Contents

1. PROBLEM STATEMENT ............................................................................................................................................................. 2
   1a. Definitions.................................................................................................................................................................................. 2
   1b. Boundaries for our Analysis (“In-Scope / Out-of-Scope”)................................................................................................. 2

2. PARTNERSHIP WITH SANDIA NATIONAL LABORATORIES................................................................................................ 4
   2a. Sandia’s mission...................................................................................................................................................................... 4
   2b. CASoS description and mission ............................................................................................................................................ 4
   2c. Stanford International Policy Studies Program.................................................................................................................. 4

3. ASSESSMENT OF THE U.S. RETAIL FINANCIAL SYSTEM ................................................................................................. 5
   3a. Structural Overview.............................................................................................................................................................. 5
   3b. Two Important Descriptive Facts about the Retail Financial System .................................................................................. 8
   3c. Historical Role of the U.S. Government................................................................................................................................. 9

4. OUR MODEL................................................................................................................................................................................11
   4a. Consumers and Merchants.................................................................................................................................................... 11
   4b. Platforms.................................................................................................................................................................................. 13

5. MODEL CALIBRATION: SIMULATED METHOD OF MOMENTS.......................................................................................... 16

6. FINDINGS .....................................................................................................................................................................................17
   6a. Simulation #1: Platform Specialization.................................................................................................................................. 17
   6b. Simulation #2: Risk-Affordability Tradeoff.......................................................................................................................... 18

7. CONCLUSION ...............................................................................................................................................................................19
   7a. Potential Extensions................................................................................................................................................................. 19
   7b. Value of Computational Modeling in Policy Settings..................................................................................................... 19

REFERENCES ................................................................................................................................................................................... 20

APPENDIX: MODEL CODE .......................................................................................................................................................... 23
   Part 1: Functions........................................................................................................................................................................... 23
   Part 2: Simulated Method of Moments .................................................................................................................................... 34
   Part 3: Evolutionary Algorithm............................................................................................................................................... 40
1. PROBLEM STATEMENT

Our goal for this project was twofold. First, we sought to identify the determinants of security and affordability in the U.S. retail financial system, as these two facets represent a dual objective for any country’s financial system. Second, with the key determinants in hand, we sought to build a computational model that could serve as a policy-focused tool for understanding the economic dynamics and tradeoff between security and affordability in the retail financial system.

1a. Definitions

**Security:** Our definition of security encompasses both protection from breach as well as what might traditionally be defined as reliability. A secure financial system is one with infrequent and inconsequential data theft, illegal use of data, fraudulent activity, illegal use of payment systems (often by nefarious actors), and infrequent and inconsequential failed transactions for operational reasons. A large number of these incidents or a few with large magnitude can degrade confidence in a financial system and steer economic participants to other, less efficient means of payment.

**Affordability:** An affordable financial system is one in which transaction costs are not prohibitive and thus allow broad access for all those that value the service. Because of the reach of financial systems into all corners of the economy, as opposed to any other product or service, affordability should be a key goal of government.

1b. Boundaries for our Analysis ("In-Scope / Out-of-Scope")

**In Scope**

- **The U.S. retail financial system.** This system is defined by the transfer of funds (cash or credit) in the economy among participants/counterparties and along an infrastructure for initiation, authorization, transfer, clearing, and settlement. Notably, the retail financial system does not include the transfer of financial assets.

- **Security and affordability (broadly).** We acknowledge that each of these contain a broad set of issues. We could easily find a narrow issue worthy of a 20-week investigation (e.g. how to regulate interchange fees between acquiring and issuing banks). Rather, we are choosing to abstract away from many specifics in order to investigate the macro-level tradeoffs in governing a total financial payment ecosystem.
Out of scope

- **Clearing & settlement systems for:**
  - **Large-value transactions.** These systems are used by financial institutions to make large-dollar, time-critical transfers. The Fedwire (primarily operated by the Federal Reserve) and CHIPS (operated by the Clearing House Interbank Payments Company) and are the two major large value systems. These systems lack the interesting dynamics we sought to analyze, as they operate more as public utilities than as complex adaptive systems.
  - **Securities.** These systems process clearing & settlement for any security trade in products such as government bonds or mortgage-backed securities. Participants in these markets include securities issuers, intermediaries such as brokers, dealers, and depository institutions, and investors. The National Securities Clearing Corporation (NSCC) clears most non-government securities trades, while the Depository Trust Company (DTC) processes settlement.

- **Non-US countries’ own financial systems or international links to the U.S. system.** Sandia undertakes scientific endeavors that advance the interests of the United States. Due to our partnership with Sandia, we've decided to take a narrow reading of Sandia's mission that does not consider the development of other countries financial systems as a primary U.S. interest. Yet, because our field of study explicitly focuses on international policy, the demonstrability of our analysis for the retail financial systems of other countries remained at least a peripheral interest.

- **Specific business plans for the retail financial system.** While our analysis arguably unearthed some interesting business strategies, any such insights were purely incidental.
2. PARTNERSHIP WITH SANDIA NATIONAL LABORATORIES

2a. Sandia’s mission

Sandia National Laboratories was founded during World War II to help the United States produce the first nuclear weapons. In the post-Cold War world, Sandia, as a center for scientific investigation, has been called on to advance the general national interest in a variety of other fields including counterterrorism, energy security, and infrastructure security. The financial payments system is an example of a critical infrastructure.

2b. CASoS description and mission

The Complex Adaptive Systems of Systems (CASoS) Engineering Initiative sits within Sandia, and was born from an analysis of technical systems with one or more of six major goals: to (1) predict, (2) prevent or cause, (3) prepare, (4) monitor, (5) recover or change, and (6) control a system. The analytic approach is rooted in theory and experimentation, but importantly makes use of simulated environments to understand the dynamics of a system. Many of the most pressing issues of our time — such as large-scale natural disasters, pandemics, global finance, global economic supply chains, and global climate change — are socio-technical systems that exhibit characteristics suitable to the CASoS approach. These challenges impel complex policy decisions, and the CASoS initiative is defined by its goal of informing such policy. Walter Beyeler leads the CASoS program.

2c. Stanford International Policy Studies Program

Stanford’s program on International Policy Studies is a two-year Master’s program to train future policymakers across a variety of fields including democratic development, national defense, and -- for the three members of this team -- international political economy. Because the program is generally a professional program, our project will be geared to practical recommendations rather than theoretical insights.
3. ASSESSMENT OF THE U.S. RETAIL FINANCIAL SYSTEM

3a. Structural Overview

Major Participants

The U.S.’s retail financial payments system – or any country’s payment system – is essentially composed of three types of parties: (1) end users, (2) settlement systems, and (3) banks.

(1) **End users** include consumers wishing to send money to each other or pay for a good or service. End users also include merchants that are on the receiving end of payment flows for many transactions. In any transaction, a starting and ending user is needed; these could both be merchants / businesses (e.g. one firm sending payment to another firm higher up the supply chain), both consumers (e.g. transferring money among family members), or one of each (e.g. a consumer paying a firm for a final product).

(2) **Settlement services (referred to in this report as “platforms”)** comprise the various firms and infrastructure that transfer the payment from one party to the other. The components include a messaging standard for electronic payments and a risk-bearing arrangement for the time that payment is “in transit.” Perhaps most critically for our project, these platforms are also described by their revenue plan in performing the transaction service.

(3) **Banks** engage in the transaction chain for two primary reasons: (1) it is they that hold the relationship and establish infrastructure with the platforms, and (2) it is they that hold the vast majority of money or credit in the economy on behalf of end users. Even when end users do not have an explicit banking relationship (as shown below in Type 2), an interfacing service with a bank relationship serves as an additional transacting party.
Transaction Flow

A generic transaction flow is described below in Figure 1, as “Type 1.” Two common alternatives are presented as Type 2 and Type 3. In Type 2, the end users do not hold a bank account. This type of flow is common in developing countries. Type 3 represents a new end-user interfacing platform (e.g. PayPal or Google Wallet) that provides additional services as its value add to the consumer, but still must operate within the usual infrastructure.

Figure 1

(2) Authors ‘own diagram
Revenue Flow

The flow of revenue, i.e. the money earned or paid by each party for performing or requesting a service, is slightly more complex. Typically, the cost of processing is born by the receiving end party, with each intermediary able to extract different shares of the revenue depending on their position in the transaction. Notably, initiating end users often pay a “negative cost” for processing, i.e. they receive benefits just for their membership with a certain bank or platform.

