ASDS-6301 Final Project

A Regression-Based Analysis of NBA Team Performance

Phuong Trinh, Sashi Kumar Soni, Ahmet Toure

2025-12-09

Import Packages

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   4.0.0     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(lubridate)
library(GGally)
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-10
```

```
library(lme4)
library(broom)
library(DescTools)
library(janitor)
```

```
##
## Attaching package: 'janitor'
```

```
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(dplyr)
library(tidyr)
library(ggplot2)
library(purrr)
```

Load dataset

```
set.seed(123)
nba <- read_csv("NBA_FULL_DATASET_2023_2025-2.csv",show_col_types = FALSE)
head(nba)
```

```
## # A tibble: 6 x 42
##     Team_ID Game_ID GAME_DATE  MATCHUP WL        W     L W_PCT   MIN   FGM   FGA
##       <dbl> <chr>   <date>     <chr>   <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1    1.61e9 002230~ 2023-10-25 ATL @ ~ L         0     1 0       240    39    93
## 2    1.61e9 002230~ 2023-10-27 ATL vs~ L         0     2 0       240    42    87
## 3    1.61e9 002230~ 2023-10-29 ATL @ ~ W         1     2 0.333   240    47    93
## 4    1.61e9 002230~ 2023-10-30 ATL vs~ W         2     2 0.5     240    48    86
## 5    1.61e9 002230~ 2023-11-01 ATL vs~ W         3     2 0.6     240    46    92
## 6    1.61e9 002230~ 2023-11-04 ATL @ ~ W         4     2 0.667   240    45    93
## # i 31 more variables: FG_PCT <dbl>, FG3M <dbl>, FG3A <dbl>, FG3_PCT <dbl>,
## #   FTM <dbl>, FTA <dbl>, FT_PCT <dbl>, OREB <dbl>, DREB <dbl>, REB <dbl>,
## #   AST <dbl>, STL <dbl>, BLK <dbl>, TOV <dbl>, PF <dbl>, PTS <dbl>,
## #   TEAM_NAME <chr>, TEAM_ID <dbl>, SEASON <chr>, OPP_ABBR <chr>,
## #   OPPONENT <chr>, OPPONENT_ID <dbl>, HOME_AWAY <chr>, PREV_GAME_DATE <date>,
## #   DAYS_REST <dbl>, BACK_TO_BACK <dbl>, ARENA_LAT <dbl>, ARENA_LON <dbl>,
## #   PREV_LAT <dbl>, PREV_LON <dbl>, TRAVEL_DISTANCE <dbl>
```

Data Preprocessing

Data Cleaning

```
# Check variable names
nba <- nba %>% clean_names()
```

Compute PLUS/MINUS

```
library(dplyr)

pm <- nba %>%
```

```
  select(game_id, team_name, opponent, pts) %>%
  inner_join(
    nba %>% select(game_id, team_name, pts) %>%
      rename(opp_team = team_name, opp_pts = pts),
    by = "game_id"
  ) %>%
  filter(team_name != opp_team) %>%
  group_by(game_id, team_name, opponent) %>%
  summarise(plus_minus = first(pts - opp_pts), .groups = "drop")
```

```
## Warning in inner_join(., nba %>% select(game_id, team_name, pts) %>% rename(opp_team = team_name, : I
## i Row 1 of `x` matches multiple rows in `y`.
## i Row 16 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
nba <- nba %>%
  left_join(pm, by = c("game_id", "team_name", "opponent"))

summary(nba$plus_minus)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     -62     -10       0       0      10      62
```

```
# Check missing values
missing_summary <- colSums(is.na(nba))
missing_summary
```

```
##         team_id         game_id       game_date         matchup              wl
##               0               0               0               0               0
##               w               l           w_pct             min             fgm
##               0               0               0               0               0
##             fga          fg_pct            fg3m            fg3a         fg3_pct
##               0               0               0               0               0
##             ftm             fta          ft_pct            oreb            dreb
##               0               0               0               0               0
##             reb             ast             stl             blk             tov
##               0               0               0               0               0
##              pf             pts       team_name      team_id_2          season
##               0               0               0               0               0
##        opp_abbr        opponent     opponent_id      home_away  prev_game_date
##               0               0               0               0              30
##       days_rest    back_to_back      arena_lat      arena_lon        prev_lat
##               0               0               0               0              30
##        prev_lon travel_distance      plus_minus
##              30               0               0
```

```
missing_summary[missing_summary > 0]
```

```
## prev_game_date       prev_lat       prev_lon
##             30             30             30
```

```r
## Handle Missing Values

# 1. Replace missing travel distance with 0 (first game of each team)
nba <- nba %>%
  mutate(
    travel_distance = ifelse(is.na(travel_distance), 0, travel_distance)
  )

# 2. Replace missing previous coordinates with current coordinates
nba <- nba %>%
  mutate(
    prev_lat = ifelse(is.na(prev_lat), arena_lat, prev_lat),
    prev_lon = ifelse(is.na(prev_lon), arena_lon, prev_lon)
  )

# 3. Replace missing previous game date with the season start date
# (or leave as NA - either is acceptable)
nba <- nba %>%
  mutate(
    prev_game_date = ifelse(is.na(prev_game_date), game_date, prev_game_date)
  )

# Check again
colSums(is.na(nba))
```

```
##         team_id         game_id       game_date         matchup              wl
##               0               0               0               0               0
##               w               l           w_pct             min             fgm
##               0               0               0               0               0
##             fga          fg_pct            fg3m            fg3a         fg3_pct
##               0               0               0               0               0
##             ftm             fta          ft_pct            oreb            dreb
##               0               0               0               0               0
##             reb             ast             stl             blk             tov
##               0               0               0               0               0
##              pf             pts       team_name       team_id_2          season
##               0               0               0               0               0
##        opp_abbr        opponent     opponent_id       home_away  prev_game_date
##               0               0               0               0               0
##       days_rest    back_to_back       arena_lat       arena_lon        prev_lat
##               0               0               0               0               0
##        prev_lon travel_distance      plus_minus
##               0               0               0
```

```r
# Check duplicate records
num_duplicates <- sum(duplicated(nba))
num_duplicates
```

```
## [1] 0
```

```r
# Remove duplicates
nba <- nba %>% distinct()
```

```r
# Drop team_id and game_id
nba <- nba %>% select(-team_id, -game_id)
```

Descriptive Statistic

```r
# Check data types
str(nba)
```

```
## tibble [4,920 x 41] (S3: tbl_df/tbl/data.frame)
##  $ game_date      : Date[1:4920], format: "2023-10-25" "2023-10-27" ...
##  $ matchup        : chr [1:4920] "ATL @ CHA" "ATL vs. NYK" "ATL @ MIL" "ATL vs. MIN" ...
##  $ wl             : chr [1:4920] "L" "L" "W" "W" ...
##  $ w              : num [1:4920] 0 0 1 2 3 4 4 5 5 6 ...
##  $ l              : num [1:4920] 1 2 2 2 2 2 3 3 4 4 ...
##  $ w_pct          : num [1:4920] 0 0 0.333 0.5 0.6 0.667 0.571 0.625 0.556 0.6 ...
##  $ min            : num [1:4920] 240 240 240 240 240 240 240 240 240 240 ...
##  $ fgm            : num [1:4920] 39 42 47 48 46 45 38 41 38 45 ...
##  $ fga            : num [1:4920] 93 87 93 86 92 93 102 85 87 87 ...
##  $ fg_pct         : num [1:4920] 0.419 0.483 0.505 0.558 0.5 0.484 0.373 0.482 0.437 0.517 ...
##  $ fg3m           : num [1:4920] 5 12 15 14 9 14 14 15 18 12 ...
##  $ fg3a           : num [1:4920] 29 32 37 30 32 41 42 39 46 33 ...
##  $ fg3_pct        : num [1:4920] 0.172 0.375 0.405 0.467 0.281 0.341 0.333 0.385 0.391 0.364 ...
##  $ ftm            : num [1:4920] 27 24 18 17 29 19 27 23 15 24 ...
##  $ fta            : num [1:4920] 33 30 22 18 32 24 32 30 18 26 ...
##  $ ft_pct         : num [1:4920] 0.818 0.8 0.818 0.944 0.906 0.792 0.844 0.767 0.833 0.923 ...
##  $ oreb           : num [1:4920] 12 9 13 4 14 19 25 8 15 10 ...
##  $ dreb           : num [1:4920] 30 35 33 32 43 33 34 28 33 31 ...
##  $ reb            : num [1:4920] 42 44 46 36 57 52 59 36 48 41 ...
##  $ ast            : num [1:4920] 24 28 32 28 26 28 27 22 23 28 ...
##  $ stl            : num [1:4920] 12 7 15 6 8 4 8 13 6 9 ...
##  $ blk            : num [1:4920] 1 6 2 7 3 6 3 7 6 7 ...
##  $ tov            : num [1:4920] 12 14 17 9 20 11 15 17 21 14 ...
##  $ pf             : num [1:4920] 19 20 17 12 16 16 16 25 19 19 ...
##  $ pts            : num [1:4920] 110 120 127 127 130 123 117 120 109 126 ...
##  $ team_name      : chr [1:4920] "Atlanta Hawks" "Atlanta Hawks" "Atlanta Hawks" "Atlanta Hawks" ...
##  $ team_id_2      : num [1:4920] 1.61e+09 1.61e+09 1.61e+09 1.61e+09 1.61e+09 ...
##  $ season         : chr [1:4920] "2023-24" "2023-24" "2023-24" "2023-24" ...
##  $ opp_abbr       : chr [1:4920] "CHA" "NYK" "MIL" "MIN" ...
##  $ opponent       : chr [1:4920] "Charlotte Hornets" "New York Knicks" "Milwaukee Bucks" "Minnesota T
##  $ opponent_id    : num [1:4920] 1.61e+09 1.61e+09 1.61e+09 1.61e+09 1.61e+09 ...
##  $ home_away      : chr [1:4920] "Away" "Home" "Away" "Home" ...
##  $ prev_game_date : num [1:4920] 19655 19655 19657 19659 19660 ...
##  $ days_rest      : num [1:4920] 5 2 2 1 2 3 2 3 2 3 ...
##  $ back_to_back   : num [1:4920] 0 0 0 1 0 0 0 0 0 0 ...
##  $ arena_lat      : num [1:4920] 35.2 33.8 43 33.8 33.8 ...
##  $ arena_lon      : num [1:4920] -80.8 -84.4 -87.9 -84.4 -84.4 ...
##  $ prev_lat       : num [1:4920] 35.2 35.2 33.8 43 33.8 ...
##  $ prev_lon       : num [1:4920] -80.8 -80.8 -84.4 -87.9 -84.4 ...
##  $ travel_distance: num [1:4920] 0 227 668 668 0 ...
##  $ plus_minus     : num [1:4920] -6 -6 17 14 9 18 -9 1 -8 6 ...
```

```r
# Convert categorical variables to factors
nba <- nba %>%
  mutate(
    game_date      = as_date(game_date),
    prev_game_date = as_date(prev_game_date),
    season         = factor(season),
    wl             = factor(wl, levels = c("L", "W")),
    home_away      = factor(home_away),
    back_to_back   = factor(back_to_back, levels = c("0", "1")),
    team_name      = factor(team_name),
    opponent       = factor(opponent)
  )

summary(nba)
```

```
##    game_date            matchup          wl             w         
##  Min.   :2023-10-24   Length:4920        L:2460   Min.   : 0.00  
##  1st Qu.:2024-01-19   Class :character   W:2460   1st Qu.: 9.00  
##  Median :2024-07-18   Mode  :character            Median :19.00  
##  Mean   :2024-07-19                               Mean   :20.74  
##  3rd Qu.:2025-01-18                               3rd Qu.:31.00  
##  Max.   :2025-04-13                               Max.   :68.00  
##                                                                  
##        l              w_pct             min              fgm       
##  Min.   : 0.00   Min.   :0.0000   Min.   :240.0   Min.   :22.00  
##  1st Qu.: 9.00   1st Qu.:0.3850   1st Qu.:240.0   1st Qu.:38.00  
##  Median :19.00   Median :0.5220   Median :240.0   Median :42.00  
##  Mean   :20.76   Mean   :0.4998   Mean   :241.3   Mean   :41.93  
##  3rd Qu.:30.00   3rd Qu.:0.6110   3rd Qu.:240.0   3rd Qu.:45.00  
##  Max.   :68.00   Max.   :1.0000   Max.   :290.0   Max.   :65.00  
##                                                                  
##       fga             fg_pct            fg3m            fg3a       
##  Min.   : 67.00   Min.   :0.2770   Min.   : 2.00   Min.   :12.00  
##  1st Qu.: 84.00   1st Qu.:0.4340   1st Qu.:10.00   1st Qu.:31.00  
##  Median : 89.00   Median :0.4710   Median :13.00   Median :36.00  
##  Mean   : 89.07   Mean   :0.4717   Mean   :13.19   Mean   :36.34  
##  3rd Qu.: 94.00   3rd Qu.:0.5100   3rd Qu.:16.00   3rd Qu.:41.00  
##  Max.   :119.00   Max.   :0.6890   Max.   :29.00   Max.   :63.00  
##                                                                   
##     fg3_pct            ftm              fta            ft_pct       
##  Min.   :0.0690   Min.   : 0.00   Min.   : 0.0   Min.   :0.0000  
##  1st Qu.:0.3080   1st Qu.:13.00   1st Qu.:17.0   1st Qu.:0.7140  
##  Median :0.3610   Median :17.00   Median :21.0   Median :0.7890  
##  Mean   :0.3624   Mean   :16.97   Mean   :21.7   Mean   :0.7814  
##  3rd Qu.:0.4150   3rd Qu.:21.00   3rd Qu.:26.0   3rd Qu.:0.8500  
##  Max.   :0.6800   Max.   :44.00   Max.   :53.0   Max.   :1.0000  
##                                                                  
##       oreb            dreb            reb             ast       
##  Min.   : 0.00   Min.   :16.00   Min.   :23.00   Min.   : 9.00  
##  1st Qu.: 8.00   1st Qu.:29.00   1st Qu.:39.00   1st Qu.:23.00  
##  Median :11.00   Median :33.00   Median :44.00   Median :26.00  
##  Mean   :10.84   Mean   :32.99   Mean   :43.83   Mean   :26.61  
##  3rd Qu.:13.00   3rd Qu.:37.00   3rd Qu.:48.00   3rd Qu.:30.00  
```

