Demand Learning under Limited Inventory: a simulation-based study

Master's Thesis submitted

to

Prof. Dr. Stefan Lessmann

Humboldt-Universität zu Berlin
School of Business and Economics
Chair of Information Systems

by

Denis Augusto Pinto Maciel
(505471)

in partial fulfillment of the requirements
for the degree of
Master of Science
Berlin, October 27, 2021

Acknowledgement

To Francisco and António.

Contents

List of Figures

List of Tables

1. Introduction

Pricing research draws on economics, computer science, operations research, marketing, and business analytics. Its practical importance is straightforward: every business must decide what to charge for its products or services.

Dynamic pricing concerns how firms adjust prices as market conditions change. It is particularly relevant when prices can be updated quickly and at little cost. The growth of e-commerce has made this possible for large inventories: an online retailer can update thousands of prices without replacing physical price tags. These low "menu costs"—the costs of changing posted prices—have contributed to the expansion of dynamic pricing research since the early 2000s.

The appropriate pricing strategy depends on the market. A firm may know its demand curve or need to learn it; it may operate alone or face competitors; and it may replenish inventory freely or sell a fixed stock before a deadline. This thesis focuses on a firm selling one product from limited inventory over a finite, discrete-period selling season. The firm does not know the demand function and must learn it while setting prices. Its central problem is the trade-off between exploration and exploitation: testing prices to learn about demand can reduce current revenue, but improve subsequent decisions.

The analysis develops this problem in stages. Section "Pricing with limited inventory" considers a seller that cannot replenish inventory and knows a deterministic demand function. It explains how prices allocate inventory across the selling season. Section "Learning demand" then introduces unknown demand and compares strategies for balancing learning with revenue maximization. Section "Learning demand with limited inventory" combines demand learning and inventory constraints through the TS-Fixed and TS-Update algorithms of Ferreira, Simchi-Levi, and Wang 2018. Section "Replication of Online Network Revenue Management with Thompson Sampling" reproduces their numerical results. Finally, Section "Competition" places pricing agents in a shared market, where each firm's demand also depends on its competitors' prices. In these simulations, competition substantially reduces the advantage of Thompson-sampling strategies over greedy pricing.

This progression separates the effects of limited inventory and demand learning before examining their interaction. Numerical examples and code implementations accompany the explanations so that the reader can connect each algorithm to its behavior. Appendix A provides instructions for reproducing the analysis and figures.

Simulation is used both to evaluate pricing strategies and to explain their behavior. It allows researchers and practitioners to examine how an algorithm responds to a particular market setting, rather than relying only on its formal description. The Python package `dynpric`, its companion `simulations` package, the manuscript, and the figure-generation code are maintained together at https://github.com/denismaciel/academia/tree/main/theses/master-demand-learning. Earlier standalone repositories are historical archives. Publishing the implementation alongside the thesis makes the analysis easier to inspect, reproduce, and extend.

2. Pricing with limited inventory

2.1. Overview

Fashion retailers and other sellers of seasonal goods often cannot replenish inventory during the selling season because production lead times are too long. Their pricing decisions must therefore respect a fixed stock: total sales cannot exceed the inventory available at the start of the season.

Demand may also change over time. A summer collection, for example, may attract its strongest demand early in the season, with interest declining as autumn approaches. Airline tickets combine a fixed number of seats with demand that can increase as departure approaches. In both cases, prices determine when scarce inventory is sold.

How should a seller allocate fixed inventory over time? We begin with two simplifying assumptions: demand is deterministic, and the seller knows the demand function in every period. Demand then depends on the current price $p(t)$ and period t, and is denoted by $d(t, p(t))$.

The assumptions of a deterministic demand, a well-informed seller and some regularity conditions on the demand function[1] allow the inventory allocation problem to be solved in a relatively simple way. The seller's maximization problem boils down to achieving the highest revenue r given the constraint that total demand across all periods must not exceed the initial inventory C.

$$\begin{aligned} \max & \sum_{t=1}^{T} r(t, d(t)) \\ \text{s.t.} \quad & \sum_{t=1}^{T} d(t) \leq C \\ & d(t) \geq 0 \end{aligned} \tag{1}$$

Solving the optimization problem, we have that

$$J(t, d^*(t)) = \pi^* \tag{2}$$

where π^* is the Lagrange multiplier on the inventory constraint and J is the marginal revenue defined as $J(t, d) \equiv \frac{\partial}{\partial d} r(t, d)$.

π^* can be interpreted as the **marginal opportunity cost of capacity**. For every inventory unit allocated to period t, there is an opportunity cost of capacity represented by the revenue that such a unit would generate if it were allocated to a different period $t + x$. Equation 2 tells us that marginal revenue must equal marginal opportunity cost of capacity at the optimum.

[1] According to Talluri and Van Ryzin 2004, the regularity conditions are: 1) The demand is continuously differentiable and strictly decreasing on price $d'(t, p) < 0$, which implies it has an inverse $p(t, d)$. 2) For sufficiently high prices, demand is zero $\inf_p d(t, p) = 0$. 3) The revenue function $r(t, p) = pd(t, p) = p(t, d)d$ is finite. 4) The marginal revenue J as a function of demand is strictly decreasing in d, i.e. $J(t, d) \equiv \frac{\partial}{\partial d} r(t, d) = p(t, d) + dp'(t, d) < 0$

For an allocation to be optimal, the marginal revenue must be equal across all periods. If that does not hold for a specific allocation, the seller can always increase her revenue by moving one unit from a period with lower marginal revenue to a period with higher marginal revenue. By doing so, she increases revenue, which implies that an allocation cannot be optimal if marginal revenue and the marginal opportunity cost of capacity are not equal.

The optimality condition stated in Equation 2 (the marginal revenue must equal the marginal opportunity cost of capacity) is equivalent to the statement that **marginal revenue must be equal across all periods**. The marginal opportunity cost of capacity of period t is equal to the highest marginal revenue among all other periods $z \neq t$. From this definition and after making sure that Equation 2 holds for every period t, it follows that all periods will have the same marginal revenue at the optimum.

2.2. A numerical example

Consider the two-period example in Talluri and Van Ryzin 2004. Demand is $d_1 = -p_1 + 100$ in the first period and $d_2 = -2p_2 + 120$ in the second. Without an inventory constraint, the seller can maximize each period's revenue independently:

$$r_1 = p_1(-p_1 + 100) \rightarrow p_1^* = 50, q_1^* = 50$$
$$r_2 = p_2(-2p_2 + 120) \rightarrow p_2^* = 30, q_2^* = 60$$

With inventory $C = 40$, the unconstrained quantities are infeasible: $q_1^* + q_2^* = 50 + 60 > C = 40$. The seller must instead divide 40 units between the two periods. For this small example, we can evaluate every feasible integer allocation. Table 1 shows that revenue is highest when 27 units are allocated to period 1, whose customers are less price-sensitive, and 13 to period 2. At this allocation, marginal revenues are as close as the integer constraint permits, consistent with the preceding argument.

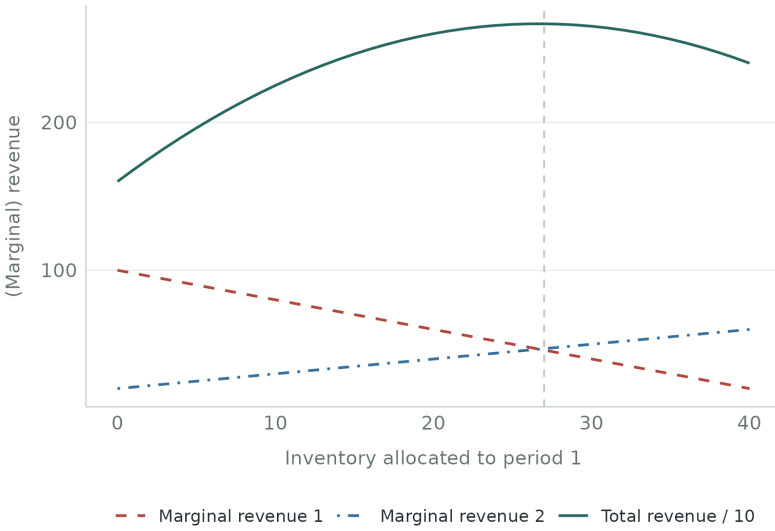
2.3. Solving the allocation problem algorithmically

Although a brute-force approach works for short sales seasons, it becomes very computationally expensive as the length of the sales seasons increases. In the following, we present a more efficient algorithm to optimally allocate inventory for a simulated 10-period sales season where each period has a (different) linear demand. The demand function for all periods have the form $q(p) = \alpha - \beta p$. For each period, the intercept α is sampled from the set of integers within the interval $[100, 200]$ with equal probability. Likewise, the slope β is sampled from the set of integers $[5, 20]$. Inventory is set to 500 units.

Table 2 describes the demand functions through their intercepts and slopes, and reports marginal revenue and quantity at the optimal allocation. Figure 1 plots revenue against quantity for each period. The marked points identify the optimal allocation. Revenue levels differ across periods, but the slopes at those points are nearly equal, reflecting the marginal-revenue condition.

