# 

**MATH7012 - Programming for Data Science**

**Spring 2021**

**Data Science Assignment**

**Declaration:**

By including this statement, we the authors of this work, verify that:

* We hold a copy of this assignment that we can produce if the original is lost or damaged.
* We hereby certify that no part of this assignment/product has been copied from any other student’s work or from any other source except where due acknowledgement is made in the assignment.
* No part of this assignment/product has been written/produced for us by another person except where such collaboration has been authorised by the subject lecturer/tutor concerned.
* We are aware that this work maybe reproduced and submitted to plagiarism detection software programs for the purpose of detecting possible plagiarism (which may retain a copy on its database for future plagiarism checking).
* We hereby certify that we have read and understand what the School of Computing, Engineering and Mathematics defines as minor and substantial breaches of misconduct as outlined in the learning guide for this unit.

Mohit Mehndiratta, 20622275

Master of Data Science

14/10/2021

# Cricket Analysis

## Utility file description:

Below is the utility file that contains some reusable function that can be executed by other files.

A brief summary of all the functions in the **Utility.R** file is:

1. **fetch\_home\_team\_total\_runs** :- This function takes two input “match\_data” and “match\_number”. This function calculate and returns total runs scored by the home team for a given match number from the match data.
2. **fetch\_away\_team\_total\_runs** :- This function takes two input “match\_data” and “match\_number”. This function calculate and returns total runs scored by the away team for a given match number from the match data.
3. **fetch\_team\_names** :- This function takes one input “match\_data” and returns the team names that exists in the given match data.

**R code:**

####################################################################  
## A function to calculate home team total scores for a given match  
## Input: match\_data, match\_number  
## Output: numeric value for total runs scored by home team.  
####################################################################

fetch\_home\_team\_total\_runs <- function(match\_data, match\_number) {  
 home\_team <- match\_data[[match\_number]][[1]]  
 home\_team\_runs <- home\_team$runs  
 return(sum(home\_team\_runs))  
}  
  
####################################################################  
## A function to calculate away team total scores for a given match  
## Input: match\_data, match\_number  
## Output: numeric value for total runs scored by away team.  
####################################################################  
  
fetch\_away\_team\_total\_runs <- function(match\_data, match\_number){  
 away\_team <- match\_data[[match\_number]][[2]]  
 away\_team\_runs <- away\_team$runs  
 return(sum(away\_team\_runs))  
}

#############################################################  
## A function to get team names from the provided match data.  
## Input: match\_data  
## Output: a vector containing team names.  
#############################################################  
  
fetch\_team\_names <- function(match\_data){  
 teams <- c() # created an empty vector  
   
 # iterating through all the matches  
 for(i in 1:length(match\_data)){  
 team\_names <- names(match\_data[[i]]) # teams playing the match  
 # adding the team names to the teams vector.  
 teams <- c(teams,team\_names[1], team\_names[2])  
 }  
   
 # removing any duplicate team name and returning it.  
 return(unique(teams))  
}

## 

## Question 1: Write the code to compute the total runs scored by each team for each match. Run your code on the data and present the results in a table showing the match number, the home and away team names and their runs.

**R Code:**

library("rjson") # importing rjson library to load json data  
library("pander") # pander library to display the table.  
source("Utility.R") # source Utility.R file so there functions are available  
  
############################################################################  
## A function to calculate total runs scored by teams, match wise.  
## Input: match\_data - A list data for all match records  
## Output: A data frame containing match wise total runs scored by home and ## away team  
############################################################################

fetch\_team\_wise\_total\_scores <- function(match\_data) {  
 # creating an empty data frame to store the final results.  
 total\_runs <- data.frame(matrix(ncol=5,nrow=length(match\_data),   
 dimnames=list(NULL, c("Match",   
 "Home\_Team",   
 "Away\_Team",  
 "Home\_Team\_Runs",  
 "Away\_Team\_Runs"))))

