***Using Apache Arrow to make an MLB Statcast Database: in Python and R***

***The Explanation:***

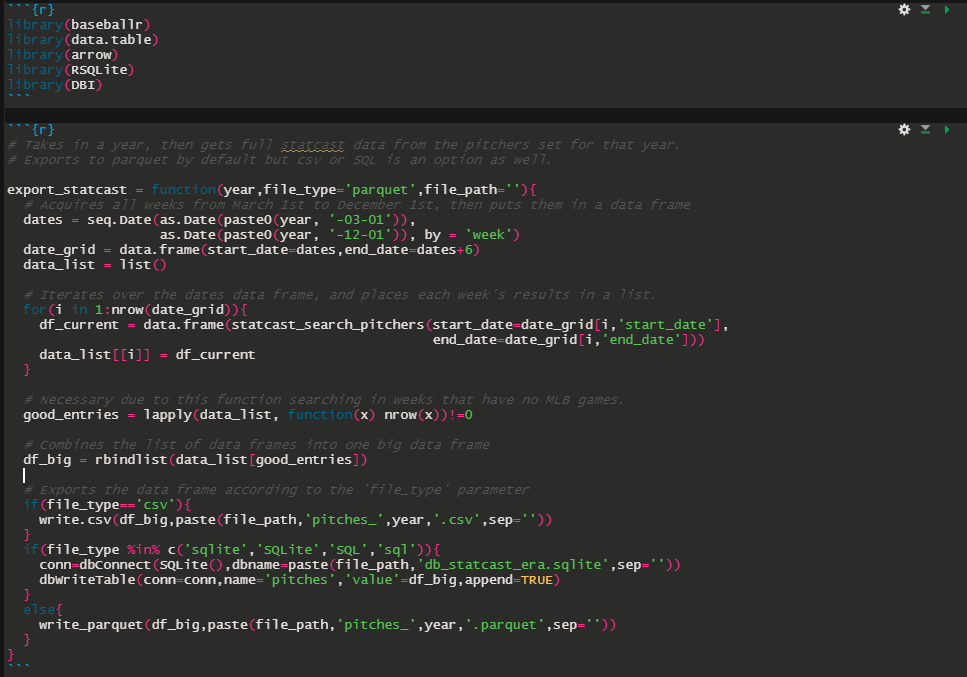
The [baseballr](https://billpetti.github.io/baseballr/) package in R, and the [pybaseball](https://github.com/jldbc/pybaseball) package in Python are godsends for anyone working with MLB statcast data. They scrape the various helpful baseball datasets and present it in a useful tabular format. This data even includes information for every single pitch thrown and the results of that pitch that wouldn’t be included on the back of a baseball card, like the spin rate of a pitch or the launch angle of a batted ball.

The trouble with this wealth of information is that there are quite a few pitches thrown in an MLB season, even any given week can have nearly 30,000 pitches thrown which coincides to 30,000 rows for just one week out of an MLB season. This can be troublesome very quickly given the hardware on your machine and your willingness to cooperate with longer runtimes when doing analysis with your data.

Enter the parquet file format, developed by Apache. Parquet is a table style file format which contains data very similarly to the ubiquitous csv format. The difference lies in the fact that parquet is a columnar stored format where csv is row based. Due to this, compression is much friendlier since entries in a column are going to be much more similar than entries in a row. Furthermore, it’s far easier to take a peek into a columnar dataset since most of the time you’ll want to be looking at only certain columns rather than every single column as you would in a csv file.

So how do we work with a parquet file? Apache Arrow is the platform to use. Libraries for Arrow exist in about a dozen popular programming languages including those in this report: Python and R. The functions included in these libraries are everything you could want: reading and writing data as well as some very fast operations on very large datasets. Sadly, aggregation is not yet supported within Arrow outside of memory so no “group\_by” or “summarize” calls can be made. But selecting specific columns and filtering can be done on truly massive datasets far faster than pandas or dplyr operations.

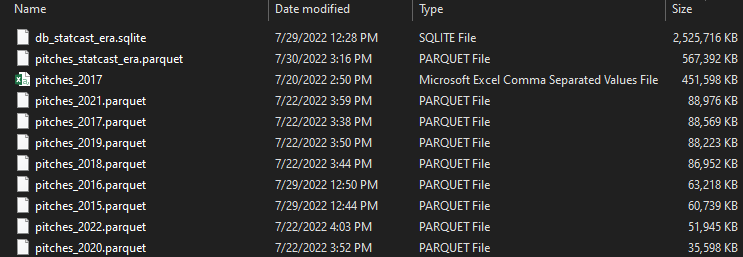
***The Code:***

******



Above, we can see the two “export\_statcast” functions: the top one, in dark mode, is written in R with the baseballr package, and the bottom one, in light mode, is written in Python with the pybaseball package. In order to acquire the statcast database on your local machine, simply run a for loop from 2015 to 2022 (or whatever year it happens to be when you read this article) and your programming language of choice will begin exporting data from each pitch straight to the file path of your choice (working directory, by default) from the start of the statcast era all the way to the present day.

***The Comparisons:***



The screenshot above illustrates perfectly the benefits of the parquet format when it comes to storage. All of the files here are parquet files aside from the full SQL database and the one csv that was made for the year 2017. When organized by size, as in the screenshot, it’s easy to see the inflation caused by the csv format. At third place, behind only the SQL and parquet files of the entire statcast era, is the csv file for just 2017. Compared to its parquet counterpart, the csv is over 5 times as large.