Project 1 Seung Min Song

Seung Min Song

2022-09-21

Team member: Ted Kim

GitHub: https://github.com/seung-m1nsong/607

Read the text file by line using the readlines( ) function.

```
file <- 'https://raw.githubusercontent.com/seung-m1nsong/607/main/Project1/tournamentinfo.txt'
#file <- './7645617.txt'
res <- readLines(file, warn=FALSE)
res[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] "-----------------------------------------------------------------------------------------"
```

Use the setdiff( ) function to exclude all border lines from the array. When the borderline is directly entered in the setdiff( ) function, the first borderline is not excluded for unknown reasons. Therefore, after clearing all the border lines using the str_replace_all( ) function, an object with a value of empty is removed.

```
res <- res %>%
        str_remove('\\-{2,}') %>%
        setdiff('')
head(res)
```

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

Remove strings except for the player's state and the pre-score from the lower row. Remove all of the first lower row, which is part of the header because it creates unnecessary columns when converted to data frames.

```
res <- res %>%
        str_replace_all('^\\sN.*|[0-9]{8}.*R\\:|(?<=\\d{3})(?:P|\\s).*', '')
head(res)
```

```
## [1] " Pair | Player Name                     |Total|Round|Round|Round|Round|Round|Round|Round| "
## [2] ""
## [3] "    1 | GARY HUA                        |6.0  |W  39|W  21|W  18|W  14|W   7|D  12|D   4|"
## [4] "   ON |  1794"
## [5] "    2 | DAKSHESH DARURI                 |6.0  |W  63|W  58|L   4|W  17|W  16|W  20|W   7|"
## [6] "   MI |  1553"
```

In the upper row, replace character (B, H, U, X, L, W, D), which have no value between the letter '|' and the letter '|', with NA. It then removes only the characters (B, H, U, X, L, W, D) that are with the values under the same conditions.

```
res[33]
```

```
## [1] "   16 | MIKE NIKITIN                   |4.0  |D  10|W  15|H    |W  39|L   2|W  36|U    |"
```

```
res <- res %>%
        str_replace_all('(?<=\\|)(B|H|U|X|L|W|D)\\s+(?=\\|)', 'NA') %>%
        str_remove_all('(?<=\\|)(B|H|U|X|L|W|D)\\s+')
res[31:36]
```

```
## [1] "   15 | ZACHARY JAMES HOUGHTON          |4.5  |19|16|30|22|54|33|38|"
## [2] "   MI |  1220"
## [3] "   16 | MIKE NIKITIN                    |4.0  |10|15|NA|39|2|36|NA|"
## [4] "   MI |  1604"
## [5] "   17 | RONALD GRZEGORCZYK              |4.0  |48|41|26|2|23|22|5|"
## [6] "   MI |  1629"
```

Use the seq_len() function to divide the upper and lower rows into two arrays, then separate values between "|" and "|" by columns. Since the upper row has an odd number as a sequence, such as 1, 3, and 5, its remainder is 1 when divided by 2. Conversely, the lower row remainder is 0.

```
seq <- seq_len(length(res)) %% 2

upper <- res[seq == 1] %>%
          str_extract_all('\\d+(?:\\.\\d{1})?|[a-zA-Z-]+(?:\\s[a-zA-Z-]+)*', simplify = TRUE)
head(upper)
```

```
##      [,1]   [,2]                 [,3]    [,4]    [,5]    [,6]    [,7]
## [1,] "Pair" "Player Name"        "Total" "Round" "Round" "Round" "Round"
## [2,] "1"    "GARY HUA"           "6.0"   "39"    "21"    "18"    "14"
## [3,] "2"    "DAKSHESH DARURI"    "6.0"   "63"    "58"    "4"     "17"
## [4,] "3"    "ADITYA BAJAJ"       "6.0"   "8"     "61"    "25"    "21"
## [5,] "4"    "PATRICK H SCHILLING" "5.5"  "23"    "28"    "2"     "26"
## [6,] "5"    "HANSHI ZUO"         "5.5"   "45"    "37"    "12"    "13"
##      [,8]    [,9]    [,10]
## [1,] "Round" "Round" "Round"
## [2,] "7"     "12"    "4"
## [3,] "16"    "20"    "7"
## [4,] "11"    "13"    "12"
## [5,] "5"     "19"    "1"
## [6,] "4"     "14"    "17"
```

```
lower = res[seq == 0] %>%
          str_extract_all('\\d+|[A-Z]{2}', simplify = TRUE)
#lower <- as.data.frame(lower)
head(lower)
```

```
##      [,1] [,2]
## [1,] ""   ""
## [2,] "ON" "1794"
## [3,] "MI" "1553"
## [4,] "MI" "1384"
## [5,] "MI" "1716"
## [6,] "MI" "1655"
```

