# NFL 2002-2021 Score Differential

Kenizzer

## Add in data and print summary

This data set was generated by Reddit user **gigantoir** https://www.reddit.com/r/NFLstatheads/comments/q73yd0/nfl_scores_20172020/ I added the 2021 data that was scrapped from https://www.footballdb.com/games/index.html and 2002-2016 data from Reddit user **yuxbni76** https://www.reddit.com/user/yuxbni76

```
Scores <- read.csv("nfl_dataset_2002-2019week6.csv", header=TRUE, sep= ",")
Scores$Home_win <- factor(Scores$score_home > Scores$score_away, labels=c("Home_loss", "Home_win"))
summary(Scores)
```

```
##      date               away               home           first_downs_away
##  Length:4631        Length:4631        Length:4631        Min.   : 3.00
##  Class :character   Class :character   Class :character   1st Qu.:15.00
##  Mode  :character   Mode  :character   Mode  :character   Median :19.00
##                                                           Mean   :18.78
##                                                           3rd Qu.:22.00
##                                                           Max.   :37.00
##  first_downs_home third_downs_away   third_downs_home   fourth_downs_away
##  Min.   : 3.00    Length:4631        Length:4631        Length:4631
##  1st Qu.:16.00    Class :character   Class :character   Class :character
##  Median :20.00    Mode  :character   Mode  :character   Mode  :character
##  Mean   :19.78
##  3rd Qu.:23.00
##  Max.   :40.00
##  fourth_downs_home  passing_yards_away passing_yards_home rushing_yards_away
##  Length:4631        Min.   : -7.0      Min.   :  6.0      Min.   :-18.0
##  Class :character   1st Qu.:164.0      1st Qu.:172.0      1st Qu.: 73.0
##  Mode  :character   Median :217.0      Median :221.0      Median :103.0
##                     Mean   :219.9      Mean   :226.6      Mean   :109.7
##                     3rd Qu.:273.0      3rd Qu.:276.0      3rd Qu.:139.0
##                     Max.   :516.0      Max.   :522.0      Max.   :351.0
##  rushing_yards_home total_yards_away total_yards_home comp_att_away
##  Min.   : -3.0      Min.   : 26.0    Min.   : 77.0    Length:4631
##  1st Qu.: 81.0      1st Qu.:270.0    1st Qu.:286.0    Class :character
##  Median :112.0      Median :329.0    Median :343.0    Mode  :character
##  Mean   :117.8      Mean   :329.6    Mean   :344.4
##  3rd Qu.:148.0      3rd Qu.:389.0    3rd Qu.:400.0
##  Max.   :378.0      Max.   :643.0    Max.   :653.0
##  comp_att_home       sacks_away         sacks_home         rushing_attempts_away
##  Length:4631        Length:4631        Length:4631        Min.   : 6.00
##  Class :character   Class :character   Class :character   1st Qu.:21.00
##  Mode  :character   Mode  :character   Mode  :character   Median :26.00
##                                                           Mean   :26.59
##                                                            3rd Qu.:32.00
##                                                            Max.   :57.00
##  rushing_attempts_home  fumbles_away     fumbles_home      int_away
##  Min.   : 6.00         Min.   :0.0000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:22.00         1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000
##  Median :28.00         Median :0.0000   Median :0.000   Median :1.0000
##  Mean   :27.83         Mean   :0.6597   Mean   :0.653   Mean   :0.9836
##  3rd Qu.:33.00         3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:2.0000
##  Max.   :60.00         Max.   :5.0000   Max.   :4.000   Max.   :6.0000
##     int_home     turnovers_away  turnovers_home  penalties_away
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Length:4631
##  1st Qu.:0.000   1st Qu.:1.000   1st Qu.:1.000   Class :character
##  Median :1.000   Median :1.000   Median :1.000   Mode  :character
##  Mean   :0.916   Mean   :1.643   Mean   :1.569
##  3rd Qu.:1.000   3rd Qu.:2.000   3rd Qu.:2.000
##  Max.   :6.000   Max.   :8.000   Max.   :7.000
##  penalties_home     redzone_away       redzone_home        drives_away
##  Length:4631        Length:4631        Length:4631        Min.   : 0.00
##  Class :character   Class :character   Class :character   1st Qu.:11.00
##  Mode  :character   Mode  :character   Mode  :character   Median :12.00
##                                                           Mean   :12.48
##                                                           3rd Qu.:14.00
##                                                           Max.   :26.00
##   drives_home   def_st_td_away   def_st_td_home   possession_away
##  Min.   : 0.0   Min.   :0.0000   Min.   :0.0000   Length:4631
##  1st Qu.:11.0   1st Qu.:0.0000   1st Qu.:0.0000   Class :character
##  Median :12.0   Median :0.0000   Median :0.0000   Mode  :character
##  Mean   :12.4   Mean   :0.3468   Mean   :0.3701
##  3rd Qu.:14.0   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :25.0   Max.   :6.0000   Max.   :6.0000
##  possession_home      score_away      score_home        Home_win
##  Length:4631        Min.   : 0.00   Min.   : 0.0   Home_loss:1987
##  Class :character   1st Qu.:14.00   1st Qu.:16.0   Home_win :2644
##  Mode  :character   Median :20.00   Median :23.0
##                     Mean   :20.82   Mean   :23.3
##                     3rd Qu.:27.00   3rd Qu.:30.0
##                     Max.   :59.00   Max.   :62.0
```

