15TH ANNUAL MIT SLOAN SPORTS ANALYTICS CONFERENCE
APRIL 8-9, 2021 | VIRTUAL
presented by ESPN 42ANALYTICS

Sharpstack: Cholesky Correlations for Building Better Lineups

Andy Sherman-Ash
andy@numberfire.com

Keith Goldner
keith@numberfire.com

numberFire[1]

1. Introduction

It's Sunday at noon and you are trying to figure out the best lineup you can possibly play. You know that the Patriots are going to do well, but who should you start? Cam Newton is going to post huge numbers against a porous Miami secondary, but do you pair him with James White, Sony Michel, or neither? A quick look at the data: when Cam Newton has performed well, James White has too—he is one of Newton's favorite targets. Sony, on the other hand, seems to perform better when Newton does not since he rarely catches the ball. It is hard for them both to have a good fantasy performance when the Patriots cannot run and throw the ball on the same play. You lock in Newton and White, sit back, and wait for the double-digit New England win.

In Daily Fantasy Sports (DFS), users draft a lineup of real-life players for that day (or week), and accrue points based on each player's performance. Users compete against other individuals or in large tournaments where the goal is to finish with the maximum possible points. When building a lineup, each player has an associated salary and position. Users are required to fill specific position slots and keep the total team salary under a salary cap.

The most common way to optimize lineups for DFS is a combinatorial optimization problem known as the knapsack problem. Users feed projections and constraints for player salaries and positions into an algorithm which spits out the lineup with the highest projected points. These lineups work great for head-to-head matchups, but they break down when used in the most popular form of DFS contests—the large tournaments or guaranteed prize pools (GPP). In these larger tournaments, we want users to better understand player-to-player correlations. By picking correlated players, DFS users can increase their chances of a high-score lineup at the expense of a higher average score. Some optimizers solve for this by imposing additional "stacking" constraints (e.g. always draft a quarterback with a wide receiver on the same team, don't play a quarterback versus an opposing defense), but building out these rules can be extremely arbitrary and time consuming.

At numberFire, we have created a new solution for helping sports fans create tournament lineups using Cholesky decomposition. Cholesky decomposition generates correlated simulations and allows users to create lineups which naturally take into account properties of stacking, without adding manual constraints. Sharpstack is a tool available for all DFS users; it begins with customized correlations between every player in a game (e.g. how is James White's performance, a

[1] numberFire is a FanDuel Group company

pass-catching running back, correlated with Cam Newton versus Sony Michel, a traditional ground-and-pound running back)—taking into account previous team tendencies and general historical trends. Using these correlated simulations along with any user-selected preferences, Sharpstack allows our customers to better understand and build potentially high-scoring lineups for any given slate or set of games. We will show via back testing how lineups generated by Sharpstack would have performed in real-money GPPs on FanDuel.

1.1. Classical Linear Optimizer

The "classic" way to make a lineup formulates the problem as a knapsack problem and solves it using a linear optimizer with constraints. In their 2018 Sloan paper, "How to Play Strategically in Fantasy Sports (and Win)", Haugh and Singal [1] formulate the problem as follows:

> We assume there are a total of P players whose performance, $\boldsymbol{\delta} \in \mathbb{R}^P$, in a given round of games is random. We will assume that
>
> $$\boldsymbol{\delta} \sim N(\boldsymbol{\mu}_{\delta}, \boldsymbol{\Sigma}_{\delta}),$$
>
> so that $\boldsymbol{\delta}$ is multivariate normally distributed with mean vector $\boldsymbol{\mu}_{\delta}$ and variance-covariance matrix $\boldsymbol{\Sigma}_{\delta}$. Our goal in the fantasy sports competition is to choose a portfolio $\boldsymbol{w} \in \{0,1\}^P$ of athletes. Typically, there are many constraints on $\boldsymbol{w}$. For example, in a typical NFL fantasy sports contest, we will only be allowed to select $C = 9$ players out of a total of $P \approx 100$ to 300 NFL players. Each player also has a certain "cost" and our portfolio cannot exceed a given budget B. These constraints on $\boldsymbol{w}$ can then be formulated as
>
> $$\sum_{p=1}^{P} w_p = C$$
>
> $$\sum_{p=1}^{P} c_p w_p \leq B$$
>
> $$w_p \in \{0,1\}, p = 1, \dots, P,$$
>
> where c_p denotes the cost of the p^{th} player. Other constraints are also typically imposed by the context organizers. These constraints include positional constraints, e.g. exactly one quarterback can be chosen, diversity constraints, e.g. you cannot select more than 4 players from any single NFL team, etc. These constraints can generally be modeled as linear constraints and we use $\mathbf{W}$ to denote the set of binary vectors $\boldsymbol{w} \in \{0,1\}^P$ that satisfy these constraints.

An example of this would be our existing linear optimizer on numberFire.com [2]. While this method is effective as long as the projection inputs are accurate, it does not natively understand the correlations of the players or that stacking is effective because it does not model what we are after in GPPs. We want to maximize our odds of winning the tournament and these optimizers maximize our average final score, as we would want in a head-to-head contest.

1.2 Linear Optimizer + Stacking Constraints

Perhaps the most common way around this issue is to add rigid stacking constraints to the problem. This is how many optimizers in the industry currently work—the most notable being ones from FantasyCruncher**,** RotoGrinders, and DailyRoto [3]. These optimizers allow users to specify additional constraints to the linear problem, such as "always pair a quarterback with a wide receiver", or "don't play a running back against the defense he is facing." The problem with this approach is that sometimes users *should* stack a quarterback with a running back, just not always—it depends on how that specific running back's performance is correlated with the quarterback.

