# Primary Market Dynamic Pricing for Sports Tickets: Theory and Application

by

Spencer David Hylen

S.B. in Electrical Engineering and Computer Science and in Mathematical Economics, Massachusetts Institute of Technology, 2022

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

Author . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Department of Electrical Engineering and Computer Science
May 6, 2022

Certified by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Anette Hosoi
Associate Dean, MIT School of Engineering
Thesis Supervisor

Accepted by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

# Primary Market Dynamic Pricing for Sports Tickets: Theory and Application

by

Spencer David Hylen

Submitted to the Department of Electrical Engineering and Computer Science
on May 6, 2022, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

## Abstract

In professional sports, fan attendance at games is one of the largest drivers of revenue and a significant focus for all sports teams. While sports tickets can benefit from dynamic pricing techniques developed for airlines or hotels, they have constraints related to limited inventory, season tickets, and the attendance-revenue trade-off that make them unique and difficult to price with existing dynamic pricing systems. In this work, we present a novel dynamic pricing strategy for sports tickets, developed with an engineering approach alongside primary market ticket data provided by the San Antonio Spurs. Using dynamic programming and matrix completion for modeling customer demand, we provide optimal price recommendations for all time periods before a game. We use these recommendations, along with information on current prices, to provide a suite of ticket price optimization and analysis tools. These tools are led by our Manual Price Adjustment Guide, which combines our optimal price recommendations with ticketing analyst expertise for a hybrid solution designed to improve ticket pricing and increase team revenue.

Thesis Supervisor: Anette Hosoi
Title: Associate Dean, MIT School of Engineering

# Acknowledgments

There are so many people to thank who have helped me along this journey that I will likely miss one. If you are reading this, you are likely one of those people who have accompanied me along some point in this journey. So (for the sake of completeness): thank you. For your support, guidance, and encouragement. This journey would have been much harder and less enjoyable without you.

I must start by thanking my advisor Peko Hosoi, who has been nothing short of phenomenal throughout this process and my entire time in the Sports Lab. She always knew the right question to ask, person to consult, or way to approach a new (and always challenging!) problem. Her unwavering support and confidence in me has been an invaluable asset throughout this process. I am forever grateful to her and Christina Chase for welcoming me into the Sports Lab and embarking on this journey with me.

I also owe a special thank you to my Sports Lab colleagues, especially Kevin Lyons, Dalton Jones, and Ferran Vidal-Codina. Thank you all for your suggestions and support. You have been a true pleasure to work alongside.

To my mom Lynn, dad Chris, and brother Trevor, thank you for loving, supporting, and encouraging me in all my endeavors (academic or otherwise). I am truly blessed to have such a loving family who have taken such good care of me, have taught me the importance of hard work and a good education, and have always been there to pick me up when I fall down.

To my teachers at Viewpoint, Calvary Christian, and Franklin, thank you for nurturing my curiosity and instilling in me a lifelong love of learning. I hope I've made you proud.

To Andrew, Aidan, and all my friends who I have made at MIT, in the dorms, on teams, in clubs, and elsewhere, thank you for encouraging me and motivating me to be a better student and person.

Lastly, to Ben, Charlie, Hayden, Julian, Keithen, Mark, Mason D., Mason O., Sam, Tyler, all my fraternity brothers at DKE, and my girlfriend Sarah, thank you for loving me, supporting me, and (most importantly) teaching me how to have fun. Better late than never.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background

In the world of professional sports, fan attendance at games is the lifeblood of the team. An arena full of passionate fans creates an exciting environment that inspires players, discourages opponents, and (most importantly) brings in revenue. The money generated by ticket sales is one of the largest drivers of revenue across all major sports leagues, making ticketing a significant focus for all sports organizations.

The importance of ticketing revenue for teams can be seen in Figure 1-1, which visualizes the percentage of total revenue derived from ticketing for each of the major American professional sports leagues. All leagues obtain at least 15% of their revenues from ticketing, with some as high as 36%, making ticketing a major revenue driver for all sports teams. Moreover, since seat costs are often fixed, every additional dollar earned from ticketing usually translates to an additional dollar of profit for a team, making ticket prices important drivers of team profitability in addition to revenue.

Unlike other sources of team revenue and profitability, however, team ticket sales are affected by a limited supply. While a team can always make and sell one additional piece of merchandise, for example, it can only sell as many tickets as it has seats in its stadium. This challenge, coupled with the importance of ticketing for the financial success of the team, makes finding optimal ticket prices a key focus and major need for sports organizations.

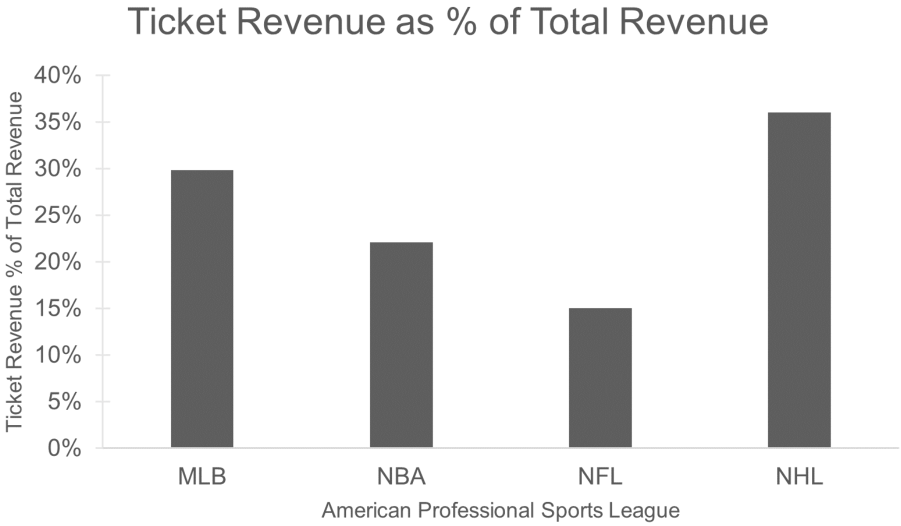


Figure 1-1: Ticket revenue as a percentage of total revenue for major American professional sports leagues. All data is from the most recent pre-Covid season [17, 18, 16].

In this work, in collaboration with the San Antonio Spurs (a professional team in the National Basketball Association), we propose a novel dynamic pricing strategy that can address this area of need for all sports organizations. Our strategy combines automated predictive pricing with human expertise to provide a solution that is adaptive, scalable, and uniquely suited to succeed in the sports organization environment.

## 1.2 Dynamic Pricing

While a team's limited inventory makes it difficult to increase revenue by selling more tickets, it can actually prove beneficial by providing the team with much more flexibility in how it prices its tickets. There is usually much more demand for tickets than there are seats available, allowing teams to shift ticket prices with demand. Prices for more popular games may be adjusted up, as fans are more interested and willing to pay more to see the game, while less popular games may have prices adjusted down since there is less interest in the game. How much a fan, or a customer, is willing to pay for a ticket, when impacted by the popularity of the game, is called the customer's "willingness to pay". In general, companies hope to raise prices when there is high demand and customers have a higher willingness to pay, and lower them when there is low demand and a lower willingness to pay.

This adjustment of prices to account for current demand conditions and customer willingness to pay, when done automatically, is known as *dynamic pricing*. It first rose to prominence in the 1980s when regulations were dropped for the airline industry, which quickly faced a similar problem to that of sports teams: given a fixed number of seats on a certain flight, how can an airline best maximize the revenue brought in by those tickets? Initially, these ticket prices were adjusted to account for factors like the destination, time, and season of the flight. Around the holidays, for example, there is likely higher demand for those seats, so the airline is able to charge higher prices. A midday, regional flight during an off-peak season, however, would likely fetch a lower price.

From the airline industry, the application of dynamic pricing has expanded to other areas of travel and entertainment. Theater patrons headed to a play, for example, are likely willing to pay different prices based on the proximity of their seat to the stage and the distance from the center of the venue to their seat. These patrons are also likely willing to pay different prices for different plays. Theater owners capitalize on this by charging variable prices for specific sections of the theater, as well as prices that can vary with the popularity of the show. In sports, with a group of rotating opponents and a stadium with many different views of the playing surface, the same customer preferences apply. As a result, similar types of dynamic pricing techniques that are used for airlines and entertainment can be applied to sports tickets to help teams set prices that bring in the most revenue for each game.

## 1.3 Sports Industry Constraints

Despite some similarities, the sports ticketing problem has a few complications that differentiate it from other dynamic pricing applications: lower seat quantities, season tickets, and the attendance-revenue trade-off. Together, these constraints create a dynamic pricing problem unique to sports.

### 1.3.1 Limited Seat Inventory

Compared to airlines and other applications of dynamic pricing, sports teams face a vastly different ticketing problem. As an example, United Airlines flew approximately 162 million passengers in 2019 [19]. For contrast, the average NBA team (with a stadium of about 18,000 seats and playing 41 games per year) would have around 0.75 million attendees per year [21]. This vast difference in volume creates two information-related issues for the team.

The main information-related constraint in dynamic pricing for sports comes from a lack of data. Since NBA teams (and sports teams in general) sell significantly less seats than other users of dynamic pricing strategies, they have significantly less information about customer willingness to pay and the demand for each game. This information is essential for selecting optimal prices, creating a difficult challenge for teams.

The second information-related constraint is the importance of each ticket and avoidance of experimentation. Because sports teams have less tickets to sell, each ticket is much more valuable, making them less likely to experiment with ticket prices. Without offering tickets at different price points, teams cannot gain the necessary information on different customers' willingness to pay. This drives the lack of information available to teams and further complicates dynamic pricing development, as it imposes a significant barrier to entry through the additional cost of gathering information. Together, these limits and costs related to collecting more information create a difficult environment for developing and implementing dynamic pricing solutions in sports.

### 1.3.2 Season Tickets

In addition to significantly less seats to sell, sports teams also have season tickets that reduce their capacity even further. A season ticket entitles the holder to attend all (or a majority) of the games for a sports team, often for a discounted price compared to buying all of the same tickets individually. For fans, this offers a cheaper way to see their team play more often. For teams, this offers a way to guarantee revenue and minimize unsold ticket risk. However, this also further reduces the number of seats the team is able to price dynamically. Furthermore, to not upset fans, teams will use the price of season tickets as the lowest price for which they will sell a ticket throughout the season. This places additional restrictions on team pricing policy, increasing the difficulty of developing dynamic pricing strategies.

### 1.3.3 Attendance-Revenue Trade-off

The final complication for sports teams is the attendance-revenue trade-off, which is a significant philosophical debate for teams. While maximizing ticket revenue is valuable, there may be incremental sales from merchandise and concessions that a team can earn by filling seats at a reduced price. Moreover, an empty stadium is likely to leave a negative impression on fans and players, hurting the team's on-court success and indirectly resulting in lower revenue for the team. Together, it is unclear if these factors make selling a seat for any price more valuable than maximizing the price for which the ticket is sold. Therefore, a team must consider whether they should maximize revenue at the expense of a more empty stadium or try to balance the two factors. This type of balance would require consideration of many more objective functions than a traditional revenue maximization, further complicating the dynamic pricing problem.

## 1.4 Contributions

In addition to the constraints surrounding dynamic pricing for sports, privacy concerns and a lack of public data have limited development of dynamic pricing technologies in sports. Here, we discuss the primary novel contributions of this work, which address some of these challenges and expand on the literature[1] through new

[1] Our novel contributions and relation to the literature is discussed further in Chapter 2.

approaches and greater access to data.

### 1.4.1 Sports Application Focus

As mentioned previously, the invention and application of dynamic pricing is not unique to sports. With the rise of e-commerce and additional prevalence of data, significant work is being done to incorporate dynamic pricing into most areas of modern commerce. However, these applications are often not directly transferable to sports. Generally, they do not operate with the same assumptions underlying sports ticketing, including:

1. Expiration of Inventory: Unsold tickets to a game expire once that game occurs, as they cannot be rolled over and sold for a later game. This is generally untrue for most other products sold, such as apparel or other physical goods.

2. Heterogeneity of Goods: Many products are homogeneous or have only a few different variations. Seats in a stadium often have enormous variety, based on the changes in viewing angle and proximity to the playing surface. Even when considering nearby seats to be identical products, the number of product variations far exceeds that in other applications.

3. Lack of Price Constraints: For almost all products, sellers are free to set their prices at any level. Prices can even be set below the item's cost if the seller is trying to clear inventory and willing to take a loss. For sports teams, however, setting individual ticket prices below season ticket prices is untenable, creating a price floor that constrains pricing and makes other pricing work inapplicable.

The absence of any one (or multiple) of these assumptions generally prevents the application of other dynamic pricing work to the sports ticketing problem. By focusing specifically on sports ticketing, our system addresses an unmet area of need for dynamically pricing sports tickets and other similar products.

### 1.4.2 Engineering Approach

Many current approaches to dynamic pricing in sports and other industries seek to identify causal factors that drive consumer demand. This understanding can be helpful for developing a dynamic pricing system, as it allows teams to measure and react to variables (such as opponent, giveaways, day of the week, etc.) that have an impact on prices. However, these approaches often fall short of providing a system that can adapt prices to changes in key variables. This limits the usefulness of these approaches for sports teams, who often do not have the resources to comb several key variables daily to adjust prices.

Our novel approach borrows from engineering, prioritizing optimal price recommendations rather than extracting how much each causal factor drives ticket prices. We provide an end-to-end ticket pricing solution that can be deployed immediately and does not require significant oversight. Moreover, our pricing tool enables the ability to fold in existing ticket analyst expertise, helping to limit organizational disruption while still improving pricing. Driven by our engineering approach, each these features provide meaningful new functionality that differs from existing methods.

### 1.4.3 Primary Market Application

Most work on dynamic pricing for sports tickets specifically has been applied to secondary markets such as StubHub or Ticketmaster, where data is more easily accessible. While these works can provide valuable insights on pricing tickets, they are not immediately applicable to tickets sold directly from teams (referred to as the "primary market"). This is because of the significant differences in inventory and the sparsity of tickets available for teams and resellers. Moreover, many underlying assumptions that drive these secondary market pricing systems are different in the primary market, making those analyses and systems irrelevant for teams (who, by definition, sell on the primary market).

In this work, we develop a dynamic pricing system specifically designed for internal team use on the primary market. This novel system is applicable across teams and leagues, and it can help improve pricing for any sports organization.

### 1.4.4 Primary Market Sales Data

The proprietary nature of the data has generally made it difficult to obtain sales data from team direct sales to customers (also known as “primary market” sales data). Without access to this data, it is nearly impossible to prescribe a pricing strategy for a team. Moreover, the works that have had access to this primary market data have focused on the types of causal analyses discussed in Section 1.4.2. Our access to and use of this primary market data provides a novel contribution to the sports ticketing dynamic pricing literature.

## 1.5 Outline

We now provide a brief overview of the remainder of this work.

This first chapter, the Introduction, has focused on contextualizing dynamic pricing in sports (including pricing challenges unique to the sports industry) and describing how this work contributes to the field.

The second chapter, on Related Work, details the relevant literature that has been produced on dynamic pricing in sports. We also describe the work done by professional organizations in this area, and cover other potentially relevant work in dynamic pricing.

The third chapter, Dynamic Price Planning, describes how we use dynamic programming to generate plans for teams that prescribe which ticket prices to set for all time periods before a game (our “optimal price plans”). We then motivate the need for a way to understand customer demand.

The fourth chapter, Customer Demand Modeling, covers how we infer customer demand from historical transactions. We detail the two types of demand inference used (continuous and discrete), describe the shortcomings of continuous inference, and cover the creation and testing of our Demand Matrices.

The fifth chapter, Applied Dynamic Price Planning, discusses how we constructed and made adjustments to our system to deploy it for the Spurs. We also describe the design and implementation of the Manual Price Adjustment Guide (our main output) and the Attendance Projection Sheet, the two main tools in our dynamic pricing toolkit.

The sixth and final chapter, our Conclusion, summarizes this work and provides a few key insights. We also describe a few potential avenues for future work to explore.

# Chapter 2

# Related Work

While there are a myriad of challenges that make sports dynamic pricing different from other applications, there have been numerous attempts to develop dynamic pricing strategies for teams. In this section, we start by highlighting existing commercial solutions that offer dynamic pricing for sports teams. We then consider academic solutions that have been developed and proposed for dynamic pricing in sports, as well as other applicable research. Throughout the chapter, we work to demonstrate the novel aspects of our work and highlight areas where it differs from existing work.

## 2.1 Commercial Solutions

Many of the proposed solutions that exist for dynamic pricing in sports have been commercialized by a number of private firms that fall into two distinct categories. Since 2012, firms like Digonex and Qcue have provided dynamic pricing information to sports teams and other organizations. However, these firms never actually possess or sell tickets. This differs from brokers like Dynasty Sports and Entertainment (now Logitix), who provide teams with no-risk revenue by purchasing large blocks of tickets and selling them on third-party exchanges. Often, these brokers will still work with the team to share pricing updates and help the team price their remaining tickets (like information firms Digonex and Qcue). In all cases, these firms are tight-lipped about how they execute their pricing strategies. From past interviews, we find that variables considered include weather, historical sales trends, current standings, star player presence, day of the week, and secondary market demand, which includes tickets resold by sites such as StubHub or Ticketmaster. Claims made by these services suggest that commercial models provide a 10-20% increase in revenue. While our approach does not incorporate secondary market data, we hope that our system will enable teams to obtain similar success to these models with a reduction in complexity.

## 2.2 Academic Solutions

Existing approaches in literature vary widely based on the problem considered and data available. While we will highlight specific examples we find especially relevant, we refer interested readers to den Boer (2015) for a comprehensive literature survey on dynamic pricing.

### 2.2.1 Ticket Pricing for Sporting Events

The main consideration in our work is ticket pricing for sporting events, which has been studied in several papers that focus on the same primary market problem we hope to address [4, 3, 6, 12, 9, 8]. However, with the exception of Jiaqi Xu et al. [8], none of these works use primary market sales from an actual team. The balance use either small, hand-gathered public datasets or take a theoretical approach to offer a managerial perspective. These papers often rely on econometric modeling or other ordinary least squares (OLS) regression fitting in order to learn consumer demand functions, which is helpful for identifying causal factors of price movement but can be limited in predictive power. As only the second work to utilize primary market sales data, our contribution has more predictive power and requires less assumptions than the work in Jiaqi Xu et al. [8].

### 2.2.2 Secondary Market Pricing

Other work in secondary markets has had more predictive success [22, 20], highlighted by Alley et al. [1]. Data for these markets is much easier to gather given their public-facing nature or has been made more readily available as work with a secondary market provider [1], allowing for development of more robust methods using machine learning and more traditional econometric models [22, 20]. However, these works focus on the problem of selling a limited number of tickets on a secondary market, without the constraints imposed on the team selling on the primary market. As a result, the applicability to our work is limited.

### 2.2.3 Other Relevant Dynamic Pricing Work

When considering the problem independent of sports, we turn to literature in multi-product dynamic pricing. Several pricing tools have been developed for retail and travel [15, 14, 11], but the most recent and notable works are Caro and Gallien [5] and Ferreira et al. [7], which focus on retail. Both investigate instances of strictly decreasing prices through markdowns and sales (respectively), which differs meaningfully from our work here where we have flexibility to adjust prices in either direction. In all cases, the current literature does not address or is not applicable to the problem facing sports teams like the Spurs, which we will address in this work.

