**Game of Trades: Using simulation to get an edge in the NBA trade market**

Basketball

ID: 5643

1. **Abstract**

In this paper I present a method to evaluate the impact of trades for NBA teams by simulating different scenarios. Assessing the potential impact of trades is a complex task as it affects not only the composition of the teams involved but also the rest of the league. For instance, the Kyrie Irving – Isaiah Thomas trade not only affects the number of wins for Cleveland and Boston but also for any team playing against these 2 teams.

The main idea is to estimate the offensive and defensive power for each team given a team roster and players’ usage. Assuming a Normal distribution in teams’ scored points, these powers serve as estimated means while the variance for both models is estimated from empirical data. Once the probability distribution is known, I can calculate the probability of any matchup and thus the number of wins in the regular season.

I will show how this model has implications beyond the trade market evaluation.[[1]](#footnote-1)

1. **The Data**

For the main model and further analysis I used the following data variables available from basketballreference.com:

"Age" "G" "GS" "MP" "FG" "FGA" "FG." "X3P"

"X3PA" "X3P." "X2P" "X2PA" "X2P." "eFG." "FT" "FTA" "FT." "ORB" "DRB"

"TRB" "AST" "STL" "BLK" "TOV" "PF" "PTS" "Season"

Once the column stats have been adjusted to per minute stats, I use the following nomenclature:

"Age" "Exp" "FGPer" "FG3Per" "FG2Per"

"effFGPer" "FTPer" "effMin" "effFG" "effFGA" "eff3PM" "eff3PA" "eff2PM" "eff2PA"

"effFTM" "effFTA" "effORB" "effDRB" "effTRB" "effAST" "effSTL" "effBLK" "effTOV"

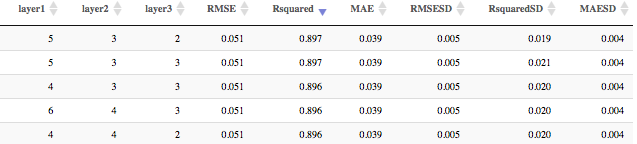
"effPF" "effPTS"

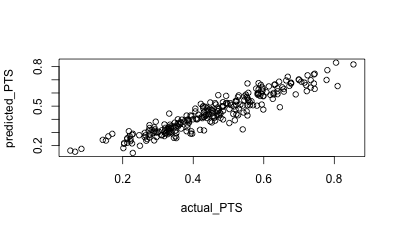
1. **The Model**

The model consists of 2 Neural Networks used to estimate team powers: One for Offense (points scored), one for Defense (points against). Inputs are players' projected per minute stats weighted by their share of minutes of play.

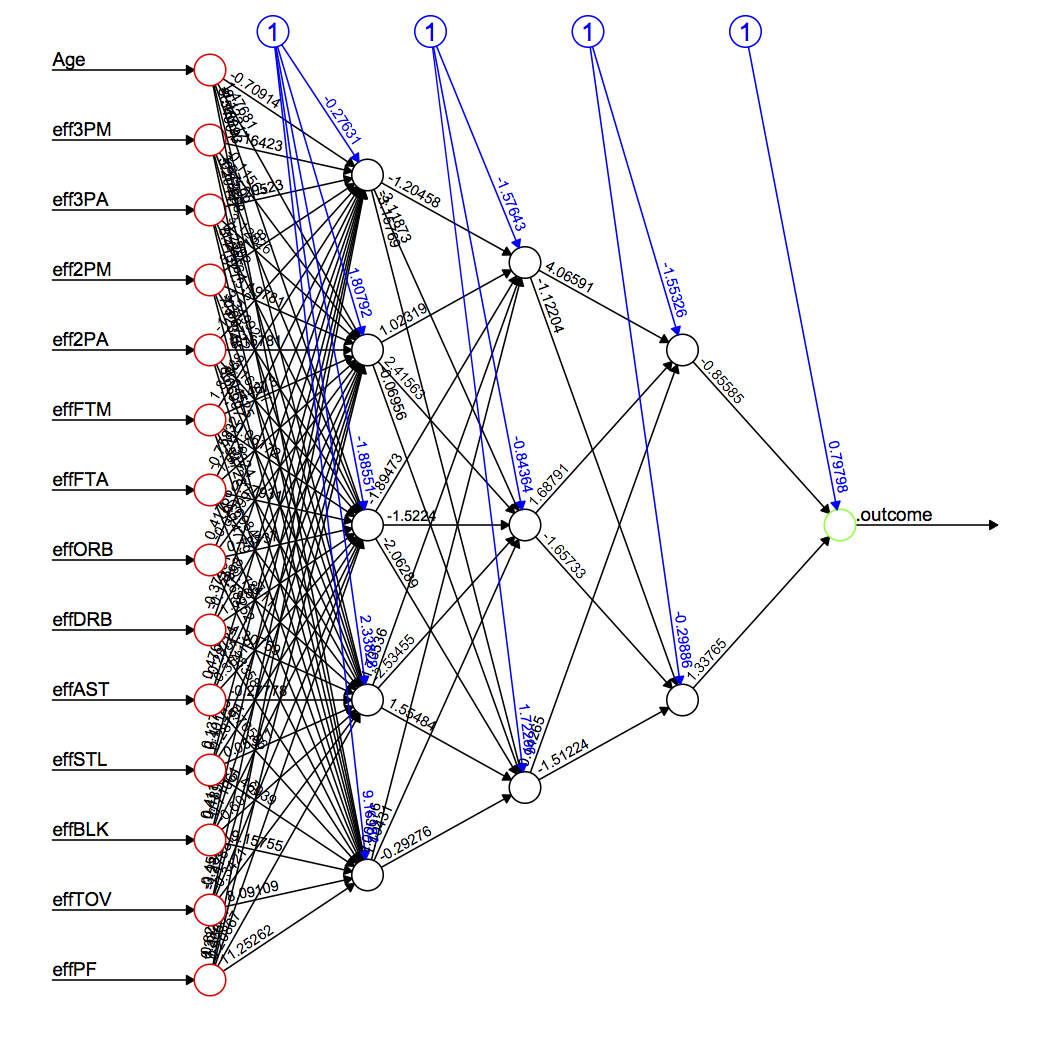
Steps:

