Week 3 Module 4: Vectorized Operations

CSCI E-5a: Programming in R

Let's clear the global computing environment:

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

Module Overview and Learning Outcomes

Hello! And welcome to Module 4: Vectorized Operations.

In this module, we'll explore the concept of *vectorized operations*, which enable us to express complicated calculations very simply.

- In section 1, we'll learn about the basic concept of vectorized operations, and use vectorized operations on a single vector.
- In section 2, we'll see how to use vectorized operations on two vectors.
- In section 3, we'll investigate how to map a function over a vector.
- In section 4, we'll think about the relationship between dot products and vectorized operations.

When you've completed this module, you'll be able to:

- Explain the concept of vectorized operations.
- Perform vectorized operations on a single vector.
- Perform vectorized operations with two vectors.
- Map a function over a vector.
- Explain how dot products are related to vectorized operations.

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

All right! Let's get started by learning about the basic concept of vectorized operations.

Section 1: Basic Vectorized Operations

Main Idea: *We can understand the basic concept of vectorized operations*

In this section, we'll learn about the basic concept of vectorized operations, and use vectorized operations on a single vector.

One of the most distinctive features of R is the concept of *vectorized* operations.

Paradoxically, this feature is so distinctive that if you've had previous experience with another programming language such as C, BASIC, or Java you might find it a little challenging to adapt to this novel approach.

The idea itself is very simple, but it's remarkable how powerful it can be.

The fundamental concept of vectorization is that operations on vectors can act in an element-wise or component-wise manner.

Arithmetic operations on vectors are evaluated component-wise.

This means that we calculate the final result by applying the operation to each of the individual components of the vectors.

Let's make a simple numeric vector:

```
first.numeric.vector <-
    c( 2, 5, 4, 8, 1)
```

What would be the result of adding the number 1 to `first.numeric.vector`?

It's weird to add a number to a vector, because they have different dimensions.

What could such an expression mean?

The answer is that we perform the operation in a "component-wise" or "element-wise" manner.

That means that we add 1 to each of the elements of the vector.

Now let's evaluate this expression:

```
first.numeric.vector + 1
```

```
## [1] 3 6 5 9 2
```

Notice that this vector is the `first.numeric.vector` with the value 1 added to each element.

We can also do this with multiplication.

Let's multiply `first.numeric.vector` by the number 2.

Before we run the R code, can you predict what the result will be?

Let's try it out:

```
first.numeric.vector * 2
```

```
## [1]  4 10  8 16  2
```

Do you see what happened?

Each of the elements or components of the vector was multiplied by the value 2.

That's what we mean by the expression "component-wise" or "element-wise".

We can also use a vectorized operation to apply exponentiation.

What happens if we raise `first.numeric.vector` to the second power?

In other words, what happens if we try to square the vector?

Again, try to figure this out for yourself first before we run the code.

OK, here goes:

```
first.numeric.vector^2
```

```
## [1]  4 25 16 64  1
```

The result of this operation is a vector consisting of the squares of each of the elements of `first.numeric.vector`.

Make sure you understand how these operations work, and how each of the values of the vector are computed.

So that's the basic concept of vectorized operations.

Now let's see how to perform vectorized operations with two vectors.

Exercise 4.1: Temperature Conversion

To convert a temperature in the Fahrenheit scale to a temperature in the Celsius scale, we first subtract 32 from the Fahrenheit temperature, and then multiply by 5/9.

Thus, to convert 86 degrees Fahrenheit to Celsius, we first subtract 32, giving 54, and then multiply by 5/9, which gives 30.

```
5/9 * (86 - 32)
```

```
## [1] 30
```

So 86 degrees Fahrenheit is the same as 30 degrees Celsius.

Here is a series of Fahrenheit temperatures:

$$75,\ 82,\ 67,\ 51,\ 77,\ 93,\ 84$$

Construct a vector to store these Fahrenheit temperatures, and then use vectorized operations to convert them to Celsius. Display the resulting temperatures with no decimal places.

Solution

Section 2: Two Vectors

Main Idea: *We can perform vectorized operations with two vectors*

In this section, we'll see how to use vectorized operations on two vectors.

In the previous section we performed vectorized operations by adding a single number to a vector or multiplying a vector by a single number.

Often, we can have two vectors where the elements of one vector are somehow associated with the elements of the other vector.

