Data Science Project

Nischal Kalahasti

6/2/2024

Introduction

The purpose of this project is to gauge your technical skills and problem solving ability by working through something similar to a real NBA data science project. You will work your way through this R Markdown document, answering questions as you go along. Please begin by adding your name to the “author” key in the YAML header. When you’re finished with the document, come back and type your answers into the answer key at the top. Please leave all your work below and have your answers where indicated below as well. Please note that we will be reviewing your code so make it clear, concise and avoid long printouts. Feel free to add in as many new code chunks as you’d like.

Remember that we will be grading the quality of your code and visuals alongside the correctness of your answers. Please try to use the tidyverse as much as possible (instead of base R and explicit loops). Please do not bring in any outside data.

Note:

Throughout this document, any `season` column represents the year each season started. For example, the 2015-16 season will be in the dataset as 2015. For most of the rest of the project, we will refer to a season by just this number (e.g. 2015) instead of the full text (e.g. 2015-16).

Answers

Part 1

Question 1:

- Offensive: 56.29% eFG
- Defensive: 47.86% eFG

Question 2: 81.42%

Question 3: 43.3%

Question 4: This is a written question. Please leave your response in the document under Question 5.

Question 5: 84.5% of games

Question 6:

- Round 1: 84.72%
- Round 2: 63.9%
- Conference Finals: 55.56%
- Finals: 77.78%

Question 7:

- Percent of +5.0 net rating teams making the 2nd round next year: 63.63%
- Percent of top 5 minutes played players who played in those 2nd round series: 79.04%

Part 2

Please show your work in the document, you don’t need anything here.

Part 3

Please write your response in the document, you don’t need anything here.

Setup and Data

```
library(tidyverse)
# Note, you will likely have to change these paths. If your data is in the same folder as this project,
# the paths will likely be fixed for you by deleting ../../Data/awards_project/ from each string.
# player_data <- read_csv("../../Data/playoffs_project/player_data.csv")
# team_data <- read_csv("../../Data/playoffs_project/team_data.csv")
```

Part 1 – Data Cleaning

In this section, you’re going to work to answer questions using data from both team and player stats. All provided stats are on the game level.

Question 1

QUESTION: What was the Warriors’ Team offensive and defensive eFG% in the 2015-16 regular season? Remember that this is in the data as the 2015 season.

```
# Here and for all future questions, feel free to add as many code chunks as you like. Do NOT put echo = F though
, we'll want to see your code.

#read in both team and player data csv files
team_data <- read_csv("team_game_data.csv")
```

```
## Rows: 27144 Columns: 41── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr   (4): off_team_name, off_team, def_team_name, def_team
## dbl  (36): season, gametype, nbagameid, offensivenbateamid, off_home, off_wi...
## date  (1): gamedate
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
player_data <- read_csv("player_game_data.csv")
```

```
## Rows: 434797 Columns: 59── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr   (5): player_name, team, team_name, opp_team, opp_team_name
## dbl  (53): nbagameid, season, gametype, nbapersonid, nbateamid, opposingnbat...
## date  (1): gamedate
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#group everything together for the Warriors team in 2015 regular season including stats to utlimately use for our eFG percentage calculation
combined_team_data_off <- team_data %>%
  filter(gametype == 2, off_team == "GSW", season == 2015) %>%
  group_by(off_team) %>%
  summarise(
    total_fgmade = sum(fgmade),
    total_fg3made = sum(fg3made),
    total_fgattempted = sum(fgattempted)
  )

#calculate the offensive eFG percentage based on formula and use the summed values from every game over the season
offensive_eFG_percent = (combined_team_data_off$total_fgmade + 0.5*combined_team_data_off$total_fg3made) * 100 / combined_team_data_off$total_fgattempted

#group everything together for the Warriors team in 2015 regular season including stats to utlimately use for our defensive eFG percentage calculation
combined_team_data_def <- team_data %>%
  filter(gametype == 2, def_team == "GSW", season == 2015) %>%
  group_by(def_team) %>%
  summarise(
    total_fgmade_allowed = sum(fgmade),
    total_fg3made_allowed = sum(fg3made),
    total_fgattempted_allowed = sum(fgattempted)
  )

#calculate the defensive eFG percentage based on formula and use the summed values from every game over the season
defensive_eFG_percent = (combined_team_data_def$total_fgmade_allowed + 0.5*combined_team_data_def$total_fg3made_allowed) * 100 / combined_team_data_def$total_fgattempted_allowed

offensive_eFG_percent
```

```
## [1] 56.29718
```

```
defensive_eFG_percent
```

```
## [1] 47.86465
```

ANSWER 1: Offensive: 56.29% eFG , Defensive: 47.86% eFG

Offensive: 56.29% eFG
Defensive: 47.86% eFG

Question 2

QUESTION: What percent of the time does the team with the higher eFG% in a given game win that game? Use games from the 2014-2023 regular seasons. If the two teams have an exactly equal eFG%, remove that game from the calculation.

```
filtered_df <- team_data %>%
  filter(gametype == 2, season >= 2014, season <= 2023)
filtered_df
```

```
## # A tibble: 23,958 × 41
##    season gametype nbagameid gamedate   offensivenbateamid off_team_name
##     <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>
##  1   2016        2  21600495 2016-12-30         1610612740 New Orleans Pelicans
##  2   2016        2  21600495 2016-12-30         1610612752 New York Knicks
##  3   2021        2  22100943 2022-03-03         1610612742 Dallas Mavericks
##  4   2021        2  22100943 2022-03-03         1610612744 Golden State Warriors
##  5   2016        2  21601032 2017-03-18         1610612741 Chicago Bulls
##  6   2016        2  21601032 2017-03-18         1610612762 Utah Jazz
##  7   2021        2  22100942 2022-03-03         1610612761 Toronto Raptors
##  8   2022        2  22200482 2022-12-23         1610612750 Minnesota Timberwolv…
##  9   2016        2  21600494 2016-12-30         1610612738 Boston Celtics
## 10   2016        2  21600494 2016-12-30         1610612748 Miami Heat
## # ℹ 23,948 more rows
## # ℹ 35 more variables: off_team <chr>, off_home <dbl>, off_win <dbl>,
## #   defensivenbateamid <dbl>, def_team_name <chr>, def_team <chr>,
## #   def_home <dbl>, def_win <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```
# Iterate through each row and decide if each off_team is beating their opponent in eFG percentage
updated_df <- filtered_df %>%
  group_by(nbagameid) %>%
  arrange(nbagameid) %>%
  filter(n() == 2) %>%
  mutate(eFG_percent = (fgmade + 0.5*fg3made) * 100 / fgattempted, opp_eFG_percent = lag(eFG_percent), IsGreater
= eFG_percent > opp_eFG_percent) %>%
  filter(eFG_percent != opp_eFG_percent | is.na(opp_eFG_percent))
updated_df
```

```
## # A tibble: 23,931 × 44
## # Groups:   nbagameid [11,979]
##    season gametype nbagameid gamedate   offensivenbateamid off_team_name
##     <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>
##  1   2014        2  21400001 2014-10-28         1610612740 New Orleans Pelicans
##  2   2014        2  21400001 2014-10-28         1610612753 Orlando Magic
##  3   2014        2  21400002 2014-10-28         1610612742 Dallas Mavericks
##  4   2014        2  21400002 2014-10-28         1610612759 San Antonio Spurs
##  5   2014        2  21400003 2014-10-28         1610612745 Houston Rockets
##  6   2014        2  21400003 2014-10-28         1610612747 Los Angeles Lakers
##  7   2014        2  21400004 2014-10-29         1610612749 Milwaukee Bucks
##  8   2014        2  21400004 2014-10-29         1610612766 Charlotte Hornets
##  9   2014        2  21400005 2014-10-29         1610612754 Indiana Pacers
## 10   2014        2  21400005 2014-10-29         1610612755 Philadelphia 76ers
## # ℹ 23,921 more rows
## # ℹ 38 more variables: off_team <chr>, off_home <dbl>, off_win <dbl>,
## #   defensivenbateamid <dbl>, def_team_name <chr>, def_team <chr>,
## #   def_home <dbl>, def_win <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```
#here we filter down to either the offensive team winning the game and having a greater efg percentage than their
opponent or the offensive team loses the game and has a lower efg percentage than their opponent (their opponent
won the game and had a higher efg percentage which still fits the requirement of a team winning a game with a hig
her efg percentage)
updated_df2 <- updated_df %>%
  filter((off_win == 1 & IsGreater == TRUE) | (off_win == 0 & IsGreater == FALSE))
updated_df2
```

```
## # A tibble: 9,753 × 44
## # Groups:   nbagameid [9,753]
##    season gametype nbagameid gamedate   offensivenbateamid off_team_name
##     <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>
##  1   2014        2  21400001 2014-10-28         1610612753 Orlando Magic
##  2   2014        2  21400002 2014-10-28         1610612759 San Antonio Spurs
##  3   2014        2  21400003 2014-10-28         1610612747 Los Angeles Lakers
##  4   2014        2  21400005 2014-10-29         1610612755 Philadelphia 76ers
##  5   2014        2  21400006 2014-10-29         1610612751 Brooklyn Nets
##  6   2014        2  21400007 2014-10-29         1610612764 Washington Wizards
##  7   2014        2  21400009 2014-10-29         1610612763 Memphis Grizzlies
##  8   2014        2  21400010 2014-10-29         1610612752 New York Knicks
##  9   2014        2  21400012 2014-10-29         1610612762 Utah Jazz
## 10   2014        2  21400013 2014-10-29         1610612756 Phoenix Suns
## # ℹ 9,743 more rows
## # ℹ 38 more variables: off_team <chr>, off_home <dbl>, off_win <dbl>,
## #   defensivenbateamid <dbl>, def_team_name <chr>, def_team <chr>,
## #   def_home <dbl>, def_win <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```
#get the amount of rows that did meet the conditions above
numerator <- updated_df2 %>%
  nrow
print(numerator)
```

```
## [1] 9753
```

```
print(updated_df)
```

```
## # A tibble: 23,931 × 44
## # Groups:   nbagameid [11,979]
##    season gametype nbagameid gamedate   offensivenbateamid off_team_name
##     <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>
##  1   2014        2  21400001 2014-10-28         1610612740 New Orleans Pelicans
##  2   2014        2  21400001 2014-10-28         1610612753 Orlando Magic
##  3   2014        2  21400002 2014-10-28         1610612742 Dallas Mavericks
##  4   2014        2  21400002 2014-10-28         1610612759 San Antonio Spurs
##  5   2014        2  21400003 2014-10-28         1610612745 Houston Rockets
##  6   2014        2  21400003 2014-10-28         1610612747 Los Angeles Lakers
##  7   2014        2  21400004 2014-10-29         1610612749 Milwaukee Bucks
##  8   2014        2  21400004 2014-10-29         1610612766 Charlotte Hornets
##  9   2014        2  21400005 2014-10-29         1610612754 Indiana Pacers
## 10   2014        2  21400005 2014-10-29         1610612755 Philadelphia 76ers
## # ℹ 23,921 more rows
## # ℹ 38 more variables: off_team <chr>, off_home <dbl>, off_win <dbl>,
## #   defensivenbateamid <dbl>, def_team_name <chr>, def_team <chr>,
## #   def_home <dbl>, def_win <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```
#get the total amount of distinct games (get our total sample)
total_games <- updated_df %>%
  summarise(total_games = n_distinct(nbagameid)) %>%
  nrow()
total_games
```

```
## [1] 11979
```

```
#get the final percentage using both numerator and denominator values
percentage = (numerator / total_games) * 100
percentage
```

```
## [1] 81.41748
```