# Iterating through each list and fetching home and away team scores  
 for(i in 1:length(match\_data)){  
 # calling the function from utility file.  
 home\_team\_total <- fetch\_home\_team\_total\_runs(match\_data, i)  
 # calling the function from utility file.  
 away\_team\_total <- fetch\_away\_team\_total\_runs(match\_data, i)  
   
 # Appending the home and away team total runs for a match in the final table.  
 total\_runs[i,]$Match = i  
 total\_runs[i,]$Home\_Team = names(match\_data[[i]][1])  
 total\_runs[i,]$Away\_Team = names(match\_data[[i]][2])  
 total\_runs[i,]$Home\_Team\_Runs = home\_team\_total  
 total\_runs[i,]$Away\_Team\_Runs = away\_team\_total  
 }  
   
 return(total\_runs)  
}

# importing match results data from the matchResults.json file.  
match\_results <- fromJSON(file = "matchResults.json")  
results <- fetch\_team\_wise\_total\_scores(match\_results)  
pander(results, caption = "Match wise total runs scored by home and away team:")

Match wise total runs scored by home and away team:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Match | Home\_Team | Away\_Team | Home\_Team\_Runs | Away\_Team\_Runs |
| 1 | Newcastle | Melbourne | 39 | 29 |
| 2 | Wollongong | Yarrawonga | 37 | 33 |
| 3 | Sydney | Geelong | 52 | 42 |
| 4 | Melbourne | Yarrawonga | 44 | 53 |
| 5 | Parramatta | Melbourne | 34 | 48 |
| 6 | Sydney | Wollongong | 50 | 37 |
| 7 | Melbourne | Aubury | 44 | 43 |
| 8 | Newcastle | Yarrawonga | 47 | 55 |
| 9 | Aubury | Yarrawonga | 37 | 45 |
| 10 | Sydney | Aubury | 47 | 39 |
| 11 | Sydney | Yarrawonga | 50 | 25 |
| 12 | Newcastle | Aubury | 35 | 42 |
| 13 | Parramatta | Aubury | 32 | 47 |
| 14 | Sydney | Melbourne | 40 | 51 |
| 15 | Wollongong | Newcastle | 45 | 50 |
| 16 | Geelong | Yarrawonga | 46 | 32 |
| 17 | Melbourne | Geelong | 43 | 44 |
| 18 | Parramatta | Newcastle | 44 | 46 |
| 19 | Parramatta | Yarrawonga | 40 | 40 |
| 20 | Wollongong | Aubury | 45 | 33 |
| 21 | Parramatta | Geelong | 40 | 46 |
| 22 | Newcastle | Geelong | 33 | 48 |
| 23 | Sydney | Newcastle | 32 | 26 |
| 24 | Wollongong | Melbourne | 50 | 47 |
| 25 | Sydney | Parramatta | 38 | 52 |
| 26 | Geelong | Aubury | 43 | 44 |
| 27 | Wollongong | Geelong | 41 | 46 |
| 28 | Parramatta | Wollongong | 52 | 40 |

### 

### Code Testing:

For Code testing, first verifying the output manually if we calculated the total runs correctly.

############################################################################  
############################## Code Testing! ###############################  
  
# Manual verification  
  
# getting first match's home team total runs  
runs <- sum(match\_results[[1]][[1]]$runs)  
# condition if the actual and observed results are equal.  
if(results[1,]$Home\_Team\_Runs == runs){  
 print("Test Case Passed.")  
} else {  
 print("Test Case Failed.")  
}

## [1] "Test Case Passed."

Now, passing a new file with new data created from IPL(Indian Premier League) match list.

A brief about the data, the data is about the match details of IPL tournament given in the form JSON format.

There are 4 teams: Punjab, Delhi, Mumbai and Chennai.

