AN EXERCISE IN SHRINKAGE AND BIASED ESTIMATORS- TENNIS MATCH ANALYSIS

Vasco Villas-Boas

May 19, 2022

I model a tennis match as a Markov model with terminal states: Player A wins or Player B wins. At match t, say Player A wins a serve against Player B with probability s_{ABt} and Player B wins a serve against Player A with probability s_{BAt}. For simplicity of the model, I assume each point in a match is independent and identically distributed (iid) with these probabilities given the server. I ultimately compute the probability that Player A or B wins (ie., the probabilitiy we reach each terminal state) given these serve win probabilities. I also take the tennis match at other levels of granularity such as the game, set, and/ or tiebreak, with analogous, simplifying iid assumptions, to construct and solve analogous Markov chains for the tennis match. I look at data from the men's ATP tour from 2011-2015, to analyze how well this Markov model of tennis helps us predict matches in a backtesting approach. The main question that my article addresses is how to effectively compute s_{ABt} and s_{BAt} to minimize error in our match predictions.

I consider the innocent approach of estimating $s_{ABt} := \frac{\textit{Player A number of serves wins in past year}}{\textit{Player A number of serves total in past year}}$. While this approach gives an unbiased estimate under an iid serve assumption in the spirit of expectation, I argue that it can be improved by accounting for some potential sources of error. (1) Player B may be a very strong or weak returner. (2) Different players face different strengths of schedules in the past year, (3) Some players play more matches than others and thus serve more times. (4) Playing more matches is typically correlated with being a better player since better players win more matches and advance in tournaments whereas weak players lose early on. (5) Different matches are played on different surfaces. To combat these sources of bias, I utilize use a useful concept from other sports, 'strenth-of schedule', and focus on implementing various shrinkage estimators such us the James-Stein estimator, T-Oracle estimator, and beta binomial regressions. For my own practice and understanding, I include key derivations and intuition when utilizing these tools. I compare the efficacy and trade-offs of various shrinkage estimators against a complex dataset in the setting of sports.

I also consider an alternate approach of predicting the tennis match using logistic regression. I find success implementing Su-In Lee, Honglak Lee, Pieter Abbeel and Andrew Y. Ng's "Efficient L_1 Regularized Logistic Regression" algorithm, about the same level of success as using built in logistic regression packages. Their algorithm solves the L_1 regularized logistic regression problem by iteratively solving strategically formulated L_1 regularized least squares problems and its implementation served as good practice for me.

1 THE SPORT OF TENNIS AND ITS SCORING SYSTEM

Singles tennis is a sport played on a rectangular court between two adversaries that stand on opposite sides of a central net[1]. In a single point, one player serves to the other and then the opponents trade shots until one is unable to make a valid return and thus loses the point.

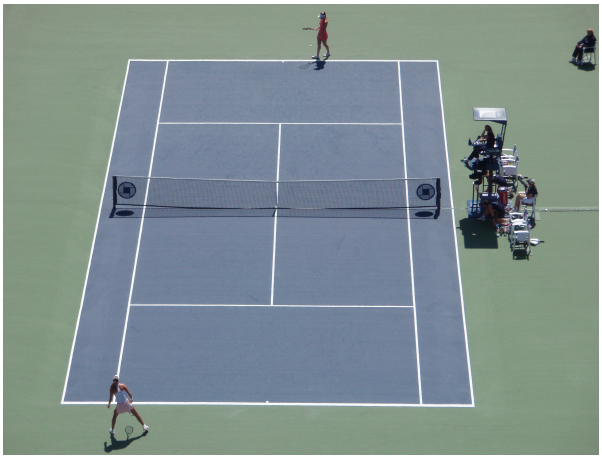


Figure 1. Peer serving to Chakvetadze at the start of a tennis point

A tennis match between Player A and Player B is divided into four entities: points, games, sets, and match. A player wins points to win a game, they win games to win a set, and they win sets to win a match.

Games start at (Player A points, Player B points): (0,0), with one player serving all points in a game. For each point won, a player advances in the sequence (0, 15, 30, 40) (this is the traditional scoring system in tennis)[2]. As an example, if Player A wins the first point, then Player B wins a point, then Player A wins a point, then Player B wins a point, and then Player B wins a point, the sequence of scores is: (0,0), (15,0), (15,15), (30,15), (30,30), (30,40). If a player wins a point when they have a score of 40, they win the game. The only exception is if the score is (40,40), then we enter deuce. At this point, a player must win two points in a row without the other winning a point to win the game. If with a score of deuce Player A wins one point, we say that Player A has the advantage. If Player B wins the subsequent point, the score returns to deuce. After each game, the game score resets to (0,0) for the next game and it's the other player's turn to serve.

The first player to win 6 games with a margin of two games from the other player, wins the set. In other words, with the game score (Player A games, Player B games): (6,4) Player A has won the set, but with the game score (6,5), Player A has not won the set.

The major tennis tournaments, until this year, had different rules with how to deal with a game score of (6,6). For example, in the US Open, if any set reached (6,6) in games, we would enter a 7-point

[1]Note that there also exists doubles tennis, played on a marginally larger court between pairs of adversaries.

[2]The origins of this seemingly strange counting system are unknown. The sport of tennis originates from *jeu de paume* (translates to *game of palm*) in 12th century France, where the sport was played with hands (players started utilizing rackets in the 16th century). In these early days, the scoring sequence was very similar (0,15,30,45). Historians speculate that the sequence mimics minutes of an hour but they're uncertain. We do know that the scoring system (0,15,30,40) has remained stable since the Victorian Era of England (1840-1900).

tiebreak (to be explained), a "shortened" approach to decide the set. In the Wimbledon up until 2019, the final set in a men's match would never enter tiebreak which meant that a player had to win a set by winning at least 6 games with a margin of two over the opponent[3]. Just this year in March, the four major tournaments agreed to have a decisive ten point tiebreak if the score reaches (6,6) in the final set.

Figure 2. Isner and Mahut after their 8 hour 11 minute Wimbledon match

In a N-point tiebreak, the first player to win at least N points, with a margin of at least two points over the opponent, wins the tiebreak (and set). One player serves first and then the other player serves twice and they keep alternating two serves each until one wins the tiebreak. As a result, the winner of the tiebreak must win at least one point on the opponent's serve.

Many tournaments have different rules on how many sets a player needs to win to win a match. In some tournaments, it's best-five-set matches (ie., the first player to win 3 sets wins the match), in others, it's best-of-three-set matches (ie., the first player to win 2 sets wins the match). In either case, it's not necessary to win by at least two sets over the other player as in other parts of the match.

1.1 Modeling the Tennis Scoring System as a Markov Chain

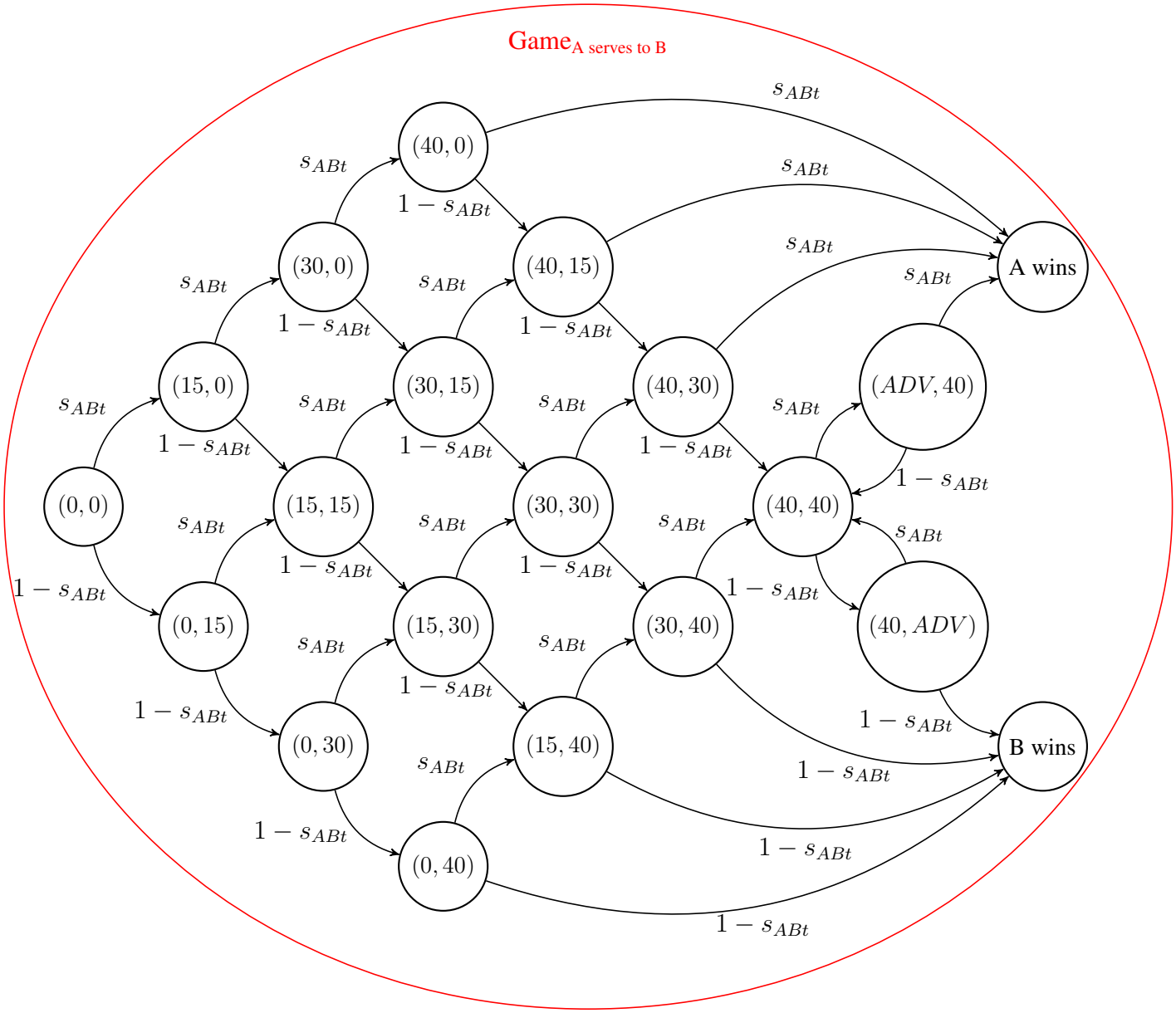
In this subsection, I'll explain how I aim to model the a match of tennis as a Markov chain. Consider a match between Player A and Player B. At any point in the match, we can read the score of the match as (number of sets A has won, number of games A has won in the current set, number of points A has won in the current game, number of sets B has won, number of games B has won in the current set, number of points B has won in the current game, current server). We can take the current score as states in a chain and define transition probabilities as the probability that A (or B) wins the next point, game, tiebreak, or set. I provide separate Markov chains at the game, set, and match levels due to visualization limitations but I think it's fairly intuitive how these chains can be combined, while carefully keeping track of the current server, to properly model a full tennis match.

[3] In the most extreme case, a Wimbledon 2010 first round match between John Isner and Nicolas Mahut had a fifth set reach a winning score of (70,68) with Isner winning the match. That set lasted 8 hours and 11 minutes! See figure 2 for a photo of delighted Isner and unhappy Mahut after the match.

1.1.1 Game-view Markov Chain

At match t, say Player A wins a serve against Player B with probability s_{ABt} and Player B wins a serve against Player A with probability s_{BAt}, where each serve is independent and identically distributed (iid)[4] given the server. In a game where A serves to B, I create a graph in which nodes represent a possible score attained in the game written as (A's score, B's score). There are two terminal nodes, "A wins" or "B wins" since either one player or the other wins the game. With our assumption that each time A serves to B, A wins with probability s_{ABt}, where each serve is iid, I can label transition probabilities between nodes as s_{ABt} or $1 - s_{ABt}$ depending on if the transition constitutes A winning a point or B. I can then numerically compute the probability that we reach node "A wins" (or node "B wins") from node (0,0) and use that as our estimate that A (or B wins the game).

[4]The Markov assumption is that transition probabilities be depend only on the current state (not on the past), transition probabilities do not need to be iid given the server as I've imposed. I make this assumption for simplicity of the model and implementation. See section 1.1.4 for a more complete discussion of the assumptions of the Markov model and the assumptions I've superimposed.

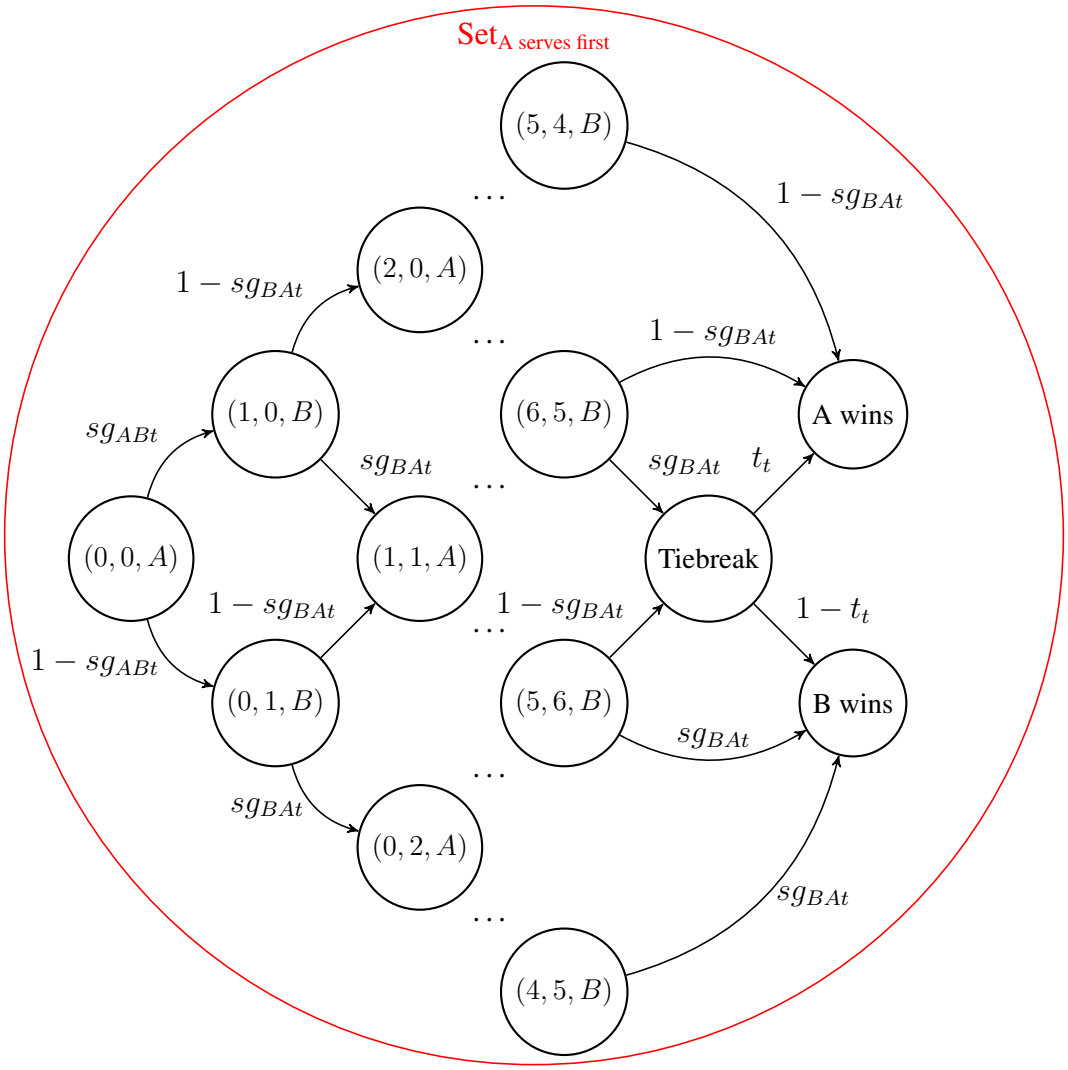


1.1.2 Set-view Markov Chain

Similar to the game level, I can construct a Markov chain for the set. I create nodes that represent possible scores in the set: (num of games A has won in the set, num of games B has won in the set, server). I add in transition probabilities that reflect when A wins the game or B wins and I add in two terminal nodes "A wins" or "B wins", that are states where A has won the set or B. At match t, say Player A wins a service game against Player B with probability sg_{ABt} and Player B wins a service game against Player A with probability sg_{BAt}, where each service game is independent and identically distributed (iid) with its respective probability depending on the server[5]. If I choose to discretize the game at the point level, I can use the computed probability in Game$_{\text{A serves to B}}$ that A wins the game to assign transition probabilities from states where A serves, vis-a-vis for Player B. Alternatively, I can discretize the set at the game level and take sg_{ABt} and sg_{BAt} as parameters to

[5]Again, I impose this iid assumption to simplify the model and its implementation.

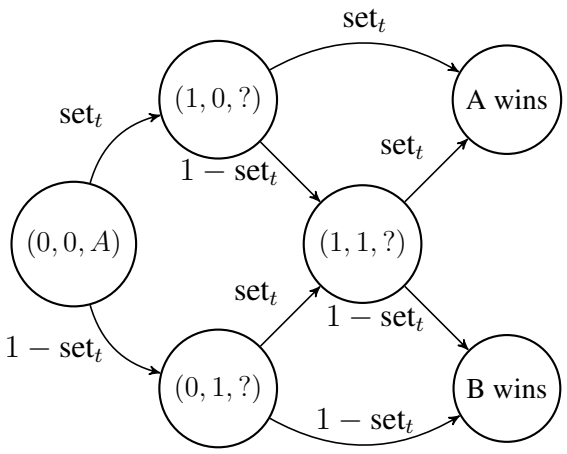
the model. Lastly, I need to take special care for the tiebreak at the end of the set. At match t, say Player A wins a tiebreak against Player B with probability t_t, where each tiebreak is independent and identically distributed (iid). I can take t_t as a native parameter to the model or use s_{ABt} and s_{BAt} to numerically compute t_t. Finally, given my transition probabilities, I can solve for the probability that each player wins the set by finding the probability of reaching each terminal state from $(0, 0, A)$.



