Predicting Pitcher Injury

4/16/2017

Roger Chow (r6chow)

Table of Contents

[Introduction 2](#_Toc480129790)

[Literature Review 3](#_Toc480129791)

[Dataset 5](#_Toc480129792)

[Approach 7](#_Toc480129793)

[Step 1: Scrape PitchFX data from MLB website and Disabled List 7](#_Toc480129794)

[Step 2: Load Data from Database to R and Impute 8](#_Toc480129795)

[Step3: Label the Data 9](#_Toc480129796)

[Step 4: Transform Count Variables 9](#_Toc480129797)

[Step 5: Exploratory Analysis and Outlier Detection 10](#_Toc480129798)

[Step 6: Multivariate Logistic Regression and Testing 11](#_Toc480129799)

[Step 7: Predctions for 2017 12](#_Toc480129800)

[Results 13](#_Toc480129801)

[Conclusions 17](#_Toc480129802)

[Appendix 1 (Markdown) 18](#_Toc480129803)

# Introduction

Injuries in Major League Baseball are costly and on the rise, particularly injuries among pitchers. In 2015, injured players placed on the disabled list cost $700 million in player salaries. Pitchers accounted for more than 50% of the disabled list. A pitcher on the disabled list may cut a team’s odds of winning the season. Cleveland Indians was first place going into 2016 postseason but having two starting pitchers on the disabled list may have cost them the World Series championship.

The objective of this project is to develop a regression model to predict pitcher injury for the next baseball season. Being able to predict pitcher injury has multiple benefits. It would help a team make plans for their current pitchers who are at high risk of being injured. It could also help the team make decisions on upcoming trade deals based on their injury risk.

# Literature Review

1. Bullard, K. (2016, January 5). Predicting Pitcher Injuries. Retrieved from <http://harvardsportsanalysis.org/2016/01/predicting-pitcher-injuries/>

This paper looks at various baseball statistics from 2010 to 2015 to determine a model for probability of injury in the following season. The focus was on injuries pertaining to pitching stress, such as arm, shoulder, back and side. Variables were tested for normality, which was the case for majority of variables. Transformation was applied to non-normal variables to improve normality. Features were eliminated based on multicollinearity with other features. Four models were created, each tested with cross-validation. The final model consisted of 13 parameters.

1. Carleton, R.A. (2013, Feb 18). Baseball Therapy – What Really Predicts Pitcher Injuries. Retrieved from <http://www.baseballprospectus.com/article.php?articleid=19653>

This article uses data up to the end of the 2011 baseball season to identify the variables associated with pitcher injuries. The dataset was isolated to active pitchers, those appearing as starters more than 80% of their appearances. Forward selection was conducted on 11 variables to find strongest predicator for injuries. The largest predictor for injury was having a previous injury.

1. Chalmer et al. (2015) Correlates With History of Injury in Youth and Adolescent Pitchers. *Arthroscopy. Anthropology: The Journal of Arthroscopic and Related Surgery*, Vol 31, No 7. 1349-1357

This paper used demographic and kinematic dataset of 400 youth and adolescent pitchers. Data was collected through surveys distributed to youth and adolescent pitchers in the author’s metropolitan area. Multivariate logistic regression analysis was used to correlate the history of pitcher injury to the variables. The study found that velocity and pitcher height were predictors of injury.

1. Conte et al. (2015) Prevalence of Ulnar Collateral Ligament Surgery in Professional Baseball Players. *The American Journal of Sports Medical, Vol 43, No. 7.* 1764-1769

This paper discusses the prevalence of Ulnar Collateral Ligament (UCL) surgery among professional baseball players. Conclusions were drawn from online question of 5088 professional ball players. The study included a t-test (P < 0.05) on continuous variables, such as age and years in professional baseball, and chi-square analysis (P < 0.05) on categorical variables, such as level and position. It found that UCL was more prevalent in pitchers compared to non-pitchers. Also, Major league pitchers have a higher prevalence of UCL than minor league pitchers.

1. Cousinau et al. (2010) Outliers detection and treatment: a review. *International Journal of Psychological Research*, Vol 3, No 1. 57-67

This paper discusses how to detect potential outliers in univariate and multivariate data. Three common tests are examined: DFFITS, Cook’s Distance and DFBETAS. Out of the three tests, the authors found that Cook’s distance was able to detect most of the outliers in their simulated data.

1. Erickson et al. (2016) Predicting and Preventing Injury in Major League Baseball. *The American Journal of Orthopedics, March/April 2016.* 152-156

This paper lists the risk factors that are potentially predictors for pitcher injury based on other studies. It also background information on the epidemiology of baseball injuries. No conclusive findings were presented in this paper and suggested more studies would be required to understand factors to prevent injuries.

1. Kotze et al. (2010) Do not log-transform count data. *Methods in Ecology and Evolution*, March 2010. Vol 1, No 2. 118-122

This paper analyzes different transformations that are commonly applied to count data. Log transformations are not recommended. If the data contains one zero record then the data set must be fiddled with by adding a value to the data before transforming. The recommendation is to use transformations based on Poisson and negative binomial distributions.

# Dataset

This project will examine three data sets: Lahman Baseball Database, Disabled List, and PitchFX, from the years 2010 to 2016.

The Lahman’s Baseball Database (<http://www.baseballheatmaps.com/disabled-list-data>) consists of dataset consists of 24 tables. The two key tables, joined by Player ID, which will be used for the analysis are:

* MASTER – Retro ID (rs ID, Player names and biographical info)
* Pitching – PlayerID, pitching statistics (wins, losses, homeruns, walks, etc.)

The Lahman’s R package will be used with already provides the tables a data frame and covers data from 1871 to 2015.

The Disabled List (<http://www.baseballheatmaps.com/disabled-list-data/>) dataset in separate CSV files for individual years spanning 2010 to 2016. Each file provides the following information:

* Player’s name or player ID
* Season and position played
* Location and type of injury
* Date and duration of injury

  The project will only consider records with the pitcher position.

The PitchFX is a system tracks pitch-by-pitch detail of every pitch thrown in every game placed since 2006. The attributes include:

* Velocity
* Movement
* Release point
* Spin
* Location
* Pitcher name
* Batter name

The data is provided in multiple XML format broken down by year, month, day, game and inning (<http://gd2.mlb.com/components/game/mlb/>).

The following is a depiction of the datasets with the attributes used for analysis and their relationship to each other.



# Approach

## Step 1: Scrape PitchFX data from MLB website and Disabled List

The first step is to scrape the MLB website to retrieve PitchFX data. The data is in an XML format. There is 4GB of XML files spanning the years 2010 to 2016. Due to the volume of data, it is not feasible to read the data into a single data frame, as attempted using the PitchRX library. To remediate this, a small .NET application was written to download each XML file from the MLB website. The .NET application is based on code available on GitHub (<https://github.com/mbents/pitchfx-data-download>). After all the files have been downloaded, the .NET application will parse the XMLs into a SQL Server dataset.

The code for the .NET application used for this can be found on GitHub

<https://github.com/r6chow/predictDL/tree/master/pitchfx-data-download-master>

## Step 2: Load Data from Database to R and Impute

After all the data has been loaded into the database, load the PitchFX and Disabled List database into R for each year as separate data frames. Impute missing values using the median as a measure of the central tendency. Aggregate the data for each pitcher for each year using the median.

The following is a snippet of code to load data SQL Server to R. The complete code can be found in Appendix 1 (Markdown).

dbhandle <- odbcDriverConnect('driver={SQL Server};server=localhost;database=PitchFx;trusted\_connection=true');  
   
impute <- function(x, fun) {  
 missing <- is.na(x)  
 replace(x, missing, fun(x[!missing]))  
 }  
   
 centroid\_fun <- median;  
 impute\_fun <- median;  
   
 for (i in 2010:2016)  
 {  
 query <-  
 paste("SELECT ", i, " as season, m.rsID, p.id, p.atbatid, a.pitcher,   
 p.x, p.y, p.start\_speed, p.end\_speed, p.sz\_top, p.sz\_bot,  
 p.px, p.pz, p.x0, p.y0, p.z0, p.vx0, p.vy0, p.vz0, p.ax, p.ay, p.az,  
 p.break\_y, p.break\_length, p.spin\_dir, p.spin\_rate, 1 as type\_ALL,  
 ...  
 CASE WHEN ISNULL(p.pitch\_type,'UN') not in ('AS', 'CH', 'CU', 'EP', 'FA', 'FC', 'FF', 'FO', 'FS', 'FT', 'IN', 'KC', 'KN', 'PO', 'SC', 'SI') THEN 1 ELSE 0 END as type\_UN  
 FROM [PitchFx",i,"].[dbo].[Pitches] p  
 INNER JOIN [PitchFx",i,"].[dbo].[AtBats] a on a.ID = p.AtBatID  
 INNER JOIN [Mapping].[dbo].[RSID\_MLBID\_MAP] m on a.pitcher = m.mlbid  
 INNER JOIN [Lahman].[dbo].[Master] ms on ms.retroID = m.rsID", sep="");  
   
 pitches\_raw <-sqlQuery(dbhandle, query);  
   
 pitches\_impute\_centroid <- ddply(pitches\_raw, ~rsID, transform,  
 x = impute(x, impute\_fun),  
 y = impute(y, impute\_fun),  
 ...   
 spin\_dir = impute(spin\_dir, impute\_fun),  
 spin\_rate = impute(spin\_rate, impute\_fun)  
 );  
   
 pitches\_aggregate <- ddply(pitches\_impute\_centroid, ~rsID, summarise,  
 season = max(season),  
 x = centroid\_fun(x),  
 y = centroid\_fun(y),  
 ...   
 break\_y = centroid\_fun(break\_y),  
 break\_length = centroid\_fun(break\_length),  
 spin\_dir = centroid\_fun(spin\_dir),  
 spin\_rate = centroid\_fun(spin\_rate),  
 );  
   
 assign(paste("pitches",i,sep=""), pitches\_aggregate);   
 };

## Step3: Label the Data

In order to build a regression, the records must be labelled. This is accomplished by joining the aggregated PitchFX data to the Disabled List data in Step 2. The data is joined by rsIDs and by disabled list season to the previous PitchFX season. This is because we want to predict f potential injured players in the following season based on current pitching season (i.e. pitching performance in 2016 will be used to predict who will be on the disabled list in 2017).

