Projecting NFL player performances from ordinal data

Paul C. Bogdan[1,2] (pbogda2@illinois.edu)
[1]University of Illinois at Urbana-Champaign
[2]Beckman Institute for Advanced Science and Technology

Abstract: In the past twenty years, statisticians and their models have revolutionized professional baseball and basketball at all levels of team operations. However, effectively modeling players' performances in the National Football League (NFL) remains elusive due to three common pitfalls in modeling NFL data: (1) football games contain many variables, (2) datasets are small, and (3) patterns are non-parametric. I developed a supervised learning approach, which seeks to overcome these challenges. My method translates publicly available expert rankings into probability distributions, predicting each player's likely performance in the upcoming week. The distributions are fit using kernel density estimation, followed by a series of processing steps to eliminate overfitting and noise. The present report overviews these calculations and results. Briefly, I also describe how these projections can be used for Monte Carlo simulations and guide sports gambling.

Keywords: Kernel density estimation, ordinal predictors, non-parametric data, sports

1. Modeling player performances

1.1. Background

The National Football League (NFL) is the premier league for professional American football. My goal is to predict how well individual football players will perform. However, statistically modeling NFL data is notoriously difficult for three broad issues, which compound each other: (1) Football games are complex. At any given moment, twenty-two players are on the field, interacting in different ways. In this sense, football games differ widely from baseball games, which can be seamlessly divided into battles between one batter and one pitcher. (2) Sample sizes are extremely limited. The are only 32 NFL teams, and each team plays only 16-17 games per season. This contrasts Baseball and Basketball seasons, which are 162 games and 82 games long respectively. (3) Player performances are extremely non-parametric. On good days, the best players may drastically beat the average, and capturing this right tail is vital for successful modeling.

A large number of features, a limited sample size, and an intricate target distribution create fertile ground for overfitting models. Hence, to model player performances, I pursued a framework with just a single predictor variable, albeit a multifaceted one – namely, I used the rankings generated by public experts.

Hundreds of experts provide weekly rankings on how well they believe each player will perform, relative to players of the same position. For instance, experts specify each week who they believe will be the best *running back* in the upcoming set of games (the RB1), who will be the second-best running back (the RB2), and so forth. Such rankings are generated for every relevant position (e.g., quarterback, wide receiver, etc.). Based on expert rankings, my models predict how each player will perform. In a sense, leveraging ranking data is a form of dimensionality reduction: Experts take in the full complexity of a football game and attempt to distill it into a single ordinal measurement.

1.2. Methods for projecting likelihood distributions

I scraped the historic performances of every player from every NFL game from 2013-2021. Player performances are measured as Fantasy Points, a metric whereby better play yields more points, which will be used for the sports betting contest detailed in Section 2. I also retrieved historic rankings for each player over the same period. Rankings were averaged across all experts to create an overall ranking for each position. Both scores and rankings data are represented for a given position by a matrix X, where x_{ij} corresponds to points scored in week j by the player ranked i. Figure 1 visualizes the X matrix for the running back position.

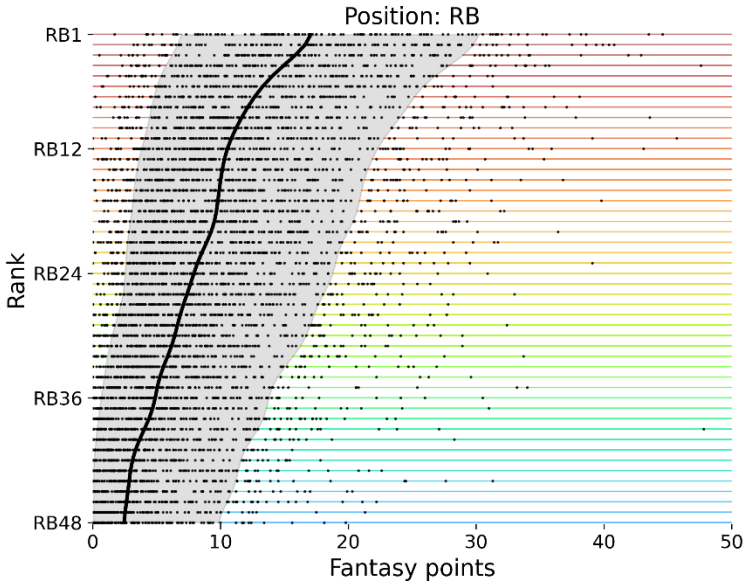

Figure 1. Historic data for the running back (RB) position. This scatterplot represents every performance from 2013-2021 for running backs ranked between RB1 (best) and RB48 (worst). The black line indicates the median performance associated with every expert ranking with the shaded region indicating the middle 80% of performances.

