# R Notebook

Code

Brandon Runyon 9/25/22

Data taken from Kaggle (https://www.kaggle.com/datasets/aliibrahim10/valorant-stats). Data taken from the leader board for the game VALORANT.

Players who perform better each round are more likely to win. Performance can be measured with variables like damage dealt or kills.

I will use this data of overall statistics for the top players during a single season of the game to try to predict their win rates.

# Load and preprocess

Load “val_stats.csv”

Hide

```
df <- read.csv("val_stats.csv")
str(df)
```

```
'data.frame':   85678 obs. of  38 variables:
 $ region          : chr  NA NA NA NA ...
 $ name            : chr  "ShimmyXD" "XSET Cryo" "PuRelittleone" "Boba" ...
 $ tag             : chr  "#NA1" "#cells" "#yoruW" "#0068" ...
 $ rating          : chr  "Radiant" "Radiant" "Radiant" "Radiant" ...
 $ damage_round    : num  136 170 148 178 150 ...
 $ headshots       : chr  "992" "879" "720" "856" ...
 $ headshot_percent: num  24.9 28.3 24 37.3 24.4 26 25.2 17.5 24.6 20.8 ...
 $ aces            : int  0 2 3 3 2 2 7 2 2 1 ...
 $ clutches        : int  140 122 117 83 71 162 186 112 189 56 ...
 $ flawless        : int  80 94 59 49 38 94 92 64 132 44 ...
 $ first_bloods    : chr  "161" "316" "216" "235" ...
 $ kills           : chr  "1,506" "1,608" "1,115" "1,134" ...
 $ deaths          : chr  "1,408" "1,187" "1,064" "812" ...
 $ assists         : chr  "703" "206" "267" "157" ...
 $ kd_ratio        : num  1.07 1.35 1.05 1.4 1.11 1.03 1.16 1.17 1.31 1.37 ...
 $ kills_round     : num  0.7 1 0.8 1 0.8 0.7 0.9 0.8 0.9 0.9 ...
 $ most_kills      : int  29 32 39 37 29 33 37 36 38 29 ...
 $ score_round     : num  209 271 228 277 231 ...
 $ wins            : int  59 52 42 32 32 57 69 44 88 29 ...
 $ win_percent     : num  59.6 65.8 65.6 62.8 62.8 58.2 55.6 62 56.8 69 ...
 $ agent_1         : chr  "Fade" "Chamber" "Yoru" "Jett" ...
 $ agent_2         : chr  "Viper" "Jett" "Jett" "Chamber" ...
 $ agent_3         : chr  "Omen" "Raze" "Chamber" "KAY/O" ...
 $ gun1_name       : chr  "Vandal" "Vandal" "Vandal" "Vandal" ...
 $ gun1_head       : int  35 41 38 51 36 40 35 26 8 29 ...
 $ gun1_body       : int  59 56 57 47 60 55 61 67 91 66 ...
 $ gun1_legs       : int  5 3 4 2 4 5 4 7 1 5 ...
 $ gun1_kills      : chr  "802" "689" "444" "754" ...
 $ gun2_name       : chr  "Phantom" "Operator" "Phantom" "Sheriff" ...
 $ gun2_head       : int  33 8 36 48 21 35 32 6 36 2 ...
 $ gun2_body       : int  62 91 61 51 71 62 64 93 59 98 ...
 $ gun2_legs       : int  5 0 3 1 8 3 5 1 5 1 ...
 $ gun2_kills      : chr  "220" "226" "231" "48" ...
 $ gun3_name       : chr  "Classic" "Phantom" "Operator" "Phantom" ...
 $ gun3_head       : int  36 32 8 44 8 29 48 35 22 23 ...
 $ gun3_body       : int  60 63 91 56 92 65 50 65 69 70 ...
 $ gun3_legs       : int  3 5 1 0 0 6 2 0 9 6 ...
 $ gun3_kills      : int  147 137 102 36 64 135 253 100 34 85 ...
```

Check the number of NAs in each column.

Hide

```
sapply(df, function(x) sum(is.na(x)))
```

```
          region             name              tag           rating     damage_round        head
shots headshot_percent             aces         clutches
           20865                0                 0                0                0
0                0                0                 0
        flawless     first_bloods            kills           deaths          assists         kd_
ratio      kills_round       most_kills      score_round
               0                0                 0                0                0
0                0                0                 0
            wins      win_percent          agent_1          agent_2          agent_3        gun1
_name        gun1_head        gun1_body        gun1_legs
               0                0                 0                0                0
0                0                0                 0
      gun1_kills        gun2_name        gun2_head        gun2_body        gun2_legs       gun2_
kills        gun3_name        gun3_head        gun3_body
               0                0                 0                0                0
0                0                0                 0
       gun3_legs       gun3_kills
               0                0
```

NA in region refers to North America, so no NA data, but needs to be handled for factors.

Convert columns to factors

Hide

```
cols <- c("region", "rating", "agent_1", "agent_2", "agent_3", "gun1_name", "gun2_name", "gun3_n
ame")
df[cols] <- lapply(df[cols], factor, exclude = NULL)
```

Some columns are read as strings instead of numbers due to commas.

