Plate Discipline

Joshua Mould

2/28/2021

Quantifying MLB Hitter Plate Discipline

Import Necessary Packages

Import MLB 2019 and 2020 pitch by pitch data

```
hit_data <- readRDS("MLB Pitches 2019-2020.rds")
pitchers_2019 <- readRDS("Statcast Pitchers Pitch Data 2019.rds")
pitchers_2020 <- readRDS("Statcast Pitchers Pitch Data 2020.rds")
pitchers <- rbind(pitchers_2019, pitchers_2020)
pitchers <- pitchers %>%
    select(game_pk, at_bat_number, pitch_number, player_name) %>%
    rename(pitcher_name = player_name)
hit_data <- hit_data %>%
    left_join(pitchers, by = c("game_pk", "at_bat_number", "pitch_number"))



hit_data <- hit_data %>%
    select(-spin_dir, -spin_rate_deprecated, -break_angle_deprecated,
        -break_length_deprecated, -tfs_deprecated, -tfs_zulu_deprecated,
        -fielder_2, -umpire, -vx0, -vy0, -vz0, -ax, -ay, -az,
        -pitcher_1, -fielder_2_1, -fielder_3, -fielder_4, -fielder_5,
        -fielder_6, -fielder_7, -fielder_8, -fielder_9, -if_fielding_alignment,
        -of_fielding_alignment)
```

Add run expectancy metrics

```
hit_data <- hit_data %>%
    run_expectancy_code(level = "pitch")
hit_data$description <- as.factor(hit_data$description)
```

Ball/Strike Model

Fit a glm model to determine balls and strikes

```
takes <- hit_data %>%
    filter(description == "ball" | description == "called_strike",
        !is.na(plate_x) & !is.na(plate_z)) %>%
    mutate(strike = ifelse(description == "called_strike", 1,
        0), plate_x2 = plate_x^2, plate_z2 = plate_z^2, plate_x3 = plate_x2 *
        plate_x, plate_z3 = plate_z2 * plate_z, plate_xz = plate_x *
        plate_z, plate_x2z = plate_x^2 * plate_z, plate_xz2 = plate_x *
        plate_z^2)
```

```r
strike_prob_model <- glm(strike ~ plate_x + plate_z + plate_x2 +
    plate_z2 + plate_xz + sz_top + sz_bot, data = takes, family = binomial(link = "logit"))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
summary(strike_prob_model)
```

```
##
## Call:
## glm(formula = strike ~ plate_x + plate_z + plate_x2 + plate_z2 +
##     plate_xz + sz_top + sz_bot, family = binomial(link = "logit"),
##     data = takes)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4618  -0.1362  -0.0023   0.1732   5.6198
##
## Coefficients:
##              Estimate Std. Error z value            Pr(>|z|)
## (Intercept) -24.55375    0.13776 -178.23 <0.0000000000000002 ***
## plate_x      -0.41555    0.03634  -11.43 <0.0000000000000002 ***
## plate_z      23.94609    0.09010  265.77 <0.0000000000000002 ***
## plate_x2     -5.29638    0.01966 -269.36 <0.0000000000000002 ***
## plate_z2     -4.90020    0.01849 -265.00 <0.0000000000000002 ***
## plate_xz      0.16751    0.01494   11.21 <0.0000000000000002 ***
## sz_top        1.65734    0.04939   33.56 <0.0000000000000002 ***
## sz_bot       -2.95028    0.08313  -35.49 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 630586  on 497770  degrees of freedom
## Residual deviance: 186162  on 497763  degrees of freedom
## AIC: 186178
##
## Number of Fisher Scoring iterations: 9
```

```r
# takes %>% add_predictions(strike_prob_model, type =
# 'response')

GetAUC(strike_prob_model, takes, takes$strike)
```

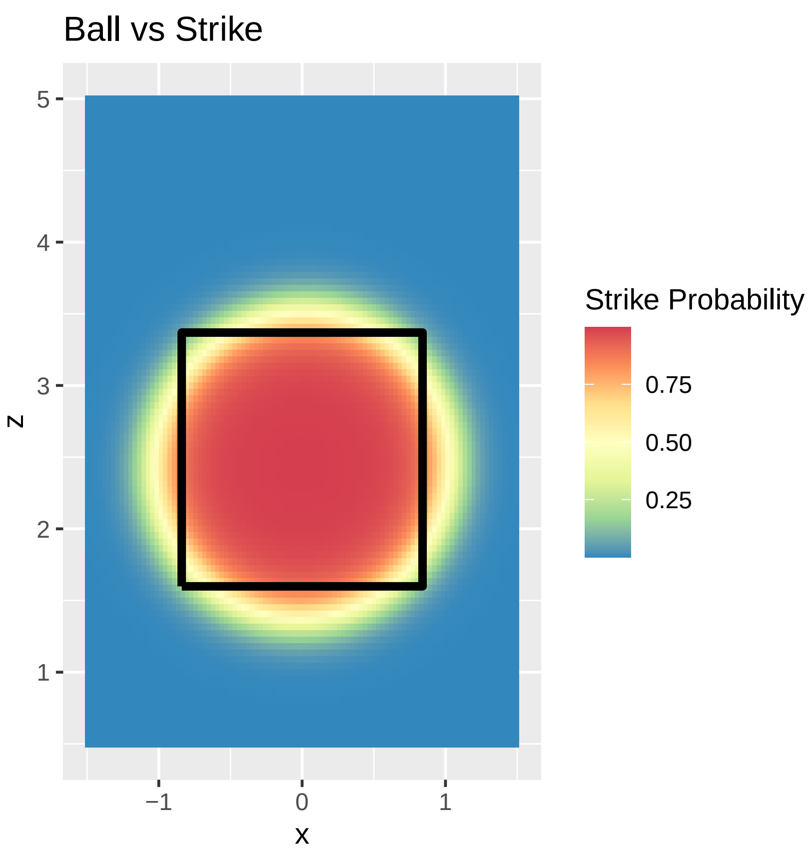
```
## [1] 0.9759137
```

```r
# takes %>% select(strike, plate_x, plate_z, plate_x2,
# plate_z2, plate_x3, plate_z3, plate_xz, plate_x2z,
# plate_xz2) %>% cor()
```

Create a strike heatmap to see the accuracy of the logistic model

```r
x <- seq(-1.5, 1.5, length.out = 100)
y <- seq(0.5, 5, length.out = 100)
preds <- data.frame(plate_x = c(outer(x, y * 0 + 1)), plate_z = c(outer(x *
    0 + 1, y)), sz_top = mean(hit_data$sz_top, na.rm = T), sz_bot = mean(hit_data$sz_bot,
    na.rm = T))
```

```r
preds <- preds %>%
    mutate(plate_x2 = plate_x^2, plate_z2 = plate_z^2, plate_x3 = plate_x2 *
        plate_x, plate_z3 = plate_z2 * plate_z, plate_xz = plate_x *
        plate_z, plate_x2z = plate_x^2 * plate_z, plate_xz2 = plate_x *
        plate_z^2) %>%
    add_predictions(strike_prob_model) %>%
    mutate(strike_prob = 1/(1 + exp(-1 * pred)))

# strike_predictions <- predict(strike_prob_model,
# type='response', newdata = preds) preds <- preds %>%
# add_column(strike_predictions)

topKzone <- mean(hit_data$sz_top, na.rm = T)
botKzone <- mean(hit_data$sz_bot, na.rm = T)
inKzone <- -0.84
outKzone <- 0.84
kZone <- data.frame(x = c(inKzone, inKzone, outKzone, outKzone,
    inKzone), y = c(botKzone, topKzone, topKzone, botKzone, botKzone))

# construct the plot

ggplot(kZone, aes(x, y)) + geom_tile(data = preds, aes(x = plate_x,
    y = plate_z, fill = strike_prob)) + scale_fill_distiller(palette = "Spectral") +
    geom_path(lwd = 1.5, col = "black") + coord_fixed() + ggtitle("Ball vs Strike") +
    labs(fill = "Strike Probability", y = "z")
```



Logic for run expectancy ball and strike data frames, create run expectancy states for next possible outcomes (ball/strike) based on current situations.

```
takes$pitch_id <- seq(1, nrow(takes))

ball_next <- takes
ball_next <- ball_next %>%
    mutate(balls_new = ifelse(balls < 3, balls + 1, 0), strikes_new = ifelse(balls <
        3, strikes, 0), outs_when_up_new = outs_when_up, on_1b_new = ifelse(balls ==
        3 & is.na(on_1b), 1, on_1b), on_2b_new = ifelse(balls ==
        3 & !is.na(on_1b), 1, on_2b), on_3b_new = ifelse(balls ==
        3 & !is.na(on_1b) & !is.na(on_2b), 1, on_3b)) %>%
    mutate(count_base_out_state_new = paste(balls_new, "-", strikes_new,
        ", ", outs_when_up_new, " outs, ", ifelse(!is.na(.$on_1b_new),
            "1b", "_"), ifelse(!is.na(.$on_2b_new), "2b", "_"),
        ifelse(!is.na(.$on_3b_new), "3b", "_")))
# merge run exxpectancy table to new data
ball_next <- ball_next %>%
    left_join(run_expectancy_state_table, by = c(count_base_out_state_new = "count_base_out_state"))

strike_next <- takes
strike_next <- strike_next %>%
    mutate(balls_new = ifelse(strikes < 2, balls, 0), strikes_new = ifelse(strikes <
        2, strikes + 1, 0), outs_when_up_new = ifelse(strikes <
        2, outs_when_up, outs_when_up + 1), on_1b_new = on_1b,
        on_2b_new = on_2b, on_3b_new = on_3b) %>%
    mutate(count_base_out_state_new = paste(balls_new, "-", strikes_new,
        ", ", outs_when_up_new, " outs, ", ifelse(!is.na(.$on_1b_new),
            "1b", "_"), ifelse(!is.na(.$on_2b_new), "2b", "_"),
        ifelse(!is.na(.$on_3b_new), "3b", "_")))
# merge run exxpectancy table to new data
strike_next <- strike_next %>%
    left_join(run_expectancy_state_table, by = c(count_base_out_state_new = "count_base_out_state"))

strike_next[which(strike_next$outs_when_up_new == 3), "avg_re.y"] <- 0
```

Combine the resulting data frames and calculate strike probability

```
takes_ball <- ball_next %>%
    select(pitch_id, balls_new, strikes_new, outs_when_up_new,
        on_1b_new, on_2b_new, on_3b_new, count_base_out_state_new,
        avg_re.y) %>%
    rename(count_base_out_state_ball = count_base_out_state_new,
        avg_re.ball = avg_re.y)

take_outcomes <- strike_next %>%
    rename(count_base_out_state_orig = count_base_out_state,
        count_base_out_state_strike = count_base_out_state_new,
        avg_re.orig = avg_re.x, avg_re.strike = avg_re.y) %>%
    left_join(takes_ball, by = "pitch_id")

take_outcomes <- take_outcomes %>%
    add_predictions(strike_prob_model, var = "strike_prob", type = "response")
```

