Andrew Ehrenberg
SI 330: Final Project Report (**see Jupyter Notebook for more explanation**)
December 14th, 2018

Introduction:

Weather conditions are an impactful element of outdoor sporting events. Inclement weather may hinder athletes from performing to their fullest ability - in a sport like baseball, offensive and defensive players are affected differently by adverse conditions - this effect could potentially translate to their statlines. When I identified a large Major League Baseball game log dataset that contained statistical breakdowns for every game in the last 50 years, I knew I could use this to build a model against the weather in the home city that day. As an avid baseball fan and former player, I was interested to see if inclement weather actually had a significant impact on elements of the game.

Originally, I sought to answer the question of whether offensive or defensive players were more strongly impacted by certain weather conditions. However, this would require me to operationalize the 'quality' of offensive and defensive performance - I attempted to do this, but didn't think I had time to build an effective model. Instead, I decided to explore weather's effects on baseball statistics more broadly - *which baseball statistics are most affected by weather? What relationships do these variables have (and which ones do they not have)? Do pitchers walk more hitters when it is very cold or very hot? Does 'good' weather (**much** easier to define than 'good' baseball performance) improve offensive statistics?* - These were questions that I would be able to answer with my existing dataset without excessive extrapolation.

Data Sources:

- ***Retrosheet Major League Baseball Game Log (1958-2016)***
  - 90,000+ rows, 160 columns (~125 MB), CSV document
  - Contains data pulled directly from the game box score, broken down by home and visiting team
  - ***Google Drive Link: (too large to preview, just download)*****:** http://goo.gl/8rx6ow
  - ***Format:*** CSV format
  - Since this is a local csv file, this can be directly read in using pandas
  - Used a DataFrame of only around 40k rows of the data (starting from 2000) and only used around 3,000 rows in my analysis
- ***WorldWeatherOnline Past Local Weather API***
  - 500+ requests allotted per day (was able to secure a premium free trial)
  - 10+ years of historical data available (was able to aggregate around 1 year of data across all of the cities/games I was accounting for)
  - Very easy-to-use interface - a query is as simple as a city name, a start date and an end date (in my case, start and end date were the same day)

- ➢ API response can be requested in JSON format and contains hourly data in 3 hour blocks, which provided me with additional granularity in identifying the time of day the game was played.
- ➢ Weather variables I focused on for my analysis were temperature, precipitation, wind, and humidity - I felt that these were the most important variables in determining the overall weather conditions (in terms of conditions that might affect an outdoor sporting event) at any given time

  Link to description and documentation:
  https://www.worldweatheronline.com/developer/api/historical-weather-api.aspx

Data Processing Steps:

- ❖ **Selecting the appropriate amount of data for the purposes of the analysis**
  - ➢ Because there is several decades of data in the game log dataset, I decided it was better to start with a subset of this and extend it if need be (the shortened dataset still goes back to 2000 and I only got weather from 2015-2016, so this proved to be more than enough)

- ❖ **Transforming the baseball log data to a format in which we can build the weather API request given the city, date, and time of day.**
  - ➢ Step 1: Changing the park codes to city names
    - ■ The original baseball logs only identifying feature in terms of location is the park code - this code corresponds to an additional text file provided with the dataset called 'park_codes.txt' - this file allowed us to map each of the park codes to a corresponding city name.

- ❖ **Set up a caching pattern to save the data from previous API requests**
  - ➢ Seeing as I had a 500 query limit each day for the Weather API, it was important that I saved the results that I had previously gathered so that I could continue building out my dataset.
  - ➢ I ran into some issues with overwriting my cache and losing data, but was able to recover most of it in the end. I learned my lesson that I should *always* back up my cache before running anything that utilizes one (you will see that there is a 'cache_backup.json' in my submission file as well, in case something were to happen while the grading team was trying to run the code).
  - ➢ I wrote a function called get_with_caching (described in the notebook as well) that takes both the request parameters and the cache data as input and returns saved cache data while querying for and caching unsaved/new requests.

- **Final data transformations prior to calling the get_with_caching() function**
  - Convert raw dates to the proper datetime format
    - The API requires a datetime as one of its input parameters, so providing a raw date would not work. Using the strptime and strftime functions, I was able to create the proper datetime object needed to pass into the API request.
  - Convert Day/Night code to hour code
    - The WorldWeatherOnline API returns hourly weather data in 3 hour blocks (0-2100), so day games would be around 12pm (aka block 4) and night games would be around 6pm (aka block 6). For the sake of simplicity, coded all of the Day/Night codes from the original baseball dataset based on this rule.
  - Map cities onto dates
    - The best way to pass this data to the weather API was by iterating through dates and then through the cities that hosted a game on that day - so keys are dates and values are a list of cities
    - This allows us to look at the weather in a number of different cities for each target date and gradually start working backwards

- **Construct a dataset within the bounds of my API usage limits**
  - Given how many cached queries I already have and how many requests I have already made in a given day, I can extend my dataset by up to 500 entries every 24 hours. If I passed the entire DataFrame to my get_with_caching function, I would get blocked from the API nearly instantly, so being careful about tending my my limits was very important throughout the project.
  - In order to pass in the appropriate amount of data, I created a new DataFrame called sample_df, which contains the maximum number of entries I can query for at that time.