The upper and lower rows are combined using the cbind( ) function. And use the slice( ) function to remove the header part that is not removed for the operation. Then, define the name of the columns.

```
df <- as.data.frame(cbind(upper, lower)) %>%
        slice(2:n())
colnames(df) <- c('Pair_Num', 'Player_Name', 'Total_Number_of_Point',
                  'R_1', 'R_2', 'R_3', 'R_4', 'R_5', 'R_6', 'R_7',
                  'Player_State', 'Player_Pre')
head(df)
```

```
##   Pair_Num         Player_Name Total_Number_of_Point R_1 R_2 R_3 R_4 R_5 R_6
## 1        1            GARY HUA                  6.0  39  21  18  14   7  12
## 2        2     DAKSHESH DARURI                  6.0  63  58   4  17  16  20
## 3        3        ADITYA BAJAJ                  6.0   8  61  25  21  11  13
## 4        4 PATRICK H SCHILLING                  5.5  23  28   2  26   5  19
## 5        5          HANSHI ZUO                  5.5  45  37  12  13   4  14
## 6        6         HANSEN SONG                  5.0  34  29  11  35  10  27
##   R_7 Player_State Player_Pre
## 1   4           ON       1794
## 2   7           MI       1553
## 3  12           MI       1384
## 4   1           MI       1716
## 5  17           MI       1655
```

```
## 6  21           OH       1686
```

Use the left_join( ) function to get the opponent's pre-score for each round, For calculation, the opponent's pre-score data type is converted into a numerical type.

```
for(i in 1:7) {
  df1 <- df %>%
         select(Pair_Num, Player_Pre)
  df1$Player_Pre <- as.numeric(df1$Player_Pre)
  colnames(df1) = c(paste0('R_', i), paste0('Player_Pre_', i))
  print(head(df1))
  df <- df %>%
          left_join(df1)
}
```

```
##   R_1 Player_Pre_1
## 1   1         1794
## 2   2         1553
## 3   3         1384
## 4   4         1716
## 5   5         1655
## 6   6         1686

## Joining, by = "R_1"

##   R_2 Player_Pre_2
## 1   1         1794
## 2   2         1553
## 3   3         1384
## 4   4         1716
## 5   5         1655
## 6   6         1686

## Joining, by = "R_2"

##   R_3 Player_Pre_3
```

```
## 1   1         1794
## 2   2         1553
## 3   3         1384
## 4   4         1716
## 5   5         1655
## 6   6         1686
```

```
## Joining, by = "R_3"
```

```
##   R_4 Player_Pre_4
## 1   1         1794
## 2   2         1553
## 3   3         1384
## 4   4         1716
## 5   5         1655
## 6   6         1686
```

```
## Joining, by = "R_4"
```

```
##   R_5 Player_Pre_5
## 1   1         1794
## 2   2         1553
## 3   3         1384
## 4   4         1716
## 5   5         1655
## 6   6         1686
```

```
## Joining, by = "R_5"
```

```
##   R_6 Player_Pre_6
## 1   1         1794
## 2   2         1553
## 3   3         1384
## 4   4         1716
## 5   5         1655
## 6   6         1686
```

```
## Joining, by = "R_6"
```

```
##   R_7 Player_Pre_7
## 1   1         1794
## 2   2         1553
## 3   3         1384
## 4   4         1716
## 5   5         1655
## 6   6         1686
```

```
## Joining, by = "R_7"
```

```
head(df)
```

```
##   Pair_Num         Player_Name Total_Number_of_Point R_1 R_2 R_3 R_4 R_5 R_6
## 1        1            GARY HUA                  6.0  39  21  18  14   7  12
## 2        2     DAKSHESH DARURI                  6.0  63  58   4  17  16  20
## 3        3        ADITYA BAJAJ                  6.0   8  61  25  21  11  13
## 4        4 PATRICK H SCHILLING                  5.5  23  28   2  26   5  19
## 5        5          HANSHI ZUO                  5.5  45  37  12  13   4  14
## 6        6         HANSEN SONG                  5.0  34  29  11  35  10  27
##   R_7 Player_State Player_Pre Player_Pre_1 Player_Pre_2 Player_Pre_3
## 1   4           ON       1794         1436         1563         1600
## 2   7           MI       1553         1175          917         1716
## 3  12           MI       1384         1641          955         1745
## 4   1           MI       1716         1363         1507         1553
## 5  17           MI       1655         1242          980         1663
## 6  21           OH       1686         1399         1602         1712
##   Player_Pre_4 Player_Pre_5 Player_Pre_6 Player_Pre_7
## 1         1610         1649         1663         1716
## 2         1629         1604         1595         1649
## 3         1563         1712         1666         1663
## 4         1579         1655         1564         1794
## 5         1666         1716         1610         1629
## 6         1438         1365         1552         1563
```

