Recurrent Adverse Health Events Dataset

The dataset was created to support research on the effect of time-varying environmental exposures on recurrent adverse health events, particularly in the context of Medicare beneficiaries. It enables investigators to study how changes in exposures over time influence repeated adverse clinical outcomes and their relationship with dependent terminal events such as cardiovascular disease (CVD) and Alzheimer's disease and related dementias (ADRD).

Table of Contents

- Data Profile
- Access and Usage
- Project Structure
- Environment Setup
- Data Requirements
- Pipeline Execution
- Output Data
- Linking to ZCTA-Level Exposures
- Data Dictionary

Data Profile

- **Description**: Monthly time-varying exposures and recurrent events with dependent terminal events
- **Time Coverage**: 2000-2021
- **Spatial Coverage**: Contiguous United States at zcta leavel
- **Population Coverage**: Elder Medicare beneficiaries (65+ years old)
- **Unit of Analysis**: Beneficiary-month

Access and Usage

These datasets are restricted-use due to the inclusion of sensitive and identifiable health information. As such, the data files themselves cannot be shared via this Dataverse entry.

Students, post-docs, and researchers conducting NSAPH-related research in collaboration with PIs at Harvard or affiliated institutions may gain access to the data, provided they have submitted a project initiation form, are listed on an approved IRB protocol, and have completed all required data security trainings.

Note: No data files are included in this entry. This documentation serves as a metadata reference to the analytical data files generated from the processed CMS data.

Project Structure

For more details on code and to reproduce the data, refer to /lab_share/data_management/data_processing/recurrent_events On ReD computing enviornment.The code will also be available on GitHub:
recurrent-adverse-health-events

```
recurrent_events/
├── conf/icd_codes.yml      # icd code used to tag the admissions
├── data/                   # Data directory structure
│   ├── input/
│   └── output/             # Final analysis datasets
├── notebooks/              # Exploratory analysis notebooks
├── src/                    # Main pipeline scripts and job files
├── requirements.yml        # Conda environment specification file
└── README.md
```

Environment Setup

Prerequisites

- Access to ReD computing environment
- Python environment with required dependencies (pandas, duckdb, pyarrow, etc.) : one can use already created container named mbsf_mortality_denom_env

System Requirements

The pipeline is designed to run on the ReD computing environment with access to:

- `/lab_share/data/lego/medicare/`
- `/lab_share/data/play-doh/`
- `/lab_share/data_management/data_processing/`

Data Requirements

The pipeline requires two main data sources linked from the NSAPH data repository:

1. **Denominator files**: Medicare beneficiary enrollment data by year
2. **Admissions data**: Hospital admission records by year

Pipeline Execution

Initial Setup

Navigate to your project directory and set up the required symbolic links:

```
# Set home directory
export HOME_DIR=$(pwd)

# Link denominator and admissions data
cd $HOME_DIR/data/input/
ln -s /lab_share/data/lego/medicare/normalization_cms .
ln -s /lab_share/data/lego/medicare/mbsf_mortality_denom .

# Link output directory
```

```
cd $HOME_DIR/data/output/
ln -s /lab_share/data/play-doh/recurrent_events .
```

Running the Pipeline

The pipeline consists of 2 sequential jobs managed by SLURM dependencies:

```
cd $HOME_DIR/src/

# Submit jobs
sbatch src/01_bene_output.sbatch
sbatch src/02_adm_output.sbatch
```

Pipeline Steps

1. `01_bene_output.sbatch`: generate final beneficiary parquet files and merging exposures data
2. `02_adm_output.sbatch`: generate final admissions parquet files

Output Data

Final datasets are stored in ReD lab workspace :

```
/lab_share/data/play-doh/recurrent_events/
```

The pipeline generates two types of yearly datasets:

1. Beneficiaries Dataset : Monthly records of Medicare beneficiary characteristics
2. Hospitalizations Dataset : Event-level records of hospital admissions with diagnosis codes

Linking to ZCTA-Level Exposures

Link external exposure data using `zcta`, `year`, and `month` keys to study the effect of time-varying exposures on recurrent events. Inner join retains only beneficiary-months with matching exposure data:

```
# Polars
merged = bene.join(exposure, on=["zcta", "year", "month"], how="inner")

# Pandas
merged = bene.merge(exposure, on=["zcta", "year", "month"], how="inner")

# DuckDB
SELECT b.*, e.pm25 FROM 'bene.parquet' b
INNER JOIN 'exposure.parquet' e USING (zcta, year, month)
```

Note: Inner join excludes beneficiary-months where exposure data is unavailable (e.g., missing ZCTAs or time periods). This ensures complete covariate information for modeling but may reduce sample size.