Figure 2

![Value Flow in a Typical $100 Transaction](image)

(2) Authors ‘own derived statistics and diagram
Distribution of Value and Volume

Millions of consumers and merchants utilize a dizzying set of banks and platforms in order to process transactions. There are many interesting characteristics of this usage related to demographic variables. For our report, however, there are two primary statistics of interest: the distribution of *value* and the distribution of *volume* across three major platforms, only considering consumer-to-merchant transactions (C2M). A brief summary of this is contained in Figure 3.

**Figure 3**

![Figure 3: Distribution of Value and Volume](image)

*Sources: (1) Voorhies, et al. (2013) Fighting poverty, profitably: Transforming the economics of payments to build sustainable, inclusive finance systems. Bill & Melinda Gates Foundation. (2) Authors' own derived statistics and diagram*

3b. Two Important Descriptive Facts about the Retail Financial System

(i) Platform specialization

Platforms can earn money from four different revenue streams:

- **Account fees** are annual or monthly fees that consumers or merchants are charged for having an account with the provider.
- **Cash-In-Cash-Out fees** in the US are most commonly associated with ATM usage fees.
- **Transaction-based fees** are fees charged per transaction on top of any fixed fees. They can be either a flat fee for a transaction (like a wire transfer fee), or a percentage of the transaction amount (like PayPal’s 2% commission).
• **Adjacencies** are other revenues that platforms can earn from their users. The largest source of adjacency revenue in the US is the interest that banks earn from lending deposits.

Traditional platforms within the U.S., such as established banks, earn most of their revenue from Adjacencies, and don’t charge for the other three services. This means that they target customers with high balances (i.e. the rich) to maximize their revenues.

The other main type of platform is one that earns most of their revenue from transaction fees. These platforms do not require any minimum balance, and often end up servicing the poor.

This results in a system where the rich have access to specialized platforms that charge them few fees and are able to use the system at low cost, while the poor use a different set of platforms with high fees.

(ii) Risk-Affordability Trade-off

There are two basic goals to any financial system: security and affordability. However, this inherently implies a trade-off between the two, since investments in security are expensive. We can have a secure, well-regulated, expensive and exclusive system, or a risky, unregulated, affordable, inclusive system. As the government is responsible for setting minimum security investment requirements, it is important to know exactly what will have to be given up. Currently there is no data on exactly what those tradeoffs look like, or guidelines on how to choose an optimal point.

3c. Historical Role of the U.S. Government

A number of federal and self-established regulatory bodies are involved in overseeing the operation of the U.S. payments system. The Federal Reserve, as steward, serves the most prominent role. The Fed has traditionally identified five major risks to the retail financial system that it monitors and mitigates:

**The Federal Reserve’s risks to the retail financial system:**

1. Fraud
2. Operational (technical or human system failures)
3. Legal (contract breaches)
4. Settlement (insolvency or illiquidity)
5. Systemic (large failures or propagation of failures)

Pertaining to mitigation strategies for each of these risks, the Federal Reserve pulls on a number of policy levers related to:

- Standards for security investments
- Consumer protection (privacy, anti-predatory regulations)
- Anti-monopoly policies
- Operations of central clearinghouses
- Price regulation (e.g. interchange fees)
- Prudential regulation (e.g. capital requirements)
4. OUR MODEL

Our model of the retail payment system includes, broadly speaking, consumers and merchants that make economically informed decisions to belong to certain payment platforms. These platforms then calculate their profit, with the most successful platform business models being replicated and sustained. The exogenous characteristics of consumers, merchants, and platforms (e.g. average income, or average transaction fee) are based on U.S. empirical data. A more detailed discussion of our model follows.

### 4a. Consumers and Merchants

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Description</th>
<th>Distribution</th>
<th>Consumer</th>
<th>Merchant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Income</td>
<td>Annual income</td>
<td>~Pareto</td>
<td>Avg. Income = $53k&lt;br&gt;Consumer Gini Coefficient = 0.477</td>
<td>Merchant Gini Coefficient = 0.927</td>
</tr>
<tr>
<td>Transaction Frequency</td>
<td>Number of economic transactions per time period</td>
<td>~Gamma</td>
<td>Mean = 79 / month</td>
<td>Determined from consumer transactions</td>
</tr>
<tr>
<td>Transaction Value</td>
<td>Monetary value for each economic transaction</td>
<td>(Income*Budget) / Transaction frequency</td>
<td>X</td>
<td>Determined from consumer transactions</td>
</tr>
<tr>
<td>Budget</td>
<td>% of income used for spending</td>
<td>~Uniform (min=0,max=1)</td>
<td>X</td>
<td>NA</td>
</tr>
<tr>
<td>Expenditures</td>
<td>Amount of income that is sent through the retail payment system in some form</td>
<td>Income * Budget</td>
<td>X</td>
<td>NA</td>
</tr>
<tr>
<td>Savings</td>
<td>Amount of income that is saved; used to meet minimum balance requirements for platforms</td>
<td>Income * (1-Budget)</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Price sensitivity</td>
<td>Multiplier to (dis)utility derived from the price of a payment method</td>
<td>Estimated via Sim. Method of Moments</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Risk sensitivity</td>
<td>Multiplier to (dis)utility derived from risk of fraud for using a certain method of payment</td>
<td>Estimated via Sim. Method of Moments</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Popularity sensitivity</td>
<td>Multiplier from the popularity of a payment method; the more people that accept a payment, the more valuable it is</td>
<td>Estimated via Sim. Method of Moments</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Sources:  
Choice Functions

Consumers and Merchants dynamically decide each time period which platforms they would like to use, and move to those platforms.

Defining Consumer and Merchant Actions

1. At the beginning of each time period, consumers evaluate their utility of all platforms.
2. Consumers and Merchants decide whether to stay with their current platform based on how happy they are. They always select the platform with maximal utility, and then also select any other platform within a certain range of that maximal utility. The additional platform selection is to capture the process of “multi-homing” in which users may have a primary payment method, but certainly secondary and even tertiary payment methods. For the sake of creating a model closer to reality, the utility evaluation includes a stochastic term that accounts for other factors.
3. If the platform they are currently using has gone bankrupt, all consumers and merchants will automatically move to a new platform.

Defining the Utility Function

Consumer Platform utility is a function with the form:

\[
Utility_{\text{Platform}} = access \times (B1 \times price - B2 \times risk + B3 \times popularity)
\]

Merchant Platform utility is a function with the form:

\[
Utility_{\text{Merchant}} = B4 \times price - B5 \times risk + B6 \times popularity
\]

Where the components of the function are defined as:

1. Popularity is the percent of merchants or consumers who use the payment platform. Consumers are only concerned with how many merchants are using the platform, and merchants are similarly only concerned with how many consumers are using the platform. If a platform is unpopular, even incredibly low prices will fail to entice consumers or merchants to use the payment system, since they won't be able to use it to complete any transactions. As the platform becomes more popular, individuals will be willing to pay higher prices to be able to use the platform. Once a critical mass is reached, enough merchants or consumers accept a payment system that an
individual can reasonably expect to use their payment method in any transaction. At this point they will stop paying more for additional units of popularity. For this reason, we model the relationship of willingness to pay a certain price and popularity as a logistic function.

2. Price is a function of:
   a. The fixed price of having an account, or the annual fee
   b. The variable price of having an account, or the transaction fee
   c. The price sensitivity of the consumer or merchant

3. Access is a dummy variable indicating whether or not the consumer has the ability to open an account based on requirements such as a minimum account balance. Whether or not they can meet the minimum account balance is determined by their income. We assume that all merchants have access to all platforms for the United States, though in a different country the model may have to be adjusted.

4. Risk is defined as the probability of a fraudulent or failed transaction, and is a characteristic of the platform. It has a negative relationship with utility.

4b. Platforms

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Description</th>
<th>Distribution and Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Account Balance</td>
<td>Required balance in order to be serviced by the platform</td>
<td>~Uniform(min=0, max=$10,000)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>U.S. average = ~$1,500</td>
</tr>
<tr>
<td>Account Fee</td>
<td>Monthly account servicing fee</td>
<td>~Uniform(min=0, max=$100)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>U.S. average = ~$25</td>
</tr>
<tr>
<td>Transaction Fee for merchants (fixed)</td>
<td>Amount charged to merchants per transaction</td>
<td>~Uniform(min=-$10, max=+$10)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>U.S. average = depends on platform...often=$0</td>
</tr>
<tr>
<td>Transaction Fee for consumers (fixed)</td>
<td>Amount charged to consumers per transaction</td>
<td>~Uniform(min=-$10, max=+$10)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>U.S. average = depends on platform...often=$0</td>
</tr>
</tbody>
</table>
Evolutionary Process

Unlike merchants and consumers, platforms do not make choices or take actions on an individual level. Instead, they evolve.

