R Notebook

Alex Baker, James Kitch, Jackson Smith, Matthew Sheridan

Introduction

Sports games are designed around the idea of finding the team that plays the "best" over a given period of time. Professional baseball teams have nine innings, soccer teams ninety minutes, and American football, hockey, and basketball clubs have sixty minutes to score more points than their opponents. While officiating, weather, and "luck" inevitably play a role in deciding the outcome, the end result of a game should hopefully be evident by in-game statistics. Bill James' arguably invented the field of sports analytics in **year**, attempting to analyze past player performance and predict future team success based on quantitative measurements rather than qualitative scouting reports. This quantitative view of player and team statistics, now known as "sabermetrics", exploded in especially in baseball where it is very easy to generate a large number of individual statistics for every player.

The National Football League (NFL) was slower to adopt sabermetrics, but the rise of remote sensing software in recent years has made it much easier for players and individuals to acquire quantitative game statistics. Football presents an interesting area of analysis since it is naturally discretized into distinct plays which can be measured quantitatively. How many net yards were gained? Was it a run or a rush play? A missed field goal attempt, and if so from how many yards? R represents an appealing

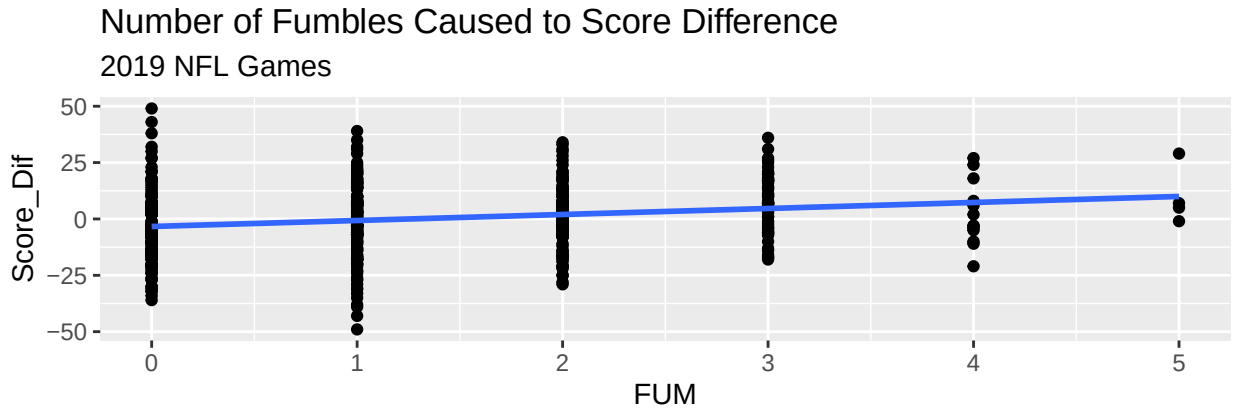
Our data comes from the nflfastR package, which contains accumulated play-by-play data from 1999 through 2021, with additional predictors beginning in 2006. We hope to create a parsimonious, streamlined model capable of predicting the score differential of an NFL game within some reasonable confidence interval. While prediction is a outcome and measure of success for any model, we also hope to use inference from our models. While we will have some variables with relatively clear-cut relationships with score differential, our analysis will also focus on how more "strategic" variables relate to success. More turnovers in a game and more total yards should correlate strongly with score differential, but we are interested in variables that have a more unclear relationship with the final outcome. Are missed extra point attempts indicative of the team as a whole having a bad day? What about third down conversion rate? And quarterback hits? These are the kinds of interactions and trends we hope to expose in our model. By using metrics that are not generally used to predict game outcomes, we hope to find insight into predicting wins that are not conventionally expected.

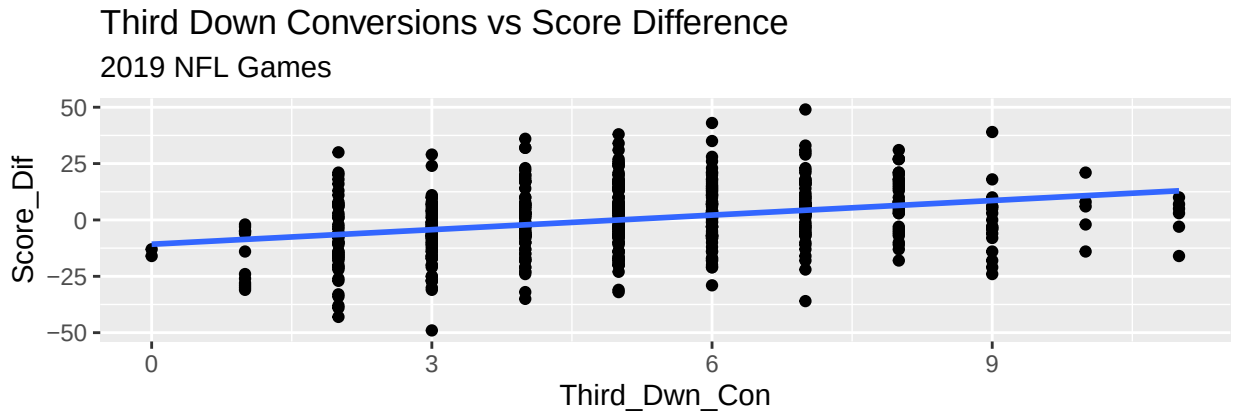
Data Cleaning, EDA

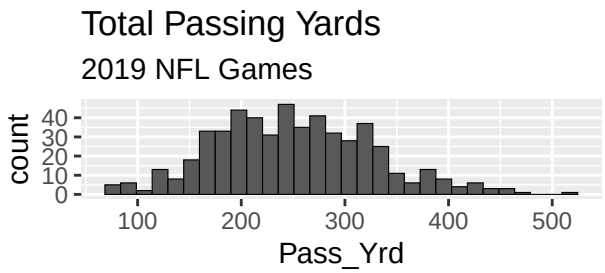
Below, we graphed the distributions for a number of different potential predictors, as well as their relationship with the response variable of interest. Most predictors are slightly right-skewed, as they have a positive support and have a slight "bell" shape for the most common values but are stretched out by the few exceptional games where a team throws for 500 yards or rushes for 250 yards. However, it is mild right-skewness as it is not to the extent that log-transforming them would make the distributions more symmetric. For example, if we log-transform `Pass_Yrd` we actually find that the the resulting distribution is left-skewed, suggesting that the log transformation was actually too strong!

