Expected ISO

This project was originally designed for use at the university level and aimed to utilize Trackman data to quantify an expected ISO on contact. Throughout this project, the necessary metrics are labeled with names such as xBA instead xBAcon, but we are ultimately focusing just on contact. The purpose of this is to create an easy way to pull a quantified "raw power" estimate for any opponent. This makes it easier to know which opposing hitters should be noted for their power potential on our scouting reports.

In order to develop expected ISO, we obviously need to develop expected Batting Average and expected Slugging on contact. In the future, these will be referred to as xBA and xSLG. The first step is to take the complete Trackman dataset from the 2018-2019 season that is available. Unfortunately, this dataset is limited to teams that have entered our data sharing agreement, meaning this sample size is relatively small for this type of analysis. I then subseted this dataset to only batted ball events and assigned batting average and slugging percentage values for every event.

```
cbb_contact <- subset(cbb_complete, !is.na(cbb_complete$ExitSpeed))
cbb_inplay <- subset(cbb_contact, cbb_contact$PitchCall == 'InPlay')
cbb_inplay$EventID <- c(1:nrow(cbb_inplay))

cbb_inplay$ba <- 0
cbb_inplay$ba[cbb_inplay$PlayResult == 'Single'] <- 1
cbb_inplay$ba[cbb_inplay$PlayResult == 'Double'] <- 1
cbb_inplay$ba[cbb_inplay$PlayResult == 'Triple'] <- 1
cbb_inplay$ba[cbb_inplay$PlayResult == 'HomeRun'] <- 1

cbb_inplay$slg <- 0
cbb_inplay$slg[cbb_inplay$PlayResult == 'Single'] <- 1
cbb_inplay$slg[cbb_inplay$PlayResult == 'Double'] <- 2
cbb_inplay$slg[cbb_inplay$PlayResult == 'Triple'] <- 3
cbb_inplay$slg[cbb_inplay$PlayResult == 'HomeRun'] <- 4
```

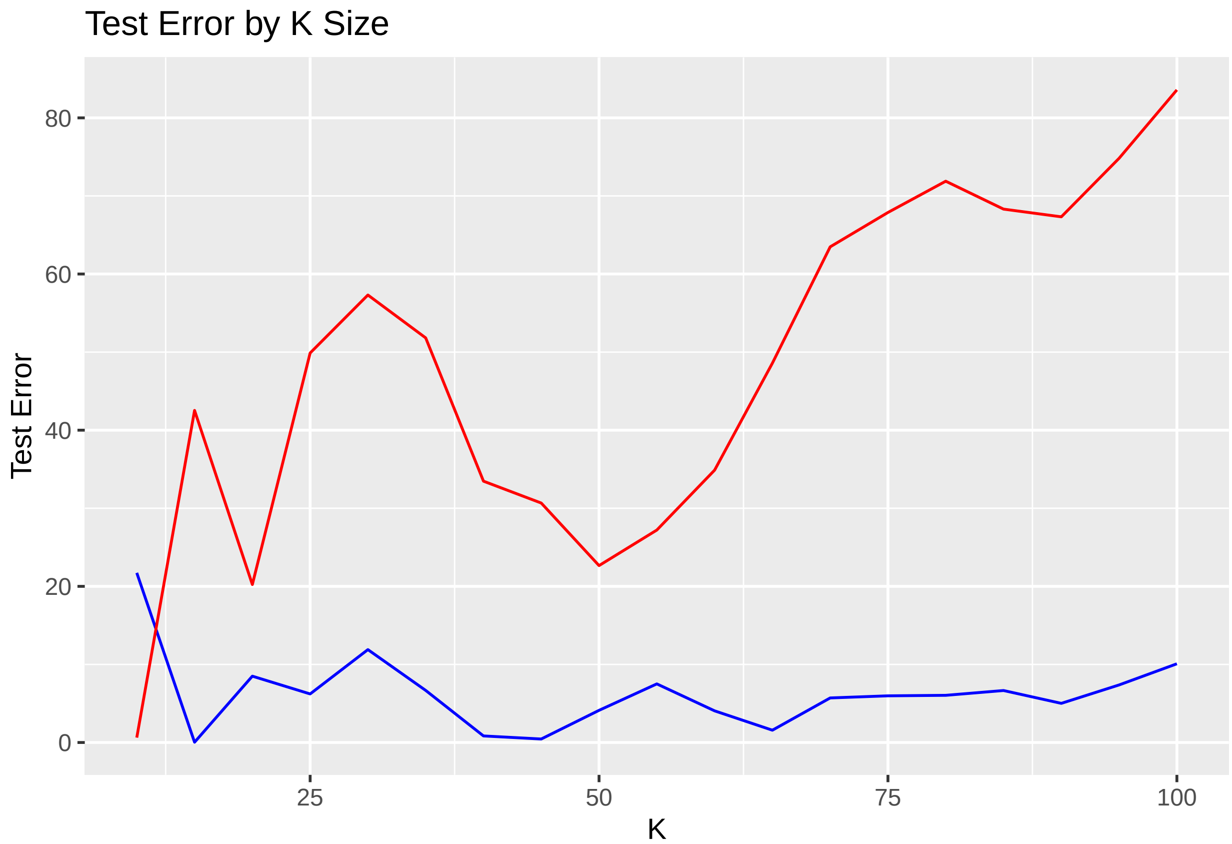
I plan to first run a kNN regression based on exit velocity and launch angle, then correct for other factors after. Due to the relatively small size of this data, I'm going to cross-validate the dataset using 10 folds. This next set of code is creating a separate dataframe for each fold, then stitching the original dataframe back together.

```
test_size <- round(nrow(cbb_inplay) * 0.1 ,0)

for(i in 1:10){
  sample_vec <- sample(1:nrow(cbb_inplay), test_size, replace = FALSE)
  sample_frame <- cbb_inplay[sample_vec,]
  cbb_inplay <- cbb_inplay[-sample_vec,]
  assign(paste("test_fold_",i,sep = ''),sample_frame)
}

cbb_inplay <- rbind(test_fold_1, test_fold_2, test_fold_3, test_fold_4, test_fold_5,
```

```
                     test_fold_6, test_fold_7, test_fold_8, test_fold_9, test_fold_10)

cbb_inplay <- subset(cbb_contact, cbb_contact$PitchCall == 'InPlay')
cbb_inplay$EventID <- c(1:nrow(cbb_inplay))

cbb_inplay$ba <- 0
cbb_inplay$ba[cbb_inplay$PlayResult == 'Single'] <- 1
cbb_inplay$ba[cbb_inplay$PlayResult == 'Double'] <- 1
cbb_inplay$ba[cbb_inplay$PlayResult == 'Triple'] <- 1
cbb_inplay$ba[cbb_inplay$PlayResult == 'HomeRun'] <- 1

cbb_inplay$slg <- 0
cbb_inplay$slg[cbb_inplay$PlayResult == 'Single'] <- 1
cbb_inplay$slg[cbb_inplay$PlayResult == 'Double'] <- 2
cbb_inplay$slg[cbb_inplay$PlayResult == 'Triple'] <- 3
cbb_inplay$slg[cbb_inplay$PlayResult == 'HomeRun'] <- 4
```

