My Portfolio

Andre Contreras

2023-2024

Machine Learning Models

As I was watching the 2024 Monaco Grand Prix- I wondered how hard it would be to build a Machine Learning model that could (somewhat) successfully predict the winner of the races to come, so I got to work and after a couple of days of trial and error I managed to build a model. I started off with just a Random Forest Classification model, but it wasn't giving me the results I wanted. Therefore, I figured I should create 3 different classification models (Support Vector Machine, Random Forest, and Gradient Boosting) to then combine them all in this Ensemble Methods model and see if that got me more precise results... It did, with incredibly high evaluation scores (shown in the Metric Evaluation Plot).

P.S I was also able to build a model to predict X driver's lap time on lap Y in the circuit Z, but that html output alone is bigger than this one so I might just keep it in a separate html.

1. Ensemble Methods Model to predict RACE WINNER

```
races <- read.csv("/Users/ajcon/Downloads/Portfolio/F1 Race Data/races.csv", na.strings = "\\N")
lap_times <- read.csv("/Users/ajcon/Downloads/Portfolio/F1 Race Data/lap_times.csv", na.strings = "\\N"
results <- read.csv("/Users/ajcon/Downloads/Portfolio/F1 Race Data/results.csv", na.strings = "\\N")
drivers <- read.csv("/Users/ajcon/Downloads/Portfolio/F1 Race Data/drivers.csv", na.strings = "\\N")
circuits <- read.csv("/Users/ajcon/Downloads/Portfolio/F1 Race Data/circuits.csv", na.strings = "\\N")
driver_standings <- read.csv("/Users/ajcon/Downloads/Portfolio/F1 Race Data/driver_standings.csv", na.s
qualifying <- read.csv("/Users/ajcon/Downloads/Portfolio/F1 Race Data/qualifying.csv", na.strings = "\\
status <- read.csv("/Users/ajcon/Downloads/Portfolio/F1 Race Data/status.csv", na.strings = "\\N")


# Step 1: Extract Features
f1data <- left_join(races, circuits, by = "circuitId")
f1data <- left_join(f1data, results, by = "raceId")
f1data <- left_join(f1data, drivers, by = "driverId")
f1data <- left_join(f1data, status, by = "statusId")

f1data <- na.omit(f1data) %>%
  filter(year >= 2021) %>%
  select(circuitId, constructorId, year, driverId, grid, position, milliseconds) %>%
  mutate(isWinner = ifelse(position == 1, TRUE, FALSE)) %>% # Feature Engineering
  mutate(grid = ifelse(grid == 0, 21, grid)) %>%
  select(-position)

set.seed(123)
```

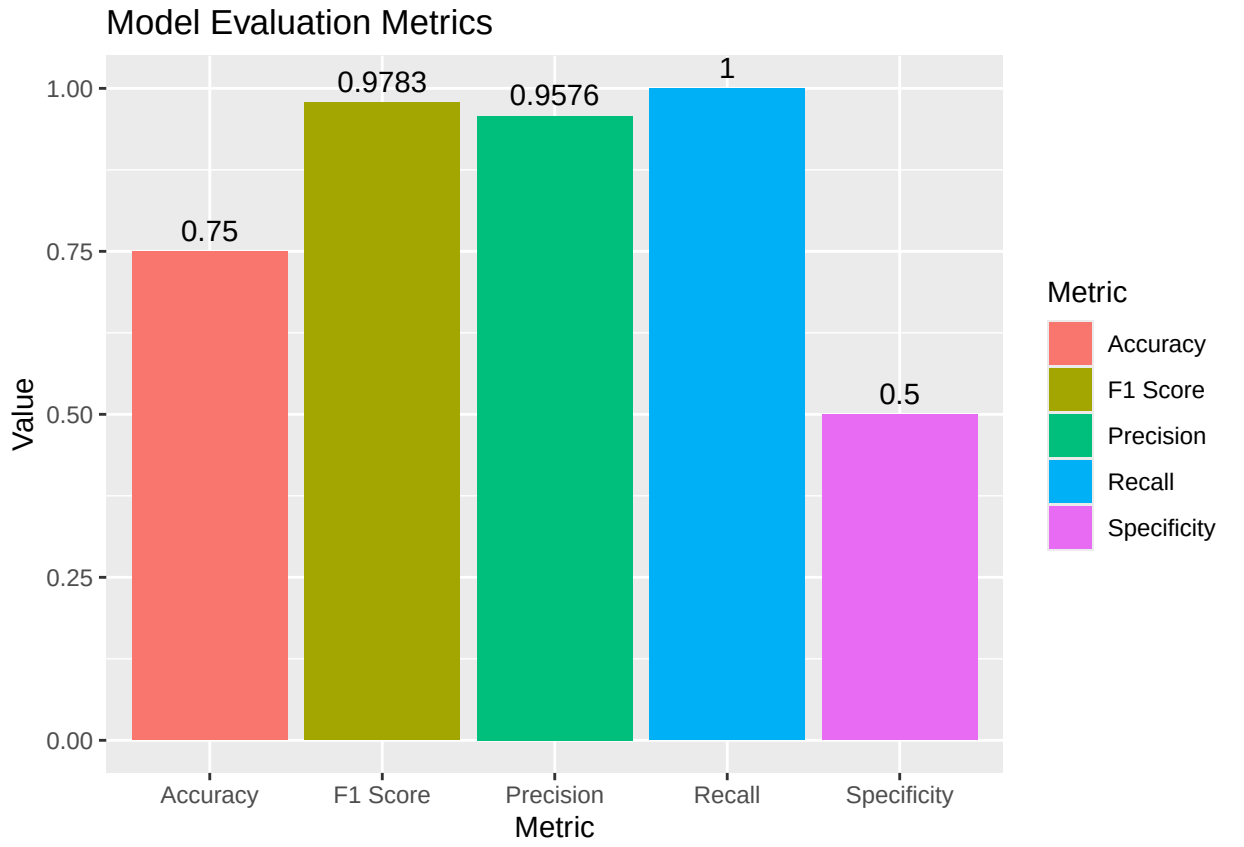
```r
f1data$isWinner <- as.factor(f1data$isWinner) # only 2 levels -- true and false so classification

# STEP 2: SPLIT DATA INTO TRAIN AND TEST SETS
split <- createDataPartition(f1data$isWinner, p = 0.8, list = FALSE)
train_data <- f1data[split, ]
test_data <- f1data[-split, ]



# --------------------------- SUPPORT VECTOR MACHINE ----------------------------
# The SVM is a supervised learning machine that classifies data by finding an optimal line that maximiz

# STEP 3A: TRAIN SVM MODEL (Classification)
svm_model <- svm(isWinner ~  constructorId + driverId + grid + year + circuitId, data = train_data, kern

svm_predictions <- predict(svm_model, newdata = test_data)

svm_binary_predictions <- ifelse(svm_predictions == TRUE, 1, 0)

# --------------------------- RANDOM FOREST CLASSIFIER ---------------------------
# The Random Forest model grows multiple decision trees which are merged together for a more accurate p

# STEP 3B: TRAIN RF MODEL (Classification)

rf_model <- randomForest(isWinner ~ constructorId + driverId + grid + year + circuitId, data = train_dat

rf_predictions <- predict(rf_model, newdata = test_data, type = "response")

rf_binary_predictions <- as.numeric(rf_predictions) - 1


# ------------------------- GRADIENT BOOSITNG CLASSIFIER -------------------------
# Boosting is one kind of ensemble Learning method which trains the model sequentially and each new mod

# STEP 3C: TRAIN GBM MODEL (Classification)

train_data$isWinner_binary <- as.numeric(train_data$isWinner) - 1

train_data <- train_data %>%
  select(circuitId, constructorId, year, driverId, grid, milliseconds, isWinner_binary)

gb_model <- gbm(isWinner_binary ~ constructorId + driverId + grid + year + circuitId, data = train_data

gb_predictions <- predict(gb_model, newdata = test_data, type = "response")

gb_binary_predictions <- ifelse(gb_predictions > 0.5, 1, 0)

# ------------------------- FINAL ENSEMBLE METHOD MODEL ------------------------

final_predictions <- ifelse(svm_binary_predictions + rf_binary_predictions + gb_binary_predictions >= 2


# STEP 4: EVALUATE MODEL --- For classification tasks, you can evaluate the model using metrics such as
```

```r
# Calculate confusion matrix
test_data$isWinner_binary <- as.numeric(test_data$isWinner) - 1

confusion_matrix <- confusionMatrix(data = as.factor(final_predictions), reference = as.factor(test_data
confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 158   7
##          1   0   7
##
##                Accuracy : 0.9593
##                  95% CI : (0.9179, 0.9835)
##     No Information Rate : 0.9186
##     P-Value [Acc > NIR] : 0.02675
##
##                   Kappa : 0.6475
##
##  Mcnemar's Test P-Value : 0.02334
##
##             Sensitivity : 1.0000
##             Specificity : 0.5000
##          Pos Pred Value : 0.9576
##          Neg Pred Value : 1.0000
##              Prevalence : 0.9186
##          Detection Rate : 0.9186
##    Detection Prevalence : 0.9593
##       Balanced Accuracy : 0.7500
##
##        'Positive' Class : 0
##
```

```r
# True Positive: Model correctly predicts a driver didn't win race (156)
# False Positive: Model incorrectly predicts driver didn't win, when they did (7)
# True Negative: Model correctly predicts a driver won the race (7)
# False Negative: Model incorrectly predicts a driver won race when they didn't (2)
balanced_accuracy <- confusion_matrix$byClass["Balanced Accuracy"]

# PRECISION = 158 / (158+7) = 0.958
precision <- confusion_matrix$byClass["Pos Pred Value"]

# SENSITIVITY = 158 / (158 + 0) = 1
recall <- confusion_matrix$byClass["Sensitivity"]

# SPECIFICITY = (7 / [7 + 7]) = 0.5
specificity <- confusion_matrix$byClass["Specificity"]

# F1 SCORE= 2 * (0.958 * 1) / (0.958 + 1) = 0.978
f1score <- confusion_matrix$byClass["F1"]

summary <- data.frame(
```

```
  "SVM Prediction" = svm_binary_predictions,
  "RF Prediction" = rf_binary_predictions,
  "GB Prediction" = gb_binary_predictions,
  "Final Prediction" = final_predictions,
  "Actual Winner" = test_data$isWinner_binary
)

# STEP 5: VISUALIZE RESULTS
metrics_df <- data.frame(
  Metric = c("Accuracy", "Precision", "Recall", "Specificity", "F1 Score"),
  Value = c(balanced_accuracy, precision, recall, specificity, f1score)
)

ggplot(metrics_df, aes(x = Metric, y = Value, fill = Metric)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = round(Value, 4)), vjust = -0.5) +
  labs(title = "Model Evaluation Metrics", y = "Value")
```




```
# STEP 6: USER INTERFACE
user_input <- data.frame(
  circuitId = rep(params$circuitId_isWinner, 20), # Replicate circuitId for all rows
  constructorId = c(9, 9, 6, 6, 1, 1, 131, 131, 117, 117, 215, 215, 214, 214, 210, 210, 15, 15, 3, 3),
  year = rep(params$year_isWinner, 20), # Replicate year for all rows
  driverId = c(830, 815, 844, 832, 846, 857, 1, 847, 4, 840, 852, 817, 842, 839, 825, 807, 822, 855, 848
  grid = rep(0, 20), # Placeholder for grid
```

```
  milliseconds = rep(0, 20) # Placeholder for milliseconds
)

ui_svm_predictions <- predict(svm_model, newdata = user_input)
ui_svm_binary_predictions <- ifelse(ui_svm_predictions == TRUE, 1, 0)

ui_rf_predictions <- predict(rf_model, newdata = user_input, type = "response")
ui_rf_binary_predictions <- ifelse(ui_rf_predictions == TRUE, 1, 0)

ui_gb_predictions <- predict(gb_model, newdata = user_input, type = "response")
ui_gb_binary_predictions <- ifelse(ui_gb_predictions > 0.5, 1, 0)

ui_final_predictions <- ifelse(ui_svm_binary_predictions + ui_rf_binary_predictions + ui_gb_binary_predi

user_input <- user_input %>%
  mutate(ui_final_predictions) %>%
  left_join(drivers, by = "driverId")

winner_index <- which(user_input$ui_final_predictions == 1)
name <- user_input$forename[winner_index]
lastname <- user_input$surname[winner_index]

circuit <- unique(races$name[races$circuitId == params$circuitId_isWinner])

cat("Predicted Winner of the", params$year_isWinner, circuit, ":", name, lastname)
```

```
## Predicted Winner of the 2024 Canadian Grand Prix : Max Verstappen
```

Accuracy: The overall accuracy of 0.9593 indicates the proportion of correct predictions made by the model on the entire test dataset.