To get around these limitations, some of these programs allow the user to go even further, specifying exactly who to stack with whom—such as, "pair Aaron Rodgers with Aaron Jones, but not Jamaal Williams." The advantage of this method is it gives the user full control over exactly how they want their stacks set up. The disadvantage is that it becomes incredibly cumbersome and time consuming to set up each potential stack in this manner; and, the rigidity of the constraints can never be dynamic enough to capture all of the possible correlations. For example, receivers in the same game are typically correlated with each other, so it can make sense to play two receivers from opposing teams together. But, if one team is favored by a lot and facing a team that does not have a highly projected receiver, it might not make sense to stack receivers from both teams in that game. For a beginner user who does not know the best way to set up a lineup, this approach has a very steep learning curve. Some of these tools also allow the user to wiggle their input projections via "randomness"; but, randomness for the sake of randomness is not useful—it only degrades the quality of the projections. So, we set out to build a tool that utilizes "smart randomness".

1.3 Building Lineups by Hand

A final approach to building lineups which bears mentioning is the "old fashioned" way, done purely by hand. For an expert user who knows who they want to play and how to construct proper stacking strategies in their lineup, this is still one of the best ways to build lineups. The problem is that for large multi-entry tournaments, building 150 lineups by hand (even if each lineup only takes 2-4 minutes) can take 6-10 hours. For single-entry contests by experts, building lineups by hand is an excellent option, but for multi-entry contests or beginner to intermediate players, it is a suboptimal process. We will see that for players who prefer to build lineups by hand, Sharpstack can serve as a starting point, providing numerous viable lineups and stacks, which can then be tweaked by the expert user, rather than starting from scratch.

2. Sharpstack

We set out to build Sharpstack to combine the best of all of the aforementioned lineup building approaches. We wanted to create a lineup building tool which knew how to stack players, understood the correlations present in sporting events, but was also easy enough for a beginner to spend a few minutes and immediately have competitive lineups with a reasonable chance to win.

There is a four-step process to generating Sharpstack lineups:

1. Generate a custom correlation matrix for all player-to-player combinations on the slate

2. Simulate the slate of games 10,000 times using each player's projection for that slate, their projected standard deviation, and the custom correlation matrix from Step 1
3. Walk through each simulation with a standard linear optimizer, creating the optimal lineup observed from that simulation—this results in up to 10,000 "candidate" lineups to be considered
4. Race each "candidate" lineup across every simulation to see which lineups performed the best over the entire set of simulations

2.1 Data

The dataset used consists of game-by-game stat lines and projections for all fantasy-eligible positions going back to 2006. Using this data, we can build a baseline positional-rank correlation matrix (Figure 1). This matrix shows the high levels of correlation between a quarterback and his offensive teammates, as well as the high negative correlation between a defense and players on the opposing team:

Base Matrix	QB	RB1	RB2	WR1	WR2	WR3	TE	K	D	Opp QB	Opp RB1	Opp RB2	Opp WR1	Opp WR2	Opp WR3	Opp TE	Opp K	Opp D
QB	100%	29%	20%	50%	48%	45%	40%	15%	-2%	49%	22%	12%	31%	27%	15%	24%	0%	-32%
RB1	29%	100%	-11%	13%	19%	10%	5%	17%	9%	22%	24%	3%	11%	12%	-1%	18%	-15%	-23%
RB2	20%	-11%	100%	2%	5%	1%	17%	9%	6%	12%	3%	11%	4%	14%	-10%	10%	-10%	-8%
WR1	50%	13%	2%	100%	10%	5%	7%	7%	-3%	31%	11%	5%	20%	16%	9%	10%	0%	-11%
WR2	48%	19%	5%	10%	100%	7%	10%	8%	-8%	27%	12%	14%	16%	17%	10%	9%	0%	-12%
WR3	45%	10%	1%	5%	7%	100%	5%	9%	-5%	20%	9%	3%	9%	10%	2%	7%	0%	-13%
TE	40%	5%	17%	7%	10%	5%	100%	14%	-3%	24%	18%	10%	10%	9%	1%	20%	0%	-13%
K	15%	17%	9%	7%	8%	9%	14%	100%	-5%	0%	15%	-10%	0%	0%	0%	0%	-15%	-40%
D	-2%	9%	6%	-3%	-8%	0%	-3%	23%	100%	-32%	-23%	-8%	-11%	-12%	-13%	-13%	-40%	-18%
Opp QB	49%	22%	12%	31%	27%	15%	24%	0%	-32%	100%	29%	20%	50%	48%	45%	40%	15%	-2%
Opp RB1	22%	24%	3%	11%	12%	-1%	18%	-15%	-23%	29%	100%	-11%	13%	19%	10%	5%	17%	9%
Opp RB2	12%	3%	11%	4%	14%	-10%	10%	-10%	-8%	20%	-11%	100%	2%	5%	1%	17%	9%	6%
Opp WR1	31%	11%	5%	20%	16%	9%	10%	0%	-11%	50%	13%	2%	100%	10%	5%	7%	7%	-3%
Opp WR2	27%	12%	14%	16%	17%	10%	9%	0%	-12%	48%	19%	5%	10%	100%	7%	10%	8%	-8%
Opp WR3	20%	9%	3%	9%	10%	2%	7%	0%	-13%	45%	10%	1%	5%	7%	100%	5%	9%	-5%
Opp TE	24%	18%	10%	10%	9%	1%	20%	0%	-13%	40%	0%	17%	7%	10%	5%	100%	14%	-3%
Opp K	0%	15%	-10%	0%	0%	0%	0%	-15%	-40%	15%	17%	9%	7%	8%	9%	14%	100%	-5%
Opp D	-32%	-23%	-8%	-11%	-12%	-13%	-13%	-40%	-18%	-2%	9%	6%	-3%	-8%	0%	-3%	23%	100%

Figure 1: Positional-Rank Correlation matrix

For the purposes of drawing simulations, the ranking of each player is determined by their pregame projection (e.g. WR1 would be the wide receiver on a team projected for the most fantasy points). In red, we see there is a 48% correlation between a quarterback and their second-best wide receiver; in blue, there is a 40% correlation between a quarterback and their top tight end.

