Home Court

Collin Travasos

2025-02-18

Packages

```
library(httr)
```

```
## Warning: package 'httr' was built under R version 4.4.2
```

```
library(rvest)
```

```
## Warning: package 'rvest' was built under R version 4.4.2
```

```
library(jsonlite)
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.2
```

```
## Warning: package 'dplyr' was built under R version 4.4.2
```

```
## -- 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   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter()         masks stats::filter()
## x purrr::flatten()        masks jsonlite::flatten()
## x readr::guess_encoding() masks rvest::guess_encoding()
## x dplyr::lag()            masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(dplyr)
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
```

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

```
library(tidyr)
library(scales)
```

```
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
library(Metrics)
```

```
##
## Attaching package: 'Metrics'
##
## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```
library(randomForest)
```

```
## randomForest 4.7-1.2
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

Scrape attendance

```
scrape_table <- function(url, year) {
  table <- read_html(url) %>%
    html_node("table") %>%
    html_table() %>%
    mutate(Year = year) %>%
    select(Year, everything())
  return(table)
}

urls <- c(
  "https://www.espn.com/nba/attendance/_/year/2024",
  "https://www.espn.com/nba/attendance/_/year/2023",
  "https://www.espn.com/nba/attendance/_/year/2022",
  "https://www.espn.com/nba/attendance/_/year/2021",
  "https://www.espn.com/nba/attendance/_/year/2020",
  "https://www.espn.com/nba/attendance/_/year/2019",
  "https://www.espn.com/nba/attendance/_/year/2018",
  "https://www.espn.com/nba/attendance/_/year/2017",
  "https://www.espn.com/nba/attendance/_/year/2016",
  "https://www.espn.com/nba/attendance/_/year/2015",
  "https://www.espn.com/nba/attendance/_/year/2014",
  "https://www.espn.com/nba/attendance/_/year/2013",
  "https://www.espn.com/nba/attendance/_/year/2012",
  "https://www.espn.com/nba/attendance/_/year/2011"
)
years <- c("2024", "2023", "2022", "2021", "2020", "2019", "2018", "2017", "2016", "2015", "2014", "201

tables <- mapply(scrape_table, urls, years, SIMPLIFY = FALSE)
espn_attendance_table <- bind_rows(tables)
```

Combine/clean attendance tables

```
espn_attendance_table <- espn_attendance_table %>%
  filter(!X2 %in% c("East", "West", "Dur", "Leb", "Gia", "Ste", "Usa", "World")) %>%
  select(-X6, -X9, -X12 ) %>%
  mutate_all(~ replace(., . == 0, NA))

colnames(espn_attendance_table) <-
  c("Year", "Season_Rank", "Team", "Home_Games", "Home_Total", "Home_Average", "Road_Games", "Road_Avera

espn_attendance_table <- espn_attendance_table %>%
  filter(!Home_Games %in% c("Home", "GMS"))

espn_attendance_table$Team <- gsub("76ers", "Sixers", espn_attendance_table$Team)
espn_attendance_table$Team <- gsub("Mavericks", "Mavs", espn_attendance_table$Team)
espn_attendance_table$Team <- gsub("Cavaliers", "Cavs", espn_attendance_table$Team)
espn_attendance_table$Team <- gsub("NY Knicks", "Knicks", espn_attendance_table$Team)
```

```r
espn_attendance_table$Team <- gsub("Trail Blazers", "Blazers", espn_attendance_table$Team)

espn_attendance_table <- espn_attendance_table %>%
  mutate(Team_Year = paste0(Team, Year)) %>%
  select(Team_Year, everything())
```

Results

```r
scrape_standings <- function(url, year) {
  espn <- read_html(url)
  tables <- espn %>%
    html_nodes("table") %>%
    html_table()

tableT <- tables[[1]]
tableR <- tables[[2]]

combined_table <- cbind(tableT, tableR) %>%
    mutate(Year = year) %>%
    select(Year, everything())

  return(combined_table)
}

urls <- c(
  "https://www.espn.com/nba/standings/_/season/2024/group/league",
  "https://www.espn.com/nba/standings/_/season/2023/group/league",
  "https://www.espn.com/nba/standings/_/season/2022/group/league",
  "https://www.espn.com/nba/standings/_/season/2021/group/league",
  "https://www.espn.com/nba/standings/_/season/2020/group/league",
  "https://www.espn.com/nba/standings/_/season/2019/group/league",
  "https://www.espn.com/nba/standings/_/season/2018/group/league",
  "https://www.espn.com/nba/standings/_/season/2017/group/league",
  "https://www.espn.com/nba/standings/_/season/2016/group/league",
  "https://www.espn.com/nba/standings/_/season/2015/group/league",
  "https://www.espn.com/nba/standings/_/season/2014/group/league",
  "https://www.espn.com/nba/standings/_/season/2013/group/league",
  "https://www.espn.com/nba/standings/_/season/2012/group/league",
  "https://www.espn.com/nba/standings/_/season/2011/group/league"
)

years <- c("2024", "2023", "2022", "2021", "2020", "2019", "2018", "2017", "2016", "2015", "2014", "201

standings_tables <- mapply(scrape_standings, urls, years, SIMPLIFY = FALSE)
espn_results_table <- bind_rows(standings_tables)
```

Combine/Clean Results Data

```r
espn_results_table <- espn_results_table %>%
  rename(Team = Var.1)

espn_results_table$Team <- sub(".*--", "", espn_results_table$Team)
```

```
espn_results_table$Team <- sub("^[A-Z]+", "", espn_results_table$Team)
espn_results_table$Team <- sub(".* ", "", espn_results_table$Team)

espn_results_table$Team <- gsub("76ers", "Sixers", espn_results_table$Team)
espn_results_table$Team <- gsub("Mavericks", "Mavs", espn_results_table$Team)
espn_results_table$Team <- gsub("Cavaliers", "Cavs", espn_results_table$Team)

espn_results_table <- espn_results_table %>%
  mutate(Team_Year = paste0(Team, Year)) %>%
  select(Team_Year, everything())
```

Ticket sales data from statista

```
Hawks <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Hawks.csv") %>%
  mutate(new_column = "Hawks")
colnames(Hawks) <- c("Season", "Dollars", "Team")

Celtics <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Celtics.csv") %>%
  mutate(new_column = "Celtics")
colnames(Celtics) <- c("Season", "Dollars", "Team")

Nets <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Nets.csv") %>%
  mutate(new_column = "Nets")
colnames(Nets) <- c("Season", "Dollars", "Team")

Hornets <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Hornets.csv") %>%
  mutate(new_column = "Hornets")
colnames(Hornets) <- c("Season", "Dollars", "Team")

Bulls <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Bulls.csv")%>%
  mutate(new_column = "Bulls")
colnames(Bulls) <- c("Season", "Dollars", "Team")

Cavs <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Cavs.csv")%>%
  mutate(new_column = "Cavs")
colnames(Cavs) <- c("Season", "Dollars", "Team")

Mavs <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Mavs.csv")%>%
  mutate(new_column = "Mavs")
colnames(Mavs) <- c("Season", "Dollars", "Team")

Nuggets <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Nuggets.csv") %>%
  mutate(new_column = "Nuggets")
colnames(Nuggets) <- c("Season", "Dollars", "Team")

Pistons <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Pistons.csv")%>%
  mutate(new_column = "Pistons")
colnames(Pistons) <- c("Season", "Dollars", "Team")

Warriors <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Warriors.csv")%>%
  mutate(new_column = "Warriors")
colnames(Warriors) <- c("Season", "Dollars", "Team")
```

```
Rockets <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Rockets.csv")%>%
  mutate(new_column = "Rockets")
colnames(Rockets) <- c("Season", "Dollars", "Team")

Pacers <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Pacers.csv")%>%
  mutate(new_column = "Pacers")
colnames(Pacers) <- c("Season", "Dollars", "Team")

Clippers <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Clippers.csv")%>%
  mutate(new_column = "Clippers")
colnames(Clippers) <- c("Season", "Dollars", "Team")

Lakers <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Lakers.csv")%>%
  mutate(new_column = "Lakers")
colnames(Lakers) <- c("Season", "Dollars", "Team")

Grizzlies <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Grizzlies.csv")%>%
  mutate(new_column = "Grizzlies")
colnames(Grizzlies) <- c("Season", "Dollars", "Team")

Heat <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Heat.csv")%>%
  mutate(new_column = "Heat")
colnames(Heat) <- c("Season", "Dollars", "Team")

Bucks <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Bucks.csv")%>%
  mutate(new_column = "Bucks")
colnames(Bucks) <- c("Season", "Dollars", "Team")

Timberwolves <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Timberwolves.csv")%>%
  mutate(new_column = "Timberwolves")
colnames(Timberwolves) <- c("Season", "Dollars", "Team")

Pelicans <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Pelicans.csv")%>%
  mutate(new_column = "Pelicans")
colnames(Pelicans) <- c("Season", "Dollars", "Team")

Knicks <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Knicks.csv")%>%
  mutate(new_column = "Knicks")
colnames(Knicks) <- c("Season", "Dollars", "Team")

Thunder <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Thunder.csv")%>%
  mutate(new_column = "Thunder")
colnames(Thunder) <- c("Season", "Dollars", "Team")

Magic <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Magic.csv")%>%
  mutate(new_column = "Magic")
colnames(Magic) <- c("Season", "Dollars", "Team")

Sixers <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Sixers.csv")%>%
  mutate(new_column = "Sixers")
colnames(Sixers) <- c("Season", "Dollars", "Team")

Suns <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Suns.csv")%>%
```

```
  mutate(new_column = "Suns")
colnames(Suns) <- c("Season", "Dollars", "Team")

Blazers <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Blazers.csv")%>%
  mutate(new_column = "Blazers")
colnames(Blazers) <- c("Season", "Dollars", "Team")

Kings <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Kings.csv")%>%
  mutate(new_column = "Kings")
colnames(Kings) <- c("Season", "Dollars", "Team")

Spurs <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Spurs.csv")%>%
  mutate(new_column = "Spurs")
colnames(Spurs) <- c("Season", "Dollars", "Team")

Raptors <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Raptors.csv")%>%
  mutate(new_column = "Raptors")
colnames(Raptors) <- c("Season", "Dollars", "Team")

Jazz <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Jazz.csv")%>%
  mutate(new_column = "Jazz")
colnames(Jazz) <- c("Season", "Dollars", "Team")

Wizards <- read.csv("C:/Users/cstra/Downloads/NBA_Gate_Reciepts_Statista/Wizards.csv")%>%
  mutate(new_column = "Wizards")
colnames(Wizards) <- c("Season", "Dollars", "Team")
```

Organize/clean ticket sales

```
statista_attendance_data <- bind_rows(Hawks, Celtics, Nets, Hornets, Bulls, Cavs, Mavs, Nuggets, Piston


statista_attendance_data$Dollars <- statista_attendance_data$Dollars * 1000000
statista_attendance_data$Dollars <- format(statista_attendance_data$Dollars, scientific = FALSE)

statista_attendance_data <- statista_attendance_data %>%
  mutate(Season = sapply(strsplit(as.character(Season), "/"), function(x) paste0("20", x[2])))

statista_attendance_data <- statista_attendance_data %>%
  filter(Season != "2011 to 2022" & Season != "20NA")

statista_attendance_data <- statista_attendance_data %>% rename(Year = Season)
statista_attendance_data <- statista_attendance_data %>% rename(Gate_Receipts = Dollars)



statista_attendance_data <- statista_attendance_data %>%
  mutate(Team_Year = paste0(Team, Year)) %>%
  select(Team_Year, everything())
```

Combine 3 data sets

```
hc_combined_dataset <- statista_attendance_data %>%
  inner_join(espn_attendance_table, by = "Team_Year") %>%
  inner_join(espn_results_table, by = "Team_Year")
```

Clean

```
hc_combined_dataset <- hc_combined_dataset %>%
  select(-c(Year.y, Team.y, Year.x, Team.x))

hc_combined_dataset <- hc_combined_dataset %>%
  separate(HOME, into = c("Home_Wins", "Home_Losses"), sep = "\\s*-\\s*") %>%
  mutate(Home_Wins = as.numeric(Home_Wins),
         Home_Losses = as.numeric(Home_Losses),
         Home_Win_Percentage = Home_Wins / (Home_Wins + Home_Losses)) #%>%

hc_combined_dataset <- hc_combined_dataset %>%
  mutate(Home_Average = as.numeric(gsub(",", "", Home_Average)),
         Home_Total = as.numeric(gsub(",", "", Home_Total)),
         Gate_Receipts = as.numeric(gsub(",", "", Gate_Receipts)))

hc_combined_dataset <- hc_combined_dataset %>%
  filter(Year != "2021")


hc_combined_dataset <- hc_combined_dataset %>%
  mutate(Gate_Receipts = format(as.numeric(Gate_Receipts), scientific = FALSE))

hc_combined_dataset <- hc_combined_dataset %>%
  mutate(Home_Win_Loss = ifelse(Home_Wins > Home_Losses, 1, 0))



summary(hc_combined_dataset)
```

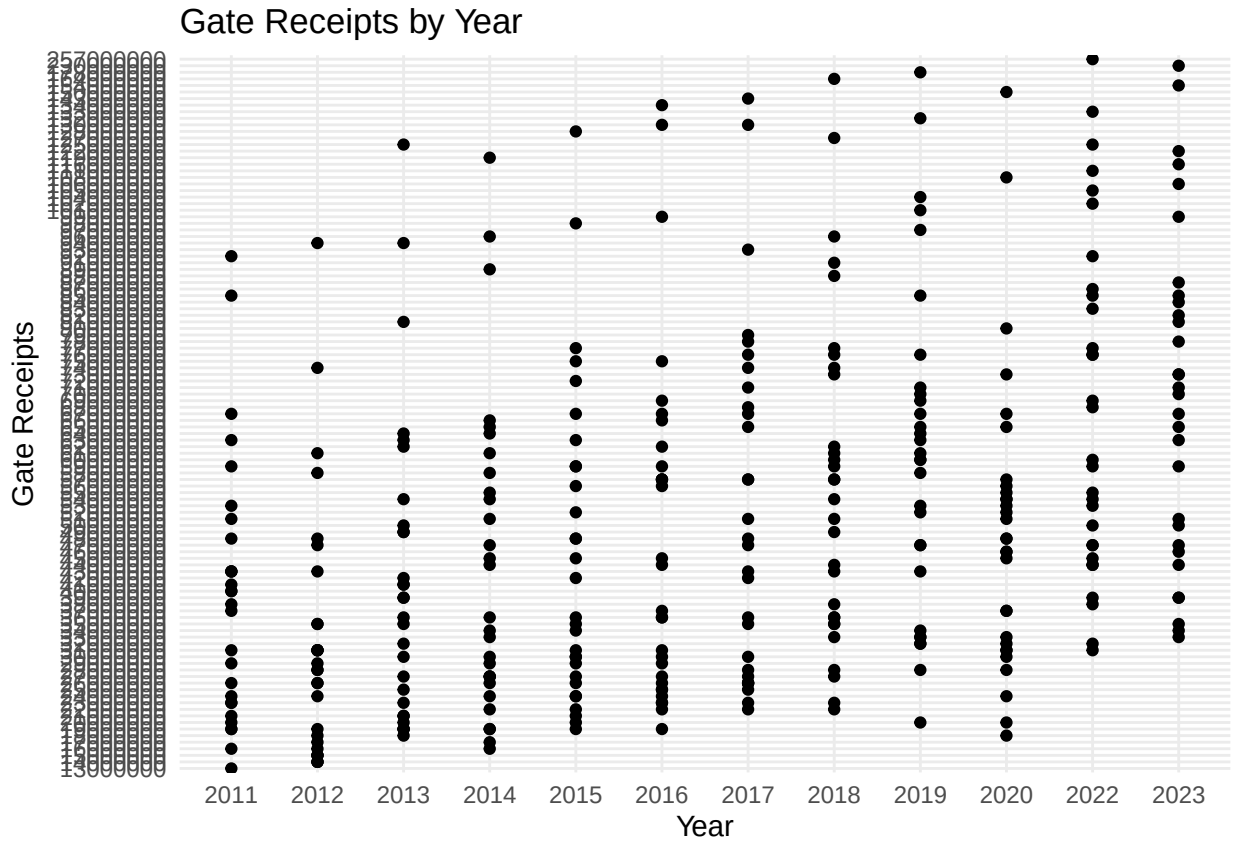
```
##   Team_Year         Gate_Receipts      Season_Rank         Home_Games
##  Length:355         Length:355         Length:355         Length:355
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##    Home_Total      Home_Average    Road_Games        Road_Average
##  Min.   :460719   Min.   :13487   Length:355         Length:355
##  1st Qu.:632259   1st Qu.:16423   Class :character   Class :character
##  Median :704886   Median :17693   Mode  :character   Mode  :character
##  Mean   :699189   Mean   :17690
##  3rd Qu.:770109   3rd Qu.:19177
##  Max.   :896944   Max.   :22161
##  Overall_Games      Overall_Average        Year               Team
##  Length:355         Length:355         Length:355         Length:355
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
```

```
##
##
##        W                L               PCT              GB
##  Min.   :10.00   Min.   : 9.00   Min.   :0.1220   Length:355
##  1st Qu.:31.00   1st Qu.:31.00   1st Qu.:0.3900   Class :character
##  Median :41.00   Median :39.00   Median :0.5120   Mode  :character
##  Mean   :39.99   Mean   :39.73   Mean   :0.5014
##  3rd Qu.:49.00   3rd Qu.:48.00   3rd Qu.:0.6100
##  Max.   :73.00   Max.   :72.00   Max.   :0.8900
##    Home_Wins      Home_Losses        AWAY               DIV
##  Min.   : 7.00   Min.   : 1.00   Length:355         Length:355
##  1st Qu.:19.00   1st Qu.:11.00   Class :character   Class :character
##  Median :23.00   Median :16.00   Mode  :character   Mode  :character
##  Mean   :23.24   Mean   :16.62
##  3rd Qu.:28.00   3rd Qu.:21.00
##  Max.   :40.00   Max.   :34.00
##      CONF                PPG            OPP PPG            DIFF
##  Length:355         Min.   : 89.60   Min.   : 88.20   Min.   :-10.40000
##  Class :character   1st Qu.: 99.15   1st Qu.: 99.85   1st Qu.: -3.10000
##  Mode  :character   Median :104.40   Median :104.70   Median :  0.30000
##                     Mean   :104.96   Mean   :104.91   Mean   :  0.05747
##                     3rd Qu.:110.70   3rd Qu.:109.95   3rd Qu.:  3.30000
##                     Max.   :120.70   Max.   :123.10   Max.   : 11.60000
##      STRK               L10            Home_Win_Percentage Home_Win_Loss
##  Length:355         Length:355         Min.   :0.1707      Min.   :0.0000
##  Class :character   Class :character   1st Qu.:0.4634      1st Qu.:0.0000
##  Mode  :character   Mode  :character   Median :0.5854      Median :1.0000
##                                        Mean   :0.5827      Mean   :0.7014
##                                        3rd Qu.:0.7073      3rd Qu.:1.0000
##                                        Max.   :0.9756      Max.   :1.0000
```

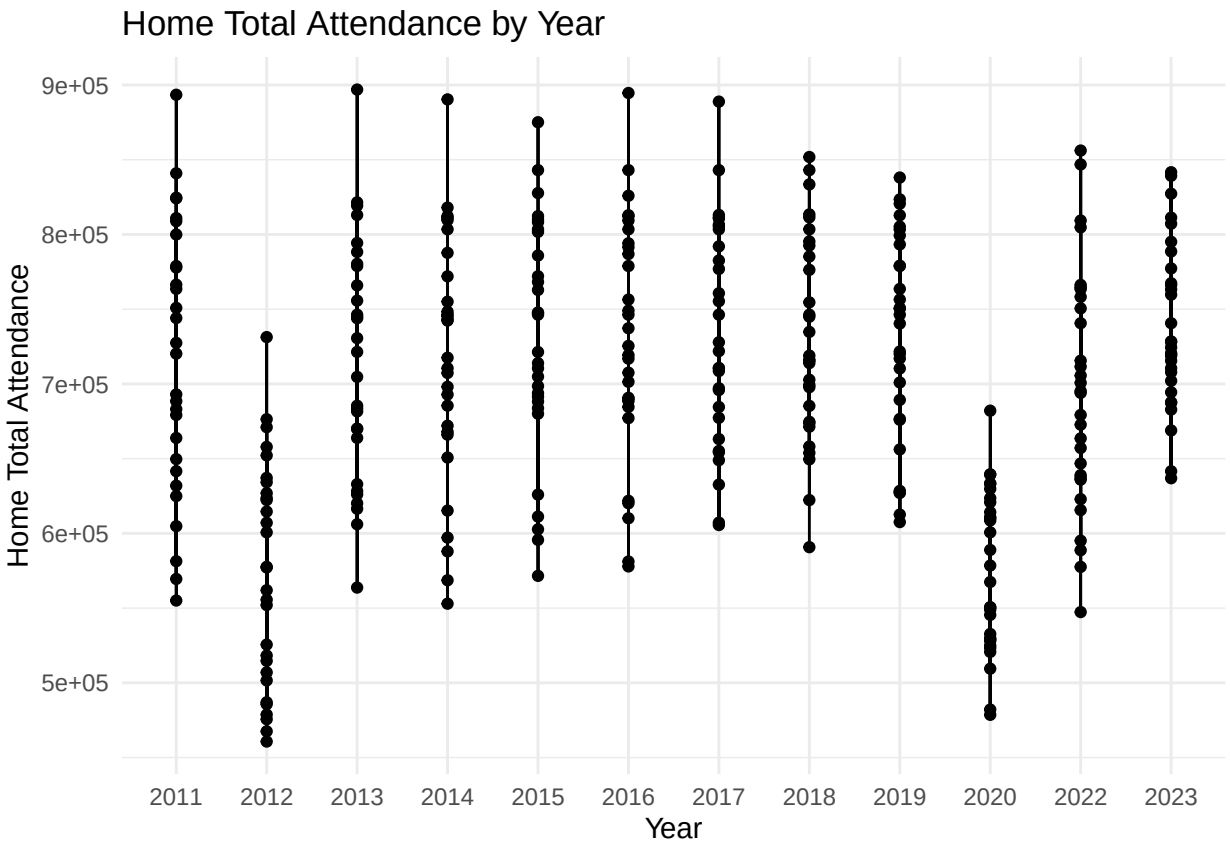
Visualizations

