I decided to break the data into 4 entities: game, batter, pitcher, and pitches. I also created a “seed” table for the “pitch call codes” or “pitch\_result”. Using the Rails terminology, where “belongs\_to” implies that that table contains the foreign key, the pitch entity belongs to a game, a pitcher, and a pitch result. A pitcher has\_many pitches. For the most part, with this data set, the game, batter and pitcher entities are little more than grouping points for pitches. The pitches table is where the PitchFX data is stored.

1. After the data import, I ran this query, thinking it should identify duplicate pitches:

select

pitch1.id, pitch2.id,

pitch1.game\_id, pitch2.game\_id,

pitch1.batter\_id, pitch2.batter\_id,

pitch1.pitcher\_id, pitch2.pitcher\_id,

pitch1.pitch\_result, pitch2.pitch\_result

from

pitches pitch1

inner join pitches pitch2 on

(pitch1.id != pitch2.id and pitch1.game\_id = pitch2.game\_id and pitch1.pitch\_per\_game = pitch2.pitch\_per\_game

and pitch1.inning = pitch2.inning and pitch1.top\_of\_inning = pitch2.top\_of\_inning and

pitch1.pitcher\_id = pitch2.pitcher\_id);

There are 20 records, each with "pitch\_per\_game" being 0. Most of the records have different pitch results. Once those are excluded, it appears there are no duplicates. I don't know what to make of a "pitch\_per\_game" being 0.

2. Attempting to determine the precise meaning of pitch\_plate\_location\_x and \_z:

My thoughts with X are that 0 is right down the middle of the plate. Since the plate is 17 inches across, that means the horizontal strike zone values should be between -8.5 inches and +8.5 inches (the stat is given in feet, so conversion is necessary).

There are approximately 40,000 pitches that were not swung at where the absolute value of x was greater than 8.5/12 feet. Approximately 32,000 of them (80%) were called balls; a little over 600 (1.5%) were intentional balls, pitchouts, or resulted in a hit batter. The remaining were called strikes.

Out of 348 hit batter outcomes, all but 2 had an absolute X value greater than 1. Hit lefties typically result in a positive value. Hit righties all result in a negaive value. I'm a little suspicious about the low

called ball rate from the previous paragraph, but it seems reasonable to conclude that 0 is right down the middle of the plate.

3. I really wish I had had the Y data for pitch\_plate\_location. I would have been very interested in doing a strikezone scatterplot.

4. I chose to keep the existing IDs from the spreadsheet for pitchers, batters, and games. In hindsight, I didn't use the batter entity all that much. During the import of data, if a table has a regular primary key (an incrementing integer of some size), then the database will provide the ID when the record is created. I did not want to have to save the record to get the entity id and hold it for later use; this would likely require importing record by record (and entity by entity within each record). Instead, I chose to read the CSV entirely, create the model objects in memory,and use the ActiveRecord-Import gem

functionality on the collection of objects (which is much faster than calling the ActiveRecord create method repeatedly).

5. I imagine that importing via pure SQL would probably be faster than instantiating Ruby objects and saving them, but it seemed more error prone. Even on my "production" virtual machine, which is running on an older computer with more limited capabilities, an import doesn't take longer than 10 minutes. My approach to performance is to make sure the behavior is correct first, then refine performance if requirements force it.

6. Breaking out the pitch\_result into a separate table, with a number of boolean attributes, allows for easy filtering of outcomes when joined to the pitches table. Rather than explicitly list all the codes that indicate, say, a strike (and risk error), one can simply join the selected pitches to pitch\_results and add a 'pitch\_results.strike = true' criteria to the 'WHERE' clause. Need to narrow it further to swinging strikes (whiffs)? Replace the WHERE with 'pitch\_results.swinging = true and contact = false'.

7. I used the Highcharts Javascript library for the line charts (Pitches per Game and Velocity). The pitch summary table was done with raw ERB/HTML. The pie chart (Pitch Breakdown) was done with the Rails Kickchart library, which itself uses the Google Charts Javascript library. Highcharts is only free for personal non-commercial use, so in a production environment with the Royals this would have to be considered.

8. Besides activerecord-import, Highcharts, Google Charts, and kickchart, other libraries I used were "Faker" (for fake pitcher/batter names and fake game dates), "will\_paginate" for paginating big collections, "pry" for debugging, and the Blueprint CSS framework.