Intro – Business section

Data Understanding and Preparation

To build a model to predict the next pitch type, we downloaded the MLB’s Pitchfx data from 2014, 2015, and 2016. The dataset includes every pitch from every regular season and playoff game during these three seasons. Each instance in the dataset is a single pitch. There are roughly 2.1 million records in the entire dataset.

Dataset Compilation/Web Scraping

Pitchfx is a pitch tracking system that records multiple pitch features including velocity, movement and spin rate for every pitch thrown in a game. The system is currently installed in every MLB Stadium and has been in use since 2006.[[1]](#footnote-1) Given the project aim, Pitchfx data is the perfect source for building a prediction model. There are several online sources for scraping Pitchfx data (Brooks Baseball[[2]](#footnote-2), Baseball Heat Maps[[3]](#footnote-3), PitchRx) that employ various methods and formats to return some or all of the desired information. The challenge was to find an option that combined ease of use with the most comprehensive output possible.

Initially, we tried using PitchRx[[4]](#footnote-4), an R-based package designed specifically for Pitchfx. Unfortunately, the scripts were designed in a way that prevented the retrieval of one comprehensive dataset. Even if we had been able to concatenate the **pitch** and **atbat** subsets, they still lacked a sizeable amount of the features we had earmarked for potential analysis.

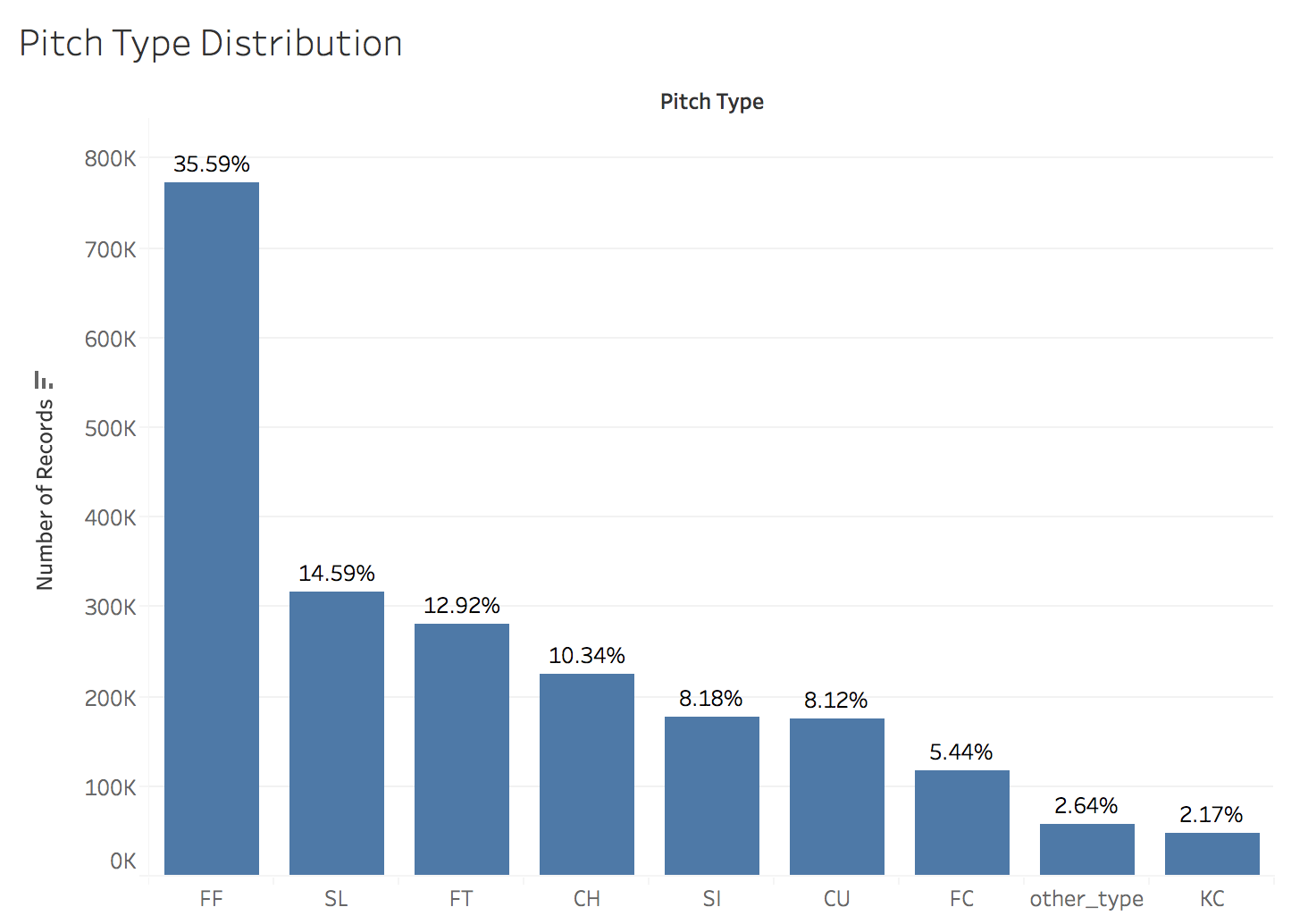
Our second approach was to entertain the idea of a customized scraper. This led us to look directly at the master archive itself, which is publicly available through MLB Gameday Directory[[5]](#footnote-5) and consists of a large number of sub-directories, the majority of which are organized by individual year, providing a hierarchical structure for each season’s data, all the way down to the pitch-by-pitch level. Information is stored in XML files, which most often represent a single inning of every game played. Navigating this complex environment would require designing a script to locate and then scrape all desired samples using BeautifulSoup, a Python library for pulling data from select file types (XML, HTML).

