Finding Commanalities Among Baseball Pitchers

Xander Schwartz

Introduction

In my study, I hope to look at Major League Baseball Pitchers and see if I can use PCA and Clustering, to predict outcomes in Major League Baseball. Each pitcher throws a mix of pitches at different rates, speeds, and movements. I hope to study whether I can more accurately predict baseball results by finding similarities between pitchers in these statistics. For example, if I find that pitcher A and pitcher B have similar profiles through PCA and Clustering, does that mean that the pitchers will have similar results? Do the best pitchers in baseball have similar profiles? Clustering will be used to help group similar pitchers together, and PCA will be used to understand the similarities and differences of the clusters. Overall, I hope that this study can be a step forward in understanding what the defining traits and skills are of a baseball pitcher's quality.

Preliminary Analysis

My data comes from Major League Baseball's "BaseballSavant" Website. My data includes an observation for each pitcher with at least 100 plate appearances for every year since 2015 (each year for each player is a different observation) who throws at least one pitch in 3 different categories.(BaseballSavant, 2020) This leaves us with 2,498 observations. For each player, I am mainly looking at 3 different pitch types: "fastball," "breaking ball," and "offspeed." (BaseballSavant, 2020) For each pitch category I am looking at four statistics: the rate at which the pitch is thrown, the average speed of the pitch (mph), the average spin of the pitch (rpm/rotations per minute), and the average break of the pitch (inches) for a total of twelve variables. I left out the rate for offspeed, as it is already in the data through the other rates (1-other rates = offspeed rate) (BaseballSavant, 2020) I also have some other statistics that look at the overall quality of the pitcher (era, woba, and exit velocity average). These will be used at the end of my analysis to see if my work accurately finds meaningful patterns of pitcher quality (i.e. if I clustered the pitchers, would certain clusters have a higher average quality?).

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.

Data Wrangling

```
# Renaming Columns
fullData <- mydata %>% drop_na()

fullPitchers <- fullData %>%
  rename("fbRate" = n_fastball_formatted) %>%
  rename("fbSpin" = fastball_avg_spin) %>%
  rename("fbSpeed" = fastball_avg_speed) %>%
  rename("fbBreak" = fastball_avg_break) %>%
```

```
  rename("bbRate" = n_breaking_formatted) %>%
  rename("bbSpin" = breaking_avg_spin) %>%
  rename("bbBreak" = breaking_avg_break) %>%
  rename("bbSpeed" = breaking_avg_speed) %>%
  rename("offSpin" = offspeed_avg_spin) %>%
  select(!n_offspeed_formatted) %>%
  rename("offSpeed" = offspeed_avg_speed) %>%
  rename("offBreak" = offspeed_avg_break)

glimpse(fullPitchers)
```

```
## Rows: 2,498
## Columns: 18
## $ last_name         <chr> "Zimmermann", "Wood", "Wilson", "Wainwright", "Trigg~
## $ first_name        <chr> " Jordan", " Travis", " Justin", " Adam", " Andrew",~
## $ year              <int> 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017~
## $ p_game            <int> 29, 39, 65, 24, 12, 33, 44, 57, 31, 32, 32, 32, 17, ~
## $ woba              <dbl> 0.372, 0.385, 0.281, 0.340, 0.316, 0.310, 0.301, 0.2~
## $ exit_velocity_avg <dbl> 89.3, 88.9, 86.6, 86.7, 88.7, 88.7, 88.0, 82.7, 86.5~
## $ era               <dbl> 6.08, 6.80, 3.41, 5.12, 4.29, 3.09, 3.79, 3.84, 2.52~
## $ fbRate            <dbl> 54.0, 76.8, 90.0, 69.6, 44.2, 66.1, 49.1, 59.3, 56.4~
## $ fbSpeed           <dbl> 92.2, 88.2, 94.9, 88.0, 88.9, 93.2, 92.6, 88.6, 93.2~
## $ fbSpin            <int> 2339, 2088, 2439, 2249, 2389, 2274, 2533, 2137, 2495~
## $ fbBreak           <dbl> 19.1, 18.3, 19.9, 14.4, 16.5, 12.9, 23.1, 19.6, 18.7~
## $ bbRate            <dbl> 39.8, 14.3, 9.8, 26.1, 55.1, 27.8, 8.4, 35.2, 29.1, ~
## $ bbSpeed           <dbl> 84.6, 77.3, 84.8, 72.5, 79.4, 85.7, 84.3, 74.4, 83.1~
## $ bbSpin            <int> 2389, 1983, 2440, 2643, 2551, 2735, 2437, 2304, 2433~
## $ bbBreak           <dbl> 8.4, 13.9, 6.0, 20.6, 10.2, 12.1, 7.1, 19.0, 8.5, 9.~
## $ offSpeed          <dbl> 86.3, 81.0, 89.9, 81.4, 82.3, 82.7, 83.3, 84.8, 84.6~
## $ offSpin           <int> 2245, 1742, 1523, 1788, 1903, 1680, 1300, 1753, 1519~
## $ offBreak          <dbl> 17.6, 17.2, 20.8, 17.3, 15.7, 17.3, 14.3, 17.2, 15.1~
```

```
head(fullPitchers, 5)
```

```
##    last_name first_name year p_game  woba exit_velocity_avg  era fbRate fbSpeed
## 1 Zimmermann     Jordan 2017     29 0.372              89.3 6.08   54.0    92.2
## 2       Wood     Travis 2017     39 0.385              88.9 6.80   76.8    88.2
## 3     Wilson     Justin 2017     65 0.281              86.6 3.41   90.0    94.9
## 4 Wainwright       Adam 2017     24 0.340              86.7 5.12   69.6    88.0
## 5     Triggs     Andrew 2017     12 0.316              88.7 4.29   44.2    88.9
##   fbSpin fbBreak bbRate bbSpeed bbSpin bbBreak offSpeed offSpin offBreak
## 1   2339    19.1   39.8    84.6   2389     8.4     86.3    2245     17.6
## 2   2088    18.3   14.3    77.3   1983    13.9     81.0    1742     17.2
## 3   2439    19.9    9.8    84.8   2440     6.0     89.9    1523     20.8
## 4   2249    14.4   26.1    72.5   2643    20.6     81.4    1788     17.3
## 5   2389    16.5   55.1    79.4   2551    10.2     82.3    1903     15.7
```

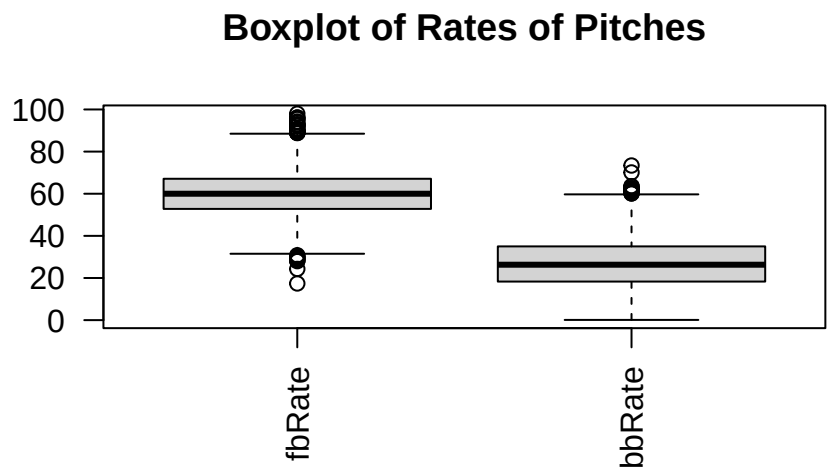
Everything seems to be in the correct format.

Univariate Analysis

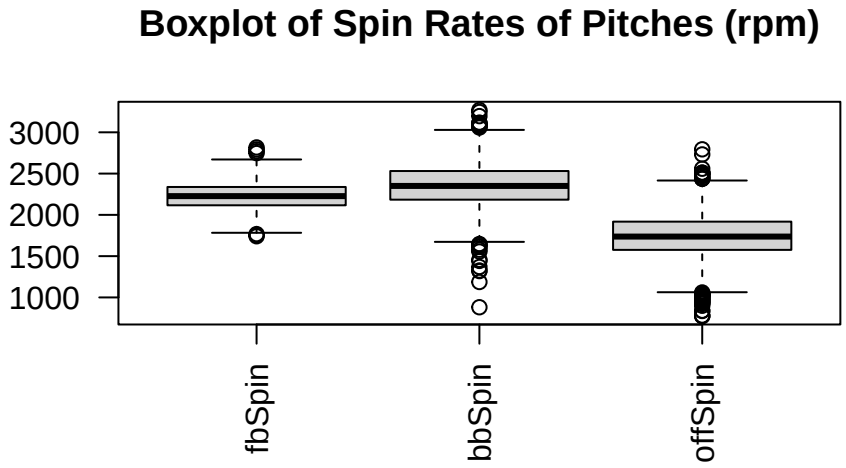
```
# A quick look at the statistics for my dataset
describe(fullPitchers)
```

```
##                    vars    n    mean     sd  median trimmed    mad     min
## last_name*            1 2498  419.11 238.89  413.00  420.68 309.86    1.00
## first_name*           2 2498  212.80 124.43  228.00  213.78 163.09    1.00
## year                  3 2498 2017.27   1.60 2017.00 2017.22   1.48 2015.00
## p_game                4 2498   33.09  19.36   30.00   31.60  19.27    4.00
## woba                  5 2498    0.32   0.04    0.32    0.32   0.04    0.17
## exit_velocity_avg     6 2498   88.24   1.57   88.30   88.26   1.48   80.40
## era                   7 2498    4.40   1.51    4.20    4.29   1.35    0.74
## fbRate                8 2498   60.32  11.13   60.00   60.07  10.67   17.40
## fbSpeed               9 2498   92.45   2.66   92.50   92.52   2.52   80.30
## fbSpin               10 2498 2226.12 155.09 2227.00 2226.26 164.57 1741.00
## fbBreak              11 2498   18.42   2.69   18.70   18.56   2.37    7.10
## bbRate               12 2498   26.83  12.22   26.30   26.52  12.31    0.10
## bbSpeed              13 2498   81.81   3.78   82.10   82.00   3.71   66.10
## bbSpin               14 2498 2356.37 269.61 2350.50 2355.84 254.27  881.00
## bbBreak              15 2498    9.99   3.96    9.30    9.73   4.00    1.90
## offSpeed             16 2498   85.06   3.17   85.30   85.23   2.97   63.00
## offSpin              17 2498 1743.75 270.23 1737.00 1744.31 249.08  772.00
## offBreak             18 2498   16.45   2.97   16.70   16.57   2.67    3.40
##                       max   range  skew kurtosis   se
## last_name*         819.00  818.00 -0.03    -1.21 4.78
## first_name*        427.00  426.00 -0.09    -1.20 2.49
## year              2020.00    5.00  0.08    -1.18 0.03
## p_game              83.00   79.00  0.62    -0.62 0.39
## woba                 0.52    0.34  0.22     0.27 0.00
## exit_velocity_avg   93.40   13.00 -0.21     0.49 0.03
## era                 12.31   11.57  0.89     1.58 0.03
## fbRate              98.00   80.60  0.19     0.25 0.22
## fbSpeed            101.10   20.80 -0.33     0.53 0.05
## fbSpin            2817.00 1076.00  0.01    -0.06 3.10
## fbBreak             25.40   18.30 -0.54     0.49 0.05
## bbRate              73.40   73.30  0.26    -0.20 0.24
## bbSpeed             92.60   26.50 -0.46     0.09 0.08
## bbSpin            3270.00 2389.00 -0.07     0.52 5.39
## bbBreak             23.60   21.70  0.57    -0.25 0.08
## offSpeed            95.20   32.20 -0.88     2.85 0.06
## offSpin           2794.00 2022.00 -0.04     0.43 5.41
## offBreak            26.30   22.90 -0.46     0.71 0.06
```

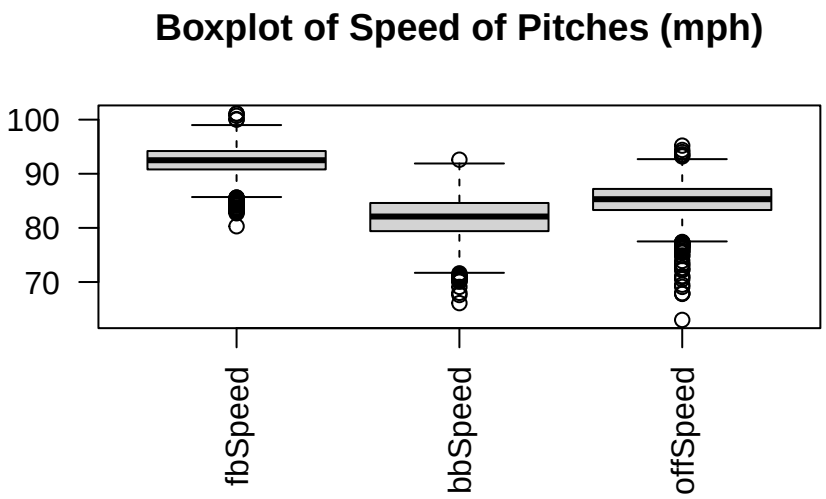
```
# Checking for Outliers
justRate <- fullPitchers %>% select(8, 12)
justSpeed <- fullPitchers %>% select(9, 13, 16)
justSpin <- fullPitchers %>% select(10, 14, 17)
justBreak <- fullPitchers %>% select(11, 15, 18)
boxplot(justRate, las = 2, main = "Boxplot of Rates of Pitches")
```



