Elo calculations

waheeb Algabri

project introduction In this project, you're given a text file with chess tournament results where the information has some structure. Your job is to create an R Markdown file that generates a .CSV file (that could for example be imported into a SQL database) with the following information for all of the players: Player's Name, Player's State, Total Number of Points, Player's Pre-Rating, and Average Pre Chess Rating of Opponents For the first player, the information would be: Gary Hua, ON, 6.0, 1794, 1605 1605 was calculated by using the pre-tournament opponents' ratings of 1436, 1563, 1600, 1610, 1649, 1663, 1716, and dividing by the total number of games played. If you have questions about the meaning of the data or the results, please post them on the discussion forum. Data science, like chess, is a game of back and forth... The chess rating system (invented by a Minnesota statistician named Arpad Elo) has been used in many other contexts, including assessing relative strength of employment candidates by human resource departments. You may substitute another text file (or set of text files, or data scraped from web pages) of similar or greater complexity, and create your own assignment and solution. You may work in a small team. All of your code should be in an R markdown file (and published to rpubs.com); with your data accessible for the person running the script.

```
library(stringr)
library(ggplot2)
library(tidyverse)
```

Loading the data I,m going to use ./ in front of the file to make my codes more explicit and portable.

```
tournment<- ("./tournamentinfo.txt")
waheeb<- readLines(tournment)
head(waheeb, 7)
```

```
## [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] "-----------------------------------------------------------------------------------------"
```

```
# remove first 4 rows that I don't need
con <- waheeb[-c(0:4)]
```

```r
# remove unnecessary spaces
con <- con[sapply(con, nchar) > 0]
```

```r
# divide odd / even rows into separate set of lines
odd <- c(seq(1, length(con), 3))
odd_a <- con[odd]

even <- c(seq(2, length(con), 3))
even_a <- con[even]
```

Data transformation I will use regex to extract the only required information.

```r
# name
name <- str_extract(odd_a, "\\s+([[:alpha:]- ]+)\\b\\s*\\|")
name <- gsub(name, pattern = "|", replacement = "", fixed = T)
# strip the space
name <- trimws(name)

# state
state <- str_extract(even_a, "[[:alpha:]]{2}")

# total_points
total_points <- str_extract(odd_a, "[[:digit:]]+\\.[[:digit:]]")
total_points <- as.numeric(as.character(total_points))

# pre_rating
pre_rating <- str_extract(even_a, ".\\: \\s?[[:digit:]]{3,4}")
pre_rating <- gsub(pre_rating, pattern = "R: ", replacement = "", fixed = T)
pre_rating <- as.numeric(as.character(pre_rating))

# opponent_number to extract opponents pair number per player
opponent_number <- str_extract_all(odd_a, "[[:digit:]]{1,2}\\|")
opponent_number <- str_extract_all(opponent_number, "[[:digit:]]{1,2}")
opponent_number <- lapply(opponent_number, as.numeric)
```

calculate Average Pre Chess Rating of Opponents and store that in a list.

```r
opp_avg_rating <- list()
for (i in 1:length(opponent_number)){
  opp_avg_rating[i] <- round(mean(pre_rating[unlist(opponent_number[i])]),2)
}
opp_avg_rating <- lapply(opp_avg_rating, as.numeric)
opp_avg_rating <- data.frame(unlist(opp_avg_rating))
head(opp_avg_rating)
```

```
##   unlist.opp_avg_rating.
## 1                1605.29
## 2                1469.29
## 3                1563.57
```