This matrix alone is enough to generate our correlated simulations; however, we know that not all teams and styles correlate in the same way. Due to play-calling tendencies, on some teams the QB1 and RB1 are highly correlated, but on other teams they are not. Further, on some teams the RB1 and defense are highly correlated because when the defense plays well and gets a lead, the RB1 will pound away at the clock until the game ends. An example of this would be Derrick Henry and the Titans defense—estimated at a +19% correlation as opposed to the baseline +9%. On other teams, the RB1 might be a pass-catching back who gets taken out in favor of a "closer"—such as Devin Singletary in 2019 (only +4% correlated with the Bills D, as Frank Gore was the closer). Unfortunately, we cannot just use a custom correlation matrix between every player in a specific game—the sample size is just too small as there are only 16 games per season in the NFL. In addition, with injuries, trades, and player movement, even long-time teammates can have a small sample of games played together. We handle this is by creating recency-weighted player-to-player rolling correlations—we pair up the game stat lines of each player and generate the correlations between them using an exponentially weighted moving correlation from the Pandas python library [4]. We take a Bayesian approach where, based on the size of the sample and our confidence in the recency-weighted correlations, we determine the weight of the custom correlations versus our league-average baseline correlations. This way, we use the base correlation as a prior for players who have little or no sample, while we use the specific correlations for those players with a long history playing together.

2.2 Projected distributions

In order to simulate effectively, we need to estimate the entire distribution of potential fantasy point outcomes. To derive this, we begin with their current projected fantasy points for the upcoming week. Projecting fantasy points is beyond the scope of this paper, but it is important to note that regardless of the optimization strategy, projected fantasy points will always be a key input. The better the projections are, the better the lineups will perform. Previous independent research has shown that numberFire projections are on par with the best in the industry [5].

Fantasy points roughly follow a Gamma distribution, according to work from Nathan Braun at fantasyMath [6]. Using empirical data from our historical game stat lines (seen in Figure 2), we estimate standard deviations for each position and projected fantasy points combination.

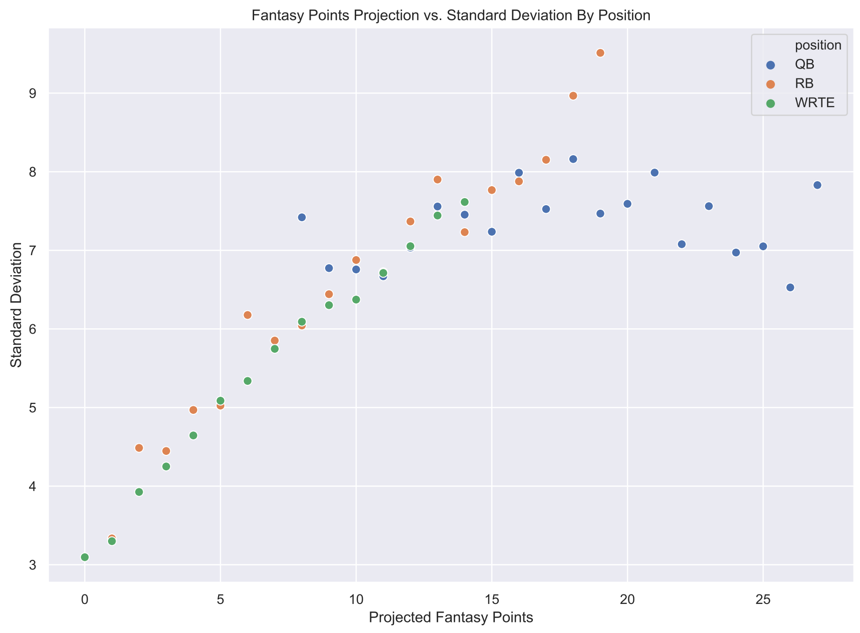


Figure 2

Finally, armed with our custom correlation matrix and projected fantasy point distributions, we are ready to generate our correlated simulations.

2.3 Cholesky Decomposition

Cholesky decomposition, also known as Cholesky factorization, is a method of decomposition for any Hermitian (or self-adjoint), positive-definite matrix. Discovered by André-Louis Cholesky for real matrices, it is extremely useful for Monte Carlo simulations with correlated variables [7].

Given any Hermitian, definite positive matrix $\boldsymbol{A}$, then $\boldsymbol{A}$ can be decomposed as:

$$\boldsymbol{A} = \boldsymbol{C_A}\boldsymbol{C_A^*}$$

where $\boldsymbol{C_A}$ is a lower triangular matrix with strictly positive diagonal entries, and $\boldsymbol{C^*}$ denotes the conjugate transpose of $\boldsymbol{C}$. Every positive-definite Hermitian matrix has a unique Cholesky decomposition [8]. If matrix $\boldsymbol{A}$ has only real entries, $\boldsymbol{C_A}$ is comprised of only real entries and $\boldsymbol{C_A^*}$ can be replaced by $\boldsymbol{C_A^T}$ (the transpose) in the decomposition [9]. A lower triangular matrix is a square matrix where all entries above the diagonal are zero:

$$\begin{bmatrix} X_{i,i} & 0 & 0 \\ \vdots & \ddots & 0 \\ X_{i,j} & \cdots & X_{j,j} \end{bmatrix}$$

In Monte Carlo simulations, we can decompose the covariance matrix, resulting in the lower triangular matrix $\boldsymbol{C_A}$. We can then apply this to a vector of uncorrelated samples $\boldsymbol{v}$, generating vector $\boldsymbol{\Omega}$, which models the covariance properties of our initial system.