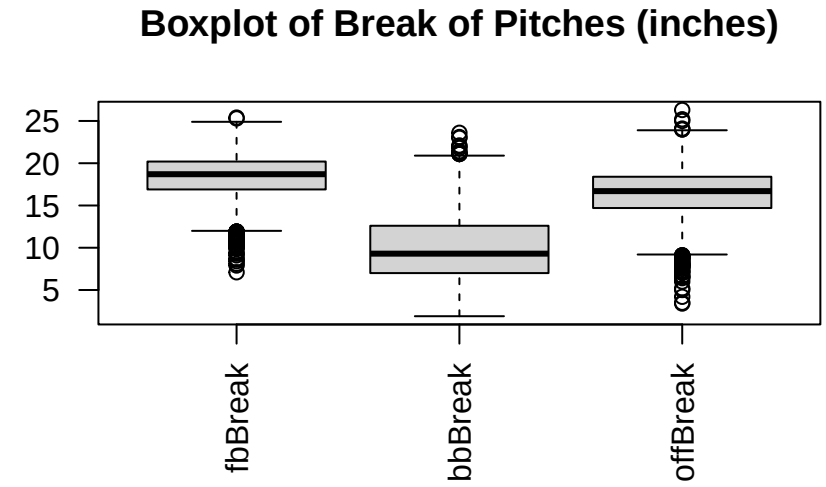

```
boxplot(justSpin, las = 2, main = "Boxplot of Spin Rates of Pitches (rpm)")
```




```
boxplot(justSpeed, las = 2, main = "Boxplot of Speed of Pitches (mph)")
```



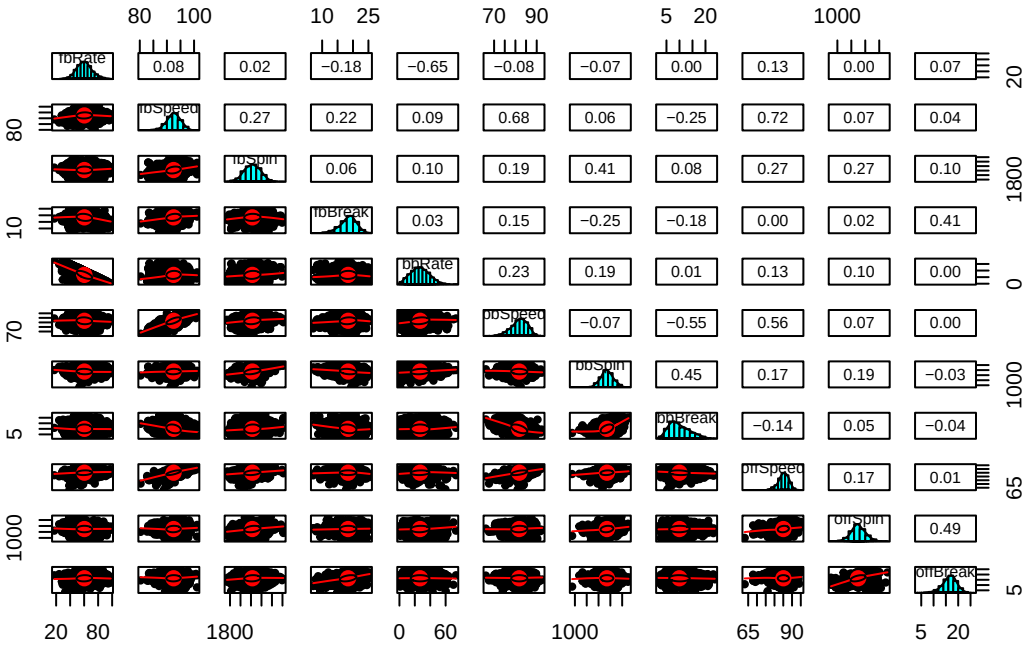

```
boxplot(justBreak, las = 2, main = "Boxplot of Break of Pitches (inches)")
```



There appear to be many univariate outliers, but that is to be expected. Baseball has always been a sport that has many outliers, particularly with a relatively low threshold for the dataset. Given that the threshold was only 100 plate appearances, this allowed for many more low-quality pitchers to be in the dataset (better pitchers tend to pitch more and a higher threshold would thus weed out worse pitchers). The downside of having a lower threshold is that there are more outliers, although I believe they should stay in the dataset.

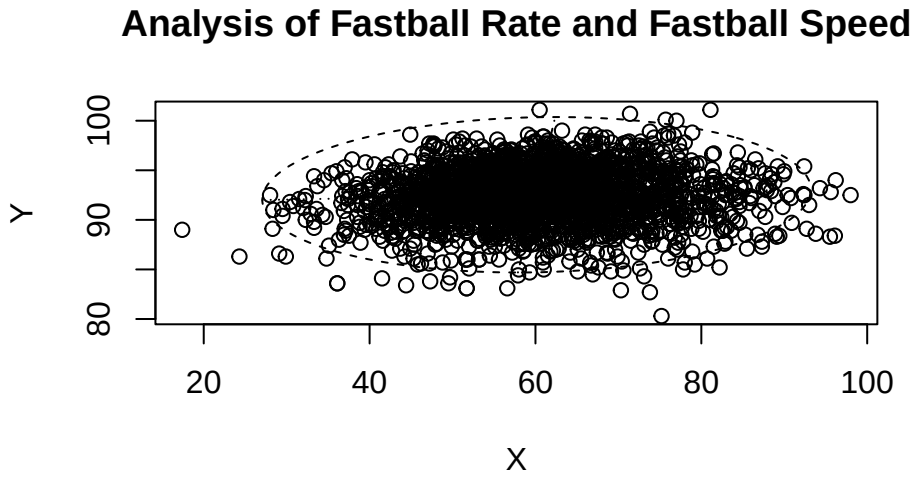
BiVariate Analysis

```
# More understandable version of GGGally
pairs.panels(fullPitchers[c(8:18)],
  density = TRUE,
  ellipses = TRUE
)
```

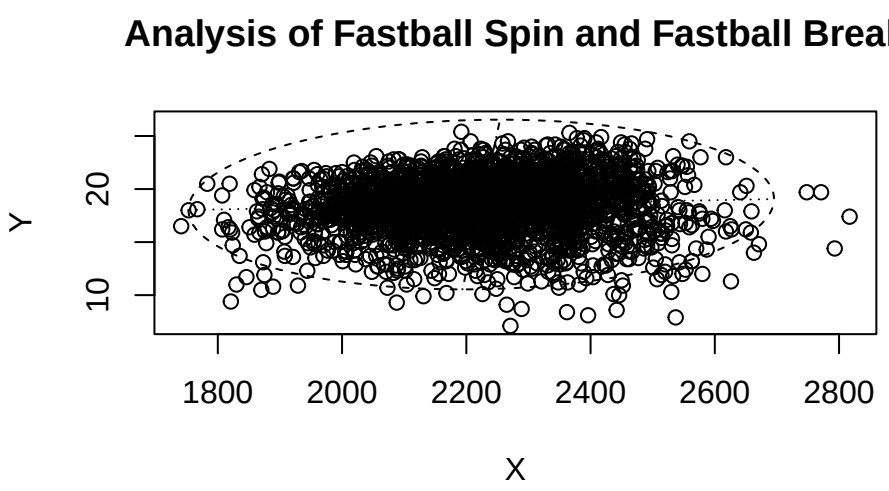


There are a wide variety of correlations, both positive and negative. Most variables seem relatively normal, although Breaking Ball Break and Breaking Ball Speed are somewhat skewed.

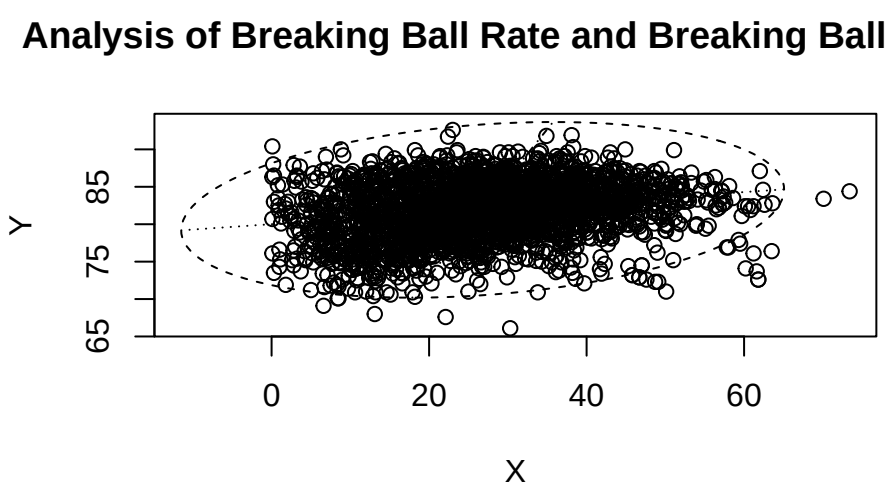
```
bvbox(fullPitchers[8:9], main = "Analysis of Fastball Rate and Fastball Speed")
```



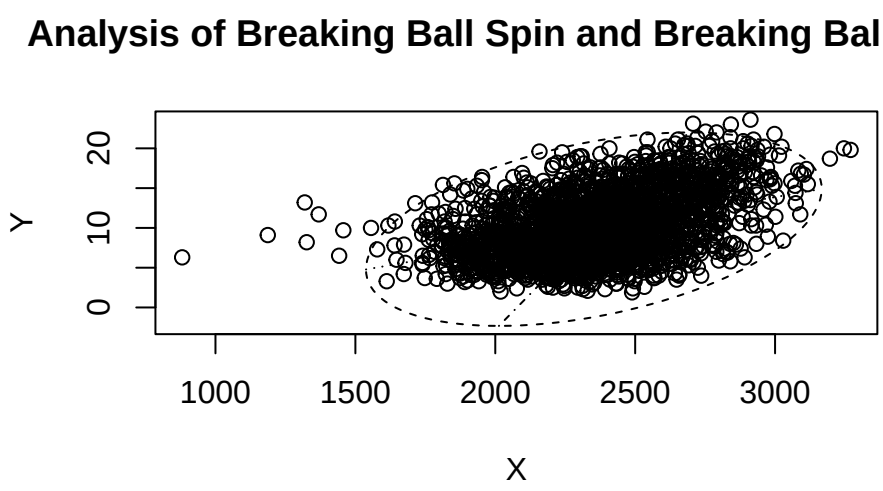

```
bvbox(fullPitchers[10:11], main = "Analysis of Fastball Spin and Fastball Break")
```



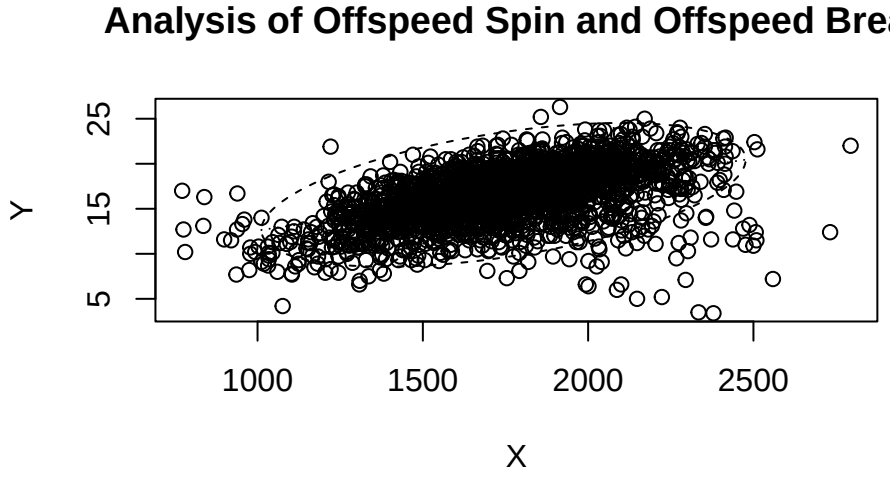

```
bvbox(fullPitchers[12:13], main = "Analysis of Breaking Ball Rate and Breaking Ball Speed")
```




```
bvbox(fullPitchers[14:15], main = "Analysis of Breaking Ball Spin and Breaking Ball Break")
```




```
bvbox(fullPitchers[17:18], main = "Analysis of Offspeed Spin and Offspeed Break")
```



There seem to be a wide variety of correlations and relationships in my data. The correlations are generally fairly low, although there are some clear relationships. There definitely are some bivariate outliers (I checked the relationships I assumed would be most meaningful), but once again, I don't believe that outliers need to be dealt with in some specific way with this data set, as the outliers are relatively expected and part of baseball. There is a subsection of pitchers who are extremely unique and removing them from the dataset would be a disservice to the overall study, as the unique players may have some commonalities that can be discovered and studied.

Methods

Explanation of Methods

PCA For my two methods, I will be using principal component analysis and factor analysis.

Principal component analysis looks at an eigendecomposition of the covariance matrix of a dataset and attempts to reduce the number of dimensions (variables) in a dataset by reexpressing a high dimensional dataset into a lower-dimensional dataset. PCA hopes to capture the maximum variances of the higher dimensional data into the lower dimensional data in as few factors as possible. The output of PCA produces loadings that indicate what each lower-dimensional factor is made up of. In this study, we will hope to use PCA to reduce the dimensions of the 11 variables that make up the dataset. This will be used to more easily have an understanding of which variables have the most influence on the dataset and to simplify the overall number of variables. A scree plot can be used to help decide how many factors are worth keeping as if there is a large dropoff, followed by a flattening, that typically indicates that is a good place to finish dimensionality reduction, as you want to find the best combination of keeping variance but reducing variables.

Clustering In my analysis, I will be using clustering to attempt to separate similar groups of baseball pitchers into distinct categories. There are many ways to cluster, but in this study, we will be using Ward's Method which is a type of agglomerative hierarchical clustering. This means that each observation starts as its own cluster and they are slowly merged into bigger clusters. This can be displayed on a dendrogram, which shows the "tree" of different clusters. Ward's method uses the smallest within-cluster sum of squares values to find the groups with the minimum variance. Clustering can also use a scree plot to determine how many clusters should the data be grouped into, as like PCA, you want to find a balance of cluster strength but also large, meaningful clusters. Finally, a cluster's strength can be measured using the silhouette coefficient.

