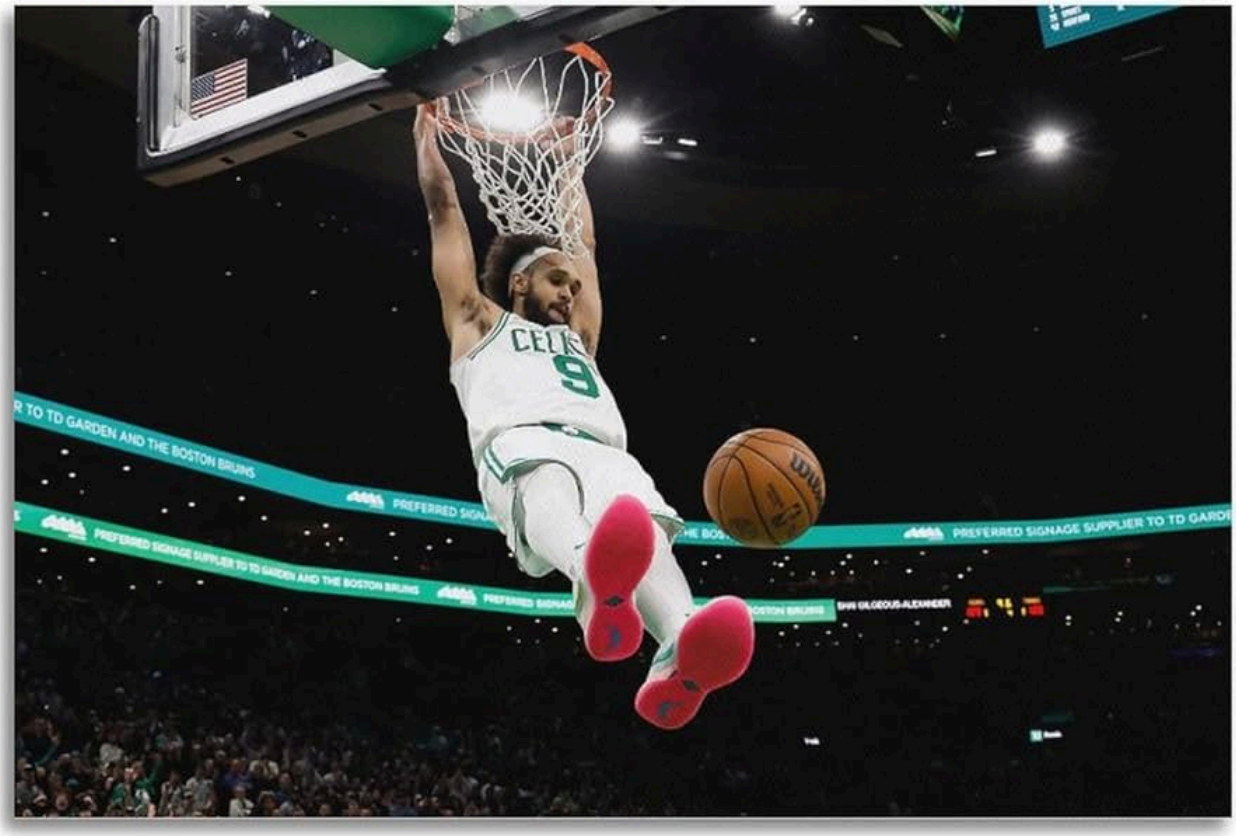


NBA Stats Website Featuring Derrick White

Nicholas Markou, Patrick Salisbury, Eric Zhou

Overview & Motivation

The focus of this project was to utilize the NBA API to develop our own sports visualization website. The visualizations we created fall into two main categories: those that we had already seen on other sports visualization sites and recreated mainly to understand how they work, and those that are more “experimental” and were created as more of a prototype to see how effective they would actually be. These visualizations are organized into a website using React for the front end and Express for the back end to serve the visualization pages, store data, and make calls to the NBA API.

There were a few sources of motivation for this project from the get-go. Every team member has an interest in basketball, and this interest extended naturally into exploring sports visualization during the project. The sports visualization field is a relatively recent one, and as a team we think there is likely a lot more potential for creativity and new ideas left in the field (which could be very profitable given how profitable the sports industry as a whole is). Another key interest was that the team wanted to see how hard it actually was to create an effective sports visualization site and work with the NBA API (spoiler alert, the API is very painful to work with). Finally, the team had already experimented with NBA data in assignment 3, so expanding it out into a full-fledged website seemed like a natural continuation for the final project.

Related Work

One of our major inspirations for our project were some of the NBA stats sites that we used as material during in-class reflections. Two major examples include Dunks and Threes (https://dunksandthrees.com/player/1627759) and Bucket List (https://bucketlist.fans/player/nba/1628436). In particular, the shot chart from Bucket List is one of the visualizations that we were inspired by and ended up improving upon for our project.

Another visualization that caught our attention was a reflection shared in class. The creators of the visualization utilized a parallel coordinate plot to plot various foods from the USDA nutrient database based on various nutritional values (https://syntagmatic.github.io/parallel/). One of the key features that really stood out was their ability to drag along an axis to select a custom range on each axis, and then filter the data to only include food entries that satisfied the new ranges. This idea of a parallel coordinate plot occurred as a great potential way to compare the stats of various NBA players.

Questions

In the beginning of the project, the main goal was to essentially create our own version of a sports visualization site to see exactly how hard it would be to do so. This question is what drove us to develop the shot chart visualization that we first used as a part of a3. Shot charts “caught our eye” from a visual perspective, but we weren’t sure exactly how effective they were for actually conveying trends in data, so a3 allowed us to perform an experiment to test the efficacy of this type of visualization.

As the project progressed, we developed sub-questions that we set out to answer by the time the project was complete. By the time we had a better (interactivity, more features) shot chart than the one shown in Bucket List, the goal shifted from just creating a stats site to seeing how hard it would be to create sports visualizations that were better than the ones we saw on some of the websites we looked at. Another question we tried to address during the project was how to create a stats site that would be less confusing to someone less familiar with the sport than what we had seen before. In particular, the Dunks and Threes website is essentially just a wall of data that would probably be overwhelming to someone unfamiliar with basketball, so we tried to address this problem in our visualizations by incorporating features like interactivity and color design into our visualizations to make them more user-friendly.

Data

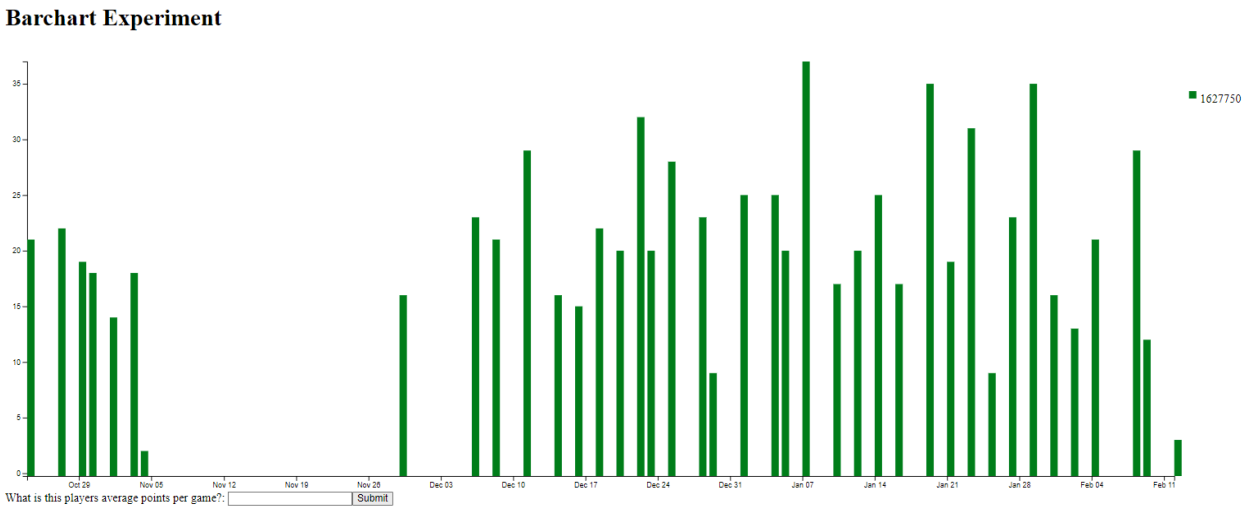
All of the data that we used during this project was collected from the hidden NBA API. Unfortunately, this API was not so simple to use, so we had to use quite a few workarounds and hacks to be able to use the API for our own stats website. The first (and probably the most challenging) aspect of the API was that nearly all commercial IP addresses are blocked from being able to make API calls. Consequently, we had to come up with an alternative hosting method for the final site that deviated from the standard approach (GitHub Pages was blocked, DigitalOcean was blocked, everything else we could think of was blocked). In the end, the team leveraged a server located on the WPI network (as well as frequent access from our home networks during the design/implementation stages) to access the API. Additionally, the team was required to use an express.js backend as the API rejected non-NBA.com origin requests because of the CORS policy. Utilizing the headers below on the express.js server successfully got us around the CORS issue

```
fetch('https://stats.nba.com/stats'+req.url.substring(4), {
    headers: {
        'Accept': '*/*',
        'Accept-Language': 'en-US,en;q=0.9',
        'Connection': 'keep-alive',
        'Origin': 'https://www.nba.com',
        'Referer': 'https://www.nba.com/',
        'Sec-Fetch-Dest': 'empty',
        'Sec-Fetch-Mode': 'cors',
        'Sec-Fetch-Site': 'same-site',
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36',
        'sec-ch-ua': '"Not A(Brand";v="99", "Google Chrome";v="121", "Chromium";v="121"',
        'sec-ch-ua-mobile': '?0',
        'sec-ch-ua-platform': '"Windows"'
    }
```