Hide

```


cols <- c("headshots", "first_bloods", "kills", "deaths", "assists", "gun1_kills", "gun2_kills")
df[cols] <- lapply(df[cols], gsub, pattern = ",", replacement = "")
df[cols] <- lapply(df[cols], as.numeric)
```

View new data

Hide

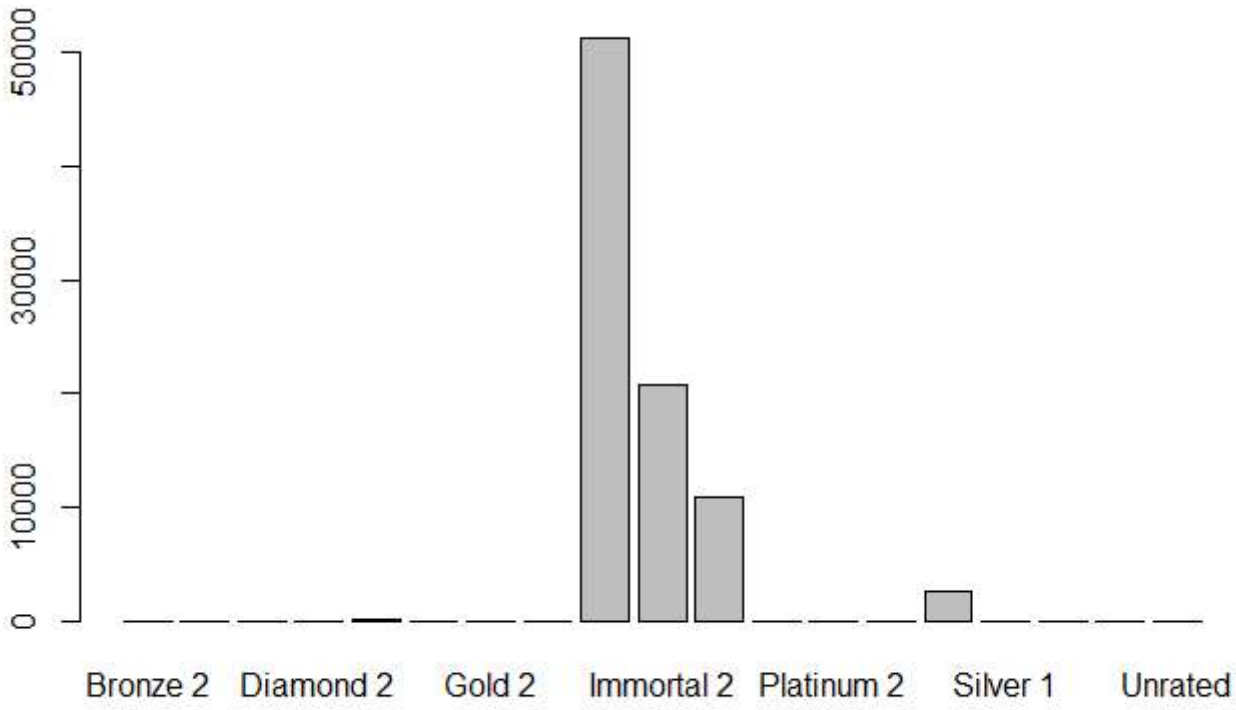
```
str(df)
```

```
'data.frame':   85678 obs. of  38 variables:
 $ region          : Factor w/ 6 levels "AP","BR","EU",..: 6 6 6 6 6 6 6 6 6 6 ...
 $ name            : chr  "ShimmyXD" "XSET Cryo" "PuRelittleone" "Boba" ...
 $ tag             : chr  "#NA1" "#cells" "#yoruW" "#0068" ...
 $ rating          : Factor w/ 19 levels "Bronze 2","Bronze 3",..: 15 15 15 15 15 15 15 15 15 15
...
 $ damage_round    : num  136 170 148 178 150 ...
 $ headshots       : num  992 879 720 856 534 ...
 $ headshot_percent: num  24.9 28.3 24 37.3 24.4 26 25.2 17.5 24.6 20.8 ...
 $ aces            : int  0 2 3 3 2 2 7 2 2 1 ...
 $ clutches        : int  140 122 117 83 71 162 186 112 189 56 ...
 $ flawless        : int  80 94 59 49 38 94 92 64 132 44 ...
 $ first_bloods    : num  161 316 216 235 137 179 311 215 515 103 ...
 $ kills           : num  1506 1608 1115 1134 869 ...
 $ deaths          : num  1408 1187 1064 812 781 ...
 $ assists         : num  703 206 267 157 213 629 614 341 440 140 ...
 $ kd_ratio        : num  1.07 1.35 1.05 1.4 1.11 1.03 1.16 1.17 1.31 1.37 ...
 $ kills_round     : num  0.7 1 0.8 1 0.8 0.7 0.9 0.8 0.9 0.9 ...
 $ most_kills      : int  29 32 39 37 29 33 37 36 38 29 ...
 $ score_round     : num  209 271 228 277 231 ...
 $ wins            : int  59 52 42 32 32 57 69 44 88 29 ...
 $ win_percent     : num  59.6 65.8 65.6 62.8 62.8 58.2 55.6 62 56.8 69 ...
 $ agent_1         : Factor w/ 19 levels "Astra","Breach",..: 6 4 19 7 7 6 7 7 4 4 ...
 $ agent_2         : Factor w/ 20 levels "","Astra","Breach",..: 19 8 8 5 14 18 7 5 8 14 ...
 $ agent_3         : Factor w/ 20 levels "","Astra","Breach",..: 12 14 5 9 5 5 9 18 9 16 ...
 $ gun1_name       : Factor w/ 16 levels "Ares","Bucky",..: 16 16 16 16 16 16 16 16 11 12 ...
 $ gun1_head       : int  35 41 38 51 36 40 35 26 8 29 ...
 $ gun1_body       : int  59 56 57 47 60 55 61 67 91 66 ...
 $ gun1_legs       : int  5 3 4 2 4 5 4 7 1 5 ...
 $ gun1_kills      : num  802 689 444 754 419 ...
 $ gun2_name       : Factor w/ 18 levels "Ares","Bucky",..: 13 12 13 14 16 13 13 12 18 12 ...
 $ gun2_head       : int  33 8 36 48 21 35 32 6 36 2 ...
 $ gun2_body       : int  62 91 61 51 71 62 64 93 59 98 ...
 $ gun2_legs       : int  5 0 3 1 8 3 5 1 5 1 ...
 $ gun2_kills      : num  220 226 231 48 65 144 369 318 626 163 ...
 $ gun3_name       : Factor w/ 18 levels "Ares","Bucky",..: 4 13 12 13 12 16 14 9 13 16 ...
 $ gun3_head       : int  36 32 8 44 8 29 48 35 22 23 ...
 $ gun3_body       : int  60 63 91 56 92 65 50 65 69 70 ...
 $ gun3_legs       : int  3 5 1 0 0 6 2 0 9 6 ...
 $ gun3_kills      : int  147 137 102 36 64 135 253 100 34 85 ...
```

