607 – Project 2

Pricilla

2025-09-29

Data transformation is an important step in any data analysis process that involves the conversion, cleaning, and organizing of data into accessible formats. It ensures that the information is accessible, consistent, secure, and finally recognized by the intended business users. This process is undertaken by organizations to utilize their data to generate timely business insights and support decision-making processes.

(1) Choose any three of the "wide" datasets

Create a .CSV file (or optionally, a MySQL database!) that includes all of the information included in the dataset. You're encouraged to use a "wide" structure similar to how the information appears in the discussion item, so that you can practice tidying and transformations as described below

1. Tournament Data

The tournament dataset contained alternating lines of player information and game details. It was in a wide format where each player's rounds were listed across a single row. To analyze performance, we tidied the data to extract each player's points, pre-rating, opponents, and average opponent rating. This allows us to calculate expected scores using the Elo rating formula and identify overperformers and underperformers.

Load Tournament.txt from Github

https://raw.githubusercontent.com/prnakyazze94/Data_607/refs/heads/main/Class%20Tournament.txt
Data Cleanup

Removed headers and separator lines.

Split lines into player lines and info lines.

Extracted player names, states, pre-ratings, points, and opponent indices.

Calculated each player's average opponent rating and rounds played.

Computed expected scores using the Elo formula. Calculated performance difference: Diff = Points - Expected.

```
# Read the file
lines <- readLines("https://raw.githubusercontent.com/prnakyazze94/Data_607/refs/heads/main/Class%20Tou
head(lines)
```

```
## [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    |"
```

Filter only player lines information.

```
# Remove separator lines and blank lines
lines <- lines[!grepl("^-+|^\\s*$", lines)]

# Remove the two header lines
lines <- lines[-c(1, 2)]

# Separate into player + info lines
player_lines <- lines[seq(1, length(lines), by = 2)]
info_lines <- lines[seq(2, length(lines), by = 2)]
```

create df results

This is a raw extraction table.

It holds directly parsed player info from the tournament text file.

```
results <- data.frame(
  PairNum = integer(),
  Name = character(),
  State = character(),
  USCF_ID = character(),
  PreRating = numeric(),
  PostRating = numeric(),
  TotalScore = numeric(),
  stringsAsFactors = FALSE
)
```

Read the information from your .CSV file into R, and use tidyr and dplyr as needed to tidy and transform your data.

Extract all Player information from text,Prepare output data frame and save it in tournament_results.csv. Output us the processed/analysis table.

It summarizes each player's performance and adds derived fields AvgOpponentRating.

```
# Extract all pre-ratings once (used for AvgOpponentRating)
pre_ratings <- sapply(info_lines, function(info) {
  rating_match <- regmatches(info, regexpr("R:\\s*\\d+", info))
  as.integer(gsub("R:\\s*", "", rating_match))
})

# Prepare output data frame
output <- data.frame(
  Player = character(),
  State = character(),
  Points = numeric(),
  PreRating = integer(),
  AvgOpponentRating = numeric(),
  PlayerNum = integer(),
  OpponentNums = character(),
  OpponentPreRatings = character(),
  stringsAsFactors = FALSE
)
```

```r
# Loop through players
for (i in seq_along(player_lines)) {
  pl <- player_lines[i]
  info <- info_lines[i]

  # Extract clean player name (remove leading number and pipe)
  name <- trimws(gsub("^\\d+\\s*\\|\\s*", "", substr(pl, 5, 36)))

  # Extract points
  points_match <- regmatches(pl, regexpr("\\|\\s*[0-9]+\\.?[0-9]*\\s*\\|", pl))
  points <- as.numeric(gsub("[^0-9.]", "", points_match))

  # Extract state (e.g., ON, MI)
  state_match <- regmatches(info, regexpr("\\b[A-Z]{2}\\b", info))
  state <- if (length(state_match) > 0) state_match else NA

  # Extract pre-rating
  rating_match <- regmatches(info, regexpr("R:\\s*\\d+", info))
  pre_rating <- as.integer(gsub("R:\\s*", "", rating_match))

  # Extract opponent indices from line 1 (rounds)
  rounds <- unlist(strsplit(pl, "\\|"))
  rounds <- rounds[4:length(rounds)]  # skip first 3 parts
  opp_indices <- as.integer(gsub("[^0-9]", "", rounds))
  opp_indices <- opp_indices[!is.na(opp_indices)]

  # Calculate average opponent rating
  if (length(opp_indices) > 0) {
    opp_ratings <- pre_ratings[opp_indices]
    avg_opp_rating <- round(mean(opp_ratings, na.rm = TRUE), 0)
  } else {
    avg_opp_rating <- NA
  }

  # Add row to output
  output <- rbind(output, data.frame(
    Player = name,
    State = state,
    Points = points,
    PreRating = pre_rating,
    AvgOpponentRating = avg_opp_rating,
    stringsAsFactors = FALSE
  ))
}

# View output
# View and save
kable(
  output %>% slice(1:20),
  caption = "Tournament_results"
)
```

