Project 1

Jawaid Hakim

2022-09-15

Contents

1 Solution

1.1 Read tournament data

Read input data. Since we will be extracting rows/columns we can convert to matrix format for easier downstream processing.

```
input_matrix <- read.csv("https://raw.githubusercontent.com/himalayahall/DATA607/main/Project1/tourname
input_matrix <- matrix(unlist(input_matrix))
```

1.2 Data preparation

Looking at the *head* of the input data we notice 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|"
```

Skip first 3 header rows.

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

Data for a player is provided on 2 rows. First row gives the name of player and games played by them. The second row gives the State and initial rank. For any given player the two rows appear consecutively, followed by a dashed separator line.

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

The 1st component will contain player name, total number of points, and games played by them. We can extract this data by starting at row 1 and scooping up every 3rd row from the input matrix (skipping over 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|"
```

The 2nd component will contain player State and pre-rating. We can extract this data by starting at row 2 and again scooping 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, n = 10)
```

```
##  [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    |"
##  [7] "   MI | 11146376 / R: 1649   ->1673     |N:3  |W    |B    |W    |B    |B    |W    |W    |"
##  [8] "   MI | 15142253 / R: 1641P17->1657P24  |N:3  |B    |W    |B    |W    |B    |W    |W    |"
##  [9] "   ON | 14954524 / R: 1411   ->1564     |N:2  |W    |B    |W    |B    |W    |B    |B    |"
## [10] "   MI | 14150362 / R: 1365   ->1544     |N:3  |W    |W    |B    |B    |W    |B    |W    |"
```

1.3 Generate static player data

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

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

player_name <- str_trim(str_extract(mPlayersAndGames, "[A-Z][^\\|]+")) # assume names start with a lett

player_state <- str_extract(mStatesAndRanks, "[A-Z][A-Z]")            # assume 2-letter abbreviation f

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

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

It's a good idea to spot check our static data by comparing it with the original. Let's print some values from both original and static data to compare visually. Player SOFIA ADINA STANESCU-BELLU has 3.5 total points and 1507 initial rank in original and sanity check. Checks pass for other players as wel so we are good to go!

```
sanity_check <- data.frame(
                    player_name[seq(4, length(mPlayersAndGames) / 2, 4)],
                    player_total_points[seq(4, length(mPlayersAndGames) / 2, 4)],
                    player_pre_rating[seq(4, length(mPlayersAndGames) / 2, 4)]
)

colnames(sanity_check) <- c('name', 'total_points', 'init_rank')

sanity_check
```

```
##                         name total_points init_rank
## 1        PATRICK H SCHILLING          5.5      1716
## 2           EZEKIEL HOUGHTON          5.0      1641
## 3              KENNETH J TACK          4.5      1663
## 4                MIKE NIKITIN          4.0      1604
## 5                 JASON ZHENG          4.0      1595
## 6            MICHAEL R ALDRICH          4.0      1229
## 7 SOFIA ADINA STANESCU-BELLU          3.5      1507
## 8      JOSHUA PHILIP MATHEWS          3.5      1441
```

```
mPlayersAndGames[seq(4, length(mPlayersAndGames) / 2, 4)]
```

```
## [1] "    4 | PATRICK H SCHILLING             |5.5  |W  23|D  28|W   2|W  26|D   5|W  19|D   1|"
## [2] "    8 | EZEKIEL HOUGHTON                |5.0  |W   3|W  32|L  14|L   9|W  47|W  28|W  19|"
## [3] "   12 | KENNETH J TACK                  |4.5  |W  42|W  33|D   5|W  38|H    |D   1|L   3|"
## [4] "   16 | MIKE NIKITIN                    |4.0  |D  10|W  15|H    |W  39|L   2|W  36|U    |"
## [5] "   20 | JASON ZHENG                     |4.0  |L  40|W  49|W  23|W  41|W  28|L   2|L   9|"
## [6] "   24 | MICHAEL R ALDRICH               |4.0  |L  28|L  47|W  43|L  25|W  60|W  44|W  39|"
## [7] "   28 | SOFIA ADINA STANESCU-BELLU      |3.5  |W  24|D   4|W  22|D  19|L  20|L   8|D  36|"
## [8] "   32 | JOSHUA PHILIP MATHEWS           |3.5  |W  61|L   8|W  44|L  18|W  51|D  26|L  13|"
```

