Applying Bayesian Hierarchical Modeling and Decision Boundaries to Fantasy Basketball

Daniel Illera and Caleb Skinner

December 10, 2024

1 Introduction

Fantasy sports are a popular way for fans to follow and interact with their favorite sport. This has become more true in an era where laws on gambling have been relaxed in various states, making fantasy both more profitable and popular than ever. Typically, it is played throughout the season, and how it is structured is determined by the sport it is based on.

the season and structure of a fantasy sport runs in conjunction with that of a traditional sport. In the case of Fantasy Basketball, before the season begins, users conduct a draft to select various players to their team. Once teams are formed, users will play against other users every week for the duration of the NBA season. After each real life NBA game i, athletes will earn "fantasy points" based on the following equation,

$$\begin{aligned}\text{SleeperScore}_i = \frac{1}{2}\text{Points}_i + \text{Rebounds}_i + \text{Assist}_i + 2\text{Steals}_i + 2\text{Blocks} - \text{Turnovers}_i + \\ \frac{1}{2}3\text{PM} + \mathbb{1}_{\text{DoubleDouble}_i} + 2\mathbb{1}_{\text{TripleDouble}_i} + 2\mathbb{1}_{\text{Points}_i>40} + 2\mathbb{1}_{\text{Points}_i>50}\end{aligned} \tag{1}$$

To emulate the structure of traditional sports, fantasy teams consists of both starters and bench players. Only players in the starting lineup are able to earn points for the team, so the user must adhere to varying rules to maneuver players to their starting lineup. In the

case of Fantasy Basketball on the Sleeper app, users select nine out of the fourteen player to be in their starting lineup each week. Every team's lineup is required to have at least three guards, three forwards, and a center; the two remaining spots can be filled by any player. One of the main tasks of the user is to determine their team's starting lineup.

In other fantasy games, such as those based off American football, this decision is fairly simple. Players only play one game a week, and once the game starts, users cannot move the player out of the lineup. Thus, selecting a starter amounts to picking the player who has the best chance to perform well and hoping for the best.

NBA players, however, play two to four games a week, but, in the Sleeper app, only one of these games earn the user points. This makes the strategy for which users employ to select their lineups completely different. After each game, users must decide whether to lock in their starters score or wait and hope a future score is higher. Once a score is locked in, it cannot be changed. The player cannot be moved to the bench, and even if they perform better later in the week, their locked score will remain the same. This process continues for each player at each game until the week concludes.

This method of choosing starters, while complicated and daunting at first, adds an extra layer of strategy to the fantasy game. Certain strategies become immediately evident, but others are elusive to even the most experienced players. We employ statistical analysis to form conclusions to help users both select the best lineups and to optimize their lock-in strategies. In recent years, sports betting has become an incredibly lucrative industry, so valuable insights into this decision making process has the chance to help people both financially and personally.

This research project will develop a normal Bayesian model that incorporates Sleeper's own projections on player performance via the priors, along with actual results from the past two seasons. Afterwards, we will develop a framework which can be employed to make

decisions based off of our model. We will conclude the project by showing some of the ways that we have used the model to impact our decision making in our fantasy basketball league.

2 Data Collection

Our data comes from ESPN's NBA box score data and includes various counting stats, such as points, steals, rebounds, and assists, for every NBA player, covering each game from the start of the 2023-2024 season through December 2, 2024. With this data, we compute each player's Sleeper Score per game (1), enabling us to construct informative priors for the variance parameters. The projections that Sleeper provides for each player before the game serve an important role as the prior mean for the hierarchical mean. Unfortunately, Sleeper has not made their projections for previous years available, so we could not utilize this information to form an informative prior on the hierarchal variance. The projections we have are all from this season and are updated as of December 2nd, 2024.

The biggest limitation of our data collection process is our inability to scrape the projections from Sleeper's website, requiring us to copy them manually. Thus, the players we inspect in this report were chosen either from one of the fantasy teams belonging to the authors, or because we thought that the player would be able to highlight a limitation of the model. In total, we include data from 38 players playing 740 games over 6 weeks.