1.1.3 Match-view Markov Chain

Lastly, I can also construct an analogous Markov chain for the match where nodes are the score in sets: (Num of sets A has won, Num of sets B has won, first server of set). At match t, say Player A wins a set against Player B with probability set_t, where each set is independent and identically distributed (iid). I add in transition probabilities according to the probability that each player wins a set. I also add in two terminal nodes, "A wins" and "B wins" that indicate Player A or B has won the match, respectively. Similar to above, set_t can be made an exogenous parameter to the model or we can compute it using the $\text{Set}_{\text{A serves first}}$, $\text{Set}_{\text{B serves first}}$ Markov chains. I draw the relatively small Markov chain for a best-of-three-set match. The question marks for the servers reflect that at this granularity, we don't know the final server in the each set and so we don't know who serves first in the sets following the first. While a problem here, this issue doesn't arise in a full model of the tennis

match since we would just have separate states for either server and add proper transitions.



1.1.4 Costs and Benefits of the Markov Model

Since the sport of tennis progresses in discrete points, games, and sets, I can fairly easily model a match as a state-based system where states are possible scores in the match. In constructing a Markov model from these states, we inherently undertake the Markov assumption: we assume that that all transitions probabilities from a state depend only on the current state. I take a further assumption in my analysis than the Markov Model requisites for simplicity. I assume that all points served in a match by a server are identically distributed. Are these good assumptions? Likely no. In a match between A and B, even if A has won the past 10 points, I assume that doesn't affect the likelihood they win the next point. The "hot-hand principle", the occurrences of streaks in "random" sequences, has been fervently debated in papers such as Gilovich, Vallone, and Tversky (1985), arguing that it is a fallacy and Sanjurju and Miller (2018), in a beautiful result, arguing that GVT's selection of streaks to examine is biased, invalidating their work. In the lens of tennis, Klaassen and Magnus (2001a) finds that tennis points are not independent and identically distributed[6]. For example, key points in the match such as "match point", are less likely to be won by the server than a typical point. Nevertheless, they find that deviations from iid points are typically small.

A simple alternative to my further assumption (that still satisfies the Markov assumption) is to make the serve win probability indexed to the score in the current game like in Matteazzi and Lisi (2017). For example, we could define $s_{ABt(0,0)}$, $s_{ABt(15,0)}$, etc., as the probability that at match t, player A wins a service point on player B when the score is (0,0) in the current game, (15,0) in the current game, respectively, etc. Such indexed service point win probabilities would also partly help control for streaks since a score of $(40, 0)$ necessarily means that the server won the past three points. There are many other ways to define states of the tennis match to take into account the past few points, games, and sets to generalize the iid points assumption- these extensions are out of the scope of this article.

[6]They conclude that winning the previous point has a positive effect on winning the current point. Though, since the authors wrote their paper in 2001 and the streak selection bias found by SM happened in 2018, KM's paper doesn't account for this bias. As explained in the appendix, this bias causes one to underestimate the presence of the "hot-hand" phenomenon. Thus, SM's implication only exacerbates KM's finding that a player winning sequential points are correlated events

There are a few more nice applications of the Markov model. First, as discussed in Klaassen and Magnus (2001b), we can efficiently solve for the probability of winning the match at any given state by analytically solving any cycles in the chain (ie., deuce), and then using dynamic programming to find the win probability from any state. Discretizing the match at the point level and given serve win probabilities s_{ABt} and s_{BAt}, and scoring rules for the match (ie., number of sets needed to win the match, etc.), a memoized function takes the following inputs.

```
# Global parameters
s_ABt = 0.8 # probability A wins a service point on B
s_BAt = 0.75 # probability B wins a service point on A
sets_to_win = 2 # number of sets a player needs to win to win the match
games_to_win = 6 # number of games a player needs to win to win the set
points_to_win = 4 # number of points a player needs to win to win the game
tiebreak_total = 7 # number of points a player needs to win to win a tiebreak

def compute_playerA_win_prbability(A_sets, B_sets, A_games, B_games, A_points,
    B_points, server):
    """ Return the probability that A wins the match from the current score and
        global parameters.

    Args:
    - A_sets: number of sets A has won
    - A_games: number of games A has won in the current set
    - A_points: number of points A has won in the current game/ tiebreak
    - B_sets: number of sets B has won
    - B_games: number of games B has won in the current set
    - B_points: number of points B has won in the current game
    - server: the server of the next point

    Return Value:
    - int in [0,1]: the probability A wins the match from the current score
    """
```

Also explained in this KM (2001b) paper, the Markov chain approach allows us to ponder changes to the tennis scoring rules (ie., sets are first to 8 games, no tiebreak) and how they impact the win probability of the match. Lastly as discussed in Gollub (2017), the Markov model allows us to compute live in-match win probabilities by computing the match win probability for a player from any score reached in the match.

1.2 Tennis Match Data

In my analysis, I take on the same dataset that's used in Gollub (2017): all men's ATP tour singles matches from 2011-2015 located in a gentleman Jack Sackmann's github (Sackmann 2017). That is, 10902 total matches! In this dataset, for each match we have the following information.

- pbp_id: unique id for the match
- date: date for the match
- tny_name: tournament to which the match belongs
- tour: tour to which the match belongs (ATP for all)
- draw: Main or qualifying for the tour (Main for all)
- server1: first server in the match

- server2: second server in the match
- winner: player that won the match
- pbp: point by point information for how the match progressed
- score: final score in the match
- adf_flag: 1 if the pbp includes information about aces and double faults, else 0
- wh_minutes: duration of match in minutes

When processing the data, I encountered some unintended difficulties. First, pbp_id's appeared multiple times in the dataset in which case I simply deleted duplicates. Some matches appeared multiple times under different ids and mis-spelled player names or slightly different tournament names. In these cases, I'd query the dataset by the 'pbp' column, a quasi-unique id for each match and manually inspected to delete repeated matches. Some matches had players names that were clearly not names (ie., they contained characters such as '/') and I simply deleted these matches from analysis. Other matches had players listed under fullnames or nicknames (ie., Stanislas Wawrinka vs Stan Wawrinka) and so I inspected and made changes to keep player names consistent. Lastly, tournaments used slightly different names in different rows: in a simple example, 'USMensClayCourtChampionship-ATPHouston2012' versus 'USMensClayCourtChampionship-ATPHouston'. It's clear these tournament names correspond to the same tournament but a simple join by tournament name would not work. I resolved the issue by removing special characters from tournament names and replacing the 'tny_name' with it's six letter prefix to aggregate matches by tournament. I followed up with manual inspection to resolve aggregation for a few extra challenging tournament names.

The 'pbp' column for each match is a string that contains information about the winner and loser of each point in the match. I provide a sample 'pbp' for the match between Brian Baker and Radek Stepanek on October 23, 2012 at the Swiss Indoors Basel Tournament.

```
# RRRR;RSRSRSARSS;SSRSS;ARSRRSRSSS;SSSS;SRSSRS;RRRSR;ARRRSASS.RSRRR;ADASS;
   RSSSRRSRRR;RRRR;SSSA;SSSS;SSSS;SSSRRRRARR;DSSSA;ASSRS;RSSSRS;RSSSS;R/SS/SS/RR
   /SS/SA/S.SSRDRSARSS;RSRSSS;SSRAS;SSSRA;RRSRSSSRRR;SSDRRR;SRRRR;SRDRSSSS;
   RSDRSSRR
```

'R' corresponds to points one by the returner in play, 'S' corresponds to points won by the server in play, 'A' corresponds to an ace by the server, 'D' corresponds to a double fault by the server, ';' corresponds to the end of a game, '.' corresponds to the end of a set, and '/' corresponds to change of server in a tiebreak. With knowledge of the first server, one can carefully iterate through the string, keeping track of the current score at each point and the current server, to calculate service point win totals, service game win totals, return point win totals, return game win totals, set point win totals, and tiebreak win totals for both players in the match.

1.3 Previous Works in the Sports Analytics Space

In the lens of tennis, the first paper (I could find) that models tennis matches as a Markov Model is Klaassen and Magnus (2001b). They noticed that one could create states for each score in the match and assign transition probabilities between states according to the likelihood that a player wins the next point. Matteazzi and Lisi (2017) builds off of this work by indexing transition probabilities to different features of the score in the current game. In trying to estimate games using the Markov model, Clarke and Benner (2005) is the first paper that takes a stab at estimating s_{ABt} and s_{BAt} for

this purpose. The authors linearly benchmark a players serve to the tournament average. Gollub (2017) follows up on this paper by correcting s_{ABt} for opponent strength of schedule and employing the James-Stein estimator to correct for number of matches played. My paper most closely follows up to Gollub (2017) in that I use the same dataset and hence can follow his guidance on common pitfalls in working with the data. I extend his work by trying a variety of shrinkage estimators to more precisely account for number of matches played. I also extend previous work by taking the tennis match at different levels of granularity besides the point level and predicting the match according to the corresponding Markov chains.

With regards to shrinkage estimators, James and Stein (1961) first introduced the James-Stein estimator for the mean of a multivariate normal distribution that has lower mean squared error than the sample mean. Efron and Morris (1975) applied the estimator to the setting of baseball to estimate batting averages for various players. Brown (2008) applied many shrinkage estimators to predict baseball averages in this same setting. Rasmusen (2021) provides an introduction of bias in estimators that serves as additional key inspiration for the choices I make in my work.

2 ESTIMATING s_{ABt}, s_{BAt}

Given my model of the tennis match, the challenge becomes effectively estimating s_{ABt}, s_{BAt} based off of data from previous matches so as to effectively predict future matches using the model. For each match in the dataset, I look at all matches in the year prior to the match of interest. What I'll call the naive approach, is to estimate s_{ABt} as

$$\hat{s}_{ABt(\text{naive})} := \frac{\text{number of serves A won in the past year}}{\text{number of serves total by A in the past year}} \tag{1}$$

I consider this approach naive because we're not accounting for some things: (1) Player B may be a very strong or weak returner. (2) Different players face different strengths of schedules in the past year. (3) Some players play more matches than others and thus have more service attempts- (4) playing more matches is typically correlated with being a better player since better players win more matches and advance in tournaments whereas weak players lose early on. (5) Different matches are played on different surfaces.

2.1 Some Notation

Let's introduce some notation here. Restricting the dataset to the 1 year period before and up to t, for game (A, B, t) between players A, B happening at time and tournament t, define:

- $swr_{At} := \frac{\text{number of serves A won in the past year}}{\text{number of serves total by A in the past year}}$
- $rwr_{At} := \frac{\text{number of returns A won in the past year}}{\text{number of returns total by A in the past year}}$
- $s_{At} := \Pr[\text{A wins a serve versus an average opponent at time and match t}]$
- $r_{At} := \Pr[\text{A wins a return versus an average opponent at time and match t}]$
- $s_{ABt} := \Pr[\text{A wins a serve versus B at time and match t}]$
- $s_{BAt} := \Pr[\text{B wins a serve versus A at time and match t}]$

2.2 Accounting for the Opponent's Return Ability

Let's take a sidebar to understand how I'll account for the opponent's return ability. Suppose it's the start of a tennis season. There are two tennis players (i, j) and they both talk smack that they will be the better player at the end of the season and schedule a match for that time. For sake of argument, suppose that the tennis season consists of a league of N players and they all play each other twice so that they face the same strength of schedule. The tennis smack talk becomes so contentious that at the end of the season, it's unsafe for the two to play. Nonetheless they want to know who is the better player. Player k jumps in, he was an average player with a winning percentage of 50% over the season, and proposes that both i and j play versus k with the following rules:

- If i beats k and k beats j, i is declared the better player.

- If j beats k and k beats i, j is declared the better player.

- If (i beats k and j beats k) or (k beats i and k beats j), they do the matches again.

Suppose all matches are independent and player i posed a winning percentage of WP_i and player j posed a winning percentage of WP_j over the long season. Then, since k has a winning percentage of 50%, that means:

$$\Pr[\text{i beats k}] = WP_i, \;\; \Pr[\text{j beats k}] = WP_j$$

since player i posed a winning percentage of WP_i versus a group that has an average winning percentage of 50%[7] and player k has a winning percentage of 50%, vis-a-vis for player j. In other words, while WP_i is a measure of the percentage of games A won over the season, it's also a measure of how likely i is to beat an average player in the league.

As a result, we can say

$$\Pr[\text{A declared better player}] = \frac{(WP_i)(1 - WP_j)}{(WP_i)(1 - WP_j) + (1 - WP_i)(WP_j)} \tag{2}$$

Now let's move back to the task of estimating s_{ABt}. Let s_{At} be the serve win percentage of A over a "representative" set of opponents at this tournament. Similarly, let r_{Bt} be the return win percentage of B against a "representative" set of opponents at this tournament. s_{At} and r_{Bt} look analogous to WP_A and WP_B since they are all equal to the probability of "success" against an average opponent (where success is aptly defined in each case). As a result, I define

$$s_{ABt} := \Pr[\text{A wins serve against B on day and tourney t}] = \frac{(s_{At})(1 - r_{Bt})}{(s_{At})(1 - r_{Bt}) + (r_{Bt})(1 - s_{At})} \tag{3}$$

I have found a nice way to account for the return ability of the opponent in computing the probability that A wins a serve on B. But, I now have the task of appropriately estimating s_{At} and r_{Bt}.

[7] Technically i probably didn't face an average winning percentage of 50% over the season. It's only true that $\frac{1}{N}\sum_{l=0}^{N} WP_l = 50\%$. In fact i faced an average winning percentage of $\frac{1}{N-1}\sum_{l=0, l\neq i}^{N} WP_l$. Thus, if i had a winning percentage above 50%, he actually faced a winning percentage below 50% over the season. However, for large enough N, 50% is a reasonable approximation of the winning percentage he faced.

2.3 Correcting for Strength-of-Schedules

As a disclaimer, this section and the next (section 2.4) are quite ad-hoc. I add and subtract probabilities from different probability spaces to produce probabilities- that's very mathematically illegal. On top of that, it's bad that some of the probabilities I produce can live outside of [0,1] and so I need to do some boundary clipping. I'm imposing myself a lot in determining how strength-of-schedule faced by player and the tournament at which the match is played affects serve win rate. Nevertheless, I consider the the spirit of the approaches reasonable and therefore implemented them. I also have an equally sized set of results where I do without these "corrections" and directly feed $swr_{At}, rwr_{At}, swr_{Bt}, rwr_{Bt}$ into equation 3 to calculate my transition parameters (serve win rates) to my hierarchical Markov model of the tennis match.

For game (A, B, t) between players A, B happening at time and tournament t, define:

$$SOSS_{Bt} := \sum_{j \in opp(B)} (\frac{\text{points won on serve by j}}{\text{total points served by j}}) * (\frac{\text{points A returned on j}}{\text{total points A returned}}) \tag{4}$$

$$SOSR_{At} := \sum_{j \in opp(A)} (\frac{\text{points won on return by j}}{\text{total points returned by j}}) * (\frac{\text{points A served on j}}{\text{total points A served}}) \tag{5}$$

Also define:

$$srel_{At} := swr_{At} - (1 - SOSR_{At}) \tag{6}$$

$$rrel_{Bt} := rwr_{Bt} - (1 - SOSS_{Bt}) \tag{7}$$

where we restrict dataset to the 1 year period before and up to t.

$SOSS_{Bt}$ intends to capture the strength of servers B faced in the past year prior to t. To gain intuition, when computing $SOSS_{Bt}$, we're taking a weighted average of the serve win rates of opponents that B faced in the last year. We weight each opponent's serve win rate by the fraction of B's serves that were against that opponent. Vis-a-vis for $SOSR_{At}$

The quantity $rrel_{At}$ is also an interesting quantity to look at: it is a measure of how much better or worse A returned versus it's set of opponents than the average player returned versus that same set of opponents. To see that, note that $1 - SOSS_{At}$, measures how well returners did on average versus A opponents, weighted by how many many times A returned versus those opponents. Analogously, $srel_{Bt}$ measures how much better or worse B served versus it's set of opponents than the average player served versus that same set of opponents.

2.3.1 An Example with Real Players

Here's abbreviated information about a match between Roger Federer and Kei Nishikori on November 19, 2015 at the ATP London Championships.

Player	swr	rwr	$SOSS$	$SOSR$	$srel$	$rrel$
R. Federer	0.722430	0.397627	0.659548	0.329475	0.051906	0.057174
K. Nishikori	0.673212	0.389979	0.657674	0.342384	0.015596	0.047653

Table 1. Pre-match stats for Federer vs Nishikori at ATP London Championships in 2015

Holistically looking at Table 1, Federer, one of the greatest of all time[8], appears to be the favorite. He had a better raw serve win rate and return win rate than Nishikori. Over the course of the season, they both faced relatively similar strengths of servers and returners. Federer faired better as a server and returner than Nishikori when you correct for strength of servers and returners faced by both players. One remark is that Nishikori is famously a very strong returner and you can see that his relative return ability is higher than his relative serve ability. In fact, there are matches in the dataset where Nishikori's $srel$ is negative and his $rrel$ is even higher than in this match.

2.4 Benchmarking s_{At}, r_{At} to the Tournament

On the tennis tour, there are three types of surfaces: grass, clay, and hardcourt. It's plausible to believe that on the different surfaces, there are different average serve and return win rates. Thus, when estimating s_{ABt}, r_{ABt}, it makes sense to account for the surface in our prediction. I go even further and account for the tournament in which the match is played (all matches in a tournament are played on the same surface). In fact, in the year 2013, the average serve win rate at the Wimbledon Tournament (grass) was 0.6628, the average serve win rate at the French Open (clay) was 0.6272, and the average serve win rate at the US Open (hardcourt) was 0.6241.

For game (A, B, t) between players A, B happening at time and tournament t, define st_t to be the average serve win rate in the past year for the tournament hosting match t. Analogously, define, rt_t to be the average return win rate in the past year for the tournament hosting match t.

Then define:

$$s_{At} := \max(\min(st_t + srel_{At}, 0.985), 0.015) \tag{8}$$

$$r_{Bt} := \max(\min(rt_t + rrel_{Bt}, 0.985), 0.015) \tag{9}$$

2.5 Summary of my approach

As a remark, I have proposed (equations 8 and 9) my first approach to estimate s_{At}, r_{At} requested in equation 3 at the end of section 2.2. To gain intuition, s_{At} intended to capture A's serve win rate versus a "representative" set of opponents at this tournament. We expect s_{At} to be increasing in the average serve win rate of the tournament: if it's "easier" to win serves at this tournament than at the typical tournament, we expect A to win more serves at this tournament than the typical tournament. Also, we expect this quantity to be increasing in $srel_{At}$: if A was a better server versus it's set of opponents than the average server versus that same set of opponents, we expect A to win more serves

[8] As of this writing Federer has 20 grandslam titles to his name, third most in the history of tennis. Only Rafael Nadal and Novak Djokovic have more with 22 grandslam victories and 21, respectively.

at this tournament. In my ad-hoc approach, I am assuming that these two terms contribute additively to s_{At} and that there are no other contributing factors.

