Determining Whether a Player Performs Better After Being Traded (2015 NBA Shot Log Data)

Blake Gonthier

8/14/17

Abstract:

Data manipulation was used in Base SAS 9.4 to perform an analysis analyzing the net combined offensive performance and defensive performance of players who were traded during the 2015 NBA season. The results displayed towards the end of the paper showed there was no statistical difference between the performances of the players before they were traded compared to after they were traded. Since this analysis uses data from only one season, inference cannot be made about the NBA in general with respect to this project question. An addendum to this project can be made by scraping all of the data from every NBA season in order to better estimate the population of NBA seasons.

Introduction:

The goal of this paper is to understand whether players who are traded significantly perform better or the same as opposed to performing worse after being traded to a new team. The results from this data analysis can be applied to all of the years the NBA was in existence in order to look for trends of the players that were traded that have characteristics of performing better to aid the decision making of NBA general managers in the future when considering certain types of players that will get what they expected or get better than what they expected.

Base SAS 9.4 was used for the analysis for the project.

Getting Familiar with the Data:

The data set used was provided by Dans Becker and was posted to the public site Kaggle with the following link below:

https://www.kaggle.com/dansbecker/nba-shot-logs

Ended up analyzing each variable and understanding what they mean, how they are represented, and if there are any unusual values.

Variable	Description	Type
Game_ID	ID designated for the game the player played in	ID
Matchup	The date and the teams involved in the game	Nominal String
Location	Home or away games (H/A)	Nominal String
W	Games that were Won or Lost	Nominal String
Final_Margin	Point Differential between teams involved in the games (+/-)	Integer
Shot_Number	Accumulating shot counter for each game (reset after each game)	Integer
Period	Period number of game (1-4 regular game time, 5-7 for extra periods	Integer
Game_Clock	Time remaining in the period in format (min:sec)	String
Shot_Clock	Time remaining until the time required to shoot the ball on a possession	Floating Point Number
Dribbles (# of)	Amount of dribbles taken before the shot was taken (0-32)	Integer
Touch_time		Floating Point Number

-Numbered subscripts are values that are coded and their respective codes are numbered in the appendix.

	How much time passed when the player had a possession before shooting a shot Up to 0.24% of values are negative (possible mistyping?)	
Shot_Dist	The shot distance in feet up to a tenth of foot in measurement preciseness	Floating Point Number
Pts_type	Shot type (2/3) that does not include free throws	Integer
Shot_result	Result of shot taken (made/missed)	Nominal String
Closest_defender	Defender's name that was closest to the player that was taking the shot	Nominal String
Closest_defender_player_id	Player ID # corresponding to the Closest_defender variable	ID
Close_def_dist	Distance in feet that the closest defender was in relation to the shooter	Floating Point Number
Fgm	1-make, 0- miss	Binary
Pts	Points accumulated for the shot (0/2/3)	Integer
Player_name	Player's associated name that took the shot	Nominal String
Player_id	Player ID # corresponding to the Player_name variable	ID

-Numbered subscripts are values that are coded and their respective codes are numbered in the appendix.

Cleaning the Data:

Before modifying the data to gain the information needed for analysis, observations that either had missing data or invalid values needed to be removed from the data set. A PROC FREQ was used to check how many values were missing from the data set.

The SAS System				
The FREQ Procedure				
Number of Variable Levels				
Variable	**Label**	**Levels**	**Missing Levels**	**Nonmissing Levels**
GAME_ID	GAME_ID	904	0	904
MATCHUP	MATCHUP	1808	0	1808
LOCATION	LOCATION	2	0	2
W	W	2	0	2
FINAL_MARGIN	FINAL_MARGIN	88	0	88
SHOT_NUMBER	SHOT_NUMBER	38	0	38
PERIOD	PERIOD	7	0	7
GAME_CLOCK	GAME_CLOCK	719	0	719
SHOT_CLOCK	SHOT_CLOCK	242	1	241
DRIBBLES	DRIBBLES	33	0	33
TOUCH_TIME	TOUCH_TIME	313	0	313
SHOT_DIST	SHOT_DIST	448	0	448
PTS_TYPE	PTS_TYPE	2	0	2
SHOT_RESULT	SHOT_RESULT	2	0	2
CLOSEST_DEFENDER	CLOSEST_DEFENDER	473	0	473
CLOSEST_DEFENDER_PLAYER_ID	CLOSEST_DEFENDER_PLAYER_ID	474	0	474
CLOSE_DEF_DIST	CLOSE_DEF_DIST	299	0	299
FGM	FGM	2	0	2
PTS	PTS	3	0	3
player_name	player_name	281	0	281
player_id	player_id	281	0	281

It is seen that the shot_clock variable is the only variable that is missing in the data set. Checking for missing values using:

```
data shot;
      set tmp1.shot_log;
      if shot_clock =.;
run;
```

```
NOTE: There were 128069 observations read from the data set TMP1.SHOT_LOG.
NOTE: The data set WORK.SHOT has 5567 observations and 21 variables.
NOTE: DATA statement used (Total process time):
      real time           1.73 seconds
      cpu time            1.12 seconds
```

-Numbered subscripts are values that are coded and their respective codes are numbered in the appendix.

The proportion of missing observations from this data set is approximately 4.3%, thus it would be sufficient to remove if needed.

Another important aspect to consider is to detect whether there are incorrect values for certain variables in the data set.

After exploring the data, the following variables had incorrect values:

-Touch_time -> Up to 0.24% of values are negative

```
data shot;
       set tmp1.shot_log;
       if touch_time<0;
run;
```

```
NOTE: There were 128069 observations read from the data set TMP1.SHOT_LOG.
NOTE: The data set WORK.SHOT has 312 observations and 21 variables.
NOTE: DATA statement used (Total process time):
      real time           1.10 seconds
      cpu time            1.09 seconds
```

-Closest_Defender and Closest_Defender_Player_ID do not have the same number of levels. There are 473 levels for the Closest_Defender and there are 474 levels for the Closest_Defender_Player_ID.

