Fantasy Football - Process Book

Raj Patel — u0902022 — raj.patel@utah.edu
Jake Maschoff — u0581770 — u0581770@utah.edu
Blaze Kotsenburg — u0676062 — u0676062@utah.edu

Github: https://github.com/maschoffjake/dataviscourse-pr-fantasyfootball
Website: https://fantasyfootball.unityx.io/
Video: https://www.youtube.com/watch?v=encb477zWE8

Overview & Motivation

The basis of fantasy football is for you to draft players for certain positions, exclusive to quarterback, running-back, wide-receiver, tide-end, a team's defense, and kicker. You then compose your own team with the players that you draft to play against other people's teams in your fantasy league. If your team scores more fantasy points than your opponent for that week, you beat your opponent. We will be focusing on only offensive players for this project because the dataset we are using only contains offensive players.

There are two different scoring techniques used for fantasy football, points per reception (PPR) and Fantasy Points. The data that we are using includes both Fantasy Points and PPR scoring techniques. Fantasy Points scoring is as follows:

- Passing Yards: **1 point** per 25 yards
- Passing Touchdowns: **4 points**
- Passing Yards (for non-QBs): **1 point** per 10 yards
- Passing Touchdowns (for non-QBs): **6 points**
- Interceptions: **-2 points**
- Fumble Lost: **-2 points**
- Rushing Yards: **1 point** per 10 yards
- Rushing Touchdowns: **6 points**
- Receiving Yards: **1 point** per 10 yards
- Receiving Touchdowns: **6 points**

PPR scoring follows the same scoring as Fantasy Points except that there is one additional scoring rule:

- Receptions: **1 point**

Picking which players to draft and which players to play for a certain week can widely change. That's what we would like our visualization to help with. You can analyze a player's historic stats and given these stats, you can help make a draft and game-time decisions. The given statistics will allow a user to analyze a player's progression over the past years, check player comparisons, and view a team's total fantasy points.

Data

Initially, we were planning on getting the data via the Pro Football Reference website. However, that required scraping and filtering the important data we wanted to use [2]. Doing this would require quite a bit of time which we could use for the visualization. Thus, we did some more research and found data that was already aggregated [3]. The dataset we chose already contained attributes that were the most important to us.

The data that we are using is already in CSV format. We will be using D3 to parse the data for visualization. The data, though, had some errors that we had to clean up. The first issue with the data was duplicate names for players. For example, there are multiple Adrian Peterson's and Alex Smith's. That led to an inability to differentiate between players via names, which is what we used as a unique identifier. To fix this we had to manually go in and find the players with duplicate names. Through some research, we were able to tag duplicate players with an underscore and a number. For example, the Adrian Peterson's in our data set are now represented as 'Adrian_Peterson_1' and 'Adrian_Peterson_2'. This took some time because we had to do it manually due to the fact that players can change teams mid-season and positions. This led us not being able to automate this data preprocessing.

Along with the duplicate player names issue, some of the data we needed to change due to it being inconsistent with other player data. Such as players with no position. Since our data relies heavily on the position that the player is, we decided to dump all players with no given position throughout their entire NFL career. Along with that, there are points in the data where there is no value listed for a specific statistic for a player. In that case, we just assigned the player a value of 0 for that stat.

The data will be parsed into JSON array of Player object as follows.

- Player name
- Years
    - Team
    - Position

- Games played for the given year
- Games started for the given year
- Total Fantasy Points
- Total PPR Points
- Total Points per Game (PPG)
- Total PPR per Game (PPRPG)
- Position Rank for the given year
- Receiving Object
    - Target
    - Receptions
    - Yards
    - Avg. Yards per Receptions
    - Touchdowns
- Passing Object
    - Completions
    - Attempts
    - Yeards
    - Touchdowns
    - Interceptions
- Rushing Object
    - Attempts
    - Yards
    - Avg. Yards per Attempts
    - Touchdowns

We changed our data layout to include an array of 'Years' objects. This was changed in order to make integrating data extraction with our year-selector design easier. Each value within the 'Years' array is an object of the player's data for a given year. You can see the data parsed within our application:

Figure 1. Parsed data.

Exploratory Data Analysis

We did a lot of research into exploring APIs to receive NFL data. We ran into a lot of issues because we weren't able to obtain an API key to use a lot of the APIs that we were originally interested in. Our original plan was to use live data for the project which would have given more helpful insight into the player stats. Since we were not able to get a key however, we began looking for historic datasets instead. We eventually found one that provided enough data for us to make a meaningful data visualization. Our data is based on historic data from the NFL from 2008 to 2018.

Design Evolution

For our visualization we have three main components: spider charts, line graphs, and overall view. The spider charts are being used to visualize single year's data of the players. The line graphs are to visualize multiple years data of the players. Finally, overall view is being used to visualize the selected players' statistics with all the other players.

Initial Design

Our initial design broke down the visualization to two views: player view and overall view. On top of the view, we put all of the interactive component such as the year selector, player selector, etc. Since we have data for multiple years, it was important for us to have the user be able to choose either between a single year or multiple years. We found a year selector with brush to be the best way to let user select the years (as seen in top of Fig. 2). We also wanted the user to be able to switch between either a team view and player view via a toggle. For the player selector, we thought a searchable dropdown to be the easiest to use.

