Dashboard_Rough_Draft

Cameron Bayer

2023-11-01

```
# Load packages
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(sf)
```

```
## Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

```
library(shiny)
library(shinydashboard)
```

```
##
## Attaching package: 'shinydashboard'
##
## The following object is masked from 'package:graphics':
##
##     box
```

```
library(ggplot2)
library(dplyr)
library(leaflet)
library(mapview)
```

```
## The legacy packages maptools, rgdal, and rgeos, underpinning the sp package,
## which was just loaded, will retire in October 2023.
## Please refer to R-spatial evolution reports for details, especially
## https://r-spatial.org/r/2023/05/15/evolution4.html.
## It may be desirable to make the sf package available;
## package maintainers should consider adding sf to Suggests:.
## The sp package is now running under evolution status 2
##      (status 2 uses the sf package in place of rgdal)
```

```
# Load in datasets
circuits = read_csv("f1db_csv/circuits.csv")
```

```
## Rows: 77 Columns: 9
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (6): circuitRef, name, location, country, alt, url
## dbl (3): circuitId, lat, lng
##
## 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.
```

```
constructor_results = read_csv("f1db_csv/constructor_results.csv")
```

```
## Rows: 12340 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): status
## dbl (4): constructorResultsId, raceId, constructorId, points
##
## 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.
```

```
constructor_standings = read_csv("f1db_csv/constructor_standings.csv")
```

```
## Rows: 13101 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): positionText
## dbl (6): constructorStandingsId, raceId, constructorId, points, position, wins
##
## 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.
```

```
constructors = read_csv("f1db_csv/constructors.csv")
```

```
## Rows: 211 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): constructorRef, name, nationality, url
## dbl (1): constructorId
##
## 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.
```

```
driver_standings = read_csv("f1db_csv/driver_standings.csv")
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 34234 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (7): driverStandingsId, raceId, driverId, points, position, positionText...
##
## 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.
```

```
drivers = read_csv("f1db_csv/drivers.csv")
```

```
## Rows: 858 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): driverRef, number, code, forename, surname, nationality, url
## dbl  (1): driverId
## date (1): dob
##
## 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.
```

```
lap_times = read_csv("f1db_csv/lap_times.csv")
```

```
## Rows: 557007 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (5): raceId, driverId, lap, position, milliseconds
## time (1): time
##
## 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.
```

```
pit_stops = read_csv("f1db_csv/pit_stops.csv")
```

```
## Rows: 10337 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): duration
## dbl  (5): raceId, driverId, stop, lap, milliseconds
## time (1): time
##
## 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.
```

```
qualifying = read_csv("f1db_csv/qualifying.csv")
```

```
## Rows: 9915 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): q1, q2, q3
## dbl (6): qualifyId, raceId, driverId, constructorId, number, position
##
## 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.
```

```
races = read_csv("f1db_csv/races.csv")
```

```
## Rows: 1101 Columns: 18
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (13): name, time, url, fp1_date, fp1_time, fp2_date, fp2_time, fp3_date...
## dbl   (4): raceId, year, round, circuitId
## date  (1): date
##
## 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.
```

```r
results = read_csv("f1db_csv/results.csv")
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 26180 Columns: 18
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (8): position, positionText, time, milliseconds, fastestLap, rank, fast...
## dbl (10): resultId, raceId, driverId, constructorId, number, grid, position0...
##
## 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.
```

```r
sprint_results = read_csv("f1db_csv/sprint_results.csv")
```

```
## Rows: 200 Columns: 16
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (6): position, positionText, time, milliseconds, fastestLap, fastestLap...
## dbl (10): resultId, raceId, driverId, constructorId, number, grid, position0...
##
## 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.
```

```r
status = read_csv("f1db_csv/status.csv")
```

```
## Rows: 139 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): status
## dbl (1): statusId
##
## 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.
```

```r
# Clean up data column names and change variable types

# circuits
names(circuits)[3] = "Circuit"
circuits = circuits%>%
  select(-url)%>%
  mutate(alt = as.integer(alt),
         circuitRef = factor(circuitRef),
         location = factor(location),
         country = factor(country))
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `alt = as.integer(alt)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```r
# constructor_results
names(constructor_results)[4] = "constructor_points"
constructor_results = constructor_results%>%
```

```r
  select(-status)


# constructor_standings
names(constructor_standings)[4] = "constructor_total_points"
names(constructor_standings)[5] = "constructor_position"
names(constructor_standings)[7] = "race_win"
constructor_standings = constructor_standings%>%
  select(-positionText)


