R Code

2024-03-27

```r
# HELPER FUNCTIONS
################################################################################
# BALL
get_ball <- function(data) {
  return(data.frame(x = data[3], y=data[4]))
}

# PLAYERS
get_players <- function(data) {
  return(data.frame(player_id=data[2:nrow(data),2], x = data[2:nrow(data),3], y=data[2:nrow(data),4]))
}

# PLAYERS
get_players_info <- function(json_data) {
  home_players <- json_data$events$home$players[[1]]
  visitor_players <- json_data$events$visitor$players[[1]]
  home_players$team <- "home"
  visitor_players$team <- "visitor"

  return(rbind(home_players, visitor_players))
}

# MOMENT INFO
get_moment_info <- function(quarter, id_play, moment, ball_positions, players_positions) {
  clock <- if (is.null(moment[[4]])) { 0 } else { moment[[4]] }
  metadata <- data.frame(quarter = quarter, play_id = id_play,  moment_id = as.character(moment[[2]]),

  ball_positions <- rbind(ball_positions,
                          cbind( metadata, get_ball(moment[[6]][1,]))
  )
  players_positions <- rbind(players_positions,
                             cbind(metadata, get_players(moment[[6]]))
  )

  return(list(ball_positions, players_positions))
}
```

```r
# LOAD DATA
################################################################################

# Load Courts data
full_court <- st_read(file.path(DATA_DIR, "nba-court-lines-05feb2024.gpkg"),
                      layer = "nba-court-lines-05feb2024")
half_court <- full_court[-1, ]
```

```r
# Create the basketball court plot
court_plot <- ggplot() +
  geom_sf(data = half_court, color = "black", fill = "transparent",
          linewidth= 1)
court_plot
```

```r
# Load shots data
shots_data <- data.table()
for (year in 2004:2019) {

  file_name <- sprintf("NBA_%d_Shots.csv", year)
  year_data <- fread(file.path(DATA_DIR, file_name))
  shots_data <- rbindlist(list(shots_data, year_data), use.names = TRUE, fill = TRUE)

}

# Make SF point object
shots_data_sf <- st_as_sf(shots_data, coords = c("LOC_X", "LOC_Y"), crs = st_crs(half_court))

# Identify shots in the restricted area
###############################################################################

# The restricted area is currently just a semi circle, so we need to connect the
# end points in order to make it a polygon
r_a <- half_court %>%
  filter(Feature=="Restricted area")

# Create polygon from coordinates
r_a_coords <- st_coordinates(r_a) # Extract coordinates
r_a_coords <- rbind(r_a_coords, r_a_coords[1,]) # Make first and last the same


r_a_polygon <- st_polygon(list(r_a_coords)) # Make polygon
r_a_polygon <- st_zm(r_a_polygon) # Only keep x and y coords
r_a_polygon <- st_make_valid(r_a_polygon)
r_a_polygon_sf <- st_sf(geometry = st_sfc(r_a_polygon, crs = st_crs(r_a))) # make sf object

# Identify shots inside restricted area using st_within
inside_r_a <- st_within(shots_data_sf, r_a_polygon_sf)

# Create_indicator
shots_data_sf$inside_r_a <- as.integer(lengths(inside_r_a) > 0)
shots_data$inside_r_a <- as.integer(lengths(inside_r_a) > 0)


# Identify shots just behind the 3 point line
###############################################################################
# We need to make the 3 point line into a polygon to identify shots behind it
tpl <- half_court %>%
  filter(Feature=="3-point line")

# Create polygon from coordinates
tpl_coords <- st_coordinates(tpl) # Extract coordinates
tpl_coords <- rbind(tpl_coords, tpl_coords[1,]) # Make first and last point the same
```

```r
tpl_polygon <- st_polygon(list(tpl_coords)) # Make polygon
tpl_polygon <- st_zm(tpl_polygon) # Only keep x and y coords
