Project 1

Jawaid Hakim

2022-09-21

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

We notice the first 3 rows are headers and do not 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 the 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 each player is provided on 2 consecutive rows, followed by a dashed separator line. 1st row contains the player name, total points, and games played by them. The 2nd row contains the State and initial rank of the player. Using this observation, let's split the input matrix into 2 components.

The 1st component will contain player name, total points, and games played. We can extract this data by starting at row 1 and scooping up every 3rd row from the input matrix.

```
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.

```
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.

```
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 with the original.

Let's print some values from both the original and static data to compare visually. Notice that player names have been successfully extracted. For example, a complex name like SOFIA ADINA STANESCU-BELLU has been extracted.

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

Next, we see SOFIA ADINA STANESCU-BELLU has 3.5 total points and 1507 in the original data. We validate the original data against our extractions. Success!

```
sofia_idx <- 28

mPlayersAndGames[sofia_idx]
```

```
## [1] "   28 | SOFIA ADINA STANESCU-BELLU      |3.5  |W  24|D   4|W  22|D  19|L  20|L   8|D  36|"
```

```
mStatesAndRanks[sofia_idx]
```

```
## [1] "   MI | 14882954 / R: 1507   ->1513     |N:3  |W    |W    |B    |W    |B    |B    |W    |"
```

1.4 Missing opponent id

The following codes may appear in a chess cross table:

```
W - win, worth 1 point

L - lose, worth 0 points

D - draw, worth 0.5 points

B - full point bye, worth 1 point (given to the left-over player when there are an odd number of player

H - half point bye, worth 0.5 points (players can request these when they know they won't be able to ma

X - win by forfeit, worth 1 point

F - lose by forfeit, worth 0 points (and usually results in automatic withdrawal from the rest of the t

U - unplayed game, worth 0 points (in a round robin, this shows up for any games that haven't been play
```

We observe that unplayed games (U), byes (B, H), and forfeits (F, U) will not contain opponents ids.

For example, there is no id of the opposing player for games 5 played by KENNETH J TACK. Game 6 played by VIRAJ MOHILE does not contain the opposing player id. Games 2-7 do not contain opposing player id for ASHWIN BALAJI.

```
kva_idx <- c(12, 58, 62)    # index for KENNETH, VIRAJ, ASHWIN
mPlayersAndGames[kva_idx]
```

```
## [1] "   12 | KENNETH J TACK                  |4.5  |W  42|W  33|D   5|W  38|H    |D   1|L   3|"
## [2] "   58 | VIRAJ MOHILE                    |2.0  |W  31|L   2|L  41|L  23|L  49|B    |L  45|"
## [3] "   62 | ASHWIN BALAJI                   |1.0  |W  55|U    |U    |U    |U    |U    |U    |"
```

Dealing with missing data is always problematic. To make downstream processing more robust, let's replace missing opposing player ids with NA.

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

Validate that missing player ids have indeed been replaced with NA.

```
mPlayersAndGames[kva_idx]
```

```
## [1] "   12 | KENNETH J TACK                  |4.5  |W  42|W  33|D   5|W  38|H  NA|D   1|L   3|"
## [2] "   58 | VIRAJ MOHILE                    |2.0  |W  31|L   2|L  41|L  23|L  49|B  NA|L  45|"
## [3] "   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 there are exactly 64 * 7 ids (64 players, 7 games per player).

```
p_opponent_ids <- as.integer(str_remove(unlist(str_extract_all(mPlayersAndGames, "[A-Z][ ]+([0-9]+|NA)"
```

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

```r
length(p_opponent_ids) == 64 * 7
```

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

We can split opposing player ids into equal sized partitions (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. Visual inspection confirms it is so.

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

aditya_id <- 3                # ADITYA's id

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

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

```r
p_opponents[aditya_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.

```
* Sum the ranks of all opponents, ignore NAs
* Count the number of opponents, ignoring NAs
* Compute average opponent rank
```

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

opp_count <- map(p_opponents,
                 function(x) sum(!is.na(x)))  # for each player, count opponent ids that are not NA

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

Let's validate our transformations. As expected, the count of games with opponent ids for KENNETH J TACK, VIRAJ MOHILE, and ASHWIN BALAJI is 6, 6, and 1, respectively.

