Introduction

Formula 1 as an auto sport is growing very rapidly in North America as well as in Asia due to the very popular Netflix series “Drive to survive” which is packed with thrill and stories. In the context of Formula One, F1 refers to the highest level of international single-seater auto racing. The F1 World Championship is governed by the Fédération Internationale de l'Automobile (FIA) and is known for its high-speed races, advanced technology, and skilled drivers. According to Tanaka et al, the teams that will be competing are called “Constructors” to determine which driver is the fastest to win the driver’s championship since 1950 and which constructor is fastest to win the constructor championship since 1958. The main role of constructor is to manufacture the car, engine within a set of rules to make the fastest car.

According to Fields et al, aerodynamics play a crucial role in the performance of Formula 1 racecars. The paper highlights how advancements in aerodynamics have allowed F1 teams to improve their lap times, increase speed and stability, and enhance overall vehicle handling. The author also discusses how F1 regulations have evolved to limit certain aerodynamic technologies in an effort to level the playing field and promote competitive racing.

Engine is one of the most important parts of the car, it is the heart of the car. In the paper According to Alten et al, the authors discuss the specific demands placed on Formula One engines and the strategies used to develop them. The paper highlights how F1 engines must be lightweight, powerful, and reliable, while also complying with strict regulations on fuel efficiency and emissions. The authors discuss the evolution of F1 engines over time, including the shift from naturally aspirated engines to turbocharged hybrids, and how teams have employed various development strategies to optimize engine performance. The paper also touches on the role of engine suppliers and the challenges they face in meeting the demands of F1 teams.

Tyres is another important component which is responsible for the movement of the vehicle, According to the Pirelli website, tires play a crucial role in the performance of Formula One cars. There are three main types of tires used in F1: slicks, intermediates, and wets. Slicks are used in dry conditions and have no grooves, providing maximum grip on the track. Intermediates are used in light rain or damp conditions and have shallow grooves to improve traction. Wets are used in heavy rain and have deep grooves to channel water away from the tire, reducing the risk of aquaplaning. The teams must manage their tires carefully throughout the weekend, selecting the right type of tire for the conditions and balancing the need for speed with the need to preserve the tires for the duration of the race. Tire management can often be a key factor in determining the outcome of an F1 race.

The reason it is important to describe about aerodynamics, engines and tires is because these factors collectively play a major role in getting the fastest lap times around the circuit. According to Modoni et al, telemetry is described as a critical component of Formula One racing. Telemetry refers to the process of transmitting data from the car to the team's engineers in real-time during a race. This data includes information on the car's speed, engine RPM, tire temperature, fuel consumption, and other key performance indicators.

The data collected through telemetry is used by the team's engineers to monitor the car's performance and make adjustments in real-time to optimize performance. Telemetry also allows engineers to diagnose potential issues with the car and make informed decisions about when to make pit stops or other strategic moves during the race.In addition to its use during races, telemetry data is also used by F1 teams to improve car performance during testing and development.

In the world of motorsports, there is a need for an app that can display real-time laptimes and telemetry data. The current apps available in the market often lack the necessary features or have certain limitations that can make them less useful for drivers and teams.

For example, some apps have a limited range of telemetry data that can be displayed, making it difficult for teams to monitor all of the important performance indicators. Other apps may not provide real-time updates or have a delay in updating the data, which can cause issues in making timely decisions during a race. Some apps may also be difficult to use, with confusing interfaces or limited functionality.

An app that can display laptimes and telemetry data in real-time would be incredibly useful for drivers and teams. It would allow them to make informed decisions about car performance, identify issues early on, and make necessary adjustments in real-time. Additionally, an app could provide a more user-friendly and customizable interface, allowing teams to view only the data that is most important to them.

One example of the need for such an app can be seen in the 2019 Brazilian Grand Prix, where Max Verstappen's team lost telemetry data during the race. This made it difficult for the team to monitor the car's performance and make necessary adjustments, ultimately leading to Verstappen's retirement from the race.

Another example can be seen in the 2020 Styrian Grand Prix, where a timing system issue caused the display of incorrect laptimes. This confusion caused issues in race strategy and made it difficult for teams to accurately assess the performance of their cars.

Using these limitations, I decided to create a **R shiny app** which displays both the laptimes as well as the telemetries of individual laps of the 2 drivers Max Verstappen and Lewis Hamilton which can be selected through a drop down.

This app would be a prototype but **not limited** to the above mentioned functions. The reason to choose R as a language were the benefits of using R language for visualisation analysis. R is a popular programming language and open-source software environment used for statistical computing and graphics.

R language is ideal for creating high-quality visualizations and graphics due to its flexibility, ease of use, and the availability of numerous packages that provide specific functionalities. R language also allows for easy manipulation of data and is widely used for exploratory data analysis and statistical modeling.

Furthermore, R language provides extensive documentation, user community support, and a wide range of resources, making it easy for users to get started with data analysis and visualization. It is highly customizable and can be tailored to meet specific analysis requirements.

Overall, the benefits of using R language for visualisation analysis include increased flexibility, ease of use, enhanced data manipulation capabilities, and a wide range of resources and community support. These factors make R language an ideal choice for data analysis and visualization, particularly in fields such as statistics and data science.

The main reason to choose R shiny to create a app is due to the advantages mentioned by Kasprza et al as below:

Easy to Use: One of the primary advantages of R Shiny is its ease of use. Shiny allows developers to build web applications without requiring extensive knowledge of web development technologies like HTML, CSS, or JavaScript.

Seamless Integration with R: R Shiny provides a seamless integration with R programming language, making it easy to build interactive web applications that leverage the full power of R's data analysis and visualization capabilities.

Rapid Prototyping: R Shiny allows for rapid prototyping and development of web applications, which is particularly useful in research settings where speed and agility are important.

Collaboration: R Shiny facilitates collaboration among developers, enabling multiple people to work on the same web application simultaneously. This feature is especially useful for research teams, as it allows for collaborative development of web applications that can be easily shared and reused.

Interactive Visualizations: R Shiny allows for the creation of highly interactive visualizations, making it easy to communicate complex data and analyses to a wider audience.

Deployment Flexibility: R Shiny provides a range of deployment options, from hosting web applications on a local server to deploying them on cloud-based platforms, providing flexibility to developers in choosing the best deployment strategy for their needs.

Overall, the use of R Shiny in web application development offers a range of benefits, including ease of use, seamless integration with R, rapid prototyping, collaboration, interactive visualizations, and deployment flexibility. These advantages make R Shiny an ideal choice for building web applications in research settings.

Code

```{r}

# Load required packages

library(shiny)

library(ggplot2)

library(dplyr)

library(gridExtra)

```

The packages shiny, ggplot2, dplyr, and gridExtra are commonly used in data analysis and visualization tasks in R.