A response variable OnDL is added and defined to be 1 if the pitcher is was on the disabled list or 0 otherwise.

The snippet of code is below. The complete code can be found in Appendix 1 (Markdown).

#use previous season to predict DL in current season  
 pitches\_dl <- merge(x=pitches, y=dl, by.x=c("rsID", "season"), by.y=c("rsid", "season\_1"), all.x = TRUE, all.y=FALSE)  
   
 pitches\_dl[pitches\_dl==""] <- NA; #replace blanks with NA  
   
 pitches\_dl$DLDays[is.na(pitches\_dl$DLDays)] <- 0; #no DL pitchers are on DL for 0 days  
 pitches\_dl$OnDL <- (ifelse(pitches\_dl$DLDays>0, 1, 0));  
   
 drops <- c("season\_dl", "DLDays");  
 pitches\_dl <- pitches\_dl[ , !(names(pitches\_dl) %in% drops)];  
   
 pitches\_dl <- pitches\_dl[complete.cases(pitches\_dl),];

## Step 4: Transform Count Variables

The Anscome Transformation (2) was applied to the count variables, as suggested by Kotze 2010, to stabilize the variances

The snippet of code is below. The complete code can be found in Appendix 1 (Markdown).

trf\_func <- function(x) {  
 return ( 2\*sqrt(x+3/8));  
}  
  
pitches\_dl\_dataset$trf\_num\_pitches <- trf\_func(pitches\_dl\_dataset$num\_pitches);  
pitches\_dl\_dataset$trf\_num\_AB <- trf\_func(pitches\_dl\_dataset$num\_AB);  
pitches\_dl\_dataset$trf\_num\_AS <- trf\_func(pitches\_dl\_dataset$num\_AS);  
...  
  
pitches\_dl\_predict$trf\_num\_pitches <- trf\_func(pitches\_dl\_predict$num\_pitches);  
pitches\_dl\_predict$trf\_num\_AB <- trf\_func(pitches\_dl\_predict$num\_AB);  
pitches\_dl\_predict$trf\_num\_AS <- trf\_func(pitches\_dl\_predict$num\_AS);  
...  
  
  
model\_dataset <- pitches\_dl\_dataset[,-grep( "^num\_" , names( pitches\_dl\_dataset ) )];  
predict\_dataset <- pitches\_dl\_predict[,-grep( "^num\_" , names( pitches\_dl\_predict ) )];

## Step 5: Exploratory Analysis and Outlier Detection

A summary of each variable is generated to view the min, max, median, mean and quantiles. The snippet of code is below.

summary(model\_dataset[which(colnames(model\_dataset) %in% original\_var)]);

summary(model\_dataset[which(colnames(model\_dataset) %in% count\_var)]);

A histogram is also generated to view distribution of the data. The snippet of code is below.

d <- melt(model\_dataset[which(colnames(model\_dataset) %in% original\_var)]);

## No id variables; using all as measure variables

ggplot(d,aes(x = value)) + facet\_wrap(~variable,scales = "free\_x") + geom\_histogram();

Potential outliers were detected using Cook’s distance. Influential outliers were defined as observations with a Cook’s distance greater than three times the mean Cook’s distance. The snippet of code is below.

selected\_i <- which(colnames(model\_dataset) %in% selected\_variables);  
   
formula\_text <- paste(response\_var, "~",  
 paste(names(model\_dataset)[selected\_i], collapse="+"));  
formula <- as.formula(formula\_text);  
  
mod <- glm(formula=formula, data=model\_dataset);  
cooksd <- cooks.distance(mod);  
  
plot(cooksd, pch=".", cex=2, main="Influential Obs by Cooks distance"); # plot cook's distance  
abline(h = 3\*mean(cooksd, na.rm=T), col="red"); # add cutoff line

QQ plot were also generated to check the normality of each variable. The snippet of code is below.

for (i in 2:(ncol(model\_dataset\_lessOutliers)-1)){   
 tmp <- model\_dataset\_lessOutliers[, i];  
 qqnorm(tmp, main = colnames(model\_dataset\_lessOutliers[i]));  
 qqline(tmp);  
}

Since there are over 20 variables to consider, correlations were calculated and plotted to identify variables that are positively or negatively correlated. This will help to eliminate some of the dependent variables used in the model.

numeric\_dataset\_lessOutliers <- model\_dataset\_lessOutliers[sapply(model\_dataset\_lessOutliers, is.numeric)];  
  
#ignore column y0 since there is 0 variance  
numeric\_dataset\_lessOutliers <- numeric\_dataset\_lessOutliers[ , !(names(numeric\_dataset\_lessOutliers) %in% c("y0", "trf\_num\_AB", "OnDL"))];  
  
#numeric\_dataset <- numeric\_dataset[1:(ncol(numeric\_dataset))];  
m\_lessOutliers <- cor(numeric\_dataset\_lessOutliers);  
corrplot::corrplot(m\_lessOutliers, type="upper");

## Step 6: Multivariate Logistic Regression and Testing

After the dataset has been prepared and outliers have been removed, logistic regression models were constructed. Models were trained on 75% of the data and tested with the remaining 25%.

The snippet of code for partitioning the data is below.

train <- createDataPartition(model\_dataset$OnDL, p=0.75, list=FALSE);  
  
training <- model\_dataset[train,];  
  
testing <- model\_dataset[-train,];

Various models were created and tested. Testing was done by examining the residual plots, the confusion matrix and accuracy and the ROC curve.

Seven models were created. The first two models examined the effect of outliers to confirm that removing the outliers does improve performance. Additional models were created based on variable correlations. Three models were created using variables that had correlations with cutoff of less than 0.25, 0.5 and 0.75. Based on the results from these models, additional models were created on the most significant variables.

The snippet of code is below for constructing one model, plotting the residual, generating the confusion matrix and plotting the ROC curve. The complete code can be found in Appendix 1 (Markdown).

highlyCorDescr <- findCorrelation(m\_lessOutliers, cutoff =0.25);  
  
lowCorColNames <- colnames(numeric\_dataset\_lessOutliers[,-highlyCorDescr]);  
  
print(lowCorColNames);

selected\_variables <- lowCorColNames;   
  
selected\_i <- which(colnames(training) %in% selected\_variables);  
   
formula\_text <- paste(response\_var, "~",  
paste(names(training\_lessOutliers)[selected\_i], collapse="+"));  
  
formula <- as.formula(formula\_text);  
  
mod\_2c = glm(formula = formula , family=binomial(logit), data=training\_lessOutliers);   
  
summary(mod\_2c);   
  
plot(predict(mod\_2c),residuals(mod\_2c));  
abline(h=0,lty=2,col="grey");  
  
mod\_2c$coefficients;   
  
pred <- ifelse(predict(mod\_2c, testing\_lessOutliers, type='response') > 0.5, 1, 0);  
confusionMatrix(data=pred, reference=testing\_lessOutliers$OnDL);   
  
prob <- predict(mod\_2c, testing\_lessOutliers, type='response');  
g2c <- roc(OnDL ~ prob, data = testing\_lessOutliers);  
plot(g2c)

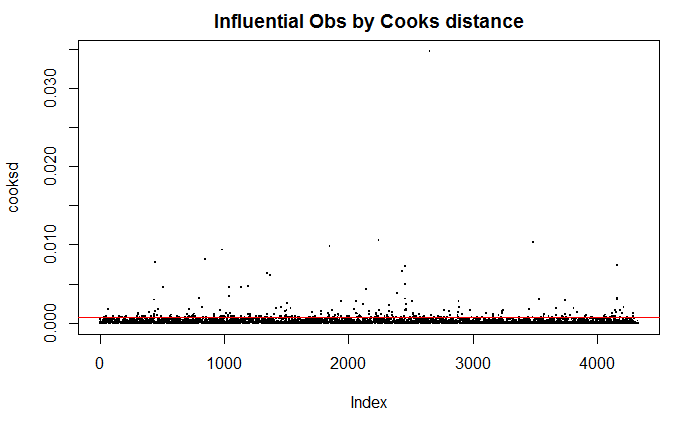
## Step 7: Predctions for 2017

Using the model with the highest overall accuracy, predict the pitchers that will likely be placed on the disabled list in 2017.

pred <- (predict(mod\_2b, pitches\_dl\_predict, type='response'))\*100;  
  
predictions <- data.frame(rsid=pitches\_dl\_predict$rsID, probabilty=pred);  
  
dbhandle <- odbcDriverConnect('driver={SQL Server};server=localhost;database=Lahman;trusted\_connection=true');  
  
query <- "SELECT retroID as rsid, nameFirst, nameLast FROM Master";  
   
players <-sqlQuery(dbhandle, query);