Calculate column Player_Pre_1 to Player_Per_7(indexed 13 to 19) average and append the value to a newly created column named 'Avg_Pre'.

```
df <- df %>%
        mutate(Avg_Pre=rowMeans(.[13:19], na.rm=TRUE))
head(df)
```

```
##   Pair_Num         Player_Name Total_Number_of_Point R_1 R_2 R_3 R_4 R_5 R_6
## 1        1            GARY HUA                  6.0  39  21  18  14   7  12
## 2        2     DAKSHESH DARURI                  6.0  63  58   4  17  16  20
## 3        3        ADITYA BAJAJ                  6.0   8  61  25  21  11  13
## 4        4 PATRICK H SCHILLING                  5.5  23  28   2  26   5  19
## 5        5          HANSHI ZUO                  5.5  45  37  12  13   4  14
## 6        6         HANSEN SONG                  5.0  34  29  11  35  10  27
##   R_7 Player_State Player_Pre Player_Pre_1 Player_Pre_2 Player_Pre_3
## 1   4           ON       1794         1436         1563         1600
## 2   7           MI       1553         1175          917         1716
## 3  12           MI       1384         1641          955         1745
## 4   1           MI       1716         1363         1507         1553
## 5  17           MI       1655         1242          980         1663
## 6  21           OH       1686         1399         1602         1712
##   Player_Pre_4 Player_Pre_5 Player_Pre_6 Player_Pre_7  Avg_Pre
## 1         1610         1649         1663         1716 1605.286
## 2         1629         1604         1595         1649 1469.286
## 3         1563         1712         1666         1663 1563.571
## 4         1579         1655         1564         1794 1573.571
## 5         1666         1716         1610         1629 1500.857
## 6         1438         1365         1552         1563 1518.714
```

Select fields 'Player_Name', 'Player_State', 'Total_Number_of_Point', 'Player_Pre', and 'Avg_Pre' and put those data into the final data frame to export a csv file.

```
df_final <- df %>%
        select('Player_Name','Player_State','Total_Number_of_Point','Player_Pre','Avg_Pre')
df_final
```

```
##                  Player_Name Player_State Total_Number_of_Point Player_Pre
## 1                   GARY HUA           ON                   6.0       1794
## 2             DAKSHESH DARURI           MI                   6.0       1553
```