Results

PCA Analysis

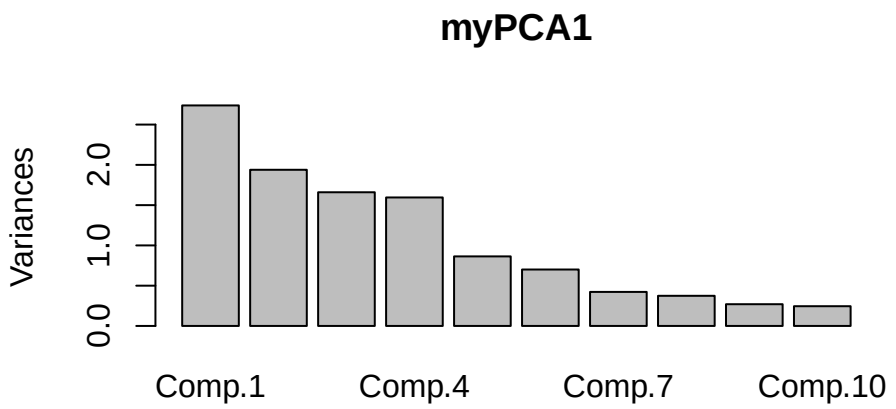
```
set.seed(123)

myPCA1 <- princomp(scale(fullPitchers[c(8:18)]), cor = TRUE, scores = TRUE)
summary(myPCA1)
```

```
## Importance of components:
##                           Comp.1   Comp.2   Comp.3   Comp.4    Comp.5    Comp.6
## Standard deviation     1.654644 1.392458 1.288270 1.262959 0.9292838 0.8369625
## Proportion of Variance 0.248895 0.176267 0.150876 0.145006 0.0785062 0.0636824
## Cumulative Proportion  0.248895 0.425162 0.576039 0.721045 0.7995509 0.8632333
##                           Comp.7    Comp.8    Comp.9   Comp.10   Comp.11
## Standard deviation     0.6500216 0.6115498 0.5194660 0.4950500 0.4393101
## Proportion of Variance 0.0384116 0.0339994 0.0245314 0.0222795 0.0175449
## Cumulative Proportion  0.9016449 0.9356443 0.9601756 0.9824551 1.0000000
```

```
unclass(loadings(myPCA1))
```

```
##              Comp.1     Comp.2     Comp.3     Comp.4     Comp.5      Comp.6
## fbRate    0.0427840  0.1519514  0.6985864  0.0295448  0.0314284  0.02283860
## fbSpeed  -0.5163811  0.0518151  0.1235094  0.1427364 -0.2081370  0.25384741
## fbSpin   -0.2575443 -0.3815854  0.1551147 -0.0363199 -0.2890210 -0.71449458
## fbBreak  -0.1766806  0.2023444 -0.1686827 -0.4769182 -0.6381038  0.06371510
## bbRate   -0.1780210 -0.2570515 -0.5980385  0.0480977  0.1595845  0.11443358
## bbSpeed  -0.5101762  0.1717790 -0.0786571  0.1463221  0.1374660 -0.10707279
## bbSpin   -0.0484858 -0.5987280  0.0835749  0.1547017 -0.1216896 -0.01566031
## bbBreak   0.2662052 -0.4765832  0.0988812  0.0238933 -0.3103698  0.46992658
## offSpeed -0.4717788 -0.0756516  0.1879728  0.2107883  0.0326406  0.38502529
## offSpin  -0.1732333 -0.3114580  0.1344318 -0.4369463  0.5496270  0.00206745
## offBreak -0.1272896 -0.0612842  0.1010111 -0.6829366  0.0795267  0.16307410
##              Comp.7     Comp.8      Comp.9    Comp.10    Comp.11
## fbRate    0.1312724  0.3465361  0.17318588  0.5565426  0.0873060
## fbSpeed  -0.0635867 -0.1250172  0.24577306 -0.2253430  0.6755898
## fbSpin   -0.2676820  0.2822506 -0.00911457 -0.1295197 -0.0216958
## fbBreak  -0.0633725 -0.2223162 -0.10288953  0.4307493 -0.1054894
## bbRate    0.0296035  0.5210370  0.08521352  0.4289365  0.2002590
## bbSpeed   0.1953225 -0.1281854  0.55611672  0.0247625 -0.5364627
## bbSpin    0.6436896 -0.3567953 -0.14474775  0.1716725  0.0337693
## bbBreak  -0.3061730  0.0804852  0.44412662 -0.0925435 -0.2616523
## offSpeed -0.1821731  0.1675841 -0.60125291 -0.0280082 -0.3450583
## offSpin  -0.3835660 -0.3920714  0.04606613  0.2295666  0.0789315
## offBreak  0.4190932  0.3629013  0.02404702 -0.4012976 -0.0395447
```

```
screeplot(myPCA1)
```



I would keep 4 Factors in my PCA, as there seems to be a steep drop-off in the proportion of variance found after the first four. This also satisfies Kaiser's Rule, which requires us to retain PCs with eigenvalues larger than 1. The first four PCs satisfy this condition, as their standard deviations (which are square roots of the eigenvalues) are also larger than 1. Overall, This is not the most effective PCA as the cumulative proportion of variance is only .72 after four variables, but it can still be useful. Some thoughts on the first four components:

The first component seems heavily weighted negatively by the three speeds. The only strong positive output would be the break on breaking balls.

The second component seems heavily weighted negatively by the three spin rates. The only positive weights of note are fastball speed, breaking ball speed, and fastball break.

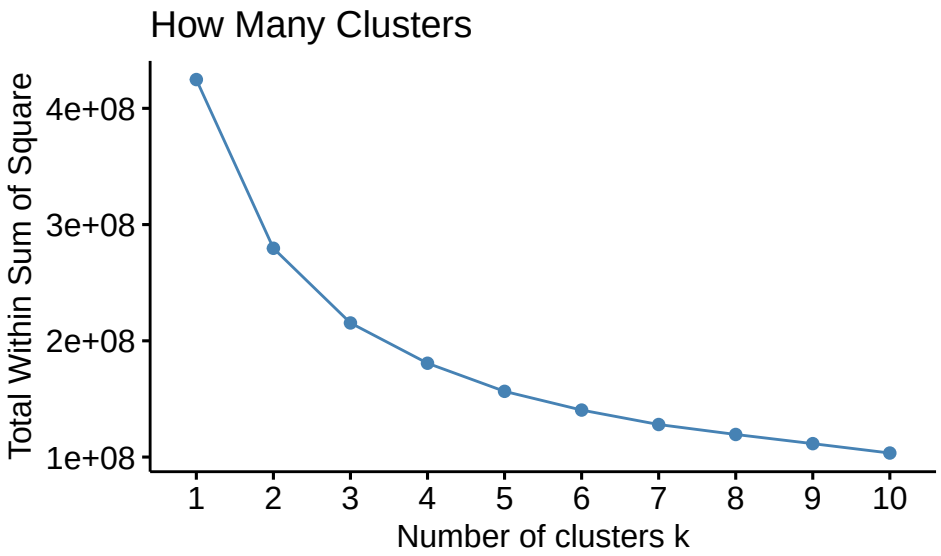
The third component has the strongest loading of the four components in fastball rate and has a large negative loading in breaking ball rate.

The fourth component has a large negative loading on fastball break, offspeed spin, and offspeed break, with a positive loading on offspeed pitch speed.

Overall, while keeping just four factors does not cover a major amount of the variance, it does seem to account for almost every variable is accounted for with a large loading (at least .38 or -.38) in one of the first four factors. This indicates that while the PCA might not be the most effective, it may be able to tell us something about our data.

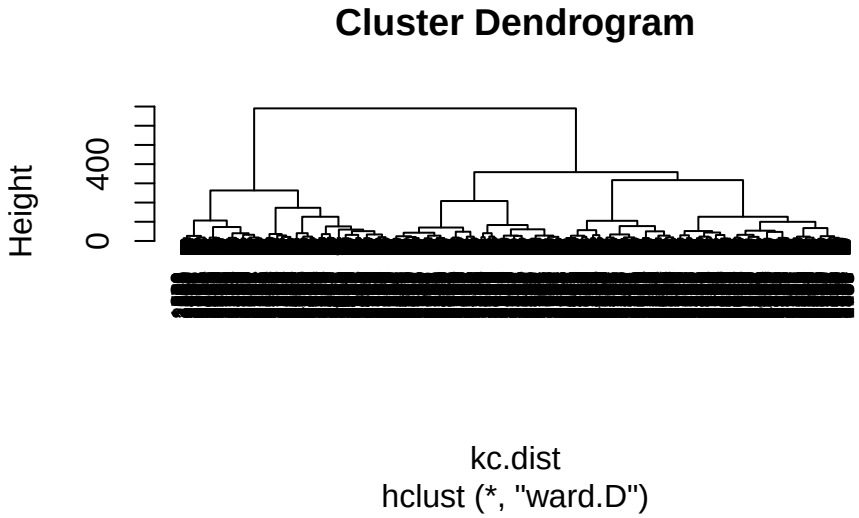
Cluster Analysis

```
fviz_nbclust(fullPitchers[c(8:18)], kmeans, method = "wss", k.max = 10) +
  ggtitle("How Many Clusters")
```



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

```
# Using Wards Method to Create my Clusters
kc.dist <- dist(scale(fullPitchers[c(8:18)]))
hcward <- hclust(kc.dist, method = "ward.D")
plot(hcward, cex = 0.7)
```




```
# Extra Wrangling
wardSol <- (cutree(hcward, h = 25))

fullPitchers <- fullPitchers %>% mutate(Cluster = wardSol)
quickPitchers <- fullPitchers %>% select(1:3, 19, 7)
(table(quickPitchers$Cluster))
```

```
##
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##  56  23  71  33  44  65  59  63  30  45  81  86  44  43  62  63  38  51  55  57
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##  66  34  38  53  44  28  69  35  27  57  51  23  48  60  54 116  35  37  11  44
##  41  42  43  44  45  46  47  48  49  50  51
##  35  98  74  46  47  44  27  19  28  36  45
```

```
cluster6 <- fullPitchers %>% filter(Cluster == 73)
```

```
cluster1 <- fullPitchers %>% filter(Cluster == 1)
cluster2 <- fullPitchers %>% filter(Cluster == 2)
cluster3 <- fullPitchers %>% filter(Cluster == 3)
cluster4 <- fullPitchers %>% filter(Cluster == 4)
cluster5 <- fullPitchers %>% filter(Cluster == 5)
cluster6 <- fullPitchers %>% filter(Cluster == 17)
```

```
# Silhouette Values
my.dist.scale <- dist(scale(fullPitchers[c(8:18)]))
attempt2 <- silhouette(fullPitchers$Cluster, my.dist.scale)
summary(attempt2)
```

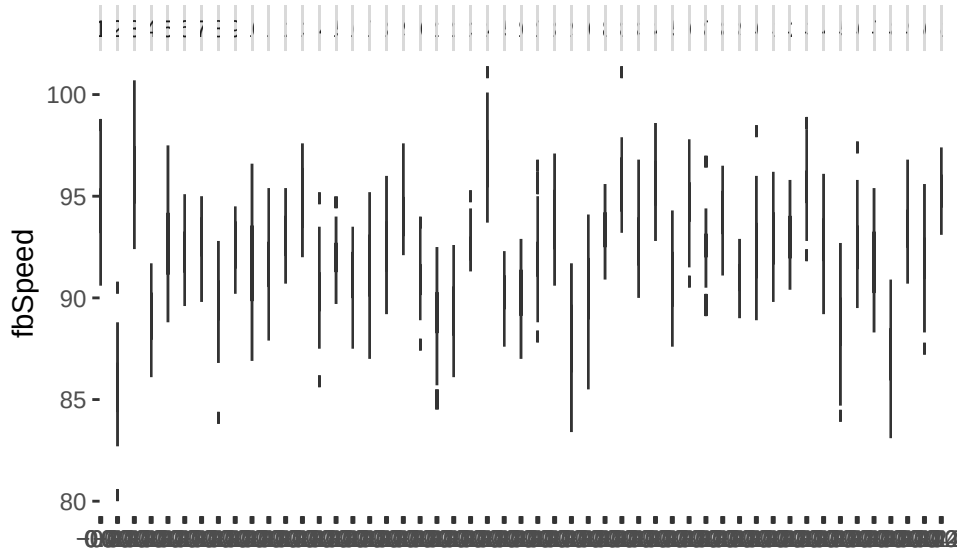
```
## Silhouette of 2498 units in 51 clusters from silhouette.default(x = fullPitchers$Cluster, dist = my.
##  Cluster sizes and average silhouette widths:
##          56          23          71          33          44          65
## -0.00104441  0.22254136 -0.05195406  0.14371489 -0.00426399  0.02529758
##          59          63          30          45          81          86
## -0.02379930  0.02675801  0.22664724  0.00306258  0.02791715  0.03737781
##          44          43          62          63          38          51
##  0.11476432  0.05953085  0.07964004  0.03423326  0.02397760  0.09657724
##          55          57          66          34          38          53
##  0.09819023  0.04931594 -0.00212525 -0.00988848  0.17079329 -0.00329828
##          44          28          69          35          27          57
##  0.03156512  0.01937360  0.06199329  0.05800370  0.03735461  0.02345015
##          51          23          48          60          54         116
##  0.06214978  0.12817172  0.09592537  0.09605548  0.01015752 -0.01559301
##          35          37          11          44          35          98
##  0.08824019  0.06850277  0.20188512 -0.02733575  0.08257264  0.02848377
##          74          46          47          44          27          19
##  0.02280305  0.04218442  0.07076686 -0.06541674  0.06493481  0.20585319
##          28          36          45
##  0.06196591  0.06611387  0.05565038
## Individual silhouette widths:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.3241 -0.0312  0.0492  0.0438  0.1231  0.4103
```

```
summary(attempt2)$avg.width
```

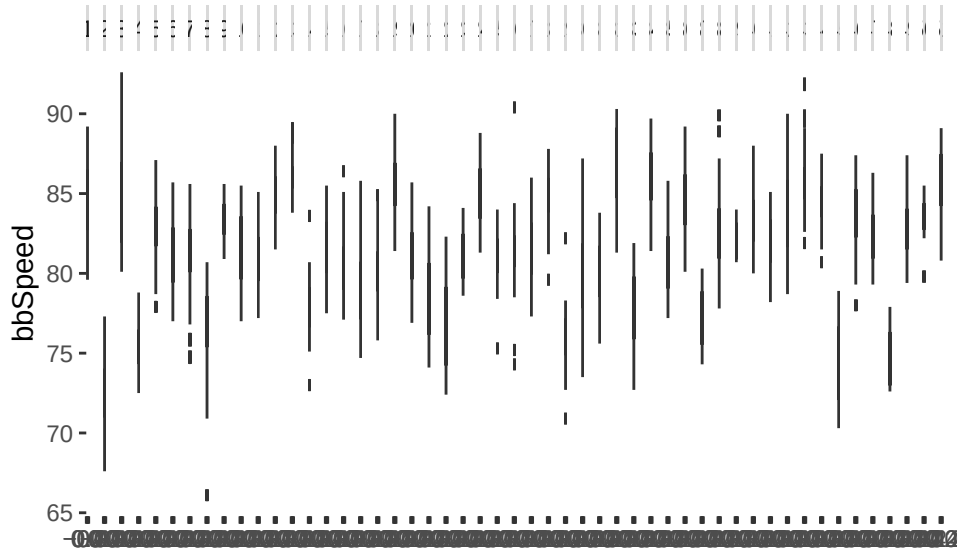
```
## [1] 0.0437739
```

Very low silhouette values. Does not indicate strong clusters.

