DATA 607 Project 1

Gregg Maloy

Introduction

The purpose of this project is to generate a CSV file from a partially structured text file. The end CSV file should be suitable for ingestion into a SQL database, ie MS SQL server.

Part 1: Load File and Inspection

The file was loaded into R Studio from Githuband consists of the results of a chess tournament.
There appears to be some structure to the file as:
1. Every third row appears to be repeated.
2. The file/columns appears to be delimited by the pipe ('|') character.

```
#Loading text file
tournament<- read_csv("https://raw.githubusercontent.com/goygoyummm/Data607_R/main/tournamentinfo.txt",

head(tournament[,1], 6)
```

```
## # A tibble: 6 x 1
##   ---------------------------------------------------------------------------~1
##   <chr>
## 1 Pair | Player Name                     |Total|Round|Round|Round|Round|Round|R~
## 2 Num  | USCF ID / Rtg (Pre->Post)       | Pts |  1  |  2  |  3  |  4  |  5  | ~
## 3 ---------------------------------------------------------------------------~
## 4 1 | GARY HUA                        |6.0  |W  39|W  21|W  18|W  14|W   7|D  1~
## 5 ON | 15445895 / R: 1794   ->1817     |N:2  |W    |B    |W    |B    |W    |B  ~
## 6 ---------------------------------------------------------------------------~
## # ... with abbreviated variable name
## #   1: '-------------------------------------------------------------------------------------------'
```

Part 2: Data Cleaning and Manipulation

The data was the cleaned and manipulated as described below:
1. 'Distinct()' was used to delete all of the rows which contained dashes (—-) except one.
2. The last row of dashes was then deleted, in this case the third row.
3. The column was then renamed 'string'.
4. A new variable 'ID' was then created which will serve as a unique identifier for each row.

```
#using distinct () to delete every third row
df<- tournament  %>% distinct()
df<- df  %>% filter(!row_number() %in% c(3))
colnames(df)[1] ="string"
#creation of new variable 'ID' to serve as unique identifier
df<-dplyr::mutate(df, ID = row_number())
```

Part 2: Data Cleaning and Manipulation *continued*

5. The dataframe was then left joined to itself using SQL via the 'sqldf' package.
   The variable 'ID' was used to join the table to itself, minus one row (join 'a.ID=b.ID-1').
   The result is a dataframe where every other row (rows 1,3,5,7, etc) represents either the header/column names or all of the players data respectively in a single row per player.

6. Every other second row(rows 2,4,6,8, etc) was then deleted, as these rows represent inaccurate information due to the manner the dataframe was joined to itself.

7. The columns that contain the strings were then delimited based on the pipe ('|') character and the resulting columns renamed.

```
#self joining the dataframe
df2 <- sqldf("SELECT a.string, b.string as'string_2'
              FROM df a
              left JOIN df b on a.ID=b.ID-1")

#delete every other row
df2<- df2  %>% filter(row_number() %% 2 != 0)

#delimit on the | character
df2<-suppressWarnings(cSplit(df2, "string","|"))
df2<-suppressWarnings(cSplit(df2, "string_2","|"))

colnames(df2)[1] ="Pair"
colnames(df2)[2] ="Player_Name"
colnames(df2)[3] ="Total_Pts"
colnames(df2)[4] ="Round_1"
colnames(df2)[5] ="Round_2"
colnames(df2)[6] ="Round_3"
colnames(df2)[7] ="Round_4"
colnames(df2)[8] ="Round_5"
colnames(df2)[9] ="Round_6"
colnames(df2)[10] ="Round_7"
colnames(df2)[11] ="State"
colnames(df2)[12] ="Pre"
```

Part 2: Data Cleaning and Manipulation *continued*

8. 'Sqldf' was then utilized to parse the column which did not include pipes and was thus not delimited. The resulting variable 'Pre_Ranking' represents each players ranking prior to the tournament.

9. The first row of the dataframe was deleted.

10. All of the 'Round' columns were separated into two variables, as the data within the 'Round' column represented two different metrics: 1. Result of the match (win, loss, draw, etc) and 2. Opponent. The resulting variables were named 'Round_1_Result_Opponent' and 'Round_1_Result'. In all 14 new variables were created.

