Data Report: Preprocessing of R Codes for EHRs

By Patrick Zhang

Table of Contents[1]

Introduction

This report aims to explain and report the R codes for preprocessing EHRs collected in Canadian hospitals during COVID-19. Three R code files were written for this research: ***Final_Canada_hosp.R***, ***smoking_and_data_merge.R***, and ***medicine.R***. Each code handles different contents of EHRs (comorbidity_other, smoking history, and medication records). This report aims to analyze the functions, execution flow and output results of three R code files in detail. The purpose of this study is to prepare for the subsequent regression by finally integrating these three data into a complete data set that can be analyzed (one-hot-encoding matrix).
The original Excel data set Canada_Hosp1_COVID_InpatientData.xlsx can be accessed in Github: https://github.com/Haimonti/CarePathways/tree/Data-Clean-Canada_Hosp. Please note that the code was written in the following order in this study: Final_Canada_hosp.R, smoking_and_data_merge.R, and medicine.R. The last few lines of smoking_and_data_merge.R and medicine.R involve some data merging content, not all of which are about data cleaning of specific content. The following is an explanation and description of the functions of each code.

Other Comorbidities

The data processed by Final_Canada_hosp.R is comorbidities_other, which can be found in column J of the Data-at-admission sheet in the excel data set.

1. Before proceeding with the formal data cleaning, please run the R packages for subsequent use.

```
library(readxl)
library(tidytext)
library(dplyr)
library(stringr)
library(ggplot2)
library(qdapDictionaries)
data(GradyAugmented)
```

[1] Click to jump to the page you want.

2. Import the original Excel data. (Make sure to put the correct location of the Excel dataset)
    - Convert comorbidities_other to character type.
    - Replace missing values (NA) with blank strings.

```
df <- read_excel("Your_file_location")
df$comorbidities_other <- as.character(df$comorbidities_other)
df$comorbidities_other[is.na(df$comorbidities_other)] <- ""
```

3. Create global variables and set them to 0 to help with data statiswtics. Also create lists and vectors to record detailed information.

```
removed_numbers <- 0
removed_typos <- 0
removed_special_chars <- 0
removed_non_english <- 0
year_list <- list()
special_chars_list <- list()
typos_removed <- character(0)
non_english_removed_list <- character(0)
stop_words_removed <- character(0)
```

4. Manually create a medical terms dictionary (You can add more if needed).

```
base_medical_terms <- c(
  "gerd", "gastroesophageal", …, "valvulopathy"
)
additional_terms <- c(
  "osteoarthritis", "hypothyroid", …,"volvulussigmoid"
)
allowed_medical_terms <- unique(c(base_medical_terms, additional_terms))
```

5. A function to perform unified data cleaning and organization:
    - Text normalization: convert everything to lowercase, remove leading and trailing spaces, and split phrases by commas.
    - Year extraction: use regular expressions to extract 4-digit years (starting with 19 or 20). Store in year_list and remove from text.
    - Number processing: remove all non-year numbers (such as "5mg") and update the removed_numbers counter.
    - Special character removal: extract and record special characters (such as #, &) and delete them from text.
    - Spelling error filtering: remove invalid words such as single letters and repeated letters (such as "aa", "bbb") and record them in typos_removed.

