Ohtani vs. Ohtani

Amber Potter

2022-12-15

Introduction

Shohei Ohtani has taken the MLB by storm as an All Star pitcher and batter. Back in Japan, he played in the Nippon Professional Baseball's Pacific League for the Hokkaido Nippon-Ham Fighters. Now, he has established himself as a starting pitcher and DH and #17 for the Los Angeles Angels. In the last two years, Ohtani has been doing what no other player has done in over a century, building the foundation for a new type of two-way flexibility in players.

But what would happen if he pitched and batted against himself?

Data

The data for this project comes from Bill Petti's `baseballr` package. It includes Statcast data representing every pitch from the 2021 season. For this project, the data was filtered to only include regular season and post-season data. To create Ohtani's pitching dataset, I filtered for only observations where Ohtani was the pitcher and where he was facing a left-handed batter. To create Ohtani's batting dataset, I filtered for only observations where he was the batter, where he was facing a right-handed pitcher, and where the pitch thrown was one of the 5 pitches that Ohtani threw in 2021 (4-Seam Fastball, Slider, Split-Finger, Curveball, and Cutter). These filterings aimed to narrow down the data to just observations most similar to Ohtani facing Ohtani.

```
library(baseballr)
library(tidyverse)
library(knitr)

statcast_data_2021 <- readRDS("~/CMSAC-Baseball-Project/data/statcast_data_2021.rds")

pitcher_ohtani <- statcast_data_2021 %>%
  filter(stand == "L") %>%
  filter(pitcher == 660271) %>%
  filter(game_type !="E", game_type != "S")

batter_ohtani <- statcast_data_2021 %>%
  filter(p_throws == "R") %>%
  filter(batter == 660271) %>%
  filter(game_type !="E", game_type != "S") %>%
  filter(pitch_name %in% unique(pitcher_ohtani$pitch_name))
```

Extracting Useful Probabilities

Before coding the simulation, I created a series of tables containing data that would be used during different branches of the at-bat decision tree within the function defining the at-bat simulation. These tables include the probabilities

of different pitches being thrown on different counts, the probabilities of different pitches being in the strikezone, the probabilities given that it is in the zone that it is hit into play or counted as a strike, the probability of a strike being a foul ball, and the probabilities that a ball hit into play is a hit.

The tables containing these probabilities as well as the code for these tables are included in the Appendix.

The plots included in my presentation visualizing a few of these tables are also located in the Appendix.

Simulating an At-Bat

In order to examine the expected outcome of Ohtani vs. Ohtani, I decided to simulate 1000 mock at-bats of this impossible situation. Tod do so, I decided I would use each possible batting count as a possible state, with three strikes or four balls as absorbing states. I also knew that the event of a hit or a walk would also need to be at-bat terminating events.

To set up this simulation, each at bat begins with an 0-0 count. I use a while loop to include the possibility of any number of pitches being thrown during an at-bat. On each count, or for each state, I incorporated the conditional probabilities of different pitches being thrown given that count. For each count, I randomly sampled from Ohtani's 5 pitches based on these probabilities to determine the pitch thrown for that iteration of the while loop. By simulating Ohtani's pitcher tendencies in this way, I tried to account for his habits or patterns, like how he throws 4-Seam Fastball's more often when behind in the count.

Then, given the type of pitch thrown, I incorporated the conditional probability that it was thrown in the strikezone. To find these probabilities, I grouped Ohtani's pitcher data by pitch type, and calculated the proportion of times where that pitch was either hit or called a strike, and the proportion of times where that pitch was called a ball. Whether the pitch was in the strikezone was then chosen randomly based on these probabilities.

If the pitch was not determined to be in the strikezone, the number of balls was increased by one and the current state was updated to reflect this.

If the pitch was simulated to be in the strikezone based on Ohtani's pitcher data, I then simulated whether Ohtani the batter would put the ball in play or not. Again using conditional probabilities, given the pitch thrown, I found the probability of that pitch being hit into play or being counted as a strike. These probabilities were used to randomly decide if the pitch was hit in play or counted as a strike.

If the pitch was in the strikezone and was determined to 'count as a strike', this included swinging strikes, called strikes, and foul balls. If the pitch was labeled as 'counted as a strike', I then found the conditional probability, given the pitch type, of the strike being a foul ball or not using Ohtani's batter data, and used those probabilities to randomly decide if it would be labeled as a foul ball in this simulation. I had to do this because foul balls should only be counted as strikes when there are not already 2 strikes in the count. If there are less than two strikes and the current pitch is a strike (including foul balls), the number of strikes in the count is then increased by one and the state is updated. If there are 2 strikes and the current pitch is a strike that is not a foul ball, the number of strikes in the count is increased by one and the state is updated also. However, if there are 2 strikes and the strike is a foul ball, the count does not change, and that state simply repeats.

