Date:19th Oct, 2020

**Assignment 1: Tidy with Tidyr**

Raj Kumar Seth (UMDS20006)

Q.1) Using prose, describe how the variables and observations are organized in each of the sample tables.

Ans:

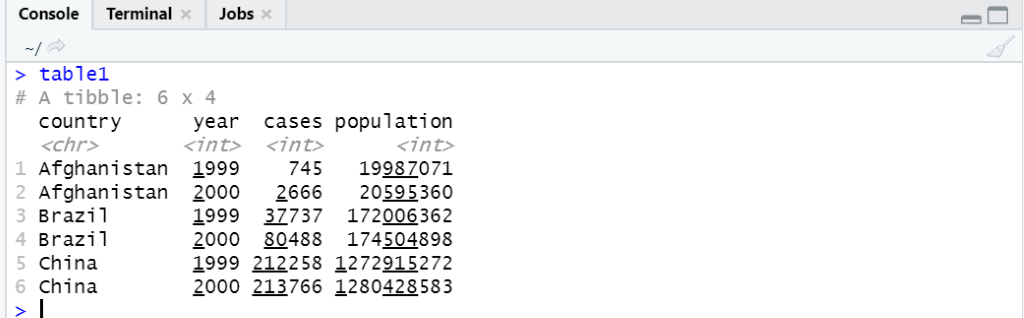
Table 1: each observation has country, year, cases, population its own row and each variable has its own column. 

Table2: each row represents the country, year, and the variable type of either case or population. The variable count represents the unique value for the variable type.

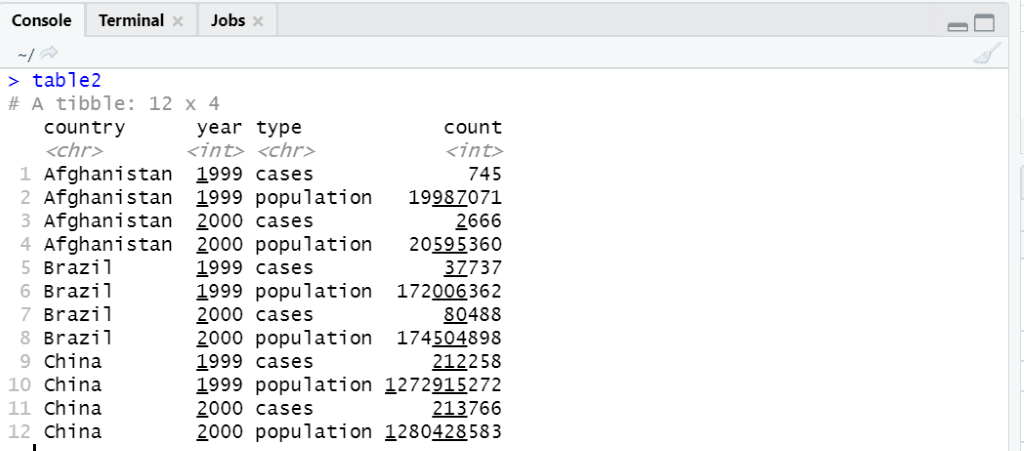


Table3: the variables case and count are mutated into a new variable rate.