Although I know I want to conduct a kNN regression, I don't know what size K is most appropriate. To do so, I am going to run a for loop that runs the analysis at a variety of different values for K. I've nested another for loop inside that will cross-validate the dataset and then keep track of the test errors at each value for K.

```
error_frame <- data.frame("K" = seq(10,100,5),
                          "xBA" = xBA_error,
                          "xSLG" = xSLG_error)

testerror_plot <- ggplot() +
  geom_line(data = error_frame, aes(x = K, y = xBA), color = "blue") +
  geom_line(data = error_frame, aes(x = K, y = xSLG), color = "red") +
  labs(x = 'K', y = 'Test Error', title = 'Test Error by K Size')
testerror_plot
```



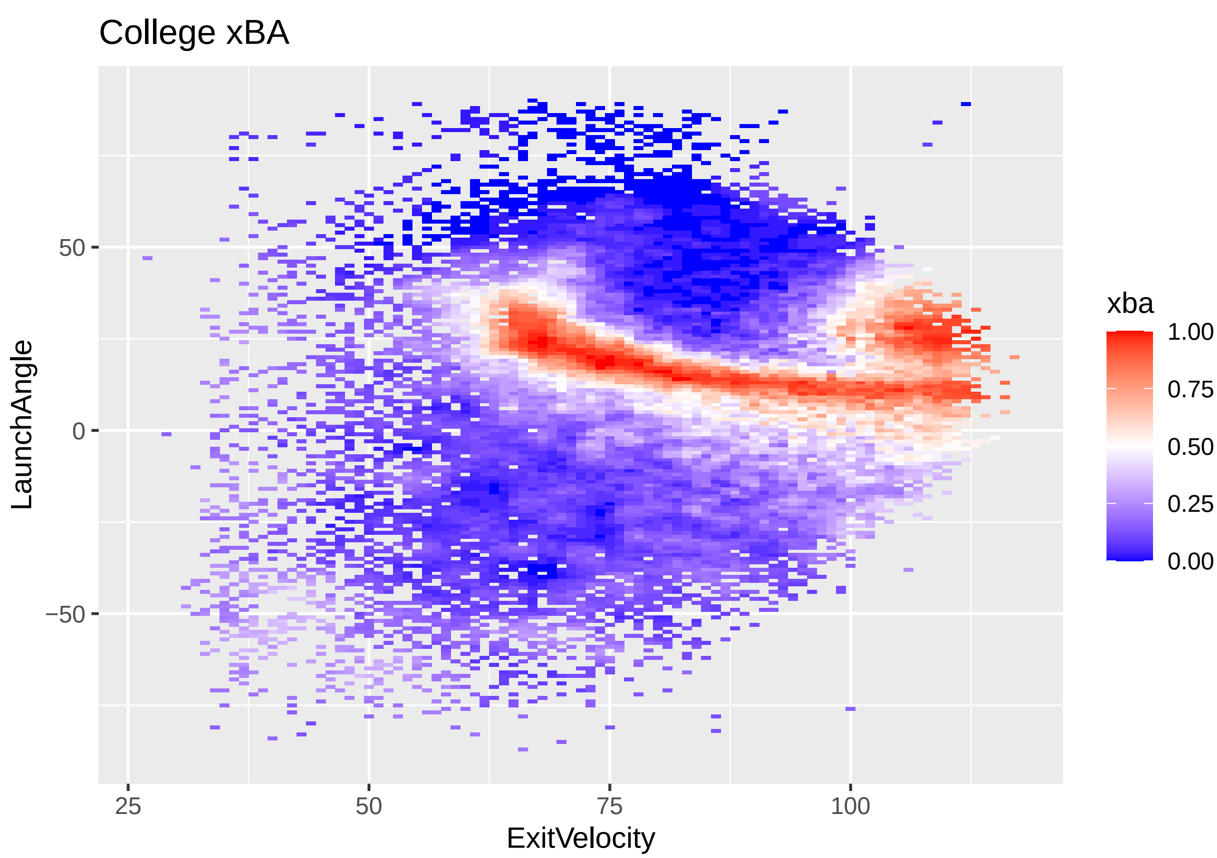
The graph indicates a general trend of larger values for K greatly increasing the test error for xSLG, while xBA only starts to noticeably increase towards the end. Since it will be a different algorithm that predicts xBA and xSLG, we could choose two different values for K. However, since the ultimate goal is to predict ISO, I will need to use the same value of K for both predictions in order to prevent the possibility of a negative ISO. Since xBA and xSLG have different scales, I decided to rank their errors at each value for K and find the lowest average of those at a particular value, instead of just adding the errors together.

```
error_frame <- error_frame[order(error_frame$xBA),]
error_frame$ba_rank <- c(1:19)
error_frame <- error_frame[order(error_frame$xSLG),]
error_frame$slg_rank <- c(1:19)
error_frame$rank_avg <- (error_frame$ba_rank + error_frame$slg_rank) / 2
error_frame <- error_frame[order(error_frame$rank_avg, error_frame$ba_rank),]
best_k <- error_frame$K[1]

xba_knn <- knnreg(xba_frame, inplay_train$ba, best_k)
xslg_knn <- knnreg(xslg_frame, inplay_train$slg, best_k)
```

