Atlanta Braves Pitching Analysis

Rob Kravec

12/23/2020

Introduction

Read in Statcast data for Braves pitchers from 2017-2020 (including playoffs)

```
# Create function to read in data. Several columns had data types changed in
# 2020, so we manually assign the data type of our choosing for those columns
read_braves <- function(year) {
  file_name <- paste0("data/Braves_", year, ".csv")
  pitch_data <- vroom(file_name,
                      col_types = c(release_spin_rate = "d",
                                    release_speed = "d",
                                    release_pos_x = "d",
                                    release_pos_y = "d",
                                    release_pos_z = "d",
                                    zone = "d",
                                    pfx_x = "d",
                                    pfx_z = "d",
                                    plate_x = "d",
                                    plate_z = "d",
                                    fielder_2 = "c", fielder_3 = "c",
                                    fielder_4 = "c", fielder_5 = "c",
                                    fielder_6 = "c", fielder_7 = "c",
                                    fielder_8 = "c", fielder_9 = "c",
                                    vx0 = "d", vy0 = "d", vz0 = "d",
                                    ax = "d", ay = "d", az = "d",
                                    sz_top = "d", sz_bot = "d",
                                    effective_speed = "d",
                                    release_extension = "d"
                                    )
                      )
  return(pitch_data)
}

# Read in 2017-2020 data using function
braves_pitch <- map_df(.x = 2017:2020, .f = read_braves)

# Look at summary of output
glimpse(braves_pitch)
```

```
## Rows: 84,343
## Columns: 89
## $ pitch_type                    <chr> "SL", "FT", "FF", "FF", "FT", "FF",...
## $ game_date                     <date> 2017-10-01, 2017-10-01, 2017-10-01...
```

```
## $ release_speed                <dbl> 85.2, 97.8, 97.8, 97.8, 97.9, 97.9,...
## $ release_pos_x                <dbl> -1.10, -1.52, -1.59, -1.61, -1.57, ...
## $ release_pos_z                <dbl> 5.79, 5.69, 5.77, 5.72, 5.75, 5.74,...
## $ player_name                  <chr> "Arodys Vizcaino", "Arodys Vizcaino...
## $ batter                       <dbl> 592885, 592885, 592885, 592885, 592...
## $ pitcher...8                  <dbl> 527055, 527055, 527055, 527055, 527...
## $ events                       <chr> "field_out", "null", "null", "null"...
## $ description                  <chr> "hit_into_play", "swinging_strike",...
## $ spin_dir                     <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ spin_rate_deprecated         <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ break_angle_deprecated       <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ break_length_deprecated      <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ zone                         <dbl> 5, 7, 11, 11, 13, 4, 13, 1, 11, 2, ...
## $ des                          <chr> "Christian Yelich flies out to cent...
## $ game_type                    <chr> "R", "R", "R", "R", "R", "R", "R", ...
## $ stand                        <chr> "L", "L", "L", "L", "L", "R", "R", ...
## $ p_throws                     <chr> "R", "R", "R", "R", "R", "R", "R", ...
## $ home_team                    <chr> "MIA", "MIA", "MIA", "MIA", "MIA", ...
## $ away_team                    <chr> "ATL", "ATL", "ATL", "ATL", "ATL", ...
## $ type                         <chr> "X", "S", "B", "B", "S", "X", "B", ...
## $ hit_location                 <chr> "8", "null", "null", "null", "null"...
## $ bb_type                      <chr> "fly_ball", "null", "null", "null",...
## $ balls                        <dbl> 2, 2, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,...
## $ strikes                      <dbl> 2, 1, 1, 1, 0, 0, 0, 2, 2, 1, 0, 2,...
## $ game_year                    <dbl> 2017, 2017, 2017, 2017, 2017, 2017,...
## $ pfx_x                        <dbl> 0.13, -1.21, -1.11, -0.99, -1.28, -...
## $ pfx_z                        <dbl> -0.06, 1.33, 1.58, 1.56, 1.36, 1.76...
## $ plate_x                      <dbl> 0.13, -0.79, -1.68, -1.75, -0.94, -...
## $ plate_z                      <dbl> 2.60, 1.98, 2.66, 3.41, 2.04, 2.45,...
## $ on_3b                        <chr> "null", "null", "null", "null", "nu...
## $ on_2b                        <chr> "null", "null", "null", "null", "nu...
## $ on_1b                        <chr> "null", "null", "null", "null", "nu...
## $ outs_when_up                 <dbl> 2, 2, 2, 2, 2, 1, 1, 0, 0, 0, 0, 2,...
## $ inning                       <dbl> 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 8,...
## $ inning_topbot                <chr> "Bot", "Bot", "Bot", "Bot", "Bot", ...
## $ hc_x                         <chr> "113.27", "null", "null", "null", "...
## $ hc_y                         <chr> "66.83", "null", "null", "null", "n...
## $ tfs_deprecated               <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ tfs_zulu_deprecated          <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ fielder_2...42               <chr> "435559", "435559", "435559", "4355...
## $ umpire                       <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ sv_id                        <chr> "171001_223124", "171001_223108", "...
## $ vx0                          <dbl> 2.5906327, 4.6076989, 2.1938308, 1....
## $ vy0                          <dbl> -123.6350, -141.7441, -141.7687, -1...
## $ vz0                          <dbl> -0.9897656, -7.3454665, -6.2723839,...
## $ ax                           <dbl> 0.7607312, -17.0473078, -15.0509318...
## $ ay                           <dbl> 26.70774, 34.09779, 35.06824, 34.33...
## $ az                           <dbl> -32.849503, -13.132368, -10.151391,...
## $ sz_top                       <dbl> 3.44, 3.44, 3.35, 3.40, 3.44, 3.50,...
## $ sz_bot                       <dbl> 1.59, 1.59, 1.68, 1.73, 1.59, 1.47,...
## $ hit_distance_sc              <chr> "378", "null", "null", "null", "203...
## $ launch_speed                 <chr> "97.4", "null", "null", "null", "nu...
## $ launch_angle                 <chr> "32", "null", "null", "null", "null...
## $ effective_speed              <dbl> 83.6, 96.2, 96.2, 96.1, 96.2, 95.5,...
```

```
## $ release_spin_rate               <dbl> 2343, 2348, 2291, 2212, 2291, 2393,...
## $ release_extension               <dbl> 5.6, 5.7, 5.7, 5.6, 5.7, 5.5, 5.8, ...
## $ game_pk                         <dbl> 492515, 492515, 492515, 492515, 492...
## $ pitcher...60                    <dbl> 527055, 527055, 527055, 527055, 527...
## $ fielder_2...61                  <chr> "435559", "435559", "435559", "4355...
## $ fielder_3                       <chr> "518692", "518692", "518692", "5186...
## $ fielder_4                       <chr> "645277", "645277", "645277", "6452...
## $ fielder_5                       <chr> "547004", "547004", "547004", "5470...
## $ fielder_6                       <chr> "621020", "621020", "621020", "6210...
## $ fielder_7                       <chr> "572669", "572669", "572669", "5726...
## $ fielder_8                       <chr> "542255", "542255", "542255", "5422...
## $ fielder_9                       <chr> "455976", "455976", "455976", "4559...
## $ release_pos_y                   <dbl> 54.90, 54.77, 54.77, 54.87, 54.80, ...
## $ estimated_ba_using_speedangle   <chr> "0.359", "null", "null", "null", "n...
## $ estimated_woba_using_speedangle <chr> "0.660", "null", "null", "null", "n...
## $ woba_value                      <chr> "0.00", "null", "null", "null", "nu...
## $ woba_denom                      <chr> "1", "null", "null", "null", "null"...
## $ babip_value                     <chr> "0", "null", "null", "null", "null"...
## $ iso_value                       <chr> "0", "null", "null", "null", "null"...
## $ launch_speed_angle              <chr> "5", "null", "null", "null", "null"...
## $ at_bat_number                   <dbl> 82, 82, 82, 82, 82, 81, 81, 80, 80,...
## $ pitch_number                    <dbl> 5, 4, 3, 2, 1, 2, 1, 4, 3, 2, 1, 5,...
## $ pitch_name                      <chr> "Slider", "2-Seam Fastball", "4-Sea...
## $ home_score                      <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,...
## $ away_score                      <dbl> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,...
## $ bat_score                       <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,...
## $ fld_score                       <dbl> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,...
## $ post_away_score                 <dbl> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,...
## $ post_home_score                 <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,...
## $ post_bat_score                  <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,...
## $ post_fld_score                  <dbl> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,...
## $ if_fielding_alignment           <chr> "Standard", "Standard", "Standard",...
## $ of_fielding_alignment           <chr> "Standard", "Standard", "Standard",...
```

```
# Based on this summary and the glossary, remove deprecated field
braves_pitch <- braves_pitch %>%
  select(-c(spin_dir, spin_rate_deprecated, break_angle_deprecated,
            break_length_deprecated, tfs_deprecated, tfs_zulu_deprecated,
            umpire))

# Remove duplicate columns for pitcher and catcher ids
braves_pitch <- braves_pitch %>%
  select(-c(pitcher...8, fielder_2...42)) %>%
  rename(pitcher = pitcher...60, fielder_2 = fielder_2...61)
```

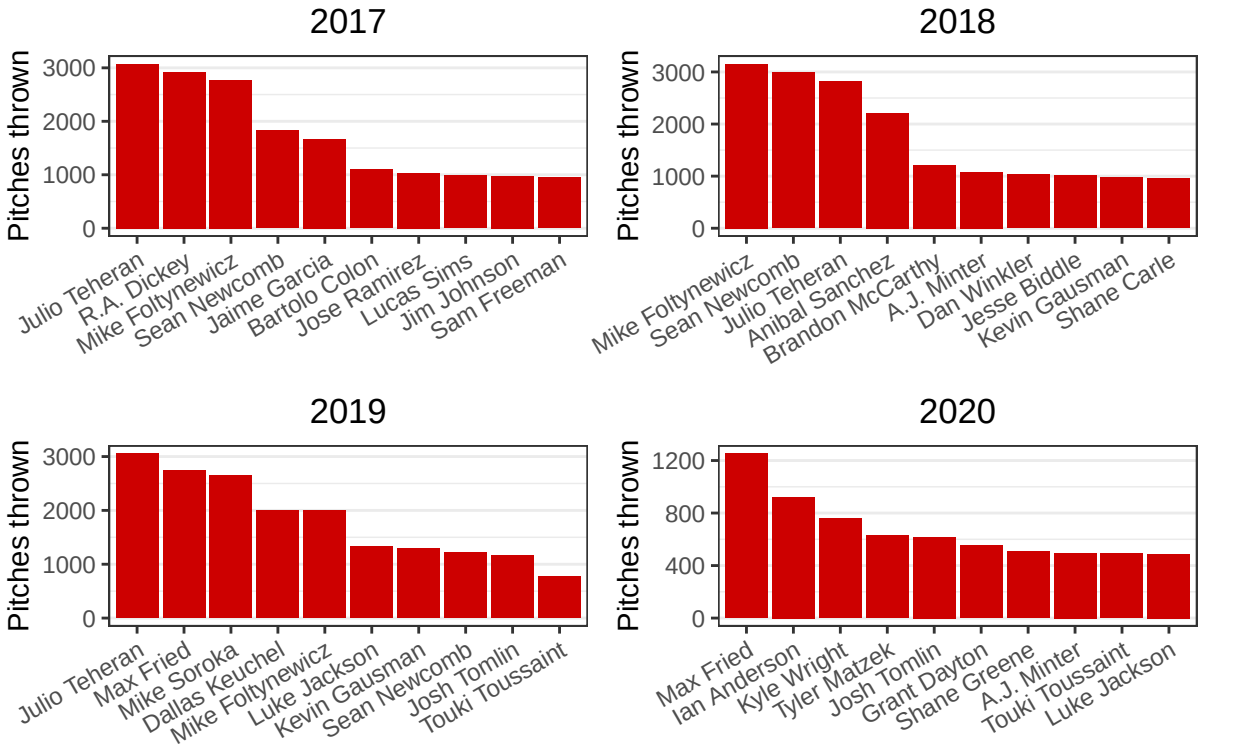
The purpose of this project is to ask and answer some interesting questions about Atlanta Braves pitching from 2017-2020. We don't have a set agenda per se. Rather, the project is a success if we uncover some previously unknown insights and find compelling ways to present them.