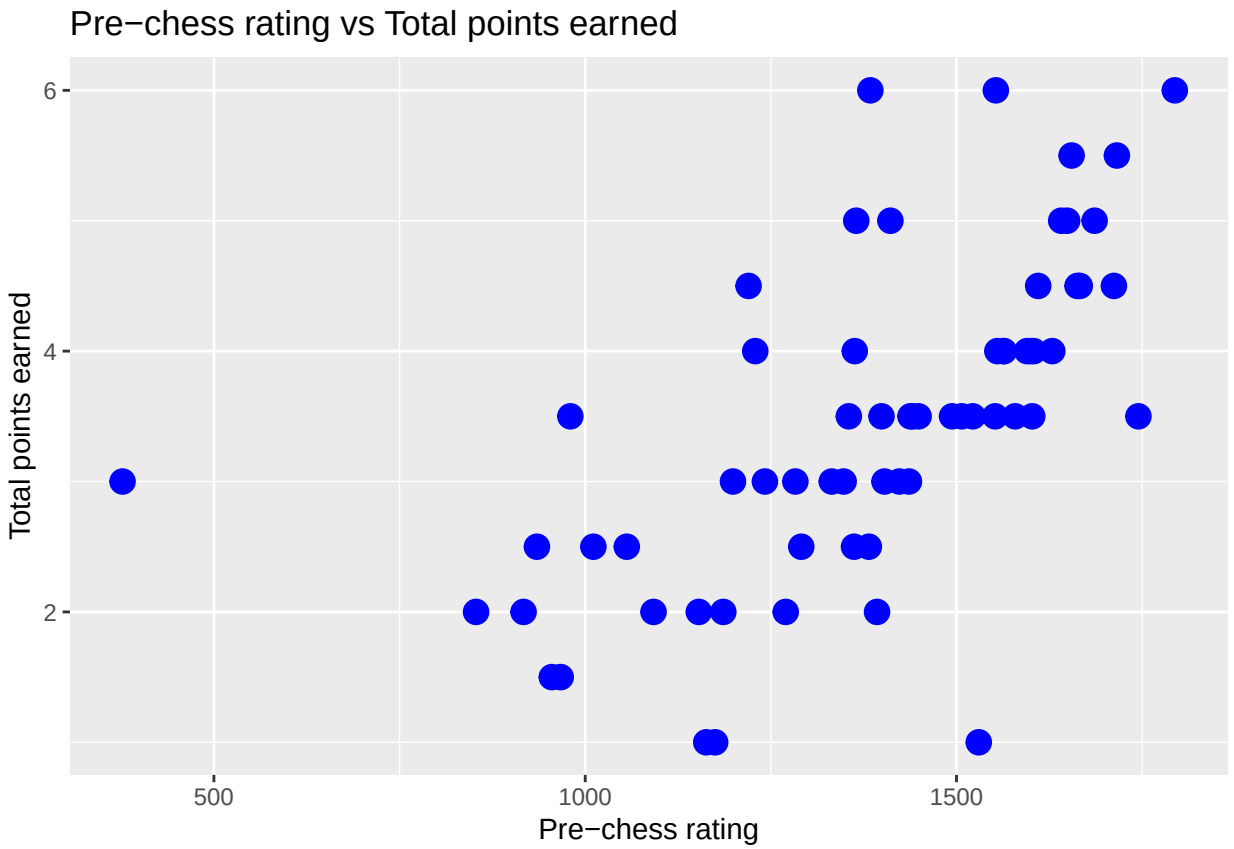
```
## 4                1573.57
## 5                1500.86
## 6                1518.71
```

create data frame

```
df <- cbind.data.frame(name, state, total_points, pre_rating, opp_avg_rating)
colnames(df) <- c("Name", "State", "Total_points", "Pre_rating", "Avg_pre_chess_rating_of_opponents")
head(df)
```

```
##                  Name State Total_points Pre_rating
## 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
##   Avg_pre_chess_rating_of_opponents
## 1                           1605.29
## 2                           1469.29
## 3                           1563.57
## 4                           1573.57
## 5                           1500.86
## 6                           1518.71
```

