Assignment 5b

2025-09-27

```
txt <- readLines("tournamentinfo.txt")
```

```
## Warning in readLines("tournamentinfo.txt"): incomplete final line found on
## 'tournamentinfo.txt'
```

```
raw <- paste(txt, collapse = "\n")
blocks <- str_split(raw, "-{5,}\\s*\\n", simplify = TRUE) |>
  as.vector() |>
  keep(~ str_squish(.x) != "")

# Parser for a single two-line player block
parse_player <- function(block) {
  name_line <- str_match(block,
    "(?m)^\\s*(\\d+)\\s*\\|\\s*([A-Z' .-]+?)\\s*\\|\\s*([0-9.]+)\\s*\\|(.*)\\|\\s*$"
  )
  state_line <- str_match(block,
    "(?m)^\\s*([A-Z]{2})\\s*\\|\\s*\\d+\\s*/\\s*R:\\s*(\\d+)[A-Z]*\\d*\\s*->\\s*\\d+"
  )

  if (any(is.na(name_line)) || any(is.na(state_line))) return(NULL)

  pairNum   <- as.integer(name_line[2])
  name      <- str_squish(name_line[3])
  totalPts  <- as.numeric(name_line[4])
  roundsStr <- name_line[5]
  state     <- state_line[2]
  preRating <- as.integer(state_line[3])

  # Extract opponent pair numbers from W/L/D results
  opps <- str_match_all(roundsStr, "[WLD]\\s*(\\d+)")[[1]]
  opps <- if (length(opps)) as.integer(opps[,2]) else integer()

  tibble(pairNum, name, state, totalPts, preRating, opps = list(opps))
}

players <- map(blocks, parse_player) |> compact()

players <- bind_rows(players)
stopifnot(is_tibble(players), nrow(players) > 0)

ratingByPair <- setNames(players$preRating, players$pairNum)

avgOpp <- map_dbl(players$opps, function(os) {
  if (length(os) == 0) return(NA_real_)
```

```
  round(mean(ratingByPair[as.character(os)]))
})

out <- players |>
  mutate(
    `Player Name` = str_to_title(name),
    `Player State` = state,
    `Total Points` = totalPts,
    `Pre Rating` = preRating,
    `Average Opponent Pre Rating` = avgOpp
  ) |>
  select(`Player Name`, `Player State`, `Total Points`, `Pre Rating`, `Average Opponent Pre Rating`) |>
  arrange(desc(`Total Points`), `Player Name`)

write_csv(out, "tournament_results.csv")
out %>% head(10)
```

```
## # A tibble: 10 x 5
##    `Player Name`       `Player State` `Total Points` `Pre Rating`
##    <chr>               <chr>                  <dbl>        <int>
##  1 Aditya Bajaj        MI                       6           1384
##  2 Dakshesh Daruri     MI                       6           1553
##  3 Gary Hua            ON                       6           1794
##  4 Hanshi Zuo          MI                       5.5         1655
##  5 Patrick H Schilling MI                       5.5         1716
##  6 Anvit Rao           MI                       5           1365
##  7 Ezekiel Houghton    MI                       5           1641
##  8 Gary Dee Swathell   MI                       5           1649
##  9 Hansen Song         OH                       5           1686
## 10 Stefano Lee         ON                       5           1411
## # i 1 more variable: `Average Opponent Pre Rating` <dbl>
```

```
# --- Elo helpers ---
elo_expected <- function(rPlayer, rOpp) {
  1 / (1 + 10^((rOpp - rPlayer) / 400))
}