Why the Atlanta Braves? I wanted to keep the amount of data manageable, so I elected to focus on my favorite team *Why 2017-2020?* There is no special significance to this time period. I wanted to keep the amount of data manageable while still being able to look at multiple (recent) seasons. 2017-2020 provided a reasonable solution, while also ensuring that all velocities were measured with a single system (Statcast). From 2008 to 2016, pitch velocities were measured via PitchF/x

EDA

Who threw the most pitches for the Braves in each year?

```
# Create data frame that contains top 10 pitch throwers per season
top_pitches <- braves_pitch %>%
  group_by(game_year) %>%
  count(player_name) %>%
  arrange(desc(n)) %>%
  dplyr::slice(1:10) %>%
  rename(pitches = n)

# Write function to generate desired plot for the given year
plot_pitches <- function(year){
  top_pitches %>%
    filter(game_year == year) %>%
    ggplot(mapping = aes(x = reorder(player_name, - pitches), y = pitches)) +
    geom_bar(stat = "identity", fill = "red3") +
    labs(y = "Pitches thrown",
         title = paste0(year)) +
    theme_bw() +
    theme(plot.title = element_text(hjust = 0.5),
          panel.grid.major.x = element_blank(),
          axis.text.x = element_text(angle = 30, hjust = 1),
          axis.title.x = element_blank())
}

# Create individual plots required to make patchwork plot of pitch leaders
# for 2017-2020
for (i in 2017:2020) {
  plot_name <- paste0("pitch_", i)
  assign(plot_name, plot_pitches(i))
}

# Build patchwork plot
patch_plot <- (pitch_2017 + pitch_2018) / (pitch_2019 + pitch_2020)
patch_plot <- patch_plot + plot_annotation(
  title = "Atlanta Braves leaders in pitches thrown per year",
  theme = theme(plot.title = element_text(hjust = 0.5, size = 20)))
patch_plot
```




```
# Save plot
ggsave(plot = patch_plot, path = "plots/", filename = "total_pitches.png")
```

Normally, in summarizing a pitcher's work for a given season, we tend to think about inning totals, not pitch totals. We opted for pitch totals here due to the unique nature of the Statcast data, and I think the results are pretty interesting!

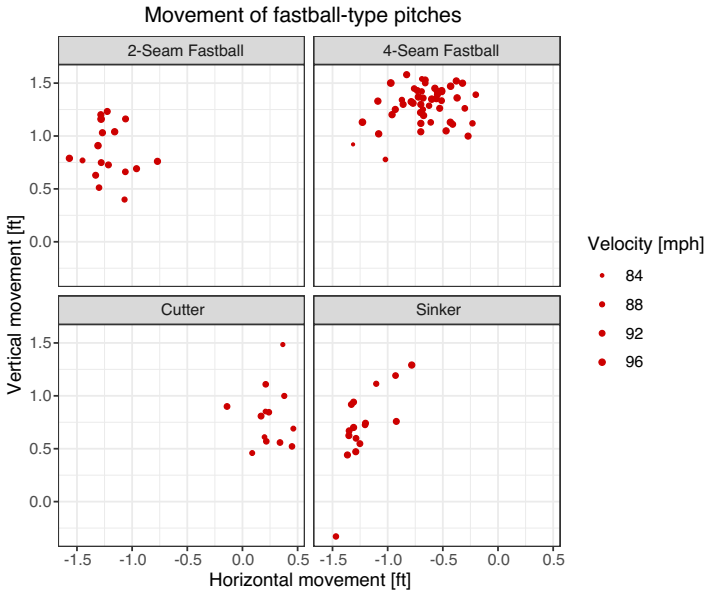
- The turnover on the Braves' pitching staff year-to-year is quite significant! Of course, we would need to contextualize (relative to other teams) before making any meaningful statements about turnover being more or less than average
- We see clear evidence of the Braves' proclivity for bringing in veteran pitchers on short contracts: R.A. Dickey, Jaime Garcia, and Bartolo Colon in 2017, Anibal Sanchez in 2018, Dallas Keuchel in 2019. Cole Hamels is noticeably absent from the 2020 plot after a shoulder injury (essentially wasting his $18M one year contract)
- 2020 was a weird season! Likely due to their prominence in the playoffs, Ian Anderson and Kyle Wright make the top 3 pitch throwers for the Braves. After those two and Max Fried, we see a slew of relievers. Chris Martin and Mark Melancon (two of the highest leverage arms in the bullpen) threw about 400 pitches each and just missed appearing in the 2020 plot

From here, we'll try to leverage the Statcast data for its strengths: individual pitch-level insights. While it was fun to look at pitch totals across seasons, summary statistics (which we could easily find elsewhere) would probably provide more intuitive and meaningful insights.

Among the most common pitch types, which pitchers have the most pitch movement?

```
# Prepare dataset
fastballs <- braves_pitch %>%
```

```r
  filter(pitch_name %in% c("2-Seam Fastball", "4-Seam Fastball",
                           "Cutter", "Sinker")) %>%
  group_by(player_name, pitch_name, p_throws) %>%
  summarise(cnt = n(),
            x_move = median(pfx_x), # Reduce influence of extreme cases with median
            z_move = median(pfx_z),
            velo = median(release_speed)) %>%
  filter(cnt >= 50) %>% # Remove pitcher / pitch combos with fewer than 50 occurrences
  ungroup() %>%
  mutate(x_move_mod = ifelse(p_throws == "L", -x_move, x_move),
         x_move_abs = abs(x_move), # Corrects for difference in L vs. R hand pitchers
         total_move = sqrt(x_move ^ 2 + z_move ^ 2) # Define Pythagorean tot movement
         ) %>%
  group_by(pitch_name) %>%
  arrange(pitch_name, desc(total_move)) #%>%
  #dplyr::slice(1:10)
# Ultimately decided not to filter by total movement for fastballs because vertical
# movement is tricky for fastballs. While it is generally accepted as a positive
# attribute for 4-seam fastballs, the same cannot be said for 2-seam fastballs,
# cutters, and sinkers. In fact, lower values of vertical movement for
# 2-seam fastballs and sinkers is likely desirable (e.g., a rising sinker is bad)

# Generate interactive scatterplot using ggirafe
fastball_plot <- ggplot(data = fastballs,
       mapping = aes(x = x_move_mod, y = z_move)) +
  geom_jitter_interactive(mapping = aes(size = velo,
                                        tooltip = paste0("Player: ", player_name,
                                                         "\n Avg velo: ",
                                                         round(velo, 1), " MPH"))
                          , color = "red3") +
  labs(x = "Horizontal movement [ft]",
       y = "Vertical movement [ft]",
       title = "Movement of fastball-type pitches",
       size = "Velocity [mph]") +
  facet_wrap(~pitch_name) +
  scale_size_continuous(range = c(0.5, 1.5)) +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

# Display plot and save as .html file
fastball_plot_interactive <- girafe({code = print(fastball_plot)},
                                    options = list(
                                      opts_tooltip(use_fill = T),
                                      opts_hover(css = "color:blue;"),
                                      opts_sizing(rescale = TRUE, width = .7)
                                      ))
fastball_plot_interactive
```



Fastballs

```
withr::with_dir('plots', saveWidget(fastball_plot_interactive,
                                    file="fastball_movement.html"))
# Last command is necessary to save plot into "plots/" directory. saveWidget
# has an issue with relative paths
```

This exercise gave some good practice with interactive plotting and showed the importance of using intuition during the EDA process. At first, it seemed like a good idea to use a Pythagorean definition of total movement (i.e., sqrt of sum of squared x and z movement components) to select pitchers with better than average movement on their pitches. However, upon reviewing the resulting filtered scatterplots, it became apparent that something was wrong (due to the absence of expected names): Some pitches (namely 2-seam fastballs and sinkers) are actually better if they have less positive z-directional movement.

To combat this issue, I decided to include all pitchers who threw at least 50 of that pitch type and use interactive points (through ggirafe) to give users helpful information. This interactivity helps to relieve the crowding issue of having many overlapping points in a small space.

A few baseball-related takeaways:

- Peter Moylan's sinker stands out as the most unique pitch on the plot, likely due to his atypical sidearm / submarine delivery. Josh Collmenter's cutter also stands out (to a lesser degree) for positive vertical movement, likely due to his exaggerated overhead delivery
- A few pitchers that helped me to spot the issue with our Pythagorean movement definition for 2-seam fastballs and sinkers include Jonny Venters, Dallas Keuchel, Julio Teheran, and Mike Soroka. These pitchers are / were known to have above average pitches in these categories, and it surprised me to see their names omitted from an originally filtered plot
- Cutters appear to have the least horizontal movement of all of the fastball-type pitches, which is doubly interesting because it is the only pitch with glove-side movement. It's unclear to me whether the small magnitude of horizontal movement is due to (1) the manner in which Statcast data is measured (i.e., some kind of offset or bias toward arm-side movement), (2) the natural tendency of fastballs to move toward the arm-side, (3) some combination of the above or something else entirely!

** Note: This plot provides some insight into the relationship between fastball velocity and movement (through the size aesthetic), but I'm not seeing a clear trend pop off of the page. A future investigation into this relationship (with some different plots and calculations) could be interesting, but I'll move on now to cover additional topics.