- shiny is a web application framework that allows developers to build interactive web applications using R. It provides an easy-to-use interface that can be used to create data-driven applications without requiring extensive knowledge of web development technologies.
- ggplot2 is a popular data visualization package in R that provides a flexible and powerful framework for creating a wide range of visualizations, including bar charts, line charts, scatter plots, and more.

- dplyr is a data manipulation package in R that provides a set of functions for performing common data manipulation tasks, such as filtering, sorting, grouping, and summarizing data. It is often used in conjunction with ggplot2 for data wrangling tasks.
- gridExtra is a package in R that provides a set of functions for arranging and combining multiple plots into a single output. It is often used in conjunction with ggplot2 to arrange multiple plots into a grid layout for improved visualization.

Together, these packages provide a powerful set of tools for data analysis and visualization in R, allowing users to perform complex data analysis tasks and create high-quality visualizations with relative ease.

```{r}

# Load telemetry data

lap_times <- read.csv("lap_times.csv")


# create an empty list to store the data

ham_ad_2021 <- list()

ver_ad_2021 <- list()


# loop through the lap numbers and load the data

for (i in 1:58) {

  ham_ad_2021[[i]] <- read.csv(paste0("ham_data_lap_2021_ad_", i, ".csv"))

  ver_ad_2021[[i]] <- read.csv(paste0("ver_data_lap_2021_ad_", i, ".csv"))

}

```


In the above code chunk, we read the “lap_times” CSV to read the data to plot the laptimes of the drivers(Max Verstappen and Lewis Hamilton). Then we create a list data type of both the drivers so that we can read the CSV’s for all 58 laps easily with help of a for loop.

```{r}
# Load lap time data
lap_subset <- subset(lap_times, raceId == 1073 & (driverId == 1 | driverId == 830))
# Define UI
ui <- fluidPage(

 # App title
 titlePanel("Abu Dhabi 2021 Lap Times & Telemetry"),

 # Sidebar with dropdown to select lap
 sidebarLayout(
  sidebarPanel(
   selectInput("lap_select", "Select Lap:", choices = unique(lap_subset$lap)),
   br(),
   tags$em("Note: Lap time data is only available for laps 2-58.")
  ),

  # Display lap times and telemetry data for selected lap
  mainPanel(
   tabsetPanel(
    tabPanel("Lap Times",
        plotOutput("lap_times_plot"),
        tags$em("This plot is a comaprison for laptimes between Lewis Hamilton(in blue) and Max Verstappen(in red)
        for Abu Dhabi Grand Prix 2021.")
    ),
    tabPanel("Telemetry Data",
        plotOutput("telemetry_table"),
        tags$em("This plot is a comaprison of various telemetries between Lewis Hamilton(in blue) and Max Verstappen(in red)
        for Abu Dhabi Grand Prix 2021 for the selected lap.")
    )
```

```
    )
  )
 )
)
```

In this code chunk, first we create a sidebar chunk which would contain a dropdown to select the lap number and also a text stating a “Note”. Then we create a main panel which contains 2 windows- 1st window is for displaying the plot of laptimes and 2nd window is to display the telemetry of the lap that is selected from the dropdown.

```{r}
# Define server
server <- function(input, output) {

 # Generate lap times plot
 output$lap_times_plot <- renderPlot({
  time_sec <- unlist(lapply(strsplit(lap_times$time, split = ":", fixed = TRUE)
                    , function(xx){ as.numeric(xx[1])*60 + as.numeric(xx[2]) }))

  lap_times$time_sec <- time_sec

  lap_times$driverId_fac <- as.factor(lap_times$driverId)
  lap_times_subset <- subset(lap_times, raceId == 1073 & (driverId == 1 | driverId == 830))
  ggplot(lap_times_subset, aes(x = lap, y = time_sec, colour = driverId_fac)) +
   geom_line() +
   scale_colour_manual(values = c("steelblue", "darkred")
                    , name = "Driver", breaks = c("1", "830")
                    , labels = c("Lewis Hamilton", "Max Verstappen")) +
   labs(y = "Lap Time (sec)", x = "Lap") + theme(legend.position = "top")
 })
```

This code chunk generates a plot of lap times for two drivers (Lewis Hamilton and Max Verstappen) in a specific race (raceId == 1073).

The lap times are initially stored as a character vector in the lap_times data frame. The lapply function is used to convert the lap times from minutes:seconds format to seconds format, and the resulting numeric vector is added as a new column to lap_times called time_sec.

The driverId column in lap_times is then converted to a factor using as.factor, and a subset of the data frame is created to include only the laps for the two specified drivers.

The ggplot function is used to create the plot, with lap number on the x-axis and lap time in seconds on the y-axis. The color of the line is determined by the driverId_fac column (which was created earlier as a factor). The scale_colour_manual function is used to specify the colors of the lines for the two drivers. The labs function is used to add axis labels, and the theme function is used to move the legend to the top of the plot.

Overall, this code generates a simple but informative visualization of lap times for two drivers in a specific race.

```{r}

# Generate telemetry table

output$telemetry_table <- renderPlot({


 for(i in 1:58)

 {

  if (input$lap_select == i)

  {

  ver_df <-ver_ad_2021[[i]]

  ham_df <-ham_ad_2021[[i]]


  ver_df$Brake <- ifelse(ver_df$Brake, 1, 0)

  ham_df$Brake <- ifelse(ham_df$Brake, 1, 0)


  p1 <- ggplot(data = ver_df, aes(x = Distance, y = Speed, color = "Max Verstappen")) +

   geom_line() +

   geom_line(data = ham_df, aes(x = Distance, y = Speed, color = "Lewis Hamilton")) +

   labs(y = "Speed(km/h)", x = "Distance(in meters)") +

   scale_color_manual(values = c("darkred", "steelblue"),

                      name = "Max Verstappen vs Lewis Hamilton",
```

```
                breaks = c("Max Verstappen", "Lewis Hamilton"),
                labels = c("Max Verstappen", "Lewis Hamilton"))

p2 <- ggplot(data = ver_df, aes(x = Distance, y = Throttle, color = "Max Verstappen")) +
 geom_line() +
 geom_line(data = ham_df, aes(x = Distance, y = Throttle, color = "Lewis Hamilton")) +
 labs(y = "Throttle(%)", x = "Distance(in meters)") +
 scale_color_manual(values = c("darkred", "steelblue"),
                name = "Max Verstappen vs Lewis Hamilton",
                breaks = c("Max Verstappen", "Lewis Hamilton"),
                labels = c("Max Verstappen", "Lewis Hamilton")) +
 guides(color = guide_legend(override.aes = list(title = "Max Verstappen vs Lewis Hamilton")))

p3 <- ggplot(data = ver_df, aes(x = Distance, y = Brake, color = "Max Verstappen")) +
 geom_line() +
 geom_line(data = ham_df, aes(x = Distance, y = Brake, color = "Lewis Hamilton")) +
 labs(y = "Brake", x = "Distance(in meters)") +
 scale_color_manual(values = c("darkred", "steelblue"),
                name = "Max Verstappen vs Lewis Hamilton",
                breaks = c("Max Verstappen", "Lewis Hamilton"),
                labels = c("Max Verstappen", "Lewis Hamilton")) +
 guides(color = guide_legend(override.aes = list(title = "Max Verstappen vs Lewis Hamilton")))

p4 <- ggplot(data = ver_df, aes(x = Distance, y = nGear, color = "Max Verstappen")) +
 geom_line() +
 geom_line(data = ham_df, aes(x = Distance, y = nGear, color = "Lewis Hamilton")) +
 labs(y = "Gear number", x = "Distance(in meters)") +
 scale_color_manual(values = c("darkred", "steelblue"),
                name = "Max Verstappen vs Lewis Hamilton",
                breaks = c("Max Verstappen", "Lewis Hamilton"),
                labels = c("Max Verstappen", "Lewis Hamilton")) +
```

```
    guides(color = guide_legend(override.aes = list(title = "Max Verstappen vs Lewis Hamilton")))



  p5 <- ggplot(data = ver_df, aes(x = Distance, y = RPM, color = "Max Verstappen")) +
    geom_line() +
    geom_line(data = ham_df, aes(x = Distance, y = RPM, color = "Lewis Hamilton")) +
    labs(y = "RPM", x = "Distance(in meters)") +
    scale_color_manual(values = c("darkred", "steelblue"),
                       name = "Max Verstappen vs Lewis Hamilton",
                       breaks = c("Max Verstappen", "Lewis Hamilton"),
                       labels = c("Max Verstappen", "Lewis Hamilton")) +
    guides(color = guide_legend(override.aes = list(title = "Max Verstappen vs Lewis Hamilton")))


  grid.arrange(p1, p2, p3, p4, p5 ,ncol = 1, nrow = 5)
}}})


}
```