If the pitch was in the strikezone, but was determined to be hit in play, I then incorporated the probabilities of it being a successful hit or an out. To find these probabilities, I defined a successful hit as a single, double, triple, or homerun. An out was defined as any hit that was not successful (including fielder's choice and errors). Given the type of pitch and that the pitch outcome was a batted ball, I used Ohtani's batter data to find the proportion of successful hits. Then whether the at-bat resulted in a successful hit or an out was randomly decided according to these probabilities.

To end the at-bat, there are 4 potential outcomes. For starters, if the number of balls increases to 4, the outcome is a walk, which is a successful outcome for the batter and a failed outcome for the pitcher. If the number of strikes increases to 3, the outcome is a strikeout, which is a failed outcome for the batter and a successful outcome for the pitcher.

If the outcome is a hit, and it is a successful hit, it is a successful outcome for the batter and a failed outcome for the pitcher. If it is an unsuccessful hit (an out), it is a failed outcome for the batter and a successful outcome for the pitcher.

As long as none of these absorbing/terminating states/events have occurred yet in an at-bat, the simulated at-bat will continue with a new pitch.

```
at_bat_sim <- function()
{
  # start with no outs and no one on
  current_balls <- 0
  current_strikes <- 0
  current_state <- "00"

  batter_success <- FALSE

  # initialize empty vectors to keep track of what happens
  sim_states_vector <- c()

  while(current_balls < 4 & current_strikes < 3 & batter_success == FALSE)
  {
    # keep track of count sequences
    sim_states_vector <- append(sim_states_vector, current_state)
    if (current_state == "00"){
      stage_count <- "0-0"
      # Pitch Selection Probabilities by Count
      stage_pitch_names <- imputed_prob_pitchtype_by_count %>%
        filter(count == stage_count) %>%
        pull(pitch_name)

      stage_pitch_proportions <- imputed_prob_pitchtype_by_count %>%
        filter(count == stage_count) %>%
        pull(pitch_proportion)

      pitch_thrown <- sample(stage_pitch_names, size=1, prob =stage_pitch_proportions)
    }
    else if (current_state == "01"){
      stage_count <- "0-1"
      # Pitch Selection Probabilities by Count
      stage_pitch_names <- imputed_prob_pitchtype_by_count %>%
        filter(count == stage_count) %>%
        pull(pitch_name)

      stage_pitch_proportions <- imputed_prob_pitchtype_by_count %>%
        filter(count == stage_count) %>%
        pull(pitch_proportion)

      pitch_thrown <- sample(stage_pitch_names, size=1, prob =stage_pitch_proportions)
    }

    else if (current_state == "02"){
      stage_count <- "0-2"
      # Pitch Selection Probabilities by Count
      stage_pitch_names <- imputed_prob_pitchtype_by_count %>%
        filter(count == stage_count) %>%
        pull(pitch_name)

      stage_pitch_proportions <- imputed_prob_pitchtype_by_count %>%
        filter(count == stage_count) %>%
        pull(pitch_proportion)
```

```
  pitch_thrown <- sample(stage_pitch_names, size=1, prob =stage_pitch_proportions)
}

else if (current_state == "10"){
  stage_count <- "1-0"
  # Pitch Selection Probabilities by Count
  stage_pitch_names <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_name)

  stage_pitch_proportions <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_proportion)

  pitch_thrown <- sample(stage_pitch_names, size=1, prob =stage_pitch_proportions)
}

else if (current_state == "11"){
  stage_count <- "1-1"
  # Pitch Selection Probabilities by Count
  stage_pitch_names <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_name)

  stage_pitch_proportions <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_proportion)

  pitch_thrown <- sample(stage_pitch_names, size=1, prob =stage_pitch_proportions)
}

else if (current_state == "12"){
  stage_count <- "1-2"
  # Pitch Selection Probabilities by Count
  stage_pitch_names <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_name)

  stage_pitch_proportions <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_proportion)

  pitch_thrown <- sample(stage_pitch_names, size=1, prob =stage_pitch_proportions)
}