For our NFL simulations, we start with the $\boldsymbol{j}$ x $\boldsymbol{j}$ correlation matrix $\boldsymbol{X}$. To generate the covariance matrix, which we will call $\boldsymbol{\beta}$, we use the following formula, where $\boldsymbol{S}$ is a vector of length $\boldsymbol{j}$ containing the standard deviations of each sampled entry:

$$\boldsymbol{\beta} = \boldsymbol{S} \cdot \boldsymbol{X} \cdot \boldsymbol{S}$$

Here, $\boldsymbol{j}$ represents the number of players we are sampling and $\boldsymbol{S}$ is a vector with each player's standard deviation. We then calculate our final correlated simulations, $\boldsymbol{\Omega}$, as the dot product of the decomposed covariance matrix and vector $\boldsymbol{v}$—an uncorrelated normally distributed vector of random variables of size $\boldsymbol{j}$:

$$\boldsymbol{\Omega} = \boldsymbol{C_A} \cdot \boldsymbol{v}$$

With our matrix of normally distributed correlated samples, **Ω**, we then translate our output to gamma distributions to more accurately model the properties of fantasy points (e.g. nonzero, right-tail skewed). An example distribution from Week 14 in 2019 can be seen in Figure 3:

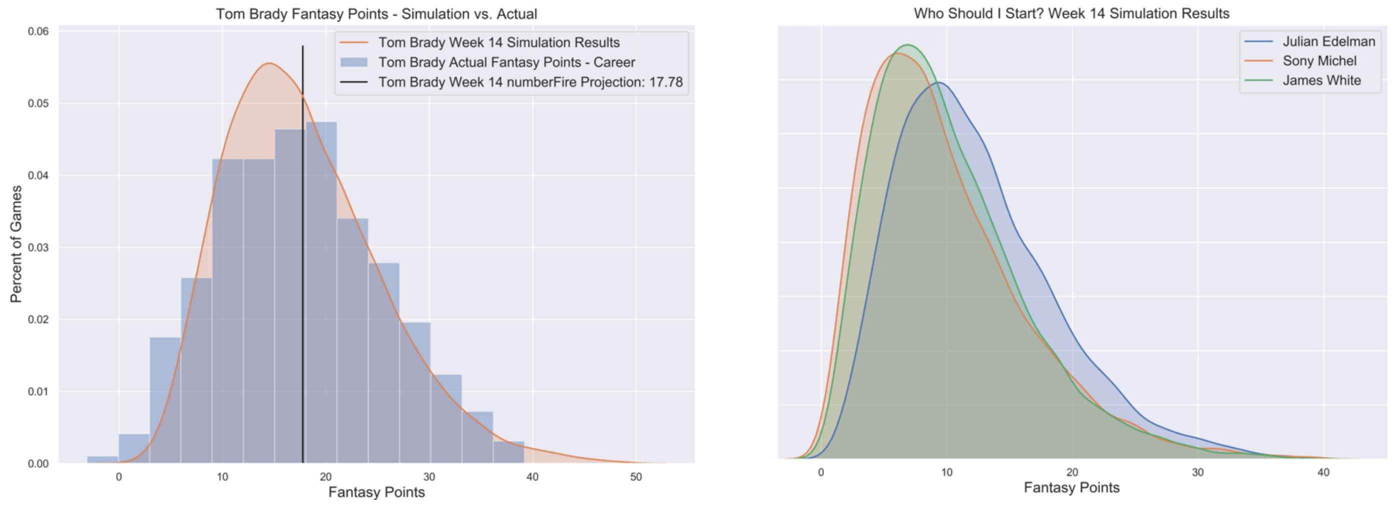


Figure 3: (Left) Tom Brady simulated fantasy points versus his actual results since 2001. (Right) Three different Patriots simulated fantasy point distribution.

The output distributions retain the properties of the inputs—the mean of each player's simulated distribution matches their numberFire projection and the correlation observed in the simulations of each player to each other player in their game matches their custom correlation matrix (Figure 4).