Most data is from the top few players (those labeled with Immortal or Radiant).

Hide

```
counts <- table(df$rating)
barplot(counts)
```




Remove all players who aren't Immortal 1/2/3, or Radiant and refactor.

Hide

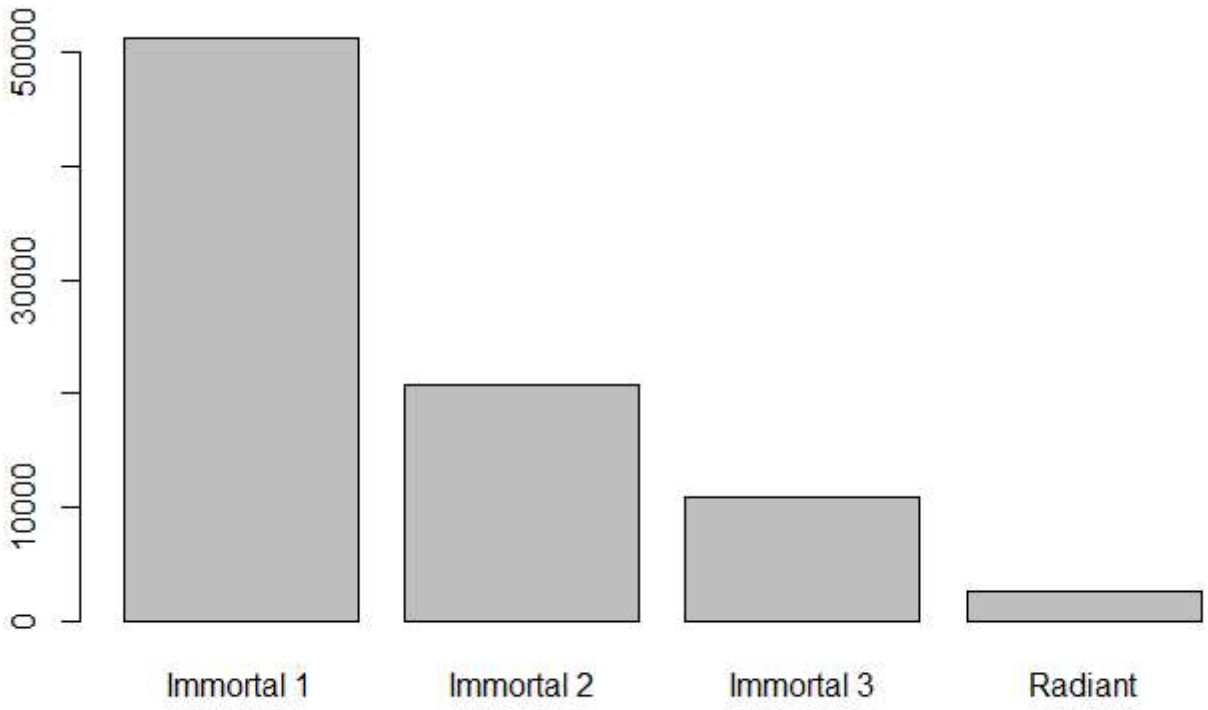
```
i <- which(df$rating == "Immortal 1" | df$rating == "Immortal 2" | df$rating == "Immortal 3" | d
f$rating == "Radiant")
df <- df[i,]
df$rating <- factor(df$rating)
str(df)
```

```
'data.frame':   85574 obs. of  38 variables:
 $ region          : Factor w/ 6 levels "AP","BR","EU",..: 6 6 6 6 6 6 6 6 6 6 ...
 $ name            : chr  "ShimmyXD" "XSET Cryo" "PuRelittleone" "Boba" ...
 $ tag             : chr  "#NA1" "#cells" "#yoruW" "#0068" ...
 $ rating          : Factor w/ 4 levels "Immortal 1","Immortal 2",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ damage_round    : num  136 170 148 178 150 ...
 $ headshots       : num  992 879 720 856 534 ...
 $ headshot_percent: num  24.9 28.3 24 37.3 24.4 26 25.2 17.5 24.6 20.8 ...
 $ aces            : int  0 2 3 3 2 2 7 2 2 1 ...
 $ clutches        : int  140 122 117 83 71 162 186 112 189 56 ...
 $ flawless        : int  80 94 59 49 38 94 92 64 132 44 ...
 $ first_bloods    : num  161 316 216 235 137 179 311 215 515 103 ...
 $ kills           : num  1506 1608 1115 1134 869 ...
 $ deaths          : num  1408 1187 1064 812 781 ...
 $ assists         : num  703 206 267 157 213 629 614 341 440 140 ...
 $ kd_ratio        : num  1.07 1.35 1.05 1.4 1.11 1.03 1.16 1.17 1.31 1.37 ...
 $ kills_round     : num  0.7 1 0.8 1 0.8 0.7 0.9 0.8 0.9 0.9 ...