The reasoning behind why we chose to deal with platforms in this manner makes sense when presented in contrast to the other possible method; setting up a decision algorithm for platforms to change their profit structures in response to observed changes their own profits and the actions of consumers and merchants in the environment. However, this would be relatively unrealistic for two reasons. First, in reality platforms have limited flexibility in changing their profit structures. While they can change the balance of prices they offer consumers vs. merchants, and how much they spend on things like security, they cannot fundamentally change their product. Secondly, decisions about those prices and expenditures are extremely dynamic and difficult to predict, often being highly platform specific. An algorithm would not likely produce realistic platform responses.

Instead, platforms are randomly generated at setup, with new ones continuously being “born” into the system. Many of these will have extremely unrealistic and unprofitable profit structures, but as soon as any platform goes bankrupt it “dies”, and is removed from the system. In this way stronger platforms emerge as more time periods pass. As hundreds of platforms are born and die within the system, the more successful ones survive, and we see a natural evolution of platforms. Eventually only the strongest and fully viable platforms
are left. We repeated this process hundreds of times to select several hundred different types of viable platforms, and only considered those in our final analysis.
5. MODEL CALIBRATION: SIMULATED METHOD OF MOMENTS

Mean squared error is found by comparing the difference between a predicted result and the empirically observed result, that is, by finding:

\[ MSE(\theta) = E[(\hat{\theta} - \theta)^2] \]

However, in some instances, \( \hat{\theta} \) is extraordinarily difficult to compute analytically but relatively easy to simulate. A Simulated Method of Moments (SMM) allows one to minimize mean-squared error by using simulated results.

In writing the model described in the previous section, we created a way to easily simulate distributions of viable payment platforms in the United States \( (\hat{\theta}) \). To prove that these simulated results were realistic, we needed to estimate the parameters for price, risk, and popularity sensitivity among merchants and consumers \( (\text{previous notation: } B1, B2, B3, B4, B5, \text{ and } B6) \) that would minimize the mean squared error of the simulated distributions with an empirical distribution of payment platforms in the United States \( (\theta) \). In this way, we calibrate our simulated results to empirical data. This empirical distribution was presented in Figure 3. The results of our calibration to match this distribution (as closely as possible) are shown in Figure 4.

**Figure 4**

|----------------|-------------------------------|------------------------------|-----------------------------|-----------------------------|----------------------------------------|
| Credit Cards   | 27.06%                        | 80.54%                       | 5.21%                       | 13.86%                      | • Risk of default borne by platform.  
|                |                               |                              | 13.86%                      |                             | • More expensive for merchants, so more revenue for platforms  
|                |                               |                              |                             |                             | • Platforms make these more attractive to consumers |
| Debit Cards    | 62.35%                        | 8.21%                        | 4.50%                       | 4.90%                       | • Risk of default borne by platform.  
|                |                               |                              |                             |                             | • Cheaper for merchants, so less revenue for platforms  
|                |                               |                              |                             |                             | • Platforms make these less attractive to consumers |
| Checks         | 10.59%                        | 11.25%                       | 90.28%                      | 81.23%                      | • Cheap processing, but risk of default borne by merchant |
6. FINDINGS

6a. Simulation #1: Platform Specialization

The first step in analyzing our simulated data was to determine whether or not the simulated platforms had differentiable patterns of specialization. Specifically, we were curious if our simulated data matched the patterns of specialization already found within the United States in terms of the ACTA model: a group of platforms with low average costs leveraging the adjacencies a long-term banking relationship allows and a group of platforms with no barriers to entry but higher average costs. To this end, we wanted to discover whether or not different income classes were paying substantially different average prices to use a payment platform.

Conducting this analysis involved calculating two values. First, we found the average income of a consumer using any given payment platform. Second, we calculated the average cost this consumer was spending to use his chosen platform. This involved computing the average value and transaction frequency of a platform’s customers, multiplying these values together to find the average amount of money flowing through a platform each month, multiplying this value by a platform’s variable fees, and adding the fixed cost of using a platform. This gives us an average monthly cost of using a platform.

Comparing the average consumer income with the average cost of using the platform, we found that higher income individuals were indeed paying statistically significant lower average costs than lower-income individuals. Conducting a simple linear regression on the two values, we find that a $1,000 increase in average consumer income correlates with a $166 decrease in average monthly cost, with a p-value less than .001.
6b. Simulation #2: Risk-Affordability Tradeoff

Having discovered our simulated data matched the platform specialization found empirically within the United States, we wanted to learn whether we could encourage lower average costs by modifying the regulatory environment in the form of increasing a payment platform’s tolerable level of risk. The logic of this policy solution stems from the "risk-affordability" tradeoff described above: that in allowing a payment platform to tolerate more risk, they will spend less on security costs and pass on these savings to consumers in the form of lower average costs. We altered the tolerable level of risk from a 0.01% chance of breach to a 0.1% chance of breach, and at each point measured a platform's average cost using the same methodology in simulation one.

When conducting this analysis on consumer prices, we found that increasing the level of tolerable risk was correlated with statistically significant lower average consumer costs. This result supports the theory of a "risk-affordability" tradeoff for consumer prices, and suggests that a policy maker might decrease the cost of using a payment platform by loosening the regulatory environment. The plot "Consumer Cost vs. Risk" demonstrates this result.

However, as payment platforms represent two-sided markets, platforms charge one price to consumers and a separate price to merchants. When looking specifically at merchant prices, we found that increasing levels of tolerable risk was correlated with statistically significant higher merchant costs, as shown in the plot "Merchant Cost vs. Risk." This represents the complete opposite result one would expect if there were indeed a "risk-affordability" tradeoff. Investigating these simulated results,
we found the reason for this inconsistency: in lowering consumer prices, certain platforms became incredibly popular amongst the consumer population, and for these platforms merchants would calculate a high utility regardless of the price they paid to use the system. In other words, the platform became so popular amongst the consumer population that merchants couldn't afford not to use the system.

7. CONCLUSION

7a. Potential Extensions

One benefit to building our computational model is the vast array of scenarios that can now be simulated and investigated. There are two broad scenario categories that can easily be incorporated into the model.

i. **Environmental Changes:** Our model was calibrated to the U.S.’s consumers, merchants, and payment systems. But the same calibration could just as easily been done for another country. We built the model to take easily-defined inputs such as “Average Consumer Income” and “Income Gini Coefficient” to allow for just this type of flexibility. Importantly, such a re-calibration would allow a comparative statics analysis across countries, to investigate how payment platforms might evolve differently in different settings.

ii. **Regulatory / Policy Changes:** Certain policies can also be introduced into the model. For example, specifying a minimum investment in security, or capping a maximum level of transaction fees could both be introduced and simulated.

7b. Value of Computational Modeling in Policy Settings

As part of our project, we hoped to serve as a sort of testing ground for this type of analysis within a policy setting. Below, we highlight four potential benefits to computational, agent-based modeling as part of policy construction.

i. The discipline of constructing a model guides the investigation. Very often thinking “in distributions” is a useful way to understanding the stochastic nature of a problem, as well as the full gamut of reactions and interactions in the system.

ii. The model provides a tool for policy discovery and investigation. It is only the beginning point for full policy development, but offers a controllable and repeatable environment for analysis.

iii. The model allows nearly unlimited scenario analyses, as mentioned in 7a.

iv. Generalized tradeoffs and relationships (“there is a risk-affordability tradeoff”) can be parameterized and quantified. We now have a better sense, quantitatively, of this relationship.
REFERENCES


APPENDIX: MODEL CODE

Composed on Rstudio Version 0.97.551

Part 1: Functions

# Function: popularity.fn
# Input: table.cons, table.merch
# Output: two popularity vectors (length = num.platforms), one among consumers and one among merchants

# Function: proxcost.fn
# Input: table.plat, table.cons, i
# Output: proxcost = cost to consumer for CICO fee or travel time to nearest branch

# Function: access.fn
# Input: table.plat, table.cons, i
# Output: access = binary indicator of whether consumer meets minimum account balance

# Function: util.calc.cons
# Input: table.plat, table.cons, i
# Output: utilities, adjusted utilities, maximum utility, and selected platforms by preference order

# Function: util.calc.merch
# Input: table.plat, table.merch, i
# Output: utilities, adjusted utilities, maximum utility, and selected platforms by preference order

# Function: calculate.use.cons
# Input: table.plat, table.cons
# Output: my.platform.use.cons = list of transaction attribution for each consumer across each platform