Table 1: Tournament_results

Player	State	Points	PreRating	AvgOpponentRating
GARY HUA	ON	6.0	1794	1605
DAKSHESH DARURI	MI	6.0	1553	1469
ADITYA BAJAJ	MI	6.0	1384	1564
PATRICK H SCHILLING	MI	5.5	1716	1574
HANSHI ZUO	MI	5.5	1655	1501
HANSEN SONG	OH	5.0	1686	1519
GARY DEE SWATHELL	MI	5.0	1649	1372
EZEKIEL HOUGHTON	MI	5.0	1641	1468
STEFANO LEE	ON	5.0	1411	1523
ANVIT RAO	MI	5.0	1365	1554
CAMERON WILLIAM MC LEMAN	MI	4.5	1712	1468
KENNETH J TACK	MI	4.5	1663	1506
TORRANCE HENRY JR	MI	4.5	1666	1498
BRADLEY SHAW	MI	4.5	1610	1515
ZACHARY JAMES HOUGHTON	MI	4.5	1220	1484
MIKE NIKITIN	MI	4.0	1604	1386
RONALD GRZEGORCZYK	MI	4.0	1629	1499
DAVID SUNDEEN	MI	4.0	1600	1480
DIPANKAR ROY	MI	4.0	1564	1426
JASON ZHENG	MI	4.0	1595	1411

```r
# save to CSV
write.csv(output, "tournament_results.csv", row.names = FALSE)
```

Tournament Results with Opponents and Rounds Played

```r
# Prepare output data frame
output <- data.frame(
  Player = character(),
  State = character(),
  Points = numeric(),
  PreRating = integer(),
  AvgOpponentRating = numeric(),
  RoundsPlayed = integer(),
  OpponentNums = character(),
  stringsAsFactors = FALSE
)

# Loop through players
for (i in seq_along(player_lines)) {
  pl <- player_lines[i]
  info <- info_lines[i]

  # Extract clean player name
  name <- trimws(gsub("^\\d+\\s*\\|\\s*", "", substr(pl, 5, 36)))

  # Extract points
  points_match <- regmatches(pl, regexpr("\\|\\s*[0-9]+\\.?[0-9]*\\s*\\|", pl))
  points <- as.numeric(gsub("[^0-9.]", "", points_match))
```

```r
  # Extract state
  state_match <- regmatches(info, regexpr("\\b[A-Z]{2}\\b", info))
  state <- if (length(state_match) > 0) state_match else NA

  # Extract pre-rating
  rating_match <- regmatches(info, regexpr("R:\\s*\\d+", info))
  pre_rating <- as.integer(gsub("R:\\s*", "", rating_match))

  # Extract opponent indices from round results
  rounds <- unlist(strsplit(pl, "\\|"))
  rounds <- rounds[4:length(rounds)]  # skip first 3 parts
  opp_indices <- as.integer(gsub("[^0-9]", "", rounds))
  opp_indices <- opp_indices[!is.na(opp_indices)]

  # Calculate average opponent rating
  if (length(opp_indices) > 0) {
    opp_ratings <- pre_ratings[opp_indices]
    avg_opp_rating <- round(mean(opp_ratings, na.rm = TRUE), 0)
  } else {
    avg_opp_rating <- NA
  }

  # Add row to output (collapse opponent indices into a string)
  output <- rbind(output, data.frame(
    Player = name,
    State = state,
    Points = points,
    PreRating = pre_rating,
    AvgOpponentRating = avg_opp_rating,
    RoundsPlayed = length(opp_indices),
    OpponentNums = paste(opp_indices, collapse = ","),
    stringsAsFactors = FALSE
  ))
}
# View and save
kable(
  output %>% slice(1:20),
  caption = "Tournament Results with Opponents and Rounds Played"
)
```

Table 2: Tournament Results with Opponents and Rounds Played

Player	State	Points	PreRating	AvgOpponentRating	RoundsPlayed	OpponentNums
GARY HUA	ON	6.0	1794	1605	7	39,21,18,14,7,12,4
DAKSHESH DARURI	MI	6.0	1553	1469	7	63,58,4,17,16,20,7
ADITYA BAJAJ	MI	6.0	1384	1564	7	8,61,25,21,11,13,12
PATRICK H SCHILLING	MI	5.5	1716	1574	7	23,28,2,26,5,19,1
HANSHI ZUO	MI	5.5	1655	1501	7	45,37,12,13,4,14,17
HANSEN SONG	OH	5.0	1686	1519	7	34,29,11,35,10,27,21
GARY DEE SWATHELL	MI	5.0	1649	1372	7	57,46,13,11,1,9,2

