Stat 345 Midterm Project Ben Ellingworth

Ben Ellingworth

3/30/2022

Timberwolves Shot Chart (Parts 1 and 3)

```
# Here are all the necessary packages needed to be loaded in to perform the task of creating a
# shot chart. Making sure all of these packages are loaded in order is essential. When
# downloading NBAstatR make sure to follow the exact code below to correctly get all the data
# loaded in. This part was tough but it ended up working out following what I did below.
#library(devtools)
#library(tidyverse)
#library(ggplot2)
#library(dplyr)
#devtools::install_github("abresler/nbastatR")
#library(nbastatR)
#library(hexbin)
#library(lubridate)
#library(gganimate)
#library(magick)
#library(formatR)

# The first step after everything is correctly loaded in is to change your
# "VROOM_CONNECTION_SIZE." When I first tried using the data from nbastatR I kept getting an
# error message and multiplying by 2 to my vroom size helped fix this issue. It may need to
# bigger depending on what you plan on working with but for this specific task this was the
# correct size that worked.
Sys.setenv("VROOM_CONNECTION_SIZE" = 131072 * 2)

#Loading in data and preparing it.
# The second step is to divide out the data that you want to create your shot charts from. For
# this specific example,  I specified the data to use (team_shots), the team I want
# (Timberwolves), as well as the seasons I wanted to use. You can tweak these variables as
# needed to get different kind of shot charts that you might want in the future. I would just
# make sure to check out the team_shots data to see what you can specify to work with, before
# plugging in different variables. Once I specified what season to use, I used the select
# function to clean the data up and select which columns I wanted to keep. This is important if
# any issues arise. Cleaning the data up allows for us to go back and try and fix the problem.
# All of the columns I needed for this shot chart were namePlayer, yearSeason,
# zoneRange,locationX, locationY, slugZone, typeShot, isShotAttempted, isShotMade,
# distanceShot.The rest were not needed so I made the judgment to not include them in our data.
# This code gives us a base to rely on for future modifications to achieve certain goals.
Tpups2022 <-teams_shots(teams = "Minnesota Timberwolves", seasons = 2022)%>%
   select(namePlayer, yearSeason, zoneRange,locationX, locationY, slugZone, typeShot,
   isShotAttempted, isShotMade, distanceShot)
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```
## Minnesota Timberwolves 2021-22 shot data
```

```
Tpups2012 <- teams_shots(teams = "Minnesota Timberwolves", seasons = 2012)%>%
   select(namePlayer, yearSeason, zoneRange,locationX, locationY, slugZone, typeShot,
   isShotAttempted, isShotMade, distanceShot)
```

```
## Minnesota Timberwolves 2011-12 shot data
```

```
Tpups2002 <- teams_shots(teams = "Minnesota Timberwolves", seasons = 2002)%>%
   select(namePlayer, yearSeason, zoneRange,locationX, locationY, slugZone, typeShot,
   isShotAttempted, isShotMade, distanceShot)
```

```
## Minnesota Timberwolves 2001-02 shot data
```

```
# Joining data in one place.
# Here we use the full join function to join the 3 separate years together. This will become
# very important once we need to facet our data and show all 3 years of shot chart data later. I
# used two separate calls to do this task. This allows us to only have to call one object for
# the shot chart and compare years.
Tpups <- full_join(Tpups2002,Tpups2012)
```

```
## Joining, by = c("namePlayer", "yearSeason", "zoneRange", "locationX",
## "locationY", "slugZone", "typeShot", "isShotAttempted", "isShotMade",
## "distanceShot")
```