Table 1: Two-period allocation problem with deterministic demand

q_1	q_2	Revenue	J_1	J_2
23	17	2646.5	54	43
24	16	2656.0	52	44
25	15	2662.5	50	45
26	14	2666.0	48	46
27	**13**	**2666.5**	**46**	**47**
28	12	2664.0	44	48
29	11	2658.5	42	49



Period	α	β	Marginal Revenue	Quantity
0	106	13	2.92	34
1	111	18	2.83	30
2	134	8	2.75	56
3	104	17	2.82	28
4	168	15	2.93	62
5	143	6	2.83	63
6	120	9	2.89	47
7	143	15	2.87	50
8	189	12	2.92	77
9	120	5	2.80	53

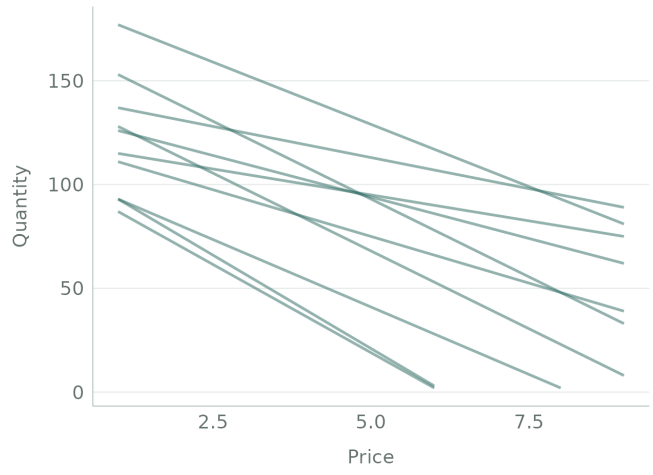


Table 2: Demand functions and optimally allocated quantities for a 10-period sales season

Enumerating every feasible allocation becomes expensive as the number of periods grows. Under the stated assumptions, the following procedure allocates inventory more efficiently:

1. Find the period with the highest marginal revenue.
2. If its marginal revenue is greater than zero, allocate one unit of inventory to it. Else, stop the algorithm.
3. Check if there is still inventory left. If yes, return to 1. Else, stop the algorithm.

This simple algorithm is guaranteed to deliver an optimal allocation and was used to compute the results in Table 2. It, however, requires strong assumptions that are unlikely to hold in a real-life pricing scenario. First, it assumes demand is deterministic. Second, it assumes the seller knows the demand function.[2] Allowing for a stochastic demand and imposing that the seller needs to learn the demand shape while setting prices unveils a new level of complexity that will be tackled in the next sections.

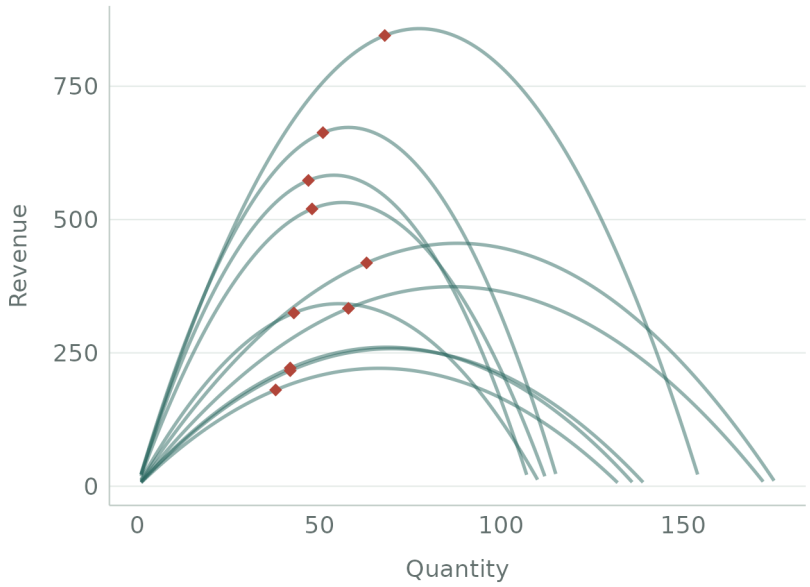


Figure 1: Revenue functions and optimal quantity allocations over a 10-period selling season

[2]To be precise, these are not the only two assumptions that we are making. In fact, there are many scenarios where the algorithm wouldn't yield optimal results. For example, we assume that customers don't act strategically or, put differently, that customers are myopic. Myopic customers do not consider that the seller might change prices in the future making it eventually more attractive for them to delay the purchase. Faced with a price today, myopic customers make a purchase decision based solely on the current price. The myopic customer assumption allows sellers to model the demand as a function of the current price only without the need to consider how a price set today might affect the demand of a future period.

Moreover, we also consider the selling product in isolation. The more realistic scenario in which prices of related goods influence a product's demand is not considered. Essentially, the assumption is that there are no substitutes or complements to the product, which are capable of affecting its demand.

3. Learning demand

3.1. Exploration and exploitation

A seller may need to learn demand while setting prices. Each price and resulting quantity provide information that can improve later decisions. Learning is costly, however: testing a price may sacrifice revenue that could have been earned by choosing the price currently believed to be best. This is the exploration-exploitation trade-off. Exploitation uses existing information to maximize expected current revenue; exploration gathers information that may improve future revenue.

The exploration-exploitation trade-off appears in statistics, economics, operations research, computer science, and electrical engineering (Russo et al. 2017). The multi-armed bandit problem provides a standard illustration. A player repeatedly chooses among several arms, each with an unknown reward distribution. Playing an arm reveals a reward and provides information about that arm. At each round, the player must decide whether to exploit the arm with the highest estimated reward or explore an alternative that may prove more rewarding.

3.2. Pricing as a multi-armed bandit problem

Dynamic pricing can also be modeled as a multi-armed bandit problem. Consider a seller of a single product. In each period, the seller is allowed to choose one price p_j out of a fixed set of discrete prices $P = \{p_1, p_2, p_3, \ldots, p_n\}$. The seller faces a stream of identical customers. For each price, there is a probability that determines how likely a customer is to buy when faced with that price. Those probabilities are completely unknown to the seller. The following describes the sequence of events that happen in a given period of the sales season:

1. The customer enters the store.
2. The store sets a price $p_j \in P$.
3. The customer buys one unit with probability θ_j.
4. The store observes the customer's decision and updates its beliefs about θ_j.

In this analogy, prices are the arms, a purchase is a successful outcome, and the revenue from that outcome is the chosen price p_j. Testing different prices allows the seller to learn their purchase probabilities.

3.3. Processing new information

Before comparing pricing strategies, we must specify how the seller learns. After setting p_j in period $t-1$, the seller observes whether the customer buys. That observation changes the information available in period t. A rule such as "choose the best price" therefore depends on both the seller's current beliefs and the way those beliefs are updated.

We assume Bayesian learning. Consider a single price p_j with unknown purchase probability θ_j. The seller represents uncertainty about this parameter with a **prior distribution**. After

observing a purchase or nonpurchase, the seller applies Bayes' rule to obtain a **posterior distribution**. The posterior incorporates the new observation and becomes the prior for the next update. This process is called **Bayesian updating**.

It is beyond the scope of this work to explain how Bayesian updating works in general.[3] But it is still useful to illustrate the process of Bayesian updating with a concrete pricing example. Again, let us consider a price p_j in isolation and assume $\theta_j = 0.7$. As the prior, we will choose $Beta(1, 1)$, a beta distribution with parameters alpha and beta set to 1. A $Beta(1, 1)$ distribution is equivalent to the uniform distribution. It implies that the seller considers any value between 0 and 1 to be the true value of θ_j with equal probability. The beta distribution is also a convenient choice because the Bayesian updating is trivial to compute. Given that a seller sets price p_j and the prior is $Beta(\alpha_j, \beta_j)$, the parameters of the posterior distribution will be:

$$(\alpha_j, \beta_j) \leftarrow \begin{cases} (\alpha_j + 1, \beta_j + 0) & \text{if customer buys} \\ (\alpha_j + 0, \beta_j + 1) & \text{otherwise.} \end{cases} \tag{3}$$

In such a setting, we let the seller set p_j and observe the purchasing decision for 1,000 consecutive periods. Figure 2 displays the prior and posterior for the first nine periods of the simulation. Next to the period number, the purchasing decision is expressed: the green checkmark means a purchase happened; the red X means no purchase happened. Take the first period. The prior is the flat red line meaning that the seller believes any value between 0 and 1 is equally likely. Then, she observes a purchase. She then updates her prior with this observation and now her beliefs assign higher probability mass to numbers closer to 1 than to 0. In period 2, the seller observes a no-purchase decision. The updating leads to a posterior where values closer to 0 are more likely than before. The process goes on and on in this fashion. It is important to notice that the prior of period t is simply the posterior of period $t - 1$. It is also important to notice that as time passes the changes from prior to posterior become milder. That is the case because in the ninth period, for example, after having observed 8 purchase decisions, an extra data point is not as informative as in, say, the second period, where the seller had only seen one data point.

Figure 3 shows how the beliefs develop as the seller gets to see more and more data across the 1,000-period sales season. Before seeing any demand, the seller assigns equal probability to all values between 0 and 1. The more data she sees, however, the more concentrated her beliefs get around the true value of 0.7. By the time we reach the last period, the seller believes that it is virtually impossible that any value smaller than 0.65 or greater than 0.75 is the true value of θ_j.

3.4. Tackling the multi-armed bandit problem

With the updating rule established, we can compare how pricing strategies use the resulting beliefs to balance exploration and exploitation.

[3] For an excellent introduction into Bayesian thinking, I enthusiastically recommend McElreath 2020.

