Pitch Clustering

Camden Kay

2022-06-05

Setup

Background

- Traditional pitch classifications are broken.
- Pitch names give people a general idea of what a pitch moves like and how hard it's thrown, but the lack of detail can cloud analysis when looking at specific pitch types.
- One pitcher's slider might match another pitcher's curveball metrics well, but they would traditionally be separated into different buckets for analyzing (unless one is looking at the fastball/breaking ball/offspeed split).
- Using clusters allows analysts to better dive into why player X's and Y's pitches may be performing differently when very similar metrically.

```r
library(tidyverse)
library(cluster)
library(umap)
library(gt)
library(gtExtras)
# library(tRead) - personal package made for common tasks
set.seed(1)

needed_columns <- c("pitch_name", "release_speed",
                    "pfx_x_pv_adj", "pfx_z",
                    "release_spin_rate")

# Load game data and remove those who didn't throw at least 150 pitches
data <- tRead::load_seasons(2021) |>
  filter(game_type == "R") |>
  group_by(pitcher) |>
  filter(n() >= 150,
         !pitch_name %in% c("Eephus", "Fastball", "Screwball")) |>
  ungroup() |>
  drop_na(all_of(needed_columns)) |>
  tRead::add_est_spin_efficiency() |>
  drop_na(est_spin_efficiency)
```

```r
# Find "average" FB
p_avgs <- data |>
  group_by(game_year, pitcher) |>
  # Top 10% of hardest pitches thrown are used as the av. FB
  top_frac(0.10, release_speed) |>
  summarize(avg_velo = mean(release_speed, na.rm=TRUE))

# Combine data with "averages"
raw_data <- data |>
  left_join(p_avgs,  by = c("game_year", "pitcher")) |>
  mutate(velo_ratio = if_else(release_speed/avg_velo > 1,
                              1, release_speed/avg_velo))

# Getting pitch averages differences
cleaned_mlb <- raw_data |>
  group_by(pitcher, player_name, pitch_name, pitch_type) |>
  summarize(avg_velo_ratio = mean(velo_ratio, na.rm = TRUE)*100,
            avg_horz = mean(pfx_x_pv_adj, na.rm = TRUE),
            avg_vert = mean(pfx_z, na.rm = TRUE),
            avg_eff = mean(est_spin_efficiency, na.rm = TRUE)) |>
  ungroup()
```

Clustering

Make the Model

```
cluster_data <- cleaned_mlb |>
  select("avg_velo_ratio", "avg_vert", "avg_horz", "avg_eff")

# Create clusters
cleaned_clusters <- pam(cluster_data, k = 17, metric = "euclidean")

# Save Medoids
write_csv(cleaned_clusters$medoids |> as_tibble(), "./Medoids.csv")
```

- I came up with 17 different subcategories within traditional pitch types
- This number is obviously affected by prior knowledge and plays a role in biasing the results of the analysis considering k-means clusters takes an input for number of clusters to produce.

Return Cluster Function

```r
# Loading saved medoids
saved_clusters <- read_csv("./Medoids.csv")

# Functions to return pitch cluster
eucDist <- function(x, y) sqrt(sum( (x-y)^2 ))

classifyNewSample <- function(newData, centroids = saved_clusters) {
  dists = apply(centroids, 1, function(y) eucDist(y,newData))
  order(dists)[1]
}

# Add clusters based on saved Medoids
mlb_clusters <- cleaned_mlb |>
  mutate(cluster = apply(cluster_data, 1, classifyNewSample),
         cluster = as.factor(cluster))
```

