MLB Starting Pitcher Clustering

Tyler Fuelling

August 16, 2020

The goal of this project was to analyze starting pitchers in the MLB at first by their pitching style rather than their level of success. Then, to examine different compositions of MLB starting pitching rotations based on the different pitching styles of the pitchers that make them up, and to see which rotation compostions have lead to the most success. The examination of different starting pitching styles will be accomplished through k-means clustering.

We begin by loading the data into a dataframe. The dataset contains data for all MLB starting picthers who have pitched at least 100 total innings during the 2010 through 2019 seasons. Because the k-means clustering will be done based on pitching style rather than the pitcher's level of success, the dataset contains metrics such as fastball percentage, pitch velocities, standardized runs by pitch type, expected velocity, first-strike percentage, strikeout percentage, walk percentage, and swinging strike percentage, rather than conventional pitching success metrics, such as ERA or WHIP. Data obtained from https://www.fangraphs.com/ (https://www.fangraphs.com/).

```r
# read the data file into a dataframe
df <- read.csv("sp_data 2010-2019.csv", header = TRUE, check.names = TRUE)
head(df)
```

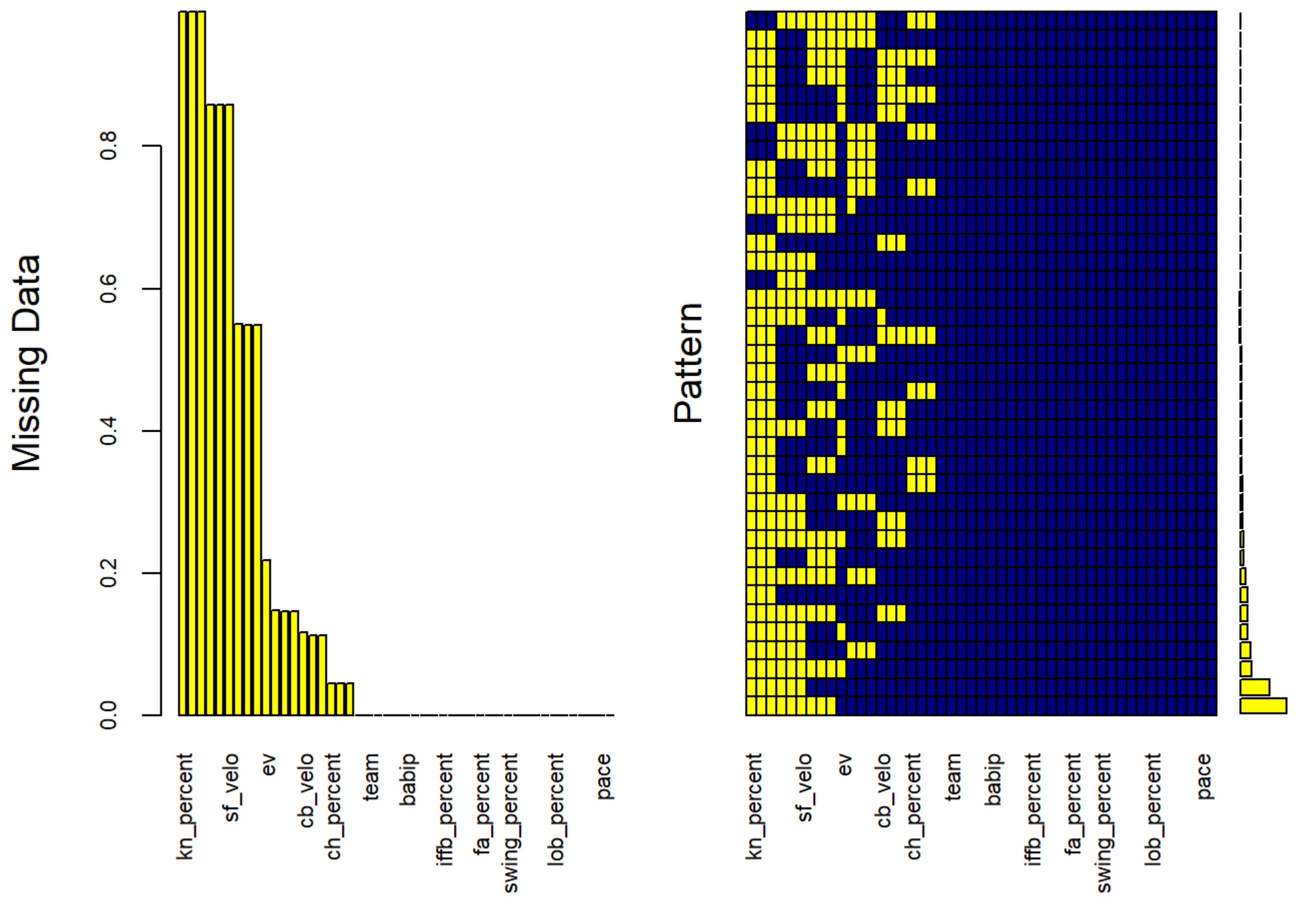
```
##           ï..Name    Team     IP    K.   BB. BABIP   GB.   LD.   FB. IFFB.   EV
## 1 Clayton Kershaw Dodgers 1995.0 0.280 0.056 0.267 0.469 0.199 0.332 0.119 86.9
## 2  Jose Fernandez Marlins  471.1 0.312 0.074 0.292 0.430 0.245 0.325 0.077 88.9
## 3    Jacob deGrom    Mets 1101.2 0.286 0.061 0.291 0.452 0.217 0.330 0.110 86.9
## 4     Mike Soroka  Braves  200.1 0.200 0.059 0.289 0.502 0.247 0.251 0.123 88.3
## 5  Walker Buehler Dodgers  318.2 0.288 0.058 0.270 0.458 0.207 0.335 0.067 88.6
## 6       Rich Hill   - - -  465.1 0.293 0.077 0.265 0.418 0.188 0.394 0.136 87.2
##   Soft.  Med. Hard. FB..1   SL.   CT.   CB.   CH.   SF. KN. wFB.C wSL.C wCT.C
## 1 0.214 0.507 0.279 0.564 0.290    NA 0.130 0.017    NA  NA  1.07  1.88    NA
## 2 0.178 0.518 0.305 0.551 0.213    NA 0.126 0.110    NA  NA  0.57  3.46    NA
## 3 0.209 0.497 0.295 0.561 0.220    NA 0.083 0.136    NA  NA  0.94  1.55    NA
## 4 0.183 0.464 0.353 0.641 0.240    NA    NA 0.119    NA  NA  1.00  1.14    NA
## 5 0.180 0.432 0.388 0.600 0.138 0.116 0.128 0.018    NA  NA  1.40  1.80  1.16
## 6 0.210 0.451 0.339 0.534 0.050 0.013 0.391 0.012 0.001  NA  1.13 -0.27  1.42
##   wCB.C wCH.C wSF.C wKN.C O.Swing. Swing. Contact. Zone. F.Strike. SwStr.  LOB.
## 1  1.69 -0.15    NA    NA    0.324  0.492    0.744 0.479     0.664  0.126 0.799
## 2  0.31  0.82    NA    NA    0.326  0.468    0.728 0.466     0.616  0.127 0.772
## 3  0.51  1.63    NA    NA    0.348  0.506    0.736 0.446     0.650  0.134 0.793
## 4    NA  1.66    NA    NA    0.335  0.480    0.785 0.429     0.629  0.103 0.774
## 5 -0.08 -3.79    NA    NA    0.334  0.494    0.764 0.467     0.649  0.117 0.749
## 6  0.73  4.53  2.45    NA    0.290  0.454    0.760 0.504     0.630  0.109 0.806
##    FBv  SLv  CTv  CBv  CHv  SFv KNv Z.Swing. O.Contact. Z.Contact. Pace
## 1 92.7 86.4   NA 73.6 84.5   NA  NA    0.674      0.568      0.835 22.3
## 2 95.2 83.7   NA 81.4 87.9   NA  NA    0.631      0.537      0.841 20.4
## 3 95.1 90.3   NA 81.4 87.5   NA  NA    0.703      0.600      0.819 21.9
## 4 92.5 83.5   NA   NA 81.4   NA  NA    0.674      0.647      0.876 22.9
## 5 96.4 86.9 92.4 80.5 90.3   NA  NA    0.676      0.617      0.847 22.5
## 6 89.6 75.4 85.1 74.2 83.0 82.8  NA    0.615      0.653      0.810 21.3
##   playerid
## 1     2036
## 2    11530
## 3    10954
## 4    18383
## 5    19374
## 6     4806
```

```
# renaming the columns of the the df for easier use
colnames(df)[1] <- "name"
df <- df %>% rename(team = Team, ip = IP, k_percent = K., bb_percent = BB., babip = BABIP, gb_pe
rcent = GB.,
                    ld_percent = LD., fb_percent = FB.,iffb_percent = IFFB., ev = EV, soft_perce
nt = Soft.,
                    med_percent = Med., hard_percent = Hard., fa_percent = FB..1, sl_percent = S
L.,
                    ct_percent = CT., cb_percent = CB., ch_percent = CH., sf_percent = SF., kn_p
ercent = KN.,
                    fa_standardized_runs = wFB.C, sl_standardized_runs = wSL.C, ct_standardized_
runs = wCT.C,
                    cb_standardized_runs = wCB.C, ch_standardized_runs = wCH.C, sf_standardized_
runs = wSF.C,
                    kn_standardized_runs = wKN.C, fa_velo = FBv, sl_velo = SLv, ct_velo = CTv, c
b_velo = CBv,
                    ch_velo = CHv, sf_velo = SFv, kn_velo = KNv, o_swing_percent = O.Swing.,
                    z_swing_percent = Z.Swing., swing_percent = Swing., o_contact_percent = O.Co
ntact.,
                    z_contact_percent = Z.Contact., contact_percent = Contact., zone_percent = Z
one.,
                    first_strike_percent = F.Strike., swinging_strike_percent = SwStr., lob_perc
ent = LOB.,
                    pace = Pace, player_id = playerid)
```

Because starting pitchers do not all throw the same types of pitches, there are many NA values present in the dataset where data is missing. For example, a pitcher who does not throw a slider will have NA values for slider percentage and slider velocity.

```
# examining the NA values present in the dataframe
aggr(df, col=c('navyblue', 'yellow'), numbers = TRUE, sortVars = TRUE, labels = names(df), cex.a
xis = .7,
     gap = 3, ylab = c("Missing Data", "Pattern"))
```

```
## Warning in plot.aggr(res, ...): not enough vertical space to display frequencies
## (too many combinations)
```


