Final Summary

Brendan Karadenes

Introduction

The purpose of this project was to examine how Major League Baseball (MLB) pitchers react to different in-game scenarios. To find an answer to this question we looked at each pitcher's pitch arsenals and tendencies. To maintain consistency, we used only qualifying pitchers (1 inning pitched per team game) from the 2023 season. All of the data was retrieved from MLB's Statcast database where we were able to see each pitcher's pitch from the season.In addition, the result of each pitch and other advanced metrics were available to view. To begin the investigation we separated each pitcher into 5 different clusters based on their pitching arsenal. We looked at pitch selection for each group and were able to visualize how each type of pitcher reacted to different situations, like different outs and counts. We compared pitchers both within and between different clusters, making a small shiny app to visualize this. Then, we built various models for each pitcher to see if certain statistics, like Earned Run Average (ERA) could predict how much the pitcher deviated from their primary pitch.

Step 1: Pitcher Clustering

To begin the project, we gathered each qualified pitcher and their 2023 season statistics into a single dataset. The list of pitchers can be viewed below:

```r
library(gt)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
```

```
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becon
```

```r
players <- data.frame(
  Player = c("Webb, Logan",
  "Gallen, Zac",
  "Cole, Gerrit",
  "Mikolas, Miles",
  "Bassitt, Chris",
  "Valdez, Framber",
  "Castillo, Luis",
  "Keller, Mitch",
  "López, Pablo",
  "Burnes, Corbin",
  "Nola, Aaron",
  "Gibson, Kyle",
  "Wheeler, Zack",
  "Gilbert, Logan",
  "Kirby, George",
  "Berríos, José",
  "Montgomery, Jordan",
  "Strider, Spencer",
  "Gausman, Kevin",
  "Alcantara, Sandy",
  "Giolito, Lucas",
  "Gray, Sonny",
  "Lynn, Lance",
  "Corbin, Patrick",
  "Snell, Blake",
  "Luzardo, Jesús",
  "Eflin, Zach",
  "Kelly, Merrill",
  "Lyles, Jordan",
  "Oviedo, Johan",
  "Cease, Dylan",
  "Elder, Bryce",
  "Steele, Justin",
  "Dunning, Dane",
  "Kremer, Dean",
  "Walker, Taijuan",
  "Sears, JP",
  "Bradish, Kyle",
  "Kikuchi, Yusei",
  "Peralta, Freddy",
  "Morton, Charlie",
  "Verlander, Justin",
  "Javier, Cristian")
)
players_table <- players %>%
  gt() %>%
  tab_header(title = "Qualified Pitchers")
players_table
```

The code below reads in player statistics from qualified pitchers in 2023 and places them in a data frame.

```r
qual_pitchers <- read.csv("qual_pitchers.csv")
```

```r
qual_pitchers %>%
  summarise(unique_count = n_distinct(pitcher))
```

```
  unique_count
1          446
```

```r
filtered_pitchers <- qual_pitchers %>%
filter(player_name %in% c(
  "Webb, Logan",
  "Gallen, Zac",
  "Cole, Gerrit",
  "Mikolas, Miles",
  "Bassitt, Chris",
  "Valdez, Framber",
  "Castillo, Luis",
  "Keller, Mitch",
  "López, Pablo",
  "Burnes, Corbin",
  "Nola, Aaron",
  "Gibson, Kyle",
  "Wheeler, Zack",
  "Gilbert, Logan",
```

Qualified Pitchers

Player
Webb, Logan
Gallen, Zac
Cole, Gerrit
Mikolas, Miles
Bassitt, Chris
Valdez, Framber
Castillo, Luis
Keller, Mitch
López, Pablo
Burnes, Corbin
Nola, Aaron
Gibson, Kyle
Wheeler, Zack
Gilbert, Logan
Kirby, George
Berríos, José
Montgomery, Jordan
Strider, Spencer
Gausman, Kevin
Alcantara, Sandy
Giolito, Lucas
Gray, Sonny
Lynn, Lance
Corbin, Patrick
Snell, Blake
Luzardo, Jesús
Eflin, Zach
Kelly, Merrill
Lyles, Jordan
Oviedo, Johan
Cease, Dylan
Elder, Bryce
Steele, Justin
Dunning, Dane
Kremer, Dean
Walker, Taijuan
Sears, JP
Bradish, Kyle
Kikuchi, Yusei
Peralta, Freddy
Morton, Charlie
Verlander, Justin
Javier, Cristian

```
  "Kirby, George",
  "Berríos, José",
  "Montgomery, Jordan",
  "Strider, Spencer",
  "Gausman, Kevin",
  "Alcantara, Sandy",
  "Giolito, Lucas",
  "Gray, Sonny",
  "Lynn, Lance",
  "Corbin, Patrick",
  "Snell, Blake",
  "Luzardo, Jesús",
  "Eflin, Zach",
  "Kelly, Merrill",
  "Lyles, Jordan",
  "Oviedo, Johan",
  "Cease, Dylan",
  "Elder, Bryce",
  "Steele, Justin",
  "Dunning, Dane",
  "Kremer, Dean",
  "Walker, Taijuan",
  "Sears, JP",
  "Bradish, Kyle",
  "Kikuchi, Yusei",
  "Peralta, Freddy",
  "Morton, Charlie",
  "Verlander, Justin",
  "Javier, Cristian"
))
```