3 Forecasting Sleeper Scores

3.1 Model Specification

Let y_{ij}, θ_{ij} denote the observed and expected Sleeper score for player i during game j. Our goal is to forecast the sleeper scores for player i at a future game j^* to calculate the probability that the player outperforms a previous game. Ideally, we would also like to

incorporate the vast amount of prior information we have available from the 2023 season. The Bayesian paradigm allows to easily accomplish these goals while also accounting for the uncertainty we have in our prior estimates. These prior estimates are the previously mentioned Sleeper projections denoted as μ_{ij}. They likely come from some complex statistical model developed by Sleeper, meaning there is some uncertainty associated with these estimates that we should properly account for. We specify our likelihood and hierarchical model as follows,

$$y_{ij}|\theta_{ij}, \sigma_i^2 \sim N(\theta_{ij}, \sigma_i^2)$$
$$\theta_{ij}|\tau^2 \sim N(\mu_{ij}, \tau^2)$$

One of the advantages of this model is the simple interpretation of each parameter. The variance of the data model, σ_i^2, represents the average squared deviation of the observed score from the expected score. This can be thought of as a measure of consistency. Since some players are more consistent than others, it would not make sense to assume that the variance is constant across them. On the other hand, the hierarchical variance, τ^2, represents the deviations from Sleeper's projections, and therefore quantifies the uncertainty from these estimates.

The Scaled Inverse Chi-Square distribution can serve as an informative conjugate prior for both $\sigma^2)i$ and τ^2. This class of prior encodes both prior degrees of freedom and prior sum of squared deviations from the mean in its shape, ν and rate, ψ, parameters respectively. If we let GP_{2023i} be the number of games played by player i and σ^2_{2023i} that same player's observed score variance in 2023. We can set $\nu_\theta = GP_{2023i} - 1$ and $\psi_\theta = \nu_\theta \sigma^2_{2023i}$, enabling us to incorporate the prior information from the 2023-2024 NBA season into our estimate of σ_i^2. Unfortunately, we do not have access to Sleeper's projections from 2023-2024, so it is difficult to construct an informative prior on τ^2. For this reason, we opted to use an uninformative Inverse Gamma prior for the hierarchical variance. Altogether, we have the following priors

for this model.

$$\sigma_i^2 \sim \text{Inv}{-}\chi^2\left(GP_{2023_i} - 1,\ (GP_{2023_i} - 1)\sigma^2_{2023_i}\right)$$
$$\tau^2 \sim \text{IG}(1, 1)$$

We employed a Gibbs sampler to obtain posterior draws. If we let N_i be the total games played by player i in the 2024-2025 season, then we can easily find the following full conditional distributions

$$\tau^2|- \sim \text{IG}\left(\frac{\sum_i N_i}{2} + 1,\ \frac{\sum_i \sum_{j=1}^{N_i} (\theta_{ij} - \mu_{ij})^2}{2} + 1\right)$$
$$\sigma_i^2|- \sim \text{IG}\left(\frac{N_i + (GP_{2023i} - 1)}{2},\ \frac{\sum_{j=1}^{N_i} (y_{ij} - \theta_{ij})^2 + (GP_{2023i} - 1)\sigma^2_{2023i}}{2}\right)$$
$$\theta_{ij}|- \sim N\left(\frac{\frac{y_{ij}}{\sigma_i^2} + \frac{\mu_{ij}}{\tau^2}}{\frac{1}{\sigma_i^2} + \frac{1}{\tau^2}},\ \left(\frac{1}{\sigma_i^2} + \frac{1}{\tau^2}\right)^{-1}\right)$$

From this sampling scheme, it is simple to obtain draws of $\tilde{y}_{ij^*}$ from the posterior predictive distribution for a new game j^*. According to Equation 2, a posterior predictive sample is obtained by first sampling τ^2 from its posterior sample and using that to sample from $p(\theta_{ij^*}|\tau^2)$, which is centered at μ_{ij^*}. Afterwards, we take a posterior draw of σ_i^2 and use that with θ_{ij^*} to obtain our posterior predictive draw from the likelihood. This process can be accomplished efficiently within the Gibbs sampler by sampling from $p(\theta_{ij^*}|\tau^2)$ and the likelihood at each iteration.

$$p(\tilde{y}_{ij^*}|y_{ij}) = \iiint p(\tilde{y}_{ij^*}, |\theta_{ij^*}, \sigma_i^2, \tau^2) p(\theta_{ij^*}|\tau^2) p(\tau^2|y_{ij}) p(\sigma_i^2|y_{ij})\, d\theta_{ij^*}\, d\sigma_i^2\, d\tau^2 \tag{2}$$

A possible extension to this model is to hierarchically model the scale parameter, al-

lowing the data to determine the contribution of this prior information. Since the scale parameter is prior degrees of freedom, the smaller the value, the less informative the prior becomes. This could be useful in the case where a player's consistency from the previous season does not reflect their consistency in the current season.