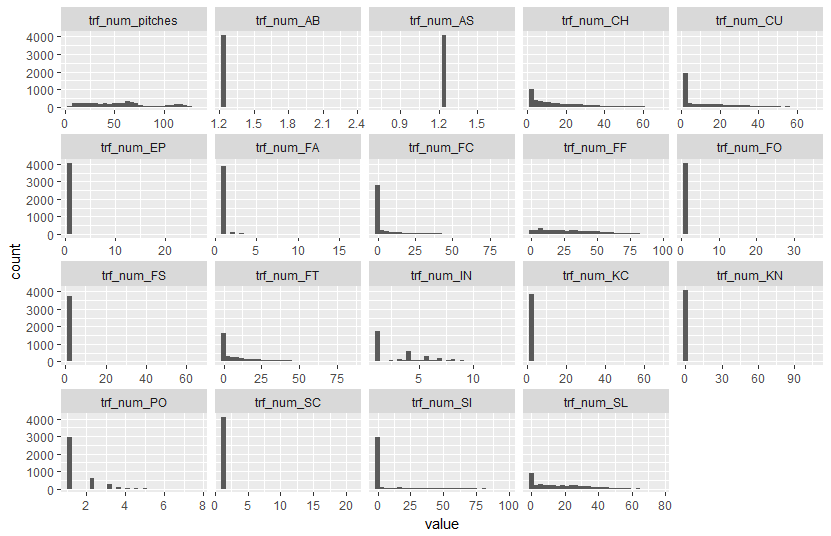
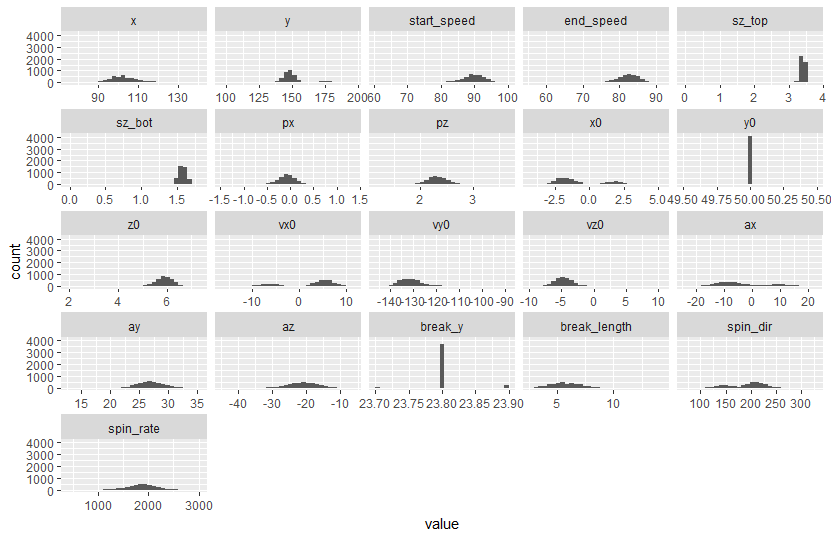
# Results

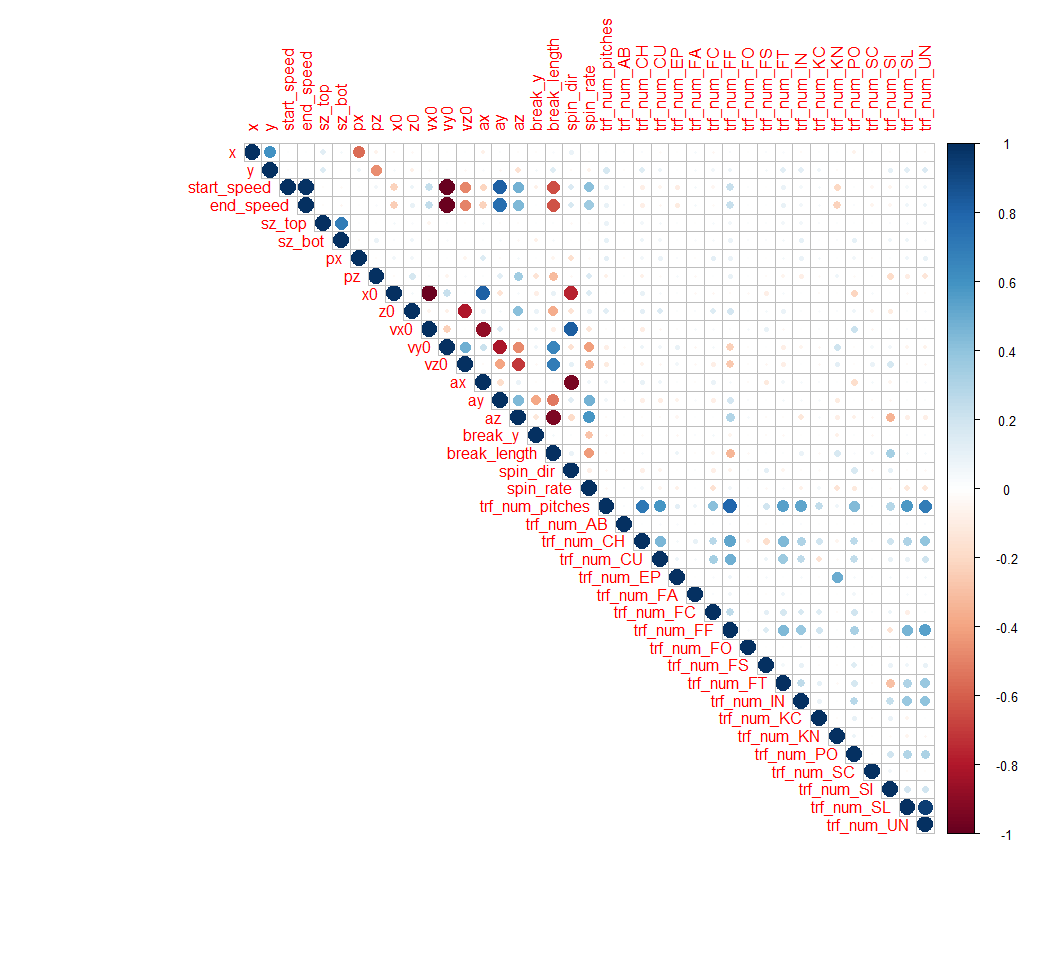
The data set used for modeling consisted of 4,327 observations after aggregating the 5,467,439 pitching events downloaded from the MLB website for the years 2010 to 2016 with 40 variables. The results are summarized below. The detailed results can be found in Appendix 2 (Markdown Knit)



Among the observations, 256 were identified to be potential outliers using Cook’s distance and excluded from the modeling.

The histogram of the physics variables (left) shows that most of the distributions are bell shaped in comparison to the transformed count variables (right)

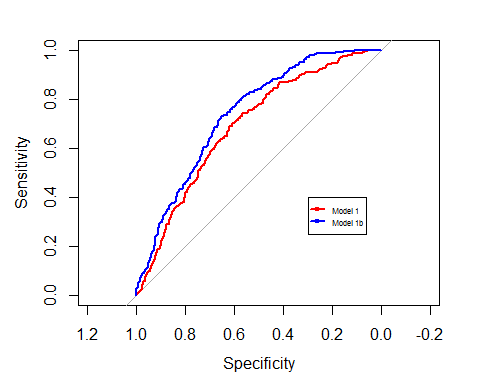


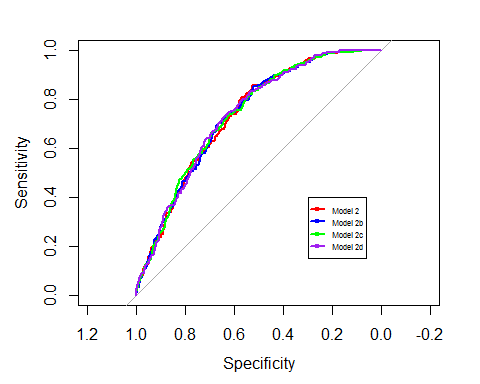


The summary of variables shows that there is no variance in the variable y0 (the distance from home plate to PitchFX system) and tr\_num\_AS (automatic strikes). Correlations were plotted for the remaining variables. Correlations can be observed between velocity variables and the acceleration and break variables, as expected.

