Homework 2

Dakota Thompson

2/19/2021

# Exercise 4 on p. 124

## Practice referring to nonsyntactic names in the following data frame by:

annoying <- tibble(   
 `1` = 1:10,  
 `2` = `1` \* 2 + rnorm(length(`1`))  
 )  
annoying

## # A tibble: 10 x 2  
## `1` `2`  
## <int> <dbl>  
## 1 1 1.19  
## 2 2 4.48  
## 3 3 5.23  
## 4 4 7.49  
## 5 5 10.1   
## 6 6 11.0   
## 7 7 13.7   
## 8 8 16.1   
## 9 9 18.3   
## 10 10 21.4

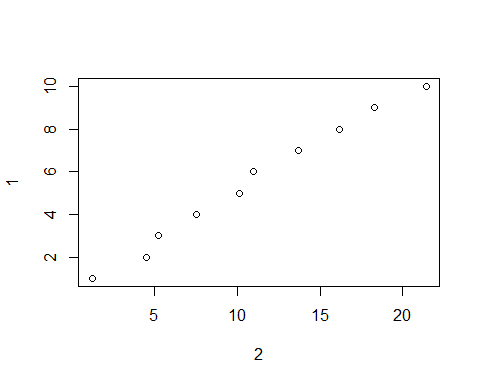
### a. Extracting the variable called 1

a <- annoying$`1`  
a

## [1] 1 2 3 4 5 6 7 8 9 10

### b. Plotting a scatterplot of 1 versus 2.

b <- plot(`1` ~ `2`, annoying)



b

## NULL

### c. Creating a new column called 3, which is 2 divided by 1.

c <- annoying %>%  
 mutate(`3` = `2`/`1`)  
c

## # A tibble: 10 x 3  
## `1` `2` `3`  
## <int> <dbl> <dbl>  
## 1 1 1.19 1.19  
## 2 2 4.48 2.24  
## 3 3 5.23 1.74  
## 4 4 7.49 1.87  
## 5 5 10.1 2.02  
## 6 6 11.0 1.83  
## 7 7 13.7 1.95  
## 8 8 16.1 2.02  
## 9 9 18.3 2.03  
## 10 10 21.4 2.14

### d. Renaming the columns to one, two, and three

d <- c %>%  
 rename("one" = `1`, "two" = `2`, "three" = `3`)  
d

## # A tibble: 10 x 3  
## one two three  
## <int> <dbl> <dbl>  
## 1 1 1.19 1.19  
## 2 2 4.48 2.24  
## 3 3 5.23 1.74  
## 4 4 7.49 1.87  
## 5 5 10.1 2.02  
## 6 6 11.0 1.83  
## 7 7 13.7 1.95  
## 8 8 16.1 2.02  
## 9 9 18.3 2.03  
## 10 10 21.4 2.14

# Exercise 5 on p. 124

## What does tibble::enframe() do?

It takes a vector and turns it into a tibble data frame with names and values.

## When might you use it?

You might want to use it when you want to do functions on the variables themselves (for example, if one was gender and one was height in a vector, you might want to easily break it out into height by gender (M or F)).

# Exercise 7 on p. 137

## Generate the correct format string to parse each of the following dates and times:

d1 <- “January 1, 2010” d2 <- “2015-Mar-07” d3 <- “06-Jun-2017” d4 <- c(“August 19 (2015)”, “July 1 (2015)”) d5 <- “12/30/14” # Dec 30, 2014 t1 <- “1705” t2 <- “11:15:10.12 PM”

d1 <- parse\_date("January 1, 2010", "%B %d, %Y")  
d2 <- parse\_date("2015-Mar-07", "%Y-%b-%d")  
d3 <- parse\_date("06-Jun-2017", "%d-%b-%Y")  
d4 <- parse\_date(c("August 19 (2015)", "July 1 (2015)"), "%B %d (%Y)")  
d5 <- parse\_date("12/30/14", "%m/%d/%y") # Dec 30, 2014  
t1 <- parse\_time("1705", "%H%M")  
t2 <- parse\_time("11:15:10.12 PM")  
  
