PRICING AND PROMOTION STRATEGIES IN A COMPETITIVE MARKET: AN AGENT-BASED APPROACH

SEYHAN DENİZ NİŞANCI

BOĞAZİÇİ UNIVERSITY

2025

PRICING AND PROMOTION STRATEGIES IN A COMPETITIVE MARKET: AN AGENT-BASED APPROACH

Thesis submitted to the

Institute for Graduate Studies in Social Sciences

in partial fulfillment of the requirements for the degree of

Master of Arts

in

Management Information Systems

by

Seyhan Deniz Nişancı

Boğaziçi University

2025

Pricing and Promotion Strategies in a Competitive Market:

An Agent-Based Approach

The thesis of Seyhan Deniz Nişancı

has been approved by:

Prof. Bertan Yılmaz Badur
(Thesis Advisor) _____

Dr. Meltem Emine Mutlutürk
(Thesis Co-Advisor) _____

Prof. Sona Mardikyan _____

Assoc. Prof. Nazım Ziya Perdahçı _____

Assist. Prof. Tuğrul Cabir Hakyemez
(External Member) _____

September 2025

DECLARATION OF ORIGINALITY

I, Seyhan Deniz Nişancı, certify that

- I am the sole author of this thesis and that I have fully acknowledged and documented in my thesis all sources of ideas and words, including digital resources, which have been produced or published by another person or institution;
- this thesis contains no material that has been submitted or accepted for a degree or diploma in any other educational institution;
- this is a true copy of the thesis approved by my advisor and thesis committee at Boğaziçi University, including final revisions required by them.

Signature……...
Date…….....

ABSTRACT

Pricing and Promotion Strategies in a Competitive Market:

An Agent-Based Approach

The fast-moving consumer goods (FMCG) market is highly competitive, marked by frequent purchases and price sensitivity, where pricing, promotion, and quality strategies critically shape consumer behavior and firm performance. For this purpose, in this study an agent-based simulation model is designed to analyze customer purchasing patterns in competitive market. The model includes two types of agents: customers and sellers. Customers make probabilistic purchase decisions using a logistic utility function, incorporating price, promotion, quality and personal characteristics like age and income. Based on historical performance data, sellers use regression-based learning techniques to update their price and promotion strategies dynamically. The research questions are addressed through three scenarios: S0, S1, S2, referring respectively to price-only competition, price and promotion competition, and quality-differentiated competition. The results show that in S0, competition lowers prices and profits but increases sales. In S1, promotion expands demand yet often reduces profits due to higher costs. In S2, quality differentiation raises prices and can yield profit gains when costs are manageable and customers value quality, while further lowering no-purchase outcomes. This study shows how agent-based modeling can extend traditional logit-oligopoly analysis by including dynamic seller learning and product differentiation. It contributes to marketing research methods and improves our understanding of customer behavior in competitive markets.

ÖZET

Rekabetçi Piyasada Fiyatlandırma ve Promosyon Stratejileri: Ajan Tabanlı Yaklaşım

Hızlı tüketim malları pazarı, sık satın alımlar ve yüksek fiyat duyarlılığıyla son derece rekabetçidir. Fiyat, promosyon ve kalite stratejileri tüketici davranışlarını ve firma performansını belirleyici rol oynar. Bu amaçla, rekabetçi piyasada müşteri satın alma kalıplarını analiz etmek için bir ajan-tabanlı simülasyon modeli tasarlanmıştır. Modelde iki tür ajan bulunmaktadır: müşteriler ve satıcılar. Müşteriler, fiyat, promosyon, kalite ve yaş ile gelir gibi kişisel özellikleri içeren lojistik fayda fonksiyonu üzerinden olasılıksal satın alma kararları vermektedir. Satıcılar ise geçmiş performans verilerine dayanarak fiyat ve promosyon stratejilerini dinamik biçimde güncellemek için regresyon tabanlı öğrenme teknikleri kullanmaktadır. Araştırma soruları üç senaryo üzerinden ele alınmıştır: S0 (yalnızca fiyat rekabeti), S1 (fiyat ve promosyon rekabeti) ve S2 (kalite farklılaştırmalı rekabet). Sonuçlar, S0 senaryosunda rekabetin fiyatları ve karları düşürdüğünü, ancak satışları artırdığını göstermektedir. S1 senaryosunda promosyon talebi genişletmekte, fakat daha yüksek maliyetler nedeniyle karlar sıklıkla azaltmaktadır. S2 senaryosunda ise kalite farklılaştırması fiyatları yükseltmekte ve maliyetler yönetilebilir olduğunda ve müşteriler kaliteye değer verdiğinde kar artışı sağlayabilmekte, ayrıca satın almama oranlarını daha da düşürmektedir. Bu çalışma, dinamik satıcı öğrenmesini ve ürün farklılaştırmasını dahil ederek ajan-tabanlı modellemenin geleneksel logit-oligopol analizini nasıl genişletebileceğini göstermektedir. Böylece pazarlama araştırma yöntemlerine katkıda bulunmakta ve rekabetçi piyasalarda tüketici davranışlarını daha iyi anlamamızı sağlamaktadır.

ACKNOWLEDGMENTS

When I began this thesis journey, an agent-based simulation model was designed to analyze customer purchasing patterns in a competitive market. Today, I am grateful for the knowledge and experience I have gained throughout the process.

A debt of gratitude is owed to Prof. Bertan Yılmaz Badur, my supportive thesis advisor, for his guidance and encouragement throughout this study.

I would also like to express my sincere gratitude to Dr. Meltem Emine Mutlutürk, my co-advisor, for her valuable insights, constructive feedback, and motivation during the preparation of this thesis.

I am also thankful to the members of my thesis jury, Prof. Sona Mardikyan, Assoc. Prof. Nazım Ziya Perdahçı, and Assist. Prof. Tuğrul Cabir Hakyemez, for their valuable comments and suggestions, which greatly contributed to the improvement of this study.

I would also like to thank my colleagues at work for their kind support and understanding during this period.

Last but not least, I would like to thank my family and friends, who provided me with their unconditional and endless support.

I hope that this study serves as a useful reference for future research and applications in this field.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ABM: Agent-Based Modeling

ANOVA: Analysis of Variance

ARUM: Additive Random Utility Model

DCM: Discrete Choice Models

FMCG: Fast-Moving Consumer Goods

IIA: Irrelevance of Irrelevant Alternatives

MNL: Multinomial Logit

ODD: Overview, Design concepts, Details

RUT: Random Utility Theory

CHAPTER 1

INTRODUCTION

In today's competitive markets, a business's ability to create dynamic, data-driven marketing strategies is just as important to its success as the value of its products. Pricing and promotion strategies have a direct impact on a company's profitability, particularly in industries like fast-moving consumer goods (FMCG), where demand is high, competition is intense and customer preferences shift quickly. At the same time, product quality significantly shapes customers purchase decisions, often determining whether a product is chosen over competing alternatives.

A key focus of economics and marketing science studies has been understanding how customers make decisions. Traditional methods have created models based on deterministic utility maximization, which makes the assumption that people would always select the option that has the greatest benefit. However, countless empirical research in behavioral economics and psychology demonstrates that people make random, inconsistent, and context-sensitive decisions. The trend toward Random Utility Theory-based models in decision-making processes has been driven by these findings. Given that individual preferences are probabilistic, discrete choice models developed in this context provide a more flexible and realistic framework.

Agent-Based Modeling (ABM) is a method used to study systems by modeling the behavior of interacting heterogeneous individual agents. The ABM approach has gained increasing attention in recent years. By providing an artificial environment in which individual-level decision makers have heterogeneous characteristics, can interact with each other and update their behavior over time,

agent-based models allow researchers to explore complex social dynamics and emergent phenomena. ABM allows us to analyze how decision rules defined at the micro level translate into outputs at the macro level. In this context, ABM, in conjunction with models of discrete choice, becomes a powerful tool for analyzing how probabilistic choice behavior of customers evolves in a dynamic FMCG market.

The FMCG market differs significantly from other markets in terms of purchase frequency, price level, and competitive intensity. In this sector, products are typically purchased on a daily or weekly basis, resulting in very high purchase frequency. Unit prices are low, and consumers tend to be highly price-sensitive, which fosters intense competition among brands.

In contrast, markets such as durable goods involve less frequent purchases, higher unit prices, and competition that is shaped by more distinct product differentiation among fewer players. These differences highlight the dynamic, high-volume, and fast-paced nature of the FMCG market. Customer behavior in these markets typically varies over time in response to changing business tactics. In order to analyze real-world decision-making in FMCG environments, it is especially appropriate to construct a simulation model that can capture both the probabilistic nature of customer decisions and the adaptive behavior of sellers. In this study, an agent-based simulation model is developed to analyze customer reactions to variables such as price, promotion and quality in a competitive market, and these needs are directly addressed by the model. An oligopolistic market structure is examined in this study where sellers dynamically update their price and promotion strategies according to their previous sales performance. Real-world dynamics in marketplaces like FMCG, where customers have a variety of options and businesses compete constantly, are reflected in such a structure. A more accurate depiction of

strategic interactions and market-level consequences is made possible by modeling this environment.

Customer preferences are modeled as a stochastic utility function with observable utility components as price,promotion and the quality of the product along with an unobservable random error term. This structure is in line with the Additive Random Utility Model (ARUM) framework widely used in the literature and the resulting choice probabilities are based on the closed form of the Multinomial Logit (MNL) model. The model is also extended to include the alternative of not choosing any seller as a no buy option.

The main purpose of the model developed in this thesis is to represent customer behavior that is highly sensitive to marketing variables such as price and promotion, and to simulate how probabilistic choice rules at the micro-level are reflected in macro-level market outcomes. The model focuses on how sellers update their marketing strategies over time in addition to modeling customer decision-making. In order to optimize future profits, each seller uses their historical sales performance and adjusts their price and promotion levels periodically. The simulation is able to capture the dynamic and adaptable character of competition, especially in fast-paced markets like FMCG, via its learning-based strategy modification mechanism.

In light of the model, this study aims to answer the following research questions:

- RQ1: How does the number of sellers affect equilibrium prices, profitability, sales volume, and overall market coverage in a monopolistic and an oligopolistic FMCG market under the price-only competition setting?

- RQ2: In a price and promotion competition setting, how do firms adjust their pricing and promotion strategies over time? How does rising competition affect the trade-off between promotion efforts and profitability, and can promotions help slow the decrease in sales and expand market coverage compared to a price-only setting?
- RQ3: In a quality-differentiated competition setting, how does a seller's product quality influence competition, pricing decisions, profitability, and sales volume?

The thesis is structured as follows: Chapter 2 provides a literature review, beginning with discrete choice models and their use in oligopolistic competition, then outlining the fundamentals of the ABM approach and its growing role in marketing research, and finally reviewing applied studies on pricing and promotion. Chapter 3 reviews ABM methodology that is used in this study using ODD (Overview, Design concepts, Details) protocol. Chapter 4 introduces three designed scenarios, including price-only, price and promotion, and quality-differentiated competition, and then reports the results. Chapter 5 discusses the findings of the study in detail, outlines its limitations, and proposes avenues for future research. Finally Chapter 6 concludes the study.

CHAPTER 2

LITERATURE REVIEW

This chapter provides a comprehensive overview of the theoretical foundations and related studies that inform the development of the simulation model presented in this thesis and is structured into five main parts. First, the fundamental concepts of Discrete Choice Models (DCM) are introduced, with an emphasis on the Random Utility Theory and the widely used Multinomial Logit (MNL) model. Second, the application of the logit model in oligopolistic markets is examined, focusing on pricing behavior, competition dynamics, and equilibrium analysis. Third, the basic principles and components of ABM are discussed as a simulation method to capture individual decision-making and emergent behavior. In the following section, the growing role of ABM in marketing and consumer behavior research is explored. Finally, applied studies that specifically integrate ABM with pricing and promotion strategies are reviewed to position this thesis within the existing body of work and identify the research gap it aims to address.

2.1 Discrete choice models

Discrete Choice Models (DCM), used to model customer behavior, provide a theoretical framework to explain the situation in that individuals make a choice among a limited number of non-exclusive alternatives. These models are based on the Random Utility Theory (RUT). According to RUT, the utility of each alternative for an individual consists of some observable components (such as price or promotion) and some unobservable components that are considered random (Anderson, de Palma & Thisse, 1992). This theoretical framework is based on the

formulation developed by McFadden (1974) and known as the Additive Random Utility Model (ARUM) (as cited in Anderson et al., 1992). In the ARUM model, total utility of an individual for an alternative is defined as follows:

$$U_i=V_i+\varepsilon_i$$

Here V_i represents observable utility and ε_i represents the unobservable random error term. The individual's decision rule is deterministic, in other words, the individual chooses the alternative with the highest utility. However, this decision process takes on a probabilistic nature due to the unobservable component for the modeler. If the error terms are assumed to be independently and double exponential distributed, then choice probabilities of an individual can be calculated as below (Anderson et al., 1992):

$$P(i) = \frac{e^{V_i/\mu}}{\sum_j e^{V_j/\mu}}$$

This structure is called as Multinomial Logit (MNL) model and widely preferred in economic modeling because of its computationally simple closed-form solution. However, Irrelevance of Irrelevant Alternatives assumption, a basic assumption of this model, can be limiting in some cases. According to Irrelevance of Irrelevant Alternatives, the relative probability of choosing two alternatives is not affected by the presence of a third alternative. Particularly when similar alternatives are introduced to decision sets, this could produce unrealistic results (Anderson et al., 1992).

Discrete choice models are widely used not only at the theoretical level but also in applied simulation. In particular, these models are frequently used to describe the decision-making processes of agents in ABM. In ABM environment, each agent calculates utility according to certain attributes, on the other hand, due to the agent heterogeneity and uncertainty, choices are defined probabilistically rather than

deterministically. In this respect, the MNL model becomes a natural choice for ABM to represent micro decision structures (Anderson et al., 1992). These models provide a powerful theoretical framework for analyzing and modeling preferences of individual statistically between various alternatives. The implementation of this framework integrated with ABM allows for understanding the transformation of micro level decision rules into system outcomes at the macro level.

2.2 Oligopoly and the logit model

Multinomial Logit (MNL) model is frequently used to analyze pricing and competitive strategies of rivals in oligopolistic markets with differentiated products. The model allows firms to forecast their market share and profitability by taking into account how customer preferences are formed under the logit structure.

Anderson, de Palma and Thisse (1992) argue that the logit model provides a robust theoretical framework, especially in markets with product diversity.Since the MNL model has a structure suitable for analyzing firms' price and profit decisions and the model can be expressed in closed form, the relationships between variables such as prices, demand levels and the number of firms can be easily analyzed. Anderson et al. (1992) analyze in detail the effects of fixed costs on equilibrium prices and the number of firms in the market.

MNL model also includes the possibility that customers do not choose any firm in the market. In this case, the customer first decides whether to buy a product. If she/he chooses to buy, she/he then chooses one of the firms. Anderson et al (1992) explain this situation with a two-stage choice process. MNL also helps to determine the equilibrium number of firms. As new firms enter the market, competition increases. When competition increases, prices fall. However, if the number of firms

is too large, firms struggle to cover their fixed costs. This implies a trade-off between competition and profitability. The model analyzes the relationship between the number of firms and customer welfare. It makes it possible to calculate the total utility (customer welfare) received by customers.

These features allow the model to not only examine customer but also firm behavior, market structure and economic equilibrium. Anderson et al. (1992) theoretically explain the influences of variables such as the number of firms, price, cost structure and exogenous alternatives on customer welfare. Therefore, the logit model provides a powerful analysis tool in competitive markets with differentiated products.

2.3 Fundamentals of the agent-based modeling (ABM) approach

Agent-Based Modeling (ABM) is a method used to study systems by modeling the behavior of interacting heterogeneous individual agents. As Bonabeau (2002) emphasizes, ABM provides a framework to capture emergent phenomena that traditional top-down modeling often fails to explain. In ABM, agents are defined as autonomous entities with several basic properties such as perception, performance (movement, communication, action), memory and decision rules (Gilbert, 2020). An agent can sense its environment, communicate with other agents, remember previous experiences and make decisions according to a set of rules.

Ahmed, Greensmith, and Aickelin (2010) compared ABM and System Dynamics models using the SIR epidemiological model and demonstrated that ABM can generate natural variance without requiring parameter changes. This finding highlights the superiority of ABM in modeling heterogeneous behaviors and micro-level uncertainties, as it enables more realistic system-level outcomes. Modelling

individual interactions at the micro level to obtain unintuitive system behaviors at the macro level is one of the main strengths of ABM. It gives ABM an important advantage in understanding emergent behaviors, which classical modeling approaches often fail to explain.

This structure provides a strong basis for analyzing the system-wide effects of individual-level factors. At this point, it is important to note that ABM can be developed either on theoretical micro-foundations or directly on empirical data. Comparative studies such as Janssen & Ostrom (2006) and Smajgl et al. (2011) show that theory-driven ABMs can produce reliable outcomes that are reasonably close to real-world observations without additional calibration, while empirical models provide higher level of detail but require costly data collection (Taghikhah, Filatova, & Voinov, 2021).

When modeling competitive markets, both customers and sellers can be modeled as agents and these decisions can be formed by social influences, environmental factors and personal experiences (Rand & Rust, 2011). These strengths make ABM particularly suitable for marketing research, as it enables the study of how individual-level behaviors aggregate into market dynamics.

2.4 The role of ABM in marketing research

The micro-level flexibility of ABM has led to its widespread use in marketing and customer behavior research, Onggo and Foramitti (2021) state that ABM is frequently used in business and management research across areas such as strategic management, marketing, operations and supply chain, financial management, and risk management to support decision making. In the same study, it is emphasized that by incorporating agents with different characteristics in the model, it is possible to

analyze how customers' individual-level responses transform into systemic outcomes over time. Similarly, Mishra et al. (2025) emphasize ABM's capability for modeling both individual choices and collective behavior dynamically, specifically in complex systems and high uncertainty areas, such as demand forecasting, new product adoption, and evaluation of promotional effectiveness.

ABM has also been increasingly applied in marketing research, particularly in studies modeling social interactions and dynamic behaviors. Nasir and Bozanta (2014) stated that ABM can be used in many areas of marketing, such as advertising, customer loyalty, pricing strategies and brand preferences, and offers a powerful method to simulate the outcomes of individual-level marketing strategies.

The growing importance of social simulation in marketing research is illustrated by Jager (2007) who proposed a framework for incorporating the four P's of marketing (product, price, place, and promotion) into market simulation. The study explains how product attributes can be connected to consumer, how price is linked to consumer budgets, how place shows accessibility, and how promotion can be represented through advertising and word-of-mouth. Instead of giving an empirical simulation, Jager suggests ways to formalize consumer decision-making in ABM models. Jager's study makes a conceptual contribution by showing that the four P's of marketing can be included in a simulation. Danaye, Kian, and Colmekcioglu (2023) are among the few who combine the Black-Box model of consumer behavior with ABM. They integrate the 4Ps of the marketing mix and social influence into their preference function and show how consumer behavior concepts can be represented in ABM.

According to Romero et al. (2023), the majority of ABM research in marketing has emphasized the diffusion of innovation and information sharing

through social networks. In their study, the ability of modeling customer differences, social interactions, and changes over time makes ABM more realistic than traditional top-down models. Sadeqi-Arani and Roozmand (2024) found that ABM is used widely in topics such as innovation diffusion, energy consumption, pricing, and social impact in their bibliometric review in which they analyzed the applications of ABM in marketing and customer behavior from 1995 to 2022.

2.5 Pricing, promotion and applied ABM studies

ABM has been increasingly applied to extend beyond the assumptions of traditional economic models, offering a way to study dynamics that cannot be captured within static equilibrium frameworks. Supantha and Sharma (2024) emphasize that the classical Arrow–Debreu general equilibrium theory is insufficient to explain the real economy, as it examines only equilibrium states under perfect competition, homogeneous products, and complete information conditions. On the other hand, disequilibrium processes are essential in real markets. To address this limitation, the authors developed an ABM that enables heterogeneous agents, monopolistic competition, perfect product differentiation, and trade in disequilibrium. Simulation results show that only a small share of the replications converged to equilibrium. In many cases, most firms exited the market, leaving only a single producer, and the initially equal distribution of wealth evolved into inequality over time. These findings highlight that ABMs are capable of capturing not only equilibria but also disequilibrium dynamics, tendencies toward monopolization, and the emergence of wealth inequality.

ABM can be utilized to analyze how customers react at the micro level and monitor how these reactions ultimately result in systemic results, in particular in the

areas of price and promotion strategies. Sanchez-Cartas's (2018) algorithm provides a new link between ABM and the industrial organization literature. Based on Game Theory, it ensures that the decisions of consumers and firms are optimal without using equilibrium equations or complicated functions. The model is divided into two components with one representing consumer behavior and the other representing pricing strategies of firms. These components integrate behavioral rules consistent with the predictions of Game Theory. In this way, the algorithm captures heterogeneous agents and imperfect information, while still ensuring rational and optimal outcomes without complex mathematical calculations. In a study involving versioning and second-order price discrimination, Chellappa and Mehra (2016) described how price differentiation can be applied to customer segments.

Based on Lee et al.'s (2014) ABM model, which tests various pricing and timing strategies for new product launches, price success is related to promotion timing and customer heterogeneity. Similarly, Delre et al (2007) showed using ABM approach that not only the presence but also the timing and targeting of promotions are playing a pivotal role in the success or failure of new product diffusion. Their results highlight that no promotion or wrongly timed promotions may lead to diffusion failure, while targeting small cohesive groups accelerates initial adoption. Zhang and Zheng (2019) examined the relationship between quality, price, and promotion and demonstrated that, when paired with moderate pricing, promotions boost purchase rates. Zhang, Levinson, and Zhu (2008) used an agent-based model to study a duopoly with public and private road providers. Pricing and capacity expansion are the two actors' variables for competition. In the model, different agent types were defined, route choices of travelers, destination, and trip frequency were simulated based on heterogeneous characteristics of agents. The results show that

customers with high income pay more for small time savings, low-income users gain little, and middle-income users lose the most since they either pay high tolls or suffer from long delays.