Before committing the time and effort to code a custom solution, we conducted one last round of research that ultimately led us to a Python-based parser/scraper[[6]](#footnote-6) that fortunately provided exactly what we needed. The scraper retrieves the features listed below and outputs them via two files (**pitch\_table.csv** and **atbat.csv**) that catalog the information on a per pitch basis:

* Game ID
* Flags
  + Spring Training
  + Regular Season
  + Playoffs etc.
* MLB Game ID
* Game Location Data
* Batter/Pitcher ID data
* Game Situation Data (balls and strikes, number of outs, inning)
* Pitch outcome sequence up to that point in the plate appearance
* Flag to designate if the pitch is the last pitch in the plate appearance
* Retrosheet-style event code

Target Variable and Features

The target variable (“pitch type”) was already given to us in the form of a two-letter string. Overall, there are eight major pitch types recorded in the MLB’s data. They are: 1) Four-Seam Fastball (FF), 2) Two-Seam Fastball (FT), 3) Cut Fastball (FC), 4) Sinker (SI), 5) Curveball (CU), 6) Change-up (CH), 7) Slider (SL), and 8) Knuckle-Curve (KC). A distribution of the classes of the target variable is shown below:



Four-seam fastballs (FF) appear much more frequently than the other classes, the majority of which range from 5% to 15%. Even though knuckle-curves comprise only 2.17% of total pitches, we decided to keep KC as a distinct class because there are a handful of pitchers who throw a lot of knuckle-curves. In other words, KC is a rare pitch, but if a particular pitcher is on the mound, KCs can occur quite frequently.

Below is a chart illustrating the features we used in our model:

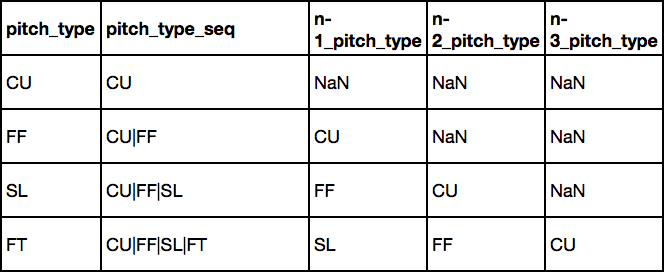
|  |  |
| --- | --- |
| **Target Variable** | Pitch Type (FF, FT, FC, SI, CU, CH, SL, and KC) |
| Game-Specific Features | * Balls and Strikes Count * Number of Outs * Pitch type of previous three pitches (n-1, n-2, and n-3) * Outcome of previous three pitches (n-1, n-2, and n-3) * Scoring Position: binary variable indicating if a baserunner is on second base or beyond |
| Pitcher-Specific Features  \*Note: for most pitching features, we used the player’s statistics **from the previous season** to avoid leakage problems. | * Pitcher-handedness (right- or left-handed) * Pitcher age * Cumulative pitch count (per game) * Strikeout Rate (Strikeouts/Batters Faced) * Walk Rate (BBs/Batters Faced) * Strikeout-to-Walk ratio * Earned Run Average (ERA) * Walks + Hits per Inning Pitched (WHIP) * Home Runs Allowed per 9 innings (HR9) * % of Batted Balls in each of the 4 categories: Ground Balls (GB%), Fly Balls (FB%), Line Drives (LD%), and Pop-Ups (POP%) * % of Pitches in each of the 8 pitch type categories: FF, FT, FC, SI, CU, CH, SL, and KC. |
| Batter-Specific Features  \*Note: for most batting features, we used the player’s statistics **from the previous season** to avoid leakage problems. | * Batter-handedness (right- or left-handed) * Position in Lineup (1 to 9) * Batting Average * On-Base Percentage * Slugging Percentage * Swing Rate (% of pitches swung at) * Contact Rate (% of swings where batter made contact) * Out-of-Zone Swing Rate (% of pitches outside the strike zone the batter swung at) * Out-of-Zone Contact Rate (% of swings outside strike zone where batter made contact) * Pitches per Plate Appearance (proxy for a batter’s “patience”) |