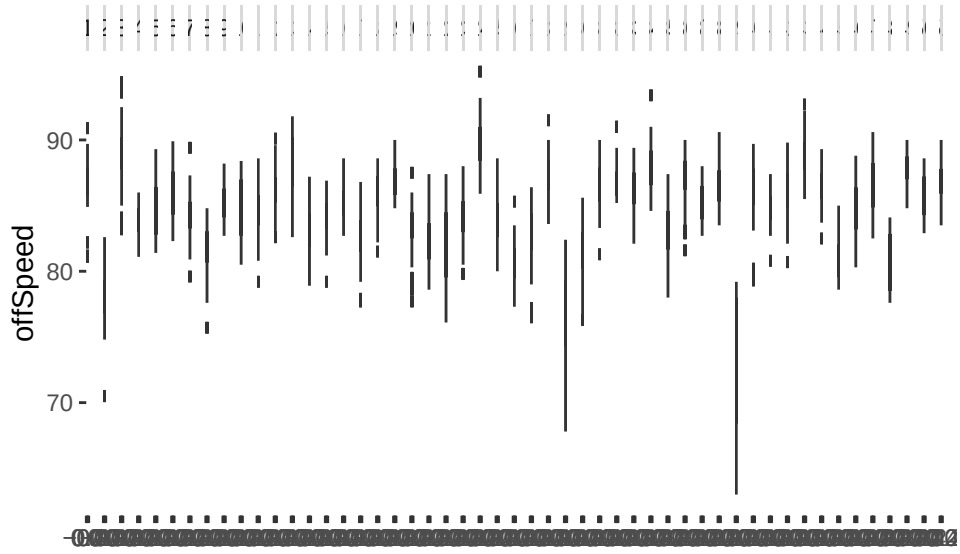
```
ggplot(fullPitchers, aes(y = fbSpeed)) +
  geom_boxplot() +
  facet_grid(. ~ Cluster)
```



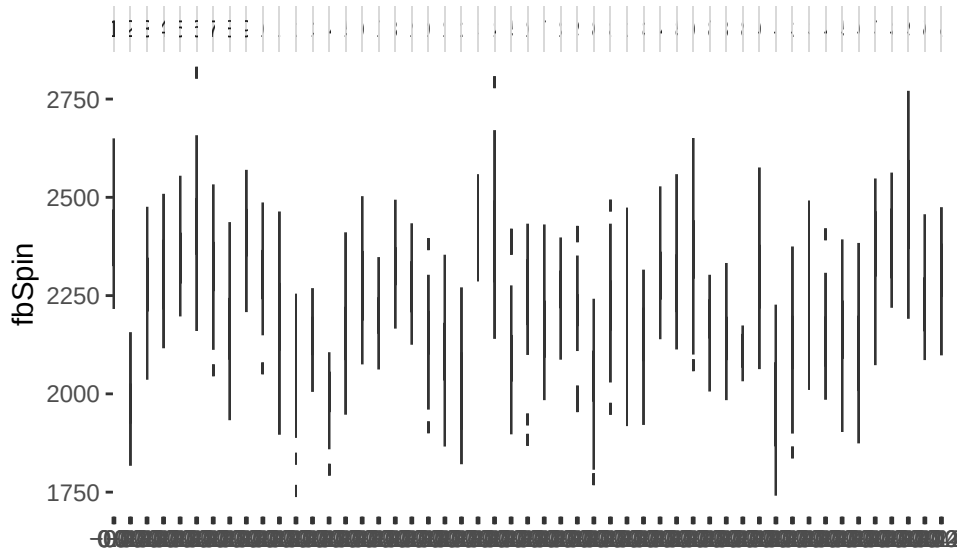

```
ggplot(fullPitchers, aes(y = bbSpeed)) +
  geom_boxplot() +
  facet_grid(. ~ Cluster)
```



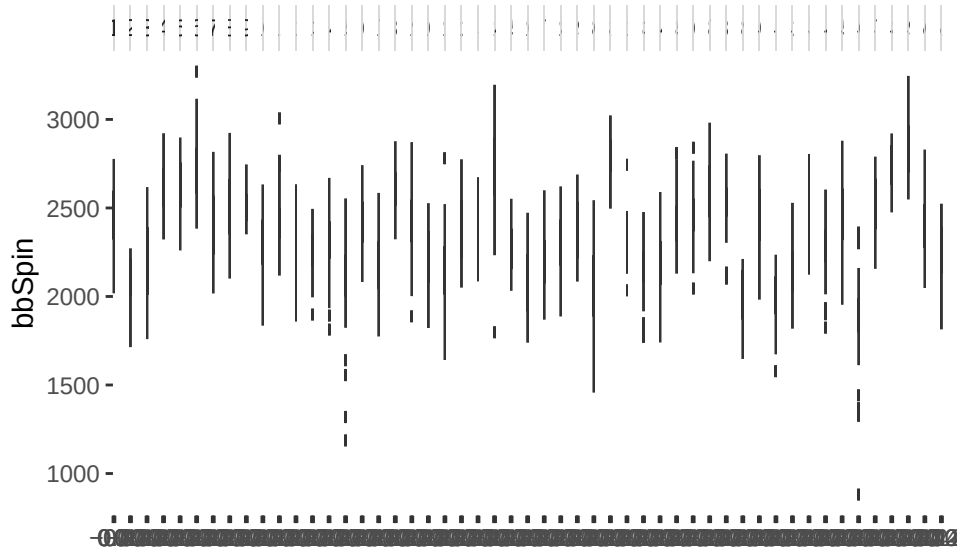

```
ggplot(fullPitchers, aes(y = offSpeed)) +
  geom_boxplot() +
  facet_grid(. ~ Cluster)
```



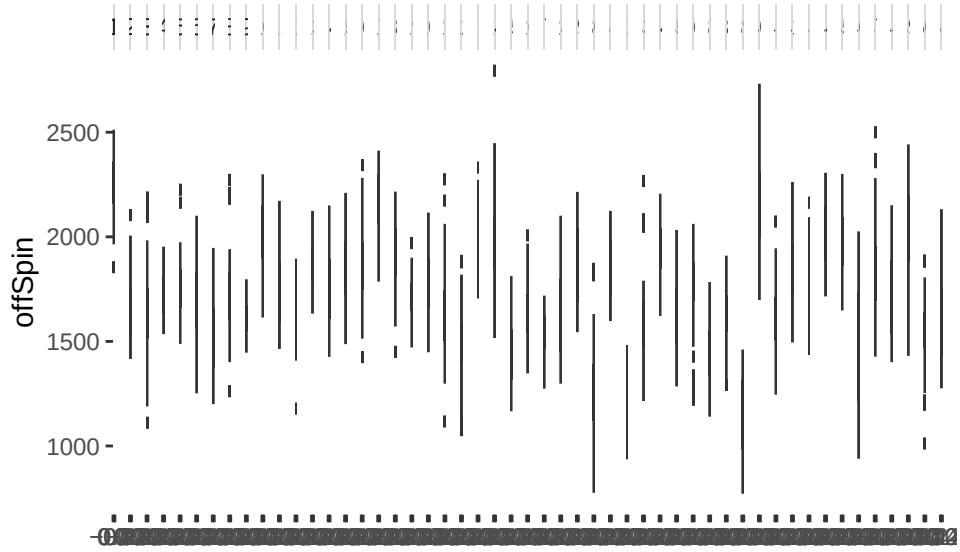

```
ggplot(fullPitchers, aes(y = fbSpin)) +
  geom_boxplot() +
  facet_grid(. ~ Cluster)
```




```
ggplot(fullPitchers, aes(y = bbSpin)) +
  geom_boxplot() +
  facet_grid(. ~ Cluster)
```




```
ggplot(fullPitchers, aes(y = offSpin)) +
  geom_boxplot() +
  facet_grid(. ~ Cluster)
```



Some quick generalizations about each cluster:

1: Fast throwing. High spin rates.

2: Slow throwing. Low spin rates.

3: Fast throwing. Low spin rates.

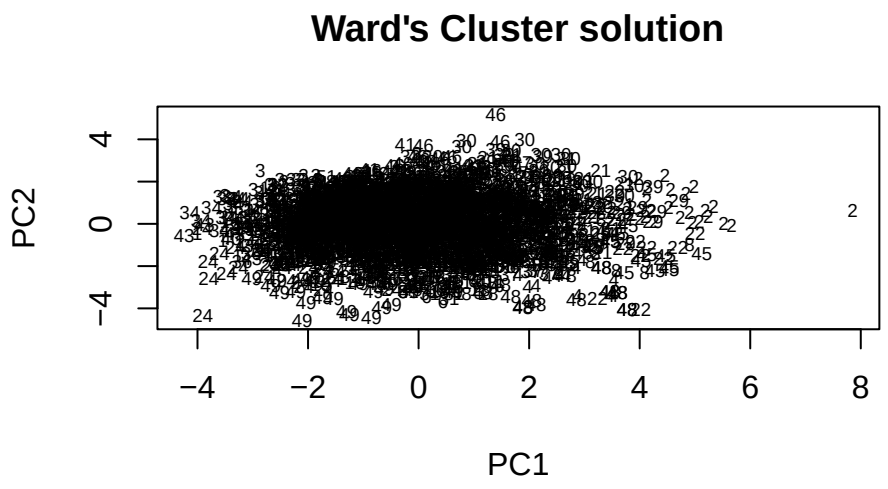
4: Mid-Fast Throwing. Mid-High spin rates.

5: Slow throwing. Mid-High spin rates.

6: Slow throwing. Mid-High spin rates.

More Analysis

```
# for clearer graph
plot(myPCA1$scores[, 1:2],
  type = "n", xlab = "PC1", ylab = "PC2",
  main = "Ward's Cluster solution"
)


text(myPCA1$scores[, 1:2], labels = wardSol, cex = 0.6)
```

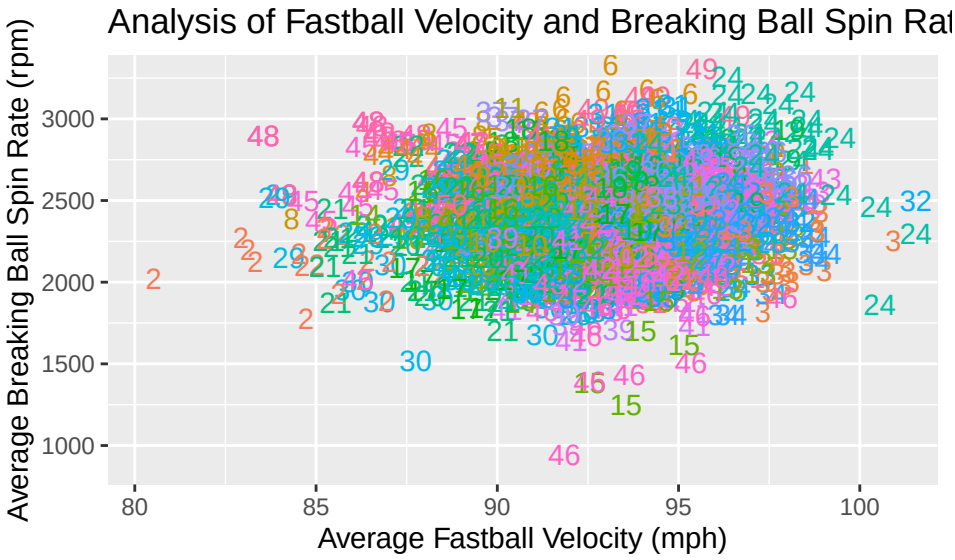



```
loadings(myPCA1)
```

```
##
## Loadings:
##          Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9 Comp.10
## fbRate            0.152  0.699                        0.131  0.347  0.173  0.557
## fbSpeed  -0.516         0.124  0.143 -0.208  0.254        -0.125  0.246 -0.225
## fbSpin   -0.258 -0.382  0.155        -0.289 -0.714 -0.268  0.282        -0.130
## fbBreak  -0.177  0.202 -0.169 -0.477 -0.638                -0.222 -0.103  0.431
## bbRate   -0.178 -0.257 -0.598         0.160  0.114         0.521         0.429
## bbSpeed  -0.510  0.172         0.146  0.137 -0.107  0.195 -0.128  0.556
## bbSpin          -0.599         0.155 -0.122         0.644 -0.357 -0.145  0.172
## bbBreak   0.266 -0.477                -0.310  0.470 -0.306         0.444
## offSpeed -0.472         0.188  0.211         0.385 -0.182  0.168 -0.601
## offSpin  -0.173 -0.311  0.134 -0.437  0.550        -0.384 -0.392         0.230
## offBreak -0.127         0.101 -0.683         0.163  0.419  0.363        -0.401
##          Comp.11
## fbRate
## fbSpeed   0.676
## fbSpin
## fbBreak  -0.105
## bbRate    0.200
```