Precision: The precision quantifies the model's ability to avoid incorrectly predicting a driver did not win the race, when they actually did (false positives). ***Therefore, a high precision score of 0.957 indicates that when the model predicts a driver as not winning the race, it is correct about 95.7% of the time.***

Sensitivity (Recall): The sensitivitiy quantifies the model's ability to successfully capture all cases where a driver didn't win (positive cases). The sensitivity score of 1 suggests that the model is able to successfully capture a high proportion of the cases where a driver actually did not win the race. ***This means that when a driver did not win the race, the model correctly identifies them as such 100% of the time.***

Specificity: The specificity quantifies the models ability to correctly identify cases where a driver won (negative cases). ***A specificity score of 0.5 means that approximately 50% of the time when a driver actually won the race, the model incorrectly predicts that they didn't win the race***

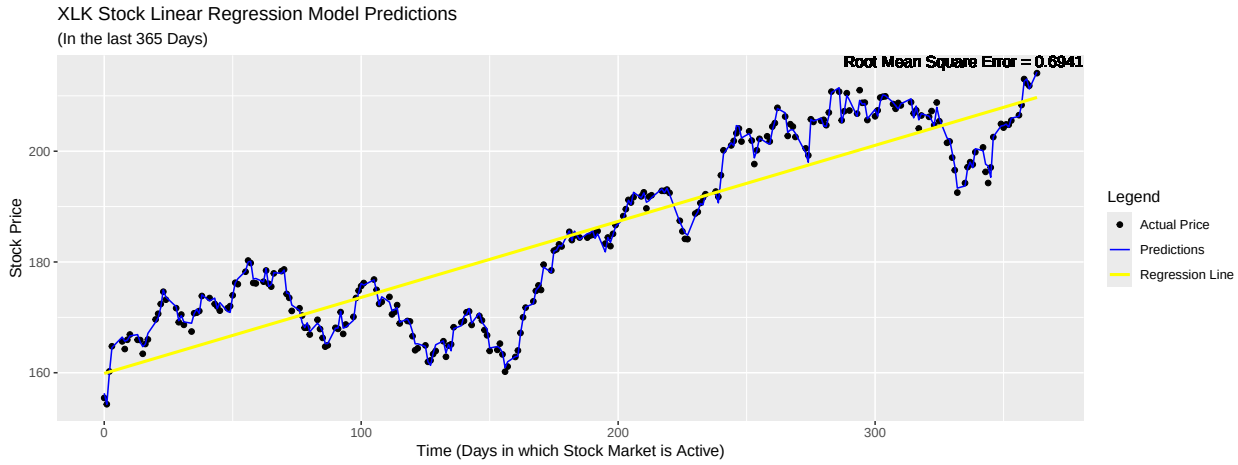
F1 Score: The F1 score of 0.978 is a harmonic mean of precision and recall, providing a balanced measure of the model's performance. It combines both the precision and sensitivity of the model into a single metric.

Balanced Accuracy: The balanced accuracy of 0.75 accounts for class imbalance by taking the average of sensitivity and specificity. It provides a more reliable measure of model performance when dealing with imbalanced datasets.

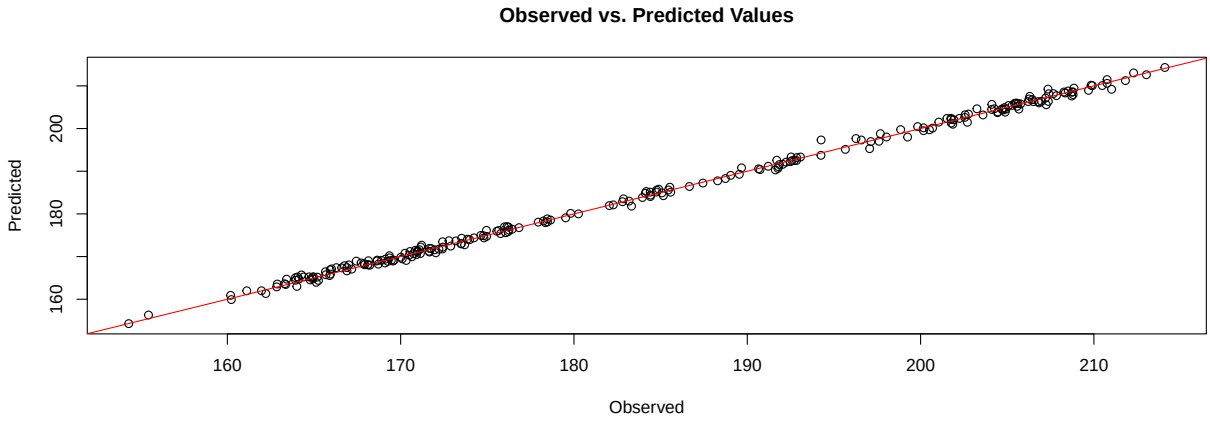
2. Linear Regression Model on XLK Stock

In the code below, I successfully train a Machine Learning model by gathering and manipulating the relevant data, splitting it into two sets (train & test), training a linear regression model, and testing it on the data set aside for testing. The resulting model displays a Root Mean Square Error of only 0.69, meaning that on average, my predictions are off by $0.69. I also included a regression line to show that on average, the stock price has gone up in the past year as well as a perfect fit line in the other plot.

```
# 1. Read the CSV files into data frames
data <- read_csv("XLK dataset.csv")

data$Date <- as.Date(data$Date, format("%m/%d/%Y"))
data$Date <- as.numeric(data$Date)

# 2. Split the dataset into training and testing subsets
training_data <- data[1:5884, ]
testing_data <- data[5885:6134, ]

# Train Model using linear regression model ('.' indicates all other variables will be used as predicto
model <- lm(Close ~ ., data = training_data)

predictions <- predict(model, newdata = testing_data)

# Evaluate performance using Root Mean Square Error
rmse <- sqrt(mean((testing_data$Close - predictions)^2))

# Plot
ggplot(testing_data, aes(Date-19500, Close)) +
  geom_point(aes(col = "Actual Price")) +  # Testing Data points (Actual price values)
  geom_line(aes(y = predictions, col = "Predictions")) +  # Add a line for model predictions
  geom_smooth(method = "lm", aes(col = "Regression Line"), se = FALSE) + # Regression Line
  labs(x = "Time (Days in which Stock Market is Active)", y = "Stock Price", title = "XLK Stock Linear
  geom_text(aes(label = paste("Root Mean Square Error =", round(rmse, 4)), x = Inf, y = Inf), hjust = 1
  scale_color_manual(name = "Legend", values = c("black", "blue", "yellow"))
```



```r
plot(testing_data$Close, predictions,
     xlab = "Observed", ylab = "Predicted",
     main = "Observed vs. Predicted Values",
     col = "black")
# Red "Perfect" Line
abline(0, 1, col = "red")
```



---

3. Linear Regression Model on House Market

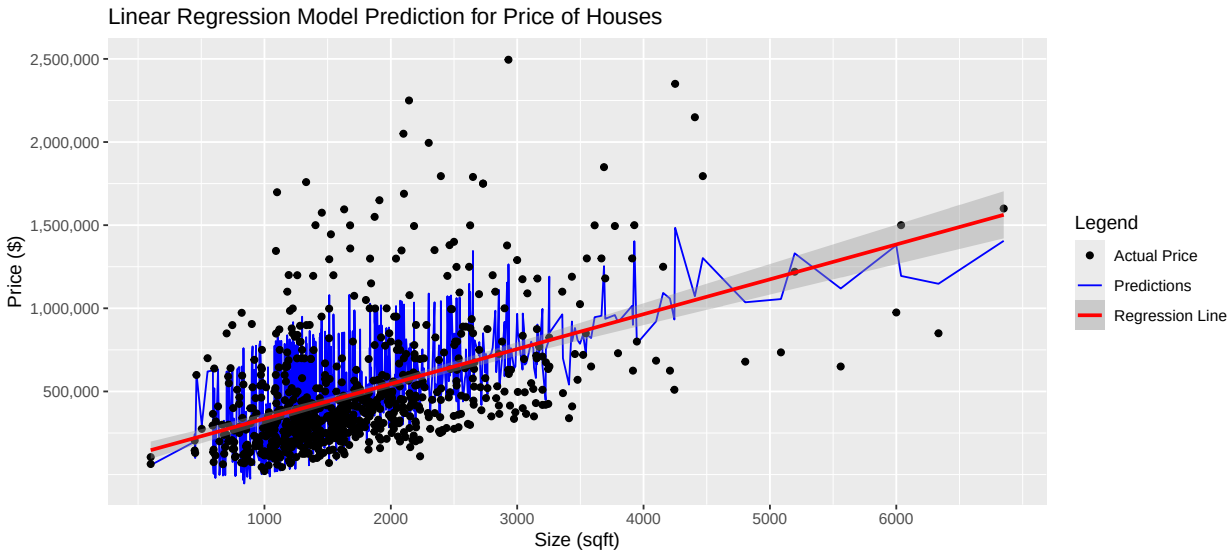
Recently, I found at our neighbors were selling their house and moving away. I was curious how much the house would cost so I went on Zillow and to my surprise, it was worth nearly $200,000 more than ours! After looking at the pictures available, it was easy to understand why: they had a finished basement, a breakfast sunroom, and their house was nearly 1000 sqft bigger than hours.

This got me wondering how important certain features have in a home, especially the number of beds, baths, and its size in sqft. I got to work building a linear regression model that is able to predict the price of a house given the features as input. All in all, it turned out pretty good.