1.2.1. Kernel Density

From X_i (vector representing all scores of players ranked i), I generated a distribution representing the likelihood a given player of rank i achieves x points,

$$p_i(x \mid X_i, \sigma_i).$$

I calculated this distribution via kernel density estimation using an exponential kernel of σ_i width. Notably, I allowed σ_i to vary between ranks because it improves the quality of the distribution: Poorly ranked players cluster around low scores (implying low σ_i), whereas better ranked players cover a wider range of scores (high σ_i). See Figure 1 and note how the 48th-ranked RBs are generally limited to 0-10 points, but the 1st-ranked RBs span 7-30 points.

I chose the optimal kernel width (σ_i) for each rank i by assessing each distribution's accuracy during k-fold cross-validation. For each cross-validation fold (k), one year of player data (X^k) was held out, and the density distribution was calculated based on the remaining training data (X^{-k}),

$$p_i(x \mid X^{-k}, \sigma_i).$$

I used the hold-out fold was to assess the distribution's accuracy. I scored accuracy as the log-likelihood of the held-out data (X^k) given the trained density distribution, $\mathrm{LL}(p_i(X^k \mid X_i^{-k}, \sigma_i)$. I averaged accuracy across all folds (K), and chose σ_i such that it maximized this accuracy,

$$\sigma_i = argmax_{\sigma_i} \sum_{k}^{K} \mathrm{LL}(p_i(X^k \mid X_i^{-k}, \sigma_i)).$$

I chose an optimal σ_i for every rank i. Figure 2 shows the resulting distributions, plotted by rank. To illustrate the effects of allowing σ_i to vary, Figure 2A shows the results when σ_i is constant across all ranks, whereas Figure 2B shows the results when σ_i varied based on the cross-validation accuracy.

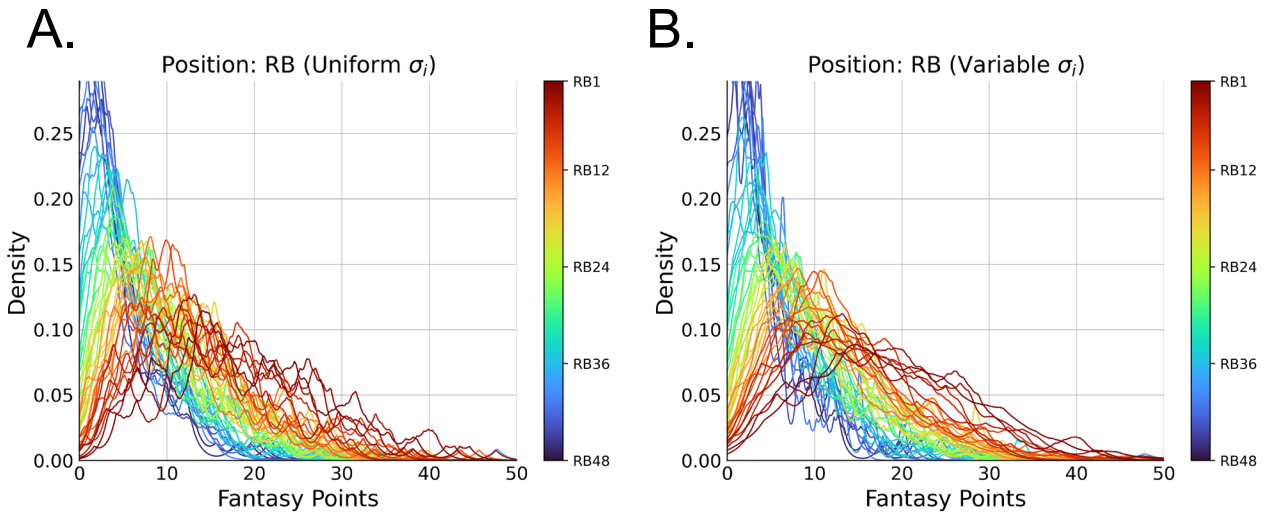

Figure 2. Applying kernel density estimation to the Figure 1 running back data. (A) The initial distribution where the kernel width is constant (σ_i = 1) for all ranks i. (B) Distribution when σ_i is permitted to vary. For better ranked players (e.g., RB1), σ_i approaches 2.0. For worse players, σ_i approaches 0.1.

