swstr_modeling

Owen McGrattan

10/21/2018

```r
# load in packages
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)
library(baseballr)
library(readr)
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

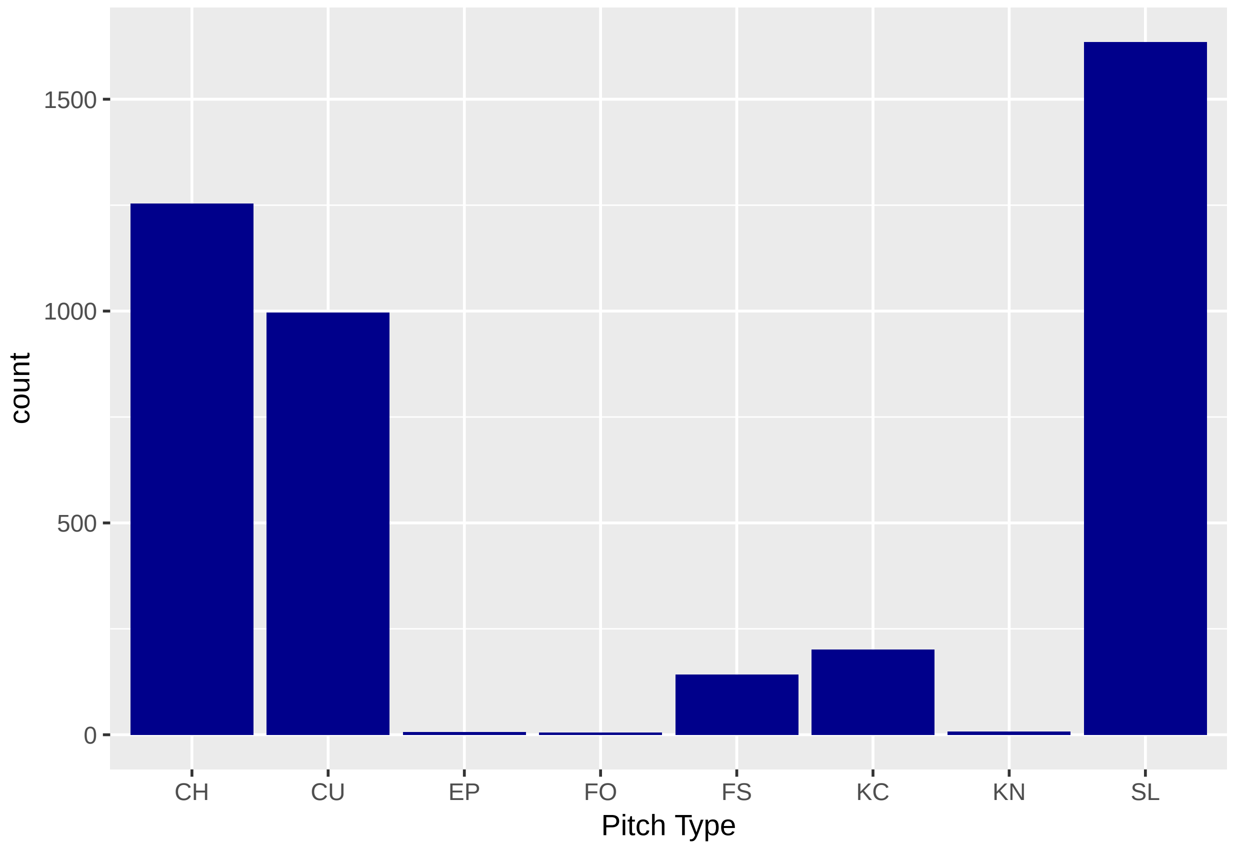
```r
library(stats)
library(rfUtilities)
```

```r
# read in grouped df
df <- read_csv("swstr_pitchers.csv",
    col_types = cols(X1 = col_skip()))
```

```
## Warning: Missing column names filled in: 'X1' [1]

