Bài 2.

Giới thiệu về R (phần 2)
Lập trình cơ bản trong R

1. Các hàm toán học thường dùng trong R:

log(x) : logarti cơ số e
log10(x), log(x,n): logarit cơ số n
exp(x): e^x
sqrt(x): căn bậc 2 của x
factorial(x): x!
choose(n,k): tổ hợp n chập k
floor(x): giá trị nguyên <x (sàn của x)
ceiling(x): giá trị nguyên > x (trần của x)
trunc(x): làm tron tới giá trị nguyên gần nhất giữa x và 0.
round(x, digits=n): làm tròn x đến n chữ số
signif(x, digits=n): hiển thị x dưới dạn dấu chấm thập phân, n tổng chữ số hiển thị
sin(x), cos(x), tan(x)
abs(x): |x|
x %/% y: lấy phần nguyên của phép chia x/y
x %% y: lấy phần dư của phép chia x/y

2. Lệnh điều kiện:

Các phép toán so sánh: == (equal), != (not equal), >, <, >=, <=.
Các phép toán logic: & (and), | (or), ! (not), && (AND with IF), || (OR with IF).

2.1 Lệnh if-esle:

Cú pháp:

```
if (logical expr) {group cmd}
if (logical expr) {group cmd1} else {group cmd2}
```

Thường sử dụng && và || trong biểu thức điều kiện của IF.

Ví dụ:

1.

```
x <- c(2,4,4,5,6)
if (all(x>0)) y <- sqrt(x)
y
```

2.

```
M <- matrix(rpois(12,3),nrow=3) #Tạo ngẫu nhiên ma trận
d <- dim(M) #Tìm số hàng/số cột của ma trận
{if (d[1] == d[2])
   { cat('M is a squared matrix\n')
     cat('The determinant of M is', det(M),'\n')
```

```
    }
else
     cat('M is not a squared matrix\n')
}
```

3.

```
#Gia he phuong trinh bac hai ax^2 + bx + c = 0 voi a≠0
delta <- b^2 - 4*a*c
if (delta < 0){
  cat('Phuong trinh vo nghiem\n')
}else{
  cat('Phuong trinh co nghiem:\n');
  sol <- c(-b + sqrt(delta), -b + sqrt(delta))/(2*a)
}
sol
```

2.2 Lệnh ifelse:

Cú pháp: ifelse(test, true_value, false_value)

Ví dụ:

```
x <- 1:10
ifelse(x<5 | x>8,x,0)
```

3. Vòng lặp: Các vòng lặp thường được dùng là 'for' và 'while', 'apply'; ít thông dụng là vòng lặp 'repeat'. Lệnh 'break' dùng để ngắt vòng lặp, 'next' dùng để nhảy sang bước lặp kế tiếp.

3.1. Vòng lặp For:

Cú pháp:

```
for (variable in sequence) {
      statements
}
```

Ví dụ:

1.

```
#Calculate and print the first 6 elements of the Fibonacci series
fib <- numeric(6)
fib[1] <- 1; fib[2] <- 1
cat('The 1st Fibonacci number is:',fib[1],'\n')
cat('The 2nd Fibonacci number is:',fib[2],'\n')
for(i in 3:6){
   fib[i] <- fib[i-1] + fib[i-2]
   cat('The ',i,'th Fibonacci number is:',fib[i],'\n')
}
```

2.

```
#Tinh n!
```

```
n = 10; fac <- 1
for(i in 1:n){
   fac <- fac*i
}
fac
```

3.

```
#stop on condition and print error message
x <- 1:10; z <- NULL
for(i in x) {
   if (x[i]<5) { z <- c(z,x[i]-1)  } else { stop("values need to be <5") }
}
z
```

3.2. Vòng lặp While:

Cú pháp:

```
while (condition) statements
```

Vòng lặp ngừng khi điều kiện trả về giá trị FALSE.