Missing Data
0.0
0.2
0.4
0.6
0.8
kn_percent
sf_velo
ev
cb_velo
ch_percent
team
babip
iffb_percent
fa_percent
swing_percent
lob_percent
pace
Pattern
kn_percent
sf_velo
ev
cb_velo
ch_percent
team
babip
iffb_percent
fa_percent
swing_percent
lob_percent
pace

```
##
##  Variables sorted by number of missings:
##                 Variable     Count
##               kn_percent 0.9894515
##     kn_standardized_runs 0.9894515
##                  kn_velo 0.9894515
##               sf_percent 0.8586498
##     sf_standardized_runs 0.8586498
##                  sf_velo 0.8586498
##                  ct_velo 0.5506329
##               ct_percent 0.5485232
##     ct_standardized_runs 0.5485232
##                       ev 0.2172996
##                  sl_velo 0.1476793
##               sl_percent 0.1455696
##     sl_standardized_runs 0.1455696
##                  cb_velo 0.1160338
##               cb_percent 0.1118143
##     cb_standardized_runs 0.1118143
##               ch_percent 0.0443038
##     ch_standardized_runs 0.0443038
##                  ch_velo 0.0443038
##                     name 0.0000000
##                     team 0.0000000
##                       ip 0.0000000
##                k_percent 0.0000000
##               bb_percent 0.0000000
##                    babip 0.0000000
##               gb_percent 0.0000000
##               ld_percent 0.0000000
##               fb_percent 0.0000000
##             iffb_percent 0.0000000
##             soft_percent 0.0000000
##              med_percent 0.0000000
##             hard_percent 0.0000000
##               fa_percent 0.0000000
##     fa_standardized_runs 0.0000000
##          o_swing_percent 0.0000000
##            swing_percent 0.0000000
##          contact_percent 0.0000000
##             zone_percent 0.0000000
##     first_strike_percent 0.0000000
##  swinging_strike_percent 0.0000000
##              lob_percent 0.0000000
##                  fa_velo 0.0000000
##          z_swing_percent 0.0000000
##        o_contact_percent 0.0000000
##        z_contact_percent 0.0000000
##                     pace 0.0000000
##                player_id 0.0000000
```

Because there are so many missing elements in the dataset, all of these NA values must be addressed in some fashion. So, multivariate imputation will be utiilized to address this issue. Imputation is a method used to fill in the missing values of a dataset with values that are estimated using the rest of the values in the dataset. In this project, a specific type of imputation, multivariate imputation via chained equations, will be implemented using the MICE package in R. This process is further explained here: https://medium.com/coinmonks/dealing-with-missing-data-using-r-3ae428da2d17 (https://medium.com/coinmonks/dealing-with-missing-data-using-r-3ae428da2d17).

```
# removing the player id column - we don't need it
df <- subset(df, select = -c(player_id))
# creating a new subset of the dataframe without the name and team columns to be used for multiv
ariate imputation (MICE package)
mice_subset <- subset(df, select = -c(name, team))

# Multivariate Imputation via Chained Equations
# creating 15 imputed data sets because the original data set contained about 15% NA's, max iter
ations of 50, using "predictive mean matching"
imputed_data <- mice(mice_subset, m = 15, maxit = 50, method = 'pmm', seed = 123)
```

```
## Warning: Number of logged events: 12003
```

```
# merging the 15 imputed data sets and appending the imputed data to the mice_subset dataframe
mice_subset <- merge_imputations(mice_subset, imputed_data, mice_subset)
# removing the columns containing NA's from the dataframe, these were replaced with the imputed
 data
mice_subset <- mice_subset[colSums(is.na(mice_subset)) == 0]

# verify that there are no more NA's present in the dataframe
table(is.na(mice_subset))
```

After performing the multivariate imputation with the MICE package, the datset is organized and then standardized to a mean of 0 and a standard deviation of 1 in order to facilitate it being used for k-means clustering.

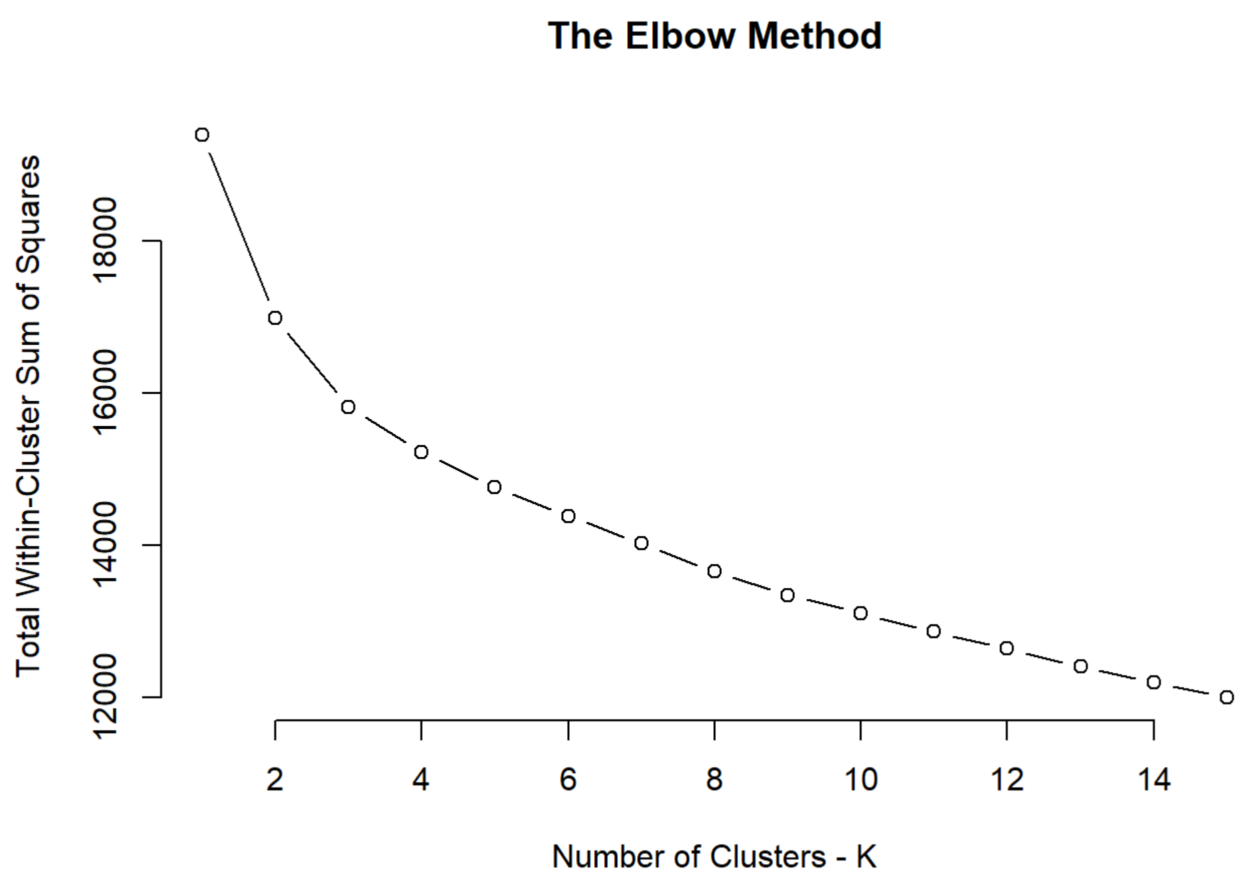
```
# make a copy of the dataframe that contains all of the numerical data, including all of the imp
uted data
imputed_df <- data.frame(mice_subset)
# add the name and team columns from the original dataframe to this dataframe that contains all
 of the data
imputed_df$name <- df$name
imputed_df$team <- df$team

# reorganizing the order of the columns of the dataframe
imputed_df <- imputed_df[c(42, 43, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 14, 15, 16, 17, 18, 19, 20
, 22, 23, 24,
                           25, 26, 12, 27, 28, 29, 30, 31, 13, 32, 33, 34, 35, 36, 21, 37, 38, 3
9, 40, 41)]

# standardizing the data so that it can be used for k-means clustering
standardized_imputed_df <- data.frame(imputed_df)
standardized_imputed_df[c(3:43)] <- scale(standardized_imputed_df[c(3:43)])
```

k-means clustering is a method/algorithm that partitions n observations into k different clusters in which each observation belongs to the cluster with the nearest mean. However, when using the k-means clustering algorithm, the user must specify k. There are two main methods for determining the optimal k: the elbow method and silhouette analysis, both of which are used in this project. The Elbow Method involves graphing the total within-cluster sum of squares for the result of k-means clustering with k clusters as a function of k, and then selecting the k at which the graph forms an "elbow". Silhouette analysis involves measuring the average distance between clusters when k-means clustering is performed with k clusters (the farther away the clusters are from each other, the better). More explanation of these methods can be found here: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a (https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a).

```
# determining the optimal number of clusters to use for k-means clustering

set.seed(123) # set the seed so that the clustering achieves the same result each time it is run
possible_k_values <- 1:15 # we will be checking k values of 1 through 15

# Elbow Method

# function that returns the total within-cluster sum of squares for the result of k-means cluste
ring with k clusters
get_total_wss <- function(k) {
  kmeans(standardized_imputed_df[, 3:43], k, nstart = 25)$tot.withinss
}
# calculate the total within-cluster sum of squares for k-means clustering with k values of 1 th
rough 15
total_wss_values <- map_dbl(possible_k_values, get_total_wss)
```

```
## Warning: did not converge in 10 iterations
```