In order to solve this issue a corresponding PROC FREQ was used in order to see if one player has more than one Closest_Defender_Player_ID:

Below is the corresponding code:

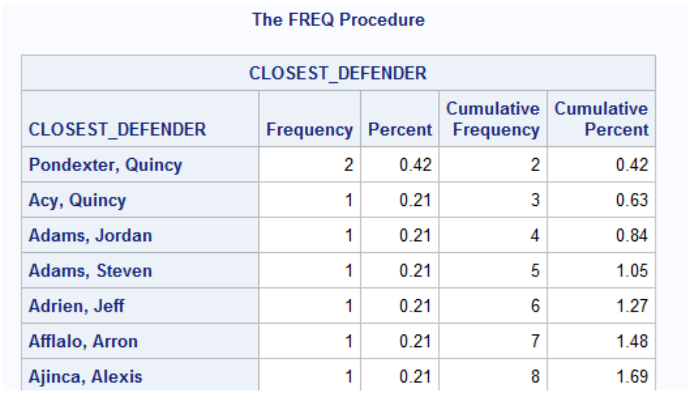
```
proc sort data=closest_defender_player_id out=closest_defender_player_id;
       by closest_defender_player_id;
run;

data check (keep=closest_defender_player_id closest_defender);
 set closest_defender_player_id;
 by closest_defender_player_id;

 if last.closest_defender_player_id;

run;proc freq data=check order=freq;
       tables closest_defender;
run;
```

-Numbered subscripts are values that are coded and their respective codes are numbered in the appendix.



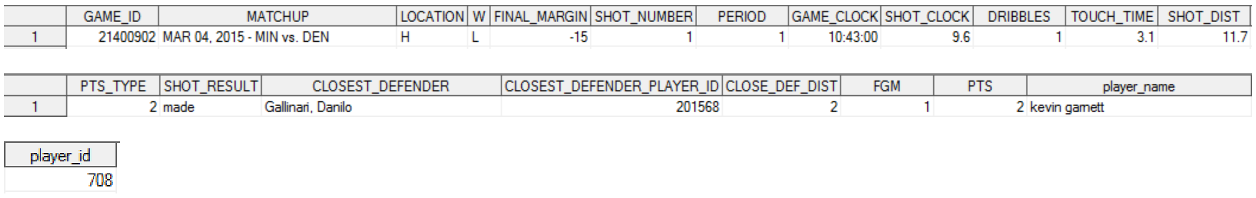
The FREQ Procedure

CLOSEST_DEFENDER				
CLOSEST_DEFENDER	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Pondexter, Quincy	2	0.42	2	0.42
Acy, Quincy	1	0.21	3	0.63
Adams, Jordan	1	0.21	4	0.84
Adams, Steven	1	0.21	5	1.05
Adrien, Jeff	1	0.21	6	1.27
Afflalo, Arron	1	0.21	7	1.48
Ajinca, Alexis	1	0.21	8	1.69

As you can see on the right, Quincy Pondexter was assigned two closest_defender_player_id values. Therefore, in the data set before doing any analysis, Quincy Pondexter needs to be assigned a different closest_defender_player_id that corresponds to his player_id. However, since Quincy Pondexter does not have a player_id value in the shot_log data set after having removed missing values, no other data manipulation is needed. It is worth noting that the results of this project can be biased due to the missing information, but cannot be fixed.

After fixing these issues, the current data as it is needs to be modified in to get some information to answer the main issue if players who were traded during the season ended up playing better. Since the data used is restricted, some of the advanced statistics normally used to evaluate players' performances will not be used, but a new scoring system will be used and will be described in detail later in the paper.

Below is an example observation from the data set:



	GAME_ID	MATCHUP	LOCATION	W	FINAL_MARGIN	SHOT_NUMBER	PERIOD	GAME_CLOCK	SHOT_CLOCK	DRIBBLES	TOUCH_TIME	SHOT_DIST
1	21400902	MAR 04, 2015 - MIN vs. DEN	H	L	-15	1	1	10:43:00	9.6	1	3.1	11.7

	PTS_TYPE	SHOT_RESULT	CLOSEST_DEFENDER	CLOSEST_DEFENDER_PLAYER_ID	CLOSE_DEF_DIST	FGM	PTS	player_name
1	2	made	Gallinari, Danilo	201568	2	1	2	kevin garnett

player_id
708

As it stands, the list of the variables needed from the observations in the dataset for the analysis and their corresponding uses are below:

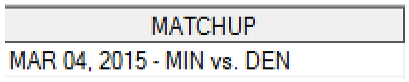
Variable Used	Purpose
Game_ID	Determining the game to game stats for the shooter
Matchup	Needed to extract the team in which the shooter currently plays for
Dribbles	Needed in order to determine the player performance score offensively
Shot_Dist	Needed in order to eliminate shots that are not determinant by the player's discretion which would skew results (end of period heaves that range from 30 feet and out), and to identify dunks/layups
Closest_Defender, Closest_Defender_ID	Needed to identify the player defending
Closest_Defender_Dist	Needed to determine a player's defensive ability
FGM	Needed to determine whether a shot was made or not
Player_Name, Player_ID	Needed to identify the player shooting

-Numbered subscripts are values that are coded and their respective codes are numbered in the appendix.

From the variables above, transformations were needed for better usability of the data. The variables that needed to be transformed were Closest_Defender, and Player_Name into a format like:

Kevin Garnett[1]

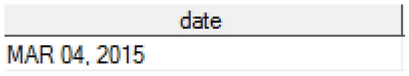
Also, from the matchup variable, the player's shooting team can be extracted, as well along with the date.



