Game Analysis

Benjamin Calderaio, Jr.

2025-02-11

Libraries

```
library(DBI)
library(RMariaDB)
library(tidyr)
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
```

```
library(htmlwidgets)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Environment Variables

Get variables from .Renviron

Functions

get_query

Returns a query string.

get_csv_data

Gets csv data and returns a data frame.

```
#' Get csv Data
#'
#' This function returns data from a database table.
#'
#' @param con The database connection
#' @param query The database query used to return the table rows
#' @return The results in a data frame
#' @examples
#' data <- get_data(con,qry)
get_csv_data <- function(fpath) {
  tryCatch({
    result_df <- read.csv(fpath)
    return(result_df)
  }, error = function(e) {
    print(paste("An error occurred:", e))
  })
}
```

get_data

Gets data from a database table and returns a data frame.

```
#' Get Database Data
#'
#' This function returns data from a database table.
#'
#' @param con The database connection
#' @param query The database query used to return the table rows
#' @return The results in a data frame
#' @examples
#' data <- get_data(con,qry)
get_data <- function(con,query) {
  tryCatch({
    result_df <- dbGetQuery(con, query)
    return(result_df)
  }, error = function(e) {
```

```
    print(paste("An error occurred:", e))
  }, finally = {
    dbDisconnect(con)
  })
}
```

get_detail_fav_covers_by_year

Transforms detail spread for favorites and returns a data frame.

```
get_detail_fav_covers_by_year <- function(leagyear,det_df) {
  leagyear_value <- leagyear
  df <- det_df %>%
    filter(leagyear == leagyear_value, isfav == 1, iscover == 1) %>%
    group_by(leagid,spread) %>%
    summarise(iscover_sum = sum(iscover, na.rm = TRUE), .groups = "drop") %>%
    ungroup()
  df <- as.data.frame(df)
  return(df)
}
```

get_detail_fav_covers_v_dog_covers_by_year

Transforms detail spread for favorites v underdogs and returns a data frame.

```
get_detail_fav_covers_v_dog_covers_by_year <- function(leagyear,det_df) {
  # det1_df <- read.csv('vw_gameteamresultsdetail.csv')
  leagyear_value <- leagyear
  df1 <- det_df %>%
    filter(leagyear == leagyear_value, isfav == 1, ispush == 0) %>%
    group_by(leagid,spread) %>%
    summarise(
      favcover = sum(iscover == 1, na.rm = TRUE),
      dogcover = sum(iscover == 0, na.rm = TRUE), .groups = "drop"
      ) %>%
    ungroup() %>%
    arrange(spread,leagid)
  df1 <- as.data.frame(df1)
  return(df1)
}
```

create_plotly_spread_bar_chart

Creates a bar chart for favorites v underdogs.

```
create_plotly_spread_bar_chart <- function(data) {
  df_nfl <- data %>% filter(leagid == 1)
  if(!is.numeric(df_nfl$spread)) {
    df_nfl$spread <- as.numeric((as.character(df_nfl$spread)))
  }
  df_nfl <- df_nfl %>% arrange(spread)
```

```
  df_nfl$spread <- factor(df_nfl$spread, levels =
sort(unique(df_nfl$spread)))
  if (is.factor(df_nfl$spread)) {
    df_nfl$spread <- as.numeric(as.character(df_nfl$spread))
  }

  new_df <- data.frame(
    spread = df_nfl$spread,
    favorite = df_nfl$favcover,
    underdog = df_nfl$dogcover
  )
  # Sort data by offense total yards in descending order
  data_sorted_off <- new_df[order(-new_df$spread), ]

  # Reshape data for plotly, mapping variable names to readable labels
  data_long <- melt(data_sorted_off, id.vars = "spread", variable.name =
"type", value.name = "covers")
  data_long$type <- ifelse(data_long$type == "favorite", "favorite",
"underdog")