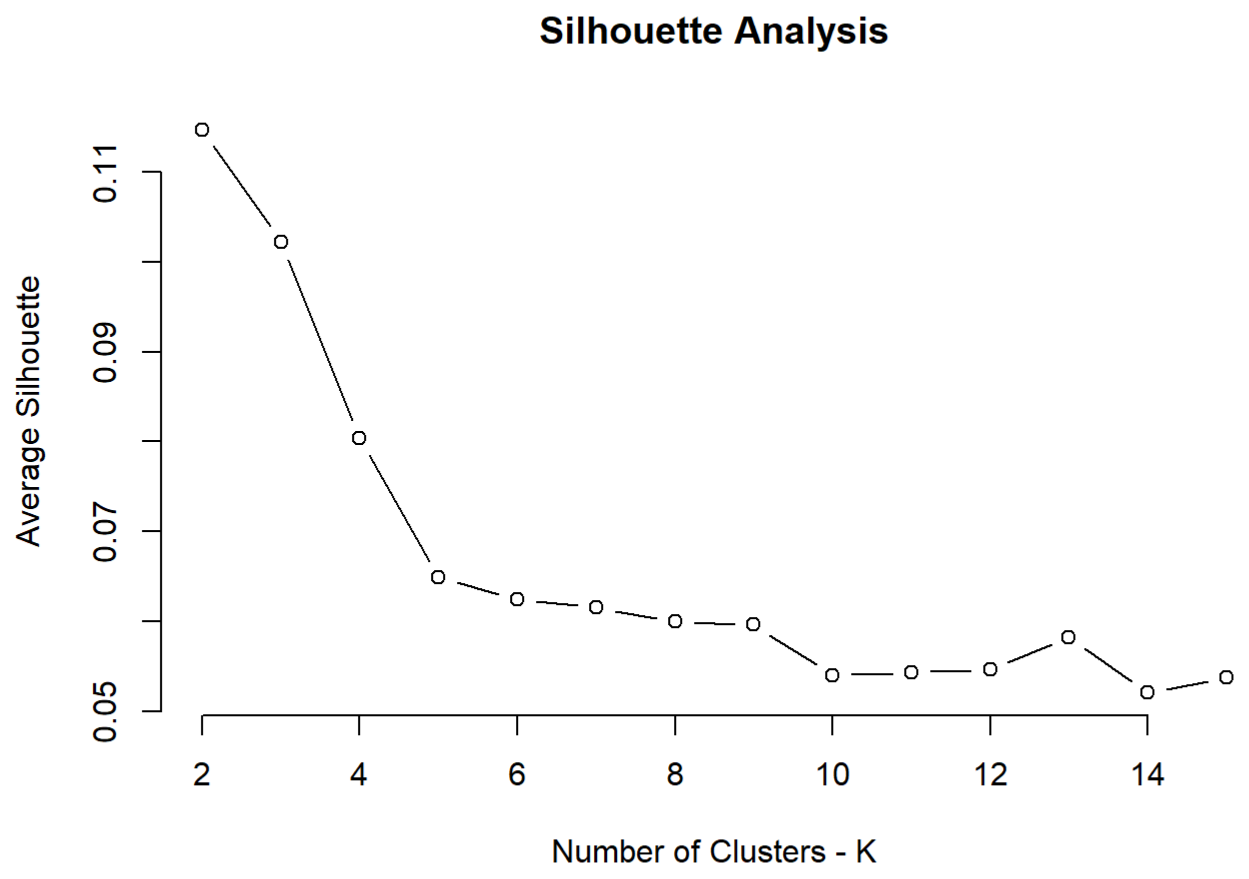
```
plot(possible_k_values, total_wss_values, type = "b", frame = FALSE, xlab= "Number of Clusters -
K",
     ylab = "Total Within-Cluster Sum of Squares", main = "The Elbow Method")
```



```
# Silhouette Analysis

possible_k_values <- 2:15 # skip k value of 1 so we don't get an error when calculating average
 silhouettes

# function that calculates the average silhouette for the result of k-means clustering with k cl
usters
get_avg_silhouette <- function(k) {
  clstrs <- kmeans(standardized_imputed_df[, 3:43], k, nstart = 25)
  silhouettes <- silhouette(clstrs$cluster, dist(standardized_imputed_df[, 3:43]))
  mean(silhouettes[, "sil_width"])
}

# calculate the average silhouette for k-means clustering with k values of 1 through 15
avg_silhouette_values <- map_dbl(possible_k_values, get_avg_silhouette)
```

```
## Warning: did not converge in 10 iterations
```

```
plot(possible_k_values, avg_silhouette_values, type = "b", frame = FALSE, xlab= "Number of Clust
ers - K",
     ylab = "Average Silhouette", main = "Silhouette Analysis")
```



After choosing 3 as the best possible value of k, k-means clustering is performed on the dataset of starting pitchers.

```
# based on the Elbow Method and Silhouette Analysis, we will make an educated guess of 3 as our
 value of k for k-means clustering
set.seed(123) # set seed

# k-means clustering with k = 3
clusters <- kmeans(standardized_imputed_df[, 3:43], 3, nstart = 25)

# add each player's cluster to the original dataframe and the imputed dataframe
df$cluster <- as.factor(clusters$cluster)
imputed_df$cluster <- as.factor(clusters$cluster)
df <- df[c(1, 2, 47, 3:46)]
imputed_df <- imputed_df[c(1, 2, 44, 3:43)]
```

In order to analyze the three clusters, cluster-wide means and medians are calculated for each feature. When examining the clusters, we can give descriptive names to each of the clusters based on their cluster-wide means and medians. Based on these pitchers' high velocities on all of their pitchers and high number of standardized runs saved with the fastball, we will call pitchers in Cluster 1 "power" pitchers. Based on these pitchers' low walk percentages and high first-strike percentages, we will call pitchers in Cluster 2 "control" pitchers. Lastly, based on their lack of identifying characteristics and the lack of features that they excel at, we will call pitchers in Cluster 3 "standard" pitchers.

```
# create seperate dataframes for each of the three clusters for both the original data and the i
mputed data
cluster1 <- subset(df, cluster == "1")
cluster2 <- subset(df, cluster == "2")
cluster3 <- subset(df, cluster == "3")
cluster1_imputed <- subset(imputed_df, cluster == "1")
cluster2_imputed <- subset(imputed_df, cluster == "2")
cluster3_imputed <- subset(imputed_df, cluster == "3")

# calculating each cluster's means for each attribute
cluster1_avgs <- colMeans(cluster1_imputed[sapply(cluster1_imputed, is.numeric)])
cluster2_avgs <- colMeans(cluster2_imputed[sapply(cluster2_imputed, is.numeric)])
cluster3_avgs <- colMeans(cluster3_imputed[sapply(cluster3_imputed, is.numeric)])
# calculating each cluster's medians for each attribute
cluster1_meds <- colMedians(cluster1_imputed[sapply(cluster1_imputed, is.numeric)])
cluster2_meds <- colMedians(cluster2_imputed[sapply(cluster2_imputed, is.numeric)])
cluster3_meds <- colMedians(cluster3_imputed[sapply(cluster3_imputed, is.numeric)])

# create a new dataframe of the per-cluster averages
cluster_avgs <- data.frame(cluster1_avgs, cluster2_avgs, cluster3_avgs)
# create a new dataframe of the per-cluster medians
cluster_meds <- data.frame(cluster1_meds, cluster2_meds, cluster3_meds)

cluster_avgs
```

```
##                          cluster1_avgs cluster2_avgs cluster3_avgs
## ip                        683.09891892  537.67280000  424.68292683
## k_percent                   0.22116216    0.17991200    0.15654268
## bb_percent                  0.07882703    0.07040800    0.08270732
## babip                       0.29451351    0.28896800    0.30396341
## gb_percent                  0.43500541    0.39451200    0.47727439
## ld_percent                  0.21203243    0.20294400    0.20615244
## fb_percent                  0.35299459    0.40255200    0.31657317
## iffb_percent                0.09725946    0.10710400    0.07805488
## soft_percent                0.18020000    0.18956000    0.17142683
## med_percent                 0.48901081    0.50538400    0.52091463
## hard_percent                0.33095676    0.30520000    0.30781707
## o_swing_percent             0.30880541    0.30200000    0.28249390
## swing_percent               0.46765946    0.46219200    0.44348171
## contact_percent             0.77600541    0.81369600    0.83118293
## zone_percent                0.43983784    0.45297600    0.44007927
## first_strike_percent        0.60875676    0.61384000    0.58756098
## swinging_strike_percent     0.10464324    0.08583200    0.07460366
## lob_percent                 0.73271892    0.72097600    0.69255488
## z_swing_percent             0.66978378    0.65546400    0.64817683
## o_contact_percent           0.63173514    0.69946400    0.70354268
## z_contact_percent           0.86129730    0.87766400    0.90245732
## pace                       22.85405405   21.28320000   21.57134146
## ev_imp                     88.35726126   88.38778667   89.09113821
## fa_percent                  0.56155676    0.53784800    0.58178049
## sl_percent_imp              0.17052252    0.13557760    0.11784512
## ct_percent_imp              0.07197333    0.09136800    0.10256748
## cb_percent_imp              0.12037550    0.11094880    0.12733211
## ch_percent_imp              0.12334775    0.15570240    0.11500813
## sf_percent_imp              0.08273550    0.08602027    0.08267683
## fa_standardized_runs       -0.11772973   -0.29592000   -0.63701220
## sl_standardized_runs_imp    0.40356036   -0.62431467   -0.60361382
## ct_standardized_runs_imp   -1.76672432   -1.86752533   -1.31353659
## cb_standardized_runs_imp   -0.33063423   -0.41814933   -0.86270325
## ch_standardized_runs_imp   -0.45291171   -0.31516267   -0.88946748
## sf_standardized_runs_imp   -0.50038198   -0.75966933   -1.96337805
## fa_velo                    93.00378378   89.24480000   91.10548780
## sl_velo_imp                84.82306306   81.40768000   83.44150407
## ct_velo_imp                89.14129730   86.10261333   87.99719512
## cb_velo_imp                78.52641441   74.23317333   77.11491870
## ch_velo_imp                85.29390991   80.83936000   83.51947154
## sf_velo_imp                85.66850450   81.53173333   84.14585366
```

```
cluster_meds
```

```
##                            cluster1_meds cluster2_meds cluster3_meds
## ip                          519.00000000  432.00000000   261.0500000
## k_percent                     0.21700000    0.17900000     0.1570000
## bb_percent                    0.07700000    0.07000000     0.0805000
## babip                         0.29500000    0.29100000     0.3040000
## gb_percent                    0.43700000    0.39000000     0.4770000
## ld_percent                    0.21100000    0.20300000     0.2055000
## fb_percent                    0.35200000    0.40500000     0.3145000
## iffb_percent                  0.09700000    0.10500000     0.0785000
## soft_percent                  0.17900000    0.18600000     0.1725000
## med_percent                   0.49000000    0.50900000     0.5220000
## hard_percent                  0.32400000    0.30500000     0.3050000
## o_swing_percent               0.30800000    0.30200000     0.2810000
## swing_percent                 0.46600000    0.46400000     0.4445000
## contact_percent               0.78000000    0.81400000     0.8305000
## zone_percent                  0.44200000    0.45300000     0.4400000
## first_strike_percent          0.60600000    0.61200000     0.5900000
## swinging_strike_percent       0.10300000    0.08600000     0.0750000
## lob_percent                   0.73300000    0.72300000     0.6960000
## z_swing_percent               0.67100000    0.65300000     0.6490000
## o_contact_percent             0.63800000    0.70000000     0.7005000
## z_contact_percent             0.86300000    0.88000000     0.9030000
## pace                         22.80000000   21.10000000    21.7000000
## ev_imp                       88.40000000   88.39333333    89.0000000
## fa_percent                    0.56800000    0.54300000     0.5865000
## sl_percent_imp                0.17100000    0.13800000     0.1085000
## ct_percent_imp                0.05413333    0.06146667     0.0783000
## cb_percent_imp                0.10600000    0.10800000     0.1220000
## ch_percent_imp                0.11800000    0.15100000     0.1070000
## sf_percent_imp                0.07233333    0.08240000     0.0720000
## fa_standardized_runs         -0.06000000   -0.33000000    -0.5850000
## sl_standardized_runs_imp      0.50000000   -0.26000000    -0.4763333
## ct_standardized_runs_imp     -0.72000000   -0.81733333    -0.7393333
## cb_standardized_runs_imp     -0.21600000   -0.42000000    -0.5500000
## ch_standardized_runs_imp     -0.24000000   -0.14000000    -0.6950000
## sf_standardized_runs_imp     -0.42133333   -0.46533333    -2.0253333
## fa_velo                      92.90000000   89.60000000    91.1500000
## sl_velo_imp                  84.90000000   81.90000000    83.4000000
## ct_velo_imp                  89.00000000   86.25333333    88.1000000
## cb_velo_imp                  78.60000000   74.70000000    77.4000000
## ch_velo_imp                  85.10000000   81.20000000    83.7500000
## sf_velo_imp                  85.80000000   81.29333333    84.2033333
```