Ví dụ:

1.

```
z<- 0
while (z<5) {
  z <- z+2
  print(z)
}
```

2.

```
#Toss a die untill a 6 is obtained
#toss <- ceiling(6*runif(1)) #ceiling function
toss <- sample(1:6,1,rep=T) #sample function
count <- 1
while (toss != 6){
   #toss <- ceiling(6*runif(1))
   toss <- sample(1:6,1,rep=T)
   cat('Toss', count,' was a', toss,'\n')
   count <- count + 1
}
cat('There was ',count-2,' tosses before the firs 6\n')
```

3.3. Lệnh Apply:

Cú pháp:

```
apply(X, MARGIN, FUN, ARGs)
```

X: mảng, ma trận hoặc data.frame
MARGIN: 1 đối với hàng, 2 đối với cột, c(1,2) cho cả hai.
FUN: hàm cần thực thi lên X.

ARGs: các tham số có thể có của hàm FUN.
Kết quả trả về của lệnh 'apply' sẽ là một véc-tơ hay ma trận.

Ví dụ:

1.

```
X <- matrix(1:24,nrow=4)
apply(X,1,sum) #Tinh tong hang
apply(X,2,sum) #Tinh tong cot
```

2.

```
# create a matrix of 10 rows x 2 columns
m <- matrix(c(1:10, 11:20), nrow = 10, ncol = 2)
# mean of the rows
apply(m, 1, mean)
# mean of the columns
apply(m, 2, mean)
# divide all values by 2
apply(m, c(1,2), function(x) x/2)
```

4. Script và hàm:

4.1. Script:

- Tập hợp các dòng lệnh.
- Tạo script: vào File -> New script; lưu file dưới định dạng '*.R'.
- Gọi lại script: > source('ten_script.R') hoặc File -> Source R code rồi chọn script cần gọi.

4.2. Hàm:

- R cho phép người dùng tự viết hàm; những hàm này sử dụng tương tự như các hàm có sẵn của R (sum, mean, var, …)

Cú pháp:

myfct <- **function**(arg1, arg2, …) { function_body }

myfct: tên hàm, do người dùng đặt, không đặt tên trùng với tên hàm đã có,
arg1, arg2, … : các tham số của hàm,
function_body: thân hàm, gồm các lệnh xử lý.
Giá trị trả về của 1 hàm thường là giá trị của biểu thức lệnh đặt ở cuối thân hàm, hoặc dùng lệnh return().
Để hàm có hiệu lực, cần biên dịch hàm 1 lần trước khi sử dụng. (chọn toàn bộ hàm, chọn Run line or selection/Ctrl + R hoặc dùn lệnh source('file.name.R') trong đó file.name.R là file mà chứa hàm bên trong).

Ví dụ:

1. Viết hàm tính sai số chuẩn,

```
stderr <- function(x) {# x: vecto du lieu
        se <- sd(x)/length(x);
```

```
        se # hoac return(se)
}
```

Sử dụng: trong console của R

```
> x <- round(rnorm(50,20,4),digit=0) #Nhap x bat ky
> stderr(x)
```

2. Viết hàm chuyển 1 số hệ thập phân sang nhị phân

```
binary <-function(x){
        i<-0
        string<-numeric(32)
        while(x>0){
                string[32-i]<-x%%2
                x<-x%/% 2
                i<-i+1
        }
        first<-match(1,string)
        string[first:32]
}
```

Sử dụng:

#Chuyen so 59 sang dang nhi phan, gan cho bien x

```
> x <- binary(59)
```

4.3. Hàm với switch: cho phép thực hiện lệnh với nhiều lựa chọn

```
central<-function(y, measure){
        switch(measure,
                Mean = mean(y),
                Geometric = exp(mean(log(y))),
                Harmonic = 1/mean(1/y),
                Median = median(y),
                stop("Measure not included"))
}
central(rnorm(100,10,2),"Harmonic")
central(rnorm(100,10,2),4)
```

5. Bài tập:

1. Tạo một vec-tơ X chứa n phần tử (n: tự cho). Viết hàm tính tổng tích lũy đến vị trí thứ i của X.

2. Thể tích hình cầu với bán kính r là: $V = 4\pi r^3/3$. Hãy viết hàm xây dựng 1 dataframe để tính thể tích hình cầu với bán kính tương ứng là 3,4,5,…,20. Cột radius lưu bán kính và cột volume lưu thể tích.