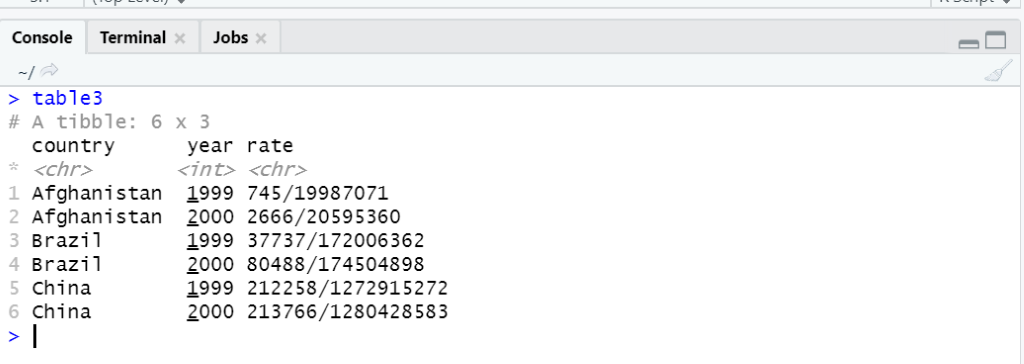
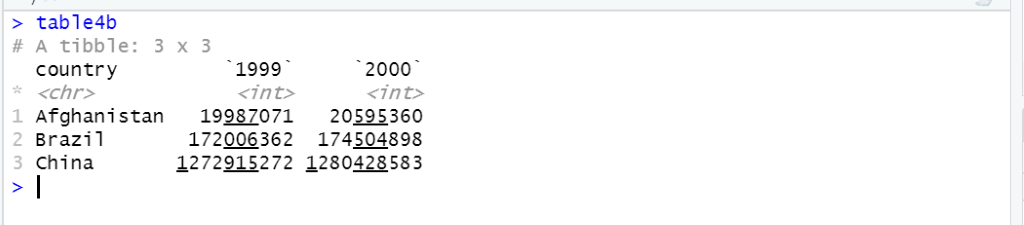
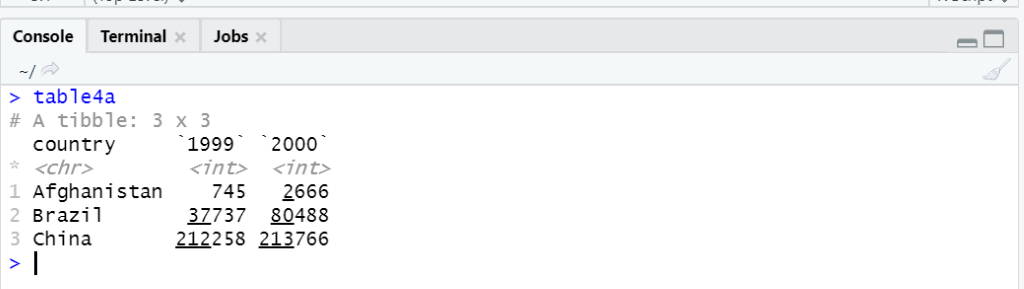


Table4a & table 4b: cases and population are represented in separate tables. Years 1999 and 2000 are treated as different variables.





Q.2) Compute the rate for table2, and table4a + table4b. You will need to perform four operations:

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

b. Extract the matching population per country per year.

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

d. Store back in the appropriate place.

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

Why?

Ans:

country\_list <- filter(table2, type == 'cases')$country

country\_list

years <- filter(table2, type == 'cases')$year

years

case\_list <- filter(table2, type == 'cases')$count

case\_list

population\_list <- filter(table2, type == 'population')$count

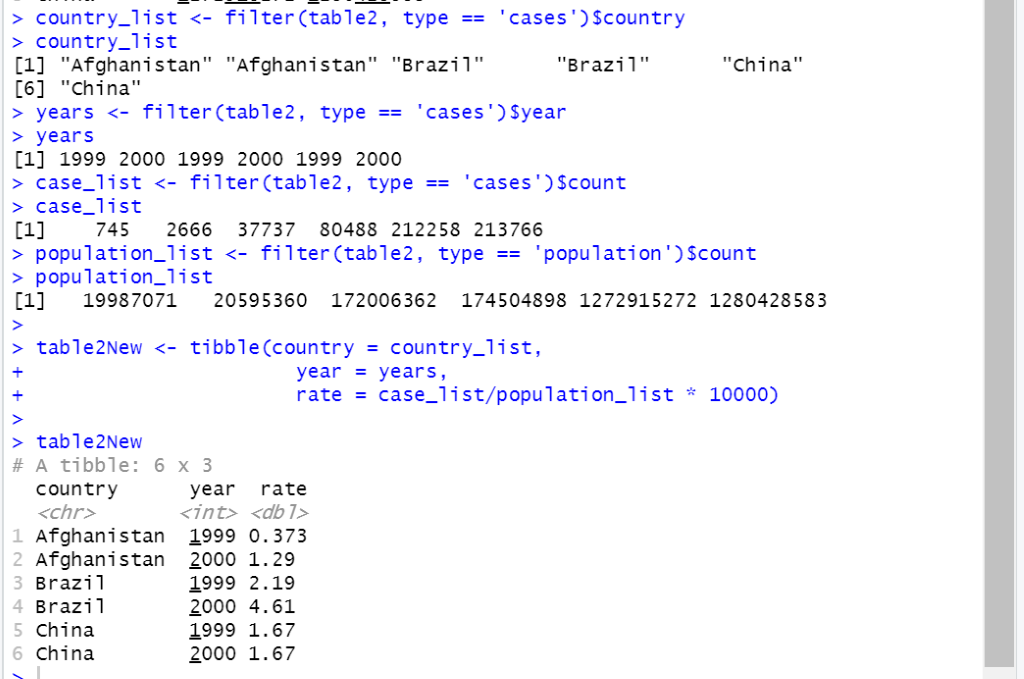
population\_list

table2New <- tibble(country = country\_list,

year = years,

rate = case\_list/population\_list \* 10000)

table2New

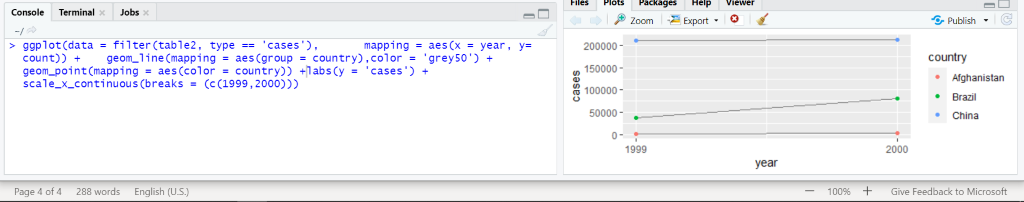


* table2 is much easier to work with, and involves fewer intermediate steps.

Q.3) Re-create the plot showing change in cases over time using table instead of table1. What do you need to do first?

Ans:

ggplot(data = filter (table2, type == 'cases'), mapping = aes(x = year, y= count)) +geom\_line(mapping = aes(group = country), color = 'grey50') + geom\_point(mapping = aes(color = country)) + labs (y = 'cases') + scale\_x\_continuous(breaks = (c (1999,2000)))



Q.4. Why are gather() and spread() not perfectly symmetrical?

Carefully consider the following example:

stocks <- tibble(

year = c(2015, 2015, 2016, 2016),half = c( 1, 2, 1, 2),return = c(1.88, 0.59, 0.92, 0.17))

stocks %>%

spread(year, return) %>%

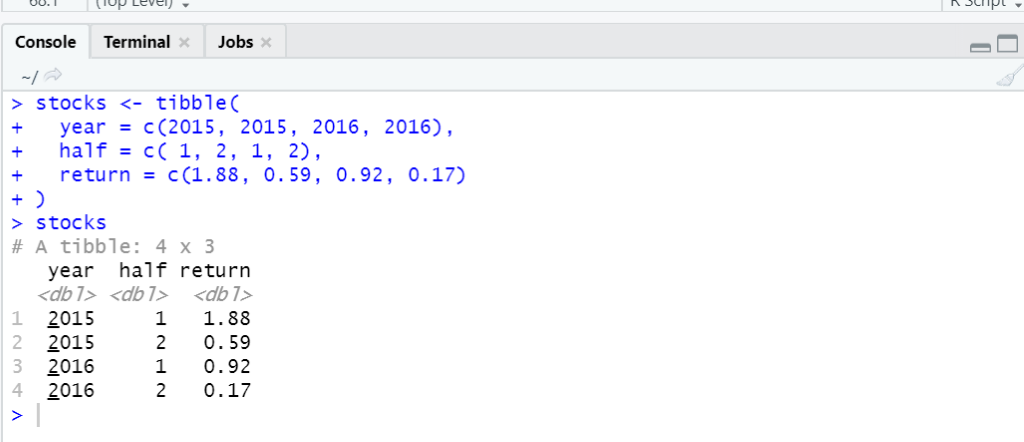
gather("year", "return", `2015`:`2016`)

(Hint: look at the variable types and think about column names.)