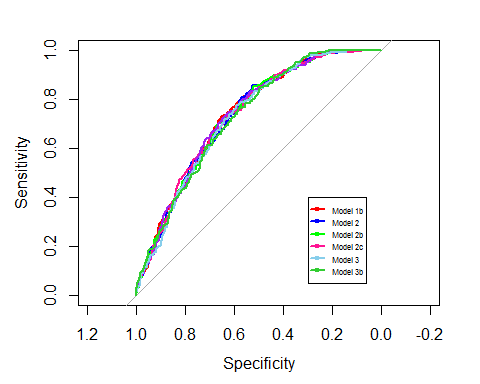
The following table compares the evaluations of the models created. A threshold of 0.4 was used to define if the predicted value is 1 (OnDL) or 0 (not OnDL)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model # | Description | # variables | Confusion Matrix and accuracy | ROC |
| 1 | All data with outliers | 39 | ## Reference  ## Prediction 0 1  ## 0 673 244  ## 1 85 79  ##  ## Accuracy : 0.6957 ## Sensitivity : 0.24458  ## Specificity : 0.88786  ## AUC : 0.690 |  |
| 1b | All data without outliers | 39 | ## Prediction 0 1  ## 0 678 174  ## 1 97 87  ##  ## Accuracy : 0.7427 ## Sensitivity : 0.33333  ## Specificity : 0.88166  ## AUC : 0.738 |  |
| 2 | Correlations cutoff = 0.25 | 17 | ## Reference  ## Prediction 0 1  ## 0 688 196  ## 1 81 65 ##  ## Accuracy : 0.7311  ## Sensitivity : 0.24904  ## Specificity : 0.89467  ## AUC : 0.734 |  |
| 2b | Correlations cutoff = 0.50 | 24 | ## Reference  ## Prediction 0 1  ## 0 688 184  ## 1 81 77  ##  ## Accuracy : 0.7427 ## Sensitivity : 0.29502  ## Specificity : 0.89467  ## AUC : 0.735 |  |
| 2c | Correlations cutoff = 0.75 | 29 | ## Reference  ## Prediction 0 1  ## 0 680 189  ## 1 89 72  ##  ## Accuracy : 0.7301  ## Sensitivity : 0.2759  ## Specificity : 0.8843  ## AUC : 0.736 |  |
| 2d | Correlations cutoff = 0.85 | 32 | ## Reference  ## Prediction 0 1  ## 0 677 174  ## 1 92 87  ##  ## Accuracy : 0.7301  ## Sensitivity : 0.2759  ## Specificity : 0.8843  ## AUC : 0.736 |  |
| 3 | Ignore Pitch Types | 22 | ## Reference  ## Prediction 0 1  ## 0 668 180  ## 1 101 81  ##  ## Accuracy : 0.7272 ## Sensitivity : 0.31034  ## Specificity : 0.86866  ## AUC : 0.729 |  |
| 3b | Select significant variables from models above | 7 | ## Reference  ## Prediction 0 1  ## 0 759 175  ## 1 40 41  ##  ## Accuracy : 0.7882 ## Sensitivity : 0.18981  ## Specificity : 0.94994  ## AUC : |  |

Comparing Model 1 (with outliers) and Model 1b (without outliers), the confusion matrix shows illustrates it generated a more accurate model with higher sensitivity and specificity. The ROC curve also shows that Model 1b has a higher overall accuracy since the plot is curving more to the top left.



Variable correlation was used to reduce the dependent variables needed for the model. Using correlation cutoff values 0.25, 0.5, 0.75 and 0.85, three models were constructed. Model 2b produced high accuracy (0.7427), sensitivity (0.29502) and specificity (0.89467) with more highly correlated variables excluded from the regression in comparison to the other models.



Two additional models were created ignoring the pitch types and selecting significant variables identified from earlier models. Model 2b was selected for the final model to predict the 2017 pitchers that are likely to be on the disabled list. It had the highest overall accuracy, sensitivity and specificity values among the models.

# Conclusions

Various logistic regression models were created with different features selected to determine the probability a pitcher would be placed on the disabled list given how he currently pitches kinetics. The most significant variable is the number of pitches, which appears to be logical since the more pitches a player threw, the higher the probability of injury. All models had an overall accuracy of greater than 70% and improved slightly with outliers and highly correlated variables removed. The factors which significantly impact the probability of a pitcher being injured are:

## px – left/right distance from the middle plate as it crosses home plate   
## pz – height of the pitch as it crosses home plate   
## z0 – height of the pitch at the initial point   
## ay – acceleration in the y axis   
## trf\_num\_FA - number of fastball  
## trf\_num\_FF – number of four seam fastball   
## trf\_num\_FT – number of two seam fastball  
## trf\_num\_IN – number of intentional ball   
## trf\_num\_SI – number of sinker

Using the model with the greatest overall accuracy, the following are the top 10 pitchers that are predicted to have the greatest probably of being on the disabled list for 2017:

J. A. Happ

Kevin Gausman

Aaron Sanchez

Jake Odorizzi\*

Ricky Nolasco

Robbie Ray

Jose Quintana

Ubaldo Jimenez

Jon Lester

Martin Perez

The 2017 has started on April 14 and already Jake Odorizzi has been placed on the disabled list based on the current injury report (<http://mlb.mlb.com/mlb/fantasy/injuries/>) for 2017.

# Appendix 1 (Markdown)

------

title: "Predicting Pitcher DL"

author: "Roger Chow"

date: "April 16, 2017"

output:

word\_document: default

pdf\_document: default

---

# Load libraries

```{r warning=FALSE}

setwd("F:/Capstone\_Workspace/predictDL/");

library('RODBC');

library('DBI');

library('plyr');

library('dplyr');

library('stringi');

library('sqldf');

library('corrplot');

library('reshape2');

library('lattice');

library('ggplot2');

library('caret');

library('logistf');

library('klaR');

library('pROC');

library('pls');

library('ROSE');

library('randomForest')

resetPar <- function() {

dev.new()

op <- par(no.readonly = TRUE)

dev.off()

op

}

par(resetPar());

```

# Prepare data

## Load from Cache

Load the data from cache if the cache exists. This is will save time when re-running the analysis since the data will not change. It takes approximately 30 minutes to load the data from scratch.

```{r warning=FALSE}

need\_load <- TRUE;

if (file.exists("pitches\_dl\_dataset.csv")){

pitches\_dl\_dataset <- read.csv("pitches\_dl\_dataset.csv");

pitches\_dl\_predict <- read.csv("pitches\_dl\_predict.csv");

drops <- c("X");

pitches\_dl\_dataset <- pitches\_dl\_dataset[ , !(names(pitches\_dl\_dataset) %in% drops)];

pitches\_dl\_predict <- pitches\_dl\_predict[ , !(names(pitches\_dl\_predict) %in% drops)];

need\_load <- FALSE;

}

```

## Load pitchFX data from Database

If there is data cached, load the data from the SQL Server database.

Impute missing values using the Median.

The median is used to measure the central tendency for each continuous variable for each pitcher. The count of the pitch types is also calculate for each pitcher in each year.

```{r warning=FALSE}

if (need\_load) {

years <- c(2010, 2011, 2012, 2013, 2014, 2015, 2016);

dbhandle <- odbcDriverConnect('driver={SQL Server};server=localhost;database=PitchFx;trusted\_connection=true');

impute <- function(x, fun) {

missing <- is.na(x)

replace(x, missing, fun(x[!missing]))

}

centroid\_fun <- median;

impute\_fun <- median;

for (i in 2010:2016)

{

query <-

paste("SELECT ", i, " as season, m.rsID, p.id, p.atbatid, a.pitcher,

p.x, p.y, p.start\_speed, p.end\_speed, p.sz\_top, p.sz\_bot,

p.px, p.pz, p.x0, p.y0, p.z0, p.vx0, p.vy0, p.vz0, p.ax, p.ay, p.az,

p.break\_y, p.break\_length, p.spin\_dir, p.spin\_rate,

1 as type\_ALL,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'AB' THEN 1 ELSE 0 END AS type\_AB,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'AS' THEN 1 ELSE 0 END AS type\_AS,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'CH' THEN 1 ELSE 0 END AS type\_CH,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'CU' THEN 1 ELSE 0 END AS type\_CU,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'EP' THEN 1 ELSE 0 END AS type\_EP,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'FA' THEN 1 ELSE 0 END AS type\_FA,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'FC' THEN 1 ELSE 0 END AS type\_FC,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'FF' THEN 1 ELSE 0 END AS type\_FF,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'FO' THEN 1 ELSE 0 END AS type\_FO,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'FS' THEN 1 ELSE 0 END AS type\_FS,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'FT' THEN 1 ELSE 0 END AS type\_FT,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'IN' THEN 1 ELSE 0 END AS type\_IN,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'KC' THEN 1 ELSE 0 END AS type\_KC,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'KN' THEN 1 ELSE 0 END AS type\_KN,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'PO' THEN 1 ELSE 0 END AS type\_PO,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'SC' THEN 1 ELSE 0 END AS type\_SC,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'SI' THEN 1 ELSE 0 END AS type\_SI,

CASE WHEN ISNULL(p.pitch\_type,'UN') = 'SL' THEN 1 ELSE 0 END AS type\_SL,

CASE WHEN ISNULL(p.pitch\_type,'UN') not in ('AS', 'CH', 'CU', 'EP', 'FA', 'FC', 'FF', 'FO', 'FS', 'FT', 'IN', 'KC', 'KN', 'PO', 'SC', 'SI') THEN 1 ELSE 0 END as type\_UN

FROM [PitchFx",i,"].[dbo].[Pitches] p

INNER JOIN [PitchFx",i,"].[dbo].[AtBats] a on a.ID = p.AtBatID

INNER JOIN [Mapping].[dbo].[RSID\_MLBID\_MAP] m on a.pitcher = m.mlbid

INNER JOIN [Lahman].[dbo].[Master] ms on ms.retroID = m.rsID", sep="");

pitches\_raw <-sqlQuery(dbhandle, query);

pitches\_impute\_centroid <- ddply(pitches\_raw, ~rsID, transform,

x = impute(x, impute\_fun),

y = impute(y, impute\_fun),

start\_speed = impute(start\_speed, impute\_fun),

end\_speed = impute(end\_speed, impute\_fun),

sz\_top = impute(sz\_top, impute\_fun),

sz\_bot = impute(sz\_bot, impute\_fun),

px = impute(px, impute\_fun),

pz = impute(pz, impute\_fun),

x0 = impute(x0, impute\_fun),

y0 = impute(y0, impute\_fun),

z0 = impute(z0, impute\_fun),

vx0 = impute(vx0, impute\_fun),

vy0 = impute(vy0, impute\_fun),

vz0 = impute(vz0, impute\_fun),

ax = impute(ax, impute\_fun),

ay = impute(ay, impute\_fun),

az = impute(az, impute\_fun),

break\_y = impute(break\_y, impute\_fun),

break\_length = impute(break\_length, impute\_fun),

spin\_dir = impute(spin\_dir, impute\_fun),

spin\_rate = impute(spin\_rate, impute\_fun)

);

pitches\_aggregate <- ddply(pitches\_impute\_centroid, ~rsID, summarise,

season = max(season),

x = centroid\_fun(x),

y = centroid\_fun(y),

start\_speed = centroid\_fun(start\_speed),

end\_speed = centroid\_fun(end\_speed),

sz\_top = centroid\_fun(sz\_top),

sz\_bot = centroid\_fun(sz\_bot),

px = centroid\_fun(px),

pz = centroid\_fun(pz),

x0 = centroid\_fun(x0),

y0 = centroid\_fun(y0),

z0 = centroid\_fun(z0),

vx0 = centroid\_fun(vx0),

vy0 = centroid\_fun(vy0),

vz0 = centroid\_fun(vz0),

ax = centroid\_fun(ax),

ay = centroid\_fun(ay),

az = centroid\_fun(az),

break\_y = centroid\_fun(break\_y),

break\_length = centroid\_fun(break\_length),

spin\_dir = centroid\_fun(spin\_dir),

spin\_rate = centroid\_fun(spin\_rate),

num\_pitches = sum(type\_ALL),

num\_AB = sum(type\_AB),

num\_AS = sum(type\_AS),

num\_CH = sum(type\_CH),

num\_CU = sum(type\_CU),

num\_EP = sum(type\_EP),

num\_FA = sum(type\_FA),

num\_FC = sum(type\_FC),

num\_FF = sum(type\_FF),

num\_FO = sum(type\_FO),

num\_FS = sum(type\_FS),

num\_FT = sum(type\_FT),

num\_IN = sum(type\_IN),

num\_KC = sum(type\_KC),

num\_KN = sum(type\_KN),

num\_PO = sum(type\_PO),

num\_SC = sum(type\_SC),

num\_SI = sum(type\_SI),

num\_SL = sum(type\_SL),

num\_UN = sum(type\_UN)

);

assign(paste("pitches",i,sep=""), pitches\_aggregate);

};

pitches <- rbind(pitches2010, pitches2011, pitches2012, pitches2013, pitches2014, pitches2015, pitches2016);

pitches <- pitches[complete.cases(pitches),];

close(dbhandle);

}

