Investigating Disparities, Clinical Variables, and Predictive Modeling in Organ Procurement

Benzon Carlitos Salazar

February 26, 2025

Project Overview

1. Project Topic

My project will focus on organ procurement disparities, clinical predictors of organ viability, and regression models for procurement success. I will analyze the **Organ Retrieval and Collection of Health Information for Donation (ORCHID) dataset**, which is publicly available from PhysioNet and includes over **133,101 deceased donor referrals** across six **Organ Procurement Organizations (OPOs)** in the U.S.

2. Main Issues and Problems

My project attempts to address three primary issues:

1. **Disparities in Organ Procurement Outcomes** – Do race, socioeconomic status, and geography impact organ procurement rates?
2. **Influence of Clinical Variables on Procurement Success** – Which medical and laboratory parameters best predict procurement success?
3. **Regression Modeling for Organ Procurement Success** – Can regression models accurately predict whether an organ will be successfully procured?

These questions are crucial because organ transplantation relies on efficient allocation and procurement. Understanding disparities and improving predictive capabilities can help optimize transplant decisions and promote equity in the system.

3. Background Information Plan

To contextualize the research, I will review:

- **Existing literature on organ procurement disparities**, including racial and socioeconomic factors influencing donation rates.
- **Clinical criteria for organ viability**, focusing on blood chemistry, hematology, and arterial blood gas levels.
- **Current regression modeling approaches** in organ transplantation, particularly logistic regression and other statistical methods used in medical decision-making.
- **Policies and regulations governing OPOs**, to understand organizational influences on procurement outcomes.

Sources for my review will include academic journals, reports from the United Network for Organ Sharing (UNOS), and relevant medicaldecision-making studies.

4. Data Collection Process

Methods and Tools

I will be relying on publicly available data from the **Organ Retrieval and Collection of Health Information for Donation (ORCHID) dataset**, hosted on **PhysioNet**. The data collection process will involve the following steps:

1. **Data Acquisition:**

- The dataset will be downloaded directly from PhysioNet using their web interface.
- If necessary, the PhysioNet API will be used for automated access.

2. **Data Structure and Organization:**

- The dataset consists of multiple CSV files, including **demographic, clinical, and procedural data** related to organ procurement.
- The tables will be linked using the unique **PatientID** field.

3. **Data Storage and Management:**

- The dataset will be stored in a **secure local repository** or **cloud storage** for easy access.
- R will be used for initial data exploration and preprocessing.

4. **Data Cleaning and Preprocessing:**

- Handling missing values by applying imputation or exclusion methods where appropriate.
- Standardizing variable formats for consistency in analysis.
- Filtering data based on relevant **study criteria**, such as time period, OPOs, or specific donor characteristics.

5. **Tools for Data Processing and Analysis:**

- **R (tidyverse, dplyr, ggplot2, caret, glmnet)** for data manipulation, visualization, and regression modeling.
- **R Markdown** for exploratory data analysis and documentation.

5. Data and Variables

The ORCHID dataset contains **demographic, clinical, and procedure information on deceased donor referrals**. Specifically, the dataset includes:

- **Demographic Data**: Race, age, sex, geographic region, socioeconomic proxies (hospital identifier, OPO region).
- **Clinical Variables**: Blood chemistry, hematology, arterial blood gas levels, infection status, cause of death, comorbidities.
- **Referral and Procurement Details**: Referral source, organ type, authorization status, OPO performance metrics.
- **Outcome Variable**: Whether the organ was successfully procured.

This dataset can be obtained from PhysioNet: https://physionet.org/content/orchid/2.0.0/

6. Questions and Concerns

Questions

1. What are the best statistical methods to adjust for confounding factors in organ procurement disparities?
2. How can missing data in clinical variables be handled to improve regression model accuracy?
3. What are the most effective regression techniques for predicting organ procurement success?