There are also applied ABM studies that involve other factors such as social networks, personal values and cultural structure. Danaye et al. (2023) modeled brand preferences by including social networks and marketing mix factors. Their preference function is a linear combination of four components: price, product features, advertising, and social influence and decision rule in their study is deterministic. The simulations show that product features and price are the most important factors, while promotion and social influence have a smaller role.

Most of the theoretical models tested in Sanchez- Cartas's study are based on two-seller markets. The author emphasizes the flexibility of the algorithm to include more than two firms, but does not demonstrate this empirically within the article (2018). In another study, Abbasi Siar, Keramati, and Motadel (2022) applied an agent-based approach to model impulse buying behavior in a retail setting, focusing particularly on the role of discounts and social swarm influences. However, the framework does not account for competitive interactions among multiple firms and instead assumes a single-store environment.The impact of non-price competitive factors on competitive dynamics is frequently emphasized in the literature. Katsamakas and Sanchez-Cartas (2024) analyze platform competition within a two-sided duopoly framework by incorporating users' Corporate Social Responsibility (CSR) preferences into the model. The simulation results demonstrate that when CSR preferences intensify, prices decrease, competition becomes fiercer, firm profitability declines, and, in contrast, consumer welfare increases. These findings

parallel our thesis, which shows how consumer heterogeneity changes the market outcomes of pricing and promotion strategies.

While previous studies have used ABM and logit models to examine pricing and promotion strategies, most of them focus limited competition. Many models include only price or only promotion, while factors such as quality differentiation are often missing. In addition, the dynamic learning process of firms in updating their strategies has not been fully explored.

This thesis addresses these gaps by developing an ABM of an oligopolistic FMCG market, where price, promotion, and quality are examined. The model also allows firms to adapt their strategies through learning based on past sales performance. By combining probabilistic customer behavior with adaptive firm strategies, the study provides a dynamic, multi-dimensional analysis of competition that has been largely absent in the literature.

CHAPTER 3

METHODOLOGY

In this study, agent-based model is structured in the framework of the ODD protocol (Overview, Design concepts, Details). ODD is a common standard that allows agent-based models to be defined in a systematic, explicit and reproducible way. This protocol describes the structure of the model, decision-making processes, interactions between agents, and sub-models step by step (Grimm et al., 2020). This chapter sets out the methodological design of the study: how the agent-based model is specified, implemented, and evaluated using the ODD protocol.

3.1 Overview

The Overview gives a high-level description of the model: its purpose and research questions, types of agents and their main state variables, the overall process flow and schedule of actions in each time step.

3.1.1 Purpose

The main purpose of the model is examining the market dynamics as a result of customer reactions to sellers' price and promotion strategies and seller differantiation via quality in a competitive FMCG market. In line with RQ1, the model compares monopoly and oligopoly settings under price-only competition to study how the number of sellers affects equilibrium outcomes. Addressing RQ2, it adds promotion and lets firms update price–promotion strategies over time via learning from past performance, evaluating the trade-off between promotional effort and profitability and its effect on market coverage. For RQ3, it introduces quality differentiation to

assess how product quality shifts competitive intensity and optimal pricing. The model incorporates customer heterogeneity (age, income and price, promotion, and quality preferences) and a probabilistic choice mechanism. The overarching aim is to provide a dynamic, micro-to-macro view of competition where adaptive sellers and stochastic customer decisions jointly generate market-level patterns.

3.1.2 Entities, state variables, scales

There are two types of agents in the model: customers and sellers. Customer agents represent individual customers in the market. Each customer has the following characteristics:

id: unique identifier,

age: assigned from a normal distribution,

income: assigned from a normal distribution,

decision: updated according to the utility value at each time step

Seller agents represent firms that offer similar products at different prices and promotion levels. Each seller has the following variables:

id: unique identifier

price: updatable value that changes over time

promotion level: one of three levels (none, low, high)

sales & revenue: updated on every time step based on customer choices

historical data: previous price, promotion and sales

quality: attribute assigned to each seller that differentiates products and affects customer utility

3.1.3 Process overview and scheduling

In each decision cycle of the simulation, sellers first evaluate their past performance using historical data on price, promotion, and sales outcomes. They employ a second-order regression model, where profit is predicted based on past price and promotion values. This regression model allows each seller to evaluate alternative pricing strategies-namely increasing, decreasing, or keeping the current price-combined with all feasible levels of promotion. For each potential combination, the expected profit is calculated. The strategy that yields the highest expected profit is then selected and implemented in the next time step.

Once sellers have updated their strategies, customers make their purchasing decisions. Each customer evaluates the available sellers by calculating the perceived utility of each option, which depends on price, promotion and quality of the product. Based on a probabilistic choice rule, each customer either selects one of the available products or chooses not to purchase at all. Through this sequential interaction-first firm-level learning and strategy update, then consumer decision-making-the model captures dynamic market behavior over time without relying on direct physical interaction among agents. This process is illustrated in Figure 1, which shows the process overview of the simulation.

3.2 Design concepts

Standard principles in the ODD protocol describe how agents behave, interact and adapt. In this study, the system-level patterns that emerge are emergence, heterogeneity, objectives, sensing, observation, adaptation, learning, prediction, and stochasticity.

In this model, the sales volumes, price and market share distributions are not directly modeled, but instead occur as a result of customer decisions and sellers updates over time. This is fundamental to the principle of emergence in the model. Individual level customers decisions emerge macro level collective patterns that are observable over time.

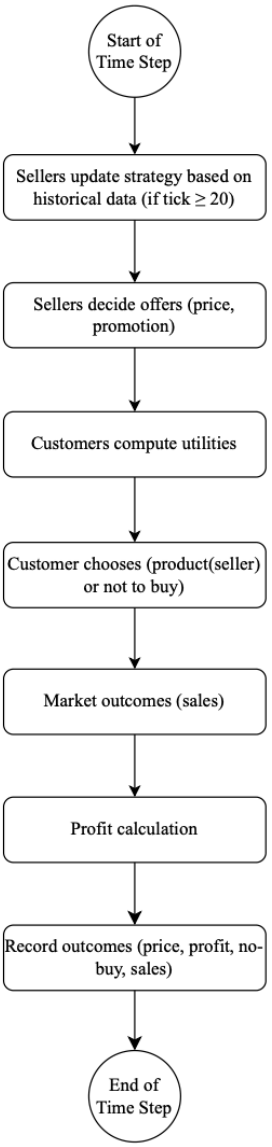

Figure 1. Flowchart of a Time Step

The main factors affecting how agents behave are heterogeneity and objectives. Customer responses to price and promotions vary depending on their age

and income. If none of the options are appealing, they will decide not to purchase anything at all.

Sensing work differently for customers and sellers. Customers can observe all prices and promotions in the market and use this information to make their decisions. Sellers, however, cannot see what others are doing or how customers behave individually. Rather, they depend on their own sales results.

The model allows to observe behaviors of agents at each time step systematically by recording key outputs. This includes prices, promotions, sales, and profits for sellers, as well as customer choices and their associated probabilities.

Other design principles are listed below:

Adaptation

Each seller builds a forecast model which uses price, promotion and sales data over the past time steps and updates its strategy based on this model. By testing various combinations of strategies-specifically different price and promotion levels-sellers apply the one that maximizes expected profits. This shows that sellers adapt their behavior according to their previous performance in the market.

Learning

The process uses quadratic regression so that the curvilinear relationship between price, promotion and profit can be captured. Sellers fit a second-order regression model to historical data and use it to estimate future profits. This demonstrates that sellers do not only adapt reactively but also learn functional relationships between input variables and profit to make forward-looking predictions.

Prediction

Sellers in the model perform predictive decision-making by using a regression-based learning mechanism. Each seller constructs a second-order regression model based on historical price, promotion, and profit data. This model is used to estimate the expected profit for a range of alternative strategies. Rather than reacting only to past performance, sellers use this model to predict future outcomes of their decisions. This predictive mechanism allows agents to choose strategies not solely based on previous results but based on forward-looking expectations about profit maximization.

Stochasticity

In the model, randomness is especially apparent in the customer choice process. Utility scores for each seller and for the "do not buy" option are calculated using customer characteristics (age, income) and the sellers' price and promotion levels. A random error term is added to represent decision uncertainty. These scores and quality are then transformed into probabilities using a logit function, and choices are made probabilistically. As a result, even under identical conditions, different outcomes can occur across simulation runs. Randomness is also present in the sellers' behavior during the initial exploration phase. Before the learning mechanism is activated, sellers randomly select price and promotion strategies within predefined bounds. This ensures variation in early strategies, promoting diversity in later learning phases. The combination of these stochastic elements ensures behavioral variability and rich system dynamics across replications.

3.3 Details

This section describes the technical specifications of the model, including the initialization of agents, the submodels for sellers and customers, and the mechanisms that guide their decisions.

3.3.1 Initialization

The simulation is started by creating a set number of customer and seller agents. Sellers have initially randomized price and promotion levels. The initial time steps of the model are defined as the exploration period. During this period, sellers randomly experiment with price and promotion strategies without learning or forecasting process. After the exploration period, sellers start making decisions using their historical data, from this point on, each agent's behavior evolves over time in line with the dynamic nature of the model. Thus, the system initially has the chance to observe the effects of various strategies.

3.3.2 Sub models

This section presents two submodels that represent the decision-making processes of sellers and customers in the simulation. The first submodel explains how each seller updates their price and promotion strategies by analyzing past performance using a quadratic regression approach. The second submodel describes how each customer evaluates the available sellers and decides whether to make a purchase, based on a utility function that incorporates both observable attributes and a random component. These two submodels work together to simulate the dynamic interaction between seller strategy updates and customer choice behavior.

3.3.2.1 Strategy selection

Profit of each seller is calculated based on the product's selling price, the unit cost of the product, promotional cost which differentiates according to the level of promotion applied and sellers' cost of quality. The profit function is defined as follows:

$$Profit_j = (Price_j - cost) * Sales_j - Promocost_j * Sales_j - costq_j * quality * Sales_j$$

Explanation of the variables are listed below:

-price$_j$: the price set by seller j at a given time step

-cost: the fixed unit cost of the product

-sales$_j$: the number of units sold by seller j at a given time step

-$promocost_j$: the promotional cost of the level of promotion selected

In the model, there are three levels of promotion. These levels aims to represent the increasing resource requirements and marketing intensity of different levels of promotional practices. Level 0 is a non-promotional situation that the seller offers the product without any discounts, coupons or visibility enhancements and therefore no promotional costs are charged. Level 1 represents a standard level of promotion which includes a small discount or digital coupon. It has low promotional cost per sale. Level 2 represents a high level of promotional effort. At this level, the seller may apply multiple tactics together, such as an aggressive discount, feature, advertisement or physical in-store display. Therefore, a high promotional cost per sale is charged. With this level structure, the model realistically represents the potential of promotional practices to increase sales volume and the impact of this effort on profitability. While sellers can attract more customers with higher levels of promotion, they also risk a reduction in net profit due to higher unit costs.

In the initial exploration phase of the model, sellers randomize their strategies and this part of the model is defined as exploration period as mentioned before. Starting from time step T+1, each seller builds a quadratic regression model to predict profit using the data from the previous T time steps. The regression model is shown as:

$$Profit = \gamma_0 + \gamma_1 * price + \gamma_2 * lowpromo + \gamma_3 * highpromo + \gamma_4 * price^2 + \gamma_5 * price * lowpromo + \gamma_6 * price * highpromo$$

This model assumes that profit can only be in a curvilinear relationship with price and promotion, not a linear one. In the real world profits may increase as price increases, but after a certain point profits decline due to diminishing returns. This quadratic regression assumption is also supported by the analysis using historical data generated by the model. Each seller performs a multivariate quadratic regression using the last T time steps of observed price, promotion, and corresponding profit data. Using the estimated model, the seller evaluates pricing scenarios: increasing the current price, decreasing it, or keeping it constant. For each of these price scenarios, all feasible promotion levels are evaluated to calculate the expected profit. By comparing the predicted outcomes, the seller selects the price-promotion combination that yields the highest predicted profit. Figure 2 shows the relationship between price and profit using both linear and quadratic regression and the results are compared.

The quadratic regression in Figure 2 reveals profits decline after reaching a maximum at a given price level. However, the linear model fails to capture this curvature and assumes a constant slope. This justifies the preference for the quadratic model not only theoretically but also empirically.

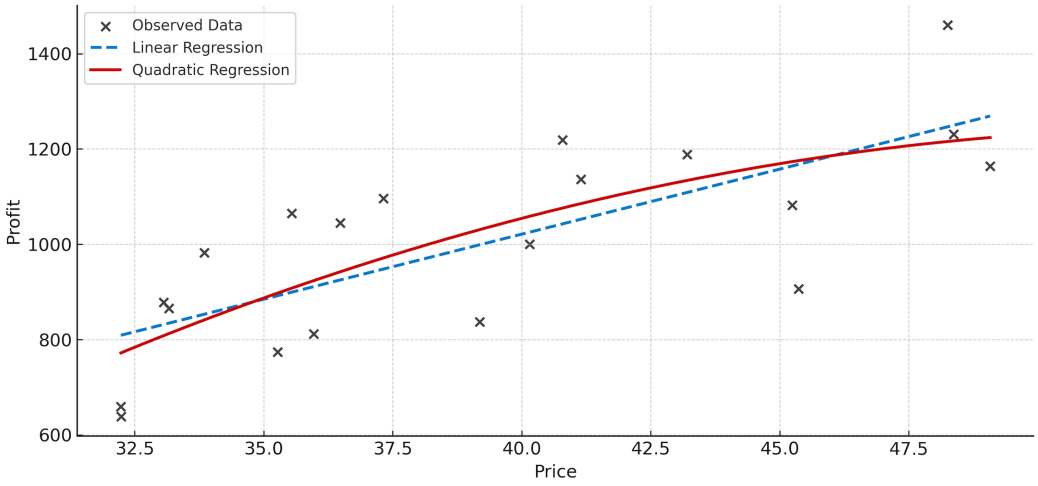

Figure 2. Linear and quadratic regression of the relationship between price and profit

The use of quadratic regression in the learning process of the model is an appropriate choice because it is in line with the economic nature of the profit function and because it yields more successful results in the tests performed on the model data.

3.3.2.2 Decision of customer

For each customer agent a utility score for the choice of sellers is computed at each time step. Perceived utility of customer i for seller j is defined as follows:

$$U_{ij} = \beta_0 + \beta_1 * age_i + \beta_2 * income_i + \beta_3 * price_j + \beta_4 * lowpromo_j + \beta_5 * highpromo_j + \varepsilon_{ij} + \beta_6 * quality_j$$

The variables in the utility function are defined as:

- age_i and $income_i$: individual characteristics of the customer

- $price_j$: the price of the product offered by the seller

- $lowpromo_j$ and $highpromo_j$: dummy variables defined by the level of promotion

- ε_{ij}: a random term representing unobserved effects at the moment of decision and modeled as N(0, 0.1)

In theory, the multiple logit model assumes that the distribution of the ε_{ij} term is Gumbel distribution. However, a normal distribution with zero mean and low variance is used to model this term in this study for ease of use and to enhance numerical stability. This method gives the model enough adaptability to add diversity and randomness to the selection process.

This structure is consistent with the multinomial logit structure commonly used in the discrete choice modeling literature and incorporates the impact of observable and unobservable effects on customer choice.

In this study, the coefficients of the utility function were assigned heuristically to to represent common customer behavior tendencies. Age was given a negative coefficient to reflect its decreasing effect on responsiveness to new offers as customers get older.Income was given a positive coefficient to capture the idea that higher income generally increases the likelihood of purchase. The coefficient of price was set negative to reproduce the expected negative effect of higher prices on purchase probability. Promotion effects were assigned positive values to capture the intuitive notion that promotions increase purchase likelihood. Finally, the coefficient of quality was given a positive value to reflect that higher product quality enhances the likelihood of purchase. These values were not meant to represent exact empirical magnitudes but to provide a consistent behavioral structure for the simulation.
After calculating the utility scores of all alternatives for each customer, the choice probabilities are calculated. In order to do this calculation a multiple logit model is used. The probability of customer i choosing alternative j is calculated as follows:

$$\mathrm{P}_{ij} = \frac{\exp\left(\mathrm{U}_{ij}\right)}{\sum_{k=1}^{J} \exp(U_{ik}) + \exp\left(\mathrm{U}_{i0}\right)}$$

This transformation has mathematically the same structure as the multinomial logit function. Multinomial logit normalizes the relative utility scores of alternatives

into a probability distribution. In this way, alternatives with higher utility are more likely to be chosen, while behavioral randomness is preserved.

Given the logit structure, the seller options and the “do not buy” alternative are evaluated together. Each customer chooses only one alternative and the sum of the probabilities is normalized so that it is always equal to 1:

$$\sum_{j=1}^{J} P_{ij} + P_{i0} = 1$$

Overall, the integration of the seller strategy selection model and the customer decision model provides a coherent framework in which adaptive seller behavior and heterogeneous customer choices together generate dynamic market outcomes.

CHAPTER 4

ANALYSIS OF THE MODEL

In this chapter, the verification of the agent-based model is carried out by comparing the choice mechanism with the theoretical MNL structure. Then, the scenarios designed to answer the research questions—S0 (price-only competition), S1 (price and promotion competition), and S2 (quality-differentiated competition)—are introduced. The effects of these scenarios on price, profit, sales, and the probability of 'no purchase' are examined, with ANOVA tests and sensitivity analyses conducted. Together, these steps verify the model and show how competition, promotion, and quality shape market outcomes. The full implementation code of the model is provided in the Appendix A.

4.1 Micro-level verification of the choice mechanism

ABM is consistent with the theoretical expectations based on multinomial logit introduced in Section 2.1. In this section, the numerical consistency of the utility function used in the agent-based model with the theoretical logit structure is tested. In particular, when all utility inputs such as age, income, promotion, price and quality are given, it is examined whether the probability of choice calculated by the model exactly matches the theoretically derived logit value.

In each test case there is one subjected firm and the firm has different number of competitors as showed in column "number of competitors". The price variable in the model does not represent absolute price, but rather the seller's relative price. Therefore, this variable directly influences the deterministic utility function:

$$U_i = -price_i$$

According to the formula :

When price = 0, the utility is zero (neutral attractiveness),

When price > 0, the product is more expensive and the utility becomes negative,

When price < 0, the product is cheaper and the utility becomes positive.

Each row in Table 1 represents the relative price combination of the subject firm and, if applicable, its competitors. Prices are indicated on a relative scale, where 0 denotes the reference price level and +1/-1 indicates price difference from the reference. As explained in Section 2.1, the multinomial logit model assumes that the probability of selecting an alternative is proportional to the exponential of its deterministic utility. Under these assumptions, the theoretical selection probability of a firm in this setting is calculated using the following multinomial logit formula:

$$P = \frac{e^{-p_{subject\ firm}}}{1 + e^{-p_{subj}} + (n-1) * e^{-p_{competitor}}}$$

It is also necessary to model the competing alternative as “no buy”. For this case the formula is given as:

$$P = \frac{e^{-p_{subj}}}{1 + e^{-p_{subj}}}$$

Table 1. Theoretical and Simulated Probabilities of Buying

Number of Subject Firm	Price of the Subject Firm	Price of the Competitor	Number of Competitor	Theoretical Probability of the Subject Firm	Theoretical Probability of Competitor	Theoretical Probability of No Buy	Simulational Probability of the Subject Firm	Simulational Probability of Competitor	Simulational Probability of No Buy
1	1		0	0.268941	0	0.731059	0.276		0.724
1	0		0	0.5	0	0.5	0.506		0.494
1	-1		0	0.731059	0	0.268941	0.716		0.284
1	0	-1	2	0.134471	0.731059	0.134471	0.133	0.745	0.122
1	0	1	2	0.365529	0.268941	0.365529	0.35	0.269	0.381
1	1	0	2	0.109232	0.593845	0.296923	0.121	0.587	0.292
1	-1	0	2	0.475367	0.349755	0.174878	0.442	0.361	0.197

In order to test this theoretical framework via simulation, the model is run with 1000 homogeneous customers for each combination of subject firm and competitor prices. All customers are defined as homogeneous and only the price variable

influences decisions. Promotion effects, quality effects, randomness (noise) and other behavioral factors (age, income) were excluded in the test environment. In each case, the simulated probability of choice was calculated by measuring how many out of 1000 customers chose the subject firm.

Table 1 presents the probabilities of customers choosing the subject firm, competitors and “no buy” option under different price and number of competitor cases. The theoretical probabilities represent the mathematical estimates obtained from the logit model, while the simulation probabilities reflect the average outcomes of simulation results. Overall, the simulation values are very close to the theoretical results. Observed small differences arise from the natural randomness in the probabilistic nature of the model.

According to Table 1, if the subject firm's price is low, its relative utility increases and its probability of being chosen rises, if the price is high its relative utility decreases and its probability of being chosen falls. Competitor price levels work symmetrically: when a competitor’s price is high, attractiveness of the subject firm increases, when the competitor’s price is low, it decreases. This finding is a probabilistic reflection of the law of demand. The findings confirm the fundamental economic principles that lowering prices increases demand, while competition, although dividing market shares, raises the overall probability of purchase.

The simulation results reflect the theoretical values with high accuracy. The differences are generally very small and are balanced. As shown in Table 1, among the cases examined, the highest absolute difference was observed in the probability of selecting the subject firm, at 3.3%,in contrast, the lowest absolute difference was 0.01% observed in some competitor probabilities. The other differences are mostly concentrated in the range of 0.5%–1.6%.

According to the customer choice mechanism based on the multinomial logit model described in Section 2.2., the probability of choosing each alternative is proportional to the exponential transformation of its deterministic utility score:

$$P(i) = \frac{e^{V_i/\mu}}{\sum_j e^{V_j/\mu}}$$

Here V_i is the utility score of seller i and μ is assumed as equal to 1. The deterministic utility function used in the model is defined as follows:

$$V = \beta_{age} * \text{age} + \beta_{income} * income + \beta_{highpromotion} * \text{high}promotion + \beta_{lowpromotion} * \text{low}promotion + \beta_{price} * price + \beta_{quality} * quality$$

However, in this verification test, utility is based on only the price of the subject firm or competitor. The utility value of the competitor and the "no buy" alternative is assumed to be as 0.

