Modeling offensive player movement in professional basketball

Steven Wu *[1] **and Luke Bornn**[1,2]

[1]**Department of Statistics and Actuarial Science, Simon Fraser University**
[2]**Sacramento Kings**

ABSTRACT

The 2013 arrival of SportVU player tracking data in all NBA arenas introduced an overwhelming amount of on-court information - information which the league is still learning how to maximize for insights into player performance and basketball strategy. Knowing where the ball and every player on the court are at all times throughout the course of the game produces almost endless possibilities, and it can be difficult figuring out where to begin. This article serves as a step-by-step guide for how to turn a data feed of one million rows of SportVU data from one NBA game into visualizable components you can use to model any player's movement. We detail some utility functions that are helpful for manipulating SportVU data before applying it to the task of visualizing player offensive movement. We conclude with visualizations of the resulting output for one NBA game, as well as what the results look like aggregated across an entire season for three NBA stars with very different offensive tendencies.

INTRODUCTION

The introduction of STATS SportVU, a six-camera system installed in every NBA arena, introduced a new era of sophisticated analytics (Franks et al., 2015b; D'Amour et al., 2015; Franks et al., 2015a). Previous to this technology, the most granular type of data was play-by-play: a text log of the major events (shot attempts, rebounds, etc.) that occur throughout the game. In contrast, SportVU's spatio-temporal data is 25 frames per second of data on the (x, y) coordinates of each of the ten players on the court, plus (x, y, z) coordinates of the ball. The result is approximately one million rows per game. For context on how much data there is to process, each of the 30 NBA teams plays 82 games per season. The resulting data can quickly become overwhelming to work with, both conceptually and computationally.

Knowing precisely where each player is throughout a game spurs interesting questions about movement. Can we understand how players move with and without the ball? Are we able to simulate player movement? Such questions also have downstream implications; for example, recent models for estimating instantaneous possession value rely on an underlying player movement model (Cervone et al., 2016).

Imagine a simulator that approximates movement for each individual player, conditioned on all of the factors involving the player - front-office decision makers could understand how different lineup permutations could potentially co-exist on the court at a finer level before committing to irreversible decisions. This article presents a comprehensive walkthrough of how we turn this raw data into results that can be used for a first-approximation simulator for NBA player movement on offense, and how this can help gain new insights on player offensive movement tendencies. We're going to focus on offensive movement, as defensive movement is largely a function of the offensive player the defender is guarding.

AVAILABILITY OF THE DATA

The data used in this article is the SportVU data for one sample game, available as a .csv file at `https://github.com/dsscollection/basketball`. There is another GitHub repository, `https://github.com/neilmj`, where efforts are made to upload raw SportVU data on a regular basis (Johnson, 2015). Several blog posts exist online that provide examples of varying detail into programatically accessing the `http://stats.nba.com` API directly (Reda, 2015) (Forsyth, 2015).

*Contact: Steven Wu at hello@stevenwu.info

LOOKING AT THE DATA

To start, we work with the spatio-temporal data coming from a game on November 1st, 2013 featuring the Miami Heat visiting the Brooklyn Nets. Note that this dataset has only 89868 rows because the spatial coordinates for the ball and ten players on the court in any given instant are given in one row, not one row per entity.

Each row in the dataset includes the spatial coordinates (in feet) for all entities on the court for one moment in time, along with contextual variables. For the uninitiated, the basketball court in an NBA arena is 94 feet by 50 feet. $(0,0)$ is the corner of the court and $(47,25)$ is the coordinate for center court.

Tables 1 and 2 offer a peek at a sample of the dataset. Table 3 details the important columns in the sample game.:

Table 1. Sample of dataset's columns listed in Table 3

	time	game	quarter	shot_clock	game_clock	x	y	z
214	8519	2013110117	1		716.44	15.25	28.35	6.89
215	8559	2013110117	1		716.40	14.47	28.21	6.81
216	8599	2013110117	1		716.36	13.98	28.05	6.63

Table 2. Sample of rest of dataset's columns listed in Table 3

	a1_ent	a1_x	a1_y	a1_event	possID
214	296572	11.35	26.88		1
215	296572	11.62	26.88	23	1
216	296572	11.90	26.87		1

Table 3. Description of data columns in 2013_11_01_MIA_BKN.csv

Column Name	**Data Type (Range)**	**Description**
time	integer	how many milliseconds have passed since the start of the game
game	integer	a unique identifier for the game that this dataset refers to
quarter	integer	the quarter of the game (games have at least 4, with possible overtimes)
shot_clock	numeric	number of seconds left in the offensive possession before a turnover occurs
game_clock	numeric	number of seconds left in the quarter
x	numeric	x coordinate on the court axis for the ball, in feet
y	numeric	y coordinate on the court axis for the ball, in feet
z	numeric	z coordinate (perpendicular to court surface) for the ball, in feet
(a|h)i_ent	integer	entity id for the i^{th} player on the court for the (a)way or (h)ome team
(a|h)i_x	numeric	x for the i^{th} player on the court for the (a)way or (h)ome team
(a|h)i_y	numeric	y for the i^{th} player on the court for the (a)way or (h)ome team
(a|h)i_event	integer	event id for the i^{th} player on the court for the (a)way or (h)ome team
possID	integer	running count of the number of changes in team possession

UTILITY FUNCTIONS

Before getting started with the movement modeling, we define some utility functions (functions which provide general functionality that are useful and reusable for other applications with the same data).