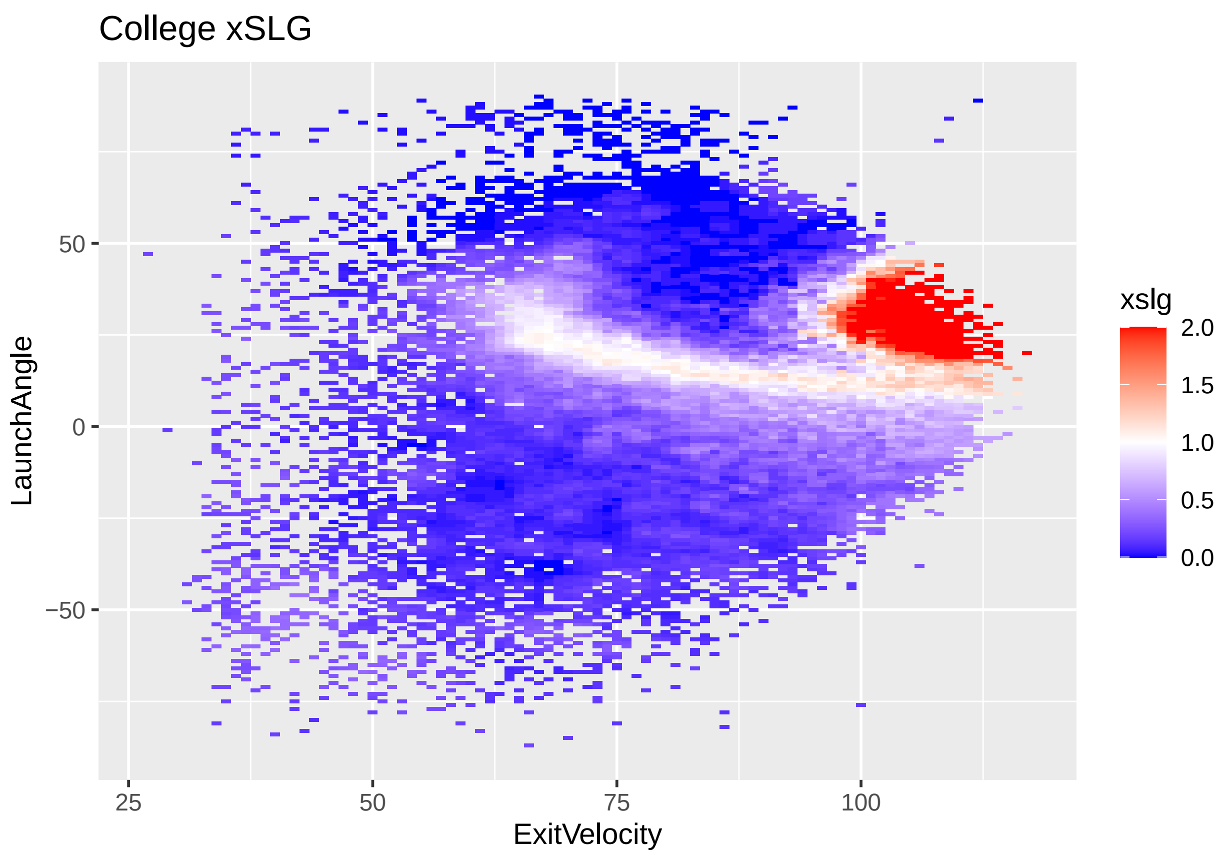
I have now fitted the knn regressions for both xBA and xSLG. In order to check the validity, heatmaps of both statistics are shown below. The shape somewhat resembles similar plots that exist for xwOBA on baseball savant, indicating to me that we are on the right track.

```
ev_vec <- c(round(cbb_inplay$ExitSpeed,0))
la_vec <- c(round(cbb_inplay$Angle,0))
```

```
expected_display <- data.frame('ExitVelocity' = ev_vec,
                               'LaunchAngle' = la_vec)
expected_display <- unique(expected_display)
display_xba <- predict(xba_knn, expected_display)
display_xslg <- predict(xslg_knn, expected_display)
expected_display$xba <- c(display_xba)
expected_display$xslg <- c(display_xslg)

xba_map <- ggplot(data = expected_display, aes(x = ExitVelocity, y = LaunchAngle)) +
  geom_tile(aes(fill = xba)) +
  scale_fill_gradient2(low = "Blue", mid = "White", high = "Red", midpoint = 0.5, limits = c(0, 1), oob
  ggtitle('College xBA')
print(xba_map)
```




```
xslg_map <- ggplot(data = expected_display, aes(x = ExitVelocity, y = LaunchAngle)) +
  geom_tile(aes(fill = xslg)) +
  scale_fill_gradient2(low = "Blue", mid = "White", high = "Red", midpoint = 1, limits = c(0, 2), oob =
  ggtitle('College xSLG')
print(xslg_map)
```