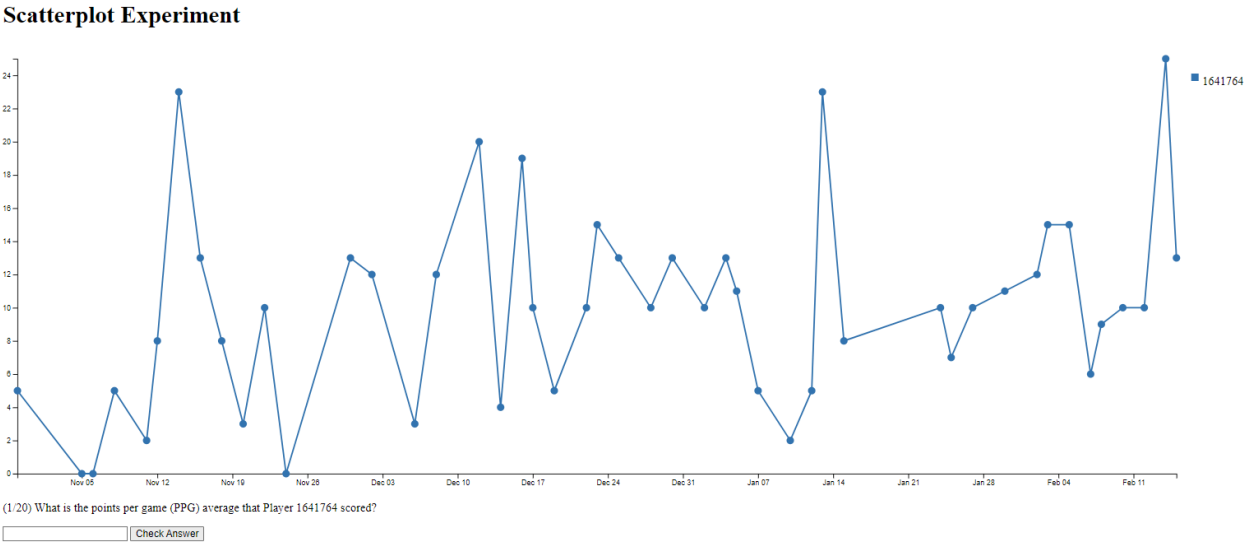
Even after getting around the blocked IP and CORS issues, the team still had issues with getting temporarily rate limited by the visualizations that could make many API calls in a short amount of time. The workaround used for this was to use the server to store some data locally that was frequently used, and then have any request for that data taken from the server instead of making another call to the NBA API.

Exploratory Data Analysis

Because we weren't analyzing brand new data (in fact, this data is so popular that the NBA API makes it hard to get a hold of on purpose), we mainly explored the data initially through the stats websites described earlier on. These other sites gave us a general understanding of what data we could actually get through the API and what types of visualizations we could focus on. Another way we explored the data early on was through using it during a3 to pull random points-per-game from for our more simple visualizations, with the goal being to have a user guess the average PPG. This allowed us to get some practice with the basics of using the data returned by the API before we created more advanced visualizations with the data available.



Guessing a random player's PPG with a Bar Chart



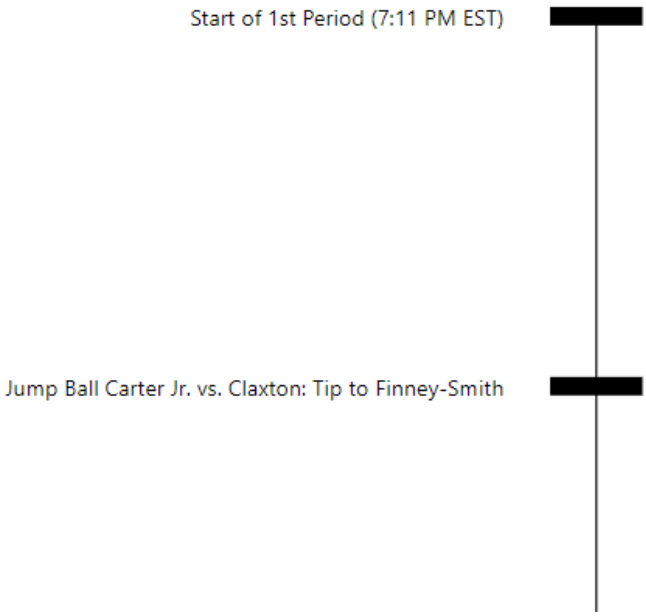
Guessing a random player's PPG with a Scatterplot

Finally, a big (and often overlooked) way that we explored the data initially was just by examining the JSON responses that were actually returned by the API. Although this does take some technical know-how to understand exactly what is going on, reading the responses themselves allowed us to see all of the possible data that we had access to from a given API call and brainstorm/design new types of visualizations based on this insight.

