Data 607 Project 1

Jean Jimenez

2023-09-20

Project 1

Overview

Project 1 assignment was to take a text file and to process it and return a csv.

The text file is of chess tournament scores and it is in the format of Chess Ranking System.

In this assignment, I will show how I processed the given data set and exported as csv.

Work and Thought Process

Importing, Reading, and Cleaning

The first step of the project was to import the text file into R and read it. I imported the text file using the url (I uploaded it to github). Read lines go line by line. Afterwards, I start cleaning the text. I first eliminate the header rows.

```
#First import text file
#will read line by line

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

```
## 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] "-----------------------------------------------------------------------------------------"
```

```
#filtering out header

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 is removed, I want to remove the dashed line that divides every player from each other in the text file. The dash line occurs every third row. To remove them, I run it through this if statement. The statement checks to see if the number of rows is divisible by 3. Afterwards, I generate a list of all multiples of 3 until the length of the text files, and I remove those lines (which remove the dashed lines). As you see in the output, Line 3 before the statement is the dashed line, now it is first line of the second player.

```
#making a conditional statement

#If the length of the # of lines is divisible by 3,
#take every third line (dashed) and remove it from our text
#aka generate a all multiples of 3 till len lines

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 essentially have two lines for each player. The first line having the player name, points, etc. The second line having State, Pre-rank, etc. I will make two vectors to put each line in them. In other words, the first line of each player will go into one, while the second line will to to two. To do this, I run a for loop that takes every other line (by checking if index is divisible by 2) and puts it in the appropriate 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, everything in line_one and everything in line_two has all the same info. The problem now is that it is still in text form. I want it to be in a data frame so that we can do stuff to it. I will now parse out each of the individual sections to get the text between the "|" character. I do this by using str_split. Then, I convert what I have for each into a matrix, and subsequently a data frame.

```
#Now line one and two are each showing the same info.
#will make each a data frame
# every value separated by  '|'  will get its own column

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 [temp] to dataframe

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
col_names_1=c('player_id','name','total_pts','rnd_1_comb','rnd_2_comb','rnd_3_comb','rnd_4_comb','rnd_5_
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','col
colnames(df_two)=col_names_2

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, we have two data frames. We have to modify them a bit. In the first data frame, the round columns show W/L/D as well as the opponent's player ID. For the assignment, we only care about the opponents player id. I used regex to only get the numbers from those columns. Then, I renamed the columns into more appropriate column names since the round columns are no longer combined with other text.

```
#Splitting each combined round x colomn n to get the opponent number
#put in new dataframe

df_one_to_split=c('rnd_1_comb','rnd_2_comb','rnd_3_comb','rnd_4_comb','rnd_5_comb','rnd_6_comb','rnd_7_

#getting only numbeer characters from each column and converting to number

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
```

```
#changing column names to be more accurate
col_names_1=c('player_id','name','total_pts','rnd_1_op','rnd_2_op','rnd_3_op','rnd_4_op','rnd_5_op','rn
colnames(df_one)=col_names_1
```

Similarly in the other data set, I had to extract the Pre-rank score of each player. The ranking was located between the 'R:' and the '->'. I again used regex to extract this number. Afterwards, I changed column names.

```
#now splitting each combined_rank columnn to get pre rating

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 want to combine both data frames into one. First, I removed columns with information that doesn't matter to us. Then I merge both data frames.

```
#Now lets combine both data frames
#first will remove columns we dont care about
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
```

```r
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
```

Afterwards, I made a new column to place the average opponent pre rank score. Then I made a for loop to retrieve and calculate this. For each row of the data frame, the loop obtains all of the opponent ids. I did this using regex. Then, I converted all those opponent ids to integers. Afterwards, I took out NA values since some players didn't have opponents for some rounds. Then, I got each of those opponents rank and took the mean. I placed that mean in the new column created.

```r
#creating new column named avg op rank
#making a loop that will get all the oposition ranks


chess_df$avg_op_pre_rank = NA

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

  # extract op ID for each round

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

  op_ids=as.integer(op_ids)

  #take out NA

  op_ids=na.omit(op_ids)

  # get prerank for each op

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

  #mean op pre rank

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

```r
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 I had everything the assignment was asking for, I organized a new data frame with everything that was wanted on the csv file. Then, I wrote the data frame into a new csv file "extracted_chess_info.csv". This new csv file will be placed in the current working directory.

```r
#making new dataframe with required format like the assignment states

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)
```

```
##                         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
```

```r
#write and export as csv
write.csv(to_be_exported_df,"extracted_chess_info.csv", row.names=FALSE)
```

Making a Function

Although not required, I felt the need to package all of this into a function. When you do this, you can call it and use it whenever you want without coding again. This is very useful especially when it comes to automation. The following is my function. It is just the same code as above but with the input vars changed. It will return the csv file. I also run an example. You can change the path to your own and run it.

```
chess_to_csv=function(txt_url, export_path_name){
  # chess_to_csv
  #by Jean Jimenez
  #09/20/2023 for Data 607 Proj 1

  #Function that takes .txt file of chess tournament notation to .csv to the specifications of the assi

  #INPUTS

  #txt_url: url of txt file in chess notation

  #export_path_name:  Path and name of csv file to be exported i.e.) "/home/Data607/proj1/chess_test.cs


  #OUTPUT
  #CSV file showing player name, player state, total points, pre rating, avg opponent pre rating


  #First import text file
  #will read line by line

  lines= readLines(txt_url)
  lines

  #filtering out header

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


  #making a conditional statement

  #If the length of the # of lines is divisible by 3,
  #take every third line (dashed) and remove it from our text
  #aka generate a all multiples of 3 till len 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 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
```

```r
#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
line_two

#Now line one and two are each showing the same info.
#will make each a data frame
# every value separated by  '|'  will get its own column

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 [temp] to dataframe

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
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)

#Splitting each combined round x colomn n to get the opponent number
#put in new dataframe

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

#getting only numbeer characters from each column and converting to number

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

head(df_one)
```

```r
#changing column names to be more accurate
col_names_1=c('player_id','name','total_pts','rnd_1_op','rnd_2_op','rnd_3_op','rnd_4_op','rnd_5_op','r
colnames(df_one)=col_names_1

#now splitting each combined_rank columnn to get pre rating

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)



#Now lets combine both data frames
#first will remove columns we dont care about
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

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

#creating new column named avg op rank
#making a loop that will get all the oposition ranks


chess_df$avg_op_pre_rank = NA

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

  # extract op ID for each round

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

  op_ids=as.integer(op_ids)

  #take out NA

  op_ids=na.omit(op_ids)
```

```
    # get prerank for each op

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

    #mean op pre rank

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

  head(chess_df)

  #making new dataframe with required format like the assignment states

  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 and export as csv
  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", "
```

```
## 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 took the text file, processed it, and returned a csv file for the specifications asked for. I packaged all of that into a function that can be replicated. That being said, the function is not perfect. It will not work if a text file is inserted if it has a slightly different format. In the future, I can fix my function to account for this. Also, this assignment reminded me of object-oriented programming in python. One idea I had but didn't explore/ another way to do this project is by making a "Player" class, with each of the columns being an attribute; and creating a function that will extract the text from the file and assign them.