Data Domain:
The data I am using for my interactive visualization is from the Lahman Baseball Database (http://seanlahman.com/download-baseball-database/); it is the same data I used for my exploratory data analysis. For this assignment, I constrained the data to only include seasons from 2010 to 2019. Downsizing the number of records in my data to only one decade serves the purpose of more fluid comparisons across seasons, as there were no major rule changes or shifts in gameplay, along with allowing all the data to be shown and processed without crashing any computers.

MLB data from the 2010s is accurate and well maintained. It is a timeframe during which I closely followed baseball myself, which gives me a strong understanding of the distribution of the data and what users might be interested in exploring. Beyond removing entries outside of this time period, I also removed data variables that were unnecessary for the purpose of my application (such as player birthday) to optimize the speed at which everything is processed. Furthermore, I added some data variables myself that represent baseball statistics which can be calculated from data variables already in the tables. Finally, I combined data tables into more readable, concise files for d3 to process, which are described below:

Teams.csv
Data entries describe total statistics for a team in a given season, including both pitching, hitting, and basic fielding stats. There are 300 records and 34 data variables:

- *Nominative*: lgID, teamID, franchName
- *Quantitative*: yearID, G, W, L, R, H, 2B, 3B, HR, TB, BB, SO, SB, CS, SB%, HBP, SF, AVG, OBP, SLG, OPS, RA, ERA, CG, SHO, SV, HA, HRA, BBA, SOA, DP
- *Ordinal*: Rank
- *Boolean*: WSWin
-

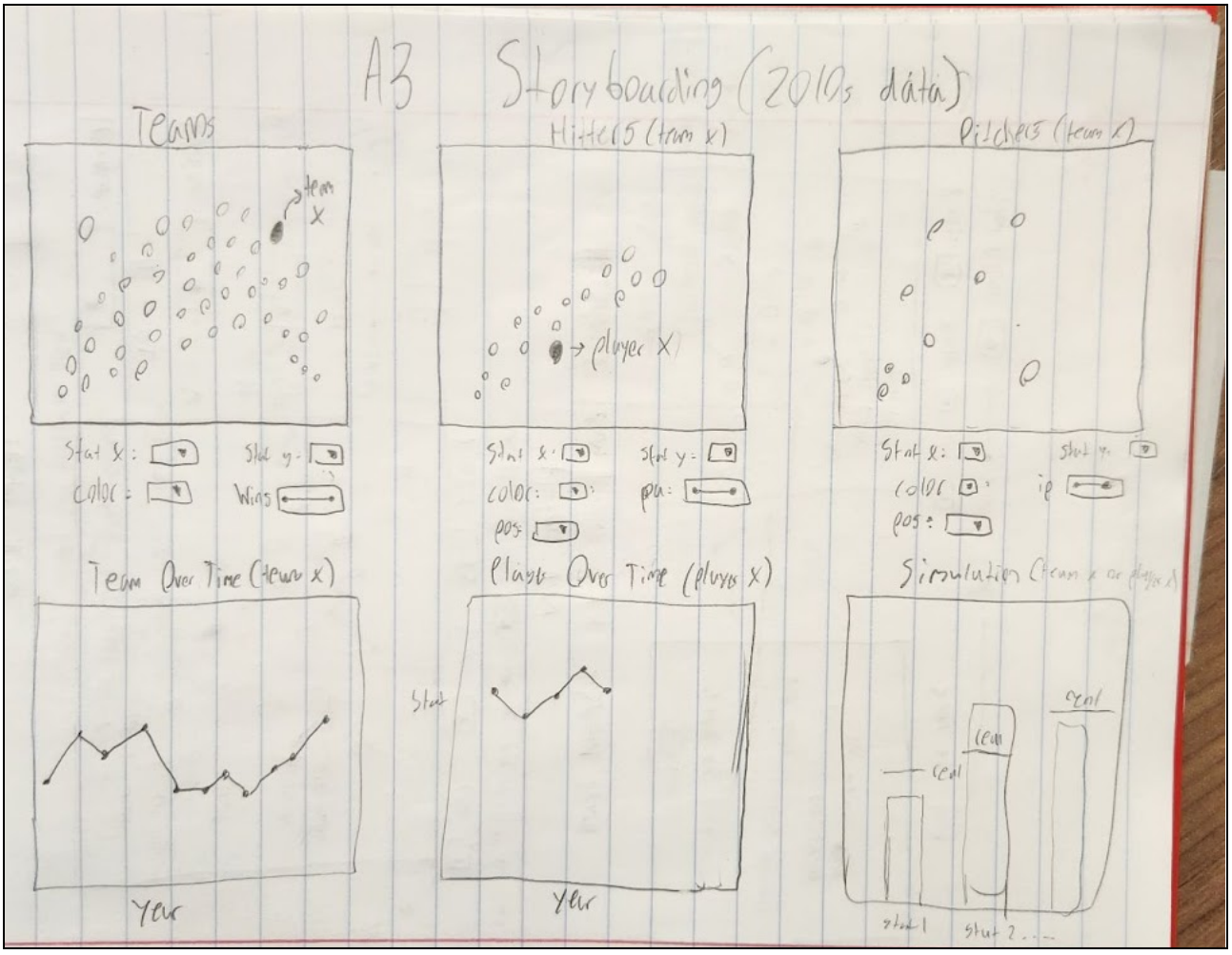
Hitters.csv
Description: Data entries describe statistics of an individual season for a given hitter. There are 6825 records and 31 data variables:

- *Nominative*: playerID, teamID, nameFirst, nameLast, position
- *Quantitative*: yearID, G, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, SB%, BB, SO, TB, AVG, OBP, SLG, OPS, IBB, HBP, SF, weight, height
- *Ordinal*: N/A
- *Boolean*: allstar

Pitchers.csv
Data entries describe statistics of an individual season for a given pitcher. There are 7867 records and 28 data variables:

- *Nominative*: playerID, teamID, nameFirst, nameLast, position
- *Quantitative*: yearID, W, L, G, GS, CG, SHO, SV, H, ER, HR, BB, SO, BAOpp, ERA, HBP, GF, R, weight, height, IP, WHIP
- *Ordinal*: N/A
- *Boolean*: allstar

Overall Storyboard:
I based my storyboarding on a combination of what I've learned throughout this class and what I, as an avid fan of baseball and baseball statistics during the 2010s, would want to explore. I started by drawing the initial idea for the layout of the application:



It consists of 6 views. There are 3 scatterplots, 1 for teams, hitters, and pitchers respectively, on the top row. The bottom row contains two timelines (line charts), 1 for teams and 1 for players (hitter or pitcher), and a discrete outcome bar chart simulation on the right. My rationale for the overall layout is to allow the user to see multiple important facets of analyzing baseball statistics at a time: correlations between metrics, comparison to peers, trends over time, and sustainability of performance.

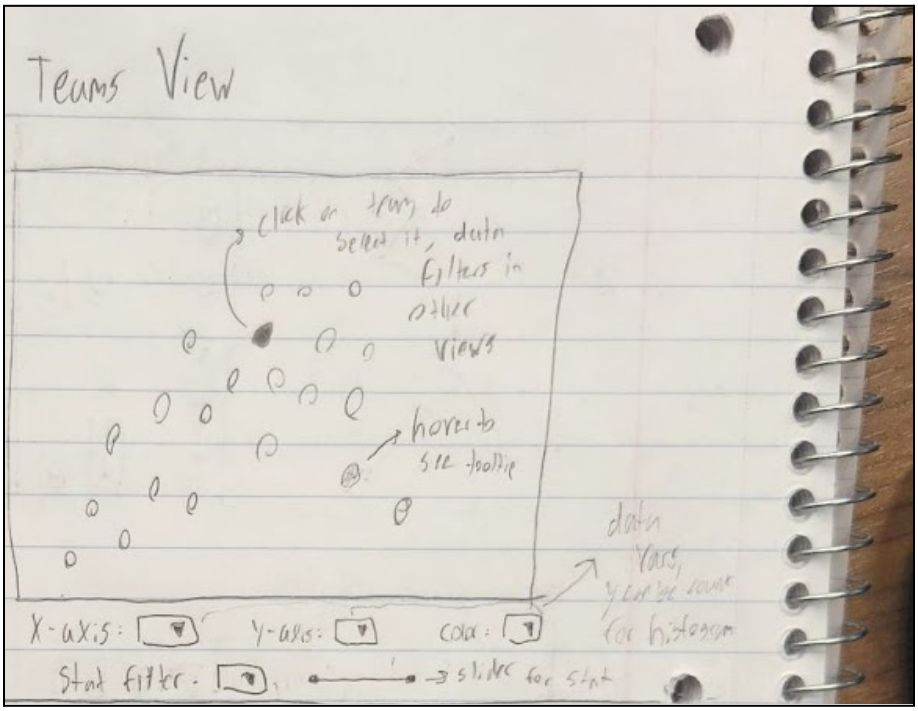
Correlation between metrics can be deduced from the scatterplots, which are intuitively located in the same row. Quantitative data variables can be easily encoded to x and y position on the scatterplot, which provides an affordance for the user in that positional data is easily and effortlessly understood by humans. Another data variable of any type would be encoded to color, allowing the user to see clusters or color trends in the plot. These scatterplots would also allow for comparison between teams, hitters, or pitchers in their performance in multiple metrics at a time.