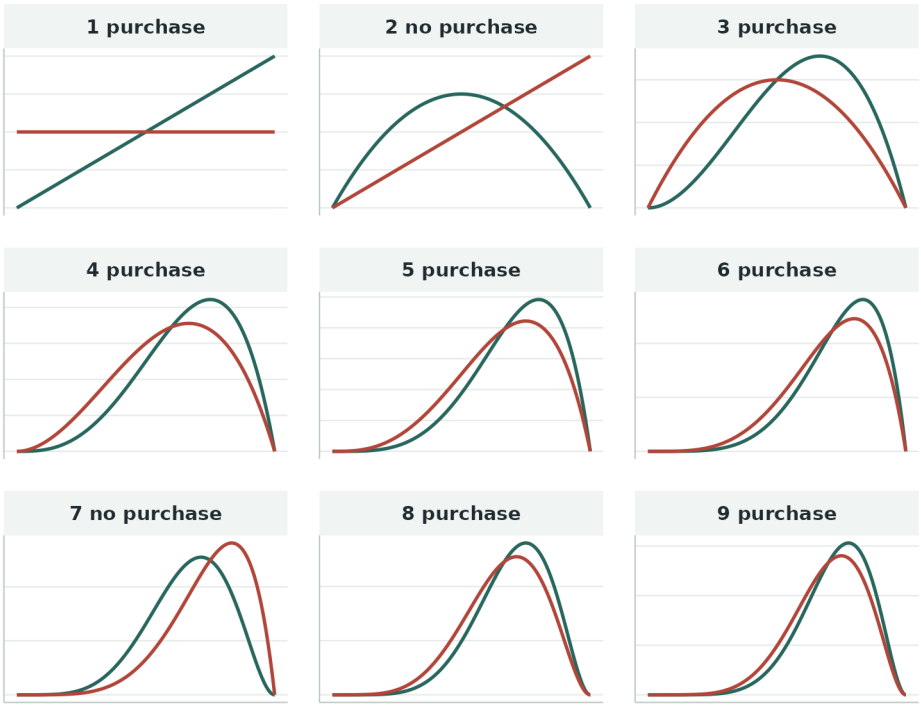


Figure 2: Bayesian Updating in the first 9 periods

Table 3: A four-arm bandit pricing problem

p_j	θ_j	Prior
29.9	0.8	$Beta(1,1)$
34.9	0.6	$Beta(1,1)$
39.9	0.3	$Beta(1,1)$
44.9	0.1	$Beta(1,1)$

Table 3 defines a four-price example. Each price p_j has purchase probability θ_j. The seller begins with a uniform prior for each probability and learns from the observed purchase decisions.

Greedy. A first strategy to tackle the exploration and exploitation trade-off is to ignore exploration and always set the price believed to yield the highest revenue. Since the seller believes $\theta_j \sim Beta(1,1)\ \forall\ j$, the expected value of purchase probability for every price is 0.5.[4] $p_4 = 44.9$ is the highest price and yields the highest expected revenue $\mathbb{E}[r] = 44.9 * 0.5$ according to the seller's priors. Thus, p_4 will be the price set in the first period by a greedy seller. By setting p_4 and observing the outcomes, the seller's belief about θ_4 will start to converge to its true value of 0.1 and the estimated expected revenue of p_4 will decrease. The seller will eventually flip the price set to p_3 and a similar process will ensue. The seller alternates between p_4 and p_3 updating her beliefs until p_2 becomes the most attractive price. Once p_2 is finally set, it will likely be played until the end of the season since the revenue

[4]The expected value of a random variable $X \sim Beta(\alpha, \beta)$ distribution is $\mathbb{E}[X] = \frac{\alpha}{\alpha+\beta}$

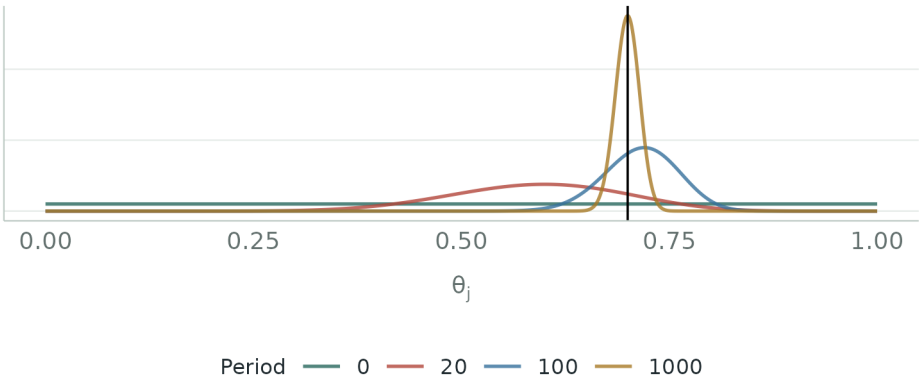


Figure 3: The shape of the beliefs distribution changes as the agent gathers more observations

estimate associated with it will increase in the eyes of the seller. The most profitable price p_1 might end up never being applied. With a greedy strategy, the seller is likely to get stuck with a suboptimal price strategy of setting p_2 and exploration only happens as a byproduct of updating beliefs about the currently-believed-to-be-the-best price.

ε-Greedy. The ε-Greedy strategy is one way to address the lack of exploration of a purely greedy strategy. The way ε-Greedy works is by setting a constant ε, which is usually a small number (e.g. 5%). Before every period, the seller samples a number from $Uniform(0, 1)$. If the sampled number is greater than ε, the seller exploits by applying the greedy strategy. If the number is less than or equal to ε, the seller randomly selects one of the available prices with equal probability in order to explore. That way ε% of the time will be explicitly devoted to exploration of different prices, while the remaining $(1 - \varepsilon)$% will serve to exploit.

Thompson Sampling. Thompson sampling does not differentiate between exploration and exploitation as the ε-Greedy algorithm. In fact, it is very similar to the way the purely greedy algorithm works except for one step. In Thompson sampling, the estimate of the unknown parameter is not the expected value of the prior. It instead *samples* the estimate $\hat{\theta}_j$ from the prior distribution. From then on, the Thompson sampling algorithm proceeds exactly as the greedy one and sets the price that maximizes expected revenue given $\hat{\theta}_j$.

3.5. Comparing strategies through simulation

We compare the three algorithms in the multi-armed bandit setting of Table 3, using 3,000 trials of 500 periods each. Figure 4 shows the distribution of total selling-season revenue. Greedy has the lowest median, slightly below 11,000, and substantial variation across trials; its worst outcomes are less than half its median. Thompson sampling has the highest median and a much narrower interquartile range. None of its simulated trials yields revenue below 9,000. Its advantage in this experiment therefore concerns both average performance and consistency.

Figure 5 shows how often each price is selected in each period. Greedy frequently remains

at the suboptimal price $p_2 = 34.9$, rather than discovering the optimal price $p_1 = 29.90$. Its price frequencies show little movement toward the end of the season. Explicit exploration helps ε-Greedy discover the optimal price, but Thompson sampling does so faster. By the final periods, Thompson sampling selects the optimal price almost 90% of the time, compared with about 75% for ε-Greedy. During exploration, ε-Greedy continues to sample all prices equally. Thompson sampling instead concentrates on prices that retain a plausible chance of being optimal.

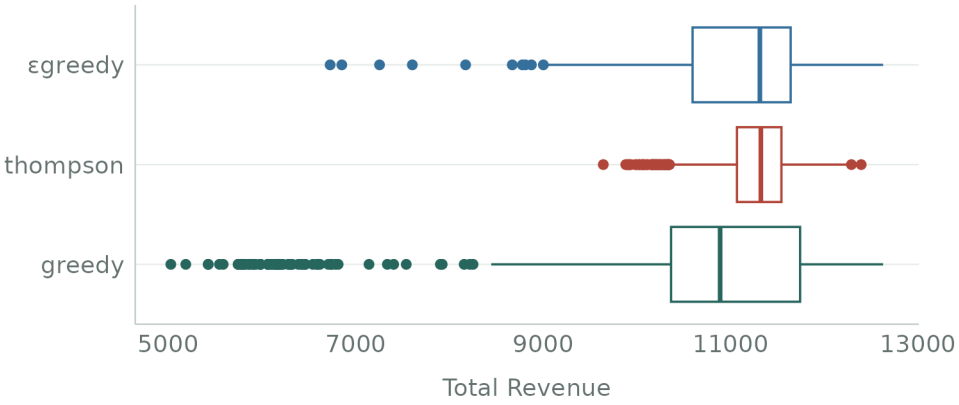


Figure 4: Distribution of total revenue across trials

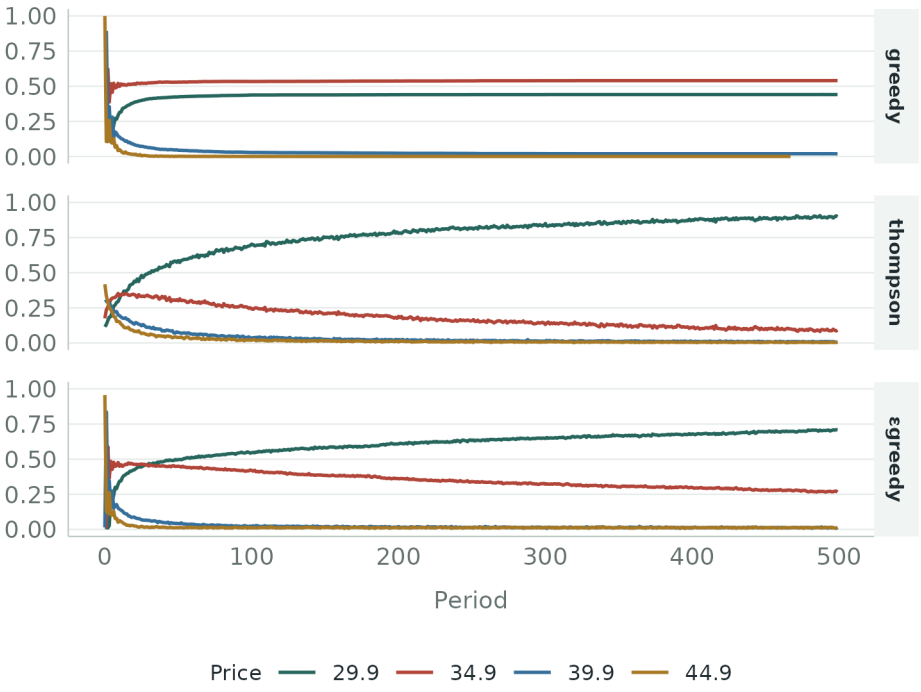


Figure 5: Frequency with which each price is chosen, by period

4. Learning demand with limited inventory

The preceding sections considered two sources of difficulty in pricing: allocating scarce inventory over time and learning unknown demand. In practice, these problems can occur together. Ferreira, Simchi-Levi, and Wang 2018 address their interaction through the online network revenue management problem. This section introduces that setting and the TS-Fixed algorithm; the next describes TS-Update and reproduces the results of Ferreira, Simchi-Levi, and Wang 2018.

A retailer begins a finite selling season with fixed inventory and seeks to maximize total revenue. The retailer observes demand as sales occur and can adjust prices at negligible cost, but does not know the demand function. Testing prices reveals demand information while consuming both selling time and inventory. Exploration must therefore leave enough inventory to exploit what has been learned.