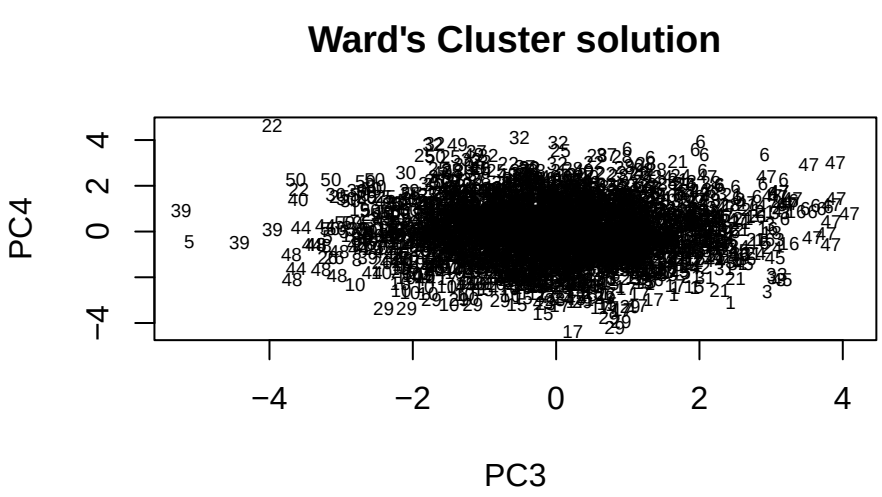
```
## bbSpeed  -0.536
## bbSpin
## bbBreak  -0.262
## offSpeed -0.345
## offSpin
## offBreak
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.091  0.091  0.091  0.091  0.091  0.091  0.091  0.091  0.091
## Cumulative Var  0.091  0.182  0.273  0.364  0.455  0.545  0.636  0.727  0.818
##                Comp.10 Comp.11
## SS loadings      1.000   1.000
## Proportion Var   0.091   0.091
## Cumulative Var   0.909   1.000
```

```
ggplot(fullPitchers, aes(x = fbSpeed, y = bbSpin,label=as.factor(Cluster), col = as.factor(Cluster) ))+g
```



The clusters and PCA seem to produce similar results. You can clearly see groupings of each cluster, although cluster five seems to get lost in the middle group.

```
# for clearer graph
plot(myPCA1$scores[, 3:4], type = "n", xlab = "PC3", ylab = "PC4", main = "Ward's Cluster solution")
text(myPCA1$scores[, 3:4], labels = wardSol, cex = 0.6)
```



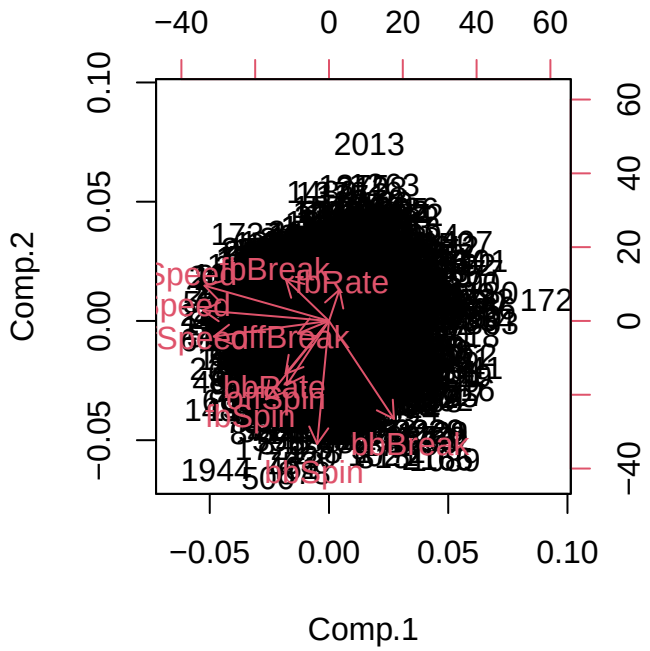

```
loadings(myPCA1)
```

```
##
## Loadings:
##          Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9 Comp.10
## fbRate            0.152  0.699                        0.131  0.347  0.173  0.557
## fbSpeed  -0.516         0.124  0.143 -0.208  0.254        -0.125  0.246 -0.225
## fbSpin   -0.258 -0.382  0.155        -0.289 -0.714 -0.268  0.282        -0.130
## fbBreak  -0.177  0.202 -0.169 -0.477 -0.638                -0.222 -0.103  0.431
## bbRate   -0.178 -0.257 -0.598         0.160  0.114         0.521         0.429
## bbSpeed  -0.510  0.172         0.146  0.137 -0.107  0.195 -0.128  0.556
## bbSpin           -0.599         0.155 -0.122         0.644 -0.357 -0.145  0.172
## bbBreak   0.266 -0.477                -0.310  0.470 -0.306         0.444
## offSpeed -0.472         0.188  0.211         0.385 -0.182  0.168 -0.601
## offSpin  -0.173 -0.311  0.134 -0.437  0.550        -0.384 -0.392         0.230
## offBreak -0.127         0.101 -0.683         0.163  0.419  0.363        -0.401
##          Comp.11
## fbRate
## fbSpeed   0.676
## fbSpin
## fbBreak  -0.105
## bbRate    0.200
## bbSpeed  -0.536
## bbSpin
## bbBreak  -0.262
## offSpeed -0.345
## offSpin
## offBreak
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.091  0.091  0.091  0.091  0.091  0.091  0.091  0.091  0.091
## Cumulative Var  0.091  0.182  0.273  0.364  0.455  0.545  0.636  0.727  0.818
##                Comp.10 Comp.11
```

```
## SS loadings      1.000   1.000
## Proportion Var   0.091   0.091
## Cumulative Var   0.909   1.000
```

An analysis of the third and fourth components is less clear. Clusters three and two seem to be more scattered, whereas one, four, five, and six seem fairly well defined. Perhaps a deeper analysis of the clusters will help to explain why this is true.

```
# Labels are player ID/row number
biplot(myPCA1)
```



Fastball rate stands out as having an opposite direction to many of the other variables.

```
favstats(era ~ wardSol, data = fullPitchers)
```

```
##    wardSol  min     Q1 median     Q3   max    mean      sd   n missing
## 1        1 1.40 3.0425  3.775 4.9925  7.33 4.03232 1.35494  56       0
## 2        2 1.85 3.7800  4.500 5.3400  8.14 4.58696 1.41591  23       0
## 3        3 1.60 2.7750  3.640 4.3200  9.90 3.80944 1.43997  71       0
## 4        4 1.61 3.9100  4.470 5.1200  7.79 4.61030 1.25324  33       0
## 5        5 1.63 3.2925  4.155 4.9150  7.77 4.24227 1.38676  44       0
## 6        6 1.65 3.2300  3.900 5.2900  9.38 4.21231 1.57878  65       0
## 7        7 2.15 3.7700  4.340 5.4450  9.55 4.69712 1.31607  59       0
## 8        8 1.97 3.3950  4.400 5.3050  8.24 4.47810 1.39006  63       0
## 9        9 1.80 3.4825  4.685 5.6350 10.00 4.82733 1.99545  30       0
## 10      10 1.69 2.8100  3.670 4.7100  7.50 4.00422 1.51666  45       0
## 11      11 1.86 3.2800  4.150 5.0800  9.90 4.52975 1.67815  81       0
## 12      12 1.60 3.7650  4.615 5.6575  8.93 4.81756 1.50851  86       0
```

```
## 13      13 1.67 3.2675  4.060 4.7575  8.63 4.18477 1.29028  44       0
## 14      14 2.03 3.7650  4.180 5.0800  8.05 4.44326 1.38367  43       0
## 15      15 2.18 3.8550  4.495 5.9250  9.31 4.81419 1.45860  62       0
## 16      16 2.34 3.7150  4.290 5.5850  9.64 4.76190 1.55492  63       0
## 17      17 2.16 3.9350  4.335 5.4275  9.83 4.70053 1.47077  38       0
## 18      18 2.00 3.4250  4.410 5.4750  8.59 4.51569 1.47805  51       0
## 19      19 2.31 3.4400  4.340 5.0900  8.60 4.47036 1.36072  55       0
## 20      20 1.62 3.9500  4.690 5.6300 10.65 4.88719 1.57142  57       0
## 21      21 1.54 3.5425  4.345 5.2375  7.86 4.41939 1.31065  66       0
## 22      22 1.97 3.6750  4.300 4.9750  7.79 4.31118 1.38266  34       0
## 23      23 2.34 3.6150  4.495 5.3675  7.22 4.54158 1.13123  38       0
## 24      24 1.55 2.6500  3.590 4.2900  9.00 3.81566 1.72349  53       0
## 25      25 1.43 3.7825  4.745 5.6825  8.21 4.83795 1.57904  44       0
## 26      26 1.85 3.1075  3.840 4.7075  8.06 4.06714 1.39251  28       0
## 27      27 2.20 3.9500  4.910 5.9700  9.00 5.06638 1.48023  69       0
## 28      28 2.28 3.3650  3.970 5.0800 10.94 4.41857 1.70926  35       0
## 29      29 2.52 3.7300  4.640 5.6950 11.58 4.94704 1.86966  27       0
## 30      30 1.23 3.6400  4.570 5.9300  9.21 4.78789 1.56104  57       0
## 31      31 2.52 3.2300  3.960 4.8500  8.59 4.17588 1.29943  51       0
## 32      32 1.50 3.5450  4.090 5.2650  7.05 4.32087 1.34105  23       0
## 33      33 2.07 3.6100  4.155 4.8925  8.06 4.33188 1.15441  48       0
## 34      34 0.74 2.9900  4.210 5.6550  7.30 4.29700 1.54934  60       0
## 35      35 2.44 3.4250  4.515 5.6425 10.13 4.69815 1.52936  54       0
## 36      36 0.79 2.7325  3.755 4.8425  7.71 3.90914 1.43635 116       0
## 37      37 2.25 3.2650  4.030 4.7650  8.96 4.26371 1.51838  35       0
## 38      38 1.30 3.5900  4.320 5.1500  7.69 4.36757 1.37146  37       0
## 39      39 1.59 2.3050  3.650 4.8300  6.41 3.73455 1.64869  11       0
## 40      40 2.32 3.2925  4.295 4.9825  8.30 4.33545 1.41756  44       0
## 41      41 1.96 3.8400  4.630 5.7350  7.88 4.74029 1.34064  35       0
## 42      42 1.96 3.4175  4.205 5.3200  8.63 4.49663 1.48221  98       0
## 43      43 1.63 2.8625  3.625 4.5600  7.32 3.81054 1.26359  74       0
## 44      44 0.92 3.4225  4.495 4.9975  9.85 4.47891 1.54801  46       0
## 45      45 2.13 3.2800  3.920 4.4500 12.03 4.28021 1.97977  47       0
## 46      46 1.21 3.3100  4.330 5.0650  9.00 4.25568 1.62388  44       0
## 47      47 1.92 3.1000  3.720 5.2700  8.74 4.32704 1.79032  27       0
## 48      48 1.55 3.0250  4.160 5.1000  6.06 4.01368 1.36500  19       0
## 49      49 1.57 2.5450  3.660 4.8125  6.95 3.80571 1.40787  28       0
## 50      50 1.92 3.4775  3.780 4.8025 12.31 4.48694 1.94171  36       0
## 51      51 1.20 3.0800  3.870 4.8900  7.41 4.05267 1.30843  45       0
```

```
favstats(woba ~ wardSol, data = fullPitchers)
```

```
##    wardSol   min      Q1 median      Q3   max     mean        sd   n missing
## 1        1 0.235 0.27775 0.3000 0.33125 0.392 0.308732 0.0403117  56       0
## 2        2 0.231 0.30900 0.3260 0.34250 0.385 0.327609 0.0345724  23       0
## 3        3 0.172 0.27400 0.3030 0.32600 0.430 0.301761 0.0461230  71       0
## 4        4 0.258 0.30800 0.3280 0.34600 0.396 0.326333 0.0339344  33       0
## 5        5 0.218 0.28575 0.3125 0.33000 0.416 0.314682 0.0432219  44       0
## 6        6 0.223 0.28100 0.3100 0.33800 0.440 0.310262 0.0448478  65       0
## 7        7 0.240 0.30600 0.3300 0.35000 0.397 0.327305 0.0362745  59       0
## 8        8 0.237 0.29550 0.3200 0.34300 0.417 0.323762 0.0396520  63       0
## 9        9 0.243 0.30525 0.3360 0.35650 0.481 0.336500 0.0588790  30       0
## 10      10 0.204 0.27200 0.2990 0.33800 0.419 0.307178 0.0527715  45       0
## 11      11 0.222 0.29100 0.3140 0.33800 0.465 0.320901 0.0446600  81       0
```

```
## 12      12 0.210 0.30825 0.3305 0.35475 0.433 0.330547 0.0406325  86       0
## 13      13 0.249 0.29775 0.3100 0.34050 0.397 0.317159 0.0340854  44       0
## 14      14 0.254 0.30700 0.3230 0.36600 0.432 0.331116 0.0392636  43       0
## 15      15 0.243 0.30975 0.3335 0.36150 0.430 0.336903 0.0399589  62       0
## 16      16 0.253 0.30300 0.3240 0.35350 0.449 0.331778 0.0400936  63       0
## 17      17 0.232 0.30825 0.3235 0.34700 0.432 0.330289 0.0398869  38       0
## 18      18 0.230 0.29150 0.3220 0.34400 0.416 0.320569 0.0451576  51       0
## 19      19 0.213 0.28850 0.3140 0.33450 0.422 0.315200 0.0437888  55       0
## 20      20 0.247 0.30000 0.3350 0.35500 0.459 0.333018 0.0478664  57       0
## 21      21 0.204 0.30325 0.3230 0.34675 0.396 0.324394 0.0371756  66       0
## 22      22 0.245 0.30600 0.3265 0.34250 0.409 0.322382 0.0354179  34       0
## 23      23 0.247 0.30100 0.3200 0.33950 0.450 0.322500 0.0387045  38       0
## 24      24 0.203 0.25900 0.2950 0.32000 0.440 0.296340 0.0527384  53       0
## 25      25 0.239 0.30325 0.3235 0.35725 0.426 0.330205 0.0410335  44       0
## 26      26 0.241 0.28075 0.3090 0.35475 0.385 0.310964 0.0452839  28       0
## 27      27 0.251 0.31800 0.3340 0.36700 0.416 0.337522 0.0387748  69       0
## 28      28 0.242 0.29100 0.3050 0.33600 0.422 0.314486 0.0374373  35       0
## 29      29 0.252 0.30150 0.3330 0.36050 0.516 0.338000 0.0558790  27       0
## 30      30 0.224 0.30700 0.3320 0.36200 0.475 0.332070 0.0482104  57       0
## 31      31 0.237 0.28850 0.3090 0.33200 0.406 0.311275 0.0366797  51       0
## 32      32 0.231 0.27950 0.3250 0.34500 0.381 0.312783 0.0416194  23       0
## 33      33 0.219 0.29375 0.3145 0.33450 0.401 0.316042 0.0349364  48       0
## 34      34 0.225 0.27700 0.3115 0.34300 0.407 0.314250 0.0438339  60       0
## 35      35 0.256 0.30000 0.3265 0.35375 0.414 0.329574 0.0385915  54       0
## 36      36 0.187 0.26775 0.3015 0.33200 0.405 0.302931 0.0454559 116       0
## 37      37 0.240 0.27650 0.3040 0.34800 0.416 0.311886 0.0433120  35       0
## 38      38 0.232 0.30100 0.3280 0.34200 0.418 0.321243 0.0395814  37       0
## 39      39 0.230 0.26150 0.2870 0.30600 0.399 0.292091 0.0477775  11       0
## 40      40 0.232 0.27425 0.3105 0.33425 0.427 0.311955 0.0455151  44       0
## 41      41 0.231 0.31100 0.3330 0.36250 0.396 0.332657 0.0386226  35       0
## 42      42 0.228 0.30175 0.3230 0.35250 0.453 0.326806 0.0436272  98       0
## 43      43 0.209 0.26200 0.2965 0.32725 0.392 0.296203 0.0433634  74       0
## 44      44 0.199 0.30350 0.3275 0.35275 0.414 0.328348 0.0428927  46       0
## 45      45 0.240 0.28300 0.3090 0.32450 0.464 0.313191 0.0488372  47       0
## 46      46 0.236 0.30175 0.3210 0.35250 0.430 0.322727 0.0394599  44       0
## 47      47 0.235 0.28100 0.3010 0.33350 0.376 0.306889 0.0378147  27       0
## 48      48 0.188 0.27550 0.2960 0.34200 0.388 0.303211 0.0517640  19       0
## 49      49 0.205 0.25650 0.2820 0.32850 0.403 0.289679 0.0483276  28       0
## 50      50 0.215 0.28775 0.3135 0.34125 0.444 0.314750 0.0482620  36       0
## 51      51 0.222 0.27700 0.3040 0.33500 0.416 0.305711 0.0422802  45       0
```

```
favstats(exit_velocity_avg ~ wardSol, data = fullPitchers)
```

