Analyzing MLB 2-Pitch Sequences: Does Swinging on the First Pitch Lead to Better Decisions?

Amor Ai

2023-05-25

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE)
```

```
swing <- read.csv("/Users/aa/Desktop/SwingDecisionsData.csv")
swing <- na.omit(swing)

#Creating new column that dichotomizes SeasonDecisionRuns
swing$WinContrib <- ifelse(swing[,19] > 0, 1, 0)

#View(swing)
```

Data Description

In this project, we will be working with a dataset that contains 414,799 pairs of pitches from the beginning of the 2023 Major League Baseball (MLB) season. Each row in the dataset corresponds to 2-pitch sequence (first 2 pitches thrown to a batter), and we have corresponding contextual information about each player's at-bat. The descriptions of all the variables in this dataset are listed below:

- `DecisionScore_2`: Value of the swing/take (no swing) decision made based on what the projected values of a swing and of a take would be for the 2nd pitch.
- `IsSwing_2`: 1 if batter swung at the 2nd pitch, 0 otherwise.
- `IsChase_2`: 1 if the batter swung at the 2nd pitch that was outside of the strike zone, 0 otherwise.
- `IsInZone_2`: 1 if the 2nd pitch was in the strike zone, 0 otherwise.
- `IsBall_2`: 1 if the 2nd pitch was called a ball, 0 otherwise.
- `PitchType_2`: Pitch type of 2nd pitch (FB = Four-Seam Fastball, CH = Changeup, CU = Curveball, SL = Slider, CT = Cutter).
- `Velocity_2`: Velocity of the 2nd pitch (in miles per hour).
- `DecisionScore_1`: Value of the swing/take decision made based on what the projected values of a swing and of a take would be for the 1st pitch
- `IsSwing_1`: 1 if batter swung at the 1st pitch, 0 otherwise.
- `IsChase_1`: 1 if the batter swung at the 1st pitch that was outside of the strike zone, 0 otherwise.
- `IsInZone_1`: 1 if the 1st pitch was in the strike zone, 0 otherwise.
- `IsBall_1`: 1 if the 1st pitch was called a ball, 0 otherwise.
- `PitchType_1`: Pitch type of 1st pitch (FB = Four-Seam Fastball, CH = Changeup, CU = Curveball, SL = Slider, CT = Cutter).
- `Velocity_2`: Velocity of the 2nd pitch (in miles per hour).
- `Velocity_1`: Velocity of the 1st pitch (in miles per hour).
- `SeasonPitches`: Number of pitches seen throughout the season so far.
- `SeasonSwingPct`: Percentage of pitches swung at throughout the season so far.

- `SeasonDecisionScore`: Average decision score throughout the season so far.
- `SeasonSwingProbability`: Expected swing percentage throughout the season so far, controlling for pitch type and location of the pitches.
- `SeasonDecisionRuns`: Calculated runs above average value on swing decisions throughout the season (i.e., runs into win contributions — the value of how much do a player contributes to a win through their batting outcomes)

Along with these 19 variables, we created an additional covariate, `WinContrib`, that dichotomizes `SeasonDecisionRuns`.

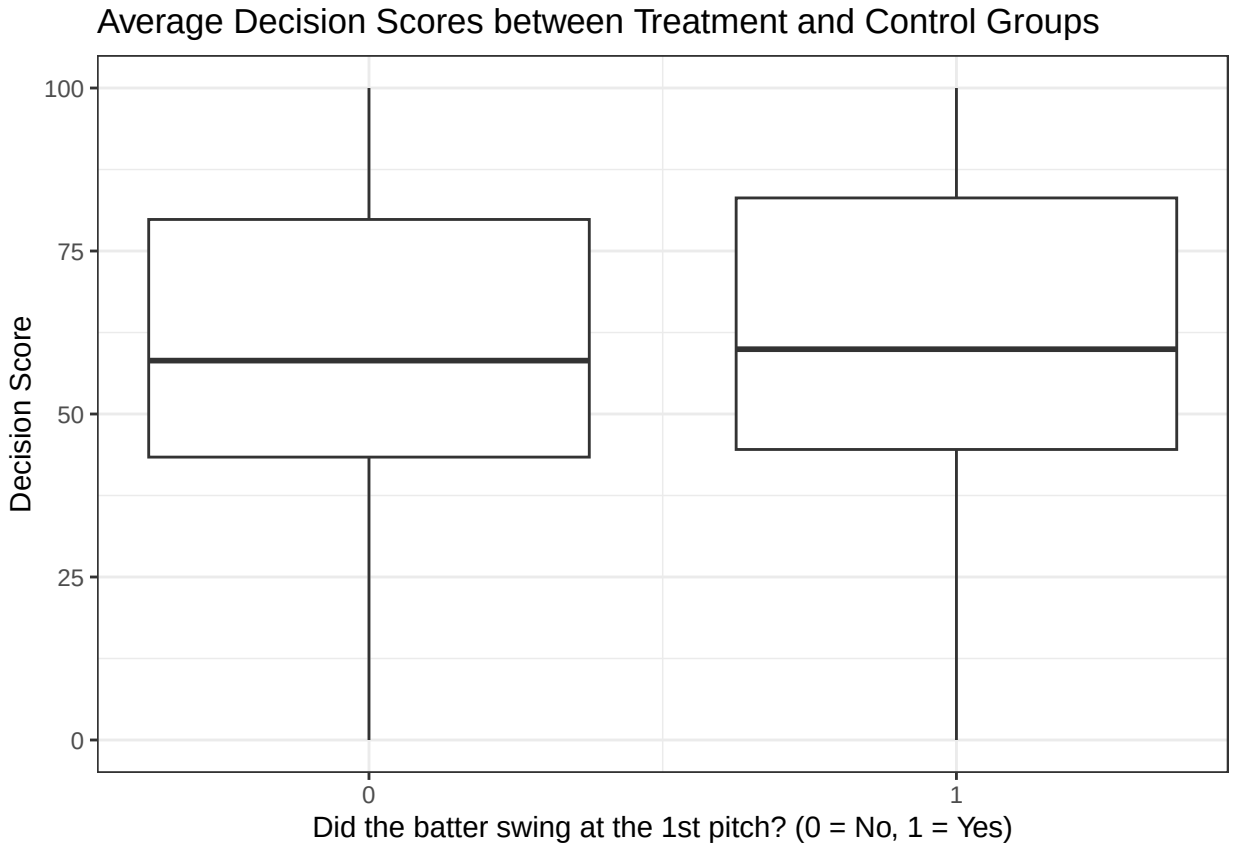
- `WinContrib`: Measure of batter's win contribution across the season; 1 if `SeasonDecisionRuns` is positive, 0 otherwise. (Higher/positive values means that batter is contributing more wins for their team)

Goal/Purpose

The overall purpose of this project is to analyze, in a 2-pitch sequence, the effect of the first pitch on the outcome of the second pitch. More specifically, does swinging on a first pitch lead to better or worse swing decisions on the subsequent pitch? In this dataset, `IsSwing_1` is the treatment variable — though not randomly assigned — and `DecisionScore_2` is the outcome variable, while all other variables are covariates. To focus on all at-bats with a Decision Score, we will remove all the rows without a decision score (`DecisionScore_2` = NA), leaving us with 410,117 observations. Of these observations, we have 268,446 control subjects and 141,671 treated subjects, with a mean decision score of 59 and ranging from 0 to 100 (higher decision scores are an indication of better decision-making from the batter). Since this is a very large dataset, it will be computationally challenging to implement several causal inference methods; thus, for the purposes of this project, we will focus on a random subset of the data (more on this in the methods section below).

What if this dataset came from a completely randomized experiment? (i.e., if `IsSwing_1` was randomly assigned)

