Zane Reeves: Final Project

2024-12-13

Introduction

Counter Strike 2 (formerly Counter Strike: Global Offensive), is a 5v5 first person shooter video game with a lively esports scene. Typically, at most, 24 rounds are played in a competitive match. Assuming no overtimes, the team winning 13 rounds first is declared the winner of a match. At the start of a game, the two teams are randomly placed as either a terrorist or a counter-terrorist task force—with both teams having a unique set of weapons to use. After 12 rounds however, the teams assigned to the terrorist/counter-terrorist task forces switch sides, providing some symmetry in every game. Furthermore, weaponry is bought with currency that is gained after a round is finished and resets to a predefined amount at the start of every match. If a team wins a round in a match they gain additional currency that can be used to buy better weapons, while the team that loses gains significantly less money on round end. After a player dies in a round, the weapons the player was holding is lost and must be rebought at the start of the next round. Each individual round is won by either: one team eliminating all players on the opposing team, the counter-terrorists stalling the terrorists until a predefined timer runs out, the terrorists plant a bomb and stall the counter-terrorist team from defusing the bomb, or the counter-terrorists defuse the bomb.

Goals

Our goals for this study is to find which features affect the outcome of a match between teams t1/t2. The predictive terms we will consider in this preliminary study are team t1's world rankings and team t2's world rankings, as well as the ELO score of each player on both teams. Additionally, it is important to note that our dataset only tracks teams ranked top twenty world wide from the years 2016-2020.

Given that online sports betting in general is growing larger as an industry throughout America since its legalization, considering how we could determine match outcomes from readily available pre-game team data could help in the future as an analyst at a gambling firm.

The source of my data is given by: https://www.kaggle.com/datasets/gabrieltardochi/counter-strike-global-offensive-matches/data

Getting a grasp on the dataset

We wish to model the probability that team t1 wins, to do this we will perform logistic regression. Our dataset, hereby refered to as csgo, has many columns containing unique features associated with the outcome of each game. In order to perform regression on these parameters, we must first modify our original dataset such that the winning team (t1/t2) isn't associated with the string "t1" or "t2" but rather a value that is either zero or one.

```
# if t1 wins = 0, else 1
csgo <- read.csv("csgo_games.csv") %>% mutate(winner =
                                                if_else(winner == "t1",
                                                        0, 1)
                                              )
print(length(csgo[,1]))
```

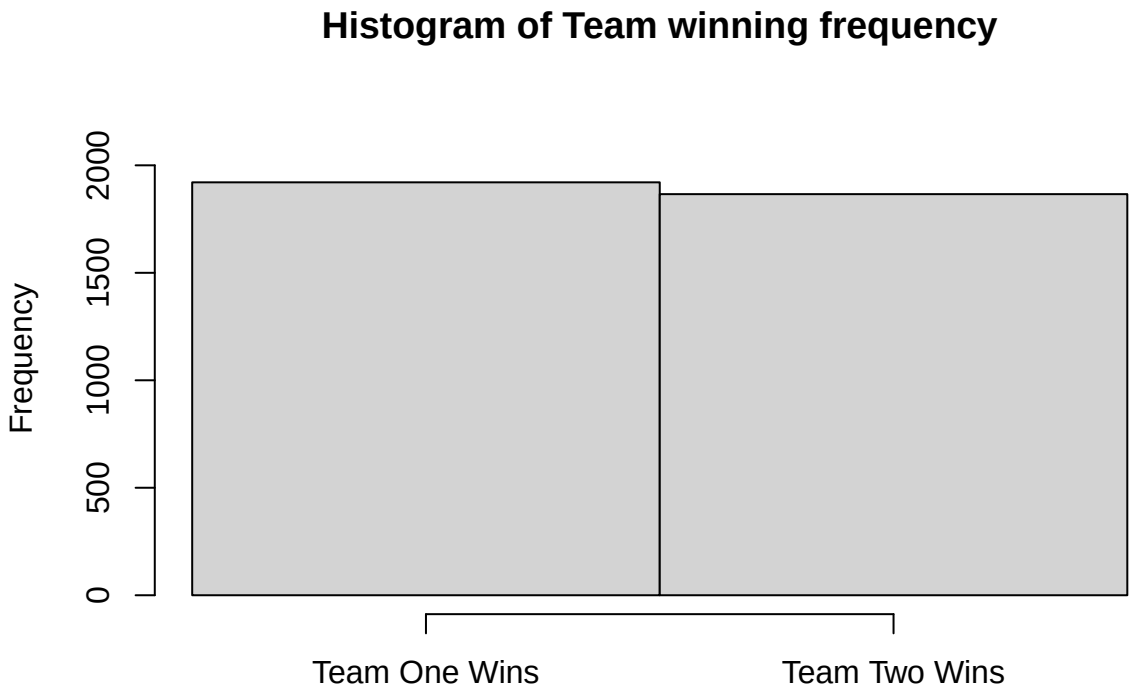
```
## [1] 3787
```

We add a winner column to csgo that is 0 if t1 wins and 1 if t2 wins. There are 3787 samples in our dataset.

Team Win Rate

After these transformations we sanity check the dataset, we should expect to see roughly the same number of wins for both teams t1 and t2.

```
hist(csgo$winner,
     ylim = c(0, 2200),
     xlim=c(0,1),
     main="Histogram of Team winning frequency",
     axes=FALSE,
     breaks=2,
     xlab = "")

axis(1, at = c(.25,.75), labels=c("Team One Wins", "Team Two Wins"))
axis(2, at=NULL)
```



As you can see, the frequency of wins in our dataset is roughly equal for both t1 and t2.

```
1-mean(csgo$winner)
```

```
## [1] 0.5072617
```

team 1 winning makes up roughly 50% of our dataset.

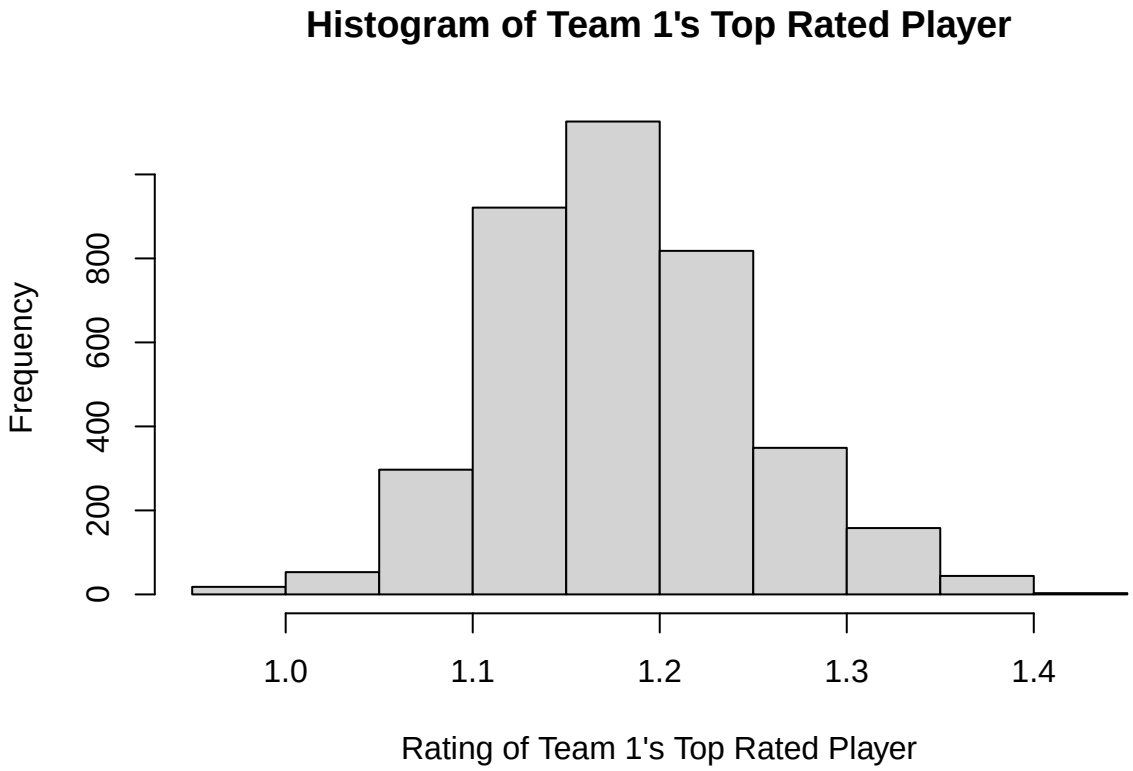
Player statistics.