Fig. 2 shows what the player view would look like for a single year. Below the player selector, there are view static text which would show the players rank, fantasy points, PPG, PPRPG, and PPR. Below that, bar charts are used to display the players stats that could be categorized (such as touchdowns made via rushing, receiving, and passes). Below that, we had more text that would display the players other stats such as the number of attempts made during passes.

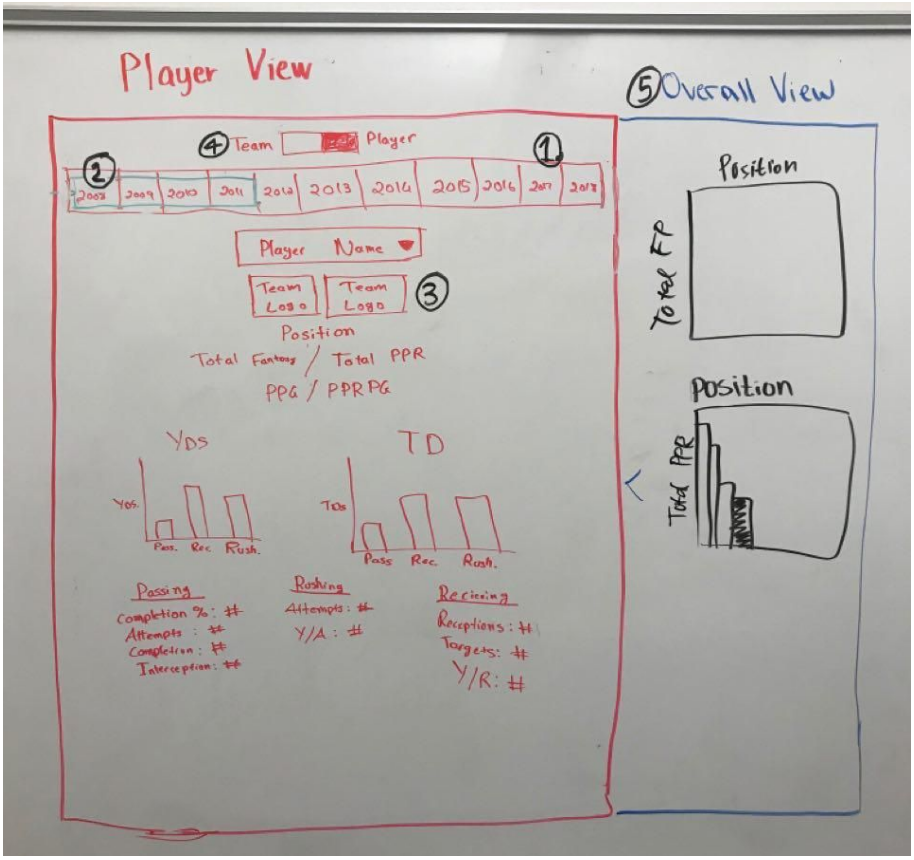

Figure 2. Initial design of visualization for single year.

Fig 3. shows the player view for multiple views. For the multiple view, most of the static texts are removed and line graphs are added for each attribute of the data. Each line graph could be hidden via a toggle. A year axis would be at the top of the line graph to visualize where the data points are for the selected years.

We also added an overall view to compare the selected player with all the other players as shown in Fig. 3 and 4. We thought about adding normal graphs for different attributes such as position vs fantasy points. We didn't know how many we could graphs we could fit a the moment so we just showed two in the figures.

