DSAC: Data visualization walkthrough

Using ggplot2

Jack Lichtenstein

2022-02-20

Loading libraries

We are going to be working with R through the tidyverse!

```
library(tidyverse)
theme_set(theme_light()) # setting a theme for ggplot2
```

Load data

The data we are going to be working with comes from the `gamezoneR` package. The package can be used to load in play-by-play data of men's college basketball games, all with charted shot locations.

```
# If gamezoneR is not installed, install
if (!require("gamezoneR")) {
  devtools::install_github(repo = "JackLich10/gamezoneR")
}

# Load in play-by-play data from this season
pbp <- gamezoneR::load_gamezone_pbp(seasons = "2021-22")

# Get a view of the data
head(pbp)
```

```
## # A tibble: 6 x 45
##   season  date       game_id play_id neutral  half home    away       home_name
##   <chr>   <date>       <dbl>   <dbl>   <dbl> <dbl> <chr>   <chr>      <chr>
## 1 2021-22 2022-01-08 2370457       1       0     1 Gonzaga Pepperdine Bulldogs
## 2 2021-22 2022-01-08 2370457       1       0     1 Gonzaga Pepperdine Bulldogs
## 3 2021-22 2022-01-08 2370457       1       0     1 Gonzaga Pepperdine Bulldogs
## 4 2021-22 2022-01-08 2370457       1       0     1 Gonzaga Pepperdine Bulldogs
## 5 2021-22 2022-01-08 2370457       1       0     1 Gonzaga Pepperdine Bulldogs
## 6 2021-22 2022-01-08 2370457       1       0     1 Gonzaga Pepperdine Bulldogs
## # ... with 36 more variables: away_name <chr>, home_timeouts <dbl>,
## #   away_timeouts <dbl>, home_score <dbl>, away_score <dbl>, score_diff <dbl>,
## #   team_id <dbl>, event_team <chr>, game_secs_remaining <dbl>,
## #   half_secs_remaining <dbl>, play_length <dbl>, desc <chr>,
## #   shot_outcome <chr>, free_throw <lgl>, three_pt <lgl>, shot_desc <chr>,
## #   loc_x <dbl>, loc_y <dbl>, shooter_id <dbl>, shooter <chr>, assist <chr>,
## #   substitution <dbl>, poss_before <chr>, poss_after <chr>, ...
```

First, we're going to perform some data wrangling to create some useful datasets for later visualizations and exploration.

```
# Create a dictionary of available games
single_games <- pbp %>%
  dplyr::group_by(game_id) %>%
  dplyr::summarise(dplyr::across(c(date, home, away), unique),
    dplyr::across(c(home_score, away_score), max),
    .groups = "drop"
  ) %>%
  dplyr::mutate(label = paste0(away, " @ ", home))

# Bind together such that it is one row per team (as opposed to one row per game)
games <- dplyr::bind_rows(
  single_games %>%
    dplyr::transmute(game_id, date, label,
      team = home, opponent = away,
      team_score = home_score, opponent_score = away_score, location = "home"
    ),
  single_games %>%
    dplyr::transmute(game_id, date, label,
      team = away, opponent = home,
      team_score = away_score, opponent_score = home_score, location = "away"
    )
) %>%
  dplyr::arrange(date)
```

```
# Function to summarize statistics from play-by-play data
summarise_games <- function(tbl) {
  tbl %>%
    dplyr::filter(!is.na(poss_before)) %>%
    dplyr::mutate(
      poss_number = as.numeric(poss_number),
      shot_made_numeric = dplyr::case_when(
        is.na(shot_outcome) ~ NA_real_,
        shot_outcome == "made" ~ 1,
        shot_outcome == "missed" ~ 0
      ),
      shot_value = dplyr::case_when(
        is.na(shot_outcome) ~ NA_real_,
        free_throw == 1 ~ 1,
        three_pt == 1 ~ 3,
        TRUE ~ 2
      ),
      points = dplyr::case_when(
        shot_made_numeric == 0 ~ 0,
        shot_made_numeric == 1 & free_throw == 1 ~ 1,
        shot_made_numeric == 1 & three_pt == 1 ~ 3,
        shot_made_numeric == 1 & three_pt == 0 & free_throw == 0 ~ 2
      )
    ) %>%
    dplyr::group_by(date, game_id, poss_before, poss_number) %>%
    dplyr::summarise(
      fgm = sum(shot_outcome == "made" & free_throw == FALSE, na.rm = TRUE),
      fga = sum(!is.na(shot_outcome) & free_throw == FALSE),
      ftm = sum(shot_outcome == "made" & free_throw == TRUE),
      fta = sum(!is.na(shot_outcome) & free_throw == TRUE),
      points = sum(points, na.rm = TRUE),
      .groups = "drop"
    ) %>%
    dplyr::group_by(date, game_id, team = poss_before) %>%
    dplyr::summarise(
      poss = dplyr::n(),
      dplyr::across(fgm:points, sum),
      .groups = "drop"
    ) %>%
    dplyr::mutate(pts_per_poss = points / poss)
}
```

```
# Summarize stats from each game
games_summarized <- pbp %>%
  summarise_games() %>%
  dplyr::left_join(games, by = c("date", "game_id", "team"))
```

