Formula One Analysis

AUTHOR
Mandy Langlois

```
        library(lubridate)
```

```
Attaching package: 'lubridate'

The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

```
        library(ggthemes)
        library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr   1.1.4     ✔ readr   2.1.5
✔ forcats 1.0.0     ✔ stringr 1.5.1
✔ ggplot2 3.5.1     ✔ tibble  3.2.1
✔ purrr   1.0.2     ✔ tidyr   1.3.1

── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ 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(knitr)
        library(forecast)
```

```
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
```

Reading my data sets,producing a table and column names. This code below is reading my driver standing csv.

```
driver_standings = read.csv(here::here("driver_standing
head(driver_standings)
```

```
  driverStandingsId raceId driverId points position
positionText wins
1                 1     18        1     10        1
1    1
2                 2     18        2      8        2
2    0
3                 3     18        3      6        3
3    0
4                 4     18        4      5        4
4    0
5                 5     18        5      4        5
5    0
6                 6     18        6      3        6
6    0
```

Reading the CSV for the constructors.

```
constructors = read.csv(here::here("constructors.csv"))
head(constructors)
```

```
  constructorId constructorRef       name nationality
1             1        mclaren    McLaren     British
2             2     bmw_sauber BMW Sauber      German
3             3       williams   Williams     British
4             4        renault    Renault      French
5             5     toro_rosso Toro Rosso     Italian
6             6        ferrari    Ferrari     Italian
                                                    url
1                       http://en.wikipedia.org/wiki/McLaren
2                     http://en.wikipedia.org/wiki/BMW_Sauber
3 http://en.wikipedia.org/wiki/Williams_Grand_Prix_Engineering
4          http://en.wikipedia.org/wiki/Renault_in_Formula_One
5             http://en.wikipedia.org/wiki/Scuderia_Toro_Rosso
6                http://en.wikipedia.org/wiki/Scuderia_Ferrari
```

```
colnames(constructors)
```

```
[1] "constructorId"  "constructorRef" "name"
"nationality"
[5] "url"
```

Reading the CSV for results. produced the table and column names.

```
results = read.csv(here::here("results.csv"))
head(results)
```

```
  resultId raceId driverId constructorId number grid position
positionText
1        1     18        1             1     22    1        1
1
2        2     18        2             2      3    5        2
2
3        3     18        3             3      7    7        3
3
4        4     18        4             4      5   11        4
4
5        5     18        5             1     23    3        5
5
6        6     18        6             3      8   13        6
6
  positionOrder points laps        time milliseconds fastestLap
rank
1             1     10   58 1:34:50.616      5690616         39
2
2             2      8   58      +5.478      5696094         41
3
3             3      6   58      +8.163      5698779         41
5
4             4      5   58     +17.181      5707797         58
7
5             5      4   58     +18.014      5708630         43
1
6             6      3   57         \\N          \\N         50
14
  fastestLapTime fastestLapSpeed statusId
1       1:27.452         218.300        1
2       1:27.739         217.586        1
3       1:28.090         216.719        1
4       1:28.603         215.464        1
5       1:27.418         218.385        1
6       1:29.639         212.974       11
```

```
colnames(results)
```

```
 [1] "resultId"        "raceId"          "driverId"
"constructorId"
 [5] "number"          "grid"            "position"
"positionText"
 [9] "positionOrder"   "points"          "laps"
"time"
[13] "milliseconds"    "fastestLap"      "rank"
"fastestLapTime"
[17] "fastestLapSpeed" "statusId"
```

Reading the CSV for pit stops

```
pit_stops = read.csv(here::here("pit_stops.csv"))
head(pit_stops)
```

```
  raceId driverId stop lap     time duration milliseconds
1    841      153    1   1 17:05:23   26.898        26898
2    841       30    1   1 17:05:52   25.021        25021
3    841       17    1  11 17:20:48   23.426        23426
4    841        4    1  12 17:22:34   23.251        23251
5    841       13    1  13 17:24:10   23.842        23842
6    841       22    1  13 17:24:29   23.643        23643
```

```
colnames(pit_stops)
```

```
[1] "raceId"       "driverId"     "stop"         "lap"
"time"
[6] "duration"     "milliseconds"
```

Reading the CSV for qualifying

```
qualifying = read.csv(here::here("qualifying.csv"))
head(qualifying)
```

```
  qualifyId raceId driverId constructorId number position
q1       q2
1         1     18        1             1     22        1
1:26.572 1:25.187
2         2     18        9             2      4        2
1:26.103 1:25.315
3         3     18        5             1     23        3
1:25.664 1:25.452
4         4     18       13             6      2        4
```

```
1:25.994 1:25.691
5         5     18        2             2      3        5
1:25.960 1:25.518
6         6     18       15             7     11        6
1:26.427 1:26.101
        q3
1 1:26.714
2 1:26.869
3 1:27.079
4 1:27.178
5 1:27.236
6 1:28.527
```

```
colnames(qualifying)
```

```
[1] "qualifyId"     "raceId"        "driverId"
"constructorId"
[5] "number"        "position"      "q1"            "q2"
[9] "q3"
```

Reading the CSV for lap times

```
lap_times = read.csv(here::here("lap_times.csv"))
head(lap_times)
```

```
  raceId driverId lap position     time milliseconds
1    841       20   1        1 1:38.109        98109
2    841       20   2        1 1:33.006        93006
3    841       20   3        1 1:32.713        92713
4    841       20   4        1 1:32.803        92803
5    841       20   5        1 1:32.342        92342
6    841       20   6        1 1:32.605        92605
```

```
colnames(lap_times)
```

```
[1] "raceId"       "driverId"     "lap"          "position"
"time"
[6] "milliseconds"
```

I want to know what variables are most predictive of race outcome. Is it the lap times, pit stops, or qualifying positions. Before i start to find my answer i want to by filtering out any missing or inconsistent data that may be present in the relevant datasets.

Notes: **(sum(is.na(df$column_name)) / length(df$column_name)) * 100** (to check the percentage of NA values in a column length= nrow or ncol). The sum of NA values in data frame results in column or row "time" divided by the returned number of rows present in the data frame results multiplied by 100 to get me a percentage of NA values

```
sum(is.na(results$time)) / nrow(results) * 100
```

```
[1] 0
```

```
sum(is.na(qualifying$milliseconds)) / nrow(qualifying)
```

```
[1] 0
```

Merging data from lap_times, results, qualifying. using common column names like raceID and driverID to merge lap times and results and added the qualifying data to combined data.

Notes: Merging Format

- total (any name acceptable) = merge(data frameA, data frameB, by="ID") for
- total(any name acceptable) = merge(data frameA, data frameB, by=c("ID","Country"))

note, Error code received, 'by' must be specified uniquely valid columns. Error fixed, i wrote raceID instead of raceId. Merge was successful.

```
lap_results = merge(lap_times, results, by= c("raceId",
merged_f1 = merge(lap_results, qualifying, by = c("race
```

Relationship between qualifying and race results, plus visualizations.

Notes:

- summary is used as a generic function used to produce result summaries of the results of various model fitting functions.
    - Smoothing method (function) to use, accepts either Null or a character vector, e.g. "lm", "glm", "gam", "loess" or a function.

Analysis: There is a positive correlation between qualifying position

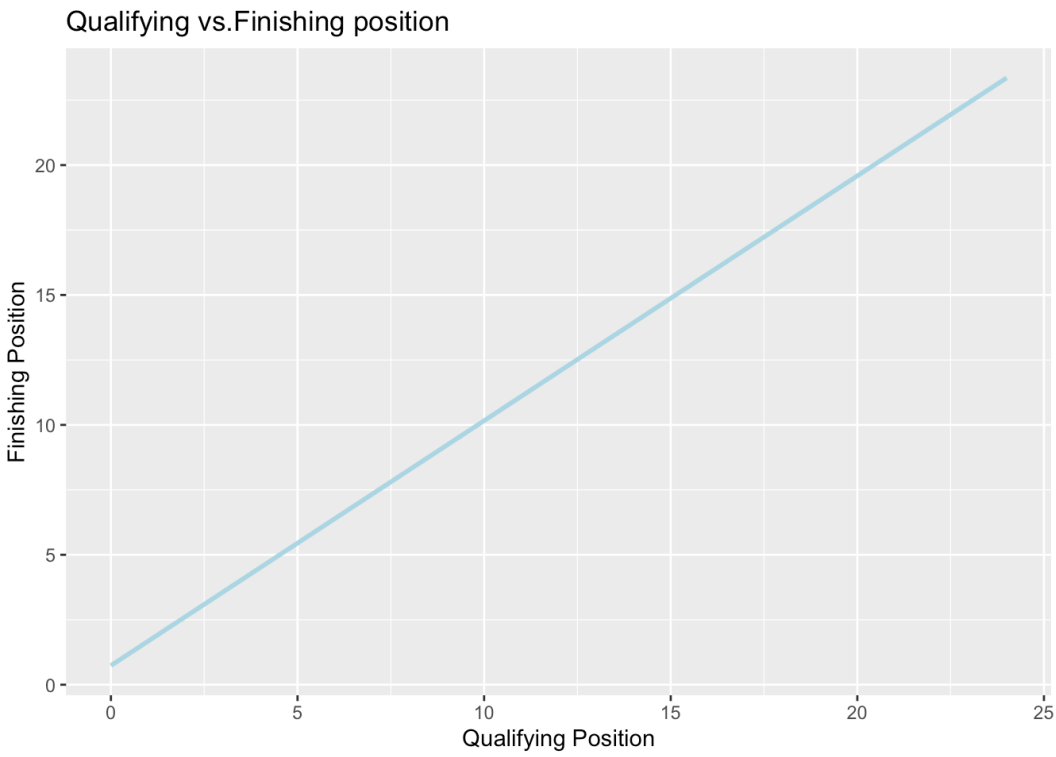
and finishing position, as grid positions determine starting grid positions on race days. This correlation is

```
summary(merged_f1$position)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   1.00    5.00   11.00   10.75   16.00   24.00 
```

```
ggplot(merged_f1, aes(x= grid, y = position)) +
  geom_smooth(method = "lm", color = "lightblue") +
  labs(title = "Qualifying vs.Finishing position", x= "
```

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



Calculating for the average lap time per driver per race.

Notes:

- **group_by** = organize data into groups based on the values in one or more columns. with other functions it can be used find the sum, mean, averages or counts for each group. Like sorting a spreadsheet by a specific column for example, a drivers name. Each “group” is like a section of the spreadsheet for that driver, and you can do math or

analysis on just that section.

- **na.rm** = a logical evaluating to TRUE or FALSE , Used when you have missing data and want R to ignore those NA values.

```
        avg_lap_time =  merged_f1 |>
          filter(!is.na(milliseconds.x)) |>
          group_by(raceId, driverId) |>
          summarize(avg_lap_time = mean(milliseconds, na.rm = T
```

```
Warning: There were 9088 warnings in `summarize()`.
The first warning was:
ℹ In argument: `avg_lap_time = mean(milliseconds, na.rm = T)`.
ℹ In group 1: `raceId = 1` and `driverId = 1`.
Caused by warning in `mean.default()`:
! argument is not numeric or logical: returning NA
ℹ Run `dplyr::last_dplyr_warnings()` to see the 9087 remaining
warnings.

`summarise()` has grouped output by 'raceId'. You can override
using the
`.groups` argument.
```

```
        head(avg_lap_time)
```

```
# A tibble: 6 × 3
# Groups:   raceId [1]
  raceId driverId avg_lap_time
   <int>    <int>        <dbl>
1      1        1           NA
2      1        2           NA
3      1        3           NA
4      1        4           NA
5      1        6           NA
6      1        7           NA
```

multiple columns are named milliseconds.x and milliseconds.y, this might be the reason why im getting NA in my columns. i will atempt to fix it below.

```
        summary(merged_f1$milliseconds.x)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
  55404   82171   92028   96735  102508 7507547
```

```
        avg_lap_time =  merged_f1 |>
          group_by(raceId, driverId) |>
          summarize(avg_lap_time = mean(milliseconds.x, na.rm =
```

```
`summarise()` has grouped output by 'raceId'. You can override
using the
`.groups` argument.
```

```
        head(avg_lap_time)
```

```
# A tibble: 6 × 3
# Groups:   raceId [1]
  raceId driverId avg_lap_time
   <int>    <int>        <dbl>
1      1        1       97564.
2      1        2       97636.
3      1        3       97612.
4      1        4       97598.
5      1        6       91822.
6      1        7       97622.
```

Renaming milliseconds.x to milliseconds for my sanity.

```
        merged_f1 = merged_f1 |>
          rename(milliseconds = milliseconds.x)

        avg_lap_time =  merged_f1 |>
          group_by(raceId, driverId) |>
          summarize(avg_lap_time = mean(milliseconds, na.rm = T
```

```
`summarise()` has grouped output by 'raceId'. You can override
using the
`.groups` argument.
```

```
        head(avg_lap_time)
```

```
# A tibble: 6 × 3
# Groups:   raceId [1]
  raceId driverId avg_lap_time
   <int>    <int>        <dbl>
```

```
1       1       1       97564.
2       1       2       97636.
3       1       3       97612.
4       1       4       97598.
5       1       6       91822.
6       1       7       97622.
```

I want to see the correlation between grid and positions.

Notes:

- **cor :** calculates the correlation coefficient between two numeric vectors. A correlation **measures the strength and direction of a linear relationship between two variables**. the correlation coefficient **(r)** always falls between **-1** and **1.**

    1. A value close to **1 (r =1)** indicates a strong positive relationship (as one variable increases, the other also increases). perfect positive correlation.

    2. A value close to **-1 (r = -1)** indicates a strong negative relationship (as one variable increases, the other decreases). perfect negative correlation.

    3. A value close to **0 (r= 0)** indicates no linear relationship. no correlation.

- “complete.obs” means to use only rows where **both variables have non-missing values**. If either grid or position has NA for a row, that row will be excluded from the correlation calculation.

    Analysis: A correlation coefficient of **0.946** is a value close to 1 indicating a strong positive relationship between the starting position and finish position. For example, If Lewis Hamilton started the 2025 Silverstone Grand Prix in pole position (1st) there’s a high very high chance of him finishing the ace in 1st or 2nd. my correlation coefficient suggest that grid position is a highly predictive of a win or a podium finish. If Alex Albon in his Williams qualifies 20th and last, the chances are high that he finishes near 20th. with that being said, there are cases where drivers have started last on the grid and end up on the podium, we saw an example of this at the 2021 Brazilian Grand Prix when he started the race weekend in last after taking a engine penalty, and made his way up to first. suggesting, factors like driver

skill also plays a significant role.

