Data 607 - Project 1 - Data Cleaning

William Jasmine

2022-09-21

Introduction

The goal of this project is to turn a sample of messy chess tournament data into a readable, ingestible `.csv` format. The `.txt` file containing the raw data can be found at the following link. The desired output of the data is to create a `.csv` file with the following variables: player name, player state, total number of points won, player's pre-tournament ELO rating, and average ELO rating of all the player's opponents.

Import Data from CSV

The first step is to import the data from the location above. The data is initially read in as a single string, and a sample of the raw data is printed out below using the `substr` function:

```
raw_data <- getURL('https://raw.githubusercontent.com/williamzjasmine/CUNY_SPS_DS/master/DATA_607/Proje

print(substr(raw_data, 1, 1000))
```

```
## [1] "-----------------------------------------------------------------------------------------\n Pai
```

Printing out the first 1,000 characters of the texts reveals a bit of a mess, but it is clear that the file is broken up by lines given the presence of the `\n` characters. The code chunk below splits the `raw_data` line by line using the `strsplit()` function to create a character vector called `file_lines`:

```
file_lines <-strsplit(raw_data, split="\n", fixed=TRUE)[[1]]
print(file_lines[1:10])
```

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

Looking at the first 10 lines outputted above, it is clear that the first four lines are header lines and contain no useful information. They are removed in the chunk below:

```
file_lines = file_lines[-1:-4]
file_lines[1:8]
```

```
## [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] "-----------------------------------------------------------------------------------------"
## [7] "    3 | ADITYA BAJAJ                    |6.0  |L   8|W  61|W  25|W  21|W  11|W  13|W  12|"
## [8] "   MI | 14959604 / R: 1384   ->1640     |N:2  |W    |B    |W    |B    |W    |B    |W    |"
```

Printing the first 8 lines of the new `file_lines` vector above confirms that these header lines have now been removed.

Parse File

Now that the text information is in an iterable format based on line, we can go through the `file_lines` vector line-by-by line to determine the information needed. Looking above it appears as though the lines have a particular pattern to them, and exist in groups of three. In this group of three lines:

1. The first line contains the pair number, name, number of points won, the results of each of the player's matches, and the pair number of each opponent they played. For instance, in line [1] above the pair number is 1, the player's name is GARY HUA, he scored 6.0 points in the tournament, and he played seven games (winning against player 39, 21, 18, 14, 7, 12, and 4).
2. The second line contains the players state and pre tournament ranking. For instance in line [2], we see that GARY HUA is from ON (Ontario) and had a pre tournament ELO ranking of 1,794.
3. The third line is a line break made up of `-` characters to separate out the information for the next player.

Based off this pattern outlined above, we can pick out all the relevant information to create the desired `.csv` file. This is done below:

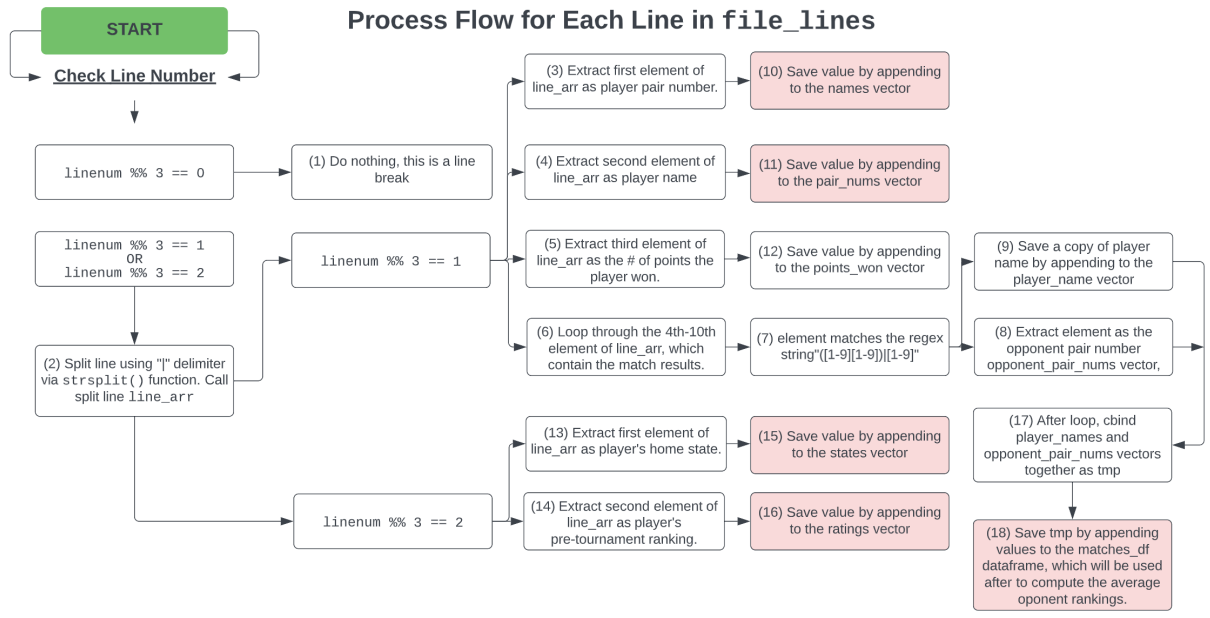
```
matches_df <- data.frame(matrix(ncol = 2, nrow = 0))
names = c()
pair_nums = c()
states = c()
points = c()
ratings = c()

for (line_num in 1:length(file_lines)){
  if (line_num %% 3 == 0){
    next # (1)
  }
  else {
    line_arr <- strsplit(file_lines[line_num], "|", fixed = TRUE) # (2)
    #print(line_arr)
    if (line_num %% 3 == 1) {
      pair_num <- as.integer(
        gsub(line_arr[[1]][1], pattern = " ", replacement = "")) # (3)
      name <- str_extract(line_arr[[1]][2], '[A-Z].*[A-Z]') # (4)
      points_won =  as.numeric(
        gsub(line_arr[[1]][3], pattern = " ", replacement = "")) # (5)
      opponent_pair_nums = c()
      for (col_num in 4:10){ # (6)
        opponent_pair_num = str_extract(line_arr[[1]][col_num],
                                        '([1-9][0-9])|[0-9]') # (7)
        if (is.na(opponent_pair_num) == FALSE) { # (7)
          opponent_pair_nums = append(as.integer(opponent_pair_num),
                                      opponent_pair_nums) # (8)
        }
```

```
      }
      player_name = rep(name, length(opponent_pair_nums)) # (9)
      tmp = cbind.data.frame(player_name, opponent_pair_nums) # (17)
      matches_df = rbind(matches_df, tmp) # (18)
      pair_nums = append(pair_nums, pair_num) # (10)
      names = append(names, name) # (11)
      points = append(points, points_won) # (12)
    }
    else if (line_num %% 3 == 2) {
      state <- gsub(line_arr[[1]][1], pattern = " ", replacement = "") # (13)
      pre_rating <- as.integer(
          gsub(gsub(str_extract(line_arr[[1]][2], ' \\d{3,4}(( )|P)'),
                    pattern=" ", replacement=''),
               pattern='P', replacement='')) # (14)
      states = append(states, state)
      ratings = append(ratings, pre_rating)
    }
    else {
      print('Something went wrong')
    }
  }

}
```