# Chapter 3

# Dynamic Price Planning

In this chapter, we will describe how we create our pricing plans, which are one piece of our dynamic pricing output. In these pricing plans, we provide an optimal price level for all time periods leading up to a game. We create these plans using dynamic programming for planning and an understanding of customer demand to infer optimal prices. In this chapter, we focus on the dynamic programming technique. We start with an overview of dynamic programming, and we describe how it can be easily applied to the ticket pricing problem. We then discuss how we create our specific price plans using dynamic programming, as well as the implementation of this process. We leave specific application and validation details for Chapter 5, where we provide those details for this dynamic program and our customer demand modeling.

## 3.1 Dynamic Programming for Pricing

In this section, we give an overview of dynamic programming, including the optimal substructure and finite subproblem criteria for a dynamic programming problem. We then discuss how the ticket pricing problem is a natural candidate to be optimally solved with dynamic programming.

### 3.1.1 Dynamic Programming Overview

Dynamic programming is an optimization technique that finds solutions to discrete problems by breaking them into simpler subproblems, each of which can be more easily solved. Recursively deconstructing the main problem this way and then reconstructing the subproblem solutions allows us to solve complicated problems more quickly and optimally than other techniques.

A common problem solved with dynamic programming is the knapsack problem. Given a set of items, each with a specific weight and value, how can we choose a subset of those items to maximize their value while keeping their weight under a predetermined limit? This is known as the "knapsack" problem because of the comparison to filling a knapsack or other piece of luggage with the most valuable collectibles or other merchandise.

Dynamic programming can be applied to the knapsack problem by considering the different subproblems created by selecting each item for inclusion in the knapsack. For example, given a set of three items, $X_1$, $X_2$, $X_3$, we can start by including $X_1$ and then looking for the optimal knapsack value we can obtain using the remaining space and remaining items ($X_2$ and $X_3$). This subproblem creation is done through a *transition function*, which is very simple here: we remove the selected item and subtract its weight from our total weight available, so that we only consider filling the remaining weight with the available items. We can then compare the optimal value from this assumption, where we include item $X_1$, to the optimal value where we assume inclusion of either $X_2$ or $X_3$. Comparing these three optimal values, we can choose whichever solution provides the overall optimal value. The ability to return an optimal overall solution from the optimal subproblem solutions in this way is known as *optimal substructure*, and it is an important attribute of discrete optimization problems that can be solved with dynamic programming.

In addition to having optimal substructure, this version of the knapsack problem is *finite*[1]. This means that there are only so many different combinations of objects

[1] We note that problems can often become intractable even when the number of subproblems is finite as a result of memory or processing constraints. Because problem structures and dynamic

that could fit into the bag. Therefore, once we have found the optimal solution to each possible recursive subproblem, we have found the optimal solution to the entire problem, since we have evaluated all possible options. The finiteness of the problem is important here: without a limit on the number of subproblems to try, the dynamic program could look forever for an optimal solution.

The example knapsack problem here can expand to any number of items and any number of constraints. By assuming an item is included in our solution, then solving the simpler subproblem, we can solve the overall knapsack problem quickly and optimally.

This dynamic programming framework can be applied to any discrete optimization problem, with the two important criteria described in the example above. First, the problem must have optimal substructure. That is, we must be able to find an optimal solution to the overall problem by breaking it down recursively into subproblems (using its transition function) and combining the optimal subproblem solutions. Secondly, the number of subproblems must be finite. Specifically, there must exist a non-infinite maximum number of subproblems we could consider when looking to solve the overall problem. Any discrete optimization problem with these two attributes can be solved with dynamic programming.

### 3.1.2 Pricing Application

When structured appropriately, it is clear to see how the ticketing problem satisfies the optimal substructure and finite subproblem criteria and can therefore be solved by dynamic programming. We start by considering the overall goal of ticketing: we have a given quantity of tickets we can sell, and we want to understand what prices to set for each time period before an upcoming game. While this can be optimized for revenue or fan attendance (or some combination of the two, depending on the organizational approach to the attendance-revenue trade-off discussed in Section 1.3.3), we focus

programming implementations can vary greatly across applications and systems, this specific finite value can vary widely. We discuss some limits specific to our implementation, including a bound on our number of subproblems, in Section 3.1.2.

primarily on maximizing revenue for our optimal solution.

We start by reframing the problem using ticket quantities instead of prices. This is done to help demonstrate the applicability of dynamic programming, which can be shown more easily with the quantity specification. The setup is similar: we have a given quantity of tickets we can sell, and we aim to maximize the revenue from selling those tickets. However, rather than solving for the price for each time period, we solve for the quantity of tickets we want to sell over each time period (while still maximizing revenue). These specifications are made equivalent by a *demand curve*, a concept from microeconomics that converts between prices and quantities[2]. For a given price $p$, the demand curve will tell us how many tickets $q$ we would sell at that price, and it can tell us what price $p$ we should set if we want to sell $q$ tickets. We can see how this specification is applied and fits the dynamic program requirements in Figure 3-1, which visualizes the evolution of the dynamic program process. For each time period (here, each day before the game), we consider all possible quantities of tickets sold. These possibilities range from selling all three of our tickets for a price $p_4$ or selling none of them for price $p_1$ (with both prices calculated from the demand curve). The dynamic program evaluates the revenue collected from each of these options, then repeats the process for each additional time period before the game.

In both cases, the solutions will return the same optimal revenue, as they have the same constraints and are optimizing the same metric. By using the demand curve, we can translate a solution to the quantity-specified problem (which gives us a list of how many tickets we should sell for each time period) into a solution to the price-specified problem (which is a list of prices to set for each time period). For the sake of more clearly demonstrating how the ticketing problem satisfies the dynamic programming criteria, we will use the quantity specification.

With the ticketing problem reframed in terms of quantities, we can now demonstrate how it exhibits optimal substructure. Consider a game that starts at time $t = 0$. We start some $t = T$ time periods before the game, and have at most $q = Q$

[2] Readers interested in more information on the topic of the demand curve and its background in microeconomics can see [10] for more information.

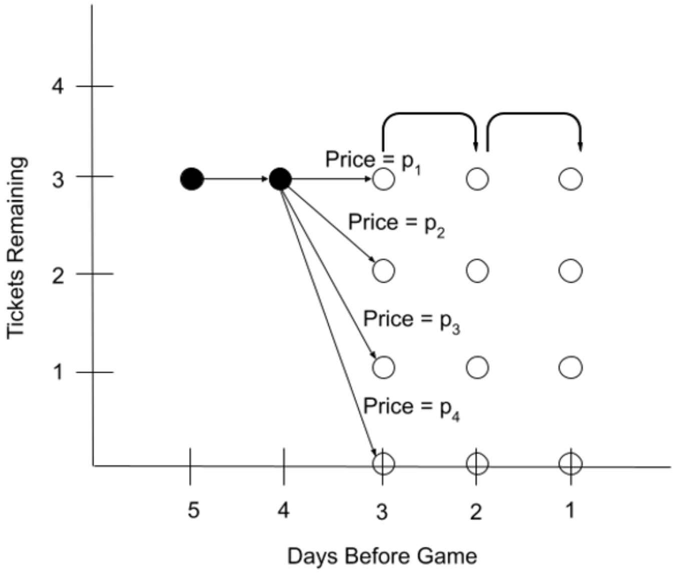


Figure 3-1: Graph depicting the evolution of the dynamic program process for the ticketing problem. At each time period (in this case, days before the game), we consider selling different quantities of tickets, and evaluate the price at which we could sell each quantity.

tickets we can sell. As time passes, we naturally get closer to the game, making $t$ strictly decreasing until $t = 0$ (at which point the game has occurred and we cannot sell any more tickets). Likewise, as we sell more tickets, $q$ strictly decreases until $q = 0$ (at which point we cannot sell any more tickets, as we have no more tickets to sell). This creates a natural subproblem structure: for each problem, we consider what revenue we would collect if we sold $q_s = 0, 1, 2, ..., Q$ tickets over that time period. We create these subproblems with our pricing transition function, which is slightly more complicated than the transition function mentioned above for the knapsack problem. The pricing transition function considers certain drivers of customer willingness to pay (discussed further in Section 4.1.1), including how far we are from the game, to provide an expected price for which we could sell each number of tickets. We build this through our customer demand modeling and price inference, of which the theory and implementation is discussed further in Chapter 4. As we build each of our subproblems with our transition function, we note that they each have $q' = q - q_s$ tickets to sell and $t - 1$ periods to sell them, based on the number of tickets $q_s$ we sold in the current period. As we can see, if we know the optimal revenue over all subproblems (i.e., all of the next time periods leading up to the game), then we can add the optimal revenue for the current time period to get the overall optimal revenue for the game. In this way, the ticketing problem satisfies the optimal substructure criteria for dynamic programming.

We next demonstrate that the subproblems for the ticketing problem are finite. As mentioned above, in the original instance of our problem we start some $t = T$ time periods before the game, and have at most $q = Q$ tickets we can sell. Since we could sell these tickets at any time period before the game, we have $Q + 1$ subproblems for each time period and $T$ time periods, giving us a total of at most $T \times (Q + 1)$ subproblems. In the NBA, most teams update ticket prices at most once per day for a season that is six months long, and the average arena capacity is around 18,000 [21]. Therefore, we have a theoretical upper bound of around 3.2 million subproblems. In our work, we have found that teams will often sell around 30% of their stadium capacity through this method and that most customers purchase tickets within 90 days of the game, giving us a practical upper limit of about 0.5 million subproblems. Since each subproblem can be stored in at most twelve bytes[3], the amount of memory required is less than 100 MB in either case, and the subproblem count for the ticketing problem can be considered finite and small enough to not pose a tractability problem.

Through the discussion above, we have demonstrated that the quantity specification of the ticketing problem satisfies the optimal substructure and finite subproblem criteria for dynamic programming. As the quantity and price specifications of the ticketing problem are equivalent, this demonstrates that both specifications of the ticketing problem satisfy the dynamic programming criteria and can be solved quickly and optimally using dynamic programming.

## 3.2 Price Plan Creation

Now that we have demonstrated that the ticket pricing problem satisfies the dynamic programming criteria, we turn to the specific details of our implementation. Here, we show the actual output of our dynamic program, and we describe how that output can be used by teams to set specific price levels at different time periods. We then describe our algorithm for setting up the dynamic program, which will further motivate the need for a ticket transition function that is addressed in Chapter 4.

### 3.2.1 Price Plan Output

As described at the beginning of this chapter, our price plan output provides an optimal price level for all time periods leading up to a game. More specifically, we segment the remaining days before the game into specific time periods (often referred to as “time buckets”) where we find that ticket sales cluster. For each of these periods, we provide an optimal price recommendation generated by the dynamic program, as well as the number of tickets we expect to sell at this price. We can do this because of

[3] In this storage case, we would have one byte for the time remaining until the game, four bytes for the quantity of tickets remaining, and one byte for a pointer from the previous subproblem. To be conservative and account for overhead and other factors, we have quadrupled this estimate of space.

the interchangeable nature of price and quantity values as described previously, with our understanding of customer demand allowing us to translate between prices and quantities.

A sample visualization of a price plan output can be see in Figure 3-2. Note that there are two lines on the graph: one representing revenue and one representing ticket price. The ticket price is provided as described above: our dynamic program returns an optimal price for each time period, which we plot. The revenue is calculated using this ticket price and the predicted quantity, which we obtain as part of our solution to the quantity-specified ticketing problem. For certain time periods, this revenue will drop to zero, before increasing during the next period. This is an intentional design of our dynamic program and a result of the limited ticket supply described in Section 1.3.1: by looking forward, the program can see that we will be able to sell those tickets for higher revenues later, so it chooses not to sell them during the periods of lower predicted prices (which are shown as zero revenue). In practice, these prices would likely be set to the overall maximum price, so the team can capture maximum revenue in the event a customer chooses to purchase a ticket during one of those periods.

Given the data available to us, we believe these price plans are optimal and will generate the maximum possible revenue for the team in expectation, given the specific ticketing constraints. However, given the importance of ticketing revenue, we can appreciate concerns of handing complete control over ticket prices to a computer program. This led us to create our Manual Price Adjustment Guide, which incorporates our price predictions and ticketing analyst input, guiding prices closer to their optimal levels using past data and analyst expertise. The Manual Price Adjustment Guide is the main output of our work, and is discussed in more detail in Section 5.3.

### 3.2.2 Dynamic Program Formatting Algorithm

In this section, we describe the algorithm we use to represent the dynamic program as a directed acyclic graph (DAG) so that it can be solved by existing Python packages. These packages are designed to solve graph problems quickly and are implemented in CPython, which is written in both C and Python and is therefore much faster

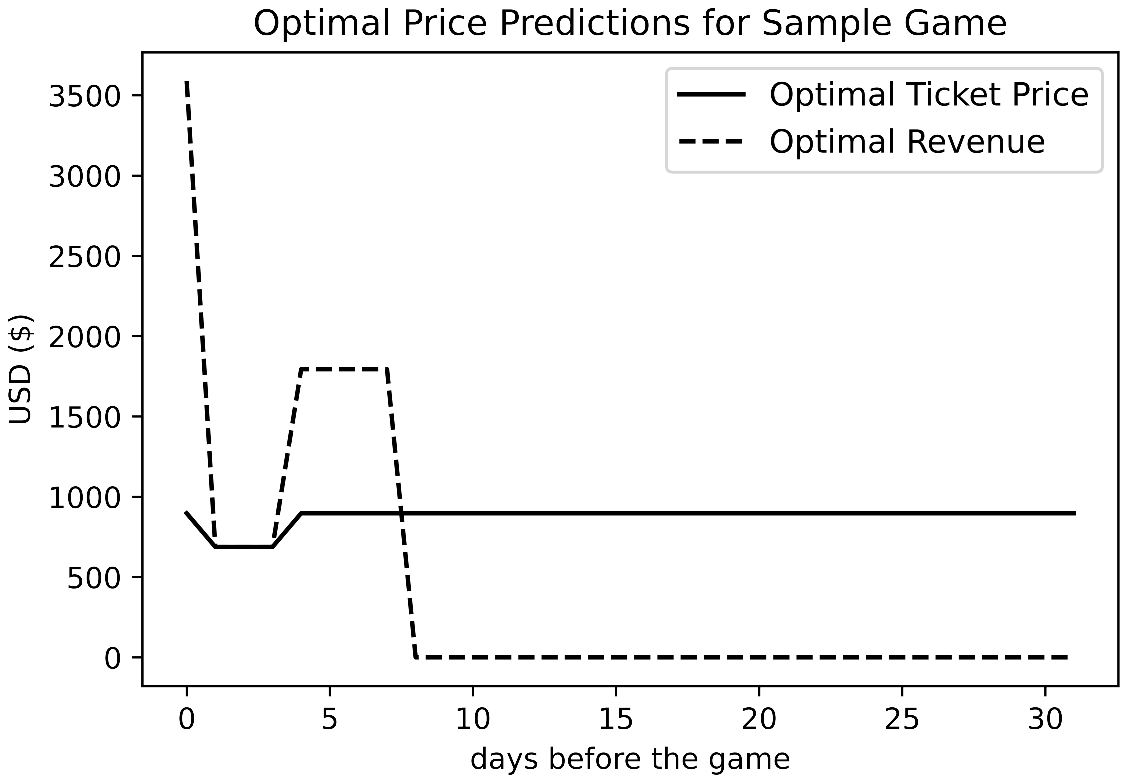


Figure 3-2: Graph depicting the optimal price plan output by our dynamic program. For each time period before the game, we have a computed optimal ticket price, as well as an optimal revenue. The two dynamic program specifications are equivalent: when we solve to calculate the number of tickets sold per period, we can determine what price those tickets are sold at, which can be combined to compute revenue (the dashed curve above).

than any pure Python implementation. Using these packages greatly decreases the runtime of our dynamic program and motivates the need for an algorithm to format the dynamic program properly. A list of these packages can be found in Appendix A, along with all other packages used in our ticketing application.

At a high level, our algorithm creates a DAG representation of the dynamic programming problem similar to the one seen in Figure 3-1. We consider each unique pair of (*ticket quantity*, *time period*) subproblems as an individual node, with edges representing transitions between time periods where we sell tickets. We weight our edges with the revenue we expect to make from that transition. Using Figure 3-1 as an example, the edge from the black node on day 4 to the white node representing zero tickets remaining on day 3 would have a weight of $3 * p_4$, since we are selling all three tickets at the transition price. Because we cannot go backwards in time or unsell tickets, there can be no cycles in our graph. With our dynamic program set up as a DAG in this way, we can work to find the maximum path from our starting node (the furthest amount of time from the game, with all ticket inventory remaining) to any of our terminal nodes (the time period where the game occurs, with any ticket inventory remaining).

Algorithm 1 shows the pseudocode of this dynamic program formatting algorithm. The Python package we use for our graph solving algorithm, `scipy`, takes the graph in the form of a square matrix, where the cell $(i, j)$ represents the weight of an edge from node $i$ to node $j$. This is the representation we create in Algorithm 1, where each row represents a node that is a unique pair of (*ticket quantity*, *time period*). Rather than setting these edges to positive values, however, we set then to *negative* values. This is because our graph solving algorithm is designed to look for a *shortest* path. By looking for a shortest path in a graph with all negative edge weights, we will ultimately find a *longest* path that maximizes our revenue.

**Algorithm 1** Dynamic Program Formatting Algorithm

```
Data: ticketTransitionFunction, list of time periods, count of available tickets
Result: Matrix representing the dynamic program as a DAG
initialization  Empty square matrix of width and height (time period count) * (available ticket count + 1)
for timeIndex in range(time period count) do
    for ticketQuantity in range(available ticket count + 1) do
        ticketMatrixRow = (timeIndex) × (available ticket count) + ticketQuantity
        ticketMatrixColumn = (timeIndex + 1) × (available ticket count)
        ticketPrice = ticketTransitionFunction(timeIndex, ticketQuantity)
        matrixValue = ticketPrice * ticketQuantity
        ticketMatrix[ticketMatrixRow][ticketMatrixColumn] = -1 * matrixValue
    end
end
```

Once this algorithm creates the DAG matrix representation of the ticketing dynamic program problem, it is passed to the `scipy` DAG solver, which returns an optimal path that is reconstructed to show the result in Figure 3-2. This price plan can then be used to set specific price levels over time or used in other applications like the Manual Price Adjustment Guide, as described in Chapter 5.

As we can see from the algorithm and the description of the dynamic programming application above, an understanding of the ticketing transition function is essential to making the dynamic program work and generating our price plans. As mentioned previously, this transition function is related to our customer demand model, which gives us an understanding of how much customers are willing to pay for tickets with different attributes (such as proximity to the game or seat quality). In the next chapter, we describe in detail how we create the ticketing transition functions through customer demand modeling, allowing us to run our dynamic program and generate the pricing plans seen in this chapter.

# Chapter 4

# Customer Demand Modeling

In this chapter we describe how we model customer demand, a key piece of our dynamic pricing model. The need for a customer demand model is motivated in Chapter 3, where we discuss how our dynamic program requires an understanding of how many tickets we can sell at what price for each time period. We start building this understanding through our demand modeling framework, which is the core of how we understand customer demand. We then describe our continuous method for modeling customer demand, and how issues related to data sufficiency and curve fitting caused the continuous method to be inadequate for our work. We then describe the discrete method we ultimately used to model customer demand, including our construction of Demand Matrices and the matrix completion methods we use for price inference. We conclude the chapter by describing some results from the Demand Matrices and this discrete inference method.

