RDS-Problems

JingjianGao

2022-10-04

12.2.1.2

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

Extract the number of TB cases per country per year. Extract the matching population per country per year. Divide cases by population, and multiply by 10000. Store back in the appropriate place. Which representation is easiest to work with? Which is hardest? Why?

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
data("table2")
table2_cases <- filter(table2,type=="cases") %>%
  rename(cases=count)%>%
  arrange(country,year)

table2_population <- filter(table2,type=="population") %>%
  rename(population=count) %>%
  arrange(country,year)

table2_cases_population_ratio <- tibble(
  year=table2_cases$year,
  country=table2_cases$country,
  cases=table2_cases$cases,
  population=table2_population$population
) %>%
  mutate(cases_population_ratio=cases/population*10000) %>%
  select(country,year,cases_population_ratio)

table2_cases_population_ratio <-table2_cases_population_ratio %>%
  mutate(type="cases_population_ratio")%>%
  rename(count=cases_population_ratio)

new_table2 <- bind_rows(table2,table2_cases_population_ratio) %>%
    arrange(country, year, type, count)
```

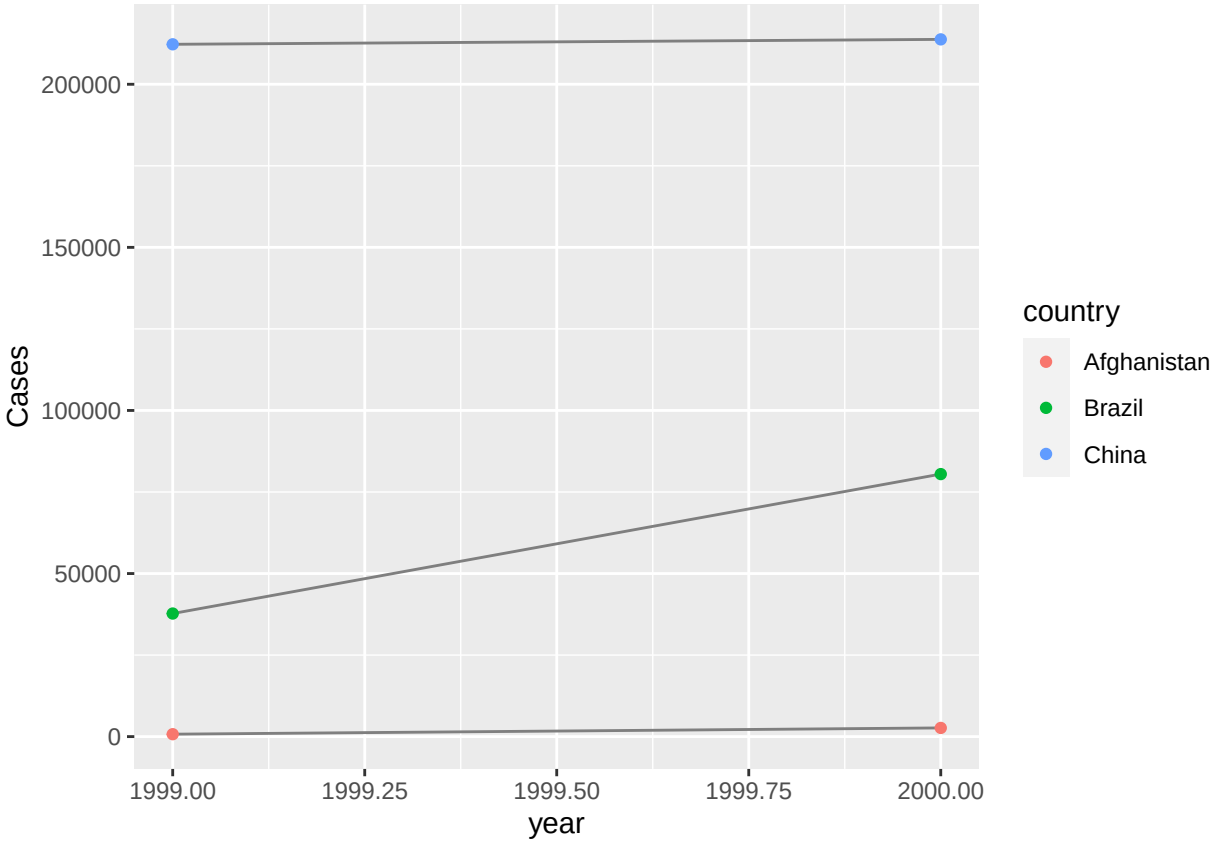
```
data("table4a")
data("table4b")
table4_ab <-  tibble(country = table4a$country,
                     `1999` = table4a[["1999"]] / table4b[["1999"]] * 10000,
                     `2000` = table4a[["2000"]] / table4b[["2000"]] * 10000
)


#As you can see from the code, I think table2 is hard to work with. Table4a and
#table4b are much easier to handle.
```

12.2.1.3

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))+
  ylab("Cases")
```



12.3.3.1

Why are pivot_longer() and pivot_wider() 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 %>% pivot_wider(names_from = year, values_from = return) %>% pivot_longer(2015:2016, names_to = “year”, values_to = “return”)

Hint: look at the variable types and think about column names.) pivot_longer() has a names_ptypes argument, e.g. names_ptypes = list(year = double()). What does it do?

```
#They are not perfectly symmetrical because the column type information is missing when we transform th
#Character, Numeric type differences, etc
```

12.3.3.2

Why does this code fail?

table4a %>% pivot_longer(c(1999, 2000), names_to = “year”, values_to = “cases”) #> Error: Can’t subset columns that don’t exist. #> Locations 1999 and 2000 don’t exist. #> There are only 3 columns.

```
#This code fails because the function interprets 1999 and 2000 to be 1999th and 2000th column instead o
```

12.3.3.3

What would happen if you widen this table? Why? How could you add a new column to uniquely identify each value?

```
people <- tribble(
  ~name,             ~names,  ~values,
  #----------------/--------/------
  "Phillip Woods",   "age",       45,
  "Phillip Woods",   "height",   186,
  "Phillip Woods",   "age",       50,
  "Jessica Cordero", "age",       37,
  "Jessica Cordero", "height",   156
)
pivot_wider(people, names_from="name", values_from = "values")
```

```
## Warning: Values from `values` are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = {summary_fun}` to summarise duplicates.
## * Use the following dplyr code to identify duplicates.
##   {data} %>%
##     dplyr::group_by(names, name) %>%
##     dplyr::summarise(n = dplyr::n(), .groups = "drop") %>%
##     dplyr::filter(n > 1L)
```

```
## # A tibble: 2 x 3
##   names  `Phillip Woods` `Jessica Cordero`
##   <chr>  <list>          <list>
## 1 age    <dbl [2]>       <dbl [1]>
## 2 height <dbl [1]>       <dbl [1]>
```

```
#If we widen this table, then it will return error because name and names columns do not uniquely ident
#We can add a new column showing the unique combination of name and names.
```

12.3.3.4

Tidy the simple tibble below. Do you need to make it wider or longer? What are the variables?

```r
preg <- tribble(
  ~pregnant, ~male, ~female,
  "yes",     NA,    10,
  "no",      20,    12
)
tidy_preg <- preg %>%
    pivot_longer(c(male, female), names_to = "sex", values_to = "count",values_drop_na = TRUE)
tidy_preg
```

```
## # A tibble: 3 x 3
##   pregnant sex    count
##   <chr>    <chr>  <dbl>
## 1 yes      female    10
## 2 no       male      20
## 3 no       female    12
```

```r
#We need to make it longer. The variables are sex, pregnant, and number of observations
```