Player	State	Points	PreRating	AvgOpponentRating	RoundsPlayed	OpponentNums
EZEKIEL HOUGHTON	MI	5.0	1641	1468	7	3,32,14,9,47,28,19
STEFANO LEE	ON	5.0	1411	1523	7	25,18,59,8,26,7,20
ANVIT RAO	MI	5.0	1365	1554	7	16,19,55,31,6,25,18
CAMERON WILLIAM MC LEMAN	MI	4.5	1712	1468	7	38,56,6,7,3,34,26
KENNETH J TACK	MI	4.5	1663	1506	6	42,33,5,38,1,3
TORRANCE HENRY JR	MI	4.5	1666	1498	7	36,27,7,5,33,3,32
BRADLEY SHAW	MI	4.5	1610	1515	7	54,44,8,1,27,5,31
ZACHARY JAMES HOUGHTON	MI	4.5	1220	1484	7	19,16,30,22,54,33,38
MIKE NIKITIN	MI	4.0	1604	1386	5	10,15,39,2,36
RONALD GRZEGORCZYK	MI	4.0	1629	1499	7	48,41,26,2,23,22,5
DAVID SUNDEEN	MI	4.0	1600	1480	7	47,9,1,32,19,38,10
DIPANKAR ROY	MI	4.0	1564	1426	7	15,10,52,28,18,4,8
JASON ZHENG	MI	4.0	1595	1411	7	40,49,23,41,28,2,9

This tidy structure allows for comparison between actual and expected performance, identification of top overperformers and underperformers, and fair performance evaluation based on opponent strength.

Elo calculation

```r
# Elo expected score function
elo_expect <- function(r_player, r_opp) {
  1 / (1 + 10 ^ ((r_opp - r_player) / 400))
}

# --- Compute rounds played robustly ---
if (exists("player_lines") && length(player_lines) == nrow(output)) {
  # Extract opponent indices from the stored player_lines (same logic as earlier)
  rounds_played <- sapply(player_lines, function(pl) {
    parts <- unlist(strsplit(pl, "\\|"))
    if (length(parts) < 4) return(0L)
    opp_parts <- parts[4:length(parts)]
    opp_indices <- as.integer(gsub("[^0-9]", "", opp_parts))
    sum(!is.na(opp_indices))
  })
} else if ("OpponentNums" %in% names(output)) {
  # If you saved OpponentNums as "2,5,8" or "2 5 8"
  rounds_played <- sapply(output$OpponentNums, function(x) {
    if (is.na(x) || x == "") return(0L)
    length(unlist(strsplit(as.character(x), "[,\\s]+")))
  })
} else {
  rounds_played <- rep(NA_integer_, nrow(output))
  warning("Could not infer RoundsPlayed from player_lines or OpponentNums. RoundsPlayed set to NA.")
}

# --- Add columns using dplyr::mutate ---
```

```r
library(dplyr)

output <- output %>%
  mutate(
    RoundsPlayed = rounds_played,
    # compute Expected only when we have the necessary values
    Expected = ifelse(
      !is.na(PreRating) & !is.na(AvgOpponentRating) & !is.na(RoundsPlayed),
      elo_expect(PreRating, AvgOpponentRating) * RoundsPlayed,
      NA_real_
    ),
    Diff = Points - Expected
  )

# View and save
kable(
  output %>% slice(1:20),
  caption = "Tournament_results_with_expected"
)
```

Table 3: Tournament_results_with_expected

Player	State	Points	PreRating	AvgOpponentRating	RoundsPlayed	OpponentNums	Expected	Diff
GARY HUA	ON	6.0	1794	1605	7	39,21,18,14,7,12,4	5.235997	0.7640032
DAKSHESH DARURI	MI	6.0	1553	1469	7	63,58,4,17,16,20,7	4.330089	1.6699111
ADITYA BAJAJ	MI	6.0	1384	1564	7	8,61,25,21,11,13,12	1.833237	4.1667632
PATRICK H SCHILLING	MI	5.5	1716	1574	7	23,28,2,26,5,19,1	4.855815	0.6441846
HANSHI ZUO	MI	5.5	1655	1501	7	45,37,12,13,4,14,17	4.957165	0.5428353
HANSEN SONG	OH	5.0	1686	1519	7	34,29,11,35,10,27,21	5.063715	-0.0637151
GARY DEE SWATHELL	MI	5.0	1649	1372	7	57,46,13,11,1,9,2	5.818777	-0.8187774
EZEKIEL HOUGHTON	MI	5.0	1641	1468	7	3,32,14,9,47,28,19	5.111718	-0.1117179
STEFANO LEE	ON	5.0	1411	1523	7	25,18,59,8,26,7,20	2.409257	2.5907435
ANVIT RAO	MI	5.0	1365	1554	7	16,19,55,31,6,25,18	1.764003	3.2359968
CAMERON WILLIAM MC LEMAN	MI	4.5	1712	1468	7	38,56,6,7,3,34,26	5.620364	-1.1203642
KENNETH J TACK	MI	4.5	1663	1506	6	42,33,5,38,1,3	4.270335	0.2296649
TORRANCE HENRY JR	MI	4.5	1666	1498	7	36,27,7,5,33,3,32	5.071768	-0.5717677
BRADLEY SHAW	MI	4.5	1610	1515	7	54,44,8,1,27,5,31	4.433854	0.0661461
ZACHARY JAMES HOUGHTON	MI	4.5	1220	1484	7	19,16,30,22,54,33,38	1.256533	3.2434665
MIKE NIKITIN	MI	4.0	1604	1386	5	10,15,39,2,36	3.890742	0.1092577