This code chunk is a part of a Shiny app that generates telemetry plots for two Formula 1 drivers, Max Verstappen and Lewis Hamilton. The app allows the user to select a lap number using a slider input, and based on the selected lap, the code generates telemetry plots for both drivers for that lap.

The code first checks which lap has been selected using the input$lap_select variable. It then retrieves the corresponding data frames for both drivers for that lap from two different lists, ver_ad_2021 and ham_ad_2021.

The code then preprocesses the data by converting the binary Brake variable to 0s and 1s using the ifelse function.

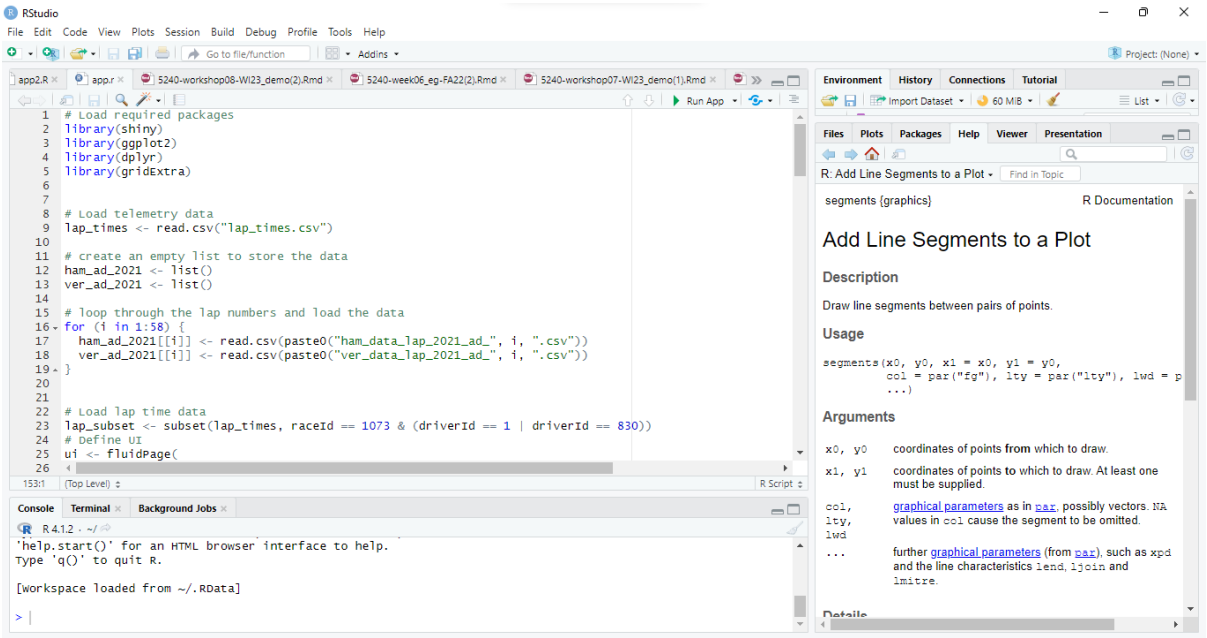
After preprocessing, the code generates five different telemetry plots using ggplot2, each for a different variable, including speed, throttle, brake, gear number, and RPM.

The grid.arrange function is used to arrange all the generated plots in a single column using ncol = 1 and nrow = 5 parameters.

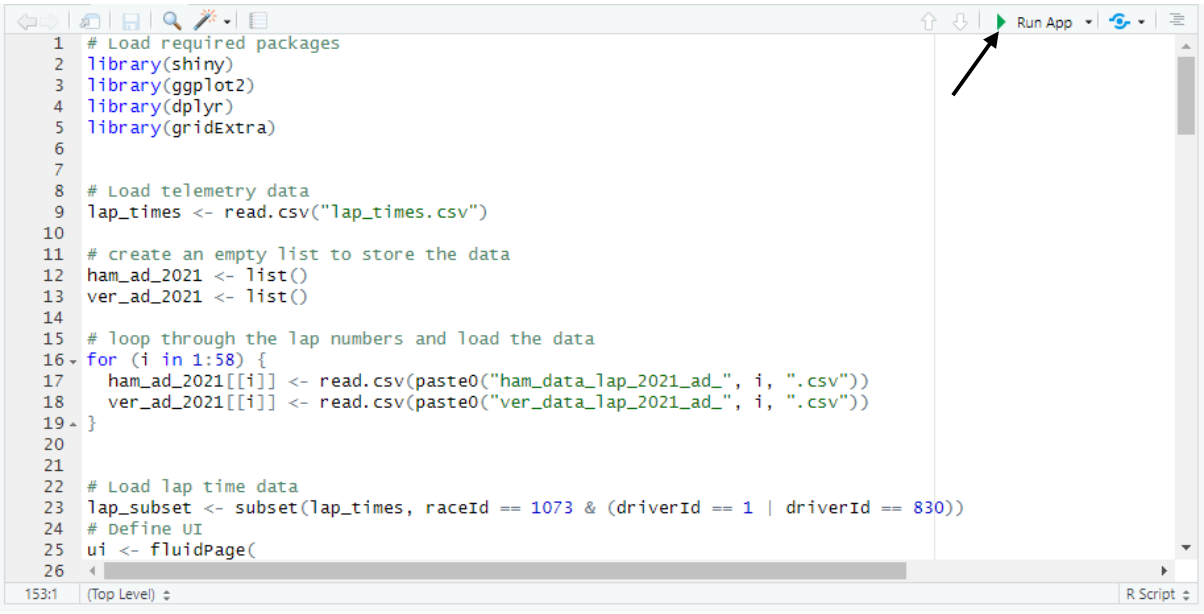
Finally, the renderPlot function is used to generate the plot in the Shiny app's output section named telemetry_table.

