Proj 2 - NBA Shot Logs

Val Wong - vmw170030

NBA Shot Logs Data Set

Data on shots taken during the 2014-2015 season, who took the shot, where on the floor was the shot taken from, who was the nearest defender, how far away was the nearest defender, time on the shot clock, and much more. The column titles are generally self-explanatory.

Useful for evaluating who the best shooter is, who the best defender is, the hot-hand hypothesis, etc. Scraped from NBA's REST API.

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

Step 1 - Loading the data

```
NBA_shotlogs <- read.csv("nba_shot_logs.csv")
numOfRows <- NROW(NBA_shotlogs)
print(paste("Total Number of Rows in NBA Shot logs: ", numOfRows, " rows."))
```

```
## [1] "Total Number of Rows in NBA Shot logs:  128069  rows."
```

Step 2 - Data Cleaning

First, we determine which columns contain NA values. Then, apply a fix/mitigation to columns with NA values. In this data set, the column 'Shot_Clock' contains NAs, which is replaced with either the mean or median of the column, with the function 'fix_NA'. Lastly, we factor columns that need to be factored to levels.

```
# Check NAs in each column
print("Overall NAs:")
```

```
## [1] "Overall NAs:"
```

```
sum(is.na(NBA_shotlogs) == TRUE)
```

```
## [1] 5567
```

```
print("SHOT_RESULT NAs:")
```

```
## [1] "SHOT_RESULT NAs:"
```

```
sum(is.na(NBA_shotlogs$SHOT_RESULT) == TRUE)
```

```
## [1] 0
```

```
print("TOUCH_TIME NAs:")
```

```
## [1] "TOUCH_TIME NAs:"
```

```
sum(is.na(NBA_shotlogs$TOUCH_TIME) == TRUE)
```

```
## [1] 0
```

```
print("DRIBBLES NAs:")
```

```
## [1] "DRIBBLES NAs:"
```

```
sum(is.na(NBA_shotlogs$DRIBBLES) == TRUE)
```

```
## [1] 0
```

```
print("SHOT_DIST NAs:")
```

```
## [1] "SHOT_DIST NAs:"
```

```
sum(is.na(NBA_shotlogs$SHOT_DIST) == TRUE)
```

```
## [1] 0
```

```
print("CLOSE_DEF_DIST NAs:")
```

```
## [1] "CLOSE_DEF_DIST NAs:"
```

```
sum(is.na(NBA_shotlogs$CLOSE_DEF_DIST) == TRUE)
```

```
## [1] 0
```

```
print("SHOT_CLOCK NAs:")
```

```
## [1] "SHOT_CLOCK NAs:"
```

```
sum(is.na(NBA_shotlogs$SHOT_CLOCK) == TRUE)
```

```
## [1] 5567
```

```
# function to mitigate presence of NAs in data set
# sample call: df$x <- fix_NA(df$x, 1)
fix_NA <- function(x, mean_mode){
  if (mean_mode == 1) { # use mean
    ifelse(!is.na(x), x, mean(x, na.rm=TRUE))
  } else {
    ifelse(!is.na(x), x, median(x, na.rm=TRUE))
  }
}

# fix NAs
NBA_shotlogs$SHOT_CLOCK <- fix_NA(NBA_shotlogs$SHOT_CLOCK, 1)

# Factoring Columns that have values that indicate levels
NBA_shotlogs$W <- factor(NBA_shotlogs$W)
NBA_shotlogs$LOCATION <- factor(NBA_shotlogs$LOCATION)
NBA_shotlogs$SHOT_RESULT <- factor(NBA_shotlogs$SHOT_RESULT)
NBA_shotlogs$PTS_TYPE <- factor(NBA_shotlogs$PTS_TYPE)
```

