Project Report

Group 22

2023-11-09

Introduction

This report provides a detailed analysis of the Shiny app developed for cricket player ODI data .The app is designed to allow users to interact with cricket data in a user-friendly manner, providing insights into player performance, team statistics, match simulations, and more.

Data

Data for the first tab of shiny is in form of list , there are mainly two lists which are named as Player_data and Player_name . In Player_data list data which is scraped is saved in well defined format with respect to player with there country . In Player_name list all player named are saved with respect to there country . The interesting thing in data saving is that the list named are of country name and of player name and in a list there is another list i.e in Player_data list there is a list for country names and in that there is a list named after players. And the same data is used to simulate the Cricket Match . Data for the second tab is saved in the form of an Rdata called Teams_Table.Rdata, which saves data in the form of a table called table, and a function called team_filter which collects data of all matches which are played by a certain team. That file saves the data of all matches, including the winners, losers, their scores, margins and more. Happiness_Index.csv saves the data of Happiness Index of countries from 2013 to 2023. gdp_per_capita.csv saves data of GDP per Capita of countries.

Scraping of data

Data for first tab that is a list which contain batting and bowling data is scraped form a website called Cricmetric. First we formed a function and that give list to the function that is player name list then we got the data in output. Function for batting data can be seen by clicking here and for bowling data click here and for how we combine data in a list to see click here.

Data stored in Teams_Table.Rdata is scraped from Cricinfo and can be seen by clicking here. We save the table containing the ODI Matches data, along with a function team_filter, which filters out data of matches played by selected team.

First of all we scraped different team's ODI data using the following code -

```r
# This is to scrape ODI data from Cricinfo year wise.

table <- NULL
for (i in 1971:2023) {
  print(i)
  hap <-paste0("https://www.espncricinfo.com/records/year/team-match-results/",
               i, "-", i, "/one-day-internationals-2")
  html <- read_html(hap)
  table1 <- html_table(html)[[1]]
  table1 <- table1 %>% filter(X1 != "Team 1")
  link <- html_elements(html, "a")
  link <- html_attr(link, "href")
  link <- unique(link)
  link <- link[substr(link, 1, 8) == "/series/"]
  table1 <- cbind(table1, paste0("https://www.espncricinfo.com", link))
  table1 <- cbind(table1, numeric(dim(table1)[1]))
  table1 <- cbind(table1, numeric(dim(table1)[1]))
  table <- rbind(table, table1)
}

colnames(table) <- c("Team 1", "Team 2", "Winner", "Margin", "Ground",
                     "Match Date","Scorecard", "Link", "Winner_Score",
                     "Loser_Score")

# This is to remove all matches, which were Draws or Ties.
# This does not allow us to scrape data of the World Cup 2019 finals.
# However, we add this data separately in the end.

table <- table %>%
  filter(Winner != "no result") %>%
  filter(Winner != "tied") %>%
  filter(Margin != "-")
links <- table$Link

# We go to the site containing the scorecard of each individual match,
# to scrape the scores of each team, and the winner, using the link in the
# previously scraped table.

for (i in 1:dim(table)[1]) {
  print(i)
  html <- read_html(links[i])
  tables <- html_table(html)
  table_1 <- tables[[1]]
  table_2 <- tables[[3]]
  name <- html_elements(html, "span")
  name <- html_text(name)
  name <- unique(name)
  c1 <- table_1$R[dim(table_1)[1] - 2]
  c2 <- table_2$R[dim(table_2)[1] - 2]
  if (!is.na(as.integer(substr(table_1$R[dim(table_1)[1] - 1], 1, 1)))) {
    c1 <- table_1$R[dim(table_1)[1] - 1]
  }
  if (!is.na(as.integer(substr(table_2$R[dim(table_2)[1] - 1], 1, 1)))) {
    c2 <- table_2$R[dim(table_2)[1] - 1]
  }
  total_1 <- NULL
  total_2 <- NULL
  if (substr(c1, nchar(c1) - 1, nchar(c1) - 1) == "/") {
    total_1 <- as.integer(substr(c1, 1, which(strsplit(c1, "")[[1]] == "/") - 1))
  }
```

```r
  else {
    total_1 <- as.integer(c1)
  }
  if (substr(c2, nchar(c2) - 1, nchar(c2) - 1) == "/") {
    total_2 <- as.integer(substr(c2, 1, which(strsplit(c2, "")[[1]] == "/") - 1))
  }
  else {
    total_2 <- as.integer(c2)
  }

  winner <- table$Winner[i]
  loser <- NULL
  if (table$`Team 1`[i] == winner) {
    loser <- table$`Team 2`[i]
  }
  else {
    loser <- table$`Team 1`[i]
  }
  ind1 <- which(name == paste0(winner, " Innings"))
  ind2 <- which(name == paste0(loser, " Innings"))
  if (ind1 < ind2) {
    table$Winner_Score[i] <- total_1
    table$Loser_Score[i] <- total_2
  }
  else {
    table$Loser_Score[i] <- total_1
    table$Winner_Score[i] <- total_2
  }
}

# We use this to determine the team which lost the match.
t1 <- table
Loser <- numeric(length(t1$`Team 1`))
for (i in 1:length(Loser)) {
  if (t1$Winner[i] == t1$`Team 1`[i]) {
    Loser[i] = t1$`Team 2`[i]
  }
  if (t1$Winner[i] == t1$`Team 2`[i]) {
    Loser[i] = t1$`Team 1`[i]
  }
}

table <- cbind(t1, Loser)
table <- table %>% select(Winner, Loser, Margin, Winner_Score, Loser_Score, Ground,
`Match Date`)

mon <- month.abb

# This is to convert the Match Date of the table to a usable form,
# such that it can be converted to Date type in R.

conv_date <- function(date) {
  dd <- strsplit(date, " ")[[1]]
  y <- dd[3]
  m <- which(mon == dd[1], arr.ind = TRUE)
  d <- dd[2]
  if (!(gregexpr(pattern ='-',d)[[1]][1])) {
    d <- substr(d, 1, nchar(d) - 1)
  }
  if (gregexpr(pattern ='-',d)[[1]][1]) {
    d <- substr(d, gregexpr(pattern ='-',d)[[1]][1] + 1, nchar(d) - 1)
  }
  date <- paste0(y, "/", m, "/", d)
  return(date)
}

for (i in 1:length(table$`Match Date`)) {
  table$`Match Date`[i] <- conv_date(table$`Match Date`[i])
}

# This is the Tied World Cup Finals, which we are adding separately.

v <- c("England", "New Zealand", "0 runs", 241, 241, "Lord's", "2019-07-14",
"Bowling")
table <- rbind(table, v)
table$`Match Date` <- as.Date(table$`Match Date`)


# This is to determine which team won, based on which innings it played.

Winning_Innings <- NULL
for (i in 1:length(table$Margin)) {
  if ((substr(table$Margin[i], nchar(table$Margin[i]) - 6, nchar(table$Margin[i]) - 1) == "wicket") | (s
    Winning_Innings <- append(Winning_Innings, "Bowling")
  }
  else {
    Winning_Innings <- append(Winning_Innings, "Batting")
  }
}

colnames(table)[7] <- "Match Date"
table <- data.frame(table, Winning_Innings)
table[4491, 7] <- as.Date("2019-07-14")

# This function filters out matches played by a certain team selected.

team_filter <- function(team) {
  table <- table %>% select(Winner, Loser, Margin, Winning_Innings, Winner_Score,
  Loser_Score, Ground, `Match Date`)
  table1 <- table %>% filter(Winner == team | Loser == team)
  Team_Score <- NULL
  Opp_Score <- NULL
  Opponent <- NULL
  Team <- NULL
  for (i in 1:length(table1$Loser)) {
    if (table1$Winner[i] == team) {
      Team_Score = append(Team_Score, table1$Winner_Score[i])
      Opp_Score = append(Opp_Score, table1$Loser_Score[i])
      Team <- append(Team, team)
      Opponent = append(Opponent, table1$Loser[i])
    }
    if (table1$Loser[i] == team) {
      Team_Score = append(Team_Score, table1$Loser_Score[i])
      Opp_Score = append(Opp_Score, table1$Winner_Score[i])
      Team <- append(Team, team)
      Opponent = append(Opponent, table1$Winner[i])
    }
  }
  table1 <- cbind(table1, Team_Score, Opp_Score, Team, Opponent) %>% select(Team,
  Opponent, Winning_Innings, Team_Score, Opp_Score, Winner, Loser, Winner_Score,
  Loser_Score, Margin, Ground, 'Match Date')
  table1$Winner_Score <- as.integer(table1$Winner_Score)
  table1$Loser_Score <- as.integer(table1$Loser_Score)
  table1$Team_Score <- as.integer(table1$Team_Score)
  table1$Opp_Score <- as.integer(table1$Opp_Score)
  return(table1)
}
colnames(table)[7] <- "Match Date"
table <- table %>% arrange(table$`Match Date`)
save(table, team_filter, file = "Data Sets/Teams_Table.Rdata")
```