After k-means clustering has been performed on the starting pitchers dataset containing only the pitching style metrics, pitching success metrics; like ERA, FIP, and WHIP; for the same set of starting pitchers over the same timeframe will be read into a new dataframe, which will also identify each starting pitcher's cluster.

```
# read the pitcher success data file into a dataframe
sp_success <- read.csv("sp_success_data 2010-2019.csv", header = TRUE, check.names = TRUE)
colnames(sp_success)[1] <- "name"
sp_success <- sp_success %>% rename(team = Team, era = ERA, siera = SIERA, whip = WHIP, fip = FI
P, xfip = xFIP,
                                   war = WAR, rar = RAR, wpa = WPA)
sp_success$cluster <- as.factor(clusters$cluster)
sp_success <- sp_success[c(1, 2, 11, 3:10)]
head(sp_success)
```

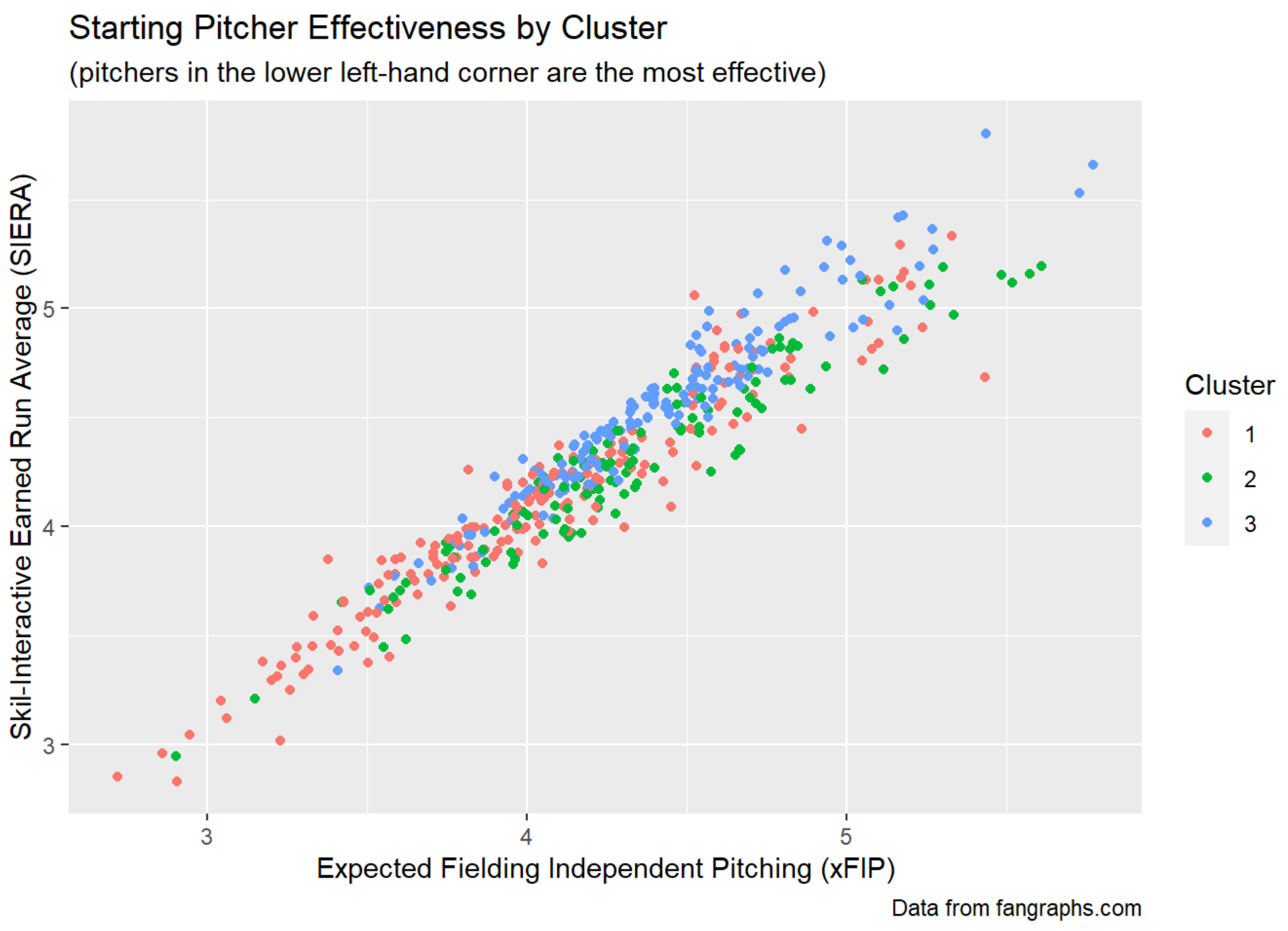
```
##              name    team cluster  era siera whip  fip xfip  war   rar   wpa
## 1 Clayton Kershaw Dodgers       1 2.31  2.96 0.96 2.64 2.86 59.0 508.8 38.21
## 2  Jose Fernandez Marlins       1 2.58  2.85 1.05 2.44 2.72 14.5 125.2  8.95
## 3    Jacob deGrom    Mets       1 2.62  3.20 1.05 2.78 3.04 31.5 281.1 18.47
## 4     Mike Soroka  Braves       1 2.79  4.26 1.15 3.38 3.82  4.6  44.8  3.25
## 5  Walker Buehler Dodgers       1 2.85  3.40 0.99 2.97 3.28  8.2  77.6  5.26
## 6       Rich Hill   - - -       2 2.92  3.45 1.05 3.43 3.55 10.0  92.6  5.24
```

In order to analyze the three clusters based on pitching success, cluster-wide means and medians are calculated for each feature.

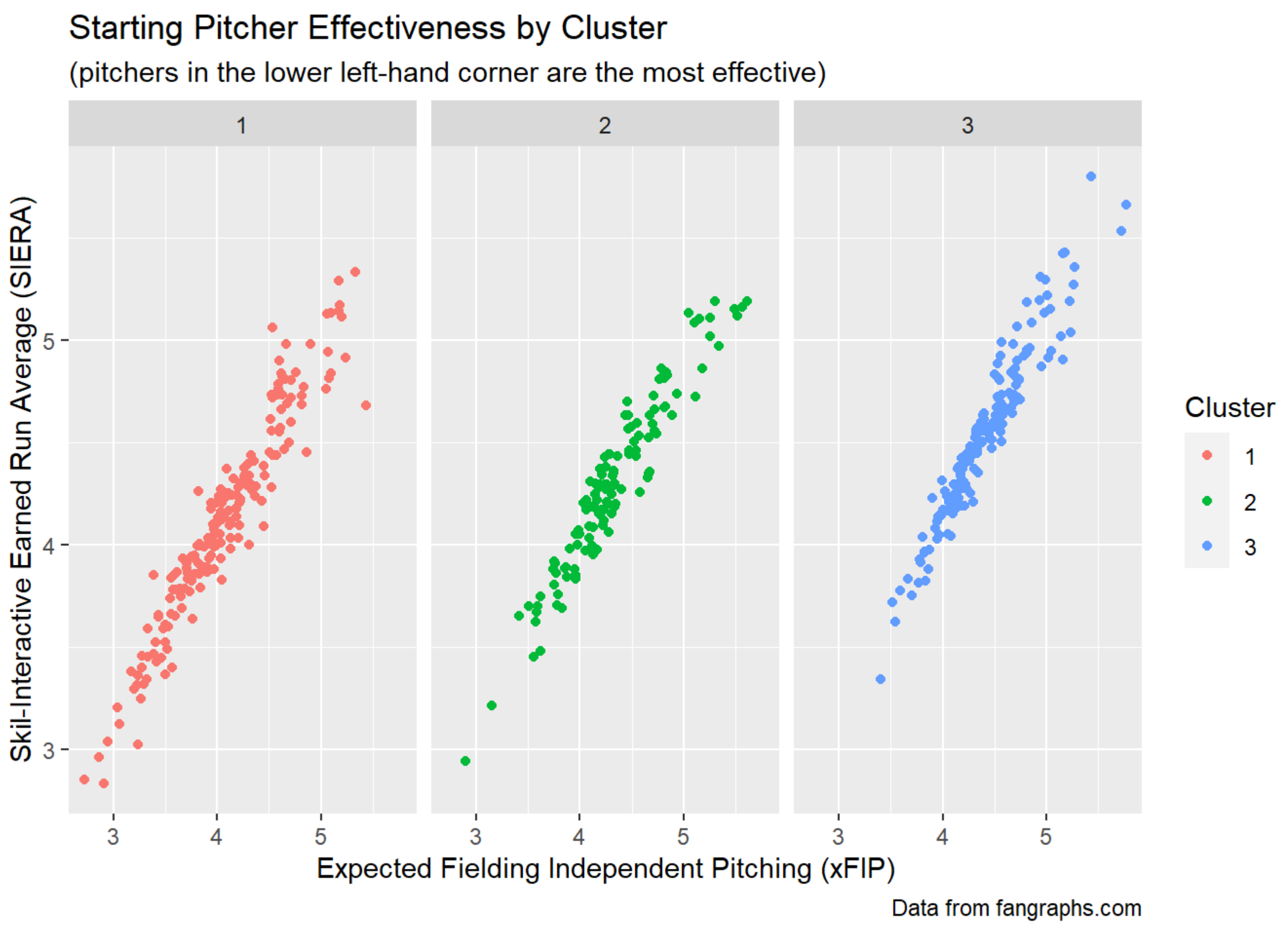
```
# create seperate dataframes for each of the three clusters for the pitcher success data
cluster1_success <- subset(sp_success, cluster == "1")
cluster2_success <- subset(sp_success, cluster == "2")
cluster3_success <- subset(sp_success, cluster == "3")

# calculating each cluster's means for each attribute
cluster1_success_avgs <- colMeans(cluster1_success[sapply(cluster1_success, is.numeric)])
cluster2_success_avgs <- colMeans(cluster2_success[sapply(cluster2_success, is.numeric)])
cluster3_success_avgs <- colMeans(cluster3_success[sapply(cluster3_success, is.numeric)])

# create a new dataframe of the per-cluster averages
cluster_success_avgs <- data.frame(cluster1_success_avgs, cluster2_success_avgs, cluster3_succes
s_avgs)
```