ANSWER 2: 81.42%

81.42%

Question 3

QUESTION: What percent of the time does the team with more offensive rebounds in a given game win that game? Use games from the 2014-2023 regular seasons. If the two teams have an exactly equal number of offensive rebounds, remove that game from the calculation.

```
filtered_df <- team_data %>%
  filter(gametype == 2, season >= 2014, season <= 2023)
filtered_df
```

```
## # A tibble: 23,958 × 41
##    season gametype nbagameid gamedate   offensivenbateamid off_team_name
##     <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>
##  1   2016        2  21600495 2016-12-30         1610612740 New Orleans Pelicans
##  2   2016        2  21600495 2016-12-30         1610612752 New York Knicks
##  3   2021        2  22100943 2022-03-03         1610612742 Dallas Mavericks
##  4   2021        2  22100943 2022-03-03         1610612744 Golden State Warriors
##  5   2016        2  21601032 2017-03-18         1610612741 Chicago Bulls
##  6   2016        2  21601032 2017-03-18         1610612762 Utah Jazz
##  7   2021        2  22100942 2022-03-03         1610612761 Toronto Raptors
##  8   2022        2  22200482 2022-12-23         1610612750 Minnesota Timberwolv…
##  9   2016        2  21600494 2016-12-30         1610612738 Boston Celtics
## 10   2016        2  21600494 2016-12-30         1610612748 Miami Heat
## # ℹ 23,948 more rows
## # ℹ 35 more variables: off_team <chr>, off_home <dbl>, off_win <dbl>,
## #   defensivenbateamid <dbl>, def_team_name <chr>, def_team <chr>,
## #   def_home <dbl>, def_win <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```
# Iterate through each row and decide if each off_team is beating their opponent in offensive rebound count
#have a column that shows this boolean
updated_df_reboff <- filtered_df %>%
  group_by(nbagameid) %>%
  arrange(nbagameid) %>%
  filter(n() == 2) %>%
  mutate(opp_off_rebound = lag(reboffensive), IsGreater = reboffensive > opp_off_rebound) %>%
  filter(reboffensive != opp_off_rebound | is.na(opp_off_rebound))
updated_df_reboff
```

```
## # A tibble: 23,205 × 43
## # Groups:   nbagameid [11,979]
##    season gametype nbagameid gamedate   offensivenbateamid off_team_name
##     <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>
##  1   2014        2  21400001 2014-10-28         1610612740 New Orleans Pelicans
##  2   2014        2  21400001 2014-10-28         1610612753 Orlando Magic
##  3   2014        2  21400002 2014-10-28         1610612742 Dallas Mavericks
##  4   2014        2  21400002 2014-10-28         1610612759 San Antonio Spurs
##  5   2014        2  21400003 2014-10-28         1610612745 Houston Rockets
##  6   2014        2  21400004 2014-10-29         1610612749 Milwaukee Bucks
##  7   2014        2  21400004 2014-10-29         1610612766 Charlotte Hornets
##  8   2014        2  21400005 2014-10-29         1610612754 Indiana Pacers
##  9   2014        2  21400005 2014-10-29         1610612755 Philadelphia 76ers
## 10   2014        2  21400006 2014-10-29         1610612738 Boston Celtics
## # ℹ 23,195 more rows
## # ℹ 37 more variables: off_team <chr>, off_home <dbl>, off_win <dbl>,
## #   defensivenbateamid <dbl>, def_team_name <chr>, def_team <chr>,
## #   def_home <dbl>, def_win <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```
#here we filter down to either the offensive team winning the game and having a greater amount in offensive rebou
nds  than their opponent or the offensive team loses the game and has a lower count of offensive rebounds than th
eir opponent (their opponent won the game and had a higher offensive rebound count which still fits the requireme
nt of a team winning a game with a higher rebound count)
updated_df2 <- updated_df_reboff %>%
  filter((off_win == 1 & IsGreater == TRUE) | (off_win == 0 & IsGreater == FALSE))
updated_df2
```

```
## # A tibble: 5,188 × 43
## # Groups:   nbagameid [5,188]
##    season gametype nbagameid gamedate   offensivenbateamid off_team_name
##     <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>
##  1   2014        2  21400001 2014-10-28         1610612753 Orlando Magic
##  2   2014        2  21400004 2014-10-29         1610612766 Charlotte Hornets
##  3   2014        2  21400007 2014-10-29         1610612764 Washington Wizards
##  4   2014        2  21400008 2014-10-29         1610612761 Toronto Raptors
##  5   2014        2  21400011 2014-10-29         1610612765 Detroit Pistons
##  6   2014        2  21400015 2014-10-29         1610612760 Oklahoma City Thunder
##  7   2014        2  21400016 2014-10-30         1610612764 Washington Wizards
##  8   2014        2  21400020 2014-10-30         1610612760 Oklahoma City Thunder
##  9   2014        2  21400021 2014-10-31         1610612763 Memphis Grizzlies
## 10   2014        2  21400022 2014-10-31         1610612741 Chicago Bulls
## # ℹ 5,178 more rows
## # ℹ 37 more variables: off_team <chr>, off_home <dbl>, off_win <dbl>,
## #   defensivenbateamid <dbl>, def_team_name <chr>, def_team <chr>,
## #   def_home <dbl>, def_win <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```
#get the total number of rows in the dataframe made above
numerator <- updated_df2 %>%
  nrow

#get the total number of distinct rows
denominator <- updated_df_reboff %>%
  summarise(denominator = n_distinct(nbagameid)) %>%
  nrow()

#get the percentage
percentage = (numerator / denominator) * 100
percentage
```

```
## [1] 43.30912
```

ANSWER 3: 43.3%

43.3%

Question 4

QUESTION: Do you have any theories as to why the answer to question 3 is lower than the answer to question 2? Try to be clear and concise with your answer.

ANSWER 4: My theory for why the answer to question 3 is lower than the answer to question 2 is that the Effective Field Goal Percentage is often a good indicator of how many points the team is going to score. Every successful field goal or every shot that goes in, whether it is a three-pointer or two-pointer, contributes immediately to the team's point total. On the other hand, however, although offensive rebounds provide additional opportunities to score, they do not directly increase the point total. At the end of the day, the team still needs to make a successful shot after the rebound to benefit from the extra possession. Poor shooting can definitely negate the benefits of extra possessions gained from offensive rebounds. Another theory as to why a team is more likely to win a game with a better eFG% than their offensive rebounds is that a higher eFG% forces the opposing team to make defensive adjustments, which can open up other scoring opportunities. An instance could be if a team is shooting well from the outside, the defense might extend, leaving more space for drives and interior shots. Offensive rebounds typically do not force the same level of strategic adjustment from the defense.

My theory for why the answer to question 3 is lower than the answer to question 2 is that the Effective Field Goal Percentage is often a good indicator of how many points the team is going to score. Every successful field goal or every shot that goes in, whether it is a three-pointer or two-pointer, contributes immediately to the team's point total. On the other hand, however, although offensive rebounds provide additional opportunities to score, they do not directly increase the point total. At the end of the day, the team still needs to make a successful shot after the rebound to benefit from the extra possession. Poor shooting can definitely negate the benefits of extra possessions gained from offensive rebounds. Another theory as to why a team is more likely to win a game with a better eFG% than their offensive rebounds is that a higher eFG% forces the opposing team to make defensive adjustments, which can open up other scoring opportunities. An instance could be if a team is shooting well from the outside, the defense might extend, leaving more space for drives and interior shots. Offensive rebounds typically do not force the same level of strategic adjustment from the defense.

Question 5

QUESTION: Look at players who played at least 25% of their possible games in a season and scored at least 25 points per game played. Of those player-seasons, what percent of games were they available for on average? Use games from the 2014-2023 regular seasons.

For example:

- Ja Morant does not count in the 2023-24 season, as he played just 9 out of 82 games this year, even though he scored 25.1 points per game.
- Chet Holmgren does not count in the 2023-24 season, as he played all 82 games this year but scored 16.5 points per game.
- LeBron James does count in the 2023-24 season, as he played 71 games and scored 25.7 points per game.

```
player_data
```

```
## # A tibble: 434,797 × 59
##    nbagameid gamedate   season gametype nbapersonid player_name  nbateamid team
##        <dbl> <date>      <dbl>    <dbl>       <dbl> <chr>            <dbl> <chr>
##  1  21700826 2018-02-10   2017        2     1627821 James Webb …    1.61e9 BKN
##  2  21700826 2018-02-10   2017        2     1626156 D'Angelo Ru…    1.61e9 BKN
##  3  21700826 2018-02-10   2017        2      203917 Nik Stauskas    1.61e9 BKN
##  4  21700826 2018-02-10   2017        2     1626143 Jahlil Okaf…    1.61e9 BKN
##  5  21700826 2018-02-10   2017        2      202391 Jeremy Lin      1.61e9 BKN
##  6  21700826 2018-02-10   2017        2      203915 Spencer Din…    1.61e9 BKN
##  7  21700826 2018-02-10   2017        2      201960 DeMarre Car…    1.61e9 BKN
##  8  21700826 2018-02-10   2017        2      203925 Joe Harris      1.61e9 BKN
##  9  21700826 2018-02-10   2017        2      203112 Quincy Acy      1.61e9 BKN
## 10  21700826 2018-02-10   2017        2     1628495 Milton Doyle    1.61e9 BKN
## # ℹ 434,787 more rows
## # ℹ 51 more variables: team_name <chr>, opposingnbateamid <dbl>,
## #   opp_team <chr>, opp_team_name <chr>, starter <dbl>, missed <dbl>,
## #   seconds <dbl>, points <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```
#filtering the playoff data to get only from 2014 to 2023
filtered_df <- player_data %>%
  filter(gametype == 2, season >= 2014, season <= 2023)
filtered_df
```

```
## # A tibble: 387,811 × 59
##    nbagameid gamedate   season gametype nbapersonid player_name  nbateamid team
##        <dbl> <date>      <dbl>    <dbl>       <dbl> <chr>            <dbl> <chr>
##  1  21700826 2018-02-10   2017        2     1627821 James Webb …    1.61e9 BKN
##  2  21700826 2018-02-10   2017        2     1626156 D'Angelo Ru…    1.61e9 BKN
##  3  21700826 2018-02-10   2017        2      203917 Nik Stauskas    1.61e9 BKN
##  4  21700826 2018-02-10   2017        2     1626143 Jahlil Okaf…    1.61e9 BKN
##  5  21700826 2018-02-10   2017        2      202391 Jeremy Lin      1.61e9 BKN
##  6  21700826 2018-02-10   2017        2      203915 Spencer Din…    1.61e9 BKN
##  7  21700826 2018-02-10   2017        2      201960 DeMarre Car…    1.61e9 BKN
##  8  21700826 2018-02-10   2017        2      203925 Joe Harris      1.61e9 BKN
##  9  21700826 2018-02-10   2017        2      203112 Quincy Acy      1.61e9 BKN
## 10  21700826 2018-02-10   2017        2     1628495 Milton Doyle    1.61e9 BKN
## # ℹ 387,801 more rows
## # ℹ 51 more variables: team_name <chr>, opposingnbateamid <dbl>,
## #   opp_team <chr>, opp_team_name <chr>, starter <dbl>, missed <dbl>,
## #   seconds <dbl>, points <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```
#get how many games each player played every season
total_games_per_season_perplayer <- filtered_df %>%
  group_by(player_name, season) %>%
  summarise(total_games = n()) %>%
  arrange(season)
```

```
## `summarise()` has grouped output by 'player_name'. You can override using the
## `.groups` argument.
```

```
total_games_per_season_perplayer
```

```
## # A tibble: 5,477 × 3
## # Groups:   player_name [1,465]
##    player_name     season total_games
##    <chr>            <dbl>       <int>
##  1 A.J. Price        2014          38
##  2 Aaron Brooks      2014          82
##  3 Aaron Gordon      2014          82
##  4 Adreian Payne     2014          83
##  5 Al Horford        2014          82
##  6 Al Jefferson      2014          82
##  7 Al-Farouq Aminu   2014          82
##  8 Alan Anderson     2014          82
##  9 Alec Burks        2014          82
## 10 Alex Kirk         2014          34
## # ℹ 5,467 more rows
```

```
#get only players who averaged more than 25 ppg during the season as well as played at least 25 percent of games
during the regular season
updated_df_playerppg <- filtered_df %>%
  filter(missed == 0) %>%
  group_by(player_name, season) %>%
  summarise(avg_pts = mean(points),
            games_played = n()) %>%
  left_join(total_games_per_season_perplayer, by = c("player_name", "season")) %>%
  mutate(atleast_played_25_percent = games_played / total_games * 100 >= 25,
         atleast_25ppg = avg_pts >= 25) %>%
  filter(atleast_played_25_percent == TRUE, atleast_25ppg == TRUE)
```

```
## `summarise()` has grouped output by 'player_name'. You can override using the
## `.groups` argument.
```

```
updated_df_playerppg
```

```
## # A tibble: 120 × 7
## # Groups:   player_name [35]
##    player_name    season avg_pts games_played total_games atleast_played_25_pe…¹
##    <chr>           <dbl>   <dbl>        <int>       <int> <lgl>
##  1 Anthony Davis    2016    28.0           75          82 TRUE
##  2 Anthony Davis    2017    28.1           75          82 TRUE
##  3 Anthony Davis    2018    25.5           57          82 TRUE
##  4 Anthony Davis    2019    26.1           62          71 TRUE
##  5 Anthony Davis    2022    25.5           57          82 TRUE
##  6 Anthony Edwar…   2023    25.9           79          82 TRUE
##  7 Bradley Beal     2018    25.6           82          82 TRUE
##  8 Bradley Beal     2019    30.5           57          64 TRUE
##  9 Bradley Beal     2020    31.3           60          72 TRUE
## 10 Damian Lillard   2016    27.0           75          82 TRUE
## # ℹ 110 more rows
## # ℹ abbreviated name: ¹atleast_played_25_percent
## # ℹ 1 more variable: atleast_25ppg <lgl>
```

```
#finding out what percent of games the players who averaged 25ppg and played at least 25 percent of games were av
ailable for on average
final_df <- updated_df_playerppg %>%
  mutate(percent_games_played = games_played * 100 / total_games) %>%
  summarise(avg_percent_gp = mean(percent_games_played, na.rm = TRUE))
final_df
```

```
## # A tibble: 35 × 2
##    player_name           avg_percent_gp
##    <chr>                          <dbl>
##  1 Anthony Davis                   81.9
##  2 Anthony Edwards                 96.3
##  3 Bradley Beal                    90.8
##  4 Damian Lillard                  88.7
##  5 De'Aaron Fox                    86.6
##  6 DeMar DeRozan                   91.5
##  7 DeMarcus Cousins                73.8
##  8 Devin Booker                    82.9
##  9 Donovan Mitchell                76.6
## 10 Giannis Antetokounmpo           85.6
## # ℹ 25 more rows
```

```
#get the mean of all the players average percents (mean of mean percents)
mean_of_means <- mean(final_df$avg_percent_gp, na.rm = TRUE)
mean_of_means
```

```
## [1] 84.48669
```