Create variable with the run expectancy total for taking a pitch

```
take_outcomes <- take_outcomes %>%
    mutate(re_take = avg_re.ball * (1 - strike_prob) - avg_re.strike *
        strike_prob)


take_outcomes %>%
    select(plate_x, plate_z, balls, strikes, on_1b, on_2b, on_3b,
        strike_prob, avg_re.strike, avg_re.ball, re_take) %>%
    arrange(re_take) %>%
    head()
```

```
## # A tibble: 6 x 11
##   plate_x plate_z balls strikes  on_1b  on_2b  on_3b strike_prob avg_re.strike
##     <dbl>   <dbl> <dbl>   <dbl>  <dbl>  <dbl>  <dbl>       <dbl>         <dbl>
## 1   -0.21    2.48     3       0 641355 572041 457759       0.995          2.81
## 2    0.23    2.31     3       0 592206 553993 458015       0.995          2.81
## 3   -0.21    2.64     3       0 624577 467793 640458       0.995          2.81
## 4    0       2.15     3       0 571448 502517 596115       0.995          2.81
## 5    0.16    2.2      3       0 621311 571448 502517       0.991          2.81
## 6    0.03    2.07     3       0 641355 571970 608369       0.990          2.81
## # ... with 2 more variables: avg_re.ball <dbl>, re_take <dbl>
```

Build xgboost Multi-Classification Probability Model

Use statcast data for creating metrics for players with the swings dataset to be used in the classification model.

```
## Data courtesy of Baseball-Reference.com. Please consider supporting Baseball-Reference by signing up
```

```
## `summarise()` has grouped output by 'player_name'. You can override using the `.groups` argument.
```

```
## Data courtesy of Baseball-Reference.com. Please consider supporting Baseball-Reference by signing up
```

Now prep for classification model with outcome column addition.

```
hit_data <- hit_data %>%
    mutate(on_1b = ifelse(!is.na(on_1b), 1, 0), on_2b = ifelse(!is.na(on_2b),
        1, 0), on_3b = ifelse(!is.na(on_3b), 1, 0))


hit_data <- hit_data %>%
    mutate(outcome = ifelse(description == "swinging_strike" |
        description == "swinging_strike_blocked" | description ==
        "foul_tip" | (strikes == 2 & description == "foul_bunt") |
        description == "bunt_foul_tip" | description == "missed_bunt",
        "Miss", ifelse(description == "foul" | description ==
            "foul_bunt", "Foul", ifelse(events == "single", "Single",
            ifelse(events == "double", "Double", ifelse(events ==
                "triple", "Triple", ifelse(events == "home_run",
                "Homerun", "Out")))))))
```

Build classification model using xgboost to determine outcome probabilities of Miss, Foul, Out, Single, Double, Triple, and Homerun.

```
swings <- hit_data %>%
    filter(description != "ball" & description != "called_strike" &
        description != "blocked_ball" & description != "pitchout" &
```

```
        description != "hit_by_pitch")

swings <- swings %>%
    filter(ERA <= 20)

set.seed(7777777)
sample <- sample(1:length(swings$game_date), length(swings$game_date) *
    0.8)
train <- swings[sample, ]
test <- swings[-sample, ]

train_lab <- as.factor(train$outcome)
test_lab <- as.factor(test$outcome)

final_lab <- levels(train_lab)

train <- train %>%
    select(plate_x, plate_z, pfx_x, pfx_z, pitch_type, balls,
        strikes, outs_when_up, release_spin_rate, stand, p_throws,
        effective_speed, release_speed, PA, BA, OBP, SLG, barrel_perc,
        avg_exit_velocity, avg_launch_angle, xAVG, xwOBA, wOBA,
        ISO, IP, ERA, GB.FB, WHIP, SO_perc, uBB_perc, overall_spin_rate,
        overall_velocity, outcome)



test <- test %>%
    select(plate_x, plate_z, pfx_x, pfx_z, pitch_type, balls,
        strikes, outs_when_up, release_spin_rate, stand, p_throws,
        effective_speed, release_speed, PA, BA, OBP, SLG, barrel_perc,
        avg_exit_velocity, avg_launch_angle, xAVG, xwOBA, wOBA,
        ISO, IP, ERA, GB.FB, WHIP, SO_perc, uBB_perc, overall_spin_rate,
        overall_velocity, outcome)


train$pitch_type <- as.numeric(as.factor(train$pitch_type)) -
    1
train$p_throws <- as.numeric(as.factor(train$p_throws)) - 1
train$stand <- as.numeric(as.factor(train$stand)) - 1


test$pitch_type <- as.numeric(as.factor(test$pitch_type)) - 1
test$p_throws <- as.numeric(as.factor(test$p_throws)) - 1
test$stand <- as.numeric(as.factor(test$stand)) - 1



train_lab <- as.numeric(as.factor(train_lab)) - 1
test_lab <- as.numeric(as.factor(test_lab)) - 1


xgb.train = xgb.DMatrix(data = as.matrix(train[, -33]), label = train_lab)
xgb.test = xgb.DMatrix(data = as.matrix(test[, -33]), label = test_lab)
```

```r
##### Model
params <- list(objective = "multi:softprob", num_class = 7, eta = 0.3,
    min_child_weight = 6.47, max_depth = 9, subsample = 0.837,
    colsample_bytree = 0.75, eval_metric = "merror")


swing_mod <- xgb.train(params = params, data = xgb.train, nrounds = 49)
swing_mod
```

```
## ##### xgb.Booster
## raw: 7.5 Mb
## call:
##   xgb.train(params = params, data = xgb.train, nrounds = 49)
## params (as set within xgb.train):
##   objective = "multi:softprob", num_class = "7", eta = "0.3", min_child_weight = "6.47", max_depth =
## xgb.attributes:
##   niter
## callbacks:
##   cb.print.evaluation(period = print_every_n)
## # of features: 32
## niter: 49
## nfeatures : 32
```

Remove unnecessary variables

```r
remove(ball_next, basic_batter_stats, basic_pitcher_stats, grouped_pitching_stats,
    grouped_stats, pitchers, pitchers_2019, pitchers_2020, run_expectancy_state_table,
    strike_next, takes_ball)

remove(train, test, takes, take_outcomes, swings)
```

Don't run this chunk, this is for the optimization of the parameters on the xgboost model which has already been done

```r
# Use built in cv to determine best value for nrounds
xgbcv <- xgb.cv(params = params, data = xgb.train, nrounds = 100,
    nfold = 5, showsd = T, stratified = T, print.every.n = 10,
    early.stop.round = 20, maximize = F)
## best iteration = 49

min(xgbcv$evaluation_log$train_merror_mean)
# Min error is 0.3959454

xgb.importance(feature_names = colnames(train), model = swing_mod)


# convert characters to factors
fact_col <- colnames(train)[sapply(train, is.character)]


train <- train[which(!is.na(train$barrel_perc)), ]
train <- train[which(!is.na(train$xAVG)), ]
train <- train[which(!is.na(train$ERA) & is.finite(train$ERA)),
    ]
```

```
train <- train[which(!is.na(train$overall_spin_rate)), ]

test <- test[which(!is.na(test$barrel_perc)), ]
test <- test[which(!is.na(test$xAVG)), ]
test <- test[which(!is.na(test$ERA) & is.finite(test$ERA)), ]
test <- test[which(!is.na(test$overall_spin_rate)), ]

# create tasks
traintask <- makeClassifTask(data = train, target = "outcome")
testtask <- makeClassifTask(data = test, target = "outcome")

# do one hot encoding`<br/>
traintask <- createDummyFeatures(obj = traintask)
testtask <- createDummyFeatures(obj = testtask)

# create learner
lrn <- makeLearner("classif.xgboost", predict.type = "response")
lrn$par.vals <- list(objective = "multi:softprob", eval_metric = "merror",
    nrounds = 100L, eta = 0.1)

# set parameter space
params <- makeParamSet(makeDiscreteParam("booster", values = c("gbtree",
    "gblinear")), makeIntegerParam("max_depth", lower = 3L, upper = 10L),
    makeNumericParam("min_child_weight", lower = 1L, upper = 10L),
    makeNumericParam("subsample", lower = 0.5, upper = 1), makeNumericParam("colsample_bytree",
        lower = 0.5, upper = 1))

# set resampling strategy
rdesc <- makeResampleDesc("CV", stratify = T, iters = 5L)

# search strategy
ctrl <- makeTuneControlRandom(maxit = 10L)

# set parallel backend
library(parallel)
library(parallelMap)
parallelStartSocket(cpus = detectCores())

# parameter tuning
mytune <- tuneParams(learner = lrn, task = traintask, resampling = rdesc,
    measures = acc, par.set = params, control = ctrl, show.info = T)
mytune
mytune$y
# 0.4584903

# set hyperparameters
lrn_tune <- setHyperPars(lrn, par.vals = mytune$x)

# train model
xgmodel <- train(learner = lrn_tune, task = traintask)


# predict model
```

```
xgpred <- predict(xgmodel, testtask, type = "response")

confusionMatrix(xgpred$data$response, xgpred$data$truth)
# Accuracy : 0.4609

########
xgb.pred = predict(swing_mod, xgb.test, reshape = T)
xgb.pred = as.data.frame(xgb.pred)
xgb.pred$label <- test$outcome
names(xgb.pred) <- levels(test$outcome)


# Use the predicted label with the highest probability
# xgb.pred$prediction = apply(xgb.pred,1,function(x)
# colnames(xgb.pred)[which.max(x)])
xgb.pred$label = swings$outcome[-sample]
colnames(xgb.pred)[1:7] <- final_lab

table(xgb.pred$label)



xgb.pred %>%
    group_by(label) %>%
    summarize(meanMiss = mean(Miss), meanOut = mean(Out), meanSingle = mean(Single),
        meanDouble = mean(Double), meanTriple = mean(Triple),
        meanHR = mean(Homerun), meanFoul = mean(Foul))
quantile(xgb.pred$Homerun, 0.99)
xgb.pred %>%
    filter(xgb.pred$Homerun > 0.075) %>%
    select(label) %>%
    table() %>%
    prop.table()
xgb.pred %>%
    select(label) %>%
    table() %>%
    prop.table()

# Check out results

xgb.pred %>%
    arrange(-Double)
```

Test EAGLE on all pitches from years 2016-2019

Import data for each year and merge pitching data with hitting data.

```
pitches_2016 <- read_rds("MLB Pitches 2016.rds")
pitches_2017 <- read_rds("MLB Pitches 2017.rds")
pitches_2018 <- read_rds("MLB Pitches 2018.rds")
pitches_2019 <- read_rds("MLB Pitches 2019.rds")

pitchers_2016 <- readRDS("Statcast Pitchers Pitch Data 2016.rds")
pitchers_2017 <- readRDS("Statcast Pitchers Pitch Data 2017.rds")
```

```
pitchers_2018 <- readRDS("Statcast Pitchers Pitch Data 2018.rds")
pitchers_2019 <- readRDS("Statcast Pitchers Pitch Data 2019.rds")

pitchers_2016 <- pitchers_2016 %>%
    select(game_pk, at_bat_number, pitch_number, player_name) %>%
    rename(pitcher_name = player_name)
pitches_2016 <- pitches_2016 %>%
    left_join(pitchers_2016, by = c("game_pk", "at_bat_number",
        "pitch_number"))

pitchers_2017 <- pitchers_2017 %>%
    select(game_pk, at_bat_number, pitch_number, player_name) %>%
    rename(pitcher_name = player_name)
pitches_2017 <- pitches_2017 %>%
    left_join(pitchers_2017, by = c("game_pk", "at_bat_number",
        "pitch_number"))

pitchers_2018 <- pitchers_2018 %>%
    select(game_pk, at_bat_number, pitch_number, player_name) %>%
    rename(pitcher_name = player_name)
pitches_2018 <- pitches_2018 %>%
    left_join(pitchers_2018, by = c("game_pk", "at_bat_number",
        "pitch_number"))

pitchers_2019 <- pitchers_2019 %>%
    select(game_pk, at_bat_number, pitch_number, player_name) %>%
    rename(pitcher_name = player_name)
pitches_2019 <- pitches_2019 %>%
    left_join(pitchers_2019, by = c("game_pk", "at_bat_number",
        "pitch_number"))


remove(pitchers_2016, pitchers_2017, pitchers_2018, pitchers_2019)
```