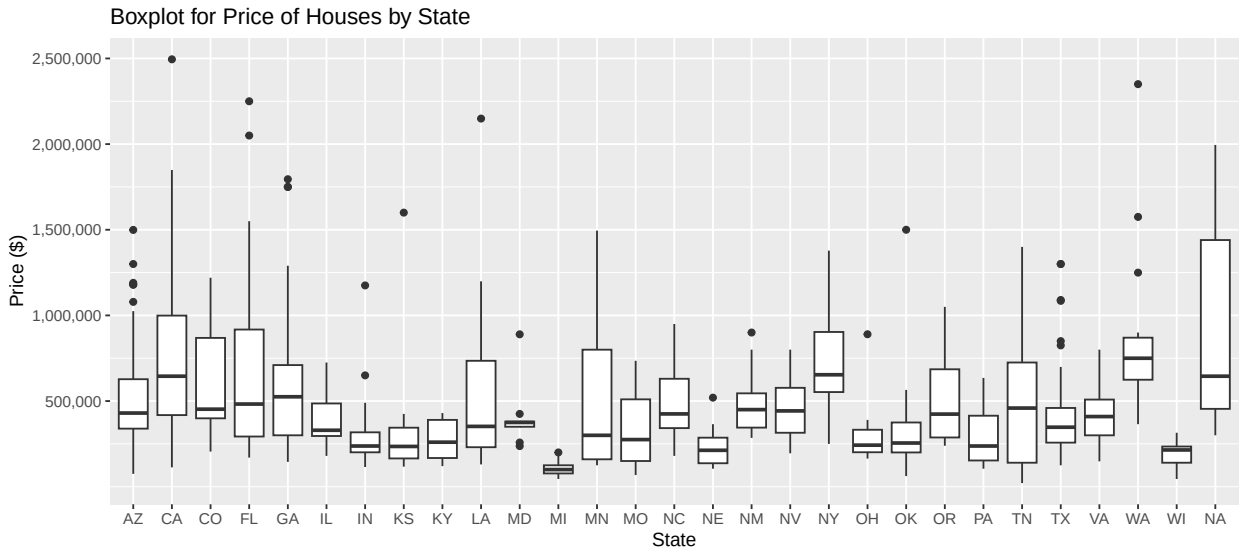
```r
set.seed(1)

# Step 1: Extract Features
data <- read_csv("USA_Housing_Data.csv")

# Changing the name of the variables I will use that have spaces because R doesn't like spaces so it wi
names(data)[names(data) == "Living Space"] <- "Living_Space"
names(data)[names(data) == "Median Household Income"] <- "Median_Household_Income"

# Assigning a unique numeric identifier for each state in the data to put in the lm() regression model
data$State <- factor(data$State)
data$State_numeric <- as.integer(data$State)

  # 1A: Removing Missing Values
  clean_data <- na.omit(data)
```

```r
  # 1B: Calculate Z-scores for each variable to detect and remove outliers
  z_scores <- scale(clean_data$Price)
  outliers <- which(abs(z_scores) > 2)

  z_scores_beds <- scale(clean_data$Beds)
  outliers_beds <- which(abs(z_scores_beds) > 2)

  z_scores_baths <- scale(clean_data$Baths)
  outliers_baths <- which(abs(z_scores_baths) > 2)

  z_scores_space <- scale(clean_data$Living_Space)
  outliers_space <- which(abs(z_scores_space) > 2)

  # 1C: Filtering the data so it's ready for modelling
  clean_data <- clean_data %>%
    filter(!Price %in% clean_data$Price[outliers]) %>%
    filter(!Beds %in% clean_data$Beds[outliers_beds]) %>%
#    filter(!Baths %in% clean_data$Baths[outliers_baths]) %>%
#    filter(!Living_Space %in% clean_data$Living_Space[outliers_space]) %>%
    select(Price, Beds, Baths, Living_Space, State, State_numeric)


# ---------- ONLY READ AFTER COMPLETING REGRESSION IN R DATACAMP COURSE ----------

#clean_data$State <- factor(clean_data$State)
# Since, State is a categorical variable, and lm() only accepts numeric variables be used in determinin

#dummy_vars <- model.matrix(~ State - 1, data = clean_data)
#clean_data <- cbind(clean_data, dummy_vars)
# Since categorical variable has more than 2 levels, we need to encode it into dummy(placeholder) varia
# ------------------------------------------------------------------------------

# Step 2: Separate into 2 datasets (Training & Testing)
split <- createDataPartition(clean_data$Price, p = 0.98, list = FALSE)

training_data <- clean_data[split, ]
testing_data <- clean_data[-split, ]


# Step 3: Train Model
model <- lm(Price ~., data = training_data)

predictions <- predict(model, newdata = testing_data)


# Step 4: Evaluate Model
rmse <- sqrt(mean((testing_data$Price - predictions)^2))

# Step 5: Display results, Price correlation with each important house feature.
ggplot(testing_data, aes(x = Living_Space, y = Price)) +
  geom_line(aes(y = predictions, color = "Predictions")) +
  geom_point(aes(color = "Actual Price")) +
  geom_smooth(method = "lm", aes(color = "Regression Line")) +
```

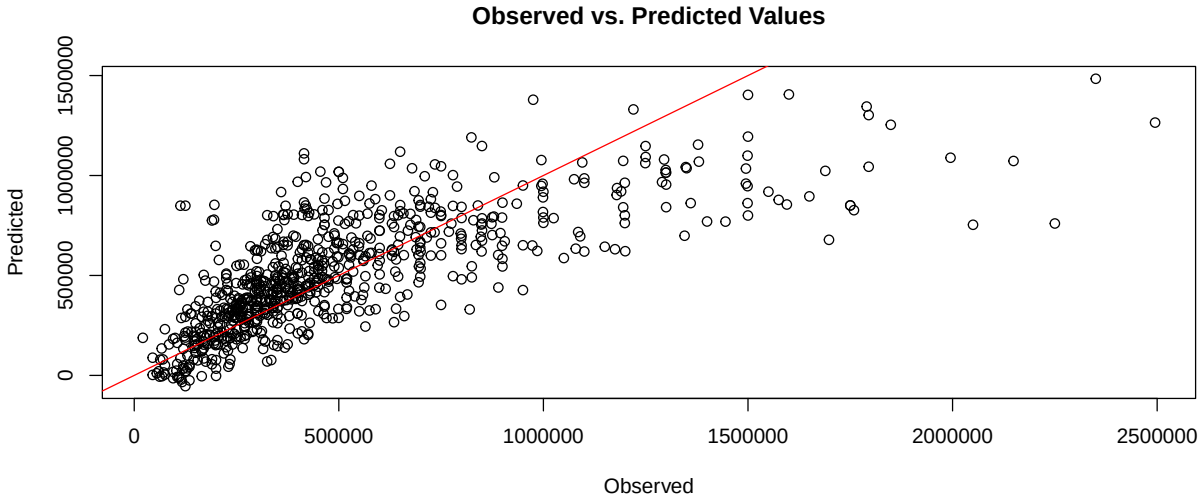
```
  labs(x = "Size (sqft)", y = "Price ($)", title = "Linear Regression Model Prediction for Price of Hous
  scale_y_continuous(labels = scales::comma, breaks = seq(500000, 3000000, by = 500000)) +
  scale_x_continuous(breaks = seq(1000, 6000, by = 1000)) +
  scale_color_manual(name = "Legend", values = c("black", "blue", "red"))
```




```
# ggplot(testing_data, aes(x = Beds, y = Price)) +
#   geom_point() +
#   geom_smooth(method = "lm") +
#   labs(x = "Number of Beds", y = "Price ($)", title = "Linear Regression Model Prediction for Price o
#   scale_y_continuous(labels = scales::comma, breaks = seq(500000, 3000000, by = 500000)) +
#   scale_x_continuous(breaks = seq(1, 7, by = 1))
#
# ggplot(testing_data, aes(x = Baths, y = Price)) +
#   geom_point() +
#   geom_smooth(method = "lm") +
#   labs(x = "Number of Bathrooms", y = "Price ($)", title = "Linear Regression Model Prediction for Pr
#   scale_y_continuous(labels = scales::comma, breaks = seq(500000, 3000000, by = 500000)) +
#   scale_x_continuous(breaks = seq(1, 6, by = 1))

# Create a mapping from state names to abbreviations
state_abbreviations <- setNames(state.abb, tolower(state.name))

# Convert state names in testing_data to abbreviations
testing_data$State_Abbreviation <- state_abbreviations[tolower(testing_data$State)]

ggplot(testing_data, aes(x = State_Abbreviation, y = Price)) +
  geom_boxplot() +
  labs(x = "State", y = "Price ($)", title = "Boxplot for Price of Houses by State") +
  scale_y_continuous(labels = scales::comma, breaks = seq(500000, 3000000, by = 500000))
```




```
plot(testing_data$Price, predictions,
     xlab = "Observed", ylab = "Predicted",
     main = "Observed vs. Predicted Values",
     col = "black")
# Red "Perfect" Line
abline(0, 1, col = "red")
```




```
# Step 6: Create Interface
# Creating a data frame with user input in param header of r markdown document
user_input <- data.frame( # If needed, use as.numeric() on numeric variables.
  Beds = params$num_beds,
  Baths = params$num_baths,
  Living_Space = params$living_space,
  State = params$state
)
```

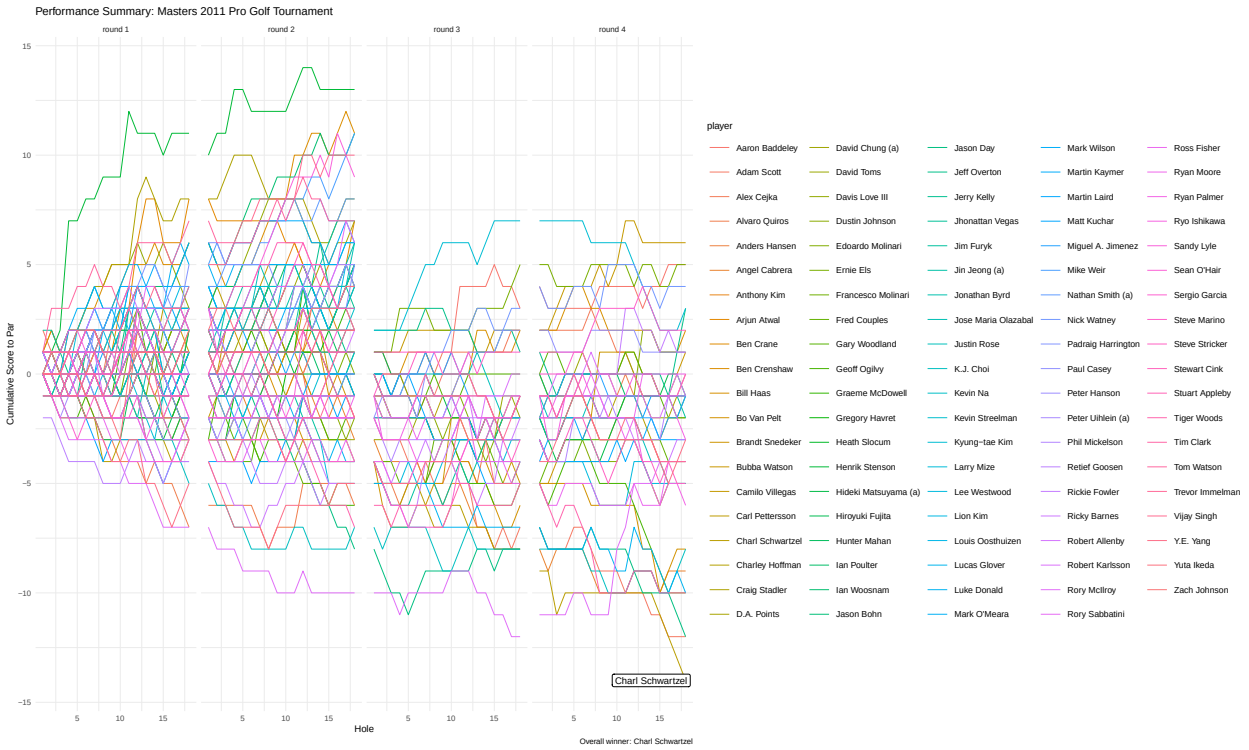
```r
# Check if state is in data
if (params$state %in% unique(clean_data$State)) {
  # Converting state input to its corresponding numeric identifier
  State_numeric <- clean_data$State_numeric[match(params$state, clean_data$State)]
  user_input$State_numeric <- State_numeric

  predicted_price <- predict(model, newdata = user_input)

  cat("Predicted price of a house in", params$state, "with", params$num_beds, "beds,", params$num_baths
} else {
  stop("Unfortunately, we do not have a record of homes in your state, thus we cannot provide a predicti
}
```

```
## Predicted price of a house in Texas with 3 beds, 3 baths, and a living space of 3000 sqft: $ 661948
```

2011 Masters Golf Tournament Project

The plot below is a line graph I created visualizing the summary of the Masters 2011 Pro Golf Tournament, along with the performance of each golfer and the overall winner of the comptetition (Charl Schwartzel).

```r
# Binds the rows of round1, round2,round3, and round4; specifies the name of the new column that will b
rounds <- bind_rows(round1, round2, round3, round4, .id = "round")


scorecard <- rounds %>%
  pivot_longer(cols = "1":"18", names_to = "hole", values_to = "score") %>%
  mutate(round = as.integer(round), hole = as.integer(hole), score = as.integer(score))


performance <- scorecard %>%
  left_join(course, by = "hole") %>%
  mutate(difference_to_par = score - par) %>%
  group_by(player) %>%
  mutate(cumulative_to_par = cumsum(difference_to_par)) %>%
  ungroup() %>%
  select(player, round, hole, difference_to_par, cumulative_to_par)

winner <- performance %>%
  filter(round == 4) %>%
  top_n(1, wt = -cumulative_to_par)

ggplot(performance) +
  geom_line(aes(x = hole, y = cumulative_to_par, col = player)) +
  facet_grid(. ~ round, labeller = labeller(round = c("1" = "round 1", "2" = "round 2", "3" = "round 3"
  geom_label(aes(x = hole - 4, y = cumulative_to_par, label = player), data = winner) +
  labs(title = "Performance Summary: Masters 2011 Pro Golf Tournament", x = "Hole", y = "Cumulative Sco
  theme_minimal() +
  theme(legend.position = "right", legend.key.size = unit(c(1, 1), "cm"), legend.text = element_text(si
```

