Linear regression in R

```
library(tidyverse)
library(openintro)
```

`lm()` function

The function that will obtain the coefficients for the least-squares line is the `lm()` function. The syntax is as follows:

`lm(response ~ explanatory, data)`

Example 1: `cherry` data with continuous x

Let's once again consider the `cherry` data from `openintro`, where we want to fit the model

$$\text{volume} = \beta_0 + \beta_1 \times \text{diameter} + \epsilon$$

What does this look like in `R`?

```
lm(volume ~ diam, data = cherry)
```

Note that the variables have to be spelled as they appear in the data frame!

The output from this line of code is:

```

Call:
lm(formula = volume ~ diam, data = cherry)

Coefficients:
(Intercept)         diam
    -36.943        5.066
```

This isn't the most informative of output, so what we will do is use an additional function called `summary()` that will give us much more information!

We will first store the output from `lm()` as a variable called `cherry_lm`:

```
cherry_lm <- lm(volume ~ diam, data = cherry)
```

Then we will use the `summary()` function and pass in the linear model:

```r
summary(cherry_lm)
```

```

Call:
lm(formula = volume ~ diam, data = cherry)

Residuals:
   Min     1Q Median     3Q    Max 
-8.065 -3.107  0.152  3.495  9.587 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -36.9435     3.3651  -10.98 7.62e-12 ***
diam          5.0659     0.2474   20.48  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.252 on 29 degrees of freedom
Multiple R-squared:  0.9353,	Adjusted R-squared:  0.9331 
F-statistic: 419.4 on 1 and 29 DF,  p-value: < 2.2e-16
```

There's a lot more information here! We can now see the b_0 and b_1 estimates, along with some extra information.

In particular, the "Multiple R-squared" quantity is the coefficient of determination R^2!

Example 2: `possum` data with categorical x

The nice thing about `lm()` is that it will automatically convert categorical variables to indicator variables!

Let's re-visit the `possum` model:

$$\text{tail length} = \beta_0 + \beta_1 \times \text{pop-other} + \epsilon$$

$$\text{pop-other} = \begin{cases} 0 & \text{if pop} = \text{Vic} \\ 1 & \text{if pop} = \text{other} \end{cases}$$

We can use the `lm()` function just as before:

```r
possum_lm <- lm(tail_l ~ pop, data = possum)
```

Sometimes we just want the coefficients. The `coef()` function will output the coefficients as a vector. These can be nice for reproducibility and in-line code:

```r
coef(possum_lm)
```

```
(Intercept)    popother 
  35.934783    1.927286 
```

factors in `R`

How does `lm()` choose which level should be the base level?

1. Pre-specified as **factor**: In `R`, variables can be coded as "factor" variables where there is a specific numeric ordering under the hood. How can we tell? Using the `str()` function, we can find the data structure of a given variable:

   ```
   str(possum$pop)
   ```

   ```
    Factor w/ 2 levels "Vic","other": 1 1 1 1 1 1 1 1 1 1 ...
   ```

   We can see that the `pop` variable has two levels, and the order goes "Vic" then "other". So "Vic" is taken as the base level.

   Tip

   If you want a different base level, we can change it using `mutate()`!

   ```
   possum |>
     mutate(pop = factor(pop, levels = c("other", "Vic"))) |>
     pull(pop) |>
     str()
   ```

   ```
    Factor w/ 2 levels "other","Vic": 2 2 2 2 2 2 2 2 2 2 ...
   ```

2. Not-specified: if your categorical variable is coded as a character/string variable and *not* a factor, the default base level is the first level in alphabetic order.

   Tip

   If you don't like this behavior, you can `mutate()` any variable to be a factor variable.

   ```
   data.frame(fruit = c("apple", "kiwi")) |>
     mutate(fruit_factor = factor(fruit, levels = c("kiwi", "apple"))) |>
     pull(fruit_factor) |>
     str()
   ```

   ```
    Factor w/ 2 levels "kiwi","apple": 2 1
   ```