Here, *online* means that decisions and learning occur sequentially as demand is observed. *Network* refers to products that consume common, limited resources: allocating a resource to one product reduces what remains for others. The framework accommodates multiple products and resources. Following the numerical analysis of Ferreira, Simchi-Levi, and Wang 2018, we focus on a single product using a single resource.

4.1. Market setting

Generally, the setting can be formally described as follows. A retailer sells N products indexed by $i \in [N]$ during a fixed-length selling season. The products consume M resources indexed by $j \in [M]$. The constants a_{ij} represent how much of resource j product i requires to be produced. The retailer starts the season with I_j units of inventory of resource j and restocking is not allowed. The selling season lasts for T periods. In each of these periods the sequence of events occur in the order listed below:

- The retailer sets a price for each product by choosing one of the allowed price vectors from the set $\{p_1, p_2, \ldots, p_K\}$ where p_k $(\forall k \in [K])$ is a vector of length N and each of its elements represents the price of one product.
- Faced with the prices set by the retailer, customers make a purchase decision. The resulting demand for every product i in period t is represented by the vector $D(t) = (D_1(t), D_2(t), \ldots, D_N(t))$. $D(t)$ is sampled from a probability distribution on $\mathbb{R}_+^N$.
- Depending on the amount of inventory left, one of the following events happens:
  - If there is enough inventory, the retailer earns $\sum_{i=1}^{N} D_i(t)P_i(t)$ in revenue and has her inventory depleted such that $I_j = I_j(t-1) - \sum_{i=1}^{N} a_{ij}$.
  - In case the demand is higher than the available inventory, the part of the demand that cannot be satisfied gets lost. The retailer's revenue is $\sum_{i=1}^{N} \tilde{D}_i(t)P_i(t)$ where $\tilde{D}_i$ is the demand that could be satisfied for product i.

The concrete scenario used for the simulation consists of one product whose production consumes one limited resource in a 1:1 proportion (one unit of product consumes one unit of resource).[5] That is, $N = 1$, $M = 1$ and $a_{11} = 1$. The selling season lasts for 500 periods ($T = 500$) and the initial inventory of the single resource is set to be a fraction of the selling season length: $I_1 = 0.25T$. The seller can choose among the prices 29.9, 34.9, 39.9, 44.9.[6] Each price is associated with a demand parameter that is entirely unknown to the retailer at the beginning of the season. In each period, the demand can be either 0 or 1 unit. The demand is a Bernoulli trial and is fully characterized by a single parameter. The parameter expresses the probability that one product will be sold (that is, demand will be 1) at a certain period. The retailer can influence the demand parameter by setting different prices.

The correspondence between price and demand parameters is the following:

- When the price is 29.9, demand equals 1 with probability 0.8.
- When the price is 34.9, demand equals 1 with probability 0.6.
- When the price is 39.9, demand equals 1 with probability 0.3.
- When the price is 44.9, demand equals 1 with probability 0.1.

Higher prices have lower purchase probabilities. The retailer does not know this relationship at the start of the season and must learn it by testing prices and observing demand.

4.2. TS-Fixed

We describe in the following how the TS-Fixed algorithm works.[7]

4.2.1. Beliefs

The retailer uses a $Beta(1, 1)$ prior for each price's purchase probability, equivalent to a uniform distribution over $[0, 1]$. These priors do not encode a downward-sloping demand curve: before observing any sales, the retailer assigns the same prior distribution to every price. The simulations show how the resulting beliefs change as observations accumulate.

4.2.2. Optimization problem

TS-Fixed selects a price in period t in three steps:

1. The retailer needs to estimate the demand. She must come up with a demand parameter for every price.

[5] Since every unit of product consumes exactly one unit of the single resource for its production, we can therefore ignore the difference between product and resources and refer directly to a limited product inventory.

[6] The reader will notice that the demand parameters and the way the seller processes information are the same as in the section Learning demand. We have picked the same numbers as Ferreira, Simchi-Levi, and Wang 2018 to lessen the cognitive overhead and provide a sense of continuity between sections.

[7] TS-Update will be described in the next section where the main results of Ferreira, Simchi-Levi, and Wang 2018 are reproduced. The differences between the two algorithms, as the reader will see, are very small.

2. The retailer uses those estimates in an optimization procedure to compute the optimal price mixture.

3. Finally, the retailer samples one price from the mixture and applies it on period t.

Demand estimation. The first step is to transform the beliefs about the demand into a number to be used by the optimization procedure. For TS-Fixed, this is done via Thompson sampling: the demand parameters (q_1, q_2, q_3, q_4) are sampled from the posterior beta distribution associated with each element of the price vector. An alternative to Thompson sampling, for instance, would be to use the greedy approach and compute the mean of the distribution instead of sampling from it.

Optimization. The result of the previous step is a set of pairs of prices and demand parameters. With this information, the retailer needs to solve an optimization problem. She wants to maximize revenue, while also managing a limited inventory. The simulation setting imposes that the retailer starts with a fixed amount of product inventory and is not allowed to replenish it during the sales season. TS-Fixed deals with the inventory constraint in a rather static way. It enforces that the expected demand resulting from the optimization is less than or equal to a constant c, which is the ratio between the inventory and the length of the selling season. c is set at the beginning of the selling season and is not updated to reflect the actual development of the inventory. That is the reason why it bears "fixed" in its name.

The optimization chooses probabilities for offering each price. The inventory constraint limits expected demand to the allowance c; the remaining constraints ensure nonnegative probabilities with total mass no greater than one.

$$
\begin{aligned}
\max_{x_1,x_2,x_3,x_4} \quad & Q = (p_1 q_1)x_1 + (p_2 q_2)x_2 + (p_3 q_3)x_3 + (p_4 q_4)x_4 \\
\text{s.t.} \quad & Q \leq c \\
& x_1 + x_2 + x_3 + x_4 \leq 1 \\
& x_1, x_2, x_3, x_4 \geq 0 \\
\text{where} \quad & c = \frac{I}{T}
\end{aligned}
$$

.

Price setting. The optimization result is a vector of probabilities $x = (x_1, x_2, x_3, x_4)$. The final step is to use x to set the actual price for period t. This is done by sampling one $p_i \in p$ from a distribution where each p_i has probability mass $x_i \in x$.

4.2.3. Simulation results

At the end of each period, we recorded metrics about the state of the trial. The information gathered was:

- Price: the price set by the seller

- Period revenue: the revenue collected
- Beliefs about demand: the mean value of the distribution of the demand parameter the seller believes is associated with each price.

The results are stored in a data frame that has a row for each period of every trial with the information mentioned above.

Revenue. Figure 6 reports revenue in each period, averaged across trials. The horizontal reference line gives the clairvoyant seller's average revenue; the varying line gives TS-Fixed's revenue. TS-Fixed initially earns about half the clairvoyant benchmark. A clairvoyant seller offers 39.9 with probability 0.75 and 44.9 with probability 0.25. TS-Fixed approaches the benchmark after about 150 periods, or 30% of the selling season, and subsequently fluctuates around 10 dollars per period.

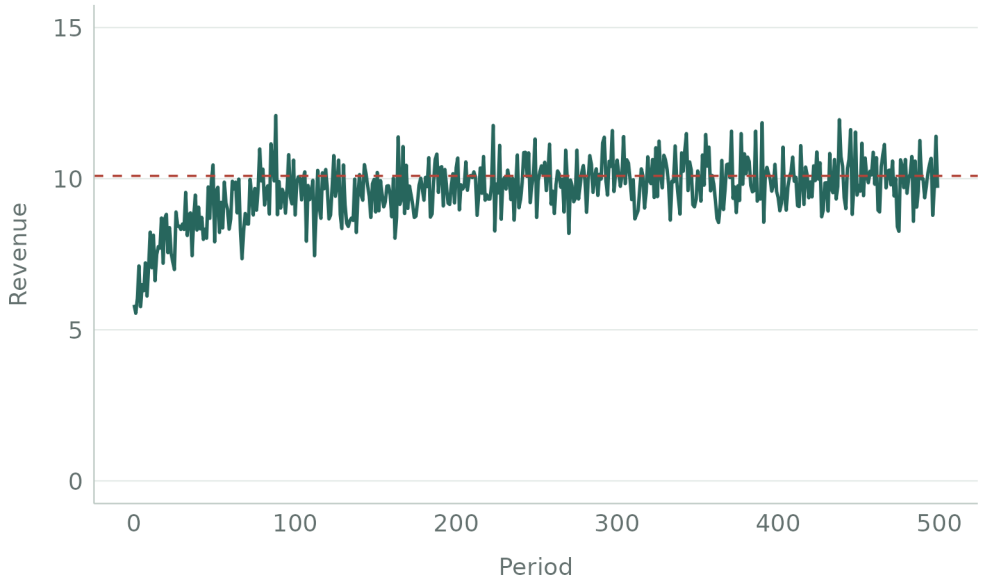


Figure 6: Average revenue per period across trials

Offered prices. Figure 7 shows how often each price is offered in each period. The price 44.9 dominates initially because all prices have the same $Beta(1, 1)$ prior: higher prices are attractive when their purchase probabilities are not yet distinguished by observations. As the retailer learns, the price mixture approaches the clairvoyant policy. Around period 300, TS-Fixed offers 39.9 about 75% of the time and 44.9 the remaining 25%.

Learning process. Each observation combines an offered price with realized demand. Starting from a $Beta(1, 1)$ prior for each price, the seller updates $Beta(\alpha, \beta)$ to $Beta(\alpha + 1, \beta)$ after a purchase and to $Beta(\alpha, \beta + 1)$ after a nonpurchase.

Figure 8 shows the average expected value from the posterior distribution of the demand parameter over time for each available price. For the very first period, the expected value averaged across simulations is close to 0.5 for all prices but 44.9. Why is that the case?

