Data Science for Beginners, University of Essex

Day 2: Common Functions in R

Dr. Howard Liu

11-01-2022

Learning Objectives Today

1. if-else functions
2. for, while, repeat loop
3. break and next statement
4. create functions

1. if-else statements

An if-else statement is a great tool for the developer trying to return an output based on a condition. In R, the syntax is:

```
# if (condition) {
#     Expr1
#   } else {
#     Expr2
# }
```

Let's start with an example. Let's say you want to write a small program to examine whether your exam score is satisfactory based on a condition (let's say 60) If the score is greater than 60, the code will print "You did great!" otherwise "You need to work harder next time".

Create vector score, and set up an if-else statement

```
score <- 59
if (score > 60) {
  print('You did great!')
} else {
  print('You need to work harder next time')
}
```

```
## [1] "You need to work harder next time"
```

```
score <- 61
if (score > 60) {
  print('You did great!')
} else {
  print('You need to work harder next time')
}
```

```
## [1] "You did great!"
```

1-2: multiple `if...else` statements

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

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

Again, let's use the exam score example.

```
score  <-  100

# Create multiple condition statement
if (score  < 60) {
  print('Gosh I should have spent more time on it..')
} else if (score  > 60  & score  <= 80) {
  print('I am doing fine, but still have room for improvement.')
} else {
  print('I nailed it!')
}
```

```
## [1] "I nailed it!"
```

```
score  <-  79

# Create multiple condition statement
if (score  < 60) {
  print('Gosh I should have spent more time on it..')
} else if (score  > 60  & score  <= 80) {
  print('I am doing fine, but still have room for improvement.')
} else {
  print('I nailed it!')
}
```

```
## [1] "I am doing fine, but still have room for improvement."
```

1-3: `ifelse()` statement

In R, the ifelse() function is a shorthand vectorized alternative to the standard if…else statement. The syntax of the ifelse() function is:

```
# ifelse(test_expression, x, y)
```

Let's do an example

```
score  <-  49
ifelse(score > 60, "Ah nice", "Oh crap")
```

```
## [1] "Oh crap"
```

We can also test a vector with the ifelse() function

```
score <- c(12, 90, 23, 14, 80, 100, 50)
ifelse(score > 60, "Ah nice", "Oh crap")
```

```
## [1] "Oh crap" "Ah nice" "Oh crap" "Oh crap" "Ah nice" "Ah nice" "Oh crap"
```

2. `for` and `while` loops

There are three types of loop in R programming: * For Loop * While Loop * Repeat Loop

2-1: `for` Loop in R

It is a type of control statement that enables one to easily construct a loop that has to run statements or a set of statements multiple times. For loop is commonly used to iterate over items of a sequence. It is an entry controlled loop, in this loop the test condition is tested first, then the body of the loop is executed, the loop body would not be executed if the test condition is false. This is THE MOST USED loop in my daily life. The syntex is:

```
# for (value in sequence)
# {
#   statement
# }
```

Example 1: using for loop to print numbers from 1 to 5

```
for (val in 1:5){
  # statement
  print(val)
}
```

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

Example 2: loop over each item in a vector

```
week = c('Sunday',
           'Monday',
           'Tuesday',
           'Wednesday',
           'Thursday',
           'Friday',
           'Saturday')

# using for loop to iterate over each string in the vector
for (day in week){

  # displaying each string in the vector
  print(day)
} # iterate over each string in a week. In each iteration, each day of the week is displ
ayed.
```

```
## [1] "Sunday"
## [1] "Monday"
## [1] "Tuesday"
## [1] "Wednesday"
## [1] "Thursday"
## [1] "Friday"
## [1] "Saturday"
```

Example 3: Use for loop to fill in values to an empty box

```
emptyBox = rep(0,10)

for(ii in 1:10){
  emptyBox[ii] = ii
  # cat("\r",ii)
  # flush.console()
}
emptyBox
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

2-2: `while` Loop in R

It is a type of control statement which will run a statement or a set of statements repeatedly unless the “given condition” becomes false. It is also an entry controlled loop, in this loop the test condition is tested first, then the body of the loop is executed, the loop body would not be executed if the test condition is false.

```
# R – While loop Syntax:
# while ( condition ) {
#   statement
# }
```

Example:

```
val = 1

# using while loop
while (val <= 5){
  # statements
  print(val)
  # after displaying the value, we add 1 and redo the loop until the value reaches the c
ondition
  val = val + 1
}
```

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

2-3: Jump Statements in Loop

We use a jump statement in loops to terminate the loop at a particular iteration or to skip a particular iteration in the loop. The two most commonly used jump statements in loops are:

Break Statement: The break keyword is a jump statement that is used to terminate the loop at a particular iteration.

```
for (val in 1: 5){
  # checking condition
  if (val == 3){
    # using break keyword
    break
  }

  # displaying items in the sequence
  print(val)
} # See! If the value of val becomes 3 then the break statement will be executed and the loop will terminate.
```

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

Next Statement: The next keyword is a jump statement which is used to skip a particular iteration in the loop.

```
for (val in 1: 5){
  # checking condition
  if (val == 3){
    # using next keyword
    next
  }

  # displaying items in the sequence
  print(val)
}
```

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

2-3: `Repeat` Loop in R

It is a simple loop that will run the same statement or a group of statements repeatedly until the stop condition has been encountered. Repeat loop does not have any condition to terminate the loop, a programmer must specifically place a condition within the loop's body and use the declaration of a break statement to terminate this loop. If no condition is present in the body of the repeat loop then it will iterate infinitely.

```
# repeat {
#   statement
#
#   if( condition ) {
#     break
#   }
# }
```

Example:

```
val = 1

repeat{
  # statements
  print(val)
  val = val + 1

  # checking stop condition
  if(val > 5){
    # using break statement
    # to terminate the loop
    break
  }
}
```

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

3. Create your own function in R

Sometimes, it makes your life easier by writing some simple functions for your usage. For example, if we want to create a function that can quickly convert fahrenheit to celsius, we can use the following example. The syntex for function is:

```
# yourFunctionName <- function(input){
#   # calculation
#   return(output) or print(output)
# }
```

Example: We want to write a function that converts F to C

```
fahrenheit_to_celsius <- function(temp_F) {
  temp_C <- (temp_F - 32) * 5 / 9
  print(temp_C)
}

fahrenheit_to_celsius(70)
```

```
## [1] 21.11111
```

```
fahrenheit_to_celsius(90)
```

```
## [1] 32.22222
```

Example: We want to write a function that raises the value to put in to the power of something, and write a complete sentence to describe it.

```
pow <- function(x, y) {
  # function to print x raised to the power y
  result <- x^y
  print(paste(x,"raised to the power", y, "is", result))
}
pow(8, 2)
```

```
## [1] "8 raised to the power 2 is 64"
```

```
pow(8, 3)
```

```
## [1] "8 raised to the power 3 is 512"
```

Great! We've finished the lecture and you can go to day2 exercise to do some additional practices for today's content.