Before making further adjustments, I'm going to go ahead and see who the top performers are in the model's current iteration. Remember all data is from 2018 and 2019 and is limited to games that took place at stadiums within our data sharing agreement. The presence of names such as Torkelson, Vaughn, and Bleday near the top indicates that we are on the right track

```
player_frame <- data.frame("ExitSpeed" = cbb_inplay$ExitSpeed,
                           "Angle" = cbb_inplay$Angle)

player_xBA <- predict(xba_knn, player_frame)
player_xSLG <- predict(xslg_knn, player_frame)
player_frame$xBA <- player_xBA
player_frame$xSLG <- player_xSLG
player_frame$BA <- cbb_inplay$ba
player_frame$SLG <- cbb_inplay$slg
player_frame$xISO <- player_frame$xSLG - player_frame$xBA
player_frame$PA <- 1
player_frame$Player <- cbb_inplay$Batter
cbb_inplay$Bearing <- as.character(cbb_inplay$Bearing)
player_frame$Direction <- round(as.numeric(cbb_inplay$Bearing),0)
```

```
## Warning: NAs introduced by coercion
```

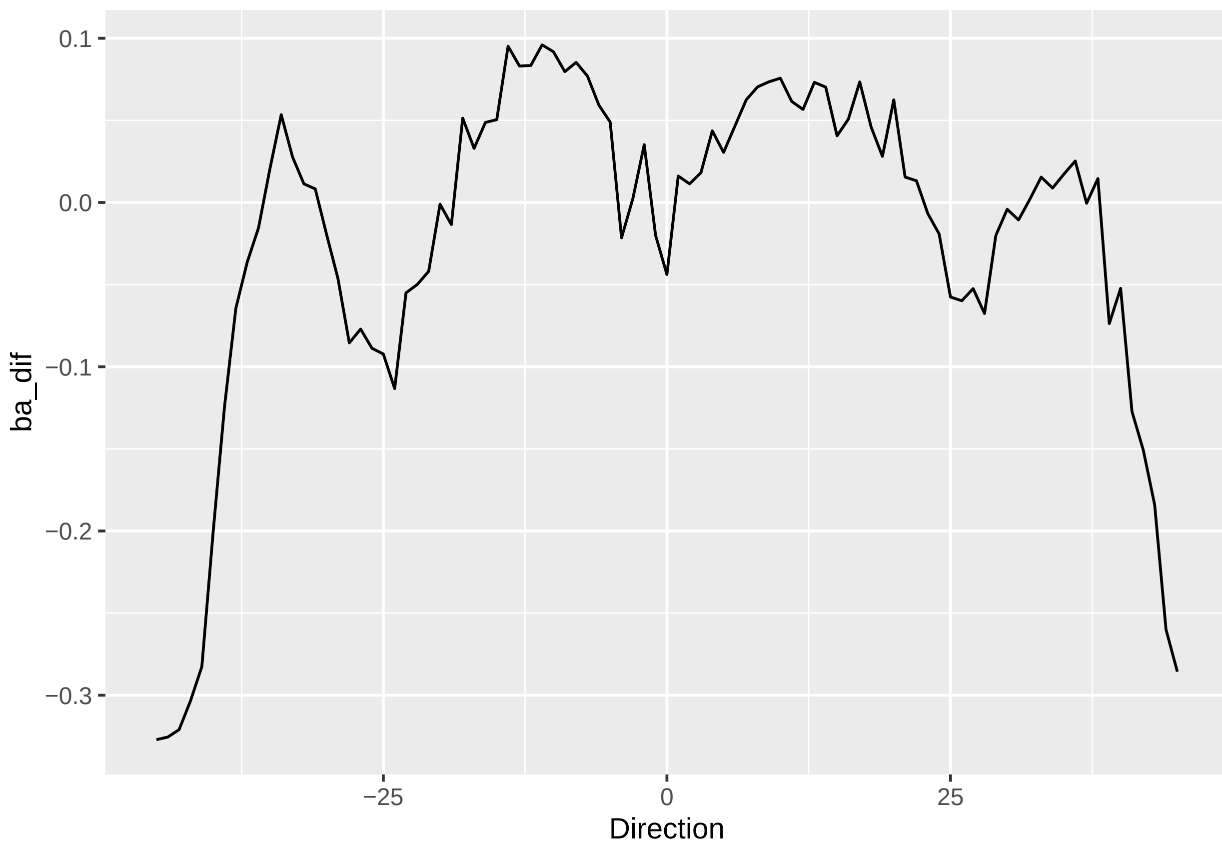
```
player_averages <- aggregate(cbind(xBA,xSLG,PA)~Player, player_frame, sum)
player_averages$xBA <- player_averages$xBA / player_averages$PA
player_averages$xSLG <- player_averages$xSLG / player_averages$PA
```

```r
player_averages$xISO <- player_averages$xSLG - player_averages$xBA

player_averages <- subset(player_averages, player_averages$PA >= 30)
player_averages <- player_averages[order(-player_averages$xISO),]

print(head(player_averages, 20))
```