- Term verification: only keep standard English words (data GradyAugmented) or predefined medical terms.
- Stop word processing: remove common stop words (such as "the", "at", "on").

```r
clean_text <- function(text) {
  text <- tolower(text)
  text <- trimws(text)
  phrases <- unlist(strsplit(text, ",\\s*"))
  all_words <- character(0)

  for (phrase in phrases) {
    phrase <- trimws(phrase)
    years <- str_extract_all(phrase, "\\b(19|20)\\d{2}\\b")[[1]]
    year_list <<- c(year_list, years)
    phrase <- gsub("\\b(19|20)\\d{2}\\b", "", phrase)

    # --- Remove Non-Year Digits ---
    non_year_digits <- str_extract_all(phrase, "\\d+")[[1]]
    removed_numbers <<- removed_numbers + length(non_year_digits)
    phrase <- gsub("\\d+", "", phrase)

    # --- Special Characters ---
    special_chars <- str_extract_all(phrase, "[^a-z ]")[[1]]
    special_chars <- special_chars[special_chars != "" & !is.na(special_chars)]
    special_chars_list <<- c(special_chars_list, special_chars)
    phrase <- gsub("[^a-z ]", "", phrase)

    # --- Split into Words ---
    words_list <- unlist(strsplit(phrase, "\\s+"))
    words_list <- words_list[words_list != ""]

    # --- Track and Remove Typos ---
    remove_condition <- (nchar(words_list) == 1) |
      ((nchar(words_list) == 2) & grepl("^([a-z])\\1$", words_list)) |
      ((nchar(words_list) == 3) & grepl("^([a-z])\\1{2}$", words_list))
    removed_typos_words <- words_list[remove_condition]
    typos_removed <<- c(typos_removed, removed_typos_words)
    removed_typos <<- removed_typos + length(removed_typos_words)
    words_list <- words_list[!remove_condition]

    # --- Validate Words (Non-English Check) ---
    words_before_valid <- words_list
    valid_words <- words_list[words_list %in% c(GradyAugmented, allowed_medical_terms)]
    removed_non_eng <- setdiff(words_before_valid, valid_words)
    non_english_removed_list <<- c(non_english_removed_list, removed_non_eng)
    removed_non_english <<- removed_non_english + length(removed_non_eng)

    # --- Remove Stop Words (but keep if in allowed_medical_terms)
    current_stop_words <- valid_words[valid_words %in% stop_words$word & !(valid_words
%in% allowed_medical_terms)]
    stop_words_removed <<- c(stop_words_removed, current_stop_words)
    words_df <- data.frame(word = valid_words, stringsAsFactors = FALSE)
    words_df <- words_df %>% filter(!(word %in% stop_words$word) | (word %in%
allowed_medical_terms))
    all_words <- c(all_words, words_df$word)
  }
  paste(all_words, collapse = " ")
}
```

6. Add the cleaned data to the original data set.

```
df$cleaned_comorbidities <- sapply(df$comorbidities_other, clean_text)
```

7. Generate data reports (csv file) using the cleaned data set.
   a. **Cleaned Dictionary**: You may access this to check if there is anything wrong in the dictionary.

   ```
   all_words_unique <- unlist(strsplit(df$cleaned_comorbidities, "\\s+")) %>%
     .[. != ""] %>%
     unique()
   dictionary_df <- data.frame(
     term = all_words_unique,
     stringsAsFactors = FALSE
   )
   write.csv(dictionary_df, "cleaned_dictionary.csv", row.names = FALSE)
   ```

   b. **Year Extraction Report**: This is the frequency counts for all extracted years.

   ```
   year_report <- data.frame(
     Year = unlist(year_list),
     stringsAsFactors = FALSE
   ) %>%
     count(Year, name = "Count") %>%
     arrange(desc(Count))
   write.csv(year_report, "counts_of_year_extraction_report.csv", row.names = FALSE)
   ```

c. **Medical Event in Year Report**: This is a supplement to the Year Extraction Report in part b, which contains specific information about the medical events that occurred each year.

```
medical_events_list <- list()
for(i in seq_along(df$comorbidities_other)) {
  text <- tolower(df$comorbidities_other[i])
  matches <- str_match_all(text, "([a-z ]+?)\\s+(\\b(?:19|20)\\d{2}\\b)")[[1]]
  if(nrow(matches) > 0){
    for(j in 1:nrow(matches)){
      event <- trimws(matches[j,2])
      year <- matches[j,3]
      if(event != ""){
        medical_events_list <- append(medical_events_list, list(c(year, event)))
      }
    }
  }
}
if(length(medical_events_list) > 0){
  medical_events_df <- do.call(rbind, medical_events_list)
  colnames(medical_events_df) <- c("year", "medical_event")
  medical_events_df <- as.data.frame(medical_events_df, stringsAsFactors = FALSE)
} else {
  medical_events_df <- data.frame(year = character(0), medical_event = character(0))
}
write.csv(medical_events_df, "counts_of_years_of_medical_events.csv", row.names =
FALSE)
```

