**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. 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 how many wins will Cleveland and Boston obtain on the regular season but also has ripple effects and will alter the number of wins for any team playing these 2 teams.

The main idea is to estimate the offensive and defensive power of each team given the team roster and players’ minute share. Assuming the number of points a team scores follows a Normal distribution, the means are the offensive and defensive powers 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.

In the first part of the paper I will describe the proposed model and methodology. The second part is mostly devoted to examples and use cases. I developed an R dashboard[[1]](#footnote-1) that I use to interact with the model and the data. I recommend the reader to play with it as you read the paper for a more interesting experience. All the code I used is available on my Github account: asRodelgo/NBA.

1. **The Data**

The basis of every statistical analysis in the paper is players’ main measurable stats. My only source of data is basketballrefence.com. There are many advanced statistics available however my goal was to have as many years of consistent stats as possible for all NBA, European and College players. This allows me go back in time to 1979-1980 where 3-point stats were first recorded. I also avoid stats that are not directly measurable from a player action, like box plus-minus. So, here are the stats I use[[2]](#footnote-2):

Age, G, MP, FG, FGA, FG%, 3PM, 3PA, 3P%, 2PM, 2PA, 2P%, eFG%, FT, FTA, FT%, ORB, DRB, TRB, AST, STL, BLK, TOV, PF, PTS

These variables measure per game stats but my interest is in stats per minute. I use the following nomenclature after transformation:

Age, 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**

My objective is to model how many games will a team win in a given regular season. Because wins depend on the capacity of teams to score points scored and not receive them[[3]](#footnote-3) I will model how many points will a team score (offensive power) and receive (defensive power) on average.

* 1. **Neural Network model**[[4]](#footnote-4)

For each offensive and defensive power I will use the output of a Neural Network. For inputs I will use weighted average of players' projected per minute stats. Where weights are their share of minutes of play. Here are the steps I take:

1. Read historical players stats per game starting with season 1979-1980 from: <http://www.basketball-reference.com/leagues/NBA_2017_per_game.html>
2. Player name is my primary key so I need to differentiate players with the same name by adding a number after the name in ascending order by decreasing age. Example: Tim Hardaway who played in the 90s and Tim Hardaway 2 (current NYK player).
3. Calculate stats per minute of play (effBLK, eff3PM, etc) for each player i:

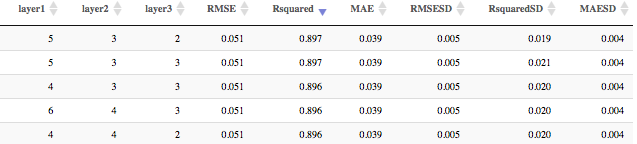
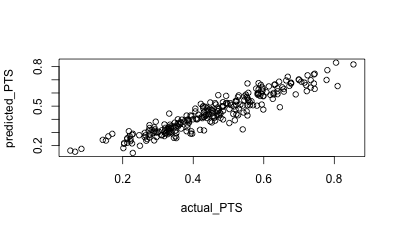
And effective minutes as MP (minutes played) over total possible minutes (48 minutes in 82 games):

Finally, adjust effMin relative to total minutes played by team Tm:

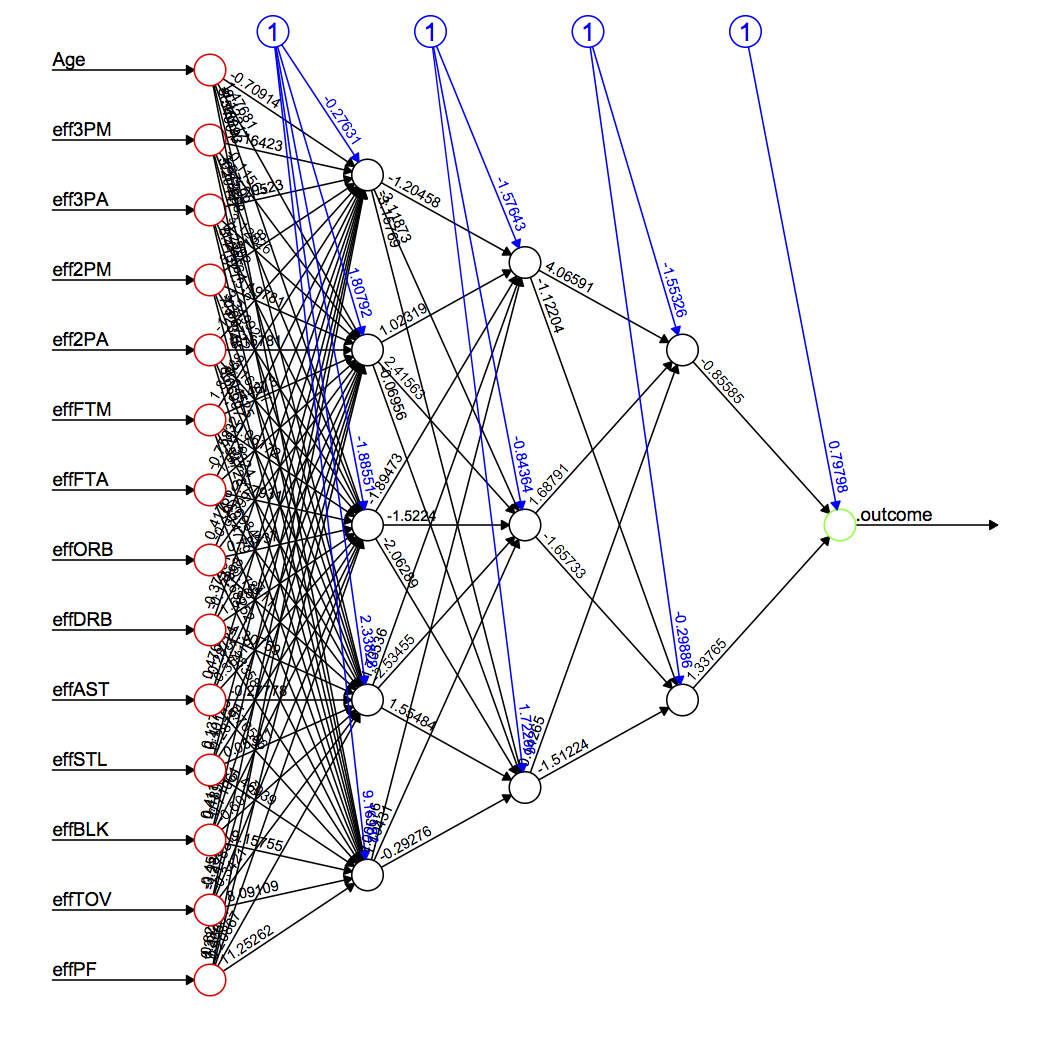
1. The input vectors for the neural network are the weighted average of all stats per team per season. The weights are the effective minutes. The total size of the resulting input vector is 1,063.

1. Remove columns that depend linearly on others: FG, FGA, FG%,3P%,2P%,FT%,effFG%, effPTS and scale the data on [0,1] for easier convergence of backpropagation algorithm[[5]](#footnote-5).
2. I use a split of 75-25% for the training-testing samples. Used 10-fold cross-validation with 10 repetitions (leave one out).
3. I train a regression neural network with 3 layers using the neuralnet[[6]](#footnote-6) R package under default parameters defined by the caret[[7]](#footnote-7) package. See Appendix A for details.
   * 1. **Results: Offense**

The best 3 models for points scored (offense) based on R-squared. Selected on top: (5-3-2)



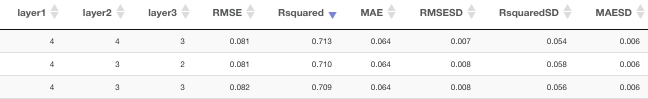
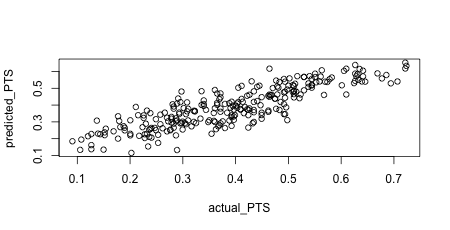
Fitted vs. actual on testing data (25% = 266 observations). Scaled to [0,1]



The Network. Visit the Model tab on the R Dashboard for details on weights connecting each of the nodes.

