POL 346 Final Project

An Analysis of Pitcher and Hitter Strategy

Jonah May

May 6, 2016

Introduction

The goal of this project is to provide three primary types of analysis designed to help Major League Baseball (MLB) hitters as they approach an at-bat. Each MLB at-bat is a strategic game between hitter and pitcher. For this analysis, I want to provide insight into being able to predict the properties of the coming pitch. Oftentimes, pitchers will intentionally throw pitches out of strike zone, trying to make hitters chase bad pitches that are difficult to hit. First, with this project I hope to illuminate how different factors can help predict if a pitcher will throw in the zone or not. Secondly, it would be extremely helpful to hitters to know how different factors can be used to predict the type of pitch that a pitcher is going to throw.

My third type of analysis involves predicting in what contexts an umpire is likely to make bad calls. Unlike my first two analyses, the third does not involve hitter-pitcher strategy, but it is still an important consideration for hitters.

Data Source/Data Processing

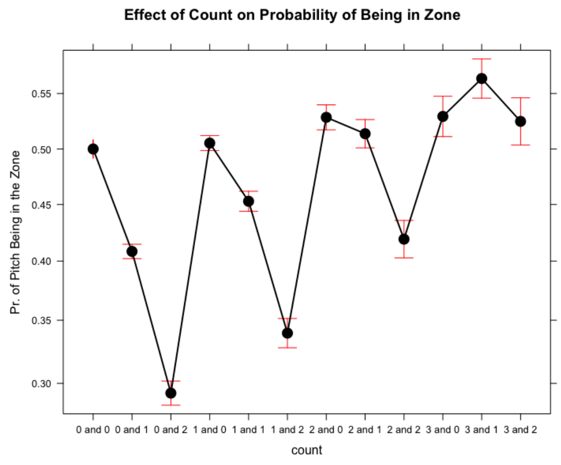
For all three of these analyses, my data source is all pitch data from the 2014 MLB season. This data was collected using MLB's pitchFx system. This collects data on the game context of the pitch, as well as the physical descriptors of the pitch such as speed, break distance, and type of pitch.

For this project, I focused on using predictors that hitters would have access to in the moments before the pitch. As a result, I avoided using the physical descriptors of the coming pitch and instead focused on the game context. This includes things like number of outs, if there are runners on base etc. While I couldn't use data on the upcoming pitch, the batter would certainly have knowledge of previous pitches. Therefore, I chose to lag pitch descriptors. This is important because we should not be considering each pitch observation as a purely independent event. Pitchers strategically string pitches together to maximize their effectiveness. For the lagged variable, I focused primarily on the type of pitch that came before. I also added the pitch number within a given at-bat.

In Zone Predicton

In order to do this analysis, I first created a binary column of whether or not coordinates of the pitch were in the strike zone. My data set already contained the lower and upper bounds of the strike zone. I was able to derive the lateral boundaries by knowing the width of a regulation plate.

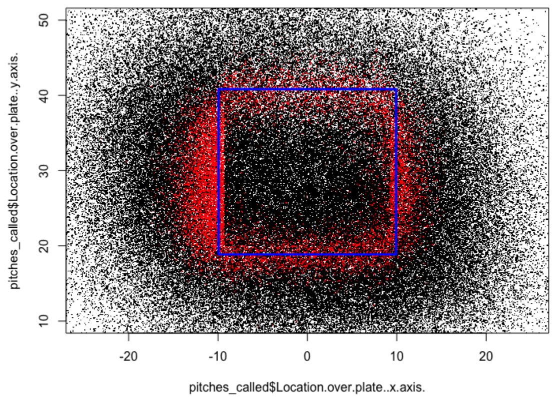
Next, I constructed a logit binomial model with the predictors. At first, I created a model with the count ,number of pitches in the At-Bat, Inning, Outs, Men on Base and the type of the last pitch. However, this yielded some results that were not significant. I reasonsed that this was because there is colinearity between the Count and the previous pitch. There are certain pitches that are more likely depending on the count. As a result, I removed the previous pitch as a predictor because the count is predictive of whether the pitch is in the zone, and the type of the previous pitch. I also removed the status of the men on base because it was insignificant. The results are below.