 $ most_kills      : int  29 32 39 37 29 33 37 36 38 29 ...
 $ score_round     : num  209 271 228 277 231 ...
 $ wins            : int  59 52 42 32 32 57 69 44 88 29 ...
 $ win_percent     : num  59.6 65.8 65.6 62.8 62.8 58.2 55.6 62 56.8 69 ...
 $ agent_1         : Factor w/ 19 levels "Astra","Breach",..: 6 4 19 7 7 6 7 7 4 4 ...
 $ agent_2         : Factor w/ 20 levels "","Astra","Breach",..: 19 8 8 5 14 18 7 5 8 14 ...
 $ agent_3         : Factor w/ 20 levels "","Astra","Breach",..: 12 14 5 9 5 5 9 18 9 16 ...
 $ gun1_name       : Factor w/ 16 levels "Ares","Bucky",..: 16 16 16 16 16 16 16 16 11 12 ...
 $ gun1_head       : int  35 41 38 51 36 40 35 26 8 29 ...
 $ gun1_body       : int  59 56 57 47 60 55 61 67 91 66 ...
 $ gun1_legs       : int  5 3 4 2 4 5 4 7 1 5 ...
 $ gun1_kills      : num  802 689 444 754 419 ...
 $ gun2_name       : Factor w/ 18 levels "Ares","Bucky",..: 13 12 13 14 16 13 13 12 18 12 ...
 $ gun2_head       : int  33 8 36 48 21 35 32 6 36 2 ...
 $ gun2_body       : int  62 91 61 51 71 62 64 93 59 98 ...
 $ gun2_legs       : int  5 0 3 1 8 3 5 1 5 1 ...
 $ gun2_kills      : num  220 226 231 48 65 144 369 318 626 163 ...
 $ gun3_name       : Factor w/ 18 levels "Ares","Bucky",..: 4 13 12 13 12 16 14 9 13 16 ...
 $ gun3_head       : int  36 32 8 44 8 29 48 35 22 23 ...
 $ gun3_body       : int  60 63 91 56 92 65 50 65 69 70 ...
 $ gun3_legs       : int  3 5 1 0 0 6 2 0 9 6 ...
 $ gun3_kills      : int  147 137 102 36 64 135 253 100 34 85 ...
```

Same table now shows the top players only.