# Function: calculate.use.merch
# Input: table.plat, table.merch
# Output: my.platform.use.merch = list of transaction attribution for each merchant across each platform

# Function: update.balance
# Input: table.plat, table.cons, table.merch
# Output: balance, consumer revenue, and merchant revenue for each platform
# 0. Model Inputs

```r
options(scipen=999)
set.seed(1)
library(VGAM)
```

# 1 CONSUMER CHARACTERISTICS

```r
create.cons = function(num.consumers,gini.cons,avginc){
  # (a) Generate Consumers from pareto distribution
  shape = 1/(2*gini.cons)+0.5 # based on gini coef, calculate shape parameter for pareto distribution
  scale = (avginc*(shape-1)) / shape # based on shape and desired expected value of income, calculate scale parameter of pareto distribution
  income.cons = rpareto(num.consumers,shape=shape,location=scale) / 12 # generate specified pareto distribution of consumers' MONTHLY income

  # (b) Generate other consumer characteristics
  budget = runif(num.consumers,min=0,max=1) # % of income that is "expenditures"
  savings = income.cons-(income.cons*budget) # % of income that is saved
  t.freq.cons = rgamma(num.consumers,shape=2,scale=40) # Gamma distribution matches (roughly) empirical distribution (source?). Average is 77/month and range is 0 to 400 [roughly]
  t.value.cons = (budget*income.cons)/t.freq.cons #(deterministic) "average" transaction value
  c.loc.x = runif(num.consumers,min=0,max=100) # random location on an x-y plane
  c.loc.y = runif(num.consumers,min=0,max=100) # random location on an x-y plane
  access  = rep(0,num.consumers) # initialize. indicator: does consumer have enough savings to meet minimum balance requirements?
  proxcost  = rep(0,num.consumers) # initialize. proximity cost of going to CiCo (e.g. ATM) location
  util.cons  = rep(0,num.consumers) # initialize. utility of from platform
  my.platform.cons  = rep(0,num.consumers) # initialize. Vector of consumer platform choice (multi-homing is possible)

  # (c) Aggregate consumer characteristics into a single table
  table.cons = data.frame(income.cons,budget,savings,t.freq.cons,t.value.cons,
                           c.loc.x,c.loc.y,
                           risk.sen.cons,price.sen.cons,pop.sen.cons,
                           access,proxcost,util.cons,my.platform.cons)

  return(table.cons) # function output
}
```
# 2. MERCHANT CHARACTERISTICS

create.merch = function(num.merch, gini.merch, table.cons) {

val = table.cons$t.value.cons
freq = table.cons$t.freq.cons

## (a) Set up merchant characteristics

shape = 1/(2^{gini.merch}+0.5) # based on gini coef, calculate shape parameter for pareto distribution
expval = sum(val*freq)/num.merch # Merchant income = consumer expenditure = trans value * trans freq.
scale = (expval*(shape-1))/shape # based on shape and desired expected value of income, calculate scale parameter of pareto distribution
income.merch = rpareto(num.merch, shape=shape, location=scale) # generate specified pareto distribution of consumers
t.value.merch = runif(num.merch, min=1, max=income.merch) # transaction value randomly between $1 and income
t.freq.merch = income.merch / t.value.merch # (deterministic) transaction frequency
util.merch = rep(0, num.merch) # initialize utility for each platform
my.platform.merch = rep(0, num.merch) # initialize. Vector of merchant platform choice (multi-homing is possible)

## (b) Aggregate merchant characteristics into a single table

table.merch = data.frame(income.merch, t.value.merch, t.freq.merch, risk.sen.merch, price.sen.merch, pop.sen.merch, util.merch, my.platform.merch)

return(table.merch) # function output
}

# 3. PLATFORM CHARACTERISTICS

create.plat = function(num.platforms) {

## (a) Set up platform characteristics

min.acct = runif(num.platforms, min=0, max=10000) # minimum account balance for consumers [$0 to $10000]
fee.acct = runif(num.platforms, min=0, max=100) # monthly account fee for consumers
fee.inter.fixed.merch = runif(num.platforms, min=-10, max=10) # fixed interchange fee, merchants [-$10 to $10]
fee.inter.perc.merch = runif(num.platforms, min=-0.1, max=0.1) # variable interchange fee, merchants [-10% to 10%]
fee.inter.fixed.cons = runif(num.platforms, min=-10, max=10) # fixed interchange fee, consumers [-$10 to $10]
fee.inter.perc.cons = runif(num.platforms, min=-0.1, max=0.1) # variable interchange fee, consumers [-10% to 10%]
}
# Function Definitions

```r
# (a) Platform Popularity

popularity.fn = function(table.cons, table.merch, table.plat) {
  d1 = table.cons # rename consumer input as new variable
  d2 = table.merch # rename merchant input as new variable
  popcons.nonzero = table(unlist(d$d$my.platform.cons)) # count total number of consumers on each platform
  return(table.plat) # function output
}
```

## 4. FUNCTIONS

### (a) Platform Popularity

```r
# Function Definitions

popularity.fn = function(table.cons, table.merch, table.plat) {
  d1 = table.cons # rename consumer input as new variable
  d2 = table.merch # rename merchant input as new variable
  popcons.nonzero = table(unlist(d$d$my.platform.cons)) # count total number of consumers on each platform
  return(table.plat) # function output
}
```

```r
# (b) Aggregate platform characteristics into a single table

table.plat = data.frame(min.acct, fee.acct, fee.cico, adopt.cost, invest.val, risk.breach, p.loc.x, p.loc.y, fee.inter.fixed.merch, fee.inter.perc.merch, fee.inter.fixed.cons, fee.inter.perc.cons, risk.c, popularity.cons, popularity.merch, adj.uncon, adj.con, balance, age)

table.plat # function output
```

```r
# (b) Aggregate platform characteristics into a single table

table.plat = data.frame(min.acct, fee.acct, fee.cico, adopt.cost, invest.val, risk.breach, p.loc.x, p.loc.y, fee.inter.fixed.merch, fee.inter.perc.merch, fee.inter.fixed.cons, fee.inter.perc.cons, risk.c, popularity.cons, popularity.merch, adj.uncon, adj.con, balance, age)

table.plat # function output
```
popmerch.nonzero = table(unlist(d2$my.platform.merch)) # count total number of merchants on each platform
popcons.all = rep(0,nrow(table.plat)) # initialize. all popularities = 0
popmerch.all = rep(0,nrow(table.plat)) # initialize. all popularities = 0
popcons.all[as.numeric(names(popcons.nonzero))] = popcons.nonzero / nrow(d1) # save nonzero consumer popularities
popmerch.all[as.numeric(names(popmerch.nonzero))] = popmerch.nonzero / nrow(d2) # save nonzero consumer popularities
ls = list(popcon=popcons.all,popmerch=popmerch.all) # put results in a list
return(ls) # function output

## (b) Merchant Utility
util.calc.merch = function(table.plat,table.merch,i){
d1 = table.plat # save platform input as new variable
d2 = table.merch[i,] # save merchant input as new variable
value = d2$t.freq.merch*d2$t.value.merch # total value of merchant transactions
price.var = d2$t.freq.merch*d1$fee.inter.fixed.merch + d2$t.freq.merch*d2$t.value.merch*d1$fee.inter.perc.merch # variable interchange fee for merchants (% + fixed, per transaction)
price.fixed = d1$fee.acct # fixed monthly price for having an account
price.all = d2$price.sen.merch*(-price.var-price.fixed)
risk = d2$risk.sen.merch*d1$risk.breach*d2$t.value.merch*(1-d1$risk.c) # risk for merchants, only if they bear the risk
pop = d2$pop.sen.merch*d1$popularity.cons # popularity multiplier times the percent popularity of platforms
util = price.all - risk + pop # merchant utility calculation
util[which (pop == 0)] = NA
rndm=rep(0,nrow(table.plat)) # initialize. random addition/subtraction on each utility (introduces randomness)
for(z in 1:nrow(table.plat)){rndm[z]=rndm[z]+rnorm(1,0,util.rndm)} # draw a new random utility disturbance for each platform
evaluated
util.adj = util + rndm # adjusted utility, based on random error
umax = max(util.adj,na.rm=TRUE)+0.0000000001 # maximum utility, plus a very small amount to prevent dividing by zero (in the case that max util is zero)
util.diff = abs((util.adj-max(util.adj,na.rm=TRUE))*umax) # absolute value of percent difference from the maximum utility
select.unordered = match(util.diff[which((util.diff<util.range))],util.diff) # select all platforms that provide utility within range of "util.range"
select = select.unordered[order(util.diff[select.unordered])] # re-order platform selections from highest to lowest utility
ls = list(util=util,util.adj = util.adj,umax=umax,select=select) # list of results
return(ls) # function output
}

