}$ conditional on s_t.
2. $\kappa > \beta\overline{\phi}$, where $\overline{\phi}$ is the expected net present value of entering the market, investing zero and earning zero profits each period, and then exiting at an optimal stopping time.

- Let $\lambda(s_t)$ be the expected number of firms entering at industry state, which is endogenously determined.

8.4.4 Strategy Profile

- Investment strategy $\iota : \mathbb{N} \times \mathcal{S} \to \mathbb{R}_+ : \iota_{it} = \iota(x_{it}, s_{-it})$.
- Exit strategy $\rho : \mathbb{N} \times \mathcal{S} \to \mathbb{R}_+ :$ an incumbent exits if and only if $\phi_{it} > \rho(x_{it}, s_{-it})$.
- Bundle them as $\mu = (\iota, \rho) \in \mathcal{M}$.
- Entry rate $\lambda : \mathcal{S} \to \mathbb{R}_+$, $\lambda \in \Lambda$.

8.4.5 Value Function

- The expected net present value for a firm at state x when its competitors state is s, given that its competitors each follows a common strategy $\mu \in \mathcal{M}$, the entry rate function is $\lambda \in \Lambda$, and the firm itself follows strategy $\mu' \in \mathcal{M}$ is:

$$V(x, s|\mu', \mu, \lambda) := \mathbb{E}_{\mu', \mu, \lambda}\left\{\sum_{k=t}^{\tau_i} \beta^{k-t}[\pi(x_{ik}, s_{-ik}) - d\iota_{ik}] + \beta^{\tau_i - t}\phi_{i,\tau_i}|x_{it} = x, s_{-it} = s\right\},$$

  where τ_i is a random variable that represents the time at which firm i exits under the exit strategy.
- We also write as $V(x, s|\mu, \lambda) := V(x, s|\mu, \mu, \lambda)$ when all firms follow the same strategy.

8.4.6 Markov-perfect Equilibrium

- A strategy profile $\mu \in \mathcal{M}$ and an entry rate $\lambda \in \Lambda$ is a Markov-perfect equilibrium if:

1. The strategy profile satisfies:

$$\sup_{\mu' \in \mathcal{M}} V(x, s|\mu', \mu, \lambda) = V(x, s|\mu, \lambda), \forall x \in \mathbb{N}, \forall s \in \overline{\mathcal{S}}.$$

2. The entry rate satisfies:

$$\sum_{s\in\overline{S}}\lambda(s)\{\beta\mathbb{E}_{\mu,\lambda}[V(x^e,s_{-i,t+1}|\mu,\lambda)|s_t=s]-\kappa\}=0,$$

$$\beta\mathbb{E}_{\mu,\lambda}[V(x^e,s_{-i,t+1}|\mu,\lambda)|s_t=s]-\kappa\leq 0,\forall s\in\overline{\mathcal{S}},$$

$$\lambda(s)\geq 0, s\in\overline{\mathcal{S}}.$$

- That is, at each industry state, either the expected entry profit is zero or the entry rate is zero.
- The authors prove that there exists a symmetric equilibrium.

8.4.7 Oblivious Strategy

- The Markov-perfect equilibrium becomes quickly computationally intractable when the number of firms increases, because firms' strategies depends on other firms' states.
- We consider a firm who is **oblivious** of other firms' exact states but only cares the **long-run average industry state** on top of its own state.
- Let $\widetilde{\mathcal{M}}\subset\mathcal{M}$ and $\widetilde{\Lambda}\subset\Lambda$ be the set of **oblivious strategies** and the set of **oblivious entry rate functions** in the sense that both $\mu\in\widetilde{\mathcal{M}}$ and $\lambda\in\widetilde{\Lambda}$ do not depend on s.

8.4.8 Long-run Average Industry State

- Then, if all firms use a common strategy $\mu\in\widetilde{\mathcal{M}}\subset\mathcal{M}$, each firm's state evolves as an independent transient Markov chain.
- Let the k-period transition probability of this Markov chain be denoted by $P_\mu^k(x,y)$.
- The expected time that a firm spends at state x is $\sum_{k=0}^\infty P_\mu^k(x^e,x)$.
- The expected lifespan of a firm is $\sum_{k=0}^\infty\sum_{x\in\mathbb{N}}P_\mu^k(x^e,x)$.
- Let $\tilde{s}_t(x):=\mathbb{E}[s_t(x)]$ be the expected number of firms at state x.
- Under the stated assumptions, if firms make decisions according to an oblivious strategy $\mu\in\widetilde{M}$ and enter according to an oblivious entry rate function $\lambda\in\widetilde{\Lambda}$, and the expected time that firm spends in the industry is finite, then:

$$\lim_{t\to\infty}\tilde{s}_t(x)=\lambda\sum_{k=0}^\infty P_\mu^k(x^e,x),$$

  for all $x\in\mathbb{N}$.
- We write the long-run average industry state as $\tilde{s}_{\mu,\lambda}(x):=\lim_{t\to\infty}\tilde{s}_t(x)$.

8.4.9 Oblivious Value Function

- For an oblivious strategy $\mu',\mu\in\widetilde{\mathcal{M}}$ and an oblivious entry rate function $\lambda\in\widetilde{\Lambda}$, we define an **oblivious value function**:

$$\widetilde{V}(x|\mu',\mu,\lambda):=\mathbb{E}_{\mu'}\left\{\sum_{k=t}^{\tau_i}\beta^{k-t}[\pi(x_{ik},\tilde{s}_{\mu,\lambda})-d\iota_{ik}]+\beta^{\tau_i-t}\phi_{i,\tau_i}|x_{it}=x\right\}.$$

- The firm approximates the value function by evaluating at the long-run average industry state.
- Note that it still depends on other firms' strategy μ because it determines the long-run average industry state.
- We also write as $\widetilde{V}(x|\mu,\lambda):=\widetilde{V}(x|\mu,\mu,\lambda)$.

8.4.10 Oblivious Equilibrium

- An oblivious strategy $\mu', \mu \in \widetilde{\mathcal{M}}$ and an oblivious entry rate function $\lambda \in \widetilde{\Lambda}$ is an **oblivious equilibrium** if:

1. The oblivious strategy satisfies:

$$\sup_{\mu' \in \widetilde{\mathcal{M}}} \widetilde{V}(x|\mu', \mu, \lambda) = \widetilde{V}(x|\mu, \lambda), \forall x \in \mathbb{N}.$$

2. The oblivious entry rate function satisfies:

$$\lambda[\beta \widetilde{V}(x^e|\mu, \lambda) - \kappa] = 0,$$

$$\beta \widetilde{V}(x^e|\mu, \lambda) - \kappa \leq 0,$$

$$\lambda \geq 0.$$

- Computing the equilibrium is surprisingly easy because the dynamic programming involves only one-dimensional state variable x_{it}.
- Given a strategy profile, first compute $\tilde{s}_{\mu,\lambda}$.
- Given $\tilde{s}_{\mu,\lambda}$, the dynamic programming is only of x_{it}.

8.4.11 Asymptotic Results and Extensions

- The authors show that under a certain condition when the market size goes to infinity, an oblivious equilibrium approximates an Markov-perfect equilibrium in the sense that a deviation from the oblivious equilibrium to a (possibly non-oblivious) best response does not improve the value for a firm.
- Benkard et al. [2015] consider an extension of the current model to the **partially oblivious equilibrium** in which a subset of firms follow a non-oblivious strategy and the other firms follow an oblivious strategy.
- Ifrach and Weintraub [2017] further extends the current model to the **moment-based Markov equilibrium** in which a firm's strategy depends on some sets of moments regarding the industry state.

Chapter 9

Auction

9.1 General symmetric information model

9.1.1 Setting

- The argument of this section is based on Hendricks and Porter [2007].
- General symmetric information model [Milgrom and Weber, 1982].
- The seller has a single item.
- There are n potential risk neutral buyers/bidders.
- Each bidder i observes a real-valued private signal X_i.
- There is a random variable or vector V that influences the value of the object to the bidders.
- The joint distribution of $(V, X_1, \cdots, X_n)$ is F.
- Bidder i's payoff is $U_i = u_i(V, X_i, X_{-i})$ when the bidder i obtains the object being sold.
- The seller announces a reserve price r.
- The primitives of the model, the number of bidders n, the distribution F, and the utility functions $\{u_i\}_{i=1}^n$, are common knowledge among buyers.
- Assumption: u_i is non-negative, continuous, increasing in each argument and symmetric in the components of X_{-i}.
- Assumption: $(V, X_1, \cdots, X_n)$ are affiliated.
  - The n random variables $X = (X_1, \cdots, X_n)$ with joint density $f(x)$ are affiliated if for all x and y, $f(x \wedge y)f(x \vee y) \geq f(x)f(y)$. If affiliated, they are non-negatively correlated.
- $\rightarrow U_1, \cdots, U_n$ are affiliated.
- Let $Y_1, \cdots, Y_{n-1}$ is the ordering of the largest through the smallest signals from $X_2, \cdots, X_n$.
- Then, $(V, X_1, Y_1, \cdots, Y_{n-1})$ are also affiliated.

9.1.2 Discussion About the Asssumptions

- The private information is assumed to be single-dimensional.
  - If the private information is multi-dimensional, the identification of type requires at least as many messages.
  - Recently, there are few empirical studies of auctions of multi-dimensional private information such as Bajari et al. [2014] and Takahashi [2018].
- The distribution of the signals is assumed to be symmetric across bidders.
  - We can introduce asymmetric signals.
  - A bidder may have more precise signals.
- The utility is independent of the winner's identify when a bidder is not awarded the item.

  - If the item is the valuable asset in the oligopoly industry, the implication to the profit can be different when the close competitor is awarded the asset.
- The number of potential bidders is assumed to be common knowledge.
  - The participation to the auction may be endogenously determined.

9.1.3 Bidding Strategy

- A **bidding strategy** for bidder i is a correspondence $\beta_i : X_i \to \mathbb{R}_+$.
- A mapping from the private signal into a non-negative real value.
- Under the stated assumptions, there exists a Bayesian Nash equilibrium with non-decreasing bid functions [Krishna, 2009].
- The following argument crucially depends on this property of the equilibrium bidding rule.

9.1.4 Special Cases

- Two special cases about the payoff functions:
- **Private value (PV)**: $u_i(v, x_i, x_{-i}) = x_i$: Bidder i knows his own valuation and is only uncertain about how much others value the item.
- **Pure common values (PCV)**: $u_i(v, x_i, x_{-i}) = v$: All buyers have the same valuation, which is unknown to them when they bid, and only learned though the private signals.
- A remark on the private value assumption:
  - The assumption holds in more general settings than we may think.
  - Suppose that $u_k(v, x_i, x_{-i})$ depends on v, x_i, x_{-i}.
  - Suppose that bidder i is uncertain about V and the distribution of V conditional on x_i is independent of x_{-i}.
  - Suppose that $u(v, x_i, x_{-i}) = u(x, x_i)$.
  - Then, $\mathbb{E}\{u_i(V, X_i)|X_i = x_i, X_{-i} = x_{-i}\} = \mathbb{E}\{u_i(V, X_i)|X_i = x_i\} := f(x_i)$ for some monotone increasing function f.
  - This is still a private value model.
  - If the distribution of V is not independent of some x_i, it is no longer a private value model.
  - The last case is referred to as the **common values (CV)** model.
- Two special cases about the signals:
- **Independent signals (IPV, ICV)**: Signals $X_1, \cdots, X_n$ are independent.
- **Affiliated signals (APV, ACV)**: Signals $X_1, \cdots, X_n$ are affiliated.
- Example: Offshore oil and gas leases.
  - V: the size of oil or gas deposits under the tract.
  - CV: bidders are uncertain about V and have different private information about the value of V because of the seismic data they obtain.
  - PV: bidders are almost certain about V or have little discrepancy in private assessment of V. But there is a heterogeneity in the costs of exploration and drilling and this information is private.
- In the following, we mostly consider a type of CV model such as $u_i(v, x_i, x_{-i}) = u(v, x_i)$.

9.2 Second-Price Auctions

9.2.1 The Button Auction

- Let:
$$w(x, y) := \mathbb{E}\{u(V, x)|X_1 = x_1, Y_1 = y_1\},$$
be the expected payoff of the bidder when her signal is x and the highest rival's signals is y_1.
- Let $\underline{x}$ is the lower bound of the support of X.

- **Button auction**:
  - The price rises continuously.
  - Bidders stays active as long as they keep fingers on the button.
  - A bidder wins once all other bidders take their fingers off.
  - The price paid by the winner is the price level when the second last bidder takes the fingers off.
- The bidding strategy is the mapping from a price level to being active or not.

9.2.2 Equilibrium Bidding Strategies

- Case 1:
  - A bidder cannot observe the prices at which the other bidders take fingers off.
  - The equilibrium strategy is to take the fingers off at the threshold:

$$\beta(x) = w(x, x).$$

  - Suppose that the other bidders follow the bidding strategy and the price level is b and the auction does not yet end.
  - Suppose that I win at this moment.
  - This means that there is at least one other bidder that has signal $y = \beta^{-1}(b)$.
  - Then, the payoff to me is $w(x, y) - b = w(x, y) - w(y, y)$.
  - It is positive if and only if $x > y$, because affiliation implies that the expected payoff is increasing in the own signal.
  - Thus, $\beta(x) = w(x, x)$ is the best response.
- Case 2:
  - Active bidders observe the prices at which rivals drop out.
  - No bidder who drops out can become active again.
  - The bidding strategy is the mapping from the number of rivals who dropped out and the prices at which they dropped out.
  - Let $\beta_k(x)$ be the price at which a bidder drops out when k rivals dropped out at prices $b_1, \cdots, b_k$.
  - At the equilibrium it should be:

$$\beta_k(x) = \mathbb{E}\{u(V, x)|X_1 = x, Y_1 = \cdots = Y_{n-k-1} = x, Y_{n-k} = \beta_{k-1}^{-1}(b_k), \cdots, Y_{n-1} = \beta_0^{-1}(b_1)\}$$

  - Suppose that the other bidders follow the bidding strategy and the price level is b and k bidders dropped at $b_1, \cdots, b_k$.
  - Suppose that I win at this moment.
  - This means there are $n - k - 1$ bidders with signals $y = \beta_k^{-1}(b)$.
  - Then, the payoff to me is:

$$\begin{aligned} &\mathbb{E}\{u(V, x)|X_1 = x, Y_1 = \cdots = Y_{n-k-1} = y, Y_{n-k} = \beta_{k-1}^{-1}(b_k), \cdots, Y_{n-1} = \beta_0^{-1}(b_1)\} \\ &- \mathbb{E}\{u(V, x)|X_1 = y, Y_1 = \cdots = Y_{n-k-1} = y, Y_{n-k} = \beta_{k-1}^{-1}(b_k), \cdots, Y_{n-1} = \beta_0^{-1}(b_1)\}. \end{aligned} \tag{9.1}$$

  - This is positive if and only if $x > y$, because affiliation implies that the expected payoff is increasing in the own signal.
  - Thus, the above bidding strategy is the best response.

9.2.3 Reserve Price

- The seller does not sell the item if the winning bid is below $r > 0$.
- The participation threshold is:

$$x^*(r) = \inf\{x : \mathbb{E}[w(x, Y_1)|X_1 = x, Y_1 < x] \geq r\}.$$

- That is, a bidder participates if and only if $x \geq x^*(r)$.

9.2.4 Variations

- If PV, i.e., $w(x, y) = x$, then, the equilibrium strategy is to participate and drop at $\beta(x) = x$ if $x \geq r$ and not to participate if $x < r$.
- This is the unique equilibrium with weakly dominant strategies.
- Under the common value assumption, there can be many asymmetric equillibria [Milgrom, 1981, Bikhchandani and Riley, 1991].
- If bidders can call their bids, the game becomes more complicated because bidders can announce a "jump" bid at any time to signal their valuations [Avery, 1998].
- Be aware of the complications arising in the versions of the English auction.

9.2.5 Estimation of a IPV Button Auction

- Parameters of interest is F, the distribution of private signals and the payoff relevant random variable V.
- The data consists of $\{w_t, n_t, r_t\}_{t=1}^T$ if $m_t \geq 1$ for auction $t = 1, \cdots, T$:
  - w_t: the winning bid;
  - n_t: the number of potential bidders;
  - r_t: the reserve price;
  - m_t: the latent variable about the number of actual bidders.
- This is the case only the winning bid is observed but not the other bids.
- The winning bid is $w_t = \max\{x_{2:n_t}, r_t\}$, where $x_{2:n_t}$ is the second highest bid among n_t bids.

9.2.6 Likelihood of IPV Button Auction

- Donald and Paarsch [1996] estimate the model with a maximum likelihood estimator.
- Assume that $F_X(\cdot) = F_X(\cdot; \theta)$ with a finite dimensional parameter θ.
- The likelihood is:
  - If $m_t = 0$, the $F_X(r_t)^{n_t}$.
  - If $m_t = 1$ then $\mathbb{P}\{m_t = 1\} = n_t F_X(r_t)^{n_t - 1}[1 - F_X(r_t)]$.
  - If $m_t > 1$, then $h_t(w_t) := n_t(n_t - 1)F_X(w_t)^{n_t - 2}[1 - F_X(w_t)]f_X(w_t)$.
- The likelihood function is, if the data is only about the auctions with $m_t \geq 1$:

$$L = \prod_{t=1}^{T} \frac{h_t(w_t)^{1\{m_t > 1\}} \mathbb{P}\{m_t = 1\}}{1 - \mathbb{P}\{m_t = 0\}}$$

- The approach is still valid when the private signals are asymmetric and/or some bidders are not risk neutral, because $b(x) = x$ is still a dominant strategy.

9.2.7 Optimal Reserve Price

- The expected revenue to the seller who values the item as x_0 when setting the reserve price at r is:

$$R = x_0 F_X(r)^n + rnF_X(r)^{n-1}[1 - F_X(r)] + \int_r^{\overline{x}} wn(n-1)F_X(w)^{n-2}[1 - F_X(w)]f_X(w)dw.$$

- The first-order condition is:

$$r = x_0 + \frac{1 - F_X(r)}{f_X(r)}.$$

- Thus, the identification of F_X allows the seller to set the revenue maximizing reserve price.

9.2.8 Likelihood of IPV English Auction with Bid Data

- The likelihood function is:

$$L = \prod_{t=1}^{T} [1 - F_X(w_t)] \left[\prod_{i=2}^{m_t} f_X(b_{it}) \right] F_X^{n_t - m_t}(r_t).$$

- $b_{1t} \geq b_{2t} \geq \cdots \geq b_{m_t} \geq r$.
- If n_t is not observed to econometrician, the econometrician can:
  - assume $n_t = n$ and estimate n as a parameter;
  - assume $n = \max_{t=1,\cdots,T}\{m_t\}$;
  - assume n_t is drawn from a parametric distribution and estimate the parameters.

9.2.9 Observed Heterogeneity

- Let z_{it} be the observed attribute of bidder i in auction t.
- Assume that:

$$x_{it} = \alpha + \beta z_{it} + u_{it}.$$

- Then:

$$x_{it} \geq b_{it} \Leftrightarrow u_{it} \geq b_{it} - \alpha - z_{it}\beta := \tilde{b}_{it}.$$

- We can first regress b_{it} on z_{it} to estimate α and β to compute $\tilde{b}_{it}$.
- Then, the rest of the argument is the same as above by replacing b_{it} with $\tilde{b}_{it}$.

9.2.10 Identification

- Athey and Haile [2002] and Athey and Haile [2007] synthesize and extend the identification arguments of various auction models.
- Button auction with the symmetric IPV framework is non-parametrically identified only by the winning bid data.
- Button auction with the asymmetric IPV framework is non-parameterically identified by the winning bid and winner's identity data.
- The non-parametric identification can fail with a common value in general.
- The actual English auctions can be dirty and not easy to characterize the equilibrium: they are open cry auctions that signals their values, bidders may not indicate they are inactive at every highest bid, and there may be a minimum bid increment.
- Haile and Tamer [2003] considers a set identification of the signal distribution:
  1. signal is no less than the higher bid by the bidder: $x_i \geq b_i$.
  2. signal is no greater than the winning bid plus the minimum bid increment: $x_i \leq w + \Delta$.
  - Let $F_{i:n}$ be the distribution of the i-th highest order statistics from F_X.
  - Let $G_{i:n}$ be the empirical distribution of the i-th highest bids.
  - By 1, we have $F_{i:n}(x) \leq G_{i:n}(x)$.
  - By 2, we have $F_{2:n}(x) \geq G_{1:n}(x + \Delta)$.
  - These inequalities put the bounds on F_X.

9.3 First-Price Auctions

9.3.1 First-Price Sealed Bid Auction

- Each bidder independently submit a bid to the auctioneer.
- The high bidder wins and pays his bid.

9.3.2 Equilibrium Bidding Strategies

- Assume IPV.
- Then $x^*(r) = r$.
- Let β be the bid function that is increasing in the signal and η is the inverse of β.
- Suppose that the other bidders follow strategy β.
- The expected profit when a bidder with signal x submits a bid b is:

$$\pi(b, x) = (x - b)F_X[\eta(b)]^{n-1}.$$

- The first-order condition is:

$$(x - b)(n - 1)F_X[\eta(b)]^{n-2} f_X[\eta(b)]\eta'(b) - F_X[\eta(b)]^{n-1} = 0.$$

- If β is the equilibrium strategy, we have:

$$[x - \beta(x)](n - 1)F_X(x)^{n-2} f_X(x) - \beta'(x)F_X(x)^{n-1} = 0.$$

- Let $G(x) = F_X(x)^{n-1}$ for $x \geq r$ and $G(r) = 0$ and $g(x) = G'(x)$.
- Then, we have:

$$[x - \beta(x)]g(x) - \beta'(x)G(x) = 0.$$

- This is a linear differential equation such that:

$$\beta'(x) + p(x)\beta(x) = q(x),$$

  with a boundary condition:

$$\beta(r) = r,$$

  where

$$p(x) = \frac{g(x)}{G(x)},$$

  and

$$q(x) = x\frac{g(x)}{G(x)}.$$

- Let $\mu(x)$ be a function such that:

$$\mu(x)p(x) = \mu'(x).$$

- Multiply $\mu(x)$ to the both sides of the first-order condition to get:

$$\begin{aligned} &\mu(x)\beta'(x) + \mu(x)p(x)\beta(x) = \mu(x)q(x) \\ &\Leftrightarrow \mu(x)\beta'(x) + \mu'(x)\beta(x) = \mu(x)q(x) \\ &\Leftrightarrow [\mu(x)\beta(x)]' = \mu(x)q(x). \end{aligned}$$

- Hence,

$$\mu(x)\beta(x) = \mu(r)\beta(r) + \int_r^x \mu(t)q(t)dt.$$

- On the other hand,

$$[\ln \mu(x)]' = p(x).$$

- Hence,
$$\mu(x) = \mu(r)\exp\left(\int_r^x p(t)dt\right) = \exp\left(\int_r^x p(t)dt\right),$$
by setting $\mu(r) = 1$.
- Now,
$$\begin{aligned}\int_r^x p(t)dt &= \int_r^x \frac{g(t)}{G(t)}dt \\ &= [\ln G(t)]_r^x.\end{aligned}$$
- Hence,
$$\mu(x) = G(x).$$
- Inserting these results gives:
$$\begin{aligned}\beta(x) &= \frac{\int_r^x \mu(t)q(t)dt}{\mu(x)} \\ &= \frac{\int_r^x G(t)t\frac{g(t)}{G(t)}dt}{G(x)} \\ &= \frac{\int_r^x tg(t)dt}{G(x)} \\ &= \frac{[tG(t)]_r^t - \int_r^x G(t)dt}{G(x)} \\ &= x - \frac{\int_r^x G(t)dt}{G(x)} \\ &= x - \frac{\int_r^x F_X(t)^{n-1}dt}{F_X(x)^{n-1}}.\end{aligned}$$
- The term $-\frac{\int_r^x F_X(t)^{n-1}dt}{F_X(x)^{n-1}}$ is called the **markdown factor**, which is decreasing in the number of bidders n and increasing in the dispersion of the value distribution.
- The assumption of a binding reserve price ensures that there is a unique symmetric equilibrium [Athey and Haile, 2007].

9.3.3 Maximum Likelihood Estimation of the IPV First-Price Auction

- Donald and Paarsch [1993] proposed a maximum likelihood estimator.
- The data consists of $\{w_t, r_t, n_t\}_{t=1}^T$ for the sample where the number of actual bidders $m_t \geq 1$.
- The probability density of having w_t is:
$$\begin{aligned}h_t(w_t) &= n_t F_X[\eta_t(w_t)]^{n_t-1} f_X[\eta_t(w_t)]\eta_t'(w_t) \\ &= \frac{n_t F_X[\eta_t(w_t)]^{n_t}}{(n_t-1)[\eta_t(w_t) - w_t]},\end{aligned}$$
where the second equation is from the first-order condition.

Because the probability of $m_t \geq 1$ is $1 - F_X(r_t)^{n_t}$, the likelihood is:
$$L = \prod_{t=1}^T \frac{h_t(w_t)}{1 - F_X(r_t)^{n_t}}.$$

- To apply this approach, we may need to have a closed-form for η, and this may require to assume a specific functional-form for F_X.

9.3.4 Non-Parametric Approach

- Guerre et al. [2000] proposed a non-parametric approach.
- The data consists of $\{\{b_{it}\}_{i=1}^{m_t}, n_t, r_t\}_{t=1}^{T}$ and some observed covariates z_{it} for t with $m_t \geq 1$.
- Assume $n_t = n$, or in other words, focus on the data with the same number of potential bidders and estimate separately across different n.
- Let $H(b)$ be the distribution of the highest rival's bid and $h(b)$ be its density.
- Then, the expected payoff of bidding b when the signal is x is:

$$\pi(b, x) = (x - b)H(b).$$

- The first-order condition with respect to b is:

$$\begin{aligned}(x - b)h(b) - H(b) &= 0 \\ \Leftrightarrow x &= b + \frac{H(b)}{h(b)}\end{aligned}$$

  where the right-hand side is actually $\eta(b)$, the inverse of the bidding strategy $\beta(x)$.
- The idea is that $H(b)$ and $h(b)$ are directly identified from the data, and so, the value $\eta(b)$ can be computed for each bid.

9.3.5 Non-Parmaetric Approach: Estimation

- Note that:

$$H(b) = F_b(b)^{n-1},$$

  and

$$h(b) = (n-1)f_b(b)F_b(b)^{n-2},$$

  where f_b and F_b are the density and distribution of the bids.

1. Estimate $f(b)$ non-parametrically, say, by a kernel regression:

$$\hat{f}_b(b) = \frac{1}{TNh_b}\sum_{t=1}^{T}\sum_{i=1}^{n} K\left(\frac{b_{it} - b}{h_b}\right)$$

$$\widehat{F}_b(b) = \frac{\#\{b_{it} = r\}}{NT} + \int_r^b \hat{f}_b(t)dt.$$

   for $b > r$.
2. Form the implied x_{it} by:

$$\hat{x}_{it} = b_{it} + \frac{\widehat{H}(b_{it})}{\hat{h}(b_{it})}.$$

3. Estimate f_X non-parametrically, say, by a kernel regression:

$$\hat{f}_X(x) = \frac{1}{NTh_x}\sum_{t=1}^{T}\sum_{i=1}^{n} K\left(\frac{\hat{x}_{it} - x}{h_x}\right).$$

   and construct:

$$\widehat{F}_X(r) = \frac{\#\{\hat{x}_{it} = r\}}{NT} + \int_r^x \hat{f}_X(t)dt.$$

- For this argument to hold, it has to be that $\eta(b)$ is strictly increasing in b. Otherwise, for the same x, multiple b can be associated.
- This approach can be extended to the symmetric IPV and affiliated values.
- Krasnokutskaya [2011] considered a model with unobserved heterogeneity, in which the bidder's cost is $c_i = x_i v$ and x_i is private and independent and v is known among bidders but not to econometrician.

Chapter 10

Assignment 1: Basic Programming in R

The deadline is **February 18 1:30pm**.

Report the following results in html format using R markdown. In other words, replicate this document. You write functions in a separate R file and put in `R` folder in the project folder. Build the project as a package and load it from the R markdown file. The execution code sholuld be written in R markdown file.

You submit:

- R file containing functions.
- R markdown file containing your answers and executing codes.
- HTML report generated from the R markdown.

10.1 Simulate data

Consider to simulate data from the following model and estimate the parameters from the simulated data.

$$y_{ij} = 1\{j = \text{argmax}_{k=1,2}\beta x_k + \epsilon_{ik}\},$$

where ϵ_{ik} follows i.i.d. type-I extreme value distribution, $\beta = 0.2$, $x_1 = 0$ and $x_2 = 1$.

1. To simulate data, first make a data frame as follows:

```
## # A tibble: 2,000 x 3
##        i     k     x
##    <int> <int> <dbl>
##  1     1     1     0
##  2     1     2     1
##  3     2     1     0
##  4     2     2     1
##  5     3     1     0
##  6     3     2     1
##  7     4     1     0
##  8     4     2     1
##  9     5     1     0
## 10     5     2     1
## # ... with 1,990 more rows
```

2. Second, draw type-I extreme value random variables. Set the seed at 1. You can use `evd` package to draw the variables. You should get exactly the same realization if the seed is correctly set.

```
## # A tibble: 2,000 x 4
##        i     k     x       e
##    <int> <int> <dbl>   <dbl>
##  1     1     1     0  0.281
##  2     1     2     1 -0.167
##  3     2     1     0  1.93
##  4     2     2     1  1.97
##  5     3     1     0  0.830
##  6     3     2     1 -1.06
##  7     4     1     0 -0.207
##  8     4     2     1  0.617
##  9     5     1     0  0.0444
## 10     5     2     1  1.92
## # ... with 1,990 more rows
```

3. Third, compute the latent value of each option to obtain the following data frame:

```
## # A tibble: 2,000 x 5
##        i     k     x       e  latent
##    <int> <int> <dbl>   <dbl>   <dbl>
##  1     1     1     0  0.281   0.281
##  2     1     2     1 -0.167   0.0331
##  3     2     1     0  1.93    1.93
##  4     2     2     1  1.97    2.17
##  5     3     1     0  0.830   0.830
##  6     3     2     1 -1.06   -0.863
##  7     4     1     0 -0.207  -0.207
##  8     4     2     1  0.617   0.817
##  9     5     1     0  0.0444  0.0444
## 10     5     2     1  1.92    2.12
## # ... with 1,990 more rows
```

4. Finally, compute y by comparing the latent values of $k = 1, 2$ for each i to obtain the following result:

```
## # A tibble: 2,000 x 6
##        i     k     x       e  latent     y
##    <int> <int> <dbl>   <dbl>   <dbl> <dbl>
##  1     1     1     0  0.281   0.281      1
##  2     1     2     1 -0.167   0.0331     0
##  3     2     1     0  1.93    1.93       0
##  4     2     2     1  1.97    2.17       1
##  5     3     1     0  0.830   0.830      1
##  6     3     2     1 -1.06   -0.863      0
##  7     4     1     0 -0.207  -0.207      0
##  8     4     2     1  0.617   0.817      1
##  9     5     1     0  0.0444  0.0444     0
## 10     5     2     1  1.92    2.12       1
## # ... with 1,990 more rows
```

10.2 Estimate the parameter

Now you generated simulated data. Suppose you observe x_k and y_{ik} for each i and k and estimate β by a maximum likelihood estimator. The likelihood for i to choose k ($y_{ik} = 1$) can be shown to be:

$$p_{ik}(\beta) = \frac{\exp(\beta x_k)}{\exp(\beta x_1) + \exp(\beta x_2)}.$$

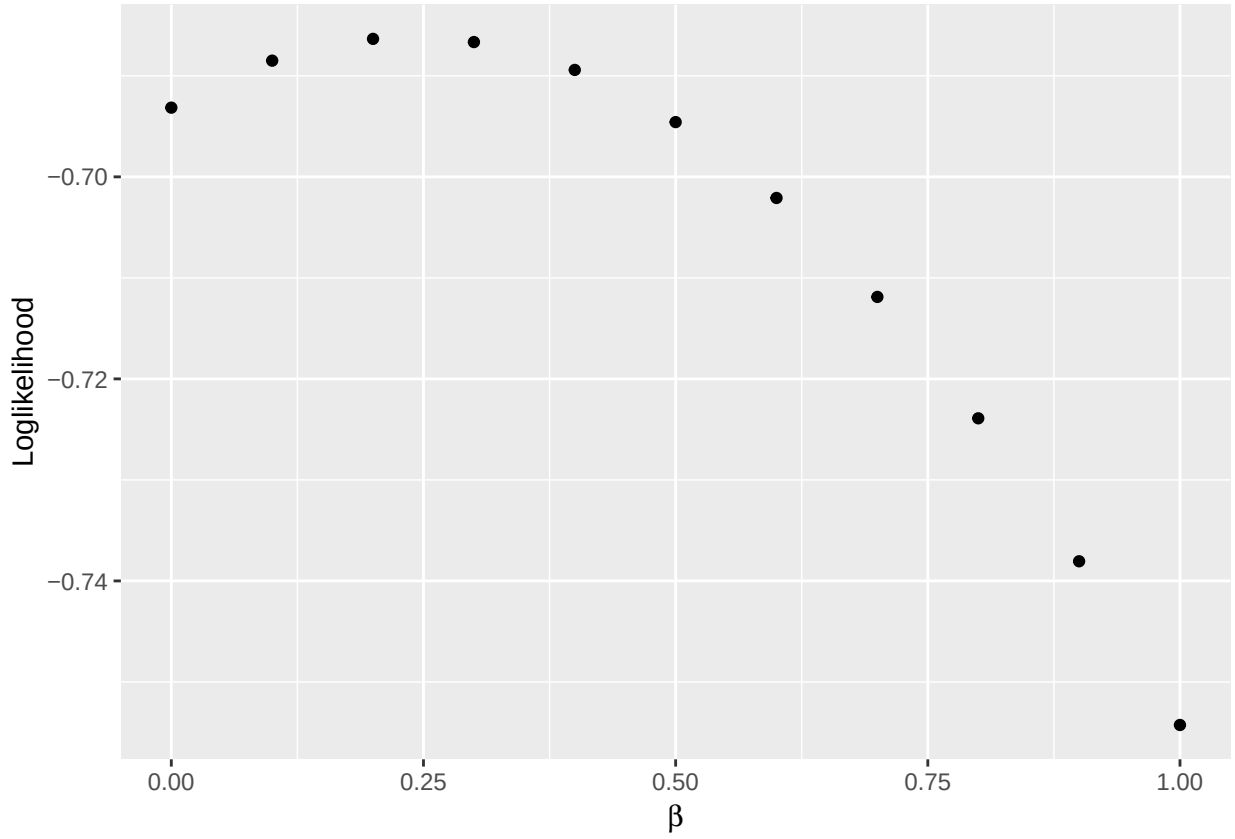
Then, the likelihood of observing $\{y_{ik}\}_{i,k}$ is:

$$L(\beta) = \prod_{i=1}^{1000} p_{i1}(\beta)^{y_{i1}} [1 - p_{i1}(\beta)]^{1-y_{i1}},$$

and the log likelihood is:

$$l(\beta) = \sum_{i=1}^{1000} \{y_{i1} \log p_{i1}(\beta) + (1 - y_{i1}) \log[1 - p_{i1}(\beta)\}.$$

1. Write a function to compute the livelihood for a given β and data and name the function `loglikelihood_A1`.
2. Compute the value of log likelihood for $\beta = 0, 0.1, \cdots, 1$ and plot the result using `ggplot2` packages. You can use `latex2exp` package to use LaTeX math symbol in the label:



1. Find and report β that maximizes the log likelihood for the simulated data. You can use `optim` function to achieve this. You will use `Brent` method and set the lower bound at -1 and upper bound at 1 for the parameter search.

```
## $par
## [1] 0.2371046
##
## $value
## [1] -0.6861689
##
## $counts
## function gradient
##       NA       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Chapter 11

Assignment 2: Production Function Estimation

The deadline is **February 28 1:30pm**.

11.1 Simulate data

Consider the following production and investment process for $j = 1, \cdots, 1000$ firms across $t = 1, \cdots, 10$ periods.

The log production function is of the form:

$$y_{jt} = \beta_0 + \beta_l l_{jt} + \beta_k k_{jt} + \omega_{jt} + \eta_{jt},$$

where ω_{jt} is an anticipated shock and η_{jt} is an ex post shock.

The anticipated shocks evolve as:

$$\omega_{jt} = \alpha \omega_{j,t-1} + \nu_{jt},$$

where ν_{jt} is an i.i.d. normal random variable with mean 0 and standard deviation σ_ν. The ex post shock is an i.i.d. normal random variable with mean 0 and standard deviation σ_η.

The product price the same across firms and normalized at 1. The price is normalized at 1. The wage w_t is constant at 0.5.

Finally, the capital accumulate according to:

$$K_{j,t+1} = (1 - \delta) K_{jt} + I_{jt}.$$

We set the parameters as follows:

parameter	variable	value
β_0	`beta_0`	1
β_l	`beta_l`	0.2
β_k	`beta_k`	0.7
α	`alpha`	0.7
σ_η	`sigma_eta`	0.2
σ_ν	`sigma_nu`	0.5
σ_w	`sigma_w`	0.1
δ	`delta`	0.05

1. Define the parameter variables as above.
2. Write a function that returns the log output given l_{jt}, k_{jt}, ω_{jt}, and η_{jt} under the given parameter values according to the above production function and name it `log_production(l, k, omega, eta, beta_0, beta_l, beta_k)`.

Suppose that the labor is determined after ω_{jt} is observed, but before η_{jt} is observed, given the log capital level k_{jt}.

3. Derive the optimal log labor as a function of ω_{jt}, η_{jt}, k_{jt}, and wage. Write a function to return the optimal log labor given the variables and parameters and name it `log_labor_choice(k, wage, omega, beta_0, beta_l, beta_k, sigma_eta)`.

As discussed in the class, if there is no additional variation in labor, the coefficient on the labor β_l is not identified. Thus, if we generate labor choice from the previous function, β_l will not be identified from the simulated data. To see this, we write a modified version of the previous function in which ω_{jt} is replaced with $\omega_{jt} + \iota_{jt}$, where ι_{jt} is an optimization error that follows an i.i.d. normal distribution with mean 0 and standard deviation 0.05. That is, the manager of the firm perceives as if the shock is $\omega_{jt} + \iota_{jt}$, even though the true shock is ω_{jt}.

4. Modify the previous function by including ι_{jt} as an additional input and name it `log_labor_choice_error(k, wage, omega, beta_0, beta_l, beta_k, iota, sigma_eta)`.

Consider an investment process such that:

$$I_{jt} = (\delta + \gamma\omega_{jt})K_{jt},$$

where I_{jt} and K_{jt} are investment and capital in level. Set $\gamma = 0.1$, i.e., the investment is strictly increasing in ω_{jt}. The investment function should be derived by solving the dynamic problem of a firm. But here, we just specify it in a reduced-form.

5. Define variable γ and assign it the value. Write a function that returns the investment given K_{jt}, ω_{jt}, and parameter values, according to the previous equation, and name it `investment_choice(k, omega, gamma, delta)`.

Simulate the data first using the labor choice without optimization error and second using the labor choice with optimization error. To do so, we specify the initial values for the state variables k_{jt} and ω_{jt} as follows.

6. Draw k_{j1} from an i.i.d. normal distribution with mean 1 and standard deviation 0.5. Draw ω_{j1} from its stationary distribution (check the stationary distribution of AR(1) process). Draw a wage. Before simulating the rest of the data, set the seed at 1.

```
## # A tibble: 1,000 x 5
##        j     t     k   omega  wage
##    <int> <dbl> <dbl>   <dbl> <dbl>
##  1     1     1     1 0.687  0.795    0.5
##  2     2     2     1 1.09   0.779    0.5
##  3     3     3     1 0.582 -0.610    0.5
##  4     4     4     1 1.80   0.148    0.5
##  5     5     5     1 1.16   0.0486   0.5
##  6     6     6     1 0.590 -1.16     0.5
##  7     7     7     1 1.24   0.568    0.5
##  8     8     8     1 1.37  -1.34     0.5
##  9     9     9     1 1.29  -0.873    0.5
## 10    10    10     1 0.847  0.699    0.5
## # ... with 990 more rows
```

7. Draw optimization error ι_{jt} and compute the labor and investment choice of period 1. For labor choice, compute both types of labor choices.

```
## # A tibble: 1,000 x 9
##        j     t     k   omega  wage     iota      l l_error       I
##    <int> <dbl> <dbl>   <dbl> <dbl>    <dbl>  <dbl>   <dbl>   <dbl>
##  1     1     1 0.687  0.795    0.5 -0.0443   1.72   1.67    0.257
##  2     2     1 1.09   0.779    0.5 -0.0961   2.06   1.94    0.381
##  3     3     1 0.582 -0.610    0.5  0.0810  -0.123 -0.0218 -0.0196
##  4     4     1 1.80   0.148    0.5  0.0260   1.89   1.92    0.391
##  5     5     1 1.16   0.0486   0.5 -0.00279  1.21   1.21    0.176
##  6     6     1 0.590 -1.16     0.5  0.0348  -0.809 -0.766  -0.120
##  7     7     1 1.24   0.568    0.5  0.00268  1.93   1.93    0.370
##  8     8     1 1.37  -1.34     0.5 -0.0655  -0.346 -0.428  -0.330
##  9     9     1 1.29  -0.873    0.5 -0.106    0.165  0.0327 -0.135
## 10    10     1 0.847  0.699    0.5 -0.0104   1.74   1.73    0.280
## # ... with 990 more rows
```

8. Draw ex post shock and compute the output according to the production function for both labor without optimization error and with optimization error. Name the output without optimization error `y` and the one with optimization error `y_error`.

```
## # A tibble: 1,000 x 12
##        j     t     k   omega  wage     iota      l l_error       I     eta
##    <int> <dbl> <dbl>   <dbl> <dbl>    <dbl>  <dbl>   <dbl>   <dbl>   <dbl>
##  1     1     1 0.687  0.795    0.5 -0.0443   1.72   1.67    0.257   0.148
##  2     2     1 1.09   0.779    0.5 -0.0961   2.06   1.94    0.381   0.0773
##  3     3     1 0.582 -0.610    0.5  0.0810  -0.123 -0.0218 -0.0196  0.259
##  4     4     1 1.80   0.148    0.5  0.0260   1.89   1.92    0.391  -0.161
##  5     5     1 1.16   0.0486   0.5 -0.00279  1.21   1.21    0.176  -0.321
##  6     6     1 0.590 -1.16     0.5  0.0348  -0.809 -0.766  -0.120   0.187
##  7     7     1 1.24   0.568    0.5  0.00268  1.93   1.93    0.370   0.361
##  8     8     1 1.37  -1.34     0.5 -0.0655  -0.346 -0.428  -0.330  -0.0113
##  9     9     1 1.29  -0.873    0.5 -0.106    0.165  0.0327 -0.135   0.377
## 10    10     1 0.847  0.699    0.5 -0.0104   1.74   1.73    0.280   0.316
## # ... with 990 more rows, and 2 more variables: y <dbl>, y_error <dbl>
```

9. Repeat this procedure for $t = 1, \cdots 10$ by updating the capital and anticipated shocks, and name the resulting data frame `df_T`.

```
## # A tibble: 10,000 x 13
##        j     t     k   omega  wage     iota      l l_error       I     eta
##    <int> <dbl> <dbl>   <dbl> <dbl>    <dbl>  <dbl>   <dbl>   <dbl>   <dbl>
##  1     1     1 0.687  0.795    0.5 -0.0443   1.72   1.67    0.257   0.148
##  2     2     1 1.09   0.779    0.5 -0.0961   2.06   1.94    0.381   0.0773
##  3     3     1 0.582 -0.610    0.5  0.0810  -0.123 -0.0218 -0.0196  0.259
##  4     4     1 1.80   0.148    0.5  0.0260   1.89   1.92    0.391  -0.161
##  5     5     1 1.16   0.0486   0.5 -0.00279  1.21   1.21    0.176  -0.321
##  6     6     1 0.590 -1.16     0.5  0.0348  -0.809 -0.766  -0.120   0.187
##  7     7     1 1.24   0.568    0.5  0.00268  1.93   1.93    0.370   0.361
##  8     8     1 1.37  -1.34     0.5 -0.0655  -0.346 -0.428  -0.330  -0.0113
##  9     9     1 1.29  -0.873    0.5 -0.106    0.165  0.0327 -0.135   0.377
## 10    10     1 0.847  0.699    0.5 -0.0104   1.74   1.73    0.280   0.316
## # ... with 9,990 more rows, and 3 more variables: y <dbl>, y_error <dbl>,
## #   nu <dbl>
```

10. Check the simulated data by making summary table.

	N	Mean	Sd	Min	Max
j	10000	500.5000000	288.6894251	1.0000000	1000.0000000
t	10000	5.5000000	2.8724249	1.0000000	10.0000000
k	10000	0.9797900	0.5838949	-1.2822534	3.2332312
omega	10000	-0.0055826	0.7025102	-2.5894171	2.6281307
wage	10000	0.5000000	0.0000000	0.5000000	0.5000000
iota	10000	-0.0000696	0.0502883	-0.1841453	0.1715419
l	10000	0.9799746	1.0965108	-3.3281023	4.9679634
l_error	10000	0.9798876	1.0971595	-3.3765433	4.9520674
I	10000	0.1793502	0.3006526	-1.2722627	3.2975332
eta	10000	0.0015825	0.2001539	-0.7650371	0.7455922
y	10000	1.8778479	1.1171035	-2.4680251	6.1228291
y_error	10000	1.8778305	1.1169266	-2.4777133	6.1196499
nu	10000	-0.0021155	0.4984324	-2.1513907	1.8253882

11.2 Estimate the parameters

For now, use the labor choice with optimization error.

1. First, simply regress y_{jt} on l_{jt} and k_{jt} using the least square method. This is likely to give an upwardly biased estimates on β_l and β_k. Why is it?

```
##
## Call:
## lm(formula = y_error ~ l_error + k, data = df_T)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.73002 -0.14117 -0.00071  0.13743  0.87983
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.892542   0.004058 219.966   <2e-16 ***
## l_error     0.997913   0.002396 416.454   <2e-16 ***
## k           0.007599   0.004503   1.688   0.0915 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2068 on 9997 degrees of freedom
## Multiple R-squared:  0.9657, Adjusted R-squared:  0.9657
## F-statistic: 1.408e+05 on 2 and 9997 DF,  p-value: < 2.2e-16
```

2. Second, take within-transformation on y_{jt}, l_{jt}, and k_{jt} and let Δy_{jt}, Δl_{jt}, and Δk_{jt} denote them. Then, regress Δy_{jt} on Δl_{jt}, and Δk_{jt} by the least squares method.

```
##
## Call:
## lm(formula = dy_error ~ -1 + dl_error + dk, data = df_T_within)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.72450 -0.13285 -0.00244  0.12931  0.77657
##
## Coefficients:
```

```
##            Estimate Std. Error t value Pr(>|t|)
## dl_error  0.9910916  0.0029548 335.413   <2e-16 ***
## dk       -0.0009029  0.0127539  -0.071    0.944
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1961 on 9998 degrees of freedom
## Multiple R-squared:  0.9184, Adjusted R-squared:  0.9184
## F-statistic: 5.629e+04 on 2 and 9998 DF,  p-value: < 2.2e-16
```

Next, we attempt to estimate the parameters using Olley-Pakes method. Estimate the first-step model of Olley-Pakes method:

$$y_{jt} = \beta_0 + \beta_1 l_{jt} + \phi(k_{jt}, I_{jt}) + \eta_{jt},$$

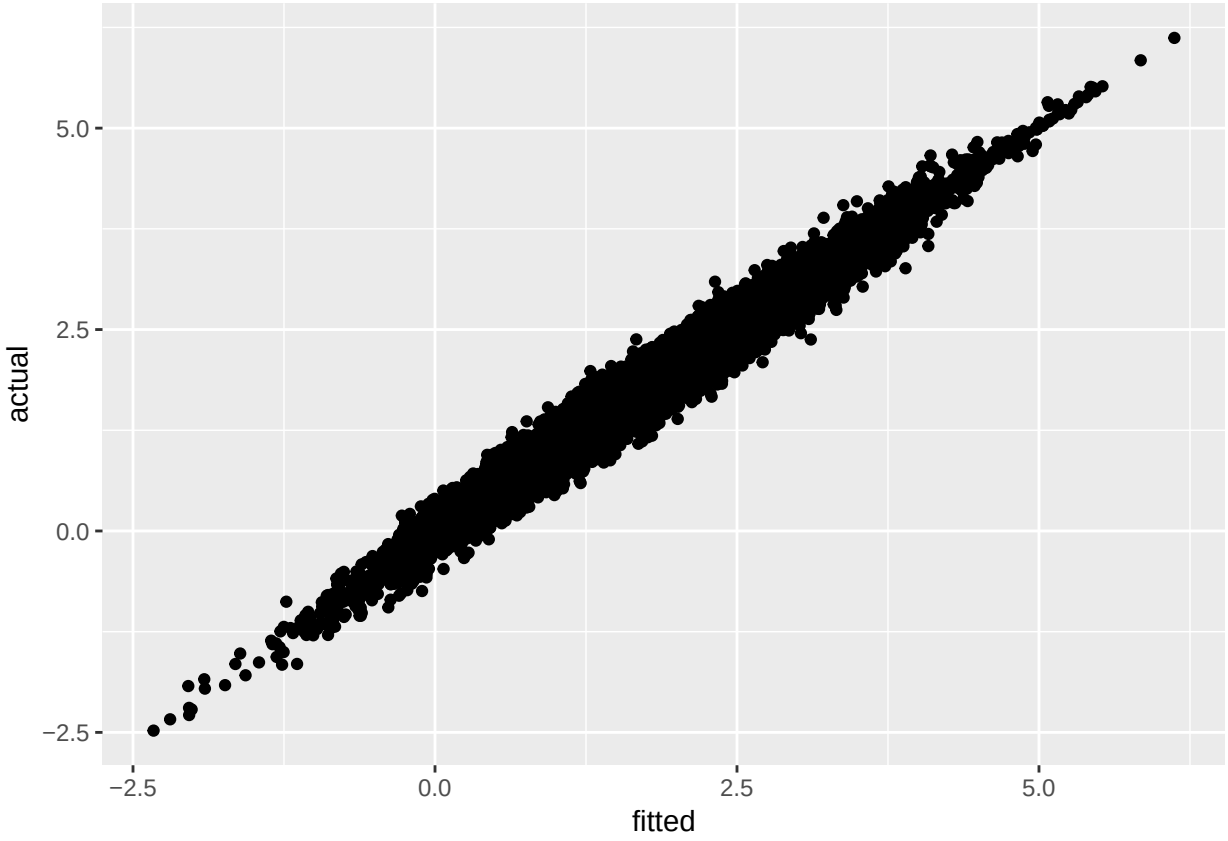
by approximating ϕ_t by a kernel function.

Remark that ϕ in general depends on observed and unobserved state variables. For this reason, in theory, ϕ should be estimated for each period. In this exercise, we assume ϕ is common across periods because we know that there is no unobserved state variables in the true data generating process. Moreover, we do not include w_t because we know that it is time -invariant. Do not forget to consider them in the actual data analysis.

You can use `npplreg` function of `np` package to estimate a partially linear model with a multivariate kernel. You first use `npplregbw` to obtain the optimal band width and then use `npplreg` to estimate the model under the optimal bandwidth. The computation of the optimal bandwidth is time consuming.

3. Return the summary of the first stage estimation and plot the fitted values against the data points.

```
##
## Partially Linear Model
## Regression data: 10000 training points, in 5 variable(s)
## With 3 linear parametric regressor(s), 2 nonparametric regressor(s)
##
##                      y(z)
## Bandwidth(s): 0.07355058 0.01435558
##
##                      x(z)
## Bandwidth(s): 0.03908594 0.01191551
##               0.01397428 3.74952954
##               0.83529394 0.00398329
##
##                   l_error        k        I
## Coefficient(s): 0.2485295 2.355522 5.345144
##
## Kernel Regression Estimator: Local-Constant
## Bandwidth Type: Fixed
##
## Residual standard error: 0.1934585
## R-squared: 0.970064
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 2
```



4. Check that β_l is not identified with the data without optimization error. Estimate the first stage model of Olley-Pakes with the labor choice without optimization error and report the result.

```
##
## Partially Linear Model
## Regression data: 10000 training points, in 5 variable(s)
## With 3 linear parametric regressor(s), 2 nonparametric regressor(s)
##
##                      y(z)
## Bandwidth(s): 0.07347226 0.01437256
##
##                      x(z)
## Bandwidth(s): 0.02960021 0.009986945
##               0.01397428 3.749529542
##               0.83529394 0.003983290
##
##                         l        k         I
## Coefficient(s): 1.180628 2.034182 0.7805077
##
## Kernel Regression Estimator: Local-Constant
## Bandwidth Type: Fixed
##
## Residual standard error: 0.1932285
## R-squared: 0.970116
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 2
```

Then, we estimate the second stage model of Olley-Pakes method:

$$y_{jt} - \hat{\beta}_l l_{jt} = \beta_0 + \beta_k k_{jt} + \alpha[\hat{\phi}(k_{j,t-1}, I_{j,t-1}) - \beta_0 - \beta_k k_{jt}] + \nu_{jt} + \eta_{jt}.$$

In this model, we do not have to non-parametetrically estimate the conditional expectation of ω_{jt} on $\omega_{j,t-1}$, because we know that the anticipated shock follows an AR(1) process. Remark that we in general have to non-parametrically estimate it.

The model is non-linear in parameters, because of the term $\alpha\beta_0$ and $\alpha\beta_k$. We estimate α, β_0, and β_k by a GMM estimator. The moment is:

$$g_{JT}(\alpha, \beta_0, \beta_k) \equiv \frac{1}{JT} \sum_{j=1}^{J} \sum_{t=1}^{T} \{y_{jt} - \hat{\beta}_l l_{jt} - \beta_0 - \beta_k k_{jt} - \alpha[\hat{\phi}(k_{j,t-1}, I_{j,t-1}) - \beta_0 - \beta_k k_{jt}]\} \begin{bmatrix} k_{jt} \\ k_{j,t-1} \\ I_{j,t-1} \end{bmatrix}.$$

5. Using the estimates in the first step, compute:

$$y_{jt} - \hat{\beta}_l l_{jt},$$

and:

$$\hat{\phi}(k_{j,t-1}, I_{j,t-1}),$$

for each j and t and save it as a data frame names `df_T_1st`.

```
## # A tibble: 10,000 x 4
##        j     t y_error_tilde phi_t_1
##    <int> <dbl>         <dbl>   <dbl>
##  1     1     1         2.34   NA
##  2     1     2         1.37    2.21
##  3     1     3         0.621   1.49
##  4     1     4         0.447   0.882
##  5     1     5         0.878   0.611
##  6     1     6         1.62    0.926
##  7     1     7         0.558   1.40
##  8     1     8         0.684   0.439
##  9     1     9         0.939   0.520
## 10     1    10         1.49    0.836
## # ... with 9,990 more rows
```

6. Compute a function that returns the value of $g_{JT}(\alpha, \beta_0, \beta_k)$ given parameter values, data, and `df_T_1st`, and name it `moment_OP_2nd`. Show the values of the moments evaluated at the true parameters.

```
## [1] -0.018507303 -0.019038229 -0.003867714
```

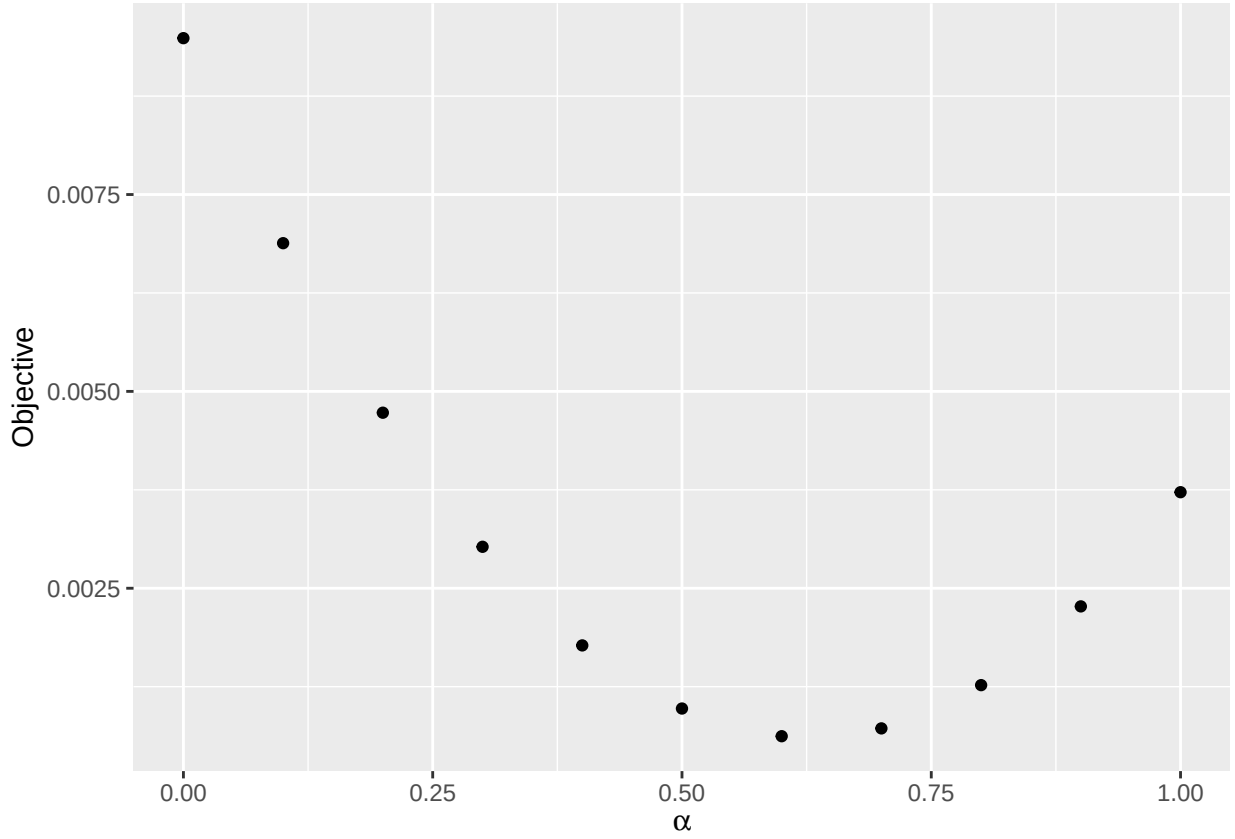
Based on the moment, we can define the objective function of a generalized method of moments estimator with a weighting matrix W as:

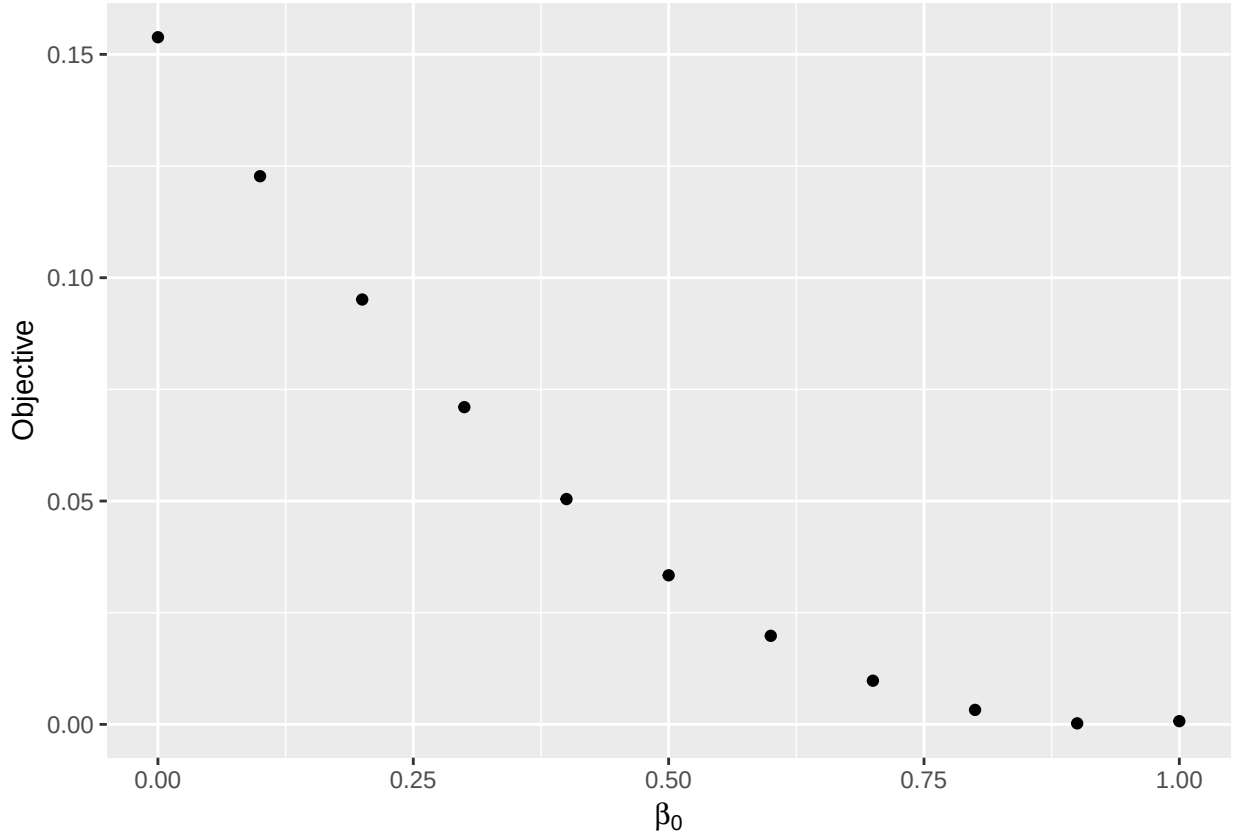
$$Q_{JT}(\alpha, \beta_0, \beta_k) \equiv g_{JT}(\alpha, \beta_0, \beta_k)' W g_{JT}(\alpha, \beta_0, \beta_k).$$

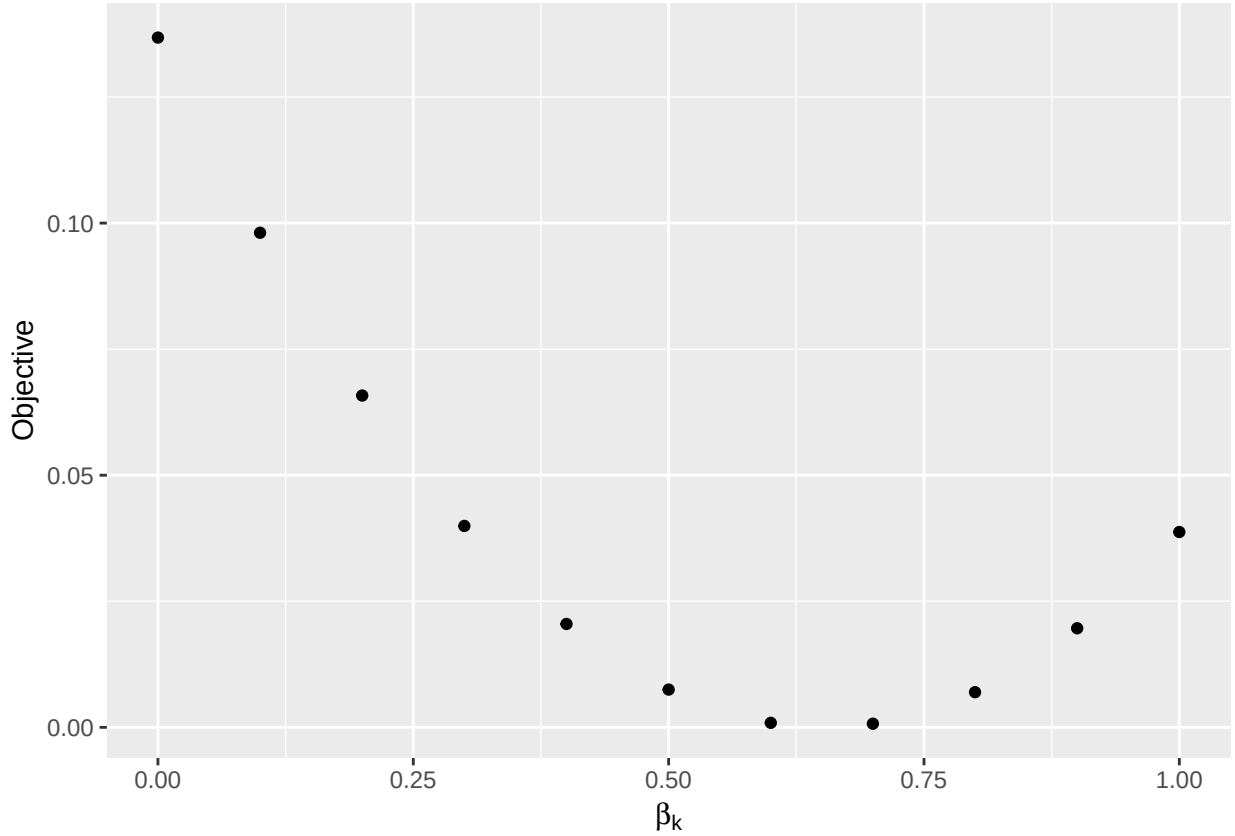
7. Write a function that returns the value of $Q_{JT}(\alpha, \beta_0, \beta_k)$ given the vector of parameter values, data, and `df_T_1st`, and name it `objective_OP_2nd`. Setting W at the identity matrix, show the value of the objective function evaluated at the true parameters.

