Stat 124: Introduction to Programming
Task 8-9: Documents with R

Anne Christine Amores
Jan Joaquin Bulao

December 16, 2023

1 Problems

1. **10 pts** Rewrite the following to eliminate the loops, first using `apply()` and then using `rowSums()`:

```
X <- matrix(runif(100000),1000,100)
z <- rep(0,1000)
for(i in 1:1000){
    for(j in 1:100){
        z[i] <- z[i] + X[i,j]
    }
}
```

   Confirm that all three versions give the same answers, but that your rewrites (`apply()` and `rowSums()`) are faster than the original. You may have to use `identical()`, and `system.time()`.

Answer

```
# Original Code
X <- matrix(runif(100000),1000,100)
z <- rep(0,1000)
for(i in 1:1000){
  for(j in 1:100){
    z[i] <- z[i] + X[i,j]
  }
}
```

```
# New Code
z1 <- apply(X,1, sum)

z2 <- rowSums(X)

# Check if same output
identical(z,z1)

## [1] TRUE

identical(z,z2)

## [1] TRUE

# Check if faster
system.time({
  X <- matrix(runif(100000),1000,100)
  z <- rep(0,1000)
  for(i in 1:1000){
    for(j in 1:100){
      z[i] <- z[i] + X[i,j]
    }
  }}
)

##    user  system elapsed
##   0.012   0.000   0.012

system.time(apply(X,1, sum))

##    user  system elapsed
##   0.002   0.000   0.003

system.time(rowSums(X))

##    user  system elapsed
##   0.001   0.000   0.000

# Thus, using apply(), and rowSums() is faster.
```

2. **10 pts** Explain what the following code does. Rewrite the following, replacing the loop with a vectorized operation. Confirm that your rewrite is faster but gives the same result. You may have to use the `identical()` function. HINT: The solution entails one line only. Use `[]` to rewrite your code.

```
n <- 100000
z <- rnorm(n)
zneg <- 0
j <- 1
for(i in 1:n){
    if(z[i] <0){
        zneg[j] <- z[i]
        j <- j+1
    }
}
```

Answer

The following code generates a random sample of 'n' numbers and creates a new vector, zneg, to store the negative values from the generated sample.

```
# Original Code
n <- 100000
z <- rnorm(n)
zneg <- 0
j <- 1
for(i in 1:n){
  if(z[i] <0){
    zneg[j] <- z[i]
    j <- j+1
  }
}

# New Code
checker <- ifelse(z<0, z, NA)
zneg.new <- checker[!is.na(checker)]

# Check if same output
identical(zneg,zneg.new)

## [1] TRUE

# Check if faster
system.time({
```

```
n <- 100000
z <- rnorm(n)
zneg <- 0
j <- 1
for(i in 1:n){
  if(z[i] <0){
    zneg[j] <- z[i]
    j <- j+1
  }
}})

##    user  system elapsed
##   0.021   0.000   0.021

system.time({
checker <- ifelse(z<0, z, NA)
zneg.new <- checker[!is.na(checker)]
})

##    user  system elapsed
##   0.003   0.000   0.003
```

---

3. **15 pts total** Recall Task 2 in the Week 7 Individual Group Task. Write a function called `square()` that prints a square composed of asterisks and number signs. The number signs are for the perimeter of the square, while asterisks are for the interior of the square. The function's argument is `s`, the square's side-length. As examples,

```
square(3):                square(4):                square(5):

                                                        #####
                              ####                      #***#
###                           #**#                      #***#
#*#                           #**#                      #***#
###                           ####                      #####
```

As you go through the items below, the state of the `square()` function improves.

(a) **8 pts** Create the `square()` function without any input validation yet.

Answer

```r
square <- function(s) {
  for (row in 1:s) {
    for (col in 1:s) {
      if (row == 1 | row == s | col == 1 | col == s) {
        cat("#")
      } else {
        cat ("*")
      }
    }
    cat("\n")
  }
}

square(3)

## ###
## #*#
## ###
```

(b) **6 pts** Copy your codes from the previous item. Add input validation to the `square()` function. The value of `s` must be a positive integer from 2 to 10 only.

Answer

```r
square <- function(s) {
  #To check if the input is a positive integer from 2 to 10
  if (s %% 1 != 0 | s < 2 | s > 10) {
    stop("Please enter an integer from 2 to 10")
  }

  for (row in 1:s) {
    for (col in 1:s) {
      if (row == 1 | row == s | col == 1 | col == s) {
        cat("#")
        } else {
          cat ("*")
        }
      }
    cat("\n")
  }
}
# Inputs of non-integers or out-of-range integers return:
square(-3)
```

```
## Error in square(-3):  Please enter an integer from 2 to 10
```

```r
square(3.5)
```

```
## Error in square(3.5):  Please enter an integer from 2 to 10
```

(c) **1 pt** Copy your codes from the previous item. Specify a default value (`s=5`) for the `square()` function.

Answer

```r
s <- 5
square <- function(s) {
  if (s %% 1 != 0 | s < 2 | s > 10) {
    stop("Please enter an integer from 2 to 10")
  }

  for (row in 1:s) {
    for (col in 1:s) {
      if (row == 1 | row == s | col == 1 | col == s) {
        cat("#")
        } else {
          cat ("*")
        }
      }
    cat("\n")
  }
}

square(s)
```

```
## #####
## #***#
## #***#
## #***#
## #####
```

End of Document.