Make function to apply predictions from models and create EAGLE

```
apply_EAGLE <- function(pitch_df, startdate, enddate) {

    hit_data <- pitch_df
    startdate <- as.Date(startdate)
    enddate <- as.Date(enddate)

    hit_data$pitch_id <- seq(1, nrow(hit_data))

    hit_data <- hit_data %>%
        run_expectancy_code(level = "pitch")
    hit_data$description <- as.factor(hit_data$description)

    ball_next <- hit_data
    ball_next <- ball_next %>%
        mutate(balls_new = ifelse(balls < 3, balls + 1, 0), strikes_new = ifelse(balls <
            3, strikes, 0), outs_when_up_new = outs_when_up,
            on_1b_new = ifelse(balls == 3 & is.na(on_1b), 1,
                on_1b), on_2b_new = ifelse(balls == 3 & !is.na(on_1b),
```

```
            1, on_2b), on_3b_new = ifelse(balls == 3 & !is.na(on_1b) &
            !is.na(on_2b), 1, on_3b)) %>%
    mutate(count_base_out_state_new = paste(balls_new, "-",
        strikes_new, ", ", outs_when_up_new, " outs, ", ifelse(!is.na(.$on_1b_new),
            "1b", "_"), ifelse(!is.na(.$on_2b_new), "2b",
            "_"), ifelse(!is.na(.$on_3b_new), "3b", "_")))
# merge run exxpectancy table to new data
ball_next <- ball_next %>%
    left_join(run_expectancy_state_table, by = c(count_base_out_state_new = "count_base_out_state"))

strike_next <- hit_data
strike_next <- strike_next %>%
    mutate(balls_new = ifelse(strikes < 2, balls, 0), strikes_new = ifelse(strikes <
        2, strikes + 1, 0), outs_when_up_new = ifelse(strikes <
        2, outs_when_up, outs_when_up + 1), on_1b_new = on_1b,
        on_2b_new = on_2b, on_3b_new = on_3b) %>%
    mutate(count_base_out_state_new = paste(balls_new, "-",
        strikes_new, ", ", outs_when_up_new, " outs, ", ifelse(!is.na(.$on_1b_new),
            "1b", "_"), ifelse(!is.na(.$on_2b_new), "2b",
            "_"), ifelse(!is.na(.$on_3b_new), "3b", "_")))
# merge run exxpectancy table to new data
strike_next <- strike_next %>%
    left_join(run_expectancy_state_table, by = c(count_base_out_state_new = "count_base_out_state"))

strike_next[which(strike_next$outs_when_up_new == 3), "avg_re.y"] <- 0

#####
hit_data_ball <- ball_next %>%
    mutate(avg_re.ball = avg_re.y - avg_re.x) %>%
    select(pitch_id, balls_new, strikes_new, outs_when_up_new,
        on_1b_new, on_2b_new, on_3b_new, count_base_out_state_new,
        avg_re.ball) %>%
    rename(count_base_out_state_ball = count_base_out_state_new)

hit_data_outcomes <- strike_next %>%
    mutate(avg_re.strike = avg_re.y - avg_re.x) %>%
    rename(count_base_out_state_orig = count_base_out_state,
        count_base_out_state_strike = count_base_out_state_new,
        avg_re.orig = avg_re.x) %>%
    left_join(hit_data_ball, by = "pitch_id")

hit_data_outcomes <- hit_data_outcomes %>%
    mutate(plate_x2 = plate_x^2, plate_z2 = plate_z^2, plate_x3 = plate_x2 *
        plate_x, plate_z3 = plate_z2 * plate_z, plate_xz = plate_x *
        plate_z, plate_x2z = plate_x^2 * plate_z, plate_xz2 = plate_x *
        plate_z^2)

hit_data_outcomes <- hit_data_outcomes %>%
    add_predictions(strike_prob_model, var = "strike_prob",
        type = "response")

hit_data_outcomes <- hit_data_outcomes %>%
    mutate(re_take = avg_re.ball * (1 - strike_prob) + avg_re.strike *
```

```r
        strike_prob)

########
hit_data$pitch_id <- seq(1, nrow(hit_data))

###### Misses
miss_next <- hit_data
miss_next <- miss_next %>%
    mutate(balls_new = ifelse(strikes < 2, balls, 0), strikes_new = ifelse(strikes <
        2, strikes + 1, 0), outs_when_up_new = ifelse(strikes <
        2, outs_when_up, outs_when_up + 1), on_1b_new = on_1b,
        on_2b_new = on_2b, on_3b_new = on_3b) %>%
    mutate(count_base_out_state_new = paste(balls_new, "-",
        strikes_new, ", ", outs_when_up_new, " outs, ", ifelse(!is.na(.$on_1b_new),
            "1b", "_"), ifelse(!is.na(.$on_2b_new), "2b",
            "_"), ifelse(!is.na(.$on_3b_new), "3b", "_")))
# merge run exxpectancy table to new data
miss_next <- miss_next %>%
    left_join(run_expectancy_state_table, by = c(count_base_out_state_new = "count_base_out_state"))

miss_next[which(miss_next$outs_when_up_new == 3), "avg_re.y"] <- 0

###### Fouls

foul_next <- hit_data
foul_next <- foul_next %>%
    mutate(balls_new = balls, strikes_new = ifelse(strikes <
        2, strikes + 1, 2), outs_when_up_new = outs_when_up,
        on_1b_new = on_1b, on_2b_new = on_2b, on_3b_new = on_3b) %>%
    mutate(count_base_out_state_new = paste(balls_new, "-",
        strikes_new, ", ", outs_when_up_new, " outs, ", ifelse(!is.na(.$on_1b_new),
            "1b", "_"), ifelse(!is.na(.$on_2b_new), "2b",
            "_"), ifelse(!is.na(.$on_3b_new), "3b", "_")))
# merge run exxpectancy table to new data
foul_next <- foul_next %>%
    left_join(run_expectancy_state_table, by = c(count_base_out_state_new = "count_base_out_state"))

###### Outs

out_next <- hit_data
out_next <- out_next %>%
    mutate(balls_new = 0, strikes_new = 0, outs_when_up_new = outs_when_up +
        1, on_1b_new = on_1b, on_2b_new = on_2b, on_3b_new = on_3b) %>%
    mutate(count_base_out_state_new = paste(balls_new, "-",
        strikes_new, ", ", outs_when_up_new, " outs, ", ifelse(!is.na(.$on_1b_new),
            "1b", "_"), ifelse(!is.na(.$on_2b_new), "2b",
            "_"), ifelse(!is.na(.$on_3b_new), "3b", "_")))
# merge run exxpectancy table to new data
out_next <- out_next %>%
    left_join(run_expectancy_state_table, by = c(count_base_out_state_new = "count_base_out_state"))

out_next[which(out_next$outs_when_up_new == 3), "avg_re.y"] <- 0
```

```r
###### Single

single_next <- hit_data
single_next <- single_next %>%
    mutate(balls_new = 0, strikes_new = 0, outs_when_up_new = outs_when_up,
        on_1b_new = ifelse(is.na(on_1b), 1, on_1b), on_2b_new = ifelse(is.na(on_1b),
            on_1b, 1), on_3b_new = ifelse(is.na(on_2b), on_2b,
            1)) %>%
    mutate(count_base_out_state_new = paste(balls_new, "-",
        strikes_new, ", ", outs_when_up_new, " outs, ", ifelse(!is.na(.$on_1b_new),
            "1b", "_"), ifelse(!is.na(.$on_2b_new), "2b",
            "_"), ifelse(!is.na(.$on_3b_new), "3b", "_")))
# merge run exxpectancy table to new data
single_next <- single_next %>%
    left_join(run_expectancy_state_table, by = c(count_base_out_state_new = "count_base_out_state")

####### Double

double_next <- hit_data
double_next <- double_next %>%
    mutate(balls_new = 0, strikes_new = 0, outs_when_up_new = outs_when_up,
        on_1b_new = NA, on_2b_new = 1, on_3b_new = ifelse(is.na(on_1b),
            on_1b, 1)) %>%
    mutate(count_base_out_state_new = paste(balls_new, "-",
        strikes_new, ", ", outs_when_up_new, " outs, ", ifelse(!is.na(.$on_1b_new),
            "1b", "_"), ifelse(!is.na(.$on_2b_new), "2b",
            "_"), ifelse(!is.na(.$on_3b_new), "3b", "_")))
# merge run exxpectancy table to new data
double_next <- double_next %>%
    left_join(run_expectancy_state_table, by = c(count_base_out_state_new = "count_base_out_state")

###### Triple
triple_next <- hit_data
triple_next <- triple_next %>%
    mutate(balls_new = 0, strikes_new = 0, outs_when_up_new = outs_when_up,
        on_1b_new = NA, on_2b_new = NA, on_3b_new = 1) %>%
    mutate(count_base_out_state_new = paste(balls_new, "-",
        strikes_new, ", ", outs_when_up_new, " outs, ", ifelse(!is.na(.$on_1b_new),
            "1b", "_"), ifelse(!is.na(.$on_2b_new), "2b",
            "_"), ifelse(!is.na(.$on_3b_new), "3b", "_")))
# merge run exxpectancy table to new data
triple_next <- triple_next %>%
    left_join(run_expectancy_state_table, by = c(count_base_out_state_new = "count_base_out_state")

####### Homerun

homerun_next <- hit_data
homerun_next <- homerun_next %>%
    mutate(balls_new = 0, strikes_new = 0, outs_when_up_new = outs_when_up,
        on_1b_new = NA, on_2b_new = NA, on_3b_new = NA) %>%
    mutate(count_base_out_state_new = paste(balls_new, "-",
        strikes_new, ", ", outs_when_up_new, " outs, ", ifelse(!is.na(.$on_1b_new),
            "1b", "_"), ifelse(!is.na(.$on_2b_new), "2b",
```

```
            "_"), ifelse(!is.na(.$on_3b_new), "3b", "_")))
# merge run exxpectancy table to new data
homerun_next <- homerun_next %>%
    left_join(run_expectancy_state_table, by = c(count_base_out_state_new = "count_base_out_state"))

########

hit_data_miss <- miss_next %>%
    mutate(avg_re.miss = avg_re.y - avg_re.x) %>%
    select(pitch_id, balls_new, strikes_new, outs_when_up_new,
        on_1b_new, on_2b_new, on_3b_new, count_base_out_state_new,
        avg_re.miss) %>%
    rename(count_base_out_state_miss = count_base_out_state_new)

hit_data_foul <- foul_next %>%
    mutate(avg_re.foul = avg_re.y - avg_re.x) %>%
    select(pitch_id, balls_new, strikes_new, outs_when_up_new,
        on_1b_new, on_2b_new, on_3b_new, count_base_out_state_new,
        avg_re.foul) %>%
    rename(count_base_out_state_foul = count_base_out_state_new)

hit_data_out <- out_next %>%
    mutate(avg_re.out = avg_re.y - avg_re.x) %>%
    select(pitch_id, balls_new, strikes_new, outs_when_up_new,
        on_1b_new, on_2b_new, on_3b_new, count_base_out_state_new,
        avg_re.out) %>%
    rename(count_base_out_state_out = count_base_out_state_new)

hit_data_single <- single_next %>%
    mutate(avg_re.single = avg_re.y - avg_re.x) %>%
    select(pitch_id, balls_new, strikes_new, outs_when_up_new,
        on_1b_new, on_2b_new, on_3b_new, count_base_out_state_new,
        avg_re.single) %>%
    rename(count_base_out_state_single = count_base_out_state_new)

hit_data_double <- double_next %>%
    mutate(avg_re.double = avg_re.y - avg_re.x) %>%
    select(pitch_id, balls_new, strikes_new, outs_when_up_new,
        on_1b_new, on_2b_new, on_3b_new, count_base_out_state_new,
        avg_re.double) %>%
    rename(count_base_out_state_double = count_base_out_state_new)

hit_data_triple <- triple_next %>%
    mutate(avg_re.triple = avg_re.y - avg_re.x) %>%
    select(pitch_id, balls_new, strikes_new, outs_when_up_new,
        on_1b_new, on_2b_new, on_3b_new, count_base_out_state_new,
        avg_re.triple) %>%
    rename(count_base_out_state_triple = count_base_out_state_new)

hit_data_homerun <- homerun_next %>%
    mutate(avg_re.homerun = avg_re.y - avg_re.x) %>%
    select(pitch_id, balls_new, strikes_new, outs_when_up_new,
        on_1b_new, on_2b_new, on_3b_new, count_base_out_state_new,
```

```r
        avg_re.homerun) %>%
    rename(count_base_out_state_homerun = count_base_out_state_new)

hit_data_outcomes <- hit_data_outcomes %>%
    left_join(hit_data_miss, by = "pitch_id") %>%
    left_join(hit_data_foul, by = "pitch_id") %>%
    left_join(hit_data_out, by = "pitch_id") %>%
    left_join(hit_data_single, by = "pitch_id") %>%
    left_join(hit_data_double, by = "pitch_id") %>%
    left_join(hit_data_triple, by = "pitch_id") %>%
    left_join(hit_data_homerun, by = "pitch_id")

hit_data_outcomes <- hit_data_outcomes %>%
    mutate(avg_re.ball = ifelse(!is.na(on_1b) & !is.na(on_2b) &
        !is.na(on_3b) & balls == 3, avg_re.ball + 1, avg_re.ball)) %>%
    mutate(avg_re.single = ifelse(is.na(on_3b), avg_re.single,
        avg_re.single + 1)) %>%
    mutate(avg_re.double = ifelse(!is.na(on_3b) & !is.na(on_2b),
        avg_re.double + 2, ifelse(!is.na(on_3b) | !is.na(on_2b),
            avg_re.double + 1, avg_re.double))) %>%
    mutate(avg_re.triple = ifelse(!is.na(on_3b) & !is.na(on_2b) &
        !is.na(on_1b), avg_re.triple + 3, ifelse((!is.na(on_3b) &
        !is.na(on_2b)) | (!is.na(on_1b) & !is.na(on_2b)) |
        (!is.na(on_3b) & !is.na(on_1b)), avg_re.triple +
        2, ifelse(!is.na(on_1b) | !is.na(on_2b) | !is.na(on_3b),
        avg_re.triple + 1, avg_re.triple)))) %>%
    mutate(avg_re.homerun = ifelse(!is.na(on_3b) & !is.na(on_2b) &
        !is.na(on_1b), avg_re.triple + 4, ifelse((!is.na(on_3b) &
        !is.na(on_2b)) | (!is.na(on_1b) & !is.na(on_2b)) |
        (!is.na(on_3b) & !is.na(on_1b)), avg_re.triple +
        3, ifelse(!is.na(on_1b) | !is.na(on_2b) | !is.na(on_3b),
        avg_re.homerun + 2, 1 + avg_re.homerun))))

######## merge to data

basic_batter_stats <- daily_batter_bref(startdate, enddate)

basic_batter_stats$bbref_id <- as.numeric(basic_batter_stats$bbref_id)

hit_data_outcomes <- hit_data_outcomes %>%
    separate(player_name, into = c("Last", "First"), sep = ", ") %>%
    mutate(Name = paste(First, Last))

hit_data_outcomes <- hit_data_outcomes %>%
    inner_join(basic_batter_stats, by = c("Name"))

hit_data_outcomes <- hit_data_outcomes %>%
    mutate(barrel = ifelse(launch_angle <= 50 & launch_speed >=
        98 & launch_speed * 1.5 - launch_angle >= 117 & launch_speed +
        launch_angle >= 124, 1, 0))

# Fix character columns
hit_data_outcomes[which(hit_data_outcomes$launch_speed_angle ==
```

```
    "null"), "launch_speed_angle"] <- NA
hit_data_outcomes[which(hit_data_outcomes$estimated_ba_using_speedangle ==
    "null"), "estimated_ba_using_speedangle"] <- NA
hit_data_outcomes[which(hit_data_outcomes$estimated_woba_using_speedangle ==
    "null"), "estimated_woba_using_speedangle"] <- NA
hit_data_outcomes[which(hit_data_outcomes$woba_value == "null"),
    "woba_value"] <- NA
hit_data_outcomes[which(hit_data_outcomes$iso_value == "null"),
    "iso_value"] <- NA

hit_data_outcomes$launch_speed_angle <- as.numeric(hit_data_outcomes$launch_speed_angle)
hit_data_outcomes$estimated_ba_using_speedangle <- as.numeric(hit_data_outcomes$estimated_ba_using_s
hit_data_outcomes$estimated_woba_using_speedangle <- as.numeric(hit_data_outcomes$estimated_woba_usi
hit_data_outcomes$woba_value <- as.numeric(hit_data_outcomes$woba_value)
hit_data_outcomes$iso_value <- as.numeric(hit_data_outcomes$iso_value)

grouped_stats <- hit_data_outcomes %>%
    group_by(batter, zone) %>%
    summarise(barrel_perc = mean(barrel, na.rm = T), avg_launch_angle = mean(launch_angle,
        na.rm = T), avg_exit_velocity = mean(launch_speed,
        na.rm = T), avg_launch_speed_angle = mean(launch_speed_angle,
        na.rm = T), xAVG = mean(estimated_ba_using_speedangle,
        na.rm = T), xwOBA = mean(estimated_woba_using_speedangle,
        na.rm = T), wOBA = mean(woba_value, na.rm = T), ISO = mean(iso_value,
        na.rm = T))

hit_data_outcomes <- hit_data_outcomes %>%
    inner_join(grouped_stats, by = c("batter", "zone"))

hit_data_outcomes <- hit_data_outcomes %>%
    separate(pitcher_name, into = c("last_name", "first_name"),
        sep = ", ") %>%
    mutate(pitcher_name = paste(first_name, last_name, sep = " "))

basic_pitcher_stats <- daily_pitcher_bref(startdate, enddate)

basic_pitcher_stats$bbref_id <- as.numeric(basic_pitcher_stats$bbref_id)

hit_data_outcomes <- hit_data_outcomes %>%
    inner_join(basic_pitcher_stats, by = c(pitcher_name = "Name"))

grouped_pitching_stats <- hit_data_outcomes %>%
    group_by(pitcher) %>%
    summarise(FF_spin_rate = mean(release_spin_rate[pitch_type ==
        "FF"], na.rm = T), CU_spin_rate = mean(release_spin_rate[pitch_type ==
        "CU"], na.rm = T), FF_velocity = mean(release_speed[pitch_type ==
        "FF"], na.rm = T), BRK_velocity = mean(release_speed[pitch_type ==
        "CU" | pitch_type == "SL"], na.rm = T), overall_spin_rate = mean(release_spin_rate,
        na.rm = T), overall_velocity = mean(release_speed,
        na.rm = T))

hit_data_outcomes <- hit_data_outcomes %>%
    inner_join(grouped_pitching_stats, by = "pitcher")
```

```r
########
hit_data_outcomes <- hit_data_outcomes %>%
    mutate(outcome = ifelse(description == "swinging_strike" |
        description == "swinging_strike_blocked" | description ==
        "foul_tip" | (strikes == 2 & description == "foul_bunt") |
        description == "bunt_foul_tip" | description == "missed_bunt",
        "Miss", ifelse(description == "foul" | description ==
            "foul_bunt", "Foul", ifelse(events == "single",
            "Single", ifelse(events == "double", "Double",
              ifelse(events == "triple", "Triple", ifelse(events ==
                "home_run", "Homerun", "Out")))))))

hit_data_outcomes <- hit_data_outcomes %>%
    filter(ERA <= 20)

all_data <- hit_data_outcomes %>%
    select(plate_x, plate_z, pfx_x, pfx_z, pitch_type, balls,
        strikes, outs_when_up, release_spin_rate, stand,
        p_throws, effective_speed, release_speed, PA, BA,
        OBP, SLG, barrel_perc, avg_exit_velocity, avg_launch_angle,
        xAVG, xwOBA, wOBA, ISO, IP, ERA, GB.FB, WHIP, SO_perc,
        uBB_perc, overall_spin_rate, overall_velocity, outcome)

all_lab <- as.factor(all_data$outcome)
all_lab <- as.numeric(as.factor(all_lab)) - 1

all_data$pitch_type <- as.numeric(as.factor(all_data$pitch_type)) -
    1
all_data$p_throws <- as.numeric(as.factor(all_data$p_throws)) -
    1
all_data$stand <- as.numeric(as.factor(all_data$stand)) -
    1

xgb.all = xgb.DMatrix(data = as.matrix(all_data[, -33]),
    label = all_lab)

## Predict
all.pred = predict(swing_mod, xgb.all, reshape = T)
all.pred = as.data.frame(all.pred)
names(all.pred) <- levels(all_data$outcome)
all.pred$label <- all_data$outcome

# Use the predicted label with the highest probability
# all.pred$prediction = apply(all.pred,1,function(x)
# colnames(all.pred)[which.max(x)])
colnames(all.pred)[1:7] <- final_lab

hit_data_outcomes <- cbind(hit_data_outcomes, all.pred)

# Calculate RE of a swing

hit_data_outcomes <- hit_data_outcomes %>%
    mutate(re_swing = avg_re.miss * Miss + avg_re.foul *
```

```
            Foul + avg_re.out * Out + avg_re.single * Single +
            avg_re.double * Double + avg_re.triple * Triple +
            avg_re.homerun * Homerun)

    # Now create the stat to evaluate players on Expected
    # Runs Added by swinging

    hit_data_outcomes <- hit_data_outcomes %>%
        mutate(expRA_swing = re_swing - re_take, swing = ifelse(description ==
            "ball" | description == "called_strike" | description ==
            "blocked_ball" | description == "pitchout", 0, 1))

    hit_data_outcomes <- hit_data_outcomes %>%
        mutate(abs_eRA_s = abs(expRA_swing), should_swing = ifelse(re_swing >
            re_take, 1, 0), correct_choice = ifelse(should_swing ==
            swing, 1, 0))

    hit_data_outcomes <- hit_data_outcomes %>%
        mutate(runs_lost_bad_dec = ifelse(correct_choice == 0,
            abs_eRA_s, 0))

    hit_data_outcomes
}
```