* Read historical players: From Season 1979-1980 when the 3 point shoot was established. (write\_playersHist.R) (data: playersHist.csv) . Size: 18927 records.
* Differentiate players with the same name by adding a number after the name in ascending order by decreasing age. (.rename\_PlayerName\_Duplicates.R). Example: Tim Hardaway who played in the 90s and Tim Hardaway 2 (current NYK player) as basketballreference does not differentiate them and I use Player as primary key.
* Calculate stats per minute of play: effStat = Stat/MP. And effMinutes as Minutes played over total possible minutes: 82 games \* 48 minutes per game: effMin = MP/3936 (.team\_prepareAll() in prepare\_rosters.R). Finally, adjust effMin relative to total minutes played by team. effMin = effMin/sum(effMin)
* The input vectors for the neural network is the weighted average of all stats per team per season. Size: 1063 records.
* Remove columns that depend linearly on others: FG, FGA, FG%,3P%,2P%,FT%,effFG%, effPTS.
* Scale the data [0,1] for easier convergence of backpropagation algorithm.
* Split sample in 75-25% training-testing. Used 10-fold cross-validation with 10 repetitions (leave one out).
* Train a neural network with 3 layers: nnetGrid <- expand.grid(layer1 = c(4,5,6), layer2 = c(3,4), layer3 = c(2,3)). Used neuralnet package under caret package in R.
* Run the network twice: Same inputs, different outputs: Average points scored and average point scored ag ainst per team per season.
  1. **Neural Network model**
     1. **Offense**

Best Offense model (5-3-2) based on R-squared: top 3 results

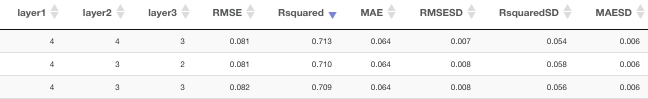
Fit on testing data (25% = 266 observations)

