The Two-for-One

Caleb Kassa

May 5, 2020

Introduction

In NBA games, it is common for a team to leverage the amount of time remaining in a quarter to earn as many possessions as possible. This is the thought process behind the two-for-one strategy. Since an NBA shot clock is 24 seconds, the strategy calls to shoot the ball with between 26 and 36 seconds left in the quarter to ensure that your team will earn possession after the other team's 24 second possession ends. Thus, your team gets two possessions at the end of the quarter, and the opponent gets one.

To measure the value of a two-for-one, I use a metric called the **Differential Gain** (DG, inspiration from https://www.racketracer.com/2015/05/12/analytics-of-optimal-2-for-1-strategy-in-nba-basketball/). The DG is the net amount of points that the team who takes the two-for-one shot gains (or loses) once their two-for-one shot is taken. For example, if the Hawks attempt a 3-point shot with 30 seconds left and make it, the opposing team then makes another 3-pointer, and the Hawks make a lay-up at the end of the quarter, the DG on that Hawks two-for-one attempt is $3 - 3 + 2 = 2$.

In this analysis, I attempt to answer a few questions about two-for-ones. Do teams actually go for the two-for-one? How much time left in the quarter makes for the optimal two-for-one shot? How does the distance from which the shot was taken affect the value of the two-for-one? How does being fouled on the shot play a role? The data I use is certainly imperfect. Some of the shots I deem as two-for-one attempts could be circumstance. Nonetheless, I believe enough of the data is relevant as this is a widely used tactic (we will see some evidence of this). I will perform a multiple linear regression using the distance from which the shot was taken, the time remaining in the quarter, and whether or not the player was fouled on the shot. If the player was fouled, the points they score from the free throws are counted. I use plays only in the first three quarters to avoid complications with end of game strategy and fouling.

Data Cleaning

To perform this analysis, I scraped play-by-play data from https://www.basketball-reference.com/ for every NBA game played from the 2014-2015 season through 2019-2019 using the script 'get_pbp_pages.R', and saved the html pages to my computer locally. Then, in the script 'create_df.R', I created a function to extract all of the play-by-play data from a Basketball Reference Play-By-Play html page, and applied that function to every game I saved on my computer, merging them all together to create one large dataframe with all of the games. Here is what some of the more important columns of the resulting dataframe look like.

```
# Load libraries
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
x <- read.csv('pbp_df.csv', as.is = T)
knitr::kable(head(x[, c(1,2,3,9,10)])) %>%
  kable_styling(latex_options = "scale_down")
```

time	play	team	points_scored	period
11:40.0	C. Boozer misses 2-pt jump shot from 8 ft	Los Angeles Lakers	0	1
11:39.0	Defensive rebound by D. Howard	Houston Rockets	0	1
11:23.0	D. Howard makes 2-pt hook shot from 5 ft (assist by J. Harden)	Houston Rockets	2	1
11:05.0	J. Hill misses 2-pt layup from 1 ft	Los Angeles Lakers	0	1
11:04.0	Defensive rebound by Team	Houston Rockets	0	1
10:49.0	Shooting foul by W. Johnson (drawn by D. Howard)	Houston Rockets	0	1

From here, I needed to make a few changes to perform the two-for-one analysis. The data cleaning is below.

```
# Get plays from the first three quarters:
x <- x %>% filter(period %in% (1:3))

# Look at plays in just the last minute of each quarter:
x <- x %>% filter(grepl("^0:", time))

# Convert time to seconds:
x$time <- as.integer(gsub("^0:", "", x$time))

# Get only the shot attempts:
x <- x %>% filter(grepl("jump shot|layup|dunk|free throw", play))

# But now free throws are counted as multiple "plays".
# Include only the last free throw, but add the points scored from the
# preceeding free throws as well.
x <- x %>% mutate(points_scored = if_else(grepl("free throw", play) &
                                            grepl("free throw", lag(play)),
                                          points_scored + lag(points_scored),
                                          points_scored))
x <- x %>% filter(!(grepl("free throw", play) & grepl("free throw", lead(play))))

# Home/away score to team/opponent score:
x$team.score <- ifelse(x$team == x$away.team, x$away.score, x$home.score)
x$opp.score <- ifelse(x$team == x$away.team, x$home.score, x$away.score)
x <- x[, -c(4,5)]

# Add an opponent column:
x$opponent <- ifelse(x$team == x$away.team, x$home.team, x$away.team)
x <- x %>% select(time, play, team, opponent, everything())

# Get the distance from which the shot was taken:
x$shotdistance <- gsub(".* (\\d+ ft).*", "\\1", x$play)
```

```
x$shotdistance <- ifelse(grepl("at rim", x$shotdistance), 0L,
                         ifelse(grepl("free throw", x$shotdistance), NA,
                                x$shotdistance))
x$shotdistance <- as.integer(gsub(" ft", "", x$shotdistance))

# Add a column for foul: 1 if yes, 0 if no:
x$foul <- ifelse(grepl("free throw", x$play), 1L, 0L)

# Let's keep all the plays in the last minute of the game in x, and
# create a new dataframe for all the two-for-one shot attempts y.
y <- x %>% filter(time <= 36)

# Add a differential gain column. This column will tell us what the
# point differential is from the time a team shoots a two-for-one shot
# to the end of that quarter.
# First, sum the total points scored for each team in the last 36 seconds
# of each quarter:
y <- y %>% group_by(gameid, period, team) %>%
  mutate(team.points = sum(points_scored))

# Then, subtract the team's total points from the opponent's total points,
# and look at shots taken between 28 and 36 seconds remaining:
y <- y %>% group_by(gameid, period) %>%
  mutate(max.teampoints = max(team.points)) %>%
  mutate(min.teampoints = min(team.points)) %>%
  mutate(diff.gain = if_else(max.teampoints == min.teampoints, max.teampoints,
                             if_else(team.points == max.teampoints,
                                     max.teampoints - min.teampoints,
                                     min.teampoints - max.teampoints))) %>%
  filter(time %in% 26:36)

nrow(y)
```