Player	State	Points	PreRating	AvgOpponentRating	RoundsPlayed	OpponentNums	Expected	Diff
RONALD GRZEGOR-CZYK	MI	4.0	1629	1499	7	48,41,26,2,23,22,5	4.751718	-0.7517184
DAVID SUNDEEN	MI	4.0	1600	1480	7	47,9,1,32,19,38,10	4.662976	-0.6629760
DIPANKAR ROY	MI	4.0	1564	1426	7	15,10,52,28,18,4,8	4.821415	-0.8214150
JASON ZHENG	MI	4.0	1595	1411	7	40,49,23,41,28,2,9	5.197749	-1.1977489

```r
write.csv(output, "tournament_results_with_expected.csv", row.names = FALSE)
```

Top overperformers

```r
kable(
  output %>%
    arrange(desc(Diff)) %>%
    slice(1:5) %>%
    select(Player, State, Points, PreRating, AvgOpponentRating, RoundsPlayed, Expected),
  caption = "Top 5 Overerperformers"
)
```

Table 4: Top 5 Overerperformers

Player	State	Points	PreRating	AvgOpponentRating	RoundsPlayed	Expected
ADITYA BAJAJ	MI	6.0	1384	1564	7	1.8332368
ZACHARY JAMES HOUGHTON	MI	4.5	1220	1484	7	1.2565335
ANVIT RAO	MI	5.0	1365	1554	7	1.7640032
AMIYATOSH PWNANANDAM	MI	3.5	980	1385	5	0.4427911
JACOB ALEXANDER LAVALLEY	MI	3.0	377	1358	7	0.0246076

Top 5 underperformers

```r
kable(
  output %>%
    arrange(Diff) %>%
    slice(1:5) %>%
    select(Player, State, Points, PreRating, AvgOpponentRating, RoundsPlayed, Expected),
  caption = "Top 5 Underperformers"
)
```

Table 5: Top 5 Underperformers

Player	State	Points	PreRating	AvgOpponentRating	RoundsPlayed	Expected
LOREN SCHWIEBERT	MI	3.5	1745	1363	7	6.301098
GEORGE AVERY JONES	ON	3.5	1522	1144	7	6.286478
JOSHUA DAVID LEE	MI	3.5	1438	1150	7	5.879655
JARED GE	MI	3.0	1332	1150	7	5.182299
ROBERT GLEN VASEY	MI	3.0	1283	1107	7	5.135436

2. movielens Dataset

The MovieLens dataset originally consisted of two separate wide files, one for user ratings and one for movie details. The ratings file included user IDs, movie IDs, and scores, while the movies file included IDs, titles, and genre indicators. By joining these two tables on the movie_id, I created a CSV that combines user ratings with movie titles, keeping the dataset structured in a wide format. Tidying this data makes it easier to perform tasks such as analyzing user preferences by movie title rather than only by ID.

Data Cleanup involved.

Joined ratings with movie titles.

Calculated average ratings per movie.

Computed baseline predictions for ratings.

Load rating data

```
# Read ratings data
ratings <- read_delim(
  "http://files.grouplens.org/datasets/movielens/ml-100k/u.data",
  delim = "\t",
  col_names = c("user_id", "movie_id", "rating", "timestamp"),
  show_col_types = FALSE
)

# Define movie columns
movie_cols <- c(
  "movie_id", "title", "release_date", "video_release_date", "IMDb_URL",
  "unknown", "Action", "Adventure", "Animation", "Children's", "Comedy", "Crime",
  "Documentary", "Drama", "Fantasy", "Film-Noir", "Horror", "Musical",
  "Mystery", "Romance", "Sci-Fi", "Thriller", "War", "Western"
)

genre_cols <- movie_cols[6:length(movie_cols)]
# Load movie data
# Read movies data
movies <- read_delim(
  "http://files.grouplens.org/datasets/movielens/ml-100k/u.item",
  delim = "|",
  col_names = movie_cols,
  locale = locale(encoding = "latin1")
```

```r
) %>%
  select(movie_id, title, all_of(genre_cols))
```

```
## Rows: 1682 Columns: 24
## -- Column specification --------------------------------------------------------
## Delimiter: "|"
## chr  (3): title, release_date, IMDb_URL
## dbl (20): movie_id, unknown, Action, Adventure, Animation, Children's, Comed...
## lgl  (1): video_release_date
##
## 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
# View the first few rows
#head(movies)
head(ratings)
```

```
## # A tibble: 6 x 4
##   user_id movie_id rating timestamp
##     <dbl>    <dbl>  <dbl>     <dbl>
## 1     196      242      3 881250949
## 2     186      302      3 891717742
## 3      22      377      1 878887116
## 4     244       51      2 880606923
## 5     166      346      1 886397596
## 6     298      474      4 884182806
```

```r
head(movies)
```

```
## # A tibble: 6 x 21
##   movie_id title    unknown Action Adventure Animation `Children's` Comedy Crime
##      <dbl> <chr>      <dbl>  <dbl>     <dbl>     <dbl>        <dbl>  <dbl> <dbl>
## 1        1 Toy Sto~       0      0         0         1            1      1     0
## 2        2 GoldenE~       0      1         1         0            0      0     0
## 3        3 Four Ro~       0      0         0         0            0      0     0
## 4        4 Get Sho~       0      1         0         0            0      1     0
## 5        5 Copycat~       0      0         0         0            0      0     1
## 6        6 Shangha~       0      0         0         0            0      0     0
## # i 12 more variables: Documentary <dbl>, Drama <dbl>, Fantasy <dbl>,
## #   `Film-Noir` <dbl>, Horror <dbl>, Musical <dbl>, Mystery <dbl>,
## #   Romance <dbl>, `Sci-Fi` <dbl>, Thriller <dbl>, War <dbl>, Western <dbl>
```