Labelling Who Has Possession

To begin querying for offensive moments, we create a function *add_possession_data_for_quarter()* that augments our data by adding two additional columns: one indicating which team has possession, and one indicating which entity has possession (-1 indicates neither team having possession, such as during the

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3201v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017

time between the game beginning at the tip-off until the moment when a player controls the ball). The hard work is done for us with the **event_id** column - it contains information on exactly which moment the possession changes between player to player (the **event_id** for possession is 23). Defensive and offensive rebound moments, which have their own **event_id**, do not explicitly follow with the possession **event_id**, so we check for those events too for tracking changes in possession.

add_possession_data() calls this function for each quarter and stitches together the result before adding column names to the two new columns of data. Table 4 shows an example of the same rows shown in Tables 1 and 2, but with the new possession data added.

Table 4. Output of add_possession_data(): Same rows as Table 1 and 2. Two new columns team_w_poss and ent_w_poss for easy querying of when a particular player or team possesses the ball

	a1_ent	a1_x	a1_y	a1_event	possID	team_w_poss	ent_w_poss
214	296572	11.35	26.88		1	a	214152
215	296572	11.62	26.88	23	1	a	296572
216	296572	11.90	26.87		1	a	296572

Filtering Out Noisy Offensive Moments

Now that we have appended data that explicitly tells us what team and what entity has possession at every moment, we could gather all offensive moments by filtering for all rows where the player's team possessed the ball by using the newly created **team_w_poss** column. However, this would be a bit too naive, as it would include two cases of moments which we would want filtered out:

1. from when the ball is carried from the team's own side of the court until when the ball passes over the midcourt (where players move mostly in a linear fashion to the offensive half of the court)

2. from when a shot attempt has gone up until when the ball either lands in the hoop, in a player's hand, or out of bounds (where players either move mostly in a linear fashion toward the hoop to crash the boards or toward their own hoop to play transition defense)

Including these moments would add a substantial amount of undesirable noise. Movement that occurs in these two cases don't reflect the on-ball and off-ball player movement in an offensive possession that we want to capture and model. What is most interesting about players' motion on offense is how they behave from when the possession starts in the half-court up until the possession ends by either a shot attempt or turnover. The details of how we remove these irrelevant moments from consideration follows below.

Offensive Moments Before Half Court

Let's review how an NBA game is structured: teams start out shooting on opposing ends for the first half, and then switch sides after half-time. To filter for only offensive moments where all offensive players have crossed the mid-court, we need a function to determine which direction each team is attacking for each half. One way to do this is to find all moments where a shot occurred for each team, find the average x coordinate value, and check which team has their average on the left hand side of the court and which team has their average on the right hand side. This is what we do in *get_directions_of_play()*, where the output is shown in Table 5.

Table 5. Output of get_directions_of_play(): one row per team and one column per half

	1st	2nd
a	right	left
h	left	right

Offensive Moments After Shot Attempts

Immediately after a shot is attempted, a player's tendency is to usually crash the paint for an offensive rebound attempt or to run backwards to set up for defense. Using the **event_id** column, we can identify all row indices where a field goal make/miss is recorded, and all row indices where a ball is possessed or

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3201v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017

rebounded. For every field goal make/miss, we find the closest event that concludes the shot attempt, and ignore the moments in between.

By encoding this logic into *get_offensive_moments()* and applying the function for this game, the number of total moments to process is reduced from 89868 to 24218. Even though approximately half of the game is split between playing offense and defense from a team's perspective, the percentage of rows kept is lower than expected because we are only keeping moments where the game clock is running.

Filtering Out Offensive Possessions By Player

We need a function that will further filter a team's offensive moments by removing any moments that don't have a player of interest on the court. We do this in *get_player_offensive_moments()* by checking the relevant team columns (either away or home) for whether the player entity identifier is in any of the five entities on the court. Our implementation also removes all columns that won't be necessary for the follow up analysis, but this part isn't strictly necessary.

LeBron James, undisputedly one of the top players in the NBA, is a player in this game with entity identifier 214152, playing for the away team Miami Heat. Table 6 shows the output of applying this player filter function. The number of moments reduced only from 24218 to 21257 as LeBron James was on the court for the majority of play, playing 42:14 out of 48:00 possible minutes for that game.

Table 6. Output of get_player_offensive_moments(): the number of columns is filtered down and we only store the player's (x,y) coordinates for each row

	time	quarter	game_clock	x	y	team_w_poss	ent_w_poss
138	11640	1	713.32	73.78	17.32	a	296572
139	11680	1	713.28	74.40	17.24	a	296572
140	11720	1	713.24	75.00	17.15	a	296572
141	11760	1	713.20	75.59	17.06	a	296572
142	11800	1	713.16	76.17	16.98	a	296572
143	11840	1	713.12	76.74	16.91	a	296572

Transposing Offensive Possessions

The last utility function we need is one that transposes all offensive moments on one side of the court to the other, which we implement in *flip_coords()*. Spatially, a cut from the right-side of the 3-point arc to the baseline in the 1st half has different (x,y) coordinates than the same cut in the 2nd half. But in terms of movement on offense, they're identical movements. To have a consistent frame of reference, we transpose all moments on the right-hand side to the left-hand side by flipping both the x and y coordinate: see Table 7 for the output.

