Data 607 Project One: Transform .txt File into Structured CSV

Kevin Kirby

2024-09-17

Overview

This is a project to take a provided txt file of Chess players, their stats, and performance in a chess tournment and produce a table that gets export to CSV. The following fields are required in the final output:

- Player's Name
- Player's State
- Total Points
- Player's Pre-Rating
- Average Pre Chess Rating of Opponents
  - This is the average pre-rating all of opponents they played in a round

```
library(RPostgres)
library(DBI)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(purrr)
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
##
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

Postgres connection

The next chunk is what my Postgres SQL connection looked like. I ran the code to produce the PDF and then went back in and removed my actual username and password. You will need to replace with your own PostgresSQL credentials for it to work.

```
pgs_db <- dbConnect(
  RPostgres::Postgres(),
```

```
  host = "localhost",
  port = 5432,
  dbname = "data_607",
  user = pgs_username,
  password = pgs_pw
)
```

Data Import from Google Cloud Platform

I downloaded the data from the file provided on Brightspace and uploaded it as a txt file to my Google Cloud Platform instance. Clicking on this link will cause the file to be downloaded to your computer.

```
tourny_txt_gcp <- "https://storage.googleapis.com/data_science_masters_files/2024_fall/data_607_data_ma
tournament_txt <- readLines(tourny_txt_gcp)
```

```
## Warning in readLines(tourny_txt_gcp): incomplete final line found on
## 'https://storage.googleapis.com/data_science_masters_files/2024_fall/data_607_data_management/projec
```

Initialize dataframe and parse txt file

This chunk:

- Initializes a dataframe with required fields, including what I'll need to calculate the average
- Parses out name, pre rating, state, and total points from the lines
- For each of the round columns, it pulls in the value that represents the row number of the opponent for that round

At the end, it's ready for the next chunk that does a version of an index() match() to bring in the right value for the round.

This part was tricky until I realized that each player has two lines of data, with a line of buffer on each side. Once I figured out that the first player I needed started at line four

```
tourny_df <- data.frame(
  name = character(),
  pre_rating = numeric(),
  state = character(),
  total_points = numeric(),
  round_one = numeric(),
  round_two = numeric(),
  round_three = numeric(),
  round_four = numeric(),
  round_five = numeric(),
  round_six = numeric(),
  round_seven = numeric(),
  stringsAsFactors = FALSE
)

for (i in seq(5, length(tournament_txt), by = 3)) {
  line_one <- tournament_txt[i]
  line_two <- tournament_txt[i + 1]

  if (grepl("\\|", line_one) && grepl("\\|", line_two)) {
    parts_line_one <- strsplit(line_one, "\\|")[[1]]
    parts_line_two <- strsplit(line_two, "\\|")[[1]]
    player_name <- trimws(parts_line_one[2])
```

```
    total_pts <- as.numeric(trimws(parts_line_one[3]))
    round_scores <- sapply(parts_line_one[4:10], function(x) {
      score <- as.numeric(gsub("[^0-9.]", "", x))
      return(ifelse(is.na(score), NA, score))
    })
    state <- trimws(parts_line_two[1])
    rating <- as.numeric(gsub(".*R: *([0-9]+).*", "\\1", parts_line_two[2]))
    new_row <- data.frame(
      name = player_name,
      pre_rating = rating,
      state = state,
      total_points = total_pts,
      round_one = round_scores[1],
      round_two = round_scores[2],
      round_three = round_scores[3],
      round_four = round_scores[4],
      round_five = round_scores[5],
      round_six = round_scores[6],
      round_seven = round_scores[7],
      stringsAsFactors = FALSE
    )
    tourny_df <- rbind(tourny_df, new_row)
  }
}
```

Looking Up Ratings Opponents

I decided on a very methodical approach to doing the lookup us the opponent ratings. I'm pretty new to regular expressions in general and wanted to have an explicit column for each player's opponent for each round and then the pre-rating.

This uses the value round_[number] from above and looks up that row number and pulls in the prerating for that opponent. It ends up creating a dataframe that has two columns per round: one for the row number and one for the pre-rating of that row number.

```
tourny_df$pre_rating <- as.numeric(as.character(tourny_df$pre_rating))
lookup_rating <- function(round_value) {
  if (!is.na(round_value) && round_value > 0 && round_value <= nrow(tourny_df)) {
    return(tourny_df$pre_rating[round_value])
  } else {
    return(NA)
  }
}

tourny_df$Rating_round_one <- sapply(tourny_df$round_one, lookup_rating)
tourny_df$Rating_round_two <- sapply(tourny_df$round_two, lookup_rating)
tourny_df$Rating_round_three <- sapply(tourny_df$round_three, lookup_rating)
tourny_df$Rating_round_four <- sapply(tourny_df$round_four, lookup_rating)
tourny_df$Rating_round_five <- sapply(tourny_df$round_five, lookup_rating)
tourny_df$Rating_round_six <- sapply(tourny_df$round_six, lookup_rating)
tourny_df$Rating_round_seven <- sapply(tourny_df$round_seven, lookup_rating)

tourny_df$opponent_pre_avg <- rowMeans(tourny_df[, c("Rating_round_one", "Rating_round_two", "Rating_ro
```

```
tourny_df <- tourny_df[, c("name", "pre_rating", "state", "total_points", "opponent_pre_avg")]
```

Final dataframe and Export

This chunk takes the final tourney_df and trims it down to the required fields for the export to a SQL table.