Read the information file into R, and use tidyr and dplyr as needed to tidy and transform your data.

use library(dplyr) to join rating and movie to create a csv with title, user_id, movie_id, rating.

```r
# Join ratings with movie titles
ratings_with_titles <- ratings %>%
  inner_join(movies %>% select(movie_id, title), by = "movie_id") %>%
  select(title, user_id, movie_id, rating )

# Preview
```

```r
##head(ratings_with_titles)
# Show as a nice table
kable(head(ratings_with_titles, 10), caption = "Sample of Ratings with Movie Titles")
```

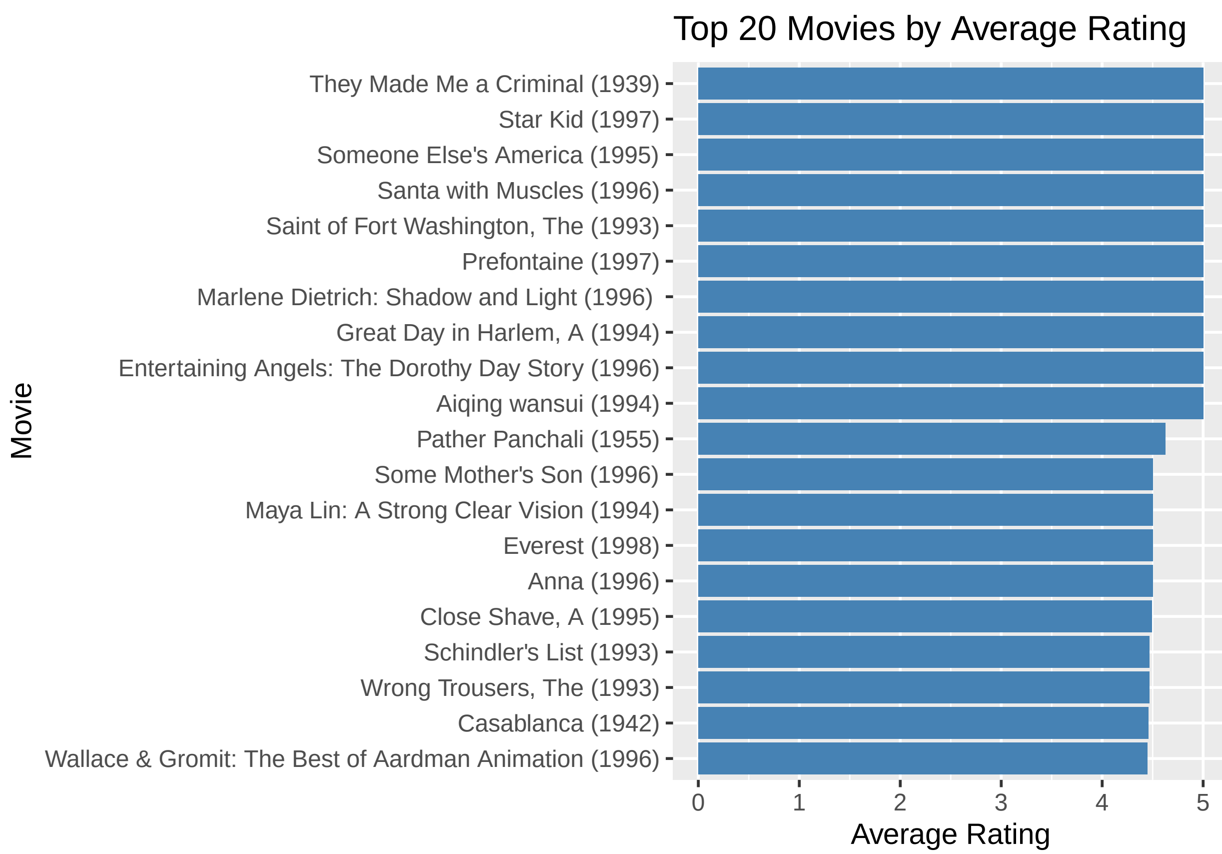
Table 6: Sample of Ratings with Movie Titles

title	user_id	movie_id	rating
Kolya (1996)	196	242	3
L.A. Confidential (1997)	186	302	3
Heavyweights (1994)	22	377	1
Legends of the Fall (1994)	244	51	2
Jackie Brown (1997)	166	346	1
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1963)	298	474	4
Hunt for Red October, The (1990)	115	265	2
Jungle Book, The (1994)	253	465	5
Grease (1978)	305	451	3
Remains of the Day, The (1993)	6	86	3

```r
# Save to CSV
write_csv(ratings_with_titles, "ratings_with_titles.csv")
```

