An Analysis of the Role of Pitch Types in Major League Baseball

Alexander S.

26/03/2019

Introduction

The dataset that will be analyzed in this report is the MLB Statcast dataset. This dataset is comprised of observations on 90 variables for every pitch thrown in the MLB regular season (and playoffs) since 2008. Initially a system created by Sportvision, called pitchfx, was used to record the data, however that system was superseded by Statcast's Trackman system at the beginning of the 2016 regular season. Both the Pitchfx and Trackman systems work in a relatively equivalant way, utilizing triangulation and radar to generate accurate measurements regarding the trajectory and position of a given pitch. As Statcast is the newer product, it provides quite a few extra measurements that Pitchfx does not, which are also a bit more useful with respect to data analysis.

The Pitchfx and Statcast datasets are both publicly available. Pitchfx data is located at the MLB GameDay website, and is stored in XML files. A good R package for retrieving pitchfx data is Carson Sievert's PitchRx package.He also wrote a nice article that goes into detail about how to use the dplyr and DBI packages to store the data in a sqlite database, as well as to make queries on that database. This article will be included in the references portion of this report. Statcast's data is available at the Baseball Savant website. It includes all of the pitchfx data as well as its own. From this page one can customize a query and have the website return a csv file containing all of the observations matching the user's specifications. Grabbing the data in this method is not quite ideal, as customization takes forever in this format, and often one wants to generate large files. Bill Petti wrote a nice R package titled baseballr which can be used to scrape the Statcast data and load it into R. Unfortunately there is a hard limit enforced by Statcast on how many observations can be scraped in a single request (both through the website and the baseballr package). For example, if all pitches from the year 2018 are requested, only the first 40,000 will be delivered. This means that anyone who interested in scarping a large chunk of the data is forced to break up their request in multiple calls, which is quite frustrating.

Despite the above mentioned setbacks, it is the Statcast data that will be analyzed in this report, as it provides a stronger set of variables, and provides all data within a single table (rather than multiple tables that must be joined within the database first before importing to R). The analysis will focus on investigating the different pitch types, the features that make them unique, and their role in the current MLB meta. The target population is of course all future pitches, as these are what we hope to better understand after this analysis. The study population will be every pitch thrown by a major league pitcher during the 2018 MLB regular season for which Statcast collected and reported measurements.

Obtaining the Data

As mentioned in the preceeding section, the data is available publicly and can be retrived in chunks of about 40,000 observations at a time. Each season has somewhere between 650 and 800 thousand observations, so loading up multiple seasons in a single can easily overwhelm working memory. Although the dataset to be explored in this report will fit in memory, it is still preferable to have all the data tucked away in a database somewhere for easy access, as an analysis of multiple seasons requires the use the databases. Therefore, this first section will be devoted to illustrating how to grab this data and load it into a database.

To begin, install the baseballr package using the following command, and load it into the session.

```
#library(devtools)
devtools::install_github("BillPetti/baseballr")
library(baseballr)
```

This package currently provides a number of scraping functions, although it is mentioned in the github documentation that all they will eventually be amalgamated into a single scrape function. Since interest is mainly on pitchers, we choose to test the two functions `scrape_statcast_savant` and `scrape_statcast_savant_pitcher_all`. To see if there is any difference in the tables returned by each function, the arbitrary date `2018-04-06` is chosen and the functions are used to call the data for this day.

```r
#Call the Statcast data from April 6th 2018 and compare resulting tables
scrape_statcast_savant("2018-04-06", "2018-04-06", type = "pitcher")%>%
  select(pitch_type:pitcher)%>%
  head()%>%
  kable()
```

pitch_type	game_date	release_speed	release_pos_x	release_pos_z	player_name	batter	pitcher
FF	2018-04-06	91.7	-2.5964	6.1537	Tyler Austin	592122	542960
FF	2018-04-06	92.3	-2.6948	6.0787	Tyler Austin	592122	542960
FF	2018-04-06	92.6	-2.8473	6.1860	Tyler Austin	592122	542960
SL	2018-04-06	85.4	-3.3153	5.9143	Ronald Torreyes	591720	542960
FF	2018-04-06	92.4	-2.7254	6.1289	Ronald Torreyes	591720	542960
SL	2018-04-06	84.4	-3.1374	6.2782	Ronald Torreyes	591720	542960

```r
scrape_statcast_savant_pitcher_all("2018-04-06", "2018-04-06")%>%
  select(pitch_type:pitcher)%>%
  head()%>%
  kable()
```

pitch_type	game_date	release_speed	release_pos_x	release_pos_z	player_name	batter	pitcher
FF	2018-04-06	91.7	-2.5964	6.1537	Brad Brach	592122	542960
FF	2018-04-06	92.3	-2.6948	6.0787	Brad Brach	592122	542960
FF	2018-04-06	92.6	-2.8473	6.1860	Brad Brach	592122	542960
SL	2018-04-06	85.4	-3.3153	5.9143	Brad Brach	591720	542960
FF	2018-04-06	92.4	-2.7254	6.1289	Brad Brach	591720	542960
SL	2018-04-06	84.4	-3.1374	6.2782	Brad Brach	591720	542960

Both functions return the same table, with one notable exception. A call to the Statcast data returns a single column titled `player name`. The `scrape_statcast_savant` function fills this column with the batters name (even though the call was made referencing pitcher), while the `scrape_statcast_savant_pitcher_all` function fills this column with the pitcher's name. It strikes as slightly odd that Statcast would not simply choose to return both of these names by default. Due to the fact many of the variables related to the actual pitch in question, the data will be collected using the `scrape_statcast_savant_pitcher_all` function. It is worth noting that there is also a column labelled `batter` which reports a number called the `batter id` (more information regarding this number to follow. I believe a complete list is unpublish, but I believe partial lists exist. See Bill Petti's stuff).

Now that a function capable of retrieving the Statcast data has been identified, gathering the data is only a matter of determining the date intervals on which to call the function. `scrape_statcast_savant_pitcher_all` cannot be simply iterated over every date of the 2018 season because the function returns an error if it is called on an interval of dates in which no games were played. Likewise it cannot be called on the entire span due to the observation number limit per query imposed by Statcast. After a little bit of fiddling around, a gap of about 9 days was found to be optimal. To create a new sqlite database, simply call the dplyr function `src_sqlite` with the path you want your database to be located at and set the `create` parameter to TRUE.

```r
#Create a new sqlite database
db <- src_sqlite("statcastDB.sqlite3", create = TRUE)

#Grab data from first day of the 2018 regular seaons
```

```
initial_data <- scrape_statcast_savant_pitcher_all("2018-03-29", "2018-03-29")

#Write the data to the database you created
dbWriteTable(db$con, "statcast", initial_data)
```

In the above code snippet, the database has been generated and data from the first day of the 2018 regular season, March 29th, has been scraped. The `RSQLite` package's `dbWriteTable` function is then used to write this data to a new table in the database titled "statcast".
The last day of the regular season was October 1st 2018, so the next step is to generate date intervals from March 30th to October 1st on which to call `scrape_statcast_savant_pitcher_all`. A simple function, titled `writeStatcastToDB` will also be written so as to allow this process to be easily looped. Any and all user-defined functions will be provided in the accompanying R code.

```
#Generate the dates to be used in the baseballr pitcher_all function.

#Set the first interval to be 9 days, including the inital date
dateInitial <- as.Date("2018-03-30")
endFirstDateInterval <- dateInitial + 8
#The last day of the season
dateFinal <- as.Date("2018-10-01")

#Generate the end points for each date interval. Append the last day to finish_dates
#because the final interval is less than 9 days
begin_dates <- seq(dateInitial, dateFinal, "9 days")
finish_dates <- seq(endFirstDateInterval, dateFinal, "9 days") %>%
  append(dateFinal)

mapply(writeStatcastToDB,
       date_init = begin_dates,
       date_fin = finish_dates,
       table = "statcast")
```

Variate Cleanup and Overview

Now that the data set has been successfully collected and stored, cleaning can begin. This data set is extremely raw, in the sense that the observations reported do not undergo any sort of valdiation or cross-examination before being uploaded. Consequently, the data set is sparsely riddled with partially erroneous observations. These will purged during the cleaning phase.
This data set also suffers from attempts to maintain compatiblity with the earlier Pitchfx system. Such measures have lead it to contain columns that are completely filled with `NA` and/or labelled as `deprecated`. Such columns are essentially useless in the present context. The next step, then, is to remove these columns. However, we do not want to overwrite the original table, as we may still need to reference it in the future. To remove the desired columns from the "statcast" table and write the resulting data to a new table inside the database without actually loading the table into the R session, I use a function proposed on Stack Overflow by user Moody_Mudskipper, which is defined below.

```
#Writes a new table in the database "data", with the name "name".
create <- function(data,name){
  DBI::dbSendQuery(data$src$con,
                   paste("CREATE TABLE", name,"AS", dbplyr::sql_render(data)))
  dplyr::tbl(data$src$con,name)
}
```

Removal of the redundant and deprecated columns can then be accomplished with the following code.

```
#Use this code to creeate a connection to the database
db <-  src_sqlite("statcastDB.sqlite3", create = FALSE)

#Choose columns to remove. These columns are know to be empty or deprecated
remove <- c("spin_dir",
            "spin_rate_deprecated",
            "break_angle_deprecated",
            "break_length_deprecated",
            "game_type",
            "tfs_deprecated",
            "tfs_zulu_deprecated",
            "umpire",
            "sv_id")

#Remove the columns and write data as a new table in the database
db %>%
  tbl("statcast")%>%
  select(-remove)%>%
  create("cleancast")
```

This requires no loading of the data because the `dbplyr` functions are essentially SQL queries masked by familiar `dplyr` function names, with the `tbl` function merely generating a reference to the "statcast" table in our database rather than loading it into the session. A table can be loaded into the session as a tibble at any point by calling the function `collect()`.

Further discussion on why these variables are deprecrated or unpopulated can be found here.

Now that the deprecated columns have been removed, we can beginning reviewing the variables that are provided by this dataset. There are 81 variables still present in the statcast data, however not all of them will be relevant to the present analysis. Here I will go over some of the more important ones, but a description of every single variable (including the ones that have already been removed) can be found on the Baseball Savant website.

First are the categorical variates. These variates give some context to each pitch in the dataset.

- `pitch_type` The type of pitch thrown, as reported by Statcast. The value is reported is one of fourteen, 2-letter strings which represent the fourteen pitches recognized by MLBAM (discussed in the next section). The classification used to be done by a neural network, but I am unsure if that is still the case.
- `player_name` Depending on how the data has been accessed, this column contains the name of the person who threw the pitch or the name of the batter who the pitch was thrown to.
- `batter` The MLBAM id number of the batter to whom the pitch was thrown
- `pitcher` The MLBAM id number of the pitcher who threw the pitch
- `events` This column indicates the result of the entire plate appearance of a batter. For all pitches that do not determine the end of a plate appearance (an out, a hit, or a walk is issued to the player at the plate, or a player on base is tagged as the 3rd out) this column takes the value `NA`.
- `description` The direct result of the pitch in question. It has the following unique values.

description
ball
blocked_ball
called_strike
foul
foul_bunt
foul_pitchout
foul_tip
hit_by_pitch
hit_into_play
hit_into_play_no_out
hit_into_play_score
missed_bunt
pitchout
swinging_pitchout
swinging_strike
swinging_strike_blocked

where the prefix "blocked" means the pitch was in the dirt or the catcher and pitcher got cross up, but the catcher was still able to block the pitch from getting behind him.
A pitchout is a ball thrown to the catcher with the intention that the catch will then fire the ball to one of the bases as quickly as possible to try and get a surprise tag on a baserunner.

- `des` A sentence description of the plate appearance. Typically specifies the batter's name, as well as other events that occured on the field as a result of the plate appearance, such as the base movement of a runner, a throwing error by the defense, etc.
- `p_throws` Determines the hand with which the pitcher throws the ball. Reported as "L" or "R".
- `stand` The side of the plate that the batter is standing on, from the pitcher's persepective.
- `on_3b, on_2b, on_1b` These columns keep track of runners on bases, recording the MLBAM id of the runner which currently occupies that base, and records the value `NA` when the base is unoccupied.
- `game_pk` A number that uniquely identifies a game. It increments up as the season goes on. Games that are played on the same day are assigned numbers in alphabetical order based on the visiting team's official name. Its starting value for the 2018 season is 529406, corresponding to Boston Red Sox @ Tampa Bay Rays. There are

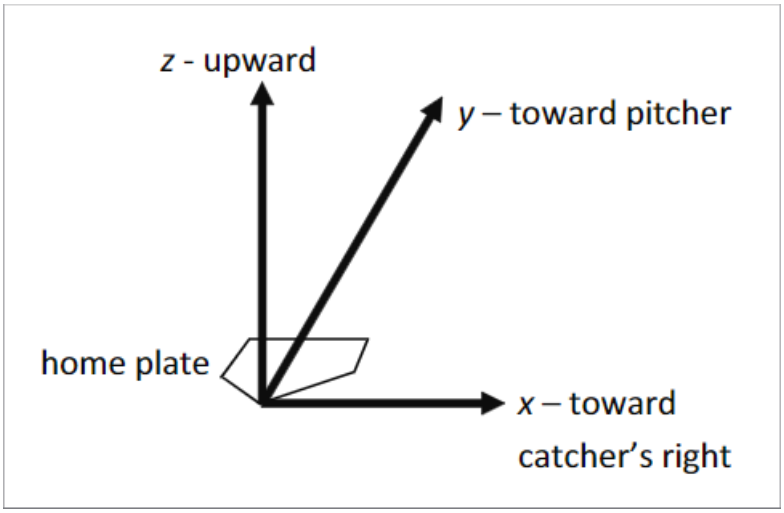
```
db%>%
  tbl("cleancast")%>%
  distinct(game_pk)%>%
  count()%>%
  kable()
```

n
2431

2431 distinct values, which can be verified as the exact number of games played by teams in the 2018 season by double checking on the Baseball-Reference website.

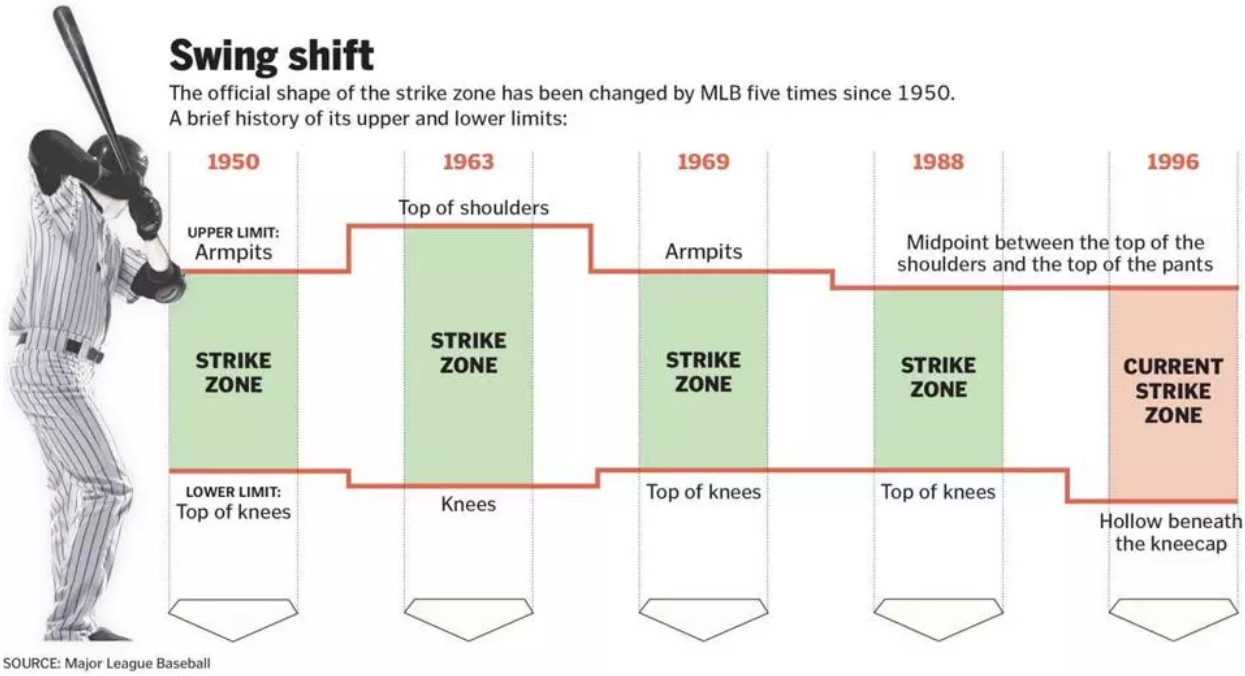
Before getting into the numerical variables, we should mention that this dataset imposes a coordinate system

on the field of play. It is given in the picture below.



The very back of home plate is the origin. Measurements on all three axes are given in feet. Home plate is 17 inches from point to front, which puts the face of the strike zone at $y = \frac{17}{12}$. the pitcher's mound is centered at $y = 60.5$. This in particular means that pitches travel in the negative y direction, and due to the pull of gravity, they travel in the negative z direction as well.

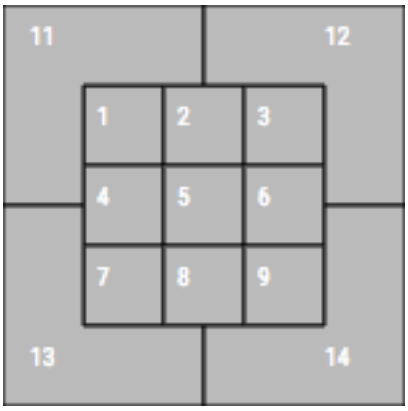
The strike zone itself sits directly above homeplate, and its outer boundaries are defined by projecting the boundaries of homeplate upward. However, its upper and lower boundaries have actually changed a quite a few times since 1950, as the following graphic demonstrates.



The current rules are a little bewildering. It seems that the current bounds enforced on the strike zone are harder to indentify than earlier versions, especially the top boundary, given that it does not relate to a definitive "landmark" that can be glanced at to quickly and easily establish the strike zone.

The following variables measure certain properties of a pitch, count occurences of certain events, or represent a label or coordinates for a location. All measurements are due to the Trackman system unless otherwise stated.

- `release_speed` The speed of the ball, in miles per hour, as it leaves the pitcher's hand
- `release_spin_rate` Spin rate of the ball, measured in revolutions per minute, as it leaves the pitcher's hand
- `balls, strikes` The number of balls and strikes that have been thrown in the plate appearance before the current pitch. All plate appeareances start with 0 balls and 0 strikes. The highest possible number of balls is 3, with a 4th ball ending the plate appearance in a walk. The highest number of strikes is 2, with the 3rd strike corresponding to an out.
- `zone` The zone of the strikezone in which the pitch was in as it crossed the front of homeplate. A figure of the zones and their locations is provided below.



- `hc_x, hc_y` These two variates give the x and y locations of a ball that is hit into the field of play.
- `hit_distance_sc` This gives the distance that a ball hit into the field of play travelled before it hit the ground or contacted a defender (whichever came first), rounded to the nearest foot. The exact method used to calculate this value has not been made public.
- `launch_speed/angle` These variates report the speed of a ball as it comes off the bat when it is hit into the field of play, and the vertical launch angle of that ball (the angle between its trajectory off the bat and the x-y plane) in degrees, respectively.
- `launch_speed_angle` This variate assigns a number between 1 and 6 to each pitch, based on its values of `launch_speed` and `launch_angle`. These groups are also given names, which corresponds to the type/quality of the contact created by the batter. The legend is as follows:

1) Weak
2) Topped
3) Hit Under
4) Flare/Burner
5) Solid Contact
6) Barrel

the full list of variates on the Statcast website linked above also contains a graphic for these groups. It shows a semicircle with a radius that corresponds to `launch_speed` and an angle that corresponds to `launch_angle`, and it shades in the different regions that corresponds to the different groups defined above.

- `at_bat_number` The plate appearance number of the game. I.e. this variate keeps track of which plate appearance of a game that a given pitch appeared in.

- `pitch_number` The pitch number of a plate appearance. For example, a value of 1 would correspond to the first pitch thrown in a given plate appearance, a value of 2 would represent the second pitch, and so on.

- `release_pos_x/y/z` These three columns give the x,y, and z position respectively, of a given pitch the moment it was released from the pitcher's hand.

