Basketball Analytics with R

Created by: Matthew Chandy

Presenter: Addison McGhee

Load Packages

```
suppressPackageStartupMessages({
  library(tidyverse)
  library(hoopR)
  library(BasketballAnalyzeR)
})
```

R Package for Manipulating + Visualizing Data

- tidyverse

```
if (!requireNamespace('tidyverse', quietly = TRUE)){
  install.packages('tidyverse')
}
```

R Package for Acquiring Data

- hoopR

```
if (!requireNamespace('devtools', quietly = TRUE)){
  install.packages('devtools')
}
devtools::install_github("sportsdataverse/sportsdataverse-R")
```

```
## rlang (1.1.5 -> 1.1.6) [CRAN]
```

```
##
## The downloaded binary packages are in
##  /var/folders/rn/yzd7dyjs6dq3pf50qjtn1z9d2qvxmx/T//RtmpYqd8KK/downloaded_packages
## -- R CMD build -------------------------------------------------------------
##      checking for file ‘/private/var/folders/rn/yzd7dyjs6dq3pf50qjtn1z9d2qvxmx/T/RtmpYqd8KK/remotes14
##   -  preparing ‘sportsdataverse’:
##    checking DESCRIPTION meta-information ...  v  checking DESCRIPTION meta-information
##   -  checking for LF line-endings in source and make files and shell scripts
##   -  checking for empty or unneeded directories
```

```
##      Omitted 'LazyData' from DESCRIPTION
##   -  building 'sportsdataverse_0.2.0.tar.gz'
##      Warning: invalid uid value replaced by that for user 'nobody'
##    Warning: invalid gid value replaced by that for user 'nobody'
##
##
```

R Package for Visualizing + Analyzing Basketball Data

- BasketballAnalyzeR

```
devtools::install_github("sndmrc/BasketballAnalyzeR")
```

```
## Skipping install of 'BasketballAnalyzeR' from a github remote, the SHA1 (9e8a01f0) has not changed s
##   Use 'force = TRUE' to force installation
```

Activity 1: Calculating the Four Factors

Our first goal will be to use `hoopR` to retrieve NBA game data. We will then calculate offensive and defensive statistics. These statistics that we will calculate are used to compute the "Four Factors".

Loading Data

We can select the 2023-2024 season using the `nba_leaguegamelog` function.

```
nba_2023 = nba_leaguegamelog(league_id = '00',
                             season = year_to_season(most_recent_nba_season() - 1))
```

However, we actually want the individual game logs, so we will use the `$` operator to select them.

```
nba_2023 = nba_leaguegamelog(league_id = '00',
                             season = year_to_season(most_recent_nba_season() - 1))$LeagueGameLog

# Look at first 6 rows
head(nba_2023)
```

```
## # A tibble: 6 x 29
##   SEASON_ID TEAM_ID  TEAM_ABBREVIATION TEAM_NAME GAME_ID GAME_DATE MATCHUP WL
##   <chr>     <chr>    <chr>            <chr>     <chr>   <chr>     <chr>   <chr>
## 1 22024     1610612~ BOS              Boston C~ 002240~ 2024-10-~ BOS vs~ W
## 2 22024     1610612~ MIN              Minnesot~ 002240~ 2024-10-~ MIN @ ~ L
## 3 22024     1610612~ LAL              Los Ange~ 002240~ 2024-10-~ LAL vs~ W
## 4 22024     1610612~ NYK              New York~ 002240~ 2024-10-~ NYK @ ~ L
## 5 22024     1610612~ LAC              LA Clipp~ 002240~ 2024-10-~ LAC vs~ L
## 6 22024     1610612~ PHX              Phoenix ~ 002240~ 2024-10-~ PHX @ ~ W
## # i 21 more variables: MIN <chr>, FGM <chr>, FGA <chr>, FG_PCT <chr>,
## #   FG3M <chr>, FG3A <chr>, FG3_PCT <chr>, FTM <chr>, FTA <chr>, FT_PCT <chr>,
## #   OREB <chr>, DREB <chr>, REB <chr>, AST <chr>, STL <chr>, BLK <chr>,
## #   TOV <chr>, PF <chr>, PTS <chr>, PLUS_MINUS <chr>, VIDEO_AVAILABLE <chr>
```