Hide

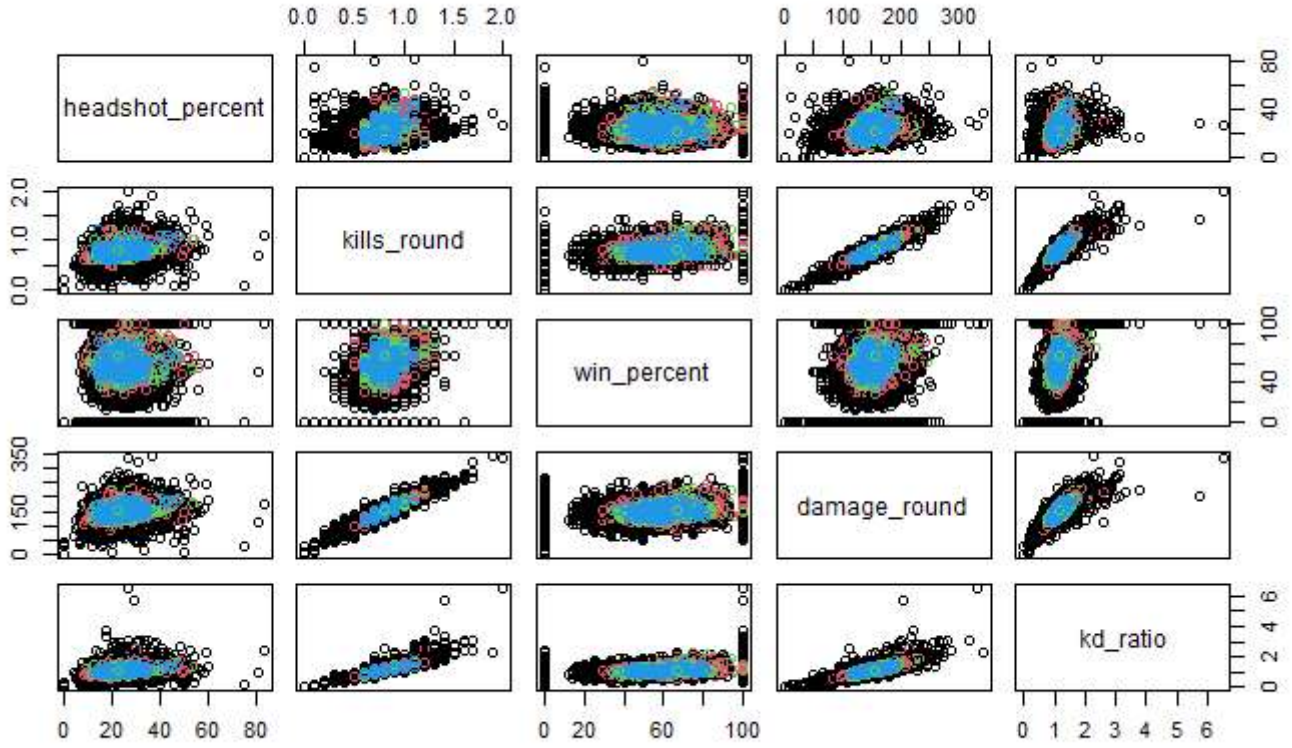
```
counts <- table(df$rating)
barplot(counts)
```




# Basic analysis of the data

Hide

```
pairs(df[,c("headshot_percent","kills_round","win_percent", "damage_round", "kd_ratio")], col=df
$rating)
```




Win percent has several with 0% or 100% win rate, which is unrealistic. These are likely players who only played the 1 required game to get their rank for the season.

These data values are also very tightly packed regardless of rating. They only seem to get tighter by rating.

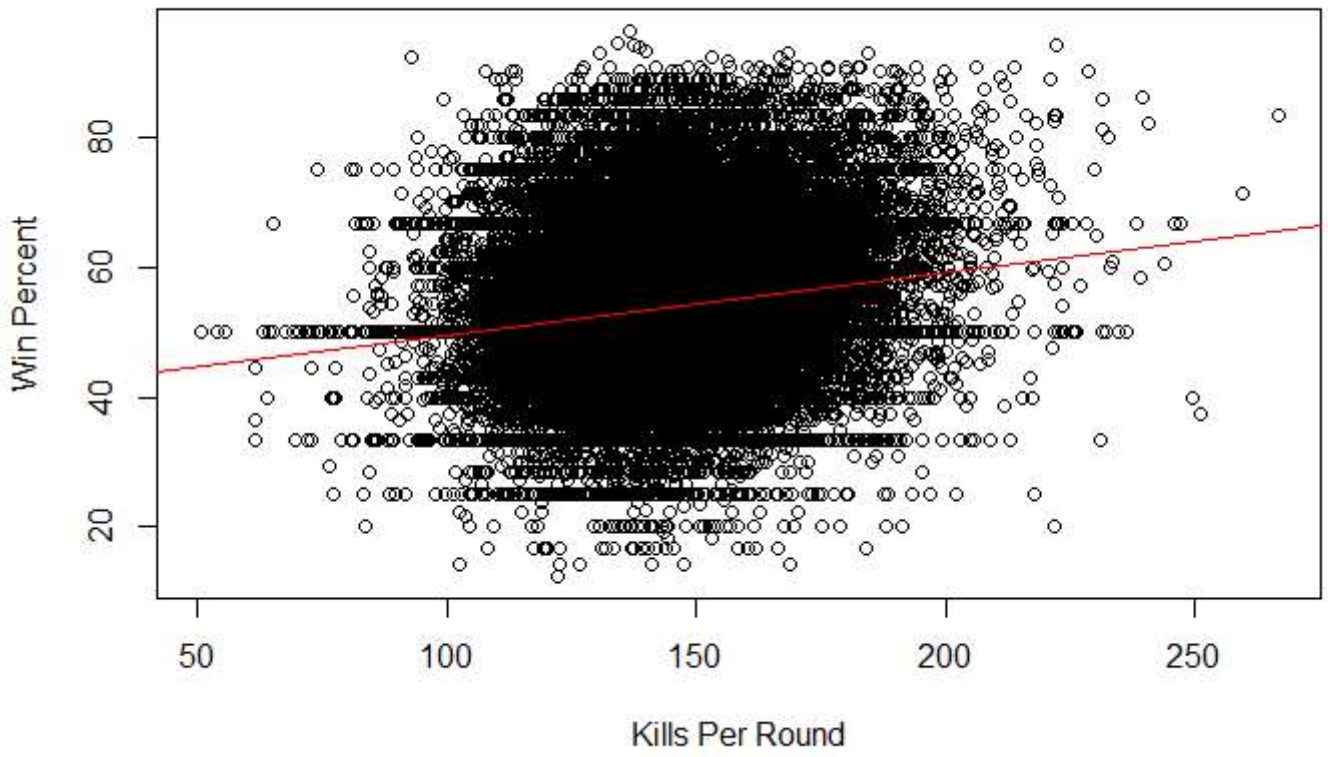
Hide

```
i <- which(df$win_percent == 0 | df$win_percent == 100)
df <- df[-i,]
```