```
##               [,1]
## [1,] 0.0007199336
```

8. Draw the graph of the objective function when one of α, β_0, and β_k are changed from 0 to 1 by 0.1 while the others are set at the true value. Is the objective function minimized at around the true value?


Objective
0.0075
0.0050
0.0025
0.00
0.25
0.50
0.75
1.00
α


Objective
0.15
0.10
0.05
0.00
0.00
0.25
0.50
0.75
1.00
β0



9. Find the parameters that minimize the objective function using `optim`. You may use `L-BFGS-B` method to solve it.

```
## $par
## [1] 0.7020260 0.9766308 0.6693945
##
## $value
## [1] 1.994601e-07
##
## $counts
## function gradient
##       10       10
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

Chapter 12

Assignment 3: Demand Function Estimation I

The deadline is **March 11 1:30pm**.

12.1 Simulate data

We simulate data from a discrete choice model. There are T markets and each market has N consumers. There are J products and the indirect utility of consumer i in market t for product j is:

$$u_{itj} = \beta_{it}' x_j + \alpha_{it} p_{jt} + \xi_{jt} + \epsilon_{ijt},$$

where ϵ_{ijt} is an i.i.d. type-I extreme random variable. x_j is K-dimensional observed characteristics of the product. p_{jt} is the retail price of the product in the market.

ξ_{jt} is product-market specific fixed effect. p_{jt} can be correlated with ξ_{jt} but x_{jt}s are independent of ξ_{jt}. $j = 0$ is an outside option whose indirect utility is:

$$u_{it0} = \epsilon_{i0t},$$

where ϵ_{i0t} is an i.i.d. type-I extreme random variable.

β_{it} and α_{it} are different across consumers, and they are distributed as:

$$\beta_{itk} = \beta_{0k} + \sigma_k \nu_{itk},$$

$$\alpha_{it} = -\exp(\mu + \omega \upsilon_{it}) = -\exp(\mu + \frac{\omega^2}{2}) + [-\exp(\mu + \omega \upsilon_{it}) + \exp(\mu + \frac{\omega^2}{2})] \equiv \alpha_0 + \tilde{\alpha}_{it},$$

where ν_{itk} for $k = 1, \cdots, K$ and υ_{it} are i.i.d. standard normal random variables. α_0 is the mean of α_i and $\tilde{\alpha}_i$ is the deviation from the mean.

Given a choice set in the market, $\mathcal{J}_t \cup \{0\}$, a consumer chooses the alternative that maximizes her utility:

$$q_{ijt} = 1\{u_{ijt} = \max_{k \in \mathcal{J}_t \cup \{0\}} u_{ikt}\}.$$

The choice probability of product j for consumer i in market t is:

$$\sigma_{jt}(p_t, x_t, \xi_t) = \mathbb{P}\{u_{ijt} = \max_{k \in \mathcal{J}_t \cup \{0\}} u_{ikt}\}.$$

Suppose that we only observe the share data:

$$s_{jt} = \frac{1}{N}\sum_{i=1}^{N} q_{ijt},$$

along with the product-market characteristics x_{jt} and the retail prices p_{jt} for $j \in \mathcal{J}_t \cup \{0\}$ for $t = 1, \cdots, T$. We do not observe the choice data q_{ijt} nor shocks $\xi_{jt}, \nu_{it}, \upsilon_{it}, \epsilon_{ijt}$.

In this assignment, we consider a model with $\xi_{jt} = 0$, i.e., the model without the unobserved fixed effects. However, the code to simulate data should be written for general ξ_{jt}, so that we can use the same code in the next assignment in which we consider a model with the unobserved fixed effects.

1. Set the seed, constants, and parameters of interest as follows.

```
# set the seed
set.seed(1)
# number of products
J <- 10
# dimension of product characteristics including the intercept
K <- 3
# number of markets
T <- 100
# number of consumers per market
N <- 500
# number of Monte Carlo
L <- 500
```

```
# set parameters of interests
beta <- rnorm(K);
beta[1] <- 4
beta
```

```
## [1]  4.0000000  0.1836433 -0.8356286
```

```
sigma <- abs(rnorm(K)); sigma
```

```
## [1] 1.5952808 0.3295078 0.8204684
```

```
mu <- 0.5
omega <- 1
```

Generate the covariates as follows.

The product-market characteristics:

$$x_{j1} = 1, x_{jk} \sim N(0, \sigma_x), k = 2, \cdots, K,$$

where σ_x is referred to as `sd_x` in the code.

The product-market-specific unobserved fixed effect:

$$\xi_{jt} = 0.$$

The marginal cost of product j in market t:

$$c_{jt} \sim \text{logNormal}(0, \sigma_c),$$

where σ_c is referred to as `sd_c` in the code.

The retail price:

$$p_{jt} - c_{jt} \sim \text{logNorm}(\gamma\xi_{jt}, \sigma_p),$$

where γ is referred to as `price_xi` and σ_p as `sd_p` in the code. This price is not the equilibrium price. We will revisit this point in a subsequent assignment.

The value of the auxiliary parameters are set as follows:

```
# set auxiliary parameters
price_xi <- 1
prop_jt <- 0.6
sd_x <- 0.5
sd_c <- 0.05
sd_p <- 0.05
```

2. `X` is the data frame such that a row contains the characteristics vector x_j of a product and columns are product index and observed product characteristics. The dimension of the characteristics K is specified above. Add the row of the outside option whose index is 0 and all the characteristics are zero.

```
X
```

```
## # A tibble: 11 x 4
##        j   x_1     x_2      x_3
##    <dbl> <dbl>   <dbl>    <dbl>
##  1     0     0  0       0
##  2     1     1  0.244  -0.00810
##  3     2     1  0.369   0.472
##  4     3     1  0.288   0.411
##  5     4     1 -0.153   0.297
##  6     5     1  0.756   0.459
##  7     6     1  0.195   0.391
##  8     7     1 -0.311   0.0373
##  9     8     1 -1.11   -0.995
## 10     9     1  0.562   0.310
## 11    10     1 -0.0225 -0.0281
```

3. `M` is the data frame such that a row contains the price ξ_{jt}, marginal cost c_{jt}, and price p_{jt}. After generating the variables, drop `1 - prop_jt` products from each market using `dplyr::sample_frac` function. The variation in the available products is important for the identification of the distribution of consumer-level unobserved heterogeneity. Add the row of the outside option to each market whose index is 0 and all the variables take value zero.

```
M
```

```
## # A tibble: 700 x 5
##        j     t    xi     c     p
##    <dbl> <int> <dbl> <dbl> <dbl>
##  1     0     1     0 0      0
##  2     1     1     0 0.951  1.93
##  3     5     1     0 0.974  1.94
##  4     6     1     0 0.980  1.96
##  5     7     1     0 0.961  1.94
##  6     8     1     0 0.989  1.99
##  7    10     1     0 1.02   2.09
##  8     0     2     0 0      0
##  9     1     2     0 0.988  2.09
## 10     2     2     0 1.04   1.96
## # ... with 690 more rows
```

4. Generate the consumer-level heterogeneity. `V` is the data frame such that a row contains the vector of shocks to consumer-level heterogeneity, (ν_i', υ_i). They are all i.i.d. standard normal random variables.

```
V
```

```
## # A tibble: 50,000 x 6
##        i     t   v_x_1   v_x_2  v_x_3    v_p
##    <int> <int>   <dbl>   <dbl>  <dbl>  <dbl>
##  1     1     1  1.02    0.731  -0.169 -1.40
##  2     2     1  0.375   0.418  -0.243 -0.899
##  3     3     1 -1.14    0.257  -2.56   1.44
##  4     4     1 -0.752   0.449   0.718  0.497
##  5     5     1  3.06    0.355   0.652  2.02
##  6     6     1  1.44   -0.0302  0.585  0.406
##  7     7     1  0.323  -0.363  -0.441  0.618
##  8     8     1 -0.107   0.392   0.823  1.56
##  9     9     1 -0.0515  0.733  -0.454  1.30
## 10    10     1  0.790   0.468   1.10  -0.241
## # ... with 49,990 more rows
```

5. Join `X`, `M`, `V` using `dplyr::left_join` and name it `df`. `df` is the data frame such that a row contains variables for a consumer about a product that is available in a market.

```
df
```

```
## # A tibble: 350,000 x 13
##        t     i     j v_x_1 v_x_2  v_x_3    v_p   x_1     x_2      x_3    xi
##    <int> <int> <dbl> <dbl> <dbl>  <dbl>  <dbl> <dbl>   <dbl>    <dbl> <dbl>
##  1     1     1     0 1.02  0.731 -0.169 -1.40      0  0       0           0
##  2     1     1     1 1.02  0.731 -0.169 -1.40      1  0.244  -0.00810     0
##  3     1     1     5 1.02  0.731 -0.169 -1.40      1  0.756   0.459       0
##  4     1     1     6 1.02  0.731 -0.169 -1.40      1  0.195   0.391       0
##  5     1     1     7 1.02  0.731 -0.169 -1.40      1 -0.311   0.0373      0
##  6     1     1     8 1.02  0.731 -0.169 -1.40      1 -1.11   -0.995       0
##  7     1     1    10 1.02  0.731 -0.169 -1.40      1 -0.0225 -0.0281      0
##  8     1     2     0 0.375 0.418 -0.243 -0.899     0  0       0           0
##  9     1     2     1 0.375 0.418 -0.243 -0.899     1  0.244  -0.00810     0
## 10     1     2     5 0.375 0.418 -0.243 -0.899     1  0.756   0.459       0
## # ... with 349,990 more rows, and 2 more variables: c <dbl>, p <dbl>
```

6. Draw a vector of preference shocks `e` whose length is the same as the number of rows of `df`.

```
head(e)
```

```
## [1] -0.01971328 -0.44401874  0.15952459  0.17658106 -0.55495888 -0.12854864
```

7. Write a function `compute_indirect_utility(df, beta, sigma, mu, omega)` that returns a vector whose element is the mean indirect utility of a product for a consumer in a market. The output should have the same length with e.

```
# compute indirect utility
u <-
  compute_indirect_utility(
    df, beta, sigma,
          mu, omega)
head(u)
```

```
##             u
## [1,] 0.000000
## [2,] 4.957950
## [3,] 4.716943
```

```
## [4,] 4.537668
## [5,] 4.672690
## [6,] 5.322723
```

8. Write a function `compute_choice(X, M, V, e, beta, sigma, mu, omega)` that first construct `df` from `X`, `M`, `V`, second call `compute_indirect_utility` to obtain the vector of mean indirect utilities `u`, third compute the choice vector `q` based on the vector of mean indirect utilities and `e`, and finally return the data frame to which `u` and `q` are added as columns.

```
# compute choice
df_choice <-
  compute_choice(X, M, V, e, beta, sigma,
                 mu, omega)
df_choice
```

```
## # A tibble: 350,000 x 16
##        t     i     j v_x_1 v_x_2  v_x_3    v_p   x_1     x_2      x_3    xi
##    <int> <int> <dbl> <dbl> <dbl>  <dbl>  <dbl> <dbl>   <dbl>    <dbl> <dbl>
##  1     1     1     0 1.02  0.731 -0.169 -1.40      0  0       0           0
##  2     1     1     1 1.02  0.731 -0.169 -1.40      1  0.244  -0.00810     0
##  3     1     1     5 1.02  0.731 -0.169 -1.40      1  0.756   0.459       0
##  4     1     1     6 1.02  0.731 -0.169 -1.40      1  0.195   0.391       0
##  5     1     1     7 1.02  0.731 -0.169 -1.40      1 -0.311   0.0373      0
##  6     1     1     8 1.02  0.731 -0.169 -1.40      1 -1.11   -0.995       0
##  7     1     1    10 1.02  0.731 -0.169 -1.40      1 -0.0225 -0.0281      0
##  8     1     2     0 0.375 0.418 -0.243 -0.899     0  0       0           0
##  9     1     2     1 0.375 0.418 -0.243 -0.899     1  0.244  -0.00810     0
## 10     1     2     5 0.375 0.418 -0.243 -0.899     1  0.756   0.459       0
## # ... with 349,990 more rows, and 5 more variables: c <dbl>, p <dbl>,
## #   u <dbl>, e <dbl>, q <dbl>
```

```
summary(df_choice)
```

```
##        t                i                j              v_x_1
##  Min.   :  1.00   Min.   :  1.0   Min.   : 0.000   Min.   :-4.302781
##  1st Qu.: 25.75   1st Qu.:125.8   1st Qu.: 2.000   1st Qu.:-0.685716
##  Median : 50.50   Median :250.5   Median : 5.000   Median : 0.000103
##  Mean   : 50.50   Mean   :250.5   Mean   : 4.639   Mean   :-0.004312
##  3rd Qu.: 75.25   3rd Qu.:375.2   3rd Qu.: 7.000   3rd Qu.: 0.668186
##  Max.   :100.00   Max.   :500.0   Max.   :10.000   Max.   : 3.809895
##      v_x_2               v_x_3                v_p
##  Min.   :-4.542122   Min.   :-3.957618   Min.   :-4.218131
##  1st Qu.:-0.678436   1st Qu.:-0.674487   1st Qu.:-0.670251
##  Median : 0.000444   Median : 0.005891   Median : 0.002309
##  Mean   :-0.001340   Mean   : 0.003736   Mean   :-0.001305
##  3rd Qu.: 0.670840   3rd Qu.: 0.678349   3rd Qu.: 0.671041
##  Max.   : 4.313621   Max.   : 4.244194   Max.   : 4.074300
##       x_1              x_2                x_3                xi
##  Min.   :0.0000   Min.   :-1.10735   Min.   :-0.9947   Min.   :0
##  1st Qu.:1.0000   1st Qu.:-0.15269   1st Qu.: 0.0000   1st Qu.:0
##  Median :1.0000   Median : 0.19492   Median : 0.2970   Median :0
##  Mean   :0.8571   Mean   : 0.06936   Mean   : 0.1186   Mean   :0
##  3rd Qu.:1.0000   3rd Qu.: 0.36916   3rd Qu.: 0.4106   3rd Qu.:0
##  Max.   :1.0000   Max.   : 0.75589   Max.   : 0.4719   Max.   :0
##        c                p                u                   e
##  Min.   :0.0000   Min.   :0.000   Min.   :-200.871   Min.   :-2.6364
```

```
##  1st Qu.:0.9417   1st Qu.:1.921   1st Qu.:  -2.202   1st Qu.:-0.3302
##  Median :0.9887   Median :1.986   Median :   0.000   Median : 0.3634
##  Mean   :0.8583   Mean   :1.718   Mean   :  -1.316   Mean   : 0.5760
##  3rd Qu.:1.0278   3rd Qu.:2.046   3rd Qu.:   1.961   3rd Qu.: 1.2415
##  Max.   :1.1996   Max.   :2.192   Max.   :  10.731   Max.   :14.0966
##        q
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.1429
##  3rd Qu.:0.0000
##  Max.   :1.0000
```

9. Write a function `compute_share(X, M, V, e, beta, sigma, mu, omega)` that first construct `df` from `X`, `M`, `V`, second call `compute_choice` to obtain a data frame with `u` and `q`, third compute the share of each product at each market `s` and the log difference in the share from the outside option, $\ln(s_{jt}/s_{0t})$, denoted by `y`, and finally return the data frame that is summarized at the product-market level, dropped consumer-level variables, and added `s` and `y`.

```
# compute share
df_share <-
  compute_share(X, M, V, e, beta, sigma,
                mu, omega)
df_share
```

```
## # A tibble: 700 x 11
##        t     j   x_1     x_2      x_3    xi     c     p     q     s      y
##    <int> <dbl> <dbl>   <dbl>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
##  1     1     0     0  0       0            0 0       0      153 0.306  0
##  2     1     1     1  0.244  -0.00810      0 0.951   1.93    49 0.098 -1.14
##  3     1     5     1  0.756   0.459        0 0.974   1.94    38 0.076 -1.39
##  4     1     6     1  0.195   0.391        0 0.980   1.96    41 0.082 -1.32
##  5     1     7     1 -0.311   0.0373       0 0.961   1.94    45 0.09  -1.22
##  6     1     8     1 -1.11   -0.995        0 0.989   1.99   131 0.262 -0.155
##  7     1    10     1 -0.0225 -0.0281       0 1.02    2.09    43 0.086 -1.27
##  8     2     0     0  0       0            0 0       0      170 0.34   0
##  9     2     1     1  0.244  -0.00810      0 0.988   2.09    50 0.1   -1.22
## 10     2     2     1  0.369   0.472        0 1.04    1.96    37 0.074 -1.52
## # ... with 690 more rows
```

```
summary(df_share)
```

```
##        t               j              x_1              x_2
##  Min.   :  1.00   Min.   : 0.000   Min.   :0.0000   Min.   :-1.10735
##  1st Qu.: 25.75   1st Qu.: 2.000   1st Qu.:1.0000   1st Qu.:-0.15269
##  Median : 50.50   Median : 5.000   Median :1.0000   Median : 0.19492
##  Mean   : 50.50   Mean   : 4.639   Mean   :0.8571   Mean   : 0.06936
##  3rd Qu.: 75.25   3rd Qu.: 7.000   3rd Qu.:1.0000   3rd Qu.: 0.36916
##  Max.   :100.00   Max.   :10.000   Max.   :1.0000   Max.   : 0.75589
##       x_3                xi          c                p
##  Min.   :-0.9947   Min.   :0   Min.   :0.0000   Min.   :0.000
##  1st Qu.: 0.0000   1st Qu.:0   1st Qu.:0.9417   1st Qu.:1.921
##  Median : 0.2970   Median :0   Median :0.9887   Median :1.986
##  Mean   : 0.1186   Mean   :0   Mean   :0.8583   Mean   :1.718
##  3rd Qu.: 0.4106   3rd Qu.:0   3rd Qu.:1.0278   3rd Qu.:2.046
##  Max.   : 0.4719   Max.   :0   Max.   :1.1996   Max.   :2.192
```

```
##        q                s                y          
##  Min.   : 25.00   Min.   :0.0500   Min.   :-1.9459  
##  1st Qu.: 43.00   1st Qu.:0.0860   1st Qu.:-1.3636  
##  Median : 51.00   Median :0.1020   Median :-1.1579  
##  Mean   : 71.43   Mean   :0.1429   Mean   :-0.9968  
##  3rd Qu.: 73.00   3rd Qu.:0.1460   3rd Qu.:-0.8316  
##  Max.   :191.00   Max.   :0.3820   Max.   : 0.0000
```

12.2 Estimate the parameters

1. Estimate the parameters assuming there is no consumer-level heterogeneity, i.e., by assuming:

$$\ln \frac{s_{jt}}{s_{0t}} = \beta' x_{jt} + \alpha p_{jt}.$$

   This can be implemented using `lm` function. Print out the estimate results.

```
## 
## Call:
## lm(formula = y ~ -1 + x_1 + x_2 + x_3 + p, data = df_share)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -0.5777 -0.1051  0.0000  0.1042  0.4913 
## 
## Coefficients:
##     Estimate Std. Error t value Pr(>|t|)    
## x_1  0.97770    0.19287   5.069 5.13e-07 ***
## x_2  0.17795    0.02945   6.043 2.46e-09 ***
## x_3 -0.87591    0.03482 -25.159  < 2e-16 ***
## p   -1.01500    0.09613 -10.559  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.1731 on 696 degrees of freedom
## Multiple R-squared:  0.9765, Adjusted R-squared:  0.9764 
## F-statistic:  7237 on 4 and 696 DF,  p-value: < 2.2e-16
```

We estimate the model using simulated share.

When optimizing an objective function that uses the Monte Carlo simulation, it is important to keep the realizations of the shocks the same across the evaluations of the objective function. If the realization of the shocks differ across the objective function evaluations, the optimization algorithm will not converge because it cannot distinguish the change in the value of the objective function due to the difference in the parameters and the difference in the realized shocks.

The best practice to avoid this problem is to generate the shocks outside the optimization algorithm as in the current case. If the size of the shocks can be too large to store in the memory, the second best practice is to make sure to set the seed inside the optimization algorithm so that the realized shocks are the same across function evaluations.

2. For this reason, we first draw Monte Carlo consumer-level heterogeneity `V_mcmc` and Monte Carlo preference shocks `e_mcmc`. The number of simulations is `L`. This does not have to be the same with the actual number of consumers `N`.

```
V_mcmc
```

```
## # A tibble: 50,000 x 6
##        i     t  v_x_1  v_x_2   v_x_3     v_p
##    <int> <int>  <dbl>  <dbl>   <dbl>   <dbl>
##  1     1     1 -1.07  -1.30   2.32    0.110
##  2     2     1 -0.730  0.684  1.07   -0.802
##  3     3     1 -0.437 -0.243  0.383  -0.318
##  4     4     1 -0.979  0.520  1.02    0.637
##  5     5     1  0.487 -0.991  0.0422  0.613
##  6     6     1 -0.805  1.15   1.08   -0.473
##  7     7     1  0.761  0.353 -2.05   -0.989
##  8     8     1  0.965  1.76  -1.34   -0.686
##  9     9     1  0.702 -0.583  0.144  -0.0259
## 10    10     1  0.213  1.60  -1.32    1.72
## # ... with 49,990 more rows
```

```
head(e_mcmc)
```

```
## [1]  0.8830453  1.1151824  3.2225788 -0.9125983  0.8022472  5.1145476
```

3. Use `compute_share` to check the simulated share at the true parameter using the Monte Carlo shocks. Remember that the number of consumers should be set at `L` instead of `N`.

```
df_share_mcmc
```

```
## # A tibble: 700 x 11
##        t     j   x_1     x_2      x_3    xi     c     p     q     s      y
##    <int> <dbl> <dbl>   <dbl>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
##  1     1     0     0  0       0           0 0      0      153 0.306  0
##  2     1     1     1  0.244  -0.00810     0 0.951  1.93    59 0.118 -0.953
##  3     1     5     1  0.756   0.459       0 0.974  1.94    46 0.092 -1.20
##  4     1     6     1  0.195   0.391       0 0.980  1.96    39 0.078 -1.37
##  5     1     7     1 -0.311   0.0373      0 0.961  1.94    51 0.102 -1.10
##  6     1     8     1 -1.11   -0.995       0 0.989  1.99   107 0.214 -0.358
##  7     1    10     1 -0.0225 -0.0281      0 1.02   2.09    45 0.09  -1.22
##  8     2     0     0  0       0           0 0      0      164 0.328  0
##  9     2     1     1  0.244  -0.00810     0 0.988  2.09    51 0.102 -1.17
## 10     2     2     1  0.369   0.472       0 1.04   1.96    33 0.066 -1.60
## # ... with 690 more rows
```

5. Vectorize the parameters to a vector `theta` because `optim` requires the maximiand to be a vector.

```
# set parameters
theta <- c(beta, sigma, mu, omega)
theta
```

```
## [1]  4.0000000  0.1836433 -0.8356286  1.5952808  0.3295078  0.8204684
## [7]  0.5000000  1.0000000
```

6. Write a function `NLLS_objective_A3(theta, df_share, X, M, V_mcmc, e_mcmc)` that first computes the simulated share and then compute the mean-squared error between the share data.

```
NLLS_objective
```

```
## [1] 0.0004878743
```

7. Draw a graph of the objective function that varies each parameter from 0.5, 0.6, $\cdots$, 1.5 of the true value. First try with the actual shocks `V` and `e` and then try with the Monte Carlo shocks `V_mcmc` and

`e_mcmc`. You will some of the graph does not look good with the Monte Carlo shocks. It will cause the approximation error.

Because this takes time, you may want to parallelize the computation using `%dopar` functionality of `foreach` loop. To do so, first install `doParallel` package and then load it and register the workers as follows:

```
registerDoParallel()
```

This automatically detect the number of cores available at your computer and registers them as the workers. Then, you only have to change `%do%` to `%dopar` in the `foreach` loop as follows:

```
foreach (i = 1:4) %dopar% {
  # this part is parallelized
  y <- 2 * i
  return(y)
}
```

```
## [[1]]
## [1] 2
##
## [[2]]
## [1] 4
##
## [[3]]
## [1] 6
##
## [[4]]
## [1] 8
```

In windows, you may have to explicitly pass packages, functions, and data to the worker by using `.export` and `.packages` options as follows:

```
temp_func <- function(x) {
  y <- 2 * x
  return(y)
}
foreach (i = 1:4,
         .export = "temp_func",
         .packages = "magrittr") %dopar% {
  # this part is parallelized
  y <- temp_func(i)
  return(y)
}
```

```
## [[1]]
## [1] 2
##
## [[2]]
## [1] 4
##
## [[3]]
## [1] 6
##
## [[4]]
## [1] 8
```

If you have called a function in a package in this way `dplyr::mutate`, then you will not have to pass `dplyr` by `.packages` option. This is one of the reasons why I prefer to explicitly call the every time I call a function.

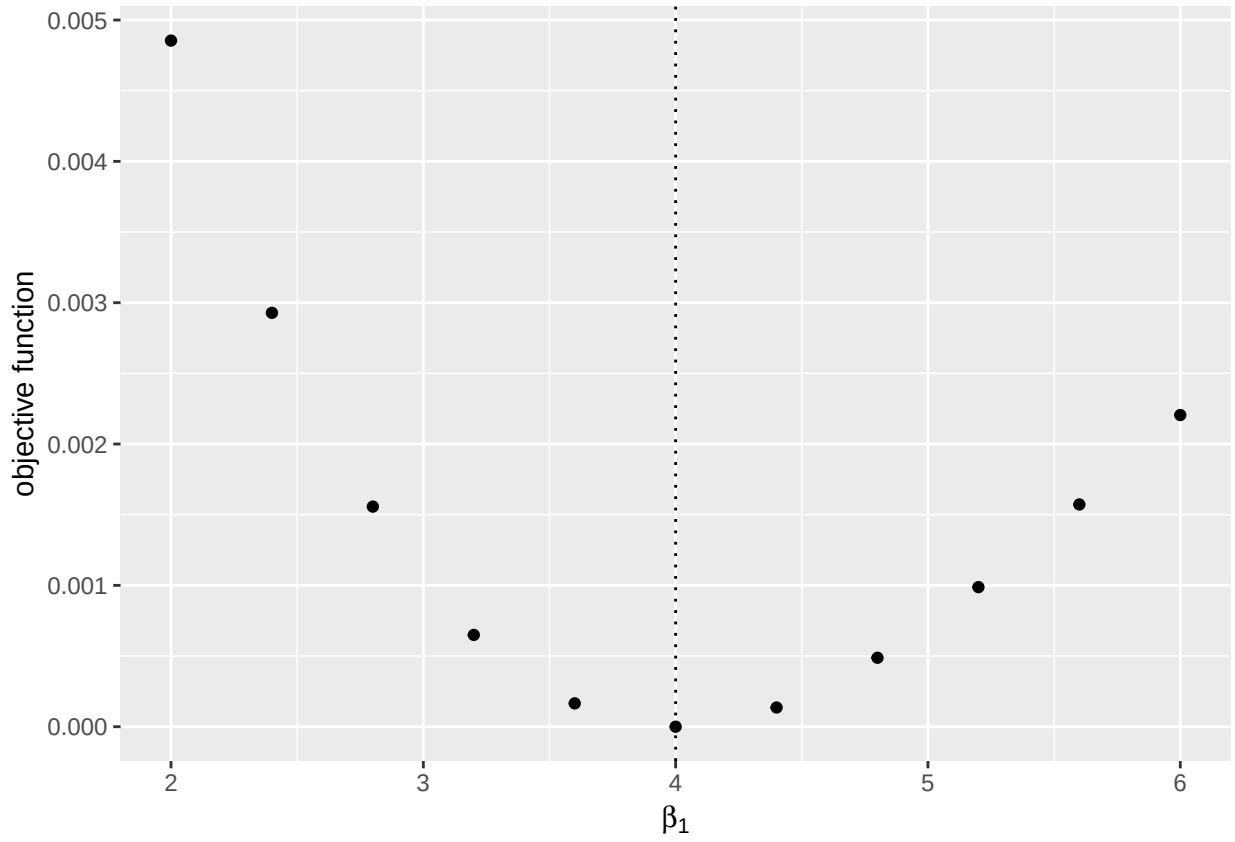
If you have compiled your functions in a package, you will just have to pass the package as follows:

```
# this function is compiled in the package EmpiricalIO
# temp_func <- function(x) {
#   y <- 2 * x
#   return(y)
# }
foreach (i = 1:4,
         .packages = c(
           "EmpiricalIO",
           "magrittr")) %dopar% {
  # this part is parallelized
  y <- temp_func(i)
  return(y)
}
```

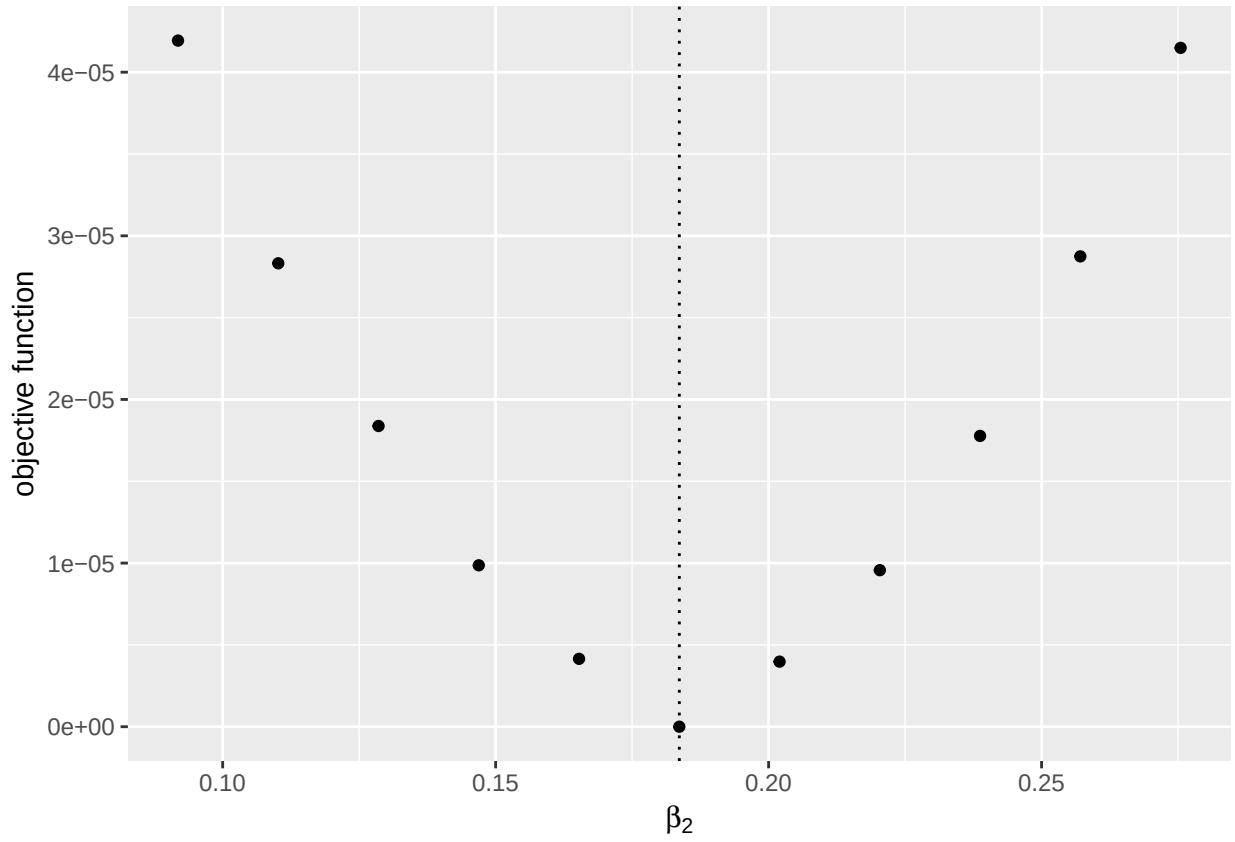
```
## [[1]]
## [1] 2
##
## [[2]]
## [1] 4
##
## [[3]]
## [1] 6
##
## [[4]]
## [1] 8
```

The graphs with the true shocks:

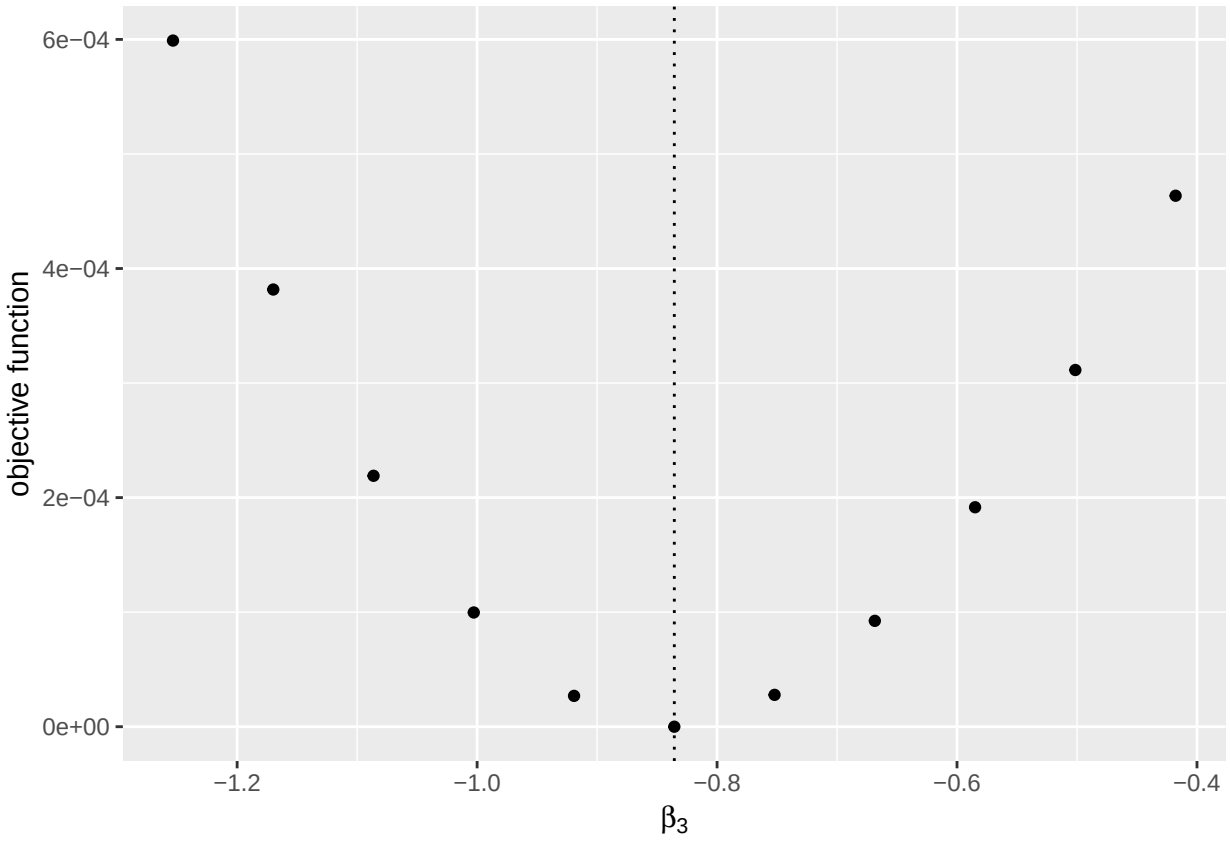
```
## [[1]]
```



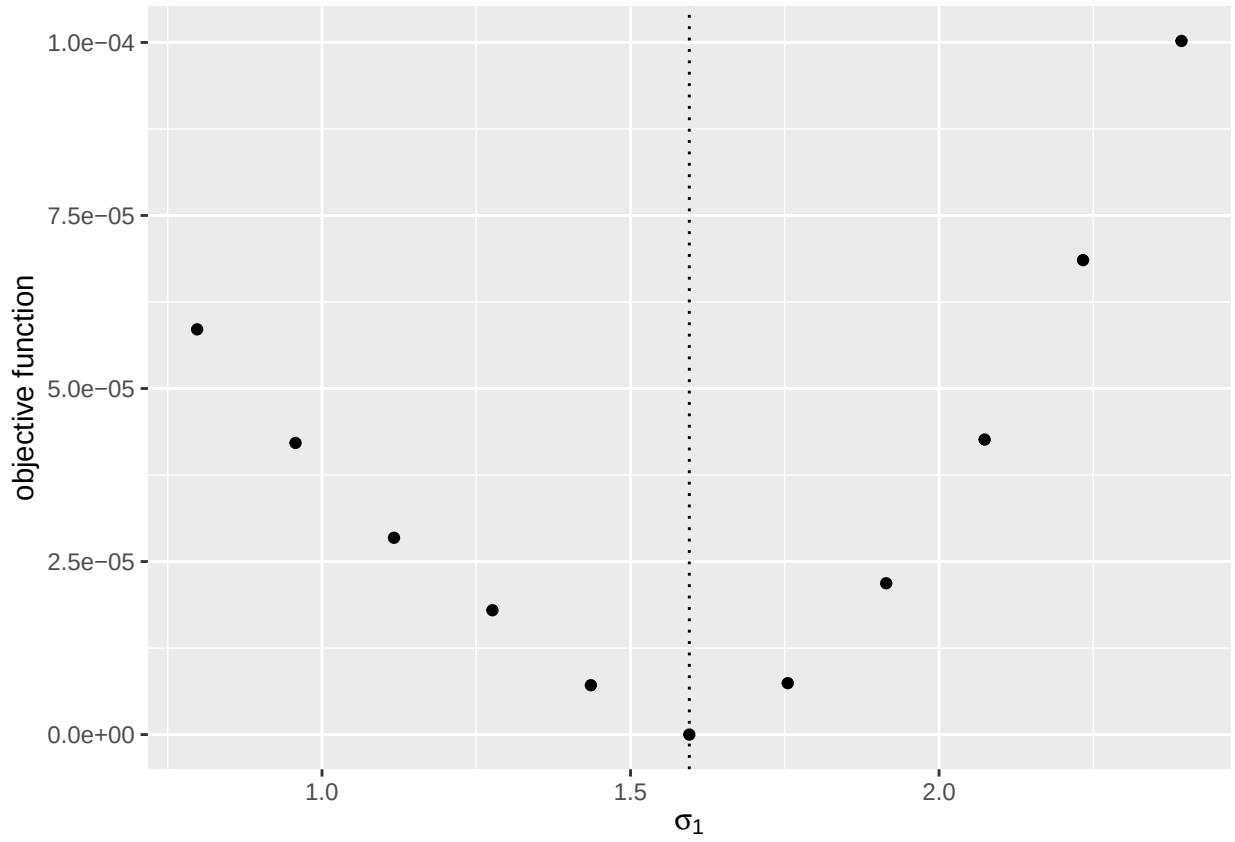

```
##
## [[2]]
```


4e−05
3e−05
2e−05
1e−05
0e+00
objective function
0.10
0.15
0.20
0.25
β2

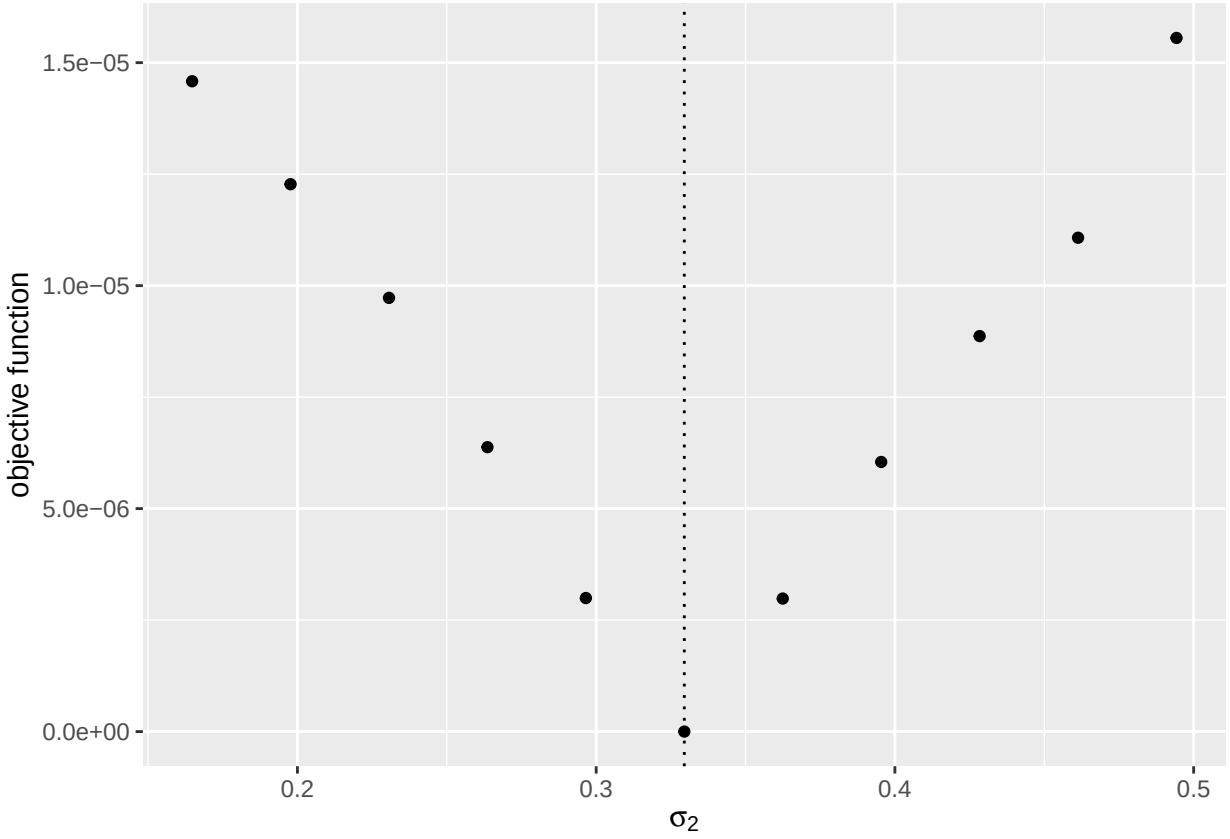
```
##
## [[3]]
```



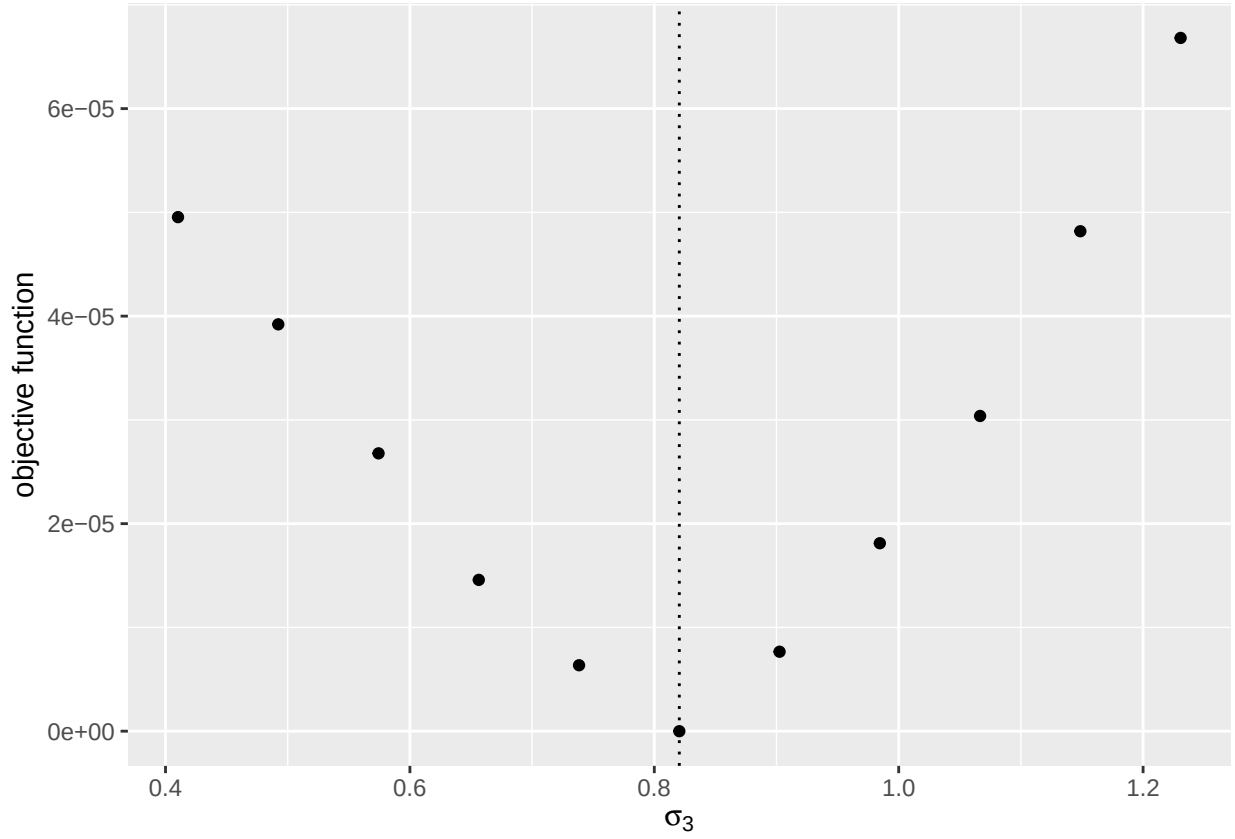

```
##
## [[4]]
```


1.0e−04
7.5e−05
5.0e−05
2.5e−05
0.0e+00
objective function
1.0
1.5
2.0
σ1

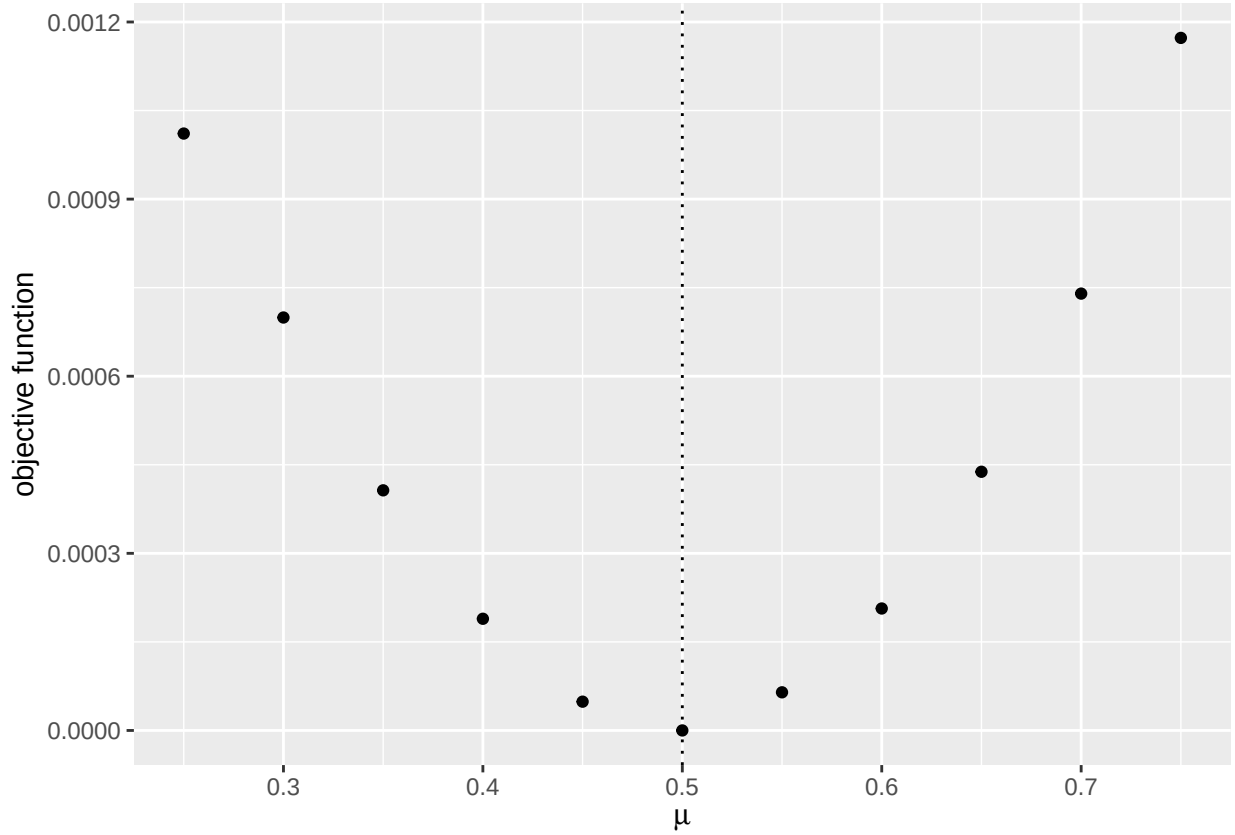
```
##
## [[5]]
```



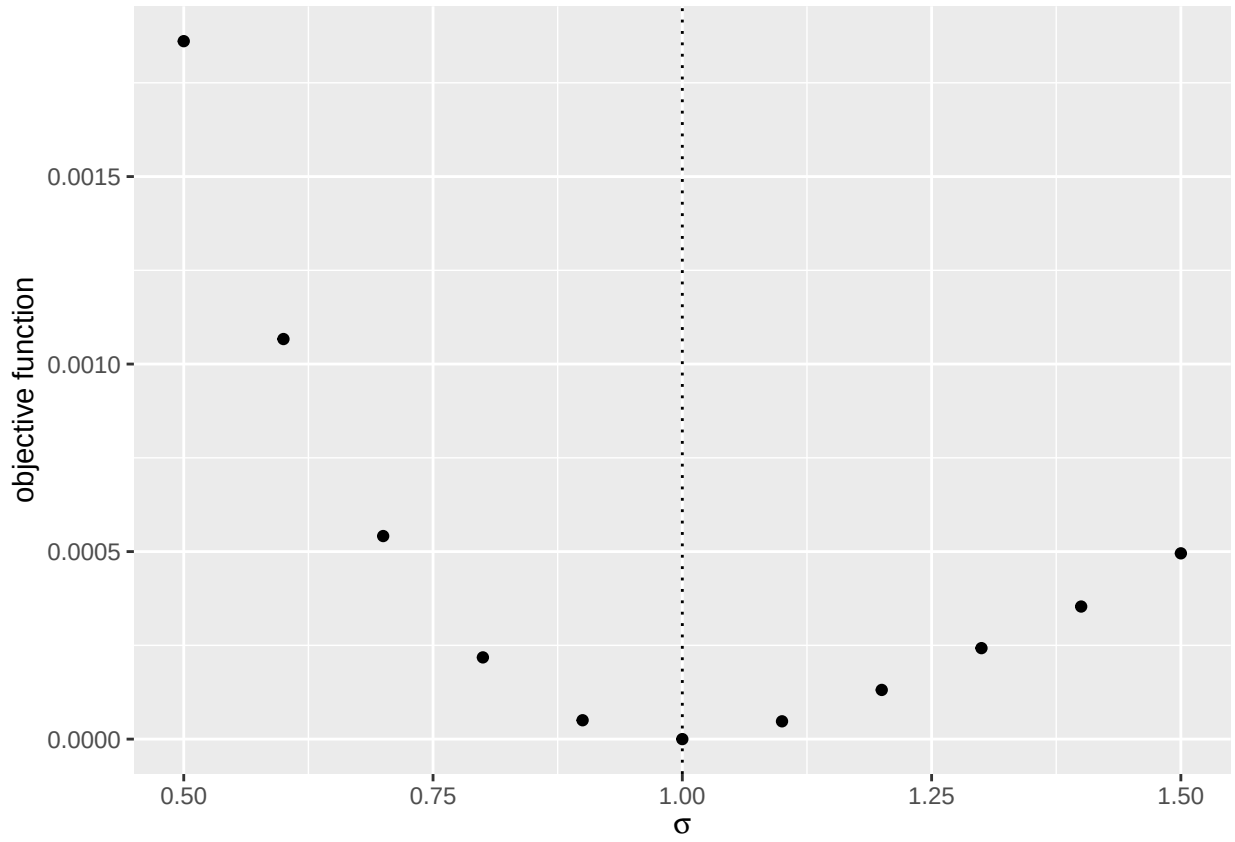

```
##
## [[6]]
```




```
##
## [[7]]
```

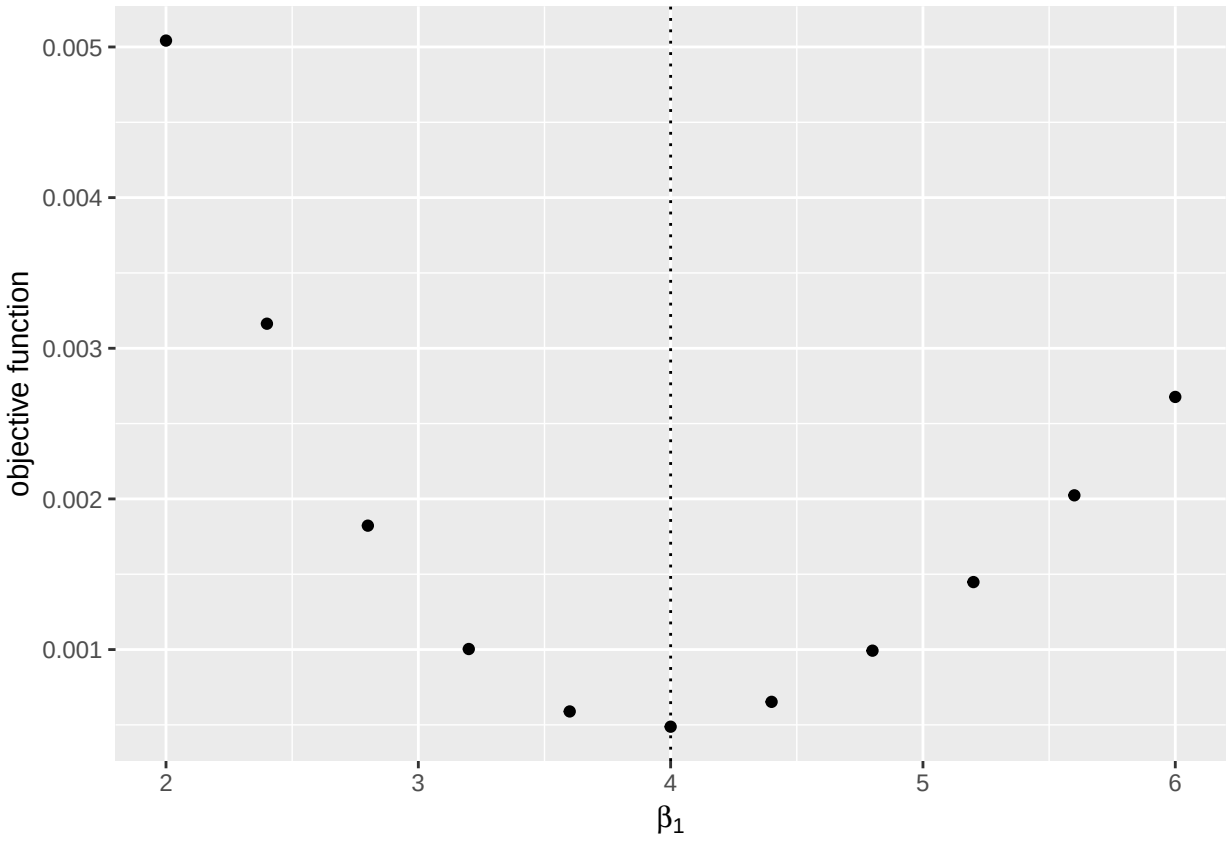

0.0012
0.0009
0.0006
0.0003
0.0000
objective function
0.3
0.4
0.5
0.6
0.7
μ

```
##
## [[8]]
```

