R for Data Science: Tidy Data

Monica Buczynski

8/14/2020

Note: The purpose of this document is to showcase a sample of skills covered in *R for Data Science* (chapter: Tidy Data) by Garrett Grolemund and Hadley Wickham. All scripts were taken from https://r4ds.had.co.nz/tidy-data.html and https://jrnold.github.io/r4ds-exercise-solutions/index.html. The code for each exercise was studied carefully for understanding and then was retyped manually into R to maximize the learning experience; however, many of the original scripts were altered for further experimentation and presentation aesthetics.

The skills that I focused on include:

- Tidy Data
- Pivoting
- Separating and uniting
- Missing values

Tidy Data

1) Compute rate per 10,000.

```
table1 %>%
  mutate(rate = cases/population * 10000)
```

```
## # 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
```

2) Compute cases per year.

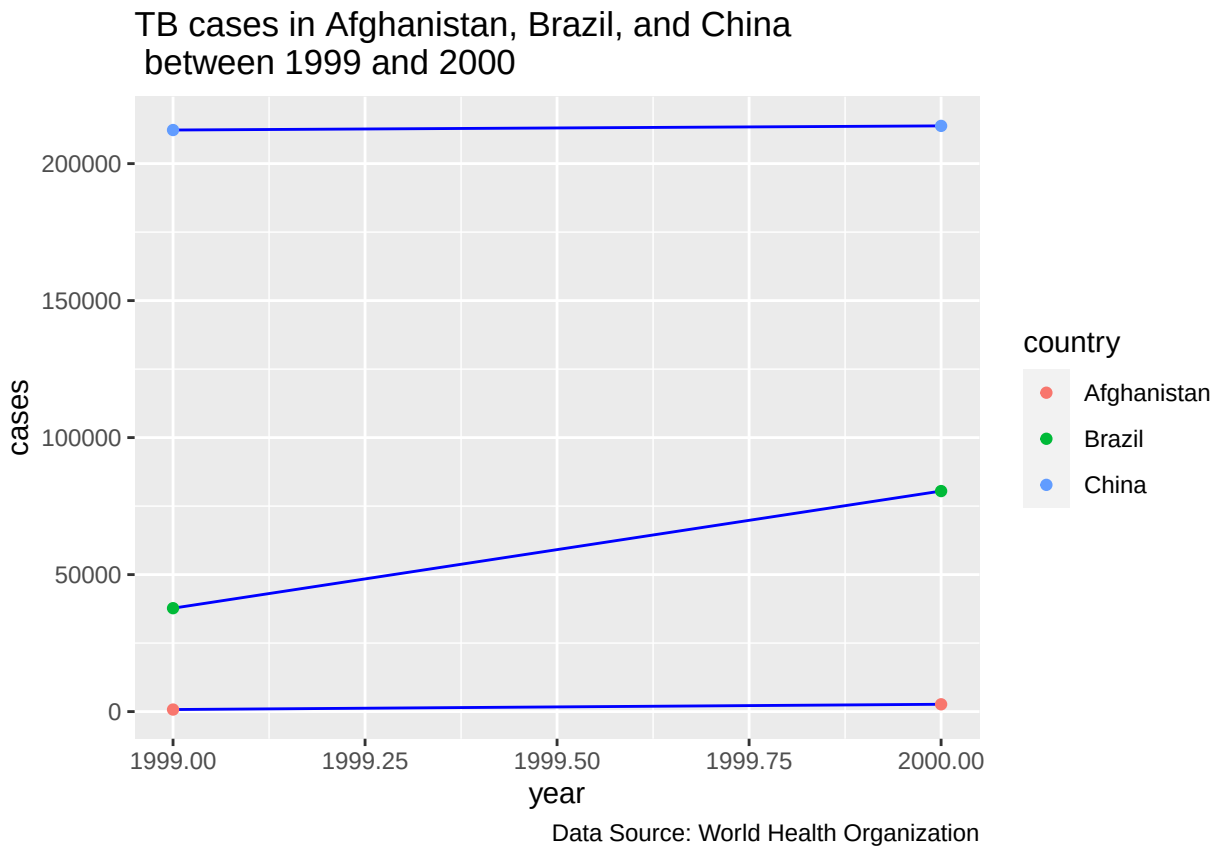
```
table1 %>%
  count(year, wt = cases)
```

```
## # A tibble: 2 x 2
##    year      n
##   <int>  <int>
## 1  1999 250740
## 2  2000 296920
```