Plot Top 20 Movies by Average Rating

```r
ratings_with_titles %>%
  group_by(title) %>%
  summarise(avg_rating = mean(rating), n = n(), .groups = "drop") %>%
  arrange(desc(avg_rating)) %>%
  slice(1:20) %>%   # top 20 movies
  ggplot(aes(x = reorder(title, avg_rating), y = avg_rating)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(title = "Top 20 Movies by Average Rating", x = "Movie", y = "Average Rating")
```



Predicted rating calculated using the baseline method.

Predicted Ratings Using the Baseline Method

The predicted rating for a movie is calculated using the baseline formula:

$$\hat{r}_i = \mu + b_i$$

Where:

- μ = global mean rating
- b_i = item bias (the average deviation of a movie's ratings from the global mean)

We then identified the 20 lowest-performing movies and computed their predicted ratings using this formula.

```
# Global mean
mu <- mean(ratings_with_titles$rating)

# Compute user bias
user_bias <- ratings_with_titles %>%
  group_by(user_id) %>%
  summarise(b_u = mean(rating - mu), .groups = "drop")

# Compute item bias
item_bias <- ratings_with_titles %>%
```

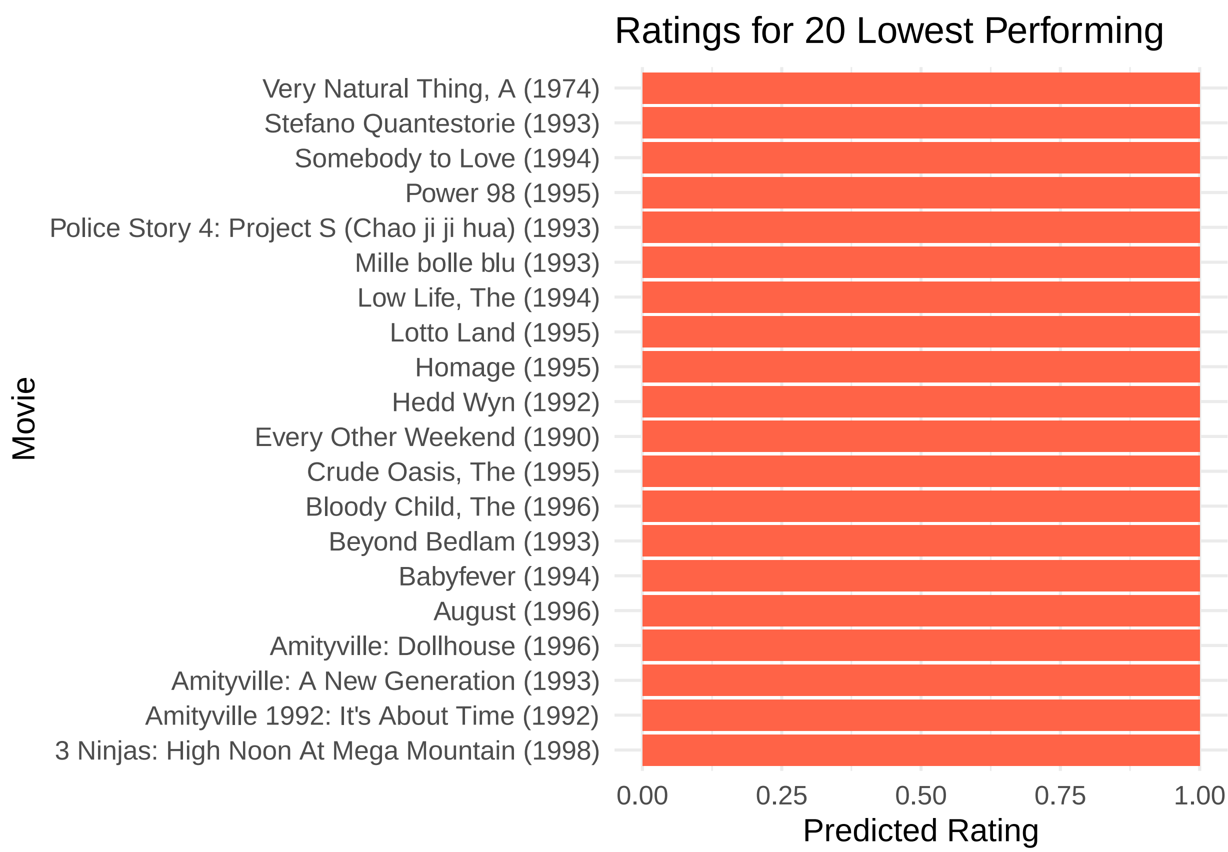
```r
  group_by(movie_id, title) %>%
  summarise(b_i = mean(rating - mu), .groups = "drop")

# Join biases back into full dataset
baseline_preds <- ratings_with_titles %>%
  left_join(user_bias, by = "user_id") %>%
  left_join(item_bias, by = c("movie_id", "title")) %>%
  mutate(pred_rating = mu + b_u + b_i)

# Find 20 lowest performing movies (by actual avg rating)
# n is the total number of people that rated the movie
lowest20 <- ratings_with_titles %>%
  group_by(movie_id, title) %>%
  summarise(avg_rating = mean(rating), n = n(), .groups = "drop") %>%
  arrange(avg_rating) %>%
  slice(1:20)

# Predicted ratings for those 20 movies
predicted_lowest20 <- lowest20 %>%
  left_join(item_bias, by = c("movie_id", "title")) %>%
  mutate(pred_rating = mu + b_i)   # aggregate baseline (ignoring user-specific bias)

# Show results
kable(predicted_lowest20, caption = "Predicted Ratings for 20 Lowest Performing Movies (Baseline)")
```

