The verb `select` is for choosing columns by name and/or position.

Consider the following code chunk:

```
select(tb_long, country, year, count)
```

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

In its most general form, the code takes a comma-separated list of the variables, by name.

```
select(tb_long, sex, age_group, everything())
```

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

You can use functions like **`starts_with()`**, **`ends_with()`**, **`contains()`**, **`matches()`**, **`num_range()`**, **`one_of()`** or **`everything()`** to text-match the names (not the values) then rearrange the order of columns, or select variables explicitly by name.

```
select(tb_long, sex:count)
```

```
## # A tibble: 157,820 x 3
##    sex   age_group count
##    <chr> <chr>     <dbl>
##  1 m     04           NA
##  2 m     04           NA
##  3 m     04           NA
##  4 m     04           NA
##  5 m     04           NA
##  6 m     04           NA
##  7 m     04           NA
##  8 m     04           NA
##  9 m     04           NA
## 10 m     04           NA
## # ... with 157,810 more rows
```

You can use `:` to choose variables in order of the columns.

```
select(tb_long, c(1, 4, 6))
```

```
## # A tibble: 157,820 x 3
##    country     type   age_group
##    <chr>       <chr>  <chr>
##  1 Afghanistan new_sp 04
##  2 Afghanistan new_sp 04
##  3 Afghanistan new_sp 04
##  4 Afghanistan new_sp 04
##  5 Afghanistan new_sp 04
##  6 Afghanistan new_sp 04
##  7 Afghanistan new_sp 04
##  8 Afghanistan new_sp 04
##  9 Afghanistan new_sp 04
## 10 Afghanistan new_sp 04
## # ... with 157,810 more rows
```

You can select by the position of the column using a number.

```
select(tb_long, -iso3)
```

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

You can drop variables by prefixing the name with `-`.

Piping with `filter` and `select`

Just like the **`filter`** verb, you can use the pipe operator to select variables. Below are some examples that use sequences of pipes to filter observations and select variables:

```
# Take tb_long and filter for only the country Australia then select all variables except iso
3
tb_long %>%
  filter(country == "Australia") %>%
  select(-iso3)
```

Give it a go!

Continue to develop your skills with **`select`** by making your way through the following exercise.

Examine the following code chunk - what do you think would be the result from running this code?

```
select(tb_long, contains("count"))
```

Is the result from the following code chunk differ from the previous code chunk?

```
select(tb_long, contains("COUNT"))
```

Can you change an argument to **`contains()`** in the latter code chunk so the results are different? If you are having trouble, look up the help file for **`contains()`**.

Tell us how you went

Within the **Comments**, share with other learners your findings from the exercise, and one or more of the following:

- **How many different ways can you come up with for selecting the columns year, age_group, sex and count in the `tb_long` data set?**
- **What happens if you accidentally select the same variable twice?**
- **How can you use `select` to rename a column?**

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