Coding Challenge UPMC Question 1

Adam Whiteside

2025-01-10

```
#load necessary packages
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
##
## 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
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

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

```
#load data
load("C:/Users/ADAMI/Downloads/upmc code/CodeChallenge2024.RData")
#load the IDs
ids <- readLines("IDs.txt", warn= FALSE)

# Convert IDs to numeric
ids <- as.numeric(ids)

# Step 1: Filter `id_map` for relevant participants
id_map_filtered <- id_map[id_map$old_id %in% ids, ]  # Filter rows where old_id matches ids

#take the new ids, what we want
selected_ids <- id_map_filtered$new_id
print(selected_ids)
```

```
##  [1] 2027 2056 2068 2069 2072 2074 2086 2090 2104 2110 2149 2162 2202 2203 2215
## [16] 2218 2227 2263 2267 2274 2292 2294 2297 2306 2320 2329 2333 2341 2343 2347
## [31] 2356 2357 2358 2364 2368 2379 2390 2396 2398 2407 2408 2411 2425 2431 2432
## [46] 2442 2443 2447 2448 2450 2452 2453 2456 2463 2465 2466 2467 2472 2478 2483
## [61] 2485 2502 2505 2507 2513 2517 2519 2521 2526 2527 2528 2535 2539 2549 2555
## [76] 2566 2570 2571 2572 2601 2602 2603 2617 2648 2668 2672 2673 2674 2734 2779
```

```
# Create a named vector for easy lookup
id_mapping <- setNames(id_map_filtered$new_id, id_map_filtered$old_id)

# Step 2: Filter `HAM_protect` using old_ids
HAM_protect_clean <- HAM_protect[HAM_protect$ID %in% names(id_mapping), ]

# Step 3: Replace old_id in `HAM_protect` with new_id from the mapping
HAM_protect_clean$new_id <- id_mapping[as.character(HAM_protect_clean$ID)]

# Step 4: Filter `HAM_sleep` using new_ids
HAM_sleep_clean <- HAM_sleep[HAM_sleep$ID %in% id_map_filtered$new_id, ]
print(head(HAM_sleep_clean))
```

```
## # A tibble: 6 × 26
## # Groups:   ID [2]
##      ID timepoint     bq_date    fug_date   ham_1_dm ham_2_gf ham_3_su ham_3a_wl
##   <dbl> <chr>         <date>     <date>        <dbl>    <dbl>    <dbl>     <dbl>
## 1  2068 baseline_ss_… 2024-01-08 NA                3        3        0         9
## 2  2068 week_1_ss_ar… NA         2024-01-16        3        1        0         9
## 3  2068 week_2_ss_ar… NA         2024-01-23        2        0        0         9
## 4  2068 week_3_ss_ar… NA         2024-01-30        2        0        0         9
## 5  2069 baseline_ss_… 2023-11-22 NA                0        1        0         9
## 6  2069 week_1_ss_ar… NA         2023-11-29        1        1        0         9
## # ℹ 18 more variables: ham_3b_wd <dbl>, ham_3c_rld <dbl>, ham_3d_asa <dbl>,
## #   ham_3e_pdw <dbl>, ham_4_ii <dbl>, ham_5_im <dbl>, ham_6_di <dbl>,
## #   ham_7_wi <dbl>, ham_8_re <dbl>, ham_9_ag <dbl>, ham_10_psya <dbl>,
## #   ham_11_soma <dbl>, ham_12_gi <dbl>, ham_13_gs <dbl>, ham_14_sex <dbl>,
## #   ham_15_hd <dbl>, ham_16_li <dbl>, ham_17_weight <dbl>
```

```
# Step 5: Define a function to calculate HAM scores and clean data for HAM_Protect

clean_and_score <- function(data, selected_ids, id_column) {
  # Filter data for selected IDs (using the new_id column in HAM_protect_clean)
  data <- data[data[[id_column]] %in% selected_ids, ]

  # Identify HAM columns
  ham_columns <- grep("^ham_", names(data), value = TRUE)

  # Exclude ham_3a to ham_3e columns
  exclude_columns <- grep("^ham_3[abcde]$", ham_columns, value = TRUE)
  ham_columns <- setdiff(ham_columns, exclude_columns)