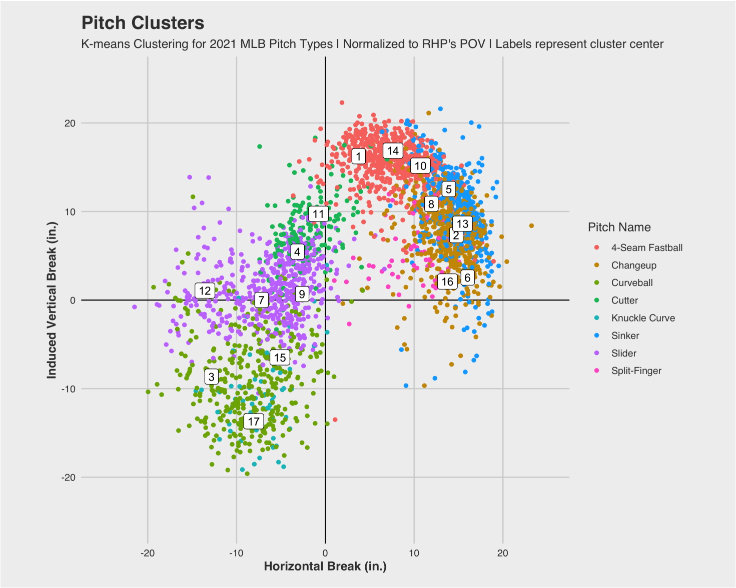
Analysis

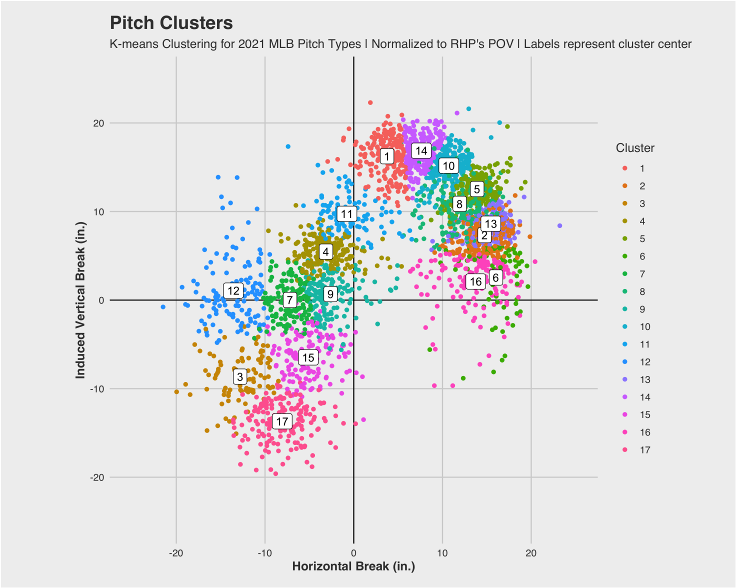
Table

```r
#Analyze old pitch_names with clusters
table(mlb_clusters$cluster, mlb_clusters$pitch_type)
# Old pitch names map out well with new clusters
# There's only a few "weird" results
```

```
## 
##        CH  CS  CU  FC  FF  FS  KC  SI  SL
##   1     0   0   0  11 193   0   0   0   0
##   2   201   0   0   0   1   8   0   1   0
##   3     0   2  78   0   0   0   7   0   7
##   4     1   0   0  76   0   2   0   0 128
##   5     6   0   0   0  38   0   0 136   0
##   6     4   0   0   0   5   0   0  68   0
##   7     0   0  17   4   0   0   2   0 134
##   8   174   0   0   0   0  13   0   0   1
##   9     3   0  11  15   0   5   3   0 133
##   10    3   0   0   0 148   0   0  55   0
##   11    1   0   0  87  15   0   0   0  12
##   12    0   1  26   0   0   0   0   0 101
##   13    2   0   0   0  12   0   0 152   0
##   14    4   0   0   1 241   0   0  14   0
##   15    0   5  92   0   1   0  17   0  32
##   16  131   0   0   0   0  23   0   1   1
##   17    0  10 145   0   0   0  32   0   1
```

