Project 1

Jawaid Hakim

2022-09-20

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/tournamen
input_matrix <- matrix(unlist(input_matrix))
```

1.2 Data preparation

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]
head(input_matrix, n = 6)
```

```
## [1] "    1 | GARY HUA                        |6.0  |W  39|W  21|W  18|W  14|W   7|D  12|D   4|"
## [2] "   ON | 15445895 / R: 1794   ->1817     |N:2  |W    |B    |W    |B    |W    |B    |W    |"
## [3] "-----------------------------------------------------------------------------------------"
## [4] "    2 | DAKSHESH DARURI                 |6.0  |W  63|W  58|L   4|W  17|W  16|W  20|W   7|"
## [5] "   MI | 14598900 / R: 1553   ->1663     |N:2  |B    |W    |B    |W    |B    |W    |B    |"
## [6] "-----------------------------------------------------------------------------------------"
```

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 of the player. For any given player, the 2 data 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 = 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|"
```

The 2nd component will contain player State and pre-rating. We extract this data by starting at row 2 and, as before, scooping up every 3rd row from the input matrix (skipping over the separator line and next player's name).

```
mStatesAndRanks <- input_matrix[seq(2, length(input_matrix), 3)]
head(mStatesAndRanks, n = 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    |"
```

1.3 Generate static player data

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

Note, there are 64 players in data set.

```r
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:[ ]+"))

NROW(player_name)
```

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

It's a good idea to spot check our static data by comparing with the original. Let's print some values from both original and static data to compare visually:

> Full names of all players have been successfully extracted. For example, SOFIA ADINA STANESCU-BELLU
> Player SOFIA ADINA STANESCU-BELLU has 3.5 total points and 1507 pre-rank in original and static data
> Visual checks pass for other players as well so we are good to go!

```r
idx <- c(1, 12, 22, 28, 58, 64)
sanity_check <- data.frame(
                    player_name[idx],
                    player_total_points[idx],
                    player_pre_rating[idx]
)

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

sanity_check
```

```
##                         name total_points init_rank
## 1                   GARY HUA          6.0      1794
## 2             KENNETH J TACK          4.5      1663
## 3           EUGENE L MCCLURE          4.0      1555
## 4 SOFIA ADINA STANESCU-BELLU          3.5      1507
## 5               VIRAJ MOHILE          2.0       917
## 6                     BEN LI          1.0      1163
```

```r
mPlayersAndGames[idx]
```

```
## [1] "    1 | GARY HUA                        |6.0  |W  39|W  21|W  18|W  14|W   7|D  12|D   4|"
## [2] "   12 | KENNETH J TACK                  |4.5  |W  42|W  33|D   5|W  38|H    |D   1|L   3|"
## [3] "   22 | EUGENE L MCCLURE                |4.0  |W  64|D  52|L  28|W  15|H    |L  17|W  40|"
## [4] "   28 | SOFIA ADINA STANESCU-BELLU      |3.5  |W  24|D   4|W  22|D  19|L  20|L   8|D  36|"
## [5] "   58 | VIRAJ MOHILE                    |2.0  |W  31|L   2|L  41|L  23|L  49|B    |L  45|"
## [6] "   64 | BEN LI                          |1.0  |L  22|D  30|L  31|D  49|L  46|L  42|L  54|"
```

```
mStatesAndRanks[idx]
```

```
## [1] "   ON | 15445895 / R: 1794   ->1817     |N:2  |W    |B    |W    |B    |W    |B    |W    |"
## [2] "   MI | 12681257 / R: 1663   ->1670     |N:3  |W    |B    |W    |B    |     |W    |B    |"
## [3] "   MI | 12405534 / R: 1555   ->1529     |N:4  |W    |B    |W    |B    |     |W    |B    |"
## [4] "   MI | 14882954 / R: 1507   ->1513     |N:3  |W    |W    |B    |W    |B    |B    |W    |"
## [5] "   MI | 14700365 / R:  917   -> 941     |     |W    |B    |W    |B    |W    |     |B    |"
## [6] "   MI | 15006561 / R: 1163   ->1112     |     |B    |W    |W    |B    |W    |B    |B    |"
```

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 SHEN. Similarly, only game 1 played by ASHWIN BALAJI has the opposing player id.

```
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 replace missing opposing player ids with NA. 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. Validate that there are exactly 64 * 7 ids (64 players and 7 games per player) 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 opponents list is the player id!

For example, ADITYA BAJAJ has player id 3, so ids of ADITYA's opponents will be found at index 3.

```
p_opponents <- split(p_opponent_ids,
                     cut(seq_along(p_opponent_ids),
                     length(mPlayersAndGames),
                     labels = FALSE))

test_id <- 3                # ADITYA's id

mPlayersAndGames[test_id]   # ADITYA's games
```

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

```
p_opponents[test_id]        # ADITYA's opponents
```

```
## $`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)))  # count all opponent ids that are not NA

opp_prerating_sum <- map(p_opponents,         # sum pre-rating of all opponents
                         function(x) sum(player_pre_rating[x],
                                         na.rm = TRUE))

opp_avg_prerating <- map2(opp_prerating_sum, # compute player average score
                          opp_count,
                          ~ round(.x / .y, 0))

tail(unlist(opp_count,            # opponent count for last 5 players
            use.names = FALSE),
            5)
```

```
## [1] 5 7 1 5 7
```

```r
tail(unlist(opp_prerating_sum,   # sum of opponent pre-ratings
            use.names = FALSE),
            5)
```

```
## [1] 6651 9291 1186 6751 8841
```

```r
tail(unlist(opp_avg_prerating,   # average opponent score
            use.names = FALSE),
            5)
```

```
## [1] 1330 1327 1186 1350 1263
```

1.7 Create result data frame

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

```r
player_opp_avg_prerating <- unlist(opp_avg_prerating) # unlist

result_df <- data.frame(player_id,        # create data.frame
                 player_name,
                 player_state,
                 player_total_points,
                 player_pre_rating,
                 player_opp_avg_prerating)

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

Lets take a look at the final results. As sanity check, notice there is data for 64 players.

```r
NROW(result_df)
```

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

```r
head(result_df, n = 5)
```

```
##   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
```

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.

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