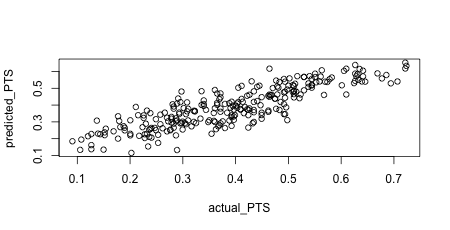
The Network



* + 1. **Defense**

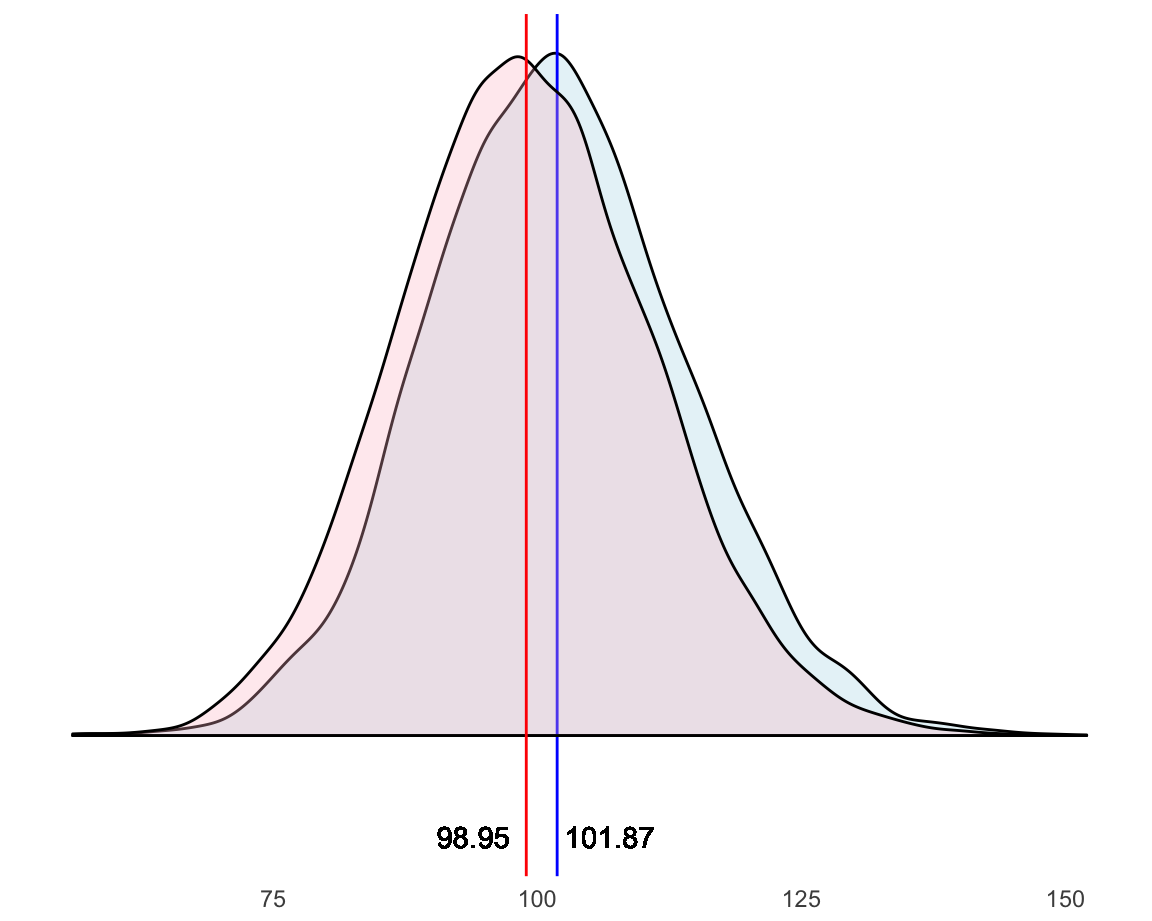
Same drill for the defense network. Here the results:



Clearly not a great fit but still good enough.

* 1. **Probability model**

Now that I have a way to estimate team’s offensive and defensive powers (average points), I can plug those into a Normal distribution probability. The choice for the Normal distribution is obvious if we explore the density of points scored by teams in the last 8 seasons: (since 2009-2010)



Where the blue density corresponds to points scored by home teams and red to away teams. Empirically, we can then assume that the average points scored are 100.41 with a standard deviation of: 12.16 pretty much consistent for home and away teams (sigmaH = 12.13, sigma = 12.0). For the sake of simplicity I will assume the sd is constant across the board which allows me to fully determine the probability model for both Offense and Defense for a given team:

$\Large{X \sim \mathrm{N}(\hat{\mu}\_O,\hat{\sigma})}$

* 1. **Wins**

Now that I can compute how many points does a team score or get scored on average, I will calculate how many wins those offensive and defensive powers award. Let’s suppose teamA plays against teamB. Let’s also suppose teamA is the home team. I know how many points teamA scores on average, call it: ptsA. Empirically we saw in 3.2. that home teams score 1.46 more points than average, thus, ptsA\* = ptsA + 1.46. Next, I need to consider teamB defensive power, call it: pts\_agB. The higher the defensive power is, the worst the defense is (as opposed to offense). So, I will add this number to teamA’s offensive power: ptsA\*\* = ptsA\* + pts\_agB. This will give me an astronomical number of points scored or a basketball game, which will be cancelled out when I calculate teamB’s offense. But in order to not just calculate the win probability but also simulate a plausible score between these 2 teams, I will subtract the overall average points (100.41) from that number. So, here’s how I calculate the points teamA will score against teamB when teamA is the home team:

ptsA\_final = ptsA + home\_court\_coeff + pts\_agB – overallPtsAvg

And for teamB as the away team:

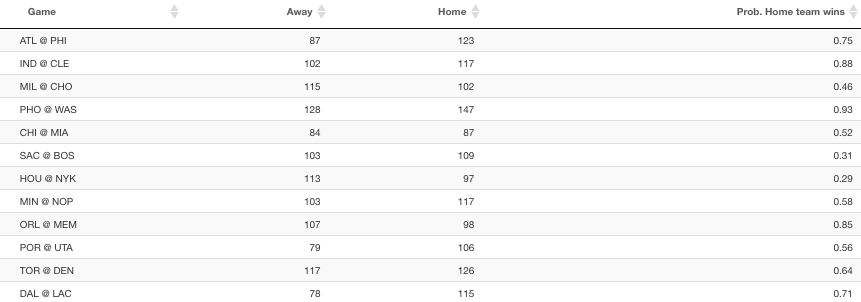
ptsB\_final = ptsB - home\_court\_coeff + pts\_agA – overallPtsAvg

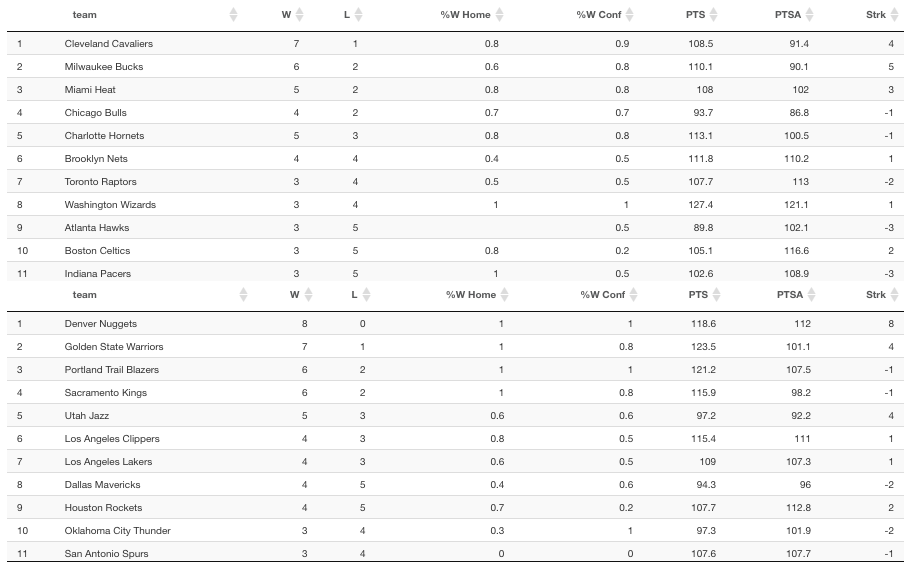
Now I can plug these 2 measures as the means of Normal distributions with the same sd = 12.16 and can calculate the probability of teamA beating teamB:

Let X = N(ptsA\_final,sd) and Y = N(ptsB\_final,sd) random variables. Then:

P(teamA beats teamB) = P(X > Y) = P(X-Y > 0) = P(N(ptsA\_final- ptsB\_final,sqrt(2)\*sd) > 0)

If I calculate these probabilities for all games in a regular season, I can analytically obtain expected wins for each team. I will show the details in the results section below but as an example, take one random day in the regular season, say November 1. This is an extract of the application dashboard:

Besides giving the win probability of the home team, I can simulate an actual game score, and thus, I can easily come up with standings and simulate an entire season. Below the Eastern Conference standings as of November 1 according to my simulation:



* 1. **Player similarity**

One of the key aspects of this model consists on being able to compare and classify players, and eventually teams. I use the t-SNE algorithm to reduce the multidimensional space defined by players’ skills into 2 dimensions. In a 2-D space I can easily calculate Euclidean distances between players and at the same time visualize these distances between players.

The inputs to this algorithm are players skills and the output are x,y coordinates which we can map using a cloud of points. I use t-SNE in different situations which will be described below, in brief:

* Similar players by age: To predict how skills of players evolve with age
* Similar players historically: To see players’ skills evolution over time
* Similar players for the current or any season: For clustering purposes

t-SNE algorithm is well described by its author. See references. Also, this article provides more insights on how the algorithm performs and what the parameters mean. I’ll leave the details to the reader to not interrupt the flow of the paper.