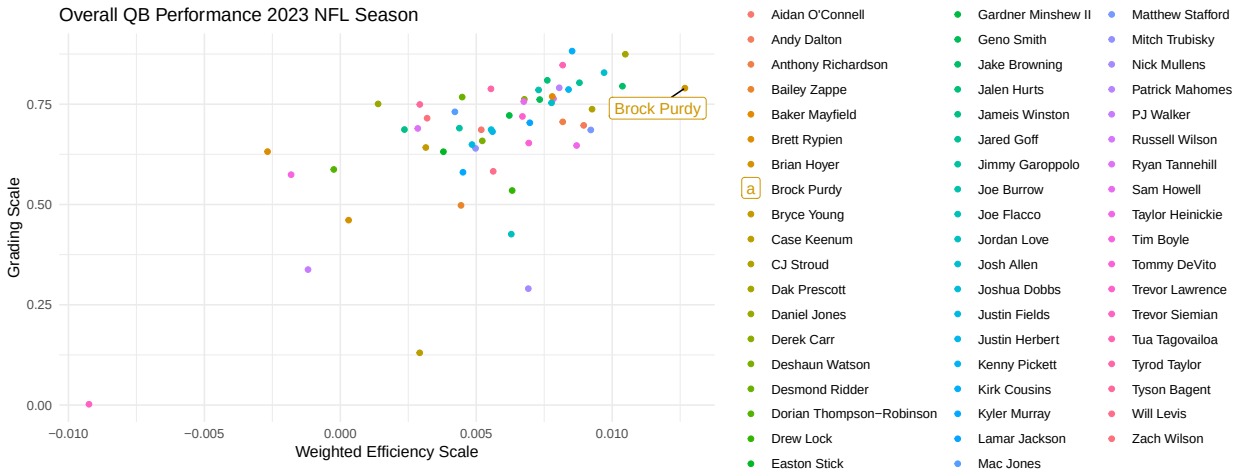


NFL 2023 QB Performance Project

Overall QB Performance:

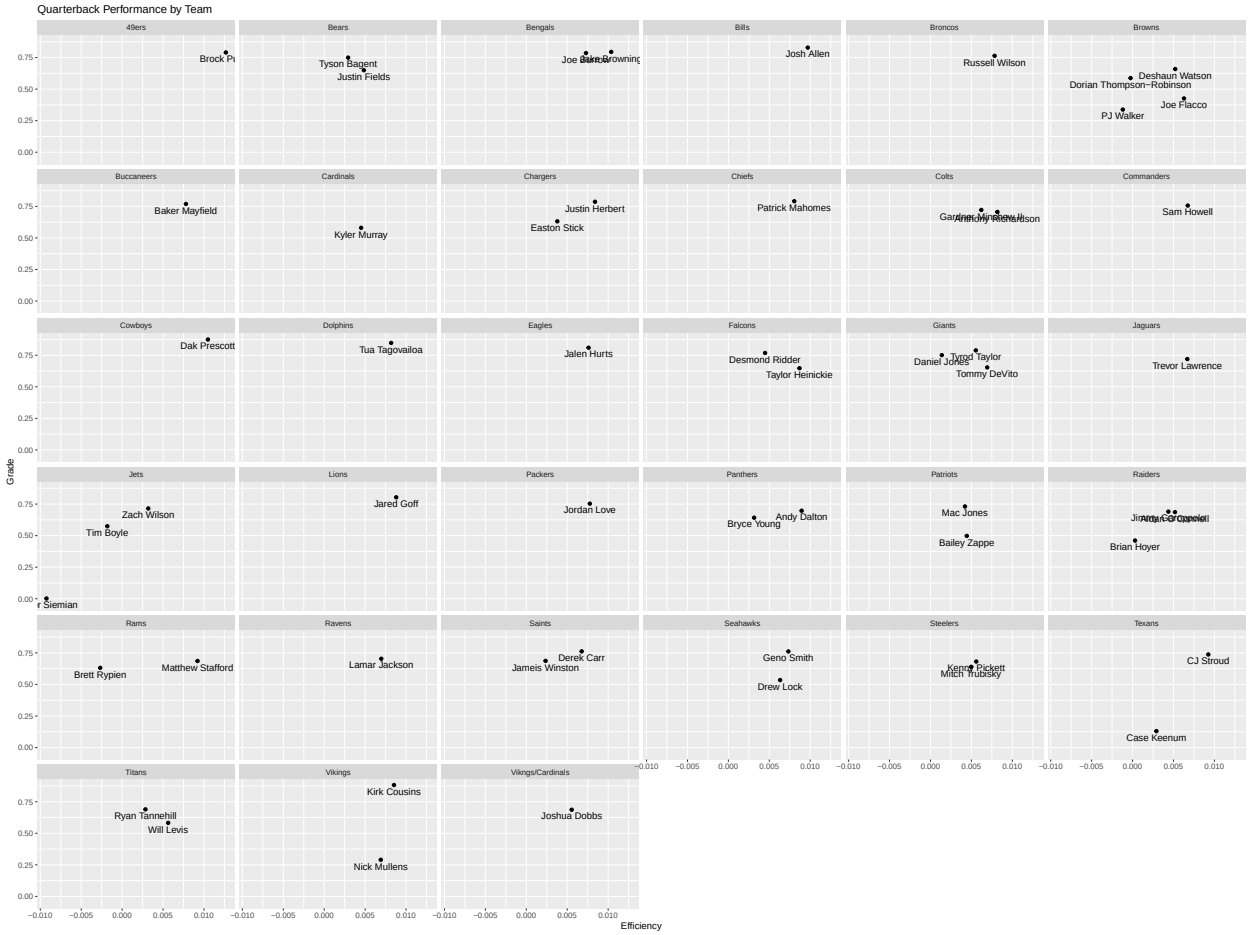
I created a spreadsheet of all the NFL Quarterbacks that played a minimum of one game in the 2023-2024 season and compiled all of their advanced statistics. I then developed a formula in Excel aiming to grade their efficiency, MVP rating, and overall grade, and transferred the file into R to visualize my findings. The name displayed is my MVP according to my grading system and efficiency formula

```
qbdata <- read_excel("C:\\Users\\ajcon\\Downloads\\Portfolio\\qbdata.xlsx")

mvp <- qbdata %>%
  top_n(1, qbdata$Grade*qbdata$Efficiency)

ggplot(qbdata, aes(x = Efficiency, y = Grade, col = QB)) +
  geom_point() +
  geom_label_repel(data = mvp, aes(label = QB), nudge_x = -0.001, nudge_y = -0.05, segment.color = "bla
  labs(x = "Weighted Efficiency Scale", y = "Grading Scale", title = "Overall QB Performance 2023 NFL S
  theme_minimal() +
  theme(legend.position = "right")
```



QB Performance by Team:

```
ggplot(qbdata, aes(x = Efficiency, y = Grade)) +
  geom_point() +
  geom_text(aes(label = QB), nudge_x = 0, nudge_y = -0.05) +
  facet_wrap(. ~ Team) +
  labs(x = "Efficiency", y = "Grade", title = "Quarterback Performance by Team")
```



A Data-led Look into the History of the Olympics

After gaining access to mulitple datasets of the Olympics containing every instance throughout every competition since the inaugural season back in 1896 (Greece) up until the 2016 Games in Brazil, I decided my free time would be well spent answering a couple of questions I, like many others (I think), have been wondering:

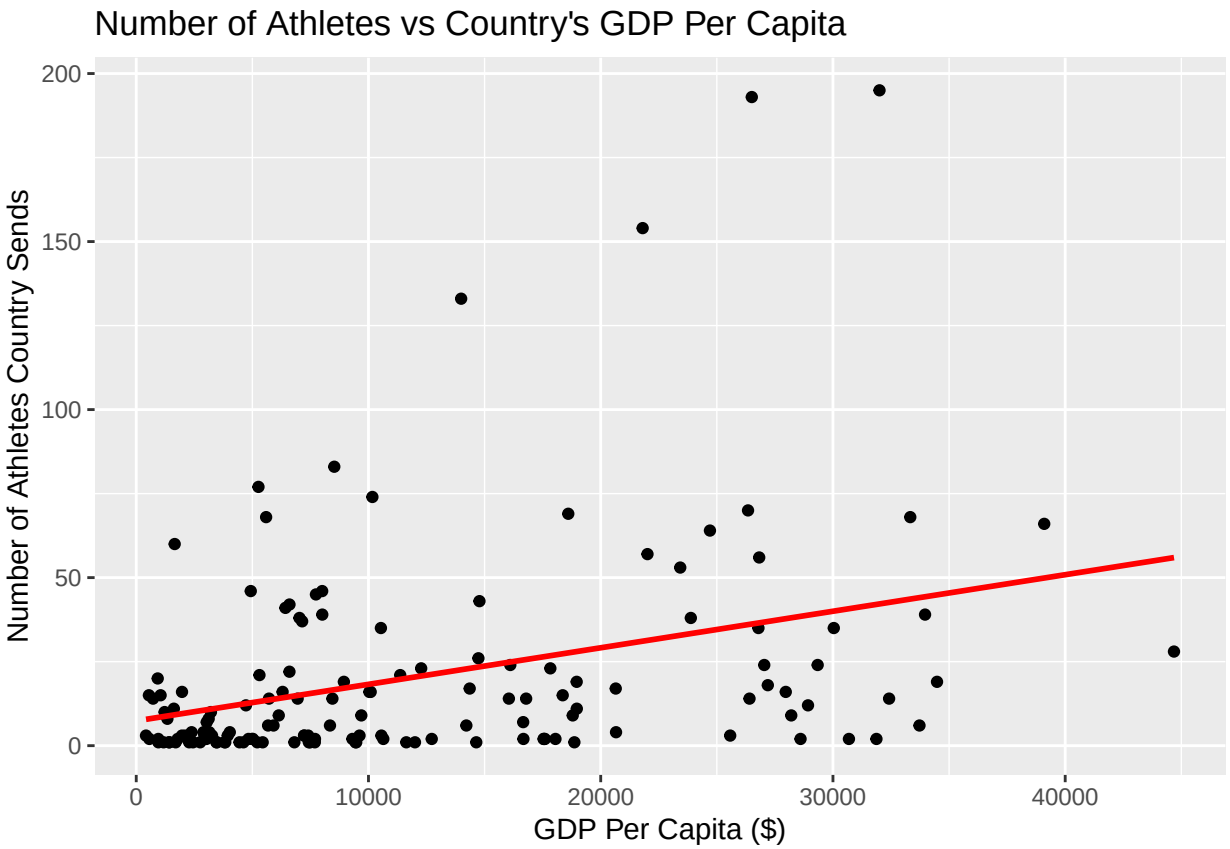
1. Does the economical stability of a country affect the number of athletes it sends to the olympics and the number of medals it wins?