```
swing %>%
  ggplot(aes(x = IsSwing_1, y = DecisionScore_2, group = IsSwing_1)) +
  geom_boxplot() +
  labs(title = "Average Decision Scores between Treatment and Control Groups",
       x = "Did the batter swing at the 1st pitch? (0 = No, 1 = Yes)",
       y = "Decision Score") +
  scale_x_continuous(breaks=seq(0,1,1), labels = c("0","1")) +
  theme_bw()
```

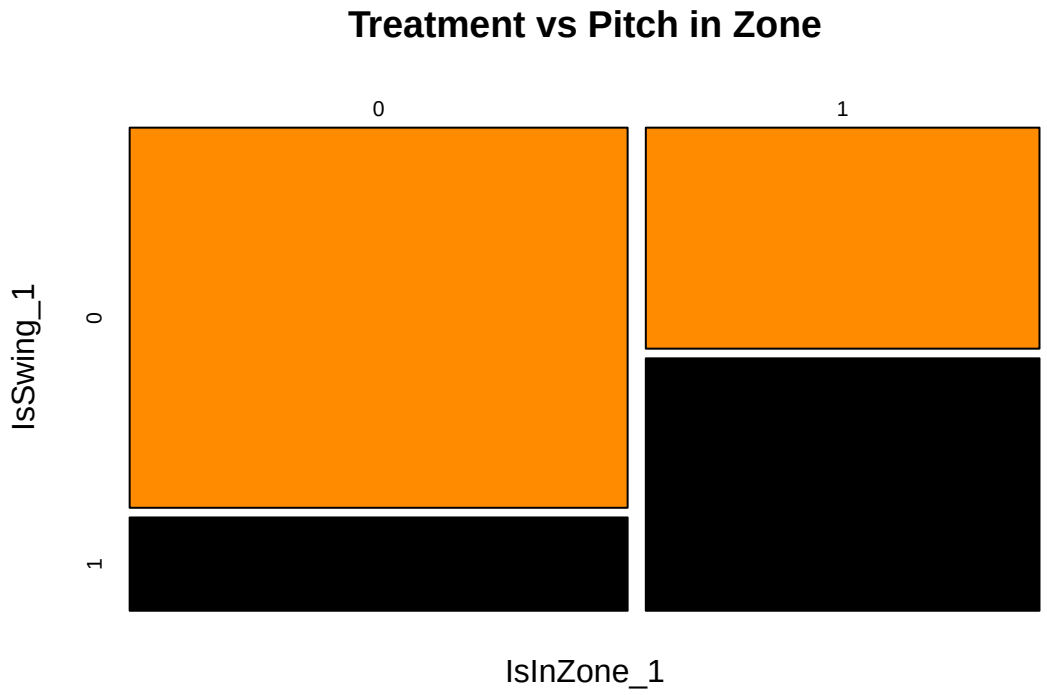



```
treat <- subset(swing, IsSwing_1 == 1)
control <- subset(swing, IsSwing_1 == 0)
ybar1 <- mean(treat$DecisionScore_2)
ybar0 <- mean(control$DecisionScore_2)
est <- ybar1 - ybar0 #ATE

s1_sq <- var(treat$DecisionScore_2)
s2_sq <- var(control$DecisionScore_2)
estVar <- s1_sq/nrow(treat) + s2_sq/nrow(control) #var

ci <- c(est - qnorm(0.975)*sqrt(estVar), est + qnorm(0.975)*sqrt(estVar))

#est
#ci
```

If this were an experiment, we'd conclude that average treatment effect is 1.417, such that batters who swing at the first pitch have a significantly higher decision score on the following pitch compared to those who don't swing at the first pitch (95% confidence interval: [1.244, 1.589]). However, this is a naive answer to our causal question, since it doesn't take into account any possible confounders in our data. Since it is certainly reasonable to assume that there may be many factors that are correlated to both the likelihood of swinging and a player's decision score, this motivates the need to apply various causal inference methods since this is an observational study after all. Before applying these methods, we will first check this assumption of potential confounders using various covariate balance assessments.

Covariate Balance Assessments

```
mosaicplot(table(swing$IsInZone_1, swing$IsSwing_1),
           main = "Treatment vs Pitch in Zone",
           xlab = "IsInZone_1", ylab = "IsSwing_1",
           col = c("darkorange", "black"))
```



From inital EDA of just one variable, `IsInZone_1`, we observe that there are there are indeed differences in distribution of pitches between treatment and control groups. More specifically, we see that batters are much less likely to swing at pitches out of the zone (`IsInZone_1` = 0) and more likely to swing at first pitches thrown in the zone (`IsInZone_1` = 1). The more comprehensive Table 1 shown below, along with a Love Plot that further emphasizes that there are covariate differences between those who swung at the first pitch versus those who did not.

