Major League Baseball Statistical Management Dashboard

Alex, Conroy (08324841)
IFN703/4 Assessment 3

Executive Summary

Sabermetrics was first introduced in Major League Baseball in the late by Bill the *Baseball Abstract* [1]. Since then it has become intrinsically linked to both strategy development and management of all teams within both Major League Baseball (MLB).

The project aims to provide a documented example of a visually excellent dashboard, as described by Edward Tufte and Stephen Few. The project significance is derived from the documentation of the visualisation and dashboarding process. While many dashboards and visualisations have been created in this area. There appears to be little to no publicly available decision frameworks for them. This is likely due to the competitive nature of the industry. The dashboard and visualisations have been built with the Team Manager of any MLB team in mind as the end user.

The data used for this project was collected from the Retrosheets website and consists of regular season data from 2000 – 2019. The visualisations presented in the dashboard were done with the purpose of presenting a fully customisable insight profile at the player and roster levels for all current players careers in the major leagues.

The software selected to present these visuals was the Plotly Dash libraries. These were chosen for their native interactions and depth of documentation. The end dashboard could display all data originally set out for the project. However, there are several areas of improvements that could occur in future works.

Introduction

Sabermetrics generally refers to the statistical measurement and analysis of in-game activities, that for part of the broader empirical analysis of Baseball [1]. SABRmetrics (Sabermetrics) is a term originally developed by Bill James and is a derivative of the acronym for the Society of American Baseball Research, which was established in 1971 [3].

Baseball can be defined as a bat-and-ball based game, which through a variety of activities aims to complete the objective of scoring runs. The team with the greatest number of runs when the match is completed is declared the winner. This project will assume as a baseline that the reader has a general knowledge on the mechanics of the game of Baseball.

Sabermetrics was first introduced in Major League Baseball in the late 1980s when the then Manager of the Oakland Athletics first employed a derivative of Bill James' strategies outlined in the *Baseball Abstract* [1]. Since then it has become intrinsically linked to both strategy development and management of all teams within both Major League Baseball (MLB), and the associated minor leagues [1]. While the strategies and metrics are widely used, due to the nature of the sporting industry, there is little to no academically robust evidence as to the best way to represent and communicate this data

Due to the importance of the data to the modern game and the depth of data that such early adoption has provided. The development of a formalise framework for player centric visualisation, that are grounded in academically sound data visualisation practices should be created to contribute to the base understanding of Sabermetricians.

Research Purpose

The purpose of the project is to create a design for a dashboard that can facilitate player and roster level decision making by the Team Manager of an MLB team or fantasy MLB team. The dashboard will attempt to present the Team Manager with an at-a-glance view of customisable visualisations and statistics at both a player and team roster level. With player level insights designed to provide the ability to profile a player to assist in general managerial tasks such as: player evaluation, trade evaluation, team comparisons, roster selection.

Research Problem & Significance

American baseball and in particular MLB are one of the most famous and long running users of mathematical modelling in sport. To the extent that both Hollywood movies and pop-culture references are routinely made on the subject [1]. The project will aim to provide a documented example of a visually excellent dashboard, as described by Edward Tufte and Stephen Few in their various works and outlined in the Dashboard design and Visualisations sections of the Approach. The project significance is derived from the documentation of the visualisation and dashboarding process. While many dashboards and visualisations have been created in this area. There appears to be little to no publicly available decision frameworks for them. This is likely due to the competitive nature of the industry.

Research Objective

The objectives for this project can largely be split into three groups. The first objective is the identification of a general set of Sabermetric statistics that would be reasonably used by a Team Manager, and derived the necessary visualisations needed to extract insight from them. The second objective is to build the necessary data pipelines to transform the Retrosheets dataset into the necessary shape and size to drive the given visualisations. The last objective is to build a dashboard that is capable of correctly displaying and controlling the visualisations, with a potential ability of user customisation to allow for personal preference in displayed metrics to be provided.

Scope

The identified scope of this project is all Retrosheets regular season data from 2000 – 2019. This was done to capture the entire major league careers of all active players. With the longest running active player Albert Pujols debuting in the 2001 season. The Dashboard will be focused on batting and pitching statistics only. However, will display general fielding positions to provide rounding on a player's profile.

Literature Review

This section aims to provide insight into the current literature that surrounds this project. The literature explored has been broken down into three general areas that reflect the different parts of the project: Sabermetrics, Sports data and visualisation, Dashboard Design.

Sabermetrics

Sabermetrics has become intrinsically linked to baseball management [1] through the works of Earnshaw Cook and Bill James in the 60s and 70s in their works Percentage Baseball [4] and the now infamous Baseball Abstracts [5]. This has led baseball to be known as one of the most data rich sports played.

As the focus of this project is the understanding and visualisation of Sabermetric metrics used in Baseball. Understanding the underlying principles of the founding methods as important first step before being able to move on to the modern iterations. This can only then be combined with the software necessary to manage and compliment the data and raw statistical analysis.

The origins of the underlying studies that led to the statistical basis of Sabermetrics can be found in several texts designed around tertiary class materials for both the Seton Hall University and West Point US Army Academy. Understanding Sabermetrics: An Introduction to the Science of Baseball Statistics by Costa, Saccoman and Huber [6] was considered a radical change from the baseball statistical analysis of the time. With the authors being early adopters of high-level mathematical modelling at the core of the statistics. This book along with Practicing Sabermetrics: Putting the Science of Baseball Statistics to Work [7], Curve Ball [8], Baseball Between the Numbers: Why Everything You Know About the Game Is Wrong [9], and The Book: Playing the Percentages in Baseball [10], provided the basis for modern Sabermetrics. The topics in the books gave a range of measurements that could be used to define a player's ability and value. These books give insight into the fundamental definitions for the types of data that the metrics produce.

A more modern investigation from the Mathematical Association of America titled Mathematics in Sport [11] specifically targets a comparison of now traditional sabermetric methods and their more contemporary counterparts. The paper discusses the changes to measurements within the sport due to advances in data capturing technology. This paper serves the purpose of determining if modern practices have in anyway changes the underlying data's definitions. Which in practice could affect any visualisations.

Concluding this section the paper by Middleton *et al* [12] conducted a profiling analysis on the Sabermetric community to gain context of how these metrics are being used in practice. The paper provides insights into how the baseball industry balances the needs for rigorous scientific analysis of its practices, with the need for constant innovations to maintain an 'edge' over competing teams. This paper complements the one from the Mathematical Association of America in providing the insights on whether the metrics to be visualised are being used in conjunction to their actual mathematical definitions.

Sports data and visualisation

Visualisations of data is viewed as a directly linked to the underlying data type when performed using data science principles. This is still the case when looking at baseball data.

There are many papers that consider the principles around generating visualisation from sports data. The paper State of the Art of Sports Data Visualization [13] predominantly explores meta-data and time-series data as it relates to baseball, among other sports. The paper provides the outline to a framework for visualisations of these types. This has provided great insight into the current project. The book Sports Analytics: Analysis, Visualisation and Decision Making in Sports Performance [14] compliments this paper with a more in depth look at the theoretical approach to the whole data lifecycle from collection through to visualisation and insights. But as stated it does not provide practical examples.

The paper Baseball4D: A Tool for Baseball Game Reconstruction & Visualization [15] Is an example of the papers found that takes a more micro-view approach to visualisations. With the focus being deep dives into sub-specialty metrics within baseball. The paper explores three-dimensional tools to reconstruct and replicate player and ball movements in a baseball match. This coupled with the Visualizations for Exploration of American Football Season and Play Data [16] paper detailing a stacked abstraction visualisation of gameplay within American Football, give examples on how to visualise game summaries. These papers begin to frame the need for quick insights; however, their methods are too niche for direct application in the project.

Dashboard Design

Dashboarding has become increasingly popular with the overall rise of the fourth wave of technological development. But as with most things on the cutting edge. There is often little oversight into a best-practices approach on how to achieve precise and suable outcomes. Stephen Few is recognised as a dashboarding subject matter expert. Having written multiple books, papers, and keeps a up to date white paper publication service.

Stephen's book Information Dashboard Design: The Effective Visual Communication of Data discussed the powerful yet fickle nature of dashboard design [17]. With a set of guidelines for mistakes to avoid enabling quick and precise viewing of information. Stephen has also written industry papers on the

concepts of taking raw information through to visualisation insights including industry examples of poor and good quality designs, with grounding in human information delivery sciences [18][19].

Approach

As discussed in the purpose the end user for this dashboard and visualisations is the Team Manager of any MLB team. However, by its nature this can also be extended to include any fantasy team league as well.

This section will give discuss which metrics are believed to display an appropriate player profile, but the overall design of the dashboard will attempt to allow for incorporation of as many other general metrics as possible. These will be outlined in the dashboard design section.

Data Transformation

The data used for this project was collected from the Retrosheets website. The data was originally collated as a zip file for each decade. With the data being represented for each year as multiple text files. Table 1 outlines the general file types and the data captured.

File	Example	Data Captured
Yearly Team Event File	2000CLE.EVA	Event file for play-by-play data of home games for the indicated team of the indicated year. File also includes general game info, such as time, date, weather. The example is the 2000 season for the Cleveland Indians, with EVA indicating they are in the American League.
Yearly Roster	CLE2000.ROS	A roster for the given team of a given year. This file contains player ID, name, team, and position information. Example is for the 2000 Cleveland Indians.
Yearly Teams	TEAM2000	A short list of all the teams that played in the given year. Data captured includes: Team ID, Team Full Name, League Status. Example is for the 2000 season.