With approaches to calculate $s_{At}, r_{At}, s_{Bt}, r_{Bt}$, according to equation 3, I now have my approach to estimate s_{ABt} and s_{BAt}, my inputs to the hierarchical Markov model.

In summary, I define:

$$s_{ABt} := \Pr[\text{A wins serve against B on day and tourney t}] = \frac{(s_{At})(1 - r_{Bt})}{(s_{At})(1 - r_{Bt}) + (r_{Bt})(1 - s_{At})}$$

where

$$s_{At} := st_t + (swr_{At} - (1 - SOSR_{At}))$$
$$r_{Bt} := rt_t + (rwr_{At} - (1 - SOSS_{Bt}))$$

3 SHRINKAGE ESTIMATORS

3.1 Introduction with a Paradox

Suppose you're interested in simultaneously estimating the proportion of attendees at Wimbledon that wear a hat each day (μ_{hat}), the fraction of red cars driving around Columbus Circle in Manhattan every day (μ_{car}), and the 2022 batting average of Aaron Judge of the New York Yankees (μ_{ba}). You will report your estimates $\hat{\mu}_{\text{hat}}, \hat{\mu}_{\text{car}}$, and $\hat{\mu}_{\text{ba}}$.

You observe N iid people at Wimbledon: $\{H_i : 1 \leq i \leq N\}$, where $H_i := 1_{\text{person i wears a hat}}$, you observe N iid cars driving through Columbus Circle: $\{C_i : 1 \leq i \leq N\}$, where $C_i := 1_{\text{car i is red}}$, and you observe N iid at-bats of Aaron Judge: $\{B_i : 1 \leq i \leq N\}$, where $B_i := 1_{\text{at-bat i is a hit}}$. For simplicity, suppose that $\text{Var}(H_i) = \text{Var}(C_i) = \text{Var}(B_i) = \sigma^2$ for all i [9]. Also define $h := \frac{\sum_{i=1}^{N} H_i}{N}$, $c := \frac{\sum_{i=1}^{N} H_i}{N}$, and $b := \frac{\sum_{i=1}^{N} A_i}{N}$.

The most intuitive report would be to give

$$(\hat{\mu}_{\text{hat}}, \hat{\mu}_{\text{car}}, \hat{\mu}_{\text{ba}}) = (h, c, b)$$

That is, estimate the sample frequency for each problem. This estimate is unbiased, meaning that in expectation across samples, we are estimating the true parameter for each problem. Can we provide a "better" estimate? The answer to that question depends on our definition of "better"- we want an estimate that's "probably closer" to the three values we're estimating. Mathematically, if we're interested in reducing the *mean squared error* of our parameter estimates, we can do better than predicting the sample frequencies (Stein 1956). In fact, the James-Stein estimator[10] estimates,

$$(\hat{\mu}_{\text{hat}}, \hat{\mu}_{\text{car}}, \hat{\mu}_{\text{ba}}) = (h(1 - \frac{\sigma^2/N}{h^2 + c^2 + b^2}), c(1 - \frac{\sigma^2/N}{h^2 + c^2 + b^2}), b(1 - \frac{\sigma^2/N}{h^2 + c^2 + b^2}))$$

[9] In our initial analysis of the paradox, it's not necessary that N be the same for all observations nor that the random variables have identical variance; it just simplifies the analysis.

[10] The reason that this estimator is called a shrinkage estimator is because of the shrinkage factor $1 - \frac{\sigma^2/N}{h^2+c^2+b^2}$. If we're worried that the shrinkage factor could take on a negative value, we could modify the shrinkage factor to $\max(1 - \sigma^2/N\frac{}{h^2+c^2+b^2}, 0)$.

have lower mean squared error than the intuitive estimates. The alarming part of this result is that Wimbledon hat attendance has seemingly become useful in predicting the batting average of Aaron Judge and the number of red cars driving around Columbus Circle. Thinking carefully, the Wimbledon hat attendance aids in estimating these two other quantities because our objective is to pick estimators to minimize the mean squared error for all three quantities simultaneously, not to pick estimators that minimize each estimator's mean squared error individually. The paradox lies in the fact that to reduce the mean squared error of the joint problem, you can do better than estimating the sample frequencies for each problem (Stein 1956). A sanity-check question: when would we ever want to SIMULTANEOUSLY estimate the proportion of attendees at Wimbledon that wear a hat each day, the fraction of red cars driving around Columbus Circle in Manhattan every day, and the 2022 batting average of Aaron Judge of the New York Yankees? Probably never! Thus, the paradox doesn't make much sense in real-life but it's something to be aware of. I'll provide a more formal introduction to the James-Stein Estimator later in this article along with a simplified proof.

Thinking about the paradox's existence is helpful for two reasons in this article. First, we observe a lot of tennis data and we want to estimate the true serve win rate of many players, among other quantities, all of which are seemingly related: shrinkage estimators sound appetizing. Second, this paradox introduces the idea that the estimator that reduces the mean squared error need not be unbiased. In fact, in this case, and many other cases, we can introduce a bias into our estimate to reduce the mean squared error of our estimator.

3.2 Bias-Variance Tradeoff of Estimators

Suppose we're trying to estimate a parameter x and give estimate $\hat{x}$ based on data D. We want $\hat{x}$ to be "close" to x. To formulate that, we typically pick $\hat{x}$ so as to minimize the *mean squared error*. Mathematically, we pick $\hat{x}$ to minimize $MSE(\hat{x}) := \mathbb{E}_D[(\hat{x} - x)^2]$. The D subscript indicates that the expectation is taken over the sampling outcome D[11]- I omit the subscript from here on out for brevity. I assume the reader is familiar with the decomposition of error into bias and variance though I'll provide the derivation here:

$$
\begin{aligned}
MSE(\hat{x}) &= \mathbb{E}[(\hat{x} - x)^2] \\
&= \mathbb{E}[((\hat{x} - \mathbb{E}[\hat{x}]) - (\mathbb{E}[\hat{x}] - x))^2] \\
&= \mathbb{E}[(\hat{x} - \mathbb{E}[\hat{x}])^2 + (\mathbb{E}[\hat{x}] - x)^2 - 2(\hat{x} - \mathbb{E}[\hat{x}])(\mathbb{E}[\hat{x}] - x)] \\
&= \mathbb{E}[(\hat{x} - \mathbb{E}[\hat{x}])^2] + \mathbb{E}[(\mathbb{E}[\hat{x}] - x)^2] - 2\mathbb{E}[(\hat{x} - \mathbb{E}[\hat{x}])(\mathbb{E}[\hat{x}] - x)] \\
&= \mathbb{E}[(\hat{x} - \mathbb{E}[\hat{x}])^2] + \mathbb{E}[(\mathbb{E}[\hat{x}] - x)^2] - 2(\mathbb{E}[\hat{x}] - x)\cancelto{0}{(\mathbb{E}[\hat{x}] - \mathbb{E}[\hat{x}])} \\
&= \mathbb{E}[(\hat{x} - \mathbb{E}[\hat{x}])^2] + (\mathbb{E}[\hat{x}] - x)^2 \\
&= \text{Var}(\hat{x}) + (\text{bias}(\hat{x}))^2
\end{aligned}
$$

To explain the nontrivial steps, in step 5, I use the fact the term $(\mathbb{E}[\hat{x}] - x)$ is independent of the outer expectation on D and so we can pull it out. Then, I simplify the other part of that last term: $\mathbb{E}[\hat{x} - \mathbb{E}[\hat{x}]] = \mathbb{E}[\hat{x}] - \mathbb{E}[\hat{x}] = 0$ since $\mathbb{E}[\hat{x}]$ is a constant and hence can be pulled out of the outer expectation by linearity. In the last step, notice that the first term is simply the definition for $\text{Var}(\hat{x})$

[11] This detail is important and glanced over by those learning and critical to analyzing this error function.

and in the second term, we can interpret $\mathbb{E}[\hat{x}] - x$ as the bias (ie., how far our estimate is from the true parameter on average).

The result, $MSE(\hat{x}) = \text{Var}(\hat{x}) + (\text{bias}(\hat{x}))^2$, has many nice interpretations and implications. Our error from predicting $\hat{x}$ can be decomposed into a variance and bias component[12]. The variance component is the part of the estimation error that arises because our sample is a random. The bias component is the estimation error if our sample were the entire population.

In estimating $\hat{x}$, we equally penalize error due to variance and bias^2, and we penalize extremes of each more due to the squared nature of each term. It is arbitrary that we pick $\hat{x}$ so as to minimize $\text{Var}(\hat{x}) + (\text{bias}(\hat{x}))^2$. We could also insert $\lambda \in [0, 1]$ and pick $\hat{x}$ so as to minimize $\lambda * \text{Var}(\hat{x}) + (1 - \lambda) * (\text{bias}(\hat{x}))^2$ or raise each of the terms to different powers. If an estimator is worse than another in both variance and bias, it's clear it's the worse estimator, but it's typically the case that one's better in one component and worse in the other and we must decide how much we care about each (ie., λ). The success of an estimator $\hat{x}$ also depends on what the true value of x ends up being. If one estimator is worse than another for all values of x, it's easy to say that it's the worse estimator; however, if the estimator is better for some values of x and worse for others, the statistician again must make a choice.

When we use an unbiased estimator, $0 = \text{bias}(\hat{x}) = \mathbb{E}[\hat{x}] - x$. That means, our estimation error $MSE(\hat{x}) = \text{Var}(\hat{x})$ comes entirely from the variance term and depends only on the sample being random. In the case of estimating the frequency of hat attendance at Wimbledon, red cars at Columbus Circle, and Aaron Judge batting average in section 3.1, the unbiased estimator was predicting the sample frequencies for each of the values. It's often the case that an unbiased estimator has the lowest mean squared error of any estimator but that's not always the case as you will see clearly in the next example.

3.3 The Zero Estimator

Suppose we observe a single draw X_1 from a distribution X with mean x and variance σ^2. Consider two different estimators $\hat{x}_1 = X_1$ and $\hat{x}_2 = 0$. In the case of $\hat{x}_1$, we estimate x as our single sample. In $\hat{x}_2$, we estimate x as 0 regardless of what we observe with X_1- I call $\hat{x}_2$ the zero estimator. Is it possible that $\hat{x}_2$ is better than $\hat{x}_1$ in terms of mean-squared error?

$$\begin{aligned} MSE(\hat{x}_1) &= \text{Var}(\hat{x}_1) + (\text{bias}(\hat{x}_1))^2 \\ &= \mathbb{E}[(\hat{x}_1 - \mathbb{E}[\hat{x}_1])^2] + (\mathbb{E}[\hat{x}_1] - x)^2 \\ &= \mathbb{E}[(X_1 - \mathbb{E}[X_1])^2] + \cancelto{0}{(\mathbb{E}[X_1] - x)^2} \\ &= \sigma^2 \end{aligned}$$

[12]Rasmusen (2021) explains nicely that these aren't necessarily the only types of errors. For example, you might have some strange preference that $\hat{x}$ not lie in the interval $[1, 3]$. That is a situational type of error and not applicable here- I'll ignore this in my future discussion.

$$\begin{aligned} MSE(\hat{x}_2) &= \text{Var}(\hat{x}_2) + (\text{bias}(\hat{x}_2))^2 \\ &= \mathbb{E}[(\hat{x}_2 - \mathbb{E}[\hat{x}_2])^2] + (\mathbb{E}[\hat{x}_2] - x)^2 \\ &= \cancelto{0}{\mathbb{E}[(0-0)^2]} + (0-x)^2 \\ &= x^2 \end{aligned}$$

As you can see from the derivation, $\hat{x}_1$ is unbiased- the bias component of $MSE(\hat{x}_1)$ equals 0 since $\mathbb{E}[\hat{x}_1] = \mathbb{E}[X_1] = x$ and thus the $MSE(\hat{x}_1)$ depends solely on it's variance component. The estimator $\hat{x}_2$ has zero variance since we always predict zero and so the $MSE(\hat{x}_2)$ depends solely on the bias component. To decide which estimator is better (according to MSE), we need to see if $MSE(\hat{x}_1) < MSE(\hat{x}_2) \iff \sigma^2 < x^2$ or vice versa. In other words, depending on the values of σ and x, an estimator that ignores the sample and always predicts zero could be "better" than the unbiased estimator. To give intuition for this result, if the variance of X is very large relative to it's mean, we don't learn very much from the sample so we might as well predict something arbitrary, say 0.

We have an issue though, we don't actually know the values of x and σ^2 when we're trying to estimate. So it's hard to say which of $\hat{x}_1$ or $\hat{x}_2$ is better apriori. If we did know x, the best estimator of x would be $\hat{x} = x$- we ignore the sample completely and the thought experiment between $\hat{x}_1$ and $\hat{x}_2$ makes no sense.

To say that one estimator is *completely superior* to another (according to the MSE criterion), we want the former estimator to have lower MSE than the latter regardless of what value x, the estimand, takes on. In the case of the zero estimator and the unbiased estimator here, we don't have a a conclusive result either way. The power of the James-Stein (1956) estimator is that it is indeed *completely superior* to the unbiased estimator of sample frequencies. Before we get to the James-Stein estimator though, there is a more simple class of shrinkage estimators to look- the Oracle estimator. Even before that, I'll summarize why shrinkage estimators are useful when looking at this tennis data.

3.4 Reason for Shrinkage Estimators in Tennis Analysis

I propose shrinkage estimators to address biases (3) and (4) in the abstract that may affect our ability to predict tennis matches if they're not accounted for.

As a reminder, in my tennis model, I assume that each player has some true serve win probability versus "representative" competition and take each serve from the past year as an iid draw from a Bernoulli distribution with that true serve win probability. To restate bias (3), some players play more matches than others and thus serve more times than others in the year. By the law of large numbers, if a player serves lot's of times, his sample serve win rate frequency will approach his true serve win probability. However, if a player serves few times, his sample serve win rate frequency could lie far from its true value. In other words, as someone trying to estimate each player's true serve win probability, I have confidence in the sample frequency if the player served many times and I don't have confidence in the sample frequency if the player served few times. To take it to the extreme, if a player served just once and won the service point, his serve win rate would be 100% but it surely

would not make sense to predict that he will win every single serve.

Shrinkage estimators gives a theoretically sound estimate of the true serve win probability. It is a weighted average of the sample serve win frequency and some other value (say the serve win rate across all players)- where we place great weight on the sample serve win rate frequency if we're confident in this value (the player served many times) and small weight otherwise.

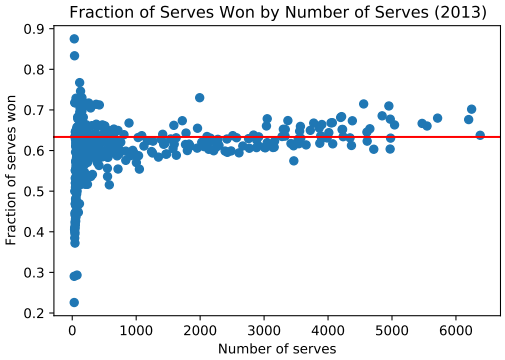


Figure 3. Fraction of Serves Won by Number of Serves in 2013

Figure 3 demonstrates the reason for shrinkage. Among those players who served few times, there's great variability in the serve win rate where as among players who served lot's of times, there's less variability. The red line is the serve win rate across all players, a candidate value, towards which to shrink our estimates since we expect, statistically speaking, serve win rates to take that value.

Figure 3 also demonstrates another bias, there's a slight positive association between the number of serves a player has done in the past year and his serve win rate, that is, bias (4) of the abstract. The reason for this pattern is that in the sport of tennis, the season consists of a succession of single elimination tournaments- if a player wins a match they advance, if they lose they're eliminated. Players who win matches tend to have a higher serve win rate (because they won the match) and then they also serve more times since they'll play another match in the continuation of the tournament.

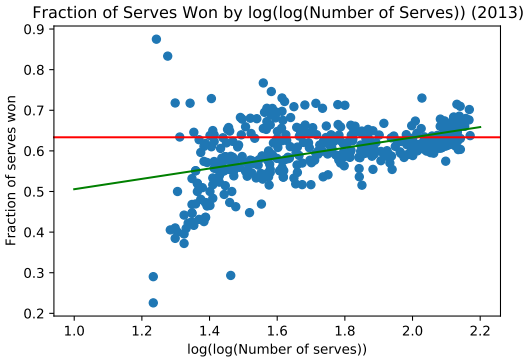


Figure 4. Fraction of Serves Won by log(log(Number of Serves)) in 2013

Figure 4 demonstrates this positive relationship in a more pronounced manner by putting number of serves on a double-log scale. Again, the red line is the serve win rate across all players and the green line is the best linear function of serve win rate on log(log(number of serves)). For players that have a

given log(log(number of serves)), we expect (statistically speaking) there serve win rate to lie on the green line, not the red line. Thus, to correct for bias (4), we want to control for the number of serves a player has attempted in our shrinkage and shrink towards the green line.

3.5 Shrinkage Estimators Definition

Consistent with our earlier discussion, suppose we're trying to estimate the mean x of some distribution X with known finite variance σ^2 given N iid draws $D = [X_1, ..., X_N]$ from the distribution. A shrinkage estimator $\hat{x}_{\text{shrink}}$ for estimating x takes the form:

$$\hat{x}_{\text{shrink}} := (1 - B)\bar{x} + Bx_0 \tag{10}$$

where $\bar{x} := \sum_{i=1}^{N} \frac{X_i}{N}$, x_0 is some value we want to "shrink" our estimate towards (typically due to prior information), and $B : (X_1, ..., X_N, X, N) \mapsto [0, 1]$ specifies how much we want to shrink our estimate towards x_0. First, notice that $\bar{x}$ is the unbiased estimator of x and so for any value of $B \neq 0$, $\hat{x}_{\text{shrink}}$ is a biased estimator. Another remark, we want B to be a nonincreasing function of N since if our sample is larger, our sample is more representative of the overall population and thus we want to place at least as much trust in $\bar{x}$. I encourage the reader to look at the subsequently presented shrinkage estimators through this form.