```
##  Max.   :28.00   Max.   :55.00   Max.   :74.00   Max.   :50.00
##
##       stl              blk              tov              pf              pts
##  Min.   : 0.00   Min.   : 0.000   Min.   : 2.0   Min.   : 4.00   Min.   : 67
##  1st Qu.: 6.00   1st Qu.: 3.000   1st Qu.:10.0   1st Qu.:16.00   1st Qu.:105
##  Median : 8.00   Median : 5.000   Median :13.0   Median :18.00   Median :114
##  Mean   : 7.84   Mean   : 5.009   Mean   :13.2   Mean   :18.66   Mean   :114
##  3rd Qu.:10.00   3rd Qu.: 7.000   3rd Qu.:16.0   3rd Qu.:21.00   3rd Qu.:123
##  Max.   :22.00   Max.   :18.000   Max.   :31.0   Max.   :35.00   Max.   :162
##
##                team_name      team_id_2            season
##  Atlanta Hawks       : 164   Min.   :1.611e+09   2023-24:2460
##  Boston Celtics      : 164   1st Qu.:1.611e+09   2024-25:2460
##  Brooklyn Nets       : 164   Median :1.611e+09
##  Charlotte Hornets   : 164   Mean   :1.611e+09
##  Chicago Bulls       : 164   3rd Qu.:1.611e+09
##  Cleveland Cavaliers: 164   Max.   :1.611e+09
##  (Other)             :3936
##    opp_abbr                         opponent     opponent_id        home_away
##  Length:4920        Atlanta Hawks       : 164   Min.   :1.611e+09   Away:2460
##  Class :character   Boston Celtics      : 164   1st Qu.:1.611e+09   Home:2460
##  Mode  :character   Brooklyn Nets       : 164   Median :1.611e+09
##                     Charlotte Hornets   : 164   Mean   :1.611e+09
##                     Chicago Bulls       : 164   3rd Qu.:1.611e+09
##                     Cleveland Cavaliers: 164   Max.   :1.611e+09
##                     (Other)             :3936
##  prev_game_date         days_rest       back_to_back   arena_lat
##  Min.   :2023-10-24   Min.   :  1.000   0:4048       Min.   :25.78
##  1st Qu.:2024-01-17   1st Qu.:  2.000   1: 872       1st Qu.:33.76
##  Median :2024-04-13   Median :  2.000                Median :38.90
##  Mean   :2024-07-16   Mean   :  3.299                Mean   :37.32
##  3rd Qu.:2025-01-16   3rd Qu.:  2.000                3rd Qu.:41.50
##  Max.   :2025-04-11   Max.   :193.000                Max.   :45.53
##
##    arena_lon          prev_lat        prev_lon       travel_distance
##  Min.   :-122.67   Min.   :25.78   Min.   :-122.67   Min.   :   0.0
##  1st Qu.:-105.01   1st Qu.:33.76   1st Qu.:-105.01   1st Qu.:   0.0
##  Median : -88.98   Median :38.90   Median : -87.92   Median : 420.5
##  Mean   : -93.27   Mean   :37.32   Mean   : -93.26   Mean   : 525.1
##  3rd Qu.: -80.84   3rd Qu.:41.50   3rd Qu.: -80.84   3rd Qu.: 857.9
##  Max.   : -71.06   Max.   :45.53   Max.   : -71.06   Max.   :2707.4
##
##    plus_minus
##  Min.   :-62
##  1st Qu.:-10
##  Median :  0
##  Mean   :  0
##  3rd Qu.: 10
##  Max.   : 62
##
```

Feature Engineering

```
# Rest_days already computed
summary(nba$days_rest)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   3.299   2.000 193.000
```

```
# Back_to_back already computed
table(nba$back_to_back)
```

```
##
##    0    1
## 4048  872
```

```
#Travel_distance already computed
summary(nba$travel_distance)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     0.0   420.5   525.1   857.9  2707.4
```

```
# Home_Away convert to numeric
nba <- nba %>%
  mutate(home_indicator = if_else(home_away == "Home", 1, 0))

# Opponent_strength (opp win percentage to date)
nba <- nba %>%
  group_by(opponent) %>%
  arrange(game_date) %>%
  mutate(
    opponent_strength = lag(cumsum(w == 1) / row_number(), default = 0)
  ) %>%
  ungroup()

summary(nba$opponent_strength)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.02500 0.03750 0.06403 0.05556 1.00000
```

Outlier Detection

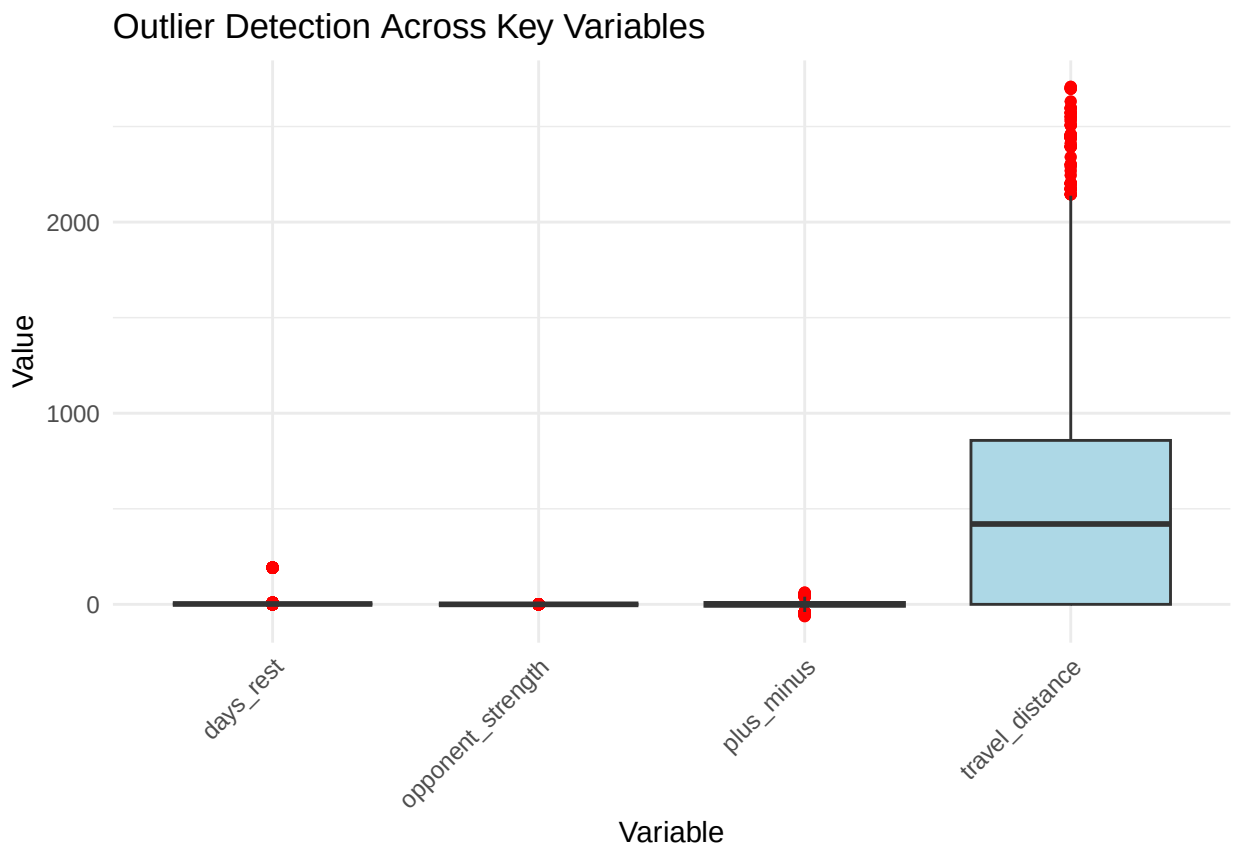
```
# IQR method
# travel_distance
Q1 <- quantile(nba$travel_distance, 0.25, na.rm = TRUE)
Q3 <- quantile(nba$travel_distance, 0.75, na.rm = TRUE)
IQR_val <- Q3 - Q1

upper_bound <- Q3 + 1.5 * IQR_val
```

```r
lower_bound <- Q1 - 1.5 * IQR_val

# days_rest
Q1_rest <- quantile(nba$days_rest, 0.25, na.rm = TRUE)
Q3_rest <- quantile(nba$days_rest, 0.75, na.rm = TRUE)
IQR_rest <- Q3_rest - Q1_rest

upper_rest <- Q3_rest + 1.5 * IQR_rest
lower_rest <- Q1_rest - 1.5 * IQR_rest

# plus_minus
Q1_pm <- quantile(nba$plus_minus, 0.25, na.rm = TRUE)
Q3_pm <- quantile(nba$plus_minus, 0.75, na.rm = TRUE)
IQR_pm <- Q3_pm - Q1_pm

upper_pm <- Q3_pm + 1.5 * IQR_pm
lower_pm <- Q1_pm - 1.5 * IQR_pm

# Opponent strength
Q1_str <- quantile(nba$opponent_strength, 0.25, na.rm = TRUE)
Q3_str <- quantile(nba$opponent_strength, 0.75, na.rm = TRUE)
IQR_str <- Q3_str - Q1_str

upper_str <- Q3_str + 1.5 * IQR_str
lower_str <- Q1_str - 1.5 * IQR_str

# Table summary
outlier_summary <- tibble(
  variable = c("travel_distance", "days_rest", "plus_minus", "opponent_strength"),
  lower_bound = c(lower_bound, lower_rest, lower_pm, lower_str),
  upper_bound = c(upper_bound, upper_rest, upper_pm, upper_str)
)

outlier_summary
```

```
## # A tibble: 4 x 3
##   variable          lower_bound upper_bound
##   <chr>                   <dbl>       <dbl>
## 1 travel_distance    -1287.        2145.
## 2 days_rest              2            2
## 3 plus_minus           -40           40
## 4 opponent_strength     -0.0208       0.101
```

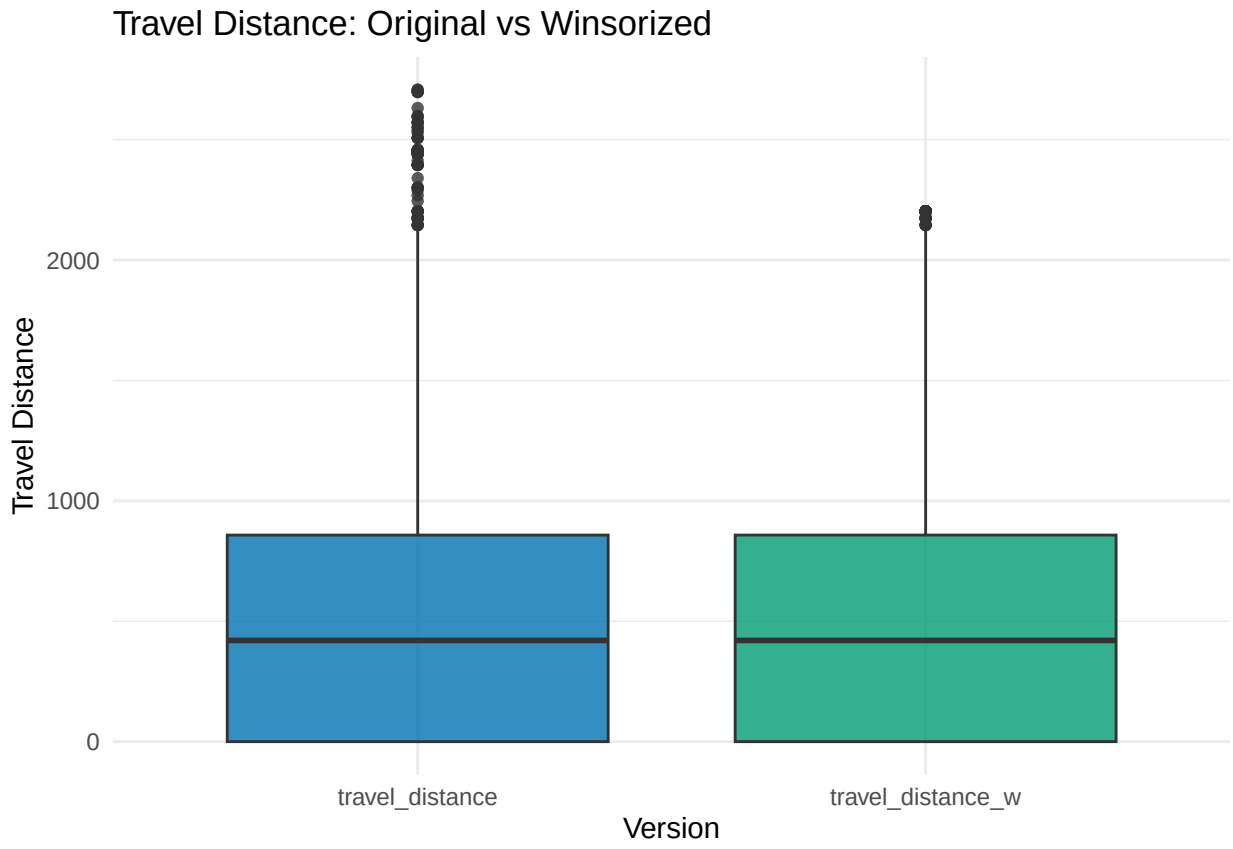
```r
# Visualize the outliers
library(tidyr)
library(ggplot2)

plot_data <- nba %>%
  select(travel_distance, days_rest, plus_minus, opponent_strength) %>%
  pivot_longer(cols = everything(), names_to = "variable", values_to = "value")

ggplot(plot_data, aes(x = variable, y = value)) +
  geom_boxplot(outlier.color = "red", fill = "lightblue") +
```