Table 7. Output of flip_coords(): notice how LeBron James' first half coordinates are transposed as the Miami Heat were attacking the right hand side of the court to start the game

	time	quarter	game_clock	x	y	team_w_poss	ent_w_poss
138	11640	1	713.32	20.22	32.68	a	296572
139	11680	1	713.28	19.60	32.76	a	296572
140	11720	1	713.24	19.00	32.85	a	296572
141	11760	1	713.20	18.41	32.94	a	296572
142	11800	1	713.16	17.83	33.02	a	296572
143	11840	1	713.12	17.26	33.09	a	296572

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3201v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017

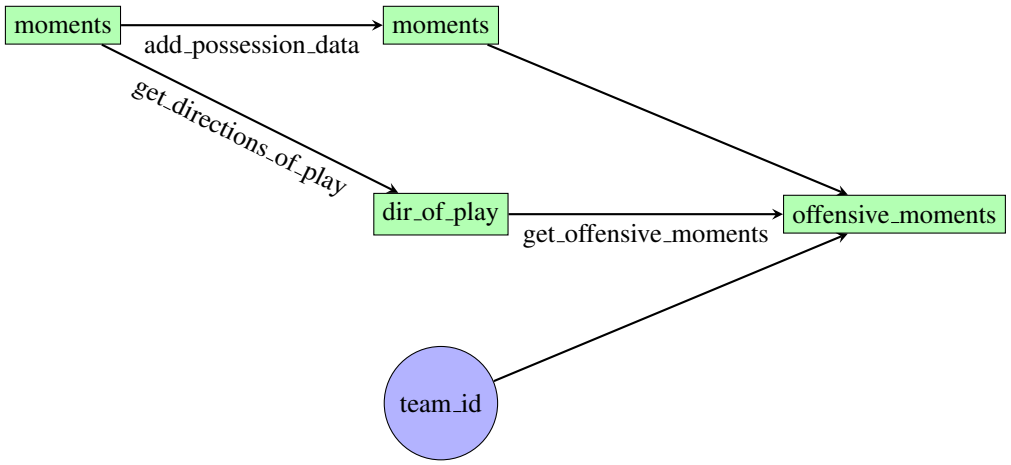

Figure 1. Flowchart of operations to transform our raw moments data and extract the relevant offensive moments for a team of interest

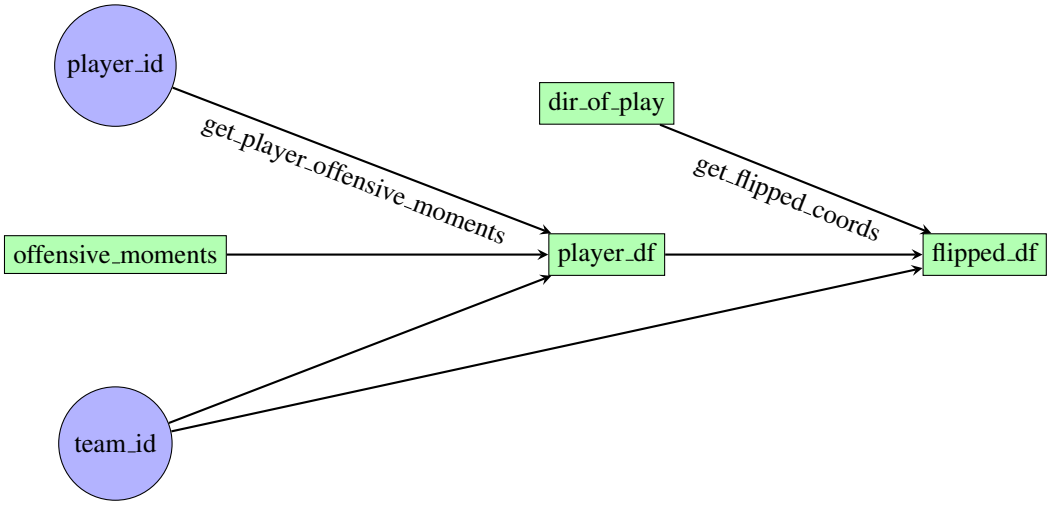

Figure 2. Flowchart of operations to obtain the relevant offensive moments for a player

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3201v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017

MOVEMENT SIMULATION FUNCTIONS

We use a helpful package called **raster** (Hijmans, 2016) which provides functionality for transforming our court space into a grid of equally sized cells by converting it into a RasterLayer object. The package provides useful functions for referencing back and forth from (x,y) coordinates to cells. For our implementation, we have $V = 600$ cells from [0,47]x[0,50] (we constrain ourselves to the left-hand half-court, since all movement in the offensive possessions are transposed to the left side of the court, and offensive possessions for each player are such that the ball and the player have crossed the half-court line for the respective direction of play). V is a parameter you can tune to be higher if you want to grid the court at a higher precision, at the expense of increased computation time and sparsity in your data.