Instruction

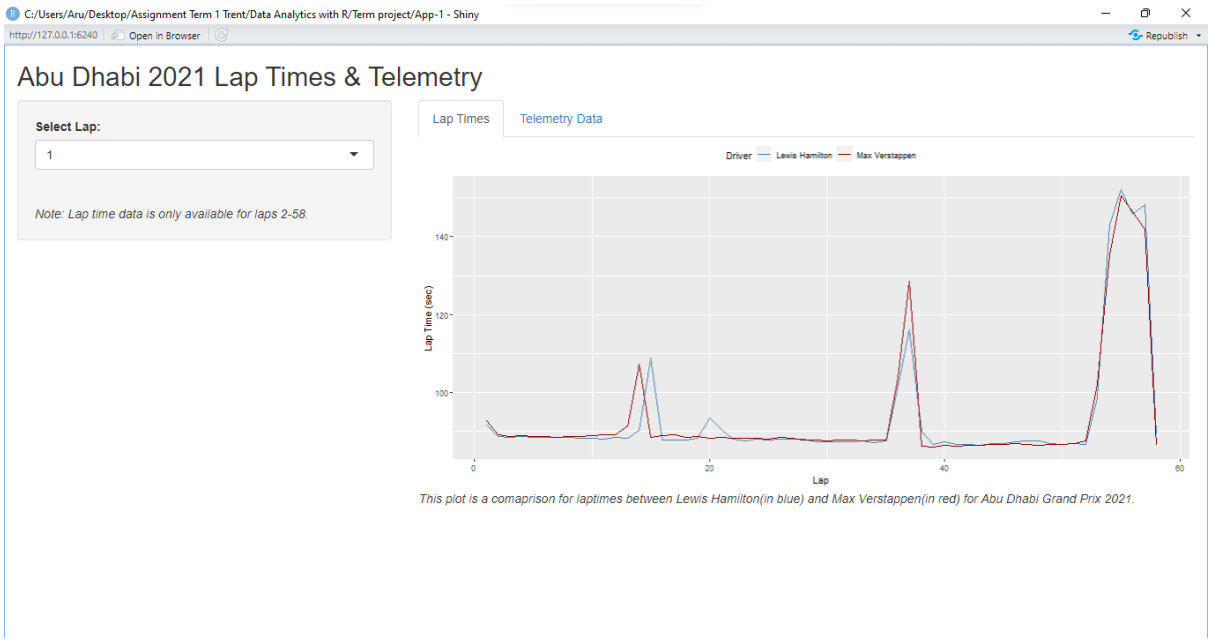
1.Open the app.r file via R studio. You should get a window as shown in figure below.



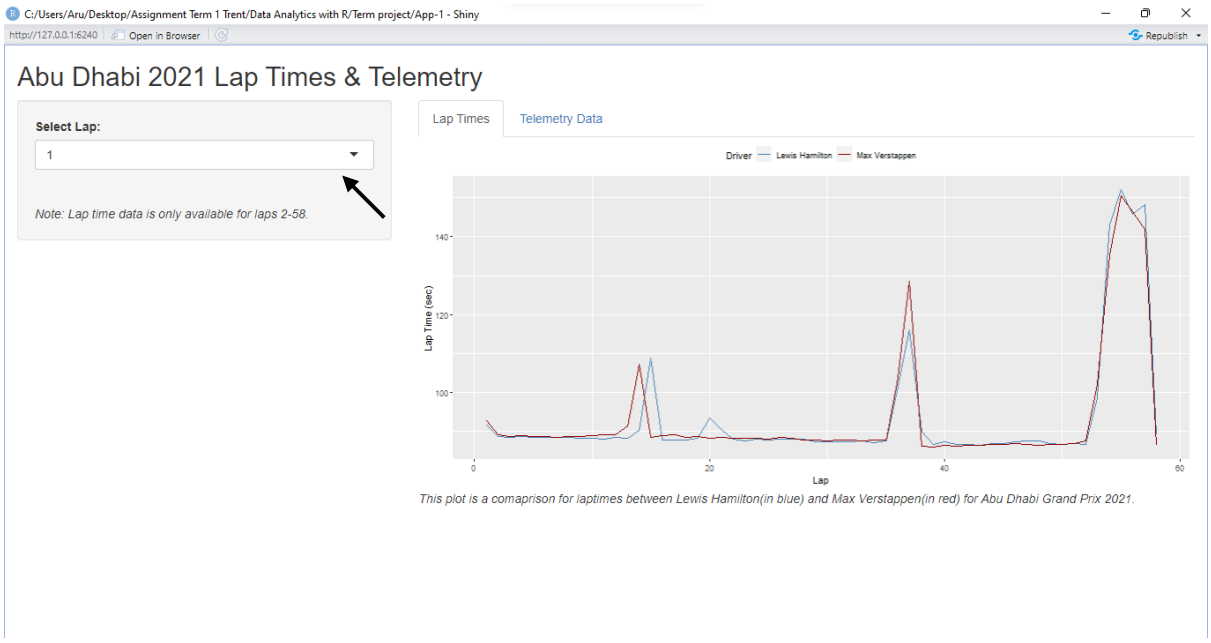
2. The left top pane contains the code about the functionality and UI of the app. If you want to run the app, press the “Run App” button as which is located as shown in below image.