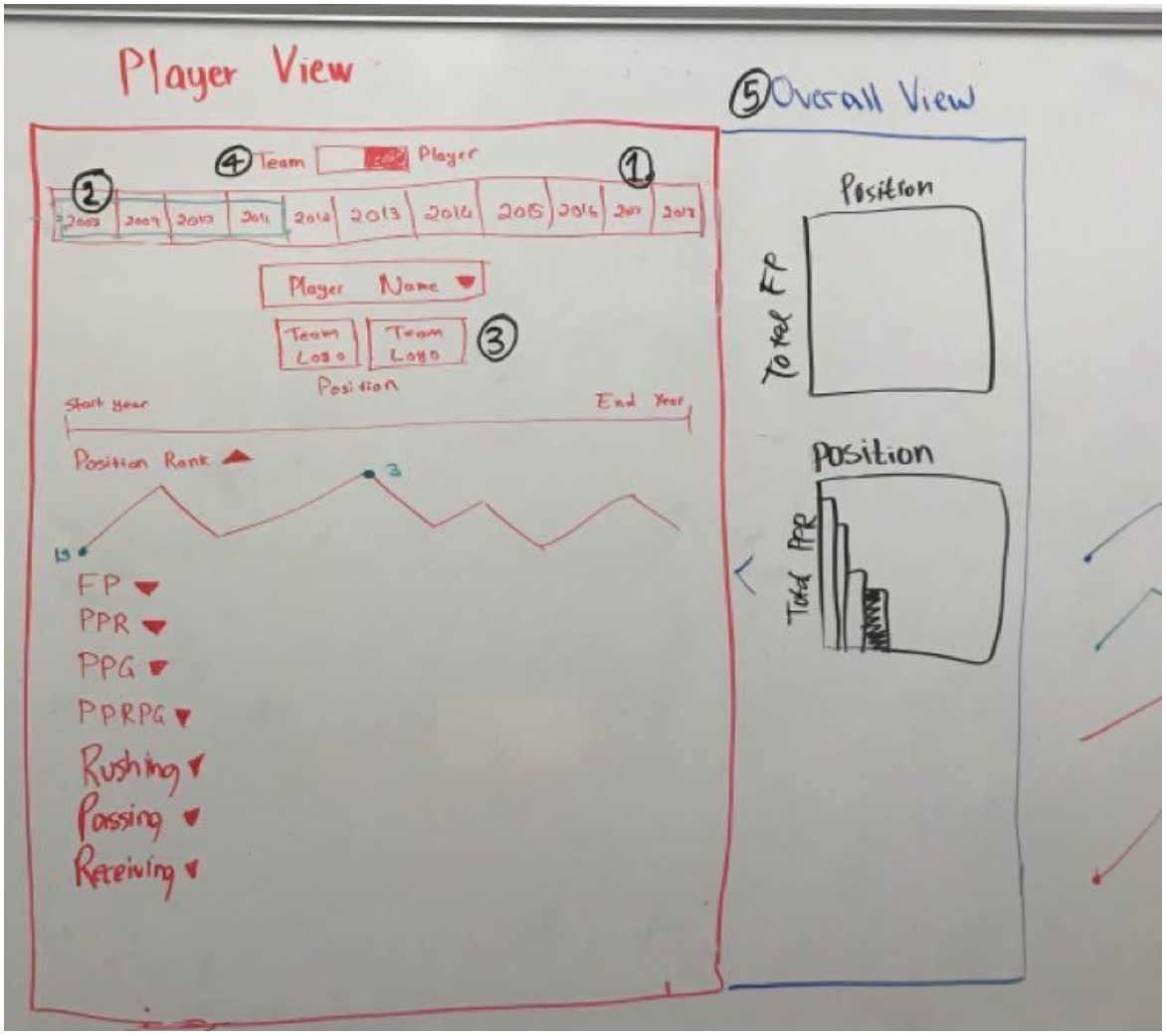

Figure 3. Initial design of player view for multiple years.

Another important part for us was to have the ability to compare players. In order to toggle compare player, we added a button on the top. Fig. 4 shows two players in compare mode for a single year. Bars are used to display each players attribute that could be easily comparable. Players are color coded. Besides the bars, not much is going on for single year comparison.