We will next use functions from the `tidyverse` to calculate various quantities. Before doing this, we will want to group our data by team. This will make it easier to compare teams later.

```
# Individual Team Data
TEAM <- nba_leaguegamelog(league_id = '00',
                          season = year_to_season(most_recent_nba_season() - 1))$LeagueGameLog %>%
  rename(Team = TEAM_ABBREVIATION) %>% # rename variable for brevity
  group_by(Team) # group teams

# Look at first 6 rows
head(TEAM)
```

```
## # A tibble: 6 x 29
## # Groups:   Team [6]
##   SEASON_ID TEAM_ID  Team  TEAM_NAME GAME_ID GAME_DATE MATCHUP WL    MIN   FGM
##   <chr>     <chr>    <chr> <chr>     <chr>   <chr>     <chr>   <chr> <chr> <chr>
## 1 22024     1610612~ BOS   Boston C~ 002240~ 2024-10-~ BOS vs~ W     240   48
## 2 22024     1610612~ MIN   Minnesot~ 002240~ 2024-10-~ MIN @ ~ L     240   35
## 3 22024     1610612~ LAL   Los Ange~ 002240~ 2024-10-~ LAL vs~ W     240   42
## 4 22024     1610612~ NYK   New York~ 002240~ 2024-10-~ NYK @ ~ L     240   43
## 5 22024     1610612~ LAC   LA Clipp~ 002240~ 2024-10-~ LAC vs~ L     265   42
## 6 22024     1610612~ PHX   Phoenix ~ 002240~ 2024-10-~ PHX @ ~ W     265   38
## # i 19 more variables: FGA <chr>, FG_PCT <chr>, FG3M <chr>, FG3A <chr>,
## #   FG3_PCT <chr>, FTM <chr>, FTA <chr>, FT_PCT <chr>, OREB <chr>, DREB <chr>,
## #   REB <chr>, AST <chr>, STL <chr>, BLK <chr>, TOV <chr>, PF <chr>, PTS <chr>,
## #   PLUS_MINUS <chr>, VIDEO_AVAILABLE <chr>
```

Let's now compute 2pt attempts for each team. We can find the number of 2pt attempts by taking the total number of field goal attempts (`FGA`) and subtracting the number of 3pt attempts (`FG3A`). We will use the `reframe` function to add up all the field goal attempts.

```
# Load 2023 league data and compute basic metrics for offense
TEAM <- nba_leaguegamelog(league_id = '00',
                          season = year_to_season(most_recent_nba_season() - 1))$LeagueGameLog %>%
  rename(Team = TEAM_ABBREVIATION) %>% # rename variable for brevity
  group_by(Team) %>% # group teams
  reframe(P2A = sum(as.integer(FGA) - as.integer(FG3A)))

head(TEAM)
```

```
## # A tibble: 6 x 2
##   Team    P2A
##   <chr> <int>
## 1 ATL    4339
## 2 BKN    3740
## 3 BOS    3338
## 4 CHA    4066
## 5 CHI    3983
## 6 CLE    3942
```

Question: Based on the previous example, how would we compute 2pt shots made?

```
# Your answer here
```

We can use similar code to calculate the rest of the quantities that we need.

```
# Load 2023 league data and compute basic metrics for offense and defense
TEAM <- nba_leaguegamelog(league_id = '00',
                          season = year_to_season(most_recent_nba_season() - 1))$LeagueGameLog %>%
  rename(Team = TEAM_ABBREVIATION) %>%
  group_by(Team) %>%
  reframe(P2A = sum(as.integer(FGA) - as.integer(FG3A)),
          P2M = sum(as.integer(FGM) - as.integer(FG3M)),
          P3A = sum(as.integer(FG3A)),
          P3M = sum(as.integer(FG3M)),
          FTA = sum(as.integer(FTA)),
          FTM = sum(as.integer(FTM)),
          OREB = sum(as.integer(OREB)),
          DREB = sum(as.integer(DREB)),
          TOV = sum(as.integer(TOV)),
          MIN = sum(as.integer(MIN) / 5))

head(TEAM)
```