1.2.2. Concatenation

The distributions are noisy. This is, in large part, due to the limited amount of data, as there are only 146 samples for each rank. To address this, I first concatenated data from neighboring ranks, establishing a much larger pool of data on which to fit kernel density,

$$X_i' = (X_{i-a}, X_{i-a+1}, \ldots X_{i+a-1}, X_{i+a}).$$

a represents how many neighboring ranks were included in the concatenation and resulting data (X_i') were submitted to the equations above in place of X_i. For example, if $a = 1$, then the distribution for rank 7 was fit using the scores of players ranked 6th, 7th, or 8th. The optimal value of $a = 5$ for the running back position was established via cross-validation. The resulting distribution is illustrated in Figure 3C.

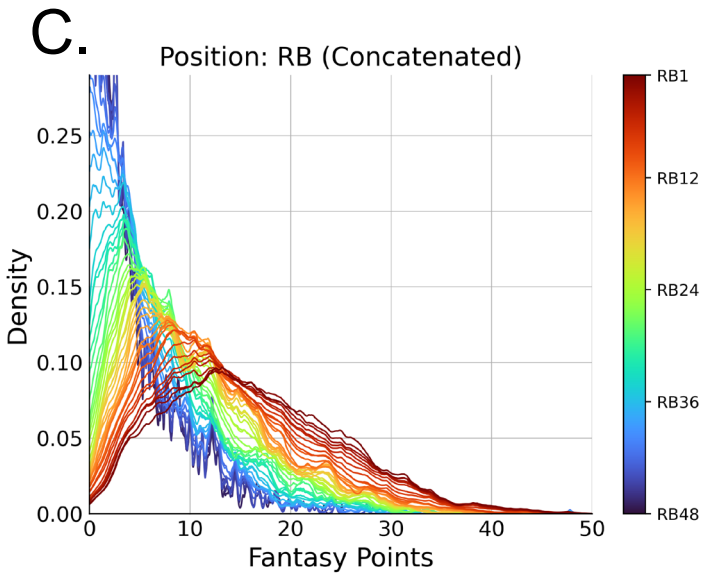

Figure 3. Running back distributions following concatenation. Concatenation used 5 closest other ranks, just above or just below.

1.2.3. Smoothing

Even after concatenation, the distributions remain fairly noisy. To rectify this, I smoothed the distribution horizontally. For example, by "horizontal smoothing", I mean that the likelihood assigned to an RB5 scoring 20 points was averaged with the likelihoods assigned to an RB5 scoring 19 points or 21 points. Smoothing used a gaussian kernel,

$$p_i'(x) = \int_0^{50} e^{(-\frac{y}{\sigma_y})^2} \, p_i(x+y) \, dy.$$

I established the optimal kernel width (σ_y) via cross-validation using a year's worth of data held out from all analyses thus far. The resulting distributions are shown in Figure 4D.

Likewise, I smoothed the data vertically. For example, the likelihood assigned to an RB5 scoring 20 points was averaged with the likelihoods assigned to an RB4 or RB6 scoring 20 points. Smoothing used a gaussian kernel with a width (σ_z) established via cross-validation (distributions shown in Figure 4E),