Step 3 - Data Exploration

Display Column Names and Structure of NBA Shot logs data set.

```
names(NBA_shotlogs) # data exploration function 1
```

```
##  [1] "GAME_ID"                   "MATCHUP"
##  [3] "LOCATION"                  "W"
##  [5] "FINAL_MARGIN"              "SHOT_NUMBER"
```

```
##  [7] "PERIOD"                    "GAME_CLOCK"
##  [9] "SHOT_CLOCK"                "DRIBBLES"
## [11] "TOUCH_TIME"                "SHOT_DIST"
## [13] "PTS_TYPE"                  "SHOT_RESULT"
## [15] "CLOSEST_DEFENDER"          "CLOSEST_DEFENDER_PLAYER_ID"
## [17] "CLOSE_DEF_DIST"            "FGM"
## [19] "PTS"                       "player_name"
## [21] "player_id"
```

```
str(NBA_shotlogs) # data exploration function 2
```

```
## 'data.frame':    128069 obs. of  21 variables:
##  $ GAME_ID                  : int  21400899 21400899 21400899 21400899 21400899 21400899 21400899 2
##  $ MATCHUP                  : chr  "MAR 04, 2015 - CHA @ BKN" "MAR 04, 2015 - CHA @ BKN" "MAR 04, 2
##  $ LOCATION                 : Factor w/ 2 levels "A","H": 1 1 1 1 1 1 1 1 2 ...
##  $ W                        : Factor w/ 2 levels "L","W": 2 2 2 2 2 2 2 2 2 ...
##  $ FINAL_MARGIN             : int  24 24 24 24 24 24 24 24 1 ...
##  $ SHOT_NUMBER              : int  1 2 3 4 5 6 7 8 9 1 ...
##  $ PERIOD                   : int  1 1 1 2 2 2 4 4 4 2 ...
##  $ GAME_CLOCK               : chr  "1:09" "0:14" "0:00" "11:47" ...
##  $ SHOT_CLOCK               : num  10.8 3.4 12.5 10.3 10.9 ...
##  $ DRIBBLES                 : int  2 0 3 2 2 2 11 3 0 0 ...
##  $ TOUCH_TIME               : num  1.9 0.8 2.7 1.9 2.7 4.4 9 2.5 0.8 1.1 ...
##  $ SHOT_DIST                : num  7.7 28.2 10.1 17.2 3.7 18.4 20.7 3.5 24.6 22.4 ...
##  $ PTS_TYPE                 : Factor w/ 2 levels "2","3": 1 2 1 1 1 1 1 1 2 2 ...
##  $ SHOT_RESULT              : Factor w/ 2 levels "made","missed": 1 2 2 2 2 2 2 2 1 2 2 ...
##  $ CLOSEST_DEFENDER         : chr  "Anderson, Alan" "Bogdanovic, Bojan" "Bogdanovic, Bojan" "Brown,
##  $ CLOSEST_DEFENDER_PLAYER_ID: int  101187 202711 202711 203900 201152 101114 101127 203486 202721 2
##  $ CLOSE_DEF_DIST           : num  1.3 6.1 0.9 3.4 1.1 2.6 6.1 2.1 7.3 19.8 ...
##  $ FGM                      : int  1 0 0 0 0 0 0 1 0 0 ...
##  $ PTS                      : int  2 0 0 0 0 0 0 2 0 0 ...
##  $ player_name              : chr  "brian roberts" "brian roberts" "brian roberts" "brian roberts"
##  $ player_id                : int  203148 203148 203148 203148 203148 203148 203148 203148 203148 2
```

Preview First and Last Few Rows

```
head(NBA_shotlogs) # data exploration function 3
```

```
##    GAME_ID                  MATCHUP LOCATION W FINAL_MARGIN SHOT_NUMBER PERIOD
## 1 21400899 MAR 04, 2015 - CHA @ BKN        A W           24           1      1
## 2 21400899 MAR 04, 2015 - CHA @ BKN        A W           24           2      1
## 3 21400899 MAR 04, 2015 - CHA @ BKN        A W           24           3      1
## 4 21400899 MAR 04, 2015 - CHA @ BKN        A W           24           4      2
## 5 21400899 MAR 04, 2015 - CHA @ BKN        A W           24           5      2
## 6 21400899 MAR 04, 2015 - CHA @ BKN        A W           24           6      2
##   GAME_CLOCK SHOT_CLOCK DRIBBLES TOUCH_TIME SHOT_DIST PTS_TYPE SHOT_RESULT
## 1       1:09   10.80000        2        1.9       7.7        2        made
## 2       0:14    3.40000        0        0.8      28.2        3      missed
## 3       0:00   12.45334        3        2.7      10.1        2      missed
## 4      11:47   10.30000        2        1.9      17.2        2      missed
## 5      10:34   10.90000        2        2.7       3.7        2      missed
## 6       8:15    9.10000        2        4.4      18.4        2      missed
##    CLOSEST_DEFENDER CLOSEST_DEFENDER_PLAYER_ID CLOSE_DEF_DIST FGM PTS
## 1    Anderson, Alan                     101187            1.3   1   2
## 2 Bogdanovic, Bojan                     202711            6.1   0   0
## 3 Bogdanovic, Bojan                     202711            0.9   0   0
```

```
## 4     Brown, Markel                    203900           3.4   0   0
## 5   Young, Thaddeus                    201152           1.1   0   0
## 6   Williams, Deron                    101114           2.6   0   0
##     player_name player_id
## 1 brian roberts    203148
## 2 brian roberts    203148
## 3 brian roberts    203148
## 4 brian roberts    203148
## 5 brian roberts    203148
## 6 brian roberts    203148
```

```
tail(NBA_shotlogs) # data exploration function 4
```

```
##         GAME_ID                  MATCHUP LOCATION W FINAL_MARGIN SHOT_NUMBER
## 128064 21400006 OCT 29, 2014 - BKN @ BOS        A L          -16           4
## 128065 21400006 OCT 29, 2014 - BKN @ BOS        A L          -16           5
## 128066 21400006 OCT 29, 2014 - BKN @ BOS        A L          -16           6
## 128067 21400006 OCT 29, 2014 - BKN @ BOS        A L          -16           7
## 128068 21400006 OCT 29, 2014 - BKN @ BOS        A L          -16           8
## 128069 21400006 OCT 29, 2014 - BKN @ BOS        A L          -16           9
##        PERIOD GAME_CLOCK SHOT_CLOCK DRIBBLES TOUCH_TIME SHOT_DIST PTS_TYPE
## 128064      2       5:05   15.30000        2        1.6       8.9        2
## 128065      3       1:52   18.30000        5        6.2       8.7        2
## 128066      4      11:28   19.80000        4        5.2       0.6        2
## 128067      4      11:10   23.00000        2        4.2      16.9        2
## 128068      4       2:37    9.10000        4        4.5      18.3        2
## 128069      4       0:12   12.45334        5        4.7       5.1        2
##        SHOT_RESULT CLOSEST_DEFENDER CLOSEST_DEFENDER_PLAYER_ID CLOSE_DEF_DIST
## 128064        made Sullinger, Jared                      203096            5.7
## 128065      missed    Smart, Marcus                      203935            0.8
## 128066        made     Turner, Evan                      202323            0.6
## 128067        made Thornton, Marcus                      201977            4.2
## 128068      missed   Bradley, Avery                      202340            3.0
## 128069        made   Bradley, Avery                      202340            2.3
##        FGM PTS  player_name player_id
## 128064   1   2 jarrett jack    101127
## 128065   0   0 jarrett jack    101127
## 128066   1   2 jarrett jack    101127
## 128067   1   2 jarrett jack    101127
## 128068   0   0 jarrett jack    101127
## 128069   1   2 jarrett jack    101127
```