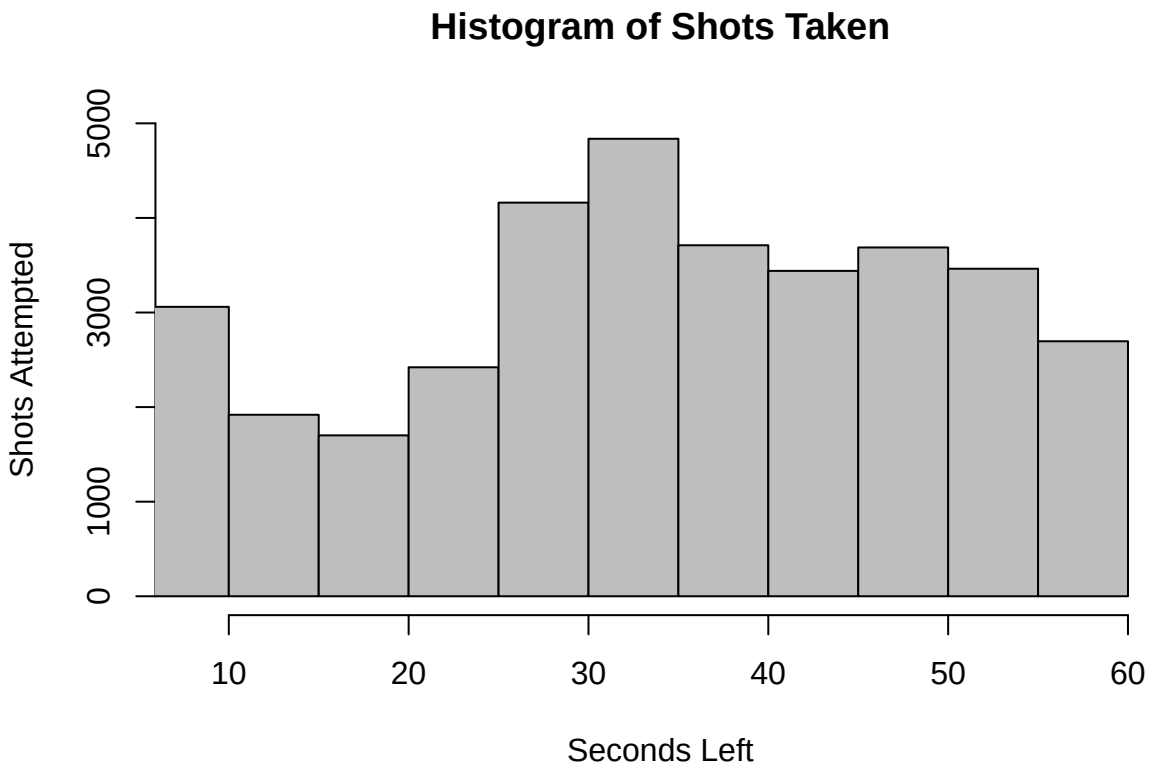
```
## [1] 9792
```

After all of the data cleaning, we are left with 9,792 two-for-one shot attempts.

Data Exploration

First, let's look at a histogram showing the distribution of the times at which shots in the last minute of a quarter are taken.

```
hist(x$time, xlab = "Seconds Left", ylab = "Shots Attempted",
     main = "Histogram of Shots Taken", xlim = c(8, 60),
     ylim = c(0, 5000), col = "grey")
```