Table 1 File Description of Raw Data collected from Retrosheets.org

After several attempts to manipulate this data into the desired game-by-game player focused format. It was decided to employ one of the Retrosheets executable file manipulators, however it is recommended that this be replaced with a completely state-of-the-art pipeline in future works. As this decision led to a restriction in the number of metrics the dashboard could display in the build phase of the project.

```
Game of 3/28/2019 -- New York at Washington (D)

  New York           AB  R  H RBI    Washington         AB  R  H RBI
Nimmo B, lf           3  0  0  0   Eaton A, rf          3  0  1  0
Alonso P, 1b          4  0  1  0   Turner T, ss         4  0  2  0
Broxton K, pr-rf      0  0  0  0   Rendon A, 3b         4  0  0  0
Cano R, 2b            4  1  2  2   Soto J, lf           3  0  0  0
Conforto M, rf        4  0  0  0   Zimmerman R, 1b      4  0  0  0
Familia J, p          0  0  0  0   Gomes Y, c           3  0  1  0
Diaz E, p             0  0  0  0   Adams M, ph          1  0  0  0
Ramos W, c            4  0  1  0   Dozier B, 2b         3  0  0  0
McNeil J, 3b          3  0  0  0   Scherzer M, p        3  0  0  0
Rosario A, ss         4  0  1  0   Miller J, p          0  0  0  0
Lagares J, cf         3  0  0  0   Grace M, p           0  0  0  0
deGrom J, p           2  0  0  0   Barraclough K, p     0  0  0  0
Lugo S, p             0  0  0  0   Robles V, cf         3  0  1  0
Smith D, ph-1b        0  1  0  0
                     -- -- -- --                      -- -- -- --
                     31  2  5  2                      31  0  5  0

New York         100 000 010 --  2
Washington       000 000 000 --  0

  New York            IP  H  R ER BB SO
deGrom J (W)         6.0  5  0  0  1 10
Lugo S               1.0  0  0  0  0  3
Familia J            1.0  0  0  0  0  0
Diaz E (S)           1.0  0  0  0  0  1

  Washington          IP  H  R ER BB SO
Scherzer M (L)       7.2  2  2  2  3 12
Miller J*            0.0  1  0  0  0  0
Grace M              0.1  1  0  0  0  0
Barraclough K        1.0  1  0  0  0  0
  * Pitched to 1 batter in 8th

DP -- New York 1
LOB -- New York 5, Washington 6
2B -- Robles V
HR -- Cano R
SB -- Turner T 3
CS -- Rosario A
HBP -- by Familia J (Eaton A)
T -- 2:44
A -- 42263
```

Figure 1 Example output of box score for Game 1 of 2019 for the Washington Nationals

The data manipulation tool chosen was the BOX.EXE [20]. This command line executable program in took the event and teams raw data files for all teams of a given year and processed them into a game-by-game box score-based space delimited file. During this process, all team name, city, and league changes were handles. Notable inclusions are the Washington Nationals being previously known as the Montreal Expos, The Huston Astros changing from the American League to the National League, and the Florida Marlins becoming the Miami Marlins. This gave the dataset a recognisable structure of player centric data that could be further processed with a state-of-the-art script. An example of the output can be seen in figure 1.

Player Name	Position	Game Date	AB	R	H	RBI
Knoblauch C	2b	4/03/2000	4	0	1	0

Table 2 Post-processed Batting Data

From this point the files were combined into one dataset spanning the desired 20-year timespan and processed into a two csv files. One for batting that consisted of the game date, player name, position, at-bat, runs, hits, runs batted in. The second for pitching that consisted of game date, player name, innings pitched, hits, runs, earned runs, walks, strikeouts. Table 2 and 3 provides an example of the dataset rows. These datasets where then mated to the 2019 teams' rosters to add another column that captured which teams all current players played for in the 2019 season.

Player Name	Game Date	IP	H	R	ER	BB	SO
Nelson J	4/03/2000	1	0	0	0	1	1

Table 3 Post-processed Pitching Data

Originally this was attempted using the python language like the rest of the analysis and dashboard creation. However, the language seemed to struggle with the space delimited nature of the data. Failing to execute the extraction correctly. For this reason, the pipeline was completed using MATLAB, see appendix 3. The finalised files consisted of ~1.2 million lines of batting and ~500,000 lines of pitching data.

Visualisations

The main focus of the visualisations for the dashboard can be represented in three different categories that combined with the metrics of choice, provide the player or team profile that is the main goal of the insights for the project. The first is the timeseries aspects, which aim to provide career spanning insights. The second is the radar plot, which aims to give insight into the overall ability of the play/team. With the last insight being player specific, that aims to show positional data, which is more of a complimentary visual to the other two graphs.

The visualisations were all built using the Plotly libraries, this was done to allow for native integration into the Dash dashboard software. This allowed for greater control and functionality within the dashboard ecosystem but did result in some complications that will be discussed within the individual visualisation subsections below.

An alternative investigation was conducted into using the ipywidgets and predominately matplotlib library to achieve the visualisations and interactions. But was ultimately changed to the singular Dash/Plotly environment, the reasons for which will be expanded upon in the dashboard section.

Visualisation examples will be conducted on Max Scherzer, who began his MLB career in 2008 with the Arizona Diamondbacks, then the Detroit Tigers in 2009, and finally the Washington Nationals in 2015. Max has received several awards including 3 CY Young awards, 4 Wins leader, 3 NL strikeout leaders, 2 immaculate innings and was a 2019 World Series Champion. Max also holds 2 tied MLB records. This makes Max an interesting example player to display the visualisations.

Metrics

In general baseball metrics are represented as either a whole number of a decimal, usually to three decimal places. The more advanced statistics are usually these decimal type metrics as they are derived from the more basic statistics that themselves are taken directly from in-game recordings.

From a visualisation point of view, all the statistics that the dashboard will be looking to incorporate will be considered discrete time-series data types. Furthermore, there will only be two things about the various metrics that will be considered from a visualisation standpoint. That is the differing scales and interpretations of what is considered a good or bad outcome. For example, the Hit metric measures a batter strike that lands in fair territory and advances the batsman to a base without this

occurring due to a fielding error. For this metric, the higher the score associated with a batter the better, while the opposite is true of a pitcher.

The scale considerations will be incorporated by pinning the axis of a visualisation to the data, rather than having a uniform axis. While the axis direction will be reversed on those metrics and visualisations that it makes sense to do so. These will be highlighted in the subsections below.

The dashboard and visualisations will support metrics from almost all metrics as defined by MLB.com's glossary [21].

Timeseries Data

The main goals of this visual is to show trend and time base insights. While the radar plot discussed below can give better visualisations on the numbers of the chosen metrics, these visual aims to provide insight at the career level. Users should check the data for gaps in play, that might indicate injury or rest periods, particularly in older player. Users will also be able to tell if the mean number that derives the radar plot the result of consistent effort over the chosen timeframe, or the rest of a skewing towards more recent games or older ones.

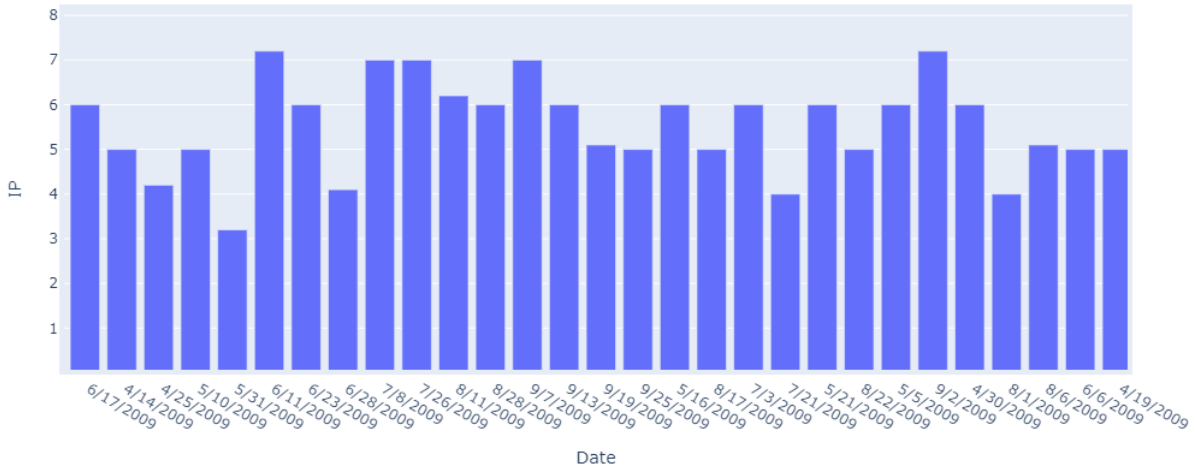


Figure 2 Innings Pitched by Max Scherzer for the Arizona Diamondbacks 2009 season

An example of the visualisation is shown above for the innings pitched for the 2009 season by Max Scherzer in his first season as a starting pitcher for the Diamondbacks. It shows a consistent effort for most of the season, with some smaller efforts towards the beginning. This appears to show that Max quickly adapted to his new position as a starting pitcher and matches the trajectory of Max's career at this stage.

