# Data 607 - Project 1

Cameron Smith

9/17/2020

## Introduction

This project was focused on the analysis of chess tournament data. Using cross-tables from a text file we needed to extract the key data using regex, transform it into a usable form, and then analyze specific elements including in particular the pre-ratings of each player and their opponents.

### Load library and read File

The Tidyverse is used in order to leverage its filtering and regex capabilities. For reproducibility I placed the text file in Github, read directly into R.

```
# Load required libraries
library(tidyverse)
```

```
## -- Attaching packages -----------------------------------------------------------------------------

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.1     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts -----------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# Get the data
raw_data <- readLines("https://raw.githubusercontent.com/cwestsmith/cuny-msds/master/datasets/tournament
```

```
## Warning in readLines("https://raw.githubusercontent.com/cwestsmith/cuny-
## msds/master/datasets/tournamentinfo.txt"): incomplete final line found on
## 'https://raw.githubusercontent.com/cwestsmith/cuny-msds/master/datasets/
## tournamentinfo.txt'
```

### Extract key fields

Key fields were extracted using the str_extract_all function. The extracted data was then put into a data frame.

```r
# Extract the required data using regular expressions
player_num <- as.numeric(unlist(str_extract_all(raw_data,"(?<=\\s{3,4})\\d{1,2}(?=\\s)")))
player_name <- unlist(str_extract_all(raw_data,"(?<=\\d\\s\\|\\s)([A-z, -]*\\s){1,}[[:alpha:]]*(?=\\s*\\
player_state <- unlist(str_extract_all(raw_data, "[[:upper:]]{2}(?=\\s\\|)"))
total_pts <- as.numeric(unlist(str_extract_all(raw_data, "(?<=\\|)\\d\\.\\d")))
player_pre_rat <- as.numeric(unlist(str_extract_all(raw_data, "(?<=R:\\s{1,2})(\\d{3,4}(?=\\s))|(\\d{3,4

# Take the extracted data and put it into a data frame
processed_data <- data.frame(player_num, player_name, player_state, total_pts, player_pre_rat)

# Check the data frame's structure to make sure it is as intended (i.e. number columns are numeric, cha
str(processed_data)
```

```
## 'data.frame':    64 obs. of  5 variables:
##  $ player_num    : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ player_name   : Factor w/ 64 levels "ADITYA BAJAJ                    ",..: 24 12 1 51 28 27 23 21
##  $ player_state  : Factor w/ 3 levels "MI","OH","ON": 3 1 1 1 1 2 1 1 3 1 ...
##  $ total_pts     : num  6 6 6 5.5 5.5 5 5 5 5 5 ...
##  $ player_pre_rat: num  1794 1553 1384 1716 1655 ...
```

### Create list of opponent player numbers

The next step was to extract the opponent player numbers, including opponents for which matches did not materialize (bys and forfeits, for example) which brought with it some challenges due to the format of the file. A combination of approaches was used to extract the data.

```r
# Had some initial challenges doing this with only regex so to make it simpler and a bit more robust I
secondary_rows <- raw_data[seq(5, 196, 3)]
opponent_num <- as.numeric(unlist(str_extract_all(secondary_rows, "(?<=\\|(W|L|D)\\s{2,3})[[:digit:]]{1
```

### Calculate the average pre chess rating (PCR) of each player's opponents

For this step I used a for loop to take sequences of each 7 entries (corresponding with the 7 chess matches for each person), add them together for the total opponents' score, then divide them by the number of entries that were not NA in that sequence to find the average.

```r
# Create matrix to store data calculated in the for loop.  Pre-populating values with NA for more effic
pcr_matrix <- matrix(data = NA, nrow = 64, ncol = 2)

# Assign readable names for the matrix
colnames(pcr_matrix) <- c("total_opp_pcr", "avg_opp_pcr")

# Initialize a variable to be used as a counter in the for loop to fill the corresponding matrix row
row_counter <- 0

# Start of for loop
for(i in seq(from=1, to=length(opponent_num)-6, by=7)){
  row_counter <- row_counter + 1

# Perform a lookup of each competitor's score based on their player number and add the up for each row
  pcr_matrix[row_counter, 1] <- (sum(subset(processed_data$player_pre_rat, processed_data$player_num %i
# Calculate the average score for each row, excluding missing entries
  pcr_matrix[row_counter, 2] <- pcr_matrix[row_counter, 1] / length(subset(opponent_num[seq(from=i, to=

}
# End of for loop

# Verify that matrix was processed properly by looking at the first few rows of output
head(pcr_matrix, 5)
```

```
##      total_opp_pcr avg_opp_pcr
## [1,]         11237    1605.286
## [2,]         10285    1469.286
## [3,]         10945    1563.571
## [4,]         11015    1573.571
## [5,]         10506    1500.857
```

### Final tidying up

The average PCRs were then rounded up to the nearest full point and added to the data frame with the other processed data.

```
# Round the figures to the nearest whole number
pcr_matrix[, 2] <- round(pcr_matrix[,2], digits = 0)

# Add average scores to data frame with other processed data and rename for readability
processed_data <- cbind(processed_data, pcr_matrix[, 2])
processed_data <- rename(processed_data, avg_opp_pcr = `pcr_matrix[, 2]`)
```

### Export of .csv file

The final data frame of processed data was then exported into a .csv file for ease of reference. A platform independent path has been used to ensure full reproducibility.

```
# Get working directory path
path <- getwd()

# Export file to working directory.  The file.path function has been used to ensure platform independen
write.csv(processed_data, file.path(path, "chess_processed_data.csv"))
```

## Conclusion

Using regex in combination with various Tidyverse functions it was possible to extract the data needed from a text file with a somewhat irregular structure. That data could then be fed into a data frame for ease of analysis, using a for loop to process complex elements and minimize the lines of code required. The complete dataframe with all processed data could then be exported into a .CSV file for further reference if/as needed. A sample of the complete dataframe is below for reference.

```r
head(processed_data, 5)
```

```
##   player_num                      player_name player_state total_pts
## 1          1 GARY HUA                                   ON       6.0
## 2          2 DAKSHESH DARURI                            MI       6.0
## 3          3 ADITYA BAJAJ                               MI       6.0
## 4          4 PATRICK H SCHILLING                        MI       5.5
## 5          5 HANSHI ZUO                                 MI       5.5
##   player_pre_rat avg_opp_pcr
## 1           1794        1605
## 2           1553        1469
## 3           1384        1564
## 4           1716        1574
## 5           1655        1501
```