The `mutate` verb is used to create new variables/columns or modify existing ones.

Using **`mutate`** , you could create a new column in your tidy long form tuberculosis (TB) data that identifies rows where the age group is unknown **("u")**.

In RStudio on your computer, copy and then run the following code chunk:

```
mutate(tb_long,
       is_unknown = age_group == "u")
```

```
## # A tibble: 157,820 x 8
##    country     iso3   year type   sex   age_group count is_unknown
##    <chr>       <chr> <dbl> <chr>  <chr> <chr>     <dbl> <lgl>
##  1 Afghanistan AFG    1980 new_sp m     04           NA FALSE
##  2 Afghanistan AFG    1981 new_sp m     04           NA FALSE
##  3 Afghanistan AFG    1982 new_sp m     04           NA FALSE
##  4 Afghanistan AFG    1983 new_sp m     04           NA FALSE
##  5 Afghanistan AFG    1984 new_sp m     04           NA FALSE
##  6 Afghanistan AFG    1985 new_sp m     04           NA FALSE
##  7 Afghanistan AFG    1986 new_sp m     04           NA FALSE
##  8 Afghanistan AFG    1987 new_sp m     04           NA FALSE
##  9 Afghanistan AFG    1988 new_sp m     04           NA FALSE
## 10 Afghanistan AFG    1989 new_sp m     04           NA FALSE
## # ... with 157,810 more rows
```

What's it doing?

`mutate` takes a data set as its first argument, then one or more expressions for creating columns. The expression should be of the form **`new_col = something else`** where **`something else`** refers to how **`new_col`** should be constructed.

What else can you do?

As another example, you could paste the sex and age_group variables back together, like they were in the original messy data. The **`str_c`** function from the **`stringr`** package says combine these two character variables together.

Other functions from this package are useful for manipulating character variables, as shown by the following code chunk:

```
mutate(tb_long,
       new_var = str_c(sex, age_group))
```

```
## # A tibble: 157,820 x 8
##    country     iso3   year type   sex   age_group count new_var
##    <chr>       <chr> <dbl> <chr>  <chr> <chr>     <dbl> <chr>
##  1 Afghanistan AFG    1980 new_sp m     04           NA m04
##  2 Afghanistan AFG    1981 new_sp m     04           NA m04
##  3 Afghanistan AFG    1982 new_sp m     04           NA m04
##  4 Afghanistan AFG    1983 new_sp m     04           NA m04
##  5 Afghanistan AFG    1984 new_sp m     04           NA m04
##  6 Afghanistan AFG    1985 new_sp m     04           NA m04
##  7 Afghanistan AFG    1986 new_sp m     04           NA m04
##  8 Afghanistan AFG    1987 new_sp m     04           NA m04
##  9 Afghanistan AFG    1988 new_sp m     04           NA m04
## 10 Afghanistan AFG    1989 new_sp m     04           NA m04
## # ... with 157,810 more rows
```

You can also construct or modify multiple columns in one **`mutate`** call, and refer to newly created columns, as shown in the following code chunk:

```
mutate(tb_long,
       is_unknown = age_group == "u",
       # now we can alter age_group using if_else
       age_group = if_else(is_unknown, NA_character_, age_group))
```

```
## # A tibble: 157,820 x 8
##    country     iso3   year type   sex   age_group count is_unknown
##    <chr>       <chr> <dbl> <chr>  <chr> <chr>     <dbl> <lgl>
##  1 Afghanistan AFG    1980 new_sp m     04           NA FALSE
##  2 Afghanistan AFG    1981 new_sp m     04           NA FALSE
##  3 Afghanistan AFG    1982 new_sp m     04           NA FALSE
##  4 Afghanistan AFG    1983 new_sp m     04           NA FALSE
##  5 Afghanistan AFG    1984 new_sp m     04           NA FALSE
##  6 Afghanistan AFG    1985 new_sp m     04           NA FALSE
##  7 Afghanistan AFG    1986 new_sp m     04           NA FALSE
##  8 Afghanistan AFG    1987 new_sp m     04           NA FALSE
##  9 Afghanistan AFG    1988 new_sp m     04           NA FALSE
## 10 Afghanistan AFG    1989 new_sp m     04           NA FALSE
## # ... with 157,810 more rows
```

When using **`mutate`** it is useful to know functions that are helpful for modifying columns.

The function **`if_else`** is equivalent to the Excel function; it says when **`is_unknown`** is **TRUE**, set the value of **`age_group`** to a special **NA** value, otherwise keep **`age_group`** as is.

Other useful functions for variables are the math operators such as **`+, - , *, \`**, the **`case_when`** and **`n()`** function from the **`dplyr`** package and summary functions like **`mean, max, min, abs`**.

Piping with `filter`, `select` and `mutate`

Earlier, we saw how a sequence of pipes can be used to filter observations and select variables. We can continue to wrangle this data by adding a pipe to create the variable **is_unknown**:

```
# Take tb_long and filter for only the country Australia,
# then select all variables except iso3,
# then create the variable is_unknown = age_group == "u"
tb_long %>%
  filter(country == "Australia") %>%
  select(-iso3) %>%
  mutate(is_unknown = age_group == "u")
```

Give it a go!

Continue to develop your skills with **`mutate`** by making your way through this exercise. If you haven't already, make sure you download `tb_long.rds` (https://github.com/datascienceprogram/ids_course_data/blob/master/tb_long.rds), store it in your project data folder e.g. **first_project/data/tb_long.rds**, and read it into your R session. Once you've done this, attempt the following exercise:

- **Recode the sex variable to have values "male" and "female" instead of "m" and "f" (hint: use `if_else`)**
- **Use `spread` to put the count values into the columns male and female, then create a new variable called `diff` which contains the difference between the male and female counts.**

Within the **Comments**, share with other learners your results from the exercise, and then respond to one or all of the following:

- **What was your experience with using `mutate` to recoding the sex variables?**
- **What did you find out from using `spread` and `mutate` to compute the differences between TB incidents? For example, were there any countries that have a large difference between male and female counts of TB?**

Don't forget to contribute to the discussion by reviewing or **'Liking'** the comments made by other learners, making sure you provide constructive feedback and commentary.