## Team colors

Team colors were extracted from https://teamcolorcodes.com, I took the first primary color for each team and created a list that will be for later use. For the Browns and Titans I took the secondary color as it seemed more *appropriate*.

```
Team_colors <- c("SF"="#AA0000",
                 "CHI"="#0B162A",
                 "CIN"="#FB4F14",
                 "BUF"="#00338D",
                 "DEN"="#FB4F14",
                 "CLE"="#FF3C00",
                 "TB"="#D50A0A",
                 "ARI"="#97233F",
                 "LAC"="#0080C6",
                 "KC"="#E31837",
                 "IND"="#002C5F",
                 "DAL"="#041E42",
                 "MIA"="#008E97",
                 "PHI"="#004C54",
                 "ATL"="#A71930",
                 "NYG"="#0B2265",
                 "JAX"="#006778",
                 "NYJ"="#125740",
                 "DET"="#0076B6",
                 "GB"="#203731",
                 "CAR"="#0085CA",
                 "NE"="#002244",
                 "LV"="#000000",
                 "LA"="#003594",
                 "BAL"="#241773",
                 "WAS"="#773141",
                 "NO"="#D3BC8D",
                 "SEA"="#002244",
                 "PIT"="#FFB612",
                 "HOU"="#03202F",
                 "TEN"="#4B92DB",
                 "MIN"="#4F2683")
```

## Machine learning

```
# Function to plot confusion matrix using ggtile plot from a confussion matrix object
# By user: Enrique Perez Herrero
# on https://stackoverflow.com/questions/46063234/how-to-produce-a-confusion-matrix-and-find-the-miscla
ggplotConfusionMatrix <- function(m){
  mytitle <- paste("Accuracy", percent_format()(m$overall[1]),
                   "Kappa", percent_format()(m$overall[2]))

  d <- as.data.frame.matrix(m$table)
  drn <- colnames(d)
  drr <- rownames(d)
  drs <- rowSums(d)
  d <- d %>% mutate_if(is.numeric, funs(./drs))
  d <- d %>% gather(x, value)
  Y <- cbind(as.data.frame(m$table), Proportion = d$value)
  Y$Reference <- fct_rev(Y$Reference) # Added this line to get a downward diagonal
  p <-
    ggplot(data = Y, aes(x = Reference, y = Prediction, fill= Proportion)) +
    geom_tile( colour = "white") +
    scale_fill_gradient(low = "white", high = "#14A02E", na.value = "white", limits=c(0,1)) +
    ggtitle(mytitle) +
    theme(legend.position = "right", axis.text.x = element_text(angle = 60, hjust = 1)) +
    guides(fill = guide_colorbar(frame.colour = "black", ticks = FALSE))
  return(p)
}
```

```r
MachineLearning_RF_ranger <- function(DF, GROUPING, TREES) {
  # 80:20 data split
  train_index <- as.data.frame(DF %>% sample_n(round(length(Scores$date) * 0.8)))
  train_index <- match(rownames(train_index), rownames(DF))
  train_x <- as.data.frame(DF[train_index, ])
  test_y <- as.data.frame(DF[-train_index, ])


  # Train set, 3705
  train_x$Date <- rownames(train_x)
  Training_meta.df <- train_x # this might fail here
  train_x <- subset(Training_meta.df, select = -c(Home_win, score_home, score_away))
  rownames(train_x) <- train_x$Sample
  train_x <- subset(train_x, select = -c(Date))
  Training_meta.df <- subset(Training_meta.df, select = c(Home_win, score_home, score_away))
  rownames(Training_meta.df) <- Training_meta.df$Date


  # Test set, 926 samples
  test_y$Date <- rownames(test_y)
  Testing_meta.df <- test_y
  test_y <- subset(Testing_meta.df, select = -c(Home_win, score_home, score_away))
  rownames(test_y) <- test_y$Sample
  test_y <- subset(test_y, select = -c(Date))
  Testing_meta.df <- subset(Testing_meta.df, select = c(Home_win, score_home, score_away))
  rownames(Testing_meta.df) <- Testing_meta.df$Date


  # Training model
  Training_grid <- expand.grid(.mtry = seq(10, length(train_x), round(length(train_x)*0.1)), .splitrule=
                               .min.node.size = c(1, 5, 10))
  train_control <- trainControl(method="cv", number=10)
  RF_CM <- list()
  RF_CM[["RF_model"]] <- train(x = train_x, y = Training_meta.df[[GROUPING]], method = "ranger", importa
                               tuneGrid = Training_grid, trControl = train_control, num.trees = TREES)
  RF_prediction_3 <- predict(RF_CM[["RF_model"]], test_y)
  RF_CM[["CMatrix"]] <- confusionMatrix(RF_prediction_3, as.factor(Testing_meta.df[[GROUPING]]), mode =
  RF_CM[["CMatrixPLOT"]] <- ggplotConfusionMatrix(RF_CM[["CMatrix"]])
  RF_CM[["VarImporance"]] <- varImp(RF_CM[["RF_model"]])
  return(RF_CM)
}



Home_win_pred <- MachineLearning_RF_ranger(Scores, "Home_win", 500)
```