# constructors
names(constructors)[3] = "team"
constructors = constructors%>%
  select(-url)%>%
  mutate(constructorRef = factor(constructorRef),
         team = factor(team),
         nationality = factor(nationality))


# driver_standings
names(driver_standings)[4] = "driver_points"
names(driver_standings)[5] = "driver_position"
names(driver_standings)[7] = "driver_win"
driver_standings = driver_standings%>%
  select(-positionText)


# drivers
names(drivers)[3] = "driver_number"
names(drivers)[4] = "driver_code"
names(drivers)[8] = "driver_nationality"
drivers = drivers%>%
  select(-url)%>%
  mutate(driverRef = factor(driverRef),
         driver_nationality = factor(driver_nationality))


# lap_times
names(lap_times)[4] = "lap_times_position"
names(lap_times)[5] = "lap_time"
names(lap_times)[6] = "lap_time_milliseconds"


# pit_stops
names(pit_stops)[3] = "stops"
names(pit_stops)[5] = "pit_stop_time"
names(pit_stops)[6] = "pit_stop_duration"
names(pit_stops)[7] = "pit_stop_duration_milliseconds"
pit_stops = pit_stops%>%
  mutate(pit_stop_duration = as.numeric(pit_stop_duration))
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `pit_stop_duration = as.numeric(pit_stop_duration)`.
```

```
## Caused by warning:
## ! NAs introduced by coercion
```

```r
# qualifying
names(qualifying)[5] = "driver_number"
names(qualifying)[6] = "qualifying_position"


# races
names(races)[5] = "grand_prix"
names(races)[6] = "race_date"
names(races)[7] = "race_time"
races = races%>%
  select(raceId:race_time)


# results
names(results)[5] = "driver_number"
names(results)[6] = "starting_grid_position"
names(results)[7] = "race_position"
names(results)[8] = "race_positionText"
names(results)[9] = "race_positionOrder"
names(results)[10] = "race_points"
names(results)[11] = "race_laps"
names(results)[12] = "race_time"
names(results)[13] = "race_time_milliseconds"
names(results)[14] = "race_fastestLap"
names(results)[15] = "race_rank"
names(results)[16] = "race_fastestLapTime"
names(results)[17] = "race_fastestLapSpeed"
drivers = drivers%>%
  select(-driver_number)


# sprint_results
names(sprint_results)[5] = "driver_number"
names(sprint_results)[6] = "sprint_grid"
names(sprint_results)[7] = "sprint_position"
names(sprint_results)[8] = "sprint_positionText"
names(sprint_results)[9] = "sprint_positionOrder"
names(sprint_results)[10] = "sprint_points"
names(sprint_results)[11] = "sprint_laps"
names(sprint_results)[12] = "sprint_time"
names(sprint_results)[13] = "sprint_time_milliseconds"
names(sprint_results)[14] = "sprint_fastestLap"
names(sprint_results)[15] = "sprint_fastestLapTime"


# Merge the necessary datasets

years = races%>%
  select(raceId, year)

circuit_info = races%>%
```

```
  select(raceId, circuitId)

core_circuits = circuits%>%
  select(circuitId, Circuit)



constructor_data = constructors%>%
  left_join(constructor_standings, by = "constructorId")%>%
  left_join(constructor_results, by = c("constructorId", "raceId"))%>%
  left_join(years, by = "raceId")


circuit_races =  circuits%>%
  full_join(races, by = "circuitId")


race_data = lap_times%>%
  full_join(pit_stops, by = c("raceId", "driverId", "lap"))%>%
  full_join(drivers, by = "driverId")%>%
  left_join(years, by = "raceId")%>%
  left_join(circuit_info, by = "raceId")%>%
  left_join(core_circuits, by = "circuitId")


standings_driver = driver_standings%>%
  full_join(drivers, by = "driverId")%>%
  full_join(races, by = "raceId")


qualifying_data = qualifying%>%
  full_join(drivers, by = "driverId")%>%
  full_join(constructors, by = "constructorId")


sprint_data = sprint_results%>%
  full_join(drivers, by = "driverId")


result_data = results%>%
  full_join(drivers, by = "driverId")%>%
  mutate(race_fastestLapSpeed = as.numeric(race_fastestLapSpeed))%>%
  full_join(years, by = "raceId")
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `race_fastestLapSpeed = as.numeric(race_fastestLapSpeed)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
all_races = circuits%>%
  select(-circuitId, -circuitRef)%>%
  rename("Location" = "location",
         "Country" = "country",
         "Latitude" = "lat",
```