We'll start by inspecting the distribution using histograms of the highest rated and worst rated players on both teams.

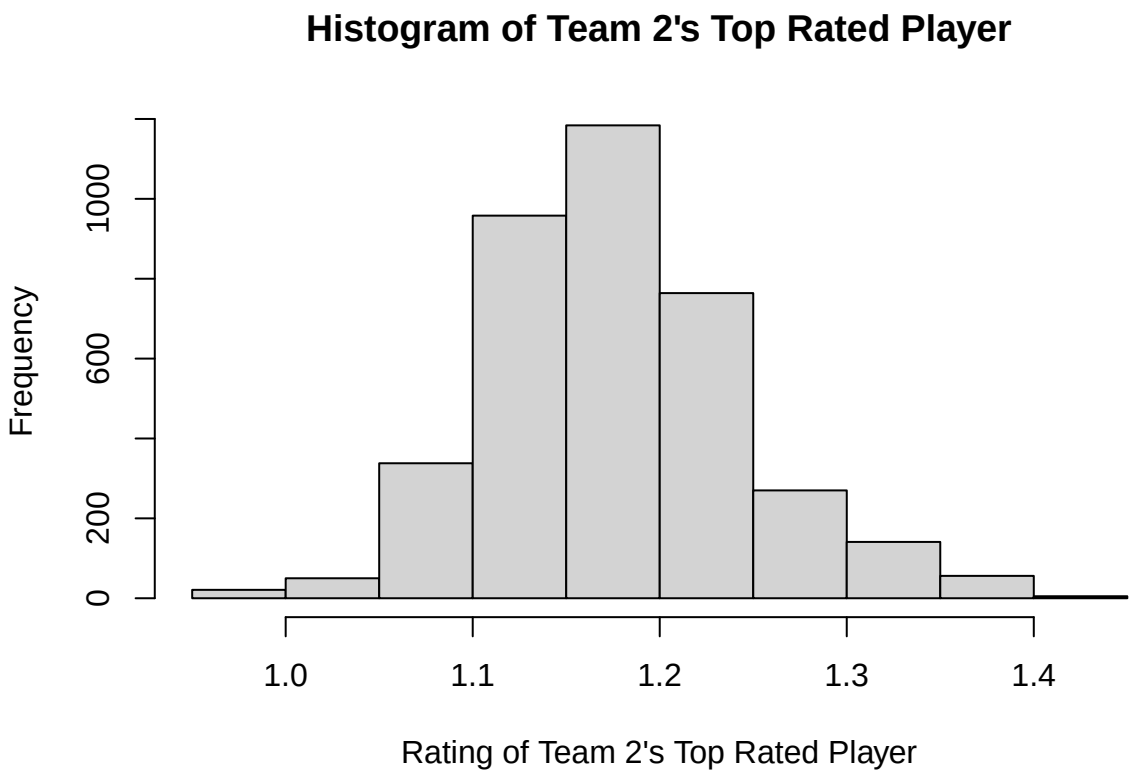
Note that each player is rated on HLTV's system (https://www.hltv.org/news/20695/introducing-rating-20) that is derived from ELO. This stat reflects what the expected contribution from a player is in terms of kills /deaths/accuracy when firing a weapon and various other in game metrics.

The player number out of five on a team represents their relative ranking to eachother.

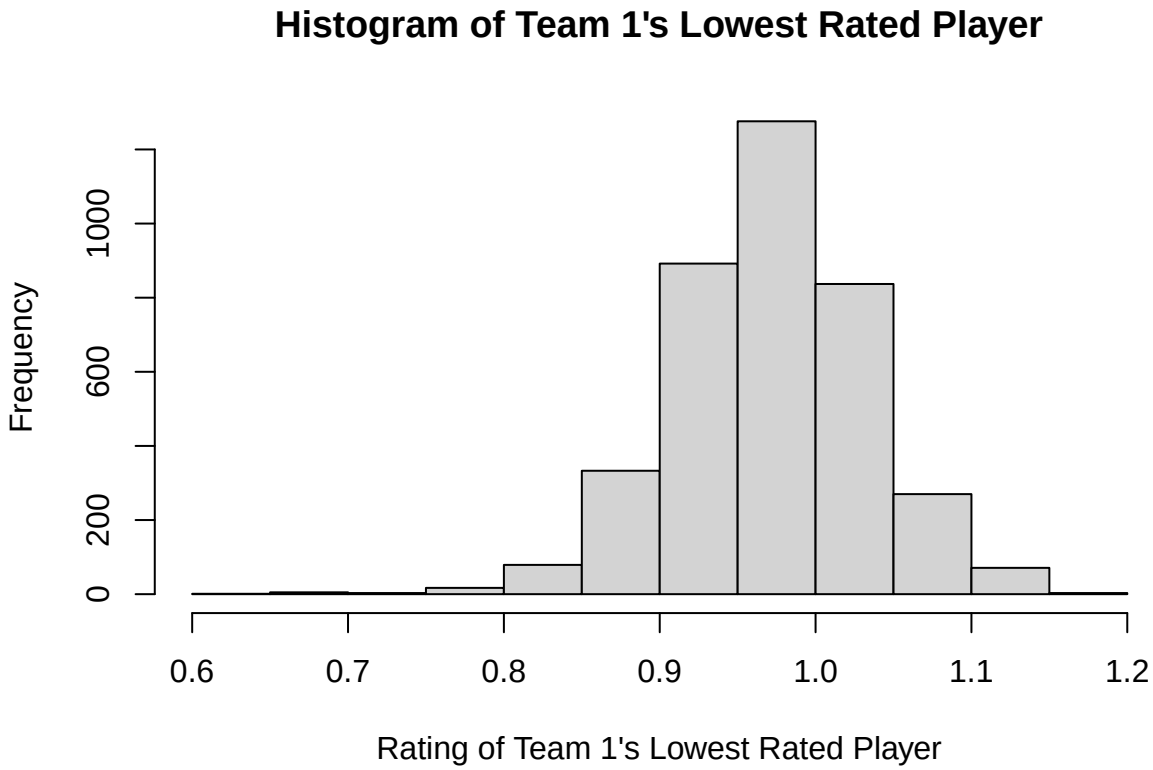
```
hist(csgo$t1_player1_rating,
     xlab = "Rating of Team 1's Top Rated Player",
     main = "Histogram of Team 1's Top Rated Player")
```



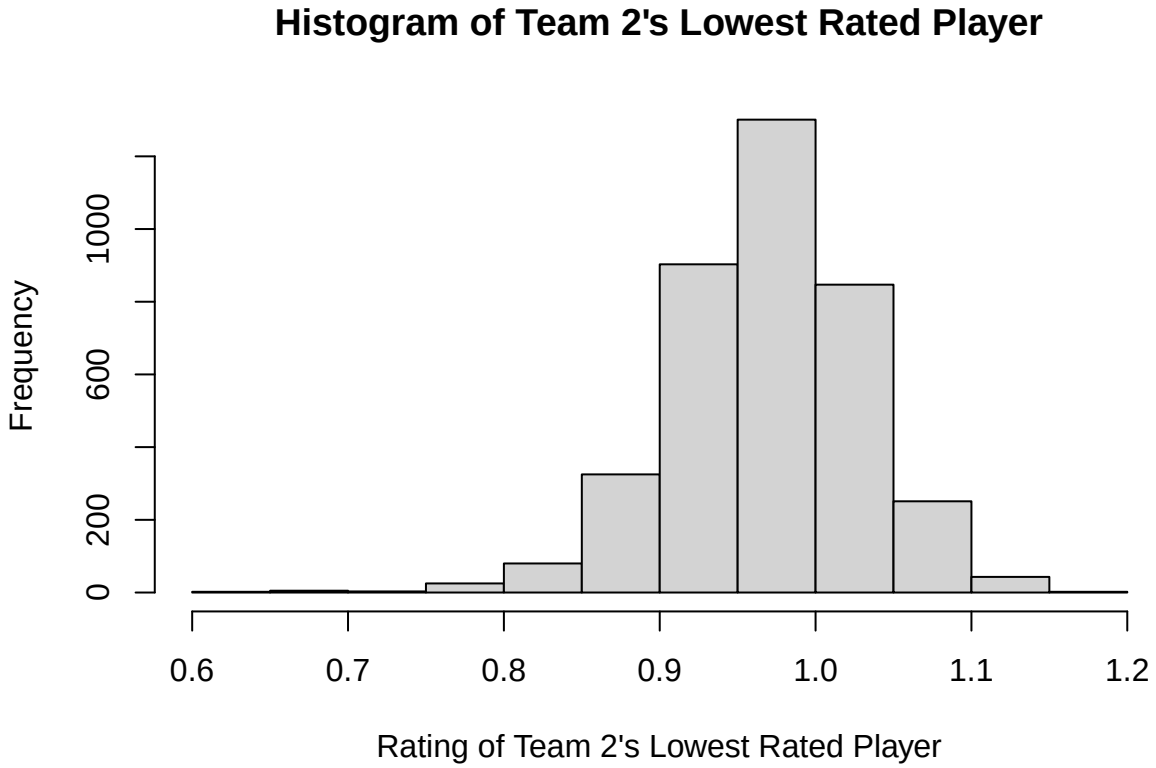

```
hist(csgo$t2_player1_rating,
     xlab = "Rating of Team 2's Top Rated Player",
     main = "Histogram of Team 2's Top Rated Player")
```




```
hist(csgo$t1_player5_rating,
     xlab = "Rating of Team 1's Lowest Rated Player",
     main = "Histogram of Team 1's Lowest Rated Player")
```



```r
hist(csgo$t2_player5_rating,
     xlab = "Rating of Team 2's Lowest Rated Player",
     main = "Histogram of Team 2's Lowest Rated Player")
```