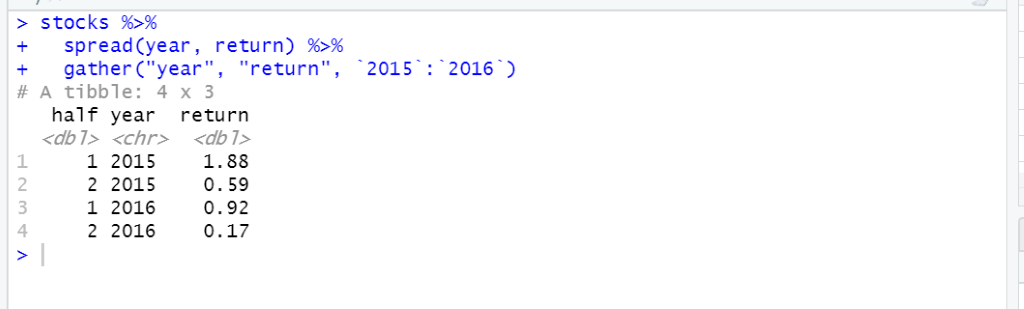
Both spread() and gather() have a convert argument. What does it do?

Ans:

Stocks table from the given code:



2nd Part of the given code:



The data type for the year was converted from dbl to chr. In the intermediate step, spread(), 2015 and 2016 became the names of the variables. So when using gather(), 2015 and 2016 were naturally treated as strings, and the variable type for year became chr.

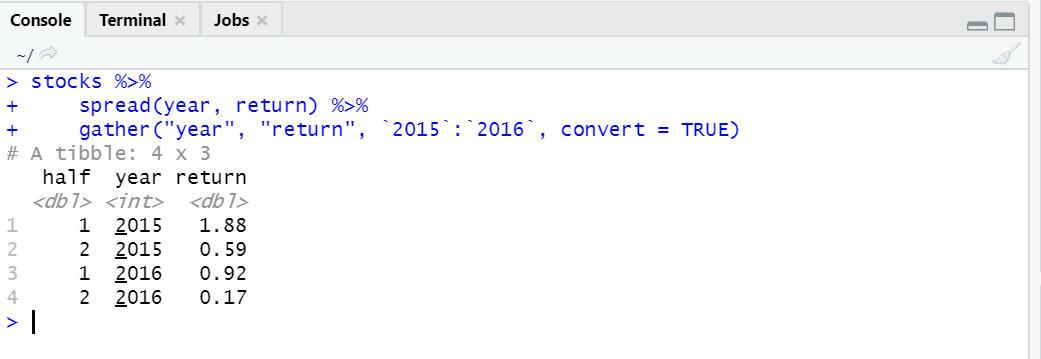
* To override this behavior, we can add convert = TRUE in gather(). It is useful if the column names are actually numeric, integer, or logical.

Code:

stocks %>%

spread(year, return) %>%

gather("year", "return", `2015`:`2016`, convert = TRUE)



Q.5)Why does this code fail?

table4a %>%

gather(1999, 2000, key = "year", value = "cases")