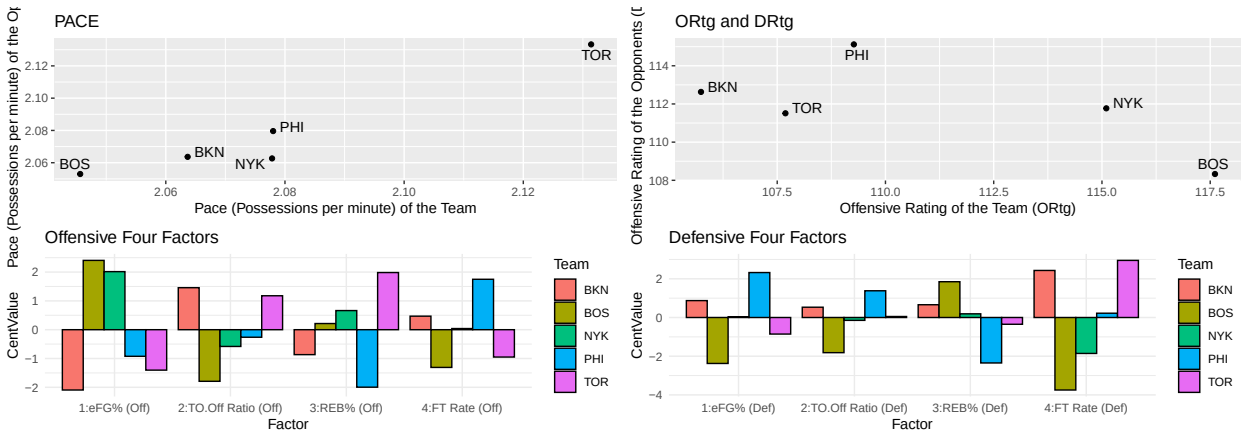
```
## # A tibble: 6 x 11
##   Team    P2A   P2M   P3A   P3M   FTA   FTM  OREB  DREB   TOV   MIN
##   <chr> <int> <int> <int> <int> <int> <int> <int> <int> <int> <dbl>
## 1 ATL    4339  2390  3007  1074  1866  1448   947  2605  1242  3860
## 2 BKN    3740  1926  3156  1089  1654  1304   873  2431  1215  3860
## 3 BOS    3338  1909  3859  1420  1540  1233   908  2707   950  3870
## 4 CHA    4066  2035  3064  1039  1593  1251   973  2634  1237  3855
## 5 CHI    3983  2213  3366  1240  1575  1273   797  2850  1177  3855
## 6 CLE    3942  2296  3310  1274  1741  1359   886  2723  1045  3845
```

We will next calculate the above quantities from the opponent's perspective. This effectively tells us about a team's defensive performance. The key difference is that we will use the command `separate(MATCHUP, c("OPP", "vs", "Team"), " ")` to look at their opponent's scoring.

```
# Load data for
OPP <- nba_leaguegamelog(league_id = '00',
                         season = year_to_season(most_recent_nba_season() - 1))$LeagueGameLog %>%
  separate(MATCHUP, c("OPP", "vs", "Team"), " ") %>%
  group_by(Team) %>%
  reframe(P2A = sum(as.integer(FGA) - as.integer(FG3A)),
          P2M = sum(as.integer(FGM) - as.integer(FG3M)),
          P3A = sum(as.integer(FG3A)), P3M = sum(as.integer(FG3M)),
          FTA = sum(as.integer(FTA)), FTM = sum(as.integer(FTM)),
          OREB = sum(as.integer(OREB)), DREB = sum(as.integer(DREB)),
          TOV = sum(as.integer(TOV)),
          MIN = sum(as.integer(MIN) / 5))
```

Once we compute the Four Factors for offense and defense, we can use the `fourfactors` command and `plot` to visualize the factors. We will also reduce the number of teams to make visulization easier to see

```
# Select teams (Brooklyn, Boston, New York Knicks, Philadelphia, Toronto)
selTeams <- c(2, 3, 20, 23, 28)

# Compute four factors
out <- fourfactors(TEAM[selTeams,], OPP[selTeams,])
plot(out)
```



Creating Shot Charts

Like with the previous example, we will use `hoopR` to pull NBA data. Creating a shot chart will require us to have play-by-play (pbp) data, so we will use the command `load_nba_pbp` to look at the most recent season.

```
# Load play-by-play (pbp) data from the Boston Celtics
pbp <- load_nba_pbp(seasons = most_recent_nba_season())
```