These histograms pass the intuition check as we expect the mean ELO rating of the highest rated member of each team to be higher than the mean ELO rating of the lowest ranked member on each team.

Regressing on Team's world rank

Each choice of winning/losing is independent within our dataset as the samples are taken at different times with the outcome of each game not affecting the outcome of another. The outcome of each game is either a win/loss. We'll assume error is logistically distributed and that error is random and independent within our dataset and that the log odds of winner is linear w.r.t it's relation- ship with other parameters.

Our dataset only considers the top 20 teams world wide throughout it's 4 year period of data collection, therefore each teams world rank is chosen from [1,20].

Therefore using the above facts/assumptions, we'll perform logistic regression on winner vs team t1's world rank.

```r
reg <- stan_glm(winner ~ t1_world_rank,
                family=binomial(link="logit"),
                data=csgo,
                refresh = 0)
```

We'll now capture the loo score on our regressor.

```r
loo_reg <- loo(reg)
```

From here we'll perform logistic regression using the world rankings of t1, t2, and their interaction as predictors for the winner of a game.

```
reg_interac <-stan_glm(winner ~ t1_world_rank*t2_world_rank,
                family=binomial(link="logit"),
                data=csgo,
                refresh = 0)
```

```
loo_reg_inter <- loo(reg_interac)
```

```
loo_compare(loo_reg, loo_reg_inter)
```

```
##             elpd_diff se_diff
## reg_interac   0.0       0.0
## reg         -28.6       7.8
```

LOO scores nearer to zero correspond to better model fit, therefore the model that takes into account both team's rankings and their interactions performs better than the regressor that only takes into account a single team's world ranking. Although, due to the issue of multicollinearity and overfitting, the regressor on a singular independent variable is still competitive due to its simplicity.

```
reg_interac$coefficients
```

```
##                 (Intercept)               t1_world_rank
##               -0.0559532764                0.0542520918
##               t2_world_rank t1_world_rank:t2_world_rank
##               -0.0493568161                0.0003792491
```

We see that the coefficients for the interaction term are near zero, this implies that the interaction term has a marginal affect on predicting match outcome and may be excessive to include in a model that describes match outcomes.

We'll use the divide by four rule to approximate the 95% confidence intervals for the percentage affect per unit of the world rank for t1. Furthermore, we'll also find the expected affect of the coefficients on the percentage odds per unit.

```
divide_by_four <- function(se, coeff) {
  mean <- (coeff)/4
  diff <- se/2
  lower_upper_95 <- c(mean + diff, mean - diff)
  c(mean, lower_upper_95)
}

se_t1 <- reg$ses["t1_world_rank"]
t1_coeff <- reg$coefficients["t1_world_rank"]

dbf_t1 <- divide_by_four(se_t1, t1_coeff)
names(dbf_t1) <- NULL

# Mean, and lower/upper 95% confidence interval for team 1 rank coefficient
print(paste("Difference of win odds given unit change in world rank for team 1: ",
            dbf_t1[1]))
```

```
## [1] "Difference of win odds given unit change in world rank for team 1:  0.0132273046183838"
```

```
print(paste("Divide by four 95% CI Upper bound team 1: ", dbf_t1[2]))
```

```
## [1] "Divide by four 95% CI Upper bound team 1:  0.0162661057837186"
```

```
print(paste("Divide by four 95% CI Lower bound team 1: ", dbf_t1[3]))
```

```
## [1] "Divide by four 95% CI Lower bound team 1:  0.0101885034530491"
```

We find that both 95% CI of the coefficients of team 1's world ranking doesn't include zero, a good sign that the coefficient is statistically significant.

Regressing on Player ELO

We'll now perform logistic regression using winner as the dependent variable and the top ranking player from both teams as the predictors. We'll make the same assumptions of the dataset as we did when we treated both team rankings as the logistic predictors for which team would win.

```
reg_players_2 <-stan_glm(winner ~ t1_player1_rating + t2_player1_rating,
                family=binomial(link="logit"),
                data=csgo,
                refresh = 0)
```

```
loo_players_2 <- loo(reg_players_2)
loo_compare(loo_players_2, loo_reg_inter)
```

```
##                elpd_diff se_diff
## reg_interac      0.0       0.0
## reg_players_2 -35.5      13.0
```

Comparing the LOO fit of the model which only considers the top player rankings from each team to the model that considers the interaction between the team's world rankings; we find that the model on player rankings is slightly out performed.

We'll now consider a model that takes into account all player rankings from both teams.

```
reg_players_10 <-stan_glm(winner ~
                          t1_player1_rating + t2_player1_rating +
                          t1_player2_rating + t2_player2_rating +
                          t1_player3_rating + t2_player3_rating +
                          t1_player4_rating + t2_player4_rating +
                          t1_player5_rating + t2_player5_rating,
                family=binomial(link="logit"),
                data=csgo,
                refresh = 0)
```

We compare this model to our original base-line interaction model.

```
loo_players_10 <- loo(reg_players_10)
loo_compare(loo_players_10, loo_reg_inter)
```

```
##                 elpd_diff se_diff
## reg_interac      0.0       0.0
## reg_players_10 -0.3      14.4
```

We find that the model that considers all player rankings is competitive to our baseline interaction model.

In light of the improved performance of the last model, we'll attempt to fit all player rankings and each team's world rankings against which team wins.

```
reg_players_10_ranks <-stan_glm(winner ~
                          t1_player1_rating + t2_player1_rating +
                          t1_player2_rating + t2_player2_rating +
                          t1_player3_rating + t2_player3_rating +
                          t1_player4_rating + t2_player4_rating +
                          t1_player5_rating + t2_player5_rating +
                          t1_world_rank + t2_world_rank,
                family=binomial(link="logit"),
```

```
                data=csgo,
                refresh = 0)
```

```
loo_players_10_rank <- loo(reg_players_10_ranks)
loo_compare(loo_players_10_rank, loo_reg_inter)
```

```
##                       elpd_diff se_diff
## reg_players_10_ranks   0.0       0.0
## reg_interac          -34.7       9.3
```

We find that this model that considers many terms out performs our baseline interaction model.