3) Visualize changes over time

```
g1 <- ggplot(table1, aes(year, cases)) +
  geom_line(aes(group = country), colour = "blue") +
  geom_point(aes(colour = country))

g1 + labs(title = "TB cases in Afghanistan, Brazil, and China\n between 1999 and 2000",
          caption = "Data Source: World Health Organization")
```



4) Compute the rate for table2. I 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 10000.
- d) Store back in the appropriate place.

```
# View table2
table2
```

First, create separate tables for cases and population and ensure that they are sorted in the same order.

```
## # 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
```

```
# 4a)
t2_cases <- filter(table2, type == "cases") %>%
  rename(cases = count) %>%
  arrange(country, year)

# 4b)
  t2_population <- filter(table2, type == "population") %>%
  rename(population = count) %>%
  arrange(country, year)

# 4c) Create a new data frame with the population and cases columns,
  # and calculate the cases per capita in a new column.

 t2_cases_per_cap <- tibble(
  year = t2_cases$year,
  country = t2_cases$country,
  cases = t2_cases$cases,
  population = t2_population$population) %>%
  mutate(cases_per_cap = (cases / population) * 10000) %>%
  select(country, year, cases_per_cap)

# 4d) To store this new variable in the appropriate location, I will add new rows to table2.

t2_cases_per_cap <- t2_cases_per_cap %>%
```

```
  mutate(type = "cases_per_cap") %>%
  rename(count = cases_per_cap)

bind_rows(table2, t2_cases_per_cap) %>%
  arrange(country, year, type, count)
```