tpl_polygon <- st_make_valid(tpl_polygon)
tpl_polygon_sf <- st_sf(geometry = st_sfc(tpl_polygon, crs = st_crs(tpl))) # make sf object

# Identify 2 pointers
two_pointer <- st_within(shots_data_sf, tpl_polygon_sf)

# Create_indicator
shots_data_sf$two_pointer <- as.integer(lengths(two_pointer) > 0)
shots_data$two_pointer <- as.integer(lengths(two_pointer) > 0)

# Calculate distance to 3 point line
tpl_distance <- as.numeric(st_distance(shots_data_sf, tpl)/100000)

# Create indicator if shot closer than 3ft to 3 point line
shots_data$tpl_shot <- as.integer(tpl_distance <= 3)
shots_data_sf$tpl_shot <- as.integer(tpl_distance <= 3)

# Set indicator to 0 if shot was a 2 pointer
shots_data <- shots_data %>%
  mutate(tpl_shot = ifelse(two_pointer == 1, 0, tpl_shot))

shots_data_sf <- shots_data_sf %>%
  mutate(tpl_shot = ifelse(two_pointer == 1, 0, tpl_shot))


# FIGURE 2.1: Court Scoring Areas
################################################################################

# Make outer polygon of the court
outer_polygon  <- list(rbind(c(-25, 0), c(-25, 47), c(25,47), c(25, 0), c(-25, 0)))
gm.outer_polygon  <- st_polygon(outer_polygon)
gm.outer_polygon <- st_sfc(gm.outer_polygon, crs = st_crs(tpl)) # Give crs

# Plot scoring regions
ggplot() +
  geom_sf(data = gm.outer_polygon, aes(fill = "3 points"),
          color = "black", linewidth = 1) +
  geom_sf(data = tpl_polygon_sf, aes(fill = "2 points"),
          color = "black", linewidth = 1) +
  geom_sf(data = half_court, color = "black", linewidth = 1) +
  scale_fill_manual(name = "",
                    values = c("3 points" = "deepskyblue3",
                               "2 points" = "indianred")) +
  theme(plot.title = element_text(hjust = 0.5, size = 20),
        legend.position = "right",
        legend.text = element_text(size = 12),
        legend.key.size = unit(1.5, "cm"),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank())
```

```r
# FIGURE 3.1: Kyrie Irving 2 point shoot GSE vs CLE 2016-01-18
################################################################################

# Read file
json_data <- fromJSON(file.path(DATA_DIR, "0021500622.json"))

# PLAYERS DATA
players <- get_players_info(json_data)

# PLAYS DATA
plays <- json_data$events$moments

# BALL + PLAYERS INIT
  ball_positions <- data.frame(quarter=NULL, play_id=NULL, moment_id=NULL,
                               time=NULL, clock=NULL, x=NULL, y=NULL)
  players_positions <- data.frame(quarter=NULL, play_id=NULL, moment_id=NULL,
                          time=NULL, clock=NULL, player_id=NULL, x=NULL, y=NULL)

  # progress_bar
  total_iterations <- length(plays)

  # Initialize progress bar
  pb <- progress_bar$new(
    format = "[:bar] :percent ETA: :eta",
    total = total_iterations
  )
  for (id_play in seq(length(plays))) {
    moments <- plays[[id_play]]

    # Add stop
    if (id_play > 20) {
      break
    }

    if (length(moments) > 0) {
      print(id_play)
      for (id_moment in seq(length(moments))) {

        moment <- plays[[id_play]][[id_moment]]

        if (nrow(moment[[6]]) > 1) {
          res <- get_moment_info(moment[[1]], id_play, moment,
                                 ball_positions, players_positions)
          ball_positions <- res[[1]]
          players_positions <- res[[2]]
        }
      }
    }
    pb$tick()
  }

  # Close progress bar
  #pb$close()

  play_id_ <- unique(ball_positions$play_id)[3]
  play_ball <- ball_positions[ball_positions$play_id == play_id_,]
  play_ball$group <- cumsum(c(0, diff(play_ball$x) < 0))