d. **Years Since Last Medical Events**: Use this year (2025) as the measurement value to calculate years passed since last medical event. All data should be integers and positive (greater than or equal to 0).

```
df$year_extracted <- as.numeric(str_extract(df$comorbidities_other,
"\\b(19|20)\\d{2}\\b"))
df <- df %>%
  mutate(years_since_medical_events = ifelse(!is.na(year_extracted), 2025 -
year_extracted, 0))
df_result <- df %>% select(id, years_since_medical_events)
write.csv(df_result, "years_since_medical_events.csv", row.names = FALSE)
```

e. **Special Characters Report**: Here is a statistics of all the special characters that were removed and their counts.

```
special_char_report <- data.frame(
  Character = unlist(special_chars_list),
  stringsAsFactors = FALSE
) %>%
  filter(Character != "" & !is.na(Character)) %>%
  count(Character, name = "Count") %>%
  arrange(desc(Count))
write.csv(special_char_report, "special_characters_report.csv", row.names = FALSE)
```

f. **Typo Analysis Report**: A typo is considered a single letter word or a word with two or three repeated letters. This report records the frequency of each typo.

```
typo_lengths <- nchar(typos_removed)
typo_analysis_summary <- data.frame(length = typo_lengths) %>%
  group_by(length) %>%
  summarise(count = n()) %>%
  filter(length %in% c(1, 2, 3))
write.csv(typo_analysis_summary, "typo_analysis_report.csv", row.names = FALSE)
```

g. **List of Removed Typos Report**: This is a supplementary report to the Typo Analysis Report, which contains information about what each typo is and their counts.

```
typo_report <- data.frame(
  Typo = typos_removed,
  stringsAsFactors = FALSE
) %>%
  count(Typo, name = "Count") %>%
  arrange(desc(Count))
write.csv(typo_report, "list_of_removed_typos_with_counts.csv", row.names = FALSE)
```

h. **Non-English Words Report**: Here is a report about all the non-English words that were removed and their counts. It may need to be examined more carefully, as some may be abbreviations or advanced medical terms that need to be added to the manual medical dictionary.

```
non_eng_report <- data.frame(
  NonEnglishWord = non_english_removed_list,
  stringsAsFactors = FALSE
) %>%
  count(NonEnglishWord, name = "Count") %>%
  arrange(desc(Count))
write.csv(non_eng_report, "non_English_words_report.csv", row.names = FALSE)
```

i. **Stop Words Report**: Here is a report about all the stop words that were removed and their counts.

```
stop_words_report <- data.frame(
  StopWord = stop_words_removed,
  stringsAsFactors = FALSE
) %>%
  count(StopWord, name = "Count") %>%
  arrange(desc(Count))
write.csv(stop_words_report, "stop_words_count.csv", row.names = FALSE)
```

j. **Cleanup Summary Report**: Here is some summary information about the frequency of each type of data being removed.

```
cleanup_summary <- data.frame(
  Metric = c(
    "Total Numbers Removed (non-years)",
    "Total Years Extracted",
    "Total Typos Removed",
    "Total Special Characters Removed",
    "Total Non-English Words Removed",
    "Total Stop Words Removed"
  ),
  Count = c(
    removed_numbers,
    sum(year_report$Count),
    sum(typo_report$Count),
    sum(special_char_report$Count),
    removed_non_english,
    sum(stop_words_report$Count)
  )
)
write.csv(cleanup_summary, "cleanup_summary_report.csv", row.names = FALSE)
```

8. Generate a binary matrix (patient id * medical terms): Use the cleaned medical dictionary as the column variable. Assign 1 to the patient if its record has this medical term, 0 otherwise.

```
create_binary_matrix <- function(cleaned_texts, dictionary_terms) {
  t(sapply(cleaned_texts, function(text) {
    words <- unlist(strsplit(text, "\\s+"))
    as.integer(dictionary_terms %in% words)
  }))
}
```

```
binary_matrix <- create_binary_matrix(df$cleaned_comorbidities, all_words_unique)
binary_df <- data.frame(id = df$id, binary_matrix, stringsAsFactors = FALSE)
colnames(binary_df) <- c("id", all_words_unique)
print(paste("Matrix dimensions:", nrow(binary_df), "x", ncol(binary_df)))
write.csv(binary_df, "binary_matrix.csv", row.names = FALSE)
```

Smoking History

The data processed by smoking_and_data_merge.R is smoking history and the year they quit, which can be found in column N & O of the Data-at-admission sheet in the excel data set. The purpose of this section is to convert qualitative data into quantitative data and use it for subsequent regression analysis of the impact of smoking history.