## (c) (non)Proximity Cost -- cost from remove cash-in-cash-out fee, or opportunity cost from traveling to branch location
proxcost.fn = function(table.plat,table.cons,i){
d1 = table.plat # save platform input as new variable
d2 = table.cons[i,] # save consumer input as new variable
wage = d2$income.cons/250/8 # calculate hourly wage based on annual salary, 250 working days per year, and 8 hours per day
distance = sqrt((d1$p.loc.x-d2$c.loc.x)^2 + (d1$p.loc.y-d2$c.loc.y)^2) #calculate euclidian distance between platform and consumer
OC = wage*distance #opportunity cost = wage * distance. SO NO TIME FACTOR??
index1 = match(OC[which(OC<d1$fee.cico)],OC) #return element numbers of opportunity cost vector that are less than the cash-in, cash-out fee
index2 = match(OC[which(OC>d1$fee.cico)],OC) #return element numbers of opportunity cost vector that are greater than the cash-in, cash-out fee
proxcost = rep(NA,nrow(table.plat)) #initialize. vector of proximity costs for platforms
proxcost[index1] = OC[index1] #for platforms where it is cheaper to travel than pay CICO fee (indexed by "index1"), cost = opp cost.
proxcost[index2] = d1$fee.cico[index2] #for platforms where it is cheaper to pay CICO fee than travel (indexed by "index2"), cost = cico fee.
return(proxcost) #function output

## (d) Access indicator -- are consumer’s savings sufficient for minimum account balance?
access.fn = function(table.plat,table.cons,i){
  d1 = table.plat #save platform input as new variable
  d2 = table.cons[i,] #save consumer input as new variable
  access = rep(NA,nrow(table.plat)) #initialize. vector of access indicators for all platforms
  for (j in 1:nrow(table.plat)){ #for each platform
    if(d2$savings*12 < d1$min.acct[j]){ #if savings are less than minimum account requirements
      access[j]=NA #return NA
    } else{
      access[j]=1 #return "1"
    }
  }
  return(access) #function output
}

## (e) Consumer Utility
util.calc.cons = function(table.plat,table.cons,i){
  d1 = table.plat #save platform input as new variable
  d2 = table.cons[i,] #save consumer input as new variable
  proxcost = proxcost.fn(table.plat,table.cons,i) #calculate proximity cost
  access = access.fn(table.plat,table.cons,i) #determine access (binary indicator)
  price.var = d2$t.freq.cons*d1$fee.inter.fixed.cons + d2$t.freq.cons*d2$t.value.cons*d1$fee.inter.perc.cons #variable interchange fee
  for consumers (% + fixed, per transaction)
  price.fixed = d1$fee.acct #fixed cost of platform
  price.all = d2$price.sen.cons*(-price.var-price.fixed) #for consumers
  risk = d2$risk.sen.cons*d1$risk.breach*d2$savings*2*d1$risk.c #risk to consumer, only if risk.c = 1(i.e. indicator of consumer bearing risk)
  pop = d2$pop.sen.cons*(d1$popularity.merch)
util = access*(price.all - risk + pop) #consumer utility calculation
util[which ((pop == 0)) = NA
rndm=rep(0,nrow(table.plat)) #initialize. random addition/ subtraction on each utility (introduces randomnes)
for(z in 1:nrow(table.plat)) {rndm[z]=rndm[z]+rnorm(10,util.rndm)} #draw a new random utility disturbance for each platform evaluated
util.adj = util + rndm #adjusted utility, based on random disturbance
if (all(is.na(util.adj))){ # if consumers have selected zero platforms
  select = NA
  ls = list(util=NA,util.adj = NA,umax=NA,select=NA)
} else { #if consumers have selected at least one platform
  umax = max(util.adj,na.rm=TRUE)+0.0000000001#maximum utility, plus a very small amount to prevent dividing by zero (in the case that max util is zero)
  util.diff = abs((util.adj-max(util.adj,na.rm=TRUE))/umax) #absolute value of percent difference from the maximum utility
  select.unordered = match(util.diff[which(util.diff<util.range)],util.diff) #select all platforms that provide utility within range of "util.range"
  select = select.unordered[order(util.diff[select.unordered])] #re-order platform selections from highest to lowest utility
  ls = list(util=util,util.adj = util.adj,umax=umax,select=select) #list of results
}
return(ls)

## (f) Allocate transaction usage for consumers
## Calculate use of each platform - Consumers want to use their cards in order of preference, but are limited by the popularity of that system AS USED BY MERCHANTS

calculate.use.cons = function(table.plat,table.cons){
  #Initialize
  my.platform.use.cons = list(NULL)
  my.platform.use.cons[1:nrow(table.cons)] = list(0)
  pop.merch = table.plat$popularity.merch

  for (i in 1:nrow(table.cons)){
    #if this consumer does not have a platform, platform.use.cons = NA
    if (all(is.na(table.cons$my.platform.cons[[i]]))){
      my.platform.use.cons[[i]] = NA
      next
    }
    for (j in 1:length(table.cons$my.platform.cons[[i]])){
      my.platform.use.cons[[i]][j] = pop.merch[1:nrow(table.cons$my.platform.cons[[i]])[j]
      if (sum(my.platform.use.cons[[i]]) == 1){
        break
      }
      }
  }
if (sum(my.platform.use.cons[[i]]) > 1) {
    remainder = sum(my.platform.use.cons[[i]]) - 1
    my.platform.use.cons[[i]][j] = my.platform.use.cons[[i]][j] - remainder
    break
}
}
return(my.platform.use.cons)
}

## (g) Allocate transaction usage for merchants

calculate.use.merch = function(table.plat, table.merch) {

    my.platform.use.merch = list(NULL)
    my.platform.use.merch[1:nrow(table.merch)] = list(0)
    pop.cons = table.plat$popularity.cons

    for (i in 1:nrow(table.merch)) {  # number of merchants
        if (all(is.na(my.platform.merch[[i]]))) {
            my.platform.use.merch[[i]] = NA
        } else {
            for (j in 1:length(table.merch$my.platform.merch[[i]]))) {  # number of choosen platforms
                my.platform.use.merch[[i]][j] = pop.cons[table.merch$my.platform.merch[[i]]][j]
            }
            if (sum(my.platform.use.merch[[i]]) == 1) {
                break
            } else if (sum(my.platform.use.merch[[i]]) > 1) {
                remainder = sum(my.platform.use.merch[[i]]) - 1
                my.platform.use.merch[[i]][j] = my.platform.use.merch[[i]][j] - remainder
                break
            }
        }
    }
    return(my.platform.use.merch)
}

## (h) Update balances of platforms

update.balance = function(table.plat, table.cons, table.merch, risktable) {


consumer.revenue = rep(0, nrow(table.plat)) #initialize revenue from consumers
merchant.revenue = rep(0, nrow(table.plat)) #initialize revenue from merchants
consumer.val = rep(0, nrow(table.plat)) #initialize value from consumers
consumer.vol = rep(0, nrow(table.plat)) #initialize volume from consumers
merchant.val = rep(0, nrow(table.plat)) #initialize value from merchants
merchant.vol = rep(0, nrow(table.plat)) #initialize volume from merchants
adj.con.rev = rep(0, nrow(table.plat)) #initialize revenue from conventional adjacencies
adj.uncon.rev = rep(0, nrow(table.plat)) #initialize revenue from unconventional adjacencies
totalrev = rep(0, nrow(table.plat)) #initialize all revenue
ratio = rep(0, nrow(table.plat)) #initialize ratio of investment:revenue for the risk level
invest.val = rep(0, nrow(table.plat)) #initialize investment necessary to maintain security
costper = rep(0, nrow(table.plat))
balance = rep(0, nrow(table.plat)) #initialize profit / loss for platforms