3	ADITYA BAJAJ	MI	6.0	1384
4	PATRICK H SCHILLING	MI	5.5	1716
5	HANSHI ZUO	MI	5.5	1655
6	HANSEN SONG	OH	5.0	1686
7	GARY DEE SWATHELL	MI	5.0	1649
8	EZEKIEL HOUGHTON	MI	5.0	1641
9	STEFANO LEE	ON	5.0	1411
10	ANVIT RAO	MI	5.0	1365
11	CAMERON WILLIAM MC LEMAN	MI	4.5	1712
12	KENNETH J TACK	MI	4.5	1663
13	TORRANCE HENRY JR	MI	4.5	1666
14	BRADLEY SHAW	MI	4.5	1610
15	ZACHARY JAMES HOUGHTON	MI	4.5	1220
16	MIKE NIKITIN	MI	4.0	1604
17	RONALD GRZEGORCZYK	MI	4.0	1629
18	DAVID SUNDEEN	MI	4.0	1600
19	DIPANKAR ROY	MI	4.0	1564
20	JASON ZHENG	MI	4.0	1595
21	DINH DANG BUI	ON	4.0	1563
22	EUGENE L MCCLURE	MI	4.0	1555
23	ALAN BUI	ON	4.0	1363
24	MICHAEL R ALDRICH	MI	4.0	1229
25	LOREN SCHWIEBERT	MI	3.5	1745
26	MAX ZHU	ON	3.5	1579
27	GAURAV GIDWANI	MI	3.5	1552
28	SOFIA ADINA STANESCU-BELLU	MI	3.5	1507
29	CHIEDOZIE OKORIE	MI	3.5	1602
30	GEORGE AVERY JONES	ON	3.5	1522
31	RISHI SHETTY	MI	3.5	1494
32	JOSHUA PHILIP MATHEWS	ON	3.5	1441
33	JADE GE	MI	3.5	1449
34	MICHAEL JEFFERY THOMAS	MI	3.5	1399
35	JOSHUA DAVID LEE	MI	3.5	1438
36	SIDDHARTH JHA	MI	3.5	1355
37	AMIYATOSH PWNANANDAM	MI	3.5	980
38	BRIAN LIU	MI	3.0	1423

```
## 39              JOEL R HENDON          MI                3.0       1436
## 40               FOREST ZHANG          MI                3.0       1348
## 41        KYLE WILLIAM MURPHY          MI                3.0       1403
## 42                  JARED GE          MI                3.0       1332
## 43          ROBERT GLEN VASEY          MI                3.0       1283
## 44         JUSTIN D SCHILLING          MI                3.0       1199
## 45                  DEREK YAN          MI                3.0       1242
## 46   JACOB ALEXANDER LAVALLEY          MI                3.0        377
## 47               ERIC WRIGHT          MI                2.5       1362
## 48              DANIEL KHAIN          MI                2.5       1382
## 49           MICHAEL J MARTIN          MI                2.5       1291
## 50                SHIVAM JHA          MI                2.5       1056
## 51             TEJAS AYYAGARI          MI                2.5       1011
## 52                  ETHAN GUO          MI                2.5        935
## 53              JOSE C YBARRA          MI                2.0       1393
## 54               LARRY HODGE           MI                2.0       1270
## 55                  ALEX KONG          MI                2.0       1186
## 56              MARISA RICCI           MI                2.0       1153
## 57                MICHAEL LU           MI                2.0       1092
## 58              VIRAJ MOHILE           MI                2.0        917
## 59          SEAN M MC CORMICK          MI                2.0        853
## 60                JULIA SHEN           MI                1.5        967
## 61              JEZZEL FARKAS          ON                1.5        955
## 62              ASHWIN BALAJI          MI                1.0       1530
## 63       THOMAS JOSEPH HOSMER          MI                1.0       1175
## 64                    BEN LI           MI                1.0       1163
##     Avg_Pre
## 1  1605.286
## 2  1469.286
## 3  1563.571
## 4  1573.571
## 5  1500.857
## 6  1518.714
## 7  1372.143
## 8  1468.429
## 9  1523.143
```

```
## 10 1554.143
## 11 1467.571
## 12 1506.167
## 13 1497.857
## 14 1515.000
## 15 1483.857
## 16 1385.800
## 17 1498.571
## 18 1480.000
## 19 1426.286
## 20 1410.857
## 21 1470.429
## 22 1300.333
## 23 1213.857
## 24 1357.000
## 25 1363.286
## 26 1506.857
## 27 1221.667
## 28 1522.143
## 29 1313.500
## 30 1144.143
## 31 1259.857
## 32 1378.714
## 33 1276.857
## 34 1375.286
## 35 1149.714
## 36 1388.167
## 37 1384.800
## 38 1539.167
## 39 1429.571
## 40 1390.571
## 41 1248.500
## 42 1149.857
## 43 1106.571
## 44 1327.000
## 45 1152.000
```

```
## 46 1357.714
## 47 1392.000
## 48 1355.800
## 49 1285.800
## 50 1296.000
## 51 1356.143
## 52 1494.571
## 53 1345.333
## 54 1206.167
## 55 1406.000
## 56 1414.400
## 57 1363.000
## 58 1391.000
## 59 1319.000
## 60 1330.200
## 61 1327.286
## 62 1186.000
## 63 1350.200
## 64 1263.000
```

Exports final data frame to CSV file.

```
write.csv(df_final, file='chess_rating.csv')
```