STAT 33B Workbook 1

Yuanrui Zhu (3034615728)

Sep 3, 2020

This workbook is due **Sep 3, 2020** by 11:59pm PT.

The workbook is organized into sections that correspond to the lecture videos for the week. Watch a video, then do the corresponding exercises *before* moving on to the next video.

Workbooks are graded for completeness, so as long as you make a clear effort to solve each problem, you'll get full credit. That said, make sure you understand the concepts here, because they're likely to reappear in homeworks, quizzes, and later lectures.

As you work, write your answers in this notebook. Answer questions with complete sentences, and put code in code chunks. You can make as many new code chunks as you like.

In the notebook, you can run the line of code where the cursor is by pressing `Ctrl` + `Enter` on Windows or `Cmd` + `Enter` on Mac OS X. You can run an entire code chunk by clicking on the green arrow in the upper right corner of the code chunk.

Please do not delete the exercises already in this notebook, because it may interfere with our grading tools.

Data Types

Watch the "Data Types" lecture video.

Exercise 1

In R, if you pass vectors with different lengths to a binary operator, the shorter one will be **recycled**. This means the elements of the shorter vector will be repeated to match the length of the longer vector.

Use the recycling rule to explain what's happening in each of these lines of code:

```r
c(1, 2) - c(3, 4, 5, 6)
```

```
## [1] -2 -2 -4 -4
```

```r
c(20, 30, 40) / 10
```

```
## [1] 2 3 4
```

```r
c(1, 3) + c(0, 0, 0, 0, 0)
```

```
## Warning in c(1, 3) + c(0, 0, 0, 0, 0): longer object length is not a multiple of
## shorter object length
## [1] 1 3 1 3 1
```

YOUR ANSWER GOES HERE: For the first one, 1 , 2 are repeated to form 1, 2, 1, 2. The corresponding value in each place is deducted form one another to get the result of -2, -2, -4, -4. For the second one, the denominator is multiplied three times to get three 10s, which result in 2, 3, 4. For the third one, it is unable for the first vector to multiply an integer number of times to get the second one, which raises an error.

Exercise 2

Run each line in the following code chunk and inspect the result. For each one, state the type and class of the result, and explain why the result has that type.

```r
c(TRUE, "hello", 3, 6)
```

```
## [1] "TRUE"  "hello" "3"     "6"
```

```r
3L + 3i
```

```
## [1] 3+3i
```

```r
c(3L, 4L, 5L) / TRUE
```

```
## [1] 3 4 5
```

YOUR ANSWER GOES HERE: first one: type - character, class - character, all other elements convert to the string since it is the only type they can all convert into.

seoncd one: type - complex, class - complex, since the expression contains "i" which stands for complex number; adding a complex number to an integer gives a complex number

third one: type - double, class - numeric, since the denominator of "TRUE" automatically transforms to 1 and each element is divided by 1 to get 3, 4, 5.

Exercise 3

Another way to create vectors is with the `rep()` function. The `rep()` function creates a vector by replicating a value or vector of values.

1. The first parameter of `rep()` is the thing to replicate. The second parameter, `times`, is the number of times to to replicate. Use `rep()` to make a vector with 10 elements, all equal to 78.

```r
rep(78, 10)
```

```
##  [1] 78 78 78 78 78 78 78 78 78 78
```

2. What happens if you pass a vector as the first argument to `rep()`? Give some examples.

Answer: The vector is repeated times equal to the second element.

```r
rep(c(1, 2, 3), 8)
```

```
##  [1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
```

3. Skim the help file `?rep`. What happens if you pass a vector as the second argument to `rep()`? The help file might seem a bit cryptic, so you'll also need to experiment. Give some examples.

```r
?rep
rep(1, c(2))
```

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

```r
rep(1:4, c(2, 3, 4, 5))
```

```
##  [1] 1 1 2 2 2 3 3 3 3 4 4 4 4 4
```

YOUR ANSWER GOES HERE:

If the first argument we pass in is an integer, and the second argument is a vector having a length not 1, the expression throws an error saying the length does not match. If we write the first argument as something like x:y, with the numbers in it equal to the length of the vector, it will repeat the corresponding numbers in the first arugment with times in the vector.

Exercise 4

Yet another way to create vectors is with the `seq()` function. The `seq()` function creates a vector that contains a sequence of numbers.

Skim the help file `?seq`. Give some examples of creating vectors with the `seq()` function.

YOUR ANSWER GOES HERE:

```r
seq(1, 8)
```

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

Exercise 5

In R, `T` and `F` are shortcuts for `TRUE` and `FALSE`.

1. What happens if you try to assign a value to `TRUE`?
2. What happens if you try to assign a value to `T`?
3. Check that what you observed in #1 and #2 is also true for `FALSE` and `F`. Why might it be safer to use `TRUE` and `FALSE` rather than `T` and `F` in code?