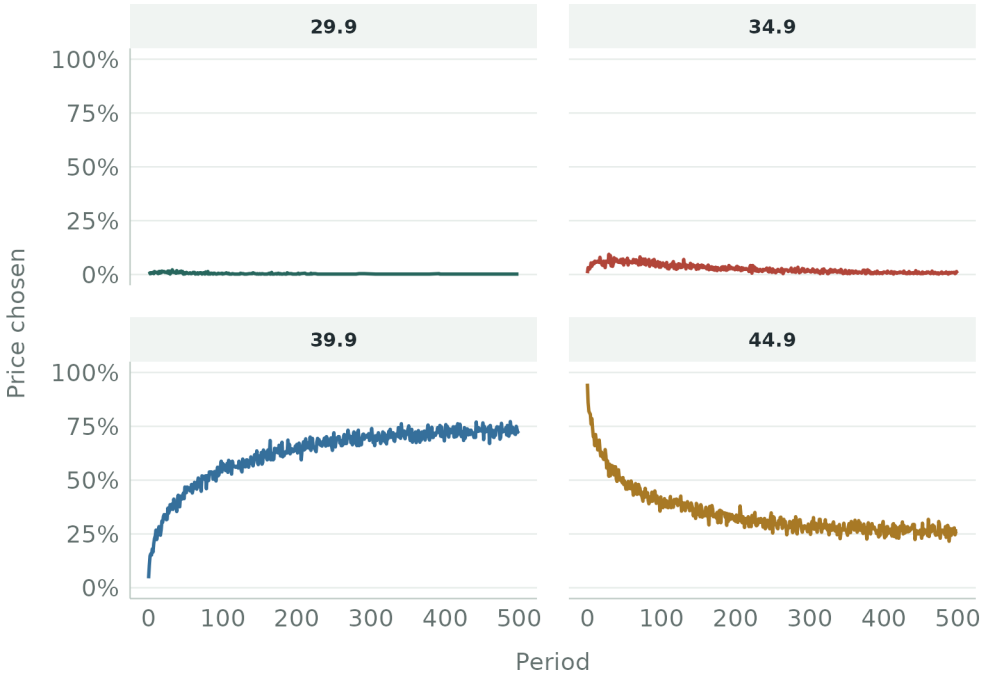


Figure 7: Frequency with which each price is offered, by period

Because 44.9 is by far the most often chosen price in the first period and for that reason, it is the first price to have its beliefs updated. When the price is 44.9, demand will be 1 with only 0.1 probability. The posterior after the first period will then be $Beta(1,2)$ 90% of the time and $Beta(1,2)$ the other 10%. In fact, it is straightforward to calculate the expected value of the mean of the posterior distribution of 44.9 right after the first period.

Knowing that the expected value of $X \sim Beta(\alpha, \beta)$ is $E[X] = \frac{\alpha}{\alpha+\beta}$ and that $q|p = 44.5 \sim Beta(1,1)$ in the first period, we have

$$E[q|p = 44.5] = 0.9\frac{1}{3} + 0.1\frac{2}{3} = 0.3\bar{6} \tag{4}$$

which is roughly the starting point of the line 44.9 in the plot.

Over the selling season, the retailer learns the purchase probabilities at 39.9 and 44.9 accurately and approaches the true probability at 34.9. The estimate at 29.9 remains inaccurate. This reflects the allocation of exploration: Thompson sampling concentrates observations on prices that appear promising, rather than estimating every purchase probability equally precisely.

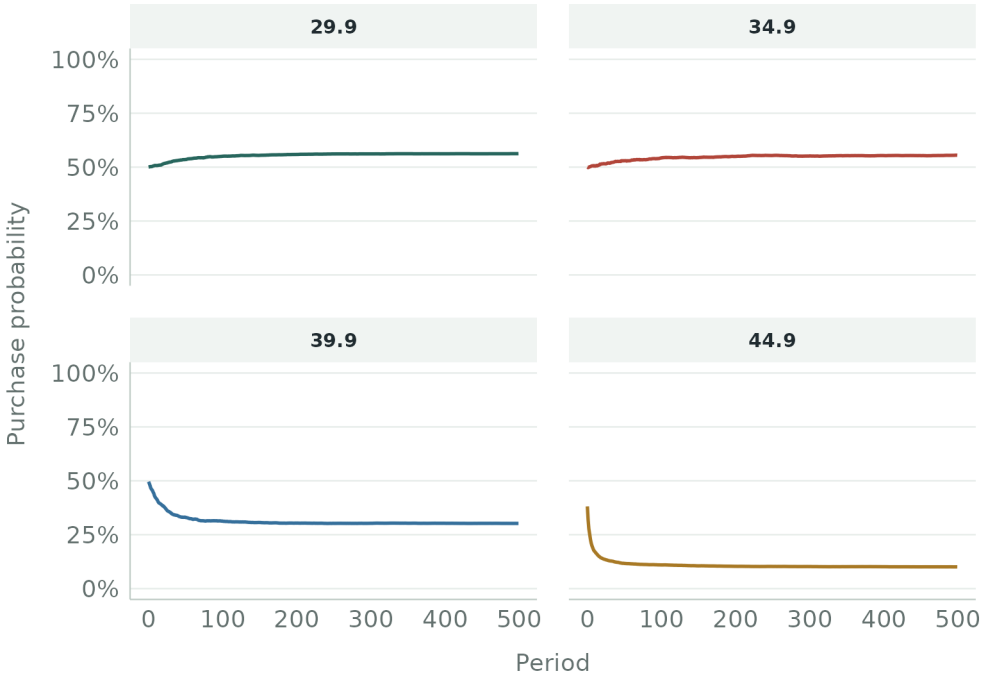


Figure 8: Beliefs about demand parameters over time averaged across trials

5. Replication of Online Network Revenue Management with Thompson Sampling

After the deep dive on the inner workings of TS-Fixed, we now reproduce the results of Ferreira, Simchi-Levi, and Wang 2018. Alongside TS-Fixed, the TS-Update algorithm is also implemented. The two algorithms are very similar. The only difference is that TS-Update takes into consideration the current stock at every period when doing the optimization. TS-Fixed, in contrast, sets a constant factor c across all the simulations. Formally, at period t $c^{Fixed} = \frac{I}{T}$, while $c^{Update} = \frac{I_t}{T-t}$, where I is the total inventory available at the beginning of the selling season and I_t is the (already depleted) inventory available at period t. Finally, a simple Thompson Sampling (TS) algorithm is also implemented that completely ignores the inventory constraint. It simply prices the product as if inventory were unlimited. Once inventory runs out, TS simply goes the rest of the selling season with zero revenue.

Figure 9 reports our replication, and Figure 10 reproduces the corresponding figure from Ferreira, Simchi-Levi, and Wang 2018. The vertical axis gives average revenue relative to the expected revenue of a clairvoyant seller; the horizontal axis gives the selling-season length. Initial inventory in the left panel is half that in the right panel.

The replication closely reproduces the original pattern. TS-Update achieves higher average revenue than TS-Fixed, although the difference narrows as the selling season lengthens. The inventory-unconstrained TS policy generally performs worst, with the exception of the short season with high inventory ($T = 100$ and $I = 0.5T$). Longer seasons widen this performance gap: TS-Fixed and TS-Update improve, while TS loses ground relative to the clairvoyant benchmark.

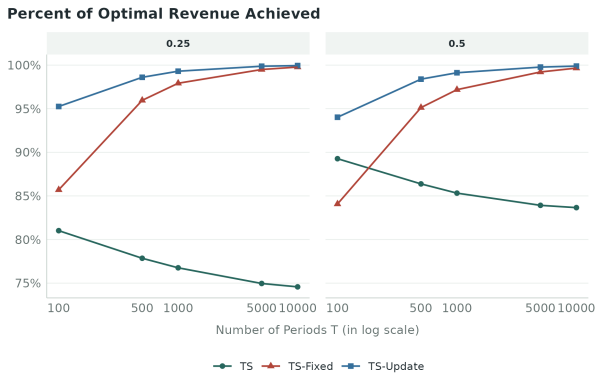


Figure 9: Revenue relative to the clairvoyant benchmark by selling-season length

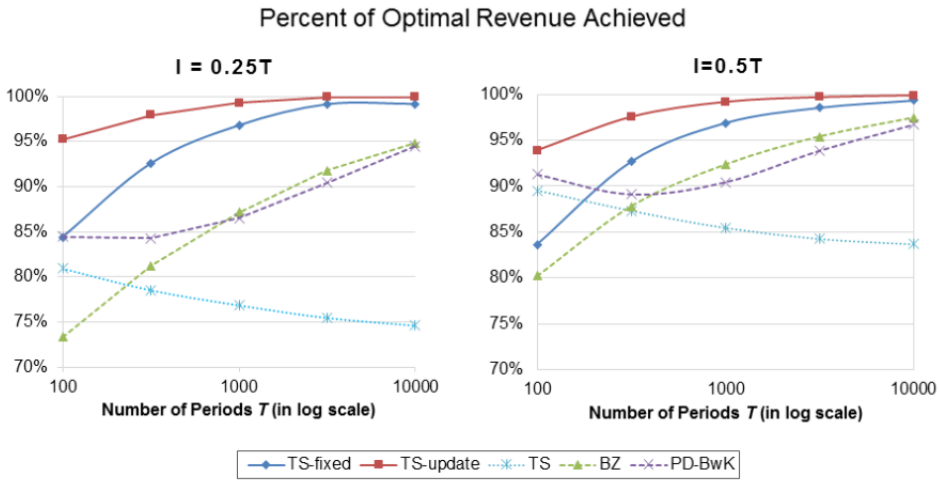


Figure 1 Performance Comparison of Dynamic Pricing Algorithms: Single Product Example

Figure 10: Original results from Ferreira, Simchi-Levi, and Wang 2018

6. Demand

The preceding examples model demand as one customer arriving per period and deciding whether to buy, so sales are either 0 or 1. This simple setting isolates the exploration-exploitation trade-off. To model a market with many customers and competing firms, we now construct demand from individual purchase decisions. Each period contains multiple customers with different willingness to pay.