```
g1 <- ggplot(hc_combined_dataset, aes(x = Year, y = Gate_Receipts)) +
  geom_line() +
  geom_point() +
  labs(title = "Gate Receipts by Year",
       x = "Year",
       y = "Gate Receipts") +
  theme_minimal()

g2 <- ggplot(hc_combined_dataset, aes(x = Year, y = Home_Total)) +
  geom_line() +
  geom_point() +
  labs(title = "Home Total Attendance by Year",
       x = "Year",
       y = "Home Total Attendance") +
  theme_minimal()

g3 <- ggplot(hc_combined_dataset, aes(x = PCT, y = Home_Average)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Home Average Attendance vs. Winning Percentage",
       x = "Winning Percentage",
```

```
       y = "Home Average Attendance") +
  theme_minimal()

g4 <- ggplot(hc_combined_dataset, aes(x = Home_Average, y = Gate_Receipts)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Gate Receipts vs. Home Average Attendance",
       x = "Home Average Attendance",
       y = "Gate Receipts") +
  theme_minimal()

g5 <- ggplot(hc_combined_dataset, aes(x = reorder(Team, PCT), y = PCT)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Winning Percentage by Team",
       x = "Team",
       y = "Winning Percentage") +
  theme_minimal()

g1
```

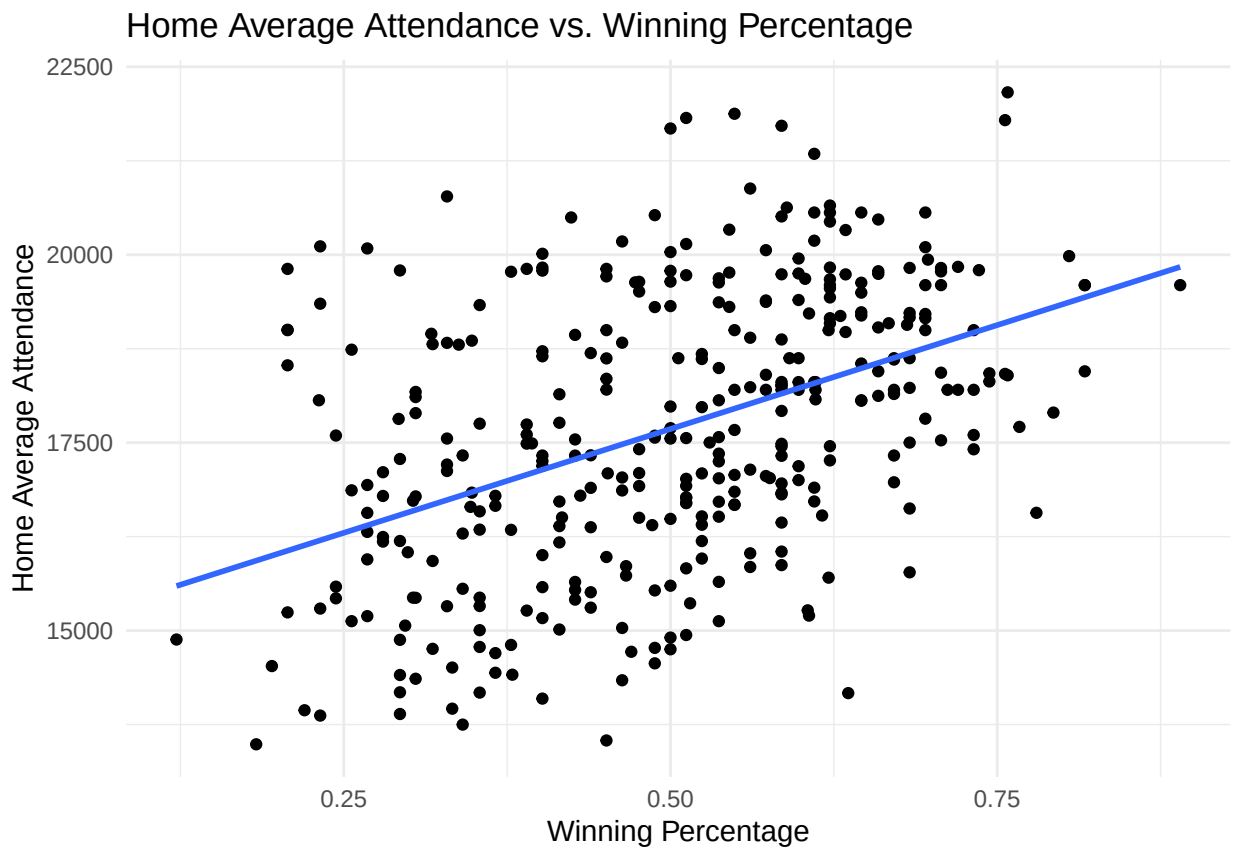


```
g2
```



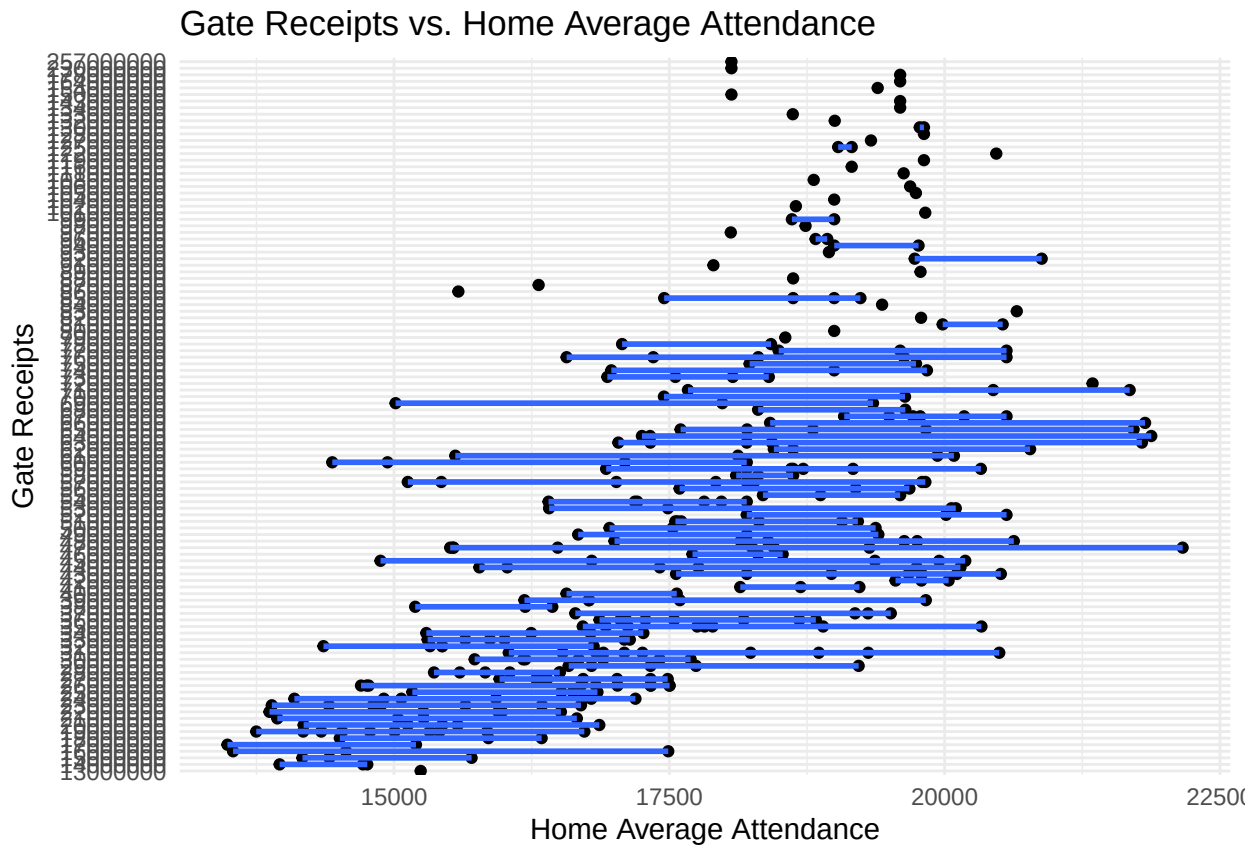

```
g3
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

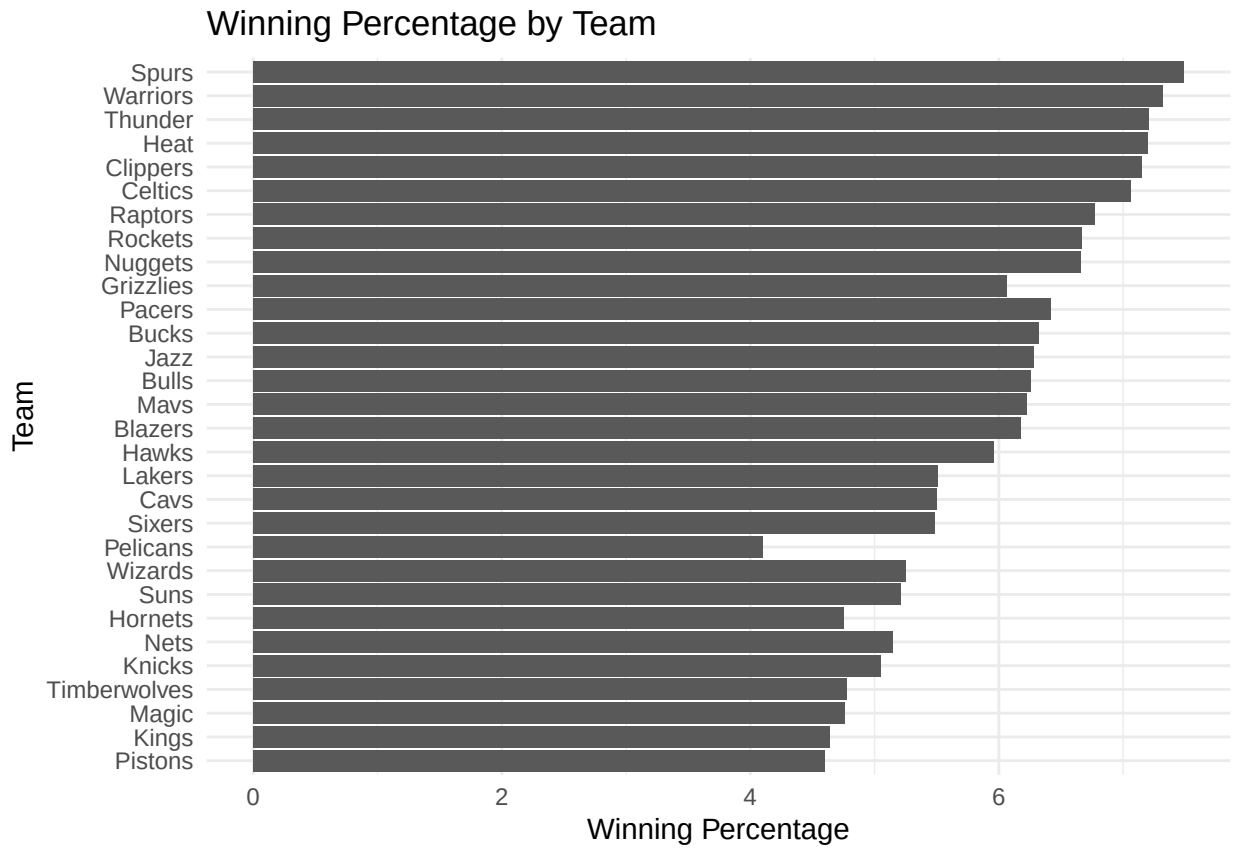



```
g4
```

```
## `geom_smooth()` using formula = 'y ~ x'
```


Gate Receipts vs. Home Average Attendance
Gate Receipts
Home Average Attendance
15000
17500
20000

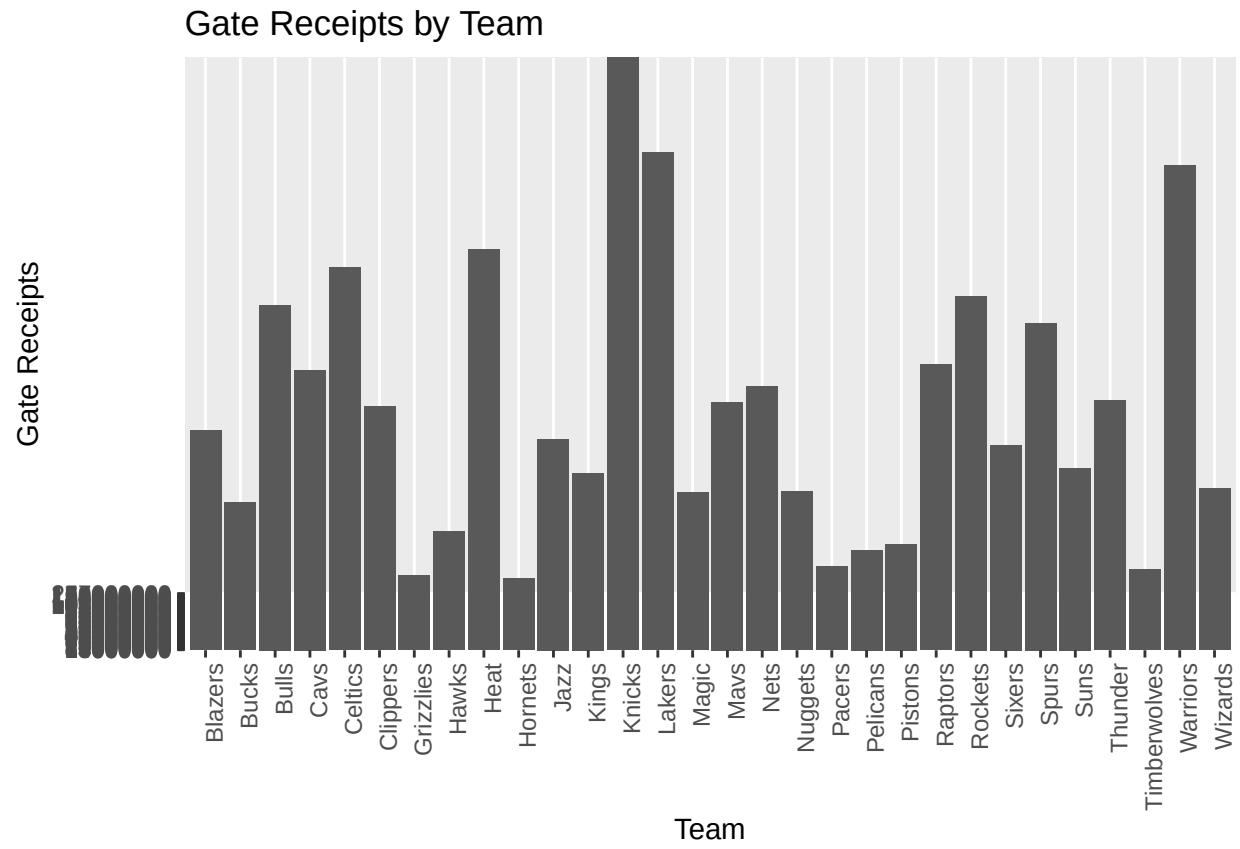
```
g5
```



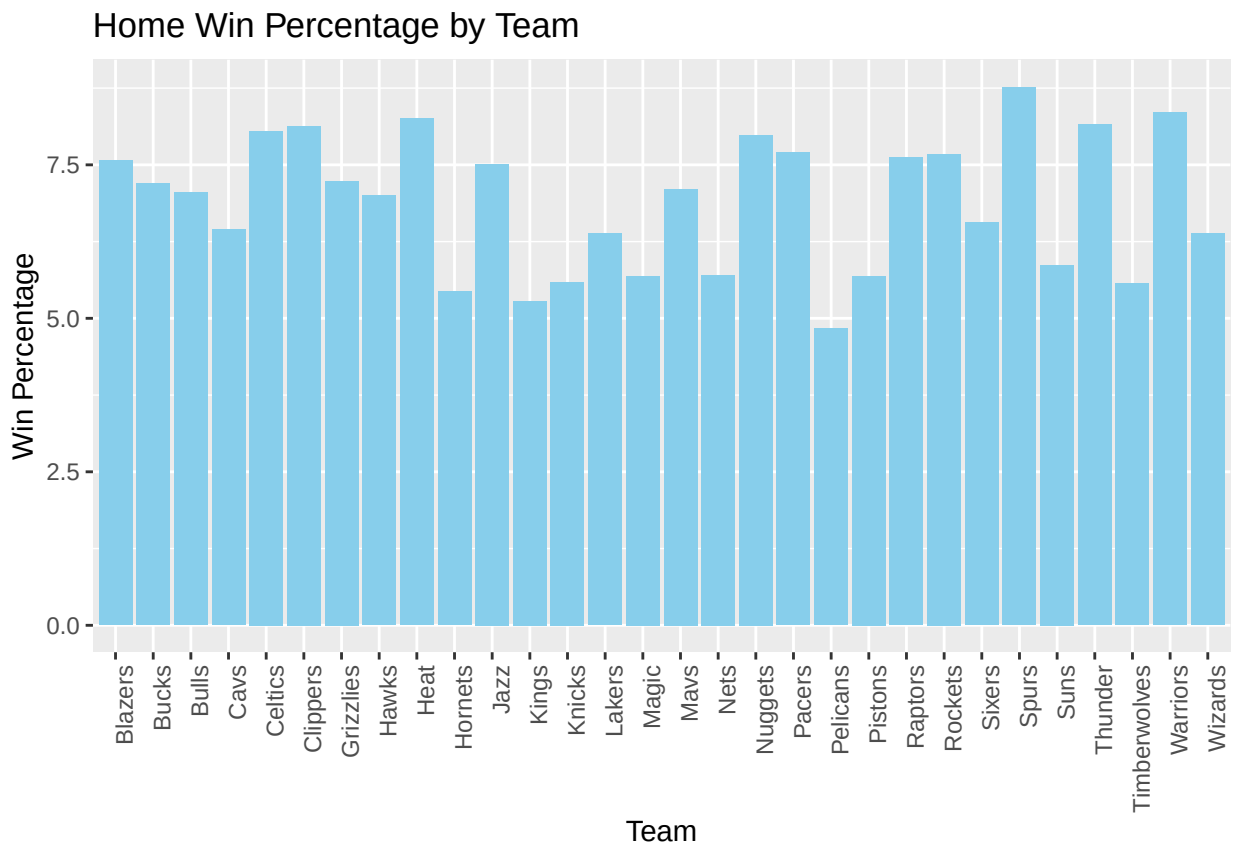
More Viz

```r
# Visualization 1: Gate Receipts by Team
v1 <- ggplot(hc_combined_dataset, aes(x = Team, y = Gate_Receipts)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Gate Receipts by Team", x = "Team", y = "Gate Receipts")

# Visualization 2: Home Win Percentage by Team
v2 <- ggplot(hc_combined_dataset, aes(x = Team, y = Home_Win_Percentage)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Home Win Percentage by Team", x = "Team", y = "Win Percentage")

# Visualization 3: Home Average Attendance by Team
v3 <- ggplot(hc_combined_dataset, aes(x = Team, y = Home_Average)) +
  geom_bar(stat = "identity", fill = "lightgreen") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Home Average Attendance by Team", x = "Team", y = "Home Average")

# Visualization 4: Scatter Plot of Gate Receipts vs. Home Average
v4 <- ggplot(hc_combined_dataset, aes(x = Home_Average, y = Gate_Receipts)) +
  geom_point() +
  geom_smooth(method = "lm", col = "red") +
  labs(title = "Gate Receipts vs. Home Average", x = "Home Average", y = "Gate Receipts")
```

