Programming in Data science
Control structures and Loops

Asad Waqar Malik

Dept. of Information Systems
Faculty of Computer Science and Information Technology
University of Malaya
Malaysia

Oct 31, 2019

Lecture Outline

Control Structure

Loops

Break and next statement

Excercise

Outline

If/Else

- If statements can be very useful in R, as they are in any programming languages. Often, you want to make choices and take action dependent on a certain value.
- Defining a choice in your code is pretty simple: If this condition is true, then carry out a certain task.

An if statement in R consists of three elements

The keyword if

logical value between parentheses

block of code between braces that has to be executed when the logical value is TRUE

If/Else

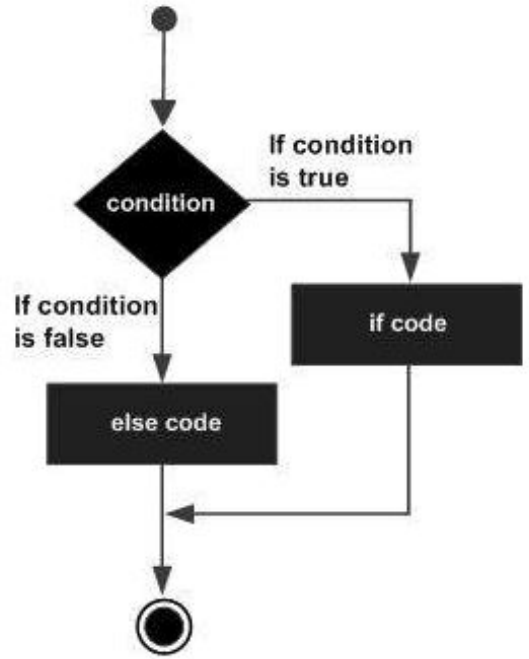
- The basic If/Else structure is:

Sample

```
1  if(boolean_expression) {
2     // statement(s) will execute if the boolean ↩
          expression is true.
3  } else {
4     // statement(s) will execute if the boolean ↩
          expression is false.
5  }
```

If/Else

If the Boolean expression evaluates to be true, then the if block of code will be executed, otherwise else block of code will be executed.



If/Else

- Here is a very small function, priceCalculator(), that calculates the price you charge to a customer based on the hours of work you did for that customer.
- The function should take the number of hours (hours) and the price per hour (pph) as input. The priceCalculator() function could be something like this:

Sample

```
1  priceCalculator <- function(hours, pph=40){
2      net.price <- hours * pph
3      round(net.price)
4  }
```

If/Else

- Now imagine you have some big clients that give you a lot of work. To keep them happy, you decide to give them a reduction of 10 percent on the price per hour for orders that involve more than 100 hours of work.
- So, if the number of hours worked is larger than 100, you calculate the new price by multiplying the price by 0.9. You can write that almost literally in your code like this:

Sample

```
priceCalculator <- function(hours, pph=40){
    net.price <- hours * pph
    if(hours > 100) {
      net.price <- net.price * 0.9
    }
    round(net.price)}
  #Calling
> priceCalculator(hours = 55)
[1] 2200
```

If/Else

Example

```
> x <- c("what","is","truth")
if("Truth" %in% x) {
   print("Truth is found")
} else {
   print("Truth is not found")
}
```

When the above code is compiled and executed, it produces the following result –

```
[1] "Truth is not found"
```

The else if statement

- We can further customize the control level with the else if statement. With else-if, you can add as many conditions as we want. The syntax is:

Example

```
if (condition1)
{     expr1     }
else if (condition2)
{    expr2      }
else if  (condition3)
{     expr3     }
else
{     expr4    }
```

Example

Sample

```
1  # Create vector quantiy
2  quantity <-  10
3  # Create multiple condition statement
4  if (quantity <20) {
5         print("Not enough for today")
6  } else if (quantity > 20  &quantity <= 30) {
7        print("Average day")
8  } else {
9         print("What a great day!")
10 }
```

Example

Categories	Products	VAT
A	Book, magazine, newspaper, etc..	8%
B	Vegetable, meat, beverage, etc..	10%
C	Tee-shirt, jean, pant, etc..	20%