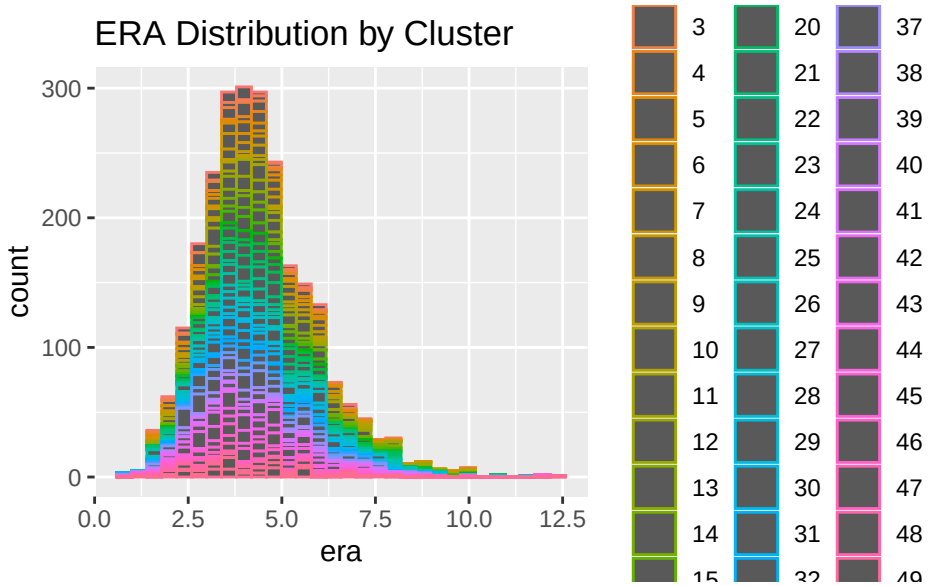
```
##    wardSol  min     Q1 median     Q3  max    mean      sd   n missing
## 1        1 84.1 86.775  87.95 88.825 91.6 87.7304 1.70240  56       0
## 2        2 84.5 86.600  87.40 88.200 90.1 87.3826 1.35434  23       0
## 3        3 82.5 86.950  88.10 89.600 91.8 88.1775 1.98128  71       0
## 4        4 85.3 87.000  88.00 88.900 93.4 88.1758 1.76653  33       0
## 5        5 83.3 87.000  88.20 89.000 91.6 88.0455 1.50495  44       0
## 6        6 85.2 87.100  88.40 89.400 92.3 88.2723 1.53465  65       0
## 7        7 86.4 87.600  88.50 89.100 91.3 88.4525 1.09203  59       0
## 8        8 82.7 86.850  87.80 88.900 91.8 87.7825 1.87096  63       0
## 9        9 85.2 87.900  89.05 90.350 91.4 88.9167 1.64340  30       0
## 10      10 85.1 86.800  88.30 89.000 91.9 88.0400 1.68811  45       0
```

```
## 11      11 84.0 87.000  88.30 89.700 92.8 88.3049 1.80249  81       0
## 12      12 83.4 87.700  88.35 89.375 92.5 88.4465 1.47391  86       0
## 13      13 85.2 87.200  88.10 88.800 91.1 88.0909 1.37871  44       0
## 14      14 80.4 87.050  88.40 89.400 90.9 88.0791 1.86401  43       0
## 15      15 85.6 87.400  88.30 89.575 91.0 88.4000 1.27228  62       0
## 16      16 85.7 87.500  88.80 89.350 91.1 88.4286 1.36906  63       0
## 17      17 85.5 87.350  88.40 88.975 91.5 88.3421 1.39044  38       0
## 18      18 85.2 87.450  88.30 89.150 91.1 88.2902 1.28456  51       0
## 19      19 84.5 87.200  88.40 89.550 90.8 88.2782 1.45331  55       0
## 20      20 84.3 87.400  88.50 89.200 91.6 88.4000 1.45627  57       0
## 21      21 84.4 87.825  88.45 89.200 90.7 88.2939 1.46727  66       0
## 22      22 84.6 86.650  87.75 88.975 90.7 87.6559 1.66099  34       0
## 23      23 85.6 87.900  88.70 89.500 92.5 88.8737 1.56476  38       0
## 24      24 83.9 86.900  87.90 89.600 91.6 88.0698 1.91096  53       0
## 25      25 84.7 87.975  88.70 89.325 91.9 88.6250 1.38045  44       0
## 26      26 85.5 86.375  87.45 88.525 92.4 87.6536 1.57703  28       0
## 27      27 85.9 87.800  88.40 89.400 92.3 88.6304 1.41829  69       0
## 28      28 84.5 86.750  88.40 89.550 92.2 88.1343 1.88414  35       0
## 29      29 85.6 87.100  87.70 88.750 91.7 87.9037 1.28826  27       0
## 30      30 84.8 87.900  88.80 89.700 93.1 88.7596 1.70544  57       0
## 31      31 85.4 87.300  88.50 89.300 90.1 88.3098 1.20071  51       0
## 32      32 85.4 87.050  88.30 89.450 91.7 88.3957 1.58472  23       0
## 33      33 81.7 87.700  88.55 89.525 91.2 88.4750 1.71644  48       0
## 34      34 85.7 87.475  88.50 89.425 92.1 88.4350 1.31482  60       0
## 35      35 84.6 87.100  87.90 89.350 90.8 88.1481 1.52992  54       0
## 36      36 83.2 87.200  88.10 89.000 91.1 88.0276 1.35207 116       0
## 37      37 83.4 87.000  87.90 89.200 91.1 87.9857 1.73258  35       0
## 38      38 83.4 86.700  87.80 89.100 91.3 87.9649 1.75439  37       0
## 39      39 84.7 85.700  86.40 86.850 88.1 86.3909 1.06344  11       0
## 40      40 85.5 87.700  88.35 89.200 91.2 88.2591 1.35429  44       0
## 41      41 86.2 88.000  89.00 90.050 91.1 88.9429 1.34957  35       0
## 42      42 85.3 87.525  88.65 89.400 92.7 88.6143 1.50928  98       0
## 43      43 85.5 87.625  88.50 89.300 91.8 88.4689 1.41885  74       0
## 44      44 85.6 87.625  88.45 89.275 91.2 88.4152 1.21215  46       0
## 45      45 82.6 86.550  87.50 88.300 92.2 87.3723 1.59083  47       0
## 46      46 84.4 87.650  88.30 89.725 92.4 88.4273 1.67809  44       0
## 47      47 85.5 86.900  88.20 89.150 90.6 88.0000 1.42559  27       0
## 48      48 84.5 86.200  86.70 88.200 90.0 87.1895 1.77166  19       0
## 49      49 83.6 86.175  87.25 88.625 91.4 87.4643 1.79551  28       0
## 50      50 85.3 87.375  88.60 89.325 91.5 88.4611 1.43552  36       0
## 51      51 84.8 87.200  88.20 89.200 91.6 88.1733 1.39698  45       0
```

The clusters are of somewhat uneven sizes (cluster 3 is about twice as big as 1,4 and,5). There are pretty sizable gaps in ERA between the clusters. Clusters one and four are dramatically better than the other clusters (a low era is good). This is also true of woba (low is once again good), where clusters one and four are the best. It's important to note however that all clusters are not clearly differentiable. The best/Q1 of cluster six (which appears to have the worst pitchers on average) are still better than the average of cluster 1. There seems to be very little change in exit velocity allowed between the different clusters.

```
# A look to see how era was distributed by cluster.
ggplot(data = fullPitchers, aes(x = era, col = as.factor(wardSol))) +
  geom_histogram() +
  ggtitle("ERA Distribution by Cluster")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
describe(cluster1[5:18])
```

More Detailed Thoughts On Clusters

```
##                    vars  n    mean     sd  median trimmed    mad     min     max
## woba                  1 56    0.31   0.04    0.30    0.31   0.04    0.23    0.39
## exit_velocity_avg     2 56   87.73   1.70   87.95   87.77   1.48   84.10   91.60
## era                   3 56    4.03   1.35    3.78    3.96   1.19    1.40    7.33
## fbRate                4 56   59.98   8.15   59.75   59.69   8.67   44.50   77.30
## fbSpeed               5 56   94.30   1.67   93.85   94.17   1.63   90.60   98.50
## fbSpin                6 56 2395.41 105.77 2396.00 2392.83 102.30 2216.00 2650.00
## fbBreak               7 56   19.23   2.23   19.05   19.21   2.15   14.30   24.30
## bbRate                8 56   27.99  10.51   27.05   28.02  12.08    5.00   48.80
## bbSpeed               9 56   84.50   2.20   84.40   84.47   2.59   79.60   89.20
## bbSpin               10 56 2455.91 170.95 2469.00 2456.59 197.19 2018.00 2777.00
## bbBreak              11 56    7.76   2.14    7.60    7.64   2.08    3.70   13.10
## offSpeed             12 56   86.70   1.93   86.55   86.84   1.26   81.10   90.90
## offSpin              13 56 2255.21 128.17 2267.00 2259.98 146.04 1855.00 2512.00
## offBreak             14 56   20.02   1.80   20.30   20.02   2.22   16.50   23.40
##                     range  skew kurtosis    se
## woba                 0.16  0.51    -0.80  0.01
## exit_velocity_avg    7.50 -0.22    -0.35  0.23
## era                  5.93  0.54    -0.48  0.18
## fbRate              32.80  0.25    -0.54  1.09
## fbSpeed              7.90  0.58    -0.06  0.22
## fbSpin             434.00  0.19    -0.71 14.13
## fbBreak             10.00  0.10    -0.53  0.30
## bbRate              43.80 -0.01    -0.87  1.41
## bbSpeed              9.60  0.10    -0.76  0.29
```

```
## bbSpin            759.00 -0.10    -0.63 22.84
## bbBreak             9.40  0.47    -0.21  0.29
## offSpeed            9.80 -0.79     1.45  0.26
## offSpin           657.00 -0.48     0.42 17.13
## offBreak            6.90 -0.07    -1.09  0.24
```