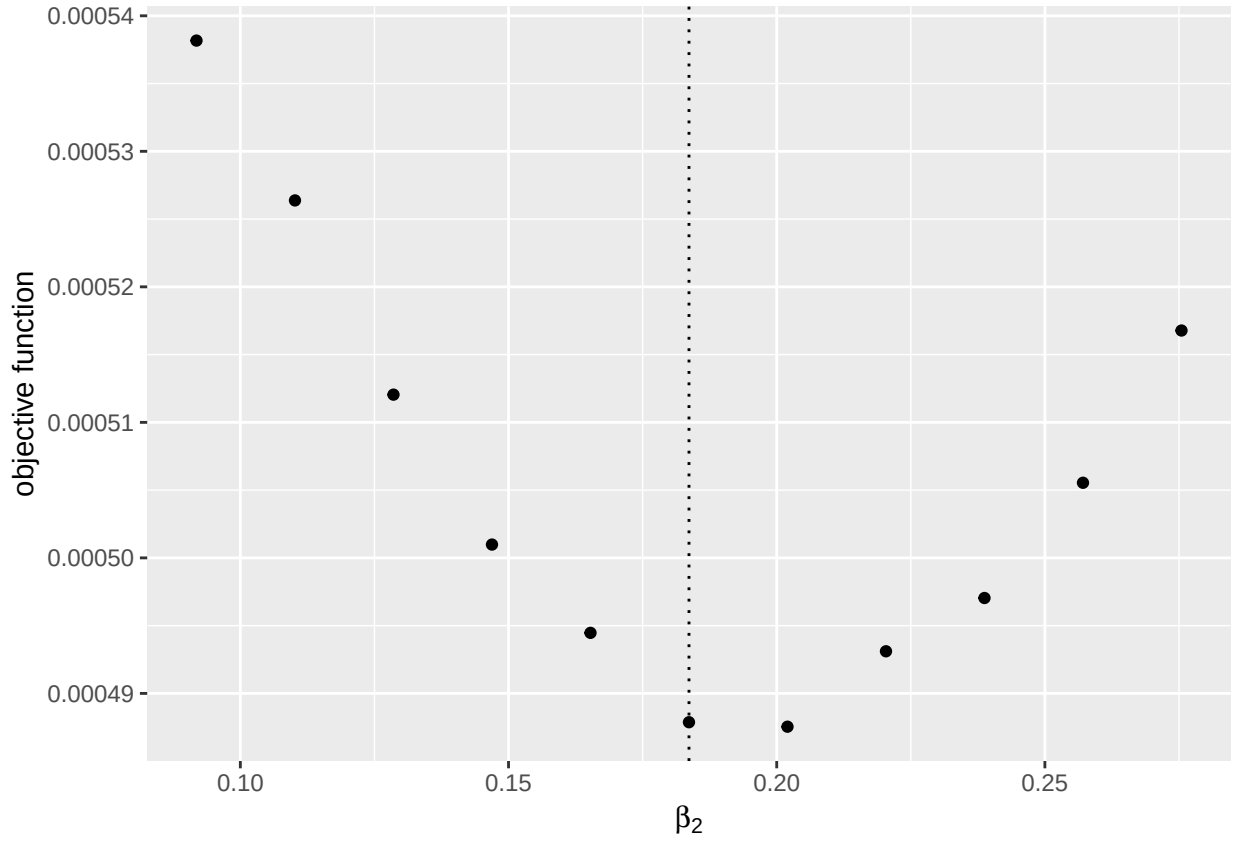


The graphs with the Monte Carlo shocks:

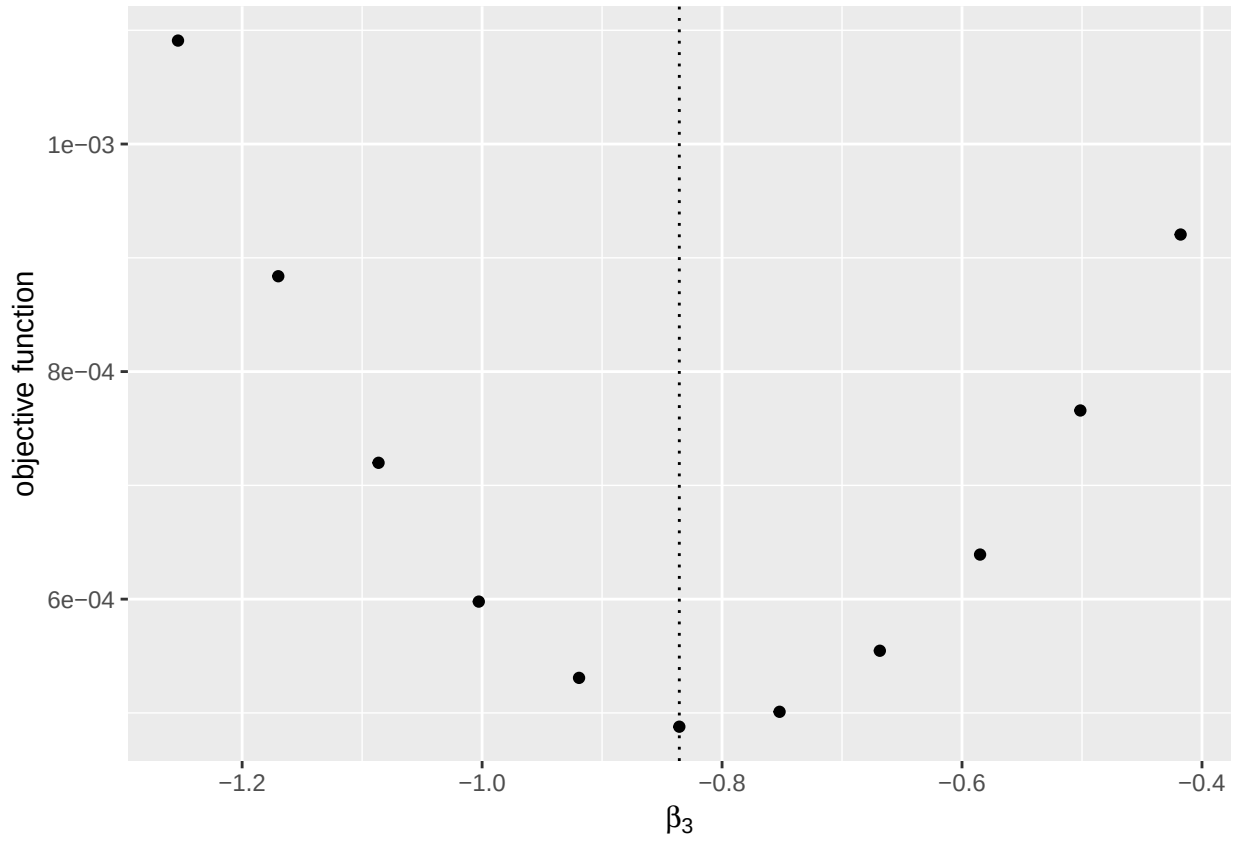
```
## [[1]]
```



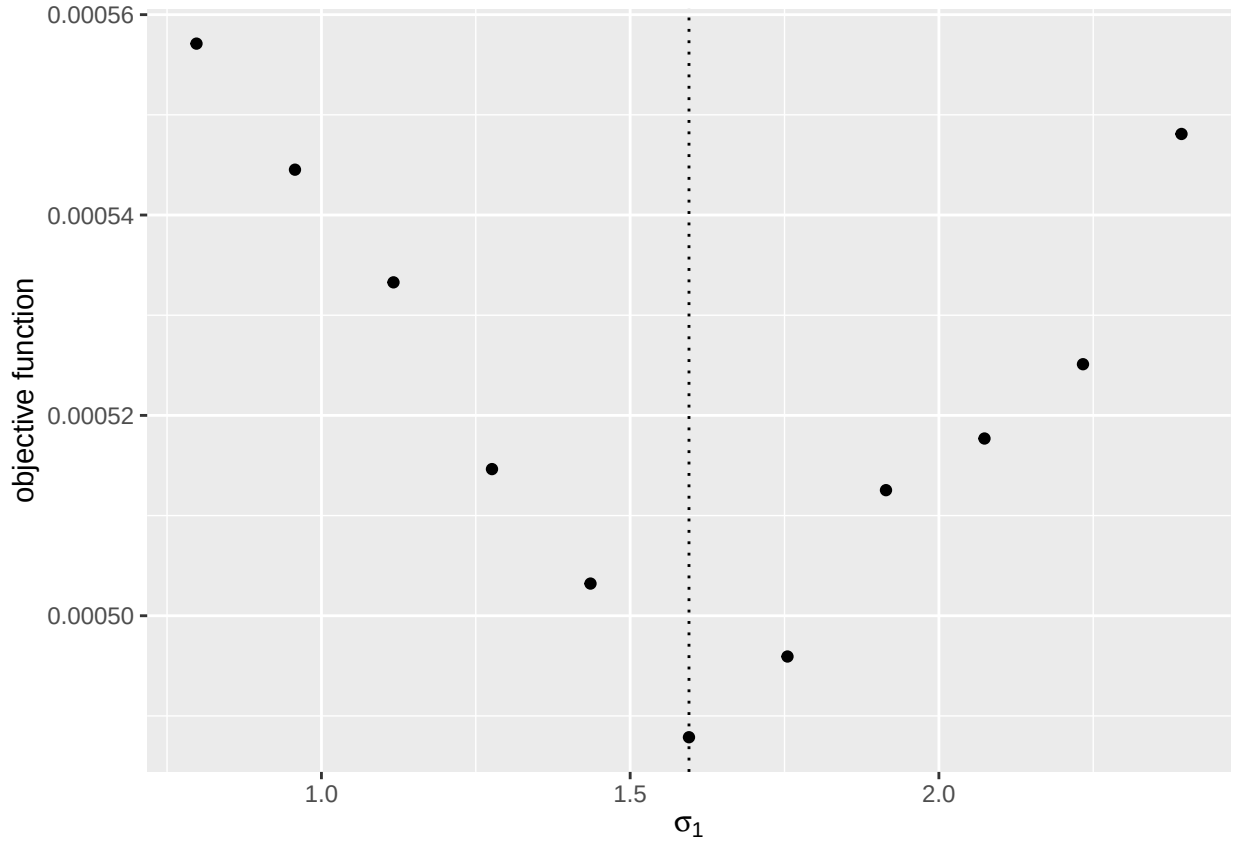

```
##
## [[2]]
```


0.00054
0.00053
0.00052
0.00051
0.00050
0.00049
objective function
0.10
0.15
0.20
0.25
β2

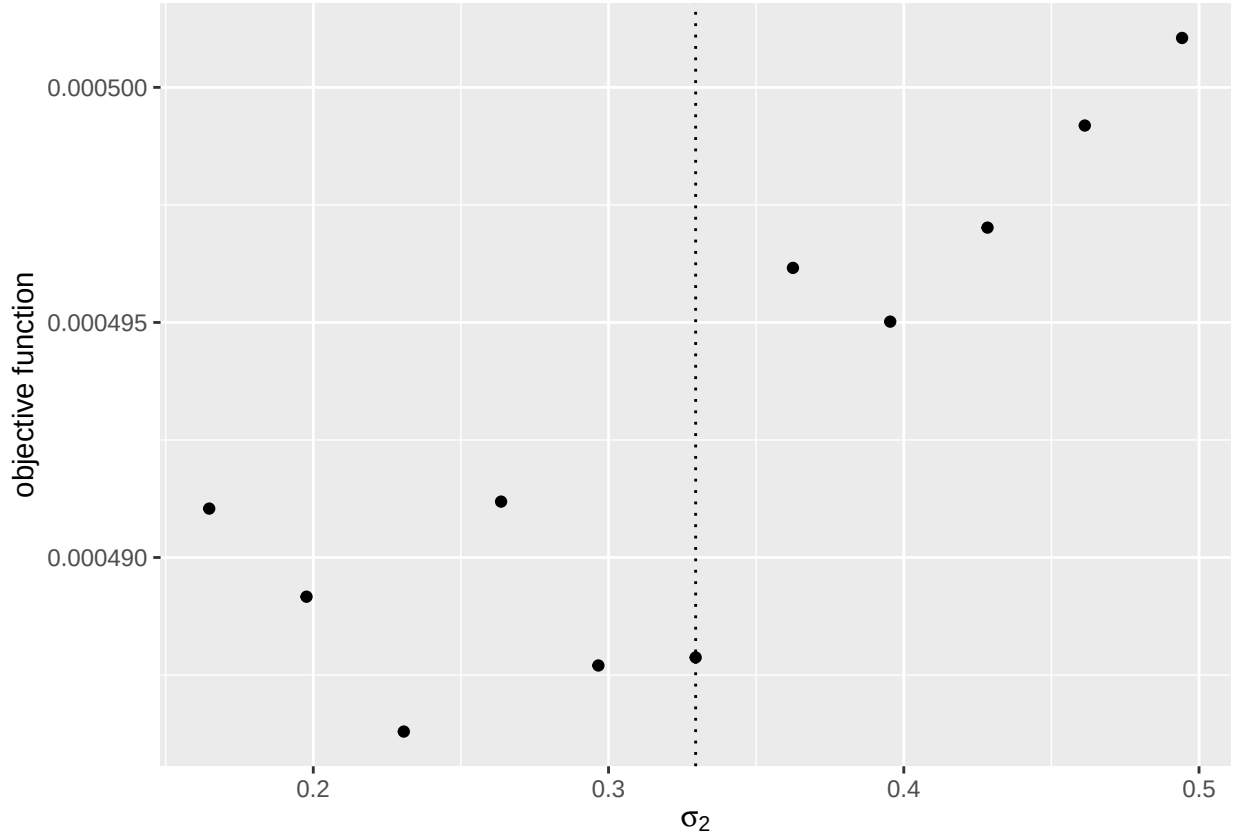
```
##
## [[3]]
```



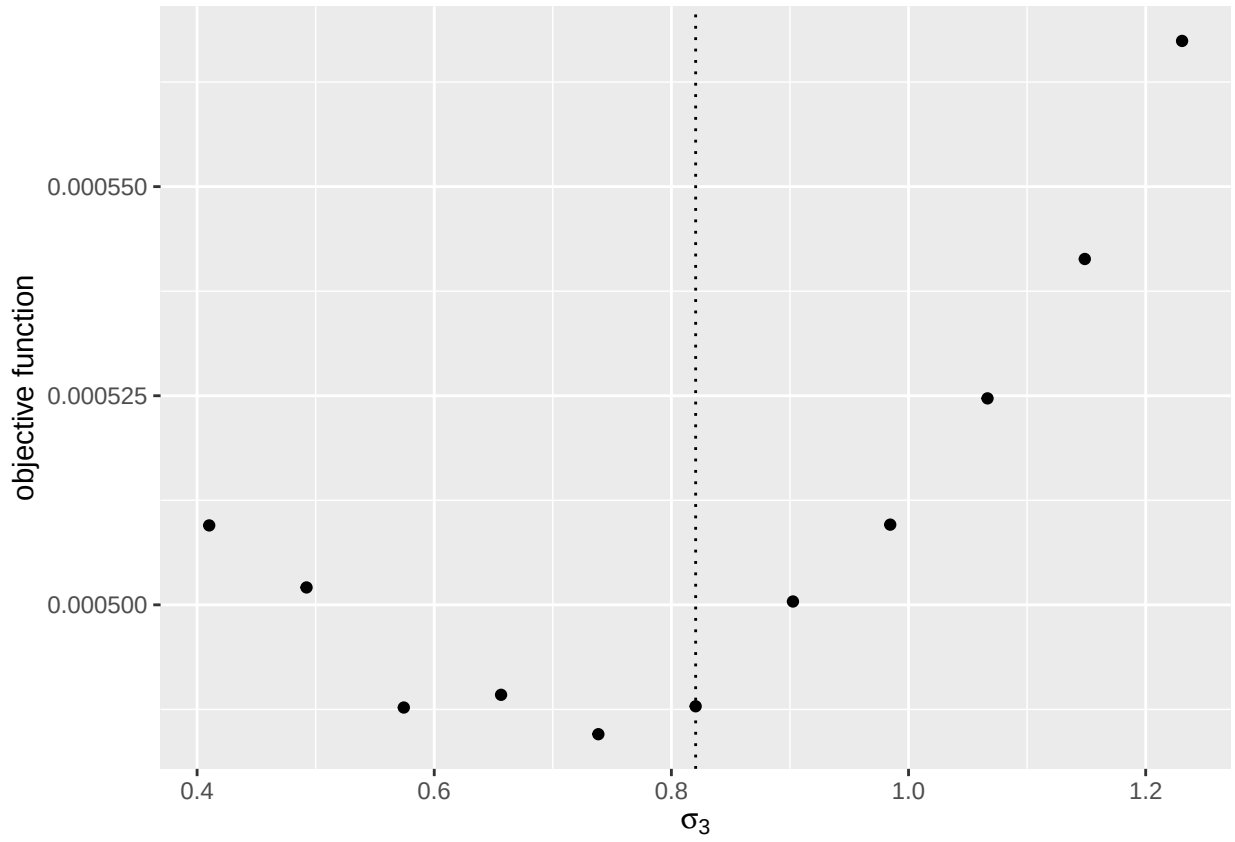

```
##
## [[4]]
```



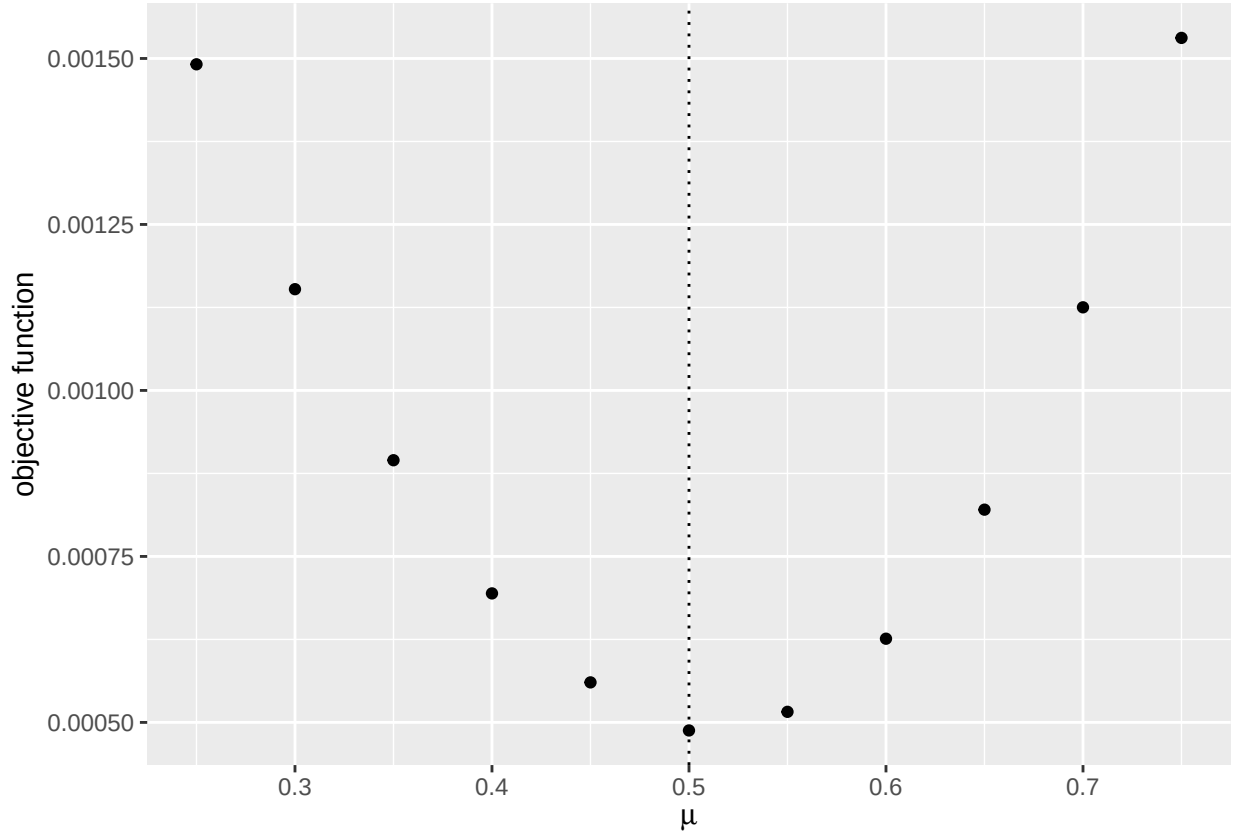

```
##
## [[5]]
```



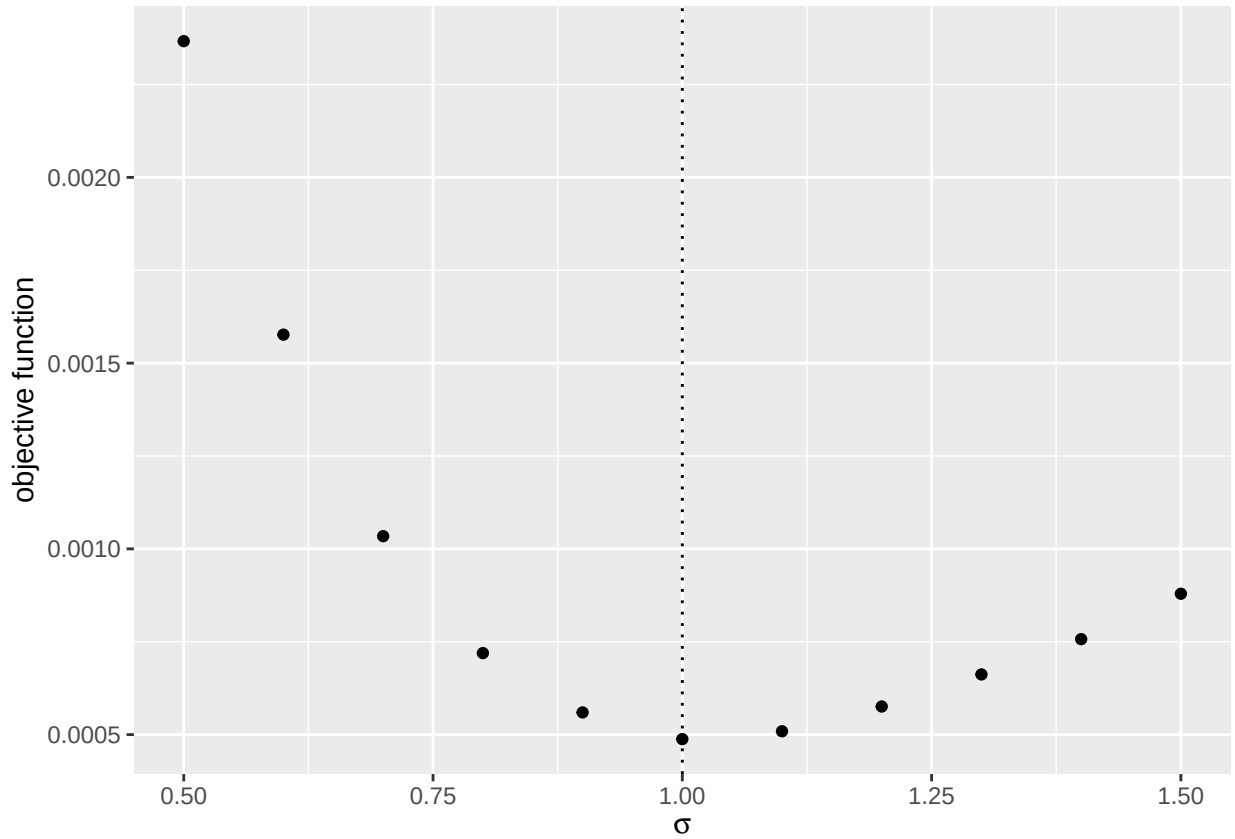

```
##
## [[6]]
```


objective function
0.000550
0.000525
0.000500
0.4
0.6
0.8
1.0
1.2
σ3

```
##
## [[7]]
```


0.00150
0.00125
0.00100
0.00075
0.00050
objective function
0.3
0.4
0.5
0.6
0.7
μ

```
##
## [[8]]
```



8. Use `optim` to find the minimizer of the objective function using `Nelder-Mead` method. You can start from the true parameter values. Compare the estimates with the true parameters.

```
## $par
## [1]  4.0425713  0.1841677 -0.8230489  1.6348574  0.3148886  0.7831262
## [7]  0.4998710  1.0138327
##
## $value
## [1] 0.0004760686
##
## $counts
## function gradient
##      263       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
##         true  estimates
## 1  4.0000000  4.0425713
## 2  0.1836433  0.1841677
## 3 -0.8356286 -0.8230489
## 4  1.5952808  1.6348574
## 5  0.3295078  0.3148886
## 6  0.8204684  0.7831262
## 7  0.5000000  0.4998710
```

```
## 8  1.0000000  1.0138327
```

Chapter 13

Assignment 4: Demand Function Estimation II

The deadline is **March 25 1:30pm**.

13.1 Simulate data

Be carefull that some parameters are changed from assignment 3. We simulate data from a discrete choice model that is the same with in assignment 3 except for the existence of unobserved product-specific fixed effects. There are T markets and each market has N consumers. There are J products and the indirect utility of consumer i in market t for product j is:

$$u_{itj} = \beta_{it}' x_j + \alpha_{it} p_{jt} + \xi_{jt} + \epsilon_{ijt},$$

where ϵ_{ijt} is an i.i.d. type-I extreme random variable. x_j is K-dimensional observed characteristics of the product. p_{jt} is the retail price of the product in the market.

ξ_{jt} is product-market specific fixed effect. p_{jt} can be correlated with ξ_{jt} but x_{jt}s are independent of ξ_{jt}. $j = 0$ is an outside option whose indirect utility is:

$$u_{it0} = \epsilon_{i0t},$$

where ϵ_{i0t} is an i.i.d. type-I extreme random variable.

β_{it} and α_{it} are different across consumers, and they are distributed as:

$$\beta_{itk} = \beta_{0k} + \sigma_k \nu_{itk},$$

$$\alpha_{it} = -\exp(\mu + \omega v_{it}) = -\exp(\mu + \frac{\omega^2}{2}) + [-\exp(\mu + \omega v_{it}) + \exp(\mu + \frac{\omega^2}{2})] \equiv \alpha_0 + \tilde{\alpha}_{it},$$

where ν_{itk} for $k = 1, \cdots, K$ and v_{it} are i.i.d. standard normal random variables. α_0 is the mean of α_i and $\tilde{\alpha}_i$ is the deviation from the mean.

Given a choice set in the market, $\mathcal{J}_t \cup \{0\}$, a consumer chooses the alternative that maximizes her utility:

$$q_{ijt} = 1\{u_{ijt} = \max_{k \in \mathcal{J}_t \cup \{0\}} u_{ikt}\}.$$

The choice probability of product j for consumer i in market t is:

$$\sigma_{jt}(p_t, x_t, \xi_t) = \mathbb{P}\{u_{ijt} = \max_{k \in \mathcal{J}_t \cup \{0\}} u_{ikt}\}.$$

Suppose that we only observe the share data:

$$s_{jt} = \frac{1}{N}\sum_{i=1}^{N} q_{ijt},$$

along with the product-market characteristics x_{jt} and the retail prices p_{jt} for $j \in \mathcal{J}_t \cup \{0\}$ for $t = 1, \cdots, T$. We do not observe the choice data q_{ijt} nor shocks $\xi_{jt}, \nu_{it}, \upsilon_{it}, \epsilon_{ijt}$.

We draw ξ_{jt} from i.i.d. normal distribution with mean 0 and standard deviation σ_ξ.

1. Set the seed, constants, and parameters of interest as follows.

```
# set the seed
set.seed(1)
# number of products
J <- 10
# dimension of product characteristics including the intercept
K <- 3
# number of markets
T <- 100
# number of consumers per market
N <- 500
# number of Monte Carlo
L <- 500
```

```
# set parameters of interests
beta <- rnorm(K);
beta[1] <- 4
beta
```

```
## [1]  4.0000000  0.1836433 -0.8356286
```

```
sigma <- abs(rnorm(K)); sigma
```

```
## [1] 1.5952808 0.3295078 0.8204684
```

```
mu <- 0.5
omega <- 1
```

Generate the covariates as follows.

The product-market characteristics:

$$x_{j1} = 1, x_{jk} \sim N(0, \sigma_x), k = 2, \cdots, K,$$

where σ_x is referred to as `sd_x` in the code.

The product-market-specific unobserved fixed effect:

$$\xi_{jt} \sim N(0, \sigma_\xi),$$

where $\sigma_x i$ is referred to as `sd_xi` in the code.

The marginal cost of product j in market t:

$$c_{jt} \sim \text{logNormal}(0, \sigma_c),$$

where σ_c is referred to as `sd_c` in the code.

The retail price:

$$p_{jt} - c_{jt} \sim \text{logNorm}(\gamma\xi_{jt}, \sigma_p),$$

where γ is referred to as `price_xi` and σ_p as `sd_p` in the code. This price is not the equilibrium price. We will revisit this point in a subsequent assignment.

The value of the auxiliary parameters are set as follows:

```
# set auxiliary parameters
price_xi <- 1
sd_x <- 2
sd_xi <- 0.5
sd_c <- 0.05
sd_p <- 0.05
```

2. `X` is the data frame such that a row contains the characteristics vector x_j of a product and columns are product index and observed product characteristics. The dimension of the characteristics K is specified above. Add the row of the outside option whose index is 0 and all the characteristics are zero.

```
X
```

```
## # A tibble: 11 x 4
##        j   x_1     x_2     x_3
##    <dbl> <dbl>   <dbl>   <dbl>
##  1     0     0  0       0
##  2     1     1  0.975  -0.0324
##  3     2     1  1.48    1.89
##  4     3     1  1.15    1.64
##  5     4     1 -0.611   1.19
##  6     5     1  3.02    1.84
##  7     6     1  0.780   1.56
##  8     7     1 -1.24    0.149
##  9     8     1 -4.43   -3.98
## 10     9     1  2.25    1.24
## 11    10     1 -0.0899 -0.112
```

3. `M` is the data frame such that a row contains the price ξ_{jt}, marginal cost c_{jt}, and price p_{jt}. After generating the variables, drop some products in each market. **In this assignment, we drop products in a different way from the last assignment**. In order to change the number of available products in each market, for each market, first draw J_t from a discrete uniform distribution between 1 and J. Then, drop products from each market using `dplyr::sample_frac` function with the realized number of available products. The variation in the available products is important for the identification of the distribution of consumer-level unobserved heterogeneity. Add the row of the outside option to each market whose index is 0 and all the variables take value zero.

```
M
```

```
## # A tibble: 746 x 5
##        j     t      xi     c     p
##    <dbl> <int>   <dbl> <dbl> <dbl>
##  1     0     1  0      0     0
##  2     6     1 -0.0514 0.980  1.91
##  3     0     2  0      0     0
##  4     1     2 -0.197  0.988  1.90
##  5    10     2 -0.354  1.05   1.82
##  6     0     3  0      0     0
##  7     1     3  0.182  1.04   2.22
##  8     2     3  0.384  1.01   2.61
##  9     3     3 -0.0562 1.02   2.01
## 10     4     3  0.441  1.01   2.62
## # ... with 736 more rows
```

4. Generate the consumer-level heterogeneity. `V` is the data frame such that a row contains the vector of shocks to consumer-level heterogeneity, (ν_i', υ_i). They are all i.i.d. standard normal random variables.

```
V
```

```
## # A tibble: 50,000 x 6
##        i     t   v_x_1  v_x_2   v_x_3    v_p
##    <int> <int>   <dbl>  <dbl>   <dbl>  <dbl>
##  1     1     1  1.16   -1.40   0.0786 -1.15
##  2     2     1 -1.05   -0.149  1.08    0.623
##  3     3     1 -0.426  -4.21   0.625  -1.14
##  4     4     1 -0.235   0.463  0.470   0.241
##  5     5     1  1.19   -0.342  0.169   0.160
##  6     6     1  0.541   0.525  0.305   1.72
##  7     7     1 -0.0893 -0.434  2.18   -0.432
##  8     8     1 -0.712   0.747 -0.306  -0.527
##  9     9     1  0.504   1.11   1.70   -1.75
## 10    10     1 -0.107   1.83  -0.841   0.693
## # ... with 49,990 more rows
```

5. Join `X`, `M`, `V` using `dplyr::left_join` and name it `df`. `df` is the data frame such that a row contains variables for a consumer about a product that is available in a market.

```
df
```

```
## # A tibble: 373,000 x 13
##        t     i     j  v_x_1  v_x_2  v_x_3    v_p   x_1   x_2   x_3      xi
##    <int> <int> <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl>   <dbl>
##  1     1     1     0  1.16  -1.40  0.0786 -1.15      0 0      0     0
##  2     1     1     6  1.16  -1.40  0.0786 -1.15      1 0.780  1.56 -0.0514
##  3     1     2     0 -1.05  -0.149 1.08    0.623     0 0      0     0
##  4     1     2     6 -1.05  -0.149 1.08    0.623     1 0.780  1.56 -0.0514
##  5     1     3     0 -0.426 -4.21  0.625  -1.14      0 0      0     0
##  6     1     3     6 -0.426 -4.21  0.625  -1.14      1 0.780  1.56 -0.0514
##  7     1     4     0 -0.235  0.463 0.470   0.241     0 0      0     0
##  8     1     4     6 -0.235  0.463 0.470   0.241     1 0.780  1.56 -0.0514
##  9     1     5     0  1.19  -0.342 0.169   0.160     0 0      0     0
## 10     1     5     6  1.19  -0.342 0.169   0.160     1 0.780  1.56 -0.0514
## # ... with 372,990 more rows, and 2 more variables: c <dbl>, p <dbl>
```

6. Draw a vector of preference shocks `e` whose length is the same as the number of rows of `df`.

```
head(e)
```

```
## [1]  0.2262454  1.3417639 -0.1693913  0.8906905  0.5558130  1.9909058
```

7. Write a function `compute_indirect_utility(df, beta, sigma, mu, omega)` that returns a vector whose element is the mean indirect utility of a product for a consumer in a market. The output should have the same length with e. (This function is the same with assignment 3. You can use the function.)

```
# compute indirect utility
u <-
  compute_indirect_utility(
    df, beta, sigma,
           mu, omega)
head(u)
```

```
##               u
## [1,]  0.0000000
```

```
## [2,]  3.3750542
## [3,]  0.0000000
## [4,] -3.3983588
## [5,]  0.0000000
## [6,]  0.8235142
```

In the previous assingment, we computed predicted share by simulating choice and taking their average. Instead, we compute the actual share by:

$$s_{jt} = \frac{1}{N}\sum_{i=1}^{N} \frac{\exp[\beta_{it}' x_j + \alpha_{it} p_{jt} + \xi_{jt}]}{1 + \sum_{k \in \mathcal{J}_t} \exp[\beta_{it}' x_k + \alpha_{it} p_{kt} + \xi_{jt}]}$$

and the predicted share by:

$$\widehat{\sigma}_j(x, p_t, \xi_t) = \frac{1}{L}\sum_{l=1}^{L} \frac{\exp[\beta_t^{(l)\prime} x_j + \alpha_t^{(l)} p_{jt} + \xi_{jt}]}{1 + \sum_{k \in \mathcal{J}_t} \exp[\beta_t^{(l)\prime} x_k + \alpha_t^{(l)} p_{kt} + \xi_{jt}]}.$$

8. To do so, write a function `compute_choice_smooth(X, M, V, beta, sigma, mu, omega)` in which the choice of each consumer is not:

$$q_{ijt} = 1\{u_{ijt} = \max_{k \in \mathcal{J}_t \cup \{0\}} u_{ikt}\},$$

but

$$\tilde{q}_{ijt} = \frac{\exp(u_{ijt})}{1 + \sum_{k \in \mathcal{J}_t} \exp(u_{ikt})}.$$

```
df_choice_smooth <-
  compute_choice_smooth(X, M, V, beta, sigma, mu, omega)
summary(df_choice_smooth)
```

```
##        t                i                j              v_x_1
##  Min.   :  1.00   Min.   :  1.0   Min.   : 0.000   Min.   :-4.302781
##  1st Qu.: 26.00   1st Qu.:125.8   1st Qu.: 2.000   1st Qu.:-0.685447
##  Median : 51.50   Median :250.5   Median : 5.000   Median :-0.000461
##  Mean   : 51.26   Mean   :250.5   Mean   : 4.807   Mean   :-0.005284
##  3rd Qu.: 77.00   3rd Qu.:375.2   3rd Qu.: 8.000   3rd Qu.: 0.665219
##  Max.   :100.00   Max.   :500.0   Max.   :10.000   Max.   : 3.809895
##      v_x_2              v_x_3               v_p
##  Min.   :-4.542122   Min.   :-3.957618   Min.   :-4.218131
##  1st Qu.:-0.678377   1st Qu.:-0.676638   1st Qu.:-0.672011
##  Median : 0.001435   Median : 0.006281   Median : 0.002166
##  Mean   :-0.001076   Mean   : 0.003433   Mean   :-0.001402
##  3rd Qu.: 0.671827   3rd Qu.: 0.679273   3rd Qu.: 0.674681
##  Max.   : 4.313621   Max.   : 4.244194   Max.   : 4.074300
##       x_1              x_2               x_3                 xi
##  Min.   :0.000   Min.   :-4.4294   Min.   :-3.97870   Min.   :-1.444460
##  1st Qu.:1.000   1st Qu.:-0.6108   1st Qu.:-0.03238   1st Qu.:-0.286633
##  Median :1.000   Median : 0.7797   Median : 1.18780   Median : 0.000000
##  Mean   :0.866   Mean   : 0.2713   Mean   : 0.44050   Mean   : 0.009578
##  3rd Qu.:1.000   3rd Qu.: 1.4766   3rd Qu.: 1.64244   3rd Qu.: 0.317352
##  Max.   :1.000   Max.   : 3.0236   Max.   : 1.88767   Max.   : 1.905138
##        c                p               u                   q
##  Min.   :0.0000   Min.   :0.000   Min.   :-284.515   Min.   :0.0000000
##  1st Qu.:0.9425   1st Qu.:1.562   1st Qu.:  -3.157   1st Qu.:0.0009063
##  Median :0.9886   Median :1.902   Median :   0.000   Median :0.0225375
##  Mean   :0.8670   Mean   :1.871   Mean   :  -1.927   Mean   :0.1340483
```

```
##  3rd Qu.:1.0322   3rd Qu.:2.394   3rd Qu.:   1.623   3rd Qu.:0.1203147
##  Max.   :1.1996   Max.   :8.211   Max.   :  19.907   Max.   :1.0000000
```

9. Next, write a function `compute_share_smooth(X, M, V, beta, sigma, mu, omega)` that calls `compute_choice_smooth` and then returns the share based on above $\tilde{q}_{ijt}$. If we use these functions with the Monte Carlo shocks, it gives us the predicted share of the products.

```
df_share_smooth <- compute_share_smooth(X, M, V, beta, sigma, mu, omega)
summary(df_share_smooth)
```

```
##        t                j               x_1              x_2
##  Min.   :  1.00   Min.   : 0.000   Min.   :0.000   Min.   :-4.4294
##  1st Qu.: 26.00   1st Qu.: 2.000   1st Qu.:1.000   1st Qu.:-0.6108
##  Median : 51.50   Median : 5.000   Median :1.000   Median : 0.7797
##  Mean   : 51.26   Mean   : 4.807   Mean   :0.866   Mean   : 0.2713
##  3rd Qu.: 77.00   3rd Qu.: 8.000   3rd Qu.:1.000   3rd Qu.: 1.4766
##  Max.   :100.00   Max.   :10.000   Max.   :1.000   Max.   : 3.0236
##       x_3                 xi                  c                p
##  Min.   :-3.97870   Min.   :-1.444460   Min.   :0.0000   Min.   :0.000
##  1st Qu.:-0.03238   1st Qu.:-0.286542   1st Qu.:0.9427   1st Qu.:1.564
##  Median : 1.18780   Median : 0.000000   Median :0.9886   Median :1.902
##  Mean   : 0.44050   Mean   : 0.009578   Mean   :0.8670   Mean   :1.871
##  3rd Qu.: 1.62290   3rd Qu.: 0.316516   3rd Qu.:1.0322   3rd Qu.:2.392
##  Max.   : 1.88767   Max.   : 1.905138   Max.   :1.1996   Max.   :8.211
##        q                 s                  y
##  Min.   :  5.912   Min.   :0.01182   Min.   :-2.9837
##  1st Qu.: 17.628   1st Qu.:0.03526   1st Qu.:-1.9109
##  Median : 28.025   Median :0.05605   Median :-1.5011
##  Mean   : 67.024   Mean   :0.13405   Mean   :-1.2219
##  3rd Qu.:100.446   3rd Qu.:0.20089   3rd Qu.:-0.5556
##  Max.   :337.118   Max.   :0.67424   Max.   : 1.1167
```

Use this `df_share_smooth` as the data to estimate the parameters in the following section.

13.2 Estimate the parameters

1. First draw Monte Carlo consumer-level heterogeneity `V_mcmc` and Monte Carlo preference shocks `e_mcmc`. The number of simulations is `L`. This does not have to be the same with the actual number of consumers `N`.

```
V_mcmc
```

```
## # A tibble: 50,000 x 6
##        i     t  v_x_1   v_x_2  v_x_3     v_p
##    <int> <int>  <dbl>   <dbl>  <dbl>   <dbl>
##  1     1     1  0.865 -0.142  -0.667  1.17
##  2     2     1  0.316  1.29   -1.56  -0.691
##  3     3     1  0.673  1.35   -0.203  0.388
##  4     4     1  0.295  0.613   1.31   0.698
##  5     5     1  0.214 -0.0878 -0.343 -0.0642
##  6     6     1 -1.06  -0.240   0.373 -0.631
##  7     7     1 -0.556  1.05   -1.21  -2.22
##  8     8     1  0.376 -2.47    1.77  -0.333
##  9     9     1 -0.872  1.36    0.508 -0.834
## 10    10     1 -0.895 -1.26   -3.04   0.821
```

```
## # ... with 49,990 more rows
```

```
head(e_mcmc)
```

```
## [1] 1.4664583 0.9890441 1.2502808 0.7103677 0.7433964 1.9116964
```

2. Vectorize the parameters to a vector `theta` because `optim` requires the maximiand to be a vector.

```
# set parameters
theta <- c(beta, sigma, mu, omega)
theta
```

```
## [1]  4.0000000  0.1836433 -0.8356286  1.5952808  0.3295078  0.8204684
## [7]  0.5000000  1.0000000
```

3. Estimate the parameters assuming there is no product-specific unobserved fixed effects ξ_{jt}, i.e., using the functions in assignment 3. To do so, first modify `M` to `M_no` in which `xi` is replaced with 0 and estimate the model with `M_no`. Otherwise, your function will compute the share with the true `xi`.

```
M_no <- M %>%
  dplyr::mutate(xi = 0)
```

```
## $par
## [1]  3.1292612  0.1834495 -0.9357865  1.5678279  0.3768438  1.1698887
## [7] -0.1108147  1.9156776
##
## $value
## [1] 0.0004291768
##
## $counts
## function gradient
##      297       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
##         true  estimates
## 1  4.0000000  3.1292612
## 2  0.1836433  0.1834495
## 3 -0.8356286 -0.9357865
## 4  1.5952808  1.5678279
## 5  0.3295078  0.3768438
## 6  0.8204684  1.1698887
## 7  0.5000000 -0.1108147
## 8  1.0000000  1.9156776
```

Next, we estimate the model allowing for the product-market-specific unobserved fixed effect ξ_{jt} using the BLP algorithm. To do so, we slightly modify the `compute_indirect_utility`, `compute_choice_smooth`, and `compute_share_smooth` functions so that they receive δ_{jt} to compute the indirect utilities, choices, and shares. Be careful that the treatment of α_i is slightly different from the lecture note, because we assumed that α_is are log-normal random variables.

4. Compute and print out δ_{jt} at the true parameters, i.e.:

$$\delta_{jt} = \beta_0' x_j + \alpha_0' p_{jt} + \xi_{jt}.$$

```
delta
```

```
## # A tibble: 746 x 3
##        t     j delta
##    <int> <dbl> <dbl>
##  1     1     0  0
##  2     1     6 -2.40
##  3     2     0  0
##  4     2     1 -1.15
##  5     2    10 -1.22
##  6     3     0  0
##  7     3     1 -1.65
##  8     3     2 -4.03
##  9     3     3 -2.69
## 10     3     4 -3.78
## # ... with 736 more rows
```

5. Write a function `compute_indirect_utility_delta(df, delta, sigma, mu, omega)` that returns a vector whose element is the mean indirect utility of a product for a consumer in a market. The output should have the same length with e. Print out the output with δ_{jt} evaluated at the true parameters. Check if the output is close to the true indirect utilities.

```
# compute indirect utility from delta
u_delta <-
  compute_indirect_utility_delta(df, delta, sigma,
                                 mu, omega)
head(u_delta)
```

```
##               u
## [1,]  0.0000000
## [2,]  3.3750542
## [3,]  0.0000000
## [4,] -3.3983588
## [5,]  0.0000000
## [6,]  0.8235142
```

```
summary(u - u_delta)
```

```
##        u
##  Min.   :-5.684e-14
##  1st Qu.:-4.441e-16
##  Median : 0.000e+00
##  Mean   :-1.279e-17
##  3rd Qu.: 3.331e-16
##  Max.   : 5.684e-14
```

6. Write a function `compute_choice_smooth_delta(X, M, V, delta, sigma, mu, omega)` that first construct `df` from `X`, `M`, `V`, second call `compute_indirect_utility_delta` to obtain the vector of mean indirect utilities `u`, third compute the (smooth) choice vector `q` based on the vector of mean indirect utilities, and finally return the data frame to which `u` and `q` are added as columns. Print out the output with δ_{jt} evaluated at the true parameters. Check if the output is close to the true (smooth) choice vector.

```
# compute choice
df_choice_smooth_delta <-
  compute_choice_smooth_delta(X, M, V, delta, sigma, mu, omega)
df_choice_smooth_delta
```

```
## # A tibble: 373,000 x 15
##        t     i     j  v_x_1  v_x_2  v_x_3    v_p   x_1   x_2   x_3      xi
##    <int> <int> <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl>   <dbl>
##  1     1     1     0  1.16  -1.40  0.0786 -1.15      0 0      0     0
##  2     1     1     6  1.16  -1.40  0.0786 -1.15      1 0.780  1.56 -0.0514
##  3     1     2     0 -1.05  -0.149 1.08    0.623     0 0      0     0
##  4     1     2     6 -1.05  -0.149 1.08    0.623     1 0.780  1.56 -0.0514
##  5     1     3     0 -0.426 -4.21  0.625  -1.14      0 0      0     0
##  6     1     3     6 -0.426 -4.21  0.625  -1.14      1 0.780  1.56 -0.0514
##  7     1     4     0 -0.235  0.463 0.470   0.241     0 0      0     0
##  8     1     4     6 -0.235  0.463 0.470   0.241     1 0.780  1.56 -0.0514
##  9     1     5     0  1.19  -0.342 0.169   0.160     0 0      0     0
## 10     1     5     6  1.19  -0.342 0.169   0.160     1 0.780  1.56 -0.0514
## # ... with 372,990 more rows, and 4 more variables: c <dbl>, p <dbl>,
## #   u <dbl>, q <dbl>
```

```
summary(df_choice_smooth_delta)
```

```
##        t                i               j              v_x_1
##  Min.   :  1.00   Min.   :  1.0   Min.   : 0.000   Min.   :-4.302781
##  1st Qu.: 26.00   1st Qu.:125.8   1st Qu.: 2.000   1st Qu.:-0.685447
##  Median : 51.50   Median :250.5   Median : 5.000   Median :-0.000461
##  Mean   : 51.26   Mean   :250.5   Mean   : 4.807   Mean   :-0.005284
##  3rd Qu.: 77.00   3rd Qu.:375.2   3rd Qu.: 8.000   3rd Qu.: 0.665219
##  Max.   :100.00   Max.   :500.0   Max.   :10.000   Max.   : 3.809895
##      v_x_2               v_x_3               v_p
##  Min.   :-4.542122   Min.   :-3.957618   Min.   :-4.218131
##  1st Qu.:-0.678377   1st Qu.:-0.676638   1st Qu.:-0.672011
##  Median : 0.001435   Median : 0.006281   Median : 0.002166
##  Mean   :-0.001076   Mean   : 0.003433   Mean   :-0.001402
##  3rd Qu.: 0.671827   3rd Qu.: 0.679273   3rd Qu.: 0.674681
##  Max.   : 4.313621   Max.   : 4.244194   Max.   : 4.074300
##       x_1              x_2               x_3                 xi
##  Min.   :0.000   Min.   :-4.4294   Min.   :-3.97870   Min.   :-1.444460
##  1st Qu.:1.000   1st Qu.:-0.6108   1st Qu.:-0.03238   1st Qu.:-0.286633
##  Median :1.000   Median : 0.7797   Median : 1.18780   Median : 0.000000
##  Mean   :0.866   Mean   : 0.2713   Mean   : 0.44050   Mean   : 0.009578
##  3rd Qu.:1.000   3rd Qu.: 1.4766   3rd Qu.: 1.64244   3rd Qu.: 0.317352
##  Max.   :1.000   Max.   : 3.0236   Max.   : 1.88767   Max.   : 1.905138
##        c                p               u                  q
##  Min.   :0.0000   Min.   :0.000   Min.   :-284.515   Min.   :0.0000000
##  1st Qu.:0.9425   1st Qu.:1.562   1st Qu.:  -3.157   1st Qu.:0.0009063
##  Median :0.9886   Median :1.902   Median :   0.000   Median :0.0225375
##  Mean   :0.8670   Mean   :1.871   Mean   :  -1.927   Mean   :0.1340483
##  3rd Qu.:1.0322   3rd Qu.:2.394   3rd Qu.:   1.623   3rd Qu.:0.1203147
##  Max.   :1.1996   Max.   :8.211   Max.   :  19.907   Max.   :1.0000000
```

```
summary(df_choice_smooth$q - df_choice_smooth_delta$q)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -1.166e-15 -6.939e-18  0.000e+00  4.540e-20  1.084e-18  1.221e-15
```

7. Write a function `compute_share_delta(X, M, V, delta, sigma, mu, omega)` that first construct `df` from `X`, `M`, `V`, second call `compute_choice_delta` to obtain a data frame with `u` and `q`, third compute the share of each product at each market `s` and the log difference in the share from the outside option, $\ln(s_{jt}/s_{0t})$, denoted by `y`, and finally return the data frame that is summarized at the product-market

level, dropped consumer-level variables, and added `s` and `y`.

```
# compute share
df_share_smooth_delta <-
  compute_share_smooth_delta(X, M, V, delta, sigma, mu, omega)
df_share_smooth_delta
```

```
## # A tibble: 746 x 11
##        t     j   x_1     x_2     x_3      xi     c     p      q      s
##    <int> <dbl> <dbl>   <dbl>   <dbl>   <dbl> <dbl> <dbl>  <dbl>  <dbl>
##  1     1     0     0  0       0       0      0      0    307.   0.614
##  2     1     6     1  0.780   1.56   -0.0514 0.980  1.91 193.   0.386
##  3     2     0     0  0       0       0      0      0    208.   0.415
##  4     2     1     1  0.975  -0.0324 -0.197  0.988  1.90 158.   0.317
##  5     2    10     1 -0.0899 -0.112  -0.354  1.05   1.82 134.   0.268
##  6     3     0     0  0       0       0      0      0    113.   0.225
##  7     3     1     1  0.975  -0.0324  0.182  1.04   2.22  25.6  0.0513
##  8     3     2     1  1.48    1.89    0.384  1.01   2.61  14.4  0.0288
##  9     3     3     1  1.15    1.64   -0.0562 1.02   2.01  18.3  0.0366
## 10     3     4     1 -0.611   1.19    0.441  1.01   2.62   9.46 0.0189
## # ... with 736 more rows, and 1 more variable: y <dbl>
```

```
summary(df_share_smooth_delta)
```

```
##        t                j               x_1              x_2         
##  Min.   :  1.00   Min.   : 0.000   Min.   :0.000   Min.   :-4.4294  
##  1st Qu.: 26.00   1st Qu.: 2.000   1st Qu.:1.000   1st Qu.:-0.6108  
##  Median : 51.50   Median : 5.000   Median :1.000   Median : 0.7797  
##  Mean   : 51.26   Mean   : 4.807   Mean   :0.866   Mean   : 0.2713  
##  3rd Qu.: 77.00   3rd Qu.: 8.000   3rd Qu.:1.000   3rd Qu.: 1.4766  
##  Max.   :100.00   Max.   :10.000   Max.   :1.000   Max.   : 3.0236  
##       x_3                 xi                  c                p         
##  Min.   :-3.97870   Min.   :-1.444460   Min.   :0.0000   Min.   :0.000  
##  1st Qu.:-0.03238   1st Qu.:-0.286542   1st Qu.:0.9427   1st Qu.:1.564  
##  Median : 1.18780   Median : 0.000000   Median :0.9886   Median :1.902  
##  Mean   : 0.44050   Mean   : 0.009578   Mean   :0.8670   Mean   :1.871  
##  3rd Qu.: 1.62290   3rd Qu.: 0.316516   3rd Qu.:1.0322   3rd Qu.:2.392  
##  Max.   : 1.88767   Max.   : 1.905138   Max.   :1.1996   Max.   :8.211  
##        q                  s                 y          
##  Min.   :  5.912   Min.   :0.01182   Min.   :-2.9837  
##  1st Qu.: 17.628   1st Qu.:0.03526   1st Qu.:-1.9109  
##  Median : 28.025   Median :0.05605   Median :-1.5011  
##  Mean   : 67.024   Mean   :0.13405   Mean   :-1.2219  
##  3rd Qu.:100.446   3rd Qu.:0.20089   3rd Qu.:-0.5556  
##  Max.   :337.118   Max.   :0.67424   Max.   : 1.1167
```

```
summary(df_share_smooth$s - df_share_smooth_delta$s)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max. 
## -1.388e-16 -1.388e-17  0.000e+00  7.046e-19  6.939e-18  2.220e-16
```

8. Write a function `solve_delta(df_share_smooth, X, M, V, delta, sigma, mu, omega)` that finds δ_{jt} that equates the actua share and the predicted share based on `compute_share_smooth_delta` by the fixed-point algorithm with an operator:

$$T(\delta_{jt}^{(r)}) = \delta_{jt}^{(r)} + \kappa \cdot \log\left(\frac{s_{jt}}{\sigma_{jt}[\delta^{(r)}]}\right),$$

where s_{jt} is the actual share of product j in market t and $\sigma_{jt}[\delta^{(r)}]$ is the predicted share of product j in market t given $\delta^{(r)}$. Multiplying κ is for the numerical stability. I set the value at $\kappa = 1$. Adjust it if the algorithm did not work. Set the stopping criterion at $\max_{jt} |\delta_{jt}^{(r+1)} - \delta_{jt}^{(r)}| < \lambda$. Set λ at 10^{-3}. Make sure that δ_{i0t} is always set at zero while the iteration.

