`group_by` is an adverb. When data is grouped, the behaviour of `summarise`, `mutate` and `filter` changes.

Give it a go!

This exercise consists of a series of code chunks that you can copy and run in RStudio on your computer.

To begin, use **`group_by`** to select columns that you want to partition the data by. For example, you could group the tidy long form TB data (https://github.com/datascienceprogram/ids_course_data/blob/master/tb_long.rds) by country.

Copy and run

Copy and run the following code chunk:

```
by_country <- group_by(tb_long, country)
by_country
```

```
## # A tibble: 157,820 x 7
## # Groups:   country [219]
##    country     iso3   year type   sex   age_group count
##    <chr>       <chr> <dbl> <chr>  <chr> <chr>     <dbl>
##  1 Afghanistan AFG    1980 new_sp m     04           NA
##  2 Afghanistan AFG    1981 new_sp m     04           NA
##  3 Afghanistan AFG    1982 new_sp m     04           NA
##  4 Afghanistan AFG    1983 new_sp m     04           NA
##  5 Afghanistan AFG    1984 new_sp m     04           NA
##  6 Afghanistan AFG    1985 new_sp m     04           NA
##  7 Afghanistan AFG    1986 new_sp m     04           NA
##  8 Afghanistan AFG    1987 new_sp m     04           NA
##  9 Afghanistan AFG    1988 new_sp m     04           NA
## 10 Afghanistan AFG    1989 new_sp m     04           NA
## # ... with 157,810 more rows
```

Then, compute the same five number summary, like you may have done in Summarise: compute summaries (https://www.futurelearn.com/courses/data-science-wrangling-and-workflow/1/steps/525937), but this time **`summarise`** will return **one** row for each group.

When you're ready, try it yourself using the following code chunk:

```
summarise(by_country,
          min = min(count, na.rm = TRUE),
          first_quartile = quantile(count, 0.25, na.rm = TRUE),
          median = median(count, na.rm = TRUE),
          third_quartile = quantile(count, 0.75, na.rm = TRUE),
          max = max(count, na.rm = TRUE))
```

```
## # A tibble: 219 x 6
##    country               min first_quartile median third_quartile   max
##    <chr>               <dbl>          <dbl>  <dbl>          <dbl> <dbl>
##  1 Afghanistan             0            139    419           772.  2449
##  2 Albania                 0              2     10            19     43
##  3 Algeria                 0            243.   378.          825.  1982
##  4 American Samoa          0              0      0             0      2
##  5 Andorra                 0              0      0             0      6
##  6 Angola                  0            386.   684.         1592.  3792
##  7 Anguilla                0              0      0             0      1
##  8 Antigua and Barbuda     0              0      0             0      3
##  9 Argentina               1            130    294.          466.   682
## 10 Armenia                 0              2     11            36    170
## # ... with 209 more rows
```

Likewise, the behaviour of `filter` changes so rows are selected **within** each group. You could choose the rows for each country where the count is equal to the country's maximum count value.

Copy and run

Copy and run the following code chunk to do this.

```
max_cases <- filter(by_country, count == max(count, na.rm = TRUE))
max_cases
```

```
## # A tibble: 364 x 7
## # Groups:   country [219]
##    country                           iso3   year type   sex   age_group count
##    <chr>                             <chr> <dbl> <chr>  <chr> <chr>     <dbl>
##  1 Bonaire, Saint Eustatius and Saba BES    2010 new_sp m     04            0
##  2 Bonaire, Saint Eustatius and Saba BES    2011 new_sp m     04            0
##  3 Bonaire, Saint Eustatius and Saba BES    2012 new_sp m     04            0
##  4 Tokelau                           TKL    2007 new_sp m     04            0
##  5 Tokelau                           TKL    2010 new_sp m     04            0
##  6 Tuvalu                            TUV    2006 new_sp m     04            7
##  7 Bonaire, Saint Eustatius and Saba BES    2010 new_sp m     514           0
##  8 Bonaire, Saint Eustatius and Saba BES    2011 new_sp m     514           0
##  9 Bonaire, Saint Eustatius and Saba BES    2012 new_sp m     514           0
## 10 Tokelau                           TKL    2007 new_sp m     514           0
## # ... with 354 more rows
```

It is important to note that the result is still grouped. If you want to remove the grouping use `ungroup` .

```
max_cases <- ungroup(max_cases)
```

Multiple groups can be specified using a comma separated list. Copy and run the following code chunk in RStudio:

```
by_year_agegroup <- group_by(tb_long, year, age_group)
```

Finally, `group_by()` changes the behaviour of `mutate` so columns are created that contain computed values within each group. For example, you could compute the total counts within each year and age group across all countries. **This is useful if you want to normalise a column**.

Try it with the following code chunk.

```
yearly_totals <- mutate(
  by_year_agegroup,
  total_count = sum(count, na.rm = TRUE),
  prop_cases = if_else(is.na(count), 0, count / total_count))
```

Count and tally observations

The `dplyr` package also has convenient functions for counting and tallying observations that are equivalent to using `group_by` and `summarise`.

For example, to count the observations within a group, you can use `count()`. Try it with the following code chunk:

```
count(tb_long, country)
```

```
## # A tibble: 219 x 2
##    country                 n
##    <chr>               <int>
##  1 Afghanistan           740
##  2 Albania               740
##  3 Algeria               740
##  4 American Samoa        740
##  5 Andorra               740
##  6 Angola                740
##  7 Anguilla              740
##  8 Antigua and Barbuda   740
##  9 Argentina             740
## 10 Armenia               740
## # ... with 209 more rows
```

```
count(tb_long, year, age_group)
```

```
## # A tibble: 370 x 3
##     year age_group     n
##    <dbl> <chr>     <int>
##  1  1980 014         424
##  2  1980 04          424
##  3  1980 1524        424
##  4  1980 2534        424
##  5  1980 3544        424
##  6  1980 4554        424
##  7  1980 514         424
##  8  1980 5564        424
##  9  1980 65          424
## 10  1980 u           424
## # ... with 360 more rows
```

Perform a weighted sum of a variable

You could also use another column to perform a weighted sum of a variable. These values are equivalent to the variable **`total_count`** in the data frame **`yearly_totals`** . Try doing this with the following code chunk:

```
count(tb_long, year, age_group, wt = count)
```

```
## # A tibble: 370 x 3
##     year age_group     n
##    <dbl> <chr>     <dbl>
##  1  1980 014          30
##  2  1980 04            0
##  3  1980 1524        121
##  4  1980 2534        126
##  5  1980 3544        118
##  6  1980 4554        133
##  7  1980 514           0
##  8  1980 5564        141
##  9  1980 65          290
## 10  1980 u             0
## # ... with 360 more rows
```

Tell us how you went

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

Were you able to perform a weighted sum of a variable, count and tally observations that are equivalent, or specify multiple groups?

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