Line charts are an excellent way of showcasing trends over time. Y position encoding can be applied to quantitative variables, an effective choice for the same reason as described above. Time on the x axis allows the user to easily see if a player or team is on an upward, downward, or steady trend.

Sustainability of performance can be portrayed by a discrete outcomes bar chart. There would be a mark on screen for the actual outcome of a player or team in some metric(s), encoded to y position. Bars for simulated outcomes of said metric(s) would be updated every few seconds. This takes advantage of the frequency approach to probabilistic outcomes - it is natural for humans to deduce how likely an event is to occur when viewing it over and over again. If the user sees the bar consistently below or above the mark which a team or player actually produced, they can deduce that the team or player is underperforming or overperforming based on underlying data. If the bar is wildly varying, the user might deduce that there is not a large enough sample size for the true data to be meaningful. The actual statistical methods for producing these simulated outcomes will be discussed in the next section.

Individual Storyboards / Interaction Description

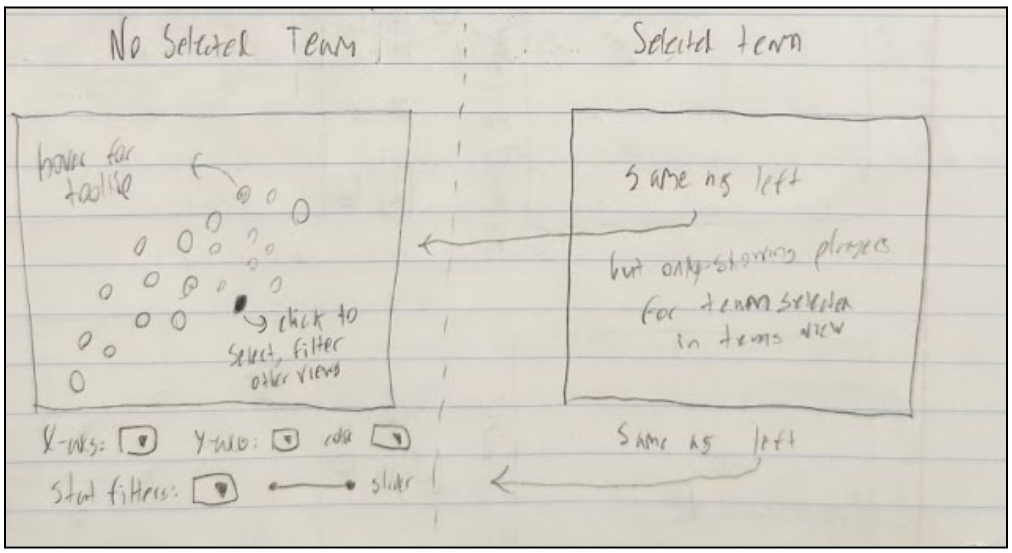
Along with the overall storyboard seen above, I created individual drawings for each view to show more detail and describe interactive behavior. To start, the team scatterplot:



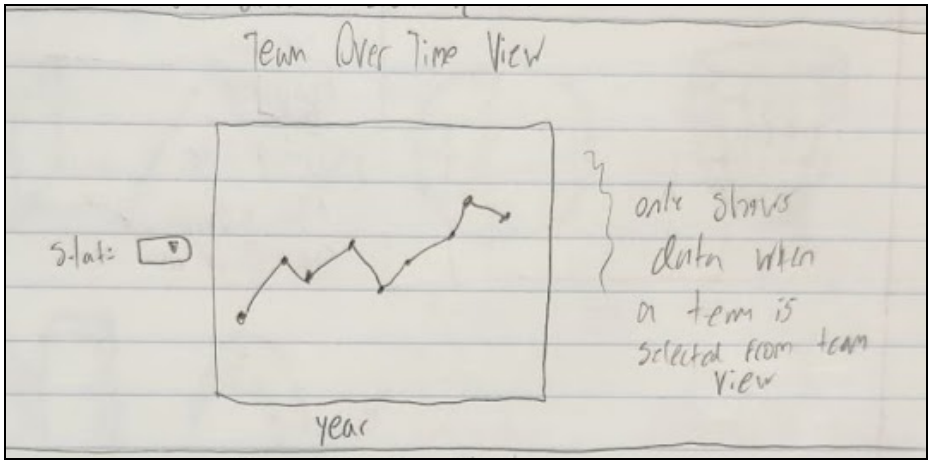
Dropdown menus are in place for x-axis, y-axis, and color, allowing the user to customize which data variables map to those encodings. This gives the user free reign over which correlations to look for and what metrics to make comparisons on. There would also be a slider where a user can filter the data being plotted based on a stat of their choosing via a dropdown menu.

Hovering the mouse over a point on the plot would show a tooltip with some basic data about the team, along with the exact values for each encoding. Clicking on a team would select it, changing its appearance to give feedback that it is indeed selected as well as filtering the hitter and pitcher views to only include players from this team. This allows the user to compare and look for trends within a team of interest.

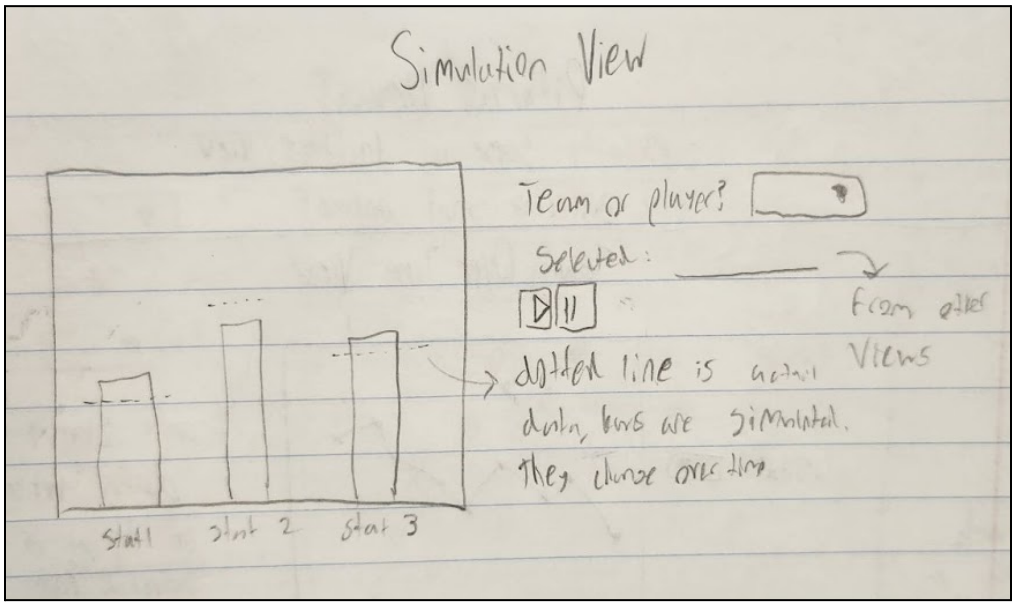
Side note: I wrote that the y-axis could be changed to "count" in this storyboard, which would change the scatterplot into a histogram, but ultimately dropped that idea upon further thought. The purpose of my tool ultimately does not include examining the underlying distribution of the data, so I thought having a way to change the plot type would add too much confusion for not a solid enough reason.



The storyboard above applies to both the hitter and pitcher scatterplots; they will have the same view and functionality with different underlying data. The x-axis, y-axis, and color dropdowns function the same as described for the team scatterplot. The slider and associated dropdown is also identical to the team view. Hovering over a point results in a tooltip and clicking a point selects it and displays it on the timeline just as described for the team plot. The only major difference is that the players shown in these views are filtered by teams selected in the team view (i.e. they must have played on the selected team).



