r4ds Ex 12

MW

2019/06/25

12.2

1

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

```
?talbe1
```

'table1', 'table2', 'table3', 'table4a', 'table4b', and 'table5' all display the number of TB cases documented by the World Health Organization in Afghanistan, Brazil, and China between 1999 and 2000. The data contains values associated with four variables (country, year, cases, and population), but each table organizes the values in a different layout. The data is a subset of the data contained in the World Health Organization Global Tuberculosis Report

2

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

```
table2 %>% spread(type, count) %>%
    mutate(cases_per_cap = (cases / population) * 10000) %>%
    gather(key=type, value=count, cases, population, cases_per_cap)
```

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

```
## 14 Afghanistan  2000 cases_per_cap 1.29e+0
## 15 Brazil       1999 cases_per_cap 2.19e+0
## 16 Brazil       2000 cases_per_cap 4.61e+0
## 17 China        1999 cases_per_cap 1.67e+0
## 18 China        2000 cases_per_cap 1.67e+0
```

```
table4a %>% inner_join(table4b, by="country") %>%
    mutate(`1999`=`1999.x`/`1999.y`*10000,
        `2000`=`2000.x`/`2000.y`*10000) %>%
    select(country, `1999`, `2000`)
```

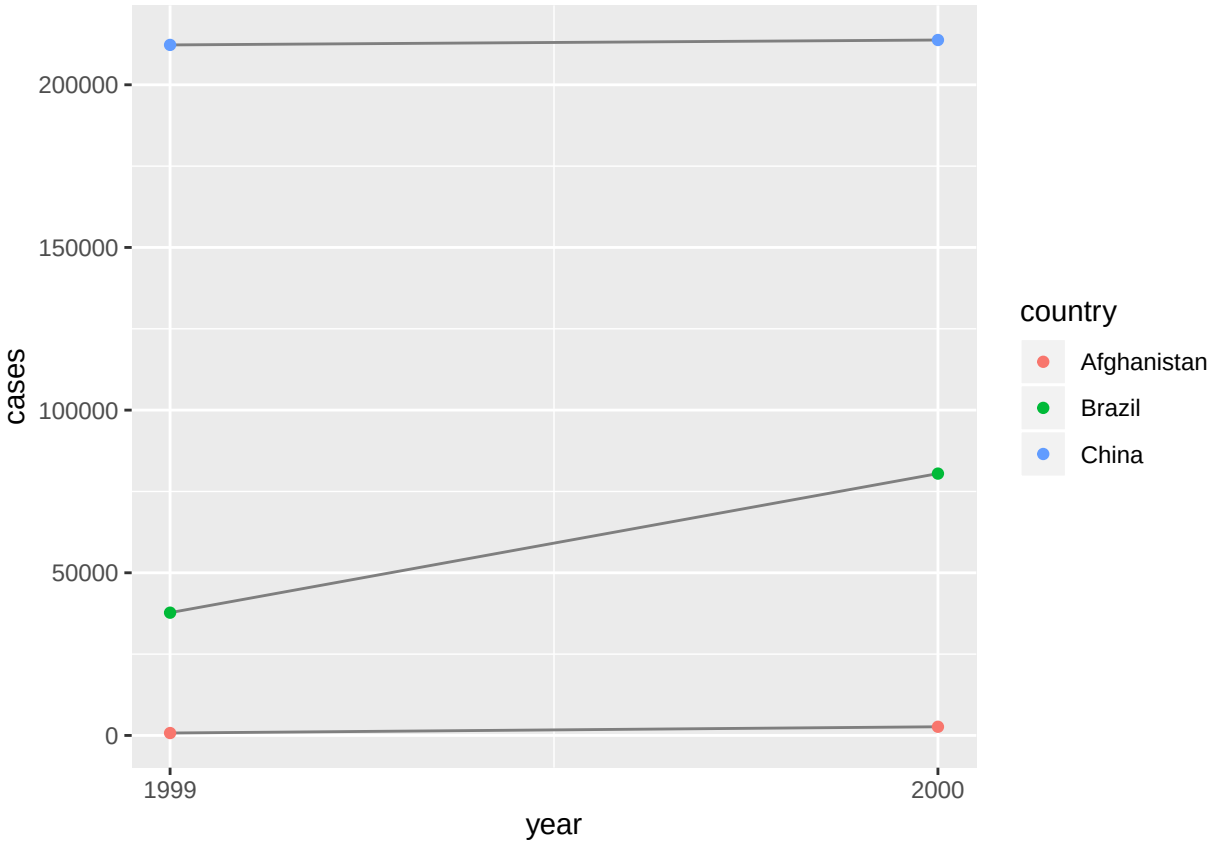
```
## # A tibble: 3 x 3
##   country     `1999` `2000`
##   <chr>        <dbl>  <dbl>
## 1 Afghanistan  0.373   1.29
## 2 Brazil       2.19    4.61
## 3 China        1.67    1.67
```