3.5.1 x_0-Oracle Estimator

Let's write the x_0-Oracle slightly differently from the structure above, as in Rasmusen (2021).

$$\hat{x}_{\text{Oracle}(x_0)} := \bar{x} - B(\bar{x} - x_0) \tag{11}$$

We're interested in computing the MSE of $\hat{x}_{\text{Oracle}(x_0)}$ to compare it to the MSE of $\bar{x}$. Again, the expectation is taken over the data D.

$$\begin{aligned}
MSE(\hat{x}_{\text{Oracle}(x_0)}) &= \mathbb{E}[((\bar{x} - B(\bar{x} - x_0)) - x)^2] \\
&= \mathbb{E}[(\bar{x} - x) - B(\bar{x} - x_0))^2] \\
&= \mathbb{E}[(\bar{x} - x)^2 + B^2(\bar{x} - x_0)^2 - 2B(\bar{x} - x)(\bar{x} - x_0)] \\
&= \frac{\sigma^2}{N} + B^2\mathbb{E}[\bar{x}^2 + x_0^2 - 2\bar{x}x_0] - 2B\mathbb{E}[\bar{x}^2 - \bar{x}x - \bar{x}x_0 + xx_0] \\
&= \frac{\sigma^2}{N} + B^2((\frac{\sigma^2}{N} + x^2) + x_0^2 - 2xx_0) - 2B((\frac{\sigma^2}{N} + x^2) - x^2 - \cancelto{0}{xx_0 + xx_0}) \\
&= \frac{\sigma^2}{N} + B^2(\frac{\sigma^2}{N} + (x - x_0)^2) - 2B\frac{\sigma^2}{N}
\end{aligned}$$

Now, we differentiate $MSE(\hat{x}_{\text{Oracle}(x_0)})$ with respect to B and set equal to 0 to find the minimizing B. It's clear the second order condition is satisfied for a global minimum since we're a differentiating a degree two polynomial on B with positive coefficient on squared term (ie., $\frac{\sigma^2}{N} + (x - x_0)^2 > 0$).

$$0 = \frac{\mathrm{d}MSE(\hat{x}_{\text{Oracle}(x_0)})}{\mathrm{d}B} = 2B(\frac{\sigma^2}{N} + (x - x_0)^2) - 2\frac{\sigma^2}{N}$$

Solving for B gives:

$$B = \frac{\sigma^2/N}{\sigma^2/N + (x - x_0)^2}$$

and consequently

$$\hat{x}_{\text{Oracle}(x_0)} = \bar{x} - \frac{\sigma^2/N}{\sigma^2/N + (x - x_0)^2}(\bar{x} - x_0)$$

and the mean squared error at the optimum B:

$$\begin{aligned} MSE(\hat{x}_{\text{Oracle}(x_0)}) &= \frac{\sigma^2}{N} + (\frac{\sigma^2/N}{\sigma^2/N + (x - x_0)^2})^2(\frac{\sigma^2}{N} + (x - x_0)^2) - 2(\frac{\sigma^2/N}{\sigma^2/N + (x - x_0)^2})\frac{\sigma^2}{N} \\ &= \frac{\sigma^2}{N}(1 - \frac{\sigma^2/N}{\sigma^2/N + (x - x_0)^2}) \end{aligned}$$

First, notice that the mean squared error for the x_0-Oracle estimator is lower than the mean squared error of the unbiased estimator ($\frac{\sigma^2}{N}$) regardless of what is the true value of x. This result is surprising since no matter how close x_0 is to x, the x_0-Oracle estimator has lower MSE than the unbiased estimator. We can have terrible prior information x_0 and still benefit. Though, if x_0 is close to x, we benefit more. Also notice that B is an increasing function of σ^2- the intuition is that if our draws from X have larger variance, they're typically further from x and so we should trust these draws less when predicting x. Also, notice that B is a decreasing function of N, which agrees with our sanity check in the shrinkage estimators definition section (section 3.5). As a remark, we do not know the value of x and so we must estimate this parameter when employing this estimator.

In the space of the tennis analysis, I will replace the oracle shrunken estimate with the computed serve win rate (without loss of generality I'll just talk about serve win frequency for brevity) in my procedure for estimating the probability that one player wins a serve on another. To make this replacement, I need to specify x_0, the value to which I'll shrink the serve win rate, and σ^2/N. x_0 I will let be the all player serve win rate. σ^2, the variance of the serve win rate of the player, I will leave its estimation for discussion in the variance section (section 3.6).

3.5.2 The James-Stein Estimator, Shrinking to 0

Now it's time to mathematically justify the paradoxical claim I made in introducing shrinkage estimators in section 3.1, that the James-Stein shrunken estimators have lower total mean squared error than the sample frequencies when we're trying to jointly estimate the hat attendance percentage at Wimbledon, the percentage of red cars in Columbus Circle, and the batting average of Aaron Judge. In other words, I want to justify the paradox that Wimbledon hat attendance is "useful" in estimating the fraction of red cars in Columbus Circle and the batting average of Aaron Judge.

For ease of notation, assume we have k Bernoulli distributions $X_1, ..., X_k$ and we're trying to estimate their means $x_1, ..., x_k$, respectively. Each distribution has identical variance σ^2 and we observe N iid

draws [13] $(X_{11}, ..., X_{1N}, X_{21}, ..., X_{2N}, ..., X_{k1}, ..., X_{kN})$ from each distribution. A reminder that in my notation, $\bar{x}_i = \frac{1}{N}\sum_{j=1}^{N} x_{ij}$ and hence $\bar{x}_i \sim \mathcal{N}(x_i, \frac{\sigma^2}{N})$ approximately.

We define the James-Stein Estimator, shrinking to 0, for x_1 with analogous expressions for $x_2, ..., x_k$:

$$\hat{x}_{1\text{JS}} := \bar{x}_1 - (k-2)\frac{\sigma^2/N}{\sum_{i=1}^{k}\bar{x}_i^2}\bar{x}_1 \tag{12}$$

To aid in demonstrating that the James-Stein estimator has lower total MSE than the unbiased sample averages, as in Rasmusen (2021), I define:

$$g(\bar{x}_1) := (k-2)\frac{\sigma^2/N}{\sum_{i=1}^{k}\bar{x}_i^2}\bar{x}_1$$

with derivative

$$\frac{\mathrm{d}g(\bar{x})}{\mathrm{d}\bar{x}} := (k-2)\frac{\sigma^2}{N}\left(\frac{1}{\sum_{i=1}^{k}\bar{x_i}^2} - \frac{2\bar{x_1}^2}{(\sum_{i=1}^{k}\bar{x_i}^2)^2}\right)$$

There's one other useful fact to know in our derivation of the MSE: $\mathbb{E}[(\bar{x}_1 - x_1)g(\bar{x}_1)] = \frac{\sigma^2}{N}\mathbb{E}[\frac{\mathrm{d}g(\bar{x}_1)}{\mathrm{d}\bar{x}_1}]$. As Rasmusen (2021) explains, this comes as an implication of Stein's Lemma from Stein (1981) for rotationally symmetric densities like the normal distribution. Now onto our simplification of the $MSE(\hat{x}_{1\text{JS}})$.

$$\begin{aligned}
MSE(\hat{x}_{1\text{JS}}) &= \mathbb{E}[(\hat{x}_{1\text{JS}} - x_1)^2] \\
&= \mathbb{E}[((\bar{x}_1 - g(\bar{x}_1)) - x_1)^2] \\
&= \mathbb{E}[((\bar{x}_1 - x_1) - g(\bar{x}_1))^2] \\
&= \mathbb{E}[(\bar{x}_1 - x_1)^2] + \mathbb{E}[g(\bar{x}_1)^2] - 2\mathbb{E}[(\bar{x}_1 - x_1)g(\bar{x}_1)] \\
&= \frac{\sigma^2}{N} + \mathbb{E}[(k-2)^2\frac{(\sigma^2/N)^2}{(\sum_{i=1}^{k}\bar{x}_i^2)^2}\bar{x}_1^2] - 2\frac{\sigma^2}{N}\mathbb{E}[(k-2)\frac{\sigma^2}{N}\left(\frac{1}{\sum_{i=1}^{k}\bar{x}_i^2} - \frac{2\bar{x}_1^2}{(\sum_{i=1}^{k}\bar{x}_i^2)^2}\right)] \\
&= \frac{\sigma^2}{N} + (k-2)^2(\frac{\sigma^2}{N})^2\mathbb{E}[\frac{\bar{x}_1^2}{(\sum_{i=1}^{k}\bar{x}_i^2)^2}] - 2(k-2)(\frac{\sigma^2}{N})^2\mathbb{E}[\frac{\sum_{i\neq 1}^{k}\bar{x}_i^2 - \bar{x}_1^2}{(\sum_{i=1}^{k}\bar{x}_i^2)^2}] \\
&= \frac{\sigma^2}{N} + (k-2)^2(\frac{\sigma^2}{N})^2\mathbb{E}[\frac{\bar{x}_1^2}{(\sum_{i=1}^{k}\bar{x}_i^2)^2}] + 2(k-2)(\frac{\sigma^2}{N})^2\mathbb{E}[\frac{\bar{x}_1^2}{(\sum_{i=1}^{k}\bar{x}_i^2)^2}] - 2(k-2)(\frac{\sigma^2}{N})^2\mathbb{E}[\frac{\sum_{i\neq 1}^{k}\bar{x}_i^2}{(\sum_{i=1}^{k}\bar{x}_i^2)^2}] \\
&= \frac{\sigma^2}{N} + (k-2)(\frac{\sigma^2}{N})^2\mathbb{E}[\frac{\bar{x}_1^2}{(\sum_{i=1}^{k}\bar{x}_i^2)^2}]((k-2)+2) - 2(k-2)(\frac{\sigma^2}{N})^2\mathbb{E}[\frac{\sum_{i\neq 1}^{k}\bar{x}_i^2}{(\sum_{i=1}^{k}\bar{x}_i^2)^2}] \\
&= \frac{\sigma^2}{N} + (k-2)(\frac{\sigma^2}{N})^2(k\mathbb{E}[\frac{\bar{x}_1^2}{(\sum_{i=1}^{k}\bar{x_i}^2)^2}] - 2\mathbb{E}[\frac{\sum_{i\neq 1}^{k}\bar{x}_i^2}{(\sum_{i=1}^{k}\bar{x}_i^2)^2}])
\end{aligned} \tag{13}$$

[13] The assumptions of identical variance, same number of samples from each distribution, and Bernoulli distributions can be relaxed but the math becomes algebraicly more complicated and is out of scope of this article. In other words, the James-Stein Estimator is indeed generalizable for different sample sizes, different variances, and different distributions.

At this point, it's difficult to tell whether $MSE(\hat{x}_{1\text{JS}})$ is larger or smaller than $MSE(\bar{x_1}) = \frac{\sigma^2}{N}$. Let's look at the sum of the MSE's across all k dimensions.

$$\begin{aligned}
\sum_{j=1}^{k} MSE(\hat{x}_{j\text{JS}}) &= \sum_{j=1}^{k}(\frac{\sigma^2}{N} + (k-2)(\frac{\sigma^2}{N})^2(k\mathbb{E}[\frac{\bar{x}_j^2}{(\sum_{i=1}^k \bar{x}_i^2)^2}] - 2\mathbb{E}[\frac{\sum_{i\neq j}^k \bar{x}_i^2}{(\sum_{i=1}^k \bar{x}_i^2)^2}])) \\
&= k\frac{\sigma^2}{N} + (k-2)(\frac{\sigma^2}{N})^2 \sum_{j=1}^{k}(k\mathbb{E}[\frac{\bar{x}_j^2}{(\sum_{i=1}^k \bar{x}_i^2)^2}] - 2\mathbb{E}[\frac{\sum_{i\neq j}^k \bar{x}_i^2}{(\sum_{i=1}^k \bar{x}_i^2)^2}]) \\
&= k\frac{\sigma^2}{N} + (k-2)(\frac{\sigma^2}{N})^2 \mathbb{E}[\frac{k\sum_{j=1}^k \bar{x}_j^2 - 2(k-1)\sum_{j=1}^k x_j^2}{(\sum_{i=1}^k \bar{x}_i^2)^2}] \\
&= k\frac{\sigma^2}{N} + (k-2)(\frac{\sigma^2}{N})^2 \mathbb{E}[\frac{(k-2(k-1))\sum_{j=1}^k \bar{x}_j^2}{(\sum_{i=1}^k \bar{x}_i^2)^2}] \\
&= k\frac{\sigma^2}{N} - (k-2)^2(\frac{\sigma^2}{N})^2 \mathbb{E}[\frac{1}{\sum_{i=1}^k \bar{x}_i^2}] \\
&< k\frac{\sigma^2}{N} = \sum_{j=1}^{k} MSE(\bar{x}_j) \text{ if } k > 2
\end{aligned}$$

We have shown that the total mean squared error when using the James-Stein estimator is less than that of using the arithmetic average for each of the components if $k > 2$. If $k = 1$, the James-Stein estimator loses, and if $k = 2$ the James-Stein estimator ties. Even if $k > 2$, it's not clear that each component will have a smaller mean squared error than the arithmetic average, as we see in equation 13- our only guarantee is that the sum of the mean squared errors of all components will be smaller with the James-Stein estimator.

3.5.3 Understanding the James-Stein Estimator

To gain some intuition for what's going on, one useful move is to compare the oracle estimator for $x_0 = 0$ and James-Stein estimator shrinking to 0:

$$\begin{aligned}
\hat{x}_{\text{Oracle}(x_0)} &:= \bar{x} - \frac{\sigma^2/N}{\sigma^2/N + x^2}\bar{x} \\
\hat{x_{1\text{JS}}} &:= \bar{x_1} - (k-2)\frac{\sigma^2/N}{\sum_{i=1}^k \bar{x_i}^2}\bar{x_1}
\end{aligned}$$

First, notice that $\mathbb{E}[X^2] = \text{Var}(X) + \mathbb{E}[X]^2 = \sigma^2/N + x^2$ and so we can write $\hat{x}_{\text{Oracle}(x_0)} = \bar{x} - \frac{\sigma^2/N}{\mathbb{E}[X^2]}\bar{x}$. In the JS estimator, for intuition-reasons, lets replace[14] the $k-2$ with k and sweep it into the denominator so that $\hat{x}_{1\text{JS}} \approx \bar{x}_1 - \frac{\sigma^2/N}{\frac{1}{k}\sum_{i=1}^k \bar{x}_i^2}\bar{x}_1$. The expressions for the two now estimators now look very similar except for the fact that the denominator of the Oracle shrinkage multiplier is the

[14]The $k-2$ (as opposed to k) comes from the fact that fact that for samples that give unrealistically large $\bar{x_1}$, we want to shrink these samples more but the correspondingly large $\bar{x_1}^2$ in the denominator results in a smaller shrinkage than desired. In other words, large sample averages are correlated with small shrinkage factors and so we replace k with $k-2$ to correct for this effect.

expectation of X^2 and the denominator of the James-Stein Estimator is the arithmetic average of the $\bar{x}_i^2$- it's a sample estimate of $\bar{x}_1^2$ in its own right if we "pretend" that all of the $\bar{x}_i$'s come from the same distribution.

Given our construction[15], in the James-Stein estimator, we're forced to have one shrinkage multiplier for all of the estimands and hence this "pretending" is the best we can do- we did prove that it's still better than the unbiased estimates! In the extreme where all distributions are the same, the denominator of the JS estimator becomes a proper unbiased of $\bar{x}_1^2$ and our estimators align even more with the corresponding Oracle estimators. In these cases where $X_1, ..., X_k$ are very similar, since our JS estimators are very similar to the analogous Oracle estimators, we're guaranteed[16] that each component will have less mean squared error than the unbiased estimates. Note this guarantee is stronger than having having the the sum of mean squared errors smaller than the sum of mean squared errors for unbiased estimates.

3.5.4 James-Stein Estimator Applied to Tennis Analysis

In the space of tennis, let's focus on estimating the serve win rate for various players in the past year. We have k Bernoulli distributions of serve win rates: $S_1, ..., S_k$ and we're trying to estimate their means $s_1, ..., s_k$, respectively. s_i should be interpreted as the true probability that player i wins a serve against a "representative" opponent and S_i the fraction of serves player i won in the past year. We say that $\frac{\sigma_i^2}{N_i} := \text{Var}(S_i)$ where N_i is the number of serves by player i in the past year and σ_i^2 is the variance of the probability of player i winning a single serve.

Let's add some additional assumptions in the lens of tennis- I want concrete-ize the notion that the serve win rates of all of the players must somehow be related. Let's say that s_i are selected iid from the following distribution: $s_i \sim \mathcal{N}(s, d)$ where s and d are the mean and variance of the distribution, respectively. Then, let's say that each S_i is selected from the distribution: $S_i \sim N(s_i, \frac{\sigma_i^2}{N_i})$ so that each S_i is also conditionally independent given the s_is. Then, algebra and some calculus implies that:

$$s_i|S_i \sim \mathcal{N}((1 - B_i)S_i + B_i s, \frac{\sigma_i^2}{N_i}(1 - B_i)) \tag{14}$$

where $B_i := \frac{\sigma_i^2/N_i}{\sigma_i^2/N_i + d}$. What's going on here? We're taking a two stage approach to generate the S_is. We're assuming that player i has its true serve win probability, s_i selected iid from a normal distribution. Then, we assume that player i wins each of its N_i serves iid with probability s_i so that S_i is approximately distributed normally (by the central limit theorem) with mean s_i and variance $\frac{\sigma_i^2}{N_i}$. We observe S_i and we want to estimate s_i and hence the derived conditional distribution in equation 14. If we want to estimate $s_i|S_i$, one intuitive approach, and that's what I'll do, is to estimate s_i as its conditional expectation. In other words, we estimate:

[15] We here required that each distribution have identical variance and that we took the same number of draws from each distribution.

[16] It's an interesting and loaded question, how similar do these distributions need to be to properly guarantee that the mean squared error of each component of the James-Stein Estimator is less than the mean squared error for the unbiased estimates? The answer to this question is out of scope of this article. Though, Rasmusen (2021) does provide a nice simple algebraic proof of a positive result to this question when all estimands are equal.

$$\hat{s}_{i(\text{tennis})} := \mathbb{E}[s_i|S_i] = (1 - B_i)S_i + B_i s \tag{15}$$

There's also nice intuition in the expression for $\hat{s}_{i(\text{tennis})}$. This estimator is a weighted average of the presumably known quantities S_i and s. We weight s more when B_i is larger- that happens when $\frac{\sigma_i^2}{N_i}$ is relatively larger than d. In other words, that means when the sample average (S_i) comes with more uncertainty, we weight the globally known variable s more. When the sample average comes with less uncertainty ($\frac{\sigma_i^2}{N_i}$ is relatively smaller than d), we weight the sample average more. It's nice to link this definition of $\hat{s}_{i(\text{tennis})}$ back to the original definition of a shrinkage estimator, x_{shrink}, in equation 10- the expressions basically say the same thing! When we're more confident in our sample average, we weight that quantity by more in our estimator; when we're less confident in our sample average, we weight our prior information more.