The visualisation looks identical when used on a team level chart, with just the y-axis changing depending as the plot will show the accumulative efforts of all the entire roster.

Both line and dot/scatter graphs were considered as substitutes for this visual. The line graph was dismissed due to the discrete nature of the data, while the scatter graph was dismissed as the almost levels nature of the data gave this visual a confusing pseudo-horizontal bar graph look to it.

Radar Plot

The main goal of the radar plot is to visualise the player or team profile for the preselected metrics. It is also used as a direct comparison at the player to player level or team to player level. Users can directly see how a player or team is performing in each metrics. Users will then be able to see if the player fits the profile that is desired from them, either in their current role or in a future role they are being considered for.

In a similar fashion, the radar plot can be used to compare if a player is a positive contributor to a team average or not, by comparing the overall size of the player and team plots.

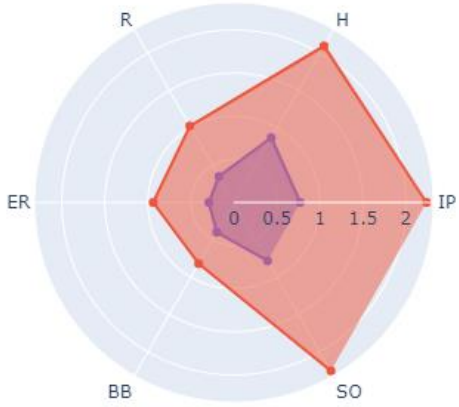


Figure 3 Dan Jennings Pitching Comparison to Washington Nationals for 2019 Season

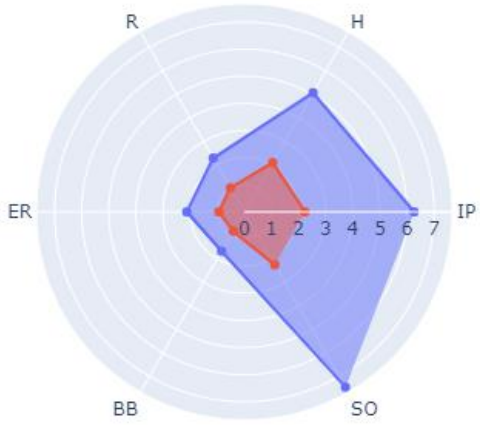


Figure 4 Max Scherzer Pitching Comparison to Washington Nationals for 2019 Season

This is shown again shown in the above examples with Max for the 2019 season, when compared to the average of the Washington Nationals pitchers. This is highlighted by the comparative plot of teammate Dan Jennings who it can be seen is one of the pitchers that performed under the average for the Nationals in 2019.

Position Data

The final visualisations aim is to provide even more context to the player profiling. The pie chart is used to give at-a-glance referencing to which positions a player has most of their career experience in. This has been designed in aiding with roster and trade analysis, to further aid in determining if the player will fit within the vision the Manager has for their team.

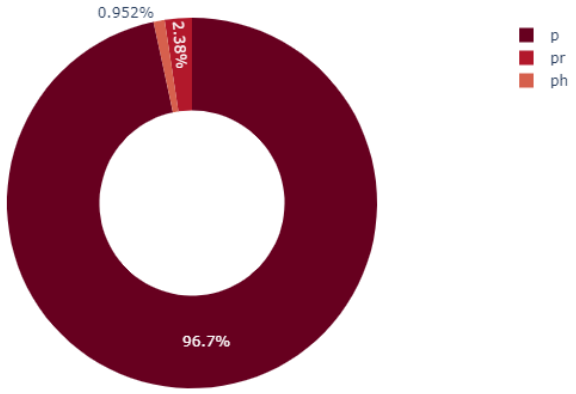


Figure 5 Max Scherzer positions played in career

The example is again looking at Max, who has spent much of his career as a pitcher, but has nominal experience as a pinch hitter, and pinch runner.

Dashboard

The dashboard application will focus on three main areas. Those being the software packages used, the design, and the functionality.

Software

As previously mentioned, the project underwent a change in the software selected for both the visualisation and dashboarding process. This was done as the new Plotly based libraries offered superior native interactivity and dynamic viewing over the visualisations, along with a deeper control over the aesthetics of the finished Dashboard.

Dashboard Functionality

With the data pipelines placing emphasis on both the length of time that the dataset consisted of as well as the capture of both teams and players. This became the basis of the controls given on the dashboard. A secondary focus was to the eventual comparative use cases. Essentially meaning that inclusive of the world series and other specialty playoffs, any team in the MLB could play one another, and any player could play for any team. This meant that the dashboard needed to support the full list of current players, and teams.

Figure 6 Blank dashboard controls including snipped on the range slider

As shown in the above figure, of the actual controls from the built dashboard. The player and team selection controllers are handled by dual dropdown boxes/ search boxes. While changes to metrics shown are handled by radio switches. This was to keep in fitting with the design parameters discussed in the next subsection. With the metrics being shown are handled by the range slider, this means that full control of the seasons shown is given to the user, as opposed to a standard slider that would just give accumulative controls to the user.

As mentioned previously, this was only built for the 2019 rosters, however, the collected data can support prior years rosters. It was just considered out-of-scope to include this functionality for this project. The only consideration needed when viewing the dashboard is that prior years of a team's performance are showing the 2019 rostered players historic metrics rather than accounting for roster changes.

Figure 7 Plotly Multi-select Dropdown Box [22]

Currently the control system does not allow for the selection of a priorly supported list of metrics as originally intended. This will be discussed further in the findings section. But it would have been displayed as a multi-select dropdown box [22], an example of which is shown above.

Dashboard Design

The dashboard design was a simple four quadrant design with a horizontal control box shown in the figure below. See appendix 1 for a full display of current dashboard.

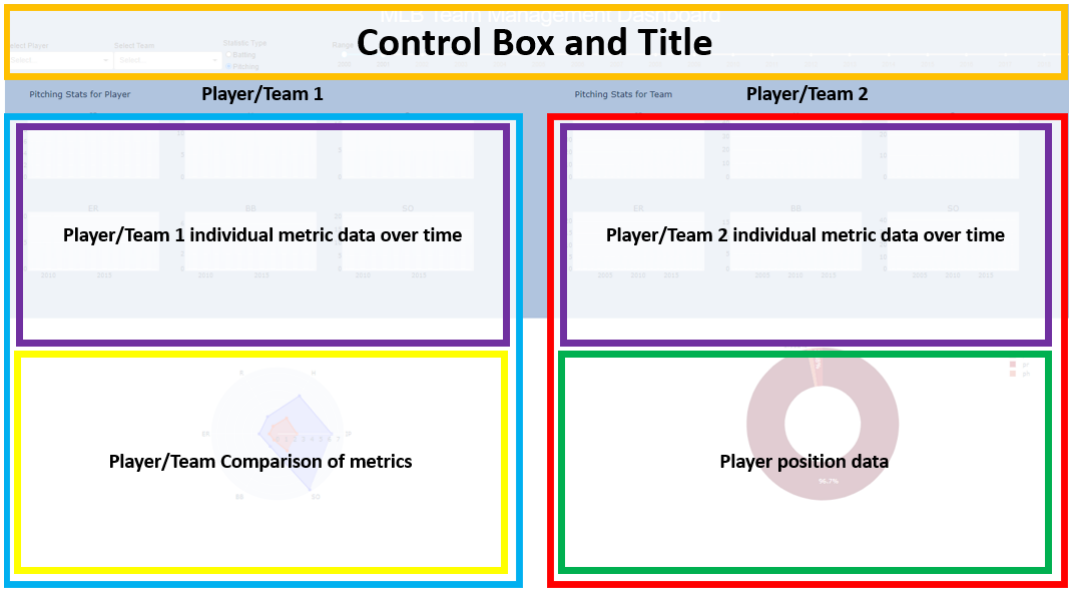


Figure 8 Dashboard with layout highlight

The four quadrants design was chosen as a results of the common pitfalls dashboard methodology outlined by Stephen Few [17] and shown in the table below. Specifically, pitfall 1. Similarly, this includes the general minimalist designs around the background colouring and aesthetics, pitfalls 11,12 and, 13. The decision to subplot all the chosen metrics into a 3x2 grid was in an attempt to adhere to avoid pitfalls 4, 2 and 9.

Pitfall	Description
1	Exceeding the boundaries of a single screen
2	Supplying Inadequate Context for the Data
3	Displaying Excessive Detail or Precision
4	Expressing Measures Indirectly
5	Choosing Inappropriate Media of Display
6	Introducing Meaningless Variety
7	Using Poorly Designed Display Media
8	Encoding Quantitative Data Inaccurately
9	Arranging the Data Poorly
10	Ineffectively Highlighting What's Important
11	Cluttering the Screen with Useless Decoration
12	Misusing or Overusing Colour
13	Designing an Unappealing Visual Display

Table 4 13 Pitfalls of Dashboard design as outlined by Stephen Few [17]

Findings

This section will discuss the end results of the approach, focusing on whether objectives of the different subsections have been met, and discussing how to rectify any failures or oversights.