We also use this code to update the Team ODI Dataset, which instead of collecting the whole table again, checks if Cricinfo has added more matches, and only adds the new matches.

```
load(file = "Data Sets/Teams_Table.Rdata")
table2 <- NULL

# Again, scrapes the data from Cricinfo.

for (i in 1971:2023) {
  print(i)
  html <- read_html(paste0("https://www.espncricinfo.com/records/year/team-match-results/",
                           i, "-", i, "/one-day-internationals-2"))
  table1 <- html_table(html)[[1]]
  table1 <- table1 %>% filter(X1 != "Team 1")
  link <- html_elements(html, "a")
  link <- html_attr(link, "href")
  link <- unique(link)
  link <- link[substr(link, 1, 8) == "/series/"]
  table1 <- cbind(table1, paste0("https://www.espncricinfo.com", link))
  table1 <- cbind(table1, numeric(dim(table1)[1]))
  table1 <- cbind(table1, numeric(dim(table1)[1]))
  table2 <- rbind(table2, table1)
}
colnames(table2) <- c("Team 1", "Team 2", "Winner", "Margin", "Ground",
"Match Date", "Scorecard", "Link", "Winner_Score", "Loser_Score")

table2 <- table2 %>% filter(Winner != "no result") %>%
  filter(Winner != "tied") %>% filter(Margin != "-")
```

```r
table2 <- rbind(c("England", "New Zealand", "England", "0 runs",
                  "Lord's", "2019-07-14", "Scorecard", "Link", 241, 241), table2)

d1 <- dim(table)[1]
d2 <- dim(table2)[1]

# Checks if any new matches have been played since we last scraped.
#  no, then it stops. If yes, it adds only the new matches, similar to the
# original scrape code.

if (d1 == d2) {
  stop("No new data.")
}
table2 <- table2[c(d1 + 1: d2),]
table2 <- table2 %>% filter(!is.na(table2$`Team 1`))
links <- table2$Link

for (i in 1:dim(table2)[1]) {
  print(i)
  html <- read_html(links[i])
  tables <- html_table(html)
  table_1 <- tables[[1]]
  table_2 <- tables[[3]]
  name <- html_elements(html, "span")
  name <- html_text(name)
  name <- unique(name)
  c1 <- table_1$R[dim(table_1)[1] - 2]
  c2 <- table_2$R[dim(table_2)[1] - 2]
  if (!is.na(as.integer(substr(table_1$R[dim(table_1)[1] - 1], 1, 1)))) {
    c1 <- table_1$R[dim(table_1)[1] - 1]
  }
  if (!is.na(as.integer(substr(table_2$R[dim(table_2)[1] - 1], 1, 1)))) {
    c2 <- table_2$R[dim(table_2)[1] - 1]
  }
  total_1 <- NULL
  total_2 <- NULL
  if (substr(c1, nchar(c1) - 1, nchar(c1) - 1) == "/") {
    total_1 <- as.integer(substr(c1, 1, which(strsplit(c1, "")[[1]] == "/") - 1))
  }
  else {
    total_1 <- as.integer(c1)
  }
  if (substr(c2, nchar(c2) - 1, nchar(c2) - 1) == "/") {
    total_2 <- as.integer(substr(c2, 1, which(strsplit(c2, "")[[1]] == "/") - 1))
  }
  else {
    total_2 <- as.integer(c2)
  }

  winner <- table2$Winner[i]
  loser <- NULL
  if (table2$`Team 1`[i] == winner) {
    loser <- table2$`Team 2`[i]
  }
  else {
    loser <- table2$`Team 1`[i]
  }
  ind1 <- which(name == paste0(winner, " Innings"))
  ind2 <- which(name == paste0(loser, " Innings"))
  if (ind1 < ind2) {
    table2$Winner_Score[i] <- total_1
    table2$Loser_Score[i] <- total_2
  }
  else {
    table2$Loser_Score[i] <- total_1
    table2$Winner_Score[i] <- total_2
  }
}

t1 <- table2
Loser <- numeric(length(t1$`Team 1`))

for (i in 1:length(Loser)) {
  if (t1$Winner[i] == t1$`Team 1`[i]) {
    Loser[i] = t1$`Team 2`[i]
  }
  if (t1$Winner[i] == t1$`Team 2`[i]) {
    Loser[i] = t1$`Team 1`[i]
  }
}
table2 <- cbind(t1, Loser)
table2 <- table2 %>% select(Winner, Loser, Margin, Winner_Score, Loser_Score,
Ground, `Match Date`)

mon <- month.abb

conv_date <- function(date) {
  dd <- strsplit(date, " ")[[1]]
  y <- dd[3]
  m <- which(mon == dd[1], arr.ind = TRUE)
  d <- dd[2]
  if (!(gregexpr(pattern ='-',d)[[1]][1])) {
    d <- substr(d, 1, nchar(d) - 1)
  }
  if (gregexpr(pattern ='-',d)[[1]][1]) {
    d <- substr(d, gregexpr(pattern ='-',d)[[1]][1] + 1, nchar(d) - 1)
  }
  date <- paste0(y, "/", m, "/", d)
  return(date)
}

for (i in 1:length(table2$`Match Date`)) {
  table2$`Match Date`[i] <- conv_date(table2$`Match Date`[i])
}
table2$`Match Date` <- as.Date(table2$`Match Date`)
```

```r
Winning_Innings <- NULL
for (i in 1:length(table2$Margin)) {
  if ((substr(table2$Margin[i], nchar(table2$Margin[i]) - 6, nchar(table2$Margin[i]) - 1) == "wicket")
    Winning_Innings <- append(Winning_Innings, "Bowling")
  }
  else {
    Winning_Innings <- append(Winning_Innings, "Batting")
  }
}


table2 <- data.frame(table2, Winning_Innings)
colnames(table2)[7] <- "Match Date"
table <- rbind(table, table2)
save(table, team_filter, file = "Data Sets/Teams_Table.Rdata")
```

After that we scraped player's individual bowling and batting data using the following code chunks -

For Bowling Data -

```r
Scrap_Bowling_data <- function(name){
 # removing spaces and adding + sign between them to make links
  name1 <- gsub(pattern = " ", replacement = "+", name )
  link <- paste0("http://www.cricmetric.com/playerstats.py?player=",name1,
               "&role=bowler&format=ODI&groupby=year")
  # name <- read_html(link) %>% html_element(".panel-heading") %>% html_text2()
  data <- read_html(link) %>%            # scraping data from the websites
    html_table()
  Data <- data[[1]]
  row_number <- dim(Data)[1]       # taking the no of rowns of the data set
  # removing commas so that the the value can be changed itno numberic
  Data$Runs <-  gsub(pattern = ",",replacement = "",x = Data$Runs)
  Data$Overs <- gsub(pattern = ",",replacement = "",x = Data$Overs)
  Data$'4s' <- gsub(pattern = ",",replacement = "",x = Data$'4s')

  # making vakues in numeric format
  Data$Innings <- as.numeric(Data$Innings)
  Data$Runs <- as.numeric(Data$Runs)
  Data$Overs <- as.numeric(Data$Overs)
  Data$Wickets <- as.numeric(Data$Wickets)
  Data$Avg <- as.numeric(Data$Avg)
  Data$SR <- as.numeric(Data$SR)
  Data$'4s' <- as.numeric(Data$'4s')
  Data$'6s' <- as.numeric(Data$'6s')
  Data1 <- Data         # making copy of the data
  # removing last row of the data that is of totoal of the columns values
  Data <- Data[-c(row_number),]
  # now summarising the data to make it in useful way and easy to read
  Main <- Data %>%
    select(Year ,Runs ,Overs, Innings , Wickets) %>%
    summarise(Year = Year ,
              Runs_per_Inning = round({Runs/Innings},3),
              Wickets_per_Inning_X10 =Wickets/Innings *10 ,
    )
  xy <- Main[,c(1,2,3)]  # making another subset of the data to make plot
  xy <- melt(xy,"Year")

  # ploting the data
  ggp <- ggplot(xy , aes(x = Year , y = value , fill = variable )) +
    geom_bar(stat = "identity", width = 0.5,position = position_dodge(width = 0.7)) +
    labs(x = "Years" , y = "Number of Wickets Or Runs Per Year ",
         title =paste0(name,"'s Performance") )


  xz  <- Data1[c(row_number),]
  # taking output that are probablities plot and data set that we have cleaned
  some <- as.numeric(xz[1,5]/xz[1,3])
  output <- list(length(3))
  output[[1]] <- some/6
  output[[2]] <- Data
  output[[3]] <- ggp
  return(output)
}
```

For Batting Data -

```
Scrap_Bating_data <- function(name){
 # function will take input a name that will be a name of a cricket player
  # here we will replace spaces with + so we can form a link
  name1 <- gsub(pattern = " ", replacement = "+", name )
  # here we formed a link
  link <- paste0("http://www.cricmetric.com/playerstats.py?player=",
                 name1,"&role=batsman&format=ODI&groupby=year")

  # here we scraped a batting data from the website Cricmetirc
  data <- read_html(link) %>%
    html_table()
  # we will select first tibble that is of batting data that we targeted
  Data <- data[[1]]
  row_number <-dim(Data)[1]
  # removing commas from the digits
  Data$Runs <-  gsub(pattern = ",",replacement = "",x = Data$Runs)
  Data$Balls <- gsub(pattern = ",",replacement = "",x = Data$Balls)
  Data$'4s' <- gsub(pattern = ",",replacement = "",x = Data$'4s')


  # taking all the charahtere value as a numeric value so that r can recoganise it
  Data$Innings <- as.numeric(Data$Innings)
  Data$Runs <- as.numeric(Data$Runs)
  Data$Balls <- as.numeric(Data$Balls)
  Data$Outs <- as.numeric(Data$Outs)
  Data$Avg <- as.numeric(Data$Avg)
  Data$SR <- as.numeric(Data$SR)
  Data$HS <- as.numeric(Data$HS)
  Data$'50' <- as.numeric(Data$'50')
  Data$'100' <- as.numeric(Data$'100')
  Data$'4s' <- as.numeric(Data$'4s')
```

```
Data$'6s' <- as.numeric(Data$'6s')
Data1 <- Data                 # making a copy of the data that is cleaned
Data <- Data[-c(row_number),]  # removing the last row that is of sum of values

Main <- Data %>%            # now summarising data in useful way
  select(Year ,Runs ,Balls, Innings , Outs) %>%
  summarise(Year = Year ,
            Rate = round({Runs/Balls}*6, 2),
            Runs_per_Inning =Runs/Innings  ,
            Balls_per_Inning = Balls/Innings,
            Not_Out_Rate = {Innings-Outs}/Innings)

xy <- Main[,c(1,3,4)]        # making a subset of the data
# now we will melt the data so it will be easy to plot in a effective way
xy <- melt(xy,"Year")
text <- Main$Rate            # now the below 4_5 lines are of ploting
text <- append(text,Main$Not_Out_Rate)
text <- round(text,3)
Data5 <- tibble(xy,text)
# here we will define ggplot
ggp <- ggplot(Data5 , aes(x = Year , y = value , fill = variable )) +
  geom_bar(stat = "identity", width = 0.5,position = position_dodge(width = 0.7)) +
  geom_text(aes(label=text), vjust=1.5, colour="black", size=3) +
  labs(x = "Years" , y = "Number of Balls Or Runs ", title =paste0(name,"'s Performance") )


# now use the last row of the data set to make some usful things ,
# that will help us to get some idea about the runs rates and probablities
# of the player to hit runs
xz  <- Data1[c(row_number),]
Fours <- as.numeric(xz[1,11]/xz[1,4])
Sixs <- as.numeric(xz[1,12]/xz[1,4])

Fo <- as.numeric(xz[1,11])*4
Six <- as.numeric(xz[1,12])*6
Tot <- as.numeric(xz[1,3])
ToB <- as.numeric(xz[1,4])
pro <- {Tot-(Fo+Six)}/6

Ones <- pro*3/ToB
Twos <-  pro*2/ToB
Threes <-  pro/ToB
zeros <- 1-Ones - Twos - Threes - Fours - Sixs
x <- NULL
x[1] <- zeros  # here we am saving the probablities and plot and cleaned data
x[2] <- Ones
x[3] <- Twos
x[4] <- Threes
x[5] <- Fours
x[6] <-Sixs
output <- list(length(3))
output[[1]] <- x
output[[2]] <- Data
```

```
  output[[3]] <- ggp
  return(output)

}
```