#> Error in combine\_vars(vars, ind\_list): Position must be between 0 and n

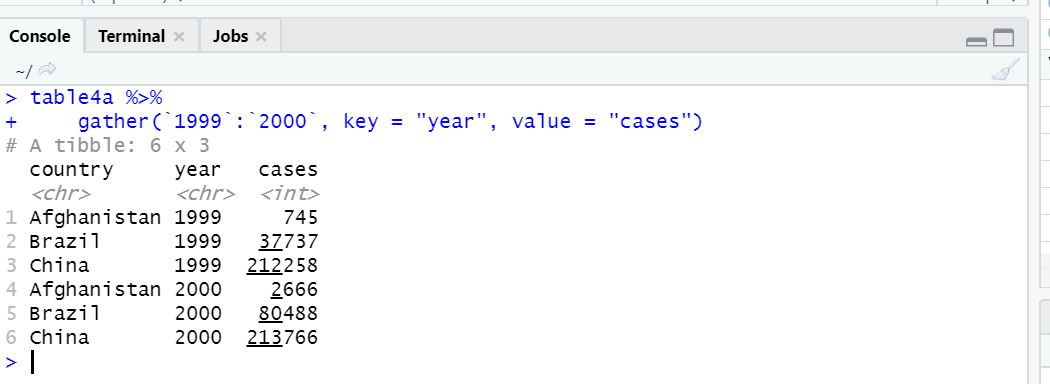
Ans:

First, we need to add backticks around 1999 and 2000 since the variable names are numeric and hence non-syntactic. Second, we need to use : to indicate we want to select all columns from 1999 to 2000 inclusive.

Rectified Code:

table4a %>%

gather(`1999`:`2000`, key = "year", value = "cases")



Q.6) Why does spreading this tibble fail? How could you add a new column to fix the problem?

people <- tribble(

~name, ~key, ~value,

#-----------------|--------|------

"Phillip Woods", "age", 45,

"Phillip Woods", "height", 186,

"Phillip Woods", "age", 50,

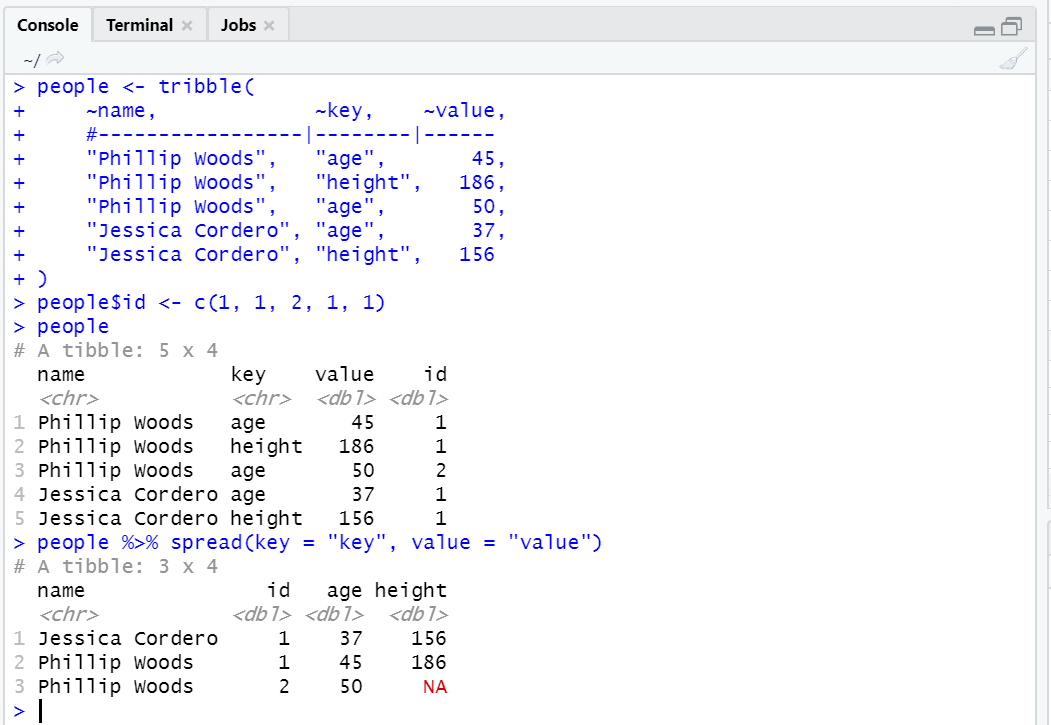
"Jessica Cordero", "age", 37,

"Jessica Cordero", "height", 156

)

Ans:

Spreading this tibble will fail because there are duplicated rows, i.e., there are two rows of “Phillip Woods” with key “age”. In other words, a single cell cannot be both 45 and 50 at the same time. We can add a new column to make those rows to be not duplicates.