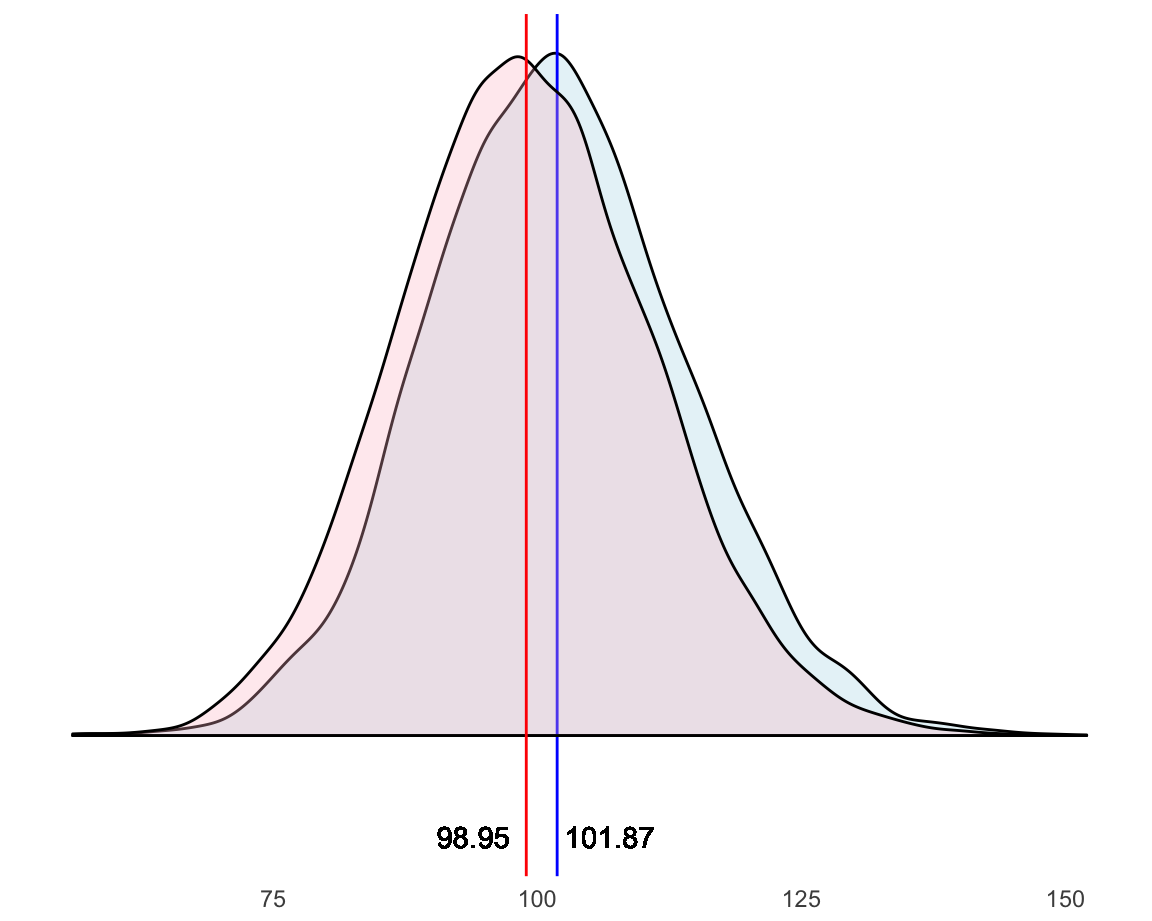
* + 1. **Results: Defense**

The best 3 models for points scored (offense) based on R-squared. Selected top: (4-4-3)



Fitted vs. actual on testing data (25% = 266 observations). Scaled to [0,1]. Clearly not as good a fit as the offensive model but still good enough.

* 1. **Probability model**

Now that I have a way to estimate team’s offensive and defensive powers, I can use them as the estimated mean parameters of a Normal probability model. 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 the red density to away teams. The sample average is 100.41 with a sample standard deviation of 12.16. Now, the variance is pretty consistent across the sample (12.13 for home teams and 12.0 for away teams) and for simplicity I take the overall sample standard deviation as the estimation for the standard deviation of both Offense and Defense Normal distributions:

* 1. **Wins**

So far I can calculate how many points a team score or receive on average, and I know the probability family that models these points. I have all the pieces I need to compute how many wins those offensive and defensive powers award. Suppose teamA plays against teamB and suppose teamA is the home team. I know how many points teamA scores on average, call it: ptsA. Empirically we saw in 3.2. how home teams score 1.46 more points on average. Call it: home\_court\_coeff. Next, I need to plug in teamB’s 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. This will give me an astronomical number of points scored for a basketball game, which will be cancelled out when I calculate teamB’s offense. However, in order to be able to simulate realistic game scores I will subtract the overall average points (100.41) from that number. So, here is how to calculate a teamA’s points adjusted by home court and rival:

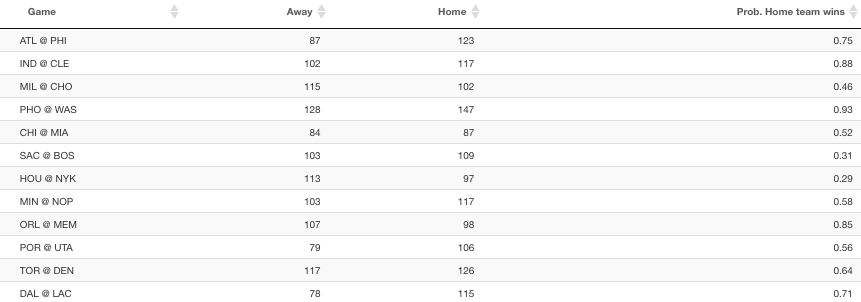
And for teamB as the away team:

Now I can plug these 2 measures as the estimated means of Normal distributions with standard deviation 12.16 and can calculate the probability of teamA beating teamB:

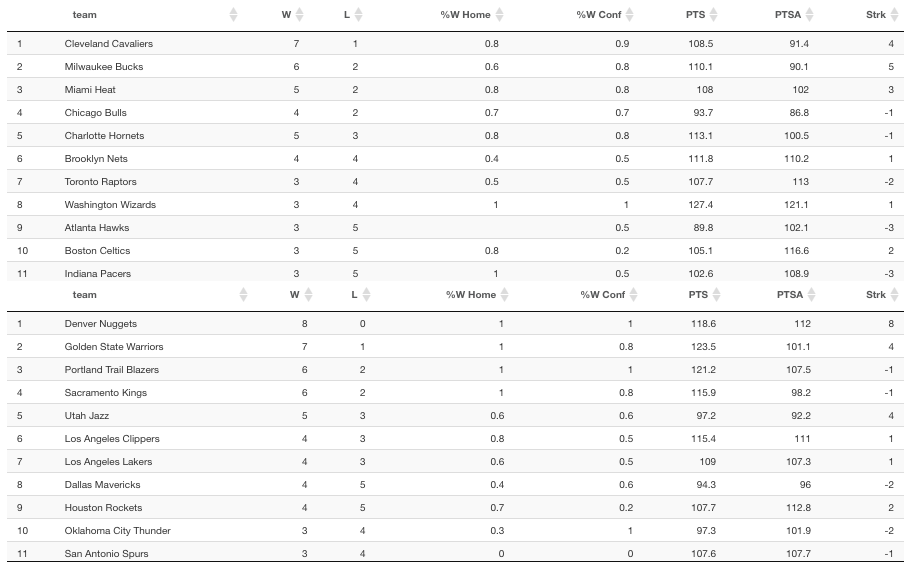
Let and be random variables and P a probability measure defined on the space of all possible events involving teams points. Then:

P(teamA beats teamB)

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. Here’s an extract from the R dashboard showing probabilities and simulated outcomes for each game:



I can easily then simulate an entire season and compute standings at any given point. 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:

1. Similar players by age: To predict how skills of players evolve with age
2. Similar players historically: To see players’ skills evolution over time
3. 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. However, 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: Predict players’ skills and their usage or minutes share.

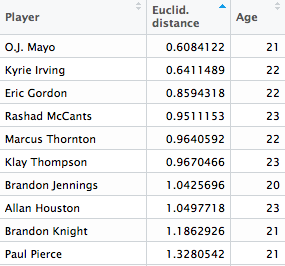
At the beginning of a new season we find 4 types of players:

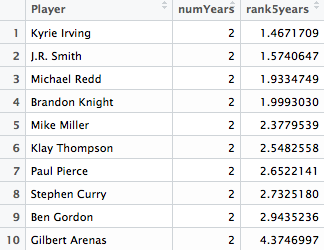
1. Returning NBA players who played in the previous season
2. Returning NBA players who didn’t play in the previous season
3. Rookies drafted from American schools
4. Rookies from international teams

For each of these I will use a different approach mostly determined by the disparity of data sources. What follows is a brief account of the methodology, for a more detailed view and the R code please see Appendix A.2.