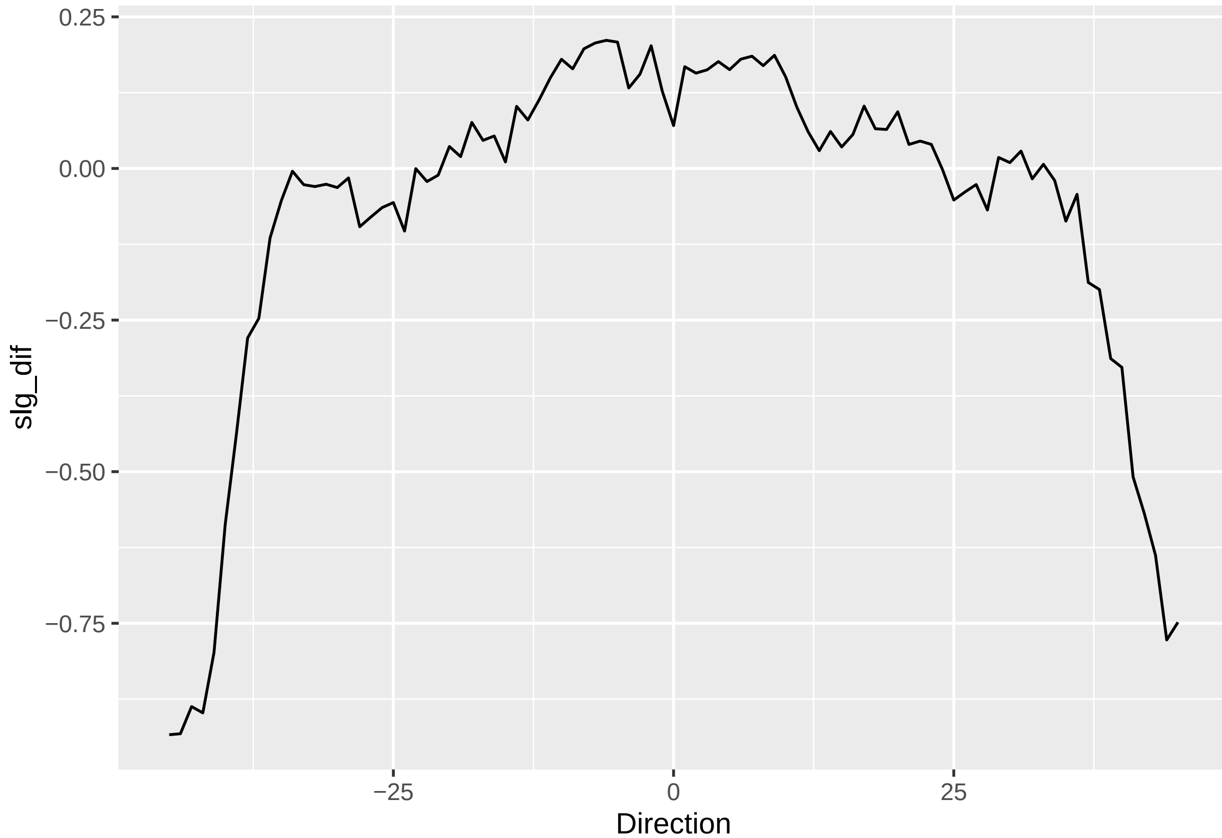
```
##                   Player       xBA      xSLG  PA      xISO
## 433      Cabell, Elijah 0.5217909 1.2035369  56 0.6817460
## 2038       McCann, Kyle 0.4908213 1.1175523  69 0.6267311
## 3229     Vaughn, Andrew 0.5036036 1.0963964  74 0.5927928
## 269          Bleday, JJ 0.4810582 1.0694180 105 0.5883598
## 158          Barr, Cole 0.4978897 1.0713366  57 0.5734469
## 3157 Torkelson, Spencer 0.4912186 1.0602151 124 0.5689964
## 2556    Radcliff, Baron 0.5186380 1.0652330  62 0.5465950
## 254      Bishop, Hunter 0.4864924 1.0261438 102 0.5396514
## 2121      Mendoza, Drew 0.5287689 1.0653729  87 0.5366039
## 1846     Lloyd, Matthew 0.4794118 1.0111111  68 0.5316993
## 3292       Washer, Jake 0.5648746 1.0939068  31 0.5290323
## 528     Chavers, Parker 0.4160643 0.9386881  83 0.5226238
## 2055     McDaniel, Joel 0.4397661 0.9469786  57 0.5072125
## 231     Berryhill, Luke 0.4063492 0.9064713  91 0.5001221
## 2921      Smith, Armani 0.4575866 0.9560335  93 0.4984468
## 2719      Sabato, Aaron 0.4788594 0.9762045 113 0.4973451
## 1471     Horanski, Luke 0.4312618 0.9276836  59 0.4964218
## 1197       Gorski, Matt 0.4961857 0.9810945  67 0.4849088
## 316     Bradley, Scotty 0.4404938 0.9204938  45 0.4800000
## 295       Boone, Trevor 0.4620072 0.9362007  62 0.4741935
```

