writting Functions in R (Part 1)

Taiwo

3/23/2021

Loops in R

For Loops

A for loop in R can be use to iterate over each element of a vector.

The for loop syntax is given as:

for (i in x) { Code }

Alternatively, or in a single line:

for (i in x) code

In both instances, x could be a vector or a list. For the argument, its means that there will be one iteration of the loop for each component of the vector x.

The code area is where all codes statements are written. For example, the following code uses for loop statement to return the square of each element in a vector.

```
x <- c(1, 2, 10, 20)
for (i in x) {
    print(i^2)
}
```

```
[1] 1
[1] 4
[1] 100
[1] 400
```

Also, the following single line format will return the same output like previous example.

```
x <- c(1, 2, 10, 20)
for (i in x) print(i^2)
```

```
[1] 1
[1] 4
[1] 100
[1] 400
```

In addition, the following example iterate over each element in vector called color to return each element as a character element.

```
colors = c("orange", "brown", "red", "blue")
for (color in colors) {
    print(paste0("color:", color))
}
```

```
[1] "color:orange"
[1] "color:brown"
[1] "color:red"
[1] "color:blue"
```

Nested For Loop

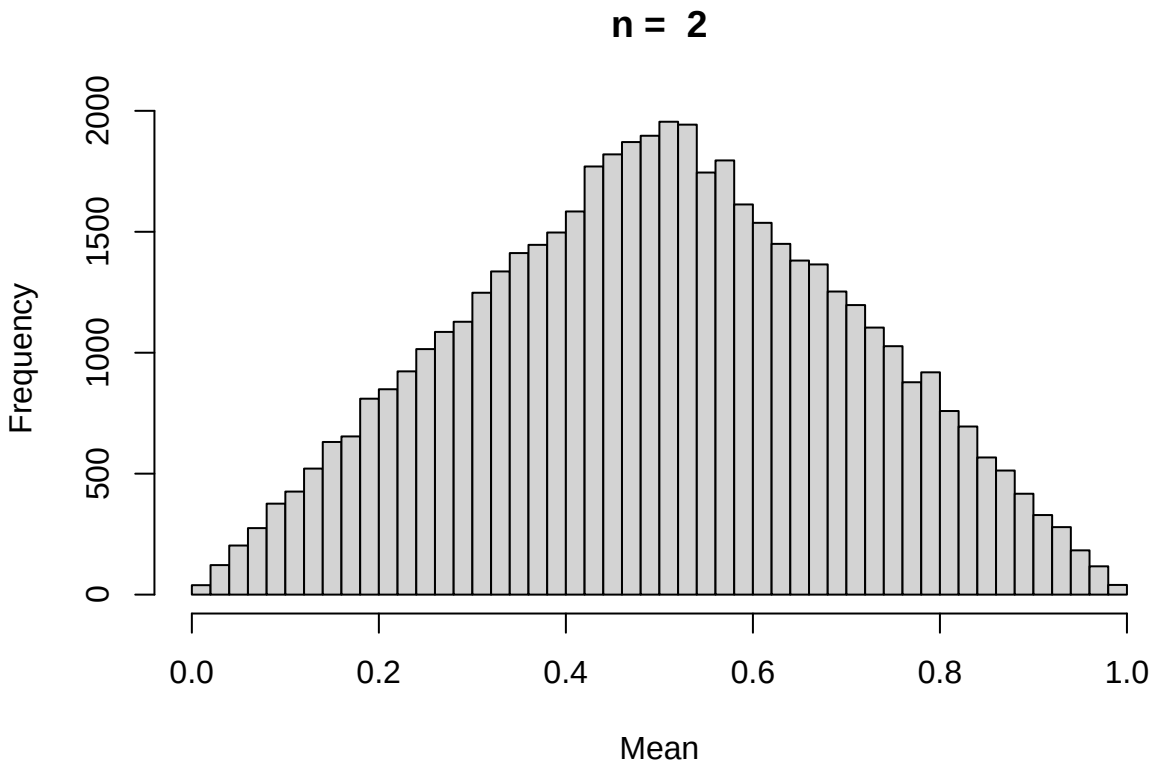
The syntax for the nested for loop is as follows: for (i in list) { Code for(j in list) { Code } }

Suppose you want to know the sample mean of n data points obtained independently of a uniform distribution over the interval (0, 1). You can solve the previous problem theoretically, but we are going to carry out a simulation study. The following code expatiate how this can be done.

```r
set.seed(1)  # Setting a seed for reproducibility
rep <- 50000  # Number of repetitions
n <- 2  #
Mean <- numeric(rep)

for (irep in 1:rep) {
    x <- runif(n)
    Mean[irep] <- mean(x)
}

hist(Mean, breaks = 40, main = paste("n = ", n))
```



While loop

In R programming, while loop is used to loop until a specific condition is met.

The syntax of while loop is given as:

while (test_expression) { statement }

While being evaluated, if the text_expression is True, the statements inside the loop is executed and the process continues otherwise, if the test_expression is evaluated to False, in which case the program is exited.

Note that you can write a while loop in a single line. In this case the program syntax is in this form: while(test_expression) #statement

By a way of illustration, lets consider the following example:

```
i <- 1
while (i <= 10) i <- i + 4
i
```

```
[1] 13
```

From the code snippet above, we assign 1 to i thereafter, calculate the value of new i by adding 4 to the last number before 10 and this is 9. By doing so, the code returns 13.

We should take note of statements like the break, repeat statements. For instance, re-write the previous example with break statement and it will return the same result as before.

```
i <- 1
while (TRUE) {
    i <- i + 4
    if (i > 10)
        break
}
i
```

```
[1] 13
```

The code instructs R to evaluate i while it is less than 10. If i is greater than 10, apply the break statement.

Another example that I am about to discuss apply the repeat statement. For instance, we could use the repeat statement to replace the while statement applied in the previous example to return similar output.

```
i <- 1
repeat {
    i <- i + 4
    if (i > 10)
        break
}
i
```

```
[1] 13
```

Another useful statement is next statement. A next statement is useful when we want to skip the current iteration of a loop without terminating it.

Just like the break statement, the next statement is mostly use with if statement and its syntax is:

if(condition) next

else, for multiple line coding use:

if (condition){ next.

For instance, let say we create a for loop to iterate over a a sequence of integers. Using the next statement, exclude number stated in the if statement.

```
val <- 1:10
for (i in val) {
    if (i == 6) {
        next
    }
    print(i)
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 7
[1] 8
[1] 9
[1] 10
```

Just as explained, the next statement exclude 6 if it is equal to the value stated in the if statement.

Loop Over Nonvector Sets

R does not support iteration over non vector sets, but we there are other methods that we can use to achieve the purpose. These are:

- lapply(), It returns a list of the same length as X, each element is the result generated when applying FUN to the corresponding element of X
- get(), return the value of named object.

Consider two 3 by 2 matrices and we want to apply R's linear regression function lm() to each of them, we can do the following:

```
# Create two matrices
u <- matrix(1:10, nrow = 5, ncol = 2)
v <- matrix(seq(1, 20, 2), nrow = 5, ncol = 2)

# Create a for loop that iterate over each matrix
for (m in c("u", "v")) {
    z <- get(m)
    print(lm(z[, 2] ~ z[, 1]))
}
```

```

Call:
lm(formula = z[, 2] ~ z[, 1])

Coefficients:
(Intercept)       z[, 1]
          5            1


Call:
lm(formula = z[, 2] ~ z[, 1])
```

```
Coefficients:
(Intercept)       z[, 1]
         10            1
```