YOUR ANSWER GOES HERE:

```
a = TRUE
a
```

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

```
b = T
b
```

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

```
c = FALSE
c
```

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

```
d = F
d
```

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

Assigning values to TRUE is the same as T, and this also holds true for FALSE and F. Since we may name other variables to T or F (or similar characters), it should be safer to assign it to TRUE and FALSE.

Matrices, Arrays, & Lists

Watch the "Matrices, Arrays, & Lists" lecture video.

Exercise 6

Recall that many of R's functions are vectorized, which means they are applied element-by-element to vectors.

1. What happens if you call a vectorized function on a matrix?
2. What happens if you call a vectorized function on an array?

Give examples to support your answer.

YOUR ANSWER GOES HERE:

```
m = matrix(seq(1, 8), 2)
m / 2
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  0.5  1.5  2.5  3.5
## [2,]  1.0  2.0  3.0  4.0
```

```r
n = array(seq(1, 8), c(2, 2, 2))
n / 2
```

```
## , , 1
##
##      [,1] [,2]
## [1,]  0.5  1.5
## [2,]  1.0  2.0
##
## , , 2
##
##      [,1] [,2]
## [1,]  2.5  3.5
## [2,]  3.0  4.0
```

The function is applied to each element of the matrix or array.

Exercise 7

Suppose we want to multiply a length-2 vector with a 2-by-2 matrix.

What happens if you use `*` to multiply them? What happens if you use `%*%`?

Give some examples that show the difference, including for vectors and matrices of other sizes.

YOUR ANSWER GOES HERE:

```r
matrix(seq(1, 4), 2)
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

```r
c(1, 2) * matrix(seq(1, 4), 2)
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    4    8
```

If I use * to multiply a vector with a matrix, I get the first row multiplied with the first element of the vector, the second row multiplied the second element of the vector, and so on. If I use the %*% it is the same as matrix multiplication.

Exercise 8

The `c()` function combines vectors, but it can also combine lists. Use `list()` to create two lists, and show that `c()` can be used to combine them.

YOUR ANSWER GOES HERE:

```r
l1 = list(1, 2, 3)
l2 = list(4, 5, 6)
c(l1, l2)
```

```
## [[1]]
## [1] 1
## 
## [[2]]
## [1] 2
## 
## [[3]]
## [1] 3
## 
## [[4]]
## [1] 4
## 
## [[5]]
## [1] 5
## 
## [[6]]
## [1] 6
```

Special Values

Watch the "Special Values" lecture video.

Exercise 9

Skim the help file for the `mean()` function.

1. What happens if you call the mean function on a vector that contains missing values? Is there a way to override this behavior?
2. What happens if you call the mean function on a vector that contains `NaN` values or infinite values?

In each case, provide examples to suport your answers.

YOUR ANSWER GOES HERE:

```r
mean(c(1, 2, NA))
```

```
## [1] NA
```

```r
mean(c(1, 2, NaN))
```

```
## [1] NaN
```

```r
mean(c(1, 2, Inf))
```

```
## [1] Inf
```

Calling a function on a vector containing NA gives NA. We can get rid of it by judging whether it is na by is.na. Calling a function on a vector containing NaN or Inf gives NaN or Inf.

Making Comparisons

Watch the “Making Comparisons” lecture video.

Exercise 10

Each of the following lines of code produces a result that, at a glance, you might not expect. Explain the reason for each result.

```
3 == "3"
```

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

```
50 < '6'
```

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

```
isTRUE("TRUE")
```

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

YOUR ANSWER GOES HERE: The first expression converts the string ‘3’ into the integer 3 and make comparison with the left hand 3, which gives true. The second expression make comparison with only the first character, that is ‘5’ and 6, which gives true. The third expression is equivalent to is.logical(x) && length(x) == 1 && !is.na(x) && x, which gives false.

Exercise 11

Suppose you want to check whether any of the values in `c(1, 2, 3)` appear in the vector `c(4, 1, 3, 1)`.

Novice R users often expect they can check with the code:

```
c(1, 2, 3) == c(4, 1, 3, 1)
```

```
## Warning in c(1, 2, 3) == c(4, 1, 3, 1): longer object length is not a multiple
## of shorter object length
```

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

1. Explain why the code above is not correct, and what’s actually happening.
2. The correct way is to use the `%in%` operator. Give some examples of using the `%in%` operator. Recall that you can access its help page with `?"%in%"`.

YOUR ANSWER GOES HERE:

```
c(1, 2, 3) %in% c(4, 1, 3, 1)
```

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

The reason that the expression is not working is that the length does not match in length and cannot adjust in terms of the first question this assignment shows (using multiplication). Even if it can, this expression sees if the corresponding value matches, which is not what we want.

Submitting Your Work

Congratulations, you made it through the first workbook!

You need to submit your work in two places:

- Submit this Rmd file with your edits on bCourses.
- Knit and submit the generated PDF file on Gradescope.