To employ this estimator, we must provide values for σ_i^2, N_i, d, s, and S_i. N_i and S_i are known- those are the number of serves by player i in the past year and the serve win rate of player i in the past year, respectively. The quantities d and σ_i^2 are variances that I must estimate to complete the estimator- I deal with these in the subsequent variance section (section 3.6). The quantity s, is the expectation of the normal distribution from which we draw the s_is- it's natural to make s the global average serve win rate[17].

$$s := \frac{\text{number of service points won by all players}}{\text{number of service points total by all players}} \tag{16}$$

3.6 Discussion of Estimating Variances

In the estimators discussed above, there are two variance quantities we need to estimate to employ our shrinkage estimator: (1) the variance of the sample serve win rate frequency ($\frac{\sigma_i^2}{N_i}$) and (2) the variance of the distribution from which we select the true serve win rates for each player (d).

3.6.1 Estimating $\frac{\sigma_i^2}{N_i}$

The first approach I use to estimate $\frac{\sigma_i^2}{N_i}$ is a "pooled variance estimate" in which I then divide by the number of attempts. To elaborate, I estimate $\hat{\sigma}_i^2 = (s)(1-s)$ and then divide that quantity by N_i to get my estimate of $\frac{\sigma_i^2}{N_i}$. As a reminder, s was defined in equation 16 as $s := \frac{\text{number of service points won by all players}}{\text{number of service points total by all players}}$ and hence the term "pooled variance". The intuition is that S_i is a Bernoulli random variable and

[17] Coincidentally, the solved expression for $\hat{s}_{i(\text{tennis})}$ was discovered analogously in the setting of an insurance company selecting a rate to cover the (potential) claim filed in period $n+1$ ($X_{i(n+1)}$) for individual i after observing n prior claims (X_{ij} for $1 \leq j \leq n$) for k individuals (Buhlmann and Straub (1970)). The insurance company assumes a risk parameter $\theta_i \sim \Pi_\theta$ and claims distributed iid for individual i: $X_{ij} \sim f_{X|\theta}(X|\theta_i)$. There are no assumptions on these distributions beyond the fact that they have a finite mean and variance. The insurance company then solves for the best linear predictor (according to mean squared error) of $X_{i(n+1)}$ given $X_1, ..., X_n$. In their solution, using B_i from our notation, they equivalently write that $\hat{x}_{i(n+1)} := (1 - B_i)\bar{X}_i + B_i x$ where $1 - B_i = \frac{n}{n + \frac{\mathbb{E}[\text{Var}(X|\theta)]}{\text{Var}(\mathbb{E}[X|\theta])}}$, $\bar{X}_i := \frac{1}{n}\sum_{j=1}^{n} X_{ij}$, and $x := \frac{1}{kn}\sum_{i=1}^{k}\sum_{j=1}^{n} X_{ij}$. In their intuition, holding everything else fixed, if the θ_i are more homogeneous, then we expect $\text{Var}(\mathbb{E}[X|\theta])$ to be small and so $1 - B_i$ small and therefore we place more weight on the global mean x when trying to predict the next claim. If the θ_i are more heterogeneous, vice-versa.

hence it's variance takes the form $\text{Var}(S_i) = p(1-p)$ where $p \in [0,1]$. For simplicity, I standardize the single serve variance estimate across players so that the variability in the variance estimate comes from N_i, the number of serves i attempted in the past year.

The second approach is to employ a cluster variance estimate. In this lens, rather than having each serve be the randomly sampled quantity, we utilize the information that serves came in clusters of matches so that the matches are the sampling unit. In the space of possible matches player i could have played in the past year, some matches were realized. The sampling units are the matches but the units of interest are still the serves in those matches. The distinction is important because the likelihood that player i wins a serve in a match varies depending on the match (one reason is that the opponents are different). The unbiased estimate for i's serves win rate is still the same with this sampling procedure (as when the serves are the sampling units)– $\bar{s}_i := \frac{\text{number of serves i won in the past year}}{\text{number of serves total by i in the past year}}$. The difference is that under this sampling procedure, we estimate the variance $\text{Var}(\bar{s}_i) := \frac{\sum_{j=1}^{k_i}(S_{ij}*N_{ij}-\bar{s}_i*N_{ij})^2}{k_i(k_i-1)\bar{N}_{ij}}$ where k_i is the number of matches played by i in the past year, N_{ij} is the number of service points i attempted in match j, and $\bar{N}_{ij}$ is the average number of service points i played in each match (Frerichs 2004)[18].

3.6.2 Estimating d

To reiterate, d, is the variance of the distribution from which we select the true serve win rates for each player. If our estimate for $\bar{s} := \frac{1}{n}\sum_{i=1}^{n}\bar{s}_i$, then the unbiased estimator for sample variance $\hat{d} := \frac{1}{n-1}\sum_{i=1}^{n}(\bar{s}_i - \bar{s})^2$ where $\bar{s}_i$ is player i's serve win rate over the past year and n is the number of players in the past year. We're assuming that each player has served enough times so that $\bar{s}_i$ is close to s_i, player i's true serve win rate drawn from the distribution in question[19].

While the previous approach for estimating d is nice and simple, it doesn't account for the fact that different players served a different number of times. The second approach I use to estimate d is again a cluster variance estimate. In this setting, rather than having each serve win rate be the randomly sampled quantity, we take players to be the sampling unit and we observe many serves from each player. The idea is that in the space of tennis players, we observed these n players in the past year and they each had there own serve win rate via a different number of total serves. When categorizing the variance of our prediction for s the "all player serve win rate", we want to factor in the notion that certain players served more than others. The unbiased estimate for the "all player serve win rate" is s as defined in equation 16– $s := \frac{\text{number of service points won by all players}}{\text{number of service points total by all players}}$. The corresponding variance of s is $\text{Var}(s) = \frac{\sum_{i=1}^{n}(S_i*N_i-s*N_i)^2}{(n-1)\bar{N}_i}$ where n is the number of players, N_i is the number of service points attempted by player i in the past year, S_i is the fraction of service points won by player i in the past year, and $\bar{N}_i$ is the average number of serves attempted by each player in the past year.

[18]Due to the $k_i - 1$ in the denominator, we do require that player i has played at least 2 matches in the past year to employ this variance estimator.

[19]The assumption isn't necessarily a bad one– it's easy to satisfy with our massive dataset. Instead of summing across all players, we can sum only across players that players that played at least 5 matches, say, then we're more convinced that $\bar{s}_i$ is indeed close to $\bar{s}$ for each player in the sum. Many many players have played at least 5 matches and so we can remove all players not satisfying this criteria and still get a nice estimate of d this way.

3.7 Beta Prior and Regression Estimators

The *Beta distribution* is a continuous distribution defined on an open or closed finite real interval. If X has a beta distribution with shape parameters $\alpha, \beta > 0$, we say that X has probability density function (pdf) and expectation:

$$f_X(x) := \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} \tag{17}$$

$$\mathbb{E}[X] = \frac{\alpha}{\alpha+\beta} \tag{18}$$

where $B(\alpha,\beta) := \frac{\Gamma(\alpha)(\Gamma(\beta)}{\Gamma(\alpha+\beta)}$ and $\Gamma(\cdot)$ is the gamma function ($B(\cdot,\cdot)$ serves as a normalizing constant so that the density f_x integrates to 1)[20]. The Beta distribution is useful to model the probability of probabilities since it's support is $[0,1]$ and since it's the conjugate prior for the Bernoulli, Binomial, and Geometric distributions. That means if we have a beta prior on some parameter p and observe an outcome of a Bernoulli, Binomial, or Geometric distributions that have probability parameter p, the posterior distribution of p given the prior and a Bayesian update is also a beta distribution. Conjugate priors are useful because they save us expensive and potentially impossible computation in computing posteriors since we know the posterior follows the same distribution as the prior.

The Beta distribution has a straightforward and interpretable Bayesian update. Suppose that we observe n iid outcomes $X_1, ...X_n$ where each $X_i \sim$ Bernoulli(p). For ease of notation, define $S_n := \sum_{i=1}^{n} X_i$. We have a prior on $p \sim$ Beta(α,β) for some shape parameters $\alpha, \beta > 0$. Our posterior distribution on p follows $p|X_1,...,X_n \sim$ Beta$(\alpha + S_n, \beta + (n - S_n))$. The posterior distribution on p is very simple to compute, we simply add the number of successes to α and the number of failures to β and define a new beta distribution with those parameters.

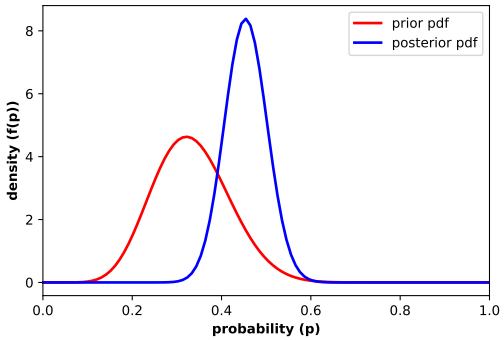


Figure 5. Demonstrating Bayesian Update for Beta Prior and Binomial Data

In figure 5, We can see a concrete example the prior and posterior densities when updating a beta distribution after observing Bernoulli outcomes. Suppose $X \sim$ Bernoulli(p) and p has prior $p \sim$ Beta$(\alpha = 10, \beta = 20)$. Suppose we observe 80 iid draws $(X_1, ..., X_{80})$ of X where 40 of them are

[20] Although not needed in this article, it's also possible to derive the beta distribution with four parameters: two shape parameters $\alpha, \beta > 0$ and two parameters a, b that determine the interval of support of the beta distribution as (a, b) or $[a, b]$. Suppose that $Y \sim$ Beta(α, β, a, b). We can define the linear transformation $Y := a + (c - a)X$ so that $X = \frac{Y-a}{c-a}$ to map the $[0, 1]$ beta distribution to the $[a, b]$ beta distribution. Then $f_Y(y) = \frac{1}{c-a} f_X(\frac{y-a}{c-a}) = \frac{(\frac{y-a}{c-a})^{\alpha-1}(1-\frac{y-a}{c-a})^{\beta-1}}{(c-a)B(\alpha,\beta)}$.

successes and 40 are failures. The posterior distribution for p is $p|X_1, .., X_{80} \sim \text{Beta}(\alpha = 50, \beta = 60)$. Graphically, there are a few nice points to observe in the update. The posterior distribution is tighter than the prior distribution since the $\alpha_{\text{posterior}} > \alpha_{\text{prior}}$ and $\beta_{\text{posterior}} > \beta_{\text{prior}}$- this reflects the fact that with more data, we're more confident about the precise area where the true p lies; the more data points we observe, the tighter the posterior will be. Second, the expectation of the posterior will lie between the expectation of the prior $(1/3)$ and the sample frequency of the data $(1/2)$- we favor the expectation of the prior more if $\alpha_{\text{prior}}, \beta_{\text{prior}}$ are relatively large compared to the amount of data points, and we favor the sample frequency more if we have lots of data points. In other words, if we're confident in our prior we weight our prior heavily, if we receive lots of data, we weight our sample heavily in judging p.

3.7.1 Beta Prior Estimator in Tennis Analysis

To apply the beta distribution to tennis analysis, there are two steps, first to fit a beta distribution to service point win rates of all players in the past year to serve as a prior. Then, to use the Bernoulli-Beta update formula for each player to compute a posterior serve win rate distribution for each player. To avoid outliers in serve win rates that result from players just playing a single match, I fit the beta distribution on players who played in at least two matches (but I compute the posterior distribution for all players, even those that played less than two matches).

To give an example, suppose A is playing against B at tournament t and I'm trying to compute the beta shrunken serve win rate for player A in the year prior to t. I take all players who played at least two matches in the year prior to t, I appended all of their serve win rates to an array: (eg., [0.64,0.658,...]). Then, I fit a beta distribution (with the interval of support set to [0,1] since serve win rates necessarily exist in that interval) on this list of serve win rates to produce Beta distribution shape parameters $\hat{\alpha}_t, \hat{\beta}_t$. From there, I compute the posterior distribution for the serve win rate of player A in the past year using the Bernoulli- Beta distribution update formula- the shape parameters become ($\hat{\alpha}_t$ + number serves won by A in past year, $\hat{\beta}_t$ + number serves lost by A in past year), respectively. Finally, I would compute the beta shrunken serve win rate s_{At} as the expectation of the posterior distribution $s_{At} := \frac{\hat{\alpha}_t + \text{number serves won by A in past year}}{\hat{\alpha}_t + \hat{\beta}_t + \text{number serves total by A in past year}}$. Here, similar to the simple example in the previous section (section 3.7) and similar in spirit to the general shrinkage estimator from section 3.5, it's clear that the beta shrunken serve win rate value is a compromise between the serve win rate predicted by the prior and the player's serve win rate frequency.

3.7.2 Beta Regression Distribution

In the typical Beta distribution X has shape parameters α, β and support on [a,b]. In the beta regression distribution, α, β are no longer native parameters to X, instead α, β are functions of regressors so that the shape of the beta distribution can vary with the regressors (it's possible for only one of α, β to be a function of regressors too).

It's actually easier to reason the construction of the beta regression model if we re-parametrize the beta distribution and for simplicity fix the interval of support to $[0, 1]$. Let $X \sim \text{Beta}(\alpha, \beta)$ with interval of support on $[0, 1]$ so that X can be interpreted as a prior on some parameter x. Let's define the mean of X as $\mu := \frac{\alpha}{\alpha+\beta}$ and the precision of X as $\phi := \alpha + \beta$ like in (Graf 2021). Note that $X \sim \text{Beta}(\mu\phi, (1-\mu)\phi)$ so that X can be equivalently characterized by mean and precision. I call this the beta regression model because I want to make μ, the mean of the beta distribution a function

of regressors. Let's define $\mu(x_j) = \sigma(\sum_{j=1}^{k} b_j x_j) = \frac{1}{1+e^{-\sum_{j=1}^{k} b_j x_j}}$. What's happening is that the beta distribution is now natively characterized by $\phi, b_1, ...b_k$ and for characteristics $[x_1, ..., x_k] \in \mathbb{B}^k$, we produce a Beta distribution. The function $\sigma : \mathbb{R}^k \mapsto [0, 1]$, often referred to as the sigmoid function, is called a link function because it "links" from regression parameters (bs) to the parameters of the probability distribution (μ). The Beta distribution is a special case of the Beta regression distribution where the link function is just a constant function.

The Beta regression distribution is useful, for one, when we expect the mean of the prior to vary with some characteristics and we want to take that into account in our prior(s). For example, suppose we want to construct a prior distribution for the probability that a bunch of newborns will graduate college. A newborn child coming from Harvard educated upper class parents has very different likelihood of graduating college than a newborn coming from a lower-class family- with one prior distribution these two newborns have the same prior for graduating Harvard. With priors controlling for parental education and income, we can have a continuum of priors for populations of newborns depending on parental income and education.

How do we fit a Beta regression distribution to data using a maximum likelihood approach? Suppose we have a N data points where data point i takes the form: $z_i = (p_i, \vec{x}_i)$ where p_i is a probability for point i that has characteristics $\vec{x}_i = x_{i1}, ..., x_{ik}$. We assume that each p_i is distributed iid from a Beta distribution with density f_p with mean μ and precision ϕ where $\mu(\vec{x}_i) = \sigma(\sum_{j=1}^{k} b_j x_{ij})$. Our objective is to solve the following maximization problem:

$$\begin{aligned}
\max_{b_1,\dots,b_k,\phi} \mathcal{L}(\text{Data}|b_1, \dots, b_k, \phi) &= \max_{b_1,\dots,b_k,\phi} \Pi_{i=1}^{N} f_p(p_i; \mu(\vec{x}_i), \phi) \\
&= \min_{b_1,\dots,b_k,\phi} -\sum_{i=1}^{N} \log[\frac{\Gamma(\phi) p_i^{\mu(\vec{x}_i)\phi-1}(1-p_i)^{(1-\mu(\vec{x}_i))\phi-1}}{\Gamma(\mu(\vec{x}_i)\phi)\Gamma((1-\mu(\vec{x}_i))\phi)}] \\
&= \min_{b_1,\dots,b_k,\phi} -\sum_{i=1}^{N} \log\Gamma(\phi) - \log\Gamma(\mu(\vec{x}_i)\phi) - log\Gamma((1-\mu(\vec{x}_i))\phi) + (\mu(\vec{x}_i)\phi - 1)log(p_i) + ((1-\mu(\vec{x}_i))\phi - 1)log(1-p_i)
\end{aligned}$$

We can write the following program to approximate the optimal $b_1, ..., b_k, \phi$ by solving the last minimization problem:

```
import numpy as np
from scipy.special import loggamma
from scipy.optimize import minimize

def calculate_mu_i(X, b):
    return 1 / (1 + np.exp(-(np.dot(X, b))))

def calculate_neg_loglikelihood(params, p, Z):
    b = params[0:-1]
    phi = params[-1]
    mu_i = calculate_mu_i(Z, b)
    ll = loggamma(phi) - loggamma(mu_i*phi) - loggamma((1 - mu_i)*phi) + (mu_i*
        phi - 1)*np.log(p) + ((1-mu_i)*phi-1)*np.log(1-p)

    return -np.sum(ll)


def calculate_optimal_beta_and_phi():
    # initialize the data for optimization
    p = ... # N dimensional list of probabilities
```

```
Z = ... # N x k matrix of characteristics indexed in the same way as p

# initialize the parameters for optimization
params = [1]*(k+1) # The first k dimensions are for b and the last dimension
    is for phi

res = minimize(calculate_neg_loglikelihood, x0=params, args=(p,Z), bounds=[(
    None,None), ..., (None,None), (0,None)], options={'ftol': 1e-06, 'gtol':
    1e-06}) # phi, precision of beta distribution, must be nonnegative

if res.success:
    return res.x
else:
    return [float("nan")]*(k+1)
```

3.7.3 Beta Regression Estimator in Tennis Analysis

To apply the Beta regression estimator to tennis analysis, there are two steps, first to fit a Beta regression distribution to service point win rates of all players in the past year. Then, to use the Bernoulli- Beta update formula for each player to compute a posterior serve win rate distribution for each player given their prior. To avoid outliers in serve win rates that result from players just playing a single match, I again fit the Beta regression distribution on players who played in at least two matches (but I compute the posterior distribution for all players, even those that played less than two matches).