2. Does hosting the olympics correlate to winning more medals that year?

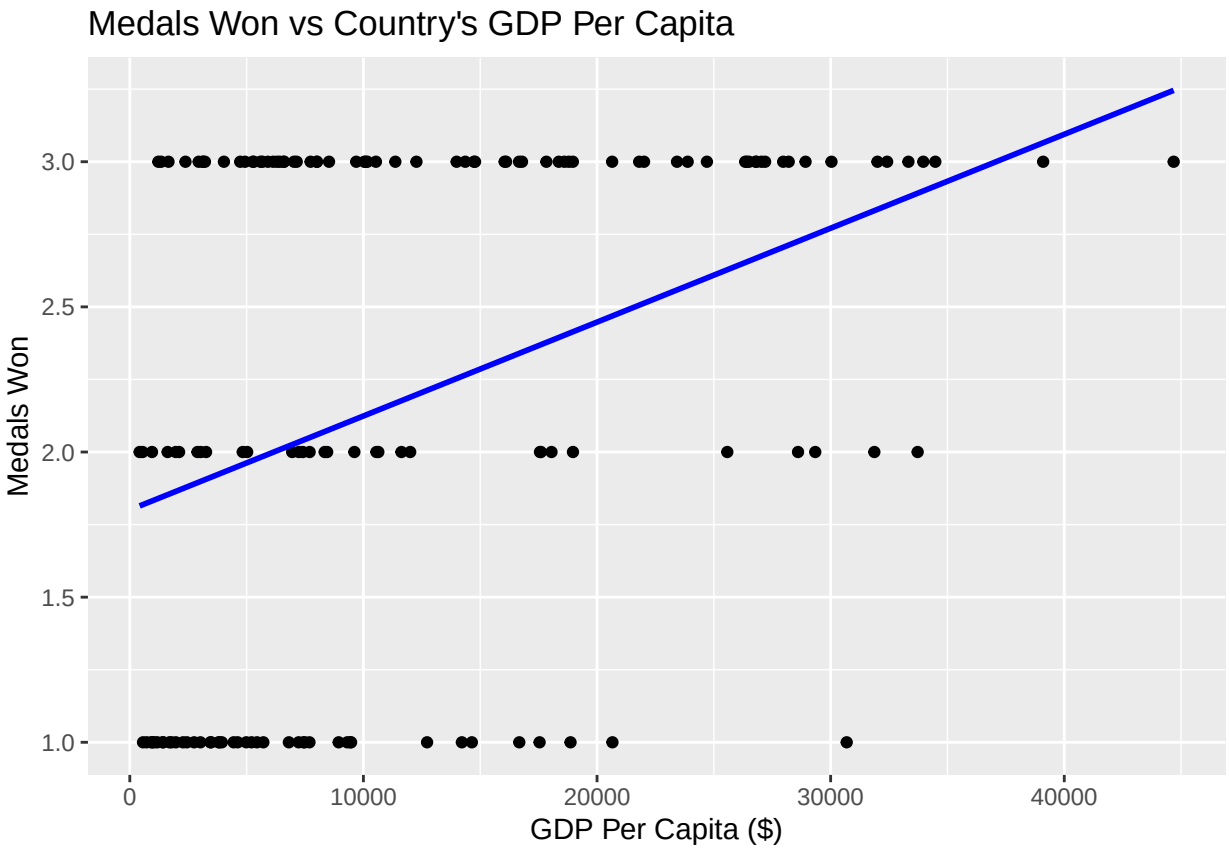
Part I

Below are the results I found for the first question, along with the code I wrote to filter and manipulate the data so I can visualize it in a more effective manner.

```r
athlete_events <- read_csv(
   file = 'athlete_events.csv',
   col_types = cols(ID = 'i', Age = 'i', Height = 'i', Year = 'i')
)

nearest_year <- function(olympics_year) {
  gapminder_year <- seq(1952, 2007, by = 5)
  nearest_year <- gapminder_year[which.min(abs(olympics_year - gapminder_year))]
  return(nearest_year)
}

olympics_data <-
  athlete_events %>%
  filter(!is.na(Medal)) %>%
  count(Games, Event, NOC, Medal, Team, Year, Name) %>%
  mutate(year = nearest_year(Year))

country_money <- gapminder %>%
  group_by(country) %>%
  select(country, year, gdpPercap)


athletes_by_country_year <- olympics_data %>%
  group_by(Team, Year) %>%
  summarise(Total_Athletes = n_distinct(Name), .groups = 'drop')

medals_by_country_year <- olympics_data %>%
  group_by(Team, Year) %>%
  summarise(Total_Medals = n_distinct(Medal), .groups = 'drop')

joined_data_athletes <- inner_join(athletes_by_country_year, country_money, by = c("Year" = "year", "Tea
  filter(!is.na(gdpPercap))

joined_data_medals <- inner_join(medals_by_country_year, country_money, by = c("Year" = "year", "Team" =
  filter(!is.na(gdpPercap))

ggplot(joined_data_athletes, aes(x = gdpPercap , y = Total_Athletes)) +
  geom_point() +
  geom_smooth(method = "lm", color = "red", se = FALSE) + #Plotting the athlete correlation
  labs(title = "Number of Athletes vs Country's GDP Per Capita", x = "GDP Per Capita ($)", y = "Number
```




```
ggplot(joined_data_medals, aes(x = gdpPercap , y = Total_Medals)) +
  geom_point() +
  geom_smooth(method = "lm", color = "blue", se = FALSE) + #lm creates a smooth line to show a clear re
    labs(title = "Medals Won vs Country's GDP Per Capita", x = "GDP Per Capita ($)", y = "Medals Won")
```



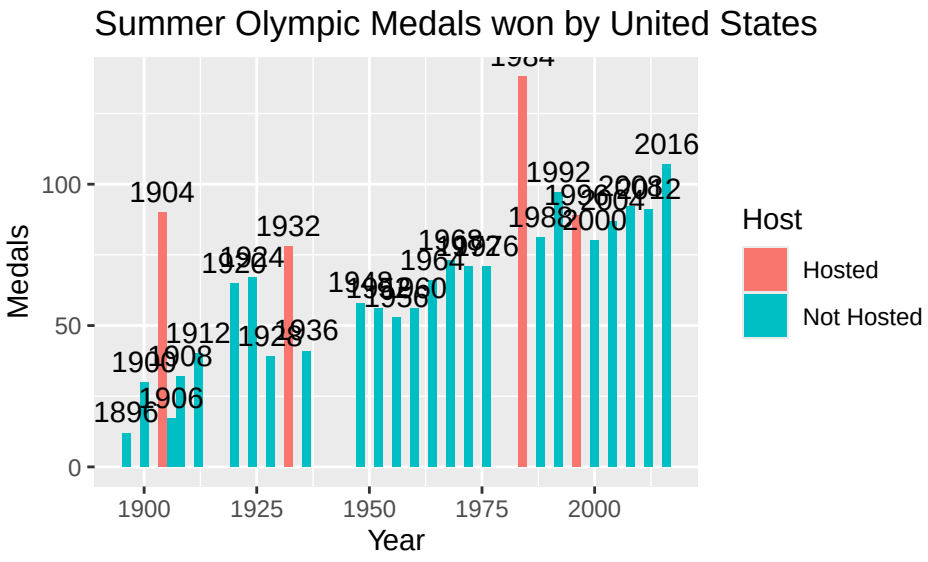
As we can see, there is in fact a positive correlation between a country's gdp per capita and the number of medals and athletes a country has. This means that the higher the gdp is, the more medals it wins and more athletes it sends to the Olympics.

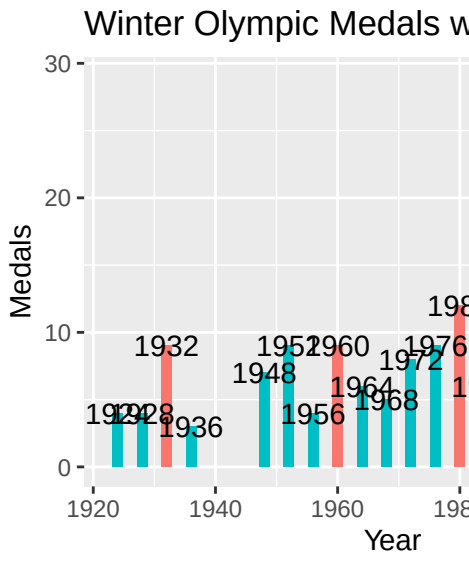
Part II

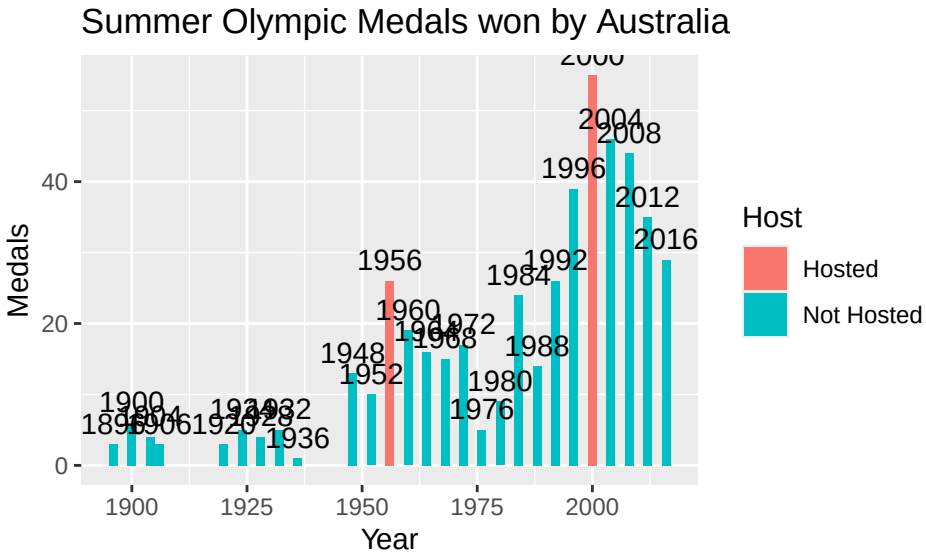
For the second question... I started by joining data sets together and creating a function that will filter the joint dataset for each country and in each of the seasons: determine whether they hosted or not. The function also displays a plot to compare the amount of medals that country won when they hosted vs when they did not. We will then compare and draw reasonable conclusions by creating a histogram containing the average number of medals all countries combined have won when they host vs in the competitions before.

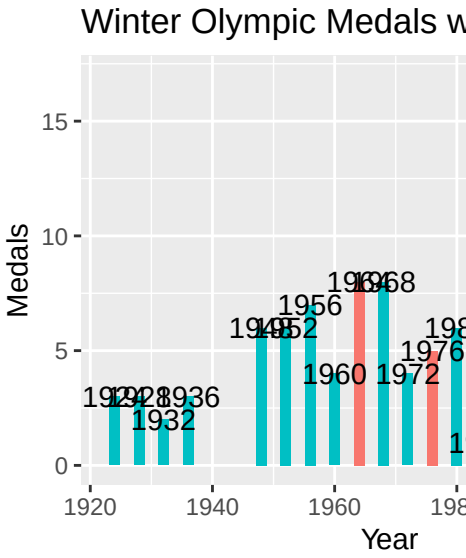
```
generate_country_medals_plot <- function(country_code, country_name, summer_hosts, winter_hosts) {

  # SUMMER
  summer_plot <- NULL

  if (length(summer_hosts) > 0) {
    summer_medals <- data %>%
      filter(NOC == country_code & !is.na(Medal) & Season == "Summer" & Year %in% c(1896:2016)) %>%
      distinct(Year, Event) %>%
      group_by(Year) %>%
      summarise(Medal_Count = n())

    summer_medals$Host <- ifelse(summer_medals$Year %in% summer_hosts, "Hosted", "Not Hosted")
```

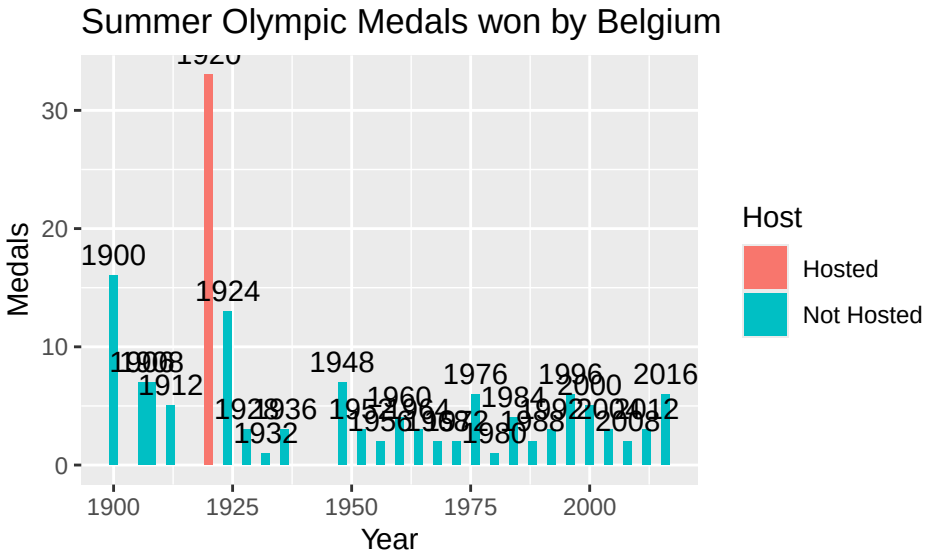
```
    summer_plot <- ggplot(summer_medals, aes(x = Year, y = Medal_Count, fill = Host)) +
      geom_bar(stat = "identity", position = "dodge") +
      geom_text(aes(label = Year), vjust = -0.5, position = position_dodge(width = 0.9)) +
      labs(x = "Year", y = "Medals", title = paste("Summer Olympic Medals won by", country_name))
  }

  # WINTER
  winter_plot <- NULL

  if (length(winter_hosts) > 0) {
    winter_medals <- data %>%
      filter(NOC == country_code & !is.na(Medal) & Season == "Winter" & Year %in% c(1896:2016)) %>%
      distinct(Year, Event) %>%
      group_by(Year) %>%
      summarise(Medal_Count = n())

    winter_medals$Host <- ifelse(winter_medals$Year %in% winter_hosts, "Hosted", "Not Hosted")

    winter_plot <- ggplot(winter_medals, aes(x = Year, y = Medal_Count, fill = Host)) +
      geom_bar(stat = "identity", position = "dodge") +
      geom_text(aes(label = Year), position = position_dodge(width = 0.9)) +
      labs(x = "Year", y = "Medals", title = paste("Winter Olympic Medals won by", country_name))
  }

  list(summer_plot = summer_plot, winter_plot = winter_plot)
}
```