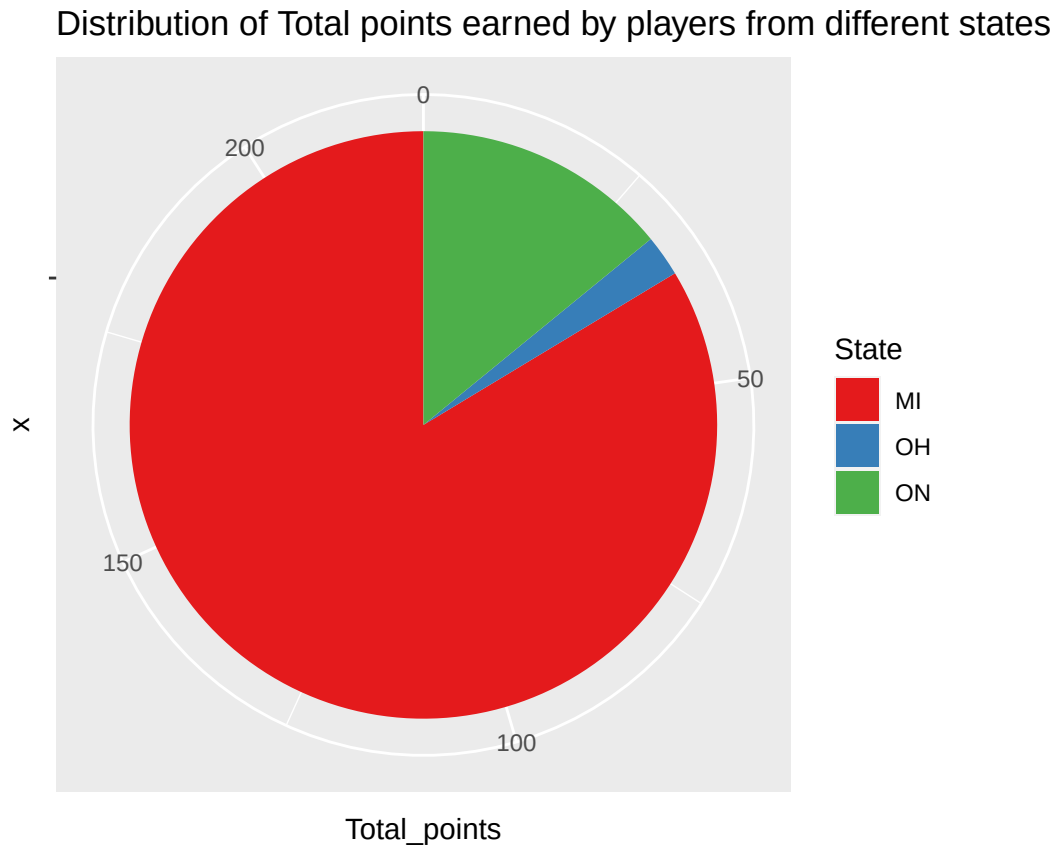
```
ggplot(data = df, aes(x = Pre_rating, y = Total_points)) +
  geom_point(size = 4, color = "blue") +
  ggtitle("Pre-chess rating vs Total points earned") +
  xlab("Pre-chess rating") +
  ylab("Total points earned")
```



Visualization

```
df_state_points <- df %>% group_by(State) %>%
  summarize(Total_points = sum(Total_points))

ggplot(data = df_state_points, aes(x = "", y = Total_points, fill = State)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar("y", start = 0) +
  ggtitle("Distribution of Total points earned by players from different states") +
  labs(fill = "State") +
  scale_fill_brewer(palette = "Set1")
```




```
write.csv(df, "chess_ratings.csv")
```