Data Dictionary

Beneficiaries File (`bene_track_all_yyyy.parquet`)

Variable Name	Type	Description
`zcta`	string	Residential 5-digit zcta code of the beneficiary
`year`	int64	Year of the data
`month`	int64	Month of given year (1-12)
`bene_id`	string	Beneficiary ID (links with `bene_id` from corresponding hospitalization files)
`age`	int64	Age calculated from date of birth (year of the enrollment data - year of DOB)
`dob`	timestamp[ns]	Date of birth (YYYY-MM-DD)
`dod`	timestamp[ns]	Date of death, if it exists (YYYY-MM-DD)
`sex`	string	Biological sex of beneficiary • 0 = Unknown • 1 = Male • 2 = Female
`race`	string	Beneficiary race/ethnicity • 0 = Unknown • 1 = White • 2 = Black • 3 = Other • 4 = Asian • 5 = Hispanic • 6 = North American Native
`y`	int64	Number of years since the beneficiary turned 65
`cohort`	int64	Year the beneficiary turned 65
`HMO`	string	Indicator of monthly HMO enrollment status
`t`	int64	Time since the beneficiary turned 65 (in months)

Hospitalizations File (`adm_track_all_yyyy.parquet`)

Event-level data with one row per hospitalization event:

Variable	Type	Description

Variable	Type	Description
bene_id	string	Beneficiary ID (links with bene_id in beneficiaries file)
adm_id	string	Unique admission/hospitalization ID
admission_date	timestamp[ns]	Date of hospital admission
discharge_date	timestamp[ns]	Date of hospital discharge
year	int64	Year of admission
month	int64	Month of admission (1-12)
day	int64	Day of admission (1-31)
diagnoses	list<item: string>	List of all diagnosis ICD codes for this hospitalization
cohort	int64	Year the beneficiary turned 66 (study entry year)
adrd_dx_1	int	ADRD found in primary (first) diagnosis position (1=Yes, 0=No)
adrd_dx_25	int	ADRD found in any diagnosis position (1=Yes, 0=No)
atrial_fib_dx_1	int	Atrial fibrillation found in primary diagnosis (1=Yes, 0=No)
atrial_fib_dx_25	int	Atrial fibrillation found in any diagnosis position (1=Yes, 0=No)
cardiac_arrest_dx_1	int	Cardiac arrest found in primary diagnosis (1=Yes, 0=No)
cardiac_arrest_dx_25	int	Cardiac arrest found in any diagnosis position (1=Yes, 0=No)
acute_myocard_infarc_dx_1	int	Acute myocardial infarction found in primary diagnosis (1=Yes, 0=No)
acute_myocard_infarc_dx_25	int	Acute myocardial infarction found in any diagnosis position (1=Yes, 0=No)
t	double	Time since study entry at event occurrence (in months)

Outcome Indicators:

- `*_dx_1`: Checks if the condition's ICD code appears in the **primary (first) diagnosis position** only
  - `1` = Primary diagnosis matches one of the condition's ICD codes
  - `0` = Primary diagnosis does not match any ICD code for this condition
- `*_dx_25`: Checks if the condition's ICD code appears in **any of the 25 diagnosis positions**
  - `1` = At least one diagnosis (positions 1-25) matches the condition's ICD codes

- `0` = None of the diagnoses match any ICD code for this condition

Note: The hospitalizations file includes only admissions where **at least one** outcome indicator equals 1. Hospitalizations with no flagged conditions are excluded. ICD code mappings are defined in `conf/icd_codes.yml`.