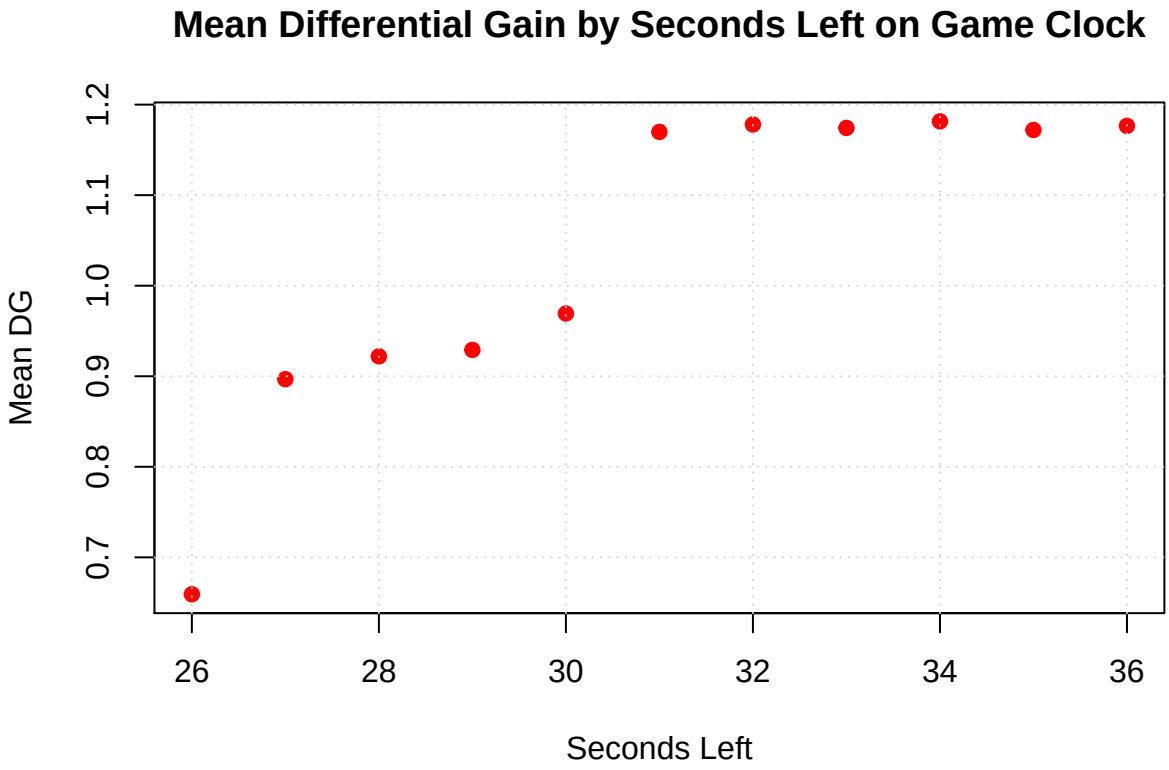
Here we can see that in the last minute of a quarter (excluding the last 10 seconds when a team would be wasting a possession if they don't shoot), shots are taken most frequently when there are between 25 and 35 seconds left in the quarter. This provides evidence to the claim that teams often do attempt the two-for-one. Let's calculate the overall mean DG on all of the attempts to see if teams are generally doing the right thing going for two-for-ones.

```
mean(y$diff.gain)
```

```
## [1] 1.05433
```

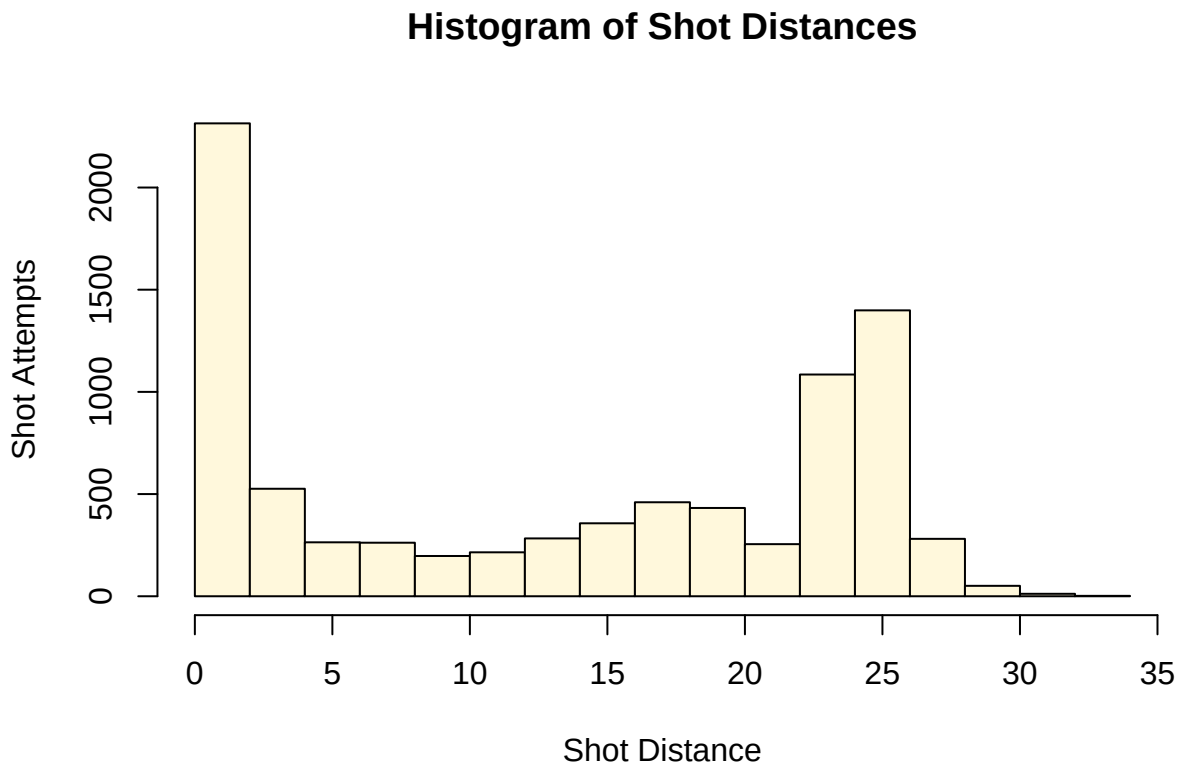
On average, a two-for-one attempt improves your point differential by 1.048 at the end of the quarter. Let's take a look at how the DG is affected by the time at which the two-for-one is attempted.

```
temp <- y %>% group_by(time) %>% mutate(mean.diffgain = mean(diff.gain)) %>%
  distinct(time, mean.diffgain) %>% arrange(time)
plot(mean.diffgain ~ time, data = temp,
     xlab = "Seconds Left", ylab = "Mean DG",
     main = "Mean Differential Gain by Seconds Left on Game Clock",
     pch = 19, col = "red")
grid()
```