Now using these two function data is scraped below is the code:

```
# for geting team names form the website this is done
Team <- read_html("https://www.thecricketer.com/Topics/mens-world-cup-2023/mens_cricket_world_cup_2023_s
  html_elements("#wl_f813f7 h3") %>%
  html_text2()

# below we scrape player name

Team_Player <- read_html("https://www.thecricketer.com/Topics/mens-world-cup-2023/mens_cricket_world_cup
  html_elements("#wl_f813f7 p") %>%
  html_text2()

# now we facing problem from here,
# player names are not correct come extra things are coming for cleaning
#the names we do the following things
Team_Player <- Team_Player[-c(1:3,14)]

Team_Player <- gsub("[(c)]","",Team_Player)
# we print all the player name and mannually correct them

Team_Player[1] <- "Hashmatullah Shahidi , Rahmanullah Gurbaz, Ibrahim Zadran,
  Riaz Hassan, Rahmat Shah, Najibullah Zadran, Mohammad Nabi, Ikram Alikhil,
  Azmatullah Omarzai, Rashid Khan, Mujeeb ur Rahman, Noor Ahmad, Fazalhaq Farooqi,
  Abdul Rahman, Naveen-ul-Haq"
Team_Player[2] <- "Pat Cummins, Sean Abbott, Alex Carey, Cameron Green,
  Josh Hazlewood, Travis Head, Josh Inglis, Marnus Labuschagne, Mitchell Marsh,
  Glenn Maxwell, Steven Smith, Mitchell Starc, Marcus Stoinis, David Warner,
  Adam Zampa"
Team_Player[3] <- "Shakib Al Hasan , Liton Das , Najmul Hossain Shanto,
  Tanzid Hasan, Towhid Hridoy, Mahmudullah, Mushfiqur Rahim, Mehidy Hasan,
  Mahedi Hasan, Tanzim Hasan Sakib, Nasum Ahmed, Shoriful Islam, Hasan Mahmud,
  Taskin Ahmed, Mustafizur Rahman"
Team_Player[4] <- "Jos Buttler , Moeen Ali, Gus Atkinson, Jonny Bairstow,
  Sam Curran, Liam Livingstone, Dawid Malan, Adil Rashid, Joe Root, Jason Roy,
  Ben Stokes, Reece Topley, David Willey, Mark Wood, Chris Woakes"
Team_Player[6] <- "SA Edwards , Max O'Dowd, Bas de Leede, Vikram Singh,
  Teja Nidamanuru, Paul van Meekeren, Colin Ackermann, Roelof van der Merwe,
  Logan van Beek, Aryan Dutt, Ryan Klein, Wesley Barresi, Saqib Zulfiqar,
  Shariz Ahmad, Michael Rippon"

# from here we create a country name list with respect to the player
Afghanistan <- strsplit(x =Team_Player[1] , split = ",",fixed = TRUE )
Afghanistan[[1]][15] <- "Naveen-ul-Haq"
Australia <- strsplit(x =Team_Player[2] , split = ",",fixed = TRUE )
Bangladesh <- strsplit(x =Team_Player[3] , split = ",",fixed = TRUE )
England <- strsplit(x =Team_Player[4] , split = ",",fixed = TRUE )
India <- strsplit(x =Team_Player[5] , split = ",",fixed = TRUE )
```

```
Netherlands <- strsplit(x =Team_Player[6] , split = ",",fixed = TRUE )
New_Zealand <- strsplit(x =Team_Player[7] , split = ",",fixed = TRUE )[[1]][-c(8)]
Pakistan <- strsplit(x =Team_Player[8] , split = ",",fixed = TRUE )
Pakistan[[1]][14] <- "Shaheen Shah Afridi"
Pakistan[[1]][15] <- "Usama Mir"
South_Africa <- strsplit(x =Team_Player[9] , split = ",",fixed = TRUE )
Sri_Lanka <- strsplit(x =Team_Player[10] , split = ",",fixed = TRUE )
Sri_Lanka[[1]][5]









#################################################################################

# Create an empty list to store the team players
Country <- vector("list", length = length(Team))

# Loop through each team
for(i in seq_along(Team)) {
  team_name <- Team[i]
  players <- strsplit(Team_Player[i], ", ")[[1]]

  # Assign the players to the list element with the team name as the list name
  Country[[team_name]] <- players
}



###############################################################################
# from here we get the list of player and now we scrap the data in a such way
that we can store then together for further use
country <- list()

for ( i in 5:length(Team)){
  team_name <- Team[i]
  for( j in 1:15){
    print(Country[[team_name]][j])
    country[[team_name]][[Country[[team_name]][j]]][["Batting"]] <-
      Scrap_Bating_data(Country[[team_name]][j])

  }
}
```

The GDP per Capital data was initially downloaded from - kaggle. It is cleaned to be used with the dataset collected, using this code.

```
gdp <- read.csv("Data Sets/gdp_per_capita.csv")
```

```r
for (i in 3:63) {
  colnames(gdp)[i] <- 1957 + i
}

# Changes country name as how we have saved in our cricket datas.

for (i in 1:length(gdp$Country.Name)) {
  if (gdp$Country.Name[i] == "United Kingdom") {
    gdp$Country.Name[i] = "England"
  }
  if (gdp$Country.Name[i] == "United Arab Emirates") {
    gdp$Country.Name[i] = "U.A.E."
  }
  if (gdp$Code[i] == "HKG") {
    gdp$Country.Name[i] = "Hong Kong"
  }
  if (gdp$Country.Name[i] == "United States") {
    gdp$Country.Name[i] = "U.S.A."
  }
}

save(gdp, file = "Data Sets/gdp_per_capita.csv")
```

The Happiness Index data was downloaded from - kaggle.

This code is used to get the Coordinates of the teams in a workable format.

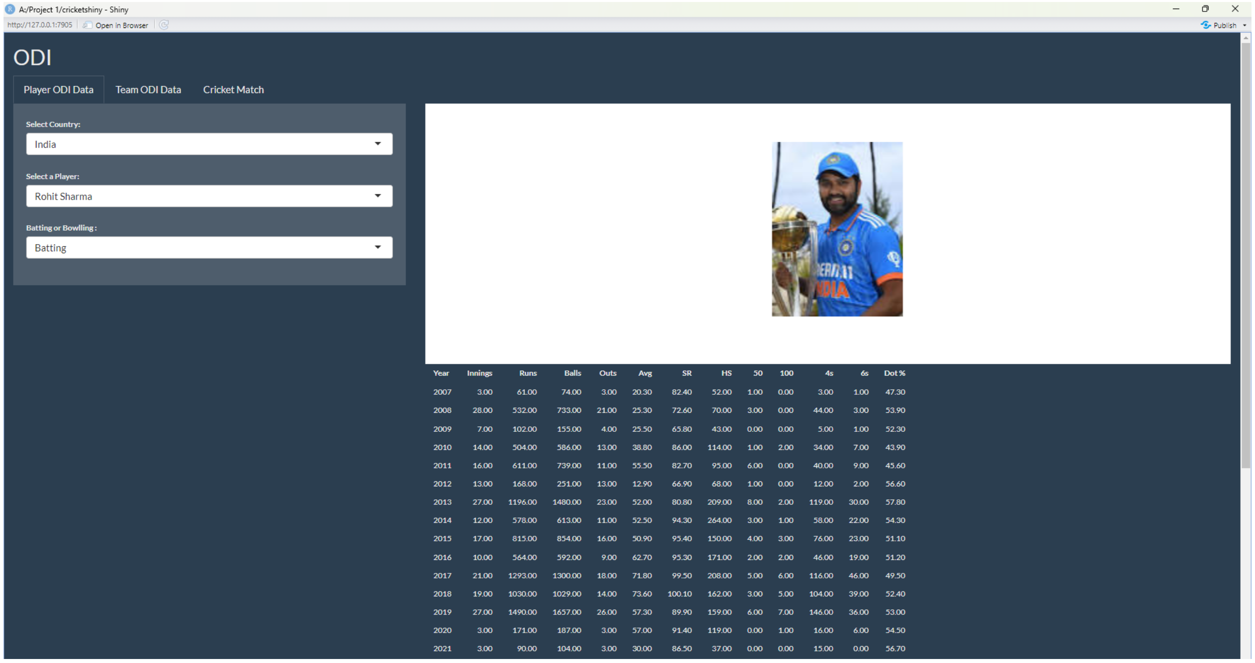
```r
# This gets all countries which form the West Indies.

west_indies <- c("Saint Lucia", "Anguilla", "Dominican Republic", "Guadeloupe",
  "Antigua", "Barbados", "Jamaica", "Trinidad", "Cayman Islands", "Bahamas", "Haiti",
  "Cuba", "Virgin Islands", "Martinique", "Saint Kitts", "Bermuda", "Saint Barthelemy")

# This is internally saved in R.

r <- map_data("world")
colnames(r)[5] <- "teams"

# This is used to change country names such that it aligns with how it's saved in our data set.
for (i in 1:length(r$teams)) {
  if (r$teams[i] %in% west_indies) {
    r$teams[i] <- "West Indies"
  }
  if (r$teams[i] == "UK") {
    r$teams[i] <- "England"
  }
  if (r$teams[i] == "United Arab Emirates") {
    r$teams[i] <- "U.A.E"
  }
  if (r$teams[i] == "USA") {
    r$teams[i] <- "U.S.A."
  }
  if (r$teams[i] == "Papua New Guinea") {
    r$teams[i] <- "P.N.G."
  }
}
coord <- r
save(coord, file = "Data Sets/Cricket_Coord.RData")
```

App Overview

The Shiny app consists of Three tabs, each dedicated to a specific aspect of cricket analysis. Now one by one deep explanation is below :

Player ODI Data

This tab allows users to explore individual player statistics for One Day International (ODI) matches. Users can select a country, choose a player, and specify whether they want to view batting or bowling statistics.



Select Country: India

Select a Player: Rohit Sharma

Batting or Bowling : Batting

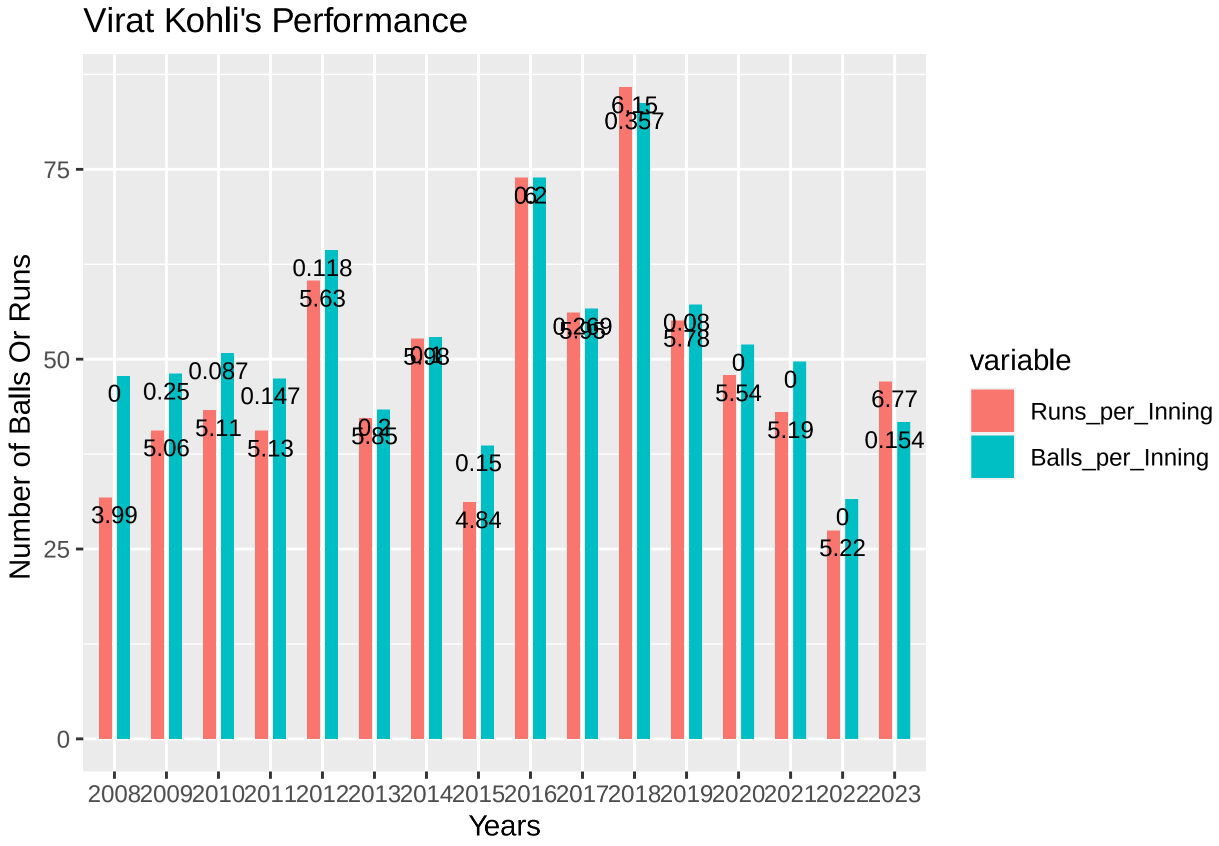
Year	Innings	Runs	Balls	Outs	Avg	SR	HS	50	100	4s	6s	Dot %
2007	3.00	61.00	74.00	3.00	20.30	82.40	52.00	1.00	0.00	3.00	1.00	47.30
2008	28.00	532.00	733.00	21.00	25.30	72.60	70.00	3.00	0.00	44.00	3.00	53.90
2009	7.00	102.00	155.00	4.00	25.50	65.80	43.00	0.00	0.00	5.00	1.00	52.30
2010	14.00	504.00	586.00	13.00	38.80	86.00	114.00	1.00	2.00	34.00	7.00	43.90
2011	16.00	611.00	739.00	11.00	55.50	82.70	95.00	6.00	0.00	40.00	9.00	45.60
2012	13.00	168.00	251.00	13.00	12.90	66.90	68.00	1.00	0.00	12.00	2.00	56.60
2013	27.00	1196.00	1480.00	23.00	52.00	80.80	209.00	8.00	2.00	119.00	30.00	57.80
2014	12.00	578.00	613.00	11.00	52.50	94.30	264.00	3.00	1.00	58.00	22.00	54.30
2015	17.00	815.00	854.00	16.00	50.90	95.40	150.00	4.00	3.00	76.00	23.00	51.10
2016	10.00	564.00	592.00	9.00	62.70	95.30	171.00	2.00	2.00	46.00	19.00	51.20
2017	21.00	1293.00	1300.00	18.00	71.80	99.50	208.00	5.00	6.00	116.00	46.00	49.50
2018	19.00	1030.00	1029.00	14.00	73.60	100.10	162.00	3.00	5.00	104.00	39.00	52.40
2019	27.00	1490.00	1657.00	26.00	57.30	89.90	159.00	6.00	7.00	146.00	36.00	53.00
2020	3.00	171.00	187.00	3.00	57.00	91.40	119.00	0.00	1.00	16.00	6.00	54.50
2021	3.00	90.00	104.00	3.00	30.00	86.50	37.00	0.00	0.00	15.00	0.00	56.70