4 Decision Rule

Now that we have modeled the performance of these professional athletes, how can we leverage the information gained to optimize the decision-making for a fantasy basketball user? The user has a fairly complex problem full of dozens of decisions each week. Each day, the user must select players to the starting lineup and, once the games are finished, accept (lock) or reject the player's performance. With fourteen players, nine spots in a starting lineup, and a competition lasting seven days, this problem is overwhelming. However, if we start simple, we can draw some tangible and sweeping conclusions.

Single Player Decision Boundary

First, we consider a single player that plays k games in a week. The user can only accept one of the k games for the final score. A naive decision-making approach is to accept an athlete's score in game y_i if it is not likely that it will be exceeded in the following games. Using our model, we can easily calculate the probability that the score will be exceeded, then accept it only if the probability is greater than 50%. At first, this approach seems optimal, but it fails to take into the account the sequential decision-making built into the optimization.

Sequential Game

Looking at a single player, deciding which score to lock is reminiscent of a sequential game commonly used in Game Theory. These games are typically drawn using decision

trees where each node represents a decision and a branch that terminates indicates that an outcome has been realized. A method used to find the "best strategy"is backwards induction. This is done by looking at the final decision node and picking the best outcome, then moving up the tree and continuing this same process until one reaches the beginning. At this point, the optimal strategy should be clear, hopefully allowing one to achieve the best outcome.

Coming back to the example of a single player, we start by looking at the final game, k, they play in the week. This score will be accepted if we chose to reject all other scores before, and thus we will refer to it as μ_k. This means that the final decision a user makes will be immediately after the conclusion of the $(k-1)^{\text{th}}$ game where they must either accept the score, y_{k-1}, they just observed or let the final game play out and be forced to accept μ_k. We call attention to the fact that y_{k-1} is known, however μ_k is not. If it were known, a rational user would obviously select the largest of two scores, let's call this value $T_{k-1} = \max\{y_{k-1}, \mu_k\}$. We note that T_{k-1} is a random variable, additionally since $T_{k-1} \geq y_{k-1}$, then $\mathrm{E}[T_{k-1}|y_{k-1}] \geq y_{k-1}$. What this implies is that if the expected value of T_{k-1} is greater than the score of game $k-1$, we can expect μ_k to be larger and therefore should reject y_{k-1}. To find the expected value of T_{k-1} we must first identify its support and its respective probabilities. Clearly, the support of T_{k-1} is y_{k-1} and μ_k. It will equal y_{k-1} whenever $y_{k-1} \geq \mu_k$, therefore $\mathbb{P}(T_{k-1} = y_{k-1}) = \mathbb{P}(y_{k-1} \geq \mu_k) = p_{k-1}$. We denote $\mathrm{E}[T_{k-1}] = p_{k-1}y_{k-1} + (1-p_{k-1})\mu_k$ as μ_{k-1}. To reiterate, we reject y_{k-1} if $\mu_{k-1}y_{k-1}$.

Continuing on to the $(k-2)^{\text{th}}$ game, upon its conclusion we would have observed y_{k-2}. If we choose to reject this score, then we will obviously expect to receive a score of μ_{k-1}. Therefore, we can set $T_{k-2} = \max\{y_{k-2}, \mu_{k-1}\}$ and find its expected value, μ_{k-2}, in the exact same way we did for T_{k-1}, rejecting y_{k-2} if it is less than μ_{k-2}. From here we continue to proceed recursively until we reach the first game of the week, ultimately finding a decision boundary. If after any game, the score is less than this decision boundary, we reject it.

One important feature of this model is that the decision boundary will start high and

fall as the week progresses. The frame work accounts for the fact that there is less risk in rejecting earlier scores, allowing the user to be a bit greedy in their decision making.

Multiple Player Decision Boundary

One Spot

First, we expand the single-player decision boundary slightly. Imagine a team has two athletes and one spot in the starting lineup. At first this appears much more complicated, but users are restricted to one player in a starting lineup at a time, so we can approach this with similar backward induction strategies as before.