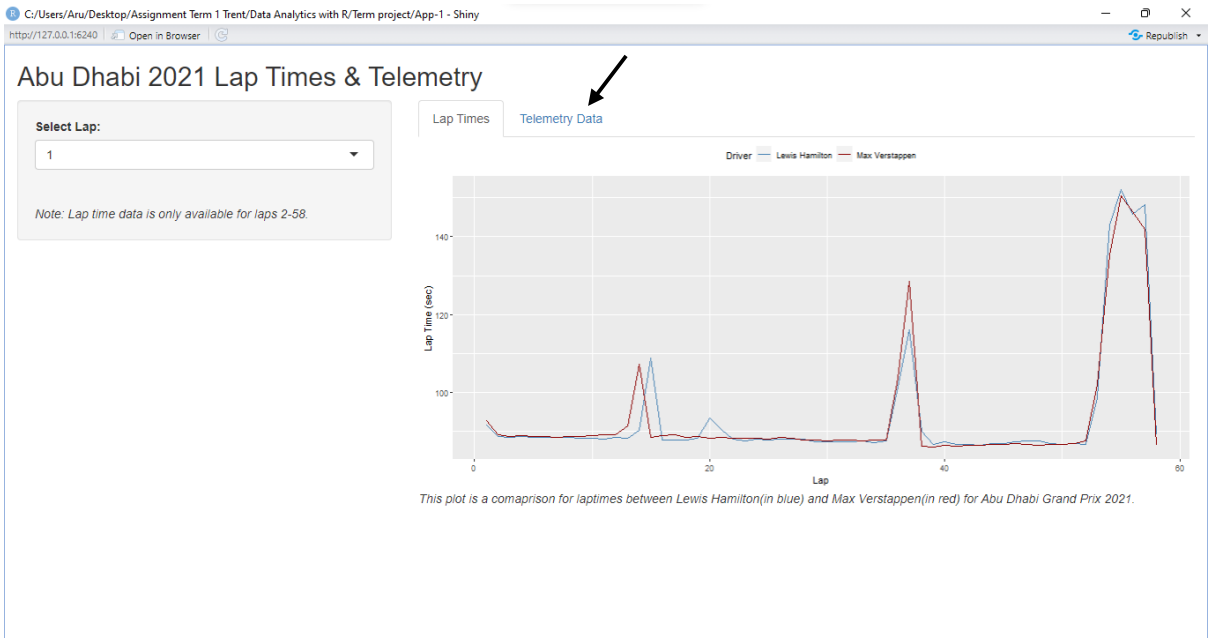
3.After pressing the “Run App” button, you should get a window pane opened in an R studio.



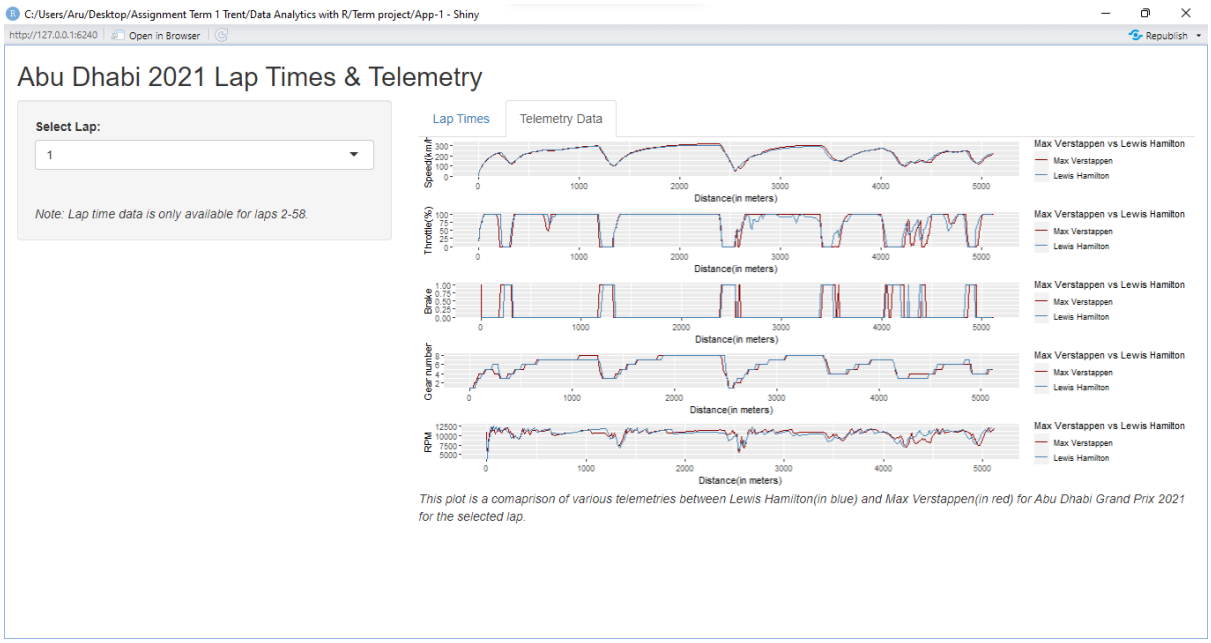
4. In this, the lap number (1-58) can be selected via a dropdown button on the left side panel.



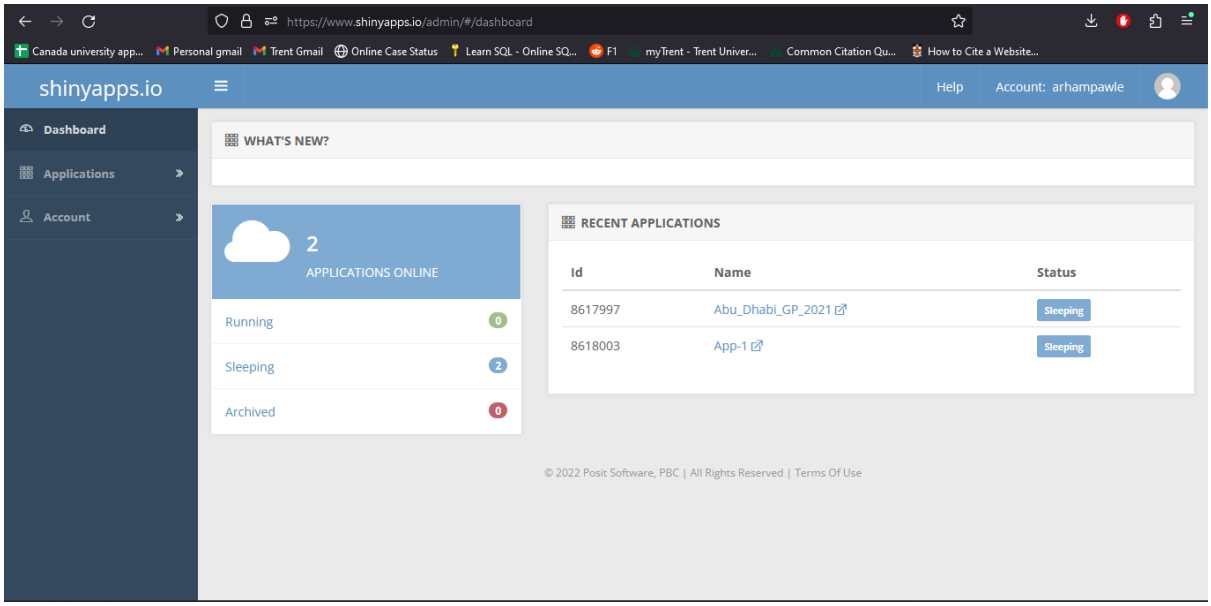
5.We have 2 windows in the main panel, one for Lap time plot of both drivers(Max Verstappen and Lewis Hamilton) for the whole race and other window for Telemetry plot for the lap selected from drop down. To see the telemetry plot, click on the ”Telemetry Data” button on the main panel.