Two of the best metrics to quantify a pitcher's performance are Skill-Interactive Earned Run Average (SIERA) and Expected Fielding Independent Pitching (xFIP) because they attempt to eliminate factors the pitcher can't control himself, such as the defense in xFIP and the type of ball in play for SIERA. More information on each of these metrics can be found here: xFIP - https://library.fangraphs.com/pitching/xfip/ (https://library.fangraphs.com/pitching/xfip/), SIERA - https://library.fangraphs.com/pitching/siera/ (https://library.fangraphs.com/pitching/siera/).

```
# creating a graphic to measure the success of each of the three clusters
ggplot(data = sp_success) +
  geom_point(mapping = aes(x = xfip, y = siera, color = cluster), position = "jitter") +
  labs(title = "Starting Pitcher Effectiveness by Cluster",
       subtitle = "(pitchers in the lower left-hand corner are the most effective)",
       x = "Expected Fielding Independent Pitching (xFIP)",
       y = "Skil-Interactive Earned Run Average (SIERA)",
       color = "Cluster",
       caption = "Data from fangraphs.com")
```



```
ggplot(data = sp_success) +
  geom_point(mapping = aes(x = xfip, y = siera, color = cluster), position = "jitter") +
  facet_wrap(~ cluster, nrow = 1) +
  labs(title = "Starting Pitcher Effectiveness by Cluster",
       subtitle = "(pitchers in the lower left-hand corner are the most effective)",
       x = "Expected Fielding Independent Pitching (xFIP)",
       y = "Skil-Interactive Earned Run Average (SIERA)",
       color = "Cluster",
       caption = "Data from fangraphs.com")
```



Based on these graphics attempting to quantify a pitcher's success through SIERA and xFIP, it can be seen that as a general rule, pitchers in Cluster 1 ("power" pitchers) are the most successful, followed by Cluster 2 ("control" pitchers), and then Cluster 3 ("standard" pitchers).

MLB Starting Pitching Rotations Analysis

Tyler Fuelling

August 23, 2020

Now that k-means clustering has been performed on the dataset of MLB starting pitchers from 2010 through 2019, the next step is to examine the compositions of MLB starting pitching rotations during that timeframe.

We begin by loading the data into a dataframe. The dataset contains data for all 300 MLB teams from the 2010 through 2019 seasons, and was created with data from https://www.fangraphs.com/ (https://www.fangraphs.com/) and https://www.baseball-reference.com/ (https://www.baseball-reference.com/).

```
# read the mlb starting pitcher rotations data file into a dataframe
df <- read_excel("sp_rotations 2010-2019.xlsx")
# examine the dataframe's dimensions
dim(df)
```

```
## [1] 300  22
```

The features of this dataset contain team success data; such as wins, losses, starting pitching SIERA, and starting pitching xFIP; as well as rotation composition data. The columns titled "spx_cluster" represent the cluster number of the starting pitcher who threw the "xth" most innings for that team that year. If one of these datapoints is NA, it means that that pitcher did not throw at least 100 innings from 2010-2019, and thus did not qualify to be in the original dataset. The columns titled "clusterx_sps" represents the number of pitchers from cluster x in that team's five-man starting pitching rotation. Lastly, the column titled "rotation_composition" contains an encoding of the team's starting pitching rotation. The first digit is the number of pitchers belonging to Cluster 1 that are a part of the team's five-man starting pitching rotation, the second digit is the number of pitchers belonging to Cluster 2 that are a part of the team's five-man starting pitching rotation, and the third digit is the number of pitchers belonging to Cluster 3 that are a part of the team's five-man starting pitching rotation. For example, a rotation composition of "221" means that that team has a starting pitching rotation consisting of 2 pitchers from Cluster 1, 2 pitchers from Cluster 2, and 1 pitcher from Cluster 3.

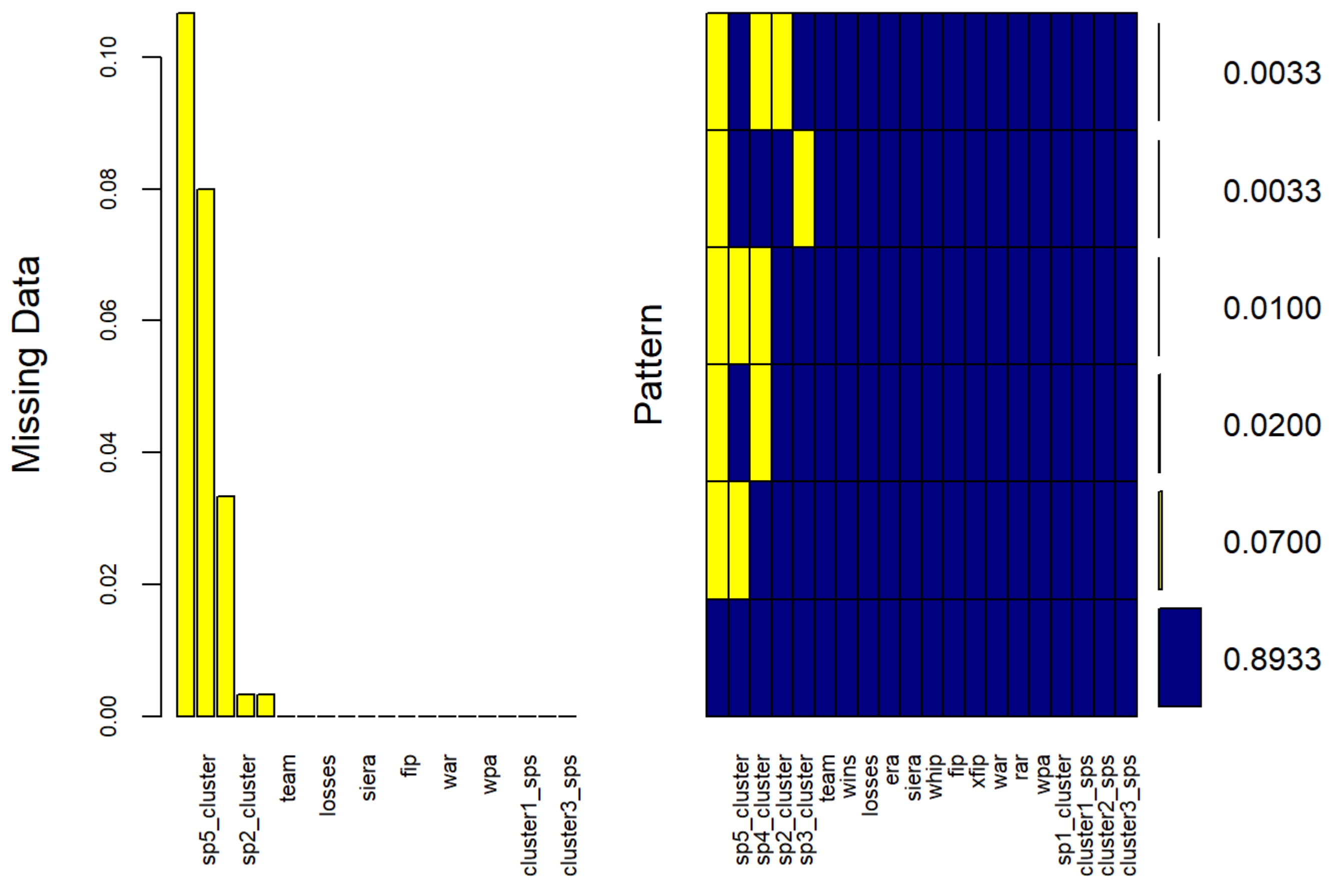
```
# removing the season and franchise columns from the dataframe
df <- subset(df, select = -c(Season, Franchise))
# renaming the columns of the the df for easier use
df <- df %>% rename(team = Team, wins = Wins, losses = Losses, era = ERA, siera = SIERA, whip =
 WHIP,
                   fip = FIP, xfip = xFIP, war = WAR, rar = RAR, wpa = WPA, sp1_cluster = `SP1
 Cluster`,
                   sp2_cluster = `SP2 Cluster`, sp3_cluster = `SP3 Cluster`, sp4_cluster = `SP4
Cluster`,
                   sp5_cluster = `SP5 Cluster`, cluster1_sps = `Cluster 1 Pitchers`,
                   cluster2_sps = `Cluster 2 Pitchers`, cluster3_sps =`Cluster 3 Pitchers`,
                   rotation_composition = `Rotation Composition` )
```

```
# examine the dataframe
head(df)
```

```
## # A tibble: 6 x 20
##   team   wins losses   era siera  whip   fip  xfip   war   rar   wpa sp1_cluster
##   <chr> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>       <dbl>
## 1 2019~    72     90  5.64  4.8   1.39  5.41  4.95   3.2  30.9 -6.59           1
## 2 2018~    80     82  4.34  4.15  1.31  4.39  4.11   9    84.4 -0.54           1
## 3 2017~    80     82  4.38  4.7   1.33  4.93  4.7    5    47.9 -3.01           2
## 4 2016~    74     88  4.6   4.67  1.43  4.79  4.79   6.9  64.1 -6.01           2
## 5 2015~    85     77  3.98  4.27  1.25  4.26  4.42   9.1  86.1 -0.08           1
## 6 2014~    98     64  3.62  3.86  1.22  3.68  3.86  11.2  99.7 -0.05           2
## # ... with 8 more variables: sp2_cluster <dbl>, sp3_cluster <dbl>,
## #   sp4_cluster <dbl>, sp5_cluster <dbl>, cluster1_sps <dbl>,
## #   cluster2_sps <dbl>, cluster3_sps <dbl>, rotation_composition <chr>
```