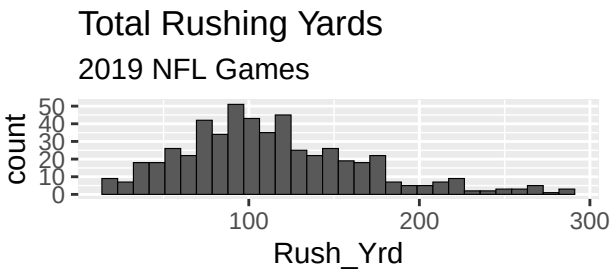
Furthermore, we can do a preliminary investigation of the relationships between some of the predictors and the outcome variable of score differential in 2019. Notably, we can see that the number of fumbles

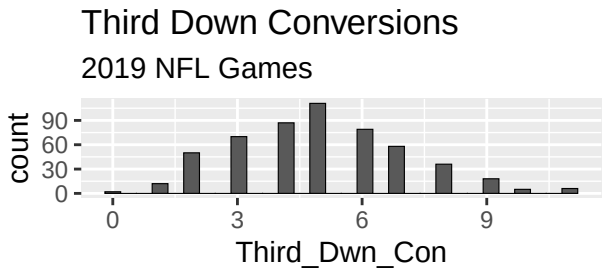
caused by a team is positively associated with score differential and that it is a relatively linear trend. The number of third down conversions is also positively associated with score differential, but this seems more complicated than a "simple" linear trend. Teams with a very large number (> 8) of third down conversions in a game appear to have a more negative score differential than teans with 5-7 third down conversions. This suggests that in future models investigating the effects of quadratic or higher order polynomial terms could be reasonable.

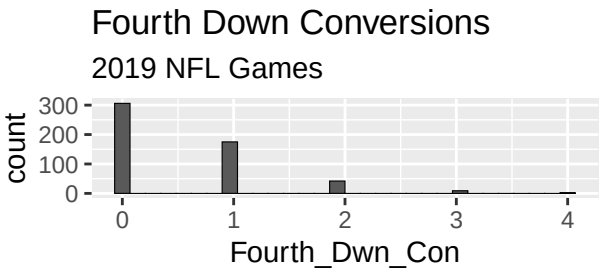


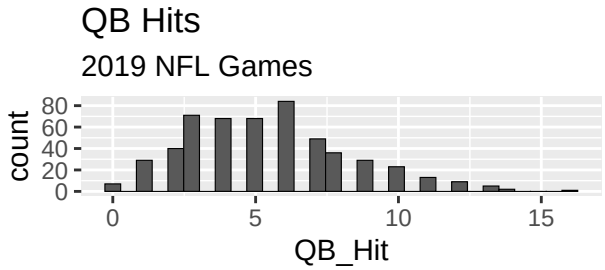













```
cor_matrix <- cor(data_2019[4:18])
cor_matrix[upper.tri(cor_matrix)] <- NA
cor_matrix <- cor_matrix %>% round(2)
cor_matrix
```

```
##                   TFL QB_Hit Pass_Yrd Rush_Yrd FG_attempt XP_missed FG_missed
## TFL              1.00     NA       NA       NA          NA         NA         NA
## QB_Hit           0.06   1.00       NA       NA          NA         NA         NA
## Pass_Yrd        -0.01   0.06     1.00       NA          NA         NA         NA
## Rush_Yrd        -0.13  -0.25    -0.21     1.00          NA         NA         NA
## FG_attempt       0.02  -0.03     0.08     0.15        1.00         NA         NA
## XP_missed       -0.06  -0.03     0.04     0.10       -0.04       1.00         NA
## FG_missed        0.05   0.05     0.10     0.04        0.41       0.02       1.00
## FUM              0.00  -0.01    -0.07    -0.01        0.18       0.09       0.06
## INT             -0.12  -0.12    -0.16     0.11        0.10       0.03       0.03
## Fourth_Dwn_Con  -0.05   0.11     0.08     0.01       -0.02       0.03       0.00
## Third_Dwn_Con   -0.10  -0.07     0.31     0.18        0.08       0.04      -0.02
## Pen              0.07   0.12     0.12    -0.02        0.00      -0.01       0.00
## RedZone_Plays   -0.20  -0.18     0.31     0.32        0.22       0.13      -0.03
## Unique_Receivers -0.03   0.02     0.15    -0.04        0.05       0.00      -0.02
## Shotgun_Plays    0.05   0.28     0.23    -0.08        0.05      -0.02      -0.02
##                   FUM   INT Fourth_Dwn_Con Third_Dwn_Con   Pen RedZone_Plays
## TFL                NA    NA             NA            NA    NA            NA
## QB_Hit             NA    NA             NA            NA    NA            NA
## Pass_Yrd           NA    NA             NA            NA    NA            NA
```