* 1. **Building the roster**

So far, I used historical data to build a model that computes win probabilities based on team’s offense and defense. But, for this to be useful I need to be able to predict these offensive and defensive powers based on the players that make each team at a given point. I need to do 2 things: Assess players’ skills and estimate players’ usage.

At the start of a new season we find 4 type of players:

1. Returning NBA players who played at least 1 minute in the previous season
2. Returning NBA players who didn’t play in the previous season
3. Rookies drafted from American schools
4. Rookies from international teams

For each of these I will use a different approach mostly determined by the disparity of data sources.

* + 1. **Returning players (cases a and b)**

For cases a and b run .computePredictedPlayerStats() from write\_teams\_predicted\_stats\_new\_season.R which returns: playersNewPredicted\_Oct20.csv

1. The algorithm runs through all players from last season and computes: .predictPlayer() (from similarityFunctions.R):
   * Top 10 similar players: run .similarPlayers() (from similarityFunctions.R): Using .tSNE\_dist() function I find the top 10 players closest in Euclidean distance in the coordinate system defined by t-SNE age map for the past 5 years of age. Similar Players returns the top 5 from that list (if any). The only constraint is that players younger than 20 or older than 39 are so scarce that their tSNEs don’t really make sense. My approach is to assign players the latter to the class of 20 year olds and the former to the 39 year olds. Provide full example here.
   * Now calculate average variation in their stats when they went from current age to age + 1. Use .tSNE\_prepare from similarityFunctions.R). The motivation behind this comes from FiveThirtyEight Carmelo. See References.
2. Possible outcomes of .predictPlayer():
   * Returns stats: Returning player who recorded stats for enough minutes of play for last season. I use the methodology described above to predict their stats.
   * Returns empty stats or no stats at all: Returning player who didn’t record stats for last season (he played in another league or was injured) or player who didn’t play enough minutes in this career. I use his stats in his last season as a player as their predicted stats.
   * Any other scenario: Players with no previous experience, I use the average player stats to predict their stats. Average player’s skills take the mean of all the numeric skills of all players’ stats from last season.

Final precaution, make sure no shooting percentages are > 1: playersNewPredicted <- mutate\_at(playersNewPredicted, vars(contains("Per")), function(x) ifelse(x >=1, quantile(x,.99), x))

* + 1. **Rookie players (cases c and d)**

The basic algorithm is write\_RookieStats() from write\_rookiesDraft.R which returns rookieStats.csv. Here are the details:

1. To run the algorithm I need to prepare the following files:
   * rookies.csv (writeAllRookies() in write\_rookiesDraft.R): Retrieves all rookies in the current rosters (The Experience field = “R”) with no stats records. I need to match this file with Players in this file:
   * collegePlayers.csv (write\_CollegePlayers() in write\_rookiesDraft.R): Querys college players who played at least 15 games and 7 min/game last season. Totally arbitrary numbers under the assumption that if fit for the NBA they would at a minimum played these type of minutes. For those players who played for more than 1 season in college, I take the average of their career stats.
2. These 2 files are retrieved from different tables in basketballreference and sometimes their names are spelled differently. I have to manually change these:

collegePlayers[grepl("Nazareth Mitrou-Long",collegePlayers$Player),]$Player <- "Naz Mitrou-Long"

collegePlayers[grepl("Royce O'Neale",collegePlayers$Player),]$Player <- "Royce O'Neal"

collegePlayers[grepl("Jacorey Williams",collegePlayers$Player),]$Player <- "JaCorey Williams"

collegePlayers[grepl("Andrew White III",collegePlayers$Player),]$Player <- "Andrew White"

collegePlayers[grepl("TJ Leaf",collegePlayers$Player),]$Player <- "T.J. Leaf"

collegePlayers[grepl("Frank Mason",collegePlayers$Player),]$Player <- "Frank Mason III"

collegePlayers[grepl("Akim Mitchell",collegePlayers$Player),]$Player <- "Akil Mitchell"