Because some teams have a member of their starting pitching rotation who does not meet the minimum qualifications of having pitched at least 100 innings during the seasons from 2010 through 2019, there are many NA values present in the dataset. In this case, the missing data will be handled by removing all of the rows that contain an NA value from the datset. This removes 32 teams from the dataset, leaving 268 teams in the rotation composition dataset.

```
# examining the NA values present in the dataframe
aggr(df, col=c('navyblue', 'yellow'), numbers = TRUE, sortVars = TRUE, labels = names(df), cex.a
xis = .7,
     gap = 3, ylab = c("Missing Data", "Pattern"))
```



```
##
##  Variables sorted by number of missings:
##             Variable       Count
##  rotation_composition 0.106666667
##           sp5_cluster 0.080000000
##           sp4_cluster 0.033333333
##           sp2_cluster 0.003333333
##           sp3_cluster 0.003333333
##                  team 0.000000000
##                  wins 0.000000000
##                losses 0.000000000
##                   era 0.000000000
##                 siera 0.000000000
##                  whip 0.000000000
##                   fip 0.000000000
##                  xfip 0.000000000
##                   war 0.000000000
##                   rar 0.000000000
##                   wpa 0.000000000
##           sp1_cluster 0.000000000
##          cluster1_sps 0.000000000
##          cluster2_sps 0.000000000
##          cluster3_sps 0.000000000
```

```
# remove all of the rows in the dataframe containing NA values
df <- na.omit(df)
table(is.na(df))
```

```
##
## FALSE
##  5360
```

```
# now the dataframe consists solely of teams with a complete rotation from the group of clustere
s pitchers
dim(df)
```

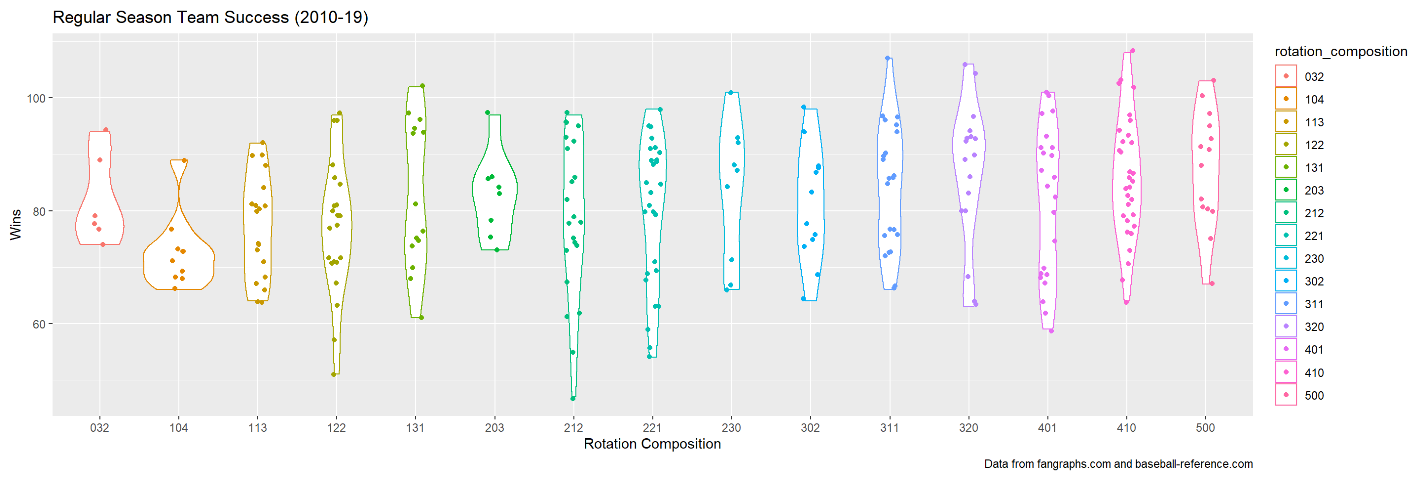
```
## [1] 268  20
```

Because there were such few occurrences of these rotation compositions throughout the past ten seasons, all of the rotation compositions that occurred less than five times throughout that timeframe were removed from the dataset. Because there are 21 possible rotation compositions, each rotation could be expected to occur about 8 percent of the time. Those occurring less than five times occurred less than 2 percent of the time, which was determined to be too small of a sample size to draw any conclusions about those specific rotation compositions.

```
# removing rotation compositions from the dataframe that occurred less than 5 times in the past
 ten years
df <- df %>%
  group_by(rotation_composition) %>%
  filter(n() >= 5)
```

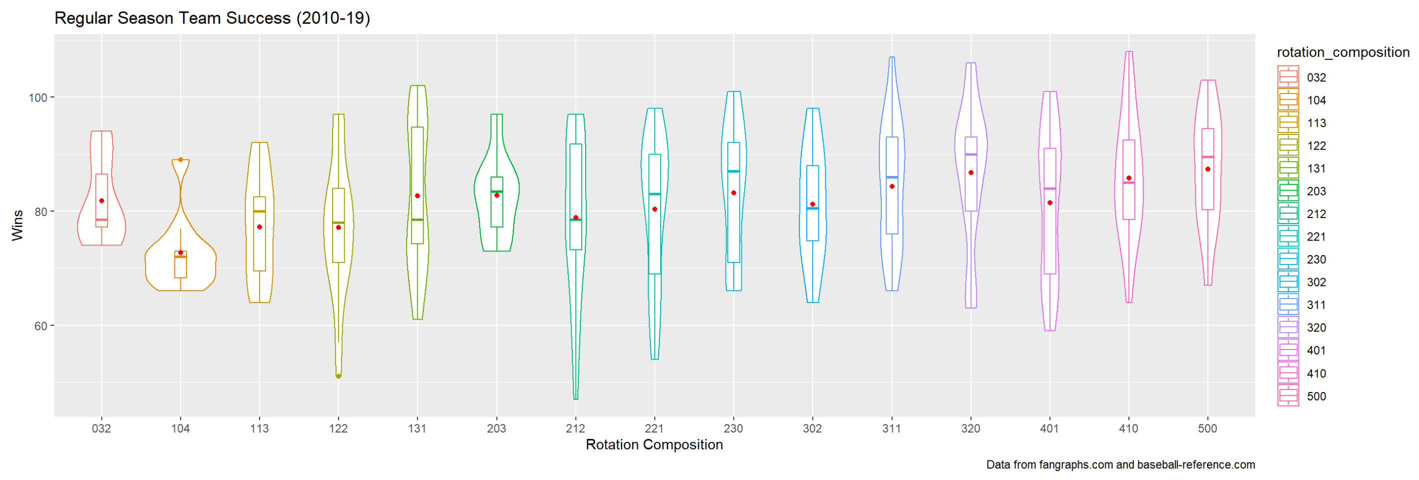
Team success by starting pitching rotation composition can be conveyed through violin plots. For the violin plot using data points, each data point inside the violins represents one team which had the rotation composition indicated by the point's color.

```
# creating a graphic to measure the team success of all teams from the last ten seasons, while e
xamining different compositions of starting pitcher rotations
ggplot(data = df, mapping = aes(x = rotation_composition, y = wins, color = rotation_compositio
n)) +
  geom_violin(scale = "area") +
  labs(title = "Regular Season Team Success (2010-19)",
       x = "Rotation Composition",
       y = "Wins",
       caption = "Data from fangraphs.com and baseball-reference.com") +
  geom_jitter(position = position_jitter(0.1))
```



For the violin plot using boxplots, the boxplot inside of each violin represents the minimum, first quartile, median, third quartile, and maximum of wins for teams that had the rotation composition indicated by the color of the violin. The red dot represents the mean number of wins for teams that had the rotation composition indicated by the color of the violin.

```
# creating a graphic to measure the team success of all teams from the last ten seasons, while e
xamining different compositions of starting pitcher rotations
ggplot(data = df, mapping = aes(x = rotation_composition, y = wins, color = rotation_compositio
n)) +
  geom_violin(scale = "area") +
  labs(title = "Regular Season Team Success (2010-19)",
       x = "Rotation Composition",
       y = "Wins",
       caption = "Data from fangraphs.com and baseball-reference.com") +
  geom_boxplot(width = 0.15) +
  stat_summary(fun = mean, geom = "point", color = "red")
```