Motivation

Simulating player movement on offense can sound like a daunting task. A simpler point of view on the same problem is that one needs to generate a new sensible spatial coordinate given the most recent spatial coordinates we observed. Such a model for player movement is explained in the aforementioned paper on estimating expected possession value (Cervone et al., 2016). The paper's summarized research output is a framework that uses the SportVU data to perform estimation of the expected number of points obtained by the end of a possession, by way of a stochastic process model that models the evolution of a basketball possession. Focusing our attention on the introduced microtransition model, the authors propose a model that describes player movement when a "major ball movement" does not occur (such as passes, shots, and turnovers). Though they build separate models for offensive and defensive players, we restrict our discussion here to the offensive player movement model, since that is the scope of the project.

For each player l, the next locations are given by:

$$\begin{aligned} x^l(t+1) &= x^l(t) + \alpha_x^l[x^l(t) - x^l(t-1)] + \eta_x^l(t) \\ y^l(t+1) &= y^l(t) + \alpha_y^l[y^l(t) - y^l(t-1)] + \eta_y^l(t) \end{aligned} \tag{1}$$

A player's coordinate at time $t+1$ is modelled as position at time t, plus the player's velocity from position at time $t-1$ to time t (weighted by a parameter α^l, which we set to 1), plus an η^l term which represents the contribution of higher order derivatives to the player movement (such as acceleration, jerk, etc.). These dynamics are nonstationary; in other words, the nature of η^l alters over space. Intuitively this makes sense, as players who are almost out of bounds will accelerate away from the edges of the court to stay in bounds, and players who accelerate toward the basket will generally decelerate when approaching their attempt to shoot.

When trying to generate a new (x_{t+1}^l, y_{t+1}^l), we trivially have (x_t^l, y_t^l) and (x_{t-1}^l, y_{t-1}^l); the challenge then is to generate a sensible η^l value. Instead of estimating the true distribution for η^l (which would involve estimating parameters for each player l) and sampling from that, we can opt for a data driven approach by collecting all of the η^l's we observe throughout the course of the game for the player, and then sampling from this collection directly whenever we want to determine a player's new position on the court.

Taking the formulas above, setting $\alpha_x^l, \alpha_y^l = 1$ and isolating for η^l, we have:

$$\begin{aligned} \eta_x^l(t) &= [x^l(t+1) - x^l(t)] - [x^l(t) - x^l(t-1)] \\ \eta_y^l(t) &= [y^l(t+1) - y^l(t)] - [y^l(t) - y^l(t-1)] \end{aligned} \tag{2}$$

From the rearranged equations we can see that player acceleration, which η^l captures, is simply the difference in player $l's$ velocity observed at the two previous time points. We refer to the collection of these calculated $\eta's$ as "empirical $\eta's$".

Obtaining the Empirical $\eta's$

Our filtered player moments aren't a completely connected sequence of movements. An NBA game has many stoppages; timeouts, out of bounds, and ends of quarters, to name a few. Consider some point in time t where (x_{t-1}, y_{t-1}) is the last moment recorded of one offensive possession. Since we have filtered out non-offensive moments, the (x_t, y_t) in our dataset is the first moment of a new offensive possession - very likely in a completely different spatial region. We need some way of recognizing situations like these,

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3201v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017

so that we don't consider these as natural movements from one location to another in our processing. We can do this by skipping any t where $dist(x_t, y_t) > \delta$, for some δ parameter we choose, which is what we do in *skip_this_iteration()*.

We chose δ by looking at the distribution of Euclidean distances from one moment to the next for LeBron James in our sample game, shown in Figure 1. The 99th percentile was 0.7524233, so we chose $\delta = 1$ as a safe, round number. For reference, Usain Bolt's record breaking 28 mph translates to 41 feet per second which equals 1.64 feet per 1/25th of a second.

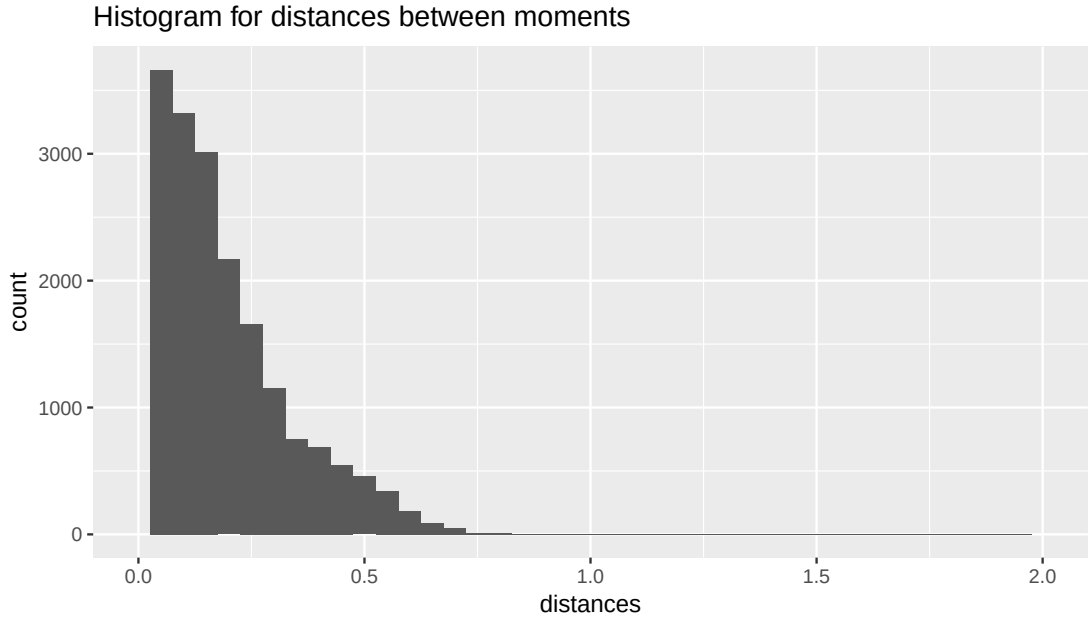

Figure 3. Distribution of movement distance from frame to frame for LeBron James for 2013/11/01 vs. the Brooklyn Nets