The code also involves data integration with an existing dataset new_data (which can be found in Github) and some quanlitative data. These quanlitative are: data are days_in_hospital_prior_to_expiration, hospital_length_of_stay, icu_length_of_stay, days_in_hospital_prior_to_icu_admission, time_on_mechanical_ventilation, days_in_hospital_prior_to_mechanical_ventilation, and days_to_first_covid19_test_negative. These data can be found in Columns E - K of the Hospital-length-of-stay sheet in the original Excel file.

1. Before proceeding with the formal data cleaning, please run the R packages for subsequent use.

   ```
   library(readxl)
   library(dplyr)
   library(readr)
   smoking_df <- read_excel("Your_file_location")
   ```

2. Report of the frequency of each type of smoking status.

   ```
   smoking_counts <- smoking_df %>%
     count(smoking_history)
   print(smoking_counts)
   write.csv(smoking_counts, "smoking_counts.csv", row.names = FALSE)
   ```

3. Filter for Ex-smokers and calculate years since quitting (2025 - year_they_quit)

   ```
   ex_smokers <- smoking_df %>%
     filter(smoking_history == "Ex-smoker") %>%
     mutate(
       year_they_quit = ifelse(is.na(year_they_quit) | year_they_quit == "", NA, year_they_quit),
       years_since_quit = ifelse(!is.na(year_they_quit), 2025 - as.numeric(year_they_quit), NA)
     ) %>%
     select(id, years_since_quit)
   ```

```
write.csv(ex_smokers, "ex_smokers_years_quit.csv", row.names = FALSE)
```

4. Create all_patients_smoking_history.csv with numeric encoding for smoking_history.

```
all_patients_smoking <- smoking_df %>%
  mutate(
    smoking_history_numeric = case_when(
      smoking_history == "Non-smoker" ~ 1,
      smoking_history == "Ex-smoker" ~ 2,
      smoking_history == "Smoker: < 30 pack years" ~ 3,
      smoking_history == "Smoker: pack years unknown" ~ 4,
      is.na(smoking_history) | smoking_history == "" ~ 0,
      TRUE ~ 0
    ),
    years_since_quit = case_when(
      smoking_history == "Ex-smoker" ~ 2025 - as.numeric(year_they_quit),
      TRUE ~ NA_real_
    )
  ) %>%
  select(id, smoking_history_numeric, years_since_quit)
write.csv(all_patients_smoking, "all_patients_smoking_history.csv", row.names = FALSE, na =
"")
```

Notes:

- *Non-smoker = 1*
- *Ex-smoker = 2*
- *Smoker: < 30 pack years = 3*
- *Smoker: pack years unknown = 4*
- *If it is empty, then it will be 0*

The following steps involve data integration:

5. Data integration with new_data by patient id.

```
new_data <- read_csv("Downloads/new.csv")
new_data_modified <- new_data %>% select(-smoking_history)
merged_data <- left_join(new_data_modified, all_patients_smoking, by = "id")
merged_data <- merged_data %>%
  mutate(years_since_quit = ifelse(is.na(years_since_quit), 0, years_since_quit))
if ("smoking_history_numeric" %in% colnames(merged_data) & "years_since_quit" %in%
colnames(merged_data)) {
  merged_data <- merged_data %>%
    relocate(smoking_history_numeric, years_since_quit, .after = weight)
}
```

```
merged_data <- merge(merged_data, df_result, by = "id", all.x = TRUE)
if (”years_since_medical_events" %in% colnames(merged_data) & "years_since_quit" %in%
colnames(merged_data)) {
  merged_data <- merged_data %>%
    relocate(years_since_medical_events, .after = years_since_quit)
}
merged_data <- merge(merged_data, binary_df, by = "id", all.x = TRUE)
write.csv(merged_data, "merged_new.csv", row.names = FALSE)
```

