

# Lab 6: Tidy Data Case Study

Anthony Hu

6/4/2021

## Introduction

To finish off Chapter 9, let's pull together everything you've learned to tackle a realistic data tidying problem. The `tidyr::who` dataset contains tuberculosis (TB) cases broken down by year, country, age, gender, and diagnosis method. The data comes from the 2014 World Health Organization Global Tuberculosis Report, available at <http://www.who.int/tb/country/data/download/en/>.

There's a wealth of epidemiological information in this dataset (7240 rows, 60 columns), but it's challenging to work with the data in the form that it's provided:

```
head(who, 10)
```

```
## # A tibble: 10 x 60
##   country    iso2 iso3  year new_sp_m014 new_sp_m1524 new_sp_m2534
##   <chr>      <chr> <chr> <int>      <int>          <int>      <int>
## 1 Afghanistan AF    AFG  1980         NA            NA            NA
## 2 Afghanistan AF    AFG  1981         NA            NA            NA
## 3 Afghanistan AF    AFG  1982         NA            NA            NA
## 4 Afghanistan AF    AFG  1983         NA            NA            NA
## 5 Afghanistan AF    AFG  1984         NA            NA            NA
## 6 Afghanistan AF    AFG  1985         NA            NA            NA
## 7 Afghanistan AF    AFG  1986         NA            NA            NA
## 8 Afghanistan AF    AFG  1987         NA            NA            NA
## 9 Afghanistan AF    AFG  1988         NA            NA            NA
## 10 Afghanistan AF    AFG  1989         NA            NA            NA
## # ... with 53 more variables: new_sp_m3544 <int>, new_sp_m4554 <int>,
## #   new_sp_m5564 <int>, new_sp_m65 <int>, new_sp_f014 <int>,
## #   new_sp_f1524 <int>, new_sp_f2534 <int>, new_sp_f3544 <int>,
## #   new_sp_f4554 <int>, new_sp_f5564 <int>, new_sp_f65 <int>,
## #   new_sn_m014 <int>, new_sn_m1524 <int>, new_sn_m2534 <int>,
## #   new_sn_m3544 <int>, new_sn_m4554 <int>, new_sn_m5564 <int>,
## #   new_sn_m65 <int>, new_sn_f014 <int>, new_sn_f1524 <int>,
## #   new_sn_f2534 <int>, new_sn_f3544 <int>, new_sn_f4554 <int>,
## #   new_sn_f5564 <int>, new_sn_f65 <int>, new_ep_m014 <int>,
## #   new_ep_m1524 <int>, new_ep_m2534 <int>, new_ep_m3544 <int>,
## #   new_ep_m4554 <int>, new_ep_m5564 <int>, new_ep_m65 <int>,
## #   new_ep_f014 <int>, new_ep_f1524 <int>, new_ep_f2534 <int>,
## #   new_ep_f3544 <int>, new_ep_f4554 <int>, new_ep_f5564 <int>,
## #   new_ep_f65 <int>, newrel_m014 <int>, newrel_m1524 <int>,
## #   newrel_m2534 <int>, newrel_m3544 <int>, newrel_m4554 <int>,
## #   newrel_m5564 <int>, newrel_m65 <int>, newrel_f014 <int>,
## #   newrel_f1524 <int>, newrel_f2534 <int>, newrel_f3544 <int>,
## #   newrel_f4554 <int>, newrel_f5564 <int>, newrel_f65 <int>
```

This is a very typical real-life example dataset. It contains redundant columns, odd variable codes, and many missing values. In short, `who` is messy, and we'll need multiple steps to tidy it. Like `dplyr`, `tidyr` is

designed so that each function does one thing well. That means in real-life situations you'll usually need to string together multiple verbs into a pipeline.

When you get the desired result for each step, change `Eval=F` to `Eval=T` and knit the document to PDF to make sure it works. After you complete the lab, you should submit your PDF file of what you have completed to Sakai before the deadline.

## Part 1: Gather Variables Together

Some observations on the data:

- It looks like `country`, `iso2`, and `iso3` are three variables that redundantly specify the country.
- `year` is clearly also a variable.
- We don't know what all the other columns are yet, but given the structure in the variable names (e.g. `new_sp_m014`, `new_ep_m014`, `new_ep_f014`) these are likely to be values, not variables.