Start the algorithm with the true δ_{jt} and check if the algorithm returns (almost) the same δ_{jt} when the actual and predicted smooth share are equated.

```
kappa <- 1
lambda <- 1e-3
delta_new <-
  solve_delta(df_share_smooth, X, M, V, delta, sigma, mu, omega, kappa, lambda)
head(delta_new)
```

```
## # A tibble: 6 x 3
##       t     j delta
##   <int> <dbl> <dbl>
## 1     1     0  0
## 2     1     6 -2.40
## 3     2     0  0
## 4     2     1 -1.15
## 5     2    10 -1.22
## 6     3     0  0
```

```
summary(delta_new$delta - delta$delta)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -3.553e-15 -4.441e-16  0.000e+00 -5.745e-17  0.000e+00  1.776e-15
```

9. Check how long it takes to compute the limit δ under the Monte Carlo shocks starting from the true δ to match with `df_share_smooth`. This is approximately the time to evaluate the objective function.

```
delta_new <-
  solve_delta(df_share_smooth, X, M, V_mcmc, delta, sigma, mu, omega, kappa, lambda)
save(delta_new, file = "data/A4_delta_new.RData")
```

```
delta_new <- get(load(file = "data/A4_delta_new.RData"))
summary(delta_new$delta - delta$delta)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.30836 -0.30499  0.00000 -0.06422  0.16800  1.36169
```

10. We use the marginal cost c_{jt} as the excluded instrumental variable for p_{jt}. Let Ψ be the weighing matrix for the GMM estimator. For now, let it be the identity matrix. Write a function `compute_theta_linear(df_share_smooth, delta, mu, omega, Psi)` that returns the optimal linear parameters associated with the data and δ. Notice that we only obtain β_0 in this way because α_0 is directly computed from the non-linear parameters by $-\exp(\mu + \omega^2/2)$. The first order condition for β_0 is:

$$\beta_0 = (X'W\Phi^{-1}W'X)^{-1}X'W\Phi^{-1}W'[\delta - \alpha_0 p], \tag{13.1}$$

where

$$X = \begin{pmatrix} x'_{11} \\ \vdots \\ x'_{J_1 1} \\ \vdots \\ x'_{1T} \\ \vdots \\ x_{J_T T} \end{pmatrix} \tag{13.2}$$

$$W = \begin{pmatrix} x'_{11} & c_{11} \\ \vdots & \vdots \\ x'_{J_1 1} & c_{J_1 1} \\ \vdots & \vdots \\ x'_{1T} & c_{1T} \\ \vdots & \vdots \\ x_{J_T T} & c_{J_T T} \end{pmatrix}, \tag{13.3}$$

$$\delta = \begin{pmatrix} \delta_1 1 \\ \vdots \\ \delta_{J_1 1} \\ \vdots \\ \delta_1 T \\ \vdots \\ \delta_{J_T T} \end{pmatrix} \tag{13.4}$$

,

where $\alpha_0 = -\exp(\mu + \omega^2/2)$. Notice that X and W does not include rows for the outwide option.

```
Psi <- diag(length(beta) + 1)
theta_linear <-
  compute_theta_linear(df_share_smooth, delta, mu, omega, Psi)
cbind(theta_linear, beta)
```

```
##          delta       beta
## x_1  3.9935855  4.0000000
## x_2  0.1552469  0.1836433
## x_3 -0.7838712 -0.8356286
```

11. Write a function `solve_xi(df_share_smooth, delta, beta, mu, omega)` that computes the values of ξ that are implied from the data, δ, and the linear parameters. Check that the (almost) true values are returned when true δ and the true linear parmaeters are passed to the function. Notice that the returend ξ should not include rows for the outside option.

```
xi_new <- solve_xi(df_share_smooth, delta, beta, mu, omega)
head(xi_new)
```

```
##               xi
## [1,] -0.05139386
## [2,] -0.19714498
## [3,] -0.35374758
## [4,]  0.18229098
```

```
## [5,]  0.38426646
## [6,] -0.05617311
```

```
xi_true <-
  df_share_smooth %>%
  dplyr::filter(j != 0) %>%
  dplyr::select(xi)
summary(xi_true - xi_new)
```

```
##        xi
##  Min.   :-4.441e-16
##  1st Qu.:-8.327e-17
##  Median : 0.000e+00
##  Mean   :-4.134e-18
##  3rd Qu.: 6.765e-17
##  Max.   : 6.661e-16
```

11. Write a function `GMM_objective_A4(theta_nonlinear, delta, df_share_smooth, Psi, X, M, V_mcmc, kappa, lambda)` that returns the value of the GMM objective function as a function of non-linear parameters `mu`, `omega`, and `sigma`:

$$\min_{\theta} \xi(\theta)' W \Phi^{-1} W' \xi(\theta),$$

where $\xi(\theta)$ is the values of ξ that solves:

$$s = \sigma(p, x, \xi),$$

given parameters θ. Note that the row of $\xi(\theta)$ and W do not include the rows for the outside options.

```
# non-linear parmaeters
theta_nonlinear <- c(mu, omega, sigma)
```

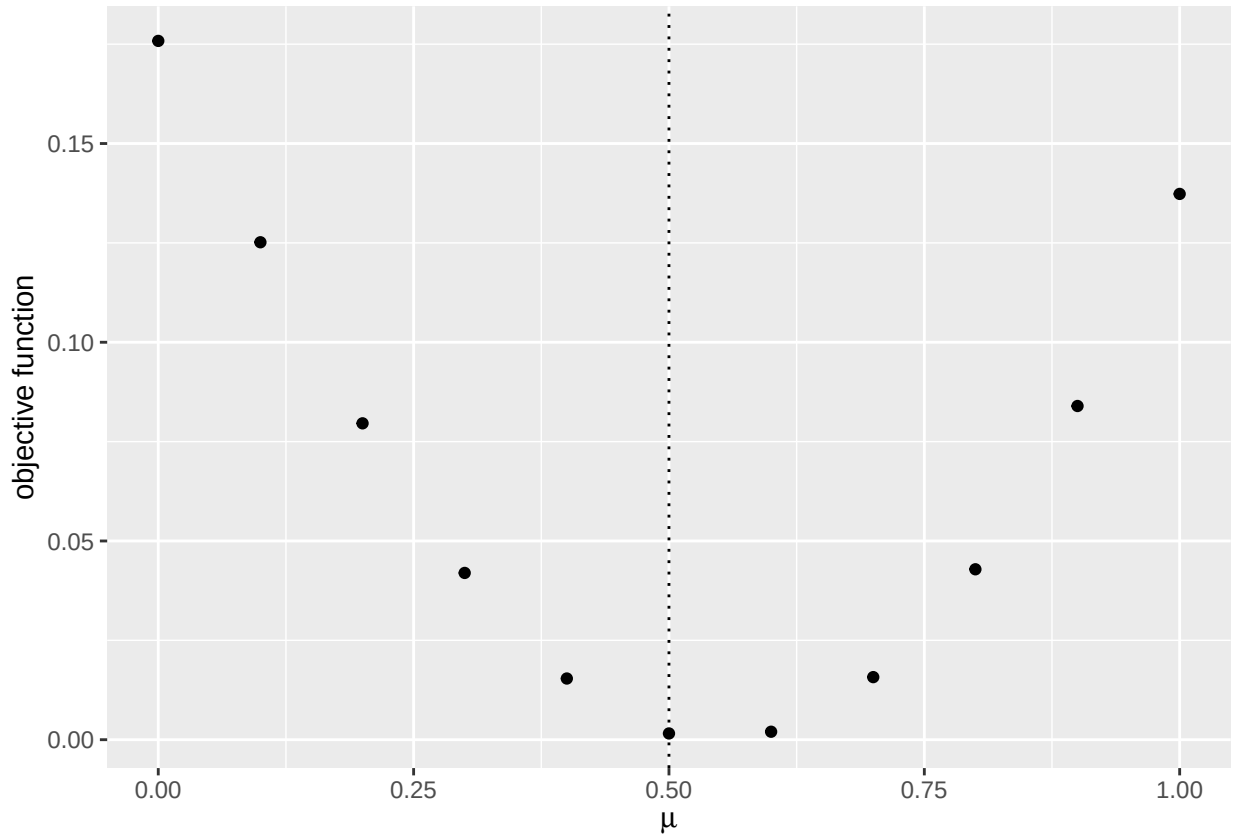
```
# compute GMM objective function
objective <-
  GMM_objective_A4(theta_nonlinear, delta, df_share_smooth, Psi,
                   X, M, V_mcmc, kappa, lambda)
save(objective, file = "data/A4_objective.RData")
```

```
objective <- get(load(file = "data/A4_objective.RData"))
objective
```

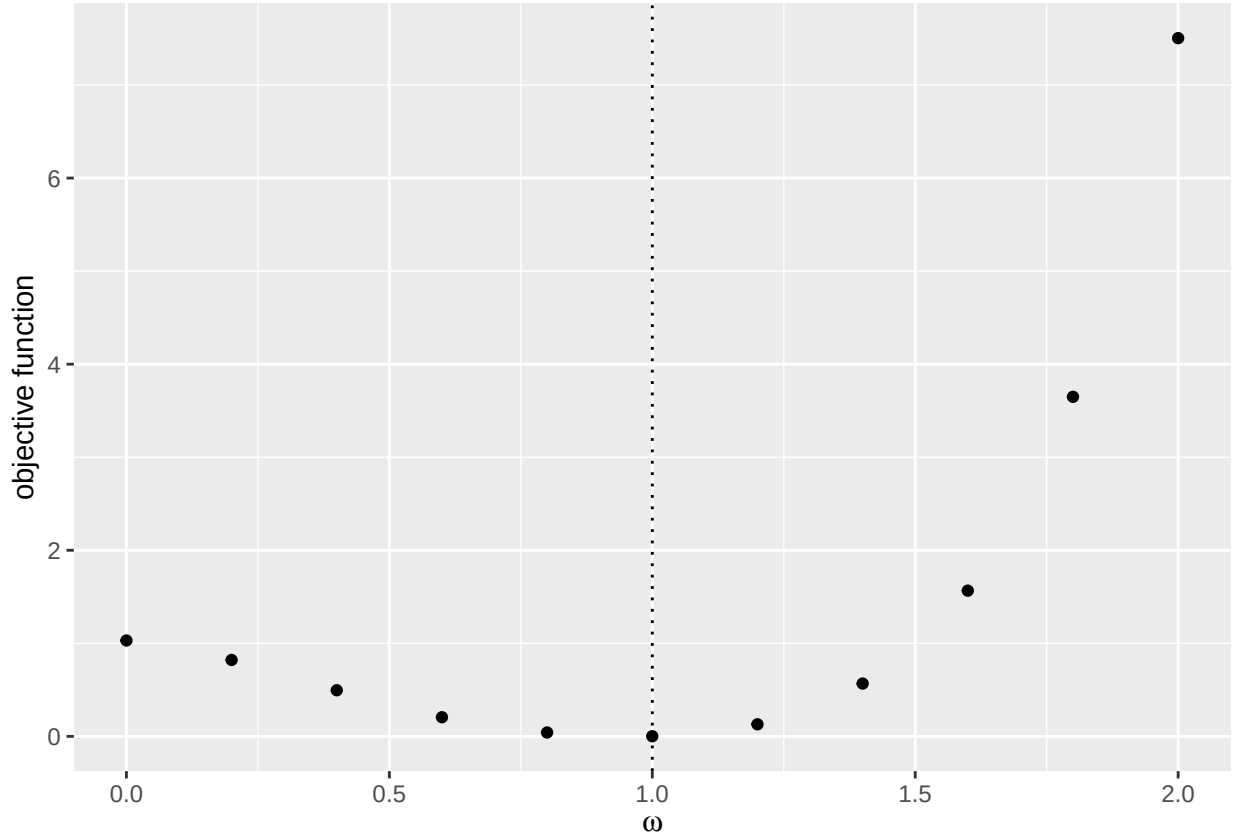
```
##           xi
## xi 0.1368324
```

12. Draw a graph of the objective function that varies each non-linear parameter from 0, 0.2, $\cdots$, 2.0 of the true value. Try with the actual shocks `V`.

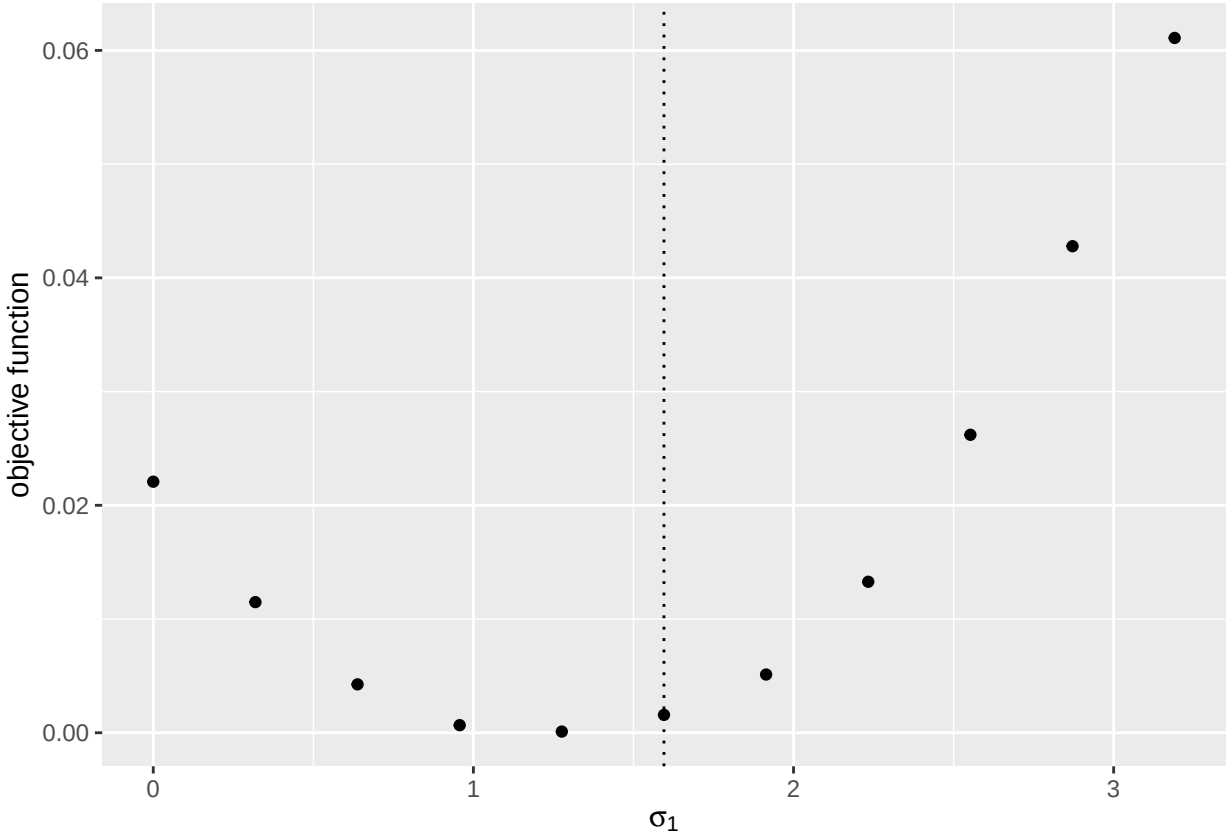
```
## [[1]]
```



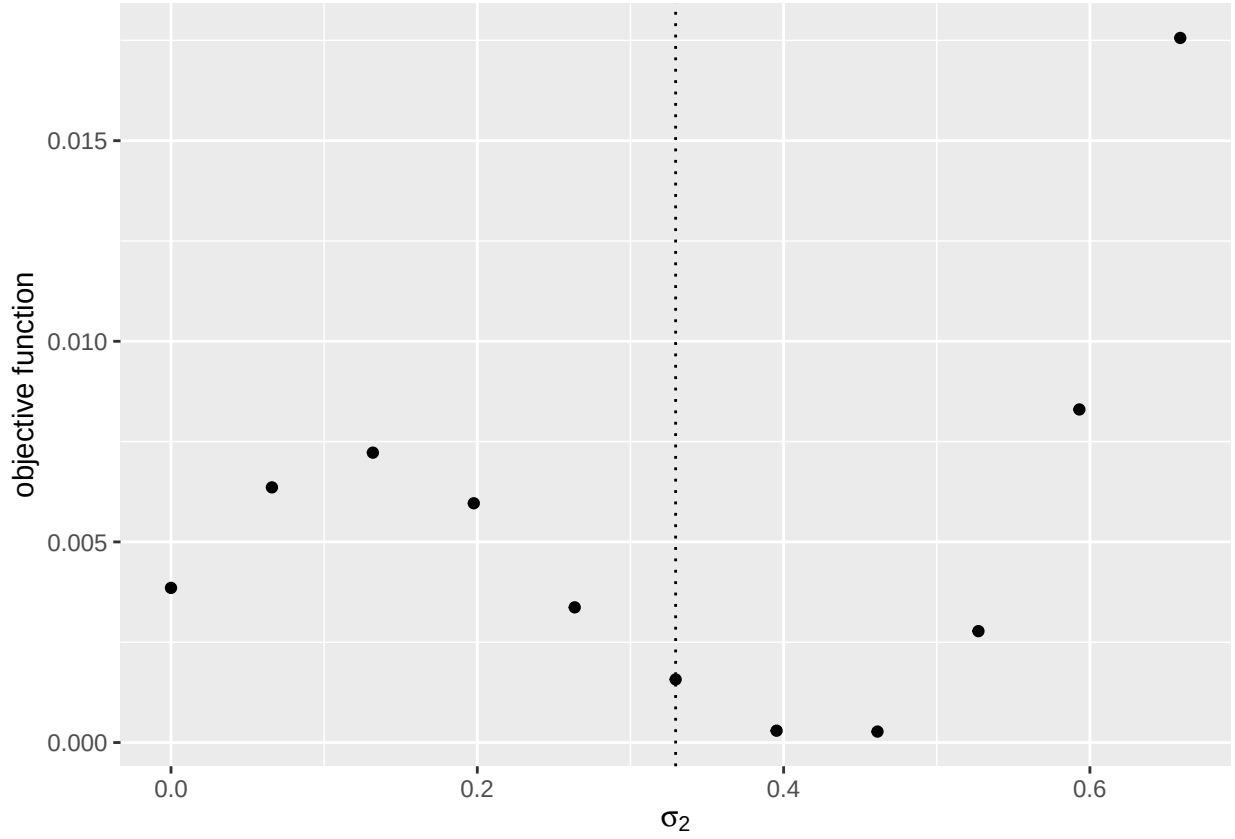

```
##
## [[2]]
```



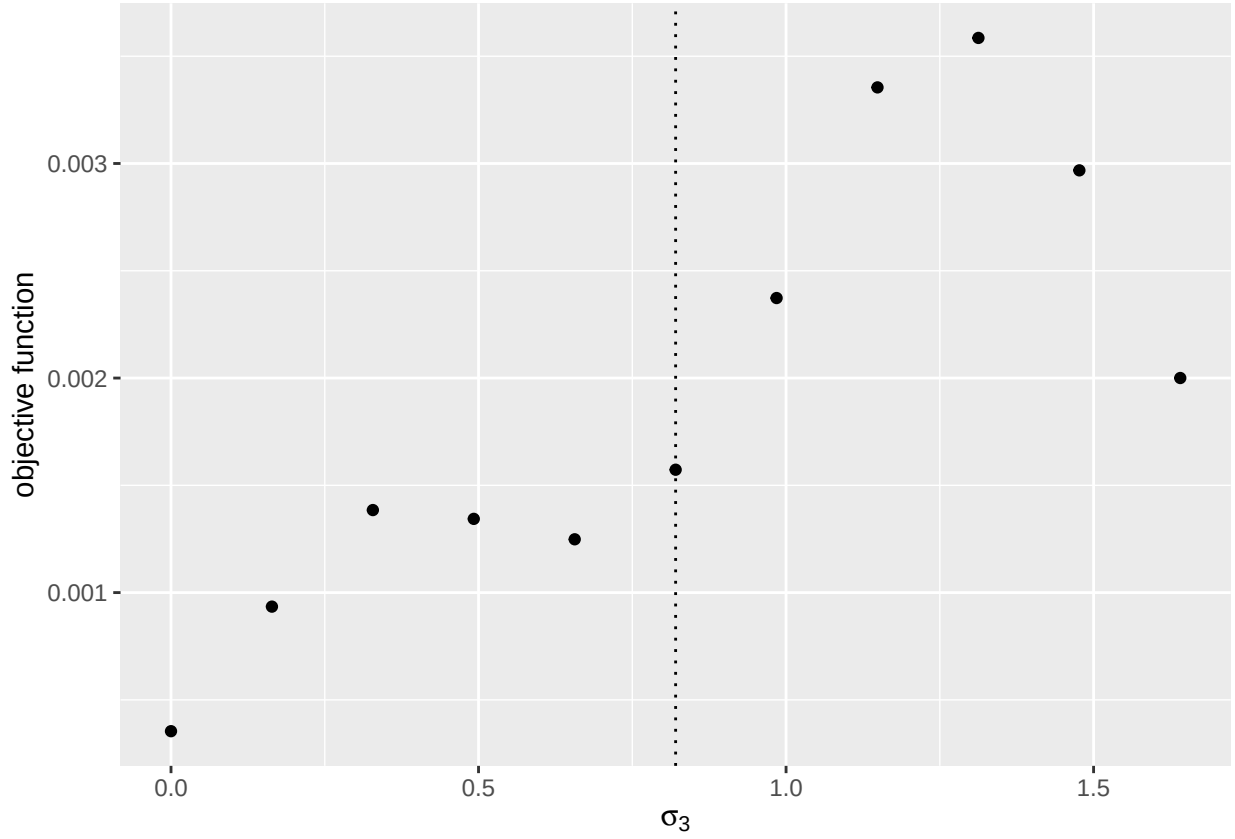

```
##
## [[3]]
```




```
##
## [[4]]
```




```
##
## [[5]]
```



13. Find non-linear parameters that minimize the GMM objective function. Because standard deviations of the same absolute value with positive and negative values have almost the same implication for the data, you can take the absolute value if the estimates of the standard deviations happened to be negative (Another way is to set the non-negativity constraints on the standard deviations).

```
## $par
## [1] 0.3635051 0.7502512 1.5852184 0.4223104 0.8757237
##
## $value
## [1] 2.064906e-18
##
## $counts
## function gradient
##       54       11
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
##           true  estimate
## [1,] 0.5000000 0.3635051
## [2,] 1.0000000 0.7502512
## [3,] 1.5952808 1.5852184
## [4,] 0.3295078 0.4223104
## [5,] 0.8204684 0.8757237
```

Chapter 14

Assignment 5: Merger Simulation

The deadline is **April 1 1:30pm**.

14.1 Simulate data

We simulate data from a discrete choice model that is the same with in assignment 4 **except for that the price is derived from the Nash equlibrium**. There are T markets and each market has N consumers. There are J products and the indirect utility of consumer i in market t for product j is:

$$u_{itj} = \beta_{it}' x_j + \alpha_{it} p_{jt} + \xi_{jt} + \epsilon_{ijt},$$

where ϵ_{ijt} is an i.i.d. type-I extreme random variable. x_j is K-dimensional observed characteristics of the product. p_{jt} is the retail price of the product in the market.

ξ_{jt} is product-market specific fixed effect. p_{jt} can be correlated with ξ_{jt} but x_{jt}s are independent of ξ_{jt}. $j = 0$ is an outside option whose indirect utility is:

$$u_{it0} = \epsilon_{i0t},$$

where ϵ_{i0t} is an i.i.d. type-I extreme random variable.

β_{it} and α_{it} are different across consumers, and they are distributed as:

$$\beta_{itk} = \beta_{0k} + \sigma_k \nu_{itk},$$

$$\alpha_{it} = -\exp(\mu + \omega \upsilon_{it}) = -\exp(\mu + \frac{\omega^2}{2}) + [-\exp(\mu + \omega \upsilon_{it}) + \exp(\mu + \frac{\omega^2}{2})] \equiv \alpha_0 + \tilde{\alpha}_{it},$$

where ν_{itk} for $k = 1, \cdots, K$ and υ_{it} are i.i.d. standard normal random variables. α_0 is the mean of α_i and $\tilde{\alpha}_i$ is the deviation from the mean.

Given a choice set in the market, $\mathcal{J}_t \cup \{0\}$, a consumer chooses the alternative that maximizes her utility:

$$q_{ijt} = 1\{u_{ijt} = \max_{k \in \mathcal{J}_t \cup \{0\}} u_{ikt}\}.$$

The choice probability of product j for consumer i in market t is:

$$\sigma_{ijt}(p_t, x_t, \xi_t) = \mathbb{P}\{u_{ijt} = \max_{k \in \mathcal{J}_t \cup \{0\}} u_{ikt}\}.$$

Suppose that we only observe the (smooth) share data:

$$s_{jt}(p_t, x_t, \xi_t) = \frac{1}{N}\sum_{i=1}^{N} \sigma_{ijt}(p_t, x_t, \xi_t) = \frac{1}{N}\sum_{i=1}^{N} \frac{\exp(u_{ijt})}{1 + \sum_{k \in \mathcal{J}_t \cup \{0\}} \exp(u_{ikt})}.$$

along with the product-market characteristics x_{jt} and the retail prices p_{jt} for $j \in \mathcal{J}_t \cup \{0\}$ for $t = 1, \cdots, T$. We do not observe the choice data q_{ijt} nor shocks $\xi_{jt}, \nu_{it}, \upsilon_{it}, \epsilon_{ijt}$.

We draw ξ_{jt} from i.i.d. normal distribution with mean 0 and standard deviation σ_ξ.

1. Set the seed, constants, and parameters of interest as follows.

```
# set the seed
set.seed(1)
# number of products
J <- 10
# dimension of product characteristics including the intercept
K <- 3
# number of markets
T <- 100
# number of consumers per market
N <- 500
# number of Monte Carlo
L <- 500
```

```
# set parameters of interests
beta <- rnorm(K);
beta[1] <- 4
beta
```

```
## [1]  4.0000000  0.1836433 -0.8356286
```

```
sigma <- abs(rnorm(K)); sigma
```

```
## [1] 1.5952808 0.3295078 0.8204684
```

```
mu <- 0.5
omega <- 1
```

Generate the covariates as follows.

The product-market characteristics:

$$x_{j1} = 1, x_{jk} \sim N(0, \sigma_x), k = 2, \cdots, K,$$

where σ_x is referred to as `sd_x` in the code.

The product-market-specific unobserved fixed effect:

$$\xi_{jt} \sim N(0, \sigma_\xi),$$

where $\sigma_x i$ is referred to as `sd_xi` in the code.

The marginal cost of product j in market t:

$$c_{jt} \sim \text{logNormal}(0, \sigma_c),$$

where σ_c is referred to as `sd_c` in the code.

The price is determined by a Nash equilibrium. Let Δ_t be the $J_t \times J_t$ ownership matrix in which the (j,k)-th element δ_{tjk} is equal to 1 if product j and k are owned by the same firm and 0 otherwise. Assume that

$\delta_{tjk} = 1$ if and only if $j = k$ for all $t = 1, \cdots, T$, i.e., each firm owns only one product. Next, define Ω_t be $J_t \times J_t$ matrix such that whose (j,k)-the element $\omega_{tjk}(p_t, x_t, \xi_t, \Delta_t)$ is:

$$\omega_{tjk}(p_t, x_t, \xi_t, \Delta_t) = -\frac{\partial s_{jt}(p_t, x_t, \xi_t)}{\partial p_{kt}} \delta_{tjk}.$$

Then, the equilibrium price vector p_t is determined by solving the following equilibrium condition:

$$p_t = c_t + \Omega_t(p_t, x_t, \xi_t, \Delta_t)^{-1} s_t(p_t, x_t, \xi_t).$$

The value of the auxiliary parameters are set as follows:

```
# set auxiliary parameters
price_xi <- 1
sd_x <- 2
sd_xi <- 0.5
sd_c <- 0.05
sd_p <- 0.05
```

2. `X` is the data frame such that a row contains the characteristics vector x_j of a product and columns are product index and observed product characteristics. The dimension of the characteristics K is specified above. Add the row of the outside option whose index is 0 and all the characteristics are zero.

```
X
```

```
## # A tibble: 11 x 4
##        j   x_1     x_2     x_3
##    <dbl> <dbl>   <dbl>   <dbl>
##  1     0     0  0       0
##  2     1     1  0.975  -0.0324
##  3     2     1  1.48    1.89
##  4     3     1  1.15    1.64
##  5     4     1 -0.611   1.19
##  6     5     1  3.02    1.84
##  7     6     1  0.780   1.56
##  8     7     1 -1.24    0.149
##  9     8     1 -4.43   -3.98
## 10     9     1  2.25    1.24
## 11    10     1 -0.0899 -0.112
```

3. `M` is the data frame such that a row contains the price ξ_{jt}, marginal cost c_{jt}, and price p_{jt}. For now, set $p_{jt} = 0$ and fill the equilibrium price later. After generating the variables, drop some products in each market. In order to change the number of available products in each market, for each market, first draw J_t from a discrete uniform distribution between 1 and J. Then, drop products from each market using `dplyr::sample_frac` function with the realized number of available products. The variation in the available products is important for the identification of the distribution of consumer-level unobserved heterogeneity. Add the row of the outside option to each market whose index is 0 and all the variables take value zero.

```
M
```

```
## # A tibble: 689 x 5
##        j     t      xi     c     p
##    <dbl> <int>   <dbl> <dbl> <dbl>
##  1     0     1  0      0         0
##  2     1     1 -0.0779 0.951     0
##  3     2     1 -0.735  1.04      0
##  4     7     1  0.194  0.961     0
```

```
##  5    10     1 -0.207  1.02      0
##  6     0     2  0      0         0
##  7     8     2  0.278  0.955     0
##  8     0     3  0      0         0
##  9     3     3 -0.0562 1.02      0
## 10     0     4  0      0         0
## # ... with 679 more rows
```

4. Generate the consumer-level heterogeneity. `V` is the data frame such that a row contains the vector of shocks to consumer-level heterogeneity, (ν_i', υ_i). They are all i.i.d. standard normal random variables.

```
V
```

```
## # A tibble: 50,000 x 6
##        i     t   v_x_1    v_x_2  v_x_3      v_p
##    <int> <int>   <dbl>    <dbl>  <dbl>    <dbl>
##  1     1     1  0.559  -0.362   -0.707  0.594
##  2     2     1 -1.00   -0.306    0.324 -0.368
##  3     3     1  0.900   0.464    0.253 -0.994
##  4     4     1  0.152  -0.640   -0.622 -0.290
##  5     5     1 -0.301  -2.18     0.151  0.475
##  6     6     1  0.0512 -1.05     0.430  0.159
##  7     7     1  0.292   0.00469  1.29   0.761
##  8     8     1  0.245  -0.330   -0.420 -0.00911
##  9     9     1  0.0827 -0.00644 -1.59  -1.02
## 10    10     1 -0.0404 -0.0764   0.259 -0.911
## # ... with 49,990 more rows
```

5. We use `compute_indirect_utility(df, beta, sigma, mu, omega)`, `compute_choice_smooth(X, M, V, beta, sigma, mu, omega)`, and `compute_share_smooth(X, M, V, beta, sigma, mu, omega)` to compute $s_t(p_t, x_t, \xi_t)$. On top of this, we need a function `compute_derivative_share_smooth(X, M, V, beta, sigma, mu, omega)` that approximate:

$$\frac{\partial s_{jt}(p_t, x_t, \xi_t)}{\partial p_{kt}} = \begin{cases} \frac{1}{N}\sum_{i=1}^{N} \alpha_i \sigma_{ijt}(p_t, x_t, \xi_t)[1 - \sigma_{ijt}(p_t, x_t, \xi_t)] & \text{if } j = k \\ -\frac{1}{N}\sum_{i=1}^{N} \alpha_i \sigma_{ijt}(p_t, x_t, \xi_t)\sigma_{kt}(p_t, x_t, \xi_t)] & \text{if } j \neq k. \end{cases}$$

The returned object should be a list across markets and each element of the list should be $J_t \times J_t$ matrix whose (j,k)-th element is $\partial s_{jt}/\partial p_{it}$ (do not include the outside option). The computation will be looped across markets. I recommend to use a parallel computing for this loop.

```
derivative_share_smooth <-
  compute_derivative_share_smooth(X, M, V, beta, sigma, mu, omega)
derivative_share_smooth[[1]]
```

```
##             [,1]        [,2]        [,3]        [,4]
## [1,] -0.55323517  0.07416782  0.22222078  0.24032215
## [2,]  0.07416782 -0.17952618  0.05347946  0.04757626
## [3,]  0.22222078  0.05347946 -0.47677137  0.18862102
## [4,]  0.24032215  0.04757626  0.18862102 -0.48803506
```

```
derivative_share_smooth[[T]]
```

```
##             [,1]        [,2]        [,3]        [,4]        [,5]
## [1,] -0.07358769  0.01724997  0.01539716  0.00728682  0.01037598
## [2,]  0.01724997 -0.18464980  0.03941164  0.02590020  0.05210013
## [3,]  0.01539716  0.03941164 -0.14947032  0.01760727  0.02950338
## [4,]  0.00728682  0.02590020  0.01760727 -0.11933630  0.04443484
```

```
## [5,]  0.01037598  0.05210013  0.02950338  0.04443484 -0.18480392
## [6,]  0.02286198  0.04875629  0.04659132  0.02338740  0.04568463
##             [,6]
## [1,]  0.02286198
## [2,]  0.04875629
## [3,]  0.04659132
## [4,]  0.02338740
## [5,]  0.04568463
## [6,] -0.18893897
```

6. Make a list `Delta` such that each element of the list is $J_t \times J_t$ matrix Δ_t.

```
Delta[[1]]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    0    1    0    0
## [3,]    0    0    1    0
## [4,]    0    0    0    1
```

```
Delta[[T]]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    0    0    0    0    0
## [2,]    0    1    0    0    0    0
## [3,]    0    0    1    0    0    0
## [4,]    0    0    0    1    0    0
## [5,]    0    0    0    0    1    0
## [6,]    0    0    0    0    0    1
```

7. Write a function `update_price(logp, X, M, V, beta, sigma, mu, omega, Delta)` that receives a price vector $p_t^{(r)}$ and returns $p_t^{(r+1)}$ by:

$$p_t^{(r+1)} = c_t + \Omega_t(p_t^{(r)}, x_t, \xi_t, \Delta_t)^{-1} s_t(p_t^{(r)}, x_t, \xi_t).$$

The returned object should be a vector whose row represents the condition for an inside product of each market. To impose non-negativity constraint on the price vector, we pass log price and exponentiate inside the function. Iterate this until $\max_{jt} |p_{jt}^{(r+1)} - p_{jt}^{(r)}| < \lambda$, for example with $\lambda = 10^{-6}$. This iteration may or may not converge. The convergence depends on the parameters and the realization of the shocks. If the algorithm does not converge, first check the code.

```
# set the threshold
lambda <- 1e-6
# set the initial price
p <- M[M$j > 0, "p"]
logp <- log(rep(1, dim(p)[1]))
p_new <- update_price(logp, X, M, V, beta, sigma, mu, omega, Delta)
# iterate
distance <- 10000
while (distance > lambda) {
  p_old <- p_new
  p_new <- update_price(log(p_old), X, M, V, beta, sigma, mu, omega, Delta)
  distance <- max(abs(p_new - p_old))
  print(distance)
}
# save
p_actual <- p_new
save(p_actual, file = "data/A5_price_actual.RData")
```

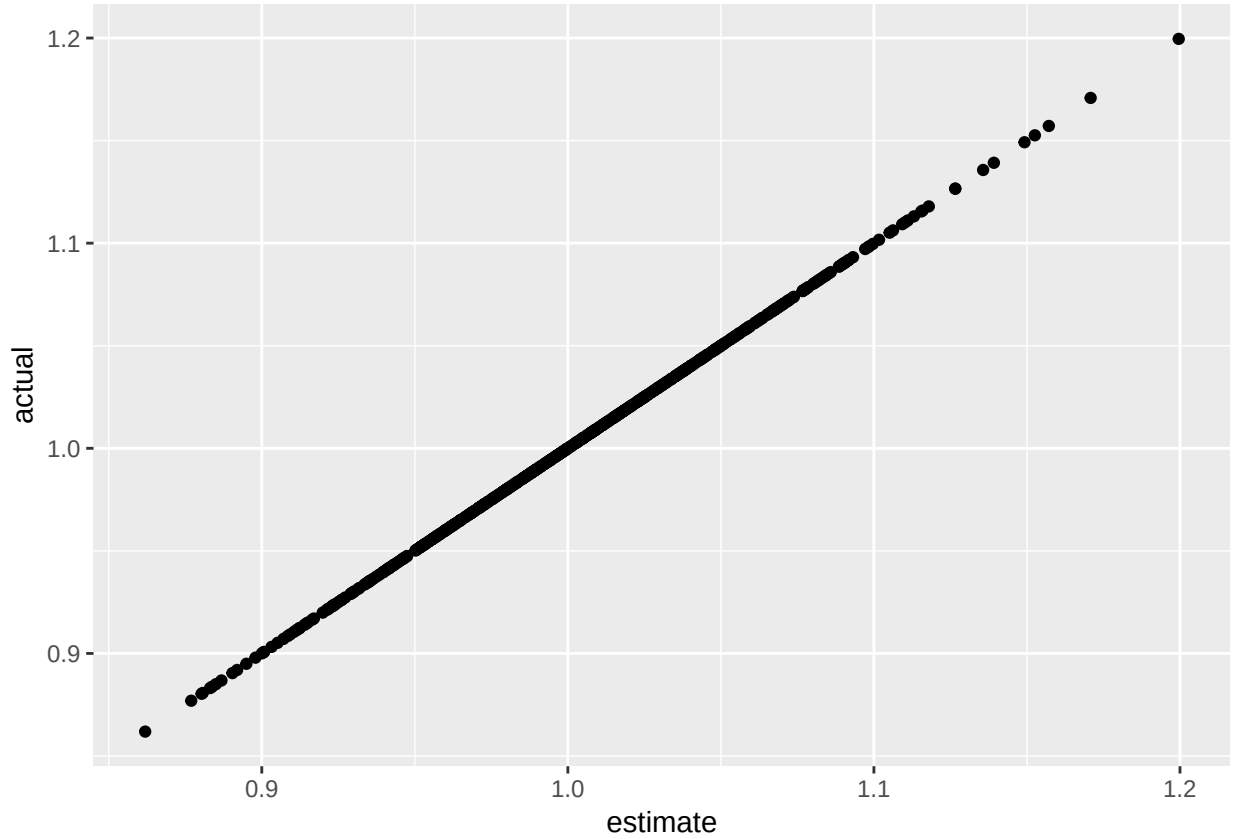
14.2 Estimate the parameters

1. Write a function `estimate_marginal_cost()` that estimate c_t by the equilibrium condition as:

$$c_t = p_t - \Omega_t(p_t, x_t, \xi_t, \Delta_t)^{-1} s_t(p_t, x_t, \xi_t)$$

Of course, in reality, we first draw Monte Carlo shocks to approximate the share, estimate the demand parameters, and use these shocks and estimates to estimate the marginal costs. In this assignment, we check the if the estimated marginal costs coincide with the true marginal costs to confirm that the codes are correctly written.

```
# load
load(file = "data/A5_price_actual.RData")
# take the logarithm
logp <- log(p_actual)
# estimate the marginal cost
marginal_cost_estimate <- estimate_marginal_cost(logp, X, M, V, beta, sigma, mu, omega, Delta)
marginal_cost_actual <- M[M$j > 0, ]$c
# plot the estimate vs actual marginal costs
marginal_cost_df <-
  data.frame(actual = marginal_cost_actual,
             estimate = marginal_cost_estimate)
ggplot(marginal_cost_df, aes(x = estimate, y = actual)) +
  geom_point()
```



2. (Optional) Translate `compute_indirect_utility`, `compute_choice_smooth`, `compute_derivative_share_smooth`, `update_price` into C++ using Rcpp and Eigen. Check that the outputs coincide at the machine precision level. I give you extra 2 points for this task on top of the usual 10 points for this assignment.

14.3 Conduct counterfactual simulation

1. Suppose that the firm of product 1 owner purchase the firms that own product 2 and 3. Let `Delta_counterfactual` be the relevant ownership matrix. Make `Delta_counterfactual`.

```
Delta_counterfactual[[1]]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    1    0    0
## [2,]    1    1    0    0
## [3,]    0    0    1    0
## [4,]    0    0    0    1
```

```
Delta_counterfactual[[T]]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    0    0    0    0    0
## [2,]    0    1    0    0    0    0
## [3,]    0    0    1    0    0    0
## [4,]    0    0    0    1    0    0
## [5,]    0    0    0    0    1    0
## [6,]    0    0    0    0    0    1
```

2. Compute the counterfactual price using the iteration with `update_price`. You can start the iteration from the equilibrium price. Show the average percentage change in the price for each product. In theory, the price of any product should not drop. But some prices can slightly drop because of the numerical errors.

```
logp <- log(p_actual)
p_new <- update_price(logp, X, M, V, beta, sigma, mu, omega, Delta_counterfactual)
distance <- 10000
while (distance > lambda) {
  p_old <- p_new
  p_new <- update_price(log(p_old), X, M, V, beta, sigma, mu, omega, Delta_counterfactual)
  distance <- max(abs(p_new - p_old))
  print(distance)
}
p_counterfactual <- p_new
save(p_counterfactual, file = "data/A5_price_counterfactual.RData")
```

j	p_change
1	0.0714188
2	0.1408300
3	0.1568128
4	-0.0016428
5	0.0053848
6	-0.0004226
7	0.0020979
8	-0.0033997
9	0.0003623
10	0.0006657

3. Write a function `compute_producer_surplus(p, marginal_cost, X, M, V, beta, sigma, mu, omega)` that returns the producer surplus for each product in each market. Compute the actual and counterfactual producer surplus under the estimated marginal costs. Show the average percentage change in the producer surplus for each product.

```
# compute actual producer surplus
producer_surplus_actual <-
  compute_producer_surplus(p_actual, marginal_cost_estimate, X, M, V, beta, sigma, mu, omega)
summary(producer_surplus_actual)
```

```
##        s
##  Min.   :0.008895
##  1st Qu.:0.038264
##  Median :0.068219
##  Mean   :0.163312
##  3rd Qu.:0.135748
##  Max.   :1.678255
```

```
# compute counterfactual producer surplus
producer_surplus_counterfactual <-
  compute_producer_surplus(p_counterfactual, marginal_cost_estimate, X, M, V, beta, sigma, mu, omega)
summary(producer_surplus_counterfactual)
```

```
##        s
##  Min.   :0.009512
##  1st Qu.:0.040473
##  Median :0.071149
##  Mean   :0.167135
##  3rd Qu.:0.140125
##  Max.   :1.678255
```

j	producer_surplus_change
1	0.0347098
2	0.0118844
3	0.0058072
4	0.0557049
5	0.0686250
6	0.0756544
7	0.0418440
8	0.0073660
9	0.0530648
10	0.0389568

4. Write a function `compute_consumer_surplus(p, X, M, V, beta, sigma, mu, omega)` that returns the consumer surplus for each consumer in each market. Compute the actual and counterfactual consumer surplus under the estimated marginal costs. Show the percentage change in the total consumer surplus.

```
# compute actual consumer surplus
consumer_surplus_actual <-
  compute_consumer_surplus(p_actual, X, M, V, beta, sigma, mu, omega)
summary(consumer_surplus_actual)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##   0.00000   0.02023   0.95853   4.14497   4.46279 236.27783
```

```
# compute counterfactual consumer surplus
consumer_surplus_counterfactual <-
```

```
  compute_consumer_surplus(p_counterfactual, X, M, V, beta, sigma, mu, omega)
summary(consumer_surplus_counterfactual)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
##    0.00000    0.01637    0.91300    4.11099    4.40189 236.29217
```

```
consumer_surplus_change <-
  (sum(consumer_surplus_counterfactual) -
     sum(consumer_surplus_actual)) /
  sum(consumer_surplus_actual)
consumer_surplus_change
```

```
## [1] -0.008198756
```

Chapter 15

Assignment 6: Entry and Exit Analysis

The deadline is **April 22 1:30pm**.

15.1 Simulate data

In this assignment, we consider a Berry-type entry model. Suppose that there are M markets indexed by $m = 1, \cdots, M$. In each market, there are N_m potential entrants such that $N_m \leq \overline{N}$. Let x_m be the K dimensional market attributes and z_{im} be the L dimensional potential entrant attributes. The size of Monte Carlo simulations in the estimation is R.

1. Set the constants as follows:

```
# set the seed
set.seed(1)
# number of markets
M <- 100
# the upper bound of the number of potential entrants
N <- 10
# the dimension of market attributes
K <- 2
# the dimension of potential entrant attributes
L <- 2
# the number of Monte Carlo simulations
R <- 100
```

The payoff of entrant i in market m is:

$$\pi_{im}(y_m) = x_m'\beta - \delta \ln\left(\sum_{i=1}^{N_m} y_{im}\right) + z_{im}'\alpha + \sqrt{1-\rho^2}\nu_{im} + \rho\epsilon_m,$$

where $y_{im} \in \{0, 1\}$ is the indicator for entrant i in market m to enter the market, and ν_{im} and ϵ_m are entrant- and market-specific idiosyncratic shocks that are drawn from an i.i.d. standard normal distribution. In each market, all the attributes and idiosyncratic shocks are observed by the potential entrants. N_m, x_m, z_{im}, and y_m are observed to econometrician but ν_{im} and ϵ_m are not.

2. Set the parameters as follows:

```
# parameters of interest
beta <- abs(rnorm(K)); beta
```

```
## [1] 0.6264538 0.1836433
```

```
alpha <- abs(rnorm(L)); alpha
```

```
## [1] 0.8356286 1.5952808
```

```
delta <- 1; delta
```

```
## [1] 1
```

```
rho <- abs(rnorm(1)); rho
```

```
## [1] 0.3295078
```

```
# auxiliary parameters
x_mu <- 1
x_sd <- 3
z_mu <- 0
z_sd <- 4
```

3. Draw exogenous variables as follows:

```
# number of potential entrants
E <- purrr::rdunif(M, 1, N); E
```

```
##   [1]  3  2  7  4  8  5  8 10  4  8 10  3  7  2  3  4  1  4  9  4  5  6  5
##  [24]  2  9  7  8  2  8  5  9  7  8  6  6  8  1  5  8  7  5  9  5  3  1  1
##  [47]  4  6  7  5 10  3  5  4  7  3  5  8  1  9  4  9  4  4  5  9  9  4  8
##  [70] 10  5  8  4  4  8  3  8  2  3  2  3  1  7  9  8  8  5  5  9  7  7  4
##  [93]  3 10  7  3  2  5 10  6
```

```
# market attributes
X <- matrix(
  rnorm(M * K, x_mu, x_sd),
  nrow = M
)
colnames(X) <- paste("x", 1:K, sep = "_")
X
```

```
##               x_1          x_2
##   [1,]  6.94119970 -2.225576890
##   [2,] -0.10166443  4.000086411
##   [3,] -2.13240388 -0.863800084
##   [4,]  2.70915888 -3.153280542
##   [5,]  0.59483619  6.607871867
##   [6,]  8.20485328  2.275301132
##   [7,]  0.88227999  0.284058697
##   [8,]  3.06921809  4.175449146
##   [9,]  1.08400648  3.659267954
##  [10,] -1.22981963 -0.857729145
##  [11,]  1.56637690  7.618307394
##  [12,] -4.41487589  0.234918910
##  [13,]  5.39666458 -3.273483951
##  [14,]  1.45976001  0.566801194
##  [15,]  7.51783501  1.622615018
##  [16,]  2.42652859  7.923935197
```

```
##  [17,] -1.12983929  1.317407104
##  [18,]  2.83217906  2.370996416
##  [19,] -1.80229289  0.768541194
##  [20,] -2.76090020 -0.002002527
##  [21,]  1.87433871  0.895821915
##  [22,] -0.32987562  3.362918817
##  [23,]  1.00331605  7.225735026
##  [24,]  1.22302397  4.082177316
##  [25,] -0.76856284  4.623725195
##  [26,] -0.70600620 -2.693970265
##  [27,]  0.59446415  3.951686710
##  [28,]  4.53426099  1.659774411
##  [29,] -3.57070040 -3.401750087
##  [30,]  2.78183856  2.563068228
##  [31,]  1.99885111  0.523736186
##  [32,]  4.18929951  5.393761936
##  [33,]  0.08744823 -1.298245999
##  [34,]  2.11005643 -0.290635262
##  [35,]  1.80129637 -1.778328492
##  [36,] -0.62756009  0.468688116
##  [37,]  4.62360342  2.206035338
##  [38,]  4.48120785 -1.195244519
##  [39,]  3.10064095  3.491119504
##  [40,]  5.76050036 -2.624248359
##  [41,]  2.67545928 -2.143953238
##  [42,] -2.82977663  5.323473121
##  [43,] -0.71979624 -2.047542396
##  [44,] -2.67383784  2.235924137
##  [45,] -0.42020191 -0.143228153
##  [46,] -0.86110003  2.228205519
##  [47,]  1.12634762  6.066619859
##  [48,] -1.73276495  5.759765300
##  [49,]  1.47408632  0.007276598
##  [50,] -0.96375393 -5.855706606
##  [51,]  6.30186181  8.492984770
##  [52,]  3.15012243  3.001198500
##  [53,]  3.73052269  2.623982008
##  [54,]  2.15255607  0.959801431
##  [55,]  6.04652824  2.530325269
##  [56,] -0.90720936  0.506872505
##  [57,] -0.38493419  2.262083930
##  [58,]  5.29684672 -0.200740232
##  [59,] -0.95208906 -3.110623633
##  [60,]  0.37785777  3.963514802
##  [61,] -0.17842379  5.559235076
##  [62,]  0.04002139  0.073778292
##  [63,]  0.16266009 -2.759869267
##  [64,]  2.48256499  2.926723917
##  [65,]  0.46800855  0.865872589
##  [66,] -0.51787239 -4.199655220
##  [67,]  5.02911648  1.006395579
##  [68,]  0.35626177 -0.890901002
##  [69,]  0.46133041 -0.022905740
##  [70,]  0.69942778 -2.469717088
```

```
##  [71,]  3.13799892  6.409425724
##  [72,]  0.77930679  0.006603891
##  [73,]  0.88709749 -3.816540237
##  [74,] -1.04498144  1.591580316
##  [75,]  0.02718918  1.789526939
##  [76,]  1.18048132 -1.957480101
##  [77,] -0.76668346 -7.666762015
##  [78,]  2.59448858 -0.921445108
##  [79,] -3.55518225  2.711522908
##  [80,]  1.91967358  0.820830172
##  [81,] -3.60934947  0.705463768
##  [82,]  0.09707162  2.682462186
##  [83,] -0.58483971 -2.559375916
##  [84,] -0.95628434  4.290331133
##  [85,]  0.82930967  0.983967915
##  [86,] -4.74307828  3.121932002
##  [87,]  4.52974994  4.102323204
##  [88,] -3.99491731  1.670441245
##  [89,] -0.39059120 -1.636122839
##  [90,] -2.34776032  4.488893668
##  [91,] -1.25245700 -5.000494834
##  [92,]  7.26149964 -0.634372220
##  [93,]  1.05218686  0.232987873
##  [94,] -2.85890159  0.501636890
##  [95,] -3.92181660  4.061391726
##  [96,]  2.35056130  1.408665679
##  [97,]  0.94432050  2.221502810
##  [98,]  0.04579488  0.791035561
##  [99,] -1.78808644  0.257006975
## [100,] -3.46238093  3.086652420
```

```
# entrant attributes
Z <-
  foreach (m = 1:M) %dopar% {
    Z_m <- matrix(
      rnorm(E[m] * L, z_mu, z_sd),
      nrow = E[m]
    )
    colnames(Z_m) <- paste("z", 1:L, sep = "_")
    return(Z_m)
  }
Z[[1]]
```

```
##           z_1        z_2
## [1,]  6.009675 -2.6090451
## [2,] -5.940622 -0.0524375
## [3,]  0.356437 -0.2639707
```

```
# unobserved market attributes
EP <- matrix(
  rnorm(M),
  nrow = M
)
EP
```

```
##               [,1]
```

```
##   [1,]  1.146228357
##   [2,] -2.403096215
##   [3,]  0.572739555
##   [4,]  0.374724407
##   [5,] -0.425267722
##   [6,]  0.951012808
##   [7,] -0.389237182
##   [8,] -0.284330662
##   [9,]  0.857409778
##  [10,]  1.719627299
##  [11,]  0.270054901
##  [12,] -0.422184010
##  [13,] -1.189113295
##  [14,] -0.331032979
##  [15,] -0.939829327
##  [16,] -0.258932583
##  [17,]  0.394379168
##  [18,] -0.851857092
##  [19,]  2.649166881
##  [20,]  0.156011676
##  [21,]  1.130207267
##  [22,] -2.289123980
##  [23,]  0.741001157
##  [24,] -1.316245160
##  [25,]  0.919803678
##  [26,]  0.398130155
##  [27,] -0.407528579
##  [28,]  1.324258630
##  [29,] -0.701231669
##  [30,] -0.580614304
##  [31,] -1.001072181
##  [32,] -0.668178607
##  [33,]  0.945184953
##  [34,]  0.433702150
##  [35,]  1.005159218
##  [36,] -0.390118664
##  [37,]  0.376370292
##  [38,]  0.244164924
##  [39,] -1.426257342
##  [40,]  1.778429287
##  [41,]  0.134447661
##  [42,]  0.765598999
##  [43,]  0.955136677
##  [44,] -0.050565701
##  [45,] -0.305815420
##  [46,]  0.893673702
##  [47,] -1.047298149
##  [48,]  1.971337386
##  [49,] -0.383632106
##  [50,]  1.654145302
##  [51,]  1.512212694
##  [52,]  0.082965734
##  [53,]  0.567220915
##  [54,] -1.024548480
```

```
##  [55,]  0.323006503
##  [56,]  1.043612458
##  [57,]  0.099078487
##  [58,] -0.454136909
##  [59,] -0.655781852
##  [60,] -0.035922423
##  [61,]  1.069161461
##  [62,] -0.483974930
##  [63,] -0.121010111
##  [64,] -1.294140004
##  [65,]  0.494312836
##  [66,]  1.307901520
##  [67,]  1.497041009
##  [68,]  0.814702731
##  [69,] -1.869788790
##  [70,]  0.482029504
##  [71,]  0.456135603
##  [72,] -0.353400286
##  [73,]  0.170489471
##  [74,] -0.864035954
##  [75,]  0.679230774
##  [76,] -0.327101015
##  [77,] -1.569082185
##  [78,] -0.367450756
##  [79,]  1.364434929
##  [80,] -0.334281365
##  [81,]  0.732750042
##  [82,]  0.946585640
##  [83,]  0.004398704
##  [84,] -0.352322306
##  [85,] -0.529695509
##  [86,]  0.739589226
##  [87,] -1.063457415
##  [88,]  0.246210844
##  [89,] -0.289499367
##  [90,] -2.264889356
##  [91,] -1.408850456
##  [92,]  0.916019329
##  [93,] -0.191278951
##  [94,]  0.803283216
##  [95,]  1.887474463
##  [96,]  1.473881181
##  [97,]  0.677268492
##  [98,]  0.379962687
##  [99,] -0.192798426
## [100,]  1.577891795
```

```
# unobserved entrant attributes
NU <-
  foreach (m = 1:M) %dopar% {
    NU_m <- matrix(
      rnorm(E[m]),
      nrow = E[m]
    )
```

```
    return(NU_m)
  }
NU[[1]]
```

```
##           [,1]
## [1,]  0.551453
## [2,] -1.081993
## [3,]  0.495678
```

4. Write a function `compute_payoff(y_m, X_m, Z_m, EP_m, NU_m, beta, alpha, delta, rho)` that returns the vector of payoffs of the potential entrants when the vector of entry decisions is `y_m`.

```
m <- 1
N_m <- dim(Z[[m]])[1]
y_m <- as.matrix(rep(1, N_m))
y_m[length(y_m)] <- 0
X_m <- X[m, , drop = FALSE]
Z_m <- Z[[m]]
EP_m <- EP[m, , drop = FALSE]
NU_m <- NU[[m]]
compute_payoff(y_m, X_m, Z_m, EP_m, NU_m, beta, alpha, delta, rho)
```

```
##           [,1]
## [1,]  5.004526
## [2,] -2.445201
## [3,]  0.000000
```

5. Assume that the order of entry is predetermined. Assume that the potential entrants sequentially decide entry according to the order of the payoff excluding the competitive effects, i.e.:

$$x_m'\beta + z_{im}'\alpha + \sqrt{1-\rho^2}\nu_{im} + \rho\epsilon_m.$$

Write a function `compute_sequential_entry(X_m, Z_m, EP_m, NU_m, beta, alpha, delta, rho)` that returns the equilibrium vector of entry at a market.

```
compute_sequential_entry(X_m, Z_m, EP_m, NU_m, beta, alpha, delta, rho)
```

```
##      [,1]
## [1,]    1
## [2,]    0
## [3,]    1
```

6. Next, assume $\rho = 0$. Assume that potential entrants simultaneously decide entry. Write a function `compute_simultaneous_entry(X_m, Z_m, EP_m, NU_m, beta, alpha, delta)` that returns the equilibrium vector of entry at a market.

```
compute_simultaneous_entry(X_m, Z_m, EP_m, NU_m, beta, alpha, delta)
```

```
##      [,1]
## [1,]    1
## [2,]    0
## [3,]    1
```

7. Write a function `compute_sequential_entry_across_markets(X, Z, EP, NU, beta, alpha, delta, rho)` compute the equilibrium entry vectors under the assumption of sequential entry. The output should be a list of entry vectors across markets. Write a function to compute the equilibrium payoffs across markets, `compute_payoff_across_markets(Y, X, Z, EP, NU, beta, alpha, delta, rho)` and check that the payoffs under the equilibrium entry vectors are non-negative. Otherwise, there are some bugs in the code.

```
Y_sequential <-
  compute_sequential_entry_across_markets(X, Z, EP, NU, beta, alpha, delta, rho)
Y_sequential[[1]]
```

```
##      [,1]
## [1,]    1
## [2,]    0
## [3,]    1
```

```
Y_sequential[[M]]
```

```
##      [,1]
## [1,]    0
## [2,]    0
## [3,]    0
## [4,]    1
## [5,]    0
## [6,]    0
```

```
payoff_sequential <-
  compute_payoff_across_markets(Y_sequential, X, Z, EP, NU, beta, alpha, delta, rho)
min(unlist(payoff_sequential))
```

```
## [1] 0
```

8. Write a function `compute_simultaneous_entry_across_markets(X, Z, EP, NU, beta, alpha, delta, rho = 0)` compute the equilibrium entry vectors under the assumption of simultaneous entry. The output should be a list of entry vectors across markets. Check that the payoffs under the equilibrium entry vectors are non-negative. Otherwise, there are some bugs in the code. I also recommend to write this function with

```
# compute simultaneous entry across markets
Y_simultaneous <-
  compute_simultaneous_entry_across_markets(X, Z, EP, NU, beta, alpha, delta)
Y_simultaneous[[1]]
```

```
##      [,1]
## [1,]    1
## [2,]    0
## [3,]    1
```

```
Y_simultaneous[[M]]
```

```
##      [,1]
## [1,]    0
## [2,]    0
## [3,]    0
## [4,]    1
## [5,]    0
## [6,]    0
```

```
payoff_simultaneous <-
  compute_payoff_across_markets(Y_simultaneous, X, Z, EP, NU, beta, alpha, delta, rho = 0)
min(unlist(payoff_simultaneous))
```

```
## [1] 0
```

15.2 Estimate the parameters

We estimate the parameters by matching the actual and predicted number of entrants in each market. To do so, we simulate the model for R times. Under the assumption of the sequential entry, we can uniquely predict the equilibrium identify of the entrants. So, we consider the following objective function:

$$\frac{1}{RM}\sum_{r=1}^{R}\sum_{m=1}^{M}\left[\sum_{i=1}^{N_m}|y_{im}-y_{im}^{(r)}|\right]^2,$$

where $y_{im}^{(r)}$ is the entry decision in r-th simulation. On the other hand, under the assumption of the simultaneous entry, we can only uniquely predict the equilibrium number of the entrants. So, we consider the following objective function:

$$\frac{1}{RM}\sum_{r=1}^{R}\sum_{m=1}^{M}\left[\sum_{i=1}^{N_m}(y_{im}-y_{im}^{(r)})\right]^2,$$

1. Draw R unobserved shocks:

```
set.seed(1)
# unobserved market attributes
EP_mc <-
  foreach (r = 1:R) %dopar% {
    EP <- matrix(
      rnorm(M),
      nrow = M
    )
    return(EP)
}
# unobserved entrant attributes
NU_mc <-
  foreach (r = 1:R) %dopar% {
  NU <-
    foreach (m = 1:M) %do% {
      NU_m <- matrix(
        rnorm(E[m]),
        nrow = E[m]
      )
      return(NU_m)
    }
  return(NU)
}
```

2. Write a function `compute_monte_carlo_sequential_entry(X, Z, EP_mc, NU_mc, beta, alpha, delta, rho)` that returns the Monte Carlo simulation. Then, write function `compute_objective_sequential_entry(Y X, Z, EP_mc, NU_mc, theta)` that callscompute_monte_carlo_sequential_entry and returns the value of the objective function given data and parameters under the assumption of sequential entry.

```
# sequential entry
theta <- theta_sequential <-
  c(beta, alpha, delta, rho)
Y <- Y_sequential
# compute monte carlo simulations
Y_mc <-
  compute_monte_carlo_sequential_entry(
    X, Z, EP_mc, NU_mc, beta, alpha, delta, rho)
Y_mc[[1]][[1]]
```

```
##      [,1]
## [1,]    1
## [2,]    0
## [3,]    1
```

```
# compute objective function
compute_objective_sequential_entry(Y, X, Z, EP_mc, NU_mc, theta)
```

```
## [1] 0.4951
```

3. Write a function `compute_objective_simultaneous_entry(Y, X, Z, EP_mc, NU_mc, theta)` that returns the value of the objective function given data and parameters under the assumption of simultaneous entry.

```
# simultaneous entry
theta <- theta_simultaneous <-
  c(beta, alpha, delta)
Y <- Y_simultaneous
# compute monte carlo simulations
Y_mc <- compute_monte_carlo_simultaneous_entry(X, Z, EP_mc, NU_mc, beta, alpha, delta)
Y_mc[[1]][[1]]
```

```
##      [,1]
## [1,]    1
## [2,]    0
## [3,]    1
```

```
# compute objective function
compute_objective_simultaneous_entry(Y, X, Z, EP_mc, NU_mc, theta)
```

```
## [1] 0.3336
```

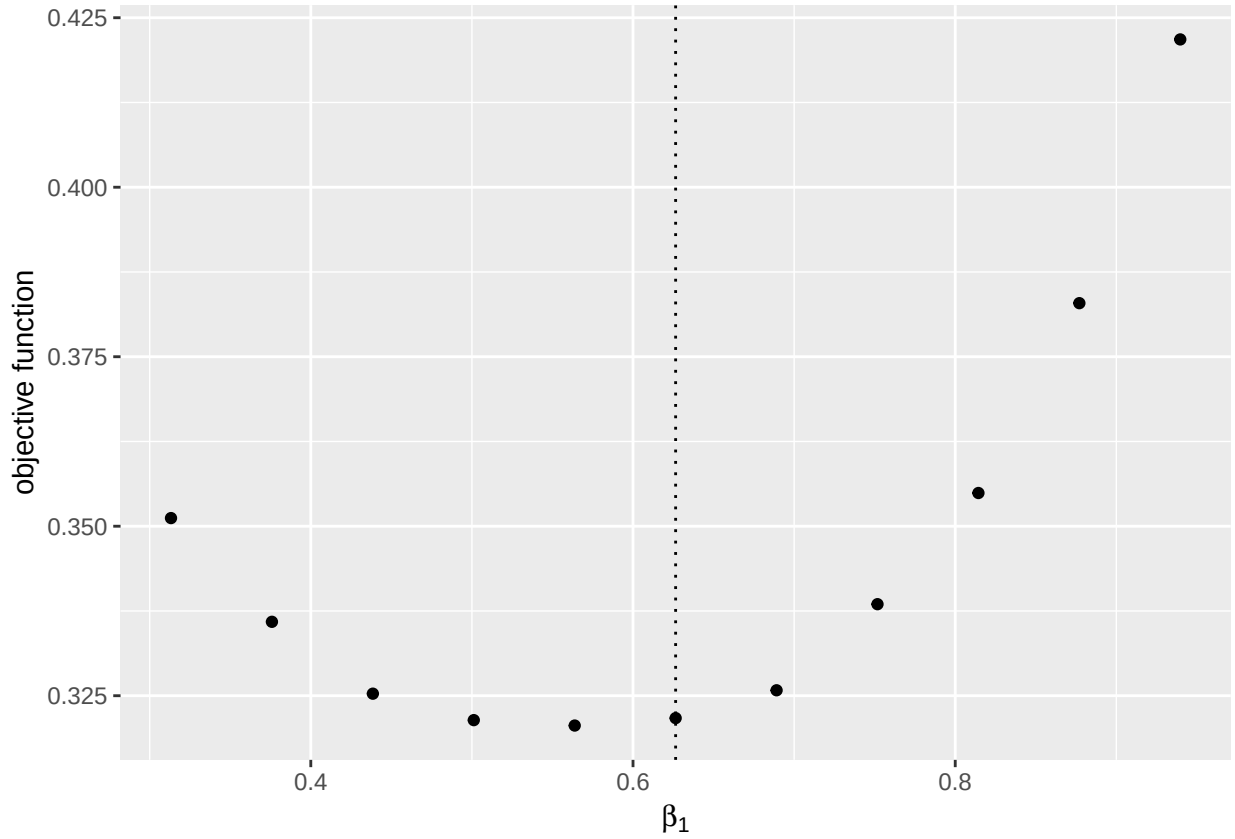
4. Check the value of the objective function around the true parameter under the assumption of the sequential entry.

```
# sequential entry
theta <- theta_sequential <-
  c(beta, alpha, delta, rho)
Y <- Y_sequential
model <- compute_sequential_entry_across_markets
label <- c(paste("\\beta_", 1:K, sep = ""),
           paste("\\alpha_", 1:L, sep = ""),
           "\\delta",
           "\\rho")
label <- paste("$", label, "$", sep = "")
# compute the graph
graph <- foreach (i = 1:length(theta)) %do% {
  theta_i <- theta[i]
  theta_i_list <- theta_i * seq(0.5, 1.5, by = 0.1)
  objective_i <-
    foreach (j = 1:length(theta_i_list),
             .combine = "rbind") %do% {
               theta_ij <- theta_i_list[j]
               theta_j <- theta
               theta_j[i] <- theta_ij
               objective_ij <-
                 compute_objective_sequential_entry(Y, X, Z, EP_mc, NU_mc, theta_j)
               return(objective_ij)
```