```
#Quantitative covariates

quantX.C <- subset(swing, IsSwing_1 == 0,
                   select = c(Velocity_1, SeasonPitches, SeasonSwingPct,
                              SeasonDecisionScore, SeasonSwingProbability,
                              SeasonDecisionRuns, WinContrib))

quantX.T <- subset(swing, IsSwing_1 == 1,
                   select = c(Velocity_1, SeasonPitches, SeasonSwingPct,
                              SeasonDecisionScore, SeasonSwingProbability,
                              SeasonDecisionRuns, WinContrib))

quantX.C_bar <- apply(quantX.C, MARGIN = 2, FUN = mean) #mean for control group
```

```r
quantX.T_bar <- apply(quantX.T, MARGIN = 2, FUN = mean) #mean for treatment group
quantX.C_sd <- apply(quantX.C, MARGIN = 2, FUN = sd) #sd for control group
quantX.T_sd <- apply(quantX.T, MARGIN = 2, FUN = sd) #sd for treatment group
```

```r
#Binary covariates

binaryX.C <- subset(swing, IsSwing_1 == 0,
                    select = IsInZone_1)

binaryX.T <- subset(swing, IsSwing_1 == 1,
                    select = IsInZone_1)

C_num <- apply(binaryX.C, MARGIN = 2,
                          FUN = sum) #number for control group
C_percent <- C_num/nrow(binaryX.C) #percent for control group

T_num <- apply(binaryX.T, MARGIN = 2,
                          FUN = sum) #number for treatment group
T_percent <- T_num/nrow(binaryX.T) #percent for treatment group
```

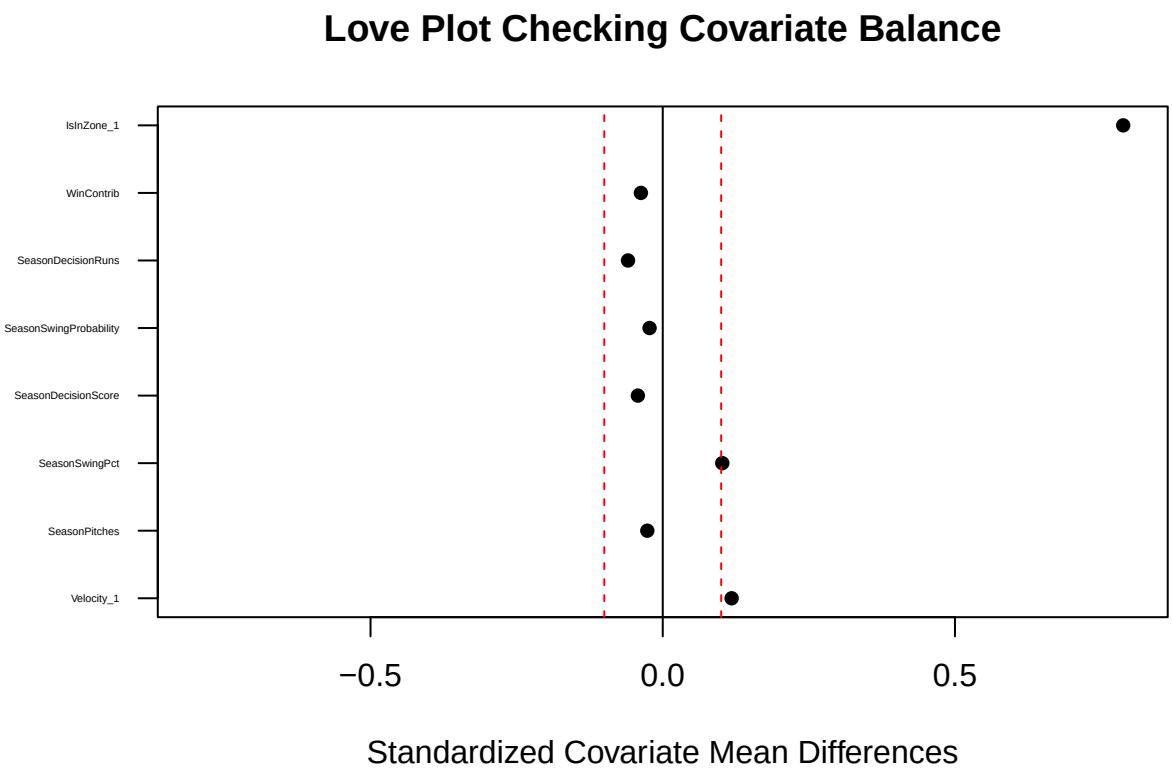
```r
#Table 1

quant_bar <- cbind(quantX.T_bar, quantX.C_bar)
quant_sd <- cbind(quantX.T_sd, quantX.C_sd)
binary_num <- cbind(T_num, C_num)
binary_percent <- cbind(T_percent, C_percent)