- **Putting it all together: making API requests and getting data from the cache**
  - Here, I make an iterative request that goes through every single date in the dictionary. This is a nested for loop that iterates through each date key and then every city in the list mapped to that date. In the second for loop, I am then able to make a request for a specific date and city. As you can see, WorldWeatherOnline has a very usable interface - I don't even need to geocode my data (to lat/long) - I simply provide a city name, a start date and and end date as input and it returns a detailed hourly weather response
  - *Why is the data put into a dictionary first rather than a DataFrame?*

- I chose to do this because I had to iterate through a dictionary in order to make one request at a time. Iteratively appending to a DataFrame cannot be done very well Pythonically, so I thought that working with Python data structures was more compatible with making a call to the requests library. By using a dictionary, I can easily iterate through cities and dates, creating a unique key for each (city, date) tuple, whose value is the portion weather API response pertaining to the time of day the game was played.

❖ **Creating the new DataFrame and constructing columns**

- ➢ Instead of dealing with merging this with the original baseball DataFrame and having to deal with 160 columns of data during the analysis portion, I created a new DataFrame that will serve as the skeleton for the data pipeline, which can then be populated to best suit our needs.
- ➢ First, as mentioned above, I must iterate through all of the dictionary keys to add additional rows to each column. I accomplish this through several list comprehensions corresponding to each column of weather data.
- ➢ These list comprehensions form lists of data entries corresponding to a particular column. This can then be passed directly into the DataFrame - I just need to provide a column name.

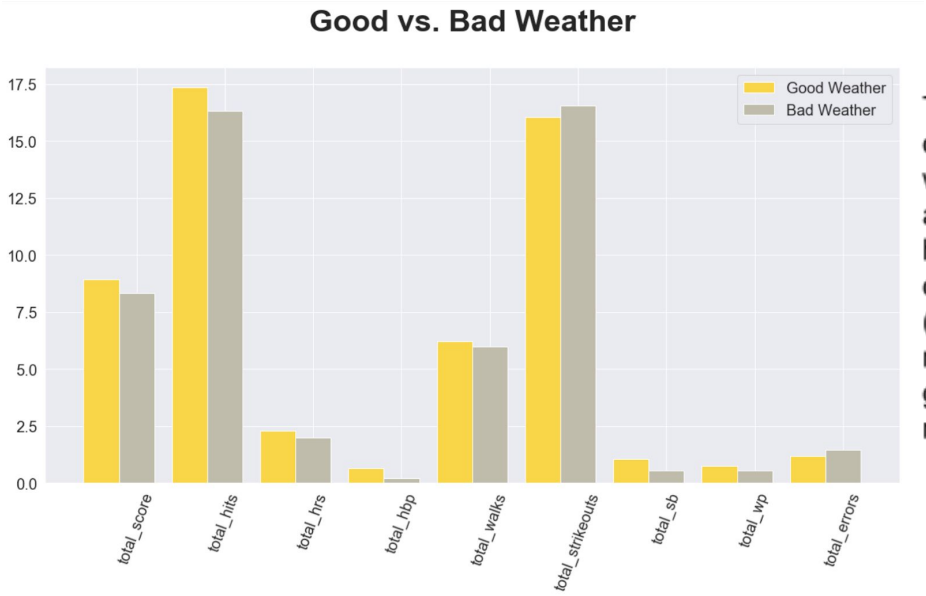
❖ **Data Categorization - classifying 'levels' of various weather output**

- ➢ In order to make powerful comparisons between varying weather conditions and MLB baseball games, segmenting certain weather variables into 'levels' makes it easier to classify a certain combination of weather conditions as 'good' or 'bad'. I approached this by constructing mini-functions with a single-purpose of classifying a single column of data.
- ➢ These various levels are a great way to organize my analysis by looking at different classifications of weather variables in relation to baseball data. Instead of having to classify each time I want to examine a variable, I can do these calculations in the dataset before I visualize anything or run any statistical tests

