Week 5 Module 2: Sample Counts and Proportions

CSCI E-5a: Programming in R

Let's clear the global environment:

```r
rm( list = ls() )
```

Module Overview and Learning Outcomes

Hello! And welcome to Module 2: Sample Counts and Proportions.

In this module, we'll learn how to calculate sample counts and sample proportions.

- In section 1, we'll learn how the `sum()` and `mean()` functions work when the input argument is a logical vector.
- In section 2, we'll use the `sum()` and `mean()` functions along with vectorized comparison operations to calculate sample counts and proportions.

At the end of this module, you'll be able to:

- Explain how R implements the `sum()` and `mean()` functions when using logical values as the input argument.
- Calculate sample counts and sample proportions using the `sum()` and `mean()` functions along with vectorized comparison operations.

There are no new built-in R functions in this module.

All right! Let's get started with learning how to summarize logical vectors.

Section 1: Summary statistics for logical Values

> **Main Idea:** *We can summarize logical vectors*

In this section, we'll learn how the `sum()` and `mean()` functions work when the input argument is a logical vector.

Before we dive in, let me make a quick remark.

This is a short video, but the topics are so important that I wanted to emphasize them by giving them their own dedicated module.

Don't let the brevity of this presentation fool you – the techniques that we'll explore are very important, and we'll use them throughout the rest of the course, so it's important that you master them.

We've seen how to use the `sum()` and `mean()` functions to summarize numeric values.

Although it doesn't really make strict mathematical sense, we can also use the `sum()` and `mean()` functions with logical values.

When we use the `sum()` and `mean()` functions with logical vectors, R converts the `TRUE` values to 1, and the `FALSE` values to 0.

Then R performs the usual mathematical calculation on these transformed values.

Thus, when using a vector of logical values, the `sum()` function returns the total number of elements that have the value `TRUE`.

For instance, here is our `first.logical.vector` again:

```
first.logical.vector <-
    c(TRUE, TRUE, FALSE, TRUE, FALSE)
```

Notice that there are 3 elements in this vector that have the value `TRUE`.

Now let's call the `sum()` function with this vector as the input argument:

```
sum( first.logical.vector )
```

```
## [1] 3
```

What R does here is to first convert each `TRUE` value to the number 1 and each `FALSE` value to the number 0 and then add these up to obtain the total.

The result is that when we call the `sum()` function with a logical vector, it will count the total number of the elements in the vector that are `TRUE`.

Similarly, when using a vector of logical values, the `mean()` function returns the proportion of elements in the vector that have the value `TRUE`.

For instance, `first.logical.vector` has a total of 5 elements, and since 3 of these have the value `TRUE` that means that 3/5 or 60% of the elements of the vector have the value `TRUE`.

Now let's run the `mean()` function on this vector:

```
mean( first.logical.vector )
```

```
## [1] 0.6
```

Once again, R first converts each `TRUE` value to the number 1 and each `FALSE` value to the number 0.

It then calculates the mean of the converted values, which will be the proportion of values that are equal to 1.

The result is that when we call the `mean()` function with a logical vector, it will return the relative proportion of the elements in the vector that are `TRUE`.

So that's how to use the `sum()` and `mean()` function to summarize the values in a logical vector.

Now let's see how to calculate sample counts and proportions.

Exercise 2.1: Sample count and proportion

Here's the vector `second.logical.vector`:

```
second.logical.vector <-
    c(TRUE, FALSE, TRUE, TRUE, FALSE)
```

Using the `sum()` function, determine the total number of elements of `second.logical.vector` that have the value `TRUE`.

Using the `mean()` function, determine the proportion of elements of `second.logical.vector` that have the value `TRUE`.

Solution

Section 2: Calculating sample counts and proportions

> **Main Idea:** *We can calculate sample counts and proportions*

In this section, we'll use the `sum()` and `mean()` functions in conjunction with vectorized comparisons to calculate sample counts and proportions.

This is a very important computational gesture, and that's why I'm creating a dedicated module for this.

The "sample count" of an event is the total number of elements in a vector that satisfy some condition.

For instance, let's consider this numeric vector:

```
numeric.vector <-
    c( 5, 7, 2, 4, 1, 6, 9, 8, 0, 3 )
```

How many elements in this vector are strictly less than 3?

We can first construct a logical vector using a vectorized comparison operation:

```
logical.vector <-
    numeric.vector < 3

logical.vector
```

```
##  [1] FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE
```

Make sure you understand how the values in `logical.vector` are determined!

Now we can use the `sum()` function to count the number of `TRUE` values in the `logical.vector`:

```
sum( logical.vector )
```

```
## [1] 3
```

Remember, an element of `logical.vector` has the value `TRUE` if and only if the corresponding element of `numeric.vector` is strictly less than 3.