Apply the EAGLE function to the data

```
remove(hit_data)
data_2016 <- apply_EAGLE(pitches_2016, startdate = "2016-04-03",
    enddate = "2016-10-02")
```

```
## Data courtesy of Baseball-Reference.com. Please consider supporting Baseball-Reference by signing up

## `summarise()` has grouped output by 'batter'. You can override using the `.groups` argument.

## Data courtesy of Baseball-Reference.com. Please consider supporting Baseball-Reference by signing up
```

```
remove(pitches_2016)
data_2017 <- apply_EAGLE(pitches_2017, startdate = "2017-04-02",
    enddate = "2017-10-01")
```

```
## Data courtesy of Baseball-Reference.com. Please consider supporting Baseball-Reference by signing up

## `summarise()` has grouped output by 'batter'. You can override using the `.groups` argument.

## Data courtesy of Baseball-Reference.com. Please consider supporting Baseball-Reference by signing up
```

```
remove(pitches_2017)
data_2018 <- apply_EAGLE(pitches_2018, startdate = "2018-03-29",
    enddate = "2018-09-30")
```

```
## Data courtesy of Baseball-Reference.com. Please consider supporting Baseball-Reference by signing up

## `summarise()` has grouped output by 'batter'. You can override using the `.groups` argument.

## Data courtesy of Baseball-Reference.com. Please consider supporting Baseball-Reference by signing up
```

```
remove(pitches_2018)
data_2019 <- apply_EAGLE(pitches_2019, startdate = "2019-03-20",
    enddate = "2019-09-29")
```

```
## Data courtesy of Baseball-Reference.com. Please consider supporting Baseball-Reference by signing up
## `summarise()` has grouped output by 'batter'. You can override using the `.groups` argument.
## Data courtesy of Baseball-Reference.com. Please consider supporting Baseball-Reference by signing up
```

```
remove(pitches_2019)
```