ANSWER 5:

84.5% of games

Question 6

QUESTION: What % of playoff series are won by the team with home court advantage? Give your answer by round. Use playoffs series from the 2014-**2022** seasons. Remember that the 2023 playoffs took place during the 2022 season (i.e. 2022-23 season).

```r
#loading in playoff data between 2014 and 2022 seasons
playoff_games <- team_data %>%
  filter(gametype == 4, season >= 2014, season <= 2022)
playoff_games
```

```
## # A tibble: 1,498 × 41
##    season gametype nbagameid gamedate   offensivenbateamid off_team_name
##     <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>
##  1   2021        4  42100406 2022-06-16         1610612738 Boston Celtics
##  2   2021        4  42100406 2022-06-16         1610612744 Golden State Warriors
##  3   2021        4  42100405 2022-06-13         1610612738 Boston Celtics
##  4   2021        4  42100405 2022-06-13         1610612744 Golden State Warriors
##  5   2021        4  42100404 2022-06-10         1610612738 Boston Celtics
##  6   2021        4  42100404 2022-06-10         1610612744 Golden State Warriors
##  7   2021        4  42100403 2022-06-08         1610612738 Boston Celtics
##  8   2021        4  42100403 2022-06-08         1610612744 Golden State Warriors
##  9   2021        4  42100402 2022-06-05         1610612738 Boston Celtics
## 10   2021        4  42100402 2022-06-05         1610612744 Golden State Warriors
## # ℹ 1,488 more rows
## # ℹ 35 more variables: off_team <chr>, off_home <dbl>, off_win <dbl>,
## #   defensivenbateamid <dbl>, def_team_name <chr>, def_team <chr>,
## #   def_home <dbl>, def_win <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```r
#assign round to each matchup in the playoff game data
get_round <- function(opponent_change) {
  cumsum(opponent_change) + 1
}

data <- playoff_games %>%
  arrange(season, gamedate) %>%
  group_by(season, off_team) %>%
  mutate(new_opponent = defensivenbateamid != lag(defensivenbateamid, default = first(defensivenbateamid))) %>%
  mutate(round = get_round(new_opponent))
data
```

```
## # A tibble: 1,498 × 43
## # Groups:   season, off_team [144]
##    season gametype nbagameid gamedate   offensivenbateamid off_team_name
##     <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>
##  1   2014        4  41400151 2015-04-18         1610612742 Dallas Mavericks
##  2   2014        4  41400151 2015-04-18         1610612745 Houston Rockets
##  3   2014        4  41400131 2015-04-18         1610612761 Toronto Raptors
##  4   2014        4  41400131 2015-04-18         1610612764 Washington Wizards
##  5   2014        4  41400141 2015-04-18         1610612740 New Orleans Pelicans
##  6   2014        4  41400141 2015-04-18         1610612744 Golden State Warriors
##  7   2014        4  41400121 2015-04-18         1610612741 Chicago Bulls
##  8   2014        4  41400121 2015-04-18         1610612749 Milwaukee Bucks
##  9   2014        4  41400171 2015-04-19         1610612757 Portland Trail Blaze…
## 10   2014        4  41400171 2015-04-19         1610612763 Memphis Grizzlies
## # ℹ 1,488 more rows
## # ℹ 37 more variables: off_team <chr>, off_home <dbl>, off_win <dbl>,
## #   defensivenbateamid <dbl>, def_team_name <chr>, def_team <chr>,
## #   def_home <dbl>, def_win <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```r
#for 1st round (same code repeated for all four rounds)
round1_series <- data %>%
  filter(round == 1) %>%
  arrange(nbagameid) %>%
  mutate(series_id = if_else(off_team < def_team,
                             paste(off_team, def_team, sep = "-"),
                             paste(def_team, off_team, sep = "-")),
         home_court = if_else(first(off_home) < first(def_home),
                             paste(first(def_team)),
                             paste(first(off_team)))) %>%

  group_by(series_id) %>%
  mutate(label = if_else(off_team == home_court, "home court advantage", NA_character_))
round1_series
```

```
## # A tibble: 772 × 46
## # Groups:   series_id [62]
##    season gametype nbagameid gamedate   offensivenbateamid off_team_name
##     <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>
##  1   2014        4  41400101 2015-04-19         1610612737 Atlanta Hawks
##  2   2014        4  41400101 2015-04-19         1610612751 Brooklyn Nets
##  3   2014        4  41400102 2015-04-22         1610612737 Atlanta Hawks
##  4   2014        4  41400102 2015-04-22         1610612751 Brooklyn Nets
##  5   2014        4  41400103 2015-04-25         1610612737 Atlanta Hawks
##  6   2014        4  41400103 2015-04-25         1610612751 Brooklyn Nets
##  7   2014        4  41400104 2015-04-27         1610612737 Atlanta Hawks
##  8   2014        4  41400104 2015-04-27         1610612751 Brooklyn Nets
##  9   2014        4  41400105 2015-04-29         1610612737 Atlanta Hawks
## 10   2014        4  41400105 2015-04-29         1610612751 Brooklyn Nets
## # ℹ 762 more rows
## # ℹ 40 more variables: off_team <chr>, off_home <dbl>, off_win <dbl>,
## #   defensivenbateamid <dbl>, def_team_name <chr>, def_team <chr>,
## #   def_home <dbl>, def_win <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```
#showing who won with home court in the first round
won_with_hca1 <- round1_series %>%
  group_by(season, off_team, series_id, label) %>%
  summarise(total_wins_inseries = sum(off_win, na.rm = TRUE)) %>%
  filter(total_wins_inseries == 4 & label == "home court advantage")
```

```
## `summarise()` has grouped output by 'season', 'off_team', 'series_id'. You can
## override using the `.groups` argument.
```

```
won_with_hca1
```

```
## # A tibble: 61 × 5
## # Groups:   season, off_team, series_id [61]
##    season off_team series_id label                total_wins_inseries
##     <dbl> <chr>    <chr>     <chr>                              <dbl>
##  1   2014 ATL      ATL-BKN   home court advantage                   4
##  2   2014 CHI      CHI-MIL   home court advantage                   4
##  3   2014 CLE      BOS-CLE   home court advantage                   4
##  4   2014 GSW      GSW-NOP   home court advantage                   4
##  5   2014 HOU      DAL-HOU   home court advantage                   4
##  6   2014 LAC      LAC-SAS   home court advantage                   4
##  7   2014 MEM      MEM-POR   home court advantage                   4
##  8   2015 ATL      ATL-BOS   home court advantage                   4
##  9   2015 CLE      CLE-DET   home court advantage                   4
## 10   2015 GSW      GSW-HOU   home court advantage                   4
## # ℹ 51 more rows
```

```
#showing only the distinct first round matchups (round 1 series)
distinct_round1_series <- round1_series %>%
  group_by(season, series_id) %>%
  distinct(series_id) %>%
  nrow()
distinct_round1_series
```

```
## [1] 72
```

```
#the percentage that shows how many of the teams with homecourt advantage won the first round
percent_wonwithhc_rd1 <- (nrow(won_with_hca1) / distinct_round1_series) * 100
percent_wonwithhc_rd1
```

```
## [1] 84.72222
```

```
#########2nd ROUND################

#all of the seocnd round matchups over the years (I assigned series IDs to each as well as identified who has hom
e court)
round2_series <- data %>%
  filter(round == 2) %>%
  arrange(nbagameid) %>%
  mutate(series_id = if_else(first(off_home) < first(def_home),
                            paste(first(def_team), first(off_team), sep = "-"),
                            paste(first(off_team), first(def_team), sep = "-")),
         home_court = if_else(first(off_home) < first(def_home),
                            paste(first(def_team)),
                            paste(first(off_team)))) %>%

  group_by(series_id) %>%
  mutate(label = if_else(off_team == home_court, "home court advantage", NA_character_))
round2_series
```

```
## # A tibble: 422 × 46
## # Groups:   series_id [35]
##    season gametype nbagameid gamedate   offensivenbateamid off_team_name
##     <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>
##  1   2014        4  41400201 2015-05-03         1610612737 Atlanta Hawks
##  2   2014        4  41400201 2015-05-03         1610612764 Washington Wizards
##  3   2014        4  41400202 2015-05-05         1610612737 Atlanta Hawks
##  4   2014        4  41400202 2015-05-05         1610612764 Washington Wizards
##  5   2014        4  41400203 2015-05-09         1610612737 Atlanta Hawks
##  6   2014        4  41400203 2015-05-09         1610612764 Washington Wizards
##  7   2014        4  41400204 2015-05-11         1610612737 Atlanta Hawks
##  8   2014        4  41400204 2015-05-11         1610612764 Washington Wizards
##  9   2014        4  41400205 2015-05-13         1610612737 Atlanta Hawks
## 10   2014        4  41400205 2015-05-13         1610612764 Washington Wizards
## # ℹ 412 more rows
## # ℹ 40 more variables: off_team <chr>, off_home <dbl>, off_win <dbl>,
## #   defensivenbateamid <dbl>, def_team_name <chr>, def_team <chr>,
## #   def_home <dbl>, def_win <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```
#showing who won with home court in the second round
won_with_hca2 <- round2_series %>%
  group_by(season, off_team, series_id, label) %>%
  summarise(total_wins_inseries = sum(off_win, na.rm = TRUE)) %>%
  filter(total_wins_inseries == 4 & label == "home court advantage")
```

```
## `summarise()` has grouped output by 'season', 'off_team', 'series_id'. You can
## override using the `.groups` argument.
```

```
won_with_hca2
```

```
## # A tibble: 23 × 5
## # Groups:   season, off_team, series_id [23]
##    season off_team series_id label                total_wins_inseries
##     <dbl> <chr>    <chr>     <chr>                              <dbl>
##  1   2014 ATL      ATL-WAS   home court advantage                   4
##  2   2014 CLE      CLE-CHI   home court advantage                   4
##  3   2014 GSW      GSW-MEM   home court advantage                   4
##  4   2014 HOU      HOU-LAC   home court advantage                   4
##  5   2015 CLE      CLE-ATL   home court advantage                   4
##  6   2015 GSW      GSW-POR   home court advantage                   4
##  7   2015 TOR      TOR-MIA   home court advantage                   4
##  8   2016 BOS      BOS-WAS   home court advantage                   4
##  9   2016 CLE      CLE-TOR   home court advantage                   4
## 10   2016 GSW      GSW-UTA   home court advantage                   4
## # ℹ 13 more rows
```

```
#showing only the distinct second round matchups (round 2 series)
distinct_round2_series <- round2_series %>%
  group_by(season, series_id) %>%
  distinct(series_id) %>%
  nrow()
distinct_round2_series
```

```
## [1] 36
```

```
#the percentage that shows how many of the teams with homecourt advantage won the second round
percent_wonwithhc_rd2 <- (nrow(won_with_hca2) / distinct_round2_series) * 100
percent_wonwithhc_rd2
```

```
## [1] 63.88889
```

```
############3rd ROUND (Conference Finals)############

#all of the conference finals matchups over the years (I assigned series IDs to each as well as identified who ha
s home court)
round3_series <- data %>%
  filter(round == 3) %>%
  arrange(nbagameid) %>%
  mutate(series_id = if_else(first(off_home) < first(def_home),
                             paste(first(def_team), first(off_team), sep = "-"),
                             paste(first(off_team), first(def_team), sep = "-")),
         home_court = if_else(first(off_home) < first(def_home),
                             paste(first(def_team)),
                             paste(first(off_team)))) %>%

  group_by(series_id) %>%
  mutate(label = if_else(off_team == home_court, "home court advantage", NA_character_))
round3_series
```

```
## # A tibble: 202 × 46
## # Groups:   series_id [16]
##    season gametype nbagameid gamedate   offensivenbateamid off_team_name
##     <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>
##  1   2014        4  41400301 2015-05-20         1610612737 Atlanta Hawks
##  2   2014        4  41400301 2015-05-20         1610612739 Cleveland Cavaliers
##  3   2014        4  41400302 2015-05-22         1610612737 Atlanta Hawks
##  4   2014        4  41400302 2015-05-22         1610612739 Cleveland Cavaliers
##  5   2014        4  41400303 2015-05-24         1610612737 Atlanta Hawks
##  6   2014        4  41400303 2015-05-24         1610612739 Cleveland Cavaliers
##  7   2014        4  41400304 2015-05-26         1610612737 Atlanta Hawks
##  8   2014        4  41400304 2015-05-26         1610612739 Cleveland Cavaliers
##  9   2014        4  41400311 2015-05-19         1610612744 Golden State Warriors
## 10   2014        4  41400311 2015-05-19         1610612745 Houston Rockets
## # ℹ 192 more rows
## # ℹ 40 more variables: off_team <chr>, off_home <dbl>, off_win <dbl>,
## #   defensivenbateamid <dbl>, def_team_name <chr>, def_team <chr>,
## #   def_home <dbl>, def_win <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```
#showing who won with home court in the conference finals
won_with_hca3 <- round3_series %>%
  group_by(season, off_team, series_id, label) %>%
  summarise(total_wins_inseries = sum(off_win, na.rm = TRUE)) %>%
  filter(total_wins_inseries == 4 & label == "home court advantage")
```

```
## `summarise()` has grouped output by 'season', 'off_team', 'series_id'. You can
## override using the `.groups` argument.
```

```
won_with_hca3
```

```
## # A tibble: 10 × 5
## # Groups:   season, off_team, series_id [10]
##    season off_team series_id label                total_wins_inseries
##     <dbl> <chr>    <chr>     <chr>                              <dbl>
##  1   2014 GSW      GSW-HOU   home court advantage                   4
##  2   2015 CLE      CLE-TOR   home court advantage                   4
##  3   2015 GSW      GSW-OKC   home court advantage                   4
##  4   2016 GSW      GSW-SAS   home court advantage                   4
##  5   2018 GSW      GSW-POR   home court advantage                   4
##  6   2019 LAL      LAL-DEN   home court advantage                   4
##  7   2020 MIL      MIL-ATL   home court advantage                   4
##  8   2020 PHX      PHX-LAC   home court advantage                   4
##  9   2021 GSW      GSW-DAL   home court advantage                   4
## 10   2022 DEN      DEN-LAL   home court advantage                   4
```

```
#showing only the distinct conference finals matchups (round 3 series)
distinct_round3_series <- round3_series %>%
  group_by(season, series_id) %>%
  distinct(series_id) %>%
  nrow()
distinct_round3_series
```

```
## [1] 18
```

```
#the percentage that shows how many of the teams with homecourt advantage won the conference finals
percent_wonwithhc_rd3 <- (nrow(won_with_hca3) / distinct_round3_series) * 100
percent_wonwithhc_rd3
```

```
## [1] 55.55556
```

```
#############4th ROUND(NBA Finals)################

#all of the Finals matchups over the years (I am filtering based on round and assigning series IDs)
round4_series <- data %>%
  filter(round == 4) %>%
  arrange(nbagameid) %>%
  mutate(series_id = if_else(first(off_home) < first(def_home),
                             paste(first(def_team), first(off_team), sep = "-"),
                             paste(first(off_team), first(def_team), sep = "-")),
         home_court = if_else(first(off_home) < first(def_home),
                             paste(first(def_team)),
                             paste(first(off_team)))) %>%

  group_by(series_id) %>%
  mutate(label = if_else(off_team == home_court, "home court advantage", NA_character_))
round4_series
```

```
## # A tibble: 102 × 46
## # Groups:   series_id [6]
##    season gametype nbagameid gamedate   offensivenbateamid off_team_name
##     <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>
##  1   2014        4  41400401 2015-06-04         1610612739 Cleveland Cavaliers
##  2   2014        4  41400401 2015-06-04         1610612744 Golden State Warriors
##  3   2014        4  41400402 2015-06-07         1610612739 Cleveland Cavaliers
##  4   2014        4  41400402 2015-06-07         1610612744 Golden State Warriors
##  5   2014        4  41400403 2015-06-09         1610612739 Cleveland Cavaliers
##  6   2014        4  41400403 2015-06-09         1610612744 Golden State Warriors
##  7   2014        4  41400404 2015-06-11         1610612739 Cleveland Cavaliers
##  8   2014        4  41400404 2015-06-11         1610612744 Golden State Warriors
##  9   2014        4  41400405 2015-06-14         1610612739 Cleveland Cavaliers
## 10   2014        4  41400405 2015-06-14         1610612744 Golden State Warriors
## # ℹ 92 more rows
## # ℹ 40 more variables: off_team <chr>, off_home <dbl>, off_win <dbl>,
## #   defensivenbateamid <dbl>, def_team_name <chr>, def_team <chr>,
## #   def_home <dbl>, def_win <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```
#showing how many teams won the series with home court
won_with_hca4 <- round4_series %>%
  group_by(season, off_team, series_id, label) %>%
  summarise(total_wins_inseries = sum(off_win, na.rm = TRUE)) %>%
  filter(total_wins_inseries == 4 & label == "home court advantage")
```

```
## `summarise()` has grouped output by 'season', 'off_team', 'series_id'. You can
## override using the `.groups` argument.
```

```
won_with_hca4
```

```
## # A tibble: 7 × 5
## # Groups:   season, off_team, series_id [7]
##   season off_team series_id label                total_wins_inseries
##    <dbl> <chr>    <chr>     <chr>                              <dbl>
## 1   2014 GSW      GSW-CLE   home court advantage                   4
## 2   2016 GSW      GSW-CLE   home court advantage                   4
## 3   2017 GSW      GSW-CLE   home court advantage                   4
## 4   2018 TOR      TOR-GSW   home court advantage                   4
## 5   2019 LAL      LAL-MIA   home court advantage                   4
## 6   2021 GSW      GSW-BOS   home court advantage                   4
## 7   2022 DEN      DEN-MIA   home court advantage                   4
```

```
#showing only the distinct Finals matchups (round 4 series)
distinct_round4_series <- round4_series %>%
  group_by(season, series_id) %>%
  distinct(series_id) %>%
  nrow()
distinct_round4_series
```

```
## [1] 9
```

```
#the percentage that shows how many of the teams with homecourt advantage won the finals
percent_wonwithhc_rd4 <- (nrow(won_with_hca4) / distinct_round4_series) * 100
percent_wonwithhc_rd4
```

```
## [1] 77.77778
```