Recall that to generate our empirical $\eta's$, we need $(x_{t-1}, y_{t-1}), (x_t, y_t)$, and (x_{t+1}, y_{t+1}) for every $t = 2, \ldots, n-1$ (where $n =$ number of player's offensive moments). In the function *get_empirical_etas()*, we iterate through a given player's filtered moments $t = 2, \ldots, n-1$ (excluding $t = 1$ and $t = n$ because there is no previous and next moment for those time points, respectively), calculate the results from Equation 2, and encode the empirical η data in a matrix with 3 columns: cell, η_x and η_y.

Table 8. Output of get_empirical_etas() for LeBron James' on-ball data: there are 15425 rows for off-ball and 4777 rows for on-ball

	cell	x	y
2	253	0.21	-0.34
3	254	0.20	-0.32
4	254	0.20	-0.30
5	254	0.18	-0.27
6	254	0.16	-0.25
7	278	0.16	-0.22

VISUALIZATION

How can we visualize our results to validate what we have done? One way to visualize the empirical $\eta's$ is to show the average acceleration vector at each cell of the court - see Figure 4 in Cervone et al. (2016) for the type of image that we seek to produce. The idea is: for each cell $v = 1, \ldots, V$, if there were collected data at that cell, we take the cell's center as the beginning of the arrow and add the average

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3201v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017

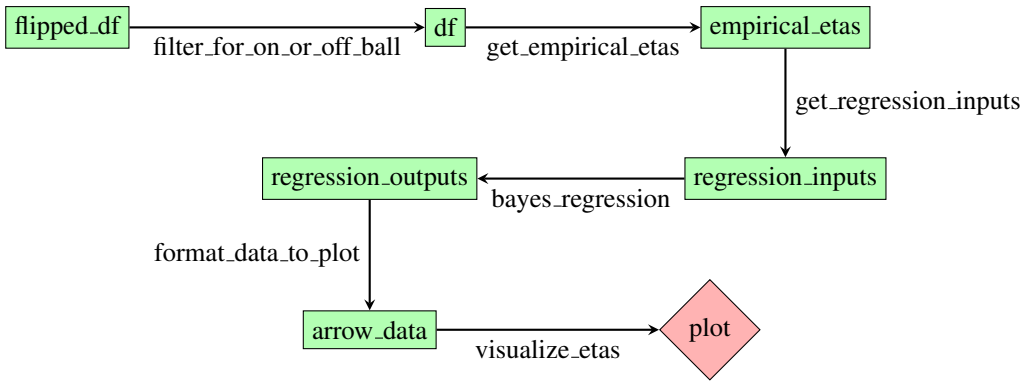

Figure 4. Flowchart of operations to extract the empirical $\eta's$ from a player's offensive moments and to produce visualizations

(η_x^l, η_y^l) to get the end of the arrow. The length of the arrow directly represents the magnitude of the average stochastic innovation observed at that cell. This is equivalent to running a linear regression twice, where each of the η_x, η_y is a response variable, and the $\mathbf{X}$ matrix is a one-hot encoded vector of which cell the η was observed in. However, this will result in a jerky plot (especially with a small sample size of the one game we are demonstrating). We use Bayesian regression, where we use a precision matrix to incorporate the spatial information of the cells and its neighbours, to give us smoother averages and thus a smoother visualization. The function *get_regression_inputs()* transforms the empirical η data into the design matrix and response variable that *bayes_regression()* will use to calculate the smoothed averages.

Table 9. Output of bayes_regression() for LeBron James' on-ball data: the result is a smoothed (x,y) pair for each of the V=600 cells

	ls_x	ls_y
V1	0.003937	-0.006572
V2	0.024007	-0.033389
V3	0.120113	-0.066792
V4	0.714535	0.069823
V5	0.568190	0.078631
V6	0.053480	0.006110

format_data_to_plot() turns our regression outputs into a format that enables straightforward plotting of our arrows. By applying the averaged magnitudes in each cell to the cell's center coordinate, the output can simply be the (x, y) coordinates of the head and tail of each arrow.