else if (current_state == "20"){
  stage_count <- "2-0"
  # Pitch Selection Probabilities by Count
  stage_pitch_names <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_name)

  stage_pitch_proportions <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_proportion)
```

```
  pitch_thrown <- sample(stage_pitch_names, size=1, prob =stage_pitch_proportions)
}

else if (current_state == "21"){
  stage_count <- "2-1"
  # Pitch Selection Probabilities by Count
  stage_pitch_names <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_name)

  stage_pitch_proportions <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_proportion)

  pitch_thrown <- sample(stage_pitch_names, size=1, prob =stage_pitch_proportions)
}

else if (current_state == "22"){
  stage_count <- "2-2"
  # Pitch Selection Probabilities by Count
  stage_pitch_names <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_name)

  stage_pitch_proportions <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_proportion)

  pitch_thrown <- sample(stage_pitch_names, size=1, prob =stage_pitch_proportions)
}

else if (current_state == "30"){
  stage_count <- "3-0"
  # Pitch Selection Probabilities by Count
  stage_pitch_names <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_name)

  stage_pitch_proportions <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_proportion)

  pitch_thrown <- sample(stage_pitch_names, size=1, prob =stage_pitch_proportions)
}

else if (current_state == "31"){
  stage_count <- "3-1"
  # Pitch Selection Probabilities by Count
  stage_pitch_names <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_name)

  stage_pitch_proportions <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_proportion)
```

```
  pitch_thrown <- sample(stage_pitch_names, size=1, prob =stage_pitch_proportions)
}

else if (current_state == "32"){
  stage_count <- "3-2"
  # Pitch Selection Probabilities by Count
  stage_pitch_names <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_name)

  stage_pitch_proportions <- imputed_prob_pitchtype_by_count %>%
    filter(count == stage_count) %>%
    pull(pitch_proportion)

  pitch_thrown <- sample(stage_pitch_names, size=1, prob =stage_pitch_proportions)
}



pitch_zone_probs <- prob_pitch_in_zone %>%
  filter(pitch_name == pitch_thrown) %>%
  pull(in_zone_prop)

pitch_in_zone <- sample(c(FALSE, TRUE),
                        size=1,
                        prob = pitch_zone_probs)

if (pitch_in_zone == TRUE){
  strike_or_batted_ball_probs_vector <- strike_or_batted_ball_probs %>%
    filter(pitch_name == pitch_thrown) %>%
    pull(probability_strike_or_batted_ball)

  strike_or_batted_ball <- sample(c("Strike", "Batted Ball"),
                                  size=1,
                                  prob = strike_or_batted_ball_probs_vector)

  if (strike_or_batted_ball == "Strike"){
    strike_or_foul_probs_vector <- strike_or_foul_probs %>%
      filter(pitch_name == pitch_thrown) %>%
      pull(prob_foul)

    strike_or_foul <- sample(c("Strike", "Foul Ball"),
                                  size=1,
                                  prob = strike_or_foul_probs_vector)

    if (!(current_strikes == 2 & strike_or_foul == "Foul Ball")){
      current_strikes <- current_strikes + 1
      current_state <- paste0(current_balls, current_strikes)
    }
    else{
      current_state <- paste0(current_balls, current_strikes)
    }
  }

  else if (strike_or_batted_ball == "Batted Ball"){
```

```
        prop_successful_hit_vector <- batter_success_probabilities %>%
          filter(pitch_name == pitch_thrown) %>%
          pull(prop_successful_hit)

        batter_hit_outcome <- sample(c("Out", "Hit"),
                                     size=1,
                                     prob = prop_successful_hit_vector)


        if (batter_hit_outcome == "Out"){
          batter_success = FALSE
        }
        else if (batter_hit_outcome == "Hit"){
          batter_success = TRUE
        }

      }

    }

    else if (pitch_in_zone == FALSE){
      current_balls <- current_balls + 1
      current_state <- paste0(current_balls, current_strikes)
    }

  }

  if (current_strikes == 3) {
    batter_success = FALSE
  }

  else if (current_balls == 4) {
    batter_success = TRUE
  }

    return(batter_success)
}
```

