The verb `gather` is for transforming your data from messy to a tidy 'long' form.

The tuberculosis (TB) notifications data (https://github.com/datascienceprogram/ids_course_data/blob/master/TB_notifications.csv) in its current form is messy. You can read the data directly into your R session by running the following code (ignore the messages generated from the code):

```
tb_messy <- read_csv("https://raw.githubusercontent.com/datascienceprogram/ids_course_data/master/TB_notifications.csv")
```

It has case counts contained in variables that represent the sex and age group in the columns prefixed by **"new_sp"** (a way of diagnosing TB), hence there are variables corresponding to sex, age group and the number of cases spread across multiple columns.

```
tb_messy
```

```
## # A tibble: 7,891 x 23
##    country iso3   year new_sp_m04 new_sp_m514 new_sp_m014 new_sp_m1524
##    <chr>   <chr> <dbl>      <dbl>       <dbl>       <dbl>        <dbl>
##  1 Afghan~ AFG    1980         NA          NA          NA           NA
##  2 Afghan~ AFG    1981         NA          NA          NA           NA
##  3 Afghan~ AFG    1982         NA          NA          NA           NA
##  4 Afghan~ AFG    1983         NA          NA          NA           NA
##  5 Afghan~ AFG    1984         NA          NA          NA           NA
##  6 Afghan~ AFG    1985         NA          NA          NA           NA
##  7 Afghan~ AFG    1986         NA          NA          NA           NA
##  8 Afghan~ AFG    1987         NA          NA          NA           NA
##  9 Afghan~ AFG    1988         NA          NA          NA           NA
## 10 Afghan~ AFG    1989         NA          NA          NA           NA
## # ... with 7,881 more rows, and 16 more variables: new_sp_m2534 <dbl>,
## #   new_sp_m3544 <dbl>, new_sp_m4554 <dbl>, new_sp_m5564 <dbl>,
## #   new_sp_m65 <dbl>, new_sp_mu <dbl>, new_sp_f04 <dbl>, new_sp_f514 <dbl>,
## #   new_sp_f014 <dbl>, new_sp_f1524 <dbl>, new_sp_f2534 <dbl>,
## #   new_sp_f3544 <dbl>, new_sp_f4554 <dbl>, new_sp_f5564 <dbl>,
## #   new_sp_f65 <dbl>, new_sp_fu <dbl>
```

To reshape your data into a tidy long form, you will need to spread the values in the columns starting with **"new_sp"** into two columns, while keeping all the other columns fixed. This operation is achieved with the verb **`gather`** .

Give it a go!

Continue to develop your skills with **`gather`** by making your way through this exercise in RStudio on your computer. Let's try a simple example first, and suppose you have TB cases for just males and females for the years 2016, 2017, 2018 *(we are ignoring country and age group for now)*.

```
tb_smaller <- tibble(year = c(2016, 2017, 2018),
                     male = c(10, 20, 30),
                     female = c(5, 15, 12))
tb_smaller
```

```
## # A tibble: 3 x 3
##    year  male female
##   <dbl> <dbl>  <dbl>
## 1  2016    10      5
## 2  2017    20     15
## 3  2018    30     12
```

What you want to end up with is a table that resembles the following:

```
tb_smaller_long <- tibble(
  year = c(2016, 2017, 2018, 2016, 2017, 2018),
  sex = c("male", "male", "male", "female", "female", "female"),
  count = c(10,20,30, 5, 15, 12)
)
tb_smaller_long
```

```
## # A tibble: 6 x 3
##    year sex    count
##   <dbl> <chr>  <dbl>
## 1  2016 male      10
## 2  2017 male      20
## 3  2018 male      30
## 4  2016 female     5
## 5  2017 female    15
## 6  2018 female    12
```

Specify three things before you gather!

Now to use `gather`, you need to specify the:

- **key (identifier)**- this is the name given to the new variable that identifies which columns were used to go from wide to long. In the example above the key is given the name “sex”
- **values (measures)**- this is the name given to the new variable that contains the actual values of the columns that were used to go from wide to long. In the example above the values column is given the name “count”
- **names** of the columns we are ‘gathering’ from wide to long form. In this example they were male and female.