# Linear Regression

Hide

```
plot(df$win_percent~df$damage_round, xlab="Kills Per Round", ylab="Win Percent")
abline(lm(df$win_percent~df$damage_round), col="red")
```




Just by graphing, it doesn't appear as though using just one variable is a good representation of win rate.

Build the regression model to see how accurate it is.

Allocate test and train data.

Hide

```
set.seed(1)
i <- sample(1:nrow(df), 0.8*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

# Single Variable Linear Regression

Run linear regression on a single variable

Hide

```
lm1 <- lm(win_percent~damage_round, data=train)
lm1
```

```
Call:
lm(formula = win_percent ~ damage_round, data = train)

Coefficients:
 (Intercept)  damage_round
    39.64389       0.09805
```

Measure covariance

Hide

```
pred <- predict(lm1, newdata=test)
cov(pred, test$win_percent) / (sd(pred)*sd(test$win_percent))
```

```
[1] 0.1577273
```

Low covariance indicates this model wasn't good at predicting the win rates.

Hide

```
summary(lm1)
```

```
Call:
lm(formula = win_percent ~ damage_round, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-40.995  -5.500  -0.735   4.906  43.538

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  39.643892   0.319525  124.07   <2e-16 ***
damage_round  0.098049   0.002225   44.07   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.966 on 67107 degrees of freedom
Multiple R-squared:  0.02813,   Adjusted R-squared:  0.02812
F-statistic:  1942 on 1 and 67107 DF,  p-value: < 2.2e-16
```

Very low R-squared shows it wasn't a good model for this data.

Calculate other statistics to evaluate.

Hide

```
correlation <- cor(pred, test$win_percent)
print(paste("correlation: ", correlation))
```