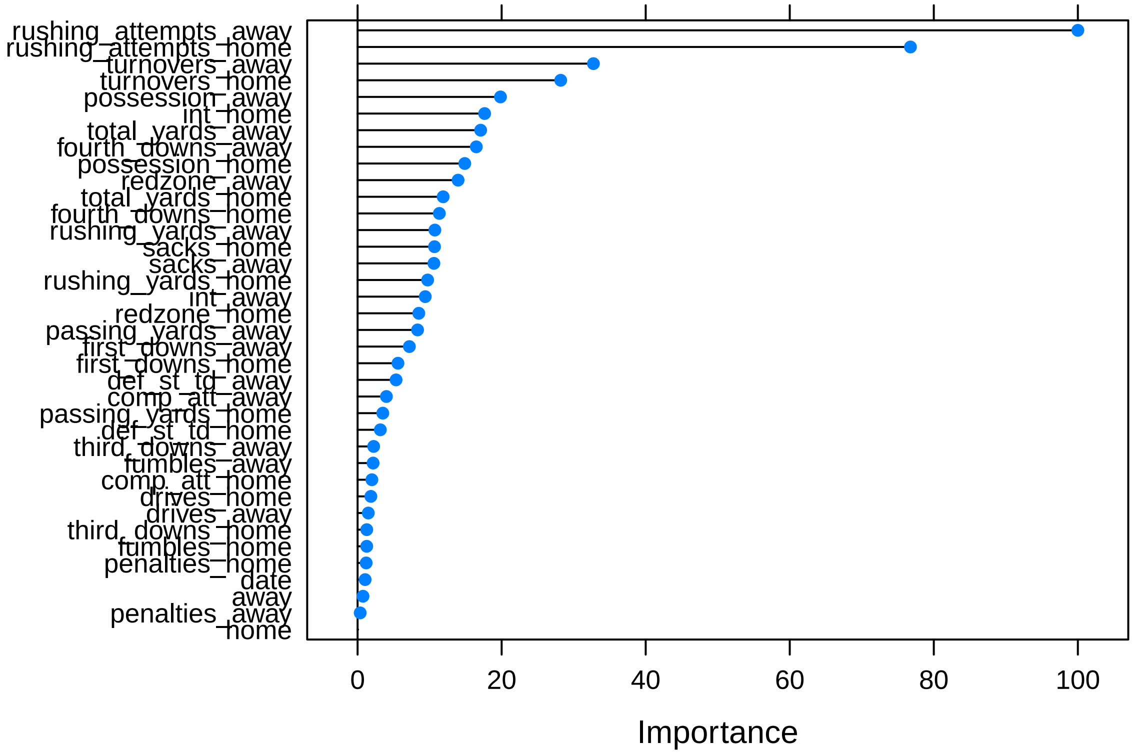
```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
```

```
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

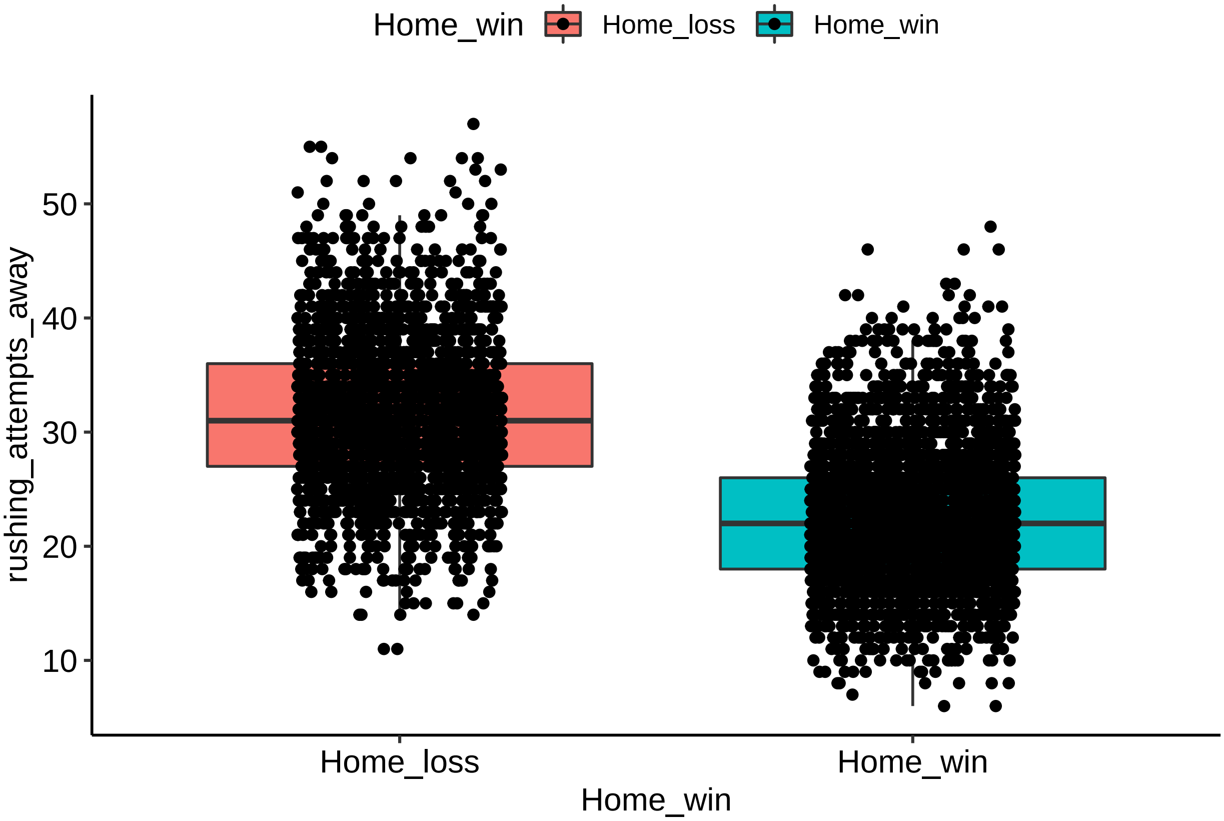
```
# It seems the model is really keying in on the number of running attempts
# My best guess is when teams are ahead they run the ball to kill the clock
# maybe this causes inflated run attempt totals
varImp(Home_win_pred$RF_model)
```