## Warning in read_tokens_(data, tokenizer, col_specs, col_names, locale_, :
## length of NULL cannot be changed

## Warning in read_tokens_(data, tokenizer, col_specs, col_names, locale_, :
## length of NULL cannot be changed

## Warning in read_tokens_(data, tokenizer, col_specs, col_names, locale_, :
## length of NULL cannot be changed
```

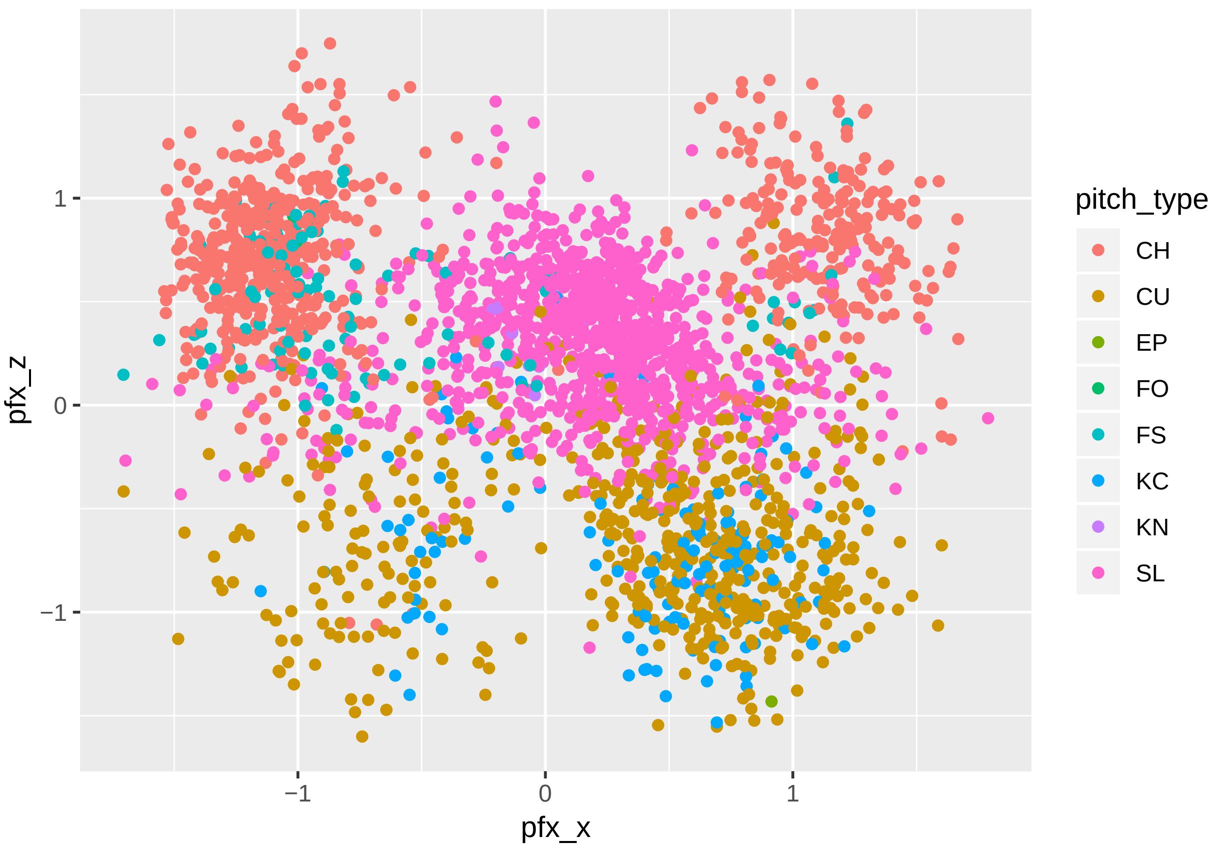
```
# rename our fastball velo column appropriately
colnames(df)[which(names(df) == "fb_diff")] <- "fb_velo"
```

```
ggplot(data.frame(df$pitch_type), aes(df$pitch_type)) + geom_bar(fill = "blue4") + labs(x = "Pitch Type
```



- If we're fitting a model it may just be best to leave the forkballs, knuckles, and ephuses out.
- How do these pitches look on a movement spectrum?

```
ggplot(filter(df, len > 150)) + geom_point(aes(x = pfx_x, y = pfx_z, color = pitch_type))
```



```r
# include closest comp pitcher, weighted for distance from pitcher
pitch_comp <- function(player, pitch_typ, year) {

  dat <- filter(df, pitch_type == pitch_typ, game_year == year)
  scaled <- select(dat, velo, release_pos_x, release_pos_z, pfx_x, pfx_z, pct)
  scaled <- as.data.frame(scale(scaled))
  scaled$player_name <- dat$player_name

  pitch <- filter(scaled, player_name == player)
  dat$dist <- sqrt((pitch$velo - scaled$velo)^2 +
                     (pitch$release_pos_x - scaled$release_pos_x)^2 +
                     (pitch$release_pos_z - scaled$release_pos_z)^2 +
                     (pitch$pfx_x - scaled$pfx_x)^2 +
                     (pitch$pfx_z - scaled$pfx_z)^2 +
                     (pitch$pct - scaled$pct)^2)
  dat <- dat[order(dat$dist),]

  return(mean(head(dat$swstr, n = 6)[2:6]))
}

df$near_comp <- mapply(pitch_comp, df$player_name, df$pitch_type, df$game_year)
```

```r
df$appch_diff <- (df$fb_appch - df$appch_angle)^2
df$dist_from_fb <- sqrt((df$plate_x - df$fb_platex)^2 + (df$plate_z - df$fb_platez)^2)
```