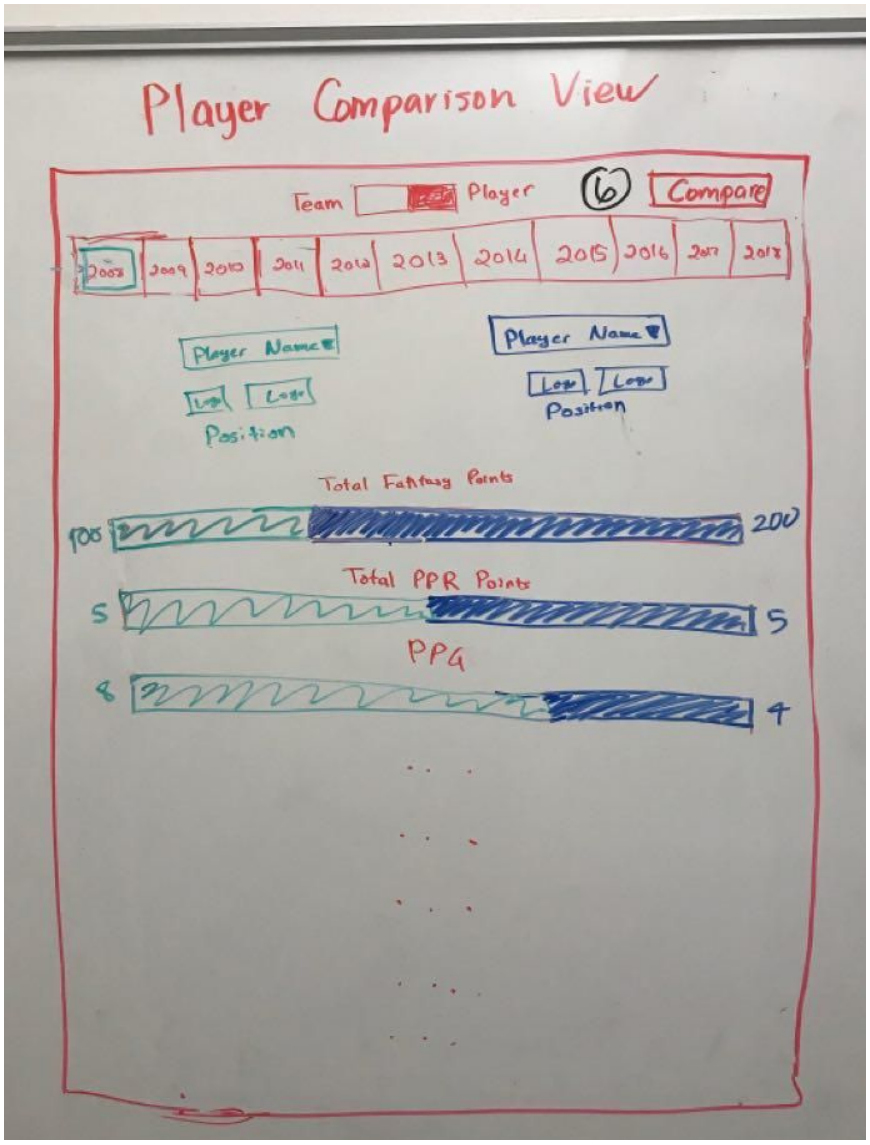

Figure 4. Comparing two players for single year.

Fig. 5 shows two players being compared for multiple years. The bars used for single year comparison would not make sense for multiple years because we would have to average the data. Averaging the data would lose finer details that could be useful to the user when comparing players. Thus, we decided to go with line graphs for each attributes similar to multiple years view when not comparing two players. For story telling, we were also thinking of showing the extreems on the line graph.

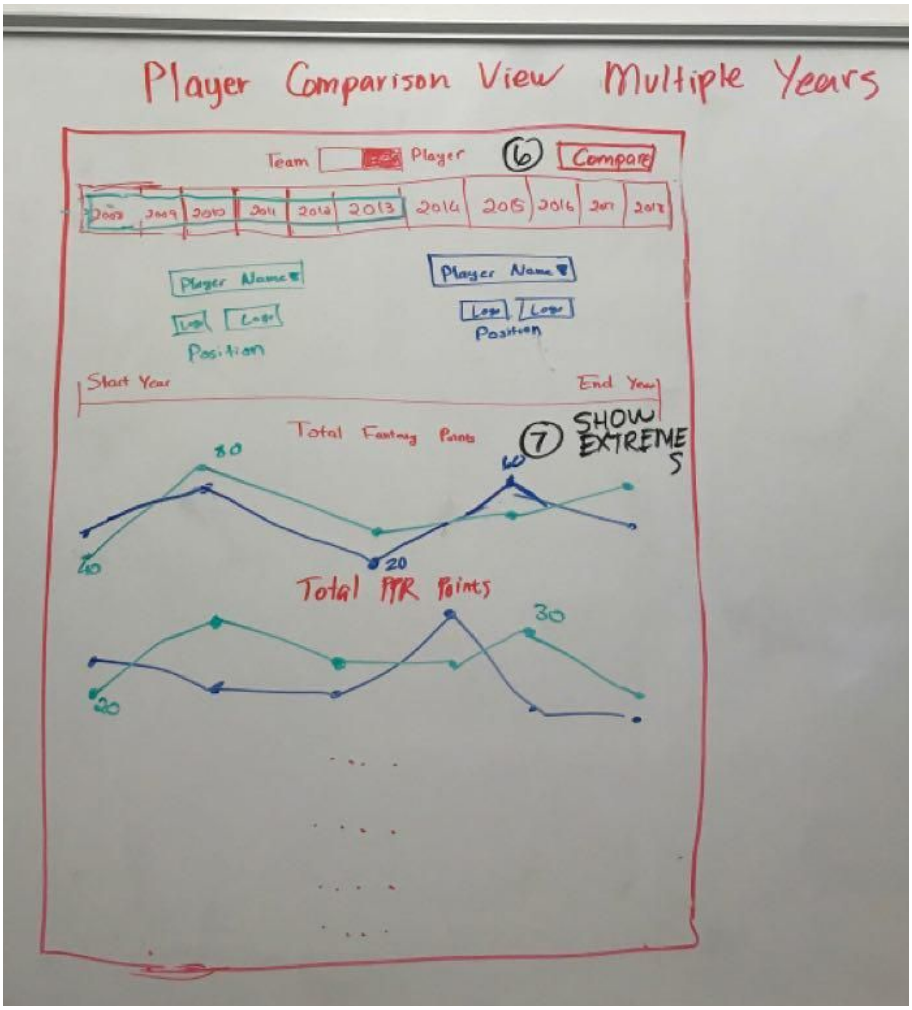

Figure 5. Players being compared for multiple years.

Design Revaluation & Final Design

Our initial design had major flaws that we discovered once we had a closer look at the data. Along with this, other flaws were also discovered when we met with the TA. The first thing we decided to change was the static texts. We also thought that switching between player view and team view was redundant and not necessary, so we removed the toggle button.

Switching between single year and multiple years was also a bit complicated in our initial design. All of the components had to be removed/hidden and new things needed to be added which complicated implementation and was confusing. Thus we splified our new design to incorporate everything in one view.

