STAT 542 Final Project R Code

Savannah Doughty, Jackson Harness, and Soraya Remaili

April 28, 2025

Part I: Importing and Joining Data

```
## Importing the 'readr' package to use 'read_csv()' function
library(readr)
## Importing the 'tidyverse' package for various data handling needs
library(tidyverse)
## Importing the separate .csv files
## We are only keeping the attributes that are necessary for the analysis

## Keeping the circuitID, circuit name, location,
## latitude, longitude, and altitude
circuits <- read_csv(file="circuits.csv") %>%
    select(circuitId, circuitRef, location, lng, lat, alt)

## Keeping the constructorID, constructor name, and constructor nationality
constructors <- read_csv(file="constructors.csv")  %>%
    select(constructorId, constructorRef, nationality)

## Keeping the driverID, driver name, and driver nationality
drivers <- read_csv(file="drivers.csv") %>%
    select(driverId, driverRef, nationality)

## Keeping the raceID, date of the race, year,
## and the circuitID (to join later)
races <- read_csv(file="races.csv") %>%
    select(raceId, date, year, circuitId)

## Keeping the raceID, driverID, constructorID (to join later),
## starting position, final position, and points scored
```

```r
results <- read_csv(file="results.csv") %>%
    select(raceId, driverId, constructorId, grid, position, points)

## For the standings data, keeping raceID, driver/constructor ID,
## and driver/constructor points and position (cumulative over the season)
driver_standings <- read_csv(file="driver_standings.csv") %>%
    select(raceId, driverId, points, position)

const_standings <- read_csv(file="constructor_standings.csv") %>%
    select(raceId, constructorId, points, position)

## Joining the results table to contain information
## from the different .csv files
results_joined <- results %>%
## We join with races, circuits,
## and drivers/constructors using the key values
inner_join(races, by = "raceId") %>%
inner_join(circuits, by = "circuitId") %>%
## Driver and constructors both have a 'nationality' field,
## so we rename them to be unique
inner_join(drivers, by = "driverId") %>%
rename(driver_nationality = nationality) %>%
inner_join(constructors, by = "constructorId") %>%
rename(constructor_nationality = nationality)

## Joining the standings tables to include information
## about both driver and constructor standings

## We must rename 'points' and 'position' in each of the tables
## so that they are unique
standings_joined <- driver_standings %>%
rename(driver_points = points, driver_position = position) %>%
    inner_join(results_joined %>% select(driverId, raceId, constructorId),
 by = c("raceId", "driverId")) %>%
inner_join(const_standings, by = c("raceId", "constructorId")) %>%
rename(const_points = points, const_position = position) %>%
inner_join(races, by = "raceId")
```

Part II: Battle of the British Constructors

```r
## Grouping by constructor nationality, we count
## the amount of race results they have
results_joined %>%
    group_by(constructor_nationality) %>%
    summarize(N = n()) %>%
    arrange(desc(N))

## Since British is the most common nationality,
## we look at the British constructors with the most results
results_joined %>%
filter(constructor_nationality == "British") %>%
group_by(constructorRef) %>%
summarize(N = n()) %>%
    arrange(desc(N))

## Importing 'wesanderson' package for color palettes
library(wesanderson)

## Doing a McLaren vs. Williams vs. Tyrrell analysis
## Selecting these three constructors and their results from our data
names <- constructors %>%
  filter(constructorRef == "mclaren" | constructorRef == "williams"
  | constructorRef == "tyrrell")
battle_brits <- standings_joined %>%
    filter(constructorId %in% names$constructorId) %>%
## Since we will be log-transforming the data,
## points must be greater than 0
  filter(!is.na(const_points), const_points > 0) %>%
  inner_join(names %>%
               select(constructorRef, constructorId), by = "constructorId")

## Plotting the constructor points over time
## (color signifies the constructors)
g <- ggplot(data = battle_brits,
            aes(y = const_points, x = year, color = constructorRef))
g + geom_point(size = 1) +
  geom_smooth(method = "loess", se = FALSE, na.rm = T) +
  labs(color = "Constructor") +xlab("Year") +
  ylab("Average Points Scored") +
  scale_color_manual(values = wes_palette("FantasticFox1")) +
  coord_trans(y = "log10")
```

