CS3300 Project 1 Write Up

Jake Bass (jab783), Jonathan Wu (jzw8), Ju Hee Lee (jl2755)

March 8, 2016

For Project 1, our group decided to analyze the performance of different NBA superstars from different shooting ranges in crunch time, which we defined as the last 5 minutes of the 4th quarter and/or overtime. The original data set our group obtained was every single field goal attempt of the five players we considered superstars from each 2015-2016 regular season NBA game. We gathered data for Stephen Curry, Lebron James, Anthony Davis, Russell Westbrook and Kevin Durant.

Data

Scraping

We collected our data from *stats.NBA.com*. By inspecting the Network tab, we found a url that returns the data we need in json format. The data contains information on every single shot taken in the NBA. Each data point in the set includes the player who took the shot, the period the shot was taken in, the time left in the period, the distance from the basket he took it from, the type of shot, the amount of points it would count for, the x and y coordinates on the basketball court, and whether he made the field goal or not. The original data set was way too large to work with, so we decided to filter it to shots taken in the last 5 minutes of the 4th quarter and/or overtime. We chose to use this subset of data because we wanted to see how these players reacted to pressure and performed in crunch time. To do the filtering, we adjusted the "Clutch Time" field in the url. The following is an example request used to obtain Stephen Curry's (player ID = 201939) statistics.

http://stats.nba.com/stats/shotchartdetail?Period=0&VsConference=&LeagueID=00&LastNGames=0&TeamID=0&Position=&Location=&Outcome=&ContextMeasure=FGA&DateFrom=&StartPeriod=&DateTo=&OpponentTeamID=0&ContextFilter=&RangeType=&Season=2015-16&AheadBehind=&PlayerID=**201939**&EndRange=&VsDivision=&PointDiff=&RookieYear=&GameSegment=&Month=0&**ClutchTime=Last%205%20Minutes**&StartRange=&EndPeriod=&SeasonType=Regular+Season&SeasonSegment=&GameID=

Filtering

With these urls, we obtained the json and saved it for each of the 5 players in the Project1/json directory. We made d3.json function calls to each of these json files, and specifically focused on `response.resultSets[0].rowSet` which returns a 2D array, with each 1D array containing 20 columns with specific fields describing each shot. From here, we filtered our data even further by focusing on columns 12, 15, 17, 18, and 20, which are SHOT_TYPE, SHOT_ZONE_RANGE, LOC_X, LOC_Y, SHOT_MADE_FLAG, respectively. The field SHOT_TYPE was used to determine whether it's a 2pt shot or a 3pt shot. The field SHOT_ZONE_RANGE was used to categorize the shots into 4 different groups based on the distance the shot was taken at. The fields LOC_X and LOC_Y, and SHOT_MADE_FLAG were used to plot the shots onto our court diagram. We plotted the made shots with red, and the missed shots with gray to be able to easily distinguish the player's strengths and weaknesses by looking at the graphs.

Visualization Implementation

We found a website: ***http://savvastjortjoglou.com/nba-shot-sharts.html*** that described the coordinates (LOC_X, LOC_Y) system used in the data from *stats.NBA.com.* However, the scale they used was not the same as the SVG coordinates. Therefore, we had to use d3.scale to scale from their x-domain [-300, 300] and their y-domain [-100, 500] to [0, 600] and [600, 0] SVG coordinates respectively.

The court was the most complicated part to graph using SVG elements. We used two SVG rectangles, a large rectangle that outlined the border of the x and y domains, and a smaller rectangle that starts at xScale(-250), yScale(422.5) and has a width of (xScale(500) - xScale(0)) and a height of (yScale(0)-yScale(470)). This smaller rectangle represents the half court of an NBA regulation basketball court. We used two more SVG rectangles to graph the painted area, the backboard, and the straight part of the 3pt lines. We also used SVG circles and arcs to draw the curved lines on the court.

We made a data structure called playerInfo so that we could easily iterate through all the players when graphing the statistics and doing statistical analysis. We also implemented several helper functions to add different SVG elements and calculate values to plot.