We can load basic player information using the `load_nba_player_box` command. We can then look at a specific team like the Boston Celtics by using `filter`. Our first task will be to grab the names of players. We need to do this because the play-by-play data doesn't have player names, only player ID numbers.

```
player_names <- load_nba_player_box(seasons = most_recent_nba_season()) %>%
  filter(team_location == "Boston", ) %>%
  group_by(athlete_id) %>%
  summarize(athlete_id = first(athlete_id),
            athlete_display_name = first(athlete_display_name))
```

Now that we have the actual names, we can filter the play-by-play data to look at Boston.

```
bos_id <- first(pbp %>%
                  filter(home_team_abbrev == 'BOS') %>%
                  select(home_team_id)) %>%
  as.numeric()

# Filter to Boston data
bos_pbp <- pbp %>% filter(team_id == bos_id)
```

Since we want to look at a shot chart, we will filter to shooting plays. We will also filter to remove any shots containing the phrase "Free Throw", as we want to only consider shots from the field. Finally, we will compute a few important quantities, mainly the (x, y) coordinate of each shot, the points scored for each shot, the shot result (make or miss), and the distance from the basket (Distance = $\sqrt{x^2 + y^2}$ by the Pythagorean Theorem). Note that the x-coordinate needs to be shifted by 25 feet due to quirks with the basketball court.

```
# Filter to shooting plays and remove free throws
bos_shots <- bos_pbp %>%
  filter(shooting_play == TRUE) %>%
  filter(!grepl("Free Throw", type_text)) %>%
  mutate(x = abs(coordinate_x_raw - 25), # Find (x, y) position of shot
         y = coordinate_y_raw,
         points = score_value,
         event_type = if_else(scoring_play, "shot", "miss"), # filter shot outcome
         shot_distance = as.integer(sqrt(x**2 + y**2))) # Use Pythagorean Theorem
```

Now that the play-by-play data is ready, we can join with the player names data. We will also filter the data to only include shots within 35 feet of the basket.

```
bos_shots <- left_join(bos_shots, player_names,  # Join datasets
                       by = join_by(athlete_id_1 == athlete_id)) %>%
  rename(player = athlete_display_name)

# Find shots within 35 foot arc
bos_shots_subset <- subset(bos_shots, shot_distance < 35)
```

From here, we can filter the data to look at a specific Celtic. Let's look at Jayson Tatum.

