DATA 607 Project 1

Adam Gersowitz

2/22/2020

Overview

Week 4 Project 1 will be focused on analyzing and manipulating a sctructured .txt file of chess tournament results.The ideal end results would be a .csv that would have the Player's Name, State, Total Number of Points, Plyaers Pre-rating and the Avergare Pre Chess Rating of Opponents.

Getting and Cleaning the Data

The first step of this project is getting the original tournamentinfo.txt file into R Studio (I've saved this file in a github repository). After getting the file I used a vector to help select the correct lines within the file for analysis. I want to skip the lines that just have dashes and I want to seperate the top and bottom lines so that I can join them back together later.

```
library (readr)
library(RCurl)


chess_raw <- read.delim("https://raw.githubusercontent.com/agersowitz/Data-607-Datasets/master/tourname

chess_line1 <- read.delim("https://raw.githubusercontent.com/agersowitz/Data-607-Datasets/master/tournam

chess_line2 <- read.delim("https://raw.githubusercontent.com/agersowitz/Data-607-Datasets/master/tournam


chess_raw <- data.frame(chess_raw)
chess_line1 <- data.frame(chess_line1)
chess_line2 <- data.frame(chess_line2)
```

Manipulating the data so that it can be joined

My next steps involve creating a field that is just the row number. Due to how this file is laid out and how I broke out the data, the row numbers in CL1 and CL2 should be associated with one chess player and therefore can be used as a key to join the data on.

```
CR<- tibble::rowid_to_column(chess_raw, "TOURNAMENT_ID")
CL1<- tibble::rowid_to_column(chess_line1, "TOURNAMENT_ID")
CL2<- tibble::rowid_to_column(chess_line2, "TOURNAMENT_ID")
```

Breaking out the Data Frame into columns for analysis

At this point there are only 2 columns in each of the data frames (TOURNAMENT_ID and chess_line1/2). I use the sparate function to split these columns out by a delimiter. "|" is the primary deliminator in this data set but other characters such as "/" and "->" are used to split the USCFID, pre and post rating fields. I then use gsub and regular expressions to clean up the data so that characters like R: and P are removed form the rating fields. Additionally I removed the first element of each data frame because the top row is just solumn headers and I have already altered them using the separate function.

```
#CL = CL1.applymap(str)
library(dplyr)
library(tidyr)

CL1<-CL1 %>% separate(chess_line1, c("Player_Num", "Player_Name", "Total_Points","Round1","Round2","Rou

CL1<-CL1 %>% separate(Round1, c("Round1_Result", "Round1_Opponent_ID"), " ", extra = "merge")
CL1<-CL1 %>% separate(Round2, c("Round2_Result", "Round2_Opponent_ID"), " ", extra = "merge")
CL1<-CL1 %>% separate(Round3, c("Round3_Result", "Round3_Opponent_ID"), " ", extra = "merge")
CL1<-CL1 %>% separate(Round4, c("Round4_Result", "Round4_Opponent_ID"), " ", extra = "merge")
CL1<-CL1 %>% separate(Round5, c("Round5_Result", "Round5_Opponent_ID"), " ", extra = "merge")
CL1<-CL1 %>% separate(Round6, c("Round6_Result", "Round6_Opponent_ID"), " ", extra = "merge")
CL1<-CL1 %>% separate(Round7, c("Round7_Result", "Round7_Opponent_ID"), " ", extra = "merge")

CL2<-CL2 %>% separate(chess_line2, c("State", "USCFIDRtgPrePost", "N","Bottom"), "\\|", extra = "merge"

CL2<-CL2 %>% separate(USCFIDRtgPrePost, c("USCFID", "RtgPrePost"), "\\/", extra = "merge")
CL2<-CL2 %>% separate(RtgPrePost, c("Rtg_Pre", "Rtg_Post"), "\\->", extra = "merge")

CL2$Rtg_Pre <- gsub("R: ","",CL2$Rtg_Pre)
CL2$Rtg_Pre <- gsub("P.*$","",CL2$Rtg_Pre)
CL2$Rtg_Post <- gsub("P.*$","",CL2$Rtg_Post)
CL2$N <- gsub("N:","",CL2$N)
CL1$Round7_Opponent_ID <- gsub("\\|$","",CL1$Round7_Opponent_ID)

CL1<-CL1[-1,]
CL2<-CL2[-1,]
```

Merging the data and creating player/results tables

Now I will merge the two data frames based on the tournament_id(row number) field I created earlier. Once these are joined properly I will select the necessary fields from the merged data set to create a players and reuslts table. This will make calculating the opponenets average pre rating much easier later on.

```
cr <- merge(CL1,CL2,by="TOURNAMENT_ID")

Players <- select(cr, Player_Num, Player_Name, Total_Points, State, USCFID, Rtg_Pre, Rtg_Post, N)
Results <- select(cr, Player_Num, Round1_Result, Round1_Opponent_ID
                                , Round2_Result, Round2_Opponent_ID
                                , Round3_Result, Round3_Opponent_ID
                                , Round4_Result, Round4_Opponent_ID
                                , Round5_Result, Round5_Opponent_ID
                                , Round6_Result, Round6_Opponent_ID
                                , Round7_Result, Round7_Opponent_ID)
```

Reshaping and Manipulating the results table so that it can be aggregated later

At this point the Results table has the results and opponents for each round as their own column in the dataframe (i.e. round1__oppoenent__id, round2__results etc.). I'll use the melt function in the reshape package to transpose this data so that the round can be a column and an opponent id will be a clumn which reduces this data frame to 4 columns. By doing this I've caused a multiplication of the amount of rows by the number of rounds so I need to break out the results into a dataframe for each round that only contains the data for that round. I then rename the variable and value fields produced by the melt function to the Round and Result columns that make sense for this dataset.I then union all 7 of the results data frames back together so that they can be in one Results data frame for analysis.