11. A new dataframe was constructed which included the columns necessary for the CSV file.

```
#creation of the Pre_Ranking variable
df3 <- sqldf("SELECT Trim(Substring(Pre,14,5)) as Pre_Ranking,*
             FROM df2")

df3<- df3  %>% filter(!row_number() %in% c(1))

#creation of the 'Round_1_Result_Opponent', 'Round_1_Result', 'Round_2_Result_Opponent'
#, 'Round_2_Result',etc
df3 <- suppressWarnings(df3 %>% separate(Round_1, c('Round_1_Result', 'Round_1_Result_Opponent')))
df3 <- suppressWarnings(df3 %>% separate(Round_2, c('Round_2_Result', 'Round_2_Result_Opponent')))
df3 <- suppressWarnings(df3 %>% separate(Round_3, c('Round_3_Result', 'Round_3_Result_Opponent')))
df3 <- suppressWarnings(df3 %>% separate(Round_4, c('Round_4_Result', 'Round_4_Result_Opponent')))
df3 <- suppressWarnings(df3 %>% separate(Round_5, c('Round_5_Result', 'Round_5_Result_Opponent')))
df3 <- suppressWarnings(df3 %>% separate(Round_6, c('Round_6_Result', 'Round_6_Result_Opponent')))
df3 <- suppressWarnings(df3 %>% separate(Round_7, c('Round_7_Result', 'Round_7_Result_Opponent')))

#creating a new dataframe which includes most of variables needed for the CSV file
#APOLOGIES THE BELOW CREATES A WARNING WHICH I COULD NOT SUPPRESS
df3 <- df3 %>% group_by( Round_1_Result_Opponent, Round_2_Result_Opponent,Round_3_Result_Opponent
                         ,Round_4_Result_Opponent, Round_5_Result_Opponent,Round_6_Result_Opponent
                         ,Round_7_Result_Opponent) %>%
  summarise(Player_Name=Player_Name,State=State,Pair=Pair,Pre_Ranking=Pre_Ranking
            ,Total_Pts=Total_Pts, Pre_Ranking_Opp=Pre_Ranking)
```

```
## `summarise()` has grouped output by 'Round_1_Result_Opponent',
## 'Round_2_Result_Opponent', 'Round_3_Result_Opponent',
## 'Round_4_Result_Opponent', 'Round_5_Result_Opponent',
## 'Round_6_Result_Opponent'. You can override using the `.groups` argument.
```

Part 2: Data Cleaning and Manipulation *continued*

14. I then join the new dataframe to itself and replace the values in the'Round_1_Result_Opponent' column with the opponent's pre-ranking score and formated the values as numeric

```
df4<- sqldf("
              SELECT
               g.Player_Name
              ,g.State
              ,g.Pre_Ranking
              ,g.Total_Pts
              ,g.Pair
              ,d.Pre_Ranking_Opp as'Round_1_Opp_Pre'
              ,d1.Pre_Ranking_Opp as'Round_2_Opp_Pre'
              ,d2.Pre_Ranking_Opp as'Round_3_Opp_Pre'
              ,d3.Pre_Ranking_Opp as'Round_4_Opp_Pre'
              ,d4.Pre_Ranking_Opp as'Round_5_Opp_Pre'
              ,d5.Pre_Ranking_Opp as'Round_6_Opp_Pre'
              ,d6.Pre_Ranking_Opp as'Round_7_Opp_Pre'
              FROM df3 g
              left join df3 d on   d.Pair=g.Round_1_Result_Opponent
              left join df3 d1 on d1.Pair=g.Round_2_Result_Opponent
              left join df3 d2 on d2.Pair=g.Round_3_Result_Opponent
              left join df3 d3 on d3.Pair=g.Round_4_Result_Opponent
              left join df3 d4 on d4.Pair=g.Round_5_Result_Opponent
              left join df3 d5 on d5.Pair=g.Round_6_Result_Opponent
              left join df3 d6 on d6.Pair=g.Round_7_Result_Opponent")

#formatting the below volumn values to numeric
df4 <- df4 %>% mutate(
    #fac1 = factor(fac1), fac2 = factor(fac2), fac3 = factor(fac3),
      Round_1_Opp_Pre = as.numeric(Round_1_Opp_Pre)
    , Round_2_Opp_Pre = as.numeric(Round_2_Opp_Pre)
    , Round_3_Opp_Pre = as.numeric(Round_3_Opp_Pre)
    , Round_4_Opp_Pre = as.numeric(Round_4_Opp_Pre)
    , Round_5_Opp_Pre = as.numeric(Round_5_Opp_Pre)
    , Round_6_Opp_Pre = as.numeric(Round_6_Opp_Pre)
    , Round_7_Opp_Pre = as.numeric(Round_7_Opp_Pre)

)
```

Part 2: Data Cleaning and Manipulation *continued*

12. I calculate the average pre-tournament ranking of opponents and wrote the file to csv

```
#calculate the average pre-tournament ranking of opponents
df5 <- df4 %>%
  mutate(Average_Pre_Score_of_Opponents =
           df4<- rowMeans(df4[c(6:12)], na.rm=T))

#including only requested variables
df5 <- df5[c('Player_Name','State','Pre_Ranking','Total_Pts','Average_Pre_Score_of_Opponents')]

#writting file to CSV
write_csv(df5 , 'Data607_Project_1_Gregg_Maloy_20230218.csv')

head(df5[,1:5], 10)
```

```
##      Player_Name State Pre_Ranking Total_Pts Average_Pre_Score_of_Opponents
## 1  JOEL R HENDON    MI        1436       3.0                         1429.571
## 2   MIKE NIKITIN    MI        1604       4.0                         1385.800
## 3      BRIAN LIU    MI        1423       3.0                         1539.167
## 4       JARED GE    MI        1332       3.0                         1149.857
## 5  SIDDHARTH JHA    MI        1355       3.5                         1388.167
## 6    LARRY HODGE    MI        1270       2.0                         1206.167
## 7   DIPANKAR ROY    MI        1564       4.0                         1426.286
## 8      ANVIT RAO    MI        1365       5.0                         1554.143
## 9   DANIEL KHAIN    MI        1382       2.5                         1355.800
## 10   ERIC WRIGHT    MI        1362       2.5                         1392.000
```

Conclusion

Although the my code could have been more elgant, the resulting CSV file is able to be ingested into a SQL database.