Create combined season data frame with commonly used metrics and EAGLE so that we have each hitter from 2016-2019

```
eye_2016 <- data_2016 %>%
    filter(PA >= 200, !is.na(expRA_swing)) %>%
    group_by(batter, Name) %>%
    summarize(year = first(season.x), runs_lost = sum(runs_lost_bad_dec)/sum(abs_eRA_s),
        wRA = sum(ifelse(swing == 1, expRA_swing, -expRA_swing)),
        EAGLE = wRA/n(), avg = first(BA), obp = first(OBP), ops = first(OPS),
        kperc = first(SO.x)/first(PA), bbperc = first(BB.x)/first(PA),
        perc_swing_out_zone = sum(ifelse(strike_prob < 0.3 &
            swing == 1, 1, 0))/sum(ifelse(strike_prob < 0.3,
            1, 0)))
```

```
## `summarise()` has grouped output by 'batter'. You can override using the `.groups` argument.
```

```
eye_2017 <- data_2017 %>%
    filter(PA >= 200, !is.na(expRA_swing)) %>%
    group_by(batter, Name) %>%
    summarize(year = first(season.x), runs_lost = sum(runs_lost_bad_dec)/sum(abs_eRA_s),
        wRA = sum(ifelse(swing == 1, expRA_swing, -expRA_swing)),
        EAGLE = wRA/n(), avg = first(BA), obp = first(OBP), ops = first(OPS),
        kperc = first(SO.x)/first(PA), bbperc = first(BB.x)/first(PA),
        perc_swing_out_zone = sum(ifelse(strike_prob < 0.3 &
            swing == 1, 1, 0))/sum(ifelse(strike_prob < 0.3,
            1, 0)))
```

```
## `summarise()` has grouped output by 'batter'. You can override using the `.groups` argument.
```

```
eye_2018 <- data_2018 %>%
    filter(PA >= 200, !is.na(expRA_swing)) %>%
    group_by(batter, Name) %>%
    summarize(year = first(season.x), runs_lost = sum(runs_lost_bad_dec)/sum(abs_eRA_s),
        wRA = sum(ifelse(swing == 1, expRA_swing, -expRA_swing)),
        EAGLE = wRA/n(), avg = first(BA), obp = first(OBP), ops = first(OPS),
        kperc = first(SO.x)/first(PA), bbperc = first(BB.x)/first(PA),
        perc_swing_out_zone = sum(ifelse(strike_prob < 0.3 &
            swing == 1, 1, 0))/sum(ifelse(strike_prob < 0.3,
            1, 0)))
```

```
## `summarise()` has grouped output by 'batter'. You can override using the `.groups` argument.
```

```
eye_2019 <- data_2019 %>%
    filter(PA >= 200, !is.na(expRA_swing)) %>%
    group_by(batter, Name) %>%
    summarize(year = first(season.x), runs_lost = sum(runs_lost_bad_dec)/sum(abs_eRA_s),
        wRA = sum(ifelse(swing == 1, expRA_swing, -expRA_swing)),
        EAGLE = wRA/n(), avg = first(BA), obp = first(OBP), ops = first(OPS),
        kperc = first(SO.x)/first(PA), bbperc = first(BB.x)/first(PA),
        perc_swing_out_zone = sum(ifelse(strike_prob < 0.3 &
```

```
            swing == 1, 1, 0))/sum(ifelse(strike_prob < 0.3,
            1, 0)))
```

```
## `summarise()` has grouped output by 'batter'. You can override using the `.groups` argument.
```

```
eye_all <- eye_2016 %>%
    full_join(eye_2017, by = c("batter")) %>%
    full_join(eye_2018, by = c("batter")) %>%
    full_join(eye_2019, by = c("batter"))

eye_all <- eye_all %>%
    rename(wRL_2016 = runs_lost.x, wRL_2017 = runs_lost.y, wRL_2018 = runs_lost.x.x,
        wRL_2019 = runs_lost.y.y)
```

Create data frame containing player stats from consecutive years so that we can see predictabililty and correlation across years

```
eye_this_next1 <- eye_2016 %>%
    left_join(eye_2017, by = c("batter", "Name"))
eye_this_next2 <- eye_2017 %>%
    left_join(eye_2018, by = c("batter", "Name"))
eye_this_next3 <- eye_2018 %>%
    left_join(eye_2019, by = c("batter", "Name"))

eye_this_next <- bind_rows(eye_this_next1, eye_this_next2, eye_this_next3)

remove(eye_this_next1, eye_this_next2, eye_this_next3)
```

Look at results of outcome probabilities xgboost model

```
all_years <- rbind(data_2016, data_2017, data_2018, data_2019)

levs = c("Miss", "Foul", "Out", "Single", "Double", "Triple",
    "Homerun")

all_years %>%
    filter(!is.na(outcome)) %>%
    group_by(outcome) %>%
    summarize(Miss = mean(Miss, na.rm = T), Foul = mean(Foul,
        na.rm = T), Out = mean(Out, na.rm = T), Single = mean(Single,
        na.rm = T), Double = mean(Double, na.rm = T), Triple = mean(Triple,
        na.rm = T), Homerun = mean(Homerun, na.rm = T)) %>%
    mutate(outcome = factor(outcome, levels = levs)) %>%
    arrange(outcome)
```

```
## # A tibble: 7 x 8
##   outcome  Miss  Foul   Out Single Double  Triple Homerun
##   <fct>   <dbl> <dbl> <dbl>  <dbl>  <dbl>   <dbl>   <dbl>
## 1 Miss    0.410 0.328 0.181 0.0532 0.0149 0.00158  0.0107
## 2 Foul    0.217 0.421 0.243 0.0751 0.0234 0.00235  0.0178
## 3 Out     0.227 0.389 0.265 0.0784 0.0228 0.00223  0.0154
## 4 Single  0.173 0.396 0.272 0.110  0.0289 0.00270  0.0177
## 5 Double  0.160 0.407 0.266 0.0932 0.0423 0.00374  0.0277
## 6 Triple  0.152 0.412 0.267 0.0906 0.0397 0.00791  0.0304
## 7 Homerun 0.164 0.420 0.247 0.0763 0.0366 0.00385  0.0523
```

Run linear regression models in order to test for significant relationships between EAGLE and other significant statistics

```r
#### Test EAGLE correlation from year to year
eye_this_next_mod <- lm(EAGLE.y ~ EAGLE.x, data = eye_this_next)
summary(eye_this_next_mod)
```

```
##
## Call:
## lm(formula = EAGLE.y ~ EAGLE.x, data = eye_this_next)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0243118 -0.0060932 -0.0001419  0.0064859  0.0265048
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 0.031851   0.001747   18.23 <0.0000000000000002 ***
## EAGLE.x     0.473621   0.031076   15.24 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.008718 on 697 degrees of freedom
##   (231 observations deleted due to missingness)
## Multiple R-squared:   0.25,  Adjusted R-squared:  0.2489
## F-statistic: 232.3 on 1 and 697 DF,  p-value: < 0.00000000000000022
```

```r
####

# Test with AVG
ba_eye_mod <- lm(avg.x ~ EAGLE.x, data = eye_this_next)
summary(ba_eye_mod)
```

```
##
## Call:
## lm(formula = avg.x ~ EAGLE.x, data = eye_this_next)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.093363 -0.019715  0.000168  0.020347  0.093616
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 0.244893   0.005042  48.573 <0.0000000000000002 ***
## EAGLE.x     0.209096   0.090604   2.308              0.0212 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03127 on 928 degrees of freedom
## Multiple R-squared:  0.005706,   Adjusted R-squared:  0.004635
## F-statistic: 5.326 on 1 and 928 DF,  p-value: 0.02123
```

```r
next_ba_eye_mod <- lm(avg.y ~ EAGLE.x, eye_this_next)
summary(next_ba_eye_mod)
```

```
##
```

```
## Call:
## lm(formula = avg.y ~ EAGLE.x, data = eye_this_next)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.099888 -0.020135 -0.000895  0.020736  0.090513
##
## Coefficients:
##              Estimate Std. Error t value            Pr(>|t|)
## (Intercept)  0.267703   0.006372  42.010 <0.0000000000000002 ***
## EAGLE.x     -0.196482   0.113373  -1.733              0.0835 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0318 on 697 degrees of freedom
##   (231 observations deleted due to missingness)
## Multiple R-squared:  0.004291,   Adjusted R-squared:  0.002862
## F-statistic: 3.003 on 1 and 697 DF,  p-value: 0.08353
```

```
both_ba_eye_mod <- lm(avg.y ~ EAGLE.x + avg.x, eye_this_next)
summary(both_ba_eye_mod)
```

```
##
## Call:
## lm(formula = avg.y ~ EAGLE.x + avg.x, data = eye_this_next)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.090794 -0.018762  0.001494  0.017980  0.090254
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)  0.14593    0.01069  13.654 <0.0000000000000002 ***
## EAGLE.x     -0.32349    0.10151  -3.187              0.0015 **
## avg.x        0.49217    0.03659  13.452 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02835 on 696 degrees of freedom
##   (231 observations deleted due to missingness)
## Multiple R-squared:  0.2098, Adjusted R-squared:  0.2075
## F-statistic: 92.37 on 2 and 696 DF,  p-value: < 0.00000000000000022
```

```
# Test for OBP
obp_eye_mod <- lm(obp.x ~ EAGLE.x, data = eye_this_next)
summary(obp_eye_mod)
```

```
##
## Call:
## lm(formula = obp.x ~ EAGLE.x, data = eye_this_next)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.096967 -0.021373  0.000221  0.019977  0.120982
##
```

```
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 0.253075   0.005268   48.04 <0.0000000000000002 ***
## EAGLE.x     1.356216   0.094678   14.32 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03268 on 928 degrees of freedom
## Multiple R-squared:  0.1811, Adjusted R-squared:  0.1802
## F-statistic: 205.2 on 1 and 928 DF,  p-value: < 0.00000000000000022
```

```
next_obp_eye_mod <- lm(obp.y ~ EAGLE.x, eye_this_next)
summary(next_obp_eye_mod)
```

```
##
## Call:
## lm(formula = obp.y ~ EAGLE.x, data = eye_this_next)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.116749 -0.022555 -0.000317  0.023362  0.123680
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept) 0.276380   0.006818  40.537 < 0.0000000000000002 ***
## EAGLE.x     0.988212   0.121302   8.147  0.00000000000000173 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03403 on 697 degrees of freedom
##   (231 observations deleted due to missingness)
## Multiple R-squared:  0.08694,    Adjusted R-squared:  0.08563
## F-statistic: 66.37 on 1 and 697 DF,  p-value: 0.000000000000001729
```

```
both_obp_eye_mod <- lm(obp.y ~ EAGLE.x + obp.x, eye_this_next)
summary(both_obp_eye_mod)
```

```
##
## Call:
## lm(formula = obp.y ~ EAGLE.x + obp.x, data = eye_this_next)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.091149 -0.019740 -0.000989  0.020860  0.100273
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)  0.15739    0.01110  14.178 <0.0000000000000002 ***
## EAGLE.x      0.21025    0.12476   1.685              0.0924 .
## obp.x        0.48624    0.03781  12.860 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03061 on 696 degrees of freedom
##   (231 observations deleted due to missingness)
```

```
## Multiple R-squared:  0.2622, Adjusted R-squared:  0.2601
## F-statistic: 123.7 on 2 and 696 DF,  p-value: < 0.00000000000000022
```

```r
# Now for OPS
ops_eye_mod <- lm(ops.x ~ EAGLE.x, data = eye_this_next)
summary(ops_eye_mod)
```

```
##
## Call:
## lm(formula = ops.x ~ EAGLE.x, data = eye_this_next)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.26205 -0.05736  0.00274  0.05298  0.32736
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)  0.55572    0.01399   39.73 <0.0000000000000002 ***
## EAGLE.x      3.61467    0.25139   14.38 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08676 on 928 degrees of freedom
## Multiple R-squared:  0.1822, Adjusted R-squared:  0.1813
## F-statistic: 206.8 on 1 and 928 DF,  p-value: < 0.00000000000000022
```

```r
next_ops_eye_mod <- lm(ops.y ~ EAGLE.x, eye_this_next)
summary(next_ops_eye_mod)
```

```
##
## Call:
## lm(formula = ops.y ~ EAGLE.x, data = eye_this_next)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32288 -0.06165 -0.00418  0.05682  0.31270
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  0.65606    0.01962  33.442 < 0.0000000000000002 ***
## EAGLE.x      1.99812    0.34903   5.725         0.0000000154 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09792 on 697 degrees of freedom
##   (231 observations deleted due to missingness)
## Multiple R-squared:  0.04491,    Adjusted R-squared:  0.04354
## F-statistic: 32.77 on 1 and 697 DF,  p-value: 0.00000001539
```

```r
both_ops_eye_mod <- lm(ops.y ~ EAGLE.x + ops.x, eye_this_next)
summary(both_ops_eye_mod)
```

```
##
## Call:
## lm(formula = ops.y ~ EAGLE.x + ops.x, data = eye_this_next)
##
```

```
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.260115 -0.058094 -0.004444  0.053218  0.296292
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)  0.38854    0.02865  13.562 <0.0000000000000002 ***
## EAGLE.x     -0.08645    0.36283  -0.238               0.812
## ops.x        0.49631    0.04153  11.951 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08925 on 696 degrees of freedom
##   (231 observations deleted due to missingness)
## Multiple R-squared:  0.2075, Adjusted R-squared:  0.2053
## F-statistic: 91.14 on 2 and 696 DF,  p-value: < 0.00000000000000022
```

