Analysis document

Brian Ruan • 21 January 2026

Setting up

Analyzing MEPS data requires a handful of packages. This process may take up to 15 minutes depending on your internet speed. You need to run this code **only once**.

```
# # Only need to run these once (comment out after done)
# install.packages("foreign")
# install.packages("devtools")
# install.packages("tidyverse")
# install.packages("readr")
# install.packages("readxl")
# install.packages("haven")
# install.packages("survey")
```

Run this every time you restart R.

You are now ready to start using the MEPS library.

Importing data sets

To import the full-year consolidated data for any specific year, the syntax generally follows the convention `read_MEPS(year = ####, type = "FYC")`. Let's import data from the year 2023. The time it takes to download data will depend on your internet connection.

```
fyc_2023 <- read_MEPS(year = 2023, type = "FYC")
```

To import the prescription data for any specific year, the syntax generally follows the convention `read_MEPS(year = ####, type = "RX")`. Let's import data from the year 2023.

```
rx_2023 <- read_MEPS(year = 2023, type = "RX")
```

To import the medical conditions data for any specific year, the syntax generally follows the convention `read_MEPS(year = ####, type = "RX")`. Let's import data from the year 2023.

```
cond_2023 <- read_MEPS(year = 2023, type = "COND")
```

Saving data sets for faster loading

Importing the data sets take time. For faster loading and processing the next time we start R, you can save the datasets into a single `.rda` file which will combine the three above data sets into an R-compatible format.

```
save(fyc_2023, rx_2023, cond_2023,
     file = "data/meps_data_2023.rda")
```

To load the data set the next time you run R, you can run this line of code after importing your libraries.

```
load("data/meps_data_2023.rda")
```

Codebooks

Codebooks are handily available on the MEPS website. For this example, the data for 2023 can be found here. For the FYC 2023 data in particular, the codebook can be found here.

Recoding

You may want to recode variables in a way that makes analysis easier. Here is an example of recoding marital status in the FYC data.

```
marital_recode <- function(df) {
  df %>%
    mutate(MARITAL = case_when(
      MARRY53X == 1 ~ "MARRIED",
      MARRY53X >= 2 & MARRY53X <=6 ~ "NON-MARRIED",
      MARRY53X == 7 ~ "MARRIED",
      MARRY53X >= 8 ~ "NON-MARRIED",
      TRUE ~ NA_character_
    )) %>%
    select(-MARRY53X)
}
```

What the code does:

- `marital_recode <- function(df)`: We will create a function called `marital_recode` that can take in a data set to recode the marital status variable.
- `mutate(MARITAL = case_when(...)`: We will create a new variable called `MARITAL` that consists of the recoded variables.
- `MARRY53X == 1 ~ "MARRIED"`: If the original coded variable `MARRY53X == 1`, which according to the codebook means "Married," we will this to `MARRIED`, and so on with the other codes.
- `select(-MARRY53X)`: We will delete the old `MARRY53X` variable so we don't use it accidentally in our analysis.

Let's use the function to recode the data.