```
  theme_minimal() +
  labs(title = "Outlier Detection Across Key Variables",
       x = "Variable", y = "Value") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Handling Outliers by Winsorization
winsorize_manual <- function(x, p = c(0.01, 0.99)) {
  qnt <- quantile(x, probs = p, na.rm = TRUE)
  x[x < qnt[1]] <- qnt[1]
  x[x > qnt[2]] <- qnt[2]
  return(x)
}

nba <- nba %>%
  mutate(
    travel_distance_w = winsorize_manual(travel_distance)
  )

library(tidyr)

plot_df <- nba %>%
  select(travel_distance, travel_distance_w) %>%
  pivot_longer(cols = everything(), names_to = "type", values_to = "value")

ggplot(plot_df, aes(x = type, y = value)) +
```

```
  geom_boxplot(fill = c("#0072B2", "#009E73"), alpha = 0.8) +
  labs(
    title = "Travel Distance: Original vs Winsorized",
    x = "Version",
    y = "Travel Distance"
  ) +
  theme_minimal()
```



Normalization and Scaling

```
nba <- nba %>%
  mutate(
    scale_travel  = scale(travel_distance),
    scale_rest    = scale(days_rest),
    scale_plus    = scale(plus_minus),
    scale_opp_str = scale(opponent_strength)
  )

## View final cleaned dataset
#glimpse(nba)
```

```r
nba <- nba %>%
  mutate(
    back_to_back = as.character(back_to_back),
    back_to_back_num = ifelse(back_to_back == "1", 1L, 0L),
    travel_distance_w = ifelse(is.na(travel_distance_w), travel_distance, travel_distance_w)
  )
```

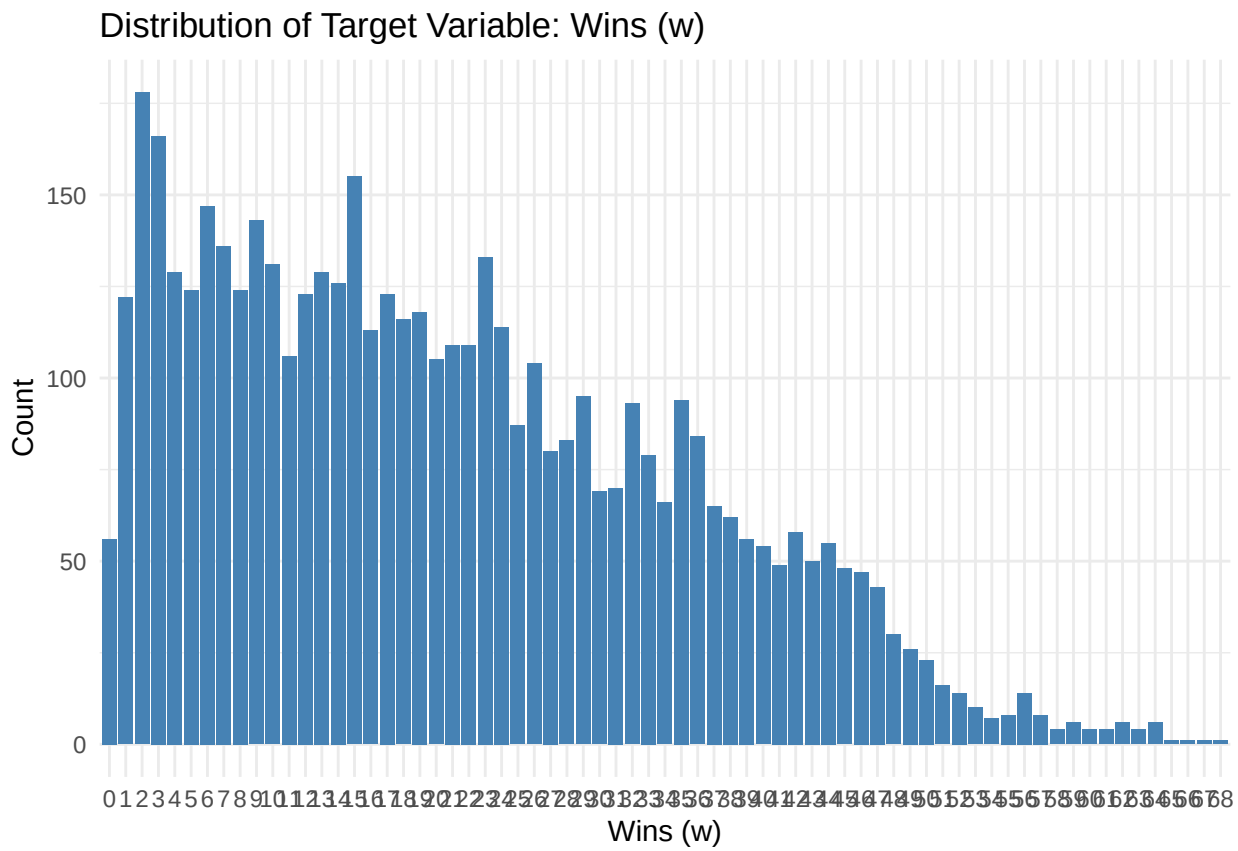
```r
# Check class distribution for the target variable 'w'

table(nba$wl)
```

```
##
##    L    W
## 2460 2460
```

```r
prop.table(table(nba$wl))
```

```
##
##   L   W
## 0.5 0.5
```

```r
# Visual bar plot
library(ggplot2)

ggplot(nba, aes(x = factor(w))) +
  geom_bar(fill = "steelblue") +
  labs(title = "Distribution of Target Variable: Wins (w)",
       x = "Wins (w)",
       y = "Count") +
  theme_minimal()
```

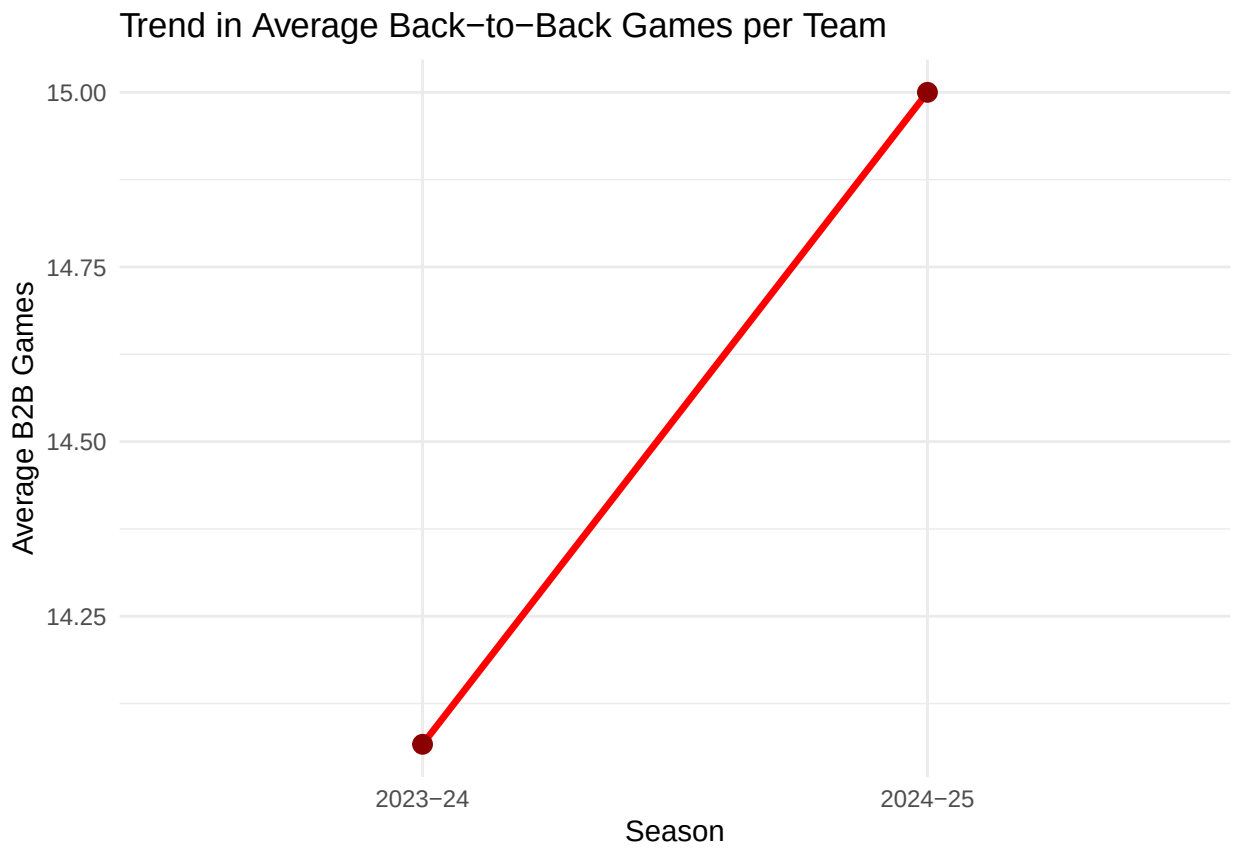


EDA

Trend over Time: B2B Games per Season

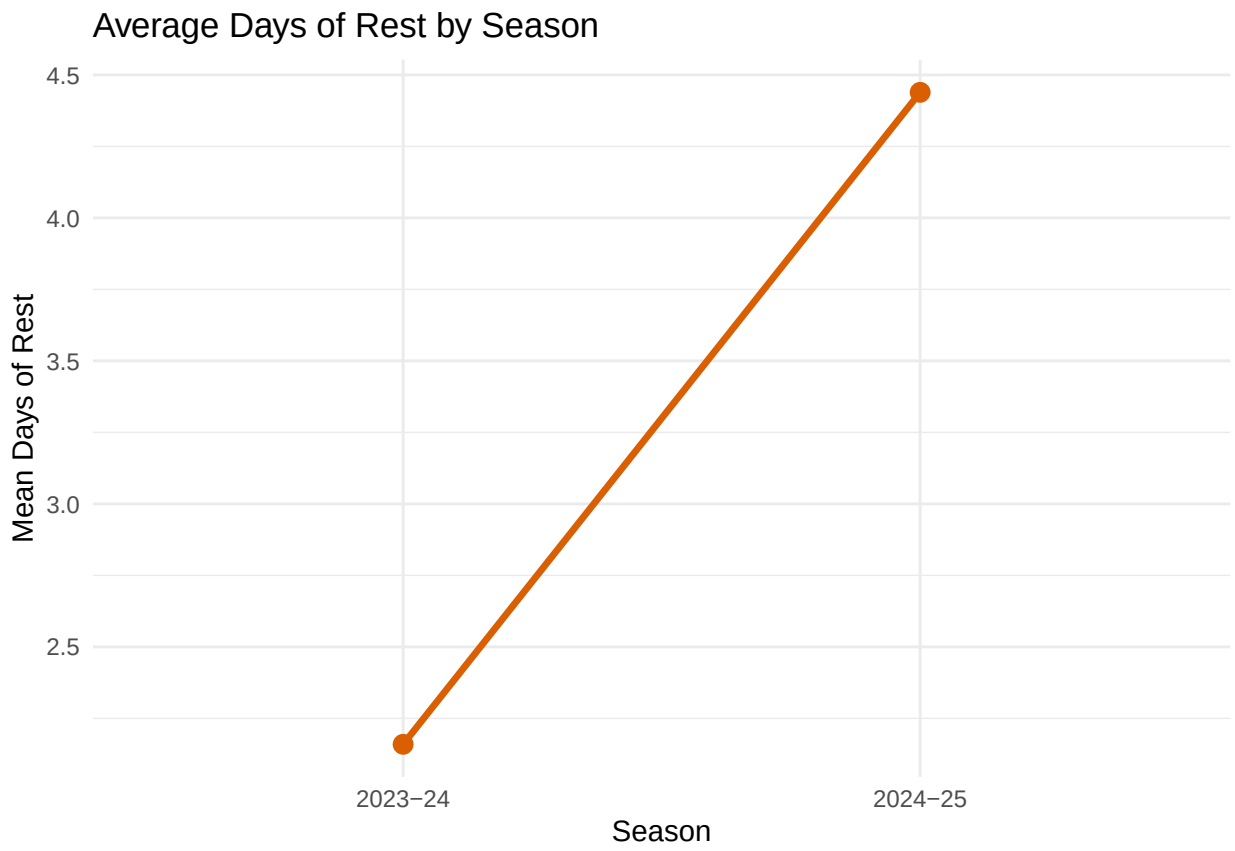
NBA scheduling has reduced B2B games over time to improve player rest.

```
library(dplyr)
library(ggplot2)

b2b_trend <- nba %>%
  group_by(season, team_name) %>%
  summarise(total_b2b = sum(back_to_back_num, na.rm = TRUE), .groups = "drop") %>%
  group_by(season) %>%
  summarise(avg_b2b = base::mean(total_b2b, na.rm = TRUE), .groups = "drop")