ANSWER 6:

Round 1: 84.72%
Round 2: 63.9%
Conference Finals: 55.56%
Finals: 77.78%

Question 7

QUESTION: Among teams that had at least a +5.0 net rating in the regular season, what percent of them made the second round of the playoffs the **following** year? Among those teams, what percent of their top 5 total minutes played players (regular season) in the +5.0 net rating season played in that 2nd round playoffs series? Use the 2014-2021 regular seasons to determine the +5 teams and the 2015-2022 seasons of playoffs data.

For example, the Thunder had a better than +5 net rating in the 2023 season. If we make the 2nd round of the playoffs **next** season (2024-25), we would qualify for this question. Our top 5 minutes played players this season were Shai Gilgeous-Alexander, Chet Holmgren, Luguentz Dort, Jalen Williams, and Josh Giddey. If three of them play in a hypothetical 2nd round series next season, it would count as 3/5 for this question.

Hint: The definition for net rating is in the data dictionary.

```
#part 1
regular_season_data <- team_data %>%
  filter(gametype == 2, season >= 2014, season <= 2021)
regular_season_data
```

```
## # A tibble: 19,038 × 41
##    season gametype nbagameid gamedate   offensivenbateamid off_team_name
##     <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>
##  1   2016        2  21600495 2016-12-30         1610612740 New Orleans Pelicans
##  2   2016        2  21600495 2016-12-30         1610612752 New York Knicks
##  3   2021        2  22100943 2022-03-03         1610612742 Dallas Mavericks
##  4   2021        2  22100943 2022-03-03         1610612744 Golden State Warriors
##  5   2016        2  21601032 2017-03-18         1610612741 Chicago Bulls
##  6   2016        2  21601032 2017-03-18         1610612762 Utah Jazz
##  7   2021        2  22100942 2022-03-03         1610612761 Toronto Raptors
##  8   2016        2  21600494 2016-12-30         1610612738 Boston Celtics
##  9   2016        2  21600494 2016-12-30         1610612748 Miami Heat
## 10   2017        2  21700768 2018-02-02         1610612747 Los Angeles Lakers
## # ℹ 19,028 more rows
## # ℹ 35 more variables: off_team <chr>, off_home <dbl>, off_win <dbl>,
## #   defensivenbateamid <dbl>, def_team_name <chr>, def_team <chr>,
## #   def_home <dbl>, def_win <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```
new_playoff_games <- team_data %>%
  filter(gametype == 4, season >= 2015, season <= 2022)
new_playoff_games
```

```
## # A tibble: 1,336 × 41
##    season gametype nbagameid gamedate   offensivenbateamid off_team_name
##     <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>
##  1   2021        4  42100406 2022-06-16         1610612738 Boston Celtics
##  2   2021        4  42100406 2022-06-16         1610612744 Golden State Warriors
##  3   2021        4  42100405 2022-06-13         1610612738 Boston Celtics
##  4   2021        4  42100405 2022-06-13         1610612744 Golden State Warriors
##  5   2021        4  42100404 2022-06-10         1610612738 Boston Celtics
##  6   2021        4  42100404 2022-06-10         1610612744 Golden State Warriors
##  7   2021        4  42100403 2022-06-08         1610612738 Boston Celtics
##  8   2021        4  42100403 2022-06-08         1610612744 Golden State Warriors
##  9   2021        4  42100402 2022-06-05         1610612738 Boston Celtics
## 10   2021        4  42100402 2022-06-05         1610612744 Golden State Warriors
## # ℹ 1,326 more rows
## # ℹ 35 more variables: off_team <chr>, off_home <dbl>, off_win <dbl>,
## #   defensivenbateamid <dbl>, def_team_name <chr>, def_team <chr>,
## #   def_home <dbl>, def_win <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```
#get the round of each matchup
get_round <- function(opponent_change) {
  cumsum(opponent_change) + 1
}

# get all the teams who made the 2nd round of the playoffs every year based on the limits we gave earlier
nextseason_playoffdata <- new_playoff_games %>%
  arrange(season, gamedate) %>%
  group_by(season, off_team) %>%
  mutate(new_opponent = defensivenbateamid != lag(defensivenbateamid, default = first(defensivenbateamid))) %>%
  mutate(round = get_round(new_opponent)) %>%
  filter(round == 2) %>%
  rename(team_name = off_team_name)
nextseason_playoffdata
```

```
## # A tibble: 372 × 43
## # Groups:   season, off_team [64]
##    season gametype nbagameid gamedate   offensivenbateamid team_name    off_team
##     <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>        <chr>
##  1   2015        4  41500231 2016-04-30         1610612759 San Antonio… SAS
##  2   2015        4  41500231 2016-04-30         1610612760 Oklahoma Ci… OKC
##  3   2015        4  41500221 2016-05-01         1610612744 Golden Stat… GSW
##  4   2015        4  41500221 2016-05-01         1610612757 Portland Tr… POR
##  5   2015        4  41500232 2016-05-02         1610612759 San Antonio… SAS
##  6   2015        4  41500232 2016-05-02         1610612760 Oklahoma Ci… OKC
##  7   2015        4  41500201 2016-05-02         1610612737 Atlanta Haw… ATL
##  8   2015        4  41500201 2016-05-02         1610612739 Cleveland C… CLE
##  9   2015        4  41500222 2016-05-03         1610612744 Golden Stat… GSW
## 10   2015        4  41500222 2016-05-03         1610612757 Portland Tr… POR
## # ℹ 362 more rows
## # ℹ 36 more variables: off_home <dbl>, off_win <dbl>, defensivenbateamid <dbl>,
## #   def_team_name <chr>, def_team <chr>, def_home <dbl>, def_win <dbl>,
## #   fg2made <dbl>, fg2missed <dbl>, fg2attempted <dbl>, fg3made <dbl>,
## #   fg3missed <dbl>, fg3attempted <dbl>, fgmade <dbl>, fgmissed <dbl>,
## #   fgattempted <dbl>, ftmade <dbl>, ftmissed <dbl>, ftattempted <dbl>,
## #   reboffensive <dbl>, rebdefensive <dbl>, reboundchance <dbl>, …
```

```
#show the offensive rating for each team
ORTG_df <- regular_season_data %>%
  group_by(season, off_team, off_team_name) %>%
  rename(team = off_team) %>%
  arrange(season) %>%
  summarise(summed_offpoints = sum(points),
            summed_offpossessions = sum(possessions),
            ORTG = summed_offpoints/(summed_offpossessions/100)) %>%
  rename(team_name = off_team_name)
```

```
## `summarise()` has grouped output by 'season', 'team'. You can override using the
## `.groups` argument.
```

```
ORTG_df
```

```
## # A tibble: 240 × 6
## # Groups:   season, team [240]
##    season team  team_name           summed_offpoints summed_offpossessions  ORTG
##     <dbl> <chr> <chr>                          <dbl>                 <dbl> <dbl>
##  1   2014 ATL   Atlanta Hawks                   8409                  7710  109.
##  2   2014 BKN   Brooklyn Nets                   8038                  7711  104.
##  3   2014 BOS   Boston Celtics                  8312                  7937  105.
##  4   2014 CHA   Charlotte Hornets               7721                  7671  101.
##  5   2014 CHI   Chicago Bulls                   8265                  7702  107.
##  6   2014 CLE   Cleveland Cavaliers             8457                  7622  111.
##  7   2014 DAL   Dallas Mavericks                8628                  7903  109.
##  8   2014 DEN   Denver Nuggets                  8320                  7975  104.
##  9   2014 DET   Detroit Pistons                 8077                  7668  105.
## 10   2014 GSW   Golden State Warri…             9016                  8081  112.
## # ℹ 230 more rows
```

```
#show the defensive rating for each team
DRTG_df <- regular_season_data %>%
  group_by(season, def_team, def_team_name) %>%
  rename(team = def_team) %>%
  arrange(season) %>%
  summarise(summed_defpoints = sum(points),
            summed_defpossessions = sum(possessions),
            DRTG = summed_defpoints/(summed_defpossessions/100))%>%
  rename(team_name = def_team_name)
```

```
## `summarise()` has grouped output by 'season', 'team'. You can override using the
## `.groups` argument.
```

```
DRTG_df
```

```
## # A tibble: 240 × 6
## # Groups:   season, team [240]
##    season team  team_name           summed_defpoints summed_defpossessions  DRTG
##     <dbl> <chr> <chr>                          <dbl>                 <dbl> <dbl>
##  1   2014 ATL   Atlanta Hawks                   7964                  7728  103.
##  2   2014 BKN   Brooklyn Nets                   8274                  7715  107.
##  3   2014 BOS   Boston Celtics                  8299                  7928  105.
##  4   2014 CHA   Charlotte Hornets               7981                  7681  104.
##  5   2014 CHI   Chicago Bulls                   8019                  7705  104.
##  6   2014 CLE   Cleveland Cavaliers             8090                  7619  106.
##  7   2014 DAL   Dallas Mavericks                8390                  7905  106.
##  8   2014 DEN   Denver Nuggets                  8611                  7960  108.
##  9   2014 DET   Detroit Pistons                 8157                  7653  107.
## 10   2014 GSW   Golden State Warri…             8188                  8111  101.
## # ℹ 230 more rows
```

```
#the plus 5 net rating teams each season based on season and team name
plus5_netrating <- ORTG_df %>%
  left_join(DRTG_df, by = c("season", "team_name")) %>%
  mutate(NRTG = ORTG - DRTG) %>%
  filter(NRTG >= 5.0)
plus5_netrating
```

```
## # A tibble: 33 × 11
## # Groups:   season [8]
##    season team.x team_name   summed_offpoints summed_offpossessions  ORTG team.y
##     <dbl> <chr>  <chr>                  <dbl>                 <dbl> <dbl> <chr>
##  1   2014 ATL    Atlanta Ha…             8409                  7710  109. ATL
##  2   2014 GSW    Golden Sta…             9016                  8081  112. GSW
##  3   2014 LAC    LA Clippers             8751                  7782  112. LAC
##  4   2014 SAS    San Antoni…             8461                  7798  109. SAS
##  5   2015 CLE    Cleveland …             8555                  7723  111. CLE
##  6   2015 GSW    Golden Sta…             9421                  8265  114. GSW
##  7   2015 OKC    Oklahoma C…             9038                  8015  113. OKC
##  8   2015 SAS    San Antoni…             8490                  7733  110. SAS
##  9   2016 GSW    Golden Sta…             9503                  8214  116. GSW
## 10   2016 HOU    Houston Ro…             9458                  8219  115. HOU
## # ℹ 23 more rows
## # ℹ 4 more variables: summed_defpoints <dbl>, summed_defpossessions <dbl>,
## #   DRTG <dbl>, NRTG <dbl>
```

```
#combine both the dataframe that has all the playoff teams and the plus 5 net rating teams
#filter those both showing which plus 5 net rating team made the playoffs the following year
joined_data <- plus5_netrating %>%
  left_join(nextseason_playoffdata, by = "team_name") %>%
  filter(season.y - season.x == 1)
```

```
## Warning in left_join(., nextseason_playoffdata, by = "team_name"): Detected an
## unexpected many-to-many relationship between `x` and `y`.
```

```
joined_data
```

```