1. For unmatched players, I will query the European database for players for their stats in either the Euroleague, domestic league or overall and match them with players in rookies.csv.
2. For those still not matched, if they are international players (playing in Europe or not) they are assigned the means of all the stats of players coming from Europe. For the rest (college players who for some reason were not matched or found) I assign the means of the stats of rookie college players.
3. Finally, because college and European players play fewer minutes, I need to take this into account when computing their minute usage comparable to NBA players. My take is this is the price they pay to jump into a more competitive league so I assume their effective per minute stats are based on 48 minute games, that is: effMin = MP/3936 (use write\_Rookies\_efficientStats from write\_rookiesDraft.R which returns: rookieEfficientStats.csv). Still, this is not realistic in most cases, as college players tend to play way fewer minutes on their rookie year than they would in college. To account for this discrepancies we can take a look at the percentage of minute reduction experienced by rookie NBA college players according to their draft pick. Data covers the last 20 years and it’s limited to players who played at least 30 games in their rookie year. Clearly, being a lower draft pick guarantees higher usage on the rookie year. Subsequently, I add the pick round to my dataset and adjust effective minutes based on the chart below:



* + 1. **Putting it all together**

At the start of a new season, in particular October 20 2017, rosters are mostly final so the starting point as the list of current players participating in the new NBA season is current\_rosters.csv which contains players and teams as of Otober 20 2017, that is right at the start of the new season and it’s obtained from: .getLatestRosters(thisSeason="2017",previousSeason = FALSE) from write\_teams\_predicted\_stats\_new\_season.R

Now I will merge this file with the previously calculated stats for returning NBA players and rookie players described in 3.5.1 and 3.5.2. If everything worked well, there will only be non-matching players because of spelling differences. In case not, and there are still players who were added last minute after we run 3.5.1 and 3.5.2, then we can run: .computePredictedPlayerStats\_Leftovers() # from compute\_PredictedLeftovers.R.

These are the unmatched players due to spelling differences:

current\_rosters[which(current\_rosters$Player == "Gary Payton II"),]$Player <- "Gary Payton 2"

current\_rosters[which(current\_rosters$Player == "Glenn Robinson III"),]$Player <- "Glenn Robinson 2"

current\_rosters[which(current\_rosters$Player == "Kelly Oubre Jr."),]$Player <- "Kelly Oubre"

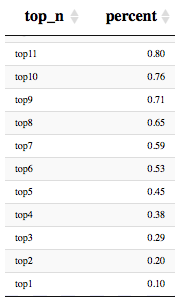
current\_rosters[which(current\_rosters$Player == "Nene"),]$Player <- "Nene Hilario"

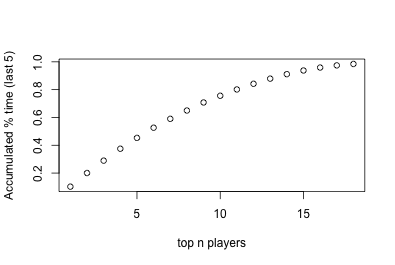
current\_rosters[which(current\_rosters$Player == "Taurean Prince"),]$Player <- "Taurean Waller-Prince"

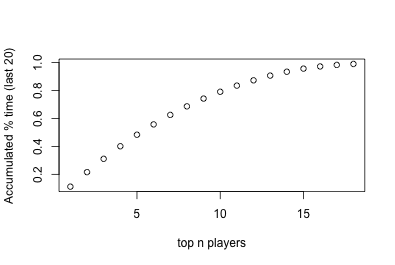
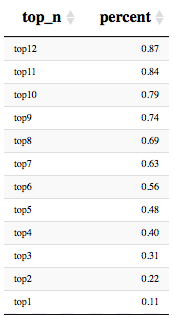
current\_rosters[which(current\_rosters$Player == "Tim Hardaway"),]$Player <- "Tim Hardaway 2"

current\_rosters[which(current\_rosters$Player == "Sheldon Mac"),]$Player <- "Sheldon McClellan"

* + 1. **Adjusting player usage at team level**

So far the computed statistics and effective minutes of play were computed individually for each player regardless of the team they play for. Needless to say, teams composed of many players playing heavy minutes will easily create an unbalance. The first step would be to transform minutes of play into percentage of play time with respect with total team minutes. This method will balance out the minutes but may not yet reflect a realistic distribution of minutes. Empirically, I looked at the average distribution of playing time. It is interesting to see how in recent years minutes of play become more spread throughout the roster. For instance, top 7 players accounted for 59% of total minutes on average in last 5 years while for the last 20 years, it accounts for 63%.





Considering this, I adjust playing time with the following parameters: top 7 players in each team sum up to 60% (round number in between 59% and 63% giving more preeminence to recent years) of total playing time with no player accumulating more than 10.5 % of it (averaging out top 1 percentage from both examples above). For instance, in a team with 3 star players, their total share of minutes may be as high as 31.5% leaving the remaining 28.5% for 4 players.