Data Transformation

As already mentioned in the approach section, the way the data pipelines were built created a limitation on the metrics achievable for visualisation. While it was identified that this was not crucial for the development of the dashboard. Should the dashboard and associated pipeline enter any for m of true usability this will need to be rectified. As one of the biggest drawbacks on the current pipeline is the inability to associate walks (BB) with batters, only being able to identify the pitcher. This suggests that there is a need to scrap the use of the BOX.exe initial pipe and create a fully customised extraction of the play by play data.

Given that this was the original plan outlaid at the beginning of the project. That was only abandoned due to the failure to create a viable pipeline in the timeframe given. It makes sense for this to be a necessary adjustment should the dashboard be fully functional for the intended audience.

Another necessary change needed to the pipelines/backend of the project is the implementation of basic database engineering/management techniques. As the large nature of the dataset has created a small lag in the calculations and visualisations being performed particularly when a new user input

is selected when the dashboard is operating. This is currently considered out-of-scope but warrants a mention.

Visualisations

The visualisation has achieved what was set out to be accomplished at the beginning of the project. However, there are improvements that can be made to the readability and additional insights than could be captured from the data.

Timeseries

This graph was the visualisation most needing of changes. The readability of the graph was less than ideal when looking at longer periods of time and other ways that will capture the desired outcomes better. One improvement would be to substitute the diagram for that of a histogram and logarithmic scatter plot [23]. Below is an example of this new visual using the Max Scherzer's career innings pitched.

Figure 9 Alternative Histogram and Logarithmic visuals for Max Scherzer at career level

These two visuals vasty improve upon the readability at the larger scales and can provide almost the same insights that were sought from the original, with the added benefit of skew from the histogram. With the histogram showing the most likely outcomes for the period of interest. With Max it shows the innings pitched will likely be between 6 and 8 with both coming in at over 100 counts each through out his career. The logarithmic insight appears to show a steep curve in the number of innings pitched early in Max's career with a general plateau afterwards. However, there are a couple of outliers to this pattern that might warrant a further investigation into smaller timeframes. For the last insight into games missed, this can be better represented with a simple number with a descriptive caption.

Lasty, the addition of small inline boxplots and min/max numbers for each visualisation would be a good way to round out the timeseries updates.

Radar Plot

Overall, the findings from the radar plot are the most positive of all the visualisations. With the graph presenting the insight in the manner with which it was intended. The only problem with the graph is in the treatment of the axis on certain metrics. This issue was originally highlighted in the metrics subsection of the previous section in that what is considered good or bad for a given metric might be a higher or low score depending on which metric it is or which position is being analysed.

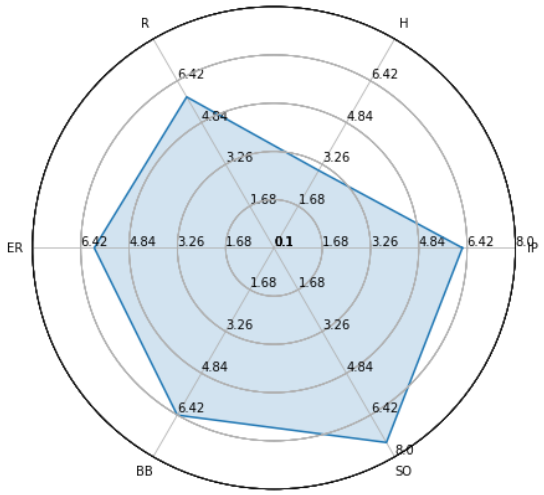


Figure 10 Radar plot for Max Scherzer's career with independent axis

This can simply be solved with an independent axis direction for each metric on the radar plot. Without this the plot itself is still relevant, it would just require a baseball subject matter expert. While the updated graph simple hold to the rule or the bigger the better. A representation of this can be seen above for the example player Max Scherzer.

Position Data

The position visualisation as it currently looks is a fine representation. However, it lacks a level of contextualisation that can and should be applied to the visualisation. This would be achieved by a simple baseball field map with labels for positions and time spent in each position given by a percentage under the labels. Below is a generic example of what this would look like.

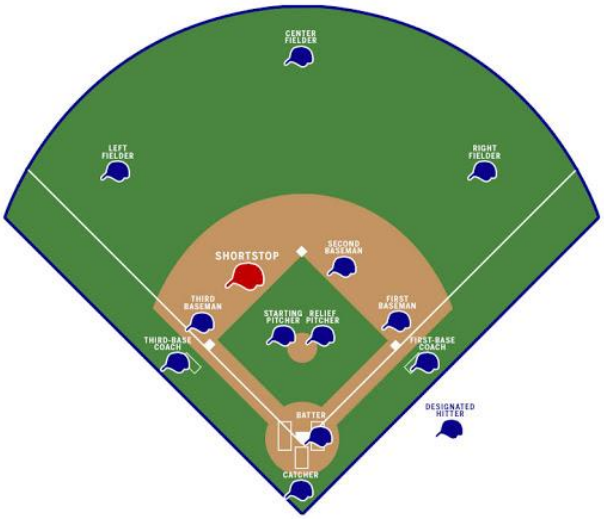


Figure 11 Defensive Positional Map [24]

There was also an initial look into the update being a sunburst plot, to enable capturing of secondary positions that a player has been asked to perform at the same time as their 'main' position. This idea was superseded by the above suggestion.

Dashboard

The dashboard has displayed the visualisations in an uncluttered and decently controllable manner. With the Plotly/Dash libraries offering long term solutions should the dashboard need to enter production.

The dashboard could benefit from a general tweak, and optimisation for an individual screen type. As it is currently setup to be dynamic. Meaning for an ultra-wide monitor there is far too much space between the visualisation, but for a smaller square monitor there is not enough room to properly display everything.

The dashboard will need the addition of tabs shown below, to properly maintain desired visualisation properties for the player v player, team v team and team v player used cases. Unless full dynamicity is possible with the Dash library.

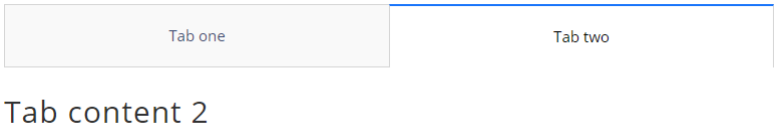


Figure 12 Plotly Dash tab structures [25]

To accomplish the changes discussed in this section it is suggested to alter the sub plotted individual metrics into one larger graph that can turn on and off the appearance of the multiple user selected metrics. This will allow for timeseries metric comparisons without making either the dashboard or the individual visualisations too busy. This is something that is supported natively by Dash [26].

Limitations

Taking into consideration all the additional insights derived from the finding section there will still be several limitations to the dashboard.

The dashboard will ideally need to be interpreted by both a baseball and statistics subject matter expert, or a minimum someone with working knowledge of both.

As the data source currently sits, the dashboard will be perpetually behind the most current information as Retrosheets only provides information for a season after it is complete. This would require the finding or creating of a new data source to overcome.

Reflection

The project was impacted in several ways that were both captured and missed by the RAID log shown in the proposal. The pandemic and related remote learning aspects captured in the log both generally slowed the process of the project. This was felt mostly in areas of life beyond the project that themselves then drove impacts on the project.

Other impacts on the overall outcome of the project included the significantly longer than expected time to achieve functional data pipelines. That impacted the time that could be devoted to the true end goals of the project, the dashboard.

The decision to begin to build the visualisations in ipywidget and matplotlib, before changing to and rapidly learning the Plotly Dash libraries was also another factor that created an overall time crunch at the end of the project. That further negatively affected the overall end results.

While these changes were undertaken in a less than ideal timeframe. They did ultimately create the basis for a much more functional tool that can be expanded upon in future works

Future Works

The future works that should be undertaken for this project would initially be to implement and investigate the additions described in the findings section of the report. With particular interest shown to the pipeline and data engineering aspects.

References

[1] 'The Truth about Sabermetrics', Samford University. https://www.samford.edu/sports-analytics/fans/2018/The-Truth-about-Sabermetrics (accessed Aug. 16, 2020).

[2] E. R. Tufte, The Visual Display of Quantitative Information, 2nd edition. Cheshire, Conn: Graphics Press, 2001.

[3] 'A Guide to Sabermetric Research – Society for American Baseball Research'. https://sabr.org/sabermetrics (accessed Aug. 16, 2020).

[4] T. M. Press, 'Percentage Baseball | The MIT Press'. https://mitpress.mit.edu/books/percentage-baseball (accessed Aug. 16, 2020).

[5] B. James, The New Bill James Historical Baseball Abstract, 1 edition. FREE PRESS, 2007.

[6] G. B. Costa, M. R. Huber, and J. T. Saccoman, Understanding Sabermetrics: An Introduction to the Science of Baseball Statistics, 2d ed. McFarland, 2019.

[7] G. B. Costa, M. R. Huber, and J. T. Saccoman, Practicing Sabermetrics: Putting the Science of Baseball Statistics to Work. McFarland, 2009.

[8] J. Albert and J. Bennett, Curve Ball: Baseball, Statistics, and the Role of Chance in the Game. Copernicus, 2001.

[9] B. Prospectus and J. Keri, Baseball Between the Numbers: Why Everything You Know About the Game Is Wrong, 1 edition. Basic Books, 2007

[10] T. Tango, M. Lichtman, and A. Dolphin, The Book: Playing The Percentages In Baseball. CreateSpace Independent Publishing Platform, 2014.