Table 10. Output of format_data_to_plot(): (x1, y1) is the center coordinate and (x2, y2) is the coordinate for the arrow head, for each of the V=600 cells

	x1	y1	x2	y2
1	1.000000	49.000000	1.001298	48.999463
2	3.000000	49.000000	3.009110	48.997654
3	5.000000	49.000000	5.048241	48.991181
4	7.000000	49.000000	7.270269	48.976741
5	9.000000	49.000000	9.201791	48.988723
6	11.000000	49.000000	11.269096	48.986372

Lastly, we need a function, *visualize_etas()*, that will take this formatted arrow data and do our plotting. The colors and size indicate the magnitude of the empirical η vector. For the code to plot the court itself, we stand on the shoulders of past giants (Gallic, 2014) and leverage existing open source code. The output of the plotting function for LeBron James in the game we are analyzing is visualized in Figure 5.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3201v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017

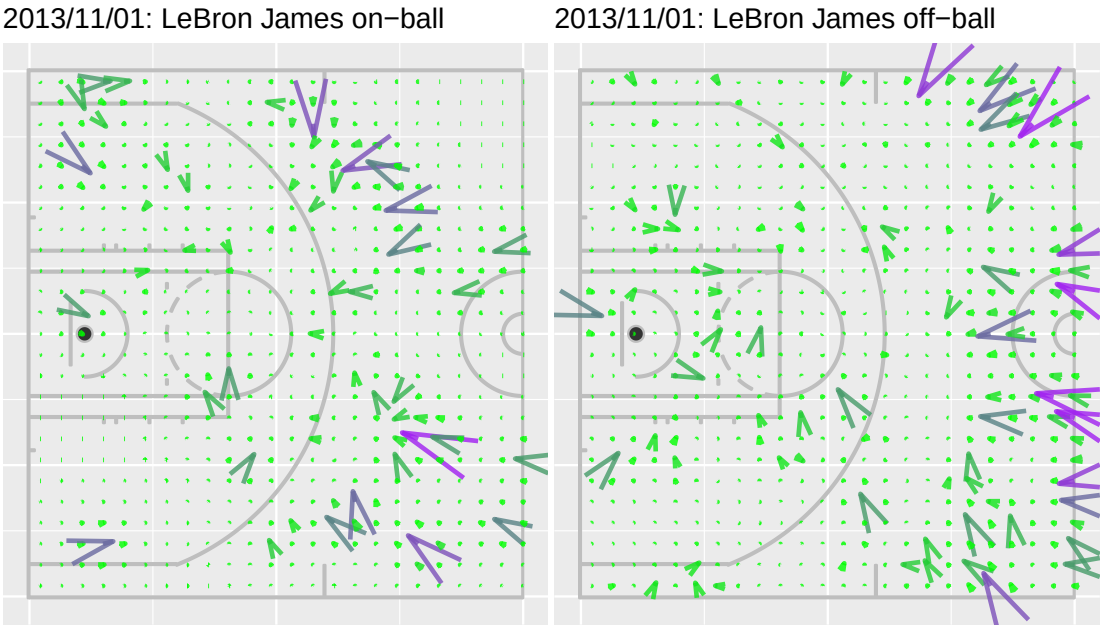

Figure 5. Visualization of smoothed empirical acceleration vectors for LeBron James for 2013/11/01 vs. the Brooklyn Nets

CONCLUSIONS

Looking at Figure 5, there are some exaggerated $\eta's$ due to the small sample size of one game. Using a season's worth of data for a player, 82 games, results in significantly smoother plots. The smoothed season plots are generated for the 2015-2016 NBA season for Andre Drummond, LeBron James, and Steph Curry in Figures 6, 7, and 8 respectively.

Upon visual inspection of these figures, the smoothed out results across an entire season make intuitive sense and pass the "smell-test" that is commonly applied when analytical methods are tried on sports; failing the "smell-test" occurs when a result derived from quantitative analysis is completely backwards from accepted consensus. We conclude with a discussion of how our results provide a visual explanation behind our intuition that these three NBA stars move using very different styles on offense.

Off-ball Plots

In general, acceleration occurs away from the out-of-bounds lines, as players are penalized with a turnover in ball possession if they step out of bounds. There is another visible trend of general acceleration toward the hoop, with the values being higher near the half-court (since players often run from their defensive half to the offensive half to set up in their team's offensive sets).

Notice how that half-court effect is the most pronounced for a center like Andre Drummond when he is without the ball. It is increasingly common to find 7 footers in the NBA like Karl Anthony Towns and Kristaps Porzingis who can shoot from beyond the 3-point line with ease; however, Drummond's offensive contributions are almost exclusively from inside the paint. As the main hub in coach Stan Van Gundy's scheme (the same one successfully deployed in the past coaching Dwight Howard in Orlando, a center with similar strengths and limitations), Drummond's priority on offense is to occupy the paint area as quickly as possible. Drummond is also the center anchor of his defense as a shot-blocking rim protector. Thus, most of his movement in his transition from defense to offense occurs in a straight line down the middle of the court between both baskets.

LeBron is a perimeter playing forward. Though versatile in the positions that he can play and in the areas of the court that he can excel in, his primary defensive assignments will have him situated on either wing. With his extreme athleticism, surrounded by good passers (including a particularly notable outlet passer for fastbreaks in Kevin Love) and a teammate who can assume primary ball-handling duties in Kyrie Irving, LeBron can sprint ahead during the transition from defense to offense to attempt a high percentage fastbreak opportunity. Thus, LeBron's acceleration is largest coming from the wings.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3201v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017