table <- rbind(round(quant_bar,2),
               round(quant_sd,2),
               round(binary_num,2),
               round(binary_percent,2))
colnames(table)[1:2] = c("Treatment", "Control")
rownames(table)[1:16] = c("1st Pitch Velocity - Mean",
                          "SeasonPitches - Mean",
                          "SeasonSwingPct - Mean",
                          "SeasonDecisionScore - Mean",
                          "SeasonSwingProbability - Mean",
                          "SeasonDecisionRuns - Mean",
                          "WinContrib - Mean",
                          "1st Pitch Velocity - Std Dev",
                          "SeasonPitches - Std Dev",
                          "SeasonSwingPct - Std Dev",
                          "SeasonDecisionScore - Std Dev",
                          "SeasonSwingProbability - Std Dev",
                          "SeasonDecisionRuns - Std Dev",
                          "WinContrib - Std Dev",
                          "Num of Pitches in the Zone",
                          "% of Pitches in the Zone")

table1 <- table[c(7, 14, 4, 11, 2, 9, 3, 10, 5, 12, 6, 13, 1, 8, 15, 16),]

knitr::kable(data.frame(table1),
             caption = "Covariates Associated with 1st Pitch")
```

Table 1: Covariates Associated with 1st Pitch

	Treatment	Control
WinContrib - Mean	0.55	0.57
WinContrib - Std Dev	0.50	0.49
SeasonDecisionScore - Mean	58.23	58.37
SeasonDecisionScore - Std Dev	3.46	3.38
SeasonPitches - Mean	108.63	109.76
SeasonPitches - Std Dev	43.13	42.83
SeasonSwingPct - Mean	0.47	0.46
SeasonSwingPct - Std Dev	0.07	0.07
SeasonSwingProbability - Mean	0.47	0.47
SeasonSwingProbability - Std Dev	0.03	0.03
SeasonDecisionRuns - Mean	0.00	0.00
SeasonDecisionRuns - Std Dev	0.01	0.01
1st Pitch Velocity - Mean	89.36	88.64
1st Pitch Velocity - Std Dev	5.96	6.21
Num of Pitches in the Zone	96534.00	84533.00
% of Pitches in the Zone	0.68	0.31

```
getStandMeanDiffs = function(X, treatment){
  t_indices = which(treatment == 1)
  c_indices = which(treatment == 0)
  X_t = X[t_indices,]
  X_c = X[c_indices,]
  X.bar = apply(X_t, MARGIN = 2, FUN = mean) -
          apply(X_c, MARGIN = 2, FUN = mean)
  X.var = apply(X_t, MARGIN = 2, FUN = var) +
          apply(X_c, MARGIN = 2, FUN = var)
  standCovMeanDiff = X.bar/(sqrt((X.var)/2))
  return(standCovMeanDiff)
}

#Covariate matrix
X <- subset(swing, select = c(Velocity_1, SeasonPitches, SeasonSwingPct,
                              SeasonDecisionScore, SeasonSwingProbability,
                              SeasonDecisionRuns, WinContrib,
                              IsInZone_1))
StandMeanDiffs <- getStandMeanDiffs(X, swing[,9])
```

```
plot(StandMeanDiffs, 1:8,
    xlab = "Standardized Covariate Mean Differences",
    ylab = "", yaxt = "n",
    pch = 16,
    xlim = c(-0.8, 0.8),
    main = "Love Plot Checking Covariate Balance")

axis(side=2, at=1:8, labels = names(StandMeanDiffs),
     las = 1, cex.axis = 0.32)

abline(v = 0, gcol = "gray")
abline(v = 0.1, col = "red", lty = 2)
abline(v = -0.1, col = "red", lty = 2)
```



According to the Love plot and the 0.1 rule-of-thumb, `IsInZone_1` clearly exhibits concerning covariate imbalance, as the batters in the treatment group were more likely, on average, to see first pitches in the strike zone. Additionally, `SeasonSwingPct` and `Velocity_1` also seems imbalanced as the batters in the treatment group appear to swing at pitches at a higher rate throughout the season and are receiving slightly higher velocity first pitches than the control group, on average.

Methods & Results

To get the causal effect of swinging at the first pitch, where swing vs no swing acts as the "treatment", we applied several causal inference methods and will discuss each of their conclusions, advantages, and drawbacks below. These included the following: Propensity score matching, coarsened exact matching (CEM), cardinality matching, and using the inverse propensity score weighting (IPW) estimator. As previously mentioned, in order to implement many of these methods, we focused on a random sample of 10% of the data, such that we will work smaller dataset of 41,012 observations. This dataset has 26,835 control subject and 14,177 treated subjects and will be referenced as the full or original dataset from here on out.

Propensity Score Matching

Motivated by unconfoundedness, where we are assuming that the treatment is independent of the potential outcomes given the covariates, we first implemented propensity score matching where we are pairing treatment and control subjects that have similar estimated propensity scores (probability of being treated conditioned on the X's). In other words, we matched each treated subject with a "similar" control subject using nearest-neighbors matching ("greedy algorithm"), where at each stage, the algorithm is making the locally optimal choice to minimize the matched distances between treated and control subjects. Since there are almost twice as many control subjects than treated, we matched 2 control subjects to 1 treatment subject — namely 1:2 propensity matching — for computational efficiency. This is an effort to try to obtain an "effectively randomized" matched dataset that exhibits covariate balance.

```
library(MatchIt)
matchedgreed.12 = match.data(matchit(formula = IsSwing_1 ~
                      Velocity_1 +SeasonPitches+SeasonSwingPct+
                           SeasonDecisionScore+
                        SeasonSwingProbability+
                           SeasonDecisionRuns+WinContrib+
                           IsInZone_1, data = swing.1,
                     method = "nearest", ratio = 2,
                     distance = "glm", link = "logit"))
```

After implementing 1:2 propensity score matching, we used calipers as a "tweak" to prevent "bad matches". Setting the caliper to 0.1, this removed any matches that have a larger distance than the indicated cutoff of 0.1, resulting in a slightly smaller matched dataset of 17558 control subjects and 13009 treated subjects. Figure 1 below illustrates the difference in covariate balance between the dataset before matching and the one after propensity score matching with calipers.