### Q1: Gather together all the columns from `new_sp_m014` to `newrel_f65`.

We don't know what those values represent yet, so we'll give them the generic name "key". We know the cells represent the count of cases, so we'll use the variable `cases`. There are a lot of missing values in the current representation, so for now we'll use `values_drop_na` just so we can focus on the values that are present.

```
who1 <- who %>%
  pivot_longer(
    cols = new_sp_m014:newrel_f65,
    names_to = "key",
    values_to = "cases",
    values_drop_na = TRUE
  )
head(who1, 10)
```

```
## # A tibble: 10 x 6
##   country    iso2 iso3   year key      cases
##   <chr>      <chr> <chr> <int> <chr>    <int>
## 1 Afghanistan AF    AFG   1997 new_sp_m014    0
## 2 Afghanistan AF    AFG   1997 new_sp_m1524   10
## 3 Afghanistan AF    AFG   1997 new_sp_m2534    6
## 4 Afghanistan AF    AFG   1997 new_sp_m3544    3
## 5 Afghanistan AF    AFG   1997 new_sp_m4554    5
## 6 Afghanistan AF    AFG   1997 new_sp_m5564    2
## 7 Afghanistan AF    AFG   1997 new_sp_m65      0
## 8 Afghanistan AF    AFG   1997 new_sp_f014     5
## 9 Afghanistan AF    AFG   1997 new_sp_f1524   38
## 10 Afghanistan AF    AFG   1997 new_sp_f2534   36
```

### Q2: Separate key column

For the `key` column, the data dictionary tells us:

- The first three letters of each column denote whether the column contains new or old cases of TB. In this dataset, each column contains new cases.
- The next two letters describe the type of TB:
  - **rel** stands for cases of relapse
  - **ep** stands for cases of extrapulmonary TB
  - **sn** stands for cases of pulmonary TB that could not be diagnosed by a pulmonary smear (smear negative)
  - **sp** stands for cases of pulmonary TB that could be diagnosed by a pulmonary smear (smear positive)
- The sixth letter gives the sex of TB patients. The dataset groups cases by males (**m**) and females (**f**).
- The remaining numbers gives the age group. The dataset groups cases into seven age groups:
  - 014 = 0 – 14 years old
  - 1524 = 15 – 24 years old
  - 2534 = 25 – 34 years old
  - 3544 = 35 – 44 years old
  - 4554 = 45 – 54 years old
  - 5564 = 55 – 64 years old
  - 65 = 65 or older
- The names are slightly inconsistent for **key** because instead of **new\_rel** we have **newrel**. Run the following code to make it consistent:

```
who2 <- who1 %>%
  mutate(key = stringr::str_replace(key, "newrel", "new_rel"))
who2
```

```
## # A tibble: 76,046 x 6
##   country    iso2 iso3  year key      cases
##   <chr>      <chr> <chr> <int> <chr>    <int>
## 1 Afghanistan AF    AFG  1997 new_sp_m014      0
## 2 Afghanistan AF    AFG  1997 new_sp_m1524     10
## 3 Afghanistan AF    AFG  1997 new_sp_m2534      6
## 4 Afghanistan AF    AFG  1997 new_sp_m3544      3
## 5 Afghanistan AF    AFG  1997 new_sp_m4554      5
## 6 Afghanistan AF    AFG  1997 new_sp_m5564      2
## 7 Afghanistan AF    AFG  1997 new_sp_m65       0
## 8 Afghanistan AF    AFG  1997 new_sp_f014      5
## 9 Afghanistan AF    AFG  1997 new_sp_f1524     38
## 10 Afghanistan AF    AFG  1997 new_sp_f2534     36
## # ... with 76,036 more rows
```

**Q2:** Separate the **key** column into columns **new**, **type** and **sexage**. Then drop the **new** column because it's constant in this dataset. Please also drop **iso2**, **iso3** as they are also redundant.

```
who3 <- who2 %>%
  separate(key, c("new", "type", "sexage"), sep = "_") %>%
  select(-c("new", "iso2", "iso3"))
who3
```

```
## # A tibble: 76,046 x 5
##   country      year type  sexage cases
##   <chr>        <int> <chr> <chr> <int>
## 1 Afghanistan 1997 sp    m014     0
## 2 Afghanistan 1997 sp    m1524    10
## 3 Afghanistan 1997 sp    m2534     6
## 4 Afghanistan 1997 sp    m3544     3
## 5 Afghanistan 1997 sp    m4554     5
## 6 Afghanistan 1997 sp    m5564     2
## 7 Afghanistan 1997 sp     m65     0
## 8 Afghanistan 1997 sp    f014     5
## 9 Afghanistan 1997 sp    f1524    38
## 10 Afghanistan 1997 sp    f2534    36
## # ... with 76,036 more rows
```

**Q3:** Separate the `sexage` column into columns `sex` and `age`. (Hint: if `sep=Number`, interpreted as positions to split at)

```
who4 <- who3 %>%
  separate(sexage, c("sex", "age"), sep = 1)
who4
```

```
## # A tibble: 76,046 x 6
##   country      year type  sex  age  cases
##   <chr>        <int> <chr> <chr> <chr> <int>
## 1 Afghanistan 1997 sp    m    014     0
## 2 Afghanistan 1997 sp    m   1524    10
## 3 Afghanistan 1997 sp    m   2534     6
## 4 Afghanistan 1997 sp    m   3544     3
## 5 Afghanistan 1997 sp    m   4554     5
## 6 Afghanistan 1997 sp    m   5564     2
## 7 Afghanistan 1997 sp    m    65     0
## 8 Afghanistan 1997 sp    f    014     5
## 9 Afghanistan 1997 sp    f   1524    38
## 10 Afghanistan 1997 sp    f   2534    36
## # ... with 76,036 more rows
```