Take a second to familiarize with yourself with the datasets we created (`single_games`, `games`, `games_summarized`, `pbp`). We are going to try to answer some interesting questions by creating visualizations!

Make a visualization to show Duke's (cumulative) point differential over the course of the season.

```
games
```

```
## # A tibble: 3,568 x 8
##    game_id date       label    team  opponent team_score opponent_score location
##      <dbl> <date>     <chr>    <chr> <chr>         <dbl>          <dbl> <chr>
##  1 2371488 2021-11-09 Jackson~ Illi~ Jackson~         71             47 home
##  2 2371501 2021-11-09 Loyola ~ Nort~ Loyola ~         83             67 home
##  3 2371525 2021-11-09 UAPB @ ~ Crei~ UAPB             90             77 home
##  4 2371545 2021-11-09 Western~ Nebr~ Western~         74             75 home
##  5 2371554 2021-11-09 St. Fra~ Wisc~ St. Fra~         81             58 home
##  6 2371638 2021-11-09 Miami (~ Geor~ Miami (~         69             72 home
##  7 2371757 2021-11-09 Bakersf~ UCLA  Bakersf~         95             58 home
##  8 2373041 2021-11-09 Kentuck~ Duke  Kentucky         79             71 home
##  9 2373052 2021-11-09 Canisiu~ Miam~ Canisius         77             67 home
## 10 2373075 2021-11-09 Bucknel~ Nort~ Bucknell         88             70 home
## # ... with 3,558 more rows
```

Make visualizations to determine the effect of home court advantage.

```
games
```

```
## # A tibble: 3,568 x 8
##    game_id date       label    team  opponent team_score opponent_score location
##      <dbl> <date>     <chr>    <chr> <chr>         <dbl>          <dbl> <chr>
##  1 2371488 2021-11-09 Jackson~ Illi~ Jackson~         71             47 home
##  2 2371501 2021-11-09 Loyola ~ Nort~ Loyola ~         83             67 home
##  3 2371525 2021-11-09 UAPB @ ~ Crei~ UAPB             90             77 home
##  4 2371545 2021-11-09 Western~ Nebr~ Western~         74             75 home
##  5 2371554 2021-11-09 St. Fra~ Wisc~ St. Fra~         81             58 home
##  6 2371638 2021-11-09 Miami (~ Geor~ Miami (~         69             72 home
##  7 2371757 2021-11-09 Bakersf~ UCLA  Bakersf~         95             58 home
##  8 2373041 2021-11-09 Kentuck~ Duke  Kentucky         79             71 home
##  9 2373052 2021-11-09 Canisiu~ Miam~ Canisius         77             67 home
## 10 2373075 2021-11-09 Bucknel~ Nort~ Bucknell         88             70 home
## # ... with 3,558 more rows
```