I will now graph the difference between expected BA and SLG and actual BA and SLG. This is done to see if there are any biases in the model that may be caused by external forces

```r
direction_averages <- aggregate(cbind(xBA,xSLG,PA,BA,SLG)~Direction, player_frame, sum)
direction_averages$xBA <- direction_averages$xBA / direction_averages$PA
direction_averages$xSLG <- direction_averages$xSLG / direction_averages$PA
direction_averages$xISO <- direction_averages$xSLG - direction_averages$xBA
direction_averages$BA <- direction_averages$BA / direction_averages$PA
direction_averages$SLG <- direction_averages$SLG / direction_averages$PA
direction_averages$ba_dif <- direction_averages$xBA - direction_averages$BA
direction_averages$slg_dif <- direction_averages$xSLG - direction_averages$SLG

direction_averages <- subset(direction_averages, direction_averages$Direction >= -45 &
                               direction_averages$Direction <= 45)

direction_line <- ggplot(data = direction_averages, aes(x = Direction, y = ba_dif)) +
  geom_line()
direction_line
```




```
direction_line2 <- ggplot(data = direction_averages, aes(x = Direction, y = slg_dif)) +
  geom_line()
direction_line2
```



As the graphs above indicate, there is a clear bias in the model based on the direction of the batted ball. Given that the positions where xBA outperforms actual BA are at the approximate normal positions for infielders, it would appear defensive positioning is at least partially the cause of this bias.

To correct this, I am going to fit a generalized additive model for both xBA and xSLG. I initially chose the number of splines based on the shape of each graph. Once I ran the GAM's the first time, the new graphs still appeared to show some form of bias, so I added a second term to correct for these. I then applied the adjustments to the xBA and xSLG metrics that already existed from the kNN regression. However, I opted to add no adjustment on batted balls that were outside of the foul lines since these were prone to unrealistic adjustments and the only batted balls included in this data set were marked "In-Play", meaning balls out of the foul lines were limited to easy pop-ups.

```
ba_adj <- gam(ba_dif~ns(Direction,7) + ns(Direction,9), data = direction_averages)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```
summary(ba_adj)
```

```
##
## Call: gam(formula = ba_dif ~ ns(Direction, 7) + ns(Direction, 9), data = direction_averages)
## Deviance Residuals:
##       Min        1Q    Median        3Q       Max
```

```
## -0.072463 -0.011368  0.001107  0.013192  0.065540
##
## (Dispersion Parameter for gaussian family taken to be 5e-04)
##
##     Null Deviance: 0.95 on 90 degrees of freedom
## Residual Deviance: 0.0359 on 75 degrees of freedom
## AIC: -421.0822
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                   Df  Sum Sq  Mean Sq F value    Pr(>F)
## ns(Direction, 7)   7 0.69768 0.099669 208.410 < 2.2e-16 ***
## ns(Direction, 9)   8 0.21643 0.027054  56.571 < 2.2e-16 ***
## Residuals         75 0.03587 0.000478
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
slg_adj <- gam(slg_dif~ns(Direction,6) + ns(Direction,9), data = direction_averages)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```
summary(slg_adj)
```

```
##
## Call: gam(formula = slg_dif ~ ns(Direction, 6) + ns(Direction, 9),
##     data = direction_averages)
## Deviance Residuals:
##       Min        1Q    Median        3Q       Max
## -0.167022 -0.024954  0.005965  0.023844  0.168962
##
## (Dispersion Parameter for gaussian family taken to be 0.0026)
##
##     Null Deviance: 7.4687 on 90 degrees of freedom
## Residual Deviance: 0.2001 on 78 degrees of freedom
## AIC: -270.6762
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                   Df Sum Sq Mean Sq F value    Pr(>F)
## ns(Direction, 6)   6 6.9233 1.15388 449.894 < 2.2e-16 ***
## ns(Direction, 9)   6 0.3454 0.05757  22.446 3.469e-15 ***
## Residuals         78 0.2001 0.00256
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
predict_frame <- data.frame("Direction" = as.numeric(as.character(cbb_inplay$Bearing)))
```

```
## Warning in data.frame(Direction = as.numeric(as.character(cbb_inplay$Bearing))):
## NAs introduced by coercion
```

```r
player_frame$ba2 <- predict(ba_adj, predict_frame)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```r
player_frame$ba2 <- ifelse(player_frame$Direction >= -45 & player_frame$Direction <= 45,
                           player_frame$ba2, 0)

predict_frame <- data.frame("Direction" = as.numeric(as.character(cbb_inplay$Bearing)))
```

```
## Warning in data.frame(Direction = as.numeric(as.character(cbb_inplay$Bearing))):
## NAs introduced by coercion
```

