Finding Commanalities Among Baseball Pitchers

Xander Schwartz

Data Read In

```
load("data/allPitches.rda")
load("data/final_pitcher_data.rda")
```

```
ID_lookup <- baseballr::get_chadwick_lu() %>%
  filter(mlb_played_last >= 2015) %>%
  mutate(name = paste(name_first, name_last)) %>%
  select(name, key_mlbam)
```

```
## Rows: 400515 Columns: 40
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (12): key_person, key_uuid, key_retro, key_bbref, key_bbref_minors, key_...
## dbl (20): key_mlbam, key_fangraphs, birth_year, birth_month, birth_day, deat...
## lgl  (8): key_sr_nfl, key_sr_nba, key_sr_nhl, key_findagrave, mlb_managed_fi...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
leader_BA <- scrape_savant_leaderboards(
  leaderboard = "expected_statistics",
  year = 2015,
  abs = 0,
  min_pa = 1,
  min_pitches = 100,
  min_field = "q",
  min_run = 0,
  player_type = "batter",
  fielding_type = "player",
  oaa_position = "",
  oaa_roles = "",
  team = "",
  arsenal_type = "n_",
  run_type = "raw",
  min2b = 0,
  min3b = 0,
  position = "",
  bats = "",
  hand = ""
)
```

```
## Rows: 896 Columns: 15
```

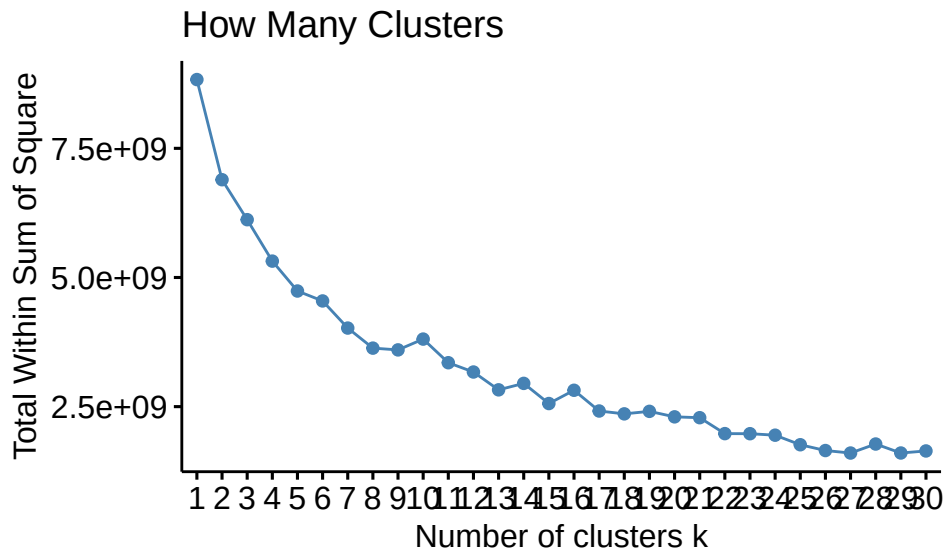
```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (5): last_name, first_name, est_ba, est_slg, est_woba
## dbl (10): player_id, year, pa, bip, ba, est_ba_minus_ba_diff, slg, est_slg_m...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
leader_BA_Pit <- scrape_savant_leaderboards(
  leaderboard = "expected_statistics",
  year = 2015,
  abs = 0,
  min_pa = 1,
  min_pitches = 100,
  min_field = "q",
  min_run = 0,
  player_type = "pitcher",
  fielding_type = "player",
  oaa_position = "",
  oaa_roles = "",
  team = "",
  arsenal_type = "n_",
  run_type = "raw",
  min2b = 0,
  min3b = 0,
  position = "",
  bats = "",
  hand = ""
)
```

```
## Rows: 734 Columns: 18

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): last_name, first_name, era
## dbl (15): player_id, year, pa, bip, ba, est_ba, est_ba_minus_ba_diff, slg, e...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

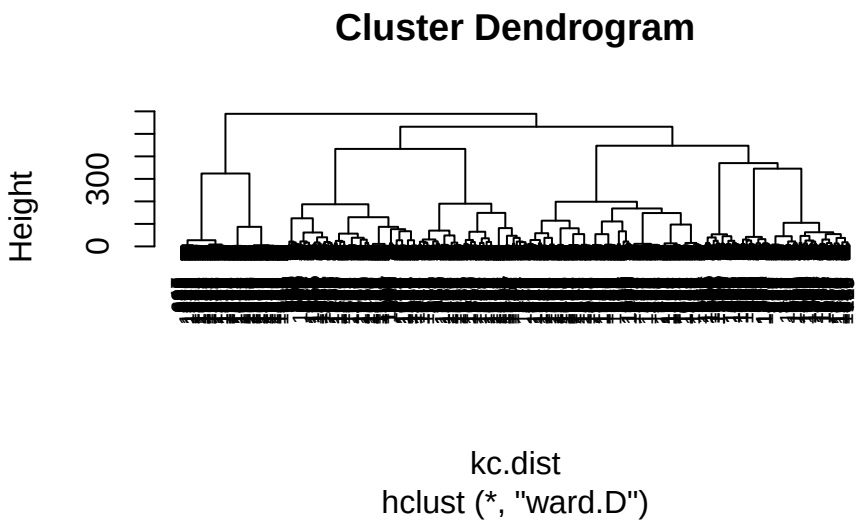
Creating Clusters