Putting this together, you can achieve the same result.

```
tb_smaller_long <- gather(tb_smaller,
                          key = "sex",
                          value = "count",
                          male, female)
tb_smaller_long
```

```
## # A tibble: 6 x 3
##    year sex    count
##   <dbl> <chr>  <dbl>
## 1  2016 male      10
## 2  2017 male      20
## 3  2018 male      30
## 4  2016 female     5
## 5  2017 female    15
## 6  2018 female    12
```

Sometimes it's easier to specify what you're not going to gather!

If you have many columns that you would like to take to long form, it can be easier to specify the columns you are **not** going to `gather` by using `-colname`.

Try it by copying and running the following code chunk:

```
tb_smaller_long <- gather(tb_smaller, "sex", "count", -year)
tb_smaller_long
```

```
## # A tibble: 6 x 3
##    year sex    count
##   <dbl> <chr>  <dbl>
## 1  2016 male      10
## 2  2017 male      20
## 3  2018 male      30
## 4  2016 female     5
## 5  2017 female    15
## 6  2018 female    12
```

Alternatively, you can choose a range of columns to `gather` using a colon (like you might do in software like Excel).

```
tb_smaller_long <- gather(tb_smaller, "sex", "count", male:female)
tb_smaller_long
```

```
## # A tibble: 6 x 3
##    year sex    count
##   <dbl> <chr>  <dbl>
## 1  2016 male      10
## 2  2017 male      20
## 3  2018 male      30
## 4  2016 female     5
## 5  2017 female    15
## 6  2018 female    12
```

Try running the following commands. What do you think the output will be?

```
gather(tb_smaller)
gather(tb_smaller, key = "sex", value = "value", male, year)
```

Take it to long form

Now all the pieces are in place, you can take our TB data from wide form to long form. Before you do, consider the **three** pieces you will need.

The **first** is the key which we will call **“sex_agegroup”**, the **second** is the value which we will call **“count”** since the values in those columns are the number of cases, and the **third** is the columns you are using to go from wide to long which are all those that start with **“new_sp”**. Since there are many, it is simpler to **exclude** the columns we aren’t gathering up - the country, year, country code.

This results in the following code:

```
tb_long <- gather(tb_messy, key = "sex_agegroup", value = "count", -country, -year, -iso3)
tb_long
```

```
## # A tibble: 157,820 x 5
##    country     iso3   year sex_agegroup count
##    <chr>       <chr> <dbl> <chr>        <dbl>
##  1 Afghanistan AFG    1980 new_sp_m04      NA
##  2 Afghanistan AFG    1981 new_sp_m04      NA
##  3 Afghanistan AFG    1982 new_sp_m04      NA
##  4 Afghanistan AFG    1983 new_sp_m04      NA
##  5 Afghanistan AFG    1984 new_sp_m04      NA
##  6 Afghanistan AFG    1985 new_sp_m04      NA
##  7 Afghanistan AFG    1986 new_sp_m04      NA
##  8 Afghanistan AFG    1987 new_sp_m04      NA
##  9 Afghanistan AFG    1988 new_sp_m04      NA
## 10 Afghanistan AFG    1989 new_sp_m04      NA
## # ... with 157,810 more rows
```

You can also reshape the data into a long form with the new **`pivot_longer()`** function from the `tidyr` package (which is also part of the `tidyverse`). As the `gather()` function remains widely used it will not be deprecated, so code containing `gather()` will continue to work. The `pivot_longer()` function provides a more intuitive approach to reshaping the data into a long form. More information about **`pivot_longer()`** as well as **`pivot_wider()`** can be found in the Pivoting vignette (https://tidyr.tidyverse.org/articles/pivot.html).

Tell us how you went

Share with other learners your results of using the different code chunks in this step.

- **Were you able to take the TB data from wide form to long form?**
- **In what way do you think you could use this verb to tidy data from a project you’re working on?**

Also consider reading and commenting on contributions made by other learners or following learners with similar interests as you.