How has offensive efficiency (measured by points per possession) changed over the course of the season?

```
games_summarized
```

```
## # A tibble: 3,568 x 15
##    date       game_id team      poss   fgm   fga   ftm   fta points pts_per_poss
##    <date>       <dbl> <chr>    <int> <int> <int> <int> <int>  <dbl>        <dbl>
##  1 2021-11-09 2371488 Illinois    68    24    55    14    21     71        1.04
##  2 2021-11-09 2371488 Jackson~    67    19    51     2     5     47        0.701
##  3 2021-11-09 2371501 Loyola ~    72    24    55    12    21     67        0.931
##  4 2021-11-09 2371501 North C~    75    29    55    17    28     83        1.11
##  5 2021-11-09 2371525 Creight~    78    38    65     7    12     90        1.15
##  6 2021-11-09 2371525 UAPB        75    27    72    13    15     77        1.03
##  7 2021-11-09 2371545 Nebraska    64    23    59    23    31     74        1.16
##  8 2021-11-09 2371545 Western~    65    30    77     6    14     75        1.15
##  9 2021-11-09 2371554 St. Fra~    56    25    62     3     4     58        1.04
## 10 2021-11-09 2371554 Wiscons~    57    29    66    13    18     81        1.42
## # ... with 3,558 more rows, and 5 more variables: label <chr>, opponent <chr>,
## #   team_score <dbl>, opponent_score <dbl>, location <chr>
```

Make visualizations to show which teams have the best offensive efficiency.

```
games_summarized
```

```
## # A tibble: 3,568 x 15
##    date       game_id team      poss   fgm   fga   ftm   fta points pts_per_poss
##    <date>       <dbl> <chr>    <int> <int> <int> <int> <int>  <dbl>        <dbl>
##  1 2021-11-09 2371488 Illinois    68    24    55    14    21     71        1.04
##  2 2021-11-09 2371488 Jackson~    67    19    51     2     5     47        0.701
##  3 2021-11-09 2371501 Loyola ~    72    24    55    12    21     67        0.931
##  4 2021-11-09 2371501 North C~    75    29    55    17    28     83        1.11
##  5 2021-11-09 2371525 Creight~    78    38    65     7    12     90        1.15
##  6 2021-11-09 2371525 UAPB        75    27    72    13    15     77        1.03
##  7 2021-11-09 2371545 Nebraska    64    23    59    23    31     74        1.16
##  8 2021-11-09 2371545 Western~    65    30    77     6    14     75        1.15
##  9 2021-11-09 2371554 St. Fra~    56    25    62     3     4     58        1.04
## 10 2021-11-09 2371554 Wiscons~    57    29    66    13    18     81        1.42
## # ... with 3,558 more rows, and 5 more variables: label <chr>, opponent <chr>,
## #   team_score <dbl>, opponent_score <dbl>, location <chr>
```

Make a scatter plot with team offensive efficiency on the x-axis and team defensive efficiency on the y-axis. Size the points by the number of possessions charted. Use the `ggrepel` package to label the points by team name.

```
games_summarized
```