The features of our dataset were obtained in three ways: 1) our core dataset from the MLB; 2) feature engineering, and 3) pitching and batting statistics from the website Baseball Prospectus.

Feature “Engineering”

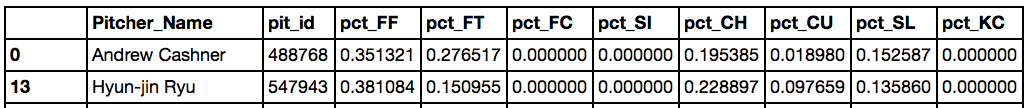
Once the dataset was downloaded, we engineered a number of features. In many cases, creating new features was difficult because of the scale of our dataset, which largely prevented us from using “for” loops. However, we still managed to create the following features:

* *Pitch Type of Previous Three Pitches*: we created three separate variables (n-1, n-2, and n-3) denoting the pitch type of the previous three pitches. To create these variables, we extracted two-letter strings from the column “pitch\_type\_sequence” provided by the MLB. Pitch\_type\_sequence tracks the sequence of pitch types for each at-bat. An illustration is provided below.

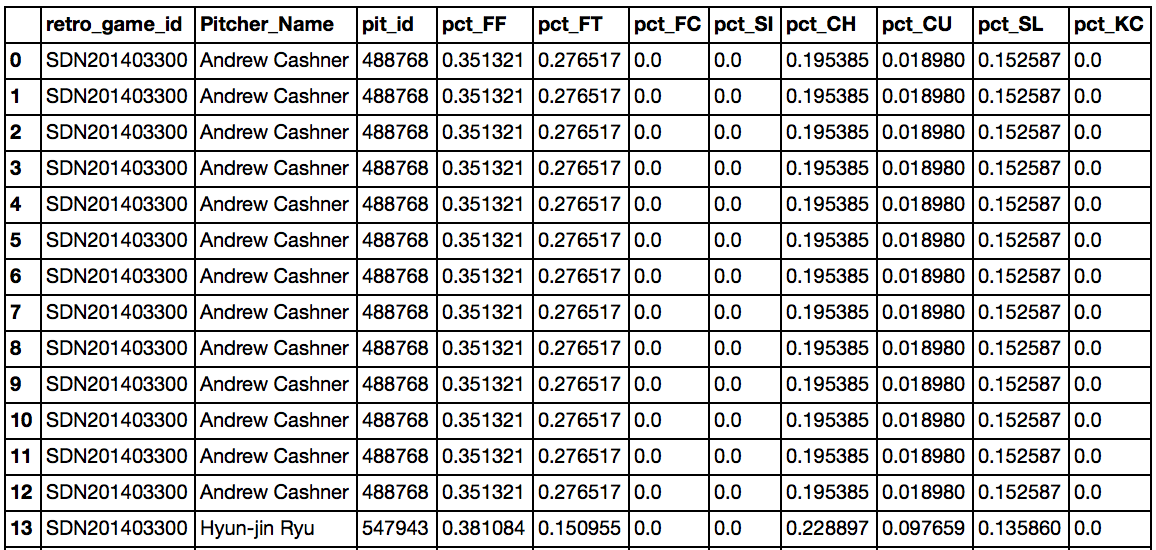


To deal with the missing values, we plugged in the pitch type “Other” – which includes all pitches that are not in the 8 majority classes. We also created three “missing” indicator variables to specify when we had missing data.