Results

After running 1000 iterations of simulated at-bats, I used the proportion of batter successes as a mock representation of Ohtani's On Base Percentage (OBP) against himself. Rerunning the code a few times, it seems to result in a mock OBP of around .350 to .400 each time. Choosing a randomization seed of 217, the simulated OBP is .384. For comparison, Ohtani had an overall OBP of .372 in 2021 and a .388 OBP against right-handed pitchers in 2021. The fact that this simulation produces results similar to Ohtani's actual batting stats is expected due to the included probabilities actually coming from his data against other pitchers, but it also shows that the simulation and incorporated decision tree are reasonably capable of determining the outcome of one of his at-bats.

```
set.seed(217)
# initializing table for all repeated simulations
simulation_data <- data.frame()
for (i in 1:1000)
{
  temp_at_bat_data <- at_bat_sim()
```

```
  simulation_data <- rbind(simulation_data, temp_at_bat_data)
}

library(knitr)

colnames(simulation_data) <- c("was_at_bat_batter_success")

simulation_data %>%
  select(was_at_bat_batter_success) %>%
  group_by(was_at_bat_batter_success) %>%
  count() %>%
  ungroup() %>%
  summarize(was_at_bat_batter_success,
            batter_success_prob =
              sub("^0.", "\\1.", sprintf("%.3f", round(n/sum(n), 3)))) %>%
  filter(was_at_bat_batter_success == TRUE) %>%
  pull(batter_success_prob) %>%
  kable(digits = 3, col.names = c("Expected On Base Percentage"))
```

Expected On Base Percentage
.384

Discussion

Limitations

Obviously this simulation represents a simplified imitation of Ohtani vs. Ohtani. We know that pitchers change their pitching based on the batter, but we don't know how Ohtani would pitch to Ohtani. We also know that batters approach batting differently and with different expectations against different pitchers, but we don't know how Ohtani bats against Ohtani. As they are the same person and could thus never actually face each other, there is no data to base their pitching/batting tendencies against each other off of. But since this is an attempt to simulate the impossible, the data used was intended to get close to predicting what might happen if they faced each other by using Ohtani pitching against players similar to Ohtani as a batter and Ohtani batting against pitches and pitchers similar to Ohtani as a pitcher as proxies for the real situation we are interested in. This is an inherently flawed method, but sufficient for our simplified mock at-bats.

Other limitations include additional simplifications made during the at-bat simulation. For example, there is no data for Ohtani throwing anything other than a 4-Seam Fastball on a 3-0 count. This is not to say he would never throw anything else, but rather that he hasn't on the few occasions that he has pitched with this count against left-handed batters. The effect of small sample sizes on probabilities appears in many parts of this simulation. Even though small sample size greatly affect the resulting probabilities, this simulation uses the observed probabilities as absolute and does not attempt to mitigate these potential inaccuracies.

This simulation also does not distinguish events like hit by pitches, caught foul balls, or other rare but potential events. It also doesn't take into account any environmental factors that might influence this imagined/theoretical match-up like the skill of the defense, the ballpark, or the weather. It also does not account for the human error of calling a strike. In this simulation, the probability of a pitch being a strike is not based on the physical location of the pitch but rather the call made about the pitch. By doing this, I attempt to capture the variance and error in calls made by umpires on called strikes.

Similarly, this simulation does not directly account for the possibility of swinging at pitch outside the strikezone. In this simulation, if a pitch is not a strike, it is always recorded as a ball. This admittedly inflates the probability of a walk. On the other hand, the outcomes of Ohtani swinging at balls are incorporated into the strikes/batted ball events given the pitch is 'in the strikezone'. Swinging at pitches outside the strikezone increases the probability

of swinging strikes and weak contact. Rather than handle these two flaws separately, this simulation simply leaves them to roughly balance each other out. It is not an ideal solution, but is a side-effect of the simplification of an at-bat.

Future Work

In the future, I would like to include data from 2022 in this simulation. Doing so would incorporated more recent data and would add in the Sinker, which Ohtani began throwing during the 2022 season. Including the 2022 season in addition to the 2021 year might make the simulation more accurate, but I would also be interested in how the results would compare between just running the simulation on 2021 data vs just running it on 2022 data.

I would also like to spend more time better accounting for many of the simplifications made such as how I accounted for swinging at pitches outside the strikezone, hit by pitches, etc.

Additionally, I think it would be interesting to adjust the code to simulate other stats like wOBA, weighting for different at-bat outcomes. Also, it might be interesting expand this simulation to the inning or game level where I could simulate the expected runs that a team of all Ohtani batters might earn against Ohtani as a pitcher.