```
[1] "correlation:  0.15772726140813"
```

Hide

```
mse <- mean((pred - test$win_percent)^2)
print(paste("mse: ", mse))
```

```
[1] "mse:  82.2696661494788"
```

Hide

```
rmse <- sqrt(mse)
print(paste("rmse: ", rmse))
```

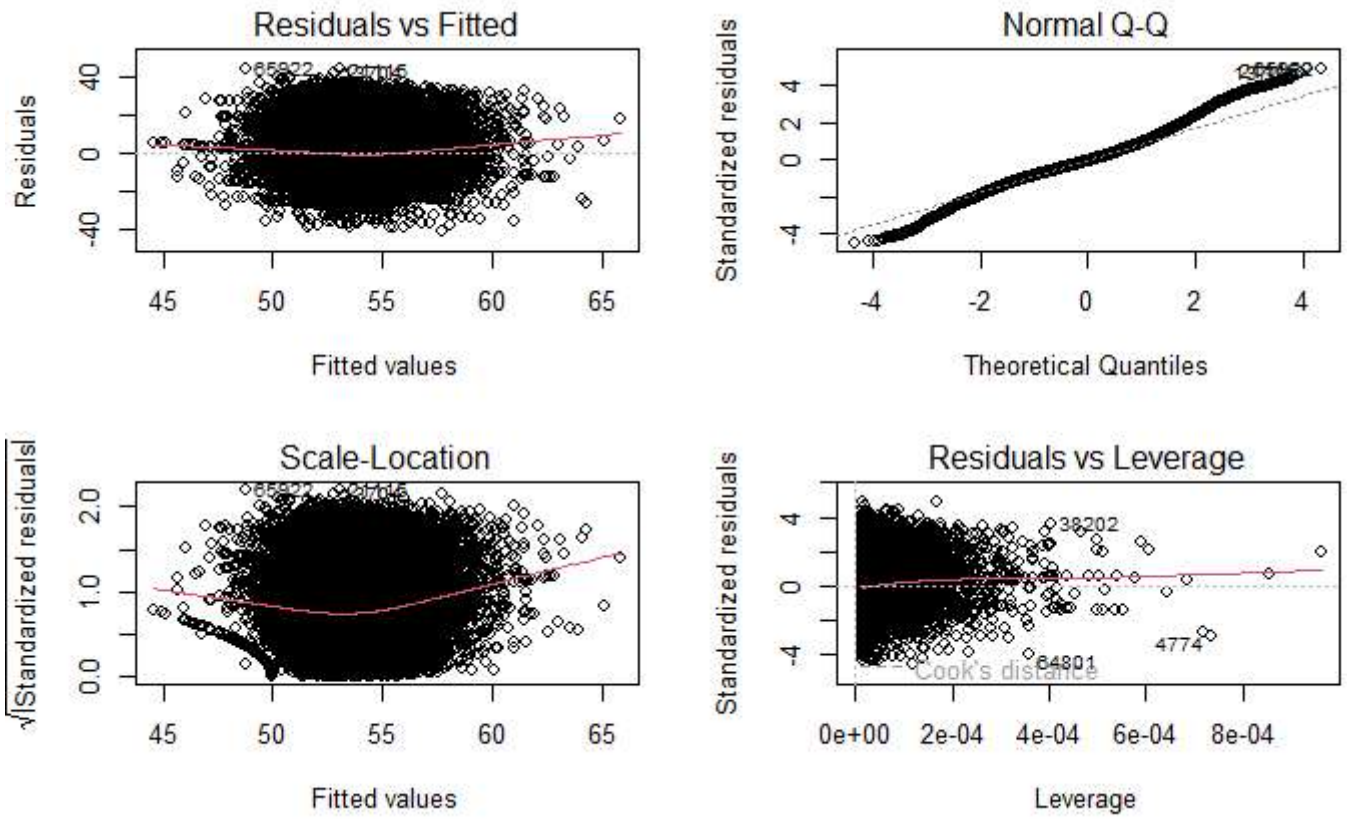
```
[1] "rmse:  9.07026273872366"
```

These statistics further indicate a bad model.

Plot the residuals to visualize the accuracy of the model.

Hide

```
par(mfrow=c(2,2))
plot(lm1)
```




This shows that the values are not well predicted due to the nature of the data.

# Multiple Variable Linear Regression

Hide

```
lm2 <- lm(win_percent~damage_round+kills_round+headshot_percent+kd_ratio+score_round, data=trai
n)
summary(lm2)
```

```

Call:
lm(formula = win_percent ~ damage_round + kills_round + headshot_percent +
    kd_ratio + score_round, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-43.047  -5.162  -0.509   4.637  40.907

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      40.779692   0.312070 130.675   <2e-16 ***
damage_round      0.100239   0.011822   8.479   <2e-16 ***
kills_round      -8.568455   1.042411  -8.220   <2e-16 ***
headshot_percent -0.121186   0.006813 -17.787   <2e-16 ***
kd_ratio         41.764956   0.450064  92.798   <2e-16 ***
score_round      -0.165170   0.008460 -19.523   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.396 on 67103 degrees of freedom
Multiple R-squared:  0.1479,    Adjusted R-squared:  0.1479
F-statistic:  2330 on 5 and 67103 DF,  p-value: < 2.2e-16
```

Run anova()

Hide

```
anova(lm1, lm2)
```

```
Analysis of Variance Table

Model 1: win_percent ~ damage_round
Model 2: win_percent ~ damage_round + kills_round + headshot_percent +
    kd_ratio + score_round
  Res.Df     RSS Df Sum of Sq      F    Pr(>F)
1  67107 5395171
2  67103 4730191  4    664980 2358.4 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

This shows that using multiple variables is better, though still not perfect.

Hide

```
pred2 <- predict(lm2, newdata=test)
cor2 <- cor(pred2, test$win_percent)
mse2 <- mean((pred2-test$win_percent)^2)
rmse2 <- sqrt(mse2)
print(paste('correlation:', cor2))
```

```
[1] "correlation: 0.38131034485929"
```

Hide

```
print(paste('mse:', mse2))
```

```
[1] "mse: 72.0973816750975"
```

Hide

```
print(paste('rmse:', rmse2))
```

```
[1] "rmse: 8.491017705499"
```

More evaluations of the second model showing slight improvement from the first.