## # A tibble: 122 × 53
##    season.x team.x team_name summed_offpoints summed_offpossessions  ORTG team.y
##       <dbl> <chr>  <chr>                <dbl>                 <dbl> <dbl> <chr>
##  1     2014 ATL    Atlanta …             8409                  7710  109. ATL
##  2     2014 ATL    Atlanta …             8409                  7710  109. ATL
##  3     2014 ATL    Atlanta …             8409                  7710  109. ATL
##  4     2014 ATL    Atlanta …             8409                  7710  109. ATL
##  5     2014 GSW    Golden S…             9016                  8081  112. GSW
##  6     2014 GSW    Golden S…             9016                  8081  112. GSW
##  7     2014 GSW    Golden S…             9016                  8081  112. GSW
##  8     2014 GSW    Golden S…             9016                  8081  112. GSW
##  9     2014 GSW    Golden S…             9016                  8081  112. GSW
## 10     2014 SAS    San Anto…             8461                  7798  109. SAS
## # ℹ 112 more rows
## # ℹ 46 more variables: summed_defpoints <dbl>, summed_defpossessions <dbl>,
## #   DRTG <dbl>, NRTG <dbl>, season.y <dbl>, gametype <dbl>, nbagameid <dbl>,
## #   gamedate <date>, offensivenbateamid <dbl>, off_team <chr>, off_home <dbl>,
## #   off_win <dbl>, defensivenbateamid <dbl>, def_team_name <chr>,
## #   def_team <chr>, def_home <dbl>, def_win <dbl>, fg2made <dbl>,
## #   fg2missed <dbl>, fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, …
```

```
#get the numerator to calculate the percentage
unique_counts_num <- joined_data %>%
  group_by(season.x, team_name) %>%
  summarise(unique_teams_everyseason = n_distinct(team_name)) %>%
  summarise(total_unique_teams = sum(unique_teams_everyseason)) %>%
  summarise(total_teams = sum(total_unique_teams, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'season.x'. You can override using the
## `.groups` argument.
```

```
unique_counts_num
```

```
## # A tibble: 1 × 1
##   total_teams
##         <int>
## 1          21
```

```
#get the denominator to calculate the percentage
count_denom <- plus5_netrating %>%
  summarise(count = n()) %>%
  summarise(total_teams = sum(count, na.rm = TRUE))
count_denom
```

```
## # A tibble: 1 × 1
##   total_teams
##         <int>
## 1          33
```

```
#finding out the percentage of teams that were +5.0 net rating actually made the second round of playoffs the fol
lowing year
percent_made_secondrd <- unique_counts_num$total_teams * 100 / count_denom$total_teams
percent_made_secondrd
```

```
## [1] 63.63636
```

```
#Part 2

#getting the minutes each player played on each team during each season using the intially given player data
minutes_perteam <- player_data %>%
  group_by(season, team_name, nbapersonid, player_name) %>%
  filter(gametype == 2) %>%
  summarise(totalmins = sum(seconds)/60)
```

```
## `summarise()` has grouped output by 'season', 'team_name', 'nbapersonid'. You
## can override using the `.groups` argument.
```

```
minutes_perteam
```

```
## # A tibble: 6,372 × 5
## # Groups:   season, team_name, nbapersonid [6,372]
##    season team_name     nbapersonid player_name     totalmins
##     <dbl> <chr>               <dbl> <chr>               <dbl>
##  1   2014 Atlanta Hawks        1882 Elton Brand         485.
##  2   2014 Atlanta Hawks        2594 Kyle Korver        2418.
##  3   2014 Atlanta Hawks      200757 Thabo Sefolosha     976.
##  4   2014 Atlanta Hawks      200794 Paul Millsap       2390.
##  5   2014 Atlanta Hawks      201143 Al Horford         2318.
##  6   2014 Atlanta Hawks      201948 Austin Daye          75.7
##  7   2014 Atlanta Hawks      201952 Jeff Teague        2228.
##  8   2014 Atlanta Hawks      201960 DeMarre Carroll    2189.
##  9   2014 Atlanta Hawks      202714 Shelvin Mack        833.
## 10   2014 Atlanta Hawks      203098 John Jenkins        297.
## # ℹ 6,362 more rows
```

```
#getting the top 5 minutes played players fro each of the +5.0 net rating teams during the regular season
new_df3 <- plus5_netrating %>%
  left_join(minutes_perteam, by = c("season", "team_name")) %>%
  group_by(season, team_name) %>%
  arrange(team_name, desc(totalmins)) %>%
  distinct(player_name, .keep_all = TRUE) %>%
  rename(regular_season = season) %>%
  slice_head(n = 5)
new_df3
```

```
## # A tibble: 165 × 14
## # Groups:   regular_season, team_name [33]
##    regular_season team.x team_name  summed_offpoints summed_offpossessions  ORTG
##             <dbl> <chr>  <chr>                 <dbl>                 <dbl> <dbl>
##  1           2014 ATL    Atlanta H…             8409                  7710  109.
##  2           2014 ATL    Atlanta H…             8409                  7710  109.
##  3           2014 ATL    Atlanta H…             8409                  7710  109.
##  4           2014 ATL    Atlanta H…             8409                  7710  109.
##  5           2014 ATL    Atlanta H…             8409                  7710  109.
##  6           2014 GSW    Golden St…             9016                  8081  112.
##  7           2014 GSW    Golden St…             9016                  8081  112.
##  8           2014 GSW    Golden St…             9016                  8081  112.
##  9           2014 GSW    Golden St…             9016                  8081  112.
## 10           2014 GSW    Golden St…             9016                  8081  112.
## # ℹ 155 more rows
## # ℹ 8 more variables: team.y <chr>, summed_defpoints <dbl>,
## #   summed_defpossessions <dbl>, DRTG <dbl>, NRTG <dbl>, nbapersonid <dbl>,
## #   player_name <chr>, totalmins <dbl>
```

```
#getting all of the team rosters showing the players who played in each playoff series every year
players_who_played_nextyearplayoff <- nextseason_playoffdata %>%
  left_join(player_data, by = c("season", "nbagameid", "team_name"))
players_who_played_nextyearplayoff
```

```
## # A tibble: 5,733 × 99
## # Groups:   season, off_team [64]
##    season gametype.x nbagameid gamedate.x offensivenbateamid team_name  off_team
##     <dbl>      <dbl>     <dbl> <date>                  <dbl> <chr>      <chr>
##  1   2015          4  41500231 2016-04-30         1610612759 San Anton… SAS
##  2   2015          4  41500231 2016-04-30         1610612759 San Anton… SAS
##  3   2015          4  41500231 2016-04-30         1610612759 San Anton… SAS
##  4   2015          4  41500231 2016-04-30         1610612759 San Anton… SAS
##  5   2015          4  41500231 2016-04-30         1610612759 San Anton… SAS
##  6   2015          4  41500231 2016-04-30         1610612759 San Anton… SAS
##  7   2015          4  41500231 2016-04-30         1610612759 San Anton… SAS
##  8   2015          4  41500231 2016-04-30         1610612759 San Anton… SAS
##  9   2015          4  41500231 2016-04-30         1610612759 San Anton… SAS
## 10   2015          4  41500231 2016-04-30         1610612759 San Anton… SAS
## # ℹ 5,723 more rows
## # ℹ 92 more variables: off_home <dbl>, off_win <dbl>, defensivenbateamid <dbl>,
## #   def_team_name <chr>, def_team <chr>, def_home <dbl>, def_win <dbl>,
## #   fg2made.x <dbl>, fg2missed.x <dbl>, fg2attempted.x <dbl>, fg3made.x <dbl>,
## #   fg3missed.x <dbl>, fg3attempted.x <dbl>, fgmade.x <dbl>, fgmissed.x <dbl>,
## #   fgattempted.x <dbl>, ftmade.x <dbl>, ftmissed.x <dbl>, ftattempted.x <dbl>,
## #   reboffensive.x <dbl>, rebdefensive.x <dbl>, reboundchance <dbl>, …
```

```
#regular season (each team)
reg_sznplus5 <- new_df3 %>%
  group_by(regular_season, team_name) %>%
  distinct(team_name)
reg_sznplus5
```

```
## # A tibble: 33 × 2
## # Groups:   regular_season, team_name [33]
##    regular_season team_name
##             <dbl> <chr>
##  1           2014 Atlanta Hawks
##  2           2014 Golden State Warriors
##  3           2014 LA Clippers
##  4           2014 San Antonio Spurs
##  5           2015 Cleveland Cavaliers
##  6           2015 Golden State Warriors
##  7           2015 Oklahoma City Thunder
##  8           2015 San Antonio Spurs
##  9           2016 Golden State Warriors
## 10           2016 Houston Rockets
## # ℹ 23 more rows
```

```
#playoff years (each team)
playoff_years <-  nextseason_playoffdata %>%
  group_by(season, team_name) %>%
  distinct(team_name)
playoff_years
```

```
## # A tibble: 64 × 2
## # Groups:   season, team_name [64]
##    season team_name
##     <dbl> <chr>
##  1   2015 San Antonio Spurs
##  2   2015 Oklahoma City Thunder
##  3   2015 Golden State Warriors
##  4   2015 Portland Trail Blazers
##  5   2015 Atlanta Hawks
##  6   2015 Cleveland Cavaliers
##  7   2015 Miami Heat
##  8   2015 Toronto Raptors
##  9   2016 Boston Celtics
## 10   2016 Washington Wizards
## # ℹ 54 more rows
```

```
#find out if the +5.0 net rating teams made the second round the year after and if so finding out if their top 5
players played in the playoffs the next year by checking if they are even on the roster of that playoff team
newdf <- reg_sznplus5 %>%
  mutate(season = regular_season + 1) %>%
  left_join(playoff_years, by = c("season", "team_name")) %>%
  filter(season - regular_season == 1) %>%
  left_join(players_who_played_nextyearplayoff, by = c("season", "team_name")) %>%
  left_join(new_df3, by = c("regular_season", "team_name")) %>%
  distinct(player_name.y, season, player_name.x) %>%
  drop_na() %>%
  group_by(season, team_name, player_name.y) %>%
  mutate(is_in_next_year = player_name.y %in% player_name.x) %>%
  select(-player_name.x) %>%
  distinct()
```

```
## Warning in left_join(., new_df3, by = c("regular_season", "team_name")):
## Detected an unexpected many-to-many relationship between `x` and `y`.
```

```
newdf
```

```
## # A tibble: 105 × 5
## # Groups:   season, team_name, player_name.y [105]
##    regular_season team_name             player_name.y   season is_in_next_year
##             <dbl> <chr>                 <chr>            <dbl> <lgl>
##  1           2014 Atlanta Hawks         Kyle Korver       2015 TRUE
##  2           2014 Atlanta Hawks         Paul Millsap      2015 TRUE
##  3           2014 Atlanta Hawks         Al Horford        2015 TRUE
##  4           2014 Atlanta Hawks         Jeff Teague       2015 TRUE
##  5           2014 Atlanta Hawks         DeMarre Carroll   2015 FALSE
##  6           2014 Golden State Warriors Stephen Curry     2015 TRUE
##  7           2014 Golden State Warriors Draymond Green    2015 TRUE
##  8           2014 Golden State Warriors Klay Thompson     2015 TRUE
##  9           2014 Golden State Warriors Harrison Barnes   2015 TRUE
## 10           2014 Golden State Warriors Andre Iguodala    2015 TRUE
## # ℹ 95 more rows
```

```
#showing what percent of each team's top 5 played players for each +5.0 net rating season played in the second ro
und series the following year (assuming they made it that next year)
percent_true_by_group <- newdf %>%
  group_by(regular_season, team_name, season) %>%
  summarize(percent_true = mean(is_in_next_year) * 100)
```

```
## `summarise()` has grouped output by 'regular_season', 'team_name'. You can
## override using the `.groups` argument.
```

```
mean_percent <- mean(percent_true_by_group$percent_true)
```