- **Country Selection**: Users can choose a country from a dropdown menu.
- **Player Selection**: Once a country is selected, users can choose a player from a dynamically generated list of players for that country.
- **Batting/Bowling Selection**: Users can select whether they want to view batting or bowling statistics.
- **Player Image**: An image of the selected player is displayed.
- **Player Data Table**: Detailed statistics of the selected player are presented in a table format. This will be like:-

```
## [[1]]
## # A tibble: 16 x 13
##     Year  Innings  Runs Balls  Outs    Avg    SR    HS  ‘50‘ ‘100‘  ‘4s‘  ‘6s‘ ‘Dot %‘
##    <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
##  1 2008         5   159   239     5  31.8  66.5    54     1     0    21     1    71.5
##  2 2009         8   325   385     6  54.2  84.4   107     2     1    36     3    51.7
##  3 2010        23   995  1169    21  47.4  85.1   118     7     3    90     4    48.3
##  4 2011        34  1381  1614    29  47.6  85.6   117     8     4   127     7    46.8
##  5 2012        17  1026  1094    15  68.4  93.8   183     3     5    92     7    43.6
##  6 2013        30  1268  1300    24  52.8  97.5   115     7     4   138    20    48.1
##  7 2014        20  1054  1058    18  58.6  99.6   139     5     4    94    20    43.6
##  8 2015        20   623   773    17  36.6  80.6   138     1     2    44     8    49
##  9 2016        10   739   739     8  92.4 100     154     4     3    62     8    40.2
## 10 2017        26  1460  1473    19  76.8  99.1   131     7     6   136    22    42.8
## 11 2018        14  1202  1172     9 134.  103.    160     3     6   123    13    41.7
## 12 2019        25  1377  1429    23  59.9  96.4   123     7     5   133     8    41.8
## 13 2020         9   431   467     9  47.9  92.3    89     5     0    35     5    41.1
## 14 2021         3   129   149     3  43    86.6    66     2     0    10     1    41.6
## 15 2022        11   302   347    11  27.5  87     113     2     1    32     2    48.1
## 16 2023        13   612   542    11  55.6 113.    166     2     3    54    15    38.7
```

This table shows data of previous year - **Player Performance Plot**: A plot visualizing the player's performance is displayed. Like this :-

```
## [[1]]
```

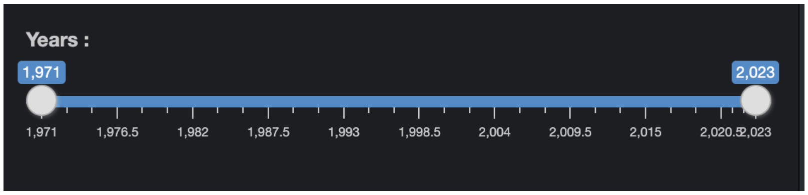


In this plot there are two coloured bars , rech are marked in right side of plot anyone can easily understand, the text on red coloured bar is of strike rate in that year and text on blue colured bar shows the rate that player out means in 1 inning what will be the probablity of player to br out is there .

Team ODI Data

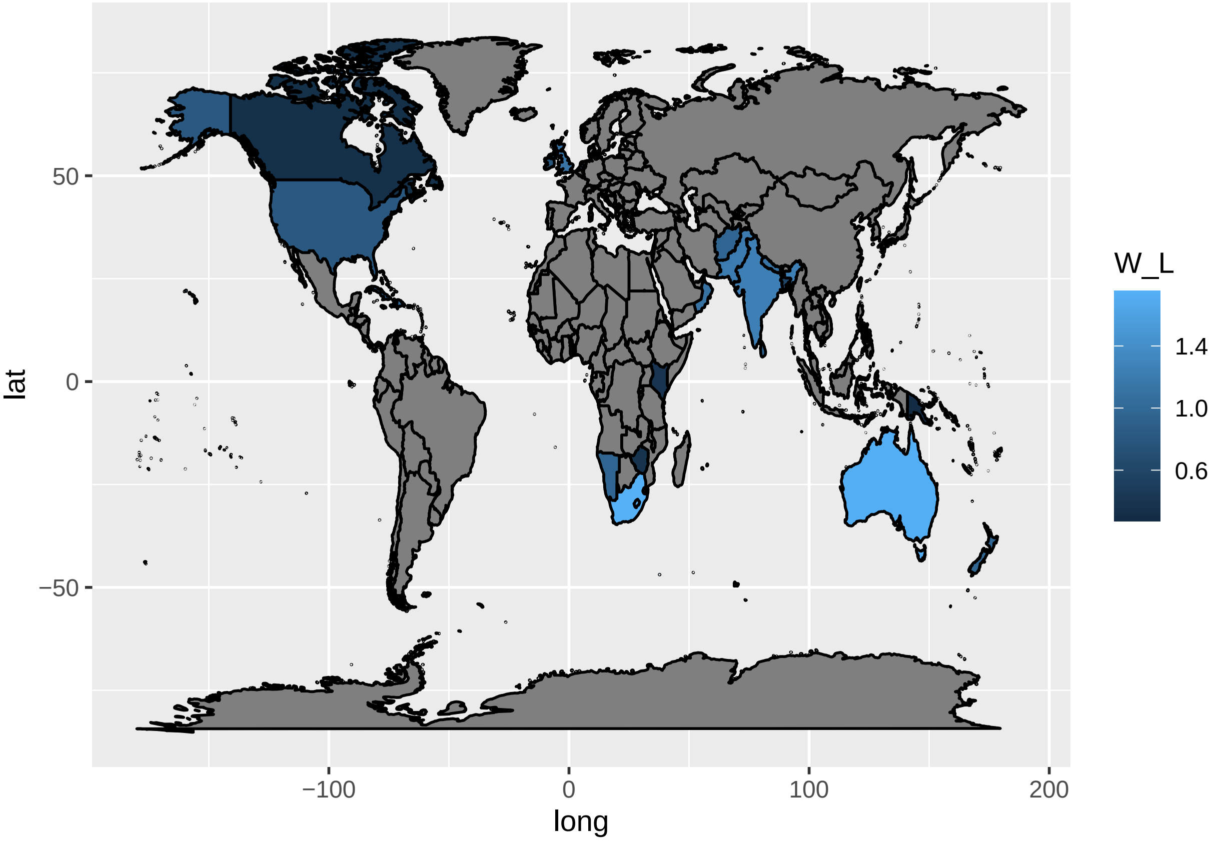
World View This sub-tab provides a global perspective of ODI match outcomes. Users can adjust the year range to view trends over time.

- **Year Selection Slider**: Users can adjust the range of years they want to analyze.
- **World Map Plot**: Users can visualise the Win to Loss Ratio of each team, color coded, in a time frame.



Years :

1,971 2,023

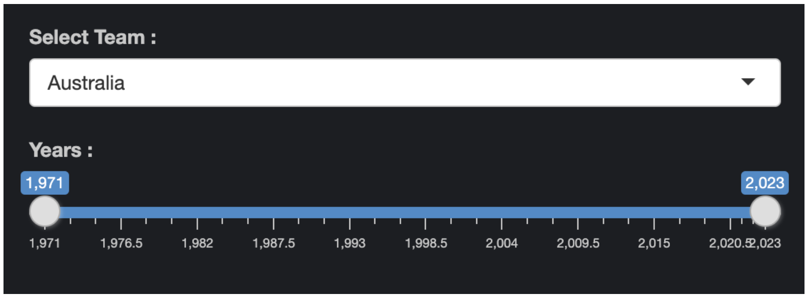


- **Table**: Users can see Win to Loss Ratio of all teams who have played ODI cricket in the specified timeframe.

```
## # A tibble: 23 x 2
##    Teams         `Win to Loss Ratio`
##    <chr>                       <dbl>
##  1 South Africa                 1.75
##  2 Australia                    1.72
##  3 India                        1.25
##  4 Pakistan                     1.2
##  5 England                      1.13
##  6 Nepal                        1.11
##  7 Oman                         1.10
##  8 West Indies                  1.02
##  9 New Zealand                  0.967
## 10 Afghanistan                  0.961
## # i 13 more rows
```

Team Overview This sub-tab allows users to select a specific team and view a summary of their performance over selected years.

- **Team Selection Dropdown**: Users can choose their ODI team from a dropdown menu.
- **Year Range Slider**: Users can specify the range of years they want to analyze.



Select Team :

Australia

Years :

1,971

2,023

- **Team Summary**:

```
##  Australia Score Opponent Score     Winning Score    Losing Score          Match Date
##  Min.    : 45.0  Length:947         Min.    : 45.0   Length:947         Min.   :1971-01-05
##  1st Qu.:178.0   Class :character   1st Qu.:171.0    Class :character   1st Qu.:1991-03-16
##  Median :217.0   Mode  :character   Median :209.0    Mode  :character   Median :2002-01-22
##  Mean   :216.8                      Mean   :208.1                       Mean   :2001-02-14
##  3rd Qu.:252.0                      3rd Qu.:245.0                       3rd Qu.:2010-06-19
##  Max.   :481.0                      Max.   :434.0                       Max.   :2023-10-20
```

- **Head-to-Head Table**: Users can see how the selected teams performs against other teams.

