Week 5 Project 1 - Data Analysis

Shri Tripathi

2024-10-04

Project 1

Overview

In this project, we are provided with a text file containing chess tournament results, where the data has some structure. The goal is to use R to extract relevant information from this file and generate a .CSV file with the following details for each player:

- Player's Name
- Player's State
- Total Number of Points
- Player's Pre-Rating
- Average Pre-Tournament Chess Rating of Opponents

Methodology and Approach

Importing, Reading, and Cleaning

The initial step of the project involved importing the text file into R and reading its content. I uploaded the text file to GitHub and accessed it via its URL. Using the `readLines()` function, I processed the file line by line. Once the data was loaded, I began cleaning it by first removing the header rows to focus on the relevant player information.

```
# First, import the text file from the provided URL
# Reading the file line by line

lines = readLines(url("https://raw.githubusercontent.com/sleepysloth12/data607_proj1/main/tournamentinf
```

```
## Warning in
## readLines(url("https://raw.githubusercontent.com/sleepysloth12/data607_proj1/main/tournamentinfo.txt
## incomplete final line found on
## 'https://raw.githubusercontent.com/sleepysloth12/data607_proj1/main/tournamentinfo.txt'
```

```
lines[1]
```

```
## [1] "-----------------------------------------------------------------------------------------"
```

```
# Removing the header rows to clean the data

lines = lines[-c(1,2,3,4)]
lines[1]
```

```
## [1] "    1 | GARY HUA                       |6.0  |W  39|W  21|W  18|W  14|W   7|D  12|D   4|"
```

Now that the header has been removed, the next step is to eliminate the dashed lines that separate each player's information. These dashed lines occur every third row. To remove them, I use an `if` statement that checks whether the row number is divisible by 3. I then generate a list of all multiples of 3 up to the length of the text file and remove those lines, effectively getting rid of the dashed lines. As shown in the output, the third line, which was previously a dashed line, is now the first line of the second player's information.

```
# Creating a conditional statement
# If the total number of lines is divisible by 3, identify every third line (which contains dashes) and

lines[3]
```

```
## [1] "-----------------------------------------------------------------------------------------"
```

```
if (length(lines)%%3 == 0){
  multiples_of_3 = seq(3, length(lines), by = 3)
  lines = lines[-c(multiples_of_3)]
}

lines[3]
```

```
## [1] "    2 | DAKSHESH DARURI                |6.0  |W  63|W  58|L   4|W  17|W  16|W  20|W   7|"
```

Now we have two lines of data for each player: the first line contains the player's name, points, etc., while the second line contains the player's state, pre-rating, and related details. I'll create two vectors to store each line separately. The first line for each player will go into one vector, and the second line into another. To achieve this, I use a `for` loop that iterates over the lines, checking if the index is divisible by 2, and places the line into the corresponding vector.

```
#Now we only have player info, with the same info in every other line
#the first line has the names and who is playing against/ win or lose
#the second line has the ranking/ state

#going to separate into two vectors separating two lines

line_one=c()
line_two=c()
id=1

for(line in lines){
  if(id%%2==0){
    line_two=c(line_two,line)
  }else {
    line_one=c(line_one,line)
  }
```

```
  id=id+1
}
line_one[1:5]
```

```
## [1] "    1 | GARY HUA                       |6.0  |W  39|W  21|W  18|W  14|W   7|D  12|D   4|"
## [2] "    2 | DAKSHESH DARURI                |6.0  |W  63|W  58|L   4|W  17|W  16|W  20|W   7|"
## [3] "    3 | ADITYA BAJAJ                   |6.0  |L   8|W  61|W  25|W  21|W  11|W  13|W  12|"
## [4] "    4 | PATRICK H SCHILLING            |5.5  |W  23|D  28|W   2|W  26|D   5|W  19|D   1|"
## [5] "    5 | HANSHI ZUO                     |5.5  |W  45|W  37|D  12|D  13|D   4|W  14|W  17|"
```

```
line_two[1:5]
```

```
## [1] "   ON | 15445895 / R: 1794   ->1817     |N:2  |W    |B    |W    |B    |W    |B    |W    |"
## [2] "   MI | 14598900 / R: 1553   ->1663     |N:2  |B    |W    |B    |W    |B    |W    |B    |"
## [3] "   MI | 14959604 / R: 1384   ->1640     |N:2  |W    |B    |W    |B    |W    |B    |W    |"
## [4] "   MI | 12616049 / R: 1716   ->1744     |N:2  |W    |B    |W    |B    |W    |B    |B    |"
## [5] "   MI | 14601533 / R: 1655   ->1690     |N:2  |B    |W    |B    |W    |B    |W    |B    |"
```