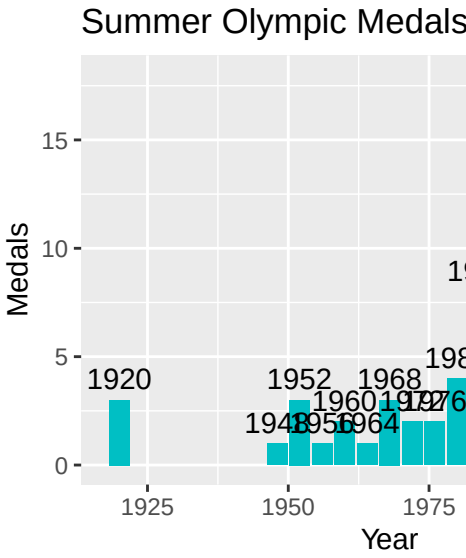


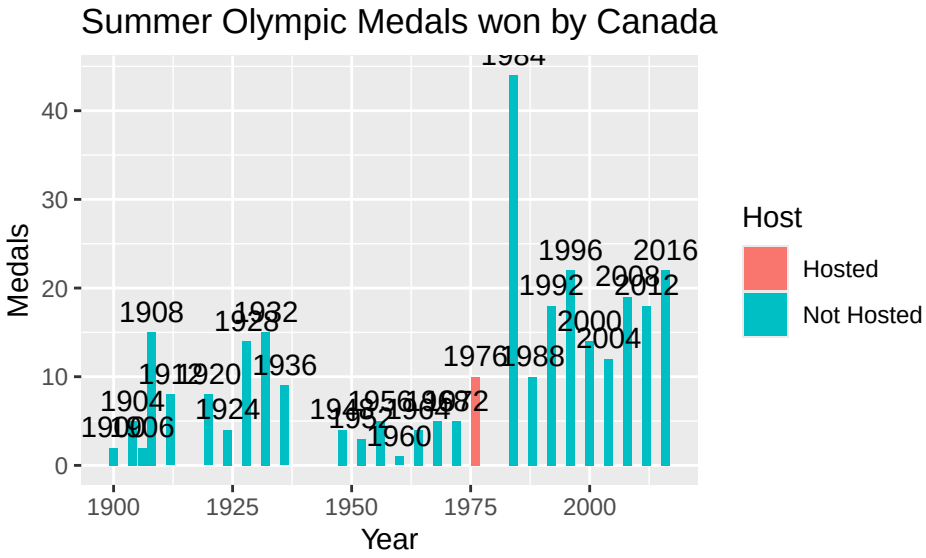


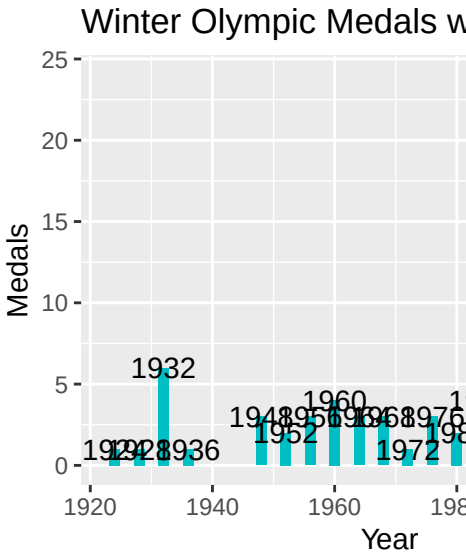

Summer Olympic Medals won by Australia
Medals
Year
Host
Hosted
Not Hosted

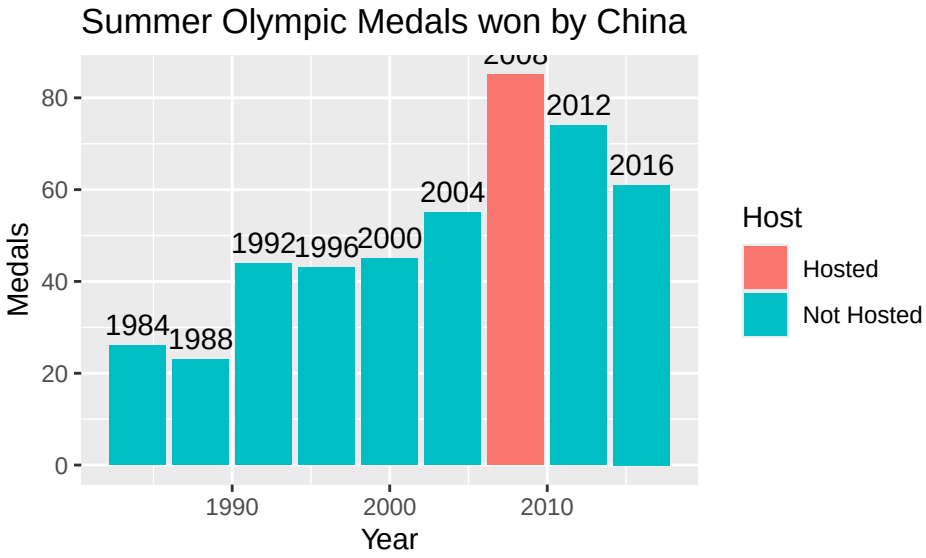

Winter Olympic Medals w
Medals
Year

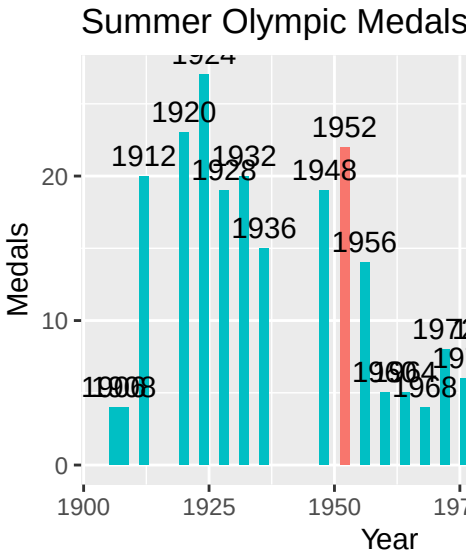

Summer Olympic Medals won by Belgium
Medals
Year
Host
Hosted
Not Hosted

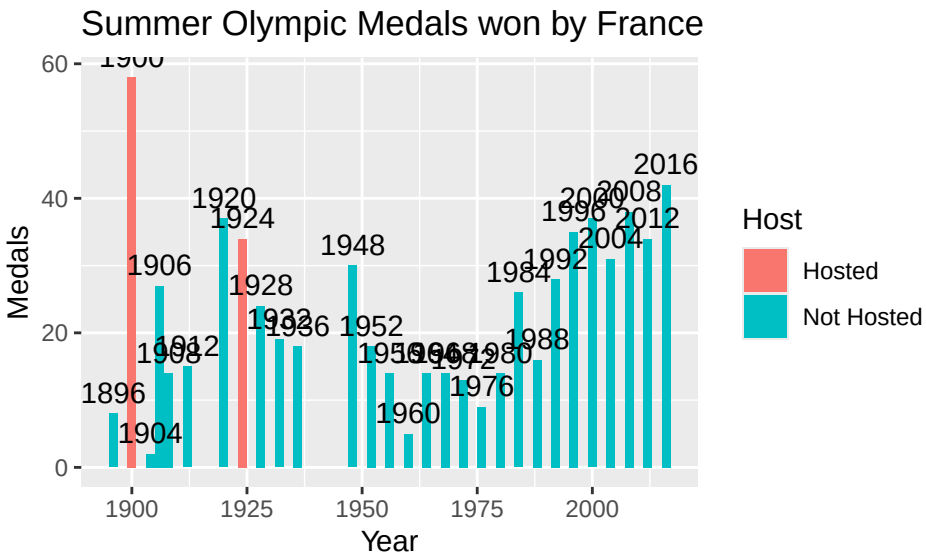

Summer Olympic Medals
Medals
Year

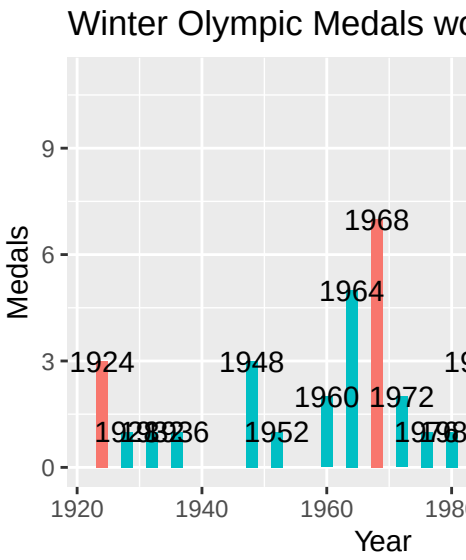

Summer Olympic Medals won by Canada
Medals
Year
Host
Hosted
Not Hosted

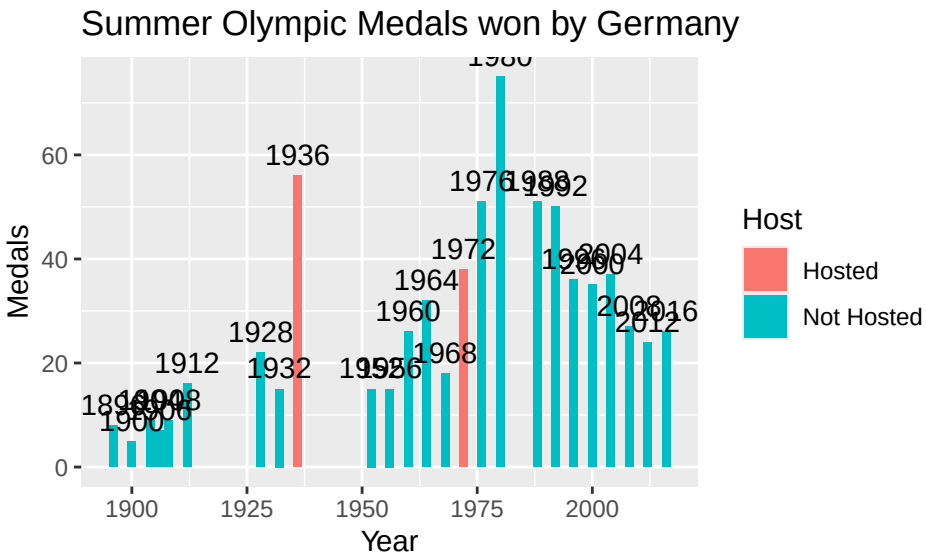

Winter Olympic Medals w
Medals
Year

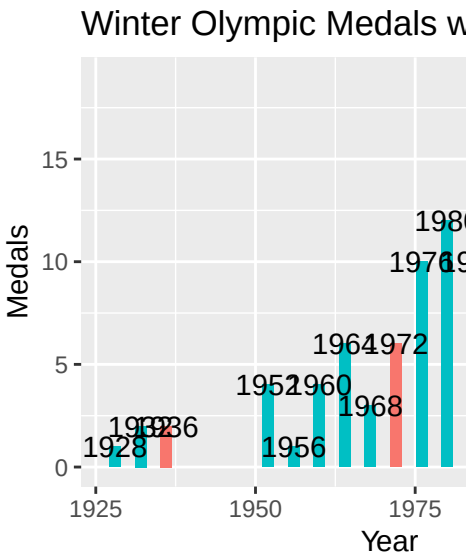

Summer Olympic Medals won by China
Medals
Year
Host
Hosted
Not Hosted

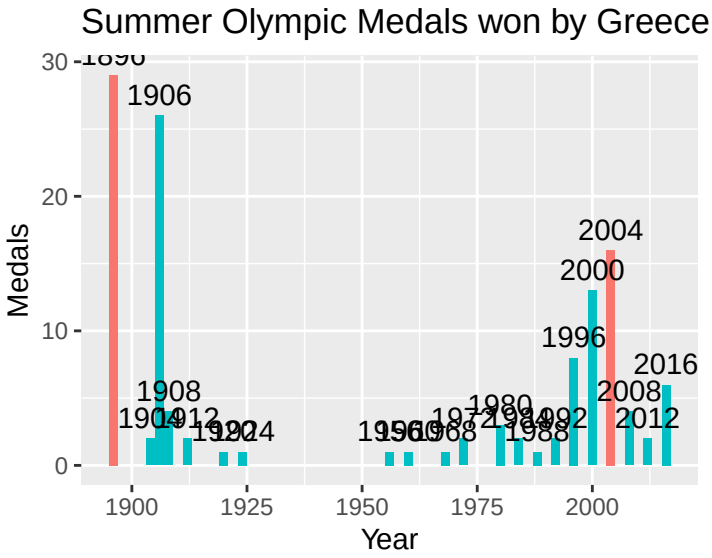

Summer Olympic Medals
Medals
Year

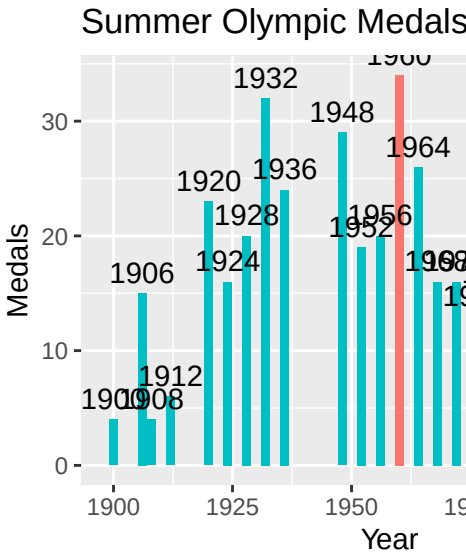