```
cor(merged_f1$grid, merged_f1$position, use = "complete
```

```
[1] 0.9465565
```

looking for the correlation between pit stops and and finishing positions

raceId: pit stops for specific races

driverId: pit stops to specific drivers

stop: # of pit stops

milliseconds: duration of pot stops

lap: number of laps before the pit stops

```
pit_stops = merge(merged_f1, pit_stops,by = c("raceId",
```

```
Warning in merge.data.frame(merged_f1, pit_stops, by =
c("raceId", "driverId"),
: column name 'milliseconds.y' is duplicated in the result
```

```
head(pit_stops)
```

```
  raceId driverId lap.x position.x   time.x milliseconds.x
resultId
1      1        1    55          6 1:30.149          90149
7573
2      1        1    34          8 1:29.723          89723
7573
3      1        1    52          7 1:29.722          89722
7573
4      1        1    41          5 1:29.824          89824
7573
5      1        1    35          8 1:29.474          89474
7573
6      1        1     1         13 1:49.088         109088
7573
  constructorId.x number.x grid position.y positionText
positionOrder points
1               1        1   18        \\N            D
```

```
20      0
2               1        1   18        \\N            D
20      0
3               1        1   18        \\N            D
20      0
4               1        1   18        \\N            D
20      0
5               1        1   18        \\N            D
20      0
6               1        1   18        \\N            D
20      0
  laps time.y milliseconds.y fastestLap rank fastestLapTime
fastestLapSpeed
1   58    \\N            \\N         39   13       1:29.020
214.455
2   58    \\N            \\N         39   13       1:29.020
214.455
3   58    \\N            \\N         39   13       1:29.020
214.455
4   58    \\N            \\N         39   13       1:29.020
214.455
5   58    \\N            \\N         39   13       1:29.020
214.455
6   58    \\N            \\N         39   13       1:29.020
214.455
  statusId qualifyId constructorId.y number.y position       q1
q2  q3 stop
1        2      3000              1        1       15 1:26.454
\\N \\N   NA
2        2      3000              1        1       15 1:26.454
\\N \\N   NA
3        2      3000              1        1       15 1:26.454
\\N \\N   NA
4        2      3000              1        1       15 1:26.454
\\N \\N   NA
5        2      3000              1        1       15 1:26.454
\\N \\N   NA
6        2      3000              1        1       15 1:26.454
\\N \\N   NA
  lap.y time duration milliseconds.y
1    NA <NA>     <NA>             NA
2    NA <NA>     <NA>             NA
3    NA <NA>     <NA>             NA
4    NA <NA>     <NA>             NA
5    NA <NA>     <NA>             NA
```

```
6    NA <NA>     <NA>             NA
```

```
head(pit_stops$`milliseconds.y`)
```

```
[1] "\\N" "\\N" "\\N" "\\N" "\\N" "\\N"
```

```
head(pit_stops$`milliseconds.y.1`)
```

```
NULL
```

```
colnames(pit_stops) = make.unique(colnames(pit_stops))
```

```
head(pit_stops)
```

```
  raceId driverId lap.x position.x   time.x milliseconds.x
resultId
1      1        1    55          6 1:30.149          90149
7573
2      1        1    34          8 1:29.723          89723
7573
3      1        1    52          7 1:29.722          89722
7573
4      1        1    41          5 1:29.824          89824
7573
5      1        1    35          8 1:29.474          89474
7573
6      1        1     1         13 1:49.088         109088
7573
  constructorId.x number.x grid position.y positionText
positionOrder points
1               1        1   18        \\N            D
20      0
2               1        1   18        \\N            D
20      0
3               1        1   18        \\N            D
20      0
4               1        1   18        \\N            D
20      0
5               1        1   18        \\N            D
20      0
6               1        1   18        \\N            D
20      0
```

```
  laps time.y milliseconds.y fastestLap rank fastestLapTime
fastestLapSpeed
1   58    \\N            \\N         39   13       1:29.020
214.455
2   58    \\N            \\N         39   13       1:29.020
214.455
3   58    \\N            \\N         39   13       1:29.020
214.455
4   58    \\N            \\N         39   13       1:29.020
214.455
5   58    \\N            \\N         39   13       1:29.020
214.455
6   58    \\N            \\N         39   13       1:29.020
214.455
  statusId qualifyId constructorId.y number.y position       q1
q2  q3 stop
1        2      3000              1        1       15 1:26.454
\\N \\N   NA
2        2      3000              1        1       15 1:26.454
\\N \\N   NA
3        2      3000              1        1       15 1:26.454
\\N \\N   NA
4        2      3000              1        1       15 1:26.454
\\N \\N   NA
5        2      3000              1        1       15 1:26.454
\\N \\N   NA
6        2      3000              1        1       15 1:26.454
\\N \\N   NA
  lap.y time duration milliseconds.y.1
1    NA <NA>     <NA>               NA
2    NA <NA>     <NA>               NA
3    NA <NA>     <NA>               NA
4    NA <NA>     <NA>               NA
5    NA <NA>     <NA>               NA
6    NA <NA>     <NA>               NA
```

```
colnames(pit_stops)[colnames(pit_stops) == "millisecond
```

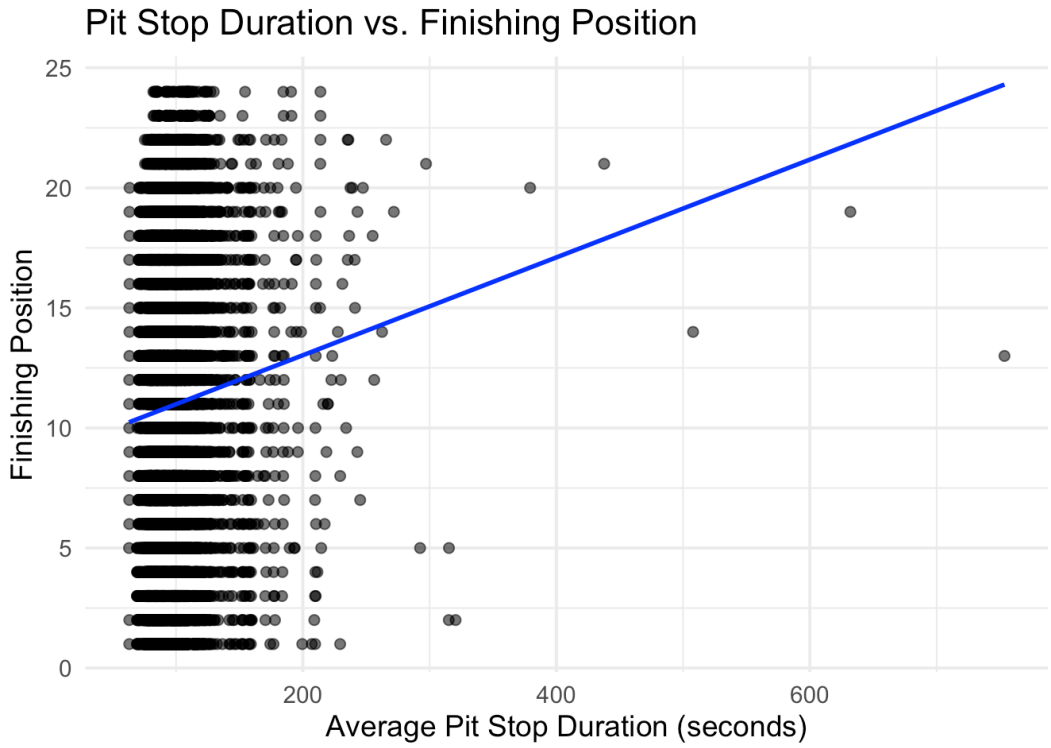
```
pit_stops_summary = pit_stops |>
  group_by(raceId, driverId) |>
  summarize(
    avg_pit_duration = mean(pit_stop_duration, na.rm =
    finishing_position = first(position)
  )
```

```
`summarise()` has grouped output by 'raceId'. You can override
using the
`.groups` argument.
```

Analysis: This scatter plot shows that, the longer the pit stops the lower your finishing position. Most of the drivers who stopped for 200 or less seconds fish in within point positions (1-10). The spread of points also indicate that other factors other than pit stops duration affect finishing positions, for example, you can see how some drivers who stop for longer than 200 seconds still finishing on the podium, if not 4th place. This suggest that factors other than pit stop duration play a role in finishing positions, factors like racing incidents like crashes that are yellow flagged and require a safety car that virtually allows the drivers to have a free pit stop without major grid position since all car will be going slow/ at a controlled pace behind the safety car. So do red flags, which require the cars to come into the pits and teams are allowed to change tires on their cars giving them a free pit stop without losing track position. other factors that may explain the higher grid finish for drivers with a long pit stops are, driver skills, race strategies to undercut or overcut, and car performance. This analysis confirms that efficient pt stops are important for better race outcomes but they arent the only determinant of success.

```
ggplot(pit_stops_summary, aes(x = avg_pit_duration / 10
  geom_point(size = 2, alpha = 0.6) +
  geom_smooth(method = "lm", se = FALSE, color = "blue"
  labs(
    title = "Pit Stop Duration vs. Finishing Position",
    x = "Average Pit Stop Duration (seconds)",
    y = "Finishing Position"
  ) +
  theme_minimal(base_size = 14)
```

```
Warning: Using `size` aesthetic for lines was deprecated in
ggplot2 3.4.0.
ℹ Please use `linewidth` instead.

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



Calculating for correlation coefficient

```
correlation = cor(pit_stops_summary$avg_pit_duration, p
head(correlation)
```

```
[1] 0.07733555
```

A table to summarize the data and help better understand the scatter plot above.

```
summary_table= pit_stops_summary |>
  summarize(
    min_pit_stop_duration = min(avg_pit_duration, na.rm
    max_pit_stop_duration = max(avg_pit_duration, na.rm
    avg_pit_duration = mean(finishing_position, na.rm=
    total_drivers = n()
  )
```

```
kable(summary_table, caption = "Summary of Pit Stop Dur
```

Summary of Pit Stop Durations and Finishing Positions

raceId	min_pit_stop_duration	max_pit_stop_duration	avg_pit_duration	tota

1	91822.24	101765.58	10.315789
2	107439.42	122738.00	10.315789
3	126133.66	149046.72	10.500000
4	96634.77	98949.60	10.500000
5	88472.76	128285.00	9.500000
6	77490.79	82015.73	10.500000
7	89393.93	97674.00	10.500000
8	82822.13	87337.22	10.500000
9	96721.83	102039.22	10.500000
10	84341.09	104266.00	10.526316
11	100899.81	105225.15	10.500000
12	114340.80	123219.33	9.250000
13	86447.28	91252.87	10.000000
14	114202.25	127015.67	10.500000
15	96049.40	104366.31	10.315789
16	78071.56	85998.38	11.352941
17	102607.53	107048.83	10.500000
18	95126.79	105505.88	10.352941
19	97831.34	170790.00	11.142857
20	95911.75	103582.05	11.142857
21	89031.38	116163.50	10.750000
22	89818.12	151878.00	10.052632
23	95299.24	105491.43	10.500000
24	82528.46	85853.03	10.500000
25	78717.79	86318.94	10.500000
26	98023.77	103206.79	10.611111
27	79083.34	85052.36	10.500000
28	83448.82	88336.86	10.500000
29	100567.35	104063.55	10.421053
30	111019.83	118350.77	10.500000
31	98254.60	109618.91	10.500000

32	112792.21	120278.03	10.500000
33	80034.75	92119.33	10.473684
34	98525.05	155805.50	10.500000
35	79597.68	82744.86	10.277778
36	88427.07	94102.00	11.500000
37	98838.04	109255.43	11.000000
38	98377.46	110132.38	10.900000
39	84557.38	136740.00	11.500000
40	77299.09	95137.00	11.500000
41	81411.67	91032.53	11.333333
42	74931.03	78238.16	11.315789
43	77917.14	139632.00	11.238095
44	83102.95	101542.71	11.500000
45	126439.30	235988.69	11.500000
46	82185.59	90820.00	11.500000
47	89692.43	97055.55	11.500000
48	89015.21	113567.40	11.500000
49	109978.77	157259.00	11.500000
50	107978.79	134646.32	11.500000
51	104971.34	115347.18	11.500000
52	74581.27	105989.00	11.500000
53	94494.82	121724.86	11.190476
54	97152.30	133359.96	11.714286
55	98678.94	131418.75	11.157895
56	88169.13	132617.22	11.095238
57	95979.42	130837.50	11.285714
58	78511.50	89875.67	11.500000
59	77974.60	83110.65	11.500000
60	85865.45	128623.00	11.450000
61	80712.97	112126.50	11.095238
62	77742.45	124957.33	10.666667

63	78968.61	87783.58	11.000000
64	78681.99	297225.00	11.333333
65	96299.16	121734.00	11.000000
66	91915.21	155779.00	11.095238
67	84754.25	95132.00	11.500000
68	104513.34	113036.55	11.500000
69	94570.06	101196.32	11.500000
70	77658.46	103205.00	11.428571
71	88725.19	96227.30	10.500000
72	98102.43	109867.00	10.500000
73	94009.32	103248.75	10.684211
74	84869.69	91034.11	10.500000
75	79345.91	110344.91	9.500000
76	78790.04	91456.33	9.500000
77	93247.81	98842.18	10.894737
78	76586.39	127400.75	10.500000
79	73742.21	76674.49	14.333333
80	78317.61	95295.83	10.500000
81	84493.13	93961.90	10.500000
82	76896.60	94367.55	10.500000
83	83507.89	100256.64	10.333333
84	87490.59	140801.50	10.500000
85	84314.32	90917.84	10.500000
86	120069.70	134696.54	10.500000
87	75501.04	119571.27	9.944444
88	100796.45	133570.44	10.105263
89	106108.64	115356.65	10.500000
90	87168.22	118965.51	10.500000
91	97633.75	105214.29	10.500000
92	93243.42	103848.63	10.500000
93	83543.06	89915.36	10.500000