I think the latter is harder than the former because the latter is separated by default and it makes us be hard to manipulate columns.

3

> Recreate the plot showing change in cases over time using table2 instead of table1. What do you need to do first?

```
table2 %>% filter(type=="cases") %>%
    ggplot(aes(year, count)) +
    geom_line(aes(group=country), colour="grey50") +
    geom_point(aes(colour=country)) +
    scale_x_continuous(breaks=unique(table2$year)) +
    ylab("cases")
```



12.3

1

Why are `gather()` and `spread()` not perfectly symmetrical? Carefully consider the following example:

`gather()` can't specify `double` in `key`, on the other hand `spread` can.

2

Why does this code fail?

1999 -> `1999` 2000 -> `2000`

3

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

Rows containing `Phillip Woods` and `age` are duplicated.

```
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
)

people %>% mutate(id=rep(1:nrow(people))) %>% spread(key=key, value=value)
```

```
## # A tibble: 5 x 4
##   name               id   age height
##   <chr>           <int> <dbl>  <dbl>
## 1 Jessica Cordero     4    37     NA
## 2 Jessica Cordero     5    NA    156
## 3 Phillip Woods       1    45     NA
## 4 Phillip Woods       2    NA    186
## 5 Phillip Woods       3    50     NA
```

4

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
)

preg %>% gather(male, female, key=pregnant, value=sex, na.rm = TRUE)
```

```
## # A tibble: 3 x 2
##   pregnant   sex
##   <chr>    <dbl>
## 1 male        20
## 2 female      10
## 3 female      12
```

12.4.3

1

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"))
```

```
## Warning: Expected 3 pieces. Additional pieces discarded in 1 rows [2].
```

```
## # A tibble: 3 x 3
##   one   two   three
##   <chr> <chr> <chr>
## 1 a     b     c
## 2 d     e     f
## 3 h     i     j
```

```
tibble(x = c("a,b,c", "d,e", "f,g,i")) %>%
  separate(x, c("one", "two", "three"))
```

```
## Warning: Expected 3 pieces. Missing pieces filled with `NA` in 1 rows [2].
```

```
## # A tibble: 3 x 3
##   one   two   three
##   <chr> <chr> <chr>
## 1 a     b     c
## 2 d     e     <NA>
## 3 f     g     i
```

extra: If ‘sep’ is a character vector, this controls what happens when there are too many pieces. There are three valid options: - "warn" (the default): emit a warning and drop extra values. - "drop": drop any extra values without a warning. - "merge": only splits at most ‘length(into)’ times

fill: If ‘sep’ is a character vector, this controls what happens when there are not enough pieces. There are three valid options: - "warn" (the default): emit a warning and fill from the right - "right": fill with missing values on the right - "left": fill with missing values on the left

```
tibble(x = c("a,b,c", "d,e,f,g", "h,i,j")) %>%
    separate(x, c("one", "two", "three"), extra="merge")
```

```
## # A tibble: 3 x 3
##   one   two   three
##   <chr> <chr> <chr>
## 1 a     b     c
## 2 d     e     f,g
## 3 h     i     j
```

```
tibble(x = c("a,b,c", "d,e", "f,g,i")) %>%
    separate(x, c("one", "two", "three"), fill="right")
```

```
## # A tibble: 3 x 3
##   one   two   three
##   <chr> <chr> <chr>
## 1 a     b     c
```

```
## 2 d     e     <NA>
## 3 f     g     i
```

2

Both `unite()` and `separate()` have a remove argument. What does it do? Why would you set it to FALSE?