ggplot(b2b_trend, aes(season, avg_b2b, group = 1)) +
  geom_line(color = "red", linewidth = 1.2) +
  geom_point(color = "darkred", size = 3) +
  theme_minimal() +
  labs(title = "Trend in Average Back-to-Back Games per Team",
       x = "Season", y = "Average B2B Games")
```



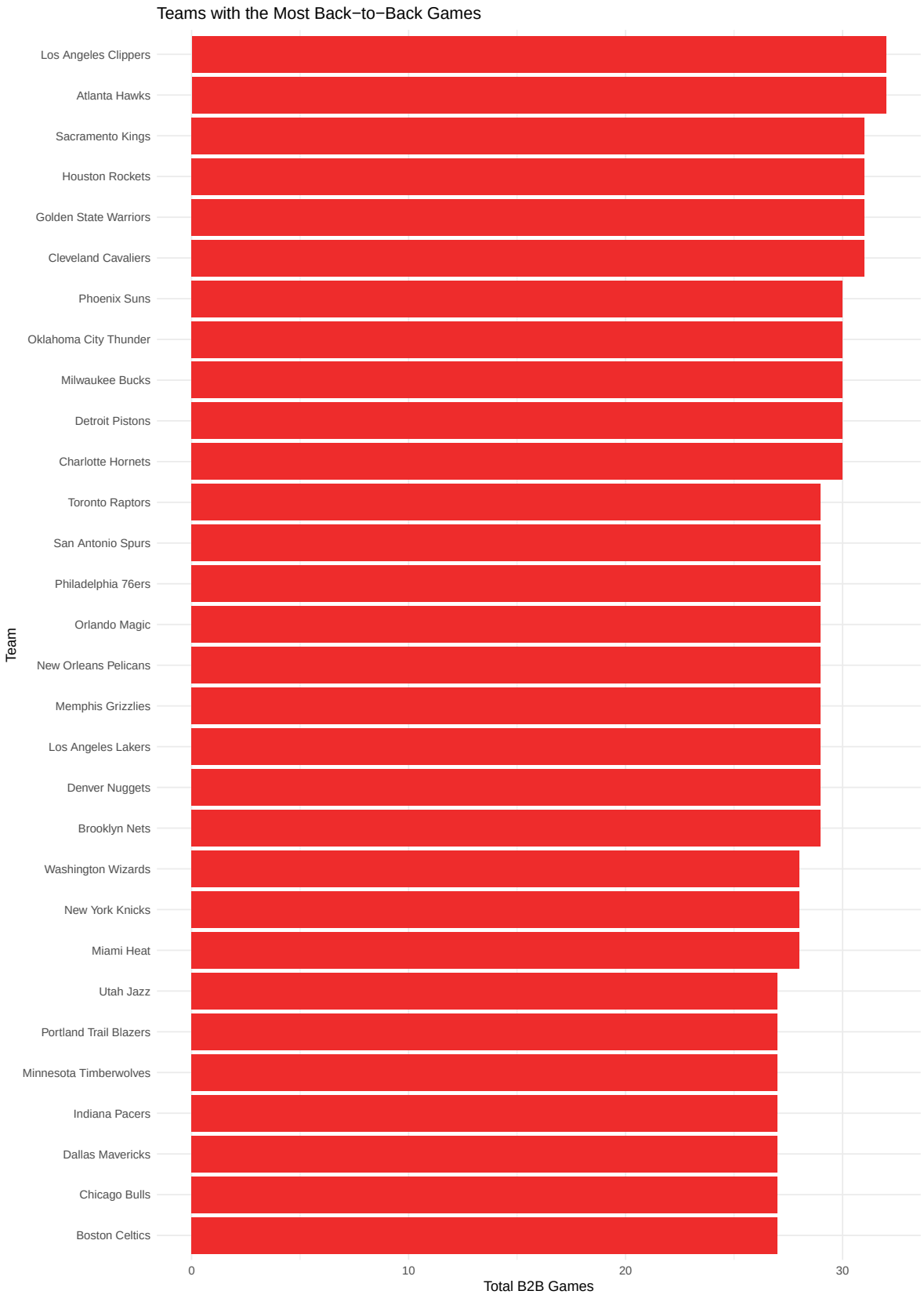
Average rest days by season

```
rest_trend <- nba %>%
  group_by(season) %>%
  summarise(avg_rest = base::mean(days_rest, na.rm = TRUE), .groups = "drop")

ggplot(rest_trend, aes(season, avg_rest, group = 1)) +
  geom_line(color = "#d95f02", linewidth = 1.2) +
  geom_point(color = "#d95f02", size = 3) +
  theme_minimal() +
  labs(title = "Average Days of Rest by Season",
       x = "Season", y = "Mean Days of Rest")
```

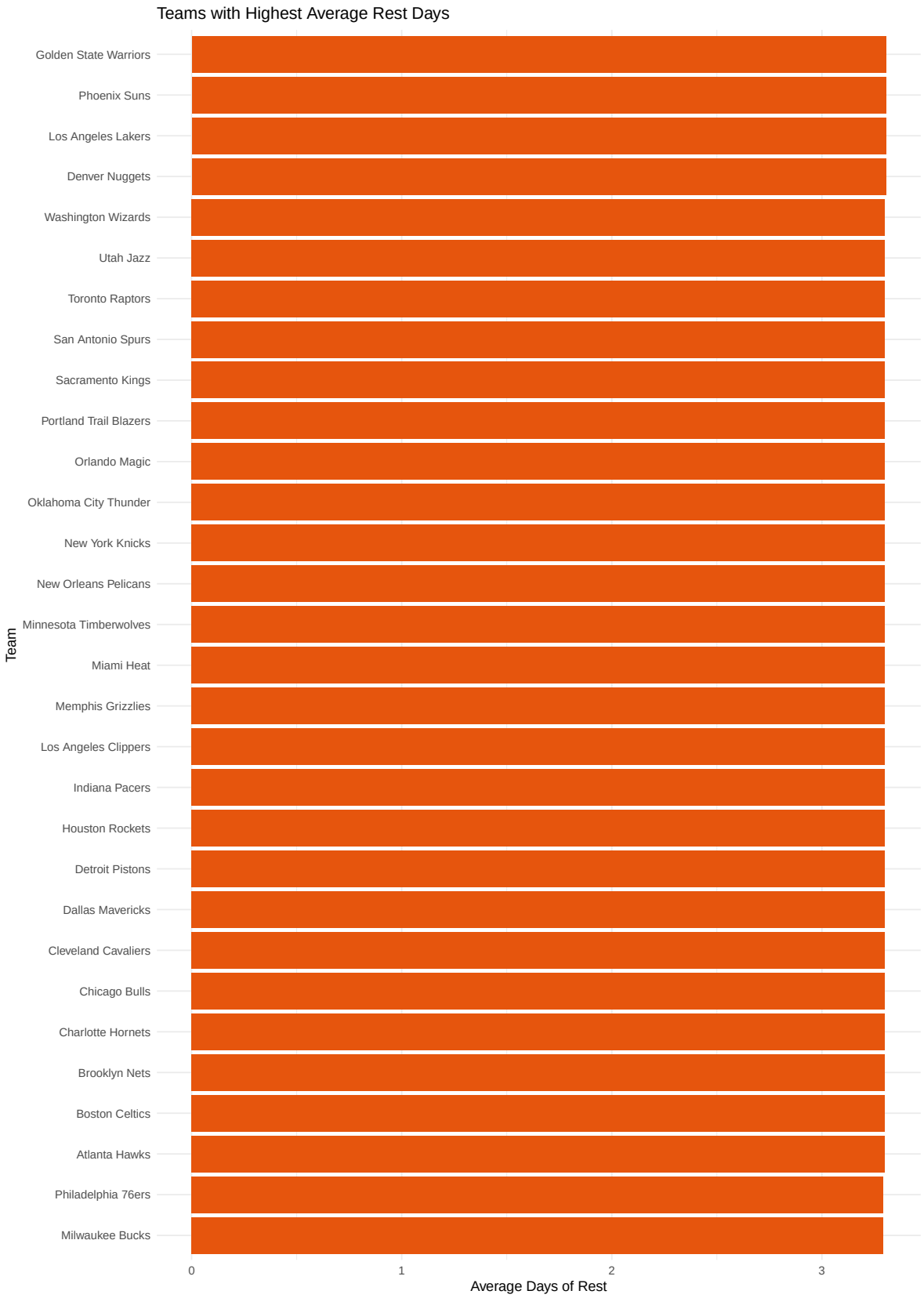


By team: Most B2B and most rest

```
# (A) Teams with the most B2B games (all seasons combined)
b2b_team <- nba %>%
  group_by(team_name) %>%
  summarise(total_b2b = sum(back_to_back_num, na.rm = TRUE), .groups = "drop") %>%
  arrange(desc(total_b2b))

ggplot(b2b_team, aes(x = reorder(team_name, total_b2b), y = total_b2b)) +
  geom_col(fill = "firebrick2") +
  coord_flip() +
  theme_minimal() +
  labs(title = "Teams with the Most Back-to-Back Games",
       x = "Team", y = "Total B2B Games")
```


Teams with the Most Back−to−Back Games
Los Angeles Clippers
Atlanta Hawks
Sacramento Kings
Houston Rockets
Golden State Warriors
Cleveland Cavaliers
Phoenix Suns
Oklahoma City Thunder
Milwaukee Bucks
Detroit Pistons
Charlotte Hornets
Toronto Raptors
San Antonio Spurs
Philadelphia 76ers
Orlando Magic
New Orleans Pelicans
Memphis Grizzlies
Los Angeles Lakers
Denver Nuggets
Brooklyn Nets
Washington Wizards
New York Knicks
Miami Heat
Utah Jazz
Portland Trail Blazers
Minnesota Timberwolves
Indiana Pacers
Dallas Mavericks
Chicago Bulls
Boston Celtics
Team
0
10
20
30
Total B2B Games

```r
# (B) Teams with the highest average rest
rest_team <- nba %>%
  group_by(team_name) %>%
  summarise(avg_rest = base::mean(days_rest, na.rm = TRUE), .groups = "drop") %>%
  arrange(desc(avg_rest))

ggplot(rest_team, aes(x = reorder(team_name, avg_rest), y = avg_rest)) +
  geom_col(fill = "#e6550d") +
  coord_flip() +
  theme_minimal() +
  labs(title = "Teams with Highest Average Rest Days",
       x = "Team", y = "Average Days of Rest")
```


Teams with Highest Average Rest Days
Golden State Warriors
Phoenix Suns
Los Angeles Lakers
Denver Nuggets
Washington Wizards
Utah Jazz
Toronto Raptors
San Antonio Spurs
Sacramento Kings
Portland Trail Blazers
Orlando Magic
Oklahoma City Thunder
New York Knicks
New Orleans Pelicans
Minnesota Timberwolves
Miami Heat
Memphis Grizzlies
Los Angeles Clippers
Indiana Pacers
Houston Rockets
Detroit Pistons
Dallas Mavericks
Cleveland Cavaliers
Chicago Bulls
Charlotte Hornets
Brooklyn Nets
Boston Celtics
Atlanta Hawks
Philadelphia 76ers
Milwaukee Bucks
Team
0
1
2
3
Average Days of Rest

Interactive full-season schedule viewer

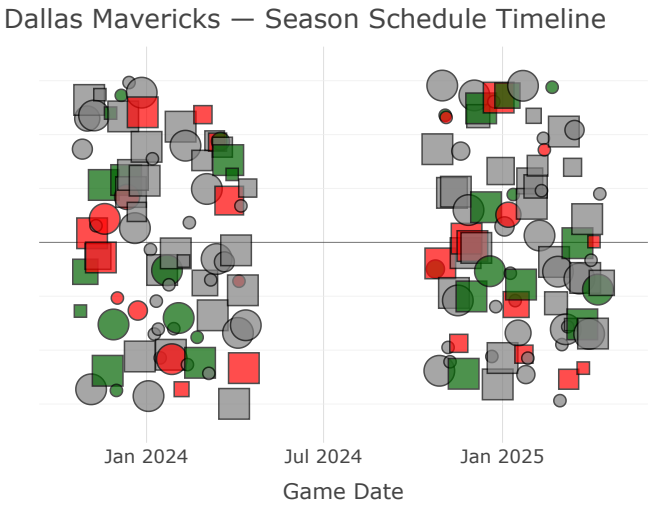
```
library(dplyr)
library(plotly)
```

```
##
## 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
```

```
team_schedule <- function(team) {

  df <- nba %>%
    filter(team_name == team) %>%
    arrange(game_date) %>%
    mutate(
      back_to_back_num = case_when(
        as.character(back_to_back) == "1" ~ 1L,
        TRUE ~ 0L
      ),
      travel_distance_w = ifelse(is.na(travel_distance_w),
                                 travel_distance,
                                 travel_distance_w),
      # jitter y-position so all points aren't on a single line
      y_jitter = runif(n(), -0.3, 0.3),

      # color rules
      color_code = case_when(
        back_to_back_num == 1 ~ "red",
        days_rest >= 3 ~ "darkgreen",
        TRUE ~ "grey"
      ),

      # shape: home/away
      marker_shape = ifelse(home_away == "Home", "circle", "square")
    )

  plot_ly(
    data = df,
    x = ~game_date,
    y = ~y_jitter,
```

```
    type = "scatter",
    mode = "markers",
    marker = list(
      size = ~pmin(travel_distance_w / 50 + 8, 20),   # travel scaling
      color = df$color_code,
      symbol = ~marker_shape,
      line = list(width = 1, color = "black")
    ),
    text = ~paste0(
      "<b>Date:</b> ", game_date,
      "<br><b>Opponent:</b> ", opponent,
      "<br><b>Home/Away:</b> ", home_away,
      "<br><b>Days Rest:</b> ", days_rest,
      "<br><b>Travel (mi):</b> ", round(travel_distance_w, 0),
      "<br><b>B2B:</b> ", ifelse(back_to_back_num == 1, "Yes", "No")
    ),
    hoverinfo = "text"
  ) %>%
    layout(
      title = paste0(team, " - Season Schedule Timeline"),
      xaxis = list(
        title = "Game Date",
        showgrid = TRUE,
        gridcolor = "lightgrey"
      ),
      yaxis = list(
        title = "",
        showticklabels = FALSE
      ),
      legend = list(orientation = "h")
    )
}

# Example:
team_schedule("Dallas Mavericks")
```

```
## file:////private/var/folders/2x/m3nyg7x50mzg_7wsnzcm7hxr0000gn/T/Rtmp7CdIv4/file11453366e15f1/widget
```


Dallas Mavericks — Season Schedule Timeline
Jan 2024
Jul 2024
Jan 2025
Game Date

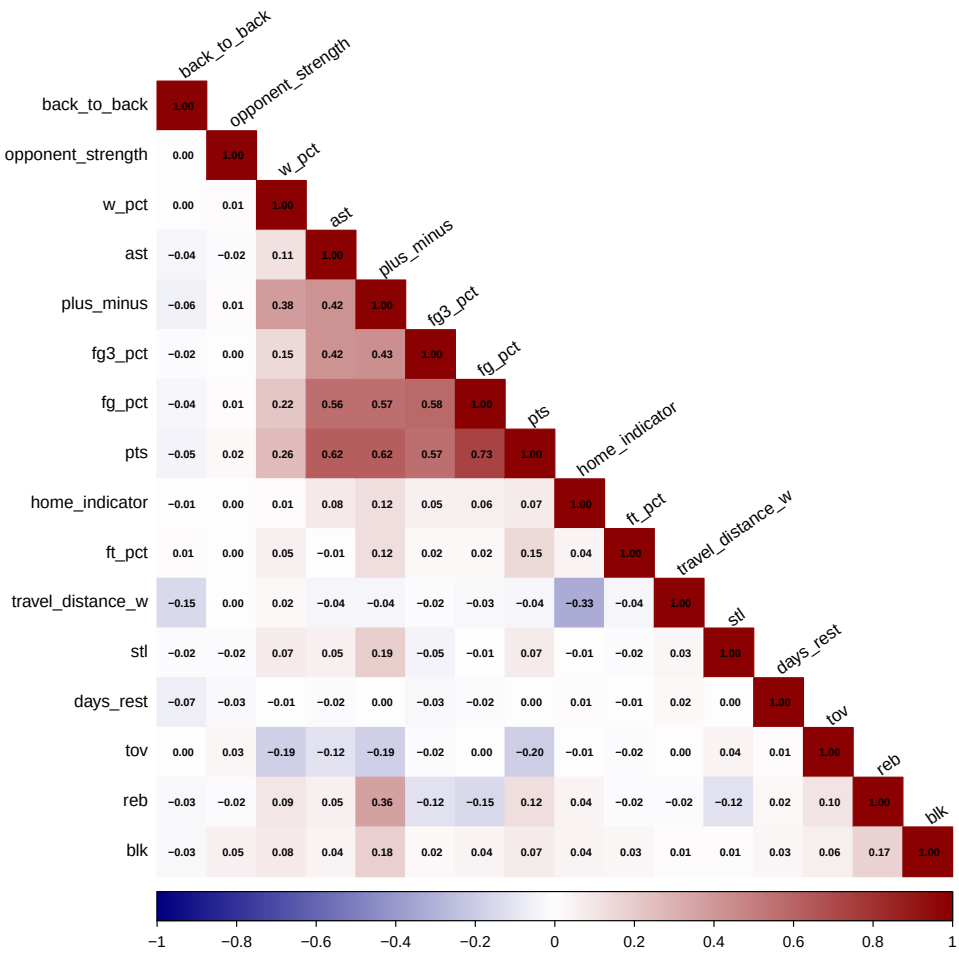
Correlation Heatmap