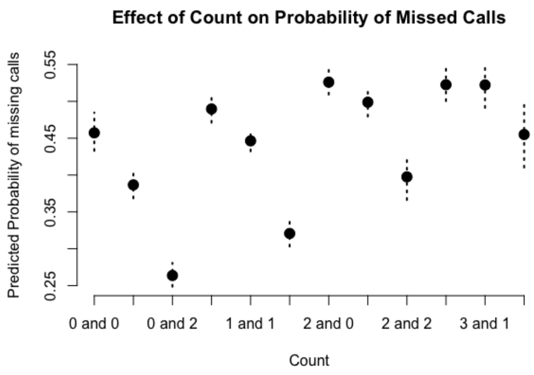


These results demonstrate a few patterns in pitcher strategy and tendencies. First, we can see all of the valleys occur when the hitter has two strikes. These valleys are when the predicted probability of the pitcher throwing in the zone is the lowest. This is because when the pitcher has the upper hand, he feels more comfortable taking risks and pitching outside the zone. We can also see the opposing trend occuring. As pitchers throw more balls, they are more likely to throw into the zone.

Missed Call Prediction

To determine if an umpire missed a call, I first subset the data to pitches where umpires had to make ball/strike decisions Then, I created a binary variable that determines whether they made the call correctly or not. Interestingly, when umpires made a call, it was incorrect 12.46% of the time. The goal for this model is to predict when umpires are most likely to make the wrong call. I have included a plot of the strike zone to show the locations where bad calls often take place. The blue box is the strike zone. Black dots are the locations of correct calls. Red dots are the locations of bad calls. Not suprisingly, most of missed calls land right on the border of of the strikezone.





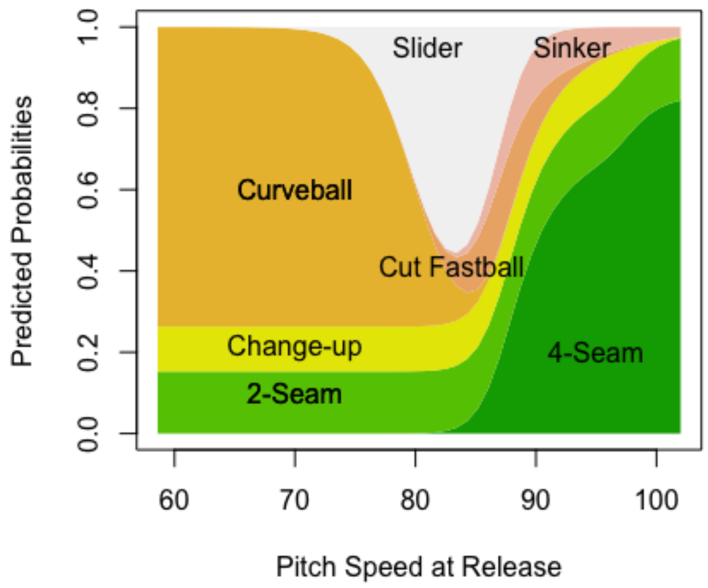
Using Monte Carlo techniques, I plotted the predicted probability of an umpire missing a call given the count. Interestingly, they have a pattern that matches the predicted probability of the Ball being in the zone from the previous section. As the probability that the ball is going to be in the zone increases, so does the probability that the call will be missed. I hypothesize that this is due to the fact that umpires are expecting the pitch to be in the zone (which is also shown by analysis in the first section), so they are more likely to make a call that confirms their expectations.

Pitch Type Prediction

For this analysis, I created a multinomial model designed to predict the upcoming pitch based on clues that a hitter can gather in the moments before swinging. For the pitch outcome, I removed infrequent pitch types such as Knuckleball or Screwball. The predictors included game context information such as: Inning, Outs and men on base. Addionally the model included the type of the last pitch. For the previous pitch type, I grouped them as either fastball, breaking ball, or off-speed. Finally, I included the speed of the pitch out of the pitchers hand. I included this because it is plausible to believe that a hitter could quickly interpret the speed of the pitch out of the pitchers hand. Each of the predictors were significant.

To determine if information on the previous pitch was important or not, I performed a liklihood ratio test against a model without infomration on the previous pitch. The test allowed us to reject the null hypothesis that the model without the information about the last pitch is just as predictive as the model with it. The p-value on the liklihood ratio test was 2.2E-16.

I created a stacked probability plot of pitch type as a function of initial pitch speed.



The plot doesn't show much variation from 60-75 MPH speed. This is largely due to the fact that very few pitches happen at that speed. As speed increases, we can see the curve ball becomes much less probable, and fastballs become much more likely. Also, in the high 80s it can be very difficult to predict the exact pitch type because there are a lot of pitches with equal probabilities in that range.