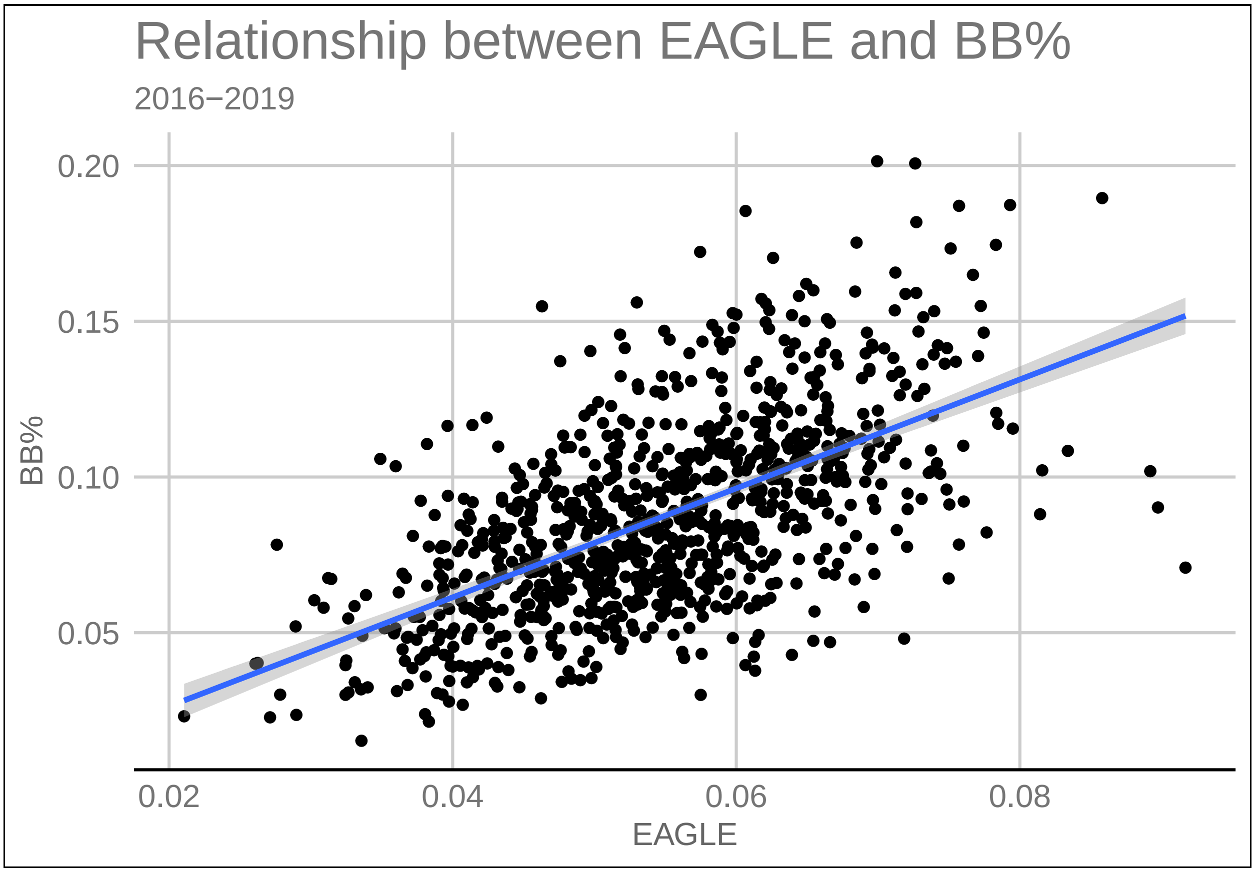
```
###
```

#Create graphs for important relationships

First create graph for walk percentage while also testing the relationship with a linear regression model

```
eye_this_next %>%
    filter(EAGLE.x >= 0) %>%
    ggplot(aes(x = EAGLE.x, y = bbperc.x)) + geom_point() + stat_smooth(method = "lm") +
    theme_gdocs() + labs(title = "Relationship between EAGLE and BB%",
    subtitle = "2016-2019") + xlab("EAGLE") + ylab("BB%")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
bbperc_mod <- lm(bbperc.x ~ EAGLE.x, data = eye_this_next)
summary(bbperc_mod)
```

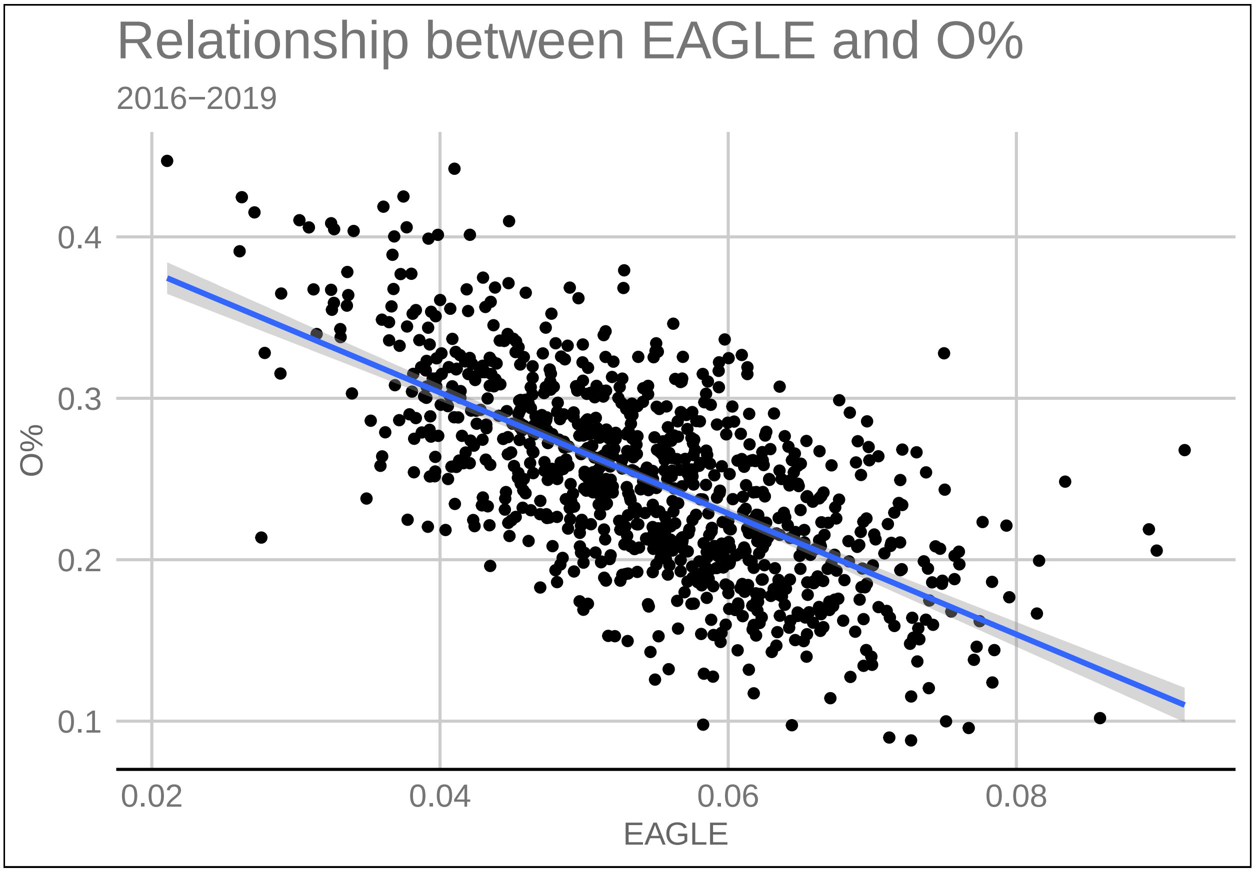
```
##
## Call:
## lm(formula = bbperc.x ~ EAGLE.x, data = eye_this_next)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.073919 -0.017433 -0.002997  0.015202  0.134005
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 0.002273   0.004152   0.548               0.584
## EAGLE.x     1.554790   0.074606  20.840 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02575 on 928 degrees of freedom
## Multiple R-squared:  0.3188, Adjusted R-squared:  0.3181
## F-statistic: 434.3 on 1 and 928 DF,  p-value: < 0.00000000000000022
```