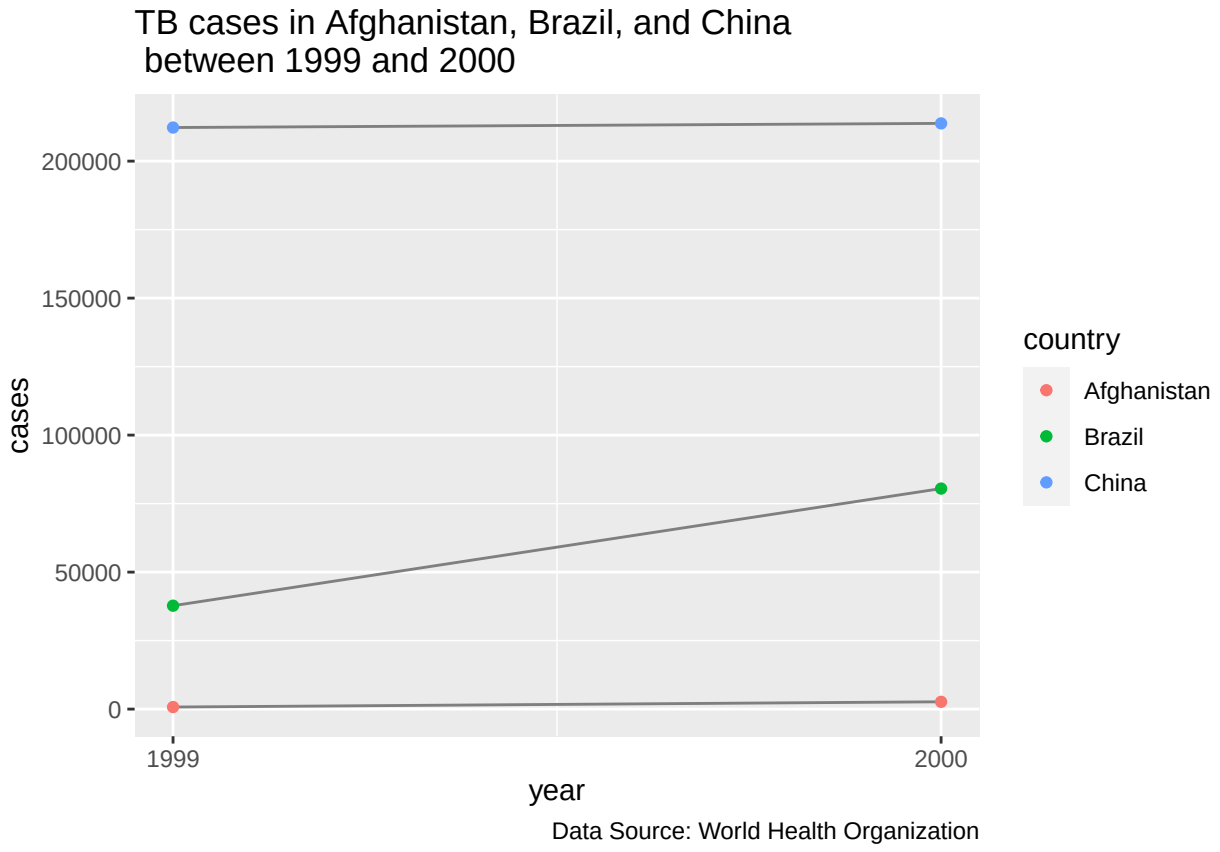
```
## # A tibble: 18 x 4
##    country      year type            count
##    <chr>       <int> <chr>           <dbl>
##  1 Afghanistan  1999 cases         7.45e+2
##  2 Afghanistan  1999 cases_per_cap 3.73e-1
##  3 Afghanistan  1999 population    2.00e+7
##  4 Afghanistan  2000 cases         2.67e+3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

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

```
# Before creating the plot with change in cases over time, we need to filter table to only
  # include rows representing cases of TB.


g2 <-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))

g2 + labs(title = "TB cases in Afghanistan, Brazil, and China\n between 1999 and 2000", y = "cases",
         caption = "Data Source: World Health Organization")
```



Pivoting

6) In table4a, the column names 1999 and 2000 represent values of the year variable, the values in the 1999 and 2000 columns represent values of the cases variable, and each row represents two observations, not one. Use pivot_longer.

Goals:

- The set of columns whose names are values, not variables. In this example, those are the columns 1999 and 2000.
- The name of the variable to move the column names to. Here it is year.
- The name of the variable to move the column values to. Here it's cases.

```
# View table4a
table4a
```

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

```
table4a %>%
  pivot_longer(c('1999', '2000'), names_to = "cases")
```

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

```
# year and cases do not exist in
  # table4a so we put their names in quotes.
```

7) Use pivot_longer to tidy table4b.

```
table4b
```

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

```
table4b %>%
  pivot_longer(c('1999', '2000'), names_to = "year", values_to = "population")
```

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

8) To combine the tidied versions:

```
tidy4a <- table4a %>%
  pivot_longer(c('1999','2000'), names_to = "year", values_to = "cases")

tidy4b <- table4b %>%
  pivot_longer(c('1999', '2000'), names_to = "year", values_to = "population")

left_join(tidy4a, tidy4b)
```

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

```
## # A tibble: 6 x 4
##   country     year   cases population
##   <chr>       <chr>  <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
```

9) Use pivot_wider when an observation is scattered across multiple rows. For example, take table2: an observation is a country in a year, but each observation is spread across two rows.

