Identifying Diagnosis and Procedure Codes in R

Using Administrative Data for Clinical and Health Services Research

Overview

- ICD-10-CM/PCS from a programming perspective
- Approaches for the SID data

ICD-10-CM

- Over 68,000 codes at time of transition from ICD-9-CM (just over 14,000 codes)
- 3–7 characters
  - First character is letter
  - Second character is number
  - 3–7 are letters or numbers
- Decimal point between third and fourth characters
  - But not in HCUP fields
- Inexact mapping between ICD-9-CM and ICD-10-CM
  - One ICD-9 $\longrightarrow$ many ICD-10
  - Many ICD-9 $\longrightarrow$ one ICD-10

ICD-10-PCS

- Over 72,000 codes at time of transition from ICD-9-CM (less than 4,000 codes)
- Always 7 characters
  - Z placeholder for when a code contains fewer than 7 characters
- All characters can be letters or numbers
  - First character is section of code book
  - Second character is body system
  - Third character is root operation
  - Fourth character is body part
  - Fifth character is approach
  - Sixth character is device
  - Seventh character is qualifier
- Similar problems with inexact matching between coding systems

Identifying codes in the SID

- Goal is to determine whether a particular diagnosis or procedure was coded during an admission
- Do any of the codes recorded in `I10_DX`n / `I10_ECAUSE`n or `I10_PR`n match any of the codes for our diagnosis or procedure of interest?

Identifying codes in the SID

- Suppose we're interested in acute myocardial infarction
- I21.0–I21.4, I22.0–I22.2, I22.8, I22.9

```
dx10_acutemi <- c(
  "I2101", "I2102", "I2109", "I2111", "I2119", "I2121", "I2129",
  "I213", "I214", "I220", "I221", "I222", "I228", "I229"
)
```

Identifying codes in the SID

- Start with single admission, single code

```
my_data <- data.frame(KEY = 1, DX1 = "I220")
```

- One way to compare values is with ==

```
my_data %>%
  mutate(
    dx_acutemi = ifelse(DX1 == "I2101" | DX1 == "I2102" | <...>, 1, 0)
  )
```

- More efficient way is with %in%

```
my_data %>%
  mutate(
    dx_acutemi = ifelse(DX1 %in% dx10_acutemi, 1, 0)
  )
```

```
##   KEY  DX1 dx_acutemi
## 1   1 I220          1
```

Identifying codes in the SID

- What if we have multiple admissions each with a single code?

```
my_data <- data.frame(
  KEY = 1:2,
  DX1 = c("I220", "E0800")
)

my_data %>%
  mutate(
    dx_acutemi = ifelse(DX1 %in% dx10_acutemi, 1, 0)
  )
```

```
##   KEY   DX1 dx_acutemi
## 1   1  I220          1
## 2   2 E0800          0
```

Identifying codes in the SID

- We have multiple columns of diagnosis and procedure codes for each admission

```
my_data <- data.frame(
  KEY = 1:3,
  DX1 = c("I220", "I209", "J80"),
  DX2 = c("E0800", "I220", "O9921")
)

my_data %>%
  mutate(
    dx_acutemi = ifelse(
      DX1 %in% dx10_acutemi | DX2 %in% dx10_acutemi, 1, 0)
  )
```

```
##   KEY  DX1   DX2 dx_acutemi
## 1   1 I220 E0800          1
## 2   2 I209  I220          1
## 3   3  J80 O9921          0
```

```
my_data
```

Identifying codes in the SID

- So real problem is that we have LOTS of columns with diagnosis and procedure codes and likely many diagnoses and procedures that we'd like to identify

```
core1p2 <- core1p %>%
  mutate(
    dx_acutemi = case_when(
      I10_DX_Admitting %in% dx10_acutemi ~ 1,
      I10_DX1 %in% dx10_acutemi ~ 1,
      I10_DX2 %in% dx10_acutemi ~ 1,
#       ...etc...
      I10_DX33 %in% dx10_acutemi ~ 1,
      I10_DX34 %in% dx10_acutemi ~ 1,
      I10_ECAUSE1 %in% dx10_acutemi ~ 1,
      I10_ECAUSE2 %in% dx10_acutemi ~ 1,
      I10_ECAUSE3 %in% dx10_acutemi ~ 1,
      I10_ECAUSE4 %in% dx10_acutemi ~ 1,
      I10_ECAUSE5 %in% dx10_acutemi ~ 1,
      I10_ECAUSE6 %in% dx10_acutemi ~ 1,
      TRUE ~ 0
    )
  )
```