Summary of entire NBA Shot logs data set.

```
summary(NBA_shotlogs) # data exploration function 5
```

```
##     GAME_ID           MATCHUP          LOCATION  W          FINAL_MARGIN
##  Min.   :21400001   Length:128069      A:64135   L:63474   Min.   :-53.0000
##  1st Qu.:21400233   Class :character   H:63934   W:64595   1st Qu.: -8.0000
##  Median :21400449   Mode  :character                       Median :  1.0000
##  Mean   :21400452                                         Mean   :  0.2087
##  3rd Qu.:21400673                                         3rd Qu.:  9.0000
##  Max.   :21400908                                         Max.   : 53.0000
##   SHOT_NUMBER         PERIOD       GAME_CLOCK          SHOT_CLOCK
##  Min.   : 1.000   Min.   :1.000   Length:128069      Min.   : 0.00
##  1st Qu.: 3.000   1st Qu.:1.000   Class :character   1st Qu.: 8.40
```

```
##  Median : 5.000   Median :2.000   Mode  :character   Median :12.45
##  Mean   : 6.507   Mean   :2.469                     Mean   :12.45
##  3rd Qu.: 9.000   3rd Qu.:3.000                     3rd Qu.:16.40
##  Max.   :38.000   Max.   :7.000                     Max.   :24.00
##     DRIBBLES        TOUCH_TIME         SHOT_DIST     PTS_TYPE  SHOT_RESULT
##  Min.   : 0.000   Min.   :-163.600   Min.   : 0.00   2:94173   made  :57905
##  1st Qu.: 0.000   1st Qu.:   0.900   1st Qu.: 4.70   3:33896   missed:70164
##  Median : 1.000   Median :   1.600   Median :13.70
##  Mean   : 2.023   Mean   :   2.766   Mean   :13.57
##  3rd Qu.: 2.000   3rd Qu.:   3.700   3rd Qu.:22.50
##  Max.   :32.000   Max.   :  24.900   Max.   :47.20
##  CLOSEST_DEFENDER   CLOSEST_DEFENDER_PLAYER_ID CLOSE_DEF_DIST
##  Length:128069      Min.   :   708             Min.   : 0.000
##  Class :character   1st Qu.:101249             1st Qu.: 2.300
##  Mode  :character   Median :201949             Median : 3.700
##                     Mean   :159039             Mean   : 4.123
##                     3rd Qu.:203079             3rd Qu.: 5.300
##                     Max.   :530027             Max.   :53.200
##       FGM              PTS         player_name          player_id
##  Min.   :0.0000   Min.   :0.0000   Length:128069      Min.   :   708
##  1st Qu.:0.0000   1st Qu.:0.0000   Class :character   1st Qu.:101162
##  Median :0.0000   Median :0.0000   Mode  :character   Median :201939
##  Mean   :0.4521   Mean   :0.9973                      Mean   :157238
##  3rd Qu.:1.0000   3rd Qu.:2.0000                      3rd Qu.:202704
##  Max.   :1.0000   Max.   :3.0000                      Max.   :204060
```

Data Exploration - Miscellaneous

```
maxDist <- max(NBA_shotlogs$SHOT_DIST)
print(paste("Farthest shot distance (in the data set): ", maxDist, " ft.")) # data exploration function
```

```
## [1] "Farthest shot distance (in the data set):  47.2  ft."
```

```
print(paste("Total Made Shots: ", sum(NBA_shotlogs$SHOT_RESULT=='made'))) # data exploration function 7
```