Design Evolution

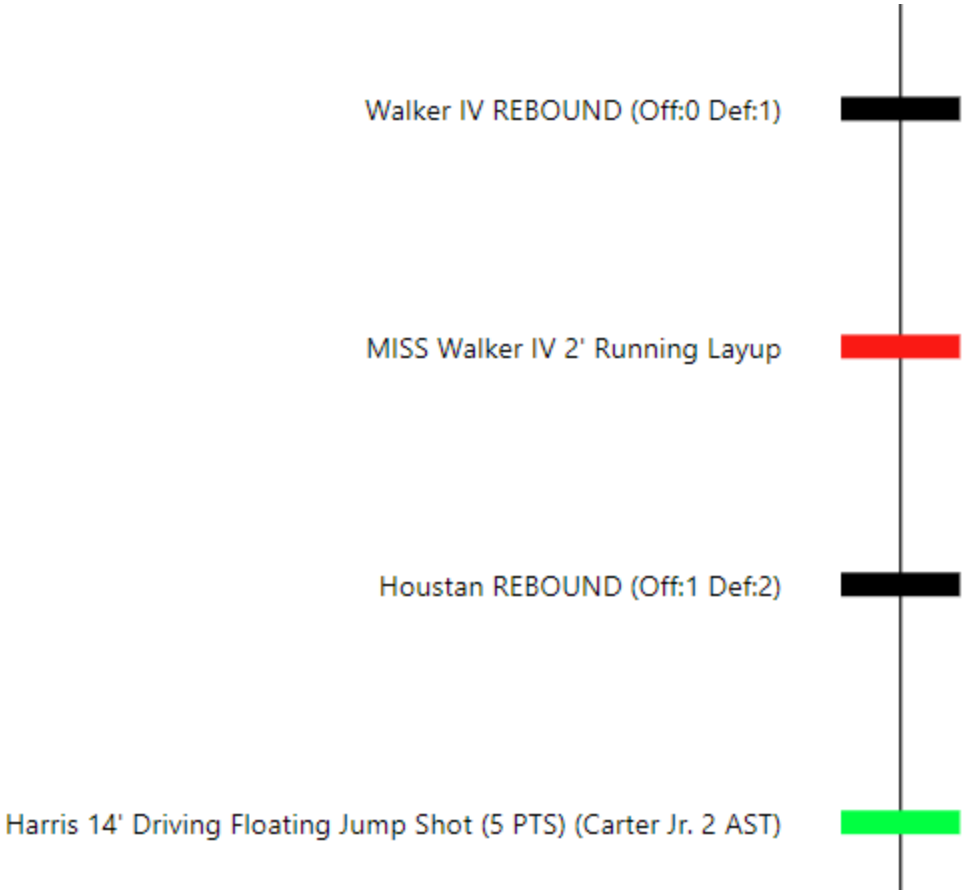
Game Timeline Page

First, the team wanted to create a visualization to show a game over time. Originally, the idea was to create a shot chart to represent shots taken in a game by every player, but that would only show part of the story. Additionally, it would be extremely similar to our player page that we created as that contains a shot chart for each player. As a result, the team decided to look into creating a visualization that acted as a timeline, where the user can briefly see events that took place throughout the game, such as: shots taken, fouls, turnovers, etc. Each event was allocated 100 vertical pixels, so there is an area in the SVG to explain the event that took place.



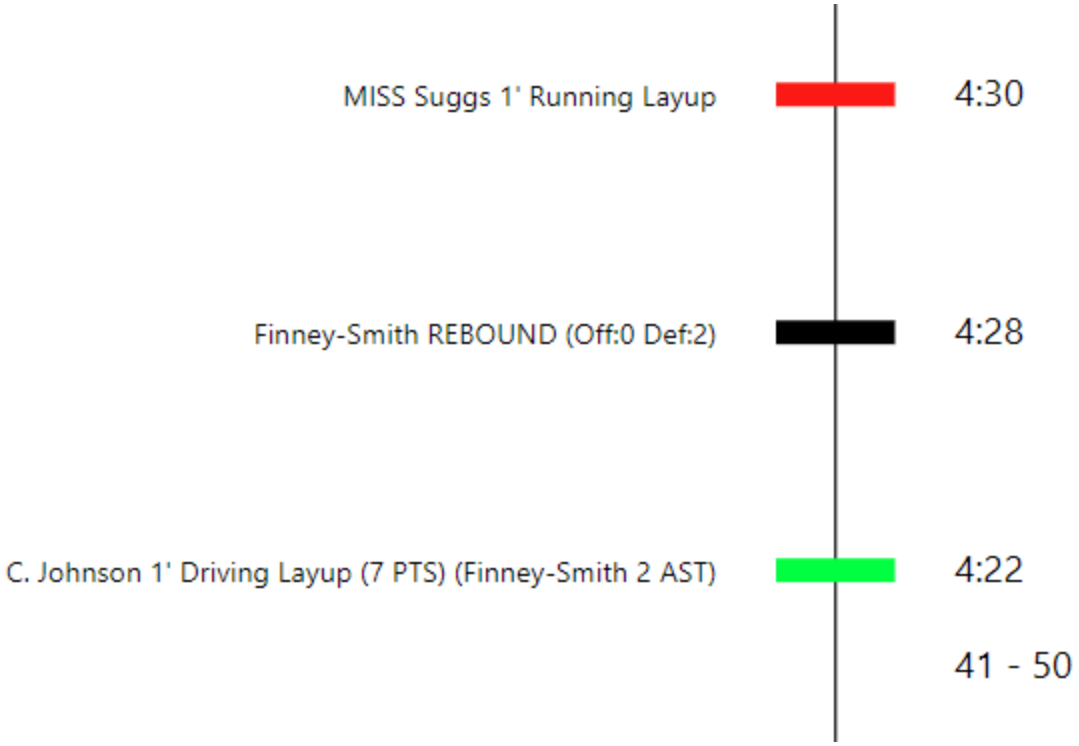
Preliminary timeline visualization or ORL vs. BKN on 2/27

While this was a simple success, the timeline was very bland to look at, and a user would have difficulty understanding which event took place without reading the event description. As a result, the team decided to add color to the rectangular tick marks, showing an event that took place. In particular, shots made had a green color, shots missed had a red color, and timeouts had a blue color. The start of a new period was always indicated with an aqua color as well.



Timeline visualization with colored rectangle tick marks

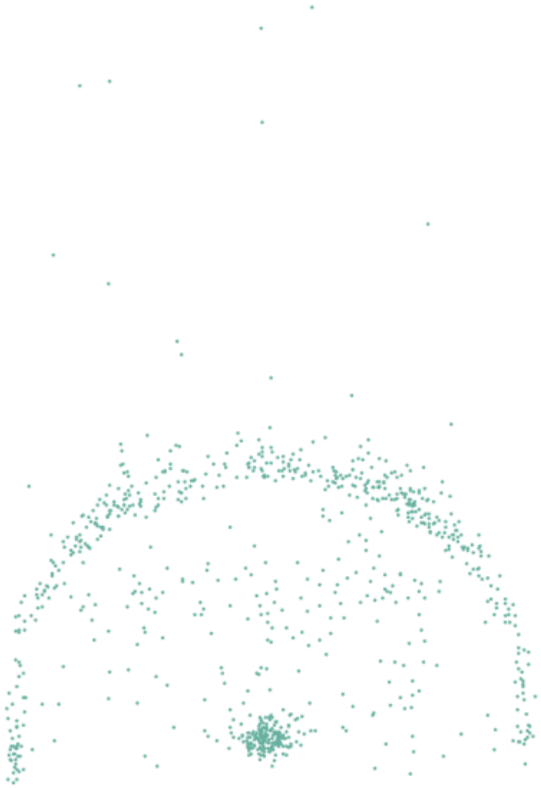
To truly make this a timeline of the game, the team decided to add the in-game clock to the side of each event so the user could easily determine how far into the period the game was. Additionally, the team decided it would be important to also add the score of the game when a team made a basket, so the user can track which team was in the lead at a certain play. Finally, to ensure all users could view this timeline and maintain the scrolling interactivity, the team applied a viewbox attribute to the SVG. This let the SVG scale with the screen resolution of the device, without distorting the text and lines drawn to the SVG. This means that this visualization can be easily viewed on multiple devices with different aspect ratios (ex. Smartphone vs Laptop).



Timeline containing the time of the event and score when updated

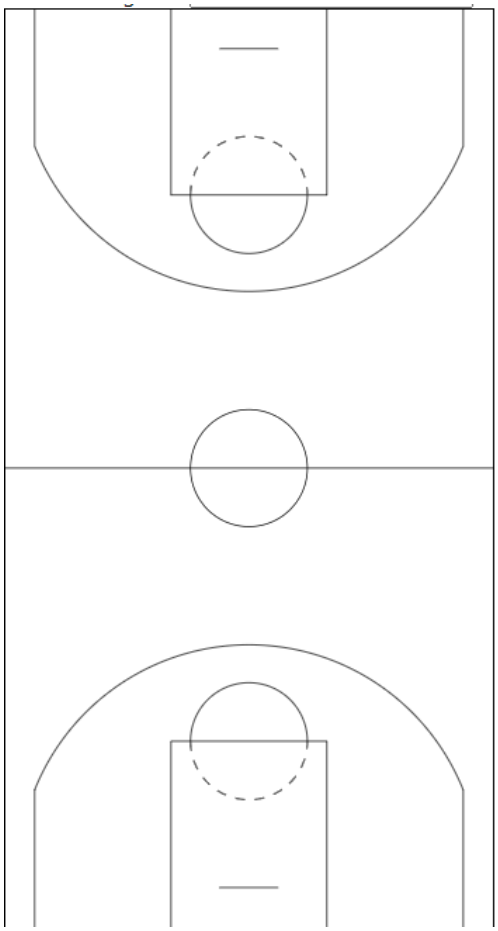
Player Page

The team wanted a way to showcase a player's shots, so we created the player page to hold this information. An easy way to do this was by creating a shot chart. Every shot taken in the NBA since the mid-1990s contains the location of every shot taken by a player, whether the shot resulted in points or was a miss. Using this, the team was able to treat this as a scatter plot, marking the location on the SVG of where the shots were taken.