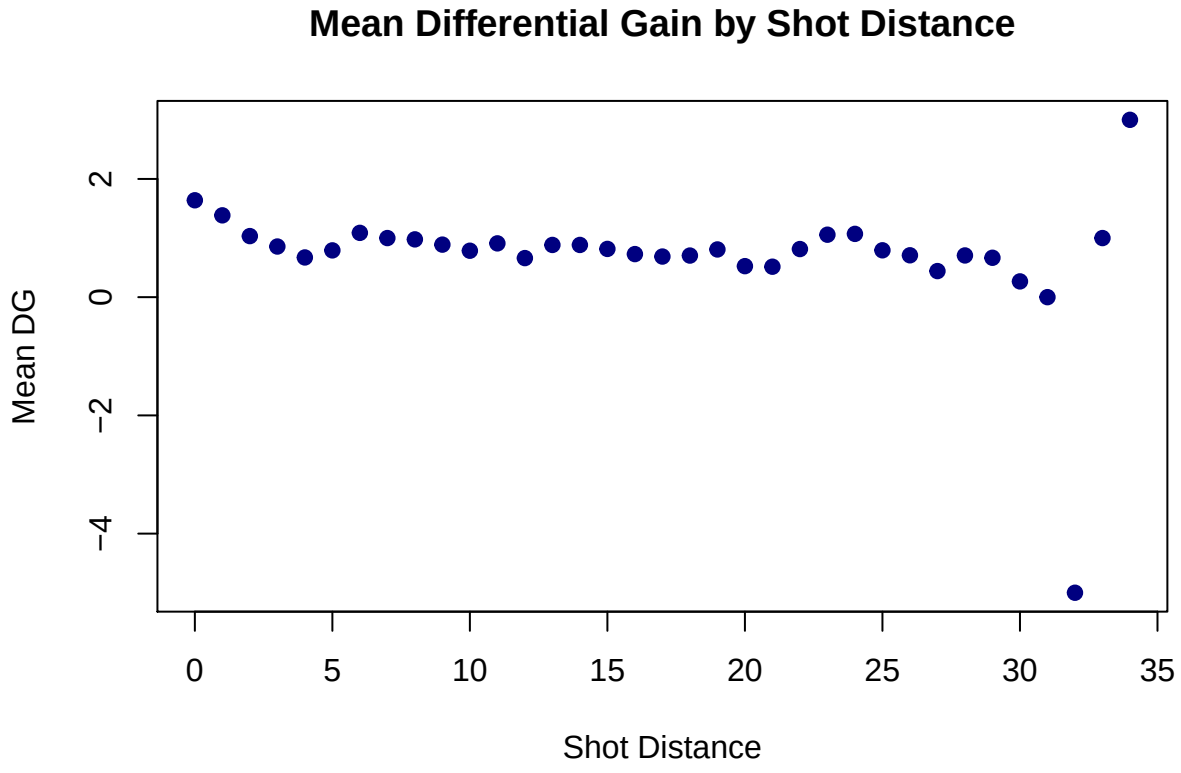


The DG tends to increase with time from 26 seconds left to 36. This is consistent with intuition; if a two-for-one is attempted with 26 seconds left in the quarter, that leaves you with only 2 seconds left on the clock if the other team uses the full 24 second shot clock. With more time after the opposing team's possession, you can get a better quality shot. There are large increases in the mean differential gain from 26 to 27 seconds, and from 31 to 32 seconds. Now let's explore shot distance and its relationship with DG.

```
temp <- y %>% group_by(shotdistance) %>% mutate(mean.diffgain = mean(diff.gain)) %>%
  distinct(shotdistance, mean.diffgain) %>% arrange(shotdistance)
hist(y$shotdistance, xlab = "Shot Distance", ylab = "Shot Attempts",
     main = "Histogram of Shot Distances", col = "cornsilk")
```




```
plot(mean.diffgain ~ shotdistance, data = temp,
     xlab = "Shot Distance", ylab = "Mean DG",
     main = "Mean Differential Gain by Shot Distance",
     pch = 19, col = "navy")
```