  # Plotly bar chart
  plot <- plot_ly(data = data_long,
                  x = ~spread,
                  y = ~covers,
                  color = ~type,
                  type = "bar",
                  colors = c("favorite" = "blue", "underdog" = "orange")) %>%
    layout(title = "Favorite vs Underdog Spread cover",
           xaxis = list(title = "Spread", tickangle = -90),
           yaxis = list(title = "Covers"),
           barmode = 'group')  # This sets bars to be side by side

  print(plot)

  # htmlwidgets::saveWidget(plot, "fav_dog_spread.html", selfcontained =
TRUE)
}
```

create_ggplot_team_bar_chart

Creates a bar chart using ggplot for total offensive and defensive yards.

```
create_ggplot_team_bar_chart <- function(data) {
  df_nfl <- data
  new_df <- data.frame(
    teamname = as.character(df_nfl$teamname),  # Convert teamname to
character
    offense = df_nfl$offtotyds,
    defense = df_nfl$deftotyds
```

```
  )
  df <- suppressWarnings(tidyr::gather(new_df, yards, total,
offense:defense))  # Create long format

  # Sort by total yards descending for each team (optional, if you want
sorting)
  # df$teamname <- factor(df$teamname, levels =
unique(df$teamname[order(df$total, decreasing = TRUE)]))

  plot <- ggplot(df, aes(teamname, total, fill=yards))
  plot <- plot +
    geom_bar(stat = "identity", position = 'dodge') +
    labs(title = "Offense vs Defense Total Yards", x = "Team", y = "Total
Yards", fill = "Type") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    scale_fill_manual(values = c("offense" = "blue", "defense" = "orange"))

  print(plot)

}
```

create_ggplot_team_box_chart

Creates team box charts using ggplot for total offensive and defensive yard distrubtion.

```
create_ggplot_team_box_chart <- function(data) {
  data_sorted_off <- data
  # Boxplot for defense total yards
  box_plot_d <- ggplot(data_sorted_off, aes(x = teamname, y = defense)) +
    geom_boxplot(fill = "red") +
    labs(title = "Distribution of Defense Total Yards",x = "Team", y = "Total
Yards") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
  print(box_plot_d)

  box_plot_o <- ggplot(data_sorted_off, aes(x = teamname, y = offense)) +
    geom_boxplot() +
    labs(title = "Distribution of Offense Total Yards",x = "Team", y = "Total
Yards") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
  print(box_plot_o)
}
```

create_plotly_team_bar_chart

Creates a bar chart using plotly for total offensive and defensive yards.

```
create_plotly_team_bar_chart <- function(data) {
  df_nfl <- data
  new_df <- data.frame(
    teamname = df_nfl$teamname,
    offense = df_nfl$offtotyds,
    defense = df_nfl$deftotyds
  )
  # Sort data by offense total yards in descending order
  data_sorted_off <- new_df[order(-new_df$offense), ]

  # Reshape data for plotly, mapping variable names to readable labels
  data_long <- melt(data_sorted_off, id.vars = "teamname", variable.name =
"type", value.name = "yards")
  data_long$type <- ifelse(data_long$type == "offense", "offense", "defense")

  # Plotly bar chart
  plot <- plot_ly(data = data_long,
                  x = ~teamname,
                  y = ~yards,
                  color = ~type,
                  type = "bar",
                  colors = c("offense" = "blue", "defense" = "orange")) %>%
    layout(title = "Offense vs Defense Total Yards",
           xaxis = list(title = "Team", tickangle = -45),
           yaxis = list(title = "Total Yards"),
           barmode = 'group')  # This sets bars to be side by side
  print(plot)