for (i in 1:nrow(table.plat)){ #for each platform

d1=table.plat[i,] #rename variable for ith row of platform

d2=table.cons #rename consumer input

d3=table.merch #rename merchant input

    ## 1. Revenue - fees

    for (j in 1:num.consumers){ #for each consumer
        if (all(is.na(d2[j,]$my.platform.cons[[1]]))){ #check if this consumer has chosen exactly zero platforms
            consumer.revenue[i] = consumer.revenue[i] + 0 #if yes, then add zero to consumer revenue
            consumer.val[i] = consumer.val[i]+0
            consumer.vol[i] = consumer.vol[i]+0
        } else{ #if not, then:
            for (k in 1:length(d2[j,]$my.platform.use.cons[[1]])){ #for each platform chosen by the jth consumer
                if (d2[j,]$my.platform.cons[[1]][k] == i){ #if the jth consumer's kth choice is the same as the ith platform, then:
                    consumer.revenue[i] = consumer.revenue[i] + #add to consumer revenue:
                    (d2$t.freq.cons[j] * d$fee.inter.fixed.cons + #revenue from fixed transaction fee
                     d2$t.freq.cons[j] * d2$t.value.cons[j] * d$fee.inter.perc.cons) * #revenue from variable transaction fee
                    d2[j,]$my.platform.use.cons[[1]][k] #multiply this revenue by the fraction of transactions that the jth consumer sends to the kth platform
                    consumer.val[i] = consumer.val[i] + d2$t.value.cons[j] * d2[j,]$my.platform.use.cons[[1]][k]
                    consumer.vol[i] = consumer.vol[i] + d2$t.freq.cons[j] * d2[j,]$my.platform.use.cons[[1]][k]
                }
            }
        }
    }
for (j in 1:num.merch) { # for each merchant
  if (all(is.na(d3[j,]$my.platform.merch[[1]]))) { # if the jth merchant has chosen exactly zero platforms
    merchant.revenue[i] = merchant.revenue[i] + 0 # if yes, then add zero to merchant revenue.
    merchant.val[i] = merchant.val[i]+0
    merchant.vol[i] = merchant.vol[i]+0
  } else { # if not, then:
    for (k in 1:length(d3[j,]$my.platform.use.merch[[1]])) { # for each platform chosen by the jth merchant
      if (d3[j,]$my.platform.merch[[1]][k] == i) { # if the jth consumer's kth choice is the same as the ith platform, then:
        merchant.revenue[i] = merchant.revenue[i] + # add to merchant revenue:
        (d3$t.freq.merch[j] * d1$fee.inter.fixed.merch + d3$t.freq.merch[j] * d3$t.value.merch[j] * d1$fee.inter.perc.merch) * # revenue from variable transaction fee
        d3[j,]$my.platform.use.merch[[1]][k] # multiply this revenue by the fraction of transactions that the jth merchant sends to the kth platform
        merchant.val[i] = merchant.val[i] + d3$t.value.merch[j] * d3[j,]$my.platform.use.merch[[1]][k]
        merchant.vol[i] = merchant.vol[i] + d3$t.freq.merch[j] * d3[j,]$my.platform.use.merch[[1]][k]
      }
    }
  }
}

## 2. Revenue - conventional adjacencies
if(table.plat$adj.con[i]==1){
  index.choice = sapply(table.cons$my.platform.cons,function(x) match(i,x)) # if consumer x has selected platform i, return the preference number; else, return NA
  index.choice[is.na(index.choice)] = 0 # set index to 0 if consumer x did not choose platform i
  rownums = cbind(seq(1:num.consumers),index.choice)# return row numbers
  index.choice[which(index.choice>0)] = rownums[which(rownums[,2]>0)] # set index to row number in consumer table if consumer x DID choose platform i
  avail.loans = sum(table.cons[index.choice,"savings"])*ROR # sum the available loan balance
  adj.con.rev[i] = avail.loans*ROR # calculate loan return based on market Rate of Return
}

## 3. Revenue - unconventional adjacencies
if(table.plat$adj.uncon[i]==1){
  avail.cons = table.plat$popularity.cons[i]*nrow(table.cons) + table.plat$popularity.merch[i]*nrow(table.merch) # count all consumers and merchants on this platform
  adj.uncon.rev[i] = avail.cons*adrev # calculate advertising or other data revenue
}

## 4. Total Revenue
totalrev[i] = consumer.revenue[i] + merchant.revenue[i] + adj.con.rev[i] + adj.uncon.rev[i]
## 5. Total Costs - Update investment level and cost per user

\[
\text{ratio}[i] = \text{risktable}[\text{match(table.plat$\text{risk.breach}[i],\text{risktable}),2}]
\]
\[
\text{invest.val}[i] = \text{ratio}[i]\times\text{totalrev}[i]
\]
\[
\text{costper}[i] = (\text{table.plat$\text{popularity.cons}[i] + \text{table.plat$\text{popularity.merch}[i]}\times(\text{cost.per.user/2})
\]

## 6. Total Profit

\[
\text{balance}[i] = \text{totalrev}[i] - \text{invest.val}[i] - \text{costper}[i] \# \text{update balance for ith platform, which is total revenue less investment costs}
\]

\}

ls = list (balance=balance, consumer.revenue=consumer.revenue, merchant.revenue=merchant.revenue,
            consumer.val=consumer.val,consumer.vol=consumer.vol,merchant.val=merchant.val,merchant.vol=merchant.vol,
            adj.con.rev=adj.con.rev,adj.uncon.rev=adj.uncon.rev,
            ratio=ratio,invest.val=invest.val,costper=costper) \# aggregate results

return(ls) \# function output

\}
Part 2: Simulated Method of Moments

# 1 MODEL INPUTS

```r
set.seed(1)

### Basic setups

# Consumer, platform, and merchant characteristics
num.consumers = 1000
num.merch = num.consumers / 10
gini.merch = 0.927
gini.cons = 0.477
avginc = 40000

# Randomness characteristics
util.rndm = 0

# Set up risk / investment table
risks = c(3.6/10000, 0.45/10000)
ratios = c(0.1, 0.1)
risktable = cbind(risks, ratios)

# Multiplier coefficients START
risk.sen.cons = 1
price.sen.cons = 1
pop.sen.cons = 1
risk.sen.merch = 1
price.sen.merch = 1
pop.sen.merch = 1

# Multiplier coefficients GRID
#utilrange = 1
#riskcons = c(0.1, 0.2)
#pricecons = c(0.05, 0.1, 0.15, 0.2, 0.25)
#popcons = c(1, 10, 100)
#riskmerch = c(5, 10, 15, 20)
#pricemerch = c(0.05, 0.1, 0.2)
#popmerch = c(400, 450, 500, 550)
```
utilrange = 1
riskcons = c(0.1)
pricecons = c(0.25)
popcons = c(100)
riskmerch = c(5)
pricemerch = c(0.05)
popmerch = c(500)

### Platforms: Credit Cards, Debit Cards, and Checks

# Credit Cards
min.acct = 1500
fee.acct = 10
fee.inter.fixed.merch = 0
fee.inter.perc.merch = 0.0225
fee.inter.fixed.cons = 0
fee.inter.perc.cons = -0.0138
fee.cico = 0
risk.breach = 3.6/10000
risk.c = 1
p.loc.x = 0
p.loc.y = 0
popularity.cons = 0
popularity.merch = 0
credit = data.frame(min.acct,fee.acct,fee.cico,risk.breach,p.loc.x,p.loc.y,
fee.inter.fixed.merch,fee.inter.perc.merch,fee.inter.fixed.cons,fee.inter.perc.cons,
risk.c,popularity.cons,popularity.merch)

# Debit Cards
min.acct = 1500
fee.acct = 10
fee.inter.fixed.merch = 0
fee.inter.perc.merch = 0.0125
fee.inter.fixed.cons = 0
fee.inter.perc.cons = -0.0065
fee.cico = 2.16
risk.breach = 3.6/10000
risk.c = 1
p.loc.x = 0
p.loc.y = 0
popularity.cons = 0
popularity.merch = 0
debit = data.frame(min.acct, fee.acct, fee.cico, risk.breach, p.loc.x, p.loc.y,
                   fee.inter.fixed.merch, fee.inter.perc.merch, fee.inter.fixed.cons, fee.inter.perc.cons,
                   risk.c, popularity.cons, popularity.merch)

# Checks
min.acct = 1500
fee.acct = 10
fee.inter.fixed.merch = 0.25
fee.inter.perc.merch = 0
fee.inter.fixed.cons = 0
fee.inter.perc.cons = 0
fee.cico = 2.16
risk.breach = 0.45/10000
risk.c = 0
p.loc.x = 0
p.loc.y = 0
popularity.cons = 0
popularity.merch = 0
checks = data.frame(min.acct, fee.acct, fee.cico, risk.breach, p.loc.x, p.loc.y,
                     fee.inter.fixed.merch, fee.inter.perc.merch, fee.inter.fixed.cons, fee.inter.perc.cons,
                     risk.c, popularity.cons, popularity.merch)

table.plat = rbind(credit, debit, checks)