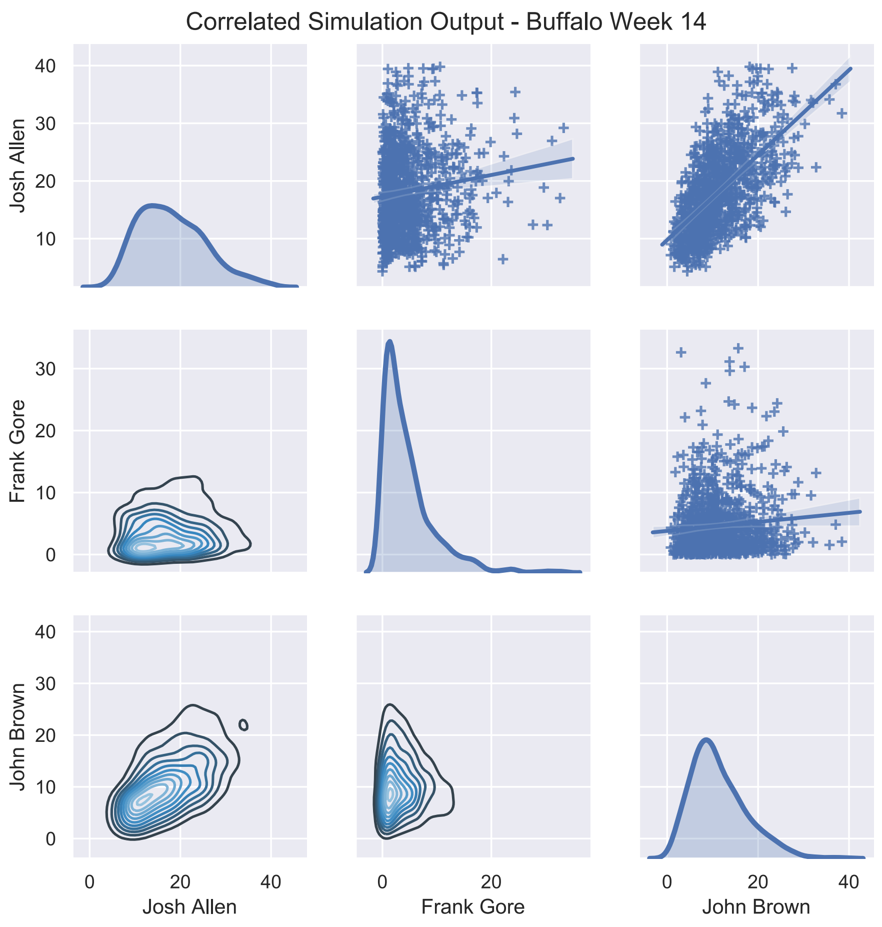
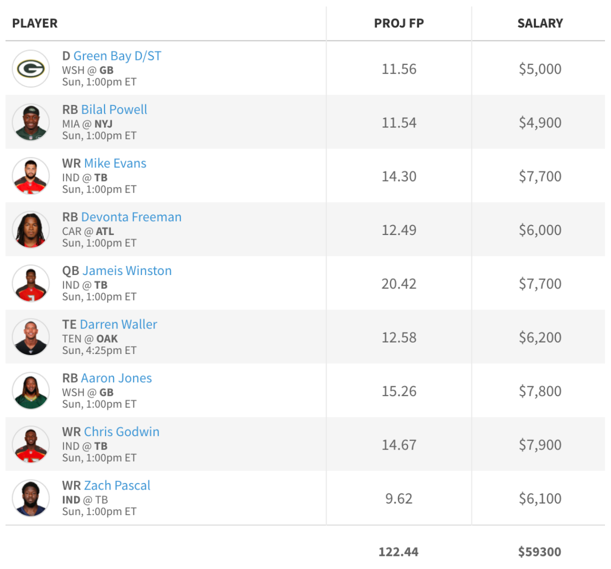


Figure 4: Josh Allen's Week 14, 2019 simulated output correlations with John Brown and Frank Gore.

We can see a very strong correlation between Josh Allen and John Brown's simulated performance compared to a weak correlation between Frank Gore and John Brown (a wide receiver and running back who we would not expect to both perform well together). We will discuss additional applications for these correlated simulations beyond daily fantasy in section 4.2.

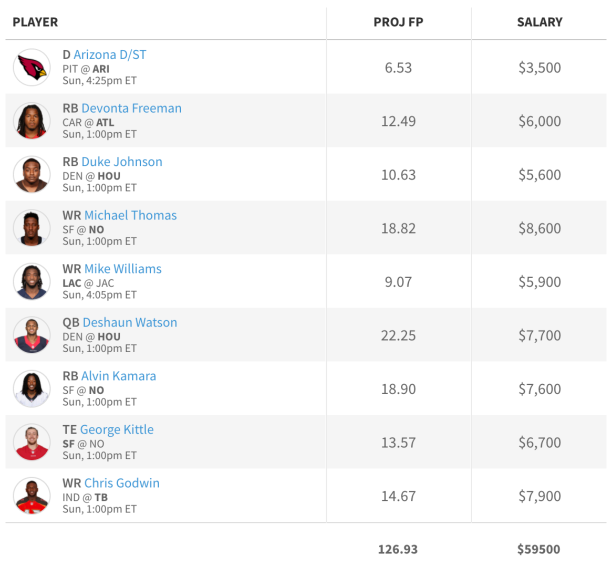
2.4 The Race

To turn these correlations into actionable DFS lineups, we first loop through each simulation, and create the optimal lineup for that simulation—the combination of players who scored the most points in each simulation, taking into account constraints (e.g. salary, position, MVP multipliers for single game contests). Once each of these "candidate" lineups has been selected, we race all of them in every simulation to determine the distribution of finishing positions for each lineup. This results in a final list of lineups outputted to the user (Figures 5 & 6):



PLAYER	PROJ FP	SALARY
D Green Bay D/ST WSH @ GB Sun, 1:00pm ET	11.56	$5,000
RB Bilal Powell MIA @ NYJ Sun, 1:00pm ET	11.54	$4,900
WR Mike Evans IND @ TB Sun, 1:00pm ET	14.30	$7,700
RB Devonta Freeman CAR @ ATL Sun, 1:00pm ET	12.49	$6,000
QB Jameis Winston IND @ TB Sun, 1:00pm ET	20.42	$7,700
TE Darren Waller TEN @ OAK Sun, 4:25pm ET	12.58	$6,200
RB Aaron Jones WSH @ GB Sun, 1:00pm ET	15.26	$7,800
WR Chris Godwin IND @ TB Sun, 1:00pm ET	14.67	$7,900
WR Zach Pascal IND @ TB Sun, 1:00pm ET	9.62	$6,100
	122.44	$59300

Figure 5: Sharpstack output with a Tampa Bay passing stack and Green Bay defense plus running back stack.



PLAYER	PROJ FP	SALARY
D Arizona D/ST PIT @ ARI Sun, 4:25pm ET	6.53	$3,500
RB Devonta Freeman CAR @ ATL Sun, 1:00pm ET	12.49	$6,000
RB Duke Johnson DEN @ HOU Sun, 1:00pm ET	10.63	$5,600
WR Michael Thomas SF @ NO Sun, 1:00pm ET	18.82	$8,600
WR Mike Williams LAC @ JAC Sun, 4:05pm ET	9.07	$5,900
QB Deshaun Watson DEN @ HOU Sun, 1:00pm ET	22.25	$7,700
RB Alvin Kamara SF @ NO Sun, 1:00pm ET	18.90	$7,600
TE George Kittle SF @ NO Sun, 1:00pm ET	13.57	$6,700
WR Chris Godwin IND @ TB Sun, 1:00pm ET	14.67	$7,900
	126.93	$59500

Figure 6: Deshaun Watson is stacked with his receiving running back, Duke Johnson, while New Orleans and San Francisco high-target players are stacked in their matchup.