	Batting		Chasing		
Team	Won	Lost	Won	Lost	Win to Loss Ratio
Bangladesh	7	1	12	0	19.0000000
Zimbabwe	18	2	11	1	9.6666667
New Zealand	49	15	46	24	2.4358974
Pakistan	40	17	30	17	2.0588235
Sri Lanka	35	22	29	13	1.8285714
India	48	32	35	25	1.4561404
England	41	37	46	26	1.3809524
West Indies	53	35	23	26	1.2459016
South Africa	33	28	17	27	0.9090909
Canada	0	0	2	0	NA

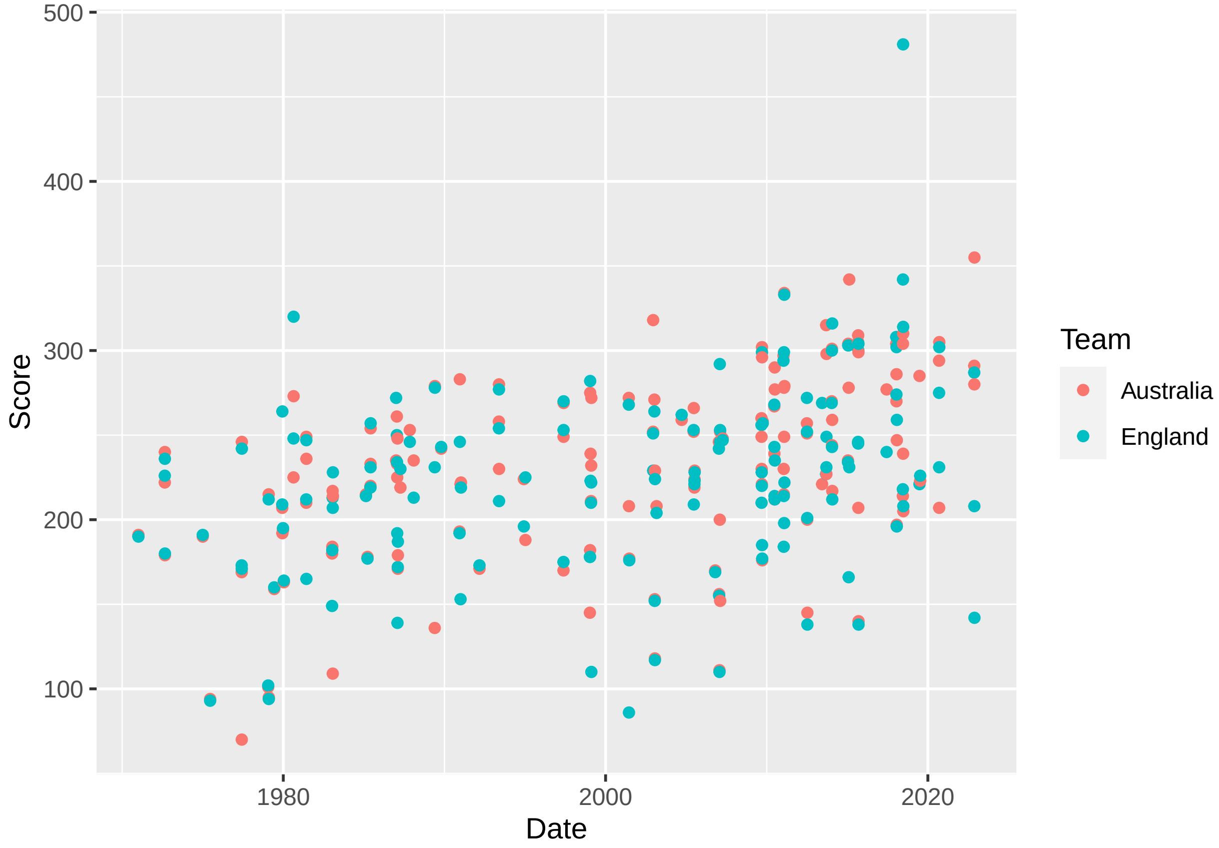
Kenya	2	0	3	0	NA
Scotland	3	0	2	0	NA
Netherlands	2	0	0	0	NA
Namibia	1	0	0	0	NA
U.S.A.	0	0	1	0	NA
ICC World XI	3	0	0	0	NA
Ireland	2	0	2	0	NA
Afghanistan	2	0	1	0	NA

Match Up This sub-tab enables users to compare the performance of two selected teams over a specified year range.

- **Team 1 Selection Dropdown**: Users can choose the first team from a dropdown menu.
- **Team 2 Selection Dropdown**: Users can choose the second team from a dropdown menu.
- **Year Range Slider**: Users can specify the range of years they want to analyze.
- **Match Summary**: A summary of the matches when the two teams are playing each other.

```
##  Australia Score England Score      Winning Score   Losing Score         Match Date
##  Min.   : 86.0   Length:150         Min.   : 70.0   Length:150         Min.   :1971-01-05
##  1st Qu.:192.0   Class :character   1st Qu.:185.8   Class :character   1st Qu.:1987-02-08
##  Median :226.0   Mode  :character   Median :219.5   Mode  :character   Median :2003-12-11
##  Mean   :224.8                      Mean   :215.6                      Mean   :2000-10-05
##  3rd Qu.:253.8                      3rd Qu.:249.0                      3rd Qu.:2012-07-09
##  Max.   :481.0                      Max.   :333.0                      Max.   :2022-11-22
```

- **Match Up Plot**: Users can see a plot of how the teams score when playing one another.

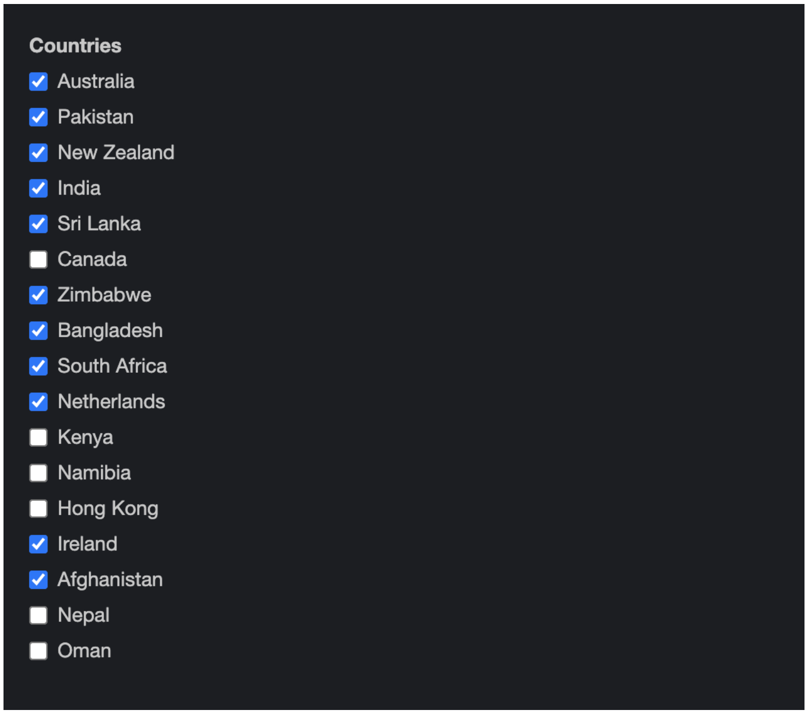


- **Match Up Table**: Users can see who won how many matches in their previous encounters, and in which innings.

```
##   Innings Australia England
## 1 Batting        41      26
## 2 Bowling        46      37
```

Happiness Index This sub-tab provides insights into the correlation between a country’s happiness index and its cricket team’s performance.

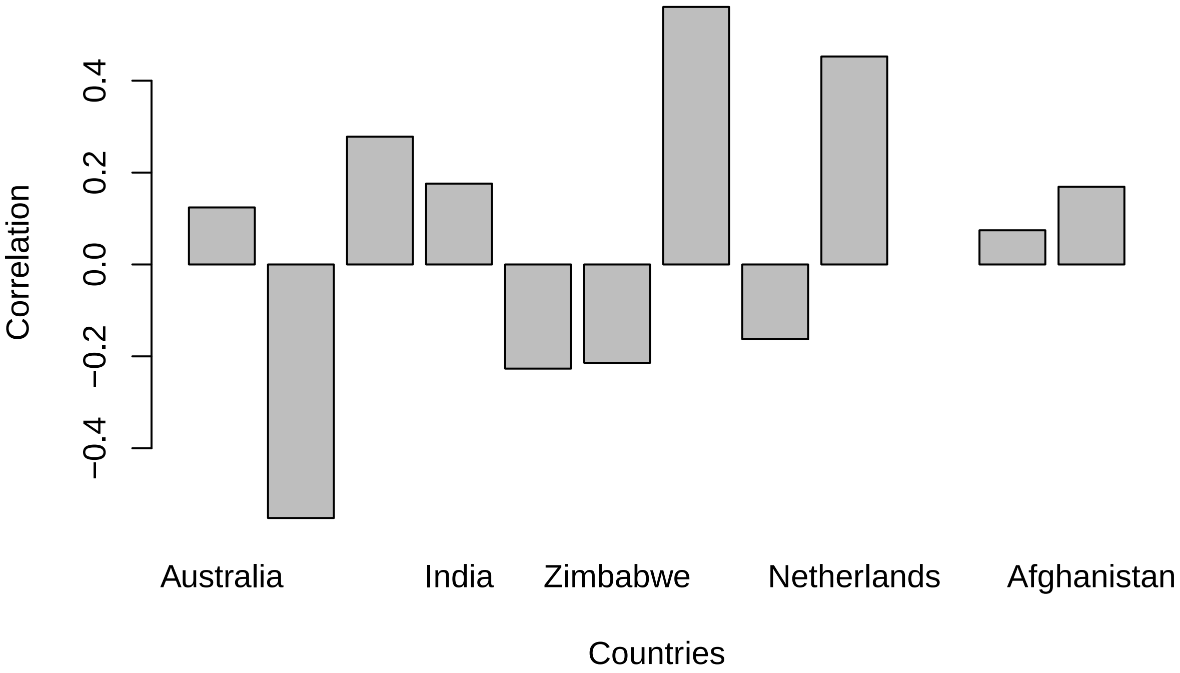
- **General View**: This view is to show the correlation of Happiness Index with the Win to Loss Ratio



Countries

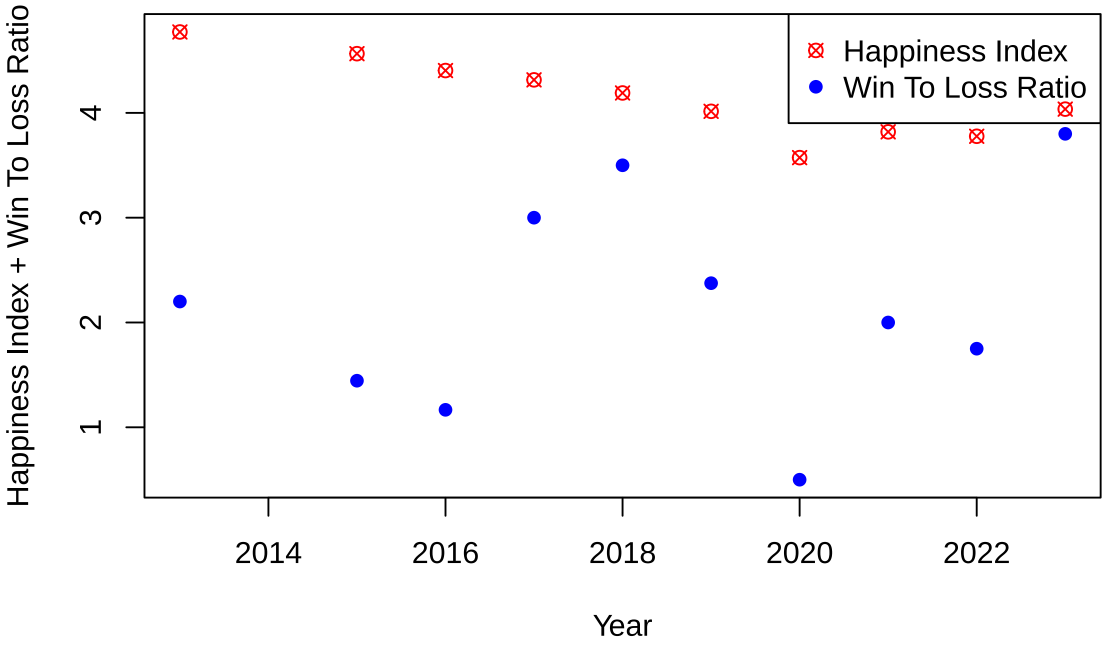
☑ Australia
☑ Pakistan
☑ New Zealand
☑ India
☑ Sri Lanka
□ Canada
☑ Zimbabwe
☑ Bangladesh
☑ South Africa
☑ Netherlands
□ Kenya
□ Namibia
□ Hong Kong
☑ Ireland
☑ Afghanistan
□ Nepal
□ Oman

of the selected countries.