We will now perform the divide by four rule to derive an estimate of percentage per unit change for which team wins depending on the rank of each team's top player..

```
se_pt1 <- reg_players_10_ranks$ses["t1_player1_rating"]
se_pt2 <- reg_players_10_ranks$ses["t2_player1_rating"]

pt1_coeff <- reg_players_10_ranks$coefficients["t1_player1_rating"]
pt2_coeff <- reg_players_10_ranks$coefficients["t2_player1_rating"]

dbf_pt1 <- divide_by_four(se_pt1, pt1_coeff)
dbf_pt2 <- divide_by_four(se_pt2, pt2_coeff)
names(dbf_pt1) <- NULL
names(dbf_pt2) <- NULL

# Mean, and lower/upper 95% confidence interval for t1's top player (divide by four)
print("Difference of win odds given unit change in world rank for team 1's top player: ")
```

```
## [1] "Difference of win odds given unit change in world rank for team 1's top player: "
```

```
print(dbf_pt1[1])
```

```
## [1] -0.3180832
```

```
print(paste("Divide by four 95% CI Upper bound team 1's top player: ",
            dbf_pt1[2]))
```

```
## [1] "Divide by four 95% CI Upper bound team 1's top player:  0.00260175233218568"
```

```
print(paste("Divide by four 95% CI Lower bound team 1's top player: ",
            dbf_pt1[3]))
```

```
## [1] "Divide by four 95% CI Lower bound team 1's top player:  -0.638768218024604"
```

```
# Mean, and lower/upper 95% confidence interval for t2's top player (divide by four)
print("Difference of win odds given unit change in world rank for team 2's top player: ")
```

```
## [1] "Difference of win odds given unit change in world rank for team 2's top player: "
```

```
print(dbf_pt2[1])
```

```
## [1] 0.292266
```

```
print(paste("Divide by four 95% CI Upper bound team 2's top player: ",
            dbf_pt2[2]))
```

```
## [1] "Divide by four 95% CI Upper bound team 2's top player:  0.616338225250527"
```

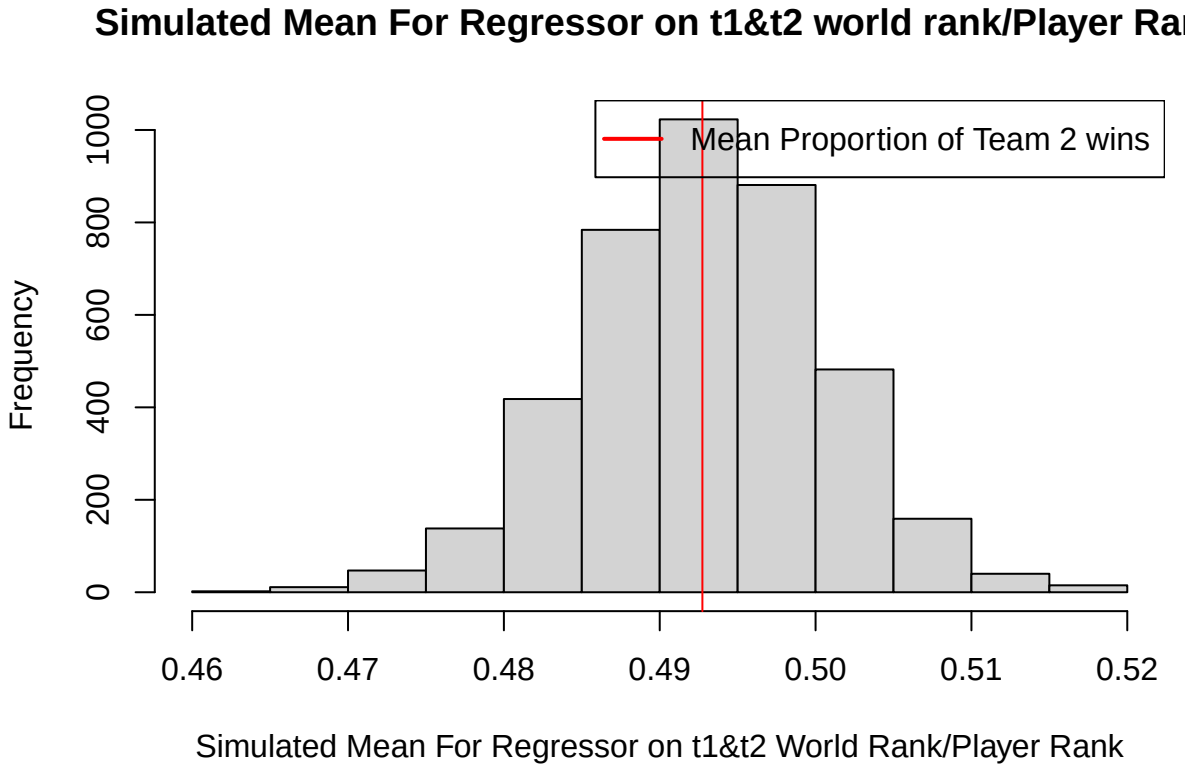
```
print(paste("Divide by four 95% CI Lower bound team 2's top player: ",
            dbf_pt2[3]))
```

```
## [1] "Divide by four 95% CI Lower bound team 2's top player:  -0.0318061397573616"
```

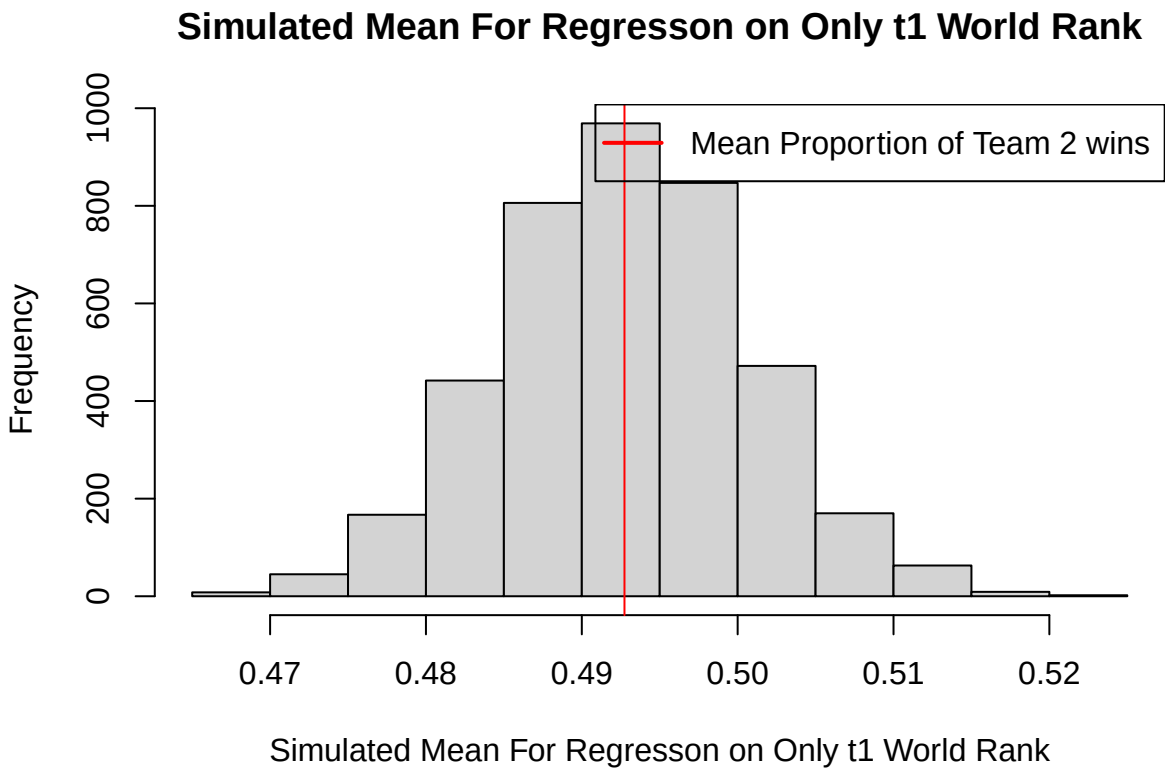
The 95% confidence interval for both coefficients of t1/t2's top players cross zero, this implies that there may be issues with the statistical significance of the coefficients. More analysis needs to be done to ascertain how significant the relationship between player ranking and match outcome is.