Identifying codes in the SID

- Things are much easier if we're only concerned with 'primary' (i.e., in `I10_DX1` only) diagnoses of our condition of interest

```
core1p2 <- core1p %>%
  mutate(
    dx1_acutemi = ifelse(I10_DX1 %in% dx10_acutemi, 1, 0)
  )
```

Identifying codes in the SID

- Identifying procedures works in a similar way

```
pr10_cabg <- c(
  "0210083", "0210088", "0210089", "021008C", "021008F", "021008W",
  "0210093", "0210098", "0210099", "021009C", "021009F", "021009W",
  "02100A3", "02100A8", "02100A9", "02100AC", "02100AF", "02100AW",
# ...etc...
)

core1p2 <- core1p %>%
  mutate(
    pr_cabg = case_when(
      I10_PR1 %in% pr10_cabg ~ 1,
      I10_PR2 %in% pr10_cabg ~ 1,
#     ...etc...
      I10_PR30 %in% pr10_cabg ~ 1,
      I10_PR31 %in% pr10_cabg ~ 1,
      TRUE ~ 0
    )
  )
```

- No special significance to the code in `I10_PR1`

Options for Programming

- Lengthy, but relatively easy to copy and modify
  - Create a vector of codes like `dx10_acutemi`
  - Must list the codes that actually appear on records

```
# Bad
dx10_major_depress <- "F33"
# Good
dx10_major_depress <- c(
  "F330", "F331" "F332", "F333", "F3340", "F3341", "F3342",
  "F338", "F339"
)
```

- Use code file associated with this lecture as template

Options for Programming

- Lengthy, but relatively easy to copy and modify

    1. Create a vector of codes like `dx10_acutemi`

    2. Use code file associated with this lecture as template

```
core1p_with_major_depress <- core1p %>%
  mutate(
    dx_major_depress = case_when(
      I10_DX_Admitting %in% dx10_major_depress ~ 1,
      I10_DX1 %in% dx10_major_depress ~ 1,
#     ...etc...
      )
  )
```

Options for Programming

- Functions defined that uses the same code
    - Saved in **admin_course_data/code2023/**
- Read definitions from external R code file with `source`

```
source(
  paste0(
    "//storage1.ris.wustl.edu/colditzg/", # <- different for Mac users
    "Active/admin_course_data/code2023/coder.R"
  )
)
```

Options for Programming

- All functions use the same first three arguments
    1. Name of input data set
    2. Name of variable with code list
    3. Name of new variable that will be made
- Return the input data set with the new column added
- Work similar to `mutate`, so each named `mutate_*`

Options for Programming

- `mutate_flag_dx` for creating 0/1 variables for diagnoses

```
core1p_with_acutemi <- mutate_flag_dx(core1p, dx10_acutemi, dx_acutemi)
```

- is equivalent to

```
core1p_with_acutemi <- mutate(
  core1p,
  dx_acutemi = case_when(
    I10_DX_Admitting %in% dx10_acutemi ~ 1,
    I10_DX1 %in% dx10_acutemi ~ 1,
    I10_DX2 %in% dx10_acutemi ~ 1,
#   ...etc...
    I10_ECAUSE5 %in% dx10_acutemi ~ 1,
    I10_ECAUSE6 %in% dx10_acutemi ~ 1,
    TRUE ~ 0
  )
)
```

Options for Programming

- `mutate_flag_dx` for creating 0/1 variables for diagnoses

```
core1p_with_acutemi <- mutate_flag_dx(core1p, dx10_acutemi, dx_acutemi)
```

- Can also be written using `%>%`

```
core1p_with_acutemi <- core1p %>%
  mutate_flag_dx(dx10_acutemi, dx_acutemi)
```

- Can define variables for multiple diagnoses/procedures in same block of code

```
core1p2 <- core1p %>%
  mutate_flag_dx(dx10_acutemi, dx_acutemi) %>%
  mutate_flag_dx(dx10_istroke, dx_istroke)
```

Options for Programming

- `mutate_flag_dx1` for creating 0/1 variables for primary diagnoses

```
core1p_with_dx1_acutemi <- core1p %>%
  mutate_flag_dx1(dx10_acutemi, dx1_acutemi)
```

- `mutate_flag_pr` for creating 0/1 variables for procedures

```
core1p_with_cabg <- core1p %>%
  mutate_flag_pr(pr10_cabg, pr_cabg)
```

- Other functions will be shown in future coding demos

References

- HCUP introduction to ICD-10 coding systems https://www.hcup-us.ahrq.gov/datainnovations/BriefIntrotoICD-10Codes041117.pdf
- Centers for Medicare and Medicaid Services General Equivalence Mappings (for translating ICD-9 to ICD-10) https://www.cms.gov/Medicare/Coding/ICD10/2018-ICD-10-CM-and-GEMs