- `vx0, vy0, vz0, ax, ay, az` The velocity and acceleration components of the model fit to each pitch. Velocity is recorded in feet per second and acceleration is record in feet per second squared. Together with the release position columns, these columns determine the parameters of a space curve which approximates the trajectory of the pitch as measured by the Trackman system. The fit of the model is given by

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + t \begin{bmatrix} vx0 \\ vy0 \\ vz0 \end{bmatrix} + \frac{1}{2}t^2 \begin{bmatrix} ax \\ ay \\ az \end{bmatrix}$$

  This paper discusses how the model is fit. Although the paper is specific to the pitchfx system and is quite old, the fundamental model has not changed. The only difference is that the instruments taking measurements on the pitch trajectory have improved.
  Notice above I have put values x_0, y_0 and z_0 into the equation, rather than the release_pos columns given by the dataset. This is because the velocity components are reported at the specific point $y = 50$. This is another artifact of the Pitchfx system. This in fact means that the dataset does not even give sufficient information to graph trajectories directly out fo the gate. Instead, one must either calculate x_0, y_0, and z_0 or the velocity vector at release. Either set can be obtained using kinematic equations from physics. We choose to generate the velocities at the release point because this leads to a more natural form of the trajectory, with $t = 0$ corresponding to the time at release. This is done in the next section.

Before continuing to the next section of anaylsis, we first not that the variates `game_pk`, `at_bat_number` and `pitch_number` determine a primary key for this dataset, which can be verified with the following code.

```
db%>%
  tbl("cleancast")%>%
  select(game_pk, at_bat_number, pitch_number)%>%
  collect()%>%
  count(game_pk, at_bat_number, pitch_number)%>%
  filter(n>1)%>%
  nrow() == 0
```

```
## [1] TRUE
```

Although it is always good to know such a key exists, the primary key is not likely to be all that useful for the analysis at hand, because interest is in the behaviour of a family of pitches rather than any given specific pitch. Further, these three variates do not provided any real information about the actual pitch thrown, so there is no apriori reason one might want to grab a pitch corresponding to some specific values of these variables.

Identifying and Rectifying Measurement Errors

In this section, we will flush out some of the measurement errors in this dataset. We will save fixing the errors until the very end of the section, at which point we will rewrite the ammended table into the database.

First lets find out how many observations we are working with for the 2018 season. This value can again be calculated without bringing the table into working memory.

```
db%>%
  tbl("cleancast")%>%
tally()
```

```
## # Source:   lazy query [?? x 1]
## # Database: sqlite 3.22.0 [C:\Users\1bart\Documents\statcastDB.sqlite3]
##        n
##    <int>
## 1 721190
```

So we have collected 721190 observations. We can verify this is the exact number of pitches recorded by Statcast for the 2018 season by using the Statcast website search function and setting the dates correctly.

It is a good idea to keep this value in mind as we explore measurement errors, so we can get an idea for the total percentage of observations affected by any one problem.

We start with one of the most important columns in the table, `pitch_type`. This variable will be an integral part of our anaylsis, so it is crucial that we verify all of its values actually make sense. In the next section, we will explain what each of the identifying strings actually mean, but for now we check the list of official pitch types against our dataset.

```
#Vector with mlbam defined pitch types
pitch_types <- c("FF", "CH", "FC", "FT", "FS", "SL", "CU",
                 "KC", "SI", "FO", "PO", "EP", "SC", "KN" )
db%>%
  tbl("cleancast")%>%
  filter(!pitch_type %in% pitch_types)%>%
  select(pitch_type, player_name, events, release_speed, vx0, ax)%>%
  collect()%>%
  head()
```

```
## # A tibble: 6 x 6
##   pitch_type    player_name     events   release_speed   vx0    ax
##   <chr>         <chr>           <chr>            <dbl> <dbl> <dbl>
## 1 180329_231427 Blake Treinen   <NA>                NA    NA    NA
## 2 180330_041121 Patrick Corbin  home_run            NA    NA    NA
## 3 180330_041103 Patrick Corbin  <NA>                NA    NA    NA
## 4 180408_024144 Reyes Moronta   <NA>                NA    NA    NA
## 5 180408_015747 Josh Fields     <NA>                NA    NA    NA
## 6 180407_204709 Hector Santiago <NA>                NA    NA    NA
```

Here we get a table of 320 pitches that were not identified by Statcast, and instead a number was inserted into the `pitch_type` column. Most of the pitches in this table contain no numerical Statcast measurements. For such pitches, there is nothing that can be done except to review the video and guess what kind of pitch it may have been. Even doing this though would not help in our analysis, because we need specifics of the pitch to truly analyze what is going on. Some, but not many, do have mostly complete Statcast measurements, but

are missing the type of pitch and pitch name, which is interesting. Unfortunately, I have not built a classifier for pitch types based on Statcast measurments yet, so currently we cannot infer the value of `pitch_type` for these observations either. Since there are a relatively small number of pitches contained in this table, they are simply removed.

Next we check the `ball` and `strikes` variables. The rules of baseball dictate that the number of `balls` cannot exceed 3, and the number of `strikes` cannot exceed 2. The following code pulls up all observations in this dataset for which this does not hold.

```
acceptaballs <- c(0,1,2,3)
allowable_strikes <- c(0,1,2)
db%>%
  tbl("cleancast")%>%
  filter(!balls %in% acceptaballs |
         !strikes %in% allowable_strikes)%>%
  select(game_pk, player_name, events, balls, strikes)%>%
  collect()%>%
  kable()
```

game_pk	player_name	events	balls	strikes
529872	Miguel Castro	strikeout_double_play	4	2
529872	Miguel Castro	NA	4	2
530969	Hector Santiago	double	4	2
530969	Hector Santiago	NA	4	2
530969	Hector Santiago	NA	4	2
530969	Hector Santiago	NA	4	2
530969	Hector Santiago	NA	4	2
530969	Hector Santiago	NA	4	2

It appears that no erroneous values have been recorded for strikes, however we have two separate games where the number of balls has been counted as 4, which is impossible. For each of the two instances, we get a table of the entire relevant plate appearance and cross reference the data with a separate party to see if we can figure out what went wrong.

```
db%>%
  tbl("cleancast")%>%
  filter(game_pk == 530969,
         player_name == "Hector Santiago",
         inning == 8,
         outs_when_up ==2)%>%
  select(pitch_type, player_name, balls, strikes, release_speed, zone)%>%
  collect()%>%
  kable()
```

pitch_type	player_name	balls	strikes	release_speed	zone
SI	Hector Santiago	3	2	92.8	6
SI	Hector Santiago	3	1	91.0	1
SI	Hector Santiago	3	0	91.4	3
SI	Hector Santiago	2	0	91.5	11
SI	Hector Santiago	1	0	91.9	12
SI	Hector Santiago	0	0	91.4	12

pitch_type	player_name	balls	strikes	release_speed	zone
CH	Hector Santiago	4	2	82.9	14
SI	Hector Santiago	4	2	92.9	4
SI	Hector Santiago	4	2	92.7	2
CH	Hector Santiago	4	2	84.1	9
SI	Hector Santiago	4	2	92.5	1
SI	Hector Santiago	4	2	92.4	6
SI	Hector Santiago	3	2	92.6	12
SI	Hector Santiago	3	1	93.5	8
SI	Hector Santiago	2	1	91.8	9
SI	Hector Santiago	2	0	92.1	1
SI	Hector Santiago	1	0	91.5	11
CH	Hector Santiago	0	0	84.2	12

This at bat corresponds to one that occurred in the Blue Jays vs White Sox game held on July 27th 2018. I cross-referenced this table with the data provided by ESPN for this game, however it was impossible to match up the plate appearance pitch-for-pitch due to a number of inconsistencies. Ideally one would consult a video in this case, however a video replay of this game could not be located. Because this is such a small sample of pitches, we choose to simply change the value of balls from 4 to 3 in this instance.

We repeat the same process for the second game, which corresponds to the game played between the Orioles and the Angels on May 3rd.

pitch_type	player_name	balls	strikes	plate_x	plate_z
SI	Miguel Castro	4	2	-0.0329	2.0502
SI	Miguel Castro	4	2	-0.4822	2.1699
SI	Miguel Castro	3	2	-1.2936	2.3496
SL	Miguel Castro	2	2	0.9241	0.5952
SI	Miguel Castro	2	1	0.4580	1.3836
SL	Miguel Castro	1	1	1.5792	1.6995
SI	Miguel Castro	1	0	0.5964	2.1692
SI	Miguel Castro	0	0	1.7620	1.6881

By cross-referencing with ESPNs full play-by-play report, we see that the 6th pitch in the sequence was a foul ball, but was recorded as a ball in the Statcast system. In this case, the correct procedure is to simply change the value 4 to the value 3.

Next, we verify the reported release speeds. This can be done by deriving the velocity vector at release, and checking its length against the reported release speed. If the reported value is correct, these numbers should be close. To do this we make use of kinematics. These equations apply because Statcast's model for the motion of the pitch is a constant acceleration model. The relevant code is given below, and the relevant functions can be found in the accompanying rmd or R file.

```
db%>%
  tbl("cleancast")%>%
  collect()%>%
  #The value of t in the trajectory model which corresponds to release time
  mutate(time_release = getTimeToReachYPosition(vy0, ay, y0=50, y_p = release_pos_y),
         time_home = getTimeToReachYPosition(vy0, ay, y0=50, y_p = 17/12 ), #time at home plate
         flight_time = time_home - time_release #The total flight time of the ball in seconds
        )%>%
  mutate(vxR = velocityAtTimet(vx0, ax, t = time_release),
```

```
        vyR= velocityAtTimet(vy0, ay, t = time_release),
        vzR = velocityAtTimet(vz0, az, t = time_release)
        )%>%
  mutate(calc_speed_release = vectorNorm(vxR, vyR, vzR)*(15/22), #15/22 converts ft/sec to mph
        diff_speed = release_speed - calc_speed_release)%>%
  filter(abs(diff_speed) > 1)%>%
  select(game_pk, player_name, release_speed, calc_speed_release, diff_speed)%>%
  kable()
```

game_pk	player_name	release_speed	calc_speed_release	diff_speed
529639	Homer Bailey	92.8	85.23553	7.564470
530204	Caleb Smith	91.7	83.02771	8.672294
530204	Caleb Smith	91.5	90.40799	1.092006
530204	Caleb Smith	84.4	83.06003	1.339972
530204	Caleb Smith	90.6	92.18669	-1.586686
530204	Robbie Erlin	77.6	83.20470	-5.604697

Interestingly, all but one of the 6 pitches that do not satisfy this bound are from the same game and in the same inning, suggesting something faulty occurred with instrument calibration during that inning. We check the game summary on ESPN and it agrees exactly with our calculated speeds. Thus, we will replace release speeds with the calculated release speeds for these cases.

As for the Homer Bailey FS pitch at 92.8 mph, this speed feels incorrect because FS pitches are not typically thrown with this kind of speed. Note that the average speed of a pitch labelled FS is

```
db%>%
  tbl("cleancast")%>%
  filter(pitch_type == "FS")%>%
  pull(release_speed)%>% #grabs the specified table column
  mean()
```

```
## [1] 85.13507
```

and the calculated velocity is 85.2. Furthermore, ESPN again agrees with the calculated speed on this pitch, so it is probably safe to conclude that the calculated value is closer to the true speed at release than the reported value in the dataset. Thus, we can replace all of these incorrect `release_speed` values with the calculated values.

Next are the `plate_x` and `plate_z` values. It is critical that these values are correct, because they give us an indication of how a certain pitch is being located in the strike zone. In this case, we proceed in a similar fashion as before – calculate the `plate_x` and `plate_z` values provided by the trajectory and compare them to the values reported by the Statcast data.

game_pk	pitch_type	player_name	plate_x	p_x	plate_z	p_z	zone
530963	FF	Tyler Bashlor	-1.1357	1.1651864	3.1168	3.102470	11
530963	SI	Edgar Santana	1.1498	-1.1434541	2.5432	2.518741	12
530963	FT	Ivan Nova	1.3039	-1.3043221	2.9661	2.966269	12
530963	FT	Ivan Nova	1.7533	-1.7533811	2.2282	2.228349	14
530963	FT	Ivan Nova	-0.8486	0.8994834	2.4446	2.427412	11
531236	FT	Jesse Chavez	0.8983	-0.8980754	2.4637	2.463597	14

It appears there have been no issues with the `plate_z` values, however there are many problems with the `plate_x` values. Again, almost all of them are from the same game, and in fact the problems only occur in two unique games. The error this time is easy to see. It appears that the `plate_x` values were reported with the incorrect sign. This is a problem that is quite easy to fix, however, we should also check the values of `zone`, because it is possible that these values were also reported incorrectly if they are a function of `plate_x` and `plate_z`. After reviewing the table, it turns out that this is in fact the case. To fix these values, we simply mirror them to the corresponding zone on the other side of the plane $x = 0$.

Above, we identified a few measurement errors present in the dataset. There may be more, but these fixes should be sufficient for the present analysis. The following code amends the measurement errors we found and writes the amended table to the database under a new name.

```
db%>%
  tbl("cleancast")%>%
  collect()->
  fix_cleancast
fix_cleancast%>%
  #Remove pitches with uniterpretable pitch type / no Statcast measurments
  filter(pitch_type %in% pitch_types)%>%
  mutate(balls = replace(balls, balls==4, 3))%>%
  #time _release: gives the value of t that returns position at release from trajetory eqn
  #time_home: value of t that returns the position the ball at home plate from trajectory eqn
  #flight_time: total time from release to home plate, in seconds
  mutate(time_release = getTimeToReachYPosition(vy0, ay, y0=50, y_p = release_pos_y),
         time_home = getTimeToReachYPosition(vy0, ay, y0=50, y_p = 17/12 ),
         flight_time = time_home - time_release
        )%>%
  mutate(vxR = velocityAtTimet(vx0, ax, t = time_release),  #The release velocities
         vyR = velocityAtTimet(vy0, ay, t = time_release),
         vzR = velocityAtTimet(vz0, az, t = time_release)
        )%>%
  #Fix incorrect release speeds
  mutate(calc_speed_release = vectorNorm(vxR, vyR, vzR)*(15/22), #15/22 converts ft/sec to mph
         diff_speed = release_speed - calc_speed_release,
         release_speed = if_else(abs(diff_speed) > 1, round(calc_speed_release,1), release_speed )
        )%>%
  #Get plate positions according to the given trajectories.
   mutate(p_z = position_at_t(release_pos_z, vzR, az, time_home, time_release),
          p_x = position_at_t(release_pos_x, vxR, ax, time_home, time_release),
          zone = if_else(abs(plate_x - p_x) >1,    #Mirror the zones to be correct
                         case_when(zone == 1 ~3,
                                   zone == 3 ~1,
                                   zone == 4 ~6,
                                   zone == 6 ~4,
                                   zone == 7 ~9,
                                   zone == 9 ~7,
                                   zone == 11 ~12,
                                   zone == 12 ~11,
                                   zone == 13 ~14,
                                   zone == 14 ~13
                                  ),
                           zone)
  )%>%
  #Drop old plate values to keep plate values consistent with trajectories.
```

```
  select(-c(plate_x, plate_z))%>%
  dbWriteTable(db$con, "calccast", ., overwrite = TRUE)

rm(fix_cleancast)
```

Comparison of Pitch Types

We can now turn our attention to analyzing the individual pitch types. As previously mentioned, there are a total of 14 pitch types in the dataset. Below is a table which gives a small breakdown of each pitch type.

Pitch Acronym	Pitch Name	Description
CH	Changeup	A changeup is one of the slowest pitches thrown in baseball, and it is predicated on deception. A good changeup will cause a hitter to start his swing well before the pitch arrives, resulting in either a swing and miss or very weak contact. They are thrown with a nearly identical motion to that of a fastball, causing the intended deception.
CU	Curveball	A curveball is a breaking pitch that has more movement than just about any other pitch. It is thrown slower and with more overall break than a slider, and it is used to keep hitters off-balance. A curveball's movement is typically 12-6 or 11-5, so that its break is mostly vertical.
EP	Eephus	The eephus is one of the rarest pitches thrown in baseball, and it is known for its exceptionally low speed and ability to catch a hitter off guard. Typically, an eephus is thrown very high in the air, resembling the trajectory of a slow-pitch softball pitch. Hitters with poor plate discipline may find themselves swinging too early on this slow pitch, for patient hitters however, it is the easiest pitch to hit in baseball.
FC	Cutter, Cut Fastball	A cutter is a version of the fastball, designed to move slightly away from the pitcher's arm-side as it reaches home plate. The cutter breaks in the opposite direction of a two-seamer, and it does so very late in its journey to home plate. This movement is designed to make sure the hitter isn't able to hit the pitch squarely.
FF	Four-seam Fastball	A four-seam fastball is almost always the fastest and straightest pitch a pitcher throws. It is also generally the most frequently utilized. The four-seam fastball is typically one of the easiest pitches for a pitcher to place, because of the lack of movement on the pitch.
FO	Forkball	One of the rarest pitches in baseball, the forkball is known for its severe downward break as it approaches the plate. Because of the torque involved with snapping off a forkball, it can be one of the more taxing pitches to throw.
FS	Splitter	Splitters are often referred to as "split-finger fastballs," but because of their break and lower velocity, they don't hold much in common with a typical fastball. They're generally thrown in the same situations that would see a pitcher throw his breaking and off-speed pitches. A splitter is generally only slightly faster than a changeup.
FT	Two-seam Fastball	A two-seam fastball is often a few ticks slower than a four-seam fastball, but it tends to have more movement. With a two-seamer, the ball moves in the same direction as whichever arm is being used to throw it (meaning a right-handed pitcher gets rightward movement on a two-seamer) and typically dives a bit right as it reaches home plate.

Pitch Acronym	Pitch Name	Description
KC	Knucle-Curve	The knuckle-curve is one of baseball's greatest paradoxes, given that a curveball is defined by its spin and a knuckleball is defined by its lack thereof. Still, the knuckle-curve produces the desired effect of the two pitches – a slow, curveball break mixed with the unpredictable fluttering of the knuckleball.
KN	Knucleball	The goal of a knuckleball is to eliminate almost all of the spin on the baseball, causing it to flutter unpredictably on its way to the plate.The obvious downside to the knuckleball is that if it isn't "dancing" it becomes very easy to hit because of the slow speed of the pitch. Because the pitch is so hard to master and the risks of throwing a bad knuckleball are so great, very few pitchers throw the knuckleball
SC	Screwball	A screwball is designed to move in the opposite direction of a pitcher's other breaking balls, which typically all break toward the pitcher's glove hand. It is one of the rarest pitches thrown in baseball, mostly because of the tax it can put on a pitcher's arm.
SI	Sinker	The sinker is a pitch with late downward movement. It's generally one of the faster pitches thrown and very similar to the 2-seam fastball (there is debate about whether the two are actually different pitches).
SL	Slider	A slider is a breaking pitch that is thrown faster than your average breaking ball. Its movement is more lateral than a curveballs, often "sliding" across the front of home plate to end up in the corner of the strike zone on the pitcher's glove side.

Source: MLBAM, taken from http://m.mlb.com/glossary/pitch-types

There is one final pitch type not mentioned in this table, and it carries the acronym PO (pitch out). It was explained earlier when discussing the variate `description`. We do not consider this pitch at all in the remaining analysis, because it is really only a pitch in the techical sense, and there is nothing of interest happening with this pitch.