- **addPoints**
  addPoints takes in the 2D array of shot stats and appends points to the basketball court specified by player_group. The x and y coordinates of the raw data are converted using xScale and yScale.
- **addImage**
  addImage takes in player index i and appends player image to the player's svg. We defined a pattern to which we appended an image, and filled a circle with the image to create a circular frame around the player image, as shown in **Figure 1.**

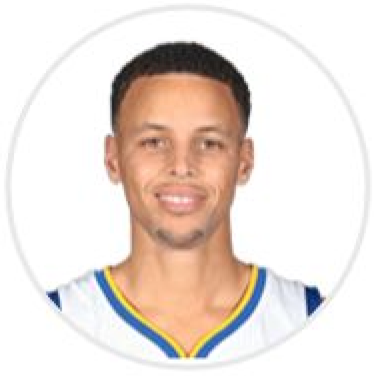

Figure 1. Image with Rounded Border

- **plotPlayerData**
  plotPlayerData takes in player index i and the id of the svg, and calls helper function addPoints to plot (x,y) points to the SVG.
- **plotShotTypeBars**
  plotShotTypeBars takes in player index i and a dictionary to fill in the 4 distance bar graphs with the computed expected points per shot averages. The dictionary contains the information of player i (totalPoints, totalShots, expectedPointsPerShot) for each of the 4 distance ranges. This function fills the SVG rect element using a scale with domain [0,2] and range [0,400]. This scale scales the average points per shot to fit the width of the bar graph in SVG coordinates.
- **plotPlayerBar**
  plotPlayerBar takes in player index and cumulative expected points per shot for that player. It plots the expected points per shot for the specified player in the summary section. Similar to plotShotTypeBars, this function uses d3.scale to convert the expected points per shot to fit the width of the bar in SVG coordinates.
- **computeShotStats**
  Given playerIndex, computeShotStats calculates the expected average points per shot for each of the 4 distance ranges. It calls the helper function plotShotTypeBars to plot the individual bar graphs and the helper function plotPlayerBar to plot the summary bar graph. For each of the shot ranges, we find the sum of the total number of points from made shots, and divide it by the sum of the total number of shots. To accomplish task, we had to check the SHOT_TYPE to see whether the shot was worth 2 points or 3 points.
- **addInfo**
  addInfo fills the left SVG for each player i with: name, stats, empty bar graphs. We used SVG rect with rounded corners to plot the bar graphs.
- **addSummaryInfo**
  addSummaryInfo fills the first SVG graph with title, subtitle, player names labels, and empty bar graphs using SVG rect with rounded corners.

To differentiate between the players, and to make our project more aesthetically pleasing, we chose the color of the bar graphs as the representative color of each player. We also used SVG text element with <tspan> element to give different designs to each of the words in a text element as shown in **Figure 2.**

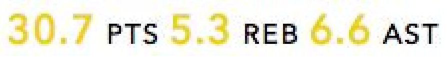
30.7 PTS 5.3 REB 6.6 AST

Figure 2. Example of Text element with <tspan>

Story of the Visualization

Our goal for the visualization was to see how effective these NBA superstars are at scoring and what their shot tendencies look like in crunch time. To examine this, we filtered our data to take into account only the last 5 minutes of the fourth quarter and overtime. While clutch data is often calculated for when neither team is ahead by more than five points, we chose not to filter our data this way for two reasons. Firstly, we wanted to have enough data to show the trends we were seeing while still keeping our analysis fresh. This meant that we wanted to show only this season, and so we didn't want to filter out more of the data without just cause. Secondly, we picked these five players for a reason: they're five of the very best in the NBA. While we did this to compare the league's elite, it also allowed us to make the assumption that if any of these players were in the game with that much time left, it was probably a close game. A cursory analysis confirmed our suspicions, and so we did not add the additional filter.

We saw that all five players had generally similar shot tendencies. While delving deeper analytically, we saw Stephen Curry had more of a tendency to shoot 3 pointers and inside shots, while Russell Westbrook and Kevin Durant's shot selections were more balanced. Anthony Davis and Lebron had tendencies to shoot mid-range and close-range shots.

We then calculated their expected points per shot from each shot zone (Less than 8 ft., 8-16ft., 16-24ft. and 24+ft.) and saw how each player stacked up against each other. We thought this would be a good way to visualize how shot distance affects their efficiency. We thought that we would see players shoot more from the zones where they had greater expected points per shot. A surprising takeaway was how inefficient Lebron James, Kevin Durant and Russell Westbrook were in crunch time. These three players are considered as some of the league's brightest stars, but for some reason seem to struggle in these situations. Something else to note was that these players' shooting performances in the clutch wasn't necessarily indicative of how these players' teams were performing in the regular season, as Westbrook and Durant's Thunder are 43-20, while Davis's Pelicans are at 24-38. Curry's Warriors, however, are sitting pretty with 55-6 and Lebron's Cavaliers are at 44-18. We posited that this could be the case that defenses might pay extra attention to known scoring threats towards the end of a game, and as a result these stars might be forced to play a more facilitating role.

In the end, it showed Stephen Curry as the most efficient player in the clutch with 1.36 estimated points per shot, Anthony Davis in second with 1.31 estimated points per shot, Lebron in third with 1.08 estimated points per shot, Durant in fourth with 1.02 estimated points per shot and Westbrook last with 0.91 estimated points per shot.