```

## Load the disabled this

If there is no data cached, load the disable list data from the SQL Server database.

This data contains the list of pitchers on the disable list in each year. Define a new column for season\_1 to denote the previous season. This is required since the pitching data from the previous season will be used to determine if the player will be on the disabled list in the current season.

```{r warning=FALSE}

if (need\_load) {

dbhandle <- odbcDriverConnect('driver={SQL Server};server=localhost;database=PitchFx;trusted\_connection=true');

query <- "

SELECT rsid, 2011 as season\_dl, sum(days) as DLDays

FROM [DisabledList].[dbo].[DL2011]

WHERE Position in ('LHP','RHP','RP','SP','P')

GROUP BY rsid

UNION

SELECT rsid, 2012 as season\_dl, sum(days) as DLDays

FROM [DisabledList].[dbo].[DL2012]

WHERE Pos in ('LHP','RHP','RP','SP','P')

GROUP BY rsid

UNION

SELECT rsid, 2013 as season\_dl, sum(days) as DLDays

FROM [DisabledList].[dbo].[DL2013]

WHERE Position in ('LHP','RHP','RP','SP','P')

GROUP BY rsid

UNION

SELECT rsid, 2014 as season\_dl, sum(days) as DLDays

FROM [DisabledList].[dbo].[DL2014]

WHERE Position in ('LHP','RHP','RP','SP','P')

GROUP BY rsid

UNION

SELECT rsid, 2015 as season\_dl, sum(days) as DLDays

FROM [DisabledList].[dbo].[DL2015]

WHERE Position in ('LHP','RHP','RP','SP','P')

GROUP BY rsid

UNION

SELECT rsid, 2016 as season\_dl, sum(days) as DLDays

FROM [DisabledList].[dbo].[DL2016]

WHERE Position in ('LHP','RHP','RP','SP','P')

GROUP BY rsid

";

dl <- sqlQuery(dbhandle, query);

dl <- dl[complete.cases(dl),];

dl$season\_1 <- dl$season-1;

close(dbhandle);

}

```

## Join the pitch and disabled list data

If there is no data cached, join the pitching data and disabled list data by season.

Define the response variable OnDL to be TRUE if the pitcher is was on the disabled list or FALSE otherwise.

The pitching data from seasons 2010 to 2015 will be used to build and test the model since the disabled list is only available up to the 2016 season.

The pitching data from 2016 will be used to predict which players are most likey be on the disable list in 2017.

```{r warning=FALSE}

if (need\_load) {

#use previous season to predict DL in current season

pitches\_dl <- merge(x=pitches, y=dl, by.x=c("rsID", "season"), by.y=c("rsid", "season\_1"), all.x = TRUE, all.y=FALSE)

pitches\_dl[pitches\_dl==""] <- NA; #replace blanks with NA

pitches\_dl$DLDays[is.na(pitches\_dl$DLDays)] <- 0; #no DL pitchers are on DL for 0 days

pitches\_dl$OnDL <- (ifelse(pitches\_dl$DLDays>0, 1, 0));

drops <- c("season\_dl", "DLDays");

pitches\_dl <- pitches\_dl[ , !(names(pitches\_dl) %in% drops)];

pitches\_dl <- pitches\_dl[complete.cases(pitches\_dl),];

pitches\_dl\_dataset <- pitches\_dl[pitches\_dl$season < 2016,]; #for modeling

pitches\_dl\_predict <- pitches\_dl[pitches\_dl$season == 2016,]; #for 2017 prediction

#pitches\_dl\_dataset$OnDL <- as.factor(ifelse(pitches\_dl\_dataset$DLDays>0, 'YES', 'NO'));

drops <- c("season");

pitches\_dl\_dataset <- pitches\_dl\_dataset[ , !(names(pitches\_dl\_dataset) %in% drops)];

pitches\_dl\_predict <- pitches\_dl\_predict[ , !(names(pitches\_dl\_predict) %in% drops)];

#write to csv to save time

write.csv(pitches\_dl\_dataset, "pitches\_dl\_dataset.csv");

write.csv(pitches\_dl\_predict, "pitches\_dl\_predict.csv");

}