To prepare for the clustering we gathered the proportion of each pitch type for each pitcher. This way, we can compare pitch selection between them. We also need to create the `count` variable to see in what count each pitch was thrown.

```
filtered_pitchers <- filtered_pitchers %>%
  unite(count, balls, strikes, sep = "-")
```

```
grouped_pitches <- filtered_pitchers %>%
  group_by(player_name, pitch_type, count) %>%
  summarise(pitch_count = n(), .groups = 'drop') %>%
  group_by(player_name, count) %>%
```

```r
  mutate(total_pitches_in_count = sum(pitch_count),
         pitch_proportion = pitch_count / total_pitches_in_count) %>%
  ungroup()
```

In addition, within each cluster, we looked at how pitchers used each pitch within each count. This was an important step because we were able to see any changes (or lack thereof) in different situations. The code above groups the data by pitcher, pitch, and count and calculates the number of pitches thrown for each combination of the three variables. Using the `pitch_proportion` variable, we are able to see the proportion of that pitch relative to all pitches thrown in that count.

```r
grouped_pitches_by_outs <- filtered_pitchers %>%
  group_by(player_name, pitch_type, outs_when_up) %>%
  summarise(pitch_count = n(), .groups = 'drop') %>%
  group_by(player_name, outs_when_up) %>%
  mutate(total_pitches_in_outs = sum(pitch_count),
         pitch_proportion = pitch_count / total_pitches_in_outs) %>%
  ungroup()

grouped_pitches_by_outs
```

```
# A tibble: 505 x 6
   player_name    pitch_type outs_when_up pitch_count total_pitches_in_outs
   <chr>          <chr>             <int>       <int>                 <int>
 1 Bassitt, Chris CH                    0           2                    32
 2 Bassitt, Chris CH                    1           7                    39
 3 Bassitt, Chris CH                    2           1                    35
 4 Bassitt, Chris CU                    0           3                    32
 5 Bassitt, Chris CU                    1           2                    39
 6 Bassitt, Chris CU                    2           8                    35
 7 Bassitt, Chris FC                    0           4                    32
 8 Bassitt, Chris FC                    1           6                    39
 9 Bassitt, Chris FC                    2           7                    35
10 Bassitt, Chris FF                    0           2                    32
# i 495 more rows
# i 1 more variable: pitch_proportion <dbl>
```

The code above is similar to the chunk working with different counts. Instead, we are grouping player name and pitch type with the number of outs and calculating the proportion of pitches thrown in each outs situation (0-2).

The code below makes a column for each pitch type and their proportion thrown with a given number of outs. The second chunk makes it so the columns now have wieghted pitch proportions for each pitch type, which will reduce the impact of small sample sizes.

```
pitchers_grouped <- grouped_pitches_by_outs %>%
  pivot_wider(names_from = pitch_type, values_from = pitch_proportion, values_fill = 0)
```

```
aggregated_pitchers_by_outs <- pitchers_grouped %>%
  group_by(player_name, outs_when_up) %>%
  summarise(across(starts_with("CH"):starts_with("KC"),
                   ~ sum(. * total_pitches_in_outs) / sum(total_pitches_in_outs),
                   .names = "weighted_{col}")) %>%
  ungroup()
```

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

```
aggregated_pitchers_by_outs
```

```
# A tibble: 114 x 12
   player_name    outs_when_up weighted_CH weighted_CU weighted_FC weighted_FF
   <chr>                 <int>       <dbl>       <dbl>       <dbl>       <dbl>
 1 Bassitt, Chris            0     0.0125      0.0188        0.025      0.0125
 2 Bassitt, Chris            1     0.0299      0.00855       0.0256     0
 3 Bassitt, Chris            2     0.00571     0.0457        0.04       0.00571
 4 Berríos, José             0     0.0667      0             0          0.0556
 5 Berríos, José             1     0.0517      0             0          0.0345
 6 Berríos, José             2     0.0583      0             0          0.025
 7 Bradish, Kyle             0     0.00526     0.0211        0          0.0316
 8 Bradish, Kyle             1     0.0174      0.0522        0          0.0261
 9 Bradish, Kyle             2     0.0370      0.0667        0          0.0667
10 Burnes, Corbin            0     0.00893     0.0357        0.179      0
# i 104 more rows
# i 6 more variables: weighted_FS <dbl>, weighted_SI <dbl>, weighted_SL <dbl>,
#   weighted_ST <dbl>, weighted_SV <dbl>, weighted_KC <dbl>
```