Off-speed pitches To determine which off-speed pitches to focus on, let's see how many pitchers threw at least 50 of each pitch.

```
braves_pitch %>%
  count(player_name, pitch_name) %>%
  filter(n >= 50) %>%
  count(pitch_name) %>%
  arrange(desc(n))
```

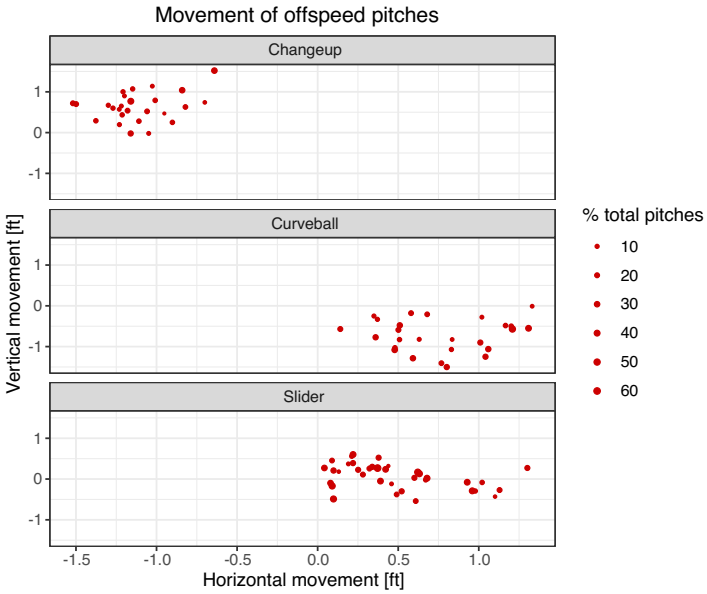
```
## # A tibble: 11 x 2
##    pitch_name          n
##    <chr>           <int>
##  1 4-Seam Fastball    49
##  2 Slider             36
##  3 Changeup           26
##  4 Curveball          26
##  5 2-Seam Fastball    17
##  6 Sinker             16
##  7 Cutter             13
##  8 Split-Finger        4
##  9 null                3
## 10 Knuckle Curve       1
## 11 Knuckleball         1
```

From this result, it is pretty safe to say that we can focus on sliders, changeups, and curveballs in this section.

To differentiate slightly from the previous section on fastball-type pitch movement, we'll find the percentage of pitches for each pitcher that are sliders, changeups, and curveballs. This information can then be conveyed through the size aesthetic (instead of velocity).

```
# Compute total number of pitches thrown by each Braves pitcher
total_pitches <- braves_pitch %>%
  count(player_name) %>%
  arrange(desc(n))

# Create data frame corresponding to off-speed pitches of interest
offspeed <- braves_pitch %>%
  group_by(player_name, pitch_name, p_throws) %>%
  summarise(cnt = n(),
            x_move = median(pfx_x), # Reduce influence of extreme cases with median
            z_move = median(pfx_z),
            velo = median(release_speed)
            ) %>%
  filter(cnt >= 50, pitch_name %in% c("Slider", "Changeup", "Curveball")) %>%
  ungroup() %>%
  # Correct for difference in L vs. R hand pitchers
  mutate(x_move_mod = ifelse(p_throws == "L", -x_move, x_move))

# Join total_pitches data frame to offspeed
offspeed_join <- left_join(x = offspeed, y = total_pitches,
                           by = c("player_name"))

# Create additional column corresponding to % of total pitches thrown
offspeed_join <- offspeed_join %>%
  mutate(perc = 100 * round(cnt / n, 3))
```

```
# Generate interactive plot
offspeed_plot <- ggplot(data = offspeed_join,
       mapping = aes(x = x_move_mod, y = z_move)) +
  geom_jitter_interactive(mapping = aes(size = perc,
                                        tooltip = paste0("Player: ", player_name,
                                                         "\n Avg velo: ",
                                                         round(velo, 1), " MPH",
                                                         "\n % of total pitches: ",
                                                         perc, "%")),
                          color = "red3") +
  labs(x = "Horizontal movement [ft]",
       y = "Vertical movement [ft]",
       title = "Movement of offspeed pitches",
       size = "% total pitches") +
  facet_wrap(~pitch_name, dir = "v") +
  scale_size_continuous(range = c(0.5, 1.5)) +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

# Display plot and save as .html file
offspeed_plot_interactive <- girafe({code = print(offspeed_plot)},
                                    options = list(
                                      opts_tooltip(use_fill = T),
                                      opts_hover(css = "color:blue;"),
                                      opts_sizing(rescale = TRUE, width = .7)
                                      ))
offspeed_plot_interactive
```




```
withr::with_dir('plots', saveWidget(offspeed_plot_interactive,
                                    file="offspeed_movement.html"))
```

A few things that stand out to me (more for fun than meaningful insight):

- Josh Collmenter again stands out for exceptional rise, this time on his changeup, which he throws about 30% of the time

- I now understand (at least part of) why hitters look so silly against Chris Martin's and Shane Greene's slider. These pitches have a ton of horizontal movement!
- It's comforting to see that Max Fried's curveball has the most vertical drop of any offspeed pitch considered. I did not expect, however, that Sean Newcomb, Robbie Erlin, and Jesse Biddle would have curveballs that come next on that list. I would consider these data points to be a great illustration that "stuff" isn't everything

How effectively can a simple model classify pitches?

During MLB games, it is of interest to fans and professionals alike to classify the pitches that pitchers throw (e.g., fastball, curveball) in real-time. These classifications appear on scoreboards across all 30 MLB venues and are often the subject of broadcast commentary. Additionally, databases of pitches thrown over time help analysts to better understand pitchers' tendencies and prepare scouting reports.

To perform this classification task automatically, Major League Baseball currently uses a ["patented pitch classification neural network software"] (https://technology.mlblogs.com/mlb-pitch-classification-64a1e32ee079) that, among other impressive features:

- Utilizes customized neural networks for each pitcher
- Appropriately handles new pitchers (rookies) with neural networks based on handedness and profiles of similar, established pitchers
- Automatically detects changes in pitch mix / arsenal (e.g., Luke Jackson added a change-up)

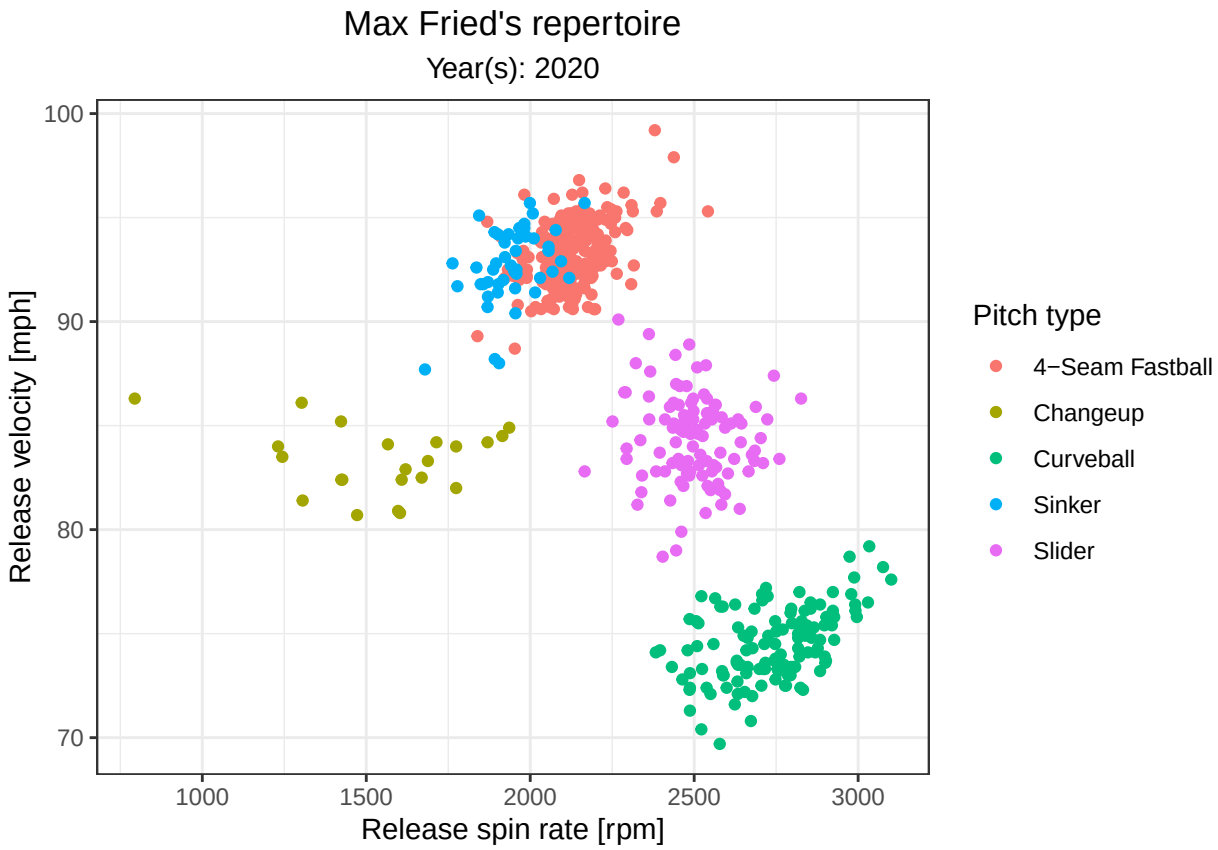
While we will by no means reproduce MLB's impressive classification work here, I'm interested to see how well we can perform the pitch classification task with a fairly simple model. A few quick notes / caveats:

- Given MLB's approach of creating a customized neural network for each pitcher, we will also build our model on one pitcher (and one year) at a time. We'll create some functions to make this process easily reproducible
- MLB trains its model on a sample of manually classified pitches, which, while still prone to some error, is likely pretty accurate. We don't have access to that same information, so we will have to accept the labels in this data set (which were generated by a model) as the truth. It's clearly not ideal to use a model to predict the output of a different model, but we're making the assumption that MLB's software is accurate the vast majority of the time
- We'll stay away from rookies in any of our test cases because there is reason to believe that MLB's model does not perform as well on these pitchers