$$p''_i(x) = \int_1^{48} e^{(-\frac{z}{\sigma_z})^2} \, p'_{i+z}(x) \, dz.$$

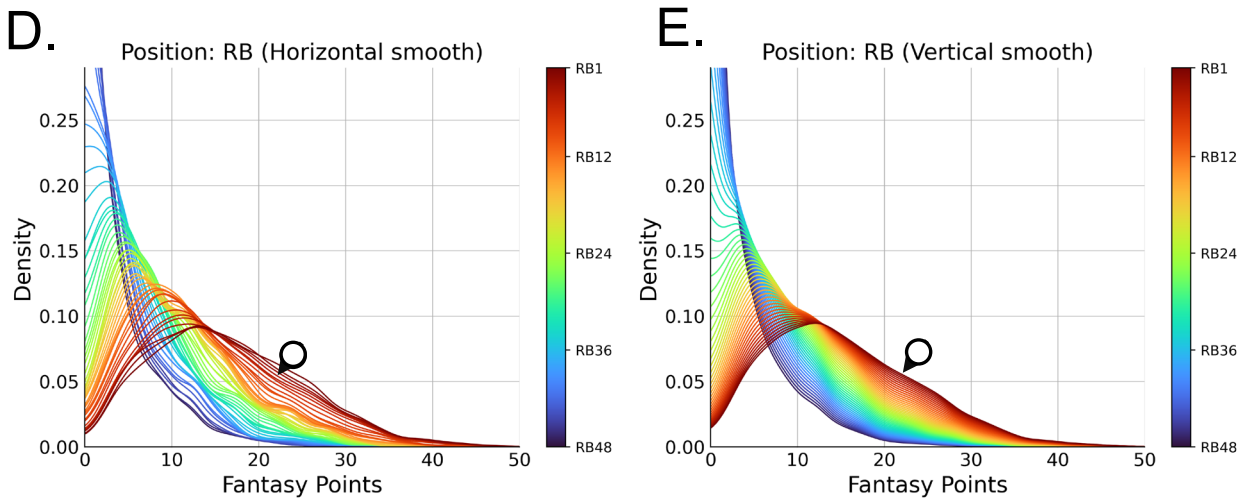

Figure 4. Smoothing the data horizontally (D) and vertically (E). A marker was added to each plot in the same location to highlight differences.

1.2.4. Correcting vertical smoothing border effects

Notice that vertical smoothing deflates the likelihood assigned to the highest-ranked players achieving many points. Figure 4 includes markers highlighting how the likelihood assigned to an RB1 scoring 22 points decreases from .07 in Figure 4D to .055 in Figure 4E after vertical smoothing. For mid-level ranks, vertically smoothing does not necessarily inflate or deflate projections because mid-level distributions are equally impacted by the distributions of higher and lower ranks. However, in smoothing the RB1 distribution, there are no higher-ranked distributions. Hence, border effects arise, whereby top-end ranks are predominantly averaged with lower ranks. This is a critical issue, as the top-end players are often of interest.

Hence, I extrapolated the performance of hypothetical players better than rank one – e.g., "RB(0)" and "RB(-1)". Extrapolation involved linear regression. Although linear regression deviates from the overall non-linear nature of the data, this is acceptable as the extrapolation simply serves to counterbalance the asymmetric averaging, and the extrapolated players will not be part of the final results.

Formally, to extrapolate ranks better than RB1 involves, I first converted the continuous $p'_i(x)$ distribution into a matrix, which represents points (x) discretely. For example, the probability a rank i player yields 20.1041 points is captured by a column representing 20.1 points,

$$P'_{ix} \approx p'_i(x).$$

Then, I fit a linear regression for every points (x) column, regressing P'_{ix} on rank i. The linear regression was fit using the top ten ranks and their data,

$$I = \begin{bmatrix} 1 \\ \vdots \\ 10 \end{bmatrix},$$

$$\beta_x = (I^T I)^{-1} I^T P'_{Ix}.$$

Using the regression weights (β_x), I extrapolated the likelihoods that the higher-ranked hypothetical players (ranks $i \leq 0$) score x points,

$$I^- = \begin{bmatrix} -24 \\ \vdots \\ 0 \end{bmatrix},$$

$$P'_{I^-x} = \beta_{x0} + I^- \beta_{x1}.$$

I appended the extrapolated distributions atop P'_{ix}, which Figure 5 illustrates. To address lower-end border effects related to lower-end players, like those ranked RB48, I did not need to extrapolate. I simply incorporated players that experts ranked as very low (e.g., players between RB49 and RB73).