  # Create the initial plot with points
  p <- ggplot() +
    geom_point(data=play_ball, aes(x=x, y=y-25, group=group), color="orange", size=2)

  play_players <- players_positions[players_positions$play_id == play_id_,]
  players_names <- NULL
  # Loop through unique player IDs
  for (player_id_ in unique(play_players$player_id)) {
    print(player_id_)
    # Get team of the player
    team <- players[players$playerid == player_id_, ]$team

    # Get player name
    player_name <- paste(players[players$playerid == player_id_, ]$firstname, players[players$playerid
    print(player_name)

    # Create a linestring for the player's trajectory
    player_line <- play_players[play_players$player_id == player_id_,]
    # Add a group identifier for each set of points
    player_line$group <- cumsum(c(0, diff(player_line$x) < 0))

    # Determine color based on team
    color <- ifelse(team == 'visitor', '#6F83FF', '#FF7E6F')
    size <- 0.5
    if (player_name == "Kyrie Irving") {
      color = 'brown2'
      size <- 1
    }

    add <- player_line[as.integer(nrow(player_line)*1.0), ]
    add$player_name <- player_name
    add$color <- color

    if (is.null(players_names)) {
      players_names <- add
    } else {
      players_names <- rbind(players_names, add)
    }

    # Add points for player paths
    p <- p + geom_point(data = player_line, aes(x = x, y = y-25, group = group), color = color, size =
  }
```

```
  full_court <- st_read(file.path(DATA_DIR, "nba-court-lines-05feb2024.gpkg"), layer = "nba-court-lines
  full_court_ <- st_geometry(full_court) * matrix(c(0, 1, -1, 0), ncol = 2)

  p2 <- p +
    geom_sf(data = full_court_, color = "black", fill = "transparent", linewidth= 1) +
    theme_void() +
    geom_text(data = players_names, aes(x = x, y = y- 25, label = player_name),
              color=players_names$color, vjust = -0.9, hjust = 0.7, size=5) +
    ggtitle("Golden State Warrios vs Cleveland Cavaliers 2016-01-18 \nKyrie Irving 2 points") +
    theme(plot.title = element_text(hjust = 0.5, size = 16, face = "bold"))

  p2
```

```
# FIGURE 4.1: Evolution of Attempted Shots by the 3 point line
################################################################################
# Filter the dataset to exclude shots taken from inside the restricted area
tpl_shots_summary <- shots_data %>%
  filter(inside_r_a != 1) %>%
  group_by(SEASON_1, tpl_shot) %>%
  summarise(Count = n(), .groups = 'drop') %>%
  group_by(SEASON_1) %>%
  mutate(Total = sum(Count),
         Percentage = (Count / Total) * 100) %>%
  ungroup()

# Create a bar plot visualizing the percentage of three-point shot attempts by season
ggplot(tpl_shots_summary,
       aes(x = SEASON_1, y = Percentage, fill = as.factor(tpl_shot))) +
  geom_bar(stat = "identity", position = "fill") +
  scale_y_continuous(labels = scales::percent_format()) +
  scale_fill_manual(name = "", values = c("#293352", "#4E84C4"),
                    labels = c('Other', "3 Point Shot")) +
  labs(title = "",
       x = "Season",
       y = "Percentage",
       fill = NULL) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
# FIGURE 4.2: Shots attempts comparison between 2004 and 2019
################################################################################
# Shots in 2004
outside_before <- shots_data %>%
  filter(SEASON_1 == 2004) %>%
  filter(inside_r_a != 1)

# Shots in 2019
outside_now <- shots_data %>%
  filter(SEASON_1 == 2019) %>%
  filter(inside_r_a != 1)

# Shots before heat map
before <- ggplot() +
  geom_sf(data = half_court, color = "black",
```

```r
          fill = "transparent", linewidth = 1) +