Q.7) Tidy the simple tibble below. Do you need to spread or gather it? What are the variables?

preg <- tribble(

~pregnant, ~male, ~female,

"yes", NA, 10,

"no", 20, 12

)

Ans:

pregnant female of some value 10 2 .

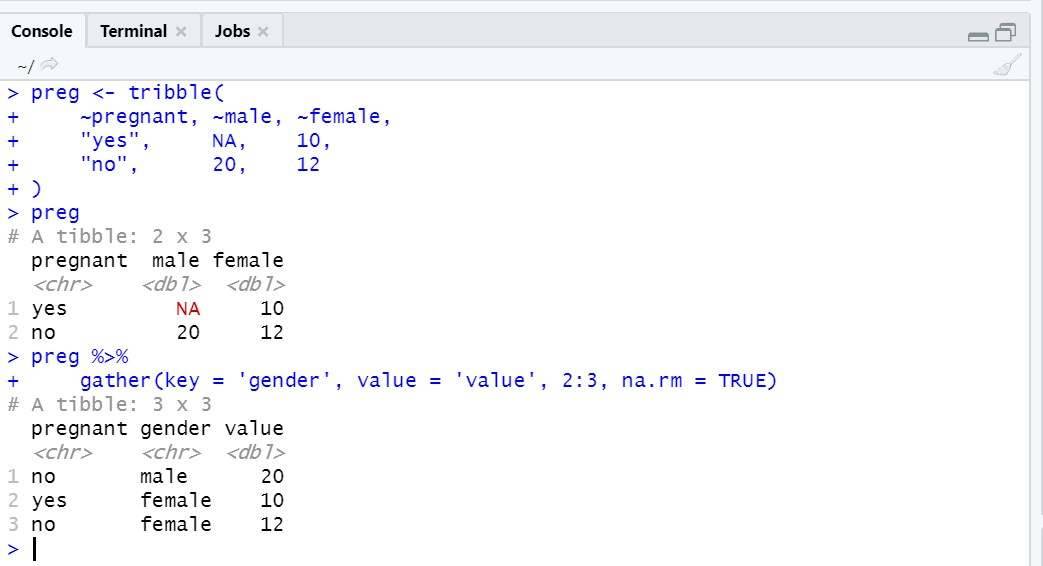
Non-pregnant male of some value 20

Non-pregnant female of some value 12

We can use gather(), and add na.rm = TRUE to remove the NA observation.

Code:

preg %>% gather(key = 'gender', value = 'value', 2:3, na.rm = TRUE)



Q.8) What do the extra and fill arguments do in separate()?

Experiment with the various options for the following two toy datasets.:

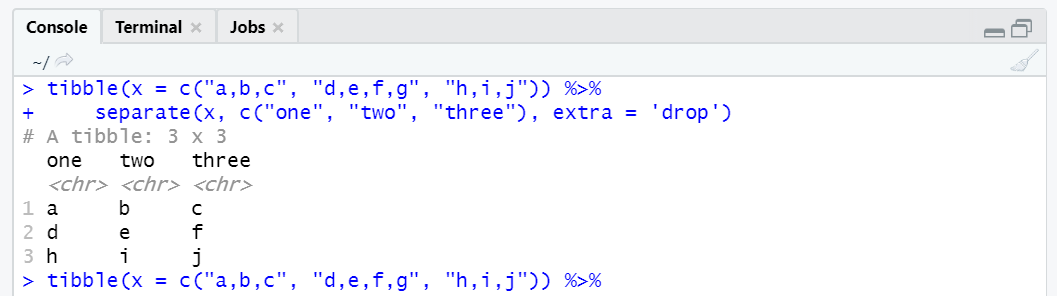
tibble(x = c("a,b,c", "d,e,f,g", "h,i,j")) %>%

separate(x, c("one", "two", "three"), extra = 'drop')