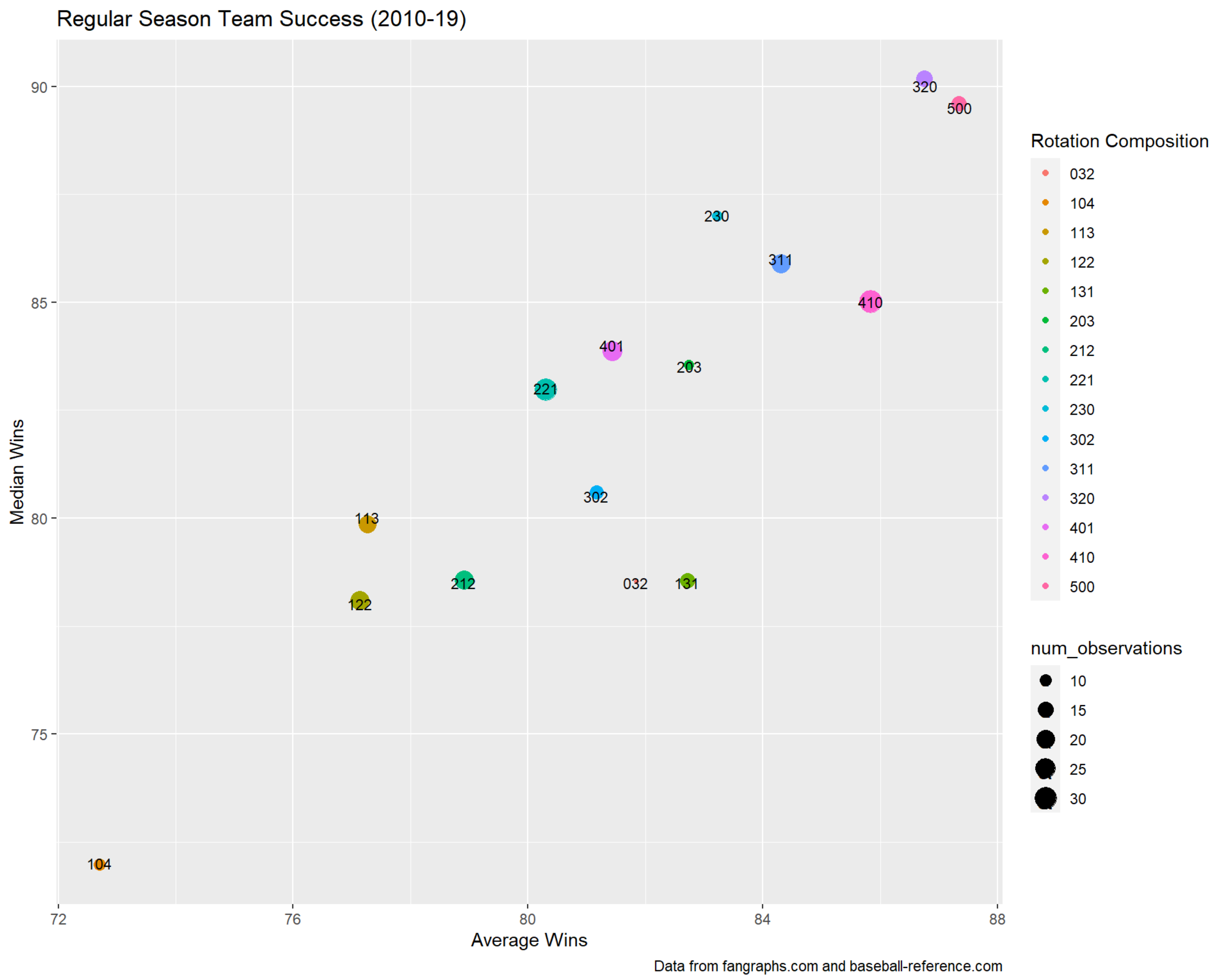
After examining the violin plots, it can clearly be seen that the most successful teams over the last ten seasons have had rotation compositions of 311, 320, 410, and 500. Now, instead of examining only team-wide success, the success of the starting pitching rotation independent of the rest of the team will be examined as well. In the same way that the success of each cluster was measured using SIERA and xFIP, the average success of each rotation composition will be measured using SIERA and xFIP as well.

```
rotations_summary <- df %>%
  group_by(rotation_composition) %>%
  summarise(num_observations = n(),
            avg_wins = mean(wins),
            median_wins = median(wins),
            avg_siera = mean(siera),
            avg_xfip = mean(xfip),
            .groups = "keep")

rotations_summary
```

```
## # A tibble: 15 x 6
## # Groups:   rotation_composition [15]
##    rotation_composition num_observations avg_wins median_wins avg_siera avg_xfip
##    <chr>                           <int>    <dbl>       <dbl>     <dbl>    <dbl>
##  1 032                                 6     81.8        78.5      4.09     4.00
##  2 104                                10     72.7        72        4.40     4.28
##  3 113                                19     77.3        80        4.16     4.06
##  4 122                                22     77.1        78        4.21     4.16
##  5 131                                14     82.7        78.5      4.06     4.02
##  6 203                                 8     82.8        83.5      4.26     4.15
##  7 212                                22     78.9        78.5      4.22     4.18
##  8 221                                29     80.3        83        4.21     4.19
##  9 230                                 9     83.2        87        4.04     3.98
## 10 302                                12     81.2        80.5      4.10     3.98
## 11 311                                22     84.3        86        4.15     4.10
## 12 320                                17     86.8        90        4.02     4.01
## 13 401                                23     81.4        84        4.34     4.26
## 14 410                                31     85.8        85        4.03     3.98
## 15 500                                14     87.4        89.5      3.97     3.89
```

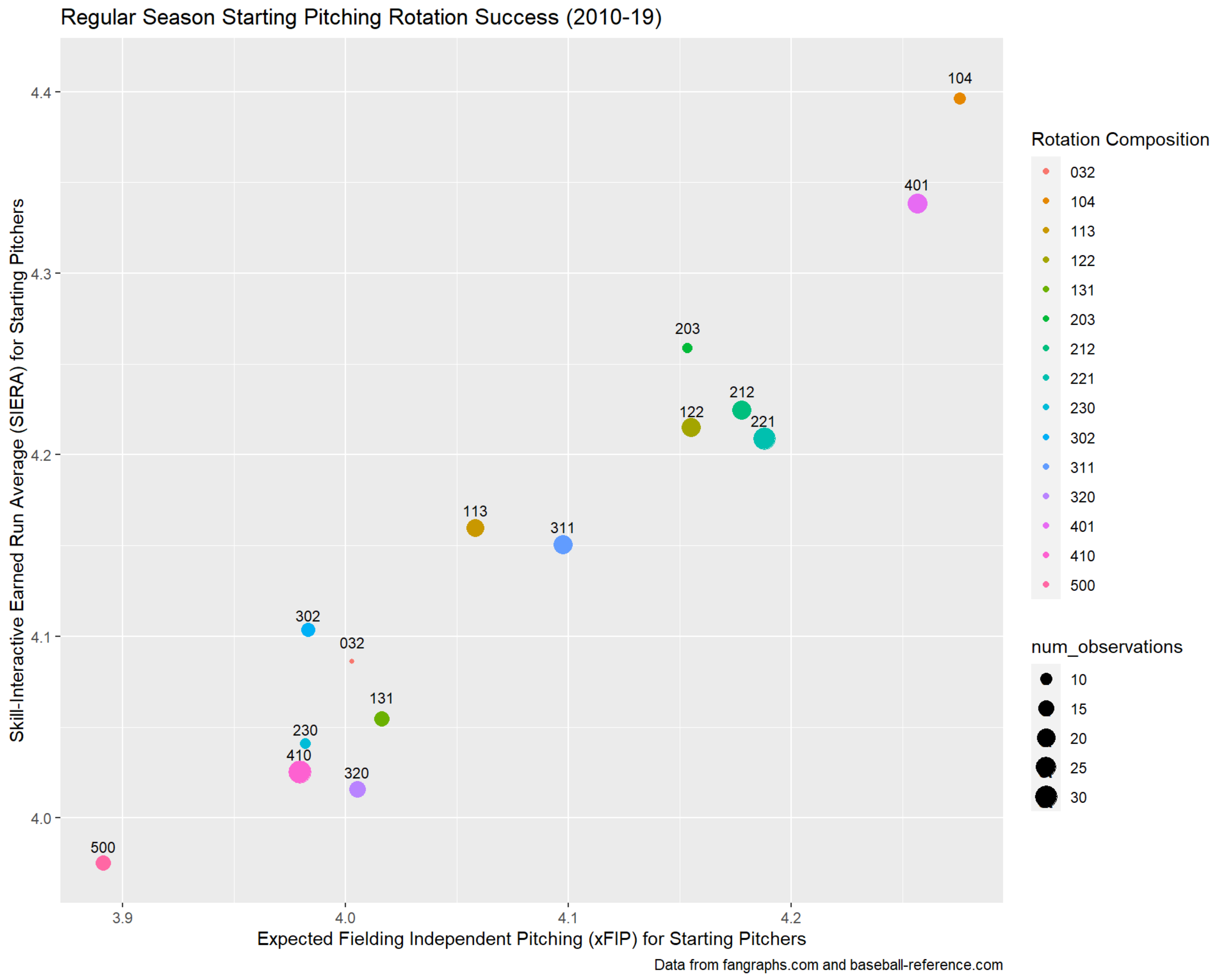
In addition to the violin plots, team successs by rotation composition can be conveyed through a scatterplot featuring average wins for teams with that rotation composition and median wins for teams with that rotation composition as variables. The rotation composition is indicated by the color of the points and the number of observations is indicated by the size of the points. For this scatterplot the points closest to the top-right corner of the scatterplot are the most successful, which are rotation compositions 320 and 500.

```
# creating a graphic to measure the team success of all teams from the last ten seasons, while e
xamining different compositions of starting pitcher rotations
ggplot(data = rotations_summary) +
  geom_point(mapping = aes(x = avg_wins, y = median_wins, color = rotation_composition, size = n
um_observations), position = "jitter") +
  labs(title = "Regular Season Team Success (2010-19)",
       color = "Rotation Composition",
       x = "Average Wins",
       y = "Median Wins",
       caption = "Data from fangraphs.com and baseball-reference.com") +
  geom_text(aes(x = avg_wins, y = median_wins, label = rotation_composition, size = 10), nudge_y
= 0.01)
```



In addition to measuring overall team success, the succcess of the starting rotation independent of the rest of the team can be conveyed through a scatterplot featuring average SIERA for teams with that rotation composition and average xFIP for teams with that rotation composition as variables. The rotation composition is indicated by the color of the points and the number of observations is indicated by the size of the points. For this scatterplot the points closest to the bottom-left corner of the scatterplot are the most successful, which is rotation composition 500.

```
# creating a graphic to measure the success of starting pitching rotations from the last ten sea
sons, while examining the different compositions of the rotations
ggplot(data = rotations_summary) +
  geom_point(mapping = aes(x = avg_xfip, y = avg_siera, color = rotation_composition, size = num
_observations), position = "jitter") +
  labs(title = "Regular Season Starting Pitching Rotation Success (2010-19)",
       color = "Rotation Composition",
       x = "Expected Fielding Independent Pitching (xFIP) for Starting Pitchers",
       y = "Skill-Interactive Earned Run Average (SIERA) for Starting Pitchers",
       caption = "Data from fangraphs.com and baseball-reference.com") +
  geom_text(aes(x = avg_xfip, y = avg_siera, label = rotation_composition, size = 10), nudge_y =
0.01)
```



