# Functions, continued

# Adding another argument

Let's say we plan to be using our `new_mean()` function to calculate proportions (i.e., the mean of a binary variable). Sometimes we'll want to report them as as percentage by multiplying the proportion by 100.

Let's name our new function `prop()`. We'll use the same structure as we did with `new_mean()`.

```
1 prop <- function(x) {
2   n <- length(x)
3   mean_val <- sum(x) / n
```

# Testing the code

Now we'll want to test on a vector of 1's and 0's.

```
1  x <- c(0, 1, 1)
```

To calculate the proportion and turn it into a percentage, we'll just multiply the mean by 100.

```
1  multiplier <- 100
2  multiplier * sum(x) / length
```

```
[1] 66.66667
```

# Testing the code

We want to give users the option to choose between a proportion and a percentage. So we'll add an argument `multiplier`. When we want to just return the proportion, we can just set `multiplier` to be 1.

```
1 multiplier <- 1
2 multiplier * sum(x) / length
```

```
[1] 0.6666667
```

```
1 multiplier <- 100
2 multiplier * sum(x) / length
```

# Adding another argument

If we add `multiplier` as an argument, we can refer to it in the function body.

```
1 prop <- function(x, multipli
2   n <- length(x)
3   mean_val <- multiplier * s
4   return(mean_val)
5 }
```

# Adding another argument

Now we can test:

```
prop(x = c(1, 0, 1, 0), multiplier = 1)
```

```
[1] 0.5
```

```
prop(x = c(1, 0, 1, 0), multiplier = 100)
```

```
[1] 50
```

# Making a default argument

Since we don't want users to have to specify `multiplier = 1` every time they just want a proportion, we can set it as a **default**.

```
prop <- function(x, multiplier = 1) {
  n <- length(x)
  mean_val <- multiplier * sum(x) / n
  return(mean_val)
}
```

Now we only need to specify that argument if we want a percentage.

```
prop(x = c(0, 1, 1, 1))
```

```
[1] 0.75
```

```
prop(x = c(0, 1, 1, 1), multiplier = 100)
```

```
[1] 75
```

# Caveats

- This is obviously not the best way to write this function!
- For example, it will still work if `x = c(123, 593, -192)`…. but it certainly won't give you a proportion or a percentage!
- We could also put `multiplier =` *any number*, and we'll just be multiplying the answer by that number – this is essentially meaningless.
- We also haven't done any checking to see whether the user is even entering numbers! We could put in better error messages so users don't just get an R default error message if they do something wrong.

```
1 prop(x = c("blah", "blah", "blah"))
```