```
         "Longitude" = "lng",
         "Altitude (m)" = "alt")

total_active_races = races%>%
  left_join(circuits, by = "circuitId")%>%
  select(year, grand_prix, Circuit, location, country, lat, lng, alt)%>%
  rename("Year" = "year",
         "Grand Prix" = "grand_prix",
         "Location" = "location",
         "Country" = "country",
         "Latitude" = "lat",
         "Longitude" = "lng",
         "Altitude (m)" = "alt")


constructor_leaderboard <- constructor_data %>%
  group_by(team, year) %>%
  summarize(max_value = max(constructor_total_points))
```

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

```
constructor_leaderboard = na.omit(constructor_leaderboard)%>%
  arrange(desc(year), desc(max_value))%>%
  group_by(year)%>%
  mutate(rank = dense_rank(desc(max_value)))%>%
  rename(Year = year,
         Rank = rank)


filtered_results = results%>%
  select(raceId, driverId, constructorId)%>%
  left_join(years, by = "raceId")%>%
  left_join(constructors, by = "constructorId")%>%
  left_join(drivers, by = "driverId")%>%
  select(year, constructorRef, team, driverRef, forename, surname)
```

```
library(shiny)
library(shinydashboard)
library(ggplot2)
library(dplyr)
library(leaflet)
library(mapview)
library(tidytext)
library(plotly)
```

```
##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(fresh)
library(shinydashboardPlus)
```

```
##
## Attaching package: 'shinydashboardPlus'
```

```
## The following objects are masked from 'package:shinydashboard':
##
##     box, dashboardHeader, dashboardPage, dashboardSidebar, messageItem,
##     notificationItem, taskItem
```

```
## The following object is masked from 'package:graphics':
##
##     box
```

```
# library(bs4Dash)

ui <- dashboardPage(
  # skin = "midnight",
  # controlbar = dashboardControlbar(skinSelector()),
  dashboardHeader(title = "Formula 1 Dashboard"),
  dashboardSidebar(
    sidebarMenu(
      menuItem("Driver Analysis", tabName = "line_plot", icon = icon("line-chart")),
      menuItem("Constructor Performance", tabName = "slope_plot", icon = icon("area-chart")),
      menuItem("Circuit Info", tabName = "world_plot", icon = icon("globe")),
      menuItem("Lap Breakdown", tabName = "lap_plots", icon = icon("calendar")),
      menuItem("Car Performance", tabName = "car_plots", icon = icon("cog"))
      # menuItem("Team Analysis", tabName = "line_plot", icon = icon("line-chart"))
    )
  ),
  dashboardBody(
    tabItems(
      tabItem(tabName = "line_plot",
              fluidRow(
                box(width = 6,
                    textInput("name", "Enter a name:", ""),
                    uiOutput("driverYearSelection"),
                    plotOutput("line_plot")),
                box(width = 6,
                    uiOutput("raceYearSelection"),
                    uiOutput("raceCircuitSelection"),
                    plotOutput("race_position")),
                box(width = 6,
                    # uiOutput("raceYearSelection"),
                    plotOutput("final_position")),
                box(width = 6,
                    plotOutput("teams_driver")),
                box(width = 6,
                    dataTableOutput("driver_table"))
                )
              ),
```

```
      tabItem(tabName = "slope_plot",
              fluidRow(
                box(width = 6,
                    selectInput("start_year", "Select a starting year:", choices = sort(unique(construc
                    selectInput("end_year", "Select an end year:", choices = sort(unique(constructor_da
                    uiOutput("teamSelection"),
                    plotOutput("slope_plot")),
                box(width = 6,
                    selectInput("constructor_start_year", "Select a starting year:", choices = sort(uni
                    plotOutput("constructor_performance")),
                box(width = 12,
                    selectInput("constructor_team", "Select a team:", choices = unique(constructor_lead
                    plotlyOutput("constructor_ranking"))
                )
              ),
      tabItem(tabName = "world_plot",
              selectInput("race_year", "Select a starting year:", choices = sort(unique(total_active_ra
              leafletOutput("world_plot")),
      tabItem(tabName = "lap_plots",
              fluidRow(
                box(width = 6,
                    # selectInput("race_position_year", "Select a year:", choices = sort(unique(race_da
                    # selectInput("race_position_circuit", "Select a circuit:", choices = sort(unique(r
                    uiOutput("raceYearSelection_driver_position"),
                    uiOutput("raceCircuitSelection_driver_position"),
                    plotOutput("every_position"))
                )
              ),
      tabItem(tabName = "car_plots",
              fluidRow(
                box(width = 6,
                    plotOutput("speed_box")),
                box(width = 6,
                    plotOutput("pit_box"))
                )
              )
      )
    )
  )