Future Considerations

For this project, a main focus was to select predictors that would be available to a hitter before the pitch, or just after the pitch is released. With that in mind, in future analyses, I would like to pull in hitter statistics to see how perceived skill of the hitter impacts a pitcher's approach to that hitter.

```
#READ IN PITCH DATA
pitches <-
read.csv(file="pitches.csv",head=TRUE,sep=",",na.string=c("NULL","(Other)"))
#Removed this column because it was empty
pitches$Pitcher.height <- NULL
#read lagged data
last.pitches <- read.csv(file="lastpitch.csv",head=TRUE,sep=",")
pitches <- cbind(pitches,last.pitches)
pitches <- subset(pitches, Count != "")
pitches$Count <- factor(pitches$Count)

##Remove uncommon pitches
pitches <- subset(pitches, Type.of.pitch..estimated.by.algo.== "2-Seam
Fastball"|
                    Type.of.pitch..estimated.by.algo.== "4-Seam Fastball"|
                    Type.of.pitch..estimated.by.algo.== "Change-up"|
                    Type.of.pitch..estimated.by.algo.== "Curveball"|
                    Type.of.pitch..estimated.by.algo.== "Cut Fastball"|
                    Type.of.pitch..estimated.by.algo.== "Sinker"|
                    Type.of.pitch..estimated.by.algo.== "Slider")

pitches <- subset(pitches, last.pitch== "2-Seam Fastball"|
                    last.pitch== "4-Seam Fastball"|
                    last.pitch== "Change-up"|
                    last.pitch== "Curveball"|
                    last.pitch== "Cut Fastball"|
                    last.pitch== "Sinker"|
                    last.pitch== "Slider")

pitches$Type.of.pitch..estimated.by.algo. <-
factor(pitches$Type.of.pitch..estimated.by.algo.)
pitches$last.pitch <- factor(pitches$last.pitch)

###determine if in zone
pitches$inzone<- ifelse(pitches$Location.over.plate..x.axis. > -9.3 &
                        pitches$Location.over.plate..x.axis. < 9.3 &
                        pitches$Location.over.plate..y.axis. >
pitches$Strike.zone..bottom. &
                        pitches$Location.over.plate..y.axis. <
pitches$Strike.zone..top.,
                        1,
                        0)

###create pitch categories
pitches$type[pitches$Type.of.pitch..estimated.by.algo. == "2-Seam Fastball"
             |pitches$Type.of.pitch..estimated.by.algo. == "4-Seam Fastball"]
<- "Fastball"
```

```
pitches$type[pitches$Type.of.pitch..estimated.by.algo. == "Change-up"] <-
"Off Speed"

pitches$type[pitches$Type.of.pitch..estimated.by.algo. == "Curveball"
             |pitches$Type.of.pitch..estimated.by.algo. == "Cut Fastball"
             |pitches$Type.of.pitch..estimated.by.algo. == "Knuckle Curve"
             |pitches$Type.of.pitch..estimated.by.algo. == "Sinker"
             |pitches$Type.of.pitch..estimated.by.algo. == "Slider"
             |pitches$Type.of.pitch..estimated.by.algo. == "Split-Finger
Fastball"
             |pitches$Type.of.pitch..estimated.by.algo. == "4-Seam Fastball"]
<- "Breaking Ball"
pitches$last.type[pitches$last.pitch == "2-Seam Fastball"
             |pitches$last.pitch == "4-Seam Fastball"] <- "Fastball"
pitches$last.type[pitches$last.pitch == "Change-up"] <- "Off Speed"

pitches$last.type[pitches$last.pitch == "Curveball"
             |pitches$last.pitch == "Cut Fastball"
             |pitches$last.pitch == "Knuckle Curve"
             |pitches$last.pitch == "Sinker"
             |pitches$last.pitch == "Slider"
             |pitches$last.pitch == "Split-Finger Fastball"
             |pitches$last.pitch == "4-Seam Fastball"] <- "Breaking Ball"

###subset of pitches that umpires had to make calls on
pitches_called <- subset(pitches,Pitch.result=="Ball" | Pitch.result=="Called
Strike")
#make column for missed calls
pitches_called$missed <-ifelse((pitches_called$inzone == 1 &
pitches_called$Pitch.result=="Ball") |
                                 (pitches_called$inzone == 0 &
pitches_called$Pitch.result=="Called Strike"),1,0)

###remove NAs
pitches <- pitches[complete.cases(pitches),]
###take a sample