[11] J. A. Gallian, Mathematics and Sports. MAA, 2010.

[12] J. Middleton, E. Murphy-Hill, and K. T. Stolee, 'Data Analysts and Their Software Practices: A Profile of the Sabermetrics Community and Beyond', Proc. ACM Hum.-Comput. Interact., vol. 4, no. CSCW1, p. 052:1–052:27, May 2020, doi: 10.1145/3392859.

[13] C. Perin, R. Vuillemot, C. D. Stolper, J. T. Stasko, J. Wood, and S. Carpendale, 'State of the Art of Sports Data Visualization', Comput. Graph. Forum, vol. 37, no. 3, pp. 663–686, 2018, doi: 10.1111/cgf.13447.

[14] A. Jayal, Sports analytics: analysis, visualisation and decision making in sports performance. Abingdon, Oxon ; Routledge, 2018.

[15] C. Dietrich, D. Koop, H. T. Vo, and C. T. Silva, 'Baseball4D: A tool for baseball game reconstruction visualization', in 2014 IEEE Conference on Visual Analytics Science and Technology (VAST), Oct. 2014, pp. 23–32, doi: 10.1109/VAST.2014.7042478.

[16] S. G. Owens and T. J. Jankun-Kelly, 'Visualizations for Exploration of American Football Season and Play Data', p. 5.

[17] S. Few, 'Dashboard Confusion Revisited', p. 6, 2007.

[18] S. Few, 'Intelligent Dashboard Design', p. 4.[19] S. Few, 'Pervasive Hurdles to Effective Dashboard Design', p. 7.

[19] S. Few, 'Pervasive Hurdles to Effective Dashboard Design', p. 7.

[20] 'Retrosheet Tools'. https://www.retrosheet.org/tools.htm (accessed Oct. 29, 2020).

[21] 'Glossary', *Major League Baseball*. http://m.mlb.com/glossary (accessed Oct. 29, 2020).

[22] 'dcc.Dropdown | Dash for Python Documentation | Plotly'. https://dash.plotly.com/dash-core-components/dropdown (accessed Oct. 29, 2020).

[23] M. Stephen, C. Gu, and H. Yang, 'Visibility Graph Based Time Series Analysis', *PLOS ONE*, vol. 10, no. 11, p. e0143015, Nov. 2015, doi: 10.1371/journal.pone.0143015.

[24] 'What is a Shortstop? | Glossary', *Major League Baseball*. http://m.mlb.com/glossary/positions/shortstop (accessed Oct. 29, 2020).

[25] 'dcc.Tabs | Dash for Python Documentation | Plotly'. https://dash.plotly.com/dash-core-components/tabs (accessed Oct. 29, 2020).

[26] 'Part 4. Interactive Graphing and Crossfiltering | Dash for Python Documentation | Plotly'. https://dash.plotly.com/interactive-graphing (accessed Oct. 29, 2020).

Appendix

Appendix 1

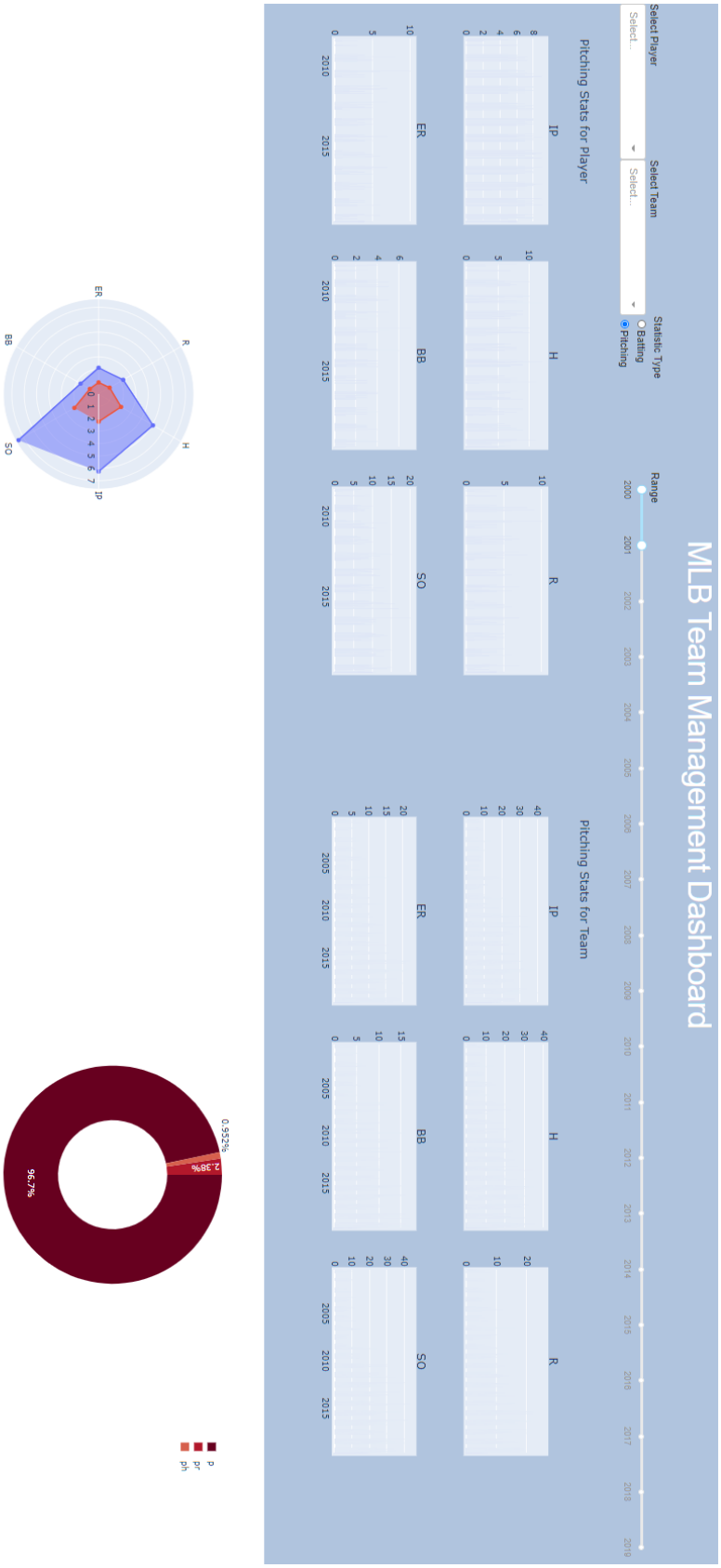


Figure 13 Full Dashboard

Appendix 2

Libraries Used

```
# import library and instance dashboard app
import sys
import pandas as pd
import re
import numpy as np
import scipy.io
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
# numpy.set_printoptions(threshold=sys.maxsize)
import dash  # (version 1.12.0) pip install dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output

#external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = dash.Dash(__name__, external_stylesheets=external_stylesheets)
```

Importing post extraction Data files

```
#import CSVs and clean data
#importing Batting and Pitching data
batting = pd.read_csv("Battings.csv",index_col=False, header=None, names=["Player", "Posit
pitching = pd.read_csv("Pitchings.csv",index_col=False, header=None, names=["Player", "Dat
# remove additional info
pitching['Player'] = pitching['Player'].str.replace(r'\([^()]*\)', '')
pitching['Player'] = pitching['Player'].str.strip()
batting['Position'] = batting['Position'].str.strip()
```

```
C:\Users\Alex\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3063: Dtyp

Columns (5) have mixed types.Specify dtype option on import or set low_memory=False.
```

```
# importing roster and combining with main dataframes
roster_file = open('2019combinedroster.txt', 'r')
roster_string = roster_file.read()
roster_df = pd.DataFrame([x.split(',') for x in roster_string.split('\n')])
```

```
roster_df['firstinitial'] = roster_df[2].astype(str).str[0]
roster_df["Player"] = roster_df[1] +" "+ roster_df["firstinitial"]
del roster_df[0]
del roster_df[1]
del roster_df[2]
del roster_df[3]
del roster_df[4]
del roster_df[6]
del roster_df['firstinitial']
roster_df = roster_df[["Player",5]]
roster_df.columns = ['Player', 'Team']
roster_df2 = roster_df
team_name_file = open('TEAM2019.txt', 'r')
team_name_string = team_name_file.read()
team_name_df = pd.DataFrame([x.split(',') for x in team_name_string.split('\n')])
del team_name_df[1]
team_name_df["teamName"] = team_name_df[2] +" "+ team_name_df[3]
del team_name_df[2]
del team_name_df[3]
team_name_df.columns = ['Team', 'teamName']
team_name_df2 = team_name_df

playerTeam = pd.merge(roster_df,  team_name_df,  on ='Team',  how ='left')
batting = pd.merge(batting,  playerTeam,  on ='Player',  how ='left')
del batting["Team"]
batting['teamName'].astype(str)

playerTeam2 = pd.merge(roster_df2,  team_name_df2,  on ='Team',  how ='left')
pitching = pd.merge(pitching,  playerTeam2,  on ='Player',  how ='left')
del pitching["Team"]
pitching['teamName'].astype(str)
```

```
0                         nan
1           Milwaukee Brewers
2                         nan
3                         nan
4                         nan
                  ...
408478      Cleveland Indians
408479      Cleveland Indians
408480    Washington Nationals
408481    Washington Nationals
408482    Washington Nationals
Name: teamName, Length: 408483, dtype: object
```