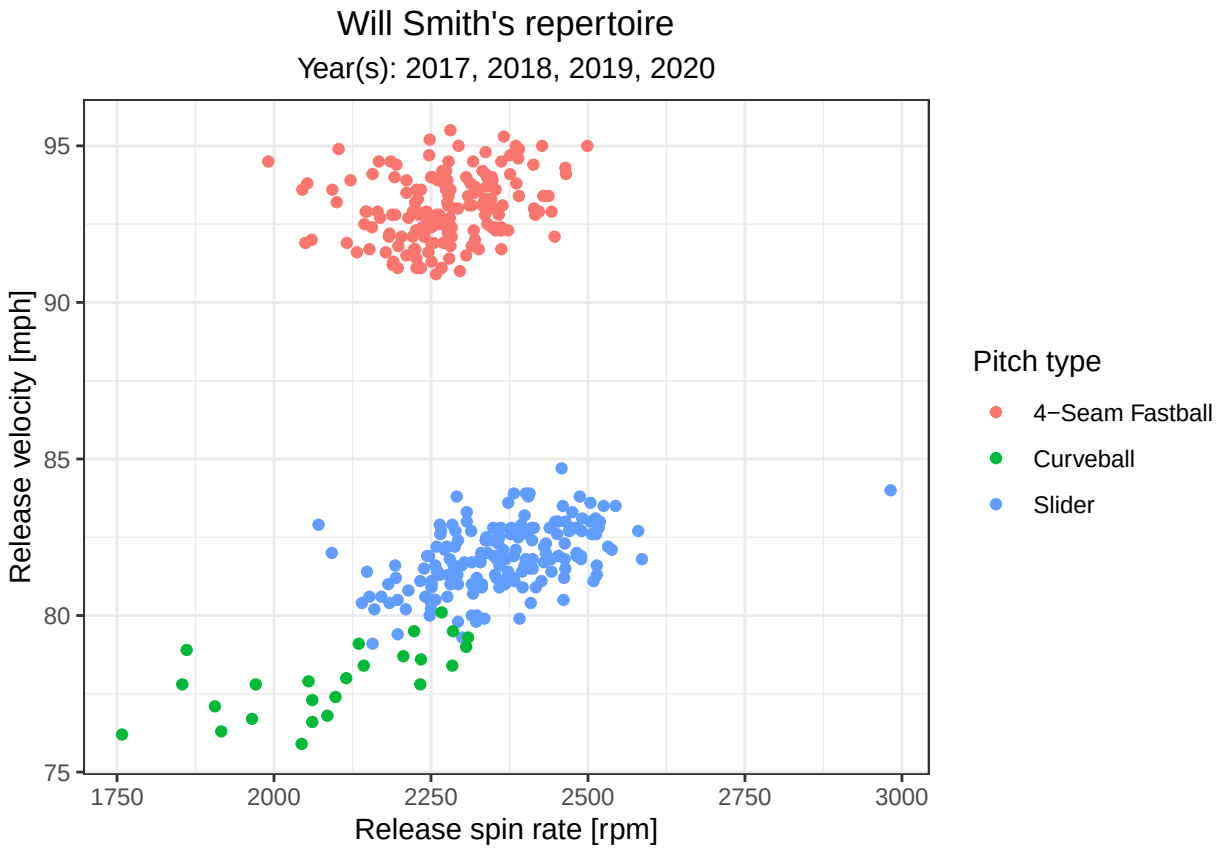
First, we'll write a function to visualize a single pitcher's arsenal over a specified period of time

```
# Generate plot of pitcher's repertoire, using stratified sample of available pitches
plot_repertoire <- function(pitcher_name, years = 2017:2020) {

  # Filter pitch-level data based on function arguments
  filter_data <- braves_pitch %>%
    filter(player_name == pitcher_name, game_year %in% years)

  # Early exit for empty data frame
  if (nrow(filter_data) == 0) {
    warning(paste0("Sorry, we don't have any data for that pitcher and year ",
    "combination. Please try again!"))
    return("")
  }

  ## Take stratified sample such that total number of points for each pitch
  ## is less than or equal to 200
  # Count incidence of pitches
```

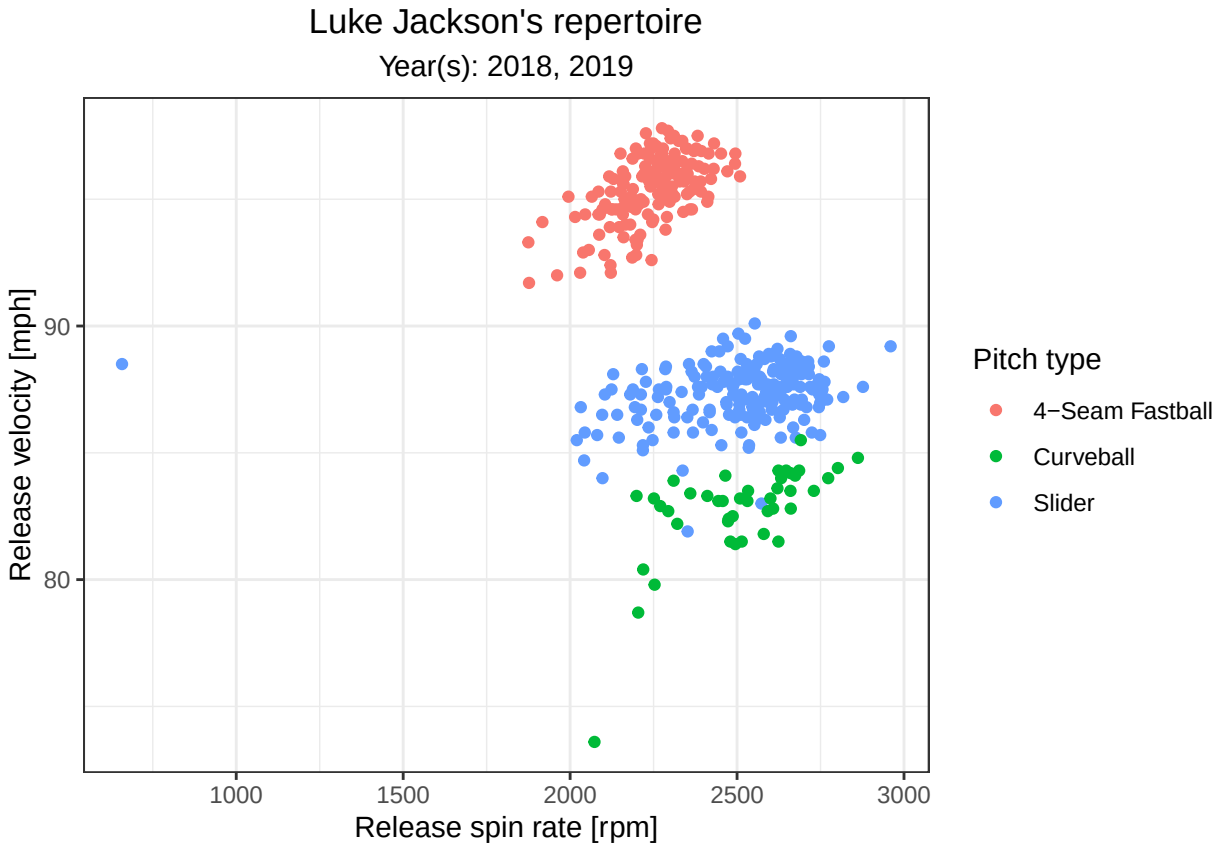
```r
  pitch_counts <- filter_data %>%
    count(pitch_name) %>%
    arrange(desc(n))

  # Calculate proportion of pitches that should be taken
  max_pitches <- 200
  sample_prop <- as.double(round(max_pitches / pitch_counts[1,2], 2))
  sample_prop <- min(sample_prop, 0.99) # Cap sample proportion

  # Take stratified sample
  sampled_pitches <- filter_data %>%
    stratified(group = "pitch_name", size = sample_prop)


  # Enumerate years
  year_vec <- paste(as.character(years), collapse = ", ")

  # Generate plot to visualize pitch clusters
  sampled_pitches %>%
    ggplot(mapping = aes(x = release_spin_rate, y = release_speed,
                         color = pitch_name)) +
    geom_point() +
    labs(x = "Release spin rate [rpm]", y = "Release velocity [mph]",
         title = paste0(pitcher_name, "'s repertoire"),
         subtitle = paste0("Year(s): ", year_vec),
         color = "Pitch type") +
    theme_bw() +
    theme(plot.title = element_text(hjust = 0.5),
          plot.subtitle = element_text(hjust = 0.5))
}

# Test function
plot_repertoire(pitcher_name = "Max Fried", years = 2020)
```




```
plot_repertoire(pitcher_name = "Will Smith")
```




```
plot_repertoire(pitcher_name = "Luke Jackson", years = 2018:2019)
```



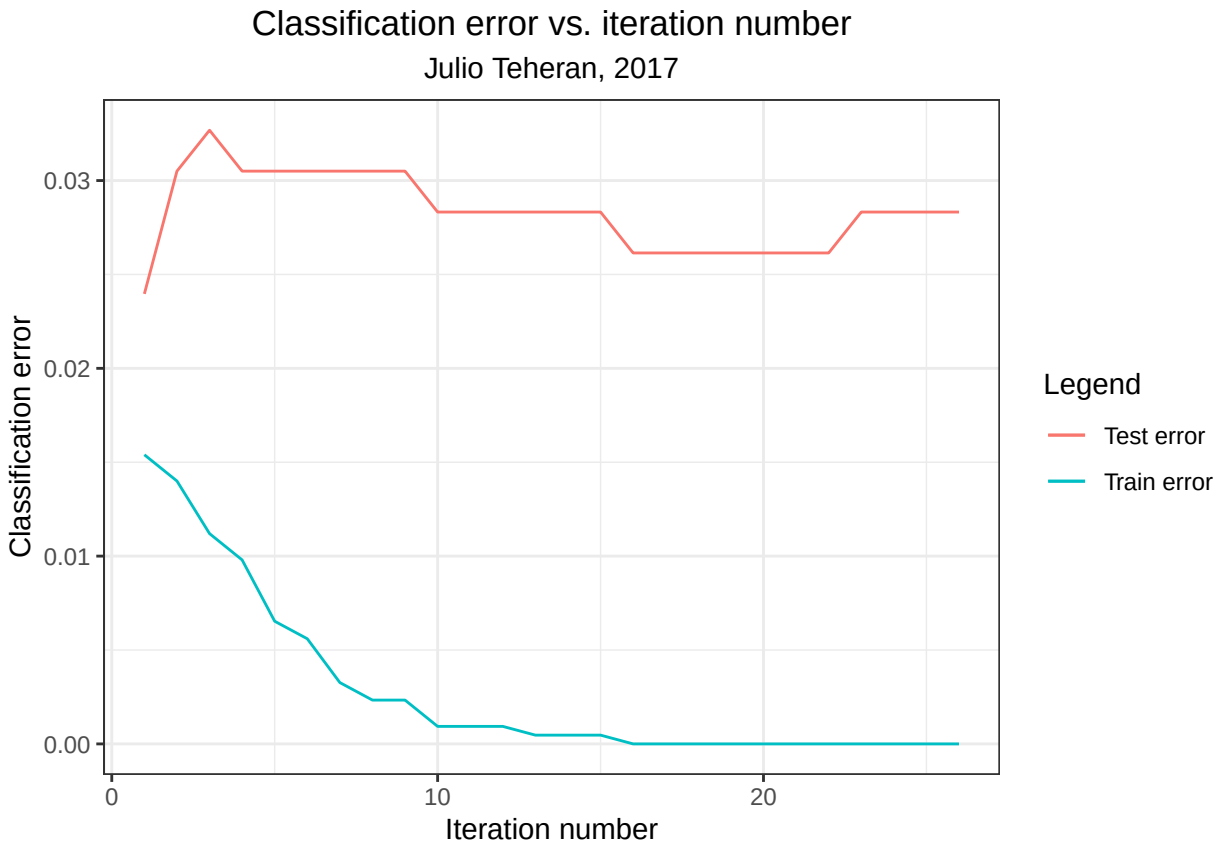

```
#plot_repertoire(pitcher_name = "Rob Kravec") # Expected to fail
#plot_repertoire(pitcher_name = "Max Fried", years = 2016) # Expected to fail

# Save plot of Max Fried's arsenal for 2020
max_fried <- plot_repertoire(pitcher_name = "Max Fried", years = 2020)
ggsave(plot = max_fried, path = "plots/", filename = "fried_arsenal.png")
```