For instance, consider the table of stock transactions from our VWAP example from the previous lecture:

Transaction	Number of shares	Price per share
1	500	$44.00
2	200	$43.00
3	300	$42.00

Notice that each element in the column representing the number of shares sold is associated with the corresponding element of the column with the information on the share price, because they are both from the same transaction.

Thus, the first entry for the number of shares is 500, and the first entry for the share price is $44.00, and these are associated because they were the values for the first transaction.

When we have coordinated vectors like these, we often perform calculations by adding the two vectors together, or even multiplying the two vectors together.

As always, the key is to remember that vectorized operations are element-wise: R performs the operation on each pair of elements.

For now, let's look at a simple example so that we can understand the mechanics of these operations.

Recall our first vector:

```
first.numeric.vector
```

```
## [1] 2 5 4 8 1
```

Let's create a second vector:

```
second.numeric.vector <-
    c(-3, 0, 12, 7, 2)
```

What do you think will happen if we add `first.numeric.vector` and `second.numeric.vector`?

Think about this, and try to come up with a plausible answer.

If we add the two vectors, then R will perform a vectorized operation, and the final result will be a vector where each component is the sum of the corresponding components of `first.numeric.vector` and `second.numeric.vector`:

```
first.numeric.vector + second.numeric.vector
```

```
## [1] -1  5 16 15  3
```

You should check each component and make sure you know how it is computed.

How about multiplication?

What do you think happens if we multiply two vectors together?

Answer: R will perform the operation on an element-wise basis, so that the elements of the resulting vector are products of the corresponding elements of the two vectors.

Let's check this:

```
first.numeric.vector * second.numeric.vector
```

```
## [1] -6  0 48 56  2
```

Again, make sure that you understand how each individual value is calculated.

So that's how to perform vectorized operations on two vectors.

Now let's see how to map a function over a vector.

Exercise 4.2: Two vectors

Here is a table of values for two variables:

X	Y
4	1
7	2
3	9
6	1
5	8

Construct vectors to represent the two variables. Then calculate the sum of these two vectors, and the product of these two vectors. Try to predict what the sum and product will be *before* you run your code, and then check to see if you got it right.

Solution

Section 3: Mapping a Function Over a Vector

Main Idea: *We can map a function over a vector*

In this section, we'll investigate how to map a function over a vector.

If we have a numeric function, we can evaluate that function over multiple input arguments by passing it a vector with the arguments.

For instance, suppose we want to construct a vector consisting of the value of $\exp(x)$ for the values 1, 2, 3, 4, and 5.

Then one way to do this is to use the `c()` function and explicitly list each value:

```
c( exp(1), exp(2), exp(3), exp(4), exp(5) )
```

```
## [1]   2.718282   7.389056  20.085537  54.598150 148.413159
```

But there's another way to do this.

In the vectorized approach, we express the sequence 1, 2, 3, 4, and 5 as a vector, and then pass this as the input argument to the `exp()` function, which will then operate on each of the elements of the input argument:

```
input.vector <- c(1, 2, 3, 4, 5)

exp( input.vector )
```

```
## [1]   2.718282   7.389056  20.085537  54.598150 148.413159
```

```
xyz <- 1:5

exp.vector <- exp( xyz )
```

We can actually make this a little more compressed.

First, note that we can use the colon operator instead of having to explicitly list each element with the `c()` function.

Second, we can just use the vector directly, without having to first store it in a variable:

```
exp( 1:5 )
```

```
## [1]   2.718282   7.389056  20.085537  54.598150 148.413159
```

This is exactly the same as before.

So that's how to map a function over a vector.

Now let's study the concept of a dot product.

Section 4: Dot Products

Main Idea: *We can understand the relationship between dot products and vectorized operations*

In this section, we'll think about the relationship between dot products and vectorized operations.

To calculate the *dot product* of two vectors, we take pairs of corresponding elements and multiply them, and then add all the terms together.

There is a close relationship between dot products and vectorized operations.

We can use a vectorized operation to multiply pairs of corresponding elements in two vectors, and then we can use the `sum()` function to add up the terms.

Thus, we can calculate a dot product very compactly using vectorized operations.

Let's see a concrete example.

Let's suppose we have two columns of numbers:

X	Y
4	1
9	3
2	6

First let's calculate the dot product of `X` and `Y`:

```
(4 * 1) + (9 * 3) + (2 * 6)
```

```
## [1] 43
```

Now let's do this using vectorized operations.