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

The algorithm runs through all players from last season and calculates the closest 10 players overall for the past 5 years of his career: For each of the past 5 years of the player’s career I rank the closest players of the same age on that particular year according to Euclidean distance on the 2-D space defined by the t-SNE map of players of that age. I put them all together and rank them according to the number of times sum of the distances. This is inspired by FiveThirtyEight’s CARMELO[[8]](#footnote-8) and it’s better explained through an example:

* 1. Take Bradley Beal from the Washington Wizards and take his last 5 years of NBA stats (from 19 to 23 years old).
  2. Calculate closest players to Bradley Beal by age. That is, calculate the t-SNE 2D map for all the 23 year old players and keep the top 20 for each age. The output is a 100 record file like this:
  3. For each of the players calculate the average Euclidean distance and the number of appearances in the list. The more appearances and the lowest distance the higher in the rank. Here’s the top 10:



* 1. Bradley Beal will be 24 during this new regular season so for each player i in this list and for each of their effStats calculate the variation when going from 23 to 24 years old:

And then the median of all 10 players variations:

Finally, update Bradley Beal’s stats according to this variation:

* 1. The only constraint to the above approach is that players younger than 20 or older than 39 are so scarce that their t-SNE maps don’t really make sense so I assign players the latter to the class of 20 year olds and the former to the 39 year olds.

Sometimes a player didn’t record stats for last season, he played in another league or was injured. I use his stats in his last season as a player as their predicted stats.

For players with no previous experience in the NBA, 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.

* + 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:
   1. 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:
   2. 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"

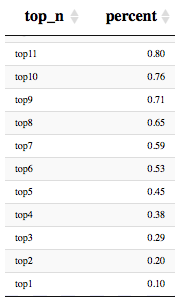
current\_rosters[which(current\_rosters$Player == "Nene"),]$Player <- "Nene Hilario"

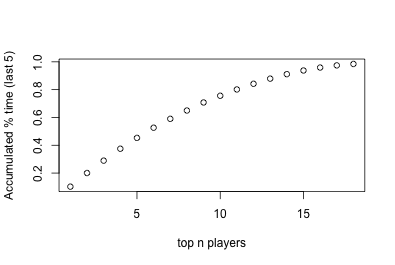
current\_rosters[which(current\_rosters$Player == "Taurean Prince"),]$Player <- "Taurean Waller-Prince"

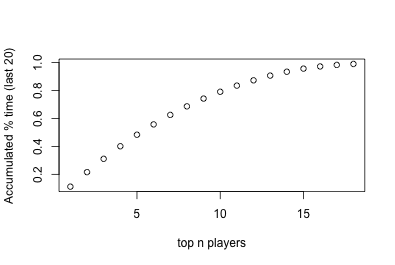
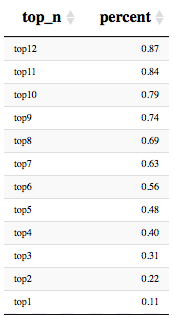
current\_rosters[which(current\_rosters$Player == "Tim Hardaway"),]$Player <- "Tim Hardaway 2"

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**

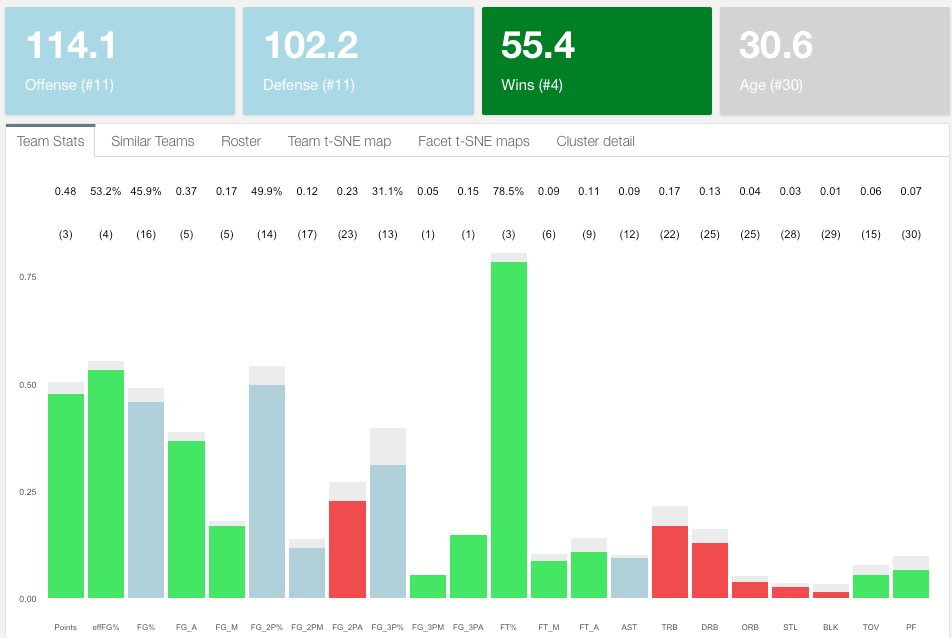
Once I defined the probability model and the inputs, i.e., players’ stats and share of playing time, I can compute expected wins by running a simulation of the regular season using win probabilities. Interestingly, I can also treat players as if they were teams, that is, I can evaluate the offensive 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.

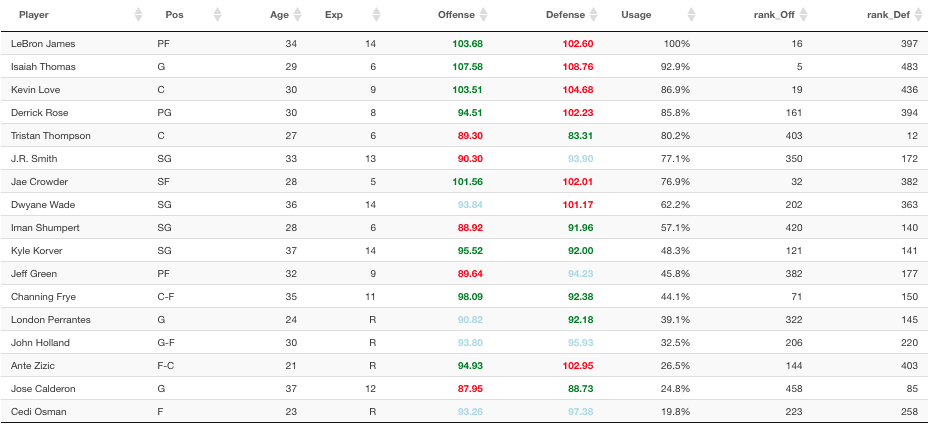
* + 1. **Team stats**

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

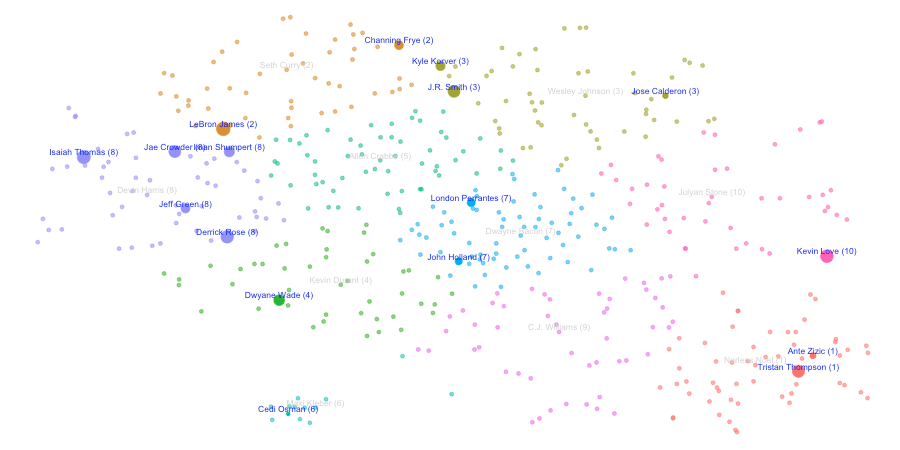


* + 1. **Roster stats**

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.