```
fviz_nbclust(final_pitcher_data[c(2:37)], kmeans, method = "wss", k.max = 30) +
  ggtitle("How Many Clusters")
```



I will keep 27 clusters and use Ward's Method to determine my clusters. I will keep 27 clusters as there appears to be a plateau after 27.

```
# Using Wards Method to Create my Clusters
kc.dist <- dist(scale(final_pitcher_data[c(2:37)]))
hcward <- hclust(kc.dist, method = "ward.D")
plot(hcward, cex = 0.7)
```




```
# Extra Wrangling
wardSol <- (cutree(hcward, k = 27))

cluster_pitcher_data <- cbind(final_pitcher_data, wardSol)
(table(cluster_pitcher_data$wardSol))
```

```
##
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
## 42 73 26 40 55 64 90 38 45 63 54 52 50  9 51 92 23 33 67 36 43 46 19 22 30 22
## 27
## 19
```

```
cluster_pitcher_data <- left_join(cluster_pitcher_data, ID_lookup, by = c("pitcher" = "key_mlbam"))
```

```
leader_BA <- leader_BA %>% select(first_name, last_name, year, player_id, ba, pa)
combinedData <- left_join(all_pitches, leader_BA, by = c("batter" = "player_id"))
leader_BA_Pit <- leader_BA %>% select(first_name, last_name, year, player_id, ba, pa)
combinedData2 <- left_join(combinedData, leader_BA_Pit, by = c("pitcher" = "player_id"))


leader_with_cluster <- left_join(combinedData2, cluster_pitcher_data) %>% filter(!is.na(wardSol))  %>%
  filter(Result == "double" | Result == "triple" | Result == "home run" | Result == "out" | Result == "
  mutate(isHit = ifelse(Result == "out", 0, 1))
```

```
## Joining, by = "pitcher"
```

Preliminary Analysis

General Terms

While this is a study designed for a general audience, understanding some baseball terminology could be useful. A few terms/parts of the data that might be helpful to understand throughout the project.

Fastball: A fastball is generally the most basic pitch in baseball. Fastballs are generally thrown the most and the fastest.

Breaking Ball: A breaking ball is usually a pitch that moves the most in baseball. There are two main types: sliders and curveballs. Sliders are thrown faster and break less (more horizontal movement than vertical). Curveballs are thrown slower and have more vertical movement.

Offspeed: Offspeed pitches are designed to look like fastballs but are thrown slower. This is meant to throw off a hitter's timing, so they swing too early.

Spin Rate: Spin rate is a measure of how often a ball spins during the time it is thrown to the plate. Spin rate is generally a measure of the quality of a pitch.

Break: Break is a measure of how much the ball moves in it's journey to the plate (horizontally or vertically).

Starter: A starter is a pitcher who pitches once every five days, but pitches the most in total quantity.

Reliever: A reliever is a pitcher who pitches frequently but for less time (fewer innings).

ERA: ERA, or Earned Run Average, is a measure of how many runs are allowed by a pitcher every nine innings. Lower is better.

wOBA: wOBA, or weighted On Base Average is a measure of how many hits (weighted by type of hit) a piker gives up each plate appearance. Lower is better (for pitchers). Exit Velocity: Exit velocity is a measure of how hard the ball is hit on average. A harder hit ball is bad for the pitcher.