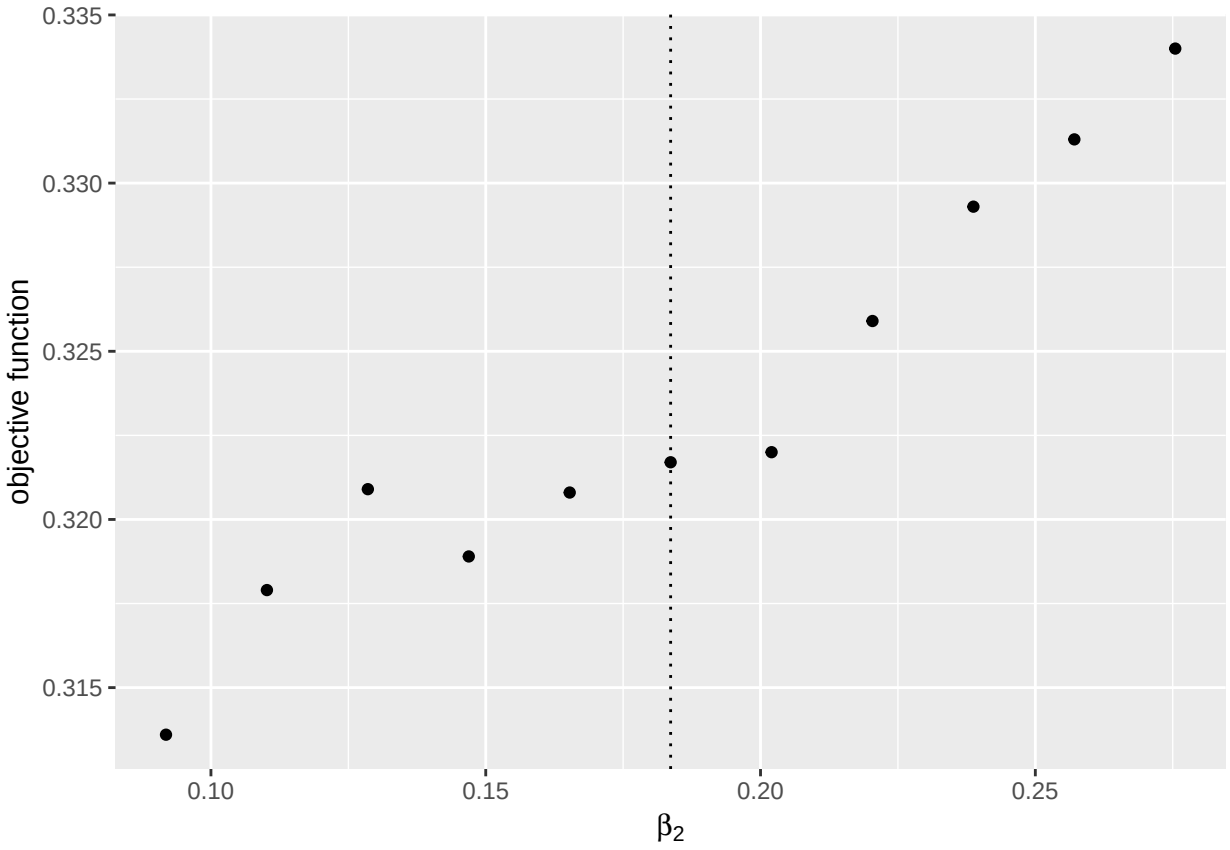
```
            }
  df_graph <- data.frame(x = theta_i_list, y = objective_i)
  g <- ggplot(data = df_graph, aes(x = x, y = y)) +
    geom_point() +
    geom_vline(xintercept = theta_i, linetype = "dotted") +
    ylab("objective function") + xlab(TeX(label[i]))
  return(g)
}
save(graph, file = "data/A6_graph_sequential.RData")
```

```
load(file = "data/A6_graph_sequential.RData")
graph
```

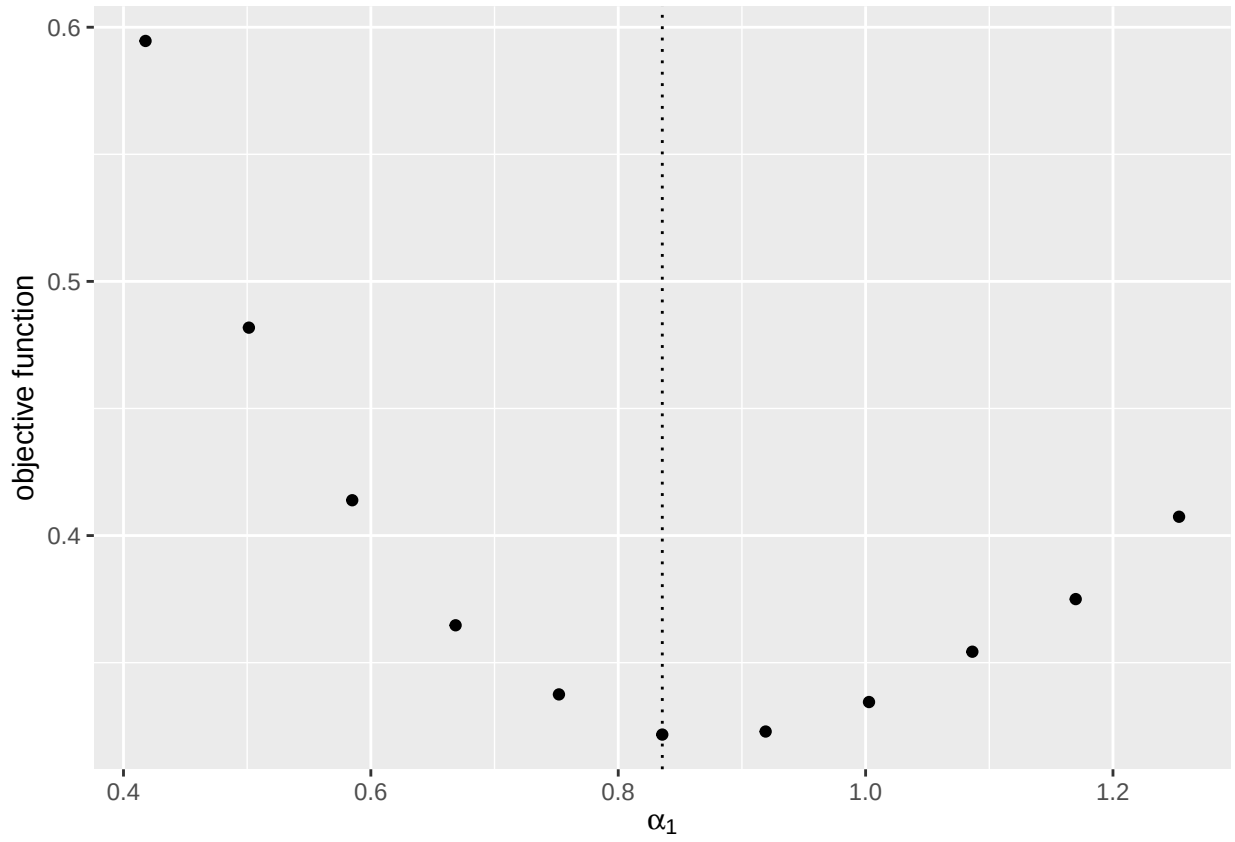
```
## [[1]]
```



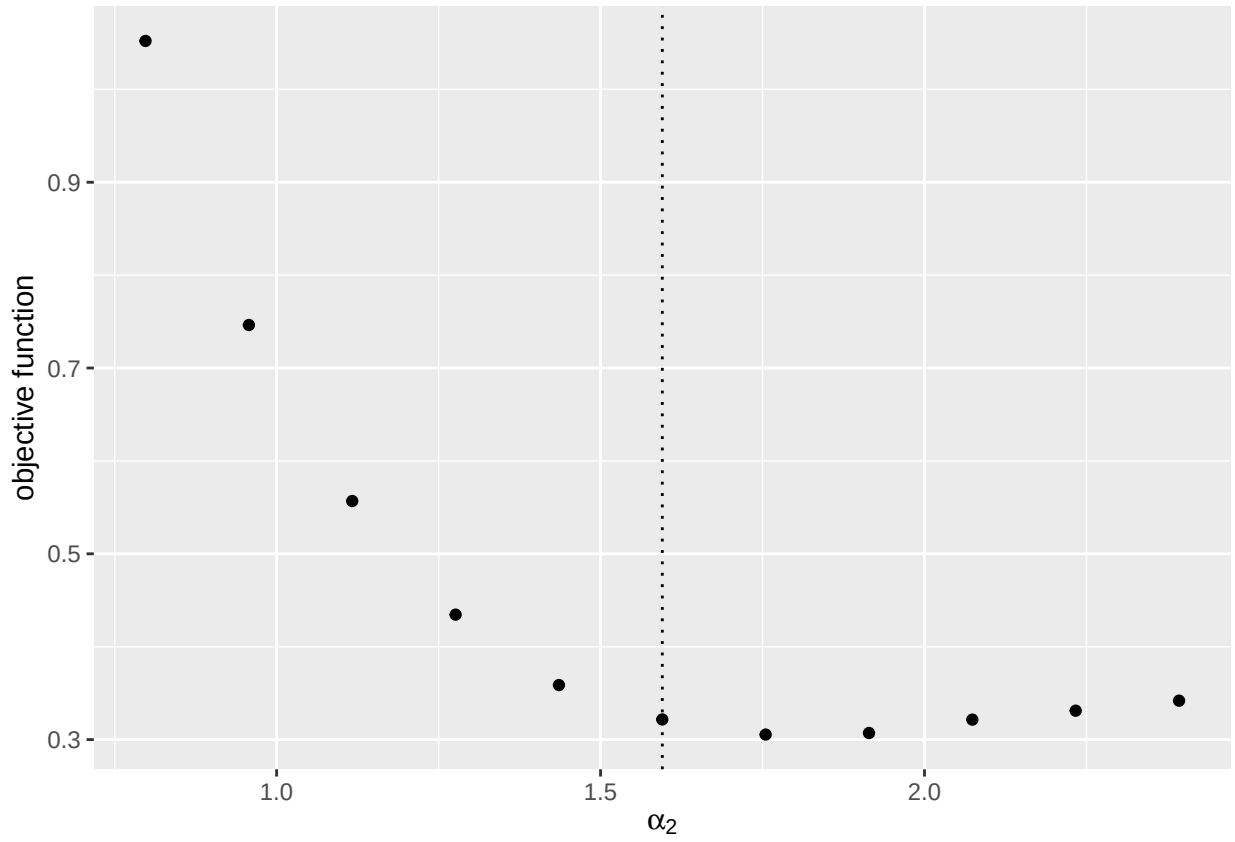

```
##
## [[2]]
```


0.335
0.330
0.325
0.320
0.315
objective function
0.10
0.15
0.20
0.25
β2

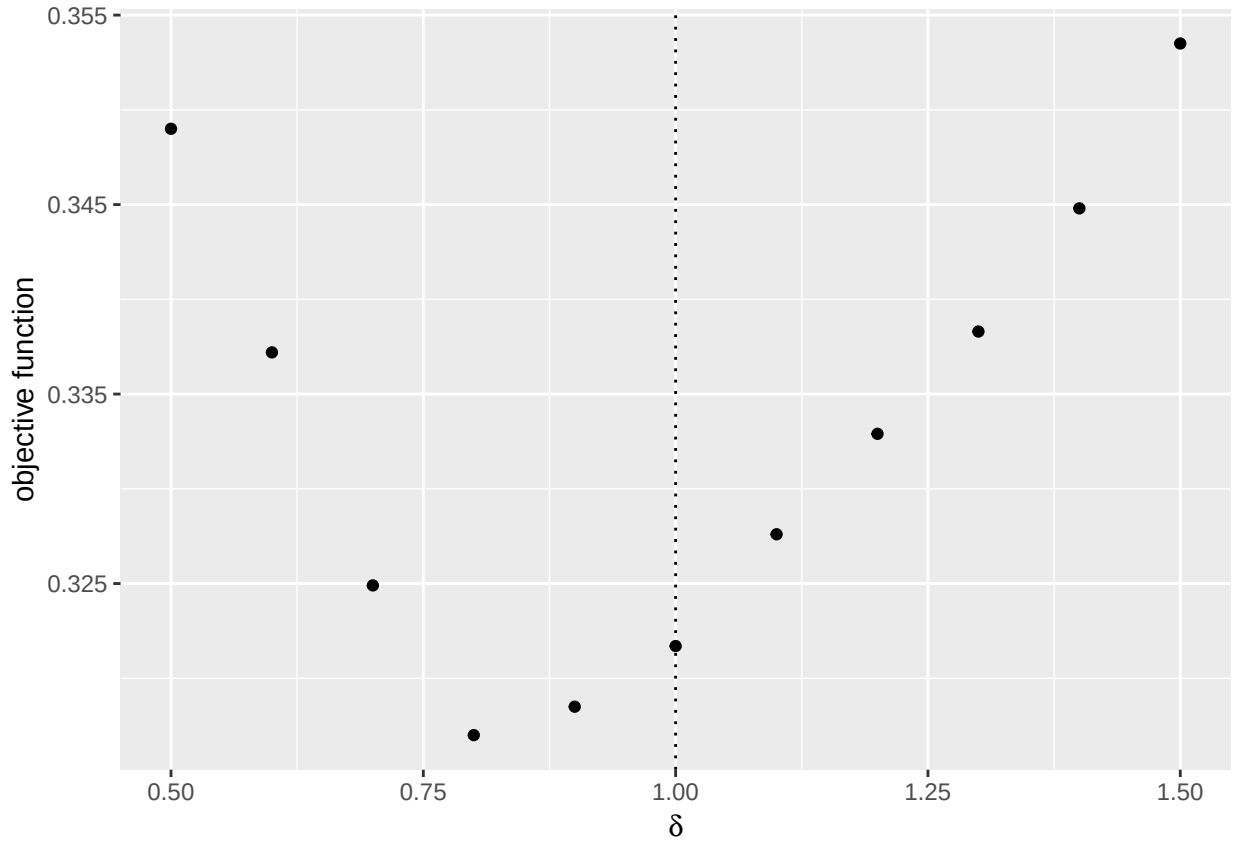
```
##
## [[3]]
```



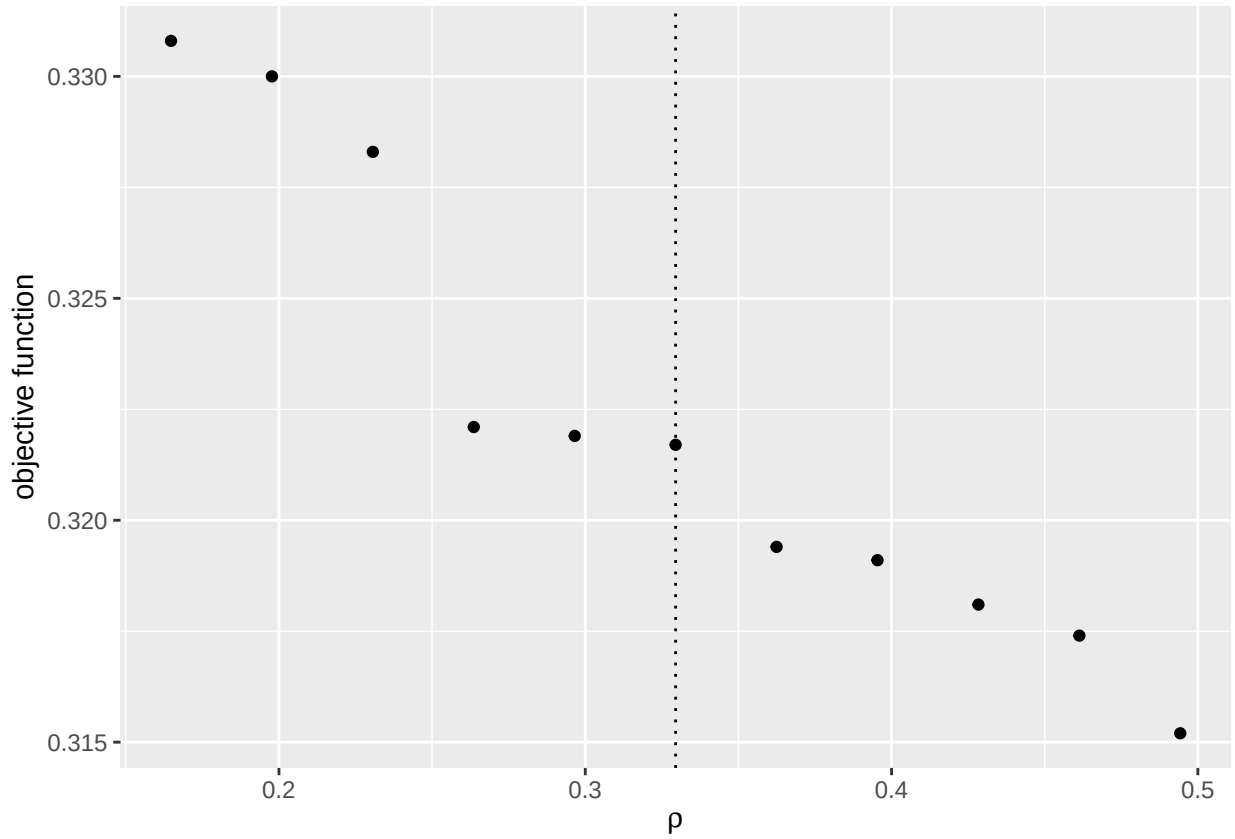

```
##
## [[4]]
```




```
##
## [[5]]
```




```
##
## [[6]]
```



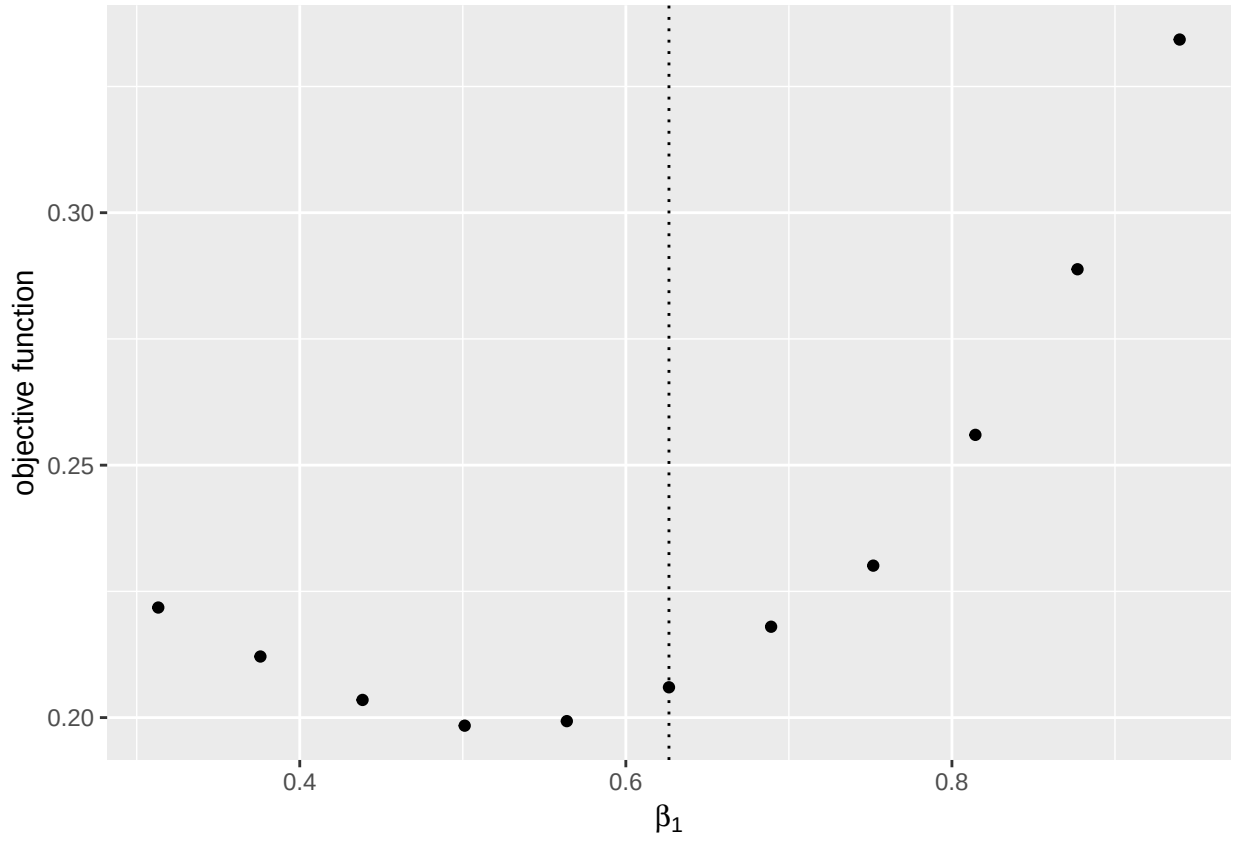
5. Check the value of the objective function around the true parameter under the assumption of the simultaneous entry.

```
# simultaneous entry
theta <- theta_simultaneous <-
  c(beta, alpha, delta)
Y <- Y_simultaneous
model <- compute_simultaneous_entry_across_markets
label <- c(paste("\\beta_", 1:K, sep = ""),
           paste("\\alpha_", 1:L, sep = ""),
           "\\delta")
label <- paste("$", label, "$", sep = "")
# compute the graph
graph <- foreach (i = 1:length(theta)) %do% {
  theta_i <- theta[i]
  theta_i_list <- theta_i * seq(0.5, 1.5, by = 0.1)
  objective_i <-
    foreach (j = 1:length(theta_i_list),
             .combine = "rbind") %do% {
               theta_ij <- theta_i_list[j]
               theta_j <- theta
               theta_j[i] <- theta_ij
               objective_ij <-
                 compute_objective_simultaneous_entry(Y, X, Z, EP_mc, NU_mc, theta_j)
               return(objective_ij)
             }
  df_graph <- data.frame(x = theta_i_list, y = objective_i)
  g <- ggplot(data = df_graph, aes(x = x, y = y)) +
```

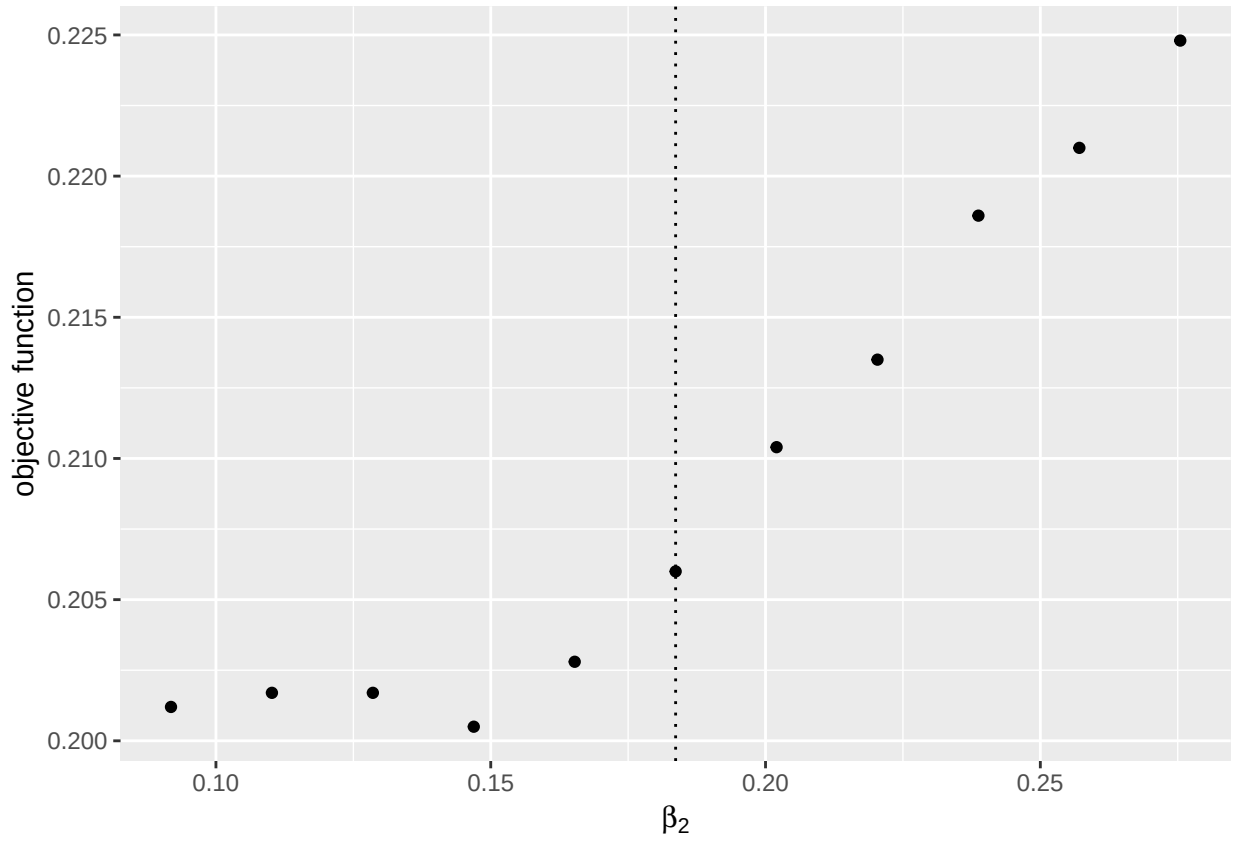
```
    geom_point() +
    geom_vline(xintercept = theta_i, linetype = "dotted") +
    ylab("objective function") + xlab(TeX(label[i]))
  return(g)
}
save(graph, file = "data/A6_graph_simultaneous.RData")
```

```
load(file = "data/A6_graph_simultaneous.RData")
graph
```

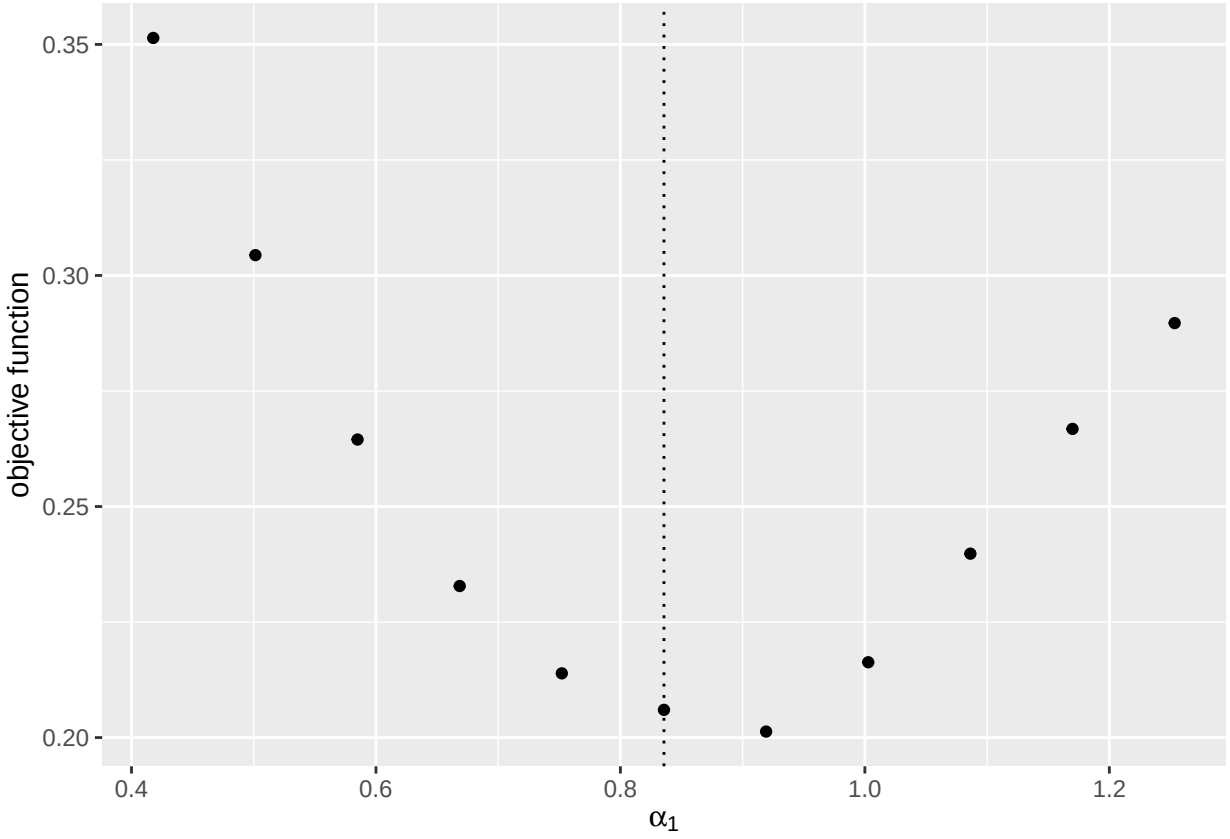
```
## [[1]]
```



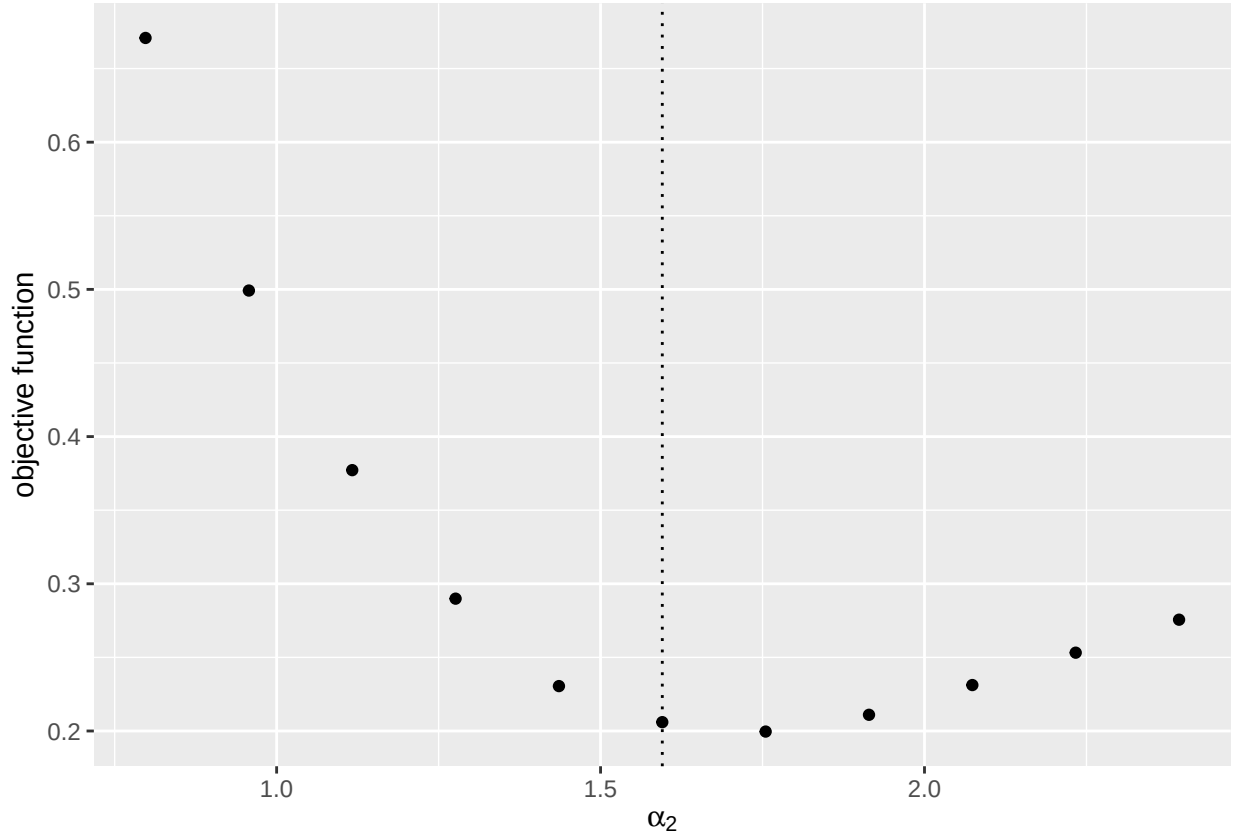

```
##
## [[2]]
```


0.225
0.220
0.215
0.210
0.205
0.200
objective function
0.10
0.15
0.20
0.25
β2

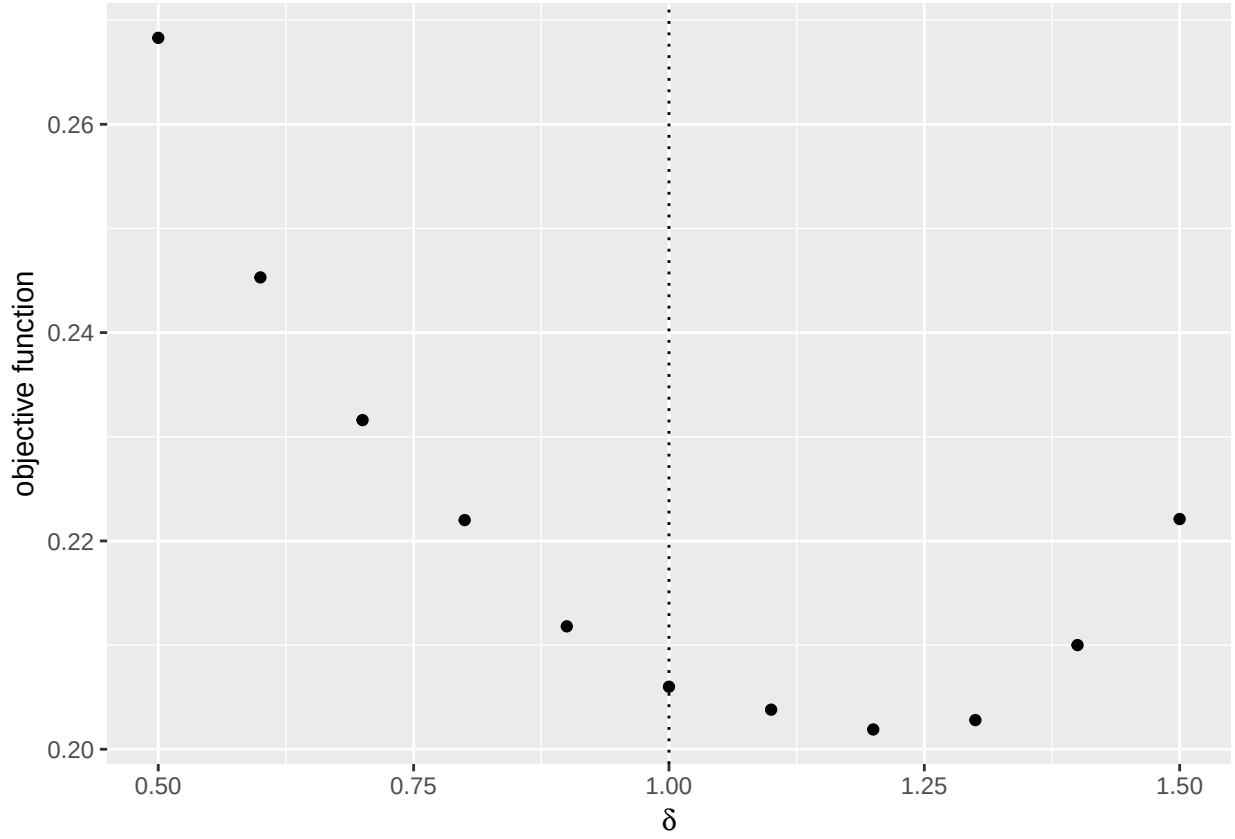
```
##
## [[3]]
```




```
##
## [[4]]
```




```
##
## [[5]]
```



6. Estimate the parameters under the assumption of the sequential entry.

```
# sequential entry
theta <- theta_sequential <-
  c(beta, alpha, delta, rho)
```

```
Y <- Y_sequential
result_sequential <-
  optim(par = theta,
        fn = compute_objective_sequential_entry,
        method = "Nelder-Mead",
        Y = Y,
        X = X,
        Z = Z,
        EP_mc = EP_mc,
        NU_mc = NU_mc)
save(result_sequential, file = "data/A6_estimate_sequential.RData")
```

```
load(file = "data/A6_estimate_sequential.RData")
result_sequential
```

```
## $par
## [1] 0.59704343 0.04938142 0.93904853 1.80598524 1.04483638 0.39060511
##
## $value
## [1] 0.2762
##
## $counts
## function gradient
```

```
##      155       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
comparison <-
  data.frame(
    actual = theta,
    estimate = result_sequential$par
  )
comparison
```

```
##      actual   estimate
## 1 0.6264538 0.59704343
## 2 0.1836433 0.04938142
## 3 0.8356286 0.93904853
## 4 1.5952808 1.80598524
## 5 1.0000000 1.04483638
## 6 0.3295078 0.39060511
```

7. Estimate the parameters under the assumption of the simultaneous entry. Set the lower bound for δ at 0.

```
# simultaneous entry
theta <- theta_simultaneous <-
  c(beta, alpha, delta, rho)
```

```
Y <- Y_sequential
result_simultaneous <-
  optim(par = theta,
        fn = compute_objective_simultaneous_entry,
        method = "Nelder-Mead",
        Y = Y,
        X = X,
        Z = Z,
        EP_mc = EP_mc,
        NU_mc = NU_mc)
save(result_simultaneous, file = "data/A6_estimate_simultaneous.RData")
```

```
load(file = "data/A6_estimate_simultaneous.RData")
result_simultaneous
```

```
## $par
## [1] 0.5787563 0.1216856 0.9521359 1.8443535 1.2177670 0.3075741
##
## $value
## [1] 0.1623
##
## $counts
## function gradient
##      141       NA
##
## $convergence
## [1] 0
```

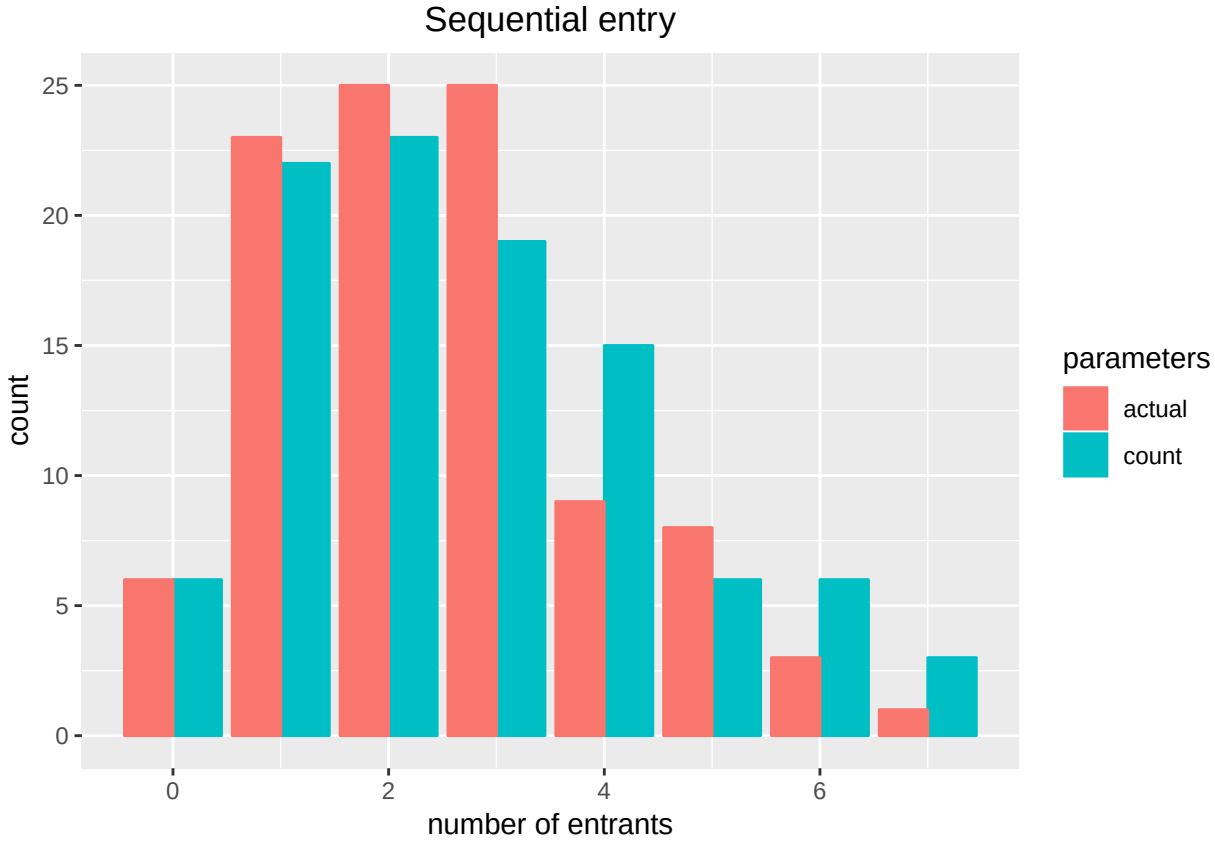
```
##
## $message
## NULL
```

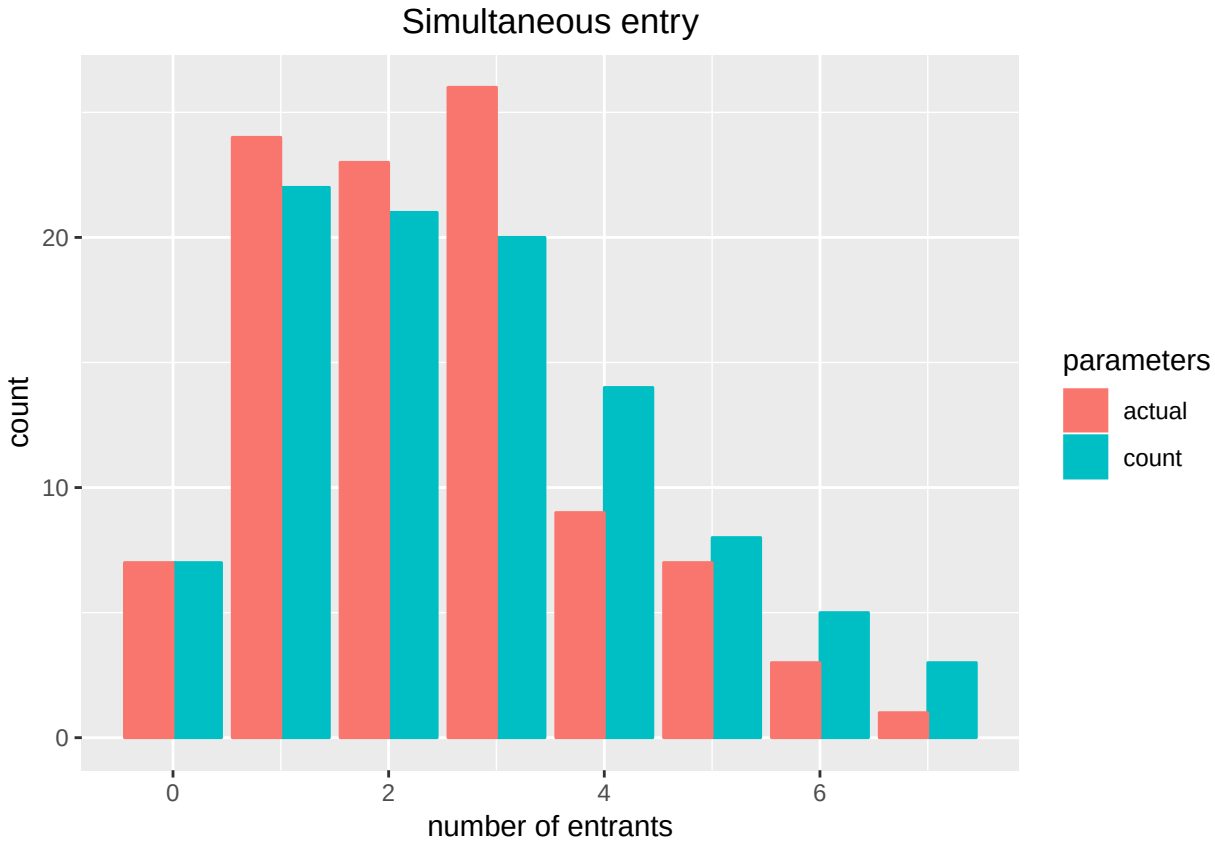
```
comparison <-
  data.frame(
    actual = theta,
    estimate = result_simultaneous$par
  )
comparison
```

```
##      actual  estimate
## 1 0.6264538 0.5787563
## 2 0.1836433 0.1216856
## 3 0.8356286 0.9521359
## 4 1.5952808 1.8443535
## 5 1.0000000 1.2177670
## 6 0.3295078 0.3075741
```

15.3 Conduct counterfactual simulations

1. Fix the first draw of the Monte Carlo shocks. Suppose that the competitive effect becomes mild, i.e. δ is changed to 0.5. Under these shocks, compute the equilibrium number of entrants across markets and plot the histogram with the estimated and counterfactual parameters. Conduct this analysis under the assumptions of sequential and simultaneous entry.




Simultaneous entry
count
20
10
0
0
2
4
6
number of entrants
parameters
actual
count

Chapter 16

Assignment 7: Dynamic Decision

The deadline is **April 29 1:30pm**.

16.1 Simulate data

Suppose that there is a firm and it makes decisions for $t = 1, \cdots, \infty$. We solve the model under the infinite-horizon assumption, but generate data only for $t = 1, \cdots, T$. There are $L = 5$ state $s \in \{1, 2, 3, 4, 5\}$ states for the player. The firm can choose $K + 1 = 2$ actions $a \in \{0, 1\}$.

The mean period payoff to the firm is:

$$\pi(a, s) := \alpha \ln s - \beta a,$$

where $\alpha, \beta > 0$. The period payoff is:

$$\pi(a, s) + \epsilon(a),$$

and $\epsilon(a)$ is an i.i.d. type-I extreme random variable that is independent of all the other variables.

At the beginning of each period, the state s and choice-specific shocks $\epsilon(a), a = 0, 1$ are realized, and the the firm chooses her action. Then, the game moves to the next period.

Suppose that $s > 1$ and $s < L$. If $a = 0$, the state stays at the same state with probability $1 - \kappa$ and moves down by 1 with probability κ. If $a = 1$, the state moves up by 1 with probability γ, moves down by 1 with probability κ, and stays at the same with probability $1 - \kappa - \gamma$.

Suppose that $s = 1$. If $a = 0$, the state stays at the same state with probability 1. If $a = 1$, the state moves up by 1 with probability γ and stays at the same with probability $1 - \gamma$.

Suppose that $s = L$. If $a = 0$, the state stays at the same state with probability $1 - \kappa$ and moves down by 1 with probability κ. If $a = 1$, the state moves down by 1 with probability κ, and stays at the same with probability $1 - \kappa$.

The mean period profit is summarized in Π as:

$$\Pi := \begin{pmatrix} \pi(0,1) \\ \vdots \\ \pi(K,1) \\ \vdots \\ \pi(0,L) \\ \vdots \\ \pi(K,L) \end{pmatrix}$$

The transition law is summarized in G as:

$$g(a, s, s') := \mathbb{P}\{s_{t+1} = s' | s_t = s, a_t = a\},$$

$$G := \begin{pmatrix} g(0,1,1) & \cdots & g(0,1,L) \\ \vdots & & \vdots \\ g(K,1,1) & \cdots & g(K,1,L) \\ & \vdots & \\ g(0,L,1) & \cdots & g(0,L,L) \\ \vdots & & \vdots \\ g(K,L,1) & \cdots & g(K,L,L) \end{pmatrix}.$$

The discount factor is denoted by δ. We simulate data for N firms for T periods each.

1. Set constants and parameters as follows:

```
# set seed
set.seed(1)
# set constants
L <- 5
K <- 1
T <- 100
N <- 1000
lambda <- 1e-10
# set parameters
alpha <- 0.5
beta <- 3
kappa <- 0.1
gamma <- 0.6
delta <- 0.95
```

2. Write function `compute_pi(alpha, beta, L, K)` that computes Π given parameters and compute the true Π under the true parameters. Don't use methods in `dplyr` and deal with matrix operations.

```
PI <- compute_PI(alpha, beta, L, K); PI
```

```
##             [,1]
## k0_l1  0.0000000
## k1_l1 -3.0000000
## k0_l2  0.3465736
## k1_l2 -2.6534264
## k0_l3  0.5493061
## k1_l3 -2.4506939
## k0_l4  0.6931472
## k1_l4 -2.3068528
## k0_l5  0.8047190
## k1_l5 -2.1952810
```

3. Write function `compute_G(kappa, gamma, L, K)` that computes G given parameters and compute the true G under the true parameters. Don't use methods in `dplyr` and deal with matrix operations.

```
G <- compute_G(kappa, gamma, L, K); G
```

```
##        l1  l2  l3  l4  l5
## k0_l1 1.0 0.0 0.0 0.0 0.0
```

```
## k1_l1 0.4 0.6 0.0 0.0 0.0
## k0_l2 0.1 0.9 0.0 0.0 0.0
## k1_l2 0.1 0.3 0.6 0.0 0.0
## k0_l3 0.0 0.1 0.9 0.0 0.0
## k1_l3 0.0 0.1 0.3 0.6 0.0
## k0_l4 0.0 0.0 0.1 0.9 0.0
## k1_l4 0.0 0.0 0.1 0.3 0.6
## k0_l5 0.0 0.0 0.0 0.1 0.9
## k1_l5 0.0 0.0 0.0 0.1 0.9
```

The exante-value function is written as a function of a conditional choice probability as follows:

$$\varphi^{(\theta_1,\theta_2)}(p) := [I - \delta\Sigma(p)G]^{-1}\Sigma(p)[\Pi + E(p)],$$

where $\theta_1 = (\alpha, \beta)$ and $\theta_2 = (\kappa, \gamma)$ and:

$$\Sigma(p) = \begin{pmatrix} p(1)' & & \\ & \ddots & \\ & & p(L)' \end{pmatrix}$$

and:

$$E(p) = \gamma - \ln p.$$

3. Write a function `compute_exante_value(p, PI, G, L, K, delta)` that returns the exante value function given a conditional choice probability. Don't use methods in `dplyr` and deal with matrix operations. When a choice probability is zero at some element, the corresponding element of $E(p)$ can be set at zero, because anyway we multiply the zero probability to the element and the corresponding element in $E(p)$ does not affect the result.

```
p <- matrix(rep(0.5, L * (K + 1)), ncol = 1); p
```

```
##       [,1]
##  [1,]  0.5
##  [2,]  0.5
##  [3,]  0.5
##  [4,]  0.5
##  [5,]  0.5
##  [6,]  0.5
##  [7,]  0.5
##  [8,]  0.5
##  [9,]  0.5
## [10,]  0.5
```

```
V <- compute_exante_value(p, PI, G, L, K, delta); V
```

```
##         [,1]
## l1  5.777876
## l2  7.597282
## l3  9.126304
## l4 10.115439
## l5 10.593438
```

The optimal conditional choice probability is written as a function of an exante value function as follows:

$$\Lambda^{(\theta_1,\theta_2)}(V)(a,s) := \frac{\pi(a,s) + \delta\sum_{s'} V(s')g(a,s,s')}{\sum_{a'}[\pi(a',s) + \delta\sum_{s'} V(s')g(a',s,s')]},$$

where V is an exante value function.

4. Write a function `compute_ccp(V, PI, G, L, K, delta)` that returns the optimal conditional choice probability given an exante value function. Don't use methods in `dplyr` and deal with matrix operations. To do so, write a function `compute_choice_value(V, PI, G, delta)` that returns the choice-specific value function. Use this for debugging by checking if the results are intuitive.

```
value <- compute_choice_value(V, PI, G, delta); value
```

```
##            [,1]
## k0_l1  5.488982
## k1_l1  3.526044
## k0_l2  7.391148
## k1_l2  5.262691
## k0_l3  9.074038
## k1_l3  6.637845
## k0_l4 10.208846
## k1_l4  7.481306
## k0_l5 10.823075
## k1_l5  7.823075
```

```
p <- compute_ccp(V, PI, G, L, K, delta); p
```

```
##             [,1]
## k0_l1 0.87685057
## k1_l1 0.12314943
## k0_l2 0.89363847
## k1_l2 0.10636153
## k0_l3 0.91954591
## k1_l3 0.08045409
## k0_l4 0.93863232
## k1_l4 0.06136768
## k0_l5 0.95257413
## k1_l5 0.04742587
```

5. Write a function that find the equilibrium conditional choice probability and ex-ante value function by iterating the update of an exante value function and an optimal conditional choice probability. The iteration should stop when $\max_s |V^{(r+1)}(s) - V^{(r)}(s)| < \lambda$ with $\lambda = 10^{-10}$.

```
output <- solve_dynamic_decision(PI, G, L, K, delta, lambda); output
```

```
## $p
##             [,1]
## k0_l1 0.82218962
## k1_l1 0.17781038
## k0_l2 0.80024354
## k1_l2 0.19975646
## k0_l3 0.83074516
## k1_l3 0.16925484
## k0_l4 0.87691534
## k1_l4 0.12308466
## k0_l5 0.95257413
## k1_l5 0.04742587
##
## $V
##        [,1]
## l1 15.46000
## l2 18.03675
## l3 20.86514
```

```
## l4 23.33721
## l5 25.15557
```

```
p <- output$p
V <- output$V
value <- compute_choice_value(V, PI, G, delta); value
```

```
##           [,1]
## k0_l1 14.68700
## k1_l1 13.15574
## k0_l2 17.23669
## k1_l2 15.84887
## k0_l3 20.10249
## k1_l3 18.51157
## k0_l4 22.62865
## k1_l4 20.66511
## k0_l5 24.52976
## k1_l5 21.52976
```

6. Write a function `simulate_dynamic_decision(p, s, PI, G, L, K, T, delta, seed)` that simulate the data for a single firm starting from an initial state for T periods. The function should accept a value of seed and set the seed at the beginning of the procedure inside the function, because the process is stochastic.

```
# set initial value
s <- 1
# draw simulation for a firm
seed <- 1
df <- simulate_dynamic_decision(p, s, PI, G, L, K, T, delta, seed); df
```

```
## # A tibble: 100 x 3
##        t     s     a
##    <int> <dbl> <dbl>
##  1     1     1     0
##  2     2     1     0
##  3     3     1     0
##  4     4     1     1
##  5     5     2     1
##  6     6     1     0
##  7     7     1     0
##  8     8     1     0
##  9     9     1     0
## 10    10     1     0
## # ... with 90 more rows
```

7. Write a function `simulate_dynamic_decision_across_firms(p, s, PI, G, L, K, T, N, delta)` that returns simulation data for N firm. For firm i, set the seed at i

```
df <- simulate_dynamic_decision_across_firms(p, s, PI, G, L, K, T, N, delta)
save(df, file = "data/A7_df.RData")
```

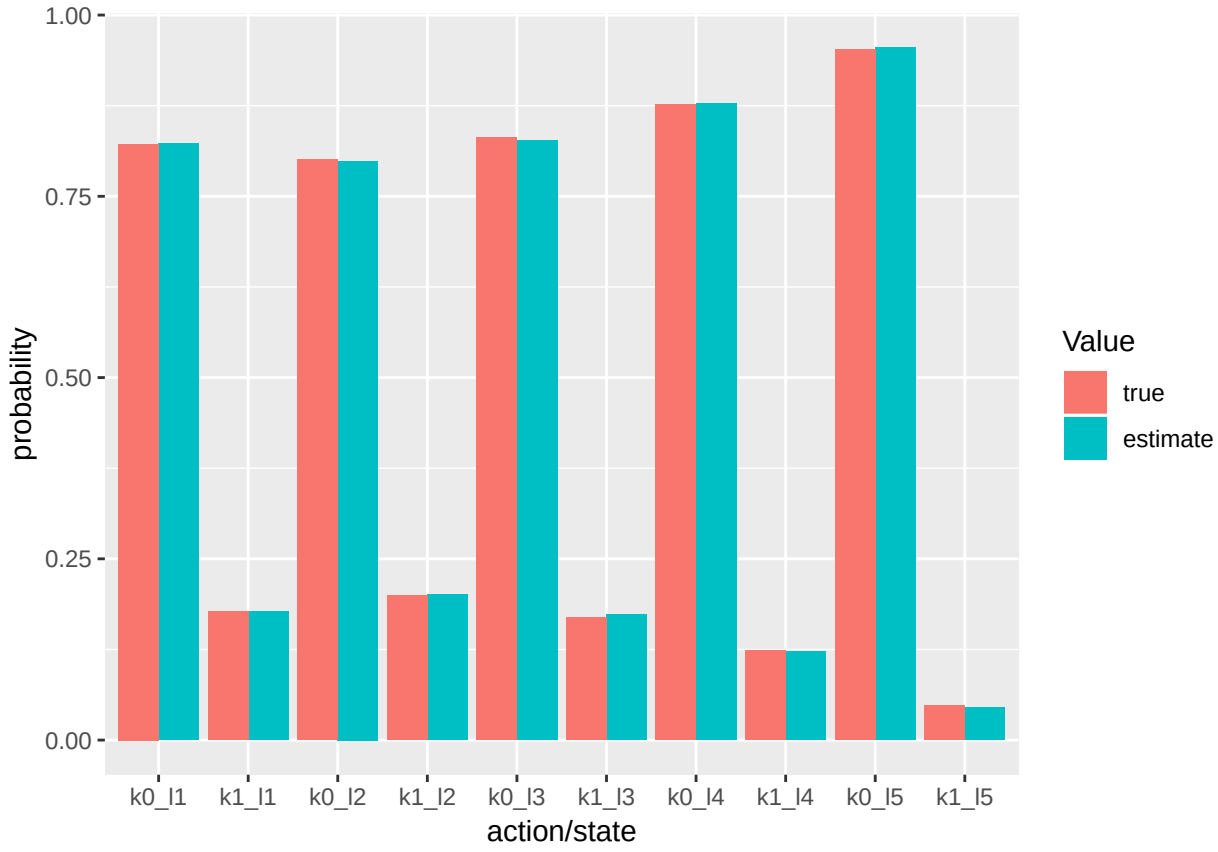
```
load(file = "data/A7_df.RData")
df
```

```
## # A tibble: 100,000 x 4
##        i     t     s     a
##    <int> <int> <dbl> <dbl>
##  1     1     1     1     0
```

```
##  2     1     2     1     0
##  3     1     3     1     0
##  4     1     4     1     1
##  5     1     5     2     1
##  6     1     6     1     0
##  7     1     7     1     0
##  8     1     8     1     0
##  9     1     9     1     0
## 10     1    10     1     0
## # ... with 99,990 more rows
```

8. Write a function `estimate_ccp(df)` that returns a non-parametric estimate of the conditional choice probability in the data. Compare the estimated conditional choice probability and the true conditional choice probability by a bar plot.

```
p_est <- estimate_ccp(df)
check_ccp <- cbind(p, p_est)
colnames(check_ccp) <- c("true", "estimate")
check_ccp <- check_ccp %>%
  reshape2::melt()
ggplot(data = check_ccp, aes(x = Var1, y = value,
                             fill = Var2)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(fill = "Value") + xlab("action/state") + ylab("probability")
```

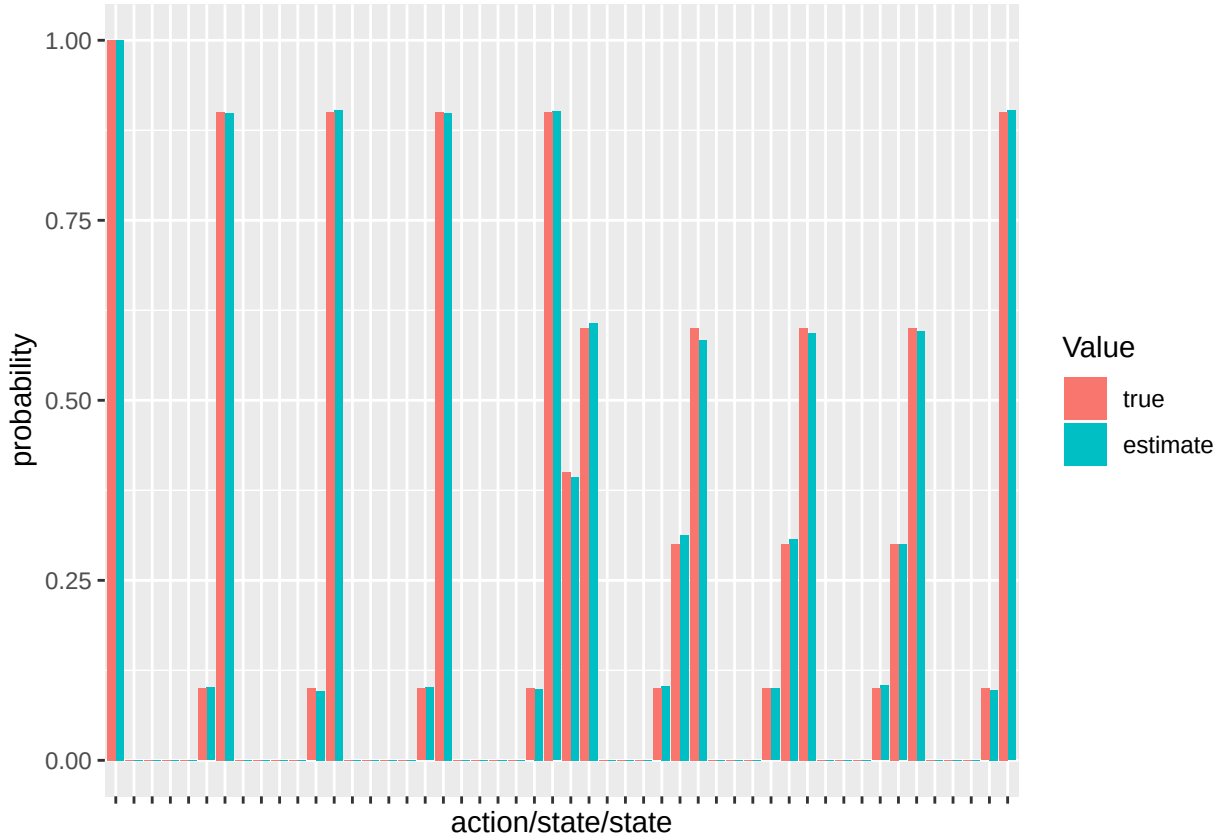


9. Write a function `estimate_G(df)` that returns a non-parametric estiamte of the transition matrix in the data. Compare the estimated transition matrix and the true transition matrix by a bar plot.

```
G_est <- estimate_G(df); G_est
```

```
##              l1         l2        l3         l4        l5
## k0_l1 1.0000000 0.00000000 0.0000000 0.00000000 0.0000000
## k1_l1 0.3930818 0.60691824 0.0000000 0.00000000 0.0000000
## k0_l2 0.1012162 0.89878384 0.0000000 0.00000000 0.0000000
## k1_l2 0.1031410 0.31276454 0.5840945 0.00000000 0.0000000
## k0_l3 0.0000000 0.09660837 0.9033916 0.00000000 0.0000000
## k1_l3 0.0000000 0.09974569 0.3071489 0.59310540 0.0000000
## k0_l4 0.0000000 0.00000000 0.1012564 0.89874358 0.0000000
## k1_l4 0.0000000 0.00000000 0.1039339 0.29966003 0.5964060
## k0_l5 0.0000000 0.00000000 0.0000000 0.09891400 0.9010860
## k1_l5 0.0000000 0.00000000 0.0000000 0.09751037 0.9024896
```

```
check_G <- data.frame(type = "true", reshape2::melt(G))
check_G_est <- data.frame(type = "estimate", reshape2::melt(G_est))
check_G <- rbind(check_G, check_G_est)
check_G$variable = paste(check_G$Var1, check_G$Var2, sep = "_")
ggplot(data = check_G, aes(x = variable, y = value,
                           fill = type)) +
    geom_bar(stat = "identity", position = "dodge") +
  labs(fill = "Value") + xlab("action/state/state") + ylab("probability") +
  theme(axis.text.x = element_blank())
```



16.2 Estimate parameters

1. Vectorize the parameters as follows:

```
theta_1 <- c(alpha, beta)
theta_2 <- c(kappa, gamma)
theta <- c(theta_1, theta_2)
```

First, we estimate the parameters by a nested fixed-point algorithm. The loglikelihood for $\{a_{it}, s_{it}\}_{i=1,\cdots,N,t=1,\cdots,T}$ is:

$$\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}[\log \mathbb{P}\{a_{it}|s_{it}\} + \log \mathbb{P}\{s_{i,t+1}|a_{it}, s_{it}\}],$$

with $\mathbb{P}\{s_{i,T+1}|a_{iT}, s_{iT}\} = 1$ for all i as $s_{i,T+1}$ is not observed.