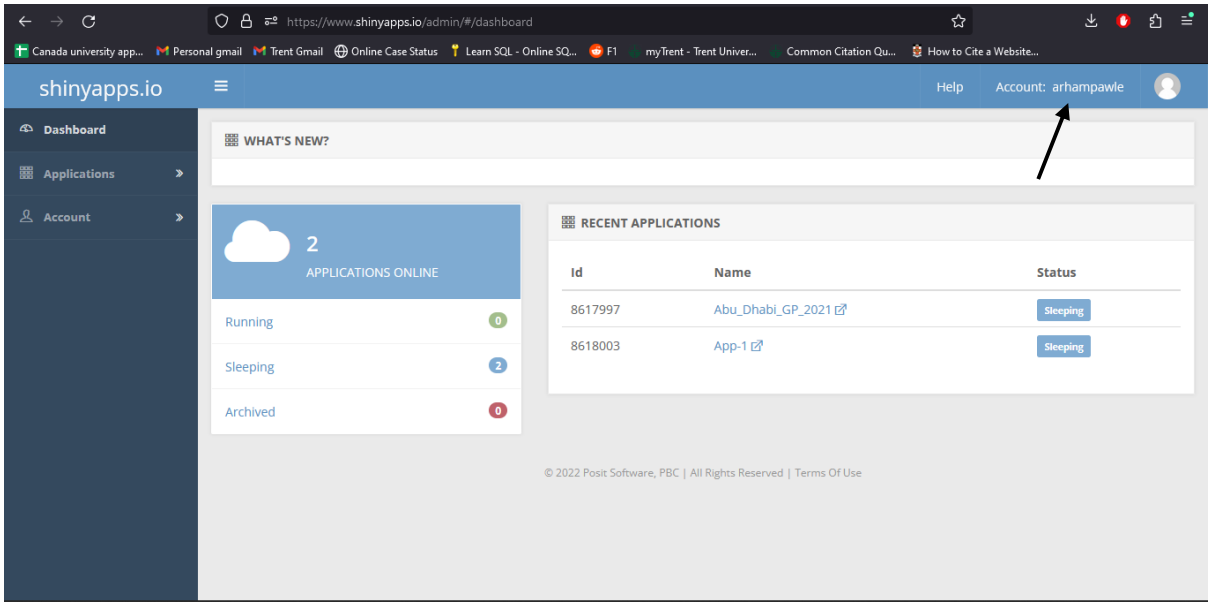
6.You should get a window pane as shown below.



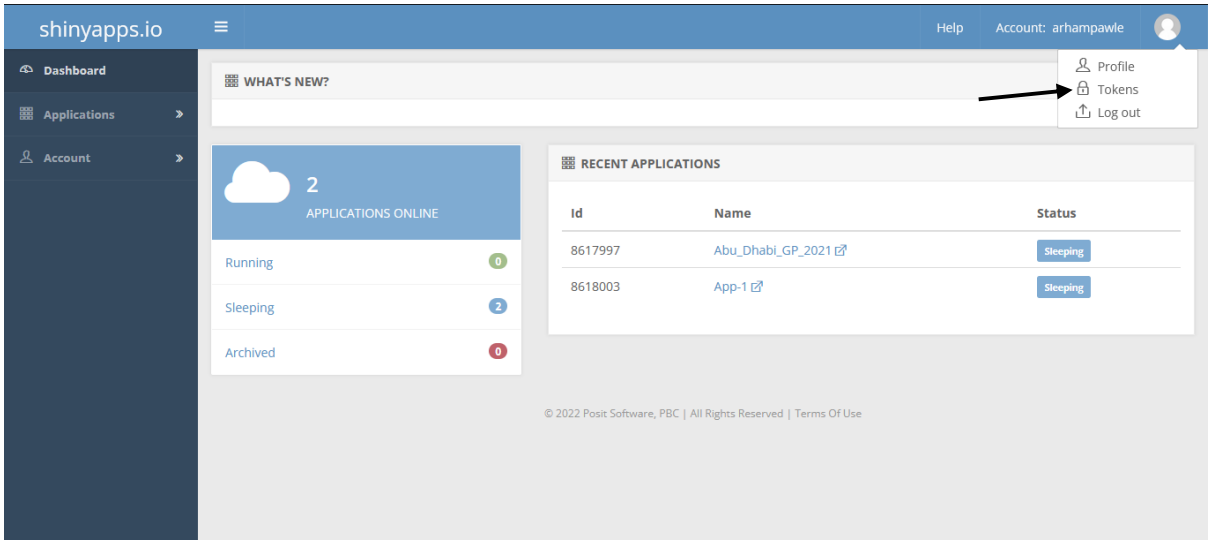
7. If you want to publish in this app via a shareable link, go to www.shinyapps.io, create a account and you should get a dashboard as shown below.



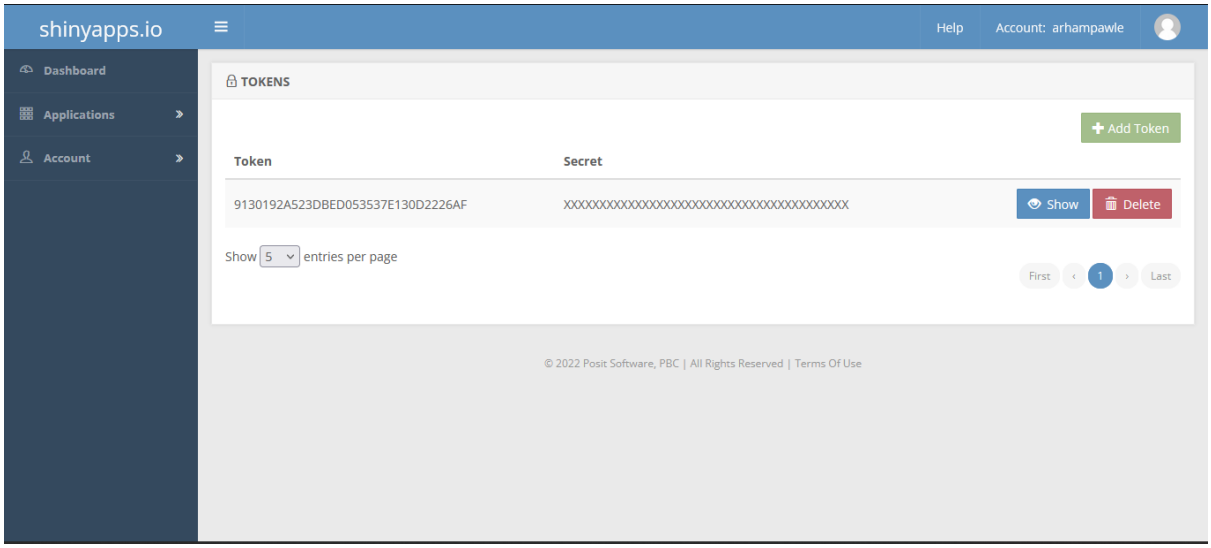
8.Now we need to create a token for our app which can be done by clicking the profile picture right to the username on the top right of the window pane.