First, we need to represent the columns of data as vectors:

```
x.vector <-
    c(4, 9, 2)

y.vector <-
    c(1, 3, 6)
```

When we multiply the two vectors, we form products of corresponding elements:

```
x.vector * y.vector
```

```
## [1]  4 27 12
```

Finally, we can add these terms together to obtain the dot product:

```
sum( x.vector * y.vector )
```

```
## [1] 43
```

So here's the general strategy for computing dot products:

- Represent the values of the two variables as vectors.
- Multiply the two vectors together.
- Add up the values with the `sum()` function.

This is a very powerful feature of R, because it enables us to compactly express a very common calculation.

It also scales efficiently: when we explicitly calculated everything by hand, the problem became more difficult as the number of values increased, but when we use vectors the code is the same regardless of the number of values.

So that's the relationship between dot products and vectorized operations.

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

Exercise 4.3: Dot Product

In Exercise 4.2 we had a table of values for two variables:

X	Y
4	1
7	2
3	9
6	1
5	8

Represent X and Y as vectors, and then use vectorized operations to calculate the dot product of X and Y.

Solution

Module Review

In this module, we explored the concept of *vectorized operations*, which enable us to express complicated calculations very simply.

- In section 1, we learned about the basic concept of vectorized operations, and used vectorized operations on a single vector.
- In section 2, we saw how to use vectorized operations on two vectors.
- In section 3, we investigated how to map a function over a vector.
- In section 4, we thought about the relationship between dot products and vectorized operations.

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

- Explain the concept of vectorized operations.
- Perform vectorized operations on a single vector.
- Perform vectorized operations with two vectors.
- Map a function over a vector.
- Explain how dot products are related to vectorized operations.

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

All right! That's it for Module 4: Vectorized Operations.

Now let's move on to Module 5.

Solutions to the Exercises

Exercise 4.1: Temperature Conversion

To convert a temperature in the Fahrenheit scale to a temperature in the Celsius scale, we first subtract 32 from the Fahrenheit temperature, and then multiply by 5/9.

Thus, to convert 86 degrees Fahrenheit to Celsius, we first subtract 32, giving 54, and then multiply by 5/9, which gives 30.

```
5/9 * (86 - 32)
```

```
## [1] 30
```

So 86 degrees Fahrenheit is the same as 30 degrees Celsisus.

Here is a series of Fahrenheit temperatures:

$$75,\ 82,\ 67,\ 51,\ 77,\ 93,\ 84$$

Construct a vector to store these Fahrenheit temperatures, and then use vectorized operations to convert them to Celsius. Display the resulting temperatures with no decimal places.

Solution

Here's the vector:

```
fahrenheit.temperature.vector <-
    c( 75, 82, 67, 51, 77, 93, 84)
```

Here's the vector with the Celsius temperatures:

```
celsius.temperature.vector <-
    (fahrenheit.temperature.vector - 32) * (5/9)
```

Now we can display this with no decimal places:

```
cat(
    "Celsius temperatures:",
    formatC(
        celsius.temperature.vector,
        format = "f",
        digits = 0
    )
)
```

```
## Celsius temperatures: 24 28 19 11 25 34 29
```

Exercise 4.2: Two vectors

Here is a table of values for two variables:

X	Y
4	1
7	2
3	9
6	1
5	8

Construct vectors to represent the two variables. Then calculate the sum of these two vectors, and the product of these two vectors. Try to predict what the sum and product will be *before* you run your code, and then check to see if you got it right.

Solution

Here is the vector for the X values:

```
x.vector <-
    c( 4, 7, 3, 6, 5)
```

Here is the vector for the Y values:

```
y.vector <-
    c( 1, 2, 9, 1, 8)
```

Here is the sum of the two vectors:

```
x.vector + y.vector
```

```
## [1]  5  9 12  7 13
```

Here is the product of the two vectors:

```
x.vector * y.vector
```

```
## [1]  4 14 27  6 40
```

Exercise 4.3: Dot Product

In Exercise 4.2 we had a table of values for two variables:

X	Y
4	1
7	2
3	9
6	1
5	8

Represent X and Y as vectors, and then use vectorized operations to calculate the dot product of X and Y.

Solution

We've already created the vectors for X and Y, so all we have to do is to calculate their dot product.

```
sum( x.vector * y.vector )
```

```
## [1] 91
```