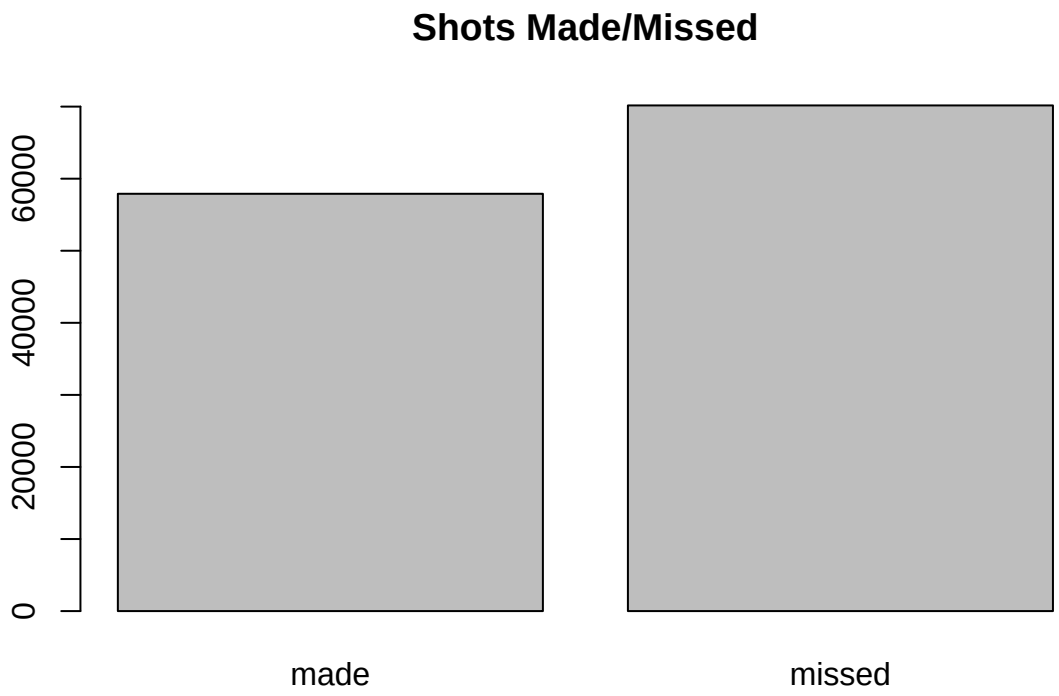
```
## [1] "Total Made Shots:  57905"
```

```
print(paste("Total Missed Shot: ", sum(NBA_shotlogs$SHOT_RESULT=='missed'))) # data exploration functio
```

```
## [1] "Total Missed Shot:  70164"
```

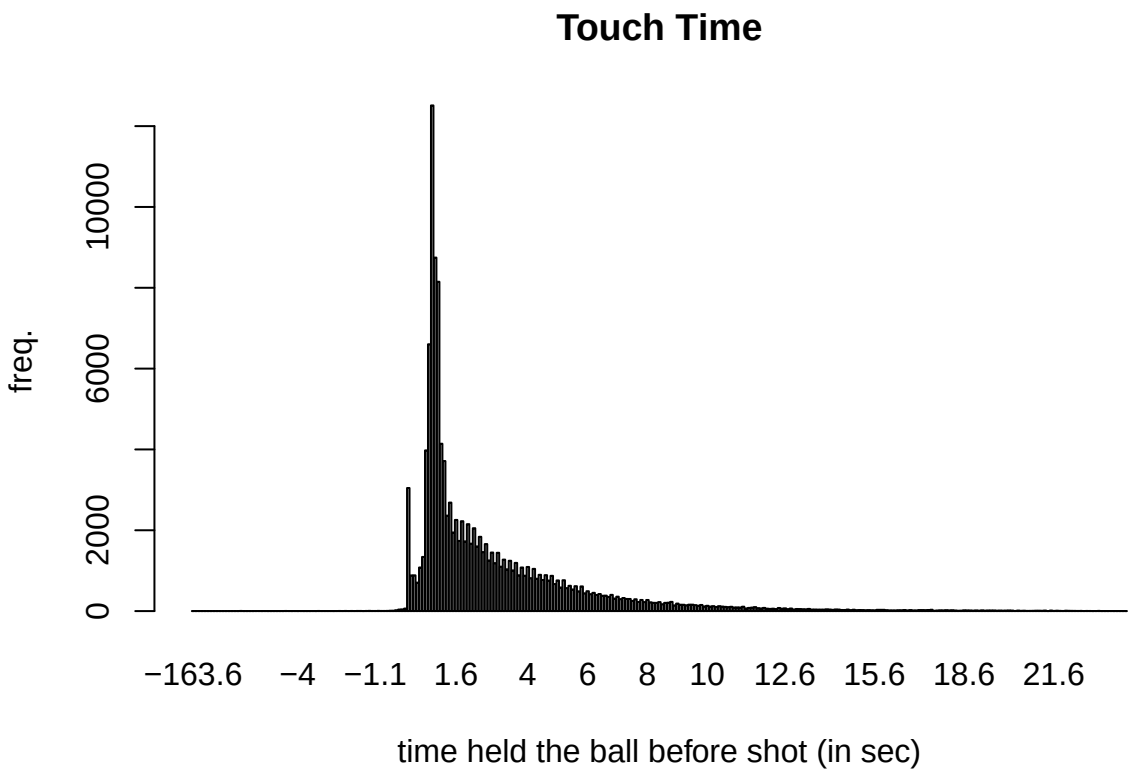
Data Visualization - Shot Made Column Distribution

```
plot(factor(NBA_shotlogs$SHOT_RESULT),
     main= "Shots Made/Missed") # data visual exploration function 1
```



