Week 3 Project 1

Hope (Amal) Jasentuliyana

2023-09-27

Observations of csv datafile:

Record-level (line by line)

- Every 4th line, starting at line 1 is dashes ——
- The first four lines [1-4] can be discarded (headers and dashes)
- Line 5 starts the data. Call this line n
- Line n + 1 contains the State & pre-rating
- Line n + 2 is dashes and can be discarded.
- Line n + 4 will be the start of the next record
- As above a record in a dataframe contains data from 2 lines from the csv

first pass:

Assuming there are no abberant records, it looks like I can: 1. delete every 4th line 2. join every odd line with the successive line (delete carriage return) 3. separate on | and trim() to get fields

```
if (! require(readr)) {
  install.packages("readr")
  library(readr)
}
```

```
## Loading required package: readr
```

```
if (! require(stringr)) {
  install.packages("stringr")
  library(stringr)
}
```

```
## Loading required package: stringr
```

```
if (! require(dplyr)) {
  install.packages("dplyr")
  library(dplyr)
}
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
if (! require(dplyr)) {
  install.packages("dplyr")
  library(dplyr)
}
df <- read_csv("./project1.txt", col_names = FALSE)
```

```
## Rows: 196 Columns: 1
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): X1
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
df
```

```
## # A tibble: 196 x 1
##    X1
##    <chr>
##  1 ---------------------------------------------------------------------------~
##  2 Pair | Player Name                     |Total|Round|Round|Round|Round|Round|~
##  3 Num  | USCF ID / Rtg (Pre->Post)       | Pts |  1  |  2  |  3  |  4  |  5  |~
##  4 ---------------------------------------------------------------------------~
##  5 1 | GARY HUA                          |6.0  |W  39|W  21|W  18|W  14|W   7|D  ~
##  6 ON | 15445895 / R: 1794   ->1817      |N:2  |W    |B    |W    |B    |W    |B  ~
##  7 ---------------------------------------------------------------------------~
##  8 2 | DAKSHESH DARURI                   |6.0  |W  63|W  58|L   4|W  17|W  16|W  ~
##  9 MI | 14598900 / R: 1553   ->1663      |N:2  |B    |W    |B    |W    |B    |W  ~
## 10 ---------------------------------------------------------------------------~
## # i 186 more rows
```

```r
# I was going to use str_replace_all() to remove the dashes, but
# since it's a dataframe, I can filter mod(3) != 1.


head(df)
```

```
## # A tibble: 6 x 1
##   X1
##   <chr>
## 1 ---------------------------------------------------------------------------~
## 2 Pair | Player Name                     |Total|Round|Round|Round|Round|Round|R~
## 3 Num  | USCF ID / Rtg (Pre->Post)       | Pts |  1  |  2  |  3  |  4  |  5  | ~
## 4 ---------------------------------------------------------------------------~
## 5 1 | GARY HUA                          |6.0  |W  39|W  21|W  18|W  14|W   7|D  1~
## 6 ON | 15445895 / R: 1794   ->1817      |N:2  |W    |B    |W    |B    |W    |B  ~
```

```r
tail(df)
```

```
## # A tibble: 6 x 1
##   X1
##   <chr>
## 1 63 | THOMAS JOSEPH HOSMER            |1.0  |L   2|L  48|D  49|L  43|L  45|H  ~
## 2 MI | 15057092 / R: 1175   ->1125      |     |W    |B    |W    |B    |B    |   ~
## 3 ---------------------------------------------------------------------------~
## 4 64 | BEN LI                           |1.0  |L  22|D  30|L  31|D  49|L  46|L  ~
```

```
## 5 MI | 15006561 / R: 1163   ->1112     |     |B    |W    |W    |B    |W    |B  ~
## 6 ---------------------------------------------------------------------------~
```

```
#196 rows, which start and end on -----
# after filtering I should have (2 * 196 / 3 )- 1  rows

df2 <- df[1:nrow(df) %% 3 != 1, ]
# OR using dplyr
df3 <- df %>% filter(row_number() %% 3 != 1)

stopifnot(nrow(df2) != (2 * nrow(df)/3 -1))


#stopifnot(identical(df2,df3))  #assertion failed!
stopifnot(nrow(setdiff(df2,df3)) == 0)

#at this point, the dataframes have an equal number of rows,
#and the no rows in df2 are not also in df3.
# since they aren't identical, the order must be different?




head(df3)
```

```
## # A tibble: 6 x 1
##   X1
##   <chr>
## 1 Pair | Player Name                    |Total|Round|Round|Round|Round|Round|R~
## 2 Num  | USCF ID / Rtg (Pre->Post)      | Pts |  1  |  2  |  3  |  4  |  5  | ~
## 3 1 | GARY HUA                         |6.0  |W  39|W  21|W  18|W  14|W   7|D  1~
## 4 ON | 15445895 / R: 1794   ->1817     |N:2  |W    |B    |W    |B    |W    |B  ~
## 5 2 | DAKSHESH DARURI                  |6.0  |W  63|W  58|L   4|W  17|W  16|W  2~
## 6 MI | 14598900 / R: 1553   ->1663     |N:2  |B    |W    |B    |W    |B    |W  ~
```

```
tail(df3)
```

```
## # A tibble: 6 x 1
##   X1
##   <chr>
## 1 62 | ASHWIN BALAJI                   |1.0  |W  55|U    |U    |U    |U    |U  ~
## 2 MI | 15219542 / R: 1530   ->1535     |     |B    |     |     |     |     |   ~
## 3 63 | THOMAS JOSEPH HOSMER            |1.0  |L   2|L  48|D  49|L  43|L  45|H  ~
## 4 MI | 15057092 / R: 1175   ->1125     |     |W    |B    |W    |B    |B    |   ~
## 5 64 | BEN LI                          |1.0  |L  22|D  30|L  31|D  49|L  46|L  ~
## 6 MI | 15006561 / R: 1163   ->1112     |     |B    |W    |W    |B    |W    |B  ~
```