## 4.1 Demand Modeling Framework

As partially described in Chapter 3, customer demand is understood as a willingness to pay a specific price for a certain quantity of tickets. Different customers usually have different willingness to pay, based on aspects of their background (such as income, age, and gender, among others) and aspects of the product they are purchasing. Individual customer willingness to pay is often difficult to capture, but can be aggregated to give an understanding of market willingness to pay, or, more specifically, how many tickets can be sold to *all* customers when different prices are set.

To understand market willingness to pay, we need to understand what drives customers to pay different prices for different tickets. Once we understand the main drivers of willingness to pay, we can filter past sales to only contain transactions that meet certain willingness to pay criteria. For example, if we determine that a seat's location impacts willingness to pay for that seat, we might filter the seats into different location categories. From there, we use prices for the transactions in each category along with inference techniques to understand what quantity of tickets we will sell at different prices for seats in each location category.

### 4.1.1 Willingness to Pay Drivers

Because it is difficult to understand each customer's willingness to pay without an understanding of their background and preferences, we focus on the aggregate willingness to pay by all customers. This is easier to model, as instead of attempting to discern features of the customer base's backgrounds, we can focus on their preferences in aggregate. Moreover, since it is unlikely that the team's fan base will change dramatically in size or demographics from year to year, this assumption should hold in practice as well. By focusing on preferences in the aggregate, we can simply look at the impact of observable ticket characteristics on customer willingness to pay. While price is an observable characteristic that does affect consumer demand, it is not the only such characteristic. Some tickets may differ with respect to their position in the stadium. Others differ with respect to the opponent, day of the game, time of the game, and other factors. Since tickets have a fixed shelf life and expire after the game, there is also an expiration quality, captured by how close we are to the game. Capturing and distinguishing between these three main non-price factors is crucial for properly modeling customer demand.

### Proximity to the Game

The first and easiest of these factors to capture is proximity to the game. This is easy to measure because we can simply calculate how much time remains until the game begins. More generally, since customer preferences are unlikely to change on a minute-by-minute or hour-by-hour basis, we can look at different days before each game and consider how many tickets might have been sold on a given day. In our application with the Spurs, we often group days together to create consolidated time periods called "time buckets"[1]. Extracting this time factor in this way allows us to then focus on the other two factors that do not change over time: seat quality and game quality.

### Seat Quality

While seat quality is not as easy to compute as proximity to the game, there are a number of factors that make it relatively simple to measure. Within the stadium, tickets will naturally fall in different sections that have similar distances and angles to the playing surface. While not identical, tickets in these sections often have similar qualities, and as a result can be aggregated together into a quality grouping that allows us to price them all similarly. For the Spurs, this ticket quality metric is referred to as a ticket's "price code". As with proximity to the game, more details on these adjustments made for the Spurs application can be found in Section 5.1.1.

### Game Quality

The final and most difficult to measure aspect of a game is the game quality itself. This is driven by a number of factors, including the opponent, the time of the game, the day of the game, any potential giveaways, and other intangible aspects that make one game more enjoyable for fans than another. Since development of such a metric could require its own thesis, we relied on the Spurs metric for game quality, referred to as a game's "tier". Each tier segments games by these intangible factors based on

[1] More details on these groupings can be found in Section 5.1.2.

how well tickets have sold in the past and how appealing an upcoming game might be. Using these game quality tiers allows us to focus on modeling how demand changes with each tier. Depending on the consistency of sales within each tier, these tiers could then be updated and revised to better segment games and improve pricing in the future.

Having isolated these three drivers of customer demand, we turn to the actual demand modeling itself. By filtering transactions into groups based on these drivers (for example, all price code 1 tickets sold on the day of a tier 3 game), we can control for them and investigate how prices and quantities within each group interact. More specifically, we can consider a certain quantity of tickets and see what price we sold those tickets at in the past (or vice-versa). Often, however, we do not have prices for all possible quantities of tickets we might want to sell, making it impossible for our dynamic program to consider all possible subproblems. This is where our price inference comes in: using the available transaction data in each group, we infer prices for quantities that we might not have a historical price for. We start with continuous price inference, which offers the benefit of a continuous price spectrum but is often difficult to implement in practice. We then turn to discrete price inference, which naturally fits the dynamic program problem structure and was our ultimate choice for modeling customer demand.

## 4.2 Continuous Price Inference

Our first approach for inferring customer demand was a continuous price approach. We essentially wanted to create a demand curve much like those from a traditional economics course, with a continuous line providing a quantity for all possible price values (and vice-versa). With a continuous price-quantity curve, we could understand approximately how many tickets we would sell for any given price one could set. This would be beneficial not only for look-forward dynamic programming, but also for projection and evaluation of current prices, as the continuity of the function would give us an expected quantity for any price level. For these reasons, continuous price

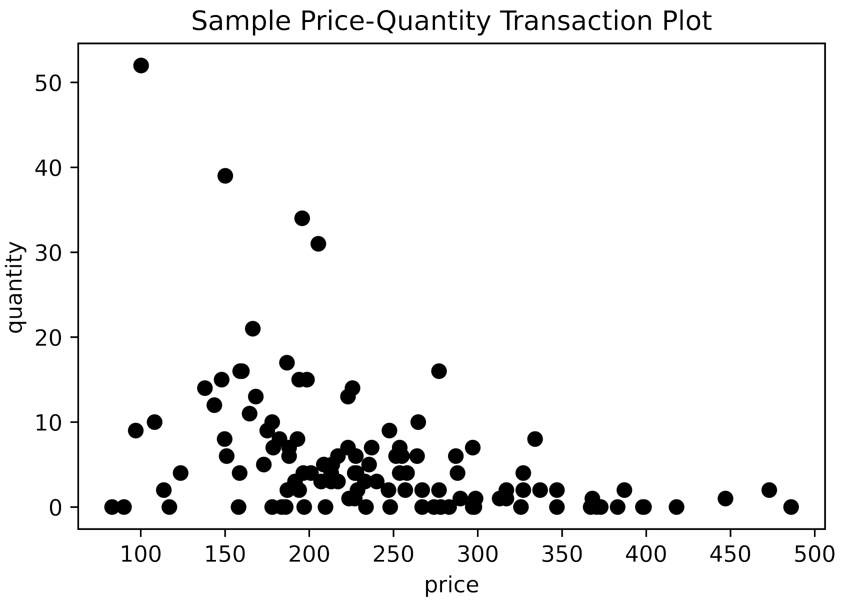


Figure 4-1: Graph depicting price and quantity pairs for each transaction in games of Tier 1 quality and seat quality price codes F-O. Prices for zero quantities are computed by linearly interpolating prices on the dates between two transactions. This data comes from the day of the game, when transactions are much more common.

inference was the first method we attempted for customer demand modeling.

Our continuous demand modeling differed slightly from the rest of our demand modeling in that we used transaction-level data rather than period-level data. As with all of our demand modeling, we start controlling for willingness-to-play drivers by filtering the transaction data to only sales of a specific proximity to the game, seat quality, and game quality. For each transaction in those groups, we plot a point representing the transaction price and the quantity sold as part of that transaction. This plot can be seen in Figure 4-1. From these transaction points, we could fit a continuous line that would achieve our goal of providing a price for all possible quantities of tickets. The goal of using this transaction-level approach was to develop a method that would allow us to price tickets not only by proximity to the game, ticket quality, and game quality, but also by the number of tickets a customer hoped to purchase.

To get this transaction level pricing, we would attempt to fit a curve to the transaction data shown in Figure 4-1. Using the function definition for this curve, we could then interchangeably move between all different price and quantity values, successfully modeling customer demand. We attempted these curve fits by using transaction binning and a couple of different curve fit functions in an effort to obtain a well-formed demand curve that would work for different quantities and prices. As we will see over the next few sections, while these approaches worked for some ticket segments, they generally struggled to fit appropriate curves for all or even most segments. This ultimately motivated our turn to discrete price inference, which we will discuss in Section 4.3.

### 4.2.1 Transaction Binning for Curve Fitting

While Figure 4-1 represents a fairly well-behaved transaction plot, most transaction plots did not turn out as well. They often had large amounts of zero-quantity points, as we can see in Figure 4-2, making traditional curve fitting difficult. Because of these zeros, we often had a near-horizontal line output from the curve fitting process, regardless of the function used in fitting (an example of this can be seen in Figure 4-3). To help address this overwhelming zero issue, we used transaction binning, which averages sets of the transaction points to create a representative subset. We then fit a curve to this subset of points, which resulted in the more realistic demand curve shapes seen in Figure 4-7. However, as we will discuss later, these demand curves often still had issues of their own that prevented them from providing good price and quantity estimates.

In selecting the subsets of points for these bins, we used two main methods. Our first method created a predetermined number of $n$ equally spaced bins along the price axis, then averaged all the points in each bin to get a representative point. While the resulting bin averages may have been made up of differing amounts of points depending on where the bin fell on the price axis (as tickets were sometimes concentrated in low or high price areas), these bins often did a better job representing the full spectrum of prices contained in the transactions. Our second method created enough bins such that each bin contained a predetermined number of $m$ points. This method also did a good job representing the full spectrum of prices, but could sometimes become

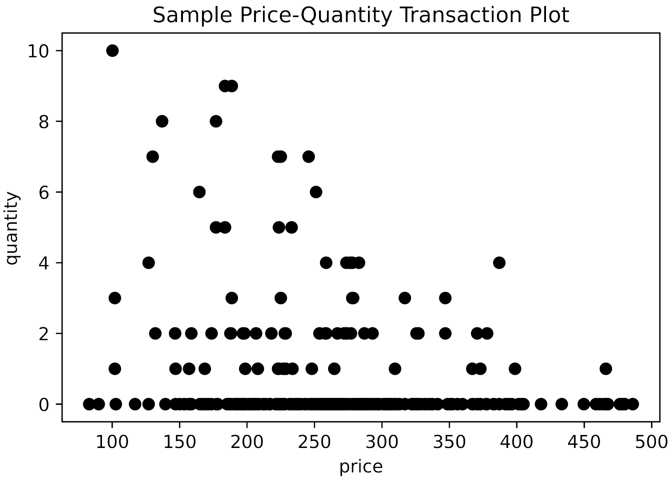


Figure 4-2: Graph depicting price and quantity pairs for each transaction in games of Tier 1 quality and seat quality price codes F-O. Prices for zero quantities are computed by linearly interpolating prices on the dates between two transactions. This data comes from between four and seven days before the game, when transactions are less common and we occasionally have days of no ticket sales.

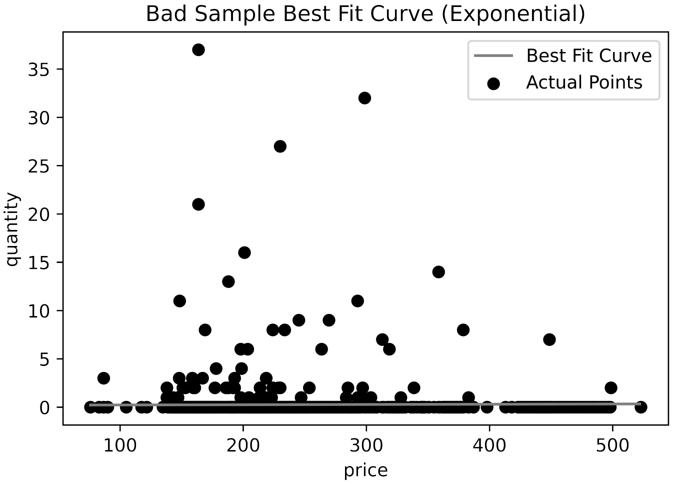


Figure 4-3: Graph depicting a bad best-fit exponential curve on price-quantity transaction data for each transaction in games of Tier 1 quality and seat quality price codes F-O. Prices for zero quantities are computed by linearly interpolating prices on the dates between two transactions. This data comes from between 61 and 90 days before the game, when transactions are less common and we often have days of no ticket sales. As we can see from the graph, the large amount of zero quantity transactions overwhelms the curve fitting function, causing it to fit an almost horizontal line at zero.

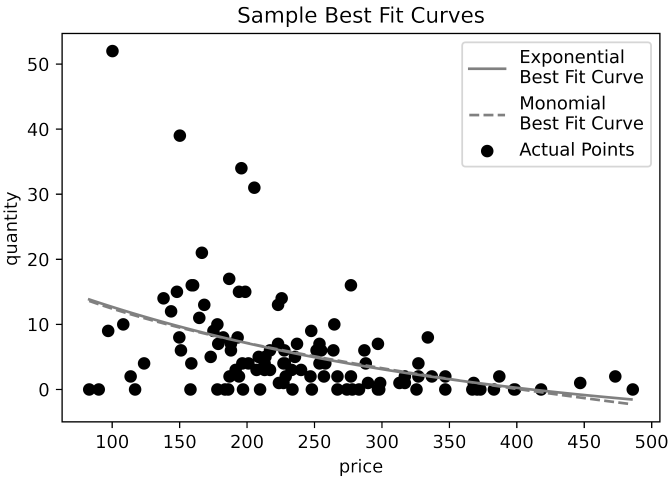


Figure 4-4: Graph depicting good best-fit exponential and monomial curves on price-quantity transaction data for each transaction in games of Tier 1 quality and seat quality price codes F-O. Prices for zero quantities are computed by linearly interpolating prices on the dates between two transactions. This data comes from the day of the game, when transactions are much more common. As we can see from the graph, both curves is well-formed (nearly identical) and provide reasonable price estimates for different quantities based on the data.

too tightly bunched in certain ranges depending on how many points were used in each bin. Examples of how these binned points relate to the full plot of transaction points can be seen in Figures 4-5 and 4-6.

### 4.2.2 Curve Fitting Functions

Once the points were binned, we attempted to fit curves to them in order to convert between all possible price and quantity values. Here, we used two main functions: an exponential function and a monomial function. We selected these functions based off patterns we were seeing in the data and the traditional shape of the demand curve, which is meant to be downward sloping and often takes a parabolic shape. The specifications for these functions can be seen in Equations 4.1 and 4.2, where $p$ is the provided price, $\hat{q}$ is the estimated quantity, and the other variables are parameters adjusted by the curve fitting library.

$$\hat{q} = a * e^{b*p+c} + d \tag{4.1}$$

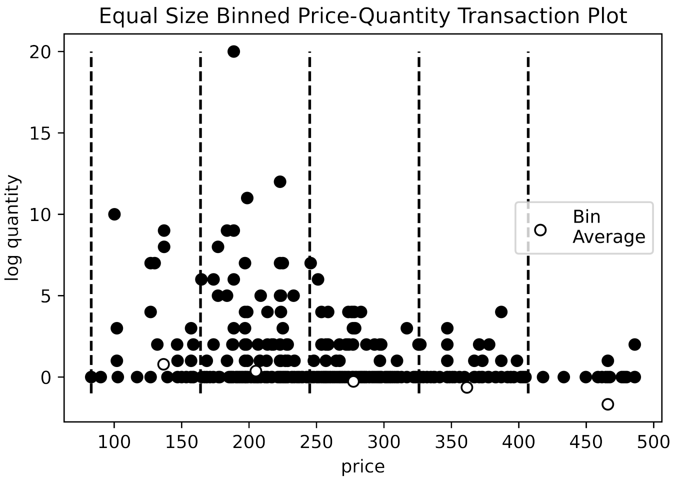


Figure 4-5: Graph depicting the transaction data from Figure 4-2, grouped into $n = 5$ bins that are equally spaced along the price axis. The average of each bin is shown in the white dots on the chart. While still mostly close to zero (data is displayed on a log scale), this method helps to remove some outliers and assist with slightly better curve fitting.

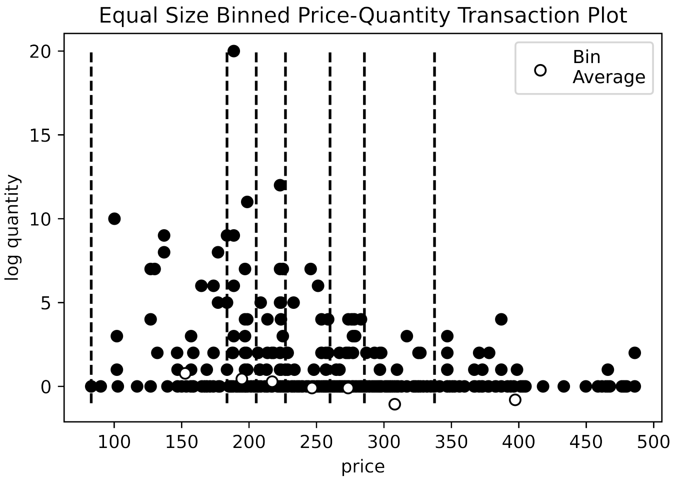


Figure 4-6: Graph depicting the transaction data from Figure 4-2, grouped into bins that each contain an equal number of points (in this case, $m = 200$). Each bin is created by taking the next 200 points (as ordered by price values) that have not yet been binned. The average of each bin is shown in the white dots on the chart. While still mostly close to zero (data is displayed on a log scale), this method helps to remove some outliers and assist with slightly better curve fitting.

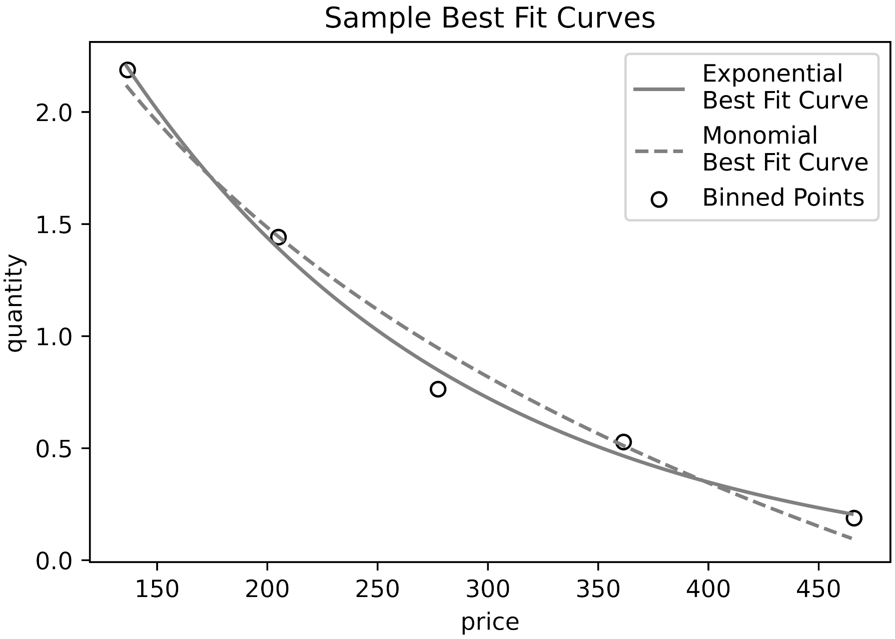


Figure 4-7: Graph depicting the binned transaction data from Figure 4-5, where data was grouped into five equally-spaced bins. These points represent the bin averages from that graph, with best fit exponential and monomial curves. As we can see, while the curve shape is more appropriate for both curves, it does not price any integer quantity of tickets, making it unusable for performing price inference.

$$\hat{q} = a * p^b + c \tag{4.2}$$

Ultimately, the results from the continuous curve fitting were mixed. While we had some good curve fits, as demonstrated in Figure 4-4, there were also poor curve fittings, such as those in Figure 4-7. Even for curves that were well-shaped, such as those in Figure 4-7, we often had cases where the quantities available were less than a single ticket or the curves did not price all possible ticket options. These results were mostly driven by sparsity in the underlying transaction data, which was not sufficient to fit continuous curves well.