Next, we'll transition to our modeling work flow, starting with an XGBoost model without hyperparameter tuning

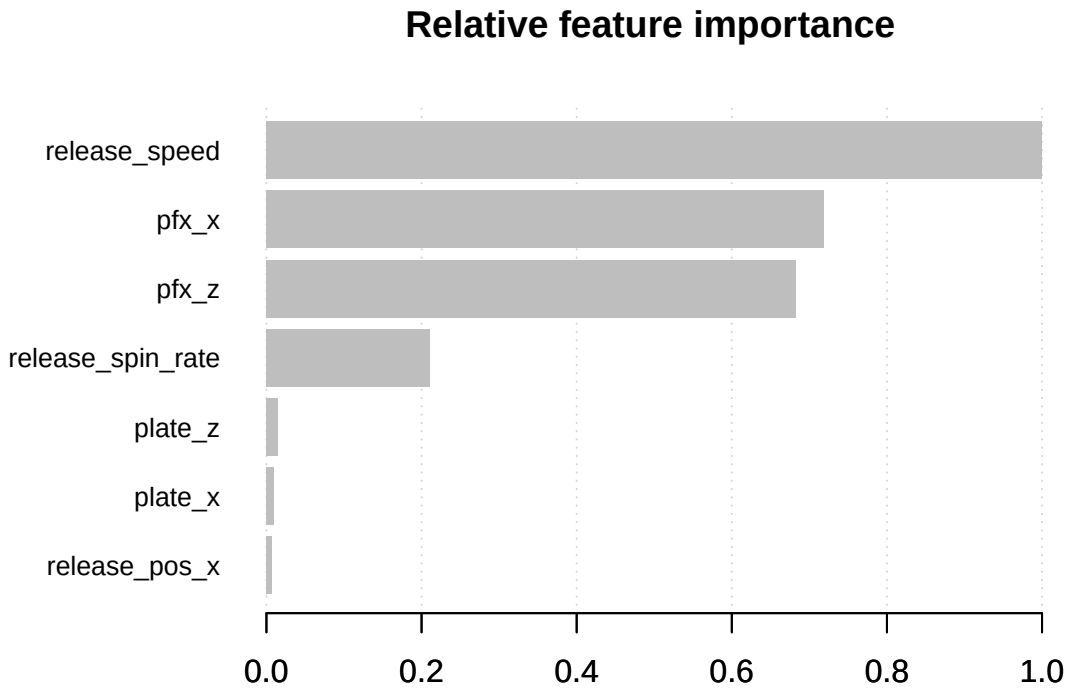
```
# Define function to build XGBoost classification model for specified pitcher
# and year
pitch_classify <- function(pitcher_name, year = 2020) {

  # Filter pitch-level data based on function arguments, and only keep columns
  # suspected to be relevant for modeling
  filter_data <- braves_pitch %>%
    filter(player_name == pitcher_name, game_year == year,
           !(pitch_name %in% c("Eephus", "Pitch Out")),
           !is.null(pitch_name), !is.na(pitch_name), pitch_name != "null") %>%
    select(pitch_name, release_speed, release_pos_x, release_pos_y,
           stand, balls, strikes, pfx_x, pfx_z, plate_x, plate_z, outs_when_up,
           inning, release_spin_rate, release_extension, release_pos_y) %>%
    mutate(pitch_name = as.factor(pitch_name))

  # Early exit for empty data frame
  if (nrow(filter_data) == 0) {
    warning(paste0("Sorry, we don't have any data for that pitcher and year ",
```

```
  "combination. Please try again!"))
  return("")
}

## Create 70/15/15 train/test/validation split
train_rows <- sample(1:nrow(filter_data),
                     size = floor(0.7 * nrow(filter_data)),
                     replace = F)
training <- filter_data[train_rows, ]
train_labels <- training[, 1] # Store labels, convert to numeric for XGBoost
training[, 1] <- as.numeric(training$pitch_name) - 1
# Now that 70% training data has been separated, split remainder evenly
# between test and validation
not_training <- filter_data[-train_rows, ]
test_rows <- sample(1:nrow(not_training),
                     size = floor(0.5 * nrow(not_training)),
                     replace = F)
testing <- not_training[test_rows, ]
test_labels <- testing[, 1] # Store labels, convert to numeric for XGBoost
testing[, 1] <- as.numeric(testing$pitch_name) - 1
validation <- not_training[-test_rows, ]
validation_labels <- validation[, 1] # Store labels, convert to numeric for XGBoost
validation[, 1] <- as.numeric(validation$pitch_name) - 1

# Create key of numbers to pitch names
pitch_key <- data.frame(word = train_labels$pitch_name,
                        number = training$pitch_name) %>%
  count(word, number) %>%
  arrange(number)

## Properly format train, test, and validation data sets
options(na.action = "na.pass") # Keep rows with NAs

# Train data
train_data <- model.matrix(pitch_name ~ ., data = training)
train_data_xg <- xgb.DMatrix(data = train_data,
                             label = training$pitch_name)
# Test data
test_data <- model.matrix(pitch_name ~ ., data = testing)
test_data_xg <- xgb.DMatrix(data = test_data,
                             label = testing$pitch_name)

# Validation data
validation_data <- model.matrix(pitch_name ~ ., data = validation)
validation_data_xg <- xgb.DMatrix(data = validation_data)
validation_labels_numeric <- validation[, 1]

# Fit model. We will use the model iteration for which the test error is minimized
params <- list(booster = "gbtree",
               objective = "multi:softmax", # multi-class classification
               num_class = length(unique(training$pitch_name))
               )
watch <- list(train = train_data_xg, test = test_data_xg)
```

```
  xgb_fit <- xgb.train(params = params,
                       data = train_data_xg,
                       nrounds = 1000,
                       print_every_n = 1000,
                       eval_metric = "merror", #(wrong cases) / #(all cases)
                       early_stopping_rounds = 25,
                       watchlist = watch,
                       verbose = F
                       )

  # Plot training error and classification error
  plot_data <- xgb_fit$evaluation_log
  plot1 <- ggplot(data = plot_data, mapping = aes(x = iter)) +
    geom_line(mapping = aes(y = train_merror, color = "Train error")) +
    geom_line(mapping = aes(y = test_merror, color = "Test error")) +
    labs(x = "Iteration number", y = "Classification error",
         title = "Classification error vs. iteration number",
         subtitle = paste0(pitcher_name, ", ", year),
         color = "Legend") +
    theme_bw() +
    theme(plot.title = element_text(hjust = 0.5),
          plot.subtitle = element_text(hjust = 0.5))
  print(plot1)

  # Predict labels on validation set
  validation_pred <- predict(xgb_fit, validation_data_xg)
  validation_pred_df <- data.frame(number = validation_pred)
  validation_pred_df <- left_join(x = validation_pred_df, y = pitch_key,
                                  by = "number")

  # Quantify validation set error rate
  print(paste0("Validation classification accuracy: ",
               round(mean(validation_pred_df$number ==
                            validation_labels_numeric$pitch_name), 4)))

  # Display confusion matrix
  conf_matrix <- confusionMatrix(data = as.factor(validation_pred_df$word),
                  reference = validation_labels$pitch_name)$table
  print(conf_matrix)

  # Display feature importance
  importance_matrix <- xgb.importance(model = xgb_fit)
  xgb.plot.importance(importance_matrix = importance_matrix, top_n = 7,
                      rel_to_first = T, main = "Relative feature importance",
                      sub = paste0(pitcher_name, ", ", year))
}

# Test function on pitcher with most pitches in each year (without repeats)
pitch_classify(pitcher_name = "Julio Teheran", year = 2017)
```

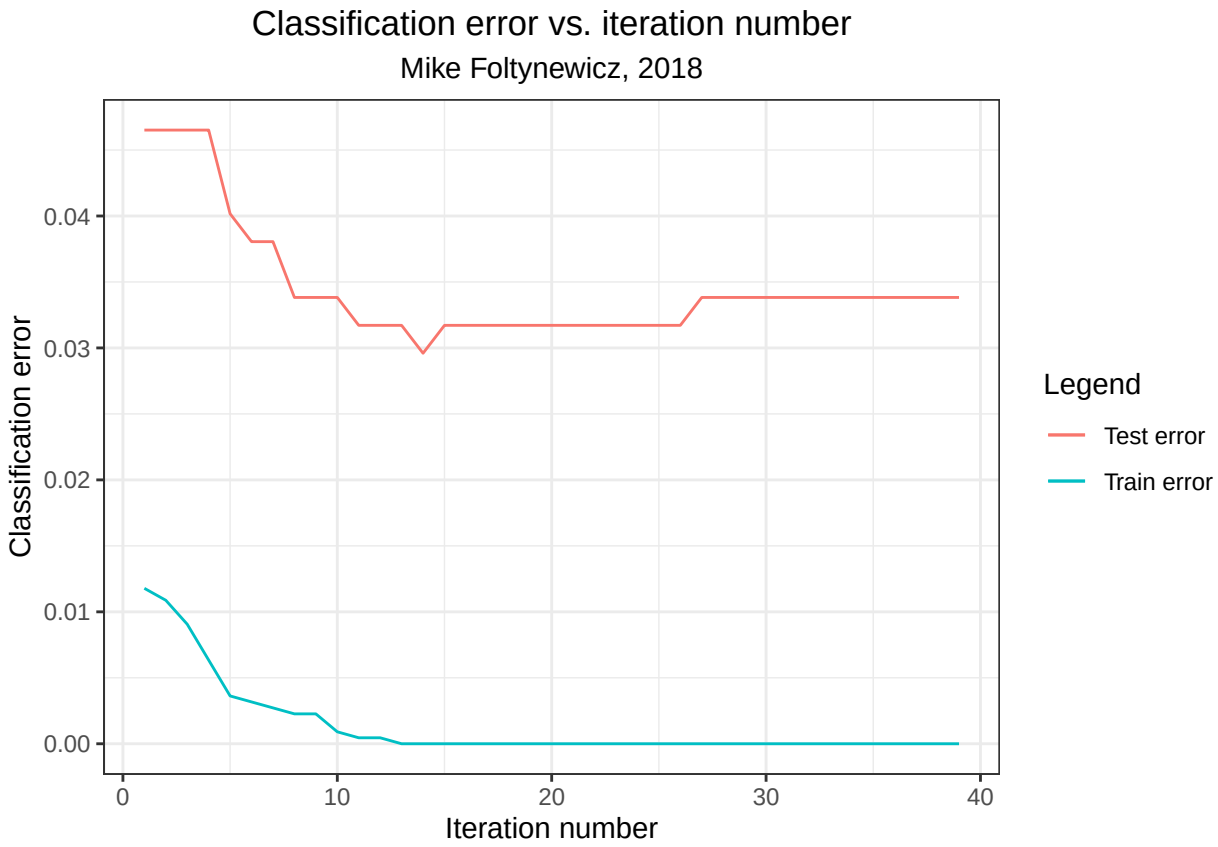



```
## [1] "Validation classification accuracy: 0.9761"
##                  Reference
## Prediction        2-Seam Fastball 4-Seam Fastball Changeup Curveball Slider
##   2-Seam Fastball             119               2        1         0      0
##   4-Seam Fastball               3             177        0         0      0
##   Changeup                      1               0       26         0      1
##   Curveball                     0               0        0        42      1
##   Slider                        0               0        2         0     85
```