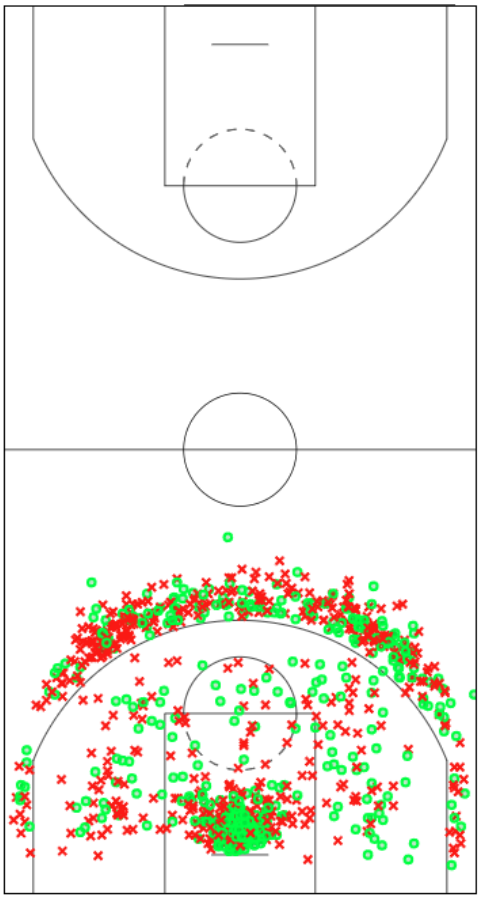
Preliminary scatter plot using LOC_X and LOC_Y data

To better understand the shots (such as if they were from within the paint, 3-point attempts, etc.) the team drew out an NBA court using SVG lines before drawing the data points. This made it easy to tell if a shot attempt was two or three points.



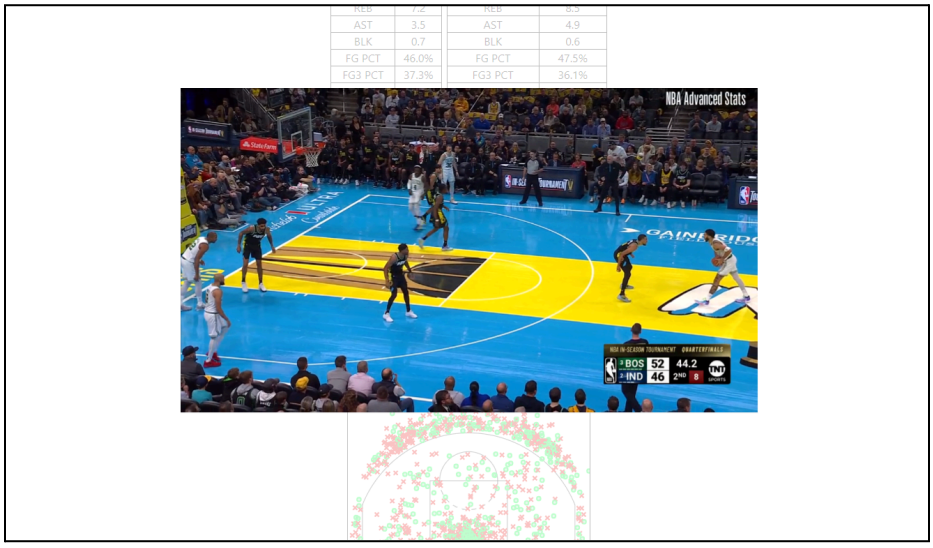
NBA Court SVG created using d3

Then, to determine if the shots were successful or not, we changed the scatter points to green circles when they got a basket in, or a red X if they were unsuccessful.



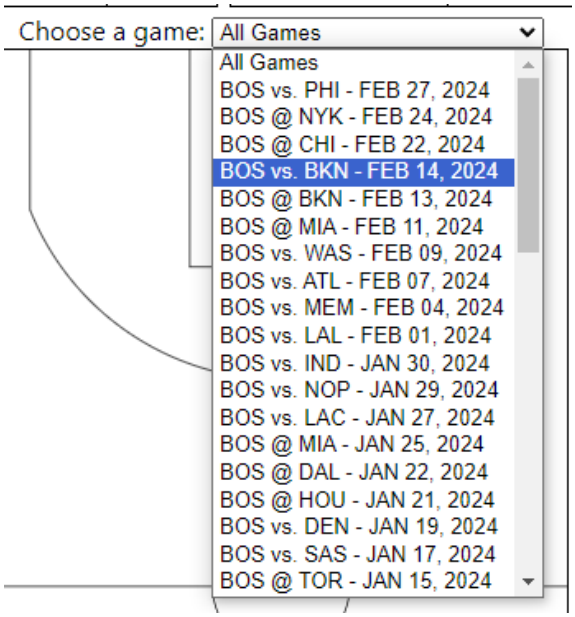
Scatter plot with court background and colored and symbolized scatter points

To make this more interactive, all plays from the start of the 2014-15 season contained video clips of the shot attempted. We then made these scatter points clickable to open a video of the shot attempt.



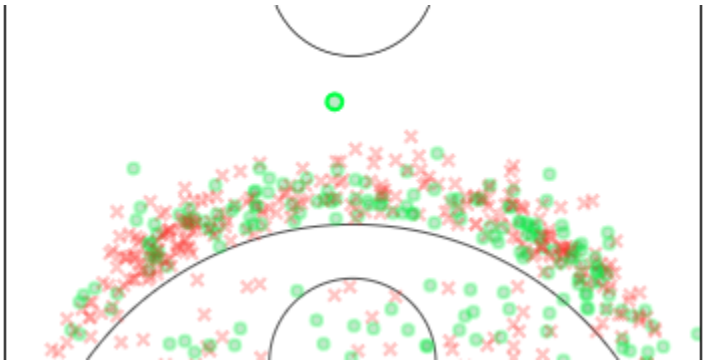
Example popup clip after a applicable scatter point is clicked

Sometimes a player will have a lot of shots taken in a given season. To filter down the data for a better viewing experience, the team decided to add a dropdown that allows the user to select a game in the selected season.



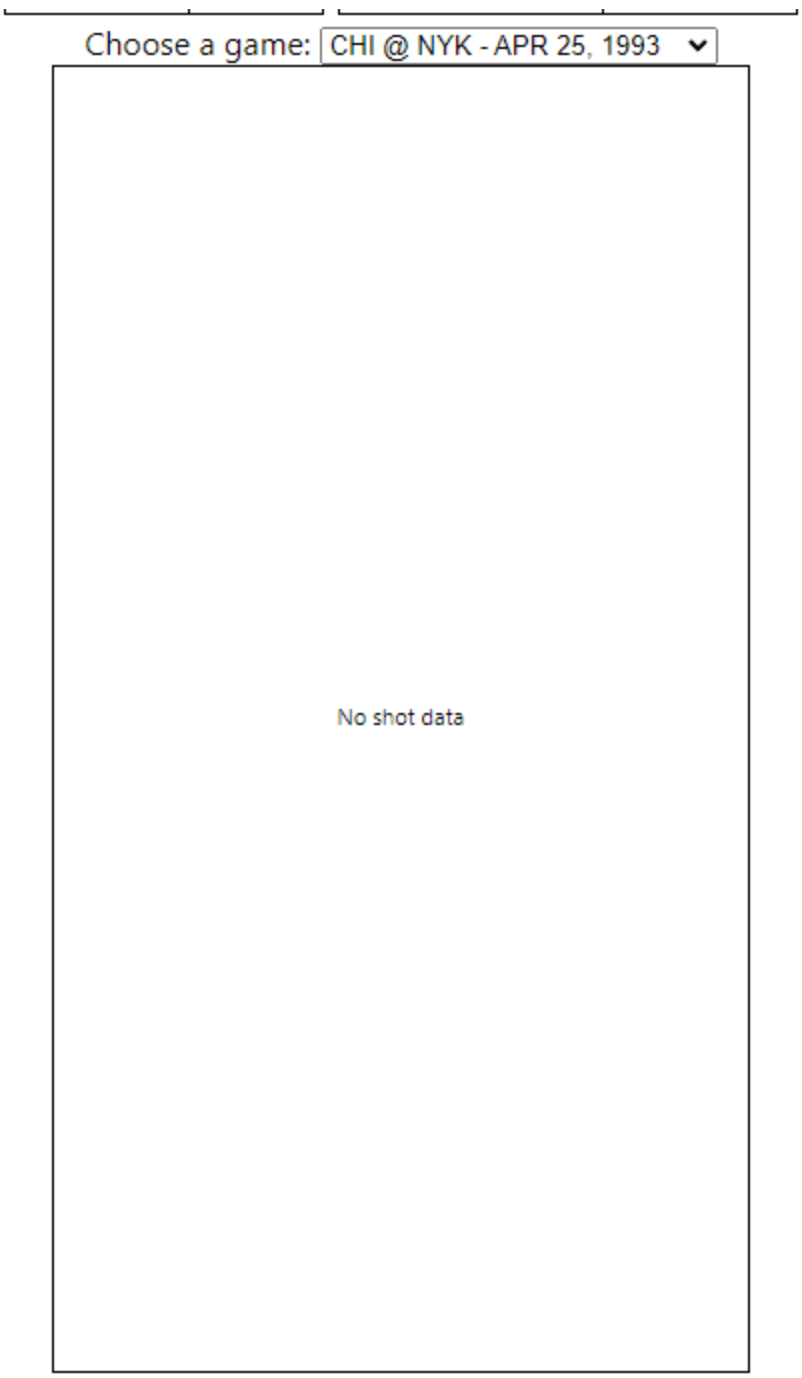
Sample dropdown of Jayson Tatum's games in the 2023-2024 season