# testing with new data  
ipl\_match\_results <- fromJSON(file = "IPLMatchResults.json")  
  
testing\_results <- fetch\_team\_wise\_total\_scores(ipl\_match\_results)  
pander(testing\_results)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Match | Home\_Team | Away\_Team | Home\_Team\_Runs | Away\_Team\_Runs |
| 1 | Punjab | Mumbai | 39 | 29 |
| 2 | Chennai | Delhi | 37 | 33 |
| 3 | Punjab | Chennai | 52 | 42 |
| 4 | Delhi | Punjab | 53 | 44 |
| 5 | Mumbai | Chennai | 43 | 43 |
| 6 | Delhi | Mumbai | 41 | 46 |

## Question 2: The winner of each match is the team who scored the most runs. Each team scores 3 points for a win and 1 point for a draw and the team with the most points at the end of the season is the season winner. Write the code to compute each team’s points after the first 14 matches and the 28 matches. Provide the results in a table (two tables, one table for each set of results), ordering teams by their points (i.e. a results ladder).

**R Code:**

library("rjson") # importing rjson library to load json data  
library("pander") # pander library to display the table.  
library("tidyverse") # importing tidyverse library

source("Utility.R") # source Utility.R file so there functions are available  
  
############################################################################  
## A function to calculate and update team scores for a particular match.  
## input: team\_score : a data frame of points table  
## match\_details: a data frame of match details like team names and ## their total runs.  
## output: returns the updated points table data frame - "team\_score"  
############################################################################

calculate\_and\_update\_scores <- function(team\_scores, match\_details){

# fetching existing home team scores for this match  
 home\_team\_scores <- team\_scores[which(team\_scores$teams == match\_details$home\_team\_name),]$scores

# fetching existing away team scores for this match  
 away\_team\_scores <- team\_scores[which(team\_scores$teams == match\_details$away\_team\_name),]$scores  
   
 # below conditions adds a score of 3 if a team(home/away) wins else adds a score of 1 if they tie  
 if(match\_details$home\_team\_runs > match\_details$away\_team\_runs){  
 home\_team\_scores <- home\_team\_scores + 3  
 }else if(match\_details$home\_team\_runs == match\_details$away\_team\_runs){  
 home\_team\_scores <- home\_team\_scores + 1  
 away\_team\_scores <- away\_team\_scores + 1  
 }else{  
 away\_team\_scores <- away\_team\_scores + 3  
 }  
   
 # updating the home team scores.  
 team\_scores[which(team\_scores$teams == match\_details$home\_team\_name),]$scores = home\_team\_scores  
 # updating the away team scores.  
 team\_scores[which(team\_scores$teams == match\_details$away\_team\_name),]$scores = away\_team\_scores  
   
 return(team\_scores)  
}  
  
#############################################################################  
## A function that will return the points table for the given match data upto ## the specified number of matches.  
## input: match\_data, number\_of\_matches.  
## output: returns a data frame of points table.  
#############################################################################

compute\_points\_table <- function(match\_data, number\_of\_matches){  
 # fetching team names  
 team\_names <- fetch\_team\_names(match\_data)  
 # creating an empty data frame to store the team wise scores.  
 team\_scores <- data.frame(teams = team\_names,  
 scores = rep(0, length(team\_names)))  
   
 # calculating points table for given number of matches  
 for(i in 1:number\_of\_matches){  
 # calling the function from utility file.  
 home\_team\_runs <- fetch\_home\_team\_total\_runs(match\_data, i)  
 # calling the function from utility file.  
 away\_team\_runs <- fetch\_away\_team\_total\_runs(match\_data, i)  
   
 # a data frame which has match details like team names and their total runs  
 match\_details <- data.frame(home\_team\_name = names(match\_data[[i]][1]),  
 away\_team\_name = names(match\_data[[i]][2]),  
 home\_team\_runs = home\_team\_runs,  
 away\_team\_runs = away\_team\_runs)  
   
 # calling method to calculate and update points table.  
 team\_scores <- calculate\_and\_update\_scores(team\_scores, match\_details)  
 }  
 # sort the teams from high to low scores.  
 team\_scores <- team\_scores %>% arrange(desc(scores))  
 return(team\_scores)  
}  
# importing match results data from the matchResults.json file.  
match\_results <- fromJSON(file = "matchResults.json")  
  
# calculating points table for first 14 matches  
result1 <- compute\_points\_table(match\_results, 14)  
pander(result1, caption = "Points table after first 14 matches")