6.1. Demand from heterogeneous customers

From a firm's perspective, the demand is just a function that takes in a price and outputs the quantity that the firm can sell at that price. In its simplest form, the demand function can be characterized by a simple deterministic linear function of the form $q(p) = \beta_0 + \beta_1 p$. To make the demand *well-behaved*, β_1 is usually set to a negative coefficient, which implies that higher prices lead to lower demand. This is in fact the demand used by Bertsimas and Perakis 2006 with an added random variable $\epsilon \sim N(0, \sigma^2)$ to make the resulting demand stochastic.

In the simulations that follow, however, we chose to compose the demand out of individual customers, similarly to what Geer et al. 2019 has implemented. Instead of a linear function with hand-picked coefficients, the demand here is derived from individual purchase decisions of multiple heterogeneous customers. Customers are heterogeneous in that they have different willingness to pay (WTP). In fact, the WTP fully characterizes how a customer makes purchase decisions. The market dynamics goes as follows. The firm sets a price.[8] Confronted with the price set by the firm, customers decide whether to buy or not.[9] Given a price p, a customer with WTP w will make a purchase if and only if $w \geq p$. After every customer has decided whether to purchase or not, the demand for the firm is simply the total number of customers that have decided to buy.

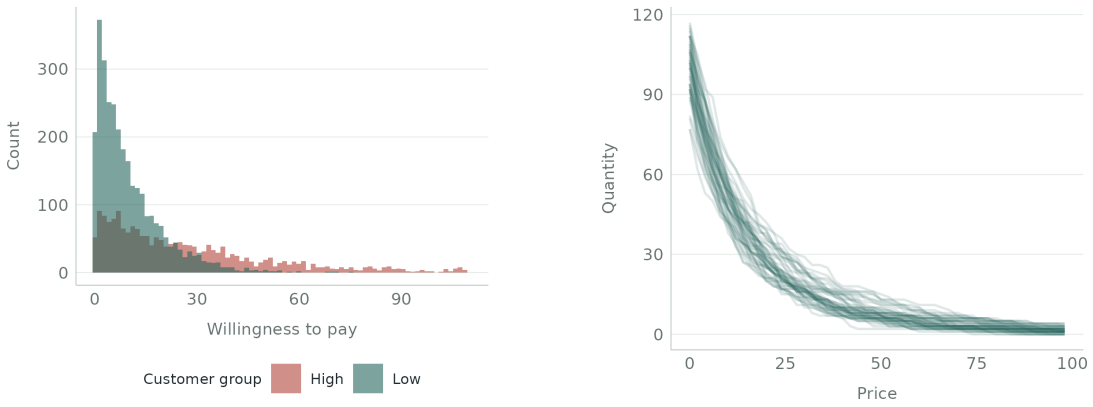
This construction requires two distributions: one for the number of customers and one for their willingness to pay. In each period, the customer count is drawn from a Poisson distribution with mean λ. Individual willingness to pay is drawn from an exponential distribution with mean β.

To illustrate and visualize how a well-behaved[10] demand function can emerge from such a setting, we run a small simulation. In Figure 11, there are two groups of customers: one with a higher average WTP and the other one with a lower WTP. The WTPs from both groups are sampled from an exponential distribution, but group's one are sampled from g_{high}, which has a higher expected value β_{high}.

[8] The firm is not able to discriminate among customers. That is, WTP is private information to the customer and the firm has no access to it. The firm sets one single price per period for all the customers in the market.

[9] For the sake of simplicity, we also assume that every customer can only buy one unit from a single firm at each period.

[10] By well-behaved, we understand a demand that goes down when price increases *ceteris paribus*.



(a) WTP distribution **(b)** Demand

Figure 11: Result of 50 simulations of the aggregated demand with the parameters $\lambda = 100$, $\beta_{low} = 10$, $\beta_{high} = 30$, $\theta_{low} = 0.6$ and $\theta_{high} = 0.4$.

$$g_{low} \sim \exp(\beta_{low})$$
$$g_{high} \sim \exp(\beta_{high})$$
$$\beta_{high} = 1.6\beta_{low}$$

The expected number of customers in the market at any given period is $\lambda = 100$. Let θ_{low} and θ_{high} be the shares of customers whose WTP are sampled from g_{low} and g_{high} respectively. The number of high and low customers is not fixed across periods. In every period t, we sample n_{high} and n_{low} from a multinomial distribution to determine how many customers belong to each group.

$$n_{low}, n_{high} \sim Multinom(n, (\theta_{low}, \theta_{high}))$$

The algorithm to generate a demand function can then be summarized as follows:

1. Sample the total number of customers n from $Poisson(\lambda)$
2. Sample n_{low} and n_{high} from $Multinom(n, (\theta_{low}, \theta_{high}))$
3. Compute n_{low} realizations of the random variable g_{low} and n_{high} realizations of g_{high}
4. Compute the aggregated demand q. For every price p, the demand q equals the count of all customers with WTP greater than or equal to p.

Figure 11 shows 50 simulated demand functions. Each line in the right panel represents one realization: demand falls as price rises, but varies across realizations because customer counts and willingness to pay are random. The left panel shows the willingness-to-pay distributions for the two customer groups.

6.2. Allowing for competition

The preceding construction does not yet specify how customers choose among competing firms. The next section therefore extends demand to depend on several firms' prices, following the approach of Geer et al. 2019.

We assume that the competitors sell identical products and that competition only takes place through price. That is, there is no product differentiation. As before, customers are characterized by WTP, but here there are two types of customers: loyals and shoppers. These two types of customers differ in how they choose a firm, but the final purchase decision will happen as long as $w \geq p$. Loyal customers are randomly assigned to one of the firms in the market and will only buy from the firm they were assigned to. Shoppers, on the other hand, will buy from the company with the lowest price (given that the lowest price is smaller than her WTP).

The algorithm used to determine the demand is the following:

- Sample total number of customers N from a Poisson distribution with mean λ. This means we expect, on average, λ customers to be in the market at any time t. The actual number of customers, of course, need not be exactly λ and will vary from period to period.
- Customers can be of two types: loyals and shoppers. The number of loyals and shoppers is $\theta_{loy}N$ and $\theta_{sho}N$, respectively, where $\theta_{loy} + \theta_{sho} = 1$.
- After determining the numbers of shoppers and loyal customers and assigning loyal customers to firms, sample each customer's willingness to pay. Loyals' WTP are sampled from $Exponential(\beta_{loy} = 40)$ while shoppers' are sampled from $Exponential(\beta_{sho} = 25)$.
- Finally, the demand for each firm can be computed and it is the sum of a) the randomly assigned loyals plus b) the shoppers if the firm had the lowest price,[11] whose WTP is higher or equal to the price set.

[11] In case more than one company set the lowest price, the shoppers are assigned randomly to them.

7. Competition

The preceding simulations show the performance of Thompson-sampling strategies when a firm's demand depends only on its own price. We now let the pricing algorithms compete in the same market. Demand depends on all firms' prices through the customer-choice mechanism described in Section 6.2.

Figure 12 reports average revenue per period for each competitive setting. All panels represent duopolies except `all`, which contains TS-Fixed, ε-Greedy, and Greedy. The Thompson-sampling advantage seen without competition largely disappears. Against Greedy, TS-Fixed performs slightly worse and TS-Update only slightly better. ε-Greedy performs worse than both Greedy and TS-Fixed. In the panel `ts_fixed_vs_itself`, A and B are two competing TS-Fixed firms.

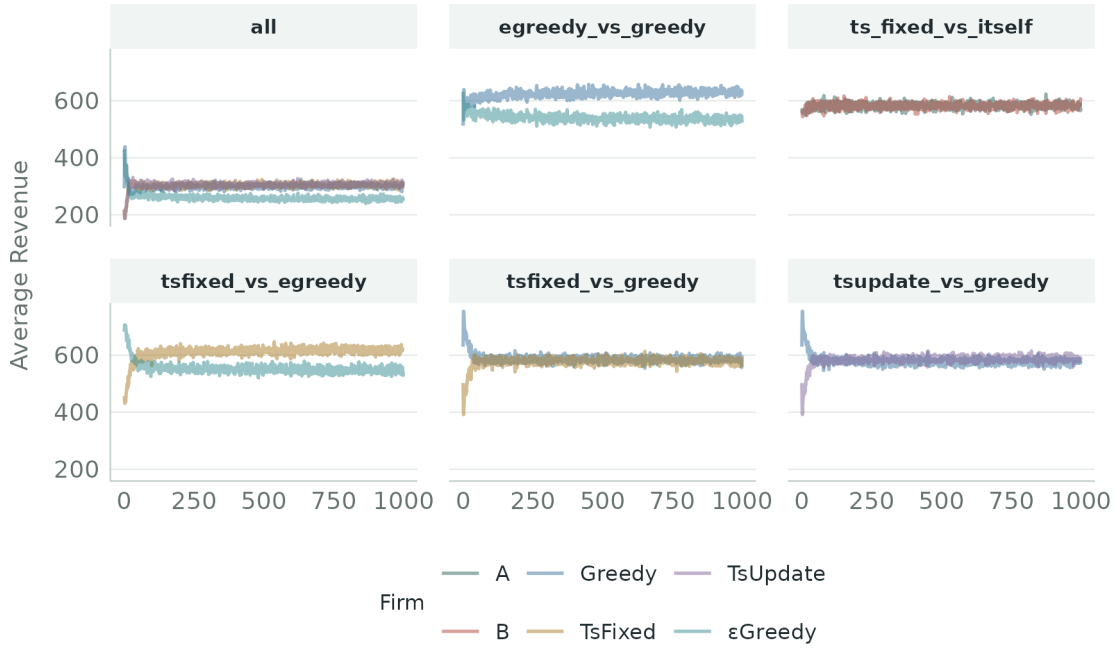


Figure 12: Average revenue across trials per period

Figure 13 summarizes how the interaction between pricing strategies of competing firms affects revenue and helps to get a better understanding of the market structure. The data underlying the plot stems from the duopoly consisting of two equal firms: TS-Fixed (firm A) vs TS-Fixed (firm B). The plot shows the scaled average revenue of firms B and A for the 16 possible price pairs that can emerge from their interaction. In every rectangle, the number before the comma is firm B's revenue, while the number after the comma is firm A's. For example, the bottom-left cell indicates that both firms achieved on average a revenue of 1 when both set the price 29.9; the square immediately to its right shows that firm A's revenue was 0.63 and firm B's 1.39 when price A was 34.9 and price B was 29.9.