When fitting the beta regression estimator to data, there are a list of characteristics we "sigmoid-regress" upon to compute the mean of the prior distribution. If you recall in figure 3, we saw that the serve win rate of player is positively associated with the number of serves a player has completed. That's because when players win more serves, they win more matches, advance in tournaments and serve more times. As a result of this relationship, I will make the mean of the prior of serve win rates for each player a sigmoid-linear function of the number of serves a player has done in the past year and a constant.

Let N_{it} be the number of service points player i attempted in the last year prior to t. Mathematically, we parameterize the prior serve win rate for player i in the year prior to t with mean μ_{it} and precision ϕ_{it} as:

$$\text{Beta}(\mu_{it}, \phi_{it}) \quad \text{where} \quad \mu(M_{it}) := \sigma(b_{0t} + b_{1t} * N_{it})$$

To fit the beta regression distribution prior to each player, I then need to plug in (serve win rate, number of serves total) data for players in the past year (that played at least two matches) into the program explained in the previous section (section 3.7.2) to compute priors for each player. From there, I estimate parameters $\hat{\mu}_{it}$ and $\hat{\phi}_{it}$ that maximize the likelihood of the prior for each player i in the year prior to t. Then, I algebraically compute the corresponding parameters $\hat{\alpha}_{it}, \hat{\beta}_{it}$ for the Beta prior distribution as $\hat{\alpha}_{it} = \hat{\mu}_{it}\hat{\phi}_{it}, \hat{\beta}_{it} = (1 - \hat{\mu}_{it})\hat{\phi}_{it}$.

From there, I compute the posterior distribution for the serve win rate of player i in the past year using the Bernoulli- Beta distribution update formula- the shape parameters become ($\hat{\alpha}_{it}$ + number serves won by i in year prior to t, $\hat{\beta}_{it}$ + number serves lost by i in year prior to t), respectively. Finally, I compute the beta shrunken serve win rate s_{it} as the expectation of the posterior distribution $s_{it} := \frac{\hat{\alpha}_{it}+\text{number serves won by i in year prior to t}}{\hat{\alpha}_{it}+\hat{\beta}_{it}+\text{number serves total by i in year prior to t}}$. Again, the Beta shrunken serve win rate value is a compromise between the serve win rate predicted by the prior (that controls for the number of serves a player attempted in the past year) and the player's serve win rate frequency.

3.8 Learning Approach Shrinkage Estimator

Recall that a shrinkage estimator for the mean x of some distribution X with known finite and variance σ^2 given N iid draws $D = [X_1, ..., X_N]$ takes the form:

$$\hat{x}_{\text{shrink}} := (1 - B)\bar{x} + Bx_0$$

where $\bar{x} := \sum_{i=1}^{N} \frac{X_i}{N}$, x_0 is some value we want to "shrink" our estimate towards (typically due to prior information), and $B : (X_1, ..., X_N, X, N) \mapsto [0, 1]$ specifies how much we want to shrink our estimate towards x_0. In the previous sections, we've discussed statistically sound approaches to compute B to have good shrinkage estimates. An alternative is to take a simple machine learning approach. We can split the data D into a training set and a test set, impose a structural form on B, use the training data to learn the optimal B to minimize some error function, and then evaluate the success of our model on the test data.

3.8.1 Learning Approach in Tennis Analysis

Recall that in my dataset, I have data on matches from the ATP tour from 2011-2015. To split the matches into training matches and test matches, I chose a classic temporal division- the 2011-2014 matches are the training set and the 2015 best-of-3-set matches are the test set (where both players have played at least two matches in the year prior to the match). While the training and test data are not perfectly uncorrelated (ie., the serve win rate frequency computed for 2015 test matches depends on 2014 matches (training data), I follow common practice to make this division. I'll also continue with the x notation (as signifying the serve win rate of some player i in the year prior to t) to not have to introduce new variables.

I impose that B takes the form $B := \frac{\sigma/N_{it}}{\sigma/N_{it}+\tau}$ where N_{it} if the number of service points i attempted in the past year and σ, τ are the parameters to learn[21]. Then:

$$\hat{x}_{\text{shrink}}(\sigma, \tau) := (1 - \frac{\sigma/N_{it}}{\sigma/N_{it} + \tau})\bar{x} + \frac{\sigma/N_{it}}{\sigma/N_{it} + \tau}x_0 \tag{19}$$

[21] I constructed the learning shrinkage coefficient, B, to closely resemble that from the James-Stein shrinkage estimator in equation 14. The σ stands in for the variance of an individual serve and the τ stands in for the variance of the distribution from which we draw year-long serve win rates. Nevertheless, the parameters σ, τ shouldn't be fully interpreted in that way. In the James-Stein model, the outcome of service point is modeled as a Bernoulli distribution and so the variance of that distribution necessarily lies in $[0, 0.25]$. There exist similar bounds on the variance of the distribution from which we draw year-long serve win rates since all year-long serve win rates necessarily lie in $[0, 1]$. In this learning approach, I only require that $\sigma, \tau > 0$.

The idea, similar in spirit to the shrinkage estimators discussed before, is that as N_{it} increases, we have more data on our player, and so we want our true serve win rate estimate to more closely resemble the sample serve win rate frequency ($\bar{x}$); when N_{it} is small and we have less data on our player, we want our estimate to more closely resemble our prior expectation (x_0). Our loss function, is simply the log-loss of all matches in our set:

$$L(\sigma, \tau, D) := -\frac{1}{M}\sum_{t=1}^{M} Y_t * \text{log(Pr(Player 1 wins match t } |\sigma, \tau)) + (1 - Y_t) * \text{log((1-Pr(Player 1 wins match t } |\sigma, \tau)))$$

where M is the number of matches played in our set, $Y_t := 1_{\{\text{Player 1 wins match t}\}}$, and Pr(Player 1 wins match t $|\sigma, \tau)$ is the computed probability that player 1 wins the match given σ, τ. We compute Pr(Player 1 wins match t $|\sigma, \tau)$ by using the shrinkage-estimated true season serve win rate probabilities for each player (these are functions of σ, τ found in equation 19), correcting these probabilities for the strength-of-schedule faced by each player, the opponent in match t, and the tournament hosting match t. These corrected values (the same corrections previously used to try to account for the biases) give us our best estimate as to the probability that Player 1 wins a serve on Player 2 at match t, and vice-versa. We feed these service point win probabilities as inputs into the match Markov chain from section 1.1 to compute the estimated probability each player wins the match- that's how we get Pr(Player 1 wins match t $|\sigma, \tau)$.

To train the data, we simply solve the following minimization problem:

$$(\sigma^*, \tau^*) := \text{argmin}_{\sigma,\tau} L(\sigma, \tau, \text{Training Set})$$

To assess the performance of this model, we evaluate $L(\sigma^*, \tau^*, \text{Test Set})$.

4 L_1 Regularized Logistic Regression

Logistic regression is a technique used to classify data into two groups and it can be generalized into classifying into many groups with multinomial logistic regression. In our tennis setting, we only care about two groups- Player A wins the match or Player B wins the match. Mathematically, suppose we're given data $D := (x_i, y_i)_{i=1}^N$ where each x_i is a vector of k characteristics for data point i, $y_i \in \{-1, 1\}$ is a classification for data point i, and N is the number of data points. In the logistic model, given a k dimensional parameter vector β, we have a function:

$$\text{Pr}(y = 1|x; \beta) := \sigma(\beta^T x) := \frac{1}{1 + e^{-\beta^T x}} \tag{20}$$

This function Pr$(y = 1|x; \beta)$ aims to represent the probability that the set of characteristics x have class label $y = 1$. The regression part of logistic regression specifies how we want to pick our parameter β.

For the sake of L_1 Regularization, suppose we impose a zero-mean Laplacian Prior on β: Pr$(\beta; b > 0) = (\frac{b}{2})^k e^{-b||\beta||_1}$. We should interpret this prior as favoring the different components of β near 0. Further, it favors sparsity over L_2 distance from 0. For example, in the $k = 2$ case, if

$||\beta||_1 = A$, then it favors $\beta_1 = [5, 0]$ over $\beta_2 = [3, 3]$, even though $||\beta_1||_2 > ||\beta_2||_2$.

Now, if we want to compute the compute the Maximum-A-Priori (*MAP*) estimate of β given our data, logistic model, and the assumption that all of our data points are independent:

$$\begin{aligned}
MAP(\beta|D; b) &= \text{argmax}_\beta \text{Pr}(\beta|D) \\
&= \text{argmax}_\beta \text{Pr}(D|\beta)\text{Pr}(\beta) \\
&= \text{argmax}_\beta \text{Pr}(\beta)\Pi_{i=1}^{N}\text{Pr}(y = y_i|x_i; \beta) \\
&= \text{argmin}_\beta - \log((\frac{b}{2})^k) - \log(e^{-b||\beta||_1}) - \sum_{i=1}^{N} \text{Pr}(y = y_i|x_i; \beta) \\
&= \text{argmin}_\beta b||\beta||_1 - \sum_{i=1}^{N} \text{Pr}(y = y_i|x_i; \beta) \qquad (21) \\
&= \text{argmin}_\beta - \sum_{i=1}^{N} \text{Pr}(y = y_i|x_i; \beta) \text{ st } ||\beta||_1 \leq c \qquad (22)
\end{aligned}$$

The last two minimization problems give the L_1 Regularized Logistic Regression Problem in two different equivalent forms. The two problems are equivalent because for any given value of b in the first problem, there exists a value of c in the second problem such that the two objective functions are equal (Rockefellar 1970); the opposite is also true. As a result, the minimizing β is the same in both problems. As a first remark, b and/ or c are typically picked via cross-validation. As a second remark, the un-regularized logistic regression problem is

$$MLE(\beta|D; b) = \text{argmin}_\beta - \sum_{i=1}^{N} \text{Pr}(y = y_i|x_i; \beta) \qquad (23)$$

The derivation is quite similar except there's no prior on β since we're in that case computing $MLE(\beta|D)$.

When applying this regularized logistic model to the space of tennis, each data point (x_i, y_i) is data on a match t- x_i are characteristics of the match such as player A serve win rate in the year prior to t, player B serve win rate in the year prior to t, tournament serve win rate in the year prior to t, etc. and y_i indicates where player A or player B won the match. N is the number of matches in our dataset that we'll use to compute the optimal β. As recommended above, I indeed will pick b by cross-validation.

4.1 ALLN 2006 Algorithm for L_1 Regularized Logistic Regression

In this subsection, I aim to summarize the L_1 Regularized Logistic Regression Algorithm of Su-In Lee, Honglak Lee, Pieter Abbeel and Andrew Y. Ng (2006), from now on shortened to ALLN (2006), and dive deeper into some technical points to make the algorithm more accessible. The objective generally solving is to iteratively solve a L_1 regularized logistic regression problem.

4.1.1 Newton-Method- 2nd Order Approximation

For simplicity, to start, suppose we're trying to iteratively solve the unregularized logistic regression problem given in equation 23. For notational reasons, let's call the objective of equation 23: $g(\beta) := -\sum_{i=1}^{N} \Pr(y = y_i|x_i; \beta)$. At iteration j, we have our current parameter $\beta^{(j)}$ and we think about how we want to reach $\beta^{(j+1)}$ to get closer to our optimal β. Recall from calculus that we can expand using a Taylor expansion:

$$g(\beta^{(j)} + \Delta\beta) = g(\beta^{(j)}) + \nabla g(\beta^{(j)})\Delta\beta + \frac{1}{2}\Delta\beta^T \nabla^2 g(\beta^{(j)})\Delta\beta + ...$$

If we truncate this Taylor expansion at the second order term, and we replace $H(\beta^{(j)}) := \nabla^2 g(\beta^{(j)})$, where H is the hessian matrix at $\beta^{(j)}$:

$$g(\beta^{(j)} + \Delta\beta) \approx g(\beta^{(j)}) + \nabla g(\beta^{(j)})\Delta\beta + \frac{1}{2}\Delta\beta^T H(\beta^{(j)})\Delta\beta$$

Our objective is to find $\Delta\beta$ so as to minimize $g(\beta^{(j)}+\Delta\beta)$. Thus, we take the gradient of $g(\beta^{(j)}+\Delta\beta)$ with respect to $\Delta\beta$, and set it equal to 0:

$$\begin{aligned} 0 &= \nabla_{\Delta\beta}(g(\beta^{(j)} + \Delta\beta)) \\ &\approx \nabla_{\Delta\beta}(g(\beta^{(j)}) + \nabla g(\beta^{(j)})\Delta\beta + \frac{1}{2}\Delta\beta^T H(\beta^{(j)})\Delta\beta) \\ &= \nabla g(\beta^{(j)}) + H(\beta^{(j)})\Delta\beta \end{aligned}$$

Solving the last line for $\Delta\beta$ gives $\Delta\beta = -H^{-1}(\beta^{(j)})\nabla g(\beta^{(j)})$. We could immediately set $\beta^{(j+1)}$ equal to $\beta^{(j)} + \Delta\beta$ but recall that $\Delta\beta$ came from an approximation and so we might be able to do better and we also want to avoid divergence. Thus, like in the algorithm, let's define a temporary variable $\gamma^{(j)}$

$$\gamma^{(j)} := \beta^{(j)} + \Delta\beta = \beta^{(j)} - H^{-1}(\beta^{(j)})\nabla g(\beta^{(j)}) \tag{24}$$

It indicates our step direction. Then, we'll define $\beta^{(j+1)}$

$$\beta^{(j+1)} := \beta^{(j)} + t\gamma^{(j)} \tag{25}$$

The parameter $t \geq 0$ is found by a line search (more on that later)[22]. We have just derived the parameter update for the quadratic Newton-Method for finding the minimum of $g(\cdot)$.

[22] I'll explain shortly what is Back-tracking line search. Note that you could also find a "good" t by creating some feasible interval for t (ie., $[0, U]$) and doing a brute force search in $t \in [0, U]$ and pick the t that makes $g(\beta^{(j+1)})$ the smallest.

4.1.2 Alternative Approach to Find $\gamma^{(j)}$

In practice, inverting $H(\beta^{(j)})$ is very expensive and thus sometimes researchers solve the linear equation $-\nabla g(\beta^{(j)}) = H(\beta^{(j)})\Delta\beta$ or employ some factorization on $H(\beta^{(j)})$ to find $\gamma^{(j)}$ (Hauser 2012). In other cases, Quasi-Newton Methods are preferred to not grapple with the Hessian matrix.

In the paper, ALLN documented another way to compute $\gamma^{(j)}$ without involving the Hessian. First, define the define matrix $X \in \mathbb{R}^{kxN}$:

$$X := [x_1 x_2 ... x_N]$$

Next, for $i \in \{1, ..., N\}$, and for the jth iteration, define the diagonal matrix $\Lambda \in \mathbb{R}^{NxN}$ and vector $z \in \mathbb{R}^N$ as:

$$\Lambda_{i,i} := \sigma(\beta^{(j)T} x_i)(1 - \sigma(\beta^{(j)T} x_i)) \tag{26}$$

$$z_i := x_i^T \beta^{(j)} + \frac{(1 - \sigma(y_i \beta^{(j)T} x_i))y_i}{\Lambda_{i,i}} \tag{27}$$

As a result, $\nabla g(\beta^{(j)}) = X\Lambda(z - X^T\beta^{(j)})$ and $H(\beta^{(j)}) = -X\Lambda X^T$. To see these facts, first see that we can rewrite the objective of the logistic regression problem in equation 23 since $y_i \in \{1, -1\}$:

$$g(\beta) = -\sum_{i=1}^{N} \Pr(y = y_i | x_i; \beta) = -\sum_{i=1}^{N} \sigma(y_i \beta^T x_i)$$

Then,

$$
\begin{aligned}
\nabla g(\beta^{(j)}) &= \sum_{i=1}^{N}(1-\sigma(y_i\beta^{(j)T}x_i))y_ix_i \\
&= \sum_{i=1}^{N} x_i\Lambda_{i,i}\frac{(1-\sigma(y_i\beta^{(j)T}x_i))y_i}{\Lambda_{i,i}} \\
&= -X\Lambda X^T\beta^{(j)} + \sum_{i=1}^{N} x_i\Lambda_{i,i}(x_i^T\beta_{(j)} + \frac{(1-\sigma(y_i\beta^{(j)T}x_i))y_i}{\Lambda_{i,i}}) \\
&= -X\Lambda X^T\beta^{(j)} + \sum_{i=1}^{N} x_i\Lambda_{i,i}z_i \\
&= X\Lambda(z - X^T\beta^{(j)}) \\
H(\beta^{(j)}) &= -\sum_{i=1}^{N}\sigma(y_i\beta^{(j)T}x_i))(1-\sigma(y_i\beta^{(j)T}x_i))\cancel{y_iy_i}1x_i^T \\
&= -\sum_{i=1}^{N} x_i\sigma(\beta^{(j)T}x_i))(1-\sigma(\beta^{(j)T}x_i))x_i^T \\
&= -\sum_{i=1}^{N} x_i\Lambda_{i,i}x_i^T \\
&= -X\Lambda X^T
\end{aligned}
$$

As a result of these facts, we can derive an expression for $\gamma^{(j)}$ from equation 24, the update direction for $\beta^{(j)}$, in terms of X, Λ, and z.

$$
\begin{aligned}
\gamma^{(j)} &= \beta^{(j)} - H^{-1}(\beta^{(j)})\nabla g(\beta^{(j)}) \\
&= \beta^{(j)} + (X\Lambda X^T)^{-1}X\Lambda(z - X^T\beta^{(j)}) \\
&= \beta^{(j)} + (X\Lambda X^T)^{-1}X\Lambda z - (X\Lambda X^T)^{-1}(X\Lambda X^T)\beta^{(j)} \\
&= (X\Lambda X^T)^{-1}X\Lambda z
\end{aligned} \tag{28}
$$

If you think carefully about equation 28, $\gamma^{(j)}$ is actually also the solution to a least squares problem:

$$
\gamma^{(j)} = \text{argmin}_\gamma||\Lambda^{1/2}X^T\gamma - \Lambda^{1/2}z||_2 \tag{29}
$$

We've carefully derived an approach to find the update direction $\gamma^{(j)}$ in the second order Newton Method by solving a least squares problem. There exist efficient and accurate iterative methods to approximate unregularized least squares problems and so it's a sound alternative to inverting the hessian in the standard newton method update in equation 24.