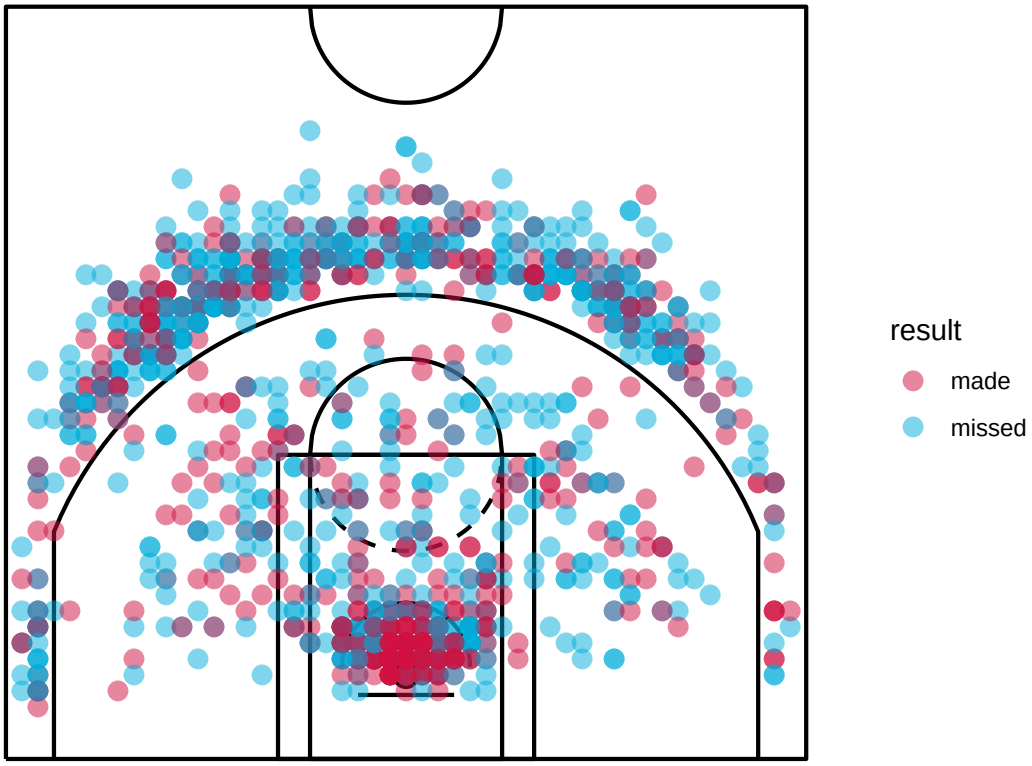
```
# Filter to Tatum
tatum_shots_subset <- bos_shots_subset %>%
  filter(player == "Jayson Tatum") %>%
  mutate(coordinate_x_adj = coordinate_x_raw - 25,
         coordinate_y_adj = coordinate_y_raw - 41.75,
         result = as.factor(if_else(event_type == "shot", "made", "missed")))

head(tatum_shots_subset)
```

```
## -- ESPN NBA Play-by-Play from hoopR data repository ------------- hoopR 2.1.0 --

## i Data updated: 2025-04-11 08:10:58 EDT

## # A tibble: 6 x 71
##   game_play_number         id sequence_number type_id type_text text  away_score
##              <int>      <dbl>           <int>   <int> <chr>     <chr>      <int>
## 1               13    4.02e10              23      92 Jump Shot Jays~          5
## 2               31    4.02e10              47     132 Step Bac~ Jays~          5
## 3               40    4.02e10              57      92 Jump Shot Jays~          7
## 4               62    4.02e10              89     121 Fade Awa~ Jays~         14
## 5               64    4.02e10              91     132 Step Bac~ Jays~         14
## 6               76    4.02e11             107     129 Running ~ Jays~         19
## # i 64 more variables: home_score <int>, period_number <int>,
```

```
## #   period_display_value <chr>, clock_display_value <chr>, scoring_play <lgl>,
## #   score_value <int>, team_id <int>, athlete_id_1 <int>, athlete_id_2 <int>,
## #   athlete_id_3 <int>, wallclock <chr>, shooting_play <lgl>,
## #   coordinate_x_raw <dbl>, coordinate_y_raw <dbl>, game_id <int>,
## #   season <int>, season_type <int>, home_team_id <int>, home_team_name <chr>,
## #   home_team_mascot <chr>, home_team_abbrev <chr>, ...
```

Finally, we will use the `shotchart` command to make the plot.

```
# Create shot chart
shotchart(data = data.frame(tatum_shots_subset), x = "coordinate_x_adj",
          y = "coordinate_y_adj", scatter=TRUE, z = "result")
```



Creating an Assist Network

Let's use the play-by-play data from the previous exercise again. This time, we will filter to look at plays involving assists. Not surprisingly, we will need this information to make an "assist" network.

```
# Filter assist plays
bos_assists <- bos_pbp %>% filter(grepl("assist", text))
```

Any given assist will involve at least two players. Hence, we will count assists for both players involved. We can use the `count` function to tally assists.

```
# Tally assists
assistee_count <- bos_assists %>%
  count(athlete_id_1) # player who assisted

assister_count <- bos_assists %>%
  count(athlete_id_2) # player that made the shot
```

We will next consider the 8 Boston players with the most assists.