```
#Caliper
matched.caliper.12 = match.data(matchit(formula = IsSwing_1 ~
                      Velocity_1 +SeasonPitches+SeasonSwingPct+
                           SeasonDecisionScore+
                        SeasonSwingProbability+
                           SeasonDecisionRuns+WinContrib+
                           IsInZone_1, data = swing.1,
                     method = "nearest", ratio = 2,
                     caliper = 0.1, std.caliper = FALSE))
#table(matched.caliper.12$IsSwing_1)
```

```
#Check covariate balance
X.caliper = subset(matched.caliper.12,
                   select = c(Velocity_1, SeasonPitches, SeasonSwingPct,
                           SeasonDecisionScore, SeasonSwingProbability,
                           SeasonDecisionRuns, WinContrib,
                           IsInZone_1))
#comparison Love plots
library(randChecks)
X.sub = subset(swing.1,
                   select = c(Velocity_1, SeasonPitches, SeasonSwingPct,
                           SeasonDecisionScore, SeasonSwingProbability,
                           SeasonDecisionRuns, WinContrib,
                           IsInZone_1))

lovePlotCompare(X1 = X.sub, indicator1 = swing.1$IsSwing_1,
X2 = X.caliper, indicator2 = matched.caliper.12$IsSwing_1,
dataNames = c("No Matching", "1:2 Matching with Caliper"))
```

Looking at the Love plot (Figure 1), we observe that the matched dataset does not convincingly have better covariate balance because the majority of the standardized covariate mean differences for the matched dataset (red triangles) are very close to the standardized covariate mean differences from the full dataset (black circles). Additionally, despite seeing a substantial improvement in covariate balance for the variable `IsInZone_1`, this matched dataset (even after applying calipers) still does not yield overall covariate balance. Thus, this motivates utilizing other techniques that ensure covariate balance, such as coarsened exact matching and cardinality matching.

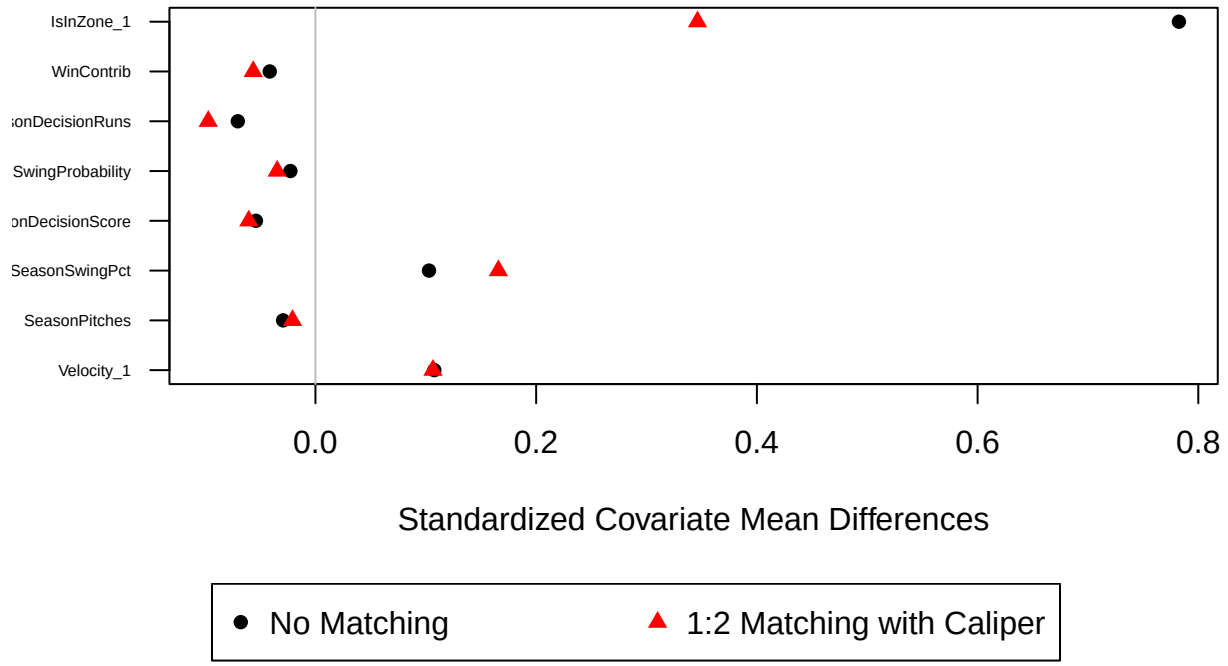


Figure 1: Comparing covariate balance between the original and matched datasest.

Coarsened Exact Matching & Cardinality Matching

Instead of matching subjects using their calculated propensity scores or using any other modeling techniques, we obtained balance directly from the actual covariates using coarsened exact matching (CEM) and cardinality matching. To implement CEM, we coarsened the 8 covariates of interest (`Velocity_1`, `SeasonPitches`, `SeasonSwingPct`, `SeasonDecisionScore`, `SeasonSwingProbability`, `SeasonDecisionRuns`, `WinContrib`, `IsInZone_1`) into blocks and then matched treatment and control subjects in the same block. After doing so, the causal effect estimation was calculated and the results are shown below:

```
att(cemObject, DecisionScore_2 ~ IsSwing_1,
    data = swing.cem)
```

```
##
##                G0    G1
## All         26835 14177
## Matched     23507 13424
## Unmatched    3328   753
##
## Linear regression model on CEM matched data:
##
## SATT point estimate: 1.103134 (p.value=0.000106)
## 95% conf. interval: [0.545543, 1.660725]
```

We recall that, in the full dataset, there are 14,177 and 26,835 treated and control subjects respectively. Meanwhile, in this CEM dataset, there are only 13,424 treated subjects and 23,507 control subjects. However, the tradeoff of having a slightly smaller sample is that there no need for balance checking since we are exactly matching on the coarsened dataset. Nevertheless, we can compare covariate balance using CEM weighting to the original data, shown in Figure 2. We see that the majority of the covariates are better-balanced after CEM, as they are closer to 0. However, despite the improvement, `IsInZone_1` is still exhibits imbalance.

