Chapter 12.3 Exercises

Nick Sun

May 1, 2019

Chapter Notes: Common problems and how to fix em

1. One variable might be spread out across multiple columns
2. One observation might be spread across multiple rows

In `table4a`, we see that the columns 1999 and 2000 are not the names of variables, but rather values of a variable. Each row represents two observations, not one.

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

```
## Joining, by = c("country", "year")
```

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

Spreading is the opposite of gathering. You use it when an observations is scattered across multiple rows. An example would be `table2`; the `type` column contains the names of columns and the values of those columns are held in the `count` column.

```
## # 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
```

```
## # 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
```

Question 1

Why are `gather` and `spread` not perfectly symmetrical? Consider the following problem:

```
## # A tibble: 4 x 3
##    year  half return
##   <dbl> <dbl>  <dbl>
## 1  2015     1   1.88
## 2  2015     2   0.59
## 3  2016     1   0.92
## 4  2016     2   0.17
```

```
## # A tibble: 2 x 3
##    half `2015` `2016`
##   <dbl>  <dbl>  <dbl>
## 1     1   1.88   0.92
## 2     2   0.59   0.17
```

```
## # A tibble: 4 x 3
##    half year  return
##   <dbl> <chr>  <dbl>
## 1     1 2015    1.88
## 2     2 2015    0.59
## 3     1 2016    0.92
## 4     2 2016    0.17
```

The table `stocks` originally has an observation spread out over multiple rows. `Spread` makes it so that `year` is now split into individual columns with `return` as the values in those columns. Running `gather` on this makes a brand new column `year` that is different from the original data in that it is a character vector instead of a dbl vector.

The `convert` argument in `gather` and `spread` essentially runs another function `type_convert` on the new columns or rows. This function converts data to appropriate types, for example logicals, integers, etc. If we rerun our earlier code with `convert=TRUE`, we get the following:

```
## # A tibble: 4 x 3
##    half  year return
##   <dbl> <int>  <dbl>
## 1     1  2015   1.88
## 2     2  2015   0.59
## 3     1  2016   0.92
## 4     2  2016   0.17
```

The output is largely unchanged except that `year` is an integer column instead of character.

Question 2: Why does this code fail?

```
# table4a %>%
#   gather(1999, 2000, key = "year", value = "cases")

table4a %>%
  gather(`1999`, `2000`, key = "year", value = "cases")
```

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

The column names have to be within backticks. Otherwise, R thinks that 1999 and 2000 are just integers. You don't need backticks if the name of the columns are characters.

Question 3

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

```
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
```

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

```
# people %>%
#   spread(key = key, value = value)
```

We get an error here because each row of output must be unique. Rows 1 and 3 have the same keys - they both have the age of Philip Woods. To fix this, we can add a column that has distinct values.

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

Question 4

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

The rules for tidy data are as follows:

1. Each variable must have its own column.
2. Each observation must have its own row.
3. Each value must have its own cell.

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

We can gather `male` and `female` into one column called `sex`. We can also remove the NA observation by setting na.rm equal to TRUE.

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