2. Write a function `compute_loglikelihood_NFP(theta, df, delta, L, K)` that compute the loglikelihood.

```
loglikelihood <- compute_loglikelihood_NFP(theta, df, delta, L, K); loglikelihood
```

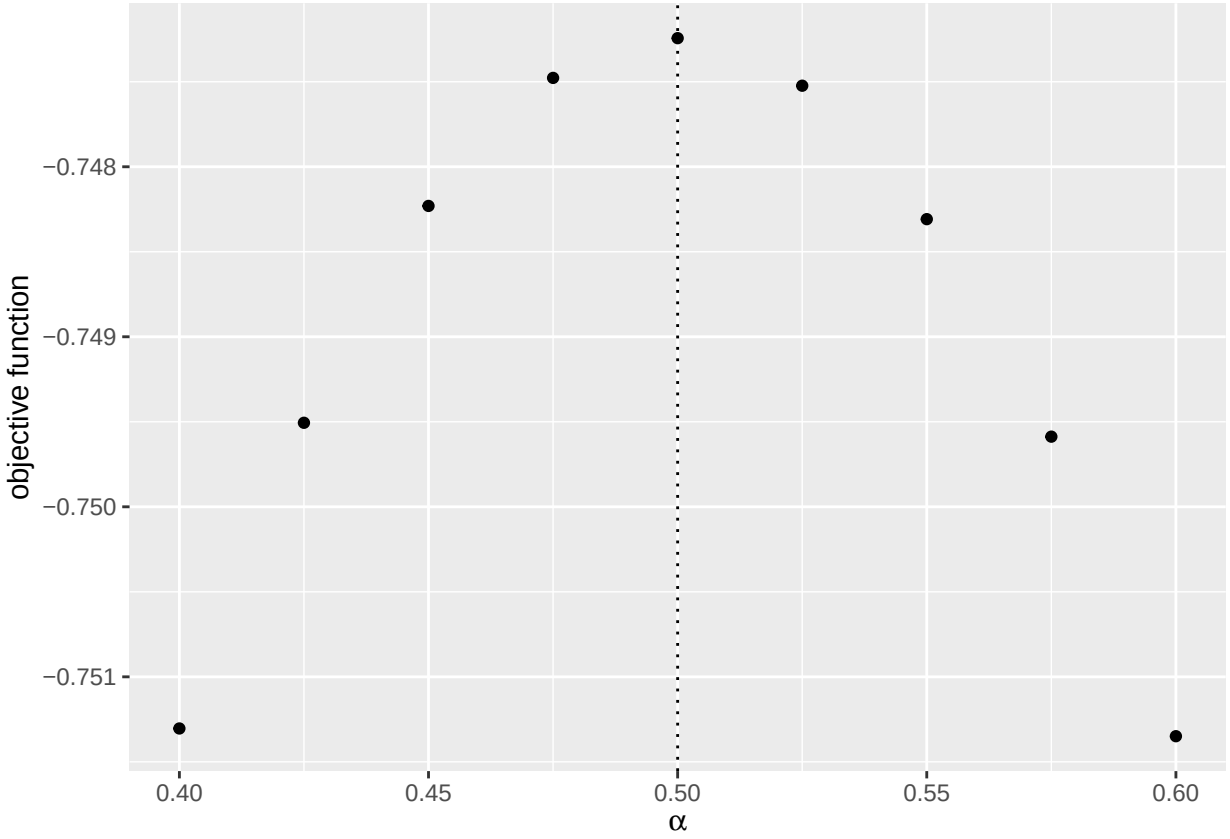
```
## [1] -0.7474961
```

3. Check the value of the objective function around the true parameter.

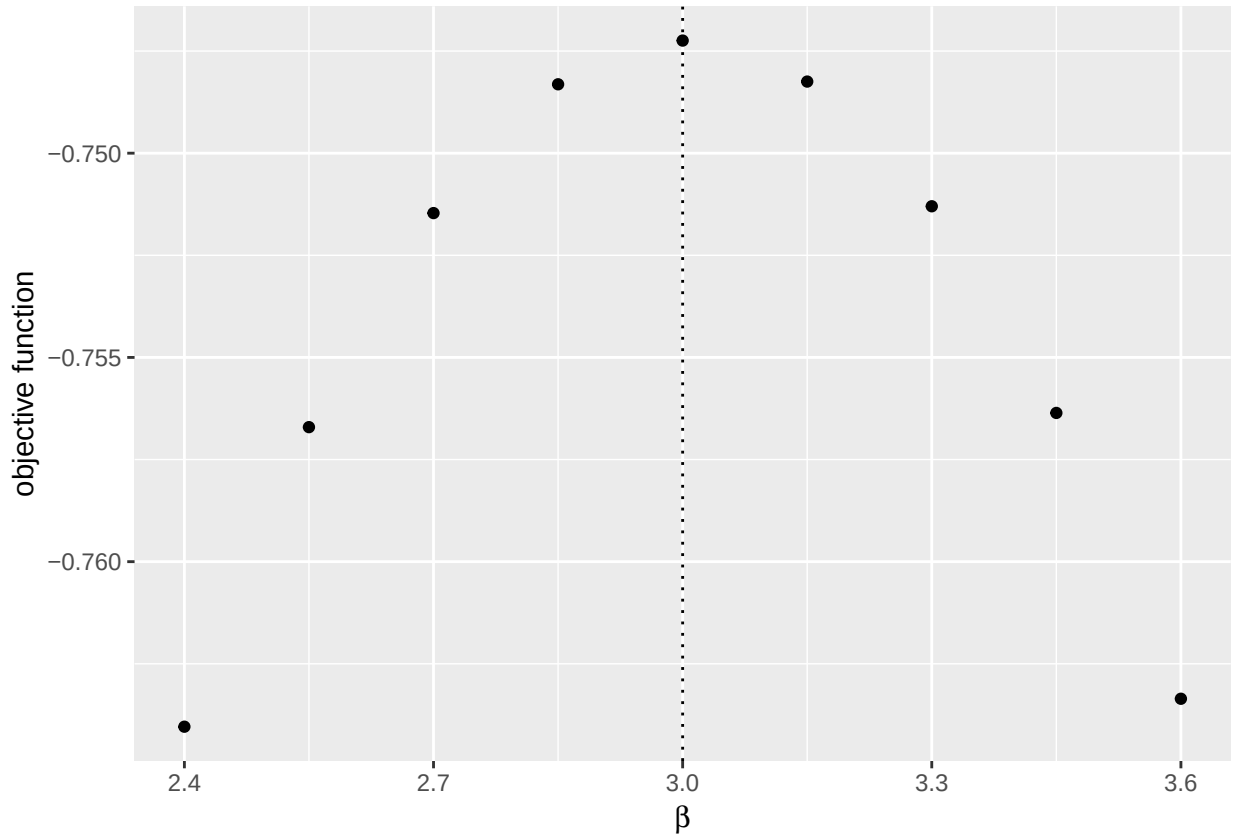
```
# label
label <- c("\\alpha", "\\beta", "\\kappa", "\\gamma")
label <- paste("$", label, "$", sep = "")
# compute the graph
graph <- foreach (i = 1:length(theta)) %do% {
  theta_i <- theta[i]
  theta_i_list <- theta_i * seq(0.8, 1.2, by = 0.05)
  objective_i <-
    foreach (j = 1:length(theta_i_list),
             .combine = "rbind") %do% {
               theta_ij <- theta_i_list[j]
               theta_j <- theta
               theta_j[i] <- theta_ij
               objective_ij <-
                 compute_loglikelihood_NFP(
                   theta_j, df, delta, L, K); loglikelihood

               return(objective_ij)
             }
  df_graph <- data.frame(x = theta_i_list, y = objective_i)
  g <- ggplot(data = df_graph, aes(x = x, y = y)) +
    geom_point() +
    geom_vline(xintercept = theta_i, linetype = "dotted") +
    ylab("objective function") + xlab(TeX(label[i]))
  return(g)
}
save(graph, file = "data/A7_NFP_graph.RData")
```

```
load(file = "data/A7_NFP_graph.RData")
graph
```

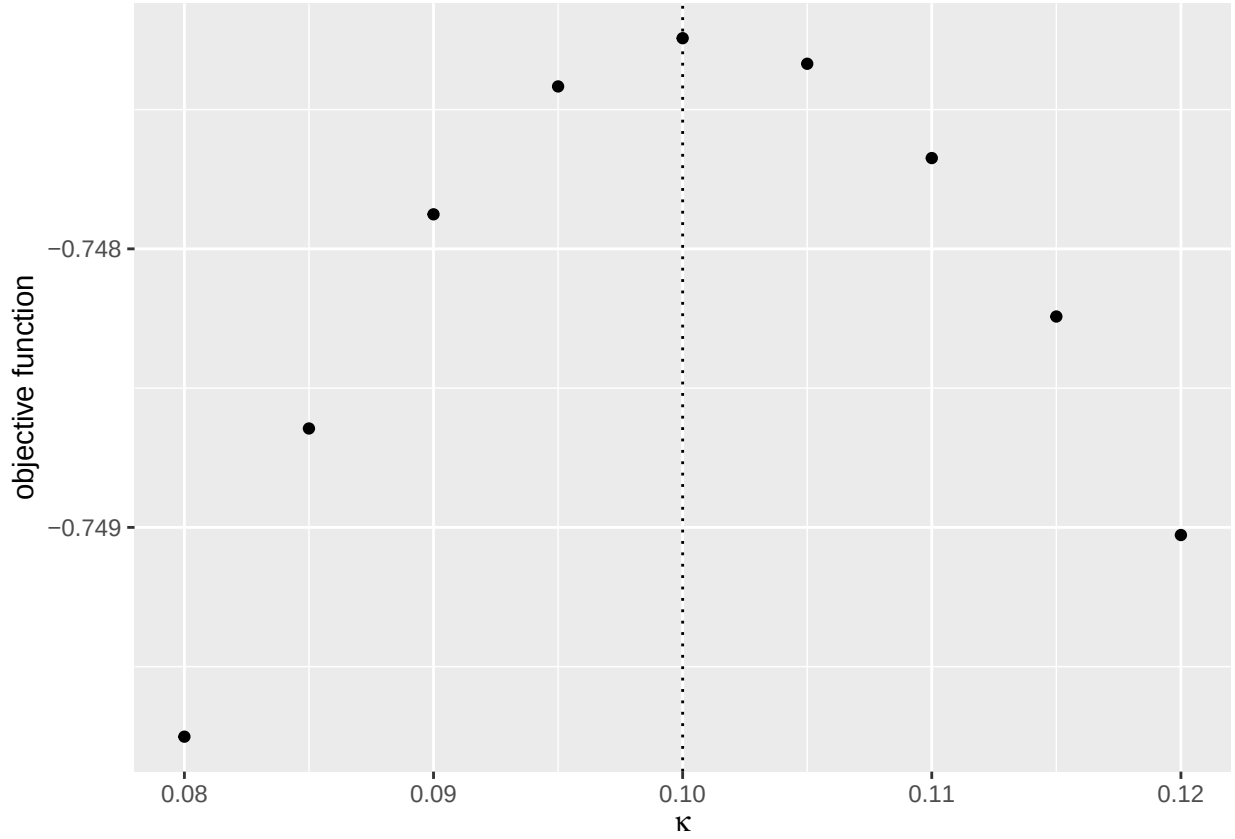
```
## [[1]]
```



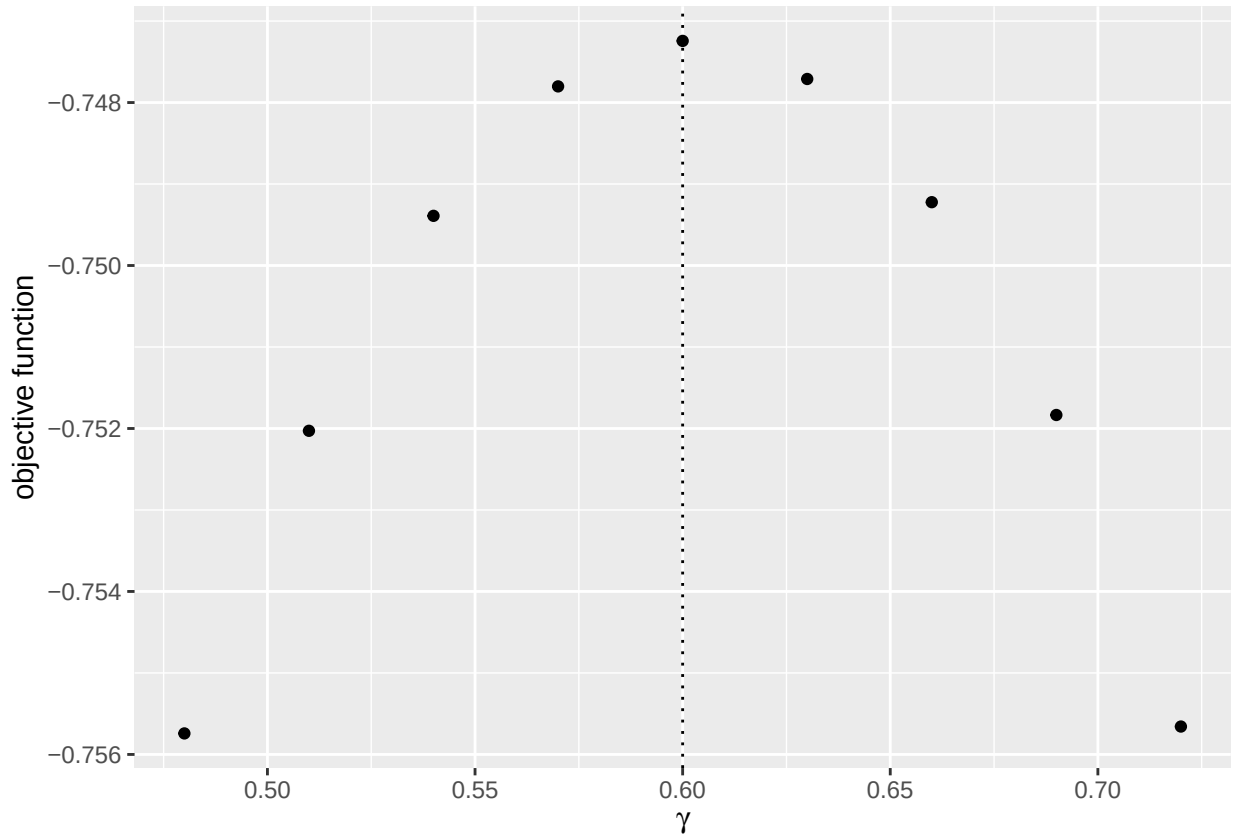

```
##
## [[2]]
```




```
##
## [[3]]
```




```
##
## [[4]]
```



4. Estiamte the parameters by maximizing the loglikelihood. To keep the model to be well-defined, impose an ad hoc lower and upper bounds such that $\alpha \in [0, 1], \beta \in [0, 5], \kappa \in [0, 0.2], \gamma \in [0, 0.7]$.

```
lower <- rep(0, length(theta))
upper <- c(1, 5, 0.2, 0.7)
NFP_result <-
  optim(par = theta,
        fn = compute_loglikelihood_NFP,
        method = "L-BFGS-B",
        lower = lower,
        upper = upper,
        control = list(fnscale = -1),
        df = df,
        delta = delta,
        L = L,
        K = K)
save(NFP_result, file = "data/A7_NFP_result.RData")
```

```
load(file = "data/A7_NFP_result.RData")
NFP_result
```

```
## $par
## [1] 0.4916153 2.9816751 0.1005993 0.6029317
##
## $value
## [1] -0.747237
##
## $counts
```

```
## function gradient
##       17       17
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

```
compare <-
  data.frame(
    true = theta,
    estimate = NFP_result$par
  ); compare
```

```
##   true  estimate
## 1  0.5 0.4916153
## 2  3.0 2.9816751
## 3  0.1 0.1005993
## 4  0.6 0.6029317
```

Next, we estimate the parameters by CCP approach.

5. Write a function `estimate_theta_2(df)` that returns the estimates of κ and γ directly from data by counting relevant events.

```
theta_2_est <- estimate_theta_2(df); theta_2_est
```

```
## [1] 0.09988488 0.59551895
```

The objective function of the minimum distance estimator based on the conditional choice probability approach is:

$$\frac{1}{KL}\sum_{s=1}^{L}\sum_{a=1}^{K}\{\hat{p}(a,s) - p^{(\theta_1,\theta_2)}(a,s)\}^2,$$

where $\hat{p}$ is the non-parametric estimate of the conditional choice probability and $p^{(\theta_1,\theta_2)}$ is the optimal conditional choice probability under parameters θ_1 and θ_2.

6. Write a function `compute_CCP_objective(theta_1, theta_2, p_est, L, K, delta)` that returns the objective function of the above minimum distance estimator given a non-parametric estimate of the conditional choice probability and θ_1 and θ_2.

```
compute_CCP_objective(theta_1, theta_2, p_est, L, K, delta)
```

```
## [1] 5.000511e-06
```

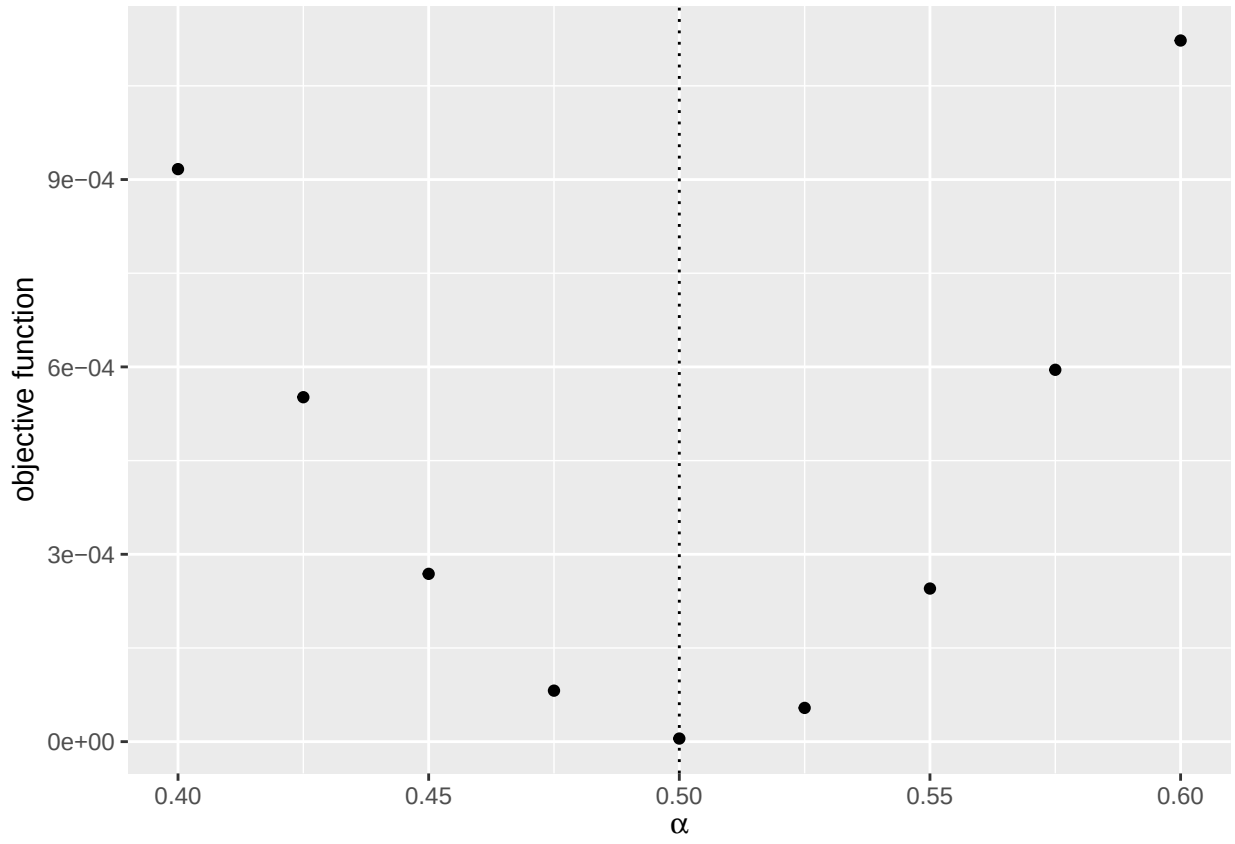
3. Check the value of the objective function around the true parameter.

```
# label
label <- c("\\alpha", "\\beta")
label <- paste("$", label, "$", sep = "")
# compute the graph
graph <- foreach (i = 1:length(theta_1)) %do% {
  theta_i <- theta_1[i]
  theta_i_list <- theta_i * seq(0.8, 1.2, by = 0.05)
  objective_i <-
    foreach (j = 1:length(theta_i_list),
             .combine = "rbind") %do% {
               theta_ij <- theta_i_list[j]
```

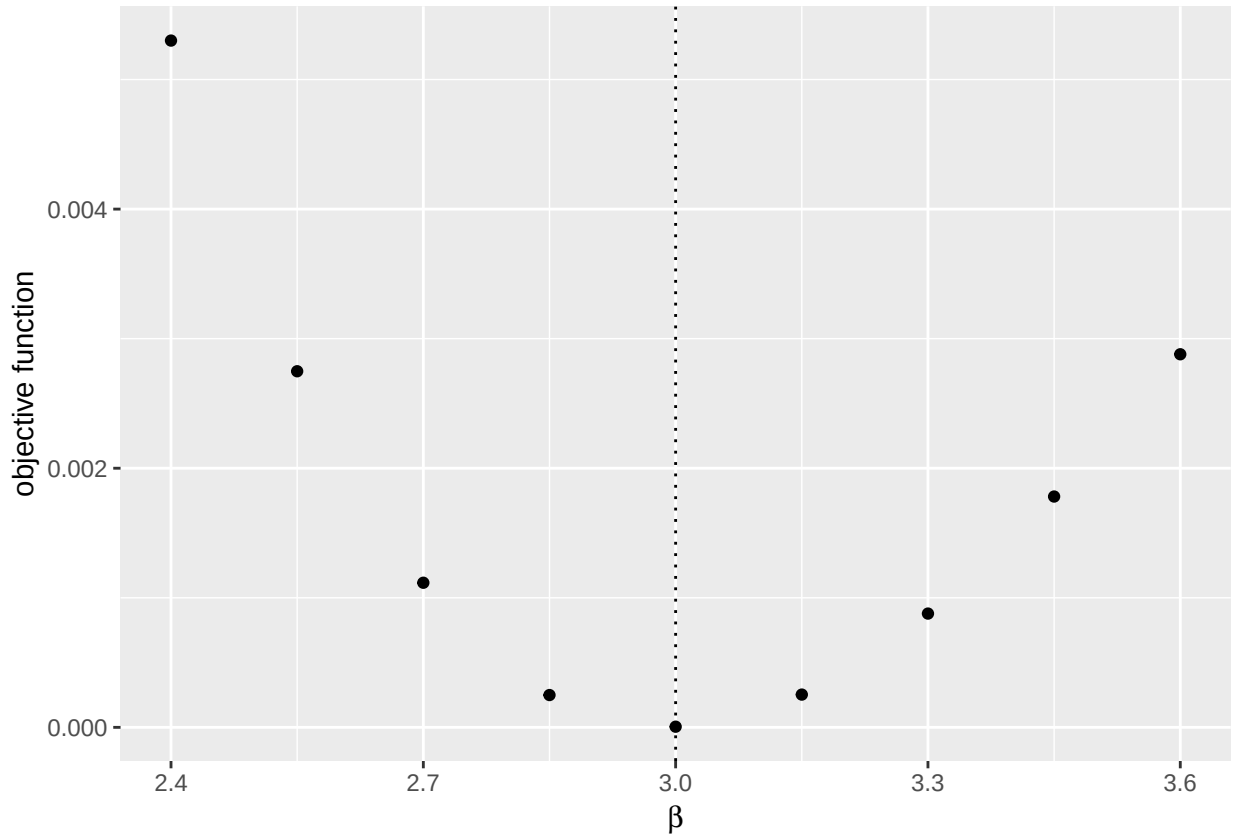
```
              theta_j <- theta_1
              theta_j[i] <- theta_ij
              objective_ij <-
                compute_CCP_objective(theta_j, theta_2, p_est, L, K, delta)
              return(objective_ij)
            }
  df_graph <- data.frame(x = theta_i_list, y = objective_i)
  g <- ggplot(data = df_graph, aes(x = x, y = y)) +
    geom_point() +
    geom_vline(xintercept = theta_i, linetype = "dotted") +
    ylab("objective function") + xlab(TeX(label[i]))
  return(g)
}
save(graph, file = "data/A7_CCP_graph.RData")
```

```
load(file = "data/A7_CCP_graph.RData")
graph
```

```
## [[1]]
```




```
##
## [[2]]
```



4. Estiamte the parameters by minimizing the objective function. To keep the model to be well-defined, impose an ad hoc lower and upper bounds such that $\alpha \in [0, 1], \beta \in [0, 5]$.

```
lower <- rep(0, length(theta_1))
upper <- c(1, 5)
CCP_result <-
  optim(par = theta_1,
        fn = compute_CCP_objective,
        method = "L-BFGS-B",
        lower = lower,
        upper = upper,
        theta_2 = theta_2_est,
        p_est = p_est,
        L = L,
        K = K,
        delta = delta)
save(CCP_result, file = "data/A7_CCP_result.RData")
```

```
load(file = "data/A7_CCP_result.RData")
CCP_result
```

```
## $par
## [1] 0.5271684 3.0644600
##
## $value
## [1] 1.790528e-06
##
## $counts
```

```
## function gradient
##       11       11
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

```
compare <-
  data.frame(
    true = theta_1,
    estimate = CCP_result$par
  ); compare
```

```
##   true  estimate
## 1  0.5 0.5271684
## 2  3.0 3.0644600
```

Chapter 17

Assignment 8: Dynamic Game

The deadline is **May 6 1:30pm**.

17.1 Simulate data

Suppose that there are $m = 1, \cdots, M$ markets and in each market there are $i = 1, \cdots, N$ firms and each firm makes decisions for $t = 1, \cdots, \infty$. In the following, I suppress the index of market, m. We solve the model under the infinite-horizon assumption, but generate data only for $t = 1, \cdots, T$. There are $L = 3$ state $\{1, 2, 3\}$ states for each each. Each firm can choose $K + 1 = 2$ actions $\{0, 1\}$. Thus, $m_a := (K + 1)^N$ and $m_s = L^N$. Let a_i and s_i be firm i's action and state and a and s are vectors of individual actions and states.

The mean period payoff to firm i is:

$$\pi_i(a, s) := \tilde{\pi}(a_i, s_i, \overline{s}) := \alpha \ln s_i - \eta \ln s_i \sum_{j \neq i} \ln s_j - \beta a_i,$$

where $\alpha, \beta, \eta > 0$, and $\alpha > \eta$. The term η means that the returns to investment decreases as rival's average state profile improves. The period payoff is:

$$\tilde{\pi}(a_i, s_i, \overline{s}) + \epsilon_i(a_i),$$

and $\epsilon_i(a_i)$ is an i.i.d. type-I extreme random variable that is independent of all the other variables.

At the beginning of each period, the state s is realized and publicly observed. Then choice-specific shocks $\epsilon_i(a_i), a_i = 0, 1$ are realized and privately observed by firm $i = 1, \cdots, N$. Then each firm simultaneously chooses her action. Then, the game moves to next period.

State transition is independent across firms conditional on individual state and action.

Suppose that $s_i > 1$ and $s_i < L$. If $a_i = 0$, the state stays at the same state with probability $1 - \kappa$ and moves down by 1 with probability κ. If $a = 1$, the state moves up by 1 with probability γ, moves down by 1 with probability κ, and stays at the same with probability $1 - \kappa - \gamma$.

Suppose that $s_i = 1$. If $a_i = 0$, the state stays at the same state with probability 1. If $a_i = 1$, the state moves up by 1 with probability γ and stays at the same with probability $1 - \gamma$.

Suppose that $s_i = L$. If $a_i = 0$, the state stays at the same state with probability $1 - \kappa$ and moves down by 1 with probability κ. If $a = 1$, the state moves down by 1 with probability κ, and stays at the same with probability $1 - \kappa$.

The mean period profit is summarized in Π as:

$$\Pi := \begin{pmatrix} \pi(1,1) \\ \vdots \\ \pi(m_a,1) \\ \vdots \\ \pi(1,m_s) \\ \vdots \\ \pi(m_a,m_s) \end{pmatrix}$$

The transition law is summarized in G as:

$$g(a,s,s') := \mathbb{P}\{s_{t+1} = s' | s_t = s, a_t = a\},$$

$$G := \begin{pmatrix} g(1,1,1) & \cdots & g(1,1,m_s) \\ \vdots & & \vdots \\ g(m_a,1,1) & \cdots & g(m_a,1,m_s) \\ & \vdots & \\ g(1,m_s,1) & \cdots & g(1,m_s,m_s) \\ \vdots & & \vdots \\ g(m_a,m_s,1) & \cdots & g(m_a,m_s,m_s) \end{pmatrix}.$$

The discount factor is denoted by δ. We simulate data for M markets with N firms for T periods.

1. Set constants and parameters as follows:

```
# set seed
set.seed(1)
# set constants
L <- 5
K <- 1
T <- 100
N <- 3
M <- 1000
lambda <- 1e-10
# set parameters
alpha <- 1
eta <- 0.3
beta <- 2
kappa <- 0.1
gamma <- 0.6
delta <- 0.95
```

2. Write a function `compute_action_state_space(K, L, N)` that returns a data frame for action and state space. Returned objects are list of data frame `A` and `S`. In `A`, column `k` is the index of an action profile, `i` is the index of a firm, and `a` is the action of the firm. In `S`, column `l` is the index of an state profile, `i` is the index of a firm, and `s` is the state of the firm.

```
output <- compute_action_state_space(L, K, N)
A <- output$A
head(A)
```

```
## # A tibble: 6 x 3
##       k     i     a
```

```
##   <int> <int> <int>
## 1     1     1     0
## 2     1     2     0
## 3     1     3     0
## 4     2     1     1
## 5     2     2     0
## 6     2     3     0
```

```
tail(A)
```

```
## # A tibble: 6 x 3
##       k     i     a
##   <int> <int> <int>
## 1     7     1     0
## 2     7     2     1
## 3     7     3     1
## 4     8     1     1
## 5     8     2     1
## 6     8     3     1
```

```
S <- output$S
head(S)
```

```
## # A tibble: 6 x 3
##       l     i     s
##   <int> <int> <int>
## 1     1     1     1
## 2     1     2     1
## 3     1     3     1
## 4     2     1     2
## 5     2     2     1
## 6     2     3     1
```

```
tail(S)
```

```
## # A tibble: 6 x 3
##       l     i     s
##   <int> <int> <int>
## 1   124     1     4
## 2   124     2     5
## 3   124     3     5
## 4   125     1     5
## 5   125     2     5
## 6   125     3     5
```

```
# dimension
m_a <- max(A$k); m_a
```

```
## [1] 8
```

```
m_s <- max(S$l); m_s
```

```
## [1] 125
```

3. Write function `compute_PI_game(alpha, beta, eta, L, K, N)` that returns a list of Π_i.

```
PI <- compute_PI_game(alpha, beta, eta, A, S)
head(PI[[N]])
```

```
##      [,1]
## [1,]    0
## [2,]    0
## [3,]    0
## [4,]    0
## [5,]   -2
## [6,]   -2
```

```
dim(PI[[N]])[1] == m_s * m_a
```

```
## [1] TRUE
```

4. Write function `compute_G_game(g, A, S)` that converts an individual transition probability matrix into a joint transition probability matrix G.

```
G_marginal <- compute_G(kappa, gamma, L, K)
G <- compute_G_game(G_marginal, A, S)
head(G)
```

```
##         1    2 3 4 5    6    7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
## [1,] 1.00 0.00 0 0 0 0.00 0.00 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [2,] 0.40 0.60 0 0 0 0.00 0.00 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [3,] 0.40 0.00 0 0 0 0.60 0.00 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [4,] 0.16 0.24 0 0 0 0.24 0.36 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [5,] 0.40 0.00 0 0 0 0.00 0.00 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [6,] 0.16 0.24 0 0 0 0.00 0.00 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0
##      23 24 25   26   27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
## [1,]  0  0  0 0.00 0.00  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [2,]  0  0  0 0.00 0.00  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [3,]  0  0  0 0.00 0.00  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [4,]  0  0  0 0.00 0.00  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [5,]  0  0  0 0.60 0.00  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [6,]  0  0  0 0.24 0.36  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##      45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67
## [1,]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [2,]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [3,]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [4,]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [5,]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [6,]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##      68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
## [1,]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [2,]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [3,]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [4,]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [5,]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [6,]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##      91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109
## [1,]  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0
## [2,]  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0
## [3,]  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0
## [4,]  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0
## [5,]  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0
## [6,]  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0
##      110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
## [1,]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
## [2,]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## [3,]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## [4,]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## [5,]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## [6,]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
dim(G)[1] == m_s * m_a
```

```
## [1] TRUE
```

```
dim(G)[2] == m_s
```

```
## [1] TRUE
```

The ex-ante-value function for a firm is written as a function of a conditional choice probability as follows:

$$\varphi_i^{(\theta_1,\theta_2)}(p) := [I - \delta\Sigma_i(p)G]^{-1}[\Sigma_i(p)\Pi_i + D_i(p)],$$

where $\theta_1 = (\alpha, \beta, \eta)$ and $\theta_2 = (\kappa, \gamma)$, $p_i(a_i|s)$ is the probability that firm i choose action a_i when the state profile is s, and:

$$p(a|s) = \prod_{i=1}^{N} p_i(a_i|s),$$

$$p(s) = \begin{pmatrix} p(1|s) \\ \vdots \\ p(m_a|s) \end{pmatrix},$$

$$p = \begin{pmatrix} p(1) \\ \vdots \\ p(m_s) \end{pmatrix},$$

$$\Sigma(p) = \begin{pmatrix} p(1)' & & \\ & \ddots & \\ & & p(L)' \end{pmatrix}$$

and:

$$D_i(p) = \begin{pmatrix} \sum_{k=0}^{K} \mathbb{E}\{\epsilon_i^k | a_i = k, 1\} p_i(a_i = k|1) \\ \vdots \\ \sum_{k=0}^{K} \mathbb{E}\{\epsilon_i^k | a_i = k, m_s\} p_i(a_i = k|m_s) \end{pmatrix}.$$

5. Write a function `initialize_p_marginal(A, S)` that defines an initial marginal condition choice probability. In the output `p_marginal`, `p` is the probability for firm `i` to take action `a` conditional on the state profile being `l`. Next, write a function `compute_p_joint(p_marginal, A, S)` that computes a corresponding joint conditional choice probability from a marginal conditional choice probability. In the output `p_joint`, `p` is the joint probability that firms take action profile `k` condition on the state profile being `l`. Finally, write a function `compute_p_marginal(p_joint, A, S)` that compute a corresponding marginal conditional choice probability from a joint conditional choice probability.

```
# define a conditional choice probability for each firm
p_marginal <- initialize_p_marginal(A, S)
p_marginal
```

```
## # A tibble: 750 x 4
##        i     l     a     p
##    <int> <int> <int> <dbl>
##  1     1     1     0   0.5
##  2     1     1     1   0.5
##  3     1     2     0   0.5
##  4     1     2     1   0.5
##  5     1     3     0   0.5
##  6     1     3     1   0.5
##  7     1     4     0   0.5
##  8     1     4     1   0.5
##  9     1     5     0   0.5
## 10     1     5     1   0.5
## # ... with 740 more rows
```

```
dim(p_marginal)[1] == N * m_s * (K + 1)
```

```
## [1] TRUE
```

```
# compute joint conitional choice probability from marginal probability
p_joint <- compute_p_joint(p_marginal, A, S)
p_joint
```

```
## # A tibble: 1,000 x 3
##        l     k     p
##    <int> <int> <dbl>
##  1     1     1 0.125
##  2     1     2 0.125
##  3     1     3 0.125
##  4     1     4 0.125
##  5     1     5 0.125
##  6     1     6 0.125
##  7     1     7 0.125
##  8     1     8 0.125
##  9     2     1 0.125
## 10     2     2 0.125
## # ... with 990 more rows
```

```
dim(p_joint)[1] == m_s * m_a
```

```
## [1] TRUE
```

```
# compute marginal conditional chocie probability from joint probability
p_marginal_2 <- compute_p_marginal(p_joint, A, S)
max(abs(p_marginal - p_marginal_2))
```

```
## [1] 0
```

6. Write a function `compute_Sigma(p_marginal, A, S)` that computes $\Sigma(p)$ given a joint conditional choice probability. Then, write a function `compute_D(p_marginal)` that returns a list of $D_i(p)$.

```
# compute Sigma for ex-ante value function calculation
Sigma <- compute_Sigma(p_marginal, A, S)
head(Sigma)
```

```
## [1] 0.125 0.000 0.000 0.000 0.000 0.000
```

```
dim(Sigma)[1] == m_s
```

```
## [1] TRUE
```

```
dim(Sigma)[2] == m_s * m_a
```

```
## [1] TRUE
```

```
# compute D for ex-ante value function calculation
D <- compute_D(p_marginal)
head(D[[N]])
```

```
##          [,1]
## [1,] 1.270363
## [2,] 1.270363
## [3,] 1.270363
## [4,] 1.270363
## [5,] 1.270363
## [6,] 1.270363
```

```
dim(D[[N]])[1] == m_s
```

```
## [1] TRUE
```

7. Write a function `compute_exante_value_game(p_marginal, A, S, PI, G, delta)` that returns a list of matrices whose i-th element represents the ex-ante value function given a conditional choice probability for firm i.

```
# compute ex-ante value funciton for each firm
V <- compute_exante_value_game(p_marginal, A, S, PI, G, delta)
head(V[[N]])
```

```
## [1] 10.786330 10.175982  9.606812  9.255459  9.115332 10.175982
```

```
dim(V[[N]])[1] == m_s
```

```
## [1] TRUE
```

The optimal conditional choice probability is written as a function of an ex-ante value function and a conditional choice probability of others as follows:

$$\Lambda_i^{(\theta_1,\theta_2)}(V_i, p_{-i})(a_i, s) := \frac{\sum_{a_{-i}} p_{-i}(a_{-i}|s)[\pi_i(a_i, a_{-i}, s) + \delta \sum_{s'} V_i(s') g(a_i, a_{-i}, s, s')]}{\sum_{a_i'} \{\sum_{a_{-i}} p_{-i}(a_{-i}|s)[\pi_i(a_i', a_{-i}, s) + \delta \sum_{s'} V_i(s') g(a_i', a_{-i}, s, s')]\}},$$

where V is an ex-ante value function.

8. Write a function `compute_profile_value_game(V, PI, G, delta, S, A)` that returns a data frame that contains information on value function at a state and action profile for each firm. In the output `value`, `i` is the index of a firm, `l` is the index of a state profile, `k` is the index of an action profile, and `value` is the value for the firm at the state and action profile.

```
# compute state-action-profile value function
value <- compute_profile_value_game(V, PI, G, delta, S, A)
value
```

```
## # A tibble: 3,000 x 4
##        i     l     k value
##    <int> <int> <int> <dbl>
##  1     1     1     1 10.2
##  2     1     1     2  9.63
##  3     1     1     3  9.90
##  4     1     1     4  9.13
##  5     1     1     5  9.90
```

```
##  6     1     1     6  9.13
##  7     1     1     7  9.55
##  8     1     1     8  8.64
##  9     1     2     1 13.0
## 10     1     2     2 12.1
## # ... with 2,990 more rows
```

```
dim(value)[1] == N * m_s * m_a
```

```
## [1] TRUE
```

9. Write a function `compute_choice_value_game(p_marginal, V, PI, G, delta, A, S)` that computes a data frame that contains information on a choice-specific value function given an ex-ante value function and a conditional choice probability of others.

```
# compute choice-specific value function
value <- compute_choice_value_game(p_marginal, V, PI, G, delta, A, S)
value
```

```
## # A tibble: 750 x 4
##        i     l     a value
##    <int> <int> <int> <dbl>
##  1     1     1     0  9.90
##  2     1     1     1  9.13
##  3     1     2     0 12.4
##  4     1     2     1 11.4
##  5     1     3     0 14.5
##  6     1     3     1 13.2
##  7     1     4     0 16.0
##  8     1     4     1 14.3
##  9     1     5     0 16.8
## 10     1     5     1 14.8
## # ... with 740 more rows
```

10. Write a function `compute_ccp_game(p_marginal, V, PI, G, delta, A, S)` that computes a data frame that contains information on a conditional choice probability given an ex-ante value function and a conditional choice probability of others.

```
# compute conditional choice probability
p_marginal <- compute_ccp_game(p_marginal, V, PI, G, delta, A, S)
p_marginal
```

```
## # A tibble: 750 x 4
##        i     l     a     p
##    <int> <int> <int> <dbl>
##  1     1     1     0 0.683
##  2     1     1     1 0.317
##  3     1     2     0 0.734
##  4     1     2     1 0.266
##  5     1     3     0 0.794
##  6     1     3     1 0.206
##  7     1     4     0 0.840
##  8     1     4     1 0.160
##  9     1     5     0 0.881
## 10     1     5     1 0.119
## # ... with 740 more rows
```

11. Write a function `solve_dynamic_game(PI, G, L, K, delta, lambda, A, S)` that find the equilib-

rium conditional choice probability and ex-ante value function by iterating the update of an ex-ante value function and a best-response conditional choice probability. The iteration should stop when $\max_s |V^{(r+1)}(s) - V^{(r)}(s)| < \lambda$ with $\lambda = 10^{-10}$. There is no theoretical guarantee for the convergence.

```
# solve the dynamic game model
output <-
  solve_dynamic_game(PI, G, L, K, delta, lambda, A, S)
save(output, file = "data/A8_equilibrium.RData")
```

```
load(file = "data/A8_equilibrium.RData")
p_marginal <- output$p_marginal; head(p_marginal)
```

```
## # A tibble: 6 x 4
##       i     l     a     p
##   <int> <int> <int> <dbl>
## 1     1     1     0 0.650
## 2     1     1     1 0.350
## 3     1     2     0 0.712
## 4     1     2     1 0.288
## 5     1     3     0 0.785
## 6     1     3     1 0.215
```

```
V <- output$V[[N]]; head(V)
```

```
## [1] 13.25670 12.39394 11.47346 10.82808 10.53018 12.39394
```

```
# compute joint conitional choice probability
p_joint <- compute_p_joint(p_marginal, A, S); head(p_joint)
```

```
## # A tibble: 6 x 3
##       l     k      p
##   <int> <int>  <dbl>
## 1     1     1 0.275
## 2     1     2 0.148
## 3     1     3 0.148
## 4     1     4 0.0795
## 5     1     5 0.148
## 6     1     6 0.0795
```

12. Write a function `simulate_dynamic_game(p_joint, l, G, N, T, S, A, seed)` that simulate the data for a market starting from an initial state for T periods. The function should accept a value of seed and set the seed at the beginning of the procedure inside the function, because the process is stochastic.

```
# simulate a dynamic game
# set initial state profile
l <- 1
# draw simulation for a firm
seed <- 1
df <- simulate_dynamic_game(p_joint, l, G, N, T, S, A, seed)
df
```

```
## # A tibble: 300 x 6
##        t     i     l     k     s     a
##    <int> <int> <dbl> <dbl> <int> <int>
##  1     1     1     1     1     1     0
##  2     1     2     1     1     1     0
##  3     1     3     1     1     1     0
```

```
##  4     2     1     1     5     1     0
##  5     2     2     1     5     1     0
##  6     2     3     1     5     1     1
##  7     3     1    26     6     1     1
##  8     3     2    26     6     1     0
##  9     3     3    26     6     2     1
## 10     4     1    26     5     1     0
## # ... with 290 more rows
```

13. Write a function `simulate_dynamic_decision_across_firms(p_joint, l, G, N, T, M, S, A, seed)` that returns simulation data for N firm. For firm i, set the seed at i

```
# simulate data across markets
df <- simulate_dynamic_decision_across_markets(p_joint, l, G, N, T, M, S, A)
save(df, file = "data/A8_df.RData")
```

```
load(file = "data/A8_df.RData")
df
```

```
## # A tibble: 300,000 x 7
##        m     t     i     l     k     s     a
##    <int> <int> <int> <dbl> <dbl> <int> <int>
##  1     1     1     1     1     1     1     0
##  2     1     1     2     1     1     1     0
##  3     1     1     3     1     1     1     0
##  4     1     2     1     1     5     1     0
##  5     1     2     2     1     5     1     0
##  6     1     2     3     1     5     1     1
##  7     1     3     1    26     6     1     1
##  8     1     3     2    26     6     1     0
##  9     1     3     3    26     6     2     1
## 10     1     4     1    26     5     1     0
## # ... with 299,990 more rows
```

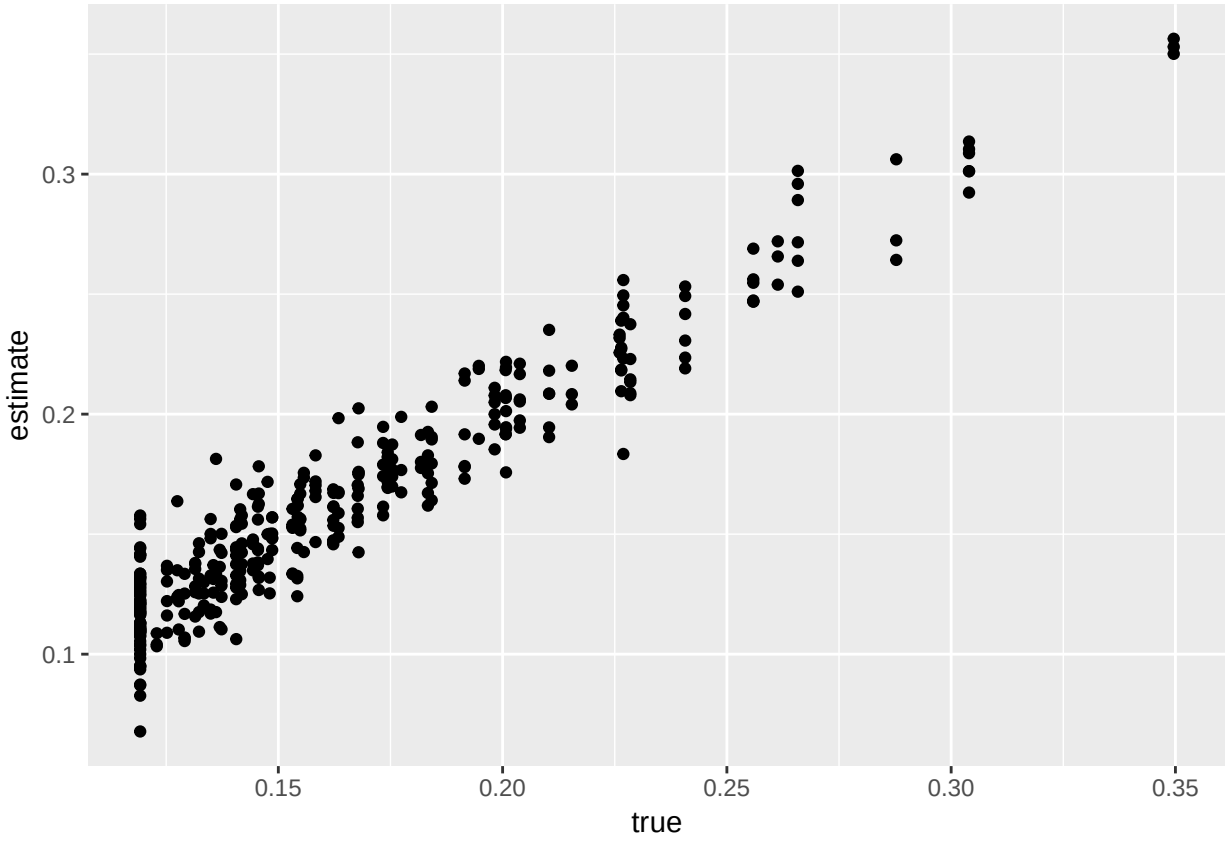
```
summary(df)
```

```
##        m                t                i           l
##  Min.   :   1.0   Min.   :  1.00   Min.   :1   Min.   :  1.00
##  1st Qu.: 250.8   1st Qu.: 25.75   1st Qu.:1   1st Qu.: 28.00
##  Median : 500.5   Median : 50.50   Median :2   Median : 53.00
##  Mean   : 500.5   Mean   : 50.50   Mean   :2   Mean   : 55.08
##  3rd Qu.: 750.2   3rd Qu.: 75.25   3rd Qu.:3   3rd Qu.: 83.00
##  Max.   :1000.0   Max.   :100.00   Max.   :3   Max.   :125.00
##        k               s               a
##  Min.   :1.000   Min.   :1.000   Min.   :0.0000
##  1st Qu.:1.000   1st Qu.:2.000   1st Qu.:0.0000
##  Median :1.000   Median :3.000   Median :0.0000
##  Mean   :2.244   Mean   :2.753   Mean   :0.1776
##  3rd Qu.:3.000   3rd Qu.:4.000   3rd Qu.:0.0000
##  Max.   :8.000   Max.   :5.000   Max.   :1.0000
```

14. Write a function `estimate_ccp_marginal_game(df)` that returns a non-parametric estimate of the marginal conditional choice probability for each firm in the data. Compare the estimated conditional choice probability and the true conditional choice probability by a bar plot.

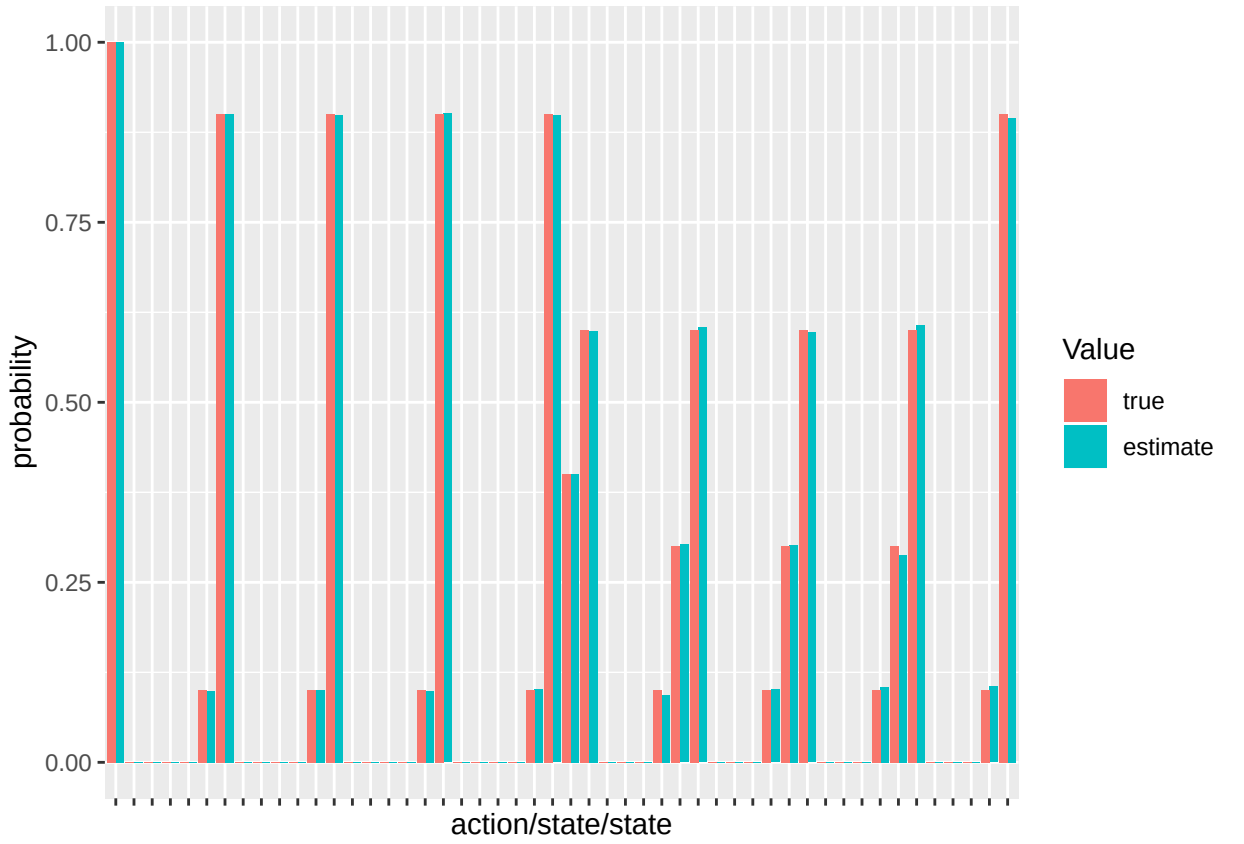
```
# non-parametrically estimate the conditional choice probability
p_marginal_est <- estimate_ccp_marginal_game(df)
```

```
check_ccp <- p_marginal_est %>%
  dplyr::rename(estimate = p) %>%
  dplyr::left_join(p_marginal, by = c("i", "l", "a")) %>%
  dplyr::rename(true = p) %>%
  dplyr::filter(a == 1)
ggplot(data = check_ccp, aes(x = true, y = estimate)) +
  geom_point() +
  labs(fill = "Value") + xlab("true") + ylab("estimate")
```



15. Write a function `estimate_G_marginal(df)` that returns a non-parametric estimate of the marginal transition probability matrix. Compare the estimated transition matrix and the true transition matrix by a bar plot.

```
# non-parametrically estimate individual transition probability
G_marginal_est <- estimate_G_marginal(df)
check_G <- data.frame(type = "true", reshape2::melt(G_marginal))
check_G_est <- data.frame(type = "estimate", reshape2::melt(G_marginal_est))
check_G <- rbind(check_G, check_G_est)
check_G$variable = paste(check_G$Var1, check_G$Var2, sep = "_")
ggplot(data = check_G, aes(x = variable, y = value,
                           fill = type)) +
    geom_bar(stat = "identity", position = "dodge") +
  labs(fill = "Value") + xlab("action/state/state") + ylab("probability") +
  theme(axis.text.x = element_blank())
```



17.2 Estimate parameters

1. Vectorize the parameters as follows:

```
theta_1 <- c(alpha, beta, eta)
theta_2 <- c(kappa, gamma)
theta <- c(theta_1, theta_2)
```

We estimate the parameters by a CCP approach.

1. Write a function `estimate_theta_2_game(df)` that returns the estimates of κ and γ directly from data by counting relevant events.

```
# estimate theta_2
theta_2_est <- estimate_theta_2_game(df); theta_2_est
```

```
## [1] 0.09995377 0.60136442
```

The objective function of the minimum distance estimator based on the conditional choice probability approach is:

$$\frac{1}{NKm_s}\sum_{i=1}^{N}\sum_{l=1}^{m_s}\sum_{k=1}^{K}\{\hat{p}_i(a_k|s_l) - p_i^{(\theta_1,\theta_2)}(a_k|s_l)\}^2,$$

where $\hat{p}_i$ is the non-parametric estimate of the marginal conditional choice probability and $p_i^{(\theta_1,\theta_2)}$ is the marginal conditional choice probability under parameters θ_1 and θ_2 given $\hat{p}_i$. a_k is k-th action for a firm and s_l is l-th state profile.