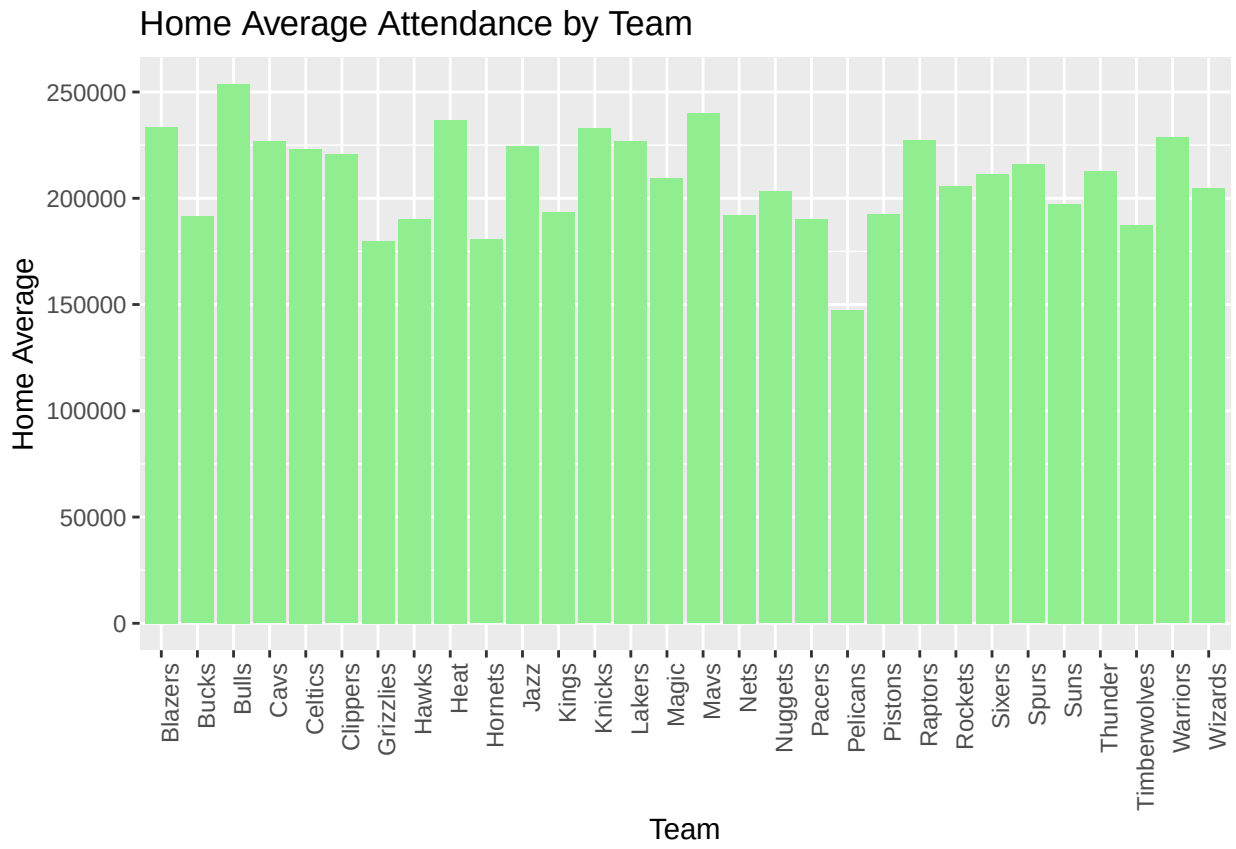
```r
# Visualization 5: Scatter Plot of Gate Receipts vs. Home Win Percentage
v5 <- ggplot(hc_combined_dataset, aes(x = Home_Win_Percentage, y = Gate_Receipts)) +
  geom_point() +
  geom_smooth(method = "lm", col = "blue") +
  labs(title = "Gate Receipts vs. Home Win Percentage", x = "Home Win Percentage", y = "Gate Receipts")

# Visualization 6: Boxplot of Gate Receipts by Team
v6 <- ggplot(hc_combined_dataset, aes(x = Team, y = Gate_Receipts)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Distribution of Gate Receipts by Team", x = "Team", y = "Gate Receipts")

# Visualization 7: Boxplot of Home Average Attendance by Team
v7 <- ggplot(hc_combined_dataset, aes(x = Team, y = Home_Average)) +
  geom_boxplot(fill = "orange") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Distribution of Home Average Attendance by Team", x = "Team", y = "Home Average")

# Visualization 8: Line Plot of Home Win Percentage Over Time
v8 <- ggplot(hc_combined_dataset, aes(x = Year, y = Home_Win_Percentage, group = Team, color = Team)) +
  geom_line() +
  labs(title = "Home Win Percentage Over Time", x = "Year", y = "Home Win Percentage")

# Visualization 9: Bar Plot of Home Win Percentage Grouped by Year
v9 <- ggplot(hc_combined_dataset, aes(x = Year, y = Home_Win_Percentage, fill = Team)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Home Win Percentage Grouped by Year", x = "Year", y = "Home Win Percentage")

# Visualization 10: Heatmap of Home Win Percentage by Team and Year
v10 <- ggplot(hc_combined_dataset, aes(x = Year, y = Team, fill = Home_Win_Percentage)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "red") +
  labs(title = "Heatmap of Home Win Percentage", x = "Year", y = "Team", fill = "Win %")

# Display the visualizations
v1
```


Gate Receipts by Team
Gate Receipts
Team
Blazers
Bucks
Bulls
Cavs
Celtics
Clippers
Grizzlies
Hawks
Heat
Hornets
Jazz
Kings
Knicks
Lakers
Magic
Mavs
Nets
Nuggets
Pacers
Pelicans
Pistons
Raptors
Rockets
Sixers
Spurs
Suns
Thunder
Timberwolves
Warriors
Wizards

```
v2
```


Home Win Percentage by Team
Win Percentage
7.5
5.0
2.5
0.0
Blazers
Bucks
Bulls
Cavs
Celtics
Clippers
Grizzlies
Hawks
Heat
Hornets
Jazz
Kings
Knicks
Lakers
Magic
Mavs
Nets
Nuggets
Pacers
Pelicans
Pistons
Raptors
Rockets
Sixers
Spurs
Suns
Thunder
Timberwolves
Warriors
Wizards
Team

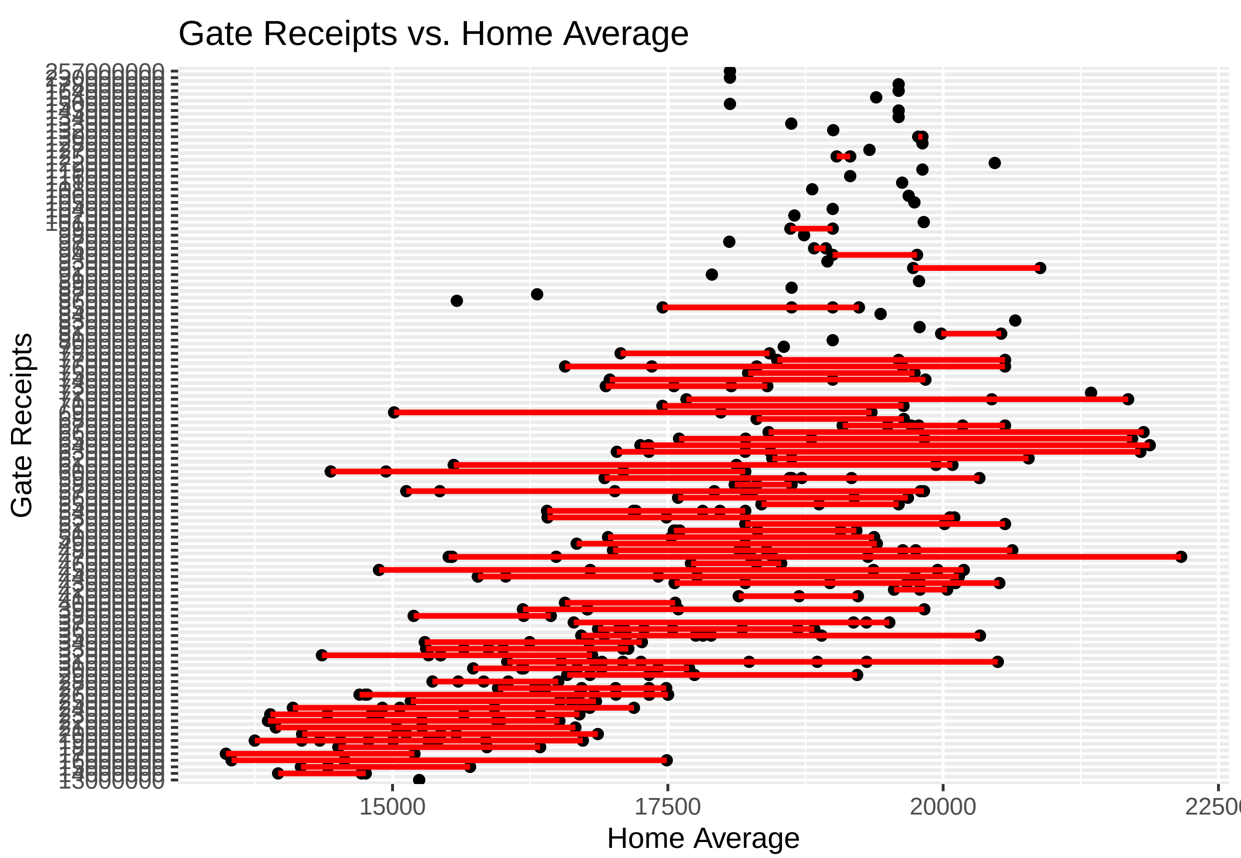
v3




```
v4
```

```
## `geom_smooth()` using formula = 'y ~ x'

## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced

## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```

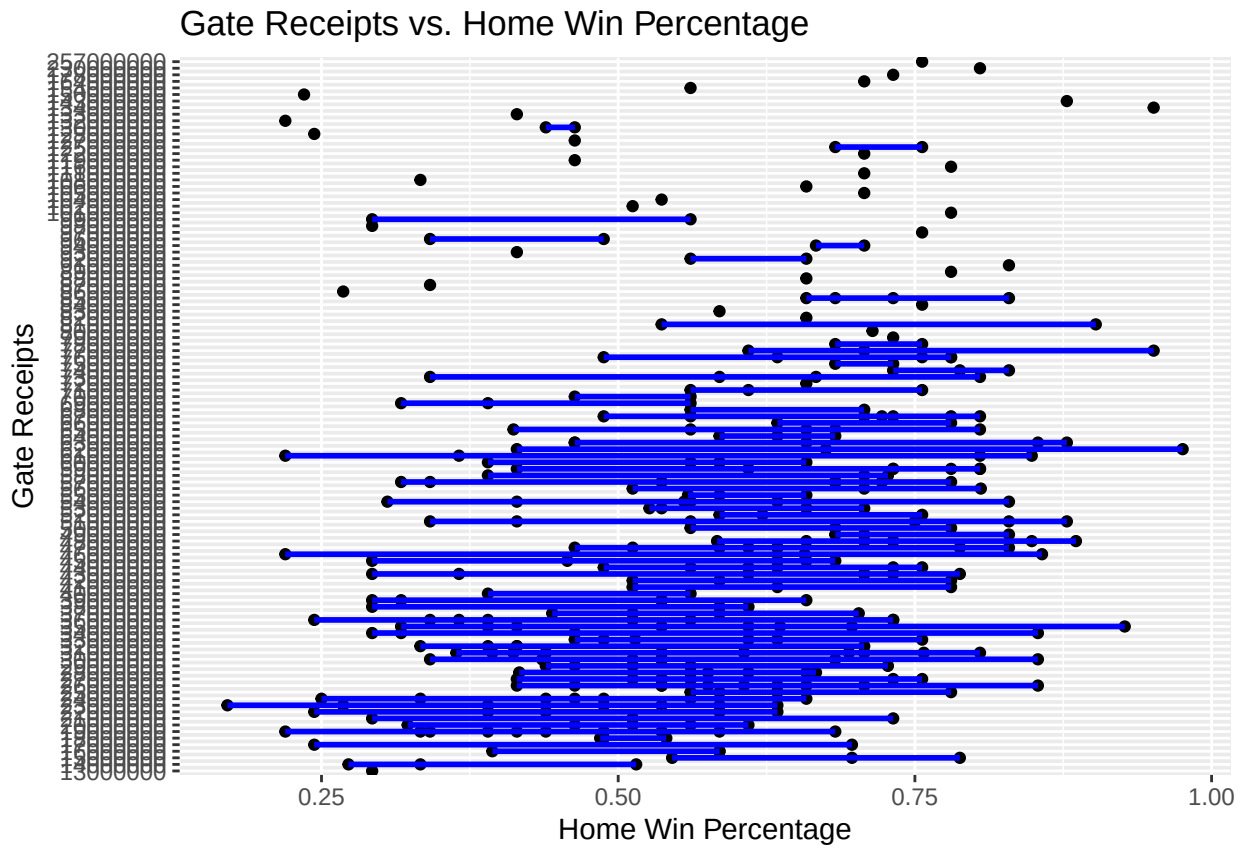
```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```



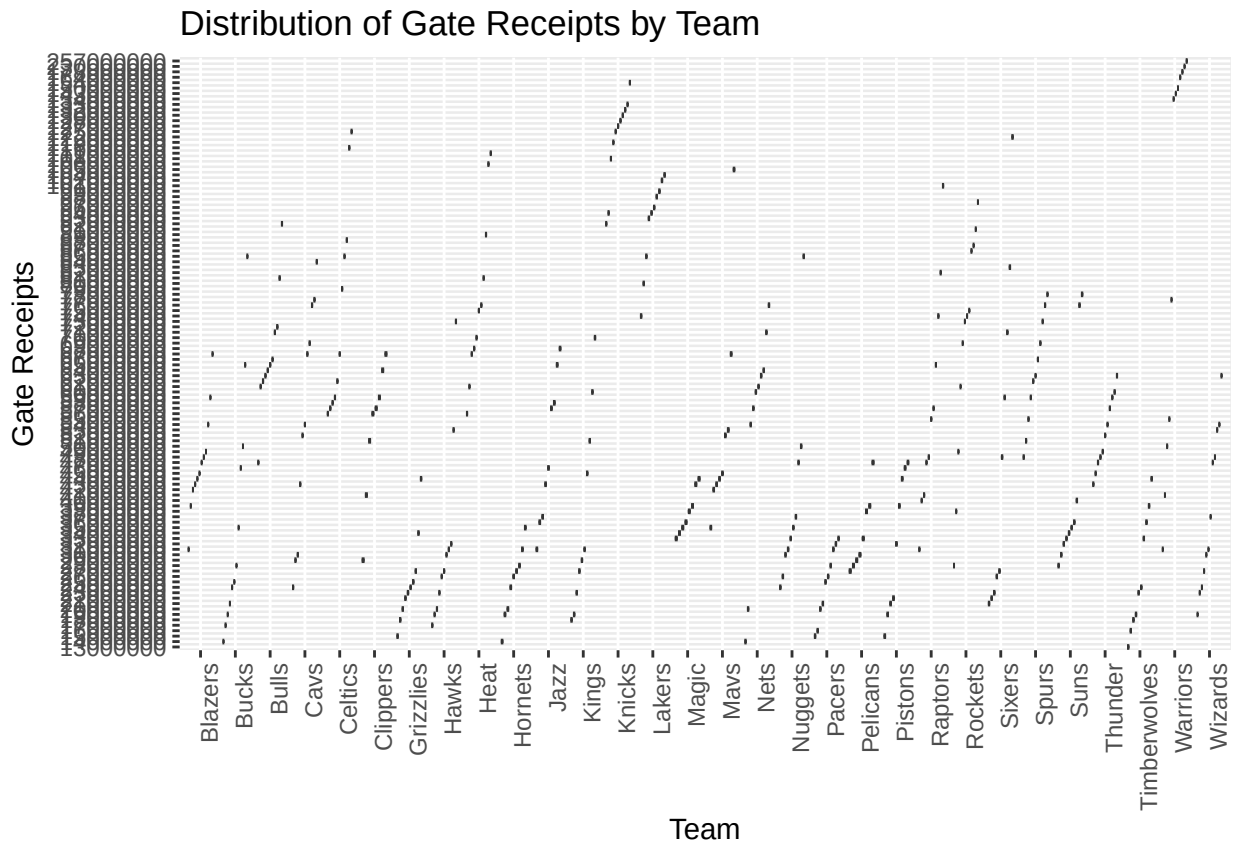

```
v5
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

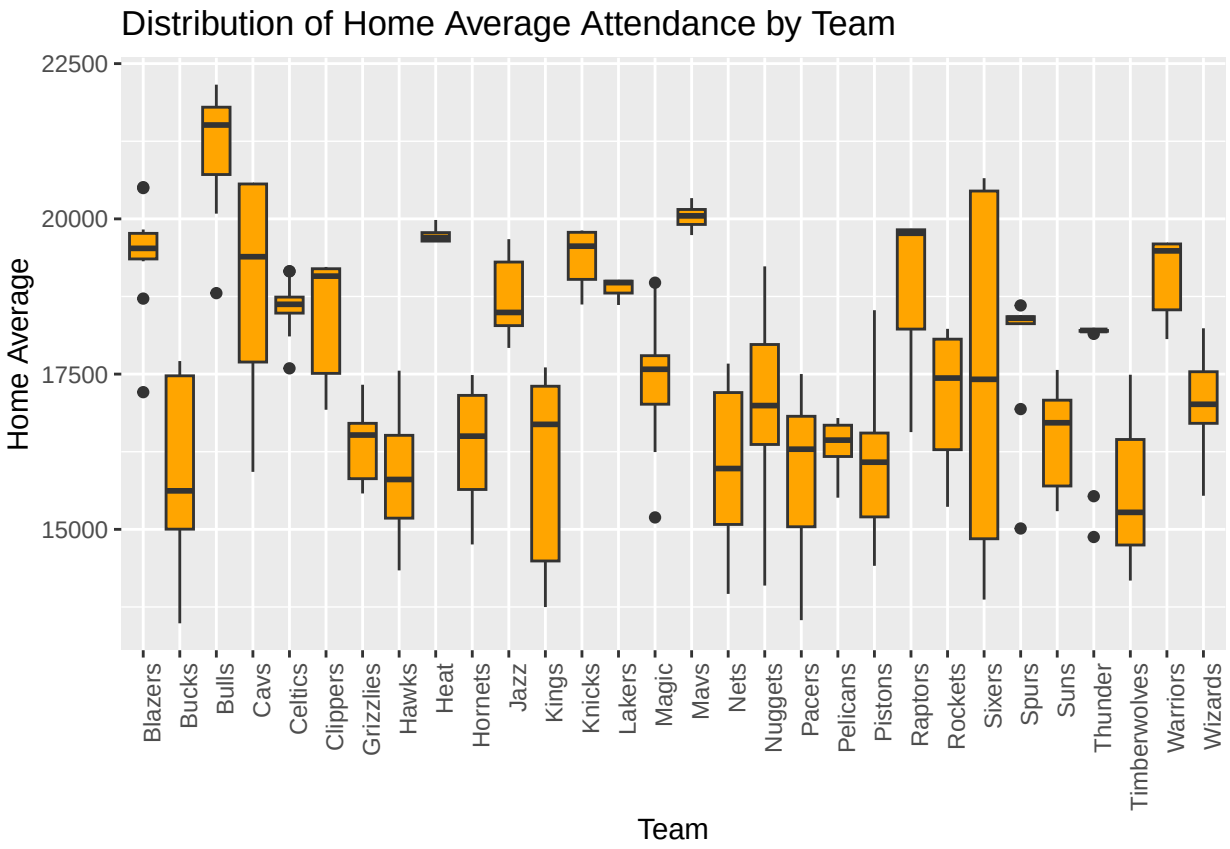
```
## Warning in qt((1 - level)/2, df): NaNs produced

## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning in qt((1 - level)/2, df): NaNs produced

## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```


Gate Receipts vs. Home Win Percentage
Gate Receipts
Home Win Percentage
0.25
0.50
0.75
1.00

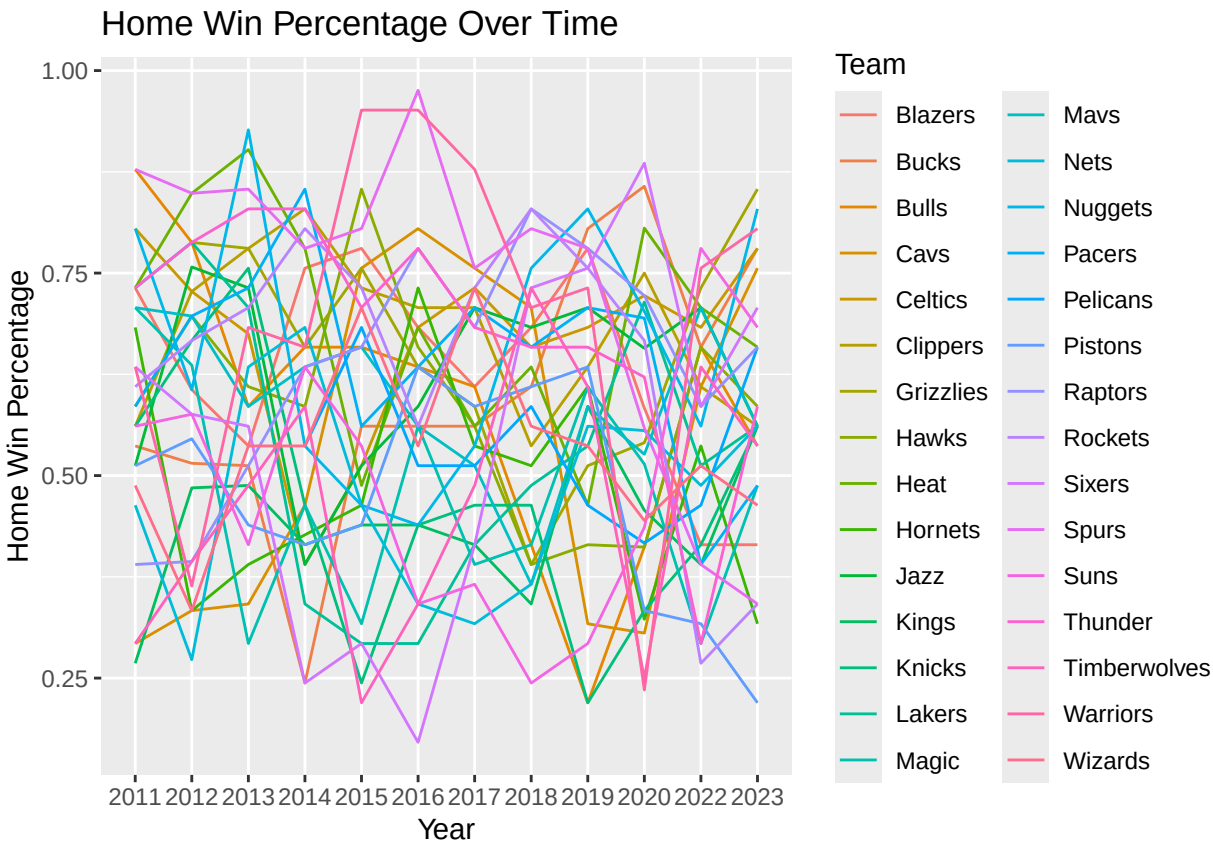
v6


Distribution of Gate Receipts by Team
Gate Receipts
Team
Blazers
Bucks
Bulls
Cavs
Celtics
Clippers
Grizzlies
Hawks
Heat
Hornets
Jazz
Kings
Knicks
Lakers
Magic
Mavs
Nets
Nuggets
Pacers
Pelicans
Pistons
Raptors
Rockets
Sixers
Spurs
Suns
Thunder
Timberwolves
Warriors
Wizards