  # Check for non-numeric values before conversion
  print("Checking non-numeric values in HAM columns before conversion:")
  print(unique(unlist(data[ham_columns])))

  # Convert HAM columns to numeric (ignoring warnings)
  data[ham_columns] <- lapply(data[ham_columns], function(x) suppressWarnings(as.numeric(as.char
acter(x))))

  # Check if any conversion failed (NA introduced where there shouldn't be any)
  if (any(is.na(data[ham_columns]))) {
    print("Warning: Some values could not be converted to numeric. Check the columns above.")
  }

  # Calculate total HAM score (sum, ignoring NAs, excluding 3a to 3e)
  data$total_ham <- apply(data[, ham_columns], 1, function(row) {
    valid_values <- row[!is.na(row)]
    if (length(valid_values) == 0) {
      return(NA)  # Return NA if all values are NA
    } else {
      return(sum(valid_values, na.rm = TRUE))  # Sum the non-NA values
    }
  })

  return(data)
}


#Apply the function to HAM_protect_clean
HAM_protect_clean <- clean_and_score(HAM_protect_clean, selected_ids, "new_id")
```

```
## [1] "Checking non-numeric values in HAM columns before conversion:"
## [1] "2"    "3"    NA     "0"    "1"    "4"    "9"    "NASK"
## [1] "Warning: Some values could not be converted to numeric. Check the columns above."
```

```
# Remove the ID column and rename new_id to ID
HAM_protect_clean <- HAM_protect_clean %>%
  select(-ID) %>%
  rename(ID = new_id)%>%
  select(ID, everything())

print(head(HAM_protect_clean))
```

```
##     ID      timepoint    bq_date   fug_date ham_1_dm ham_2_gf ham_3_su
## 1 2027 baseline_arm_1 2003-04-20       <NA>        2        3        2
## 2 2027  3_month_arm_1       <NA> 2003-07-09        3        2        2
## 3 2027   1_year_arm_1       <NA> 2004-04-12        3        3        2
## 4 2027   2_year_arm_1       <NA> 2005-03-27        3        0        0
## 5 2027   3_year_arm_1       <NA> 2006-04-19        3        3        2
## 6 2027   4_year_arm_1       <NA> 2007-04-26        3        1        2
##   ham_3a_wl ham_3b_wd ham_3c_rld ham_3d_asa ham_3e_pdw ham_4_ii ham_5_im
## 1        NA        NA         NA         NA         NA        2        2
## 2        NA        NA         NA         NA         NA        0        0
## 3        NA        NA         NA         NA         NA        2        1
## 4        NA        NA         NA         NA         NA        0        2
## 5        NA        NA         NA         NA         NA        0        2
## 6        NA        NA         NA         NA         NA        0        2
##   ham_6_di ham_7_wi ham_8_re ham_9_ag ham_10_psya ham_11_soma ham_12_gi
## 1        1        2        2        1           3           2         0
## 2        1        3        1        1           4           3         1
## 3        1        3        1        1           1           2         0
## 4        1        0        0        1           3           2         0
## 5        1        3        1        0           3           2         1
## 6        0        2        0        0           0           1         0
##   ham_13_gs ham_14_sex ham_15_hd ham_16_li ham_17_weight total_ham
## 1         2          0         2         0             0        26
## 2         2          0         2         0             2        27
## 3         2          2         0         0             0        24
## 4         2          0         0         0             0        14
## 5         2          0         2         0             2        27
## 6         2          0         0         0             0        13
```

```r
#repeat for sleep

clean_and_score_sleep <- function(data, selected_ids, id_column) {
  # Filter data for selected IDs (using the ID column in ham_sleep)
  data <- data[data[[id_column]] %in% selected_ids, ]

  # Identify HAM columns
  ham_columns <- grep("^ham_", names(data), value = TRUE)

  # Exclude ham_3a to ham_3e columns
  exclude_columns <- grep("^ham_3[abcde]$", ham_columns, value = TRUE)
  ham_columns <- setdiff(ham_columns, exclude_columns)