server <- function(input, output) {

   output$raceYearSelection <- renderUI({
     years <- race_data %>%
       filter(driverRef == input$name) %>%
       select(year) %>%
       unique() %>%
       arrange(desc(year)) %>%
       pull(year)

     selectInput("driver_race_year", "Select a year:", choices = years)
     })
```

```
output$raceCircuitSelection <- renderUI({
  drive_circuit <- race_data %>%
    filter(driverRef == input$name & year == input$driver_race_year) %>%
    select(Circuit) %>%
    unique()

  selectInput("driver_race_circuit", "Select a Circuit:", choices = drive_circuit)
  })

 output$raceYearSelection_driver_position <- renderUI({
   position_years <- race_data %>%
       select(year) %>%
       unique() %>%
       arrange(desc(year)) %>%
       pull(year)
   selectInput("driver_race_year_driver_position", "Select a year:", choices = position_years)
   })

 output$raceCircuitSelection_driver_position <- renderUI({
   position_drive_circuit <- race_data %>%
     filter(year == input$driver_race_year_driver_position) %>%
     select(Circuit) %>%
     unique()
   selectInput("driver_race_circuit_driver_position", "Select a Circuit:", choices = position_drive_
   })


 output$driverYearSelection <- renderUI({
   drive_races_year <- standings_driver %>%
     filter(driverRef == input$name) %>%
     select(year) %>%
     unique() %>%
     arrange(desc(year)) %>%
     pull(year)

   selectInput("driver_year", "Select a year:", choices = drive_races_year)
  })


 output$teamSelection <- renderUI({
   active_teams <- constructor_data %>%
     filter(year >= input$start_year, year <= input$end_year) %>%
     group_by(team) %>%
     filter(n_distinct(year) == 2) %>%
     ungroup()

   selectInput("active_teams", "Select a team:", choices = active_teams$team)
  })
```

```
filtered_data_line_plot <- reactive({
  subset(standings_driver, driverRef == input$name)
})

output$line_plot <- renderPlot({
  ggplot(filtered_data_line_plot(), aes(x = round, y = driver_points, group = year, color = factor(yea
    geom_line(aes(color = "grey"), size = 1) +
    geom_line(data = filtered_data_line_plot() %>% filter(year == input$driver_year), aes(color = "red
    labs(title = "Points Over Rounds",
         x = "Round",
         y = "Points",
         color = "Year") +
    scale_color_manual(values = c("grey" = "grey", "red" = "red"))+
    theme(legend.position = "none")
})

filtered_data_slope_plot <- reactive({
  start_year <- as.integer(input$start_year)
  end_year <- as.integer(input$end_year)

  subset(constructor_data, year %in% c(start_year, end_year)) %>%
    group_by(team, year) %>%
    summarize(max_value = max(constructor_total_points))
})

output$slope_plot <- renderPlot({
  selected_team = input$active_teams

  ggplot(filtered_data_slope_plot(), aes(x = year, y = max_value, group = team, color = team)) +
    geom_line(size = 1) +
    geom_point(size = 3) +
    geom_line(data = filtered_data_slope_plot() %>% filter(team == input$active_teams), color = "red"
    geom_point(data = filtered_data_slope_plot() %>% filter(team == input$active_teams), color = "red
    scale_color_manual(values = c(selected_team = "red", "Other Teams" = "gray")) +
    labs(
      x = "Year",
      y = "Max Value",
      title = "Slopegraph of Max Values for Each Team (2021-2022)"
    )
})


filtered_data_world_plot <- reactive({
  active_races = total_active_races%>%
    filter(Year == input$race_year)
})

output$world_plot <- renderLeaflet({
  mapview(filtered_data_world_plot(), xcol = "Longitude", ycol = "Latitude", zcol = "Altitude (m)", c
})
```

```
filtered_data_race_position <- reactive({
  subset(race_data, driverRef == input$name & year == input$driver_race_year & Circuit == input$drive
})

output$race_position <- renderPlot({
  ggplot(filtered_data_race_position(), aes(lap, lap_times_position))+
    geom_point()+
    geom_line()+
    scale_y_reverse()
})
```