Based on difference in ratings between the chess players and each of their opponents in our Project 1 tournament, calculate each player's expected score (e.g. 4.3) and the difference from their actual score (e.g 4.0). List the five players who most overperformed relative to their expected score, and the five players that most underperformed relative to their expected score. You'll find some small differences in different implementation of ELO formulas.You may use any reasonably-sourced formula, but please cite your source.

```
# Import the data from CSV file
chess_data <- read.csv("chess_ratings.csv")
```

```
# Calculate the performance ratings for each player
chess_data$perf_rating <- chess_data$Pre_rating + (chess_data$Total_points - 3) * 100

# Calculate the average performance rating of opponents for each player
opponents <- subset(chess_data, select=c("Name", "Avg_pre_chess_rating_of_opponents"))
colnames(opponents) <- c("Opponent", "Avg_rating")
chess_data$Opponent_rating <- merge(chess_data, opponents, by.x="Name", by.y="Opponent")$Avg_rating
chess_data$avg_perf_opponents <- round(mean(chess_data$Opponent_rating), 2)

# Print the output
head(chess_data)
```

```
##   X                Name State Total_points Pre_rating
## 1 1            GARY HUA    ON          6.0       1794
## 2 2     DAKSHESH DARURI    MI          6.0       1553
## 3 3        ADITYA BAJAJ    MI          6.0       1384
## 4 4 PATRICK H SCHILLING    MI          5.5       1716
## 5 5          HANSHI ZUO    MI          5.5       1655
## 6 6         HANSEN SONG    OH          5.0       1686
##   Avg_pre_chess_rating_of_opponents perf_rating Opponent_rating
## 1                           1605.29        2094         1563.57
## 2                           1469.29        1853         1213.86
## 3                           1563.57        1684         1406.00
## 4                           1573.57        1966         1384.80
## 5                           1500.86        1905         1554.14
## 6                           1518.71        1886         1186.00
##   avg_perf_opponents
## 1             1378.6
## 2             1378.6
## 3             1378.6
## 4             1378.6
## 5             1378.6
## 6             1378.6
```

```
# calculate expected score and difference from actual score for each player
chess_data$expected_score <- 1 / (1 + 10^((chess_data$Opponent_rating - chess_data$perf_rating)/400))
chess_data$score_diff <- chess_data$Total_points - chess_data$expected_score
```

```
# sort by overperformers
overperformers <- head(chess_data[order(chess_data$score_diff, decreasing = TRUE), ], 5)
```

```
# sort by underperformers
underperformers <- head(chess_data[order(chess_data$score_diff), ], 5)
```

```
# print results
cat("Five players who most overperformed relative to their expected score:\n")
```

```
## Five players who most overperformed relative to their expected score:
```

```
print(overperformers)
```

```
##   X                Name State Total_points Pre_rating
## 3 3        ADITYA BAJAJ    MI          6.0       1384
## 1 1            GARY HUA    ON          6.0       1794
## 2 2     DAKSHESH DARURI    MI          6.0       1553
## 5 5          HANSHI ZUO    MI          5.5       1655
## 4 4 PATRICK H SCHILLING    MI          5.5       1716
##   Avg_pre_chess_rating_of_opponents perf_rating Opponent_rating
## 3                           1563.57        1684         1406.00
## 1                           1605.29        2094         1563.57
## 2                           1469.29        1853         1213.86
## 5                           1500.86        1905         1554.14
## 4                           1573.57        1966         1384.80
##   avg_perf_opponents expected_score score_diff
```

```
## 3             1378.6      0.8320598   5.167940
## 1             1378.6      0.9549291   5.045071
## 2             1378.6      0.9753780   5.024622
## 5             1378.6      0.8828513   4.617149
## 4             1378.6      0.9659623   4.534038
```

```
cat("\nFive players who most underperformed relative to their expected score:\n")
```

```
##
## Five players who most underperformed relative to their expected score:
```

```
print(underperformers)
```

```
##     X                 Name State Total_points Pre_rating
## 62 62        ASHWIN BALAJI    MI          1.0       1530
## 63 63 THOMAS JOSEPH HOSMER    MI          1.0       1175
## 64 64               BEN LI    MI          1.0       1163
## 53 53        JOSE C YBARRA    MI          2.0       1393
## 60 60           JULIA SHEN    MI          1.5        967
##    Avg_pre_chess_rating_of_opponents perf_rating Opponent_rating
## 62                           1186.00        1330         1497.86
## 63                           1350.20         975         1391.00
## 64                           1263.00         963         1483.86
## 53                           1345.33        1293         1106.57
## 60                           1330.20         817         1356.14
##    avg_perf_opponents expected_score score_diff
## 62             1378.6     0.27562263  0.7243774
## 63             1378.6     0.08357862  0.9164214
## 64             1378.6     0.04750223  0.9524978
## 53             1378.6     0.74520069  1.2547993
## 60             1378.6     0.04296150  1.4570385
```

- You may use any reasonably-sourced formula, but please cite your source.

This is the website which I got some useful steps for the Elo rating system

FiveThirtyEight