2. Write a function `compute_CCP_objective_game(theta_1, theta_2, p_est, L, K, delta)` that returns the objective function of the above minimum distance estimator given a non-parametric estimate of the conditional choice probability and θ_1 and θ_2.

```
# compute the objective function of the minimum distance estimator based on the CCP approach
objective <- compute_CCP_objective_game(theta_1, theta_2, p_marginal_est, A, S, delta, lambda)
save(objective, file = "data/A8_objective.RData")
```

```
load(file = "data/A8_objective.RData")
objective
```

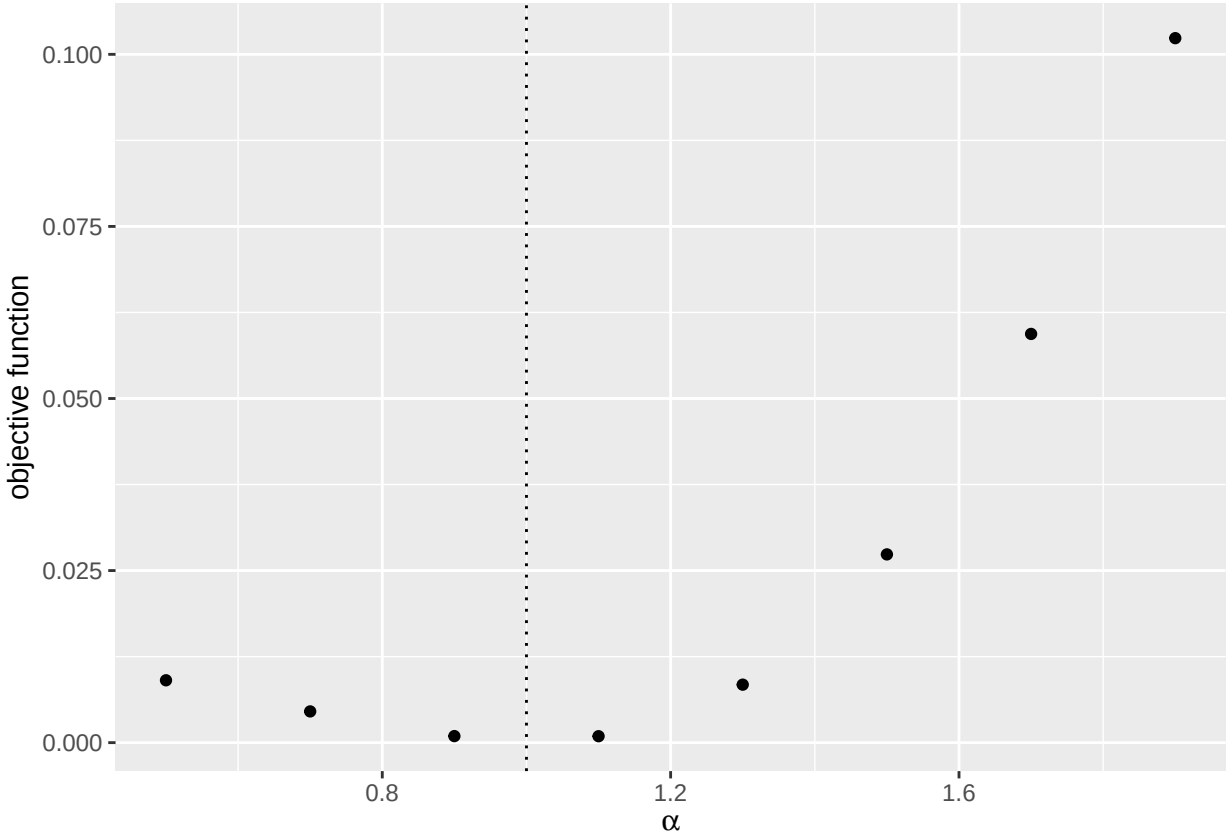
```
## [1] 0.0002737567
```

3. Check the value of the objective function around the true parameter.

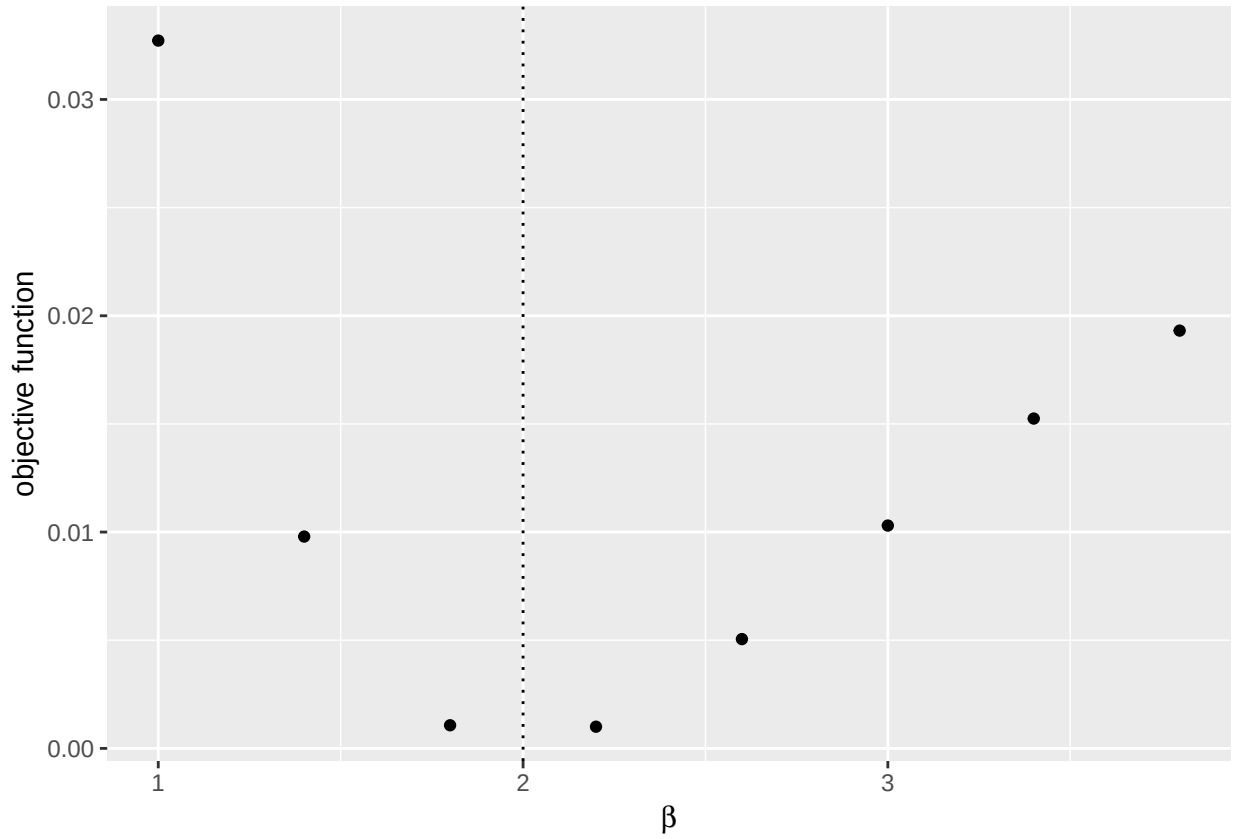
```
# label
label <- c("\\alpha", "\\beta", "\\eta")
label <- paste("$", label, "$", sep = "")
# compute the graph
graph <- foreach (i = 1:length(theta_1)) %do% {
  theta_i <- theta_1[i]
  theta_i_list <- theta_i * seq(0.5, 2, by = 0.2)
  objective_i <-
    foreach (j = 1:length(theta_i_list),
             .combine = "rbind") %dopar% {
               theta_ij <- theta_i_list[j]
               theta_j <- theta_1
               theta_j[i] <- theta_ij
               objective_ij <-
                 compute_CCP_objective_game(theta_j, theta_2, p_marginal_est, A, S, delta, lambda)
               return(objective_ij)
             }
  df_graph <- data.frame(x = theta_i_list, y = objective_i)
  g <- ggplot(data = df_graph, aes(x = x, y = y)) +
    geom_point() +
    geom_vline(xintercept = theta_i, linetype = "dotted") +
    ylab("objective function") + xlab(TeX(label[i]))
  return(g)
}
save(graph, file = "data/A8_CCP_graph.RData")
```

```
load(file = "data/A8_CCP_graph.RData")
graph
```

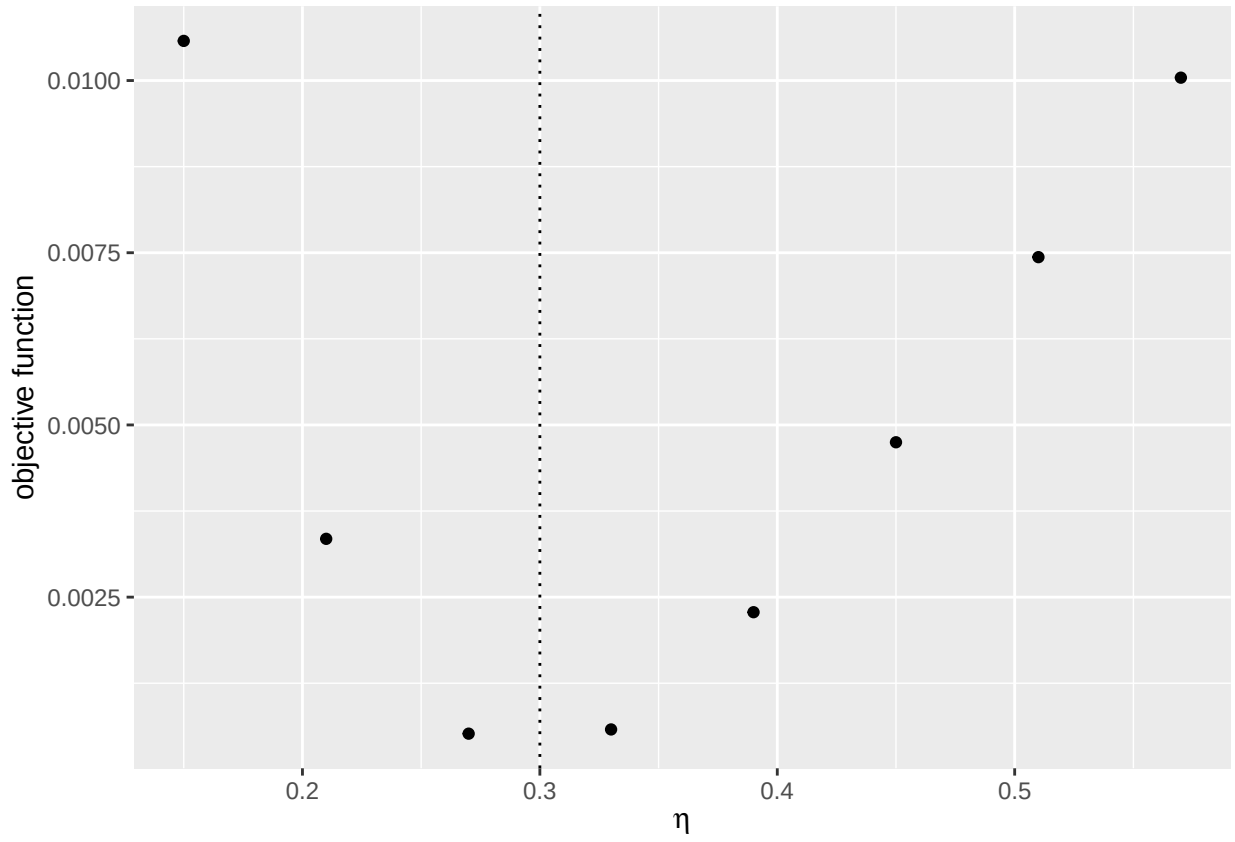
```
## [[1]]
```




```
##
## [[2]]
```




```
##
## [[3]]
```



4. Estimate the parameters by minimizing the objective function. To keep the model to be well-defined, impose an ad hoc lower and upper bounds such that $\alpha \in [0, 1], \beta \in [0, 5], \delta \in [0, 1]$.

```
lower <- rep(0, length(theta_1))
upper <- c(1, 5, 0.3)
CCP_result <-
  optim(par = theta_1,
        fn = compute_CCP_objective_game,
        method = "L-BFGS-B",
        lower = lower,
        upper = upper,
        theta_2 = theta_2_est,
        p_marginal_est = p_marginal_est,
        A = A,
        S = S,
        delta = delta,
        lambda = lambda)
save(CCP_result, file = "data/A8_CCP_result.RData")
```

```
load(file = "data/A8_CCP_result.RData")
CCP_result
```

```
## $par
## [1] 1.000000 2.011446 0.294446
##
## $value
## [1] 0.0002702126
##
```

```
## $counts
## function gradient
##       12       12
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

```
compare <-
  data.frame(
    true = theta_1,
    estimate = CCP_result$par
  ); compare
```

```
##   true estimate
## 1  1.0 1.000000
## 2  2.0 2.011446
## 3  0.3 0.294446
```

Chapter 18

Assignment 9: Auction

The deadline is **May 20 1:30pm**.

18.1 Simulate data

We simulate bid data from a second- and first-price sealed bid auctions.

First, we draw bid data from a second-price sealed bid auctions. Suppose that for each auction $t = 1, \cdots, T$, there are $i = 1, \cdots, n_t$ potential bidders. At each auction, an auctioneer allocates one item and sets the reserve price at r_t. When the signal for bidder i in auction t is x_{it}, her expected value of the item is x_{it}. A signal x_{it} is drawn from an i.i.d. beta distribution $B(\alpha, \beta)$. Let $F_X(\cdot; \alpha, \beta)$ be its distribution and $f_X(\cdot; \alpha, \beta)$ be the density. A reserve is set at 0.2. n_t is drawn from a Poisson distribution with mean λ. An equilibrium strategy is such that a bidder participates and bids $\beta(x) = x$ if $x \geq r_t$ and does not participate otherwise.

1. Set the constants and parameters as follows:

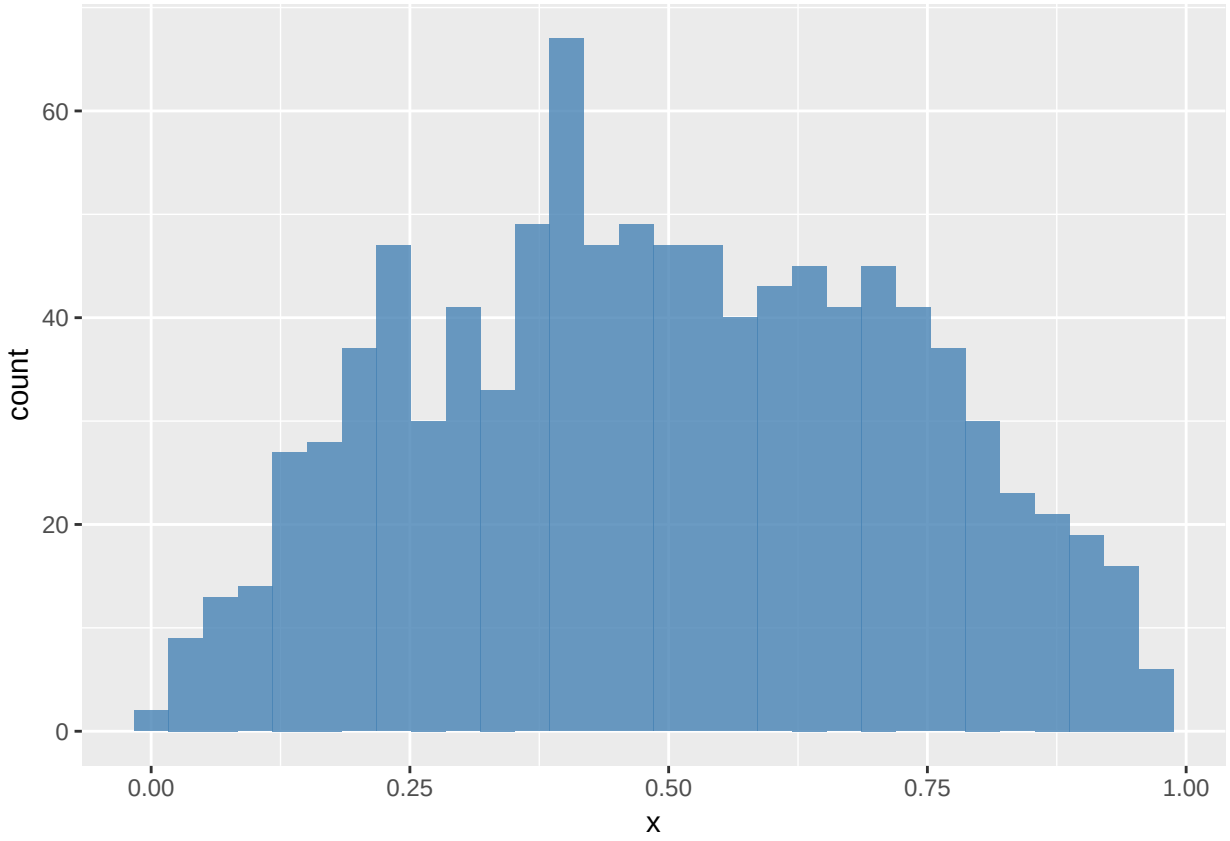
```
# set seed
set.seed(1)
# number of auctions
T <- 100
# parameter of value distribution
alpha <- 2
beta <- 2
# parameters of reserve price distribution
gamma <- 1
delta <- 3
# prameters of number of potential bidders
lambda <- 10
```

2. Draw a vector of valuations and reservation prices.

```
# number of bidders
N <- rpois(T, lambda)
# draw valuations
valuation <-
  foreach (tt = 1:T, .combine = "rbind") %do% {
    n_t <- N[tt]
    header <- expand.grid(t = tt, i = 1:n_t)
    return(header)
  }
```

```
valuation <- valuation %>%
  tibble::as_tibble() %>%
  dplyr::mutate(x = rbeta(length(i), alpha, beta))
ggplot(valuation, aes(x = x)) + geom_histogram(fill = "steelblue", alpha = 0.8)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# draw reserve prices
reserve <- 0.2
reserve <- tibble::tibble(t = 1:T, r = reserve)
```

3. Write a function `compute_winning_bids_second(valuation, reserve)` that returns a winning bid from each second-price auction. It returns nothing for an auction in which no bid was above the reserve price. In the output, `t` refers to the auction index, `m` to the number of actual bidders, `r` to the reserve price, and `w` to the winning bid.

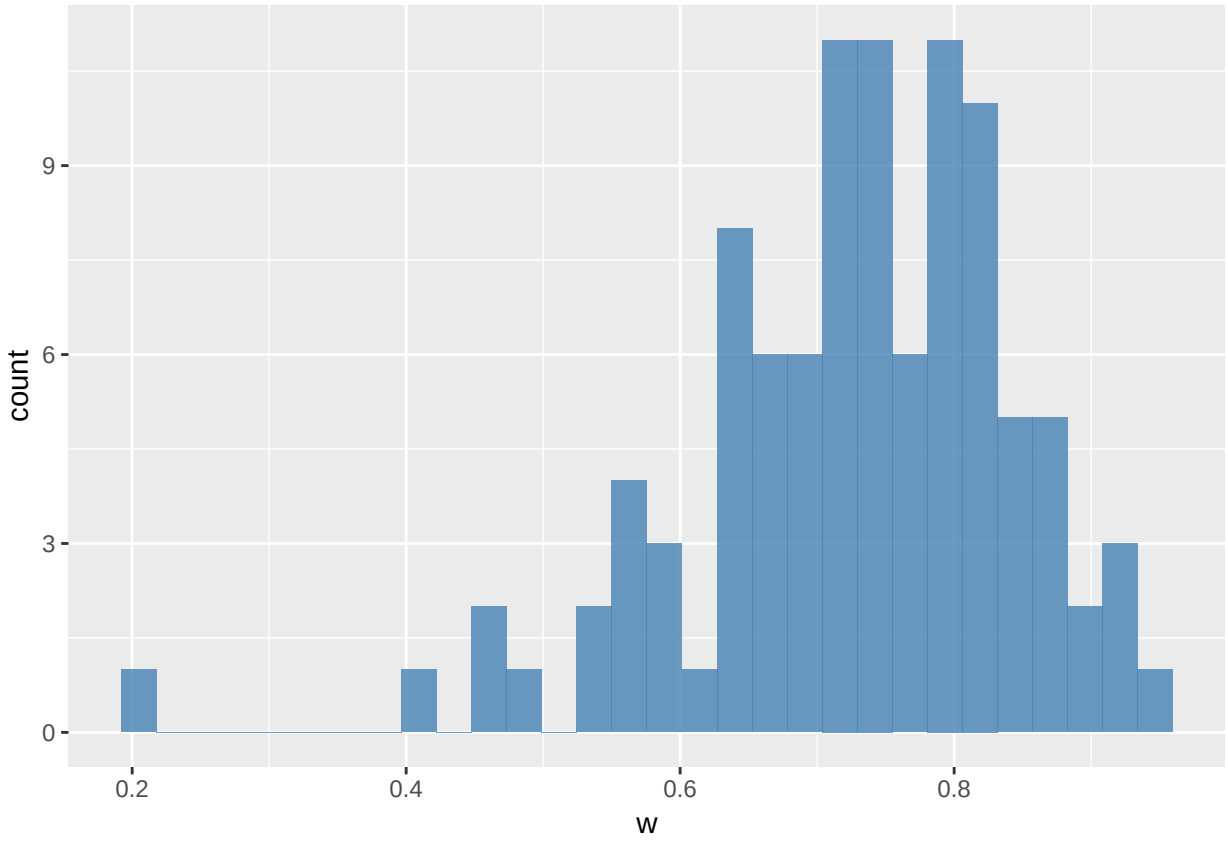
```
# compute winning bids from second-price auction
df_second_w <-
  compute_winning_bids_second(valuation, reserve)
df_second_w
```

```
## # A tibble: 100 x 5
##       t     n     m     r     w
##   <int> <int> <int> <dbl> <dbl>
##  1     1     8     8   0.2 0.637
##  2     2    10    10   0.2 0.647
##  3     3     7     5   0.2 0.484
##  4     4    11     8   0.2 0.804
##  5     5    14    12   0.2 0.920
```

```
##  6     6    12    11   0.2 0.942
##  7     7    11     9   0.2 0.810
##  8     8     9     9   0.2 0.724
##  9     9    14    14   0.2 0.880
## 10    10    11     9   0.2 0.677
## # ... with 90 more rows
```

```
ggplot(df_second_w, aes(x = w)) + geom_histogram(fill = "steelblue", alpha = 0.8)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Next, we simulate bid data from first-price sealed bid auctions. The setting is the same as the second-price auctions expect for the auction rule. An equilibrium bidding strategy is to participate and bid:

$$\beta(x) = x - \frac{\int_{r_t}^{x} F_X(t)^{N-1}}{F_X(x)^{N-1}},$$

if $x \geq r$ and not to participate otherwise when there are more than one active bidder. If $x \geq r$ and there is only one bidder, the bidder bids r_t.

4. Write a function `bid_first(x, r, alpha, beta, n)` that returns the equilibrium bid. To integrate a function, use `integrate` function in R. It returns 0 if $x < r$.

```
# compute bid from first-price auction
n <- N[1]
m <- N[1]
x <- valuation[1, "x"] %>% as.numeric(); x
```

```
## [1] 0.3902289
```

```
r <- reserve[1, "r"] %>% as.numeric(); r
```

```
## [1] 0.2
```

```
b <- bid_first(x, r, alpha, beta, n); b
```

```
## [1] 0.3596662
```

```
x <- r/2; x
```

```
## [1] 0.1
```

```
b <- bid_first(x, r, alpha, beta, n); b
```

```
## [1] 0
```

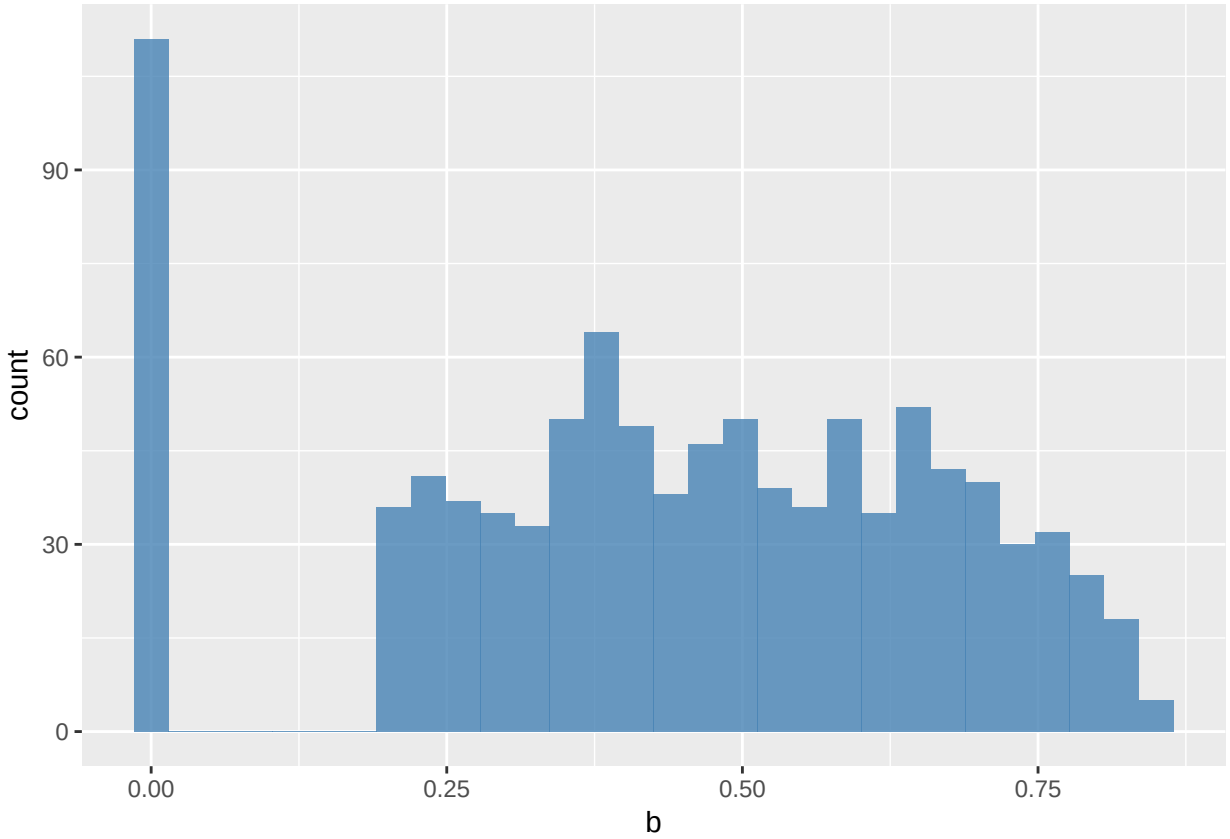
```
b <- bid_first(1, r, alpha, beta, n); b
```

```
## [1] 0.7978258
```

5. Write a function `compute_bids_first(valuation, reserve, alpha, beta)` that returns bids from each first-price auctions. It returns bids below the reserve price.

```
# compute bid data from first-price auctions
df_first <- compute_bids_first(valuation, reserve, alpha, beta)
ggplot(df_first, aes(x = b)) + geom_histogram(fill = "steelblue", alpha = 0.8)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



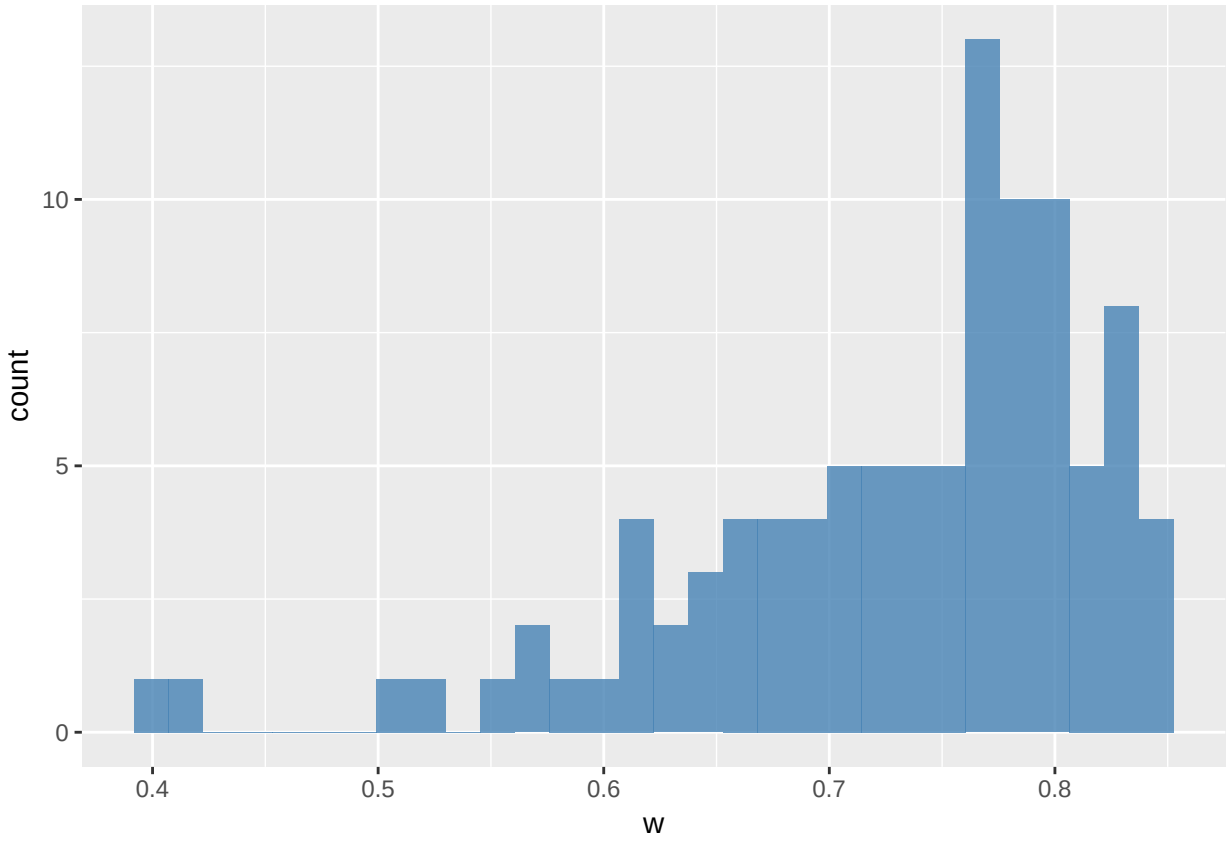
6. Write a function `compute_winning_bids_first(valuation, reserve, alpha, beta)` that returns only the winning bids from each first-price auction. It will call `compute_bids_first` inside the function. It does not return anything when no bidder bids above the reserve price.

```
# compute winning bids from first-price auctions
df_first_w <-
  compute_winning_bids_first(valuation, reserve, alpha, beta)
df_first_w
```

```
## # A tibble: 100 x 5
##        t     n     m     r     w
##    <int> <int> <int> <dbl> <dbl>
##  1     1     8     8   0.2 0.731
##  2     2    10    10   0.2 0.638
##  3     3     7     5   0.2 0.525
##  4     4    11     8   0.2 0.818
##  5     5    14    12   0.2 0.842
##  6     6    12    11   0.2 0.833
##  7     7    11     9   0.2 0.772
##  8     8     9     9   0.2 0.753
##  9     9    14    14   0.2 0.849
## 10    10    11     9   0.2 0.803
## # ... with 90 more rows
```

```
ggplot(df_first_w, aes(x = w)) + geom_histogram(fill = "steelblue", alpha = 0.8)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



18.2 Estimate the parameters

We first estimate the parameters from the winning bids data of second-price auctions. We estimate the parameters by maximizing a log-likelihood.

$$l(\alpha, \beta) := \frac{1}{T} \sum_{t=1}^{T} \ln \frac{h_t(w_t)^{1\{m_t > 1\}} \mathbb{P}\{m_t = 1\}}{1 - \mathbb{P}\{m_t = 0\}},$$

where:

$$\mathbb{P}\{m_t = 0\} := F_X(r_t)^{n_t},$$
$$\mathbb{P}\{m_t = 1\} := n_t F_X(r_t; \alpha, \beta)^{n_t - 1}[1 - F_X(r_t; \alpha, \beta)].$$

$$h_t(w_t) := n_t(n_t - 1) F_X(w_t; \alpha, \beta)^{n_t - 2}[1 - F_X(w_t; \alpha, \beta)] f_X(w_t; \alpha, \beta).$$

1. Write a function `compute_p_second_w(w, r, m, n, alpha, beta)` that computes $\mathbb{P}\{m_t = 1\}$ if $m_t = 1$ and $h_t(w_t)$ if $m_t > 1$.

```
# compute probability density for winning bids from a second-price auction
w <- df_second_w[1, ]$w
r <- df_second_w[1, ]$r
m <- df_second_w[1, ]$m
n <- df_second_w[1, ]$n
compute_p_second_w(w, r, m, n, alpha, beta)
```

```
## [1] 2.752949
```

2. Write a function `compute_m0(r, n, alpha, beta)` that computes $\mathbb{P}\{m_t = 0\}$.

```
# compute non-participation probability
compute_m0(r, n, alpha, beta)
```

```
## [1] 1.368569e-08
```

2. Write a function `compute_loglikelihood_second_price_w(theta, df_second_w)` that computes the log-likelihood for a second-price auction winning bid data.

```
# compute log-likelihood for winning bids from second-price auctions
theta <- c(alpha, beta)
compute_loglikelihood_second_price_w(theta, df_second_w)
```

```
## [1] 0.9849261
```

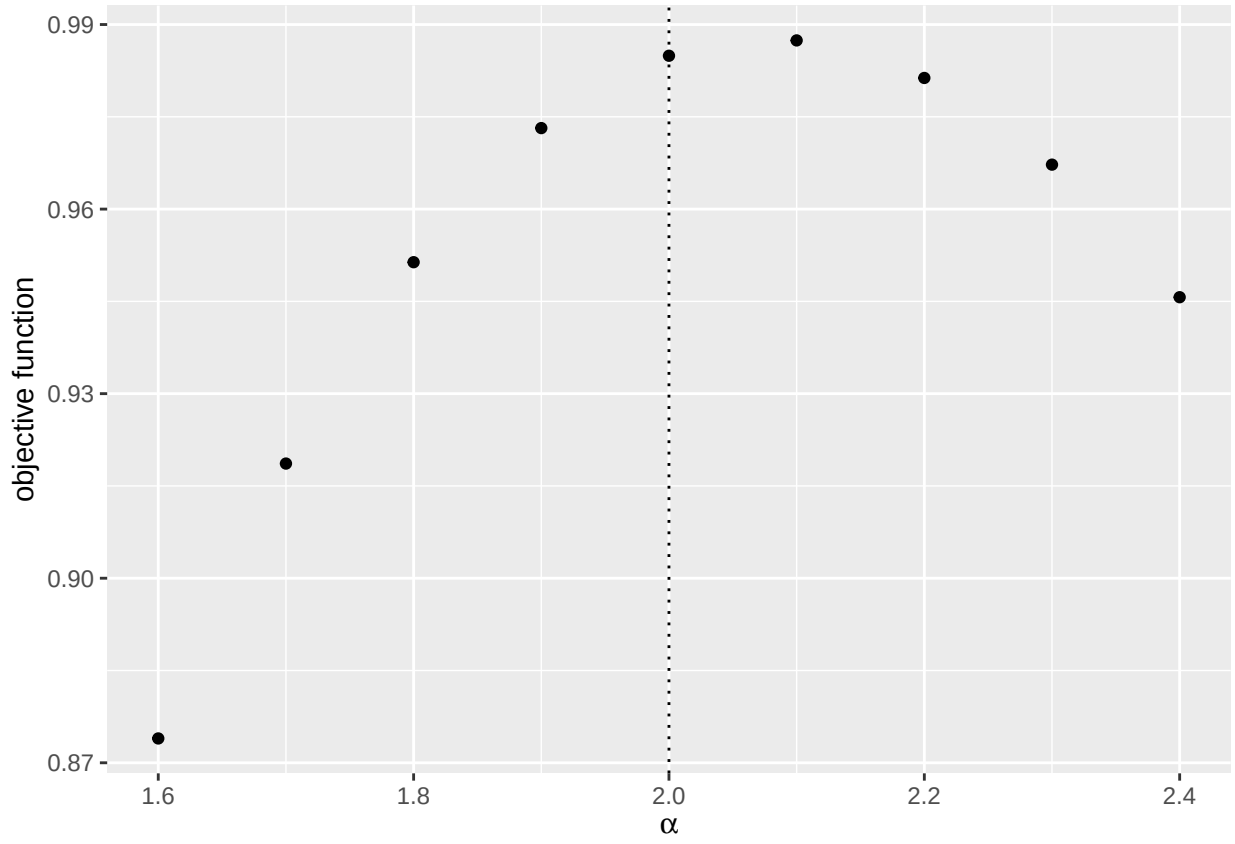
3. Compare the value of objective function around the true parameters.

```
# label
label <- c("\\alpha", "\\beta")
label <- paste("$", label, "$", sep = "")
# compute the graph
graph <- foreach (i = 1:length(theta)) %do% {
  theta_i <- theta[i]
  theta_i_list <- theta_i * seq(0.8, 1.2, by = 0.05)
  objective_i <-
    foreach (j = 1:length(theta_i_list),
             .combine = "rbind") %dopar% {
               theta_ij <- theta_i_list[j]
               theta_j <- theta
               theta_j[i] <- theta_ij
```

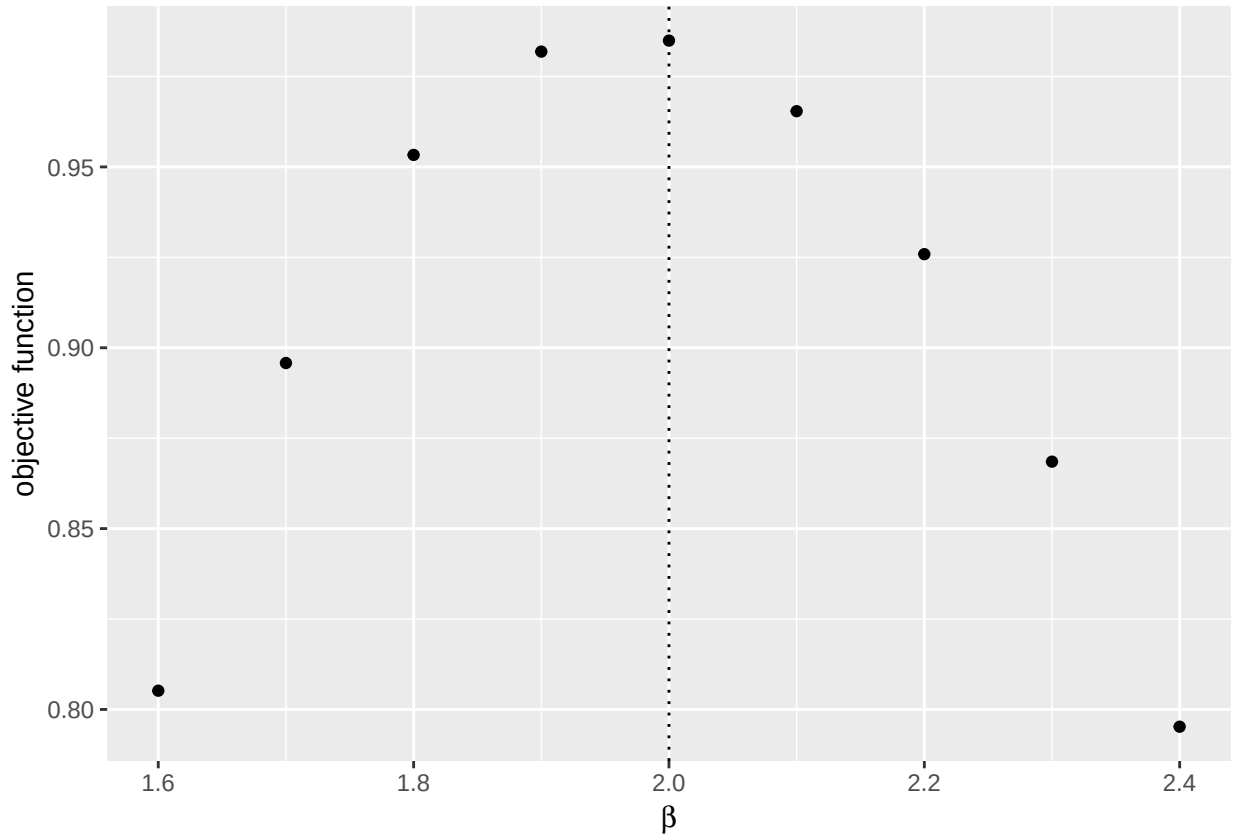
```
              objective_ij <-
                compute_loglikelihood_second_price_w(
                  theta_j, df_second_w)
              return(objective_ij)
            }
  df_graph <- data.frame(x = as.numeric(theta_i_list),
                         y = as.numeric(objective_i))
  g <- ggplot(data = df_graph, aes(x = x, y = y)) +
    geom_point() +
    geom_vline(xintercept = theta_i, linetype = "dotted") +
    ylab("objective function") + xlab(TeX(label[i]))
  return(g)
}
save(graph, file = "data/A9_second_parametric_graph.RData")
```

```
load(file = "data/A9_second_parametric_graph.RData")
graph
```

```
## [[1]]
```




```
##
## [[2]]
```



4. Estimate the parameters by maximizing the log-likelihood.

```
result_second_parametric <-
  optim(
    par = theta,
    fn = compute_loglikelihood_second_price_w,
    df_second_w = df_second_w,
    method = "L-BFGS-B",
    control = list(fnscale = -1)
  )
save(result_second_parametric, file = "data/A9_result_second_parametric.RData")
```

```
load(file = "data/A9_result_second_parametric.RData")
result_second_parametric
```

```
## $par
## [1] 2.199238 2.078327
##
## $value
## [1] 0.9883372
##
## $counts
## function gradient
##       11       11
##
## $convergence
## [1] 0
##
```

```
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

```
comparison <-
  data.frame(
    true = theta,
    estimate = result_second_parametric$par
  )
comparison
```

```
##   true estimate
## 1    2 2.199238
## 2    2 2.078327
```

Next, we estimate the parameters from the winning bids data from first-price auctions. We estimate the parameters by maximizing a log-likelihood.

5. Write a function `inverse_bid_equation(x, b, r, alpha, beta, n)` that returns $\beta(x)-b$ for a bid b. Write a function `inverse_bid_first(b, r, alpha, beta, n)` that is an inverse function `bid_first` with respect to the signal, that is,
$$\eta(b) := \beta^{-1}(b).$$

To do so, we can use a built-in function called `uniroot`, which solves x such that $f(x) = 0$ for scalar x. In `uniroot`, `lower` and `upper` are set at r_t and $\beta(1)$, respectively.

```
r <- df_first_w[1, "r"] %>%
  as.numeric()
n <- df_first_w[1, "n"] %>%
  as.integer()
b <- 0.5 * r + 0.5
x <- 0.5
# compute invecrse bid equation
inverse_bid_equation(x, b, r, alpha, beta, n)
```

```
## [1] -0.1421105
```

```
# compute inverse bid
inverse_bid_first(b, r, alpha, beta, n)
```

```
## [1] 0.6653238
```

The log-likelihood conditional on $m_t \geq 1$ is:

$$l(\alpha, \beta) := \frac{1}{T}\sum_{t=1}^{T} \log \frac{h_t(w_t)}{1 - F_X(r_t)^{n_t}},$$

where the probability density of having w_t is:

$$\begin{aligned} h_t(w_t) &= n_t F_X[\eta_t(w_t)]^{n_t-1} f_X[\eta_t(w_t)]\eta_t'(w_t) \\ &= \frac{n_t F_X[\eta_t(w_t)]^{n_t}}{(n_t-1)[\eta_t(w_t) - w_t]}, \end{aligned}$$

where the second equation is from the first-order condition.

6. Write a function `compute_p_first_w(w, r, alpha, beta, n)` that returns $h_t(w)$. Remark that the equilibrium bid at specific parameters is `bid_first(1, r, alpha, beta, n)`. If the observed winning bid `w` is above the upper limit, the function will issue an error. Therefore, inside the function `compute_p_first_w(w, r, alpha, beta, n)`, check if `w` is above `bid_first(1, r, alpha, beta, n)` and if so return 10^{-6}.

```
# compute probability density for a winning bid from a first-price auction
w <- 0.5
compute_p_first_w(w, r, alpha, beta, n)
```

```
## [1] 0.2720049
```

```
upper <- bid_first(1, r, alpha, beta, n)
compute_p_first_w(upper + 1, r, alpha, beta, n)
```

```
## [1] 1e-06
```

7. Write a function `compute_loglikelihood_first_price_w(theta, df_first_w)` that computes the log-likelihood for a first-price auction winning bid data.

```
# compute log-likelihood for winning bids for first-price auctions
compute_loglikelihood_first_price_w(theta, df_first_w)
```

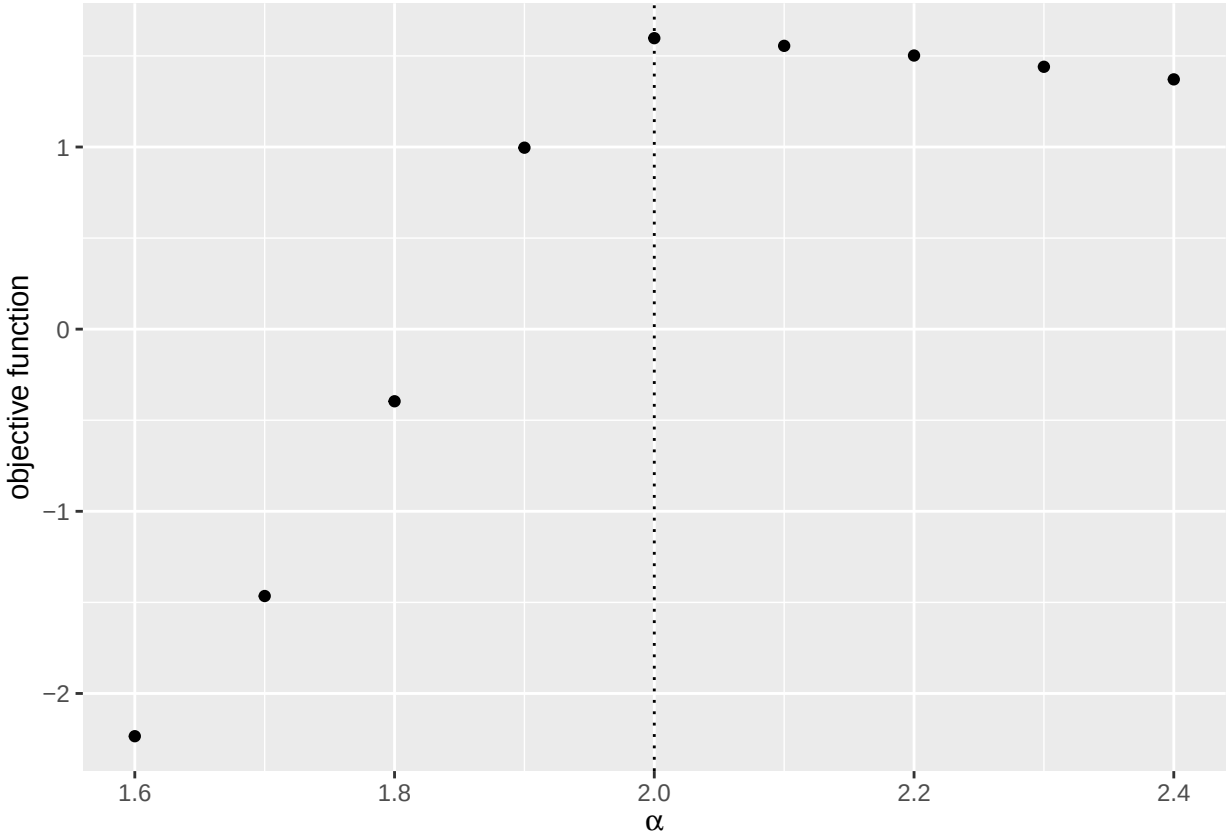
```
## [1] 1.597414
```

8. Compare the value of the objective function around the true parameters.

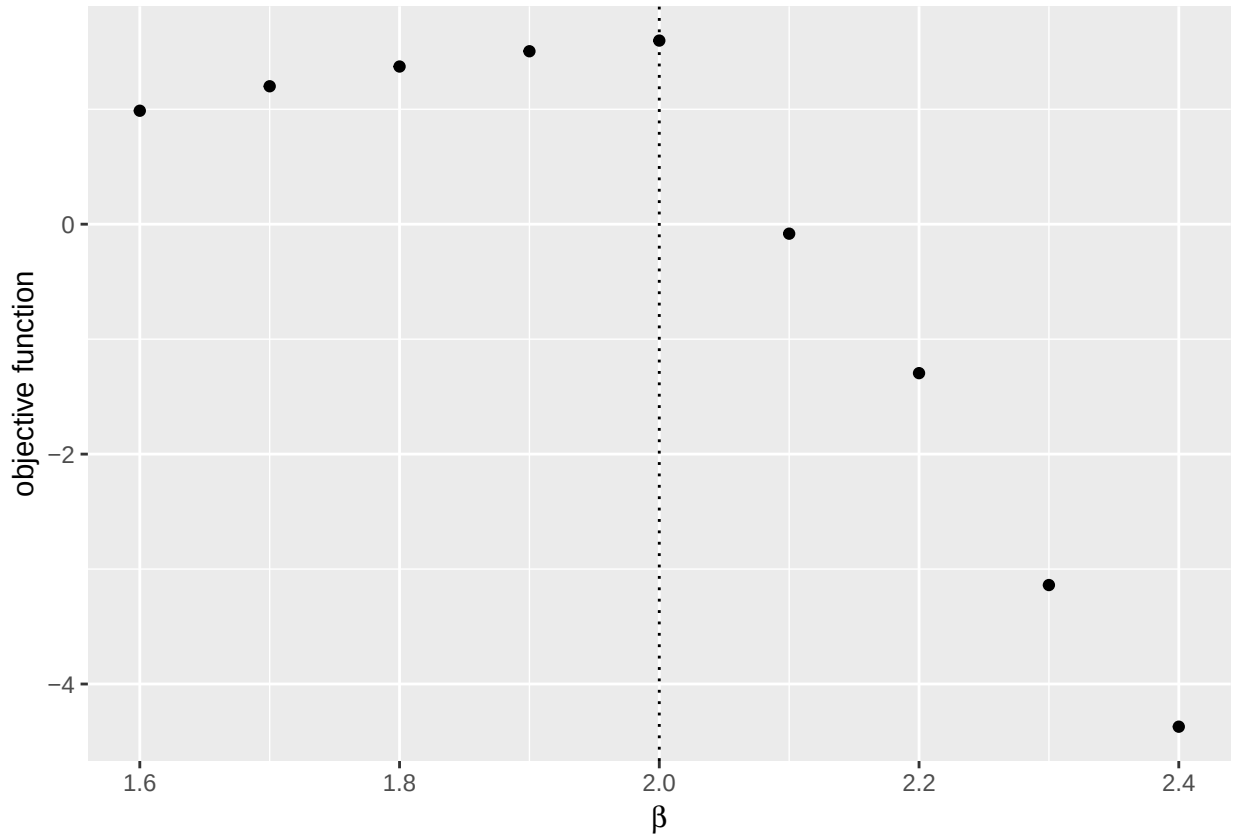
```
theta <- c(alpha, beta)
# label
label <- c("\\alpha", "\\beta")
label <- paste("$", label, "$", sep = "")
# compute the graph
graph <- foreach (i = 1:length(theta)) %do% {
  theta_i <- theta[i]
  theta_i_list <- theta_i * seq(0.8, 1.2, by = 0.05)
  objective_i <-
    foreach (j = 1:length(theta_i_list),
             .combine = "rbind") %do% {
               theta_ij <- theta_i_list[j]
               theta_j <- theta
               theta_j[i] <- theta_ij
               objective_ij <-
                 compute_loglikelihood_first_price_w(
                   theta_j, df_first_w)
               return(objective_ij)
             }
  df_graph <- data.frame(x = as.numeric(theta_i_list),
                         y = as.numeric(objective_i))
  g <- ggplot(data = df_graph, aes(x = x, y = y)) +
    geom_point() +
    geom_vline(xintercept = theta_i, linetype = "dotted") +
    ylab("objective function") + xlab(TeX(label[i]))
  return(g)
}
save(graph, file = "data/A9_first_parametric_graph.RData")
```

```
load(file = "data/A9_first_parametric_graph.RData")
graph
```

```
## [[1]]
```




```
##
## [[2]]
```



9. Estimate the parameters by maximizing the log-likelihood. Set the lower bounds at zero. Use the Nelder-Mead method. Otherwise the parameter search can go to extreme values because of the discontinuity at the point where the upper limit is below the observed bid.

```
result_first_parametric <-
  optim(
    par = theta,
    fn = compute_loglikelihood_first_price_w,
    df_first_w = df_first_w,
    method = "Nelder-Mead",
    control = list(fnscale = -1)
  )
save(result_first_parametric, file = "data/A9_result_first_parametric.RData")
```

```
load(file = "data/A9_result_first_parametric.RData")
result_first_parametric
```

```
## $par
## [1] 1.977676 2.004715
##
## $value
## [1] 1.607161
##
## $counts
## function gradient
##       91       NA
##
## $convergence
```

```
## [1] 0
##
## $message
## NULL
```

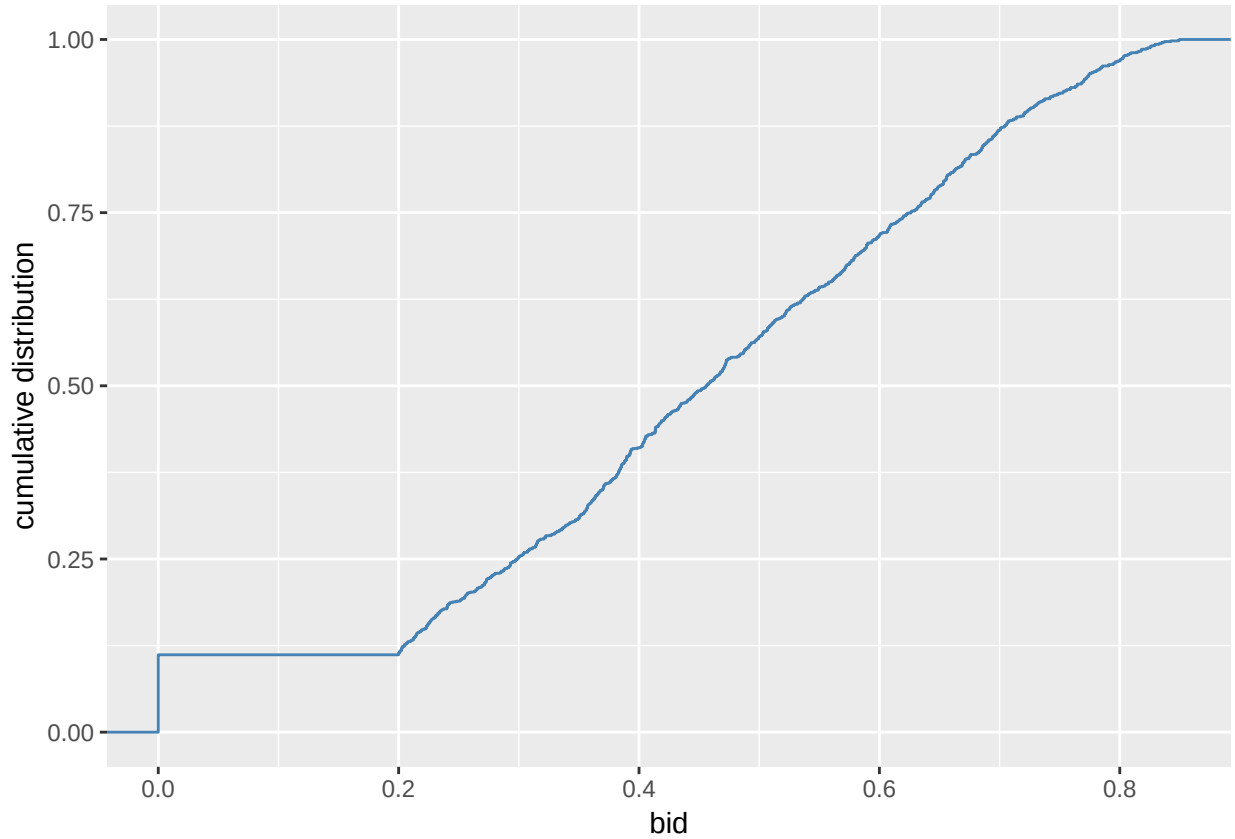
```
comparison <-
  data.frame(
    true = theta,
    estimate = result_first_parametric$par
  )
comparison
```

```
##   true estimate
## 1    2 1.977676
## 2    2 2.004715
```

Finally, we non-parametrically estimate the distribution of the valuation using bid data from first-price auctions `df_first`.

10. Write a function `F_b(b)` that returns an empirical cumulative distribution at `b`. This can be obtained by using a function `ecdf`. Also, write a function `f_b(b)` that returns an empirical probability density at `b`. This can be obtained by combining functions `approxfun` and `density`.

```
# cumulative distribution
ggplot(df_first, aes(x = b)) + stat_ecdf(color = "steelblue") +
  xlab("bid") + ylab("cumulative distribution")
```



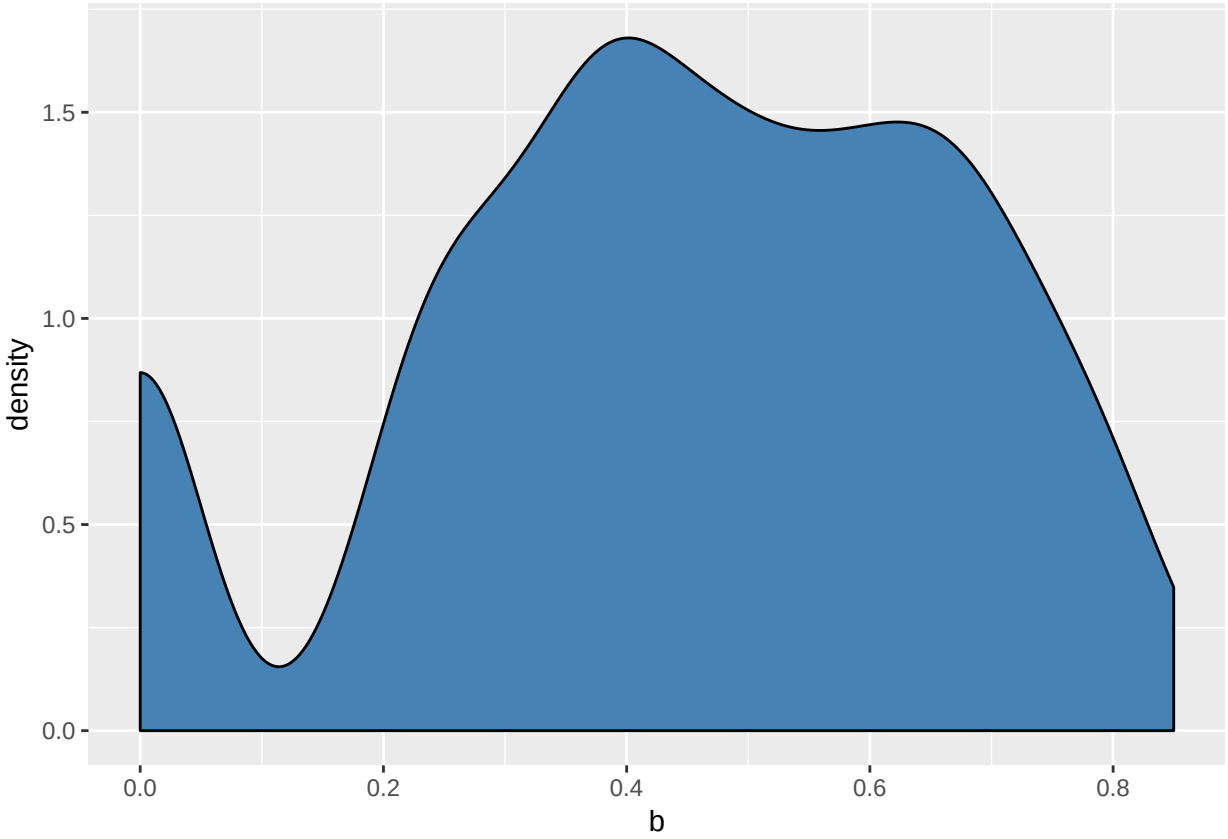

```
F_b <- ecdf(df_first$b)
F_b(0.4)
```

```
## [1] 0.4104628
```

```
F_b(0.6)
```

```
## [1] 0.7173038
```

```
# probability density
ggplot(df_first, aes(x = b)) + geom_density(fill = "steelblue")
```




```
f_b <- approxfun(density(df_first$b))
f_b(0.4)
```

```
## [1] 1.680124
```

```
f_b(0.6)
```

```
## [1] 1.469983
```

The equilibrium distribution and density of the highest rival's bid are:

$$H_b(b) := F_b(b)^{n-1},$$

$$h_b(b) := (n-1) f_b(b) F_b(b)^{n-2}.$$

11. Write a function `H_b(b, n)` and `h_b(b)` that return the equilibrium distribution and density of the highest rival's bid at point `b`.