# Multiple Variable Linear Regression with Interaction

Hide

```
lm3 <- lm(win_percent~damage_round+kills_round+headshot_percent+kd_ratio+score_round+damage_roun
d*kills_round*headshot_percent*kd_ratio*score_round, data=train)
summary(lm3)
```

```
Call:
lm(formula = win_percent ~ damage_round + kills_round + headshot_percent +
    kd_ratio + score_round + damage_round * kills_round * headshot_percent *
    kd_ratio * score_round, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-44.211  -5.116  -0.453   4.639  41.157

Coefficients:
                                                                 Estimate Std. Error t value Pr
(>|t|)
(Intercept)                                                  -3.949e+01  3.689e+01  -1.071  0.
28440
damage_round                                                  1.557e+00  1.172e+00   1.329  0.
18397
kills_round                                                   4.434e+01  1.510e+02   0.294  0.
76905
headshot_percent                                              3.605e+00  1.658e+00   2.174  0.
02968 *
kd_ratio                                                      5.200e+02  7.873e+01   6.605 4.0
1e-11 ***
score_round                                                  -1.634e+00  8.960e-01  -1.824  0.
06814 .
damage_round:kills_round                                     -2.303e+00  1.950e+00  -1.181  0.
23775
damage_round:headshot_percent                                -4.773e-02  4.993e-02  -0.956  0.
33915
kills_round:headshot_percent                                  4.155e-01  6.166e+00   0.067  0.
94628
damage_round:kd_ratio                                        -3.574e+00  1.252e+00  -2.855  0.
00430 **
kills_round:kd_ratio                                         -3.689e+02  1.326e+02  -2.783  0.
00540 **
headshot_percent:kd_ratio                                    -1.447e+01  3.215e+00  -4.501 6.7
7e-06 ***
damage_round:score_round                                      8.843e-03  3.452e-03   2.561  0.
01043 *
kills_round:score_round                                       1.917e+00  1.076e+00   1.783  0.
07466 .
headshot_percent:score_round                                  2.422e-02  3.646e-02   0.664  0.
50654
kd_ratio:score_round                                         -1.103e+00  9.416e-01  -1.171  0.
24143
damage_round:kills_round:headshot_percent                     1.078e-02  7.949e-02   0.136  0.
89217
damage_round:kills_round:kd_ratio                             3.506e+00  1.573e+00   2.229  0.
02585 *
damage_round:headshot_percent:kd_ratio                        1.225e-01  5.123e-02   2.392  0.
01677 *
kills_round:headshot_percent:kd_ratio                         1.009e+01  5.284e+00   1.910  0.
```

```
05615 .
damage_round:kills_round:score_round                             -7.728e-03  3.162e-03  -2.444  0.
01454 *
damage_round:headshot_percent:score_round                        -1.493e-04  1.377e-04  -1.085  0.
27805
kills_round:headshot_percent:score_round                         -2.994e-02  4.338e-02  -0.690  0.
49009
damage_round:kd_ratio:score_round                                 6.331e-03  3.322e-03   1.905  0.
05672 .
kills_round:kd_ratio:score_round                                  5.457e-01  7.657e-01   0.713  0.
47605
headshot_percent:kd_ratio:score_round                             3.110e-02  3.840e-02   0.810  0.
41803
damage_round:kills_round:headshot_percent:kd_ratio               -8.081e-02  6.214e-02  -1.301  0.
19343
damage_round:kills_round:headshot_percent:score_round             2.360e-04  1.209e-04   1.952  0.
05092 .
damage_round:kills_round:kd_ratio:score_round                    -5.199e-03  1.248e-03  -4.167 3.0
9e-05 ***
damage_round:headshot_percent:kd_ratio:score_round               -2.456e-04  1.315e-04  -1.868  0.
06171 .
kills_round:headshot_percent:kd_ratio:score_round                -2.659e-02  3.094e-02  -0.859  0.
39010
damage_round:kills_round:headshot_percent:kd_ratio:score_round   1.558e-04  4.942e-05   3.153  0.
00162 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.367 on 67077 degrees of freedom
Multiple R-squared:  0.1541,    Adjusted R-squared:  0.1537
F-statistic: 394.1 on 31 and 67077 DF,  p-value: < 2.2e-16
```

Run anova()

Hide

```
anova(lm2, lm3)
```

```
Analysis of Variance Table

Model 1: win_percent ~ damage_round + kills_round + headshot_percent +
    kd_ratio + score_round
Model 2: win_percent ~ damage_round + kills_round + headshot_percent +
    kd_ratio + score_round + damage_round * kills_round * headshot_percent *
    kd_ratio * score_round
  Res.Df     RSS Df Sum of Sq      F    Pr(>F)
1  67103 4730191
2  67077 4696029 26     34161 18.767 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

This shows that using interaction didn’t change the result.

Hide

```
pred3 <- predict(lm3, newdata=test)
cor3 <- cor(pred3, test$win_percent)
mse3 <- mean((pred3-test$win_percent)^2)
rmse3 <- sqrt(mse3)
print(paste('correlation:', cor3))
```

```
[1] "correlation: 0.383791517018242"
```

Hide

```
print(paste('mse:', mse3))
```

```
[1] "mse: 71.9354401840263"
```

Hide

```
print(paste('rmse:', rmse3))
```

```
[1] "rmse: 8.4814762974394"
```

# Conclusion

All of these models didn't work for this data set. They were unable to accurately distinguish between the data points since they were so tightly packed.

This likely didn't work due to the wide range of differently performing players. Any time a player does better, someone else will do worse, but they still remain in the data set.