Concerns

- **Class Imbalance**: Procurement success may be significantly rarer than failures, which could affect model validity.
- **Data Bias**: Referral processes and OPO performance may introduce biases into the dataset.

Data Extraction and Clean up

All code chunks will not be evaluated since clean up was already performed prior to knitting this document.

Summary of Data Cleanup Process

The data cleanup involved reading raw CSV files from the ORCHID dataset, transforming them into a structured format, and saving the cleaned data as Parquet files. The key steps included:

1. Reading Raw Data:

- CSV files for various event categories (e.g., ABG, CBC, Chemistry, Culture, Fluid Balance, Hemodynamics, etc.) were read into R.

2. Data Cleaning and Transformation:

- Unnecessary columns (`RowID`, unnamed columns) were dropped.
- Data was grouped by event type (e.g., `abg_name`, `cbc_name`, `chem_name`) and converted to a wide format using `pivot_wider()`.
- Numeric values were coerced into appropriate data types (`double`).
- An `opo_group` identifier was extracted from `PatientID` to categorize patients.
- `tidyr::fill()` was used to propagate missing values up and down within each group.
- Data was grouped by `PatientID`, `time_event`, and other relevant identifiers, then deduplicated using `slice_head()`.

3. Data Output & Cleanup:

- The cleaned datasets were saved in Parquet format for efficient storage and processing.
- Raw CSV files were removed after ensuring successful cleanup.
- Memory was cleared using `rm(list = ls(all.names = TRUE))` and `gc()` to optimize performance.

This process ensured that data was well-structured, deduplicated, and efficiently stored while maintaining data integrity.

Below is an example of what the whole clean up process looked like.

```r
library(magrittr)

orchid_folder <- here::here("data-raw/physionet.org/files/orchid/2.0.0")
```

```r
abg_events <-
  readr::read_csv(here::here(orchid_folder, "ABGEvents.csv"))

abg_events_proc <-
  abg_events %>%
  dplyr::select(-c("...1", "RowID")) %>%
  dplyr::group_by(abg_name) %>%
  dplyr::mutate(row_id = dplyr::row_number()) %>%
  tidyr::pivot_wider(names_from = abg_name, values_from = value) %>%
  dplyr::ungroup() %>%
  dplyr::select(-row_id) %>%
  dplyr::mutate(opo_group = as.factor(substring(PatientID, 0, 4)),
                abg_ventilator_mode = as.character(abg_ventilator_mode),
                PH = as.double(PH),
                PCO2 = as.double(PCO2),
                PO2 = as.double(PO2),
                HCO3 = as.double(HCO3),
                BE = as.double(BE),
                O2SAT = as.double(O2SAT),
                FIO2 = as.double(FIO2),
                Rate = as.double(Rate),
                TV = as.double(TV),
                PEEP = as.double(PEEP),
                PIP = as.double(PIP))

abg_events_final <-
  abg_events_proc %>%
  dplyr::group_by(PatientID, time_event, abg_ventilator_mode, opo_group) %>%
  tidyr::fill(dplyr::everything(), .direction = "updown") %>%
  dplyr::slice_head() %>%
  dplyr::ungroup()

abg_events_final %>% nanoparquet::write_parquet(here::here("data", "abg_events.parquet"))
rm(abg_events, abg_events_proc, abg_events_final)
```

```r
# List of files to remove
files_to_remove <- c(
  "ABGEvents.csv",
  "calc_deaths.csv",
  "CBCEvents.csv",
  "ChemistryEvents.csv",
  "CultureEvents.csv",
  "FluidBalanceEvents.csv",
  "HemoEvents.csv",
  "referrals.csv",
  "SerologyEvents.csv"
)

# Remove files only if they exist
purrr::walk(files_to_remove, ~ {
  file_path <- here::here(orchid_folder, .x)
  if (file.exists(file_path)) {
    file.remove(file_path)
  }
})

rm(list = ls(all.names = TRUE)) # clear all objects including hidden objects
invisible(gc()) # free up memory
```