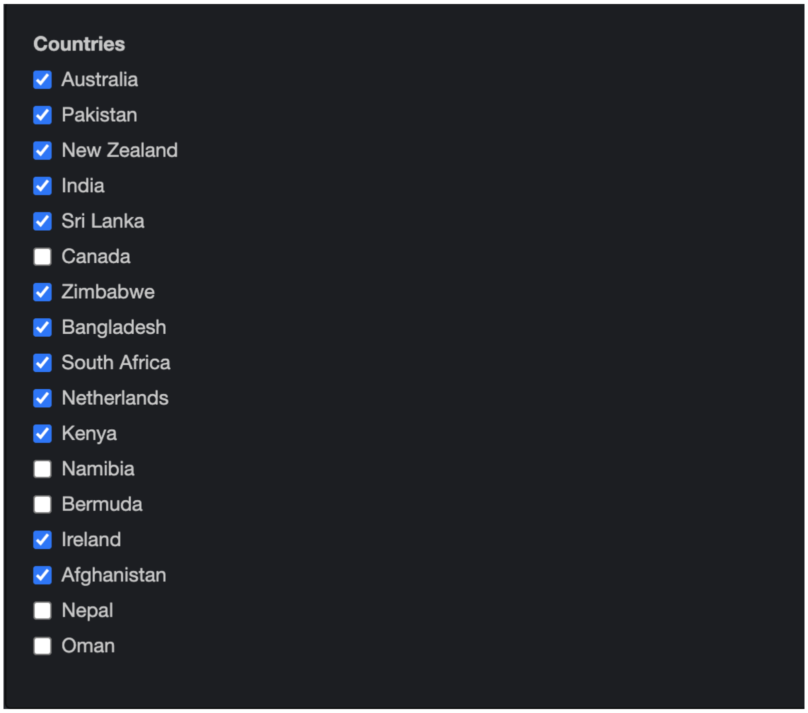
- **Country-Specific View**: This view can be used to visualise the variation of Happiness Index and Win to Loss Ratio of the selected country with year.

India



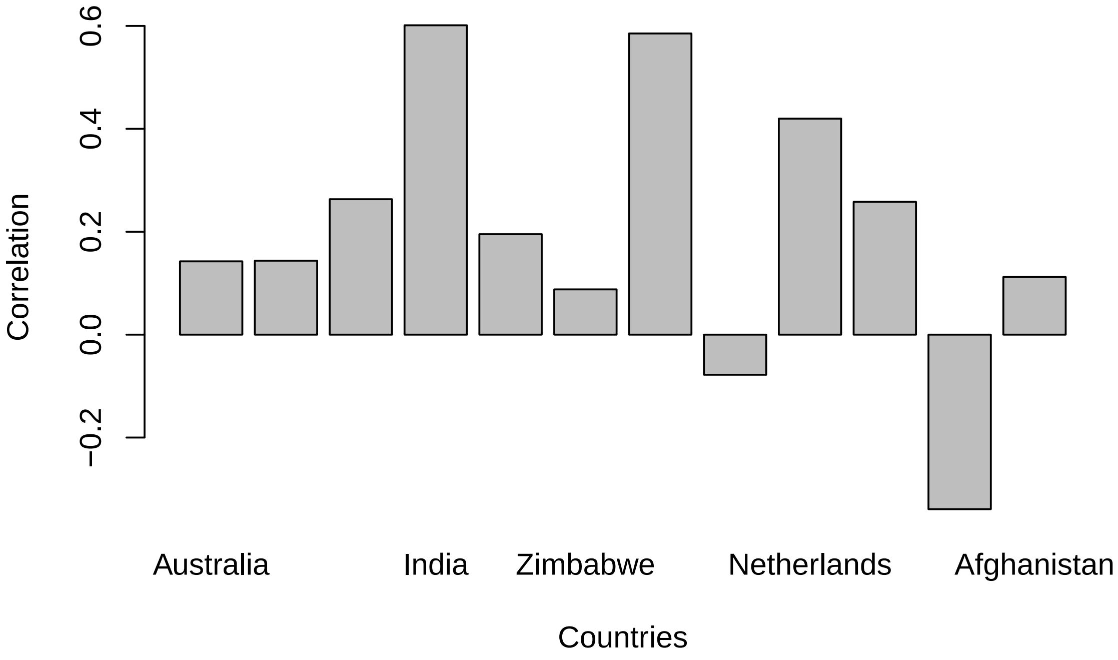
GDP This sub-tab explores the correlation between a country's Gross Domestic Product (GDP) and its cricket team's performance.

- **General View**: This view is to show the correlation of GDP with the Win to Loss Ratio of the selected countries.

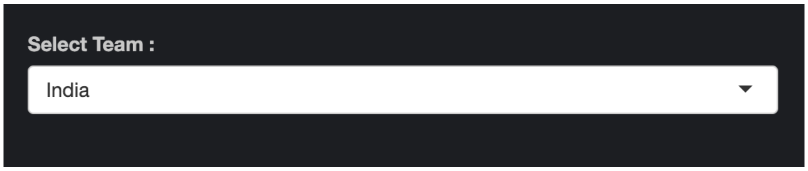


Countries

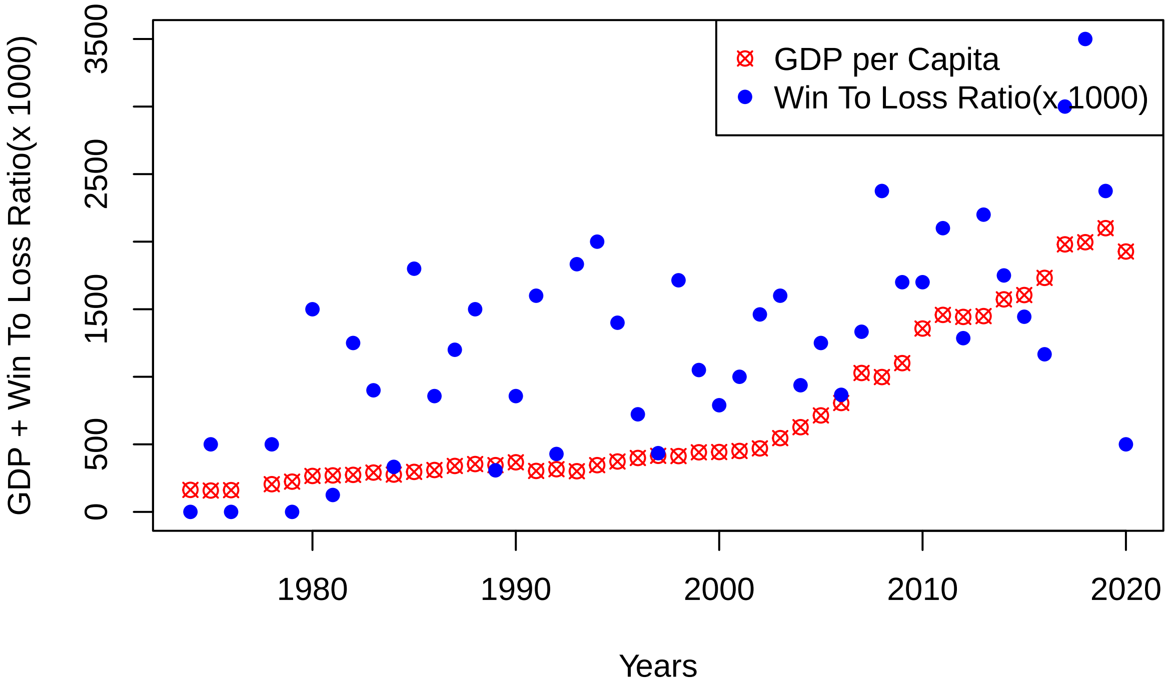
☑ Australia
☑ Pakistan
☑ New Zealand
☑ India
☑ Sri Lanka
☐ Canada
☑ Zimbabwe
☑ Bangladesh
☑ South Africa
☑ Netherlands
☑ Kenya
☐ Namibia
☐ Bermuda
☑ Ireland
☑ Afghanistan
☐ Nepal
☐ Oman



- **Country-Specific View**: This view can be used to visualise the variation of GDP and Win to Loss Ratio of the selected country with year.

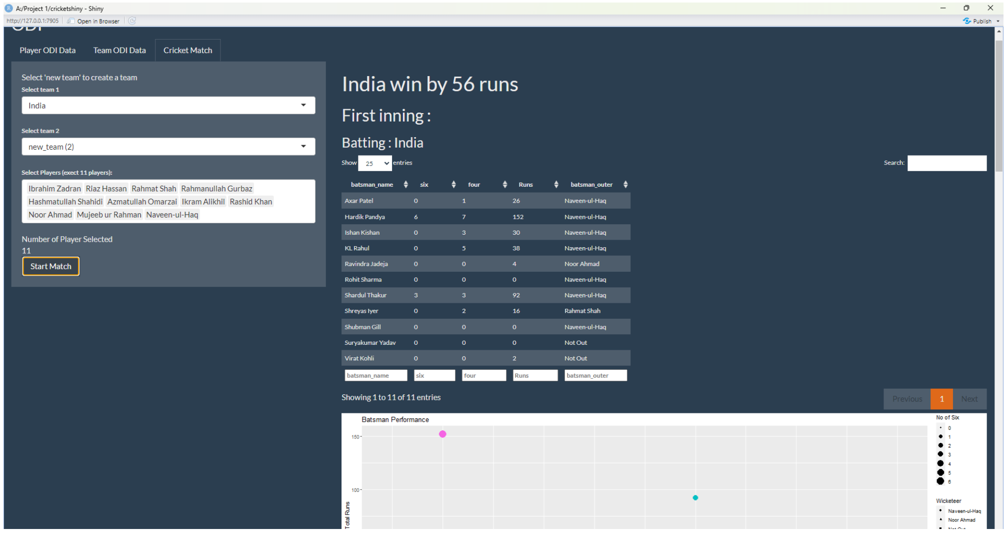


Select Team : India



Cricket Match

This tab allows users to simulate a cricket match between two teams. Users can select existing teams or create custom teams by choosing players.