4.1.3 Reintroducing the Regularization

To introduce regularization to this iterative least squares approach to solve logistic regression, we augment the minimization problem in equation 29 with the L_1 constraint.

$$\gamma^{(j)} = \text{argmin}_\gamma ||\Lambda^{1/2} X^T \gamma - \Lambda^{1/2} z||_2 \quad \text{s.t. } ||\gamma||_1 \leq c \tag{30}$$

Or equivalently

$$\gamma^{(j)} = \text{argmin}_\gamma ||\Lambda^{1/2} X^T \gamma - \Lambda^{1/2} z||_2 + b||\gamma||_1 \tag{31}$$

Now, when we look for the update direction, $\gamma^{(j)}$, at iteration j, we update $\beta^{(j)}$ according to this L_1 regularized least squares problem.

4.1.4 The ALLN Algorithm (Slightly Modified)

Algorithm 1 L_1 Regularized Logistic Regression- Iterative Lasso Least Squares

1: $\beta^{(0)} \leftarrow \vec{0}$
2: **for** j=0 to maxIterations **do**
3: Compute Λ and z according to equations 26 and 27 using $\beta^{(j)}$
4: Use a Lasso regression solver to compute the solution, $\gamma^{(j)}$, to the constrained minimization problem in equation 31
5: $\beta^{(j+1)} \leftarrow \beta^{(j)} + t\gamma^{(j)}$ where $t \geq 0$ is found using a backtracking line search that minimizes the objective of equation 21, $g(\cdot)$
6: Evaluate the objective function of equation 21 at $\beta^{(j+1)}$
7: **if** stopping condition met **then**
8: **return** $\{$"converged" : True, "beta" : $\beta^{(j+1)}\}$
9: **end if**
10: **end for**
11: **return** $\{$"converged" : False, "beta" : $\beta^{(j+1)}\}$

As a first remark, ALLN title their algorithm "IRLS-LARS" which stands for "Iterated Reweighted Least Squares- Least Angle Regression". Iterated Reweighted Least Squares comes from the update in equation 30: the authors choose to view that minimization problem as a solution to a weighted (weight each data point according to $\Lambda^{1/2}$) least squares problem. The least squares problem is "reweighted" since the weights $\Lambda^{1/2}$ depend on $\beta^{(j)}$ and hence change at each iteration. They use the Least Angle Regression Algorithm with the Lasso modification (Efron *et al.* 2004) to efficiently solve each the regularized least squares problem at each iteration.

In my implementation of the algorithm seen in Algorithm 1, I use sklearn's built-in Lasso regression solver at line (4)[23]. Next, the algorithm above is written for one value of b, the hyperparameter used to determine the amount of regularization. I use cross-validation to pick a good b.

For the stopping condition, there were two natural candidates- stop if consecutive βs were "close enough" according to some distance metric in $\mathbb{R}^k$ or stop if consecutive objective values were "close enough". I chose to use the later stopping condition which introduces a $tol \in \mathbb{R}$ that determines when we considered our sequence of $\beta^{(j)}$ converged. Lastly, there's the notion of backtracking line search to pick the optimal t in line (5) of the algorithm that I'll discuss in the next subsection.

[23] Sklearn also has a built-in Lasso regression solver that uses the LARS method, I didn't find extra success with it.

4.1.5 Backtracking Line Search

Backtracking line search is a method to determine how much to move along a given search direction to minimize a differentiable objective function with known (or estimated) gradient. It starts with a large step-size in the search direction and then gradually reduces the step-size (ie., backtracks) until the objective function decreases by an "adequate" amount.

For the backtracking line search algorithm pseduocode, let f be the differentiable objective function we aim to iteratively minimize. Let x be the starting position, $\nabla f(x)$ be the known (or estimated) gradient of f at x, and let p be the candidate search direction[24]. Also, let $t_0 > 0$ be the maximum candidate step size, and $\tau \in (0, 1)$ and $c \in (0, 1)$ be control parameters.

Algorithm 2 Backtracking Line Search

$\alpha \leftarrow -c(\nabla f(x)^T p)$
$i \leftarrow 0$
while $f(x) - f(x + t_i p) \geq t_i \alpha - \epsilon$ **do**
    $i \leftarrow i + 1$
    $t_i \leftarrow \tau t_{i-1}$
end while
return t_i

The while loop stopping condition, $f(x) - f(x + t_i p) \geq t_i \alpha$ is known as the Armijo–Goldstein condition of sufficient decrease (Burke 2020). I add in the $0 < \epsilon \ll 1$ term to help with convergence of the backtracking line search algorithm. We typically pick t_0 to be fairly large since it took nontrivial effort to find the search direction p.

4.2 Logistic Regression Characteristics- Tennis Analysis

When trying to predict the result of a match x between $p1$ and $p2$ using logistic regression, I still need to specify the characteristics I chose from x to make predictions. I chose a few different sets of characteristics that correspond to the approaches I took to discretize the match in the Markov model:

Serve, return discretization:

- $p1$_serve_win_rate $- p2$_serve_win_rate
- $p1$_return_win_rate $- p2$_return_win_rate
- $p1$_opp_serve_win_rate $- p2$_opp_serve_win_rate
- $p1$_opp_return_win_rate $- p2$_opp_return_win_rate
- $p1$_serves_total $- p2$_serves_total
- $p1$_returns_total $- p2$_returns_total

Service game, return game, tiebreak discretization:

- $p1$_service_game_win_rate $- p2$_service_game_win_rate
- $p1$_return_game_win_rate $- p2$_return_game_win_rate
- $p1$_tiebreak_win_rate $- p2$_tiebreak_win_rate
- $p1$_opp_service_game_win_rate$- p2$_opp_service_game_win_rate
- $p1$_opp_return_game_win_rate$- p2$_opp_return_game_win_rate
- $p1$_opp_tiebreak_win_rate $- p2$_opp_tiebreak_win_rate
- $p1$_service_games_total $- p2$_service_games_total
- $p1$_return_games_total $- p2$_return_games_total
- $p1$_tiebreaks_total $- p2$_tiebreaks_total

[24]As a remark, when conducting gradient descent, $p := \nabla f(x)$ since we search in the direction of the gradient. However, in the Newton method, using our notation above, $p := \gamma^{(j)} \neq \nabla g(x)^T$.

Set discretization:

- $p1$_set_win_rate $-$ $p2$_set_win_rate
- $p1$_opp_set_win_rate $-$ $p2$_opp_set_win_rate
- $p1$_sets_total $-$ $p2$_sets_total

Service game, return game, serve and return for tiebreak discretization:

- $p1$_service_game_win_rate $-$ $p2$_service_game_win_rate
- $p1$_return_game_win_rate $-$ $p2$_return_game_win_rate
- $p1$_serve_win_rate $-$ $p2$_serve_win_rate
- $p1$_return_win_rate $-$ $p2$_return_win_rate
- $p1$_opp_service_game_win_rate$-$$p2$_opp_service_game_win_rate
- $p1$_opp_return_game_win_rate$-$$p2$_opp_return_game_win_rate
- $p1$_opp_serve_win_rate $-$ $p2$_opp_serve_win_rate
- $p1$_opp_return_win_rate $-$ $p2$_opp_return_win_rate
- $p1$_service_games_total $-$ $p2$_service_games_total
- $p1$_return_games_total $-$ $p2$_return_games_total
- $p1$_serves_total $-$ $p2$_serves_total
- $p1$_returns_total $-$ $p2$_returns_total

While there are many other sets of characteristics I could choose, I picked these because they're fairly representative of the kinds of biases I've looked at and I needed to pick something. I have data on matches from 2011-2015- like in the learning model (section 3.8), I use the 2011-2014 as training matches and use the 2015 best-of-3-set matches (where both players had played at least two matches in the year prior to the match) to assess the accuracy of my model. I also normalized my data to not have any bias towards characteristics that naturally take larger values (ie., $|p1\text{\_serves\_total} - p2\text{\_serves\_total}| > |p1\text{\_serve\_win\_rate} - p2\text{\_serve\_win\_rate}|$, usually) because weights on larger characteristics cost less in terms of regularization in how they impact our match prediction.

5 RESULTS

5.1 Loss Function: Log-Loss

To evaluate each of our approaches for predicting tennis matches, we need to specify a loss function that indicates how well we did for each method on a specific set of test matches. Matching the loss function we used for our learning model, I choose log-loss[25] of all matches in our set:

$$L(D; a_i) := -\frac{1}{M}\sum_{t=1}^{M} Y_t * \text{log(Pr(Player 1 wins match t}|a_i)) + (1 - Y_t) * \text{log((1-Pr(Player 1 wins match t}|a_i)))$$

where M is the number of matches played in our set, $Y_t := 1_{\{\text{Player 1 wins match t}\}}$, and Pr(Player 1 wins match t$|a_i$) is the estimated probability that player 1 wins the match computed by approach a_i. As a remark, I choose to evaluate each of our approaches on the set of best-of-3-set matches in 2015 (where both players had played at least two matches in the year prior to the match)- that's the test set.

To interpret the log-loss expression, as indicated by the word-choice "loss", an approach a_i is "better" than another if it's estimator has smaller loss. When computing the log-loss, for each match, we add the log of the estimated probability that we attributed to the result that happened. The smaller the

[25]The log-loss function takes an analogous form to the objective of the maximum likelihood estimator for estimating the probability parameter p of some binomial distribution Bin(n, p) given n iid outcomes from the distribution. The derivation and correspondence is fairly straightforward after choosing to maximize the log of the likelihood of the data- it is tangential and left out for brevity.

probability we attribute to the event that happened, the more negative the number we'll add in the sum which will result in a larger contribution to the log-loss. In other words, if player 1 wins the match (ie., $Y_t = 1$), the second term equals 0 and so we just add log(Pr(Player 1 wins match t$|a_i$)) to the log-loss quantity. In numbers, if player 1 wins the match and with approach a_i we estimated he had a 20% chance of winning the match, we'll add $\log(20\%)$ to our log-loss. In the end, we divide the sum of logs from each match by M, the number of matches, so that our result can be interpreted as a per match quantity. As a remark, by dividing by M, we're also implicitly specifying that we weight each match equally when evaluating our approach a_i.

Finally, from log-loss, we can generate an implied average prediction probability for approach a_i as $iapp(a_i) := e^{-L(D;a_i)}$. This quantity can be interpreted as– if we somehow knew the winner and always attributed probability $iapp(a_i)$ to him, we would get that same log-loss $L(D; a_i)$ quantity. As a sanity check, if $iapp(a_i) < 0.5$, then approach a_i predicts the winner worse than random guessing (according to log-loss). If $iapp(a_i) > 0.5$, then approach a_i estimates winner probabilities better than random guessing (according to log-loss).

5.2 Results Tables

I separate the results by approach used to discretize the tennis match in the Markov model. In table 2, I use service point win probabilities to determine the probability each player wins the match and compute log-loss that way. In table 3, I use service game win probabilities to determine the probability that each player wins the match (I settle a tiebreak using a tiebreak win probability that determines the probability that each player wins the tiebreak). In table 4, I use set win probabilities to determine the probability that each player wins the match. In table 5, I use service game win probabilities to determine the probability that each player wins the match (I settle by using serve win probabilities for each point in the tiebreak).

Approach	Log-Loss	Implied Average Prediction Probability
Random Guessing	0.69315	0.5
Server Sample Frequency	0.67107	0.51116
Returner Correction	0.61758	0.53925
Tourney, SOS Correction	0.61641	0.53988
Oracle Shrinkage (clustered variance)	0.63480	0.53004
Oracle Shrinkage (clustered variance, SOS, tourney)	0.62942	0.53290
Oracle Shrinkage (pooled variance)	0.63519	0.52983
Oracle Shrinkage (pooled variance, SOS, tourney)	0.62847	0.53341
JS Shrinkage (clustered variances)	0.61675	0.53970
JS Shrinkage (clustered variances, SOS, tourney)	0.61713	0.53949
JS Shrinkage (cluster and pooled variances)	0.61573	0.54025
JS Shrinkage (cluster and pooled variances, SOS, tourney)	0.61599	0.54011
JS Shrinkage (sum square deviation and pooled variances)	0.61576	0.54023
JS Shrinkage (sum square deviation and pooled variances, SOS, tourney)	0.61587	0.54018
Beta Distribution	0.61399	0.54119
Beta Distribution (SOS, tourney)	0.61572	0.54025
Beta Regression	0.61181	0.54237
Beta Regression (SOS, tourney)	0.61383	0.54127
Learning Approach	0.61611	0.54005
L_1 Reg Logistic Regression (ALLN (2006) modified))	0.60266	0.54735
L_1 Reg Logistic Regression (sklearn built-in algorithm)	0.60208	0.54767

Table 2. Using service point win probabilities

Approach	Log-Loss	Implied Average Prediction Probability
Random Guessing	0.69315	0.5
Server Sample Frequency	0.66741	0.51304
Returner Correction	0.62819	0.53356
Tourney, SOS Correction	0.61902	0.53847
Oracle Shrinkage (clustered variance)	0.64043	0.52707
Oracle Shrinkage (clustered variance, SOS, tourney)	0.62334	0.53615
Oracle Shrinkage (pooled variance)	0.63965	0.52748
Oracle Shrinkage (pooled variance, SOS, tourney)	0.62154	0.53712
JS Shrinkage (clustered variances)	0.63027	0.53245
JS Shrinkage (clustered variances, SOS, tourney)	0.61595	0.54013
JS Shrinkage (cluster and pooled variances)	0.62957	0.53282
JS Shrinkage (cluster and pooled variances, SOS, tourney)	0.61254	0.54197
JS Shrinkage (sum square deviation and pooled variances)	0.62869	0.53329
JS Shrinkage (sum square deviation and pooled variances, SOS, tourney)	0.61334	0.54154
Beta Distribution	0.62783	0.53375
Beta Distribution (SOS, tourney)	0.61927	0.53834
Beta Regression	N/A	N/A
Beta Regression (SOS, tourney)	N/A	N/A
Learning Approach	0.61332	0.54155
L_1 Reg Logistic Regression (ALLN (2006) modified))	0.60936	0.54370
L_1 Reg Logistic Regression (sklearn built-in algorithm)	0.60927	0.54375

Table 3. Using service game win probabilities with tiebreak decided by parameter

Approach	Log-Loss	Implied Average Prediction Probability
Random Guessing	0.69315	0.5
P_1 Set Frequency	0.65854	0.51761
Returner Correction	0.61621	0.53999
Tourney, SOS Correction	0.61845	0.53878
Oracle Shrinkage (clustered variance)	0.63541	0.52972
Oracle Shrinkage (clustered variance, SOS, tourney)	0.62515	0.53519
Oracle Shrinkage (pooled variance)	0.63287	0.53107
Oracle Shrinkage (pooled variance, SOS, tourney)	0.62040	0.53773
JS Shrinkage (clustered variances)	0.61306	0.54169
JS Shrinkage (clustered variances, SOS, tourney)	0.60751	0.54471
JS Shrinkage (cluster and pooled variances)	0.61063	0.54301
JS Shrinkage (cluster and pooled variances, SOS, tourney)	0.60090	0.54832
JS Shrinkage (sum square deviation and pooled variances)	0.61146	0.54256
JS Shrinkage (sum square deviation and pooled variances, SOS, tourney)	0.60422	0.54650
Beta Distribution	0.60892	0.54394
Beta Distribution (SOS, tourney)	0.60480	0.54618
Beta Regression	0.61010	0.54330
Beta Regression (SOS, tourney)	0.62912	0.53306
Learning Approach	0.59715	0.55037
L_1 Reg Logistic Regression (ALLN (2006) modified))	0.60626	0.54539
L_1 Reg Logistic Regression (sklearn built-in algorithm)	0.60641	0.54531

Table 4. Using set win probabilities

5.3 Analyzing the Results

While it's hard to concretely interpret the log-loss of each approach as an absolute quantity, the random guessing log-loss and the "Implied Average Prediction Probability" provide some nice reference points. A random guessing approach would try to predict the result of each match by randomly picking a winner. In other words, if an approach did worse than random guessing on average, it would be a bad approach; fortunately all of the approaches to better than random guessing.

In rows 2-4 of each results table, where there is no shrinkage, the biggest jump in improvement in

Approach	Log-Loss	Implied Average Prediction Probability
Random Guessing	0.69315	0.5
Server Sample Frequency	0.67161	0.51089
Returner Correction	0.61650	0.53983
Tourney, SOS Correction	0.60601	0.54552
Oracle Shrinkage (clustered variance)	0.63355	0.53070
Oracle Shrinkage (clustered variance, SOS, tourney)	0.61816	0.53894
Oracle Shrinkage (pooled variance)	0.63320	0.53089
Oracle Shrinkage (pooled variance, SOS, tourney)	0.61772	0.53917
JS Shrinkage (clustered variances)	0.61640	0.53988
JS Shrinkage (clustered variances, SOS, tourney)	0.60537	0.54587
JS Shrinkage (cluster and pooled variances)	0.61493	0.54068
JS Shrinkage (cluster and pooled variances, SOS, tourney)	0.60419	0.54652
JS Shrinkage (sum square deviation and pooled variances)	0.61469	0.54081
JS Shrinkage (sum square deviation and pooled variances, SOS, tourney)	0.60429	0.54646
Beta Distribution	0.61339	0.54151
Beta Distribution (SOS, tourney)	0.60341	0.54694
Beta Regression	0.61081	0.54291
Beta Regression (SOS, tourney)	0.60428	0.54647
Learning Approach	0.60399	0.54663
L_1 Reg Logistic Regression (ALLN (2006) modified))	0.60019	0.54871
L_1 Reg Logistic Regression (sklearn built-in algorithm)	0.59958	0.54904

Table 5. Using service game win probabilities with tiebreak decided by serve win probability

log-loss comes when I do "returner correction" from section 2.2. When we're trying to think about the probability that A wins a serve on B, it makes sense that we'll have a more accurate estimate when we not only factor how well A serves, but also how well B returns.

Of the shrinkage estimators, the James-Stein estimators do perform better than the Oracle estimators. The James-Stein estimators take into account information about all players when yielding their estimate while the Oracle estimator only takes into account information about the player of interest- the observation that James-Stein performs better than Oracle is likely a consequence of this fact.

