BDSI R workshop – Data wrangling

Robert Cope & Terry Neeman

2022

We begin with the cancer data.

library(tidyverse)

cancer\_df <- read\_csv('data/Cuzick\_2010\_breast\_cancer\_density.csv') %>%  
 mutate(case\_f = factor(case, labels = c("Control","Case")),  
 ARM\_f = factor(ARM, labels = c("Placebo", "Tamoxifen")))

str(cancer\_df)

# Data manipulation

An important skill is to be able to extract bits of your data that you are interested in, calculate summary statistics, etc. We will cover a few key functions that help do this in R (they come from the package dplyr which is part of the tidyverse). Chapter 5 of R for data science <https://r4ds.had.co.nz> includes information on these functions.

## filter()

The first thing you might want to do with data is filter it. The function filter does this. Its first argument is the data, and further arguments are what you want to filter by. For example, if we want all the records of patients receiving Tamoxifen, we write:

filter(cancer\_df, ARM\_f == 'Tamoxifen')

### The pipe %>%

Within the tidyverse there is a convenient way to combine these functions, called the pipe (denoted by %>%). A pipe gives the thing on the left to the function on the right, as its first argument. For example:

cancer\_df %>%  
filter(ARM\_f == 'Tamoxifen')

Note that you need == if you are doing a comparison. In R, == asks if two things are equal, whereas = does assignment (and will create an error in filter()).

You can filter by multiple things.

cancer\_df %>%  
 filter(ARM\_f == 'Tamoxifen',density == 90)

You can assign the output of your filter to a variable.

tam\_90 <- cancer\_df %>% filter(ARM\_f == 'Tamoxifen',density == 90)  
tam\_90$case\_f

### Exercise 1 – get the Case records

cancer\_df %>%  
 filter(case\_f == 'Case')

### Exercise 2 – get the records of patients at least 70 years old

cancer\_df %>%  
 filter(AGE >= 70)

We can use more complex expressions. For example, to get those records with density 50, 55 or 60 (note that density here is only measured in intervals of 5):

cancer\_df %>% filter(density == 50 | density == 55 | density == 60)

or

cancer\_df %>% filter(density %in% c(50,55,60))

An exclamation mark ! means “not”. We can use this to do the opposite, i.e., get those records with any density other than 50, 55 or 60:

cancer\_df %>%   
 filter(!(density %in% c(50,55,60)))

## arrange()

It can be useful to sort things. You can do this using arrange().

cancer\_df %>% arrange(density)

To reverse the order, use desc()

cancer\_df %>% arrange(desc(density))

### Exercise 3 – arrange by Age (descending)

cancer\_df %>%   
 arrange(desc(AGE))

### Exercise 3b – arrange by density and then Age (descending)

cancer\_df %>%  
 arrange(desc(density),desc(AGE))

## select()

Selecting particular columns isn’t too important given how small these data is, but you can still do it, using select():

cancer\_df %>%select(AGE, density)

(it also lets you rename variables)

This is equivalent to:

cancer\_df[,c('AGE', 'density')]

This also lets you reorder variables:

## mutate()

Mutate lets you transform the data and create a new variable with the result. Note that it doesn’t actually change the data, unless you tell it to.

cancer\_df %>%mutate(density\_prop = density / 100)

Tip: ?mutate has a list of helpful mutate functions.

### The pipe %>% (part 2)

The pipe lets you chain commands together!

cancer\_df %>%  
 mutate(density\_prop = density / 100) %>%  
 select(ARM\_f, case\_f, AGE, BMI, density\_prop) %>%  
 filter(AGE < 40) %>%  
 arrange(AGE, BMI)

You should almost always do these things iteratively – you don’t need to jump straight to the answer!!

### Exercise 4 – extract the records of of patients over age 50 that had density <= 30 and had cancer, sorted descending by age

cancer\_df %>%  
 filter(case\_f == 'Case', AGE >=50, density <=30) %>%  
 arrange(desc(AGE))

## group\_by() and summarise()

Often you wish to aggregate your data. To do this, we can combine the pipe with the group\_by() function, and summarise() (or filter, mutate, etc.)

cancer\_df %>%   
 group\_by(case\_f) %>%  
 summarise(num\_cases = n())

Hint: ?summarise also has a helpful list of functions

### Exercise 5 — how many subjects for each treatment (ARM)?

cancer\_df %>%   
 group\_by(ARM\_f) %>%  
 summarise(num\_cases = n())

### Exercise 6 — how many records were there of each combination of ARM and case

cancer\_df %>%   
 group\_by(ARM\_f, case\_f) %>%  
 summarise(num\_cases = n())

### Exercise 7 — what was the mean age and bmi of individuals with and without cancer?

cancer\_df %>%  
 group\_by(case\_f) %>%  
 summarise(mean\_age = mean(AGE), mean\_bmi = mean(BMI, na.rm = TRUE))

(we had to do na.rm here because otherwise you get NAs)

You can chain together many such functions. It helps to do this iteratively. You can also use mutate instead of summarise.

For example, what proportion of people with density at least 80 who received the placebo ended up getting cancer:

cancer\_df %>%  
 filter(density >= 80, ARM\_f == 'Placebo') %>%  
 group\_by(case\_f) %>%  
 summarise(count = n()) %>%   
 mutate(proportion = count / sum(count))

### Exercise 8. Which density values had at least 80 records?

cancer\_df %>%  
 group\_by(density) %>%  
 summarise(count = n()) %>%  
 filter(count >= 80)

### Exercise 9. Summarise breast density values (Mean and SD) for each ARM (treatment) and Status(Case/Control). Include the number of subjects for each subgroup.

cancer\_df %>%  
 group\_by(ARM\_f, case\_f) %>%  
 summarise(count = n(),   
 mean\_density = mean(density) %>% round(1),   
 sd = sd(density) %>% round(2))

### Exercise 10. Create age groups - 40 or under, 41 - 60, and over 60 using the cut() function. Then for each age group, compute the number and proportion of cases.

cancer\_df %>%  
 mutate(AGE\_group = cut(AGE, breaks = c(0,40,60,200),  
 labels = c("40 or under","41 - 60","over 60"))) %>%  
 group\_by(AGE\_group) %>%  
 summarise(total=n(),  
 cases = sum(case),   
 percent = round(cases/total\*100) )