Now, both `line_one` and `line_two` contain the correct information, but it is still in raw text form. The next step is to convert this data into a structured format. To do that, I'll extract the individual sections from the text by splitting each line at the "|" character using `str_split()`. Once the data is split, I'll convert the results into a matrix and then transform it into a data frame, allowing us to perform further operations on it.

```
# Now both line_one and line_two contain the correct information.
# Converting each into a data frame by splitting the values separated by '|', where each part gets its

split_data_one = lapply(line_one, function(x) strsplit(x, "\\|"))
split_data_two = lapply(line_two, function(x) strsplit(x, "\\|"))

# Moving from list to matrix, and then converting to data frame

split_data_one_mat = do.call(rbind, lapply(split_data_one, function(x) unlist(x[[1]])))
df_one = as.data.frame(split_data_one_mat, stringsAsFactors = FALSE)

split_data_two_mat = do.call(rbind, lapply(split_data_two, function(x) unlist(x[[1]])))
df_two = as.data.frame(split_data_two_mat, stringsAsFactors = FALSE)

# Adding column names to each data frame
col_names_1 = c('player_id', 'name', 'total_pts', 'rnd_1_comb', 'rnd_2_comb', 'rnd_3_comb', 'rnd_4_comb
colnames(df_one) = col_names_1

col_names_2 = c('state', 'comb_rank', 'idk', 'col_rnd_1', 'col_rnd_2', 'col_rnd_3', 'col_rnd_4', 'col_r
colnames(df_two) = col_names_2

# Preview the first few rows of both data frames
head(df_one)
```

```
##   player_id                          name total_pts rnd_1_comb rnd_2_comb
## 1         1   GARY HUA                          6.0        W  39       W  21
## 2         2   DAKSHESH DARURI                   6.0        W  63       W  58
## 3         3   ADITYA BAJAJ                      6.0        L   8       W  61
```

```
## 4        4   PATRICK H SCHILLING                 5.5        W  23      D  28
## 5        5   HANSHI ZUO                          5.5        W  45      W  37
## 6        6   HANSEN SONG                         5.0        W  34      D  29
##   rnd_3_comb rnd_4_comb rnd_5_comb rnd_6_comb rnd_7_comb
## 1      W  18      W  14      W   7      D  12      D   4
## 2      L   4      W  17      W  16      W  20      W   7
## 3      W  25      W  21      W  11      W  13      W  12
## 4      W   2      W  26      D   5      W  19      D   1
## 5      D  12      D  13      D   4      W  14      W  17
## 6      L  11      W  35      D  10      W  27      W  21
```

```
head(df_two)
```

```
##    state                          comb_rank   idk col_rnd_1 col_rnd_2 col_rnd_3
## 1    ON   15445895 / R: 1794   ->1817      N:2       W         B         W
## 2    MI   14598900 / R: 1553   ->1663      N:2       B         W         B
## 3    MI   14959604 / R: 1384   ->1640      N:2       W         B         W
## 4    MI   12616049 / R: 1716   ->1744      N:2       W         B         W
## 5    MI   14601533 / R: 1655   ->1690      N:2       B         W         B
## 6    OH   15055204 / R: 1686   ->1687      N:3       W         B         W
##   col_rnd_4 col_rnd_5 col_rnd_6 col_rnd_7
## 1     B         W         B         W
## 2     W         B         W         B
## 3     B         W         B         W
## 4     B         W         B         B
## 5     W         B         W         B
## 6     B         B         W         B
```