To visualize these results we looked at various different visuals including hierarchal clustering trees and heat maps. This proved to be difficult because with the amount of different pitchers the output was messy and difficult to see the results. One of the solutions to this problem was cutting the tree which cut down the number of clusters and made larger groups of similar pitchers. Viewing pitch usage by cluster provided a baseline on how we viewed each cluster and their pitch selection. However, it needed to be adjusted based on league averages for each different pitch. For example, four seam fastballs tend to be used more on average than others by starting pitchers, so we gave each different pitch a weight based on these averages in order to evenly compare them and provide more accurate clusterings. To achieve this we used ranks on each pitch to make comparisons.

```r
library(tidyverse)
ranked_pitchers_by_outs <- aggregated_pitchers_by_outs %>%
  group_by(player_name, outs_when_up) %>%
  mutate(
    rank_FF = rank(weighted_FF, ties.method = "average"),
    rank_CH = rank(weighted_CH, ties.method = "average"),
    rank_SL = rank(weighted_SL, ties.method = "average"),
    rank_SI = rank(weighted_SI, ties.method = "average"),
    rank_CU = rank(weighted_CU, ties.method = "average"),
    rank_FC = rank(weighted_FC, ties.method = "average"),
    rank_FS = rank(weighted_FS, ties.method = "average"),
    rank_ST = rank(weighted_ST, ties.method = "average"),
    rank_KC = rank(weighted_KC, ties.method = "average"),
    rank_SV = rank(weighted_SV, ties.method = "average")
  ) %>%
  ungroup()
```

In the dataset above, each pitcher has a rank for each pitch.

Based off of the ranked pitches we were able to make more accurate clusters with the pitch selections weighted for general pitch usage. We did this in two ways: one ignoring the number of outs and just recording pitch selection in general and another by looking at pitch selection based on the number of outs for each cluster.

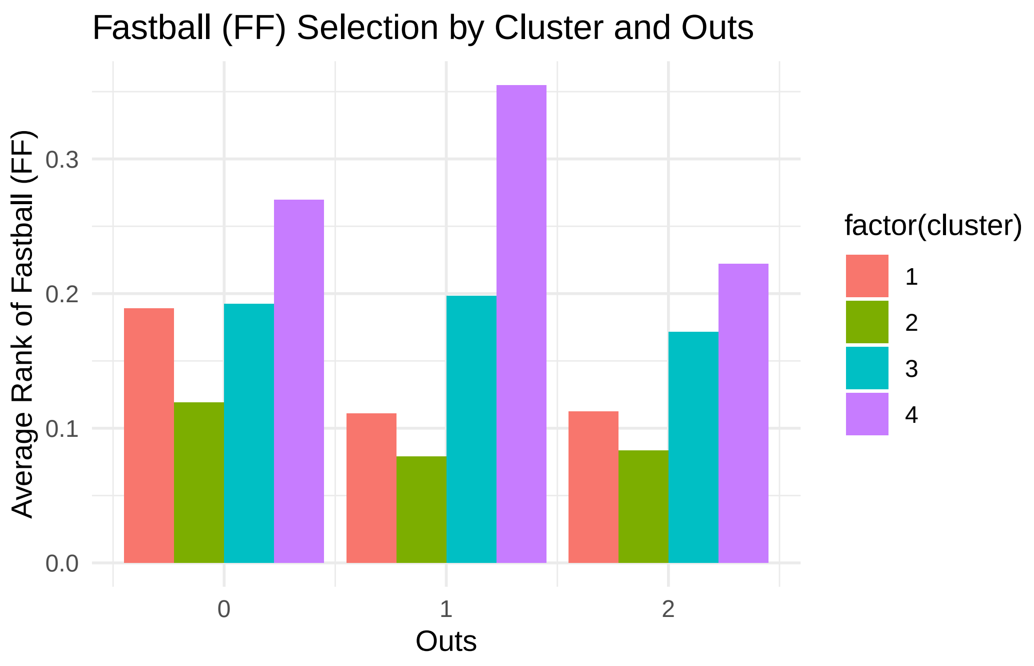
The chunk below sets up the pitching data for clustering based on pitch selection under given numbers of outs.

```r
pitching_data_for_clustering <- aggregated_pitchers_by_outs %>%
  select(-player_name, -outs_when_up)
distance_matrix <- dist(pitching_data_for_clustering, method = "euclidean")
hc_pitchers <- hclust(distance_matrix, method = "ward.D2")
```

```r
clusters <- cutree(hc_pitchers, k = 4)

aggregated_pitchers_by_outs <- aggregated_pitchers_by_outs %>%
  mutate(cluster = clusters)
```

```r
ggplot(aggregated_pitchers_by_outs, aes(x = outs_when_up, y = weighted_FF, fill = factor(clu
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Fastball (FF) Selection by Cluster and Outs",
       x = "Outs", y = "Average Rank of Fastball (FF)") +
  theme_minimal()
```



In the graph above you can see the pitch selection remained fairly consistent with each number of outs. Cluster 3 threw more fastballs when there was 0 outs compared to 1 or 2 outs. Also, cluster 5 threw significantly more fastballs with 1 out than when they had 0 or 2 outs.