  create_ggplot_team_box_chart(data_sorted_off)
  # htmlwidgets::saveWidget(plot, "off_def_totyds.html", selfcontained =
TRUE)
}
```

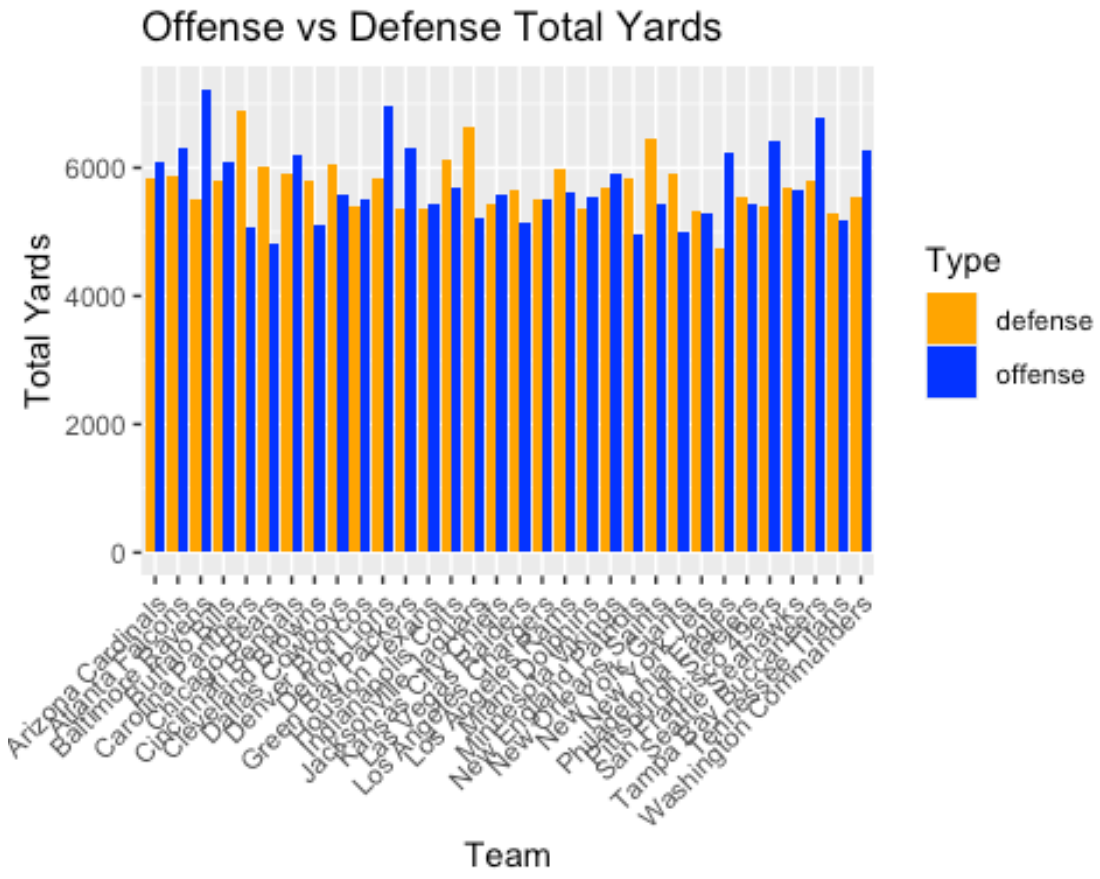
main

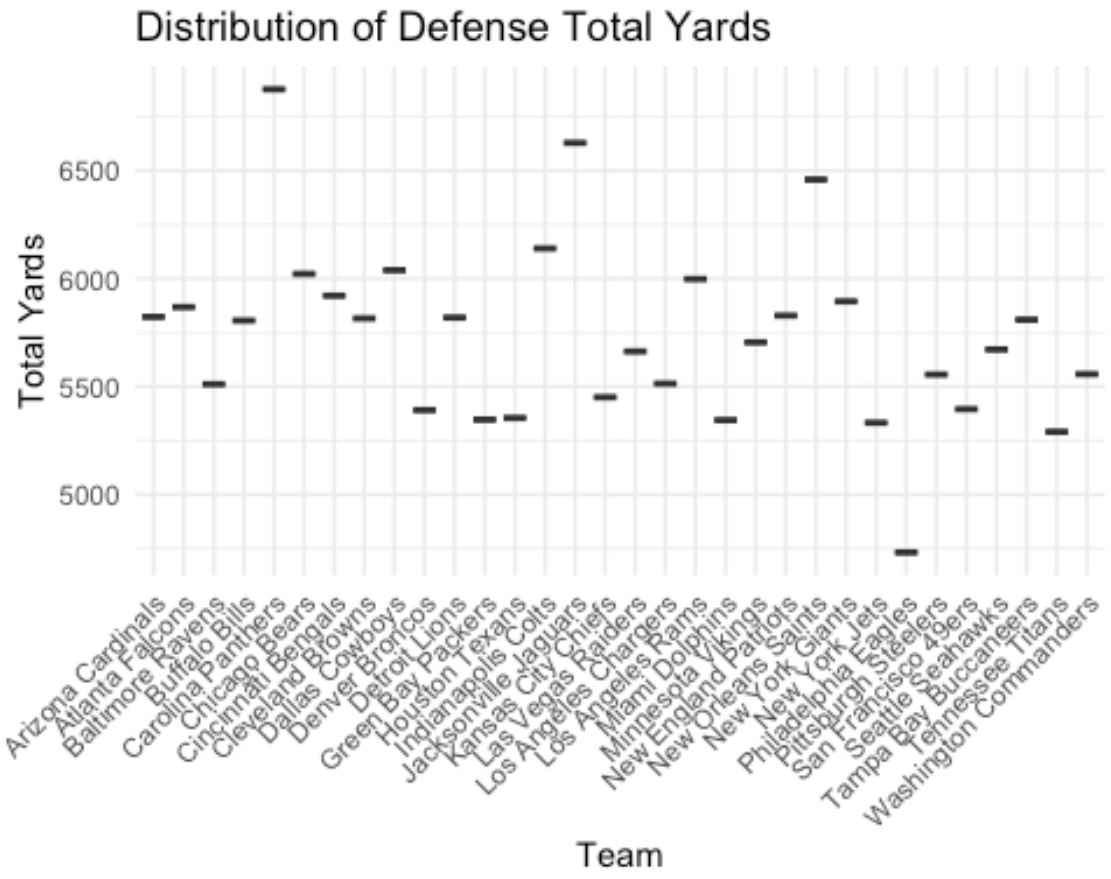
