Lab 4: Control Structures and Function

Anthony Hu

June 11, 2021

Introduction

The main purpose of this lab is to practice control structures in R:

- `if` and `else`: testing a condition and acting on it
- `for`: execute a loop a fixed number of times
- `while`: execute a loop while a condition is true

You will need to modify the code chunks so that the code works within each of chunk (usually this means modifying anything in ALL CAPS). You will also need to modify the code outside the code chunk. When you get the desired result for each step, change `Eval=F` to `Eval=T` and knit the document to HTML to make sure it works. After you complete the lab, you should submit your HTML file of what you have completed to Sakai before the deadline.

Part 1: Vector and Control Structures

a) (2 points) Write code that creates a vector `x` that contains `100` random observations from the standard normal distribution (this is the normal distribution with the mean equal to `0` and the variance equal to `1`).

```
#
x <- rnorm(100, mean = 0, sd = 1)
x
```

```
##   [1] -0.42141758  0.23421165  0.44643030 -0.95939611  0.90444587
##   [6] -0.55565430  1.46991008  2.36267218  0.63236658  0.70674126
##  [11]  2.38823994 -1.72642796 -2.09031740  0.53657689  0.49707230
##  [16] -0.58717005  0.76101897  0.65390506 -0.99563458  1.11521524
##  [21] -0.62911638 -1.16414806  1.03167537 -1.30883637  0.52160056
##  [26] -0.38239581  1.14816071  1.07716460  1.99729596  0.04100221
##  [31]  0.90394654 -1.11175408  2.12713424  0.64903931 -2.16881358
##  [36]  1.60385005  0.20505670 -2.17776339 -0.40328322 -2.50618152
##  [41]  0.20876319  1.89170415  1.44580834  1.23452133 -0.98607190
##  [46] -0.89893603 -1.24393193 -0.15284547  0.53876841  0.47006912
##  [51]  1.59297241 -0.38993849 -0.19539297 -0.46005511  2.20848739
##  [56]  1.70787228  1.29110493 -1.08250711 -0.10151257  0.42809295
##  [61] -1.08272720  0.32870882  1.07009227 -1.09673291 -0.11091814
##  [66] -0.74151631  2.11784206  2.63398597 -0.28265957  0.69375219
##  [71] -0.66525160 -0.22463968  0.40084729 -0.82482822  0.77012160
##  [76]  0.01126763  0.69272092 -0.14558868  0.92549043  0.87231864
##  [81] -1.64353181 -1.44551238  1.32103160 -1.45448317  1.09917185
##  [86]  0.20716464  0.58435285  0.80334138  0.83345007  0.27176789
##  [91]  0.20049376 -0.30138773 -1.66345638  1.52519432  0.17607367
##  [96]  0.58186616  1.12654013  0.26015745 -0.84889618 -0.18706744
```

```
length(x)
```

```
## [1] 100
```

b) (2 points) Write code that replaces the observations in the vector `x` that are greater than or equal to `0` with a string of characters `"non-negative"` and the observations that are smaller than `0` with a string of characters `"negative"`. Hint: try `ifelse()` funtion.

```
#
a <- ifelse(x >= 0, "non-negative", "negative")
a
```

```
##   [1] "negative"     "non-negative" "non-negative" "negative"
##   [5] "non-negative" "negative"     "non-negative" "non-negative"
##   [9] "non-negative" "non-negative" "non-negative" "negative"
##  [13] "negative"     "non-negative" "non-negative" "negative"
##  [17] "non-negative" "non-negative" "negative"     "non-negative"
##  [21] "negative"     "negative"     "non-negative" "negative"
##  [25] "non-negative" "negative"     "non-negative" "non-negative"
##  [29] "non-negative" "non-negative" "non-negative" "negative"
##  [33] "non-negative" "non-negative" "negative"     "non-negative"
##  [37] "non-negative" "negative"     "negative"     "negative"
##  [41] "non-negative" "non-negative" "non-negative" "non-negative"
##  [45] "negative"     "negative"     "negative"     "negative"
##  [49] "non-negative" "non-negative" "non-negative" "negative"
##  [53] "negative"     "negative"     "non-negative" "non-negative"
##  [57] "non-negative" "negative"     "negative"     "non-negative"
##  [61] "negative"     "non-negative" "non-negative" "negative"
##  [65] "negative"     "negative"     "non-negative" "non-negative"
##  [69] "negative"     "non-negative" "negative"     "negative"
##  [73] "non-negative" "negative"     "non-negative" "non-negative"
##  [77] "non-negative" "negative"     "non-negative" "non-negative"
##  [81] "negative"     "negative"     "non-negative" "negative"
##  [85] "non-negative" "non-negative" "non-negative" "non-negative"
##  [89] "non-negative" "non-negative" "non-negative" "negative"
##  [93] "negative"     "non-negative" "non-negative" "non-negative"
##  [97] "non-negative" "non-negative" "negative"     "negative"
```