This was a step in the right direction, however an issue came up when trying to visualize the data. Since each pitcher doesn't have every single pitch seen above in their arsenal, I needed to filter the data into three broader categories: fastballs, which included fourseams, sinkers, and cutters, offspeed pitches, which included changeups, knuckleballs, and splitters, and finally breaking balls, which included curveballs, sliders, knuckle curveballs, sweepers, and slurves. The code for this is shown below:

```r
type_pitches <- filtered_pitchers %>%
  filter(!(pitch_type %in% c("CS", "PO"))) %>%
  mutate(pitch_group = case_when(
    pitch_type %in% c("FF", "SI", "FC") ~ "Fastball",
    pitch_type %in% c("CH", "FS", "KN") ~ "Offspeed",
    pitch_type %in% c("CU", "SL", "ST", "KC", "SV") ~ "Breaking_Ball"
  ))
```

With these new pitch groupings we once again clustered pitchers based on their arsenals with the 3 broad pitch types. Similarly, we cut the hierarchal clustering tree into 5 groups and viewed the pitch selection in each one by calculating the average proportion of pitches in each of the three groups.

```
divided_pitches <- type_pitches %>%
  group_by(player_name, pitch_group) %>%
  summarize(count = n(), .groups = "drop") %>%
  group_by(player_name) %>%
  mutate(proportion = count / sum(count)) %>%
  select(-count) %>%
  pivot_wider(names_from = pitch_group, values_from = proportion, values_fill = 0)
```

```
cluster_pitches <- divided_pitches %>%
  select(Breaking_Ball, Fastball, Offspeed)
```

```
Adding missing grouping variables: `player_name`
```

```
dist_matrix <- dist(cluster_pitches, method = "euclidean")
```

```
Warning in dist(cluster_pitches, method = "euclidean"): NAs introduced by
coercion
```

```
hc <- hclust(dist_matrix, method = "ward.D2")
```

```
clusters <- cutree(hc, k = 5)
cluster_pitches.eight <- cluster_pitches %>%
  ungroup() %>%
  mutate(cluster = clusters)
```

```
pitch_selection <- cluster_pitches.eight %>%
  group_by(cluster) %>%
  summarize(
    avg_breaking_ball = mean(Breaking_Ball, na.rm = TRUE),
    avg_fastball = mean(Fastball, na.rm = TRUE),
    avg_offspeed = mean(Offspeed, na.rm = TRUE),
    count = n()
  )
```

The chunks below filter out “pitches” that were counted on caught stealings and pick-offs, we won’t be using those in the data because they weren’t actual pitches. Then it ranks each pitcher by pitch type usage with different numbers of outs.

```r
type_pitches_outs <- filtered_pitchers %>%
  filter(!(pitch_type %in% c("CS", "PO"))) %>%
  mutate(pitch_group = case_when(
    pitch_type %in% c("FF", "SI", "FC") ~ "Fastball",
    pitch_type %in% c("CH", "FS", "KN") ~ "Offspeed",
    pitch_type %in% c("CU", "SL", "ST", "KC", "SV") ~ "Breaking_Ball"
  ))
```