Visualisation Definitions

```
# building team and player input changes
#player level pitching and batting filtered by input
player_filter_pitching = pitching[pitching['Player']== name]
player_filter_batting = batting[batting['Player']== name]
```

```
# team level filtering
teamName = 'Detroit Tigers'
team_filter_pitching = pitching[pitching['teamName']== teamName]
team_filter_batting = batting[batting['teamName']== teamName]


#position of player filtered generation and visualisation for dashboard
def position_generator(name):
    player_filter_batting = batting[batting['Player']== name]
    player_position = player_filter_batting['Position'].value_counts(normalize=True) * 100
    values = player_position
    names = player_filter_batting['Position'].unique()
    fig = px.pie(values=values, names=names, color_discrete_sequence=px.colors.sequential.
    return(fig)


# batting radar player vs team generation and visualisation for dashboard
def batting_radar_PvT(name, teamName):
    player_filter_batting = batting[batting['Player']== name]
    team_filter_batting = batting[batting['teamName']== teamName]
    radar_batting_player = player_filter_batting.mean()
    radar_batting_team = team_filter_batting.mean()

    radar_batting_player = player_filter_batting.mean()
    radar_batting_team = team_filter_batting.mean()
    batting_categories = ['AB', 'R', 'H', 'RBI']

    fig = go.Figure()

    fig.add_trace(go.Scatterpolar(
          r=radar_batting_player,
          theta=batting_categories,
          fill='toself',
          name='Player'
    ))
    fig.add_trace(go.Scatterpolar(
          r=radar_batting_team,
          theta=batting_categories,
          fill='toself',
          name='Team'
    ))

    fig.update_layout(
      polar=dict(
        radialaxis=dict(
          visible=True,
        )),
      showlegend=False
    )

    return(fig)



# batting radar player vs team generation and visualisation for dashboard
def batting_radar_T(teamName):
```

```
    team_filter_batting = batting[batting['teamName']== teamName]
    radar_batting_team = team_filter_batting.mean()

    radar_batting_player = player_filter_batting.mean()
    radar_batting_team = team_filter_batting.mean()
    batting_categories = ['AB', 'R', 'H', 'RBI']

    fig = go.Figure()

    fig.add_trace(go.Scatterpolar(
          r=radar_batting_team,
          theta=batting_categories,
          fill='toself',
          name='Team'
    ))

    fig.update_layout(
      polar=dict(
        radialaxis=dict(
          visible=True,
        )),
      showlegend=False
    )

    return(fig)



# pitching radar player vs team generation and visualisation for dashboard
def pitching_radar_PvT(name, teamName):
    player_filter_pitching = pitching[pitching['Player']== name]
    team_filter_pitching = pitching[pitching['teamName']== teamName]
    radar_pitching_player = player_filter_pitching.mean()
    radar_pitching_team = team_filter_pitching.mean()
    pitching_categories = ['IP', 'H', 'R', 'ER', 'BB', 'SO']

    fig = go.Figure()

    fig.add_trace(go.Scatterpolar(
          r=radar_pitching_player,
          theta=pitching_categories,
          fill='toself',
          name='Player'
    ))
    fig.add_trace(go.Scatterpolar(
          r=radar_pitching_team,
          theta=pitching_categories,
          fill='toself',
          name='Team'
    ))

    fig.update_layout(
      polar=dict(
        radialaxis=dict(
          visible=True,
        ))
```

```
          )),
        showlegend=False
      )

      return(fig)


  # pitching radar team generation and visualisation for dashboard
  def pitching_radar_T(teamName):
      team_filter_pitching = pitching[pitching['teamName']== teamName]
      radar_pitching_team = team_filter_pitching.mean()
      pitching_categories = ['IP', 'H', 'R', 'ER', 'BB', 'SO']

      fig = go.Figure()

      fig.add_trace(go.Scatterpolar(
            r=radar_pitching_team,
            theta=pitching_categories,
            fill='toself',
            name='Team'
      ))

      fig.update_layout(
        polar=dict(
          radialaxis=dict(
            visible=True,
          )),
        showlegend=False
      )

      return(fig)


  # batting bar/time series player generation and visualisation for dashboard
  def batting_bar_P(name):
      player_filter_batting = batting[batting['Player']== name]
      fig1 = px.bar(player_filter_batting,x='Date', y="AB")
      fig2 = px.bar(player_filter_batting,x='Date', y="R")
      fig3 = px.bar(player_filter_batting,x='Date', y="H")
      fig4 = px.bar(player_filter_batting,x='Date', y="RBI")

      trace1 = fig1['data'][0]
      trace2 = fig2['data'][0]
      trace3 = fig3['data'][0]
      trace4 = fig4['data'][0]

      fig = make_subplots(rows=2, cols=2, shared_xaxes=True)
      fig.add_trace(trace1, row=1, col=1)
      fig.add_trace(trace2, row=1, col=2)
      fig.add_trace(trace3, row=2, col=1)
      fig.add_trace(trace4, row=2, col=2)

      fig.update_layout(
          autosize=False,
          width=1000,
          height=500,
```

```
            margin=dict(
                l=50,
                r=50,
                b=100,
                t=100,
                pad=4
            ),
            paper_bgcolor="LightSteelBlue",
        )

        return(fig)


    # batting bar/time series player generation and visualisation for dashboard
    def batting_bar_T(name):
        team_filter_batting = batting[batting['teamName']== teamName]
        team_filter_batting.groupby('Date')
        team_filter_batting['Date'] =pd.to_datetime(team_filter_pitching.Date)
        team_filter_batting.sort_values(by='Date')

        fig1 = px.bar(team_filter_batting,x='Date', y="AB")
        fig2 = px.bar(team_filter_batting,x='Date', y="R")
        fig3 = px.bar(team_filter_batting,x='Date', y="H")
        fig4 = px.bar(team_filter_batting,x='Date', y="RBI")

        trace1 = fig1['data'][0]
        trace2 = fig2['data'][0]
        trace3 = fig3['data'][0]
        trace4 = fig4['data'][0]

        fig = make_subplots(rows=2, cols=2, shared_xaxes=True)
        fig.add_trace(trace1, row=1, col=1)
        fig.add_trace(trace2, row=1, col=2)
        fig.add_trace(trace3, row=2, col=1)
        fig.add_trace(trace4, row=2, col=2)

        fig.update_layout(
            autosize=False,
            width=1000,
            height=500,
            margin=dict(
                l=50,
                r=50,
                b=100,
                t=100,
                pad=4
            ),
            paper_bgcolor="LightSteelBlue",
        )

        return(fig)


    # batting bar/time series team generation and visualisation for dashboard
    def pitching_bar_T(name):
        team_filter_pitching = pitching[pitching['teamName']== teamName]
```

```
    team_filter_pitching.groupby('Date')
    team_filter_pitching['Date'] =pd.to_datetime(team_filter_pitching.Date)
    team_filter_pitching.sort_values(by='Date')

    fig1 = px.bar(team_filter_pitching,x='Date', y='IP')
    fig2 = px.bar(team_filter_pitching,x='Date', y="H")
    fig3 = px.bar(team_filter_pitching,x='Date', y="R")
    fig4 = px.bar(team_filter_pitching,x='Date', y="ER")
    fig5 = px.bar(team_filter_pitching,x='Date', y="BB")
    fig6 = px.bar(team_filter_pitching,x='Date', y="SO")

    trace1 = fig1['data'][0]
    trace2 = fig2['data'][0]
    trace3 = fig3['data'][0]
    trace4 = fig4['data'][0]
    trace5 = fig5['data'][0]
    trace6 = fig6['data'][0]

    fig = make_subplots(rows=2, cols=3, shared_xaxes=True, subplot_titles=("IP", "H", "R",
    fig.add_trace(trace1, row=1, col=1)
    fig.add_trace(trace2, row=1, col=2)
    fig.add_trace(trace3, row=1, col=3)
    fig.add_trace(trace4, row=2, col=1)
    fig.add_trace(trace5, row=2, col=2)
    fig.add_trace(trace6, row=2, col=3)

    fig.update_layout(
        autosize=False,
        width=1000,
        height=500,
        title_text="Pitching Stats for Team",
        margin=dict(
            l=50,
            r=50,
            b=100,
            t=100,
            pad=4
        ),
        paper_bgcolor="LightSteelBlue",
    )
    return(fig)


# batting bar/time series player generation and visualisation for dashboard
def pitching_bar_P(name):

    player_filter_pitching = pitching[pitching['Player']== name]
    player_filter_pitching['Date'] =pd.to_datetime(player_filter_pitching.Date)

    fig1 = px.bar(player_filter_pitching,x='Date', y="IP")
    fig2 = px.bar(player_filter_pitching,x='Date', y="H")
    fig3 = px.bar(player_filter_pitching,x='Date', y="R")
    fig4 = px.bar(player_filter_pitching,x='Date', y="ER")
    fig5 = px.bar(player_filter_pitching,x='Date', y="BB")
    fig6 = px.bar(player_filter_pitching,x='Date', y="SO")
```

```
        trace1 = fig1['data'][0]
        trace2 = fig2['data'][0]
        trace3 = fig3['data'][0]
        trace4 = fig4['data'][0]
        trace5 = fig5['data'][0]
        trace6 = fig6['data'][0]

        fig = make_subplots(rows=2, cols=3, shared_xaxes=True,subplot_titles=("IP", "H", "R",
        fig.add_trace(trace1, row=1, col=1)
        fig.add_trace(trace2, row=1, col=2)
        fig.add_trace(trace3, row=1, col=3)
        fig.add_trace(trace4, row=2, col=1)
        fig.add_trace(trace5, row=2, col=2)
        fig.add_trace(trace6, row=2, col=3)

        fig.update_layout(
            autosize=False,
            width=1000,
            height=500,
            title_text="Pitching Stats for Player",
            margin=dict(
                l=50,
                r=50,
                b=100,
                t=100,
                pad=4
            ),
            paper_bgcolor="LightSteelBlue",
        )
        return(fig)


    # Visualisations checker
    # inputs and outputs for Dashboard
    # inputs
    #name = 'Scherzer M'
    #name = 'Stanton G'
    #teamName = 'Washington Nationals'
    #outputs
    #batting_radar_PvT(name, teamName)
    #pitching_radar_PvT(name, teamName)

    #position_generator(name)

    #batting_bar_P(name)
    #pitching_bar_P(name) #inserted

    #pitching_bar_T(name)
    #batting_bar_T(name)

    #pitching_radar_T(teamName)
    #batting_radar_T(teamName)
```