6. Data integration with the length of stay data.

```
df_additional <- read_excel("Your_file_location")
columns_to_keep <- c("id", "days_in_hospital_prior_to_expiration", "hospital_length_of_stay",
                     "icu_length_of_stay", "days_in_hospital_prior_to_icu_admission",
                     "time_on_mechanical_ventilation",
"days_in_hospital_prior_to_mechanical_ventilation",
                     "days_to_first_covid19_test_negative")
df_additional <- df_additional %>% select(all_of(columns_to_keep))
merged_data <- merge(merged_data, df_additional, by = "id", all.x = TRUE)
merged_data <- merged_data %>% mutate_all(~replace(., is.na(.), 0))
write.csv(merged_data, "merged_new.csv", row.names = FALSE)
```

Medication Dosage & Frequency

The data processed by medication.R is medications, which can be found in column M of the Data-at-admission sheet in the Excel data set. The purpose of this section is to perform regression analysis on the dosage and frequency of medication through data processing. This code will finally merge the data with the new dataset generated at the end of smoking_and_data_merge.R.

1. Before proceeding with the formal data cleaning, please run the R packages for subsequent use.

```
library(readxl)
library(jsonlite)
library(dplyr)
library(tidyr)
library(stringr)
library(tibble)
library(forcats)
library(readr)
library(data.table)
```

2. Import the original Excel data.

```
df <- read_excel("Your_file_location") %>%
  mutate(id = as.character(id))
```

3. Create an empty table data to store the parsed drug name, dosage, and frequency information.

```
data <- tibble(
  id = character(),
  medications = character(),
  dosage = character(),
  frequency = character()
)
```

4. Process JSON data and perform data re-organization.
    - Missing value processing: If the medications field is empty, directly record the empty line and skip subsequent processing.
    - JSON cleaning: The JSON in the original data may contain line breaks and escape characters, which need to be cleaned and parsed.
    - JSON parsing: Use fromJSON to convert the string into structured data (list or data frame).
    - Data extraction: Extract the drug name, dosage, and frequency line by line according to the JSON structure and store them in medication_data.
    - Empty value processing: The %||% operator ensures that missing fields are replaced with empty strings.

```
`%||%` <- function(a, b) if (!is.null(a)) a else b

for (i in seq_len(nrow(df))) {

  row_id <- df$id[i]

  meds_string <- df$medications[i]

  if (is.na(meds_string) || meds_string == "") {

    medication_data <- bind_rows(medication_data, tibble(

      id = row_id, medications = "", dosage = "", frequency = ""

    ))

    next

  }

    meds_string_clean <- meds_string %>%

    str_replace_all("[\r\n]", "") %>%
  str_replace_all("\\\\", "") %>%
  str_replace_all('^"|"$', "")
meds_list <- fromJSON(meds_string_clean)
if (is.null(meds_list) || length(meds_list) == 0) {
  medication_data <- bind_rows(medication_data, tibble(
    id = row_id, medications = "", dosage = "", frequency = ""
  ))
  next
}
if (is.data.frame(meds_list)) {
  for (j in seq_len(nrow(meds_list))) {
    medication_data <- bind_rows(medication_data, tibble(
      id = row_id,
      medications = meds_list$medications[j] %||% "",
      dosage = meds_list$dosage[j] %||% "",
      frequency = meds_list$frequency[j] %||% ""
    ))
  }
} else if (is.list(meds_list)) {
  for (j in seq_along(meds_list)) {
    entry <- meds_list[[j]]
    if (is.list(entry)) {
      medication_data <- bind_rows(medication_data, tibble(
        id = row_id,
        medications = entry$medications %||% "",
        dosage = entry$dosage %||% "",
        frequency = entry$frequency %||% ""
      ))
    } else {
      medication_data <- bind_rows(medication_data, tibble(
        id = row_id,
        medications = as.character(entry) %||% "",
        dosage = "",
        frequency = ""
      ))
    }
  }
} else {
  medication_data <- bind_rows(medication_data, tibble(
    id = row_id,
    medications = as.character(meds_list) %||% "",
    dosage = "",
    frequency = ""
  ))
}
}
write_csv(data, "medication_data.csv")
```