```

## Transform count variables

Apply Anscombe transformation to count variables

```{r warning=FALSE}

trf\_func <- function(x) {

return ( 2\*sqrt(x+3/8));

#return ( log(x+1));

}

pitches\_dl\_dataset$trf\_num\_pitches <- trf\_func(pitches\_dl\_dataset$num\_pitches);

pitches\_dl\_predict$trf\_num\_pitches <- trf\_func(pitches\_dl\_predict$num\_pitches);

#

# for (t in c("AB", "AS", "CH", "CU", "EP", "FA", "FC", "FF", "FO", "FS", "FT", "IN","KC", "KN", "PO", "SC", "SI", "SL", "UN"))

# {

# expression <- paste("pitches\_dl\_dataset$num\_", t, sep="");

# eval\_expression <- eval(parse(text=expression)) /pitches\_dl\_dataset$num\_pitches;

# assign\_var = paste("pitches\_dl\_dataset$trf\_num\_", t, sep="");

# assign(assign\_var, eval\_expression)

#

# expression <- paste("pitches\_dl\_dataset$num\_", t, sep="");

# eval\_expression <- eval(parse(text=expression))/pitches\_dl\_predict$num\_pitches;

# assign\_var = paste("pitches\_dl\_predict$trf\_num\_", t, sep="");

# assign(assign\_var, eval\_expression)

#

# }

pitches\_dl\_dataset$trf\_num\_AB <- trf\_func(pitches\_dl\_dataset$num\_AB)

pitches\_dl\_dataset$trf\_num\_AS <- trf\_func(pitches\_dl\_dataset$num\_AS)

pitches\_dl\_dataset$trf\_num\_CH <- trf\_func(pitches\_dl\_dataset$num\_CH)

pitches\_dl\_dataset$trf\_num\_CU <- trf\_func(pitches\_dl\_dataset$num\_CU)

pitches\_dl\_dataset$trf\_num\_EP <- trf\_func(pitches\_dl\_dataset$num\_EP)

pitches\_dl\_dataset$trf\_num\_FA <- trf\_func(pitches\_dl\_dataset$num\_FA)

pitches\_dl\_dataset$trf\_num\_FC <- trf\_func(pitches\_dl\_dataset$num\_FC)

pitches\_dl\_dataset$trf\_num\_FF <- trf\_func(pitches\_dl\_dataset$num\_FF)

pitches\_dl\_dataset$trf\_num\_FO <- trf\_func(pitches\_dl\_dataset$num\_FO)

pitches\_dl\_dataset$trf\_num\_FS <- trf\_func(pitches\_dl\_dataset$num\_FS)

pitches\_dl\_dataset$trf\_num\_FT <- trf\_func(pitches\_dl\_dataset$num\_FT)

pitches\_dl\_dataset$trf\_num\_IN <- trf\_func(pitches\_dl\_dataset$num\_IN)

pitches\_dl\_dataset$trf\_num\_KC <- trf\_func(pitches\_dl\_dataset$num\_KC)

pitches\_dl\_dataset$trf\_num\_KN <- trf\_func(pitches\_dl\_dataset$num\_KN)

pitches\_dl\_dataset$trf\_num\_PO <- trf\_func(pitches\_dl\_dataset$num\_PO)

pitches\_dl\_dataset$trf\_num\_SC <- trf\_func(pitches\_dl\_dataset$num\_SC)

pitches\_dl\_dataset$trf\_num\_SI <- trf\_func(pitches\_dl\_dataset$num\_SI)

pitches\_dl\_dataset$trf\_num\_SL <- trf\_func(pitches\_dl\_dataset$num\_SL)

pitches\_dl\_dataset$trf\_num\_UN <- trf\_func(pitches\_dl\_dataset$num\_UN)

pitches\_dl\_predict$trf\_num\_AB <- trf\_func(pitches\_dl\_predict$num\_AB)

pitches\_dl\_predict$trf\_num\_AS <- trf\_func(pitches\_dl\_predict$num\_AS)

pitches\_dl\_predict$trf\_num\_CH <- trf\_func(pitches\_dl\_predict$num\_CH)

pitches\_dl\_predict$trf\_num\_CU <- trf\_func(pitches\_dl\_predict$num\_CU)

pitches\_dl\_predict$trf\_num\_EP <- trf\_func(pitches\_dl\_predict$num\_EP)

pitches\_dl\_predict$trf\_num\_FA <- trf\_func(pitches\_dl\_predict$num\_FA)

pitches\_dl\_predict$trf\_num\_FC <- trf\_func(pitches\_dl\_predict$num\_FC)

pitches\_dl\_predict$trf\_num\_FF <- trf\_func(pitches\_dl\_predict$num\_FF)

pitches\_dl\_predict$trf\_num\_FO <- trf\_func(pitches\_dl\_predict$num\_FO)

pitches\_dl\_predict$trf\_num\_FS <- trf\_func(pitches\_dl\_predict$num\_FS)

pitches\_dl\_predict$trf\_num\_FT <- trf\_func(pitches\_dl\_predict$num\_FT)

pitches\_dl\_predict$trf\_num\_IN <- trf\_func(pitches\_dl\_predict$num\_IN)

pitches\_dl\_predict$trf\_num\_KC <- trf\_func(pitches\_dl\_predict$num\_KC)

pitches\_dl\_predict$trf\_num\_KN <- trf\_func(pitches\_dl\_predict$num\_KN)

pitches\_dl\_predict$trf\_num\_PO <- trf\_func(pitches\_dl\_predict$num\_PO)

pitches\_dl\_predict$trf\_num\_SC <- trf\_func(pitches\_dl\_predict$num\_SC)

pitches\_dl\_predict$trf\_num\_SI <- trf\_func(pitches\_dl\_predict$num\_SI)

pitches\_dl\_predict$trf\_num\_SL <- trf\_func(pitches\_dl\_predict$num\_SL)

pitches\_dl\_predict$trf\_num\_UN <- trf\_func(pitches\_dl\_predict$num\_UN)

model\_dataset <- pitches\_dl\_dataset[,-grep( "^num\_" , names( pitches\_dl\_dataset ) )];

predict\_dataset <- pitches\_dl\_predict[,-grep( "^num\_" , names( pitches\_dl\_predict ) )];

```

# Exploratory Analysis

## Summary of data set

### List of variables

```{r warning=FALSE}

dependent\_var <- colnames(model\_dataset[,-grep( "^OnDL" , names( model\_dataset ) )])[-1];

original\_var <- dependent\_var[1:22];

count\_var <- dependent\_var[22:40];

response\_var <- "OnDL";

```

### Summary of continous variables

```{r warning=FALSE}

summary(model\_dataset[which(colnames(model\_dataset) %in% original\_var)]);

```

### Summary of count variables after transforming

```{r warning=FALSE}

summary(model\_dataset[which(colnames(model\_dataset) %in% count\_var)]);

```

## Histogram of data

### Continuous Variables

```{r warning=FALSE}

d <- melt(model\_dataset[which(colnames(model\_dataset) %in% original\_var)]);

ggplot(d,aes(x = value)) + facet\_wrap(~variable,scales = "free\_x") + geom\_histogram();

```

### Count Variables

```{r warning=FALSE}

d <- melt(model\_dataset[which(colnames(model\_dataset) %in% count\_var)]);

ggplot(d,aes(x = value)) + facet\_wrap(~variable,scales = "free\_x") + geom\_histogram();

```

## Outliers

(1) Fox, John. (1991). Regression Diagnostics: An Introduction. Sage Publications.

```{r warning=FALSE}

selected\_variables <- dependent\_var;

selected\_i <- which(colnames(model\_dataset) %in% selected\_variables);

formula\_text <- paste(response\_var, "~",

paste(names(model\_dataset)[selected\_i], collapse="+"));

formula <- as.formula(formula\_text);

mod <- glm(formula=formula, data=model\_dataset);

cooksd <- cooks.distance(mod);

plot(cooksd, pch=".", cex=2, main="Influential Obs by Cooks distance"); # plot cook's distance

abline(h = 3\*mean(cooksd, na.rm=T), col="red"); # add cutoff line

# text(x=1:length(cooksd)+1, y=cooksd, labels=ifelse(cooksd>10\*mean(cooksd, na.rm=T),names(cooksd),""), col="red")

```

### Remove outliers

```{r warning=FALSE}

influential <- as.numeric(names(cooksd)[(cooksd > 3\*mean(cooksd, na.rm=T))]);

model\_dataset\_lessOutliers <- model\_dataset[-(influential[!is.na(influential)]), ];

```

### Summary of data after outliers removed

```{r warning=FALSE}

summary(model\_dataset\_lessOutliers);

```

## Histogram of data

### Continuous Variables with outliers

```{r warning=FALSE}

d <- melt(model\_dataset[which(colnames(model\_dataset) %in% original\_var)]);

ggplot(d,aes(x = value)) + facet\_wrap(~variable,scales = "free\_x") + geom\_histogram();

```

### Count Variables with outliers

```{r warning=FALSE}

d <- melt(model\_dataset[which(colnames(model\_dataset) %in% count\_var)]);

ggplot(d,aes(x = value)) + facet\_wrap(~variable,scales = "free\_x") + geom\_histogram();

```

### Continuous Variables without outliers

```{r warning=FALSE}

d <- melt(model\_dataset\_lessOutliers[which(colnames(model\_dataset\_lessOutliers) %in% original\_var)]);

ggplot(d,aes(x = value)) + facet\_wrap(~variable,scales = "free\_x") + geom\_histogram();

```

### Count Variables without outlisers

```{r warning=FALSE}

d <- melt(model\_dataset\_lessOutliers[which(colnames(model\_dataset\_lessOutliers) %in% count\_var)]);

ggplot(d,aes(x = value)) + facet\_wrap(~variable,scales = "free\_x") + geom\_histogram();

```

## Check normality using QQ plot without outliers

```{r warning=FALSE}

par(mar=c(4,4,4,4))

for (i in 2:(ncol(model\_dataset\_lessOutliers)-1)){

tmp <- model\_dataset\_lessOutliers[, i];

qqnorm(tmp, main = colnames(model\_dataset\_lessOutliers[i]));

qqline(tmp);

}

```

##Display correlation

### All variables

```{r warning=FALSE}

numeric\_dataset <- model\_dataset[sapply(model\_dataset, is.numeric)];

#ignore column y0 since there is 0 variance

numeric\_dataset <- numeric\_dataset[ , !(names(numeric\_dataset) %in% c("y0", "OnDL"))];

#numeric\_dataset <- numeric\_dataset[1:(ncol(numeric\_dataset))];

m <- cor(numeric\_dataset);

corrplot::corrplot(m, type="upper");

#corrplot::corrplot.mixed(m);

```

### All variables without outliers

```{r warning=FALSE}

numeric\_dataset\_lessOutliers <- model\_dataset\_lessOutliers[sapply(model\_dataset\_lessOutliers, is.numeric)];

#ignore column y0 since there is 0 variance

numeric\_dataset\_lessOutliers <- numeric\_dataset\_lessOutliers[ , !(names(numeric\_dataset\_lessOutliers) %in% c("y0", "trf\_num\_AB", "OnDL"))];

#numeric\_dataset <- numeric\_dataset[1:(ncol(numeric\_dataset))];

m\_lessOutliers <- cor(numeric\_dataset\_lessOutliers);

corrplot::corrplot(m\_lessOutliers, type="upper");

#corrplot::corrplot.mixed(m\_lessOutliers);

```

# Model Building with all data

## Model 1 All variables

### Create training and testing set using 75% training and 25% testing

```{r warning=FALSE}

set.seed(123)

train <- createDataPartition(model\_dataset$OnDL, p=0.75, list=FALSE);

training <- model\_dataset[train,];

write.csv(training, "training.csv");

testing <- model\_dataset[-train,];

threshold <- 0.4;

```

### Construct Model

```{r warning=FALSE}

selected\_variables <- dependent\_var;

selected\_i <- which(colnames(training) %in% selected\_variables);

formula\_text <- paste(response\_var, "~",

paste(names(training)[selected\_i], collapse="+"));

formula <- as.formula(formula\_text);

mod\_1 = glm(formula = formula , family=binomial(logit), data=training);

```

### Summary

The summary shows that variables which have some significance to the outcomes are:

end\_speed, sz\_bot, pz, z0, vz0, break\_y, break\_length, trf\_num\_pitches, trf\_num\_CH, trf\_num\_FT, trf\_num\_SI, trf\_num\_UN

```{r warning=FALSE}

summary(mod\_1);

