Project 1

Jawaid Hakim

2022-09-15

Contents


```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.1
## v readr   2.1.2     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

1 Solution

1.1 Read tournament data

```
input_data <- read.csv("https://raw.githubusercontent.com/himalayahall/DATA607/main/Project1/tournamentinfo.txt")
```

1.2 Data preparation

To make it easier to work with this data let's *unlist* and convert to matrix format.

```
input_matrix <- matrix(unlist(input_data))
```

Looking at the *head* of the input data we notice that the first 3 rows are header rows and don't contain player info.

```
head(input_matrix, n = 10)
```

```
##       [,1]
##  [1,] " Pair | Player Name                     |Total|Round|Round|Round|Round|Round|Round|Round| "
##  [2,] " Num  | USCF ID / Rtg (Pre->Post)       | Pts |  1  |  2  |  3  |  4  |  5  |  6  |  7  | "
##  [3,] "-----------------------------------------------------------------------------------------"
##  [4,] "    1 | GARY HUA                        |6.0  |W  39|W  21|W  18|W  14|W   7|D  12|D   4|"
##  [5,] "   ON | 15445895 / R: 1794   ->1817     |N:2  |W    |B    |W    |B    |W    |B    |W    |"
##  [6,] "-----------------------------------------------------------------------------------------"
##  [7,] "    2 | DAKSHESH DARURI                 |6.0  |W  63|W  58|L   4|W  17|W  16|W  20|W   7|"
##  [8,] "   MI | 14598900 / R: 1553   ->1663     |N:2  |B    |W    |B    |W    |B    |W    |B    |"
##  [9,] "-----------------------------------------------------------------------------------------"
## [10,] "    3 | ADITYA BAJAJ                    |6.0  |L   8|W  61|W  25|W  21|W  11|W  13|W  12|"
```

Let's skip over the first 3 rows.

```
input_matrix <- input_matrix[-1:-3]
```

Next observation is that data for a player is provided in 2 separate rows. First row gives the name of player and games played by them. The second row gives the State and starting rank. For any given player the two rows appear consecutively, followed by a deshed separator line.

Using this observation, we split the input matrix into 2 components.

For the 1st component, player name and games played by them, we start at 1 and scoop up every 3rd row from the input matrix (skipping over the player state/rank and separator line).

```
mPlayersAndGames <- input_matrix[seq(1, length(input_matrix), 3)]
head(mPlayersAndGames, n = 10)
```

```
##  [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|"
##  [6] "    6 | HANSEN SONG                    |5.0  |W  34|D  29|L  11|W  35|D  10|W  27|W  21|"
##  [7] "    7 | GARY DEE SWATHELL              |5.0  |W  57|W  46|W  13|W  11|L   1|W   9|L   2|"
##  [8] "    8 | EZEKIEL HOUGHTON               |5.0  |W   3|W  32|L  14|L   9|W  47|W  28|W  19|"
##  [9] "    9 | STEFANO LEE                    |5.0  |W  25|L  18|W  59|W   8|W  26|L   7|W  20|"
## [10] "   10 | ANVIT RAO                      |5.0  |D  16|L  19|W  55|W  31|D   6|W  25|W  18|"
```

For the 2nd component, player State and initial ranking, we start at 2 and scoop up every 3rd row from the input matrix (skipping over the player name and separator line).

```
mStatesAndRanks <- input_matrix[seq(2, length(input_matrix), 3)]
head(mStatesAndRanks)
```

```
## [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    |"
## [6] "   OH | 15055204 / R: 1686   ->1687     |N:3  |W    |B    |W    |B    |B    |W    |B    |"
```

1.3 Generate static player data

At this point we have the necessary components to generate all static data - player id, name, state, total points, and initial ranking.

```
player_id <- as.integer(str_extract(mPlayersAndGames, '\\d+'))
length(player_id)
```

```
## [1] 64
```

```
player_name <- str_extract(mPlayersAndGames, "[A-Z]+.[A-Z]+")
length(player_name)
```

```
## [1] 64
```

```
player_state <- str_extract(mStatesAndRanks, "[A-Z][A-Z]")
length(player_state)
```

```
## [1] 64
```

```
player_total_points <- as.numeric(str_extract(mPlayersAndGames, "[0-9]+\\.[0-9]"))
length(player_total_points)
```

```
## [1] 64
```

```
player_init_rank <- as.numeric(str_remove(str_extract(mStatesAndRanks, "R:[ ]+[0-9]{1,}"), "R:[ ]+"))
length(player_init_rank)
```

```
## [1] 64
```

1.4 Data preparation for computing average opponent ranking

We also observe that some games do not contain the id of the opposing player. For example, there is no id of the opposing player for games 6 and 7 played by JULIA Similarly, only game 1 has opposing player id for ASHWIN.