```
## ranger variable importance
##
##   only 20 most important variables shown (out of 37)
##
##                       Overall
## rushing_attempts_away 100.000
## rushing_attempts_home  76.760
## turnovers_away         32.741
## turnovers_home         28.200
## possession_away        19.840
## int_home               17.635
## total_yards_away       17.086
## fourth_downs_away      16.480
## possession_home        14.877
## redzone_away           13.954
## total_yards_home       11.880
## fourth_downs_home      11.351
## rushing_yards_away     10.727
## sacks_home             10.681
## sacks_away             10.595
## rushing_yards_home      9.725
## int_away                9.392
## redzone_home            8.500
## passing_yards_away      8.328
## first_downs_away        7.189
```

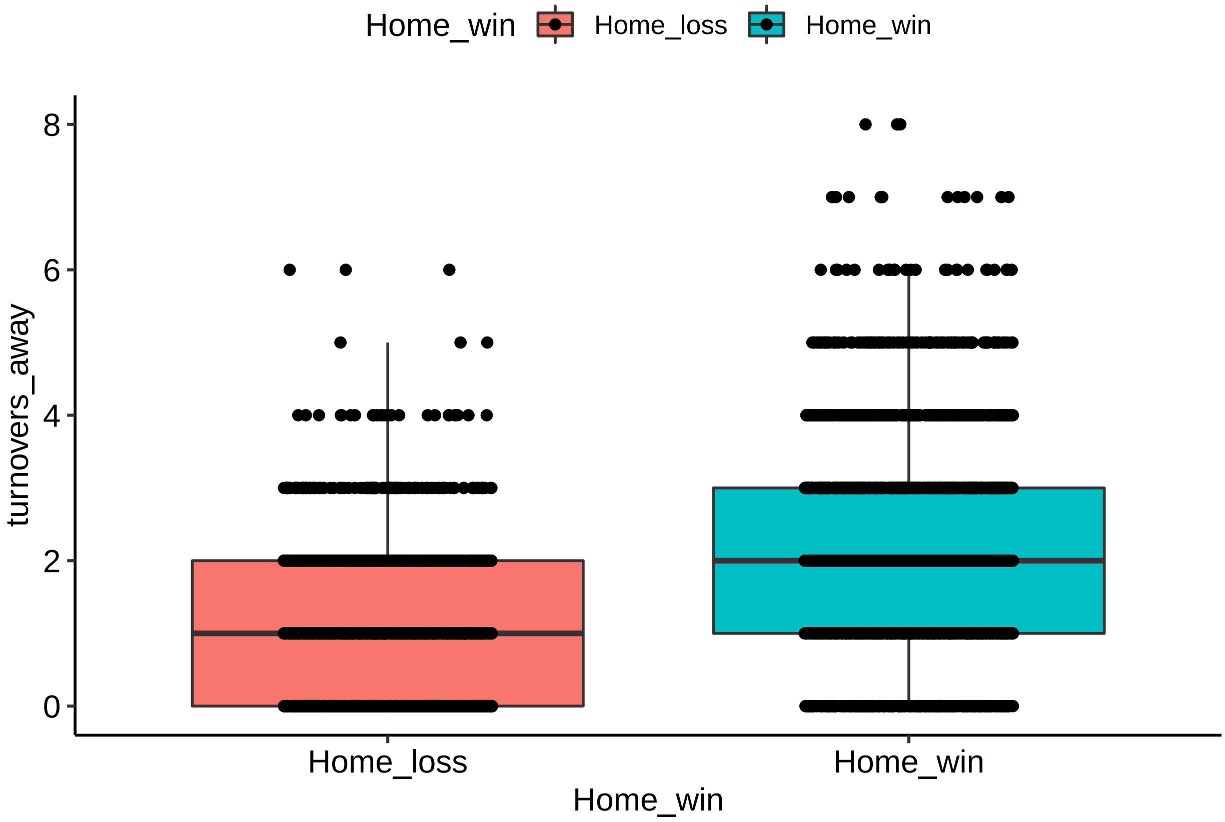
```
plot(Home_win_pred[["VarImporance"]])
```



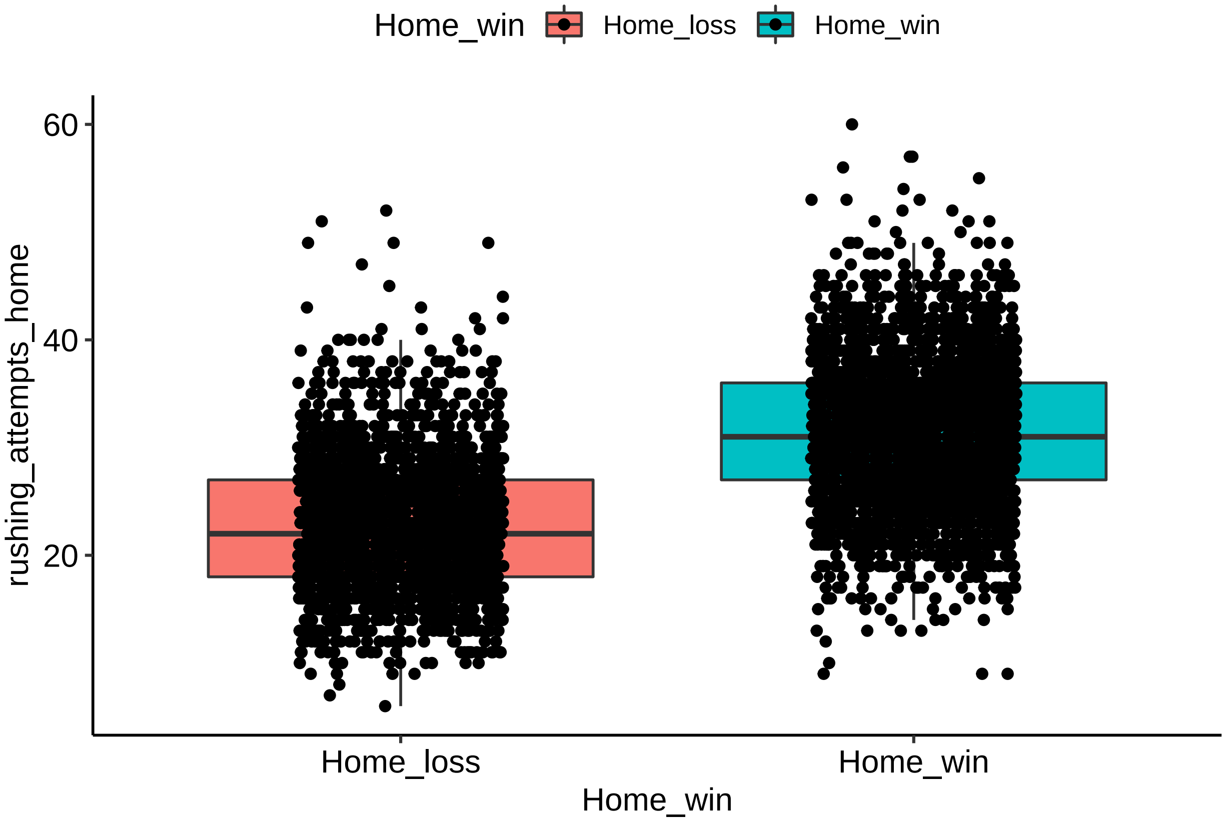
```
ggplot(Scores, aes(x = Home_win, y = rushing_attempts_away, fill = Home_win)) + geom_boxplot(outlier.sha
```