Checking the Best performing models

```
post_check_10 <- posterior_epred(reg_players_10_ranks,
                           draws = 4000)

post_check_simple <- posterior_epred(reg,
                           draws = 4000)

y_check_simple <- rowMeans(post_check_simple, 1)
y_check_10 <-rowMeans(post_check_10, 1)

y_sd_simp <- sqrt(y_check_simple*(1-y_check_simple))
y_sd_10 <- sqrt(y_check_10*(1-y_check_10))

obs_prop <- mean(csgo$winner)
obs_sd <- sd(csgo$winner)

hist(y_check_10,
     xlab="Simulated Mean For Regressor on t1&t2 World Rank/Player Rank",
     main="Simulated Mean For Regressor on t1&t2 world rank/Player Rank")
abline(v = obs_prop, col = "red")
legend("topright", legend = c("Mean Proportion of Team 2 wins"),
col = c("red"), pch = c(NA), lty = c( 1), lwd = c(2))
```




```
hist(y_check_simple,
     xlab="Simulated Mean For Regresson on Only t1 World Rank",
     main="Simulated Mean For Regresson on Only t1 World Rank")
abline(v = obs_prop, col="red")
legend("topright", legend = c("Mean Proportion of Team 2 wins"),
col = c("red"), pch = c(NA), lty = c( 1), lwd = c(2))
```




```
p_val_10 <- mean(obs_prop >= y_check_10)
p_val_simp <- mean(obs_prop >= y_check_simple)
```

```
# Proportion of simulated means from the regressor on all player ratings/team
# ratings that is less than observed mean
print(p_val_10)
```

```
## [1] 0.49075
```

```
# Proportion of simulated means from the regressor on only team 1's world rating
# that is less than observed mean
print(p_val_simp)
```

```
## [1] 0.50025
```

We see that both regressors have simulated means that are roughly centered about our observed mean in the above. This indicates proper simulation of our dataset using our models.

```
p_val_simp_sd <- mean(abs(y_sd_simp - obs_sd))
print(p_val_simp_sd)
```

```
## [1] 0.0001309172
```

```
p_val_10_sd <- mean(abs(y_sd_10 - obs_sd))
print(p_val_10_sd)
```

```
## [1] 0.0001252604
```

The mean difference in SD is very small (~0.0001318924) for both models against the true observed SD. This fact along with the fact that our models properly simulate the mean of our sample indicates that our models properly simulate and predict items trends our data.

We now perform Posterior Predictive Checking using the Null model (Linear Logit no intercept) on our data.

```r
null_model <- stan_glm(winner ~ 1,
                  family = binomial(link = "logit"),
                  data=csgo,
                  refresh=0)

null_preds <- posterior_epred(null_model)
null_check <-rowMeans(null_preds, 1)

sd_null <- sqrt(null_check*(1-null_check))

# Proportion of simulated means of Null model less than observed mean
print(mean(obs_prop >= null_check))
```

```
## [1] 0.50825
```

```r
# Mean magnitude difference of simulated SD of null model against observed SD
print(mean(abs(sd_null - obs_sd)))
```

```
## [1] 0.0001362993
```

Since the proportion of simulated means is roughly 1/2, and the magnitude difference of SD is roughly small (~.00012) we can infer that the Null model captures general behavior of our sample.

We will now perform LOO and compare the ELPDs acrossed our variance models.

```r
loo_null <- loo(null_model)
loo_compare(loo_null, loo_players_10_rank)
```

```
##                       elpd_diff se_diff
## reg_players_10_ranks    0.0       0.0
## null_model           -102.6      14.8
```

```r
loo_compare(loo_null, loo_reg)
```

```
##            elpd_diff se_diff
## reg          0.0       0.0
## null_model -39.2       8.9
```

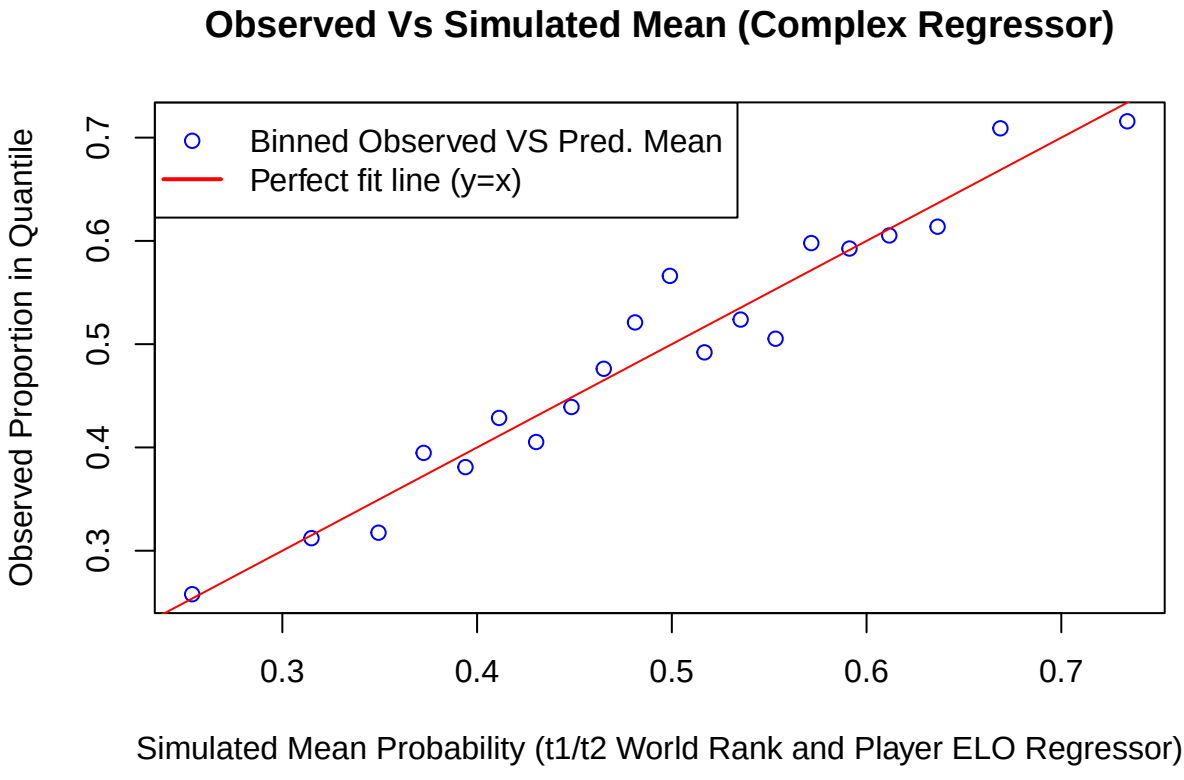
When well behaved, ELPD_diff is approximately normal, therefore since the diff. in ELPD is larger than 2 * SE_DIFF for both models we can be fairly certain that the difference in statistical power between the Null model and the regression on only world ranking and world ranking/player rankings is significant.