The poor results of our continuous price inference, along with a desire to move away from transaction-based pricing to better fit our dynamic programming, inspired a shift to a different type of price inference. In particular, we realized that we only needed to price whole numbers of tickets, rather than the full range of decimal ticket quantities priced by this continuous method. As a result, we shifted to the discrete price inference approach initially discussed in Section 4.1, which became the method we ultimately used in our application.

## 4.3 Discrete Price Inference

With our struggles to fit continuous price curves documented in Section 4.2, we realized that performing discrete inference would offer many benefits and fit our dynamic programming approach better than the continuous method. The main benefit was the data required for this discrete inference: because we only generate prices for discrete ticket quantities, rather than the continuous range of quantities, we do not need all of the data required to fit a full continuous curve. Moreover, we do not truly need all the information provided by a continuous price curve, as our dynamic program is framed from the perspective of selling a certain number of tickets each period. Therefore, we only need prices for discrete quantity values, rather than price estimates for all (even non-integer) quantity values. With these built-in advantages, a move to a discrete inference was a natural fit.

This shift to discrete modeling led us to create the Demand Matrix, which is the backbone of our price recommendations. Like a price curve, a Demand Matrix tells us at what price we will be able to set to sell a certain number of tickets. However, rather than do this on a transaction basis, a Demand Matrix does this on a time period basis. This structure also fits our dynamic program better and allows us to make better recommendations.

In this section, we cover the details of our Demand Matrix implementation. We start by discussing the Demand Matrix itself, what it represents, and how we construct it. We then describe how we can use matrix completion to infer prices for ticket quantities where we have no available data. Finally, we conclude by discussing the actual matrix completion methods we used and the results of those methods.

### 4.3.1 The Demand Matrix

As described above, the Demand Matrix is the backbone of our price recommendations. The Demand Matrix serves as the engine that drives all aspects of our pricing tools, especially our dynamic program. In this section, we describe the intuition behind the Demand Matrix, along with how it is created.

The Demand Matrix was originally created to solve the problems with continuous price inference described in Section 4.2. Specifically, we did not have sufficient data to fit continuous price curves well. As we found, we did not necessarily need all of the information from a continuous price curve. Since our dynamic program was only considering the sale of discrete ticket quantities, we could focus on predicting prices for discrete ticket quantities. We also realized that our dynamic programming approach, as specified, was solving the problem of what price to set if we wanted to sell $q$ tickets over a time period $t$ (rather than how to price a certain number of tickets in a transaction). While the additional transaction-level pricing would have been helpful (which is why we initially started with continuous price inference), it was not strictly necessary. This led us to switch from the per-transaction inference done in the continuous space to a per-time period inference. Together, these realizations helped inspire the creation of the Demand Matrix.

A Demand Matrix, as seen in Figure 4-8, is a matrix of prices, where the cell in row $i$ and column $j$ represents the price at which one could sell $j$ tickets over time period $i$. We create a unique Demand Matrix for each different combination of seat quality and game quality (effectively controlling for these variables), so all cells with prices are filled in based on sales data that are in the seat and game quality groups for this Demand Matrix. As an example from the figure, the price of \$437.00 in cell $(1, 2)$ represents the average price of all tickets from tier 1 games where we sold exactly two Charter tickets over the period of 31 to 90 days before the game. Since tickets are often sold in groups of multiple tickets per transaction, we take the average price of all given transactions (rather than an average price weighted by the number of tickets in each transaction). Repeating this process for all possible quantities of tickets and

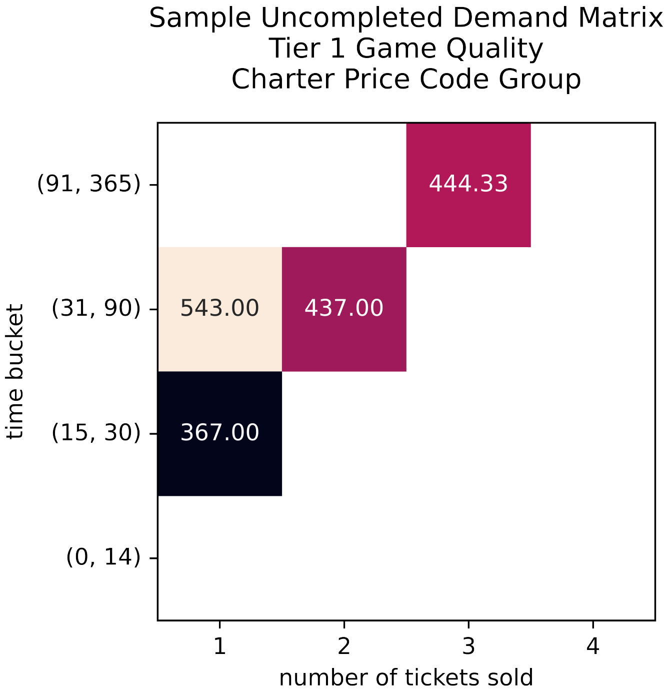


Figure 4-8: A sample uncompleted Demand Matrix for tickets to tier 1 games in the Charter price code group. Each cell represents the average transaction price for which we have sold exactly that many tickets over that time period in the past, with the x-axis denoting the ticket quantities and the y-axis denoting the time periods.

all time periods gives us the Demand Matrix as shown in the figure.

A pseudocode algorithm for the Demand Matrix creation process can be see in Algorithm 2. It follows the same procedure as above, but iterates over games instead of quantities. This is because each game only provides one sample per time period. As we need to know exactly how many tickets were sold over that time period, we can only consider sales from that game once, rather than over multiple quantity values. Also, since we may have multiple games where we sold the same amount of tickets over the same time period, we initially store lists of prices, rather than an actual average. We then compute the average once we have considered all games and time periods, as can be seen in the second half of the algorithm.

**Algorithm 2** Demand Matrix Creation Algorithm

```
Data: sales transaction data, seat quality, game quality, list of time periods, count
      of available tickets
Result: Demand Matrix as described in Section 4.3.1
initialization Empty square matrix of width (available ticket count) and height (time
 period count)
filteredTransactionData = SELECT * FROM (sales transaction data) WHERE
 seatQuality == (seat quality) and gameQuality == (game quality)
for timeIndex in range(time period count) do
    timeTransactionData = SELECT * FROM (filteredTransactionData) WHERE
     daysBefore in (time period count)[timeIndex]
    for gameLabel in timeTransactionData.gameLabels do
        gameTransactionData = SELECT * FROM timeTransactionData WHERE
         gameLabel == timeTransactionData.gameLabels
        gameSales = SUM(gameTransactionData.seatCount)
        matrixCellSales = DemandMatrix[timeIndex][gameSales]
        DemandMatrix[timeIndex][gameSales] = APPEND(matrixCellSales,
         avgGamePrice)
    end
end
for timeIndex in range(time period count) do
    for gameSales in range(available ticket count) do
        matrixCellSales = DemandMatrix[timeIndex][gameSales]
        DemandMatrix[timeIndex][gameSales] = AVERAGE(matrixCellSales)
    end
end
```

As mentioned above, within each Demand Matrix we segment transactions based on their proximity to the game. To control for the seat and game quality, we filter for those attributes before creating the Demand Matrix. As a result, we end up with a collection of Demand Matrices. An example collection of Demand Matrices created for different game qualities (measured in tiers) and ticket qualities (measured by price code groups) can be seen in Figure 4-9, demonstrating how the Demand Matrices scale to capture all drivers of willingness to pay.

As we can see from Figures 4-8 and 4-9, there are often cases where we have not sold certain ticket quantities over specific time periods (visualized by the white, empty squares in the Demand Matrices). In these cases, similarly to the continuous case, we need to infer a price. However, because of the structure of the Demand Matrix, we do not need to rely on continuous curve fitting or other such methods. Instead, we can take advantage of the matrix structure and use matrix completion, which we describe in Section 4.3.2.

### 4.3.2 Demand Matrix Completion for Price Inference

With the structure of the Demand Matrix as described in Section 4.3.1, a method was needed to infer the missing prices and complete the Demand Matrix so we could price all different quantities of tickets over all time periods. This is known as a matrix completion problem, and it is well-studied in math and computer science.

A matrix completion problem is the task of inferring values for missing or hidden values of a partially observed matrix. The most widely-known example of the matrix completion task is the Netflix problem, where a matrix is provided with the entry in row $i$ and column $j$ representing the rating of movie $j$ by user $i$ (if user $i$ has watched movie $j$, otherwise it is blank). The goal is to predict the missing values in the matrix to make better recommendations to users on what to watch next. When solved well, the completed matrix can be used to make ratings recommendations for all users on all movies.

The key assumption underlying the matrix completion problem is the low rank assumption, which implies that the true completed matrix has as low a rank as possible[2]. This essentially means that the matrix columns are linear combinations of each other, that is, given two column vectors $\vec{x}$ and $\vec{y}$ in the same matrix, another

[2] Readers interested in more detail behind this assumption and matrix completion in general can find more information in [13].

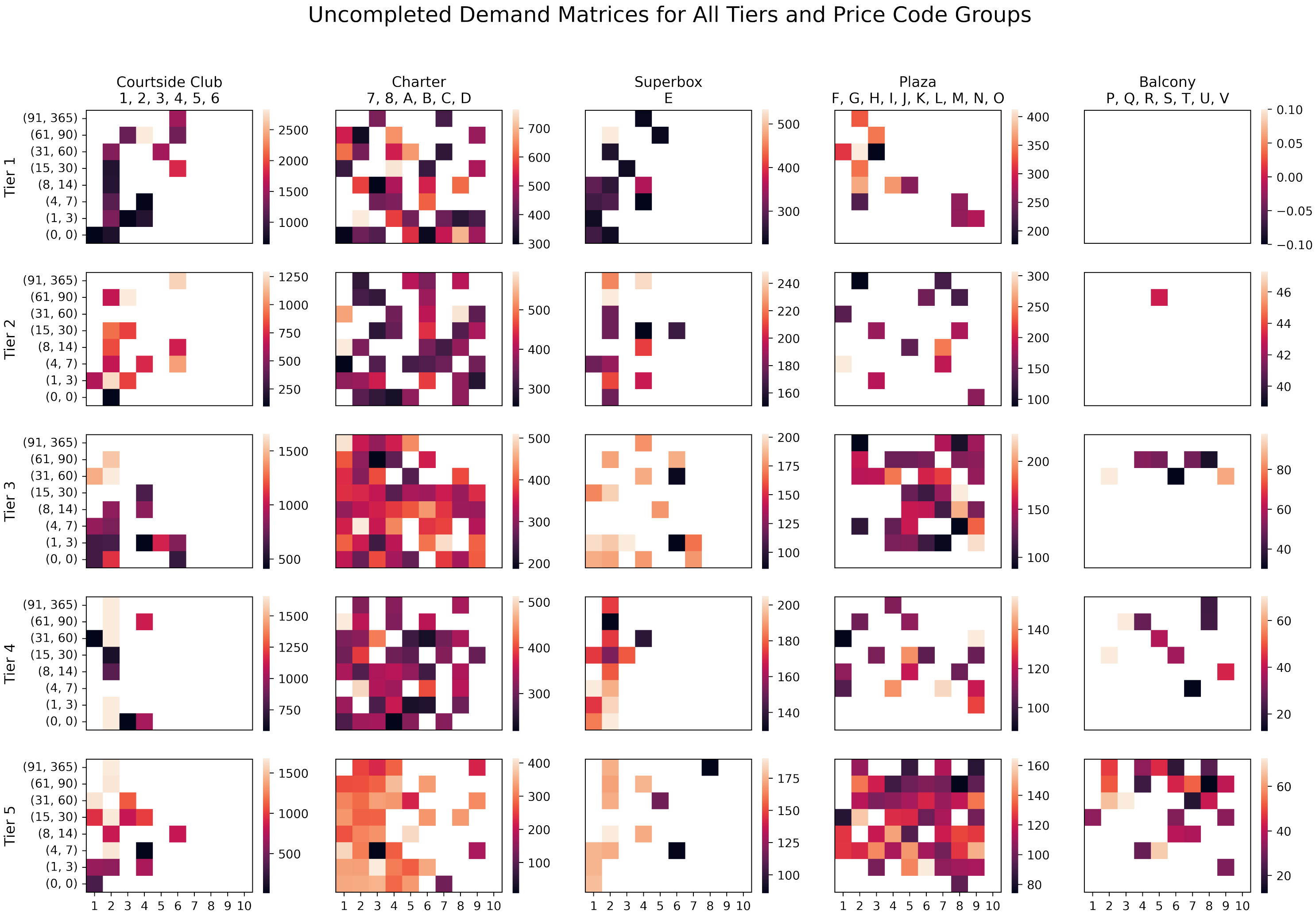


Figure 4-9: A sample collection of uncompleted Demand Matrices for all combinations of price codes and price code groups. Along the major x-axis are the different price code groups (which decrease in quality from left to right) and along the major y-axis are the different game quality tiers (which decrease in quality from top to bottom). The minor axes are the same as in a standard Demand Matrix, like the one shown in Figure 4-8: the y-axis represents different time periods, while the x-axis represents different ticket quantities.

column vector $\vec{z}$ in that matrix could be represented as $\vec{z} = a * \vec{x} + b * \vec{y}$ (where $a$ and $b$ are scalar constants). While the specifics of how this assumption affects matrix completion methods and the matrix completion problem as a whole are outside the scope of this work, it is important to address how the Demand Matrix satisfies this assumption in order to assert the correctness of our matrix completion methods described in Section 4.3.3.

As initially set up, there is no inherent reason for the Demand Matrix to satisfy the low rank assumption. This would imply that different quantities of tickets are priced similarly over all time periods, which is not necessarily true. However, there is reason to believe that ticket prices should increase regularly for all quantities as we approach the game, and that relationships between different quantities of tickets should stay similar over time. For this reason, transposing the Demand Matrix such that we have ticket quantities as row values and time periods as column values allows it to satisfy the low rank assumption, enabling us to use a full suite of matrix completion methods to infer the missing prices.

As described in this section, the structure of the Demand Matrix and its ability to satisfy the low rank assumption (when transposed) make it a great candidate for price inference through matrix completion methods. In these next two sections, we turn to describing the variety of methods we used to complete the Demand Matrices and their results.

### 4.3.3 Demand Matrix Completion Methods

When performing matrix completion on our Demand Matrices, we used three main methods: nearest neighbors (NN) completion, singular value decomposition (SVD) completion, and a hybrid method.

Our first matrix completion method was a standard NN approach, which looks at the existing values in each column to gauge the similarity between different columns. Using these similarities, the NN method will match complimentary columns that are similar in certain cells, but have other cells where one column is missing a value that exists in the other. For our NN implementation, we used the KNNImputer package from the sklearn library. Our calculations used five neighbors and weighted neighbor contributions based on their similarity. This method had the benefit of being explicitly designed for matrix completion and implemented by sklearn, which has a large array of prediction and inference packages that are considered top-of-the-line.

Our second matrix completion method was an SVD approach, which used a matrix average and singular value decomposition to fill in missing matrix values. A singular value decomposition (SVD) is a way of factoring a matrix that produces two component matrices and a singular value matrix, which has only non-negative real values on its diagonal and zeroes elsewhere [2]. These values represent the relative contribution of different rows and columns of the component matrices to the original matrix factorization. Dropping the lowest of these values, for example, will likely change the matrix by very little, while dropping the largest singular value could leave the matrix unrecognizable. By dropping some of the lowest singular values and reconstructing the matrix, we achieve a smoothing effect that can help prevent any large changes in price throughout the matrix. However, since the singular value decomposition process requires a full matrix to be performed, we needed to fill any missing values with the matrix average before performing the decomposition, which may have introduced more error than the NN approach. Once the matrix was filled, we performed an SVD, factoring the matrix. After dropping the three lowest singular values, we would then reconstruct the matrix using the new singular values and original component matrices, returning our completed matrix. This process was implemented using the linalg package from the numpy library, and provided the benefit of offering a high-fidelity reconstruction of the matrix while also smoothing the price values.

The third and final matrix completion method we used is referred to as a "hybrid" method and is a combination of the first two: we used NN to fill in all the missing values, then performed an SVD and dropped the lowest three singular values. This was an attempt to provide the completion benefits of the NN method (without taking a matrix average) and the smoothing benefits of the SVD.

To implement all of these methods, we transposed the matrix before performing completion to satisfy the low rank assumption (as described in Section 4.3.1).

We would then perform the matrix completion operation and transpose the matrix again afterwards, returning it to the original Demand Matrix format. Together, these three methods provided an array of approaches with different strengths to tackle the Demand Matrix completion problem.

### 4.3.4 Demand Matrix Completion Results

The results of our Demand Matrix completion methods are evaluated in two ways. The easiest way to view these results is through the heatmap representation seen in Figures 4-10 and 4-11, which visualize the completed versions of the Demand Matrices seen earlier in Figures Figures 4-8 and 4-9[3]. The additional values and colors on the heatmaps help demonstrate how price information from other areas was used to infer prices for missing values.

In addition to the visualizations, results for a Demand Matrix completion method can be generated by comparing their predicted prices with actual primary sales data. To perform this type of testing, we split our primary sales data into training and testing sets, selecting 80% of the games contained in the transaction data for the training set and 20% for the testing set. When generating the sets, we ensured that they had an approximately equal mix (within 3%) of games from different years and of different qualities to ensure there were no confounding timing or game quality factors driving the results. Using the training set, we created our uncompleted Demand Matrices, which were then filled in with the matrix completion methods described in Section 4.3.3. Once completed, we went through each of the time periods contained in the Demand Matrices. For each time period, we calculated the number of tickets sold in the testing set during that time period. We were then able to select our recommended price from the completed Demand Matrix by looking at the cell whose row and column values correspond with the examined time period and computed quantity sold. Finally, we compared that Demand Matrix predicted price to the actual price the test set tickets had been sold for (averaged over the number of test transactions). Using the quantity sold in conjunction with this price comparison,

[3] Additional examples of completed Demand Matrix visualizations can be seen in Appendix B.

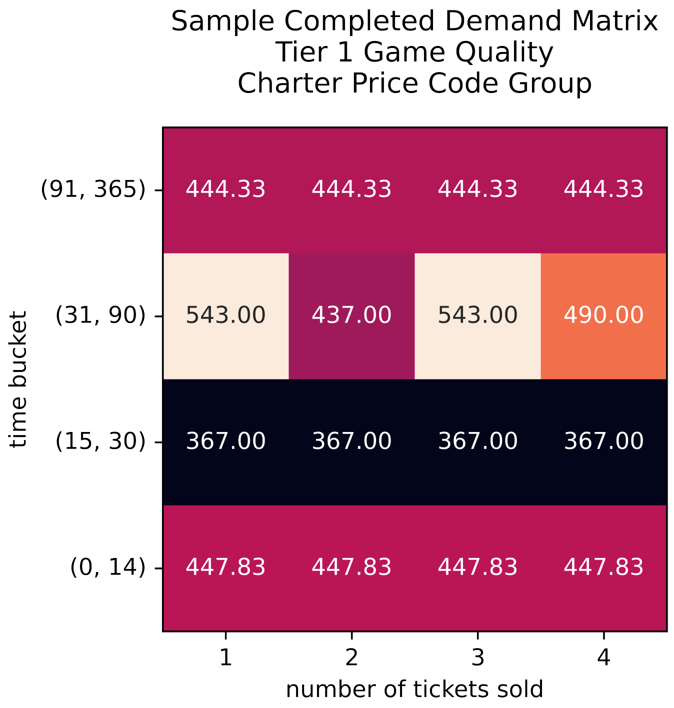


Figure 4-10: A sample completed Demand Matrix for tickets to tier 1 games in the Charter price code group, completed using the NN method. Each cell represents the average transaction price for which we have sold exactly that many tickets over that time period in the past, with the x-axis denoting the ticket quantities and the y-axis denoting the time periods. In the case of cells that are now filled compared to 4-8, those prices were inferred using the NN matrix completion method.