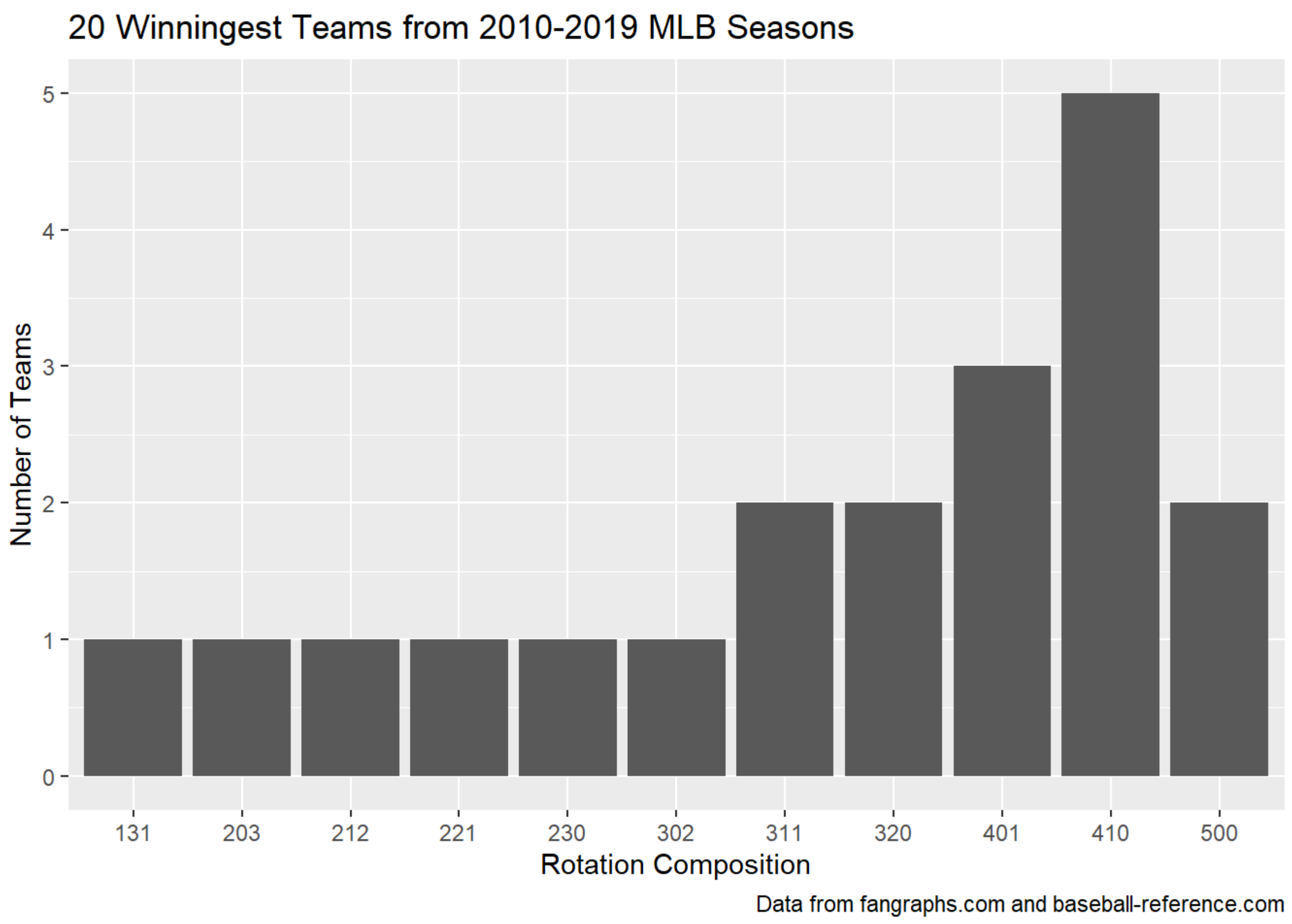
Based on analysis to this point, it seems that on average, the starting pitching rotation composition of 500 is the most successful. Now, instead of looking at average success levels, each rotation composition's proclivity towards the highest levels of success will be examined. In order to do so, the 20 winningest teams from the past ten seasons are graphed by their rotation composition. It can be seen that 25% of these teams have a rotation composition of 410, which is the highest percentage of any of the rotation compositions.

```
# sort the df by wins
df_sorted_by_wins <- df[order(-df$wins),]
# take the top 20 rows of the df (top ~7.5% of the teams in the past 10 years)
top_teams_by_wins <- df_sorted_by_wins[1:20,]
```

```
# graphing the different rotation compositions in the 20 most successful teams
winningest_teams_by_rotation_composition <- top_teams_by_wins %>%
  group_by(rotation_composition) %>%
  summarise(num_observations = n(),
            .groups = "keep")
winningest_teams_by_rotation_composition
```

```
## # A tibble: 11 x 2
## # Groups:   rotation_composition [11]
##    rotation_composition num_observations
##    <chr>                           <int>
##  1 131                                 1
##  2 203                                 1
##  3 212                                 1
##  4 221                                 1
##  5 230                                 1
##  6 302                                 1
##  7 311                                 2
##  8 320                                 2
##  9 401                                 3
## 10 410                                 5
## 11 500                                 2
```

```
ggplot(data = winningest_teams_by_rotation_composition) +
  geom_bar(mapping = aes(x = rotation_composition, y = num_observations), stat = "identity") +
  labs(title = "20 Winningest Teams from 2010-2019 MLB Seasons",
       x = "Rotation Composition",
       y = "Number of Teams",
       caption = "Data from fangraphs.com and baseball-reference.com")
```



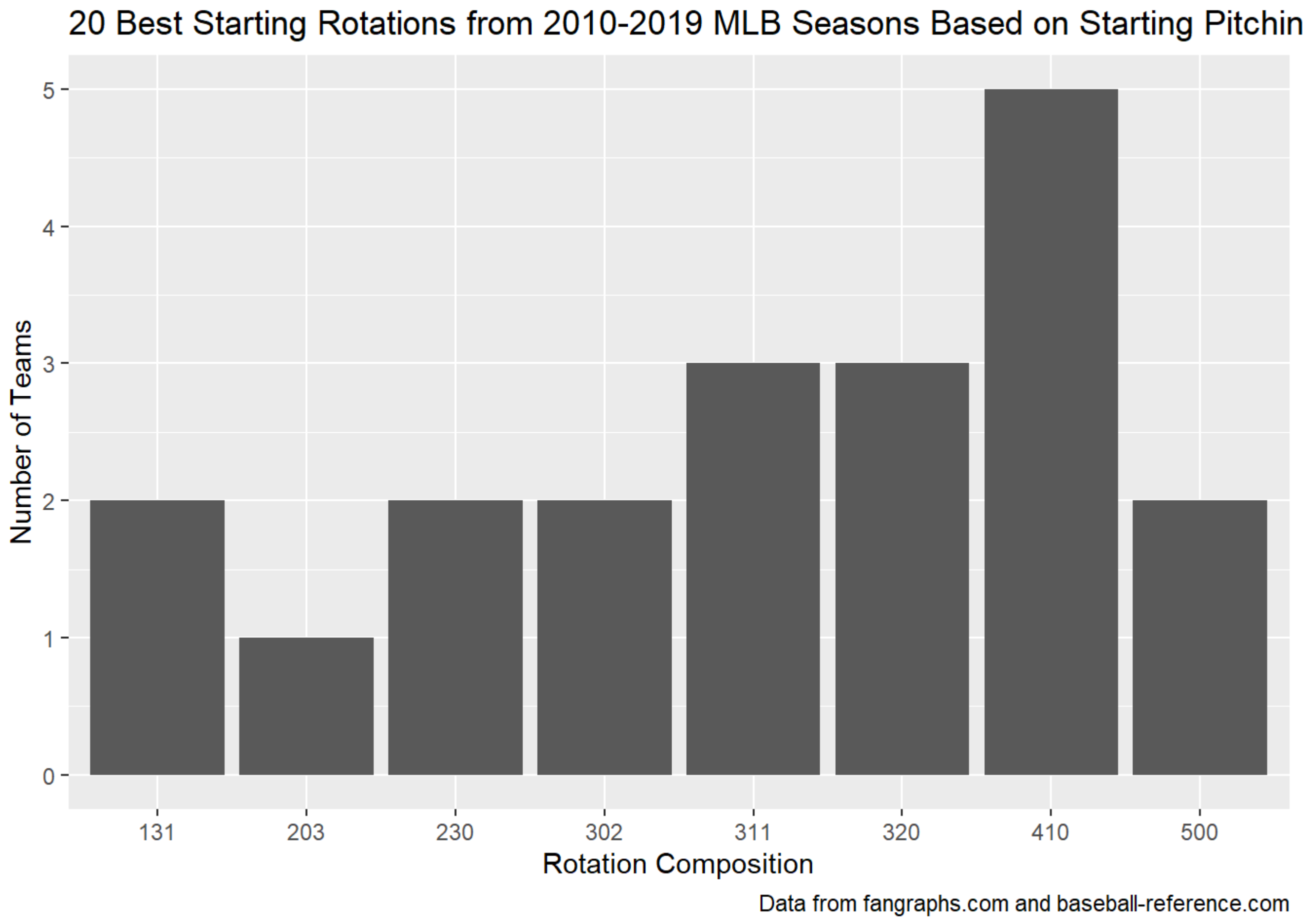
Next, in order to to examine each rotation composition's proclivity to the highest levels of starting pitching rotation success independent of the rest of the team, the 20 teams with the best starting pitching xFIP the from the past ten seasons are graphed by their rotation composition. It can be seen that 25% of these teams have a rotation composition of 410, which is the highest percentage of any of the rotation compositions.

```
# sort the df by SP xFIP
df_sorted_by_xfip <- df[order(df$xfip),]
# take the top 20 rows of the df (top ~7.5% of the teams in the past 10 years in sp xFIP)
top_teams_by_xfip <- df_sorted_by_xfip[1:20,]
```

```
# graphing the different rotation compositions in the 20 most successful teams by SP xFIP
lowest_xfip_teams_by_rotation_composition <- top_teams_by_xfip %>%
  group_by(rotation_composition) %>%
  summarise(num_observations = n(),
            .groups = "keep")
lowest_xfip_teams_by_rotation_composition
```

```
## # A tibble: 8 x 2
## # Groups:   rotation_composition [8]
##   rotation_composition num_observations
##   <chr>                           <int>
## 1 131                                 2
## 2 203                                 1
## 3 230                                 2
## 4 302                                 2
## 5 311                                 3
## 6 320                                 3
## 7 410                                 5
## 8 500                                 2
```

```
ggplot(data = lowest_xfip_teams_by_rotation_composition) +
  geom_bar(mapping = aes(x = rotation_composition, y = num_observations), stat = "identity") +
  labs(title = "20 Best Starting Rotations from 2010-2019 MLB Seasons Based on Starting Pitching xFIP",
       x = "Rotation Composition",
       y = "Number of Teams",
       caption = "Data from fangraphs.com and baseball-reference.com")
```



After analyzing the data, it can be seen that the rotation composition which consistently leads to both the highest average team success and highest average starting pitching rotation success independent of the rest of the team is 500, or 5 "power" pitchers. However, the rotation composition which allows for the possibility of the highest levels of team success and the highest levels of starting pitching rotation success independent of the rest of the team is 410, or 4 "power" pitchers and 1 "control" pitcher. Either way, both of these rotation compositions contain at least 4 "power pitchers", showing that the "power pitcher" is extremely valuable in MLB starting pitching rotations. Nonetheless, the fact that rotation composition 410 has lead teams to extremely high levels of success during the past ten seasons displays the fact that there is value in having a "control pitcher" as a member of the starting pitching rotation as well. However, it is clear from the data that the "power pitcher" is king in the MLB, at least when it comes to putting together a team's starting rotation.