R Markdown main method.

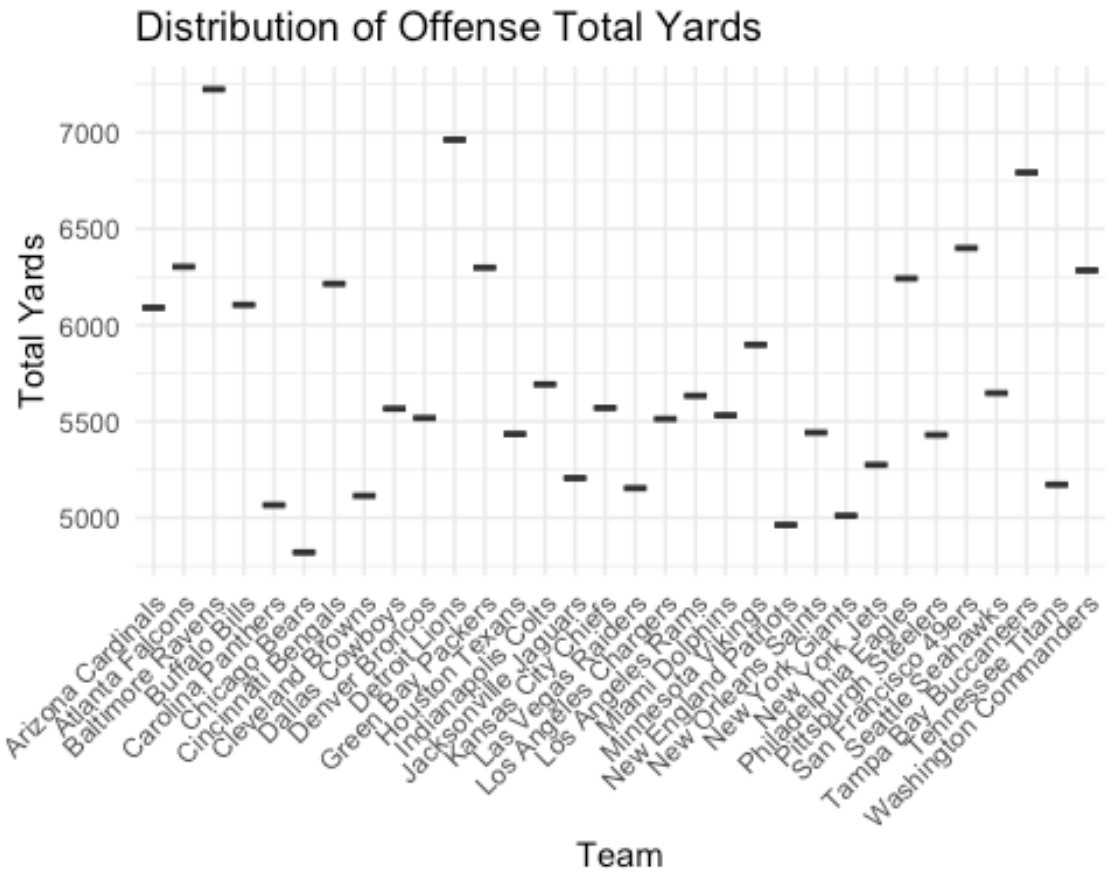
```
main <- function(){
  csv_data <- get_csv_data('teamseasontotals_2024.csv')
  create_ggplot_team_bar_chart(csv_data)
  create_plotly_team_bar_chart(csv_data)

