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))
}
```

```r
#################################################################
## 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:

```r
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
```

```r
  # 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
```

```r
#####################################################################
## 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)
}
```

```r
##############################################################################
## 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