ANSWER 7: 63.63% , 79.04%

Percent of +5.0 net rating teams making the 2nd round next year: 63.63%
Percent of top 5 minutes played players who played in those 2nd round series: 79.04%

Part 2 – Playoffs Series Modeling

For this part, you will work to fit a model that predicts the winner and the number of games in a playoffs series between any given two teams.

This is an intentionally open ended question, and there are multiple approaches you could take. Here are a few notes and specifications:

1. Your final output must include the probability of each team winning the series. For example: "Team A has a 30% chance to win and team B has a 70% chance." instead of "Team B will win." You must also predict the number of games in the series. This can be probabilistic or a point estimate.

2. You may use any data provided in this project, but please do not bring in any external sources of data.

3. You can only use data available prior to the start of the series. For example, you can't use a team's stats from the 2016-17 season to predict a playoffs series from the 2015-16 season.

4. The best models are explainable and lead to actionable insights around team and roster construction. We're more interested in your thought process and critical thinking than we are in specific modeling techniques. Using smart features is more important than using fancy mathematical machinery.

5. Include, as part of your answer:

- A brief written overview of how your model works, targeted towards a decision maker in the front office without a strong statistical background.
- What you view as the strengths and weaknesses of your model.
- How you'd address the weaknesses if you had more time and/or more data.
- Apply your model to the 2024 NBA playoffs (2023 season) and create a high quality visual (a table, a plot, or a plotly) showing the 16 teams' (that made the first round) chances of advancing to each round.

```
#install.packages("ggplot2", repos = "http://cran.us.r-project.org")
#install.packages("plotly", repos = "http://cran.us.r-project.org")

library(ggplot2)
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
#all of the regular season data for all teams
regular_season_data <- team_data %>% filter(gametype == 2)
regular_season_data
```

```
## # A tibble: 23,958 × 41
##    season gametype nbagameid gamedate   offensivenbateamid off_team_name
##     <dbl>    <dbl>     <dbl> <date>                  <dbl> <chr>
##  1   2016        2  21600495 2016-12-30         1610612740 New Orleans Pelicans
##  2   2016        2  21600495 2016-12-30         1610612752 New York Knicks
##  3   2021        2  22100943 2022-03-03         1610612742 Dallas Mavericks
##  4   2021        2  22100943 2022-03-03         1610612744 Golden State Warriors
##  5   2016        2  21601032 2017-03-18         1610612741 Chicago Bulls
##  6   2016        2  21601032 2017-03-18         1610612762 Utah Jazz
##  7   2021        2  22100942 2022-03-03         1610612761 Toronto Raptors
##  8   2022        2  22200482 2022-12-23         1610612750 Minnesota Timberwolv…
##  9   2016        2  21600494 2016-12-30         1610612738 Boston Celtics
## 10   2016        2  21600494 2016-12-30         1610612748 Miami Heat
## # ℹ 23,948 more rows
## # ℹ 35 more variables: off_team <chr>, off_home <dbl>, off_win <dbl>,
## #   defensivenbateamid <dbl>, def_team_name <chr>, def_team <chr>,
## #   def_home <dbl>, def_win <dbl>, fg2made <dbl>, fg2missed <dbl>,
## #   fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>,
## #   fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>,
## #   ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>, …
```

```
#getting the mean(aver age) of their team stats based on season and team
average_stats <- regular_season_data %>%
  group_by(season, off_team) %>%
  summarise(across(fg2made:shotattemptpoints,  mean, na.rm = TRUE)) %>%
  rename(team = off_team)
```

```
## Warning: There was 1 warning in `summarise()`.
## ℹ In argument: `across(fg2made:shotattemptpoints, mean, na.rm = TRUE)`.
## ℹ In group 1: `season = 2014`, `off_team = "ATL"`.
## Caused by warning:
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.
## Supply arguments directly to `.fns` through an anonymous function instead.
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
##
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'season'. You can override using the
## `.groups` argument.
```

```
average_stats
```

```
## # A tibble: 300 × 29
## # Groups:   season [10]
##    season team  fg2made fg2missed fg2attempted fg3made fg3missed fg3attempted
##     <dbl> <chr>   <dbl>     <dbl>        <dbl>   <dbl>     <dbl>        <dbl>
##  1   2014 ATL      28.1      27.4         55.5    9.98      16.3         26.2
##  2   2014 BKN      30.8      32.2         63.1    6.60      13.3         19.9
##  3   2014 BOS      30.9      32.4         63.3    8.05      16.6         24.6
##  4   2014 CHA      29.5      36.0         65.4    6.07      13.0         19.1
##  5   2014 CHI      28.7      31.9         60.6    7.87      14.4         22.3
##  6   2014 CLE      27.6      27.1         54.7   10.1       17.4         27.5
##  7   2014 DAL      30.8      29.6         60.4    8.93      16.5         25.4
##  8   2014 DEN      29.7      32.8         62.5    8.05      16.7         24.8
##  9   2014 DET      28.5      32.4         60.9    8.57      16.3         24.9
## 10   2014 GSW      30.8      29.2         60     10.8       16.3         27.0
## # ℹ 290 more rows
## # ℹ 21 more variables: fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>,
## #   ftmade <dbl>, ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>,
## #   rebdefensive <dbl>, reboundchance <dbl>, assists <dbl>,
## #   stealsagainst <dbl>, turnovers <dbl>, blocksagainst <dbl>,
## #   defensivefouls <dbl>, offensivefouls <dbl>, shootingfoulsdrawn <dbl>,
## #   possessions <dbl>, points <dbl>, shotattempts <dbl>, andones <dbl>, …
```

```
# Calculate wins and losses for each team based on season and team name
team_records <- regular_season_data %>%
  # Wins for offensive team
  group_by(season, off_team) %>%
  summarise(wins = sum(off_win), losses = sum(def_win)) %>%
  rename(team = off_team) %>%
  # Wins for defensive team
  bind_rows(
    regular_season_data %>%
      group_by(season, def_team) %>%
      summarise(wins = sum(def_win), losses = sum(off_win)) %>%
      rename(team = def_team)
  ) %>%
  # Summarize total wins for each team
  group_by(season, team) %>%
  summarise(total_wins = sum(wins)/2, total_losses = sum(losses)/2)
```

```
## `summarise()` has grouped output by 'season'. You can override using the
## `.groups` argument.
```

```
## `summarise()` has grouped output by 'season'. You can override using the
## `.groups` argument.`summarise()` has grouped output by 'season'. You can
## override using the `.groups` argument.
```

```
# Display the number of wins for each team during the regular season
print(team_records)
```

```
## # A tibble: 300 × 4
## # Groups:   season [10]
##    season team  total_wins total_losses
##     <dbl> <chr>      <dbl>        <dbl>
##  1   2014 ATL           60           22
##  2   2014 BKN           38           44
##  3   2014 BOS           40           42
##  4   2014 CHA           33           49
##  5   2014 CHI           50           32
##  6   2014 CLE           53           29
##  7   2014 DAL           50           32
##  8   2014 DEN           30           52
##  9   2014 DET           32           50
## 10   2014 GSW           67           15
## # ℹ 290 more rows
```

```
#creation of four crucial features (Offensive rating, defensive rating, offensive eFG perent, and defensive eFG p
ercent)

#offensive team rating
ORTG_df_new <- regular_season_data %>%
  group_by(season, off_team, off_team_name) %>%
  rename(team = off_team) %>%
  arrange(season) %>%
  summarise(summed_offpoints = sum(points),
            summed_offpossessions = sum(possessions),
            ORTG = summed_offpoints/(summed_offpossessions/100)) %>%
  rename(team_name = off_team_name)
```

```
## `summarise()` has grouped output by 'season', 'team'. You can override using the
## `.groups` argument.
```

```
ORTG_df_new
```

```
## # A tibble: 300 × 6
## # Groups:   season, team [300]
##    season team  team_name          summed_offpoints summed_offpossessions  ORTG
##     <dbl> <chr> <chr>                         <dbl>                 <dbl> <dbl>
##  1   2014 ATL   Atlanta Hawks                  8409                  7710  109.
##  2   2014 BKN   Brooklyn Nets                  8038                  7711  104.
##  3   2014 BOS   Boston Celtics                 8312                  7937  105.
##  4   2014 CHA   Charlotte Hornets              7721                  7671  101.
##  5   2014 CHI   Chicago Bulls                  8265                  7702  107.
##  6   2014 CLE   Cleveland Cavaliers            8457                  7622  111.
##  7   2014 DAL   Dallas Mavericks               8628                  7903  109.
##  8   2014 DEN   Denver Nuggets                 8320                  7975  104.
##  9   2014 DET   Detroit Pistons                8077                  7668  105.
## 10   2014 GSW   Golden State Warri…            9016                  8081  112.
## # ℹ 290 more rows
```

```
#defensive team rating
DRTG_df_new <- regular_season_data %>%
  group_by(season, def_team, def_team_name) %>%
  rename(team = def_team) %>%
  arrange(season) %>%
  summarise(summed_defpoints = sum(points),
            summed_defpossessions = sum(possessions),
            DRTG = summed_defpoints/(summed_defpossessions/100))%>%
  rename(team_name = def_team_name)
```

```
## `summarise()` has grouped output by 'season', 'team'. You can override using the
## `.groups` argument.
```

```
DRTG_df_new
```

```
## # A tibble: 300 × 6
## # Groups:   season, team [300]
##    season team  team_name          summed_defpoints summed_defpossessions  DRTG
##     <dbl> <chr> <chr>                         <dbl>                 <dbl> <dbl>
##  1   2014 ATL   Atlanta Hawks                  7964                  7728  103.
##  2   2014 BKN   Brooklyn Nets                  8274                  7715  107.
##  3   2014 BOS   Boston Celtics                 8299                  7928  105.
##  4   2014 CHA   Charlotte Hornets              7981                  7681  104.
##  5   2014 CHI   Chicago Bulls                  8019                  7705  104.
##  6   2014 CLE   Cleveland Cavaliers            8090                  7619  106.
##  7   2014 DAL   Dallas Mavericks               8390                  7905  106.
##  8   2014 DEN   Denver Nuggets                 8611                  7960  108.
##  9   2014 DET   Detroit Pistons                8157                  7653  107.
## 10   2014 GSW   Golden State Warri…            8188                  8111  101.
## # ℹ 290 more rows
```

```
#combining all the offensive teams stats from the whole seasons
combined_team_data_off2 <- team_data %>%
  group_by(season, off_team) %>%
  rename(team= off_team) %>%
  summarise(
    total_fgmade = sum(fgmade),
    total_fg3made = sum(fg3made),
    total_fgattempted = sum(fgattempted)
  )
```

```
## `summarise()` has grouped output by 'season'. You can override using the
## `.groups` argument.
```

```
#Offensive eFG column creation
combined_team_data_off2$offensive_eFG_percent = (combined_team_data_off2$total_fgmade + 0.5*combined_team_data_of
f2$total_fg3made) * 100 / combined_team_data_off2$total_fgattempted

#combining all the defensive teams stats from the whole seasons
combined_team_data_def2 <- team_data %>%
  group_by(season, def_team) %>%
  rename(team=def_team) %>%
  summarise(
    total_fgmade_allowed = sum(fgmade),
    total_fg3made_allowed = sum(fg3made),
    total_fgattempted_allowed = sum(fgattempted)
  )
```

```
## `summarise()` has grouped output by 'season'. You can override using the
## `.groups` argument.
```

```
#Defensive eFG column creation
combined_team_data_def2$defensive_eFG_percent = (combined_team_data_def2$total_fgmade_allowed + 0.5*combined_team
_data_def2$total_fg3made_allowed) * 100 / combined_team_data_def2$total_fgattempted_allowed


#install all the necessary packages needed for models
#install.packages("randomForest", dependencies = T)
#install.packages("caret")
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.2
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.1.2
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
#installing glue for string interpolation
#install.packages("glue")
library(glue)

#getting the playoff teams head to head record against each other in the regular season depending on which playof
f year we are on
h2h_record <- regular_season_data %>%
  mutate(series_id = if_else(off_team < def_team,
                             paste(off_team, def_team, sep = "-"),
                             paste(def_team, off_team, sep = "-"))) %>%
  group_by(season, off_team, series_id) %>%
  rename(team = off_team) %>%
  summarize(total_win_againstopponent_regszn = sum(off_win), total_loss_againstopponent_regszn = sum(def_win))
```

```
## `summarise()` has grouped output by 'season', 'team'. You can override using the
## `.groups` argument.
```

```
h2h_record
```

```
## # A tibble: 8,690 × 5
## # Groups:   season, team [300]
##    season team  series_id total_win_againstopponent_reg…¹ total_loss_againstop…²
##     <dbl> <chr> <chr>                                <dbl>                 <dbl>
##  1   2014 ATL   ATL-BKN                                  4                     0
##  2   2014 ATL   ATL-BOS                                  2                     1
##  3   2014 ATL   ATL-CHA                                  2                     2
##  4   2014 ATL   ATL-CHI                                  2                     1
##  5   2014 ATL   ATL-CLE                                  3                     1
##  6   2014 ATL   ATL-DAL                                  2                     0
##  7   2014 ATL   ATL-DEN                                  1                     1
##  8   2014 ATL   ATL-DET                                  3                     1
##  9   2014 ATL   ATL-GSW                                  1                     1
## 10   2014 ATL   ATL-HOU                                  2                     0
## # ℹ 8,680 more rows
## # ℹ abbreviated names: ¹total_win_againstopponent_regszn,
## #   ²total_loss_againstopponent_regszn
```

```
#creation of our datasets and defining multiple features for our model to use (only using past round 1 series ins
tead of other rounds as our eventual goal is to predict round 1 series for 2023)
distinct_round1_seriess <- round1_series %>%
  group_by(season, series_id, off_team, def_team) %>%
  summarise(total_games_in_series = n(), wins_in_series = sum(off_win), games_at_home = sum(off_home, na.rm = TRU
E)) %>%
  rename(team = off_team) %>%
  left_join(team_records, by=c("season", "team")) %>%
  left_join(average_stats, by=c("season", "team")) %>%
  left_join(ORTG_df_new, by=c("season", "team")) %>%
  left_join(DRTG_df_new, by=c("season", "team")) %>%
  left_join(combined_team_data_off2, by = c("season", "team"))%>%
  left_join(h2h_record, by = c("season", "team", "series_id")) %>%
  mutate(NRTG = ORTG - DRTG, winner = ifelse(wins_in_series == 4, TRUE, FALSE))