The storyboard above applies to both the team and player line charts. A dropdown menu exists for the user to choose which metric they are viewing over time, and the lines that are plotted correspond to team selected in the team scatterplot for the team timeline, or a hitter/pitcher selected from one of the other scatterplots for the player timeline.



Lastly, the simulation view is storyboarded above. A dropdown menu would allow the user to pick whether they'd like to simulate a team or player, which would then control which stats are being displayed in the bar chart. The stats would be fixed, based on what I can simulate. The name of the selected team or player would then be displayed. A pause and play button would control whether the outcomes are being changed.

For teams, the outcome variable being simulated would be wins. A win is defined by scoring more runs than an opponent. Thus, how many runs a team scores and allows per game is a great predictor of how many wins they have (this was confirmed by my EDA for assignment 2!). I know as a baseball fan that teams sometimes win many more or less games than one would expect based on how many runs they score and allow per game throughout the season. I envision that, to simulate a team's season, I would use normal distributions centered at a team's runs scored and runs allowed per game to probabilistically determine if they win or lose a game. Doing this 162 times to represent the 162 games in an MLB season would give me a simulated win total. Viewing this simulated total next to the actual total repeatedly would give insight to if the team overperformed or underperformed based on their underlying runs scored and runs allowed.

For hitters and pitchers, the outcome variable(s) would differ and potentially be chosen by the user. A hitter's *plate appearances* represent how many times they got a chance to hit during a season. The hitter's probability of certain batting outcomes per plate appearance could be calculated based on the selected season, and simulated again for that number of plate appearances. Same goes for pitchers with *innings pitched.* As a fan, I've seen hitters and pitchers have statistics for a season that are very extreme due to a low sample size of *plate appearances* or *innings pitched*. Allowing the user to simulate a season and potentially see the wide variety of outcomes based on underlying probabilities would show how much weight they should be putting into the player's performance for that season.

Overall, I believe that the layout of the different views is easy to follow. The customizability within plots allows the user to explore what they wish, and the interactions between plots allows the user to answer more questions about team and player performance. I could envision someone being curious if a team stat correlates with winning. Then, upon seeing which teams do best in said stat, selecting a team to see which players contributed, selecting players to see how they performed over time, and simulating the selected team or player. Maybe I'm too much of a baseball analytics fan…but that sounds like fun to me!

Changes Between Storyboard and Final Product:
There were not many changes between my storyboard ideation and what I ended up creating:

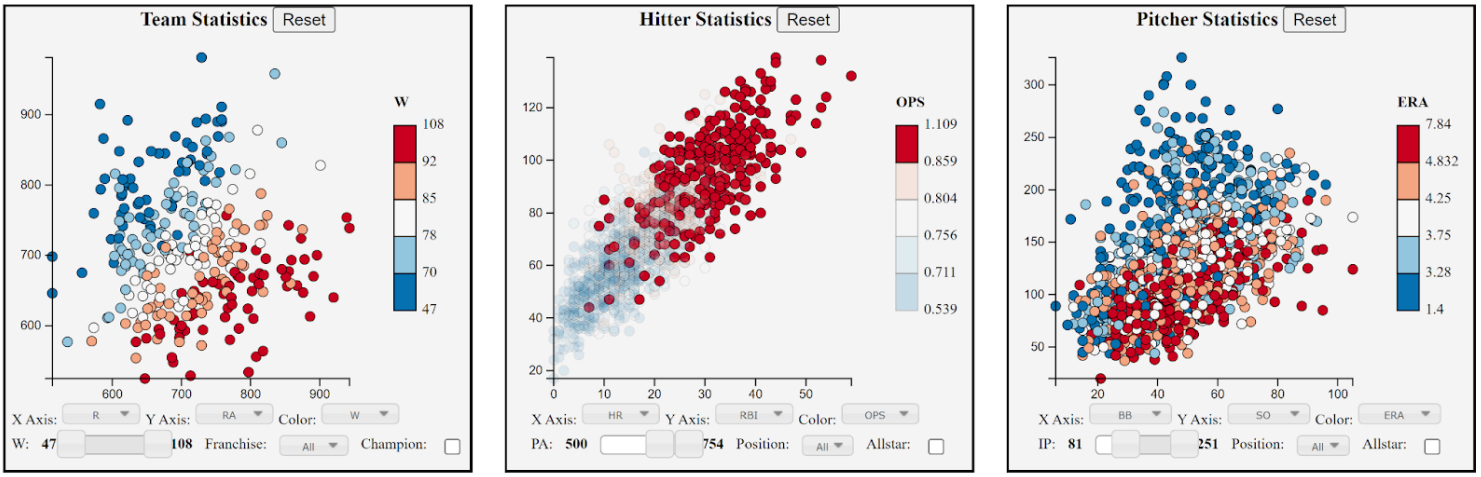
- Multi-team/player selection
- Dropdown / checkbox filters for the scatterplots
- Slider range has predetermined data variable it is filtering on
    - This is to aid in the analysis of teams and players based on what I as an avid baseball fan know is important. It will not be as overwhelming as having to choose between over 20 different variables.
- Timelines display an average per year along with selected teams or players
    - Filters for the average calculation
- Interactive legends
- Separate hitter / pitcher timelines in place of a simulation view
    - Getting rid of the simulation view was a difficult choice because I think it would be a useful tool for baseball analysts. That being said, I don't think it would be easily interpreted by any user, even someone who is a baseball fan but doesn't care for numerical analysis.
    - Uncertainty / sustainable performance can still be analyzed via the slider filters. The data variables encoded to the hitter and pitcher slider are chosen to account for sample size, and default to higher values to display only players who played a lot.