The code above contains a lot of different `if` statements and `for` loops, making it a little difficult to understand the logical flow on the surface. As such, the following graphic provides a flowchart of what it performs:



.

The flowchart describes the logical flow of the code, but doesn't specify the exact regex matching or string manipulation that was used to extract the necessary values. In order to view this detail, the numbers in the flowchart cells correspond to the commented numbers in the above code specifying exactly where each action took place.

The red cells indicate the times in which desired values are saved to various vectors to be used in the following code. Five of these vectors (`names`, `pair_nums`, `states`, `points`, and `ratings`) are combined below into what will be our final dataframe (`final_df`) using the `cbind` function. Appropriate column names are also given

to this dataframe.

```
final_df = cbind.data.frame(names, pair_nums, states, points, ratings)
colnames(final_df) <- c('player_name', 'pair_number',
                        'state', 'points_won', 'pre_rating')
colnames(matches_df) <- c('player_name', 'pair_number')

head(final_df)
```

```
##           player_name pair_number state points_won pre_rating
## 1            GARY HUA           1    ON        6.0       1794
## 2     DAKSHESH DARURI           2    MI        6.0       1553
## 3        ADITYA BAJAJ           3    MI        6.0       1384
## 4 PATRICK H SCHILLING           4    MI        5.5       1716
## 5          HANSHI ZUO           5    MI        5.5       1655
## 6         HANSEN SONG           6    OH        5.0       1686
```

The output above shows that our data is now very close to being in the desired format: the only column now missing is the average rating of each player's opponent.

Calc Average Ratings

To calculate the average rating of each opponent, we can use the `matches_df` dataframe created in the cell chunk containing the primary parsing code. This dataframe contains a record of all the games that were played, namely a list of each player name and all of their associated opponent pair numbers. This can also be clarified in the graphic above by following the logical flow after cell (6). The chunk below calculates the average rating of each opponent using a `groupby()`:

```
avg_opponent_score_df <-
  matches_df %>%
    left_join(final_df, by = "pair_number", suffix=c("", "_y")) %>%
      group_by(player_name) %>%
        summarise(avg_opponent_score=mean(pre_rating))
```

The results are stored in a dataframe called `avg_opponent_score_df`.