```
H_b(0.4, n)
```

```
## [1] 0.001962983
```

```
h_b(0.4, n)
```

```
## [1] 0.05624476
```

When a bidder bids b, the implied valuation of her is:

$$x = b + \frac{H_b(b)}{h_b(b)}.$$

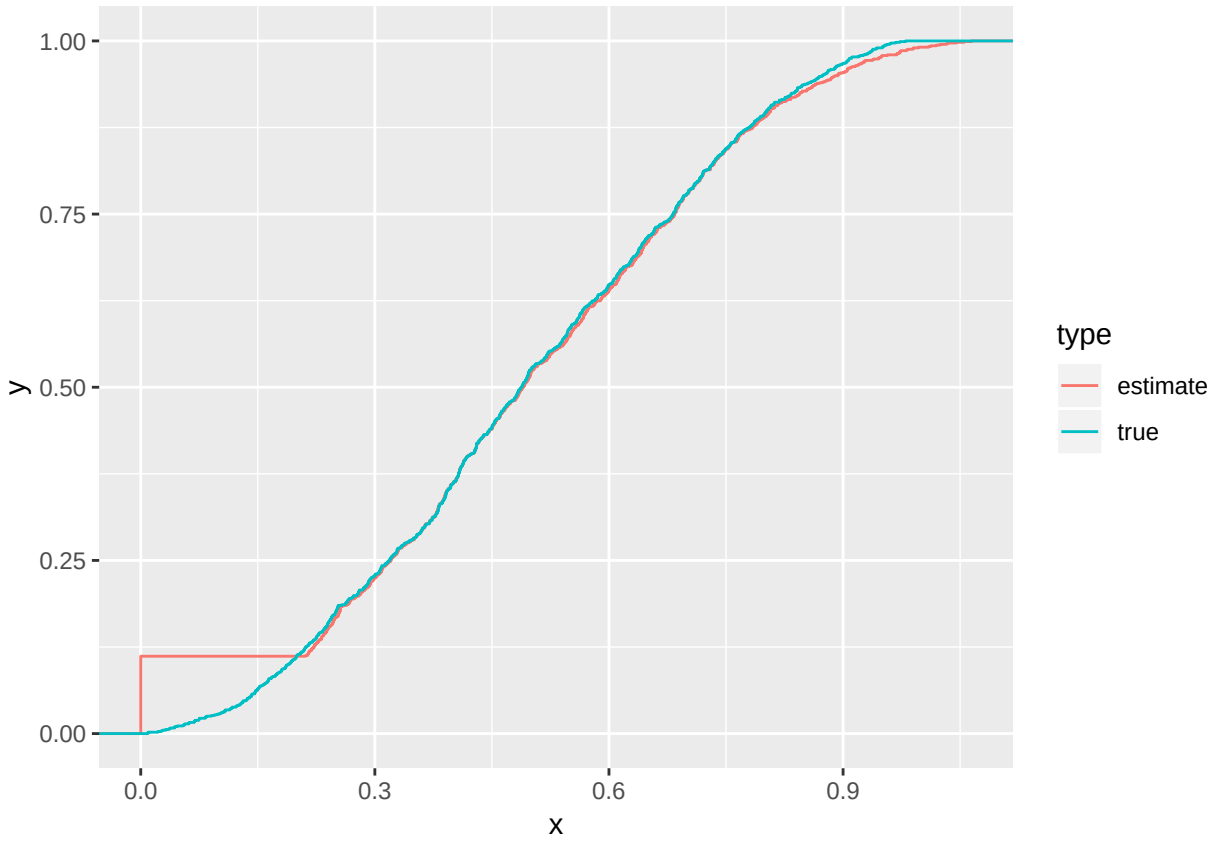
12. Write a function `compute_implied_valuation(b, n, r)` that returns the implied valuation given a bid. Let it return $x = 0$ if $b < r$, because we cannot know the value when the bid is below the reserve price.

```
r <- df_first[1, "r"]
n <- df_first[1, "n"]
compute_implied_valuation(0.4, n, r)
```

```
##           n
## 1 0.4349007
```

13. Obtain the vector of implied valuations from the vector of bids and draw the empirical cumulative distribution. Overlay it with the true empirical cumulative distribution of the valuations.

```
valuation_implied <- df_first %>%
  dplyr::rowwise() %>%
  dplyr::mutate(x = compute_implied_valuation(b, n, r)) %>%
  dplyr::ungroup() %>%
  dplyr::select(x) %>%
  dplyr::mutate(type = "estimate")
valuation_true <- valuation %>%
  dplyr::select(x) %>%
  dplyr::mutate(type = "true")
valuation_plot <- rbind(valuation_true, valuation_implied)
ggplot(valuation_plot, aes(x = x, color = type)) + stat_ecdf()
```


1.00
0.75
0.50
0.25
0.00
y
0.0
0.3
0.6
0.9
x
type
estimate
true

Chapter 19

Integrating with C++ using Rcpp and RcppEigen

19.1 Prerequisite

- In this chapter, we learn how to integrate C++ using Rcpp and RcppEigen.
- `RcppEigen` is a package to use a linear algebra library `Eigen` with R. The original `Eigen` library and its documentation is found in their website.
- Instead of `RcppEigen`, you may want to use `RcppArmadillo`. `Armadillo` is another libear algebra library in C++.
- We presume that:
  - C/C++ environment is installed to the computer.
    - For OSX, you can install Apple Developer Tools.
    - For Windows, you can try Rtools.
  - `Rcpp` and `RcppEigen` are installed.
  - The project is created in RStudio from `File > New Project > New Directory > R Package using RcppEigen`. In the following, I use the project name `EmpiricalIO` but the name can be as you like.
- We presume the you have the following folder and file structure from the root directory:
  - `main`.
    - `main.R`: all executable statements are written in this file.
  - `R`.
    - `functions.R`: all function definitions in R are written in this file.
  - `src`.
    - `functions.cpp`: all function definitions in C++ are written in this file.
      - It includes:
        ```
        "src/functions.cpp"
        -------------------
        #include <Rcpp.h>
        #include <RcppEigen.h>
        ```
    - Inside `functions.cpp`, avoid using name spaces.
    - `Makevars`: compilation flags for osx/Linux should be written here.
    - `Makevars.win`: compilation flags for Windows should be written here.
  - `inst`.

    * `include`: header files for external libraries in C/C++ are stored here.

- `Makevars/Makevars.win` should be:

  ```
  Makevars
  --------
  PKG_CPPFLAGS = -w -std=c++11 -I../inst/include/ -O3
  ```

  ```
  Makevars.win
  ------------
  PKG_CPPFLAGS = -w -std=c++11 -I../inst/include/ -O3
  ```

  - `-w` is for supprssing some warnings in `Eigen`. If you want to keep warnings shown, this can be removed.
  - `-std=c++11` is for using the latest functionalities of C++ (optional).
  - `-I../inst/include/` is for setting the header path to `../inst/include/` (optional).

19.2 Workflow

- To minimize the likelihood of bugs and the time to edit and debug the code, I recommend you to follw the following workflow.
- This workflow is based on my experience. If you find better workflow, you can overwrite by your own way.

1. Write a procedure in `main/main.R`.

```
# main/main.R
x <- 1:4
y1 <- x^2
```

2. Rewrite the procedure to a function in `main/main.R`.

```
# main/main.R
# compute coefficient-wise square
compute_square <-
  function(x) {
    y <- x^2
    return(y)
  }
```

3. Execute the function in `main/main.R` and check that the output is the same with the output of the original

```
# main/main.R
y2 <- compute_square(x)
max(abs(y1 - y2))
```

```
## [1] 0
```

4. Cut and paste the function to `R/functions.R`.
5. Build the package and load the function from the library.
6. Check if the function can be executed as in step 3.
7. Copy the function to `src/functions.cpp` and comment them out.

   ```
   // src/functions.cpp
   // # compute coefficient-wise square
   // compute_square <-
   ```

```
//   function(x) {
//     y <- x^2
//     return(y)
//   }
```

8. Write function in C++ by translating the copied and pasted R code.
    - The function name should be consistent with the function name in R. I often put `_rcpp` to the end of the name.
    - Put `// [[Rcpp::export]]` above the name of the function if you want to call this function directly from R. Otherwise, the wrapper function to call from R is not created.
    - The class of the inputs and output should be consistent with `Rcpp` objects. This will be explained later.

```
// src/functions.cpp
// # compute coefficient-wise square
// compute_square <-
// [[Rcpp::export]]
Eigen::VectorXd compute_square_rcpp(Eigen::VectorXd x) {
  //   function(x) {
  Eigen::VectorXd y = x.array().square();
  //     y <- x^2
  //     return(y)
  //   }
  return(y);
}
```

9. Check if clean and rebuild work.
    - If there is a compilation error happens, debug until the compilation succeeds.
    - Sometimes, deleting `R/RcppExports.R` and `R/RcppExports.cpp` may be need when re-compile the functions.
10. Clean up the code by eliminating the copied and pasted R code.

```
// src/functions.cpp
// compute coefficient-wise square
// [[Rcpp::export]]
Eigen::VectorXd compute_square_rcpp(Eigen::VectorXd x) {
  Eigen::VectorXd y = x.array().square();
  return(y);
}
```

11. Now by calling the library, you should be able to use the function written in C++ in R. Check if it returns a valid value and if the output is the same as the output of the R function.

```
# main/main.R
y3 <- compute_square_rcpp(x)
max(abs(y2 - y3))
```

```
## [1] 0
```

12. If there is a run-time bug in the C++ function, you may have to use some debugger for C++ to debug the function.
    - In osx, you can debug the C++ function called from R function in the following way.
    - Open the terminal and run R with the debugger `lldb` by typing the following command in the terminal:
      ```
      # terminal
      R -d lldb
      ```
    - Run `main/main.R` by typing the following command in the terminal:

```
# terminal
run -f main/main.R
```

- This should execute the R source code.
- After `library(EmpiricalIO)` is read, stop the process by `Ctrl + C` before the function in question is called. If there is no time gap between them, set `Sys.sleep()` in R to buy some time.
- As you stop the process, set the break point at the function in question by typing in the terminal as:
  ```
  # terminal
  br s -n compute_square_rcpp
  ```
  and then continue the process by typing `c` in the terminal.
- For the rest, refer to the documentation of lldb.
- There is no such an easy way in Windows. You will have to establish an environment in which you can run a cpp file with executable statemtns and call the debugging functions from the file. Then, you can use some debuggers such as `gdb` to debug the functions inside the cpp file.

13. If you need to modify the function, first rewrite the R function and then follow the same step to rewrite the C++ function. Never start the debugging from C++ functions.

19.3 Passing R Objects as Rcpp Objects

- R class corresponds to the following Rcpp class:

R	Rcpp
`logical`	`Logical`
`integer`	`Integer`
`numeric`	`Numeric`
`complex`	`Complex`
`character`	`String`
`Date`	`Date`
`POSIXct`	`Datetime`

- R data structure corresponds to the following Rcpp data structure:

R	Rcpp
`vector`	`Vector`
`matrix`	`Matrix`
`data.frame`	`DataFrame`
`list`	`List`

- For example, a numeric vector in R is passed to `Rcpp::NumericVector` in Rcpp, an integer matrix is to `Rcpp::IntegerMatrix`, and so on.

```
# main/main.R
# numeric vector
x <- rnorm(5)
x
```

```
## [1]  2.18543791 -0.01282801 -0.30530893 -0.58421346  0.77126850
```

```
# numeric matrix
Y <- matrix(rnorm(2*5), nrow = 2)
Y
```

```
##           [,1]       [,2]       [,3]       [,4]      [,5]
## [1,] 2.106189 -0.2612649 -0.7788302 -0.4213451  1.218305
## [2,] 0.412157  2.0737836  1.1315325 -1.0217474 -1.799761
```

- Let's write a C++ function that just receives a numeric vector and returns the numeric vector, and receives a numeric matrix and returns the numeric matrix.

```
// src/functions.cpp
// [[Rcpp::export]]
Rcpp::NumericVector pass_numeric_vector_to_rcpp(Rcpp::NumericVector x) {
  return(x);
}
// [[Rcpp::export]]
Rcpp::NumericMatrix pass_numeric_matrix_to_rcpp(Rcpp::NumericMatrix Y) {
  return(Y);
}
```

- Check if you can pass R objects and get the right result:

```
# main/main.r
x_rcpp <- pass_numeric_vector_to_rcpp(x)
max(abs(x - x_rcpp))
```

```
## [1] 0
```

```
Y_rcpp <- pass_numeric_matrix_to_rcpp(Y)
max(abs(Y - Y_rcpp))
```

```
## [1] 0
```

- **Exercise**: Write functions `pass_integer_vector_to_rcpp`, `pass_integer_matrix_to_rcpp`, `pass_list_to_rcpp`, `pass_data_frame_to_rcpp` that receive an integer vector, list, and data frame and just return themselves.

```
# main/main.r
# integer vector
z <- 1:5
z
```

```
## [1] 1 2 3 4 5
```

```
# integer matrix
W <- matrix(rep(1, 4), nrow = 2)
W
```

```
##      [,1] [,2]
## [1,]    1    1
## [2,]    1    1
```

```
# list
L <- list(x = x, Y = Y, z = z)
L
```

```
## $x
## [1]  2.18543791 -0.01282801 -0.30530893 -0.58421346  0.77126850
##
## $Y
##           [,1]       [,2]       [,3]       [,4]      [,5]
## [1,] 2.106189 -0.2612649 -0.7788302 -0.4213451  1.218305
## [2,] 0.412157  2.0737836  1.1315325 -1.0217474 -1.799761
```

```
##
## $z
## [1] 1 2 3 4 5
```

```
# data frame
D <- data.frame(x1 = rnorm(5), x2 = rnorm(5))
D
```

```
##           x1         x2
## 1 -0.30824994 -1.5570357
## 2  0.01551524  1.9231637
## 3 -0.44231772 -1.8568296
## 4 -1.63800773 -2.1061184
## 5 -0.64140116  0.6976485
```

```
z_rcpp <- pass_integer_vector_to_rcpp(z)
max(abs(z - z_rcpp))
```

```
## [1] 0
```

```
W_rcpp <- pass_integer_matrix_to_rcpp(W)
max(abs(W - W_rcpp))
```

```
## [1] 0
```

```
L_rcpp <- pass_list_to_rcpp(L)
max(abs(unlist(L) - unlist(L_rcpp)))
```

```
## [1] 0
```

```
D_rcpp <- pass_data_frame_to_rcpp(D)
max(abs(D - D_rcpp))
```

```
## [1] 0
```

19.4 Passing R Objects as Eigen Objects

- R data structure corresponds to the following Eigen data structure:

R	Eigen
`vector`	`Eigen::VectorX`
`matrix`	`Eigen::MatrixX`

- If you pass an `integer` vector and matrix, the corresponding Eigen objects are `Eigen::VectorXi` and `Eigen::MatrixXi`.
- If you pass an `numeric` vector and matrix, the corresponding Eigen objects are `Eigen::VectorXd` and `Eigen::MatrixXd`.
- The class of the output can be Eigen class. If you return `Eigen::VectorXd`, `Eigen::MatrixXd`, then they are automatically converted to the corresponding R objects.
- Check if you can pass `x`, `Y`, and `z` as follows:

```
// src/functions.cpp
// [[Rcpp::export]]
Eigen::VectorXd pass_numeric_vector_to_eigen(Eigen::VectorXd x) {
  return(x);
}
```

```
// [[Rcpp::export]]
Eigen::MatrixXd pass_numeric_matrix_to_eigen(Eigen::MatrixXd Y) {
  return(Y);
}
```

```
# main/main.r
x_eigen <- pass_numeric_vector_to_eigen(x)
max(abs(x - x_eigen))
```

```
## [1] 0
```

```
Y_eigen <- pass_numeric_matrix_to_eigen(Y)
max(abs(Y - Y_eigen))
```

```
## [1] 0
```

- **Exercise**: Write functions `pass_integer_vector_to_rcpp` and `pass_integer_matrix_to_rcpp` that receive an integer vector and integer matrix and just return themselves.

```
# main/main.r
z_eigen <- pass_integer_vector_to_eigen(z)
max(abs(z - z_eigen))
```

```
## [1] 0
```

```
W_eigen <- pass_integer_matrix_to_eigen(W)
max(abs(W - W_eigen))
```

```
## [1] 0
```

- If you pass a vector and matrix by `Eigen::VectorX` and `Eigen::MatrixX`, the objects are **copied** to the new objects. This means that the new memory is allocated. If you are going to modify the passed object inside the C++ function, the objects have to be copied. Otherwise, you can just map the objects in the following way so that the new memory is not allocated, whereas you cannot modify the objects in the C++ function.

```
// src/functions.cpp
// [[Rcpp::export]]
Eigen::VectorXd map_numeric_vector_to_eigen(Eigen::Map<Eigen::VectorXd> x) {
  return(x);
}
// [[Rcpp::export]]
Eigen::MatrixXd map_numeric_matrix_to_eigen(Eigen::Map<Eigen::MatrixXd> Y) {
  return(Y);
}
```

```
# main/main.r
x_eigen_map <- map_numeric_vector_to_eigen(x)
max(abs(x - x_eigen_map))
```

```
## [1] 0
```

```
Y_eigen_map <- map_numeric_matrix_to_eigen(Y)
max(abs(Y - Y_eigen_map))
```

```
## [1] 0
```

- I recommend to directly pass R vectors and matrices to `Eigen::VectorX` and `Eigen::MatrixX` rather than to `Vector` and `Matrix` in Rcpp, because `Eigen::VectorX` and `Eigen::MatrixX` have richer methods for linear algebra.

- R list cannot be directly translated to Eigen objects, but the list of vectors and matrices, and the list of list of R objects can be passed to Eigen in the following way.

```
// src/functions.cpp
// [[Rcpp::export]]
Rcpp::List pass_list_to_eigen(Rcpp::List L) {
  // double vector
  Eigen::VectorXd x(Rcpp::as<Eigen::VectorXd>(L.at(0)));
  // double matrix
  Eigen::MatrixXd Y(Rcpp::as<Eigen::MatrixXd>(L.at(1)));
  // integer vector
  Eigen::VectorXi z(Rcpp::as<Eigen::VectorXi>(L.at(2)));
  // integer matrix
  Eigen::MatrixXi W(Rcpp::as<Eigen::MatrixXi>(L.at(3)));
  // return
  Rcpp::List output = Rcpp::List::create(x, Y, z, W);
  return(output);
}
```

```
# main/main.r
list_1 <- list()
list_1[[1]] <- x
list_1[[2]] <- Y
list_1[[3]] <- z
list_1[[4]] <- W
list_1
```

```
## [[1]]
## [1]  2.18543791 -0.01282801 -0.30530893 -0.58421346  0.77126850
##
## [[2]]
##          [,1]       [,2]       [,3]       [,4]      [,5]
## [1,] 2.106189 -0.2612649 -0.7788302 -0.4213451  1.218305
## [2,] 0.412157  2.0737836  1.1315325 -1.0217474 -1.799761
##
## [[3]]
## [1] 1 2 3 4 5
##
## [[4]]
##      [,1] [,2]
## [1,]    1    1
## [2,]    1    1
```

```
list_1_eigen <- pass_list_to_eigen(list_1)
max(abs(unlist(list_1) - unlist(list_1_eigen)))
```

```
## [1] 0
```

- You can also pass a list with named arguments and return a named list as follows:

```
// src/funtions.cpp
// [[Rcpp::export]]
Rcpp::List pass_named_list_to_eigen(Rcpp::List L) {
  // double vector
  Eigen::VectorXd x(Rcpp::as<Eigen::VectorXd>(L["x"]));
  // double matrix
  Eigen::MatrixXd Y(Rcpp::as<Eigen::MatrixXd>(L["Y"]));
```

```
  // integer vector
  Eigen::VectorXi z(Rcpp::as<Eigen::VectorXi>(L["z"]));
  // integer matrix
  Eigen::MatrixXi W(Rcpp::as<Eigen::MatrixXi>(L["W"]));
  // return
  Rcpp::List output = Rcpp::List::create(
    Rcpp::Named("x") = x,
    Rcpp::Named("Y") = Y,
    Rcpp::Named("z") = z,
    Rcpp::Named("W") = W);
  return(output);
}
```

```
# main/main.r
list_2 <- list()
list_2$x <- x
list_2$Y <- Y
list_2$z <- z
list_2$W <- W
list_2
```

```
## $x
## [1]  2.18543791 -0.01282801 -0.30530893 -0.58421346  0.77126850
##
## $Y
##          [,1]       [,2]       [,3]       [,4]      [,5]
## [1,] 2.106189 -0.2612649 -0.7788302 -0.4213451  1.218305
## [2,] 0.412157  2.0737836  1.1315325 -1.0217474 -1.799761
##
## $z
## [1] 1 2 3 4 5
##
## $W
##      [,1] [,2]
## [1,]    1    1
## [2,]    1    1
```

```
list_2_eigen <- pass_named_list_to_eigen(list_2)
max(abs(unlist(list_2) - unlist(list_2_eigen)))
```

```
## [1] 0
```

- You can also access to the column of `Rcpp::DataFrame` in the similar way.

```
// src/functions.cpp
// [[Rcpp::export]]
Eigen::VectorXd extract_column_from_data_frame(Rcpp::DataFrame D) {
  // pass column x1 of D to Eigen::VectorXd named x1
  Eigen::VectorXd x1(Rcpp::as<Eigen::VectorXd>(D["x1"]));
  return(x1);
}
```

```
# main/main.R
x1 <- extract_column_from_data_frame(D)
max(abs(D$x1 - x1))
```

```
## [1] 0
```

- This allows us to pass whatever objects in R to a C++ function.
- If you are planning to translate R functions to C/C++, from the beginning, you should write the R functions in the way inputs and outpus can be passed to C/C++ as above.

19.5 Passing R Objects in Other Objects in C/C++

- `integer` and `numeric` scalars in R can be simply passed to `int` and `double` in C/C++.
- Vectors in R can be passed to `std::vector<int>` or `std::vector<double>` objects. This may be helpful if you want to use the methods for `std::vector`.

19.6 Manipulating Objects in a C++ Function

- As mentioned above, the best practice is to pass vectors and matrices to `Eigen::VectorXd`/`Eigen::VectorXi` and `Eigen::MatrixXd`/`Eigen::MatrixXi` rather than `Rcpp::NumericVector`/`Rcpp::IntegerVector` and `Rcpp::NumericMatrix`/`Rcpp::IntegerMatrix`.
- The other objects will be passed as `Rcpp::DataFrame` or `Rcpp::List` and at the end converted to `Eigen::VectorXd`/`Eigen::VectorXi` and `Eigen::MatrixXd`/`Eigen::MatrixXi` usin `Rcpp::as` as explained above.
- The rest of manipulation will be done using the methods in Eigen. Here I introduce basic operations. For the detail, refer to the online document of Eigen.

19.6.1 Matrix and Vector Arithmetic

- Addition and subtraction:
  - Binary operator + as in `a + b`.
  - Binary operator − as in `a - b`.
  - Unary operator − as in `- a`.
- Scalar multiplication and division:
  - Binary operator × as in `matrix * scalar`.
  - Binary operator × as in `scalar * matrix`.
  - Binary operator / as in `matrix / scalar`.
- Transposition:
  - Transposition as in `matrix.transpose()`.
- Matrix-matrix and matrix-vector multiplication:
  - This part is different from R.
  - Matrix multiplication is as in `A * B`.
  - Coefficientwise multiplication is explained later but is as in `A.array() * B.array()`.
- Following the best practice, first write R function for these operations in `R/functions.R` and run in `main/main.R`:

```
# R/functions.R
# matrix and vector arithmetic
matrix_vector_arithmetic <-
  function(A, B, v, c) {
    # addition and subtraction
    X1 <- A + B
    X2 <- A - B
    X3 <- -A
    # scalar multiplication and division
    X4 <- A * c
```

```
    X5 <- c * A
    X6 <- A / c
    # transpose
    X7 <- t(A)
    # matrix-matrix and matrix-vector multiplication
    X8 <- A %*% t(B)
    X9 <- A %*% v
    # return
    return(list(X1 = X1,
                X2 = X2,
                X3 = X3,
                X4 = X4,
                X5 = X5,
                X6 = X6,
                X7 = X7,
                X8 = X8,
                X9 = X9))
  }
```

```
# main/main.R
set.seed(1)
# addition subtraction
A <- matrix(rnorm(2*4), nrow = 2)
B <- matrix(rnorm(2*4), nrow = 2)
c <- 3
v <- matrix(rnorm(4), nrow = 4)
output <- matrix_vector_arithmetic(A, B, v, c)
```

- Next, write these operations in C++ function:

```
// src/functions.cpp
// [[Rcpp::export]]
Rcpp::List matrix_vector_arithmetic_rcpp(
Eigen::MatrixXd A, Eigen::MatrixXd B,
Eigen::VectorXd v, double c) {
  // addition and subtraction
  Eigen::MatrixXd X1 = A + B;
  Eigen::MatrixXd X2 = A - B;
  Eigen::MatrixXd X3 = - A;
  // scalar multiplication and division
  Eigen::MatrixXd X4 = A * c;
  Eigen::MatrixXd X5 = c * A;
  Eigen::MatrixXd X6 = A / c;
  // transpose
  Eigen::MatrixXd X7 = A.transpose();
  // matrix-matrix and matrix-vector multiplication
  Eigen::MatrixXd X8 = A * B.transpose();
  Eigen::VectorXd X9 = A * v;
  // return
  Rcpp::List output =
Rcpp::List::create(
  Rcpp::Named("X1") = X1,
  Rcpp::Named("X2") = X2,
  Rcpp::Named("X3") = X3,
  Rcpp::Named("X4") = X4,
```

```
      Rcpp::Named("X5") = X5,
      Rcpp::Named("X6") = X6,
      Rcpp::Named("X7") = X7,
      Rcpp::Named("X8") = X8,
      Rcpp::Named("X9") = X9);
      return(output);
    }
```

- Then, we can check if this yields (almost) the same result with the R function:

```
# main/main.R
output_rcpp <- matrix_vector_arithmetic_rcpp(A, B, v, c)
max(abs(unlist(output) - unlist(output_rcpp)))
```

```
## [1] 5.551115e-17
```

- The output looks like:

```
# main/main.R
output_rcpp
```

```
## $X1
##             [,1]      [,2]       [,3]      [,4]
## [1,] -0.05067246 0.6761526 -0.2917328 1.6123600
## [2,] -0.12174506 1.9851240 -3.0351683 0.6933911
##
## $X2
##            [,1]      [,2]      [,3]       [,4]
## [1,] -1.2022352 -2.347410 0.9507484 -0.6375019
## [2,]  0.4890317  1.205438 1.3942315  0.7832583
##
## $X3
##            [,1]       [,2]       [,3]       [,4]
## [1,]  0.6264538  0.8356286 -0.3295078 -0.4874291
## [2,] -0.1836433 -1.5952808  0.8204684 -0.7383247
##
## $X4
##           [,1]      [,2]       [,3]     [,4]
## [1,] -1.879361 -2.506886  0.9885233 1.462287
## [2,]  0.550930  4.785842 -2.4614052 2.214974
##
## $X5
##           [,1]      [,2]       [,3]     [,4]
## [1,] -1.879361 -2.506886  0.9885233 1.462287
## [2,]  0.550930  4.785842 -2.4614052 2.214974
##
## $X6
##             [,1]       [,2]       [,3]      [,4]
## [1,] -0.20881794 -0.2785429  0.1098359 0.1624764
## [2,]  0.06121444  0.5317603 -0.2734895 0.2461082
##
## $X7
##            [,1]       [,2]
## [1,] -0.6264538  0.1836433
## [2,] -0.8356286  1.5952808
## [3,]  0.3295078 -0.8204684
```

```
## [4,]  0.4874291  0.7383247
##
## $X8
##           [,1]       [,2]
## [1,] -1.280368 -0.8861152
## [2,]  3.857726  2.3497425
##
## $X9
## [1] -0.2184706  1.2674165
```

19.6.2 The Array Class and Coefficient-wise Operations

- When you want to have coefficient-wise operations, you first use `.array()` method to convert the object to an array and then apply methods for arrays.
- The resulting array can be assigned to a matrix object implicitly or explicitly by using `.matrix()` method.
- Coefficient-wise multiplication:
  - `A * B` in R is `A.array() * B. array()` in Eigen.
- Other coefficient-wise math functions:
  - Absolute value: `A.array().abs()`.
  - Exponential: `A.array().exp()`.
  - Logarithm: `A.array().log()`.
  - Power: `A.array().power(r)`.
  - For the other methods, refer to the online document.
- Write functions in R in `R/functions.R` and run in `main/main.R`:

```r
# R/functions.R
coefficientwise_operation <-
  function(A, B, r) {
    # coefficient-wise multiplication
    X1 <- A * B
    # other coefficient-wise math functions
    X2 <- abs(A)
    X3 <- exp(A)
    X4 <- log(abs(A))
    X5 <- A^r
    # return
    return(list(
      X1 = X1,
      X2 = X2,
      X3 = X3,
      X4 = X4,
      X5 = X5
    ))
  }
```

```r
# main/main.R
# coefficient-wise operations
r <- 2
output <- coefficientwise_operation(A, B, r)
```

- Write C++ function in `src/functions.cpp` and call from `main/main.R`:

```
// src/functions.cpp
// [[Rcpp::export]]
Rcpp::List coefficientwise_operation_rcpp(
  Eigen::MatrixXd A,
  Eigen::MatrixXd B,
  int r
) {
  // coefficient-wise multiplication
  Eigen::MatrixXd X1 = A.array() * B.array();
  // other coefficient-wise math functions
  Eigen::MatrixXd X2 = A.array().abs();
  Eigen::MatrixXd X3 = A.array().exp();
  Eigen::MatrixXd X4 = A.array().abs().log();
  Eigen::MatrixXd X5 = A.array().pow(r);
  // return
  Rcpp::List output =
    Rcpp::List::create(
      Rcpp::Named("X1") = X1,
      Rcpp::Named("X2") = X2,
      Rcpp::Named("X3") = X3,
      Rcpp::Named("X4") = X4,
      Rcpp::Named("X5") = X5
    );
  return(output);
}
```

```
output_rcpp <- coefficientwise_operation_rcpp(A, B, r)
max(abs(unlist(output) - unlist(output_rcpp)))
```

```
## [1] 2.220446e-16
```

```
output_rcpp
```

```
## $X1
##             [,1]       [,2]       [,3]        [,4]
## [1,] -0.36070042 -1.2632876 -0.2047036  0.54832401
## [2,] -0.05608254  0.6219094  1.8170912 -0.03317559
##
## $X2
##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.6264538 0.8356286 0.3295078 0.4874291
## [2,] 0.1836433 1.5952808 0.8204684 0.7383247
##
## $X3
##           [,1]      [,2]      [,3]     [,4]
## [1,] 0.5344838 0.4336018 1.3902836 1.628125
## [2,] 1.2015872 4.9297132 0.4402254 2.092427
##
## $X4
##            [,1]       [,2]       [,3]       [,4]
## [1,] -0.4676802 -0.1795710 -1.1101553 -0.7186105
## [2,] -1.6947599  0.4670498 -0.1978799 -0.3033716
##
## $X5
```

```
##            [,1]      [,2]      [,3]      [,4]
## [1,] 0.39244438 0.6982752 0.1085754 0.2375871
## [2,] 0.03372487 2.5449208 0.6731684 0.5451234
```

19.6.3 Solving Linear Least Squares Systems

- It is often required to solve a linear least squares system $A \cdot x = b$.
- Solving using SVD decomposition:

```
A.bdcSvd(Eigen::ComputeThinU | Eigen::ComputeThinV).solve(b)
```

- Solving using QR decomposition:

```
A.colPivHouseholderQr().solve(b)
```

- Write R function in `R/functions.R` and run in `main/main.R`:

```r
# R/functions.R
solve_least_squares <-
  function(A, B) {
    x <- solve(A, B)
    return(x)
  }
```

```r
# main/main.R
set.seed(1)
A <- matrix(rnorm(4 * 4), nrow = 4)
B <- matrix(rnorm(4 * 4), nrow = 4)
output <- solve_least_squares(A, B)
```

- Write C++ function in `src/functions.cpp` and call in `main/main.R`:

```
// src/functions.cpp
// solve least squares using SVD decomposition
// [[Rcpp::export]]
Eigen::MatrixXd solve_least_squares_svd(
  Eigen::MatrixXd A,
  Eigen::MatrixXd B
) {
  Eigen::MatrixXd x = A.bdcSvd(Eigen::ComputeThinU | Eigen::ComputeThinV).solve(B);
  return(x);
}
// solve least squares using QR decomposition
// [[Rcpp::export]]
Eigen::MatrixXd solve_least_squares_qr(
    Eigen::MatrixXd A,
    Eigen::MatrixXd B
) {
  Eigen::MatrixXd x = A.colPivHouseholderQr().solve(B);
  return(x);
}
```

```r
# main/main.R
output_svd <- solve_least_squares_svd(A, B)
output_qr <- solve_least_squares_qr(A, B)
max(abs(output - output_svd))
```

```
## [1] 8.881784e-16
```

```
max(abs(output - output_qr))
```

```
## [1] 1.554312e-15
```

```
output_svd
```

```
##             [,1]       [,2]       [,3]       [,4]
## [1,]  0.65530863 -1.2441297 -1.0799835  0.5926519
## [2,] -1.34801870  0.1453720  0.7313845 -2.2353988
## [3,]  1.38750763 -0.3405502 -0.7649107  1.5987206
## [4,] -0.06376266 -0.4632165 -0.2296862  0.4681169
```

```
output_qr
```

```
##             [,1]       [,2]       [,3]       [,4]
## [1,]  0.65530863 -1.2441297 -1.0799835  0.5926519
## [2,] -1.34801870  0.1453720  0.7313845 -2.2353988
## [3,]  1.38750763 -0.3405502 -0.7649107  1.5987206
## [4,] -0.06376266 -0.4632165 -0.2296862  0.4681169
```

19.6.4 Accessing to Elements of a Matrix and Vector

- You can access to the size information and elements of a matrix and vector as follows:
- Sizes: `A.rows()` for row numbers and `A.cols()` for column numbers.
- Element: `A(i, j)` for $(i+1, j+1)$-th element.
- Rows and columns: `A.row(i)` for $i+1$-th row and `A.col(j)` for $j+1$-th column.
- Write R function in `R/functions.R` and run in `main/main.R`:

```
# R/functions.R
access <-
  function(A, i, j) {
    I <- nrow(A)
    J <- ncol(A)
    a_ij <- A[i, j]
    a_i <- A[i, ]
    a_j <- A[, j]
    return(
      list(
        I = I,
        J = J,
        a_ij = a_ij,
        a_i = a_i,
        a_j = a_j
      )
    )
  }
```

```
# main/main.R
output <- access(A, 1, 2)
```

- Write C++ function in `src/functions.cpp` and call in `main/main.R`:

```
// src/functions.cpp
```

```
// [[Rcpp::export]]
Rcpp::List access_rcpp(
    Eigen::MatrixXd A,
    int i,
    int j
  ) {
    int I = A.rows();
    int J = A.cols();
    double a_ij = A(i - 1, j - 1);
    Eigen::VectorXd a_i = A.row(i - 1);
    Eigen::VectorXd a_j = A.col(j - 1);
    Rcpp::List output =
      Rcpp::List::create(
        Rcpp::Named("I") = I,
        Rcpp::Named("J") = J,
        Rcpp::Named("a_ij") = a_ij,
        Rcpp::Named("a_i") = a_i,
        Rcpp::Named("a_j") = a_j
      );
    return(output);
  }
```

```
# main/main.R
output_rcpp <- access_rcpp(A, 1, 2)
max(abs(unlist(output) - unlist(output_rcpp)))
```

```
## [1] 0
```

```
output_rcpp
```

```
## $I
## [1] 4
##
## $J
## [1] 4
##
## $a_ij
## [1] 0.3295078
##
## $a_i
## [1] -0.6264538  0.3295078  0.5757814 -0.6212406
##
## $a_j
## [1]  0.3295078 -0.8204684  0.4874291  0.7383247
```

Bibliography

Daniel A Ackerberg. Timing Assumptions and E¢ ciency: Empirical Evidence in a Production Function Context. Technical report, 2016.

Daniel A. Ackerberg, Kevin Caves, and Garth Frazer. Identification Properties of Recent Production Function Estimators. *Econometrica*, 83(6):2411–2451, November 2015. ISSN 0012-9682. doi: 10.3982/ECTA13408.

S N Afriat. The Construction of Utility Functions from Expenditure Data. *International Economic Review*, 8(1), 1967.

Hyungtaik Ahn and James L. Powell. Semiparametric estimation of censored selection models with a nonparametric selection mechanism. *Journal of Econometrics*, 58(1-2):3–29, July 1993. ISSN 0304-4076. doi: 10.1016/0304-4076(93)90111-H.

Donald W. K. Andrews and Gustavo Soares. Inference for Parameters Defined by Moment Inequalities Using Generalized Moment Selection. *Econometrica*, 78(1):119–157, 2010. ISSN 1468-0262. doi: 10.3982/ECTA7502.

Angus Deaton and John Muellbauer. *Economics and Consumer Behavior*. Cambridge University Press, New York, 1980.

Nikolay Archak, Anindya Ghose, and Panagiotis G. Ipeirotis. Deriving the Pricing Power of Product Features by Mining Consumer Reviews. *Management Science*, 57(8):1485–1509, 2011. ISSN 0025-1909. doi: 10.1287/mnsc.1110.1370.

Peter Arcidiacono and Robert A. Miller. Conditional Choice Probability Estimation of Dynamic Discrete Choice Models with Unobserved Heterogeneity. *Econometrica*, 79(6):1823–1867, 2011. ISSN 0012-9682.

Timothy B Armstrong. Large Market Asymptotics for Differentiated Product Demand Estimators with Economic Models of Supply. *Econometrica*, 84(5):1961–1980, 2016. doi: 10.3982/ECTA10600.

Susan Athey and Philip A. Haile. Identification of Standard Auction Models. *Econometrica*, 70(6):2107–2140, 2002. ISSN 1468-0262. doi: 10.1111/j.1468-0262.2002.00435.x.

Susan Athey and Philip A. Haile. Chapter 60 Nonparametric Approaches to Auctions. In James J. Heckman and Edward E. Leamer, editors, *Handbook of Econometrics*, volume 6, pages 3847–3965. Elsevier, January 2007. doi: 10.1016/S1573-4412(07)06060-6.

Christopher Avery. Strategic Jump Bidding in English Auctions. *Review of Economic Studies*, 65(2):185–210, April 1998. ISSN 0034-6527, 1467-937X. doi: 10.1111/1467-937X.00041.

Patrick Bajari, C. Lanier Benkard, and Jonathan Levin. Estimating Dynamic Models of Imperfect Competition. *Econometrica*, 75(5):1331–1370, 2007. ISSN 0012-9682.

Patrick Bajari, Stephanie Houghton, and Steven Tadelis. Bidding for Incomplete Contracts: An Empirical Analysis of Adaptation Costs. *American Economic Review*, 104(4):1288–1319, April 2014. ISSN 0002-8282. doi: 10.1257/aer.104.4.1288.

Patrick Bayer, Fernando Ferreira, Robert Mcmillan, Joseph Altonji, Pat Bajari, Steve Berry, Sandra Black, David Card, Ken Chay, David Cutler, Hanming Fang, David Figlio, Edward Glaeser, David Lee, Steven Levitt, Enrico Moretti, Tom Nechyba, Jesse Rothstein, Kim Rueben, Holger Sieg, Chris Taber, and Chris Timmins. A Unified Framework for Measuring Preferences for Schools and Neighborhoods. Technical Report 4, 2007.

C. Lanier Benkard, Przemyslaw Jeziorski, and Gabriel Y. Weintraub. Oblivious equilibrium for concentrated industries. *The RAND Journal of Economics*, 46(4):671–708, October 2015. ISSN 07416261. doi: 10.1111/1756-2171.12102.

Steve Berry and Elie Tamer. Identification in Models of Oligopoly Entry. 2006.

Steven Berry and Ariel Pakes. The Pure Characteristics Demand Model. *International Economic Review*, 48(4):1193–1225, December 2007. doi: 10.1111/j.1468-2354.2007.00459.x.

Steven Berry, James Levinsohn, and Ariel Pakes. Automobile Prices in Market Equilibrium. *Econometrica*, 63(4):841–890, 1995. ISSN 1529-2401. doi: 10.1523/JNEUROSCI.4697-10.2011.

Steven Berry, Amit Gandhi, and Philip Haile. Connected Substitutes and Invertibility of Demand. *Econometrica*, 81(5):2087–2111, September 2013. ISSN 0012-9682. doi: 10.3982/ECTA10135.

Steven T. Berry. Estimation of a Model of Entry in the Airline Industry. *Econometrica*, 60(4):889–917, 1992. ISSN 0012-9682. doi: 10.2307/2951571.

Sushil Bikhchandani and John G Riley. Equilibria in open common value auctions. *Journal of Economic Theory*, 53(1):101–130, February 1991. ISSN 0022-0531. doi: 10.1016/0022-0531(91)90144-S.

N. Bloom and J. Van Reenen. Measuring and Explaining Management Practices Across Firms and Countries. *The Quarterly Journal of Economics*, 122(4):1351–1408, November 2007. ISSN 0033-5533. doi: 10.1162/qjec.2007.122.4.1351.

Laura Blow, Martin Browning, and Ian Crawford. Revealed Preference Analysis of Characteristics Models. Technical report, 2008.

Serguey Braguinsky, Atsushi Ohyama, Tetsuji Okazaki, and Chad Syverson. Acquisitions, Productivity, and Profitability: Evidence from the Japanese Cotton Spinning Industry. *American Economic Review*, 105(7):2086–2119, July 2015. ISSN 0002-8282. doi: 10.1257/aer.20140150.

Timothy F. Bresnahan. Departures from marginal-cost pricing in the American automobile industry: Estimates for 1977–1978. *Journal of Econometrics*, 17(2):201–227, November 1981. ISSN 0304-4076. doi: 10.1016/0304-4076(81)90027-0.

Timothy F. Bresnahan. The oligopoly solution concept is identified. *Economics Letters*, 10(1-2):87–92, January 1982. ISSN 0165-1765. doi: 10.1016/0165-1765(82)90121-5.

Timothy F Bresnahan. The Empirical Renaissance in Industrial Economics. *The Journal of Industrial Economics*, 35(4):457–482, 1987.

Timothy F. Bresnahan. Chapter 17 Empirical studies of industries with market power. In *Handbook of Industrial Organization*, volume 2, pages 1011–1057. Elsevier, January 1989. ISBN 978-0-444-70435-1. doi: 10.1016/S1573-448X(89)02005-4.

Timothy F. Bresnahan and Peter C. Reiss. Entry and competition in concentrated markets. *Journal of Political Economy*, 99(5):977, October 1991. ISSN 00223808. doi: 10.1086/261786.

David P Byrne, Susumu Imai, Neelam Jain, Vasilis Sarafidis, and Masayuki Hirukawa. Identification and Estimation of Differentiated Products Models using Market Size and Cost Data. Technical report, 2015.

N Scott Cardell. Variance Components Structures for the Extreme-Value and Logistic Distributions with Application to Models of Heterogeneity. *Econometric Theory*, 13:185–213, 1997. doi: 10.1017/S0266466600005727.

Xiaohong Chen. Chapter 76 Large Sample Sieve Estimation of Semi-Nonparametric Models. *Handbook of Econometrics*, 6:5549–5632, January 2007. ISSN 1573-4412. doi: 10.1016/S1573-4412(07)06076-X.

Clayton M. Christensen. *The Innovator's Dilemma : When New Technologies Cause Great Firms to Fail*. HarperBusiness, New York, 1997. ISBN 1-4221-9602-X.

Wilbur Chung and Juan Alcácer. Knowledge Seeking and Location Choice of Foreign Direct Investment in the United States. *Management Science*, 48(12), 2002. doi: 10.1287/mnsc.48.12.1534.440.

Federico Ciliberto and Elie Tamer. Market Structure and Multiple Equilibria in Airline Markets. *Econometrica*, 77(6):1791–1828, 2009. ISSN 1468-0262. doi: 10.3982/ECTA5368.

Flavio Cunha, James J. Heckman, and Susanne M. Schennach. Estimating the Technology of Cognitive and Noncognitive Skill Formation. *Econometrica*, 78(3):883–931, May 2010. ISSN 0012-9682. doi: 10.3982/ECTA6551.

Jan De Loecker. Product Differentiation, Multiproduct Firms, and Estimating the Impact of Trade Liberalization on Productivity. *Econometrica*, 79(5):1407–1451, September 2011. ISSN 0012-9682. doi: 10.3982/ECTA7617.

Angus Deaton and John Muellbauer. An Almost Ideal Demand System. Technical Report 3, 1980.

Stephen G. Donald and Harry J. Paarsch. Piecewise Pseudo-Maximum Likelihood Estimation in Empirical Models of Auctions. *International Economic Review*, 34(1):121–148, 1993. ISSN 0020-6598. doi: 10.2307/2526953.

Stephen G. Donald and Harry J. Paarsch. Identification, Estimation, and Testing in Parametric Empirical Models of Auctions within the Independent Private Values Paradigm. *Econometric Theory*, 12(3):517–567, 1996. ISSN 0266-4666.

U. Doraszelski and J. Jaumandreu. R&D and Productivity: Estimating Endogenous Productivity. *The Review of Economic Studies*, 80(4):1338–1383, October 2013. ISSN 0034-6527. doi: 10.1093/restud/rdt011.

Ulrich Doraszelski and Mark Satterthwaite. Computable Markov-perfect industry dynamics. *The RAND Journal of Economics*, 41(2):215–243, 2010. ISSN 0741-6261.

Jean-Pierre Dubé, Jeremy T Fox, and Che-Lin Su. Improving the Numerical Performance of Static and Dynamic Aggregate Discrete Choice Random Coefficients Demand Estimation. *Econometrica*, 80(5):2231–2267, 2012. doi: 10.3982/ECTA8585.

Richard Ericson and Ariel Pakes. Markov-Perfect Industry Dynamics: A Framework for Empirical Work. *The Review of Economic Studies*, 62(1):53–82, 1995. ISSN 0034-6527. doi: 10.2307/2297841.

Joseph Farrell and Carl Shapiro. American Economic Association Horizontal Mergers: An Equilibrium Analysis. *Source: The American Economic Review*, 80(1):107–126, 1990.

Amit Gandhi and Jean-François Houde. Measuring Substitution Patterns in Differentiated Products Industries. 2015.

Amit Gandhi, Salvador Navarro, David Rivers, Dan Ackerberg, Richard Blundell, Juan Esteban Carranza, Allan Collard-Wexler, Ulrich Do-Raszelski, Steven Durlauf, Jeremy Fox, Silvia Goncalves, Phil Haile, Jinyong Hahn, Joel Horowitz, Jean-Francois Houde, Lance Lochner, Aureo De Paula, Amil Petrin, Mark Roberts, Nicolas Roys, Todd Stinebrickner, Chad Syverson, Chris Taber, and Quang Vuong. On the Identification of Gross Output Production Functions. Technical report, 2017.

Nicola Gennaioli, Rafael La Porta, Florencio Lopez-de-Silanes, and Andrei Shleifer. Human Capital and Regional Development. *The Quarterly Journal of Economics*, 128(1):105–164, February 2013. doi: 10.1093/qje/qjs050.

Matthew Gentzow. Valuing new goods in a model with complementarity : Online newspapers. Technical Report 3, 2004.

W M Gorman. A Possible Procedure for Analysing Quality Differentials in the Egg Market. Technical Report 5, 1980.

Gautam Gowrisankaran, Aviv Nevo, and Robert Town. Mergers When Prices Are Negotiated: Evidence from the Hospital Industry. *American Economic Review*, 105(1):172–203, January 2015. ISSN 0002-8282. doi: 10.1257/aer.20130223.

Zvi Griliches and Jacques Mairesse. Production Functions: The Search for Identification. In Steinar Strom, editor, *Econometrics and Economic Theory in the Twentieth Century: The Ragnar Frisch Centennial Symposium*. Cambridge University Press, Cambridge, MA, 1998. doi: 10.3386/w5067.

Emmanuel Guerre, Isabelle Perrigne, and Quang Vuong. Optimal Nonparametric Estimation of First-price Auctions. *Econometrica*, 68(3):525–574, 2000. ISSN 1468-0262. doi: 10.1111/1468-0262.00123.

Philip A. Haile and Elie Tamer. Inference with an Incomplete Model of English Auctions. *Journal of Political Economy*, 111(1):1–51, February 2003. ISSN 0022-3808. doi: 10.1086/344801.

Jessie Handbury and David E Weinstein. Goods Prices and Availability in Cities. *The Review of Economic Studies*, 82(1):258–296, 2014. ISSN 0034-6527. doi: 10.1093/restud/rdu033.

Jonathan E Haskel, Sonia C Pereira, and Matthew J Slaughter. Does Inward Foreign Direct Investment Boost the Productivity of Domestic Firms? *Review of Economics and Statistics*, 89(3):482–496, August 2007. ISSN 0034-6535. doi: 10.1162/rest.89.3.482.

Hausman, Leonard, and Zona. Competitive Analysis with Differenciated Products. *Annales d'Économie et de Statistique*, (34):159–180, 1994. ISSN 0769489X. doi: 10.2307/20075951.

Ken Hendricks and Robert H. Porter. Chapter 32 An Empirical Perspective on Auctions. In M. Armstrong and R. Porter, editors, *Handbook of Industrial Organization*, volume 3, pages 2073–2143. Elsevier, January 2007. doi: 10.1016/S1573-448X(06)03032-9.

B Y T Homas J H Holmes. The Diffusion of Wal-Mart and Economies of Density. *Econometrica*, 79(1): 253–302, 2011. ISSN 0012-9682. doi: 10.3982/ecta7699.

V. Joseph Hotz and Robert A. Miller. Conditional Choice Probabilities and the Estimation of Dynamic Models. *The Review of Economic Studies*, 60(3):497–529, 1993. ISSN 0034-6527. doi: 10.2307/2298122.

H S Houthakker. Revealed Preference and the Utility Function Author. *Economica*, 17(66):159–174, 1950.

Cheng Hsiao. Chapter 4 Identification. *Handbook of Econometrics*, 1:223–283, January 1983. ISSN 1573-4412. doi: 10.1016/S1573-4412(83)01008-9.

Chang-Tai Hsieh and Peter J. Klenow. Misallocation and Manufacturing TFP in China and India. *Quarterly Journal of Economics*, 124(4):1403–1448, November 2009. doi: 10.1162/qjec.2009.124.4.1403.

Hidehiko Ichimura and Petra E. Todd. Chapter 74 Implementing Nonparametric and Semiparametric Estimators. *Handbook of Econometrics*, 6:5369–5468, January 2007. ISSN 1573-4412. doi: 10.1016/S1573-4412(07)06074-6.

Bar Ifrach and Gabriel Y. Weintraub. A Framework for Dynamic Oligopoly in Concentrated Industries. *The Review of Economic Studies*, 84:1106–1150, September 2017. ISSN 0034-6527. doi: 10.1093/restud/rdw047.

Mitsuru Igami. Estimating the Innovator's Dilemma: Structural Analysis of Creative Destruction in the Hard Disk Drive Industry, 1981–1998. *Journal of Political Economy*, 125(3):798–847, June 2017. ISSN 0022-3808. doi: 10.1086/691524.

Mitsuru Igami, Takuo Sugaya, We Thank, Joseph Harrington, John Asker, Robert Porter, Michael Whinston, Michihiro Kandori, Philip Haile, Steven Berry, Fiona Scott Morton, Alvin Klevorick, Matthew Weinberg, Jean-François Houde, and Louis Kaplow. Measuring the Incentive to Collude: The Vitamin Cartels, 1990-1999. 2018.

Marc Ivaldi and Frank Verboven. Quantifying the effects from horizontal mergers in European competition policy. *International Journal of Industrial Organization*, 23(9-10):669–691, December 2005. ISSN 0167-7187. doi: 10.1016/J.IJINDORG.2005.08.004.

Dale W. Jorgenson. Chapter 31 Econometric methods for modeling producer behavior. In *Handbook of Econometrics*, volume 3, pages 1841–1915. Elsevier, January 1986. ISBN 978-0-444-86187-0. doi: 10.1016/S1573-4412(86)03011-8.

Kenneth L. Judd. *Numerical Methods in Economics*. MIT Press, 1998. ISBN 0-262-10071-1.

Hiroyuki Kasahara and Katsumi Shimotsu. Nonparametric Identification of Finite Mixture Models of Dynamic Discrete Choices. *Econometrica*, 77(1):135–175, 2009. ISSN 1468-0262. doi: 10.3982/ECTA6763.

Elena Krasnokutskaya. Identification and Estimation of Auction Models with Unobserved Heterogeneity. *The Review of Economic Studies*, 78(1):293–327, January 2011. ISSN 1467-937X, 0034-6527. doi: 10.1093/restud/rdq004.

Vijay Krishna. *Auction Theory*. Academic Press, September 2009. ISBN 978-0-08-092293-5.

Kelvin J. Lancaster. A New Approach to Consumer Theory. *Journal of Political Economy*, 74(2):132–157, April 1966. ISSN 0022-3808. doi: 10.1086/259131.

James Levinsohn and Amil Petrin. Estimating Production Functions Using Inputs to Control for Unobservables. *Review of Economic Studies*, 70(2):317–341, April 2003. ISSN 0034-6527. doi: 10.1111/1467-937X.00246.

Thierry Magnac and David Thesmar. Identifying Dynamic Discrete Decision Processes. *Econometrica*, 70 (2,):801–816, 2002.

Charles F. Manski. Chapter 43 Analog estimation of econometric models. *Handbook of Econometrics*, 4: 2559–2582, January 1994. ISSN 1573-4412. doi: 10.1016/S1573-4412(05)80012-1.

Jacob Marschak and William H Andrews. Random Simultaneous Equations and the Theory of Production Author. Technical Report 4, 1944.

Eric Maskin and Jean Tirole. A Theory of Dynamic Oligopoly, I: Overview and Quantity Competition with Large Fixed Costs. Technical Report 3, 1988a.

Eric Maskin and Jean Tirole. A Theory of Dynamic Oligopoly, I: Overview and Quantity Competition with Large Fixed Costs. *Econometrica*, 56(3):549, May 1988b. ISSN 00129682. doi: 10.2307/1911700.

Rosa L. Matzkin. Nonparametric and Distribution-Free Estimation of the Binary Threshold Crossing and The Binary Choice Models. *Econometrica*, 60(2):239–270, 1992. ISSN 0012-9682. doi: 10.2307/2951596.

Rosa L. Matzkin. Chapter 42 Restrictions of economic theory in nonparametric methods. *Handbook of Econometrics*, 4:2523–2558, January 1994. ISSN 1573-4412. doi: 10.1016/S1573-4412(05)80011-X.

Rosa L. Matzkin. Chapter 73 Nonparametric identification. *Handbook of Econometrics*, 6:5307–5368, January 2007. ISSN 1573-4412. doi: 10.1016/S1573-4412(07)06073-4.

Marjorie B. McElroy. Additive General Error Models for Production, Cost, and Derived Demand or Share Systems. *Journal of Political Economy*, 95(4):737–757, August 1987. ISSN 0022-3808. doi: 10.1086/261483.

Daniel McFadden and Kenneth Train. Mixed MNL models for discrete response. *Journal of Applied Econometrics*, 15(5):447–470, 2000. ISSN 08837252. doi: 10.1002/1099-1255(200009/10)15:5<447::aid-jae570>3.3.co;2-t.

Daniel Mcfadden, John Geweke, Vassilis Hajivassiliou, Reed Johnson, Tony Lancaster, Peter Rossi, Kenneth Train, and Steve Waters. Computing Willingness-to-pay in Random Utility Models. Technical report, 1995.

Daniel L Mcfadden. Conditional Logit Analysis of Qualitative Choice Behavior. In P. Zarembka, editor, *Frontiers in Econometrics*. Academic Press, New York, 1974.

Paul R. Milgrom. Rational Expectations, Information Acquisition, and Competitive Bidding. *Econometrica*, 49(4):921–943, 1981. ISSN 0012-9682. doi: 10.2307/1912511.

Paul R. Milgrom and Robert J. Weber. A Theory of Auctions and Competitive Bidding. *Econometrica*, 50 (5):1089–1122, 1982. ISSN 0012-9682. doi: 10.2307/1911865.

Nathan H. Miller and Matthew C. Weinberg. Understanding the Price Effects of the MillerCoors Joint Venture. *Econometrica*, 85(6):1763–1791, 2017. ISSN 0012-9682. doi: 10.3982/ecta13333.

John Muellbauer. Household Production Theory, Quality, and the "Hedonic Technique. *The American Economic Review*, 64(6):977–994, 1974.

John Muellbauer. Community Preferences and the Representative Consumer Author. *Econometrica*, 44(5): 979–999, 1976.

Richard F Muth. Household Production and Consumer Demand Functions. Technical Report 3, 1966.

Aviv Nevo. Identification of the Oligopoly Solution Concept in a Differentiated-Products Industry. *Economics Letters*, 59(3):391–395, June 1998. ISSN 0165-1765. doi: 10.1016/S0165-1765(98)00061-5.

Aviv Nevo. Mergers with differentiated products: The case of the ready-to-eat cereal industry. Technical Report 3, 2000.

Aviv Nevo. Measuring Market Power in the Ready-to-Eat Cereal Industry. *Econometrica*, 69(2):307–342, March 2001. ISSN 0012-9682. doi: 10.1111/1468-0262.00194.

G Steven Olley and Ariel Pakes. The Dynamics of Productivity in the Telecommunications Equipment Industry. Technical Report 6, 1996.

Martin Pesendorfer and Philipp Schmidt-Dengler. Asymptotic Least Squares Estimators for Dynamic Games. *Review of Economic Studies*, 75(3):901–928, July 2008. ISSN 0034-6527, 1467-937X. doi: 10.1111/j.1467-937X.2008.00496.x.

Richard Stone. *The Measurement of Consumers' Expenditure and Behaviour in the United Kingdom 1920-1938, Vol. 1.* Cambridge University Press, Cambridge, 1954.

Marcel K Richter. Revealed Preference Theory. *Econometrica*, 34(3), 1966.

Sherwin Rosen. Hedonic Prices and Implicit Markets: Product Differentiation in Pure Competition. *Journal of Political Economy*, 82(1):34–55, 1974.

John Rust. Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher. *Econometrica*, 55(5):999, September 1987. ISSN 00129682. doi: 10.2307/1911259.

Stephen P. Ryan. The Cost of Environment Regulation in a Concentrated Industry. *Econometrica*, 80(3): 1019–1061, 2012. ISSN 0012-9682.

Marc Rysman. Competition Between Networks: A Study of the Market for Yellow Pages. Technical report, 2004.

P A Samuelson. A Note on the Pure Theory of Consumer's Behaviour. *Economica*, 5(17):61–71, 1938.

Kenneth A Small and Harvey S Rosen. Applied Welfare Economics with Discrete Choice Models. *Econometrica*, 49(1):105–130, 1981.

Howard Smith. Supermarket Choice and Supermarket Competition in Market Equilibrium. *Review of Economic Studies*, 71(1):235–263, January 2004. ISSN 0034-6527. doi: 10.1111/0034-6527.00283.

John Sutton. Chapter 35 Market Structure: Theory and Evidence. In M. Armstrong and R. Porter, editors, *Handbook of Industrial Organization*, volume 3, pages 2301–2368. Elsevier, January 2007. doi: 10.1016/S1573-448X(06)03035-4.

Chad Syverson. What Determines Productivity? *Journal of Economic Literature*, 49(2):326–365, June 2011. ISSN 0022-0515. doi: 10.1257/jel.49.2.326.

Hidenori Takahashi. Strategic design under uncertain evaluations: Structural analysis of design-build auctions. *The RAND Journal of Economics*, 49(3):594–618, 2018. ISSN 1756-2171. doi: 10.1111/1756-2171.12246.

Elie Tamer. Incomplete Simultaneous Discrete Response Model with Multiple Equilibria. *Review of Economic Studies*, 70(1):147–165, January 2003. ISSN 0034-6527, 1467-937X. doi: 10.1111/1467-937X.00240.

Kenneth Train. *Discrete Choice Methods with Simulation*. Cambridge University Press, Cambridge, 2009.

Hal R Varian. The Nonparametric Approach to Demand Analysis. *Econometrica*, 50(4):945–973, 1982.

Gabriel Y. Weintraub, C. Lanier Benkard, and Benjamin Van Roy. Markov Perfect Industry Dynamics With Many Firms. *Econometrica*, 76(6):1375–1411, 2008. ISSN 1468-0262. doi: 10.3982/ECTA6158.

G. Werden and L. Frobe. The Effects of Mergers in Differentiated Products Industries : Logit Demand and Merger Policy. *Journal of Law, Economics, and Organization*, 10:407–426, 1993. doi: 10.1093/oxfordjournals.jleo.a036857.

Jeffrey M Wooldridge. On estimating firm-level production functions using proxy variables to control for unobservables. *Economics Letters*, 104:112–114, 2009. doi: 10.1016/j.econlet.2009.04.026.