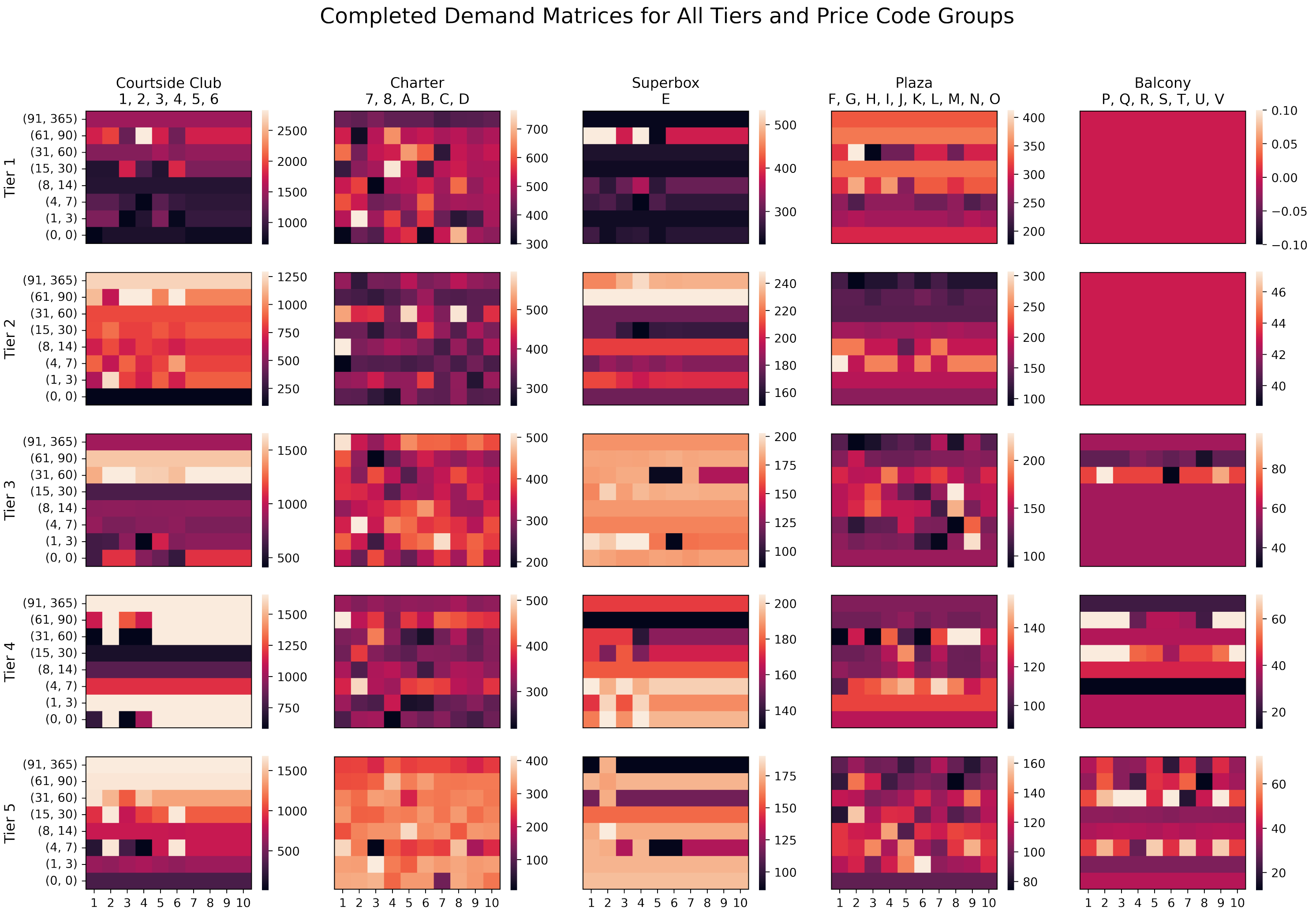


Figure 4-11: A sample collection of completed Demand Matrices for all combinations of price codes and price code groups, completed using the NN method. Along the major x-axis are the different price code groups (which decrease in quality from left to right) and along the major y-axis are the different game quality tiers (which decrease in quality from top to bottom). The minor axes are the same as in a standard Demand Matrix, like the one shown in Figure 4-8: the y-axis represents different time periods, while the x-axis represents different ticket quantities. In the case of cells that are now filled compared to 4-9, those prices were inferred using the NN matrix completion method.

we were able to compute revenue differences alongside our price differences for each matrix completion method.

When evaluating a matrix completion method, we hope to see that the prices predicted by our completed Demand Matrices are relatively close or slightly above historical prices, suggesting that we are pricing tickets in-line with where they have been in the past with a slight bias towards increasing prices. From our tests described above, this would be reflected in price and revenue differences that are fairly close to zero, with a slight bias in the positive direction (again implying that we have a slight bias towards increasing prices).

The actual results from each of these three methods can be found in Table 4.1. In addition to comparing the three matrix completion methods, we compared two methods of measuring ticket quality: through price code groups and individual price codes. This difference is explained thoroughly in Section 5.1.1, and amounts to a difference in how data is aggregated between specific price codes and groups of price codes. Individual price codes may generate more relevant Demand Matrices, since they only use transactions from the same type of seat, but they often have less data points than those in groups, potentially decreasing their accuracy. Including this grouping difference in our matrix completion results allows us to compare those aggregation methods as well.

As we can see from Table 5.1.1, the third "hybrid" completion method performed the best based on our evaluation metrics, with the closest to zero (but slightly positive) revenue and price differences, suggesting it is the most appropriate completion method to use in our applications. Surprisingly, the methods using Demand Matrices for each individual price code reported much higher revenue differences but a lower percentage of positive price differences than those that used a Demand Matrix for the price code groups. This is likely a result of lower amounts of data used for the individual price codes, giving skewed prices for certain quantity levels. More information on the grouping of price codes can be found in Section 5.1.1 and Chapter 5 as a whole, where we discuss specific adjustments made in our applications of the techniques described here. This result suggests we should use the hybrid method in tandem with the price

| Method / Price Code Type | Positive Price Diff. | Price Diff. Mean / Median | Revenue Diff. Sum |
|---|---|---|---|
| NN / Group | 54.83% | \$13.66 / \$2.82 | \$149,398.76 |
| SVD / Group | 60.51% | \$13.37 / \$7.56 | \$146,225.19 |
| **Hybrid / Group** | **54.06%** | **\$12.79 / \$2.59** | **\$139,935.42** |
| NN / Individual | 46.77% | \$11.55 / \$-2.43 | \$269,512.91 |
| SVD / Individual | 47.14% | \$11.47 / \$-1.65 | \$263,133.92 |
| Hybrid / Individual | 47.03% | \$11.43 / \$-2.38 | \$265,320.45 |

Table 4.1: Results from our matrix completion tests, performed as described in Section 4.3.4. The first column contains the matrix completion method used, as well as the price code type (either grouped, as described in Section 5.1.1, or individual, where we create a new Demand Matrix for each price code). The second column lists the percentage of transactions with a positive price difference, that is, the number of transactions where our inferred price was higher than the actual transaction price. The third column lists the mean and median of the price differences. The final column lists the sum of the revenue differences between the revenues we inferred and the actual revenues of the transactions. We note that the hybrid method applied to groups of codes (**bold**) performs the best overall, as it has the smallest revenue difference while still reporting mostly positive price differences. Results for individual price codes are mixed, with lower positive price difference percentages but higher revenue difference sums, and may be a result of limited data.

code groups, which is the procedure we ultimately follow in Chapter 5.

While these matrix completion method results are helpful and promising, they do not provide the same validation as a full comparison with secondary market data. This is because there is only one price listed on the Spurs platform each time a ticket is sold. As a result, we cannot know if a customer would have paid more for that ticket without seeing a similar transaction with a different price. This validation, done with secondary market data from StubHub, is performed in Section 5.2. While we are ultimately not able to validate nor reject any of these methods from the results in Section 5.2, the results shown here suggest the hybrid completion method is well-suited for Demand Matrix completion.

Overall, the results of our matrix completion methods are promising, and they suggest that our discrete price inference is providing accurate and reasonable prices that can be used to create optimal price plans with our dynamic program. Together, these methods enable us to dynamically price sports tickets, providing the optimal prices over time shown in Figure 3-2. These optimal prices are useful on their own, and they can provide a large amount of value for a team. However, we believe they can be more useful when combined with the expertise of ticketing analysts who currently price tickets teams. In Chapter 5, we discuss the different applications we developed for the Spurs and how they combine optimal prices and ticketing analyst expertise to maximize value for the team.

# Chapter 5

# Applied Dynamic Price Planning

As we have now detailed the theory and implementation of our dynamic programming and matrix completion approach to dynamic pricing, we move to the specifics of our dynamic pricing application with the Spurs. We begin this chapter with a brief review of adjustments made to apply our theory to the Spurs system, including the use of the team's price codes and time buckets to measure seat quality and proximity to the game. We then discuss our methods for validating the price plans created by our dynamic program, and share some results of comparisons with StubHub sales data. Next, we turn our attention to the main deliverable of this work: the Manual Price Adjustment Guide. We discuss how the guide is created and how it can be used within an organization to get the best of dynamic pricing and ticketing analyst expertise. We move on to our Attendance Projection Sheet, and then conclude with a summary of the software and procedures shared with the Spurs.

## 5.1 Spurs Application Adjustments

Throughout this work, many of the references to specific aspects of the dynamic pricing system have been left intentionally general to allow for a focus on the actual attributes of the system itself. These primarily include the variables and procedures used to refer to seat quality as well as proximity to the game. Because we simply used the Spurs tiers to measure game quality (as described in Section 4.1.1), there

| **Price Code Group Name** | **Group Price Codes** |
|---|---|
| Courtside Club | 1, 2, 3, 4, 5, 6 |
| Charter | 7, 8, A, B, C, D |
| Superbox | E |
| Plaza | F, G, H, I, J, K, L, M, N, O |
| Balcony | P, Q, R, S, T, U, V |

Table 5.1: Table containing the names of the Spurs' price code groups, along with the price codes that fall into each group. The price code groups offer an additional level of clustering that can be applied to help group seats with those of similar quality.

are no adjustments for that attribute here. In this section, we discuss how both seat quality and proximity to the game are addressed in our application with the Spurs, including groupings and generalizations made to ensure we have sufficient data for making price predictions.

### 5.1.1 Price Code Group Adjustment

As discussed in Chapter 4, seat quality is one of the primary drivers of customer willingness to pay for a ticket. Seats that are closer to the field of play and have better views are likely to command higher prices than those that are further from the action. While all seats will have their own unique view, each seat can often be grouped and priced with others nearby. For the Spurs, these seat groupings are called *price codes*, and they range from 1-8 and A-V (with price codes 1-8 having higher prices than those in A-V, and prices decreasing with higher numbers and later letters in the alphabet). Each price code represents a similar set of seats with comparable views of the court that can be priced the same.

In addition to the price codes, the Spurs also have *price code groups*, which create a level of grouping for seats. The Spurs have five price code groups, which are listed (along with the price codes each group contains) in Table 5.1. These price code groups provide an additional level of aggregation that can be further applied to help group seats with those of similar quality.

Because of constraints related to available data, there are generally not enough points to do price predictions for each individual price code (as suggested by our results in Section 4.3.4). As a result, our application for the Spurs aggregates seats by their price code group when creating our Demand Matrices, giving us one Demand Matrix for each combination of game quality tier (of which there are five) and price code group for a total of 25 Demand Matrices. This aggregation provides more transactions per Demand Matrix than only aggregating by price codes, which should allow for more accurate predictions.

### Price Code Adjustment Map

We note that this strategy of using price code groups is helpful for creating Demand Matrices, but is problematic when using them for pricing. This is because we need to price tickets by *price code*, not price code *group*, and the Demand Matrices created in this manner represent prices for price code groups as a whole. To address this problem, we created the Price Code Adjustment Map, which uses all our transaction data to compute an average price for the specific price code we are looking at as well as the group that price code is a part of. By taking the ratio of these two averages, we have a multiplier that can be applied to our entire Demand Matrix for that price code group. When multiplied by this multiplier, the prices in the Demand Matrix then reflect prices for that specific price code, rather than the group as a whole. An example of this for the Charter price code group can be seen in Figure 5-1, where the horizontal line represents the group average and each point represents the price code averages. The ratio of these points to the horizontal line would then be the multiplier for that price code. The somewhat linear nature of these averages and tightness of price code averages allows us to use this type of adjustment, which enables the use of additional transactions for prediction that would otherwise not be included.

Together, the use of the Spurs' price code groups and our Price Code Adjustment Map help us to properly capture the impact of seat quality on customer willingness to pay.

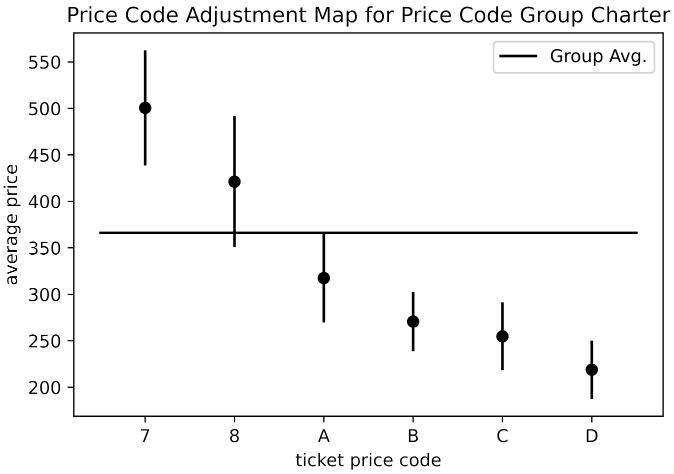


Figure 5-1: Graph depicting the average price for each price code in the Charter price code group, along with the group average. The ratio of each price code's average to the group average is used as a multiplier to adjust Demand Matrix prices, which are created using data for all price codes in the group but need to be able to price individual price codes.

### 5.1.2 Time Bucket Adjustment

In addition to seat quality, proximity to the game is another primary driver of customer willingness to pay. Because there are only so many tickets, however, there often are not sufficient transactions on every day before a game to allow us to make good predictions. As a result, we create groups of days before the game (similar to the price code groups), which allow us to see higher concentrations of ticket volumes. These groups of days, which we refer to as "time buckets", were provided by the Spurs based on their experience and how they group ticket sales. The time buckets, along with the name of the buckets, can be found in Table 5.2. We note that creating and using these time buckets implies an inherent assumption that customer demand is unchanged between different days in the same bucket. While this may not necessarily be true for the buckets currently used by the Spurs, the time buckets can be easily adjusted and changed so this core assumption remains true (so long as there is no overlap between buckets). The use of these time buckets, like that of the price code groups above, is another adjustment made to our dynamic pricing process to ensure there is sufficient data to make reasonable recommendations.

| Time Bucket Name | Bucket Days Before the Game |
|---|---|
| 3+ Months Out | 91-365 |
| 2-3 Months Out | 61-90 |
| 1-2 Months Out | 31-60 |
| 2 Weeks - 1 Month Out | 15-30 |
| 1-2 Weeks Out | 8-14 |
| 4-7 Days Out | 4-7 |
| 1-3 Days Out | 1-3 |
| Day Of | 0 |

Table 5.2: Table containing the names of the Spurs' time buckets, along with the periods of days that the time buckets cover. The inherent assumption when creating time buckets is that customer demand is constant within a bucket. For example, in the "1-2 Weeks Out" bucket above, we assume that customer demand for a ticket on day 9 is the same as on day 13.

## 5.2 Price Plan Validation

With the adjustments made to our dynamic pricing strategy as described in Section 5.1, we are able to generate our 25 different Demand Matrices which can be used to create optimal price plans. However, validation of these price plans is nearly impossible without an understanding of where ticket inventories began or where they ended for each game, since remaining ticket quantity is a key variable in the dynamic program. As a result, validation of our price plans is limited to comparisons of our Demand Matrix prices with those from historical transactions. If the price we set for a certain quantity of tickets is fairly close to the historical price, then we can feel confident that our price recommendations are accurate. This is the analysis performed with our matrix completion results in Section 4.3.4, where we compare Demand Matrices created from a training set of transaction data to a held out test set of transaction data to examine which completed matrix produced the closest set of prices.

However, because we are implementing dynamic pricing as a way to increase profit, we also want to know how much opportunity exists for price increases and how close we are to capturing the maximum price available. In general, this maximum price is thought to exist on secondary market exchanges such as StubHub, where broker desire to optimize profit enables them to get the best price possible for each ticket. By comparing our prices to those for tickets on secondary market exchanges such as StubHub, we can get a better sense for the full pricing opportunity and how close we are to capturing it.

### 5.2.1 StubHub Transaction Comparison

As mentioned above, comparing our prices to those of transactions from StubHub should provide us with an accurate sense of how much pricing opportunity exists and how much of that opportunity we are able to capture. This procedure was not straightforward: because StubHub did not use the same price codes as the Spurs, we needed to create a map that used a seat's section and row to assign it a price code. Once that was done, we could use our Demand Matrices to provide price recommendations for each seat and compare those to the actual prices the tickets were sold for.

However, we quickly noticed that the price difference between our recommendations and the StubHub sales was *negative*, that is, we were recommending prices higher than those that tickets were sold for on StubHub. This was a wholly counterintuitive result: since tickets sold on StubHub are not price-constrained in any direction and are sold by brokers who are purely hoping to maximize profit, they should represent the maximum possible price a ticket could be sold for. This result was also worrying to us, as it implied that we were recommending prices that were far higher than any customer would pay.

Ultimately, however, we found that our high price recommendations were not a result of an overzealous pricing system, but rather the result of consistently higher pricing from the Spurs. We found this by matching as many possible StubHub transactions with comparable transactions from the Spurs (based on the transaction price code, days before the game, and the number of tickets sold). A histogram of the resulting price differences can be seen in Figure 5-2, demonstrating that Spurs prices were meaningfully higher than StubHub prices for a vast majority of transactions.