We replaced the static texts for single year with spider charts. We categorized the attributes to points, receiving, rushing, and passing for all the players. Fig. 6 shows the spider chart for the points category. In the figure, two players are being displayed, one in

red and other in blue. The spider charts can be rotated by clicking on the attribute name. Doing this will also display some storytelling aspect of our view as seen in Fig. 6.

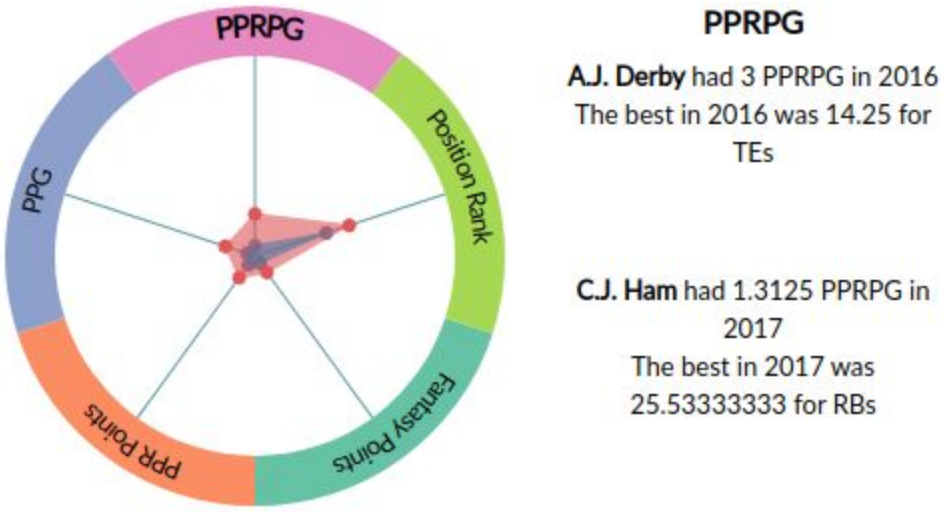

Figure 6. Spider chart displaying two players data for single year.

The line graphs in our final design has mostly been same compared to our initial design. However, we did remove the ability to hide certain attributes as it was not necessary. Fig. 7 below shows the visualization. The dotted lines are for player 2 in compare mode. On hover, a tooltip is shown to display the data and what the line correspond to. The lines are also color coded to match the attribute shown in the spider chart (Fig. 6). We did not add a Y-Axis to our line graph because we wanted to show the trend of the players over the years.

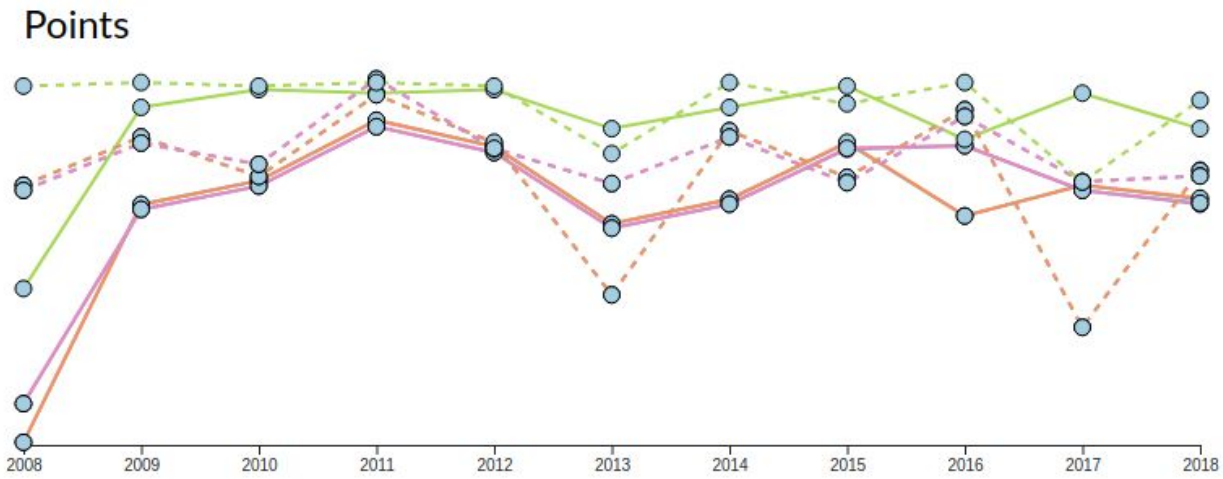

Figure 7. Line graphs for multiple years.

We also simplified our overall view by a lot. Instead of having multiple graphs, we decided to have the axis be changeable by the user. This allowed for a larger graph that is able to better visualize all of the players (as seen in Fig. 8). The overall view only shows the players of the same position since that would be able to give the most information about that selected player compared to others. Each circle has a tooltip which shows the player and few of their stats. Each of the circles below is color coded based on their position. In compare mode, if the selected player 2 played in a different position than other circles would also be shown in their corresponding colors for the position.

In addition to this, we also added a storytelling component which is toggled by a button. The storytelling shows the extreme points that show the highest ranked players for the selected players positon.