```
mPlayersAndGames[60]
```

```
## [1] "   60 | JULIA SHEN                    |1.5  |L   33|L   34|D   45|D   42|L   24|H    |U    |"
```

```
mPlayersAndGames[62]
```

```
## [1] "   62 | ASHWIN BALAJI                    |1.0  |W  55|U    |U    |U    |U    |U    |U    |"
```

Visual inspection of the full data shows missing player id for games with codes [H, U, B, X].

To make downstream processing more robust let's repair missing opposing player ids with 0. After the transformation we observe that the missing values have been filled in with 0.

```
mPlayersAndGames <- str_replace_all(mPlayersAndGames, "\\|([HUBX])([ \t\f\n])+", "|\\1\\2  0")
mPlayersAndGames[60]
```

```
## [1] "   60 | JULIA SHEN                       |1.5  |L  33|L  34|D  45|D  42|L  24|H   0|U   0|"
```

```
mPlayersAndGames[62]
```

```
## [1] "   62 | ASHWIN BALAJI                    |1.0  |W  55|U   0|U   0|U   0|U   0|U   0|U   0|"
```

Now we extract all opposing player ids into a flattened list. Notice there are exactly 64 * 7 ids since we made sure that missing ids were replaced by 0.

```
p_opponent_ids <- as.integer(str_remove(unlist(str_extract_all(mPlayersAndGames, "[A-Z][ ]+[0-9]+")), "
length(p_opponent_ids) == 64 * 7
```

```
## [1] TRUE
```

Scores for exactly 7 games were reported for each player. So we can split opposing player ids into partitions of 7 each.

Index into the resulting list is the player id! For example, player id for ADITYA BAJAJ is 3, so ids of ADITYA's opponents are to be found at index 3.

```
p_opponents <- split(p_opponent_ids,             # Applying split() function
                     cut(seq_along(p_opponent_ids),
                     length(mPlayersAndGames),
                     labels = FALSE))
mPlayersAndGames[3]
```

```
## [1] "    3 | ADITYA BAJAJ                    |6.0  |L   8|W  61|W  25|W  21|W  11|W  13|W  12|"
```

```
p_opponents[3]
```

```
## $`3`
## [1]  8 61 25 21 11 13 12
```

1.5 Calculate average rank of opponents for all players

Now we are ready to calculate the average rank of opponents for each player.

```
player_avg_score <- numeric(length(p_opponents)) # init results vector
cp_curr_id <- 1                                     # current player id
for (opponents in p_opponents)                    # loop over all opponent splits
{
    cp_op_count <- 0                               # init count of opponents for current player
    cp_sum_op_rank <- 0                            # init sum of opposing player ranks
    for (ops_id in opponents) {                    # loop over all opponent
            if (ops_id > 0) {                      # skip missing opponent ids
                cp_op_count <- cp_op_count + 1     # inc opponent count
                cp_sum_op_rank <- cp_sum_op_rank + player_init_rank[ops_id] # sum opposing player rank
            }
    }

    if (cp_op_count > 0) {
        avg_score <- round(cp_sum_op_rank / cp_op_count, 0)  # compute avg rank of opposing players
    }
    else {
        avg_score = 0
    }

    player_avg_score[cp_curr_id] <- avg_score      # store avg rang

    cp_curr_id <- cp_curr_id + 1                   # inc current player id
}
player_avg_score
```

```
##  [1] 1605 1469 1564 1574 1501 1519 1372 1468 1523 1554 1468 1506 1498 1515 1484
## [16] 1386 1499 1480 1426 1411 1470 1300 1214 1357 1363 1507 1222 1522 1314 1144
## [31] 1260 1379 1277 1375 1150 1388 1385 1539 1430 1391 1248 1150 1107 1327 1152
## [46] 1358 1392 1356 1286 1296 1356 1495 1345 1206 1406 1414 1363 1391 1319 1330
## [61] 1327 1186 1350 1263
```

1.6 Create result data frame

Now let's warp all computed attributes into a data frame.

```
df <- data.frame(player_id, player_name, player_state, player_init_rank, player_total_points, player_avg
colnames(df) <- c('ID', 'Name', 'State', 'Initial_Rank', 'Total_Points', 'Avg_Opponent_Pre_Rating')
```

Lets take a look at the final results.

```
df
```

```
##    ID                Name State Initial_Rank Total_Points
## 1   1            GARY HUA    ON         1794          6.0
## 2   2     DAKSHESH DARURI    MI         1553          6.0
## 3   3        ADITYA BAJAJ    MI         1384          6.0
## 4   4           PATRICK H    MI         1716          5.5
## 5   5          HANSHI ZUO    MI         1655          5.5
## 6   6         HANSEN SONG    OH         1686          5.0
## 7   7           GARY DEE    MI         1649          5.0
```