```
getWeightedSCMDs = function(X, indicator, weights){
var.treatment = apply(X[indicator == 1,], MARGIN = 2, FUN = var)
var.control = apply(X[indicator == 0,], MARGIN = 2, FUN = var)
var.pooled = (var.treatment + var.control)/2
K = ncol(X)
#then, the vector of (weighted) SCMDs is:
covMeanDiffs = vector(length = K)
for(k in 1:K){
treatedMean = sum( X[,k]*indicator*weights )/sum( indicator*weights )
controlMean = sum( X[,k]*(1-indicator)*weights)/sum( (1-indicator)*weights )
covMeanDiffs[k] = (treatedMean - controlMean)/sqrt(var.pooled[k])
}
return(covMeanDiffs)
}

X.CEM = subset(data.cem,
                   select = c(Velocity_1, SeasonPitches, SeasonSwingPct,
                              SeasonDecisionScore, SeasonSwingProbability,
                              SeasonDecisionRuns, WinContrib,
                              IsInZone_1))

covMeanDiffs.cem = getWeightedSCMDs(X = X.CEM, indicator = data.cem$IsSwing_1, weights = cemObject$w)

covMeanDiffs = getStandardizedCovMeanDiffs(X.sub,
                                           swing.1$IsSwing_1)

plot(covMeanDiffs, 1:8,
xlab = "Standardized Covariate Mean Differences",
ylab = "", main = "", yaxt = "n",
pch = 16)
points(covMeanDiffs.cem, 1:8, col = "orange", pch = 17)
abline(v = 0, col = "gray")
axis(side=2, at=1:8,
     labels = names(covMeanDiffs),
     las = 1, cex.axis = 0.33)
legend("right",
legend = c("Original Data", "Data with CEM Weights"),
col = c("black", "orange"),
pch = c(16, 17))

abline(v = 0.1, col = "red", lty = 2)
abline(v = -0.1, col = "red", lty = 2)
```

In terms of our results from the output, we see that the p-value (0.0001) certainly less than 0.05, so we reject the null hypothesis that the average treatment effect for the treated (MATT) within the matched dataset is equal to 0. The MATT is estimated to be 1.103 with a 95% confidence interval between 0.545543 and 1.660725. Since the confidence interval does not contain 0, this is another indication of a significant treatment effect — but only for the treated subjects in this particular smaller dataset!

Another automatic optimization method is cardinality matching, where it finds the largest dataset that satisfies covariate balance constraints (e.g., all standardized covariate mean differences are below 0.1). Because of the size of the full dataset, we again used the random sample of the data (10%) from before. Upon applying this method, we encountered an error suggesting: "The optimization problem may be infeasible.

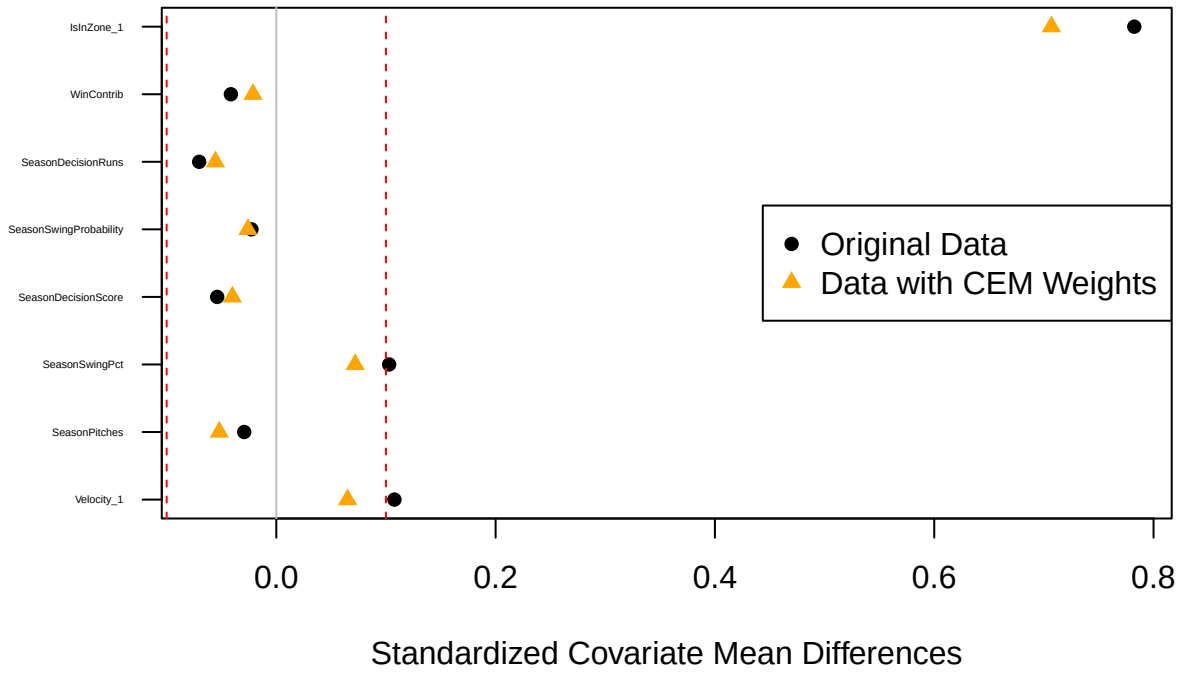


Figure 2: Checking covariate balance between unweighted and CEM weighted SCMDs.

Try increasing the value of 'tols'." This means that there was no dataset found to satisfy the tolerance level of 0.01, which was the rule-of-thumb we used as the indication of covariate balance. Since finding a subset does not seem to be feasible using this dataset, we attempted to weigh subjects such that they approximate an experiment instead, shown in the following method.