Sources

- Petti B, Gilani S (2022). baseballr: Acquiring and Analyzing Baseball Data. https://billpetti.github.io/baseballr/, https://github.com/BillPetti/baseballr.
- Shohei Ohtani Stats, Baseball-Reference.com. (2000). Baseball-Reference.com. https://www.baseball-reference.com/players/o/ohtansh01.shtml
- MLB Stats. (2019). Shohei Ohtani Stats, Fantasy & News. MLB.com. https://www.mlb.com/player/shohei-ohtani-660271

Appendix

Table 1: Probabilities of Throwing Different Pitches

```
prob_pitchtype_by_count <- pitcher_ohtani %>%
  mutate(count = paste0(balls, "-", strikes)) %>%
  group_by(pitch_name, count) %>%
  count() %>%
  ungroup(pitch_name) %>%
  summarize(pitch_name, count, n, pitch_proportion = round(n/sum(n), 4), .groups = "drop")

# table with pitch probabilities for different counts
# with pitches not thrown on given count as 0
imputed_prob_pitchtype_by_count <- prob_pitchtype_by_count %>%
  rbind(c("1-1", "Curveball", 0, 0)) %>%
  rbind(c("2-0", "Curveball", 0, 0)) %>%
  rbind(c("2-1", "Curveball", 0, 0)) %>%
  rbind(c("3-0", "Cutter", 0, 0)) %>%
  rbind(c("3-0", "Slider", 0, 0)) %>%
  rbind(c("3-0", "Split-Finger", 0, 0)) %>%
  rbind(c("3-0", "Curveball", 0, 0)) %>%
  rbind(c("3-1", "Split-Finger", 0, 0)) %>%
  rbind(c("3-1", "Curveball", 0, 0)) %>%
  arrange(count, pitch_name)
```

```r
# table of pitch probabilities for different counts
imputed_prob_pitchtype_by_count %>%
  kable(digits = 4, col.names= c("Count", "Pitch Name", "Frequency","Probability"))
```

Count	Pitch Name	Frequency	Probability
0-0	4-Seam Fastball	153	0.5604
0-0	Curveball	32	0.1172
0-0	Cutter	18	0.0659
0-0	Slider	39	0.1429
0-0	Split-Finger	31	0.1136
0-1	4-Seam Fastball	49	0.4083
0-1	Curveball	3	0.025
0-1	Cutter	6	0.05
0-1	Slider	17	0.1417
0-1	Split-Finger	45	0.375
0-2	4-Seam Fastball	25	0.3521
0-2	Curveball	1	0.0141
0-2	Cutter	2	0.0282
0-2	Slider	3	0.0423
0-2	Split-Finger	40	0.5634
1-0	4-Seam Fastball	81	0.6639
1-0	Curveball	3	0.0246
1-0	Cutter	10	0.082
1-0	Slider	16	0.1311
1-0	Split-Finger	12	0.0984
1-1	4-Seam Fastball	40	0.4167
1-1	Curveball	0	0
1-1	Cutter	11	0.1146
1-1	Slider	29	0.3021
1-1	Split-Finger	16	0.1667
1-2	4-Seam Fastball	27	0.2967
1-2	Curveball	4	0.044
1-2	Cutter	1	0.011
1-2	Slider	8	0.0879
1-2	Split-Finger	51	0.5604
2-0	4-Seam Fastball	39	0.8298
2-0	Curveball	0	0
2-0	Cutter	3	0.0638
2-0	Slider	4	0.0851
2-0	Split-Finger	1	0.0213
2-1	4-Seam Fastball	26	0.5652
2-1	Curveball	0	0
2-1	Cutter	7	0.1522
2-1	Slider	12	0.2609
2-1	Split-Finger	1	0.0217
2-2	4-Seam Fastball	13	0.2031
2-2	Curveball	3	0.0469
2-2	Cutter	1	0.0156
2-2	Slider	15	0.2344
2-2	Split-Finger	32	0.5
3-0	4-Seam Fastball	22	1
3-0	Curveball	0	0
3-0	Cutter	0	0
3-0	Slider	0	0
3-0	Split-Finger	0	0

Count	Pitch Name	Frequency	Probability
3-1	4-Seam Fastball	24	0.8
3-1	Curveball	0	0
3-1	Cutter	4	0.1333
3-1	Slider	2	0.0667
3-1	Split-Finger	0	0
3-2	4-Seam Fastball	9	0.2195
3-2	Curveball	1	0.0244
3-2	Cutter	2	0.0488
3-2	Slider	21	0.5122
3-2	Split-Finger	8	0.1951