```

```
## `summarise()` has grouped output by 'season', 'series_id', 'off_team'. You can
## override using the `.groups` argument.
```

```
distinct_round1_seriess
```

```
## # A tibble: 144 × 52
## # Groups:   season, series_id, team [144]
##    season series_id team  def_team total_games_in_series wins_in_series
##     <dbl> <chr>     <chr> <chr>                    <int>          <dbl>
##  1   2014 ATL-BKN   ATL   BKN                          6              4
##  2   2014 ATL-BKN   BKN   ATL                          6              2
##  3   2014 BOS-CLE   BOS   CLE                          4              0
##  4   2014 BOS-CLE   CLE   BOS                          4              4
##  5   2014 CHI-MIL   CHI   MIL                          6              4
##  6   2014 CHI-MIL   MIL   CHI                          6              2
##  7   2014 DAL-HOU   DAL   HOU                          5              1
##  8   2014 DAL-HOU   HOU   DAL                          5              4
##  9   2014 GSW-NOP   GSW   NOP                          4              4
## 10   2014 GSW-NOP   NOP   GSW                          4              0
## # ℹ 134 more rows
## # ℹ 46 more variables: games_at_home <dbl>, total_wins <dbl>,
## #   total_losses <dbl>, fg2made <dbl>, fg2missed <dbl>, fg2attempted <dbl>,
## #   fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>, fgmade <dbl>,
## #   fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>, ftmissed <dbl>,
## #   ftattempted <dbl>, reboffensive <dbl>, rebdefensive <dbl>,
## #   reboundchance <dbl>, assists <dbl>, stealsagainst <dbl>, turnovers <dbl>, …
```

```
#################################################################################################################
#################################################################################################################
########
#Prediction for who the Winner will be in all the previous playoff years before 2023 (the data that is already gi
ven)

#splitting the data into training and testing data
set.seed(123)
trainIndex <- createDataPartition(distinct_round1_seriess$winner, p = .6,
                                  list = FALSE,
                                  times = 1)
trainData <- distinct_round1_seriess[trainIndex, ]
testData <- distinct_round1_seriess[-trainIndex, ]


#Using logistic regression for guessing/predicting the winner (only two classes to predict from)
logitModel <- glm(winner ~ total_wins + total_losses + NRTG + offensive_eFG_percent, data = trainData, family = b
inomial)
summary(logitModel)
```

```
##
## Call:
## glm(formula = winner ~ total_wins + total_losses + NRTG + offensive_eFG_percent,
##     family = binomial, data = trainData)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.98882  -0.62920  -0.02853   0.55805   2.22912
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -16.8389    12.8863  -1.307   0.1913
## total_wins              0.2641     0.1169   2.258   0.0239 *
## total_losses           -0.2643     0.1164  -2.272   0.0231 *
## NRTG                   -0.2621     0.2528  -1.037   0.2997
## offensive_eFG_percent   0.2508     0.1644   1.525   0.1271
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 121.994  on 87  degrees of freedom
## Residual deviance:  67.637  on 83  degrees of freedom
## AIC: 77.637
##
## Number of Fisher Scoring iterations: 6
```

```
#using random forest trees to predict winner
rfmodel <- randomForest(formula = winner ~ total_wins + total_losses + NRTG, data = trainData, ntree = 500, mtry
= 3, importance = TRUE)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
#adding columns to show the predicted probabilities for each team to win the series
testData$predicted_prob_with_lg <- predict(logitModel, newdata = testData, type = "response")
testData$predicted_prob_with_rf <- predict(rfmodel, newdata = testData, interval = "confidence")


#outputting the string showing what chance each team has to win the series in another dataframe
output <- testData %>%
  mutate(opposing_team_probability = 1 - predicted_prob_with_lg,
         final_output = glue("{team} has a {predicted_prob_with_lg*100} chance to win and {def_team} has a {oppos
ing_team_probability* 100} chance."))


#################################################################################################################
#################################################################################################################
########

#Prediction for Amount of Games the Series Goes

#Using random forest (a multi-class classification model)
#splitting the data into training and testing datasets and giving indices
trainIndex2 <- createDataPartition(distinct_round1_seriess$total_games_in_series, p = .8,
                                  list = FALSE,
                                  times = 1)
trainData2 <- distinct_round1_seriess[trainIndex2, ]
testData2 <- distinct_round1_seriess[-trainIndex2, ]

#omitting all NA values from the training data
trainData2 <- na.omit(trainData2)
testData2 <- na.omit(testData2)

#converting all types to numeric if it can be a factor
trainData2 <- trainData2 %>%
  mutate_if(is.factor, as.numeric)
```

```
## `mutate_if()` ignored the following grouping variables:
```

```
testData2 <- testData2 %>%
  mutate_if(is.factor, as.numeric)
```

```
## `mutate_if()` ignored the following grouping variables:
```

```
#the predicted amount of games the series goes to (creation of that column)
rfModelGames <- randomForest(as.factor(total_games_in_series) ~ ., data = trainData2, ntree = 100)
testData2$predicted_num_games <- predict(rfModelGames, newdata = testData2)
confusionMatrix(as.factor(testData2$predicted_num_games), as.factor(testData2$total_games_in_series))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 4 5 6 7
##          4 4 0 0 0
##          5 3 7 4 0
##          6 0 2 4 2
##          7 0 0 0 2
##
## Overall Statistics
##
##                Accuracy : 0.6071
##                  95% CI : (0.4058, 0.785)
##     No Information Rate : 0.3214
##     P-Value [Acc > NIR] : 0.001737
##
##                   Kappa : 0.448
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 4 Class: 5 Class: 6 Class: 7
## Sensitivity            0.5714   0.7778   0.5000  0.50000
## Specificity            1.0000   0.6316   0.8000  1.00000
## Pos Pred Value         1.0000   0.5000   0.5000  1.00000
## Neg Pred Value         0.8750   0.8571   0.8000  0.92308
## Prevalence             0.2500   0.3214   0.2857  0.14286
## Detection Rate         0.1429   0.2500   0.1429  0.07143
## Detection Prevalence   0.1429   0.5000   0.2857  0.07143
## Balanced Accuracy      0.7857   0.7047   0.6500  0.75000
```

```
#around 71 percent accuracy with the predicted number of games using random forest and all of the features in the
dataset


######For the 2023 playoffs first round ONLY(we do not know about future rounds yet)############################
##############################################################################################################
#########################################################################################

#reading in the 2023 first round playoffs csv file
current_season_playoffs<- read_csv("~/Documents/2023season_playoffs.csv")
```

```
## Rows: 16 Columns: 4── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (3): series_id, off_team, def_team
## dbl (1): season
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#joining all the previous dataframes (team records, team offensive rating, team defensive rating, head to head re
cords, combined team data offensively, combined team data defensively) into one bssed on common columns
current_season_playoffs <- current_season_playoffs %>%
  group_by(season, off_team) %>%
  rename(team = off_team) %>%
  left_join(team_records, by=c("season", "team")) %>%
  left_join(average_stats, by=c("season", "team")) %>%
  left_join(ORTG_df_new, by=c("season", "team")) %>%
  left_join(DRTG_df_new, by=c("season", "team")) %>%
  left_join(combined_team_data_off2, by = c("season", "team")) %>%
  left_join(h2h_record, by = c("season", "team", "series_id")) %>%
  mutate(NRTG = ORTG - DRTG)
current_season_playoffs
```

```
## # A tibble: 16 × 48
## # Groups:   season, team [16]
##    season series_id team  def_team total_wins total_losses fg2made fg2missed
##     <dbl> <chr>     <chr> <chr>         <dbl>        <dbl>   <dbl>     <dbl>
##  1   2023 NOP-OKC   NOP   OKC              49           33    30.0      24.8
##  2   2023 NOP-OKC   OKC   NOP              57           25    31.3      23.9
##  3   2023 DEN-LAL   DEN   LAL              57           25    32.3      25.2
##  4   2023 DEN-LAL   LAL   DEN              47           35    31.8      24.3
##  5   2023 MIN-PHX   MIN   PHX              56           26    28.6      23.7
##  6   2023 MIN-PHX   PHX   MIN              49           33    30.0      23.5
##  7   2023 DAL-LAC   DAL   LAC              50           32    28.5      21.6
##  8   2023 DAL-LAC   LAC   DAL              51           31    29.7      23.8
##  9   2023 BOS-MIA   BOS   MIA              64           18    27.4      20.3
## 10   2023 BOS-MIA   MIA   BOS              46           36    27.4      24.5
## 11   2023 NYK-PHI   NYK   PHI              50           32    28.1      24.8
## 12   2023 NYK-PHI   PHI   NYK              47           35    29.4      26.7
## 13   2023 IND-MIL   IND   MIL              47           35    33.8      23.6
## 14   2023 IND-MIL   MIL   IND              49           33    29.0      21.5
## 15   2023 CLE-ORL   CLE   ORL              48           34    28.3      22.1
## 16   2023 CLE-ORL   ORL   CLE              47           35    29.5      24.2
## # ℹ 40 more variables: fg2attempted <dbl>, fg3made <dbl>, fg3missed <dbl>,
## #   fg3attempted <dbl>, fgmade <dbl>, fgmissed <dbl>, fgattempted <dbl>,
## #   ftmade <dbl>, ftmissed <dbl>, ftattempted <dbl>, reboffensive <dbl>,
## #   rebdefensive <dbl>, reboundchance <dbl>, assists <dbl>,
## #   stealsagainst <dbl>, turnovers <dbl>, blocksagainst <dbl>,
## #   defensivefouls <dbl>, offensivefouls <dbl>, shootingfoulsdrawn <dbl>,
## #   possessions <dbl>, points <dbl>, shotattempts <dbl>, andones <dbl>, …
```

```
#predicting probabilities and the winner with the already existing logistic regression model
current_season_playoffs$predicted_prob_with_logr <- predict(logitModel, newdata = current_season_playoffs, type =
"response")
current_season_playoffs <- current_season_playoffs %>%
  group_by(series_id) %>%
  mutate(predicted_winner_with_logr = case_when(is.na(lag(predicted_prob_with_logr)) ~ predicted_prob_with_logr >
lead(predicted_prob_with_logr), TRUE ~ predicted_prob_with_logr > lag(predicted_prob_with_logr)))


#predicting probabilities and the winner with the already made random forest model
current_season_playoffs$predicted_prob_with_rf <- predict(rfmodel, newdata = current_season_playoffs, interval =
"confidence")
current_season_playoffs <- current_season_playoffs %>%
  group_by(series_id) %>%
  mutate(predicted_with_rf = case_when(is.na(lag(predicted_prob_with_rf)) ~ predicted_prob_with_rf > lead(predict
ed_prob_with_rf), TRUE ~ predicted_prob_with_rf > lag(predicted_prob_with_rf)))



###########Observation 1: ###############
#The model starts by reading the current season playoff data and joining it with various historical and statistic
al data sets (team records, offensive and defensive ratings, head-to-head records, and combined team data) based
on common columns such as season, team, and series ID.
#The combined data includes important metrics like Offensive Rating (ORTG), Defensive Rating (DRTG), and Net Rati
ng (NRTG, calculated as ORTG - DRTG).
#The logistic regression model (logitModel) is used to predict the probabilities of each team winning a game base
d on the combined dataset.
#The probabilities are calculated, and a decision is made for each series by comparing the probabilities for each
team.
#Similarly, a random forest model (rfmodel) is used to predict the winning probabilities.
#This model also outputs the predicted probabilities, which are then compared to determine the predicted winner f
or each series.
#For both models, the predicted probabilities are compared within each series to identify the team with the highe
r probability of winning.
#The predictions are then used to label the predicted winner for each series.

###########Observation 2: ################
# My model's strengths were that it had comprehensive data integration, used multiple models, and had great insig
hts. Firstly, I believe that the model leverages a wide array of data accounting for everything, including team r
ecords, advanced metrics, and head-to-head records, providing a holistic view of each team's performance.
#With the usage of multiple models, using both logistic regression and random forest models, the approach combine
s the strengths of both linear and non-linear predictive modeling techniques, potentially increasing the robustne
ss of predictions.
#It also used contextual insights using important metrics like Offensive and Defensive Ratings, and Net Ratings p
rovide context-specific insights that are crucial for predicting game outcomes.
#My model's weaknesses were the following:
#Data Dependency: The accuracy of the model heavily depends on the quality and completeness of the historical dat
a. Missing or inaccurate data can significantly impact predictions.
#Overfitting: The random forest model, while powerful, is prone to overfitting, especially with limited data. Thi
s means it might perform well on historical data but less so on new, unseen data.
#Complexity and Interpretability: The random forest model is a black-box model, making it difficult to interpret
and understand the decision-making process compared to the logistic regression model.

###########Observation 3: ##################
#If given more time and data, I would focus on incorporating more data from additional seasons, including player-
level statistics and more granular in-game data to improve the model’s predictive power. Model tuning is also imp
ortant as I have to perform extensive hyperparameter tuning for the random forest model to minimize overfitting a
nd improve generalization to new data.
#I would have also combined predictions from multiple models (beyond logistic regression and random forest) using
ensemble techniques to potentially improve prediction accuracy.
#If there was a possiblity, I would have also created new features that capture more complex interactions between
players and teams, such as player injuries, in-game strategies, and coaching styles.

########### Observation 4: ###################
p <- ggplot(current_season_playoffs, aes(x = current_season_playoffs$series_id, y = current_season_playoffs$predi
cted_prob_with_logr, group = team, color = team)) +
  geom_line(size = 1) +
  geom_point(size = 3) +
  scale_y_continuous(labels = scales::percent_format()) +
  labs(title = "NBA Playoff Teams' Chances of Advancing to Next Round",
       x = "Playoff Series",
       y = "Probability of Advancing",
       color = "Team") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.

## Warning: Please use `linewidth` instead.
```

```
# Convert ggplot to plotly for interactivity
p <- ggplotly(p)
print(p)
```