So by counting the total number of elements of `logical.vector` that have the value `TRUE`, we are also counting the total number of elements of `numeric.vector` that are strictly less than 3.

You should manually check `numeric.vector` to make sure that there are exactly 3 elements that are strictly less than 3. (There are.)

```
numeric.vector
```

```
##  [1] 5 7 2 4 1 6 9 8 0 3
```

We can put all of this into one line of code:

```
sum( numeric.vector < 3 )
```

```
## [1] 3
```

If you've had any prior programming experience, stop for a moment to think about this expression.

It's very weird if you come from a more traditional background, because it's strange to use a logical comparison as an input argument for a numeric function like `sum()`.

In order to make sense of this code, you have to have a strong understanding of vectorized operations and logical values in R.

Let's try this again.

The "sample proportion" of an event is the proportion of elements in a vector that satisfy some condition.

What proportion of elements of `numeric.vector` are greater than or equal to 6?

Again, let's first construct the logical vector:

```
logical.vector <-
    numeric.vector >= 6

logical.vector
```

```
##  [1] FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE
```

Make sure you understand how these values are determined!

Now we can call the `mean()` function with this logical vector:

```
mean( logical.vector )
```

```
## [1] 0.4
```

There are 4 elements that had a value greater than or equal to 6, and the total number of elements of `numeric.vector` is 10, so the proportion of elements of `numeric.vector` that are greater than or equal to 6 is 4/10, or 40%.

Again, we can put this into a single line of code:

```
mean( numeric.vector >= 6 )
```

```
## [1] 0.4
```

We don't even have to use numeric vectors for these methods.

Let's consider this vector:

```r
character.vector <-
    c( "To", "be", "or", "not", "to", "be" )
```

How many elements of this vector have the value “be”?

```r
sum( character.vector == "be" )
```

```
## [1] 2
```

What proportion of the elements of this vector are equal to “be”?

```r
mean( character.vector == "be" )
```

```
## [1] 0.3333333
```

The ideas that we’ve explored here are very important, and we’ll use them throughout the rest of the course, so make sure that you understand them.

So that’s how to calculate sample counts and proportions.

Now let’s review what we’ve learned in this module.

Exercise 2.2: Sample count and proportion

Consider this numeric vector:

```r
second.numeric.vector <-
    c( -8.4, 7.3, 2.5, 0.1, -4.4, 6.3, 1.9 )
```

How many elements of this vector are greater than or equal to 4.2?

What proportion of elements of this vector are strictly less than 1.8?

Solution

Module Review

In this module, we learned how to calculate sample counts and sample proportions.

- In section 1, we learned how the `sum()` and `mean()` functions work when the input argument is a logical vector.
- In section 2, we saw how to use the `sum()` and `mean()` functions along with vectorized comparison operations to calculate sample counts and proportions.

Now that you’ve completed this module, you should be able to:

- Explain how R implements the `sum()` and `mean()` functions when using logical values as the input argument.
- Calculate sample counts and sample proportions using the `sum()` and `mean()` functions along with vectorized comparison operations.

There were no new built-in R functions in this module.

All right! That’s it for Module 2: Sample Counts and Proportions.

Now let’s move on to Module 3: Logical Indexing.

Solutions to the Exercises

Exercise 2.1: Sample count and proportion

Here's the vector `second.logical.vector`:

```
second.logical.vector <-
    c(TRUE, FALSE, TRUE, TRUE, FALSE)
```

Using the `sum()` function, determine the number of elements of `second.logical.vector` that have the value `TRUE`.

Using the `mean()` function, determine the proportion of elements of `second.logical.vector` that have the value `TRUE`.

Solution

To count the number of elements of the vector that have the value `TRUE`, we use the `sum()` function:

```
sum( second.logical.vector )
```

```
## [1] 3
```

To determine the proportion of elements of the vector that have the value `TRUE`, we use the `mean()` function:

```
mean( second.logical.vector )
```

```
## [1] 0.6
```

Exercise 2.2: Sample count and proportion

Consider this numeric vector:

```
second.numeric.vector <-
    c( -8.4, 7.3, 2.5, 0.1, -4.4, 6.3, 1.9 )
```

How many elements of this vector are greater than or equal to 4.2?

What proportion of elements of this vector are strictly less than 1.8?

Solution

How many elements of this vector are greater than or equal to 4.2?

```
sum( second.numeric.vector >= 4.2 )
```

```
## [1] 2
```

What proportion of elements of this vector are strictly less than 1.8?

```
mean( second.numeric.vector < 1.8 )
```

```
## [1] 0.4285714
```