94	79588.50	86570.88	10.500000
95	82423.39	101763.82	10.315789
96	92585.02	98785.26	10.555556
97	75782.90	94429.50	10.894737
98	82601.56	112059.94	9.187500
99	77401.90	82938.19	10.500000
100	82924.95	91293.27	10.500000
101	76285.58	82405.73	10.500000
102	81801.87	88682.91	10.500000
103	119643.24	139481.23	9.500000
104	85253.74	91320.79	10.368421
105	95578.93	102668.58	10.500000
106	95603.49	103218.76	10.500000
107	74386.63	87118.00	10.500000
108	97967.66	112950.75	10.500000
109	98967.77	107286.27	10.222222
110	101439.78	140196.50	10.500000
111	85355.77	89866.52	10.500000
112	86568.20	107871.17	10.000000
113	73114.32	93105.17	9.722222
114	78705.26	93264.50	9.722222
115	78194.16	90767.86	10.500000
116	94727.03	100149.33	10.500000
117	77845.90	83275.58	10.500000
118	88625.90	93408.64	10.500000
119	79533.87	156356.83	10.823529
120	84878.00	90829.64	10.500000
121	84147.89	155483.50	10.736842
122	75298.43	84779.30	10.500000
123	96447.98	102632.33	10.500000
125	100944.86	109688.67	11.500000

140	98484.87	168522.25	10.105263
156	76203.29	98810.50	11.500000
158	97275.64	132970.00	11.500000
160	88706.06	94754.00	11.500000
161	88835.13	94819.75	11.500000
173	101762.92	108298.88	11.500000
175	94865.38	108964.52	11.315789
176	80052.57	93680.95	11.000000
177	89759.53	97820.16	11.700000
191	94930.97	114290.38	11.380952
192	80996.49	88997.67	11.666667
193	90502.43	96022.63	11.500000
199	99008.00	115316.91	11.500000
200	76677.27	117393.00	11.150000
201	107732.98	117167.50	11.000000
203	134863.80	163104.36	10.800000
207	93598.59	108323.00	12.222222
208	80097.08	86228.40	11.095238
209	93357.15	130711.40	11.950000
210	88075.37	103814.50	11.095238
211	116220.23	137648.00	11.285714
212	84935.88	89418.73	11.500000
213	86308.26	92549.79	11.105263
214	80910.83	86456.18	11.500000
215	89519.75	93840.88	11.650000
216	107978.80	196256.00	11.047619
224	96042.95	116481.00	10.315789
225	92858.82	108031.62	11.047619
226	94118.26	126478.39	11.500000
227	83678.70	96595.00	10.500000
228	90891.37	100744.96	11.000000

229	96608.39	125824.67	10.937500
230	110604.72	183566.00	9.294118
337	121640.73	154479.00	12.500000
338	96836.74	152461.75	11.850000
339	100507.36	123228.16	12.521739
340	114324.34	235357.88	11.750000
341	87031.83	96747.50	11.636364
342	84754.72	116397.00	12.565217
343	91855.52	98178.59	12.500000
344	80477.94	107208.00	12.500000
345	105781.95	115135.30	12.500000
346	97657.69	105505.00	12.500000
347	78490.51	196051.67	12.500000
348	86651.01	104020.00	12.500000
349	118380.73	149313.60	12.739130
350	86501.36	92565.64	12.818182
351	115960.31	135195.48	12.500000
352	102402.32	144370.50	12.000000
353	183651.09	315301.67	12.500000
354	78757.79	90721.44	12.500000
355	108669.76	114943.45	12.545454
841	92590.67	111482.39	11.500000
842	104639.86	113979.50	12.500000
843	103896.89	109135.35	12.500000
844	93406.17	102357.21	11.545454
845	90050.02	97551.33	12.000000
846	82576.28	105871.95	12.608696
847	116644.29	379323.61	12.500000
848	104845.07	112561.94	12.500000
849	102330.69	130091.00	12.500000
850	97505.57	105481.09	12.500000

851	91461.96	107424.88	12.500000
852	118291.08	127245.04	12.272727
853	91437.21	170977.00	12.380952
854	117159.95	127208.14	12.500000
855	102894.85	109184.36	12.500000
856	106945.35	113736.40	12.500000
857	90583.37	122778.50	12.260870
858	106034.29	174331.00	12.000000
859	77992.45	82941.10	12.500000
860	95217.00	103730.00	11.619048
861	133715.33	194763.57	12.500000
862	103338.02	110538.58	12.500000
863	100192.81	105286.75	12.500000
864	90138.56	96242.00	12.500000
865	81622.53	100399.60	12.050000
866	79279.80	83766.50	12.500000
867	109122.15	113433.39	12.000000
868	98294.00	148087.50	12.217391
869	81580.03	85620.44	12.500000
870	87905.84	93454.20	12.500000
871	121557.50	177324.50	13.600000
872	90211.72	93734.62	12.000000
873	117812.68	127353.00	12.500000
874	100683.81	106424.65	12.636364
875	105248.20	118108.00	12.045454
876	91179.07	97518.60	12.500000
877	115126.43	123225.00	12.565217
878	102772.66	108137.83	12.500000
879	85321.00	92127.26	12.500000
880	93159.05	97854.98	11.523810
881	106012.16	125239.00	11.500000

882	103338.30	109138.48	11.500000
883	101061.37	112151.06	11.500000
884	90251.45	99912.67	11.500000
885	84289.68	114196.40	11.500000
886	78987.76	85917.28	11.500000
887	106390.34	109371.65	11.500000
888	100358.33	103735.85	11.500000
890	87849.21	92672.76	11.500000
891	114140.82	121446.75	11.500000
892	88931.17	92163.98	11.285714
893	116760.57	123564.86	11.500000
894	110187.71	115457.86	11.500000
895	98288.70	113201.57	10.500000
896	91203.12	162390.00	11.500000
897	107020.11	111589.45	11.809524
898	106377.64	110662.69	11.238095
899	78257.75	84310.50	11.500000
900	97872.11	115760.02	11.450000
901	107606.68	128704.00	11.380952
902	103879.35	116078.82	11.500000
903	103800.18	120580.40	11.500000
904	91896.29	96934.68	11.500000
905	84200.78	120442.40	11.400000
906	84922.68	122897.86	10.800000
907	74295.44	79473.76	11.500000
908	169463.35	507859.22	10.750000
909	83924.09	88442.04	11.904762
910	96929.40	117124.64	11.500000
911	115376.27	129985.00	11.500000
912	89627.09	95338.40	11.500000
913	120079.92	123725.48	11.095238

914	152341.39	189621.50	11.500000
915	103976.30	115919.44	11.000000
916	107228.30	146788.00	9.470588
917	76092.32	78640.74	9.500000
918	108047.62	112030.22	9.500000
926	95070.12	99972.69	8.846154
927	108317.73	115536.62	10.000000
928	106260.98	110977.44	10.500000
929	100101.91	106067.06	10.000000
930	92008.41	98374.16	10.500000
931	82306.34	86970.53	10.500000
932	78759.21	83917.77	10.000000
933	76294.79	115883.38	9.833333
934	104514.84	140577.33	9.375000
936	90721.32	96531.91	10.500000
937	116753.19	169080.50	10.473684
938	88314.87	139892.00	10.500000
939	116952.33	125127.23	10.500000
940	99745.43	106009.42	10.500000
941	110019.32	127623.29	9.823529
942	118798.27	182097.00	10.500000
943	85567.44	118053.00	10.500000
944	77029.44	81995.88	10.421053
945	107457.73	114454.27	10.368421
948	96172.00	152368.76	11.190476
949	98503.44	105927.22	11.500000
950	105962.34	109043.07	11.500000
951	104943.34	109791.22	11.421053
952	92424.50	98555.10	12.500000
953	91911.96	134034.29	11.500000
954	78075.66	82832.69	11.000000

955	109262.08	122823.50	11.500000
956	72159.19	82316.50	11.500000
957	109535.21	169411.17	11.000000
958	86144.50	89774.34	11.500000
959	81256.72	84630.36	11.500000
960	121888.00	146593.88	10.750000
961	87699.79	255096.17	11.000000
962	113917.21	120694.26	11.500000
963	103314.30	113059.00	11.809524
964	98176.09	101473.92	11.500000
965	105225.32	143182.00	11.500000
966	84949.32	87842.36	10.750000
967	112979.58	183791.96	11.714286
968	106982.05	117461.60	11.500000
969	88625.82	98880.04	10.500000
970	104574.25	143879.33	10.526316
971	97499.09	101414.46	10.157895
972	101706.60	136684.00	9.352941
973	87219.65	160999.00	10.842105
974	79337.87	81436.82	10.500000
975	79787.91	90710.10	10.555556
976	110943.33	190867.50	10.000000
977	69134.13	96711.00	10.736842
978	95831.96	100933.62	10.277778
979	85524.47	95917.95	10.736842
980	111904.57	131154.50	10.500000
981	85515.32	90014.71	10.500000
982	127647.31	164392.38	11.941176
983	96451.61	100098.35	10.947368
984	99079.13	119639.75	10.263158
985	100553.41	114599.60	10.500000

986	81500.73	89441.60	10.000000
987	77271.30	82520.88	10.117647
988	102801.13	106383.41	10.500000
989	91369.92	96536.75	10.500000
990	96876.14	136290.67	10.500000
991	102435.36	104416.82	10.500000
992	115820.41	138904.50	10.611111
993	86817.76	96406.76	10.117647
994	79164.19	81677.87	10.500000
995	78108.49	82119.68	10.055556
996	102101.60	108266.74	10.277778
997	69239.77	72752.14	10.500000
998	95937.50	174472.00	10.500000
999	79092.00	84813.78	10.500000
1000	83377.53	86588.56	10.157895
1001	113965.36	149157.88	9.705882
1002	87065.74	102536.56	10.105263
1003	109370.67	114419.34	10.578947
1004	98965.68	125453.25	10.500000
1005	98812.49	126835.38	10.500000
1006	101047.20	139512.50	10.500000
1007	83160.48	87985.33	10.500000
1008	73648.82	79128.35	10.500000
1009	108734.22	141122.83	10.526316
1010	88402.16	94167.47	10.500000
1011	98564.26	103047.62	10.500000
1012	98684.82	104370.75	9.500000
1013	108096.90	113452.53	10.500000
1014	85034.32	89500.22	10.500000
1015	79595.35	90650.25	10.500000
1016	76367.74	80775.62	10.500000

1017	95682.98	100268.31	10.500000
1018	69321.44	72564.15	10.500000
1019	93624.08	112916.33	10.500000
1020	96379.81	112843.00	10.500000
1021	81482.80	85284.69	10.500000
1022	114220.68	194682.00	10.789474
1023	85408.77	89293.80	10.500000
1024	111061.82	118035.89	10.500000
1025	104974.23	110796.71	10.578947
1026	94360.67	99836.76	10.500000
1027	81815.55	86808.17	10.500000
1028	100636.66	106325.45	10.500000
1029	74513.31	80181.00	10.500000
1030	102649.36	106971.19	10.500000
1031	71762.67	77656.52	10.500000
1032	70009.62	104275.50	10.500000
1033	82463.90	91351.53	10.500000
1034	101563.13	112054.64	10.368421
1035	91961.40	95241.02	10.500000
1036	83413.32	87674.63	10.500000
1037	113728.00	116688.66	10.684211
1038	89167.76	149980.50	10.500000
1039	130779.71	142662.64	10.611111
1040	106421.96	110042.21	10.611111
1041	94655.98	98121.08	10.500000
1042	81770.12	85382.25	10.500000
1043	79672.96	85233.56	10.500000
1044	105850.22	121195.73	10.500000
1045	126096.75	753493.00	10.052632
1046	62932.34	63356.00	11.277778
1047	105248.09	109414.87	10.500000

1051	88920.54	92888.29	10.500000
1052	98641.02	103801.06	10.000000
1053	101964.83	121719.08	10.315789
1054	85930.62	93673.00	10.500000
1055	84661.82	88196.50	10.500000
1056	76113.08	79664.84	10.500000
1057	108112.48	157530.94	10.500000
1058	69562.32	123661.00	10.500000
1059	98976.79	102192.35	10.500000
1060	70909.06	73790.97	10.157895
1061	136601.62	152379.60	10.947368
1062	106902.84	631957.67	11.562500
1063	207071.00	245276.00	10.500000
1064	75074.93	79249.22	10.500000
1065	87689.40	106783.00	10.157895
1066	102660.40	107000.47	10.500000
1067	94208.67	98348.77	10.500000
1069	101367.00	109074.43	10.500000
1070	83367.41	87741.22	10.166667
1071	78068.32	82164.49	10.500000
1072	98543.00	194902.86	10.500000
1073	91524.18	95192.05	10.000000
1074	100781.02	103814.26	10.500000
1075	98876.14	104390.94	9.777778
1076	90802.55	99028.00	10.500000
1077	87745.81	124153.17	10.526316
1078	96139.54	102172.17	10.500000
1079	88492.05	92043.80	10.500000
1080	103646.08	120910.67	10.500000
1081	109805.75	114928.74	10.500000
1082	81366.00	83570.53	10.500000

1083	159044.44	256319.25	10.411765
1084	71160.46	75696.50	10.500000
1085	99114.24	104884.45	10.500000
1086	85370.17	88657.96	10.500000
1087	117111.23	125651.00	10.684211
1088	80008.98	82321.85	10.500000
1089	88488.53	92926.10	10.500000
1091	124326.07	139084.86	10.500000
1092	132285.86	135590.25	10.611111
1093	105768.94	115016.00	10.500000
1094	83334.21	86545.68	10.500000
1095	82711.64	84947.79	10.833333
1096	90791.62	94472.51	10.500000
1098	98890.11	103275.77	10.500000
1099	97324.00	104664.67	10.500000
1100	102973.50	158630.00	10.684211
1101	109067.37	112844.61	10.500000
1102	92249.84	94511.23	10.500000
1104	80782.47	87954.27	10.500000
1105	79968.79	81994.11	10.500000
1106	80547.83	82801.67	10.500000
1107	72304.32	82047.92	10.500000
1108	94706.61	101208.46	10.500000
1109	84123.34	124981.00	10.500000
1110	112510.23	117123.70	10.736842
1111	84215.16	120835.62	10.500000
1112	86689.08	89101.97	10.473684
1113	103184.16	105515.96	9.722222
1114	102989.08	292526.20	10.500000
1115	92266.11	96705.15	10.833333
1116	102167.18	108074.83	10.500000