Table 7: Predicted Ratings for 20 Lowest Performing Movies (Baseline)

movie_id	title	avg_rating	n	b_i	pred_rating
314	3 Ninjas: High Noon At Mega Mountain (1998)	1	5	-2.52986	1
437	Amityville 1992: It's About Time (1992)	1	5	-2.52986	1
439	Amityville: A New Generation (1993)	1	5	-2.52986	1
599	Police Story 4: Project S (Chao ji ji hua) (1993)	1	1	-2.52986	1
784	Beyond Bedlam (1993)	1	2	-2.52986	1
830	Power 98 (1995)	1	1	-2.52986	1
852	Bloody Child, The (1996)	1	1	-2.52986	1
858	Amityville: Dollhouse (1996)	1	3	-2.52986	1
1308	Babyfever (1994)	1	2	-2.52986	1
1309	Very Natural Thing, A (1974)	1	1	-2.52986	1
1320	Homage (1995)	1	1	-2.52986	1
1325	August (1996)	1	1	-2.52986	1
1329	Low Life, The (1994)	1	1	-2.52986	1
1334	Somebody to Love (1994)	1	2	-2.52986	1
1339	Stefano Quantestorie (1993)	1	1	-2.52986	1
1340	Crude Oasis, The (1995)	1	1	-2.52986	1
1341	Hedd Wyn (1992)	1	1	-2.52986	1
1343	Lotto Land (1995)	1	1	-2.52986	1
1348	Every Other Weekend (1990)	1	1	-2.52986	1
1349	Mille bolle blu (1993)	1	1	-2.52986	1

Plot Predicted Ratings for 20 Lowest Performing Movies.

```r
# Plot Predicted Ratings for 20 Lowest Performing Movies
ggplot(predicted_lowest20, aes(x = reorder(title, pred_rating), y = pred_rating)) +
  geom_col(fill = "tomato") +
  coord_flip() +  # horizontal bars
  labs(
    title = "Ratings for 20 Lowest Performing",
    x = "Movie",
    y = "Predicted Rating"
  ) +
  theme_minimal(base_size = 12)
```



3. Test score data

The test score dataset was a wide structure, with each subject (Math, Science, History) represented as its own column. While this format is easy to read for humans, it is not optimal for analysis in R. Using pivot_longer(), I transformed the dataset into tidy form, where each row represents a student, subject, score combination. This tidy structure allows for straightforward filtering, grouping, and visualization of student performance across different subjects.

https://raw.githubusercontent.com/prnakyazze94/Data_607/refs/heads/main/Test_score

Load data into R

```r
# Read the file
Score <- read_csv("https://raw.githubusercontent.com/prnakyazze94/Data_607/refs/heads/main/Test_score",
```

```r
                       show_col_types = FALSE)
print(Score)
```

```
## # A tibble: 3 x 4
##   Name     Math Science History
##   <chr>   <dbl>   <dbl>   <dbl>
## 1 Alice      90      85      88
## 2 Bob        78      82      80
## 3 Charlie    85      89      92
```

Read the information from your .CSV file into R, and use tidyr and dplyr as needed to tidy and transform your data.

Score is a wide dataset (each subject in its own column). To make it tidy (long format, one row per student, subject, score), I use pivot_longer() from tidyr.

```r
# Assuming Score already looks like your example
Score_tidy <- Score %>%
  pivot_longer(
    cols = c(Math, Science, History),   # the subject columns
    names_to = "Subject",               # new column for subject name
    values_to = "Score"                 # new column for score values
  )

# View tidy data
#print(Score_tidy)
kable(Score_tidy)
```

Name	Subject	Score
Alice	Math	90
Alice	Science	85
Alice	History	88
Bob	Math	78
Bob	Science	82
Bob	History	80
Charlie	Math	85
Charlie	Science	89
Charlie	History	92

```r
# Optionally save to CSV
write_csv(Score_tidy, "Score_tidy.csv")
```