MATCHUP
MAR 04, 2015 - MIN vs. DEN

The first team's abbreviation after the date represents the shooter's current team. From this, the scan function in SAS can be used in order to extract this information in order to be used to determine when a particular player switched teams during the season[2] (code in appendix).

Using the extraction date in the form from the figure to the right, a SAS date referencing the number of days since January 1, 1960 needs to be made to easily sort the game dates for coding mentioned in the analysis portion of the paper[3]



date
MAR 04, 2015

Lastly from the variables above, new variables need to be made for the analysis for each observation and are as follows below:

Contested_shot[4] - Binary variable with a 1 representing a shot that is contested (defined as closest_defender_dist < 5 feet)

Perform_o- Calculated offensive performance metric with the following scoring (explanation of weighting will be discussed in the analysis portion):

Perform_d-Calculated defensive performance metric with the following scoring (explanation of weighting will be discussed in the analysis portion):

Overall_metric- Is the sum of Perform_o and Perform_d

Game_avg- Is the per game accumulation average of the overall_metric (explanation will be discussed later in analysis section)

-Numbered subscripts are values that are coded and their respective codes are numbered in the appendix.

Analysis:

In deciding what metric to use, it was necessary to analyze which variables would be valuable in determining a player's overall performance. There are several aspects to take into account, such as the time of year, the team that is being played, the shot type (2 or 3 point shot), whether the shot is contested or not, the defender that is guarding the shooter, the amount of time left on the shot clock, etc. All of these factors play a role in completely defining the conditions of the shooter in order to determine which players perform the best. In order to simplify the analysis, the only variables used to determine the performance of a player are as follows:

Closest_Defender_Dist

Contested_Shot

Dribbles

FGM

Shot_Dist

These variables can be used in a variety of ways in order to describe the skill of a player. The performance metrics each describe all possible situations of a player who shoots the ball in a normal game situation, which is defined as an in-game situation where the shot clock or the game clock does not affect the player's basketball play and only player skill is represented.

Perform_o_5-

In the NBA there are few players who are able create their own shots, which means that these players do not need the help of their teammates in order to make a shot. These players on the offensive side of the ball are rare to find, and since the NBA has evolved to become a scorer's league, these metrics are significantly higher than the defensive metrics. Some examples of players like this include Kyrie Irving, Chris Paul, and James Harden. Thus, as an innateness to my rating system, these players potentially are more likely to score higher as opposed to "big men" (players who are not primarily ball handlers).

a) If a shot is a made with the uncontested shots with many dribbles situation:

Made +5.5: In this situation, the player creates space on his own using his dribbling skill, while also completing the shot, which also reflects his shooting skill.

Missed -0.75: If the shot is missed it is, then he is missing an open shot (where no one is within 5 feet of him) and should be penalized more than a contested shot.

b) If a shot is made/missed with the contested shots low number of dribbles situation:

> Made +4: In this situation, the player shoots but does not attempt to use his dribbling skill. However, in this situation, the player takes a poor shot thus reflecting poor shot selection, but nonetheless shows exceptional skill by making a contested shot.
>
> Missed -0.50: Since this shot is a contested shot, a miss should not be counted against the player as much.

c) If a shot is made/missed with the contested shots with many dribbles situation:

> Made +3: For this situation, the player attempts to use his dribbling ability, but is unable to create distance. However, the player still makes the shot and is rewarded for it, but not as much as situation B.
>
> Missed -0.50: Same reasoning for miss as situation B.

d) If a shot is made/missed with the uncontested shots with low number of dribbles that are not layups or dunks, which are defined as shots whose shot distance is less than 3.5 feet (half of length of the average NBA wingspan)

> Made +2: This situation takes into account of the teamwork aspect. Thus, the only skill that is displayed is the player's shooting and nothing else. Since the player did not create the play, the player is not rewarded as many points for the offensive performance metric.
>
> Missed -0.75: Same reasoning for miss as situation A.

e) If a shot is made/missed with the uncontested shots with low number of dribbles that are layups or dunks

> Made +1: This make situation is the easiest shot in the game, which is an assisted layup/dunk. The player's skill is not reflected at all except for being at the right place at the right time, but the player is still rewarded by scoring a basket.
>
> Missed -2: The player is heavily penalized for missing a shot in this situation, since these are plays that a standard high school basketball player should make all the time

Even though such a high weight was applied to the offensive side of basketball, defensive metrics are still useful in evaluating a player's skill. Nevertheless, since the shooting skills of basketball players nowadays are better than in past years and sometimes nothing can be done to make a player miss a shot, the metrics for these stats are weighted less with respect to the offensive metric.

Perform_d$_6$- Calculated defensive performance metric with the following scoring (explanation of weighting will be discussed in the analysis portion):

a) If a shot is made/missed with the defender contesting a shot:

Made -0.5: In this situation, the defender puts himself in a good position to make the shooter more likely to miss a shot, but nonetheless fails and is penalized.