```
# View table2a

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
```

```
table2 %>%
  pivot_wider(names_from = type, values_from = count)
```

```
## # 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
```

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

```
# Example data

(preg <- tribble(
  ~pregnant, ~male, ~female,
  "yes", NA, 10,
  "no", 20, 12
))
```

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

```
# The variables are:
  # sex ("female", "male")
  # pregnant ("yes", "no")
  # count, which is a non-negative integer representing the number of observations

(preg_tidy <- preg %>%
  pivot_longer(c(male, female), names_to = "sex", values_to = "count"))
```

```
## # A tibble: 4 x 3
##   pregnant sex    count
##   <chr>    <chr>  <dbl>
## 1 yes      male      NA
## 2 yes      female    10
## 3 no       male      20
## 4 no       female    12
```

```
# Remove the (male, pregnant) row with a missing value to simplify the tidied data frame.

(preg_tidy2 <- preg %>%
  pivot_longer(c(male, female), names_to = "sex", values_to = "count", values_drop_na = TRUE))
```

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

```
# I can clean the data further by storing the variables as logical vectors

(preg_tidy3 <- preg_tidy2 %>%
    mutate(
      female = sex == "female",
      pregnant = pregnant == "yes") %>%
    select(female, pregnant, count))
```

```
## # A tibble: 3 x 3
##   female pregnant count
##   <lgl>  <lgl>    <dbl>
## 1 TRUE   TRUE        10
```

```
## 2 FALSE  FALSE       20
## 3 TRUE   FALSE       12
```

```
#  Compare the filter() calls to select non-pregnant females from preg_tidy2 and preg_tidy.

(filter(preg_tidy2, sex == "female", pregnant == "no"))
```

```
## # A tibble: 1 x 3
##   pregnant sex    count
##   <chr>    <chr>  <dbl>
## 1 no       female    12
```

```
(filter(preg_tidy3, female, !pregnant))
```

```
## # A tibble: 1 x 3
##   female pregnant count
##   <lgl>  <lgl>    <dbl>
## 1 TRUE   FALSE       12
```

Separating and uniting

11) Use the separate() function to fix table3 which has one column (rate) that contains two variables (cases and population).

```
# View table3
table3
```

```
## # A tibble: 6 x 3
##   country      year rate
## * <chr>       <int> <chr>
## 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
```

```
table3 %>%
  separate(rate, into = c("cases", "population"))
```

```
## # A tibble: 6 x 4
##   country      year cases  population
##   <chr>       <int> <chr>  <chr>
## 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
```

12) Convert cases and population (which are character columns after using separate()) to integers.

```
table3 %>%
  separate(rate, into = c("cases", "population"), convert = TRUE)
```

```
## # 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
```

13) Use sep = 2 arrangement to separate the last two digits of each year.

```
table3 %>%
  separate(year, into = c("century", "year"), sep = 2)
```

```
## # A tibble: 6 x 4
##   country     century year  rate
##   <chr>       <chr>   <chr> <chr>
```

```
## 1 Afghanistan 19      99    745/19987071
## 2 Afghanistan 20      00    2666/20595360
## 3 Brazil      19      99    37737/172006362
## 4 Brazil      20      00    80488/174504898
## 5 China       19      99    212258/1272915272
## 6 China       20      00    213766/1280428583
```

14) Use unite() to combine multiple columns into a single column of table5.

```
# View table5

table5
```

```
## # A tibble: 6 x 4
##   country     century year  rate
## * <chr>       <chr>   <chr> <chr>
## 1 Afghanistan 19      99    745/19987071
## 2 Afghanistan 20      00    2666/20595360
## 3 Brazil      19      99    37737/172006362
## 4 Brazil      20      00    80488/174504898
## 5 China       19      99    212258/1272915272
## 6 China       20      00    213766/1280428583
```