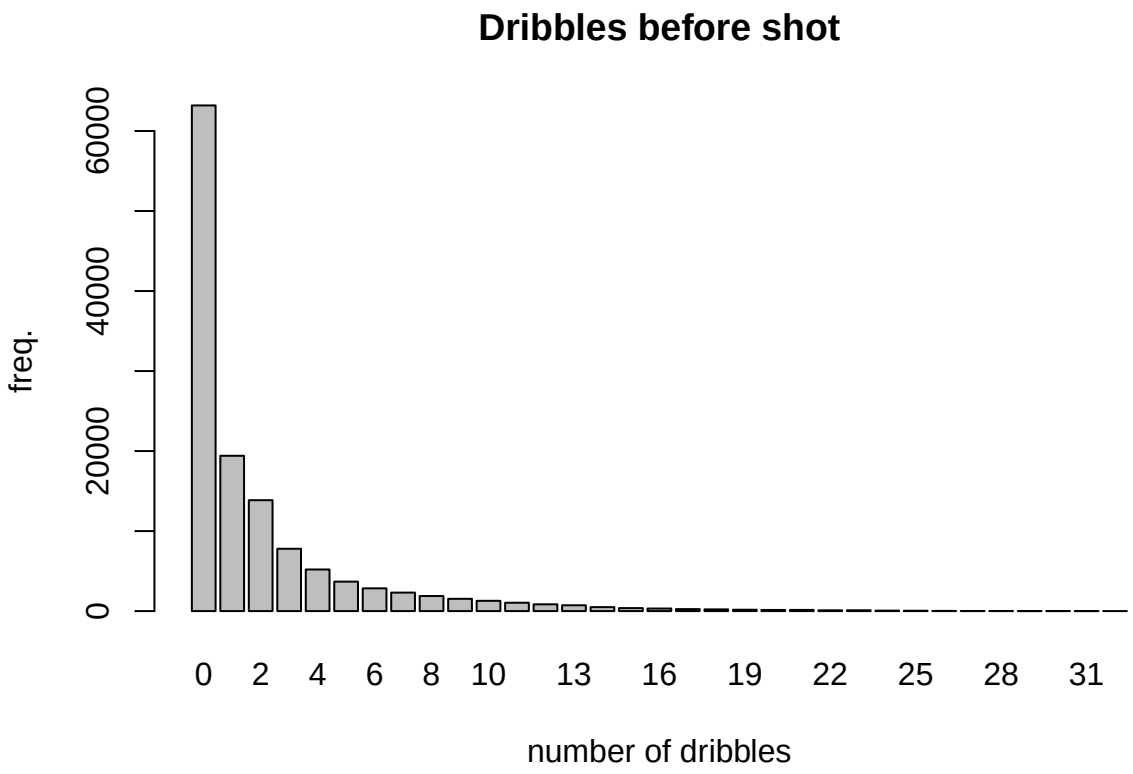
Data Visualization - Touch Time Column Distribution

```
plot(factor(NBA_shotlogs$TOUCH_TIME),
     main= "Touch Time",
     xlab= "time held the ball before shot (in sec)",
     ylab= "freq.")  # data visual exploration function 2
```



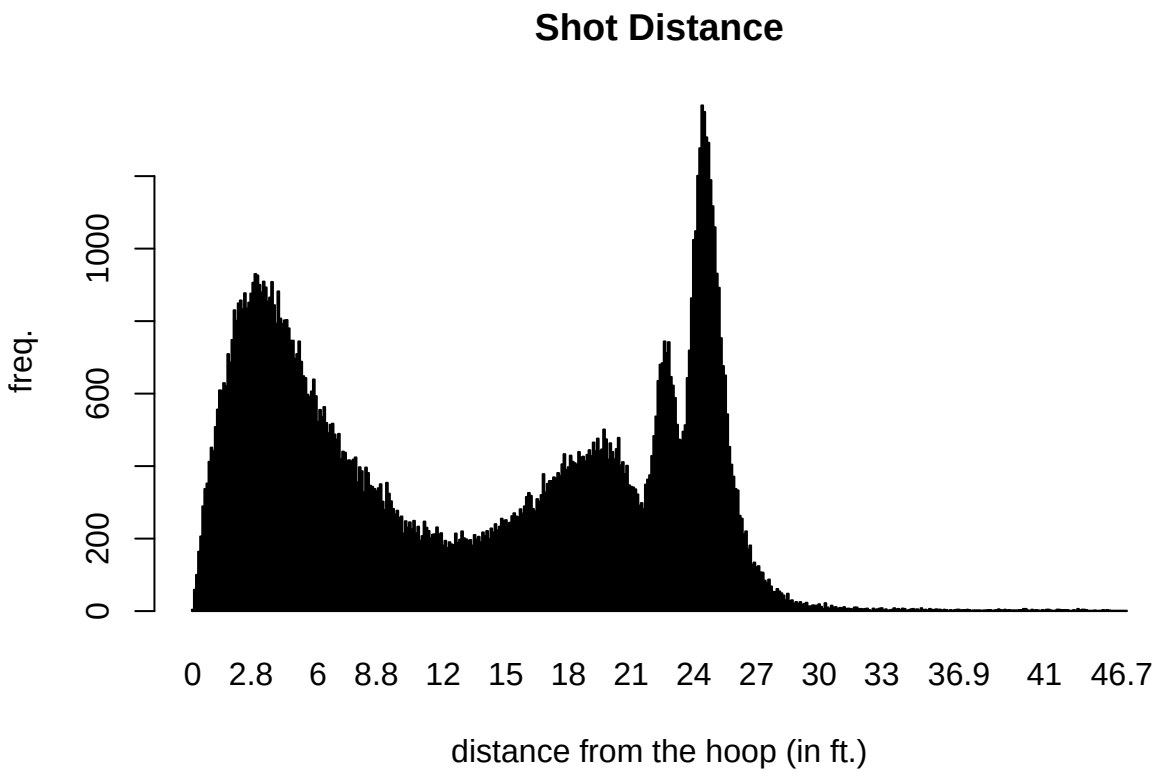
Data Visualization - Dribbles Column Distribution

```
plot(factor(NBA_shotlogs$DRIBBLES),
     main= "Dribbles before shot",
     xlab="number of dribbles",
     ylab= "freq.")  # data visual exploration function 3
```