Points table after first 14 matches

|  |  |
| --- | --- |
| teams | scores |
| Sydney | 12 |
| Melbourne | 9 |
| Yarrawonga | 9 |
| Aubury | 6 |
| Newcastle | 3 |
| Wollongong | 3 |
| Geelong | 0 |
| Parramatta | 0 |

# calculating points table for all 28 matches  
result2 <- compute\_points\_table(match\_results, 28)  
pander(result2, caption = "Points table after 28 matches")

Points table after 28 matches

|  |  |
| --- | --- |
| teams | scores |
| Sydney | 15 |
| Geelong | 15 |
| Yarrawonga | 10 |
| Newcastle | 9 |
| Melbourne | 9 |
| Wollongong | 9 |
| Aubury | 9 |
| Parramatta | 7 |

### Code Testing:

Testing Code with the new IPL(Indian Premier League) matches.

############################################################################  
############################## Code Testing! ###############################  
  
ipl\_match\_results <- fromJSON(file = "IPLMatchResults.json")  
# calculating points table for first 3 matches  
testing\_results1 <- compute\_points\_table(ipl\_match\_results, 3)  
pander(testing\_results1, caption = "Points table after first 3 matches")

Points table after first 3 matches

|  |  |
| --- | --- |
| teams | scores |
| Punjab | 6 |
| Chennai | 3 |
| Mumbai | 0 |
| Delhi | 0 |

# calculating points table for all 6 matches  
testing\_results2 <- compute\_points\_table(ipl\_match\_results, 6)  
pander(testing\_results2, caption = "Points table after first 6 matches")

Points table after first 6 matches

|  |  |
| --- | --- |
| teams | scores |
| Punjab | 6 |
| Mumbai | 4 |
| Chennai | 4 |
| Delhi | 3 |

## Question 3: An investigation into home game advantage in planned for the coming months. Write a function that takes the file “matchResults.json” as its only argument, and returns a count of the number of home games each team has had and the total number of games each team has played in. Provide these results in a data frame. Run the function on the provided “matchResults.json” and show the resulting table output.

**R Code:**

library("rjson") # importing rjson library to load json data  
library("pander") # pander library to display the table.  
source("Utility.R") # source Utility.R file so there functions are available

################################################################  
## A function to get home matches of the given team.  
## input: match\_data, team\_name  
## output: returns the number of home matches played by the team  
################################################################

fetch\_home\_matches <- function(match\_data, team\_name){  
 count <- 0 # initial count  
 for(i in 1:length(match\_data)){  
 team\_names <- names(match\_data[[i]])  
 if(team\_names[1] == team\_name){  
 count <- count + 1  
 }  
 }  
 return(count)  
}  
  
####################################################################  
## A function to get total matches of the given team.  
## input: match\_data, team\_name  
## output: returns the number of total matches played by the team  
####################################################################

fetch\_total\_matches <- function(match\_data, team\_name){  
 count <- 0 # initial count  
 for(i in 1:length(match\_data)){  
 team\_names <- names(match\_data[[i]])  
 if(team\_name %in% team\_names){  
 count <- count + 1  
 }  
 }  
 return(count)  
}  
  
############################################################################  
## A function that gives home and total matches played team wise.  
## input: file\_name: a JSON file which contains match data.  
## output: returns a data frame containing team wise home and total matches.  
############################################################################

fetch\_matches\_played <- function(file\_name){  
 # importing match data from the given file.  
 match\_results <- fromJSON(file = file\_name)

# get team names from the match data  
 team\_names <- fetch\_team\_names(match\_results)  
   
 # an empty data frame that contains home and total matches count team wise  
 matches\_played <- data.frame(home\_matches = rep(0, length(team\_names)),  
 total\_matches = rep(0, length(team\_names)))  
 rownames(matches\_played) <- team\_names  
   
 # iterating through list of teams and identifying their home and total matches and assigning to the "matches\_played" data frame.  
 for(team\_name in team\_names){  
 home\_matches <- fetch\_home\_matches(match\_results, team\_name)  
 total\_matches <- fetch\_total\_matches(match\_results, team\_name)  
   
 matches\_played[team\_name,]$home\_matches <- home\_matches  
 matches\_played[team\_name,]$total\_matches <- total\_matches  
 }  
   
 return(matches\_played)  
   