```
output$final_position <- renderPlot({
  driver_filter = standings_driver%>%
    filter(driverRef == input$name)

  counter = driver_filter%>%
    count(driver_position)

  ggplot(counter, aes(driver_position, n))+
    geom_bar(stat = "identity")+
    coord_flip()+
    labs(x = "Fianl Position", y = "Count")+
    scale_x_reverse(breaks = seq(1, max(driver_filter$driver_position), by = 1))
})
```

```
output$teams_driver <- renderPlot({
  driver <- filtered_results %>%
    filter(driverRef == input$name) %>%
    group_by(driverRef, team) %>%
    unique()%>%
    count()


  ggplot(driver, aes(x = reorder(team, n), y = n)) +
    geom_bar(stat = "identity") +
    labs(x = "Team",
         y = "Number of Seasons") +
    theme_minimal()+
    coord_flip()
})
```

```
filtered_constructor <- reactive({
  result <- constructor_data %>%
    filter(year == input$constructor_start_year)%>%
    group_by(team, year) %>%
    summarize(max_value = max(constructor_total_points))%>%
    mutate(year = factor(year))
})

output$constructor_performance <- renderPlot({
  ggplot(filtered_constructor(), aes(x = reorder_within(team, max_value, year), y = max_value, fill =
    geom_bar(stat = "identity") +
    labs(title = "2022 Constructors Championship", x = "Team", y = "Points") +
    coord_flip() +
    scale_x_reordered() +
    facet_wrap(~year, ncol = 1, scales = "free_y")
    })




  filtered_constructor_ranking <- reactive({
    constructor_leaderboard$tooltip <- with(constructor_leaderboard,
                                            paste("Points:", max_value))

    subset(constructor_leaderboard, team == input$constructor_team)
  })


output$constructor_ranking <- renderPlotly({
  constructor_leaderboard$tooltip <- with(constructor_leaderboard,
                                          paste("Points:", max_value))

  const_plot <- ggplot(constructor_leaderboard, aes(Year, Rank)) +
    geom_point(aes(text = tooltip), color = "grey") +
    geom_line(data = filtered_constructor_ranking(), color = "red")+
    geom_point(data = filtered_constructor_ranking(), aes(text = tooltip), color = "red")+
    # labs(title = "Lines and Points Plot")+
    scale_y_reverse()

  ggplotly(const_plot)

})



output$every_position <- renderPlot({
  test = race_data%>%
    filter(year == input$driver_race_year_driver_position, Circuit == input$driver_race_circuit_drive

  ggplot(test, aes(lap, lap_times_position, color = driverRef))+
    geom_line()+
    scale_y_reverse()
  })
```

```
output$speed_box <- renderPlot({
  race_2004 = result_data%>%
    filter(year > 2004)

  ggplot(race_2004, aes(x = factor(year), y = race_fastestLapSpeed)) +
    geom_boxplot() +
    labs(title = "Fastest Race Speed by Year", x = "Year", y = "Fastest Spped (km/h)") +
    theme_minimal()

})


output$pit_box <- renderPlot({
  pit_times = pit_stops%>%
    left_join(years, by = "raceId")

  ggplot(pit_times, aes(x = factor(year), y = pit_stop_duration)) +
    geom_boxplot() +
    theme_minimal()

})




output$driver_table <- renderDataTable({



  driver_filter_table = standings_driver%>%
    filter(year == "2022")%>%
    group_by(driverRef)%>%
    mutate(podium = ifelse(driver_position <= 3, TRUE, FALSE)) %>%
    mutate(score = ifelse(driver_position <= 10, TRUE, FALSE)) %>%
    summarize(max_value = max(driver_points, na.rm = TRUE),
              Wins = max(driver_win, na.rm = TRUE))%>%
    arrange(desc(max_value))


  result_year = results%>%
    left_join(years, by = "raceId")

  results_filter_table = result_year%>%
    filter(year == "2022")%>%
    group_by(driverId)%>%
    mutate(podium = ifelse(race_position <= 3, TRUE, FALSE)) %>%
    mutate(score = ifelse(race_position <= 10, TRUE, FALSE)) %>%
    summarize(Podiums = sum(podium, na.rm = TRUE),
              Scores = sum(score, na.rm = TRUE))
```

```
    # counter_table = driver_filter_table%>%
    #   count(driver_position)

    datatable(driver_filter_table, options = list(pageLength = 5))
    })


}

shinyApp(ui, server)
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```