```
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
# ---- Select only important model features ----
important_vars <- nba %>%
  select(
    w_pct,                      # target
    travel_distance_w,          # travel
    days_rest,
    back_to_back,
    home_indicator,
    opponent_strength,          # difficulty factor
    fg_pct, fg3_pct, ft_pct,    # shooting efficiency
    reb, ast, stl, blk, tov,    # core box score stats
    plus_minus, pts             # scoring metrics
  )

# --- Ensure all variables are numeric ---
important_vars <- important_vars %>%
  mutate(across(everything(), as.numeric))

# --- Compute correlation matrix ---
cor_mat <- cor(important_vars, use = "pairwise.complete.obs")
```

```
corrplot(
  round(cor_mat, 2),
  method = "color",
  type = "lower",
  order = "hclust",
  tl.col = "black",
  tl.srt = 35,
  tl.cex = 0.9,
  number.cex = 0.55,
  number.font = 2,
  addCoef.col = "black",
  mar = c(1,1,1,1),
  cl.cex = 0.8,
  col = colorRampPalette(c("navy", "white", "darkred"))(200)
)
```



Modeling

```
# Prepare data for modeling
nba_glm <- nba %>%
  mutate(
    win = ifelse(wl == "W", 1, 0),
    back_to_back_num = ifelse(back_to_back == "1", 1, 0),
    travel_distance_w = ifelse(is.na(travel_distance_w), travel_distance, travel_distance_w),
    home_indicator = ifelse(home_away == "Home", 1, 0)
  ) %>%
  select(
    win,
    # Schedule factors
    travel_distance_w,
    days_rest,
```

```
    back_to_back_num,
    home_indicator,
    opponent_strength,
    # ADD safe performance variables (do NOT leak outcome)
    fg_pct,
    fg3_pct,
    ast,
    reb,
    tov
  ) %>%
  tidyr::drop_na()
```

Logistic Regression Model

```
full_glm <- glm(
  win ~ travel_distance_w + days_rest + back_to_back_num +
        home_indicator + opponent_strength +
        fg_pct + fg3_pct + ast + reb + tov,
  data = nba_glm,
  family = binomial(link = "logit")
)

summary(full_glm)
```

```
##
## Call:
## glm(formula = win ~ travel_distance_w + days_rest + back_to_back_num +
##     home_indicator + opponent_strength + fg_pct + fg3_pct + ast +
##     reb + tov, family = binomial(link = "logit"), data = nba_glm)
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -2.396e+01  7.351e-01 -32.597  < 2e-16 ***
## travel_distance_w  5.539e-05  7.705e-05   0.719 0.472219
## days_rest          1.215e-03  2.486e-03   0.489 0.624959
## back_to_back_num  -1.133e-01  1.023e-01  -1.108 0.267958
## home_indicator     3.038e-01  8.109e-02   3.746 0.000180 ***
## opponent_strength  4.938e-01  3.391e-01   1.456 0.145381
## fg_pct             3.127e+01  1.201e+00  26.035  < 2e-16 ***
## fg3_pct            6.643e+00  6.069e-01  10.946  < 2e-16 ***
## ast               -3.524e-02  9.365e-03  -3.763 0.000168 ***
## reb                2.233e-01  7.970e-03  28.023  < 2e-16 ***
## tov               -1.707e-01  1.074e-02 -15.901  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6820.6  on 4919  degrees of freedom
## Residual deviance: 4188.3  on 4909  degrees of freedom
```

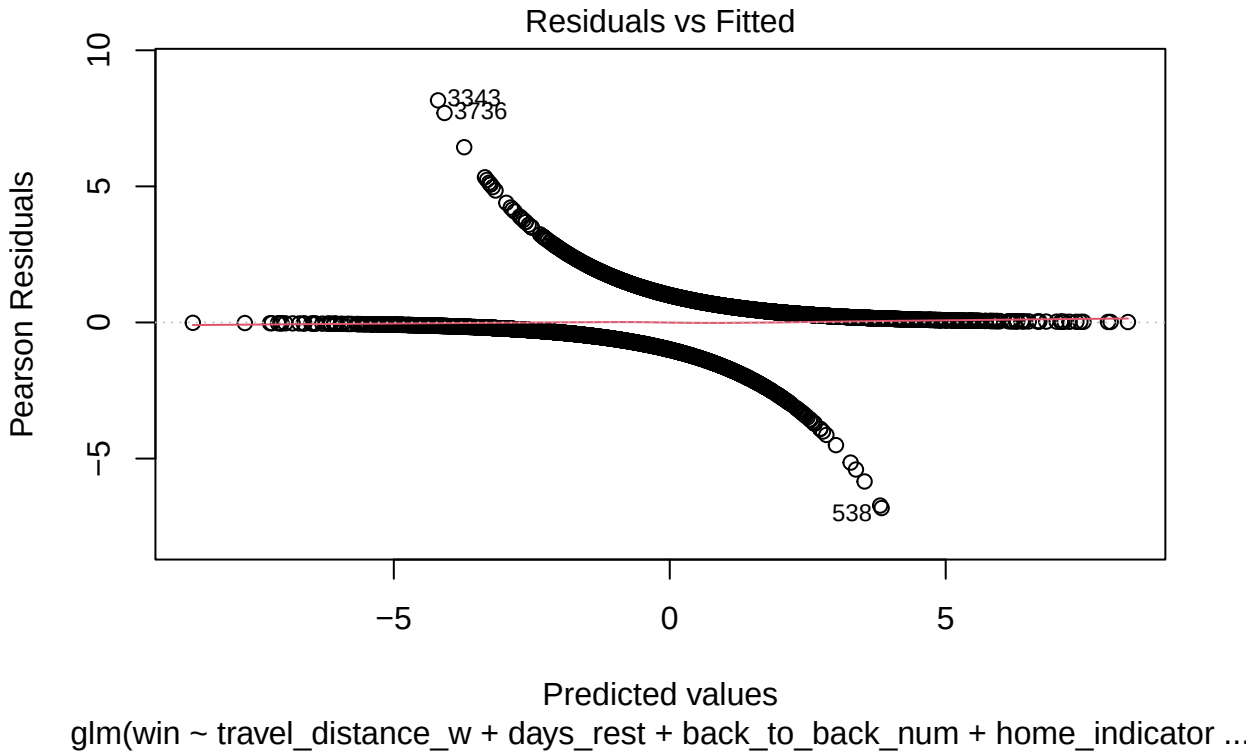
```
## AIC: 4210.3
##
## Number of Fisher Scoring iterations: 5
```

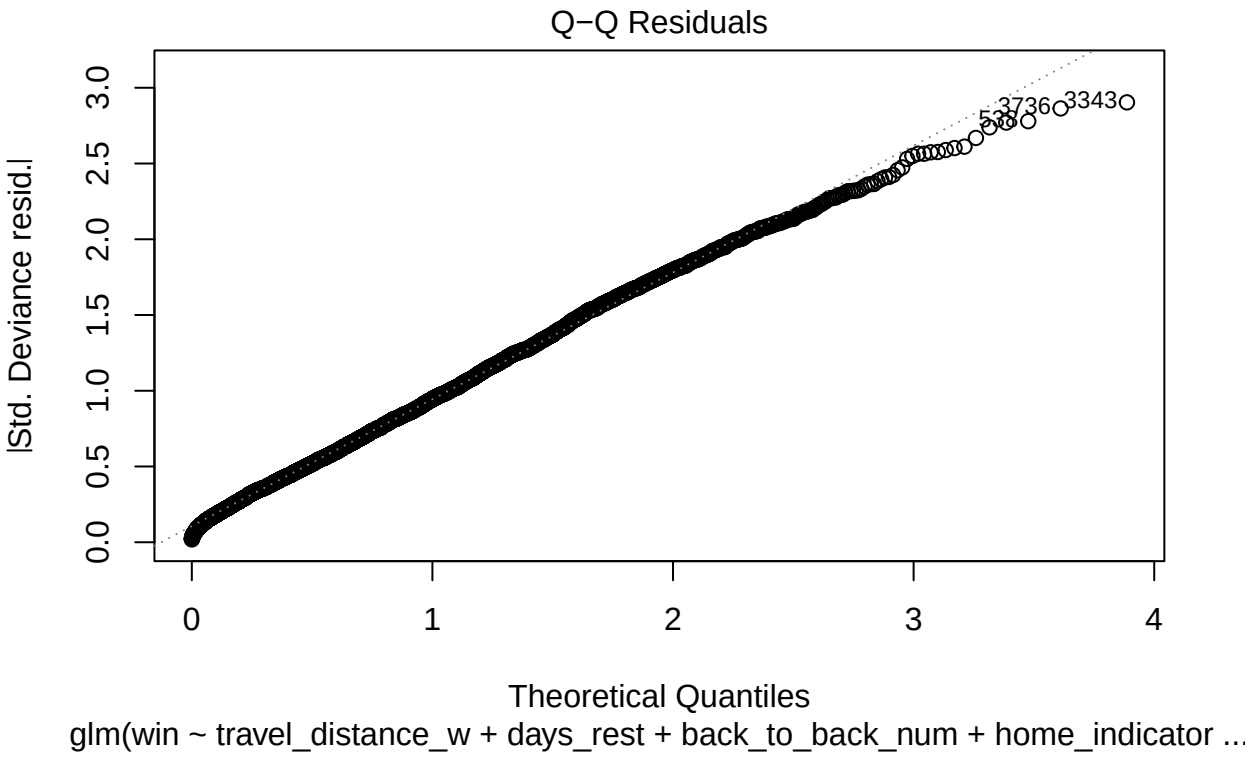
```
ll_null <- logLik(glm(win ~ 1, data = nba_glm, family = binomial))
ll_full <- logLik(full_glm)

# McFadden R2
R2 <- 1 - (as.numeric(ll_full) / as.numeric(ll_null))
R2
```

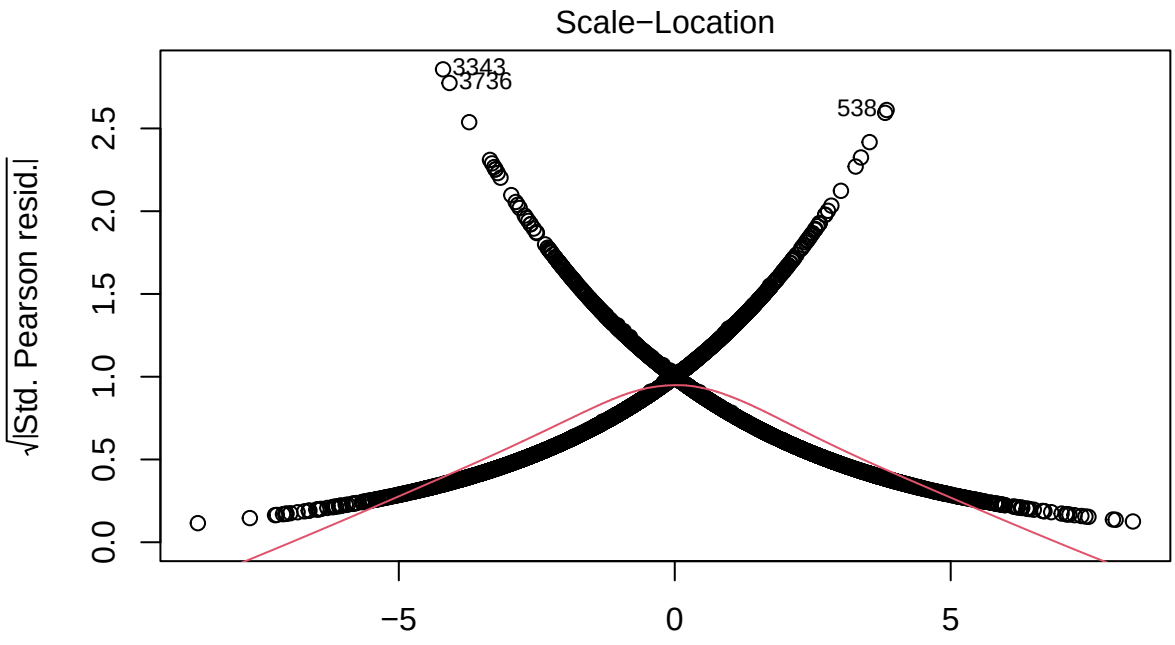
```
## [1] 0.38593
```

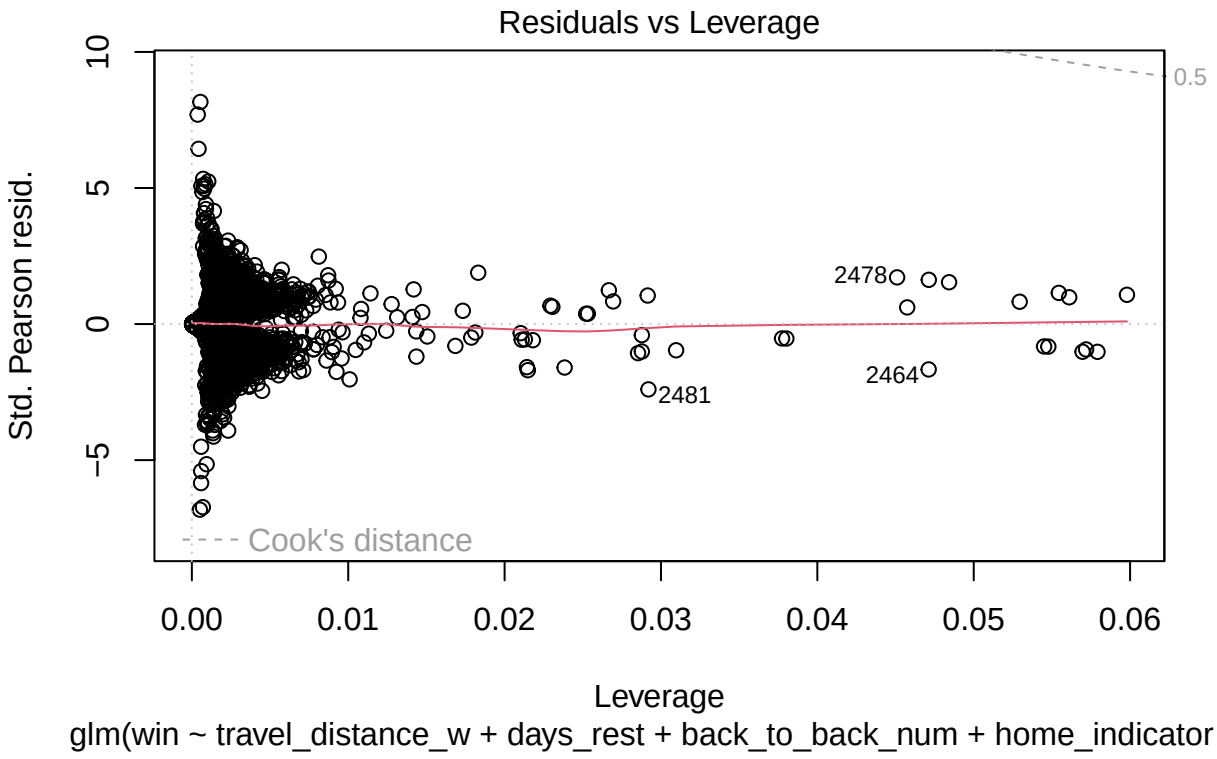
Marginal Plots