1117	86645.58	115299.85	10.500000
1118	98716.82	152290.14	10.529412
1119	106965.78	124844.50	10.500000
1120	90045.24	91885.96	10.500000
1121	96574.42	100464.31	10.500000
1122	96865.46	121453.00	10.500000
1123	83221.43	85628.46	10.000000
1124	129501.25	247418.50	10.277778
1125	104132.63	115062.70	10.500000
1126	95611.86	96946.79	10.500000
1127	81353.21	86033.90	10.500000
1128	110199.41	113826.50	9.625000
1129	90684.67	96082.70	10.500000
1130	80306.47	82892.08	10.500000
1131	71307.01	73465.32	10.500000

grouped summary

```
group_summary = pit_stops_summary |>
  group_by(raceId) |>
  summarize(
    avg_pit_duration = mean(avg_pit_duration, na.rm = T
    avg_finishing_position = mean(finishing_position, n
    total_drivers = n()
  )

head(group_summary)
```

```
# A tibble: 6 × 4
  raceId avg_pit_duration avg_finishing_position total_drivers
   <int>            <dbl>                  <dbl>         <int>
1      1           97341.                   10.3            19
2      2          110425.                   10.3            19
3      3          129788.                   10.5            20
4      4           97848.                   10.5            20
5      5           92489.                    9.5            16
6      6           78933.                   10.5            20
```

```
kable(group_summary, caption = "Summary of Pit Stop Dur
```

Summary of Pit Stop Duration and Average Finishing Position

raceId	avg_pit_duration	avg_finishing_position	total_drivers
1	97340.52	10.315789	19
2	110425.19	10.315789	19
3	129788.08	10.500000	20
4	97847.92	10.500000	20
5	92489.36	9.500000	16
6	78932.54	10.500000	20
7	90819.26	10.500000	20
8	84433.73	10.500000	20
9	97917.99	10.500000	20
10	86682.02	10.526316	19
11	102239.65	10.500000	20
12	115932.31	9.250000	16
13	88203.32	10.000000	19
14	116362.29	10.500000	20
15	100263.20	10.315789	19
16	79695.60	11.352941	17
17	103740.83	10.500000	20
18	98677.16	10.352941	17
19	103076.63	11.142857	21
20	97836.24	11.142857	21
21	93624.41	10.750000	20
22	94461.87	10.052632	19
23	97747.14	10.500000	20
24	83906.85	10.500000	20
25	79942.54	10.500000	20
26	100830.78	10.611111	18
27	82423.32	10.500000	20

28	85028.14	10.500000	20
29	102155.32	10.421053	19
30	114585.28	10.500000	20
31	100055.80	10.500000	20
32	115670.38	10.500000	20
33	82354.89	10.473684	19
34	102614.26	10.500000	20
35	80933.92	10.277778	18
36	90572.04	11.500000	22
37	100943.19	11.000000	21
38	100918.07	10.900000	20
39	89187.49	11.500000	22
40	79924.59	11.500000	22
41	88437.19	11.333333	21
42	76311.60	11.315789	19
43	83323.15	11.238095	21
44	86131.31	11.500000	22
45	145020.12	11.500000	22
46	84012.50	11.500000	22
47	91566.12	11.500000	22
48	91877.47	11.500000	22
49	114768.17	11.500000	22
50	112135.97	11.500000	22
51	108004.28	11.500000	22
52	79432.25	11.500000	22
53	97748.24	11.190476	21
54	101244.45	11.714286	21
55	103205.14	11.157895	19
56	92351.62	11.095238	21
57	99805.33	11.285714	21
58	81123.00	11.500000	22

59	81038.50	11.500000	22
60	90552.03	11.450000	20
61	84909.10	11.095238	21
62	86825.39	10.666667	15
63	81045.61	11.000000	21
64	90775.46	11.333333	21
65	100789.53	11.000000	21
66	98392.59	11.095238	21
67	86858.24	11.500000	22
68	107571.02	11.500000	22
69	97156.49	11.500000	22
70	82851.94	11.428571	21
71	90553.79	10.500000	20
72	101152.62	10.500000	20
73	97004.78	10.684211	19
74	86866.41	10.500000	20
75	84181.20	9.500000	18
76	82980.87	9.500000	18
77	95054.51	10.894737	19
78	82206.38	10.500000	20
79	75135.38	14.333333	6
80	81934.27	10.500000	20
81	86993.15	10.500000	20
82	80001.72	10.500000	20
83	86759.65	10.333333	18
84	93090.80	10.500000	20
85	86161.99	10.500000	20
86	126689.36	10.500000	20
87	79512.67	9.944444	18
88	105881.37	10.105263	19
89	108739.83	10.500000	20

90	91474.78	10.500000	20
91	100149.90	10.500000	20
92	95471.34	10.500000	20
93	85527.42	10.500000	20
94	81725.22	10.500000	20
95	87454.31	10.315789	19
96	94613.19	10.555556	18
97	78633.80	10.894737	19
98	89561.89	9.187500	16
99	79004.41	10.500000	20
100	85803.97	10.500000	20
101	77877.90	10.500000	20
102	84468.99	10.500000	20
103	127097.99	9.500000	16
104	86885.21	10.368421	19
105	97683.88	10.500000	20
106	97936.11	10.500000	20
107	76924.77	10.500000	20
108	101370.00	10.500000	20
109	102013.91	10.222222	18
110	108271.44	10.500000	20
111	87144.48	10.500000	20
112	90828.45	10.000000	17
113	75747.35	9.722222	18
114	81800.23	9.722222	18
115	80848.63	10.500000	20
116	96884.55	10.500000	20
117	79685.56	10.500000	20
118	90179.61	10.500000	20
119	92003.54	10.823529	17
120	87093.47	10.500000	20

121	90349.41	10.736842	19
122	79471.29	10.500000	20
123	98522.30	10.500000	20
125	104862.07	11.500000	22
140	104316.93	10.105263	19
156	78695.91	11.500000	22
158	102540.42	11.500000	22
160	91527.53	11.500000	22
161	90631.05	11.500000	22
173	104640.34	11.500000	22
175	102373.00	11.315789	19
176	83720.00	11.000000	21
177	93411.02	11.700000	20
191	101520.64	11.380952	21
192	83592.67	11.666667	21
193	93127.67	11.500000	22
199	107202.12	11.500000	22
200	82985.89	11.150000	20
201	109882.87	11.000000	21
203	144091.88	10.800000	15
207	97581.21	12.222222	18
208	81603.13	11.095238	21
209	98102.94	11.950000	20
210	91653.80	11.095238	21
211	122516.47	11.285714	21
212	86522.37	11.500000	22
213	87604.64	11.105263	19
214	83771.29	11.500000	22
215	91332.76	11.650000	20
216	119546.13	11.047619	21
224	100124.39	10.315789	19

225	98310.24	11.047619	21
226	98984.99	11.500000	22
227	86164.03	10.500000	20
228	93409.04	11.000000	21
229	105453.93	10.937500	16
230	120776.10	9.294118	17
337	126284.31	12.500000	24
338	107193.10	11.850000	20
339	105052.13	12.521739	23
340	124383.93	11.750000	20
341	89731.18	11.636364	22
342	87878.02	12.565217	23
343	94023.78	12.500000	24
344	84084.52	12.500000	24
345	107566.33	12.500000	24
346	99641.78	12.500000	24
347	86664.97	12.500000	24
348	90082.38	12.500000	24
349	123880.19	12.739130	23
350	88492.68	12.818182	22
351	119793.47	12.500000	24
352	106850.38	12.000000	19
353	210184.59	12.500000	24
354	80925.49	12.500000	24
355	110574.88	12.545454	22
841	96150.35	11.500000	22
842	107474.42	12.500000	24
843	105752.73	12.500000	24
844	95474.07	11.545454	22
845	92928.00	12.000000	23
846	97836.04	12.608696	23

847	223306.78	12.500000	24
848	107533.09	12.500000	24
849	106661.27	12.500000	24
850	100438.22	12.500000	24
851	95528.92	12.500000	24
852	120716.59	12.272727	22
853	101054.70	12.380952	21
854	120735.60	12.500000	24
855	104487.12	12.500000	24
856	109048.95	12.500000	24
857	94630.71	12.260870	23
858	111688.76	12.000000	23
859	80135.87	12.500000	24
860	98625.41	11.619048	21
861	177738.23	12.500000	24
862	104830.76	12.500000	24
863	102132.70	12.500000	24
864	92139.73	12.500000	24
865	83682.06	12.050000	20
866	80518.23	12.500000	24
867	110833.59	12.000000	23
868	101882.12	12.217391	23
869	83062.95	12.500000	24
870	89478.44	12.500000	24
871	126537.34	13.600000	20
872	91578.54	12.000000	23
873	123420.61	12.500000	24
874	102758.67	12.636364	22
875	107929.69	12.045454	22
876	93145.58	12.500000	24
877	116548.06	12.565217	23

878	104967.75	12.500000	24
879	89700.06	12.500000	22
880	95050.77	11.523810	21
881	109265.42	11.500000	22
882	105105.49	11.500000	22
883	102849.30	11.500000	22
884	92411.99	11.500000	22
885	104579.28	11.500000	22
886	81078.91	11.500000	22
887	107817.51	11.500000	22
888	102134.50	11.500000	22
890	89692.27	11.500000	22
891	116463.02	11.500000	22
892	89985.52	11.285714	21
893	118829.32	11.500000	22
894	113346.37	11.500000	22
895	100583.34	10.500000	20
896	95984.60	11.500000	22
897	108950.14	11.809524	21
898	107886.06	11.238095	21
899	79799.58	11.500000	22
900	101485.35	11.450000	20
901	111329.81	11.380952	21
902	106240.42	11.500000	22
903	106574.52	11.500000	22
904	94007.08	11.500000	22
905	88188.38	11.400000	20
906	88615.53	10.800000	20
907	75727.86	11.500000	22
908	205409.69	10.750000	20
909	85658.02	11.904762	21

910	101467.42	11.500000	22
911	118454.53	11.500000	22
912	91447.61	11.500000	22
913	121156.32	11.095238	21
914	157366.65	11.500000	22
915	106313.11	11.000000	21
916	111391.08	9.470588	17
917	77189.10	9.500000	18
918	109923.32	9.500000	18
926	97192.65	8.846154	13
927	110925.45	10.000000	19
928	107960.11	10.500000	20
929	102243.06	10.000000	19
930	94460.62	10.500000	20
931	84500.60	10.500000	20
932	80399.11	10.000000	19
933	82074.77	9.833333	18
934	109958.47	9.375000	16
936	93501.49	10.500000	20
937	121871.90	10.473684	19
938	93056.70	10.500000	20
939	121077.05	10.500000	20
940	102019.69	10.500000	20
941	112768.07	9.823529	17
942	125854.27	10.500000	20
943	88801.30	10.500000	20
944	78804.62	10.421053	19
945	109710.94	10.368421	19
948	115039.07	11.190476	21
949	100796.93	11.500000	20
950	107726.41	11.500000	22

951	107242.63	11.421053	19
952	94182.27	12.500000	20
953	100359.37	11.500000	22
954	79912.43	11.000000	21
955	111880.36	11.500000	22
956	75205.49	11.500000	22
957	116629.78	11.000000	21
958	87865.52	11.500000	22
959	82858.03	11.500000	22
960	142078.54	10.750000	20
961	97778.61	11.000000	21
962	116176.57	11.500000	20
963	106396.04	11.809524	21
964	99938.96	11.500000	22
965	109138.61	11.500000	22
966	86297.43	10.750000	20
967	149986.01	11.714286	21
968	109247.22	11.500000	22
969	91696.86	10.500000	20
970	110492.27	10.526316	19
971	99947.54	10.157895	19
972	105563.16	9.352941	17
973	93298.66	10.842105	19
974	80681.44	10.500000	20
975	81622.84	10.555556	18
976	148709.26	10.000000	19
977	71897.59	10.736842	19
978	97851.82	10.277778	18
979	87430.77	10.736842	19
980	116827.63	10.500000	20
981	87506.08	10.500000	20

982	133379.11	11.941176	17
983	98519.30	10.947368	19
984	102096.62	10.263158	19
985	103198.83	10.500000	20
986	83586.37	10.000000	19
987	78650.47	10.117647	17
988	104883.17	10.500000	20
989	93377.60	10.500000	20
990	100452.32	10.500000	20
991	103183.51	10.500000	20
992	122300.44	10.611111	18
993	89660.70	10.117647	17
994	79856.44	10.500000	20
995	79605.02	10.055556	18
996	104113.93	10.277778	18
997	70763.54	10.500000	20
998	104521.76	10.500000	20
999	83076.94	10.500000	20
1000	85098.50	10.157895	19
1001	118696.90	9.705882	17
1002	89337.35	10.105263	19
1003	111655.65	10.578947	19
1004	103034.90	10.500000	20
1005	102109.21	10.500000	20
1006	106239.18	10.500000	20
1007	85796.49	10.500000	20
1008	75105.13	10.500000	20
1009	112649.15	10.526316	19
1010	90628.44	10.500000	20
1011	100449.27	10.500000	20

1012	101044.46	9.500000	18
1013	110393.86	10.500000	20
1014	87476.51	10.500000	20
1015	80990.77	10.500000	20
1016	78165.59	10.500000	20
1017	97791.85	10.500000	20
1018	70595.59	10.500000	20
1019	96183.69	10.500000	20
1020	99340.87	10.500000	20
1021	83561.32	10.500000	20
1022	120458.20	10.789474	19
1023	87167.76	10.500000	20
1024	116666.93	10.500000	20
1025	107190.13	10.578947	19
1026	96719.50	10.500000	20
1027	83452.49	10.500000	20
1028	102800.45	10.500000	20
1029	78507.91	10.500000	20
1030	105038.75	10.500000	20
1031	75995.19	10.500000	20
1032	74483.03	10.500000	20
1033	84510.49	10.500000	20
1034	102924.82	10.368421	19
1035	93405.58	10.500000	20
1036	85359.27	10.500000	20
1037	115670.57	10.684211	19
1038	118808.01	10.500000	20
1039	139672.19	10.611111	18
1040	108136.43	10.611111	18
1041	96322.01	10.500000	20
1042	83542.52	10.500000	20

1043	83908.29	10.500000	20
1044	108879.42	10.500000	20
1045	160241.48	10.052632	19
1046	63148.47	11.277778	18
1047	107031.49	10.500000	20
1051	90777.28	10.500000	20
1052	100418.07	10.000000	19
1053	116386.40	10.315789	19
1054	87637.69	10.500000	20
1055	86418.24	10.500000	20
1056	77322.34	10.500000	18
1057	148852.89	10.500000	20
1058	73880.14	10.500000	20
1059	100622.57	10.500000	20
1060	72044.99	10.157895	19
1061	139115.98	10.947368	19
1062	144191.08	11.562500	16
1063	225823.95	10.500000	20
1064	77188.83	10.500000	20
1065	93405.67	10.157895	19
1066	104672.05	10.500000	20
1067	95853.13	10.500000	20
1069	103987.11	10.500000	20
1070	85141.30	10.166667	18
1071	79687.34	10.500000	20
1072	156839.45	10.500000	20
1073	93902.11	10.000000	19
1074	103109.24	10.500000	20
1075	102333.92	9.777778	18
1076	92603.47	10.500000	20
1077	90854.86	10.526316	19