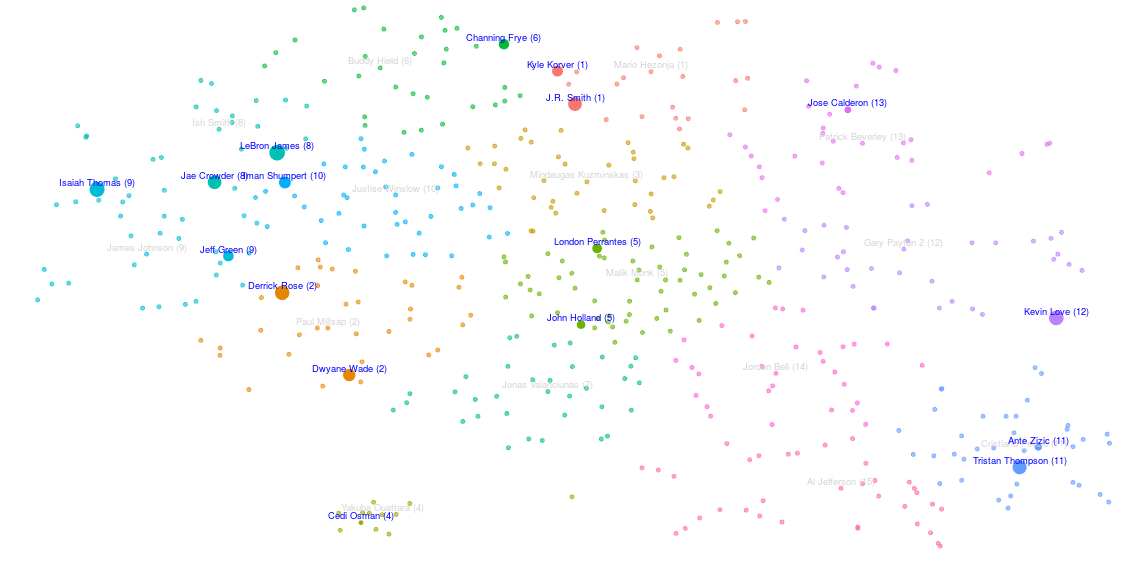
****

* + 1. **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 and size of balloons to player usage

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.

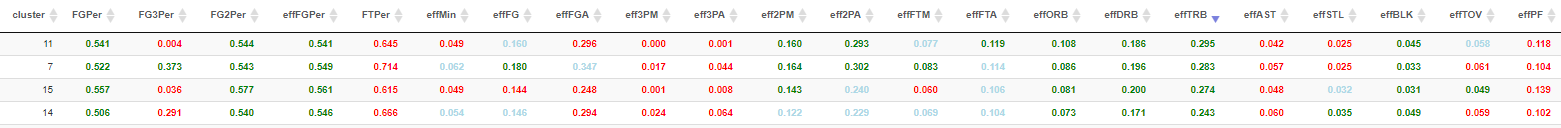




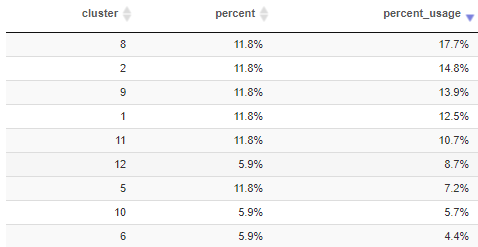
In my view a higher number of clusters will possibly represent the player typology better. See now for 15 clusters how LeBron is no longer associated with Fry but now with Jae Crowder.

Still not what one would expect. An interesting analysis would be to identify weak areas in a team and the type of player in the market that would fill that gap. I already know Cleveland’s weakness is in the rebounding, blocking and stealing areas on the defensive end but also needs to improve field gold percentages on the offensive end. If we look at the 15-cluster k-means detail, these are the cluster with better of those:

Clusters 11, 7, 15 and 14 fit the bill perfectly as almost all of those categories are green:



Now I look at how Cleveland players are distributed across clusters in headcount and by usage which is more important, as we saw, to estimate wins:



Except for cluster 11, filled by Tristan Thompson and minimally by Ante Zizic, the rest of the clusters are not represented. I will go into more detail on how to look for players that fit a team’s needs. For now, I will move on to team similarities as an alternative way of assessing the potential of a team.

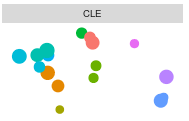
* + 1. **Team similarity**

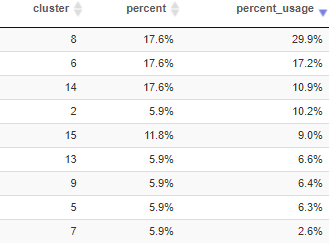
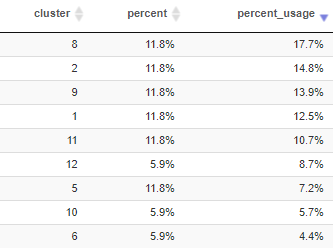
When looking at teams’ rosters and structure there is always a reference, the gold standard if you like. Arguably, in 2017, this is the Golden State Warriors and my model seems to back this up:



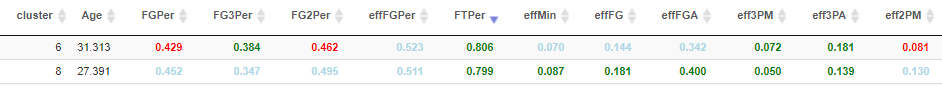
As expected it’s green across the board and they’re number one in wins. They’re not perfect and they probably should improve on the free throw area but not much else. I would argue whatever shape they show on the t-SNE map should be close to an optimal distribution in the 2-D space. Here’s Golden State’s distribution vs the rest of the teams colored by 15 K-means clusters:

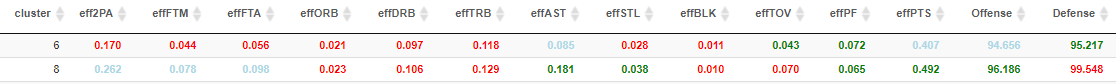
A closer look at GSW vs Cleveland:





Totally different shapes, both winning teams but clearly GSW has been more successful lately. They seem to concentrate almost 50% of their usage on clusters 8 and 6 against barely 20% for Cleveland. I can look at what defines those clusters:



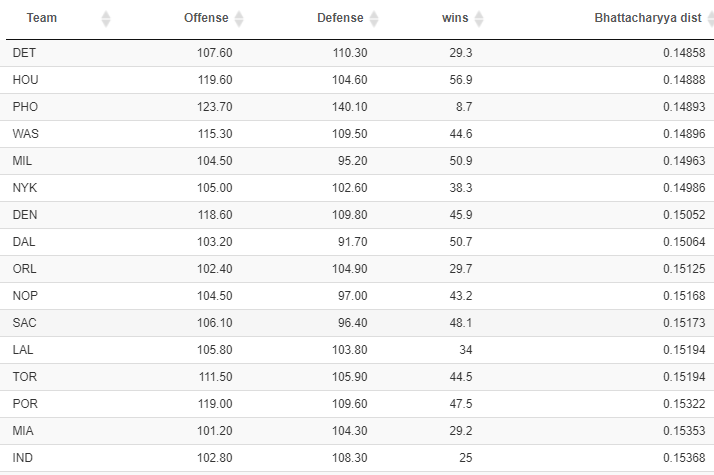


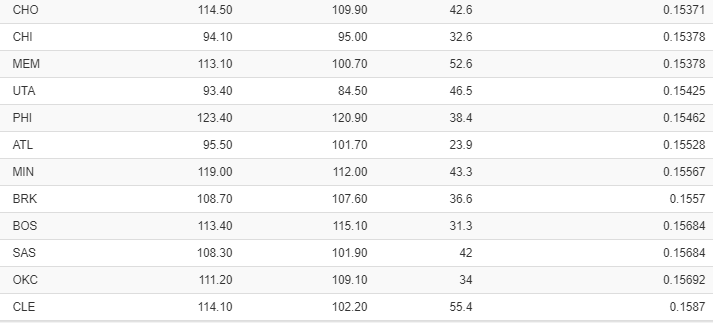
Cluster 8 corresponds to heavily offensive players, high in scoring and field goal percentage and playing heavy minutes. Unsurprisingly Steph Curry, Klay Thompson and Draymond Green belong to it, or LeBron and Jae Crowder in Cleveland. On the other hand, cluster 6 is also characterized by 3-point shooters who play fewer minutes, more experienced players (Age over 31 on average) with good defensive skills. This is the realm of Shaun Livingston, David West or Nick Young versus Channing Fry in Cleveland. This analysis could continue in many different ways and it’s a lot of fun exploring the data.