Select 'new team' to create a team

Select team 1: India

Select team 2: new_team (2)

Select Players (exect 11 players): Ibrahim Zadran Riaz Hassan Rahmat Shah Rahmanullah Gurbaz Hashmatullah Shahidi Azmatullah Omarzai Ikram Alikhil Rashid Khan Noor Ahmad Mujeeb ur Rahman Naveen-ul-Haq

Number of Player Selected
11

India win by 56 runs

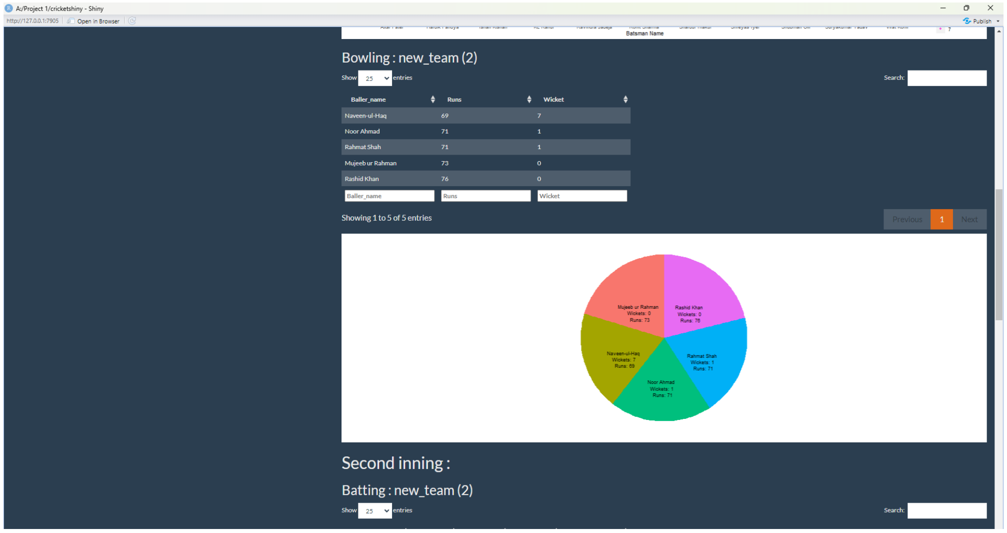
First inning :

Batting : India

Show 25 entries Search:

batsman_name	six	four	Runs	batsman_outer
Axar Patel	0	1	26	Naveen-ul-Haq
Hardik Pandya	6	7	152	Naveen-ul-Haq
Ishan Kishan	0	3	30	Naveen-ul-Haq
KL Rahul	0	5	38	Naveen-ul-Haq
Ravindra Jadeja	0	0	4	Noor Ahmad
Rohit Sharma	0	0	0	Naveen-ul-Haq
Shardul Thakur	3	3	92	Naveen-ul-Haq
Shreyas Iyer	0	2	16	Rahmat Shah
Shubman Gill	0	0	0	Naveen-ul-Haq
Suryakumar Yadav	0	0	0	Not Out
Virat Kohli	0	0	2	Not Out

Showing 1 to 11 of 11 entries



Bowling : new_team (2)

Show 25 entries Search:

Baller_name	Runs	Wicket
Naveen-ul-Haq	69	7
Noor Ahmad	71	1
Rahmat Shah	71	1
Mujeeb ur Rahman	73	0
Rashid Khan	76	0

Showing 1 to 5 of 5 entries

Second inning :

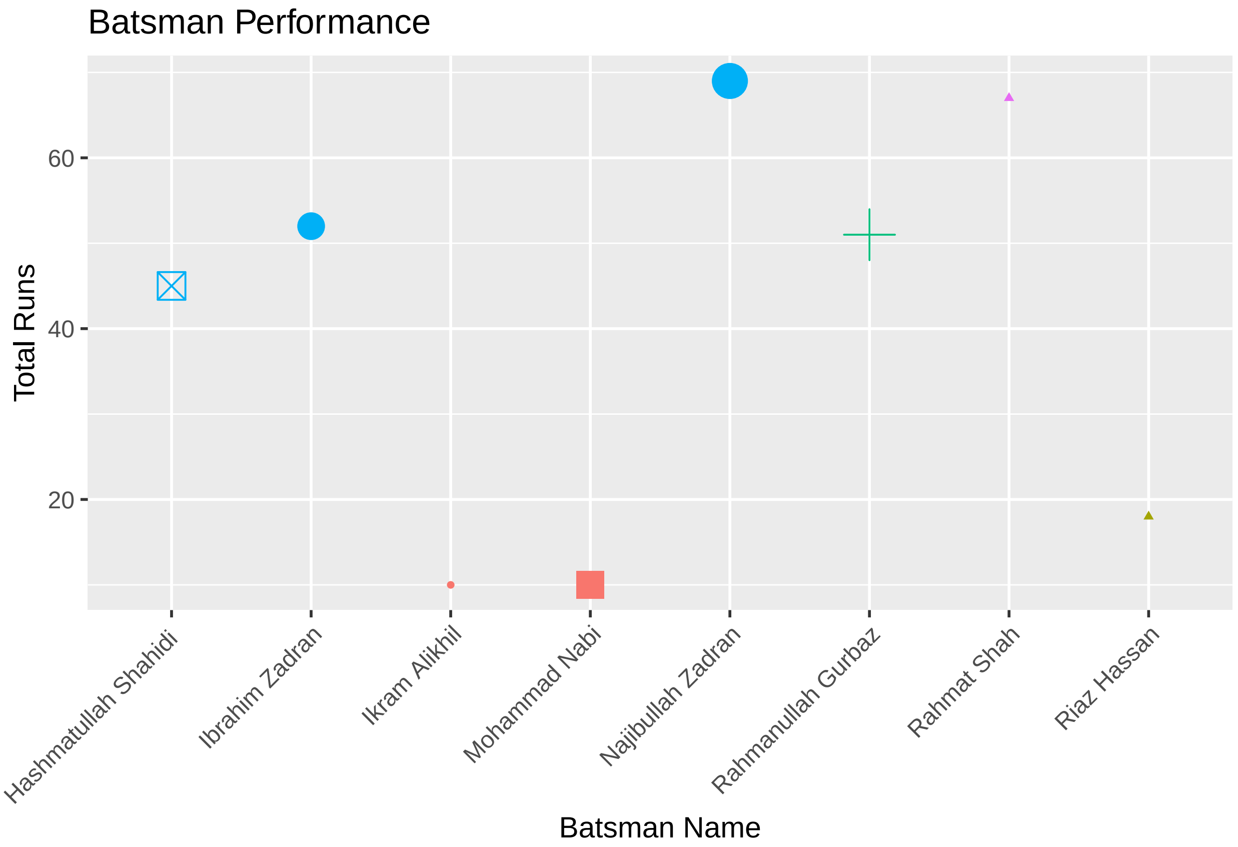
Batting : new_team (2)

Show 25 entries Search:

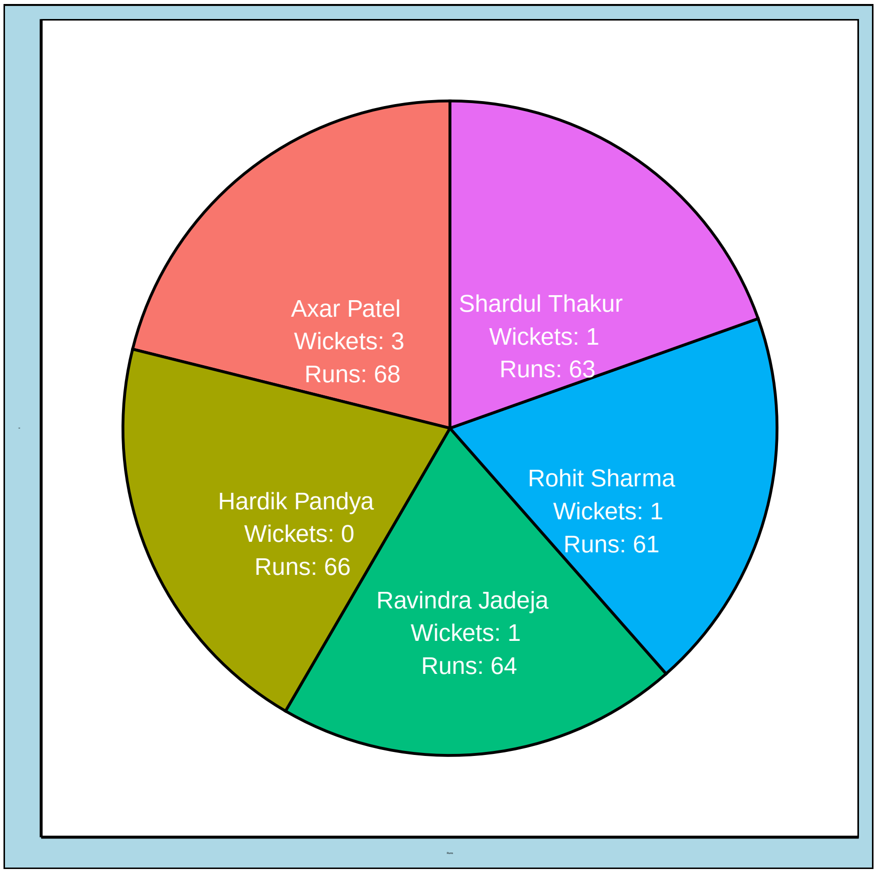
- **Team Selection**: Users can choose existing teams or create new teams by selecting players.
- **Player Selection**: Users can select players for the teams.
- **Start Match Button**: Initiates the match simulation.
- **Match Outcome**: Displays the results of the simulated match, including scores, batting, and bowling performances. Outcome will be of batting data table and there visualisation as follows :-

```
## [[1]]
## [[1]][[1]]
## # A tibble: 8 x 5
##   batsman_name               six  four  Runs batsman_outer
##   <chr>                    <dbl> <dbl> <dbl> <chr>
## 1 "Hashmatullah Shahidi "      1     4    45 "Shardul Thakur"
## 2 "Ibrahim Zadran"             1     4    52 "Axar Patel"
## 3 "Ikram Alikhil"              0     0    10 "Axar Patel"
## 4 "Mohammad Nabi"              1     0    10 "Ravindra Jadeja"
## 5 "Najibullah Zadran"          2     4    69 "Axar Patel"
## 6 "Rahmanullah Gurbaz"         2     3    51 "Rohit Sharma "
## 7 "Rahmat Shah"                0     5    67 "Not Out"
## 8 "Riaz Hassan"                0     1    18 "Not Out"
```

```
##
## [[1]][[2]]
## # A tibble: 5 x 3
##   Baller_name        Runs Wicket
##   <chr>             <dbl>  <dbl>
## 1 "Axar Patel"         68      3
## 2 "Ravindra Jadeja"    64      1
## 3 "Rohit Sharma "      61      1
## 4 "Shardul Thakur"     63      1
## 5 "Hardik Pandya"      66      0
##
##
## [[2]]
## [[2]][[1]]
## # A tibble: 5 x 5
##   batsman_name      six  four  Runs batsman_outer
##   <chr>           <dbl> <dbl> <dbl> <chr>
## 1 "KL Rahul"          0     6    97 Rashid Khan
## 2 "Rohit Sharma "     1     1    30 Ikram Alikhil
## 3 "Shreyas Iyer"      0     0     8 Rashid Khan
## 4 "Shubman Gill"      1     6    75 Not Out
## 5 "Virat Kohli"       0    13   157 Not Out
##
## [[2]][[2]]
## # A tibble: 5 x 3
##   Baller_name         Runs Wicket
##   <chr>              <dbl>  <dbl>
## 1 Rashid Khan           67      2
## 2 Ikram Alikhil         79      1
## 3 Azmatullah Omarzai    86      0
## 4 Mujeeb ur Rahman      81      0
## 5 Rahmat Shah           54      0
##
##
## [[3]]
## [1] "Team 2 win by 8 wickets"
```



This plot explains the ablove dataframe in which x axis is batsman name and y is no of runs and col shows the no 4 hit by a player and size of point shows the no of sixs , and shape show who our the player , this plot was made in plotly but pdf formet doesnt support so it is converted in ggplot . . .



this pie plot tell us about the ballers data , all is explined in the plot defined values are mentioned in the area covered.