```
ggplot(Scores, aes(x = Home_win, y = turnovers_away, fill = Home_win)) + geom_boxplot(outlier.shape = N
```



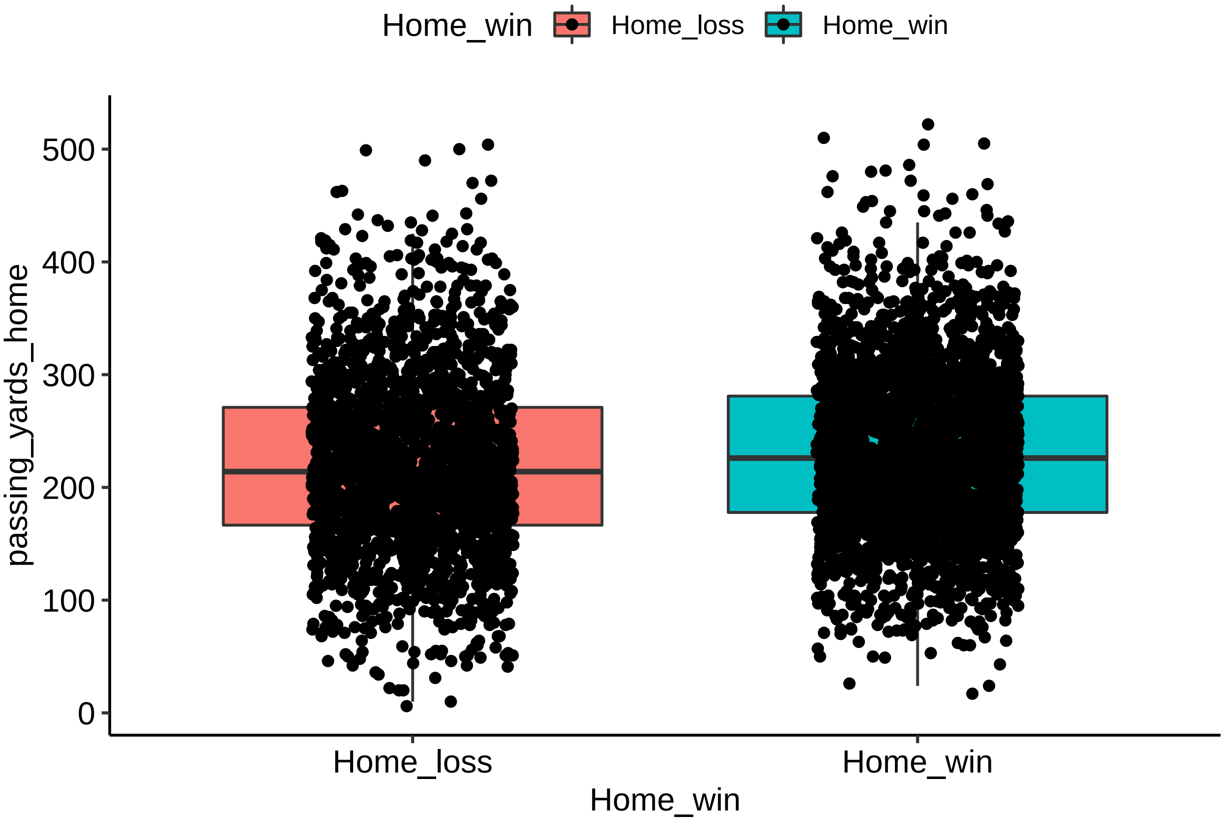
```
ggplot(Scores, aes(x = Home_win, y = rushing_attempts_home, fill = Home_win)) + geom_boxplot(outlier.sha
```



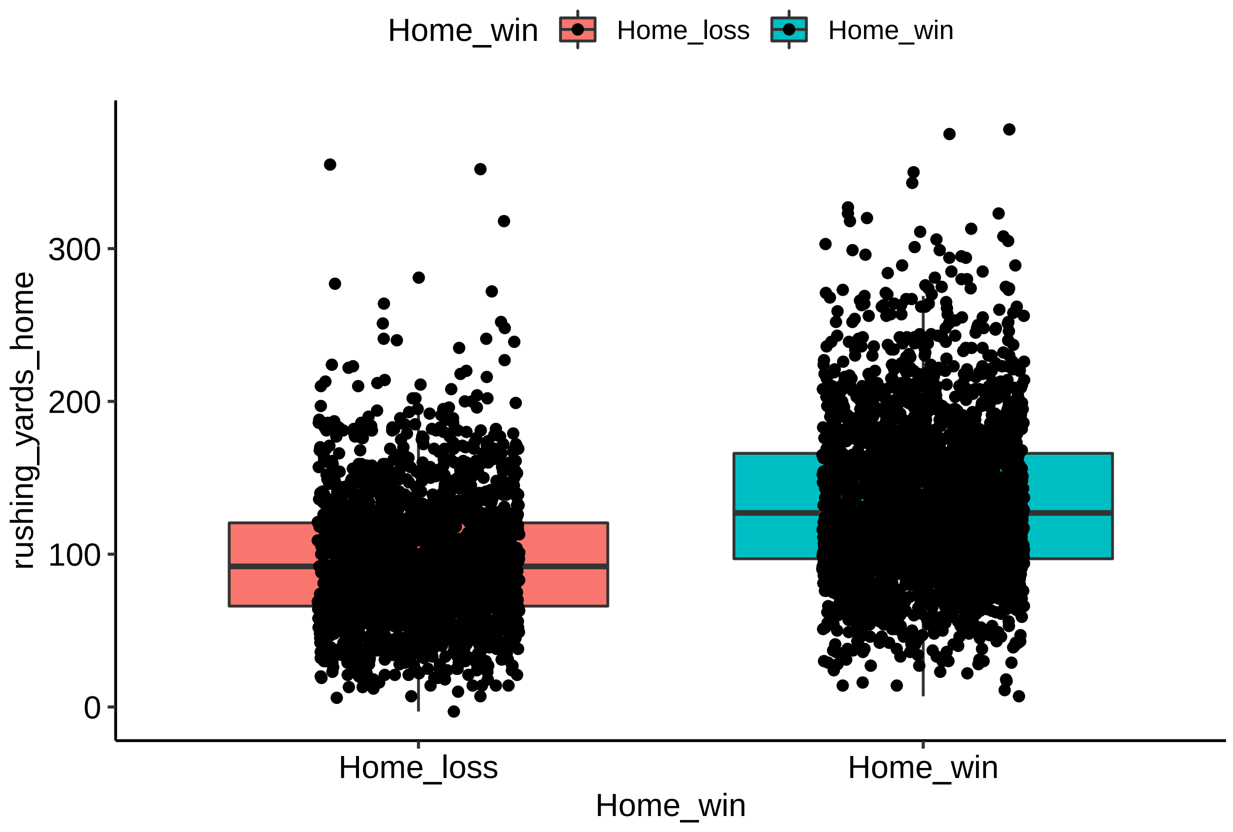
```
anova(lm(passing_yards_home ~ Home_win, data = Scores))
```