Part III: Analysis of Position

```r
## We want to look at the drivers most able to make comebacks
## We need to convert 'position', a character, to a numeric value
results_joined <- results_joined %>%
mutate(position = as.numeric(position))

## We define a comeback by the driver's end position
## vs. their start position (the bigger the difference, the better)
comebacks <- results_joined %>%
    group_by(driverRef) %>%
    summarize(avgComeback = mean(grid - position, na.rm = T), N = n()) %>%
    arrange(desc(avgComeback))

## However, many of the drivers with good average comebacks
## raced very few times (let's institute a 50 race limit)
comebacks_mult <- results_joined %>%
    group_by(driverRef) %>%
    summarize(avgComeback = mean(grid - position, na.rm = T), N = n()) %>%
    filter(N > 50) %>%
    arrange(desc(avgComeback))

## Doing the same analysis, but for constructors instead
## Here, we'll institute a 200 race limit (which is proportional)
comebacks_mult <- results_joined %>%
    group_by(constructorRef) %>%
    summarize(avgComeback = mean(grid - position, na.rm = T), N = n()) %>%
    filter(N > 200) %>%
    arrange(desc(avgComeback))

## Looking at drivers who run the biggest deficits
## a.k.a. drivers who lose the most places
deficits<- results_joined %>%
    group_by(driverRef) %>%
    summarize(avgDeficit = mean(position - grid, na.rm = T), N = n()) %>%
    filter(N > 50) %>%
    arrange(desc(avgDeficit))

## Would a worse driver be cut before 50 races?
## Let's lower the required N a bit
deficits<- results_joined %>%
    group_by(driverRef) %>%
```

```
    summarize(avgDeficit = mean(position - grid, na.rm = T), N = n()) %>%
    filter(N > 5) %>%
    arrange(desc(avgDeficit))

## Finding the drivers who start
## from pole most often (we see the more famous ones…)
best_poles<- results_joined %>%
    filter(grid == 1) %>%
    group_by(driverRef) %>%
    summarize(N = n()) %>%
    arrange(desc(N))

## Finding the constructors who start
## from pole most often (we see the more famous ones…)
best_poles<- results_joined %>%
    filter(grid == 1) %>%
    group_by(constructorRef) %>%
    summarize(N = n()) %>%
    arrange(desc(N))

## Finding the drivers who are the best
## at qualifying position overall
best_starts<- results_joined %>%
    group_by(driverRef) %>%
    summarize(avgStart = mean(grid, na.rm = T), N = n()) %>%
    filter(N > 50) %>%
    arrange(avgStart)
```

Part IV: Geographic Races

```
## Finding the most common circuits (most raced at)
by_circuits <- results_joined %>%
    group_by(circuitRef) %>%
    summarize(N = n() ) %>%
    inner_join(circuits, by = "circuitRef") %>%
    arrange(desc(N))

## Loading the 'mosaic' package to get world map data
library(mosaic)
world_map <- mosaic::mWorldMap()$data
```

```r
## Using latitude and longitude to plot
## altitude of each racing circuit
ggplot() + geom_polygon(data = world_map,
                        aes(x = long, y = lat, group = group),
                        fill = "lightgray", color = "white") +
  geom_point(data = by_circuits,
             aes(x = lng, y = lat, color = alt), size = 3) +
  scale_color_gradientn(colors = wes_palette("Zissou1"),
                        name = "Altitude")

## Plotting the number of races raced at that circuit (worldwide)
ggplot() + geom_polygon(data = world_map,
                        aes(x = long, y = lat, group = group),
                        fill = "lightgray", color = "white") +
  geom_point(data = by_circuits,
             aes(x = lng, y = lat, color = N), size = 3) +
  scale_color_gradientn(colors = wes_palette("Zissou1"),
                        name = "Number of Races")

## Plotting the number of races again, but focused on Europe
ggplot() + geom_polygon(data = world_map,
                        aes(x = long, y = lat, group = group),
                        fill = "lightgray", color = "white") +
  geom_point(data = by_circuits,
             aes(x = lng, y = lat, color = N), size = 3) +
             coord_fixed(xlim = c(-25, 45), ylim = c(35, 75))+
  scale_color_gradientn(colors = wes_palette("Zissou1"),
                        name = "Number of Races")

## Let's see who the top 3 drivers (in races won) are
most_wins <- results_joined %>%
    filter(position == 1) %>%
    group_by(driverRef) %>%
    summarize(N = n()) %>%
    arrange(desc(N))

## We'll first look at Lewis Hamilton
l_hamilton <- results_joined %>%
    filter(driverRef == "hamilton" ) %>%
    group_by(circuitRef) %>%
    summarize(N = n(), avgPts = mean(points, na.rm = T)) %>%
    arrange(desc(N))
```

```r
## Joining his results so we can make the map
by_circuits <- results_joined %>%
    group_by(circuitRef) %>%
    summarize(N = n() ) %>%
    inner_join(circuits, by = "circuitRef") %>%
    inner_join(l_hamilton, by = "circuitRef") %>%
    arrange(desc(N.y))

## His most raced circuit is his home GP
## Let's plot the other GPs on the map
ggplot() + geom_polygon(data = world_map,
                        aes(x = long, y = lat, group = group),
                        fill = "lightgray", color = "white") +
  geom_point(data = by_circuits,
             aes(x = lng, y = lat, color = N.y, size = avgPts))+
  scale_color_gradientn(colors = wes_palette("Zissou1"),
                        name = "Number of Races")

## Now let's look at Michael Schumacher
m_schum <- results_joined %>%
    filter(driverRef == "michael_schumacher" ) %>%
    group_by(circuitRef) %>%
    summarize(N = n(), avgPts = mean(points, na.rm = T)) %>%
    arrange(desc(N))

## Joining his results so we can make the map
by_circuits <- results_joined %>%
    group_by(circuitRef) %>%
    summarize(N = n() ) %>%
    inner_join(circuits, by = "circuitRef") %>%
    inner_join(m_schum, by = "circuitRef") %>%
    arrange(desc(N.y))

## His most raced circuit is the Barcelona GP
## Let's plot the other GPs on the map
ggplot() + geom_polygon(data = world_map,
                        aes(x = long, y = lat, group = group),
                        fill = "lightgray", color = "white") +
  geom_point(data = by_circuits,
             aes(x = lng, y = lat, color = N.y, size = avgPts))+
  scale_color_gradientn(colors = wes_palette("Zissou1"),
                        name = "Number of Races")
```

```
## Finally, let's look at Max Verstappen
m_verstap <- results_joined %>%
    filter(driverRef == "max_verstappen" ) %>%
    group_by(circuitRef) %>%
    summarize(N = n(), avgPts = mean(points, na.rm = T)) %>%
    arrange(desc(N))

## Joining his results so we can make the map
by_circuits <- results_joined %>%
    group_by(circuitRef) %>%
    summarize(N = n() ) %>%
    inner_join(circuits, by = "circuitRef") %>%
    inner_join(m_verstap, by = "circuitRef") %>%
    arrange(desc(N.y))

## His most raced circuit is the Austrian GP
## (that's his constructor's home race)
## Let's plot the other GPs on the map
ggplot() + geom_polygon(data = world_map,
                        aes(x = long, y = lat, group = group),
                        fill = "lightgray", color = "white") +
  geom_point(data = by_circuits,
             aes(x = lng, y = lat, color = N.y, size = avgPts))+
  scale_color_gradientn(colors = wes_palette("Zissou1"),
                        name = "Number of Races")
```