We believe there are two possible reasons for this result. The first reason is the

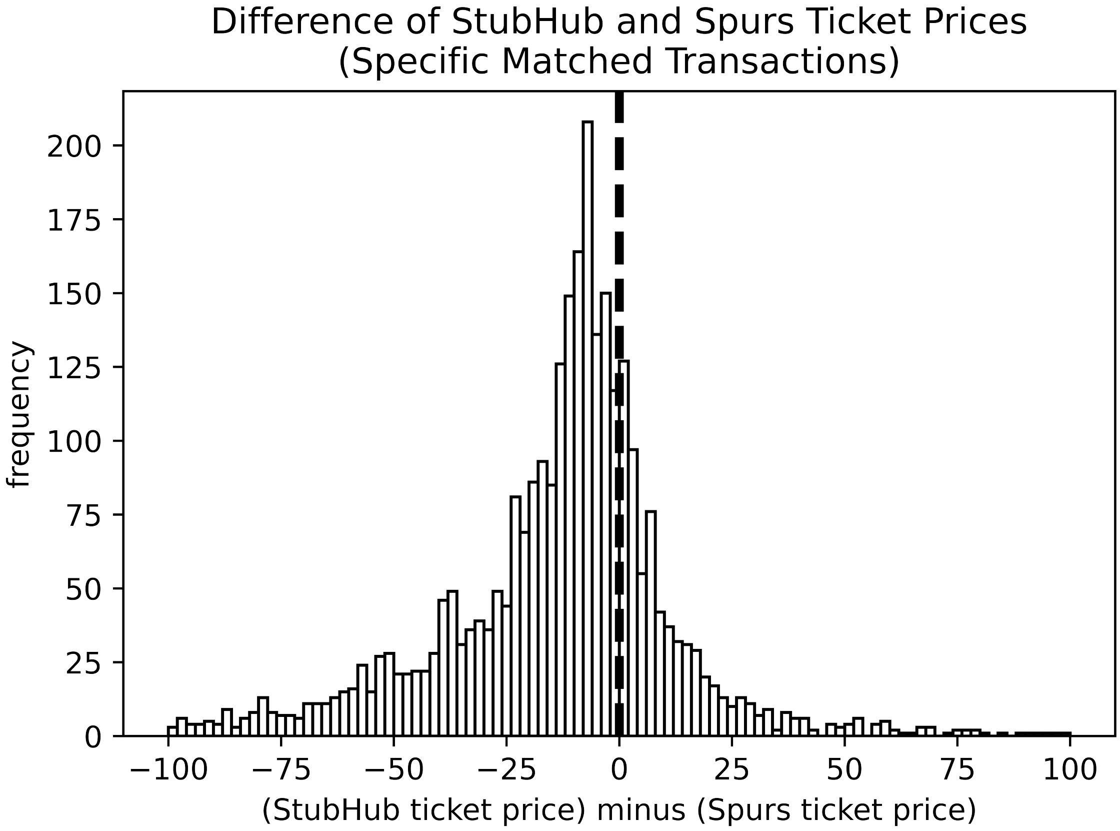


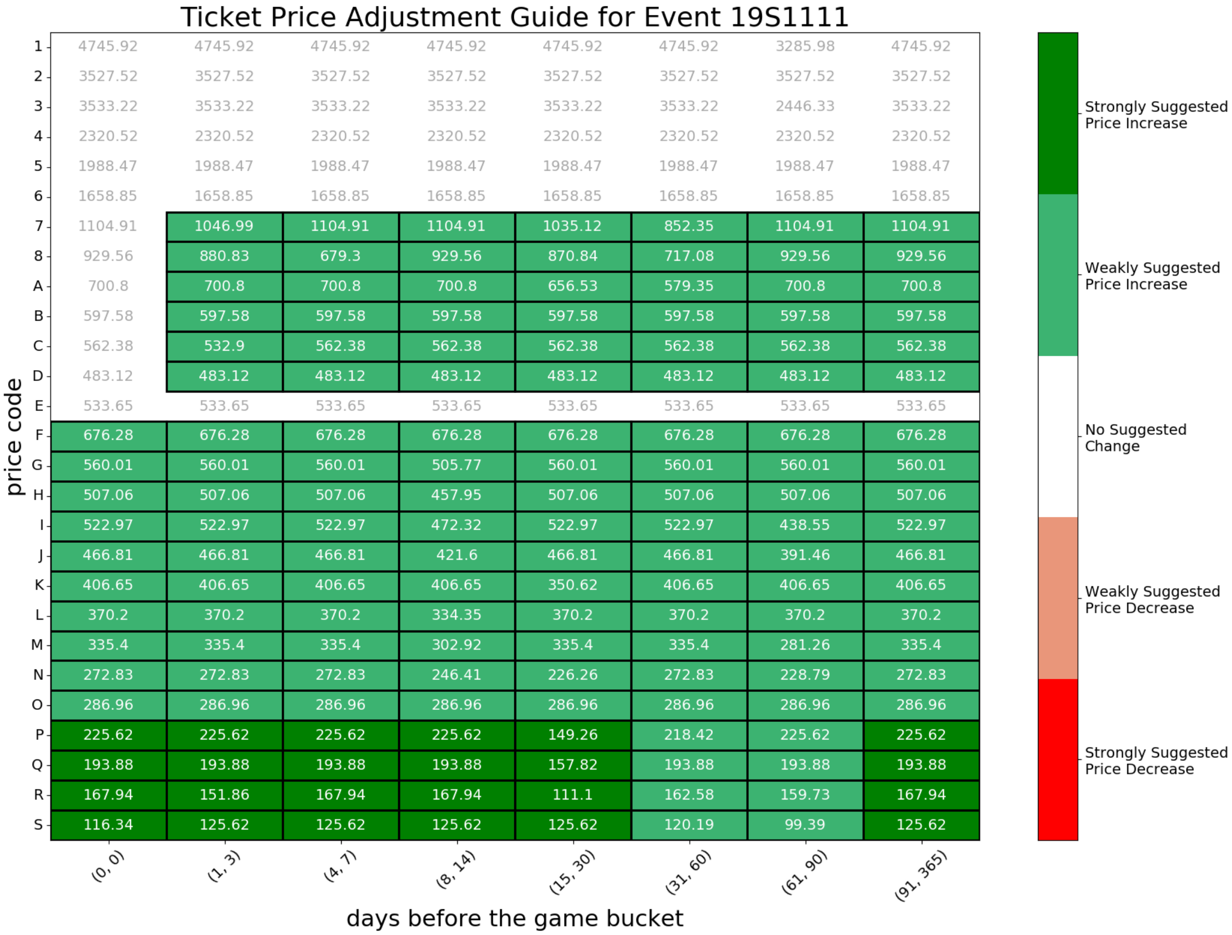
Figure 5-2: A histogram representing the frequency of transactions with different ticket prices between the StubHub price and the Spurs price. As we can see, the number and value of transactions where the Spurs charged a higher price than StubHub far outweighs the number and value of transactions where the reverse occurred.

number of season ticket holders in the secondary market, which we believe is far greater than the number of brokers. As these season ticket holders are likely unable to attend the game in question, they are willing to sell their ticket at any price, since they receive nothing for the ticket if they do not sell it. The second reason is the pricing power the Spurs have by selling tickets through Spurs.com. As many fans recognize and assign value to the brand name, they are willing to pay more for a ticket directly from the Spurs rather than on a secondary market. Together, we believe these two reasons explain the discrepancy between the prices charged by the Spurs and those on StubHub.

Overall, despite access to valuable secondary market data, we were unable to measure the possible pricing opportunity or confirm the validity of our pricing recommendations. As a result, we cannot recommend immediate deployment of our optimal price plans. Instead, we suggest usage of our main tools described in Sections 5.3 and 5.4. These tools work to combine the optimal price planning work we have performed here with ticketing analyst expertise to provide a "best-of-both-worlds" approach that will help improve current ticket pricing for the Spurs and help validate our dynamic pricing strategy.

## 5.3 Manual Price Adjustment Guide

The Manual Price Adjustment Guide is the main output of our work and the tool that we believe will be most valuable to the Spurs. This guide, an example of which can be seen in Figure 5-3, is designed to help ticketing analysts improve ticket prices by highlighting areas where our optimal price plan has identified an opportunity to meaningfully increase or decrease prices. The intent of the Manual Price Adjustment Guide is for the ticketing analyst to use it as a directional guide for adjusting prices, using their own expertise to set the actual level for the adjustment. In this way, the team gains all of the benefit from the direction of the suggested shift, which we believe should be generally accurate, while reducing possible risk of mispricing as a result of insufficient data.



| price code | (0, 0) | (1, 3) | (4, 7) | (8, 14) | (15, 30) | (31, 60) | (61, 90) | (91, 365) |
|---|---|---|---|---|---|---|---|---|
| 1 | 4745.92 | 4745.92 | 4745.92 | 4745.92 | 4745.92 | 4745.92 | 3285.98 | 4745.92 |
| 2 | 3527.52 | 3527.52 | 3527.52 | 3527.52 | 3527.52 | 3527.52 | 3527.52 | 3527.52 |
| 3 | 3533.22 | 3533.22 | 3533.22 | 3533.22 | 3533.22 | 3533.22 | 2446.33 | 3533.22 |
| 4 | 2320.52 | 2320.52 | 2320.52 | 2320.52 | 2320.52 | 2320.52 | 2320.52 | 2320.52 |
| 5 | 1988.47 | 1988.47 | 1988.47 | 1988.47 | 1988.47 | 1988.47 | 1988.47 | 1988.47 |
| 6 | 1658.85 | 1658.85 | 1658.85 | 1658.85 | 1658.85 | 1658.85 | 1658.85 | 1658.85 |
| 7 | 1104.91 | 1046.99 | 1104.91 | 1104.91 | 1035.12 | 852.35 | 1104.91 | 1104.91 |
| 8 | 929.56 | 880.83 | 679.3 | 929.56 | 870.84 | 717.08 | 929.56 | 929.56 |
| A | 700.8 | 700.8 | 700.8 | 700.8 | 656.53 | 579.35 | 700.8 | 700.8 |
| B | 597.58 | 597.58 | 597.58 | 597.58 | 597.58 | 597.58 | 597.58 | 597.58 |
| C | 562.38 | 532.9 | 562.38 | 562.38 | 562.38 | 562.38 | 562.38 | 562.38 |
| D | 483.12 | 483.12 | 483.12 | 483.12 | 483.12 | 483.12 | 483.12 | 483.12 |
| E | 533.65 | 533.65 | 533.65 | 533.65 | 533.65 | 533.65 | 533.65 | 533.65 |
| F | 676.28 | 676.28 | 676.28 | 676.28 | 676.28 | 676.28 | 676.28 | 676.28 |
| G | 560.01 | 560.01 | 560.01 | 505.77 | 560.01 | 560.01 | 560.01 | 560.01 |
| H | 507.06 | 507.06 | 507.06 | 457.95 | 507.06 | 507.06 | 507.06 | 507.06 |
| I | 522.97 | 522.97 | 522.97 | 472.32 | 522.97 | 522.97 | 438.55 | 522.97 |
| J | 466.81 | 466.81 | 466.81 | 421.6 | 466.81 | 466.81 | 391.46 | 466.81 |
| K | 406.65 | 406.65 | 406.65 | 406.65 | 350.62 | 406.65 | 406.65 | 406.65 |
| L | 370.2 | 370.2 | 370.2 | 334.35 | 370.2 | 370.2 | 370.2 | 370.2 |
| M | 335.4 | 335.4 | 335.4 | 302.92 | 335.4 | 335.4 | 281.26 | 335.4 |
| N | 272.83 | 272.83 | 272.83 | 246.41 | 226.26 | 272.83 | 228.79 | 272.83 |
| O | 286.96 | 286.96 | 286.96 | 286.96 | 286.96 | 286.96 | 286.96 | 286.96 |
| P | 225.62 | 225.62 | 225.62 | 225.62 | 149.26 | 218.42 | 225.62 | 225.62 |
| Q | 193.88 | 193.88 | 193.88 | 193.88 | 157.82 | 193.88 | 193.88 | 193.88 |
| R | 167.94 | 151.86 | 167.94 | 167.94 | 111.1 | 162.58 | 159.73 | 167.94 |
| S | 116.34 | 125.62 | 125.62 | 125.62 | 125.62 | 120.19 | 99.39 | 125.62 |

Figure 5-3: Sample Manual Price Adjustment Guide for the game coded “19S1111”. As described in Section 5.3, the dark green highlighted cells indicate areas with a suggested increase that has both significant magnitude and transaction quantity used in generating that significant change. Lighter green cells indicate high magnitude increases we are less confident in, or smaller increases that we are just as confident in. The same color scale applies for negative adjustments as well, which are not seen in this particular Manual Price Adjustment Guide.

The Manual Price Adjustment Guide is formatted as follows. Each row of the adjustment guide represents a single price code, while each column represents a single time bucket. The value in each cell represents how much our optimal price plan believes we should increase (or decrease) the price for a single ticket in that price code at that point in time, as compared to the current price. The coloring of the text and cells indicates the magnitude of the recommended change, as well as the number of transactions used to generate that recommendation. Dark green or dark red cells represent high magnitude changes that were generated from Demand Matrices with a significant transaction count, indicating we are more confident in them. Light green or light red cells are high magnitude changes that were generated from fewer transactions, or smaller changes generated from a similar number of transactions as the dark colored cells. In these cases, we indicate to the analyst that this change is worth acknowledging, but might be too small or uncertain to be worth making. Cells with no fill do not have enough supporting transactions to provide a recommendation, but may be colored to indicate what direction we think prices should move. A Manual Price Adjustment Guide is generated for each upcoming game, along with a summary sheet detailing the significant price recommendations for all upcoming games.

Use of the Manual Price Adjustment Guide allows the Spurs to gradually adjust to the recommendations of our dynamic pricing strategy. By signaling directionality on price changes, rather than providing specific magnitudes, prices can be adjusted as much or as little as the team is comfortable with. As the strategy is fine-tuned with additional data and parameter tweaks, the recommendations will only become more robust, allowing the team to further improve pricing over time.

### 5.3.1 Guide Creation Process

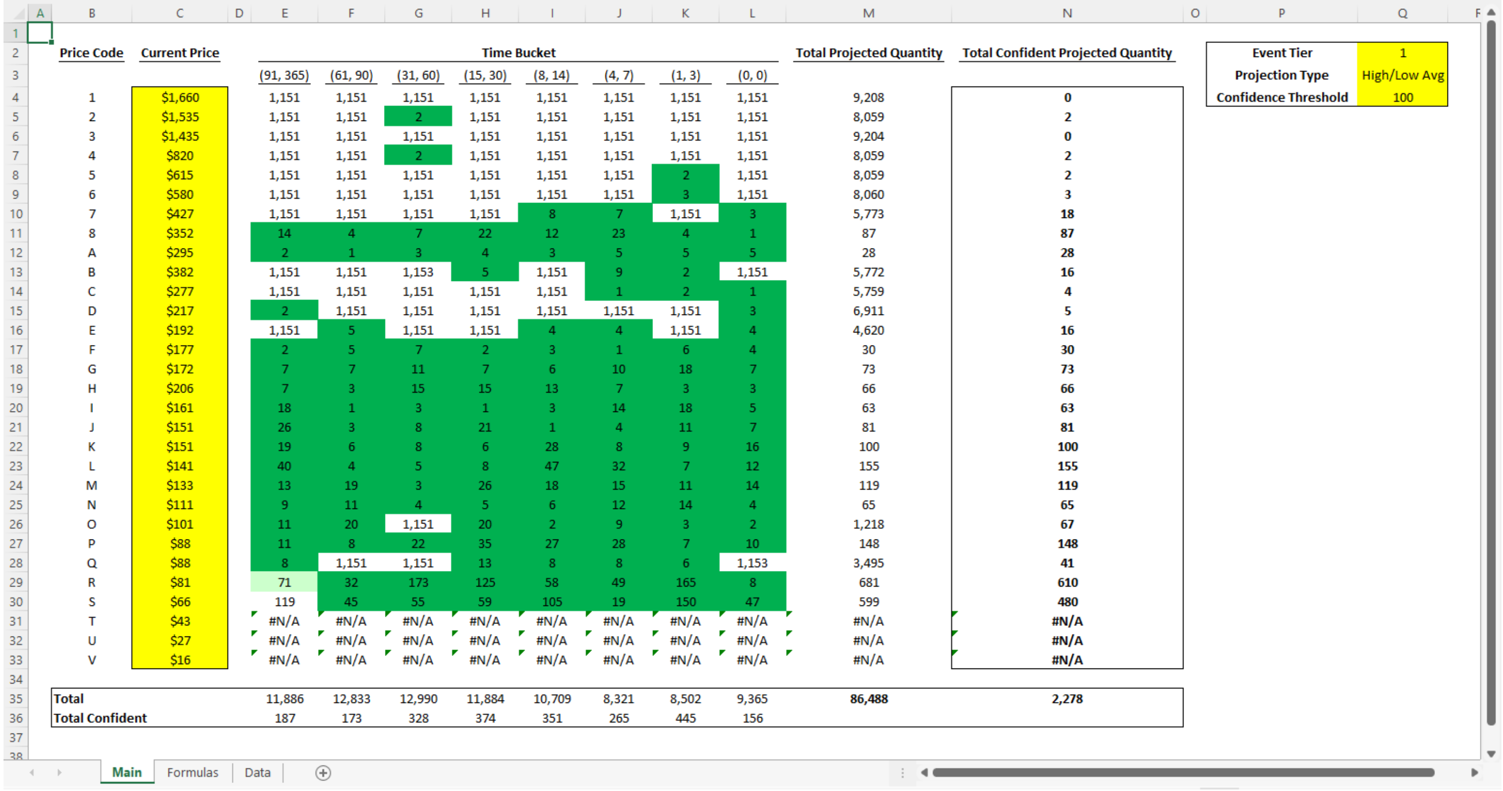
To create the Manual Price Adjustment Guide, we start by using our dynamic program and completed Demand Matrices to generate optimal price plans for all price codes for an upcoming game, based on the quantity remaining in each price code. An example of one of these optimal price plans can be seen earlier in Figure 3-2. Once the price plan is generated, we compare the recommended price at each time period to the current price set for that price code. If the difference meets certain thresholds related to a price increase or decrease, we color the text of that cell a different shade of green or red to signify the recommended change (as described above). If the quantity of transactions used in the underlying Demand Matrix for that recommendation also meets certain thresholds, then we color the entire cell the shade of green or red, signifying that this is a meaningful price change with sufficient transaction quantity to support it. By repeating this process for all price codes, we create the Manual Price Adjustment Guide seen in Figure 5-3.

### 5.3.2 Using the Manual Price Adjustment Guide

Use of the Manual Price Adjustment Guide is as follows. When evaluating ticket prices for an upcoming game, a ticketing analyst opens the relevant guide for that game. Looking at each of the cells, they can find the current time period they are in based on how far away the game is. Scanning this column up and down will reveal any fully colored cells, which signify a significantly suggested price increase or decrease. Based on their intuition, the analyst can then decide whether to accept that increase or decrease, and how much to adjust the price by. As the analyst uses the Manual Price Adjustment Guides more and gains a better understanding for the accuracy of their recommendations, they can better tailor the magnitude of these adjustments, leading to improved ticket pricing and increased profits for the team.