Additionally, to show which scatter point the user is hovering over, the other scatter points have their opacity reduced, and the hovered point grows in size.



Shot chart visualization with the top most scatter point hovered, fading out the other scatter points

Finally, after reviewing the page, the team decided it would be best to add a viewbox to the shot chart SVG as well. This prevented the shot chart from being too large to be viewed on devices with small or low-resolution screens. Lastly, for games that happened before the 1990s, they did not include shot chart data. Instead of leaving an empty shot chart, the team decided it would be better to have text appear instead that indicates the shot chart data is not available for the given selection.

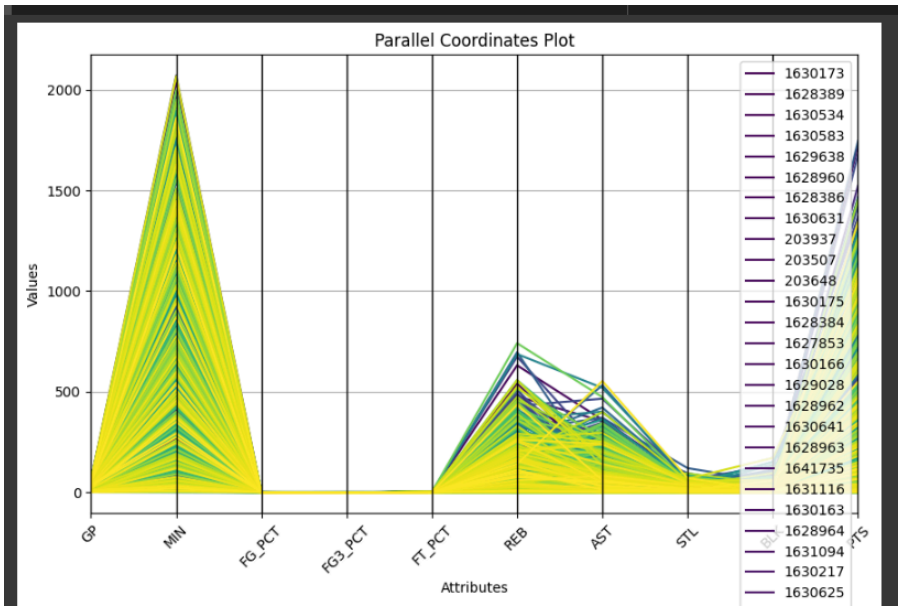


Michael Jordan's Shot chart from a game in 1993, which doesn't contain any shot data

Parallel Coordinate Page

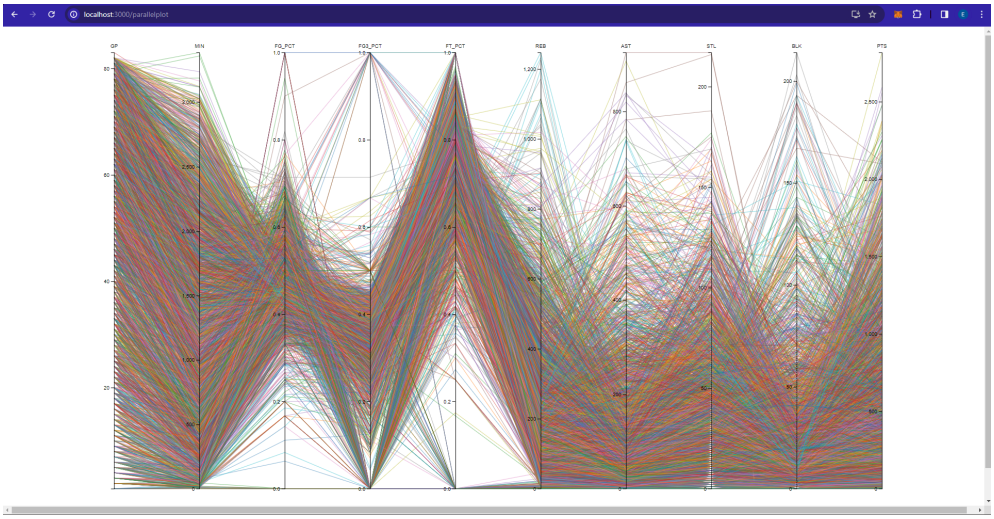
As mentioned earlier in the Related Work section, we wanted to develop a similar parallel coordinates plot for various statistics of an NBA player. By utilizing the playercareerstats NBA API endpoint, a Python script was created to obtain all entries from the API and store them in a CSV file to act as a static database. To my knowledge there is no singular API call that grabs

the information needed, and attempting to grab them real-time would likely result in getting ratelimited. Since the Pandas plotting library has a function for parallel coordinate plots, a quick mock-up was created:



Prototype mockup in Python.

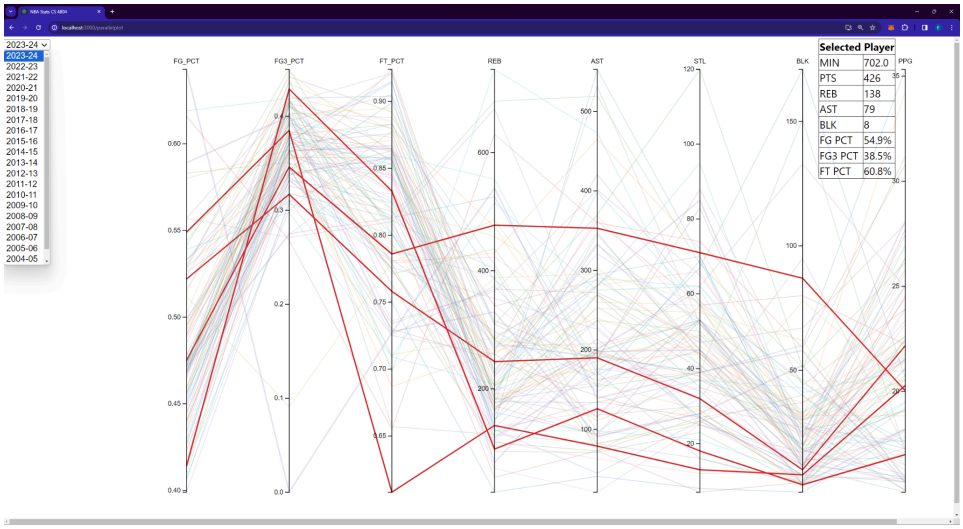
Referencing a bare-bones example from D3's gallery, an initial plot was generated with columns that most basketball fans tend to focus on. But, as seen below, there was zero filter on the lines drawn, meaning all players from all years were included, making it nearly impossible to distinguish a singular player.



First working version of drawing the lines.

After implementing the obvious choice to only include one season's worth of data, we also chose to only include the top 100 players when ranked by seasonal points-per-game (PPG) average.