Plot 1: Visualizing Pitching Tendencies

```
prob_pitchtype_by_count %>%
  ggplot(aes(x = count, y = n, fill = pitch_name)) +
  geom_bar(stat = "identity", position = "fill") +
  scale_fill_viridis_d(option = "D") +
  theme_minimal() +
  labs(x = "Count (Balls-Strikes)",
       y = "Percent",
       title = "Proportions of Pitch Types Thrown on Different Counts",
       fill = "Pitch Type")
```

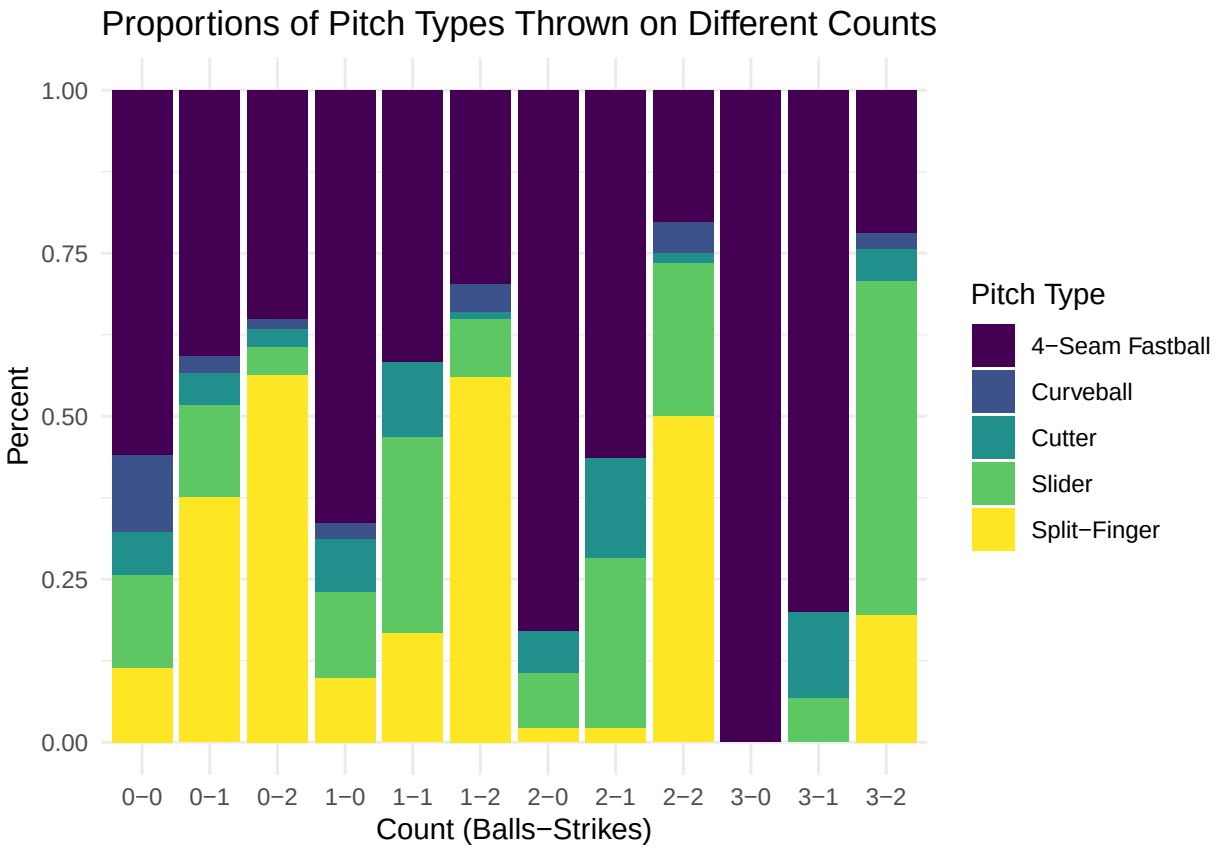

Table 2: Probabilities of Different Pitches Being in the Strikezone

```
# probability of pitch in zone given pitch name
prob_pitch_in_zone <- pitcher_ohtani %>%
  mutate(in_strikezone = case_when(type == "S" | type == "X" ~ TRUE,
                                   type == "B" ~ FALSE)) %>%
  group_by(pitch_name, in_strikezone) %>%
  summarize(in_zone_counts = n()) %>%
  summarize(pitch_name,
            in_strikezone,
            in_zone_counts,
            in_zone_prop = in_zone_counts/sum(in_zone_counts))


prob_pitch_in_zone %>%
  kable(col.names= c("Pitch Name", "In the Strikezone?", "Frequency","Probability"), digits = 4)
```