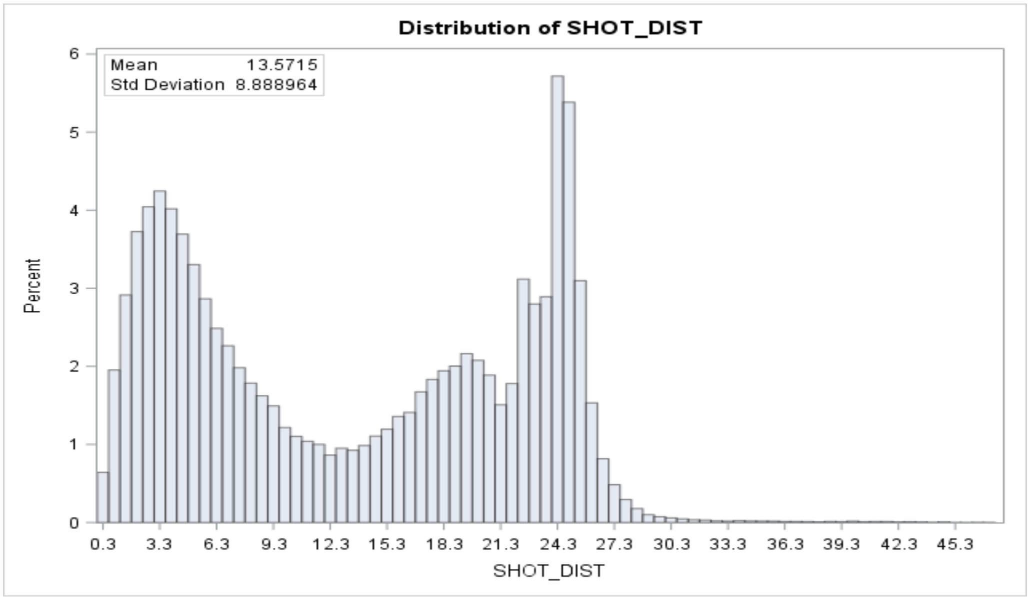
Missed +0.5: The defender is able to put himself in a good position to where he impairs the shooter enough to miss the shot and defensive metric is rewarded.

b) If a shot is made/missed with the closest defender not contesting a shot.

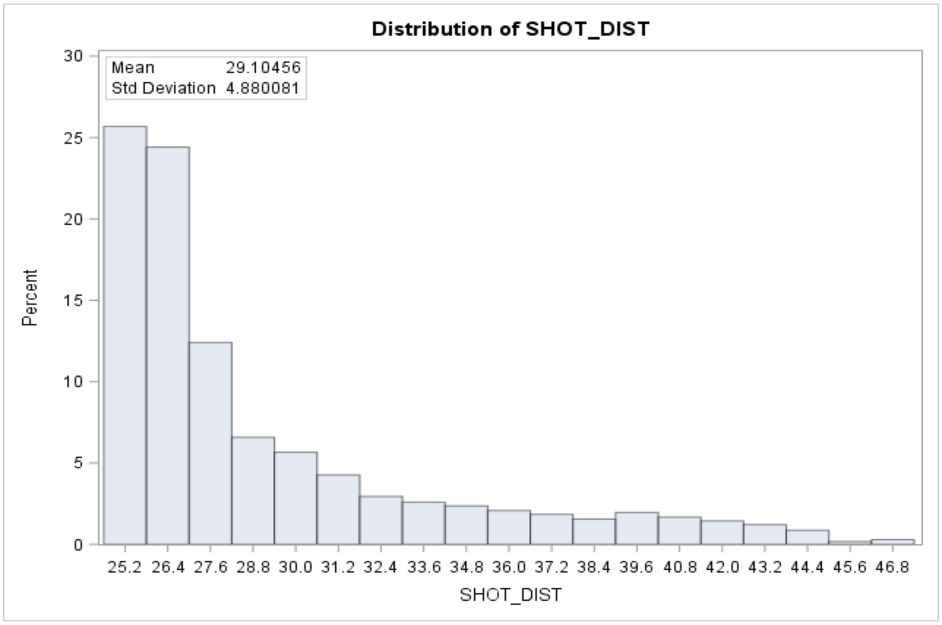
Made or missed -1: The defender does not do his job in staying with his man or has a communication mistake with one of his teammates and is penalized whether the shooter misses or makes the shot since the shot result was not dependent upon the defender's defense.

Before running the code to score all of these metrics, further modifications need to be made in order to make sure that the resulting scores are accurately reflecting the players' abilities.

Need to take out long shots that are heaved or thrown at the goal due to extraneous circumstances, such as the end of a period of a game. Determining what baseline to use for these kinds of shots, PROC UNIVARIATE was used in SAS to produce a histogram of the distribution of all the shot distances in the data set.