Another design feature that was implemented was the ability to highlight a line when hovered over with the mouse. The selected player would be highlighted, alongside any other players that were on the same team. Additionally, a dropdown menu was implemented so you could load data for each of the year's acquired from the original API call, and a stat table similar to the one in the Player Page was also added so users could get more information regarding the selected player.



Working highlighting, dropdown menu, and prototype stat table.

Since Rebounds, Assists, Steals, and Blocks were still season totals, the graph and stat table were modified to calculate the averages, which was equivalent to the respective statistic divided by the number of games played.

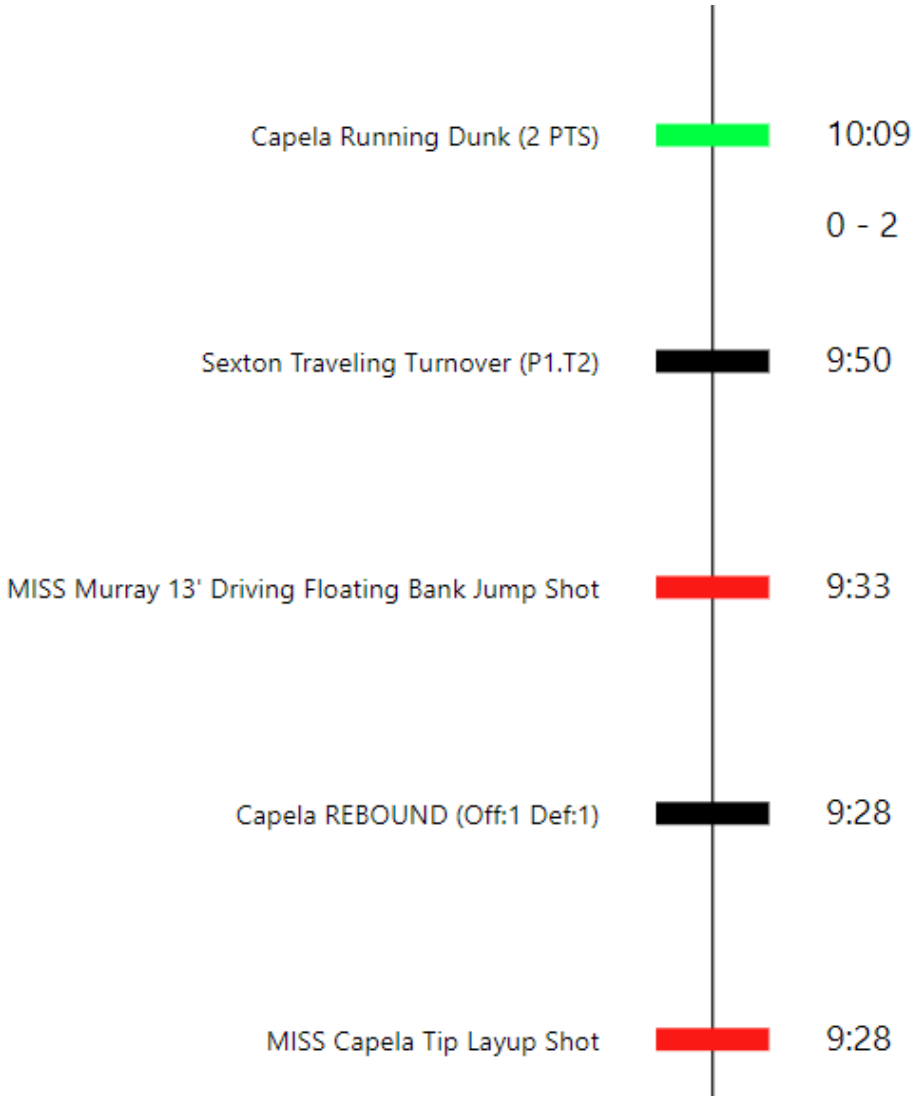
Another major interactive feature implemented is the ability to actually apply a filter onto any axis, filtering the players to only include those that fall within the modified axis's range. Screenshot examples of this in action are demonstrated later in the Implementation section.

Implementation

Game Timeline Page

To implement the game timeline page, we first had to set up a route in React using react-router. This route would require a game ID to follow the route that opened up the timeline page. With this ID, the react page would then request the play-by-play data to our backend, returning a JSON list that contains each play-by-play information for the game. The SVG was then created

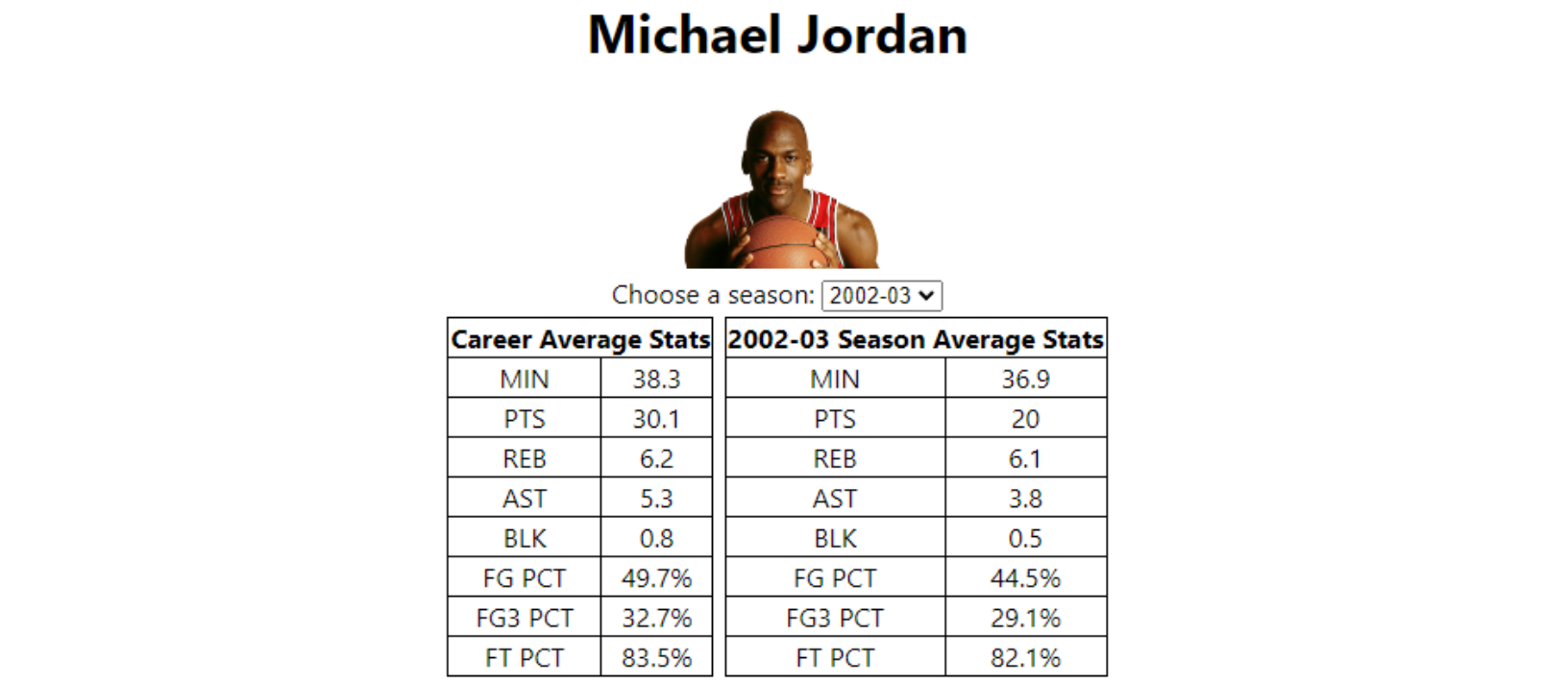
with a height that was scaled based on the number of entries in the list. It would take the length of the list, multiply it by 100 then set the SVG height to that result. This allowed 100 pixels of vertical space to fill in information about the given play/event that took place. Utilizing the event type, we were able to adjust the fill color of the rectangular tick mark, so it was easy to visually see the play that happened, such as timeouts, shots made, and shots missed. Using the SVG text element and assigning it a text-anchor position, it was simple to add information about the play directly next to the tick mark without overlapping the timeline. This lets the visualization have the play information on the left side of the timeline and the time/score on the right side. To make this visualization scalable, simply adding a viewbox with the originally described height and width maintained the data at the expected points for the SVG. Afterwards, we were able to set the SVG width based on the wideness of the display while maintaining the aspect ratio of the SVG as defined by the viewbox.