Pitch Name	In the Strikezone?	Frequency	Probability
4-Seam Fastball	FALSE	184	0.3622
4-Seam Fastball	TRUE	324	0.6378
Curveball	FALSE	25	0.5319
Curveball	TRUE	22	0.4681
Cutter	FALSE	27	0.4154
Cutter	TRUE	38	0.5846
Slider	FALSE	44	0.2651
Slider	TRUE	122	0.7349
Split-Finger	FALSE	97	0.4093
Split-Finger	TRUE	140	0.5907

Plot 2: Visualizing the Probabilities of Different Pitches Thrown in the Zone

```
prob_pitch_in_zone %>%
  ggplot(aes(x = pitch_name, y = in_zone_counts, fill = in_strikezone)) +
  geom_bar(stat = "identity", position = "fill") +
  scale_fill_viridis_d(option = "D") +
  theme_minimal() +
  labs(x = "Pitch Types",
       y = "Percent",
       title = "Proportions of Pitch Types Thrown in the Strikezone",
       fill = "In the zone?")
```

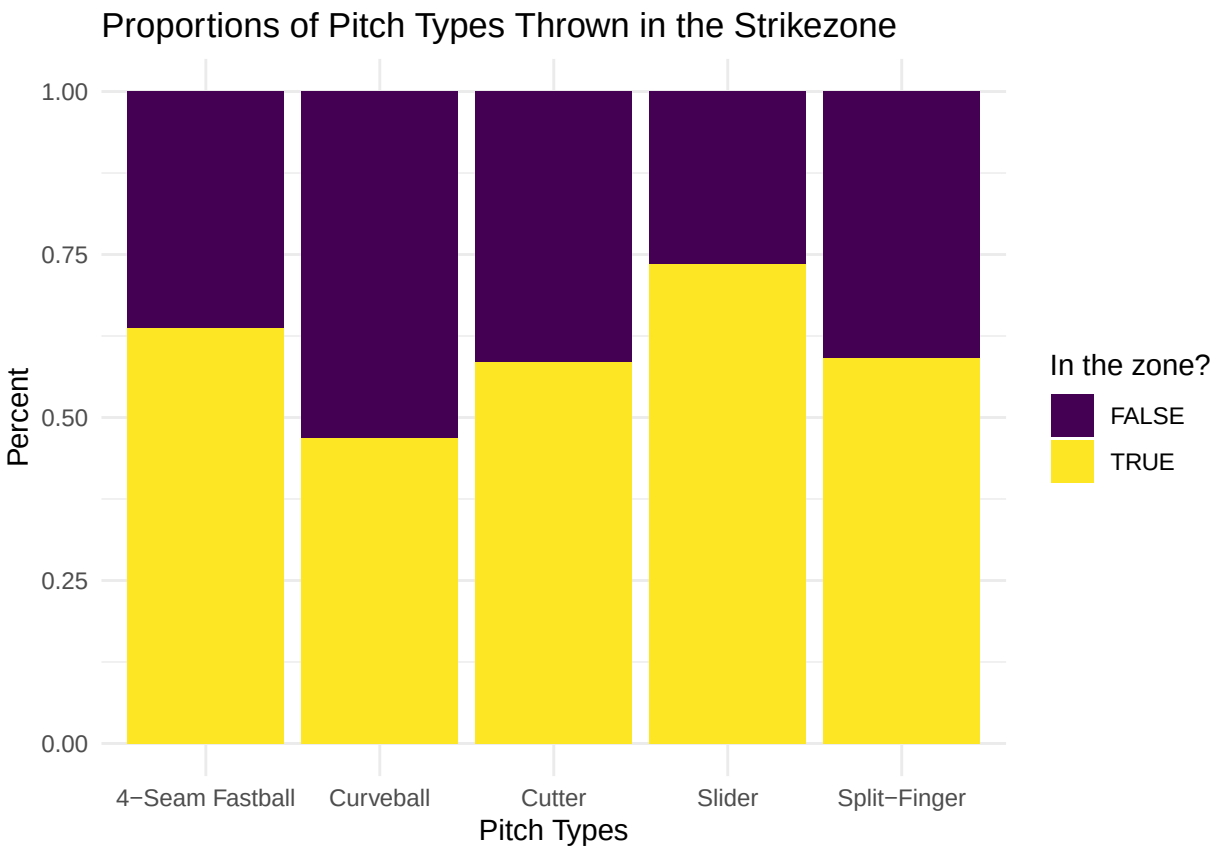

Table 3: Probabilities of Pitch in the Strikezone being a Strike or a Batted Ball

```
strike_or_batted_ball_probs <- batter_ohtani %>%
        filter(type == "S" | type == "X") %>%
        group_by(pitch_name, type) %>%
        summarize(strike_battedball_count = n()) %>%
        summarize(pitch_name, type,
                  strike_battedball_count,
                  probability_strike_or_batted_ball =
                    strike_battedball_count/sum(strike_battedball_count))


strike_or_batted_ball_probs %>%
  kable(col.names=
          c("Pitch Name", "Strike (S) or Batted Ball (X)", "Frequency","Probability"),
        digits = 4)
```

Pitch Name	Strike (S) or Batted Ball (X)	Frequency	Probability
4-Seam Fastball	S	278	0.8201
4-Seam Fastball	X	61	0.1799
Curveball	S	80	0.7843
Curveball	X	22	0.2157
Cutter	S	61	0.7821
Cutter	X	17	0.2179
Slider	S	92	0.7931

Pitch Name	Strike (S) or Batted Ball (X)	Frequency	Probability
Slider	X	24	0.2069
Split-Finger	S	27	0.7714
Split-Finger	X	8	0.2286

Table 4: Probabilities of a Strike being a Foul Ball

```
strike_or_foul_probs <- batter_ohtani %>%
          filter(type == "S") %>%
          mutate(is_foul = case_when(description == "foul" ~ TRUE,
                                     TRUE ~ FALSE)) %>%
          group_by(pitch_name, is_foul) %>%