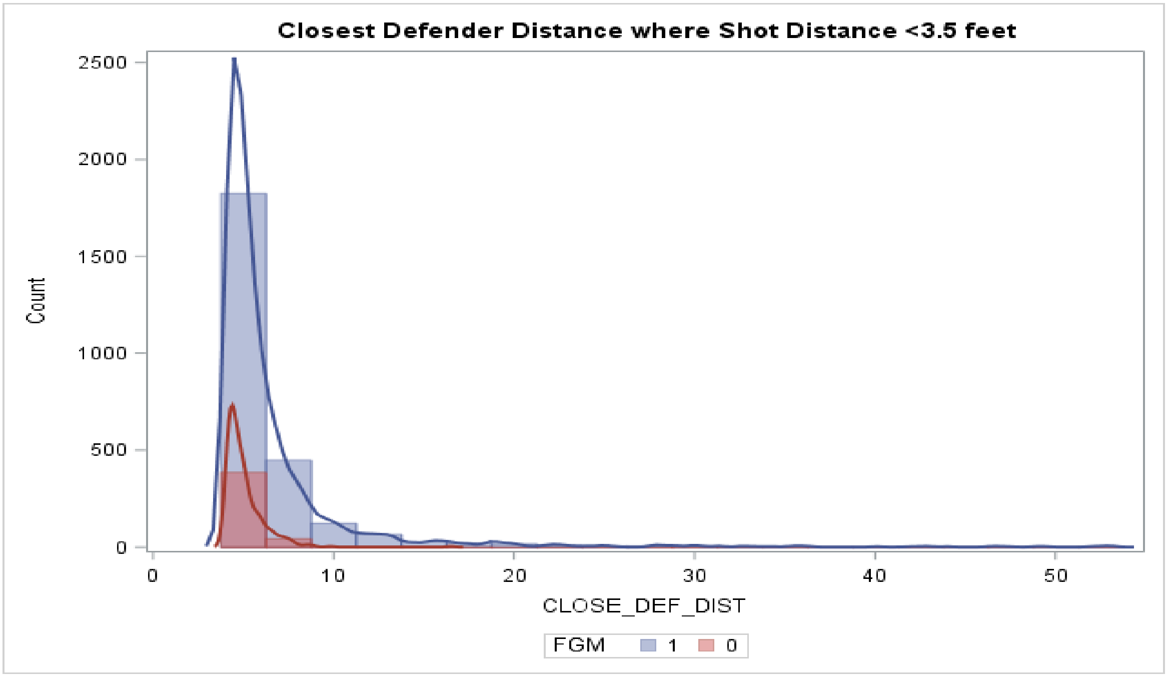


In order to pick the best baseline, a subset of the shot_log data was made for shots that were taken when there was less than 4 seconds on the period clock, less than 2 seconds on the shot clock with the shot distance being greater than 25 feet. A PROC UNIVARIATE7 was applied to the dataset to produce a histogram of the subset.



Knowing that these throws counted as shots are not common, you can tell where the drop-off is between shots that were intentionally taken that were a display of the player's skill and shots that were just throws that are not imperative to most of the game. The chosen distance for heaves was 30 feet.

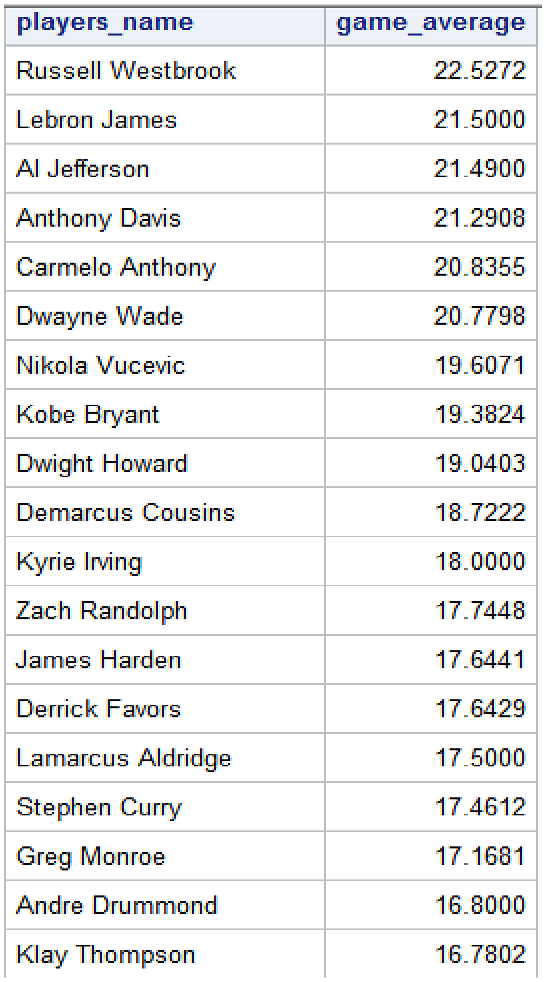
The next variable that needs to be modified is the closest defender distance variable, which can negatively impact a player's defensive metric on a fast-break layup or dunk or in a rare case where the primary defender who gave up the shot is not the closest defender. For example, a defender gets beat easily by the driving player and does not even attempt to chase the offensive player to the goal in which case another player who is closer to the offensive player with the ball would have been charged with a negative defensive score . In order to determine approximately which distance this is using the given data was done using PROC SGPLOT8.



It is obvious that a Close_Def_Dist of 10 feet is around a reasonable estimate given the significant drop-off occurring around that value.

Now the next step programming wise is to add the perform_o and perform_d metrics for all of the players for each observation. Doing this makes it easier to cumulatively sum up all of these metrics using the sum function in SAS for each player for each game into one variable overall_metric[9].

Once this is done, a per game average for the overall_metric is made for each player[10], and a corresponding table showing the players with the highest per game averages is displayed below:



players_name	game_average
Russell Westbrook	22.5272
Lebron James	21.5000
Al Jefferson	21.4900
Anthony Davis	21.2908
Carmelo Anthony	20.8355
Dwayne Wade	20.7798
Nikola Vucevic	19.6071
Kobe Bryant	19.3824
Dwight Howard	19.0403
Demarcus Cousins	18.7222
Kyrie Irving	18.0000
Zach Randolph	17.7448
James Harden	17.6441
Derrick Favors	17.6429
Lamarcus Aldridge	17.5000
Stephen Curry	17.4612
Greg Monroe	17.1681
Andre Drummond	16.8000
Klay Thompson	16.7802

This table is consistent with the public perception of the NBA's best players.

Then the scores were standardized to get a better gauge at how much better the elite players were compared to the rest. A table of the standardized data of the best players in the league for the 2015 season is below[11]:

Obs	player_name	game_average
1	Lebron James	3.27373
2	Anthony Davis	3.04865
3	Russell Westbrook	3.04789
4	Stephen Curry	2.80640
5	Nikola Vucevic	2.76080
6	Carmelo Anthony	2.73950
7	Dwayne Wade	2.71259
8	Al Jefferson	2.63322
9	Demarcus Cousins	2.58121
10	Chris Paul	2.44270
11	Monta Ellis	2.38830
12	Kyrie Irving	2.37631
13	Klay Thompson	2.35354
14	Blake Griffin	2.28782
15	Lamarcus Aldridge	2.22204
16	Derrick Favors	2.20410
17	James Harden	2.18870
18	Dwight Howard	2.12595
19	Goran Dragic	2.06804

Descriptive Statistics of the Metric is below:

Analysis Variable : game_average				
N	Mean	Std Dev	Minimum	Maximum
472	4.5494704	6.1106016	-3.6346154	24.5539216

Since the scores for all of the players are available, the next step is to subset the scores of players who were traded into a before_traded and after_traded datasets and using this data to do further analysis whether the players performed just as good or were better with their new teams[12].