1. **The Results**

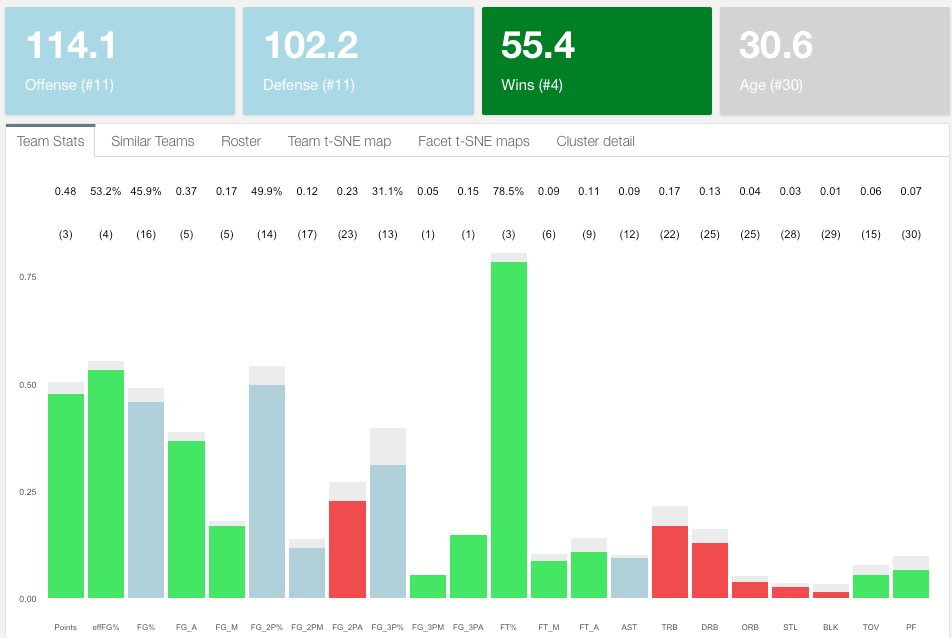
Once I defined the probability model and the inputs, i.e., players’ stats and share of playing time, I can compute expected wins by running a simulation of the regular season using win probabilities. Interestingly, I can also treat players as if they were teams, that is, I can evaluate the offensive and defensive strength of players by applying the neural network model to them under the assumption they are teams composed of many versions of the same player. Example: How good offensively and defensively would a team be with 5 Kevin Durants on the court?

* 1. **The Teams**

With these premises I can already present some team stats in a dashboard.

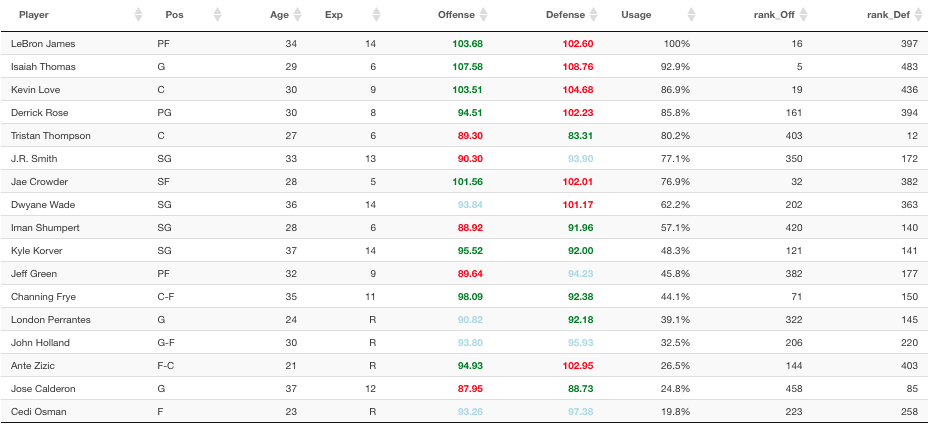
* + 1. **Team stats**

I rank teams in all categories and split them in 3: top third, middle third and bottom third assigning, respectively, colors green, light blue and red. In the example for the Cleveland Cavaliers, I can quickly spot strengths and weaknesses. My model gives them 55.4 wins in the regular season, they rank #11 in both offense and defense and they are expected to be #1 in 3 points made and attempted although only #13 in 3-point percentage. Clearly their weak spot is in rebounding, blocks and steals. Unsurprisingly, 2-point attempts ranks low as they are expected to be a shooting more 3s. Finally, they are the oldest team in the league averaging 30.6