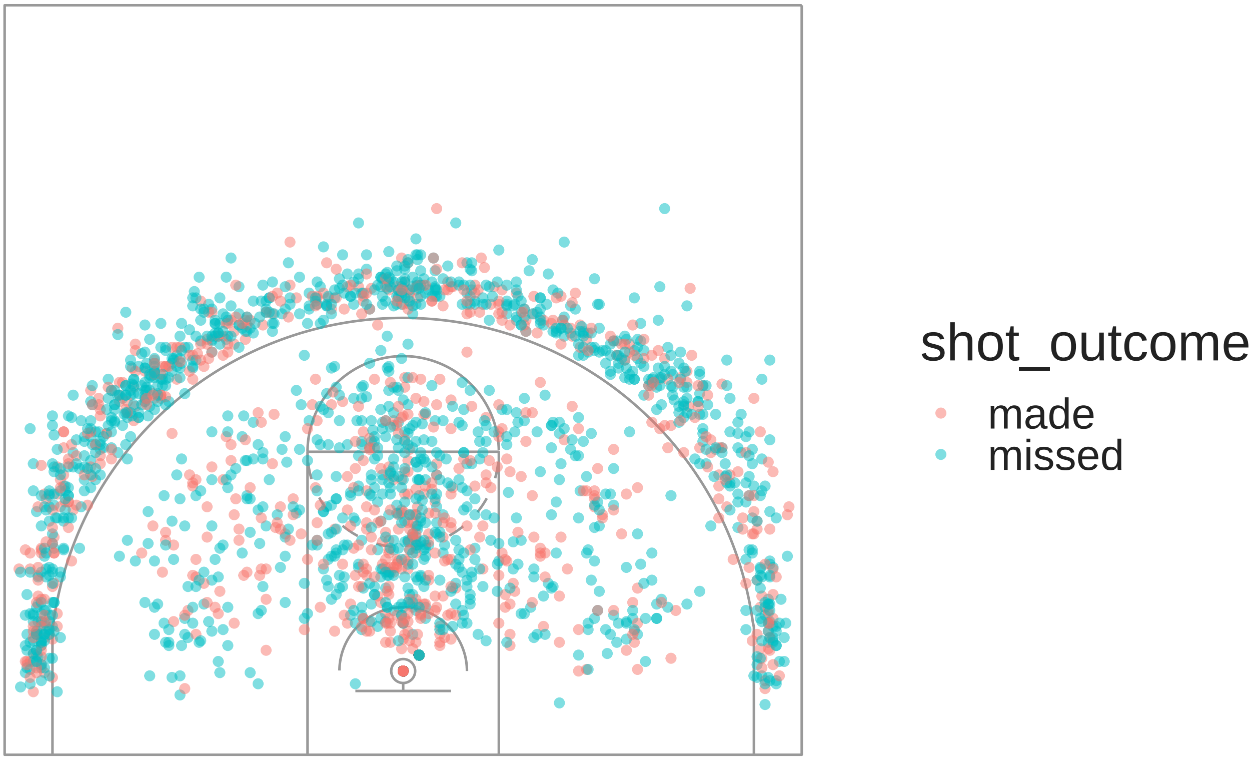
```
## # A tibble: 3,568 x 15
##    date       game_id team      poss   fgm   fga   ftm   fta points pts_per_poss
##    <date>       <dbl> <chr>    <int> <int> <int> <int> <int>  <dbl>        <dbl>
##  1 2021-11-09 2371488 Illinois    68    24    55    14    21     71        1.04
##  2 2021-11-09 2371488 Jackson~    67    19    51     2     5     47        0.701
##  3 2021-11-09 2371501 Loyola ~    72    24    55    12    21     67        0.931
##  4 2021-11-09 2371501 North C~    75    29    55    17    28     83        1.11
##  5 2021-11-09 2371525 Creight~    78    38    65     7    12     90        1.15
##  6 2021-11-09 2371525 UAPB        75    27    72    13    15     77        1.03
##  7 2021-11-09 2371545 Nebraska    64    23    59    23    31     74        1.16
##  8 2021-11-09 2371545 Western~    65    30    77     6    14     75        1.15
##  9 2021-11-09 2371554 St. Fra~    56    25    62     3     4     58        1.04
## 10 2021-11-09 2371554 Wiscons~    57    29    66    13    18     81        1.42
## # ... with 3,558 more rows, and 5 more variables: label <chr>, opponent <chr>,
## #   team_score <dbl>, opponent_score <dbl>, location <chr>
```

Let's make some shot charts! The greatest part of the `gamezoneR` package is how many shot locations (x, y) are charted. Let's look at Duke's shot attempts this season:

```
# Find Duke game IDs
duke_game_ids <- games %>%
  dplyr::filter(team == "Duke") %>%
  dplyr::pull(game_id)

# Find Duke shot attempts
duke_shots <- pbp %>%
  dplyr::filter(game_id %in% duke_game_ids) %>%
  dplyr::filter(!is.na(loc_x))
```

Here is a *very* basic shot chart for Duke:

```
gamezoneR::base_court +
  geom_point(
    data = duke_shots,
    aes(loc_x, loc_y, color = shot_outcome),
    alpha = 0.5
  )
```



Play around with different versions of shot charts. Make some by a particular shooter, by a particular game, etc.