The verification tests presented in this section demonstrate that the model produces consistent customer choices with the theoretical MNL structure. However, it is also necessary to highlight the limitations of one of the core assumptions of the logit model which is referred as the IIA (Irrelevance of Irrelevant Alternatives).

According to the IIA assumption, adding a third similar alternative should not change the relative probability between two options. However, the test results show that this assumption can produce unrealistic outcomes in the model.

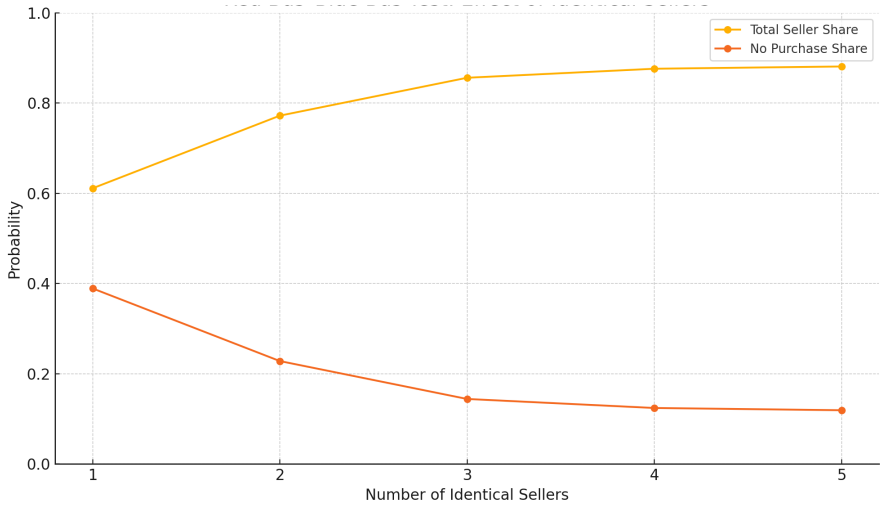

Figure 3. IIA assumption test: effect of identical sellers

As illustrated in Figure 3, when identical sellers are added to the market, the aggregate seller share increases, while the probability of "no purchase" decreases. In reality, since these sellers are indistinguishable, the overall choice probability should remain unchanged. This result clearly demonstrates the limitation of the IIA assumption in the MNL framework. Therefore, although the model successfully replicates the logit structure, the restrictive nature of the IIA assumption must be taken into account when interpreting the results.

4.2 Scenarios

Three scenarios (S0, S1, S2) were designed to address the research questions presented in the Introduction. S0 examines price-only competition, S1 incorporates promotion as an additional decision variable alongside price, and S2 analyzes the impact of quality differences on market dynamics. These three scenarios aim to reveal the effects on prices, profitability, sales, and market coverage. Table 2 summarizes the scenarios designed in this study, the key variables considered in each, and the corresponding research questions.

Table 2. Scenarios

ID	Scenario	Variables	Related Research Question
S0	Price-only competition	Price	RQ1: How does the number of sellers affect equilibrium prices, profitability, sales volume, and overall market coverage in a monopolistic and an oligopolistic FMCG market under the price-only competition setting?
S1	Price & promotion competition	Price, Promotion	RQ2: In a price and promotion competition scenario, how do firms adjust their pricing and promotion strategies over time? How does rising competition affect the balance between promotion efforts and profitability, and can promotions help slow the decrease in sales and expand market coverage compared to a price-only setting?
S2	Quality-differentiated competition	Price, Quality	RQ3: In a quality-differentiated competition setting, how does a seller's product quality influence competition, pricing decisions, profitability, and sales volume?

General settings for all scenarios are listed below:

- Total simulation time: 120 time steps.
- Each scenario is replicated 20 times.
- The number of consumers is 1000.
- The no-buy option is enabled, allowing consumers to opt out of purchasing.
- Consumer demographic characteristics are randomly assigned using normal distributions:
    - Age: Mean = 25, Standard Deviation = 5
    - Income: Mean = 30, Standard Deviation = 8
- The baseline values that normalize utility to zero are defined as $price_0 = 50,\ age_0 = 25,\ and\ income_0 = 40.$

- β_p=−0.1 (higher prices reduce utility)
- β_a=−0.01 (older age reduces utility)
- β_y=0.05 (higher income increases utility)
- The utility of the no-buy option is zero.
- Unit cost is 25.
- During the first 20 time steps, each seller sets random prices uniformly distributed between 30 and 70 to collect sales and profit data.
- Price updates are applied every time step
- Utility scores are transformed into choice probabilities using the multinomial logit function, and each consumer makes a probabilistic decision.

4.2.1 S0 - baseline simulation scenario: price - only competition

This section presents the baseline scenario designed for the initial validation of the model. The aim is to observe the behavior and outputs of the model in a market environment where only the price is varied, while promotion and quality are excluded. This setup serves as a reference scenario to understand how the model operates under simplified conditions before analyzing more complex strategic behaviors. Specific scenario setup parameters for S0 are listed below:

- The only decision variable is the price of the product. Prices are adjusted for every time step after the 20th time step (time step > 20) so as to maximise profit.
- 1, 2, 3, 4, and 5 seller cases are simulated separately.
- Starting from time step 21, sellers use the last 20 time steps of price and profit history to build a second-order polynomial regression model whose

independent variable is price and dependent variable is realized profit at every time step.

- Five price candidates are tested as it is mentioned in Section 3.3.3.1. Strategy Selection using 80%, 90%, 100%, 110% and 120% of the most recent price. The one that yields the highest predicted profit is selected as the next price.

The utility score of each consumer for each seller is calculated as follows:

$$U_{ij} = \beta_p * (\text{price}_j - \text{price}_0) + \beta_a * (\text{age}_i - \text{age}_0) + \beta_y * (\text{income}_i - \text{income}_0)$$

Each seller's profit at a given time step is calculated as:

$$Profit_j = (Price_j - cost) * Sales_j$$

For each number-of-sellers case, average price time series are plotted in Figure 4. Each panel represents a scenario with 1 to 5 sellers, with the horizontal axis showing time steps and the vertical axis showing the prices set by the sellers. For the 1-seller case, the plots also include ±1 standard deviation bands.In oligopoly scenarios, each seller's price development is plotted in a different color. This scenario allows detailed observation of how the customers and sellers behave and learn in a pure price-based competition setup, and serves to validate the strategy update mechanism that will be used in subsequent scenarios.

As the number of sellers increases, downward trend in price levels is observed clearly. Figure 4 shows that, during the first 20 time steps, prices are randomly determined, leading to high volatility, then, prices are updated based on the pricing strategy. For all cases, prices tend to decline after the 20th time step (time step > 20) and stabilize at a certain equilibrium level. As the number of sellers increases, the equilibrium price level becomes lower. The case with 1 seller (monopoly), prices stabilize around 46 over time.

Whereas in the cases where the total number of firms is 3 and 5 , these values decrease to the ranges of approximately 42–44 and 38–40, respectively. This decrease is also confirmed in the “price” row of Table 3. For instance, when there are 5 sellers, each seller’s price falls within the 38.42 to 38.48 range.

In the same way, the average profit per seller has diminished significantly. Figure 5 shows the profit dynamics over time for different number of sellers. The horizontal axis indicates time steps, while the vertical axis shows profit values. In monopoly, the average profit and its ±1 standard deviation band are presented. In oligopoly, each seller’s profit is shown in a different color. As shown, the profit level diminishes to about 2200–2400 units when there are 5 sellers, whereas it reaches about 9900 units when there is only one seller. Table 3 also shows this decrease in profit. Price margins narrow in highly competitive markets, and the distribution of customers among sellers results to lower sales volumes per seller, both of which have an adverse impact on profitability.

Another notable outcome is the decline in average sales per seller as the number of sellers increases, reflecting a reduction in individual market share. However, total market sales increase due to lower prices attracting a greater number of customers. This result suggests that competition contributes positively to overall market coverage by encouraging broader customer participation.

Table 3 presents the sales volume, price, and profit values for each seller, and the number of “not buy” for each total number of seller cases. The “Total Number of Sellers” column indicates the number of sellers in the scenario, while the “Variable” column specifies the type of measured metric such as sales, price, or profit. The columns from “Seller 1” to “Seller 5” display the corresponding values for each seller. Each cell shows the average of the values obtained in the last time step across

replications, with the standard deviation of the related variables given in parentheses. The final column, “Number of ‘Not Buy’,” shows the number of customers who did not make a purchase in the last time step, again reported as the mean of replications and standard deviation.

"no-buy" probability, which is another important finding relating to customer behavior, falls sharply as the number of sellers increases, ranging from approximately 0.561 in the monopoly case to about 0.102 in the five-seller case, shown in Table 3's rightmost column. But as more sellers are added, the rate of decrease slows down indicating that the market is getting close to saturation and that each new seller's marginal contribution is decreasing.

The simulations summarized in Table 3, Figure 4 and Figure 5 show that competition promotes customers to buy, decreases prices, decreases profitability, and helps maintain the stability of the market.

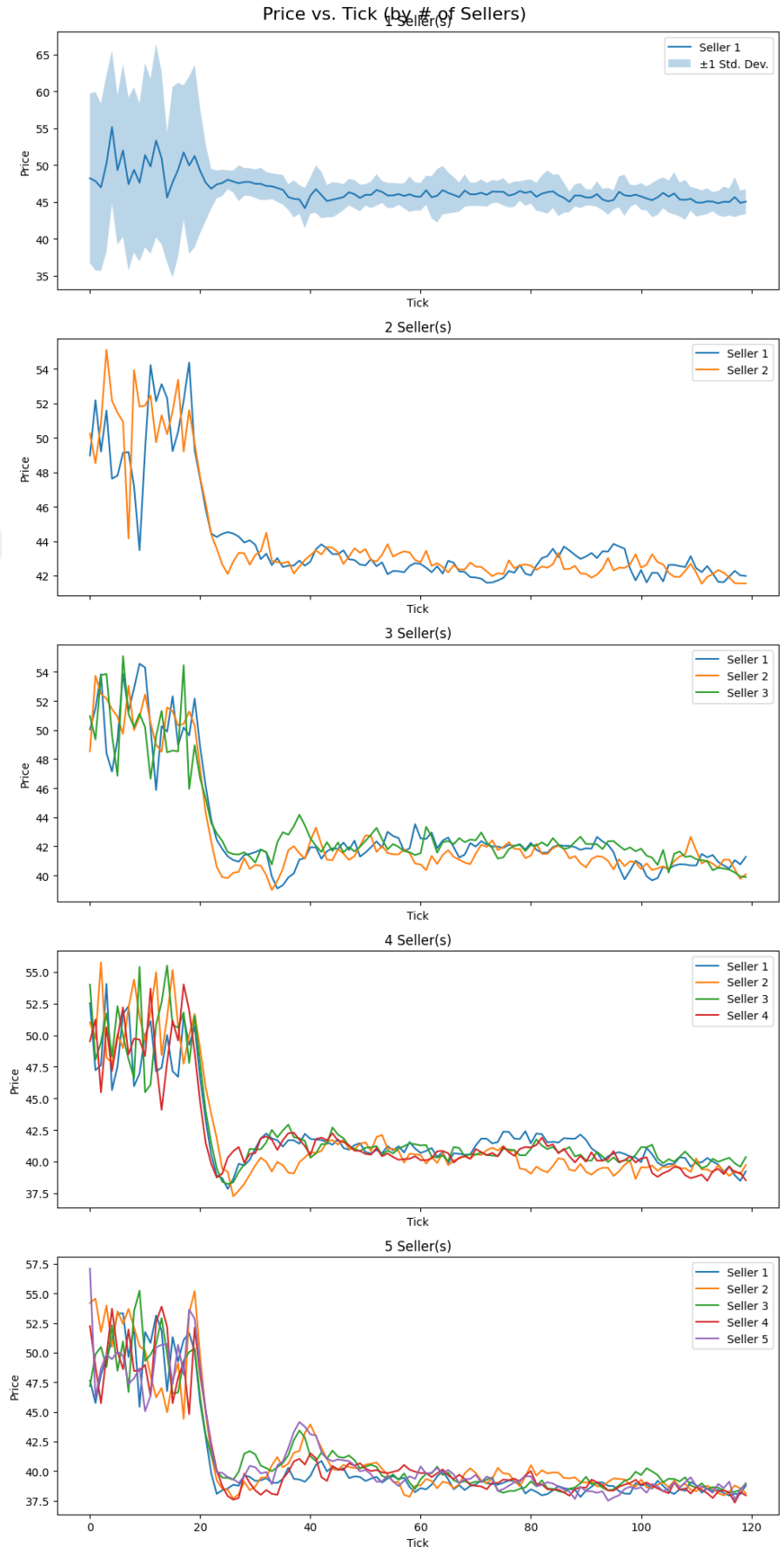

Figure 4. Price as a function of time by number of sellers

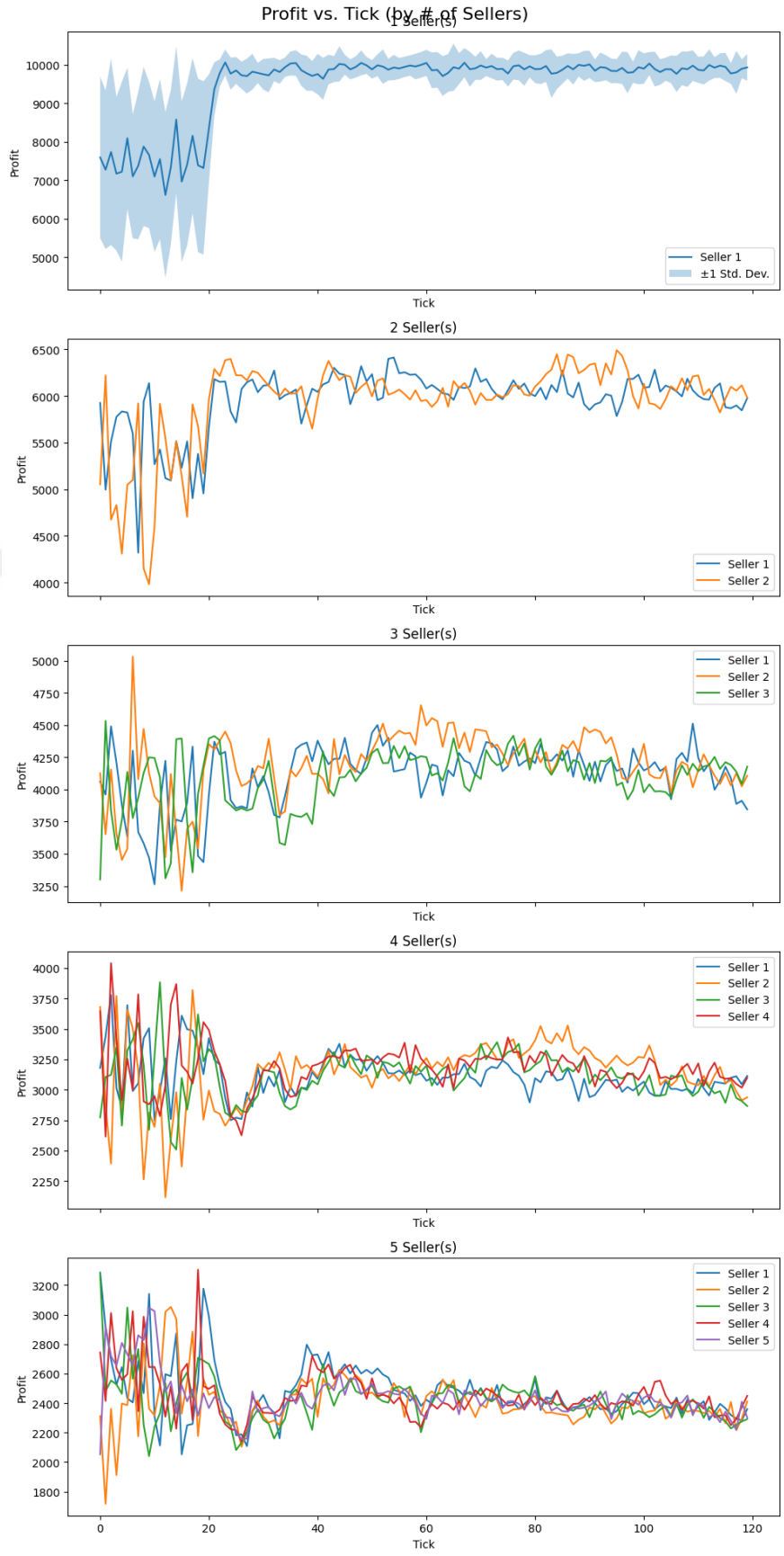

Figure 5. Profit as a function of time by number of sellers

Table 3. Results of Baseline Simulation Scenario

Total Number of Sellers	Variable	Seller 1	Seller 2	Seller 3	Seller 4	Seller 5	Number of "Not Buy"	Theoretical Price
1	sales	501.9 (56.1)					498.1 (56.1)	45
	price	46.21 (1.64)						
	profit	9867.49 (245.58)						
2	sales	357.8 (105.5)	342.2 (83.9)				300.01 (53.8)	40.98
	price	41.52 (2.92)	42.47 (1.66)					
	profit	6051.61 (475.57)	5874.78 (664.02)					
3	sales	283.1 (103.6)	276.0 (84.5)	239.6 (101.5)			201.3 (73.1)	38.93
	price	41.03 (3.41)	41.87 (3.54)	41.49 (2.86)				
	profit	4288.87 (737.24)	4183.14 (773.07)	4310.30 (681.97)				
4	sales	218.1 (57.0)	201.4 (48.6)	222.6 (52.0)	221.2 (40.4)		136.7 (24.0)	37.87
	price	40.03 (4.35)	39.29 (2.39)	39.74 (3.23)	39.25 (2.99)			
	profit	2887.70 (523.16)	3037.68 (503.72)	3054.96 (568.99)	3026.66 (504.37)			
5	sales	183.9 (42.4)	180.2 (28.7)	185.3 (38.3)	170.3 (34.4)	178.1 (43.6)	102.2 (17.8)	37.24
	price	39.88 (2.44)	39.16 (3.39)	37.80 (2.25)	38.02 (1.66)	38.10 (3.85)		
	profit	2270.01 (359.90)	2313.98 (413.79)	2392.48 (339.04)	2437.53 (363.93)	2320.01 (425.15)		

The simulated model is validated by comparing the simulated results with the theoretically computed prices based on homogeneous customers. As shown in the

rightmost column of Table 3, the “Theoretical Price” values are calculated from the analytical Nash equilibrium solution of the game. The simulation results indicate that the simulated prices converge closely to these theoretical equilibrium prices, confirming the model’s validity.

It is expected that, sellers operating in the same market setting have similar pricing and profitability levels due to the homogeneity of the sellers in terms of the characteristics of the products they sell. In order to test this expectation, a one-way ANOVA was conducted for the market scenarios with 3 and 5 sellers. These expectations are tested by the following hypothesis:

H_0: The mean prices and profits are equal.

H_1: At least one seller’s mean price and profit is different.

Table 4. ANOVA Analysis of Baseline Scenario

Number of Sellers	Variable	F(df1, df2)	p-value	η^2	Decision ($\alpha = 0.05$)
3	Price	F(2, 57) = 0.8309	0.44086	0.0551	Fail to reject H_0
3	Profit	F(2, 57) = 1.0461	0.35794	0.0684	Fail to reject H_0
5	Price	F(4, 95) = 0.9841	0.42007	0.0398	Fail to reject H_0
5	Profit	F(4, 95) = 0.1960	0.93994	0.0082	Fail to reject H_0

Table 4 shows the results of the one-way ANOVA for price and profit in the market scenarios with three and five sellers. The table includes the F-statistic, p-value, effect size (η^2), and the decision at the 5% significance level. For both price and profit, all p-values are higher than 0.05. It means there are no statistically significant differences between sellers. The small effect sizes also show that there are very few differences between sellers and that the observed variance is mostly the result of random within-scenario variation. These results support the expectation that sellers with similar products in the same market tend to have similar prices and

profitability levels. S0 is modified by introducing heterogeneity among firms varying the unit cost structure.

Table 5. Sensitivity Analysis of Baseline Scenario

Total Number of Sellers	Variable	Seller 1	Seller 2	Seller 3	Seller 4	Seller 5	Number of "Not Buy"
3	price	36.81 (3.51)	40.20 (2.97)	49.41 (5.66)			
	profit	8534.20 (1071.14)	4387.42 (781.37)	1572.25 (393.74)			187.1 (39.2)
	sales	398.4 (62.6)	293.7 (47.5)	120.8 (39.6)			
5	price	31.41 (3.76)	35.55 (2.58)	38.92 (2.92)	44.05 (2.93)	47.06 (2.17)	
	profit	5413.55 (704.14)	3454.07 (491.60)	2185.29 (285.47)	1293.18 (247.24)	835.39 (162.99)	93.6 (20.2)
	sales	346.9 (79.7)	228.1 (49.6)	162.8 (36.8)	97.2 (29.8)	71.4 (17.5)	

Table 5 presents the effects of different cost structures on price, profit, sales volume, and “no-buy” values in market with three and five sellers. Sellers produce the same product at different costs, therefore, these cost differences do not affect customer utilities. Sellers’ unit costs are evenly distributed between 20 - 30 for 3-seller case and between 15 - 35 for 5-seller case. Higher cost levels lead to higher prices, lower profits, and reduced sales volumes. In 3-seller case, the lowest-cost seller achieves the highest profit and sales volume, whereas the highest-cost seller records substantially lower profit and sales. A similar pattern is observed in the five-seller case, where sellers with lower costs set more competitive prices, achieve broader market coverage, and have lower “no-buy” rates. These findings indicate that different cost structures directly shape competitive advantage, with lower-cost sellers enjoying significant benefits in price competitiveness, sales, and profitability.