Below is a list of players who changed teams and their corresponding game average metric before and after the corresponding player got traded:

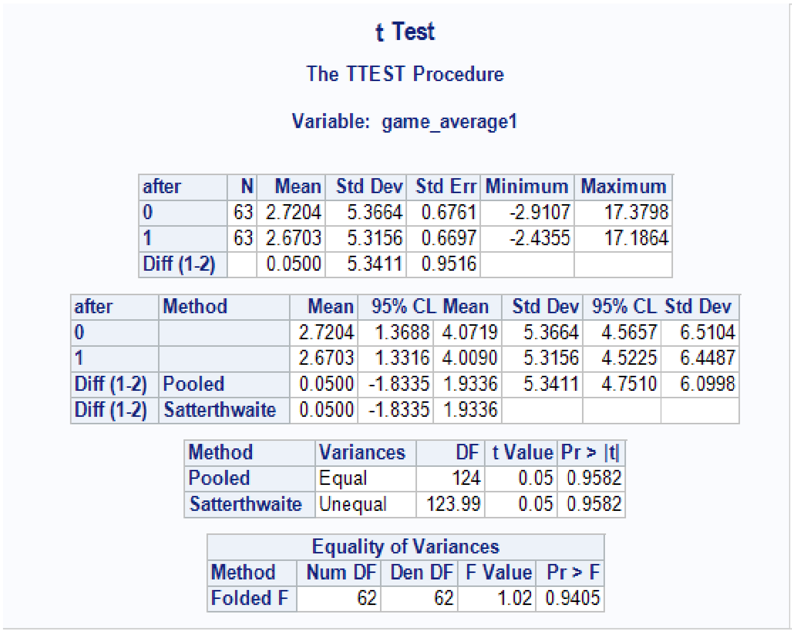
Obs	player_name	game_average_bef	game_average_af
1	Kevin Garnett	5.8512	5.7609
2	Andre Miller	3.9826	3.8693
3	Amare Stoudemire	14.1765	14.0461
4	Tayshaun Prince	-1.1400	-1.2059
5	Kendrick Perkins	3.6078	3.4545
6	Mo Williams	9.1563	10.1543
7	Josh Smith	-2.9107	-2.4355
8	J.R. Smith	-1.7609	-2.1863
9	Jameer Nelson	-1.5217	-1.4310
10	Shavlik Randolph	-1.2083	-0.9286
11	Shawne Williams	2.6221	2.6685
12	Rajon Rondo	-1.3636	-1.5957
13	Lou Amundson	-0.6250	-1.0370
14	Jeff Green	12.1894	9.8364
15	Corey Brewer	-2.5000	-2.1034
16	Brandan Wright	-1.0000	-0.8571
17	Thaddeus Young	14.1117	13.8287
18	Arron Afflalo	12.3798	11.5877
19	Ramon Sessions	3.1042	2.7616
20	Anthony Tolliver	-1.1750	-1.5625
21	Dj Augustin	7.9954	8.0533
22	JaVale McGee	-1.4000	-1.3810
23	Goran Dragic	17.3798	17.1864
24	Jonas Jerebko	3.2826	3.7067
25	Marcus Thornton	7.9038	7.2733
26	A.J. Price	-0.9500	-1.0294
27	Alonzo Gee	3.8264	3.7961
28	Quincy Pondexter	-1.1111	-1.3667
29	Timofey Mozgov	7.0074	8.5205
30	Gary Neal	7.3547	7.5969
31	Ish Smith	-0.1111	-0.4167
32	Lance Thomas	-1.4474	-1.3065
33	Enes Kanter	12.5663	13.0227
34	Brandon Knight	13.8650	13.4079
35	Iman Shumpert	-2.7273	-2.2619
36	Reggie Jackson	10.2150	10.4500
37	Norris Cole	5.1685	5.7264
38	Kyle Singler	3.7917	3.5902
39	Isaiah Thomas	10.5870	10.9245

Obs	player_name	game_average_bef	game_average_af
40	Dion Waiters	-1.6563	-1.6897
41	Thomas Robinson	-1.2308	-1.1935
42	Austin Rivers	-0.9706	-1.1636
43	Miles Plumlee	-1.2264	-1.2193
44	Jae Crowder	-1.0227	-1.5536
45	Will Barton	-0.2250	-0.5962
46	Pablo Prigioni	3.0732	2.7234
47	Alexey Shved	-1.5313	-1.2727
48	Chris Johnson	-1.1111	-0.6818
49	JaMychal Green	0.0000	-0.5000
50	Jorge Gutierrez	-0.8333	-0.3889
51	Isaiah Canaan	-0.6111	-0.9200
52	Michael Carter-Williams	10.6159	10.1141
53	Nate Wolters	-0.2222	-0.1429
54	Reggie Bullock	-0.6905	-0.6522
55	Gigi Datome	0.5000	1.0000
56	Brandon Davies	-2.1250	-1.8750
57	Troy Daniels	-0.7143	-0.4737
58	Tyler Ennis	-1.0000	-0.9333
59	Kj Mcdaniels	5.8873	5.6038
60	Dwight Powell	-0.5000	-1.2692
61	Adreian Payne	-1.0000	-2.0000
62	Tarik Black	-1.6957	-1.8026
63	Zoran Dragic	0.5000	1.0000

It must be noted that player usage is not considered in this analysis, which could affect the results.

Then a 2 sample independent t-test was used to determine if the difference between the two population subsets were different from one another where 0 is defined as before being traded and 1 being defined as after being traded.