Clustered Plots



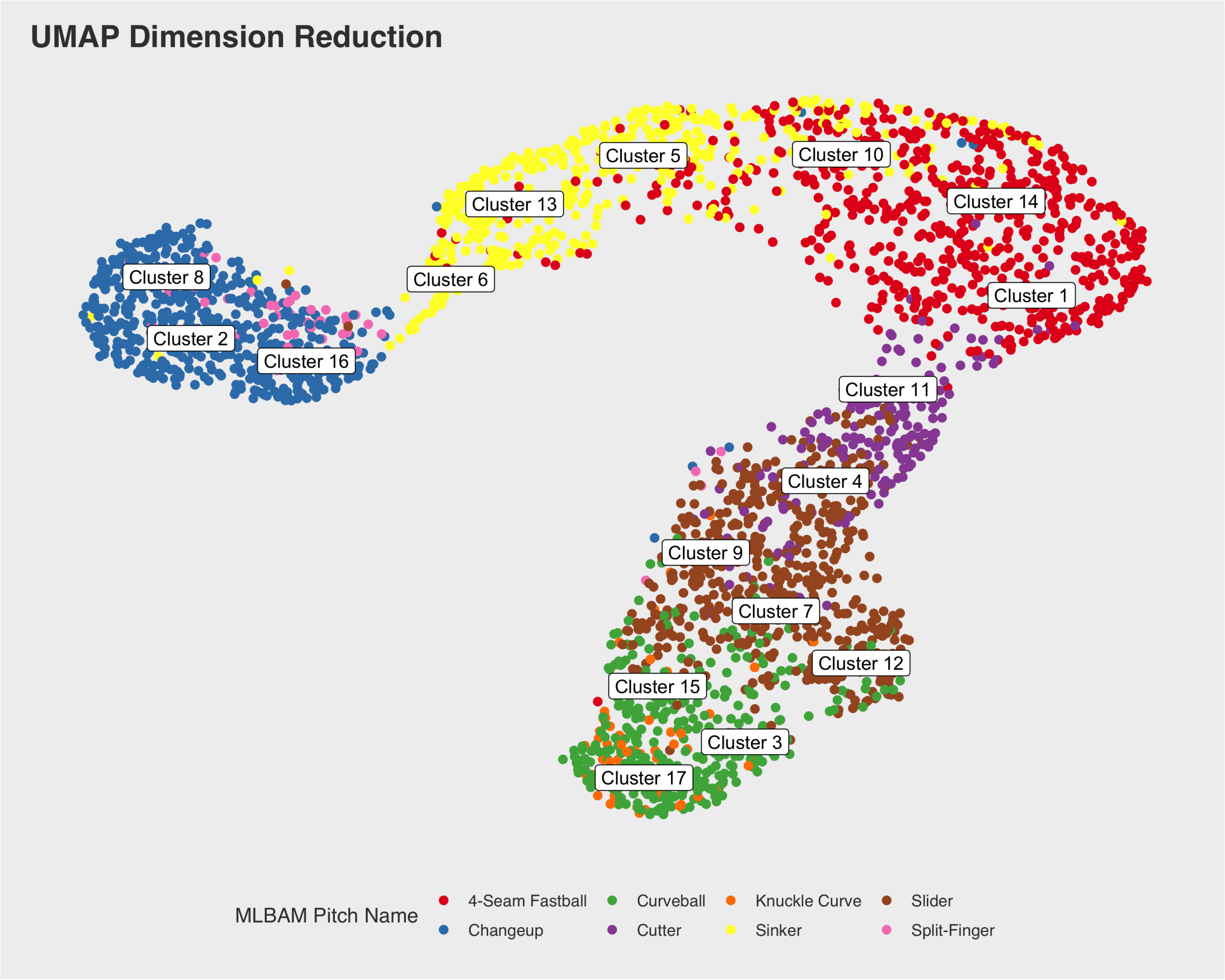


- When plotted by movement numbers, the clusters have very little overlap.
- The only true blend occurs between lower vertical break fastballs and changeups/splitters which would be expected as the main separator between those pitches is velocity.