S0 stress test was conducted with 10,000 customers and 250 sellers over 250 ticks and 10 replications. The results show that the average price was 36.137, the average profit per seller was 410.50, and the average sales per seller were 39.93. The total number of no-buy customers in the market at the last time step was 19. These findings indicate that even under large-scale settings, the core dynamics of Scenario S0 are preserved: prices converge toward relatively low levels, sales per seller remain limited, and the no-buy approaches to zero.

4.2.2 S1 – extended simulation scenario: price and promotion competition

This section presents an extended version of the baseline scenario where sellers compete not only by price but also by promotion levels. The purpose is to analyze how the introduction of promotional strategies influences market outcomes such as price dynamics, profitability, and amount of sales.

Scenario Setup Parameters

- Three promotion levels are available:
    - None (0)
    - Low (1) promotion adds 0.2 to utility and 2.5 to the base cost per unit sold.
    - High (2) promotion adds 0.5 to utility and 5.0 to the base cost per unit sold.
- Both price and promotion level are decision variables.
- The no-buy option is enabled, allowing consumers to abstain from purchasing.
- Number of sellers: 1, 3, 5, 10, 20, and 50 cases are simulated separately.

The utility score U_{ij} of consumer i for seller j is computed as:

$$U_{ij} = \beta_p * (\text{price}_j - \text{price}_0) + \beta_a * (\text{age}_i - \text{age}_0) + \beta_y * (\text{income}_i - \text{income}_0) + \beta_{lp} * \text{lowpromo}_j + \beta_{hp} * \text{highpromo}_j$$

Where:

- $\beta_{lp} = 0.2$ for low promotion
- $\beta_{hp} = 0.5$ for high promotion
- From time step 21 onward, sellers use the last 20 time steps of their own price, promotion, and profit history to train a second-degree polynomial regression model whose independent variable is price and promotion level and dependent variable is realized profit.
- Five price candidates(80%, 90%, 100%, 110% and 120% of the most recent price) and three promotion levels(no promotion, low promotion and high promotion) are tested as it is mentioned in Section 3.2.2.1. Strategy Selection using. The strategy with the highest predicted profit is selected for the next time step.

This mechanism enables adaptive behavior, where sellers explore, learn, and converge toward more profitable strategies over time.

Each seller's profit at time step t is calculated as:

$$Profit_j = (Price_j - cost) * Sales_j - Promotion\ Cost_j * Sales_j$$

This scenario provides a dynamic environment to test how firms behave under joint price and promotion competition, and how competitive intensity (number of sellers) shapes strategic adaptation.

Appendix B presents sales volume, price, profit, and promotion strategy for each seller, and the number of "not buy" for each total number of seller cases. There are two values in the cells of sales, price, and profit rows. The value on top represents the mean, while the value in parentheses indicates the standard deviation.

These values are calculated for the corresponding variable at the final time step across replications. Promotion rows show frequency distribution of promotions with respect to replications. Appendix B shows that increased number of sellers effects promotion strategies of sellers. Seller did not offer any promotions in 10 out of 20 periods, applying high promotions 9 times and low promotion 1 time in monopoly. Similarly, in the three-seller case, there is no promotion in approximately 35%, and there are high promotions in 46% and low promotions in 18% of the periods. Promotions became central to the marketing strategy under moderate competition such as 5-seller case. The high promotion rate increased to 46%, the low promotion rate to 17%, and the share of no-promotion periods fallen to 36%. When number of sellers is 10, high promotion rate reached approximately 50%, no-promotion share fell to 35%. In competitive markets (10, 20, and 50 sellers), sellers are modeled as homogeneous such as smaller ones, but reporting individual results becomes redundant because of the increased number of sellers. Therefore individual-level results are not reported, and only average outcomes are presented to reflect overall market dynamics. Standard deviation of prices and profits are also indicated in Appendix B which shows the dispersion of these values. In highly competitive markets (markets with 20 and 50 sellers), more balanced promotion strategies are observed. The high promotion rate, which was highest with 10 sellers, decreased to 43% in 20-sellers case and 41% in 50 sellers case. Also, Appendix B indicates that in markets with over 20 sellers, promotional activities occurred during 66% of all replications, demonstrating that such sales approaches have effectively become standard practice in these market conditions.

As expected in S0, sellers operating in the same market setting have similar pricing and profitability levels, given the homogeneity of the sellers in terms of the characteristics of the products they sell. In order to test this expectation, a one-way ANOVA was conducted for the market scenarios with 3 and 5 sellers. Table 6 reports the results of this analysis for the price and profit variables. These expectations are tested by the following hypothesis:

H_0: The mean prices and profits are equal.

H_1: At least one seller's mean price and profit is different.

Across all cases, the p-values are above the 0.05 threshold, and therefore the null hypothesis cannot be rejected, price and profit show no statistically significant differences between sellers. Effect sizes (η^2) are consistently low, confirming that most of the observed variance is due to within-scenario randomness rather than systematic differences between sellers. These findings demonstrate that, consistent with the assumption of homogeneous sellers, firms competing under both price and promotion settings exhibit similar pricing and profitability outcomes.

Table 6. ANOVA Analysis of Scenario 1

Number of Sellers	Variable	F(df1, df2)	p-value	η^2	Decision ($\alpha = 0.05$)
3	Price	F(2, 57) = 0.0656	0.93654	0.0046	Fail to reject H_0
3	Profit	F(2, 57) = 0.3739	0.68971	0.0256	Fail to reject H_0
5	Price	F(4, 95) = 0.8771	0.48078	0.0356	Fail to reject H_0
5	Profit	F(4, 95) = 0.4478	0.77373	0.0185	Fail to reject H_0

In addition to the main results of the S1 scenario, a set of sensitivity analyses were conducted in order to test the robustness of the findings against variations in

effects of promotion on utility and promotion costs. These analyses are divided into two sets. First set focuses on 1-seller cases, and the second one focuses on 2-seller cases. Sensitivity analyses are conducted by varying the effects of low promotion (β_{lp}) and effects of high promotion (β_{hp}) on utility, the low promotion cost (PromoCost_{lp}), and the high promotion cost (PromoCost_{hp}). These parametric variations are tested in order to examine how changes in utility coefficients of promotions and promotion costs for sellers are reflected in market outcomes such as prices, sales volumes, profitability, and the distribution of promotional strategies. The parameter combinations of first set (S1- SA1) for the base case and the four sensitivity cases are summarized in Table 7.

Table 7. Sensitivity Analyis of S1- SA1

Effect of Low Promotion on Utility	Effect of High Promotion on Utility	Cost of Low Promotion	Cost of High Promotion	Average Price	Average Profit	Average Sales	Distribution of Promotion	Average No Purchase
0.2 (Base)	0.5 (Base)	2.5	5	48.01	9852	497	9×High, 1xLow, 10×None	487.05
0.2	0.9	2.5	5	52.91	11883	520	19×High, 1×None	480
0.2	0.5	5	20	46.51	9798	468	19×None, 1×Low	532
0	0.9	0	0	50.60	14805	582	20×High	418
0	0	0	10	46.23	9868	467	13×Low, 7×None	533

In the base case, where utility coefficient of low promotion is modest and utility coefficient of high promotion moderate ($\beta_{lp}= 0.2$, $\beta_{hp}= 0.5$) with low promotion costs ($\text{PromoCost}_{lp}= 2.5$, $\text{PromoCost}_{hp} = 5$), prices stabilize around 48, sales volumes remain close to 487, and profits average about 9853. Promotion

strategies are split almost evenly between no promotion and high promotion, while low promotion is nearly zero. This indicates that under moderate utility coefficients of promotions and moderate costs, sellers tend to prefer either high promotions or no promotions. In first sensitivity case, effect of high promotion on utility is increased (β_{lp}= 0.2, β_{hp}= 0.9), while costs remain as in the base case (PromoCost_{lp}= 2.5, PromoCost_{hp} = 5). In this case, seller performance improves: prices increase to 52.91, sales reach 520, and profits rise to 11883. Almost all replications end with high promotion as the chosen strategy, showing that when consumers get higher utility from high promotion and costs are low, firms usually prefer high promotions. Second sensitivity case represents higher promotion costs (PromoCost_{lp}= 5, PromoCost_{hp} = 20) with base case utilities. Under these conditions, sellers almost stop using promotions, with 19 out of 20 runs showing no promotion. Prices fall to 46.51, sales drop to 468, and profits go down to 9798. More customers decide not to buy, which shows that when costs are too high, firms avoid promotions and both demand and profits decrease.

The third sensitivity case shows the situation where promotion costs are eliminated (PromoCost_{lp}= 0, PromoCost_{hp} = 0) and the utility coefficient of low promotion is zero (β_{lp}= 0), while the utility coefficient of high promotion remains strong (β_{hp}= 0.9). Under these conditions the most advantageous results are observed: prices average 50.60, sales volumes rise to 582, and profits reach 14805, the highest among all cases. In every replication, the seller adopted high promotion. These findings indicate that when promotions strongly influence consumer utility and create no cost for firms, their maximum potential is realized.

In contrast, fourth sensitivity case shows the mostt disadvantageous conditions: utility coefficients of promotion are set to zero (β_{lp}= 0, β_{hp}= 0), where

low promotion is costless, but high promotion carries cost ($\text{PromoCost}_{lp} = 0$, $\text{PromoCost}_{hp} = 10$). Under these settings, prices fall to 46.23, sales decline to 467, and profits remain at 9868, close to the base case but with weaker promotional dynamics. Firms choose mainly the low promotion or no promotion strategies. This illustrates that when promotions have no effect on consumer utility but still have cost, seller faces inefficient strategic trade-offs. Overall, the sensitivity analysis for 1-seller case shows that market outcomes are highly dependent on the balance between the effect of promotion on utility and promotion costs.

The parameter combinations of second set (S1- SA2) for the base case and two sensitivity cases are summarized in Table 8. In base case, utility coefficients of low promotion is modest and utility coefficients of high promotion is moderate ($\beta_{lp} = 0.2$, $\beta_{hp} = 0.5$), Promotion costs are set to 2.5 units for low promotion and 5 units for high promotion. Under these conditions, average prices stabilize around 47.65 and 45.26, while sales volumes remain close at 343 and 365. Profits reach 5403 and 6111, indicating no significant asymmetry between the sellers. Sellers mainly preferred high promotion or no promotion, with little use of low promotion. These findings indicate that when promotions have a moderate effect on customer utility and involve some cost, sellers are more likely to choose either high promotion or no promotion, while low promotion is rarely preferred.

Second and third cases include asymmetry in promotion costs between sellers. In these cases, Seller 1 faces no promotion costs and Seller 2 faces a cost of 15 units for high promotion costs 15 while low promotion remains costless. In second case, effects of promotion on utility remain same as the base case ($\beta_{lp} = 0.2$, $\beta_{hp} = 0.5$). Under these conditions, Seller 1 almost entirely chooses high promotion, while Seller 2 is largely restricted to low promotion. This outcome shows how cost

asymmetry pushes firms towards divergent strategies, giving the costless seller a clear advantage. In third case, utility coefficients of promotion are changed (β_{lp}= 0.1, β_{hp}= 0.9) while costs are the same as case 2. With the same asymmetric costs, Seller 1 almost entirely prefers high promotion while Seller 2 continues to prefer almost entirely low promotion. Compared to base case, second and third cases diverge but they are very similar to each other. This is mainly caused by the same cost structure. In the second case, the asymmetry in promotion costs already led Seller 1 to prefer high promotion and Seller 2 to prefer low promotion. In Case 3, even though effect of high promotion on utility increases and effect of low promotion on utility decreases, the same cost structure keeps sellers in the same strategies. Therefore, the distribution of promotion strategies remains almost the same, while the main differences appear in profits, sales, and no-purchase rates.

Table 8. Sensitivity Analyis of Scenario 1 for S1- SA2

Effects of Low Promo on Utility	Effects of High Promo on Utility	Seller	Cost of Low Promo	Cost of High Promo	Average Price	Average Profit	Average Sales	Distribution of Promotion	Average No Purchase
0.2	0.5	Seller 1	2.5	5	47.65	5403.22	343.30	3×None, 3×Low, 14×High	292.00
		Seller 2	2.5	5	45.26	6110.62	364.70	9×None, 1×Low, 10×High	
0.2	0.5	Seller 1	0	0	46.46	8079.84	390.50	1×Low, 19×High	281.25
		Seller 2	0	15	45.11	6504.06	328.25	1×None, 19×Low	
0.1	0.9	Seller 1	0	0	46.79	9016.44	438.70	1×None, 1×Low, 18×High	235.80
		Seller 2	0	15	42.89	5290.90	325.50	1×None, 19×Low	

4.2.3 S2- quality-differentiated competition

In this scenario, each seller produces the product with a distinct quality level evenly distributed between −0.5 and 0.5. For example, if there are three sellers, product quality ranges from −0.5 for Seller 1 to 0.5 for Seller 3, with intermediate values assigned proportionally. This introduces heterogeneity among sellers. The quality of the product affects unit cost which is modeled as a linear function of quality:

$cost_j = C_{base} + costq * Quality_j$

Where:

- $C_{base} = 25$ (base unit cost)
- $costq = 10$ (additional cost per unit of quality)
- $Quality_j =$ intrinsic, unchanging quality level assigned to seller j (between -0.5 and 0.5)

Scenario setup parameters:

- minimum quality= -0.5 (minimum quality value)
- maximum quality = 0.5 (maximum quality value)
- β_q= 1.0 (coefficient of quality in the utility function)
- $costq_j$ = 10.0 (additional unit cost per unit of quality)
    - The no-buy option is enabled, allowing consumers to abstain from purchasing.
    - Number of sellers: 1, 3, 5, 11, 21, and 51 cases are simulated separately. These numbers were selected to preserve symmetry in the quality distribution between −0.5 and 0.5 with a midpoint at zero, which is ensured by using odd numbers.

In addition to price and demographic factors, the quality of the product also affects the utility. The utility score U_{ij} of consumer i for seller j is computed as:

$$U_{ij} = \beta_p * (\text{price}_j - \text{price}_0) + \beta_a * (\text{age}_i - \text{age}_0) + \beta_y * (\text{income}_i - \text{income}_0) + \beta_q * quality_j$$

- From time step 21 onward, sellers use the last 20 time steps of their own price, and profit history to train a second-degree polynomial regression model whose independent variable is price and dependent variable is realized profit.
- Five price candidates (80%, 90%, 100%, 110%, and 120% of the most recent price) are tested, as mentioned in Section 3.2.2.1, *Strategy Selection*. The one that yields the highest predicted profit is selected as the next price.

Each seller's profit at time step t is calculated as:

$$Profit_j = (Price_j - cost) * Sales_j - costq_j * quality * Sales_j$$

Table 9 summarizes statistics from the simulations for different numbers of sellers. It lists average sales per seller, prices, profits, and the number of customers who did not buy in each case. The table shows that the "no-buy" value is around 543.70 in the case of a monopoly, decreases to 194 in 3-sellers case, and further decreases to 106.20 in 5-sellers case. This decrease indicates that different quality–price combinations lead customers to find suitable options, and the heterogeneity enables greater market participation. Figure 6 shows the variation of sales, profits and prices as a function of quality for competitive markets (total number of sellers ≥ 11). Each panel in Figure 6(a) shows a seller's sales volume against its product quality for different market sizes. It shows that there is no consistent relationship between quality and sales. This suggests that sales are affected not only by product quality but also by pricing strategies and competitive conditions.

Table 9. Results of Scenario 2

Total Number of Sellers	Variable	Seller 1	Seller 2	Seller 3	Seller 4	Seller 5	Number of "Not Buy"
1	sales	456.30 (36.98)					543.70 (36.98)
	price	46.55 (1.79)					
	profit	9772.96 (386.62)					
	quality	0					
3	sales	271.45 (79.43)	258.05 (60.28)	276.50 (51.13)			194.00 (31.64)
	price	36.45 (4.77)	41.93 (2.19)	46.14 (2.56)			
	profit	4108.06 (634.00)	4288.06 (829.75)	4356.01 (548.08)			
	quality	-0.5	0	0.5			
5	sales	171.75 (22.39)	191.20 (51.20)	179.00 (41.05)	171.25 (34.56)	180.60 (48.91)	106.20 (18.18)
	price	34.01 (2.19)	35.50 (2.53)	38.87 (3.04)	41.44 (2.08)	43.45 (4.09)	
	profit	2369.97 (276.80)	2380.35 (373.20)	2378.34 (339.97)	2327.55 (260.93)	2248.90 (317.34)	
	quality	-0.5	-0.25	0	0.25	0.5	

Each point in Figure 6(b), shows a seller's profit against its product quality for different market sizes. As shown in Table 7, in the 3-seller case all sellers earn positive profits, but there are small differences between sellers. Their profits range approximately between 4108 and 4356. This indicates that although the highest-quality seller charges a higher price, the profit advantage remains limited.

In the 5-seller case, profits are compressed into a narrower range: between 2248 to 2380 as presented in the Table 7. Figure 6(b) also reflects this pattern: with 11, 21, and 51 sellers, profit points fluctuate, but profits decline for all and differences between sellers shrink.

In all cases, it is clearly shown in Figure 6(c) that sellers offering higher-quality products set higher prices, while sellers offering lower-quality products set lower prices. In other words, as quality increases, prices tend to increase as well. Even though this tendecy is very clear in less competitive markets, it appears in more competitive markets aswell. The general relationship between product quality and price is clearly maintained.

Figure 6. Effect of Quality on Sales Profit and Price in Competitive Markets

It is expected that sellers in 3-seller case and 5-seller case will adopt different optimal pricing strategies and achieve different profitability levels because of different quality attributes. Therefore, the one-way ANOVA tests for S2 are expected to reveal statistically significant differences in both mean prices and mean profits among sellers. These expectations are tested by the following hypothesis:

H_0: The mean prices and profits are equal.

H_1: At least one seller's mean price and profit is different.

Table 10. ANOVA Analysis of Scenario 2

Number of Sellers	Variable	F(df1, df2)	p-value	η^2	Decision ($\alpha = 0.05$)
n = 3	Price	F(2, 57) = 44.4328	2.34×10^{-12}	0.7572	Reject H_0
n = 3	Profit	F(2, 57) = 0.5764	0.5652	0.0389	Fail to reject H_0
n = 5	Price	F(4, 95) = 49.7664	1.63×10^{-22}	0.6769	Reject H_0
n = 5	Profit	F(4, 95) = 0.5907	0.6702	0.0243	Fail to reject H_0

The ANOVA results showed in Table 10 indicate that, under quality heterogeneity in S2, sellers' prices differ significantly in both the 3-seller and 5-seller cases, with large effect sizes showing that higher-quality sellers consistently charge higher prices. In contrast, profits remain not significantly different across sellers in both cases. Small η^2 values means that price increases from quality do not lead to higher profits because costs rise as well. Taken together, these results demonstrate that while quality heterogeneity leads to systematic and statistically significant difference in pricing strategies, it does not generate significant difference in profitability among sellers.

In addition to the main S2 settings ($costq_j$ =10.0, β_q= 1.0), sensitivity analysis are conducted. Specifically, $costq_j$values of 2 (low quality cost) and 10 (high quality cost) were combined with β_q values of 1 (strong effect of quality on utility) and 0.3 (weak effect of quality on utility). These parameter variations were tested in order to explore how changes in the cost of quality and effect of quality on

utility, reflected as coefficient of quality in the utility function, influence market outcomes under different cost and utility conditions.

Figure 7. Sensitivity analysis of scenario 2 for $costq_j$=2 and β_q= 1.0

Figure 7 illustrates that when the quality cost is low ($costq_j$=2) and effect of quality on utility is strong (β_q= 1.0), sales volumes increase when quality increases. Even though 51-seller case is noisier due to stronger competition and learning dynamics, the upward trend remains visible. It shows that consumers systematically prefer higher-quality sellers under strong effect of quality on utility. The upward trend also becomes prominent in profits. In contrast to the main S2 setting, the reduction in additional unit cost per unit of quality allows higher-quality sellers to achieve higher profits. However the slope decreases when competitiveness increases.

Prices, similar to sales volume and profits, also increase with quality. However, as the number of sellers increases, the slope of this increase diminishes and some fluctuations are observed.