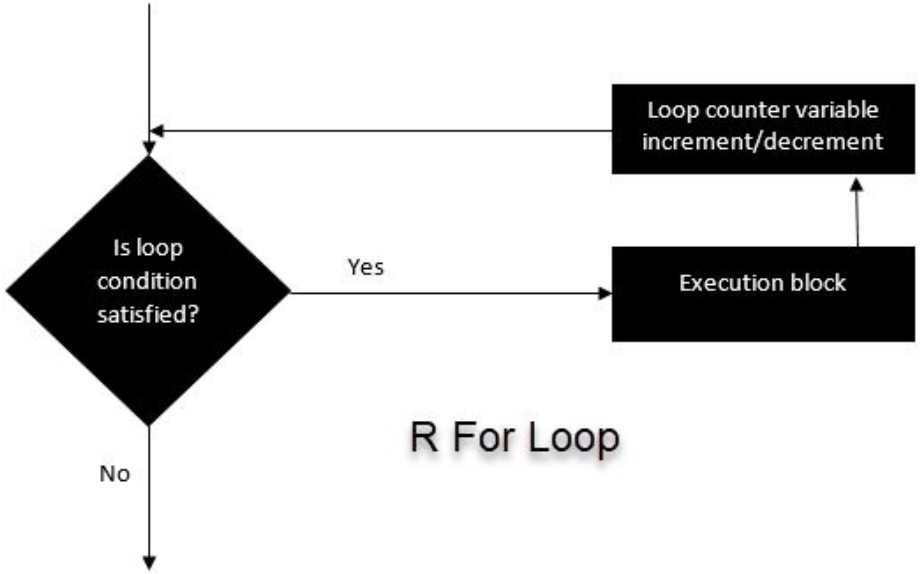
We can write a chain to apply the correct VAT rate to the product a customer bought.

```
1  category <- "A"
2  price <- 10
3  if (category =="A")
4  { cat("A vat rate of 8% is applied.","The total ↩
       price is",price *1.08)  }
5  else if (category =="B")
6  {  cat("A vat rate of 10% is applied.","The total ↩
       price is",price *1.10)  }
7  else {  cat("A vat rate of 20% is applied.","The ↩
       total price is",price *1.20)  }
```

Outline

for Loops

- A for loop is very valuable when we need to iterate over a list of elements or a range of numbers.
- Loop can be used to iterate over a list, data frame, vector, matrix or any other object.
- The loop structure is as follow.



for Loops

For Loop Syntax and Examples

```
1  For (i in vector) {
2       Exp
3  }
```

Example – We iterate over all the elements of a vector and print the current value.

```
1  # Create fruit vector
2  fruit <- c("Apple", "Orange", "Mango", "Banana")
3  for ( i in fruit){
4   print(i)
5  }
```

for Loops

Example : creates a non-linear function by using the polynomial of x between 1 and 4 and we store it in a list

```
1  # Create an empty list
2  list <- c()
3  # Create a for statement to populate the list
4  for (i in seq(1, 4, by=1)) {
5    list[[i]] <- i*i
6  }
7  print(list)
8  ## [1]   1   4 9 16
```

for Loop over a list

Looping over a list is just as easy and convenient as looping over a vector. Let's see an example

```
# Create a list with three vectors
fruit <- list(Basket = c("Apple", "Orange", "Passion fruit", "Banana"),
Money = c(10, 12, 15), purchase = FALSE)
for (p  in fruit)
{
  print(p)
}
## [1] "Apple" "Orange" "Passion fruit" "Banana"
## [1] 10 12 15
## [1] FALSE
```

for Loop over a matrix

A matrix has 2-dimension, rows and columns. To iterate over a matrix, we have to define two for loop, namely one for the rows and another for the column.

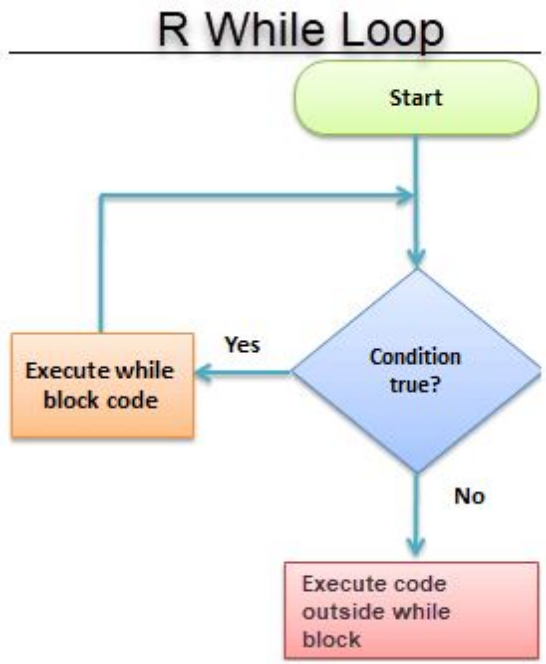
Sample

```
1  # Create a matrix
2  mat <- matrix(data = seq(10, 20, by=1), nrow = 6, ↩
       ncol =2)
3  # Create the loop with r and c to iterate over the ↩
       matrix
4  for (r in 1:nrow(mat))
5       for (c in 1:ncol(mat))
6            print(paste("Row", r, "and column",c, "↩
                 have values of", mat[r,c]))
```

while Loop in R

A loop is a statement that keeps running until a condition is satisfied. The syntax for a while loop is the following:

```
1  while (condition) {
2        Exp
3  }
```



while Loop in R

Create a loop and after each run add 1 to the stored variable. You need to close the loop, therefore explicitely tells R to stop looping when the variable reached 10.

```
#Create a variable with value 1
begin <- 1
#Create the loop
while (begin <= 10){
#See which we are
cat("This is loop number",begin)
#add 1 to the variable begin after each loop
begin <- begin+1
print(begin)
}
```

while Loop in R

You bought a stock at price of 50 dollars. If the price goes below 45, we want to short it. Otherwise, we keep it in our portfolio. The price can fluctuate between -10 to +10 around 50 after each loop. You can write the code as follow:

Code

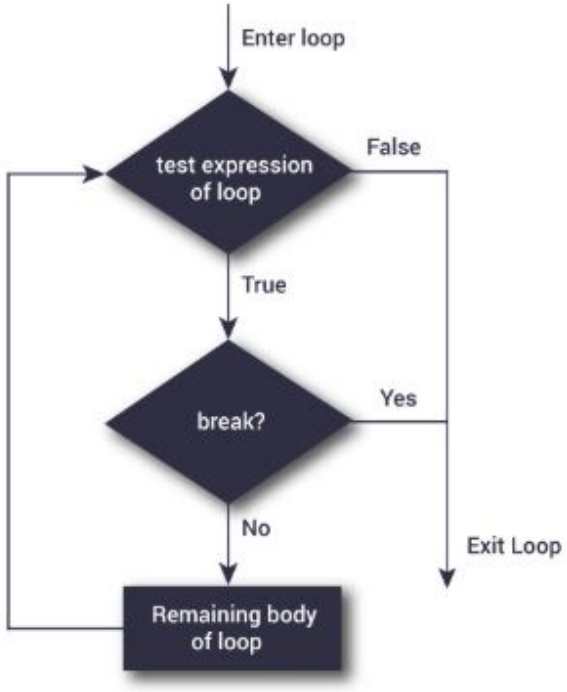
```
set.seed(123)
stock <- 50 # Set variable stock and price
price <- 50
loop <- 1 # Loop variable counts the number of loops
while (price > 45){
# Create a random price between 40 and 60
price <- stock + sample(-10:10, 1)
# Count the number of loop
loop = loop +1
# Print the number of loop
print(loop) }
```

Outline

break statement

- A break statement is used inside a loop (repeat, for, while) to stop the iterations and flow the control outside of the loop.
- The syntax of break statement is:

```
1  if (test_expression) ↩
       {
2  break
3  }
```



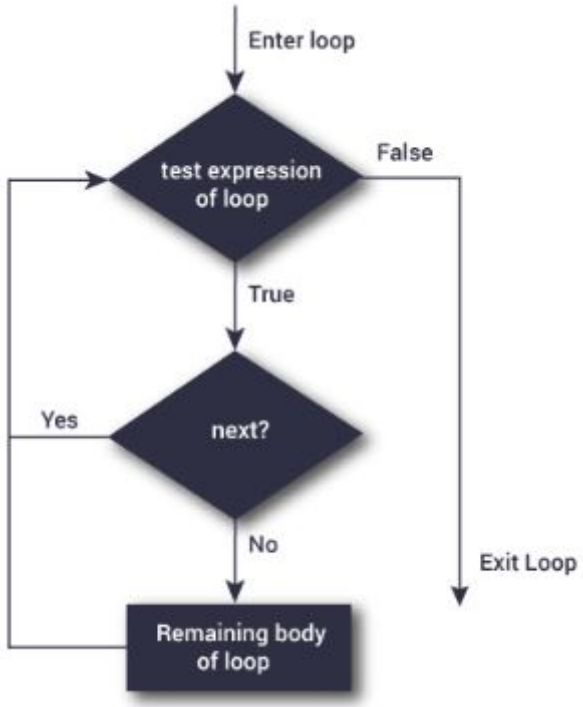
break statement

Example: break statement

```
1   x <- 1:5
2   for (val in x) {
3   if (val == 3){
4   break
5   }
6   print(val)
7   }
8   #Output
9   [1] 1
10  [1] 2
```

next statement

- A next statement is useful when we want to skip the current iteration of a loop without terminating it.
- On encountering next, the R parser skips further evaluation and starts next iteration of the loop.




```
1  if (test_condition) {
2  next
3  }
```

next statement

Example: next statement

```
1  x <- 1:5
2  for (val in x) {
3  if (val == 3){
4  next
5  }
6  print(val)
7  }
8  #Output
9  [1] 1
10 [1] 2
11 [1] 4
12 [1] 5
```

repeat Loop in R

- A repeat loop is used to iterate over a block of code multiple number of times.
- There is no condition check in repeat loop to exit the loop.
- We must ourselves put a condition explicitly inside the body of the loop and use the break statement to exit the loop. Failing to do so will result into an infinite loop.

Syntax of repeat loop

```
1  repeat {
2  statement
3  }
```

repeat Loop in R

Example: repeat loop

```
x <- 1
repeat {
print(x)
x = x+1
if (x == 6){
break
}
}
#Output
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

Outline

1. Write a for loop that iterates over the numbers 1 to 7 and prints the cube of each number using print().
2. Write a for loop that iterates over the column names of the inbuilt iris dataset and print each together with the number of characters in the column name in parenthesis. Example output: Sepal.Length (12). Use the following functions print(), paste0() and nchar().
3. Write a while loop that prints out standard random normal numbers (use rnorm()) but stops (breaks) if you get a number bigger than 1.
4. Using next adapt the loop from last exercise so that doesn't print negative numbers.
5. Using a for loop simulate the flip a coin twenty times, keeping track of the individual outcomes (1 = heads, 0 = tails) in a vector that you preallocte.