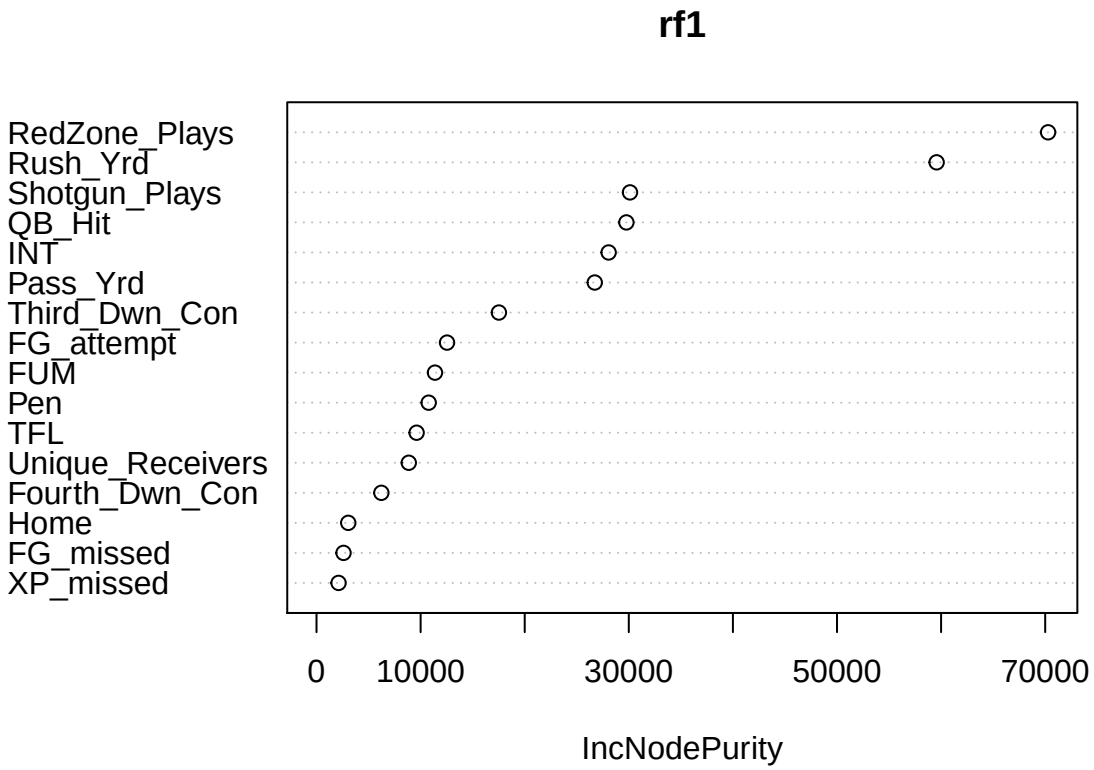
```
## Rush_Yrd             NA    NA             NA            NA    NA            NA
## FG_attempt           NA    NA             NA            NA    NA            NA
## XP_missed            NA    NA             NA            NA    NA            NA
## FG_missed            NA    NA             NA            NA    NA            NA
## FUM                1.00    NA             NA            NA    NA            NA
## INT                0.09  1.00             NA            NA    NA            NA
## Fourth_Dwn_Con    -0.10 -0.02           1.00            NA    NA            NA
## Third_Dwn_Con      0.06  0.06          -0.04          1.00    NA            NA
## Pen                0.04 -0.02          -0.04         -0.02  1.00            NA
## RedZone_Plays      0.04  0.18           0.13          0.34 -0.07          1.00
## Unique_Receivers -0.01 -0.02           0.08          0.10  0.05          0.04
## Shotgun_Plays    -0.09 -0.15           0.27          0.13  0.13          0.04
##                  Unique_Receivers Shotgun_Plays
## TFL                            NA            NA
## QB_Hit                         NA            NA
## Pass_Yrd                       NA            NA
## Rush_Yrd                       NA            NA
## FG_attempt                     NA            NA
## XP_missed                      NA            NA
## FG_missed                      NA            NA
## FUM                            NA            NA
## INT                            NA            NA
## Fourth_Dwn_Con                 NA            NA
## Third_Dwn_Con                  NA            NA
## Pen                            NA            NA
## RedZone_Plays                  NA            NA
## Unique_Receivers             1.00            NA
## Shotgun_Plays                0.12             1
```

Initial Modeling:

The code for initializing the models has been hidden for space constraints. The linear model was using all predictors, the stepwise model stepped from an intercept only model to all predictors, and the randomforest has not been tuned (YET) and uses all predictors, as well as the default parameters. We can see that the RMSE's are pretty close together for all the models, but the linear linear model seems to outperform all.

```
maxnodes <- c(100, 500, 1000)
mtry <- c(1,5,8,10,12)

combs <- expand.grid(mtry, maxnodes)
colnames(combs) <- c("mtry", "maxnodes")
combs$RMSE_train <- rep(NA, nrow(combs))
combs$RMSE_test <- rep(NA, nrow(combs))

set.seed(139)
for(i in 1:nrow(combs)) {
  temp_forest <- randomForest(Score_Dif~.-Game_id-Team_Name-year,data=data_2017_19, mtry = combs$mtry[i]
  combs$RMSE_train[i] <- RMSE(data_2017_19$Score_Dif, temp_forest$predicted)
  combs$RMSE_test[i] <- RMSE(data_2021$Score_Dif, predict(temp_forest, new = data_2021))
}
combs$RMSE_difference <- combs$RMSE_test - combs$RMSE_train
```

```
rf1 <- randomForest(Score_Dif~.-Game_id-Team_Name-year,
                    data=data_2017_19,
                    mtry = combs[which.min(combs$RMSE_test),]$mtry,
                    maxnodes = combs[which.min(combs$RMSE_test),]$maxnodes,
                    ntree=200)



varImpPlot(rf1)
```



A random forest was fit using the combined game data from years 2017-2019. One advantage to using a random forest in this case is that it represents a good way to measure the predictive power of the predictors chosen, due to the non-parametric nature of the random forest and lack of assumptions needed to fit the model. After fitting a random forest with desired predictors, we can plot the Variable Importance of the predictors, to see how much predictive power is lost when the predictor is removed from the tree, and the greater the disparity the more important said predictor is. The one disadvantage to this method is that while we are able to tell that a certain predictor is very important, we are unable to interpret whether the relationship between the predictor and the response is positive or negative. For example, we see that QB_hits (the number of hits a team's quarterback takes) is considered important. We may assume, as logic follows, that minimizing the amount of times your quarterback is hit is most beneficial for a large score differential. However, we cannot determine that this is the case from only this plot.

A random forest was fit over a range of different hyper-parameters. Different values of mtry, ranging from 1 (essentially a random forest) to 12, reflect the number of variables randomly sampled at each split within the random forest, where higher values add more complexity. Secondly is maxnodes, which represents the maximum number of terminal nodes present in the resulting forest. The number of trees within our forest

was fixed at 200 to maintain a reasonable run time. After cross-validating the RMSE between predicted and true values among training (2017-2019 data) and testing (2021 data) sets, the lowest test set RMSE was selected, and it was found that the optimal set of hyper parameters from our tests is mtry = 8 and maxnodes = 1000. Looking at the calculated differences in RMSEs on training and testing data, no models appear to be egregiously over-fit. Our selected hyper-parameters do not produce one of the largest magnitude RMSE differences, so it appears that our tuned model is acceptable.