An exhaustive game-theoretic analysis of the duopoly is beyond the scope. Still, there are a few basic conclusions about the market structure that one can draw from analyzing the simulation outcomes. For example, there is only one Nash equilibrium in pure strategies: to set the pair $(29.9, 29.9)$. Setting 29.9 dominates all other prices for both firms. This means that to firm A – regardless of what firm B does – setting the price to 29.9 is the best action. These characteristics are, as should be clear, unknown to the firms at the beginning of the simulation. The firms even ignore the fact that they are in a competitive environment. None of the firms take into consideration the prices set by their competitors when setting their prices. Only indirectly does the price of firm A affect the actions of firm B and only because the price of B influences the demand experienced by firm A. In this competitive setting, firms have ultimately a wrong model of the world. Firms do not take into consideration that their demand function is conditional on the competitor's price. The correct model of the world would need to account for the fact the demand for a firm's product depends not only on its price but also on the price of its competitor since that's the way we designed the market.

These simulations illustrate how model misspecification can weaken the advantage of a learning algorithm. The agents estimate demand without accounting for competitors' prices, even though those prices determine customer choices. In this setting, Thompson-sampling strategies lose much of their advantage, and a simple greedy rule can perform well. The result highlights a practical distinction: a firm may need to learn not only demand parameters, but also which variables belong in its demand model.

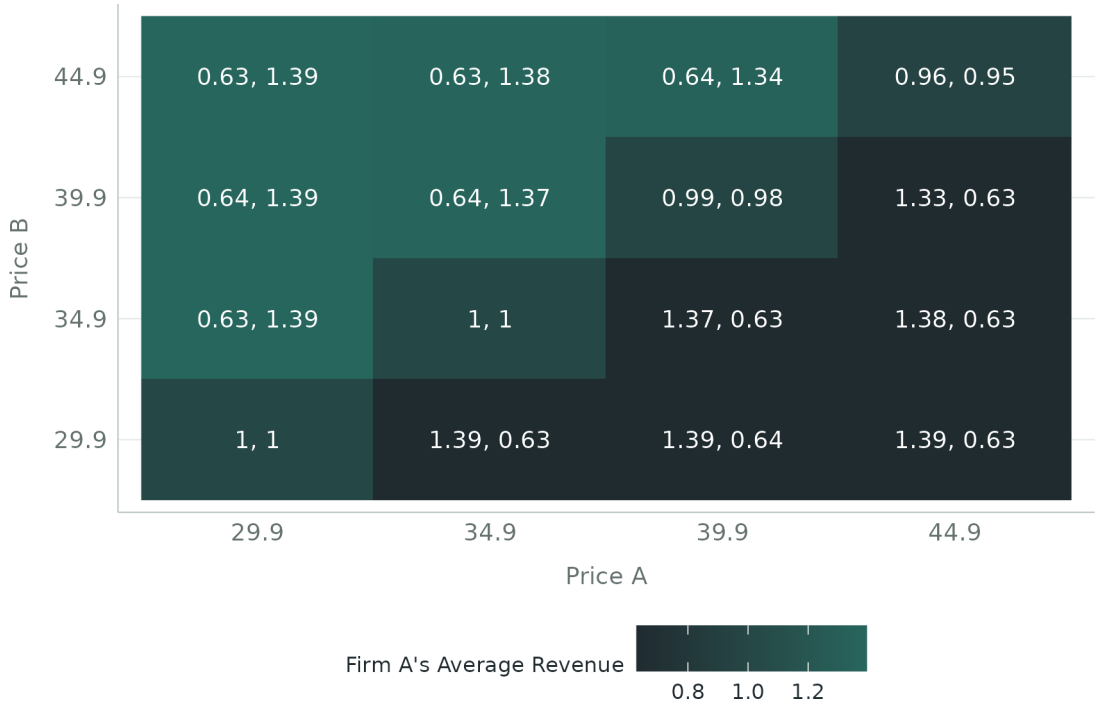


Figure 13: Scaled average revenue of firm A for different price combinations

8. Literature Review

Dynamic pricing combines decisions about current revenue with questions about future demand and available inventory. Den Boer 2015 survey the development of this literature through more than 350 papers. M. Chen and Z.-L. Chen 2015 focus on three related areas: multiple products, competition, and limited demand information. Aviv and Vulcano 2012 provide an accessible introduction to the underlying models. The discussion below focuses on work relevant to learning demand under inventory constraints.

Gallego and Van Ryzin 1994 study a firm selling perishable goods under stochastic demand before a deadline. Unlike the discrete-period models used in this thesis, their model operates in continuous time. They derive a closed-form optimal policy for exponential demand and show that a fixed-price policy can approach optimal performance asymptotically. This result concerns a full-information setting, in which the seller does not need to learn demand.

Aviv and Pazgal 2002 introduce learning into a continuous-time model with a finite selling season. Customers arrive according to a Poisson process with an unknown rate, and each customer purchases with probability $exp(-p)$. The seller faces both random arrivals and uncertainty about the arrival-rate parameter. A Gamma distribution represents beliefs about that parameter and is updated as observations arrive. Because the optimal policy generally lacks an explicit solution, the authors compare three heuristics numerically: fixed pricing, certainty-equivalent pricing, and a naive policy that neither accounts for arrival-rate uncertainty nor learns about it.

Kleinberg and Leighton 2003 use regret to evaluate dynamic pricing algorithms analytically. Regret measures the expected revenue forgone relative to an optimal clairvoyant seller. Customers arrive with a willingness to pay, observe a price, and decide whether to purchase. The authors derive upper and lower regret bounds for three settings: identical buyer valuations, valuations drawn from a fixed distribution, and buyer valuations without a distributional assumption.

Boer and Zwart 2014 propose controlled variance pricing (CVP). For a broad class of demand models, they establish strong consistency: the policy eventually learns the optimal price.

Besbes and Zeevi 2012 consider demand learning with limited inventory. Their "blind" policies divide the selling season into an exploration phase and an exploitation phase. Observations at different prices provide estimates of demand rates, which feed a linear optimization problem used to select prices for exploitation. The authors show that these policies asymptotically approach the performance of a clairvoyant seller.

Upper confidence bound (UCB) methods offer another approach to exploration. They select an arm using an optimistic estimate of its reward, based on a confidence bound. Standard bandit formulations do not directly account for scarce inventory. Badanidiyuru, Kleinberg, and Slivkins 2018 extend the framework to resource constraints in the "Bandits with Knapsacks" problem.

Keskin and Zeevi 2014 study a monopolist selling over T periods. When the seller initially knows none of the demand parameters, the smallest achievable revenue loss relative to a

clairvoyant seller is of order $\sqrt{T}$. Knowing expected demand at one incumbent price changes the attainable rate to order $\log T$. The comparison shows how a small amount of prior demand information can affect learning performance.

Besbes and Zeevi 2015 examine model misspecification: the seller fits a parametric linear model even though actual demand may have a different form. Their analysis shows that an overly restrictive model need not produce a large revenue loss. This distinction between accurate demand estimation and effective pricing is relevant to the competitive simulations in this thesis.

Harrison, Keskin, and Zeevi 2012 consider a seller facing one potential customer per period, with an unknown probability of purchase. Under the myopic Bayesian policy (MBP), learning occurs passively as the seller chooses prices based on current beliefs. The authors show that a constrained version of MBP can keep revenue loss relative to a clairvoyant seller bounded by a constant as the selling season grows.

Araman and Caldentey 2009 study nonperishable goods with limited inventory over a potentially infinite selling horizon. Future revenue is discounted, making the timing of sales relevant even without a fixed expiration date. The model also allows changes in assortment. In one setting, the retailer must sell the existing stock before changing the assortment; in another, the change can occur before inventory is exhausted. The authors characterize properties of the optimal policy and evaluate computationally feasible approximations through numerical simulations.

Simulation infrastructure is also important for studying competition. Boissier et al. 2017 introduce Price Wars, an open-source, microservice-based framework for simulating pricing agents in online marketplaces. Its architecture is designed to support large markets with many agents. The source code is available at https://github.com/hpi-epic/pricewars.

Reinforcement learning provides further approaches to dynamic pricing. Kastius and Schlosser 2021 evaluate Deep Q-Networks and Soft Actor Critic in duopoly and oligopoly settings. In some duopolies, dynamic programming provides an optimal-policy benchmark. Under certain conditions, the learning agents develop collusive behavior without direct communication.

Mueller, Syrgkanis, and Taddy 2018 show that, under specified conditions, revenue maximization in a multiproduct shop can be formulated as online bandit convex optimization. They evaluate their method under stationary demand, demand shocks, drifting demand, and misspecification. Their results suggest applications to high-dimensional pricing problems involving more than 1,000 products.

Together, these studies distinguish several challenges that a pricing algorithm may face: uncertainty about demand parameters, limited inventory, changing demand, competition, and model misspecification. This thesis uses simulations to examine how selected Thompson-sampling strategies respond when inventory constraints and competitive demand are combined.

9. Conclusions

This thesis examines how a seller can learn demand while pricing a product with limited inventory. The analysis first isolates inventory allocation and demand learning, then combines them and introduces competition.

When demand is known and deterministic, pricing allocates inventory across the selling season. Under the assumptions used here, an optimal allocation equalizes marginal revenue across periods. Unknown or stochastic demand makes this allocation problem more difficult because the seller must make decisions before learning the relevant demand parameters.

Demand learning creates an exploration-exploitation trade-off. The multi-armed bandit framework makes this trade-off explicit: each price is an arm, and observations reveal information about its expected revenue. TS-Fixed and TS-Update combine Thompson sampling with constrained optimization to account for limited inventory. TS-Fixed uses a constant inventory allowance, whereas TS-Update adjusts the allowance as inventory and remaining selling time change.