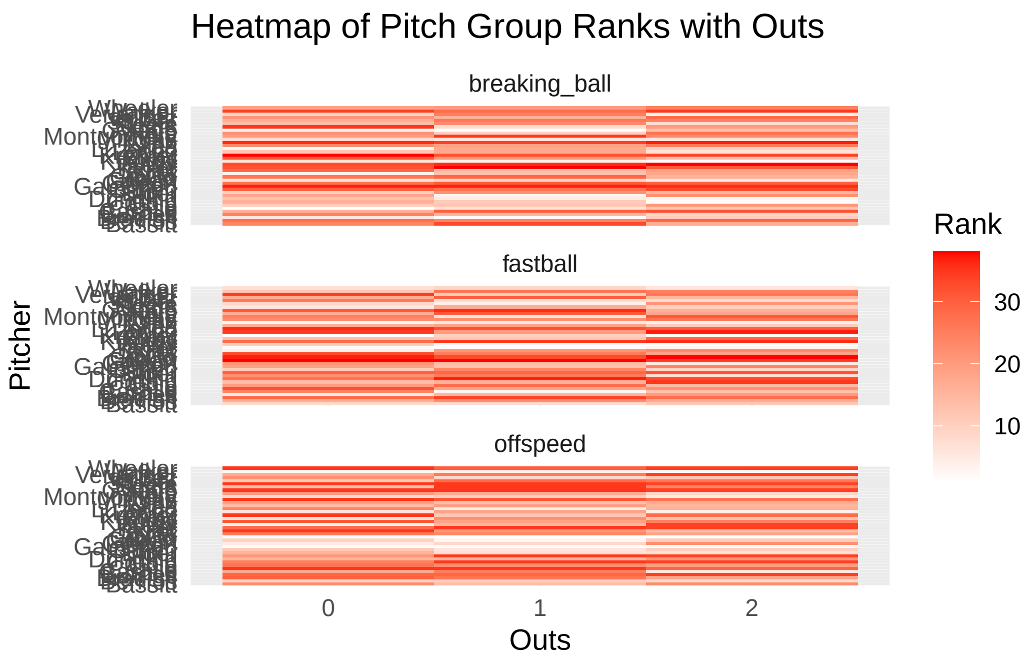
```r
type_pitches_outs_sum <- type_pitches_outs %>%
  group_by(player_name, outs_when_up, pitch_group) %>%
  summarize(count = n(), .groups = "drop") %>%
  group_by(player_name, outs_when_up) %>%
  mutate(proportion = count/sum(count)) %>%
  select(-count) %>%
  pivot_wider(names_from = pitch_group, values_from = proportion, values_fill = 0)
```

```r
ranks_outs <- type_pitches_outs_sum %>%
  mutate(last_name = str_extract(player_name, "^[^,]+")) %>%
  group_by(outs_when_up) %>%
  mutate(
    rank_breaking_ball = rank(-Breaking_Ball),
    rank_fastball = rank(-Fastball),
    rank_offspeed = rank(-Offspeed)
  ) %>%
  ungroup() %>%
  select(player_name, outs_when_up, rank_breaking_ball, rank_fastball, rank_offspeed, last_na
```

```r
ranks_outs_long <- ranks_outs %>%
  pivot_longer(cols = starts_with("rank"), names_to = "pitch_group", values_to = "rank") %>%
  mutate(pitch_group = gsub("rank_", "" ,pitch_group))
```

To visualize these results we looked at plots and heatmaps to see differences between the different clusters.

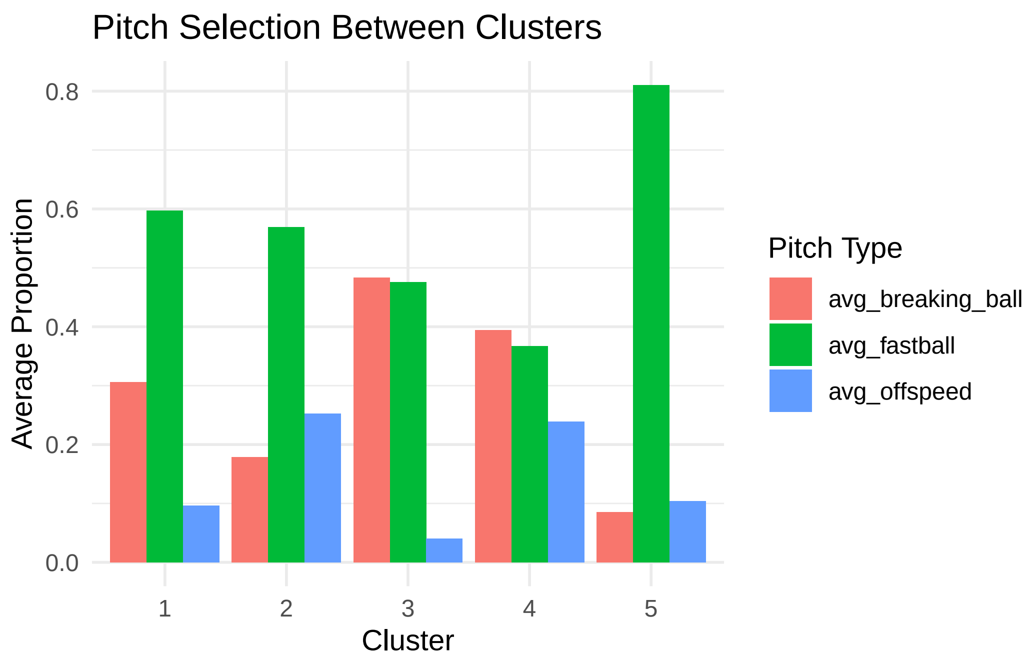
```r
ggplot(data = ranks_outs_long, mapping = aes(x = outs_when_up, y = last_name, fill = rank)) +
  geom_tile() +
  facet_wrap(~ pitch_group, ncol = 1) +
  scale_fill_gradient(low = "white", high = "red") +
  labs(
    x = "Outs",
    y = "Pitcher",
    fill = "Rank",
    title = "Heatmap of Pitch Group Ranks with Outs"
  ) +
  theme_minimal()
```



```
pitch_selection <- cluster_pitches.eight %>%
  group_by(cluster) %>%
  summarize(
    avg_breaking_ball = mean(Breaking_Ball, na.rm = TRUE),
    avg_fastball = mean(Fastball, na.rm = TRUE),
    avg_offspeed = mean(Offspeed, na.rm = TRUE),
    count = n()
  )
```

```
pitch_selection_plot <- pitch_selection %>%
  pivot_longer(cols = starts_with("avg"), names_to = "pitch_type", values_to = "proportion")
```

```r
ggplot(data = pitch_selection_plot, mapping = aes(x = factor(cluster), y = proportion, fill =
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Cluster", y = "Average Proportion", fill = "Pitch Type", title = "Pitch Selectio
  theme_minimal()
```