Let's look at a hypothetical example with two players x_1 and x_2. The two players will occasionally play on the same day, and their playing schedule may look something like Table 1.

M	T	W	R	F	Sa	Su
$x_{1,m}$			$x_{1,r}$		$x_{1,sa}$	$x_{1,su}$
$x_{2,m}$	$x_{2,t}$		$x_{2,r}$			$x_{2,su}$

Table 1: Two Player Schedule

In this hypothetical example, both players play on Sunday. Since the user can only select one athlete each day, it is reasonable to assume that the user will select the player with the highest expected value. We term the player in the starting lineup as s and the expected value μ_{su}.

$$s_{su} = \mathbb{1}_{(\mathbb{E}[x_{1,su}]>\mathbb{E}[x_{2,su}])}x_{1,su} + \mathbb{1}_{(\mathbb{E}[x_{2,su}]>\mathbb{E}[x_{1,su}])}x_{2,su}$$
$$\mathbb{E}[T_{su}] = \mu_{su} = \max\{\mathbb{E}[x_{1,su}], \mathbb{E}[x_{2,su}]\}$$

On Saturday, clearly $s_{sa} = x_{1,sa}$ and the user will accept the score of s_{sa} if it exceeds $\mathbb{E}[T_{su}]$. In keeping with former notation, we can term T_{sa} as the expected final score after

Saturday's games.

$$T_{sa} = \max\{s_{sa}, \mathbb{E}[T_{su}]\}$$

The expectation of T_{sa} is simple to compute and represents the decision boundary for games before Saturday.

$$\mathbb{E}[T_{sa}] = \mathbb{E}[\max\{s_{sa}, \mathbb{E}[T_{su}]\}]$$

If only one player plays each day, it is simple to iteratively roll this back until Monday. However, if both players play on the same day (as in our hypothetical Thursday), the decision is more complicated. A naive approach is select the player with the higher expected value for that day. However, since the decision boundary is $\mathbb{E}[T_{sa}]$, the user should select the player most likely to exceed the decision boundary. Each player has a unique variance, so it is possible for a player with a higher expected value to have a lower probability of exceeding the decision boundary. We term p_{x_l} the probability of player x exceeding the decision boundary at game l. Thus, the starting player s_l at game l will be

$$s_l = \mathbb{1}_{(p_{x_{1,l}} > p_{x_{2,l}})} x_{1,l} + \mathbb{1}_{(p_{x_{2,l}} > p_{x_{1,l}})} x_{2,l}$$

Altogether, we can capture a simple decision boundary. The rule allows the user to consider any number of players for one spot in the lineup.

$$\begin{aligned} T_l &= \max\{s_l, \mathbb{E}[T_{l+1}]\} \\ \mathbb{E}[T_l] &= \mathbb{E}[\max\{s_l, \mathbb{E}[T_{l+1}]\}] \\ &\text{Lock if } y_l \geq \mathbb{E}[T_{l+1}] \end{aligned} \tag{3}$$

Multiple Spots

Consider a more flexible situation with m athletes in contention for n spots in the starting lineup. While adhering to the same general principles, this decision boundary is more complex. Let's approach a hypothetical example with five players and two spots. This should make the equations more tangible. We term these players $x_1, ..., x_5$. Their schedule may look something like Table 2.

M	T	W	R	F	Sa	Su
$x_{1,m}$				$x_{1,f}$		$x_{1,su}$
	$x_{2,t}$	$x_{2,w}$			$x_{2,sa}$	
	$x_{3,t}$	$x_{3,w}$				$x_{3,su}$
$x_{4,m}$				$x_{4,f}$		$x_{4,su}$
$x_{5,m}$			$x_{5,r}$		$x_{5,sa}$	

Table 2: Multiple Player Schedule

The user will be placed in two distinct cases. In the first case, there are at least two open spots in the lineup. In the second case, some of the athletes have already been accepted, so only one spot remains. The decision boundary in the second case simplifies to the single selection approach, so we only need to focus on the first case.

The trick is recognizing that each of the user's decisions are sequential. For $n = 2$, the user only needs one decision boundary, because as soon as a player exceeds this decision boundary, the problem simplifies to the aforementioned $n = 1$ case. The same holds for $n = 3$. After an athlete's score is selected, the problem becomes an $n = 2$ problem, etc.