d1

## [1] "2010-01-01"

d2

## [1] "2015-03-07"

d3

## [1] "2017-06-06"

d4

## [1] "2015-08-19" "2015-07-01"

d5

## [1] "2014-12-30"

t1

## 17:05:00

t2

## 23:15:10

# Exercise 2 on p. 151

## Compute the rate for table2, and table4a + table4b. You will need to perform four operations:

table2

## # A tibble: 12 x 4  
## country year type count  
## <chr> <int> <chr> <int>  
## 1 Afghanistan 1999 cases 745  
## 2 Afghanistan 1999 population 19987071  
## 3 Afghanistan 2000 cases 2666  
## 4 Afghanistan 2000 population 20595360  
## 5 Brazil 1999 cases 37737  
## 6 Brazil 1999 population 172006362  
## 7 Brazil 2000 cases 80488  
## 8 Brazil 2000 population 174504898  
## 9 China 1999 cases 212258  
## 10 China 1999 population 1272915272  
## 11 China 2000 cases 213766  
## 12 China 2000 population 1280428583

table4a

## # A tibble: 3 x 3  
## country `1999` `2000`  
## \* <chr> <int> <int>  
## 1 Afghanistan 745 2666  
## 2 Brazil 37737 80488  
## 3 China 212258 213766

table4b

## # A tibble: 3 x 3  
## country `1999` `2000`  
## \* <chr> <int> <int>  
## 1 Afghanistan 19987071 20595360  
## 2 Brazil 172006362 174504898  
## 3 China 1272915272 1280428583

### a. Extract the number of TB cases per country per year.

table2cases <- table2 %>%  
 filter(type == "cases") %>%  
 rename("cases" = count) %>%  
 arrange(country, year)  
table2cases

## # A tibble: 6 x 4  
## country year type cases  
## <chr> <int> <chr> <int>  
## 1 Afghanistan 1999 cases 745  
## 2 Afghanistan 2000 cases 2666  
## 3 Brazil 1999 cases 37737  
## 4 Brazil 2000 cases 80488  
## 5 China 1999 cases 212258  
## 6 China 2000 cases 213766

For Table 2, we first take out all the rows that are cases from type and then change the name of the count of the cases to just cases for when we later combine the 2 tables.

### b. Extract the matching population per country per year.

table2pop <- table2 %>%  
 filter(type == "population") %>%  
 rename("population" = count) %>%  
 arrange(country, year)  
table2pop

## # A tibble: 6 x 4  
## country year type population  
## <chr> <int> <chr> <int>  
## 1 Afghanistan 1999 population 19987071  
## 2 Afghanistan 2000 population 20595360  
## 3 Brazil 1999 population 172006362  
## 4 Brazil 2000 population 174504898  
## 5 China 1999 population 1272915272  
## 6 China 2000 population 1280428583

For Table 2, we then take out all the rows that are population from type and then change the name of the count of the population to just population for when we later combine the 2 tables. We also want to make sure to organize the data so rows are matching in both tables.

### c. Divide cases by population, and multiply by 10,000.

table2combine <- tibble(  
 country = table2cases$country,  
 year = table2cases$year,  
 cases = table2cases$cases,  
 population = table2pop$population  
)  
table2combine

## # A tibble: 6 x 4  
## country year cases population  
## <chr> <int> <int> <int>  
## 1 Afghanistan 1999 745 19987071  
## 2 Afghanistan 2000 2666 20595360  
## 3 Brazil 1999 37737 172006362  
## 4 Brazil 2000 80488 174504898  
## 5 China 1999 212258 1272915272  
## 6 China 2000 213766 1280428583

table2rte <- table2combine %>%  
 mutate(rate = (cases / population) \* 10000)  
table2rte

## # A tibble: 6 x 5  
## country year cases population rate  
## <chr> <int> <int> <int> <dbl>  
## 1 Afghanistan 1999 745 19987071 0.373  
## 2 Afghanistan 2000 2666 20595360 1.29   
## 3 Brazil 1999 37737 172006362 2.19   
## 4 Brazil 2000 80488 174504898 4.61   
## 5 China 1999 212258 1272915272 1.67   
## 6 China 2000 213766 1280428583 1.67

We then finally combine the 2 tables to have the matching rows. After that, we mutate the table to add on the rate of the cases per population per year.