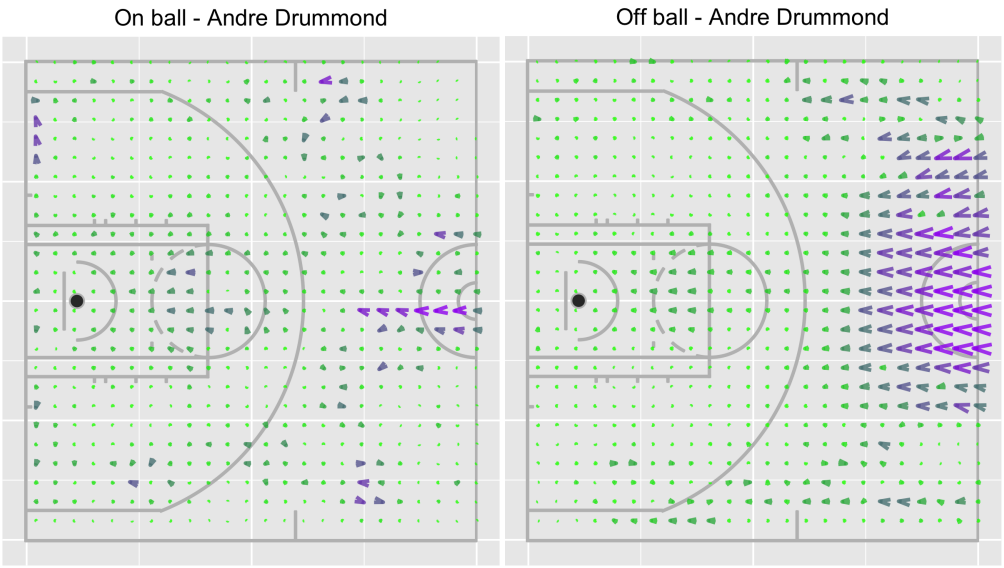

Figure 6. 2015-2016 Andre Drummond Empirical $\eta's$

Steph Curry in comparison has relatively small acceleration values. He is the primary ball-handler on his team and is much more likely to be leading a fastbreak opportunity with the ball in his hands than running on the wing without the ball as a finisher like LeBron. Famously known for effortlessly draining 3-point shots several feet past the 3-point line, Curry doesn't need to accelerate much to end up in a scoring position once he crosses the half-court line.

On-ball Plots

As a player with traditional NBA center limitations, Drummond does not consistently accelerate with the ball with an intent to score. His offensive strengths consists of attempting shots within the paint immediately after catching the ball. His other shot attempts include shots taken after backing his defender down in the post (which consists of dribbling the ball with his back to the basket to inch closer towards it, generating almost no acceleration).

LeBron is a prolific driver to the basket with the ball in his hands and has perenially been one of the league's best at this skill since he entered the league. He has a unique combination of height, body strength, and dribbling ability which allows him to deflect and absorb contact from defenders on his way toward successfully scoring near the rim.

Comparing Curry's plot to LeBron's, Curry has an even wider range of angles where he successfully accelerates toward the hoop while dribbling. Curry's magic is that he is not hyper athletic like LeBron (or even a positional counterpart, like Russell Westbrook). His ubiquitous shot-making ability coupled with one of the best dribbling abilities in the NBA explain why he had a historic offensive season in his 2015-2016 Most Valuable Player campaign. However, his on-ball plot helps illuminate exactly how much more action he was able to generate toward the rim with the ball in his hands relative to even a top performing peer such as LeBron.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3201v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017