```r
# shrink our sample down to those who have thrown at least 150 of an offspeed pitch, none of the more o
samp <- df %>% filter(len >= 150, pitch_type != "KN", pitch_type != "FO", pitch_type != "EP")
```

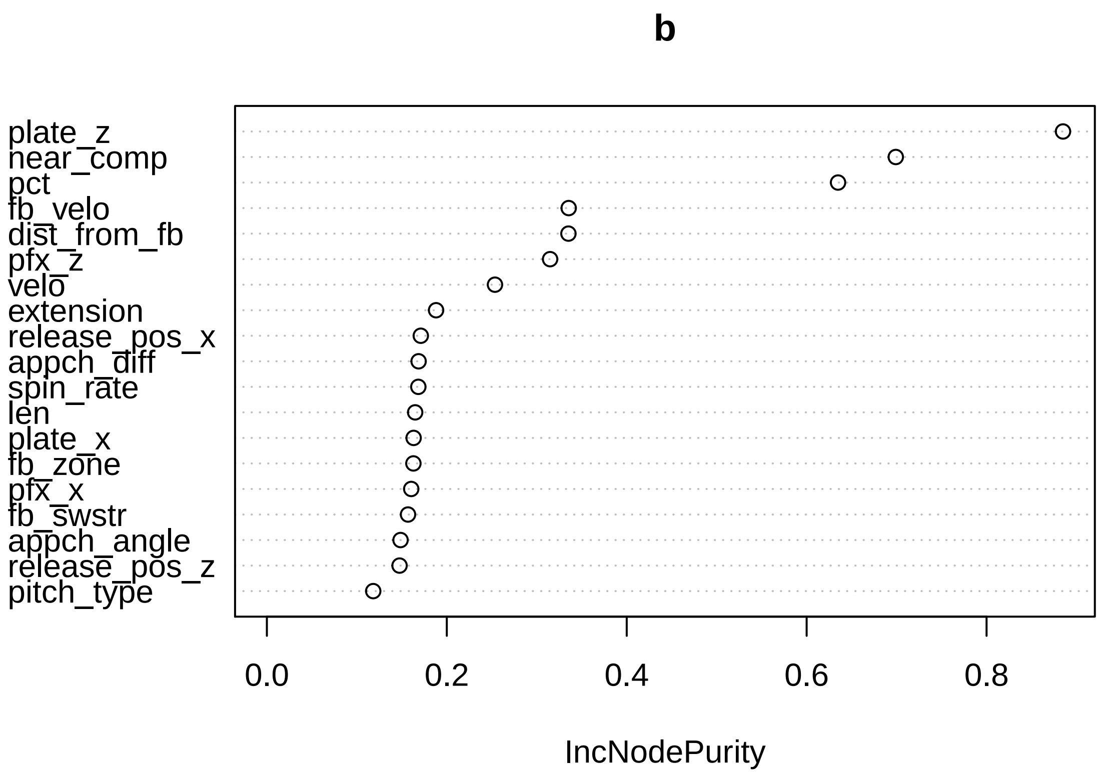
```r
samp$pitch_type <- as.factor(samp$pitch_type)
samp$handedness <- as.factor(samp$handedness)
```

```r
set.seed(121)
# create test and train data
samp <- samp %>% mutate(id = row_number())

train <- na.omit(samp %>% sample_frac(.80))
test <- na.omit(anti_join(samp, train, by = "id"))
train <- select(train, -c(id, handedness, fb_pfxx, fb_pfxz, fb_appch, fb_platex, fb_platez))
test <- select(test, -c(id, handedness, fb_pfxx, fb_pfxz, fb_appch, fb_platex, fb_platez))
```

```r
# random forest

b <- randomForest(formula = swstr ~ .,
                data = train[,-c(1, 3)], na.action = na.omit, ntree = 1000, mtry = 8, replace = T, corr
print(b)
```

```
##
## Call:
##  randomForest(formula = swstr ~ ., data = train[, -c(1, 3)], ntree = 1000,      mtry = 8, replace = 
##                Type of random forest: regression
##                      Number of trees: 1000
## No. of variables tried at each split: 8
##
##           Mean of squared residuals: 0.001453
##                     % Var explained: 46.82
##                       Test set MSE: 0
##                     % Var explained: 44.21
##   Bias correction applied:
##   Intercept:  -0.0001228813
##       Slope:  1.112216
```

```r
varImpPlot(b)
```



- Taking a general overview of our model, an R^2 of ~ 0.44 tells us a strong deal given just pure "stuff" and not taking the context of each individual pitch. Of course there's a good deal of variation that we should not expect to be explained in the data I fed into the model. While I do give pct usage and total num of pitches thrown, I do not give any information about usage by count, sequencing, batter handedness, leverage, so on and so forth.
- It's a little bit frustrating that nearest comps was one of our strongest predictors but it makes sense given that the context of each individual pitch matters and that we wouldn't be able to grasp any part of that with the data given to the model. However not so surprising was that avg vertical location was a key factor as well as percent usage.
- We've put together a decent model but it has a finite application. Given that swstr% is something that'll stabilize fairly quickly it doesn't necessarily help identify current MLB candidates that will see a jump in performance, we can simply look at MLB pitchers who generate a strong amount of whiffs on one or more misused offspeed pitches. This prediction is something that better serves identifying the minor league talent (in an ideal scenario where we have movement data on these pitches) who project to produce above average whiffs and therefore strikeouts. It'd greatly benefit our model to include swstr%'s for given players in our data set who have pitched in either the AA or AAA level. Unfortunately, while we can grab swstr% for pitchers, we do not have publically available pitch-by-pitch data with each pitch type tracked.