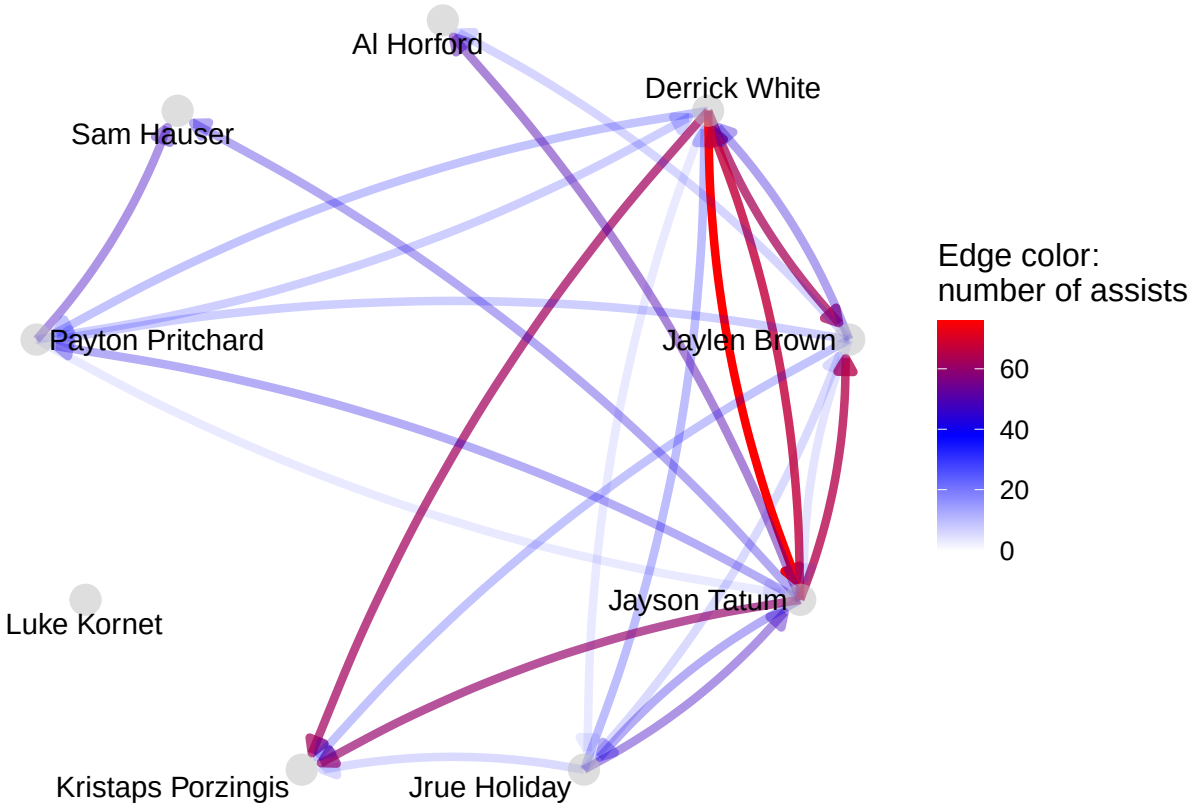
```
# Find players with most assists
bos_assists <- bos_assists %>%
  filter(athlete_id_1 %in%
           (assistee_count %>% arrange(desc(n)) %>% slice(1:8) %>%
           select(athlete_id_1))[[1]],
         athlete_id_2 %in%
           (assister_count %>% arrange(desc(n)) %>% slice(1:8) %>%
           select(athlete_id_2))[[1]])
```

We can then join the datasets to have the data together in one place.

```
# Join data
bos_assists <- left_join(bos_assists, player_names, by = join_by(athlete_id_1 == athlete_id))
bos_assists <- left_join(bos_assists, player_names, by = join_by(athlete_id_2 == athlete_id))
```

Finally, we can use the `assistnet` and `plot` commands to make our assist network.

```
# Create assist network chart
out <- assistnet(bos_assists,
                 assist = "athlete_display_name.y",
                 player = "athlete_display_name.x",
                 points = "score_value",
                 event.type = "type_text")

plot(out, layout="circle", edge.thr=30)
```



Introduction to Expected Points: Tatum's Duce

Jayson Tatum's 2PFG% this season is 54%. What is his expected points on a 2-point field goal attempt?