1078	99946.91	10.500000	20
1079	90123.96	10.500000	20
1080	109976.76	10.500000	20
1081	112689.55	10.500000	20
1082	82890.26	10.500000	20
1083	169993.16	10.411765	17
1084	72573.58	10.500000	20
1085	102951.97	10.500000	20
1086	86666.90	10.500000	20
1087	119314.01	10.684211	19
1088	80962.14	10.500000	20
1089	91069.71	10.500000	20
1091	127397.76	10.500000	20
1092	134424.15	10.611111	18
1093	110580.96	10.500000	20
1094	84899.27	10.500000	20
1095	83588.70	10.833333	18
1096	92237.34	10.500000	20
1098	100583.51	10.500000	20
1099	98861.04	10.500000	20
1100	145571.68	10.684211	19
1101	110544.16	10.500000	20
1102	93329.40	10.500000	20
1104	85075.63	10.500000	20
1105	81107.17	10.500000	20
1106	81532.44	10.500000	20
1107	73660.31	10.500000	20
1108	98492.09	10.500000	20
1109	88345.42	10.500000	20
1110	114321.43	10.736842	19

1111	115494.67	10.500000	20
1112	87785.44	10.473684	19
1113	104044.53	9.722222	18
1114	117267.41	10.500000	20
1115	93552.82	10.833333	18
1116	103581.29	10.500000	20
1117	103993.22	10.500000	20
1118	104429.71	10.529412	17
1119	108583.75	10.500000	20
1120	91002.99	10.500000	20
1121	98133.45	10.500000	20
1122	99314.08	10.500000	20
1123	84588.51	10.000000	19
1124	137689.59	10.277778	18
1125	109150.87	10.500000	20
1126	96196.95	10.500000	20
1127	82788.34	10.500000	20
1128	111800.66	9.625000	16
1129	91675.67	10.500000	20
1130	81480.63	10.500000	20
1131	72187.37	10.500000	20

Next I wanted to explore how much do individual drivers versus constructors (teams) contribute to overall race outcomes.

```
head(constructors)
```

```
constructorId constructorRef       name nationality
```

```
1             1        mclaren    McLaren     British
2             2     bmw_sauber BMW Sauber      German
3             3       williams   Williams     British
4             4        renault    Renault      French
5             5     toro_rosso Toro Rosso     Italian
6             6        ferrari    Ferrari     Italian
                                                    url
1                        http://en.wikipedia.org/wiki/McLaren
2                      http://en.wikipedia.org/wiki/BMW_Sauber
3 http://en.wikipedia.org/wiki/Williams_Grand_Prix_Engineering
4          http://en.wikipedia.org/wiki/Renault_in_Formula_One
5             http://en.wikipedia.org/wiki/Scuderia_Toro_Rosso
6                http://en.wikipedia.org/wiki/Scuderia_Ferrari
```

```
head(driver_standings)
```

```
  driverStandingsId raceId driverId points position
positionText wins
1                 1     18        1     10        1
1    1
2                 2     18        2      8        2
2    0
3                 3     18        3      6        3
3    0
4                 4     18        4      5        4
4    0
5                 5     18        5      4        5
5    0
6                 6     18        6      3        6
6    0
```

```
driver_data = merge(results, driver_standings, by = c("
head(driver_data)
```

```
  raceId driverId resultId constructorId number grid position.x
positionText.x
1      1       10     7557             7     10   19          4
4
2      1       15     7556             7      9   20          3
3
3      1       16     7562            10     20   16          9
9
4      1       17     7565             9     14    8         12
```

```
12
5      1       18     7554             23     22    1          1
1
6      1        2     7563              2      6    9         10
10
  positionOrder points.x laps        time milliseconds
fastestLap rank
1             4        5   58      +4.435      5660219
53    6
2             3        6   58      +1.604      5657388
50   10
3             9        0   58      +6.335      5662119
43   11
4            12        0   57         \\N          \\N
38    8
5             1       10   58 1:34:15.784      5655784
17    3
6            10        0   58      +7.085      5662869
48    5
  fastestLapTime fastestLapSpeed statusId driverStandingsId
points.y position.y
1       1:28.416         215.920        1              8250
5          4
2       1:28.916         214.706        1              8249
6          3
3       1:28.943         214.640        1              8255
0          9
4       1:28.508         215.695       11              8258
0         12
5       1:28.020         216.891        1              8247
10         1
6       1:28.283         216.245        1              8256
0         10
  positionText.y wins
1              4    0
2              3    0
3              9    0
4             12    0
5              1    1
6             10    0
```

```
merged_construct = merge(driver_data, constructors, by
head(merged_construct)
```

```
  constructorId raceId driverId resultId number grid position.x
positionText.x
1             1    533      175    12978      8   14         \\N
R
2             1    244       87     4829      7   10          5
5
3             1    352       18    20686      1    5          4
4
4             1    302      102     6397      1    3          3
3
5             1    640      262    15816     14   13          7
7
6             1   1062      846    25180      4    6         \\N
R
  positionOrder points.x laps     time milliseconds fastestLap
rank
1            24        0   16       \\N          \\N        \\N
\\N
2             5        2   77       \\N          \\N        \\N
\\N
3             4       12   53  +13.522      5440845         53
2
4             3        4   70       \\N          \\N        \\N
\\N
5             7        0   55 +1:24.83      4777430        \\N
\\N
6            15        0    2       \\N          \\N        \\N
0
  fastestLapTime fastestLapSpeed statusId driverStandingsId
points.y position.y
1             \\N             \\N       20             57338
1         16
2             \\N             \\N       11             10269
3          9
3       1:33.529         223.515        1             64189
189          5
4             \\N             \\N       11              7154
50          2
5             \\N             \\N        1             53429
0         22
6             \\N             \\N        4             70514
113          3
  positionText.y wins constructorRef    name nationality
1             16    0        mclaren McLaren     British
2              9    0        mclaren McLaren     British
```

```
3              5    2        mclaren McLaren     British
4              2    3        mclaren McLaren     British
5             22    0        mclaren McLaren     British
6              3    0        mclaren McLaren     British
                                  url
1 http://en.wikipedia.org/wiki/McLaren
2 http://en.wikipedia.org/wiki/McLaren
3 http://en.wikipedia.org/wiki/McLaren
4 http://en.wikipedia.org/wiki/McLaren
5 http://en.wikipedia.org/wiki/McLaren
6 http://en.wikipedia.org/wiki/McLaren
```

Summarizing data by constructorId to calculate total points scored by drivers and constructors.

```
constructor_summary = driver_data |>
  group_by(constructorId) |>
  summarize(
    total_constructor_points = sum(points.x, na.rm = TR
    avg_constructor_points= mean(points.x, na.rm= TRUE)
    total_races = n()
  )
head(constructor_summary)
```

```
# A tibble: 6 × 4
  constructorId total_constructor_points avg_constructor_points
total_races
          <int>                    <dbl>                  <dbl>
<int>
1             1                    6688.                   3.56
1876
2             2                     308                    2.26
136
3             3                    3628                    2.22
1637
4             4                    1777                    2.28
780
5             5                     500                    0.942
531
6             6                   10772.                   4.49
2399
```

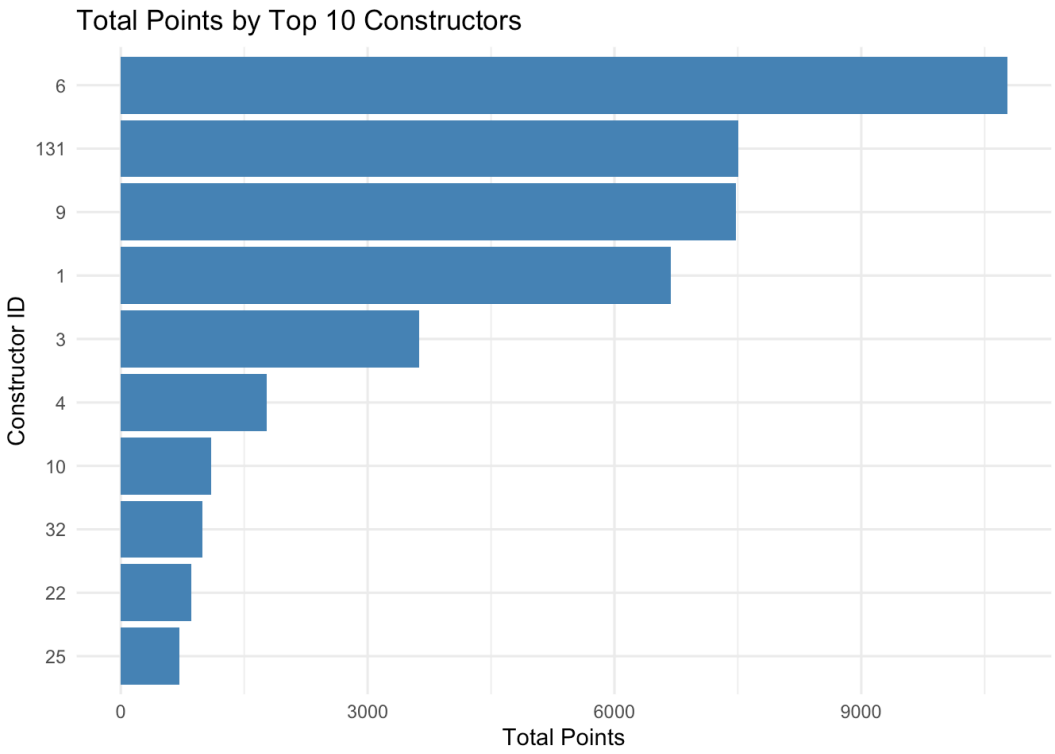
summarizing data by driverId to find the average race performance per driver and constructor.

```
driver_summary = driver_data |>
  group_by(driverId) |>
  summarize(
    total_driver_points = sum(points.x, na.rm = TRUE),
    avg_driver_points = mean(points.x, na.rm = TRUE),
    total_races = n()
  )

head(driver_summary)
```

```
# A tibble: 6 × 4
  driverId total_driver_points avg_driver_points total_races
     <int>               <dbl>             <dbl>       <int>
1        1               4714.             13.7          343
2        2                259               1.44         180
3        3               1594.              7.74         206
4        4               2304               5.88         392
5        5                105               0.946        111
6        6                  9               0.257         35
```

Plotting constructor points

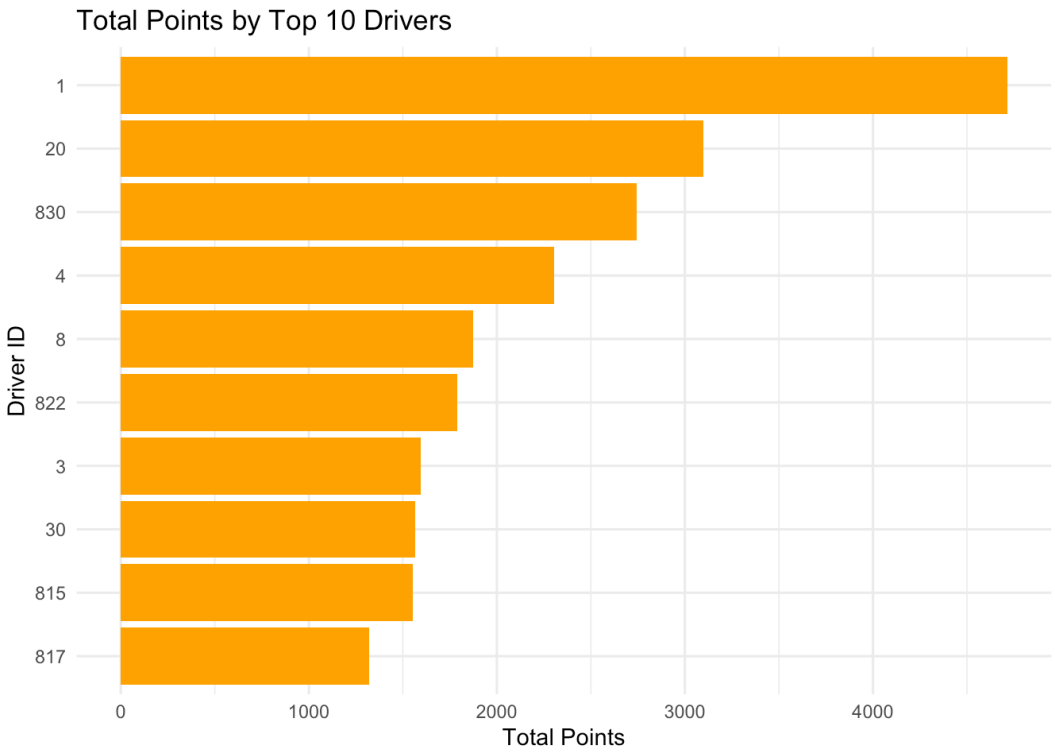
```
top_constructors = constructor_summary |>
  arrange(desc(total_constructor_points)) |>
  slice_head(n = 10)


ggplot(top_constructors, aes(x = reorder(factor(constru
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(
    title = "Total Points by Top 10 Constructors",
    x = "Constructor ID",
    y = "Total Points"
  ) +
  coord_flip() +
  theme_minimal()
```



Plotting driver points

```
top_drivers = driver_summary |>
  arrange(desc(total_driver_points)) |>
  slice_head(n = 10)
```

```
ggplot(top_drivers, aes(x = reorder(factor(driverId), t
  geom_bar(stat = "identity", fill = "orange") +
  labs(
    title = "Total Points by Top 10 Drivers",
    x = "Driver ID",
    y = "Total Points"
  ) +
  theme_minimal() +
  coord_flip()
```




```
drivers = read.csv(here::here("drivers.csv"))
head(drivers)
```

```
  driverId  driverRef number code forename    surname
dob nationality
1        1   hamilton     44  HAM    Lewis   Hamilton 1985-01-
07     British
2        2   heidfeld    \\N  HEI     Nick   Heidfeld 1977-05-
10      German
3        3    rosberg      6  ROS     Nico    Rosberg 1985-06-
27      German
4        4     alonso     14  ALO Fernando     Alonso 1981-07-
29     Spanish
5        5 kovalainen    \\N  KOV   Heikki Kovalainen 1981-10-
19     Finnish
6        6   nakajima    \\N  NAK   Kazuki   Nakajima 1985-01-
11    Japanese
                                              url
1    http://en.wikipedia.org/wiki/Lewis_Hamilton
2     http://en.wikipedia.org/wiki/Nick_Heidfeld
3      http://en.wikipedia.org/wiki/Nico_Rosberg
4   http://en.wikipedia.org/wiki/Fernando_Alonso
5 http://en.wikipedia.org/wiki/Heikki_Kovalainen
```

```
6   http://en.wikipedia.org/wiki/Kazuki_Nakajima
```

```
constructor_summary_with_names = constructor_summary |>
  left_join(constructors, by = "constructorId") |>
  mutate(constructor_name = name)
head(constructor_summary_with_names)
```

```
# A tibble: 6 × 9
  constructorId total_constructor_points avg_constructor_points
total_races
          <int>                    <dbl>                  <dbl>
<int>
1             1                    6688.                   3.56
1876
2             2                     308                    2.26
136
3             3                    3628                    2.22
1637
4             4                    1777                    2.28
780
5             5                     500                    0.942
531
6             6                   10772.                   4.49
2399
# ℹ 5 more variables: constructorRef <chr>, name <chr>,
nationality <chr>,
#   url <chr>, constructor_name <chr>
```

Notes:

filtering only the top 10 constructors, for an easier read.