Final Application Description:
My final application consists of a title and 4 distinct rows, which I will briefly describe 1 by 1.

Top Row - Menu

This row consists only of buttons. The welcome button displays a short message introducing the application, the same message that is displayed on the user's first time loading the web page. The reset all button resets the filters and selections on all the charts. The data source button displays a message describing where the data is from. The help buttons all provide descriptions on how to use and interpret the specific type view they apply to (scatterplot, timeline, abbreviation table)

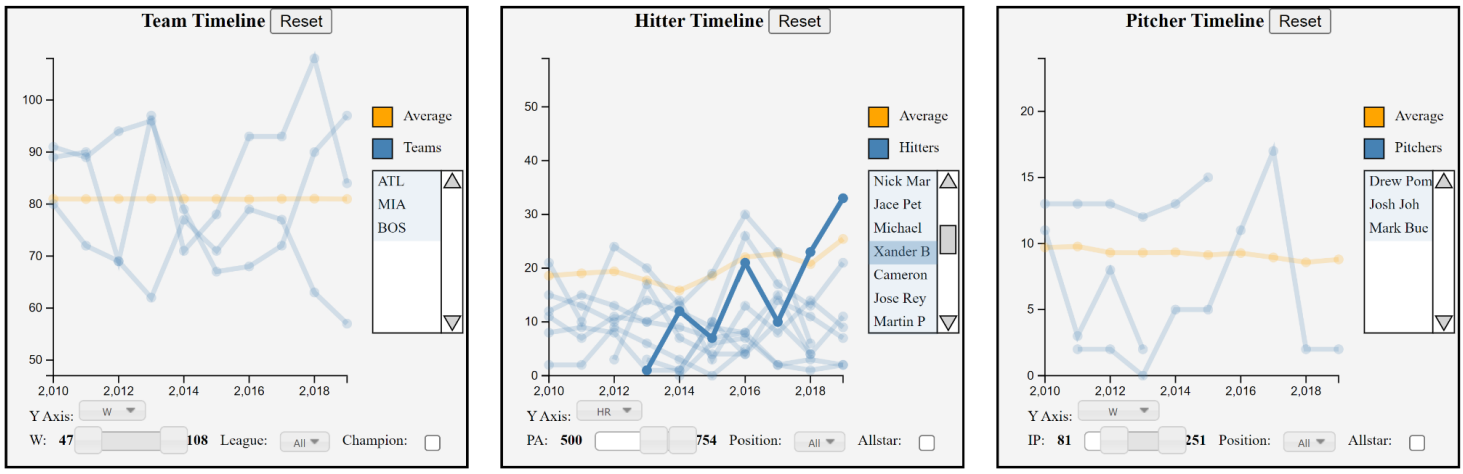
2nd Row- Scatterplots



This row displays scatterplots for team, hitter, and pitcher data. Each plot consists of controls for which data variables are encoded to position and color, a slider range filter, dropdown menu filter, and checkbox filter. The data variables that these filters control are displayed next to them (abbreviations are explained in the 4th row). Each plot also contains a reset button which puts the filters back to their defaults and deselects all points. The legend is calculated such that 20% of the data is in each color range. White is the middle 20%, darker red is higher, darker blue is lower. Dark colors for extremes makes them stand out more to the eye.