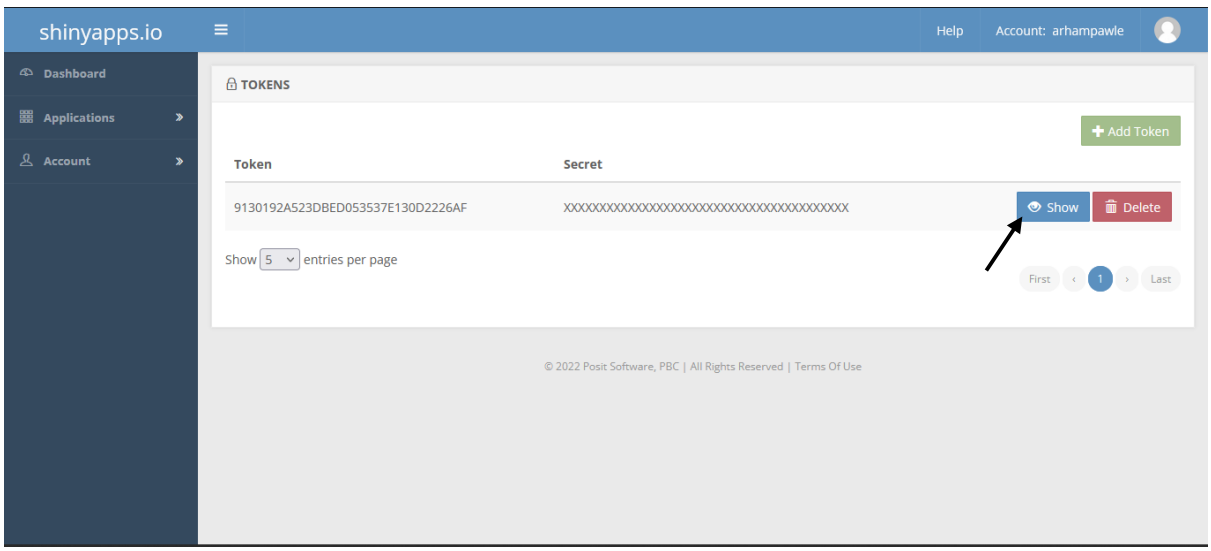
You should get a dropdown menu when you click the profile picture, select “Tokens” from the menu.



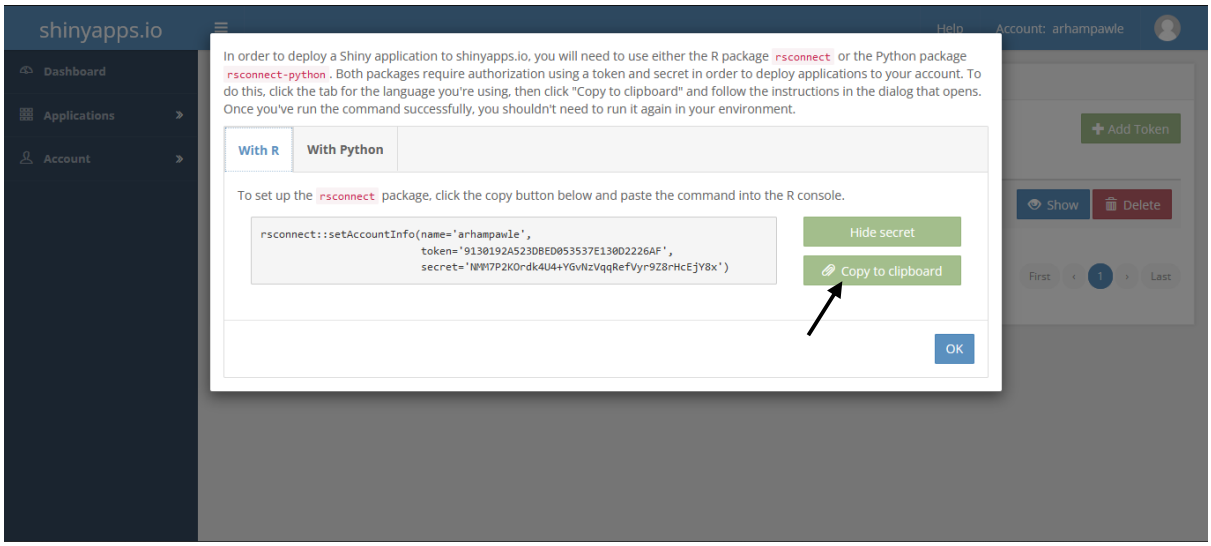
10.You should get the Token page containing all the tokens that are being used if you have created any R shiny app prior.



11.Click on the show button to reveal the “Secret”

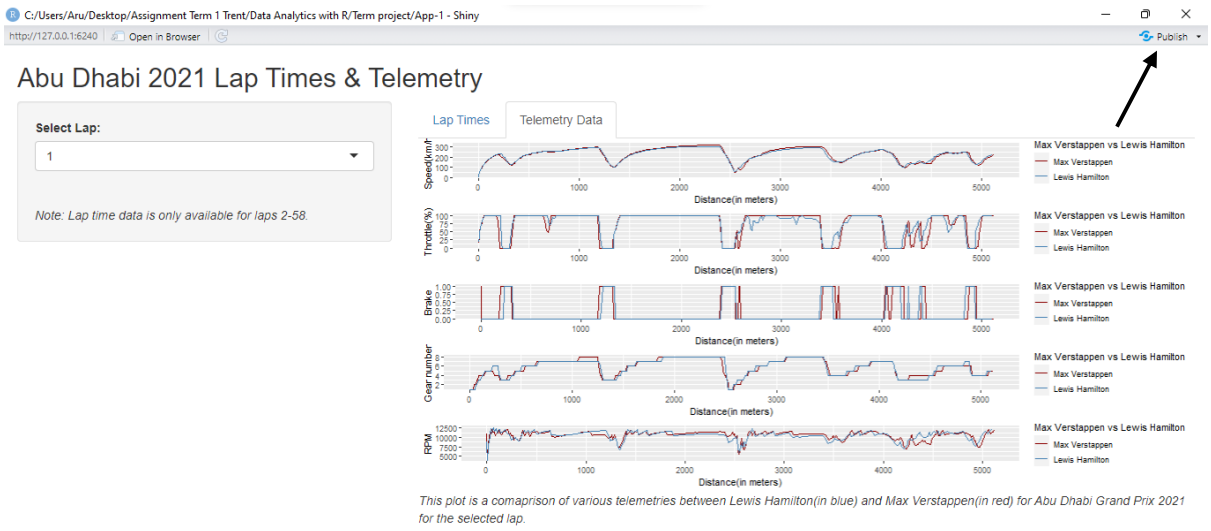


12.Click on the “Copy to clipboard” button to get the Token for the app.