Inspecting Graphical Fit

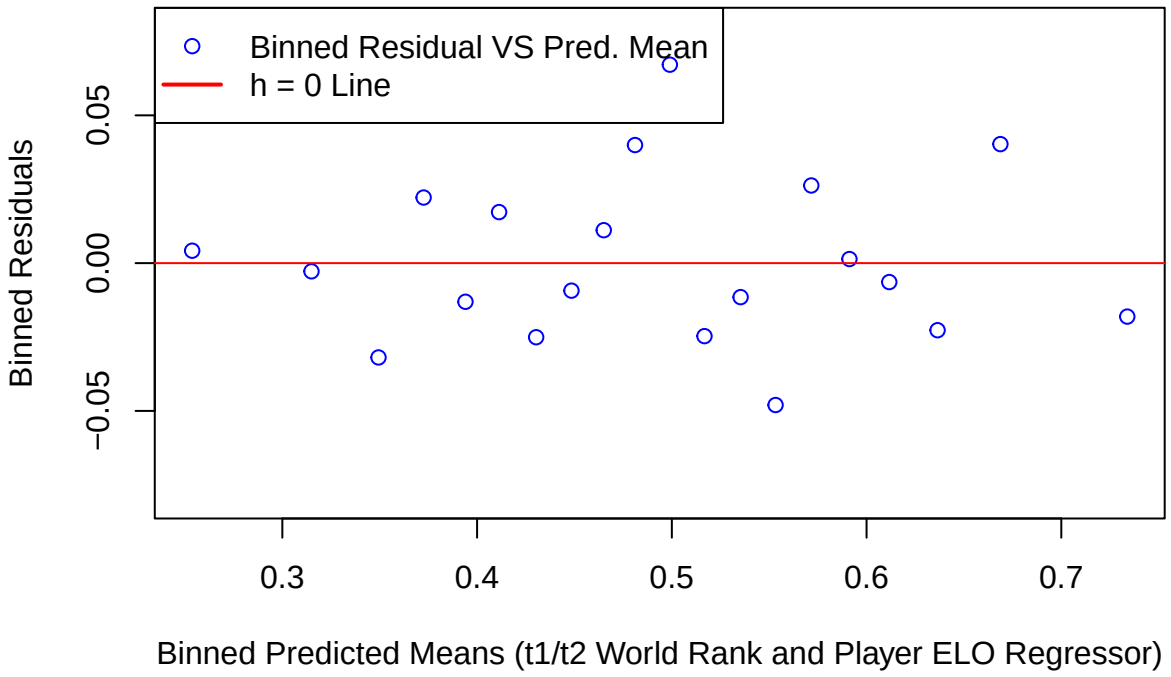
We will now inspect the graphical fit of the logistic regressor as well as the distribution of its binned Residuals.

```r
pred <- predict(reg_players_10_ranks, type="response")
n_bins = 20

prob <- cut(pred,
                breaks = quantile(pred,
                                  probs = seq(0,
                                              1,
                                              length.out = n_bins + 1)),
                include.lowest = TRUE)

bin_stats <- aggregate(cbind(csgo$winner, pred),
```

```
                        by = list(prob),
                        FUN = mean)

bin_stats <- do.call(data.frame, bin_stats)

names(bin_stats) <- c("Bin", "obs_mean", "pred_mean")

plot(bin_stats$pred_mean, bin_stats$obs_mean, pch = 1, col = "blue",
     xlab = "Simulated Mean Probability (t1/t2 World Rank and Player ELO Regressor)",
     ylab = "Observed Proportion in Quantile",
     main = "Observed Vs Simulated Mean (Complex Regressor)")
abline(0, 1, lty = 1, col = "red")
legend("topleft", legend = c("Binned Observed VS Pred. Mean", "Perfect fit line (y=x)"),
col = c("blue", "red"), pch = c(1, NA), lty = c(NA, 1), lwd = c(NA, 2))
```




```
plot(y=bin_stats$obs_mean - bin_stats$pred_mean,
     x= bin_stats$pred_mean,
     xlab = "Binned Predicted Means (t1/t2 World Rank and Player ELO Regressor)",
     ylab = "Binned Residuals",
     ylim = c(-.08, .08),
     col = "blue")

legend("topleft", legend = c("Binned Residual VS Pred. Mean",
                             "h = 0 Line"),
col = c("blue", "red"), pch = c(1, NA), lty = c(NA, 1), lwd = c(NA, 2))

abline(h=0, col = "red")
```



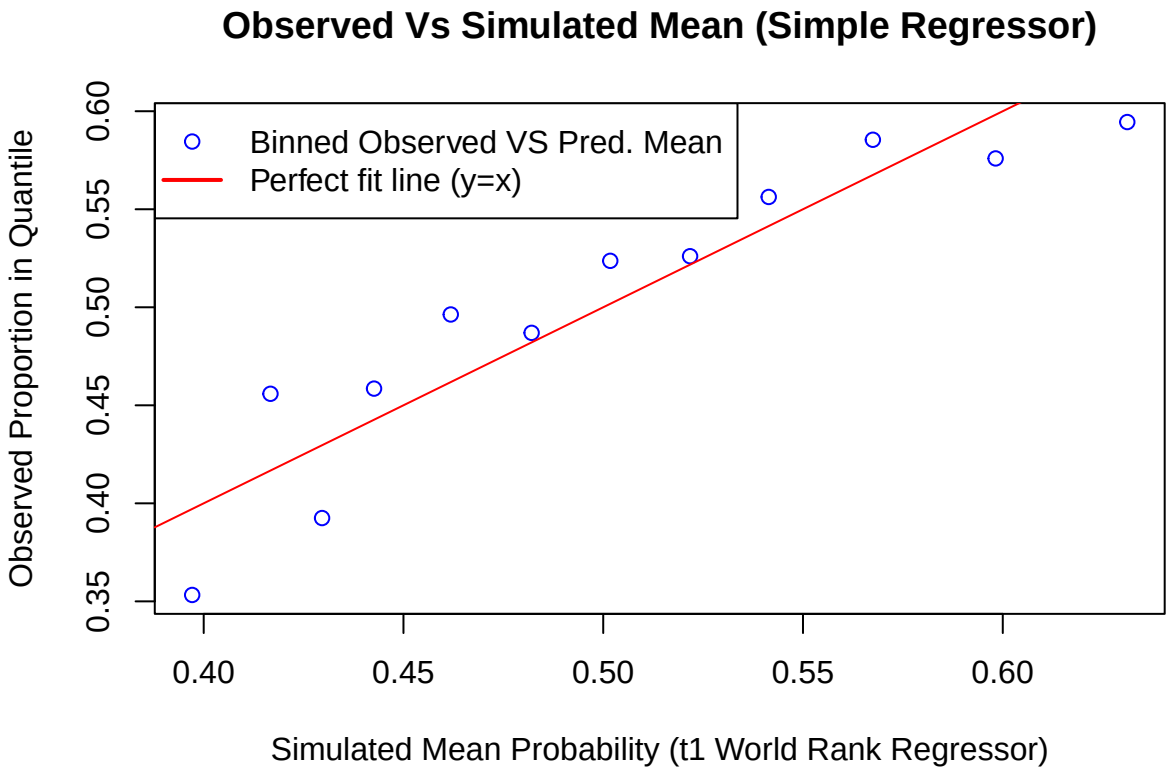
In the plot depicting Observed vs Predicted Mean, we bin our sample y acrossed the first 20 quantiles. We repeat this process for our predictions from our regressor that considers world ranking and player ranking. We then graph the mean acrossed these first 20 quantiles of the observed proportion against the predicted proportion. Datapoints above the line indicate under-predictions and below over-predictions, since our datapoints are all fairly close to the y=x line, we can deduce a good fit.

For the binned residuals plot, we see that the residuals are normally distributed around zero. This is a good sign that our data follows a linear model and exhibits uniform variance acrossed its datapoints.

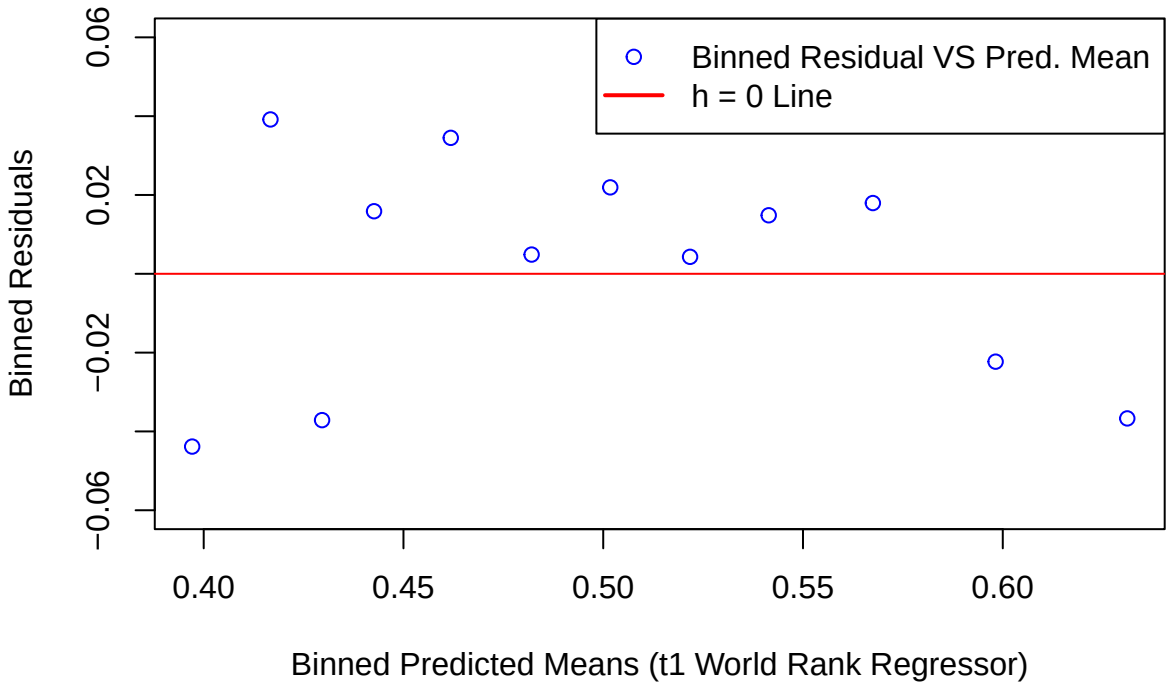
We repeat this process for the model that considers only the world ranking for both teams.

```
pred <- predict(reg, type="response")
n_bins = 12

prob <- cut(pred,
                breaks = quantile(pred,
                                  probs = seq(0,
                                              1,
                                              length.out = n_bins + 1)),
                include.lowest = TRUE)

bin_stats <- aggregate(cbind(csgo$winner, pred),
                       by = list(prob),
                       FUN = mean)

bin_stats <- do.call(data.frame, bin_stats)
names(bin_stats) <- c("Bin", "obs_mean", "pred_mean")

plot(bin_stats$pred_mean, bin_stats$obs_mean,
     pch = 1,
     col = "blue",
     xlab = "Simulated Mean Probability (t1 World Rank Regressor)",
     ylab = "Observed Proportion in Quantile",
     main = "Observed Vs Simulated Mean (Simple Regressor)")
```

```r
abline(0, 1, lty = 1, col = "red")
legend("topleft", legend = c("Binned Observed VS Pred. Mean", "Perfect fit line (y=x)"),
col = c("blue", "red"), pch = c(1, NA), lty = c(NA, 1), lwd = c(NA, 2))
```




```r
plot(y=bin_stats$obs_mean - bin_stats$pred_mean,
     x= bin_stats$pred_mean,
     xlab="Binned Predicted Means (t1 World Rank Regressor)",
     ylab="Binned Residuals",
     ylim=c(-.06, .06),
     col = "blue")

abline(h=0, col = "red")

legend("topright", legend = c("Binned Residual VS Pred. Mean",
                              "h = 0 Line"),
       col = c("blue", "red"), pch = c(1, NA),
       lty = c(NA, 1), lwd = c(NA, 2))
```