```
plot(full_glm)
```



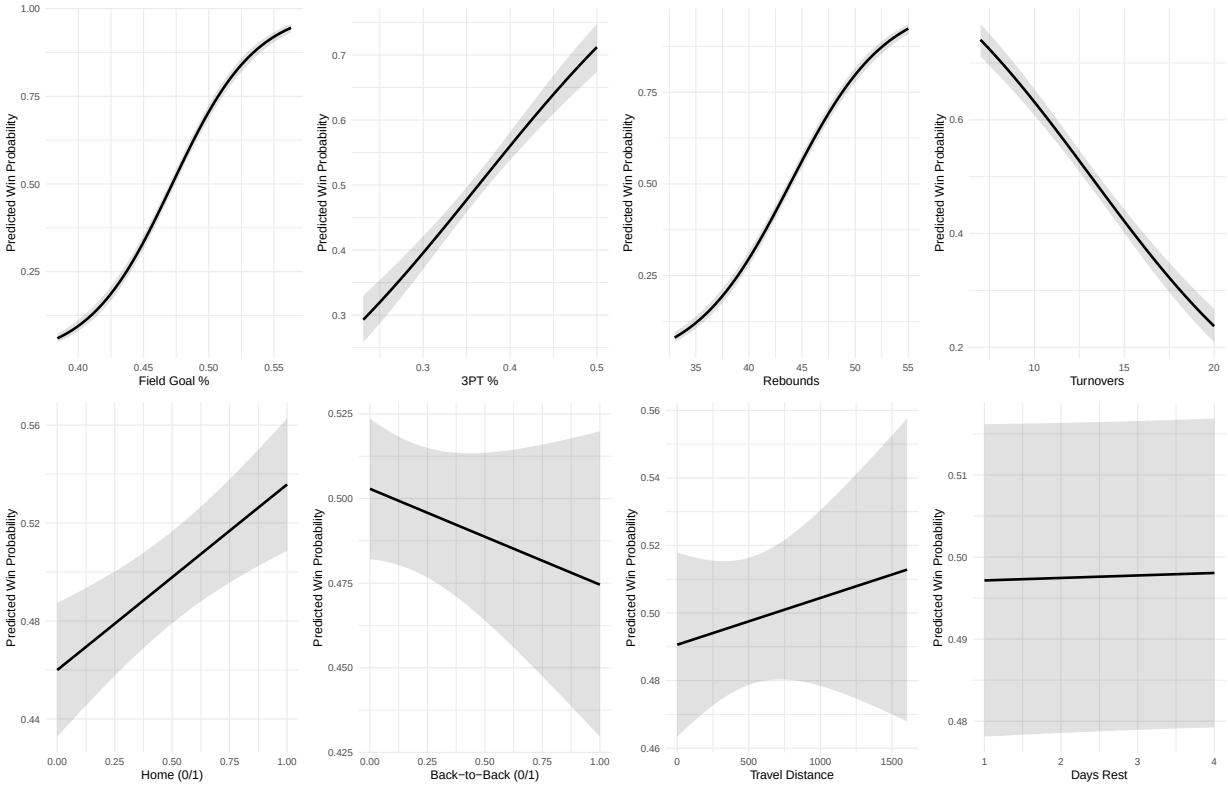

Q−Q Residuals
|Std. Deviance resid.|
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
1
2
3
4
3736
3343
Theoretical Quantiles
glm(win ~ travel_distance_w + days_rest + back_to_back_num + home_indicator ...


Scale−Location
3343
3736
538
√|Std. Pearson resid.|
0.0
0.5
1.0
1.5
2.0
2.5
−5
0
5
Predicted values
glm(win ~ travel_distance_w + days_rest + back_to_back_num + home_indicator ...




```r
library(ggplot2)
library(patchwork)

df <- model.frame(full_glm)

make_typical <- function(df) {
  out <- df[1, , drop = FALSE]
  for (nm in names(df)) {
    if (is.numeric(df[[nm]])) {
      out[[nm]] <- base::mean(df[[nm]], na.rm = TRUE)
    } else {
      out[[nm]] <- levels(df[[nm]])[1]
    }
  }
  out
}

marginal_plot <- function(model, df, var, label = var, n = 100) {
  base_row <- make_typical(df)

  if (is.numeric(df[[var]])) {
    lo <- stats::quantile(df[[var]], 0.05, na.rm = TRUE)
    hi <- stats::quantile(df[[var]], 0.95, na.rm = TRUE)
    grid <- base_row[rep(1, n), , drop = FALSE]
    grid[[var]] <- seq(lo, hi, length.out = n)
  } else {
```

```
    levs <- levels(factor(df[[var]]))
    grid <- do.call(rbind, lapply(levs, function(L) {
      r <- base_row
      r[[var]] <- factor(L, levels = levs)
      r
    }))
  }

  p <- suppressWarnings(predict(model, newdata = grid, type = "link", se.fit = TRUE))
  grid$fit <- plogis(p$fit)
  grid$lwr <- plogis(p$fit - 1.96 * p$se.fit)
  grid$upr <- plogis(p$fit + 1.96 * p$se.fit)

  g <- if (is.numeric(df[[var]])) {
    ggplot(grid, aes(x = .data[[var]], y = fit)) +
      geom_ribbon(aes(ymin = lwr, ymax = upr), alpha = 0.15) +
      geom_line(linewidth = 1)
  } else {
    ggplot(grid, aes(x = .data[[var]], y = fit)) +
      geom_point(size = 3) +
      geom_errorbar(aes(ymin = lwr, ymax = upr), width = 0.15)
  }

  g +
    labs(x = label, y = "Predicted Win Probability") +
    theme_minimal(base_size = 10) +
    theme(
      plot.title = element_blank(),  # ensure no title
      plot.margin = margin(6, 6, 6, 6)
    )
}

p1 <- marginal_plot(full_glm, df, "fg_pct",            "Field Goal %")
p2 <- marginal_plot(full_glm, df, "fg3_pct",           "3PT %")
p3 <- marginal_plot(full_glm, df, "reb",               "Rebounds")
p4 <- marginal_plot(full_glm, df, "tov",               "Turnovers")
p5 <- marginal_plot(full_glm, df, "home_indicator",    "Home (0/1)")
p6 <- marginal_plot(full_glm, df, "back_to_back_num",  "Back-to-Back (0/1)")
p7 <- marginal_plot(full_glm, df, "travel_distance_w", "Travel Distance")
p8 <- marginal_plot(full_glm, df, "days_rest",         "Days Rest")

# Combine 2×4 with no global annotation
(p1 | p2 | p3 | p4) /
(p5 | p6 | p7 | p8)
```



BIC, AIC

```
# Backward elimination (step using AIC; start from full model)
# (uses base step() which by default uses AIC; change k=log(n) for BIC)
backward_aic <- step(full_glm, direction = "backward", trace = 1)
```

```
## Start:  AIC=4210.31
## win ~ travel_distance_w + days_rest + back_to_back_num + home_indicator +
##     opponent_strength + fg_pct + fg3_pct + ast + reb + tov
##
##                     Df Deviance    AIC
## - days_rest          1   4188.5 4208.5
## - travel_distance_w  1   4188.8 4208.8
## - back_to_back_num   1   4189.5 4209.5
## <none>                   4188.3 4210.3
## - opponent_strength  1   4190.4 4210.4
## - home_indicator     1   4202.4 4222.4
## - ast                1   4202.6 4222.6
## - fg3_pct            1   4314.7 4334.7
## - tov                1   4470.4 4490.4
## - fg_pct             1   5122.0 5142.0
## - reb                1   5365.9 5385.9
##
## Step:  AIC=4208.54
## win ~ travel_distance_w + back_to_back_num + home_indicator +
##     opponent_strength + fg_pct + fg3_pct + ast + reb + tov
##
##                     Df Deviance    AIC
```

```
## - travel_distance_w  1   4189.1 4207.1
## - back_to_back_num   1   4189.9 4207.9
## <none>                    4188.5 4208.5
## - opponent_strength  1   4190.6 4208.6
## - home_indicator     1   4202.6 4220.6
## - ast                1   4202.9 4220.9
## - fg3_pct            1   4314.7 4332.7
## - tov                1   4470.5 4488.5
## - fg_pct             1   5122.2 5140.2
## - reb                1   5366.6 5384.6
##
## Step:  AIC=4207.07
## win ~ back_to_back_num + home_indicator + opponent_strength +
##     fg_pct + fg3_pct + ast + reb + tov
##
##                     Df Deviance    AIC
## - back_to_back_num   1   4190.7 4206.7
## <none>                    4189.1 4207.1
## - opponent_strength  1   4191.2 4207.2
## - home_indicator     1   4202.9 4218.9
## - ast                1   4203.4 4219.4
## - fg3_pct            1   4315.1 4331.1
## - tov                1   4471.2 4487.2
## - fg_pct             1   5122.3 5138.3
## - reb                1   5366.6 5382.6
##
## Step:  AIC=4206.71
## win ~ home_indicator + opponent_strength + fg_pct + fg3_pct +
##     ast + reb + tov
##
##                     Df Deviance    AIC
## <none>                    4190.7 4206.7
## - opponent_strength  1   4192.8 4206.8
## - home_indicator     1   4204.5 4218.5
## - ast                1   4205.0 4219.0
## - fg3_pct            1   4316.5 4330.5
## - tov                1   4473.1 4487.1
## - fg_pct             1   5129.1 5143.1
## - reb                1   5372.9 5386.9
```

```
summary(backward_aic)
```

```
##
## Call:
## glm(formula = win ~ home_indicator + opponent_strength + fg_pct +
##     fg3_pct + ast + reb + tov, family = binomial(link = "logit"),
##     data = nba_glm)
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -23.957177   0.729446 -32.843  < 2e-16 ***
## home_indicator      0.285529   0.076868   3.715 0.000204 ***
## opponent_strength   0.491201   0.338708   1.450 0.146997
## fg_pct             31.313648   1.200284  26.089  < 2e-16 ***
```

```
## fg3_pct             6.622838   0.606258  10.924  < 2e-16 ***
## ast                -0.035225   0.009355  -3.765 0.000166 ***
## reb                 0.223482   0.007962  28.068  < 2e-16 ***
## tov                -0.170740   0.010732 -15.910  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6820.6  on 4919  degrees of freedom
## Residual deviance: 4190.7  on 4912  degrees of freedom
## AIC: 4206.7
##
## Number of Fisher Scoring iterations: 5
```

```
# --- Forward selection (start from intercept; use AIC) ---
null_mod <- glm(win~ 1, data = nba_glm)  # start with intercept only
scope_formula <- formula(full_glm)

forward_aic <- step(null_mod, scope = scope_formula, direction = "forward", trace = 1)
```

```
## Start:  AIC=7145.79
## win ~ 1
##
##                     Df Deviance    AIC
## + fg_pct             1   958.96 5923.1
## + fg3_pct            1  1078.37 6500.5
## + ast                1  1106.15 6625.6
## + reb                1  1122.69 6698.6
## + tov                1  1204.28 7043.8
## + home_indicator     1  1220.69 7110.4
## + back_to_back_num   1  1226.65 7134.4
## + travel_distance_w  1  1229.00 7143.8
## <none>                  1230.00 7145.8
## + opponent_strength  1  1229.95 7147.6
## + days_rest          1  1229.99 7147.7
##
## Step:  AIC=5923.11
## win ~ fg_pct
##
##                     Df Deviance    AIC
## + reb                1   789.03 4965.5
## + tov                1   933.78 5794.2
## + fg3_pct            1   947.04 5863.6
## + ast                1   953.75 5898.3
## + home_indicator     1   954.59 5902.6
## + back_to_back_num   1   957.76 5919.0
## <none>                   958.96 5923.1
## + travel_distance_w  1   958.73 5923.9
## + days_rest          1   958.90 5924.8
## + opponent_strength  1   958.96 5925.1
##
## Step:  AIC=4965.47
## win ~ fg_pct + reb
```