## 5.4 Attendance Projection Sheet

The Attendance Projection Sheet is the second main output of our work, and a tool that will help actively validate the price recommendations made by our dynamic pricing system. This Excel spreadsheet uses the data and predictions contained in our Demand Matrices to predict ticket sales for each upcoming time bucket. Ticketing analysts can interactively engage with the sheet by adjusting the current prices and watching as attendance projections respond to these adjustments. As a result, using this tool can help give them a sense for how accurate our price recommendations are:



| Price Code | Current Price | Time Bucket | | | | | | | | Total Projected Quantity | Total Confident Projected Quantity |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | (91, 365) | (61, 90) | (31, 60) | (15, 30) | (8, 14) | (4, 7) | (1, 3) | (0, 0) | | |
| 1 | $1,660 | 1,151 | 1,151 | 1,151 | 1,151 | 1,151 | 1,151 | 1,151 | 1,151 | 9,208 | 0 |
| 2 | $1,535 | 1,151 | 1,151 | 2 | 1,151 | 1,151 | 1,151 | 1,151 | 1,151 | 8,059 | 2 |
| 3 | $1,435 | 1,151 | 1,151 | 1,151 | 1,151 | 1,151 | 1,151 | 1,151 | 1,151 | 9,204 | 0 |
| 4 | $820 | 1,151 | 1,151 | 2 | 1,151 | 1,151 | 1,151 | 1,151 | 1,151 | 8,059 | 2 |
| 5 | $615 | 1,151 | 1,151 | 1,151 | 1,151 | 1,151 | 1,151 | 2 | 1,151 | 8,059 | 2 |
| 6 | $580 | 1,151 | 1,151 | 1,151 | 1,151 | 1,151 | 1,151 | 3 | 1,151 | 8,060 | 3 |
| 7 | $427 | 1,151 | 1,151 | 1,151 | 1,151 | 8 | 7 | 1,151 | 3 | 5,773 | 18 |
| 8 | $352 | 14 | 4 | 7 | 22 | 12 | 23 | 4 | 1 | 87 | 87 |
| A | $295 | 2 | 1 | 3 | 4 | 3 | 5 | 5 | 5 | 28 | 28 |
| B | $382 | 1,151 | 1,151 | 1,153 | 5 | 1,151 | 9 | 2 | 1,151 | 5,772 | 16 |
| C | $277 | 1,151 | 1,151 | 1,151 | 1,151 | 1,151 | 1 | 2 | 1 | 5,759 | 4 |
| D | $217 | 2 | 1,151 | 1,151 | 1,151 | 1,151 | 1,151 | 1,151 | 3 | 6,911 | 5 |
| E | $192 | 1,151 | 5 | 1,151 | 1,151 | 4 | 4 | 1,151 | 4 | 4,620 | 16 |
| F | $177 | 2 | 5 | 7 | 2 | 3 | 1 | 6 | 4 | 30 | 30 |
| G | $172 | 7 | 7 | 11 | 7 | 6 | 10 | 18 | 7 | 73 | 73 |
| H | $206 | 7 | 3 | 15 | 15 | 13 | 7 | 3 | 3 | 66 | 66 |
| I | $161 | 18 | 1 | 3 | 1 | 3 | 14 | 18 | 5 | 63 | 63 |
| J | $151 | 26 | 3 | 8 | 21 | 1 | 4 | 11 | 7 | 81 | 81 |
| K | $151 | 19 | 6 | 8 | 6 | 28 | 8 | 9 | 16 | 100 | 100 |
| L | $141 | 40 | 4 | 5 | 8 | 47 | 32 | 7 | 12 | 155 | 155 |
| M | $133 | 13 | 19 | 3 | 26 | 18 | 15 | 11 | 14 | 119 | 119 |
| N | $111 | 9 | 11 | 4 | 5 | 6 | 12 | 14 | 4 | 65 | 65 |
| O | $101 | 11 | 20 | 1,151 | 20 | 2 | 9 | 3 | 2 | 1,218 | 67 |
| P | $88 | 11 | 8 | 22 | 35 | 27 | 28 | 7 | 10 | 148 | 148 |
| Q | $88 | 8 | 1,151 | 1,151 | 13 | 8 | 8 | 6 | 1,153 | 3,495 | 41 |
| R | $81 | 71 | 32 | 173 | 125 | 58 | 49 | 165 | 8 | 681 | 610 |
| S | $66 | 119 | 45 | 55 | 59 | 105 | 19 | 150 | 47 | 599 | 480 |
| T | $43 | #N/A | #N/A | #N/A | #N/A | #N/A | #N/A | #N/A | #N/A | #N/A | #N/A |
| U | $27 | #N/A | #N/A | #N/A | #N/A | #N/A | #N/A | #N/A | #N/A | #N/A | #N/A |
| V | $16 | #N/A | #N/A | #N/A | #N/A | #N/A | #N/A | #N/A | #N/A | #N/A | #N/A |
| Total | | 11,886 | 12,833 | 12,990 | 11,884 | 10,709 | 8,321 | 8,502 | 9,365 | 86,488 | 2,278 |
| Total Confident | | 187 | 173 | 328 | 374 | 351 | 265 | 445 | 156 | | |

| | |
|---|---|
| Event Tier | 1 |
| Projection Type | High/Low Avg |
| Confidence Threshold | 100 |

Figure 5-4: A screenshot of the Attendance Projection Sheet interface. The yellow highlighted cells indicate areas for the user to interact, while the rest of the cells provide information on projected attendance values. Cells marked in green indicate a confident attendance projection, while those with no highlight indicate less confidence in the recommendation. Formulas and Data tabs at the bottom of the workbook contain background calculations and data from the Demand Matrices used to inform these attendance projections. Cells with "#N/A" values have no prediction.

the closer the attendance projection to the actual attendance, the more accurate the price recommendations. Since the attendance projections are constantly updating alongside the Demand Matrices, this provides an active validation process that can be monitored and used as a metric for how reliable our price recommendations are.

A screenshot of the Attendance Projection Sheet can be seen in Figure 5-4. In addition to providing attendance estimates for each price code, the sheet also provides a confidence indicator (based on the highlight of the cell) to denote whether we are confident in a certain recommendation. This is part of our design to help users evaluate and understand when our projections are thought to be accurate, and how far those projections might be from the true attendance for that game.

The Attendance Projection Sheet functions by using the information in our Demand Matrices to make informed ticket quantity estimates. When new optimal price plans are generated, the values in the Demand Matrices are written to a hidden sheet in the Attendance Projection Sheets. When a current ticket price is input by an analyst into column C, the Attendance Projection Sheet uses formulas to look up the closest matching prices in that ticket's Demand Matrix. Based on the difference between the quantity values that correspond with these closest prices, we assign the confidence ranking shown in green. These quantity values are then summed over different time periods and price codes to get our overall attendance projections.

With the Attendance Projection Sheet providing an additional tool for active validation, the Spurs are provided with a full toolkit that allows them to evaluate and embrace our dynamic pricing strategy as much as they are comfortable with. By providing this active validation tool, the Spurs can continue to evaluate the efficacy of our solution over time as it takes in more data and parameters are adjusted.

## 5.5 Dynamic Pricing Software Deployment

To deploy these tools and our overall dynamic pricing strategy, Python files implementing the entire system, including dynamic programming, Demand Matrix completion, Manual Price Adjustment Guide creation, and Attendance Projection Sheet data management (among other functions) were handed off to the Spurs for deployment. The package was designed with a main operation file to run all processes, as well as a single file with all adjustable parameters. The main operation file can be set to run daily, updating Manual Price Adjustment Guides and data for the Attendance Projection Sheet as new transactions come in and we come closer to upcoming games. Adjusting the parameter file enables continued iteration on different game tiers, price code groups, and time buckets as described in Section 5.1. When compiled, this package structure enables easy use for the Spurs, who can simply set the process file to run daily in the background of other processes. Overall, the Spurs gain the valuable ticket pricing insights provided by our Manual Price Adjustment Guide and Attendance Projection Sheet, with no daily upkeep and less than a full day of initial setup.

# Chapter 6

# Conclusion

In this section, we summarize the key insights of our work, provide a few avenues for additional future work, and give a brief closing statement.

## 6.1 Key Insights

In this section, we discuss the three key insights we believe future researchers should take away from this work. These insights include the inconsistency and opportunity in ticket pricing, principles for business software design, and the discrete optimization framework developed in our work. While not comprehensive, we believe they highlight the key contributions of this work.

### 6.1.1 Inconsistency and Opportunity in Ticket Pricing

As we have found from our work in customer demand modeling (in Chapter 4) and examination of the Spurs ticket transactions, there is a large degree of disparity in how tickets are priced. Often, we have found that similar tickets for similar games have been priced significantly differently, for reasons that are not immediately obvious. We believe these inconsistencies arise because of ticketing analyst expertise and biases that are not immediately apparent. On one hand, there is likely analyst expertise that is not captured in the current measurements of proximity to the game, seat quality, and game quality. Because these analysts have a better understanding of what separates game or seat quality than is codified in the current system, they may adjust prices in a way that seems inconsistent with the current willingness to pay metrics (i.e., a price that is much higher or lower than tickets in the same time bucket, tier, and price code) but is actually more optimal. However, because these understandings are not codified, they may be applied inconsistently by different analysts, resulting in price adjustments that are less optimal. Together, we believe these reasons partially explain the results of Section 4.3.4, which suggest there are inconsistencies in ticket pricing based on the spread (positive and negative) of how our prices compare to those of past transactions.

These inconsistent prices provide a meaningful opportunity for improvement in pricing processes and increase of revenue. Better understanding of why ticketing analysts make different adjustments (for example, a belief a game should be Tier 2 when it is labeled as Tier 1) can help isolate the drivers behind those adjustments and eliminate the inconsistent application of those adjustments that draws prices away from their optimal levels. More optimal prices, in turn, increase ticket revenue. Our system enables implementation of the process improvement described here, as by tweaking the designations for different time buckets, price codes, and game tiers, the system's price recommendations should become more consistent, indicating that we are representing more of the analyst expertise and moving closer to optimal ticket prices.

### 6.1.2 Trust in Business Software Design

When designing software for business use, that software's likelihood of implementation by the business can be improved by ensuring the software process is clear and enables collaboration with human operators. In general, businesses are uncomfortable with giving software complete control over revenue-driving processes, especially software that is unclear or difficult to understand. While making the software process interpretable for all stakeholders can help improve the business's trust in the software, helping to increase chances of acceptance and use, it also helps to include some step for human input or oversight in the software design. By allowing a human operator to step in and provide input or evaluate the process before it affects revenue, businesses can feel more comfortable trying the software, giving them an opportunity to see its impact and building trust in the software.

These principles of a clear software process and enabled human collaboration were key in our development of the Manual Price Adjustment Guide and Attendance Projection Sheet. Through extensive documentation and team discussions, the processes for both pieces of software were made clear to all stakeholders in the Spurs organization. Moreover, both incorporate a human input in their process, with the Manual Price Adjustment Guide allowing a ticketing analyst to make the final decision on pricing while the Attendance Projection Sheet uses analyst adjustments of current prices to update its attendance projections. These features have allowed the Spurs to develop a good deal of trust in the software, and should enable it to be rapidly deployed upon delivery.

### 6.1.3 Dynamic Program & Matrix Completion Framework

We believe the dynamic programming and matrix completion framework can be applied to any discrete optimization problem with incomplete information. There are some constraints that the discrete optimization problem must satisfy, including the optimal substructure attribute, finite subproblem attribute, and the low rank assumption (among others). If these constraints are met, then this framework should provide a useful first approach or full solution to the discrete optimization problem in question, as we have seen with this dynamic ticket pricing problem.

## 6.2 Future Work

While we have made significant progress on the dynamic pricing problem for sports tickets, there is still additional work that can be done. In this section, we discuss some of these main areas for future work, including further data collection and parameter tuning, development of adaptive game quality tiers, and additional secondary market testing. This list, while not comprehensive, represents the main continued areas of interest for the Spurs and areas that we believe will bear the most immediately fruitful results.

### 6.2.1 Additional Data Collection and Parameter Tuning

Data sufficiency for our consumer demand model was one of the primary issues we faced when developing our dynamic pricing strategy. We made several adjustments to account for this by aggregating data from different seat qualities and time periods, as described in Section 5.1. However, even with these adjustments, we still believe there may not be sufficient transaction volume to create reliable recommendations. As the system is designed to constantly improve with additional data, continuing to feed transactions from historical and more recent games will further improve its recommendations and generate more reliable prices.

When making the aggregation adjustments described in Section 5.1, we grouped seat qualities and filtered into time buckets based on the Spurs' current methods for grouping tickets. Because these ticket groupings may not have been designed for a predictive system like ours, they may not be ideal aggregations and may not properly capture customer demand. Furthermore, there are additional parameters that have not been fine-tuned, such as the price change thresholds and confidence thresholds used in the Manual Price Adjustment Guide and Attendance Projection Sheet. Further experimentation with these aggregation parameters and value thresholds could radically improve system performance by better capturing customer demand and better alerting the team when a pricing improvement opportunity exists.

### 6.2.2 Adaptive Game Quality Tiers

In the current version of the Spurs' tiers, which measure game quality, a game's tier is assigned statically based on factors about the game such as the day of the week, the opponent, and other variables that might drive customer demand. However, it is known that other non-static factors, such as recent team performance, can affect a customer's desire to attend the next home game, changing the game quality as a result. Moreover, while a game may be assigned a certain tier initially, it may sell differently over time, putting it closer to a different game quality tier. In both cases, price recommendations could be improved by dynamically assigning different tiers to a game based on developments that occur as we approach the game.

### 6.2.3 Additional Secondary Market Testing

As described in Section 5.2, one of the major barriers to validating the performance of our system was the tendency of StubHub tickets to be priced *lower* than those sold directly by the Spurs. Without an understanding of a ticket price ceiling, which we expect to find on an open market with more sophisticated firms, we cannot compute the value of pricing tickets optimally. One possible reason for the lower prices on StubHub is that most tickets on the platform are sold by season ticket holders, who may be less concerned with making a profit and more inclined to sell the ticket for a lower price if they are unable to attend the game. Further analysis of this secondary market data, including work to segment the tickets into season ticket holder sales and broker sales, could allow for more accurate analysis of where true prices are and better validation of our price recommendations.

## 6.3 Closing

Overall, we feel this thesis has established a strong framework for dynamic pricing sports tickets, incorporating advanced predictive analytics techniques with ticketing analyst expertise to provide intelligent pricing recommendations. Further adjustments and tweaks, along with our active validation techniques, will allow the system to reach its full potential and contribute to increasing ticket revenues for the San Antonio Spurs.

# Appendix A

# Software Packages

This appendix includes a list of the different (non-default) Python packages used in development of our dynamic pricing strategy. Please note some sub-dependencies may be missing from this list.

1. pandas
2. numpy
3. matplotlib
4. seaborn
5. scipy
6. sklearn
7. xlsxwriter

# Appendix B

# Additional Demand Matrix Completion Examples

The following appendix contains additional figures visualizing the results of some different matrix completion methods.

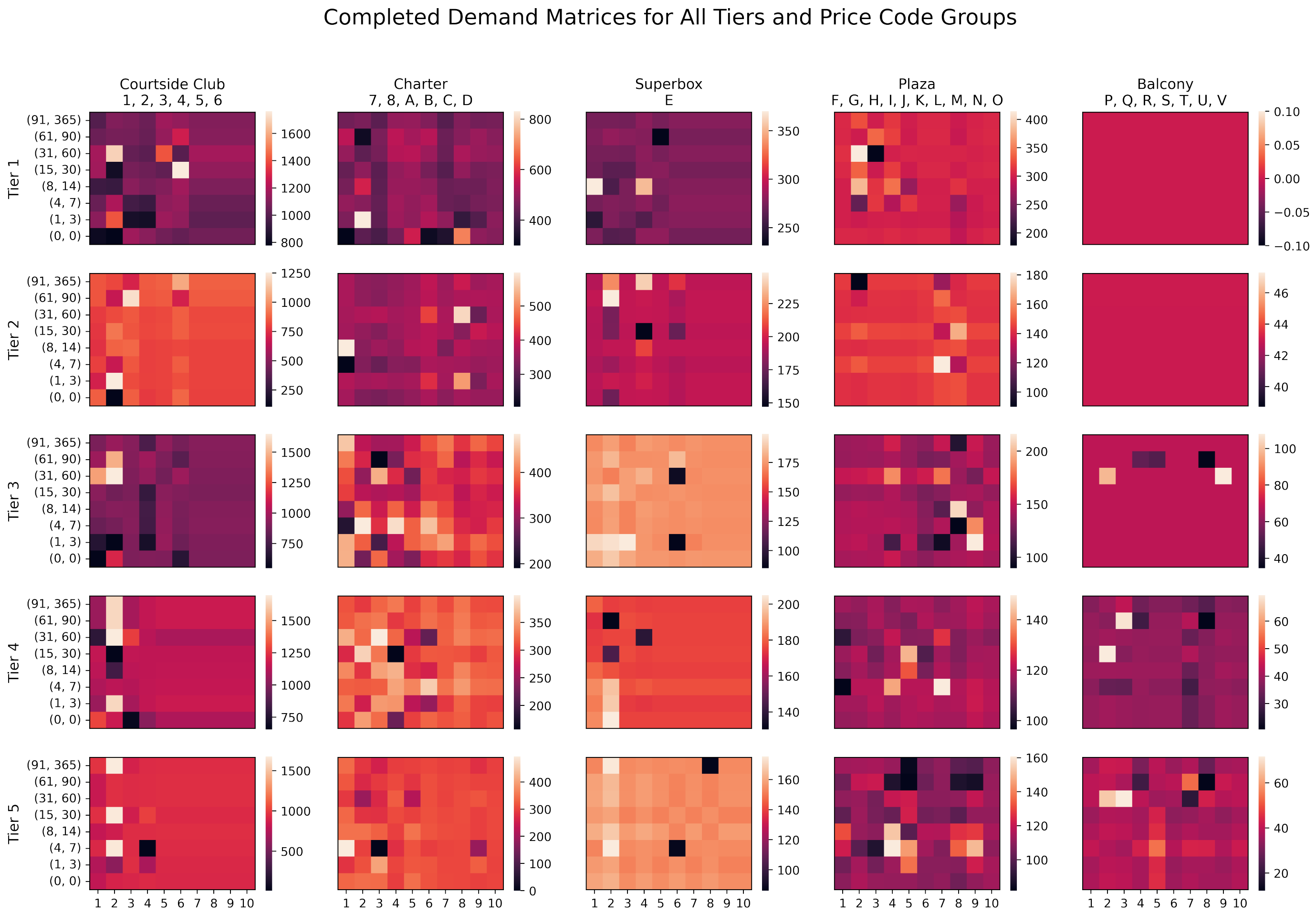


Figure B-1: A sample collection of completed Demand Matrices for all combinations of price codes and price code groups, completed using the SVD method. Along the major x-axis are the different price code groups (which decrease in quality from left to right) and along the major y-axis are the different game quality tiers (which decrease in quality from top to bottom). The minor axes are the same as in a standard Demand Matrix, like the one shown in Figure 4-8: the y-axis represents different time periods, while the x-axis represents different ticket quantities. In the case of cells that are now filled compared to 4-9, those prices were inferred using the SVD matrix completion method.