* *Pitch Result of Previous Three Pitches:* we followed a similar procedure as above, creating three separate variables (n-1, n-2, n-3) to denote the pitch result of the previous three pitches. To clarify, “pitch result” indicates whether the pitch was a ball (B), swinging strike (S), called strike (C), foul tip (F), or batted ball (X). We extracted characters from a column called “pitch\_seq”.
* *Scoring Position*: the MLB’s dataset used a bizarre metric (‘start\_bases’) for the presence of baserunners. Effectively, the MLB assigned 1 point if a runner was on first base, 2 points for a runner on second base, and 3 points for a runner on third base. The values thus ranged from 0 (nobody on) to 6 (bases loaded). To simplify matters, we decided to convert “start\_bases” into a binary indicator variable “scoring position”. In baseball, a runner is considered to be in scoring position if he is on second base or beyond. As such, our binary variable equaled “0” if “start\_bases” < 2, and “1” if start\_bases >= 2.
* *Cumulative Pitch Count:* this variable measures the number of pitches thrown by the pitcher *at the time* of his next pitch. A pitcher might demonstrate different tendencies depending on how many pitches he has already thrown in a game. To create the variable, we used the *groupby* function to partition the dataset based on ‘game\_id’ and ‘pitcher\_id’, then used the *cumcount()* function in the same line.
* *Batter Position:* This feature is a proxy for batter quality: good players are generally in the 1-5 spots, while bad players are in the 6-9 spots. To create this feature, we first created another variable (“first\_pitch”) to signal whether the pitch was the first pitch in an at-bat (introducing the start of a new at-bat). We then used the *groupby* function to partition the data based on ‘game\_id’ and ‘bat\_home\_id’ (a binary variable indicating which team was at bat). Afterwards, we used the *cumsum()* function on the “first\_pitch” column and “divided” the result by modulo 9. (e.g. 12 % 9 = 3).
* *Percent of Pitches for each Pitch Class:* For each pitcher in the dataset, we calculated the percentage of his pitches that were FF, FT, FC, SI, SL, CU, CH, and KC *in the previous year*. To build these eight variables (pct\_FF, pct\_FT, etc.), we divided our large dataset (of 2 million records) into three smaller data frames: 2014 pitch records, 2015 pitch records, and 2016 pitch records. We also downloaded the MLB’s 2013 Pitch F/x data (“2013 pitch records”). The next step was to take the unique ‘Pitcher\_ID’ codes from the 2014, 2015, and 2016 datasets. Using 2014 as an example, we took each ‘Pitcher\_ID’ in 2014 and obtained: i) the number of pitches in the 2013 data thrown by that pitcher in each pitch class (FF, FT,…); and ii) the total number of pitches thrown by the pitcher in 2013. An example of the output is below:

**

Once we had the pitch distribution for each pitcher\_ID in 2014 (using 2013 data), we merged this information to the 2014 pitch records, using an “inner” merge with “pit\_id” as the key. Effectively, we did a one-to-many merge. An example below:



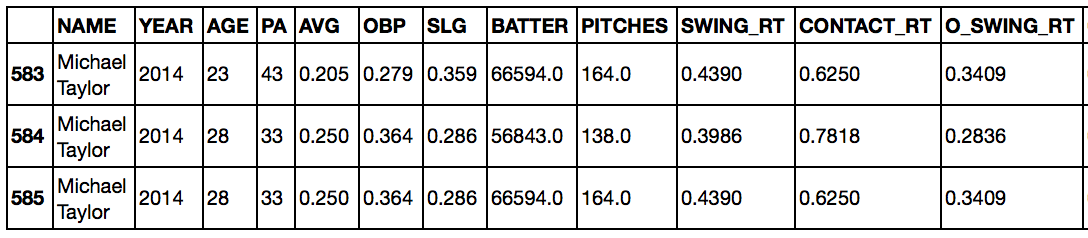
Finally, we concatenated the 2014, 2015 and 2016 pitch records to “re-create” our large dataset.