We now check just how many of each pitch there is

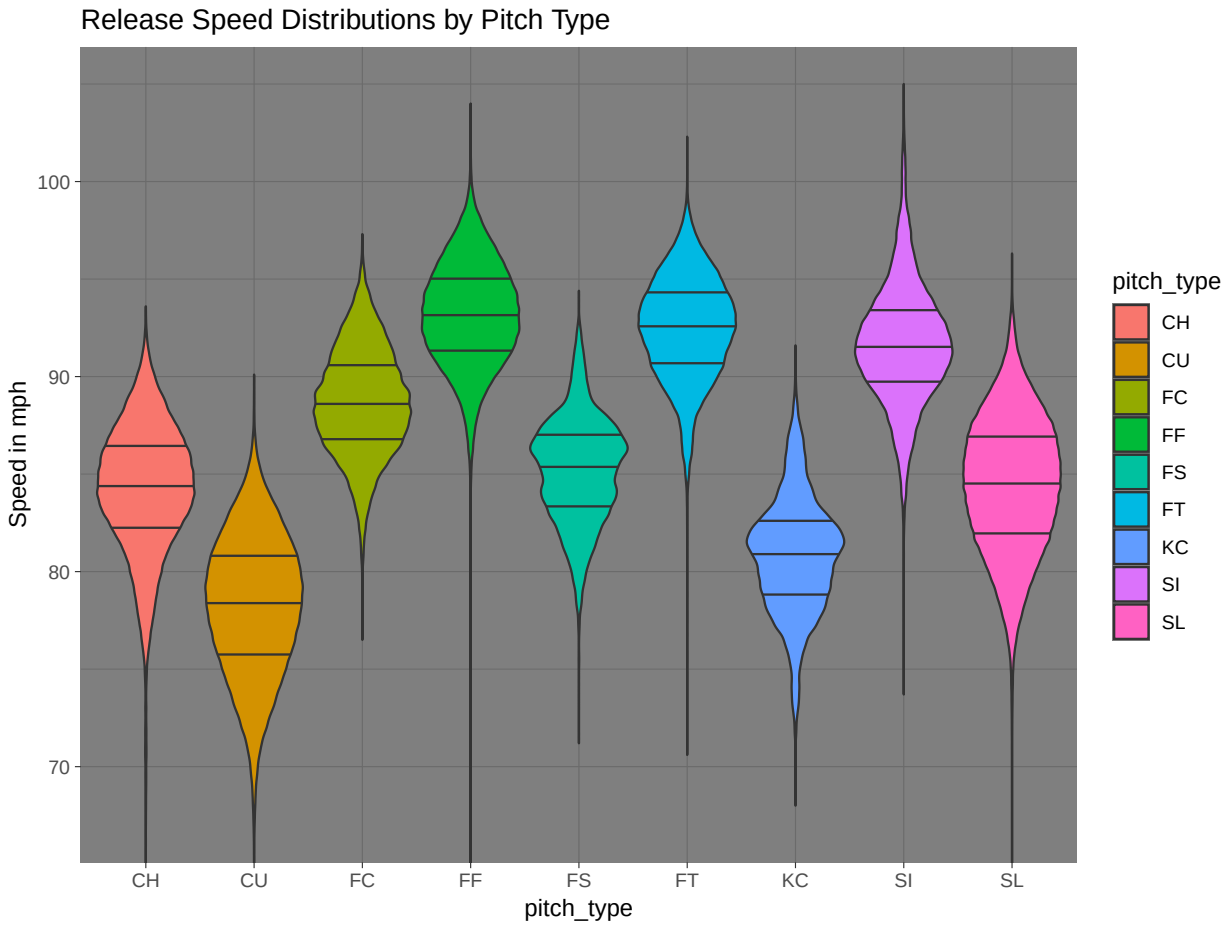
```
db%>%
  tbl("calccast")%>%
  pull(pitch_type)%>%
  table()
```

```
## .
##     CH     CU     EP     FC     FF     FO     FS     FT     KC     KN
##  74272  59438    204  39208 253657     83  10002  80265  16120      6
##     PO     SC     SI     SL
##     89     37  59206 113839
```

We see immediately that FF is the most common pitch by a wide margin, with the slider (SL) coming second with under half as any observations. Unfortunately, we have only 84 forkballs, 37 screwballs, and 204 eephus pitches observed. We choose to drop both FO and SC, as their low number of observations tend to skew their results in the following analyses. We also drop the eephus pitch, as it is truly not that interesting, and is essentially a novelty pitch. Althought the Knuckleball seems to have quite a few observations, once stratification on other variables begins, the ensuing srata of Knuckleballs become quite small, which in turn leads to skewed results. Furthermore, there are currently no active knuckleballers in the league, which speaks to its position in the metagame as something of a relic. For these reasons we also drop KN from the analysis.

We now investigate what features of each pitch type make it unique. First, we look at the distribution of the speed at release for each pitch type.

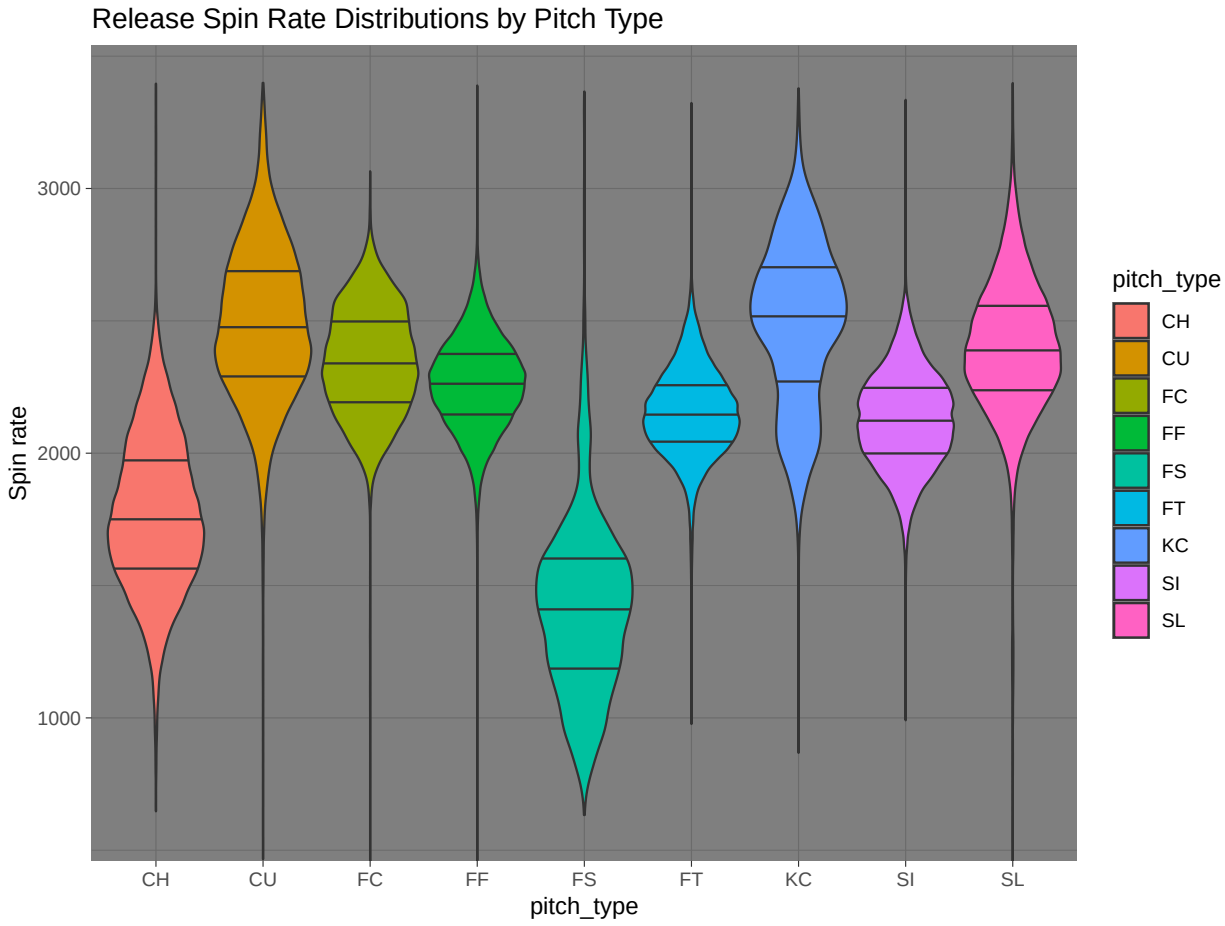
```
db%>%
  tbl("calccast")%>%
  filter(!is.na(release_speed),
         !pitch_type %in% c("FO", "PO", "SC", "KN", "EP"),
         #Remove poorly thrown pitches, which may skew the data
         !description %in% c("blocked_ball", "hit_by_pitch"))%>%
  select(pitch_type, release_speed)%>%
ggplot(., aes(pitch_type, release_speed, fill = pitch_type)) +
geom_violin(scale = "width",
            adjust = 0.7,
            draw_quantiles = c(0.25, 0.5, 0.75)
  ) +
  coord_cartesian(ylim = c(67,105)) +
  ggtitle("Release Speed Distributions by Pitch Type") +
labs(y = "Speed in mph") +
  theme_dark()
```



The first thing to notice here is that there are clearly a few lower tail outliers in the data, which is likely due to pitches thrown by position players–an event that occurs occassionally during blowout games. Ignoring these, we can see that FF, FT, and SI are generally the fastest pitches being thrown, with the highest median being the FF at around 92.5 mph. Given that two of these pitches have the word "fast" in their name, this isn't an entirely surprising result. We also see that the distribution of CH is extremely similar to that of FF, but lies in a lower speed tier. It does, however, have some overlap with the FF distribution in terms of

observed release speeds. Based on the MLB blubs, this may mean that one pitcher's changeup (CH) may technically the same as another pitcher's FF. This illustrates that an FF and CH may not be defined by pre-determined speed tiers, but instead defined by the difference in the speed with which they are thrown by a specific pitcher.
The CU and SL pitches appear to have the largest variance in terms of speed, with CU also being the slowest overall pitch. This is likely due to the fact that CU and SL pitches rely on tricky movement to deceive a batter, rather than pure speed. The buldges that give the FS and KC pitches a unique shape are likely due to extreme use by one or two pitchers, as these pitches themselves are a bit rarer than the rest.

The next variable we will investigate is the amount of spin that is measure on the pitch at release.

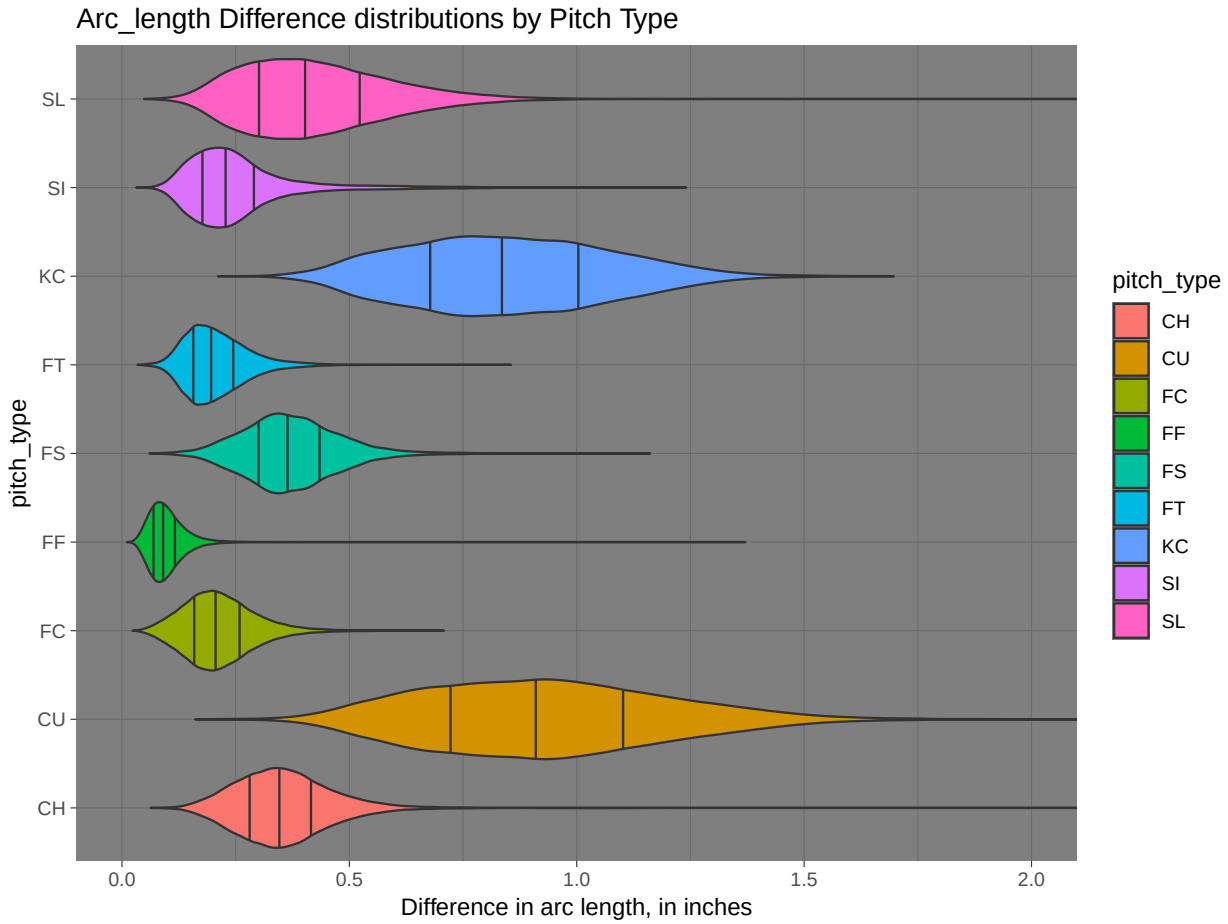


Again we see a few outliers, but otherwise the distributions seem reasonably well defined. we see CU, KC, and SL with the highest median values. Interestingly, these are also the pitches which made up the lowest speed tier in the first plot. These are also the pitches that use movement to deceive batters, so this plot indicates that they are probably generating this movement by utlizing spin. This result also appears somewhat contrary to MLBAM's blurb on KC pitches, which seemed to suggest that they were somehow curveballs without spin.

The FF, FT, and SI distributions are also somewhat similar here, as they were in the speed graphs. They again appear as the distributions with the least amount of overall variance as well. As the MLB blurbs suggested, this low variance may mean that FF, FT, and SI pitches are easier to throw with some consistent properties–they may be more robust with respect to a change in the finger positions and throwing motions used to throw them. Unlike CU, KC, and SL, the spin on these pitches is not specifically meant for movement, and so it is probably for stability–akin to a football. The high variance in the CH distribution

relative to the FF distribution may indicate that spin is less important for a changeup, or it may indicate that putting spin equivalent to that of a fastball on a slower pitch is difficult. Again we see the overlap of these two distributions that we saw before, again indicating that a specific CH may only be a CH relative to the pitcher's own FF.
The most interesting pitch here may be the FS (splitter). It is clearly in a low tier all its own when it comes to spin rate. It had about middle tier speed but low spin, which may suggest that it actually has some knuckling action.

In the interest of determining just how much curve there is to a pitch's trajectory in the general case, we devise a measure that may help to illustrate this. We proceed by finding the arc length of each pitch trajectory and comparing it with the length of the line that goes directly from the trajectory's start point to end point. For straight pitches, these lengths should be nearly identical, but for pitches with a lot of movement – i.e. paths that are not straight – this value could be large. To do this, we make use of the `pracma` package's `arc_length` function. The code and resulting graph are given below. Because calculating all of these trajectories is computationally intensive, we write the arc length data directly to the table in the database, so that they can be called upon later at any time.



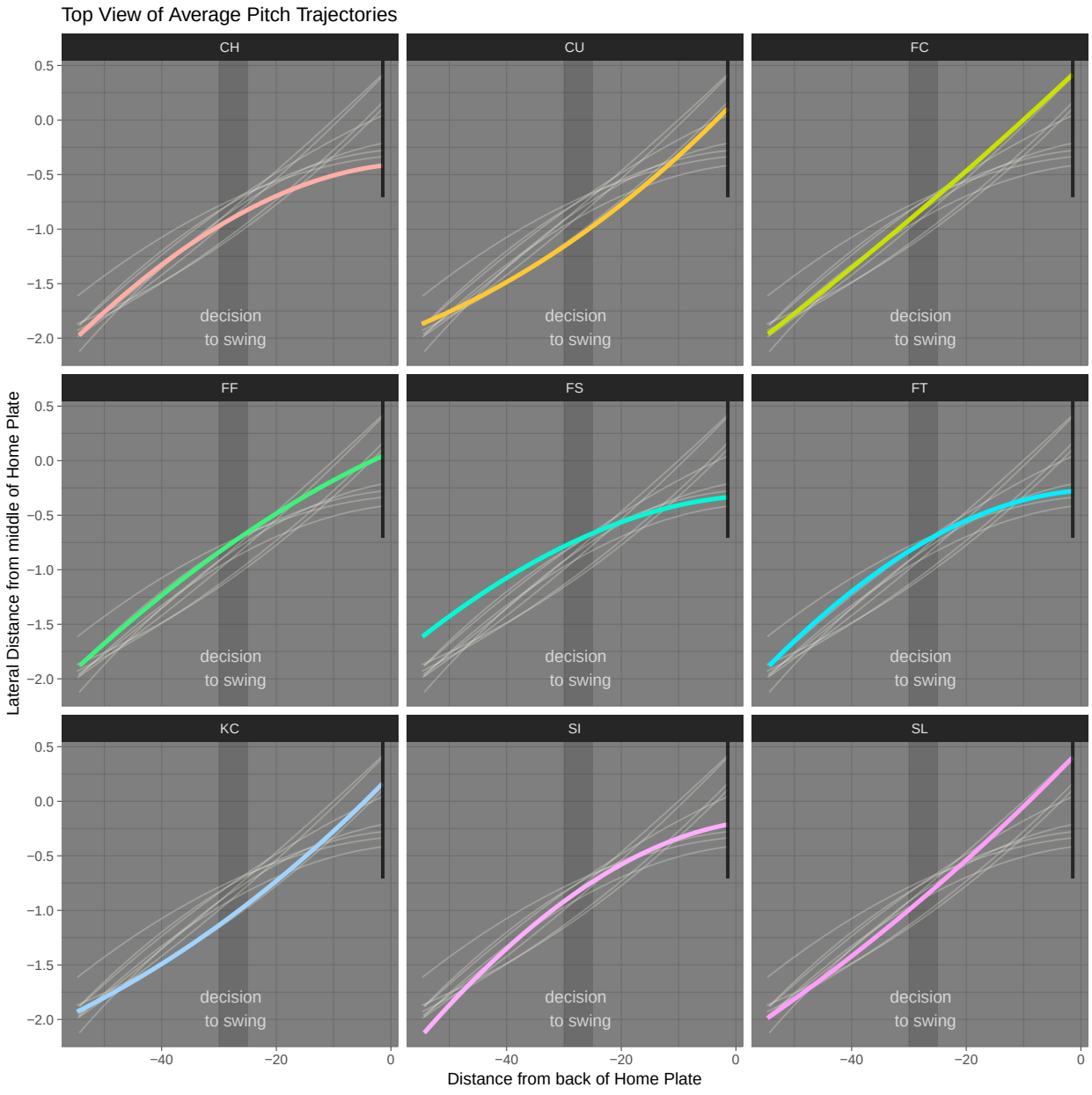
This graph is interesting in that it seems to suggest there are four tiers of pitches, in terms of deviance from a linear path. We base the tiers on the quantiles, which nearly line up inside a tier, but are completely below or above the quantiles of other tiers. The first tier, as one would expect, is simply the fastball. With most values close to 0 and having very little variance relative to the other pitches, it appears that this pitch is consistently the straightest. The second tier consists of the SI, FT and FC pitches. According to the MLBAM blurbs, all of these pitches are intended to have late movement, so that is likely what is being

picked up here. This idea is also consistent with the earlier plots, which show that these pitches are a bit slower than the fastball, with approximately the same amount of spin. It is possible that this combination provides the extra movement at the end, similar to how a frisbee bends as it loses speed.
The next tier is comprised of CH, FS, and SL, and is the most surprising. First, from the above graphs, and again from the blurbs, we surmised that a CH was basically an FF but slower. Here we see that this is certainly not the case. We know this because, although there was overlap in speed and spin for these two pitches, there is significantly less overlap here in movement. This suggests that a CH will always have more movement than an FF, regardless of whose specific CH and FF pitch are being discussed. Taking into account the large variation we saw in spin rate as well, its probably safe to conclude that the CH is indeed its own unique pitch, and not just a slower fastball.
Given the fact that the FS pitch lives in an approximately equal speed tier with the CH, and that these two pitches also had the lowest spin distributions, its not surprising that FS shares a movement tier with CH. What is surprising, is that SL is in this tier. An SL pitch, in theory, should have as much movement as a CU, but with more horizontal break. Here we see that it does have the third highest median, however it quite clearly lives in a tier below that of the CU and KC pitches, which suggests that its movement is not quite as extreme as these pitches.