Using the aforementioned hyper-parameters, the model was fit again and a variable importance plot was generated. This plot shows which variables have the most predictive power in the random forest. We can see that the highest predictors in terms of variable importance are Red Zone Plays and rushing yards, which appear to be in a tier of their own. This is followed by another tier that includes shotgun plays, QB hits, interceptions and passing yards. The rest of the predictors are of much lower importance based off of this plot. This suggests that these mentioned predictors have the most important relationships with score differential for any given game.

It seems obvious that the number of touchdowns a team scores would have large impact on their score differential. In the broader context of football, it makes sense that red zone plays would be considered highly important for predicting score differential because, considering that the red zone is the area within 20 yards of the opponent's end zone, this is where points are scored, and the more plays that are run in this area the more points should be scored. It also makes logical sense that rushing yards should be almost as important, both because teams that are leading by large amounts tend to run much more in order to posses the ball, and conversely teams that have great success running tend to have more control of the game and time of possession. In terms of the "second tier" predictors, QB_hits makes sense as a solid predictor. Presumably, the fewer hits your quarterback is taking, then the better chance you have of putting up more points than your opponent. The number of interceptions a team gets gives more opportunities to score while having to drive fewer yards down the field than if the ball was obtained otherwise (such as a punt, or a kickoff). The number of passing yards and shotgun plays are slightly less interpretable in terms of their importance. Passing yards could be a positive predictor because a team that amasses lots of passing yardage likely scores many points. However, a team that is behind by a large amount may gain many passing yards towards the end of the game in a meaningless effort to catch up to the winning team (this is referred to as "garbage time" statistics, when the outcome of the game is relatively determined, but play still occurs and players amass stats). Similarly, the number of stats taken out of the shotgun could be the mark of a team being successfully aggressive in their passing game, or that a team is struggling behind and trying to play catch-up. It would be best to investigate the nature of the relationship of these predictors with our chosen response in some sort of other linear model in order to examine the sign and size of the associated coefficient.

LMER Modeling

```
data_2017_19_scale <- data_2017_19
data_2017_19_scale[,6:7] <- data_2017_19_scale[,6:7] %>% sapply(scale, scale=F)


lmer_all <- lmer(Score_Dif ~. -Game_id -Team_Name -year + (1|Team_Name),data=data_2017_19)

lmer_step <- lmer(Score_Dif ~ Home + QB_Hit + Pass_Yrd + Rush_Yrd + FG_attempt +
    XP_missed + FG_missed + FUM + INT + Fourth_Dwn_Con + Third_Dwn_Con +
    RedZone_Plays + Unique_Receivers + Shotgun_Plays + (1|Team_Name),
    data=data_2017_19_scale)

lmer_step_slope <- lmer(Score_Dif ~ Home + QB_Hit + Pass_Yrd + Rush_Yrd + FG_attempt +
    XP_missed + FG_missed + FUM + INT + Fourth_Dwn_Con + Third_Dwn_Con +
    RedZone_Plays + Unique_Receivers + Shotgun_Plays + (1|Team_Name),
    data=data_2017_19_scale)
```

```
AIC(lmer_all, lmer_step, lmer_step_slope, lm1)
```

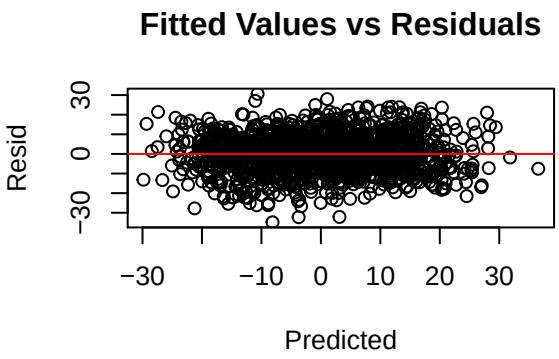
```
##                  df      AIC
## lmer_all         19 11875.70
## lmer_step        17 11869.88
## lmer_step_slope 17 11869.88
## lm1              18 11911.88
```

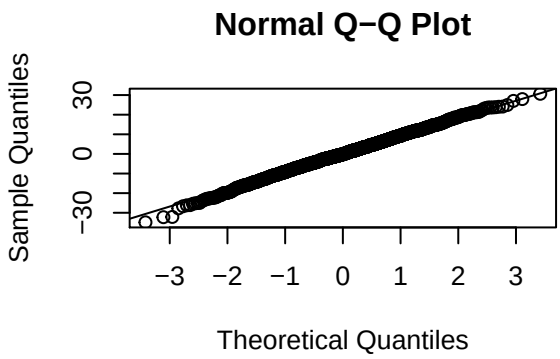
Next, we fit a Linear Mixed-Effects model with random intercepts by team to see if that had any impact on our predictions. Given that observations can be grouped by team and there are 32 teams, a mixed-effects model was a natural next step in our modeling approach. We fit two different types of mixed-effects models: one with all available predictors, and one with the predictors chosen by sequential variable selection. We found that the inclusion of a random intercept did not improve prediction substantially, with the random intercept term only accounting for $\approx 9\%$ of the total variance in the model. Most coefficient estimates were very similar to the standard OLS run previously. Some of this may be due to the fact that there are a relatively large number of observations per team. Although there are 32 teams, each team has 48 - 60 "measurements" or games in the dataset and thus the benefit of linear mixed-effects modeling in shrinking outlier intercepts towards the mean is less pronounced.

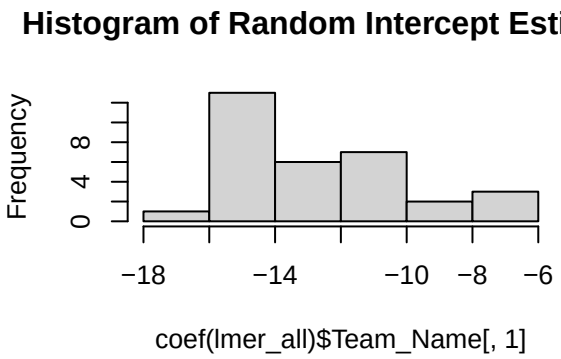
Nonetheless, a random intercept model with only the predictors returned by the stepwise variable selection process as outlined previously did increase the impact of the random intercept and decreased the AIC. We also fit a linear mixed-effects model with a random intercept by team and random slope for the relationship of total rushing yards with final score differential. In theory, if some teams were primarily a "passing" team, total rushing yards might have less of an outcome on final score differential than if their primary game strategy relied on the run. However, we find that this random slope model actually worsens the model AIC suggesting that this is not a particularly important trend to model.