At this point I am going to take a different approach and just do regex substitutions.

```
myRegex = "^-+$"  # field consisting only of one or more dashes
df4 <- df[!grepl("^-+$",df$X1),]

stopifnot(nrow(df4)==nrow(df2))
stopifnot(identical(df2,df4))

View(df4)  #df2 an df4 have the same data. df3 may have  a different ordering.
```

```
# now concatenate even rows to odd ones.
df4odd <- df4[1:nrow(df4) %% 2 == 1,]
df4even <- df4[1:nrow(df4) %% 2 ==0,]
df4even <- df4even %>% rename(X2 = X1)
df5 <- cbind(df4odd,df4even)

stopifnot(nrow(df5)== nrow(df4odd))

df5 <- df5 %>% mutate(X3=paste0(X1,X2)) %>% select(X3)
View(df5)
```

Now split on the "|" delimeter ### Field-level parsing

```
1. Field 1: Player/pair number [will be used to join with other players to get average pre-rating of op
2. Field 2: Name  [text]
3. Field 3: Rating [decimal, 1 digit after decimal]
4. Field 4-10: [W,L,D]\s+(\d+) opponent_number. opponent_number is integer.
5. Field 11: State. [A-Z]{2}.  [text]
6. Field 12: [0-9]{8}\s+/R:\s+(\d+) pre-rating  [integer]
7. Fields 13-20: unused for this analysis
```

```
library(tidyr)


# the following did not work due to a terminal delimeter "|"
if (FALSE) {
  newColNames <- c("playerNum",
                   "playerName",
                   "playerRating",
                   str_c("opponentOutcome",1:7),
                   "State_Province",
                   "preRating",
                   str_c("unused",1:8)

                   )



  df5Split <- df5 %>% separate_wider_delim(X3,delim="|",
        names = newColNames,too_many="debug")
}

newColNames <- c("playerNum",
                 "playerName",
                 "playerRating",
                 str_c("opponentOutcome",1:7),
                 "State_Province",
                 "preRating",
                 str_c("unused",1:8),
                "terminalDelimeter"
                 )

df5Split <- df5 %>% separate_wider_delim(X3,delim="|",
       names = newColNames)
```

Parse the opponent outcomes:

```
df5Split2 <- df5Split %>% mutate(across(starts_with("opponentOutcome"),
            ~str_match(str_trim(.),"[W,L,D]\\s+(\\d+)")[,2] ))

df5Split3 <- df5Split2 %>%
    mutate(preRating = str_match(preRating,
                "\\d+\\s+/\\s+R:\\s+(\\d+)\\D")[,2])

# remove the header row
df5Split3 <- df5Split3[2:nrow(df5Split),]
```

Now for each row and opponent we need to join the opponent outcome to the pre rating of the opponent.

Or if we get a vector of opponent numbers, we could get the average pre-rating of matching opponents.

```
#the join below was failing due to spaces in the playerNum
df5Split3$playerNum <- trimws(df5Split3$playerNum, which = "both")

finalDf <- df5Split3 %>% rowwise() %>% mutate( avgOpponentRating = {
  opponentIDs <- c(across(starts_with("opponentOutcome"))) #vector of IDs
  opponentRatings <- df5Split3$preRating[match(opponentIDs,
                                        df5Split3$playerNum)]
  opponentRatings <- as.numeric(opponentRatings)
  mean(opponentRatings,na.rm=TRUE)

}) %>% ungroup()
View(finalDf)
```

Final cleaning and saving goes here.