### d. Store back in the appropriate place.

table2rte <- table2rte %>%  
 mutate(type = "rate") %>%  
 rename(count = rate)  
table2rte

## # A tibble: 6 x 6  
## country year cases population count type   
## <chr> <int> <int> <int> <dbl> <chr>  
## 1 Afghanistan 1999 745 19987071 0.373 rate   
## 2 Afghanistan 2000 2666 20595360 1.29 rate   
## 3 Brazil 1999 37737 172006362 2.19 rate   
## 4 Brazil 2000 80488 174504898 4.61 rate   
## 5 China 1999 212258 1272915272 1.67 rate   
## 6 China 2000 213766 1280428583 1.67 rate

table2final <- bind\_rows(table2, table2rte) %>%  
 arrange(country, year, type, count)  
  
table2final <- table2final %>%  
 mutate(cases=NULL, population=NULL)  
table2final

## # A tibble: 18 x 4  
## country year type count  
## <chr> <int> <chr> <dbl>  
## 1 Afghanistan 1999 cases 7.45e+2  
## 2 Afghanistan 1999 population 2.00e+7  
## 3 Afghanistan 1999 rate 3.73e-1  
## 4 Afghanistan 2000 cases 2.67e+3  
## 5 Afghanistan 2000 population 2.06e+7  
## 6 Afghanistan 2000 rate 1.29e+0  
## 7 Brazil 1999 cases 3.77e+4  
## 8 Brazil 1999 population 1.72e+8  
## 9 Brazil 1999 rate 2.19e+0  
## 10 Brazil 2000 cases 8.05e+4  
## 11 Brazil 2000 population 1.75e+8  
## 12 Brazil 2000 rate 4.61e+0  
## 13 China 1999 cases 2.12e+5  
## 14 China 1999 population 1.27e+9  
## 15 China 1999 rate 1.67e+0  
## 16 China 2000 cases 2.14e+5  
## 17 China 2000 population 1.28e+9  
## 18 China 2000 rate 1.67e+0

Finally, we have to put it back in the original data frame. To do that we change rate to math the original tables conventions with type and count, then stick the tables together.

table4together <- tibble(  
 country = table4a$country,  
 cases99 = table4a$`1999`,  
 pop99 = table4b$`1999`,  
 rate99 = table4a$`1999` / table4b$`1999` \* 10000,  
 cases00 = table4a$`2000`,  
 pop00 = table4b$`2000`,  
 rate00 = table4a$`2000` / table4b$`2000` \* 10000)  
table4together

## # A tibble: 3 x 7  
## country cases99 pop99 rate99 cases00 pop00 rate00  
## <chr> <int> <int> <dbl> <int> <int> <dbl>  
## 1 Afghanistan 745 19987071 0.373 2666 20595360 1.29  
## 2 Brazil 37737 172006362 2.19 80488 174504898 4.61  
## 3 China 212258 1272915272 1.67 213766 1280428583 1.67

For the 4a and 4b tables, we can easily do everything in one fell swoop. We just take each variable from each table and combine them into one tibble.

## Which representation is easiest to work with? Which is hardest? Why?

I don’t think either is particularly fun to work with, however I do think in the long run table 2 will be easier to use. While 4a&b was much faster to fix to figure out the rate, the years are hard to get out of the table having been attached to the cases and population. On the other hand, 2 took much more work, but every single variable is in its own column and will be easier to group in different ways later.