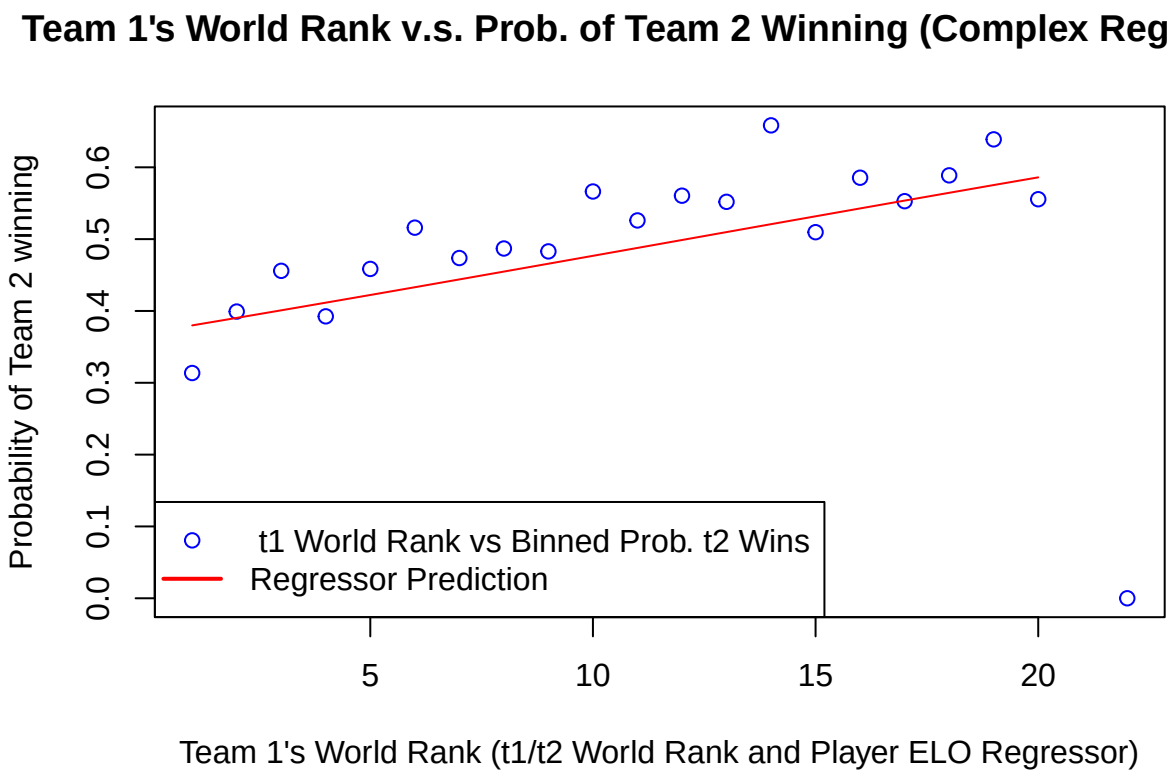
We see that the predicted first ten quantiles of our data-set closely resembles the observed mean of the first ten quantiles for the model that considers only team rankings. Furthermore, we observe that the binned residual plot is normally distributed with mean zero. This indicates that our data has a logistic relationship and has equal variance acrossed terms.

Predictions with fixed player/team 2 ranking and varing Team 1 ranking

In the below figure, we fix team 2's world ranking at 10, and all player rankings at the mean of team 1's top player. We than vary team 1's world ranking from 1 to 20 on the regressor that considers team ranking and player ranking for both teams.

```
input <-reg_players_10_ranks$data

bin_mean <- aggregate(winner ~ t1_world_rank, data=input, FUN = mean)

t2_rank <- 10
t1_rank <- seq(1, 20)
t1_p1_rank <- rep(mean(csgo$t1_player1_rating), 20)

toy_data <- data.frame(
  t1_world_rank = t1_rank,
  t2_world_rank = rep(t2_rank, 20),
  t1_player1_rating = t1_p1_rank,
  t1_player2_rating = t1_p1_rank,
  t1_player3_rating = t1_p1_rank,
  t1_player4_rating = t1_p1_rank,
  t1_player5_rating = t1_p1_rank,
  t2_player1_rating = t1_p1_rank,
  t2_player2_rating = t1_p1_rank,
  t2_player3_rating = t1_p1_rank,
```

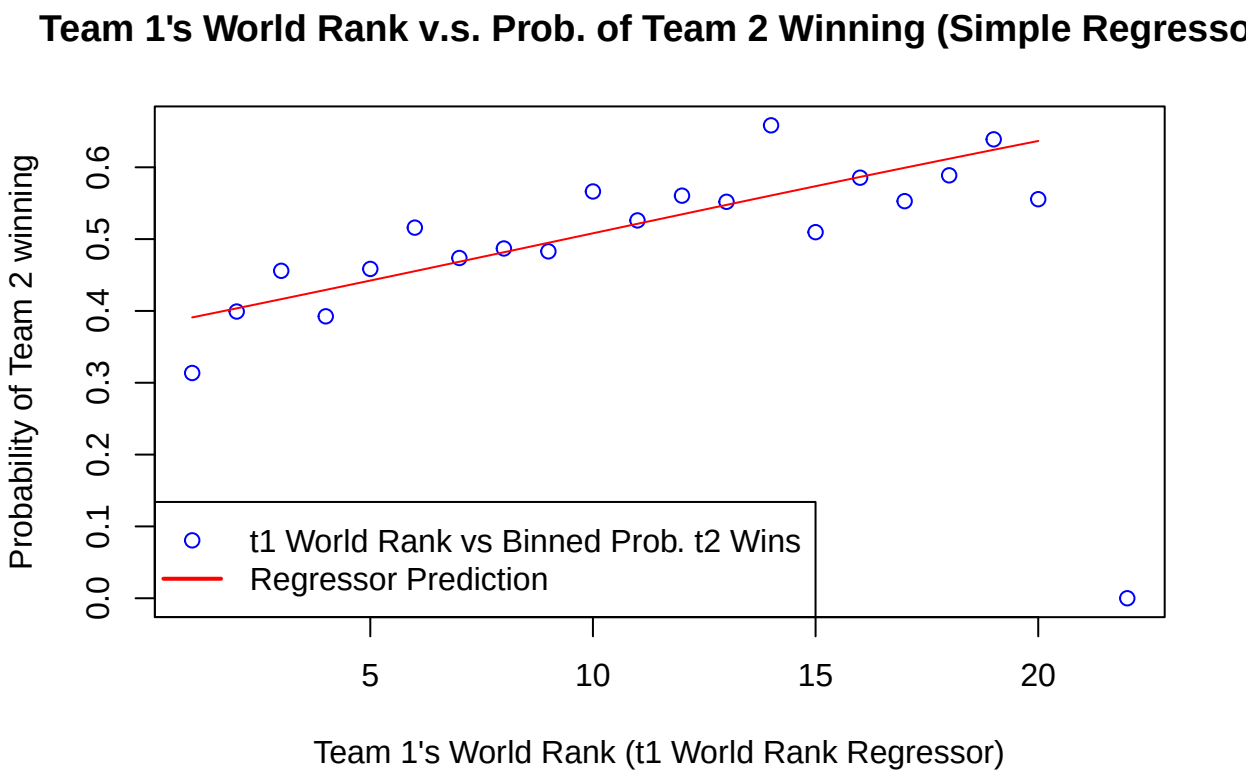
```
  t2_player4_rating = t1_p1_rank,
  t2_player5_rating = t1_p1_rank
)


reg_pred <- predict(reg_players_10_ranks, newdata = toy_data, type = "response")

plot(bin_mean$t1_world_rank,
     bin_mean$winner,
     xlab = "Team 1's World Rank (t1/t2 World Rank and Player ELO Regressor)",
     ylab = "Probability of Team 2 winning",
     main = "Team 1's World Rank v.s. Prob. of Team 2 Winning (Complex Regressor)",
     col = "blue",)

lines(toy_data$t1_world_rank,
      reg_pred,
      col="red")

legend("bottomleft", legend = c(" t1 World Rank vs Binned Prob. t2 Wins",
                                "Regressor Prediction"),
       col = c("blue", "red"),
       pch = c(1, NA),
       lty = c(NA, 1),
       lwd = c(NA, 2))
```