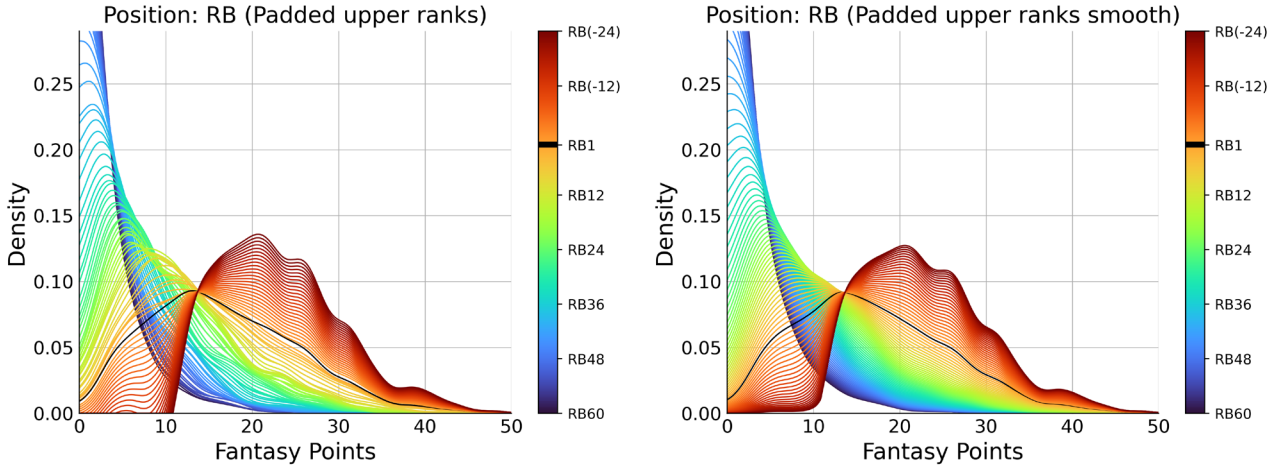

Figure 5. Illustration of players outside RB1 to RB48. To highlight where the extrapolation began, the RB1 distribution is delineated in black. The left plot shows the distributions without vertical smoothing, and the right plot shows the distributions post-smoothing.

After incorporating the extrapolated players into the distribution matrix, I vertically smoothed the distributions using a gaussian kernel smoothing,

$$p''_i(x) = \int_{-24}^{73} e^{\left(-\frac{z}{\sigma_z}\right)^2} p'_{i+z}(x)\, dz.$$

1.3. Results for each position

Figure 6 provides the final distributions for each position. Notice that the distributions are oftentimes not smooth, with slight humps and plateaus. These humps and plateaus are not noise, as increasing the horizontal smoothing to eliminate them lowers model fit on held-out data. These patterns arise due to touchdowns (worth 6 points) being a discrete factor influencing fantasy points. For example, see the wide receiver (WR) plot in the top right. There is first a plateau from 8-13 points and then a lower plateau from 17-21 points. Wide receiver performances at the lower plateau were likely linked to one more touchdown than those from the earlier one. Further below, Figure 7 provides the cumulative distributions for each position.