  con <- db_con()
  qry <- get_query(Sys.getenv("TABLE_GAMEDETAILS"))
  db_data <- get_data(con,qry)

  df <- get_detail_fav_covers_v_dog_covers_by_year(2024,db_data)
  create_plotly_spread_bar_chart(df)
}
```

Output

```
main()
```


Offense vs Defense Total Yards
Total Yards
6000
4000
2000
0
Team
Type
defense
offense


Distribution of Defense Total Yards
Total Yards
6500
6000
5500
5000
Arizona Cardinals
Atlanta Falcons
Baltimore Ravens
Buffalo Bills
Carolina Panthers
Chicago Bears
Cincinnati Bengals
Cleveland Browns
Dallas Cowboys
Denver Broncos
Detroit Lions
Green Bay Packers
Houston Texans
Indianapolis Colts
Jacksonville Jaguars
Kansas City Chiefs
Las Vegas Raiders
Los Angeles Chargers
Los Angeles Rams
Miami Dolphins
Minnesota Vikings
New England Patriots
New Orleans Saints
New York Giants
New York Jets
Philadelphia Eagles
Pittsburgh Steelers
San Francisco 49ers
Seattle Seahawks
Tampa Bay Buccaneers
Tennessee Titans
Washington Commanders
Team


Distribution of Offense Total Yards
Total Yards
7000
6500
6000
5500
5000
Arizona Cardinals
Atlanta Falcons
Baltimore Ravens
Buffalo Bills
Carolina Panthers
Chicago Bears
Cincinnati Bengals
Cleveland Browns
Dallas Cowboys
Denver Broncos
Detroit Lions
Green Bay Packers
Houston Texans
Indianapolis Colts
Jacksonville Jaguars
Kansas City Chiefs
Las Vegas Raiders
Los Angeles Chargers
Los Angeles Rams
Miami Dolphins
Minnesota Vikings
New England Patriots
New Orleans Saints
New York Giants
New York Jets
Philadelphia Eagles
Pittsburgh Steelers
San Francisco 49ers
Seattle Seahawks
Tampa Bay Buccaneers
Tennessee Titans
Washington Commanders
Team