It’s really hard to measure the dissimilarity of shapes by just looking at the 2-D maps so I wanted to give an analytical measure. For this purpose I use a distance measure based on the Bhattacharyya distance to compare probability distributions.

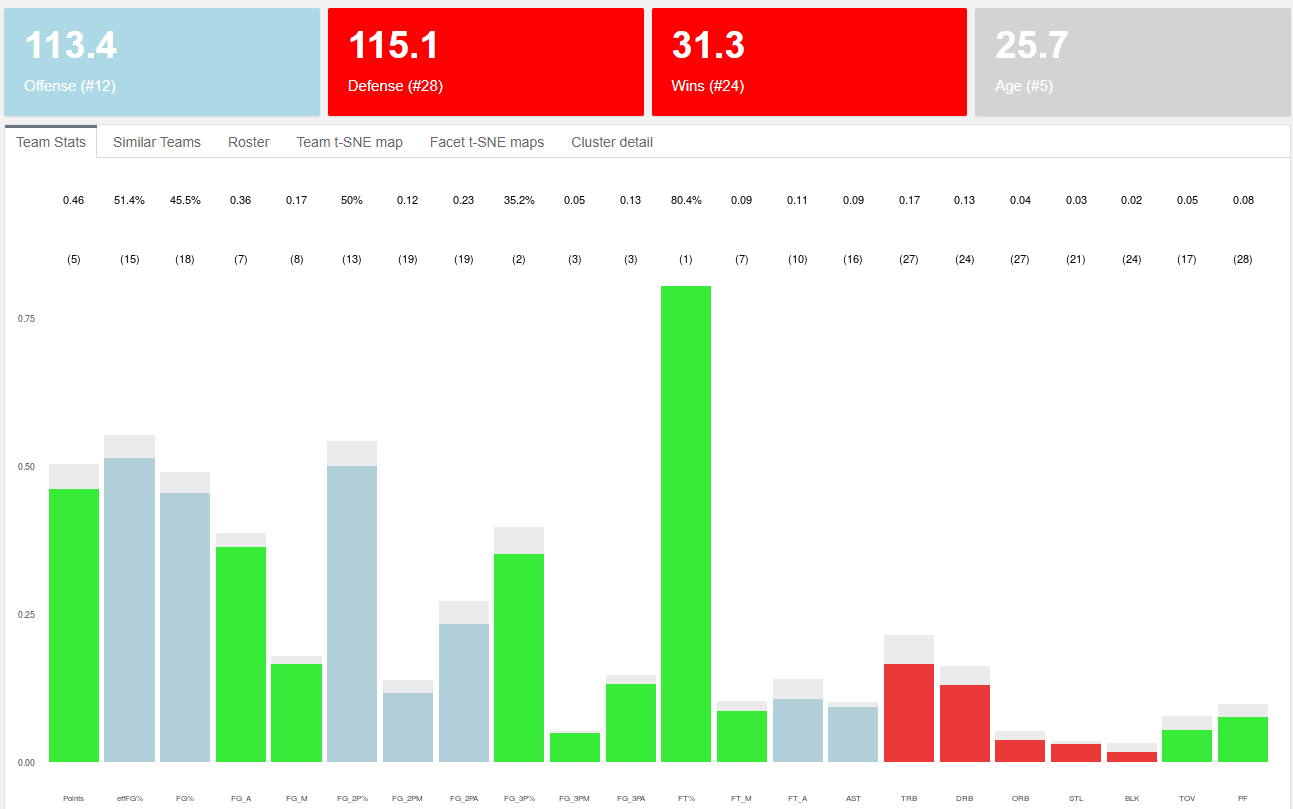
Formula goes here

A visual exploration tells me that, for instance, San Antonio or the Clippers have a very different shape from Golden State. The analytical results confirm this. Besides, Cleveland is the furthest apart team which doesn’t mean much in terms of wins, only that they have 2 different winning styles. Actually, one of the closest teams to Golden State is Phoenix which my model predicts a very low number of wins.





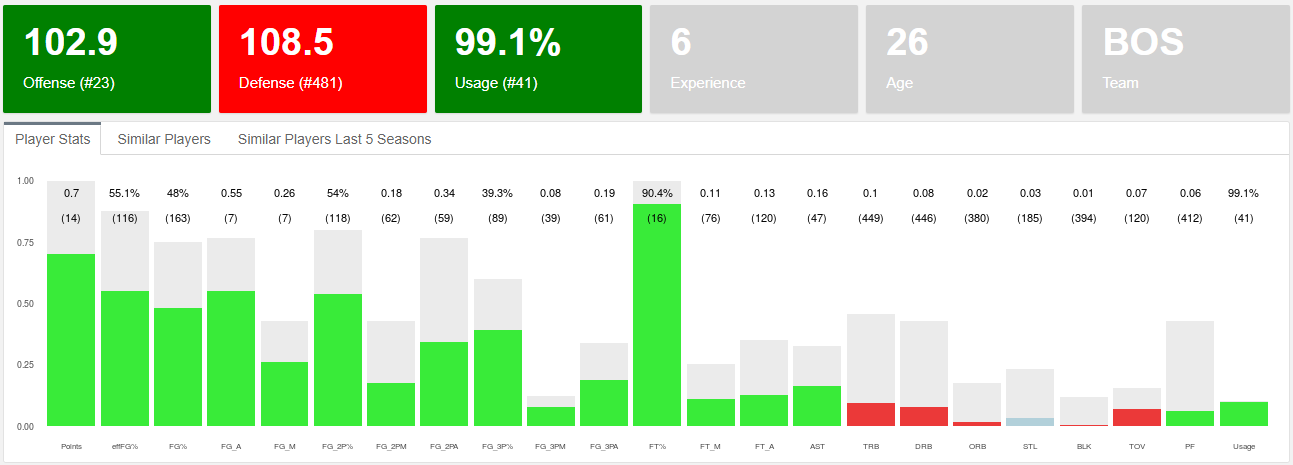
The final predicted wins for a team depend on many factors, one being the accuracy of player predictions, which is particularly hard the younger the roster is (like Boston). Sometimes it’s also a question of how to optimally distribute minutes of play. Let’s take a look at Boston:

****On paper, Boston looks pretty good across the board. Not many reds, very consistent overall. Looking at just the individual team stats one would think they should be expected to win more games than a 31.3. Before jumping to conclusions and declare Boston’s coach as the one of the best ever (he might actually be in the conversation), let’s look at the minute share in Boston’s roster:



Boston’s offense ranks ok at #12 but their defense is the problem. According to the model, they are the third worst defense in the league. Looking at the roster, their best 3 defensive players: Allen, Baynes and Larkin are at or near the bottom in usage. This is a clear case in which a redistribution of minutes would possible benefit a team’s expected wins. I will actually test this hypothesis in the Trade section of the paper.