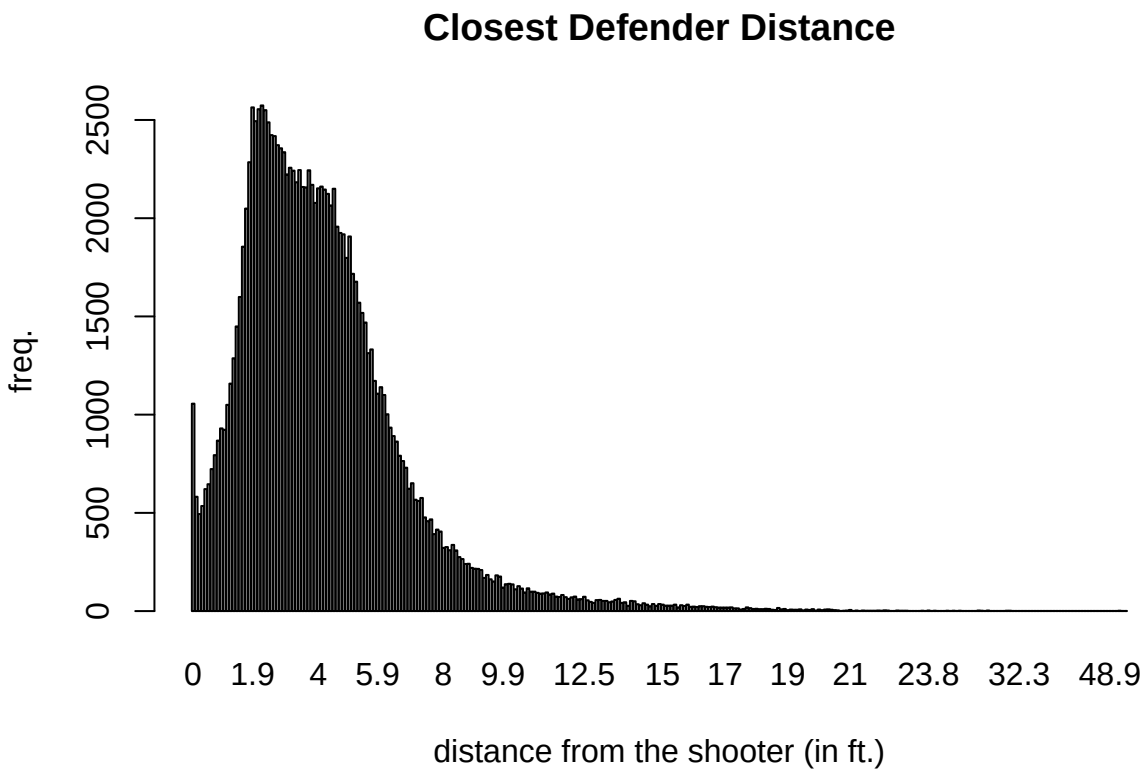
Data Visualization - Shot Distance Column Distribution

```
plot(factor(NBA_shotlogs$SHOT_DIST),
     main= "Shot Distance",
     xlab= "distance from the hoop (in ft.)",
     ylab= "freq.")  # data visual exploration function 4
```