  geom_density_2d_filled(outside_before,
                         mapping = aes(x = LOC_X, y = LOC_Y, fill = ..level..),
                         contour_var = "ndensity",
                         breaks = seq(0.1, 1.0, length.out = 50), alpha = .8) +
  scale_x_continuous(limits = c(-27.5, 27.5)) +
  scale_y_continuous(limits = c(0, 45)) +
  labs(title = 'Shots attempts 2004') +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5, size = 20),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank())

# Shots now heat map
now <- ggplot() +
  geom_sf(data = half_court, color = "black",
          fill = "transparent", linewidth = 1) +
  geom_density_2d_filled(outside_now, mapping =
                           aes(x = LOC_X, y = LOC_Y, fill = ..level..),
                         contour_var = "ndensity",
                         breaks = seq(0.1, 1.0, length.out = 50), alpha = .8) +
  scale_x_continuous(limits = c(-27.5, 27.5)) +
  scale_y_continuous(limits = c(0, 45)) +
  labs(title = 'Shots attempts 2019') +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5, size = 20),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank())

# Add plots side by side
side_by_side_plots <- grid.arrange(before, now, ncol = 2)
```

```r
print(side_by_side_plots)


# FIGURE 4.3: MVPs shots heatmap
################################################################################
# Define a function to create the plot for a player
create_player_plot <- function(shots_data, year, player_name) {

  # filter data by player and only keen shots made
  data <- shots_data %>%
      filter(SEASON_1 == year) %>%
      filter(SHOT_MADE == TRUE) %>%
      filter(str_detect(PLAYER_NAME, player_name))

  title <- paste0(player_name, ' - ', year)

  # Make heat map
  ggplot() +
    geom_sf(data = half_court, color = "black",
            fill = "transparent", linewidth = 1) +
    geom_point(data = data, aes(x = LOC_X, y = LOC_Y),
               size = 1, alpha = 0.1) +
    geom_density_2d_filled(data,
                           mapping = aes(x = LOC_X, y = LOC_Y, fill = ..level..),
                           contour_var = "ndensity",
                           breaks = seq(0.03, 1.0, length.out = 80), alpha = .7) +
    scale_x_continuous(limits = c(-27.5, 27.5)) +
    scale_y_continuous(limits = c(0, 45)) +
    labs(title = title) +
    theme(legend.position = "none",
          plot.title = element_text(hjust = 0.5, size = 20),
          axis.title.x = element_blank(),
          axis.title.y = element_blank(),
          axis.text.x = element_blank(),
          axis.text.y = element_blank(),
          axis.ticks = element_blank(),
          panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          panel.background = element_blank())
}

# Create plots for each player
player1 <- create_player_plot(shots_data, 2013, "LeBron James")
player2 <- create_player_plot(shots_data, 2016, "Stephen Curry")
player3 <- create_player_plot(shots_data, 2019, "Giannis Antetokounmpo")

# Arrange plots side by side
side_by_side_plots <- grid.arrange(player1, player2, player3, ncol = 3)
```

```r
print(side_by_side_plots)

# FIGURE 4.4: SHOT SUCESS RATE BY DISTANCE
################################################################################
# Extract the hoop location
hoop_location <- half_court %>%
  filter(Feature == "Hoop") %>%
  st_geometry()

# Calculate distance to hoop
shot_distances <- st_distance(shots_data_sf, hoop_location)
shots_data_sf$shot_distances <- as.numeric(shot_distances/100000)

# Calculate points per shot
shots_data_sf$points <- 3 # 3 pointers