Graph for relationship with O% and also test it with a linear regression model

```
eye_this_next %>%
    filter(EAGLE.x >= 0) %>%
    ggplot(aes(x = EAGLE.x, y = perc_swing_out_zone.x)) + geom_point() +
```

```
    stat_smooth(method = "lm") + theme_gdocs() + labs(title = "Relationship between EAGLE and O%",
    subtitle = "2016-2019") + xlab("EAGLE") + ylab("O%")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



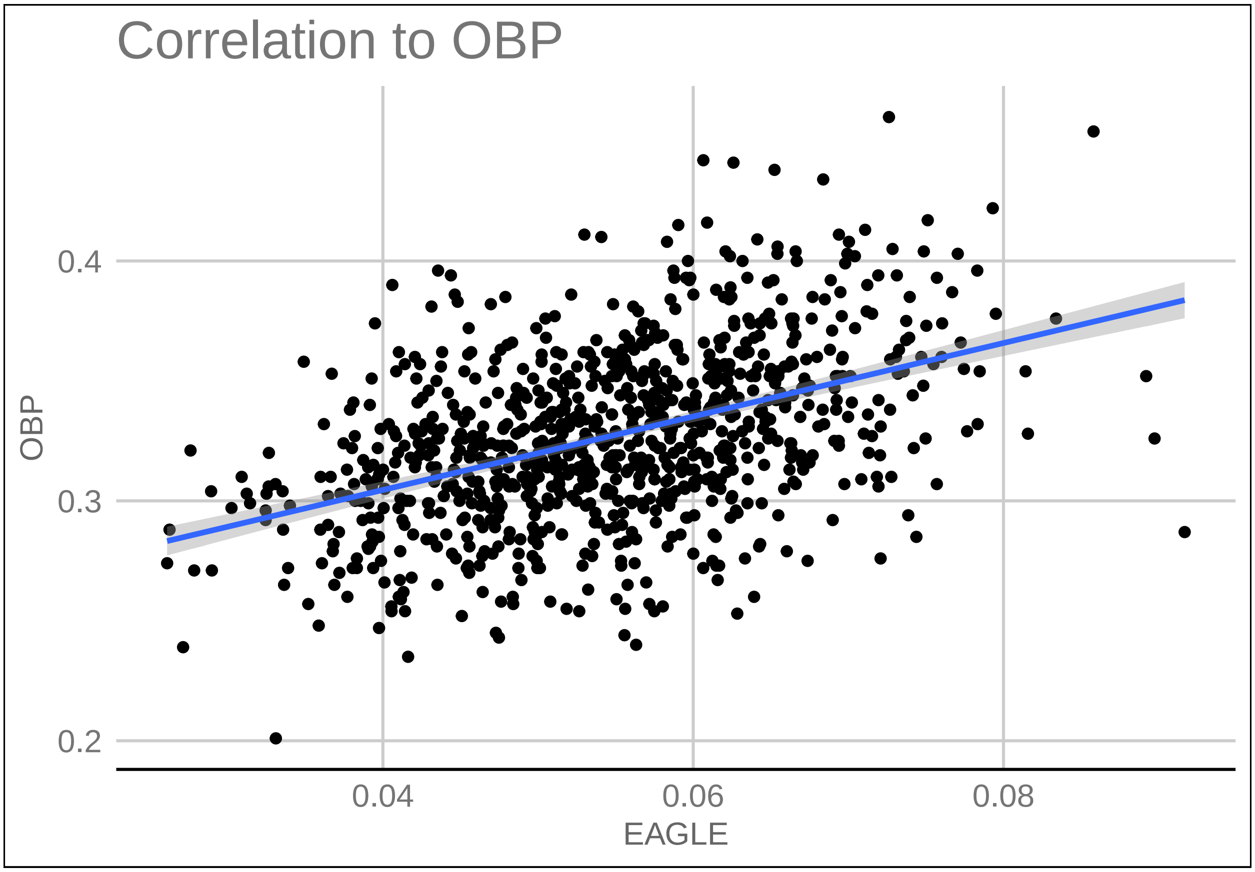

```
operc_mod <- lm(perc_swing_out_zone.x ~ EAGLE.x, data = eye_this_next)
summary(operc_mod)
```

```
##
## Call:
## lm(formula = perc_swing_out_zone.x ~ EAGLE.x, data = eye_this_next)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.14505 -0.03294 -0.00073  0.02953  0.49238
##
## Coefficients:
##              Estimate Std. Error t value            Pr(>|t|)
## (Intercept)  0.470556   0.007833   60.08 <0.0000000000000002 ***
## EAGLE.x     -4.049996   0.140759  -28.77 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04858 on 928 degrees of freedom
## Multiple R-squared:  0.4715, Adjusted R-squared:  0.4709
## F-statistic: 827.9 on 1 and 928 DF,  p-value: < 0.00000000000000022
```

Create graph for relationship of EAGLE with OBP

```
eye_this_next %>%
    filter(!is.na(EAGLE.x), !is.na(obp.x), EAGLE.x >= 0.025) %>%
    ggplot(aes(EAGLE.x, obp.x)) + geom_point() + stat_smooth(method = "lm") +
    theme_gdocs() + ggtitle("Correlation to OBP") + xlab("EAGLE") +
    ylab("OBP")
```

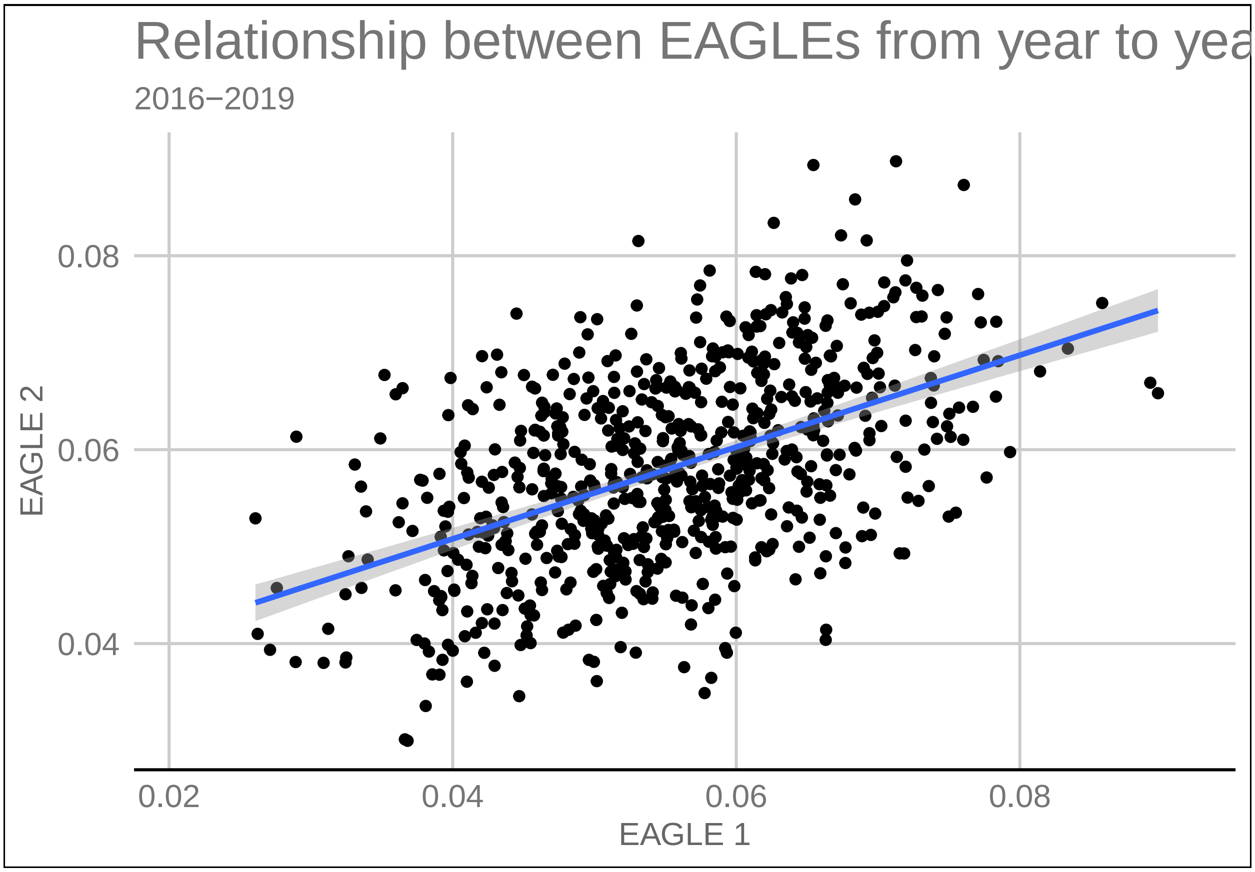
```
## `geom_smooth()` using formula 'y ~ x'
```



Create graph to highlight the correlation of EAGLE from year to year

```
eye_this_next %>%
    filter(EAGLE.x >= 0) %>%
    ggplot(aes(x = EAGLE.x, y = EAGLE.y)) + geom_point() + stat_smooth(method = "lm") +
    theme_gdocs() + labs(title = "Relationship between EAGLEs from year to year",
    subtitle = "2016-2019") + xlab("EAGLE 1") + ylab("EAGLE 2")
```

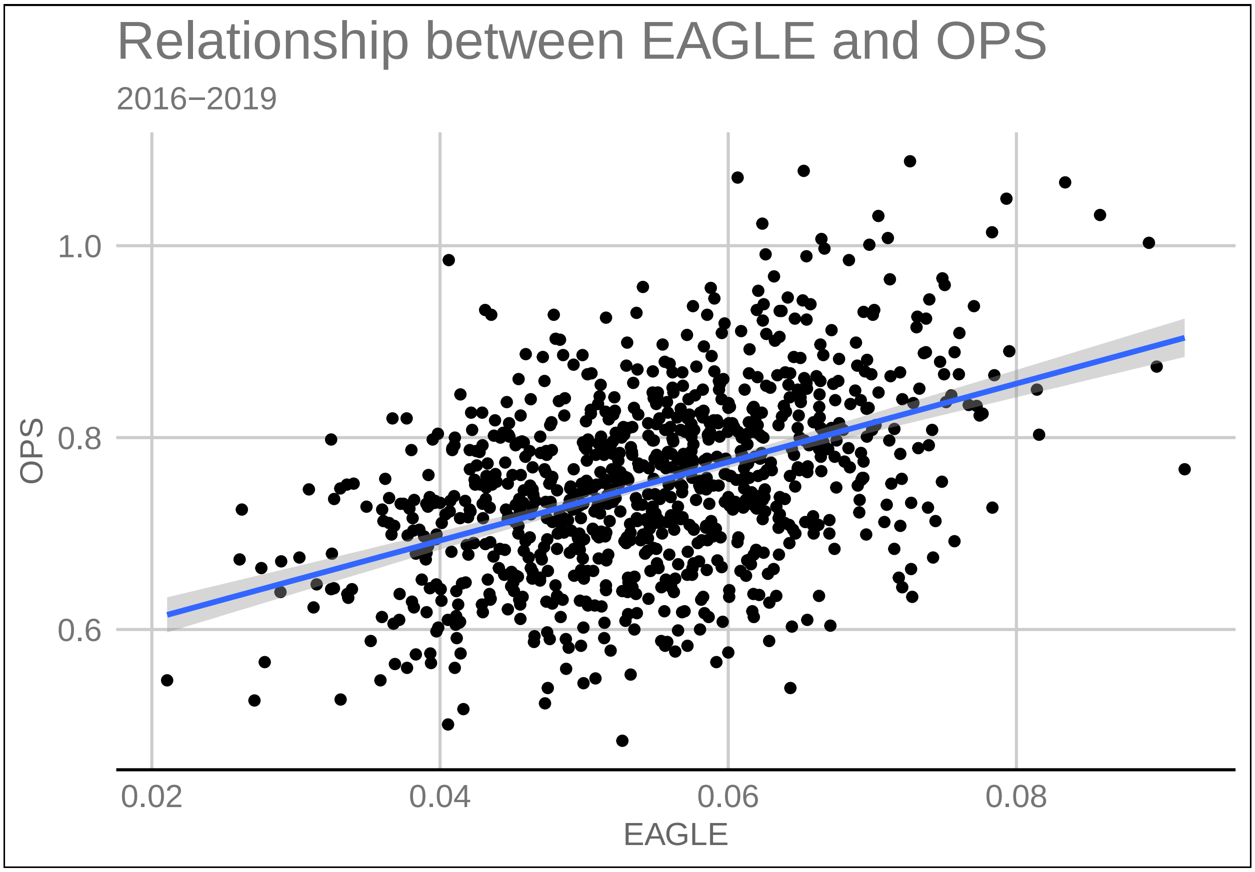
```
## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 229 rows containing non-finite values (stat_smooth).