```
table5 %>%
  unite(new, century, year)
```

```
## # A tibble: 6 x 3
##   country     new   rate
##   <chr>       <chr> <chr>
## 1 Afghanistan 19_99 745/19987071
## 2 Afghanistan 20_00 2666/20595360
## 3 Brazil      19_99 37737/172006362
## 4 Brazil      20_00 80488/174504898
## 5 China       19_99 212258/1272915272
## 6 China       20_00 213766/1280428583
```

```
# Use sep = "" argument to remove the underscore in "new" column

(table5a <- table5%>%
  unite(new, century, year, sep = ""))
```

```
## # A tibble: 6 x 3
##   country     new   rate
##   <chr>       <chr> <chr>
## 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
```

```
# Use separate function to separate "rate" column into "cases" and "population" and use rename function

table5a %>%
    separate(rate, into = c("cases", "population"), convert = TRUE) %>%
rename(year = new)
```

```
## # A tibble: 6 x 4
##   country     year   cases population
##   <chr>       <chr>  <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
```

Missing Values

- Explicitly, i.e. flagged with NA.
- Implicitly, i.e. simply not present in the data.

15)

```
# View original dataset

(stocks <- tibble(
  year   = c(2015, 2015, 2015, 2015, 2016, 2016, 2016),
  qtr    = c(   1,    2,    3,    4,    2,    3,    4),
  return = c(1.88, 0.59, 0.35,   NA, 0.92, 0.17, 2.66)
))
```

```
## # A tibble: 7 x 3
##    year   qtr return
##   <dbl> <dbl>  <dbl>
## 1  2015     1   1.88
## 2  2015     2   0.59
## 3  2015     3   0.35
## 4  2015     4  NA
## 5  2016     2   0.92
## 6  2016     3   0.17
## 7  2016     4   2.66
```

```
# The return for the fourth quarter of 2015 is explicitly missing, because the cell
  #where its value should be instead contains NA.
# The return for the first quarter of 2016 is implicitly missing, because it simply
  #does not appear in the dataset.

# Make implicit values explicit

stocks %>%
  pivot_wider(names_from = year, values_from = return)
```

```
## # A tibble: 4 x 3
##     qtr `2015` `2016`
##   <dbl>  <dbl>  <dbl>
## 1     1   1.88  NA
## 2     2   0.59   0.92
## 3     3   0.35   0.17
## 4     4  NA      2.66
```

```
# Drop explicit missing values.

stocks %>%
  pivot_wider(names_from = year, values_from = return) %>%
  pivot_longer(
    cols = c('2015', '2016'),
    names_to = "year",
    values_to = "return",
    values_drop_na = TRUE)
```

```
## # A tibble: 6 x 3
##     qtr year  return
```

```
##   <dbl> <chr>  <dbl>
## 1     1 2015    1.88
## 2     2 2015    0.59
## 3     2 2016    0.92
## 4     3 2015    0.35
## 5     3 2016    0.17
## 6     4 2016    2.66
```

```
# Make missing values explicit using complete().

stocks %>%
  complete(year, qtr)
```

```
## # A tibble: 8 x 3
##    year   qtr return
##   <dbl> <dbl>  <dbl>
## 1  2015     1   1.88
## 2  2015     2   0.59
## 3  2015     3   0.35
## 4  2015     4  NA
## 5  2016     1  NA
## 6  2016     2   0.92
## 7  2016     3   0.17
## 8  2016     4   2.66
```

```
# Fill missing values with fill() which carries the last observation forward.

treatment <- tribble(
  ~ person,           ~ treatment, ~response,
  "Derrick Whitmore", 1,           7,
  NA,                 2,           10,
  NA,                 3,           9,
  "Katherine Burke",  1,           4
)

treatment %>%
  fill(person)
```

```
## # A tibble: 4 x 3
##   person           treatment response
##   <chr>                <dbl>    <dbl>
## 1 Derrick Whitmore         1        7
## 2 Derrick Whitmore         2       10
## 3 Derrick Whitmore         3        9
## 4 Katherine Burke          1        4
```