To further investigate the movements of each pitch type, we find the average trajectory of each pitch type and plot the projection of the trajectory on the xy plane and the yz plane. The xy projection will show us how each pitch is moving laterally, while the yz projection should demonstrate how each moves vertically. We do this only for right-hand pitchers. The plot for left-handed pitchers would be equivalent, but mirrored about the plane $x = 0$.

```
# The means_tbl was created in code not shown. Each of its rows contains the
# mean of each of the 9 trajectory parameters for one of the pitch types.

# Create a table called projection_plots, containing the points to be plotted
#for each  of the mean trajectories
means_tbl%$%
  do.call(rbind, mapply( plotProj, xr = release_pos_x,
                                   yr = release_pos_y,
                                   vxr = vxR,
                                   vyr = vyR,
                                   ax = ax,
                                   ay = ay,
                                   tf = time_home,
                                   pitch_type = pitch_type,
                                   SIMPLIFY = FALSE)
          )->
  projection_plots

# Add a column called "repeat_type" which is a carbon copy of the pitch_type column
projection_plots%>%
  mutate(repeat_type = pitch_type)->
    projection_plots

#Here we plot the trajectories.
projection_plots%>%
  ggplot(aes(x=-y, y=x, group = pitch_type, color = pitch_type)) +
  #Creates the swing bar and the accompanying text
  annotate("rect", xmin=-30, xmax=-25,
                   ymin=-Inf, ymax=Inf,
                   alpha=0.15, fill="black"
          ) +
  annotate("text",
```

```
          x = -27.5,
          y = -1.9,
          label = "decision \n to swing", color = "white", alpha = 0.7
         ) +
 #Creates the background plot which includes all of the trajectories
 geom_line(data = transform(projection_plots, pitch_type = NULL),
           aes(group = repeat_type),
           color = "ivory", alpha = 0.25
          ) +
 #Graphs the colored trajectories
 geom_line(lwd = 1.4) +
 scale_color_hue(l=85, c=100) +
 #Graphs the vertical bar representing the strike zone
 annotate("segment",
           y = (-17/24),
           yend = Inf,
           x =-17/12,
           xend = -17/12,
           color = "black", alpha = 0.7, lwd = 1.1
         ) +
#Title and theme options
theme_dark() +
 ggtitle("Top View of Average Pitch Trajectories") +
 xlab("Distance from back of Home Plate") +
 ylab("Lateral Distance from middle of Home Plate") +
 theme(legend.position = "none") +
 facet_wrap(~pitch_type, nrow = 3)
```

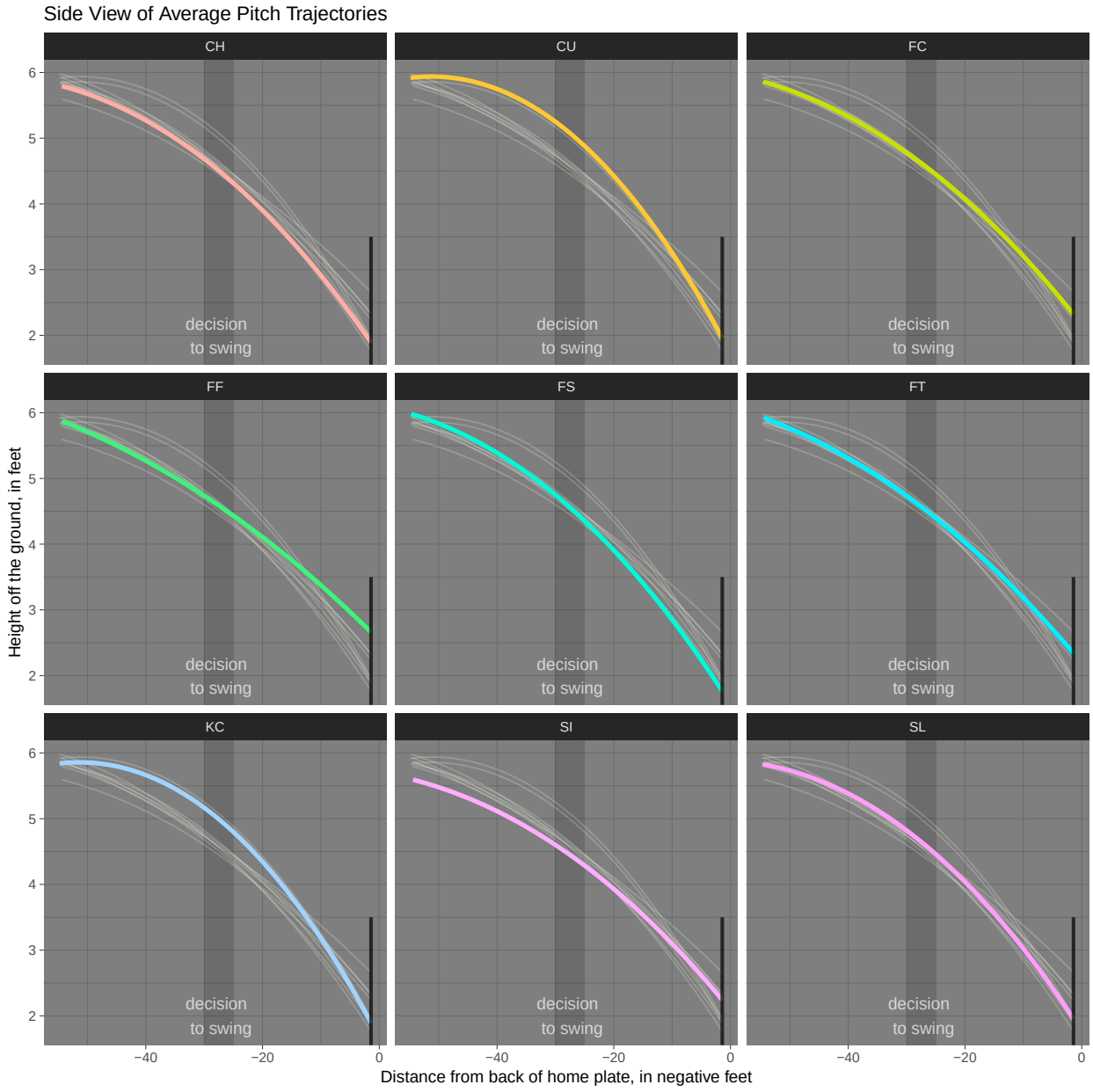



```
rm(projection_plots)
```

In this plot we see a top-down view of each pitch's average trajectory with release point in the bottom right hand corner, and the strike zone in the top right. It is immediately obvious that pitches are not released directly in front of the strike zone, but instead off to the side and then travel somewhat diagonally toward to strike zone (which is marked by the vertical line in the top right). The gray bar labelled "decision to swing" approximates the point in the trajectory of a pitch at which the batter must decide whether or not to swing. Of course this point will vary depending on pitch speed, swing speed, and a variety of other factors, which is why it is shown here as a ribbon rather than a line. It is mainly here as a visual aid. The graph shows that the FC and SL pitches are typically located on the left (relative to the pitcher) side of the strike zone and therefore travel furthest laterally. These would be moving away from a right-handed batter and in on the hands of a left-handed batter. On the flip side, the CH, FT, and SI pitches illustrate similar average trajectories, starting out toward the middle of the strike zone and then cutting back toward the pitcher's

arm side and ending up on the right side of the strike zone. The FS pitch sports a similar trajectory to this as well, but seems to provide less lateral break. The FF, KC, and CU pitches seem to be located at the middle of the plate on average, with the KC and CU pitches illustrating lateral break toward the left side of the strike zone, which is the opposite direction of the break seen on the CH, FT, and SI pitches.

Next we plot the yz projections, which will help illustrate vertical movement of each pitch. The code for these plots is similar, and so is omitted.

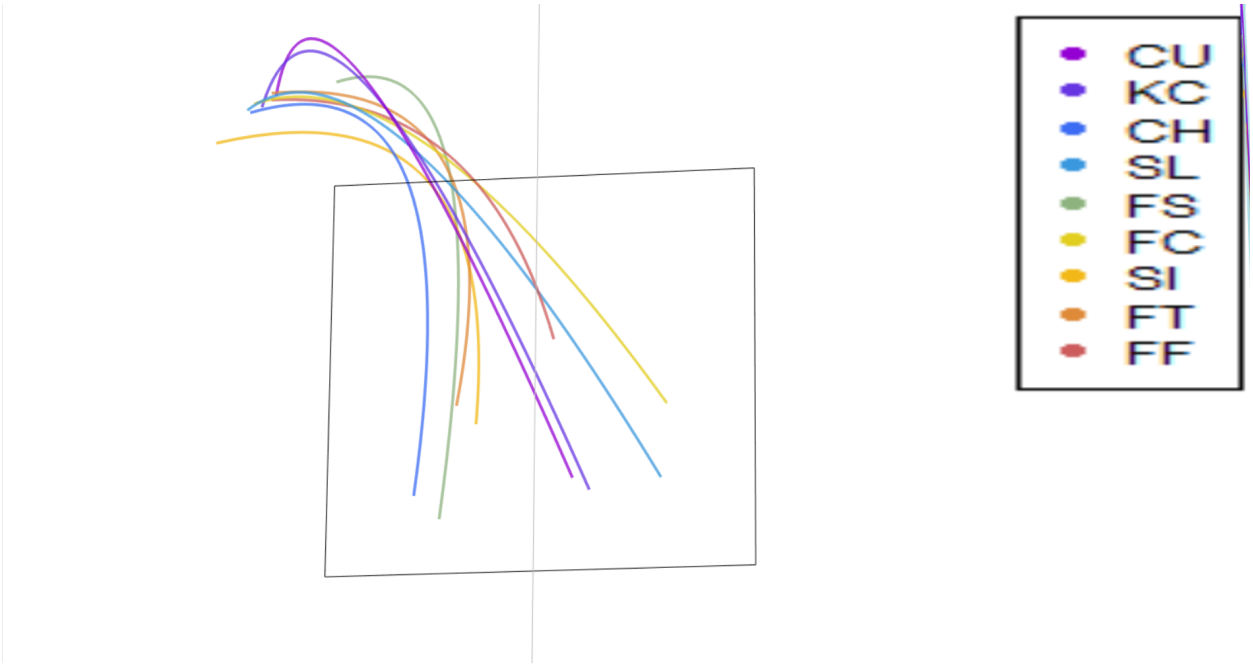


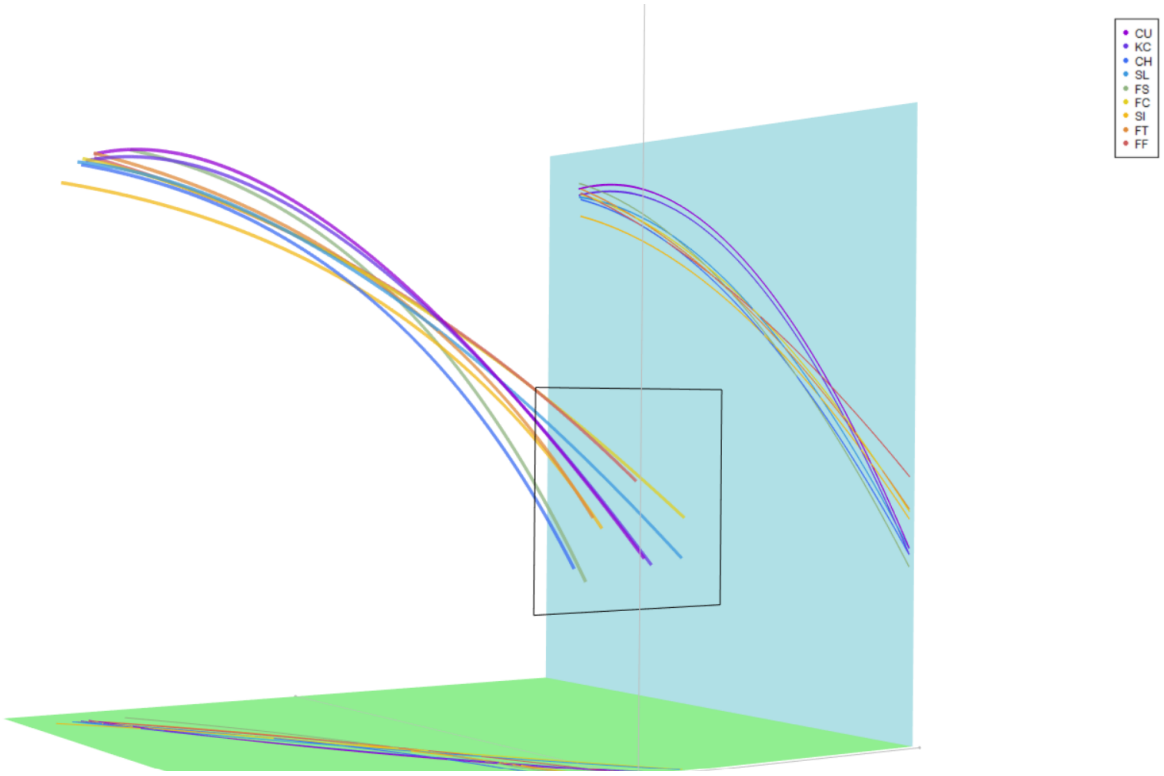
The first thing to notice here is that all of the pitches have a strong downward trajectory, showing much greater vertical movement than they did lateral movement. This is due in part to gravity, but also to the fact that the pitcher stands on a mound above the field, and so throws from an elevated position with respect to the strike zone.

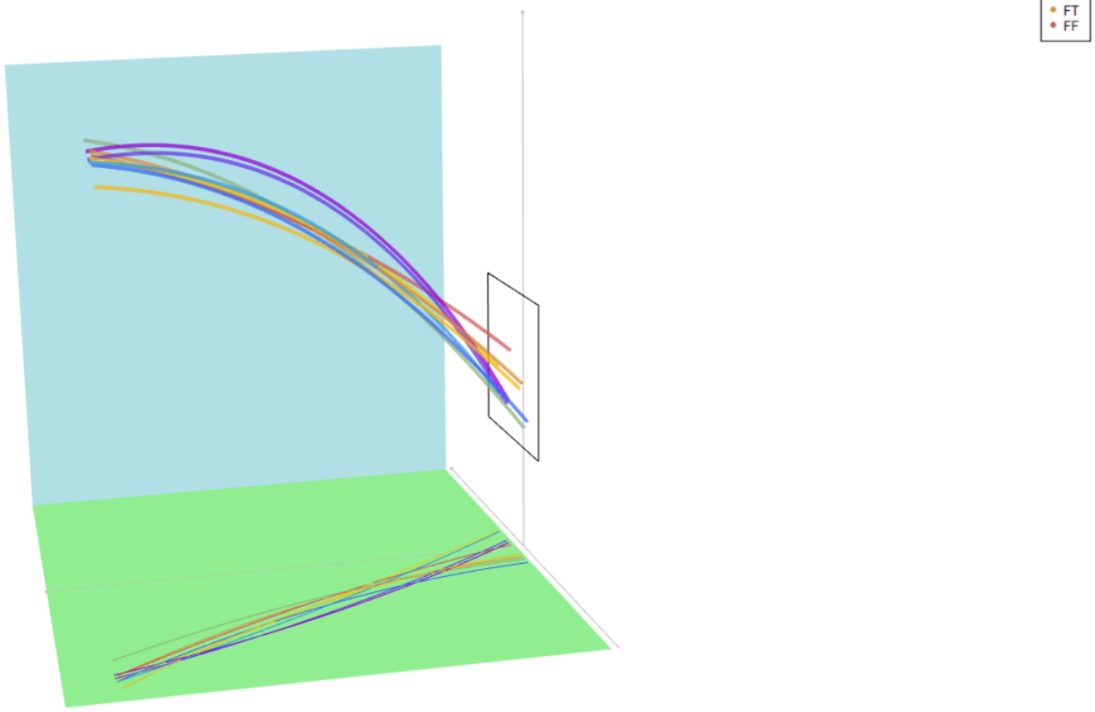
In terms of individual pitch movement, we see that the FS pitch has the highest average release point and

lowest average position at the strike zone, making it the pitch with the most vertical movement on average, and therefore the truest "sinking" pitch. This is in contrast to the SI (sinker) and FT, which are a bit faster and demonstrate much less sink on average than their names/blurb suggest. It is possible that with their higher speed, they do not need to sink as much to deceive a batter. The also demonstrated not insignificant lateral movement in the previous plot, which may indicate that they rely on a combination of sinking and lateral movement to generate deception. The KC and CU have much more bowed or curved trajectories than the other pitches, which is not surprising given their name. These two pitches are the highest at the "decision to swing" point, and also end up quite low in the strike zone, adding more evidence to the idea that they are some of the more deceptive pitches in terms of vertical movement. Finally, we also get more confirmation that the CH is different than the FF, as it shows considerably more dip in its trajectory than the FF, and also ends up as one of the lowest pitches at the strike zone, while the FF is the highest on average. Not surprisingly, the FC is the second highest pitch on average, and at this point seems to resemble something of an upper zone SL, albeit with a faster speed and a bit less overall lateral movement.

Finally, to put all of this information together, we create a 3D plot of these mean trajectories using the `rgl` package. The projections are also graphed and the trajectories are colored by speed. Only a snapshot of the graph is shown here, but the code to generate the graph has been provided in the accompanying .R file. Playing around with this plot can really help develop an intuitive understanding of the general movement of each pitch, and if you have the time and inclination, you should do so before reading the rest of the analysis. These plots for example, illustrate the stark difference between the CH and FF pitches we have begun to notice, with the CH demonstrating significant, and very interesting movement compared to the FF.




CU
KC
CH
SL
FS
FC
SI
FT
FF


CU
KC
CH
SL
FS
FC
SI
FT
FF

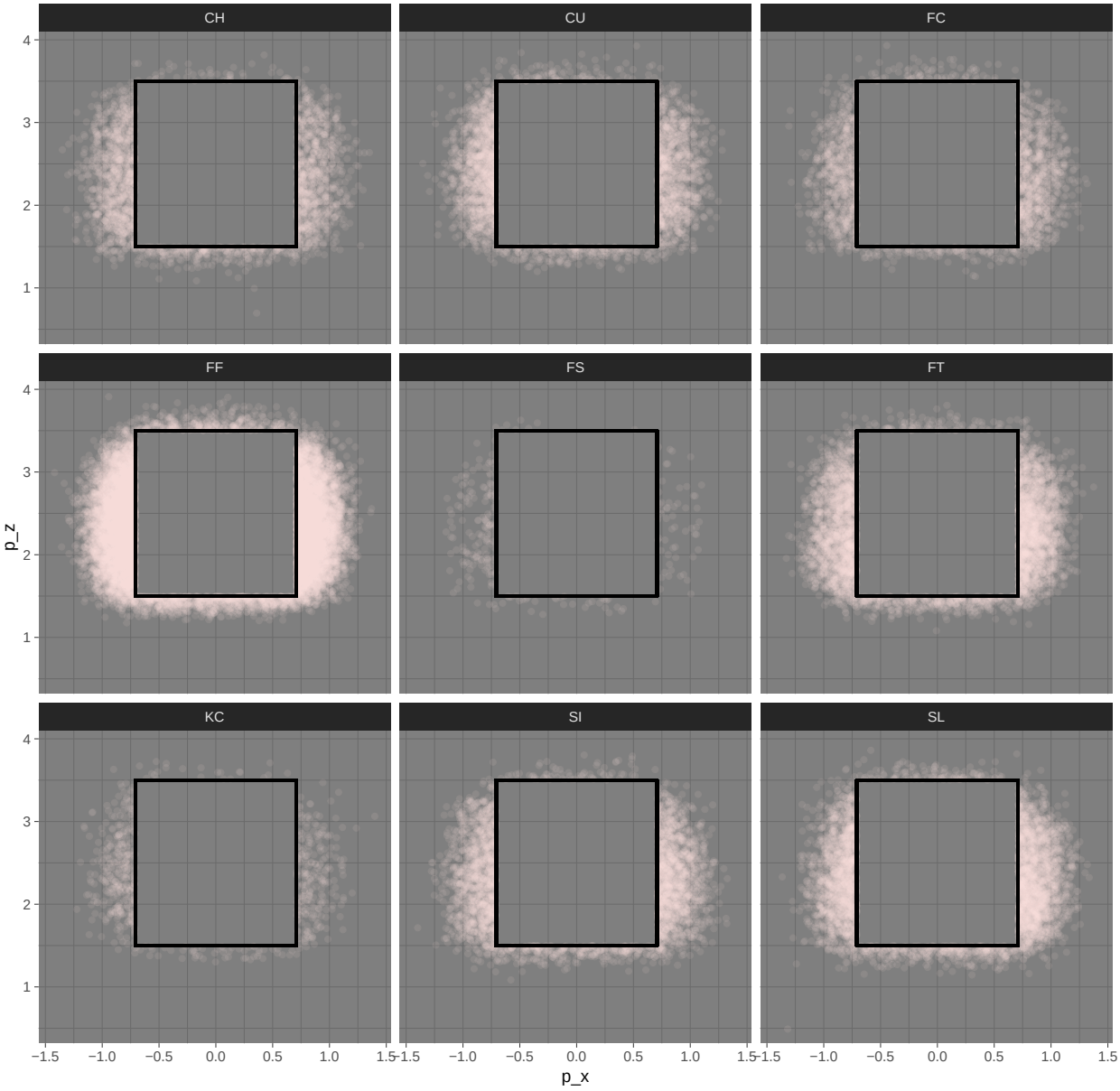
Evaluating the Role of Each Pitch Type

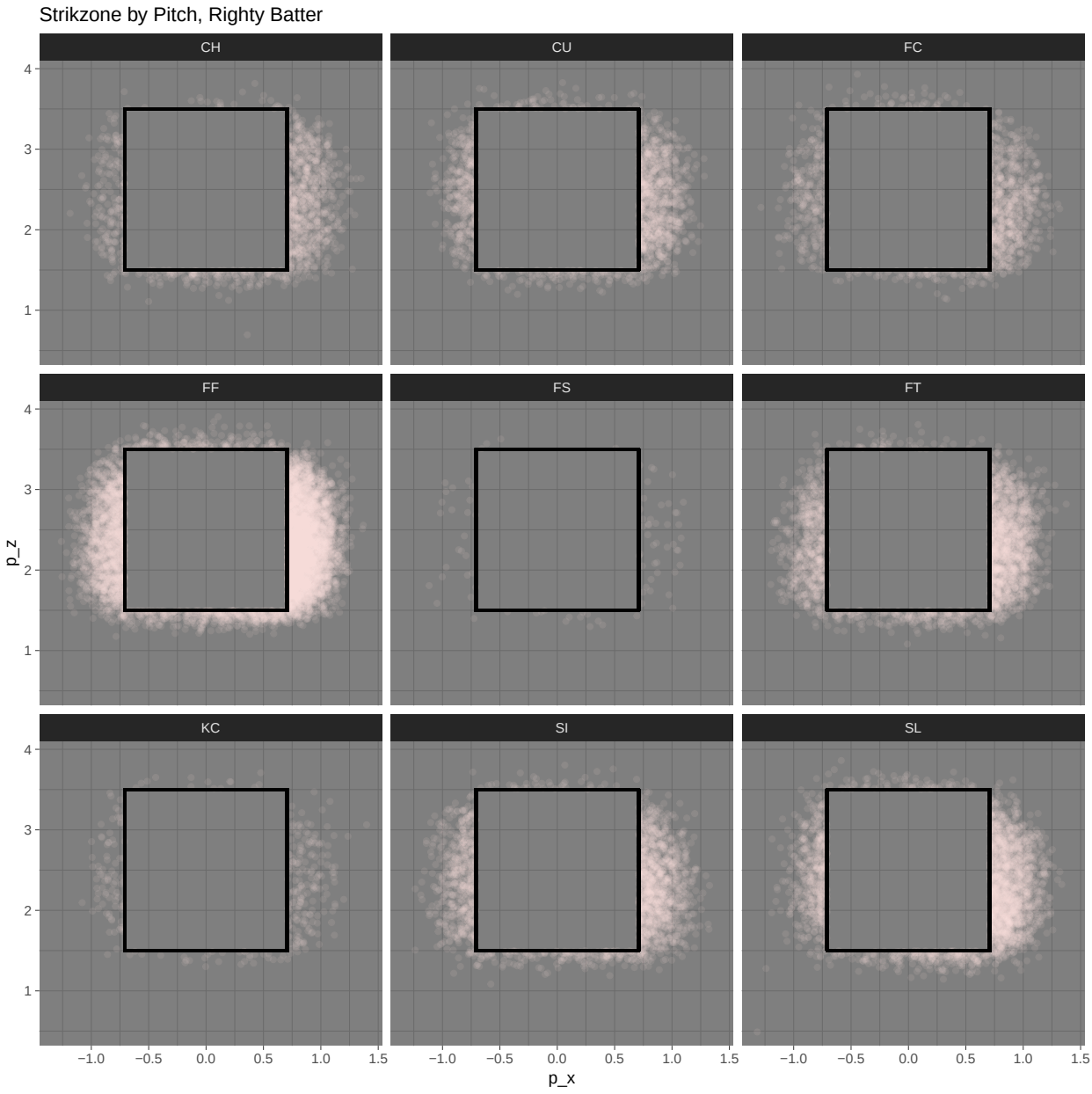
Now that we have some idea of how each pitch moves, we want to understand the benefit each pitch type brings to a pitcher's arsenal. That is, we want to decern the purpose of each pitch, and in what situations it can be utilized most effectively.