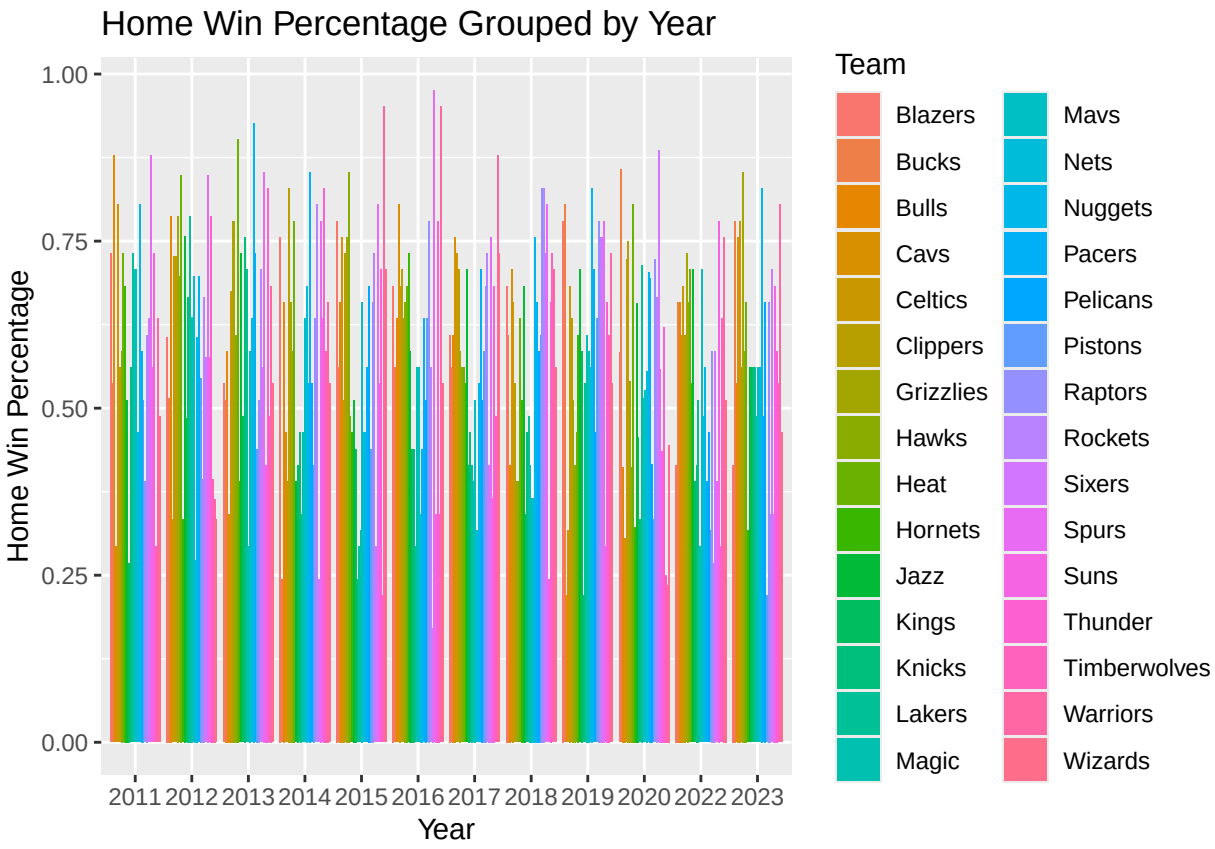
v7


Distribution of Home Average Attendance by Team
Home Average
22500
20000
17500
15000
Blazers
Bucks
Bulls
Cavs
Celtics
Clippers
Grizzlies
Hawks
Heat
Hornets
Jazz
Kings
Knicks
Lakers
Magic
Mavs
Nets
Nuggets
Pacers
Pelicans
Pistons
Raptors
Rockets
Sixers
Spurs
Suns
Thunder
Timberwolves
Warriors
Wizards
Team

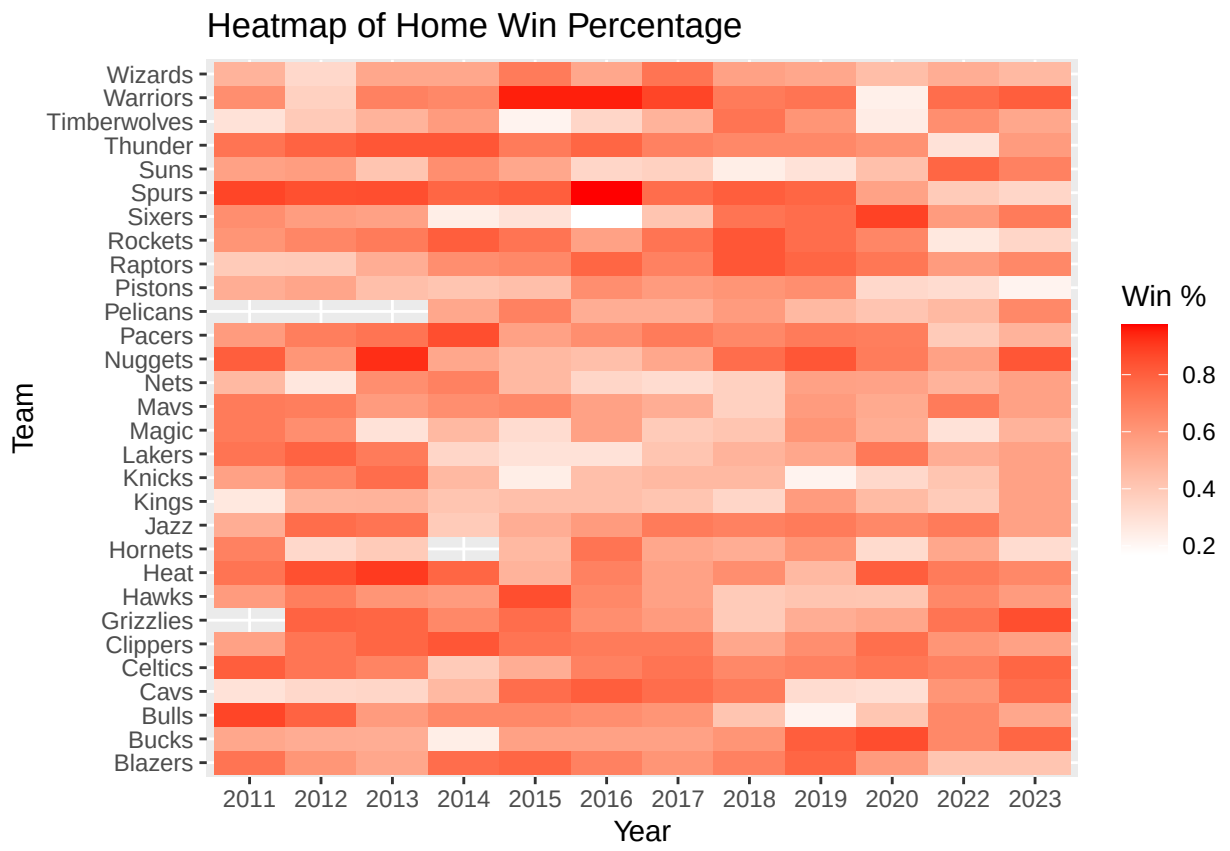
v8


Home Win Percentage Over Time
Home Win Percentage
1.00
0.75
0.50
0.25
2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2022 2023
Year
Team
Blazers
Bucks
Bulls
Cavs
Celtics
Clippers
Grizzlies
Hawks
Heat
Hornets
Jazz
Kings
Knicks
Lakers
Magic
Mavs
Nets
Nuggets
Pacers
Pelicans
Pistons
Raptors
Rockets
Sixers
Spurs
Suns
Thunder
Timberwolves
Warriors
Wizards

v9


Home Win Percentage Grouped by Year
Home Win Percentage
1.00
0.75
0.50
0.25
0.00
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2022
2023
Year
Team
Blazers
Bucks
Bulls
Cavs
Celtics
Clippers
Grizzlies
Hawks
Heat
Hornets
Jazz
Kings
Knicks
Lakers
Magic
Mavs
Nets
Nuggets
Pacers
Pelicans
Pistons
Raptors
Rockets
Sixers
Spurs
Suns
Thunder
Timberwolves
Warriors
Wizards

```
v10
```

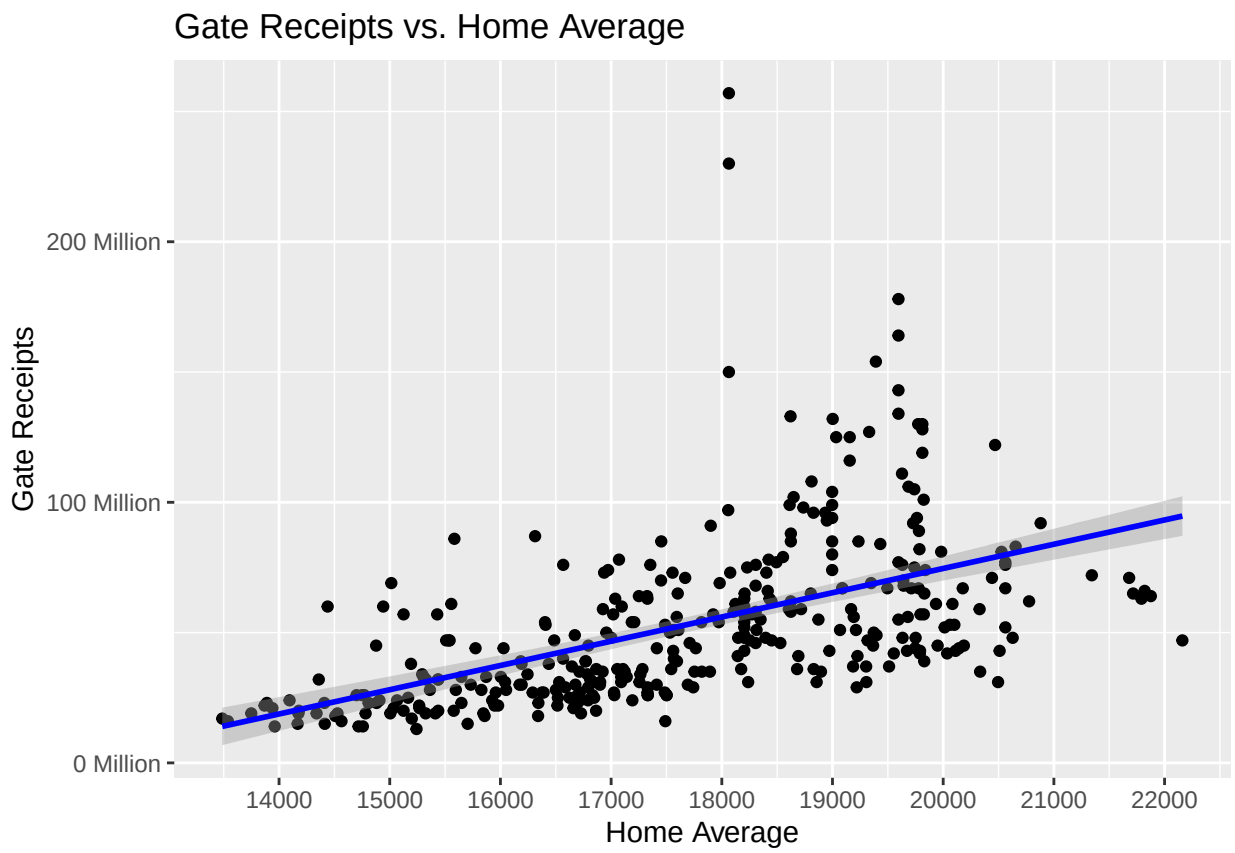


more

```r
# Ensure Gate_Receipts are numeric
hc_combined_dataset <- hc_combined_dataset %>%
  mutate(Gate_Receipts = as.numeric(Gate_Receipts))

# Scatterplot of Gate Receipts vs. Home Average
plot1 <- ggplot(hc_combined_dataset, aes(x = Home_Average, y = Gate_Receipts)) +
  geom_point() +
  geom_smooth(method = "lm", col = "blue") +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_y_continuous(labels = function(x) paste0(x / 1e6, " Million")) +
  labs(title = "Gate Receipts vs. Home Average", x = "Home Average", y = "Gate Receipts")

# Scatterplot of Gate Receipts vs. Home Total
plot2 <- ggplot(hc_combined_dataset, aes(x = Home_Total, y = Gate_Receipts)) +
  geom_point() +
  geom_smooth(method = "lm", col = "green") +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_y_continuous(labels = function(x) paste0(x / 1e6, " Million")) +
  labs(title = "Gate Receipts vs. Home Total", x = "Home Total", y = "Gate Receipts")

# Scatterplot of Gate Receipts vs. Home Win Percentage
plot3 <- ggplot(hc_combined_dataset, aes(x = Home_Win_Percentage, y = Gate_Receipts)) +
  geom_point() +
  geom_smooth(method = "lm", col = "red") +
```

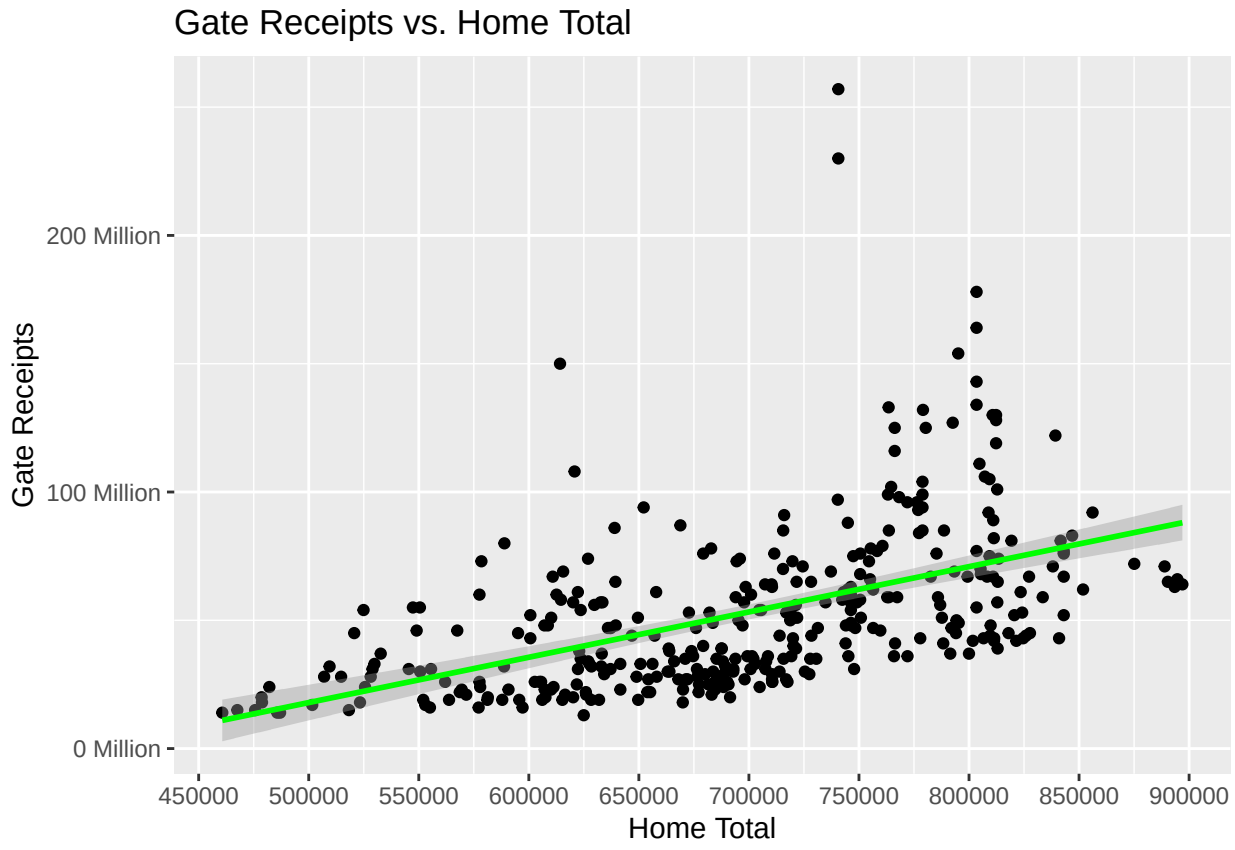
```r
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_y_continuous(labels = function(x) paste0(x / 1e6, " Million")) +
  labs(title = "Gate Receipts vs. Home Win Percentage", x = "Home Win Percentage", y = "Gate Receipts")

# Scatterplot of Home Average vs. Home Win Percentage
plot4 <- ggplot(hc_combined_dataset, aes(x = Home_Average, y = Home_Win_Percentage)) +
  geom_point() +
  geom_smooth(method = "lm", col = "purple") +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 10)) +
  labs(title = "Home Average vs. Home Win Percentage", x = "Home Average", y = "Home Win Percentage")

# Scatterplot of Home Total vs. Home Win Percentage
plot5 <- ggplot(hc_combined_dataset, aes(x = Home_Total, y = Home_Win_Percentage)) +
  geom_point() +
  geom_smooth(method = "lm", col = "orange") +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 10)) +
  labs(title = "Home Total vs. Home Win Percentage", x = "Home Total", y = "Home Win Percentage")

# Display the plots
plot1
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



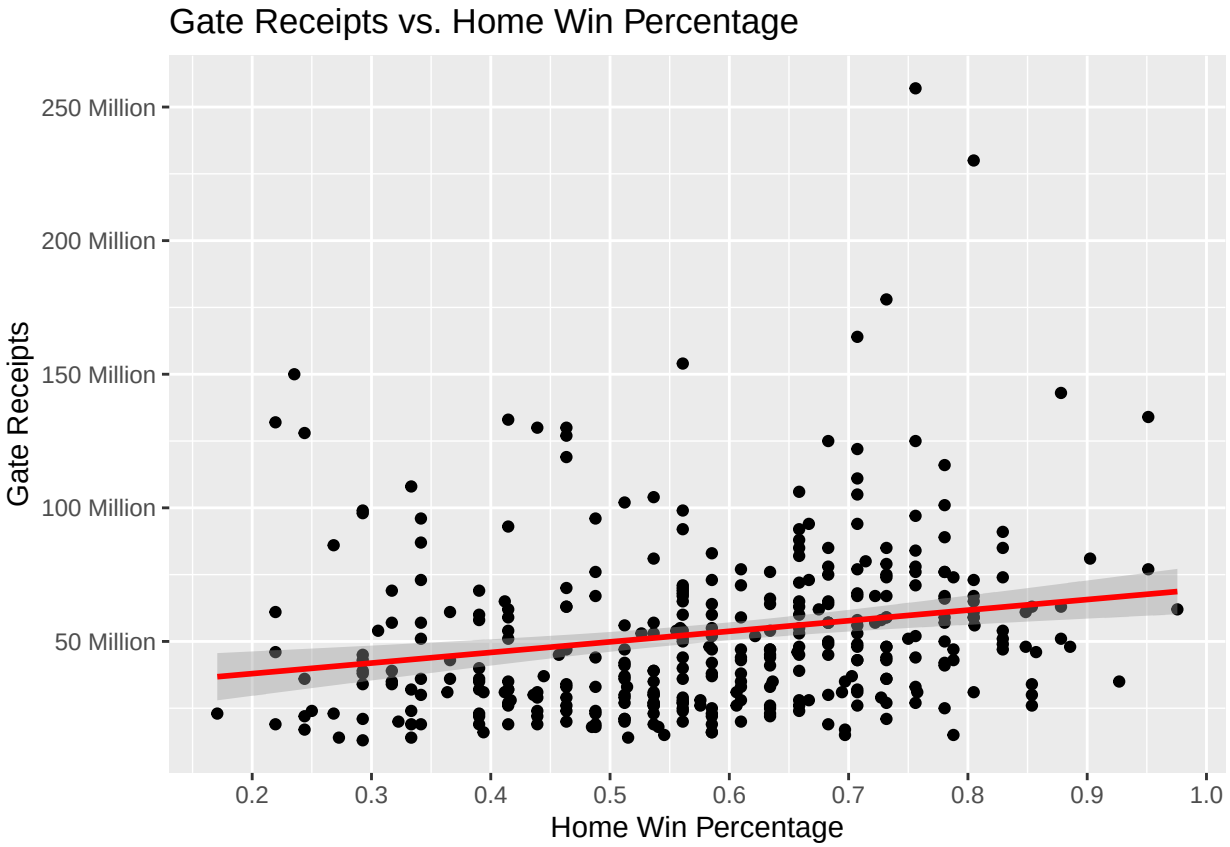
```
plot2
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