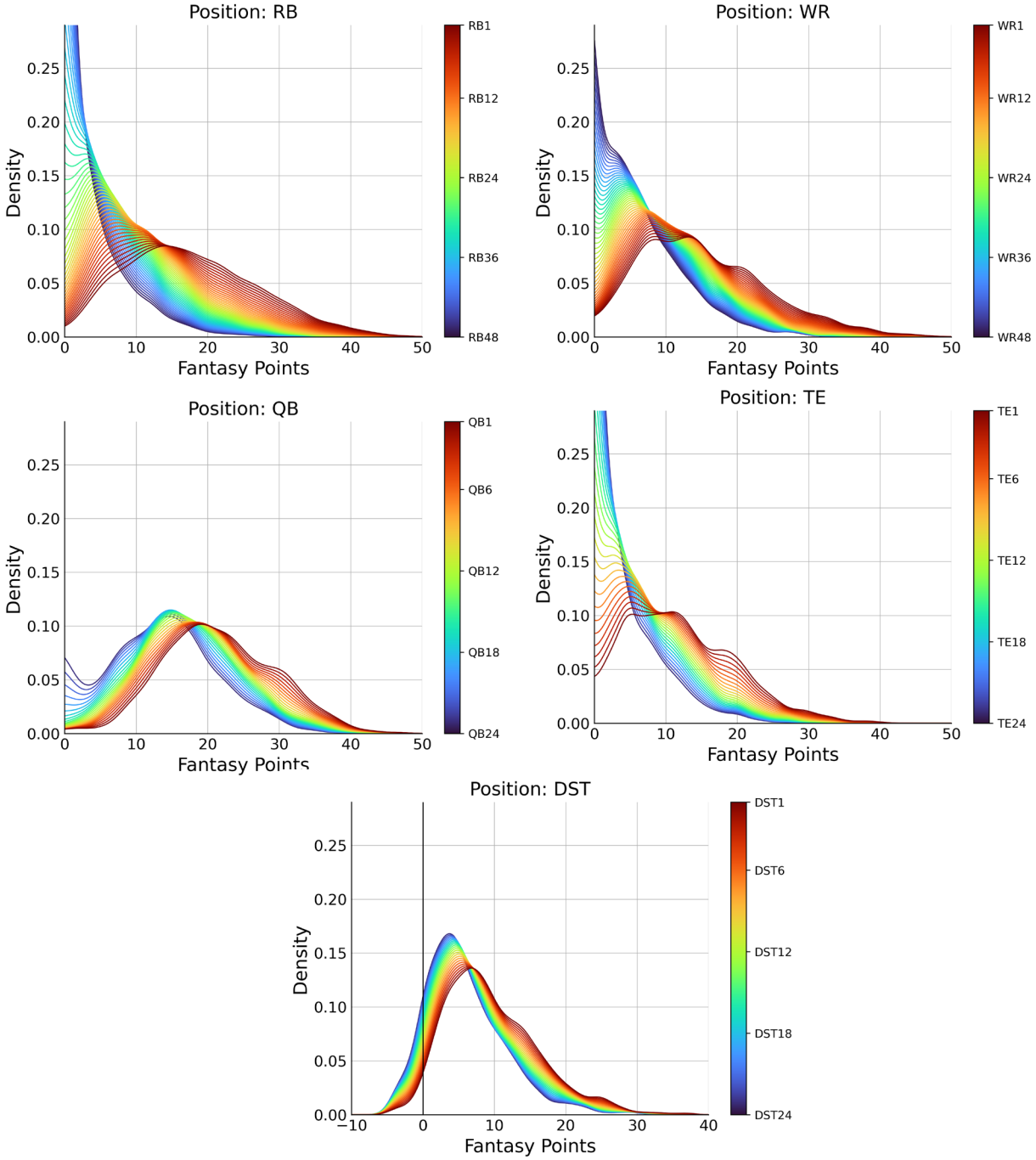

Figure 6. Player performance distributions by position. Running back (RB), wide receiver (WR), quarterback (QB), tight end (TE), defense/special teams (DST). DST is plotted to a minimum range of -10 because sub-zero scores are common for that position.