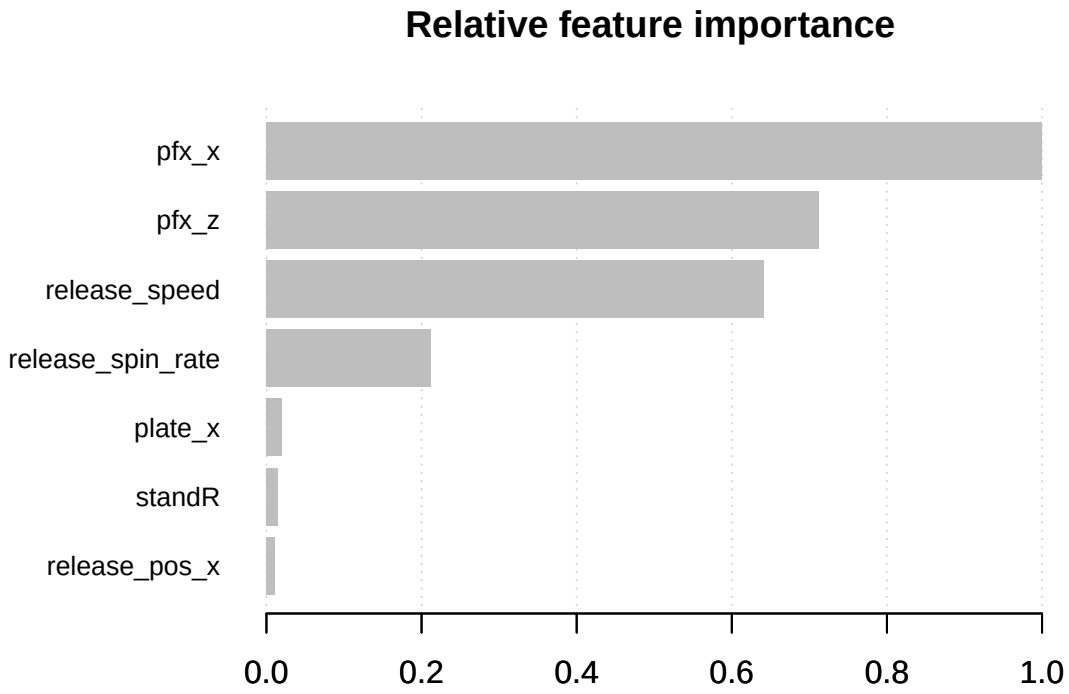


Julio Teheran, 2017

```
pitch_classify(pitcher_name = "Mike Foltynewicz", year = 2018)
```

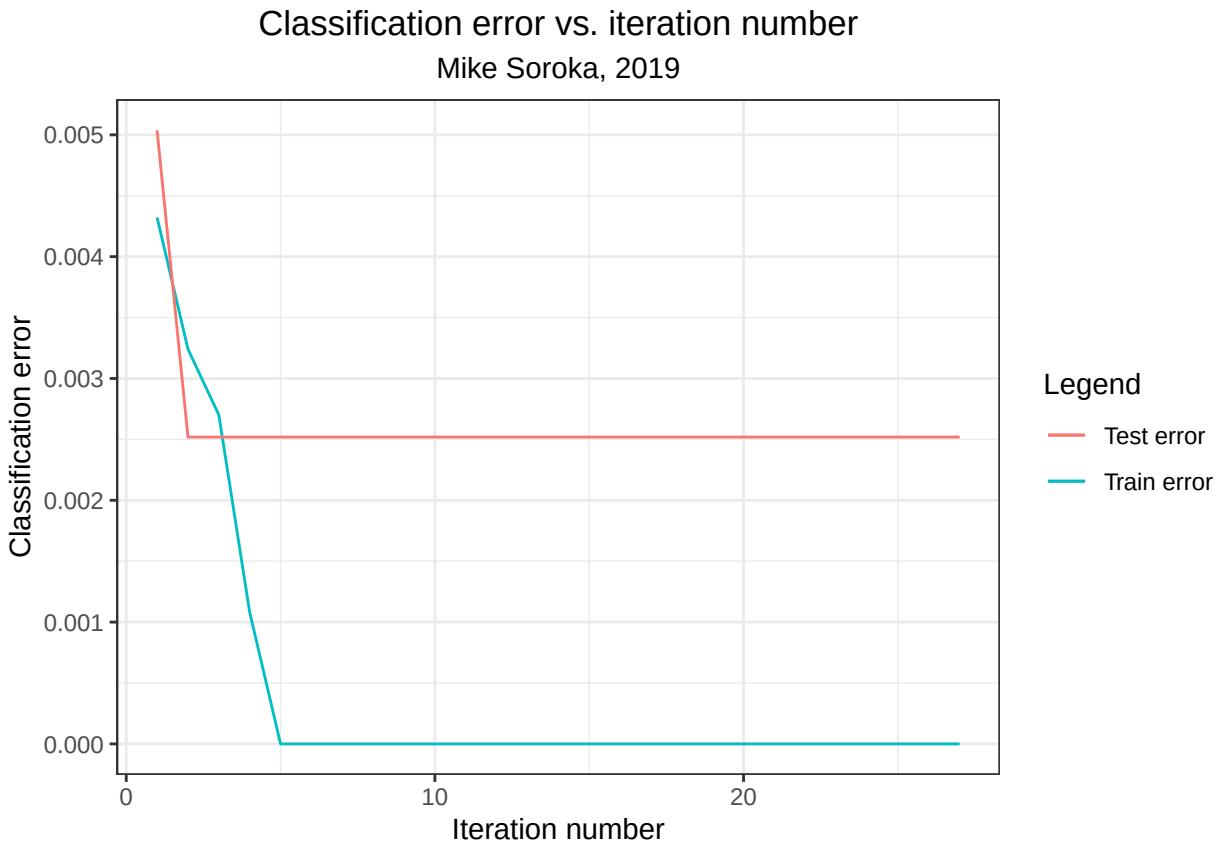


```
## [1] "Validation classification accuracy: 0.9767"
##                  Reference
## Prediction        2-Seam Fastball 4-Seam Fastball Changeup Curveball Slider
##   2-Seam Fastball              71               2        1         0      0
##   4-Seam Fastball               2             186        0         0      0
##   Changeup                      0               0       26         0      0
##   Curveball                     0               0        0        46      3
##   Slider                        0               0        0         3    133
```

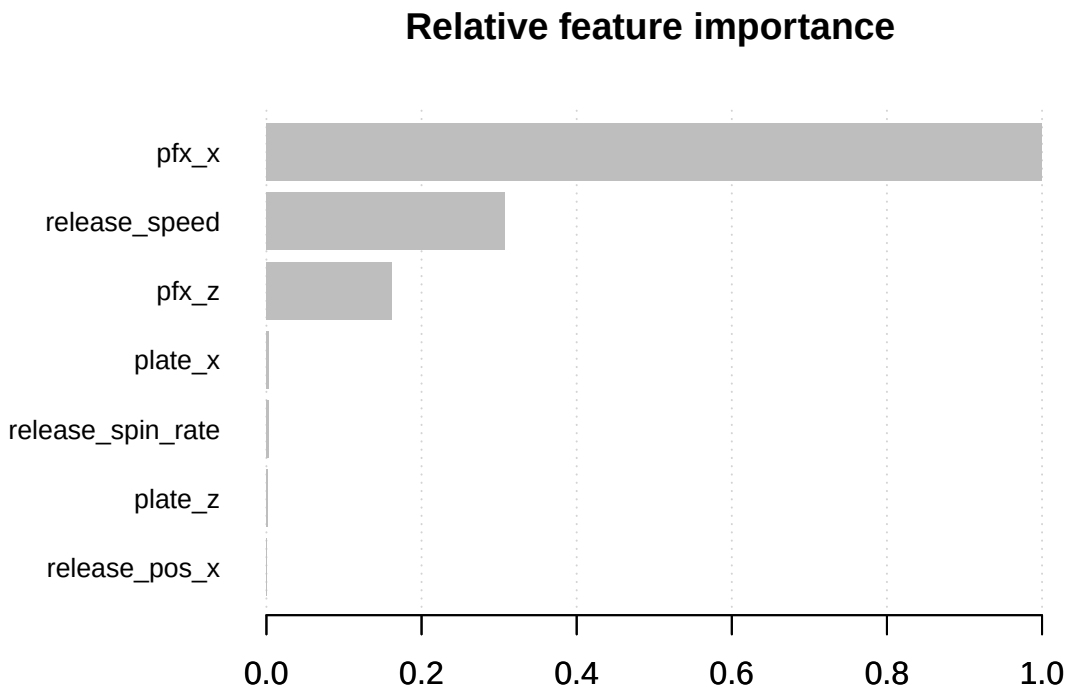


Mike Foltynewicz, 2018

```
pitch_classify(pitcher_name = "Mike Soroka", year = 2019)
```