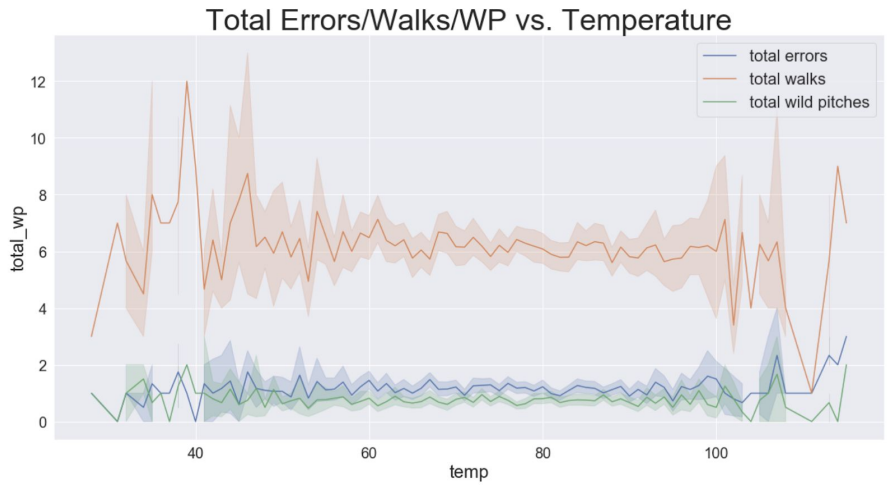
❖ **Populating and organizing the baseball data**

- ➢ There is way to do this without dropping all the home and visitor columns at the end, but it was much easier to do a column-wise operation to derive the totals and then delete the home and visitor columns. For the purposes of this analysis, we don't really care about home vs. visiting team - we are more interested to see how the weather affected overall performance. (which affects the statlines of both teams, which is why it is reasonable to focus on total stats rather than team or player-specific ones)

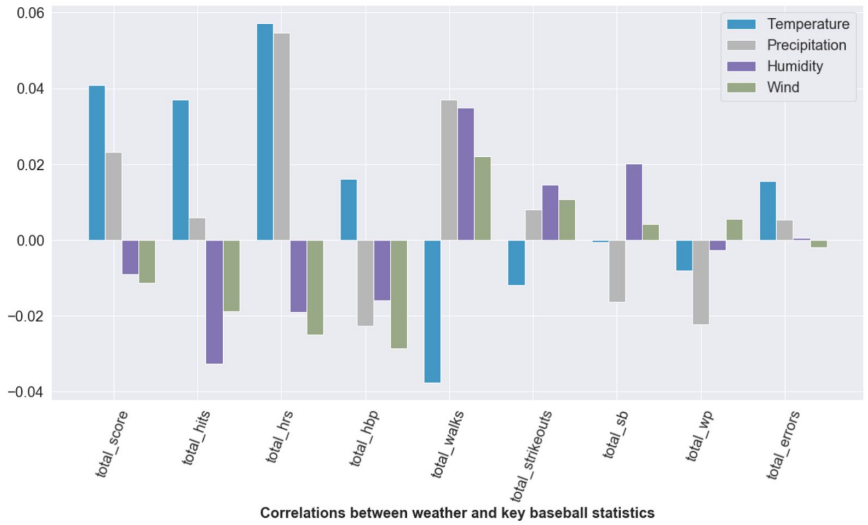
RESULTS (more results and analysis included in the Jupyter Notebook):



There are some noticable differences in some of the mean values between good and bad weather games. Good weather games have about a +1 edge in both total score and total hits. This makes some intuitive sense, as offenses should benefit from better weather (but defenses do as well!) There are also noticably more stolen bases in good weather games. Bad weather games have slightly more errors and strikeouts.



Instability at lower temperatures and higher temperatures with relative stability in the middle. While much of this instability could be explained by increased comfort in moderate weather, this may also have to do with the fact there there are less games played at extreme temperatures, which make variations in the data more noticable.



Correlations between weather and key baseball statistics

Although there are no particularly strong correlations, the matrix and bar graph below give us an interesting sense of which relationships are strongest (on a relative scale). These correlations (albeit weak) tend to make logical sense within the context of the game. For instance, the strongest positive correlations are for total home runs for temperature and precipitation. From a temperature standpoint, this makes logical sense, as hitters are looser in warmer conditions.

Discussion and conclusion:

This was a really interesting project for me. As a lifelong baseball guy, I thought I would have a pretty good sense of the impact weather has on the game. As it turns out, it is harder to predict outcomes and/or performance based on weather conditions than one might think - some of the common-sense patterns are readily visible, but obtaining statistically significant results on something like weather - inherently part of the game - is very challenging. I also learned a lot about manipulating data with pandas and how it can be used to build a combined data model.

I would have done a lot more with more time. Although I started early, this project eventually took a backseat to other finals once I got far enough ahead on it. I was not able to create a classifier for good/poor offensive or defensive performance - this would have been important in answering my original question of whether different conditions gave an advantage to the offense or the defense. I also would have done more to manipulate the baseball data - I spent most of my time looking at variation in the weather dataset when I also could have approached the problem inversely and constructed distributions and models based on specific criteria concerning the baseball stats rather than the weather conditions.

There were so many ways to look at this problem it was almost overwhelming. I definitely want this to be a project that I continue to perfect and understand better.

For a more detailed explanation of some of the data processing steps and the actual data analysis results, look at the markdown cells in my Jupyter Notebook, which presents my code, my tables, and my visualizations if a readable way.