# Placeholders for other characteristics that are irrelevant to the SMM
invest.val = rep(0.3) #initialize.
adj.uncon = rep(0.3) #indicator: can merchants capitalize on unconventional adjacencies?
adj.con = rep(0.3) #indicator: can merchants capitalize on conventional adjacencies?
balance = rep(0.3) #initialize. capital reserves from merchants profit
age = rep(0.3) #initialize. number of periods that platform has existed
ROR = 0.05/12 #monthly rate of return on account balances
adrev = 10 #monthly revenue per customer attributed to "value of data" (e.g. advertising)
min.invest = 0 #minimum security investment
cost.per.user = 0 #cost of maintaining each user (consumers and merchants)

table.plat = data.frame(table.plat, invest.val, adj.uncon, adj.con, balance, age)

############################################################
# 2. PARAMETER ESTIMATION

results = NULL
# 1 draw consumer / merchant characteristics

table.cons = create.cons(num.consumers,gini.cons,avginc)
table.merch = create.merch(num.merch,gini.merch,table.cons)

iteration=1

for (a in utilrange){
for (b in riskcons){
for (c in pricecons){
for (d in popcons){
for (e in riskmerch){
for (f in pricemerch){
for (g in popmerch){

    # 2. randomly assign
    my.platform.cons = list(NULL) #initialize list of platforms chosen by each consumer
    my.platform.merch = list(NULL) #initialize list of platforms chosen by each merchant
    for (i in 1:num.consumers){my.platform.cons[[i]] = sample(1:nrow(table.plat),1)} #randomly assign each consumer to a single platform
    for (i in 1:num.merch){my.platform.merch[[i]] = sample(1:nrow(table.plat),1)} #randomly assign each merchant to a single platform
    table.cons$my.platform.cons = my.platform.cons #append list of choice vectors to consumer table
    table.merch$my.platform.merch = my.platform.merch #append list of choice vectors to merchant table

    pop = popularity.fn(table.cons,table.merch,table.plat) #calculate popularity
    table.plat$popularity.cons = pop$popcon #extract consumer popularity by platform, and save to platform table
    table.plat$popularity.merch = pop$popmerch #extract merchant popularity by platform, and save to platform table

    # 3. select 1 combination of coefficients
    util.range = a
    table.cons$risk.sen.cons = b
transratio.cons = table.cons$t.value.cons / table.cons$t.freq.cons
    table.cons$price.sen.cons = c
    table.cons$pop.sen.cons = d
    table.merch$risk.sen.merch = e
    #transratio.merch = table.merch$t.value.merch / table.merch$t.freq.merch
    #table.merch[which(transratio.merch>2000),]$price.sen.merch = 10
    #table.merch[which(transratio.merch<=1500),]$price.sen.merch = 0.1
    table.merch$price.sen.merch = f
    table.merch$pop.sen.merch = g

    # 4. allow consumers / merchants to make a choice
    for (t in 12){

for(i in 1:num.consumers){
  choice = util.calc.cons(table.plat,table.cons,i) # calculate utility
  table.cons$util.cons[i] = choice$umax # extract maximum utility
  table.cons$my.platform.cons[[i]] = choice$select # extract selected platforms
}

pop = popularity.fn(table.cons,table.merch,table.plat) # calculate popularity

for(i in 1:num.merch){
  choice = util.calc.merch(table.plat,table.merch,i) # calculate utility
  table.merch$util.merch[i] = choice$umax # extract maximum utility
  table.merch$my.platform.merch[[i]] = choice$select # extract selected platforms
}

pop = popularity.fn(table.cons,table.merch,table.plat) # calculate popularity

# 6. calculate distribution of value / volume, and save

for(i in 1:num.consumers){
    table.cons$my.platform.use.cons = calculate.use.cons(table.plat,table.cons) # get transactions attribution and add to consumer table
    table.merch$my.platform.use.merch = calculate.use.merch(table.plat,table.merch) # get transactions attribution and add to merchant table
}

revenues = update.balance(table.plat,table.cons,table.merch,risktable)
result = c(iteration,a,b,c,d,e,"cond.",g,
           (revenues$consumer.vol + revenues$merchant.vol)/ sum(revenues$merchant.vol+revenues$consumer.vol),
           (revenues$consumer.val + revenues$merchant.val)/ sum(revenues$merchant.val+revenues$consumer.val))
results = rbind(results,result)

iteration=iteration+1
print(iteration)
}
}

results = data.frame(results,row.names=NULL)
names=c("iter","utilrange","riskcons","pricecons","popcons","riskmerch","pricemerch","popmerch",
...)
### (a) Comparison vs. True distribution

cc.vol = 0.2706 #true volume of credit card transactions
dc.vol = 0.6235 #true volume of debit card transactions
chk.vol = 0.1059 #true volume of check transactions
cc.val = 0.0521 #true value of credit card transactions
dc.val = 0.0450 #true value of debit card transactions
chk.val = 0.9028 #true value of check transactions
true = c(cc.vol,dc.vol,chk.vol,cc.val,dc.val,chk.val) #combine into a vector

vol.diff.fn = function(x){ #comparison vs. consumer value distribution across the 3 "platforms"
calendar.diff = as.numeric(((x[9]-true[1])/true[1])^2)
    #calendar.diff = as.numeric(abs((x[9]-true[1])/true[1]))
dc.diff = as.numeric(((x[10]-true[2])/true[2])^2)
    #dc.diff = as.numeric(abs((x[10]-true[2])/true[2]))
chk.diff = as.numeric(((x[11]-true[3])/true[3])^2)
    #chk.diff = as.numeric(abs((x[11]-true[3])/true[3]))
total.diff = mean(c(calendar.diff,dc.diff,chk.diff))
return(total.diff)
}

val.diff.fn = function(x){ #comparison vs. consumer volume distribution across the 3 "platforms"
calendar.diff = as.numeric(((x[12]-true[4])/true[4])^2)
    #calendar.diff = as.numeric(abs((x[12]-true[4])/true[4]))
dc.diff = as.numeric(((x[13]-true[5])/true[5])^2)
    #dc.diff = as.numeric(abs((x[13]-true[5])/true[5]))
chk.diff = as.numeric(((x[14]-true[6])/true[6])^2)
    #chk.diff = as.numeric(abs((x[14]-true[6])/true[6]))
total.diff = mean(c(calendar.diff,dc.diff,chk.diff))
return(total.diff)
}

# calculate differences for each row
val.diff = apply(results,1,function(x) val.diff.fn(x))
vol.diff = apply(results,1,function(x) vol.diff.fn(x))
total.diff = (vol.diff + val.diff) / 2
Part 3: Evolutionary Algorithm

\[
\text{start = Sys.time()} \\
\text{set.seed(10)} \\
\]

# Consumer, platform, and merchant characteristics
num.consumers = 1000 \\
num.merch = num.consumers / 10 \\
num.platforms = 10 \\
gini.cons = 0.477 \\
gini.merch = 0.927 \\
avginc = 40000 \\

# Randomness characteristics
util.rndm = 0 \\
util.range = 1 \\

# Multiplier coefficients
risk.sen.cons = 0.1 \\
price.sen.cons = 0.25 \\
pop.sen.cons = 100 \\
risk.sen.merch = 5 \\
price.sen.merch = 0.05 \\
pop.sen.merch = 500 \\

# "Macro" characteristics
ROR = 0.05/12 #monthly rate of return on account balances \\
adrev = 1 #monthly revenue per customer attributed to "value of data" (e.g. advertising) \\
min.invest = 1000 #minimum security investment \\
cost.per.user = 100 / 12 #monthly cost per user to platforms \\

inputs = data.frame(num.consumers,num.merch,num.platforms, \\
gini.cons,gini.merch,avginc, \\
util.rndm,util.range, \\
risk.sen.cons,price.sen.cons,pop.sen.cons, \\
risk.sen.merch,price.sen.merch,pop.sen.merch, \\
ROR,adrev,min.invest) \\

# Set up risk / investment table
#minrisk = 0.0001*c \\
#maxrisk = 0.01*c \\
#risks = seq(minrisk,maxrisk,by=0.0001*c)
#minratio = (1/1000)*c
#maxratio = (1/10)*c
#ratios = seq(maxratio,minratio,by=-(1/1000)*c)
#risktable = cbind(risks,ratios)