```
##
##                     Df Deviance    AIC
## + tov                1   748.19 4706.0
## + fg3_pct            1   773.30 4868.4
## + home_indicator     1   787.11 4955.5
## + back_to_back_num   1   788.64 4965.1
## <none>                   789.03 4965.5
## + opponent_strength  1   788.92 4966.8
## + ast                1   788.99 4967.2
## + travel_distance_w  1   789.01 4967.3
## + days_rest          1   789.03 4967.5
##
## Step:  AIC=4705.99
## win ~ fg_pct + reb + tov
##
##                     Df Deviance    AIC
## + fg3_pct            1   733.20 4608.4
## + home_indicator     1   746.48 4696.8
## + ast                1   747.49 4703.4
## + back_to_back_num   1   747.83 4705.6
## <none>                   748.19 4706.0
## + opponent_strength  1   747.93 4706.3
## + travel_distance_w  1   748.17 4707.9
## + days_rest          1   748.18 4707.9
##
## Step:  AIC=4608.41
## win ~ fg_pct + reb + tov + fg3_pct
##
##                     Df Deviance    AIC
## + ast                1   731.04 4595.9
## + home_indicator     1   731.70 4600.3
## + back_to_back_num   1   732.81 4607.8
## <none>                   733.20 4608.4
## + opponent_strength  1   732.91 4608.5
## + days_rest          1   733.17 4610.2
## + travel_distance_w  1   733.19 4610.4
##
## Step:  AIC=4595.92
## win ~ fg_pct + reb + tov + fg3_pct + ast
##
##                     Df Deviance    AIC
## + home_indicator     1   729.38 4586.7
## + back_to_back_num   1   730.63 4595.2
## <none>                   731.04 4595.9
## + opponent_strength  1   730.80 4596.3
## + days_rest          1   731.02 4597.8
## + travel_distance_w  1   731.03 4597.8
##
## Step:  AIC=4586.73
## win ~ fg_pct + reb + tov + fg3_pct + ast + home_indicator
##
##                     Df Deviance    AIC
## + back_to_back_num   1   728.99 4586.1
## <none>                   729.38 4586.7
```

```
## + opponent_strength  1   729.13 4587.1
## + travel_distance_w  1   729.28 4588.0
## + days_rest          1   729.36 4588.6
##
## Step:  AIC=4586.07
## win ~ fg_pct + reb + tov + fg3_pct + ast + home_indicator + back_to_back_num
##
##                     Df Deviance    AIC
## <none>                   728.99 4586.1
## + opponent_strength  1   728.74 4586.4
## + travel_distance_w  1   728.94 4587.7
## + days_rest          1   728.98 4588.0
```

```
summary(forward_aic)
```

```
##
## Call:
## glm(formula = win ~ fg_pct + reb + tov + fg3_pct + ast + home_indicator +
##     back_to_back_num, data = nba_glm)
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -2.7543313  0.0667508 -41.263  < 2e-16 ***
## fg_pct            4.3485492  0.1383656  31.428  < 2e-16 ***
## reb               0.0304262  0.0008563  35.531  < 2e-16 ***
## tov              -0.0242409  0.0014415 -16.816  < 2e-16 ***
## fg3_pct           0.8711476  0.0830518  10.489  < 2e-16 ***
## ast              -0.0052743  0.0013290  -3.969 7.34e-05 ***
## home_indicator    0.0367596  0.0110334   3.332  0.00087 ***
## back_to_back_num -0.0234947  0.0144079  -1.631  0.10302
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1484095)
##
##     Null deviance: 1230.00  on 4919  degrees of freedom
## Residual deviance:  728.99  on 4912  degrees of freedom
## AIC: 4586.1
##
## Number of Fisher Scoring iterations: 2
```

Check for Multicollinearity

```
# VIF Check
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:DescTools':
##
##     Recode

## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:purrr':
##
##     some
```

```
vif_values <- vif(full_glm)
vif_values
```

```
## travel_distance_w         days_rest  back_to_back_num    home_indicator
##          1.140456          1.008810          1.033908          1.116124
## opponent_strength           fg_pct          fg3_pct               ast
##          1.004421          1.877146          1.304678          1.320447
##               reb               tov
##          1.608652          1.147233
```

The model performs poorly on held-out testing data even VIF values show < 2. Classic symptom of original overfitting. Then we need Lasso/Ridge.

Train/Test Split

```
set.seed(123)

n <- nrow(nba_glm)
train_idx <- sample(1:n, size = 0.7 * n)

train_data <- nba_glm[train_idx, ]
test_data  <- nba_glm[-train_idx, ]

y_train <- train_data$win
y_test  <- test_data$win

# Create the model
X_train <- model.matrix(win ~ ., data = train_data)[, -1]
X_test  <- model.matrix(win ~ ., data = test_data)[, -1]
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
# Convert X_test to data frame (glm requires data.frame)
X_test <- as.data.frame(X_test)

# 1. Get predicted probabilities
log_prob <- predict(full_glm, newdata = X_test, type = "response")

# 2. Convert y_test to numeric (for WIN target)
y_num <- as.numeric(as.character(y_test))

# 3. AUC
log_auc <- auc(y_num, log_prob)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

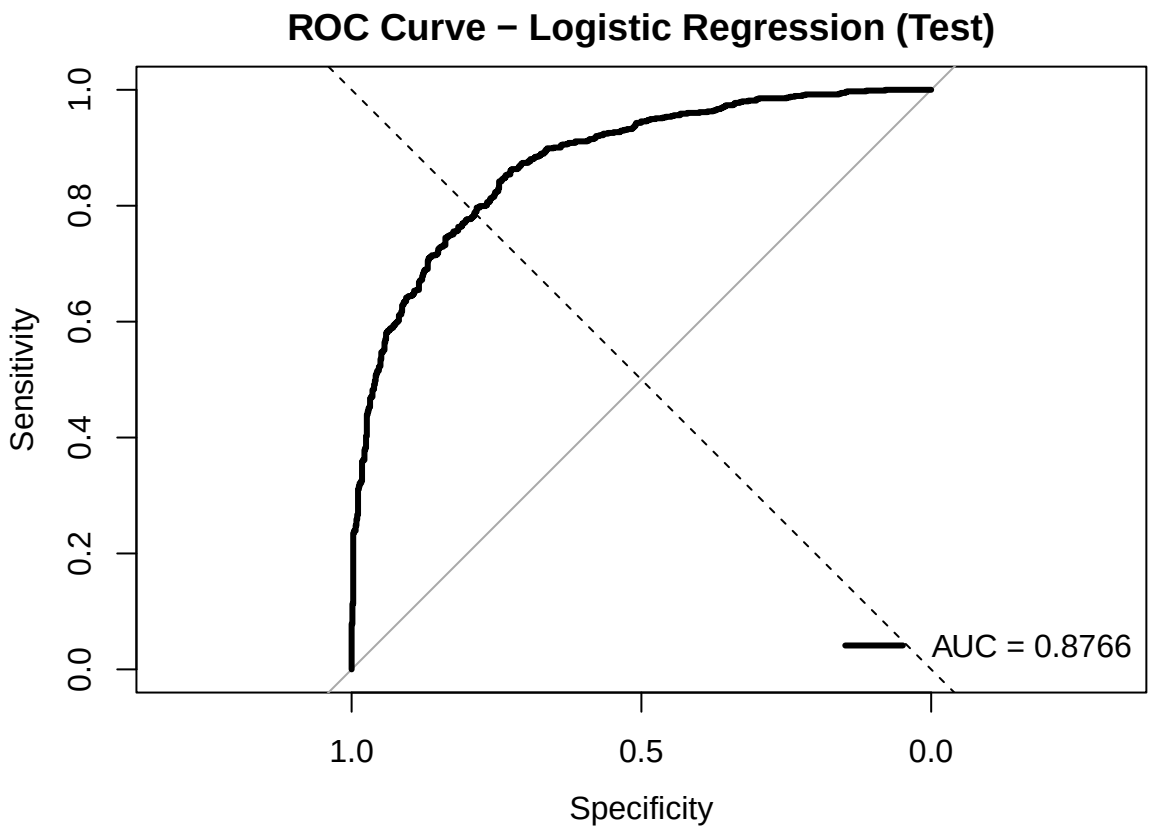
```
log_auc
```

```
## Area under the curve: 0.8766
```

```
# 4. Convert to classes
log_pred <- ifelse(log_prob > 0.5, 1, 0)

# 5. Accuracy
log_accuracy <- base::mean(log_pred == y_num)
log_accuracy
```

```
## [1] 0.7879404
```

```
library(pROC)

log_roc <- roc(response = y_num, predictor = log_prob, levels = c(0,1), direction = "<")

# plot
plot(log_roc, lwd = 3, main = "ROC Curve - Logistic Regression (Test)")
abline(a = 0, b = 1, lty = 2)  # diagonal
legend("bottomright",
       legend = paste0("AUC = ", round(auc(log_roc), 4)),
       lwd = 3, bty = "n")
```




```r
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following objects are masked from 'package:DescTools':
##
##     MAE, RMSE

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(ggplot2)
library(dplyr)

# --- Confusion matrix (with metrics) ---
ref  <- factor(y_num,    levels = c(0, 1))   # actual
pred <- factor(log_pred, levels = c(0, 1))   # predicted

cm <- caret::confusionMatrix(pred, ref, positive = "1")
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 567 157
##          1 156 596
##
##                Accuracy : 0.7879
##                  95% CI : (0.7662, 0.8085)
##     No Information Rate : 0.5102
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5757
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7915
##             Specificity : 0.7842
##          Pos Pred Value : 0.7926
##          Neg Pred Value : 0.7831
##              Prevalence : 0.5102
##          Detection Rate : 0.4038
##    Detection Prevalence : 0.5095
##       Balanced Accuracy : 0.7879
##
##        'Positive' Class : 1
##
```

```
# Quick access to key metrics:
cm$overall["Accuracy"]
```

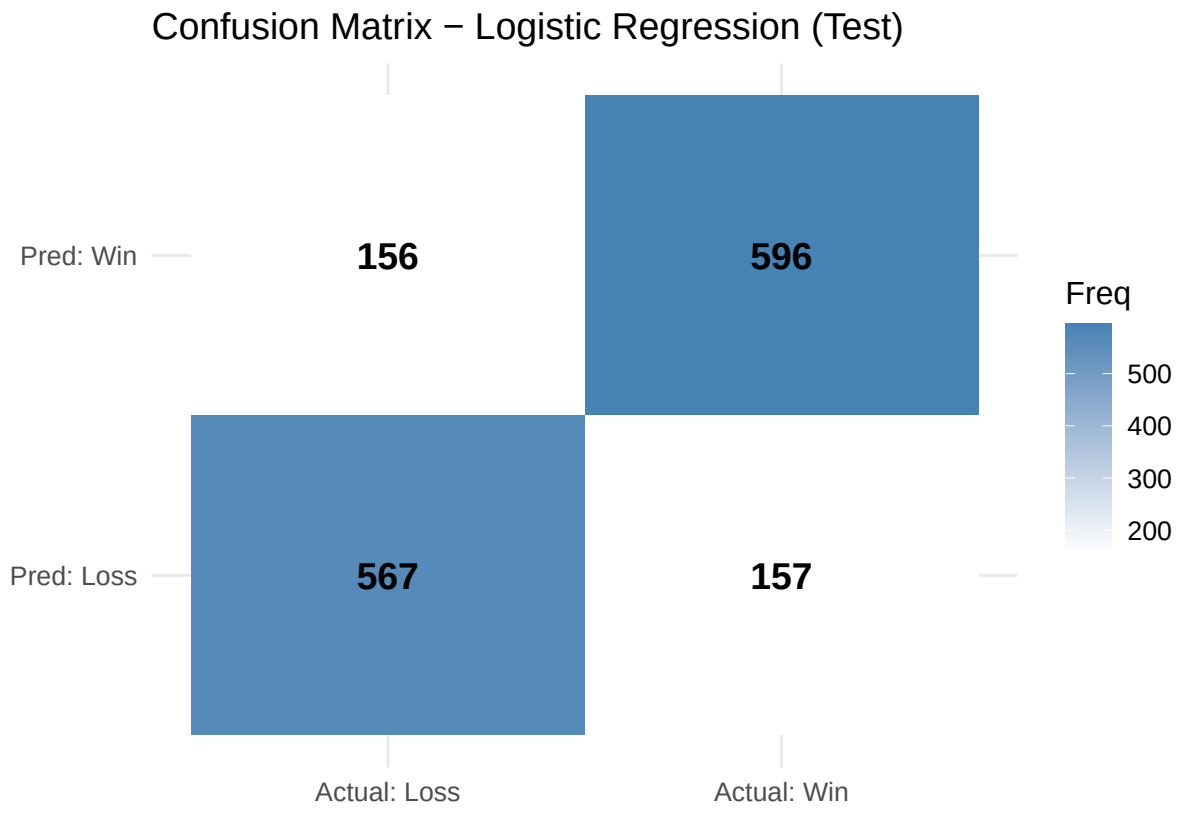
```
##  Accuracy
## 0.7879404
```

```
cm$byClass[c("Sensitivity", "Specificity", "Balanced Accuracy")]
```

```
##       Sensitivity       Specificity Balanced Accuracy
##         0.7915007         0.7842324         0.7878665
```

```
# --- Plot heatmap of confusion matrix ---
cm_df <- as.data.frame(cm$table) %>%
  mutate(
    Reference = factor(Reference, levels = c("0","1"), labels = c("Actual: Loss","Actual: Win")),
    Prediction = factor(Prediction, levels = c("0","1"), labels = c("Pred: Loss","Pred: Win"))
  )

ggplot(cm_df, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), size = 5, fontface = "bold") +
  scale_fill_gradient(low = "white", high = "steelblue") +
  labs(
    title = "Confusion Matrix - Logistic Regression (Test)",
```

```
    x = "", y = ""
  ) +
  theme_minimal(base_size = 12)
```