```
mPlayersAndGames[kva_idx]
```

```
## [1] "   12 | KENNETH J TACK                  |4.5  |W  42|W  33|D   5|W  38|H  NA|D   1|L   3|"
## [2] "   58 | VIRAJ MOHILE                    |2.0  |W  31|L   2|L  41|L  23|L  49|B  NA|L  45|"
## [3] "   62 | ASHWIN BALAJI                   |1.0  |W  55|U  NA|U  NA|U  NA|U  NA|U  NA|U  NA|"
```

```
d <- unlist(opp_count, use.names = FALSE)         # opponent counts
d[kva_idx]
```

```
## [1] 6 6 1
```

Visual inspection shows that for games played by KENNETH, the opponent ids are 42, 33, 5, 38, NA, 1, and 3.

```
kenneth_idx <- 12
mPlayersAndGames[kenneth_idx]
```

```
## [1] "   12 | KENNETH J TACK                  |4.5  |W  42|W  33|D   5|W  38|H  NA|D   1|L   3|"
```

Let's hand compute the sum and average of KENNETH's opponent rank. The sum of opponent pre-ranks is 9037 and the average is 1506.

```
mPlayersAndGames[kenneth_idx]
```

```
## [1] "   12 | KENNETH J TACK                  |4.5  |W  42|W  33|D   5|W  38|H  NA|D   1|L   3|"
```

```
mStatesAndRanks[c(42, 33, 5, 38, 1, 3)]
```

```
## [1] "   MI | 14462326 / R: 1332   ->1256     |     |B    |W    |B    |B    |W    |W    |B    |"
## [2] "   MI | 14691842 / R: 1449   ->1421     |     |B    |W    |B    |W    |B    |W    |B    |"
## [3] "   MI | 14601533 / R: 1655   ->1690     |N:2  |B    |W    |B    |W    |B    |W    |B    |"
## [4] "   MI | 15108523 / R: 1423   ->1439     |N:4  |W    |B    |W    |W    |     |B    |B    |"
## [5] "   ON | 15445895 / R: 1794   ->1817     |N:2  |W    |B    |W    |B    |W    |B    |W    |"
## [6] "   MI | 14959604 / R: 1384   ->1640     |N:2  |W    |B    |W    |B    |W    |B    |W    |"
```

```
kenneth_sum <- sum(1332, 1449, 1655, 1423, 1794, 1384) # sum of opponent ranks
kenneth_sum
```

```
## [1] 9037
```

```
kenneth_avg <- round(kenneth_sum / 6, 0)                     # avg opponent rank
kenneth_avg
```

```
## [1] 1506
```

Now, let's make sure the hand computed sum/average matches our transformations. Checks out!

```
kenneth_sum == opp_prerating_sum[kenneth_idx]
```

```
##   12
## TRUE
```

```
kenneth_avg == opp_avg_prerating[kenneth_idx]
```

```
##   12
## TRUE
```

1.7 Create result data frame

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

```
player_opp_avg_prerating <- unlist(opp_avg_prerating, use.names = FALSE) # unlist

result_df <- data.frame(player_id,        # create result 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 final sanity check, notice there is data for 64 players and data for KENNETH matches earlier validations.

```
head(result_df, 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
```

```
NROW(result_df)
```

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

```
result_df[kva_idx, ]
```

```
##    id           name state tot_points pre_rating avg_opponent_pre_rating
## 12 12 KENNETH J TACK    MI        4.5       1663                    1506
## 58 58   VIRAJ MOHILE    MI        2.0        917                    1391
## 62 62  ASHWIN BALAJI    MI        1.0       1530                    1186
```

1.8 Generate output CSV

Now we can generate the output CSV file in current working directory.

```
write.csv(result_df, "player_analysis.csv", row.names = FALSE)  # write csv, skip row names since it's
```