          summarize(foul_count = n()) %>%
          summarize(pitch_name, is_foul, foul_count, prob_foul = foul_count/sum(foul_count))

strike_or_foul_probs %>%
  kable(col.names= c("Pitch Name", "Is a Foul Ball?", "Frequency","Probability"), digits = 4)
```

Pitch Name	Is a Foul Ball?	Frequency	Probability
4-Seam Fastball	FALSE	168	0.6043
4-Seam Fastball	TRUE	110	0.3957
Curveball	FALSE	57	0.7125
Curveball	TRUE	23	0.2875
Cutter	FALSE	39	0.6393
Cutter	TRUE	22	0.3607
Slider	FALSE	68	0.7391
Slider	TRUE	24	0.2609
Split-Finger	FALSE	22	0.8148
Split-Finger	TRUE	5	0.1852

Table 5: Probabilities of Ball Hit in Play being a Successful Hit

```
# probabilities of Ohtani's batting success given a hit
# against different pitch types
batter_success_probabilities <- batter_ohtani %>%
  filter(type == "X") %>%
  mutate(success = case_when(type == "X" &
                               events %in%
                               c("single", "double", "triple", "homerun") ~ TRUE,
                             TRUE ~ FALSE)) %>%
  group_by(pitch_name, success) %>%
  count() %>%
  ungroup(success) %>%
  summarize(pitch_name, success, n, proportion_success = n/sum(n),
            .groups = "drop")

batter_success_probabilities %>%
  kable(col.names= c("Pitch Name", "Successful Hit?", "Frequency","Probability"), digits = 4)
```

Pitch Name	Successful Hit?	Frequency	Probability
4-Seam Fastball	FALSE	42	0.6885
4-Seam Fastball	TRUE	19	0.3115
Curveball	FALSE	15	0.6818
Curveball	TRUE	7	0.3182
Cutter	FALSE	14	0.8235
Cutter	TRUE	3	0.1765
Slider	FALSE	19	0.7917
Slider	TRUE	5	0.2083
Split-Finger	FALSE	6	0.7500
Split-Finger	TRUE	2	0.2500

Plot 3: Conditional Probabilities of a Successful Hit Given the Pitch and the Ball in Play

```
batter_success_probabilities %>%
  ggplot(aes(x = pitch_name, y = n, fill = success)) +
  geom_bar(stat = "identity", position = "fill") +
  scale_fill_viridis_d(option = "D") +
  theme_minimal() +
  labs(x = "Pitch Types",
       y = "Percent",
       title = "Probability of Successful Batted Ball Outcomes",
       fill = "Base Hit?") +
  theme(legend.position = "bottom")
```