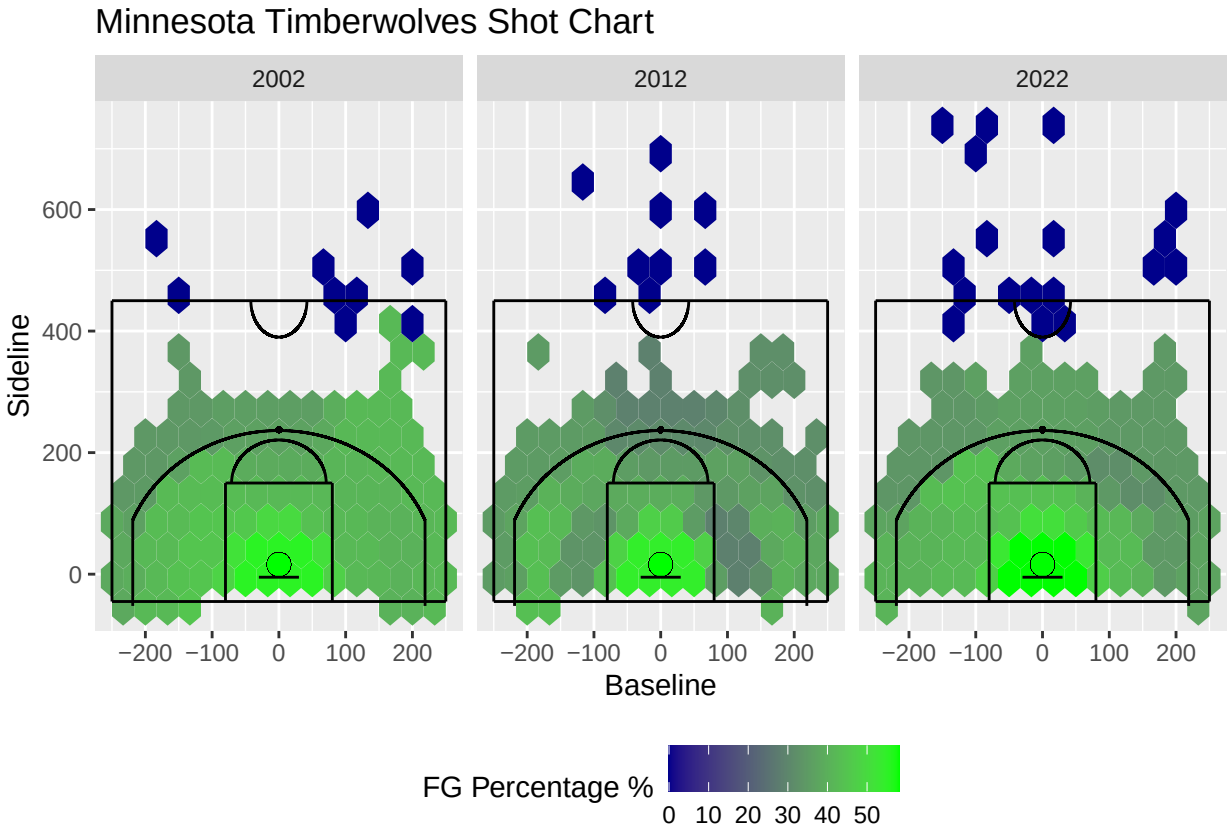
```
Tpups2 <- full_join(Tpups, Tpups2022)
```

```
## Joining, by = c("namePlayer", "yearSeason", "zoneRange", "locationX",
## "locationY", "slugZone", "typeShot", "isShotAttempted", "isShotMade",
## "distanceShot")
```

```
#Mutating Data
# Here is where we start to mutate the data and add an extra column based on our purpose for the
# shot chart. For this specific shot chart, I wanted to see the fg percentage from each zone on
# the floor to get a more accurate report of our 2022 season. So first, I took the data gathered
#from the step before and grouped it by yearSeason and zoneRange first to group shots in similar
# distances together from that specific year. Next, I grouped the new data by the zone of these
# shots using slugZone. This took shots of similar distance and groups them as either from the
# left wing, center, right wing, or either corner. Third, I separated the type of shot (2pt or
# 3pt) to help specify the close shots between long 2pt and short 3pt shots. Lastly, once
# everything is grouped with these 3 specific variables, I used the mutate function to create a
# column that gives the fg% from these specifically grouped zones. I did this by taking the sum
# of the isShotMade variable and dividing it by the isShotAttempted. This is basically a "mean"
# function that will show up later. Finally, once this is finished we can get to the final part
```

```
# of plotting the data. This gives us the main idea for our shot chart.
madeShot2 <- Tpups2 %>%
   group_by(yearSeason,zoneRange,slugZone,typeShot)%>%
mutate(fgPercentage = sum(isShotMade)/sum(isShotAttempted)*100)

#ggplot
# This is where we use the data, cleaned and manipulated above, to create our shot chart. First
# we must identify a name for it, then call ggplot() with our data we want to use inside. This
# gives us our blank canvas to add to it whatever we like. Notice I used the the data set we
# created with our new fg% in it (madeShot2). This is essential to get this project to work.
shotChart <- ggplot(madeShot2) +

   # Next I used stat_summary_hex to help bin the points into hexagons rather than geom_point.
   # Geom_point has too many points and is difficult to draw conclusions from, so binning the
   # points is essential. Here is where we define our x,y,z variable we want to use. These
   # variables are the inputs into the function. Our x and y variables are the respective
   #locations found in our data above, that plot the specific shots where they actually were on
   # the floor. The z however, is what we want our color to be based on. For this example,
   # wanted to see the difference in FG% at different parts of the court so I set the z to be =
   # to fgPercentage. The last part was specify what function we wanted to use for these points.
   # As mentioned before, the FG% is a mean function so I specified it in quotes and decided how
   # many bins we wanted to use.
   stat_summary_hex(aes(x = locationX, y = locationY, z = fgPercentage),fun = "mean",
   bins = 15)+

# Here is where we can decide the colors. Make sure to pick 2 colors that are opposite so the
# viewers can see the differences between areas. I chose green and blue because they are
# different and the Timberwolves colors.
scale_fill_gradient(high = "green", low = "darkblue", "FG Percentage %")+

# Here I used xlab and ylab to label the axis. I chose baseline and sideline because that will
# show the viewers where on the court is is. I also named the title the year to make the chart
# look better and give an easy reference for the viewers to look back at. I did this by a
# simple ggtitle() call.
   xlab("Baseline") +