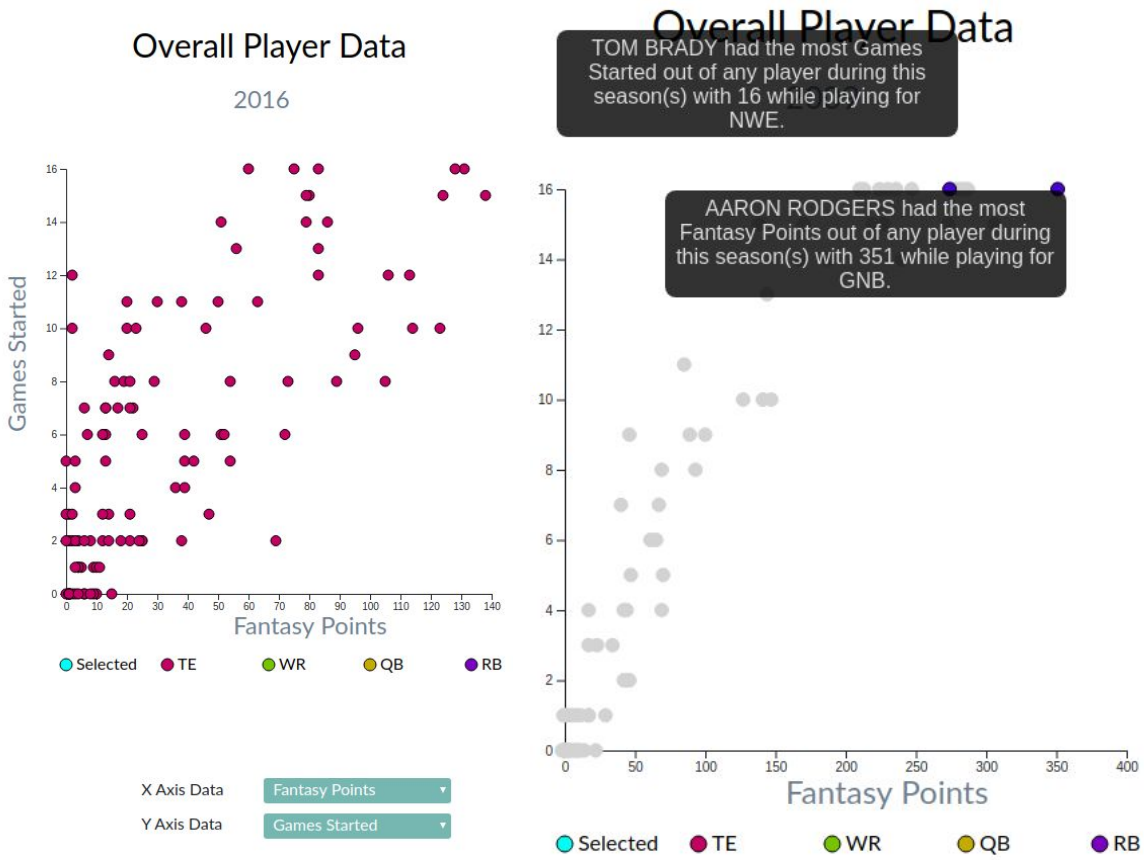

Figure 8. Overall view in final design along with storytelling component.

Fig. 9 shows all of the components together in single player view and compare mode. On the left we have compare toggle that adds another dropdown player selector. Under the player selector, we have a year selector which can be brushed. Brushing over the years updates the line graphs and the overall view data. Clicking on the circle for year selector updates the spider chart. Clicking on a circle in the overall view updates the

selected player 1. All of the views are connected to each other with the player and year selector controlling most of the things.