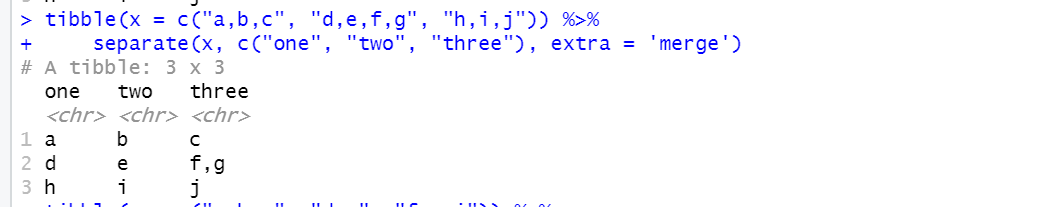
Ans:

* extra controls what happens when the separated pieces are more than the number of variables defined in the dataset.
* The default option is warn, which shows a warning and drops extra pieces.
* The option drop drops extra pieces without giving a warning.

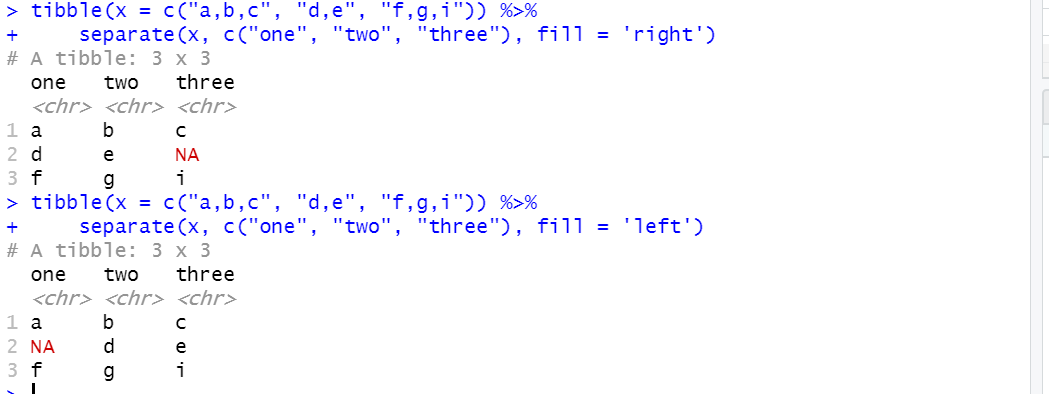
Code:



The option merge will only split at most length in time.



It controls what happens if the separated pieces are less than the number of variables defined into. By default, it fills with NA on the right and gives a warning. The option right fills with NAs on the right without a warning.



Q.9)Both unite() and separate() have a remove argument. What does it do?

Why would you set it to FALSE?

Ans:

The remove argument is set to TRUE by default.

It removes input columns from the output data frame.

If set to FALSE, the original separate column, or the united columns, are retained in the output.

Example:

table5 %>% unite(new, century, year, sep = "")

table5 %>% unite(new, century, year, sep = "", remove = FALSE)