We see that this model only slightly under-predicts our binned probabilities. This could be due to assuming all players take on the mean of team 1's top player rating.

We now perform this same analysis for the regressor only considering the world ranking of team 1.

```r
input <-reg$data

bin_mean <- aggregate(winner ~ t1_world_rank, data=input, FUN = mean)

t2_rank <- 10
t1_rank <- seq(1, 20)
t1_p1_rank <- rep(mean(csgo$t1_player1_rating), 20)

toy_data <- data.frame(
  t1_world_rank = t1_rank,
  t2_world_rank = rep(t2_rank, 20)
)


reg_pred <- predict(reg, newdata = toy_data, type = "response")

plot(bin_mean$t1_world_rank,
     bin_mean$winner,
     xlab="Team 1's World Rank (t1 World Rank Regressor)",
     ylab = "Probability of Team 2 winning",
     main = "Team 1's World Rank v.s. Prob. of Team 2 Winning (Simple Regressor)",
     col = "blue")

lines(toy_data$t1_world_rank,
      reg_pred,
      col="red")

legend("bottomleft", legend = c("t1 World Rank vs Binned Prob. t2 Wins",
                                "Regressor Prediction"),
       col = c("blue", "red"),
       pch = c(1, NA),
       lty = c(NA, 1),
       lwd = c(NA, 2))
```



We see a much more centered prediction using the regressor on only team 1's world ranking.

Conclusion

We see that the model that considers only team 1's world rank, very accurately predicts a team's probability of winning with a singular predictor. Although, the regressor that considers both team's world rankings along with player ranks scores a high LOO score, there are statistical significance issues with it's coefficients as well as multicollinearity of its variables that could potentially damage the reliability of predictions outside of the training sample.

Furthermore, when we compared the model predicting on Team 1's world rank to the NULL model using LOO. We see statistically significant performance of our custom regressor over the Null model. Additionally, the residuals of our logistic regression are normally distributed in this model, indicating a proper fit with homogeneity. This satisfies several core assumptions necessary for our regressor on team 1's world rank to forecast data outside of the sample. And finally, when we applied the divide by four rule on this regressor we found that the 95% CI didn't cross zero for the coefficient, thus granting additional confidence in the ability of our regressor to predict future matches.

Overall, only making the assumptions that the log odds of t1 winning a game is linear with team 1's world ranking, each game played in the data set is independent, and since we are only considering a singular independent variable no collinearity can exist acrossed variables. We can infer future outcomes of games given only a singular team's world rank.

Although, there may be potential issues considering match ups of teams that are not strictly within the top twenty world wide. As our regressor is trained on match outcomes of only those games played between the teams ranked twenty world wide. In the future, to improve prediction of games where both teams are not elite on the world scene, it would be enlightening to fit a model on data that includes game outcomes and data of teams that are not strictly top twenty.

Coefficients of final Regressors for Recreation of Study

```
reg$coefficients
```

```
##   (Intercept) t1_world_rank
##   -0.49500689    0.05290922
```

```
summary(reg)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       binomial [logit]
##  formula:      winner ~ t1_world_rank
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 3787
##  predictors:   2
##
## Estimates:
##                mean   sd   10%   50%   90%
## (Intercept)   -0.5    0.1 -0.6  -0.5  -0.4
## t1_world_rank  0.1    0.0  0.0   0.1   0.1
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 0.5    0.0  0.5   0.5   0.5
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##               mcse Rhat n_eff
## (Intercept)   0.0  1.0  2880
## t1_world_rank 0.0  1.0  2957
## mean_PPD      0.0  1.0  3318
## log-posterior 0.0  1.0  1664
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```
reg_players_10_ranks$coefficients
```

```
##       (Intercept) t1_player1_rating t2_player1_rating t1_player2_rating
##        0.40980575       -1.27233293        1.16906417       -0.84606820
## t2_player2_rating t1_player3_rating t2_player3_rating t1_player4_rating
##        1.29988770        0.32834048       -0.43972088       -1.54292244
## t2_player4_rating t1_player5_rating t2_player5_rating     t1_world_rank
##        1.35711601       -2.22490851        1.67196318        0.04446329
##     t2_world_rank
##       -0.03637415
```

```
summary(reg_players_10_ranks)
```

```
##
## Model Info:
##  function:     stan_glm
```

```
##  family:       binomial [logit]
##  formula:      winner ~ t1_player1_rating + t2_player1_rating + t1_player2_rating +
##     t2_player2_rating + t1_player3_rating + t2_player3_rating +
##     t1_player4_rating + t2_player4_rating + t1_player5_rating +
##     t2_player5_rating + t1_world_rank + t2_world_rank
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 3787
##  predictors:   13
##
## Estimates:
##                     mean   sd   10%   50%   90%
## (Intercept)        0.4    1.1 -1.0   0.4   1.8
## t1_player1_rating -1.3    0.6 -2.1  -1.3  -0.5
## t2_player1_rating  1.2    0.7  0.3   1.2   2.0
## t1_player2_rating -0.9    1.1 -2.2  -0.8   0.5
## t2_player2_rating  1.3    1.1 -0.1   1.3   2.7
## t1_player3_rating  0.3    1.2 -1.2   0.3   1.9
## t2_player3_rating -0.5    1.2 -2.0  -0.4   1.1
## t1_player4_rating -1.5    1.1 -2.9  -1.5  -0.1
## t2_player4_rating  1.4    1.1  0.0   1.4   2.7
## t1_player5_rating -2.2    0.7 -3.1  -2.2  -1.3
## t2_player5_rating  1.7    0.7  0.8   1.7   2.6
## t1_world_rank      0.0    0.0  0.0   0.0   0.1
## t2_world_rank      0.0    0.0  0.0   0.0   0.0
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 0.5    0.0  0.5   0.5   0.5
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##                    mcse Rhat n_eff
## (Intercept)        0.0  1.0  5032
## t1_player1_rating 0.0  1.0  4098
## t2_player1_rating 0.0  1.0  4373
## t1_player2_rating 0.0  1.0  3143
## t2_player2_rating 0.0  1.0  3062
## t1_player3_rating 0.0  1.0  3106
## t2_player3_rating 0.0  1.0  3133
## t1_player4_rating 0.0  1.0  3363
## t2_player4_rating 0.0  1.0  3726
## t1_player5_rating 0.0  1.0  4470
## t2_player5_rating 0.0  1.0  5277
## t1_world_rank     0.0  1.0  6663
## t2_world_rank     0.0  1.0  5514
## mean_PPD          0.0  1.0  4187
## log-posterior     0.1  1.0  1668
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```