```

### COefficients

```{r warning=FALSE}

mod\_1$coefficients;

```

### Odds Ratio

```{r warning=FALSE}

exp(mod\_1$coefficients);

```

### Residual

```{r warning=TRUE}

plot(predict(mod\_1),residuals(mod\_1));

abline(h=0,lty=2,col="grey");

```

### Performance

```{r warning=FALSE}

pred <- ifelse(predict(mod\_1, testing, type='response') > threshold, 1, 0)

confusionMatrix(data=pred, reference=testing$OnDL, positive='1');

```

### ROC curve

```{r warning=FALSE}

prob <- predict(mod\_1, testing, type='response');

g1 <- roc(OnDL ~ prob, data = testing);

roc.curve(testing$OnDL, prob, plotit =F)

plot(g1)

```

# Model Building without outliers

## Model 1(b) All variables

### Create training and testing set using 75% training and 25% testing

```{r warning=FALSE}

train\_lessOutliers <- createDataPartition(model\_dataset\_lessOutliers$OnDL, p=0.75, list=FALSE);

training\_lessOutliers <- model\_dataset\_lessOutliers[train\_lessOutliers,];

testing\_lessOutliers <- model\_dataset\_lessOutliers[-train\_lessOutliers,];

```

### Construct Model

```{r warning=FALSE}

selected\_variables <- dependent\_var;

selected\_i <- which(colnames(training\_lessOutliers) %in% selected\_variables);

formula\_text <- paste(response\_var, "~",

paste(names(training\_lessOutliers)[selected\_i], collapse="+"));

formula <- as.formula(formula\_text);

mod\_1b = glm(formula = formula , family=binomial(logit), data=training\_lessOutliers);

```

### Summary

The summary shows that variables which have some significance to the outcomes are:

end\_speed, break\_length, trf\_num\_pitches, trf\_num\_FA, trf\_num\_FT, trf\_num\_KN

```{r warning=FALSE}

summary(mod\_1b);

```

### COefficients

```{r warning=FALSE}

mod\_1b$coefficients;

```

### Odds Ratio

```{r warning=FALSE}

exp(mod\_1b$coefficients);

```

### Residual

```{r warning=TRUE}

plot(predict(mod\_1b),residuals(mod\_1b));

abline(h=0,lty=2,col="grey");

```

### Performance

```{r warning=FALSE}

pred <- ifelse(predict(mod\_1b, testing\_lessOutliers, type='response') > threshold, 1, 0)

confusionMatrix(data=pred, reference=testing\_lessOutliers$OnDL, positive='1');

```

### ROC curve

```{r warning=FALSE}

prob <- predict(mod\_1b, testing\_lessOutliers, type='response');

g1b <- roc(OnDL ~ prob, data = testing\_lessOutliers);

roc.curve(testing\_lessOutliers$OnDL, prob, plotit =F)

plot(g1b)

```

### Compare ROC of model with outliers vs model without outliers

The comparison of the ROC curve between the model with outliers (red) and one without outliers (blue) shows that the model has better performance once outliers are removed.

```{r warning=FALSE}

plot(g1, col='red')

plot(g1b, add=TRUE, col='blue')

legend(0.3,0.4, c("Model 1","Model 1b"), lty=c(1,1), lwd=c(2.5,2.5),col=c("red","blue"), pch=1, cex=0.5);

```

## Model 2 Only low correlation variables (less than 0.25)

### Low correlation variables

```{r warning=FALSE}

highlyCorDescr <- findCorrelation(m\_lessOutliers, cutoff = .25)

lowCorColNames<- colnames(numeric\_dataset[,-highlyCorDescr]);

print(lowCorColNames);

```

### Construct Model

```{r warning=FALSE}

selected\_variables <- lowCorColNames;

selected\_i <- which(colnames(training\_lessOutliers) %in% selected\_variables);

formula\_text <- paste(response\_var, "~",

paste(names(training\_lessOutliers)[selected\_i], collapse="+"));

formula <- as.formula(formula\_text);

mod\_2 = glm(formula = formula , family=binomial(logit), data=training\_lessOutliers);

```

### Summary

```{r warning=FALSE}

summary(mod\_2);

```

### Residual

```{r warning=TRUE}

plot(predict(mod\_2),residuals(mod\_2));

abline(h=0,lty=2,col="grey");

```

### COefficients

```{r warning=FALSE}

mod\_2$coefficients;

```

### Odds ratio

```{r warning=FALSE}

exp(mod\_2$coefficients);

```

### Performance

```{r warning=FALSE}

pred <- ifelse(predict(mod\_2, testing\_lessOutliers, type='response') > threshold, 1, 0);

confusionMatrix(data=pred, reference=testing\_lessOutliers$OnDL, positive='1');

```

### ROC curve

```{r warning=FALSE}

prob <- predict(mod\_2, testing\_lessOutliers, type='response');

g2 <- roc(OnDL ~ prob, data = testing\_lessOutliers);

roc.curve(testing\_lessOutliers$OnDL, prob, plotit = F);

plot(g2)

```

## Model 2(b) Only low correlation variables (less than 0.5)

### Low correlation variables

```{r warning=FALSE}

highlyCorDescr <- findCorrelation(m\_lessOutliers, cutoff = .5)

lowCorColNames<- colnames(numeric\_dataset[,-highlyCorDescr]);

print(lowCorColNames);

```

### Construct Model

```{r warning=FALSE}

selected\_variables <- lowCorColNames;

selected\_i <- which(colnames(training\_lessOutliers) %in% selected\_variables);

formula\_text <- paste(response\_var, "~",

paste(names(training\_lessOutliers)[selected\_i], collapse="+"));

formula <- as.formula(formula\_text);

mod\_2b = glm(formula = formula , family=binomial(logit), data=training\_lessOutliers);

```

### Summary

```{r warning=FALSE}

summary(mod\_2b);

```

### Residual

```{r warning=TRUE}

plot(predict(mod\_2b),residuals(mod\_2b));

abline(h=0,lty=2,col="grey");

```

### COefficients

```{r warning=FALSE}

mod\_2b$coefficients;

```

### Odds ratio

```{r warning=FALSE}

exp(mod\_2b$coefficients);

```

### Performance

```{r warning=FALSE}

pred <- ifelse(predict(mod\_2b, testing\_lessOutliers, type='response') > threshold, 1, 0);

confusionMatrix(data=pred, reference=testing\_lessOutliers$OnDL, positive='1');

```

### ROC curve

```{r warning=FALSE}

prob <- predict(mod\_2b, testing\_lessOutliers, type='response');

g2b <- roc(OnDL ~ prob, data = testing\_lessOutliers);

roc.curve(testing\_lessOutliers$OnDL, prob, plotit = F);

plot(g2b)

```

## Model 2(c) Only low correlation variables (less than 0.75)

### Low correlation variables

```{r warning=FALSE}

highlyCorDescr <- findCorrelation(m\_lessOutliers, cutoff = .75);

lowCorColNames <- colnames(numeric\_dataset\_lessOutliers[,-highlyCorDescr]);

#filter\_pitches\_dl\_dataset <- filteredDescr[,-highlyCorDescr]

#all\_variables <- colnames(numeric\_dataset);

print(lowCorColNames);

```

### Construct Model

```{r warning=FALSE}

selected\_variables <- lowCorColNames;

selected\_i <- which(colnames(training) %in% selected\_variables);

formula\_text <- paste(response\_var, "~",

paste(names(training\_lessOutliers)[selected\_i], collapse="+"));

formula <- as.formula(formula\_text);

mod\_2c = glm(formula = formula , family=binomial(logit), data=training\_lessOutliers);

```

### Summary

```{r warning=FALSE}

summary(mod\_2c);

```

### Residual

```{r warning=TRUE}

plot(predict(mod\_2c),residuals(mod\_2c));

abline(h=0,lty=2,col="grey");

```

### COefficients

```{r warning=FALSE}

mod\_2c$coefficients;

```

### Performance

```{r warning=FALSE}

pred <- ifelse(predict(mod\_2c, testing\_lessOutliers, type='response') > threshold, 1, 0);

confusionMatrix(data=pred, reference=testing\_lessOutliers$OnDL, positive='1');

```

### ROC curve

```{r warning=FALSE}

prob <- predict(mod\_2c, testing\_lessOutliers, type='response');

g2c <- roc(OnDL ~ prob, data = testing\_lessOutliers);

roc.curve(testing\_lessOutliers$OnDL, prob, plotit = F)

plot(g2c)

```

## Model 2(d) Only low correlation variables (less than 0.85)

### Low correlation variables

```{r warning=FALSE}

highlyCorDescr <- findCorrelation(m\_lessOutliers, cutoff =0.85);

lowCorColNames <- colnames(numeric\_dataset\_lessOutliers[,-highlyCorDescr]);

#filter\_pitches\_dl\_dataset <- filteredDescr[,-highlyCorDescr]

#all\_variables <- colnames(numeric\_dataset);

print(lowCorColNames);

```

### Construct Model

```{r warning=FALSE}

selected\_variables <- lowCorColNames;

selected\_i <- which(colnames(training) %in% selected\_variables);

formula\_text <- paste(response\_var, "~",

paste(names(training\_lessOutliers)[selected\_i], collapse="+"));

formula <- as.formula(formula\_text);

mod\_2d = glm(formula = formula , family=binomial(logit), data=training\_lessOutliers);