```
rownames(tourny_df) <- NULL

tourny_df <- tourny_df[, c("name", "pre_rating", "state", "total_points", "opponent_pre_avg")]
```

This sends it to my postgres SQL instance and ends with writing it as a CSV to my DATA 607 folder that syncs with my Github:

```
dbExecute(pgs_db, "SET search_path TO all_tables;")
```

```
## [1] 0
```

```
create_query <- "
CREATE TABLE IF NOT EXISTS all_tables.tournaments_txt_table (
    name TEXT,
    pre_rating NUMERIC,
    state TEXT,
    total_points NUMERIC,
    opponent_pre_avg NUMERIC
);
"
dbExecute(pgs_db, create_query)
```

```
## NOTICE:  relation "tournaments_txt_table" already exists, skipping
```

```
## [1] 0
```

```
dbWriteTable(pgs_db, name = "tournaments_txt_table", value = tourny_df, overwrite = TRUE, row.names = F

write.csv(tourny_df, "/Users/kevinkirby/Desktop/github_sync/data_science_masters_work/2024_Fall/data_60
```

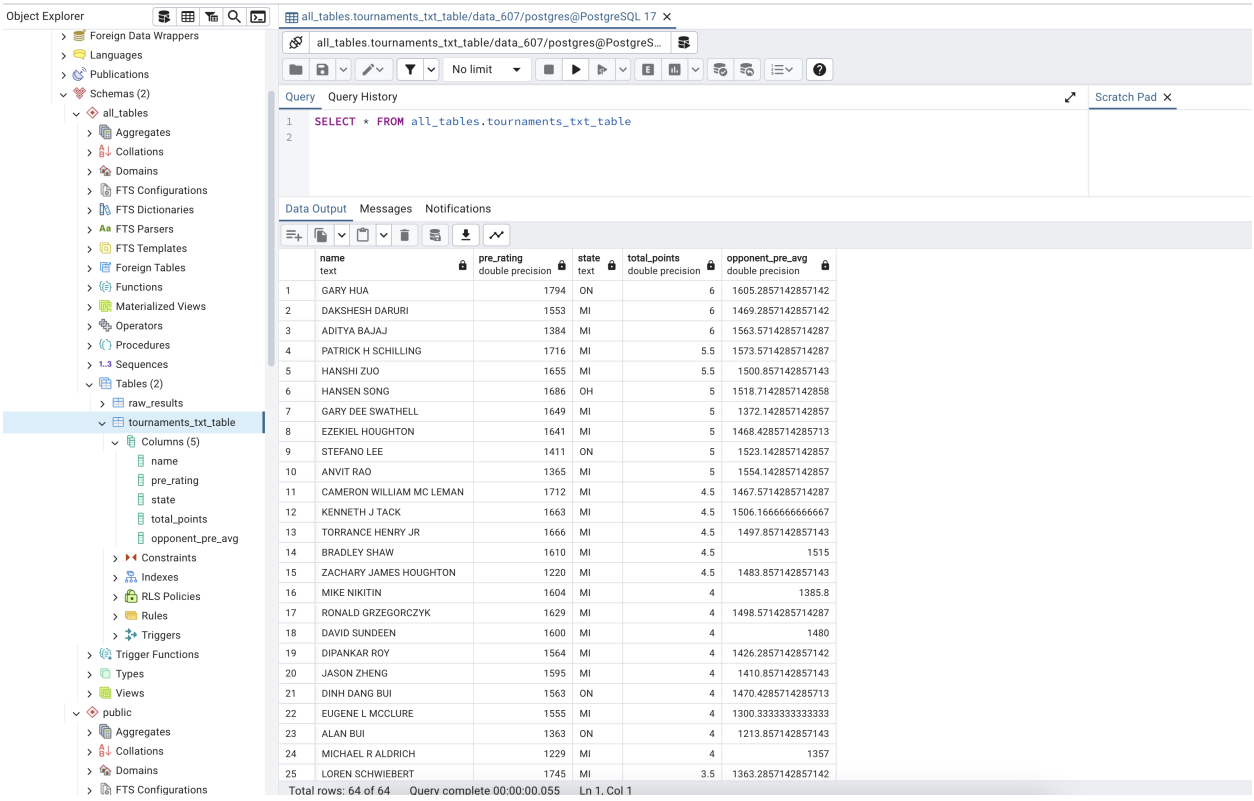

Figure 1: Picture from my Postgres