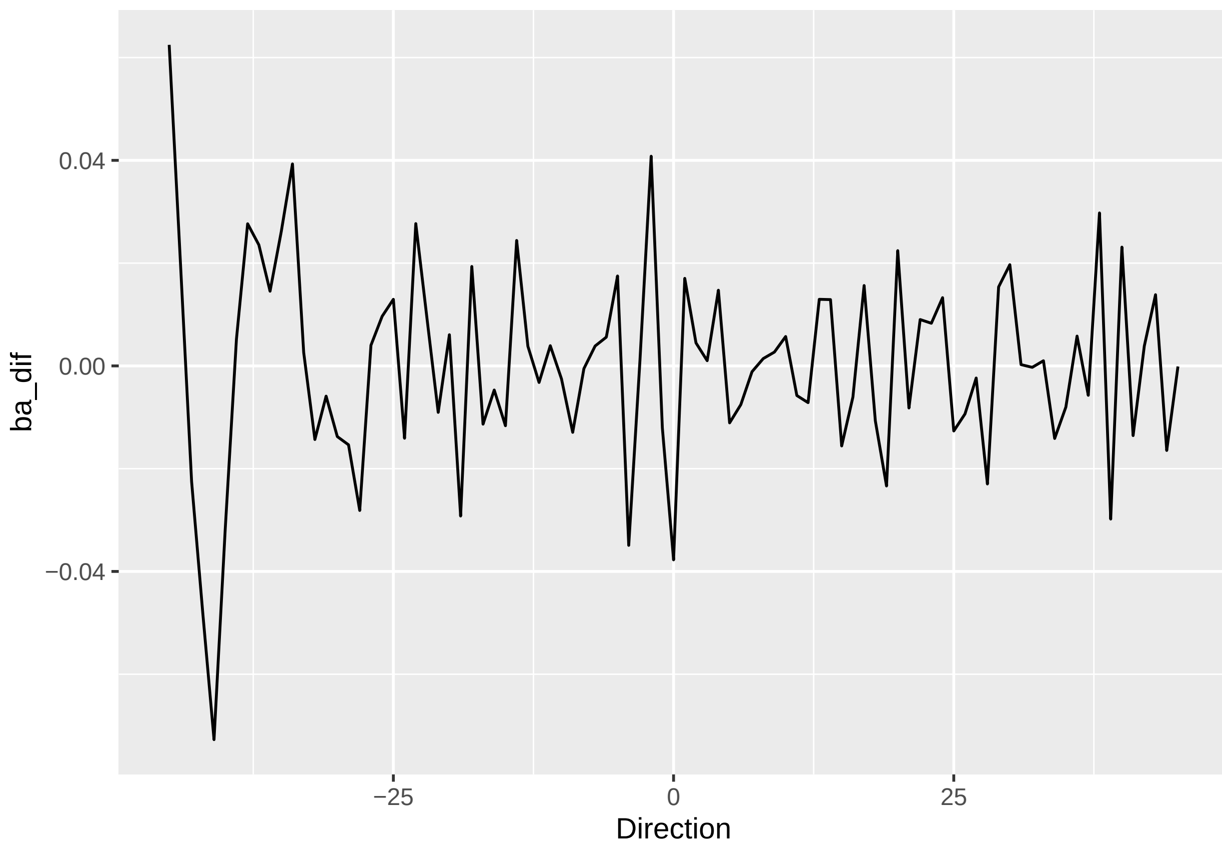
```r
player_frame$slg2 <- predict(slg_adj, predict_frame)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

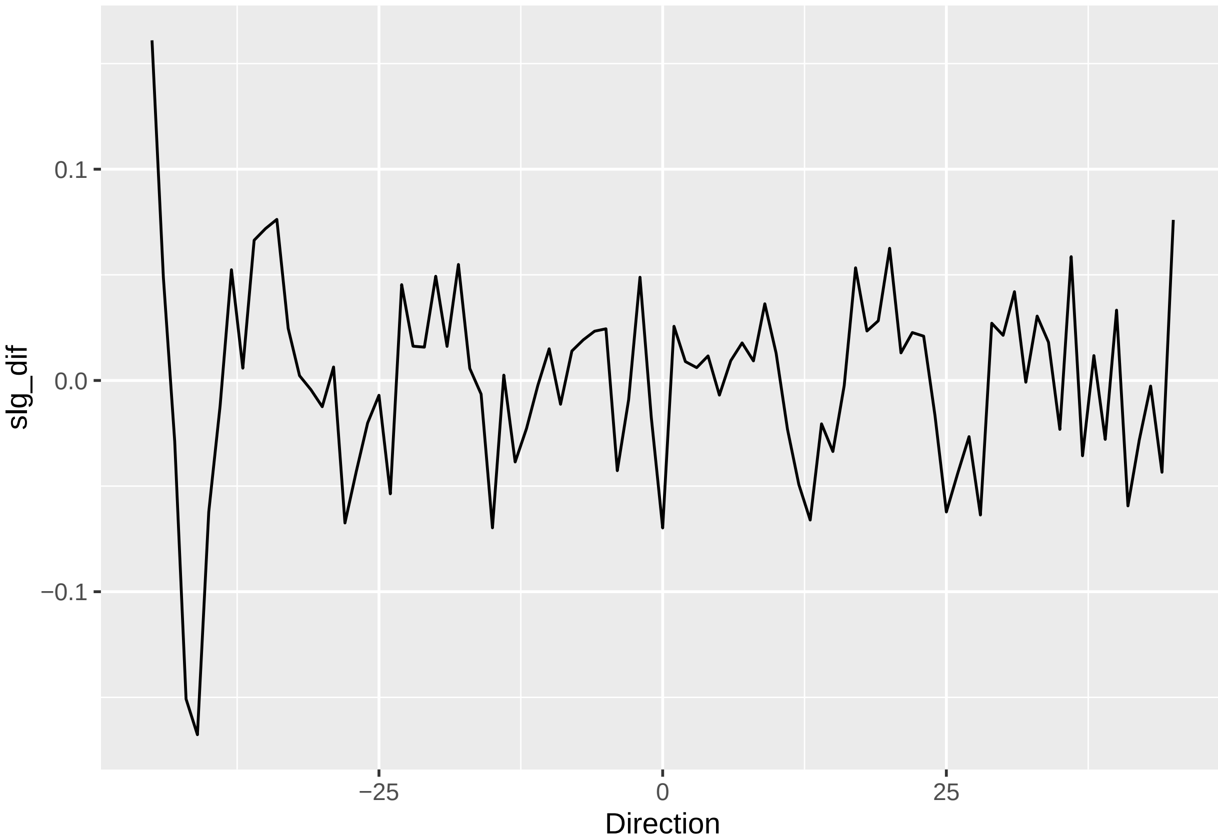
```r
player_frame$slg2 <- ifelse(player_frame$Direction >= -45 & player_frame$Direction <= 45,
                           player_frame$slg2, 0)
```

After adding these values, it's time to graph the difference between expected and actual batting average and slugging percentage to see if the GAM appeared to correct the bias.