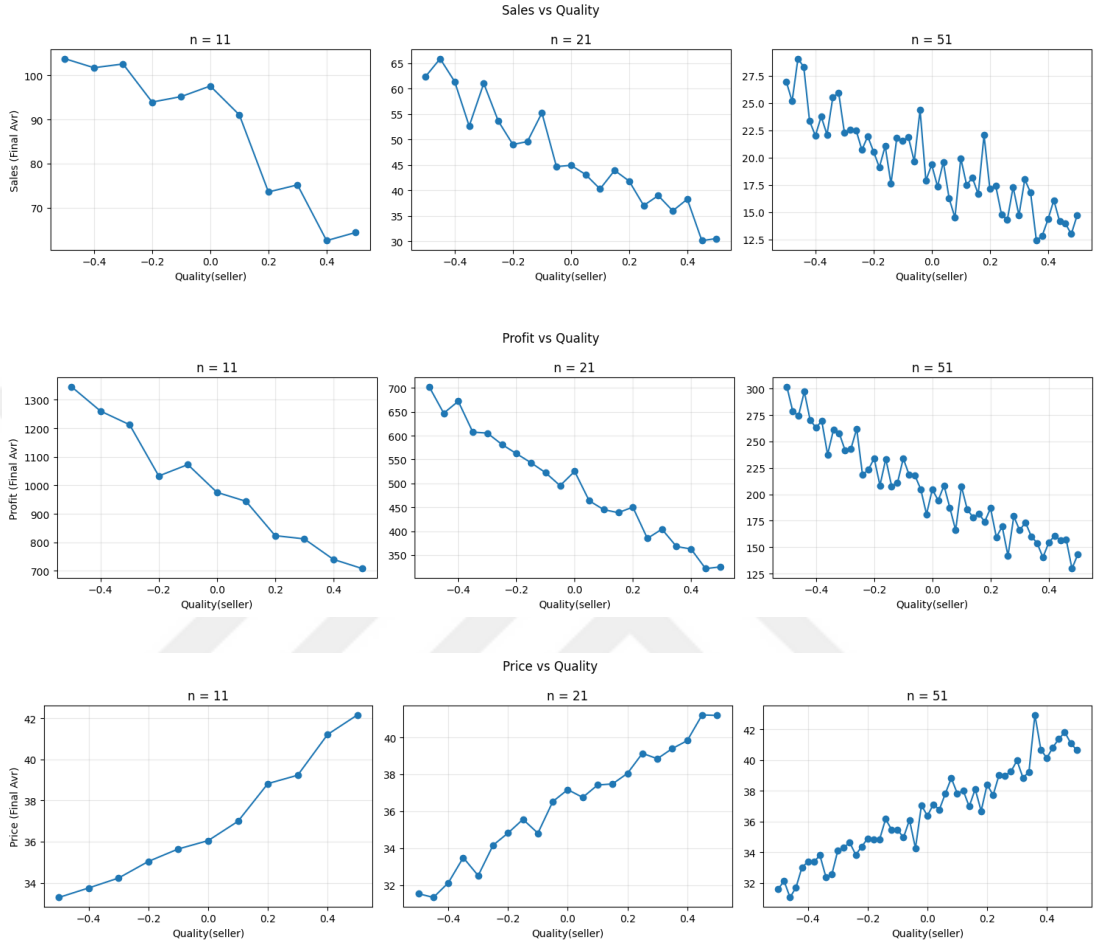

Figure 8. Sensitivity analysis of scenario 2 for $costq_j$=10 and β_q= 0

The simulation results for high quality cost ($costq_j$=10) and weak utility coefficient of quality (β_q= 0.3) are presented in Figure 8. Across all cases (11-seller, 21-seller, 51-seller), increasing quality is related with a decline in sales. With a weak effect of quality on utility, quality becomes less important for consumers. Also, the combination of higher quality and higher prices lowers the probability of buying. In line with this, profits also decline as quality increases. This pattern can be attributed to the high cost of quality, which raises production expenses, and the weak impact of

quality on consumer utility. In contrast, prices exhibit an upward trend with quality, as the increasing cost of production directly drives sellers to set higher price levels.

The simulation results for low quality cost ($costq_j$=2) and weak effect of quality on utility (β_q= 0.3) are shown in *Figure 9*. In 11-seller case, increase in quality leads to partial rise in sales and profit. In 21-seller and 51-seller cases, even though there is a positive relationship between quality and these two dependent variables, the sales and profit series are more volatile and no strong trend is observed. For prices, a positive relationship with quality is observed such as sales and profit. In 11-seller and 21-seller cases, prices rise with quality but with some fluctuations. In the 51-seller case, the general trend is also positive, although there are more fluctuations. This shows that, due to the low cost of higher quality, the increase in quality is reflected softly in prices, but the low quality effect on utility prevents quality improvements from creating strong benefits on customer buying behavior.

The sensitivity results obtained under different parameter combinations of the S2 scenario illustrates three fundamental relationships. These relationships show different patterns depending on the number of sellers and market conditions. In the main S2 setting (high cost, strong utility coefficient of quality), no systematic relationship between sales and quality is observed. Under low cost–strong utility coefficient of quality, sales increase with quality, whereas under high cost–weak utility coefficient of quality, sales decrease as quality rises. In the low cost–weak utility coefficient of quality cases, sales show a slight increase with quality, but the series become more volatile, particularly as the number of sellers increases. In the main S2 setting, profit differences remain limited due to cost pressures. The profit–quality relationship is positive under low cost–strong utility coefficients of quality

and low cost–weak utility coefficient of quality, while it turns negative under high cost–weak utility coefficient of quality. On the other hand, price–quality relationship is positive in all parameter combinations.

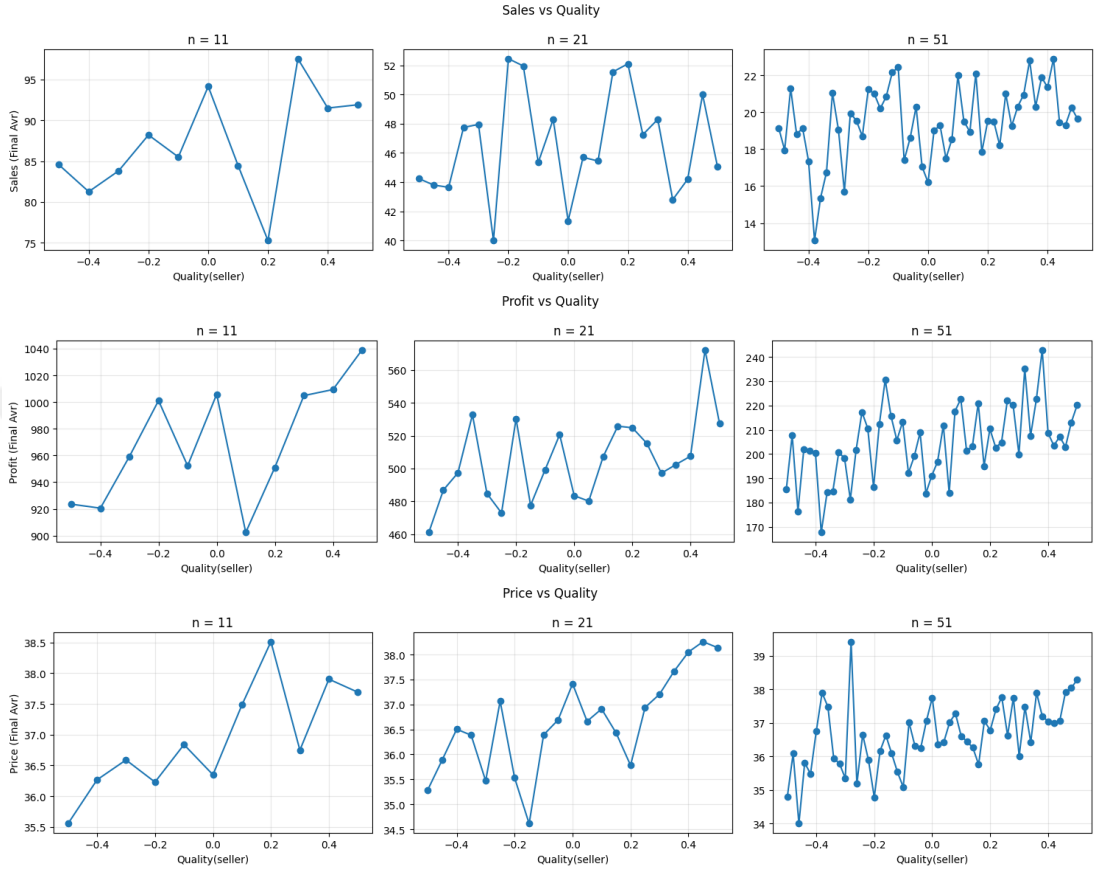

Figure 9. Sensitivity analysis of scenario 2 for $costq_j$=*2* and β_q= *0.3*

Table 11. Overview of Sensitivity Analysis Results of S2

Cost Regime	Response to Quality	Sensitivity ID	Sales	Profit	Price
10 (Base Case)	1 (Base Case)	S2- Sa0	No systematic relationship between sales and quality is observed.	Profits do not differ significantly across sellers.	Prices systematically increase with quality.
2	1	S2- Sa1	Sales increase with quality, and customers clearly prefer higher-quality sellers.	Profits increase with quality, giving higher-quality sellers an advantage.	Prices increase with quality, but the slope diminishes as competition intensifies.
10	0.3	S2- Sa2	Sales decline as quality increases.	Profits decrease with quality due to rising costs.	Prices increase with quality, mainly driven by higher production costs.
2	0.3	S2- Sa3	Sales slightly increase with quality but remain volatile under stronger competition.	Profits slightly increase with quality, though without a stable trend.	Prices increase with quality, but the increase is smoother compared to other cases.

Overall, the findings summarized in Table 11 show that sales–quality and profit–quality relationships vary across parameter conditions and the series become more volatile under high competition. These fluctuations make it difficult to address conclusions based solely on visual analysis. Therefore, to statistically test the effect of quality of sellers on price and profit, linear regressions were conducted for the 21-seller case. For each regression, the following hypotheses were evaluated through the t-test of overall model significance:

- H_0: The regression model is not statistically significant (slope coefficient = 0).
- H_1 : The regression model is statistically significant (slope coefficient ≠ 0).

The regression results in Table 12 confirm the visual patterns and also measure their strength. For the price–quality relationship, all parameter settings give a positive and significant slope. In high cost cases quality explains almost all of the variation in price ($R^2 \approx 0.97$). For low costs explanatory power is moderate ($R^2 \approx 0.46$). This matches the noisier patterns seen in the figures. Overall, the price–quality relationship is stable, but its strength mainly depends on the cost level.

For the profit–quality relationship, the results vary by parameter setting. In High Cost–Strong effect on utility case, the relationship is not significant ($p \approx 0.61$), showing that high costs offset the potential advantages of quality.

In the Low Cost–Strong effect on utility case, profits increase significantly with quality ($p \approx 0.00$), showing that inexpensive and high quality generates significant profit advantages. In contrast, in High Cost–Weak effect on utility case, profits decline with quality. In Low Cost – Weak effect on utility case, profits increase only slightly with quality.

Table 12. Regression Results of Scenario 2

Variable	Sensitivity ID	t	p	Significance	R2	Slope Coefficient
Price	S2- Sa0	26.2431	0	significant	0.9732	11.04
Profit	S2- Sa0	0.5248	0.6058	not significant	0.0143	7.51
Price	S2- Sa1	4.0523	0.0007	significant	0.4636	2.29
Profit	S2- Sa1	20.9349	0	significant	0.9584	400.80
Price	S2- Sa2	24.8449	0	significant	0.9701	10.19
Profit	S2- Sa2	-24.8443	0	significant	0.9701	-337.36
Price	S2- Sa3	4.0502	0.0007	significant	0.4633	2.50
Profit	S2- Sa3	3.2859	0.0039	significant	0.3624	41.80

CHAPTER 5

RESULTS & DISCUSSION

This chapter discusses the results of the simulation experiments. The findings are structured according to the research questions and scenario designs introduced in Chapter 4.

The micro-level verification in Section 4.1 shows that the choice probabilities produced by the agent-based model are very close to the theoretical results of the MNL model. This consistency means the model reproduces the main behavior of the logit framework and supports the validity of scenario analyses. The small differences between simulated and theoretical values are negligible and caused by the inherent randomness of the model. As emphasized in the literature, such micro-level consistency is important because it shows that individual decision rules can be reliably reflected in overall market outcomes (Rand & Rust, 2011). This verification increases the credibility of the study and shows that agent-based models can effectively apply random utility theory in dynamic competitive settings.

Simulations were carried out under three main scenarios. In the baseline scenario (S0), there is only price competition among firms. In Scenario 1 (S1), promotion is introduced alongside price as an additional decision variable. Scenario 2 (S2) introduces quality differences to examine how heterogeneity among products influences market outcomes. Together, these scenarios make it possible to compare the different dynamics that emerge when firms compete through promotion or quality in competition.

In S0 scenario, as competition increased, prices, profits and market shares of sellers decreased, as expected. In contrast, total sales increased and the no-buy rate

fell, indicating that stronger competition enhanced market penetration. An important validation finding in this scenario is that the equilibrium prices reached in the simulation closely matched the theoretically derived equilibrium prices based on homogeneous customers. This alignment shows that the model not only operates consistently within itself but also successfully produces theoretical expectations and thereby strengthens the reliability of the subsequent analyses. This outcome is consistent with logit-oligopoly theory, where increasing the number of firms lowers equilibrium prices and compresses profits while expanding consumer welfare (Anderson et al, 1992).

In S1, promotion is added to S0. Prices fall more sharply and the no-buy rate falls further, which expands market penetration. On the other hand, promotion costs reduce profits and lead to sharper trade-offs compared to price-only competition. Sensitivity analyses show that promotion strategies depend entirely on the balance between effect of promotion on utility and promotion costs. When effect of promotion on utility is strong and costs are low, high promotion becomes dominant. When costs rise or effect of promotion on utility is weak, promotion almost disappears. Therefore, promotion becomes prominent as a tool that increases coverage but carries significant risks for profit margins. This finding parallels prior studies showing that promotions accelerate adoption and market coverage but often at the expense of profitability (Zhang & Zheng, 2019).

Quality differentiation is added to the price competition in S0 for S2 scenario. Higher-quality sellers are observed to set higher prices. But higher prices do not always mean higher profits, because higher quality also brings higher costs. Only when costs are low and effect of promotion on utility is strong, high-quality sellers gain a profit advantage. Compared to S0, another key finding is that quality

differentiation further reduces the no-buy rate and increases the likelihood that customers can find a product that fits their preferences. In this way, quality becomes a lasting factor that sustains price differences, while its impact on profitability depends on the balance between costs and customers' willingness to pay.

When the three scenarios are considered together, a consistent conclusion emerges: stronger competition benefits consumers. In S0, competition reduces price margins and broadens market coverage. In S1, this effect is reinforced by the attractiveness of promotions. In S2, a similar expansion of coverage is achieved through quality differentiation. From sellers' perspective, sustainable profitability depends on the careful and efficient use of promotions in S1, and on quality investments being profitable when willingness to pay exceeds marginal cost in S2.

Overall, these outcomes are in line with logit-oligopoly predictions and are consistent with findings from ABM studies in marketing.

This thesis contributes to the literature by showing that promotion is not only something that increases consumer utility but also a costly and strategic decision for sellers. Previous approaches based on fixed rules, without probabilistic choice, give useful insights but capture less of the realism offered by utility-based models. Here, a multinomial logit (MNL) choice structure is combined with sellers learning in a dynamic and competitive setting, extending the ABM literature on pricing and promotion strategies in competitive FMCG markets (Anderson et al, 1992).

Despite the valuable contributions of this study, there are also limitations. One limitation lies in the treatment of quality. In the current model, sellers' quality levels are fixed, whereas in practice quality could be a strategic decision which is adjusted alongside price and promotion. Therefore, the model abstracts from the dynamic nature of quality adaptation, which may limit the scope of the results.

Another limitation relates to customer representation and data calibration. Demographic structure of customers is represented by normal distributions of age and income. While this simplifies the simulations, it does not capture the diversity of customer segments found in real markets. Testing the model with alternative distributions or explicit segmentation would make the findings more generalizable. Moreover, the model has not been calibrated against real category data, which limits its external validity. Although the theoretical insights are valuable, empirical calibration and validation would be necessary before the framework can be applied as a decision-support tool in practice.

The learning mechanism also presents certain simplifications. Firms adapt their strategies solely based on their own historical outcomes, without incorporating the observable history of competitors such as prices, promotions, or sales. In reality, firms often adjust their strategies according to their competitors' actions.

Finally, the reliance on the multinomial logit (MNL) introduces the well-known independence of irrelevant alternatives (IIA) assumption. This can lead to unrealistic substitution patterns when close alternatives are present. More flexible models, such as nested logit or mixed logit, could alleviate this limitation and provide a richer representation of consumer choice.

The limitations of the study can be summarized as: Dynamic quality updates, alternative demographic settings, and the incorporation of competitor information into firms' learning rules would all add realism and robustness. Calibration with real data and the use of more advanced discrete choice models would further increase the practical value of the findings.

Besides the limitations of this study, there are also several directions for future research. One is to improve how sellers adapt their strategies. Using reinforcement

learning, sellers could update prices and promotions in a more flexible way under changing competition.

On the customer side, the model does not yet include intertemporal decision-making. Adding links between past and current purchases would make customer behavior more realistic and show how loyalty develops over time.

Another useful extension would be to include social networks. Since customer choices can be influenced by peers, modeling these effects would allow testing of network-based marketing strategies by sellers (Karakaya et al, 2011).

Finally, the model could be enriched with leader–follower dynamics. In such a setup, some sellers act as leaders and others follow leaders and makes it possible to study price leadership and promotion signaling.

Together, these improvements would make the model more realistic and increase both its theoretical and practical value by enabling budget allocation for promotions and threshold rules for quality investment.

CHAPTER 6

CONCLUSION

This study developed an agent-based model to analyze pricing, promotion, and quality strategies in fast-moving consumer goods (FMCG) markets. These markets are highly competitive, with frequent purchases and strong price sensitivity, making them a suitable environment to study dynamic seller and customer interactions. The model integrated a multinomial logit choice mechanism with firm learning, allowing both probabilistic consumer decisions and adaptive seller strategies.

The three research questions posed in the Introduction Chapter are answered by the three scenarios: S0, S1, S2 respectively. In S0 (RQ1), competition lowered prices and profits but expanded total sales and reduced no-buy rates, with equilibrium prices converging to theoretical predictions. In S1 (RQ2), promotion increased market penetration but profits declined due to promotion costs, depending on the balance between effect of promotion on utility and costs. In S2 (RQ3), quality differentiation raised prices, but profit advantages occurred only when costs were low and customers valued quality strongly, while heterogeneity in quality further reduced no-buy rates.

The study contributes theoretically by demonstrating how agent-based modeling extends logit-oligopoly analysis to settings with heterogeneity of customers, learning, and trade-offs of price and promotion strategies and product differentiation via quality. It contributes practically by providing a model that firms can calibrate with their own data to test “what-if” scenarios, guiding strategic choices in pricing, promotions, and quality investments.