I encourage you to explore the data more! Answer questions you find interesting! While making this tutorial I decided to look into free throw attempt rates by home and away, specifically looking at Duke and the

Cameron Crazies.

```
games_summarized %>%
  dplyr::filter(opponent == "Duke") %>%
  dplyr::group_by(opponent, location) %>%
  dplyr::summarise(
    games = dplyr::n(),
    fta = mean(fta),
    .groups = "drop"
  )
```

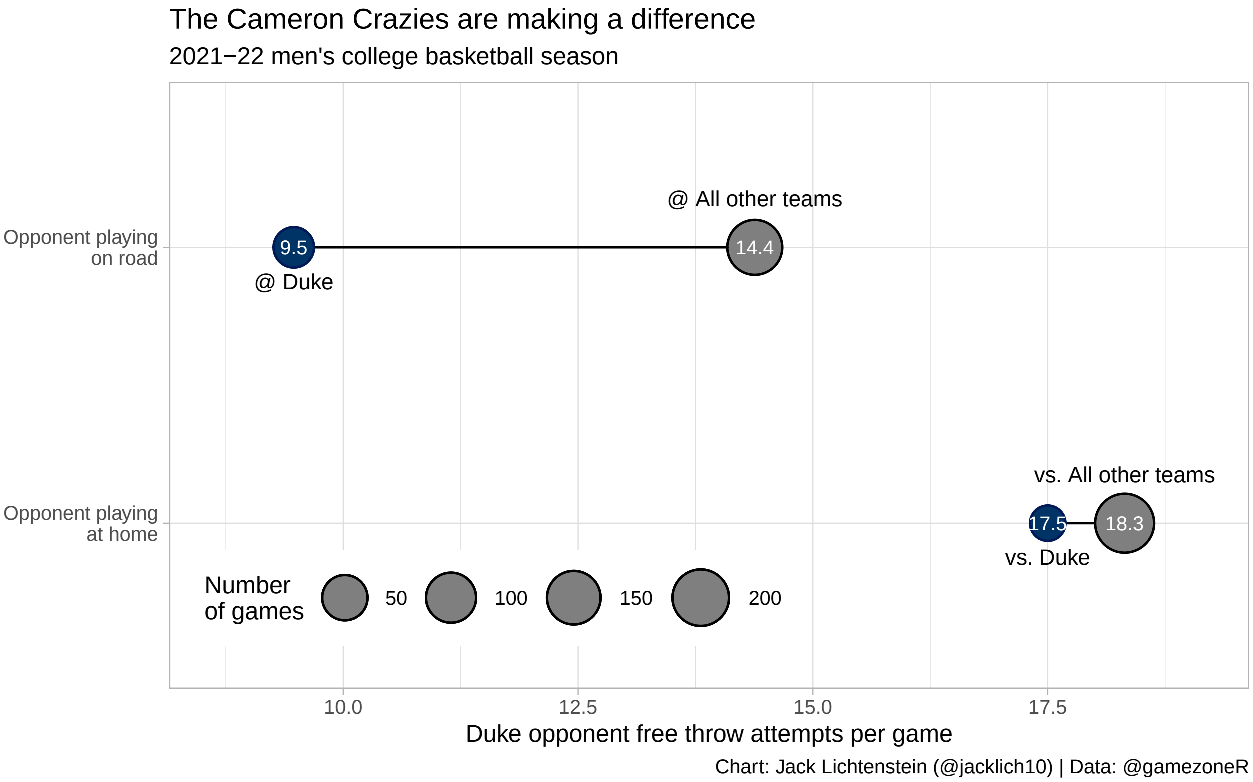
```
## # A tibble: 2 x 4
##   opponent location games   fta
##   <chr>    <chr>    <int> <dbl>
## 1 Duke     away        19  9.47
## 2 Duke     home         8 17.5
```

```
games_summarized %>%
  dplyr::filter(team == "Duke") %>%
  dplyr::group_by(team, location) %>%
  dplyr::summarise(
    games = dplyr::n(),
    fta = mean(fta),
    .groups = "drop"
  )
```

```
## # A tibble: 2 x 4
##   team  location games   fta
##   <chr> <chr>    <int> <dbl>
## 1 Duke  away         8  14.1
## 2 Duke  home        19  18.8
```