## Warning: Removed 229 rows containing missing values (geom_point).
```



Create graphs showing the relationship of EAGLE with the current year's OPS and with the next year's OPS

```
eye_this_next %>%
    filter(EAGLE.x >= 0) %>%
    ggplot(aes(x = EAGLE.x, y = ops.x)) + geom_point() + stat_smooth(method = "lm") +
    theme_gdocs() + labs(title = "Relationship between EAGLE and OPS",
    subtitle = "2016-2019") + xlab("EAGLE") + ylab("OPS")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

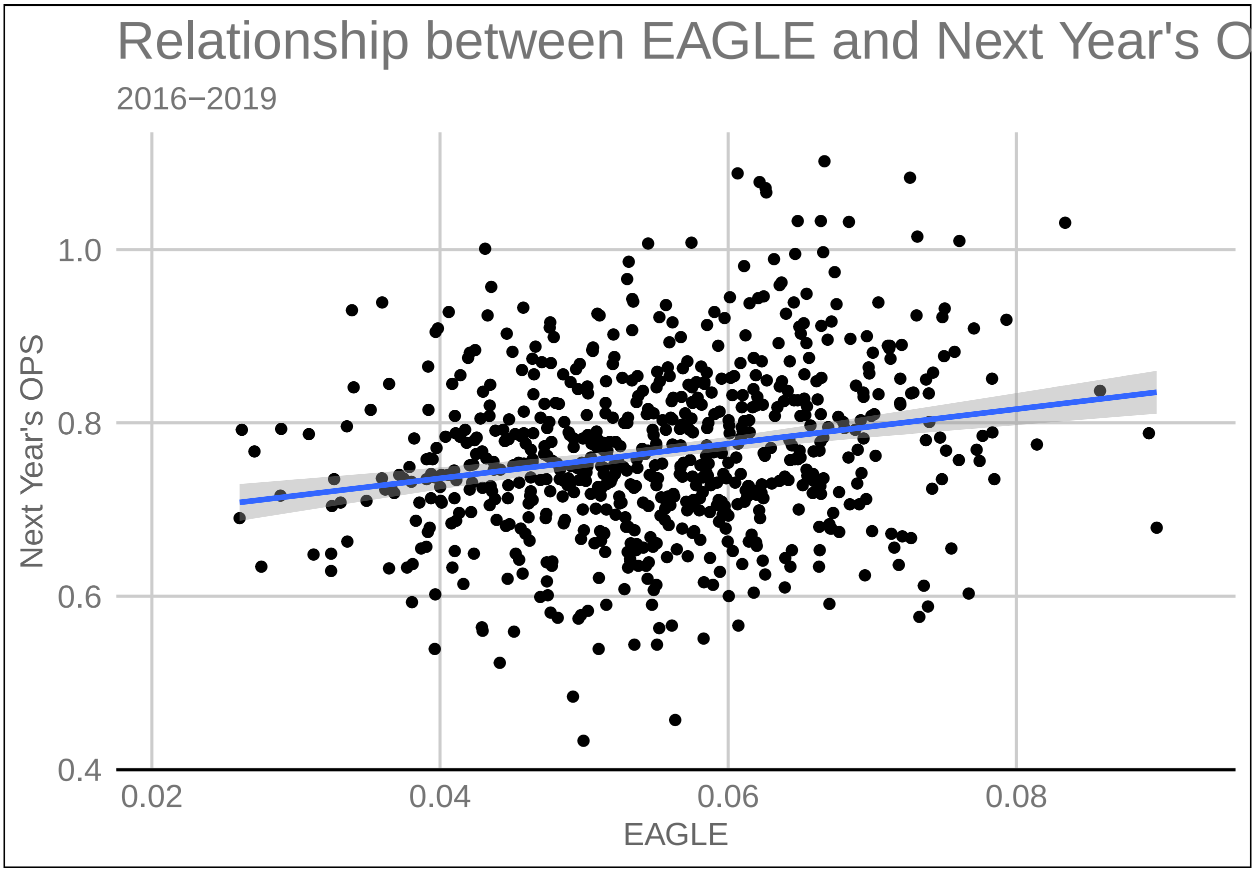



```
eye_this_next %>%
    filter(EAGLE.x >= 0) %>%
    ggplot(aes(x = EAGLE.x, y = ops.y)) + geom_point() + stat_smooth(method = "lm") +
    theme_gdocs() + labs(title = "Relationship between EAGLE and Next Year's OPS",
    subtitle = "2016-2019") + xlab("EAGLE") + ylab("Next Year's OPS")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 229 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 229 rows containing missing values (geom_point).
```



Bring the data of all the years together to calculate predictions from percentiles in addition to testing correlation. Also check out who is the best/worst in EAGLE

```
all_eye <- all_years %>%
    mutate(season.x = as.numeric(season.x)) %>%
    filter(PA >= 300, !is.na(expRA_swing)) %>%
    group_by(Name, season.x) %>%
    summarize(runs_lost = sum(runs_lost_bad_dec)/sum(abs_eRA_s),
        wRA = sum(ifelse(swing == 1, expRA_swing, -expRA_swing)),
        ba = first(BA), obp = first(OBP), slg = first(SLG), ops = first(OPS),
        pitches = n(), bb_perc = first(BB.x)/first(PA), o_perc = sum(ifelse(strike_prob <
            0.5 & swing == 1, 1, 0), na.rm = T)/sum(ifelse(strike_prob <
            0.5, 1, 0), na.rm = T)) %>%
    mutate(EAGLE = wRA/pitches)
```

```
## `summarise()` has grouped output by 'Name'. You can override using the `.groups` argument.
```

```
all_eye %>%
    select(Name, season.x, EAGLE, bb_perc, o_perc) %>%
    arrange(-EAGLE) %>%
    head()
```

```
## # A tibble: 6 x 5
## # Groups:   Name [6]
##   Name               season.x  EAGLE bb_perc o_perc
##   <chr>                 <dbl>  <dbl>   <dbl>  <dbl>
## 1 Khris Davis            2018 0.0897  0.0902  0.235
## 2 Juan Soto              2019 0.0893  0.164   0.177
```

```
## 3 Anthony Rendon      2019 0.0873  0.124   0.201
## 4 Joey Votto          2017 0.0858  0.190   0.126
## 5 J.D. Martinez       2017 0.0834  0.108   0.276
## 6 George Springer     2019 0.0821  0.121   0.186
```

```
all_eye %>%
    select(Name, season.x, EAGLE, bb_perc, o_perc) %>%
    arrange(EAGLE) %>%
    head()
```

```
## # A tibble: 6 x 5
## # Groups:   Name [6]
##   Name                 season.x  EAGLE bb_perc o_perc
##   <chr>                   <dbl>  <dbl>   <dbl>  <dbl>
## 1 Freddy Galvis            2016 0.0261  0.0401  0.405
## 2 Salvador Perez           2016 0.0262  0.0403  0.445
## 3 Billy Hamilton           2016 0.0276  0.0783  0.239
## 4 Billy Burns              2016 0.0278  0.0301  0.345
## 5 Dee Strange-Gordon       2016 0.0289  0.0520  0.336
## 6 Gerardo Parra            2016 0.0290  0.0236  0.389
```

```
eagle_perc <- quantile(all_eye$EAGLE, probs = c(0.1, 0.25, 0.75,
    0.9))
percentiles <- data_frame(percentile = c("10th percentile", "25th percentile",
    "75th percentile", "90th percentile"), EAGLE.x = eagle_perc)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
```

```
percentiles %>%
    add_predictions(ops_eye_mod, var = "OPS")
```

```
## # A tibble: 4 x 3
##   percentile      EAGLE.x   OPS
##   <chr>             <dbl> <dbl>
## 1 10th percentile  0.0420 0.707
## 2 25th percentile  0.0488 0.732
## 3 75th percentile  0.0629 0.783
## 4 90th percentile  0.0694 0.807
```

```
percentiles %>%
    add_predictions(next_ops_eye_mod, var = "Next_OPS")
```

```
## # A tibble: 4 x 3
##   percentile      EAGLE.x Next_OPS
##   <chr>             <dbl>    <dbl>
## 1 10th percentile  0.0420    0.740
## 2 25th percentile  0.0488    0.754
## 3 75th percentile  0.0629    0.782
## 4 90th percentile  0.0694    0.795
```

```
cor(all_eye$EAGLE, all_eye$ops)
```

```
## [1] 0.4811066
```

```
cor(all_eye$o_perc, all_eye$ops)
```

```
## [1] -0.1391981
```

Current Year EAGLE correlated with current year statistics and next year's EAGLE

```
eye_this_next %>%
    ungroup() %>%
    select(EAGLE.x, avg.x, obp.x, ops.x, kperc.x, bbperc.x, perc_swing_out_zone.x,
        EAGLE.y) %>%
    rename(o_perc = perc_swing_out_zone.x, EAGLE_cur_year = EAGLE.x,
        EAGLE_next_year = EAGLE.y) %>%
    cor(use = "complete.obs")
```

```
##                 EAGLE_cur_year         avg.x      obp.x       ops.x     kperc.x
## EAGLE_cur_year      1.00000000  0.09300825  0.4848711  0.48072229  0.17974473
## avg.x               0.09300825  1.00000000  0.7040934  0.68931152 -0.48648571
## obp.x               0.48487114  0.70409339  1.0000000  0.80350785 -0.24907403
## ops.x               0.48072229  0.68931152  0.8035079  1.00000000 -0.07120939
## kperc.x             0.17974473 -0.48648571 -0.2490740 -0.07120939  1.00000000
## bbperc.x            0.60456365 -0.05104823  0.6507292  0.41077364  0.14740155
## o_perc             -0.66161729  0.07438396 -0.4371811 -0.14633563 -0.01996747
## EAGLE_next_year     0.49995699 -0.03585982  0.3033430  0.28183145  0.18078968
##                    bbperc.x      o_perc EAGLE_next_year
## EAGLE_cur_year   0.60456365 -0.66161729      0.49995699
## avg.x           -0.05104823  0.07438396     -0.03585982
## obp.x            0.65072916 -0.43718107      0.30334302
## ops.x            0.41077364 -0.14633563      0.28183145
## kperc.x          0.14740155 -0.01996747      0.18078968
## bbperc.x         1.00000000 -0.72510245      0.48754849
## o_perc          -0.72510245  1.00000000     -0.49166837
## EAGLE_next_year  0.48754849 -0.49166837      1.00000000
```