Other Pitching and Batter Statistics

We obtained several batting and pitching features from a baseball sabermetrics website called Baseball Prospectus.[[7]](#footnote-7) The site has a variety of tables for batting and pitching statistics. On the pitching side, we used the “Pitcher Season – Rates”[[8]](#footnote-8) and “Pitcher Season – BIP” (Balls-in-Play)[[9]](#footnote-9) tables. For batting, we used the “Batter Season - Standard“[[10]](#footnote-10) and “Batter Plate Discipline”[[11]](#footnote-11) tables. We collected data from the 2013, 2014, and 2015 seasons.

* *Pitcher Season – Rates*: This table contained features on the overall effectiveness of each pitcher, such as ERA, WHIP, UBBr (walks/batters faced), SOr (strikeouts/batters faced), SO/BB (strikeout-to-walk ratio), and HR9 (home runs allowed per 9 innings). We also obtained the pitcher’s age. We thought pitcher age could potentially be a significant feature, with the idea that older pitchers tend to rely on slower pitches as they lose arm strength. We were not exactly sure how the “performance features” in this table would influence our target variable (pitch type), but we had the vague hypothesis that high-performing pitchers might exhibit similar characteristics in their pitch type selection. For example, elite pitchers might be more clever in how they mix up their pitches than below-average pitchers.
* *Pitcher Season – Balls-in-Play:* This table included features on the percentage of batted balls against each pitcher that were: 1) ground balls; 2) fly balls; 3) line drives; and 4) pop-ups. We were motivated to obtain these features because in baseball parlance, pitchers are frequently described as “ground ball pitchers” or “fly ball pitchers”. Ground ball pitchers tend to rely on pitches with substantial downward movement, such as sinkers (SI) and two-seam fastballs (FTs). Fly ball pitchers, on the other hand, tend to throw high fastballs to induce weak upward contact.
* *Batter Season – Standard*: This table contained basic batting statistics such as batting average, on-base percentage, and slugging percentage. We speculated that pitchers might use different pitch types against good hitters and bad hitters.
* *Batter Plate Discipline:* This table included features on the hitter’s tendencies during an at-bat. Important features included Swing\_Rate (% of pitches swung at), Contact\_Rate (% of swings that result in contact), O\_Swing\_Rate (for pitches outside the strike zone), and O\_Contact Rate (for pitches outside the strike zone). Pitchers are likely to pitch more aggressively to hitters with a high swing rate. An “aggressive” pitching strategy usually involves throwing pitches outside the strike zone to lure impatient hitters into swinging at an errant pitch. Against hitters with a high swing rate, pitchers might rely on pitches that are difficult to control (such as a knuckle-curve (KC)). However, if the batter rarely swings at pitches outside the zone, the pitcher will probably choose pitch types with higher levels control (such as a regular four-seam fastball).

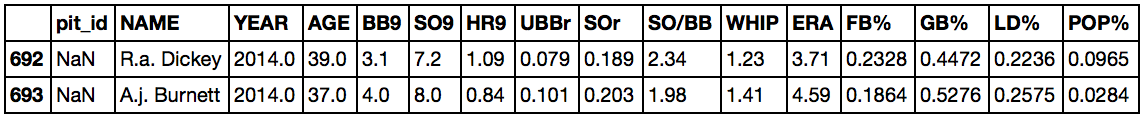
We wish we had obtained the data using an advanced scraping method, but we relied on brute force here, copying and pasting all of the information into Excel. We had three years’ worth of data (2013-2015) and four tables for each year, meaning 12 Excel files altogether. Next, for each year, we merged the two batting tables and we merged the two pitching tables. This step was tricky because the only available key to merge on was ‘NAME’, and there were several players who shared the same name. (One of the batting tables had playerID, but the other batting table did not). To resolve this issue, we first conducted an “outer” merge to see if there were any duplicated names.[[12]](#footnote-12) An example from 2014 is below.



After finding the duplicated names, we had to go back and designate one of the duplicated names with an underscore (e.g. Michael Taylor\_1) in the Excel file to distinguish between the two players. Luckily, there weren’t too many cases where this needed to be done, and we were extremely careful while making these modifications.

Once this step was complete, we had to link the Baseball Prospectus player IDs (denoted as ‘BATTER’ in the example above) to the MLB player IDs used in our original dataset. To accomplish this, we downloaded an online csv file mapping the Baseball Prospectus to the MLB IDs.[[13]](#footnote-13) Linking the Baseball Prospectus batting data to the MLB IDs was relatively simple because we had player IDs for the batters.