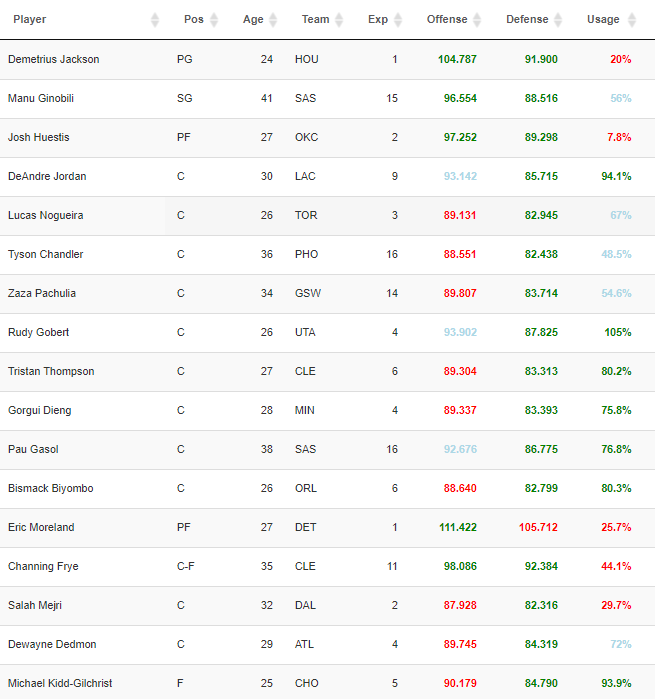
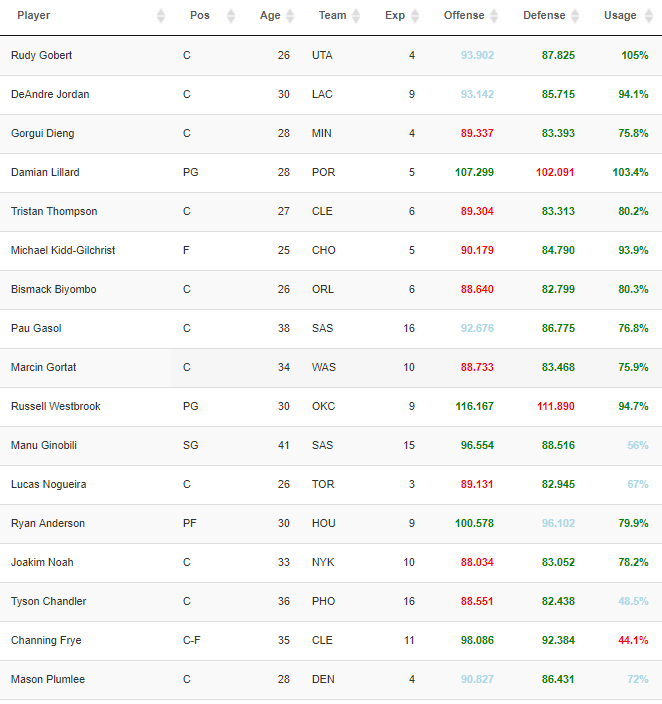
* 1. **The Players**

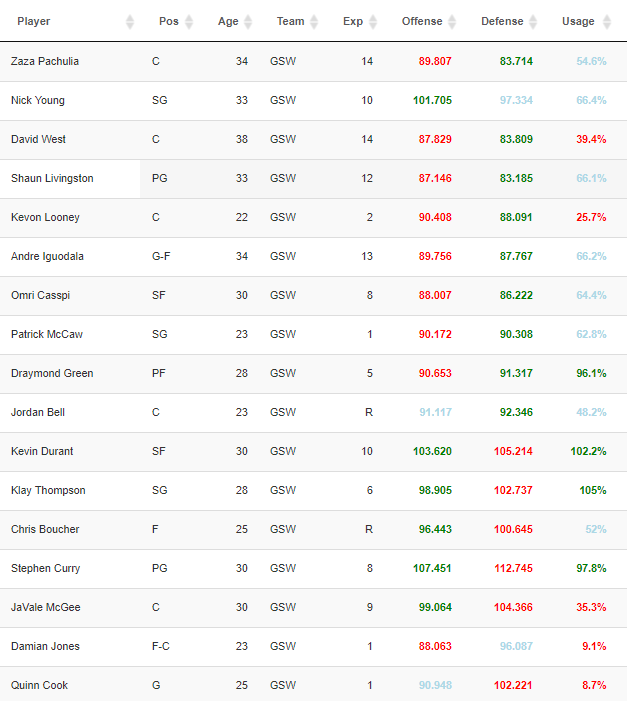
Similarly to the analysis we can do on teams, we can also perform on players. As discussed above, I can apply the Offense and Defense neural network models to players under the assumption that they are teams in which all players are exactly the same. Imagine a team with 5 Kyrie Irving and a bench full of Kyrie Irvings. Well, let’s look at what the model would say about it:

A team full of Kyrie Irvings competing against regular NBA teams would be pretty good offensively but will struggle defensively as nobody would be able to protect the rim, grab enough rebounds, etc. That’s my interpretation of what the model shows.

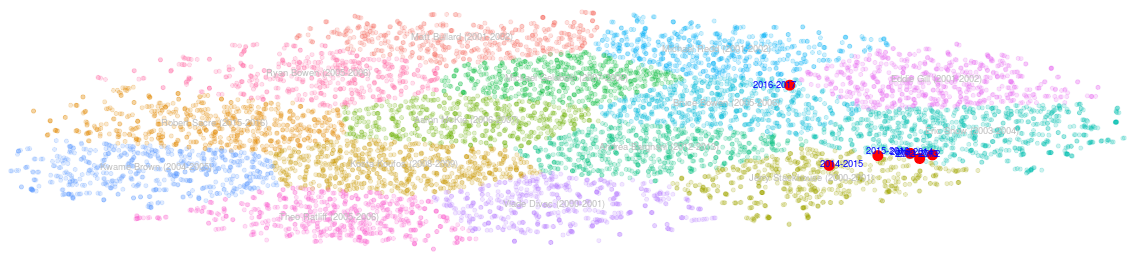
* + 1. **Player plus minus**

This leads me to define what I call a player’s plus-minus, nothing to do with the Box plus minus used on regular stat sheets. I define plus-minus as the Offense minus the Defense, which allows me to rank the best 2-way players and identify where potential improvements in wins can come from for each team. In addition, I also compute the adjusted plus-minus which adjust for the usage or minute share of each player. Here are the top 15 ranked players using both plus-minus (left table) and adjusted plus-minus (right table):



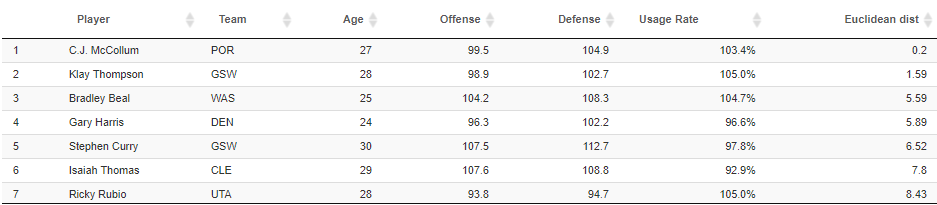
Demetrius Jackson is the most valuable 2-way player according to the model, followed by Manu Ginobili and Josh Huestis. Both Jackson and Huestis have very low predicted usage in their teams so boosting their minutes of play would result in more wins for Houston and OKC, according to my model. When we look at adjusted plus-minus, it’s defensive giants Gobert and DeAndre Jordan on top along with Tristan Thompson, or Bismarck Biyombo. But also small players like Damian Lillard or Russell Westbrook. GSW however is missing from the top 15 except for Zaza Pachulia. Why is this? A closer inspection tells me that GSW are actually the team with the most positive plus-minus in the league (7). And those who are negative, happen to be extremely good offensively, a lethal combination.

* + 1. **Player evolution**

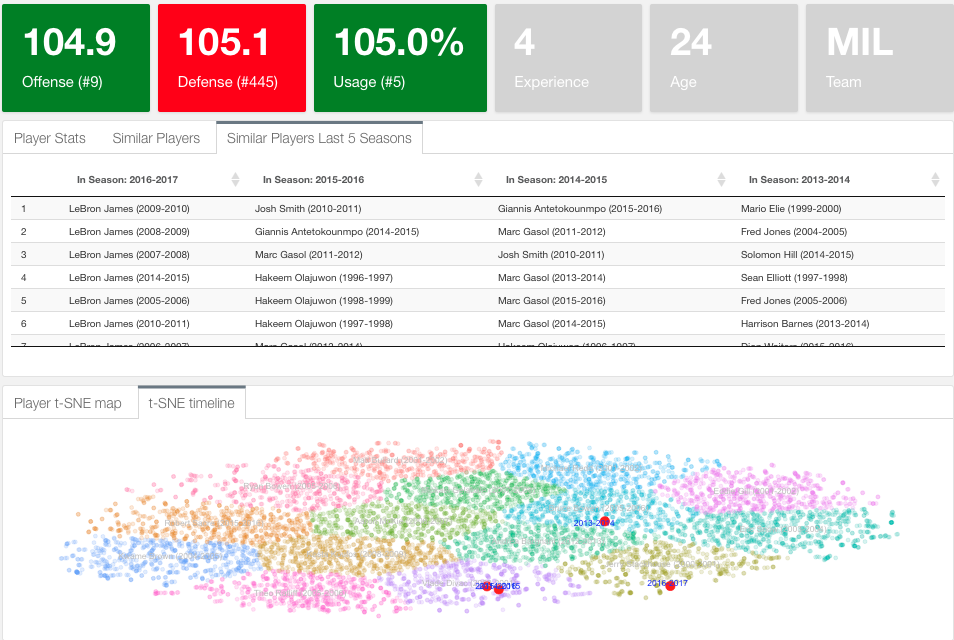
But it would be premature to evaluate players by their plus-minus, especially those with little experience as their numbers respond to a smaller sample size. However, for those with a little more experience, we can look at the consistency of their game for a more robust player evaluation. I do that by computing a t-SNE model with all players for the last 20 seasons. The result is a 2-D cloud of points similar to the previous one with the advantage of seeing how players evolve over time. Let’s look at Kyrie Irving once again:

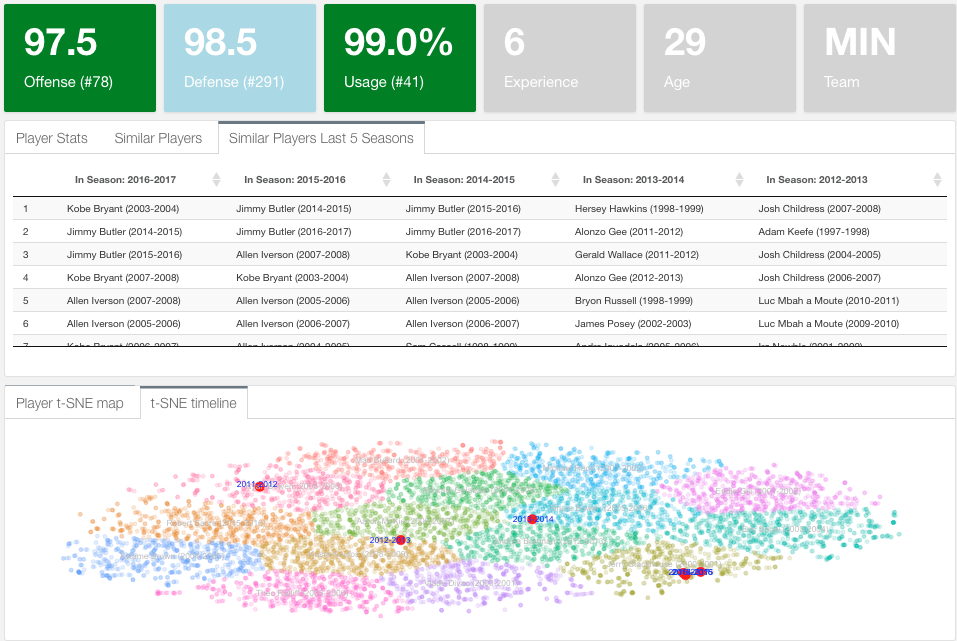
His stats look pretty consistent throughout his career except for last season in which he clearly departs from the zone he used to dwell. I will look at the composition of clusters later on (not yet computed this bit). We can think of different reasons why this sudden change which definitely would have been a lot harder to spot by simply looking at his stat sheet, but let’s try to see if from the perspective of who were Kyrie’s neighbors for the past 5 years:

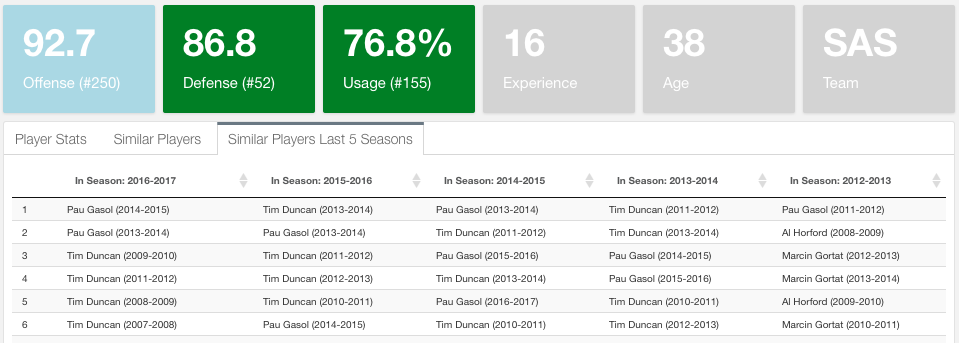
Seems like a drastic change in neighbors from Clyde Drexler, Darrel Armstrong or Hardaway to C.J. McCollum, Bradley Beal or Leandro Barbosa. This seems to reinforce the opinion of Kyrie becoming more like a sidekick type of player next to LeBron and wanting out to be back closer to the neighborhood of the greats.

Unsurprisingly, after applying the model and taking the t-SNE map of current predicted season stats, these are the predicted most similar players to Kyrie:

Let’s now confirm analytically the evolution of two players who we know have taken different paths: Giannis and Derrick Rose:

Giannis path has been spectacularly improving year by year. From Marc Gasol to Olajuwon, and last year to being the closest to LeBron James. Another interesting evolution happened to Jimmy Butler who basically moved from not stellar stats to being a mix between Allen Iverson and Kobe we see in today’s game

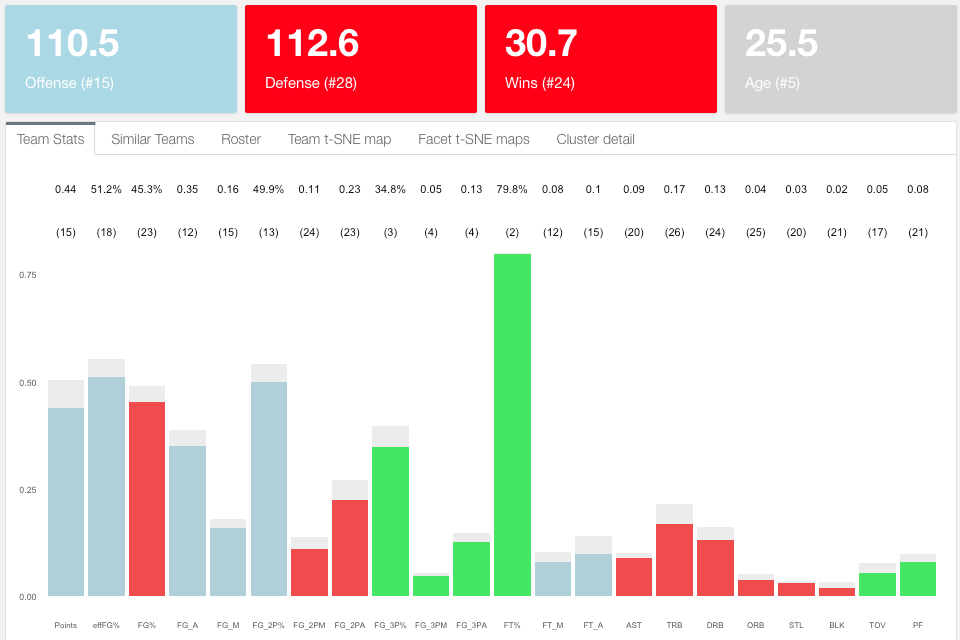
I will finalize this section with a real case scenario. When the San Antonio Spurs had to explore the market for a replacement for Tim Duncan they didn’t use my model because it didn’t exist yet but probably used a similar one. Just check out similar players to Pau Gasol:

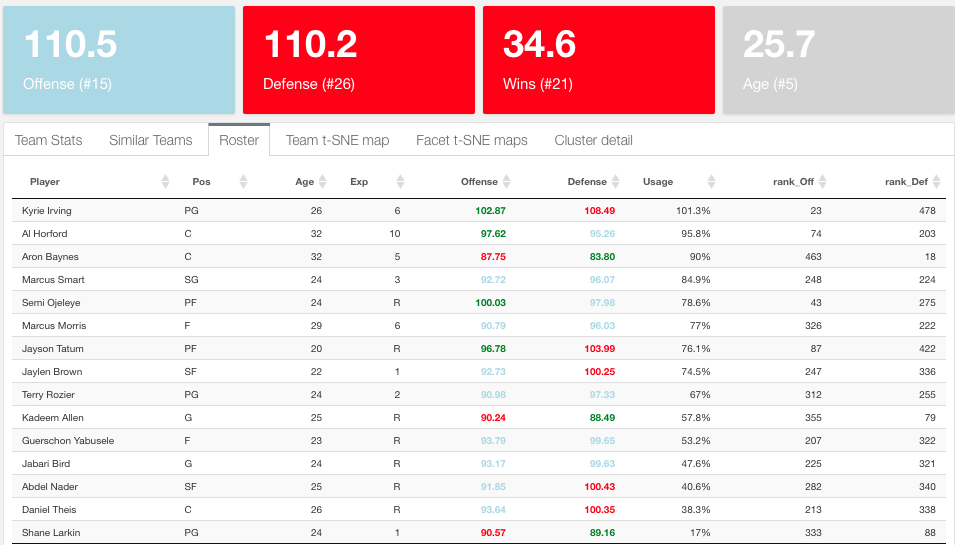


1. **The Trades**

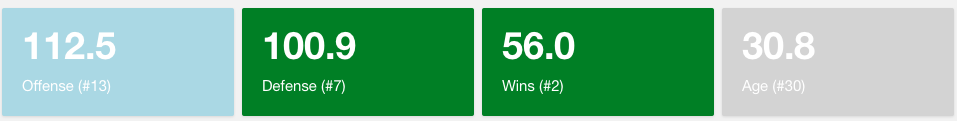
The results section set the tone of the type of analysis that can be made with the described model. The next step if to test some of the hypothesis presented and the robustness of the model. I will start by defining what is a trade. I consider a trade any change in a team’s roster that affects their Offense and Defense powers. In particular, a trade can be a traditional trade of 2 or more players between 2 or more teams. Can also be a player leaving a team and the NBA (due to injury, retirement or traded to another league). Finally, it can also be a player who increases his usage within his team.

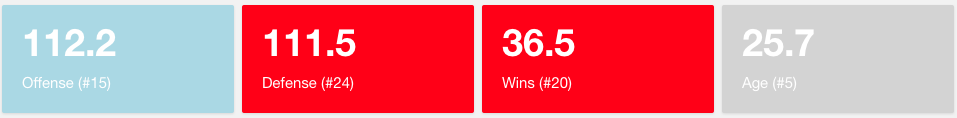
The key aspect of a trade is that when the trade happens, the whole league stats and team rosters are re-calculated as a change in a team’s Offensive and Defensive power will affect not only their number of wins but also every other team’s wins. Now let’s get to the examples:

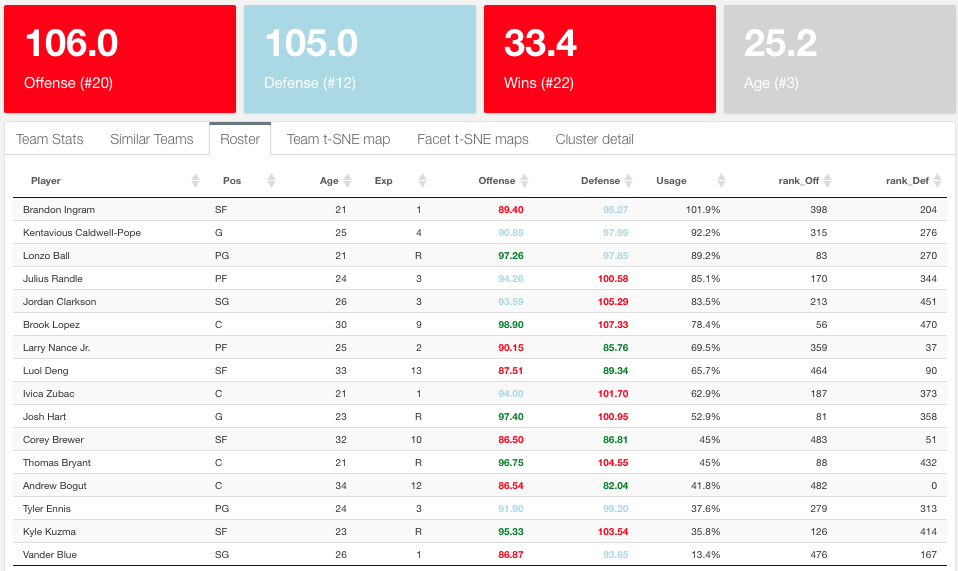
Let’s focus on Boston again and let’s see how removing Gordon Hayward from the roster affects the predicted stats for the team.

Boston gets a little worse but not much, Gordon Hayward represents only half a win, which may be surprising. The positive aspect is that the defense improves. Let’s now increase the usage of Aaron Baynes from his current 13.1 minutes per game (taken from effMin\*3936) to 30 minutes per game and see how this further impacts the team:

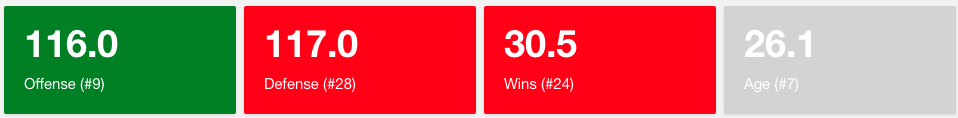
As expected increasing Baynes minutes improves Boston wins by 4. Being careful with tweaking offensive players’ minutes I could follow this pattern to improve Boston enough to be a team to compete for the playoffs. Similarly, I can measure the impact of Isaiah Thomas injury on the Cavs:

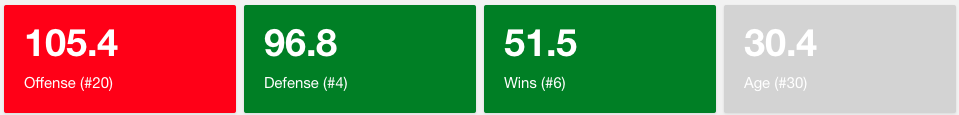


Although there is a drop in offense, the defense gets better as players like Tristan Thompson will play more minutes. Consequently, the number of wins increases as well by 1.5 total wins. To see the ripple effects any trade have for the rest of the teams, below we see how Boston also improves its predicted wins by 1.5:

And, why not, let’s trade LeBron to the current Lakers. This is the Lakers before LeBron:

After acquiring LeBron James the offense improves but the defense needs tweaking just as with Boston. By default, a star player will get you more baskets but someone has to defend.

Finally, what happens with Cleveland after LeBron? It declines significantly, losing 4.5 wins and 7 points of offensive power:



* 1. **Create your own player**

Why stop there? I can create my own player and plug it into any team. This will also allow me to understand how the neural networks pick up the stats and return offense and defense outputs.

• 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 A: R scripts**

**A.1. Neural Network model**

1. write\_playersHist.R returns: playersHist.csv
2. .rename\_PlayerName\_Duplicates.R() from rename\_PlayerName\_Duplicates.R
3. .team\_prepareAll() from prepare\_rosters.R
4. .computeModel\_neuralnet() from neural\_network.R

7. neuralnet default parameters:

neuralnet(formula, data, hidden = 1, threshold = 0.01,

stepmax = 1e+05, rep = 1, startweights = NULL,

learningrate.limit = NULL,

learningrate.factor = list(minus = 0.5, plus = 1.2),

learningrate=NULL, lifesign = "none",

lifesign.step = 1000, algorithm = "rprop+",

err.fct = "sse", act.fct = "logistic",

linear.output = TRUE, exclude = NULL,

constant.weights = NULL, likelihood = FALSE)

**A.2. Predicting Player Skills**

**A.2.1. 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

The algorithm runs through all players from last season and computes: .predictPlayer() (from similarityFunctions.R):

1. 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.
   1. 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():
   1. 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.
   2. 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.
   3. 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:
   1. 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:
   2. 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. https://alberto-sanchez-rodelgo.shinyapps.io/Player\_Dashboard/ [↑](#footnote-ref-1)
2. See for example: https://www.basketball-reference.com/leagues/NBA\_2017\_per\_game.html [↑](#footnote-ref-2)
3. For a more formal justification see: http://andrewgelman.com/2014/02/25/basketball-stats-dont-model-probability-win-model-expected-score-differential/ [↑](#footnote-ref-3)
4. See Appendix A for details on the R code. Also on my Github: [asRodelgo/NBA](https://github.com/asRodelgo/NBA) [↑](#footnote-ref-4)
5. See “Should I standardize the input variables?” in http://www.faqs.org/faqs/ai-faq/neural-nets/part2/ [↑](#footnote-ref-5)
6. https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf [↑](#footnote-ref-6)
7. http://topepo.github.io/caret/index.html [↑](#footnote-ref-7)
8. https://projects.fivethirtyeight.com/carmelo/ [↑](#footnote-ref-8)