

Basic Course on R: Programming Structures 2 Practical Answers

Elizabeth Ribble*

20-24 May 2019

Contents

1	Part A: Scope	2
2	Part B: <code>if()</code> Statements, <code>warning()</code> , and <code>stop()</code>	4

*emcclel3@msudenver.edu

1 Part A: Scope

1. For each of the following sets of commands, give the value that will be returned by the last command. Try to answer without using R.

```
a) w <- 5
   f <- function(y) {
     return(w + y)
   }
   f(y = 2)

## [1] 7
```

This will return 7 because `w` is 5 and we are evaluating the function at `y = 2`.

```
b) w <- 5
   f <- function(y) {
     w <- 4
     return(w + y)
   }
   f(y = 2)

## [1] 6
```

This will return 6 because `w` is reassigned as 4 inside the function and we are evaluating the function at `y = 2`.

2. Among the variables `w`, `d`, and `y`, which are global to `f()` and which are local?

```
w <- 2
f <- function(y) {
  h <- function() {
    d <- 3
    return(w + y)
  }
  return(d * h())
}
```

The object `w` is global to `f()` while `d` and `y` are local to `f()`.

3. Do the following in R.

a) Try:

```
myFun1 <- function() {  
  a <- 2  
  b <- 3  
  myFun2(3)  
}  
myFun2 <- function(y) {  
  return(y + a + b)  
}  
myFun1()  
## [1] 6
```

What happens?

We get an error message because `a` and `b` are local to `myFun1` so the function `myFun2` can't find them in the global environment.

b) Now try:

```
a <- 1  
b <- 2  
myFun1()  
## [1] 6
```

What happens?

We get the value 6 because the values `a` and `b` are global so `myFun2` can find them and use them in its commands.

4. What value for `w` will be printed in the last line below? Try to answer without using R.

```
w <- 1  
f <- function(y) {  
  g <- function() {  
    w <- 3  
    return(2)  
  }  
  return(g())  
}  
f(y = 1)
```

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

```
w
```

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

We get the value 3 because the superassign operator overwrote the original assignment of `w`.

5. What value for `w` will be printed in the last line below? Try to answer without using R.

```
w <- 1
f <- function(y) {
  w <- 2
  g <- function() {
    w <- 3
    return(2)
  }
  return(g())
}
f(y = 1)
```

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

```
w
```

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

We get the value 1 because the superassign operator only overwrote the assignment of `w` within the `f()` function.

2 Part B: `if()` Statements, `warning()`, and `stop()`

The functions `warning()` and `stop()` are used to print a warning message and to stop the execution of the function call and print an error message. For example:

```
noNegMean <- function(x) {
  if(all(x < 0)) {
    stop("All values in x are negative")
  }
}
```

```

}
if(any(x < 0)) {
  x[x < 0] <- 0
  warning("Negative values in x replaced by zero")
}
return(mean(x))
}

```

1. The file **nonegmean.txt** contains the above code; source it into R and then pass **noNegMean()** a vector containing some negative and some positive values. What happens?

```

source("nonegmean.txt")
noNegMean(c(-1,0,1))

## Warning in noNegMean(c(-1, 0, 1)): Negative values in x replaced
by zero

## [1] 0.3333333

```

We get the warning message and it returned 0.3333, which is the average of 0, 0, 1.

2. What happens when you pass **noNegMean()** a vector containing all negative values?

```

source("nonegmean.txt")
#noNegMean(c(-1,-1,-1)) # not run; error message

```

We get the error message and nothing is returned.

3. Write a function **ratio()** that takes two arguments, **x** and **y**, and attempts to compute the ratio **x/y**. If both **x == 0 & y == 0**, the function should stop and print an error message about dividing 0 by 0. If **y == 0** (but not **x**), the function should print a warning message about dividing by 0, and then return **x/y** (which will be **Inf**). In all other cases, it should return **x/y**.

Test your **ratio()** function first using two nonzero values for **x** and **y**, then using a nonzero **x** but **y = 0**, and finally using **x = 0** and **y = 0**.

```
ratio <- function(x,y) {  
  if(x == 0 & y == 0) {  
    stop("Cannot divide zero by zero.")  
  }  
  if(y == 0) {  
    warning("Cannot divide by zero.")  
  }  
  ratio <- x/y  
  return(ratio)  
}  
  
ratio(2,3)  
  
## [1] 0.6666667  
  
ratio(0,0)  
  
## Error in ratio(0, 0): Cannot divide zero by zero.  
  
ratio(1,0)  
  
## Warning in ratio(1, 0): Cannot divide by zero.  
  
## [1] Inf
```