Dashboard

```
## Dashboard Layout
colors = {
    'background': "LightSteelBlue",
    'text': '#FFFFFF'
}
app.layout =  html.Div(style={'backgroundColor': colors['background']}, children=[
    html.H1(
        id='title',
        children='MLB Team Management Dashboard',
        style={
            'textAlign': 'center',
            'color': colors['text']
        }
    ),

    html.Div([
    html.Label('Select Player'),
    dcc.Dropdown(
        id='select-player',
        options=[
            {'label': 'Max Scherzer', 'value': 'Scherzer M'},
            {'label': 'Stephen Strasburg', 'value': 'Strasburg S'},
            {'label': 'Sean Doolittle', 'value': 'Doolittle S'},
            {'label': 'Brian Dozier', 'value': 'Dozier B'},
            {'label': 'Dan Jennings', 'value': 'Jennings D'}
        ],
        value=''),
    ], style={'width': '10%', 'display': 'inline-block', 'vertical-align': 'middle'}),

    html.Div([
    html.Label('Select Team'),
    dcc.Dropdown(
        id='select-team',
        options=[
            {'label':'Milwaukee Brewers','value':'Milwaukee Brewers'},
            {'label':'San Diego Padres', 'value':'San Diego Padres'},
            {'label':'Chicago Cubs','value':'Chicago Cubs'},
            {'label':'Houston Astros', 'value':'Houston Astros'},
            {'label':'Minnesota Twins', 'value':'Minnesota Twins'},
            {'label':'Philadelphia Phillies','value':'Philadelphia Phillies'},
            {'label':'St. Louis Cardinals', 'value':'St. Louis Cardinals'},
            {'label':'Miami Marlins', 'value':'Miami Marlins'},
            {'label':'Oakland Athletics','value':'Oakland Athletics'},
            {'label':'Toronto Blue Jays', 'value':'Toronto Blue Jays'},
            {'label':'Tampa Bay Rays', 'value':'Tampa Bay Rays'},
            {'label':'Detroit Tigers','value':'Detroit Tigers'},
            {'label':'Pittsburgh Pirates', 'value':'Pittsburgh Pirates'},
            {'label':'Seattle Mariners', 'value':'Seattle Mariners'},
            {'label':'Baltimore Orioles','value':'Baltimore Orioles'},
            {'label':'Washington Nationals', 'value':'Washington Nationals'},
```

```
            {'label':'Chicago White Sox', 'value':'Chicago White Sox'},
            {'label':'Anaheim Angels','value':'Anaheim Angels'},
            {'label':'Cleveland Indians', 'value':'Cleveland Indians'},
            {'label':'New York Mets', 'value':'New York Mets'},
            {'label':'New York Yankees','value':'New York Yankees'},
            {'label':'Atlanta Braves', 'value':'Atlanta Braves'},
            {'label':'Los Angeles Dodgers', 'value':'Los Angeles Dodgers'},
            {'label':'Colorado Rockies','value':'Colorado Rockies'},
            {'label':'Arizona Diamondbacks', 'value':'Arizona Diamondbacks'},
            {'label':'Boston Red Sox', 'value':'Boston Red Sox'},
            {'label':'Kansas City Royals','value':'Kansas City Royals'},
            {'label':'Texas Rangers', 'value':'Texas Rangers'},
            {'label':'Cincinnati Reds', 'value':'Cincinnati Reds'}
        ],
        value=''),
        ], style={'width': '10%', 'display': 'inline-block', 'vertical-align': 'middle'}),

    html.Div([
    html.Label('Statistic Type'),
    dcc.RadioItems(
        id='select-statistic',
        options=[
            {'label': 'Batting', 'value': 'Batting'},
            {'label': 'Pitching', 'value': 'Pitching'}
        ],
        value='Pitching'
    ),
    ], style={'width': '10%', 'display': 'inline-block', 'vertical-align': 'middle'}),

    html.Div([
    html.Label('Range'),
    dcc.RangeSlider(
    min=0,
    max=19,
    step=None,
    marks={
        0: '2000',
        1: '2001',
        2: '2002',
        3: '2003',
        4: '2004',
        5: '2005',
        6: '2006',
        7: '2007',
        8: '2008',
        9: '2009',
        10: '2010',
        11: '2011',
        12: '2012',
        13: '2013',
        14: '2014',
        15: '2015',
        16: '2016',
        17: '2017',
        18: '2018',
```

```
            19: '2019'},
        value=[0, 10]
        )
        ], style={'width': '70%', 'display': 'inline-block', 'vertical-align': 'middle'}),

        html.Div([
            dcc.Graph(id = 'pitch-bar-plot', figure = pitching_bar_P(name)),
            ], style={'width': '50%', 'display': 'inline-block', 'vertical-align': 'middle'}),

        html.Div([
            dcc.Graph(id = 'team-bar-plot', figure = pitching_bar_T(name)),
            ], style={'width': '50%', 'display': 'inline-block', 'vertical-align': 'middle'}),

        html.Div([
            dcc.Graph(id = 'pitch-radar-plot', figure = pitching_radar_PvT(name, teamName)),
            ], style={'width': '50%', 'display': 'inline-block', 'vertical-align': 'middle'}),

        html.Div([
            dcc.Graph(id = 'position-plot', figure = position_generator(name)),
            ], style={'width': '50%', 'display': 'inline-block', 'vertical-align': 'middle'}),

    ])
```

```
C:\Users\Alex\anaconda3\lib\site-packages\ipykernel_launcher.py:5: SettingWithCopyWar


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
```

```
# Dashbaord Running
if __name__ == '__main__':
    app.run_server(debug=False)
```

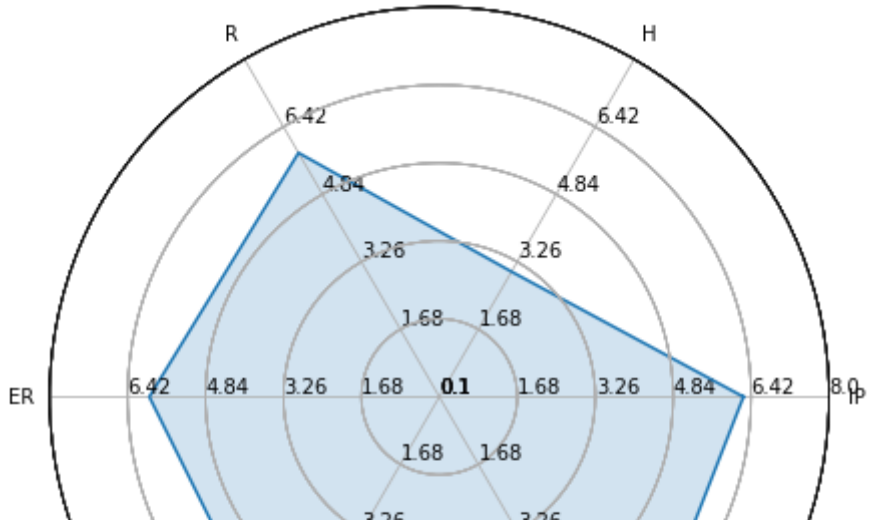
```
Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:8050/ (Press CTRL+C to quit)
127.0.0.1 - - [29/Oct/2020 12:53:18] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [29/Oct/2020 12:53:18] "GET /_dash-component-suites/dash_renderer/react
127.0.0.1 - - [29/Oct/2020 12:53:18] "GET /_dash-component-suites/dash_renderer/polyf
127.0.0.1 - - [29/Oct/2020 12:53:18] "GET /_dash-component-suites/dash_renderer/prop-
127.0.0.1 - - [29/Oct/2020 12:53:18] "GET /_dash-component-suites/dash_renderer/react
127.0.0.1 - - [29/Oct/2020 12:53:18] "GET /_dash-component-suites/dash_core_component
127.0.0.1 - - [29/Oct/2020 12:53:18] "GET /_dash-component-suites/dash_html_component
```

```
127.0.0.1 - - [29/Oct/2020 12:53:18] "GET /_dash-component-suites/dash_core_component
127.0.0.1 - - [29/Oct/2020 12:53:18] "GET /_dash-component-suites/dash_renderer/dash_
127.0.0.1 - - [29/Oct/2020 12:53:19] "GET /_dash-dependencies HTTP/1.1" 200 -
127.0.0.1 - - [29/Oct/2020 12:53:19] "GET /_favicon.ico?v=1.16.3 HTTP/1.1" 200 -
127.0.0.1 - - [29/Oct/2020 12:53:20] "GET /_dash-layout HTTP/1.1" 200 -
127.0.0.1 - - [29/Oct/2020 12:53:20] "GET /_dash-component-suites/dash_core_component
127.0.0.1 - - [29/Oct/2020 12:53:20] "GET /_dash-component-suites/dash_core_component
127.0.0.1 - - [29/Oct/2020 12:53:20] "GET /_dash-component-suites/dash_core_component
127.0.0.1 - - [29/Oct/2020 12:53:20] "GET /_dash-component-suites/dash_core_component
```