```
# cardMatch.0.2 = match.data(matchit(formula = IsSwing_1 ~
#                         Velocity_1 + SeasonSwingPct +
#                           IsInZone_1, data = swing.1,
#                       method = "cardinality", ratio = 2,
#                       tol = 0.01))
```

Inverse Propensity Score Weighting

As a common alternative to matching, inverse propensity score weighting (IPW) can be used to estimate causal effects through the use of a propensity score model. First, we used logistic regression to produce propensity score estimates for the dataset, where all of the 8 covariates are used as predictors (with no interactions). After computing the inverse propensity score weights, we considered the possibility of extreme weights that may greatly skew our estimates. We subsequently checked the effective sample size, which in this case is 30601.44, and the ratio between this effective sample size and the original sample size is 0.746, or about 3/4 of the original sample. In other words, by using IPW here, we are only reducing the sample size by roughly 25%. Now, after plugging the estimated propensity scores into the equation for the classic IPW estimator, we got the following point estimate: $\hat{\tau}_{\text{IPW1}} = 0.880$. In an effort to reduce the variance of our estimator, we then used the "normalized" IPW estimator ($\hat{\tau}_{\text{IPW2}}$), where we are running a weighted linear regression of the outcome on the treatment, where the weights correspond to the inverse propensity score weights. The summary output using this estimator is displayed below:

```
psEst = predict(glm(IsSwing_1 ~ Velocity_1 + SeasonPitches +SeasonSwingPct + SeasonDecisionScore + Seas
SeasonDecisionRuns + WinContrib + IsInZone_1, data = swing.1, family = "binomial"), type = "response")
```

```r
#summary(psEst) No extreme weights

#the estimated propensity scores just for the treatment group are:
psEst.t = psEst[swing$IsSwing_1 == 1]
#summary(psEst.t)
#the estimated propensity scores just for the control group are:
psEst.c = psEst[swing$IsSwing_1 == 0]
#summary(psEst.c)

  #T # Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   # 0.11    0.19    0.23    0.34    0.52    0.77  127634
  #C # Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   # 0.10    0.19    0.23    0.35    0.53    0.75  241471

psWeights = swing.1$IsSwing_1/psEst +
  (1-swing.1$IsSwing_1)/(1-psEst)

#Effective sample size
n.eff.ps = (sum(psWeights))^2/sum(psWeights^2)
#n.eff.ps
## [1] 30601.44
#the full sample size is
n = nrow(swing.1)
#the ratio is:
#n.eff.ps/n

#only reducing the sample by roughly 25%
```

```r
###Classic IPW estimator point estimate = 0.8798019
# mean(swing.1$IsSwing_1*swing.1$DecisionScore_2/psEst -
# (1-swing.1$IsSwing_1)*swing.1$DecisionScore_2/(1-psEst))
```

```r
#Normalized IPW estimator
linReg.w = lm(DecisionScore_2 ~ IsSwing_1,
              data = swing.1, weights = psWeights)
library(lmtest); library(sandwich)
normIPW = coeftest(linReg.w, vcov = vcovHC(linReg.w, type = "HC2"))
normIPW
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error  t value  Pr(>|t|)
## (Intercept) 58.74080    0.16391 358.3785 < 2.2e-16 ***
## IsSwing_1    0.88899    0.30707   2.8951  0.003793 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#normIPW[2] #tauhat.ipw2
confint(normIPW)[2,] #CI
```

```
##     2.5 %    97.5 %
## 0.2871299 1.4908534
```

We observe that the point estimate of $\hat{\tau}_{\text{IPW2}}$ is 0.889, with a 95% confidence interval of [0.287, 1.491]. We see that this confidence interval does not contain 0, and we have a low p-value of 0.004; thus, we reject the null hypothesis that the average treatment effect (ATE) is equal to 0. If the propensity scores were well-estimated, we know that this estimator is indeed unbiased. To ensure that this is in fact the case, we can check covariate balance via weighted standardized covariate mean differences.

```
#weighted SCMD from IPW:
covMeanDiffs.ipw = getWeightedSCMDs(X = X.sub, indicator = swing.1$IsSwing_1, weights = psWeights)

plot(covMeanDiffs, 1:8,
xlab = "Standardized Covariate Mean Differences",
ylab = "", main = "", yaxt = "n",
pch = 16)
points(covMeanDiffs.ipw, 1:8, col = "orange", pch = 17)
abline(v = 0, col = "gray")
axis(side=2, at=1:8,
     labels = names(covMeanDiffs),
     las = 1, cex.axis = 0.33)
legend("right",
legend = c("Original Data", "IPW Data"),
col = c("black", "orange"),
pch = c(16, 17))