However, linking the Baseball Prospectus pitching data to the MLB IDs was much more difficult, because Baseball Prospectus did not include PlayerIDs in the particular pitching tables we chose. As a result, we needed to merge with “Name” as the key. This was problematic, however, because the MLB and Baseball Prospectus often provided slightly different player names. For example, the MLB uses “R.A. Dickey” while Baseball Prospectus uses “R.a. Dickey”. To resolve this issue, we employed a similar strategy as above, using an “outer merge” to find examples where the MLB and Baseball Prospectus names were not identical – in particular, when the column “pit\_id” was empty.



In this case, we would have gone back to the original Excel files and edited the names above as “R.A. Dickey” and “A.J. Burnett”. This editing process was tedious but necessary.

Finally, once this process was done, we split our large, 2-million record dataset into 2014, 2015, and 2016 data frames. We then took the 2013 batting data and merged it to the 2014 records using an inner merge with “bat\_id” as the key. We employed a similar merge between the 2013 pitching data and the 2014 records, this time using “pit\_id” as the key. We repeated this procedure for the other two years, and then concatenated the 2014, 2015, and 2016 records back into one large data frame.

Other Data Manipulation

**Game Type Description**

The raw dataset includes a feature to describe the exact type of game that produced each individual sample: *Regular Season, Wild-Card Game, Divisional Series, League Championship Series (LCS), World Series, Spring Training* and *Unknown*. In order to keep our analysis limited to official games and lower the number of NaNs, we disregarded the pitches thrown in the *Spring Training* and *Unknown* categories*.*

**Creating Dummy Variables**

After cleaning the dataset, it was necessary to perform some feature engineering on a group of seven k-class variables in order to run our baseline Logistic Regression model:

* Pitch Type
* Batter Position
* Months
* Year
* Number of Outs
* Ball Count
* Strike Count

The required k-dummy variables were generated using the **One Hot Encoding** module.

**Normalization**

While many of our features were between 0 and 1 (mostly percentages), we had seven additional variables that needed be expressed on a scale of 0 to 1. The features below were normalized using **minmax**. Compared to other normalization techniques, minmax provides a result with smaller standard deviations, which can suppress the effect of outliers.

* Pitcher Age
* Cumulative Pitch Count
* Home Runs Surrendered Per 9 Innings (HR9)
* Strikeout to Walk Ratio (SO/BB)
* Walks + Hits per Innings Pitched (WHIP)
* Earned Run Average (ERA)
* Pitches Per Plate Appearance

1. <http://www.fangraphs.com/library/misc/pitch-fx/> [↑](#footnote-ref-1)
2. <http://www.brooksbaseball.net/> [↑](#footnote-ref-2)
3. <http://www.baseballheatmaps.com/> [↑](#footnote-ref-3)
4. <http://cpsievert.github.io/pitchRx/> [↑](#footnote-ref-4)
5. <http://gd2.mlb.com/components/game/mlb/> [↑](#footnote-ref-5)
6. <http://www.beyondtheboxscore.com/2015/9/24/9374949/a-new-python-based-pitchf-x-parser-scraper> [↑](#footnote-ref-6)
7. <http://www.baseballprospectus.com/sortable/> [↑](#footnote-ref-7)
8. Pitcher Rates: http://www.baseballprospectus.com/sortable/index.php?cid=1928886 [↑](#footnote-ref-8)
9. Pitcher BIP Stats: http://www.baseballprospectus.com/sortable/index.php?cid=1819106 [↑](#footnote-ref-9)
10. Batter Standard: http://www.baseballprospectus.com/sortable/index.php?cid=1918875 [↑](#footnote-ref-10)
11. Batter Plate Discipline: http://www.baseballprospectus.com/sortable/index.php?cid=1858217 [↑](#footnote-ref-11)
12. When you do an outer merge, and there are two players with the same name, you get some weird results. Say you have player A (‘Tom Jones’) and player B (‘Tom Jones’), and two data frames X and Y. The merged dataset will have three records with the name ‘Tom Jones’. One record will have player A’s records from X and Y, a second record will have player B’s records from X and Y. The third record, however, will combine player B’s records from X and player A’s records from Y. [↑](#footnote-ref-12)
13. 'http://www.baseballprospectus.com/sortable/playerids/playerid\_list.csv.

    We then used the *pd.io.parsers.read\_csv* function to download it into iPython. [↑](#footnote-ref-13)