Summer Olympic Medals won by France
Medals
Year
Host
Hosted
Not Hosted

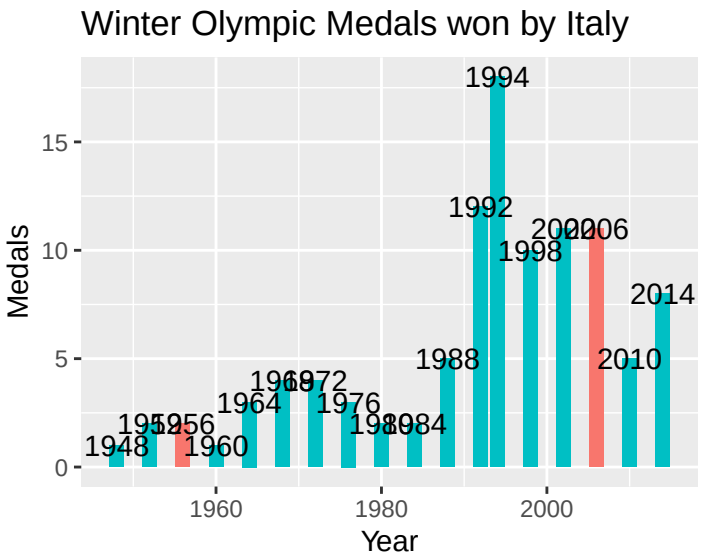

Winter Olympic Medals wo
Medals
Year

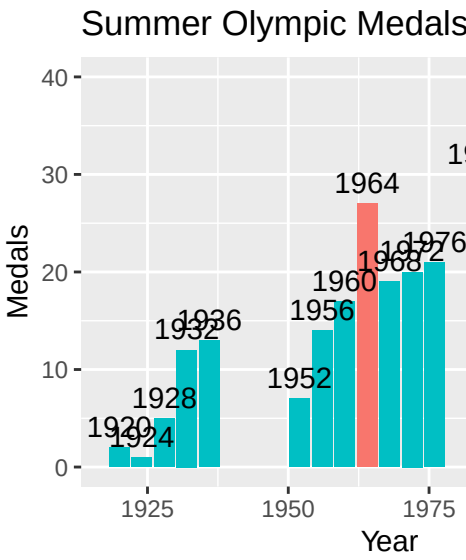

Summer Olympic Medals won by Germany
Medals
Year
Host
Hosted
Not Hosted

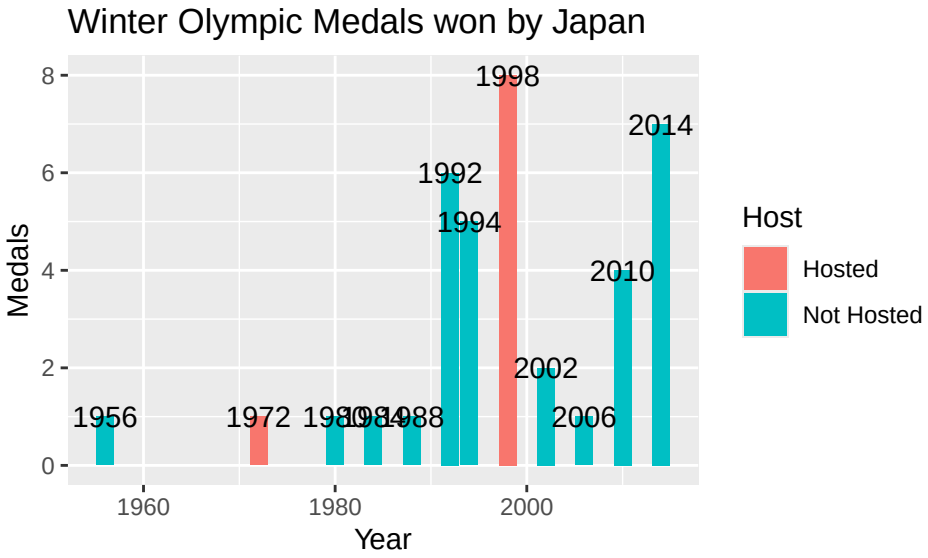

Winter Olympic Medals w
Medals
Year

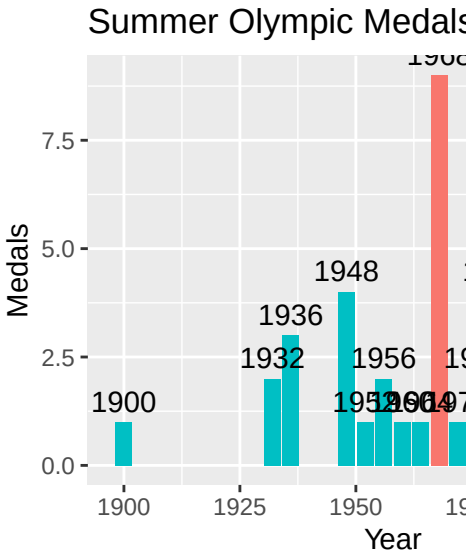

Summer Olympic Medals won by Greece
Medals
Year
Host
Hosted
Not Hosted

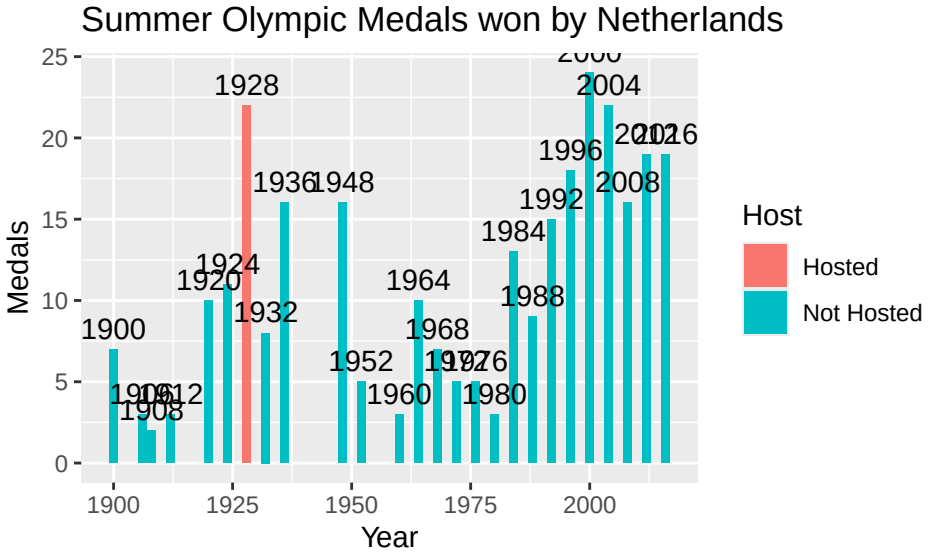

Summer Olympic Medals
Medals
Year

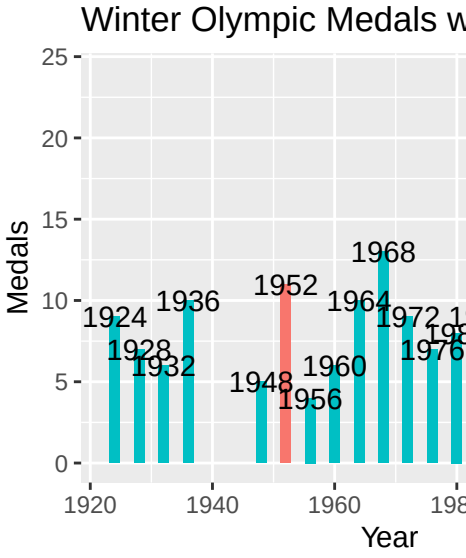

Winter Olympic Medals won by Italy
Medals
Year
Host
Hosted
Not Hosted

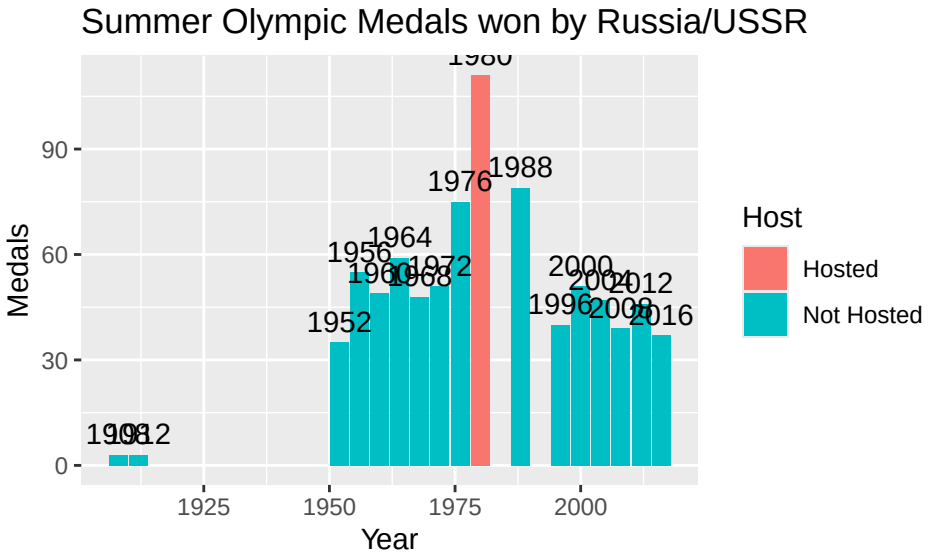

Summer Olympic Medals
Medals
Year

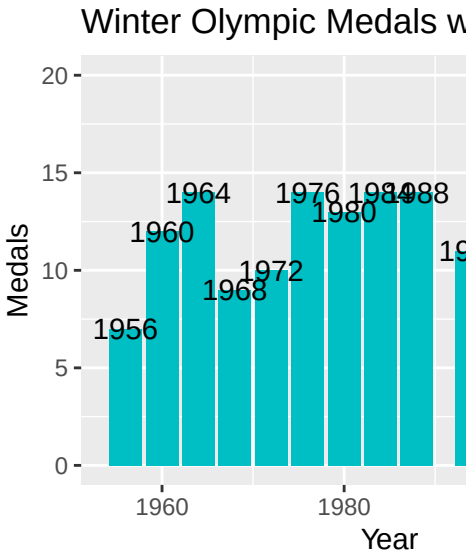

Winter Olympic Medals won by Japan
Medals
Year
Host
Hosted
Not Hosted
1956
1972
1998
2014
1992
1994
2010
2002
2006