```
## Analysis of Variance Table
##
## Response: passing_yards_home
##             Df   Sum Sq Mean Sq F value    Pr(>F)
## Home_win     1   122335  122335  20.189 7.186e-06 ***
## Residuals 4629 28049298    6059
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ggplot(Scores, aes(x = Home_win, y = passing_yards_home, fill = Home_win)) + geom_boxplot(outlier.shape
```



```
ggplot(Scores, aes(x = Home_win, y = rushing_yards_home, fill = Home_win)) + geom_boxplot(outlier.shape
```



```
# How often do you win if you rush for 100 yards
summary(Scores[Scores$rushing_yards_home > 100,]$Home_win) / sum(summary(Scores[Scores$rushing_yards_ho
```

```
## Home_loss  Home_win
##  30.39971  69.60029
```

```
# How often do you win if you rush for 150 yards
summary(Scores[Scores$rushing_yards_home > 150,]$Home_win) / sum(summary(Scores[Scores$rushing_yards_ho
```

```
## Home_loss  Home_win
##  19.12965  80.87035
```

```
# How often do you win if you rush for 200 yards
summary(Scores[Scores$rushing_yards_home > 200,]$Home_win) / sum(summary(Scores[Scores$rushing_yards_ho
```

```
## Home_loss  Home_win
##  10.41009  89.58991
```

```
# How often do you win if you rush for 250 yards
summary(Scores[Scores$rushing_yards_home > 250,]$Home_win) / sum(summary(Scores[Scores$rushing_yards_ho
```

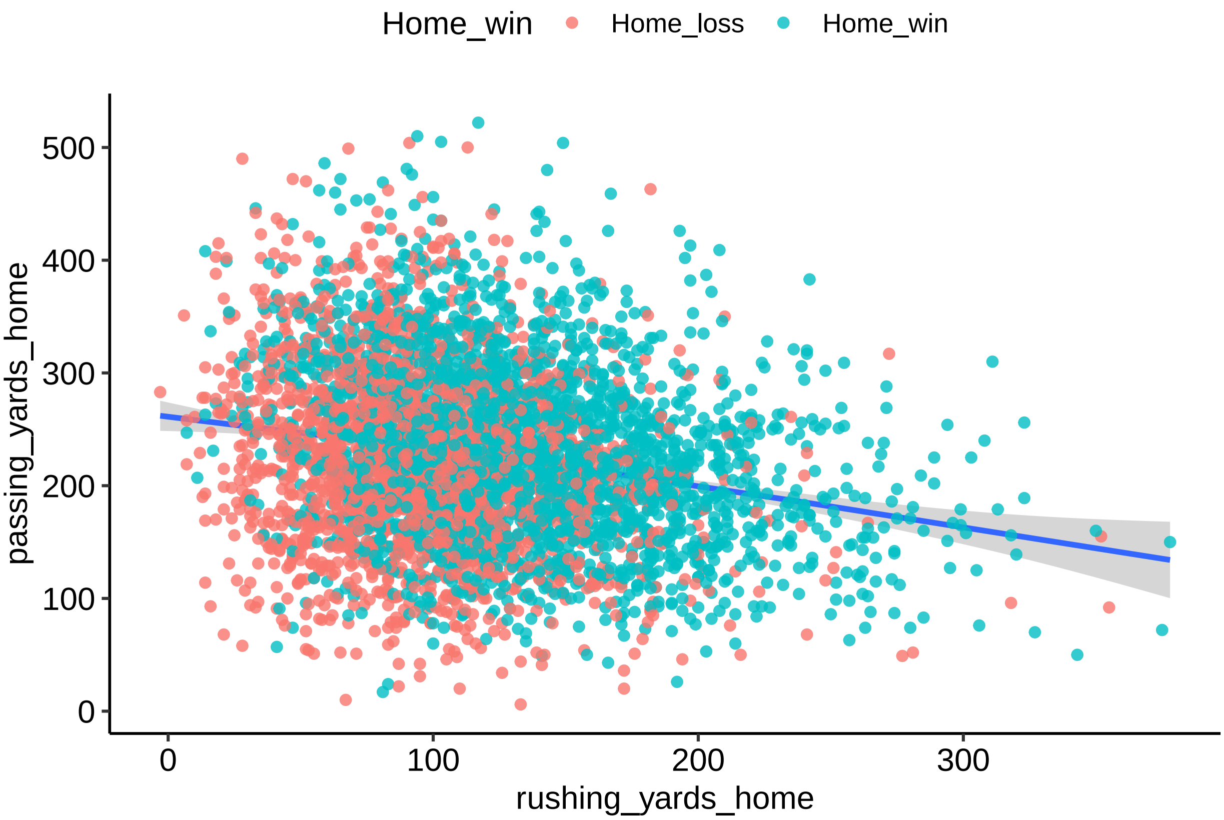
```
## Home_loss  Home_win
##  10.46512  89.53488
```