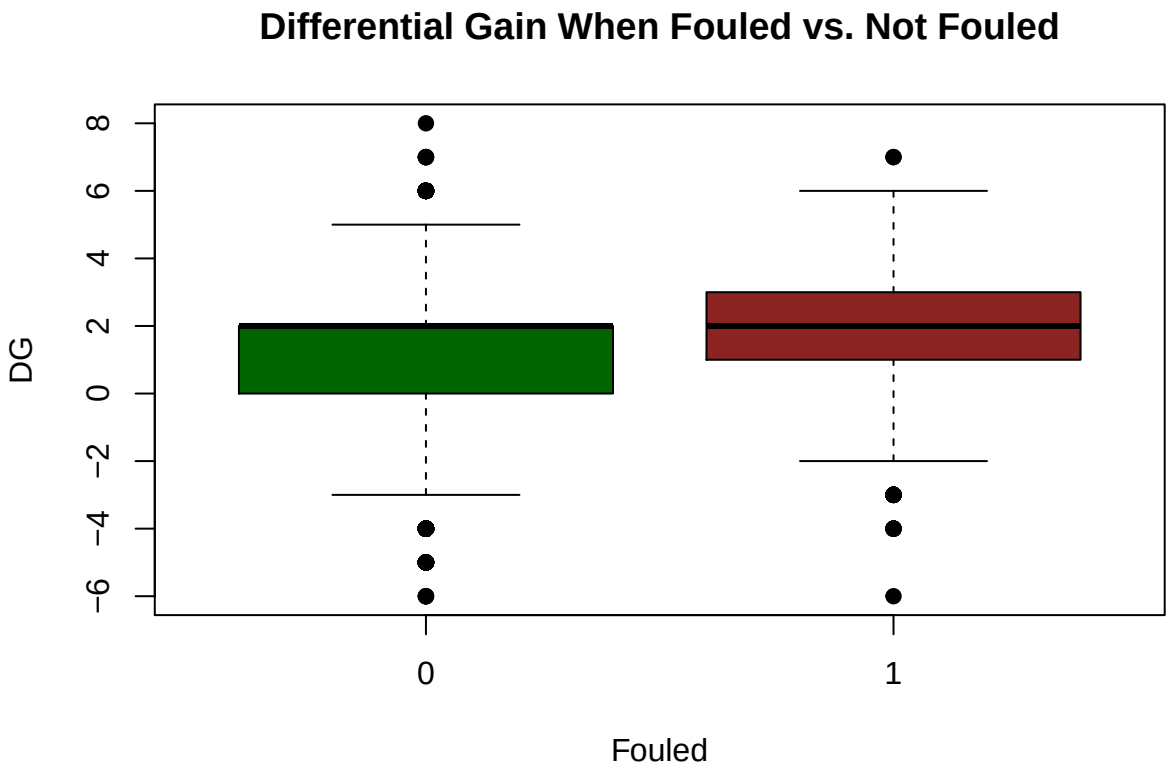


In the histogram, we see shots are taken most frequently close to the rim, and from between 22 and 25 feet. The NBA 3-point line is 22.75 feet away from the rim, so the histogram shows us what we already know: most shots taken are layups or threes. In the scatterplot, we can see some outliers as the shot distance goes up to over 32 feet, 10+ feet beyond the 3-point line. Let's remove those now.

```
# Need NA because free throw attempts
y <- y %>% filter(shotdistance < 32 | is.na(shotdistance))
```

Without the outliers, the differential gain is highest for shots at the rim, levels off at about 0.7 as shot distance increases, and then slightly increases up to about 1 as the shot distance approaches that of a 3-pointer. Lastly, let's take a look at how getting to the free throw line affects the differential gain.

```
boxplot(y$diff.gain ~ y$foul, xlab = "Fouled", ylab = "DG",
        main = "Differential Gain When Fouled vs. Not Fouled", pch = 19,
        col = c("darkgreen", "brown4"))
```




```
print(paste0("DG when fouled = ", mean(y$diff.gain[y$foul == 1]),
      "; DG when not fouled = ", mean(y$diff.gain[y$foul == 0])))
```

```
## [1] "DG when fouled = 1.59126700071582; DG when not fouled = 0.965443279313632"
```

When a player gets to the free throw line on a two-for-one shot attempt, the average DG increases to 1.59, about 0.63 points more than the average when a player is not fouled. Now, let's check the assumptions of a multiple linear regression before making predictions.

Predictions

Below, we will construct matrices to perform a multiple linear regression, solving the equation $Y = XB$. Y will be an $n \times 1$ column vector with $n = 9792$ being the number of two-for-one attempts, and each entry being the differential gain of the attempt. X will be an $n \times m$ design matrix with $m = 44$ for the number of indicator

variables we have. We have an indicator variable for each amount of seconds remaining ($time26 \dots time36$), an indicator variable for each shot distance ($distance0 \dots distance31$), and one last indicator for whether the player was fouled or not ($foul$). B will be an $m \times 1$ column vector that stores the predicted values when we solve the regression.

While it would have been convenient to perform a regression with time and distance as continuous variables, the shot distance when a player is fouled is null (the distance of a free throw is not representative of what the shot distance variable is measuring). Since both the time and distance variables were recorded as discrete values in the play-by-play data, this was how I was able to include all three variables with the null cases.