```

```r
shots_data_sf <- shots_data_sf %>%
  mutate(points= ifelse(two_pointer == 1, 2, points)) # 2 pointers
shots_data_sf <- shots_data_sf %>%
  mutate(points= ifelse(SHOT_MADE == FALSE, 0, points)) # missed shots

# Keep data relevent for the plot
shots_data_sf_plot <- shots_data_sf %>%
  filter(shot_distances <= 50, SEASON_1 %in% c(2004, 2019))

# Plot
ggplot(data = shots_data_sf_plot, aes(x = shot_distances, y = as.numeric(points),
                                      colour = as.factor(SEASON_1))) +
  geom_smooth(se = FALSE, method = "gam", formula = y ~ s(x, bs = "cs")) +
  scale_y_continuous(breaks = seq(from = 0, to = 2, by = 0.5)) +
  scale_color_manual(values = c("2004" = "deepskyblue3", "2019" = "indianred")) +
  theme_classic() +
  labs(
    x = "Shot Distance (feet)",
    y = "Expected points",
    colour = "Season"
  ) +
  geom_vline(xintercept = 24, linetype = "dashed", color = "black") +
  annotate("text", x = 30, y = 1.5, label = "3 point line", vjust = -0.5)
```

```r
# FIGURE 4.5: Stephen Curry shots and shooting percentage in 2016
################################################################################
# Keep steph currys 2016 season
shots_curry <- shots_data %>%
  filter(SEASON_1 == 2016) %>%
  filter(str_detect(PLAYER_NAME, "Stephen Curry")) %>%
  select(PLAYER_NAME, GAME_DATE, EVENT_TYPE, SHOT_MADE, SHOT_TYPE, LOC_X, LOC_Y)

shots_curry

# Create scatter plot
scatter_plot <- ggplot() +
  geom_sf(data = half_court, color = "black",
          fill = "transparent", linewidth = 1) +
  geom_point(data = shots_curry,
             aes(x = LOC_X, y = LOC_Y, # Add points with transparency
                                   color = SHOT_MADE), alpha = 0.3) +
  labs(title = "Shots Scatterplot"
     , x = "LOC_X", y = "LOC_Y") +  # Add title and axis labels
  theme_minimal() +
  scale_x_continuous(limits = c(-27.5, 27.5)) +
  scale_y_continuous(limits = c(0, 45)) +
  labs(title = 'Stephen Curry 2016 shots') +
  theme(legend.position = "bottom",
        plot.title = element_text(hjust = 0.5, size = 20),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank())

# Print the scatter plot
print(scatter_plot)
```

```r
shots_data$LOC_X_simple <- as.integer(shots_data$LOC_X/4)*4
shots_data$LOC_Y_simple <- as.integer(shots_data$LOC_Y/4)*4

shot_summary <- shots_data %>%
  group_by(LOC_X_simple, LOC_Y_simple) %>%
  summarize(total_shots = n(), made_shots = sum(SHOT_MADE))

# Calculate the acceptance rate
shot_summary$acceptance_rate <- shot_summary$made_shots / shot_summary$total_shots * 100

# Create the heatmap plot
heatmap_plot <- ggplot() +
  geom_sf(data = half_court, color = "grey", fill = "transparent",
          linewidth = 1, alpha=0.8) +  # Plot the basketball court
  geom_raster(data = shot_summary, aes(x = LOC_X_simple, y = LOC_Y_simple,
                                       fill = acceptance_rate), alpha=0.5) +
  scale_fill_gradient(low = "red", high = "darkblue",
                      name = "Acceptance Rate") +  # Customize fill color scale
  scale_x_continuous(limits = c(-27.5, 27.5)) +  # Set x-axis limits
  scale_y_continuous(limits = c(0, 45)) +  # Set y-axis limits
  labs(title = "Shooting Percentage Heatmap") +  # Add title
  theme_minimal() +
  theme(legend.position = "bottom",
        plot.title = element_text(hjust = 0.5, size = 20),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank())

# Print the heatmap plot
print(heatmap_plot)
```

```r
side_by_side_plots <- grid.arrange(scatter_plot, heatmap_plot, ncol = 2)
```

```r
print(side_by_side_plots)
```