- i wanted to use the actual constructor name instead of the constructor ID to make better sense of my data.

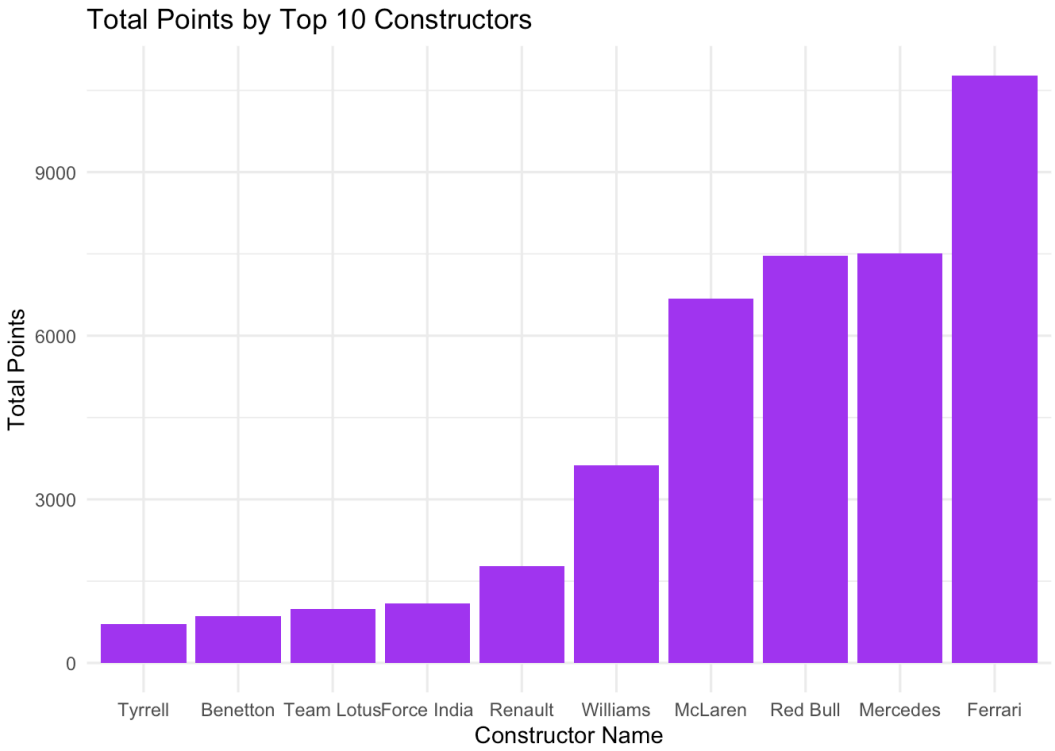
```
top_constructors = constructor_summary_with_names |>
  arrange(desc(total_constructor_points)) |>
  slice_head(n = 10)
```

Notes:

- Coord_flip() flips the x and y axis, turns a vertical graph/chart horizontally. Its useful when there's long labels like the constructors names.

- stat = identity, tells R to use the actual values from the data instead of counting.
- theme_minimal(), uses basic formatting and makes the charts easier to read.

```
ggplot(top_constructors, aes(x = reorder(constructor_na
  geom_bar(stat = "identity", fill = "purple") +
  labs(
    title = "Total Points by Top 10 Constructors",
    x = "Constructor Name",
    y = "Total Points"
  ) +
  theme_minimal()
```



Total of races by these top 10 constructors.

```
total_constrcutor_races = top_constructors |>
  group_by(constructor_name) |>
  summarize(
    total_races =sum(total_races, na.rm = TRUE)
  )
head(total_constrcutor_races)
```

```
# A tibble: 6 × 2
  constructor_name total_races
  <chr>                  <int>
1 Benetton                 505
2 Ferrari                 2399
3 Force India              421
4 McLaren                 1876
5 Mercedes                 628
6 Red Bull                 759
```

I wanted to merge the data containing the driver names and the data containing total driver points and average driver points

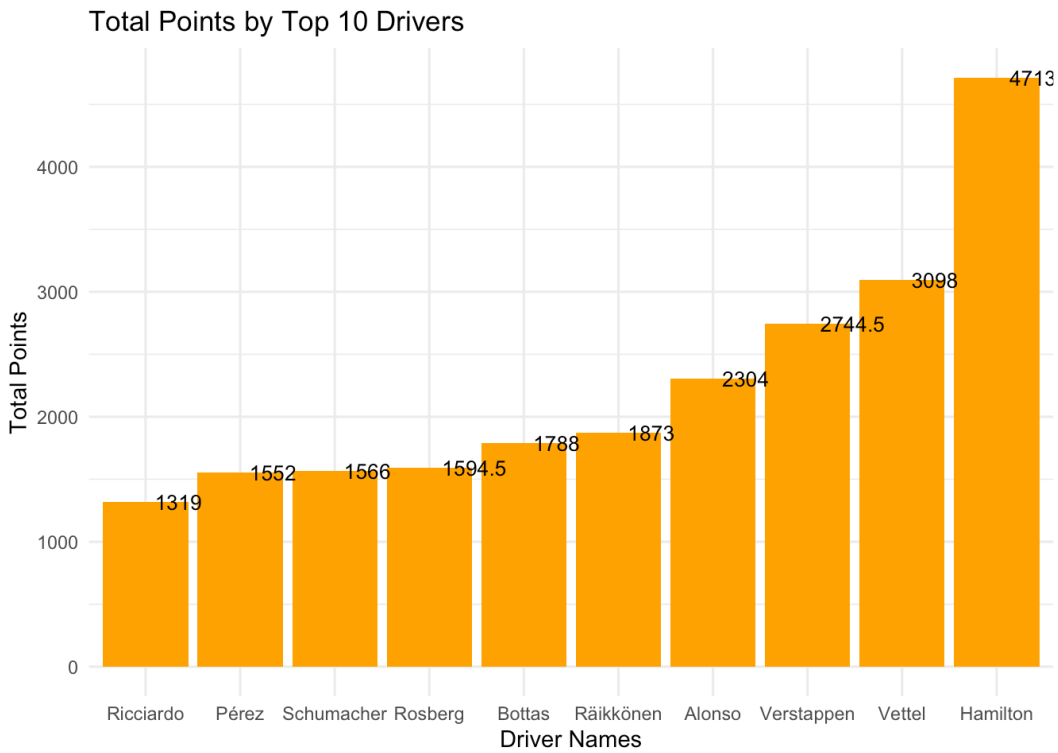
```
        top_drivers = merge(drivers, driver_summary, by = "driv
        head(top_drivers)
```

```
  driverId  driverRef number code forename    surname
dob nationality
1        1   hamilton     44  HAM    Lewis   Hamilton 1985-01-
07     British
2        2   heidfeld    \\N  HEI     Nick   Heidfeld 1977-05-
10      German
3        3    rosberg      6  ROS     Nico    Rosberg 1985-06-
27      German
4        4     alonso     14  ALO Fernando     Alonso 1981-07-
29     Spanish
5        5 kovalainen    \\N  KOV   Heikki Kovalainen 1981-10-
19     Finnish
6        6   nakajima    \\N  NAK   Kazuki   Nakajima 1985-01-
11    Japanese
                                          url
total_driver_points
1    http://en.wikipedia.org/wiki/Lewis_Hamilton
4713.5
2     http://en.wikipedia.org/wiki/Nick_Heidfeld
259.0
3      http://en.wikipedia.org/wiki/Nico_Rosberg
1594.5
4   http://en.wikipedia.org/wiki/Fernando_Alonso
2304.0
5 http://en.wikipedia.org/wiki/Heikki_Kovalainen
105.0
6   http://en.wikipedia.org/wiki/Kazuki_Nakajima
9.0
```

```
  avg_driver_points total_races
1        13.7419825         343
2         1.4388889         180
3         7.7402913         206
4         5.8775510         392
5         0.9459459         111
6         0.2571429          35
```

```
top_drivers = top_drivers |>
  arrange(desc(total_driver_points)) |>
slice_head(n=10)
```

```
  ggplot(top_drivers, aes(x = reorder(surname, total_dr
  geom_bar(stat = "identity", fill = "orange") +
    geom_text(aes(label = total_driver_points), hjust =
  labs(
    title = "Total Points by Top 10 Drivers",
    x = "Driver Names",
    y = "Total Points"
  ) +

  theme_minimal()
```



```
        top_drivers = top_drivers|>
          arrange(desc(total_driver_points))|>
          select(surname, total_driver_points )



        head(top_drivers)
```

```
     surname total_driver_points
1   Hamilton              4713.5
2     Vettel              3098.0
3 Verstappen              2744.5
4     Alonso              2304.0
5  Räikkönen              1873.0
6     Bottas              1788.0
```

The 2025 Formula 1 season will see Lewis Hamilton, one of the most sucesful drivers in f1 history joining the Ferrari, making what will be the most successful driver constructor pairing. Below I will look to make an informed prediction for Ferrari's success with Lewis Hamilton. While this won't be a perfect prediction due to a lack of data, i will use the historical data trends and averages to create an estimate.

For Lewis Hamilton

- avg points per season
- points contribution relative to his team. Mercedes and McLaren

```
        seasons = read.csv(here::here("seasons.csv"))
        head(seasons)
```

```
  year                                                  url
1 2009 http://en.wikipedia.org/wiki/2009_Formula_One_season
2 2008 http://en.wikipedia.org/wiki/2008_Formula_One_season
3 2007 http://en.wikipedia.org/wiki/2007_Formula_One_season
4 2006 http://en.wikipedia.org/wiki/2006_Formula_One_season
5 2005 http://en.wikipedia.org/wiki/2005_Formula_One_season
6 2004 http://en.wikipedia.org/wiki/2004_Formula_One_season
```

```
        races = read.csv(here::here("races.csv"))
        head(races)
```

```
  raceId year round circuitId                  name       date
time
1      1 2009     1         1 Australian Grand Prix 2009-03-29
06:00:00
2      2 2009     2         2  Malaysian Grand Prix 2009-04-05
09:00:00
3      3 2009     3        17    Chinese Grand Prix 2009-04-19
07:00:00
4      4 2009     4         3    Bahrain Grand Prix 2009-04-26
12:00:00
5      5 2009     5         4    Spanish Grand Prix 2009-05-10
12:00:00
6      6 2009     6         6     Monaco Grand Prix 2009-05-24
12:00:00
                                                      url
fp1_date fp1_time
1 http://en.wikipedia.org/wiki/2009_Australian_Grand_Prix
\\N      \\N
2  http://en.wikipedia.org/wiki/2009_Malaysian_Grand_Prix
\\N      \\N
3    http://en.wikipedia.org/wiki/2009_Chinese_Grand_Prix
\\N      \\N
4    http://en.wikipedia.org/wiki/2009_Bahrain_Grand_Prix
\\N      \\N
5    http://en.wikipedia.org/wiki/2009_Spanish_Grand_Prix
\\N      \\N
6     http://en.wikipedia.org/wiki/2009_Monaco_Grand_Prix
\\N      \\N
  fp2_date fp2_time fp3_date fp3_time quali_date quali_time
sprint_date
1      \\N      \\N      \\N      \\N        \\N        \\N
\\N
2      \\N      \\N      \\N      \\N        \\N        \\N
\\N
3      \\N      \\N      \\N      \\N        \\N        \\N
\\N
4      \\N      \\N      \\N      \\N        \\N        \\N
\\N
5      \\N      \\N      \\N      \\N        \\N        \\N
\\N
6      \\N      \\N      \\N      \\N        \\N        \\N
\\N
  sprint_time
1         \\N
2         \\N
```

```
3          \\N
4          \\N
5          \\N
6          \\N
```

Merging race and driver standings.

```
        driver_standings = left_join(driver_standings, races, b
```

Notes:

- Analyzing Hamilton's performance
- Replacing Lewis's driver id #1 with his name

  After verifying the the total points by year/seaon scored by LH, i discovered that those total points were wrong from my original data.

- linear regression model will be used to predict Ferrari's 2025 points based on historical driver and constructor performances. we can find the relationship between:

  a. Ferrari's past constructor points

  b. Hamilton's historical points per season

  c. Hamilton's contribution to his teams (MCLaren, Mercedes) total point

- **Formula:**

  Predicted Ferrari points = β0 + β1 (Hamilton's Avg points) + β2 (Ferrari's Avg Points)

  β0= intercept.

  β1= weight assigned to Hamilton's performance.

  β2= weight assigned to Ferrari's historical performance.