```

### Summary

```{r warning=FALSE}

summary(mod\_2d);

```

### Residual

```{r warning=TRUE}

plot(predict(mod\_2d),residuals(mod\_2d));

abline(h=0,lty=2,col="grey");

```

### COefficients

```{r warning=FALSE}

exp(mod\_2d$coefficients);

```

### Performance

```{r warning=FALSE}

pred <- ifelse(predict(mod\_2d, testing\_lessOutliers, type='response') > threshold, 1, 0);

confusionMatrix(data=pred, reference=testing\_lessOutliers$OnDL, positive='1');

```

### ROC curve

```{r warning=FALSE}

prob <- predict(mod\_2d, testing\_lessOutliers, type='response');

g2d <- roc(OnDL ~ prob, data = testing\_lessOutliers);

roc.curve(testing\_lessOutliers$OnDL, prob, plotit = F)

plot(g2d)

```

## Compare ROC Curve of model by correlation

```{r warning=FALSE}

plot(g2, col='red'); #cutoff 0.25

plot(g2b, add=TRUE, col='blue') #cutoff 0.5

plot(g2c, add=TRUE, col='green') #cutoff 0.75

plot(g2d, add=TRUE, col='purple') #cutoff 0.85

legend(0.3,0.4, c("Model 2","Model 2b","Model 2c", "Model 2d"), lty=c(1,1), lwd=c(2.5,2.5),col=c("red","blue","green", "purple"), pch=1, cex=0.5);

```

## Model 3 Original continuous variables

### Construct Model

```{r warning=FALSE}

selected\_variables <- original\_var;

selected\_i <- which(colnames(training\_lessOutliers) %in% selected\_variables);

formula\_text <- paste(response\_var, "~",

paste(names(training)[selected\_i], collapse="+"));

formula <- as.formula(formula\_text);

mod\_3 = glm(formula = formula , family=binomial(logit), data=training\_lessOutliers);

```

### Summary

```{r warning=FALSE}

summary(mod\_3);

```

### Residual

```{r warning=TRUE}

plot(predict(mod\_3),residuals(mod\_3));

abline(h=0,lty=2,col="grey");

```

### COefficients

```{r warning=FALSE}

mod\_3$coefficients;

```

### Odds Raio

```{r warning=FALSE}

exp(mod\_3$coefficients);

```

### Performance

```{r warning=FALSE}

pred <- ifelse(predict(mod\_3, testing\_lessOutliers, type='response') > threshold, 1, 0);

confusionMatrix(data=pred, reference=testing\_lessOutliers$OnDL, positive='1');

```

### ROC curve

```{r warning=FALSE}

prob <- predict(mod\_3, testing\_lessOutliers, type='response');

g3 <- roc(OnDL ~ prob, data = testing\_lessOutliers);

roc.curve(testing\_lessOutliers$OnDL, prob, plotit = F)

plot(g3)

```

## Model 3(b) Siginificant Continuous Variables + num pitches

### Construct Model

```{r warning=FALSE}

selected\_variables <- c("trf\_num\_pitches", "start\_speed", "vy0", "vz0", "break\_y" );

selected\_i <- which(colnames(training\_lessOutliers) %in% selected\_variables);

formula\_text <- paste(response\_var, "~",

paste(names(training)[selected\_i], collapse="+"));

formula <- as.formula(formula\_text);

mod\_3b = glm(formula = formula , family=binomial(logit), data=training\_lessOutliers);

```

### Summary

```{r warning=FALSE}

summary(mod\_3b);

```

### Residual

```{r warning=TRUE}

plot(predict(mod\_3b),residuals(mod\_3b));

abline(h=0,lty=2,col="grey");

```

### COefficients

```{r warning=FALSE}

mod\_3b$coefficients

exp(mod\_3b$coefficients);

```

### Performance

```{r warning=FALSE}

pred <- ifelse(predict(mod\_3b, testing\_lessOutliers, type='response') > threshold, 1, 0);

confusionMatrix(data=pred, reference=testing\_lessOutliers$OnDL, positive='1');

```

### ROC curve

```{r warning=FALSE}

prob <- predict(mod\_3b, testing\_lessOutliers, type='response');

g3b <- roc(OnDL ~ prob, data = testing\_lessOutliers);

roc.curve(testing\_lessOutliers$OnDL, prob, plotit = F)

plot(g3b)

```

### Compare all models by ROC curve

```{r warning=FALSE}

plot(g1b, col='red');

plot(g2, col='blue', add=TRUE);

plot(g2b, col='green', add=TRUE);

plot(g2c, col='deeppink', add=TRUE);

plot(g2d, col='purple', add=TRUE);

plot(g3, col='skyblue', add=TRUE);

plot(g3b, col='limegreen', add=TRUE);

legend(0.3,0.4, c("Model 1b","Model 2","Model 2b","Model 2c","Model 3","Model 3b"), lty=c(1,1), lwd=c(2.5,2.5),col=c("red","blue","green","deeppink","skyblue","limegreen"), pch=1, cex=0.5);

```

#Cross Validation on significant variables

```{r warning=FALSE}

# highlyCorDescr <- findCorrelation(m\_lessOutliers, cutoff = .85);

#

# lowCorColNames <- colnames(numeric\_dataset\_lessOutliers[,-highlyCorDescr]);

#

#

# #filter\_pitches\_dl\_dataset <- filteredDescr[,-highlyCorDescr]

# #all\_variables <- colnames(numeric\_dataset);

# print(lowCorColNames);

selected\_variables <- c("y","z0", "ay", "trf\_num\_FA", "trf\_num\_FC", "trf\_num\_FF", "trf\_num\_FT", "trf\_num\_IN", "trf\_num\_SI");

selected\_i <- which(colnames(training) %in% selected\_variables);

formula\_text <- paste(response\_var, "~",

paste(names(training\_lessOutliers)[selected\_i], collapse="+"));

formula <- as.formula(formula\_text);

# False positive rate

fpr <- NULL

# False negative rate

fnr <- NULL

# True positive rate

tpr <- NULL

# True negative rate

tnr <- NULL

auc <- NULL

# Number of iterations

k <- 500

# # Initialize progress bar

# pbar <- create\_progress\_bar('text')

# pbar$init(k)

# Accuracy

acc <- NULL

set.seed(123)

for(i in 1:k)

{

# Train-test splitting

# 95% of samples -> fitting

# 5% of samples -> testing

smp\_size <- floor(0.75 \* nrow(model\_dataset\_lessOutliers))

index <- sample(seq\_len(nrow(model\_dataset\_lessOutliers)),size=smp\_size)

train <- model\_dataset\_lessOutliers[index, ]

test <- model\_dataset\_lessOutliers[-index, ]

# Fitting

model <- glm(formula=formula,family=binomial,data=model\_dataset\_lessOutliers)

# Predict results

results\_prob <- predict(model,test,type='response')

# If prob > 0.4 then 1, else 0

results <- ifelse(results\_prob > 0.4,1,0)

# Actual answers

answers <- test$OnDL;

# Accuracy calculation

misClasificError <- mean(answers != results)

# Collecting results

acc[i] <- 1-misClasificError

# Confusion matrix

cm <- confusionMatrix(data=results, reference=answers, positive='1')

tnr[i] <- cm$table[1]/(cm$table[1]+cm$table[2])

tpr[i] <- cm$table[4]/(cm$table[3]+cm$table[4])

fpr[i] <- cm$table[2]/(cm$table[1]+cm$table[2])

fnr[i] <- cm$table[3]/(cm$table[3]+cm$table[4])

auc[i] <- roc.curve(test$OnDL, results\_prob, plotit = F)$auc

# pbar$step()

}

# Average accuracy of the model

mean(acc)

# Average au of the model

mean(auc)

# Histogram of accuracy

hist(acc,xlab='Accuracy',ylab='Freq', col='cyan',border='blue',density=30)

# Histogram of auc

hist(auc,xlab='AUC',ylab='Freq', col='cyan',border='blue',density=30)

# Boxplot of accuracy

boxplot(acc,col='cyan',border='blue',horizontal=T,xlab='Accuracy', main='Accuracy CV')

# Boxplot of auc

boxplot(auc,col='cyan',border='blue',horizontal=T,xlab='AUC', main='AUC CV')

# Confusion matrix and plots of fpr and fnr

mean(fpr)

mean(fnr)

mean(tpr)

mean(tnr)

hist(fpr,xlab='% of fnr',ylab='Freq',main='FPR',

col='cyan',border='blue',density=30)

hist(fnr,xlab='% of fnr',ylab='Freq',main='FNR',

col='cyan',border='blue',density=30)

hist(tpr,xlab='% of fnr',ylab='Freq',main='TPR',

col='cyan',border='blue',density=30)

hist(tnr,xlab='% of fnr',ylab='Freq',main='TNR',

col='cyan',border='blue',density=30)

```

# Predictions for 2017

Model 2b has the highest accuracry rate.

```{r warning=FALSE}

pred <- (predict(mod\_2b, pitches\_dl\_predict, type='response'))\*100;

predictions <- data.frame(rsid=pitches\_dl\_predict$rsID, probabilty=pred);

dbhandle <- odbcDriverConnect('driver={SQL Server};server=localhost;database=Lahman;trusted\_connection=true');

query <- "SELECT retroID as rsid, nameFirst, nameLast FROM Master";

players <-sqlQuery(dbhandle, query);

predictions\_players <- merge(x=predictions, y=players, by="rsid", all.x=TRUE);

head(predictions\_players[rev(order(predictions\_players$probabilty)),], 20);

```