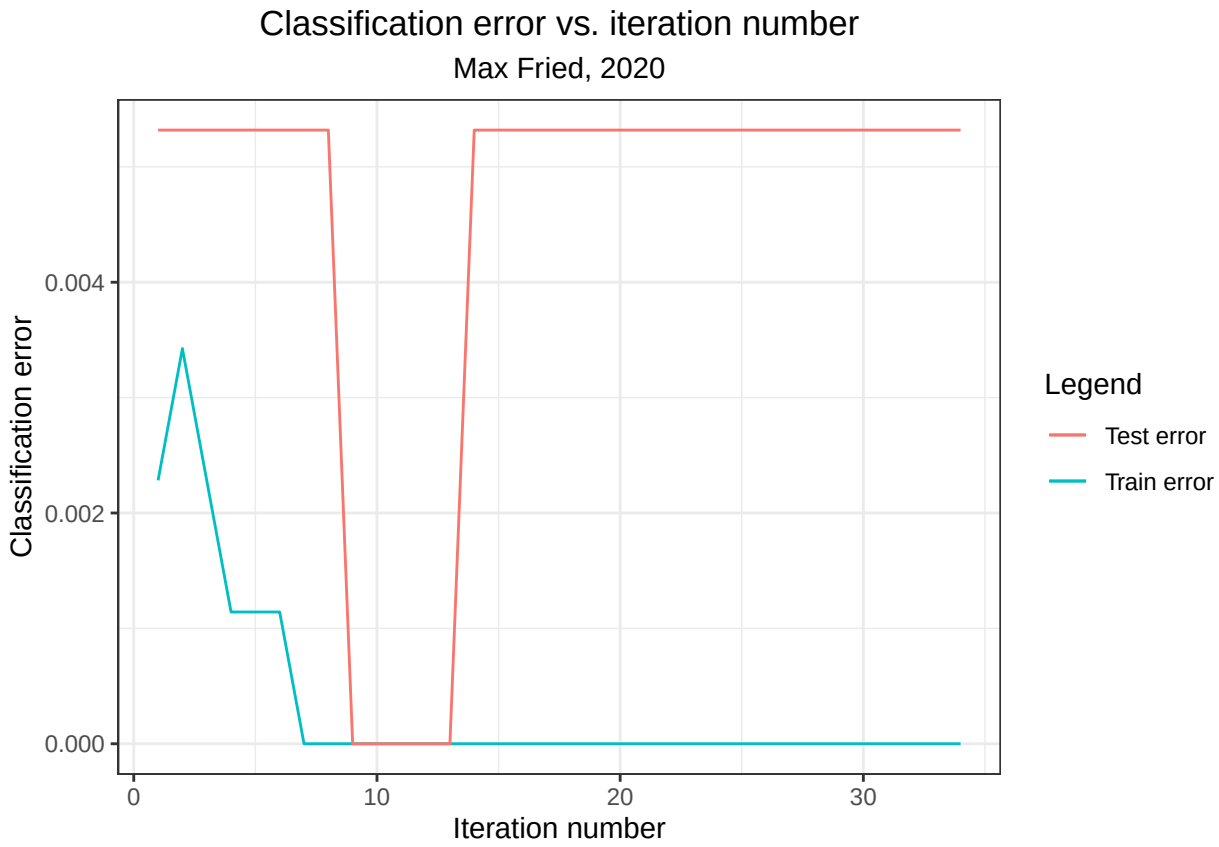



```
## [1] "Validation classification accuracy: 0.995"
##                  Reference
## Prediction        2-Seam Fastball 4-Seam Fastball Changeup Slider
##   2-Seam Fastball             179               1        0      0
##   4-Seam Fastball               0              80        1      0
##   Changeup                      0               0       48      0
##   Slider                        0               0        0     88
```

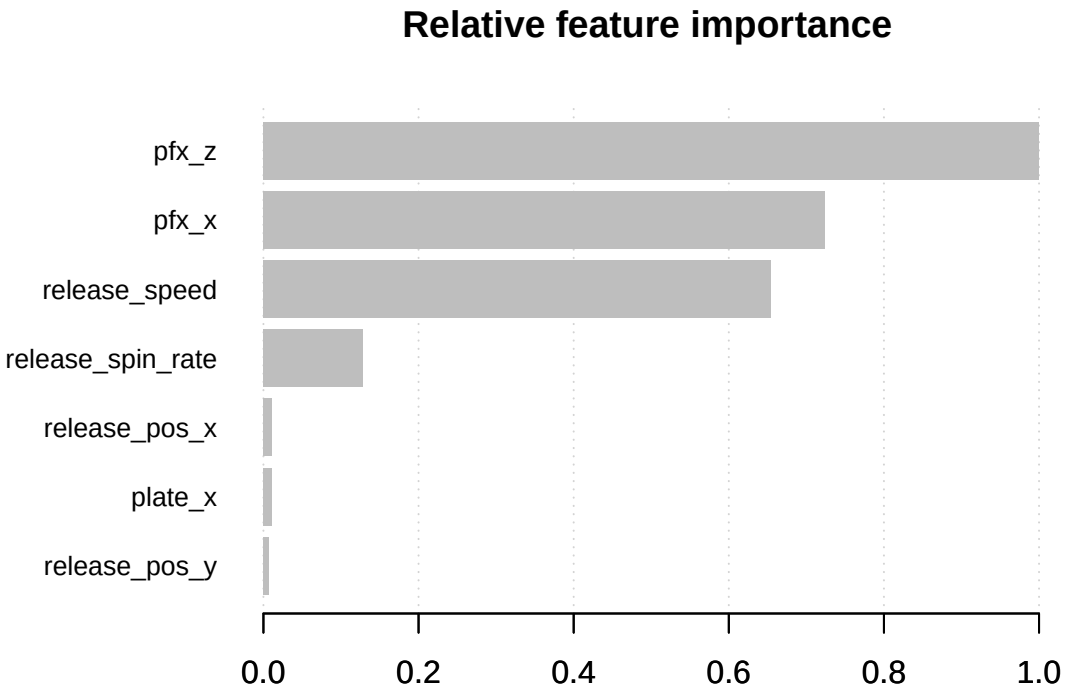


Mike Soroka, 2019

```
pitch_classify(pitcher_name = "Max Fried", year = 2020)
```



```
## [1] "Validation classification accuracy: 0.9947"
##                  Reference
## Prediction        4-Seam Fastball Changeup Curveball Sinker Slider
##   4-Seam Fastball              67        0         0      0      0
##   Changeup                      0       10         0      0      0
##   Curveball                     0        0        52      0      0
##   Sinker                        1        0         0     16      0
##   Slider                        0        0         0      0     42
```



Max Fried, 2020

At this point, we could theoretically work on hyperparameter tuning to further improve the performance of our models on unseen data. However, the validation set performance is already quite good in the examples examined, ranging from 96.2% - 99.5%. As a result, I'd say that we've accomplished our stated objective of determining whether a rather simple model could do a good job classifying pitches.

A couple of observations:

- It appears that the decision to model each pitcher separately is warranted. Looking at relative feature importance plots, we can see that pitch movement is most important for some pitchers (e.g., Mike Foltynewicz, Max Fried), while velocity plays the largest role for others (e.g., Julio Teheran). In the future, it would be interesting to investigate how much model performance would degrade if multiple pitchers were pooled together.
- As expected, 2-Seam Fastballs can be tricky to distinguish from 4-Seam Fastballs. Fastballs are occasionally classified as changeups (and vice versa). Breaking pitches (e.g., curveball, slider) are another potential culprit of misclassification

A couple of caveats:

- The examples shown in this document are all starting pitchers with high numbers of pitchers thrown. This modeling work flow was also tested on relief pitchers (not pictured), and results were similarly good

- This exercise is by no means meant to de-legitimize the complex modeling efforts that MLB undertakes for pitch classification. MLB faces additional challenges that may warrant more complex modeling, including but not limited to:

1. Classifying future pitches with past data. In our modeling effort, we simply consider a year's worth of pitches and split into training, testing, and validation data. We also do not consider rookie pitchers without a track record of past pitches thrown
2. Facing scrutiny from fans and professionals. It is possible that some of the validation classification results that look quite good to us (e.g., 96% accuracy) would actually be unacceptable on a national broadcast

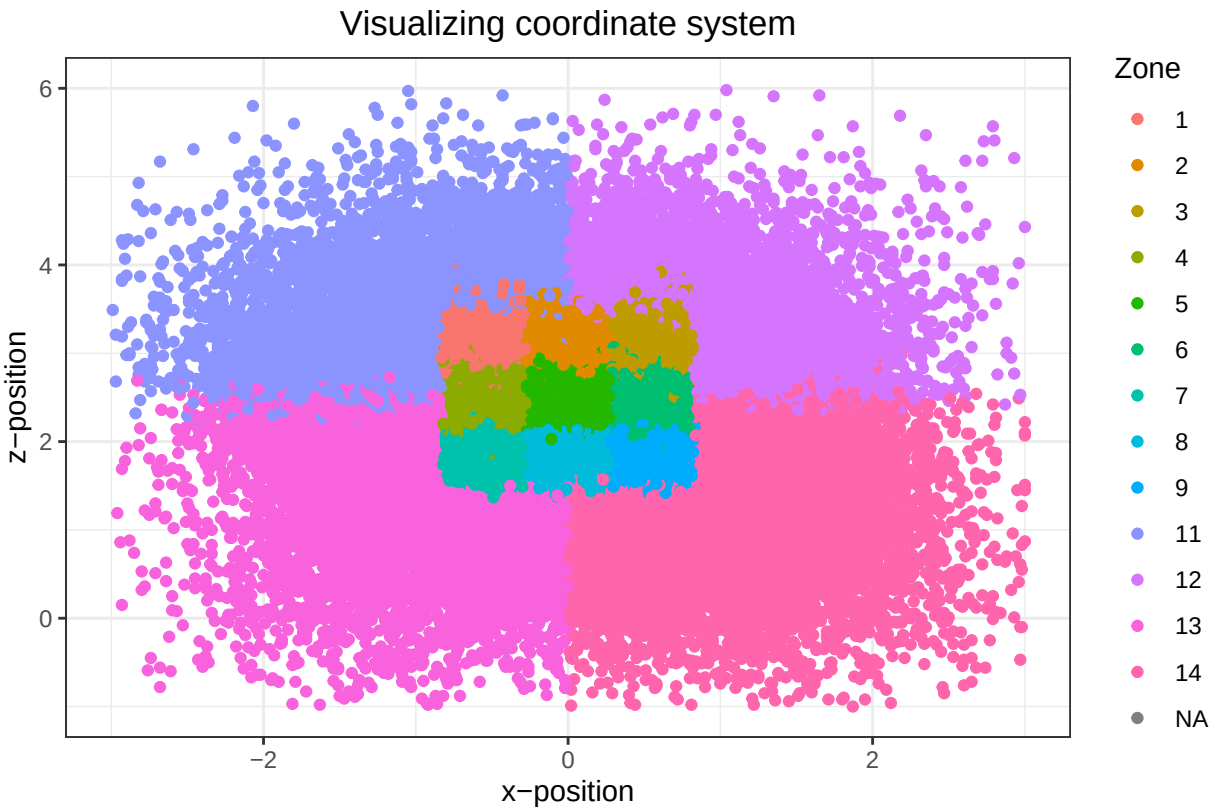
How often do umpires make the right ball / strike calls? Are certain pitchers more prone to inaccurate calls?