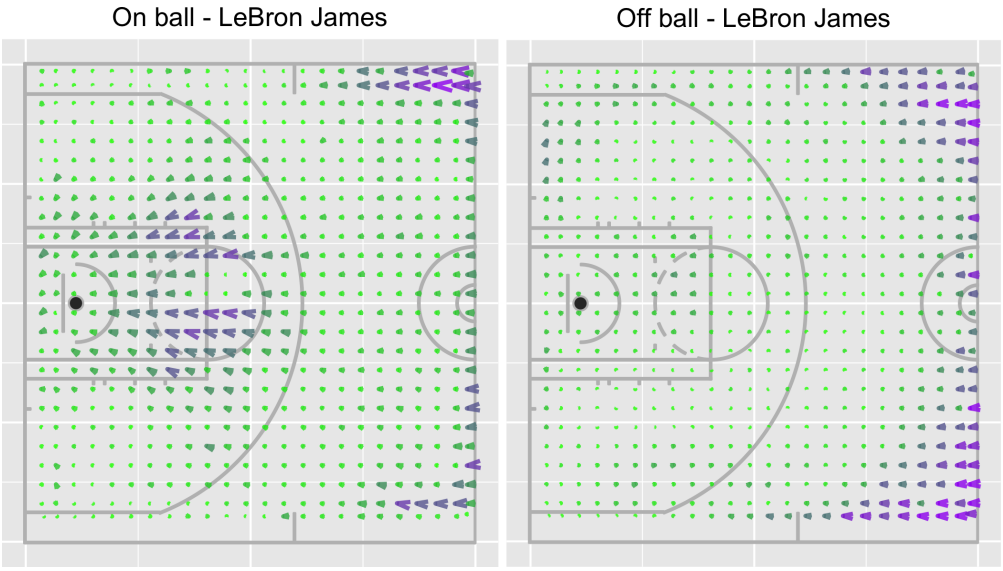

Figure 7. 2015-2016 LeBron James Empirical $\eta's$

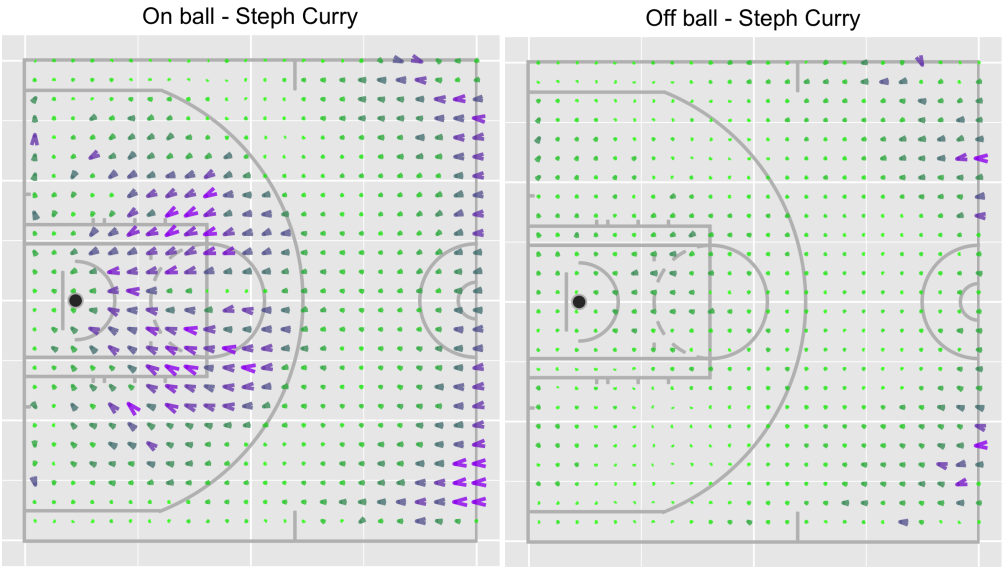

Figure 8. 2015-2016 Steph Curry Empirical $\eta's$

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3201v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017

Applications

From the discussion of the visualized results for Drummond, Curry, and LeBron, we see that a player's empirical $\eta's$ can be used as features to describe different types of playing styles on offense. Analysts of the NBA are generally interested in methods that can quantify new insights that impact player evaluation. Comprehensive player evaluation allows teams to tackle higher level questions; some examples include game preparation (preparing for players on the opposing team) and roster management (scouting for possible acquisitions to a team's roster).

Thanks to increased discourse and acceptance of the role of analytics in the NBA, offense has trended toward a focus on having a higher proportion of field goal attempts of either three-point shots or shots resulting from drives toward the rim. This has resulted in a premium being placed on finding players who excel at either skill (or ideally, both). Players who consistently show the ability to attack the basket with the ball from the perimeter create a lot more value than just an attempt at a high probability field goal make; they can cause perimeter help defenders to leave their marks which results in uncontested three-point shots, and they can cause interior help defenders to leave their marks which can result in easier shots in the paint. These visualizations can help identify such players who can directly cause disorganization in the opposing team's defense.

There are some existing metrics which can approximately quantify this skill, but they can come up short if you want a more complete picture of a player's ability to drive towards the opposing basket. Number of drives per game misses the spatial context of each attempt; a general manager would not want to compose a roster where all their best drivers favor going right, as that can clog up the spacing of the team's offense. Number of made layups/dunks per game doesn't help you find players who execute an attacking drive correctly at every point in time and miss the final finish. If you can find a player who has an on-ball plot similar to Steph Curry but can't finish like him yet, you can acquire him at a discount of his market value and focus on improving that weakness during player development training. Being able to quantitatively identify players who have the potential to blossom into stars offers a large edge in the NBA, where player acquisitions are constrained by salary caps and competing interest from other teams.

REFERENCES

Cervone, D., D'Amour, A., Bornn, L., and Goldsberry, K. (2016). *A Multiresolution Stochastic Process Model for Predicting Basketball Possession Outcomes*.

D'Amour, A., Cervone, D., Bornn, L., and Goldsberry, K. (2015). *Move or Die: How Ball Movement Creates Open Shots in the NBA*.

Forsyth, D. (2015). Exploring nba data in python. `http://www.danielforsyth.me/exploring_nba_data_in_python/`. Accessed: 2017-06-21.

Franks, A., Miller, A., Bornn, L., and Goldsberry, K. (2015a). *Characterizing the Spatial Structure of Defensive Skill in Professional Basketball*.

Franks, A., Miller, A., Bornn, L., and Goldsberry, K. (2015b). *Counterpoints: Advanced Defensive Metrics for NBA Basketball*.

Gallic, E. (2014). Drawing a basketball court with r. `http://egallic.fr/drawing-a-basketball-court-with-r/`. Accessed: 2017-06-21.

Hijmans, R. J. (2016). Introduction to the 'raster' package. `https://cran.r-project.org/web/packages/raster/vignettes/Raster.pdf`. R package version 2.5-8.

Johnson, N. M. (2015). Basketball data github repository. `https://github.com/neilmj/BasketballData/tree/master/2016.NBA.Raw.SportVU.Game.Logs`. Accessed: 2017-06-21.

Reda, G. (2015). Web scraping 201: finding the api. `http://www.gregreda.com/2015/02/15/web-scraping-finding-the-api/`. Accessed: 2017-06-21.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3201v1 | CC BY 4.0 Open Access | rec: 28 Aug 2017, publ: 28 Aug 2017