-Results given bySAS Enterprise Guide



t Test

The TTEST Procedure

Variable: game_average1

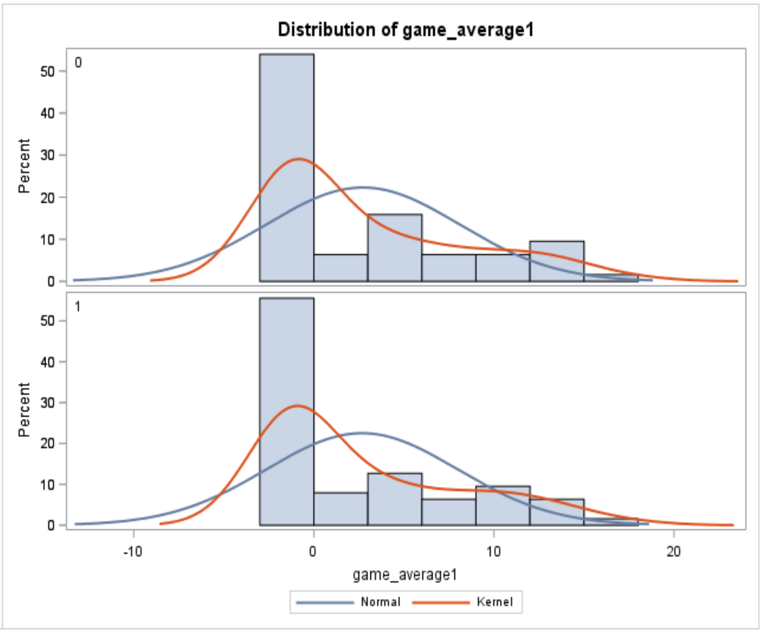
after	N	Mean	Std Dev	Std Err	Minimum	Maximum
0	63	2.7204	5.3664	0.6761	-2.9107	17.3798
1	63	2.6703	5.3156	0.6697	-2.4355	17.1864
Diff (1-2)		0.0500	5.3411	0.9516		

after	Method	Mean	95% CL Mean		Std Dev	95% CL Std Dev	
0		2.7204	1.3688	4.0719	5.3664	4.5657	6.5104
1		2.6703	1.3316	4.0090	5.3156	4.5225	6.4487
Diff (1-2)	Pooled	0.0500	-1.8335	1.9336	5.3411	4.7510	6.0998
Diff (1-2)	Satterthwaite	0.0500	-1.8335	1.9336			

Method	Variances	DF	t Value	Pr > |t|
Pooled	Equal	124	0.05	0.9582
Satterthwaite	Unequal	123.99	0.05	0.9582

Equality of Variances

Method	Num DF	Den DF	F Value	Pr > F
Folded F	62	62	1.02	0.9405



Using the information gathered from this test, it is concluded that collectively the players did not significantly outperform or underperform after being traded to a new team in 2015. This conclusion can change if extra data is collected for all players who were ever traded in NBA history in order to determine if there is a pattern or it can change if additional information is provided about the players such as position, physical attributes, minutes played, etc.

Appendix:

The first data set that is loaded in the code below with the two level name tmp1.shot_log is the raw data file converted from an excel file to a SAS dataset from the link:

https://www.kaggle.com/dansbecker/nba-shot-logs

Code:

*It is worth noting that the code is listed in the order in which it is run in order to get the wanted results.

1.

```
data player_transformation;
       set tmp1.shot_log; /*after opening raw data set*/
       players_name=propcase(player_name);
       closest_defender=scan(closest_defender,2,',')!!scan(closest_defender,1,',')
;
       if shot_clock ne .;
       drop player_name;
       if players_name="Mnta Ellis" then players_name="Monta Ellis";
run;
proc print data=player_transformation (obs=10);
 where players_name="Monta Ellis";
run;

data shooter_team_set;
       set player_transformation;
       Shooter_Team=scan(matchup,5,' ');
run;
proc print data=shooter_team_set(obs=10);

run;

3.

data date_formatted;                              /* changing the dates into SAS
dates*/
       set shooter_team_set;
       date=scan(matchup,1,'-');
       month= substr(date,1,3);
       day=substr(date,5,2);
       year=substr(date,9,4);
       daten=cat(day,month,year);

       date_fix=strip(daten);
       date_fix1=compress(date_fix);

       sas_date=input(date_fix1,date9.);

       drop day year month daten date_fix date_fix1;

run;
proc print data=date_formatted (obs=33);
run;
```

4.

```
data contested_shot; /* start footnote 3*/
      set date_formatted;
      if close_def_dist<5 then
            contested_shot=1;
      else contested_shot=0;
run;
```

5.

```
data o_metric;
      set contested_shot;

      if fgm=1 and contested_shot=0 and dribbles>=2 then  /*situation a*/
            perform_o=5.5;
      else if fgm=0 and contested_shot=0 and dribbles>=2 then
            perform_o=-0.75;


      if fgm=1 and contested_shot=1 and dribbles<2 then  /*situation b*/
            perform_o=4;
      else if fgm=0 and contested_shot=1 and dribbles<2 then
            perform_o=-0.50;

      if fgm=1 and contested_shot=1 and dribbles>=2 then  /*situation c*/
            perform_o=3;
      else if fgm=0 and contested_shot=1 and dribbles>=2 then
            perform_o=-0.50;

      if fgm=1 and contested_shot=0 and dribbles<2 and shot_dist>=3.5 then
/*situation d*/
            perform_o=2;
      else if fgm=0 and contested_shot=0 and dribbles<2 and shot_dist>=3.5 then
            perform_o=-0.75;

      if fgm=1 and contested_shot=0 and dribbles<2 and shot_dist<3.5 then
/*situation e*/
            perform_o=1;
      else if fgm=0 and contested_shot=0 and dribbles<2 and shot_dist<3.5 then
            perform_o=-2;
 run;
proc print data=o_metric (obs=100);
run;
```