APPENDIX A

FULL IMPLEMENTATION CODE OF THE MODEL

```
#### TheoreticalvsSimulatedProbabilities ####
import pandas as pd
import numpy as np
data=[{"price_S":1,"price_C":np.nan,"no_of_competitors":0},{"price_S":0,"price_C
":np.nan,"no_of_competitors":0},{"price_S":-
1,"price_C":np.nan,"no_of_competitors":0},{"price_S":0,"price_C":-
1,"no_of_competitors":2},{"price_S":0,"price_C":1,"no_of_competitors":2},{"price
_S":1,"price_C":0,"no_of_competitors":2},{"price_S":-
1,"price_C":0,"no_of_competitors":2}]
df=pd.DataFrame(data)
#Theoretical
def compute_theoretical_probs(row):
    z_subject=-row["price_S"]
    z_no_buy=0
    if row["no_of_competitors"]==0:
        exp_s=np.exp(z_subject)
        exp_n=np.exp(z_no_buy)
        denom=exp_s+exp_n
        return
pd.Series({"theor_prob_subj":exp_s/denom,"theor_prob_C":0.0,"theor_prob_no_buy
":exp_n/denom})
    elif row["no_of_competitors"]==2:
        z_competitor=-row["price_C"]
        exp_s=np.exp(z_subject)
        exp_c=2*np.exp(z_competitor)
        exp_n=np.exp(z_no_buy)
        denom=exp_s+exp_c+exp_n
        return
pd.Series({"theor_prob_subj":exp_s/denom,"theor_prob_C":exp_c/denom,"theor_pr
ob_no_buy":exp_n/denom})
#Simulated
def simulate_probabilities(row,n_customers=1000):
    z_subject=-row["price_S"]
    z_no_buy=0
    if row["no_of_competitors"]==0:
        logits=np.array([z_subject,z_no_buy])
        probs=np.exp(logits-np.max(logits))
        probs/=probs.sum()
        choices=np.random.choice([0,1],size=n_customers,p=probs)
        return
pd.Series({"sim_prob_subj":np.mean(choices==0),"sim_prob_C":0.0,"sim_prob_no_
buy":np.mean(choices==1)})
    elif row["no_of_competitors"]==2:
        z_competitor=-row["price_C"]
        logits=np.array([z_subject,z_competitor,z_competitor,z_no_buy])
```

```
        probs=np.exp(logits-np.max(logits))
        probs/=probs.sum()
        choices=np.random.choice([0,1,2,3],size=n_customers,p=probs)
        return
pd.Series({"sim_prob_subj":np.mean(choices==0),"sim_prob_C":np.mean((choices==1)|(choices==2)),"sim_prob_no_buy":np.mean(choices==3)})
#Probabilities
df[["theor_prob_subj","theor_prob_C","theor_prob_no_buy"]]=df.apply(compute_theoretical_probs,axis=1)
df[["sim_prob_subj","sim_prob_C","sim_prob_no_buy"]]=df.apply(simulate_probabilities,axis=1)
#Table
print(df.to_string(index=False))
#### S0-Figures ####
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
#Parameters
TICKS=120
REPLICATIONS=20
CUSTOMERS=1000
COST=25
LEARNING_START_TICK=20
USE_INITIAL_ONLY=False
#Customer distribution parameters
AGE_MEAN=25
AGE_STD=5
INCOME_MEAN=30
INCOME_STD=8
#Utility function coefficients
BETA_PRICE=-0.1
BETA_AGE=-0.01
BETA_INCOME=0.05
def generate_customers():
   ages=np.random.normal(AGE_MEAN,AGE_STD,CUSTOMERS)
   incomes=np.random.normal(INCOME_MEAN,INCOME_STD,CUSTOMERS)
   return ages,incomes
def softmax(utilities):
   exp_util=np.exp(utilities)
   return exp_util/exp_util.sum(axis=1,keepdims=True)
def simulate_tick(prices,ages,incomes):
   REF_PRICE=50
   REF_AGE=25
   REF_INCOME=40
   n_sellers=len(prices)
   all_utils=np.zeros((CUSTOMERS,n_sellers+1))  # +1 for no-buy
   for i in range(n_sellers):
      reduced_price=prices[i]-REF_PRICE
      reduced_age=ages-REF_AGE
```

```
        reduced_income=incomes-REF_INCOME

all_utils[:,i]=(BETA_PRICE*reduced_price+BETA_AGE*reduced_age+BETA_IN
COME*reduced_income)
    all_utils[:,-1]=0  # no-buy utility
    probs=softmax(all_utils)
    choices=[np.random.choice(n_sellers+1,p=p) for p in probs]
    sales=[choices.count(i) for i in range(n_sellers)]
    return sales
def seller_strategy(history_prices,history_profits):
    recent_price=history_prices[-1]
    poly=PolynomialFeatures(degree=2)
    X_poly=poly.fit_transform(np.array(history_prices).reshape(-1,1))
    model=LinearRegression().fit(X_poly,history_profits)
    candidates=[recent_price*0.9,recent_price,recent_price*1.1]
    preds=[model.predict(poly.transform([[p]]))[0] for p in candidates]
    return candidates[np.argmax(preds)]
def run_single_simulation(n_sellers):
    prices=[np.random.uniform(30,70) for _ in range(n_sellers)]
    price_history=[[] for _ in range(n_sellers)]
    profit_history=[[] for _ in range(n_sellers)]
    ages,incomes=generate_customers()
    for tick in range(TICKS):
        if tick<LEARNING_START_TICK:
            prices=[np.random.uniform(30,70) for _ in range(n_sellers)]
        else:
            for i in range(n_sellers):
                if USE_INITIAL_ONLY:
                    hist_prices=price_history[i][:LEARNING_START_TICK]
                    hist_profits=profit_history[i][:LEARNING_START_TICK]
                else:
                    hist_prices=price_history[i][-LEARNING_START_TICK:]
                    hist_profits=profit_history[i][-LEARNING_START_TICK:]
                prices[i]=seller_strategy(hist_prices,hist_profits)
        sales=simulate_tick(prices,ages,incomes)
        for i in range(n_sellers):
            profit=(prices[i]-COST)*sales[i]
            price_history[i].append(prices[i])
            profit_history[i].append(profit)
    return price_history,profit_history
def run_simulation(n_sellers):
    all_prices=[]
    all_profits=[]
    for _ in range(REPLICATIONS):
        price_hist,profit_hist=run_single_simulation(n_sellers)
        all_prices.append(price_hist)
        all_profits.append(profit_hist)
    return all_prices,all_profits
def plot_5_panels(all_histories,metric_name):
    fig,axs=plt.subplots(5,1,figsize=(10,20),sharex=True)
```

```
    ticks=np.arange(TICKS)
    for seller_count in range(1,6):
        metric=all_histories[seller_count-1]
        ax=axs[seller_count-1]
        for i in range(seller_count):
            seller_data=np.array([rep[i] for rep in metric])
            mean_series=seller_data.mean(axis=0)
            std_series=seller_data.std(axis=0)
            ax.plot(ticks,mean_series,label=f"Seller {i+1}")
            if seller_count==1:
                ax.fill_between(ticks,mean_series-
std_series,mean_series+std_series,alpha=0.3,label="±1 Std. Dev.")
        ax.set_title(f"{seller_count} Seller(s)")
        ax.set_xlabel("Tick")
        ax.set_ylabel(metric_name.capitalize())
        ax.legend()
    plt.suptitle(f"{metric_name.capitalize()} vs. Tick (by # of Sellers)",fontsize=16)
    plt.tight_layout()
    plt.show()
#Main Execution
price_results=[]
profit_results=[]
for num_sellers in range(1,6):
    price_histories,profit_histories=run_simulation(num_sellers)
    price_results.append(price_histories)
    profit_results.append(profit_histories)
#Plotting
plot_5_panels(price_results,"price")
plot_5_panels(profit_results,"profit")
#Tick 120- Average Prices and Profits
print("\n=== Tick 120- Average Prices and Profits ===")
for idx,num_sellers in enumerate(range(1,6)):
    print(f"\n--- {num_sellers} Seller Scenario ---")
    for seller in range(num_sellers):
        final_prices=[price_results[idx][rep][seller][119] for rep in
range(REPLICATIONS)]
        final_profits=[profit_results[idx][rep][seller][119] for rep in
range(REPLICATIONS)]
        avg_price=np.mean(final_prices)
        avg_profit=np.mean(final_profits)
        print(f"Seller {seller+1}: Average Price = {avg_price:.2f}, Average Profit =
{avg_profit:.2f}")
#### S0- Table ####
import numpy as np
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
#Parameters
TICKS=120
REPLICATIONS=20
```

```
CUSTOMERS=1000
COST=25
LEARNING_START_TICK=20
USE_INITIAL_ONLY=False
AGE_MEAN=25
AGE_STD=5
INCOME_MEAN=30
INCOME_STD=8
BETA_PRICE=-0.1
BETA_AGE=-0.01
BETA_INCOME=0.05
def generate_customers():
    ages=np.random.normal(AGE_MEAN,AGE_STD,CUSTOMERS)
    incomes=np.random.normal(INCOME_MEAN,INCOME_STD,CUSTOMERS)
    return ages,incomes
def softmax(utilities):
    exp_util=np.exp(utilities)
    return exp_util/exp_util.sum(axis=1,keepdims=True)
def simulate_tick(prices,ages,incomes):
    REF_PRICE=50
    REF_AGE=25
    REF_INCOME=40
    n_sellers=len(prices)
    all_utils=np.zeros((CUSTOMERS,n_sellers+1))  # +1 for no-buy
    for i in range(n_sellers):
        reduced_price=prices[i]-REF_PRICE
        reduced_age=ages-REF_AGE
        reduced_income=incomes-REF_INCOME

all_utils[:,i]=(BETA_PRICE*reduced_price+BETA_AGE*reduced_age+BETA_IN
COME*reduced_income)
    all_utils[:,-1]=0  # no-buy utility
    probs=softmax(all_utils)
    choices=[np.random.choice(n_sellers+1,p=p) for p in probs]
    sales=[choices.count(i) for i in range(n_sellers)]
    no_buy=choices.count(n_sellers)
    return sales,no_buy
def seller_strategy(history_prices,history_profits):
    recent_price=history_prices[-1]
    poly=PolynomialFeatures(degree=2)
    X_poly=poly.fit_transform(np.array(history_prices).reshape(-1,1))
    model=LinearRegression().fit(X_poly,history_profits)

candidates=[recent_price*0.8,recent_price*0.9,recent_price,recent_price*1.1,recent_
price*1.2]
    preds=[model.predict(poly.transform([[p]]))[0] for p in candidates]
    return candidates[np.argmax(preds)]
def run_single_simulation(n_sellers):
    price_history=[[] for _ in range(n_sellers)]
    profit_history=[[] for _ in range(n_sellers)]
```

```
    sales_history=[[] for _ in range(n_sellers)]
    no_buy_history=[]
    ages,incomes=generate_customers()
    prices=[np.random.uniform(30,70) for _ in range(n_sellers)]
    next_prices=prices.copy()  # time alignment: prices to be used in this tick are here
    for tick in range(TICKS):
        prices=next_prices.copy()
        sales,no_buy=simulate_tick(prices,ages,incomes)
        no_buy_history.append(no_buy)
        for i in range(n_sellers):
            profit=(prices[i]-COST)*sales[i]
            price_history[i].append(prices[i])
            profit_history[i].append(profit)
            sales_history[i].append(sales[i])
        #Learning
        if tick>=LEARNING_START_TICK:
            new_prices=[]
            for i in range(n_sellers):
                if USE_INITIAL_ONLY:
                    hist_prices=price_history[i][:LEARNING_START_TICK]
                    hist_profits=profit_history[i][:LEARNING_START_TICK]
                else:
                    hist_prices=price_history[i][-LEARNING_START_TICK:]
                    hist_profits=profit_history[i][-LEARNING_START_TICK:]
                new_price=seller_strategy(hist_prices,hist_profits)
                new_prices.append(new_price)
            next_prices=new_prices
        else:
            next_prices=[np.random.uniform(30,70) for _ in range(n_sellers)]
    return price_history,profit_history,sales_history,no_buy_history
def run_simulation(n_sellers):
    price_histories_all=[]
    profit_histories_all=[]
    sales_histories_all=[]
    no_buy_histories_all=[]
    for _ in range(REPLICATIONS):
        price_hist,profit_hist,sales_hist,no_buy_hist=run_single_simulation(n_sellers)
        price_histories_all.append(price_hist)
        profit_histories_all.append(profit_hist)
        sales_histories_all.append(sales_hist)
        no_buy_histories_all.append(no_buy_hist)
    return
price_histories_all,profit_histories_all,sales_histories_all,no_buy_histories_all
summary_data=[]
columns=["# sellers","metric","seller 1","seller 2","seller 3","seller 4","seller 5"]
for num_sellers in range(1,6):

price_histories,profit_histories,sales_histories,no_buy_histories=run_simulation(num
_sellers)
    price_row=[num_sellers,"price"]
```

```
    profit_row=["","profit"]
    for i in range(num_sellers):
        prices_last_tick=[rep[i][-1] for rep in price_histories]
        profits_last_tick=[rep[i][-1] for rep in profit_histories]
        price_avg=np.mean(prices_last_tick)
        price_std=np.std(prices_last_tick)
        profit_avg=np.mean(profits_last_tick)
        profit_std=np.std(profits_last_tick)
        price_row.append(f"{price_avg:.2f} ± {price_std:.2f}")
        profit_row.append(f"{profit_avg:.2f} ± {profit_std:.2f}")
    while len(price_row)<7:
        price_row.append("")
        profit_row.append("")
    summary_data.append(price_row)
    summary_data.append(profit_row)
summary_df=pd.DataFrame(summary_data,columns=columns)
print("\n=== SummaryTable (Only Price and Profit) ===")
print(summary_df.to_string(index=False))
#Sales and No-Buy Statistics
print("\n=== Average Sales and No-Buy per Seller ===")
for num_sellers in range(1,6):
    _,_,sales_histories,no_buy_histories=run_simulation(num_sellers)
    print(f"\n--- {num_sellers} Seller Scenario ---")
    for i in range(num_sellers):
        seller_sales=[rep[i][-1] for rep in sales_histories]
        mean_sales=np.mean(seller_sales)
        std_sales=np.std(seller_sales)
        print(f"Seller {i+1}: Average Sales = {mean_sales:.1f} ({std_sales:.1f})")
    no_buy_last_tick=[rep[-1] for rep in no_buy_histories]
    no_buy_mean=np.mean(no_buy_last_tick)
    no_buy_std=np.std(no_buy_last_tick)
    no_buy_rate_mean=(no_buy_mean/CUSTOMERS)*100
    no_buy_rate_std=(no_buy_std/CUSTOMERS)*100
    print(f"No-Buy (number): {no_buy_mean:.1f} ({no_buy_std:.1f})")
    print(f"No-Buy ratio: %{no_buy_rate_mean:.2f} (%{no_buy_rate_std:.2f})")


#### ANOVA (S0) — differences between sellers within n=3 and n=5 ####
import pandas as pd
import numpy as np
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.multicomp import pairwise_tukeyhsd
# 1) For n=3 and n=5, collect observations at the last tick (each seller per replication
is one observation)
records=[]  # {'n','replication','seller_id','price','profit'}
for n in [3,5]:
    price_histories,profit_histories,_,_=run_simulation(n)
    # price_histories: len=REPLICATIONS; each element: list(len=n) with tick series
inside
```

```
    for rep_idx in range(REPLICATIONS):
        for s_idx in range(n):
            price_last=float(price_histories[rep_idx][s_idx][-1])
            profit_last=float(profit_histories[rep_idx][s_idx][-1])
records.append({'n':n,'replication':rep_idx+1,'seller_id':s_idx+1,'price':price_last,'prof
it':profit_last})
df_last=pd.DataFrame(records)
# 2) Types & cleaning
for col in ['n','replication','seller_id']:
    df_last[col]=pd.to_numeric(df_last[col],errors='coerce',downcast='integer')
for col in ['price','profit']:
    df_last[col]=pd.to_numeric(df_last[col],errors='coerce')
df_last=df_last.dropna(subset=['n','seller_id','price','profit']).copy()
df_last['n']=df_last['n'].astype('category')
df_last['seller_id']=df_last['seller_id'].astype('category')
# 3) Helpers
def anova_within_n(data:pd.DataFrame,y:str,n_value:int):
    """One-way ANOVA: y ~ C(seller_id) (differences between sellers within a fixed
n)"""
    sub=data.loc[data['n']==n_value].copy()
    model=ols(f'{y} ~ C(seller_id)',data=sub).fit()
    anova_tbl=sm.stats.anova_lm(model,typ=2)  # single factor: typ=2 == typ=3
    ss_eff=float(anova_tbl.loc['C(seller_id)','sum_sq'])
    ss_res=float(anova_tbl.loc['Residual','sum_sq'])
    eta_sq=ss_eff/(ss_eff+ss_res)  # effect size
    pval=float(anova_tbl.loc['C(seller_id)','PR(>F)'])
    Fval=float(anova_tbl.loc['C(seller_id)','F'])
grp=(sub.groupby('seller_id',observed=True)[y].agg(n='count',mean='mean',sd='std'))
    return anova_tbl,eta_sq,pval,Fval,grp,sub
def maybe_tukey(sub:pd.DataFrame,y:str,alpha=0.05):
    """Apply Tukey HSD if number of sellers > 2."""
    if sub['seller_id'].nunique()<=2:
        return None
    return
pairwise_tukeyhsd(endog=sub[y].values,groups=sub['seller_id'].values,alpha=alpha)
# 4) H₀ / H₁ (your requested format)
print("\nHypotheses (Price):")
print("n=3  H0: The mean prices of Sellers 1, 2, 3 are equal.")
print("     H1: At least one seller's mean price differs.")
print("n=5  H0: The mean prices of Sellers 1…5 are equal.")
print("     H1: At least one seller's mean price differs.")
print("\nHypotheses (Profit):")
print("n=3  H0: The mean profits of Sellers 1, 2, 3 are equal.")
print("     H1: At least one seller's mean profit differs.")
print("n=5  H0: The mean profits of Sellers 1…5 are equal.")
print("     H1: At least one seller's mean profit differs.")
# 5) Report: for n=3 and n=5, price & profit
for n_value in [3,5]:
    for y in ['price','profit']:
```

```
        print(f"\n================  ANOVA (S0): {y.upper()} ~ C(seller_id) |
n={n_value}  ================")
        anova_tbl,eta_sq,pval,Fval,grp,sub=anova_within_n(df_last,y,n_value)
        print(anova_tbl)  # PR(>F) column: p-value
        print(f"\nF = {Fval:.4f} | p-value = {pval:.6g} | eta-squared = {eta_sq:.4f}")
        print(f"\n--- Group summary [{y}] (n={n_value}) ---")
        print(grp)
        # Optional: Tukey HSD (to see which seller pairs differ)
        try:
            tuk=maybe_tukey(sub,y,alpha=0.05)
            if tuk is not None:
                print(f"\n--- Tukey HSD [{y}] (n={n_value}, alpha=0.05) ---")
                print(tuk.summary())
        except Exception as e:
            print(f"[WARN] Tukey HSD could not be run [{y}, n={n_value}]: {e}")

#### S0- Sensitivity Analysis ####
import numpy as np
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
# --- Parameters ---
TICKS=120
REPLICATIONS=20
CUSTOMERS=1000
LEARNING_START_TICK=20
USE_INITIAL_ONLY=False
AGE_MEAN=25
AGE_STD=5
INCOME_MEAN=30
INCOME_STD=8
BETA_PRICE=-0.1
BETA_AGE=-0.01
BETA_INCOME=0.05
# Only run for n=3 and n=5
SELLER_COUNTS=[3,5]
def seller_costs(n,low=15,high=35):
    """Return an evenly spaced cost vector in [low, high] for n sellers."""
    return np.linspace(low,high,n)
def generate_customers():
    ages=np.random.normal(AGE_MEAN,AGE_STD,CUSTOMERS)
    incomes=np.random.normal(INCOME_MEAN,INCOME_STD,CUSTOMERS)
    return ages,incomes
def softmax(utilities):
    exp_util=np.exp(utilities)
    return exp_util/exp_util.sum(axis=1,keepdims=True)
def simulate_tick(prices,ages,incomes):
    REF_PRICE=50
    REF_AGE=25
    REF_INCOME=40
```

```
    n_sellers=len(prices)
    all_utils=np.zeros((CUSTOMERS,n_sellers+1))  # +1 for no-buy
    reduced_age=ages-REF_AGE
    reduced_income=incomes-REF_INCOME
    for i in range(n_sellers):
        reduced_price=prices[i]-REF_PRICE

all_utils[:,i]=(BETA_PRICE*reduced_price+BETA_AGE*reduced_age+BETA_IN
COME*reduced_income)
    all_utils[:,-1]=0  # no-buy utility
    probs=softmax(all_utils)
    choices=[np.random.choice(n_sellers+1,p=p) for p in probs]
    sales=[choices.count(i) for i in range(n_sellers)]
    no_buy=choices.count(n_sellers)
    return sales,no_buy
def seller_strategy(history_prices,history_profits):
    recent_price=history_prices[-1]
    poly=PolynomialFeatures(degree=2)
    X_poly=poly.fit_transform(np.array(history_prices).reshape(-1,1))
    model=LinearRegression().fit(X_poly,history_profits)
    # Candidate set (0.8, 0.9, 1.0, 1.1, 1.2)

candidates=[recent_price*0.8,recent_price*0.9,recent_price,recent_price*1.1,recent_
price*1.2]
    preds=[model.predict(poly.transform([[p]]))[0] for p in candidates]
    return candidates[int(np.argmax(preds))]
def run_single_simulation(n_sellers):
    price_history=[[] for _ in range(n_sellers)]
    profit_history=[[] for _ in range(n_sellers)]
    sales_history=[[] for _ in range(n_sellers)]
    no_buy_history=[]
    costs=seller_costs(n_sellers,low=15,high=35)  # seller-specific costs
    ages,incomes=generate_customers()
    # Initial price set; but in the first 20 ticks, regenerate at each tick
    next_prices=[np.random.uniform(30,70) for _ in range(n_sellers)]
    for tick in range(TICKS):
        prices=next_prices.copy()
        sales,no_buy=simulate_tick(prices,ages,incomes)
        no_buy_history.append(no_buy)
        for i in range(n_sellers):
            profit=(prices[i]-costs[i])*sales[i]  # use costs[i] instead of COST
            price_history[i].append(prices[i])
            profit_history[i].append(profit)
            sales_history[i].append(sales[i])
        # Learning
        if tick>=LEARNING_START_TICK:
            new_prices=[]
            for i in range(n_sellers):
                if USE_INITIAL_ONLY:
                    hist_prices=price_history[i][:LEARNING_START_TICK]
```

```
                hist_profits=profit_history[i][:LEARNING_START_TICK]
            else:
                hist_prices=price_history[i][-LEARNING_START_TICK:]
                hist_profits=profit_history[i][-LEARNING_START_TICK:]
            # Check for sufficient data (if too short, keep current price)
            if len(hist_prices)>=3 and len(hist_profits)>=3:
                new_price=seller_strategy(hist_prices,hist_profits)
            else:
                new_price=prices[i]
            new_prices.append(new_price)
        next_prices=new_prices
    else:
        # First 20 ticks: regenerate uniformly at each tick
        next_prices=[np.random.uniform(30,70) for _ in range(n_sellers)]
  return price_history,profit_history,sales_history,no_buy_history,costs
def run_simulation(n_sellers):
  price_histories_all=[]
  profit_histories_all=[]
  sales_histories_all=[]
  no_buy_histories_all=[]
  costs_vec=None
  for _ in range(REPLICATIONS):

price_hist,profit_hist,sales_hist,no_buy_hist,costs=run_single_simulation(n_sellers)
    if costs_vec is None:
        costs_vec=costs
    price_histories_all.append(price_hist)
    profit_histories_all.append(profit_hist)
    sales_histories_all.append(sales_hist)
    no_buy_histories_all.append(no_buy_hist)
  return
price_histories_all,profit_histories_all,sales_histories_all,no_buy_histories_all,costs_
vec
# ==================== RUN AND SUMMARIZE
====================
summary_blocks=[]
for num_sellers in SELLER_COUNTS:

price_histories,profit_histories,sales_histories,no_buy_histories,costs_vec=run_simul
ation(num_sellers)
  # Summary table (tick 120)
  columns=["# sellers","metric"]+[f"seller {i+1} (c={int(costs_vec[i])})" for i in
range(num_sellers)]
  # Pad to 5 columns if desired (to keep a fixed number of columns)
  while len(columns)<7:
    columns.append("")
  summary_data=[]
  # PRICE
  price_row=[num_sellers,"price"]
  for i in range(num_sellers):
```

```
            prices_last_tick=[rep[i][-1] for rep in price_histories]
            price_avg=np.mean(prices_last_tick)
            price_std=np.std(prices_last_tick)
            price_row.append(f"{price_avg:.2f} ± {price_std:.2f}")
        while len(price_row)<7:
            price_row.append("")
        summary_data.append(price_row)
        # PROFIT
        profit_row=["","profit"]
        for i in range(num_sellers):
            profits_last_tick=[rep[i][-1] for rep in profit_histories]
            profit_avg=np.mean(profits_last_tick)
            profit_std=np.std(profits_last_tick)
            profit_row.append(f"{profit_avg:.2f} ± {profit_std:.2f}")
        while len(profit_row)<7:
            profit_row.append("")
        summary_data.append(profit_row)
        # SALES (optional but useful)
        sales_row=["","sales"]
        for i in range(num_sellers):
            seller_sales_last=[rep[i][-1] for rep in sales_histories]
            mean_sales=np.mean(seller_sales_last)
            std_sales=np.std(seller_sales_last)
            sales_row.append(f"{mean_sales:.1f} ({std_sales:.1f})")
        while len(sales_row)<7:
            sales_row.append("")
        summary_data.append(sales_row)
        # NO-BUY (total)
        no_buy_last_tick=[rep[-1] for rep in no_buy_histories]
        no_buy_mean=np.mean(no_buy_last_tick)
        no_buy_std=np.std(no_buy_last_tick)
        no_buy_rate_mean=(no_buy_mean/CUSTOMERS)*100
        no_buy_rate_std=(no_buy_std/CUSTOMERS)*100
        nobuy_row=["","no-buy",f"{no_buy_mean:.1f}
({no_buy_std:.1f})",f"%{no_buy_rate_mean:.2f} (%{no_buy_rate_std:.2f})"]
        while len(nobuy_row)<7:
            nobuy_row.append("")
        summary_data.append(nobuy_row)
        summary_df=pd.DataFrame(summary_data,columns=columns)
        summary_blocks.append((num_sellers,costs_vec,summary_df))
    # --- Print ---
    for n,costs_vec,df in summary_blocks:
        print(f"\n=== Summary Table — n={n} | costs={list(map(int,costs_vec))} ===")
        print(df.to_string(index=False))

    #### S1- Scenario ####
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    from sklearn.linear_model import LinearRegression
```

```
pd.set_option('display.max_columns',None)
pd.set_option('display.width',1000)
#Parameters
TICKS=120
REPLICATIONS=20
CUSTOMERS=1000
SELLER_COUNTS=[1,3,5,10,20,50]
COST=25
PROMO_COSTS={'none':0,'low':2.5,'high':5}
PROMO_EFFECTS={'none':0.0,'low':0.2,'high':0.5}
PROMO_CATEGORIES=['none','low','high']
LEARNING_START_TICK=20
USE_INITIAL_ONLY=False
#Output Settings
SELLER_ID=4  # 5th seller (0-index)
TICK_TO_PLOT=23  # tick to plot
TARGET_N_FOR_PLOT=5  # plot only for n=5
PRINT_REGRESSION=False  # True -> print intercept/coefficients
PLOT_REGRESSION=False  # True -> plot regression
#Customer distribution
AGE_MEAN=25
AGE_STD=5
INCOME_MEAN=30
INCOME_STD=8
AGE_BASE=25
INCOME_BASE=40
PRICE_BASE=50
#Utility coefficients
BETA_PRICE=-0.1
BETA_INCOME=0.05
BETA_AGE=-0.01
BETA_PROMO={'low':0.2,'high':0.5,'none':0.0}
def generate_customers():
    ages=np.random.normal(AGE_MEAN,AGE_STD,CUSTOMERS)
    incomes=np.random.normal(INCOME_MEAN,INCOME_STD,CUSTOMERS)
    return ages,incomes
def softmax(utilities):
    exp_util=np.exp(utilities)
    return exp_util/exp_util.sum(axis=1,keepdims=True)
def
plot_quadratic_for_tick(prices_used,promos_used,profits_used,tick,seller_idx,model
):
    lo=float(np.percentile(prices_used,5)) if len(prices_used)>1 else 30.0
    hi=float(np.percentile(prices_used,95)) if len(prices_used)>1 else 70.0
    if lo>=hi:
        lo,hi=30.0,70.0
    price_range=np.linspace(lo,hi,200)
    plt.figure(figsize=(8,6))
    for promo in ['none','low','high']:
        mask=(np.array(promos_used)==promo)
```

```
plt.scatter(np.array(prices_used)[mask],np.array(profits_used)[mask],label=f"Observ
ation ({promo})")
    for promo in ['none','low','high']:
        low=1 if promo=='low' else 0
        high=1 if promo=='high' else 0

Xr=np.column_stack([price_range,np.full_like(price_range,low),np.full_like(price_r
ange,high),price_range**2,price_range*low,price_range*high])
        preds=model.predict(Xr)
        plt.plot(price_range,preds,linestyle='--',label=f"Prediction ({promo})")
    plt.xlabel("Price")
    plt.ylabel("Profit")
    plt.title(f"Tick {tick} – Seller {seller_idx+1} quadratic regression")
    plt.legend()
    plt.grid(True)
    plt.show()
def simulate_one_run(n_sellers,enable_plot_this_run=False):
    prices=list(np.random.uniform(30,70,n_sellers))
    promos=list(np.random.choice(PROMO_CATEGORIES,size=n_sellers))
    history={i:{'price':[],'promo':[],'sales':[],'profit':[]} for i in range(n_sellers)}
    ages,incomes=generate_customers()
    for tick in range(TICKS):
        utilities=np.zeros((CUSTOMERS,n_sellers+1))
        reduced_age=ages-AGE_BASE
        reduced_income=incomes-INCOME_BASE
        for j in range(n_sellers):
            beta_promo=BETA_PROMO[promos[j]]
            reduced_price=prices[j]-PRICE_BASE

utilities[:,j]=(BETA_PRICE*reduced_price+BETA_AGE*reduced_age+BETA_INC
OME*reduced_income+beta_promo)
        utilities[:,-1]=0
        probs=softmax(utilities)
        choices=[np.random.choice(n_sellers+1,p=p) for p in probs]
        sales=[choices.count(j) for j in range(n_sellers)]
        profits=[(prices[j]-COST)*sales[j]-PROMO_COSTS[promos[j]]*sales[j] for j in
range(n_sellers)]
        for j in range(n_sellers):
            history[j]['price'].append(prices[j])
            history[j]['promo'].append(promos[j])
            history[j]['sales'].append(sales[j])
            history[j]['profit'].append(profits[j])
        if tick<LEARNING_START_TICK:
            prices=list(np.random.uniform(30,70,n_sellers))
            promos=list(np.random.choice(PROMO_CATEGORIES,size=n_sellers))
        elif tick>=LEARNING_START_TICK:
            for j in range(n_sellers):
                hist=history[j]
                if USE_INITIAL_ONLY:
```

```
                prices_used=hist['price'][:LEARNING_START_TICK]
                promos_used=hist['promo'][:LEARNING_START_TICK]
                profits_used=hist['profit'][:LEARNING_START_TICK]
            else:
                prices_used=hist['price'][-LEARNING_START_TICK:]