Cluster one: Of the clusters, these are the best pitchers. They throw the hardest and have among the highest spin rates on all their pitches. Their breaking balls do not break dramatically, indicating more usage of sliders than curveballs. They have pretty normal rates at which they throw their pitches. Notable cluster one pitchers include Max Scherzer†, Gerrit Cole†, Jake Arrieta†, Blake Snell†, and Justin Verlander†.

```
describe(cluster2[5:18])
```

```
##                   vars  n    mean     sd  median trimmed    mad     min     max
## woba                 1 23    0.33   0.03    0.33    0.33   0.03    0.23    0.38
## exit_velocity_avg    2 23   87.38   1.35   87.40   87.39   1.19   84.50   90.10
## era                  3 23    4.59   1.42    4.50    4.55   1.16    1.85    8.14
## fbRate               4 23   68.70   8.35   69.70   69.03  10.53   52.00   80.70
## fbSpeed              5 23   85.61   2.29   85.40   85.62   1.63   80.30   90.50
## fbSpin               6 23 1980.30  89.02 1955.00 1976.95  69.68 1817.00 2157.00
## fbBreak              7 23   16.38   2.16   16.20   16.30   2.52   13.10   21.00
## bbRate               8 23   12.17   4.74   11.70   11.85   4.74    5.00   22.10
## bbSpeed              9 23   72.57   2.58   73.00   72.58   2.08   67.60   77.30
## bbSpin              10 23 2077.26 145.97 2070.00 2089.26 163.09 1714.00 2272.00
## bbBreak             11 23   13.08   2.28   13.90   13.25   1.78    7.90   16.40
## offSpeed            12 23   78.12   2.60   78.40   78.25   2.08   70.50   82.60
## offSpin             13 23 1675.00 182.97 1624.00 1658.21 151.23 1416.00 2104.00
## offBreak            14 23   16.26   1.64   15.90   16.14   1.78   13.70   19.80
##                    range  skew kurtosis    se
## woba                0.15 -0.69     0.67  0.01
## exit_velocity_avg   5.60 -0.06    -0.55  0.28
## era                 6.29  0.41     0.16  0.30
## fbRate             28.70 -0.30    -1.24  1.74
## fbSpeed            10.20 -0.03    -0.13  0.48
## fbSpin            340.00  0.43    -0.76 18.56
## fbBreak             7.90  0.26    -0.89  0.45
## bbRate             17.10  0.47    -0.85  0.99
## bbSpeed             9.70 -0.08    -0.64  0.54
## bbSpin            558.00 -0.67    -0.25 30.44
## bbBreak             8.50 -0.71    -0.63  0.47
## offSpeed           12.10 -0.70     1.17  0.54
## offSpin           688.00  0.86    -0.36 38.15
## offBreak            6.10  0.53    -0.45  0.34
```

Cluster two: These pitchers throw the slowest with high rates of offspeed pitches. Despite using them the most, their offspeed pitches have among the lowest speeds and spin rates. These pitchers also have a high break on their breaking balls which indicates more curveball usage. Despite this, they have a middling overall skill level which indicates that there is probably some underlying variable that makes these pitchers better despite their pitches not always being the best, such as the best ability to control their pitches and throw them where they want to. Notable pitchers include Dallas Keuchal†, Kyle Hendricks†, Max Fried, Hyun Jin Ryu, and Aaron Nola.

```
describe(cluster3[5:18])
```

```
##                    vars  n    mean     sd  median trimmed    mad     min     max
## woba                  1 71    0.30   0.05    0.30    0.30   0.04    0.17    0.43
## exit_velocity_avg     2 71   88.18   1.98   88.10   88.26   1.78   82.50   91.80
## era                   3 71    3.81   1.44    3.64    3.67   1.22    1.60    9.90
## fbRate                4 71   75.58   7.55   74.30   75.32   6.97   61.40   90.00
## fbSpeed               5 71   96.09   1.64   96.00   96.07   1.93   92.40  100.70
## fbSpin                6 71 2269.79 103.98 2265.00 2271.26  93.40 2036.00 2476.00
## fbBreak               7 71   19.64   1.64   19.70   19.68   1.48   15.90   23.70
## bbRate                8 71   17.56   7.21   17.80   17.59   8.75    4.00   31.20
## bbSpeed               9 71   85.02   2.92   85.60   84.98   2.82   80.10   92.60
## bbSpin               10 71 2209.45 216.29 2245.00 2208.91 299.49 1759.00 2618.00
## bbBreak              11 71    7.39   2.63    7.30    7.18   2.97    3.40   15.40
## offSpeed             12 71   88.84   2.19   88.90   88.88   1.63   83.20   94.40
## offSpin              13 71 1639.04 215.18 1628.00 1634.16 177.91 1111.00 2188.00
## offBreak             14 71   16.67   2.89   15.90   16.52   2.67   11.30   26.30
##                     range  skew kurtosis    se
## woba                 0.26  0.14     0.76  0.01
## exit_velocity_avg    9.30 -0.31    -0.16  0.24
## era                  8.30  1.36     3.20  0.17
## fbRate              28.60  0.34    -0.88  0.90
## fbSpeed              8.30  0.16    -0.47  0.20
## fbSpin             440.00 -0.13    -0.56 12.34
## fbBreak              7.80 -0.20     0.02  0.19
## bbRate              27.20 -0.03    -1.05  0.86
## bbSpeed             12.50  0.05    -0.64  0.35
## bbSpin             859.00 -0.01    -1.18 25.67
## bbBreak             12.00  0.62    -0.18  0.31
## offSpeed            11.20 -0.14     0.61  0.26
## offSpin           1077.00  0.24     0.30 25.54
## offBreak            15.00  0.69     0.61  0.34
```

Cluster three: Like cluster two, these pitchers are notable for having a high rate of offspeed pitches, but they throw their pitches much faster and are more likely to use sliders. Cluster three does not appear to have any statistics at which they stand out, although they have among the lowest spin rates. They are not the best or the worst at anything. In terms of results, these pitchers seem to be very average across the board. Although I do not have a scientific way of measuring this, cluster three seems to be made up of more relievers than the other clusters. Interestingly, this is by far the biggest cluster. Notable cluster three pitchers include Josh Hader, James Paxton, Luis Castillo, Johnny Cueto, and Kirby Yates.

```
describe(cluster4[5:18])
```

```
##                    vars  n    mean     sd  median trimmed    mad     min     max
## woba                  1 33    0.33   0.03    0.33    0.33   0.03    0.26    0.40
## exit_velocity_avg     2 33   88.18   1.77   88.00   88.05   1.48   85.30   93.40
## era                   3 33    4.61   1.25    4.47    4.57   0.85    1.61    7.79
## fbRate                4 33   70.70   7.13   71.80   71.40   5.93   52.50   82.50
## fbSpeed               5 33   89.06   1.52   89.10   89.10   1.63   86.10   91.70
## fbSpin                6 33 2300.09 108.44 2269.00 2293.93  88.96 2116.00 2509.00
## fbBreak               7 33   15.30   1.38   15.10   15.24   1.63   12.90   18.10
## bbRate                8 33   22.93   7.44   23.40   22.53   8.75   11.90   38.20
```

```
## bbSpeed              9 33   75.32   1.49   75.10   75.29   1.48   72.50   78.80
## bbSpin              10 33 2607.85 146.65 2591.00 2609.00 145.29 2322.00 2922.00
## bbBreak             11 33   15.42   3.70   14.90   15.24   4.45    7.60   23.10
## offSpeed            12 33   83.86   1.29   83.90   83.90   1.19   81.10   86.00
## offSpin             13 33 1722.21 115.30 1735.00 1719.04 126.02 1535.00 1953.00
## offBreak            14 33   16.25   1.75   16.20   16.23   1.63   12.50   20.90
##                     range  skew kurtosis    se
## woba                 0.14 -0.11    -0.58  0.01
## exit_velocity_avg    8.10  0.70     0.59  0.31
## era                  6.18  0.32     0.60  0.22
## fbRate              30.00 -0.89     0.24  1.24
## fbSpeed              5.60 -0.25    -1.00  0.26
## fbSpin             393.00  0.48    -0.89 18.88
## fbBreak              5.20  0.36    -0.83  0.24
## bbRate              26.30  0.31    -0.86  1.29
## bbSpeed              6.30  0.20    -0.60  0.26
## bbSpin             600.00  0.00    -0.70 25.53
## bbBreak             15.50  0.27    -0.73  0.64
## offSpeed             4.90 -0.31    -0.67  0.22
## offSpin            418.00  0.18    -0.94 20.07
## offBreak             8.40  0.27     0.11  0.31
```

Cluster four: Cluster four is clearly the second-best cluster behind cluster 1. These pitchers stand out for having among the highest spin rates in every type of pitch, and that they pitch at relatively high speeds. Notably, cluster four pitchers throw breaking balls at by far the highest rate of all the clusters (40.9%). These pitchers seem to thrive on having really high-quality pitches (high spin), even if they don't throw them the fastest. Notable cluster four pitchers include Shane Bieber†, Jacob deGrom†, Corey Kluber†, Luis Severino, and Madison Bumgarner.

```
describe(cluster5[5:18])
```

```
##                    vars  n    mean     sd  median trimmed    mad     min     max
## woba                  1 44    0.31   0.04    0.31    0.31   0.03    0.22    0.42
## exit_velocity_avg     2 44   88.05   1.50   88.20   88.07   1.56   83.30   91.60
## era                   3 44    4.24   1.39    4.16    4.18   1.30    1.63    7.77
## fbRate                4 44   43.75   7.28   44.35   44.28   5.41   17.40   58.90
## fbSpeed               5 44   92.57   2.20   92.45   92.50   2.37   88.80   97.50
## fbSpin                6 44 2353.59  76.71 2349.50 2350.97  85.99 2197.00 2555.00
## fbBreak               7 44   17.72   1.61   17.60   17.74   1.70   14.40   21.30
## bbRate                8 44   46.90   8.59   46.45   46.38   9.27   32.60   73.40
## bbSpeed               9 44   82.76   2.21   82.80   82.80   1.93   77.90   87.10
## bbSpin               10 44 2622.20 165.66 2627.00 2629.58 183.10 2260.00 2898.00
## bbBreak              11 44   11.07   2.33   11.10   11.04   2.45    6.70   17.40
## offSpeed             12 44   84.86   2.27   84.40   84.77   2.52   81.40   89.30
## offSpin              13 44 1774.36 148.02 1791.50 1768.17 124.54 1488.00 2225.00
## offBreak             14 44   16.84   1.51   16.85   16.86   1.70   13.60   19.70
##                     range  skew kurtosis    se
## woba                 0.20  0.40    -0.01  0.01
## exit_velocity_avg    8.30 -0.31     1.07  0.23
## era                  6.14  0.35    -0.18  0.21
## fbRate              41.50 -1.07     2.37  1.10
## fbSpeed              8.70  0.25    -0.53  0.33
## fbSpin             358.00  0.33    -0.25 11.56
```

```
## fbBreak             6.90 -0.03    -0.75  0.24
## bbRate             40.80  0.64     0.30  1.30
## bbSpeed             9.20 -0.15    -0.59  0.33
## bbSpin            638.00 -0.30    -0.70 24.97
## bbBreak            10.70  0.28     0.25  0.35
## offSpeed            7.90  0.32    -1.20  0.34
## offSpin           737.00  0.59     1.09 22.31
## offBreak            6.10 -0.08    -0.83  0.23
```

Cluster five: These pitchers are once again somewhat average. Their overall quality is average, and as is the quality and speed of their pitches for the most part. Cluster five pitchers tend to be fastball first pitches, throwing fastballs at the highest rates (despite middling speed) and breaking balls at the lowest rate. If they throw a breaking ball, it tends to be a curveball, as evidenced by the high break, but they do have the highest spin rate on their curveball. Notable cluster five pitchers include Trevor Bauer†, Stephen Strasburg, Yu Darvish, Charlie Morton, and Marcus Stroman.

```
describe(cluster6[5:18])
```

```
##                    vars  n    mean     sd  median trimmed    mad     min     max
## woba                  1 38    0.33   0.04    0.32    0.33   0.03    0.23    0.43
## exit_velocity_avg     2 38   88.34   1.39   88.40   88.30   1.33   85.50   91.50
## era                   3 38    4.70   1.47    4.34    4.57   0.87    2.16    9.83
## fbRate                4 38   57.30   7.22   58.55   57.47   6.75   35.10   72.10
## fbSpeed               5 38   91.43   2.10   91.90   91.52   2.00   87.00   95.20
## fbSpin                6 38 2198.63  75.16 2205.00 2198.38  68.94 2062.00 2348.00
## fbBreak               7 38   20.07   1.81   19.75   20.07   1.93   15.60   23.70
## bbRate                8 38   19.73   7.32   18.30   19.56   8.75    7.20   35.30
## bbSpeed               9 38   79.07   2.74   78.15   78.87   1.85   74.70   85.80
## bbSpin               10 38 2168.92 180.96 2208.50 2176.22 161.60 1774.00 2585.00
## bbBreak              11 38   10.66   3.12   10.45   10.63   2.89    4.60   16.30
## offSpeed             12 38   82.54   2.06   82.90   82.62   1.41   77.70   86.80
## offSpin              13 38 2180.32 161.32 2219.50 2190.31 143.81 1786.00 2412.00
## offBreak             14 38   19.79   2.34   19.75   19.82   2.59   14.30   24.00
##                     range  skew kurtosis    se
## woba                 0.20  0.42     0.32  0.01
## exit_velocity_avg    6.00  0.28     0.00  0.23
## era                  7.67  1.27     2.11  0.24
## fbRate              37.00 -0.53     0.71  1.17
## fbSpeed              8.20 -0.46    -0.75  0.34
## fbSpin             286.00 -0.08    -0.75 12.19
## fbBreak              8.10 -0.03    -0.57  0.29
## bbRate              28.10  0.25    -1.10  1.19
## bbSpeed             11.10  0.75    -0.28  0.44
## bbSpin             811.00 -0.36    -0.33 29.36
## bbBreak             11.70  0.16    -0.87  0.51
## offSpeed             9.10 -0.51    -0.04  0.33
## offSpin            626.00 -0.64    -0.40 26.17
## offBreak             9.70 -0.10    -0.81  0.38
```

Cluster Six: This cluster seems to contain the on average worst pitchers. They have among the lowest spin rates and do not have good speed on their pitches. They throw breaking balls at among the highest rates, but their spin rate is not good. They do not seem to clearly throw just sliders or curveballs, as their breaking ball break is in the middle. Their pitches all seem to be outliers in terms of how they break, which perhaps

means these pitchers have more unique pitching styles and/or motions. Notable cluster six pitchers include Clayton Kershaw†, Chris Sale, Mike Clevinger, Patrick Corbin, and Zack Greinke.