pitch_samples <- pitches[sample(nrow(pitches), 10000),]
pitch_samples$Type.of.pitch..estimated.by.algo. <-
factor(pitch_samples$Type.of.pitch..estimated.by.algo.)


###
### set baselines
pitch_samples$Count <- relevel(pitch_samples$Count,"0 and 0")
pitch_samples$last.pitch <- relevel(pitch_samples$last.pitch,"4-Seam
Fastball")
pitch_samples$Type.of.pitch..estimated.by.algo. <-
relevel(pitch_samples$Type.of.pitch..estimated.by.algo.,"4-Seam Fastball")
```

```
pitch_samples$Men.on.base <- relevel(pitch_samples$Men.on.base,"No men on")

pitches_called$Count <- relevel(pitches_called$Count,"0 and 0")
pitches_called$last.pitch <- relevel(pitches_called$last.pitch,"4-Seam
Fastball")
pitches_called$Men.on.base <- relevel(pitches_called$Men.on.base,"No men on")
pitches$Count <- relevel(pitches$Count,"0 and 0")
pitches$last.pitch <- relevel(pitches$last.pitch,"4-Seam Fastball")
pitches$Type.of.pitch..estimated.by.algo. <-
relevel(pitches$Type.of.pitch..estimated.by.algo.,"4-Seam Fastball")
pitches$Men.on.base <- relevel(pitches$Men.on.base,"No men on")


##### pitch velocity vs. speed
####not used in report
#### shows inverse relationship
plot(pitches$Pitch.count,pitches$Pitch.end.speed..mph.)
#####

####



###Predicting If In Zone
library(pscl)
library(MASS)
library(effects)
##create model
inzone_model_test <- glm(inzone~Count+
                           at.bat.pitch.count+
                           Inning+
                           Outs+
                           Men.on.base+
                           last.pitch
                ,data=pitches
                ,family=binomial)
#remove insignifancies
inzone_model_refined <- glm(inzone~Count+at.bat.pitch.count+Inning+Outs
                         ,data=pitches
                         ,family=binomial)


stargazer(inzone_model_refined, type = "latex")
###Plot the effects of count on inzone
typical_val <- function(var){
  if(length(unique(var)==2)){
    0 #We made the modal category the reference!
  }else{
    median(var,na.rm=TRUE) #Could be mean
  }
}

plot(effect("Count",inzone_model_refined
             ,confidence.level=0.95
             ,typical=typical_val)
     ,ylab="Pr. of Pitch Being in the Zone"
     ,xlab="count"
     ,main="Effect of Count on Probability of Being in Zone"
)


###
### Predicting if the call is missed
###plot strike zone
plot(pitches_called$Location.over.plate..x.axis.
     ,pitches_called$Location.over.plate..y.axis.
     ,pch="."
     ,col=as.factor(pitches_called$missed)
     ,xlim=c(-25,25)
     ,ylim=c(10,50))
rect(-9.93, 18.86, 9.93, 40.83,
     border = "blue", lwd=3)

###make model for missed calls
missed_model <- glm(missed~Count+at.bat.pitch.count+Inning+Outs+Men.on.base
                    ,data=pitches_called
                    ,family=binomial)

summary(missed_model)

#Make plot using monte carlo simulation
new_data <-missed_model$model
pitchcount <- sort(unique(pitches$Count))
data_copies <- lapply(pitchcount
                      ,FUN=function(x) #x will be each value in verbal_scores
                      {
                        new_data$Count <- x
                        data_copies<-
model.matrix(missed_model$formula,new_data)[,-1]
                      })
set.seed(1234)
library(MASS)
coef_samples <- mvrnorm(500
                        ,mu=coef(missed_model)[2:22]
                        ,Sigma=vcov(missed_model)[2:22,2:22]
)
```

```r
prob <- lapply(data_copies
               ,FUN=function(x)
               {
                 print(dim(x))
                 print(dim(coef_samples))
                 linear_pred <- x %*% t(coef_samples)
                 1 / (1 + exp(linear_pred * -1))
               }
)

avg_prob <- lapply(prob,colMeans)

predicted_ci <- sapply(avg_prob #sapply returns array, rather than list
                       ,FUN=quantile
                       ,probs=c(0.025,0.5,0.975)
)


predicted_ci <- t(predicted_ci)

plot(predicted_ci[,2]~c(1:12)
     ,cex=1.5
     ,pch=19
     ,ylim=c(min(predicted_ci),max(predicted_ci))
     ,main="Effect of Count on Probability of Missed Calls"