At any given game, the user is only permitted n players in the starting lineup. Let's call these selections $s_{i,l}$ for the i^{th} player and l^{th} game. As with before, the user should choose s_i by selecting the n players that maximize the probability of exceeding the decision boundary. We term this probability $p_{i,l}$. We order the probabilities for each game l by setting the largest probability equal to $p_{l_{(1)}}$, second largest equal to $p_{l_{(2)}}$, and so on. Next, we select the n starting players that have the highest probabilities.

Now, let's consider the user's decision on day $k-1$. Since the user has only one decision boundary (in this step), they should begin with considering the strongest result of the day $s_{(n),k-1}$. If this result exceeds the lowest expected total result on day k, $\mathbb{E}[T_{(1),k}] = \mu_{(1),k}$, then the user should accept. If so, the user should repeat the process by removing this accepted player from the pool.

Working backward, we compute the expected total scores of the $i = 1, ..., n$ spots after game l, $T_{(i),l}$. For example,

$$T_{(n),l} = \{s_{1,l}, ..., s_{n,l}, \mathbb{E}[T_{1,l+1}], ..., \mathbb{E}[T_{n,l+1}]\}_{(2n)}$$

$$...$$

$$T_{(1),l} = \{s_{1,l}, ..., s_{n,l}, \mathbb{E}[T_{1,l+1}], ..., \mathbb{E}[T_{n,l+1}]\}_{(n+1)}$$

As before, we solve for the expectation of these values and use them to form decision boundaries. In words, the user should accept the best performer of day l if the score exceeds the lowest total expected score of day $l+1$. The decision boundary after game l is

$$\text{Lock if } s_{(n),l} > \mathbb{E}[T_{(1),l+1}] \tag{4}$$

There are two major takeaways from this approach. As games are played and the week progresses, the decision boundaries shrink. Conversely, as players are locked, there are less spots available, and the decision boundaries increase. Intuitively, this makes sense. With a higher sample size, the user can choose to be greedy and only accept high scores. However, in the last few days of the week, the user must accept worse outcomes.

5 Results

On our GitHub page, we have posted code that will compute decision boundaries for selecting up to three spots from a pool of any number of players using the backwards induction method. Ultimately, the goal is to expand these decision boundaries to include nine spots and fourteen players. However, while larger boundaries are mathematically feasible, they become more and more expensive computationally due to the nested relationship of the decision boundaries. In practice, fantasy users can divide their fantasy teams into smaller groups of players and use the decision boundaries to make decisions on a smaller scale.

We test the strength of the backwards induction method (and the model on which it rests) by applying it to real NBA games and computing the final scores the users would have received by following the backwards induction decision boundaries. Testing the methods with more than one spot are complicated problems, so we focus on three simple cases: one player for one spot, two players for one spot, and five players for one spot. In the future, it is important to verify the strength of the backwards induction methods on more complex problems (for example, eight players for three spots).

We compare the backwards induction method with three naive methods: the cdf method, expected value method, and do nothing (Nick Di) method.

In the cdf method, we compute the cdf of the maximum sleeper score of the remaining games in the week and set the decision boundary equal to the median value. In other words, the decision boundary is set so that the player has a 50% chance to exceed it at some point in the remaining games. In practice, this is computed simply with the posterior predictive densities of all future games. This method is strong and intuitive, but we argue the cdf method boundaries overestimate the optimal boundaries. The user can only observe one game at a time, so they may be forced into a poor decision.

The expected value method sets the decision boundary for each game in a week to the

largest expected value for any game that week. In general, this method provides satisfactory decision boundaries, particularly for simple problems. However, the method sets static decision boundaries, so the boundaries at game i are equal to game $i + k$. Typically, this leads the method to underestimate the optimal decision boundaries in complex situations.

Third, the do nothing (Nick Di) method sets the decision boundary arbitrarily high, so the user always accepts the last score. This method's name is derived from the failing strategy of one of our classmates.

Overall, the decision boundaries for the cdf and expected value are very similar to the backwards induction method in simple cases. As the problems get more complicated and more players and spots are introduced, we posit that the backwards induction method will be both more computationally feasible and mathematically.