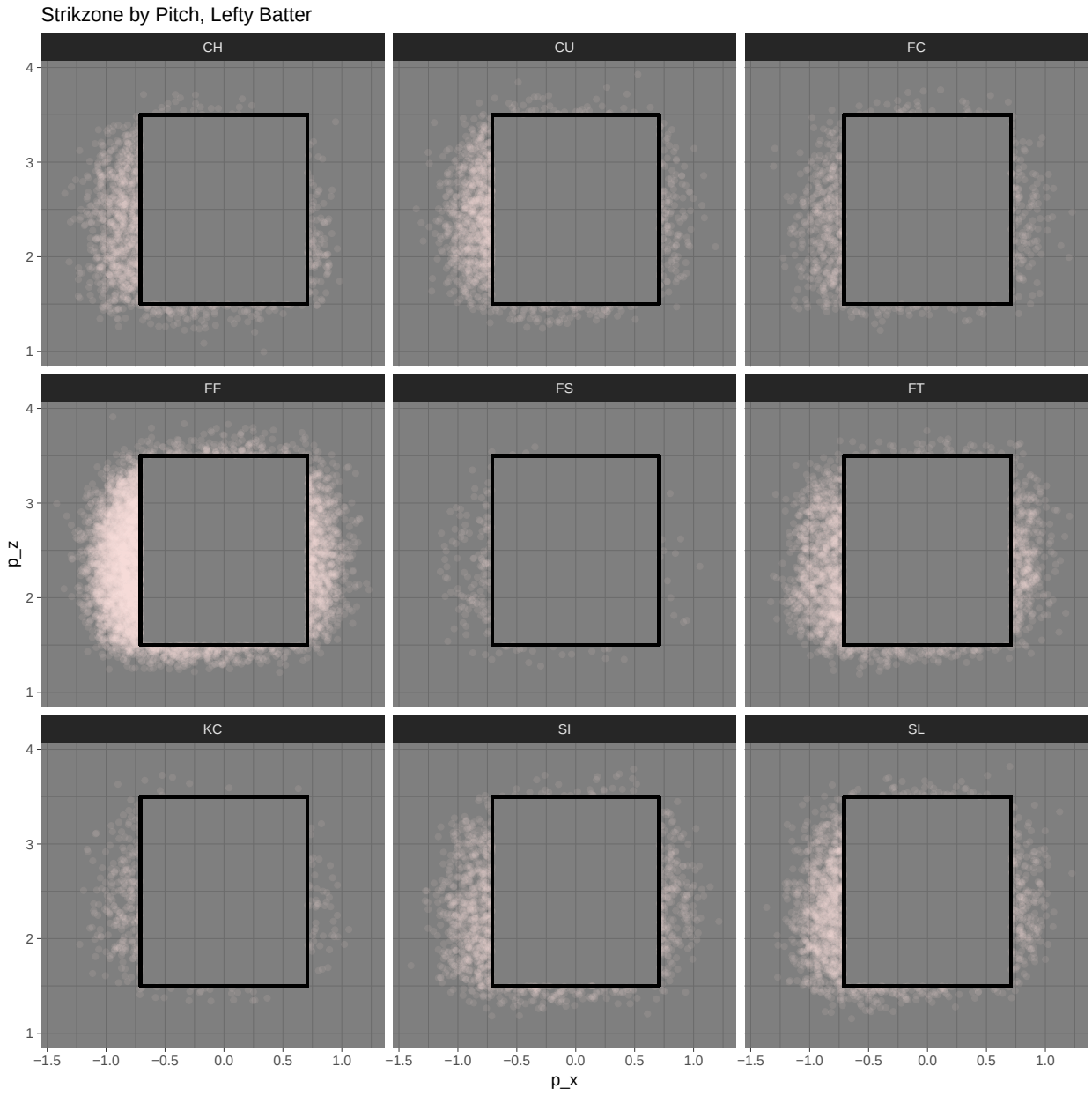
One desirable quality of a pitch is to produce a strike – whether its a strike looking or a strike swinging (a whiff). The first type of strike, a strike looking, is completely determined by the particular person umpiring that game. Due to the extreme movement, speed and variability in major league pitches, it is possible that umpires have more trouble calling certain types of pitches. If this is the case, it is possible that some pitches have more liberal, or more strict "called strike" zones. We can test for this by plotting all pitches that were called strikes–not swung on and deemed inside the strike zone by the umpire–outside of the strike zone, and stratify by type. Rather than also stratifying by pitcher handedness, we instead stratify by batter handedness. We do this because horizontal pitch locations already tell us a bit about the handedness of the pitcher who threw it. For example, we expect a right-handed pitcher's slider to be located on the left-hand side of the strike zone from the pitcher's perspective, or the right-hand side of the strike zone from the catcher's (guy standing behind home plate) perspective. It is also more likely that the umpire's strike zone is more strongly influenced by the location and size of the batter rather than the handedness of the pitcher.

```
#All pitchers
db%>%
  tbl("calccast")%>%
  filter(description =="called_strike",
         !pitch_type %in% c("PO", "FO", "EP", "SC", "KN"),
         (abs(p_x) > 17/24) |
           p_z > 3.5 |
           p_z < 1.5)%>%
  # Notice we do not call the collect() function here before
  # sending to ggplot. This was initially a mistake. I do not know
  # how it retrieves the data when it is left this way. Possibly
  # the pass to ggplot implicitly calls the collect() function
  select(pitch_type, description, p_x, p_z)%>%
  ggplot(., aes(x= p_x, y = p_z)) +
  geom_point( color = "mistyrose", shape = 19, alpha = 0.1) +
  geom_rect(aes(xmin = -17/24,
                xmax =  17/24,
                ymin =  1.5,
                ymax =  3.5),
                fill = NA, color = "black", size = 1) +
  facet_wrap(~pitch_type, nrow = 3) +
  ggtitle("Strikzone by Pitch") +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme_dark()
```

Strikzone by Pitch




Strikzone by Pitch, Righty Batter
CH
CU
FC
FF
FS
FT
KC
SI
SL
p_z
p_x
4
3
2
1
−1.0
−0.5
0.0
0.5
1.0
1.5



From these plots we can see that umpires are far better at defining the strike zone vertically than they are horizontally. This is actually a bit of a surprise, given the definition of the strike zone and the fact that home plate is always there and provides a clear reference point for the outer bounds of the zone. However, because this problem is nearly uniform across pitches, it is likely that this error is due to the umpire's perspective from his location behind the plate. We further note that stratifying by batter handedness shows that pitches away from the batter are routinely called strikes outside the strike zone. We cannot say for certain that pitches inside are called less often, as it may be the case, for example, that pitches inside are swung on more often. We can say that pitches inside do sometimes get called strikes as well, but there does not seem to be a stronger or weak bias on this call from one pitch to another.

From these plots we conclude that no one pitch type provides an advantage when its comes to "called strike" zone, however pitches away from the batter are commonly called strikes when they are in fact balls, so this seems like a fairly safe place to throw any of the pitch types.

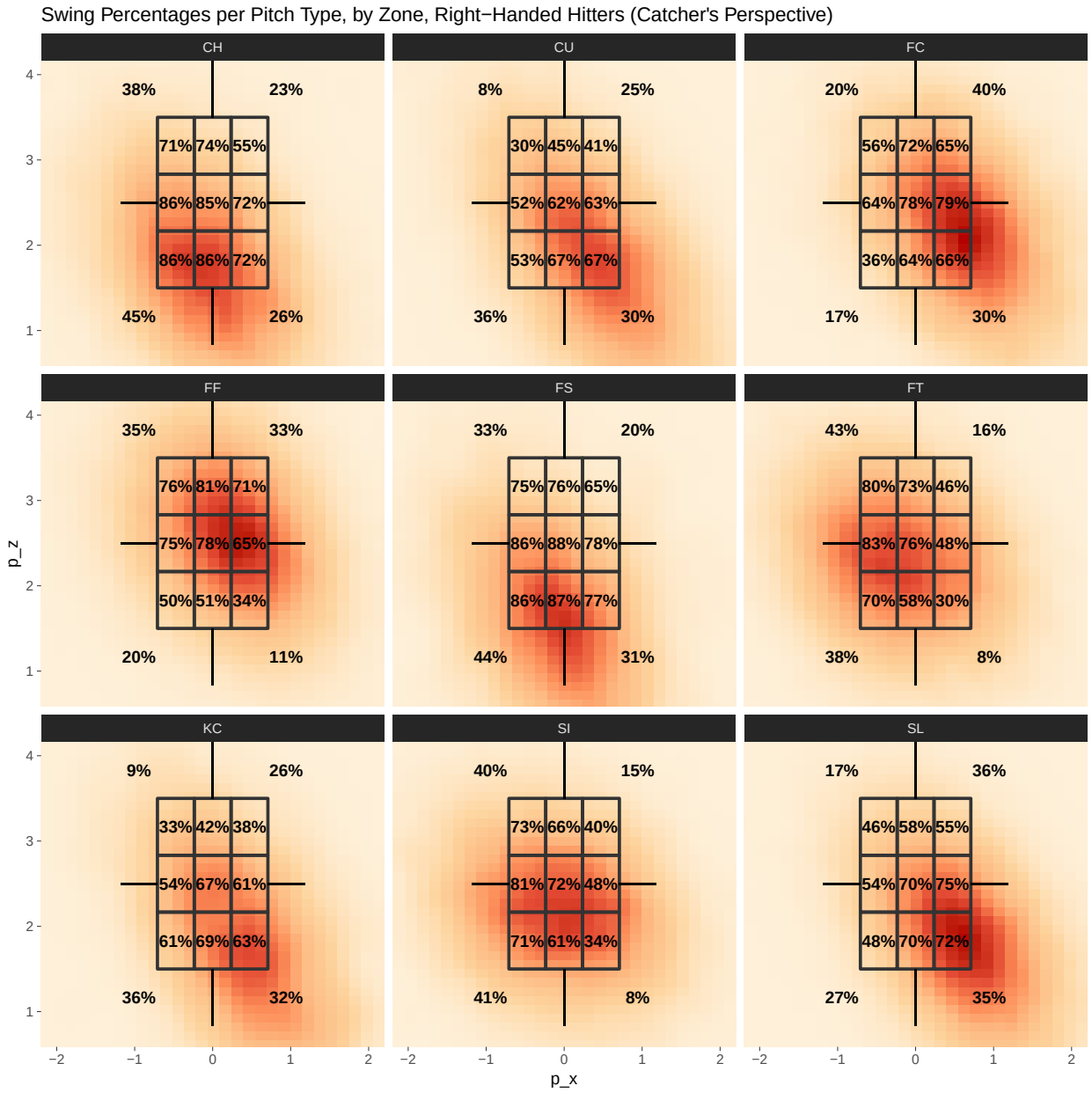
We now move on to an investigation of where in the strike zone, or the area surrounding it, each pitch

type generates swings from the batter. Initially we are looking at all swing events—misses, hits, fouls etc. We stratify the data using the variate **zone**, which gives a nice spatial breakdown of the data, and makes summarizing easier.

```
# Vectors used to specify text locations inside the zones
# They are used in the mutate/case_when below
zones_x <- c(1,4,7)
zones_y <- c(1,2,3)
corners <- c(11,12,13,14)

# Used to plot the actual zones in the Statcast coordinates
# using ggplot
test_zones <- tibble(
x_lower = rep(seq(-17/24, 17/72, length.out =3), each = 3),
x_upper = rep(seq(-17/72,17/24, length.out =3), each = 3),
y_lower = rep( seq(3/2,17/6, length.out = 3), 3),
y_upper = rep( seq(13/6,7/2, length.out = 3), 3),
)

#All descriptions that correspond to a batter swinging their bat at the pitch
swing_events <- c("foul",
                  "foul_tip",
                  "hit_into_play",
                  "hit_into_play_no_out",
                  "hit_into_play_score",
                  "swinging_strike",
                  "swinging_strike_blocked")
db%>%
  tbl("calccast")%>%
  filter(!pitch_type %in% c("PO", "FO", "EP", "SC", "KN"),
         p_throws == "R",
         stand == "R",
         !is.na(zone)
        )%>%
  select(pitch_type,
         description,
         p_x,
         p_z,
         zone
        )%>%
  collect()%>%
  group_by(pitch_type, zone)%>%
  #Calculate whiff percent for each pitch type, in each zone
  summarise(swing_prop =
             sum(description %in% swing_events
                 )/n()
            )%>%
  #Defines the coordinates for plotting the whiff percentage
  #in the center of zone to which it corresponds
  mutate( zone_x = case_when(
                               zone %in% zones_x      ~ -17/36,
                               zone %in% (zones_x+1)  ~  0,
                               zone %in% (zones_x+2)  ~  17/36,
                               zone == 11           ~ -34/36,
```

```
                              zone == 12           ~  34/36,
                              zone == 13           ~ -34/36,
                              zone == 14           ~  34/36,
                              ),
            zone_y = case_when(
                              zone %in% zones_y    ~  19/6,
                              zone %in% (zones_y+3)  ~  15/6,
                              zone %in% (zones_y+6)  ~  11/6,
                              zone == 11           ~  23/6,
                              zone == 12           ~  23/6,
                              zone == 13           ~  7/6,
                              zone == 14           ~  7/6,
                              )
          ) ->
  summarized_swings_R
db%>%
  tbl("calccast")%>%
  filter(!pitch_type %in% c("PO", "FO", "EP", "SC", "KN"),
         p_throws == "R",
         stand == "R",
         !is.na(zone)
        )%>%
  select(pitch_type,
         description,
         p_x,
         p_z,
         zone
        )%>%
  collect()%>%
  group_by(pitch_type, zone)%>%
  ggplot() +
  #A heatmap of the pitch locations
  stat_density_2d( aes( x = p_x, y = p_z, fill = ..density..),
                   geom = "raster",
                   contour = FALSE,
                   show.legend = FALSE
                 ) +
  scale_fill_distiller(palette="OrRd", direction=1) +
  coord_cartesian(xlim = c(-2, 2),
                  ylim = c(0.75,4)
                 ) +
  #This plots the percentages
  geom_text(data = summarized_swings_R,
            aes(x=zone_x, y=zone_y, label = paste0(round(100*swing_prop), "%"),
            fontface = "bold")
           ) +
  geom_rect(data = test_zones,
            aes( xmin = x_lower,
                 xmax = x_upper,
                 ymin = y_lower,
                 ymax = y_upper),
                 fill = NA, color = "grey20",lwd = 1
           ) +
```

```
# Plots the exterior lines that define the zones outside of
# the strike zone
geom_segment(aes(x = 0, y = 3.5 , xend =  0, yend = 3.5 + (2/3))) +
geom_segment(aes(x = 0, xend = 0, y = 0.8333, yend = 1.5)) +
geom_segment(aes(x = 17/24, xend = (17/24)+(17/36), y = 2.5, yend = 2.5)) +
geom_segment(aes(x = (-17/24 - 17/36), xend = -17/24, y = 2.5, yend = 2.5)) +
ggtitle("Swing Percentages per Pitch Type, by Zone, Right-Handed Hitters (Catcher's Perspective)") +
theme(legend.position = "none") +
theme_dark() +
facet_wrap(~pitch_type, nrow = 3)
```

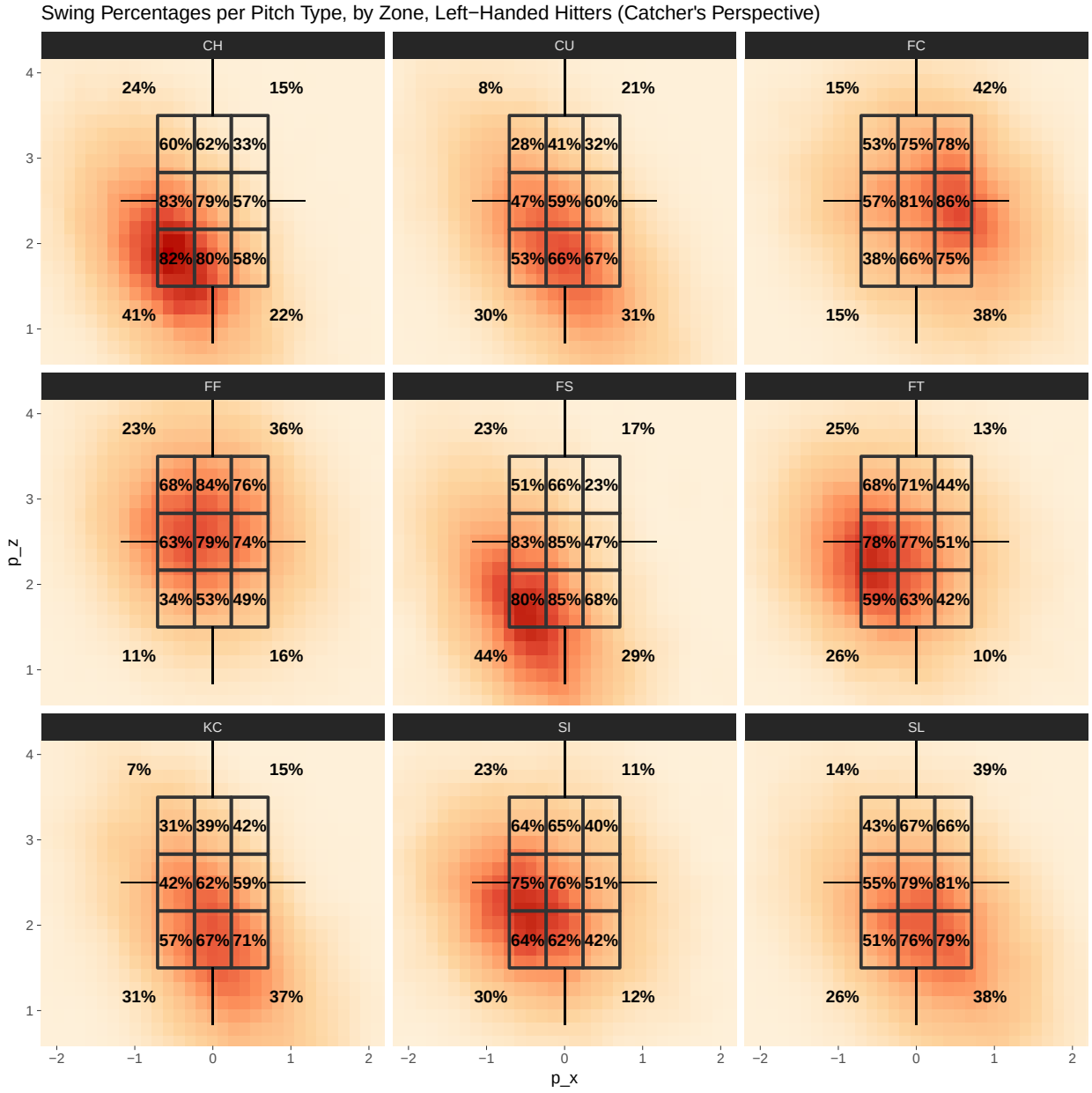


```
rm(summarized_swings_R)
```

Here we choose to add a heat map in the background of the plot based on location frequency, rather than the size of the number reported in each zone, to give an indication of how confident we can be in drawing conclusions from each of the numbers—it helps indicate which numbers have a good amount of observations to support them.