Ridge and Lasso Logistic Regression

```
# Ridge (alpha=0)
library(glmnet)

set.seed(123)

ridge_cv <- cv.glmnet(
  X_train, y_train,
  family = "binomial",
  alpha = 0
)

ridge_model <- glmnet(
  X_train, y_train,
  family = "binomial",
  alpha = 0,
  lambda = ridge_cv$lambda.min
```

```
)

# Lasso (alpha=1)
set.seed(123)

lasso_cv <- cv.glmnet(
  X_train, y_train,
  family = "binomial",
  alpha = 1
)

lasso_model <- glmnet(
  X_train, y_train,
  family = "binomial",
  alpha = 1,
  lambda = lasso_cv$lambda.min
)
```

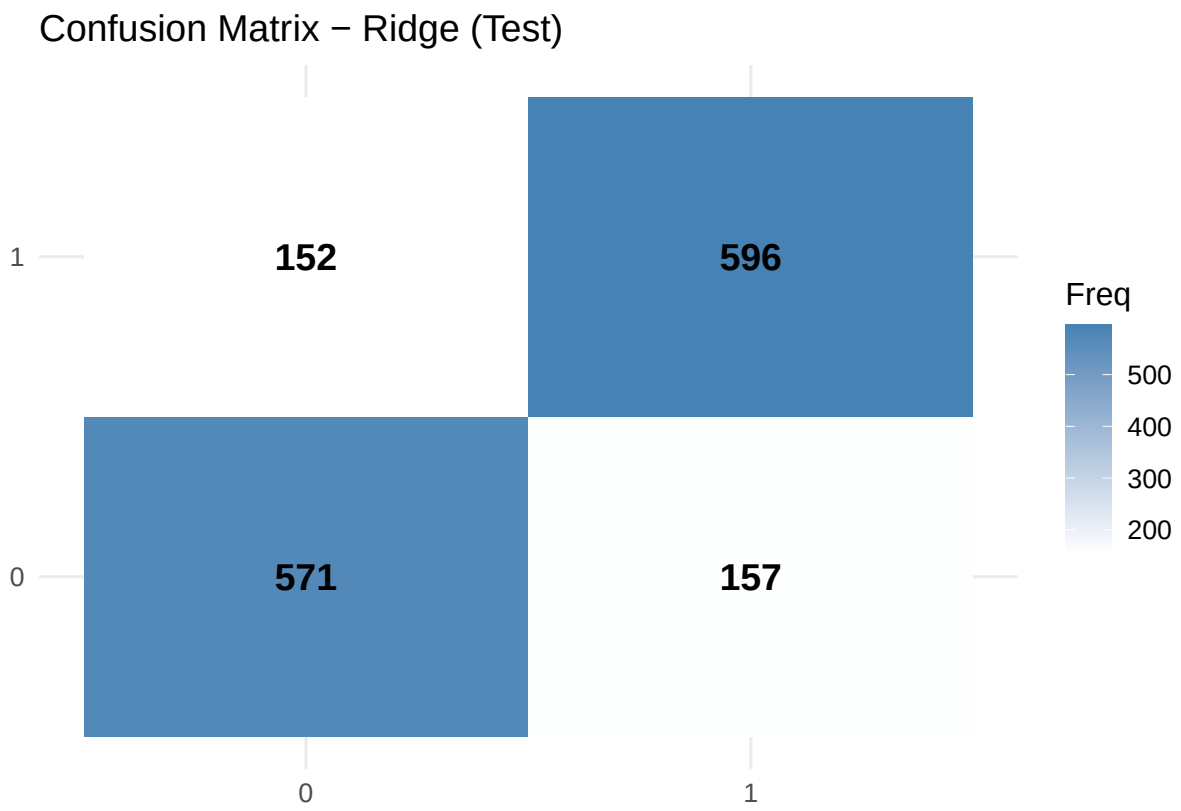
```
# 1) Capture the training column order
train_cols <- colnames(X_train)

# 2) Recreate the test design matrix with the SAME formula you used for X_train
#    If you built X_train via: model.matrix(win ~ ., data = train_data)[,-1]
#    then do the same here:
X_test <- model.matrix(win ~ ., data = test_data)[, -1, drop = FALSE]

# 3) Add any columns that exist in train but not in test (rare, e.g., missing factor levels)
missing_cols <- setdiff(train_cols, colnames(X_test))
if (length(missing_cols) > 0) {
  add_mat <- matrix(0, nrow = nrow(X_test), ncol = length(missing_cols))
  colnames(add_mat) <- missing_cols
  X_test <- cbind(X_test, add_mat)
}

# 4) Drop any extra columns that appear in test but not train, and reorder to match train
X_test <- X_test[, train_cols, drop = FALSE]

# 5) Ensure plain numeric matrix (glmnet calls as.matrix internally)
storage.mode(X_test) <- "double"
```

Predict on TEST DATA

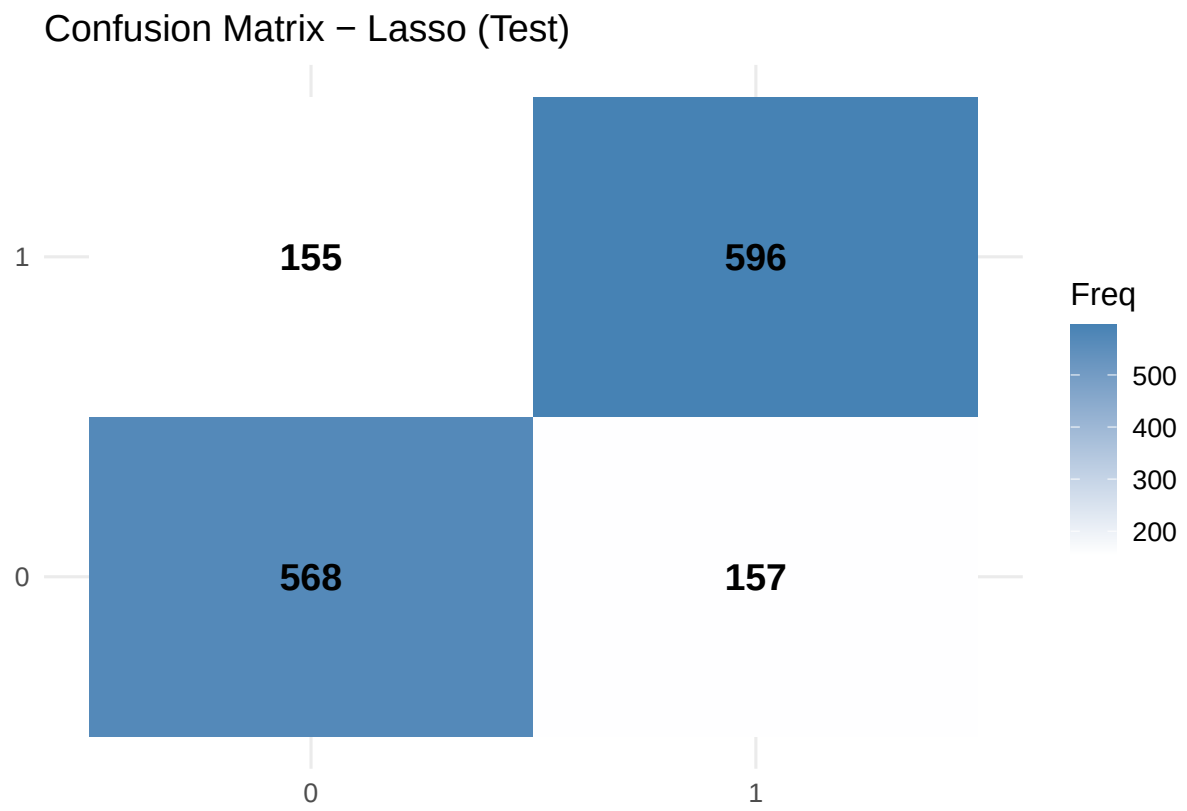
```
# Ridge
ridge_prob <- as.vector(predict(ridge_model, newx = X_test, type = "response"))

# Lasso
lasso_prob <- as.vector(predict(lasso_model, newx = X_test, type = "response"))
```

Confusion Matrix

```
## ---- confusion_matrix_plots, message=FALSE, warning=FALSE --------------

# Ensure predicted classes exist
ridge_pred <- ifelse(ridge_prob >= 0.5, 1, 0)
lasso_pred <- ifelse(lasso_prob >= 0.5, 1, 0)

# Helper to build cm data.frame in the shape you want
make_cm_df <- function(actual, predicted) {
  cm <- table(Reference = actual, Prediction = predicted)
  df <- as.data.frame(cm)
  # enforce 0/1 order for clean axis display
  df$Reference  <- factor(df$Reference,  levels = c(0,1))
  df$Prediction <- factor(df$Prediction, levels = c(0,1))
  df
}

cm_df_ridge <- make_cm_df(y_test, ridge_pred)
cm_df_lasso <- make_cm_df(y_test, lasso_pred)

library(ggplot2)

# Plot: Ridge
ggplot(cm_df_ridge, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), size = 5, fontface = "bold") +
  scale_fill_gradient(low = "white", high = "steelblue") +
  labs(
    title = "Confusion Matrix - Ridge (Test)",
    x = "", y = ""
  ) +
  theme_minimal(base_size = 12)
```




```
# Plot: Lasso
ggplot(cm_df_lasso, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), size = 5, fontface = "bold") +
  scale_fill_gradient(low = "white", high = "steelblue") +
  labs(
    title = "Confusion Matrix - Lasso (Test)",
    x = "", y = ""
  ) +
  theme_minimal(base_size = 12)
```



Compute AUC, Accuracy and ROC Curves

```
library(pROC)

# AUC
ridge_auc <- auc(y_test, ridge_prob)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
lasso_auc <- auc(y_test, lasso_prob)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
ridge_auc
```

```
## Area under the curve: 0.8734
```

```
lasso_auc
```

```
## Area under the curve: 0.8758
```

```
# Convert to classes
ridge_pred <- ifelse(ridge_prob > 0.5, 1, 0)
lasso_pred <- ifelse(lasso_prob > 0.5, 1, 0)

# Accuracy
ridge_accuracy <- base::mean(ridge_pred == y_test)
lasso_accuracy <- base::mean(lasso_pred == y_test)

ridge_accuracy
```

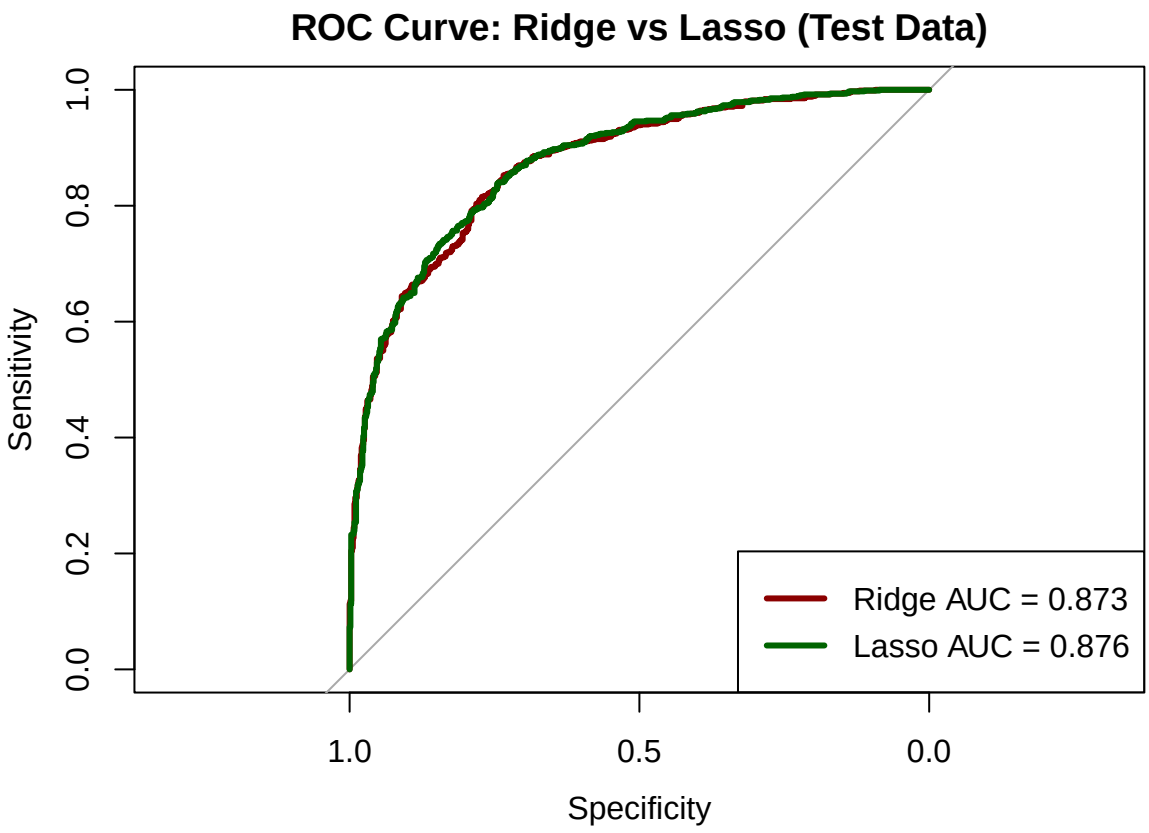
```
## [1] 0.7906504
```

```
lasso_accuracy
```

```
## [1] 0.7886179
```

Plot ROC Curves

```
ridge_roc <- roc(response = y_test,
                 predictor = ridge_prob,
                 levels = c(0, 1),
                 direction = "<",
                 quiet = TRUE)

lasso_roc <- roc(response = y_test,
                 predictor = lasso_prob,
                 levels = c(0, 1),
                 direction = "<",
                 quiet = TRUE)

plot(ridge_roc, col = "darkred", lwd = 3,
     main = "ROC Curve: Ridge vs Lasso (Test Data)")

plot(lasso_roc, col = "darkgreen", lwd = 3, add = TRUE)

legend("bottomright",
       legend = c(
         paste0("Ridge AUC = ", round(ridge_auc, 3)),
         paste0("Lasso AUC = ", round(lasso_auc, 3))
       ),
       col = c("darkred", "darkgreen"),
       lwd = 3)
```


ROC Curve: Ridge vs Lasso (Test Data)
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.0
0.5
0.0
Specificity
Ridge AUC = 0.873
Lasso AUC = 0.876