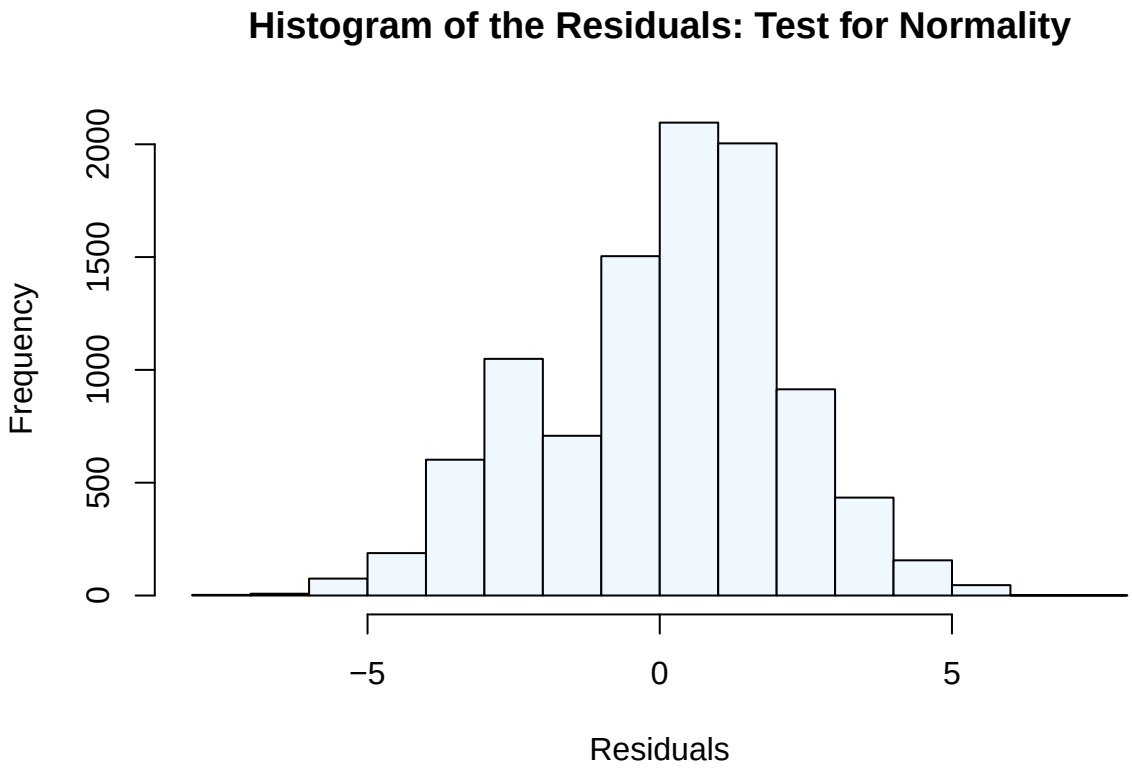
```
# Build matrices for regression:
Y <- matrix(y$diff.gain)
# Do not need to add 1 to ncol for "foul" column because NA is included as a
# unique shot distance when we don't sort it.
X <- matrix(0L, nrow = nrow(y),
            ncol = (length(unique(y$shotdistance)) + length(unique(y$time))))
colnames(X) <- c("foul", paste0("distance", sort(unique(y$shotdistance))),
                 paste0("time", sort(unique(y$time))))

# Fill the matrix:
for (i in 1:nrow(y)){
  shotdistance <- as.integer(y[i, "shotdistance"])
  time <- as.integer(y[i, "time"])
  X[i, "foul"] <- as.integer(y[i, "foul"])
  X[i, paste0("time", time)] <- 1L
  if (is.na(shotdistance) == F) {
  X[i, paste0("distance", shotdistance)] <- 1L
  }
}

# To get rid of collinearity, let's remove a variable and use that as our
# reference point. I choose to remove the "foul" variable.
B <- lsfit(X[,-1], Y, intercept = F)

# Create predictions:
y$preds <- NA
for (i in 1:nrow(y)) {
  time <- paste0("time", y[i, "time"])
  distance <- paste0("distance", y[i, "shotdistance"])
  if (y[i, "foul"] == 1) {
    y[i, "preds"] <- B$coefficients[time]
  }
  else y[i, "preds"] <- B$coefficients[time] + B$coefficients[distance]
}
```

Check Assumptions

Before proceeding with analysis, we will first need to check the assumptions of a linear model: normality of the residuals, constant variability of the residuals, independence of the residuals, and linear relationships between the predicted and observed values.

Normality of the Residuals

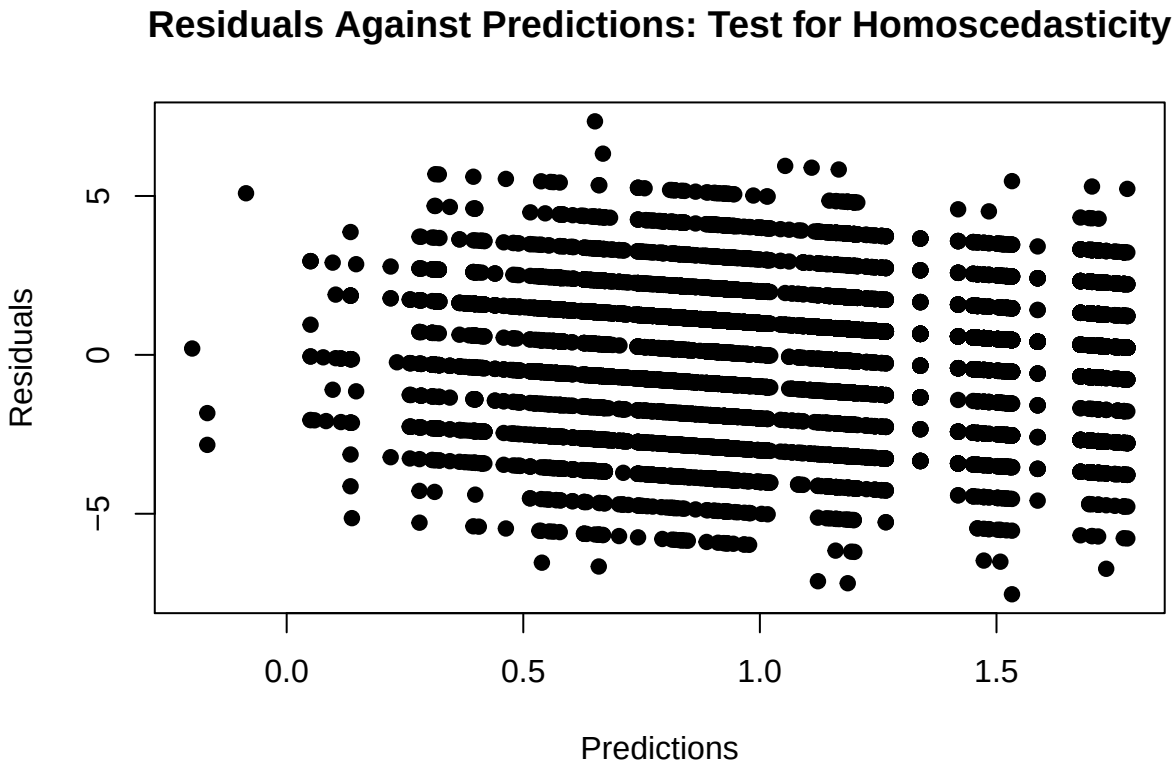
```
par(mfrow = c(1,1))
hist(B$residuals, xlab = "Residuals",
     main = "Histogram of the Residuals: Test for Normality", col = "aliceblue")
```



The residuals are approximately normally distributed.

