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()