The fitted values vs residuals plot and Normal Q-Q plots are both reasonable for this random-intercept mixed-effects model, demonstrating that constant variance and normality of residuals are acceptable assumptions for this model. However, there is some concern with the distribution of fitted random intercept estimates–the histogram is neither normal-shaped or symmetric. This could potentially be fixed with **Blank but it is probably not an issue at the end of the day**.

```
par(mfrow=c(2, 2))
plot(resid(lmer_all)~predict(lmer_all), main="Fitted Values vs Residuals", xlab="Predicted", ylab="Resi
abline(h=0, col = "red")
#normality of residuals
qqnorm(resid(lmer_all))
qqline(resid(lmer_all))
#check normality of random effects - non-normal
hist(coef(lmer_all)$Team_Name[,1], main="Histogram of Random Intercept Estimates")
```







Ridge, Lasso, and Stepwise Regression

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.0.5
```

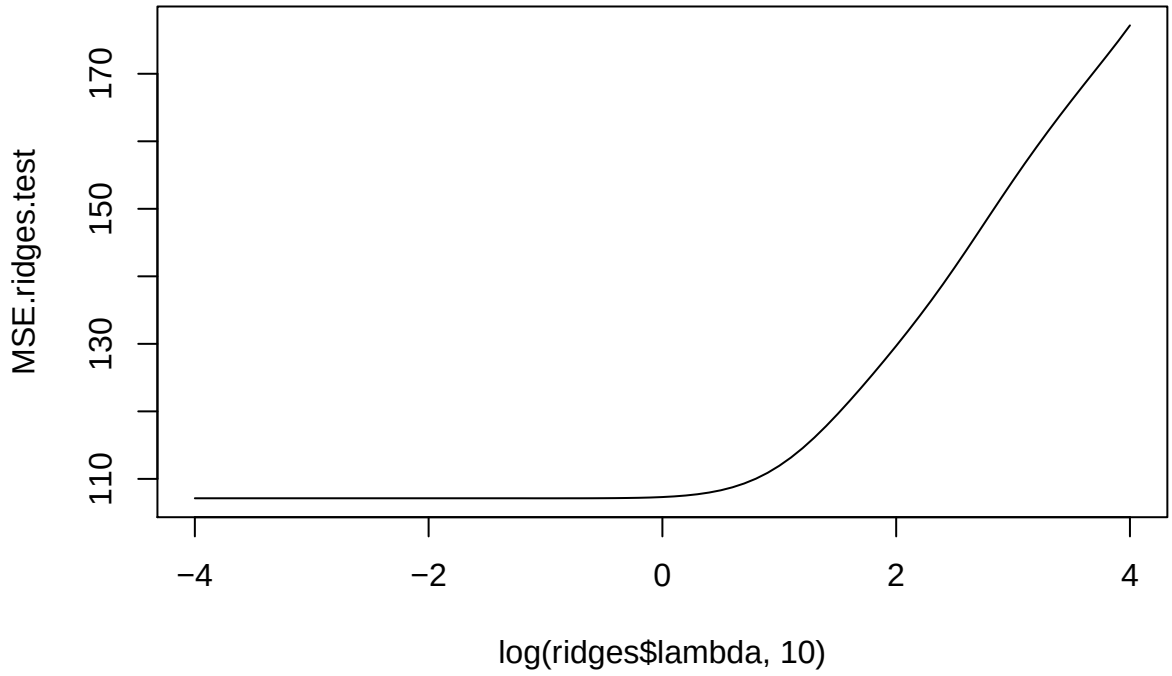
```
## Loaded glmnet 4.1-2
```

```
library(Rcpp)
```

```
## Warning: package 'Rcpp' was built under R version 4.0.5
```

```
library(Matrix)
#Baseline Linear Models
X = model.matrix(lm1)[,-1] #drop the intercept

#Ridge Model
ridges = glmnet(X, data_2017_19$Score_Dif, alpha = 0, lambda = 10^seq(-4,4,0.1), standardize = F)
#coef(ridges)
#ridges$lambda

n.test = nrow(data_2021)
```

```
lm.test = lm(formula(lm1), data = data_2021)
Xtest = model.matrix(lm.test)[,-1] #drop the intercept again
yhats.ridges = predict(ridges, Xtest)
residuals.ridge = (data_2021$Score_Dif - yhats.ridges)^2
MSE.ridges.test = apply(residuals.ridge, 2, sum)/n.test

plot(MSE.ridges.test~log(ridges$lambda,10),type="l")
```



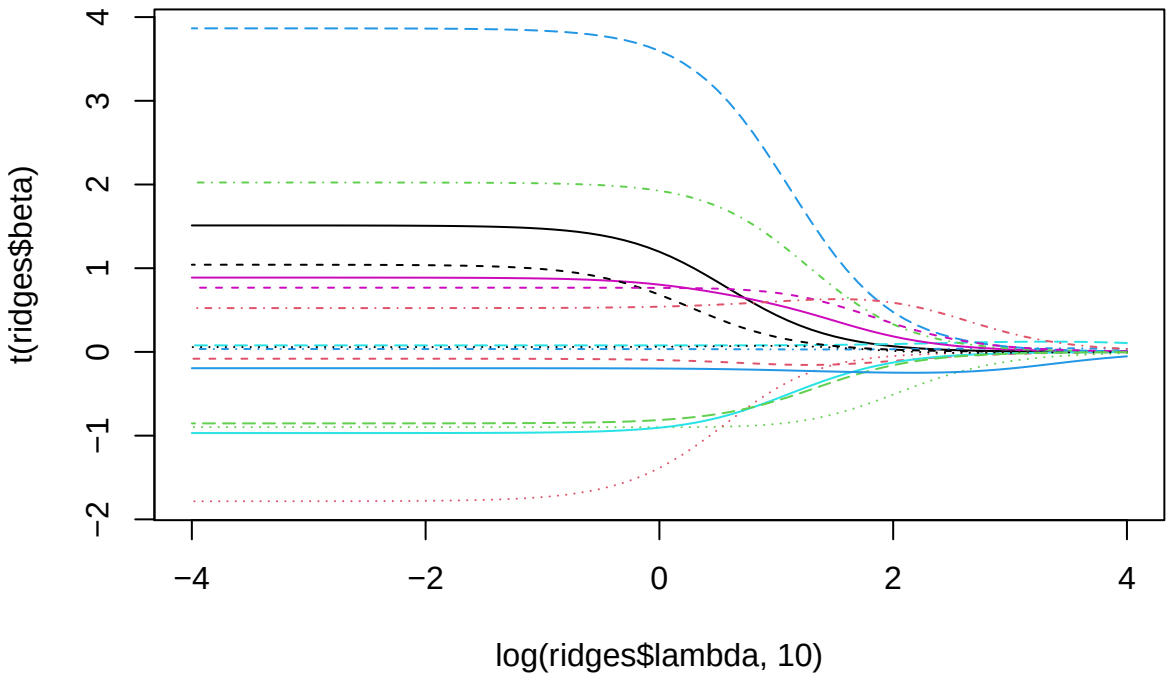