To start with, we'll have to define an "objective strike zone" based on the Statcast data. I'll propose the following rules /assumptions (based mostly on trial and error with plotted results):

- The width (horizontal dimension) of the strike zone is equal to the width of home plate (17 inches)
- The height (vertical dimension) of the strike zone is appropriately described by the `sz_top` and `sz_bot` fields
- The `plate_x` and `plate_z` fields are defined based on the center of the baseball at the time it crosses home plate
- A regulation baseball has a diameter of 3 inches
- For a pitch to be called a strike, the baseball must at least touch the edge of the strike zone

Based on these rules / assumptions, we create an indicator to see whether a pitch should have been marked as a ball or a strike. Then, we can compare the true outcome to our expected outcome.

```
# Visualize coordinates for each Statcast zone (just to make sure I'm
# understanding the coordinate system properly)
coord_system <- ggplot(data = braves_pitch,
       mapping = aes(x = plate_x, y = plate_z, color = as.factor(zone))) +
  geom_point() +
  labs(x = "x-position", y = "z-position",
       title = "Visualizing coordinate system",
       color = "Zone",
       caption = "We can conclude that the horizontal axis is centered at 0") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.caption = element_text(hjust = 0.5)) +
  lims(x = c(-3, 3), y = c(-1, 6))
coord_system
```



We can conclude that the horizontal axis is centered at 0

```
# The plot turned out to be quite aesthetically pleasing, so I'll save it
ggsave(plot = coord_system, path = "plots/", filename = "coord_system.png")

# Create expected ball / strike indicator
ball_rad <- 1.5 / 12 # Baseball radius in feet
plate_width <- 17 / 12 # Width of front of plate in feet
braves_pitch <- braves_pitch %>%
  mutate(exp_call = case_when(
    type == "X" ~ "X", # If ball is put in play, we don't call it ball or strike
    abs(plate_x) > plate_width / 2 + ball_rad ~ "B", # Pitch is inside or outside
    plate_z > sz_top + ball_rad ~ "B", # Pitch is high
    plate_z < sz_bot - ball_rad ~ "B", # Pitch is low
    TRUE ~ "S" # All other pitches should be called a strike
  ))

### Quick visualization of our expected ball and strike calls vs. reality:
### For this visualization, it would be best to plot the strikes on top of the
### balls to see the extent of the true strike zone

# Create data frame with only balls and called strikes
taken_pitches <- braves_pitch %>%
  filter(description %in% c("ball", "called_strike"))

# Step 1: Plot of expected calls
expected_call_plot <- ggplot() +
  geom_point(subset(taken_pitches, exp_call == "B"),
             mapping = aes(x = plate_x, y = plate_z, color = "B")) +
```

```
  geom_point(subset(taken_pitches, exp_call == "S"),
             mapping = aes(x = plate_x, y = plate_z, color = "S")) +
  labs(x = "x-position", y = "z-position",
       title = "Expected ball / strike calls",
       color = "Expected call") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5)) +
  lims(x = c(-3, 3), y = c(-1, 6))

# Step 2: Plot of actual calls
actual_call_plot <- ggplot() +
  geom_point(subset(taken_pitches, type == "B"),
             mapping = aes(x = plate_x, y = plate_z, color = "B")) +
  geom_point(subset(taken_pitches, type == "S"),
             mapping = aes(x = plate_x, y = plate_z, color = "S")) +
  labs(x = "x-position", y = "z-position",
       title = "Actual ball / strike calls",
       color = "Actual call") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5)) +
  lims(x = c(-3, 3), y = c(-1, 6))

# Step 3: Patchwork combination of two plots above
strikezone_compare <- expected_call_plot + actual_call_plot

# Save plot
ggsave(plot = strikezone_compare, path = "plots/",
       filename = "strikezone_compare.png")
```

From this comparison plot, we can see that:

- Our "objective" strike zone may be a bit narrower than the one that most umpires enforce. A strike heat map would do a better job showing how often pitches on the horizontal fringes of the strike zone are called strikes
- The actual strike zone follows more of an oval than a rectangle, suggesting that umpires will allow pitches to push the bounds in one direction (but not both at the same time)

From here, we can compare our expected ball / strike call with the actual call in a more quantitative manner

```
# Add column to denote whether expected call agrees with actual call
taken_pitches <- taken_pitches %>%
  relocate(type, .after = of_fielding_alignment) %>%
  mutate(correct_call = ifelse(type == exp_call, 1, 0))

# How often is the expected call made?
mean(taken_pitches$correct_call)
```

```
## [1] 0.8997532
```

```
# In which direction is there more likely to be a discrepancy?
mean(subset(taken_pitches, exp_call == "B")$correct_call) # We expect B, called B
```

```
## [1] 0.911963
```

```
mean(subset(taken_pitches, exp_call == "S")$correct_call) # We expect S, called S
```

```
## [1] 0.8724974
```

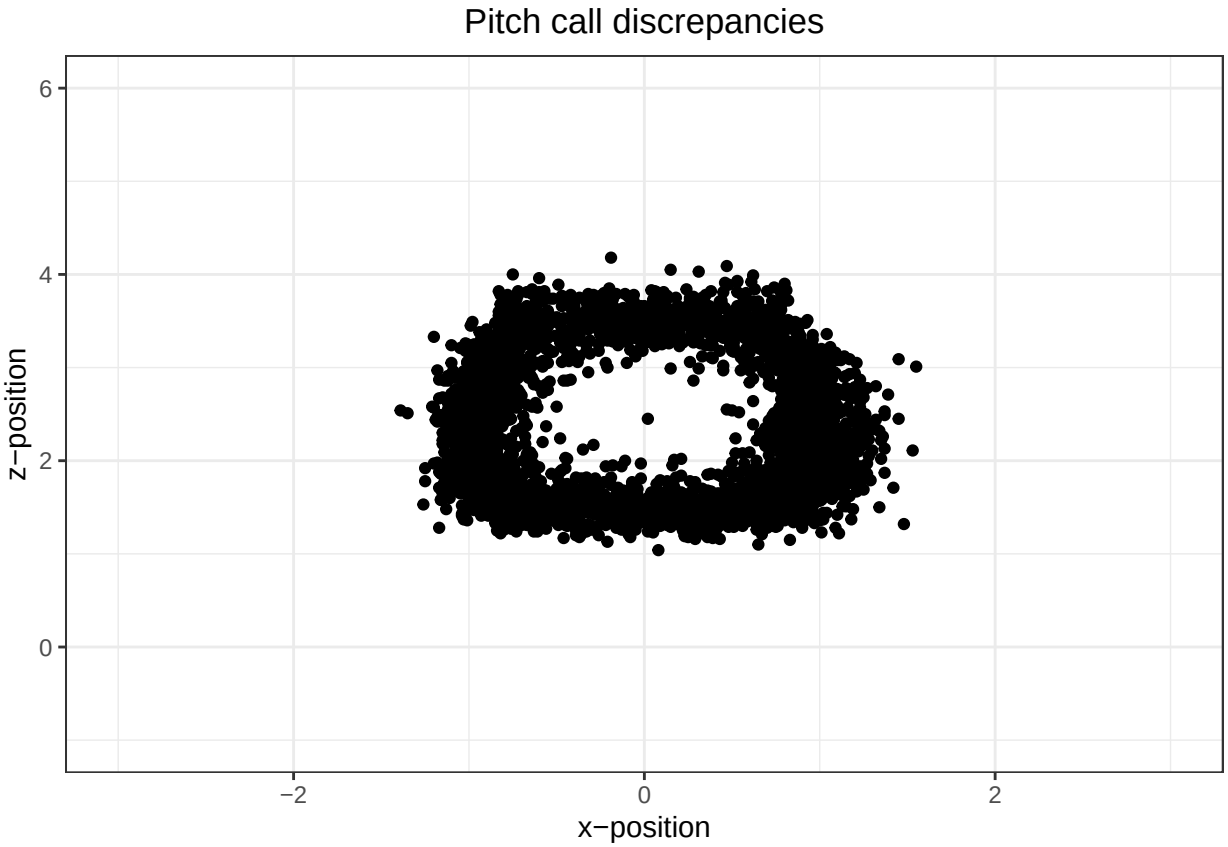
```r
mean(subset(taken_pitches, type == "B")$correct_call) # Called B, expect B
```

```
## [1] 0.9410598
```

```r
mean(subset(taken_pitches, type == "S")$correct_call) # Called S, expect S
```

```
## [1] 0.8161656
```

```r
# Plot discrepancies
taken_pitches %>%
  filter(correct_call == 0) %>%
  ggplot(mapping = aes(x = plate_x, y = plate_z)) + geom_point() +
  theme_bw() +
  labs(x = "x-position", y = "z-position",
       title = "Pitch call discrepancies") +
  theme(plot.title = element_text(hjust = 0.5)) +
  lims(x = c(-3, 3), y = c(-1, 6))
```



We observe 90% agreement between our proposed strike zone and actual ball / strike calls. The highest incidence of disagreement occurs on pitches that are called strikes in reality (of which we expect 18% to be called balls).

From here, we could look at whether a number of different factors (e.g., count, batter handedness, pitcher handedness, score, runners on base) affect the size of the strike zone. For example, I found a 2012 Fangraphs article that measures the area of the strike zone by count and batter handedness.

Instead, we'll pump the breaks here and recognize that strike zones are quite complex and could easily be the subject of their own, dedicated project. We quickly check to see if there is a noticeable trend for Braves pitchers related to the percentage of "missed calls" Sadly, nothing in particular stands out to me (other the

large spread in incidence of these "missed calls"), and I'd like to give this topic more time and attention down the road.

```
# Create indicator of different miss types
taken_pitches <- taken_pitches %>%
  mutate(disagree_called_ball = ifelse(correct_call == 0 & type == "B", 1, 0),
         disagree_called_strike = ifelse(correct_call == 0 & type == "S", 1, 0),
         ball = ifelse(type == "B", 1, 0),
         strike = ifelse(type == "S", 1, 0)
         )

# Summarise results by pitcher
miss_summary <- taken_pitches %>%
  group_by(player_name) %>%
  summarise(pitches = n(),
            strikes = sum(strike),
            balls = sum(ball),
            miss_call = 1 - mean(correct_call),
            miss_called_ball = sum(disagree_called_ball),
            miss_called_strike = sum(disagree_called_strike)
            ) %>%
  mutate(stolen_strike_perc = miss_called_strike / strikes,
         unwarranted_ball = miss_called_ball / balls,
         delta = miss_called_strike / strikes - miss_called_ball / balls) %>%
  filter(pitches >= 100, strikes >= 50) %>%
  arrange(desc(miss_call))
```

Future wish list

EDA

- More in-depth analysis of strike zones (using linked article and others available online as starting points)
- Pitch velocity over time for pitchers (during game, during season, year-to-year)
- In-depth scouting report on particular pitcher(s) of interest

Visualization

- Plot the trajectory of a pitch (possibly creating mock-up strike zone and MLB Gameday-type aesthetic)

Modeling

- Predict which pitch will be thrown in certain situations