† Indicates having won a Cy-Young Award which is given to the two best pitchers (one in each league) at the end of each season.

Conclusion

Overall, I found that my PCA and clustering produced lackluster statistical results but meaningful baseball results. Both PCA and clustering produced results that were not strong - PCA needed many variables to properly account for the variance and clustering produced clusters that were not particularly strongly grouped, as evidenced by the low silhouette coefficients. However, after digging deeper into some of the numbers, the clustering, in particular, produced results that clearly distinguished different types of pitchers. As a baseball fan, I could see clear commonalities among the pitchers in each cluster, even if the numbers were not always particularly clear. If I were to expand upon this study in the future, I would like to cross reference my results with hitters. I would like to answer the question "are there certain hitters who thrive against one cluster of pitchers, and do poorly against another?" Before conducting this study, I hoped to find whether I could use these methods to answer the question of "is a certain type of baseball pitcher the best?" While my analysis showed small differences in the types of baseball pitchers (cluster one was on average much better than cluster six, but there are plenty of cluster six pitchers who were better than cluster one pitchers), it is clear that there is not one prototype of baseball pitcher that will always be the best. Even if that is the less interesting answer, it serves as a reminder that anyone in baseball can succeed, but that certain traits are more often successful.

Citations

Major League Baseball. “Baseball Savant: Trending MLB Players, Statcast and Visualizations.” Baseballsavant.com, 28 Oct. 2020, baseballsavant.mlb.com/.

```
load("/Users/XanderPA/Desktop/Old Amherst /Competition Stats/Final Project/files/individualpitches.rds")
```

```
library(stringr)
events <- all_pitches %>%
  filter(!is.na(events)) %>%
  select(player_name, pitcher, batter, events) %>%
  mutate(Result = ifelse(str_detect(events, "ball"), "ball", ifelse(str_detect(events, "strike")& !str_
```

```
events <- events %>%
  filter(Result == "double" | Result == "triple" | Result == "home run" | Result == "out" | Result == "
  mutate(isHit = ifelse(Result == "out", 0, 1))
```

```
leader_BA <- scrape_savant_leaderboards(
  leaderboard = "expected_statistics",
  year = 2019,
  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: 897 Columns: 15

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (4): last_name, first_name, est_ba, est_slg
## dbl (11): 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.
```

```r
leader_BA_Pit <- scrape_savant_leaderboards(
  leaderboard = "expected_statistics",
  year = 2019,
  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: 826 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.
```

```r
leader_BA <- leader_BA %>% select(first_name, last_name, year, player_id, ba, pa)
combinedData <- left_join(events, 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"))

fullPitchers <- fullPitchers %>% mutate(first_name = trimws(first_name))

allData <- left_join(combinedData2, fullPitchers, by = c("last_name.y" = "last_name", "year.y" = "year"
```

```r
groupBatterPitcher <- allData %>%
  group_by(first_name.x, first_name.y, last_name.x, last_name.y) %>%
  summarise(P_AB = n())
```

```
## `summarise()` has grouped output by 'first_name.x', 'first_name.y', 'last_name.x'. You can override u
```

```r
groupBatterPitcherHits <- allData %>%
  group_by(first_name.x, first_name.y, last_name.x, last_name.y) %>%
  summarise(P_Hits = sum(isHit))
```

```
## `summarise()` has grouped output by 'first_name.x', 'first_name.y', 'last_name.x'. You can override u
```

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

```
## Joining, by = c("first_name.x", "first_name.y", "last_name.x", "last_name.y")
```

```
allData <- inner_join(allData, pitcherCombo)
```

```
## Joining, by = c("first_name.x", "last_name.x", "first_name.y", "last_name.y")
```

```
groupBatterCluster <- allData %>%
  group_by(first_name.x, , last_name.x, Cluster) %>%
  summarise(C_AB = n())
```

```
## `summarise()` has grouped output by 'first_name.x', 'last_name.x'. You can override using the `.group
```

```
groupBatterClusterHits <- allData %>%
  group_by(first_name.x, , last_name.x, Cluster) %>%
  summarise(C_Hits = sum(isHit))
```

```
## `summarise()` has grouped output by 'first_name.x', 'last_name.x'. You can override using the `.group
```

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

```
## Joining, by = c("first_name.x", "last_name.x", "Cluster")
```

```
allData <- inner_join(allData, clusterCombo)
```

```
## Joining, by = c("first_name.x", "last_name.x", "Cluster")
```

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

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

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
##           -2.0485            4.8509            0.0327           -0.0193
```

```
##      P_Hits:P_AVG  P_AB:P_Hits:P_AVG
##           -0.1640             0.0192
##
## Degrees of Freedom: 21492 Total (i.e. Null);  21487 Residual
## Null Deviance:       25900
## Residual Deviance: 25800     AIC: 25800
```

```
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, fami
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
##          -1.99185           5.17627           0.02695          -0.01782
##            C_Hits             C_AVG       C_AB:C_Hits       C_AB:C_AVG
##           0.01505          -0.92599           0.00252                NA
##      C_Hits:C_AVG  C_AB:C_Hits:C_AVG
##           0.13783          -0.00874
##
## Degrees of Freedom: 21492 Total (i.e. Null);  21484 Residual
## Null Deviance:       25900
## Residual Deviance: 25800     AIC: 25800
```

```
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     21490      25789
## 2     21484      25776  6    12.17   0.0583 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: isHit
##
## Terms added sequentially (first to last)
##
##
##                   Df Deviance Resid. Df Resid. Dev            Pr(>Chi)
## NULL                                21492      25884
```

```
## ba.x               1    95.50     21491      25789 <0.0000000000000002 ***
## ba.y               1     0.11     21490      25789              0.7408
## C_AB               1     3.24     21489      25785              0.0720 .
## C_Hits             1     3.16     21488      25782              0.0756 .
## C_AVG              1     3.01     21487      25779              0.0829 .
## C_AB:C_Hits        1     0.08     21486      25779              0.7835
## C_AB:C_AVG         0     0.00     21486      25779
## C_Hits:C_AVG       1     0.07     21485      25779              0.7889
## C_AB:C_Hits:C_AVG  1     2.62     21484      25776              0.1053
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
library(ResourceSelection)
```

```
## ResourceSelection 0.3-5   2019-07-22
```

```
hoslem.test(x = filteredData$isHit,y=fitted(clusterMod), g=9)
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  filteredData$isHit, fitted(clusterMod)
## X-squared = 26.29, df = 7, p-value = 0.000447
```

```
hoslem.test(x = filteredData$isHit,y=fitted(bothMod))
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  filteredData$isHit, fitted(bothMod)
## X-squared = 17.37, df = 8, p-value = 0.0264
```

```
hoslem.test(x = filteredData$isHit,y=fitted(pitcherMod))
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  filteredData$isHit, fitted(pitcherMod)
## X-squared = 27.94, df = 8, p-value = 0.000486
```

```
allData <- 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
```

```
mean(filteredData$isHit)
```

```
## [1] 0.290001
```

```
table(all_pitches$events)
```

```
##
##               catcher_interf           caught_stealing_2b
##                           60                          172
##           caught_stealing_3b         caught_stealing_home
##                           10                           12
##                       double                   double_play
##                         8566                          405
##                  field_error                    field_out
##                         1358                        72576
##              fielders_choice          fielders_choice_out
##                          409                          289
##                   force_out                game_advisory
##                         3569                            2
##    grounded_into_double_play                 hit_by_pitch
##                         3469                         1992
##                     home_run                    other_out
##                         6798                           36
##                  passed_ball                   pickoff_1b
##                            1                           10
##                   pickoff_2b                   pickoff_3b
##                            4                            3
##   pickoff_caught_stealing_3b pickoff_caught_stealing_home
##                            1                            3
##           runner_double_play                     sac_bunt
##                            1                          777
##         sac_bunt_double_play                     sac_fly
##                            3                         1142
##          sac_fly_double_play                      single
##                           11                        26040
##               stolen_base_2b                    strikeout
##                            1                        42874
##        strikeout_double_play                      triple
##                          160                          785
##                  triple_play                       walk
##                            3                        15206
##                   wild_pitch
##                            5
```

```
table(filteredData$events)
```

```
##
##                     double               double_play              field_error
##                       1154                        55                      150
##                  field_out       fielders_choice_out                force_out
##                       9671                        41                      423
## grounded_into_double_play                  home_run                other_out
```

```
##                      410                      979                        5
##   sac_bunt_double_play     sac_fly_double_play                   single
##                        1                        1                     3480
##                strikeout   strikeout_double_play                   triple
##                     4970                       32                      121
```

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

sum(testSamp$isHit)
```

```
## [1] 1483
```

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

```
## [1] 1448.88
```

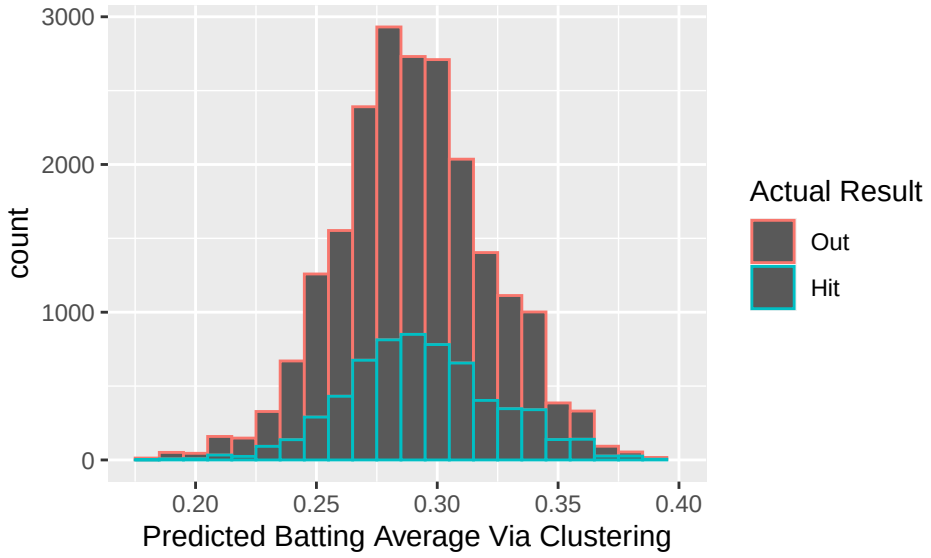
```
sum(testSamp$isHitP)
```

```
## [1] 1447.96
```

```
ggplot(allData, aes(x=isHitC, col = as.factor(isHit))) +
  geom_histogram(binwidth = .01) +
  xlim(.175,.4)+xlab("Predicted Batting Average Via Clustering")+ scale_color_discrete(labels=c("Out","H
```

```
## Warning: Removed 67 rows containing non-finite values (stat_bin).
```

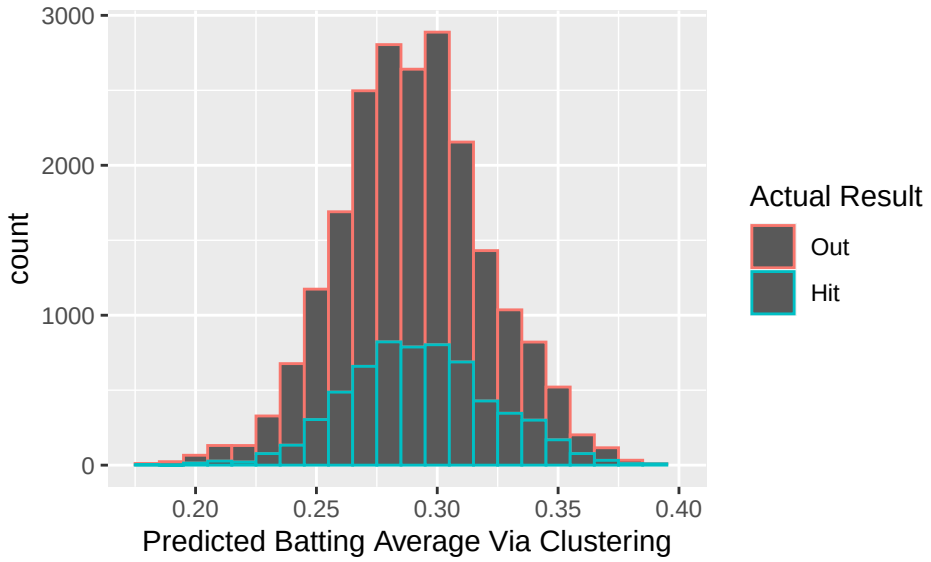
```
## Warning: Removed 4 rows containing missing values (geom_bar).
```



```
ggplot(allData, aes(x=isHitP, col = as.factor(isHit))) +
  geom_histogram(binwidth = .01) +
  xlim(.175,.4)+xlab("Predicted Batting Average Via Clustering")+ scale_color_discrete(labels=c("Out","H
```

```
## Warning: Removed 107 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 4 rows containing missing values (geom_bar).
```




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

```
##      vars     n mean   sd median trimmed  mad  min  max range  skew kurtosis se
## X1      1 21493 0.29 0.03   0.29    0.29 0.03 0.15 0.51  0.36  0.16     1.82  0
## X11     1 21493 0.29 0.03   0.29    0.29 0.03 0.14 0.41  0.27 -0.07     0.63  0
```

‘