MA388 Sabermetrics: Lesson 12

Behaviors by Count - Umpire Tendencies

LTC Jim Pleuss

```r
library(tidyverse)
library(knitr)
```

Today we're going to take our first look at pitch data in terms of where a pitch crossed (or didn't cross) home plate and the umpire's associated call.

```r
load("./balls_strikes_count.Rdata")
```

Here's a glimpse of the data frame we'll use.

```r
umpires |>
  head() |>
  kable()
```

season	umpire	batter_hand	pitch_type	balls	strikes	px	pz	called_strike
2012	Mike Muchlinski	L	FF	3	1	-0.31	5.00	0
2012	Mike Muchlinski	R	SL	0	0	-0.18	2.46	1
2012	Mike Muchlinski	L	FF	2	1	0.78	3.62	0
2012	Mike Muchlinski	L	FA	1	1	-0.85	0.89	0
2012	Mike Muchlinski	L	SL	0	0	-0.55	3.85	0

season	umpire	batter_hand	pitch_type	balls	strikes	px	pz	called_strike
2012	Mike Muchlinski	L	FF	2	1	-0.76	2.97	1

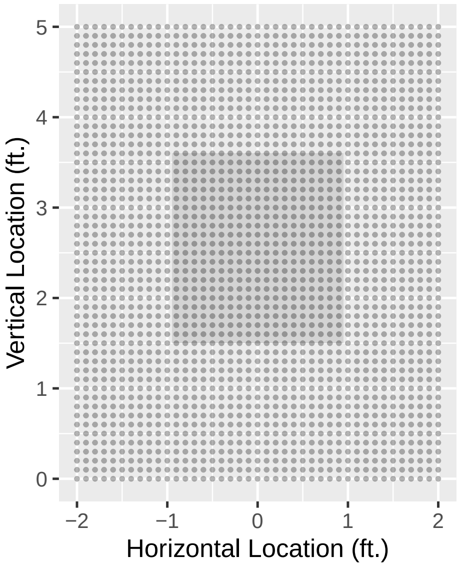
Do umpires adjust their strike zone based on the count? How might we go about gathering evidence?

We'll want to look at strike vs. ball calls throughout the strike zone and surrounding area, so let's build a data frame consisting of combinations of horizontal locations from -2 (two feet to the left of the middle of home plate) to +2 (two feet to the right of home plate) and vertical locations from the ground (value of 0) to five feet above the ground.

```
pred_area <- expand.grid(px = seq(-2, 2, by = 0.1),
                        pz = seq(0, 5, by = 0.1))
```

Let's create a base plot with the strike zone outlined and then add our grid on top.

```
k_zone_plot <- ggplot(pred_area, aes(x = px, y = pz)) +
  geom_rect(xmin = -0.947, xmax= 0.947, ymin = 1.5, ymax = 3.6,
            fill = "lightgray", alpha = 0.3) +
  coord_equal() +
  scale_x_continuous("Horizontal Location (ft.)", limits = c(-2, 2)) +
  scale_y_continuous("Vertical Location (ft.)", limits = c(0, 5))
k_zone_plot +
  geom_point(size = .5, alpha = 0.3)
```

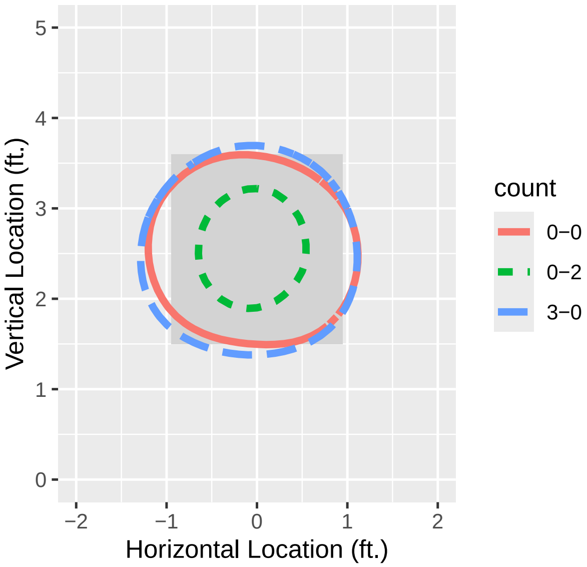


Digging into the Code!

What’s happening here? Annotate the code.

```
ump_count_fits <- umpires |>
  filter(batter_hand == "R",
         balls == 0 & strikes == 0 |
         balls == 3 & strikes == 0 |
         balls == 0 & strikes == 2) |>

  mutate(count = paste(balls, strikes, sep = "-")) %>%

  # split(_$count) |>
  split(pull(., count)) |>

  map(sample_n, 3000) %>%

  map(~loess(called_strike ~ px + pz,
             data = .,
             control = loess.control(surface = "direct"))) |>

  map(predict, newdata = pred_area) |>

  map(~data.frame(fit = as.numeric(.))) |>

  map_df(bind_cols, pred_area, .id = "count")
```

```
k_zone_plot %+%
  filter(ump_count_fits,
         fit < 0.6 & fit > 0.4) +
  geom_contour(aes(z = fit, color = count, linetype = count),
               binwidth = 0.1, lwd = 1.5)
```


count
0−0
0−2
3−0
5
4
3
2
1
0
−2
−1
0
1
2
Vertical Location (ft.)
Horizontal Location (ft.)

My Explanation: If we believe the count may affect an umpire's strike zone, we'll need to capture each unique count similarly to how we've previously captured the `STATE` when analyzing run expectancy. To do this we create a variable for count by combining the number of balls and strikes. We've narrowed our interest to right-handed batters and a neutral count, a hitter's count, and a pitcher's count. We then split the data on count and randomly draw 3,000 pitches that occurred with that count.

We then fit a LOESS model that attempts to fit a locally estimated surface to the average strike call value (0 vs. 1) at each of our data points (pitches).

Next, we compute a set of predictions for each point on our grid for each count (one model per split in our original data frame). We then convert the numeric matrices returned by predict into a numeric vector called `fit` before turning those vectors into one dimensional data frames. These data frames are then added to the `pred_area` data frame with the `bind_cols()` command, effectively adding the likelihood of a strike call to every `(px,pz)` combination for every count (0-0, 0-2, 3-0).