The Beta distribution, Beta regression[26], learning approach, and logistic regression approaches perform best of all estimators. It's nice to see that the Beta regression approach does perform better than the beta distribution approach since the Beta distribution approach is simply a special case of the Beta regression distribution as I explain in section 3.7.2. It's also nice to see that the ALLN (2006) algorithm I implemented performs as well as the native sklearn regularized logistic regression algorithm.

Of the different ways of modeling the tennis match, it seems like "Using set win probabilities" from 4 and "Using service game win probabilities with tiebreak decided by serve win probability" from 5 perform best when trying to predict the outcome of the match. In other words, when you model the tennis match using a Markov model with iid serves given the server, it works better if you discretize the match at the service game or set level than at the point level. Also, it appears that modeling the tiebreak using serve win probabilities is better than using the native tiebreak parameter. That's because players over the course of the year don't play many tiebreaks[27] and so the tiebreak parameter is very noisy.

[26]The N/A in table 3 signifies that the beta regression model did not converge when fitting on the data. Specifically, there parameter which failed to converge was the tiebreak win probability parameter.

[27]Players play approximately 14 tiebreaks per year on average

One other remark is I only tried to predict best-of-3-set matches. As a result, the matches have implied average prediction probabilities that hover just over 0.5 because the outcomes of these matches are fairly noisy. If I try to predict best-of-5-set matches, these approaches give log-loss of 0.47 which implies an average prediction probability of 0.625. In best-of-5-set matches, since the match is longer, there's more time for the better player to exert his abilities over the worse player[28]. Thus, these best-of-5-set matches are easier to predict than the best-of-3-set matches.

6 CONCLUSION

I aimed to derive some useful biased and shrinkage estimators in light of tennis data. I took the approach of modeling tennis matches as a Markov model with the additional assumption that all serves in a match are iid given the server, discretized the match at various levels, and solved for the implied probability that each player had to win the match given the discretization. For instance, given probabilities that one player wins a serve against another, I computed the likelihood that each player won the match.

For my own learning, I included many of the derivations behind the estimators and how biased estimators can have lower mean squared error than unbiased estimators. I also spoke about reasons for biased estimators in the tennis setting. For instance, how to predict a match between A and B when A and B may have played a very different number of matches in the past year.

In the end, I also implemented a nice regularized logistic regression algorithm from the ALLN (2006) paper. The purpose here was to take an approach that didn't involve the Markov model of the tennis match, to predict tennis matches.

[28] A similar pattern can be noticed in the sport of basketball. In the NCAA men's basketball tournament, it's a single elimination tournament, and so there are many upsets in which the lower rated team defeats the higher rated team. Meanwhile, in the NBA playoffs, to advance in the series a team needs to win 4 matches and so there are many less upsets.

REFERENCES

Tennis. URL: https://en.wikipedia.org/wiki/Tennis.

Introduction to Buhlmann credibility, February 2, 2010. URL: https://mathmodelsblog.wordpress.com/2010/02/02/introduction-to-buhlmann-credibility/.

Fifth-set tie-breaks to go to 10 points at the Grand Slams, March 16, 2022. URL: https://www.marca.com/en/tennis/2022/03/16/62324494e2704e82a28b459c.html.

Lawrence D. Brown. In-season prediction of batting averages: A field test of empirical Bayes and Bayes methodologies. *The Annals of Applied Statistics*, 2(1):113 – 152, 2008. URL: https://doi.org/10.1214/07-AOAS138.

James Burke. Line Search Methods, 2020. URL: https://sites.math.washington.edu/~burke/crs/408/notes/nlp/line.pdf.

Francisco Cribari-Neto and Achim Zeileis. Beta Regression in R. *Journal of Statistical Software*, 34(2):1–24, 2010. URL: https://www.jstatsoft.org/index.php/jss/article/view/v034i02.

Norman R. Draper and R. Craig van Nostrand. Ridge Regression and James-Stein Estimation: Review and Comments. *Technometrics*, 21(4):451–466, 1979. URL: http://www.jstor.org/stable/1268284.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least Angle Regression. *The Annals of Statistics*, 32(2):407–451, 2004. URL: http://www.jstor.org/stable/3448465.

Bradley Efron and Carl Morris. Data Analysis Using Stein's Estimator and its Generalizations. *Journal of the American Statistical Association*, 70(350):311–319, 1975. URL: http://www.jstor.org/stable/2285814.

Merrill Fabry. Why Is Tennis Scored So Weirdly?, August 21, 2019. URL: https://time.com/5040182/tennis-scoring-system-history/.

Ralph R. Frerichs. *Rapid Surveys*, chapter 5. 2004. URL: http://www.ph.ucla.edu/epi/rapidsurveys/RScourse/chap5rapid_2004.pdf.

Thomas Gilovich, Robert Vallone, and Amos Tversky. The Hot Hand in Basketball: On the Misperception of Random Sequences. *Cognitive Psychology*, 17(3):295–314, 1985. URL: https://www.sciencedirect.com/science/article/pii/0010028585900106.

Jacob Gollub. Producing Win Probabilities for Professional Tennis Matches from any Score. *Bachelor's thesis, Harvard College*, 2017. URL: http://nrs.harvard.edu/urn-3:HUL.InstRepos:41024787.

Christian Graf. A Guide to the Regression of Rates and Proportions, February 18, 2021. URL: https://towardsdatascience.com/a-guide-to-the-regression-of-rates-and-proportions-bcfe1c35344f#:~:text=The%20general%20idea%20of%20the,by%20maximizing%20the%20corresponding%20likelihood.

P. J. Green. Iteratively Reweighted Least Squares for Maximum Likelihood Estimation, and some Robust and Resistant Alternatives. *Journal of the Royal Statistical Society. Series B (Methodological)*, 46(2):149–192, 1984. URL: http://www.jstor.org/stable/2345503.

Kris Hauser. B553 Lecture 6: Multivariate Newton's Method and Quasi-Newton methods, January 25, 2012. URL: https://people.duke.edu/~kh269/teaching/b553/newtons_method.pdf.

W. James and Charles Stein. Estimation with Quadratic Loss. *Berkeley Symposium on Mathematical Statistics and Probability*, 4.1:361–379, 1961. URL: http://www.stat.yale.edu/~hz68/619/Stein-1961.pdf.

Aerin Kim. Beta Distribution — Intuition, Examples, and Derivation, January 08, 2020. URL: https://towardsdatascience.com/beta-distribution-intuition-examples-and-derivation-cf00f4db57af.

Sarah Kirkham. The longest match in history, June 24, 2015. URL: https://www.wimbledon.com/en_GB/news/articles/2015-06-24/the_longest_match_in_history.html.

Franc J G M Klaassen and Jan R Magnus. Are Points in Tennis Independent and Identically Distributed? Evidence From a Dynamic Binary Panel Data Model. *Journal of the American Statistical Association*, 96(454):500–509, 2001. URL: https://doi.org/10.1198/016214501753168217.

Su-In Lee, Honglak Lee, Pieter Abbeel, and Andrew Y. Ng. Efficient L1 Regularized Logistic Regression. In *AAAI*, pages 401–408, 2006. URL: http://www.aaai.org/Library/AAAI/2006/aaai06-064.php.

Francesco Matteazzi and Francesco Lisi. A follow-up study on the issue of i.i.d. points in tennis, 2017. URL: http://www.mathsportinternational2017.math.unipd.it/slides/MS2017_Matteazzi.pdf.

Joshua B. Miller and Adam Sanjurjo. Surprised by the Hot Hand Fallacy? A Truth in the Law of Small Numbers. *Econometrica*, 86(6):2019–2047, 2018. URL: http://www.jstor.org/stable/44955325.

Thomas P. Minka. A comparison of numerical optimizers for logistic regression, March 26, 2007. URL: https://tminka.github.io/papers/logreg/minka-logreg.pdf.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

R. Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, 1970. URL: http://www.jstor.org/stable/j.ctt14bs1ff.

Eric Rasmusen. Understanding Shrinkage Estimators: From Zero to Oracle to James-Stein. 2021. URL: http://www.rasmusen.org/papers/shrinkage-rasmusen.pdf.

David Robinson. Understanding the beta distribution (using baseball statistics), December 20, 2014. URL: http://varianceexplained.org/statistics/beta_distribution_and_baseball/.

David Robinson. Understanding beta binomial regression (using baseball statistics), May 31, 2016. URL: http://varianceexplained.org/r/beta_binomial_baseball/.

Jeff Sackmann. tennis_pointbypoint, September 14, 2017. URL: https://github.com/JeffSackmann/tennis_pointbypoint.

Chris Said. Empirical Bayes for multiple sample sizes, May 3, 2017. URL: https://chris-said.io/2017/05/03/empirical-bayes-for-multiple-sample-sizes/.

Charles Stein. Inadmissibility of the Usual Estimator for the Mean of a Multivariate Normal Distribution. *Berkeley Symposium on Mathematical Statistics and Probability*, 3.1:197–206, 1956. URL: http://www.stat.yale.edu/~hz68/619/Stein-1956.pdf.

Keyvon Vafa. Is the Hot Hand Fallacy a Fallacy?, January 09, 2017. URL: http://keyonvafa.com/hot-hand/.

APPENDIX

Hot Hand Principle

If you recall in section 1.1.4 (Costs and Benefits of the Markov Model), in my using of the Markov model for tennis matches, I undertook the additional simplifying assumption that all points in a match are independent and all service points for a particular server are identically distributed. The "hot-hand principle", the occurrences of streaks in "random" sequences, has been debated academic literature. In tennis, for instance, if a player has won 10 points in a row a believer in the "hot-hand principle" would think the likelihood they win the next point is larger than the probability they win the typical point. A non-believer would think the likelihood they win the next point is equal to the probability they win the typical point. Papers such as Gilovich, Vallone, and Tversky (1985), argue that the "hot-hand principle" is a fallacy and Sanjurju and Miller (2018), in a beautiful result, argue that GVT's selection of streaks to examine is biased, invalidating their work. In this appendix section, I want to present ST (2018)'s key insight.

Suppose you flip a fair coin ("H" or "T") 50 times. Every time you flip an "H", you write down the result of the next flip. What is the expected proportion of "H" you will write down? It's actually less than 1/2. To see that, consider the experiment run with 3 total flips as done in the ST (2018) paper.

3-flip sequence	# of recorded flips	proportion of "H" on recorded flips
TTT	0	-
TTH	0	-
THT	1	0
HTT	1	0
THH	1	1
HTH	1	0
HHT	2	1/2
HHH	2	1

Table 6. Streaks with 3 Coin Flips

Since each 3-flip sequence is an equally likely outcome of 3 total coin flips, the expected proportion of "H" on recorded flips is $5/12 < 1/2$. What does this mean? We know that flips of a fair coin are independent and the expected proportion of "H" is less than 1/2. Suppose we're trying to see if tennis service points of a player are subject to the "hot-hand principle" and our null hypothesis is that all service points are won iid with probability p. After a streak, the player doesn't need to win a fraction of serves that's greater than p to be considered "hot", in fact, they can win a fraction of serves that's less than p and be considered "hot".

Let's see what's going on mathematically. Suppose that we have a sequence of n iid random variables $D := (X_1, ..., X_n)$ where each $X_i \sim$ Bernoulli(p). Interpret each X_i as a coin flip that can land "H" or "T" and P$(X_i =$ "H"$) = p$. We want to find the expected probability of "H" for trials that follow k consecutive "H"s. Let $S_k(D) := \{i$ s.t. the ith flip follows k consecutive "H"s$\}$. Then, we want to show that for any randomly selected subsequence of D with k consecutive "H", the conditional likelihood of getting "H" on the next flip is smaller than p, the unconditional likelihood of getting an

"H" on the next flip.

Proof.

Let t be some arbitrary element of $S_k(D)$ and let i be a uniformly random chosen element from $S_k(D)$. We want to show that $\text{Pr}(X_t = \text{"H"}|i = t) < \text{Pr}(X_t = \text{"H"}) = p$.

We know that:

$$(t \in S_k(D) \ \ \& \ \ X_t = \text{"H"}) \Rightarrow t + 1 \in S_k(D)$$
$$(t \in S_k(D) \ \ \& \ \ X_t = \text{"T"}) \Rightarrow t + 1 \notin S_k(D)$$

So, if $X_t = \text{"H"}$, the cardinality of $S_k(D)$ is larger than if $X_t = \text{"T"}$. That implies, the likelihood that our uniformly randomly selected index i from $S_k(D)$ equals t is smaller in the case that $X_t = \text{"H"}$. So, $\text{Pr}(i = t|X_t = \text{"H"}) < \text{Pr}(i = t|X_t = \text{"T"})$. Now,

$$\begin{aligned}
&\text{Pr}(i = t|X_t = \text{"H"}) < \text{Pr}(i = t|X_t = \text{"T"}) \\
&\Rightarrow \frac{\text{Pr}(X_t = \text{"H"}|i = t)\text{Pr}(i = t)}{\text{Pr}(X_t = \text{"H"})} < \frac{\text{Pr}(X_t = \text{"T"}|i = t)\text{Pr}(i = t)}{\text{Pr}(X_t = \text{"T"})} \\
&\Rightarrow \frac{\text{Pr}(X_t = \text{"H"}|i = t)}{\text{Pr}(X_t = \text{"H"})} < \frac{\text{Pr}(X_t = \text{"T"}|i = t)}{\text{Pr}(X_t = \text{"T"})} \\
&\Rightarrow \frac{\text{Pr}(X_t = \text{"H"}|i = t)}{p} < \frac{1 - \text{Pr}(X_t = \text{"H"}|i = t)}{1 - p} \\
&\Rightarrow (1 - p)\text{Pr}(X_t = \text{"H"}|i = t) < p(1 - \text{Pr}(X_t = \text{"H"}|i = t)) \\
&\Rightarrow p > \text{Pr}(X_t = \text{"H"}|i = t)
\end{aligned}$$

□

Generalized Linear Model for Match-up Serve Win Rate

In the sections above, I imposed a specific process to generate match-up serve win rates. In a tennis match between A and B at time and tournament t, I was interested in finding the quantity s_{ABt} that was the probability A won a serve on B in match t. I used the year long serve win rate of player A, swr_{At}, year long return win rate of player, rwr_{Bt}, did some shrinkage on both values and accounted for other biases to produce s_{At}, r_{Bt}. Then, I defined $s_{ABt} := \frac{(s_{At})(1-r_{Bt})}{(s_{At})(1-r_{Bt})+(r_{Bt})(1-s_{At})}$ as in section 2.2. Here, I propose an alternate approach to estimate s_{ABt}.

Model Construction

Suppose that all tennis players i at time t are endowed with characteristics a_{itS}, a_{itR} that measure the strength of i as a server and the strength of i as a returner, respectively. In a match between A and B at time t, we say that the probability that A wins a serve on B is

$$s(a_{AtS}, a_{BtR}) := \sigma(b_0 + b_1 a_{AtS} + b_2 a_{BtR}) = \frac{1}{1 + exp(-(b_0 + b_1 a_{AtS} + b_2 a_{BtR}))} \tag{32}$$

where σ is the sigmoid function. We want $s(a_{AtS}, a_{BtR})$ to be increasing in A's serving ability and decreasing in B's return ability. Given data on a bunch of matches, we want to estimate $b_0, b_1, b_2, a_{AtS}, a_{AtR}, a_{BtS}, a_{BtR}$.

Estimation Approach

For a match between A and B at time t, again, restrict the dataset to all matches in the immediate year prior to t to produce the set of matches A_t where $M := |A_t|$. Also, let N be the number of distinct players in matches in A_t.

Let $a := [a_{1tS}, ..., a_{itS}, ..., a_{MtS}, a_{1tR}, ..., a_{itR}, ..., a_{NtR}]^T$. For each match m in A_t between i and j construct $x_{m1} := [0, ...0, 1, 0, ...0, 1, 0, ..., 0]^T$ where x_{m1} equals one at the index where a has a_{itS} and a_{jtR} and else 0. Also construct $x_{m2} := [0, ...0, 1, 0, ...0, 1, 0, ..., 0]^T$ where x_{m2} equals one at the index where a has a_{jtS} and a_{itR} and else 0. Then construct the design matrix $X = [x_{11}x_{12}...x_{m1}x_{m2}...x_{M1}x_{M2}]^T$.

Next, let y_{m1} be the realized serve win rate by player i on player j in match m and y_{m2} be the realized serve win rate by player j on player i in match m. Construct $y := [y_{11}, y_{12}, ..., y_{m1}, y_{m2}, ..., y_{M1}, y_{M2}]^T$.

To understand the estimation approach, let $b_1, b_2 = 1$ in equation 32. Under this restriction and our model, for any $x \in X$ defining a serve direction i on j in a match m, it is also true that $s(a_{itS}, a_{jtR}) = \sigma(b_0 + a^T x)$. I can in turn, try to estimate a as a generalized linear model where X is the design matrix and y is the target. The values b_1, b_2 are necessarily fixed for a given generalized linear model regression but I can find their optimal quantities using cross-validation.

I suspect that there are better estimation strategies for this model that take advantage of the fact that X has lots of columns and is very sparse. I need to read up on this. As one idea, there may exist sets of players that only played against each other. In that case, their serve and return strength values can be estimated separately from the serve and return strength values of the other players. As another idea, maybe we can make the targets of the generalized linear model regression to be James-Stein shrunken serve win rates for the match. As a final note, there may be better objectives (error functions) to try in estimation.

Results

Using this model of serve point win rates, plugging these rates into the Markov model for the tennis match, I achieve a log-loss of 0.64799 (which implies an average implied prediction probability of 0.52310) on predicting tennis matches in the set of best-of-3 matches from 2015 (where both players had played at least two matches in the year prior). I suspect my estimation strategy may have not perfectly behaved due to the number of columns of X- I believe that a better log-loss is achievable with this model.

I did try another objective function for my estimation: I set up a weighted generalized linear regression where weights were equal to the inverse of the variance of the data point. In other words, the more serves in a match from A to B, the more confident we were that the serve win rate (from A to B) was closer to the truth, and so the greater we weighted that data point in the loss function for

estimating serve and return strengths. This approach for estimating serve and return strengths yielded a worse log-loss when plugging in the implied serve win probabilities into the Markov Model for tennis matches (log-loss: 0.6775, implied average prediction probability: 0.5079).

Currently, other extension of this model are out of scope.