In Figure 5, from the 2019 Week 14 NFL Main Slate on FanDuel, we see a QB-WR-WR stack of Jameis Winston, Chris Godwin, and Mike Evans, as well as Zach Pascal from that same game—all positively correlated with each other. Additionally, Aaron Jones is paired with his defense.

The second lineup (Figure 6) shows a stack between Deshaun Watson and his receiving back—Duke Johnson (Sharpstack chose Johnson, the receiving back, rather than Carlos Hyde who is more of a ground-and-pound player)—as well as Alvin Kamara, George Kittle, and Michael Thomas, who are all playing in the same game and positively correlated with each other. These correlated lineups are exactly the types of lineups top DFS players make when they create lineups by hand, while Sharpstack creates them dynamically with the press of a button.

3. Back testing

To show the efficacy of the solution and prove these lineups perform well in GPPs, we ran a series of back tests using historical FanDuel data. We compared the top 150 lineups, across 39 single game NFL slates from 2019 (5850 lineups each) from Sharpstack, the traditional knapsack method, as well as the top 150 lineups sorted by salary, to give some reasonable benchmarks.

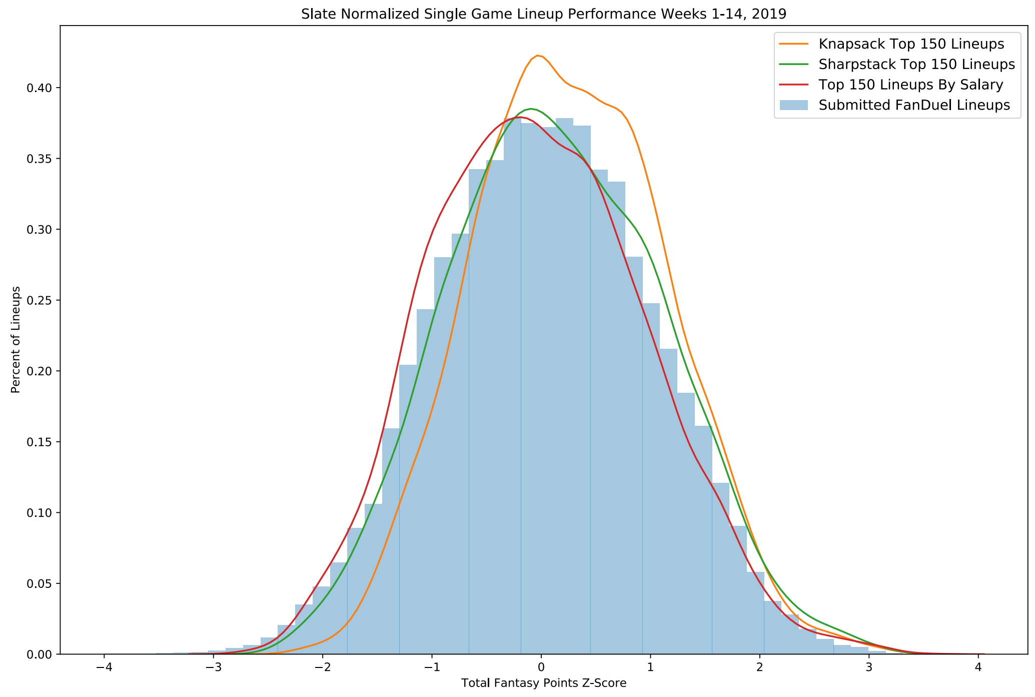


Figure 7: Normalized Lineup Scores, Single Game GPPs Weeks 1-14 2019

Figure 7 compares the z-score of total FanDuel points scored by lineups in different groups. We use the z-score so that we can accurately compare results across slates, since some slates are significantly higher or lower scoring than other slates depending on how many points are scored during the NFL game being contested. The curves show the distribution of z-scores for each of the four groups. Lineups sorted by salary—the simple benchmark which does not account for correlations or projections—performs about on par with the average lineup on FanDuel. The Knapsack lineups—which take into account projections but not correlations—perform the best on average with the highest mean z-score (0.276). This indicates the average knapsack lineup is beating 61.03% of scores in the contest. Despite having a lower mean z-score (0.134, beating 55% of scores), the Sharpstack lineups have a much higher z-score standard deviation (.976 compared to .878 for knapsack), which leads to a higher percentage of entries in the 95th and 98th percentiles—where we find all the big prizes in GPPs. Put another way, the top 5% of Sharpstack lineups are defeating 95.99% of scores in the contest, while the top 5% of Knapsack lineups are better than 95.90% of scores. Similarly—and more consequentially—the top 2% of Sharpstack lineups outperform 98.46% of real lineups, while the top 2% of Knapsack lineups outperform only 97.67%. We can see this in the plot in the bottom right where the green Sharpstack curve overtakes the Knapsack curve. These findings validate our assumption that using a Knapsack solver is the optimal strategy for maximizing mean score, but not for maximizing odds of a high, tournament-winning score.

In GPPs, however, we care about return on investments (ROI) rather than score. To test this, we simulated the three strategies modeled above. From each of the 39 slates tested, we chose a $1, 150-max entry GPP contest on FanDuel. The ROI each strategy would have achieved had they entered the top 150 lineups in each slate is shown in Figure 8.