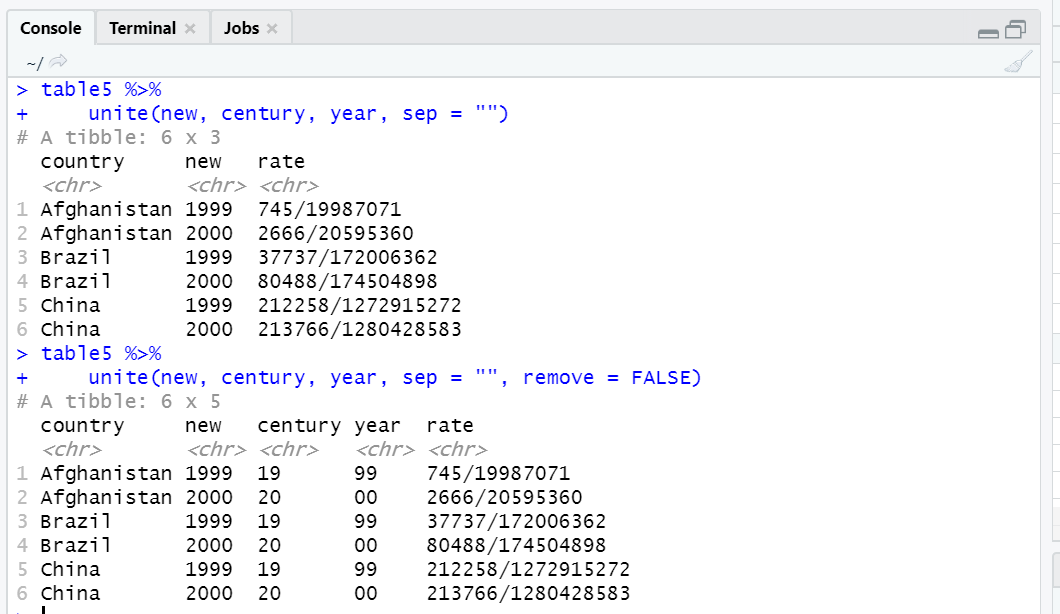
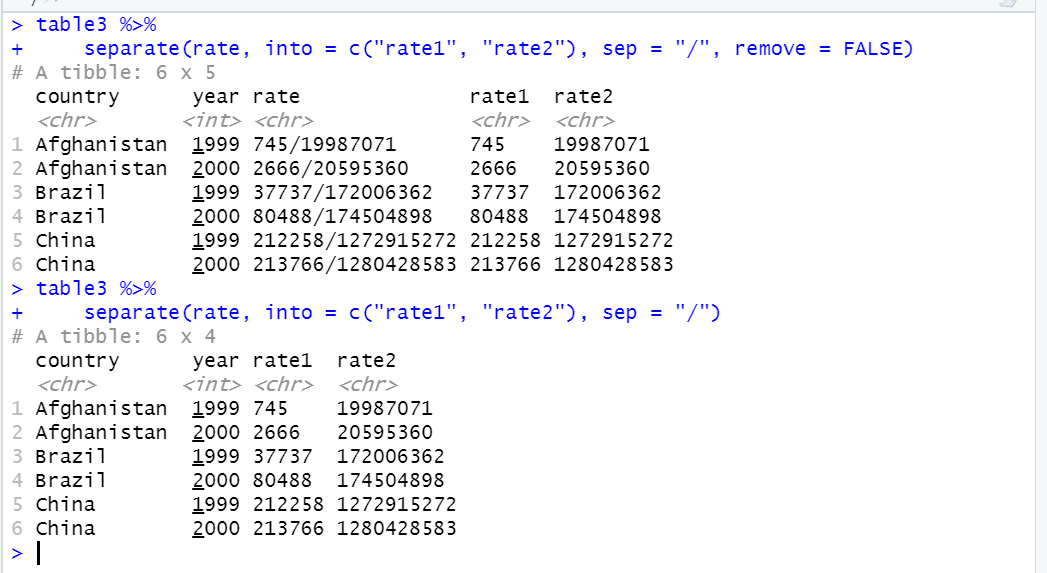


table3 %>% separate(rate, into = c("rate1", "rate2"), sep = "/", remove = FALSE)

table3 %>% separate(rate, into = c("rate1", "rate2"), sep = "/", remove = FALSE)



Q.10)Compare and contrast separate() and extract(). Why are there three variations of separation (by position, by separator, and with groups), but only one unit?

Ans:

extract() uses regular expressions to capture groups and turn groups into multiple columns.

There are many ways to separate a column into multiple columns. In contrast, there is only one way to put together multiple columns into a single column.

Q.11)Compare and contrast the fill arguments to spread() and complete().

Ans:

In spread(), all NAs will be replaced by the fill value. The fill argument only takes in one value.

In complete(), NAs under different variables can be replaced by different values.

The fill argument takes in a list that specifies the values to replace NA for different variables.

Q.12)What does the direction argument to fill() do?

Ans:

The default value is down. Any NAs will be replaced by the previous non-missing value. The filling direction can be reversed if direction is set to up.

Example:

df <- data.frame(Month = 1:6, Year = c(2000, rep(NA, 5)))

df %>% fill(Year)