Putting it All Together

The final step to having all the required data is to join our `final_df` to the newly created `avg_opponent_score_df` and pulling the average opponent score. This is done below:

```
final_df <-
  final_df %>%
    left_join(avg_opponent_score_df, by='player_name')

final_df <- final_df[-2]
final_df
```

```
##                   player_name state points_won pre_rating avg_opponent_score
## 1                    GARY HUA    ON        6.0       1794           1605.286
## 2              DAKSHESH DARURI    MI        6.0       1553           1469.286
## 3                 ADITYA BAJAJ    MI        6.0       1384           1563.571
## 4          PATRICK H SCHILLING    MI        5.5       1716           1573.571
## 5                   HANSHI ZUO    MI        5.5       1655           1500.857
## 6                  HANSEN SONG    OH        5.0       1686           1518.714
```

```
## 7           GARY DEE SWATHELL    MI       5.0       1649          1372.143
## 8            EZEKIEL HOUGHTON    MI       5.0       1641          1468.429
## 9                 STEFANO LEE    ON       5.0       1411          1523.143
## 10                  ANVIT RAO    MI       5.0       1365          1554.143
## 11   CAMERON WILLIAM MC LEMAN    MI       4.5       1712          1467.571
## 12             KENNETH J TACK    MI       4.5       1663          1506.167
## 13          TORRANCE HENRY JR    MI       4.5       1666          1497.857
## 14              BRADLEY SHAW    MI       4.5       1610          1515.000
## 15     ZACHARY JAMES HOUGHTON    MI       4.5       1220          1483.857
## 16               MIKE NIKITIN    MI       4.0       1604          1385.800
## 17         RONALD GRZEGORCZYK    MI       4.0       1629          1498.571
## 18              DAVID SUNDEEN    MI       4.0       1600          1480.000
## 19              DIPANKAR ROY    MI       4.0       1564          1426.286
## 20               JASON ZHENG    MI       4.0       1595          1410.857
## 21              DINH DANG BUI    ON       4.0       1563          1470.429
## 22           EUGENE L MCCLURE    MI       4.0       1555          1300.333
## 23                  ALAN BUI    ON       4.0       1363          1213.857
## 24          MICHAEL R ALDRICH    MI       4.0       1229          1357.000
## 25           LOREN SCHWIEBERT    MI       3.5       1745          1363.286
## 26                  MAX ZHU     ON       3.5       1579          1506.857
## 27             GAURAV GIDWANI    MI       3.5       1552          1221.667
## 28 SOFIA ADINA STANESCU-BELLU    MI       3.5       1507          1522.143
## 29           CHIEDOZIE OKORIE    MI       3.5       1602          1313.500
## 30         GEORGE AVERY JONES    ON       3.5       1522          1144.143
## 31               RISHI SHETTY    MI       3.5       1494          1259.857
## 32      JOSHUA PHILIP MATHEWS    ON       3.5       1441          1378.714
## 33                  JADE GE     MI       3.5       1449          1276.857
## 34     MICHAEL JEFFERY THOMAS    MI       3.5       1399          1375.286
## 35           JOSHUA DAVID LEE    MI       3.5       1438          1149.714
## 36              SIDDHARTH JHA    MI       3.5       1355          1388.167
## 37       AMIYATOSH PWNANANDAM    MI       3.5        980          1384.800
## 38                  BRIAN LIU    MI       3.0       1423          1539.167
## 39              JOEL R HENDON    MI       3.0       1436          1429.571
## 40              FOREST ZHANG     MI       3.0       1348          1390.571
## 41        KYLE WILLIAM MURPHY    MI       3.0       1403          1248.500
## 42                  JARED GE     MI       3.0       1332          1149.857
## 43          ROBERT GLEN VASEY    MI       3.0       1283          1106.571
## 44         JUSTIN D SCHILLING    MI       3.0       1199          1327.000
## 45                  DEREK YAN    MI       3.0       1242          1152.000
## 46   JACOB ALEXANDER LAVALLEY    MI       3.0        377          1357.714
## 47                ERIC WRIGHT    MI       2.5       1362          1392.000
## 48               DANIEL KHAIN    MI       2.5       1382          1355.800
## 49           MICHAEL J MARTIN    MI       2.5       1291          1285.800
## 50                SHIVAM JHA     MI       2.5       1056          1296.000
## 51            TEJAS AYYAGARI    MI       2.5       1011          1356.143
## 52                 ETHAN GUO    MI       2.5        935          1494.571
## 53              JOSE C YBARRA    MI       2.0       1393          1345.333
## 54               LARRY HODGE     MI       2.0       1270          1206.167
## 55                  ALEX KONG    MI       2.0       1186          1406.000
## 56               MARISA RICCI    MI       2.0       1153          1414.400
## 57                 MICHAEL LU    MI       2.0       1092          1363.000
## 58              VIRAJ MOHILE     MI       2.0        917          1391.000
## 59          SEAN M MC CORMICK    MI       2.0        853          1319.000
## 60                 JULIA SHEN    MI       1.5        967          1330.200
```

```
## 61              JEZZEL FARKAS    ON        1.5        955           1327.286
## 62              ASHWIN BALAJI    MI        1.0       1530           1186.000
## 63       THOMAS JOSEPH HOSMER    MI        1.0       1175           1350.200
## 64                     BEN LI    MI        1.0       1163           1263.000
```

And that's it! The text file has now been transformed into an R dataframe containing the desired columns. This R dataframe is converted into a `.csv` in the cell below, and can be used for further analysis.

```
write.csv(final_df,"tournamentinfo.csv", row.names = FALSE)
```