  # Check for non-numeric values before conversion
  print("Checking non-numeric values in HAM columns before conversion:")
  print(unique(unlist(data[ham_columns])))

  # Convert HAM columns to numeric (ignoring warnings)
  data[ham_columns] <- lapply(data[ham_columns], function(x) suppressWarnings(as.numeric(as.char
acter(x))))

  # Check if any conversion failed (NA introduced where there shouldn't be any)
  if (any(is.na(data[ham_columns]))) {
    print("Warning: Some values could not be converted to numeric. Check the columns above.")
  }

  # Calculate total HAM score (sum, ignoring NAs, excluding 3a to 3e)
  data$total_ham <- apply(data[, ham_columns], 1, function(row) {
    valid_values <- row[!is.na(row)]
    if (length(valid_values) == 0) {
      return(NA)  # Return NA if all values are NA
    } else {
      return(sum(valid_values, na.rm = TRUE))  # Sum the non-NA values
    }
  })

  return(data)
}


# Call the function with the appropriate arguments for Ham_sleep
HAM_sleep_clean <- clean_and_score_sleep(HAM_sleep, id_map_filtered$new_id, "ID")
```

```
## [1] "Checking non-numeric values in HAM columns before conversion:"
## [1]  3  2  0  1  4 NA  9
## [1] "Warning: Some values could not be converted to numeric. Check the columns above."
```

```r
print(head(HAM_sleep_clean))
```

```
## # A tibble: 6 × 27
## # Groups:   ID [2]
##      ID timepoint     bq_date    fug_date   ham_1_dm ham_2_gf ham_3_su ham_3a_wl
##   <dbl> <chr>         <date>     <date>        <dbl>    <dbl>    <dbl>     <dbl>
## 1  2068 baseline_ss_… 2024-01-08 NA                3        3        0         9
## 2  2068 week_1_ss_ar… NA         2024-01-16        3        1        0         9
## 3  2068 week_2_ss_ar… NA         2024-01-23        2        0        0         9
## 4  2068 week_3_ss_ar… NA         2024-01-30        2        0        0         9
## 5  2069 baseline_ss_… 2023-11-22 NA                0        1        0         9
## 6  2069 week_1_ss_ar… NA         2023-11-29        1        1        0         9
## # ℹ 19 more variables: ham_3b_wd <dbl>, ham_3c_rld <dbl>, ham_3d_asa <dbl>,
## #   ham_3e_pdw <dbl>, ham_4_ii <dbl>, ham_5_im <dbl>, ham_6_di <dbl>,
## #   ham_7_wi <dbl>, ham_8_re <dbl>, ham_9_ag <dbl>, ham_10_psya <dbl>,
## #   ham_11_soma <dbl>, ham_12_gi <dbl>, ham_13_gs <dbl>, ham_14_sex <dbl>,
## #   ham_15_hd <dbl>, ham_16_li <dbl>, ham_17_weight <dbl>, total_ham <dbl>
```

```
#remove rows with NA values for total_ham
HAM_protect_clean <- HAM_protect_clean[!is.na(HAM_protect_clean$total_ham), ]
HAM_sleep_clean <- HAM_sleep_clean[!is.na(HAM_sleep_clean$total_ham), ]

# Combine ham_protect_clean and ham_sleep_clean
combined_data <- bind_rows(HAM_protect_clean, HAM_sleep_clean)

# Ensure that 'timepoint' is a factor (so it doesn't interfere with date comparison)
combined_data$timepoint <- as.factor(combined_data$timepoint)

# Convert 'bq_date' and 'fug_date' to date format if they aren't already
combined_data$bq_date <- as.Date(combined_data$bq_date, format="%Y-%m-%d")
combined_data$fug_date <- as.Date(combined_data$fug_date, format="%Y-%m-%d")

# Create a new column for the first consent date: choose the earlier date between bq_date and fu
g_date
combined_data$first_consent_date <- pmin(combined_data$bq_date, combined_data$fug_date, na.rm =
TRUE)


# For each unique ID, calculate the earliest consent date
combined_data <- combined_data %>%
  group_by(ID) %>%
  mutate(first_consent_date = min(first_consent_date, na.rm = TRUE)) %>%
  ungroup()