```
ridges$lambda[which.min(MSE.ridges.test)]
```

```
## [1] 0.1
```

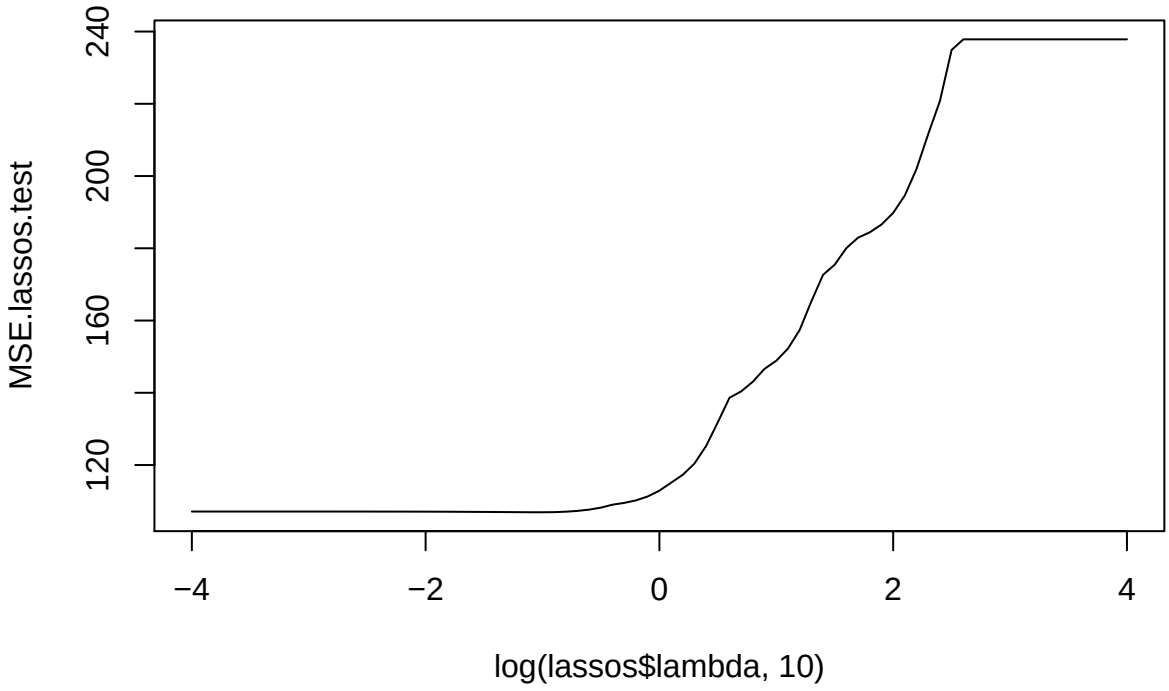
```
min(MSE.ridges.test)
```

```
## [1] 107.1261
```

```
matplot(log(ridges$lambda, 10), t(ridges$beta),type="l")
```




```
#Lasso Model
lassos = glmnet(X, data_2017_19$Score_Dif, alpha = 1, lambda = 10^seq(-4,4,0.1), standardize = F)
#coef(lassos)
#lassos$lambda

yhats.lassos = predict(lassos, Xtest)
residuals.lasso = (data_2021$Score_Dif - yhats.lassos)^2
MSE.lassos.test = apply(residuals.lasso, 2, sum)/n.test

plot(MSE.lassos.test~log(lassos$lambda,10),type="l")
```



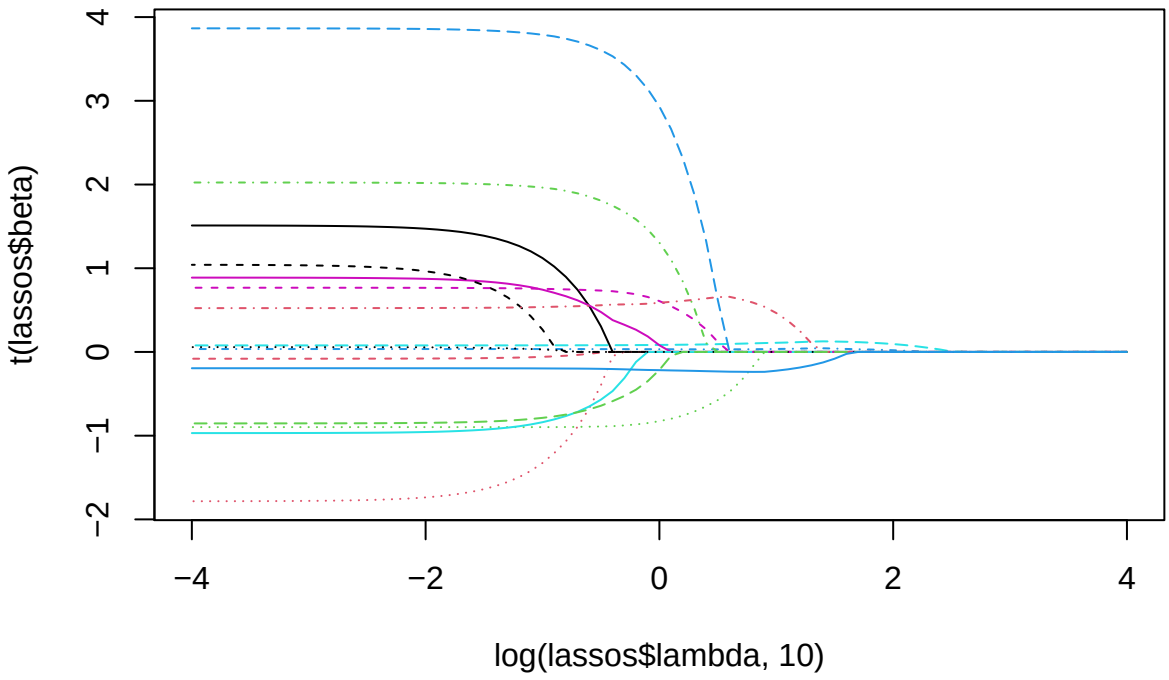