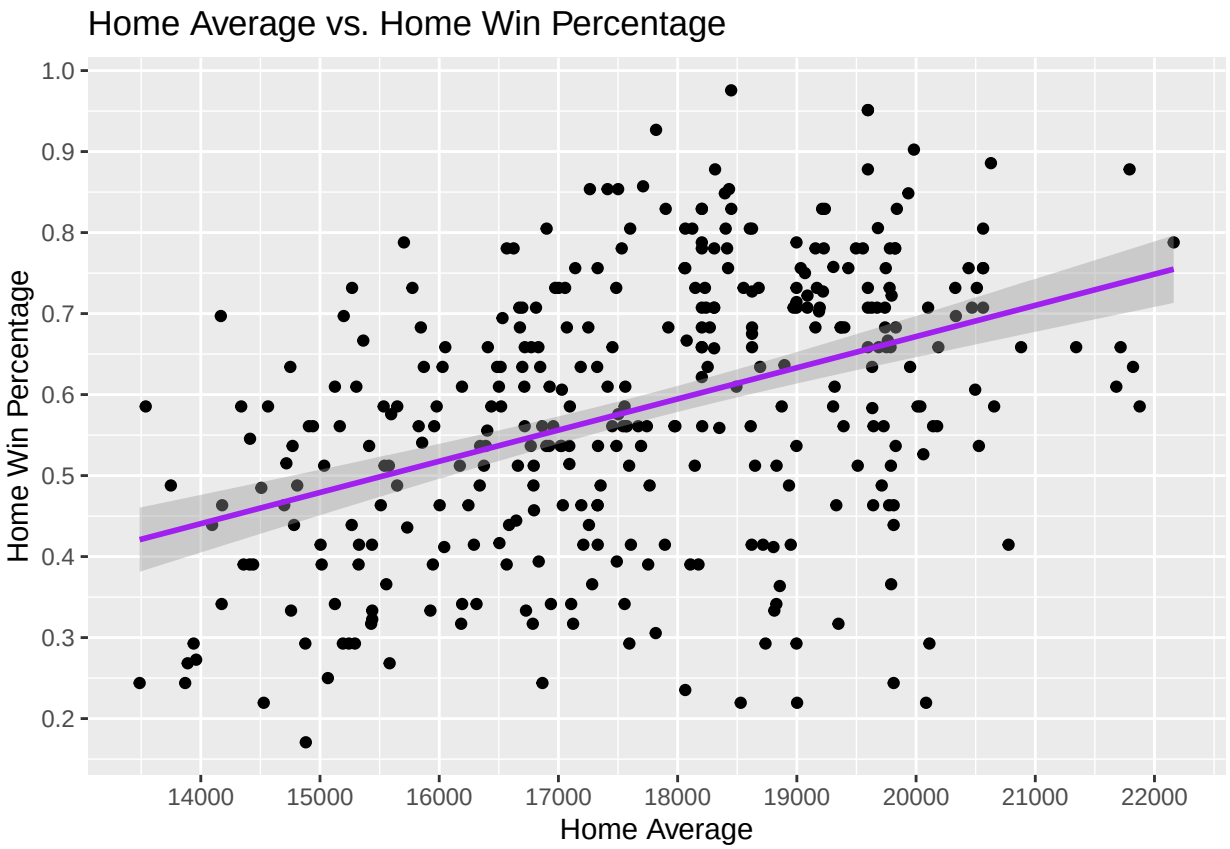

```
plot3
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



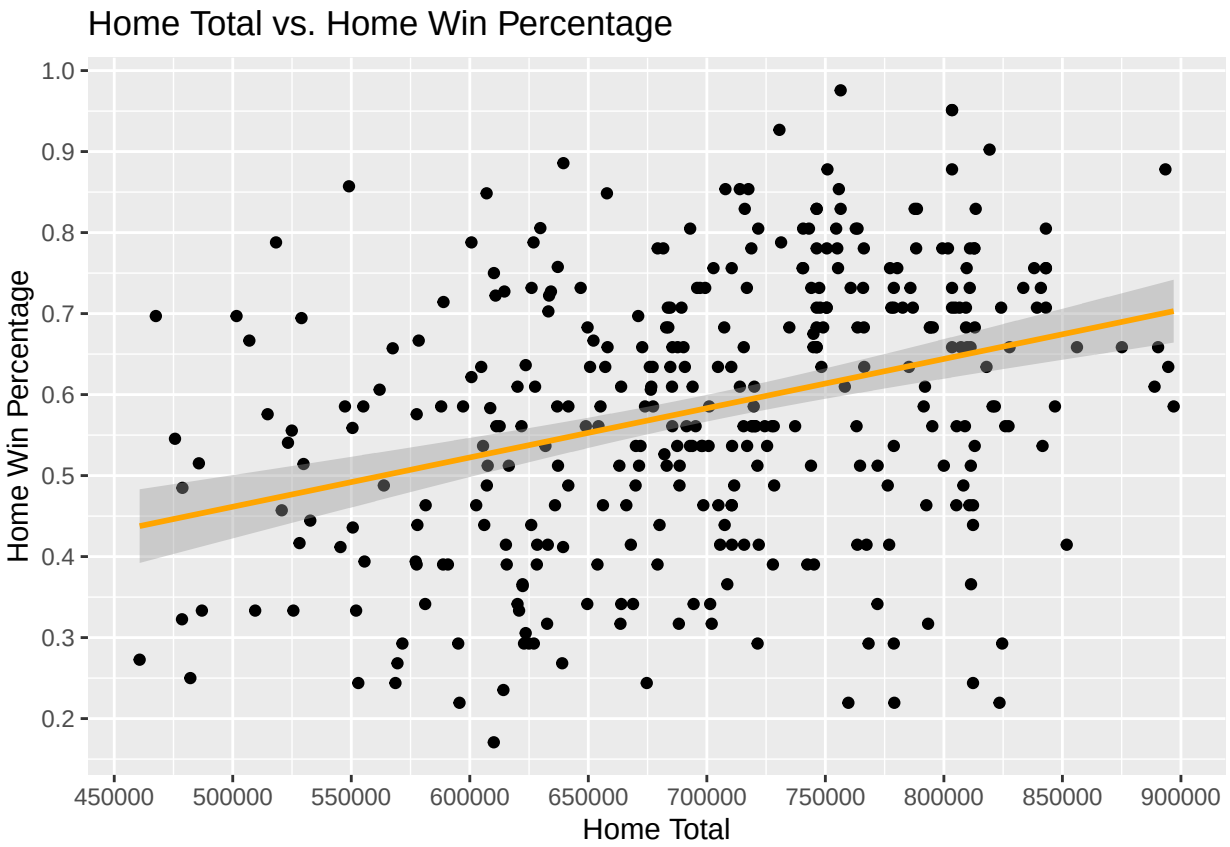

```
plot4
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
plot5
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Model Time

Linear Regression

```
reg1 <- lm(Home_Win_Percentage ~ Home_Total, data = hc_combined_dataset)
reg2 <- lm(Home_Win_Percentage ~ Gate_Receipts, data = hc_combined_dataset)
reg3 <- lm(Home_Win_Percentage ~ Home_Total + Gate_Receipts, data = hc_combined_dataset)
reg4 <- lm(Home_Total ~ Gate_Receipts, data = hc_combined_dataset)
reg5 <- lm(Gate_Receipts ~ Home_Total, data = hc_combined_dataset)
reg6 <- lm(Home_Total ~ Gate_Receipts + PCT, data = hc_combined_dataset)
reg7 <- lm(Gate_Receipts ~ Home_Total + PCT, data = hc_combined_dataset)

summary(reg1)
```

```
##
## Call:
## lm(formula = Home_Win_Percentage ~ Home_Total, data = hc_combined_dataset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.43880 -0.11469  0.01438  0.10763  0.36579
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.574e-01  6.376e-02   2.468   0.0141 *
## Home_Total  6.083e-07  9.041e-08   6.729 6.91e-11 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1578 on 353 degrees of freedom
## Multiple R-squared:  0.1137, Adjusted R-squared:  0.1112
## F-statistic: 45.28 on 1 and 353 DF,  p-value: 6.913e-11
```

```
summary(reg2)
```

```
##
## Call:
## lm(formula = Home_Win_Percentage ~ Gate_Receipts, data = hc_combined_dataset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.44935 -0.11767  0.01435  0.12482  0.38356
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.268e-01  1.671e-02  31.522  < 2e-16 ***
## Gate_Receipts 1.052e-09  2.686e-10   3.917 0.000107 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1641 on 353 degrees of freedom
## Multiple R-squared:  0.04166,    Adjusted R-squared:  0.03895
## F-statistic: 15.35 on 1 and 353 DF,  p-value: 0.0001074
```

```
summary(reg3)
```

```
##
## Call:
## lm(formula = Home_Win_Percentage ~ Home_Total + Gate_Receipts,
##     data = hc_combined_dataset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.43549 -0.11010  0.01762  0.10433  0.36123
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.739e-01  6.719e-02   2.588    0.010 *
## Home_Total    5.668e-07  1.048e-07   5.410 1.16e-07 ***
## Gate_Receipts 2.350e-10  2.993e-10   0.785    0.433
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1579 on 352 degrees of freedom
## Multiple R-squared:  0.1152, Adjusted R-squared:  0.1102
## F-statistic: 22.92 on 2 and 352 DF,  p-value: 4.379e-10
```

```r
summary(reg4)
```

```
##
## Call:
## lm(formula = Home_Total ~ Gate_Receipts, data = hc_combined_dataset)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -252493  -49038    9068   46875  182056
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.226e+05  8.170e+03   76.21   <2e-16 ***
## Gate_Receipts 1.442e-03  1.313e-04   10.98   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 80220 on 353 degrees of freedom
## Multiple R-squared:  0.2546, Adjusted R-squared:  0.2525
## F-statistic: 120.6 on 1 and 353 DF,  p-value: < 2.2e-16
```

```r
#summary(reg5)
#summary(reg6)
#summary(reg7)
```

Logistic Regression

```r
log_reg1 <- glm(Home_Win_Loss ~ Home_Total, data = hc_combined_dataset, family = binomial)
log_reg2 <- glm(Home_Win_Loss ~ Gate_Receipts, data = hc_combined_dataset, family = binomial)
log_reg3 <- glm(Home_Win_Loss ~ Home_Total + Gate_Receipts, data = hc_combined_dataset, family = binomia

summary(log_reg1)
```

```
##
## Call:
## glm(formula = Home_Win_Loss ~ Home_Total, family = binomial,
##     data = hc_combined_dataset)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.864e+00  9.244e-01  -4.180 2.92e-05 ***
## Home_Total   6.853e-06  1.349e-06   5.079 3.80e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 432.86  on 354  degrees of freedom
## Residual deviance: 404.70  on 353  degrees of freedom
## AIC: 408.7
##
## Number of Fisher Scoring iterations: 4
```

```r
summary(log_reg2)
```

```
##
## Call:
## glm(formula = Home_Win_Loss ~ Gate_Receipts, family = binomial,
##     data = hc_combined_dataset)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   4.550e-01  2.345e-01   1.940   0.0523 .
## Gate_Receipts 7.765e-09  4.096e-09   1.896   0.0580 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 432.86  on 354  degrees of freedom
## Residual deviance: 428.89  on 353  degrees of freedom
## AIC: 432.89
##
## Number of Fisher Scoring iterations: 4
```

```r
summary(log_reg3)
```

```
##
## Call:
## glm(formula = Home_Win_Loss ~ Home_Total + Gate_Receipts, family = binomial,
##     data = hc_combined_dataset)
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -4.144e+00  9.797e-01  -4.229 2.35e-05 ***
## Home_Total     7.550e-06  1.566e-06   4.823 1.42e-06 ***
## Gate_Receipts -3.877e-09  4.275e-09  -0.907    0.364
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 432.86  on 354  degrees of freedom
## Residual deviance: 403.90  on 352  degrees of freedom
## AIC: 409.9
##
## Number of Fisher Scoring iterations: 4
```

Random Forest

```r
hc_combined_dataset$Home_Win_Loss <- as.factor(hc_combined_dataset$Home_Win_Loss)

rf1 <- randomForest(Home_Win_Loss ~ Home_Total, data = hc_combined_dataset, ntree = 500)
rf2 <- randomForest(Home_Win_Loss ~ Gate_Receipts, data = hc_combined_dataset, ntree = 500)
rf3 <- randomForest(Home_Win_Loss ~ Home_Total + Gate_Receipts, data = hc_combined_dataset, ntree = 500)
```

```r
print(rf1)
```

```
##
## Call:
##  randomForest(formula = Home_Win_Loss ~ Home_Total, data = hc_combined_dataset,      ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 38.31%
## Confusion matrix:
##    0   1 class.error
## 0 37  69   0.6509434
## 1 67 182   0.2690763
```

```r
print(rf2)
```

```
##
## Call:
##  randomForest(formula = Home_Win_Loss ~ Gate_Receipts, data = hc_combined_dataset,      ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 33.8%
## Confusion matrix:
##    0   1 class.error
## 0 40  66   0.6226415
## 1 54 195   0.2168675
```

```r
print(rf3)
```

```
##
## Call:
##  randomForest(formula = Home_Win_Loss ~ Home_Total + Gate_Receipts,      data = hc_combined_dataset,
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 32.11%
## Confusion matrix:
##    0   1 class.error
## 0 36  70   0.6603774
## 1 44 205   0.1767068
```

RF split data

```r
set.seed(99)
```

```r
train_index_rf <- createDataPartition(hc_combined_dataset$Home_Win_Loss, p = 0.8, list = FALSE)
train_data_rf <- hc_combined_dataset[train_index_rf, ]
test_data_rf <- hc_combined_dataset[-train_index_rf, ]

control <- trainControl(method = "cv", number = 5, search = "grid")

grid <- expand.grid(mtry = c(1, 2))

rf_grid <- train(Home_Win_Loss ~ Home_Total + Gate_Receipts, data = train_data_rf,
                 method = "rf",
                 trControl = control,
                 tuneGrid = grid,
                 ntree = 500)  # Added ntree parameter here



pred <- predict(rf_grid, test_data_rf)

conf_matrix <- confusionMatrix(pred, test_data_rf$Home_Win_Loss)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  8 10
##          1 13 39
##
##                Accuracy : 0.6714
##                  95% CI : (0.5488, 0.7791)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.7459
##
##                   Kappa : 0.1844
##
##  Mcnemar's Test P-Value : 0.6767
##
##             Sensitivity : 0.3810
##             Specificity : 0.7959
##          Pos Pred Value : 0.4444
##          Neg Pred Value : 0.7500
##              Prevalence : 0.3000
##          Detection Rate : 0.1143
##    Detection Prevalence : 0.2571
##       Balanced Accuracy : 0.5884
##
##        'Positive' Class : 0
##
```

XG Boost Binary

```r
set.seed(99)
train_index_xg <- createDataPartition(hc_combined_dataset$Home_Win_Loss, p = 0.8, list = FALSE)
train_data_xg <- hc_combined_dataset[train_index_xg, ]
test_data_xg <- hc_combined_dataset[-train_index_xg, ]

train_matrix <- xgb.DMatrix(data = as.matrix(train_data_xg[, c("Home_Total", "Gate_Receipts")]), label =
test_matrix <- xgb.DMatrix(data = as.matrix(test_data_xg[, c("Home_Total", "Gate_Receipts")]), label = t

params <- list(
  booster = "gbtree",
  objective = "binary:logistic",
  eta = 0.05,
  max_depth = 10,
  eval_metric = "error"
)

xgb_model <- xgboost(data = train_matrix, nrounds = 500, verbose = 1, print_every_n = 50)
```

```
## [1]  train-rmse:0.948989
## [51] train-rmse:0.089465
## [101]    train-rmse:0.028328
## [151]    train-rmse:0.010194
## [201]    train-rmse:0.003733
## [251]    train-rmse:0.003197
## [301]    train-rmse:0.003197
## [351]    train-rmse:0.003197
## [401]    train-rmse:0.003197
## [451]    train-rmse:0.003197
## [500]    train-rmse:0.003197
```

```r
pred <- predict(xgb_model, test_matrix)
pred_label <- ifelse(pred > 0.5, 1, 0)

conf_matrix <- confusionMatrix(as.factor(pred_label), as.factor(test_data_xg$Home_Win_Loss))
```

```
## Warning in confusionMatrix.default(as.factor(pred_label),
## as.factor(test_data_xg$Home_Win_Loss)): Levels are not in the same order for
## reference and data. Refactoring data to match.
```

```r
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  0  0
##          1 21 49
##
##                Accuracy : 0.7
##                  95% CI : (0.5787, 0.8038)
##     No Information Rate : 0.7
```

```
##     P-Value [Acc > NIR] : 0.5586
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 1.275e-05
##
##             Sensitivity : 0.0
##             Specificity : 1.0
##          Pos Pred Value : NaN
##          Neg Pred Value : 0.7
##              Prevalence : 0.3
##          Detection Rate : 0.0
##    Detection Prevalence : 0.0
##       Balanced Accuracy : 0.5
##
##        'Positive' Class : 0
##
```

Regression

```
set.seed(99)
train_index_xg <- createDataPartition(hc_combined_dataset$Home_Win_Percentage, p = 0.8, list = FALSE)
train_data_xg <- hc_combined_dataset[train_index_xg, ]
test_data_xg <- hc_combined_dataset[-train_index_xg, ]

train_matrix <- xgb.DMatrix(data = as.matrix(train_data_xg[, c("Home_Total", "Gate_Receipts")]), label =
test_matrix <- xgb.DMatrix(data = as.matrix(test_data_xg[, c("Home_Total", "Gate_Receipts")]), label = t

params <- list(
  booster = "gbtree",
  objective = "reg:squarederror",
  eta = 0.05,
  max_depth = 10,
  subsample = 0.8,
  colsample_bytree = 0.8,
  alpha = 0.1, # L1 regularization term
  lambda = 0.1, # L2 regularization term
  eval_metric = "rmse"
)

cv <- xgb.cv(
  params = params,
  data = train_matrix,
  nrounds = 1000,
  nfold = 5,
  showsd = TRUE,
  stratified = TRUE,
  print_every_n = 50,
  early_stopping_rounds = 10,
  maximize = FALSE
)
```

```
## [1]  train-rmse:0.181865+0.003786    test-rmse:0.183835+0.014592
```

```
## Multiple eval metrics are present. Will use test_rmse for early stopping.
## Will train until test_rmse hasn't improved in 10 rounds.
##
## Stopping. Best iteration:
## [35] train-rmse:0.104279+0.001720    test-rmse:0.160696+0.012073
```

```
best_nrounds <- cv$best_iteration
xgb_model <- xgboost(
  data = train_matrix,
  params = params,
  nrounds = best_nrounds,
  verbose = 1,
  print_every_n = 50
)
```

```
## [1]  train-rmse:0.181311
## [35] train-rmse:0.107531
```

```
pred <- predict(xgb_model, test_matrix)

actual <- test_data_xg$Home_Win_Percentage
rmse_value <- rmse(actual, pred)
cat("RMSE: ", rmse_value, "\n")
```

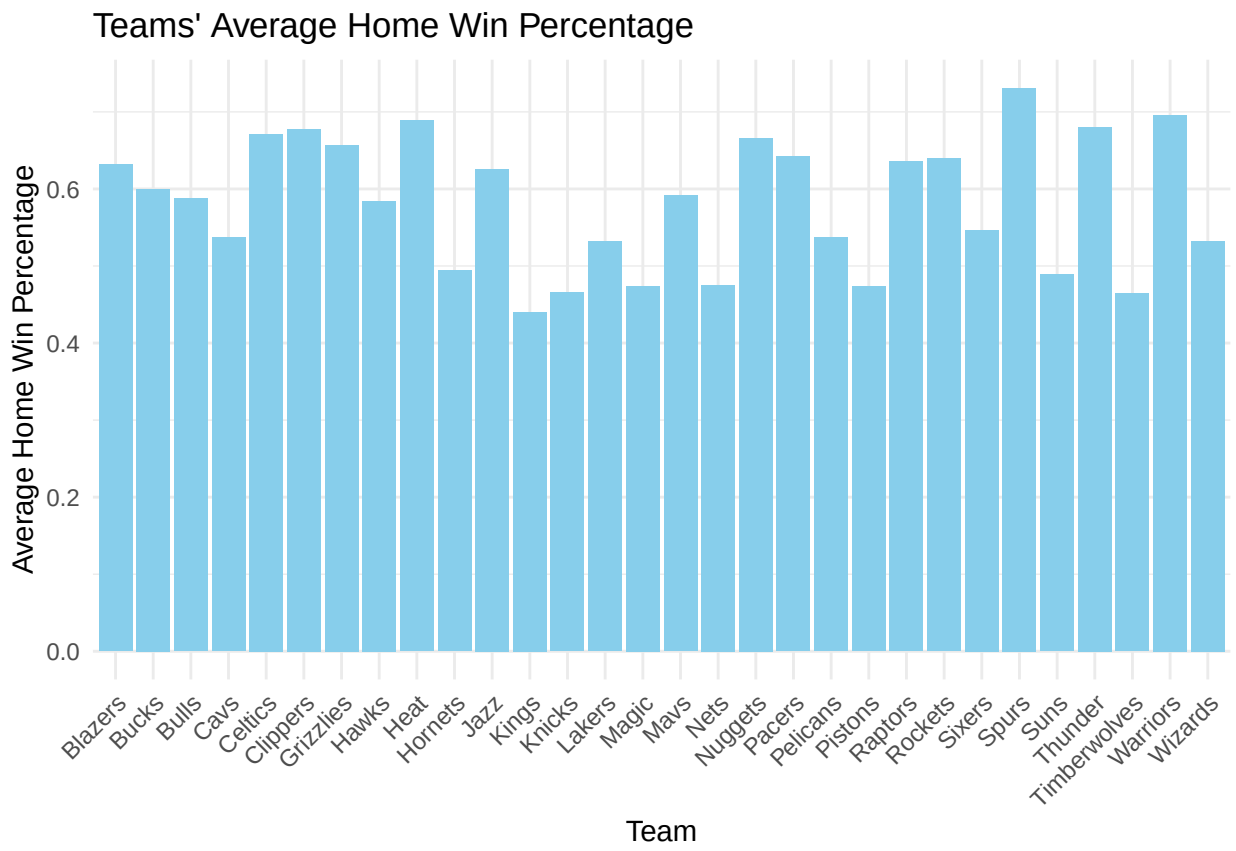
```
## RMSE:  0.1621745
```

```
rmse_value
```