1.08

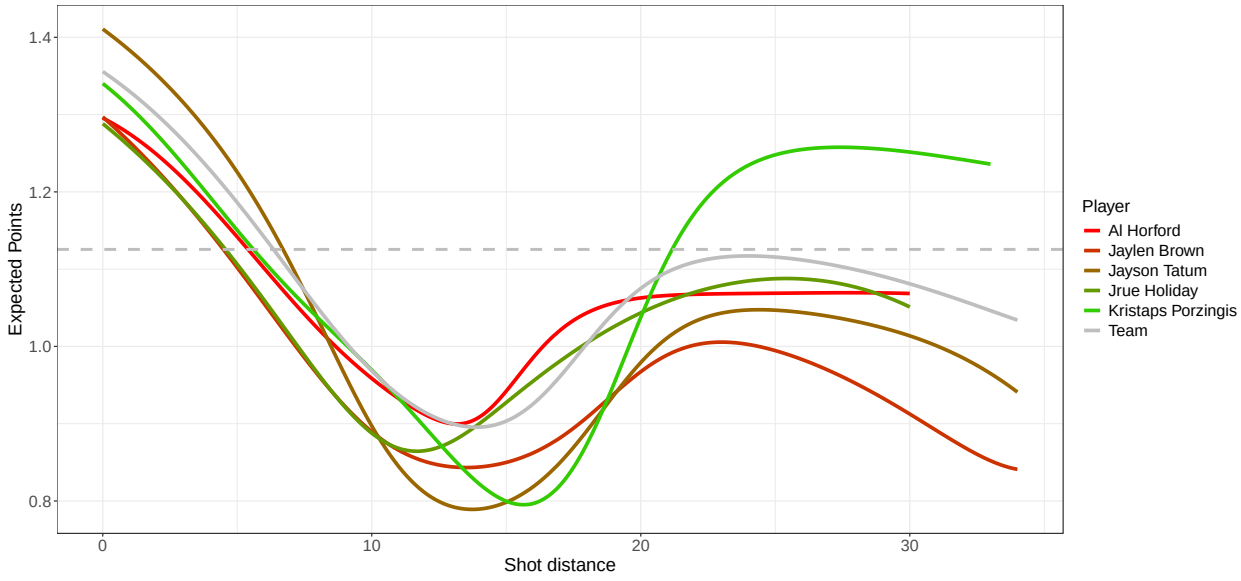
Question: How Do Expected Points Change Based on Distance from the Basket?

(Conditional) Expected Points for the Celtics' Starters

We will now consider how the expected points for a player changes based on the distance from the basket. The `expectedpts` command also allows us to consider other variables like play length and total time remaining in the game. Note that we will be using the same data set from before (`bos_shots_subset`).

```
pl <- c("Al Horford", "Jaylen Brown", "Jayson Tatum", "Jrue Holiday",
        "Kristaps Porzingis")

# Set color palette to be mix of red and greens
mypal <- colorRampPalette(c("red", "green"))

expectedpts(data = data.frame(bos_shots_subset), players = pl, bw = 10,
            var = "shot_distance", col.team = "gray", palette = mypal,
            col.hline = "gray") +
  theme(axis.title = element_text(size = 16),
```

```
        axis.text = element_text(size = 14),
        legend.text = element_text(size = 14),
        legend.title = element_text(size = 15))
```



Cluster Analysis

Now that we've done the previous examples together, look at the code below and try to explain what each line is doing. Do you see commands or code patterns from before?

```
# This code is for the "bubble plot", which is useful for viewing clusters

# Get team information from the previous year (2023)
TEAM <- nba_leaguegamelog(league_id = '00', season = year_to_season(
  most_recent_nba_season() - 1))$LeagueGameLog %>%
  rename(Team = TEAM_ABBREVIATION) %>%
  filter(!is.na(WL)) %>%
  mutate(WINS = ifelse(WL == "W", 1, 0)) %>%
  group_by(Team) %>%
  reframe(TEAM_PTS = sum(as.integer(PTS)), WINS = sum(WINS))

# Separate information by team and opponent
OPP <- nba_leaguegamelog(league_id = '00', season = year_to_season(
  most_recent_nba_season() - 1))$LeagueGameLog %>%
  separate(MATCHUP, c("OPP", "vs", "Team"), " ") %>%
  group_by(Team) %>%
  reframe(OPP_PTS = sum(as.integer(PTS)))

# Merge data
df <- merge(TEAM, OPP, by = "Team")

# Perferm k-group clustering
kclu2 <- kclustering(df[,-1], labels = df$Team, k=5)
cluster <- as.factor(kclu2$Subjects$Cluster)
```

```r
# Save to data to be put in Bubble Plot
Xbubble <- data.frame(Team = df$Team,
                      TEAM_PTS = df$TEAM_PTS,
                      OPP_PTS = df$OPP_PTS, cluster,
                      WINS = df$WINS
                      )
labs <- c("TEAM_PTS", "OPP_PTS", "cluster", "WINS")

# Create bubble plot
bubbleplot(Xbubble, id="Team", x="TEAM_PTS", y="OPP_PTS",
           col="cluster", size = "WINS", labels=labs)
```