Step 2: Situational Changes

After clustering the pitchers we moved on to see how they reacted in different ball-strike counts. Specifically, we were interested in how their pitch selection varied in different ball-strike counts.

```r
specific_counts <- type_pitches %>%
  #filter(count %in% c("0-2", "1-2")) %>%
  group_by(player_name, pitch_group, count) %>%
  summarize(count_specific = n(), .groups = "drop") %>%
  group_by(player_name, count) %>%
  mutate(proportion_specific = count_specific - sum(count_specific)) %>%
  select(player_name, pitch_group, count, proportion_specific)
```

To find this, we looked at the average change in proportion when the pitcher was ahead in the count (0-2 and 1-2) for each cluster. The purpose of this was to see which pitch was their go-to in a strikeout count and if it differed at all from their usual arsenal.

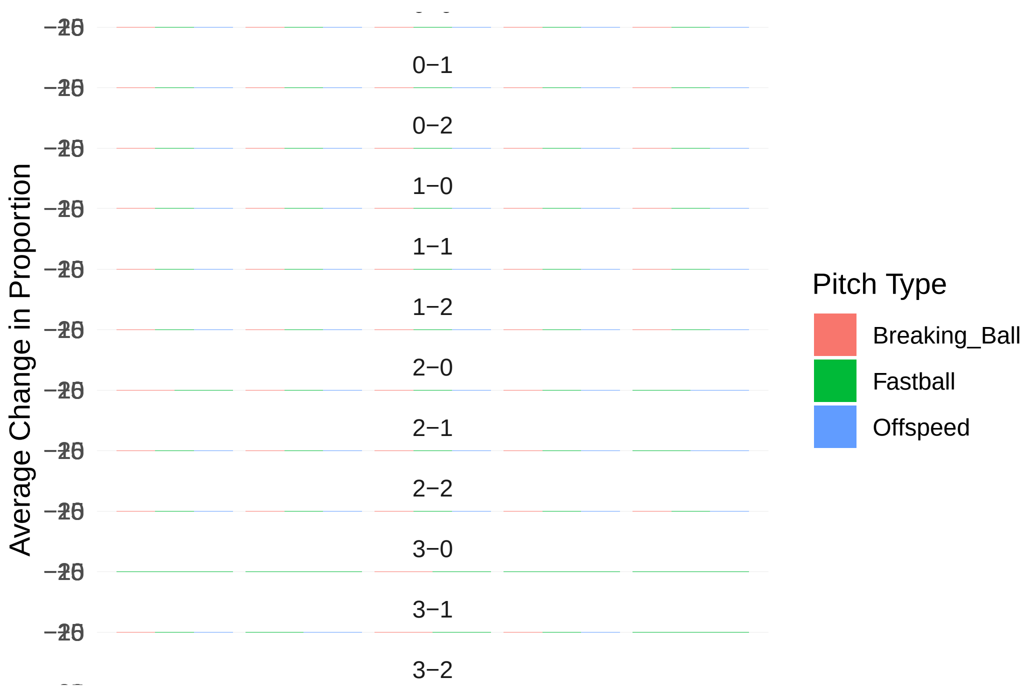
```r
specific_counts <- type_pitches %>%
  #filter(count %in% c("0-2", "1-2")) %>%
  group_by(player_name, pitch_group, count) %>%
  summarize(count_specific = n(), .groups = "drop") %>%
  group_by(player_name, count) %>%
  mutate(proportion_specific = count_specific - sum(count_specific)) %>%
  select(player_name, pitch_group, count, proportion_specific)
```

```r
general_selection <- divided_pitches %>%
  pivot_longer(cols = c(Fastball, Breaking_Ball, Offspeed),
               names_to = "pitch_group", values_to = "proportion_general")
```

```r
comparison <- specific_counts %>%
  left_join(general_selection, by = c("player_name", "pitch_group")) %>%
  mutate(proportion_difference = proportion_specific - proportion_general)
```

```r
comparison_cluster <- comparison %>%
  left_join(cluster_pitches.eight %>% select(player_name, cluster), by = "player_name") %>%
  group_by(cluster, count, pitch_group) %>%
  summarize(avg_proportion_difference = mean(proportion_difference, na.rm = TRUE), .groups =
```

```r
ggplot(data = comparison_cluster, aes(x = factor(cluster), y = avg_proportion_difference, fil
  geom_bar(stat = "identity", position = "dodge") +
  facet_wrap(~ count, ncol = 1) +
  labs(x = "Cluster", y = "Average Change in Proportion", fill = "Pitch Type",
       title = "Change in Pitch Selection in 0-2 and 1-2 Counts by Cluster") +
  theme_minimal()
```