I then made a plot which I posted on twitter. This is the code for the plot, if interested.

```
# If ggtext is not installed, install
if (!require("ggtext")) {
  install.packages("ggtext")
}

# Find all Duke opponents
duke_opponents <- games %>%
  dplyr::filter(team == "Duke") %>%
  dplyr::pull(opponent)

# Duke color
duke_color <- gamezoneR::mbb_team_info$primary_color[gamezoneR::mbb_team_info$team_name == "Duke"]

# Duke fill
duke_fill <- gamezoneR::mbb_team_info$tertiary_color[gamezoneR::mbb_team_info$team_name == "Duke"]

# Find Duke opponent free throw attempts by home/away, playing Duke/not Duke
duke_opp_fta <- games_summarized %>%
  dplyr::filter(team %in% duke_opponents) %>%
  dplyr::mutate(playing_duke = ifelse(opponent == "Duke", "duke", "others")) %>%
  dplyr::group_by(location = ifelse(location == "home", "Opponent playing\nat home", "Opponent playing\
  dplyr::summarise(
    games = dplyr::n(),
    fta = mean(fta),
    .groups = "drop"
  )

duke_opp_fta %>%
  tidyr::pivot_wider(
    names_from = playing_duke,
    values_from = c(games, fta)
  ) %>%
  ggplot(aes(y = location)) +
  ggtext::geom_richtext(aes(
    x = fta_duke,
    label = ifelse(location == "Opponent playing\non road", "@ Duke", "vs. Duke"),
    vjust = ifelse(location == "away", -1.75, 2.25)
  ),
  size = 3.5, hjust = 0.5,
  fill = NA, label.color = NA, # remove background and outline
  label.padding = grid::unit(rep(0, 4), "pt")
  ) +
  ggtext::geom_richtext(aes(
    x = fta_others,
    label = ifelse(location == "Opponent playing\non road", "@ All other teams", "vs. All other teams")
  ),
  size = 3.5, vjust = -1.75, hjust = 0.5,
  fill = NA, label.color = NA, # remove background and outline
  label.padding = grid::unit(rep(0, 4), "pt")
  ) +
  geom_segment(aes(fta_duke, xend = fta_others, yend = location),
    color = "black"
  ) +
  geom_point(aes(fta_duke,
    size = games_duke,
    color = duke_color, fill = duke_fill
  ),
  stroke = 0.8, pch = 21
  ) +
  geom_point(aes(fta_others, size = games_others),
    stroke = 0.8, pch = 21, fill = "grey50"
  ) +
  geom_text(aes(fta_duke, label = scales::number(fta_duke, accuracy = 0.1)),
    size = 3, color = "white"
  ) +
  geom_text(aes(fta_others, label = scales::number(fta_others, accuracy = 0.1)),
    size = 3, color = "white"
  ) +
  scale_size_continuous(range = c(7, 12)) +
  scale_x_continuous(expand = expansion(mult = c(0.15, 0.15))) +
  scale_color_identity() +
  scale_fill_identity() +
  guides(shape = guide_legend(override.aes = list(size = 0.25))) +
  theme(
    axis.title.y = element_text(angle = 0, vjust = 0.5),
    legend.position = c(0.3, 0.15),
    legend.direction = "horizontal"
  ) +
  labs(
    title = "The Cameron Crazies are making a difference",
    subtitle = "2021-22 men's college basketball season",
    x = "Duke opponent free throw attempts per game",
    y = NULL,
    size = "Number\nof games",
    caption = "Chart: Jack Lichtenstein (@jacklich10) | Data: @gamezoneR"
  )
```



Go explore the data yourself! Visualize where teams like to shoot from relative to league average. Visualize where teams are most efficient shooting from. Look at free throw attempt rates for other teams. Do whatever interests you!