Example timeline containing play-by-play information

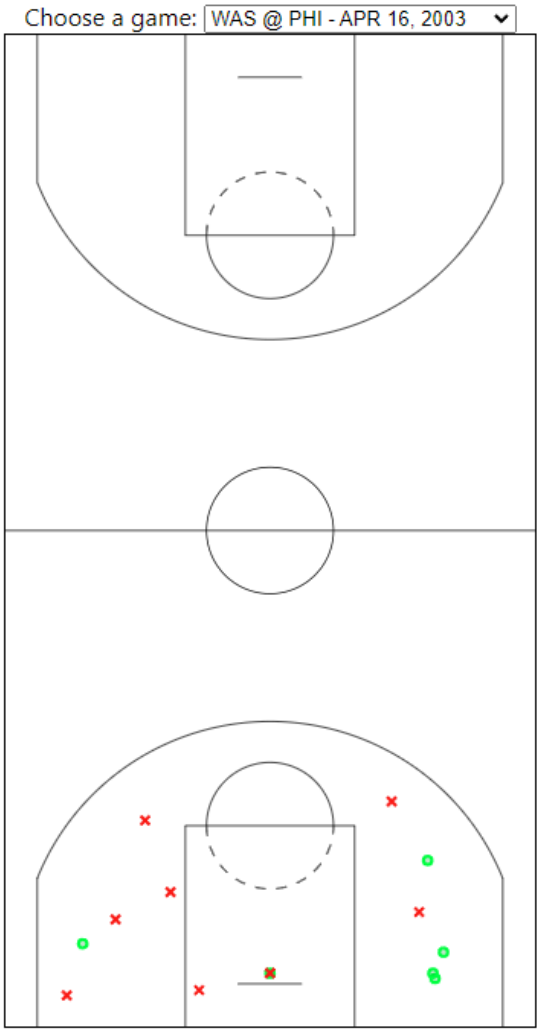
Player Page

Similarly to the Timeline page, we set up a route for the Player page in a similar way, but now requiring the player ID instead of the game ID. With this player ID, the front end sends multiple requests to the NBA API via our backend. This data included all shots attempted for the selected season, season averages, and career averages. The season and career data is then inserted into a table above the Shot Chart. Additionally, a player headshot image is added to the top of the page if available. The shot data is filtered depending on the game selection above the shot chart, and if all games are selected no filtering is done. The NBA API also contains an endpoint to locate video clips of shots attempted starting in the 2014-15 season. Utilizing this, a small video player was implemented as a popup when a scatter point is clicked. Before clicking on a scatter point, the visualization updates to indicate which scatter point the user is currently hovering over, while making the other points extremely transparent. This makes it easier for the user to know exactly which shot attempt they are about to view. When a shot attempt doesn't have a video clip, an alert box appears notifying the user that there is no available video of the shot attempt online. Finally, as the NBA API has data that goes back to the 1950s, there is a lot of missing shot data. To account for this, the shot chart is replaced with a message indicating no data is available for seasons before shot data was collected. This can be seen in Michael Jordan's rookie season on his player page, whereas his final season on the Washington Wizards in 2002-03 contains the shot chart data.



Career Average Stats		2002-03 Season Average Stats	
MIN	38.3	MIN	36.9
PTS	30.1	PTS	20
REB	6.2	REB	6.1
AST	5.3	AST	3.8
BLK	0.8	BLK	0.5
FG PCT	49.7%	FG PCT	44.5%
FG3 PCT	32.7%	FG3 PCT	29.1%
FT PCT	83.5%	FT PCT	82.1%

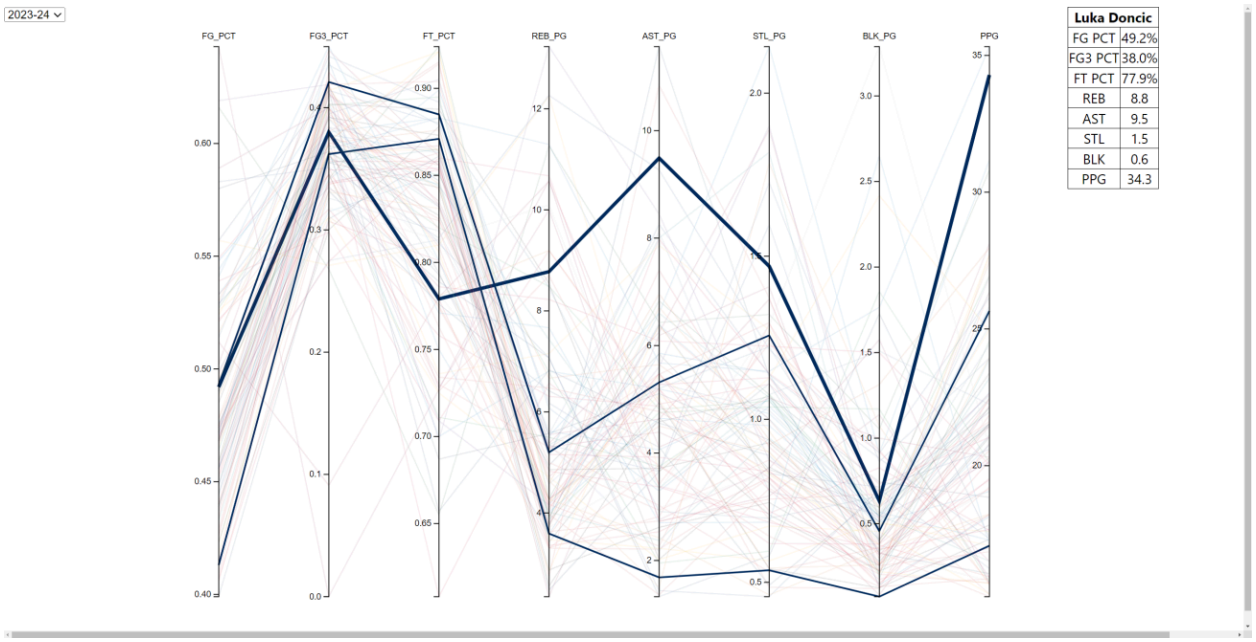
Michael Jordan's career stats and 2002-03 season stats on the player page



Michael Jordan's shot chart for his final regular season game in the NBA on April 16th, 2003

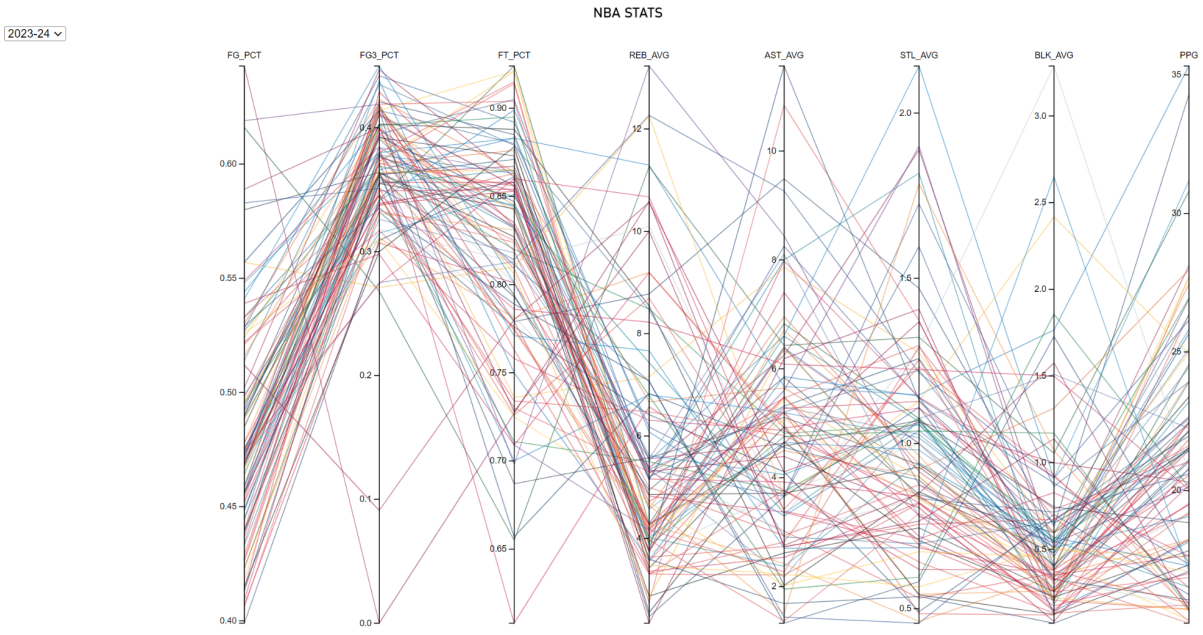
Parallel Coordinate Page

As seen in earlier iterations shown in the Design section, the user loads into a page with the top 100 players by PPG for the current 2023-24 season. Users have the option to switch between seasons with the dropdown menu on the top left. One major interactive feature is the ability to highlight a player's line, which subsequently highlights their teammates and brings up a stat table for that player. The selected player is bolded the most, with teammates still being highlighted, and all other players becoming almost transparent (still slightly visible so a comparison between the selected player and everyone else is easier to see).