UMAP

```r
custom_config <- umap.defaults
custom_config$n_neighbors = 200
custom_config$min_dist = .4

umap_data <- mlb_clusters |>
  select("avg_velo_ratio", "avg_vert", "avg_horz", "avg_eff")

umap_testing <- umap(umap_data, config = custom_config)

umap_plot_data <- mlb_clusters |>
  mutate(x = umap_testing$layout[,1],
         y = umap_testing$layout[,2])
```

UMAP Plot


UMAP Dimension Reduction
Cluster 5
Cluster 10
Cluster 13
Cluster 14
Cluster 8
Cluster 6
Cluster 1
Cluster 2
Cluster 16
Cluster 11
Cluster 4
Cluster 9
Cluster 7
Cluster 12
Cluster 15
Cluster 3
Cluster 17
MLBAM Pitch Name
4-Seam Fastball
Curveball
Knuckle Curve
Slider
Changeup
Cutter
Sinker
Split-Finger

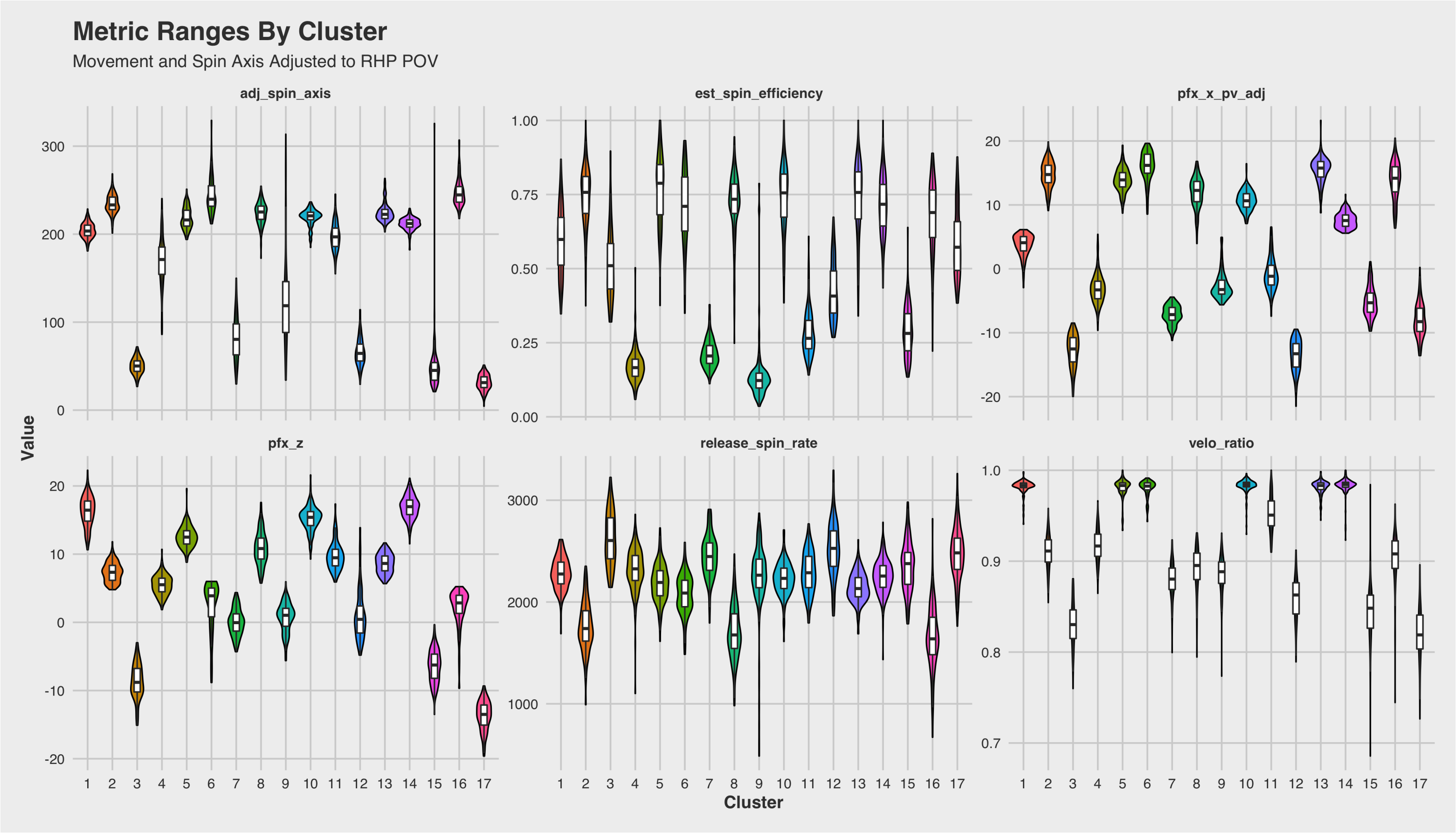
Testing

```r
# Map of pitchers and the clusters each of their pitches belong to
pitcher_pitch_map <- mlb_clusters |>
  select(pitcher, player_name, pitch_name, cluster)

# Combine the map with raw data and find averages for each pitch type
combined_data <- raw_data |>
  left_join(pitcher_pitch_map) |>
  mutate(adj_spin_axis = if_else(p_throws == "R",
                                 spin_axis, 360-spin_axis)) |>
  group_by(pitcher, pitch_name, cluster) |>
  summarize(velo_ratio = mean(velo_ratio, na.rm = TRUE),
            pfx_z = mean(pfx_z, na.rm = TRUE),
            pfx_x_pv_adj = mean(pfx_x_pv_adj, na.rm = TRUE),
            est_spin_efficiency =
              mean(est_spin_efficiency, na.rm = TRUE),
            release_spin_rate = mean(release_spin_rate, na.rm = TRUE),
            adj_spin_axis = mean(adj_spin_axis, na.rm = TRUE)) |>
  ungroup()
```

