My Quarto Document

AUTHOR
Benjamin Cook

PUBLISHED
September 18, 2024

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
1 + 1
```

```
[1] 2
```

You can add options to executable code like this

```
[1] 4
```

The `echo: false` option disables the printing of code (only output is displayed).

##———————————————————————————— ## R code for 502 Lab, week 4: Individual Exercises ## Howard Liu ## University of South Carolina ## ————————————————————————————

1. Functions ————————————————————————————

What is a function that displays all the objects currently stored in

the memory? Write it below after deleting the line that reads

“WRITE YOUR ANSWER (code) HERE”, and execute it.

ls() character(0)

Create a new object named x5 that is a number 100.

```
x5 =100
```

Calculate the square root of x5 using the sqrt() function

```
sqrt(x5)
```

Calculate the square root of x5 by raising it to the power of 0.5.

Your numeric answer should be exactly the same as when you used the

sqrt() function. This is because taking the sqaure root of something

is equivalent to raising it to the power of 0.5.

```
x5^0.5
```

Create an object called x6 that is equal to 31.8734.

```
x6 =31.8734
```

Use the round() function to get the value of x6 rounded off to

three decimal places

```
round(x6, 3)
```

Functions floor() and ceiling() can also be used to trim a number

down to an integer: apply both of these functions to x6 and compare

the outputs. Can you guess what these functions do?

```
floor_x6 <- floor(x6)

ceiling_x6 <- ceiling(x6)
```

To find out if your hunch was right, refer to the help file of these

functions. Write a code to open up the help file for the floor function.

```
help(floor)
```

2. Vectors ——————————————————————————————

Create an object called “vec.a” which is a vector consisting of

the numbers, 1, 3, 5, 7. You need to use the c function.

```
vec.a <- c(1, 3, 5, 7)
```

Create a vector called “vec.b” consisting of the numbers, 2, 4, 6, 8.

```
vec.b <- c(2, 4, 6, 8)
```

Subtract vec.b from vec.a

```
result <- vec.b - vec.a
```

Create a new vector called vec.c by multiplying vec.a by vector vec.b

```
vec.c <- vec.a * vec.b
```

Create a new vector called vec.d by taking the square root of each

member of vec.c

```
vec.d <- sqrt(vec.c)
```

What is the third element of the vec.d vector? Find out using

square bracket. Note that since this is a vector, you only need to

provide a single number inside the brackets.

```
vec.d[3]
```

Create a new vector called vec.e consisting of all the integers

from 1 through 100. You should use the seq function, rather than writing

down all the 100 integers individually.

```
vec.e <- seq(1, 100)
```

The mean function calculates the arithmetic mean of the numbers stored

in an object. Using the mean function, calculate the mean of the vec.e vector.

```
mean(vec.e)
```

As we saw in the joint exercise, the sum function calculates the sum of all

the elements in an object. Calculate the sum of the vec.e vector.

```
sum(vec.e)
```

The length function returns the number of elements stored in an object.

Using the length function, find the number of elements stored in the vec.e

vector.

```
length(vec.e)
```

The mean of an object can be obtained by sum(X)/length(X) because

the defininition of the mean is the sum of elements divided by the number of

elements. Now, using the sum and length functions, calculate the mean of

the vec.e vector. Compare the answer with that obtained with the mean function

```
sum(vec.e) / length(vec.e)
```

We have learned that the by argument specifies an increment. For example,

```
seq(from = 0, to = 10, by = 2)
```

This creates a sequence that starts from 0 and ends with 10, and with

an increment of 2.

Now, create a new object called olympic which is a sequence that

starts from 1896 and ends with 2012, with an increment of 4.

```
olympic <- seq(from = 1896, to = 2012, by = 4)
```

How many elements does the olympic vector contain? That is, what is

the length of this vector? Find out by applying a function (not by

manually counting the number of elements).

```
length(olympic)
```

So there are 30 elements in the olympic vector. Display all the

elements contained in the olympic vector. These are the years

where olympic games were (supposed to be) held. Display the

contents of the olympic vector.

olympic

Find out how many olympic games will have been held by the year

2400. Use the length and seq functions.

olympic.2 <- seq(from = 1896, to = 2400, by = 4)

num_olympics <- length(olympic) num_olympics

3. Matrices ——————————————————————————————-

Create a new vector called “v1” consisting of the following numbers:

1, 3, 5, 7, 9, 11

Find out the length of this vector (Don’t count the numbers by hand;

use an appropriate function).

v1 <- c(1, 3, 5, 7, 9, 11)

We will conver this vector into a matrix. That is, we will rearrange this

vector so that it will have two dimensions (rows and columns).

Since this vector has 6 numbers, if we want the matrix to have two

rows, how many columns will there be?

Create a matrix called mat.v using the following command:

matrix(data = v1, nrow = 2)

mat.v <- matrix(data = v1, nrow = 2) mat.v

Take a look at the contens of this matrix.

How many columns are there? 3

Notice how the numbers in vec.v are used to fill up the cells of mat.v.

We can see that R did it “by column”. That is, R first filled up the

first column of mat.v with the first two elements of vec.v, then moved

on to the second and third columns.

You can use the byrow argument to change this. This argument takes

one of two values, TRUE or FALSE (or T or F). That is, we write

matrix(data = v1, nrow = 2, byrow = TRUE)

Now, create an object called mat.w using the command above.

```
mat.w <- matrix(data = v1, nrow = 2, byrow = TRUE)
```

Compare mat.v and mat.w. Do you see that R filled up the cells

“by row” to create the mat.w matrix ?

Many functions in R have arguments that take TRUE or FALSE like

the byrow argument we just used. In most cases, functions have a

default value. In the case of the matrix function, the default

value for the byrow argument is FALSE, meaning that, if you don’t

specify anything, R will automatically sets byrow = FALSE.

Find the number in the second row, second column of mat.w

```
mat.w[2, 2]
```

Find the number in the second row, second column of mat.v

```
mat.v[2, 2]
```

Finally, execute the entire contents of this R file by pressing

Ctrl + A and then pressing Ctrl + Enter.

Make sure that you don’t get any error message. If you get an

error message, it’s probably because you forgot to comment out

something.

End of file