#LVL = c(1,2,3,4,5,6,7,8,9,10)
for(lvl in LVL){
  minrisk = 0.0001
  maxrisk = 0.1
  risks = seq(minrisk,maxrisk,by=0.0001)
  minratio = (1/1000)
  maxratio = (1/2)
  ratios = seq(maxratio,minratio,by=(maxratio-minratio)/1000)[-1]
  full.risktable = cbind(risks,ratios)
  risktable = full.risktable[1:(lvl+99),]

  #attach(inputs)
  # 1. Draw consumers, merchants, and platforms
  table.cons = create.cons(num.consumers,gini.cons,avginc)
  table.merch = create.merch(num.merch,gini.merch,table.cons)

  #detach(inputs)
  table.cons$run = rep(0,num.consumers)
  table.merch$run = rep(0,num.merch)

  #1a initial vectors of 'winning' platforms
  table.winning.plat = NULL
  table.ultra.winning.plat = NULL
  cons.archive = NULL
  merch.archive = NULL
  plat.tradeoffs = NULL

  t=25  #set how many iterations you want
  for (k in 1:t){  #Number of matches
    table.cons$run = k
    table.merch$run = k
  }
table.winning.plat = NULL #initialize for new rounds
for (f in 1:10){  #f rounds of choosing the best of 10 platforms
  table.plat = create.plat(num.platforms) #create new group of ten platforms

  # 2a. random assignment to platforms
  my.platform.cons = list(NULL) #initialize, list of platforms chosen by each consumer
  my.platform.merch = list(NULL) #initialize, list of platforms chosen by each merchant
  for (i in 1:num.consumers){my.platform.cons[[i]] = sample(1:nrow(table.plat),1)} #randomly assign each consumer to a single platform
  for (i in 1:num.merch){my.platform.merch[[i]] = sample(1:nrow(table.plat),1)} #randomly assign each merchant to a single platform

  # 2b. update popularity in platform table
  table.cons$my.platform.cons = my.platform.cons #append list of choice vectors to consumer table
  table.merch$my.platform.merch = my.platform.merch #append list of choice vectors to merchant table
  pop = popularity.fn(table.cons,table.merch,table.plat) #calculate popularity
  table.plat$popularity.cons = pop$popcon #extract consumer popularity by platform, and save to platform table
  table.plat$popularity.merch = pop$popmerch #extract merchant popularity by platform, and save to platform table

  for (z in 1:2){ #repeat choice dynamics twice
    # 4a. consumer choice
    for (i in 1:num.consumers){
      choice = util.calc.cons(table.plat,table.cons,i) #calculate utility
      table.cons$util.cons[[i]] = choice$umax #extract maximum utility
      table.cons$my.platform.cons[[i]] = choice$select #extract selected platforms
    }

    # 4b. update consumer popularity in platform table, and calculate consumer transaction attribution
    pop = popularity.fn(table.cons,table.merch,table.plat) #calculate popularity
    table.plat$popularity.cons = pop$popcon #extract consumer popularity by platform, and save to platform table
    table.cons$my.platform.use.cons = calculate.use.cons(table.plat,table.cons) #get transactions attribution and add to consumer table

    # 5a. merchant choice
    for (i in 1:num.merch){
      choice = util.calc.merch(table.plat,table.merch,i) #calculate utility
      table.merch$util.merch[[i]] = choice$umax #extract maximum utility
      table.merch$my.platform.merch[[i]] = choice$select #extract selected platforms
    }

    # 5b. update merchant popularity in platform table, and calculate merchant transaction attribution
pop = popularity.fn(table.cons,table.merch,table.plat)  #calculate popularity
    table.plat$popularity.merch = pop$popmerch  #extract merchant popularity by platform, and save to platform table
    table.merch$my.platform.use.merch = calculate.use.merch(table.plat,table.merch)  #get transactions attribution and add to merchant table

# 6. Platform revenue calculation
revenues = update.balance(table, plat, table.cons, table.merch, risktable)
    table.plat$balance = revenues$balance
    table.plat$invest.val = revenues$invest.val

# 7. Calculate the 'winning' platform (the platform with the highest balance)
    table.winning.plat = rbind(table.winning.plat, table.plat[match(max(table.plat$balance),table.plat$balance),])
print(f)
}

# 8. Now, final round between the f 'winning' platforms

    table.plat = table.winning.plat

# 8.1 Random assignment to platforms
    my.platform.cons = list(NULL)  #initialize. list of platforms chosen by each consumer
    my.platform.merch = list(NULL)  #initialize. list of platforms chosen by each merchant
    for (i in 1:num.consumers){my.platform.cons[[i]] = sample(1:nrow(table.plat),1)}  #randomly assign each consumer to a single platform
    for (i in 1:num.merch){my.platform.merch[[i]] = sample(1:nrow(table.plat),1)}  #randomly assign each merchant to a single platform

# 8.2 Update popularity in platform table
    table.cons$my.platform.cons = my.platform.cons  #append list of choice vectors to consumer table
    table.merch$my.platform.merch = my.platform.merch  #append list of choice vectors to merchant table
    pop = popularity.fn(table.cons,table.merch,table.plat)  #calculate popularity
    table.plat$popularity.cons = pop$popcons  #extract consumer popularity by platform, and save to platform table
    table.plat$popularity.merch = pop$popmerch  #extract merchant popularity by platform, and save to platform table

for (z in 1:2){ #repeat choice dynamics twice
    # 8.3. a. Consumer choice
    for(i in 1:num.consumers){
        choice = util.calc.cons(table.plat,table.cons,i)  #calculate utility
            table.cons$util.cons[i] = choice$umax  #extract maximum utility
            table.cons$my.platform.cons[[i]] = choice$select  #extract selected platforms
        }
# 8.3.b. update consumer popularity in platform table, and calculate consumer transaction attribution
pop = popularity.fn(table.cons, table.merch, table.plat)  # calculate popularity

  table.plat$popularity.cons = pop$popcon  # extract consumer popularity by platform, and save to platform table
  table.cons$my.platform.use.cons = calculate.use.cons(table.plat, table.cons)  # get transactions attribution and add to consumer table

# 8.4.a. merchant choice
for (i in 1:num.merch) {
  choice = util.calc.merch(table.plat, table.merch, i)  # calculate utility
  table.merch$util.merch[i] = choice$umax  # extract maximum utility
  table.merch$my.platform.merch[[i]] = choice$select  # extract selected platforms
}

# 8.5.b. update merchant popularity in platform table, and calculate merchant transaction attribution
pop = popularity.fn(table.cons, table.merch, table.plat)  # calculate popularity

  table.plat$popularity.merch = pop$popmerch  # extract merchant popularity by platform, and save to platform table
  table.merch$my.platform.use.merch = calculate.use.merch(table.plat, table.merch)  # get transactions attribution and add to merchant table
}

# 8.6. platform revenue calculation
revenues = update.balance(table.plat, table.cons, table.merch, risktable)

  table.plat$balance = revenues$balance
  table.plat$invest.val = revenues$invest.val

# 9. Pull out relevant figures for the ultimate winning platform (highest balance)
winning.platform.num = match(max(table.plat$balance), table.plat$balance)

# 9.a Index of consumers using winning platform

  index.cons = rep(0, num.consumers)

  for (i in 1:num.consumers) {
    if (winning.platform.num %in% table.cons$my.platform.cons[[i]]) {
      index.cons[i] = 1
    }
  }

  cons.archive = rbind(cons.archive, table.cons[(index.cons == 1),])

# 9.b Index of merchants using winning platform

  index.merch = rep(0, num.merch)

  for (i in 1:num.merch) {
if ( winning.platform.num %in% table.merch$my.platform.merch[[i]]){
    index.merch[i] = 1
}
merch.archive = rbind(merch.archive,table.merch[(index.merch==1),])

# 9.c  Number of people with no access to winning platform
access=NULL
for (i in 1:num.consumers){
  access = rbind(access, access.fn(table.plat[winning.platform.num,,], table.cons, i))
}
no.access = length(access[which(is.na(access)==TRUE)])/ num.consumers
table.plat$no.access = no.access

# Step 10. Put final winning platform in table
  table.ultra.winning.plat = rbind(table.ultra.winning.plat, table.plat [match(max(table.plat$balance),table.plat$balance),])

# Step 11. Print status
print(k)
}

### Save data
  cons.archive = cons.archive[-c(14,16)]
  merch.archive = merch.archive[-c(8,10)]
  write.table(cons.archive,file=paste("~/Documents/consarchive",lvl,".txt",sep=""))
  write.table(merch.archive,file=paste("~/Documents/mercharchive",lvl,".txt",sep=""))
  write.table(table.ultra.winning.plat,file=paste("~/Documents/ultrawin",lvl,".txt",sep=""))

}