```
# How often do you win if you rush for 300 yards
summary(Scores[Scores$rushing_yards_home > 300,]$Home_win) / sum(summary(Scores[Scores$rushing_yards_hom
```

```
## Home_loss  Home_win
##  15.78947  84.21053
```

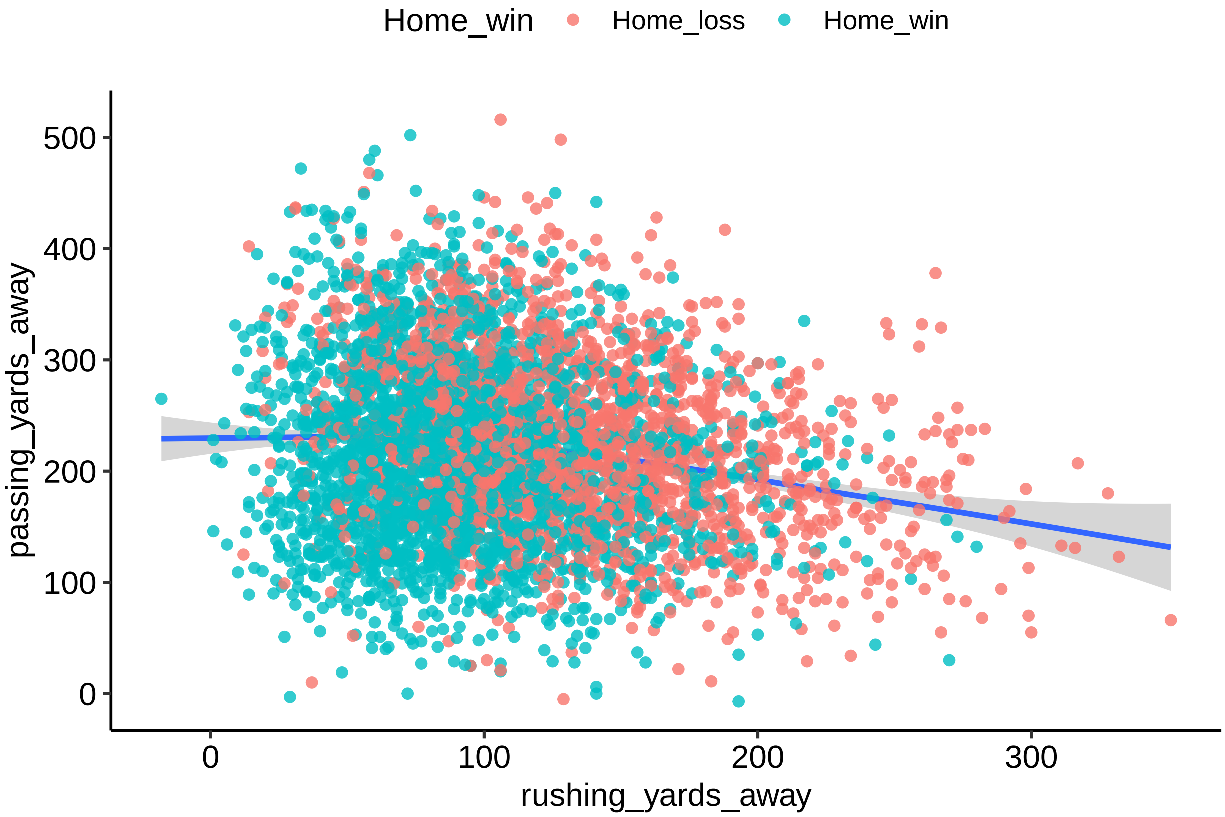
```
ggplot(Scores, aes(x = rushing_yards_home , y = passing_yards_home)) + geom_smooth() + geom_point(aes(co
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



```
ggplot(Scores, aes(x = rushing_yards_away , y = passing_yards_away)) + geom_smooth() + geom_point(aes(co
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```


Home_win
● Home_loss
● Home_win
500
400
300
200
100
0
passing_yards_away
0
100
200
300
rushing_yards_away