abline(v = 0.1, col = "red", lty = 2)
abline(v = -0.1, col = "red", lty = 2)
```

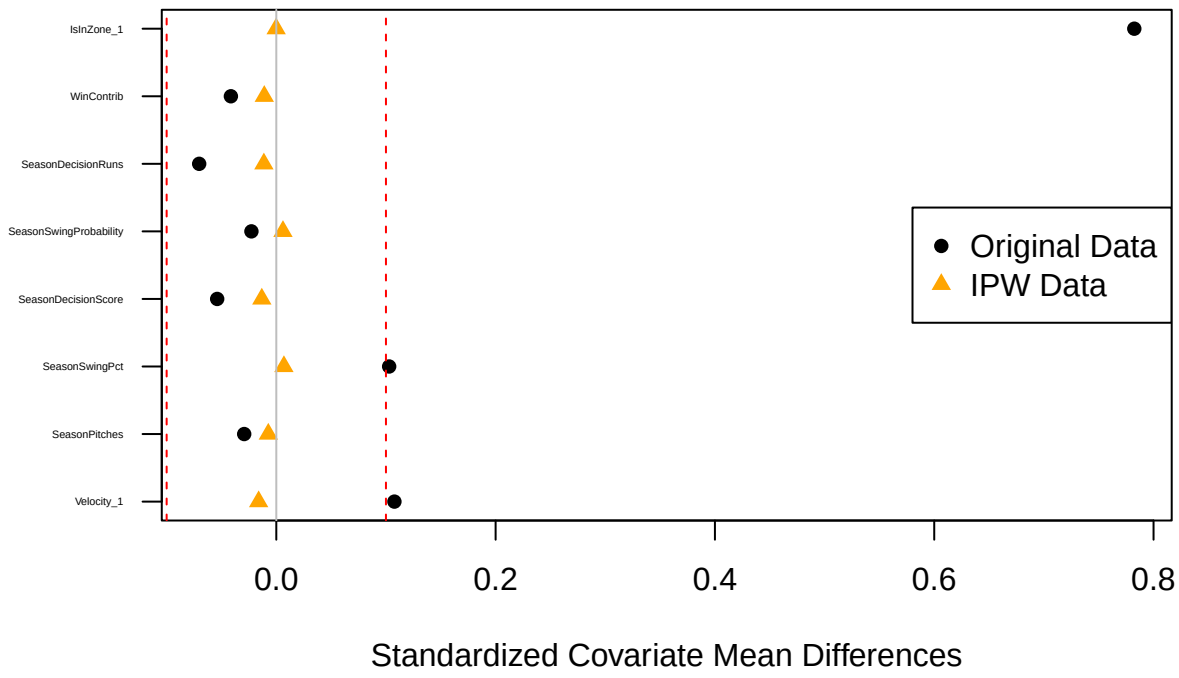


Figure 3: Checking covariate balance between unweighted and IPW weighted SCMDs

From Figure 3, we see that IPW-based covariate mean differences are centered around zero and all look to be well-balanced. Thus, this suggests that the propensity scores do exhibit the balancing property, which gives some credence to the unbiasedness of our estimator.

Interpretations & Conclusions

In the previous section, we implemented several causal inference methods, some of which were successful and some that were not. We started off with propensity score matching, where we were seeking to find a subset of the data where the treatment approximated an experiment. After implementing nearest neighbors (greedy) matching, in addition to applying calipers to throw out poor matches, we observed that the Love plot still exhibited imbalance. Since matching is only useful if it produces similar treat and control group (i.e., covariate balance), we could not analyze this data as a matched-pairs experiment and thus were motivated to try a different method instead. Working around the need for balance checking, we implemented coarsened exact matching, where the covariates of interest were automatically categorized and subjects were exactly matched. The estimated causal effect of swinging at the first pitch was significant for the treated subjects in this CEM dataset, such that swinging at the first pitch leads to higher decision scores of the following pitch on average (ATT = 1.103). On the other hand, we were not able to estimate causal effects using cardinality matching since finding a dataset under the constraint that standardized covariate mean differences were all less than 0.01 was infeasible. Realizing that the treatment effect from utilizing CEM was only for the treated subjects in the smaller matched dataset, we implemented a final method, inverse propensity score weighting (IPW) to get at the average treatment effect (ATE) for the full dataset. The estimated ATE came out to be 0.889, which similar to the results from CEM, suggests that swinging at the first pitch leads to higher decision scores on average. This ATE should unbiased if the propensity score model is well-specified, and the above balance from Figure 3 gives some credence to this, and thus we can say that this treatment effect targets the full-population ATE. Yet it should be noted that in all of our causal inference methods, including IPW, we were only using a random sample of the data, so the lack of generalizability should be considered.

Taken all together, significant positive treatment effects from our causal inference methods suggest that **swinging on the first pitch DOES lead to better decisions on the following pitch**. However, the actual improvement in decision score does seem small when compared to the range of decision scores across the players throughout the league. Since the standard deviation of all decision scores is 26.520, a roughly 1 unit increase in decision scores may seem marginal and a small effect, but is still an increase nonetheless. In conclusion, although these results should be taken with reservations (as discussed in the limitations below), players might want to be more intentional about swinging at first pitches — even if you swing and miss, evidence suggests that at least you're more likely to make a better decision on the next pitch!

Limitations & Future Steps

In this project, we started off by using matching to obtain covariate balance and thus less biasedly estimate the causal effect. However, in doing so, we are estimating the causal effects for a matched sample, instead of the entire population. Additionally, the causal estimate from CEM was only for the treated subjects in the matched dataset, furthering the issue of generalizability especially if there is treatment effect heterogeneity. Nevertheless, estimating the causal effect well for some people is better than estimating poorly for all people. In addition to this, we are also making the Unconfoundedness assumption — that there should be no unobserved confounders that correlate with the potential outcomes and the treatment, after conditioning on the covariates — and thus assuming that matching approximates a randomized experiment. However, Unconfoundedness does not seem plausible given this dataset because many of the key variables in judging the quality of batters are left out (i.e., batting average, on-base percentage, etc.), some of which may definitely be a potential confounder. Future research could investigate this by conducting sensitivity analyses, which places bounds on how much Unconfoundedness can be violated and discovers to what extent can Unconfoundedness be violated without changing our results. As we discussed earlier, one of the main concerns regarding our results was that in order to implement many of these methods, we focused on a random sample of only 10% of the data. Therefore, possible next steps could include applying these methods on a larger dataset, and especially one that includes additional player statistics that better contextualizes a players at-bat — doing so would greatly help us unbiasedly estimate the causal effect.