6.

```
data d_metric;
      set contested_shot;

      if close_def_dist<10;

      if contested_shot=1 and fgm=0 then
            perform_d=0.50;
      else if contested_shot=1 and fgm=1 then
            perform_d=-0.50;

      if contested_shot=0 then
            perform_d=-1;
      closest_def=compbl(scan(closest_defender,2,',')!!scan(closest_defender,1,',
'));
      defend_obs=1;
      drop closest_defender;

run;
```

7.

```
data heaves_sub;  /*not part of used data set*/
      set contested_shot;
      if (game_clock<'00:05:00't or shot_clock<2)and shot_dist>25;
run;

proc univariate data=heaves_sub;
      var shot_dist;
      histogram;
      inset mean std;
run;
```

8.

```
title'Closest Defender Distance where Shot Distance <3.5';
proc sgplot data=d_metric;
      where shot_dist<3.5;
      histogram close_def_dist / group=fgm transparency=0.5 scale=count;
      density close_def_dist / type=kernel group=fgm ;
run;
title;
```

9.

```
data final_in;
      set d_metric(rename=(CLOSEST_DEFENDER_PLAYER_ID=id
      closest_def=player_name players_name=offensive_player ))
      o_metric(rename=(player_id=id players_name=player_name )) ;
run;
proc print data=final_in (obs=100);
run;

proc sort data=final_in;
      by id  sas_date;
run;
/*proc print data=final_in (obs=1000);
      var id  game_id sas_date team_change;
run;*/

data overall;
      set final_in ;
      by id ;

      if defend_obs=1 then player_team=scan(matchup,7, " ");
      else player_team=scan(matchup,5, " ");

      if first.id=1  then do ;
            overall_metric=0;
            overall_metric=sum(perform_d,perform_o);
            /*putlog overall_metric= perform_d= perform_o=;*/
```

```
          output;
      end;

      if first.id=0  then do;
          overall_metric=sum(perform_d,perform_o);
          /*putlog overall_metric= perform_d= perform_o=;*/
          output;
      end;

      if last.id=1  then do;
          overall_metric=sum(perform_d,perform_o);
          /*putlog overall_metric= perform_d= perform_o=;*/
          output;
      end;

run;

proc print data=overall (obs=10);
run;

2.

proc sort data=overall;
      by id game_id;
      run;
proc print data= shooter_team_set (obs=300);
      var player_id shooter_team  game_id ;
      run;
data team_change; /*max team change for one player is 1*/
      set overall;

      retain player_team_1 player_id_1 ;
      retain team_change 0;

      if player_team_1 ne player_team and player_id_1=id then do
          team_change=team_change+1;
          player_team_1=player_team;
      end;
      else if player_id_1 ne id then do
          team_change=0;
          player_team_1=player_team;
          player_id_1=id;
      end;
run;
proc print data=team_change (obs=600);
      var id player_team team_change;
run;

10.

proc sort data=team_change;
      by id game_id;
run;
proc freq data=team_change nlevels;

tables id player_name;
run;
proc print data=team_change (obs=1000);
where player_name="Brook Lopez";
var id player_name ;
```

```
run;
data average_per_game game_metric;
      set team_change;
      by id game_id ;

      if first.id=0 and first.game_id=0 then do ;
            sum_game_metrics+overall_metric;
            output game_metric;
            end;
      if first.id=0 and first.game_id=1 then do;
            sum_game_metrics+overall_metric;
            i+1;
            output game_metric;
      end;
      else if first.id=1 and first.game_id=1 then do;
            sum_game_metrics=0;
            i=0;
            sum_game_metrics+overall_metric;
            i+1;
            output game_metric;
      end;

      if  last.id=1 and last.game_id=1 then do;
            sum_game_metrics+overall_metric;
            output game_metric;
            game_average=sum_game_metrics/i;

      end;

      if game_average ne . then output average_per_game;

run;
proc freq data=game_metric nlevels;
      tables id/noprint;
run;

proc sort data=average_per_game;
      by descending game_average;
run;

proc print data=average_per_game;
      var player_name game_average ;
run;
proc means data=average_per_game;
      var game_average;
run;
```

11.

```
proc standard data=average_per_game mean=0 std=1 out=z_mean_overall;
  var game_average ;
RUN;
proc print data=z_mean_overall;
var player_name game_average;
run;
```

12.

```
data team_change_bef_data team_change_af_data;   /*separating the data between the
players that changed teams and that didnt change teams*/
      set game_metric;
      if team_change=0 then                        /*problem with having stats for
players before changing teams*/
            output team_change_bef_data;
      if team_change=1 then
            output team_change_af_data;
run;
proc sort data=team_change_af_data;
      by id;
run;
data team;
      set team_change_af_data;
      by id;
if last.id=1 then output;
      keep id;
run;
proc print data=team;
run;

data team_change_bef; /*sorting out the players stats before they were traded*/
      merge team(in=t) team_change_bef_data (in=tcb);
      by id;
      if t=1 and tcb=1;

run;

proc sort data=team_change_bef;
      by id i;
run;
proc sort data=team_change_af_data;
      by id i;
run;

data team_change_bef_avg;
      set team_change_bef;
      by id i;
      if last.id=1  then
            game_average_bef=sum_game_metrics/i;
      if last.id=1;
run;
data team_change_af_avg;
      set team_change_af_data;
      by id i;
      if last.id=1  then;
            game_average_af=sum_game_metrics/i;
      if last.id=1;
run;
proc sort data=team_change_af_avg;
      by id;
run;
proc sort data=team_change_bef_avg;
      by id;
run;
```

```
data combined_avg;
	merge team_change_bef_avg team_change_af_avg;
	by id;
	keep player_name game_average_bef game_average_af;
run;

proc print data=combined_avg;
run;
proc print data=team_change_af_avg;
	var player_name game_average_af;
run;
```