We compare the final scores yielded by the four methods over 127 player-week combinations. We removed each instance where a player was injured during the week and only included weeks 3 and beyond. We display the results in Table 3. Interestingly, the expected value performs slightly better than the cdf and backwards induction methods. While this is unexpected at first, the decision boundaries proposed by the three methods are so similar in simple cases like this, that a few results can have an outsized impact. For instance, in our data set of 127 player-weeks, the expected value and backwards induction decision boundaries induced the same decisions in 80% (101/127) of cases. To be more certain about the optimal decision boundary in a simple situations, we would need more data.

Next, we compare the four methods in a more complex situation, with two players for one spot. Adding a second player drastically increases our sample size, but it confuses the variance, because now many of the samples are correlated. This makes it more difficult to determine statistical significance. We compute the final points using each decision boundary method for all two player combinations of the 38 players. After removing weeks where

Method	Final Points
Backwards Induction	29.20
CDF	29.19
Expected Value	29.36
Nick Di	23.57

Table 3: One Player for One Spot

player's receive an injury, we have 1970 observations. We display the results in Table 4. As expected, the final points for all four methods increase, as user's are able to prioritize the stronger player. In this situation, the backwards induction method performs better than the other three methods. This suggests that the backwards induction method is better suited to scale to more complicated problems.

Method	Final Points
Backwards Induction	32.68
CDF	32.53
Expected Value	32.49
Nick Di	25.41

Table 4: Two Players for One Spot

Last, we compare the methods with five players for one spot. As before, adding five players increases the sample size, but it also increases the correlation between the samples. To save computational cost, we chose a subset of 17 of the 38 players. After removing weeks where player's receive an injury, we have 6573 observations. We display the results in Table 5. Here, we see that the backwards induction method begins to separate from the rest of the group. This contributes to our presumption that the backwards induction method is preferred for complex problems.

Method	Final Points
Backwards Induction	36.81
CDF	33.92
Expected Value	35.75
Nick Di	30.76

Table 5: Five Players for One Spot

6 Discussion

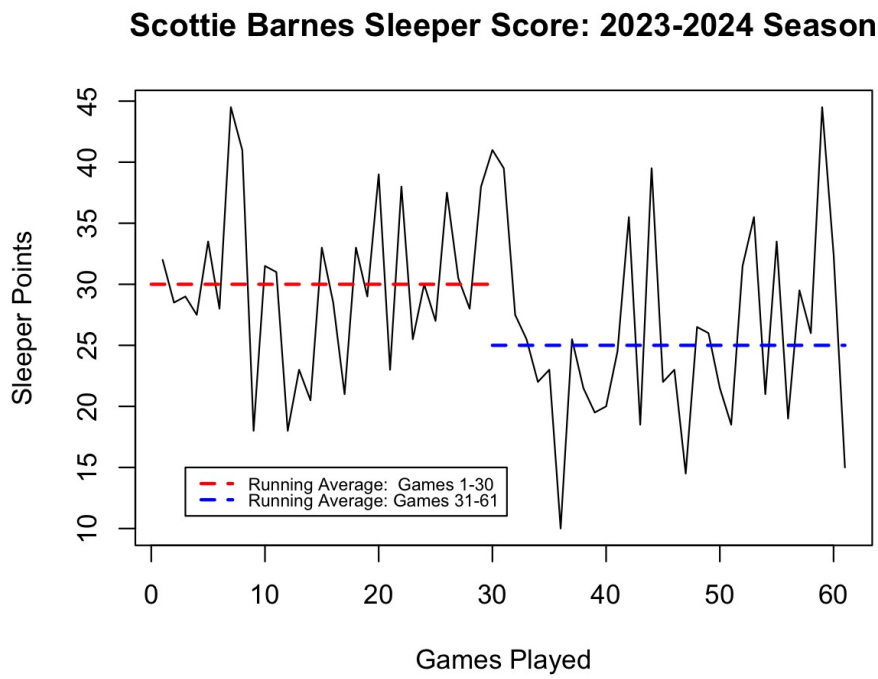
6.1 Limitations of the Model

With this model specification, we make the very strong assumption that Sleeper scores are independent between players. In a few cases, this is certainly false. For example, if a point guard performs well by recording many assists, then his teammates' scores are likely to increase by scoring off of the assists. In general, however, most players are likely to be independent if they are on different teams and not directly competing.