}  
  
team\_matches <- fetch\_matches\_played("matchResults.json")  
pander(team\_matches)

|  |  |  |
| --- | --- | --- |
|  | home\_matches | total\_matches |
| **Newcastle** | 4 | 7 |
| **Melbourne** | 3 | 7 |
| **Wollongong** | 5 | 7 |
| **Yarrawonga** | 0 | 7 |
| **Sydney** | 7 | 7 |
| **Geelong** | 2 | 7 |
| **Parramatta** | 6 | 7 |
| **Aubury** | 1 | 7 |

### Code Testing:

Testing Code with the new IPL(Indian Premier League) matches.

############################################################################  
############################## Code Testing! ###############################  
  
testing\_results3 <- fetch\_matches\_played("IPLMatchResults.json")  
pander(testing\_results3)

|  |  |  |
| --- | --- | --- |
|  | home\_matches | total\_matches |
| **Punjab** | 2 | 3 |
| **Mumbai** | 1 | 3 |
| **Chennai** | 1 | 3 |
| **Delhi** | 2 | 3 |

## Question 4: Provide a table containing the set of player names of players that played for Parramatta, and their average number of Wickets per game. This table is to be used to provide a bowling award. Order the table from highest to lowest average Wickets and split ties using the number of matches (e.g. if two players have an average of 2, but one has played more matches, then the player who has played more matches should ordered in front of the other).

**R Code:**

library("rjson") # importing rjson library to load json data  
library("pander") # pander library to display the table.  
library("tidyverse") # importing tidyverse library  
  
####################################################################  
## A function to get the match details of the given team.  
## input: match\_data, team\_name  
## output: returns a data frame containing match details of the team.  
####################################################################

get\_team\_matches <- function(match\_data, team\_name){  
 indexes <- c()  
 for(i in 1:length(match\_data)){  
 team\_names <- names(match\_data[[i]])  
 if(team\_name %in% team\_names){  
 indexes <- c(indexes, i)  
 }  
 }  
 return(match\_data[indexes])  
}  
  
##############################################################  
## A function get player names from a given set of player Ids.  
## input: player\_names\_list, playerIds  
## output: returns a vector of player names..  
##############################################################

get\_player\_names\_by\_id <- function(player\_names\_list, playerIds){  
 playerNames <- c()  
 for(playerId in playerIds){  
 name <- player\_names\_list[player\_names\_list$ID == playerId,]$playerNames  
 playerNames <- c(playerNames, name)  
 }  
 return(playerNames)  
}

###########################################################################  
## A function to calculate player details of a given team, such as their ## total wickets taken,total matches played and average wickets per match.  
## input: match\_data - a list of all the matches   
## team\_name - a team name to fetch player details for.  
## teams\_info - a list of all teams and their players.  
## player\_nams - a data frame of all player names and their id.  
## output: returns a data frame containing player's total wickets taken and ## matches played for the given team.  
###########################################################################  
calculate\_player\_details <- function(match\_data, team\_name, teams\_info, player\_nams){  
 # getting players of the given team  
 player\_ids <- sort(teams\_info[[team\_name]])  
 player\_names <- (player\_nams %>% filter(ID %in% player\_ids))$playerNames  
   
 # an empty data frame for player details with wickets and matches fields.  
 player\_details <- data.frame(playerNames = player\_names,  
 wickets = rep(0, length(player\_names)),  
 matches = rep(0, length(player\_names)))  
 # iterating over match data to fetch player details that played for given team in the season.  
 for(i in 1:length(match\_data)){  
 players <- match\_data[[i]][[team\_name]]$playerID  
 wickets <- match\_data[[i]][[team\_name]]$wickets  
 players <- get\_player\_names\_by\_id(player\_nams, players)  
   