**Q4:** Put all steps in one code chunk with pipe operator

```
#
a = who %>%
  pivot_longer(
    cols = new_sp_m014:newrel_f65,
    names_to = "key",
    values_to = "cases",
    values_drop_na = TRUE
  ) %>%
  mutate(key = stringr::str_replace(key, "newrel", "new_rel")) %>%
  separate(key, c("new", "type", "sexage"), sep = "_") %>%
  select(-c("new", "iso2", "iso3")) %>%
```

```
separate(sexage, c("sex", "age"), sep = 1)
a
```

```
## # A tibble: 76,046 x 6
##   country      year type  sex   age  cases
##   <chr>      <int> <chr> <chr> <chr> <int>
## 1 Afghanistan 1997 sp    m    014     0
## 2 Afghanistan 1997 sp    m   1524    10
## 3 Afghanistan 1997 sp    m   2534     6
## 4 Afghanistan 1997 sp    m   3544     3
## 5 Afghanistan 1997 sp    m   4554     5
## 6 Afghanistan 1997 sp    m   5564     2
## 7 Afghanistan 1997 sp    m    65     0
## 8 Afghanistan 1997 sp    f    014     5
## 9 Afghanistan 1997 sp    f   1524    38
## 10 Afghanistan 1997 sp    f   2534    36
## # ... with 76,036 more rows
```

## More Exercises

First, let's import a Comma Separated Values .csv file that exists on the internet. The .csv file `dem_score.csv` contains ratings of the level of democracy in different countries spanning 1952 to 1992 and is accessible at [https://moderndive.com/data/dem\\_score.csv](https://moderndive.com/data/dem_score.csv). Let's use the `read_csv()` function from the `readr` package to read it off the web, import it into R, and save it in a data frame called `dem_score`. In the following part, we're going to focus on only data corresponding to Guatemala.

```
dem_score <- read_csv("https://moderndive.com/data/dem_score.csv")
```

```
##
## -- Column specification -----
## cols(
##   country = col_character(),
##   '1952' = col_double(),
##   '1957' = col_double(),
##   '1962' = col_double(),
##   '1967' = col_double(),
##   '1972' = col_double(),
##   '1977' = col_double(),
##   '1982' = col_double(),
##   '1987' = col_double(),
##   '1992' = col_double()
## )
```

```
dem_score
```

```
## # A tibble: 96 x 10
##   country '1952' '1957' '1962' '1967' '1972' '1977' '1982' '1987' '1992'
##   <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Albania     -9     -9     -9     -9     -9     -9     -9     -9     5
## 2 Argentina   -9     -1     -1     -9     -9     -9     -8     8     7
```

```
## 3 Armenia      -9      -7      -7      -7      -7      -7      -7      -7      7
## 4 Australia    10      10      10      10      10      10      10      10      10
## 5 Austria      10      10      10      10      10      10      10      10      10
## 6 Azerbaij~    -9      -7      -7      -7      -7      -7      -7      -7      1
## 7 Belarus      -9      -7      -7      -7      -7      -7      -7      -7      7
## 8 Belgium      10      10      10      10      10      10      10      10      10
## 9 Bhutan       -10     -10     -10     -10     -10     -10     -10     -10     -10
## 10 Bolivia      -4      -3      -3      -4      -7      -7      8      9      9
## # ... with 86 more rows
```

**Q5:** In the following part, we're going to focus on only data corresponding to Guatemala.

```
guat_dem <- dem_score %>%
  filter(country == "Guatemala")
guat_dem
```

```
## # A tibble: 1 x 10
##   country '1952' '1957' '1962' '1967' '1972' '1977' '1982' '1987' '1992'
##   <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Guatemala      2     -6     -5      3      1     -3     -7      3      3
```

**Q6:** Gather the columns and put column names to a new variable year and put values to a new variable democracy\_score. Make sure the year column is of integer type.

```
guat_dem_tidy <- guat_dem %>%
  pivot_longer(cols = 2:10,
               names_to = "year",
               values_to = "democracy_score") %>%
  mutate(year = as.numeric(year))
guat_dem_tidy
```

```
## # A tibble: 9 x 3
##   country   year democracy_score
##   <chr>    <dbl>           <dbl>
## 1 Guatemala 1952             2
## 2 Guatemala 1957            -6
## 3 Guatemala 1962            -5
## 4 Guatemala 1967             3
## 5 Guatemala 1972             1
## 6 Guatemala 1977            -3
## 7 Guatemala 1982            -7
## 8 Guatemala 1987             3
## 9 Guatemala 1992             3
```

**Q7:** Generate a plot based on the `guat_dem_tidy` data to reflect the democracy trend in Guatemala.

```
#  
ggplot(data = guat_dem_tidy, mapping = aes(x = year, y = democracy_score)) +  
  geom_smooth()+  
  labs(y = "Democracy Score", title = "Democracy Score in Guatemala from 1952 to 1992")
```