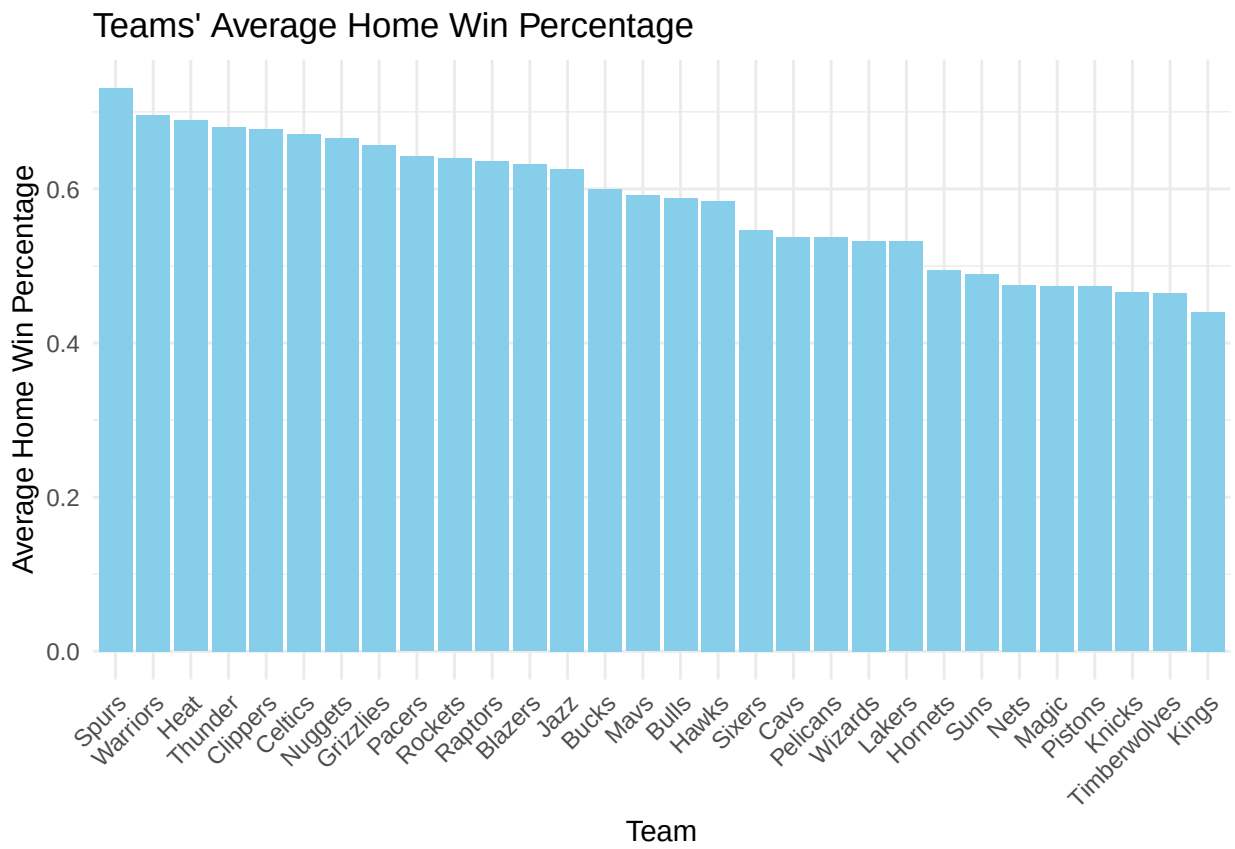
```
## [1] 0.1621745
```

graphs

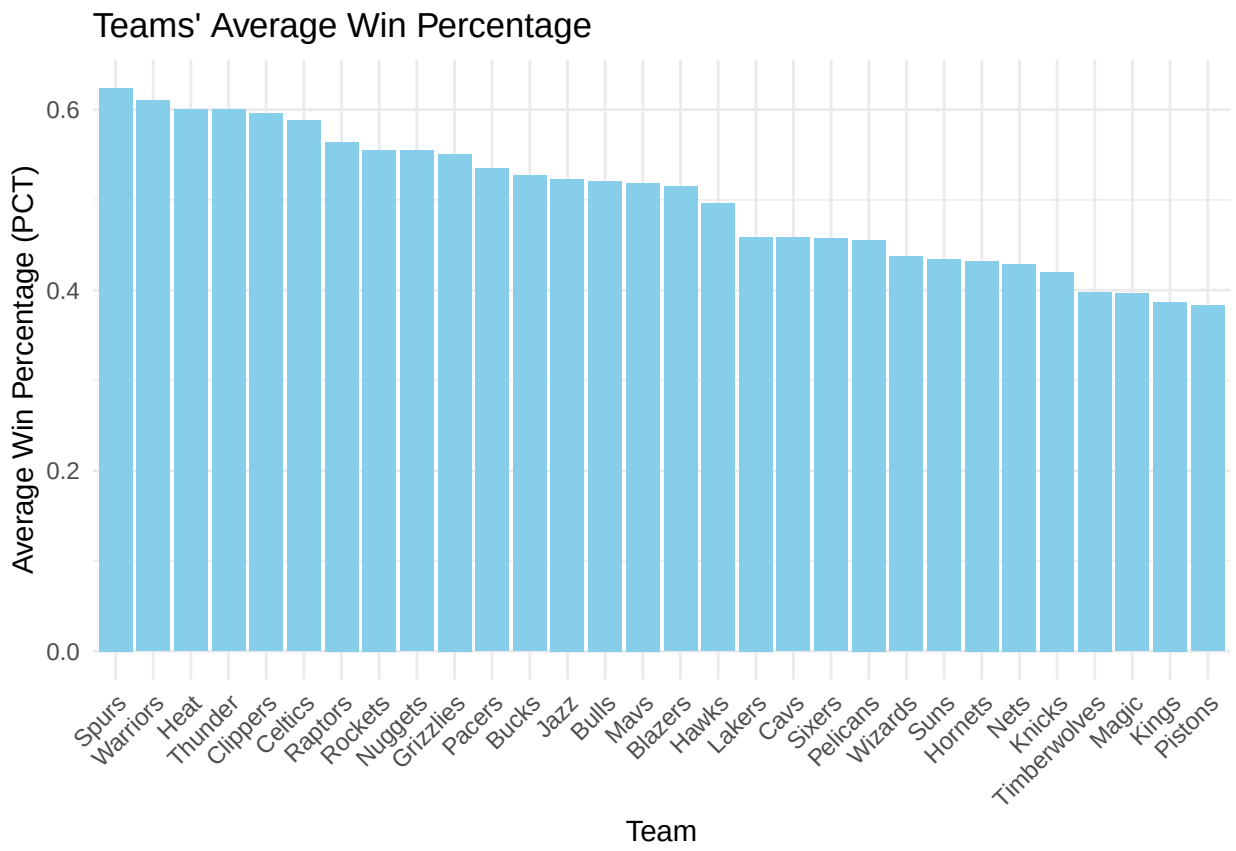
```
agg_data <- aggregate(Home_Win_Percentage ~ Team, data = hc_combined_dataset, mean)

ggplot(agg_data, aes(x = Team, y = Home_Win_Percentage)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Teams' Average Home Win Percentage",
       x = "Team",
       y = "Average Home Win Percentage") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



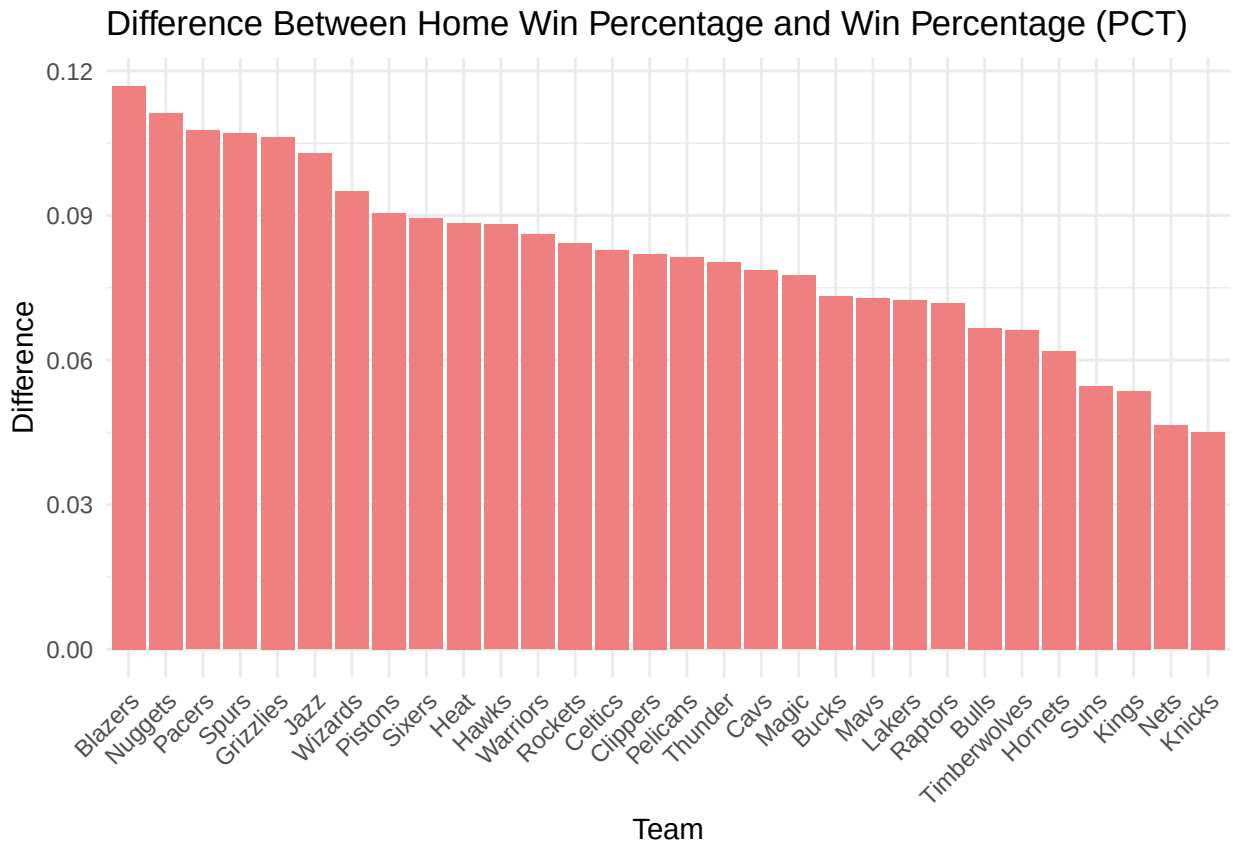

```r
agg_data <- aggregate(Home_Win_Percentage ~ Team, data = hc_combined_dataset, mean)
agg_data <- agg_data[order(-agg_data$Home_Win_Percentage), ]  # Sort in descending order

ggplot(agg_data, aes(x = reorder(Team, -Home_Win_Percentage), y = Home_Win_Percentage)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Teams' Average Home Win Percentage",
       x = "Team",
       y = "Average Home Win Percentage") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
agg_data <- aggregate(PCT ~ Team, data = hc_combined_dataset, mean)
agg_data <- agg_data[order(-agg_data$PCT), ]  # Sort in descending order

ggplot(agg_data, aes(x = reorder(Team, -PCT), y = PCT)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Teams' Average Win Percentage",
       x = "Team",
       y = "Average Win Percentage (PCT)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



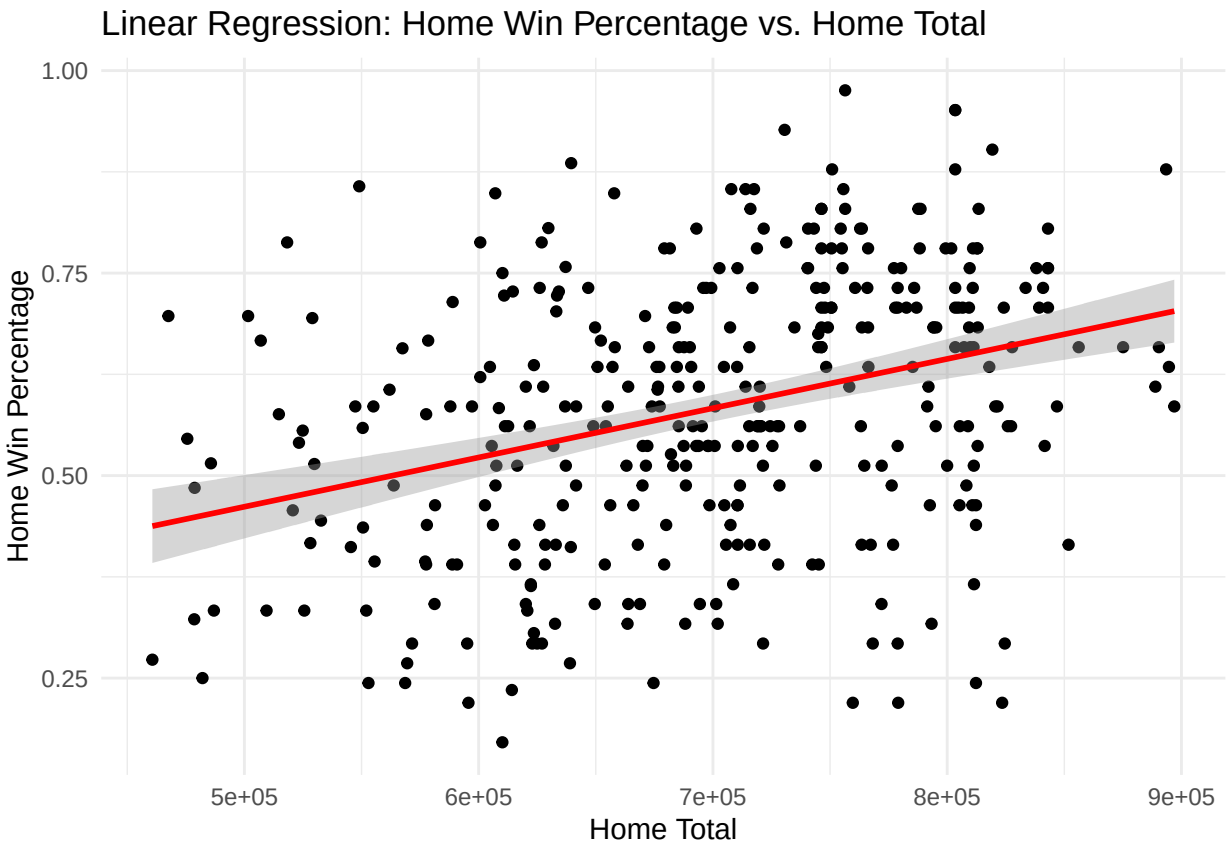

```r
agg_data <- aggregate(cbind(Home_Win_Percentage, PCT) ~ Team, data = hc_combined_dataset, mean)

agg_data$Difference <- agg_data$Home_Win_Percentage - agg_data$PCT

ggplot(agg_data, aes(x = reorder(Team, -Difference), y = Difference)) +
  geom_bar(stat = "identity", fill = "lightcoral") +
  labs(title = "Difference Between Home Win Percentage and Win Percentage (PCT)",
       x = "Team",
       y = "Difference") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
agg_data1 <- aggregate(Home_Win_Percentage ~ Team, data = hc_combined_dataset, mean)
agg_data2 <- aggregate(PCT ~ Team, data = hc_combined_dataset, mean)
agg_data3 <- aggregate(cbind(Home_Win_Percentage, PCT) ~ Team, data = hc_combined_dataset, mean)
agg_data4 <- aggregate(Home_Total ~ Gate_Receipts, data = hc_combined_dataset, mean)
agg_data5 <- aggregate(Gate_Receipts ~ Home_Total, data = hc_combined_dataset, mean)
#agg_data6 <- aggregate(Home_Total ~ cbind(Gate_Receipts, PCT), data = hc_combined_dataset, mean)
#agg_data7 <- aggregate(Gate_Receipts ~ cbind(Home_Total, PCT), data = hc_combined_dataset, mean)

p1 <- ggplot(hc_combined_dataset, aes(x = Home_Total, y = Home_Win_Percentage)) +
  geom_point() +
  geom_smooth(method = "lm", col = "red") +
  labs(title = "Linear Regression: Home Win Percentage vs. Home Total",
       x = "Home Total",
       y = "Home Win Percentage") +
  theme_minimal()

p2 <- ggplot(hc_combined_dataset, aes(x = Gate_Receipts, y = Home_Win_Percentage)) +
  geom_point() +
  geom_smooth(method = "lm", col = "red") +
  labs(title = "Linear Regression: Home Win Percentage vs. Gate Receipts",
       x = "Gate Receipts",
       y = "Home Win Percentage") +
  theme_minimal()

p3 <- ggplot(hc_combined_dataset, aes(x = Home_Total + Gate_Receipts, y = Home_Win_Percentage)) +
  geom_point() +
```