Part V: Fathers and Sons

```
## Let's do a father-son comparison
## We'll use these pairs:
## the Rosbergs, the Verstappens, and the Schumachers

## For each pair:
## Get the father/son data and give
## them the correct role (father vs. son)

## All the fathers started racing
## before 1995, so that's our cutoff

## Need to rename Nico Rosberg's driverRef
```

```r
rosbergs <- results_joined %>%
    filter(str_detect(driverRef, "rosberg")) %>%
group_by(driverRef) %>%
    summarize(N = n(), startYear = min(year), driverId = min(driverId)) %>%
    mutate(relationship = ifelse(startYear < 1995, "father", "son"))
rosbergs$driverRef[rosbergs$driverRef == "rosberg"] <- "nico_rosberg"

## Need to rename Jos Verstappen's driverRef
verstpns <- results_joined %>%
    filter(str_detect(driverRef, "verstappen")) %>%
group_by(driverRef) %>%
    summarize(N = n(), startYear = min(year), driverId = min(driverId)) %>%
    mutate(relationship = ifelse(startYear < 1995, "father", "son"))
verstpns$driverRef[verstpns$driverRef == "verstappen"] <- "jos_verstappen"

## Must use regex to differentiate from Ralf Schumacher
## i.e. we only want Mick or Michael Schumacher
schums <- results_joined %>%
    filter(str_detect(driverRef, "^[^r].*schumacher")) %>%
group_by(driverRef) %>%
    summarize(N = n(), startYear = min(year), driverId = min(driverId)) %>%
    mutate(relationship = ifelse(startYear < 1995, "father", "son"))

## Joining the father/son points data with the standings data
fathers_sons <- rbind(rosbergs, verstpns, schums)
fathers_sons_joined <- filter(standings_joined,
                              driverId %in% fathers_sons$driverId)
fathers_sons_standings <- fathers_sons_joined %>%
    inner_join(fathers_sons, by = "driverId") %>%
    group_by(driverId, year) %>%
    summarize(N=n(),
              avgPts = mean(driver_points, na.rm = T),
              driverRef = min(driverRef),
              relationship = min(relationship))

## Plotting the average points in a season between fathers and sons
g <- ggplot(data = fathers_sons_standings,
            aes(y = avgPts, x = year,
                color =driverRef,
                shape = relationship))
g + geom_point(size = 5) + scale_color_brewer(palette = "Dark2")
```

```r
## Joining the father/son position data with the standings data
fathers_sons_standings <- fathers_sons_joined %>%
    inner_join(fathers_sons, by = "driverId") %>%
    group_by(driverId, year) %>%
    summarize(N=n(), avgPos= mean(driver_position, na.rm = T), driverRef =
min(driverRef), relationship = min(relationship))

## Plotting the average position in a season between fathers and sons
## Note that the y-axis is flipped
## to reduce confusion about 'position' being from 1st-20th
g <- ggplot(data = fathers_sons_standings,
            aes(y = avgPos, x = year,
                color =driverRef,
                shape = relationship))
g+ geom_point(size = 5) +
  scale_color_brewer(palette = "Dark2") + scale_y_reverse()

## However, F1 regulations have changed over the years
## i.e. 20th was not always the last position
max_seats <- results_joined %>%
    group_by(year) %>%
    summarize(numSeats = max(grid))

## Ploting the change in the number of seats over the years
g <- ggplot(data = max_seats, aes(y = numSeats, x = year))
g+ geom_line(linewidth = 3, color = "#FF8815")

## Let's find the WDC (World Driver's Championship) years

## First, we find the last race of the season to find
## the points of the driver at the end of the season
last_races <-  fathers_sons_joined %>%
    group_by(year) %>%
    summarize(lastRace = max(date), races %>% filter(date == lastRace) %>%
select(raceId))

## Joining to get driver points
## and filter by the seasons that they were P1
wc_years <- last_races %>%
    inner_join(fathers_sons_joined %>%
                   select(-year, -date), by = "raceId") %>%
inner_join(fathers_sons %>%
```

```
            select(driverId, driverRef), by = "driverId") %>%
    filter(driver_position == 1)

## Adding this information to our graph
wc_seasons <- fathers_sons_standings %>%
 inner_join(wc_years %>% select(-driverId), by = c("year", "driverRef"))

## Let's add gold circles for WDC seasons
g <- ggplot(data = fathers_sons_standings,
            aes(y = avgPos, x = year,
                color =driverRef,
                shape = relationship))
g + geom_point(size = 3) + geom_point(data = wc_seasons,
                                      aes(x = year, y = avgPos),
                                      shape = 21, fill = NA,
                                      color = "gold",
                                      size = 6, stroke = 2)  +
  scale_color_brewer(palette = "Dark2") +scale_y_reverse()
```