Editing the Data Frame

Now that we have two data frames, we need to make some modifications. In the first data frame, the round columns include the outcome of the game (W/L/D) along with the opponent's player ID. Since we are only interested in the opponent's player ID for this assignment, I used regex to extract only the numeric values from these columns. Afterward, I renamed the columns to more appropriate names, as they now only contain the opponent's player IDs.

```
# Extracting the opponent player IDs from each combined round column
# and updating the existing data frame

df_one_to_split = c('rnd_1_comb', 'rnd_2_comb', 'rnd_3_comb', 'rnd_4_comb', 'rnd_5_comb', 'rnd_6_comb',

# Extracting only the numeric values from each column and converting to numeric type

for (col in df_one_to_split) {
  df_one[[col]] = as.numeric(gsub("[^0-9]", "", df_one[[col]]))
}

head(df_one)
```

```
##   player_id                              name total_pts rnd_1_comb rnd_2_comb
## 1         1   GARY HUA                           6.0          39         21
## 2         2   DAKSHESH DARURI                    6.0          63         58
```

```
## 3        3   ADITYA BAJAJ                        6.0           8         61
## 4        4   PATRICK H SCHILLING                 5.5          23         28
## 5        5   HANSHI ZUO                          5.5          45         37
## 6        6   HANSEN SONG                         5.0          34         29
##   rnd_3_comb rnd_4_comb rnd_5_comb rnd_6_comb rnd_7_comb
## 1         18         14          7         12          4
## 2          4         17         16         20          7
## 3         25         21         11         13         12
## 4          2         26          5         19          1
## 5         12         13          4         14         17
## 6         11         35         10         27         21
```

```
# Renaming columns to accurately reflect opponent player IDs
col_names_1 = c('player_id', 'name', 'total_pts', 'rnd_1_op', 'rnd_2_op', 'rnd_3_op', 'rnd_4_op', 'rnd_5
colnames(df_one) = col_names_1
```

Similarly, in the second data set, I needed to extract the Pre-Rank score for each player. The ranking was positioned between 'R:' and '->'. I used regex to extract this numeric value. After extracting the pre-rank, I updated the column names to reflect the cleaned data.

```
library(stringr)
df_two$comb_rank=str_extract(df_two$comb_rank, "(?<=R: )\\d+")
df_two$comb_rank=as.numeric(df_two$comb_rank)
col_names_2=c('state','pre_rank','idk','col_rnd_1','col_rnd_2','col_rnd_3','col_rnd_4','col_rnd_5','col
colnames(df_two)=col_names_2
head(df_two)
```

```
##    state pre_rank   idk col_rnd_1 col_rnd_2 col_rnd_3 col_rnd_4 col_rnd_5
## 1    ON      1794 N:2       W         B         W         B         W
## 2    MI      1553 N:2       B         W         B         W         B
## 3    MI      1384 N:2       W         B         W         B         W
## 4    MI      1716 N:2       W         B         W         B         W
## 5    MI      1655 N:2       B         W         B         W         B
## 6    OH      1686 N:3       W         B         W         B         B
##   col_rnd_6 col_rnd_7
## 1     B         W
## 2     W         B
## 3     B         W
## 4     B         B
## 5     W         B
## 6     W         B
```