```
groupBatterPitcher <- leader_with_cluster %>%
  group_by(pitcher, batter) %>%
  summarise(P_AB = n())
```

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

```
groupBatterPitcherHits <- leader_with_cluster %>%
  group_by(batter, pitcher) %>%
  summarise(P_Hits = sum(isHit))
```

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

```
pitcherCombo <- inner_join(groupBatterPitcherHits, groupBatterPitcher)
```

```
## Joining, by = c("batter", "pitcher")
```

```
all_info <- inner_join(leader_with_cluster, pitcherCombo)
```

```
## Joining, by = c("pitcher", "batter")
```

```
groupBatterCluster <- all_info %>%
  group_by(batter, wardSol) %>%
  summarise(C_AB = n())
```

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

```
groupBatterClusterHits <- all_info %>%
  group_by(batter, wardSol) %>%
  summarise(C_Hits = sum(isHit))
```

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

```
clusterCombo <- inner_join(groupBatterClusterHits, groupBatterCluster)
```

```
## Joining, by = c("batter", "wardSol")
```

```
all_info_cluster <- inner_join(all_info, clusterCombo)
```

```
## Joining, by = c("batter", "wardSol")
```

Cluster Modeling

```
filteredData <- all_info_cluster %>%
  filter(P_AB > 5 & C_AB > 5) %>%
  drop_na()

options(scipen = 10000)

# so it doesn't know the result of this AB
filteredData <- filteredData %>%
  mutate(P_AB = P_AB - 1) %>%
  mutate(C_AB = C_AB - 1) %>%
  mutate(P_Hits = P_Hits - isHit) %>%
  mutate(C_Hits = C_Hits - isHit)

filteredData <- filteredData %>%
  mutate(P_AVG = P_Hits / P_AB) %>%
  mutate(C_AVG = C_Hits / C_AB)
simpleMod <- glm(isHit ~ ba.x + ba.y, data = filteredData, family = "binomial")
pitcherMod <- glm(isHit ~ ba.x + ba.y + P_AB * P_Hits:P_AVG, data = filteredData, family = "binomial")
pitcherMod
```

```
##
## Call:  glm(formula = isHit ~ ba.x + ba.y + P_AB * P_Hits:P_AVG, family = "binomial",
##     data = filteredData)
##
## Coefficients:
```

```
##       (Intercept)              ba.x              ba.y              P_AB
##          -1.53278           2.32309          -0.01060          -0.00169
##      P_Hits:P_AVG  P_AB:P_Hits:P_AVG
##           0.04033          -0.00051
##
## Degrees of Freedom: 182699 Total (i.e. Null);  182694 Residual
## Null Deviance:       220000
## Residual Deviance: 219000    AIC: 219000
```

```
clusterMod <- glm(isHit ~ ba.x + ba.y + C_AB * C_Hits * C_AVG, data = filteredData, family = "binomial")
bothMod <- glm(isHit ~ ba.x + ba.y + C_AB * C_Hits * C_AVG + P_AB * P_Hits * P_AVG, data = filteredData
clusterMod
```

```
##
## Call:  glm(formula = isHit ~ ba.x + ba.y + C_AB * C_Hits * C_AVG, family = "binomial",
##     data = filteredData)
##
## Coefficients:
##       (Intercept)              ba.x              ba.y              C_AB
##         -1.628043          1.842988         -0.028376          0.000704
##            C_Hits             C_AVG       C_AB:C_Hits       C_AB:C_AVG
##         -0.000518          0.575783         -0.000024                NA
##      C_Hits:C_AVG  C_AB:C_Hits:C_AVG
##          0.003853          0.000057
##
## Degrees of Freedom: 182699 Total (i.e. Null);  182691 Residual
## Null Deviance:       220000
## Residual Deviance: 219000    AIC: 219000
```

```
anova(simpleMod, clusterMod, test = "LR")
```

```
## Analysis of Deviance Table
##
## Model 1: isHit ~ ba.x + ba.y
## Model 2: isHit ~ ba.x + ba.y + C_AB * C_Hits * C_AVG
##   Resid. Df Resid. Dev Df Deviance            Pr(>Chi)
## 1    182697     219320
## 2    182691     219200  6    119.9 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(simpleMod, pitcherMod, test = "LR")
```

```
## Analysis of Deviance Table
##
## Model 1: isHit ~ ba.x + ba.y
## Model 2: isHit ~ ba.x + ba.y + P_AB * P_Hits:P_AVG
##   Resid. Df Resid. Dev Df Deviance       Pr(>Chi)
## 1    182697     219320
## 2    182694     219274  3    46.09 0.000000000542 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(clusterMod, bothMod, test = "LR")
```

```
## Analysis of Deviance Table
##
## Model 1: isHit ~ ba.x + ba.y + C_AB * C_Hits * C_AVG
## Model 2: isHit ~ ba.x + ba.y + C_AB * C_Hits * C_AVG + P_AB * P_Hits *
##     P_AVG
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1    182691     219200
## 2    182685     219174  6    25.44 0.000284 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
predicted_data <- filteredData %>%
  mutate(isHitP = predict(pitcherMod, filteredData, type = "response")) %>%
  mutate(isHitC = predict(clusterMod, filteredData, type = "response")) %>%
  mutate(isHitBoth = predict(bothMod, filteredData, type = "response"))
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```r
mean(predicted_data$isHit)
```