- **Time Series Forecasting:** used for long term trends.

  a. A time series model predicts future performance based on past trends over time.

b. looks at Ferrari's constructor points from previous seasons predict their expected 2025 points.

```
Hamilton_id = 1
  Hamilton_perf = driver_standings |>
  filter(driverId == Hamilton_id) |>
  group_by(year) |>
  summarize(total_points = sum(points, na.rm = TRUE))


  head(Hamilton_perf)
```

```
# A tibble: 6 × 2
   year total_points
  <int>        <dbl>
1  2007         1096
2  2008          952
3  2009          337
4  2010         2415
5  2011         2381
6  2012         2148
```

I found some f1 stat solely on lewis on espn

```
Hamilton_st = read_csv(here::here("Hamilton_st.csv"))
```

```
New names:
Rows: 19 Columns: 10
── Column specification
────────────────────────────────────────────────────────
Delimiter: "," dbl
(8): YEAR, RANK, STARTS, WINS, POLES, TOP 5, TOP 10, PTS lgl
(2): ...9, ...10
ℹ Use `spec()` to retrieve the full column specification for
this data. ℹ
Specify the column types or set `show_col_types = FALSE` to
quiet this message.
• `` -> `...9`
• `` -> `...10`
```

```
head(Hamilton_st, 35)
```

```
# A tibble: 19 × 10
```

```
    YEAR  RANK STARTS  WINS POLES `TOP 5` `TOP 10`   PTS ...9
...10
   <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>    <dbl> <dbl> <lgl>
<lgl>
 1  2007     2     17     4     6      14       16  109  NA
NA
 2  2008     1     18     5     7      13       15   98  NA
NA
 3  2009     5     16     2     4       6        9   49  NA
NA
 4  2010     4     19     3     1      13       15  240  NA
NA
 5  2011     5     19     3     1      13       16  227  NA
NA
 6  2012     4     20     4     7      10       14  190  NA
NA
 7  2013     4     19     1     5      13       17  189  NA
NA
 8  2014     1     19    11     7      16       16  384  NA
NA
 9  2015     1     19    10    11      17       18  381  NA
NA
10  2016     2     21    10    11      18       19  380  NA
NA
11  2017     1     20     9    10      18       20  363  NA
NA
12  2018     1     21    11    10      20       20  408  NA
NA
13  2019     1     21    11     5      19       21  412  NA
NA
14  2020     1     16    11    10      15       16  347  NA
NA
15  2021     2     21     8     7      19       20  388. NA
NA
16  2022     6     22     0     0      15       19  240  NA
NA
17  2023     3     21     0     1      12       20  234  NA
NA
18  2024     7     24     2     0      10       21  223  NA
NA
19    NA    NA     NA    NA    NA      NA       NA   NA  NA
NA
```

The table above is missing data from 2013-2024. I want to check tyo see if my csv file is fully loaded.

```
        nrow(Hamilton_st)
```

```
[1] 19
```

```
        str(Hamilton_st)
```

```
spc_tbl_ [19 × 10] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ YEAR  : num [1:19] 2007 2008 2009 2010 2011 ...
 $ RANK  : num [1:19] 2 1 5 4 5 4 4 1 1 2 ...
 $ STARTS: num [1:19] 17 18 16 19 19 20 19 19 19 21 ...
 $ WINS  : num [1:19] 4 5 2 3 3 4 1 11 10 10 ...
 $ POLES : num [1:19] 6 7 4 1 1 7 5 7 11 11 ...
 $ TOP 5 : num [1:19] 14 13 6 13 13 10 13 16 17 18 ...
 $ TOP 10: num [1:19] 16 15 9 15 16 14 17 16 18 19 ...
 $ PTS   : num [1:19] 109 98 49 240 227 190 189 384 381 380 ...
 $ ...9  : logi [1:19] NA NA NA NA NA NA ...
 $ ...10 : logi [1:19] NA NA NA NA NA NA ...
 - attr(*, "spec")=
  .. cols(
  ..   YEAR = col_double(),
  ..   RANK = col_double(),
  ..   STARTS = col_double(),
  ..   WINS = col_double(),
  ..   POLES = col_double(),
  ..   `TOP 5` = col_double(),
  ..   `TOP 10` = col_double(),
  ..   PTS = col_double(),
  ..   ...9 = col_logical(),
  ..   ...10 = col_logical()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
        Hamilton_st = Hamilton_st |>
        filter(!is.na(PTS))
        head(Hamilton_st, 30)
```

```
# A tibble: 18 × 10
    YEAR  RANK STARTS  WINS POLES `TOP 5` `TOP 10`   PTS ...9
...10
   <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>    <dbl> <dbl> <lgl>
<lgl>
 1  2007     2     17     4     6      14       16  109  NA
NA
```

```
 2  2008     1     18     5     7      13       15   98  NA
NA
 3  2009     5     16     2     4       6        9   49  NA
NA
 4  2010     4     19     3     1      13       15  240  NA
NA
 5  2011     5     19     3     1      13       16  227  NA
NA
 6  2012     4     20     4     7      10       14  190  NA
NA
 7  2013     4     19     1     5      13       17  189  NA
NA
 8  2014     1     19    11     7      16       16  384  NA
NA
 9  2015     1     19    10    11      17       18  381  NA
NA
10  2016     2     21    10    11      18       19  380  NA
NA
11  2017     1     20     9    10      18       20  363  NA
NA
12  2018     1     21    11    10      20       20  408  NA
NA
13  2019     1     21    11     5      19       21  412  NA
NA
14  2020     1     16    11    10      15       16  347  NA
NA
15  2021     2     21     8     7      19       20  388. NA
NA
16  2022     6     22     0     0      15       19  240  NA
NA
17  2023     3     21     0     1      12       20  234  NA
NA
18  2024     7     24     2     0      10       21  223  NA
NA
```

```
constructors = read.csv(here::here("constructors.csv"))
head(constructors)
```

```
  constructorId constructorRef       name nationality
1             1        mclaren    McLaren     British
2             2     bmw_sauber BMW Sauber      German
3             3       williams   Williams     British
4             4        renault    Renault      French
5             5     toro_rosso Toro Rosso     Italian
```

```
6             6        ferrari    Ferrari     Italian
                                                                  url
1                         http://en.wikipedia.org/wiki/McLaren
2                       http://en.wikipedia.org/wiki/BMW_Sauber
3 http://en.wikipedia.org/wiki/Williams_Grand_Prix_Engineering
4          http://en.wikipedia.org/wiki/Renault_in_Formula_One
5             http://en.wikipedia.org/wiki/Scuderia_Toro_Rosso
6                http://en.wikipedia.org/wiki/Scuderia_Ferrari
```

```
constructor_standings = read.csv(here::here("constructo
head(constructor_standings)
```

```
  constructorStandingsId raceId constructorId points position
positionText wins
1                      1     18             1     14        1
1    1
2                      2     18             2      8        3
3    0
3                      3     18             3      9        2
2    0
4                      4     18             4      5        4
4    0
5                      5     18             5      2        5
5    0
6                      6     18             6      1        6
6    0
```

```
constructor_results= read.csv(here::here("constructor_r
head(constructor_standings)
```

```
  constructorStandingsId raceId constructorId points position
positionText wins
1                      1     18             1     14        1
1    1
2                      2     18             2      8        3
3    0
3                      3     18             3      9        2
2    0
4                      4     18             4      5        4
4    0
5                      5     18             5      2        5
5    0
6                      6     18             6      1        6
```

```
6    0
```

```
constructors = read.csv(here::here("constructors.csv"))
head(constructor_standings)
```

```
  constructorStandingsId raceId constructorId points position
positionText wins
1                      1     18             1     14        1
1    1
2                      2     18             2      8        3
3    0
3                      3     18             3      9        2
2    0
4                      4     18             4      5        4
4    0
5                      5     18             5      2        5
5    0
6                      6     18             6      1        6
6    0
```

```
races = read.csv(here::here("races.csv"))
head(races)
```

```
  raceId year round circuitId                  name       date
time
1      1 2009     1         1 Australian Grand Prix 2009-03-29
06:00:00
2      2 2009     2         2  Malaysian Grand Prix 2009-04-05
09:00:00
3      3 2009     3        17    Chinese Grand Prix 2009-04-19
07:00:00
4      4 2009     4         3    Bahrain Grand Prix 2009-04-26
12:00:00
5      5 2009     5         4    Spanish Grand Prix 2009-05-10
12:00:00
6      6 2009     6         6     Monaco Grand Prix 2009-05-24
12:00:00
                                                    url
fp1_date fp1_time
1 http://en.wikipedia.org/wiki/2009_Australian_Grand_Prix
\\N      \\N
2  http://en.wikipedia.org/wiki/2009_Malaysian_Grand_Prix
\\N      \\N
```

```
3    http://en.wikipedia.org/wiki/2009_Chinese_Grand_Prix
\\N      \\N
4    http://en.wikipedia.org/wiki/2009_Bahrain_Grand_Prix
\\N      \\N
5    http://en.wikipedia.org/wiki/2009_Spanish_Grand_Prix
\\N      \\N
6     http://en.wikipedia.org/wiki/2009_Monaco_Grand_Prix
\\N      \\N
  fp2_date fp2_time fp3_date fp3_time quali_date quali_time
sprint_date
1      \\N      \\N      \\N      \\N        \\N        \\N
\\N
2      \\N      \\N      \\N      \\N        \\N        \\N
\\N
3      \\N      \\N      \\N      \\N        \\N        \\N
\\N
4      \\N      \\N      \\N      \\N        \\N        \\N
\\N
5      \\N      \\N      \\N      \\N        \\N        \\N
\\N
6      \\N      \\N      \\N      \\N        \\N        \\N
\\N
  sprint_time
1         \\N
2         \\N
3         \\N
4         \\N
5         \\N
6         \\N
```

```
Ferrari_stat = constructors |>
  filter(name == "Ferrari")
head(Ferrari_stat)
```

```
  constructorId constructorRef    name nationality
1             6        ferrari Ferrari     Italian
                                        url
1 http://en.wikipedia.org/wiki/Scuderia_Ferrari
```

Filtering only Ferrari's season by season

```
Ferrari_standings = constructor_standings |>
  filter(constructorId == 6)
head(Ferrari_standings)
```

```
  constructorStandingsId raceId constructorId points position
positionText wins
1                      6     18             6      1        6
6    0
2                     12     19             6     11        3
3    1
3                     23     20             6     29        2
2    2
4                     34     21             6     47        1
1    3
5                     45     22             6     63        1
1    4
6                     56     23             6     69        1
1    4
```

```
Ferrari_results = constructor_results |>
  filter(constructorId == 6)
head(Ferrari_results)
```

```
  constructorResultsId raceId constructorId points status
1                    6     18             6      1    \\N
2                   12     19             6     10    \\N
3                   23     20             6     18    \\N
4                   34     21             6     18    \\N
5                   45     22             6     16    \\N
6                   57     23             6      6    \\N
```

```
Ferrari_data = merge(Ferrari_standings, Ferrari_results
```

```
Ferrari_data = merge(Ferrari_data, races, by = "raceId"
```

```
Ferrari_data = merge(Ferrari_data, constructors, by = "
head(Ferrari_data)
```

```
  constructorId raceId constructorStandingsId points.x position
positionText
1             6      1                  4046        0        9
9
2             6      2                  8541        0       10
10
3             6      3                  8571        0        9
```

```
9
4            6      4                   8591        3        9
9
5            6      5                   8601        6        7
7
6            6      6                   8681       17        4
4
  wins constructorResultsId points.y status year round
circuitId
1    0                3832        0    \\N 2009     1
1
2    0                3839        0    \\N 2009     2
2
3    0                3849        0    \\N 2009     3
17
4    0                3857        3    \\N 2009     4
3
5    0                3866        3    \\N 2009     5
4
6    0                3874       11    \\N 2009     6
6
                name.x       date     time
1 Australian Grand Prix 2009-03-29 06:00:00
2  Malaysian Grand Prix 2009-04-05 09:00:00
3    Chinese Grand Prix 2009-04-19 07:00:00
4    Bahrain Grand Prix 2009-04-26 12:00:00
5    Spanish Grand Prix 2009-05-10 12:00:00
6     Monaco Grand Prix 2009-05-24 12:00:00
                                                    url.x
fp1_date fp1_time
1 http://en.wikipedia.org/wiki/2009_Australian_Grand_Prix
\\N      \\N
2  http://en.wikipedia.org/wiki/2009_Malaysian_Grand_Prix
\\N      \\N
3    http://en.wikipedia.org/wiki/2009_Chinese_Grand_Prix
\\N      \\N
4    http://en.wikipedia.org/wiki/2009_Bahrain_Grand_Prix
\\N      \\N
5    http://en.wikipedia.org/wiki/2009_Spanish_Grand_Prix
\\N      \\N
6     http://en.wikipedia.org/wiki/2009_Monaco_Grand_Prix
\\N      \\N
  fp2_date fp2_time fp3_date fp3_time quali_date quali_time
sprint_date
1      \\N      \\N      \\N      \\N        \\N        \\N
```

```
\\N
2      \\N      \\N      \\N      \\N        \\N        \\N
\\N
3      \\N      \\N      \\N      \\N        \\N        \\N
\\N
4      \\N      \\N      \\N      \\N        \\N        \\N
\\N
5      \\N      \\N      \\N      \\N        \\N        \\N
\\N
6      \\N      \\N      \\N      \\N        \\N        \\N
\\N
  sprint_time constructorRef  name.y nationality
1         \\N        ferrari Ferrari     Italian
2         \\N        ferrari Ferrari     Italian
3         \\N        ferrari Ferrari     Italian
4         \\N        ferrari Ferrari     Italian
5         \\N        ferrari Ferrari     Italian
6         \\N        ferrari Ferrari     Italian
                                          url.y
1 http://en.wikipedia.org/wiki/Scuderia_Ferrari
2 http://en.wikipedia.org/wiki/Scuderia_Ferrari
3 http://en.wikipedia.org/wiki/Scuderia_Ferrari
4 http://en.wikipedia.org/wiki/Scuderia_Ferrari
5 http://en.wikipedia.org/wiki/Scuderia_Ferrari
6 http://en.wikipedia.org/wiki/Scuderia_Ferrari
```

```r
ferrari_comb = Ferrari_data |>
  group_by(year) |>
  summarize(
    total_points = max(points.x, na.rm = TRUE),
    total_wins = sum(wins, na.rm = TRUE),
    avg_position = mean(position, na.rm = TRUE)
  )
head(ferrari_comb, 90)
```

```
# A tibble: 67 × 4
    year total_points total_wins avg_position
   <int>        <dbl>      <int>        <dbl>
 1  1958           40         11         1.6
 2  1959           32          9         2.14
 3  1960           26          1         2.75
 4  1961           45         19         1.14
 5  1962           18          0         4
 6  1963           26          5         3.9
```

```
 7  1964           45         12         3.1
 8  1965           26          0         2.9
 9  1966           31          8         1.71
10  1967           20          0         4.1
# ℹ 57 more rows
```

i wanted to change the column names like YEAR to match my ferrari's dataset

```
        hamilton_data = Hamilton_st |>
          select(YEAR,PTS, WINS) |>
          rename(year = YEAR, hamilton_points = PTS,hamilton_wi
        head(hamilton_data,50)
```

```
# A tibble: 18 × 3
    year hamilton_points hamilton_wins
   <dbl>           <dbl>         <dbl>
 1  2007             109             4
 2  2008              98             5
 3  2009              49             2
 4  2010             240             3
 5  2011             227             3
 6  2012             190             4
 7  2013             189             1
 8  2014             384            11
 9  2015             381            10
10  2016             380            10
11  2017             363             9
12  2018             408            11
13  2019             412            11
14  2020             347            11
15  2021             388.            8
16  2022             240             0
17  2023             234             0
18  2024             223             2
```

Notes:

- **Inner Join:**

  a. only keeps matches

  b. used when you only need the rows that exist in both tables.