Combining and Calculating

I wanted to combine both data frames into a single one. First, I removed the columns containing information that is not relevant to our analysis. After cleaning up the data, I merged the two data frames into one.

```
# Now let's combine both data frames
# First, we'll remove the columns that are not relevant to our analysis

df_two$col_rnd_1 = NULL
df_two$col_rnd_2 = NULL
```

```
df_two$col_rnd_3 = NULL
df_two$col_rnd_4 = NULL
df_two$col_rnd_5 = NULL
df_two$col_rnd_6 = NULL
df_two$col_rnd_7 = NULL
df_two$idk = NULL

# Combine the two data frames into one

chess_df = cbind(df_one, df_two)
chess_df$player_id = as.integer(chess_df$player_id)
head(chess_df)
```

```
##   player_id                              name total_pts rnd_1_op rnd_2_op
## 1         1  GARY HUA                              6.0         39       21
## 2         2  DAKSHESH DARURI                       6.0         63       58
## 3         3  ADITYA BAJAJ                          6.0          8       61
## 4         4  PATRICK H SCHILLING                   5.5         23       28
## 5         5  HANSHI ZUO                            5.5         45       37
## 6         6  HANSEN SONG                           5.0         34       29
##   rnd_3_op rnd_4_op rnd_5_op rnd_6_op rnd_7_op  state pre_rank
## 1       18       14        7       12        4    ON      1794
## 2        4       17       16       20        7    MI      1553
## 3       25       21       11       13       12    MI      1384
## 4        2       26        5       19        1    MI      1716
## 5       12       13        4       14       17    MI      1655
## 6       11       35       10       27       21    OH      1686
```

Next, I created a new column to store the average pre-rank score of each player's opponents. To calculate this, I used a `for` loop to process each row of the data frame. For each player, the loop retrieves the opponent IDs using regex and converts them to integers. I removed any `NA` values since some players did not have opponents for all rounds. Then, for each valid opponent, I extracted their pre-rank score and calculated the mean, storing it in the newly created column.

```
# Creating a new column to store the average opponent pre-rank score
# Looping through each row to calculate the average opponent pre-rank

chess_df$avg_op_pre_rank = NA

# Loop through each row in chess_df
for (i in 1:nrow(chess_df)) {

  # Extract opponent IDs for each round
  op_ids = unlist(chess_df[i, grep("rnd_\\d+_op", names(chess_df))])

  # Convert opponent IDs to integers
  op_ids = as.integer(op_ids)

  # Remove NA values
  op_ids = na.omit(op_ids)

  # Get the pre-rank of each opponent
  op_ranks = chess_df[chess_df$player_id %in% op_ids, "pre_rank"]
```

```
  # Calculate the mean pre-rank of opponents
  chess_df$avg_op_pre_rank[i] = mean(op_ranks, na.rm = TRUE)
}

# Preview the updated data frame
head(chess_df)
```

```
##   player_id                          name total_pts rnd_1_op rnd_2_op
## 1         1  GARY HUA                         6.0         39       21
## 2         2  DAKSHESH DARURI                  6.0         63       58
## 3         3  ADITYA BAJAJ                     6.0          8       61
## 4         4  PATRICK H SCHILLING              5.5         23       28
## 5         5  HANSHI ZUO                       5.5         45       37
## 6         6  HANSEN SONG                      5.0         34       29
##   rnd_3_op rnd_4_op rnd_5_op rnd_6_op rnd_7_op  state pre_rank avg_op_pre_rank
## 1       18       14        7       12        4    ON      1794        1605.286
## 2        4       17       16       20        7    MI      1553        1561.333
## 3       25       21       11       13       12    MI      1384        1665.000
## 4        2       26        5       19        1    MI      1716        1573.571
## 5       12       13        4       14       17    MI      1655        1587.667
## 6       11       35       10       27       21    OH      1686        1518.714
```