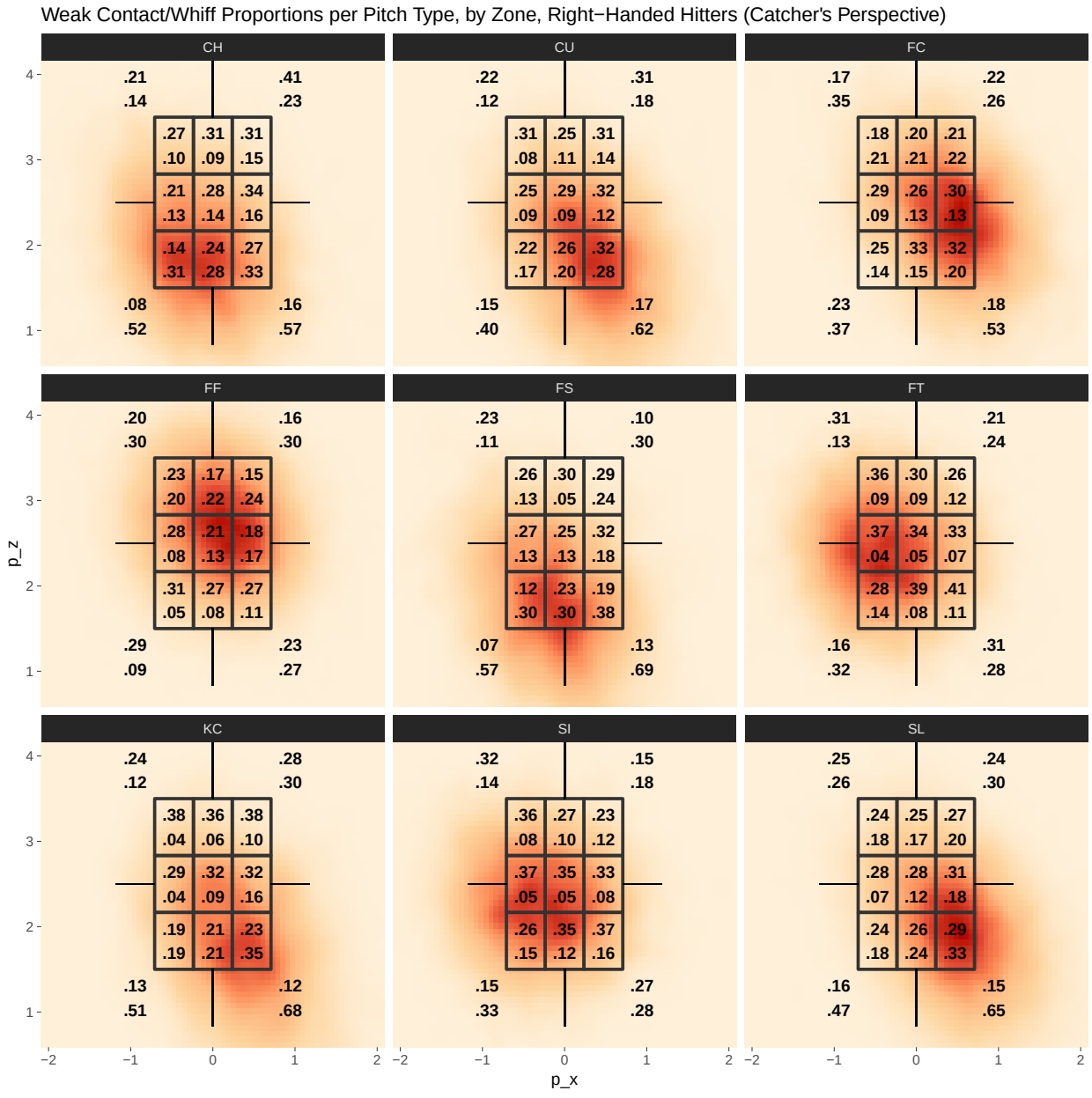
In general, we see that pitches inside the zone are swung at more often than pitches outside the zone. There are instances of highish percentages outside the zone though. For example, SL pitches were able to generate a swing 35 percent of the time in zone 14 (bottom right). KC, FS, CU, and FC pitches were able to generate higher than 30% swing rates in this zone as well. Even more insanely, CH and FS pitches were able to generate swings almost half the time they were located down and inside (zone 13). The FT and SI pitches also report high values in this zone, but their distributions in this zone seem to be concentrated a little higher above the ground. As for above the strike zone, we see FF, FT and FC generating fairly high swing percentages with regularity.

Inside the strike zone, we are interested in pitches that generate low swing rates in areas that are dark on the heatmap. This is because a pitch inside the zone that is not swung on is basically a free strike against the batter (assuming the umpire actually calls it). Some pitches that seem to fit this description are the SI and FT pitches down and away (zone 9) and middle away (zone 6). We also notice that FF pitches in zone 9 are swung on only 34% or the time, suggesting that 66% of the time an FF located here is a free strike against a right-handed batter. Next, we generate the same graph for left-handed hitters.



From the heatmap we can see that SL and CU use is down when right-handed pitchers face lefty bats, and their distributions have shifted slightly away from zone 9 more toward the zone 8. CH and FS use seems to be up, with their distributions concentrated down and away from the hitter. The distribution of SI pitches also seems to have shifted away from the batter, where before it was somewhat uniform across the middle of the zone. Outside the zone, we see high swing rates again being generated by CH and FS in zone 13, which is now down and away from the batter, and SL, CU, KC and FC generating relatively high swing rates in zone 14, which is now down and in. Swing rates on FT and SI in zone 11 have come way down, suggesting that they are not as deceptive/tantalizing to left-handed batters when placed in this zone. FF and FC pitches maintain high swing rates in zone 12, but their use in this zone has notably decreased. Inside the zone, we do not see very many low swing rate/ high pitch count combinations. The best seems to be the FF in zone 7, which is again down and away from the batter. We also might suggest that the CU and KC pitches are sneakily good up and away (zone 1), as are FC pitches down and away, but we would like to see a few more observations of these events before making this conclusion concrete.

A pitch generating a swing from a batter is not always a bad thing. In fact, sometimes this is the desired outcome of a pitch. Often, a pitcher wants a batter to swing at a pitch and miss, so as to generate a strike against the batter. A pitcher also likes it when a batter swings at a pitch but makes poor contact, resulting in a ball that is put weakly into play and easily turned into an out. Our next task is to investigate which pitch types excel at causing these types of swing outcomes. To do this, we restrict our attention to all pitches that were swung on. We then count the proportion of weak contact events, using the `launch_speed_angle` variable in the dataset.We also count swinging strike events using the `description` variate. We then graph these in the strike zone exactly like the plots above. Originally I generated two different plots here, however I have decided to present them in a single plot here in the interest of space. We create this plot for both right-handed batters and left-handed batters.



Here the heatmap gives some semblance of the relative frequency of a swing event in each zone, per pitch type. It serves as a proxy for illuminating where each pitch type looks tantalizing enough for the batter to

swing at, but does not truly represent this because these heatmaps are also dependent on where a pitcher chooses to throw each pitch type. What we are looking for is a combination of high swing rate (the previous graph), with a high relative frequency of swing events (dark on the heatmap), and a high weak contact rate or whiff rate. Areas with low swing events (light on this graph) are considered to have too few observations and are ignored.

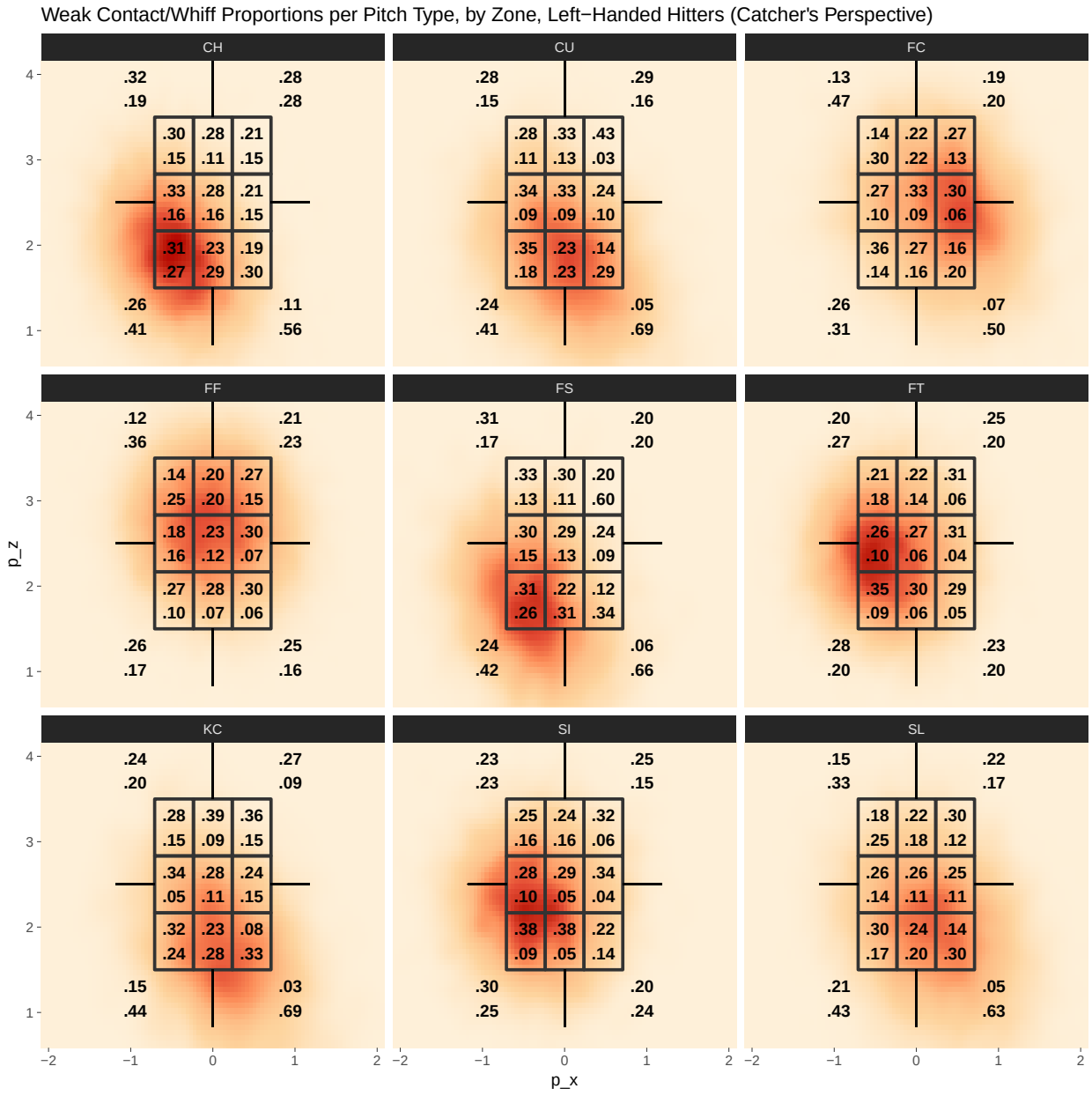
Not surprisingly, we see that pitches outside the zone are missed more frequently when they are swung at. However, some pitches did generate swings in these zones with regularity. For instance, CH and FS pitches had high swing rates in all of the lower zones, and here they are show impressively high whiff rates across these bottom 5 zones (13, 7,8,9, 14) as well. Their weak contact rates in zones 7,8,9 are not bad either, and when you add them with their whiff rates, you find that these pitches generate a desirable outcome over 50% of the time they are swung on in each of the lower zones, save for zone 7, where they both suffer rather low weak contact rates.

KC, CU, and SL saw a high frequency of useage and swing percentages in zone 9 and 14, and here they also back that up with high whiff rates. We see that thier effectiveness in generating swinging strikes is restricted to these zones however, as a shift to another zone laterally or vetically results in a noticable reduction in whiff rate. Therefore, these pitches need to be located a bit more precisely than the CH and FS pitches, if the intention is to consistently generate a swinging strike. However, all three pitches show quite good weak contact rates in the middle and lower right zones as well, so not locating them precisely in the corner is not always the end of the world in terms of a desirable outcome for the pitcher.

Recalling the called strikes plots, we may see movement pitches in zone 14 being swung on with regularity here because batters are aware of umpire bias on pitches away. This bias may be exacerbating the effectiveness of these pitches (FS, CH, KC, CU, SL) down and away from the batter outside the zone.

The FF and FC pitches had the highest frequency and swing rates in the upper zones. They follow that up here with the highest upper zone whiff rates. However, these whiff rates still pale in comparison to the whiff rates seen in the analgous lower zones. This is probably because these pitches are fairly straight and therefore are a bit easier to square up, so long as the batter can cope with their speed. Thier middle zone weak contact rates are better than their upper zone weak contact rates, but the overall sums are higher in the upper zones, suggesting that a good outcome occurs with greater probability when these pitches are place up in the zone.

As for the SI and FT pitches, they showed decently high swing rates in the upper zones as well, however, they do not follow this up with high whiff rates. This suggests that, when batters swing at these pitches, they are typically making contact. However, these pitches do possess the highest weak contact rates in areas of high swing frequency (dark on the heatmap), suggesting that these pitches are primarily used to induce weak contact. This would be the kind of pitch you would want to use if you were trying to induce a double play, for example. It is also worth noting that thier work is done in the heart of the strike zone, as well as in zone 11, areas that most of the other pitches avoid.



Overall, it seems that right-hand pitches do not generate whiffs against left-handed batters at the same rate that they do against right-handed batters. In fact, just about every whiff rate has dropped a bit in this graph compared to the analogous one for righties. One of the exceptions to this rule is the CU pitch. Although it has received reduced useage, its whiff rates in the lower five zones have increased, albeit marginally, and could just be due to random error.

Otherwise, the results are very much the same in terms of whiff rates, with FS and CH dominating the the lower 5 zones, KC, SL, and CU showing good results in zones 9 and 14, FF and FC being primarily effective in the upper zones, with FC also contributing a decent whiff rate in the upper portion of zone 14. Also, FT and SI are still not generating noteworthy whiff rates in their zones of high usage.

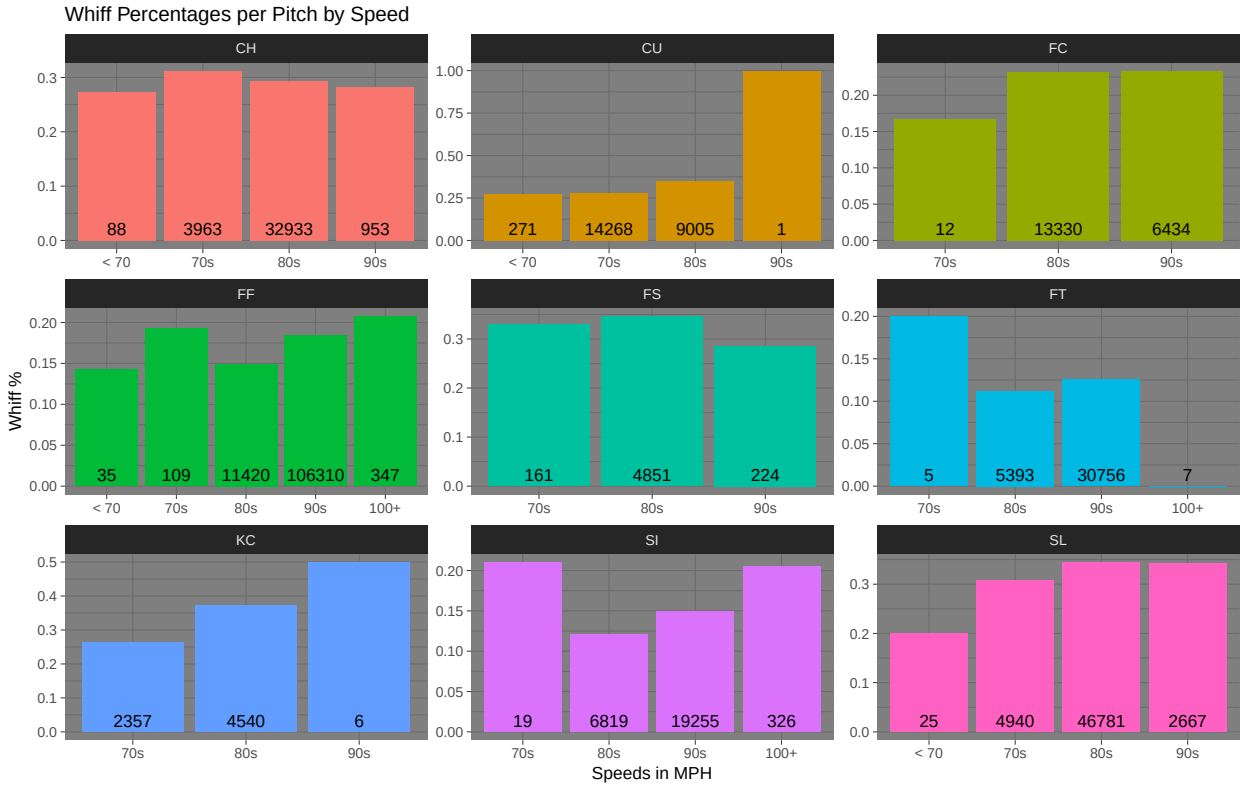
With respect to weak contact, we see a large increase in zone 7 for CH and FS pitches. This is likely due to the fact that zone 7 is down and away from a lefty hitter, and thus it is much harder for them to create good contact on balls in this zone. In fact, we do not see weak contact rate lower than .27 for any pitch in zone 7. The largest weak contact rates in high frequency areas again belong to the SI and FT pitches, and the FF and FC pitches maintain decent weak contact rates in the upper part of the zone. Finally, we create

a plot of the sum of weak contract rate and whiff rate, to give a more concrete picture of where each pitch is producing a desirable outcome in the strike zone when it is swung upon.

Certainly the above graphs demonstrate that the effectiveness of a pitch in generating a swinging strike/weak contact is heavily influenced by its location, however, we now want to know if whiff and weak contact rates are also influenced by speed and spin rate. The following plots will visualize the information and help us draw conclusions regarding this matter. Note that, because the above graphs showed that each pitch performs essentially the same function regardless of the handedness of the batter, we create the next two graphs using all of the pitch data (all pitchers against all batters).