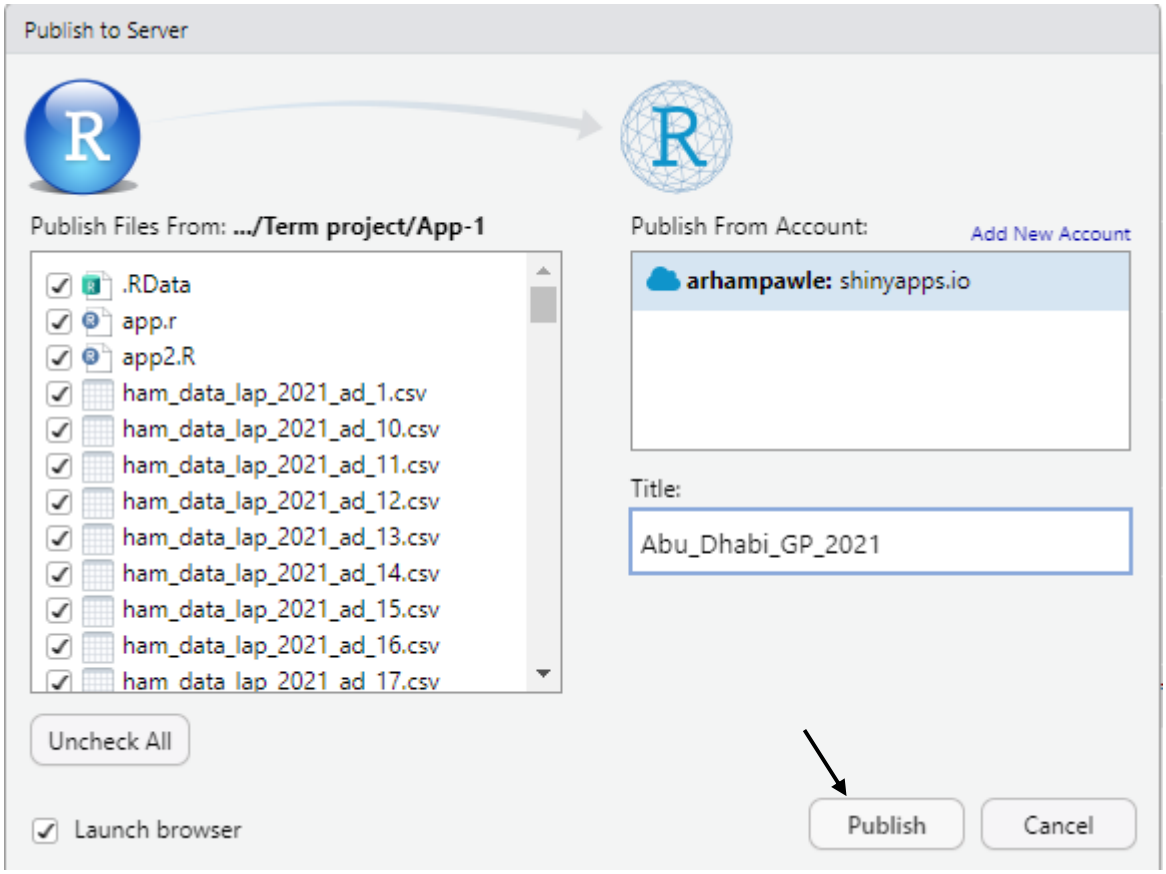


13.Paste the token in the Console of the R studio and press the Enter button from your keyboard.

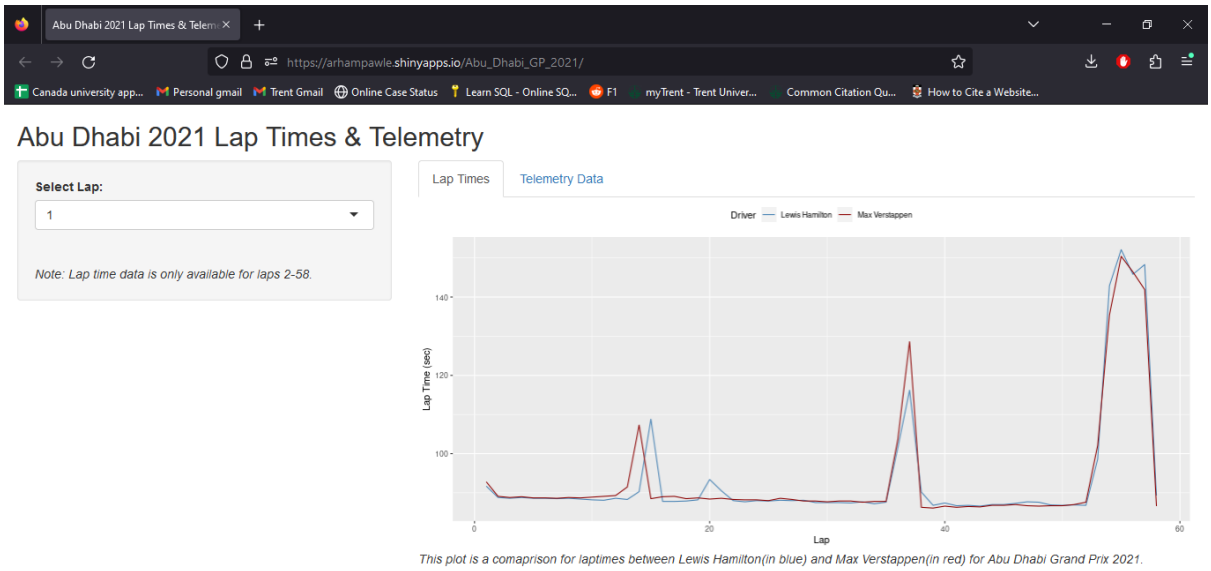
14.If you want to publish in this app via a shareable link, click the button “Publish” button.



15.You should get a window as shown in below pic. Enter the Title of your choice and press the “Publish” button.



15.After publishing the “Publish” button , it takes few minutes(installs dependencies) before opening the app on your default browser as shown in pic.



16.The link in the URL bar is a shareable link (https://arhampawle.shinyapps.io/Abu_Dhabi_GP_2021/) which can be shared with anyone irrespective if they have R installed in their device or not.