Exporting Data

After completing all the necessary steps for the assignment, I organized the required information into a new data frame that contained everything needed for the CSV file. Then, I saved this data frame as "extracted_chess_info.csv" and placed it in the current working directory.

```
# Creating a new data frame with the required format as specified in the assignment

to_be_exported_list = list(Player_Name = chess_df$name,
                           Player_State = chess_df$state,
                           Total_Points = chess_df$total_pts,
                           Player_Pre_Rating = chess_df$pre_rank,
                           Average_Opponent_Pre_Rating = chess_df$avg_op_pre_rank)
to_be_exported_df = as.data.frame(to_be_exported_list)

# Preview the new data frame
head(to_be_exported_df)
```

```
##                         Player_Name Player_State Total_Points Player_Pre_Rating
## 1  GARY HUA                                  ON         6.0                1794
## 2  DAKSHESH DARURI                           MI         6.0                1553
## 3  ADITYA BAJAJ                              MI         6.0                1384
## 4  PATRICK H SCHILLING                       MI         5.5                1716
## 5  HANSHI ZUO                                MI         5.5                1655
## 6  HANSEN SONG                               OH         5.0                1686
##   Average_Opponent_Pre_Rating
## 1                    1605.286
## 2                    1561.333
## 3                    1665.000
```

```
## 4                    1573.571
## 5                    1587.667
## 6                    1518.714
```

```
# Write and export the data frame as a CSV file
write.csv(to_be_exported_df, "extracted_chess_info.csv", row.names = FALSE)
```