3. Trong file data01.xls, dùng lệnh if và vòng lặp for để tạo biến Index theo yêu cầu sau: Nếu Age <=60 thì Index = 0; 60 < Age <=70 thì Index = 1; 70< Age<=80 thì Index = 2 và Age >80 thì Index = 3.

4. File data11.xls chứa số liệu về chiều cao của 1 loại cây trồng theo bảng tần số dạng khoảng. Thực hiện các bước sau:

a. Đọc số liệu từ data11.xls và gán vào 1 dataframe.

b. Viết một hàm tính tham số là các biến trong dataframe vừa nhập, xuất ra các giá trị sau: chiều cao bé nhất, lớn nhất của cây, trung bình mẫu, phương sai mẫu hiệu chỉnh.

5. Cho vec-tơ X chứa n giá trị quan sát, phân vị thứ p được xác định như sau

   - Sắp xếp dữ liệu theo thứ tự tăng dần (từ nhỏ đến lớn).
   - Tính chỉ số i:

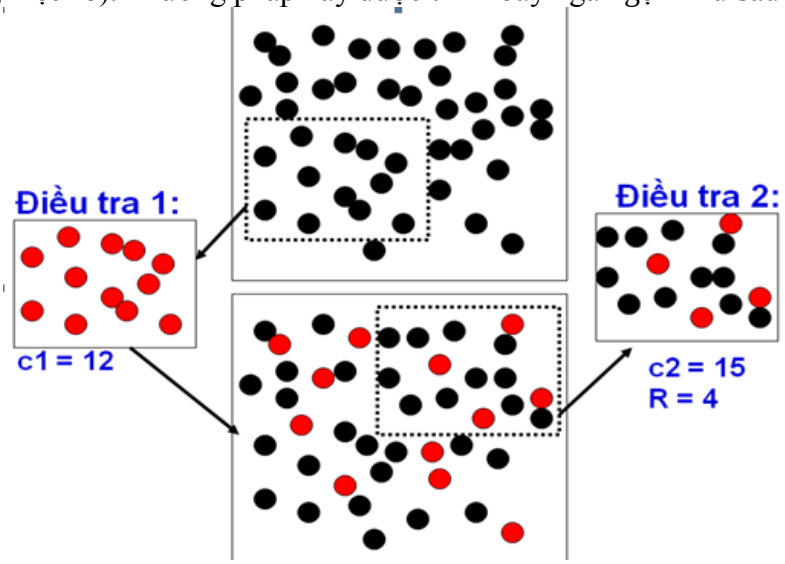
$$i = \left(\frac{p}{100}\right) n$$

   - Nếu i không phải là số nguyên, làm tròn i. Phân vị thứ p chính là giá trị nằm ở vị trí thứ i đã được làm tròn.

     Nếu i nguyên, phân vị thứ i chính là giá trị trung bình của 2 giá trị nằm ở vị trí thứ i và thứ i + 1.

Hãy viết hàm **phanvi(X, P)** cho kết quả là phân vị thứ p từ vec-tơ X.

6. Phương pháp Capture-Recapture thường được dùng để ước lượng kích cở của quần thể (Ví dụ: Số cá trong một hồ). Phương pháp này được trình bày ngắn gọn như sau :



$$\frac{c_1}{N} \simeq \frac{R}{c_2}$$

Nên ta có thể dùng N^* để ước tính kích thước quần thể :

$$N^* = \frac{c_1 c_2}{R}$$

Để tăng tính chính xác thì ta điều chỉnh công thức trên , để N^* trở thành một ước lượng không chệch của N

$$N^* = \frac{(c_1+1)(c_2+1)}{R+1} - 1$$

$$\text{var}(N^*) = \frac{(c_1+1)(c_2+1)(c_1-R)(c_2-R)}{(R+1)^2(R+2)}$$

Vậy khoảng tin cậy 95% của kích cở quần thể là

$$\left(N^* - 1.96\sqrt{\text{var}(N^*)}, N^* + 1.96\sqrt{\text{var}(N^*)} \right)$$

Hãy viết hàm mô phỏng phương pháp trên, hàm nay có:
Input : Kích cỡ thât sự của quần thê , số cá thể lấy lần 1 , số cá thể lấy lần 2
Output: Kích cở quần thể được ước tính bằng phương pháp C-R, Khoảng tin cậy 95%