`remove: If 'TRUE' , remove input columns from output data frame.`

3

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

`separate()` can split columns into multiple columns by separator. On the other hands, `extract()` can't. But `extract()` can use regular expression.

12.5.1

1

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

- `spread()`
- – fill: If set, missing values will be replaced with this value. Note that there are two types of missingness in the input: explicit missing values (i.e. 'NA'), and implicit missings, rows that simply aren't present. Both types of missing value will be replaced by 'fill'.
- `complete()`
- – fill: A named list that for each variable supplies a single value to use instead of 'NA' for missing combinations.

2

What does the direction argument to `fill()` do?

`down` and `up`.

12.6.1

1

In this case study, I set `na.rm = TRUE` just to make it easier to check that we had the correct values. Is this reasonable? Think about how missing values are represented in this dataset. Are

there implicit missing values? What's the difference between an `NA` and zero?

It depends on whether `NA` in this data shows `no data about TB` or `patients don't have TB`

```
who %>% complete(year, country) %>% nrow()
```

```
## [1] 7446
```

```
who %>% nrow()
```

```
## [1] 7240
```

There are implicit rows.

2

What happens if you neglect the `mutate()` step? (`mutate(key = stringr::str_replace(key, "newrel", "new_rel"))`)

```
who %>% gather(key, value, new_sp_m014:newrel_f65, na.rm = TRUE) %>%
    separate(key, c("new", "var", "sexage")) %>%
    select(-new, -iso2, -iso3) %>%
    separate(sexage, c("sex", "age"), sep = 1)
```

```
## Warning: Expected 3 pieces. Missing pieces filled with `NA` in 2580 rows
## [73467, 73468, 73469, 73470, 73471, 73472, 73473, 73474, 73475, 73476,
## 73477, 73478, 73479, 73480, 73481, 73482, 73483, 73484, 73485, 73486, ...].
```

```
## # A tibble: 76,046 x 6
##    country      year var   sex   age   value
##    <chr>       <int> <chr> <chr> <chr> <int>
##  1 Afghanistan  1997 sp    m     014       0
##  2 Afghanistan  1998 sp    m     014      30
##  3 Afghanistan  1999 sp    m     014       8
##  4 Afghanistan  2000 sp    m     014      52
##  5 Afghanistan  2001 sp    m     014     129
##  6 Afghanistan  2002 sp    m     014      90
##  7 Afghanistan  2003 sp    m     014     127
##  8 Afghanistan  2004 sp    m     014     139
##  9 Afghanistan  2005 sp    m     014     151
## 10 Afghanistan  2006 sp    m     014     193
## # ... with 76,036 more rows
```

Many errors happen, because `separate()` emits "too few values".

3

I claimed that `iso2` and `iso3` were redundant with country. Confirm this claim.

```
who %>% nest(-country) %>% nrow()
```

```
## [1] 219
```

```
who %>% nest(-country, -iso2, -iso3) %>% nrow()
```

```
## [1] 219
```

Above results show that `country`, `iso2`, and `iso3` is complete matching.

4

For each country, year, and sex compute the total number of cases of TB. Make an informative visualisation of the data.

```
who %>% gather(key, value, new_sp_m014:newrel_f65, na.rm = TRUE) %>%
    mutate(key = stringr::str_replace(key, "newrel", "new_rel")) %>%
    separate(key, c("new", "var", "sexage")) %>%
    select(-new, -iso2, -iso3) %>%
    separate(sexage, c("sex", "age"), sep = 1) %>%
    group_by(country, year, sex) %>%
    summarize(cases=sum(value)) %>%
    unite(country_sex, country, sex, remove=FALSE) %>%
    ggplot(aes(x=year, y=cases, group=country_sex, color=sex)) +
    geom_line()
```