- **party analogy example:** if i had a party list that has **Mandy, Ashley, Kharl** and another list that has **Mandy, Ashley, Odin**. Inner join will only keep **Mandy** and **Ashley** because they were on both list.

  Left Join:

  a. keeps all rows from the left, adds matches from the right.

  b. Used when you want to keep all data from the left and add details if they exist.

- Lets say the party list has **Mandy, Ashley, Kharl** and another list has **has Mandy, Ashley, Odin. LEFT JOIN** will keep **Mandy, Ashley, Kharl** and since kharl has no match on the second list it will result in a NA.

  Right Join:

  a. keeps all the rows to the right and adss matches from the left.

- if the party list on the right has **Mandy, Ashley, Odin** and your list has **Mandy, Ashley, Kharl. RIGHT JOIN** will keep **Mandy, Ashley, Odin** but since Odin wasnt on my list it will result in NA

total_points = Ferraris total points

total wins= Ferrari's total wins

avg_positions= Ferrari's average finish position

```
        ferrari_hamilton_data = merge(hamilton_data, ferrari_co
        head(ferrari_hamilton_data, 30)
```

```
   year hamilton_points hamilton_wins total_points total_wins
avg_position
1  2007           109.0             4        204.0         78
1.000000
2  2008            98.0             5        172.0         87
1.500000
3  2009            49.0             2         70.0          6
5.058824
4  2010           240.0             3        396.0         42
2.526316
5  2011           227.0             3        375.0         11
```

```
3.000000
6  2012           190.0             4        400.0          43
3.000000
7  2013           189.0             1        354.0          32
2.473684
8  2014           384.0            11        216.0           0
3.578947
9  2015           381.0            10        428.0          35
2.000000
10 2016           380.0            10        398.0           0
2.476190
11 2017           363.0             9        522.0          65
1.850000
12 2018           408.0            11        571.0          81
1.761905
13 2019           412.0            11        504.0          24
2.000000
14 2020           347.0            11        131.0           0
5.294118
15 2021           387.5             8        323.5           0
3.636364
16 2022           240.0             0        554.0          67
1.772727
17 2023           234.0             0        406.0           8
3.590909
18 2024           223.0             2        302.0          15
2.000000
```

The total_points column representing Ferrari's total points seems off. point totals are abnormaly high for a single season.

```
summary(ferrari_hamilton_data$total_points)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   70.0   237.5   385.5   351.5   422.5   571.0
```

I verified the data under the total_points column by using years 2007-2009 and verifying that the points from my table matches that of the formula one website.

```
summary(ferrari_comb$total_points)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
    7.0    38.5    74.0   148.2   202.5   571.0
```

```
head(ferrari_comb,20)
```

```
# A tibble: 20 × 4
    year total_points total_wins avg_position
   <int>        <dbl>      <int>        <dbl>
 1  1958         40           11         1.6
 2  1959         32            9         2.14
 3  1960         26            1         2.75
 4  1961         45           19         1.14
 5  1962         18            0         4
 6  1963         26            5         3.9
 7  1964         45           12         3.1
 8  1965         26            0         2.9
 9  1966         31            8         1.71
10  1967         20            0         4.1
11  1968         32            7         3.73
12  1969          7            0         5.7
13  1970         52           13         5.38
14  1971         33           19         1.82
15  1972         33            5         3.5
16  1973         12            0         4.38
17  1974         65           25         1.8
18  1975         72.5         36         2.14
19  1976         83           70         1
20  1977         95           43         1.12
```

```
ferrari_comb = Ferrari_data |>
  group_by(year) |>
  summarize(
    total_points = max(points.x, na.rm = TRUE),
    total_wins= n_distinct(raceId[wins > 0]),
    avg_position = mean(position, na.rm = TRUE)
  )
head(ferrari_comb, 70)
```

```
# A tibble: 67 × 4
    year total_points total_wins avg_position
   <int>        <dbl>      <int>        <dbl>
 1  1958           40          6         1.6
 2  1959           32          5         2.14
 3  1960           26          1         2.75
```

```
 4  1961           45          6         1.14
 5  1962           18          0         4
 6  1963           26          5         3.9
 7  1964           45          5         3.1
 8  1965           26          0         2.9
 9  1966           31          6         1.71
10  1967           20          0         4.1
# ℹ 57 more rows
```

Debugging, the race win for 2007 is 9 not 17. all atemps made to fix this failed.

```
        ferrari_wins_check = Ferrari_data |>
          select(year, raceId, wins) |>
          arrange(year, raceId)
        head(ferrari_wins_check,20)
```

```
   year raceId wins
1  1958    765    0
2  1958    766    0
3  1958    767    0
4  1958    769    0
5  1958    770    1
6  1958    771    2
7  1958    772    2
8  1958    773    2
9  1958    774    2
10 1958    775    2
11 1959    756    0
12 1959    758    0
13 1959    759    1
14 1959    761    2
15 1959    762    2
16 1959    763    2
17 1959    764    2
18 1960    746    0
19 1960    747    0
20 1960    749    0
```

```
        ferrari_2007 = Ferrari_data |>
          filter(year == 2007, wins > 0) |>
          distinct(raceId) |>
          count()
```

```
        head(ferrari_2007)
```

```
   n
1 17
```

re-merging the new Ferrari data with Hamilton's data.

```
        ferrari_hamilton_merge = merge(hamilton_data, ferrari_c
        head(ferrari_hamilton_merge)
```

```
  year hamilton_points hamilton_wins total_points total_wins
avg_position
1 2007             109             4          204         17
1.000000
2 2008              98             5          172         17
1.500000
3 2009              49             2           70          6
5.058824
4 2010             240             3          396         19
2.526316
5 2011             227             3          375         11
3.000000
6 2012             190             4          400         19
3.000000
```

- Before merging again, the wins column also has an abnormally high numbers. in 2007 there were 17 Grand Prix, with Ferrari winning 9 of those races. This issue might be ocuring due to duplication in points and wins after merging
- **n_distinct() :** counts how many different unique values exist in a column. using n_distinct(race_Id[win ==1]) will allow me to find the races won by Ferrari. this and anything else i did failed. i will proceed with what i have, keep in mid that my data is lacking.

```
        colnames(ferrari_hamilton_merge) = c("year", "hamilton_
        head(ferrari_hamilton_merge)
```

```
  year hamilton_points hamilton_wins ferrari_points
ferrari_wins       NA
1 2007             109             4            204
17 1.000000
2 2008              98             5            172
```

```
17 1.500000
3 2009              49             2             70
6 5.058824
4 2010             240             3            396
19 2.526316
5 2011             227             3            375
11 3.000000
6 2012             190             4            400
19 3.000000
```

Building a linear regression model where:

- Ferrari points are the dependent variable.
- Hamilton's points are the independent variable.

  summary(lm_model) will show**:**

- **Intercept (β0) =** Ferrari's base performance.
- **Hamilton's coefficient (β1) =** how much Ferrari's points increase per Hamilton point.
- **R^2=** how well Hamilton's performance will explain Ferrari's points.

  assuming that Lewis scores 400 points this season

  ferrari_points ~ hamilton_points tells R that:

- ferrari's points are the dependent variable (whats being predicted)
- hamilton's points are the independent varialble (the factor that affects ferrari's points).
- **data= ferrari_hamilton_merge =** tells R to use merged ferrari and hamilton datasets.

Analysis:

Intercept (β0) = 174.47

If Hamilton scores 0 points, Ferrari would still score 174.47 points. Hamilton's coefficient (**β1) = 0.655, meaning for each additional point Hamilton scores, Ferrari's point increases by 0.655.**

Ferrari points = 174.47 + (0.655 X Hamilton points). If Hamilton scores 420 points, 174.47 + (0.655 X 420)= **449.57 Points** will be scored by Ferrari in 2025.

Residuals:

1 year: -41.90 points less than the model predicted

4 year: 64.24 more points than predicted.

Fitted:

```
        lm_model = lm(ferrari_points ~ hamilton_points, data =
        head(lm_model)
```

```
$coefficients
    (Intercept) hamilton_points
    174.4674658       0.6553709

$residuals
          1           2           3           4           5
6
 -41.902894  -66.693814 -136.580640   64.243519   51.763341
101.012064
          7           8           9          10          11
12
  55.667435 -210.129890    3.836222  -25.508407  109.632899
129.141208
         13          14          15          16          17
18
  59.519725 -270.881167 -104.923688  222.243519   78.175744
-18.615176

$effects
    (Intercept) hamilton_points
   -1491.170350      313.942584     -110.576290       72.175454
60.925336

     113.674998       68.424988     -215.823231       -1.573258
-30.823267

     105.926577      121.176988       51.177025     -273.073569
-110.948199
```

```
     230.175454       86.675399       -9.074701

$rank
[1] 2

$fitted.values
       1        2        3        4        5        6        7
8
245.9029 238.6938 206.5806 331.7565 323.2367 298.9879 298.3326
426.1299
       9       10       11       12       13       14       15
16
424.1638 423.5084 412.3671 441.8588 444.4803 401.8812 428.4237
331.7565
      17       18
327.8243 320.6152

$assign
[1] 0 1
```

```
hamilton_2025_points = 420
ferrari_2025_predic = predict(lm_model, newdata = data.
head(ferrari_2025_predic)
```

```
       1
449.7232
```

I want to compare Hamilton's impact with Ferrari's historical trend. Comparing regression VS. Time series forecasting.

Notes:

Model ARIMA(0,0,0), assumes Ferrari's future points will follow the avg of their past points. With non-zero mean, the model is not predicting Ferrari to score 0 points but instead uses the historical average.

- the model predict that Ferrari will score 351 points in 2025 based on historical data. According to the chart below, the model is not 100% sure that Ferrari will take 351 points. the model is 80% sure that Ferrari will score between 165-538 points.
- 95% confidence, the model is 95% sure that Ferrari in the wider ranger of 165-538 points.

- if Ferrari improves they are predicted to score over 500 points. If Ferrari struggles they are expected to score as low as 165 points.

  It is important to not that the model is guessing, so Lewis might change this prediction.

- forecast helps predict the future.

-

```
ferrari_time_series = ts(ferrari_hamilton_merge$ferrari

arima_model = auto.arima(ferrari_time_series)

ferrari_forecast = forecast(arima_model, h=1)

head(ferrari_forecast)
```

```
$method
[1] "ARIMA(0,0,0) with non-zero mean"

$model
Series: ferrari_time_series
ARIMA(0,0,0) with non-zero mean

Coefficients:
          mean
      351.4722
s.e.   33.3715

sigma^2 = 21225:  log likelihood = -114.69
AIC=233.39   AICc=234.19   BIC=235.17

$level
[1] 80 95

$mean
Time Series:
Start = 2025
End = 2025
Frequency = 1
[1] 351.4722

$lower
```

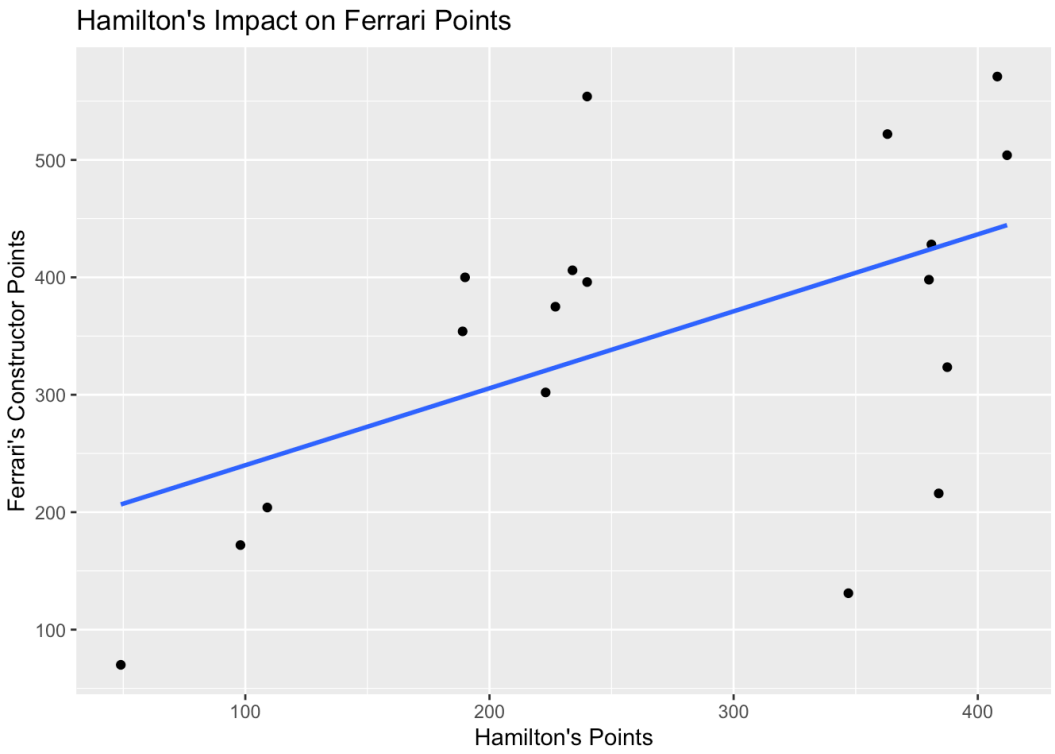
```
Time Series:
Start = 2025
End = 2025
Frequency = 1
          80%      95%
2025 164.7662 65.93004

$upper
Time Series:
Start = 2025
End = 2025
Frequency = 1
          80%      95%
2025 538.1782 637.0144
```

How Hamilton effect Ferrari.

a positive trend shows that Hamilton will have a positive impact on ferrari.

```
        ggplot(ferrari_hamilton_merge, aes(x = hamilton_points,
          geom_point() +
          geom_smooth(method = "lm", se = FALSE) +
          labs(title = "Hamilton's Impact on Ferrari Points",
               x = "Hamilton's Points", y = "Ferrari's Construc
```

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




```
plot(ferrari_forecast, main = "Ferrari Constructor Poin
     xlab = "Year", ylab = "Constructor Points")
```