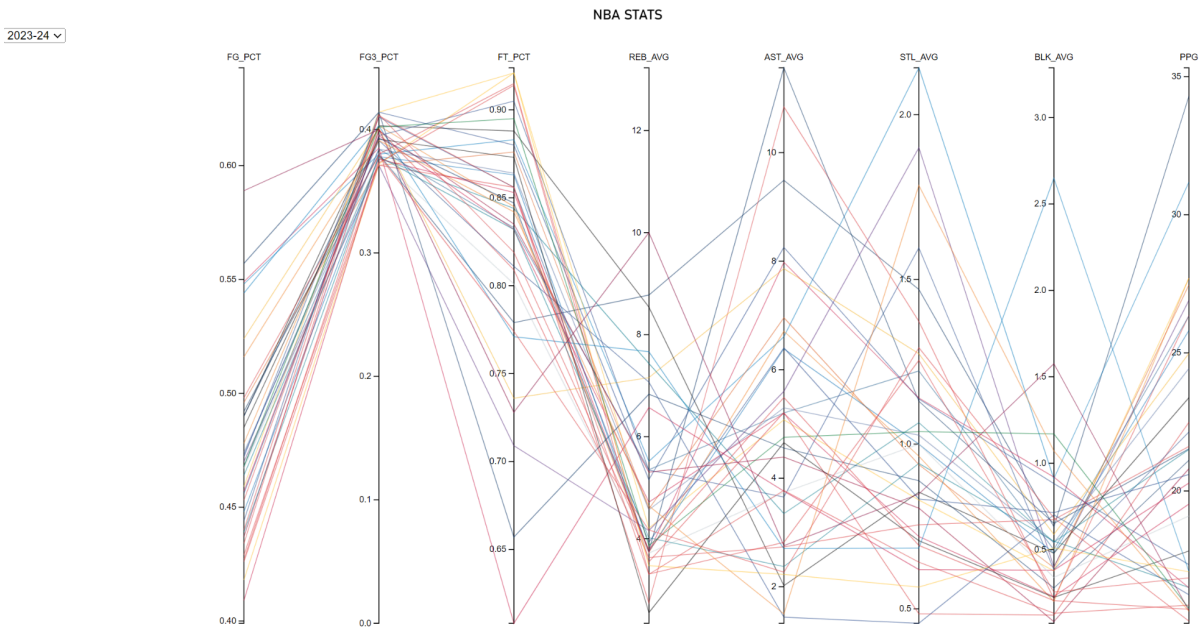


Finalized page.

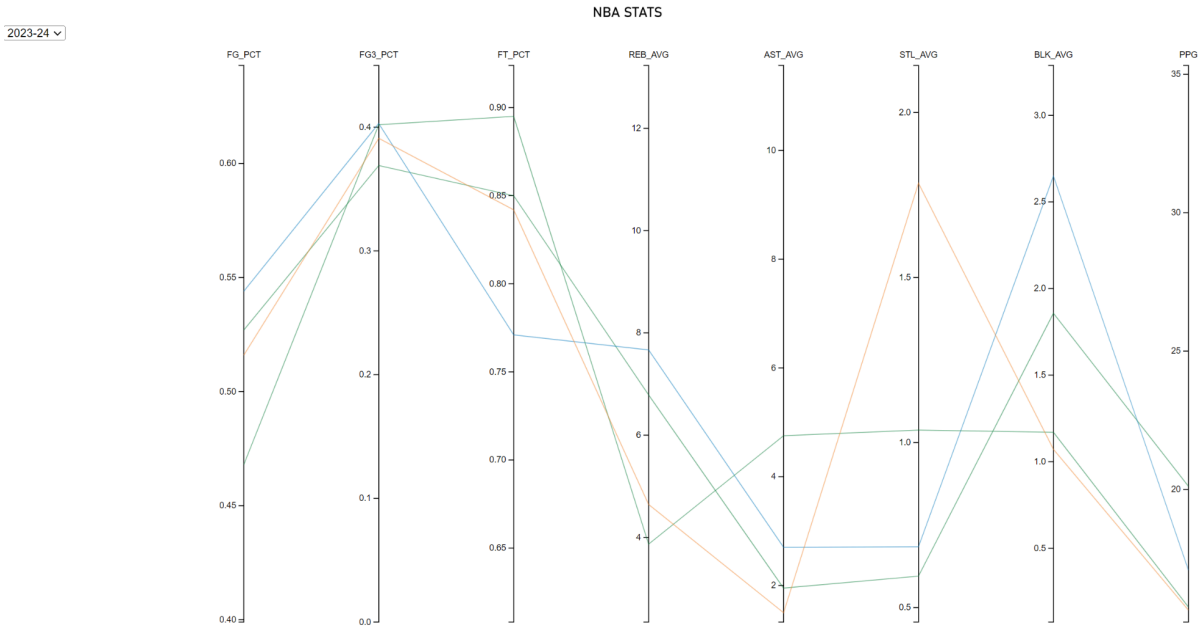
The Design section also mentioned at the end the ability to apply filters on each axis. By clicking two points on any axis, lines are redrawn to only include those that fall within the range. This filtered data persists through multiple axis selections, and the user can reset the axes back to the default range by middle clicking their mouse (bare with the low quality screenshots from a screen share; video provides a better demonstration of this interactive feature).



2023-24 default data.



Filtering 3-point percentage roughy between 37% and 42%



Resulting players after adding filters on Rebounds and Blocks.

Evaluation

Overall, the team created some interesting and fun-to-use visualizations during the project related to sports visualizations. The Shot Chart that we created has a lot more going on than both the Shot Chart we originally found on Bucket List as well as the first prototype of the Shot Chart that we had developed. In this sense, I think the extra features in our Shot Chart succeed in their goal of making a visualization that is more user-friendly and accessible to

non-enthusiasts of basketball. The mouse-over functionality of individual shots, as well as the ability to play a shot by clicking on it in the shot chart, allows even novices to be able to learn how the visualization works just by tinkering with it in a way that the original shot chart did not provide.

The parallel coordinate plot was something that we found to be a visualization that isn't normally seen in sports visualization, and believed to have been implemented well in this project. While this did end up being successfully implemented, we believe that the interactive aspects such as player and team highlighting, along with the axis filtering, is what makes this style feasible at all with this amount of data. Even after filtering the data, having up to 100 points still is a bit messy, and without filtering, the plot would become way too overwhelming and lose a lot of its effectiveness. However, we do feel that this ended up being a good way of quickly comparing players to everyone else during a season. For future improvements, instead of clicking on an axis to get a range, brushing the axis may result in a smoother interaction for range selection.

The game timeline visualization was a quick visualization put together to showcase everything that can happen in a game. While simple, a user can quickly read through it and identify key events that occurred, such as clutch buzzer-beaters at the end of the game. The team believes more can be added to the timeline visualization to make it even more successful such as adding a Shot Chart that scrolls with the user alongside the timeline. As some events have location data attached to them, such as shot attempts, it could be possible to iteratively add to the shot chart as a user scrolls through the timeline. The team also would've liked to investigate filtering down games, as some can become very large, especially those that go into overtime. One possible filtering that can be done is keeping only score-altering events (shots made, free throws), and removing other events such as fouls.