Test Homoscedasticity

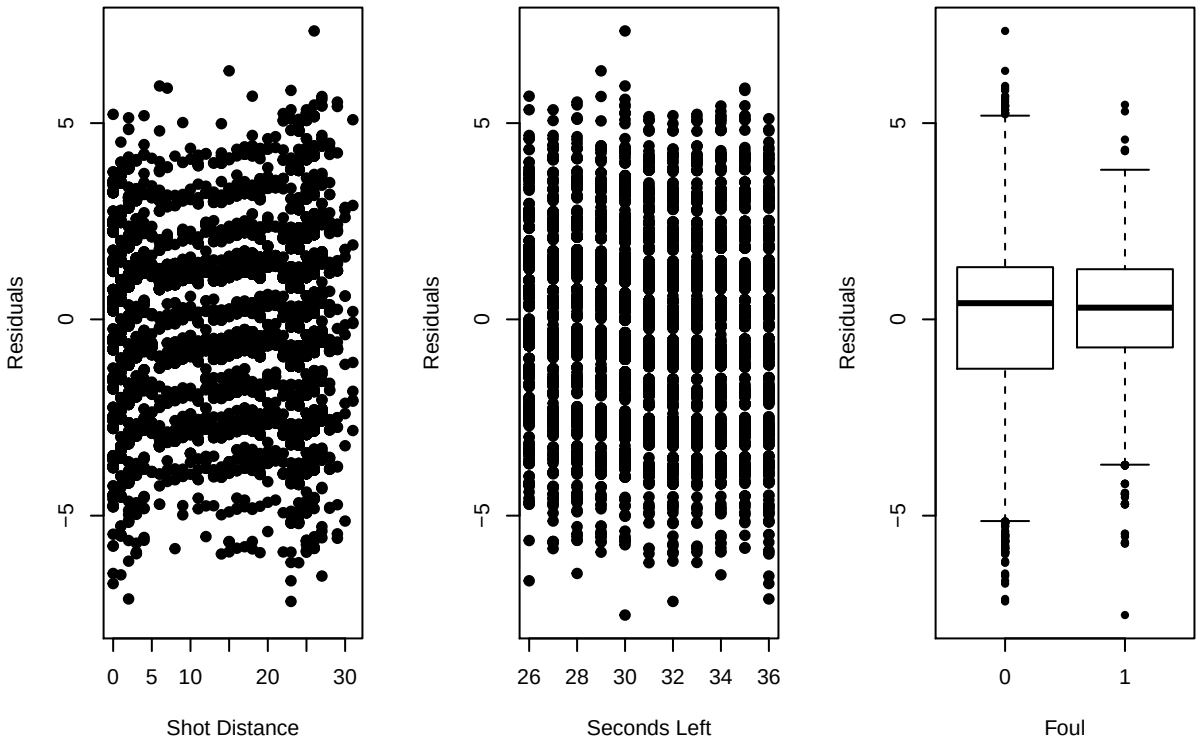
```
plot(B$residuals ~ y$preds, xlab = "Predictions", ylab = "Residuals",
     main = "Residuals Against Predictions: Test for Homoscedasticity", pch = 19)
```



The residuals do not seem to be strongly correlated with the predictions.