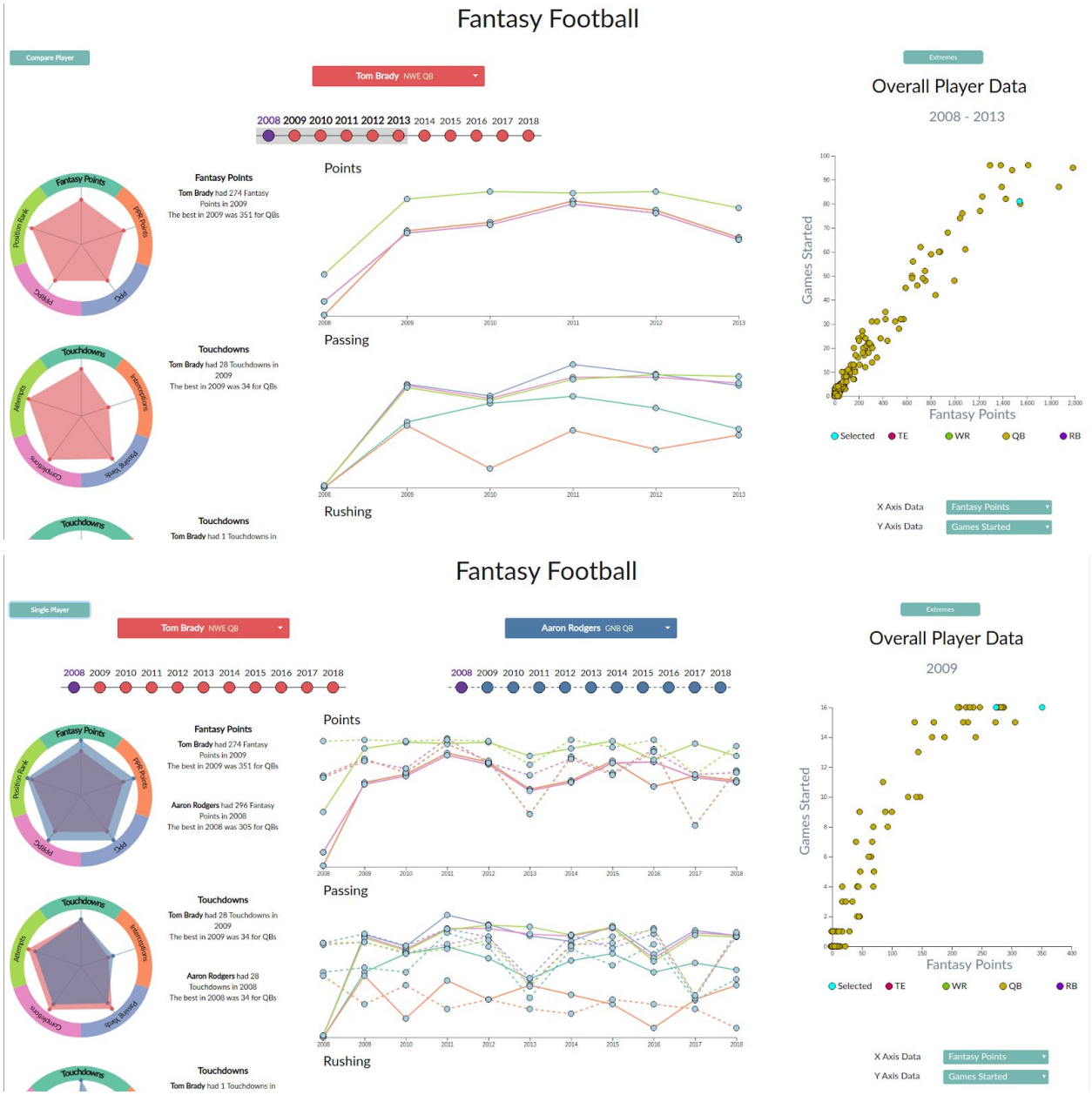

Figure 9. All the components together with single player and compare mode. .

Related Work

For our year slider, we got our inspiration from Homework 6 (Fig. 10). We really liked the idea of the user having to brush over something in order to produce specific data and cause live updates to their data being displayed.

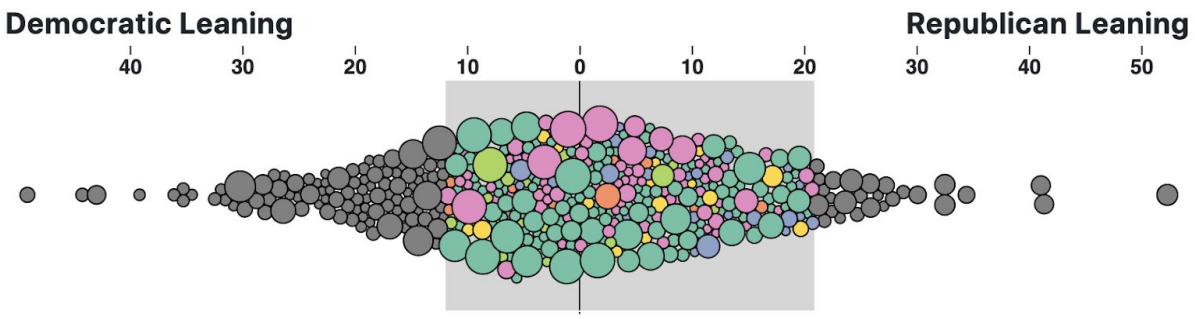

Figure 10. Selecting circles via brush from homework 6.

Another item that we got inspiration for was the spider chart. As mentioned, we decided to go with a spider chart instead of individual bar charts since it was hard to display that mean bar charts for so many different stats. With a spider chart, we were able to show all of these statistics in a much more condensed area. When creating and implementing our spider chart we really like the way this one looked:

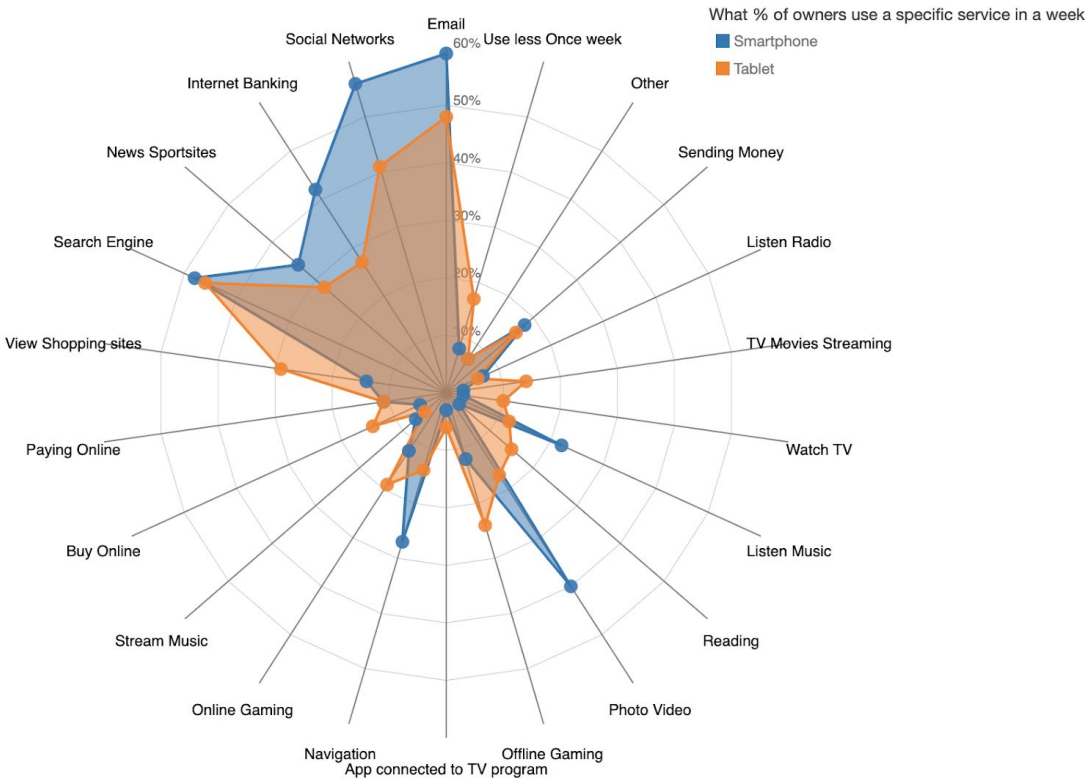

Figure 11. Spider chart we saw in our research. (http://bl.ocks.org/nbremer/6506614)

The way that there is an outer circle, lower opacity areas, and points for each of the values makes it look very clean. A lot of our design inspiration came from this design.

Implementation Process

We began our project by finding a dataset that fit our needs for our project goal. The dataset we found was originally in csv format, so we parsed the data into a JSON object to make accessing the data more organized. Once we had the data parsed, we began to create our views and dropdowns to allow the user to access and manipulate the data.

Once we had a general idea of what we wanted our data to look like, we each split into our own tasks/modules. Raj was assigned to the Player class and was in charge of creating dropdowns, line graphs, and the year selector. Jake was also assigned to the Player class and dealt with the logic and design of the spider charts. Finally, Blaze put together the Overall class which displayed data in a scatterplot. Below is a description of how each module was implemented.

For most of the implementations, we have been organized and have tried separating each component to their own functions. We used D3 to manipulate all of our

visualizations along with jQuery to manipulate containers that D3 could not. We used CSS for most of our styling and Javascript for creating classes. For the implementation, we broke everything into three classes: main, player view, and overall view. Along with this, we have a script file which initiates the three classes and parses the data.

Overall Class

The idea of the Overall class is to allow a user to see the overall data in the NFL for every player that shares the same position as the selected player(s). These players also have to have played in the NFL during the same selected time span. This class shares several callbacks with the player class so that both classes can communicate with each other and manipulate the data being displayed.

The Overall class allows a user to select the data that they want to see for each axis. Every data point has a tooltip that gives a short summary of that player and their data for each axis. A player can also be selected from the scatter plot, which in turn, manipulates the data in the Player view. Data points in the scatter plot are color coded based on the players position (for comparing players) and a color for the current selected players. Finally, the scatter plot will provide the extremes of the chart with a toggle button at the top. This button shows the player with the highest value in the x-axis and the player with the highest view in the y-axis. This can be viewed in Figure 8 above.

Player Class

The Player class shows data for a player over the time span that they played in the NFL. This view also allows users to compare two players together. Players of any position can be compared against each other. There are a lot of different data points that we wanted a user to be allowed to view, so we had to get a little bit creative with some of the charts. We chose to use spider charts to help compare the stats between two players for a single year. Refer to Figure 6.

Since the spider charts only focus on comparing the stats of players for a single year, we also needed a way to compare players over multiple years. We ended up doing a line chart (Figure 7) for this so that we could display each players progression over each season they played. This allows a user to determine if a player was continuing to excel or not.

Evaluation

Overall, the project was a great success. The outcome of our project was much better than our initial design, and portrays the data exactly how we wanted to. We really believe that this tool will help people decide on players that they will draft in future

fantasy drafts. Going forward we would like to incorporate future data sets as the public data keeps growing for the NFL.

References

[1] https://www.sportsbusinessdaily.com/Journal/Issues/2019/09/02/Media/Fantasy.aspx

[2] https://www.pro-football-reference.com/

[3] https://www.reddit.com/r/fantasyfootball/comments/cbwmvy/heres_a_spreadsheet_with_10_years_of_fantasy_data