* + 1. **Roster stats**

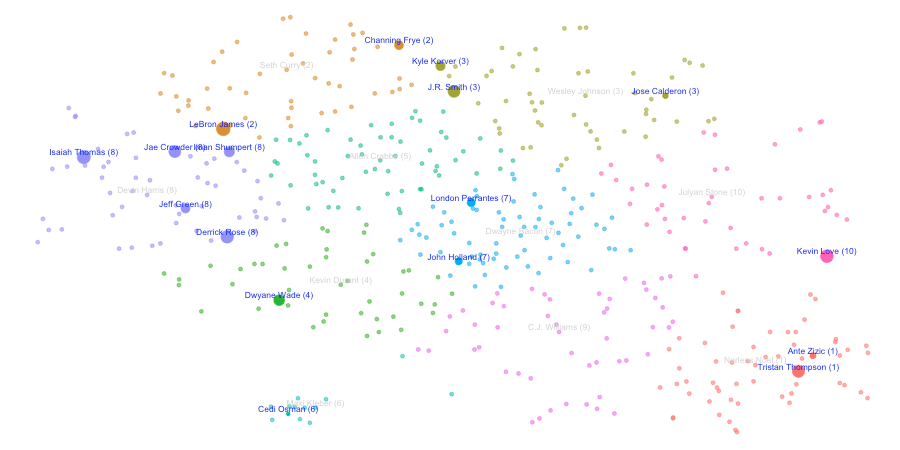
I can look at the composition of the team and individually spot offensive and defensive contributions and usage by player. Again, green means top third and red bottom third. Lebron, Isaiah and Love are all top 20 offensively and they all have a high usage rate as expected. Defensively, Tristan Thompson is their best asset ranking #12 in the league. Cleveland also features 2 players who are in the top third in both offense and defense which we will see is uncommon and provides great value added to a team: Kyle Korver and Channing Frye. In the players section I provide a more detailed analysis.

****

* + 1. **Similar teams**

ensively

* + 1. **Team t-SNE map**

Using t-SNE I can plot each player on a 2-D map and perform different exploratory and analytical analysis. This is how Cleveland players are distributed based on t-SNE. Colors correspond to K-means cluster of size 10.

I could add more clusters or reduce the number of them. Clusters represent different types of players and in order to make a more informed decision on how many clusters (or player types) exist in the league, I can look at the stats that define each of the clusters. If we stick to 10 clusters:

This shows LeBron is placed in the same cluster (2) as Channing Frye. A closer inspection shows that this is the group of veteran players that play low minutes and shoot from long range and assist but don’t have great percentages and don’t rebound, steal or block very often.





My view is a higher number of cluster will possibly represent better the player typology. See now for 15 clusters:

* + 1. **Cluster analysis**
  1. **The Players**

This i

This is the first

• A Normal distribution is centered around each team’s estimated power (offense and defense) and a fixed common variance (based on empirical data). With the full probability distribution I can simulate any matchup.

• Player similarity by age is computed using t-SNE algorithm which also allows for 2-D visualization of the data..

As conference policy, we do not support LaTeX, so we ask that you use this template instead. We understand that math typesetting can be more cumbersome in MS Word, but we suggest using MS Word’s equation editor. Equations will look like this:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

This is a reference to equation (1) that updates after fields are updated. Notice that each equation is contained in its own table, and the equation numbers are inserted using fields. See [this tutorial](https://www.youtube.com/watch?v=wM57WvO20KA) for more information on this technique.

We do not insist on any specific conventions related to figures, tables, and captions.

1. **Section**

This is the body of text under the third main section.

1. **Section**
   1. **Subsection (Cambria, Bold, 12pt)**

This is the first paragraph of the body of text under the first subsection of the first main section.

This is the second paragraph to give you sense of the spacing.

* + 1. **Sub-subsection (Cambria, Bold, 12pt)**

This is the first paragraph of the body of text under the first subsection of the first main section. Subsections can be nested as far as you want, though the font for the subsection headers remain the same (Cambria, 11pt).

* 1. **Subsection**

This is the first paragraph of the body of text under the first subsection of the first main sectionThis is the body of text under the fourth main section.

**References**

[1] Reference #1 cited using any mainstream citation style (e.g. APA, MLA).

[2] Reference #2

.

.

.

[n] Reference #n

**Appendix**

An appendix is not required, but if you have one please include it here.

1. Footnotes are permitted and should be formatted as shown here (Cambria, 11pt). [↑](#footnote-ref-1)