All points are highlighted when hovered over along with displaying a tooltip with basic information and exact values for each encoding. Hovering over a color on the legend also highlights points with those colors, as seen by the hitter statistics plot in the middle. Clicking on a point colors it yellow and selects it for the timelines in the 3rd row. There is no limit to how many points can be selected. Team selections also filter the hitter and pitcher plots to only show players who played on any of the selected teams.

3rd Row- Timelines



This row consists of timelines for teams, hitters, and pitchers. For each timeline, the user can choose which stat is encoded to y position. Hovering over the line itself highlights it and brings up a tooltip with the team or player name. Hovering over a point highlights the whole line and displays the name, year, and exact y-axis value.

Each view contains filters identical to the corresponding scatterplot above. These filters control how the averages are calculated, so the user can compare a hitter to the average all star hitter (i.e. good player), for example.

The legend consists of a color square for average or an individual player / team. Hovering over the color will highlight all lines of that color. The players / teams that are being shown on the timeline are displayed in the window below the color squares. If more than 7 are being shown, a scrollbar appears allowing the user to see all of the team or player names in the window. Hovering over a name highlights that line on the plot, as seen in the hitter timeline in the middle. It also displays a tooltip with the full name since it is often truncated (this disappears when I take a screenshot so it is missing from the above photo).

4th Row - Abbreviation Tables

Team Abbreivation Legend

ATL

Abbreviation	Full	Is a high or low value "good"?
ATL	Atlanta Braves	N/A

Hitter Abbreviation Legend

h

Abbreviation	Full	Is a high or low value "good"?
H	Hits	High
HBP	Hit By Pitches	High
HR	Home Runs	High

Pitcher Abbreviation Legend

Search for abbreviations...

Abbreviation	Full	Is a high or low value "good"?
BAOpp	Opposing Batting Average	Low
BB	Walks Given Up	Low
CG	Complete Games	High
ER	Earned Runs	Low
ERA	Earned Run Average	Low
G	Games Pitched In	High
GF	Games Finished	High
GS	Games Started	High
H	Hits	Low

This final row consists of searchable tables with descriptions of abbreviations for team names and statistics. It shows the abbreviation, full name, and whether it is good for a player to have a high or low value for a stat abbreviation.

Typing in the search bar limits what is shown in the table to abbreviations that match what you have typed, as shown in the team and hitter legends on the left and middle. Hovering over a row highlights it in gray, turns the text blue, and underlines it to indicate the presence of a hyperlink. Clicking on the row will open a tab to the official MLB website description of a statistic or team page.

Development Process:

The first step in developing this application was finding, cleaning, filtering, and finalizing the data. Finding and cleaning the data was easy because I used the same data for my EDA. Filtering and finalizing the data took roughly 5 hours. This seems like a lot in hindsight, but it took me a while to figure out which tables out of many I wanted to use and if I wanted to filter it. Once I figured that out, I quickly filtered it to only 2010s using Tableau Prep. Adding calculated fields to finalize the data took time as well, which I did in excel.

The next step was to storyboard. This took roughly 1 hour. Being an avid fan of baseball analytics allowed me to quickly plan what I thought would be an intuitive layout and useful interaction behavior. The drawing was most of the process (I'm a lot better at coding than I am at art…)

The final step was coding. This was the longest aspect of the development process by far, taking 40-50 hours. I know that sounds like a lot, and maybe way too much for a single assignment, but I really enjoyed working on this! I LOVE baseball data and really enjoy working with d3, so I found it really fun to develop this application. I added as many components as I could, even implementing my own scrollbar using an svg rect and d3 event handlers for the timeline legends. I'd say 10% of the coding was spent on html, 5% css, 85% javascript. A few hours went into setting up the layout in html and css, and then it was almost all javascript from there. I hope you enjoy!