Creating a Function

Although it wasn't required, I decided to package all of the code into a function. Doing this allows you to easily reuse the code whenever needed without having to rewrite it. This is especially helpful for automation. The function below is the same as the code from above but with input variables changed to make it more flexible. It returns a CSV file. I also ran an example, and you can modify the file path to your own and run it as well.

```
chess_to_csv=function(txt_url, export_path_name){
  # chess_to_csv

  # Function to convert a chess tournament .txt file into a .csv file as specified in the assignment

  # INPUTS
  # txt_url: URL of the chess notation text file
  # export_path_name: Path and name of the CSV file to be exported (e.g., "/home/Data607/proj1/chess_te

  # OUTPUT
  # A CSV file containing player name, player state, total points, pre-rating, and average opponent pre


  # First, import the text file and read it line by line

  lines= readLines(txt_url)
  lines

  # Remove the header rows

  lines=lines[-c(1,2,3,4)]


  # Check if the total number of lines is divisible by 3
  # Remove every third line (these are the dashed lines)

  if (length(lines)%%3 == 0){
    multiples_of_3=seq(3, length(lines),by=3)
    lines=lines[-c(multiples_of_3)]
  }

  lines

  # Now, we only have player information in alternating lines
  # The first line contains names and match results
  # The second line contains ranking and state info

  # Separate the data into two vectors: one for the first line and one for the second line
```

```
line_one=c()
line_two=c()
id=1

for(line in lines){
  if(id%%2==0){
    line_two=c(line_two,line)
  }else {
    line_one=c(line_one,line)
  }
  id=id+1
}
line_one
line_two

# Now each vector has the relevant information for the players
# Split the data in both vectors based on the '|' separator and convert them into data frames

split_data_one=lapply(line_one, function(x) strsplit(x, "\\|"))
split_data_two=lapply(line_two, function(x) strsplit(x, "\\|"))

# Convert the lists to matrices and then to data frames

split_data_one_mat=do.call(rbind, lapply(split_data_one, function(x) unlist(x[[1]])))
df_one=as.data.frame(split_data_one_mat, stringsAsFactors = FALSE)
split_data_two_mat=do.call(rbind, lapply(split_data_two, function(x) unlist(x[[1]])))
df_two=as.data.frame(split_data_two_mat, stringsAsFactors = FALSE)

# Add appropriate column names for both data frames
col_names_1=c('player_id','name','total_pts','rnd_1_comb','rnd_2_comb','rnd_3_comb','rnd_4_comb','rnd
colnames(df_one)=col_names_1
col_names_2=c('state','comb_rank','idk','col_rnd_1','col_rnd_2','col_rnd_3','col_rnd_4','col_rnd_5','
colnames(df_two)=col_names_2

head(df_one)
head(df_two)

# Extract the opponent player IDs from the combined round columns

df_one_to_split=c('rnd_1_comb','rnd_2_comb','rnd_3_comb','rnd_4_comb','rnd_5_comb','rnd_6_comb','rnd_

# Extract only the numeric characters from these columns and convert them to numbers

for (col in df_one_to_split){
  df_one[[col]]=as.numeric(gsub("[^0-9]","",df_one[[col]]))
}

head(df_one)

# Update column names to reflect opponent IDs
col_names_1=c('player_id','name','total_pts','rnd_1_op','rnd_2_op','rnd_3_op','rnd_4_op','rnd_5_op','
colnames(df_one)=col_names_1
```

```r
# Extract the pre-rating for each player from the 'comb_rank' column

library(stringr)

df_two$comb_rank=str_extract(df_two$comb_rank, "(?<=R: )\\d+")
df_two$comb_rank=as.numeric(df_two$comb_rank)
col_names_2=c('state','pre_rank','idk','col_rnd_1','col_rnd_2','col_rnd_3','col_rnd_4','col_rnd_5','c
colnames(df_two)=col_names_2
head(df_two)


# Remove unnecessary columns from df_two

df_two$col_rnd_1=NULL
df_two$col_rnd_2=NULL
df_two$col_rnd_3=NULL
df_two$col_rnd_4=NULL
df_two$col_rnd_5=NULL
df_two$col_rnd_6=NULL
df_two$col_rnd_7=NULL
df_two$idk=NULL
head(df_two)

# Combine the two data frames into one

chess_df=cbind(df_one,df_two)
chess_df$player_id=as.integer(chess_df$player_id)
head(chess_df)

# Create a new column for the average opponent pre-rank

chess_df$avg_op_pre_rank = NA

# Loop through each row and calculate the average pre-rank of the opponents

for (i in 1:nrow(chess_df)) {

  # Extract opponent IDs for each round

  op_ids = unlist(chess_df[i, grep("rnd_\\d+_op", names(chess_df))])

  op_ids=as.integer(op_ids)

  # Remove NA values (in case some players have no opponents in certain rounds)

  op_ids=na.omit(op_ids)

  # Get the pre-rank of each opponent

  op_ranks = chess_df[chess_df$player_id %in% op_ids, "pre_rank"]

  # Calculate and store the mean pre-rank of the opponents
```

```
    chess_df$avg_op_pre_rank[i] = mean(op_ranks, na.rm = TRUE)
  }

  head(chess_df)

  # Create a new data frame with the required format as per the assignment

  to_be_exported_list=list(Player_Name=chess_df$name,
                           Player_State=chess_df$state,
                           Total_Points=chess_df$total_pts,
                           Player_Pre_Rating=chess_df$pre_rank,
                           Average_Opponent_Pre_Rating=chess_df$avg_op_pre_rank)
  to_be_exported_df=as.data.frame(to_be_exported_list)

  head(to_be_exported_df)

  # Write the final data frame to a CSV file at the specified export path

  write.csv(to_be_exported_df,export_path_name, row.names=FALSE)

}
```

```
chess_to_csv("https://raw.githubusercontent.com/sleepysloth12/data607_proj1/main/tournamentinfo.txt",
             "C:/Users/16462/Desktop/data607/Project1/function_test.csv")
```

```
## Warning in readLines(txt_url): incomplete final line found on
## 'https://raw.githubusercontent.com/sleepysloth12/data607_proj1/main/tournamentinfo.txt'
```

Conclusion

In this project, I processed the chess tournament text file and generated a CSV file based on the given specifications. I encapsulated the entire process into a reusable function. However, the function is not perfect—it won't handle text files with a slightly different format. In the future, I plan to improve the function to handle such variations. This assignment also reminded me of object-oriented programming in Python. One alternative approach I considered but didn't explore was creating a `Player` class, where each column would be an attribute, and using a method to extract and assign the relevant data from the file.