```r
player_frame$xBA <- player_frame$xBA - player_frame$ba2
player_frame$xSLG <- player_frame$xSLG - player_frame$slg2

direction_averages <- aggregate(cbind(xBA,xSLG,PA,BA,SLG)~Direction, player_frame, sum)
direction_averages$xBA <- direction_averages$xBA / direction_averages$PA
direction_averages$xSLG <- direction_averages$xSLG / direction_averages$PA
direction_averages$xISO <- direction_averages$xSLG - direction_averages$xBA
direction_averages$BA <- direction_averages$BA / direction_averages$PA
direction_averages$SLG <- direction_averages$SLG / direction_averages$PA
direction_averages$ba_dif <- direction_averages$xBA - direction_averages$BA
direction_averages$slg_dif <- direction_averages$xSLG - direction_averages$SLG

direction_averages <- subset(direction_averages, direction_averages$Direction >= -45 &
                               direction_averages$Direction <= 45)

direction_line <- ggplot(data = direction_averages, aes(x = Direction, y = ba_dif)) +
  geom_line()
direction_line
```




```
direction_line2 <- ggplot(data = direction_averages, aes(x = Direction, y = slg_dif)) +
  geom_line()
direction_line2
```



There appear to still be some issues right down the foul lines, but overall the differences appear to be random. Obviously an ideal model would have no variation, but I feel that the variance that does exist is largely noise due to the relatively small size of this dataset and not due to an external force that requires correction.

```
player_averages <- aggregate(cbind(xBA,xSLG,PA)~Player, player_frame, sum)
player_averages$xBA <- player_averages$xBA / player_averages$PA
player_averages$xSLG <- player_averages$xSLG / player_averages$PA
player_averages$xISO <- player_averages$xSLG - player_averages$xBA

player_averages <- subset(player_averages, player_averages$PA >= 30)
player_averages <- player_averages[order(-player_averages$xISO),]

print(head(player_averages, 20))
```

```
##                   Player       xBA      xSLG  PA      xISO
## 433      Cabell, Elijah 0.5190033 1.1835110  56 0.6645077
## 2037       McCann, Kyle 0.4879431 1.1087055  69 0.6207624
## 3228     Vaughn, Andrew 0.5058775 1.1039205  74 0.5980430
## 269          Bleday, JJ 0.4890859 1.0823803 105 0.5932945
## 158          Barr, Cole 0.5074929 1.1000640  56 0.5925711
## 3156 Torkelson, Spencer 0.5075968 1.0912515 124 0.5836547
## 2555    Radcliff, Baron 0.5224448 1.0840558  62 0.5616110
## 254      Bishop, Hunter 0.4887589 1.0338769 102 0.5451180
## 3291       Washer, Jake 0.5616378 1.0908222  31 0.5291844
## 1470     Horanski, Luke 0.4333701 0.9619095  59 0.5285394
```

```
## 231     Berryhill, Luke 0.4201209 0.9475818  91 0.5274610
## 2120      Mendoza, Drew 0.5149345 1.0408046  87 0.5258700
## 528     Chavers, Parker 0.4049588 0.9270379  83 0.5220790
## 2054     McDaniel, Joel 0.4512774 0.9659491  57 0.5146717
## 1655      Kavadas, Niko 0.4624297 0.9747977  84 0.5123680
## 2920      Smith, Armani 0.4637686 0.9738385  93 0.5100698
## 2718      Sabato, Aaron 0.4814773 0.9907795 113 0.5093022
## 1845     Lloyd, Matthew 0.4592847 0.9511124  68 0.4918277
## 2861        Shinn, Ryan 0.5013147 0.9923014  90 0.4909866
## 295       Boone, Trevor 0.4808353 0.9712040  62 0.4903687
```

There is slight movement from our initial table to now, meaning the correction was made in the model. An initial glance at the top performers according to this metric looks promising. Top picks like Vaughn, Bleday, Torkelson, Bishop, and Sabato all appearing towards the top indicates that this is capturing some level of power production.

The purpose of this model isn't to estimate overall production, but rather to anticipate how much power a particular batter may hit for WHEN they make contact. In the future, this metric can be referenced when compiling scouting reports so that coaches and pitchers can know which opposing hitters are most likely to hit for power if they are able to square up on a pitch. It is not an all-encompassing offensive metric, but helps tell a part of the story of a hitter.