```
## [1] 0.28879
```

```r
table(predicted_data$events)
```

```
##
##                    double               double_play              field_error
##                      9540                       475                      1469
##                 field_out       fielders_choice_out                 force_out
##                     84446                       315                      3835
## grounded_into_double_play                  home_run                 other_out
##                      4051                      6536                        44
##          pickoff_error_2b      sac_bunt_double_play       sac_fly_double_play
##                         1                         2                        18
##                    single                 strikeout     strikeout_double_play
##                     31003                     39828                       196
##                    triple               triple_play
##                       935                         6
```

```r
table(predicted_data$events)
```

```
##
##                    double               double_play              field_error
##                      9540                       475                      1469
##                 field_out       fielders_choice_out                 force_out
```

```
##                      84446                        315                       3835
## grounded_into_double_play                   home_run                  other_out
##                       4051                       6536                         44
##          pickoff_error_2b      sac_bunt_double_play       sac_fly_double_play
##                          1                          2                         18
##                     single                  strikeout     strikeout_double_play
##                      31003                      39828                        196
##                     triple                triple_play
##                        935                          6
```

```
set.seed(5)
testSamp <- sample(predicted_data, 5000)

sum(testSamp$isHit)
```

```
## [1] 1486
```

```
sum(testSamp$isHitC)
```

```
## [1] 1445.58
```

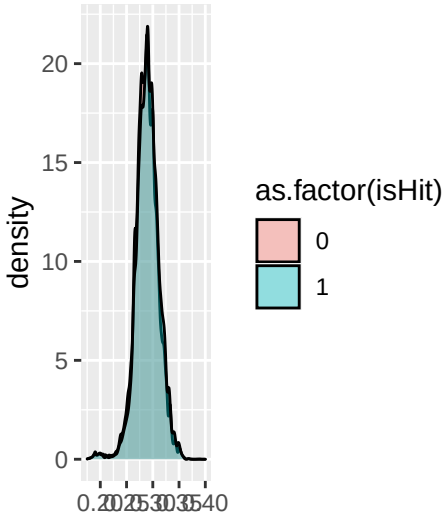
```
sum(testSamp$isHitP)
```

```
## [1] 1444.15
```

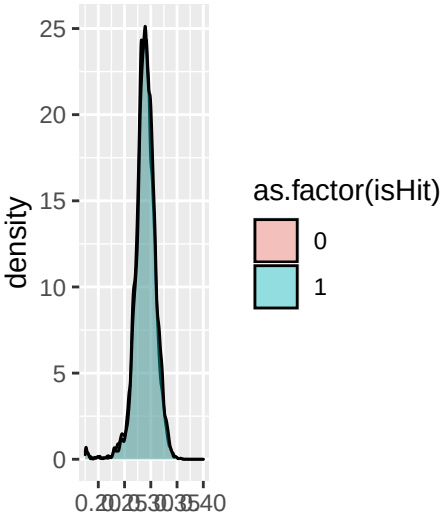
```
c <- ggplot(predicted_data, aes(x = isHitC, fill = as.factor(isHit))) +
   geom_density(alpha=.4)  +
  xlim(.175, .4) +
  xlab("Predicted Batting Average Via Clustering") +
  scale_color_discrete(labels = c("Out", "Hit")) +
  labs(color = "Actual Result")
p <- ggplot(predicted_data, aes(x = isHitP, fill = as.factor(isHit))) +
  geom_density(alpha=.4) +
  xlim(.175, .4) +
  xlab("Predicted Batting Average Via Pitcher") +
  scale_color_discrete(labels = c("Out", "Hit")) +
  labs(color = "Actual Result")

cowplot::plot_grid(c,p)
```

```
## Warning: Removed 16 rows containing non-finite values (stat_density).
```

```
## Warning: Removed 98 rows containing non-finite values (stat_density).
```



d Batting Average Via Clustering



Predicted Batting Average Via Pitcher

```
rbind(describe(x = predicted_data$isHitP), describe(x = predicted_data$isHitC))
```

```
##     vars      n mean   sd median trimmed  mad  min  max range  skew kurtosis se
## X1     1 182700 0.29 0.02   0.29    0.29 0.02 0.17 0.45  0.28 -0.94     5.02  0
## X11    1 182700 0.29 0.02   0.29    0.29 0.02 0.17 0.39  0.22 -0.41     2.03  0
```

‘