c) (2 points) Write `for`-Loop to count how many observations in the vector `x` are non-negative and how many observations are negative. (There are many easier ways to solve this problem. Please use `for`-Loop to practice the things learned in the lecture.)

```
#
num_non_nega = 0
num_negative = 0
for(b in a){
  if(b == "negative"){
    num_negative = num_negative + 1
  }
  else{
    num_non_nega = 1 + num_non_nega
  }
```

```
}
num_non_nega
```

```
## [1] 59
```

```
num_negative
```

```
## [1] 41
```

Part 2: Matrix and Control Structures

a) (4 points) Create a 100000 by 10 matrix `A` with the numbers 1 : 1000000. Create a `for`-loop that calculates the sum for each row of the matrix and save the results to a vector `sum_row`. (Don't print the whole matrix in your submission as the matrix is very large. Otherwise, you'll lose scores for it.)

```
#
A <- matrix(1:1000000, nrow = 100000, ncol = 10)
sum_row <- vector()
for(y in 1:100000){
  sum_row[y] <- sum(A[y,])
}
head(sum_row, 10)
```

```
##  [1] 4500010 4500020 4500030 4500040 4500050 4500060 4500070 4500080
##  [9] 4500090 4500100
```

Verify that your results are consistent with what you obtain with the built-in `rowSums` function.

```
sum_row_rowSums = as.integer(rowSums(A))
head(sum_row_rowSums)
```

b) (4 points) Another common loop structure that is used is the `while` loop, which functions much like a `for` loop, but will only run as long as a test condition is `TRUE`. Modify your `for` loop from exercise (a) and make it into a `while` loop. Write code to check if the results from `for` loop are the same as the results from `while` loop.

```
#
sum_row_2 <- vector()
y = 0
while(y < 100001){
  sum_row_2[y] <- sum(A[y,])
  y = y + 1
}
head(sum_row_2, 10)
```

```
##  [1] 4500010 4500020 4500030 4500040 4500050 4500060 4500070 4500080
##  [9] 4500090 4500100
```

Part 3: Simulation Study

Suppose that $X_1, \ldots, X_n$ are independent and identically distributed (iid) binomial random variables such that

$$P(X_i = x \mid k, p) = \binom{k}{x} p^x (1-p)^{k-x}, \quad x = 0, 1, \ldots, k$$

for all $i = 1, \ldots, n$. Assume that both k and p are unknown and use the method of moments to obtain point estimators of both parameters. This somewhat unusual application of the binomial model has been used to estimate crime rates for crimes that are known to have many unreported occurrences. For such a crime, both the true reporting rate, p, and the total number of occurrences, k, are unknown. Equating the first two sample moments to those of the population yields the system of equations

$$\bar{X} = kp \quad \text{and} \quad \frac{1}{n}\sum_{i=1}^{n} X_i^2 = kp(1-p) + k^2p^2,$$

where $\bar{X}$ is the sample mean. Solving for k and p leads to

$$\hat{k} = \frac{\bar{X}^2}{\bar{X} - (1/n)\sum_{i=1}^{n}(X_i - \bar{X})^2} \quad \text{and} \quad \hat{p} = \frac{\bar{X}}{\hat{k}}.$$

It is difficult to analyze the performance of $\hat{k}$ and $\hat{p}$ analytically so you are asked to perform a simulation study using `R`. The idea is to generate random samples and investigate the performance of $\hat{k}$ and $\hat{p}$ using random samples.

Q1

1. Generate a single simple random sample vector `x` of length `n = 50` from the binomial distribution with the parameters `k = 10`, `p = 0.4`. (1 point)

```
k = 10
p = 0.4
x = rbinom(50,k,p)
x
```

```
##  [1] 4 0 5 3 6 2 2 2 6 3 5 3 4 2 4 1 3 5 3 3 6 5 3 3 5 4 2 4 5 5 5 4 3 4 4
## [36] 5 6 3 7 6 3 6 2 3 5 4 3 3 5 3
```

Q2

2. Write a function that takes a sample vector as its input and returns the estimates of `k` and `p` given above. (4 points)

```
est_kp = function(x){
  X_bar = mean(x)
  n = length(x)
  k_hat = (X_bar ^ 2) / (X_bar - var(x))
  p_hat = X_bar / k_hat
  return(c(k_hat,p_hat))
}
est_kp(x)
```

```
## [1] 8.8719843 0.4328231
```