Transform Data

```
# Turn the combined data into a long df for plotting with a facet_wrap
filtered_data <- combined_data |>
  select(cluster, velo_ratio, pfx_z, pfx_x_pv_adj,
         est_spin_efficiency, release_spin_rate, adj_spin_axis)

long_data <- filtered_data |>
  pivot_longer(!cluster, names_to = "metric", values_to = "value")
```

Metric Ranges and Distributions



Metric Averages

Pitch Metric Averages By Cluster

Movement And Spin Axis Adjusted To RHP POV | Arranged By Descending Velo Ratio

CLUSTER	VELO RATIO	IND. VERT. BREAK (IN.)	ADJ. HORZ. BREAK (IN.)	EST. SPIN EFFICIENCY	SPIN RATE	ADJ. SPIN AXIS
14	98.4%	16.87	7.62	71.4%	2250	212
10	98.3%	15.18	10.71	73.8%	2228	219
1	98.2%	16.25	3.75	59.5%	2284	204
13	98.1%	8.62	15.45	73.4%	2144	225
5	98.1%	12.54	13.91	76.8%	2186	218
6	98.0%	2.55	16.03	70.8%	2073	244
11	95.2%	9.74	−0.77	28.3%	2286	198
4	91.8%	5.47	−3.13	17.0%	2321	169
2	91.0%	7.34	14.75	74.5%	1762	235
16	90.4%	2.10	13.75	67.6%	1647	248
8	89.1%	10.87	11.94	72.0%	1700	224
9	88.6%	0.68	−2.61	13.4%	2253	125
7	87.9%	0.01	−7.18	21.3%	2451	82
12	85.9%	1.07	−13.57	42.7%	2517	66
15	84.3%	−6.47	−5.12	28.7%	2346	48
3	83.0%	−8.64	−12.81	51.3%	2624	50
17	82.1%	−13.68	−8.06	58.8%	2474	32

Only first four metrics used in clustering model

- The fastball clusters clearly separate themselves by having an average velo ratio of > 95%

Manually Input Pitch Groups

```
# Manually created tibble based on pitch metrics
manually_set_cluster_names
```

```
## # A tibble: 17 × 3
##    cluster pitch_group        pitch_class
##    <fct>   <chr>              <chr>
##  1 1       Fastball           Fastball
##  2 2       Changeup-Splitter  Offspeed
##  3 3       Curveball          Breaking Ball
##  4 4       Slutter            Breaking Ball
##  5 5       Fastball           Fastball
##  6 6       Sinker             Fastball
##  7 7       Slider             Breaking Ball
##  8 8       Changeup-Splitter  Offspeed
##  9 9       Slider             Breaking Ball
## 10 10      Fastball           Fastball
## 11 11      Cutter             Fastball
## 12 12      Slider             Breaking Ball
## 13 13      Sinker             Fastball
## 14 14      Fastball           Fastball
## 15 15      Curveball          Breaking Ball
## 16 16      Changeup-Splitter  Offspeed
## 17 17      Curveball          Breaking Ball
```