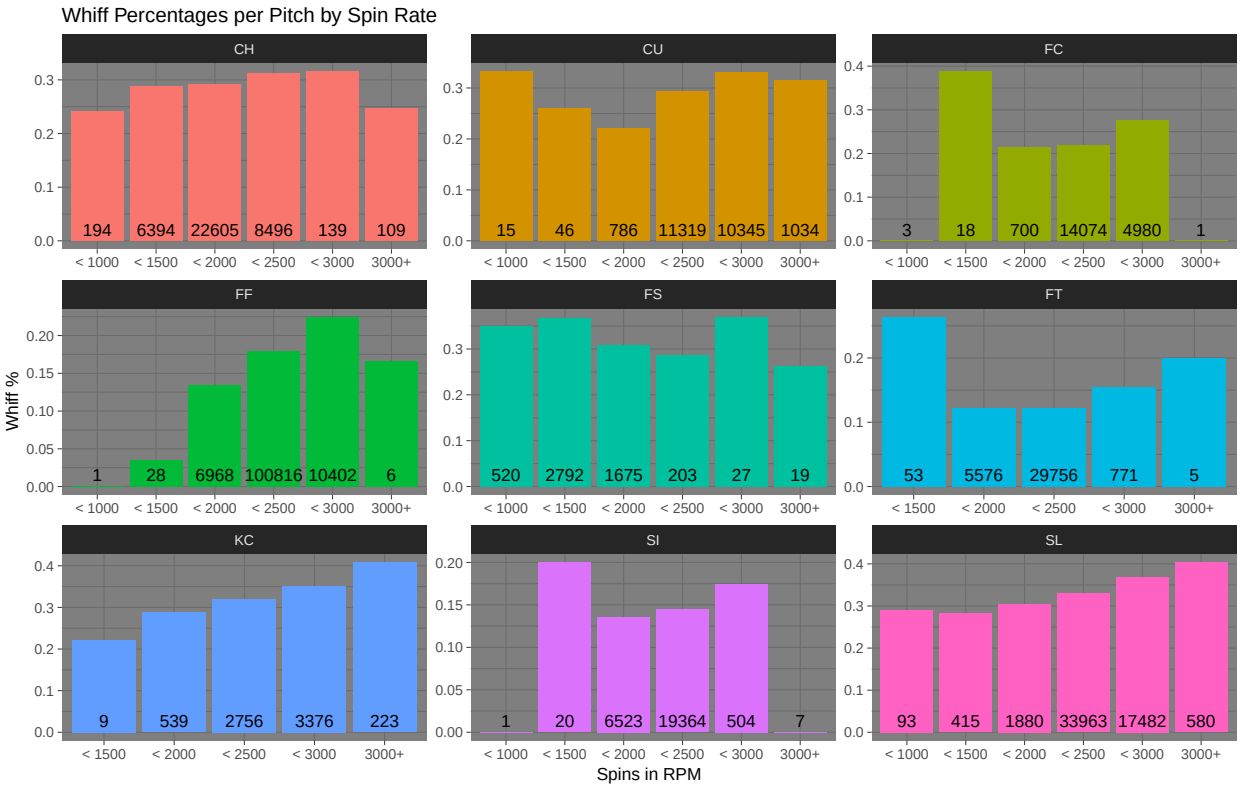
```
order <- c("< 70", "70s", "80s", "90s", "100+")

db%>%
  tbl("calccast")%>%
  filter(!pitch_type %in% c("PO", "FO", "EP", "SC", "KN"),
         !is.na(release_speed),
         description %in% swing_events
        )%>%
  select(pitch_type,
         description,
         release_speed
        )%>%
  collect()%>%
  mutate(speed_bins = case_when(
                          release_speed < 70                          ~ "< 70",
                          release_speed >= 70 & release_speed < 80  ~ "70s",
                          release_speed >= 80 & release_speed < 90  ~ "80s",
                          release_speed >= 90 & release_speed < 100 ~ "90s",
                          release_speed >=100                         ~ "100+"
  ))%>%
  group_by(pitch_type, speed_bins)%>%
  #Calculate whiff percent for each pitch type, in each zone
  summarise(whiff_prop =
              sum(description %in% c("swinging_strike_blocked",
                                     "swinging_strike")
                  )/n(),
              total = n()
            )%>%
  ggplot(aes( x = factor(speed_bins, levels = order), y = whiff_prop)) +
  geom_bar(aes(fill = pitch_type),
           stat = "identity",
           show.legend = FALSE
          ) +
  # Report the total number of swing events in each category
  # to help recognize cases with low observations that
  # may be skewing the graphs
  geom_text(aes(label = total, y = -Inf),
            position = position_dodge(0.9),
            vjust = -1.1
           ) +
  ggtitle("Whiff Percentages per Pitch by Speed") +
  xlab("Speeds in MPH") +
  ylab("Whiff %") +
  theme(legend.position = "none") +
```

```
  theme_dark() +
  facet_wrap(~pitch_type, nrow = 3, scales = "free")
```

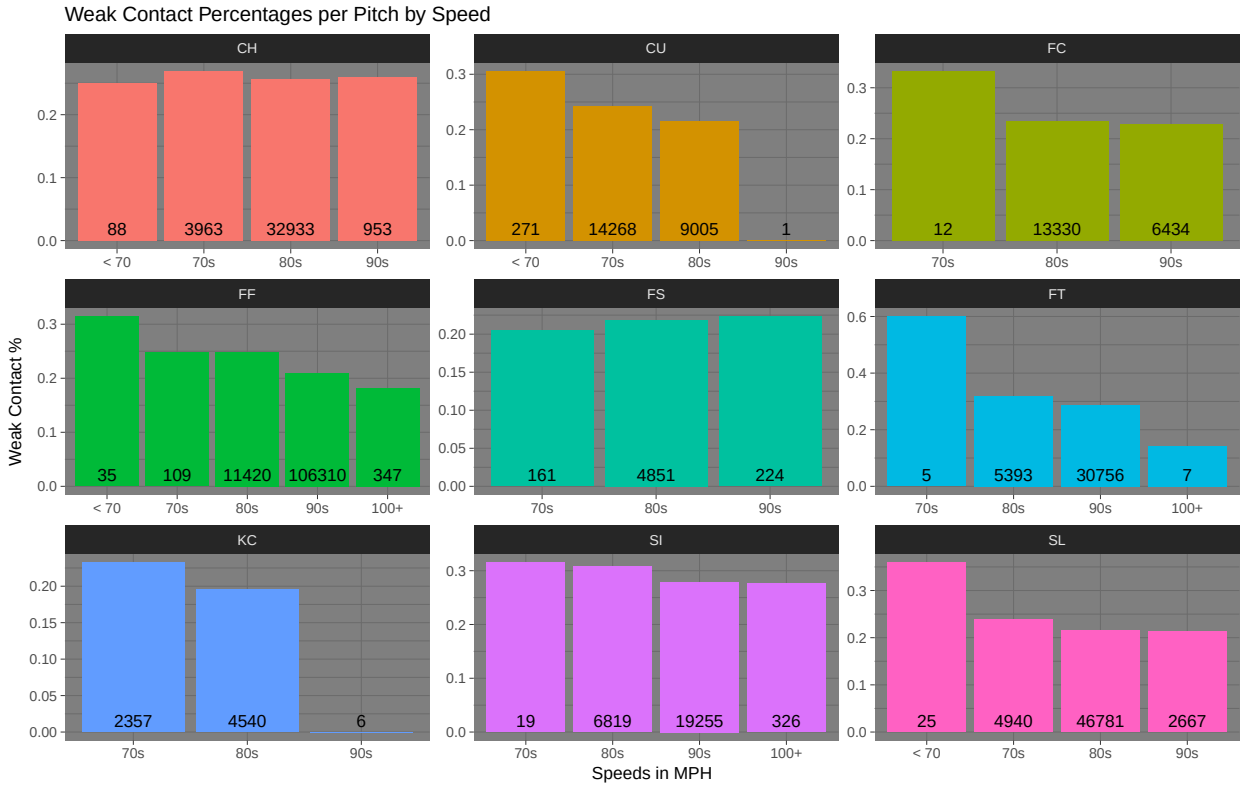


The height of each bar is the whiff percent, and the number on that bar represents the total number of swing events in that particular category. With most of the pitches we see the trend one might intuitively expect – that a faster pitch leads to a higher percentage of whiffs. Assuming that good FF pitches are at least 80mph, we also see an upward trend in the FF plot. Interestingly, we see an approximately downward trend in the CH plot. This trend actually makes sense though, as the CH pitch is an "off-speed" pitch whose deception is derived from being slower than expected. This graph possibly illustrates that throwing a CH pitch fast neuters the feature that makes it effective, leading to reduced performance in terms of whiff rate.

FS, FC and CU seem to be rather robust in performance with respect to speed, showing little variation across categories. Here we ignore the 90s category for CU pitches, because it contains only a single observation.



For just about every pitch, we see that an increase in rpms leads to an approxmiately better whiff rate. The exception seems to be the FS pitch, which shows an approxmiately decreasing trend in whiff rate with respect to an increase in rpms. One of the unique properties of the FS pitch was its distribution of relatively low spin rates. Therefore, this graph suggests that a relatively low spin rate is a key ingredient in an FS pitch if the intention is to generate swinging strikes.



Slightly paradoxically, we see slower pitches generating better weak contact. This trend is present in every plot save CH and FS, whose weak contact rate does not seem to be strongly related to speed.

```
order <- c("< 1000", "< 1500", "< 2000", "< 2500", "< 3000", "3000+")

db%>%
  tbl("calccast")%>%
  filter(!pitch_type %in% c("PO", "FO", "EP", "SC", "KN"),
         !is.na(release_spin_rate),
         description %in% swing_events
        )%>%
  select(pitch_type,
         description,
         release_spin_rate,
         launch_speed_angle
        )%>%
  collect()%>%
  mutate(spin_bins = case_when(
                      release_spin_rate < 1000                                ~"< 1000",
                      release_spin_rate >= 1000 & release_spin_rate < 1500    ~"< 1500",
                      release_spin_rate >= 1500 & release_spin_rate < 2000    ~"< 2000",
                      release_spin_rate >= 2000 & release_spin_rate < 2500    ~"< 2500",
                      release_spin_rate >= 2500 & release_spin_rate < 3000    ~"< 3000",
                      release_spin_rate >= 3000                               ~"3000+"
  ))%>%
  group_by(pitch_type, spin_bins)%>%
  #Calculate whiff percent for each pitch type, in each zone
  summarise(weak_prop = sum(launch_speed_angle %in% weak_values)/n(),
```

```
            total = n()
           )%>%
ggplot(aes( x = factor(spin_bins, levels = order), y = weak_prop)) +
geom_bar(aes(fill = pitch_type),
          stat = "identity",
          show.legend = FALSE
         ) +
# Report the total number of swing events in each category
# to help recognize cases with low observations that
# may be skewing the graphs
geom_text(aes(label = total, y = -Inf),
           position = position_dodge(0.9),
           vjust = -1.1
          ) +
ggtitle("Weak Contact per Pitch by Spin Rate") +
xlab("Spins in RPM") +
ylab("Weak Contact %") +
theme(legend.position = "none") +
theme_dark() +
facet_wrap(~pitch_type, nrow = 3, scale = "free")
```

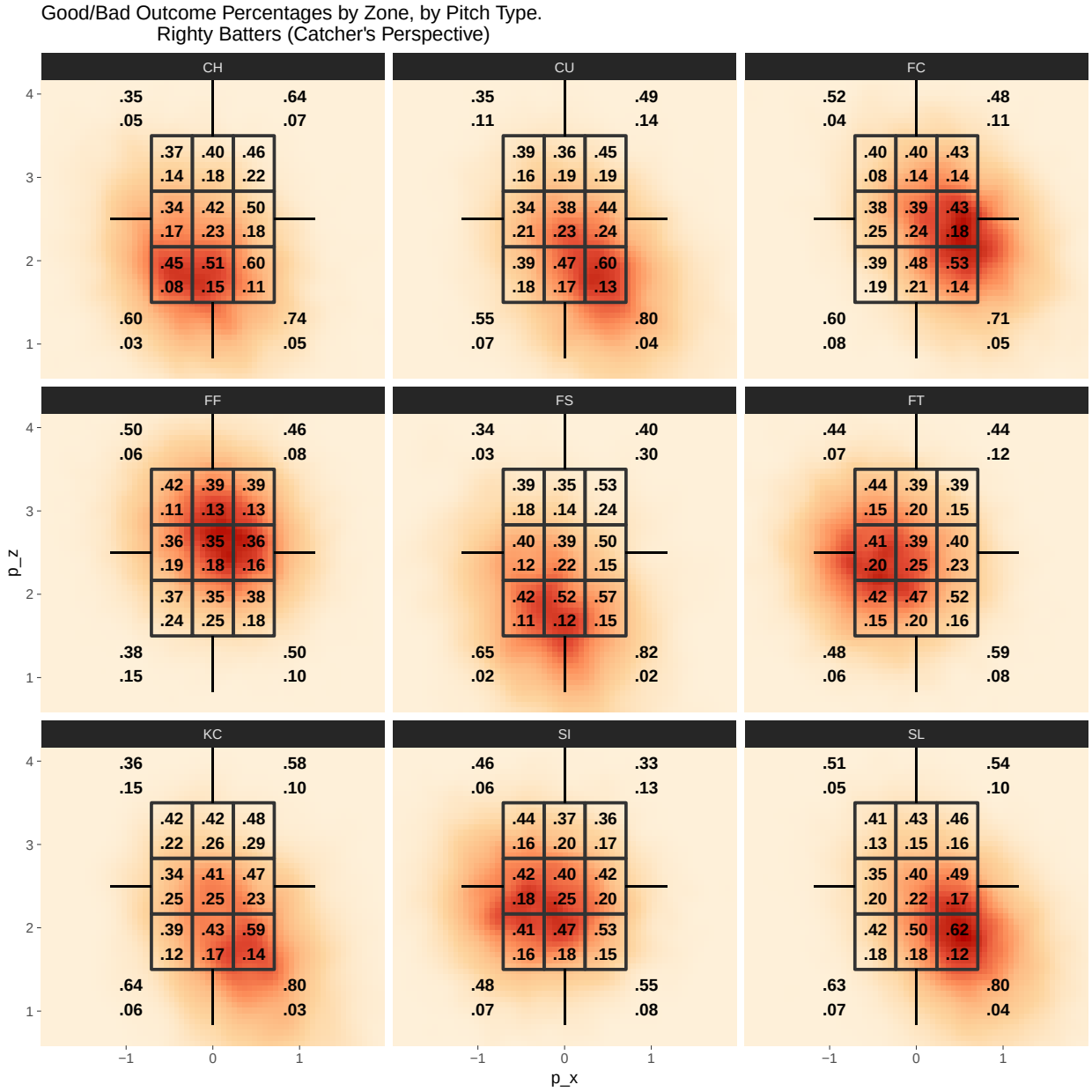
Interestingly, we also see global decreasing trends in weak contact rate with respect to an increase in rpms for each of the pitches. The steepest drop seems to be FF pitches. The exceptions to the decreasing trend are again CH and FS. Changeup weak contact rate does not seem to be overly related to spin rate, however there is a noticable jump at 3000+ rpms, where around 35 of 109 CH pitches with this rpm were hit weakly. It may be possible that extremely high rpms would improve the weak contact rates of CH pitches. The FS

pitch seems to have an optimal spin rate of between 1500 and 2500 rpms if the goal is to generate weak contact.

There is one interesting pattern we see in these four graphs. For each pitch, it appears that a slower speed with less spin generates more weak contact, and a faster speed with more spin generates more whiffs. There doesnt seem to be any combination for any of the pitches that really excels in both. That is, for whichever combination of these variables a pitch performs best in whiff rate, it also seems to perform worst in weak contact rate, i.e. the plots for whiff rate against speed and spin rate seem to show trends that are exactly inverse to the weak contact rate plots against these same variables. The exception is CH pitches with speed in the 70s mph category had the highest whiff rates and weak contact rates for that pitch.

An obvious next step here would be to look at whiff rates and weak contact rates as a function of both speed and spin rate, fitting a model or plotting some graphs in an effort to find the optimal combination of these two variables for each pitch. However, with only a single season of data, this stratification gives many strata with small sample sizes and so should really be done with 2 or more seasons worth of data in order to develop significant results.

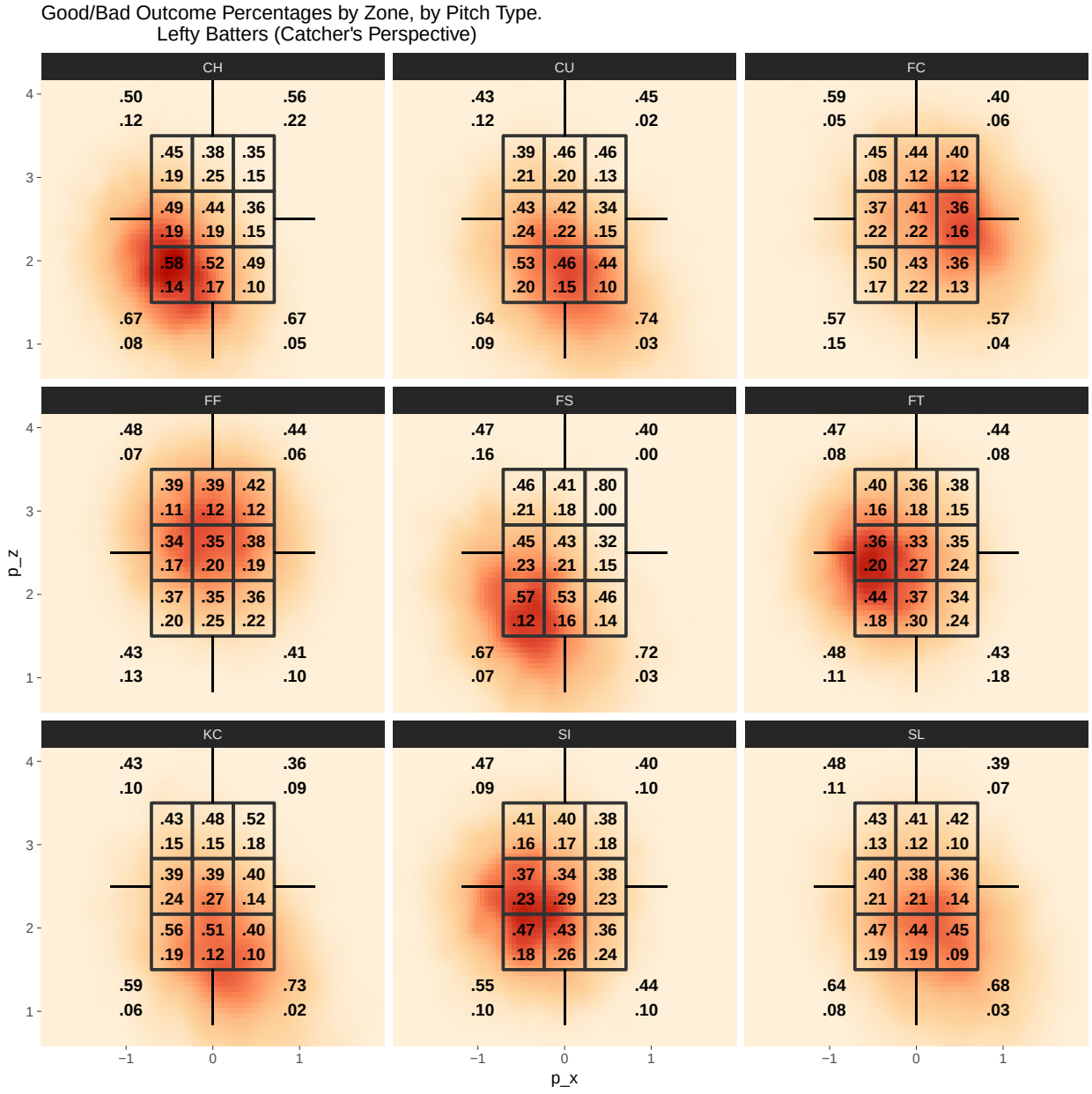
Now that we have analyzed good outcomes, we must turn attention to less desirable outcomes. We now want to know where each pitch type is getting lit up. We will use the other 3 values of `launch_speed_angle` to isolate these events. We also add the sum of the earlier two metrics, whiff rate and weak contact rate, to provide a full picture of the effectiveness of each pitch in each zone. We will call the sum of these two metrics the good outcome rate or desirable outcome rate.
We note that `launch_speed_angle` together with `description` containing `swinging_strike` exhaust all swing events, with the exception of fouls and foul tips. The issue with fouls is that they are a rather neutral outcome. A fouled pitch is good for the pitcher because it counts as a strike against the batter if the strike count is below 2. A foul ball is also good for a hitter though, because they get a lot of information about a pitch when they foul it off, which they can then use to adjust accordingly and get even closer to squaring it up the next time it is thrown. We will not remove fouls from the set of swing events, because they clearly belong there. What is not clear is how they should be measured/counted, and so they are left out of consideration in this analysis. We mention this only because the sums in the following plots are not 1, and we wanted to account for why this is the case.



The first thing we notice is that the CH and FS pitches down and inside to a right-handed batter are two of the safest pitches in baseball. In zone 7 they have the two lowest hard hit rates seen anywhere by any pitch inside the strike zone, and this is accompanied by fairly good desirable outcome rates as well. Further, when these pitches miss their spot, they are still more effective than other pitches in the same zones. For example, suppose a pitcher ends up missing his spot and placing one of these pitches up in zone 4 or 5 instead of zone 7/8. In these two zones these pitches still have equal or better hard hit rates than every other pitch, save for the FF which beats them in zone 5. If the pitcher instead misses down, we see that the hitter cannot do anything with the pitch anyway, as they are being hit extremely infrequently in this zone and back that up with high desirable outcome rates here too. Combine this with the fact that these areas are fairly dark for these pitches, and we get a picture of two very efficient pitches.