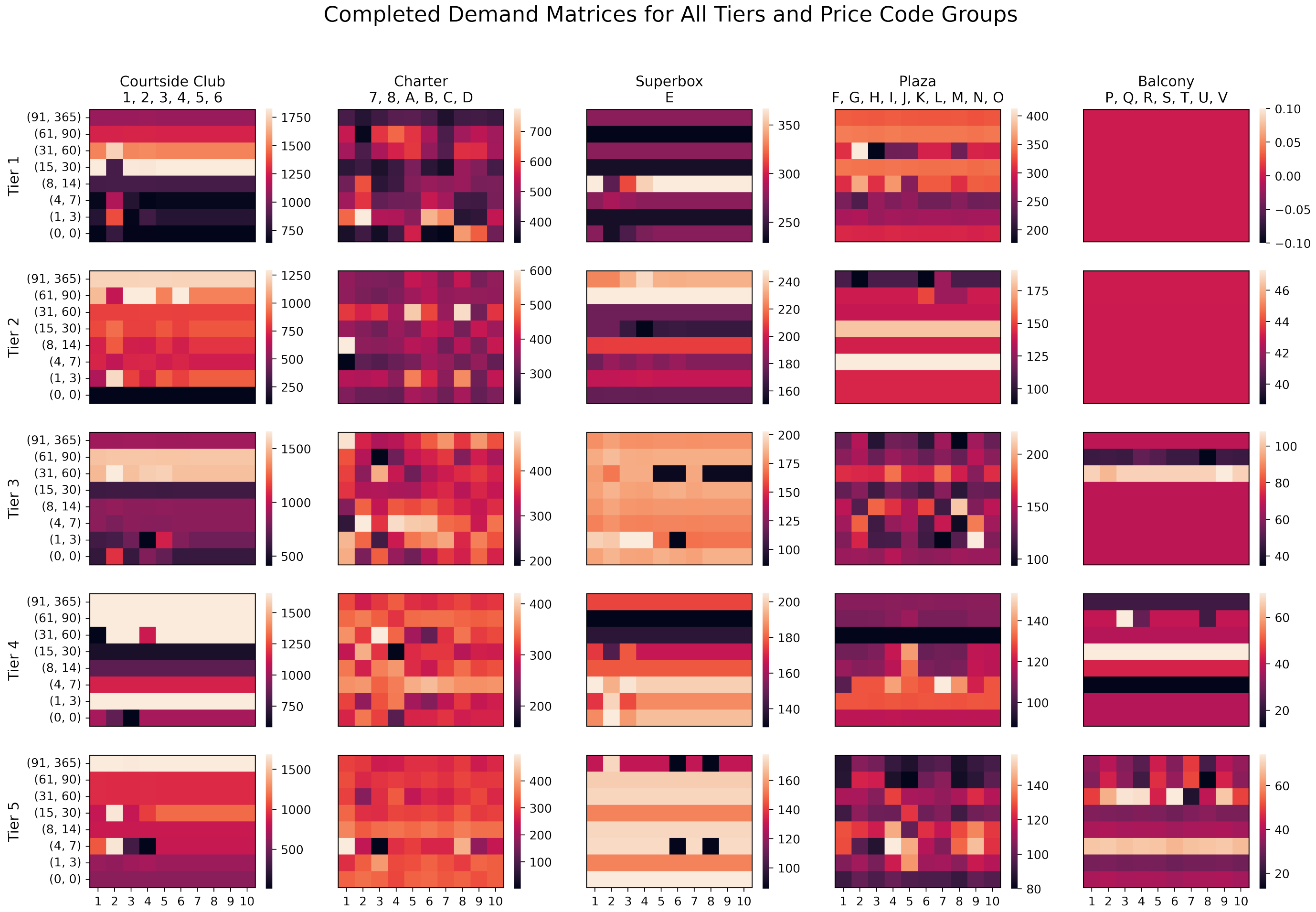


Figure B-2: A sample collection of completed Demand Matrices for all combinations of price codes and price code groups, completed using the hybrid method. Along the major x-axis are the different price code groups (which decrease in quality from left to right) and along the major y-axis are the different game quality tiers (which decrease in quality from top to bottom). The minor axes are the same as in a standard Demand Matrix, like the one shown in Figure 4-8: the y-axis represents different time periods, while the x-axis represents different ticket quantities. In the case of cells that are now filled compared to 4-9, those prices were inferred using the hybrid matrix completion method.

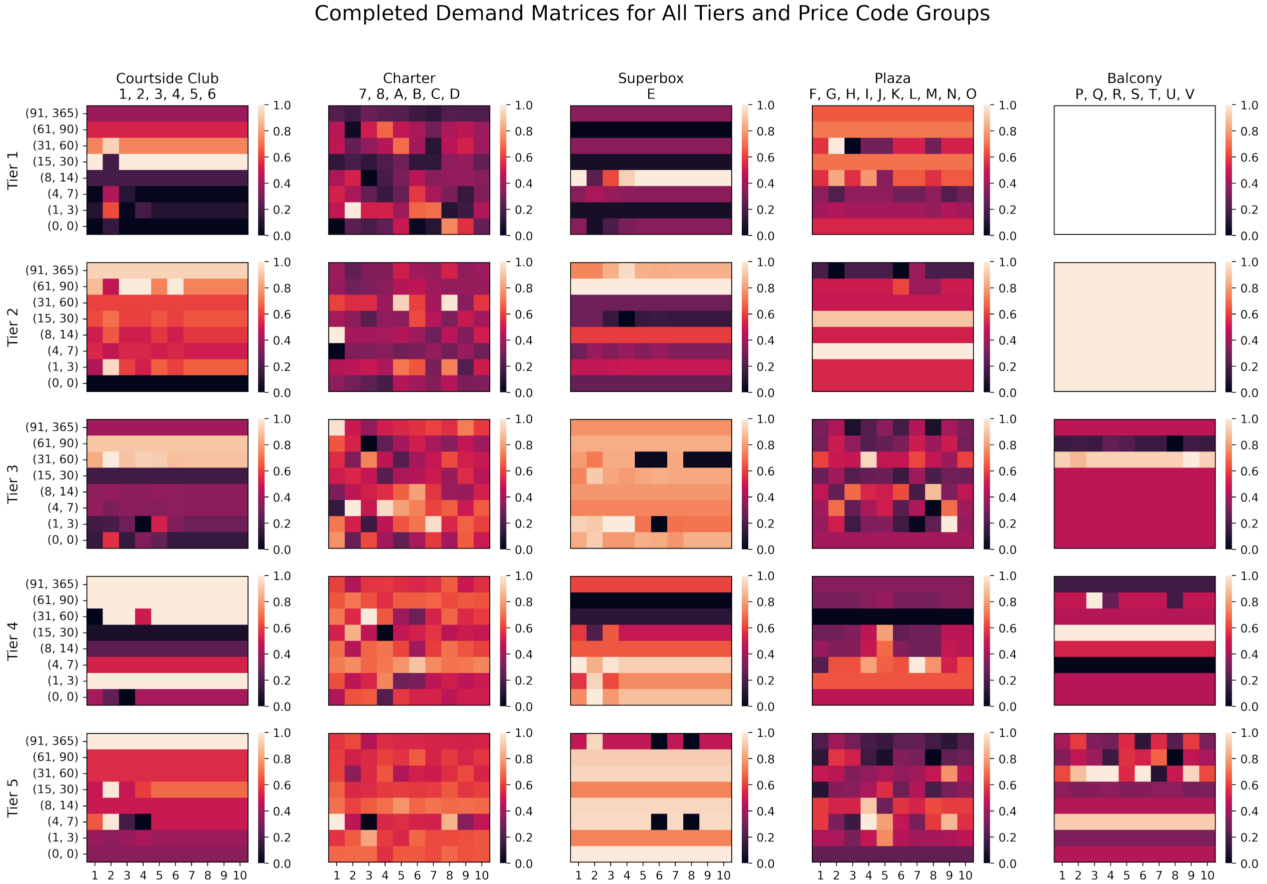


Figure B-3: A sample collection of completed Demand Matrices for all combinations of price codes and price code groups, completed using the NN method. These matrices have also been normalized on a 0-1 scale to allow for easier comparison of relative differences in price values. Along the major x-axis are the different price code groups (which decrease in quality from left to right) and along the major y-axis are the different game quality tiers (which decrease in quality from top to bottom). The minor axes are the same as in a standard Demand Matrix, like the one shown in Figure 4-8: the y-axis represents different time periods, while the x-axis represents different ticket quantities. In the case of cells that are now filled compared to 4-9, those prices were inferred using the NN matrix completion method.

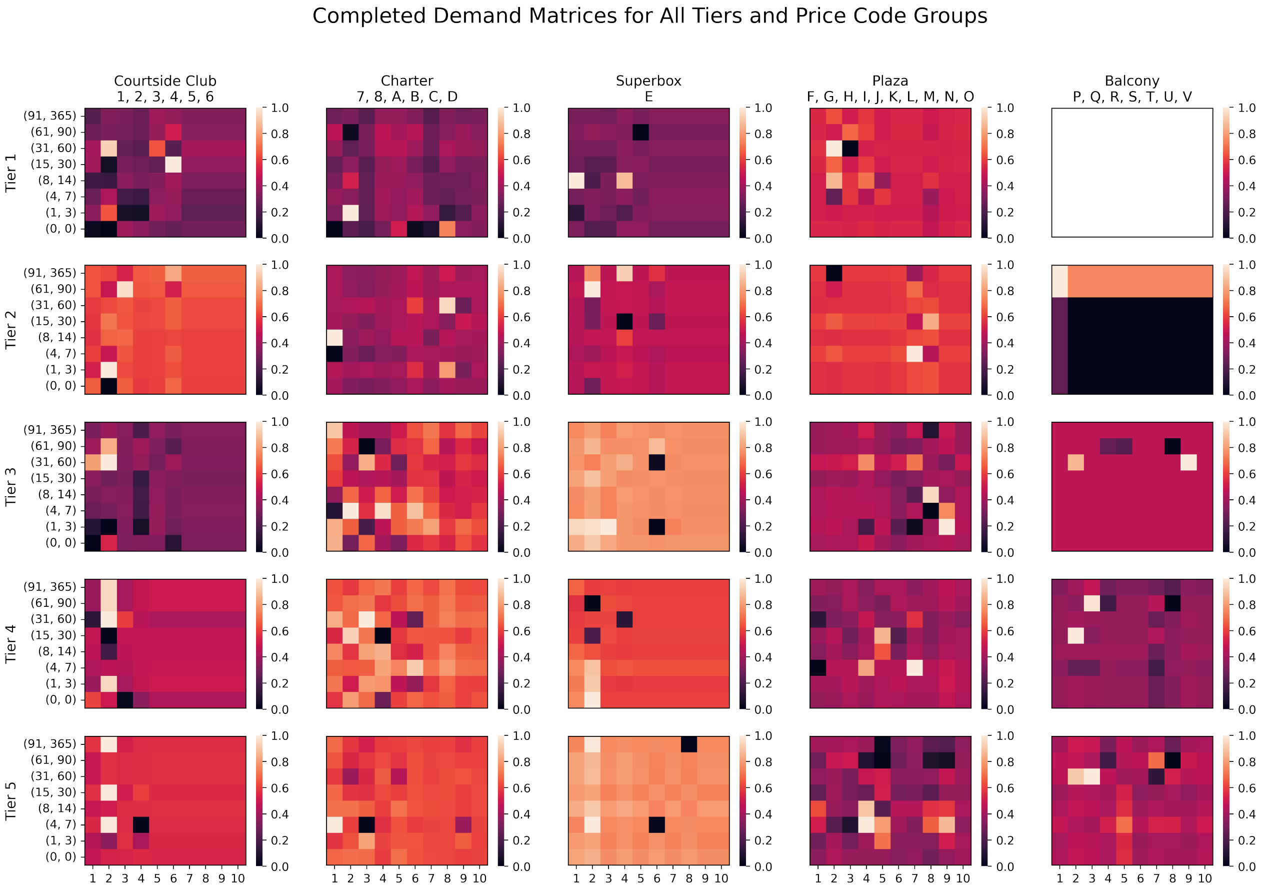


Figure B-4: A sample collection of completed Demand Matrices for all combinations of price codes and price code groups, completed using the SVD method. These matrices have also been normalized on a 0-1 scale to allow for easier comparison of relative differences in price values. Along the major x-axis are the different price code groups (which decrease in quality from left to right) and along the major y-axis are the different game quality tiers (which decrease in quality from top to bottom). The minor axes are the same as in a standard Demand Matrix, like the one shown in Figure 4-8: the y-axis represents different time periods, while the x-axis represents different ticket quantities. In the case of cells that are now filled compared to 4-9, those prices were inferred using the SVD matrix completion method.

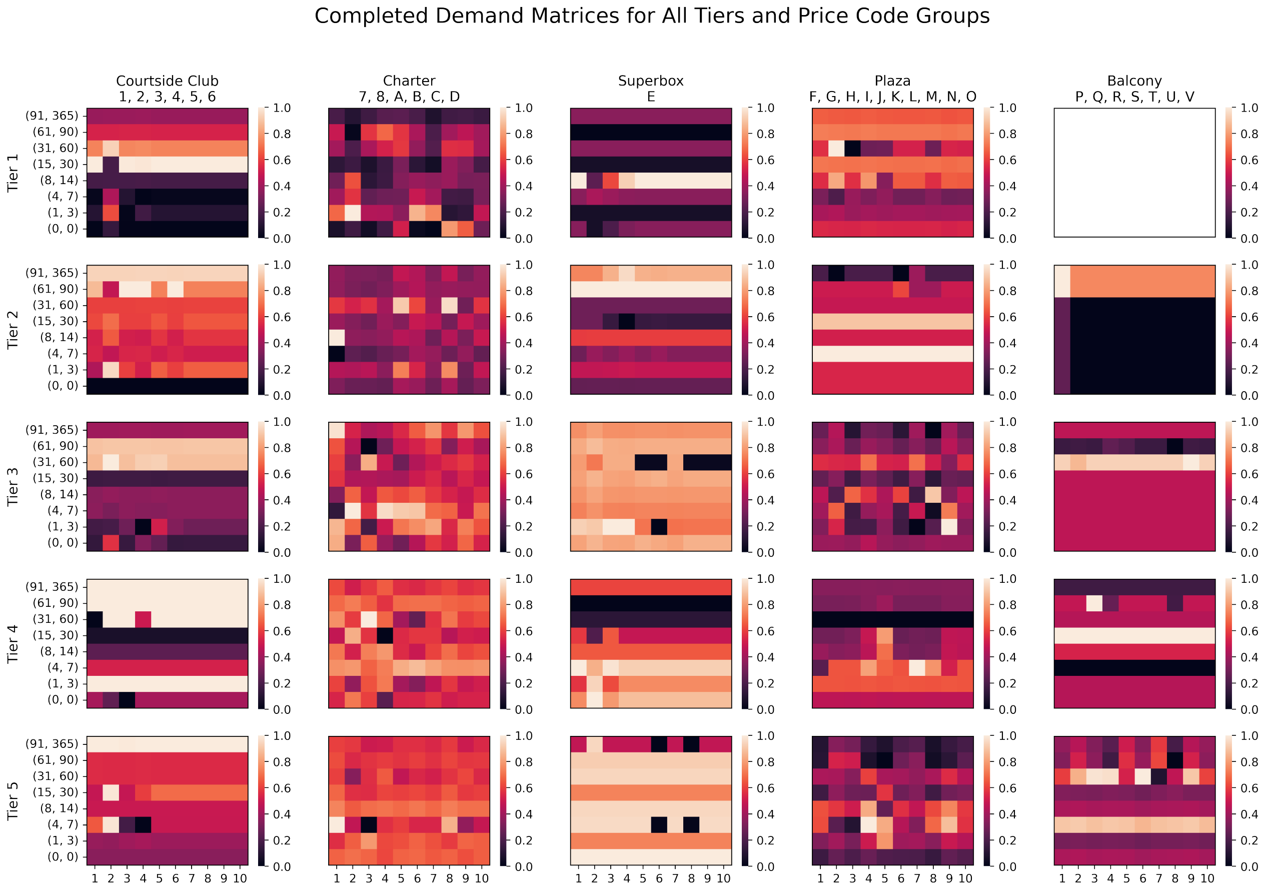


Figure B-5: A sample collection of completed Demand Matrices for all combinations of price codes and price code groups, completed using the hybrid method. These matrices have also been normalized on a 0-1 scale to allow for easier comparison of relative differences in price values. Along the major x-axis are the different price code groups (which decrease in quality from left to right) and along the major y-axis are the different game quality tiers (which decrease in quality from top to bottom). The minor axes are the same as in a standard Demand Matrix, like the one shown in Figure 4-8: the y-axis represents different time periods, while the x-axis represents different ticket quantities. In the case of cells that are now filled compared to 4-9, those prices were inferred using the hybrid matrix completion method.

# Bibliography

[1] M. Alley, M. Biggs, R. Hariss, C. Herrmann, M. Li, and G. Perakis. Pricing for heterogeneous products: Analytics for ticket reselling. 2019.

[2] Reza Bagheri. Understanding singular value decomposition and its application in data science. Towards Data Science, January 2020.

[3] G.L. Barlow. Capacity management in the football industry. *Yield management: Strategies for the service industries*, pages 303–314, 2000.

[4] A. Bouchet, M. Troilo, and B.R. Walkup. Dynamic pricing usage in sports for revenue management. *Managerial Finance*, 42(9):913–921, 2016.

[5] F. Caro and J. Gallien. Clearance pricing optimization for a fast-fashion retailer. *Operations research*, 60(6):1404–1422, 2012.

[6] S. Duran, J.L. Swann, and E. Yakıcı. Dynamic switching times from season to single tickets in sports and entertainment. *Optimization Letters*, 6(6):1185–1206, 2012.

[7] K. J. Ferreira, B. H. A. Lee, and D. Simchi-Levi. Analytics for an online retailer: Demand forecasting and price optimization. *Manufacturing & Service Operations Management*, 18(1):69–88, 2016.

[8] Xu J Jiaqi, PS Fader, and S Veeraraghavan. Designing and evaluating dynamic pricing policies for major league baseball tickets. *Manufacturing & Service Operations Management*, 2019.

[9] C Kemper and C Breuer. How efficient is dynamic pricing for sport events? designing a dynamic pricing model for bayern munich. *International Journal of Sport Finance*, 11(1), 2016.

[10] Will Kenton. Demand curve. Investopedia, May 2021.

[11] D. Koushik, J. A. Higbie, and C. Eister. Retail price optimization at intercontinental hotels group. *Interfaces*, 42(1):45–57, 2012.

[12] P. Sainamm, S. Balasubramanian, and B.L. Bayus. Consumer options: Theory and an empirical application to a sports market. *Journal of Market Research*, 47(3):401–414, 2010.

[13] Cyrus Samii. Matrix completion. URL: https://cyrussamii.com/wp-content/uploads/2018/11/matrix-completion-3.html, November 2018.

[14] B. C. Smith, J. F. Leimkuhler, and R. M. Darrow. Yield management at american airlines. *Interfaces*, 22(1):8–31, 1992.

[15] S. A. Smith and D. D. Achabal. Clearance pricing and inventory policies for retail chains. *Management Science*, 44(3):285–300, 1998.

[16] Statista. Percentage of ticketing revenue in the mlb since 2006.

[17] Statista. Percentage of ticketing revenue in the nba since 2006.

[18] Statista. Percentage of ticketing revenue in the nfl since 2006.

[19] Statista. Revenue passengers united airlines.

[20] A Sweeting. Dynamic pricing behavior in perishable goods markets: Evidence from secondary markets for major league baseball tickets. *Journal of Political Economy*, 120(6):1133–1172, 2012.

[21] Wikipedia. List of national basketball association arenas.

[22] JD Zhu. Effect of resale on optimal ticket pricing: Evidence from major league baseball tickets. Technical report, Working paper, Texas A&M University.