After visualizing the proportions, we chose to standardize them using z-scores, which allowed us to make more effective comparisons. We then compared the standardized changes in pitch usages both within and between the different clusters.

The code below reformats the data and computes the average change in pitch usage by pitch type and cluster.

```r
count_prop <- type_pitches %>%
#  filter(count %in% c("0-2", "1-2")) %>%
  group_by(player_name, count) %>%
  mutate(total_pitches = n()) %>%
  group_by(player_name, pitch_group, count) %>%
  summarise(proportion = n() / unique(total_pitches), .groups = 'drop')
```

```r
# adding cluster column
comparison <- comparison %>%
  left_join(cluster_pitches.eight %>% select(player_name, cluster), by = "player_name")
```

```r
count_diff <- count_prop %>%
  inner_join(comparison, by = c("player_name", "count", "pitch_group")) %>%
  mutate(difference = proportion_general - proportion)
```

```
count_diff <- count_diff %>%
  select(-proportion_specific, -proportion_difference)
```

```
avg_diff <- count_diff %>%
  group_by(player_name, count, cluster) %>%
  summarise(
    change_fastball = mean(difference[pitch_group == "Fastball"], na.rm = TRUE) %>% replace_
    change_breaking_ball = mean(difference[pitch_group == "Breaking_Ball"], na.rm = TRUE) %>
    change_offspeed = mean(difference[pitch_group == "Offspeed"], na.rm = TRUE) %>% replace_
  ) %>%
  ungroup()
```

```
`summarise()` has grouped output by 'player_name', 'count'. You can override
using the `.groups` argument.
```

Following this and reformatting the data, we made a small shiny app to visualize the results. The shiny app allows you to visualize the different pitch selection for each cluster under different counts. This includes seeing how the pitchers changed their fastball, breaking ball, and offspeed pitch selection. In addition, it includes a 3D plot to visualize the different changes. Since there are three different pitch groupings, a 3D visual was neccesary to view the data and took us one step closer to look into modeling.

```
library(plotly)
library(shiny)
ui <- fluidPage(
  titlePanel("3D Pitch Type Change by Pitcher"),
  sidebarLayout(
    sidebarPanel(
      selectInput("count", "Select Count", choices = unique(avg_diff$count)),
      selectInput("cluster", "Select Cluster", choices = unique(avg_diff$cluster))
    ),
    mainPanel(
      plotlyOutput("pitch3DPlot")
    )
  )
)

# Define server logic
server <- function(input, output) {
  filtered_data <- reactive({
    avg_diff %>%
      filter(count == input$count, cluster == input$cluster)
  })

  output$pitch3DPlot <- renderPlotly({
    plot_ly(
      data = filtered_data(),
      x = ~change_fastball,
      y = ~change_breaking_ball,
      z = ~change_offspeed,
      type = "scatter3d",
      mode = "markers",
      text = ~player_name,
      marker = list(size = 5)
    ) %>%
      layout(
        scene = list(
          xaxis = list(title = "Change in Fastball"),
          yaxis = list(title = "Change in Breaking Ball"),
          zaxis = list(title = "Change in Offspeed")
        ),
        title = "3D Plot of Pitch Type Changes by Pitcher"
      )
  })
}

shinyApp(ui = ui, server = server)
```