Independence of Residuals

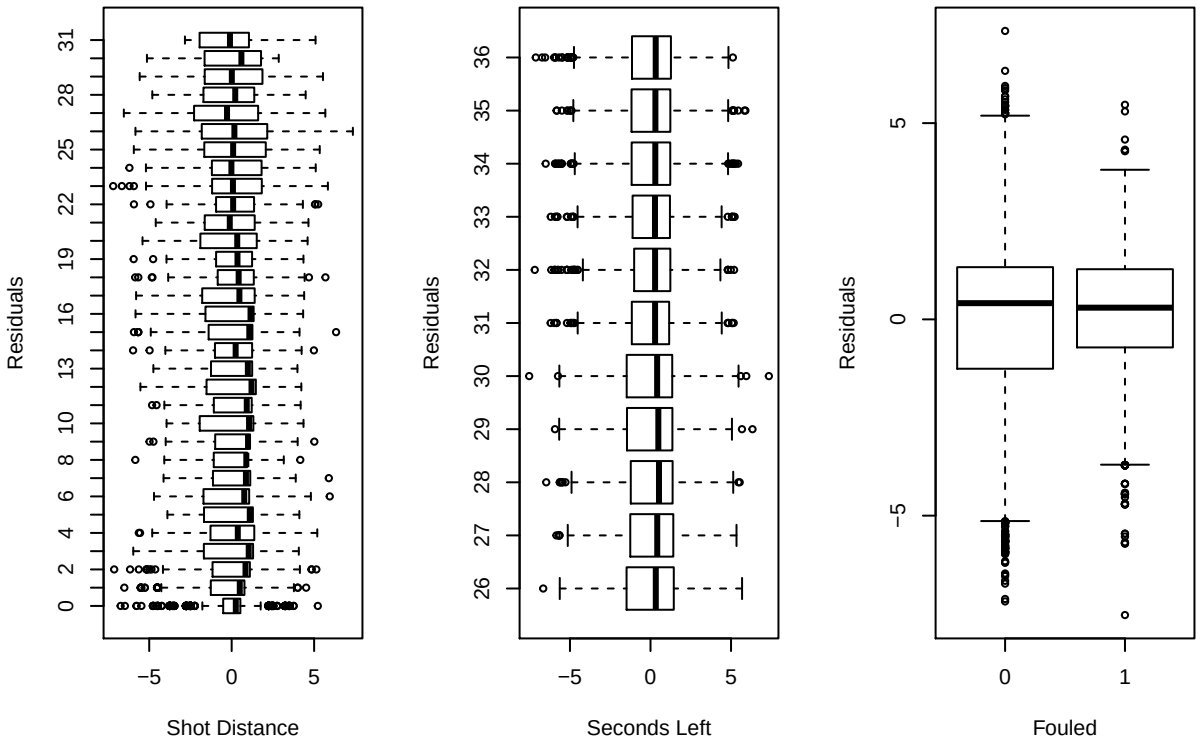
```
par(mfrow = c(1,3))
plot(B$residuals ~ y$shotdistance, xlab = "Shot Distance",
     ylab = "Residuals", pch = 19)
plot(B$residuals ~ y$time, xlab = "Seconds Left", ylab = "Residuals", pch = 19)
boxplot(B$residuals ~ y$foul, xlab = "Foul", ylab = "Residuals", pch = 19)
```



The residuals do no have any apparent correlation with any of the predictors.

Linearity Between Observed and Predicted

```
par(mfrow = c(1,3))
boxplot(B$residuals ~ y$shotdistance, xlab = "Shot Distance", ylab = "Residuals",
        horizontal = T)
boxplot(B$residuals ~ y$time, xlab = "Seconds Left", ylab = "Residuals",
        horizontal = T)
boxplot(B$residuals ~ y$foul, xlab = "Fouled", ylab = "Residuals")
```



Plotting the residuals against each predictor shows no notable differences in variability between groups. Now we can proceed with analysis of the regression.

Analysis

Below are the coefficients from the model.

```
B$coefficients
```

```
##   distance0   distance1   distance2   distance3   distance4   distance5
##  0.05427705 -0.19362998 -0.55488296 -0.74852490 -0.90458608 -0.81967381
##   distance6   distance7   distance8   distance9  distance10  distance11
## -0.47929217 -0.58646235 -0.60974514 -0.71567972 -0.77565756 -0.63403016
##  distance12  distance13  distance14  distance15  distance16  distance17
## -0.92585839 -0.70295922 -0.70035076 -0.79144596 -0.86588336 -0.92838434
##  distance18  distance19  distance20  distance21  distance22  distance23
## -0.87447519 -0.77099887 -1.05370955 -1.07411077 -0.77639885 -0.52915588
##  distance24  distance25  distance26  distance27  distance28  distance29
## -0.51634690 -0.79019243 -0.88126059 -1.13823278 -0.90393574 -0.95563960
##  distance30  distance31      time26      time27      time28      time29
## -1.31299863 -1.61863048  1.18872933  1.45089826  1.41892713  1.45977393
##      time30      time31      time32      time33      time34      time35
##  1.53275317  1.71588268  1.71480862  1.72218543  1.70185338  1.69586734
##      time36
##  1.67760146
```

To make a prediction, simply add the desired coefficients. For instance, a shot taken 5 feet from the basket with 28 seconds left is predicted to have a $-0.80 + 1.37 = 0.57DG$. If a player gets fouled with 32 seconds left, $DG = 0 + 1.66 = 1.66$. Note, if a player is fouled, the distance is not recorded and the predicted differential gain is simply the coefficient of the time at which the shot was taken. Recall that since we removed the $foul$ variable to avoid collinearity, the coefficients of the other variables assume that $foul = 0$. Also recall that if a

player gets fouled, the shot distance is not recorded. Therefore, the negative values for all of the distances indicates that getting fouled on a two-for-one attempt results in a higher differential gain than getting a shot off from any distance. The coefficient for a shot at the rim, however, is nearly zero (-0.000006), and the next lowest is from 1-2 feet away from the basket. **This could advise that driving to the rim for a shot at the basket and the potential to draw contact for a foul is more advisable than shooting a 3 in a two-for-one situation**.

Regarding the time left, the coefficients indicate that the best time to shoot a two-for-one is with 34 or 35 seconds left, and the worst time to shoot is with 26 seconds left, leaving the team with little time to get a good shot off of the second possession.

Conclusion

In this case study, we verified whether or not teams attempt the two-for-one, and analyzed the effects of shot distance, time, and the presence of fouling on the strategy. We used differential gain (DG) to measure the benefit of a two-for-one, and saw a benefit of employing the strategy. Our regression verified what we inferred about the optimal time of the two-for-one attempt: within the 26-36 second window, it is more effective to shoot earlier so that the team is left with more time to get a quality shot on the second possession. We also learned that getting fouled maximizes the return on a two-for-one attempt, and shots at the rim are better than 3-pointers.