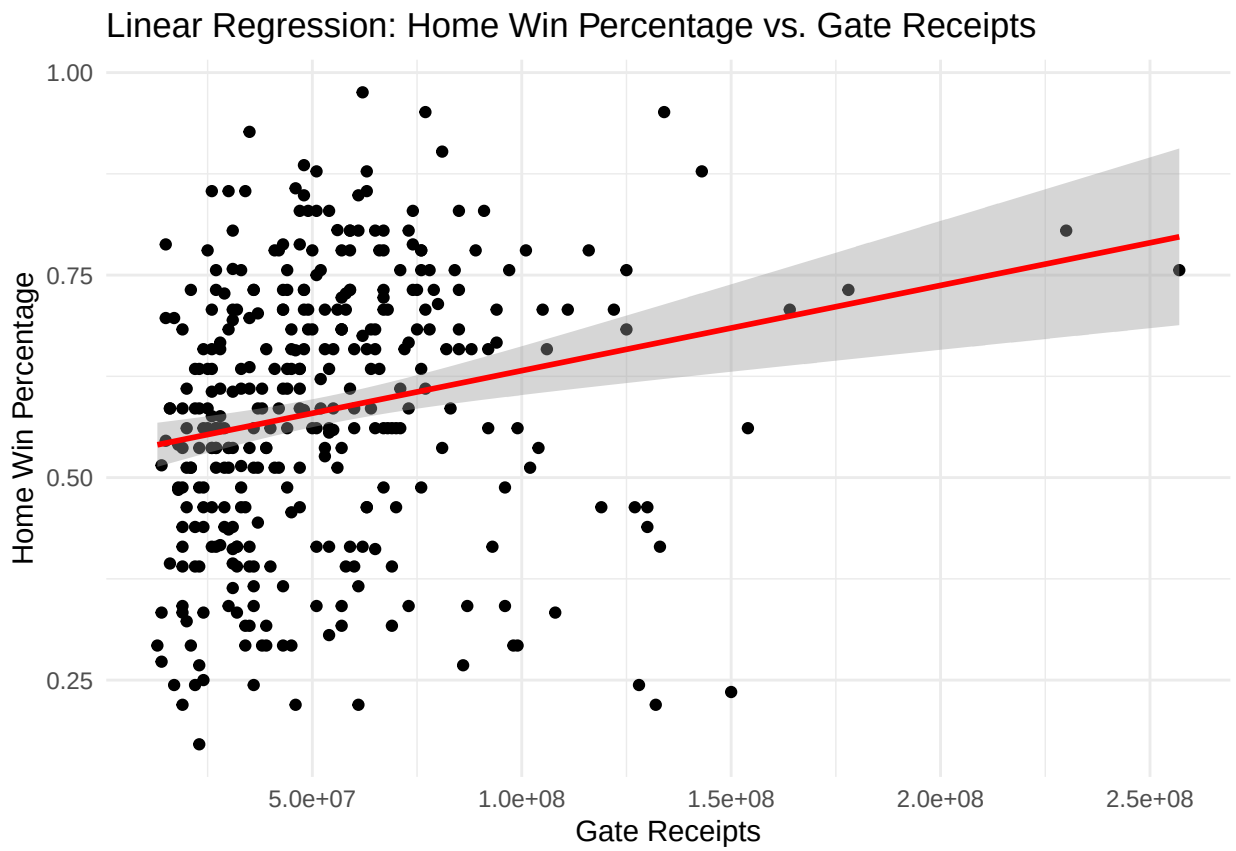
```
  geom_smooth(method = "lm", col = "red") +
  labs(title = "Linear Regression: Home Win Percentage vs. Home Total + Gate Receipts",
       x = "Home Total + Gate Receipts",
       y = "Home Win Percentage") +
  theme_minimal()

p4 <- ggplot(hc_combined_dataset, aes(x = Gate_Receipts, y = Home_Total)) +
  geom_point() +
  geom_smooth(method = "lm", col = "blue") +
  labs(title = "Linear Regression: Home Total vs. Gate Receipts",
       x = "Gate Receipts",
       y = "Home Total") +
  theme_minimal()

p5 <- ggplot(hc_combined_dataset, aes(x = Home_Total, y = Gate_Receipts)) +
  geom_point() +
  geom_smooth(method = "lm", col = "blue") +
  labs(title = "Linear Regression: Gate Receipts vs. Home Total",
       x = "Home Total",
       y = "Gate Receipts") +
  theme_minimal()

p6 <- ggplot(hc_combined_dataset, aes(x = Gate_Receipts + PCT, y = Home_Total)) +
  geom_point() +
  geom_smooth(method = "lm", col = "green") +
  labs(title = "Linear Regression: Home Total vs. Gate Receipts + PCT",
       x = "Gate Receipts + PCT",
       y = "Home Total") +
  theme_minimal()

p7 <- ggplot(hc_combined_dataset, aes(x = Home_Total + PCT, y = Gate_Receipts)) +
  geom_point() +
  geom_smooth(method = "lm", col = "green") +
  labs(title = "Linear Regression: Gate Receipts vs. Home Total + PCT",
       x = "Home Total + PCT",
       y = "Gate Receipts") +
  theme_minimal()

p1
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



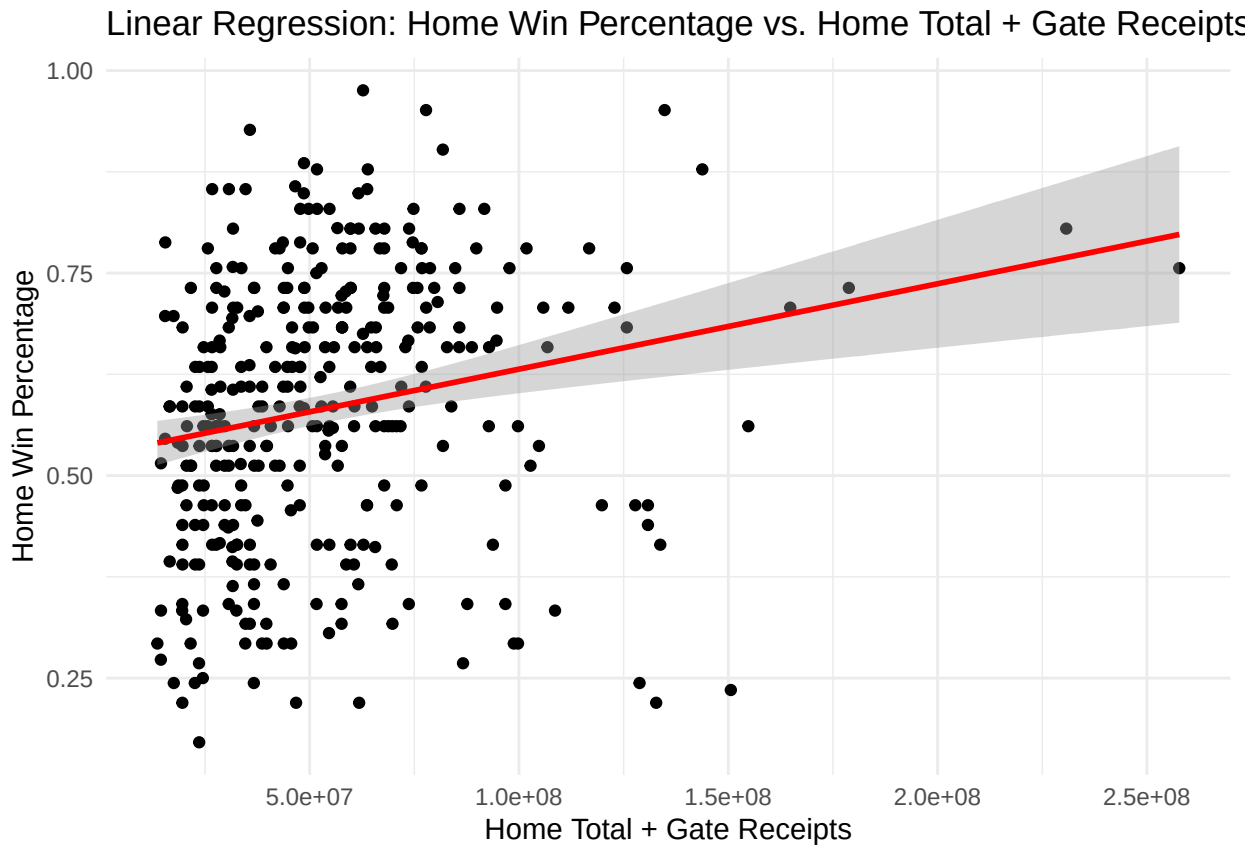

```
p2
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



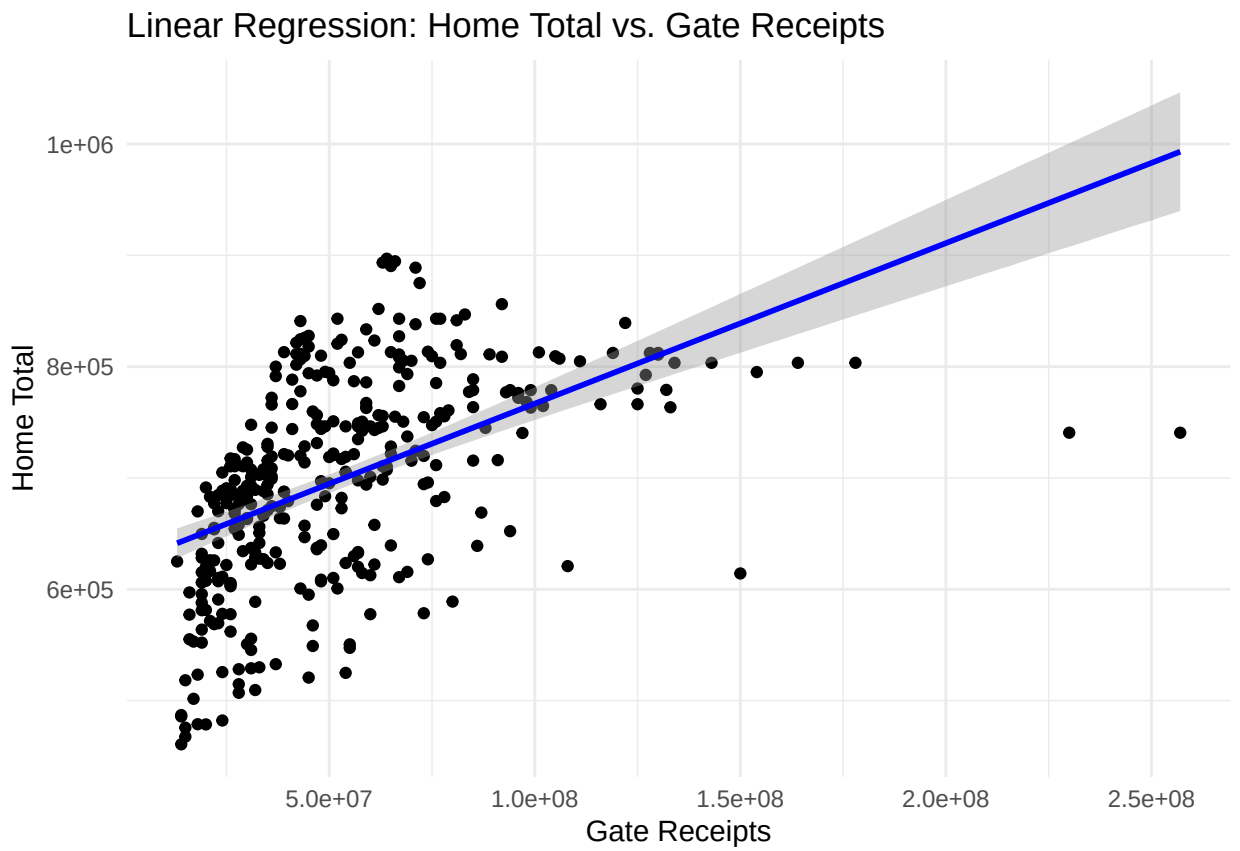

```
p3
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



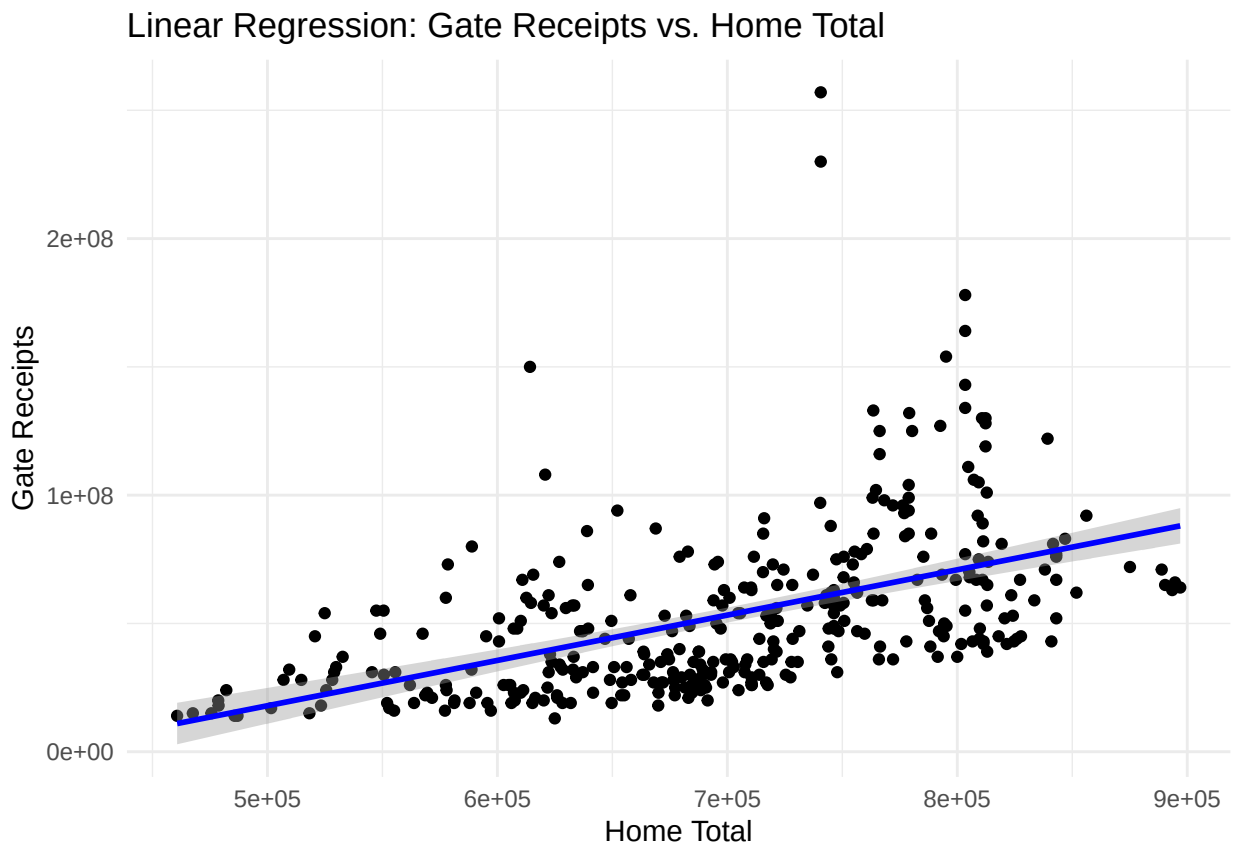

```
p4
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



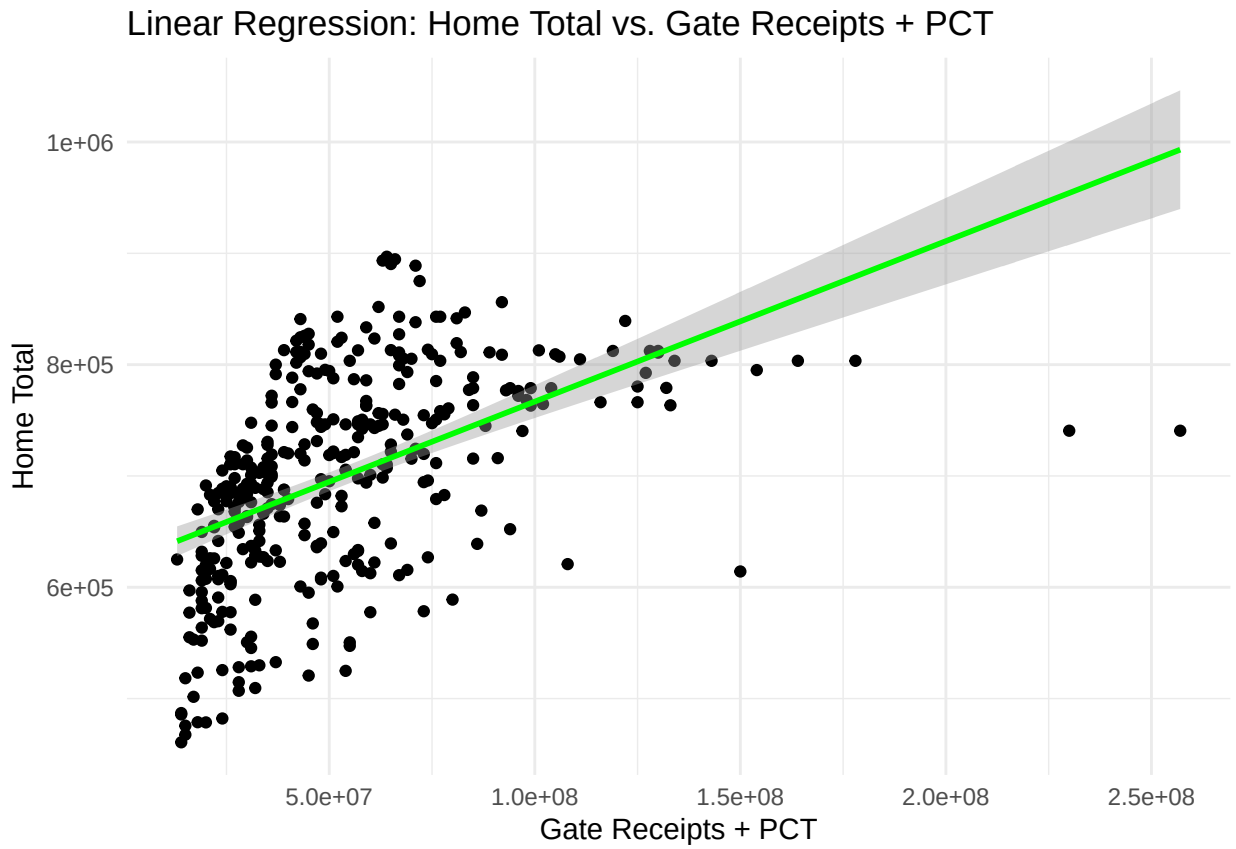

```
p5
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



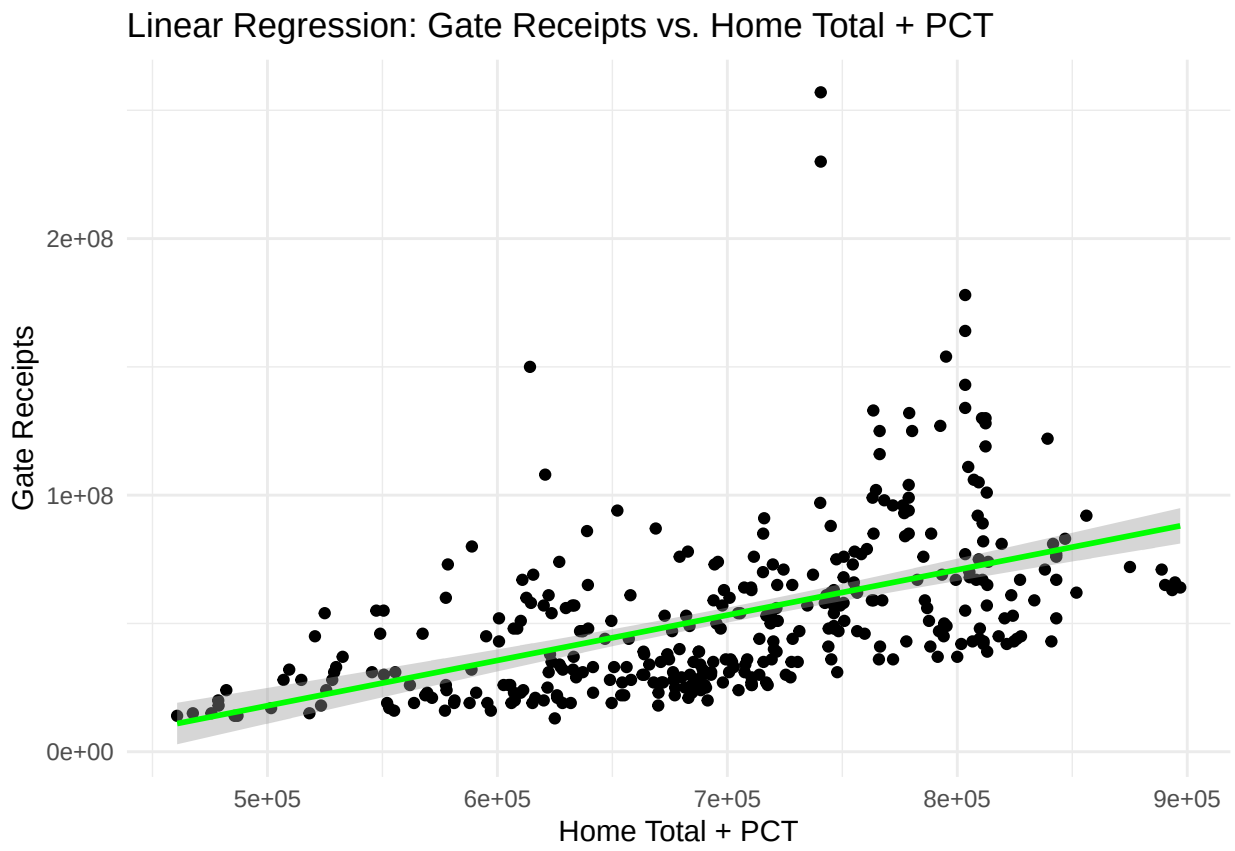