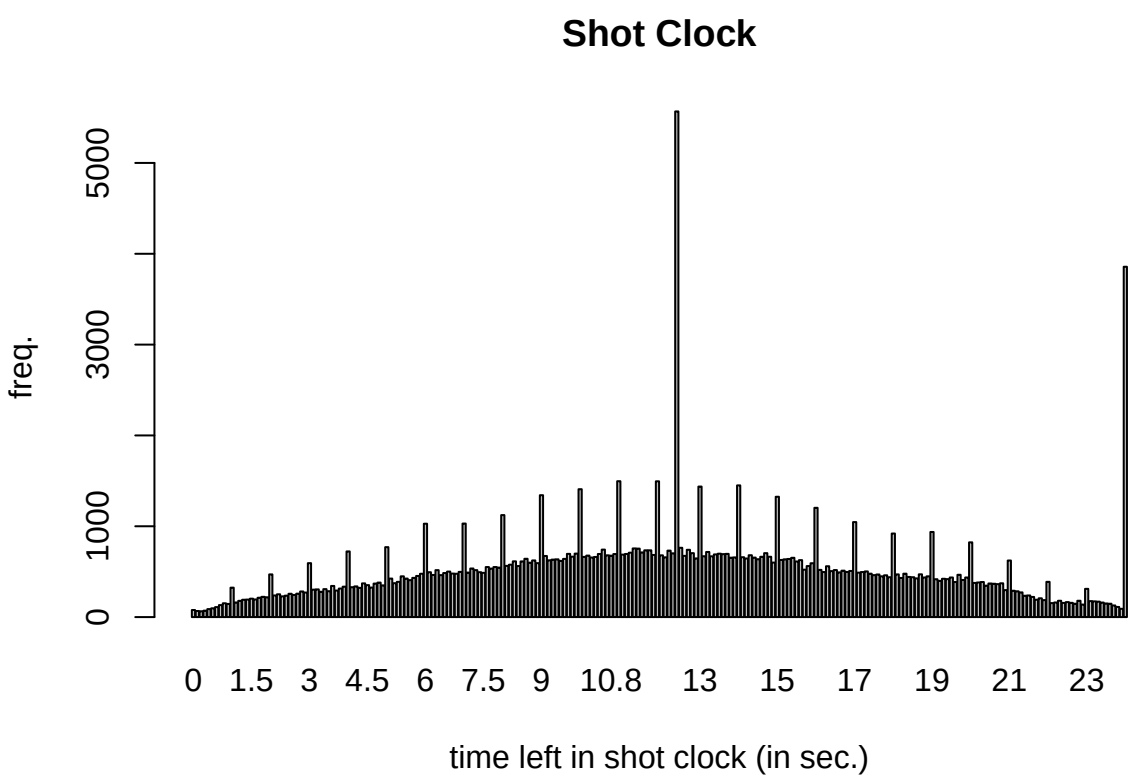
Data Visualization - Closest Defender Distance Column Distribution

```
plot(factor(NBA_shotlogs$CLOSE_DEF_DIST),
     main= "Closest Defender Distance",
     xlab= "distance from the shooter (in ft.)",
     ylab= "freq.")  # data visual exploration function 5
```



Data Visualization - Shot Clock Column Distribution

```
plot(factor(NBA_shotlogs$SHOT_CLOCK),
     main= "Shot Clock",
     xlab= "time left in shot clock (in sec.)",
     ylab= "freq.")  # data visual exploration function 5
```



Step 4.0 - Dividing into train and test sets.

Divide data randomly. 75% into train and 25% into test.

```
set.seed(1234)
i <- sample(1:nrow(NBA_shotlogs), nrow(NBA_shotlogs)*0.75, replace=FALSE)
train <- NBA_shotlogs[i,] # 75% in train
test <- NBA_shotlogs[-i,] # 25% in test
```

Step 4.1 - Logistic Regression Model

Logistic Regression Model exhibits good probabilistic output and is computationally inexpensive. The simplicity of Logistic regression and effectiveness only works optimally on linear data. With However, it can prove to have poor performance on nonlinear data.

```
glm <- glm(SHOT_RESULT ~ TOUCH_TIME + DRIBBLES + SHOT_DIST + CLOSE_DEF_DIST + SHOT_CLOCK, data=train, fa
summary(glm)
```

```
##
## Call:
## glm(formula = SHOT_RESULT ~ TOUCH_TIME + DRIBBLES + SHOT_DIST +
##     CLOSE_DEF_DIST + SHOT_CLOCK, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1057  -1.1596   0.8241   1.0718   3.8508
##
```

```
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -0.1164570  0.0244669  -4.760 1.94e-06 ***
## TOUCH_TIME      0.0494646  0.0061910   7.990 1.35e-15 ***
## DRIBBLES       -0.0200500  0.0052664  -3.807 0.000141 ***
## SHOT_DIST       0.0611653  0.0009534  64.152  < 2e-16 ***
## CLOSE_DEF_DIST -0.1009876  0.0031247 -32.319  < 2e-16 ***
## SHOT_CLOCK     -0.0158466  0.0012377 -12.803  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 132318  on 96050  degrees of freedom
## Residual deviance: 126886  on 96045  degrees of freedom
## AIC: 126898
##
## Number of Fisher Scoring iterations: 4
```

Logistic Regression Model Evaluation on Test

```
probsglm <- predict(glm, newdata=test, type="response")
predsglm <- ifelse(probsglm > 0.5, "missed", "made")

library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
confusionMatrix(factor(predsglm), factor(test$SHOT_RESULT))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  made missed
##     made    6900   5043
##     missed  7460  12615
##
##                Accuracy : 0.6095
##                  95% CI : (0.6041, 0.6149)
##     No Information Rate : 0.5515
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.198
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4805
##             Specificity : 0.7144
##          Pos Pred Value : 0.5777
##          Neg Pred Value : 0.6284
##              Prevalence : 0.4485
##          Detection Rate : 0.2155
##    Detection Prevalence : 0.3730
##       Balanced Accuracy : 0.5975
```

```
##
##        'Positive' Class : made
##
```

```
accglm <- mean(predsglm == test$SHOT_RESULT)
print(paste("Accuracy Logistic Regression: ", accglm))
```

```
## [1] "Accuracy Logistic Regression:  0.609500905740521"
```

Logistic Regression Model resulted in an accuracy of 0.6095, Sensitivity of 0.4805, and Specificity of 0.7144. The accuracy being moderate rather than strong clearly roots from the models poor performance because of the data's nonlinearity.