The simulations reproduce the main numerical findings of Ferreira, Simchi-Levi, and Wang 2018. Inventory-aware Thompson-sampling strategies perform well in the noncompetitive setting, and their performance improves as the selling season lengthens. Ignoring inventory constraints can sacrifice revenue even when demand is learned accurately.

The competitive simulations qualify this result. Demand is constructed from individual customers' willingness to pay, allowing a firm's sales to depend on competitors' prices. The agents continue to learn as though demand depended only on their own prices. Under this misspecification, the advantage of TS-Fixed and TS-Update over Greedy and ε-Greedy narrows substantially, while Greedy performs better than in the noncompetitive simulations.

These results emphasize the importance of the model underlying a learning algorithm. Learning its parameters accurately is not enough if the model omits an important determinant of demand. The findings concern the simulated markets considered here; they do not establish a universal ranking of pricing strategies across competitive settings.

References

Araman, Victor F and René Caldentey (2009). "Dynamic pricing for nonperishable products with demand learning". In: *Operations research* 57.5, pp. 1169–1188.

Aviv, Yossi and Amit Pazgal (2002). "Pricing of short life-cycle products through active learning". In: *Under revision for Management Science*, pp. 1–32.

Aviv, Yossi and Gustavo Vulcano (2012). "Dynamic list pricing". In: *The Oxford handbook of pricing management*.

Badanidiyuru, Ashwinkumar, Robert Kleinberg, and Aleksandrs Slivkins (2018). "Bandits with knapsacks". In: *Journal of the ACM* 65.3, 13:1–13:55. DOI: 10.1145/3164539.

Bertsimas, Dimitris and Georgia Perakis (2006). "Dynamic pricing: A learning approach". In: *Mathematical and computational models for congestion charging*. Springer, pp. 45–79.

Besbes, Omar and Assaf Zeevi (2012). "Blind network revenue management". In: *Operations Research* 60.6, pp. 1537–1550. DOI: 10.1287/opre.1120.1103.

– (2015). "On the (surprising) sufficiency of linear models for dynamic pricing with demand learning". In: *Management Science* 61.4, pp. 723–739.

Boer, Arnoud V den and Bert Zwart (2014). "Simultaneously learning and optimizing using controlled variance pricing". In: *Management science* 60.3, pp. 770–783.

Boissier, Martin et al. (2017). "Data-driven repricing strategies in competitive markets: An interactive simulation platform". In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pp. 355–357.

Chen, Ming and Zhi-Long Chen (2015). "Recent developments in dynamic pricing research: multiple products, competition, and limited demand information". In: *Production and Operations Management* 24.5, pp. 704–731.

Den Boer, Arnoud V (2015). "Dynamic pricing and learning: historical origins, current research, and new directions". In: *Surveys in operations research and management science* 20.1, pp. 1–18.

Ferreira, Kris Johnson, David Simchi-Levi, and He Wang (2018). "Online network revenue management using thompson sampling". In: *Operations research* 66.6, pp. 1586–1602.

Gallego, Guillermo and Garrett Van Ryzin (1994). "Optimal dynamic pricing of inventories with stochastic demand over finite horizons". In: *Management science* 40.8, pp. 999–1020.

Geer, Ruben van de et al. (2019). "Dynamic pricing and learning with competition: insights from the dynamic pricing challenge at the 2017 INFORMS RM & pricing conference". In: *Journal of Revenue and Pricing Management* 18.3, pp. 185–203.

Harris, Charles R. et al. (Sept. 2020). "Array programming with NumPy". In: *Nature* 585.7825, pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

Harrison, J Michael, N Bora Keskin, and Assaf Zeevi (2012). "Bayesian dynamic pricing policies: Learning and earning under a binary prior distribution". In: *Management Science* 58.3, pp. 570–586.

Kastius, Alexander and Rainer Schlosser (2021). "Dynamic pricing under competition using reinforcement learning". In: *Journal of Revenue and Pricing Management*, pp. 1–14.

Keskin, N Bora and Assaf Zeevi (2014). "Dynamic pricing with an unknown demand model: Asymptotically optimal semi-myopic policies". In: *Operations Research* 62.5, pp. 1142–1167.

Kleinberg, Robert and Tom Leighton (2003). "The value of knowing a demand curve: Bounds on regret for online posted-price auctions". In: *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.* IEEE, pp. 594–605.

McElreath, Richard (2020). *Statistical rethinking: A Bayesian course with examples in R and Stan.* CRC press.

Mueller, Jonas, Vasilis Syrgkanis, and Matt Taddy (2018). "Low-rank bandit methods for high-dimensional dynamic pricing". In: *arXiv preprint arXiv:1801.10242.*

Peng, Roger D (2011). "Reproducible research in computational science". In: *Science* 334.6060, pp. 1226–1227.

Reinhart, Carmen M and Kenneth S Rogoff (2010). "Growth in a Time of Debt". In: *American economic review* 100.2, pp. 573–78.

Russo, Daniel et al. (2017). "A tutorial on thompson sampling". In: *arXiv preprint arXiv:1707.02038.*

Talluri, Kalyan T and Garrett Van Ryzin (2004). *The theory and practice of revenue management.* Vol. 1. Springer.

Virtanen, Pauli et al. (2020). "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17, pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

A. Results Replication

> *Replication is the ultimate standard by which scientific claims are judged. With replication, independent investigators address a scientific hypothesis and build up evidence for or against it. The scientific community's "culture of replication" has served to quickly weed out spurious claims and enforce on the community a disciplined approach to scientific discovery. However, with computational science and the corresponding collection of large and complex data sets the notion of replication can be murkier. [...] Many studies — for example, in climate science — require computing power that may not be available to all researchers.* (Peng 2011).

In the quote above, Peng 2011 is making the case for reproducible research. In a nutshell, reproducible research consists of not only publishing the research results but also making available the code and the data used to get to them. In a world with various practical constraints, reproducible research is often the best possible alternative to the full replication of a given study. Full replication involves independently gathering new data, analyzing it, and either corroborating or rejecting the conclusions of the study being fully replicated. This, of course, is often unattainable. In those situations, reproducible research comes into play. By publishing code and data along with results, authors can submit their work to greater scientific scrutiny, which gives more credibility to the scientific process. In fact, significant mistakes have been spotted only by inspecting the instruments used for analysis.[12] Finding those sort of errors is only possible if the authors make their data and code available to the public.

Basing a paper on simulations results come with considerable drawbacks. The simulated scenarios can be often unrealistic and many aspects of reality might be ignored for the sake of simplicity, so that the simulations can be computed by a regular computer. Simulation-based results do, however, allow for full replication with relative ease. Anyone willing to replicate the results of the present thesis need not go to the field and gather data herself. On the contrary, one can just regenerate the data with the simulation code and analyze the results. Thus, any mistake in the data generating process can be easily spotted, which brings much more transparency to the research. In that spirit, we have made all the code - both for generating the data and for subsequent analysis - available in a single GitHub repository: `https://github.com/denismaciel/uni-master-thesis`. The repository is the canonical source for the thesis source, `dynpric`, the simulation entrypoints, and the generated-figure pipeline. Earlier standalone repositories for `dynpric` and the simulation code are historical archives. Any interested reader is encouraged to replicate our results, build upon them, and eventually correct mistakes.

The code is organized as a monorepo:

- `src/dynpric` is a Python package that provides the primitive objects used to build the market simulations.

[12] See the controversy arousing from Reinhart and Rogoff 2010's mistake where the authors mistakenly excluded some rows from the computation of their results.

- `src/simulations` contains the actual code for running the resumable simulations and analyzing simulation outputs.
- `src/figures` contains figure generators that do not themselves define resumable simulation entrypoints.

A.1. dynpric

`dynpric` is based mostly on Python's standard library. We do, however, use numpy (Harris et al. 2020) and scipy (Virtanen et al. 2020) for numerical optimizations.

To run the code, use the repository's reproducible environment:

`uv run python -m simulations -help`

We list below the types and the interfaces used to build a simulation trial. From the type definitions, one notes that the overall structure of a period in a sales season is fairly straightforward. The complexity is delegated to the implementation of the different firms and demands. Looking closer at the types, a sales season is simulated by the `simulate_market` function. The function signature makes clear that a market is composed of firms, a demand function, and a logger callback used to extract structured observations from each period. In every period of the sales season, the prices set by the firms are collected via getting the attribute method `Firm.price`. Those prices are passed to `Demand.allocate`, which returns the demanded quantity for each firm. Finally, the method `Firm.observe_market` is called at the end of the period so that the firms can update their prices based on what has happened in the sales season so far. The complexity lies in the implementation of how the firms set the prices and, given those prices, how demand is allocated across firms.

```
def simulate_market(
    n_periods: int,
    firms: list[Firm],
    demand: Demand,
    logger: Callable[
        [Firm, PricesSet, DemandRealized],
        dict[str, object] | None,
    ],
) -> tuple[list[Log], History]: ...

class Firm(Protocol):
    name: str

    @property
    def price(self) -> float: ...

    def observe_market(self, history: History) -> None: ...

class Demand(Protocol):
    def allocate(self, prices_set: PricesSet) -> DemandRealized: ...
```

```
DemandRealized = Mapping[Firm, Quantity]

PricesSet = Mapping[Firm, Price]

class Period(NamedTuple):
    prices: PricesSet
    demand: DemandRealized

LogValue = str | int | float | object
LogInfo = dict[str, LogValue]
Log = dict[Firm, LogInfo]

History = list[Period]
TrialResults = tuple[list[Log], History]
```

A.2. How to replicate results

Detailed instructions on how to replicate each of the figures in this work can be found in the repository `README.md`.

Declaration of Authorship

I hereby confirm that I have authored this Master's thesis independently and without use of others than the indicated sources. All passages which are literally or in general matter taken out of publications or other sources are marked as such.

Berlin, October 27, 2021

Denis Augusto Pinto Maciel