Part VI: World Constructor's Championships

```
## What if we think about other uses for the championship data?

## Let's look at the constructors
consts <- standings_joined %>%
    inner_join(constructors, by = "constructorId") %>%
    inner_join(drivers %>% select(-nationality), by = "driverId")

## Finding the last race of the season to find
## the points of the constructor at the end of the season
last_races <-  results_joined%>%
    group_by(year) %>%
    summarize(lastRace = max(date), races %>% filter(date == lastRace) %>%
select(raceId))

## Let's only plot the eight most established
## teams in F1 to increase visibility on the graph
top_8 <- results_joined %>%
    group_by(constructorRef) %>%
    summarize(N = n()) %>%
```

```
    arrange(desc(N)) %>%
slice_head(n = 8)

## Calculating the average position of the constructor
consts_positions <- consts %>%
    filter(constructorRef %in% top_8$constructorRef) %>%
    group_by(constructorRef, year) %>%
    summarize(avgPos = mean(const_position, na.rm = T))

## Joining to get constructor points
## and filter by the seasons that they were P1
wc_years <- consts %>%
filter(date %in% last_races$lastRace, const_position == 1) %>%
    inner_join(consts_positions, by = c("year", "constructorRef"))

## Did one of the constructor's drivers win the WDC?
## Finding the 'double win' years
double_wins <- wc_years %>%
    filter(driver_position == 1)

## Let's plot the double-win seasons with a gold ring
## and only WCC seasons with a blue ring
g <- ggplot(data = consts_positions, aes(y = avgPos,
                                         x = year, color =constructorRef))
g + geom_point(size = 3) + geom_point(data = wc_years,
                                      aes(x = year, y = avgPos),
                                      shape = 21, fill = NA,
                                      color = "deepskyblue",
                                      size = 6, stroke = 2) +
  geom_point(data = double_wins,
             aes(x = year, y = avgPos),
             shape = 21, fill = NA, color = "gold",
             size = 6, stroke = 2)+
  scale_color_brewer(palette = "Set2") + scale_y_reverse()
```