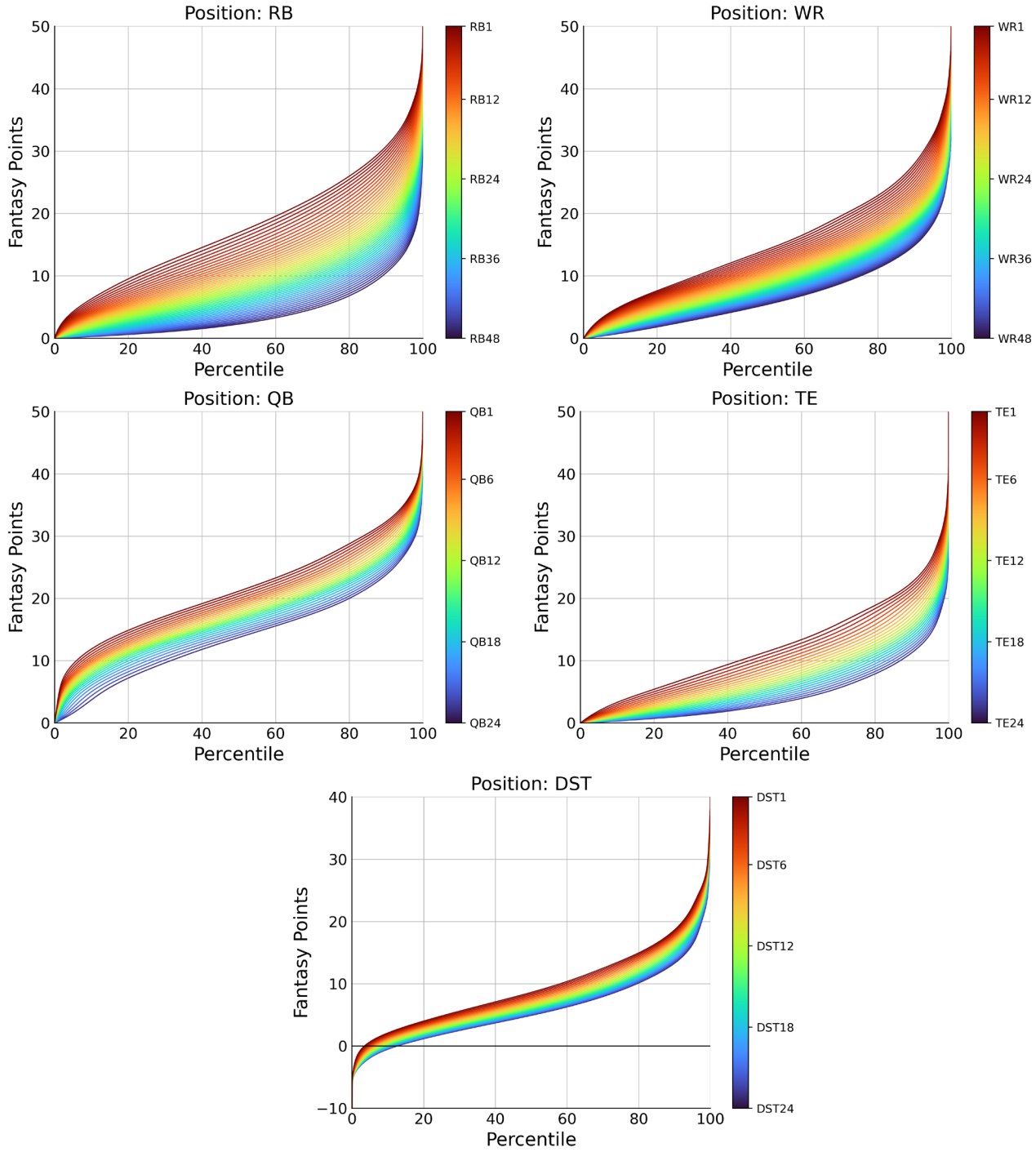

Figure 7. Player performance cumulative distributions by position. Running back (RB), wide receiver (WR), quarterback (QB), tight end (TE), defense/special teams (DST). DST is plotted to a minimum range of -10, as sub-zero scores are common for that position.

2. Daily Fantasy and Monte Carlo simulation

2.1. Background

Daily Fantasy Football is a form of NFL gambling. It is popular, yielding $7.6 billion in global revenue for sportsbooks. In a Daily Fantasy contest, users are given a *salary* (e.g., $25,000), and with that money, users create *lineups* by purchasing NFL players out of a pool using that salary. For example, a three-player lineup may require one quarterback, one running back, and one wide receiver, and a user may select players, Allen (costs $10,000), Barkley ($9,000), and Claypool ($6,000). Users aim to create a lineup made up of players who score many points, where players' scores depend on how well they perform in their real-life NFL games that week. For instance, if Allen performs well – throwing three touchdown passes and gaining 300 passing yards – he will earn 24.0 points that week. If Barkley then scores 14.1 points, and Claypool scores 6.2 points, the three-player lineup scores, in total, 44.3 points.

Users submit their lineups to contests, where every user has an identical salary and crafts a lineup from an identical pool of players. Contests range from small (2 users) to large (millions of users). Users whose lineups score the most points win the contest and receive earnings from the contest's prize pool. Hence, the goal of daily fantasy is to forecast what players will perform well and create a lineup made up of those players.

Note that users cannot simply select all the players whom the consensus agrees are the best. Users' salaries are limited, and contest organizers define contests such that better players usually consume more of users' salaries. For example, the organizers may project Allen to be a high-scoring quarterback and so price him at $10,000. On the other hand, they may project Dalton to be a mediocre quarterback and price him at $7,000. Even if a user believes Allen will likely score more points than Dalton, it may still be advantageous to include Dalton in the lineup if they believe Dalton will outperform the price assigned to him.