```
p6
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



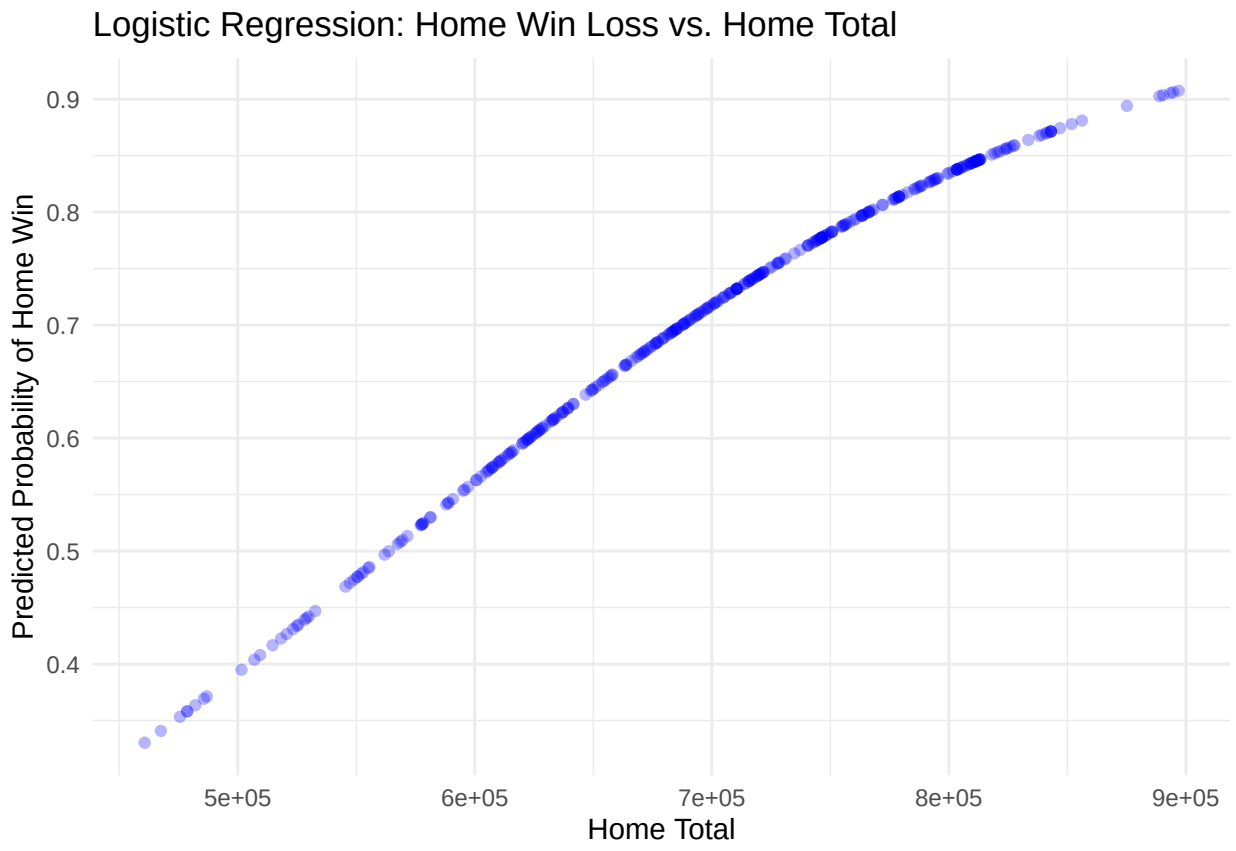

```
p7
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

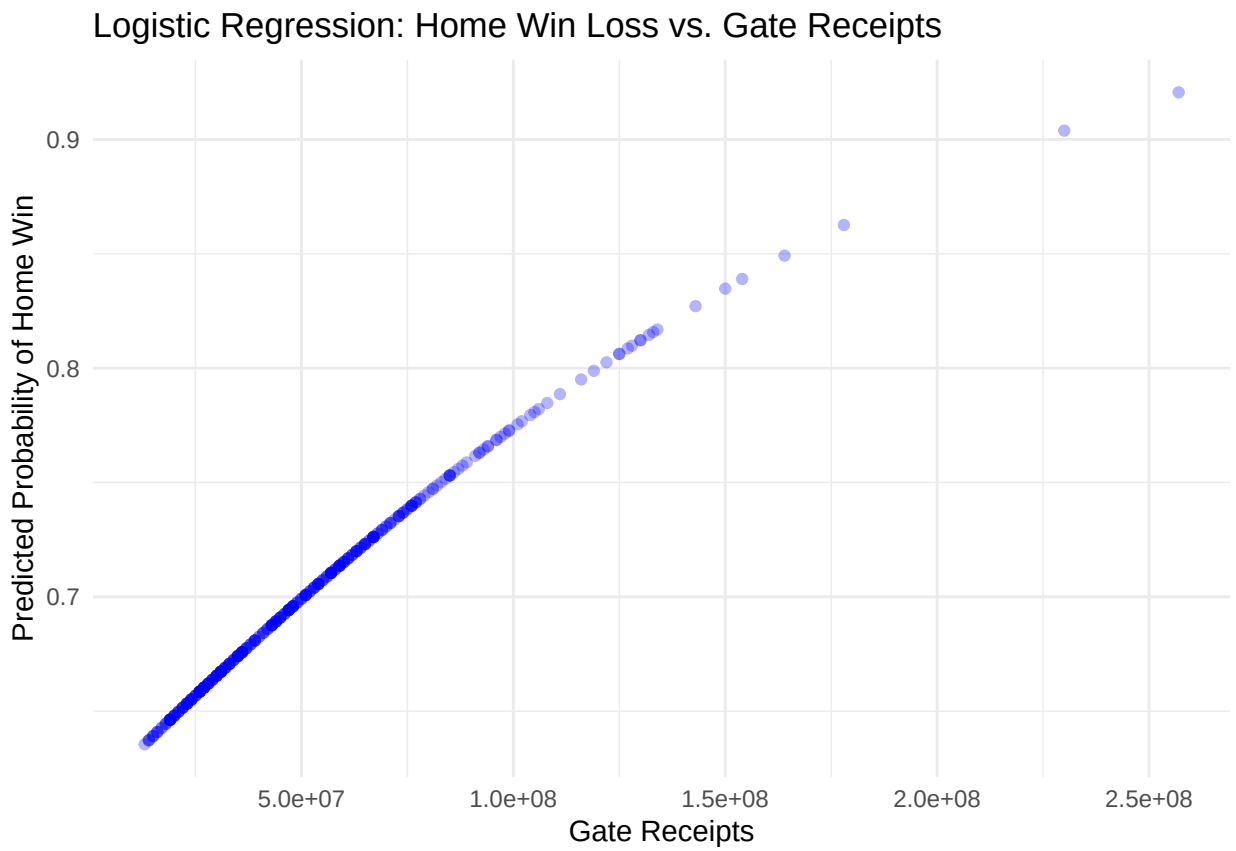



```
hc_combined_dataset$pred1 <- predict(log_reg1, type = "response")
hc_combined_dataset$pred2 <- predict(log_reg2, type = "response")
hc_combined_dataset$pred3 <- predict(log_reg3, type = "response")

p8 <- ggplot(hc_combined_dataset, aes(x = Home_Total, y = pred1)) +
  geom_point(alpha = 0.3, color = "blue") +
  #geom_smooth(method = "lm", col = "red") +
  labs(title = "Logistic Regression: Home Win Loss vs. Home Total",
       x = "Home Total",
       y = "Predicted Probability of Home Win") +
  theme_minimal()

p9 <- ggplot(hc_combined_dataset, aes(x = Gate_Receipts, y = pred2)) +
  geom_point(alpha = 0.3, color = "blue") +
  #geom_smooth(method = "lm", col = "red") +
  labs(title = "Logistic Regression: Home Win Loss vs. Gate Receipts",
       x = "Gate Receipts",
       y = "Predicted Probability of Home Win") +
  theme_minimal()

# Plot for log_reg3
p10 <- ggplot(hc_combined_dataset, aes(x = Home_Total + Gate_Receipts, y = pred3)) +
  geom_point(alpha = 0.3, color = "blue") +
  #geom_smooth(method = "lm", col = "red") +
  labs(title = "Logistic Regression: Home Win Loss vs. Home Total + Gate Receipts",
       x = "Home Total + Gate Receipts",
```

```
       y = "Predicted Probability of Home Win") +
  theme_minimal()

p8
```

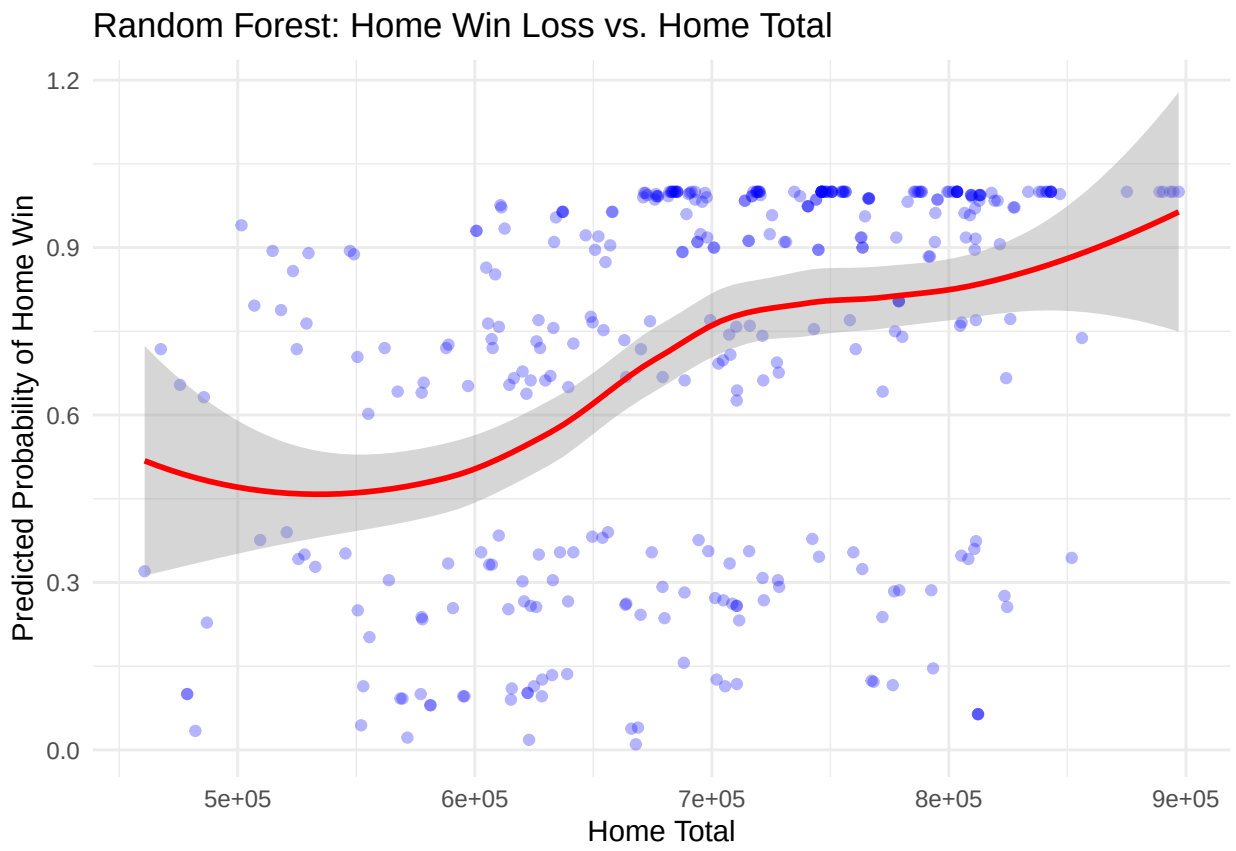



```
p9
```



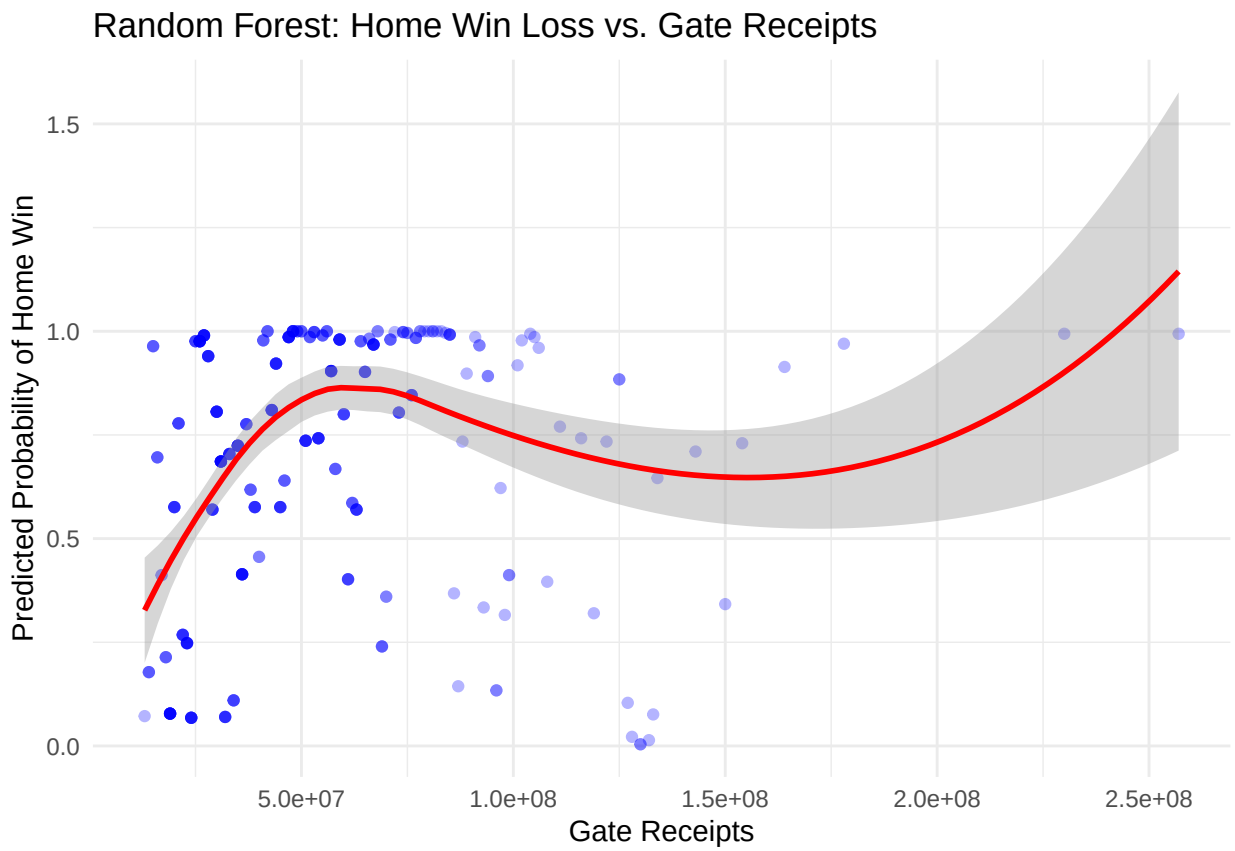

```
hc_combined_dataset$rf_pred1 <- predict(rf1, hc_combined_dataset, type = "prob")[,2]
hc_combined_dataset$rf_pred2 <- predict(rf2, hc_combined_dataset, type = "prob")[,2]

p11 <- ggplot(hc_combined_dataset, aes(x = Home_Total, y = rf_pred1)) +
  geom_point(alpha = 0.3, color = "blue") +
  geom_smooth(method = "loess", col = "red") +
  labs(title = "Random Forest: Home Win Loss vs. Home Total",
       x = "Home Total",
       y = "Predicted Probability of Home Win") +
  theme_minimal()

p12 <- ggplot(hc_combined_dataset, aes(x = Gate_Receipts, y = rf_pred2)) +
  geom_point(alpha = 0.3, color = "blue") +
  geom_smooth(method = "loess", col = "red") +
  labs(title = "Random Forest: Home Win Loss vs. Gate Receipts",
       x = "Gate Receipts",
       y = "Predicted Probability of Home Win") +
  theme_minimal()

p11
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

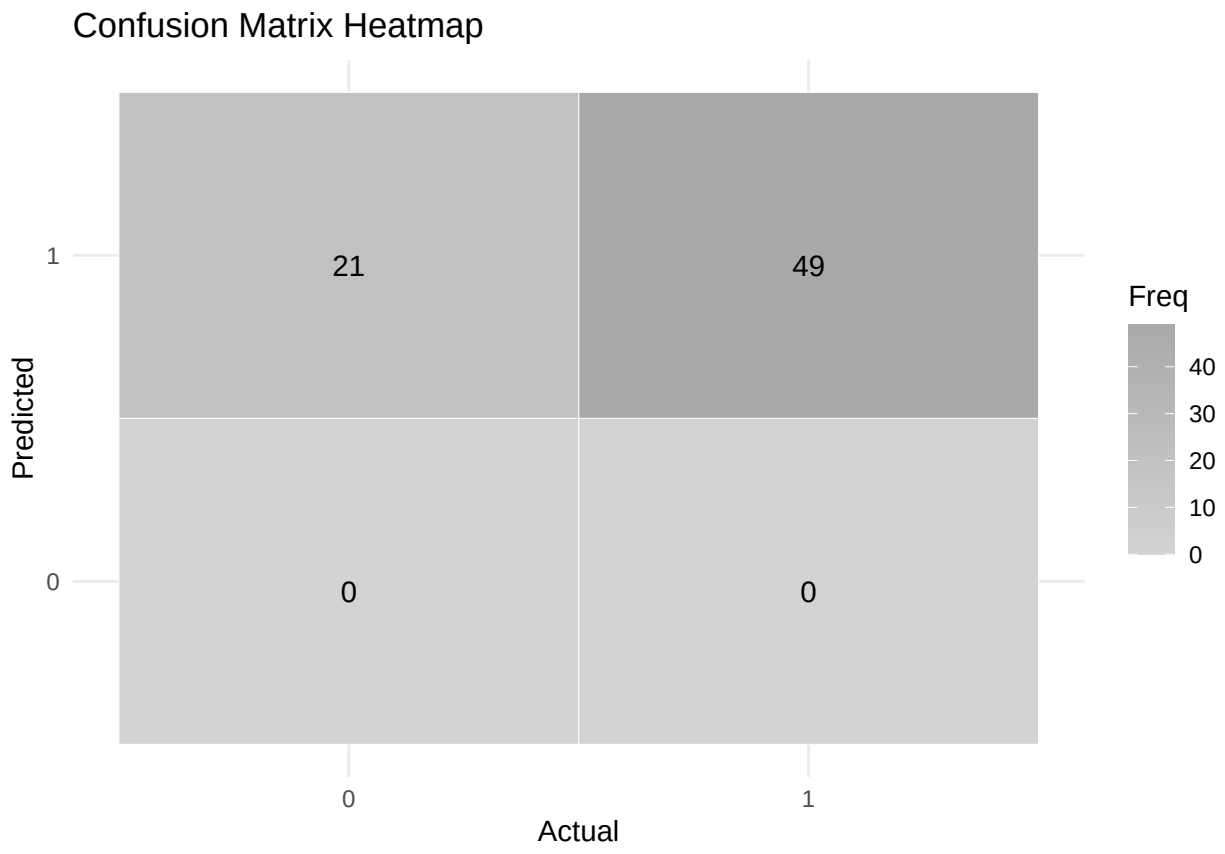



```
p12
```

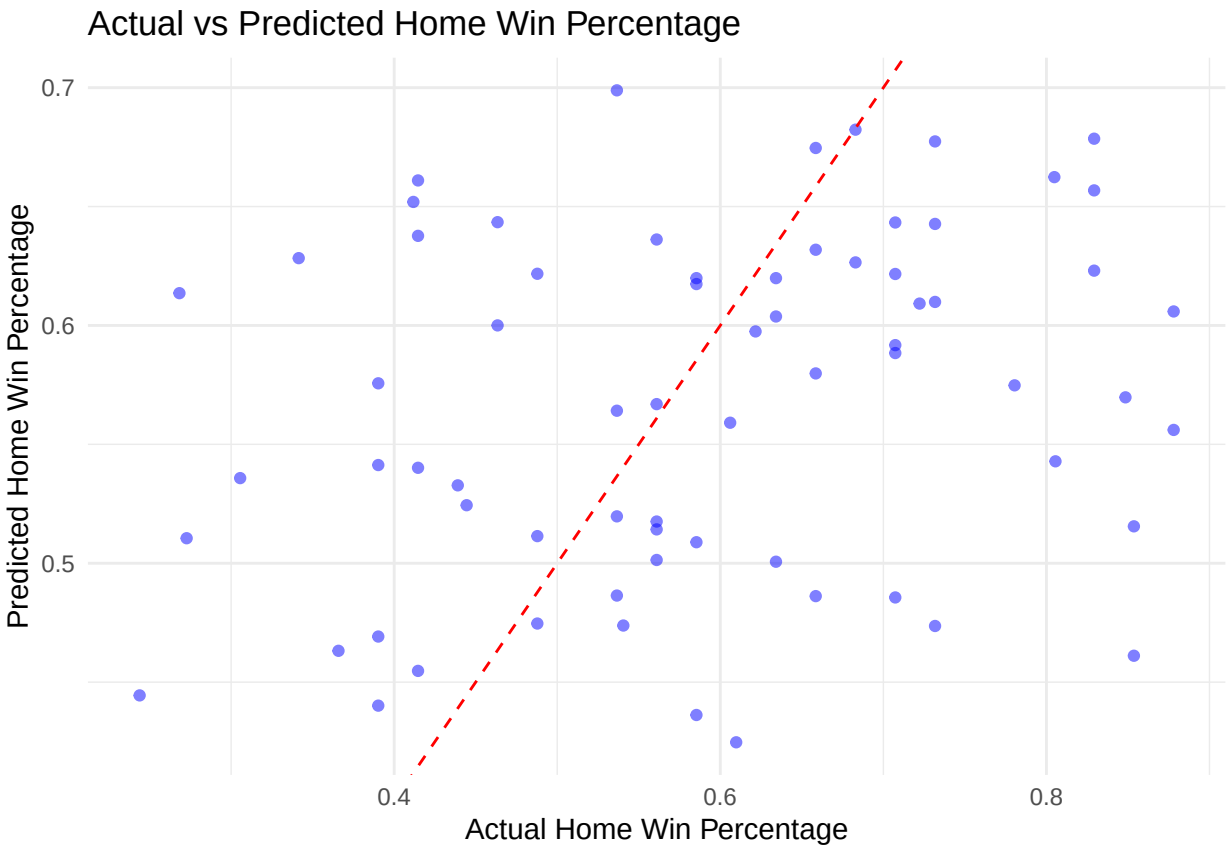
```
## `geom_smooth()` using formula = 'y ~ x'
```




```
conf_matrix_df <- as.data.frame(conf_matrix$table)
names(conf_matrix_df) <- c("Prediction", "Reference", "Freq")

p13 <- ggplot(data = conf_matrix_df, aes(x = Reference, y = Prediction)) +
  geom_tile(aes(fill = Freq), color = "white") +
  scale_fill_gradient(low = "lightgrey", high = "darkgrey") +
  geom_text(aes(label = Freq), vjust = 1) +
  labs(title = "Confusion Matrix Heatmap",
       x = "Actual",
       y = "Predicted") +
  theme_minimal()

p13
```




```
results <- data.frame(Actual = actual, Predicted = pred)

p14 <- ggplot(results, aes(x = Actual, y = Predicted)) +
  geom_point(color = "blue", alpha = 0.5) +
  geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +
  labs(title = "Actual vs Predicted Home Win Percentage",
       x = "Actual Home Win Percentage",
       y = "Predicted Home Win Percentage") +
  theme_minimal()


p14
```


Actual vs Predicted Home Win Percentage
Predicted Home Win Percentage
0.7
0.6
0.5
0.4
0.6
0.8
Actual Home Win Percentage