Part 3 – Finding Insights from Your Model

Find two teams that had a competitive window of 2 or more consecutive seasons making the playoffs and that under performed your model’s expectations for them, losing series they were expected to win. Why do you think that happened? Classify one of them as bad luck and one of them as relating to a cause not currently accounted for in your model. If given more time and data, how would you use what you found to improve your model?

```r
expected <- testData %>%
  mutate(expected_win = ifelse(predicted_prob_with_lg > 0.5, 1, 0), actual_winner = ifelse(wins_in_series == 4, 1
, 0))

# Identify teams with a competitive window of 2 or more consecutive seasons making the playoffs
competitive_teams <- expected %>%
  group_by(team) %>%
  filter(n_distinct(season) >= 2)
competitive_teams
```

```
## # A tibble: 48 × 56
## # Groups:   team [16]
##    season series_id team  def_team total_games_in_series wins_in_series
##     <dbl> <chr>     <chr> <chr>                    <int>          <dbl>
##  1   2014 ATL-BKN   ATL   BKN                          6              4
##  2   2014 ATL-BKN   BKN   ATL                          6              2
##  3   2014 BOS-CLE   BOS   CLE                          4              0
##  4   2014 BOS-CLE   CLE   BOS                          4              4
##  5   2014 CHI-MIL   MIL   CHI                          6              2
##  6   2015 ATL-BOS   ATL   BOS                          6              4
##  7   2015 CHA-MIA   MIA   CHA                          7              4
##  8   2015 CLE-DET   CLE   DET                          4              4
##  9   2015 GSW-HOU   HOU   GSW                          5              1
## 10   2015 LAC-POR   POR   LAC                          6              4
## # ℹ 38 more rows
## # ℹ 50 more variables: games_at_home <dbl>, total_wins <dbl>,
## #   total_losses <dbl>, fg2made <dbl>, fg2missed <dbl>, fg2attempted <dbl>,
## #   fg3made <dbl>, fg3missed <dbl>, fg3attempted <dbl>, fgmade <dbl>,
## #   fgmissed <dbl>, fgattempted <dbl>, ftmade <dbl>, ftmissed <dbl>,
## #   ftattempted <dbl>, reboffensive <dbl>, rebdefensive <dbl>,
## #   reboundchance <dbl>, assists <dbl>, stealsagainst <dbl>, turnovers <dbl>, …
```

```r
#get the underperfomring teams that were actually expected to win but did not
underperforming_teams <- competitive_teams %>%
  filter(actual_winner == FALSE & expected_win == 1)
```

ANSWER : The two teams that had a competitive window of 2 or more consecutive seasons making the playoffs that underperformed were definitely LA Clippers (2016-17, 2017-18, and 2018-19 seasons) and the Milwaukee Bucks (the 2021-22 season and the 2022-23 season) because according to the features that I used, predicating most of the winning success on team net rating and record, they disappointed and went below expectations. Perhaps, the reason for the Clippers could be that I did not pay attention to player status considering injuries since those players were together for awhile so chemistry should not be a problem. Everyone also always talk about the "Clipper Curse" so it could be bad luck as well. The Utah Jazz, on the other hand, could have had coaching/schematic problems and teaqm chemistry issues as well as fatigue (none of which are factor that I considered in my model). If given more time and data, I would have used those as features, and my model would have had better accuracy perhaps.