Average score per subject

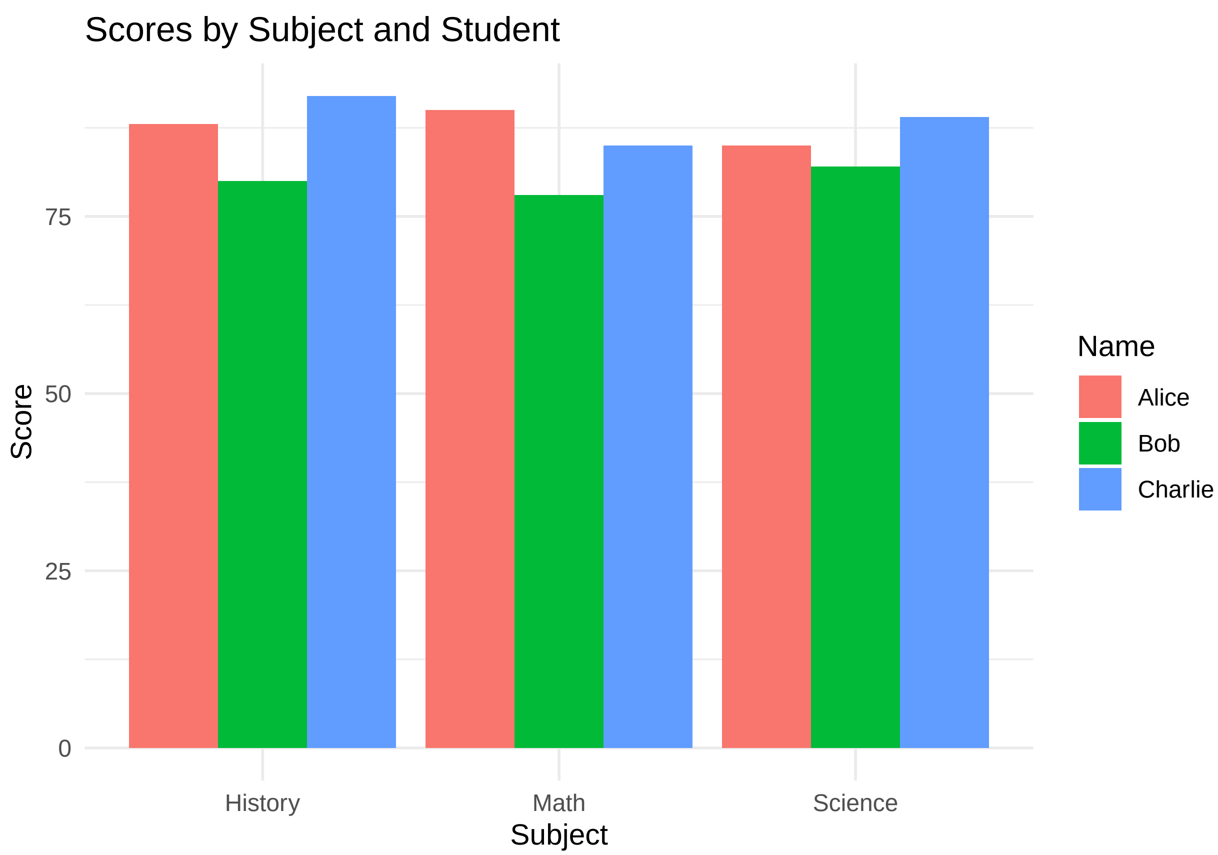
```r
Score %>%
  pivot_longer(cols = c(Math, Science, History), names_to = "Subject", values_to = "Score") %>%
  group_by(Subject) %>%
  summarise(Average = mean(Score, na.rm = TRUE), .groups = "drop")
```

```
## # A tibble: 3 x 2
```

```
##   Subject Average
##   <chr>     <dbl>
## 1 History    86.7
## 2 Math       84.3
## 3 Science    85.3
```

Plot Scores by Subject and Student

```r
ggplot(Score_tidy, aes(x = Subject, y = Score, fill = Name)) +
  geom_col(position = "dodge") +
  labs(title = "Scores by Subject and Student") +
  theme_minimal()
```



Plot Student Performance Across Subjects

```r
ggplot(Score_tidy, aes(x = Subject, y = Score, group = Name, color = Name)) +
  geom_line(size = 1) +
  geom_point(size = 3) +
  labs(title = "Student Performance Across Subjects") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

CONCLUCIOM For the tournament data, cleaning and parsing allowed calculation of derived metrics like average opponent rating and expected Elo scores, which highlighted top performers and underperformers. For the MovieLens dataset, joining ratings with movie metadata and applying a baseline prediction model allowed identification of both high and low performing movies and estimation of predicted ratings. Finally, for the test scores dataset, converting the wide format into tidy long format enabled calculation of subject wise averages and visualization of individual student performance trends.

These illustrated the importance of tidying data, calculating summary statistics, and applying basic predictive models to extract insights. The cleaned datasets are saved as CSV files, providing reproducible, analysis-ready resources for future work.

Resource https://www.geeksforgeeks.org/data-analysis/what-is-data-transformation/