NOTE: If I knew I was going to return to this code with additional rounds or tournaments I would have added a tournament number field or a date field so that I could add on to this code. Additionally, I would have created a function that iterates through the rounds to preform the transformation below rather than writing it out for each round individually.

```r
#install.packages('reshape')
library(reshape)
Results1 <- melt(Results, id=(c("Player_Num", "Round1_Opponent_ID")))
Results2 <- melt(Results, id=(c("Player_Num", "Round2_Opponent_ID")))
Results3 <- melt(Results, id=(c("Player_Num", "Round3_Opponent_ID")))
Results4 <- melt(Results, id=(c("Player_Num", "Round4_Opponent_ID")))
Results5 <- melt(Results, id=(c("Player_Num", "Round5_Opponent_ID")))
Results6 <- melt(Results, id=(c("Player_Num", "Round6_Opponent_ID")))
Results7 <- melt(Results, id=(c("Player_Num", "Round7_Opponent_ID")))


Results1 <- Results1[ which(Results1$variable=='Round1_Result'), ]
Results2 <- Results2[ which(Results2$variable=='Round2_Result'), ]
Results3 <- Results3[ which(Results3$variable=='Round3_Result'), ]
Results4 <- Results4[ which(Results4$variable=='Round4_Result'), ]
Results5 <- Results5[ which(Results5$variable=='Round5_Result'), ]
Results6 <- Results6[ which(Results6$variable=='Round6_Result'), ]
Results7 <- Results7[ which(Results7$variable=='Round7_Result'), ]


Results1$variable <- gsub("Round1_Result","1",Results1$variable)
Results2$variable <- gsub("Round2_Result","2",Results2$variable)
Results3$variable <- gsub("Round3_Result","3",Results3$variable)
Results4$variable <- gsub("Round4_Result","4",Results4$variable)
Results5$variable <- gsub("Round5_Result","5",Results5$variable)
Results6$variable <- gsub("Round6_Result","6",Results6$variable)
Results7$variable <- gsub("Round7_Result","7",Results7$variable)

library(plyr)
Results1<-rename(Results1, c("Round1_Opponent_ID"="Opponent_Num", "variable"="Round", "value"="Result")
Results2<-rename(Results2, c("Round2_Opponent_ID"="Opponent_Num", "variable"="Round", "value"="Result")
Results3<-rename(Results3, c("Round3_Opponent_ID"="Opponent_Num", "variable"="Round", "value"="Result")
Results4<-rename(Results4, c("Round4_Opponent_ID"="Opponent_Num", "variable"="Round", "value"="Result")
Results5<-rename(Results5, c("Round5_Opponent_ID"="Opponent_Num", "variable"="Round", "value"="Result")
Results6<-rename(Results6, c("Round6_Opponent_ID"="Opponent_Num", "variable"="Round", "value"="Result")
Results7<-rename(Results7, c("Round7_Opponent_ID"="Opponent_Num", "variable"="Round", "value"="Result")

Results<-rbind(Results1,Results2,Results3,Results4,Results5,Results6,Results7)
```

Cleaning the data and performing the average analysis

First I trim and convert the appropriate fields to numeric fields so that they can be joined in my sql statement as well as produce a result that will round to a decimal place if I choose. I then write a SQL statement that selects the appropriate fields as well as preforms the analysis to get the average opponent pre rating. I then write the results of this query to a .csv.

```
Results$Opponent_Num<-trimws(Results$Opponent_Num)
Results$Player_Num<-trimws(Results$Player_Num)
Players$Player_Num<-trimws(Players$Player_Num)

Results$Opponent_Num<-as.numeric(Results$Opponent_Num)
Results$Player_Num<-as.numeric(Results$Player_Num)
Players$Player_Num<-as.numeric(Players$Player_Num)
Players$Rtg_Pre<-as.numeric(Players$Rtg_Pre)

#install.packages('sqldf')
library(sqldf)
final_csv <-sqldf(
'SELECT distinct c.player_name,c.state, c.total_points, c.Rtg_Pre,
round((sum(b.rtg_pre) over(partition by a.player_num)/count(opponent_num) over (partition by a.player_nu
                  FROM Results a join Players b on a.Opponent_Num = b.Player_num
                  join players c on c.player_num = a.player_num
                  order by a.player_num asc')
write.csv(final_csv,"C:/Users/gerso/Documents/DATA-607/Project 1 final dataset.csv", row.names = FALSE)
```

Conclusion

After converting the data set into two tables that can be joined on player number it is a lot easier to analyze this data set. Now we can move beyond informational data gathering and determine more complex things such as the average rating of the opponents.