```
mStatesAndRanks[seq(4, length(mStatesAndRanks) / 2, 4)]
```

```
## [1] "   MI | 12616049 / R: 1716   ->1744     |N:2  |W    |B    |W    |B    |W    |B    |B    |"
## [2] "   MI | 15142253 / R: 1641P17->1657P24  |N:3  |B    |W    |B    |W    |B    |W    |W    |"
## [3] "   MI | 12681257 / R: 1663   ->1670     |N:3  |W    |B    |W    |B    |     |W    |B    |"
## [4] "   MI | 10295068 / R: 1604   ->1613     |N:3  |B    |W    |     |B    |W    |B    |     |"
## [5] "   MI | 14529060 / R: 1595   ->1569     |N:4  |W    |B    |W    |B    |W    |B    |W    |"
## [6] "   MI | 13469010 / R: 1229   ->1300     |N:4  |B    |W    |B    |B    |W    |W    |B    |"
## [7] "   MI | 14882954 / R: 1507   ->1513     |N:3  |W    |W    |B    |W    |B    |B    |W    |"
## [8] "   ON | 14073750 / R: 1441   ->1433     |N:4  |W    |B    |W    |B    |W    |B    |W    |"
```

1.4 Missing opponent id

We 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 missing values have been replaced by NA.

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

mPlayersAndGames[60]
```

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

```
mPlayersAndGames[62]
```

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

1.5 Data preparation for computing average opponent ranking

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]+|NA)"
```

```
## Warning: NAs introduced by coercion
```

```
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, ADITYA BAJAJ has player id 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
```

1.6 Calculate average rank of opponents for all players

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

- Count the number of opponents, ignoring NA opponent id
- Sum the ranks of all opponents
- Compute average opponent rank

```
opp_count <- map(p_opponents, function(x) sum(!is.na(x)))

opp_rank_sum <- map(p_opponents, function(x) sum(player_pre_rating[x], na.rm = TRUE))

player_avg_score <- map2(opp_rank_sum, opp_count, ~ round(.x / .y, 0))
```

This was my initial attempt to compute average rank. Worked but not very elegant!

```
# 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_pre_rating[ops_id] # sum opposing player ra
#              }
#     }
#
#     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
```

1.7 Create result data frame

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

```
player_avg_score <- unlist(player_avg_score) # unlist

df <- data.frame(player_id,                  # create data.frame
                 player_name,
                 player_state,
                 player_total_points,
                 player_pre_rating,
                 player_avg_score)

colnames(df) <- c('id',                      # change column names
                  'name',
                  'state',
                  'tot_points',
                  'pre_rating',
                  'avg_opponent_pre_rating')
```

Lets take a look at the final results.

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

```
##    id                name state tot_points pre_rating avg_opponent_pre_rating
## 1   1            GARY HUA    ON        6.0       1794                    1605
## 2   2     DAKSHESH DARURI    MI        6.0       1553                    1469
## 3   3        ADITYA BAJAJ    MI        6.0       1384                    1564
## 4   4 PATRICK H SCHILLING    MI        5.5       1716                    1574
## 5   5          HANSHI ZUO    MI        5.5       1655                    1501
## 6   6         HANSEN SONG    OH        5.0       1686                    1519
## 7   7   GARY DEE SWATHELL    MI        5.0       1649                    1372
## 8   8    EZEKIEL HOUGHTON    MI        5.0       1641                    1468
## 9   9         STEFANO LEE    ON        5.0       1411                    1523
## 10 10          ANVIT RAO    MI        5.0       1365                    1554
```

1.8 Generate output CSV

Now we can generate the output CSV file in current working directory. No need to generate row names, player ids already start at 1 and increase in increments of 1.

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