```
## 8   8     EZEKIEL HOUGHTON    MI         1641          5.0
## 9   9          STEFANO LEE    ON         1411          5.0
## 10 10            ANVIT RAO    MI         1365          5.0
## 11 11      CAMERON WILLIAM    MI         1712          4.5
## 12 12            KENNETH J    MI         1663          4.5
## 13 13       TORRANCE HENRY    MI         1666          4.5
## 14 14         BRADLEY SHAW    MI         1610          4.5
## 15 15        ZACHARY JAMES    MI         1220          4.5
## 16 16         MIKE NIKITIN    MI         1604          4.0
## 17 17   RONALD GRZEGORCZYK    MI         1629          4.0
## 18 18        DAVID SUNDEEN    MI         1600          4.0
## 19 19         DIPANKAR ROY    MI         1564          4.0
## 20 20          JASON ZHENG    MI         1595          4.0
## 21 21            DINH DANG    ON         1563          4.0
## 22 22             EUGENE L    MI         1555          4.0
## 23 23             ALAN BUI    ON         1363          4.0
## 24 24            MICHAEL R    MI         1229          4.0
## 25 25     LOREN SCHWIEBERT    MI         1745          3.5
## 26 26              MAX ZHU    ON         1579          3.5
## 27 27       GAURAV GIDWANI    MI         1552          3.5
## 28 28          SOFIA ADINA    MI         1507          3.5
## 29 29     CHIEDOZIE OKORIE    MI         1602          3.5
## 30 30         GEORGE AVERY    ON         1522          3.5
## 31 31         RISHI SHETTY    MI         1494          3.5
## 32 32        JOSHUA PHILIP    ON         1441          3.5
## 33 33              JADE GE    MI         1449          3.5
## 34 34      MICHAEL JEFFERY    MI         1399          3.5
## 35 35         JOSHUA DAVID    MI         1438          3.5
## 36 36        SIDDHARTH JHA    MI         1355          3.5
## 37 37 AMIYATOSH PWNANANDAM    MI          980          3.5
## 38 38            BRIAN LIU    MI         1423          3.0
## 39 39                JOEL R    MI         1436          3.0
## 40 40         FOREST ZHANG    MI         1348          3.0
## 41 41         KYLE WILLIAM    MI         1403          3.0
## 42 42             JARED GE    MI         1332          3.0
## 43 43          ROBERT GLEN    MI         1283          3.0
## 44 44             JUSTIN D    MI         1199          3.0
## 45 45            DEREK YAN    MI         1242          3.0
## 46 46      JACOB ALEXANDER    MI          377          3.0
## 47 47          ERIC WRIGHT    MI         1362          2.5
## 48 48         DANIEL KHAIN    MI         1382          2.5
## 49 49            MICHAEL J    MI         1291          2.5
## 50 50           SHIVAM JHA    MI         1056          2.5
## 51 51       TEJAS AYYAGARI    MI         1011          2.5
## 52 52            ETHAN GUO    MI          935          2.5
## 53 53                JOSE C    MI         1393          2.0
## 54 54          LARRY HODGE    MI         1270          2.0
## 55 55            ALEX KONG    MI         1186          2.0
## 56 56         MARISA RICCI    MI         1153          2.0
## 57 57           MICHAEL LU    MI         1092          2.0
## 58 58         VIRAJ MOHILE    MI          917          2.0
## 59 59              SEAN M    MI          853          2.0
## 60 60           JULIA SHEN    MI          967          1.5
## 61 61        JEZZEL FARKAS    ON          955          1.5
```

```
## 62 62        ASHWIN BALAJI    MI         1530          1.0
## 63 63        THOMAS JOSEPH    MI         1175          1.0
## 64 64               BEN LI    MI         1163          1.0
##    Avg_Opponent_Pre_Rating
## 1                     1605
## 2                     1469
## 3                     1564
## 4                     1574
## 5                     1501
## 6                     1519
## 7                     1372
## 8                     1468
## 9                     1523
## 10                    1554
## 11                    1468
## 12                    1506
## 13                    1498
## 14                    1515
## 15                    1484
## 16                    1386
## 17                    1499
## 18                    1480
## 19                    1426
## 20                    1411
## 21                    1470
## 22                    1300
## 23                    1214
## 24                    1357
## 25                    1363
## 26                    1507
## 27                    1222
## 28                    1522
## 29                    1314
## 30                    1144
## 31                    1260
## 32                    1379
## 33                    1277
## 34                    1375
## 35                    1150
## 36                    1388
## 37                    1385
## 38                    1539
## 39                    1430
## 40                    1391
## 41                    1248
## 42                    1150
## 43                    1107
## 44                    1327
## 45                    1152
## 46                    1358
## 47                    1392
## 48                    1356
## 49                    1286
## 50                    1296
```

```
## 51                    1356
## 52                    1495
## 53                    1345
## 54                    1206
## 55                    1406
## 56                    1414
## 57                    1363
## 58                    1391
## 59                    1319
## 60                    1330
## 61                    1327
## 62                    1186
## 63                    1350
## 64                    1263
```

1.7 Generate output CSV

Now we can generate the output CSV file in the current working directory. No need to generate row names since player ids are more than adequate.

```
write.csv(df, "player_analysis.csv", row.names = FALSE)
```