Another potential critique of this model is that it fails to account for dependencies across time. We observed two potential types of temporal relationships. First, a player's role or team changes suddenly in the middle of a season can introduce time dependencies. For example, Figure 1 shows Scottie Barnes' Sleeper score through the course of the 2023-2024 NBA season. Barnes' average Sleeper points suddenly dips in the middle of the season. This happens to coincide with the departure of his teammate OG Annuoby. This major event changed Scottie Barnes' role and affected his performance on the court. A second form of time dependency arises when recent performances affects a players current performance. It can be argued that players performances in the NBA from day to day are independent of each other. However, we do tend to see moments where players consistently perform poorly or well. More research is needed to determine the impact of these momentum swings.

However, we argue that any temporal effect is minimized by our use of Sleeper's projections as the prior mean. It most likely accounts for sudden role changes or stretches of

Figure 1:



strong or poor play. Centering our posterior predictive distributions at Sleeper's projections allows our model to adapt as the season progresses and players change.

Since the model assumes the Sleeper scores follow a normal distribution, the symmetric shape imposes the unreasonable assumption that the probability of outperforming expectations is equal to the probability of underperforming. The best players in the NBA may be unlikely to perform poorly, but at any given game may dish out a monster performance, meaning that their distribution could be right skewed and not symmetric. Similarly, some players may be more likely underperform, so their scores follow a left skewed distribution. As mentioned earlier, the unimodality of the normal distribution also limits this model. Some players, especially younger ones, are very inconsistent and will often have bimodal distributions. The solution for both of these problems would be to have a likelihood which adapts to the different possible shapes each distribution can have. The main reason we opted for the normal likelihood is because it gave us a simple framework to use to construct priors on the

variance parameters. The next step for this model is to employ nonparametric techniques to better adapt to the data.

References

Gelman, A. (2006). Prior distributions for variance parameters in hierarchical models. *Bayesian Analysis*, *1*(3), 515-533. doi: 10.1214/06-BA117A

Appendix

Our R code can be found on GitHub at `https://github.com/CalebSkinner1/SleeperBayesian`. The README can give you instructions on how to run the code and reproduce our results.

Self Reflection

Daniel's Self-Reflection

I already have a decent background in Bayesian statistics, and after spending a summer doing research with Dr.Kowal, my foundation felt pretty solid. Not only was it strengthened, but I also got to learn how to work collaboratively with others. It started with the homework, where some of us first year students would help lead each other to the solution of the problem, in some cases working through it together. With this rhythm already set, it felt natural to find a partner for this project. I approached Caleb with the idea, and we both kind of ran with it.

We started by first ironing out the details of the model, then we had to tackle the problem of creating a decision rule based off of it. I spent a bit of time looking through texts on decision theory, but I couldn't quite find how that could be used for this particular

problem. Caleb and I discussed this, and I brought up sequential games that I had learned in a game theory class almost four years ago. In that moment, I saw a light bulb go off in his head and he absolutely ran with the idea. I cannot begin to describe how happy I am with the product we have today.

The collaboration has been excellent for the both of us. Work was divided up evenly, we both had trust in each other's abilities and were able to work effectively because of it. We both wish we could have dedicated more time, but we both had other commitments that made that difficult to accomplish. We feel did our best, although there is so much more that could be done. I would love to expand this model using Gaussian Processes and to include in this paper the parameter expanded version of the model that I coded up on December 7th. The question, regardless of the lack of "useful"application, is still interesting and can serve as the basis of a tiny interesting research project.

Caleb's Self-Reflection

I have been interested in Bayesian statistics for a long time, but this is the first project where I experienced the creativity and depth of Bayesian Hierarchical modeling. This project encouraged me not only to apply the Bayesian tactics we learned in class, but also to research new innovative solutions to our own complex problem. This fantasy basketball problem is multidimensional and unique; it cannot be solved sufficiently with a cookie-cutter model. We had to think through the problems, research, and consider many solutions. This process was not always easy, but it was incredibly enriching. As I look toward to my next four plus years at Rice and career in statistics, I anticipate this project and this class will represent a major point in my overall growth.

Overall, there's not much I would have changed about this project. Both Daniel and I worked really hard and contributed in a lot of ways to the thought-work and implementation of each part of the project. I enjoyed working with him and learning from his different

way of thinking. I particularly glad that we chose a unique problem of which we were both passionate. This gave us the energy and resolve to work hard. Altogether, I have learned that it's these types of nuanced problems that excite me to do research. Daniel and I hope to continue this project into Christmas Break, and I am so glad that this Bayesian class enticed us to put in the work and draw some really cool conclusions.