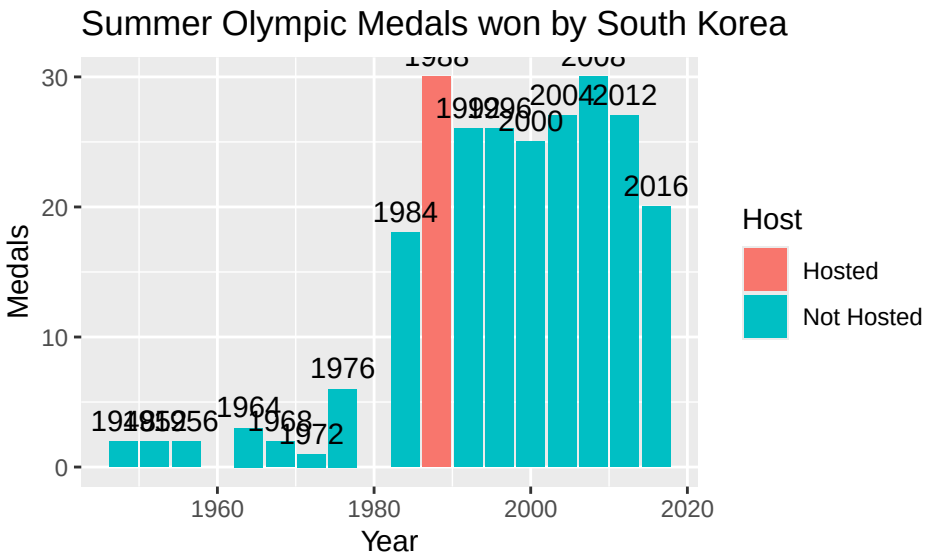

Summer Olympic Medals
Medals
Year
1900
1936
1932
1948
1956

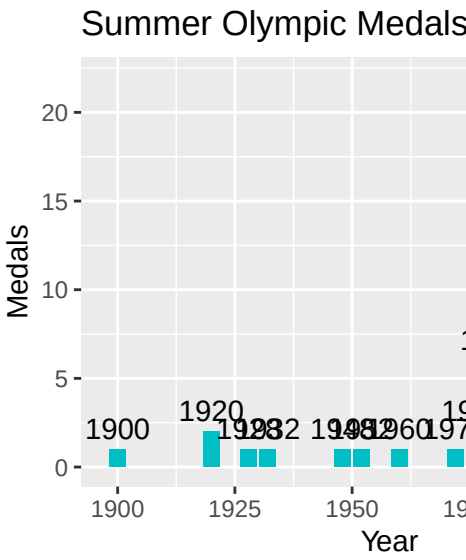

Summer Olympic Medals won by Netherlands
Medals
Year
Host
Hosted
Not Hosted
1928
2004
1936
1948
1996
2008
1992
1984
1924
1920
1932
1964
1988
1900
1968
1952
1960
1980

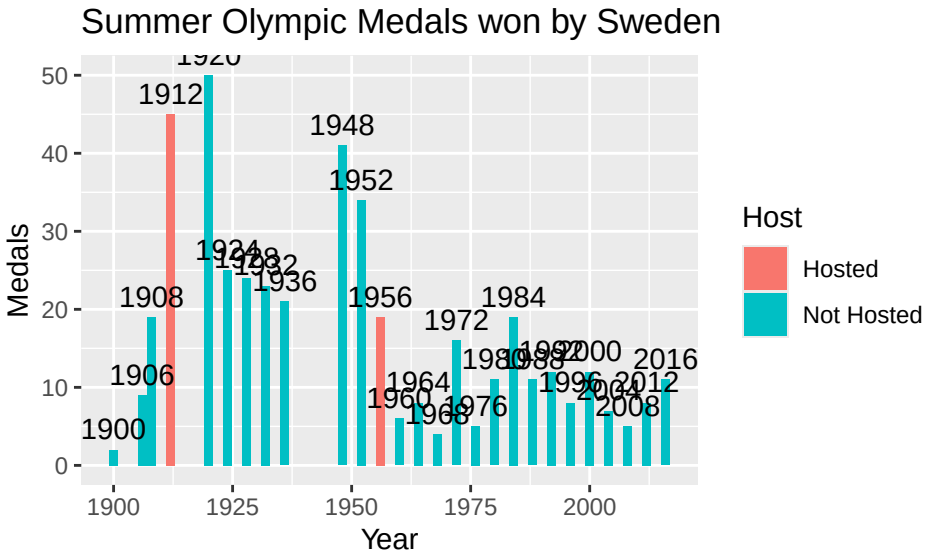

Winter Olympic Medals w
Medals
Year
1924
1936
1952
1968
1928
1932
1964
1972
1948
1960
1956
1976

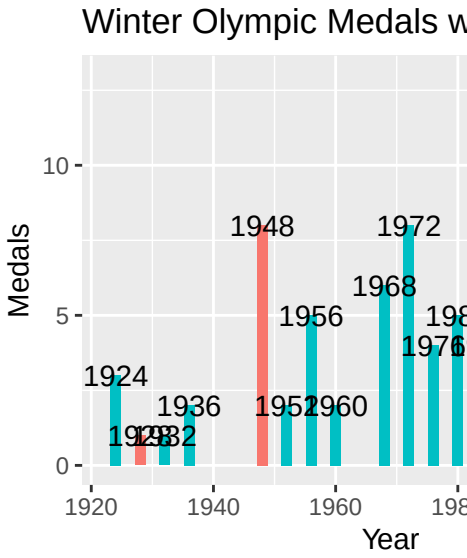

Summer Olympic Medals won by Russia/USSR
Medals
Year
Host
Hosted
Not Hosted

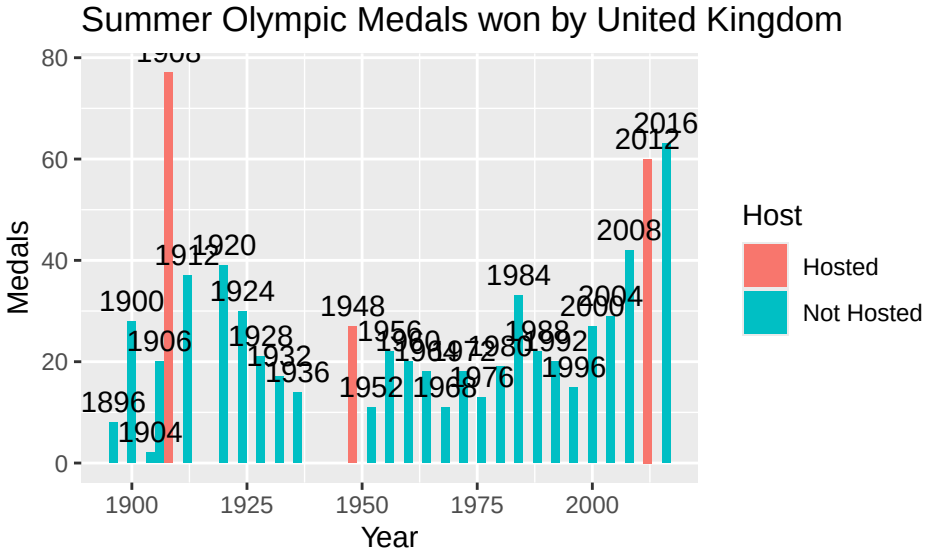

Winter Olympic Medals w
Medals
Year

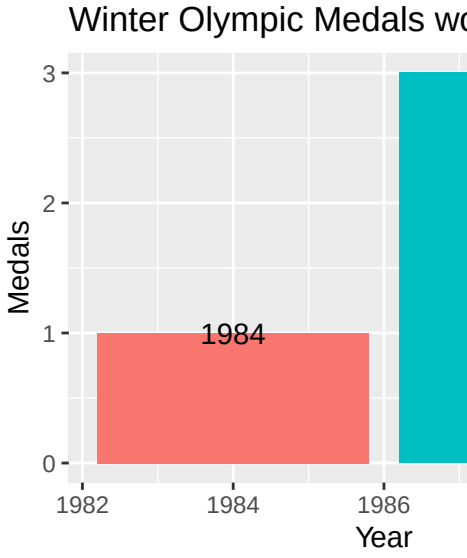

Summer Olympic Medals won by South Korea
Medals
Year
Host
Hosted
Not Hosted


Summer Olympic Medals
Medals
Year



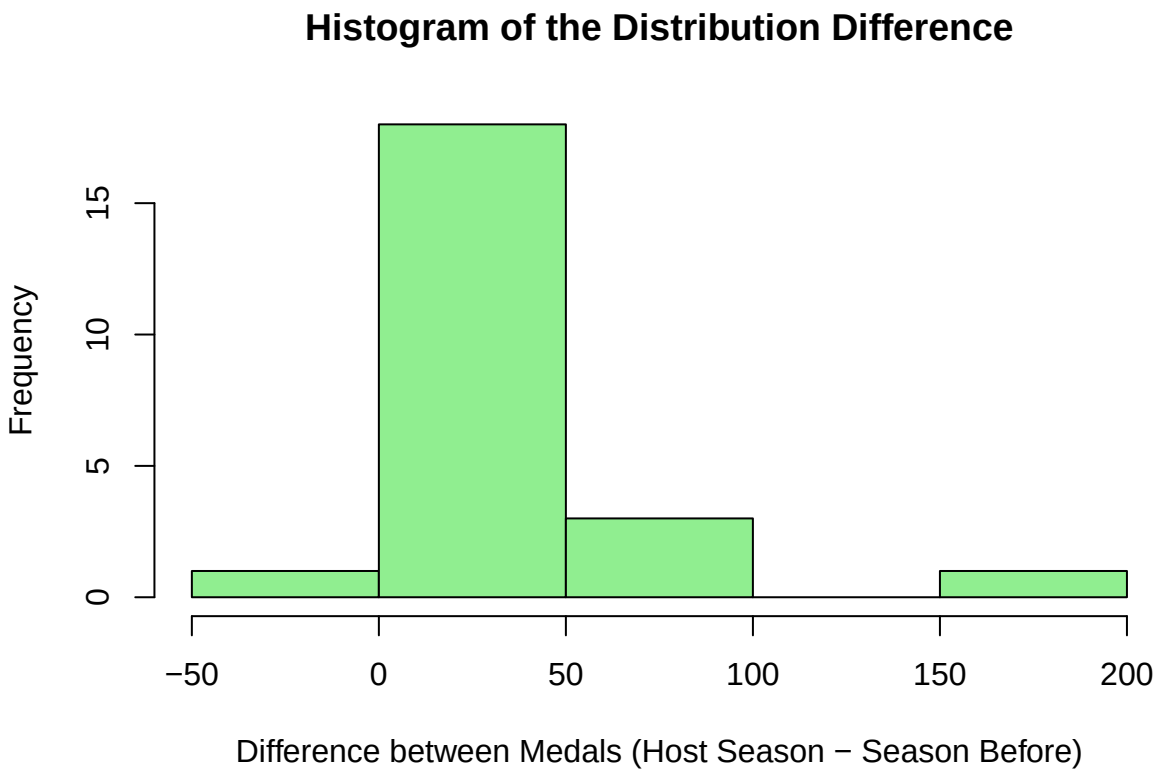






As stated earlier, I created a histogram of the difference of medals **(by subtracting the medals won when they host minus the medals won in the olympic season directly prior)** to draw a reasonable conclusion.

```
hist_data <- tibble(
  NOC = c("USA", "AUS", "AUT", "BEL", "BRA", "CAN", "CHN", "FIN", "FRA", "GER", "GRE", "ITA", "JPN", "M
  Medals_Won_Host = c(451, 83, 13, 33, 18, 39, 85, 22, 109, 102, 45, 47, 36, 9, 22, 29, 130, 30, 22, 64
  Medals_Won_Year_Before = c(265, 49, 8, 5, 17, 29, 55, 19, 52, 39, 16, 33, 22, 1, 11, 18, 84, 18, 4, 5

new <- hist_data %>%
  mutate(Distribution_difference = Medals_Won_Host - Medals_Won_Year_Before)

hist(new$Distribution_difference, xlab = "Difference between Medals (Host Season - Season Before)", mai
```



We can see there is a positive host effect country on the amount of medals won when a country hosts the olympics vs when they don't because there is an overall positive difference.

I am also currently in the process of completing my seventh project - Developing 2 more models to predict the if a patient has Alzheimers, and predicting the champion of the Premier League next season.