```
lassos$lambda[which.min(MSE.lassos.test)]
```

```
## [1] 0.1
```

```
min(MSE.lassos.test)
```

```
## [1] 106.9294
```

```
matplot(log(lassos$lambda, 10), t(lassos$beta),type="l")
```




```
#Stepwise Models
#Stepwise Backward
step.back = step(lm1, direction = "backward", k = 2, trace = F)
formula(step.back)
```

```
## Score_Dif ~ Home + QB_Hit + Pass_Yrd + Rush_Yrd + FG_attempt +
##     XP_missed + FG_missed + FUM + INT + Fourth_Dwn_Con + Third_Dwn_Con +
##     RedZone_Plays + Unique_Receivers + Shotgun_Plays
```

```
#Stepwise Both
step.both = step(step.back, scope = c(lower = Score_Dif ~ . - Game_id - Team_Name, upper = formula(Scor
formula(step.both)
```

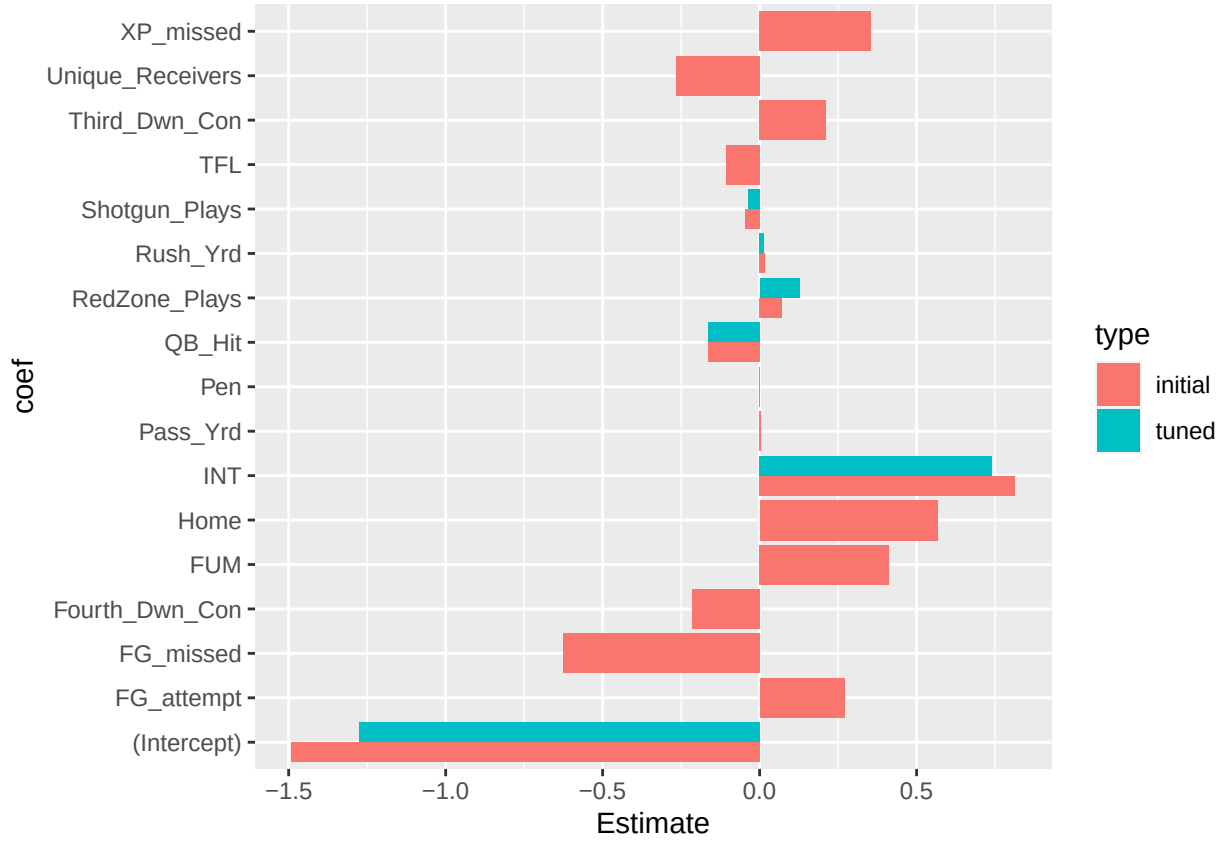
```
## Score_Dif ~ Home + QB_Hit + Pass_Yrd + Rush_Yrd + FG_attempt +
##     XP_missed + FG_missed + FUM + INT + Fourth_Dwn_Con + Third_Dwn_Con +
##     RedZone_Plays + Unique_Receivers + Shotgun_Plays
```

Logistic applications:

Although our main focus was predicting score differential, we thought that predicting the probability of a win would be an interesting extension to our project, and would highlight some of the oversights of our models as well as some interesting scenarios that just can't be handled meaningfully (looking at you, Bill Belichick). In this section, we fit a glm model with the parameter of being in the binomial family for the

response, which in this case is wins. Our assumptions are minimal, as seen in the lectures, and all we need is independent observations which we generally have. We recorded each teams' wins for each game, to be used as a response. Then, we created two glm's, one with the top 5 predictors from the random forest variable importance (Rush Yards, RedZone Plays, Qb Hits, Shotgun plays, and interceptions) as well as a model with all predictors except for the game id, year, team, and score differential. Using these models, we could create models showing what things contributed to the probability of a win. Doing this, we could compare the two models' coefficients and see what changed from model to model:

```
data_2017_19_test = data.frame(data_2017_19)

data_2017_19_test$wins = ifelse(data_2017_19$Score_Dif > 0, 1, 0)

win.log = glm(wins~.-Score_Dif-year, data=data_2017_19_test[,seq(3, ncol(data_2017_19_test))], family="

best.log = glm(wins~RedZone_Plays + Rush_Yrd+Shotgun_Plays +QB_Hit+INT, data = data_2017_19_test, famil

init_coef_log = data.frame(summary(win.log)$coefficients)
best_coef_log = data.frame(summary(best.log)$coefficients)

init_coef_log$coef = row.names(init_coef_log)
best_coef_log$coef = row.names(best_coef_log)
init_coef_log$type = "initial"
best_coef_log$type = "tuned"

ggplot(rbind(init_coef_log, best_coef_log), aes(fill=type, y=Estimate, x=coef)) +
    geom_bar(position="dodge", stat="identity") + coord_flip()
```



Our plot shows most notably that the intercept lost magnitude when predictors were removed, and as a result changed the magnitude of the kept predictors. Most importantly of which was rushing yards, whose coefficient decreased a lot from the initial model (which may not look like a ton on this plot). The decrease from the initial model to the tuned model was 0.016 to 0.013, and is significant because rushing yards are usually on the scale of about 120 to 180 per team per game.

Interestingly, both models had a worse train accuracy than test accuracy, which was in the years 2017-2019, as seen in the table below. Showing that our model wasn't too overfit and did have some predictive power.

```
new_2021 = data.frame(data_2021)
new_2021$year = 2021

new_2021$wins = ifelse(data_2021$Score_Dif > 0, 1, 0)

predicted_wins_initial = 1*(predict(win.log, type = "response")>0.5)
predicted_wins_best = 1*(predict(best.log, type = "response")>0.5)

Train_Acc_Init = sum(predicted_wins_initial == data_2017_19_test$wins)/nrow(data_2017_19_test)
Train_Acc_Best = sum(predicted_wins_best == data_2017_19_test$wins)/nrow(data_2017_19_test)

pred_test_init = 1*(predict(win.log, newdata=new_2021, type = "response")>0.5)
pred_test_best = 1*(predict(best.log, newdata=new_2021, type="response")>0.5)

Test_Acc_Init = sum(pred_test_init == new_2021$wins)/nrow(new_2021)
Test_Acc_Best = sum(pred_test_best == new_2021$wins)/nrow(new_2021)

data.frame(Train_Acc_Init, Train_Acc_Best, Test_Acc_Init, Test_Acc_Best)
```

```
##   Train_Acc_Init Train_Acc_Best Test_Acc_Init Test_Acc_Best
## 1      0.7940075      0.7509363      0.806701     0.7860825
```

Using these models, we were able to explore some interesting extensions, such as the Patriots week 4 bout versus the Buccaneers. In this game, the Patriots exceeded expectations and held the defending super bowl champions to 6 points in the first half, and 13 in the second, good for a win in many weeks. Importantly, they played them close all game and the Bucs won only on a last second field goal. What is strange, however, is that our model showed the Patriots as having a 3.78% probability of a win in the tuned model, and a 1.21% chance in the initial model. The Bucs were given a 70.2% and 86.5% probability in the tuned and untuned model, respectively. So, why did our model completely miss on a game that, honestly, should have been won by the Patriots? Taking a closer look, we can see that the Pats had -1 rush yards (no, not an error). Our model heavily favors rush yards and gives passing yards almost no weights, especially in the tuned model where passing yards meant nothing. The Patriots often perform unusual game plans, such as only throwing 3 times in a 14-10 win versus the Buffalo Bills on December 6, 2021. These game plans clearly throw off our model, showing that it really isn't robust to model a win probability logistically off of in game data (for the Patriots, at least). There's something about the "eye test" (a theoretical, informal "test" that involves years of watching and analyzing games, making snap judgements about outcomes and decisions while watching the game). Someone watching the week 4 Patriots-Bucs game surely wouldn't be giving the Patriots a ~4% chance of winning after seeing them absolutely dominate in the first half, yet our model says that not having any rush yards is highly correlated with losing the game.

Importantly, the model did actually get that prediction correct, but odd scenarios like this can mess with the model's accuracy (off the top of my head, the falcon's game versus the saints where they had 34 rushing yards and won is an example of a false prediction). Overall, rating one metric too highly can mess with the model, just as in any model, so we must be careful about determining direct causation.

In the long run, we'd be able to claim that rushing for more yards likely means you are winning and don't have to pass, which points to there being a win, but a team could just rush for every play and rack up the

yards and still lose, which is why we cannot say that rushing yards increase your chances of winning, but rather we associate higher amounts of rushing yards with a higher chance of winning.

```
ne.game.init = predict(win.log, newdata = new_2021[new_2021$Game_id=="2021_04_TB_NE" & new_2021$Home==1
```

```
##        123
## 0.01212202
```

```
tb.game.init = predict(win.log, newdata = new_2021[new_2021$Game_id=="2021_04_TB_NE" & new_2021$Home==0
```

```
##       124
## 0.8652279
```

```
ne.game.best = predict(best.log, newdata = new_2021[new_2021$Game_id=="2021_04_TB_NE" & new_2021$Home==
```

```
##       123
## 0.0378281
```

```
tb.game.best = predict(best.log, newdata = new_2021[new_2021$Game_id=="2021_04_TB_NE" & new_2021$Home==
```

```
##       124
## 0.7022904
```

```
data.frame(NE_Chance_Tuned = ne.game.best, NE_Chance_Untuned = ne.game.init, TB_Chance_Tuned = tb.game.
```

```
##  NE_Chance_Tuned NE_Chance_Untuned TB_Chance_Tuned TB_Chance_Untuned
##        0.0378281        0.01212202       0.7022904         0.8652279
```