Ultimately, finding success in Daily Fantasy requires (1) projecting how well each player will perform, (2) considering each player's salary, and (3) combining that information to create a lineup that is most likely to be among the top-scoring lineups in a given contest.

2.2. Monte Carlo simulations

I use Monte Carlo simulations to identify optimal lineups. Tens of millions of lineups are created based on permutations of players from the available pool, constrained by players' salaries. Then, the lineups are scored across ten thousand iterations, wherein each player's points are simulated. This Monte Carlo design is an effective way to score a lineup (the sum of its players' scores) while accounting for the non-parametric nature of players' performances.

2.2.1. Team-level simulation

To simulate weekly performances, every player is assigned some number of points based on their expert ranking. This simulation is slightly more involved than simply sampling from the above distributions. We must consider that, for a given week, players' scores correlate with the scores of other players on the same team. For example, if a quarterback had success throwing the ball, this implies their wide receivers likely caught the ball well. Specifically, a 90^{th}-percentile performance from a quarterback, on average, corresponds to a 63^{rd}-percentile performance from their team's top wide receiver. I used historic data to assess this covariation and represented this covariation in percentiles as a multivariate distribution. To simulate players' scores, I used the conjunction between this multivariate team distribution and the distributions above.

2.2.2. Filtering candidate lineups

Based on the contest rules for the salary assigned to each player, many lineups are created. However, simply testing every single possible lineup is not feasible, as this quantity is far above computational limits (over 10^{15} possible lineups). Hence, I use a basic filtering procedure to trim the player pool to 10-20 players per position. For filtering, I calculate the mean of the points distribution for each player and regress the mean points on salary. Then, only players highest above the regression line (i.e., those expected to most overperform their cost) are included in the lineups. Based on my settings for how many players to include, this procedure generally yields between 10^7 and 10^8 lineups.

Note that even after this filtering, managing over 10^7 lineups for 10^4 simulations requires a variety of computational maneuvers to prevent memory overflow, particularly for parallel processing. My code is meant to run on a consumer-level computer (64 GB of RAM) without intermittent saves to a drive. My code is optimized. It simulates and scores 10^7 lineups for 10^4 simulations in just a few hours, although stable results can also generally be achieved in under 10 minutes if stricter preliminary filtering procedures are used (10^6 lineups) and fewer simulations are run (1,000-3,000).

2.3. Future directions

In truth, the aim of Daily Fantasy is not to create the lineup that will score the most points *per se* but to create the lineup that scores more points than every other contest entrant's lineup. To see this distinction, consider a player who is projected to overperform their salary but will be included in 90% of entrants' lineups. One may want to avoid such a player because, although they will likely do well, their success would be shared with everybody else. To account for phenomena like this, the Monte Carlo simulations should also project what percentage of entrants are likely to include a given player. Presently, my framework scrapes these projections from another online source and incorporates this data to evaluate a lineup's likelihood of scoring among the top. In the future, I would be interested in deriving my own projections for what percentage of entrants will include a given player. However, this would be difficult, as these projections may map less well onto expert rankings and thus arrives at the many-variable problem noted at the beginning.

A potentially more straightforward future direction is to no longer use the average across all experts as the sole ordinal predictor. My initial attempts to filter experts involved assessing their historical correlations between experts' predicted rankings and actual player performances. Filtering experts on that basis was not effective in improving model fit, although potentially better techniques can be discovered. Alternatively, I could pursue a strategy whereby data are averaged across experts later in the process. Distributions can be constructed for each expert, and then those distributions can be averaged.

Finally, it is worth considering how NFL trends shift year to year. For instance, some years may see an uptick in passing. Potentially, year-to-year trends deserve investigation. I performed preliminary analyses regressing out the effect of year (normalized to the last year in the training set). Although this ultimately did not improve model fit, future attempts may be worthwhile.

3. Conclusion

Using the models constructed here, I ran a profit on the Daily Fantasy site FanDuel in 2020 and 2021 across roughly a thousand contest entries. The model yields the most success in smaller contests of 2-100 entrants but limited success in contests with over 500 entrants. I expect this is because larger contests require more consideration of what players other entrants include in their lineups, and the source I am using for that data may be subpar. Nonetheless, I overall earned a 6% return on investment. Notably, it would be 3 times as high if not for the steep 12-20% site fee (vigorish) on winnings.