# Calculate mean HAM score (excluding 3a-3e)
combined_data <- combined_data %>%
  group_by(ID) %>%
  mutate(mean_ham = mean(total_ham, na.rm = TRUE)) %>%
  ungroup()

combined_data <- combined_data %>%
  group_by(ID) %>%
  mutate(
    # Calculate the time difference from first consent date for each row
    time_diff = abs(as.numeric(difftime(bq_date, first_consent_date, units = "days")) - 365),

    # Find the row with the smallest time difference (closest to 1 year after first consent)
    ham_one_year_after = total_ham[which.min(time_diff)],

    # Calculate the time difference from today's date for each row (latest score)
    latest_time_diff = abs(as.numeric(difftime(Sys.Date(), pmin(bq_date, fug_date, na.rm = TRU
E), units = "days"))),

    # Find the latest HAM score closest to today's date
    latest_ham = total_ham[which.min(latest_time_diff)]
  ) %>%
  ungroup()
```

```
print(head(combined_data))
```

```
## # A tibble: 6 × 33
##      ID timepoint     bq_date    fug_date   ham_1_dm ham_2_gf ham_3_su ham_3a_wl
##   <dbl> <fct>         <date>     <date>        <dbl>    <dbl>    <dbl>     <dbl>
## 1  2027 baseline_arm… 2003-04-20 NA                2        3        2        NA
## 2  2027 3_month_arm_1 NA         2003-07-09        3        2        2        NA
## 3  2027 1_year_arm_1  NA         2004-04-12        3        3        2        NA
## 4  2027 2_year_arm_1  NA         2005-03-27        3        0        0        NA
## 5  2027 3_year_arm_1  NA         2006-04-19        3        3        2        NA
## 6  2027 4_year_arm_1  NA         2007-04-26        3        1        2        NA
## # ℹ 25 more variables: ham_3b_wd <dbl>, ham_3c_rld <dbl>, ham_3d_asa <dbl>,
## #   ham_3e_pdw <dbl>, ham_4_ii <dbl>, ham_5_im <dbl>, ham_6_di <dbl>,
## #   ham_7_wi <dbl>, ham_8_re <dbl>, ham_9_ag <dbl>, ham_10_psya <dbl>,
## #   ham_11_soma <dbl>, ham_12_gi <dbl>, ham_13_gs <dbl>, ham_14_sex <dbl>,
## #   ham_15_hd <dbl>, ham_16_li <dbl>, ham_17_weight <dbl>, total_ham <dbl>,
## #   first_consent_date <date>, mean_ham <dbl>, time_diff <dbl>,
## #   ham_one_year_after <dbl>, latest_time_diff <dbl>, latest_ham <dbl>
```

```
final_df <- combined_data %>%
  group_by(ID) %>%
  summarise(
    # Sum of total_ham for each ID
    total_ham_sum = sum(total_ham, na.rm = TRUE),