Step 4.2 - Naive Bayes Model

I selected the Naive Model for its real-time predicting as it can prove to be very fast in terms of performance but is weak when predictors are not independent (naive assumption). I, also, selected Naive Bayes as to act as a good **baseline** in comparing performances of other classification algorithms.

```
library(e1071)
nb <- naiveBayes(SHOT_RESULT ~ TOUCH_TIME + DRIBBLES + SHOT_DIST + CLOSE_DEF_DIST + SHOT_CLOCK, data=tra
```

Naive Bayes Model Evaluation on Test

```
p1 <- predict(nb, newdata=test, type="class")

library(caret)
table_nb <- confusionMatrix(p1, test$SHOT_RESULT)
table_nb
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  made missed
##     made    7067   5716
##     missed  7293  11942
##
##                Accuracy : 0.5937
##                  95% CI : (0.5883, 0.5991)
##     No Information Rate : 0.5515
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1702
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4921
##             Specificity : 0.6763
##          Pos Pred Value : 0.5528
##          Neg Pred Value : 0.6208
##              Prevalence : 0.4485
##          Detection Rate : 0.2207
##    Detection Prevalence : 0.3992
##       Balanced Accuracy : 0.5842
##
##        'Positive' Class : made
##
```

```
accnb <- mean(p1 == test$SHOT_RESULT)
print(paste("Naive Bayes Accuracy: ", accnb))
```

```
## [1] "Naive Bayes Accuracy:  0.59369729527141"
```

Naive Bayes Model resulted in an accuracy of 0.5937, Sensitivity of 0.4921, and Specificity of 0.6763. Here, the accuracy was really close to that of Logistic Regression. Poor performance of this model can be attributed to the naive assumption of predictors.

Step 4.3 - kNN Classification Model

I selected the kNN Classification Model for its advantages in making no assumptions about the shape of the data. However, because I am using it with high dimensions, it could possibly suffer from poor performance.

```
set.seed(1234)

library(caret)

train_knn <- NBA_shotlogs[i, c(11,10,12,17,9)]
test_knn <- NBA_shotlogs[-i, c(11,10,12,17,9)]
trainlabels_knn <- NBA_shotlogs[i, 14]
testlabels_knn <- NBA_shotlogs[-i, 14]

library(class)
knn_pred <- knn(train=train_knn, test=test_knn, cl=trainlabels_knn, k=1)


# kNN Classification Model Evaluation on Test
results_knn <- (knn_pred == testlabels_knn)
accknn <- length(which(results_knn ==TRUE)) / length(results_knn)
table(factor(knn_pred), factor(testlabels_knn))
```

```
##
##           made missed
##   made    7195   7441
##   missed  7165  10217
```

```
table_knn <- confusionMatrix(knn_pred, test$SHOT_RESULT)
table_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  made missed
##     made    7195   7441
##     missed  7165  10217
##
##                Accuracy : 0.5438
##                  95% CI : (0.5383, 0.5493)
##     No Information Rate : 0.5515
##     P-Value [Acc > NIR] : 0.99719
##
##                   Kappa : 0.0795
##
##  Mcnemar's Test P-Value : 0.02288
##
```

```
##             Sensitivity : 0.5010
##             Specificity : 0.5786
##          Pos Pred Value : 0.4916
##          Neg Pred Value : 0.5878
##              Prevalence : 0.4485
##          Detection Rate : 0.2247
##    Detection Prevalence : 0.4571
##       Balanced Accuracy : 0.5398
##
##        'Positive' Class : made
##
```

```
print(paste("kNN Classification Accuracy: ", accknn))
```

```
## [1] "kNN Classification Accuracy:  0.543819101755263"
```

kNN Classification Model resulted in an accuracy of 0.5438, Sensitivity of 0.5010, and Specificity of 0.5786. In the case of kNN classification, the accuracy proved to be lower than both logistic regression and naive bayes. This is most likely due to the high dimensions and un-scaled data when creating the model.

Step 5 - Results Analysis

Metrics Comparison: **Accuracy:** - Logistic Regression: 0.6095 - Naive Bayes: 0.5937 - kNN Classification: 0.5438

Sensitivity (True Positive Rate): - Logistic Regression: 0.4805
- Naive Bayes: 0.4921 - kNN Classification: 0.5010

Specificity (True Negative Rate): - Logistic Regression: 0.7144 - Naive Bayes: 0.6763 - kNN Classification: 0.5786

Overall, it seems the Logistic Regression did the best in terms of accuracy. However, Naive Bayes was right behind by thousandths. kNN Classification lagged behind Logistic and Naive Bayes even though it didn't make assumptions about the data which was surprising. It's important to note that even though Logistic Regression achieved the best accuracy and specificity, kNN had the best sensitivity when evaluated on the test data.

As stated, Logistic Regression Model exhibits the strengths of being computationally inexpensive and good probabilistic output. Furthermore, it is able to separate classes well as long as they can be linearly separable. The data set most likely performed better with Logistic Regression Model, due to the fact that its disadvantages was less impactful to the disadvantages of Naive Bayes and kNN Classification.

The script was able to identify a moderate accuracy between the target, SHOT_RESULT, and the predictors,TOUCH_TIME, DRIBBLES, SHOT_DIST, CLOSE_DEF_DIST, and SHOT_CLOCK. These findings will most likely not be too useful outside of learning from this specific set of data, however, if proper measures were taken such as selecting fewer dimensions or scaling for kNN, would likely increase effectiveness of models and accuracy of the evaluation.