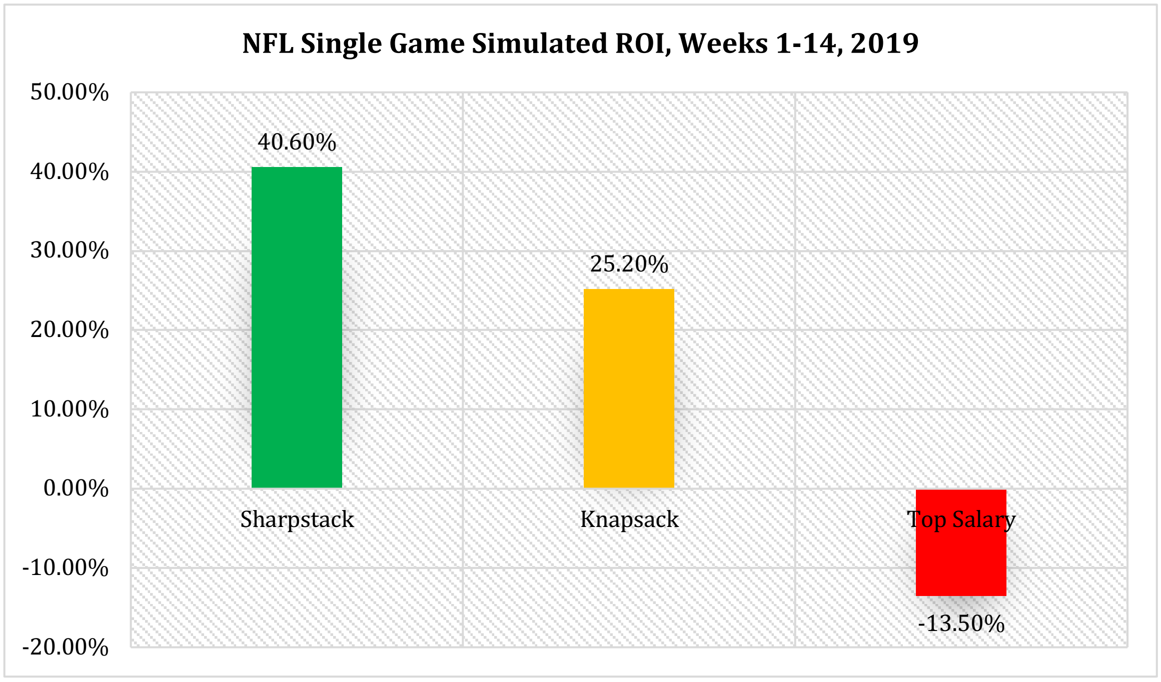


Figure 8: Simulated ROI in single game slates

Sharpstack finished with an ROI of 40.6%, Knapsack finished with an ROI of 25.2%, and Top Salary finished with an ROI of -13.5%. These results are consistent with what we would expect—the Top Salary strategy, our control, should be unable to overcome the site fees. We can see from the z-score distribution that it mirrors the average lineup, which would return an ROI exactly inverse to the site fees. We also expect the Knapsack to be a positive ROI strategy, because at the lowest stakes on FanDuel, using quality, timely projections and Knapsack is still likely to outperform the average entry. Sharpstack performed best despite its lineups finishing lower on average than the Knapsack strategy. This is thanks to its aforementioned higher variance (higher ceiling, lower floor) of outcomes based on smart positive correlations.

4. Future Work

4.1 Potential Improvements

There are a series of future improvements which could be made to increase the effectiveness and power of Sharpstack. First, because the correlated simulations are essentially a shortcut to a full game simulation, if one has a full sport simulator down to the play-by-play level, such as NFLGameSim.com [10], Sabersim.com [11] or a Markov model [12], the game simulation results can replace the correlated simulations, with the same end-to-end functionality (e.g. looping through each simulation, finding the optimal lineup, and racing them against each other).

Another useful iteration would be to have player-specific standard deviations. Currently, we have player-specific mean projections and positional-specific standard deviations. For example, Jarvis Landry and John Brown might be projected for the same amount of fantasy points, in which case we would assign each of them the same standard deviation, but as football fans we know that John Brown mostly runs deep routes, while Jarvis Landry mostly plays underneath[2]. Because Brown catches the ball deeper on average, with fewer receptions per game, John Brown's true results should have a higher standard deviation than Landry's, who will be much more predictable. Keeping small sample size in mind, a likely improvement here would be to use a Hierarchical Bayesian model, taking into account historical positional standard deviations as a prior and adjusting those based on the observed player's game logs and sample size. We did some research into whether this would improve the predictability of a wide receiver's range of outcomes, but found that for the vast majority of receivers, it did not. There are, however, a small minority of receivers who specialize in the deep pass, such as John Brown, that do consistently have standard deviations higher than league average, improving the predictability of their distribution.

The Sharpstack process is computationally intensive, so there are large potential gains in terms of speed, memory, and processing power through parallelization. In an ideal world, users would be able to modify the projection of any player and run the process end-to-end on-demand. To save the user time, we cache the top set of lineups from a constantly running thread and serve a subset to the user. This has the advantage of displaying up to 150 lineups instantly to the user—something no other optimizer currently does—while also allowing the user to lock and exclude any number of players before re-running the program.

We are also in the process of developing new methods for optimizing the correlated simulations, aside from the traditional knapsack linear programming solution. One approach we have developed uses a genetic algorithm to learn the optimal multi-lineup strategy, maximizing cross-lineup performance.

4.2 Additional Use Cases

The usefulness of correlated simulations is not limited to NFL DFS. We have extended Sharpstack to work with MLB contests—using a correlation matrix based on batting order for hitters. We are in the process of developing an NHL version of Sharpstack, given that correlations in hockey are crucially important when building lineups. Since stacking is not as important in NBA DFS (the correlations between players and teammates are fairly weak), there is not much of a demand to extend the methodology to basketball. This framework can also be easily modified to support other DFS sites and contests, such as DraftKings or Yahoo, by adjusting the projections and constraints to fit those sites.