 # iterating over the players played for the given team in a particular match and fetching their total wickets taken, plus incrementing their total matches and updating them into original data frame of "player\_details".   
 for(i in 1:length(players)){  
 total\_wickets <- player\_details[player\_details$playerNames %in% players[i],]$wickets + wickets[i]  
   
 player\_details[player\_details$playerNames %in% players[i],]$wickets <- total\_wickets  
   
 total\_matches <- player\_details[player\_details$playerNames %in% players[i],]$matches + 1  
   
 player\_details[player\_details$playerNames %in% players[i],]$matches <- total\_matches  
 }  
 }  
   
 # adding a new column averageWickets for average wickets taken per match.  
 player\_details <- player\_details %>%   
 mutate(averageWickets = wickets / matches)  
 # removing the rows which have null averageWickets(scenario: if a player didn't played any match)  
 player\_details <- na.omit(player\_details)  
 return(player\_details)  
}

#############################################################################  
## A function to fetch bowlers average wickets taken per match  
## input: match\_data - a list of all the matches   
## team\_name - a team name to fetch player details for.  
## teams\_members - a list of all teams and their players.  
## player\_names - a data frame of all player names and their id.  
## output: returns a data frame containing player's average wickets per match  
#############################################################################  
  
fetch\_bowlers\_average\_wickets <- function(match\_data, team\_name, team\_members, player\_names){  
 # fetching all the matches played by provided team.  
 team\_matches <- get\_team\_matches(match\_data, team\_name)  
 # calculate player details of the given team.  
 player\_details <- calculate\_player\_details(team\_matches, team\_name,   
 team\_members, player\_names)  
 # sorting the bowlers by highest average wickets to lowest average wickets.  
 player\_details <- player\_details %>% arrange(desc(averageWickets), desc(matches))  
   
 # subset the original data frame to get players and their average wickets per match only.  
 players\_average\_wickets <- subset(player\_details, select = c(playerNames, averageWickets))  
   
 return(players\_average\_wickets)  
}  
  
# loading match data from matchResults.json file  
match\_results <- fromJSON(file = "matchResults.json")  
# loading team members details from teamMembers.json file  
team\_members <- fromJSON(file = "teamMembers.json")  
# loading player names from playerNames.csv file  
player\_names <- read.csv("playerNames.csv")  
  
result <- fetch\_bowlers\_average\_wickets(match\_results, "Parramatta", team\_members, player\_names)  
pander(result)

|  |  |
| --- | --- |
| playerNames | averageWickets |
| GUINEVERE Q. | 3 |
| INDIE N. | 3 |
| CHESTER G. | 3 |
| GRAYSON P. | 2 |
| ROBYN F. | 2 |
| ANTHONY-JAY O. | 2 |
| OLLIE S. | 1.75 |
| AUSTIN D. | 1.5 |
| OLAF J. | 1 |
| TOBIAS D. | 1 |
| EVELYNN S. | 1 |
| YUSAIRAH M. | 1 |
| ALFRED G. | 0.5 |
| MACKENZIE V. | 0 |
| JOBY C. | 0 |
| EMILIA P. | 0 |
| AREN O. | 0 |

### Code Testing:

Testing Code with the new IPL(Indian Premier League) matches.

#############################################################################  
############################# Code Testing! #################################  
  
ipl\_match\_results <- fromJSON(file = "IPLMatchResults.json")  
# loading team members details from teamMembers.json file  
ipl\_team\_members <- fromJSON(file = "IPLTeamMembers.json")  
# loading player names from playerNames.csv file  
ipl\_player\_names <- read.csv("IPLPlayerNames.csv")  
  
# fetching Punjab's bowlers average wicket taken per match  
testing\_results4 <- fetch\_bowlers\_average\_wickets(ipl\_match\_results, "Punjab",  
 ipl\_team\_members, ipl\_player\_names)  
pander(testing\_results4)

|  |  |
| --- | --- |
| playerNames | averageWickets |
| M. Shami | 3.5 |
| Arshdeep Singh | 3 |
| Ravi Bishnoi | 3 |
| KL Rahul | 2.5 |
| Mayank Aggarwal | 2 |
| Chris Gayle | 2 |