we used the following code block to run the simulation for cricket match between two teams -

```
Cricket_Match <- function(Team_Data_1 , Team_Data_2,Team_name_1,Team_name_2){
# here will make a function that will semulates a cricket match

{
  ball <- NULL          # defing ball as null to get values in it

  for(i in 1:11){
  # for i in 1:11 means there are 11 players in a team but will select 5 best bowlers
    tryCatch({
      ball[i] <- Team_Data_2[[i]]$Bowling[[1]]
    }, error=function(e){})
  }
  ball[is.na(ball)] <- 0

  bowler_seq <- NULL
  for ( i in 1:5){
    bowler_seq[i] <- which(ball == max(ball), arr.ind = TRUE)[1]
    ball[bowler_seq[i]] <- 0
  }
```

```
Baller_name <- NULL
# defining some variable that we will use in the function
Ball_runs <- NULL
B_wicket <- NULL

Order <- rep(bowler_seq,10)
# this sequence is make to define the order of bowling


batsman_name <- NULL
Batsman_6 <- NULL
Batsman_4 <- NULL
batsman_run <- NULL
batsman_outer <- NULL



Runs <- 0
Ind <- 1

for(i in 1:50){
  # i is the no of ovwers here that will be 50 and it will take one by one all the values

  if(Ind == 11){
    break
    # this is for when 10 player got out and 11 player came to bets that
    #the match will be over and so we beak the loop at 11
  }
  over <- NULL           # this will bw used to take data of oveers in it


  Or <- Order[i]

  Wicket <- 0


  for(j in 1:6){
    # here j is the no of ball in  a over
    if(Ind == 11){
      # this is also same when 11 player come to bet the loop will break
      # and function will stop so that match will stop
      break
    }
    # We defined the values zero so that is this variable in function not
    # take any value then NA will not come zero will come if it takes value that it will be replaced
    batsman_run[6*(i-1) + j] <- 0
    Batsman_6[6*(i-1) + j] <- 0
    Batsman_4[6*(i-1) + j] <- 0
    # here we take a sample of out or not on based of probablities that
    # we calculated by data that was scraped
    Sam <- sample(c("Out","Not_Out"),1,replace = TRUE,prob = c(Team_Data_2[[Or]]$Bowling[[1]],1-Team_D

     # what if player will not out obeviously it will hit some runs or zero can be there
```

```r
     if(Sam == "Not_Out"){
#      print(Team_Data_1[[Ind]]$Batting[[1]])
      #this is the sample witch is based of players probablity to hit runs  according to previoud dat
      Run <- sample(c(0,1,2,3,4,6),1,replace = TRUE,prob = Team_Data_1[[Ind]]$Batting[[1]])


       Runs <- Runs + Run
      over[j] <- Run                      ##### here we will store the values in variables
      batsman_name[6*(i-1) + j] <- Team_name_1[Ind]
      batsman_run[6*(i-1) + j] <- Run
      # we know when there are 1 or 3 run strike changes that we have done here
      if(Run == 1 || Run == 3){


        nam <- Team_name_1[Ind]
        nam_ <- Team_name_1[Ind + 1]
        Team_name_1[Ind + 1] <- nam
        Team_name_1[Ind] <- nam_



        Runner <- Team_Data_1[[Ind]]$Batting
        Facer <-Team_Data_1[[Ind +1]]$Batting
        Team_Data_1[[Ind+1]]$Batting <- Runner
        Team_Data_1[[Ind]]$Batting <- Facer

      }


      if (Run == 4){                    # if 4 runs we will store it in a variable
        Batsman_4[6*(i-1)+j] <- 1
      }


      if (Run == 6){                       # this is for 6
        Batsman_6[6*(i-1)+j] <- 1
      }



    }

    else {

      batsman_name[6*(i-1) + j] <- Team_name_1[Ind]
      # what if player got out       # here we store wicket and bowlers name
      batsman_outer[6*(i-1)+j] <- Team_name_2[Or]
      Ind <- Ind + 1
      Wicket <- Wicket + 1
      over[j] <- 0
    }
    if (j == 6 ){
      # this is the last ball of the over,strike will change so we do it here
      nam <- Team_name_1[Ind]
      nam_ <- Team_name_1[Ind + 1]
      Team_name_1[Ind + 1] <- nam
      Team_name_1[Ind] <- nam_


      B_wicket[i] <- Wicket
      Baller_name[i] <- Team_name_2[Or]
      Ball_runs[i] <- sum(over)

      Runner <- Team_Data_1[[Ind]]$Batting
      Facer <-Team_Data_1[[Ind +1]]$Batting
      Team_Data_1[[Ind+1]]$Batting <- Runner
      Team_Data_1[[Ind]]$Batting <- Facer

      # Over_Data[[i]] <- Over
    }


   }




  }

  baller_data <- tibble(Baller_name, Ball_runs,B_wicket )
  # here we make tibble of all the data that a got from above loops
  baller_data <- baller_data %>%
    # here we arrange then in a useful way
    group_by(Baller_name) %>%
    summarise(Runs = sum(Ball_runs),
              Wicket = sum(B_wicket)) %>%
    arrange(desc(Wicket))


  batsman_data <- tibble( batsman_name, Batsman_6 ,Batsman_4,batsman_run )
  batsman_data <- batsman_data %>%
    # this is of batsman data
    group_by(batsman_name) %>%
    summarise(six = sum(Batsman_6),
              four = sum(Batsman_4),
              Runs = sum(batsman_run))

  batsman_outer <- as.vector(na.omit(batsman_outer))
  len <- dim(batsman_data)[1]
  # some na are introduced so we removed them

  if( length(batsman_outer) != len){
    for ( i in {length(batsman_outer) + 1}:len){
      batsman_outer[i] <- "Not Out"
    }
  }
```

```
  batsman_data$batsman_outer <- batsman_outer



  team_1_run <- sum(batsman_data$Runs)
  # savinf all useful thing in a list so we can take them in a return
  team_2_wicket <- sum(baller_data$Wicket)

  output_1 <- list(length(2))
  output_1[[1]] <- batsman_data
  output_1[[2]] <- baller_data
}
#####################
# Team 2 Bats
# here, the same process for inning 2
####################

{
  ball <- NULL
  for(i in 1:11){
    tryCatch({
      ball[i] <- Team_Data_1[[i]]$Bowling[[1]]
    }, error=function(e){})
  }
  ball[is.na(ball)] <- 0

  bowler_seq <- NULL
  for ( i in 1:5){
    bowler_seq[i] <- which(ball == max(ball), arr.ind = TRUE)[1]
    ball[bowler_seq[i]] <- 0
  }




  Baller_name <- NULL
  Ball_runs <- NULL
  B_wicket <- NULL

  Order <- rep(bowler_seq,10)


  batsman_name <- NULL
  Batsman_6 <- NULL
  Batsman_4 <- NULL
  batsman_run <- NULL
  batsman_outer <- NULL



  Runs <- 0
  Ind <- 1
```

```
for(i in 1:50){

  if(Ind == 11){
    break
  }
  over <- NULL


  Or <- Order[i]

  Wicket <- 0


  for(j in 1:6){
    if(Ind == 11){
      break
    }
    batsman_run[6*(i-1) + j] <- 0
    Batsman_6[6*(i-1) + j] <- 0
    Batsman_4[6*(i-1) + j] <- 0
    Sam <- sample(c("Out","Not_Out"),1,replace = TRUE,prob = c(Team_Data_1[[Or]]$Bowling[[1]],1-Tea
    if(Sam == "Not_Out"){
      #      print(Team_Data_2[[Ind]]$Batting[[1]])
      Run <- sample(c(0,1,2,3,4,6),1,replace = TRUE,prob = Team_Data_2[[Ind]]$Batting[[1]])


      Runs <- Runs + Run
      over[j] <- Run
      batsman_name[6*(i-1) + j] <- Team_name_2[Ind]
      batsman_run[6*(i-1) + j] <- Run
      if(Run == 1 || Run == 3){


        nam <- Team_name_2[Ind]
        nam_ <- Team_name_2[Ind + 1]
        Team_name_2[Ind + 1] <- nam
        Team_name_2[Ind] <- nam_



        Runner <- Team_Data_2[[Ind]]$Batting
        Facer <-Team_Data_2[[Ind +1]]$Batting
        Team_Data_2[[Ind+1]]$Batting <- Runner
        Team_Data_2[[Ind]]$Batting <- Facer

      }


      if (Run == 4){
        Batsman_4[6*(i-1)+j] <- 1
      }
      if (Run == 6){
        Batsman_6[6*(i-1)+j] <- 1
      }



    }

    else {

      batsman_name[6*(i-1) + j] <- Team_name_2[Ind]
      batsman_outer[6*(i-1)+j] <- Team_name_1[Or]
      Ind <- Ind + 1
      Wicket <- Wicket + 1
      over[j] <- 0
    }
    if (j == 6 ){

      nam <- Team_name_2[Ind]
      nam_ <- Team_name_2[Ind + 1]
      Team_name_2[Ind + 1] <- nam
      Team_name_2[Ind] <- nam_


      B_wicket[i] <- Wicket
      Baller_name[i] <- Team_name_1[Or]
      Ball_runs[i] <- sum(over)

      Runner <- Team_Data_2[[Ind]]$Batting
      Facer <-Team_Data_2[[Ind +1]]$Batting
      Team_Data_2[[Ind+1]]$Batting <- Runner
      Team_Data_2[[Ind]]$Batting <- Facer


    }


  }




}

baller_data <- tibble(Baller_name, Ball_runs,B_wicket )
baller_data <- baller_data %>%
  group_by(Baller_name) %>%
  summarise(Runs = sum(Ball_runs),
            Wicket = sum(B_wicket)) %>%
  arrange(desc(Wicket))


batsman_data <- tibble( batsman_name, Batsman_6 ,Batsman_4,batsman_run )
```

```
    batsman_data <- batsman_data %>%
      group_by(batsman_name) %>%
      summarise(six = sum(Batsman_6),
                four = sum(Batsman_4),
                Runs = sum(batsman_run))

    batsman_outer <- as.vector(na.omit(batsman_outer))
    len <- dim(batsman_data)[1]

    if( length(batsman_outer) != len){
      for ( i in {length(batsman_outer) + 1}:len){
        batsman_outer[i] <- "Not Out"
      }
    }
    batsman_data$batsman_outer <- batsman_outer


    team_2_run <- sum(batsman_data$Runs)
    team_1_wicket <- sum(baller_data$Wicket)

    output_2 <- list(length(2))
    output_2[[1]] <- batsman_data
    output_2[[2]] <- baller_data
  }

  output <- list(length(2))
  output[[1]] <- output_1
  output[[2]] <-output_2


  if (team_1_run == team_2_run){
    output[[3]] <- "Match Tie"
  }


  if ( team_1_run > team_2_run){
    by <- team_1_run - team_2_run
    output[[3]] <- paste( "Team 1 win by :" , by ,"runs")
  } else {
    by <- 11 - team_1_wicket
    output[[3]] <- paste( "Team 2 win by :" , by ,"wickets")
  }
    return(output)
}
```

Code Evaluation

The R code for the Shiny app demonstrates proficiency in utilizing the Shiny package, data manipulation, and visualization techniques. The app's functionality is well-organized, allowing for seamless interaction with the cricket data. You can find codes here.Click here

Interesting data-driven questions to think about

- What is the correlation between a country's performance and its GDP and its Happiness index?
- Can we get a visual representations of data using graphs and chart?
- Can we view a team's recent performance statistics, such as recent form and trends?
- How can we access a summary of each previous encounter between two sides in the app?
- Can we filter match history by specific teams, years, or tournaments?
- Is there a feature to watch or re-enact classic matches from the past?
- Can we find the win-loss ratio for different countries participating in the ODI World Cup?
- Are historical statistics and records available for past World Cup tournaments?
- Can we choose between default teams or create a custom team with my favorite players?
- What level of control and customization is available during the match simulation?

Conclusion

Access to individual player information throughout their career is a fantastic feature, especially for cricket fans who want to dig deep into player statistics and achievements. This can improve the overall cricketing experience. The ability to run live simulations with default teams or custom lineups is an appealing feature. It allows users to test strategies, create dream teams, and experience the excitement of live matches within the app. Displaying win-loss ratios for different countries in the ODI World Cup adds a competitive element to the app. Users can compare their customized team's performance to that of real-world national teams. Being capable to access summaries of previous encounters between two sides provides historical context. It allows users to research rivalries and historical events in cricket. Viewing a team's past performance, along with graphs, is a useful feature for those interested in in-depth analysis. It enables users to make data-driven decisions and understand team trends. It's interesting to see the correlation between GDP per capita, Happiness Index, and win-loss ratios of World Cup countries. It gives the app an educational and analytical dimension, appealing to users who are interested in the broader societal impact on sports performance.