                promos_used=hist['promo'][-LEARNING_START_TICK:]
                profits_used=hist['profit'][-LEARNING_START_TICK:]
            X,y=[],[]
            for p,pr,prof in zip(prices_used,promos_used,profits_used):
                low=1 if pr=='low' else 0
                high=1 if pr=='high' else 0
                X.append([p,low,high,p**2,p*low,p*high])
                y.append(prof)
            model=LinearRegression().fit(X,y)
            #Plot/print condition: only selected tick & seller
            if enable_plot_this_run and (tick==TICK_TO_PLOT) and
(j==SELLER_ID):
                if PRINT_REGRESSION:
                    print(f"\n[REGRESSION] Tick {tick}, Seller {j+1}")
                    print("Intercept:",model.intercept_)
                    print("Coefficients:",model.coef_)
                if PLOT_REGRESSION:

plot_quadratic_for_tick(prices_used,promos_used,profits_used,tick,j,model)
            #New price/promo selection via simple search space
            candidates=[]
            for f_multiplier in [0.8,0.9,1.0,1.1,1.2]:
                for promo in PROMO_CATEGORIES:
                    p_new=prices[j]*f_multiplier
                    low=1 if promo=='low' else 0
                    high=1 if promo=='high' else 0
                    x_new=[[p_new,low,high,p_new**2,p_new*low,p_new*high]]
                    profit_pred=model.predict(x_new)[0]
                    candidates.append((profit_pred,p_new,promo))
            _,new_price,new_promo=max(candidates,key=lambda t:t[0])
            prices[j]=new_price
            promos[j]=new_promo
    final_prices=[history[j]['price'][-1] for j in range(n_sellers)]
    final_profits=[history[j]['profit'][-1] for j in range(n_sellers)]
    final_promos=[history[j]['promo'][-1] for j in range(n_sellers)]
    final_sales=[history[j]['sales'][-1] for j in range(n_sellers)]
    final_no_sales=CUSTOMERS-sum(final_sales)
    return final_prices,final_profits,final_promos,final_sales,final_no_sales
#report
results={}
plotted=False
for n in SELLER_COUNTS:
    all_prices,all_profits,all_promos=[],[],[]
    all_sales,all_no_sales=[],[]
    for rep in range(REPLICATIONS):
```

```
        enable_plot=(not plotted) and (n==TARGET_N_FOR_PLOT) and (rep==0)
and (SELLER_ID<n)

p,pr,promo,sales,no_sales=simulate_one_run(n,enable_plot_this_run=enable_plot)
        if enable_plot:
            plotted=True
        all_prices.append(p)
        all_profits.append(pr)
        all_promos.append(promo)
        all_sales.append(sales)
        all_no_sales.append(no_sales)
    price_arr=np.array(all_prices)  # shape: (REPS, n)
    profit_arr=np.array(all_profits)  # shape: (REPS, n)
    sales_arr=np.array(all_sales)  # shape: (REPS, n)
    no_sales_arr=np.array(all_no_sales)  # shape: (REPS,)
    #Seller-based averages and profits (between replications)
    price_means_per_seller=np.mean(price_arr,axis=0)
    price_stds_per_seller=np.std(price_arr,axis=0)
    profit_means_per_seller=np.mean(profit_arr,axis=0)
    profit_stds_per_seller=np.std(profit_arr,axis=0)

results[n]={'price_mean_per_seller':price_means_per_seller,'profit_mean_per_seller'
:profit_means_per_seller,'promos':all_promos,'sales_arr':sales_arr,'no_sales_mean':n
p.mean(no_sales_arr),'no_sales_std':np.std(no_sales_arr)}
#Table
table=[]
for n in SELLER_COUNTS:
    res=results[n]
    price_row=[n,'price']
    profit_row=[n,'profit']
    promo_row=[n,'promo']
    if n<=5:
        for i in range(n):
            price_mean=res['price_mean_per_seller'][i]
            price_std_r=res['price_std_per_seller'][i]  # rep std
            profit_mean=res['profit_mean_per_seller'][i]
            profit_std_r=res['profit_std_per_seller'][i]
            price_row.append(f"{price_mean:.2f} ± {price_std_r:.2f} [rep std]")
            profit_row.append(f"{profit_mean:.2f} ± {profit_std_r:.2f} [rep std]")
            promos_runs=[run[i] for run in res['promos']]
            counts=pd.Series(promos_runs).value_counts()
            promo_str=", ".join([f"{v} times {k}" for k,v in counts.items()])
            promo_row.append(promo_str)
        avg_price_mean=np.mean(res['price_mean_per_seller'])
        avg_profit_mean=np.mean(res['profit_mean_per_seller'])
        price_row+=[""]*(6-len(price_row))+[f"{avg_price_mean:.2f} ±
{res['price_std_of_means']:.2f} (std of seller means) ({res['price_mean_of_stds']:.2f}
mean of seller stds)"]
        profit_row+=[""]*(6-len(profit_row))+[f"{avg_profit_mean:.2f} ±
{res['profit_std_of_means']:.2f} ({res['profit_mean_of_stds']:.2f})"]
```

```
        promo_row+=['']*(6-len(promo_row))+['']
    else:
        avg_price_mean=np.mean(res['price_mean_per_seller'])
        avg_profit_mean=np.mean(res['profit_mean_per_seller'])
        flat_promos=[p for run in res['promos'] for p in run]
        counts=pd.Series(flat_promos).value_counts()
        promo_str=", ".join([f"{v} times {k}" for k,v in counts.items()])
        price_row+=['']*5+[f"{avg_price_mean:.2f} ± {res['price_std_of_means']:.2f}
({res['price_mean_of_stds']:.2f} "]
        profit_row+=['']*5+[f"{avg_profit_mean:.2f} ± {res['profit_std_of_means']:.2f}
({res['profit_mean_of_stds']:.2f} "]
        promo_row+=['']*5+[promo_str]
    table.append(price_row)
    table.append(profit_row)
    table.append(promo_row)
df_table=pd.DataFrame(table,columns=['total number of
sellers','','seller1','seller2','seller3','seller4','seller5','all sellers average'])
pd.set_option('display.max_colwidth',None)  # disable cell truncation
print("\n--- Main Results Table ---\n")
print(df_table)
#Additional Output
print("\n--- Additional Output ---")
#for n = 1, 3, 5 average sales and its std of last time step
for n in [1,3,5]:
    if n in results:
        res=results[n]
        sales_arr=res['sales_arr']  # (REPS, n)
        means_per_seller=np.mean(sales_arr,axis=0)
        stds_per_seller=np.std(sales_arr,axis=0)  # rep std
        print(f"\n[n = {n}] seller based average sales and its std of last time step [rep
std]:")
        for i in range(n):
            print(f"  Seller {i+1}: {means_per_seller[i]:.2f} ± {stds_per_seller[i]:.2f}
[rep std]")
        print(f"  → Group stds: {std_of_means:.2f} (std)")
#No purchase
print("\nNo purchase (for each n; replication averages ± std):")
for n in SELLER_COUNTS:
    res=results[n]
    print(f"  n = {n}: {res['no_sales_mean']:.2f} ± {res['no_sales_std']:.2f}")

#### ANOVA (S1) -  differences between sellers within n=3 and n=5 ####
import pandas as pd
import numpy as np
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.multicomp import pairwise_tukeyhsd
# 1) For n=3 and n=5, collect last-tick observations (each seller per replication is one
observation)
records=[]  # {'n','replication','seller_id','price','profit'}
```

```
for n in [3,5]:
    for rep_idx in range(REPLICATIONS):

final_prices,final_profits,final_promos,final_sales,final_no_sales=simulate_one_run(
n)
        for s_idx in range(n):

records.append({'n':n,'replication':rep_idx+1,'seller_id':s_idx+1,'price':float(final_pri
ces[s_idx]),'profit':float(final_profits[s_idx]),'promo':final_promos[s_idx]})  # keep
promo for potential future analysis
df_last=pd.DataFrame(records)
# 2) Types & cleaning
for col in ['n','replication','seller_id']:
    df_last[col]=pd.to_numeric(df_last[col],errors='coerce',downcast='integer')
for col in ['price','profit']:
    df_last[col]=pd.to_numeric(df_last[col],errors='coerce')
df_last=df_last.dropna(subset=['n','seller_id','price','profit']).copy()
df_last['n']=df_last['n'].astype('category')
df_last['seller_id']=df_last['seller_id'].astype('category')
# 3) Helpers
def anova_within_n(data:pd.DataFrame,y:str,n_value:int):
    """One-way ANOVA: y ~ C(seller_id) (differences between sellers within a fixed
n)"""
    sub=data.loc[data['n']==n_value].copy()
    model=ols(f'{y} ~ C(seller_id)',data=sub).fit()
    anova_tbl=sm.stats.anova_lm(model,typ=2)  # single factor: typ=2 == typ=3
    ss_eff=float(anova_tbl.loc['C(seller_id)','sum_sq'])
    ss_res=float(anova_tbl.loc['Residual','sum_sq'])
    eta_sq=ss_eff/(ss_eff+ss_res)  # effect size (eta-squared)
    pval=float(anova_tbl.loc['C(seller_id)','PR(>F)'])
    Fval=float(anova_tbl.loc['C(seller_id)','F'])

grp=(sub.groupby('seller_id',observed=True)[y].agg(n='count',mean='mean',sd='std'))
    return anova_tbl,eta_sq,pval,Fval,grp,sub
def maybe_tukey(sub:pd.DataFrame,y:str,alpha=0.05):
    """Apply Tukey HSD if number of sellers > 2."""
    if sub['seller_id'].nunique()<=2:
        return None
    return
pairwise_tukeyhsd(endog=sub[y].values,groups=sub['seller_id'].values,alpha=alpha)
# 4) H0 / H1 (requested format)
print("\nHypotheses (Price):")
print("n=3  H0: The mean prices of Sellers 1, 2, 3 are equal.")
print("     H1: At least one seller's mean price differs.")
print("n=5  H0: The mean prices of Sellers 1…5 are equal.")
print("     H1: At least one seller's mean price differs.")
print("\nHypotheses (Profit):")
print("n=3  H0: The mean profits of Sellers 1, 2, 3 are equal.")
print("     H1: At least one seller's mean profit differs.")
print("n=5  H0: The mean profits of Sellers 1…5 are equal.")
```

```
print("     H1: At least one seller's mean profit differs.")
# 5) Report: for n=3 and n=5, price & profit
for n_value in [3,5]:
   for y in ['price','profit']:
      print(f"\n================  ANOVA (S1): {y.upper()} ~ C(seller_id) |
n={n_value}  ================")
      anova_tbl,eta_sq,pval,Fval,grp,sub=anova_within_n(df_last,y,n_value)
      print(anova_tbl)  # PR(>F) column: p-value
      print(f"\nF = {Fval:.4f} | p-value = {pval:.6g} | eta-squared = {eta_sq:.4f}")
      print(f"\n--- Group summary [{y}] (n={n_value}) ---")
      print(grp)
      # Optional: Tukey HSD (to see which seller pairs differ)
      try:
         tuk=maybe_tukey(sub,y,alpha=0.05)
         if tuk is not None:
            print(f"\n--- Tukey HSD [{y}] (n={n_value}, alpha=0.05) ---")
            print(tuk.summary())
      except Exception as e:
         print(f"[WARN] Tukey HSD could not be run [{y}, n={n_value}]: {e}")
#Sensitivity Analyis of S1- SA1
#Parameters are changed to run the code S1- Scenario
# Sensitivity Analyis of S1- SA2
#Parameters are changed to run the code S1- Scenario


#### S2 – Scenario #####
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
# --- Pandas display settings ---
pd.set_option('display.max_columns',None)
pd.set_option('display.width',1000)
# --- Parameters (S2) ---
TICKS=120
REPLICATIONS=20
CUSTOMERS=1000
SELLER_COUNTS=[1,3,5,11,21,51]
COST=25
LEARNING_START_TICK=20
USE_INITIAL_ONLY=False  # False: always regress on last 20 ticks
#Customer distribution
AGE_MEAN=25
AGE_STD=5
INCOME_MEAN=30
INCOME_STD=8
AGE_BASE=25
INCOME_BASE=40
PRICE_BASE=50
#Utility coefficients
BETA_PRICE=-0.1
```

```
BETA_INCOME=0.05
BETA_AGE=-0.01
# --- Quality params (S2) ---
QUALITY_MIN=-0.5
QUALITY_MAX=0.5
BETA_QUALITY=1  # Effect of quality on utility
COSTQ=10.0  # Per-unit cost coefficient for quality (linear)
#Helpers
def generate_customers():
   ages=np.random.normal(AGE_MEAN,AGE_STD,CUSTOMERS)
   incomes=np.random.normal(INCOME_MEAN,INCOME_STD,CUSTOMERS)
   return ages,incomes
def generate_qualities(n_sellers:int):
   # If single seller, quality 0.0
   if n_sellers==1:
      return np.array([0.0],dtype=float)
   # For multiple sellers, split [-0.5, 0.5] equally across n_sellers; ascending order
   return np.linspace(QUALITY_MIN,QUALITY_MAX,n_sellers,dtype=float)
def softmax_stable(utilities):
   u=utilities-utilities.max(axis=1,keepdims=True)
   exp_u=np.exp(u)
   return exp_u/exp_u.sum(axis=1,keepdims=True)
def simulate_one_run(n_sellers):
   prices=np.random.uniform(30,70,n_sellers).astype(float)
   qualities=generate_qualities(n_sellers)  # quality vector
   history={i:{'price':[],'sales':[],'profit':[]} for i in range(n_sellers)}
   ages,incomes=generate_customers()
   for tick in range(TICKS):
      # First 20 ticks: re-randomize prices at each tick
      if tick<LEARNING_START_TICK:
         prices=np.random.uniform(30,70,n_sellers).astype(float)
      # ----- Utility (including no-buy) -----
      utilities=np.zeros((CUSTOMERS,n_sellers+1))
      reduced_age=ages-AGE_BASE
      reduced_income=incomes-INCOME_BASE
      for j in range(n_sellers):
         reduced_price=prices[j]-PRICE_BASE

utilities[:,j]=(BETA_PRICE*reduced_price+BETA_AGE*reduced_age+BETA_INC
OME*reduced_income+BETA_QUALITY*qualities[j])  # constant quality
contribution
      # No-buy column: 0 utility (reference)
      utilities[:,-1]=0.0
      probs=softmax_stable(utilities)
      choices=[np.random.choice(n_sellers+1,p=p) for p in probs]
      counts=np.bincount(choices,minlength=n_sellers+1)
      sales=counts[:n_sellers].astype(int)
      no_sales=int(counts[-1])
      # Profit
      unit_costs=COST+COSTQ*qualities
```

```
        profits=(prices-unit_costs)*sales
        # Write to history
        for j in range(n_sellers):
            history[j]['price'].append(float(prices[j]))
            history[j]['sales'].append(int(sales[j]))
            history[j]['profit'].append(float(profits[j]))
        # Learning & price update
        if tick>=LEARNING_START_TICK:
            for j in range(n_sellers):
                hist=history[j]
                if USE_INITIAL_ONLY:
                    prices_used=hist['price'][:LEARNING_START_TICK]
                    profits_used=hist['profit'][:LEARNING_START_TICK]
                else:
                    prices_used=hist['price'][-LEARNING_START_TICK:]
                    profits_used=hist['profit'][-LEARNING_START_TICK:]
                # Features: p, p^2 (intercept automatic)
                X=[[p,p**2] for p in prices_used]
                y=profits_used
                model=LinearRegression().fit(X,y)
                # Candidate prices: multipliers 0.8..1.2
                candidates=[]
                for f_multiplier in [0.8,0.9,1.0,1.1,1.2]:
                    p=prices[j]*f_multiplier
                    x_new=[[p,p**2]]
                    profit_pred=model.predict(x_new)[0]
                    candidates.append((profit_pred,p))
                # Choose highest expected profit
                _,new_price=max(candidates,key=lambda t:t[0])
                prices[j]=float(new_price)
                # If no sales in last 20 ticks, reduce price
                recent_sales_sum=sum(hist['sales'][-LEARNING_START_TICK:])
                if recent_sales_sum==0:
                    prices[j]*=0.9
    # Final values (tick 120)
    final_prices=[history[j]['price'][-1] for j in range(n_sellers)]
    final_profits=[history[j]['profit'][-1] for j in range(n_sellers)]
    final_sales=[history[j]['sales'][-1] for j in range(n_sellers)]
    final_no_sales=CUSTOMERS-sum(final_sales)
    return final_prices,final_profits,final_sales,final_no_sales
# ---- Multiple runs ----
results={}
for n in SELLER_COUNTS:
    all_prices,all_profits=[],[]
    all_sales,all_no_sales=[],[]
    for _ in range(REPLICATIONS):
        p,pr,sales,no_sales=simulate_one_run(n)
        all_prices.append(p)
        all_profits.append(pr)
        all_sales.append(sales)
```

```
        all_no_sales.append(no_sales)
    price_arr=np.array(all_prices)  # (R, n)
    profit_arr=np.array(all_profits)  # (R, n)
    sales_arr=np.array(all_sales)  # (R, n)
    no_sales_arr=np.array(all_no_sales)  # (R,)
results[n]={'price_mean':np.mean(price_arr,axis=0),'price_std':np.std(price_arr,axis=
0),'profit_mean':np.mean(profit_arr,axis=0),'profit_std':np.std(profit_arr,axis=0),'sale
s_mean':np.mean(sales_arr,axis=0),'sales_std':np.std(sales_arr,axis=0),'no_sales_mea
n':float(np.mean(no_sales_arr)),'no_sales_std':float(np.std(no_sales_arr)),'price_stdm
':float(np.std(np.mean(price_arr,axis=1))),'profit_stdm':float(np.std(np.mean(profit_a
rr,axis=1))),'sales_stdm':float(np.std(np.mean(sales_arr,axis=1))),'qualities':generate_
qualities(n)}
# ---- Table: per-seller for small n; summary for large n ----
table=[]
for n in SELLER_COUNTS:
    res=results[n]
    price_row=[n,'price']
    profit_row=[n,'profit']
    quality_row=[n,'quality']  # quality row
    if n<=5:
        for i in range(n):
            price=f"{res['price_mean'][i]:.2f} ± {res['price_std'][i]:.2f}"
            profit=f"{res['profit_mean'][i]:.2f} ± {res['profit_std'][i]:.2f}"
            q_val=f"{res['qualities'][i]:.3f}"
            price_row.append(price)
            profit_row.append(profit)
            quality_row.append(q_val)
        while len(price_row)<8:
            price_row.append('');profit_row.append('');quality_row.append('')
    else:
        avg_price=f"{np.mean(res['price_mean']):.2f} ± {np.mean(res['price_std']):.2f}"
        avg_profit=f"{np.mean(res['profit_mean']):.2f} ±
{np.mean(res['profit_std']):.2f}"
        q_min,q_max=np.min(res['qualities']),np.max(res['qualities'])
        q_summary=f"[{q_min:.3f}, {q_max:.3f}]"
        price_row+=['']*5+[avg_price]
        profit_row+=['']*5+[avg_profit]
        quality_row+=['']*5+[q_summary]
    table.append(price_row)
    table.append(profit_row)
    table.append(quality_row)
df_table=pd.DataFrame(table,columns=['total number of
sellers','','seller1','seller2','seller3','seller4','seller5','all sellers average'])
print("\n--- Main Results Table (S2: quality, no promo) ---\n")
print(df_table)
#Also print quality values to console
print("\n--- Seller Quality Values ---\n")
for n in SELLER_COUNTS:
    qs=results[n]['qualities']
    if n<=5:
```

```
        for i,q in enumerate(qs,start=1):
            print(f"n={n} | Seller {i} quality = {q:.3f}")
    else:
        q_str=", ".join([f"{q:.3f}" for q in qs])
        print(f"n={n} | qualities = [{q_str}]")
    print()
print("\n--- Final Sales and No Purchase Summary at Tick 120 ---\n")
for n in SELLER_COUNTS:
    res=results[n]
    if n<=5:
        for i in range(n):
            print(f"Seller {i+1}: sales = {res['sales_mean'][i]:.2f} ±
{res['sales_std'][i]:.2f}")
    else:
        print(f"Avg seller sales: {res['sales_mean'].mean():.2f} ±
{res['sales_std'].mean():.2f}")
    print(f"No Purchase: {res['no_sales_mean']:.2f} ± {res['no_sales_std']:.2f}")
    if n in [11,21,51]:
        print(f"STDm - Price: {res['price_stdm']:.2f}, Profit: {res['profit_stdm']:.2f},
Sales: {res['sales_stdm']:.2f}")
    print()
#### ANOVA (S2) -  differences between sellers within n=3 and n=5 ####
import pandas as pd
import numpy as np
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.multicomp import pairwise_tukeyhsd
# 1) For n=3 and n=5, collect last-tick observations (each replication × each seller =
1 observation)
records=[]  # {'n','replication','seller_id','price','profit','quality'}
for n in [3,5]:
    for rep_idx in range(REPLICATIONS):
        final_prices,final_profits,final_sales,final_no_sales=simulate_one_run(n)
        qualities=generate_qualities(n)
        for s_idx in range(n):

records.append({'n':n,'replication':rep_idx+1,'seller_id':s_idx+1,'price':float(final_pri
ces[s_idx]),'profit':float(final_profits[s_idx]),'quality':float(qualities[s_idx])})
df_last=pd.DataFrame(records)
# 2) Types & cleaning
for col in ['n','replication','seller_id']:
    df_last[col]=pd.to_numeric(df_last[col],errors='coerce',downcast='integer')
for col in ['price','profit','quality']:
    df_last[col]=pd.to_numeric(df_last[col],errors='coerce')
df_last=df_last.dropna(subset=['n','seller_id','price','profit']).copy()
df_last['n']=df_last['n'].astype('category')
df_last['seller_id']=df_last['seller_id'].astype('category')
# 3) Helpers
def anova_within_n(data:pd.DataFrame,y:str,n_value:int):
```

```
    """One-way ANOVA: y ~ C(seller_id) (differences between sellers within a fixed
n)"""
    sub=data.loc[data['n']==n_value].copy()
    model=ols(f'{y} ~ C(seller_id)',data=sub).fit()
    anova_tbl=sm.stats.anova_lm(model,typ=2)  # single factor: typ=2 == typ=3
    ss_eff=float(anova_tbl.loc['C(seller_id)','sum_sq'])
    ss_res=float(anova_tbl.loc['Residual','sum_sq'])
    eta_sq=ss_eff/(ss_eff+ss_res)  # effect size (eta^2)
    pval=float(anova_tbl.loc['C(seller_id)','PR(>F)'])
    Fval=float(anova_tbl.loc['C(seller_id)','F'])

grp=(sub.groupby('seller_id',observed=True)[y].agg(n='count',mean='mean',sd='std'))
    return anova_tbl,eta_sq,pval,Fval,grp,sub
def maybe_tukey(sub:pd.DataFrame,y:str,alpha=0.05):
    """Apply Tukey HSD if number of sellers > 2."""
    if sub['seller_id'].nunique()<=2:
        return None
    return
pairwise_tukeyhsd(endog=sub[y].values,groups=sub['seller_id'].values,alpha=alpha)
# 4) Report: for n=3 and n=5, price & profit
for n_value in [3,5]:
    for y in ['price','profit']:
        print(f"\n==========  ANOVA (S2): {y.upper()} ~ C(seller_id) |
n={n_value}  ==========")
        anova_tbl,eta_sq,pval,Fval,grp,sub=anova_within_n(df_last,y,n_value)
        print(anova_tbl)  # PR(>F) column: p-value
        print(f"\nF = {Fval:.4f} | p-value = {pval:.6g} | eta-squared = {eta_sq:.4f}")
        print(f"\n--- Group summary [{y}] (n={n_value}) ---")
        print(grp)
        # Optional: Tukey HSD (to see which seller pairs differ)
        try:
            tuk=maybe_tukey(sub,y,alpha=0.05)
            if tuk is not None:
                print(f"\n--- Tukey HSD [{y}] (n={n_value}, alpha=0.05) ---")
                print(tuk.summary())
        except Exception as e:
            print(f"[WARN] Tukey HSD could not be run [{y}, n={n_value}]: {e}")
# (If needed, you can include quality as a covariate and run ANCOVA or a two-
factor ANOVA:
# e.g., y ~ C(seller_id) + quality or y ~ C(seller_id) * C(quality_bin); but for now we
keep a single-factor setup.)
#Sensitivity Analyis of S2
#Parameters are changed to run the code S2 – Scenario
#Regression Results#
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from scipy.stats import linregress

TICKS = 120
```

```
REPLICATIONS = 20
CUSTOMERS = 1000
COST = 25
LEARNING_START_TICK = 20
USE_INITIAL_ONLY = False
ANALYSIS_TICK = 120

AGE_MEAN, AGE_STD = 25, 5
INCOME_MEAN, INCOME_STD = 30, 8
AGE_BASE, INCOME_BASE, PRICE_BASE = 25, 40, 50

BETA_PRICE, BETA_INCOME, BETA_AGE = -0.1, 0.05, -0.01

QUALITY_MIN, QUALITY_MAX = -0.5, 0.5

def generate_customers():
    ages = np.random.normal(AGE_MEAN, AGE_STD, CUSTOMERS)
    incomes = np.random.normal(INCOME_MEAN, INCOME_STD, CUSTOMERS)
    return ages, incomes

def generate_qualities(n_sellers: int):
    if n_sellers == 1:
        return np.array([0.0], dtype=float)
    return np.linspace(QUALITY_MIN, QUALITY_MAX, n_sellers, dtype=float)

def softmax_stable(utilities):
    u = utilities - utilities.max(axis=1, keepdims=True)
    exp_u = np.exp(u)
    return exp_u / exp_u.sum(axis=1, keepdims=True)

def simulate_one_run(n_sellers, costq, beta_quality,
analysis_tick=ANALYSIS_TICK):
    prices = np.random.uniform(30, 70, n_sellers).astype(float)
    qualities = generate_qualities(n_sellers)
    history = {i: {'price': [], 'sales': [], 'profit': []} for i in range(n_sellers)}
    ages, incomes = generate_customers()

    for tick in range(TICKS):
        if tick < LEARNING_START_TICK:
            prices = np.random.uniform(30, 70, n_sellers).astype(float)
        utilities = np.zeros((CUSTOMERS, n_sellers + 1))
        reduced_age = ages - AGE_BASE
        reduced_income = incomes - INCOME_BASE
        for j in range(n_sellers):
            reduced_price = prices[j] - PRICE_BASE
            utilities[:, j] = (
                BETA_PRICE   * reduced_price +
                BETA_AGE     * reduced_age +
                BETA_INCOME  * reduced_income +
                beta_quality * qualities[j]
```

```
            )
        utilities[:, -1] = 0.0
        probs = softmax_stable(utilities)
        choices = [np.random.choice(n_sellers + 1, p=p) for p in probs]
        counts = np.bincount(choices, minlength=n_sellers + 1)
        sales = counts[:n_sellers].astype(int)
        unit_costs = COST + costq * qualities
        profits = (prices - unit_costs) * sales
        for j in range(n_sellers):
            history[j]['price'].append(float(prices[j]))
            history[j]['sales'].append(int(sales[j]))
            history[j]['profit'].append(float(profits[j]))
        if tick >= LEARNING_START_TICK:
            for j in range(n_sellers):
                hist = history[j]
                if USE_INITIAL_ONLY:
                    prices_used = hist['price'][:LEARNING_START_TICK]
                    profits_used = hist['profit'][:LEARNING_START_TICK]
                else:
                    prices_used = hist['price'][-LEARNING_START_TICK:]
                    profits_used = hist['profit'][-LEARNING_START_TICK:]
                X = [[p, p**2] for p in prices_used]
                y = profits_used
                model = LinearRegression().fit(X, y)
                best_val, best_price = -np.inf, prices[j]
                for mult in [0.8, 0.9, 1.0, 1.1, 1.2]:
                    p_cand = prices[j] * mult
                    pred = model.predict([[p_cand, p_cand**2]])[0]
                    if pred > best_val:
                        best_val, best_price = pred, p_cand
                prices[j] = float(best_price)
                if sum(hist['sales'][-LEARNING_START_TICK:]) == 0:
                    prices[j] *= 0.9

    idx = max(1, min(analysis_tick, TICKS)) - 1
    prices_at = [history[j]['price'][idx]  for j in range(n_sellers)]
    profits_at = [history[j]['profit'][idx] for j in range(n_sellers)]
    sales_at   = [history[j]['sales'][idx]  for j in range(n_sellers)]
    no_sales_at = CUSTOMERS - sum(sales_at)
    return prices_at, profits_at, sales_at, no_sales_at, qualities

def simulate_case_means(n_sellers, costq, beta_quality,
replications=REPLICATIONS, analysis_tick=ANALYSIS_TICK):
    all_prices, all_profits = [], []
    qs_ref = None
    for _ in range(replications):
        p, pr, _, _, q = simulate_one_run(n_sellers, costq, beta_quality,
analysis_tick=analysis_tick)
        all_prices.append(p)
        all_profits.append(pr)
```

```
        if qs_ref is None:
            qs_ref = np.array(q, dtype=float)
    price_mean  = np.mean(np.array(all_prices), axis=0)
    profit_mean = np.mean(np.array(all_profits), axis=0)
    return qs_ref, price_mean, profit_mean

def regression_report(x, y, label="Model"):
    x = np.asarray(x); y = np.asarray(y)
    n = len(x)
    res = linregress(x, y)
    slope, intercept, r, p, stderr = res.slope, res.intercept, res.rvalue, res.pvalue,
res.stderr
    df = n - 2
    t_stat = slope / stderr if (stderr is not None and stderr > 0) else np.nan
    F_stat = t_stat**2 if not np.isnan(t_stat) else np.nan
    r2 = r**2
    return {
        "label": label,
        "n": n,
        "df": df,
        "slope": slope,
        "intercept": intercept,
        "t": t_stat,
        "p": p,
        "F": F_stat,
        "R2": r2,
        "significant": "Yes" if p < 0.05 else "No"
    }

def run_all_cases(n_sellers=21, analysis_tick=ANALYSIS_TICK):
    CASES = [
        {"quality_cost": 10.0, "beta_quality": 1.0},
        {"quality_cost":  2.0, "beta_quality": 1.0},
        {"quality_cost": 10.0, "beta_quality": 0.3},
        {"quality_cost":  2.0, "beta_quality": 0.3},
    ]
    rows = []
    for case in CASES:
        qc, bq = case["quality_cost"], case["beta_quality"]
        q, price_mean, profit_mean = simulate_case_means(
            n_sellers=n_sellers, costq=qc, beta_quality=bq, analysis_tick=analysis_tick
        )
        rows.append(
            regression_report(q, price_mean, label=f"Price ~ Quality (qc={qc}, bq={bq},
tick={analysis_tick})")
        )
        rows.append(
            regression_report(q, profit_mean, label=f"Profit ~ Quality (qc={qc},
bq={bq}, tick={analysis_tick})")
        )
```

```
    df = pd.DataFrame(rows,
columns=["label","n","df","slope","intercept","t","p","F","R2","significant"])
    return df

if __name__ == "__main__":
    summary_df = run_all_cases(n_sellers=21, analysis_tick=ANALYSIS_TICK)
    print("\n=== Regression Significance Summary (n=21, replication means, tick="
          f"{ANALYSIS_TICK}) ===\n")
    print(summary_df.to_string(index=False, float_format=lambda x: f"{x:.4f}"))
```