Pitch Results Averages

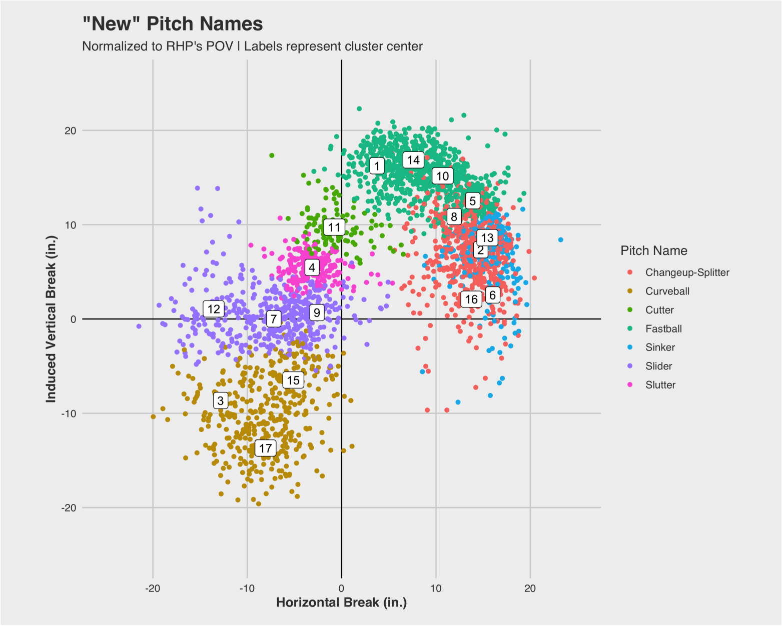
Pitch Results By Cluster

Results averaged across pitchers and are not weighted by # of pitches thrown

CLUSTER	# THROWN	RV/100	WOBA	XWOBA	WHIFF %	PUT AWAY RATE	HARD HIT %
1	78316	0.395	0.364	0.362	20.2%	17.4%	43.0%
2	31682	0.398	0.316	0.314	27.7%	16.7%	32.2%
3	19589	−0.088	0.274	0.263	32.2%	21.5%	31.6%
4	51645	−0.119	0.315	0.301	30.4%	20.3%	35.8%
5	46089	0.094	0.354	0.343	17.5%	16.9%	39.6%
6	22019	0.679	0.382	0.350	16.2%	16.0%	42.7%
7	33059	0.170	0.304	0.285	33.7%	20.6%	33.4%
8	23920	0.421	0.312	0.296	30.7%	17.0%	30.2%
9	40649	0.126	0.291	0.280	35.4%	21.7%	34.7%
10	60829	0.444	0.363	0.353	21.1%	17.5%	44.1%
11	29628	0.060	0.346	0.348	22.3%	18.9%	37.4%
12	31462	−0.246	0.272	0.257	34.8%	23.7%	28.4%
13	45757	0.499	0.371	0.354	14.8%	16.9%	40.8%
14	94045	0.415	0.374	0.366	21.6%	18.0%	45.3%
15	19221	0.423	0.298	0.284	30.8%	19.3%	37.3%
16	29246	0.622	0.304	0.285	31.8%	19.5%	35.0%
17	30673	0.692	0.300	0.284	29.7%	20.1%	37.3%

- Unsurprisingly, the only pitches that had a negative average run value per 100 pitches were breaking ball variants.
- When compared back to the movement plots we can The slutter (cluster 4), the sweeping slider (12), and sweeping curveball (3) all performed extremely well in the 2021 season.

"New" Pitch Groups



- While there is still a little overlap between pitch groups, pitches are much better contained based on their movement profiles

Summary

- Adding these extra layers of context can enhance the performed analysis by granting coaches, players, and analysts the ability to compare a pitch to smaller (more accurate) representations of a pitch.
- This model is not perfect as it is greatly influenced by my prior beliefs, but it lays the foundation for what pitch clustering can bring to analysis.