Step 3: Modeling

The next step in the process was to test out some models to see if we could predict the amount a pitcher varies from their average selection in different situations. Also, we included statistical measures as explanatory variables to see if their is a difference in variation between more and less successful pitchers. To calculate the variation we used the sum of the changes in the squared pitch groups (fastballs, breaking balls, and offspeed) and the absolute sum of the changes by taking the absolute value of each change for each type of pitch. The code for which can be found below:

```
pitching23 <- read_csv("pitching23.csv")
```

```
New names:
```

```
Rows: 44 Columns: 39
-- Column specification
-------------------------------------------------------- Delimiter: "," chr
(4): Player, Team, Lg, Player-additional dbl (35): Rk, ERA...3, Season, Age, W,
L, W-L%, Dec, ERA...12, G, GS, CG, SH...
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.
* `ERA` -> `ERA...3`
* `ERA` -> `ERA...12`
```

```r
era_pitchers <- pitching23 %>%
  separate(Player, into = c("Firstname", "Lastname"), sep = " ", extra = "merge") %>%
  mutate(Player = paste(Lastname, Firstname, sep = ", "),
         ERA = ERA...3) %>%
  select(Player, ERA)
```

```r
# Perform the join
avg_diff <- avg_diff %>%
  left_join(era_pitchers, by = c("player_name" = "Player"))
```

```r
# Calculate SSE and absolute sum of changes for each pitcher
error_df <- avg_diff %>%
  group_by(player_name, count) %>%
  summarize(
    SSE = sum((change_fastball)^2 + (change_breaking_ball)^2 + (change_offspeed)^2),
    rSSE = sqrt(SSE),
    abs_sum = sum(abs(change_fastball) + abs(change_breaking_ball) + abs(change_offspeed)),
    ERA = first(ERA)  # Keep ERA the same for all counts of a player
  ) %>%
  ungroup()
```

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

For the actual modeling we used statistics like ERA, different counts, strikeouts, WHIP, FIP, strikeout-to-walk ratio, innings pitched, and batters faced to predict SSE and the absolute sum of changes. We attempted various model types including linear and log models to find the best fit or at least look for significant explanatory variables.

```
so_pitchers <- pitching23 %>%
  separate(Player, into = c("Firstname", "Lastname"), sep = " ", extra = "merge") %>%
  mutate(Player = paste(Lastname, Firstname, sep = ", "),
         ERA = ERA...3) %>%
  select(Player, SO)
```

```
# Perform the join
avg_diff_so <- avg_diff %>%
  left_join(so_pitchers, by = c("player_name" = "Player"))
```

```
# Calculate SSE and absolute sum of changes for each pitcher
error_df_so <- avg_diff_so %>%
  group_by(player_name, count) %>%
  summarize(
    SSE = sqrt(sum((change_fastball)^2 + (change_breaking_ball)^2 + (change_offspeed)^2)),
    abs_sum = sum(abs(change_fastball) + abs(change_breaking_ball) + abs(change_offspeed)),
    SO = first(SO)  # Keep ERA the same for all counts of a player
  ) %>%
  ungroup()
```

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

```
error_df_so_adj <- error_df_so %>%
  mutate(
    # Split 'count' into balls and strikes
    balls = as.numeric(sub("-.*", "", count)),
    strikes = as.numeric(sub(".*-", "", count)),
    # Calculate adjusted_count
    adjusted_count = strikes - balls
  )
```

```
pitching23_cut <- pitching23 %>%
  mutate(Player = sub("^(\\S+)\\s+(.*)$", "\\2, \\1", Player)) %>%
  select(Player, FIP, WHIP, `SO/BB`, BF, IP) # Select only the desired columns
```

```
stat_df <- error_df_so_adj %>%
  left_join(pitching23_cut, by = c("player_name" = "Player"))
```

```r
mod_int3 <- lm(log(SSE) ~ as.factor(adjusted_count) + FIP + `SO/BB` + IP, data = stat_df)
summary(mod_int3)
```

```

Call:
lm(formula = log(SSE) ~ as.factor(adjusted_count) + FIP + `SO/BB` +
    IP, data = stat_df)

Residuals:
    Min      1Q  Median      3Q     Max
-2.9505 -0.3716  0.1443  0.4557  1.6057

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)
(Intercept)                 -0.248894   0.672291  -0.370 0.711408
as.factor(adjusted_count)-2 -0.163465   0.177677  -0.920 0.358096
as.factor(adjusted_count)-1 -0.602318   0.167886  -3.588 0.000373 ***
as.factor(adjusted_count)0  -0.934577   0.167771  -5.571 4.55e-08 ***
as.factor(adjusted_count)1  -0.965611   0.174418  -5.536 5.46e-08 ***
as.factor(adjusted_count)2  -0.443546   0.192990  -2.298 0.022037 *
FIP                         -0.067337   0.058864  -1.144 0.253301
`SO/BB`                      0.037960   0.030507   1.244 0.214081
IP                          -0.002998   0.003085  -0.972 0.331696
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7201 on 419 degrees of freedom
Multiple R-squared:  0.1765,    Adjusted R-squared:  0.1608
F-statistic: 11.23 on 8 and 419 DF,  p-value: 1.988e-14
```

In this original model we see the adjusted count of -1, 0, 1, and 2 all being significant, suggesting that in those situations where the pitcher is either even, down 1 ball, up 1 strike, or up 2 strikes, pitchers are deviating from their usual arsenal. However, statistical performance measures such as FIP, strikeout-to-walk ratio, and innings pitched don't seem to have a significant effect.

Conclusion