Altered Radar Plot Visualisation

```
# Alternative Radar plot with independent axis scale and direction
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns # improves plot aesthetics

player_filter_pitching = pitching[pitching['Player']== name]
radar_pitching_player = player_filter_pitching.mean()

def _invert(x, limits):
    """inverts a value x on a scale from
    limits[0] to limits[1]"""
    return limits[1] - (x - limits[0])

def _scale_data(data, ranges):
    """scales data[1:] to ranges[0],
    inverts if the scale is reversed"""
    for d, (y1, y2) in zip(data[1:], ranges[1:]):
        assert (y1 <= d <= y2) or (y2 <= d <= y1)
    x1, x2 = ranges[0]
    d = data[0]
    if x1 > x2:
        d = _invert(d, (x1, x2))
        x1, x2 = x2, x1
    sdata = [d]
    for d, (y1, y2) in zip(data[1:], ranges[1:]):
        if y1 > y2:
            d = _invert(d, (y1, y2))
            y1, y2 = y2, y1
        sdata.append((d-y1) / (y2-y1)
                     * (x2 - x1) + x1)
    return sdata

class ComplexRadar():
    def __init__(self, fig, variables, ranges,
                 n_ordinate_levels=6):
        angles = np.arange(0, 360, 360./len(variables))

        axes = [fig.add_axes([0.1,0.1,0.9,0.9],polar=True,
                label = "axes{}".format(i))
                for i in range(len(variables))]
```

```
            l, text = axes[0].set_thetagrids(angles,
                                             labels=variables)
            [txt.set_rotation(angle-90) for txt, angle
                 in zip(text, angles)]
            for ax in axes[1:]:
                ax.patch.set_visible(False)
                ax.grid("off")
                ax.xaxis.set_visible(False)
            for i, ax in enumerate(axes):
                grid = np.linspace(*ranges[i],
                                   num=n_ordinate_levels)
                gridlabel = ["{}".format(round(x,2))
                             for x in grid]
                if ranges[i][0] > ranges[i][1]:
                    grid = grid[::-1] # hack to invert grid
                              # gridlabels aren't reversed
                gridlabel[0] = "" # clean up origin
                ax.set_rgrids(grid, labels=gridlabel,
                              angle=angles[i])
                #ax.spines["polar"].set_visible(False)
                ax.set_ylim(*ranges[i])
            # variables for plotting
            self.angle = np.deg2rad(np.r_[angles, angles[0]])
            self.ranges = ranges
            self.ax = axes[0]
        def plot(self, data, *args, **kw):
            sdata = _scale_data(data, self.ranges)
            self.ax.plot(self.angle, np.r_[sdata, sdata[0]], *args, **kw)
        def fill(self, data, *args, **kw):
            sdata = _scale_data(data, self.ranges)
            self.ax.fill(self.angle, np.r_[sdata, sdata[0]], *args, **kw)


    if __name__ == "__main__":
        # example data
        variables = ("IP", "H", "R", "ER", "BB", "SO")
        data = radar_pitching_player
        ranges = [(0.1, 8), (8, 0.1), (8, 0.1),
                  (8, 0.1), (8, 0.1), (0.1, 8)]

        # plotting
        fig1 = plt.figure(figsize=(6, 6))
        radar = ComplexRadar(fig1, variables, ranges)
        radar.plot(data)
        radar.fill(data, alpha=0.2)
        plt.show()
```




```
player_filter_pitching = pitching[pitching['Player']== name]
fig1 = px.histogram(player_filter_pitching, x="IP")
fig2 = px.scatter(player_filter_pitching,x='Date', y="IP",log_x=True,log_y=True)

trace1 = fig1['data'][0]
trace2 = fig2['data'][0]


fig = make_subplots(rows=1, cols=2, shared_xaxes=True,subplot_titles=("Histogram", "Logari
fig.add_trace(trace1, row=1, col=1)
fig.add_trace(trace2, row=1, col=2)

fig.update_layout(
    autosize=False,
    width=1000,
    height=500,
    title_text="Pitching Stats for Max Scherzer",
    margin=dict(
        l=50,
        r=50,
        b=100,
        t=100,
        pad=4
    )
)
fig.show()
```

Appendix 3

```
clear all
close all
clc

%% Read in file

fileID = fopen('20002019Seasons.txt'); % open file

fileID = fopen('20002019Seasons.txt', 'r');

i = 1;
while ~feof(fileID) %test for end of file
    file{i,1} = fgetl(fileID);
    i = i+1;
end
fclose(fileID);

clear fileID

%% Loop through ...

i = 1; % i is the line number for the text file
j = 1; % j is the line number for battings
k = 1; % k is the line number for pitchings
readBattings = 0;
readPitchings = 0;
while i < length(file)
    i
    line = file{i};

    % Each data set starts with the 'Game' line
    % Does 'Game' appear in line
    % If 'Game' appears, we want to start reading batting data
    if xor(contains(line, 'Game'), contains(line, 'Game!'))
        readBattings = 1;

        % Extract date
        dd = strfind(line, '/');
        Date = line(14:max(dd)+4);

        i = i+3;
```

```
  continue
end

% Does '-- --' appear (i.e. final line)
if contains(line, '-- --')
  readBattings = 0;
   i = i+1;
  continue
end

if readBattings == 1
   % only read team 1 player if present
   if strcmp(line(1:22),'                      ') == 0
      Battings(j).Player = line(1:22);
      Battings(j).Date   = Date;
      Battings(j).AB     = str2num(line(23:25));
      Battings(j).R      = str2num(line(26:28));
      Battings(j).H      = str2num(line(29:31));
      Battings(j).RBI    = str2num(line(32:34));
      j = j+1;
   end

   % If no further play on team 2, skip
   if length(line) < 40
      i = i+1;
      continue
   end

   Battings(j).Player = line(36:57);
   Battings(j).Date   = Date;
   Battings(j).AB     = str2num(line(58:60));
   Battings(j).R      = str2num(line(61:63));
   Battings(j).H      = str2num(line(64:66));
   Battings(j).RBI    = str2num(line(67:69));
   j = j+1;
end

% Search for start of pitching data
if contains(line, 'IP  H  R ER BB SO')
   readPitchings = 1;
   readPitchings_first = 1;
   i = i+1;
   continue
```

```
    end

    if readPitchings == 1

        if length(line) == 0;
            line = ' ';
        end

        if line(1) == ' '
            if readPitchings_first == 1
                readPitchings_first = 0;
                i = i+2;
                continue
            else
                readPitchings = 0;
                i = i+1;
                continue
            end
        end

        Pitchings(k).Player = line(1:22);
        Pitchings(k).Date   = Date;
        Pitchings(k).IP     = str2num(line(23:25));
        Pitchings(k).H      = str2num(line(26:28));
        Pitchings(k).R      = str2num(line(29:31));
        Pitchings(k).ER     = str2num(line(32:34));
        Pitchings(k).BB     = str2num(line(35:37));
        Pitchings(k).SO     = str2num(line(38:40));
        k = k+1;
    end

    i = i+1;
end

%% Export Pitchings

fileID = fopen('Pitchings.csv', 'w');
    for i = 1:length(Pitchings)
        textLine = [Pitchings(i).Player ',' Pitchings(i).Date ',' num2str(Pitchings(i).IP) ','
num2str(Pitchings(i).H) ',' num2str(Pitchings(i).R) ',' num2str(Pitchings(i).ER) ','
num2str(Pitchings(i).BB) ',' num2str(Pitchings(i).SO) '\n'];
        fprintf(fileID, textLine);
    end
```

```
fclose(fileID);
%% Export Battings
fileID2 = fopen('Battings.csv', 'w');
   for i = 1:length(Battings)
      textLine = [Battings(i).Player ',' Battings(i).Date ',' num2str(Battings(i).AB) ','
num2str(Battings(i).R) ',' num2str(Battings(i).H) ',' num2str(Battings(i).RBI) '\n'];
      fprintf(fileID2, textLine);
   end
fclose(fileID2);
```