5. Standardize drug names and dosages. Convert the JSON formatted data into a table format that is easier to read. Each medication used by an ID will have a row of data, including medication name, dosage, and frequency. An ID may have several rows or none. *You may add more rules for standarization.*

```
data <- data %>%
  filter(!is.na(medications) & str_trim(medications) != "")
invalid_freqs <- c("Please Select an option", "NA", "", NA)
data <- data %>%
  mutate(frequency = ifelse(toupper(frequency) %in% toupper(invalid_freqs), "", frequency))
formats_to_remove <- c(
  "TABLETS?", "CAPLETS?", …, "&AMP;"
)
format_regex <- paste0("(?i)\\b(", paste(formats_to_remove, collapse = "|"), ")\\b")
data <- data %>%
  mutate(medications = toupper(medications),
         medications = gsub(format_regex, "", medications),
         …,
         medications = ifelse(str_detect(medications, "ACETAMIOPHEN"), "ACETAMINOPHEN", medications)
  )
```

6. Generates different reports and a one-hot encoding matrix.
   a. Frequency Report: Medication use frequency statistics (e.g., "BID," "TID").

```
freq_report <- data %>%
  filter(frequency != "") %>%
  count(frequency, name = "count")
write.csv(freq_report, "frequency_report.csv", row.names = FALSE)
```

   b. Medications Report: Number of people using each medication.

```
med_report <- data %>%
  count(medications, name = "count")
write.csv(med_report, "medications_report.csv", row.names = FALSE)
```

   c. One-Hot Matrix: The rows represent patient IDs, and the columns represent medications. 1 means the patient used the medication, 0 otherwise.

```
one_hot <- data %>%
  distinct(id, medications) %>%
  mutate(value = 1L) %>%
  pivot_wider(
    id_cols = id,
    names_from = medications,
    values_from = value,
    values_fill = list(value = 0L)
```

```
 )
write.csv(one_hot, "one_hot_matrix.csv", row.names = FALSE)
```

d. Medication Dosages Report: A summary of all doses of each medication (e.g., "500 mg," "200 mg")

```
dosage_report <- data %>%
  filter(!is.na(dosage) & str_trim(dosage) != "") %>%
  group_by(medications) %>%
  summarise(dosages = paste(sort(unique(dosage)), collapse = ", "), .groups = "drop")
write.csv(dosage_report, "medication_dosages.csv", row.names = FALSE)
```

7. Medication frequency encoding and matrix generation. Encode the text frequency into numeric values and use them as values in the matrix. The row variable is id and the column variable is medication_dosage.

```
data <- data %>%
  mutate(
    dosage_clean = ifelse(is.na(dosage) | str_trim(dosage) == "", "UNKNOWN", str_trim(dosage)),
    med_dose = paste(medications, dosage_clean, sep = "_")
  )
data <- data %>%
  mutate(
    frequency = ifelse(is.na(frequency) | str_trim(frequency) == "", "NA", frequency)
  )
freq_map <- c(
  "OD" = 1, "BID" = 2, "TID" = 3, "QUID" = 4,
  "PRN" = 5, "WEEKLY" = 6, "EVERY OTHER DAY" = 7,
  "MWF" = 8, "MONTHLY" = 9, "NA" = 0
)
data <- data %>%
  mutate(
    frequency_clean = toupper(str_trim(frequency)),
    freq_value = freq_map[frequency_clean]
  )
freq_matrix <- data %>%
  select(id, med_dose, freq_value) %>%
  group_by(id, med_dose) %>%
  summarise(freq_value = max(freq_value, na.rm = TRUE), .groups = "drop") %>%
  pivot_wider(names_from = med_dose, values_from = freq_value, values_fill = 0)
write.csv(freq_matrix, "frequency_matrix.csv", row.names = FALSE)
```

8. Final data integration

```
merged_new <- read_csv("Documents/UB/Spring 2025/Research/csv file
extract/merged_new.csv")
merged_data <- merge(merged_new, freq_matrix, by = "id", all.x = TRUE)
merged_data[is.na(merged_data)] <- 0
merged_data <- merged_data[, -2]
write.csv(merged_data, "final_data.csv", row.names = FALSE)
```

Tips & Further Questions

Some packages are based on R.4.3.3, please update your R studio version.

If you have any questions, please contact me.

Enjoy your R code journey!