APPENDIX B

RESULTS OF SCENARIO 1

Total Number of Sellers	Variable	Seller 1	Seller 2	Seller 3	Seller 4	Seller 5	Number of "Not Buy"	Average of All Sellers
1	sales	487.05 (68.86)					512.95 (68.86)	
	price	48.01 (4.25)						
	profit	9852.91 (383.68)						
	promo	10 times none, 9 times high, 1 times low						
3	sales	273.70 (87.80)	271.65 (81.71)	277.70 (97.37)			176.95 (37.72)	
	price	43.36 (4.43)	43.81 (3.79)	42.94 (4.01)				
	profit	3919.89 (655.81)	3997.28 (637.77)	3897.93 (645.20)				
	promo	8 times high, 6 times none, 6 times low	12 times high, 6 times none, 2 times low	9 times none, 8 times high, 3 times low				
5	sales	161.80 (48.57)	189.00 (83.03)	195.55 (64.85)	162.90 (46.98)	193.40 (77.76)	97.35 (30.61)	
	price	42.47 (5.43)	41.06 (5.05)	40.41 (4.05)	41.46 (3.52)	41.00 (6.74)		
	profit	2136.48 (468.06)	2150.41 (574.14)	2347.37 (358.92)	2267.88 (562.59)	2183.89 (451.41)		
	promo	10 times high, 7 times none, 3 times low	10 times high, 9 times none, 1 times low	7 times none, 7 times high, 6 times low	10 times high, 7 times none, 3 times low	9 times high, 7 times none, 4 times low		

10	sales price profit promo		45.65 (7.98)	95.44 (25.72) 40.72 (0.97) 1072.44 (72.02) 9.3 times high, 7 times none, 3.7 times low
20	sales price profit promo		21.45 (4.93)	48.93 (17.08) 45.56 (13.95) 489.69 (32.78) 8.6 times high, 6.7 times none, 4.7 times low
50	sales price profit promo		8.10 (2.66)	19.84 (8.63) 43.09 (9.06) 198.00 (14.02) 8.26 times high, 6.88 times none, 4.86 times low

REFERENCES

Abbasi Siar, S., Keramati, M. A., & Motadel, M. R. (2022). Emergence of consumer impulse buying behavior with agent-based modeling approach. *Journal of Applied Research on Industrial Engineering*, *9*(3), 203-230.

Ahmed, A., Greensmith, J., & Aickelin, U. (2013). Variance in System Dynamics and Agent Based Modelling Using the SIR Model of Infectious Disease. *arXiv preprint arXiv:1307.2001*.

Anderson, S. P., de Palma, A., & Thisse, J.-F. (1992). *Discrete choice theory of product differentiation*. The MIT Press. https://doi.org/10.7551/mitpress/2450.001.0001

Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, *99*(suppl. 3), 7280–7287. https://doi.org/10.1073/pnas.082080899

Bozanta, A., & Nasır, A. (2014). Usage of agent-based modeling and simulation in marketing. *Journal of Advanced Management Science Vol*, *2*(3).

Chen, A. W. (2014). Brand choice prediction in wine consumption. *International Journal of Business Marketing and Management, 1*(1), 1–17.

Danyae, N., Ramez, K., & Colmekcioglu, N. (2023). An agent-based modeling approach for understanding drivers of consumer decisions on foreign vs domestic products: case study of a local refrigerator market. *International Journal of Information Technology & Decision Making, 22*(3), 1107–1134.

Delre, S. A., Jager, W., Bijmolt, T. H., & Janssen, M. A. (2007). Targeting and timing promotional activities: An agent-based model for the takeoff of new products. *Journal of Business Research*, *60*(8), 826-835.

Gilbert, N. (2020). *Agent-based models* (2nd ed.). SAGE Publications. https://doi.org/10.4135/9781506355580

Grimm, V., Berger, U., DeAngelis, D. L., Polhill, J. G., Giske, J., & Railsback, S. F. (2020). The ODD protocol for describing agent-based and other simulation models: A second update to improve clarity, replication, and structural realism. *Journal of Artificial Societies and Social Simulation*, *23*(2), 7. https://doi.org/10.18564/jasss.42 59

Guadagni, P. M., & Little, J. D. (1983). A logit model of brand choice calibrated on scanner data. *Marketing Science*, *2*(3), 203-238.

Jager, W. (2007). The four P's in social simulation, a perspective on how marketing could benefit from the use of social simulation. *Journal of Business Research*, *60*(8), 868-875.

Katsamakas, E., & Sanchez-Cartas, J. M. (2024). Responsible users and platform competition: A computational model. *Heliyon*, *10*(2).

Karakaya, Ç., Badur, B., & Aytekin, C. (2011). Analyzing the effectiveness of marketing strategies in the presence of word of mouth: Agent-based modeling approach. *Journal of Marketing Research and Case Studies, 2011*, 1-17.

Lee, K., Lee, H., & Kim, C. O. (2014). Pricing and timing strategies for new product using agent-based simulation of behavioural consumers. *Journal of Artificial Societies and Social Simulation, 17*(2), 1. https://doi.org/10.18564/jasss.2326

Mishra, S., Silva, T. L., Hellemo, L., Jaehnert, S., Egner, L. E., Petersen, S. A., ... & Bordin, C. (2025). Agent-based modeling: Insights into consumer behavior, urban dynamics, grid management, and market interactions. *Energy Strategy Reviews*, *57*, 101613.

Onggo, B. S., & Foramitti, J. (2021, December). Agent-based modeling and simulation for business and management: a review and tutorial. In *2021 Winter Simulation Conference (WSC)* (pp. 1-15). IEEE.

Rand, W., & Rust, R. T. (2011). Agent-based modeling in marketing: Guidelines for rigor. *International Journal of research in Marketing*, *28*(3), 181-193.

Romero, E., Chica, M., Damas, S., & Rand, W. (2023). Two decades of agent-based modeling in marketing: A bibliometric analysis. *Progress in Artificial Intelligence*, *12*(3), 213-229.

Sanchez-Cartas, J. M. (2018). Agent-based models and industrial organization theory. A price-competition algorithm for agent-based models based on Game Theory. *Complex Adaptive Systems Modeling*, *6*(1), 2.

Sadeqi-Arani, Z., & Roozmand, O. (2024). A Review of Three Decades Using Agent-Based Modelling and Simulation in Marketing and Consumer Behavior. *Journal of Optimization in Industrial Engineering*, *1*(16), 2.

Sun, B., Neslin, S. A., & Srinivasan, K. (2003). Measuring the impact of promotions on brand switching when consumers are forward looking. Journal of Marketing Research, 40(4), 389-405.

Supantha, S., & Sharma, N. K. (2024). A Dynamic Agent Based Model of the Real Economy with Monopolistic Competition, Perfect Product Differentiation, Heterogeneous Agents, Increasing Returns to Scale and Trade in Disequilibrium. *arXiv preprint arXiv:2401.07070*.

Taghikhah, F., Filatova, T., & Voinov, A. (2021). Where does theory have it right? A comparison of theory-driven and empirical agent based models. *Journal of Artificial Societies and Social Simulation*.

Zhang, L., Levinson, D. M., & Zhu, S. (2008). Agent-based model of price competition, capacity choice, and product differentiation on congested networks. *Journal of Transport Economics and Policy, 42*(3), 435–461.

Zhang, N., & Zheng, X. (2019). Agent-based simulation of consumer purchase behaviour based on quality, price and promotion. *Enterprise Information Systems, 13*(10), 1427–1441. https://doi.org/10.1080/17517575.2019.1654133