Beyond DFS, the simulations can be used in season-long fantasy formats for sit-start decisions. The goal of season-long fantasy in head-to-head leagues (the most common format) is not to score the

[2] In 2019 Brown's average numberFire projection was 7.969 and Landry's was 7.929. John Brown's Air Yards, however, were 1,667, while Landry's were 1,338 [13].

most points, but rather create the lineup which maximizes your win probability, taking into account both your players and your opponent's players. We have developed a tool to optimize a season-long lineup based on these correlations and the current weekly matchup to maximize win probability. This can result in some counterintuitive decisions where the optimal choice is *not* to start the highest projected players. A favored team will want to reduce its inter-lineup correlation, whereas an underdog team would like to increase it.

These simulations are also being used to help generate probabilities to exceedingly difficult questions due to the nature of same-game correlations. We can answer questions like: What are the odds Cam Newton scores 20 fantasy points? What are the odds Cam Newton scores 20 or more fantasy points AND Julian Edelman scores 10 or more? What are the odds that Julian Edelman is the highest scoring player in the Patriots game?

5. Discussion

Using Cholesky decomposition, we built a framework for generating correlated simulations between all players in an NFL game. These simulations can be used to create competitive daily fantasy lineups which organically exhibit stacking properties, previously only accomplished by time-consuming manual processes and adding rigid constraints. Our back tests show the effectiveness of these generated lineups compared with other submitted FanDuel lineups, as well as the existing industry standard linear optimizer, in large tournaments or GPPs. Sharpstack is currently live on numberFire.com for any user to better understand and build potentially high-scoring lineups for any given slate or set of games.

References

[1] Haugh, Martin B., and Raghav Singal. "How to Play Strategically in Fantasy Sports (and Win)." *MIT Sloan Sports Analytics Conference*, 2018, www.sloansportsconference.com/wp-content/uploads/2018/02/1001.pdf.

[2] Zachariason, JJ. "Introducing Our Brand New Daily Fantasy Football Tools." *numberFire*, 6 Sept. 2016, www.numberfire.com/nfl/news/10348/introducing-our-brand-new-daily-fantasy-football-tools.

[3] "Tools – Lineup Optimizers & Projections." *Daily Fantasy Sports 101*, 2019, www.dailyfantasysports101.com/tools/.

[4] McKinney, W., & others. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference* (Vol. 445, pp. 51–56), https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.core.window.EWM.corr.html

[5] Reda, Greg. "Which Site Has the Best Fantasy Football Projections?" Datascope Analytics, 18 Dec. 2014, datascopeanalytics.com/blog/which-site-has-the-best-fantasy-football-projections/.

[6] Braun, Nathan. "Fantasy Math -- Fantasy Football Start Sit Advice." *Fantasy Math -- Fantasy Football Start Sit Advice*, 2017, fantasymath.com/.

[7] O'Connor, J J, and E F Robertson. "Andre-Louis Cholesky." *School of Mathematics and Statistics, University of St Andrews, Scotland* , Aug. 2005, mathshistory.st-andrews.ac.uk/Biographies/Cholesky.html.

[8] Golub, Gene H.; Van Loan, Charles F. (1996). Matrix Computations (3rd ed.). Baltimore: Johns Hopkins. ISBN 978-0-8018-5414-9.

[9] Horn, Roger A.; Johnson, Charles R. (1985). Matrix Analysis. Cambridge University Press. ISBN 0-521-38632-2.

[10] "About Game Sim." *NFL Game Simulator*, 2019, www.nflgamesim.com/About.asp.

[11] "Learn More." *Sabersim*, 2019, www.sabersim.com/dfs/tools.

[12] Goldner, Keith. "A Markov Model of Football: Using Stochastic Processes to Model a Football Drive." *Journal of Quantitative Analysis in Sports*, vol. 8, no. 1, 2012, doi:10.1515/1559-0410.1400.

[13] Hermsmeyer, Josh. "Air Yards Sortable Data." *Air Yards*, 2018, airyards.com/tables.html.

Appendix A

This table shows the highest estimated correlated players for Week 14, 2019. The majority of the top pairs are quarterbacks and receivers.

Highest Estimated Correlated Players Week 14, 2019

Rank	Player 1	Player 2	Correlation
1	Russell Wilson	Tyler Lockett	64.33%
2	Josh Allen	John Brown	63.27%
3	Carson Wentz	Alshon Jeffery	62.41%
4	Ryan Fitzpatrick	Michael Gesicki	61.98%
5	Kyler Murray	Larry Fitzgerald	58.52%
6	Ryan Fitzpatrick	DeVante Parker	58.01%
7	Ryan Tannehill	A.J. Brown	55.05%
8	Mitchell Trubisky	Allen Robinson	55.01%
9	Lamar Jackson	Seth Roberts	53.23%
10	Mitchell Trubisky	Anthony Miller	52.63%
11	Aaron Rodgers	Davante Adams	52.51%
12	Eli Manning	Sterling Shepard	52.44%
13	Jared Goff	Cooper Kupp	52.40%
14	Russell Wilson	D.K. Metcalf	51.49%
15	Gardner Minshew	D.J. Chark	51.36%
16	Gardner Minshew	Josh Lambo	51.28%
17	Lamar Jackson	Justice Hill	51.14%
18	Lamar Jackson	Marquise Brown	50.94%
19	Baker Mayfield	Odell Beckham	50.66%
20	Carson Wentz	Jake Elliott	50.59%
21	Drew Brees	Michael Thomas	49.60%
22	Gardner Minshew	Chris Conley	48.90%
23	Jimmy Garoppolo	George Kittle	48.84%
24	Kirk Cousins	Kyle Rudolph	48.26%
25	Kyle Allen	D.J. Moore	47.24%