# Re-parse each block to capture BOTH the opponent numbers and the W/D/L result letters.
parse_player_rounds <- function(block) {
  name_line <- str_match(block,
    "(?m)^\\s*(\\d+)\\s*\\|\\s*([A-Z' .-]+?)\\s*\\|\\s*([0-9.]+)\\s*\\|(.*)\\|\\s*$"
  )
  state_line <- str_match(block,
    "(?m)^\\s*([A-Z]{2})\\s*\\|\\s*\\d+\\s*/\\s*R:\\s*(\\d+)[A-Z]*\\d*\\s*->\\s*\\d+"
  )
  if (any(is.na(name_line)) || any(is.na(state_line))) return(NULL)

  pairNum   <- as.integer(name_line[2])
  name      <- str_squish(name_line[3])
  totalPts  <- as.numeric(name_line[4])
  roundsStr <- name_line[5]
  state     <- state_line[2]
  preRating <- as.integer(state_line[3])
```

```r
  # Capture tokens like "W 12", "D 7", "L 3"; ignore H/U/B byes
  toks <- str_match_all(roundsStr, "([WLD])\\s*(\\d+)")[[1]]
  if (nrow(toks) == 0) {
    opps <- integer()
    res  <- character()
  } else {
    res  <- toks[,2]
    opps <- as.integer(toks[,3])
  }

  tibble(
    pairNum, name = str_to_title(name), state, preRating, totalPts,
    opps = list(opps),
    results = list(res)
  )
}

# Build a table with per-player expected vs actual-from-opponents
players_rounds <- map(blocks, parse_player_rounds) |> compact() |> bind_rows()

# Map of pair -> preRating already computed earlier:
# ratingByPair <- setNames(players$preRating, players$pairNum)

calc_expected_and_actual <- function(preRating, opps, results) {
  # expected from Elo vs each actual opponent
  if (length(opps) == 0) {
    expected <- 0
  } else {
    oppRatings <- ratingByPair[as.character(opps)]
    expected <- sum(elo_expected(preRating, oppRatings))
  }
  # actual points from played rounds only (W=1, D=0.5, L=0)
  if (length(results) == 0) {
    actualFromOpps <- 0
  } else {
    pts <- case_when(
      results == "W" ~ 1,
      results == "D" ~ 0.5,
      TRUE ~ 0
    )
    actualFromOpps <- sum(pts)
  }
  tibble(expectedScore = expected, actualFromOpps = actualFromOpps, delta = actualFromOpps - expected)
}

elo_summary <- players_rounds |>
  mutate(tmp = pmap(list(preRating, opps, results), calc_expected_and_actual)) |>
  unnest(tmp) |>
  select(
    `Player Name` = name,
    `Player State` = state,
    `Pre Rating` = preRating,
    `Actual Points (played rounds)` = actualFromOpps,
```

```
    `Expected Points` = expectedScore,
    `Delta (Actual - Expected)` = delta
  ) |>
  arrange(desc(`Delta (Actual - Expected)`))

# Top 5 overperformers
top_over <- elo_summary |> slice_max(`Delta (Actual - Expected)`, n = 5, with_ties = FALSE)

# Top 5 underperformers
top_under <- elo_summary |> slice_min(`Delta (Actual - Expected)`, n = 5, with_ties = FALSE)

top_over
```

```
## # A tibble: 5 x 6
##   `Player Name`            `Player State` `Pre Rating` Actual Points (played r~1
##   <chr>                    <chr>                 <int>                      <dbl>
## 1 Aditya Bajaj             MI                     1384                        6
## 2 Zachary James Houghton   MI                     1220                        4.5
## 3 Anvit Rao                MI                     1365                        5
## 4 Jacob Alexander Lavalley MI                      377                        3
## 5 Stefano Lee              ON                     1411                        5
## # i abbreviated name: 1: `Actual Points (played rounds)`
## # i 2 more variables: `Expected Points` <dbl>,
## #   `Delta (Actual - Expected)` <dbl>
```

```
top_under
```

```
## # A tibble: 5 x 6
##   `Player Name`      `Player State` `Pre Rating` `Actual Points (played rounds)`
##   <chr>              <chr>                 <int>                           <dbl>
## 1 Loren Schwiebert   MI                     1745                             3.5
## 2 George Avery Jones ON                     1522                             3.5
## 3 Larry Hodge        MI                     1270                             1
## 4 Jared Ge           MI                     1332                             3
## 5 Rishi Shetty       MI                     1494                             3.5
## # i 2 more variables: `Expected Points` <dbl>,
## #   `Delta (Actual - Expected)` <dbl>
```

```
elo_summary |> write_csv("elo_expected_vs_actual.csv")

final_table <- out |>
  left_join(
    elo_summary,
    by = c("Player Name", "Player State", "Pre Rating")
  ) |>
  relocate(`Expected Points`, .after = `Total Points`) |>
  relocate(`Delta (Actual - Expected)`, .after = `Expected Points`)

write_csv(final_table, "tournament_results_with_elo.csv")
final_table |> arrange(desc(`Delta (Actual - Expected)`)) |> head(10)
```

```
## # A tibble: 10 x 8
```

```
##    ‘Player Name‘            ‘Player State‘ ‘Total Points‘ ‘Expected Points‘
##    <chr>                    <chr>                  <dbl>             <dbl>
##  1 Aditya Bajaj             MI                       6              1.95
##  2 Zachary James Houghton   MI                       4.5            1.37
##  3 Anvit Rao                MI                       5              1.94
##  4 Jacob Alexander Lavalley MI                       3              0.0432
##  5 Stefano Lee              ON                       5              2.29
##  6 Dakshesh Daruri          MI                       6              3.78
##  7 Ethan Guo                MI                       2.5            0.295
##  8 Tejas Ayyagari           MI                       2.5            1.03
##  9 Michael R Aldrich        MI                       4              2.55
## 10 Amiyatosh Pwnanandam     MI                       3.5            0.773
## # i 4 more variables: ‘Delta (Actual - Expected)‘ <dbl>, ‘Pre Rating‘ <int>,
## #   ‘Average Opponent Pre Rating‘ <dbl>, ‘Actual Points (played rounds)‘ <dbl>
```