The KC, CU, and SL pitches are also fairly safe options when they are located down and away from a righty bat. Their desirable outcome rates in zone 9 are the highest seen anywhere in the strike zone, and they combine this with quite low hard hit rates as well. In the case of the CU and SL, based on the darkness of the heatmap, they are able to produce these results extremely consistently. The SL seems to be the best of

these pitches in terms of minimizing damage when its location is missed up in the zone. All three have very good performance when missed low. One other thing worth noting is that the CU and SL pitches also have good numbers across the top of the strike zone. It may benefit these pitches to perhaps see increased use up here, if it is possible. If then can be used effectively up in the zone as well, they would be two of the best performing pitches, having better zone coverage than any other pitch. In its area of heavy useage, the FF pitch shows decently low hard hit rates, but also not overly great desirable outcome rates compared to other pitches. These low numbers suggest that FF pitches generate quite a few fouls. This is likely from batters tracking the pitch well, but just not being able to keep up with its sheer amount of speed. Since fouls do generate strikes, this may be part of the strategy in use of the FF pitch, as they seem to be generate lots of swing events. Zone 1 seems to be the best performer for FF, as it produces numbers here equal to that of FS in zone 7. The FC pitch is nearly equivalent to the FF pitch in the upper zones, and the SL pitch is a uniformly better option in all of the zones where FC sees heavy useage (5,6,9,13).
The FT pitch seems like something of a gamble in the middle of the zone, with hard hit rates of 0.2 and 0.25 in its two zones of highest useage and swing frequency. This means that it is getting taken for a ride one in every 4 or 5 swings here, which is probably not desirable. These pitches seem to perform better overall when used in zone 1 and zone 7, showing increased or equivalent desirable outcome rates and simultaneously better hard hit rates. Since these zones are still inside the strike zone, it should be safe to focus use of these pitches here.



Against lefties, we again see a lot of the same. One noteworthy difference is that we see a hard hit rate of 0.3 produced by the FT pitch, which is the highest value we've seen. In fact, both SI and FT have record values for hard hit rate in zones 5 and 8. This essentially makes them a gamble in these zones, and here should probably be avoided. They do maintain good performance in zones 1 and 7, and should likely have their use restricted to these zones.

FS and CH pitches have had a slight bump in both their desirable outcome rate and hard hit rate in zones 7 and 8, with their desirable outcome values in this zone 7 now being the highest seen anywhere in the strike zone. Missing up and in with the FS has become more of a gamble, but the CH maintains good performance in these zones.

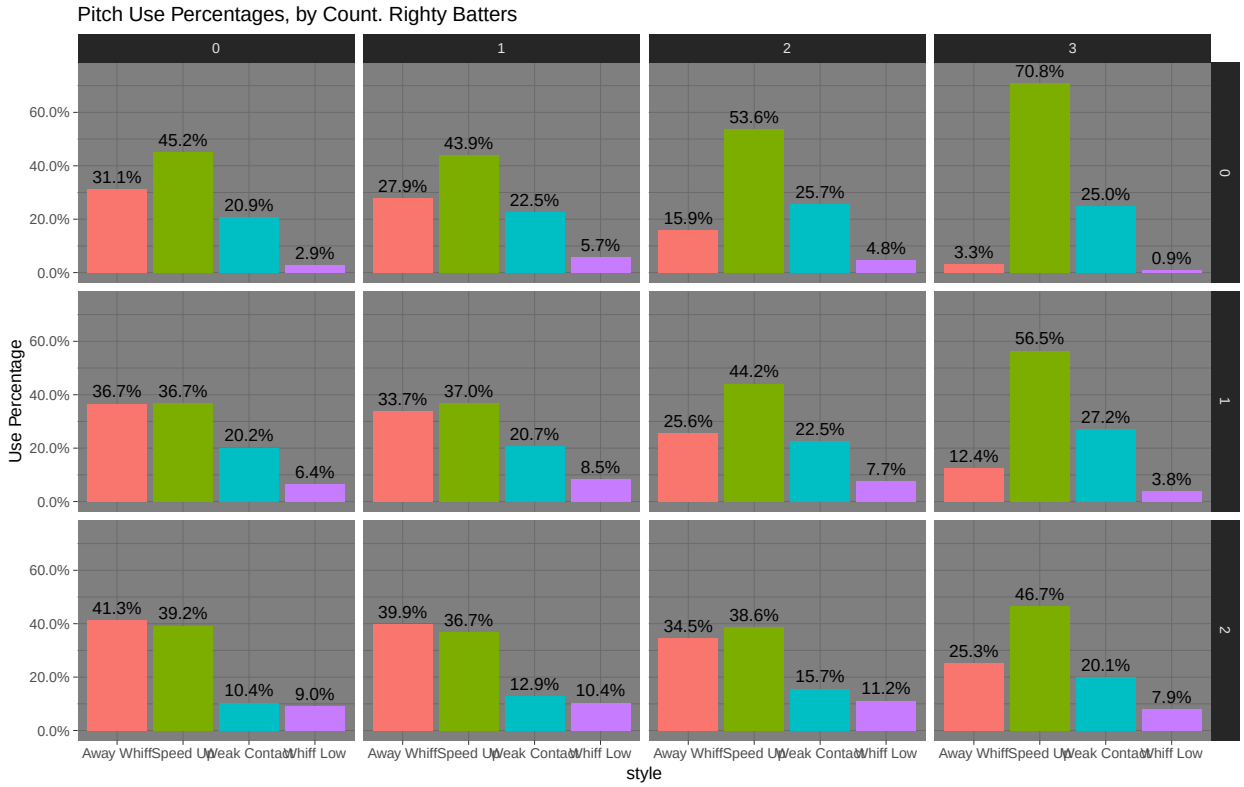
Although their useage has been significantly reduced, we see KC, CU, and SL pitches maintaining good performance in zones 9 and 14. Their hard hit rates in zone 9 are the best seen inside the strike zone on this plot. Given these results, it is a mystery why their useage has been reduced here. Pitchers are probably afraid to thrown a pitch that moves into the power band of a batter, but this dataset does not seem to suggest that these pitches are being severely punished when they miss up over the plate. For instance, it

seems that a misplaced SL or CU is still better than a well placed FT or SI. Maybe their success here is due to their rather rare useage against lefty hitters. Its possible that lefty hitters don't expect these pitches to come out from a right handed pitcher when they are at the plate, so that when the pitcher actually does decide to throw them, they are taken by surprise. It would be interesting to see how these numbers might stablize with increased useage of these pitches.

I also wanted to try and see if, when a pitch is hit in a certain location, whether or not the defense could predict where the pitch would end up. That is, I wanted to see if there was consistency in the location a certain pitch is hit, and the location it ends up at on the field. To do this, I generated a loon ggplot, with a plot of hard hit ball locations in the strike zone stratified by pitch, linked to a plot of the hit locations of these pitches. The hit locations require a transformation because they are not reported in a known metric. You can find the suggested transformation here near the bottom of the page. Unfortunately, I think there were simply too many observations, as my computer took a very long time to laod the graph and was very slow when interacting with it. I was therefore not able to get any information from the plot, however I have included the code in the rmd and R files, so that you can take a look at it if you so choose. The plot really is quite cool, and I wish I had better technology so I could have fiddled with it more. You can find it around `line 1633` in the accompanying R file.

Now that we have an idea of where each pitch is commonly used, and what its primary role is, we can begin to think about the balls and strikes count. We are interested in knowing whether or not the current count influence the choice of pitch type, i.e are different pitch types more suited to different count situations? If we were to graph the proportion of all 9 pitches for all 12 possible counts, we would get a very cluttered graph that does not provide very much information. To deal with this issue, we group pitch types together, based on the desirable outcomes that they generate most successfully. The composition of each group can be seen in the code below that generates the graph.

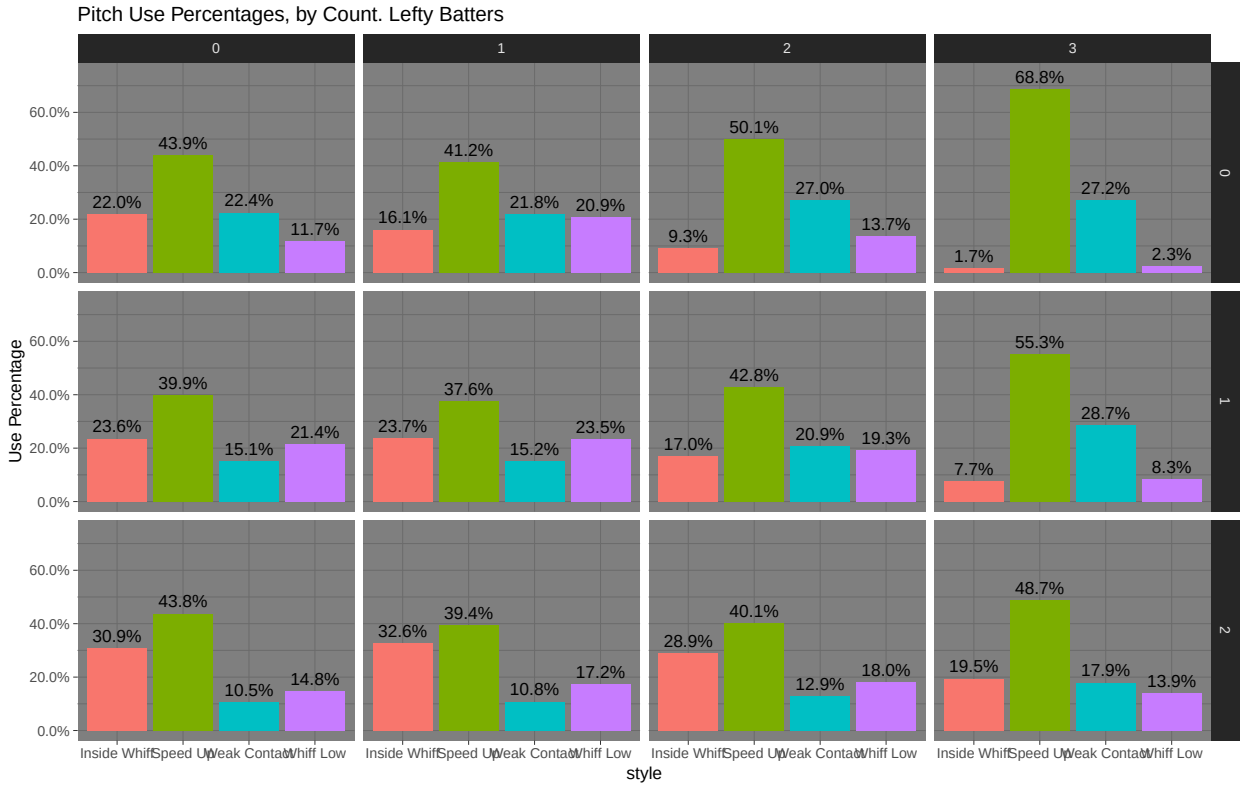
```
db%>%
  tbl("calccast")%>%
  filter(!pitch_type %in% c("PO", "SC", "EP", "FO", "KN"),
         p_throws == "R",
         stand =="R"
        )%>%
  select(pitch_type, balls, strikes)%>%
  collect()%>%
  mutate(
    style = case_when(pitch_type %in% c("FF", "FC")          ~ "Speed Up",
                      pitch_type %in% c("FT", "SI")          ~ "Weak Contact",
                      pitch_type %in% c("KC", "CU", "SL")    ~ "Away Whiff",
                      pitch_type %in% c("CH", "FS")          ~ "Whiff Low"
                     )
  )%>%
  ggplot(., aes(x= style, group = balls)) +
    geom_bar(aes(y = ..prop.., fill = factor(..x..) ), stat="count") +
    geom_text(aes( label = scales::percent(..prop..),
                   y= ..prop.. ), stat= "count", vjust = -.5) +
    #coord_cartesian( ylim = c(-Inf,0.8))+
    ggtitle("Pitch Use Percentages, by Count. Righty Batters") +
    labs(y = "Use Percentage") +
    guides(fill = FALSE) +
    theme_dark() +
    facet_grid(strikes~balls) +
    scale_y_continuous(labels = scales::percent,
                       limits = c(NA, 0.75))
```



Number of balls is across the top, and number of strikes is down the side. What we see here is something quite interesting. We see a very clear pattern in useage for all four categories. When a pitcher is "behind" in the count (faced with a higher ball count than strike count) to a righty hitter, they lean heavily on their fastball. In the extreme case of 3 balls and no strikes, we see pitchers throwing a fastball 70 percent of the time. When you include the fact that the weak contact pitches are nearly as fast as the FF and FC pitches, we see that pitchers rely almost exclusively on their fastest pitches when they are facing high ball counts. More generally, for these "Speed up" and "Weak Contact" pitches, we see that for any given strike amount, an increase in balls leads to a higher percentage in use. Given our earlier plots, this actually makes a ton of sense. When a pitcher is facing a high ball count, they know that they need to throw a pitch that is highly likely to be a called strike if it is not swung on. The FF, FC, FT, and SI pitches showed their best results inside the strike zone. Pair that with the fact that these pitches can be thrown accurately with more consistency, and you see why their useage is high in these situations.

On the flip side, when pitchers are "ahead in the count" (faced with more strikes than balls), the percentage of whiff pitch use goes up. These pitches have a lot more movement on them, and as we saw earlier, are also effective outside of the strike zone. It is likely that their extreme use in high strike/low ball counts reflects this fact, and that on these counts pitchers are throwing these pitches outside the zone on purpose. As the ball count increases, the risk of this strategy becomes greater, and pitchers begin leaning toward safer alternatives.

I think maybe the most interesting thing about this graph is the smoothness in the transitions across the plots – there are no sudden "spikes" in the graph, and nothing sees stable useage regardless of the count. In fact, it almost appears as a red/green gradient as you move from the bottom left corner up to the top right corner. This may be occurring because pitchers are people, and as such, each one makes different decisions regarding risk. A transition from one plot to another may represent a movement across risk thresholds, where the change we see in the transition represents the change in strategy of those pitchers whose risk thresholds we have crossed.



Note that we have changed the name of the group comprised of KC, CU, and SL from "Away Whiff" to "Inside Whiff" to reflect the fact that their trajectories are now toward the inside of a lefty batter. Notably, KC, CU, and SL use is down here on all counts, and FS and CH use is up, as we saw earlier in the strike zone plots. The patterns in this plot are not quite as prominent as in the previous one, however we do still see heavier dependence on "Speed Up" and lower use of "Inside Whiffs" with an increase in the ball count. The biggest change is the "Whiff Low" category, which sees its largest share on 1-0, 0-1, and 1-1 counts, rather than on 2 strike count as expected. The movement toward "Whiff Low" pitches on these counts may be due to the fact that "Inside Whiff" pitches are generally riskier against lefty hitters, and pitchers who would normally throw a KC, CU, or SL pitch to a righty batter on one of these counts are instead choosing to throw a "Whiff Low" pitch, if they possess one.

Summary and Final Remarks

In the preceding analysis, we delved into the some of the more common pitch types in Major League Baseball to try and ascertain what properties made them distinct, and what role they served in terms of optimal outcome. Here we give a brief report of our results for each pitch.

FF, the fastball, is the fastest pitch, straightest pitch, and most common pitch. This was mentioned in the original blurb, but was also confirmed by the data. It works somewhat effectively as a strikeout pitch in the upper portion of the zone, but its main purpose may be to generate fouls by overwhelming batters with blazing speed. This idea is supported by high swing rates on FF pitches inside the zone, but relative low percentages of both good and bad outcomes. It is one of the only pitches that makes use of the upper part of the strike zone. Thus, even though it is the most common pitch, its purpose and placement seem to also make it one of the most unique.

CH, the changeup, was thought to be a slower version of a pitcher's FF, but this turned out to be quite false. They are typically slower than the FF, but they generally have a lot more movement and less spin. They are most effective in the bottom half the zone, whereas FF pitches are used primarily up top. Although they are not normally thought of as whiff pitches, this dataset illustrates that they are quite good at generating

whiffs, and do so far more efficiently than FF pitches. Their best performance was in zone 7, where they generated high whiffs, good weak contact, and were almost never barrelled up. They are also used to great effect against opposite handed batters low and away, making them one of the most versatile pitches.

FS, splitters, have average speed, but possess the lowest spin tier. This combination likely gives it a bit of a "knuckling" effect. Combine this feature with the fact that the average trajectory of the FS pitch showed the greatest overall vertical movement, and you begin to understand why this pitch proved to be one of the most effective pitches at generating whiffs when placed low in the zone. It is also one of the most versatile, showing good performance in quite a few locations inside and outside the lower strike zone, and also against both kinds of batters. It sees increased used over KC, CU, and SL pitches when the pitcher faces a batter of the opposite handedness, likely due to its movement away from such a batter. The fact that this pitch is quite dominate and also one of the rarest pitches included in the analysis, makes one wonder why it is not utlized by more pitchers. Perhaps it is a difficult pitch to master, or maybe its rarety is driving its effectiveness. In any case, the overall effectiveness of this pitch was one of the many surprises to come out of this analysis.

FT, the two-seam fastball, is a bit slower than the FF, but possesses a bit more movement. One of the most surprising result with this pitch is that it is definitely not a whiff generating pitch, as I had initially thought. Instead, its main job is to generate weak contact so that the defense can make easy plays in the field. Another surprise was that this pitch was also the one most prone to being barrelled up, displaying especially scary hard hit rates when thrown against lefty hitters. This was not expected from a pitch that is supposed to have deceptive late movement. We noticed that it actually saw increased performance in the zones peripheral to its zones of highest use, suggesting that many pitchers may not be utilizing this pitch effectively.

SI, the sinker, displayed nearly identical results to the FT pitch throughout the entire analysis. In fact, there is even debate among players as to whether or not the SI and FT pitches are even different pitches at all. From the results of this analysis, we would conclude that they may indeed be nearly identical pitches. You can read more about this discussion here.

FC, the cut fastball, or simply cutter, turned out to be a bit of a disappointment. It did demonstrate good speed and lateral movement, and performed decently well in generating desirable outcomes in the upper zones, as well as zone 6 and 13. However, in all of its zones of heavy useage it was equivalent to or bested by other pitches. That is, its role in the upper zone was matched by the FF pitch, and its role in the lower zones was matched and beaten of the SL pitch. Unlike the other pitches in this analysis, the cutter just did not seem to carve out its own unique role or purpose, and it would be difficult using this dataset to argue that a pitcher might want to think about starting to throw one.

SL, the slider, CU, the curveball, and KC, the knucklecurve, seem to be three pitches that excel at primarily the same role. All three utilize high spin rates to generate curved trajectories displaying strong vertical and lateral movement. In the trajectory projection plots, they actually illustrated some of the steepest vertical movement between the point at which a batter must choose to swing and the point at which they cross homeplate. This combination of spin and curve seem to be very efficient at luring batters to swing, as these pitches generated the best overall desirable outcome proportions in zones 9 and 14, and did so with extreme consistency. Pitchers abuse this consistency in situations where they do not necessarily need to throw a "called strike" pitch, and may purposely throw these pitches a little outside the zone in hopes of generating another strike, and possibly an out. The SL proved to be the strongest overall pitch of the three, closely followed by the CU. The KC lagged behind a bit with the occassional high hard hit rate, but otherwise was nearly as effective as the others. All three pitches saw decreased use against opposing batters with handedness opposite to that of the pitcher. This makes sense from a theoretical stand point, but the data showed that lefties were not punishing these pitches as often as one might think when a pitcher actually dared to throw them. We also saw that these pitches may have untapped potential higher in the zone, if they could be thrown there consistently. We conclude that pitchers may want to try experimenting with these pitches, as it may be possible to expand their zone of use, turning them into the most versatile and effective pitches in baseball.

A few final remarks. The analysis done here is definitely just a scratch on the surface of what can be done

with this dataset. There were a million questions left unanswered, and most of the analyses were not as thorough they could have been. Still, what we did find was interesting. We found that each pitch has a pretty clear purpose in terms of the outcome a pitcher wants it to generate. We also saw that each pitch's use was quite localized in the strike zone, in that each pitch showed specific locations of heavy use. Further, for pitches with differing purposes, these locations hardly ever overlap, and together they cover nearly the entire strike zone. This certainly did not have to be the case, and was definitely not a result I was expecting. We also saw that pitchers use their pitches in different proportions based on the ball-strike count. This shows that pitchers really do understand the strengths and weaknesses of each of their pitches, and they strategically choose their pitches based on this knowledge. I think that this analysis serves as a great jumping off point, but there are definitely places that need to be expanded on. For example, we did not explore pitch sequencing at all. One question we might like to explore is whether a low pitch is more effective when it is preceded by a high pitch. We may also like to see what pitch arsenals are most common, or how pitchers are combining the different types of pitches into effective arsenals. I think the star plots in the `loon` package would be a very effective way to visualize these pitch arsenals. We mentioned that pitchers may be locating pitches outside the zone on low ball counts on purpose, because its a safer spot for the pitch and it may still generate a strike. We could investigate this by plotting pitch type locations by count. I have done this for a single pitcher, and the results were quite interesting. You could also use these numbers to fit a model that choose a pitch sequence that would simultaneously maximize strike probability and minimize hard hit probability, and see what you get. The possibilities here are almost endless, and I am excited to contine exploring this dataset even following the conclusion of this project.