    # Mean of total_ham for each ID
    mean_ham = mean(total_ham, na.rm = TRUE),

    # Score closest to one year after first consent date
    ham_one_year_after = first(ham_one_year_after),

    # Latest score (closest to today's date)
    latest_ham = first(latest_ham)
  ) %>%
  ungroup()

print(final_df)
```

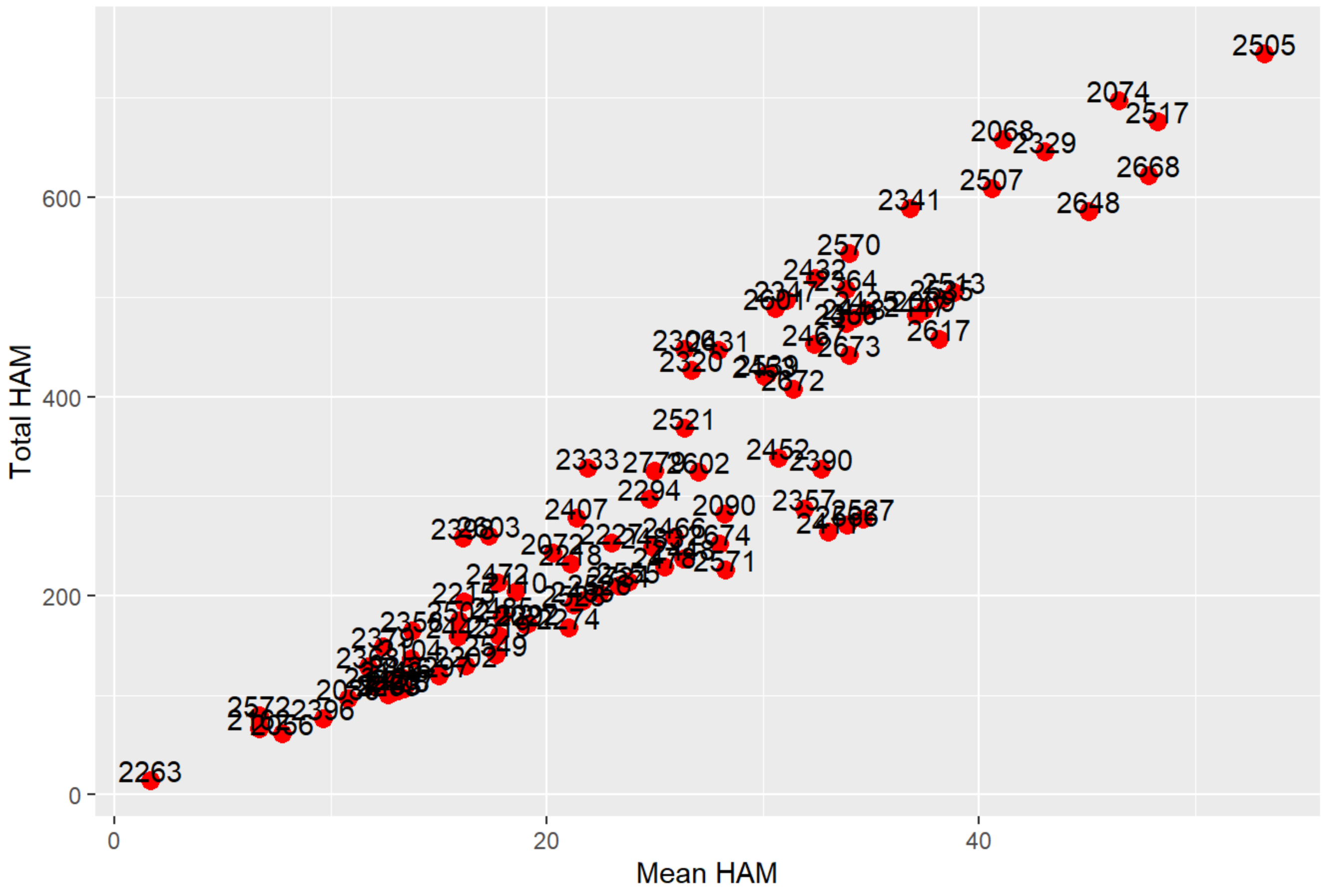
```
## # A tibble: 90 × 5
##       ID total_ham_sum mean_ham ham_one_year_after latest_ham
##    <dbl>         <dbl>    <dbl>              <dbl>      <dbl>
##  1  2027           171    19                    26          9
##  2  2056            62     7.75                 26          4
##  3  2068           658    41.1                  39         61
##  4  2069           487    37.5                  11         56
##  5  2072           243    20.2                  14          6
##  6  2074           697    46.5                  32         70
##  7  2086            97    10.8                  32          9
##  8  2090           282    28.2                  17         58
##  9  2104           137    13.7                  22         11
## 10  2110           204    18.5                  21         59
## # ℹ 80 more rows
```

```
#Plot to see MEAN HAM vs Total HAM
library(reshape2)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.2
```

```
ggplot(final_df, aes(x = mean_ham, y = total_ham_sum, label = ID)) +
  geom_point(size = 3, color = "red") +
  geom_text(nudge_y = 10) +
  labs(title = "Mean HAM vs Total HAM Score", x = "Mean HAM", y = "Total HAM")
```

Mean HAM vs Total HAM Score


Total HAM
Mean HAM