   ylab("Sideline") +
   ggtitle("Minnesota Timberwolves Shot Chart")+

# Now is when I used the facet_wrap function to facet the different shot charts from the 3
# selected years above. I did this by choosing the yearSeason variable to seperate the data.
# This accomplished the goal. Lastly, I moved the legend position to the bottom of the charts
# to make it easier to read and allow the charts to be bigger on the screen.
   facet_wrap(~yearSeason)+
   theme(legend.position = "bottom")+

# Last was plotting the court to fit the actual dimensions of a NBA court. My method for
# attacking this problem was to put points at the main parts of the court and move my respective
# lines to fit them. For example, I left the 3pt dot as a reference. NbastatR uses a 1/10 foot
# ratio and an NBA 3pt shot is 23.75 feet away from the top of the key. So, I placed
# a point at (0, 237.5) and manipulated my numbers to move the 3pt arc to match with that
# point. The rest were lines that were straight and just needed to have either x or y set at
# the right spot. Looking up the dimensions of an NBA court and translating it to this graph
# was the toughest part.
```

```
   #3PT Line. I found the outline of the 3pt line on GitHub by Ed Kupfer and scaled it to fit
   # the graph of our shot chart. I did this by adding 41.75 and * by 9.95 at the end.
  geom_path(data=data.frame(x=c(-22,-22,-22000:(-1)/1000,1:22000/1000,22,22), y=-c(47,47-169/12,41.75-sq
47-169/12,47)+41.75)*9.95,
aes(x=x, y=y))+
   #Half court line
   geom_path(data=data.frame(x=c(-250,250),y=c(450,450)), aes(x=x, y=y))+
   #Side lines
   geom_path(data=data.frame(x=c(-250,-250), y= c(-45,450)), aes(x=x, y=y))+
    geom_path(data=data.frame(x=c(250,250), y= c(-45,450)), aes(x=x, y=y))+
   # Baseline
   geom_path(data=data.frame(x=c(-250,250), y = c(-45,-45)), aes(x=x, y=y))+
   # Free Throw line and lane lines
      geom_path(data=data.frame(x=c(-80,80), y = c(150,150)), aes(x=x, y=y))+
    geom_path(data=data.frame(x=c(-80,-80), y = c(-45,150)), aes(x=x, y=y))+
   geom_path(data=data.frame(x=c(80,80), y = c(-45,150)), aes(x=x, y=y))+
   # Semi Circle at the half court line. This one needed a little scale of the x axis by
   # multiply by 7 and moving up along the y axis by adding 450.
   geom_path(data=data.frame(x=c(-6000:(-1)/1000,1:6000/1000)*7,
   y=-c(sqrt(6^2-c(-6000:(-1)/1000,1:6000/1000)^2)*10)+450),aes(x=x,y=y))+
   #Semi Circle above the free throw line. Multiplying by 11.811 from both the y and x axis gave
    # use the correct dimensions of the semi circle.
   geom_path(data=data.frame(x=c(-6000:(-1)/1000,1:6000/1000)*11.811,                                y=c(sq
   geom_point(aes(0,237.5), size = 0.5)+

   # Placing two points and making them these sizes help give us the rim. I was not able to load
   # in an circle so I decided to place a smaller green dot above the black dot. This is fine
   # because all of the hexagons under these dots were green so there is not data lost. I placed
   # the backboard 40 from the baseline which was -45.
   geom_point(aes(0,16), size = 3.625, color = "black")+
      geom_point(aes(0,16), size = 3.3, color = "green")+
   geom_path(data=data.frame(x=c(-30,30), y = c(-5,-5)), aes(x=x, y=y))
shotChart
```


Minnesota Timberwolves Shot Chart
2002
2012
2022
Sideline
600
400
200
0
−200 −100 0 100 200
Baseline
FG Percentage %
0 10 20 30 40 50