     ,xlab="Count"
     ,ylab="Predicted Probability of missing calls"
     ,axes=FALSE)
axis(1,at=1:12,labels=c("0 and 0",
                        "0 and 1",
                        "0 and 2",
                        "1 and 0",
                        "1 and 1",
                        "1 and 2",
                        "2 and 0",
                        "2 and 1",
                        "2 and 2",
                        "3 and 0",
                        "3 and 1",
                        "3 and 2"
                        ))
axis(2)
abline(h=0,lty=2)
for(i in 1:12){
  lines(c(i,i),c(predicted_ci[i,1],predicted_ci[i,3])
        ,lty=3
        ,lwd=2)
```

```
}
###


### predicting pitch type
library(nnet)
library(stargazer)

#use samples for time efficiency
pitch_samples <-
pitch_samples[!is.na(pitch_samples$Pitch.start.speed..mph.),]
#make model
pitch_model_last <-
multinom(Type.of.pitch..estimated.by.algo.~Pitch.start.speed..mph.+at.bat.pit
ch.count+last.type+Inning+Outs+Men.on.base
                         ,Hess=TRUE
                         ,model=TRUE
                         ,data=pitch_samples,maxit=500)
#make model without last pitch info
pitch_model_no_last <-
multinom(Type.of.pitch..estimated.by.algo.~Pitch.start.speed..mph.+Inning+Out
s+Men.on.base
                          ,Hess=TRUE
                          ,model=TRUE
                          ,data=pitch_samples,maxit=500)
library(lmtest)
#see if last pitch info matters or not
lrtest(pitch_model_no_last,pitch_model_last) #Why 5 degrees of freedom?


stargazer(pitch_model_last
          ,type="text" #Can create latex table,
          #which RMarkdown can render.
          ,star.char = c("","","*")
          ,star.cutoffs =c(NA,NA,.05)
)
###Stacked plot of pitch speed and probability of pitch type
sim_speed <- seq(min(pitch_samples$Pitch.start.speed..mph.)
               ,max(pitch_samples$Pitch.start.speed..mph.)
               ,length.out=50)


new_data <- pitch_model_last$model


#Get linear predictors
lin_preds <- lapply(sim_speed,
                    function(x){
                      new_data$Pitch.start.speed..mph. <- x
                      new_X <- model.matrix(pitch_model,new_data)
                      new_X %*% t(coef(pitch_model_last)) #What does this
produce?
                    })


#Get predicted probabilities
pred_probs <-  lapply(lin_preds,
                      function(x){
                        numerator <- exp(x)
                        sum_accross_cats <- rowSums(numerator)
                        denominator <- 1 + sum_accross_cats
                        #Divide each column in numerator by common
denominator.
                        probs <- apply(numerator,2
                                       ,function(y){
                                         y/denominator
                                       }
                        )
                        cbind(1-rowSums(probs),probs) #Add prob. of baseline
                      })

#Get average across observed values
avg_probs <- sapply(pred_probs,colMeans)
avg_probs <- t(avg_probs)

#Plot "stacked" probabilities
cum_probs <- apply(avg_probs,1,cumsum)
cum_probs <- t(cum_probs)

#Four shades of gray for plot
plot_col<- terrain.colors(7)
print(cum_probs)#Plot:
plot(cum_probs[,1]~sim_speed
     ,type="n" #Empty plot
     ,ylim=c(0,1)
     ,ylab="Predicted Probabilities"
     ,xlab="Pitch Speed at Release"
)
polygon(c(sim_speed,rev(sim_speed))
        , c(cum_probs[,1],rep(0,50))
        ,border=NA
        ,col=plot_col[1])
for(i in 2:7){
  polygon(c(sim_speed,rev(sim_speed))
          , c(cum_probs[,i],rev(cum_probs[,i-1]))
          ,border=NA
          ,col=plot_col[i])
}


text(95,0.2,"4-Seam",col="black")
text(70,0.1,"2-Seam",col="black")
text(70,0.1,"2-Seam",col="black")
text(70,0.21,"Change-up",col="black")
text(70,0.6,"Curveball",col="black")
text(70,0.6,"Curveball",col="black")
text(83,0.41,"Cut Fastball",col="black")
text(93,.95,"Sinker",col="black")
text(81,.95,"Slider",col="black")

###
```