UNIVERSIDAD NACIONAL DE LOJA

PERIODO ACADÉMICO: OCTUBRE 2019 MARZO 2020
PRACTICA # 4
ASIGNATURA: SIMULACIÓN
RESULTADO DE APRENDIZAJE DE LA PRÁCTICA: Entiende las estructuras básicas de programación en R
TIEMPO PLANIFICADO: 3 HORAS
NUMERO DE ESTUDIANTES: Sexto ciclo (Paralelo A)

1. **Nombre: Deiby Patricio Calva** **Fecha: 12/12/2019**

2. **TEMA: Programación básica en R**

3. **OBJETIVOS:**

   - Comprende las estructuras básicas if, ifelse, for.
   - Comprende el uso de vectorización.
   - Usa los conocimientos aprendidos en teoría para su posterior aplicación práctica.

4. **RECURSOS NECESARIOS:**

   - R-studio.
   - Computador de Laboratorios

5. **INSTRUCCIONES:**

   - Prohibido consumo de alimentos
   - Prohibido equipo de diversión, celulares etc.
   - Prohibido jugar
   - Prohibido mover o intercambiar los equipos de los bancos de trabajo
   - Prohibido sacar los equipos del laboratorio sin autorización.
   - Ubicar los equipos y accesorios en el lugar dispuesto por el responsable del laboratorio, luego de terminar las prácticas.
   - Uso adecuado de equipos

6. **ACTIVIDADES POR DESARROLLAR:**

   1. What will this conditional expression return?

```r
x <- c(1,2,-3,4)

if(all(x>0)){
  print("All Postives")
} else{
  print("Not all positives")
}
```

Rspta: Comprueba con la función 'all' si todos los valores del vector x son positivos

   2. Which of the following expressions is always FALSE when at least one entry of a logical vector xis TRUE?
      A. all(x)
      B. any(x)
      C. any(!x)
      D. all(!x)

3. The function nchar tells you how many characters long a character vector is. Write a line of code that assigns to the object new_names the state abbreviation when the state name is longer than 8 characters.

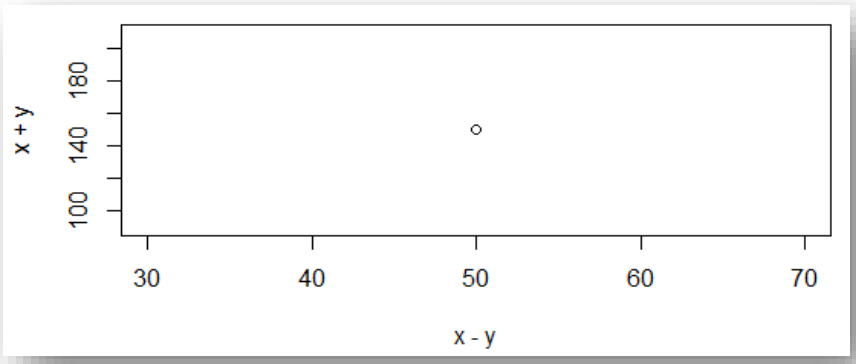
```
22  library(dslabs)
23
24  new_names <- ifelse(nchar(murders$state)>8,#Asigna la abreviatura del estado cuando el nombre de
25  murders$abb,murders$state)              #estado tiene más de 8 caracteres
26  new_names
```

4. Create a function sum_n that for any given value, say n, computes the sum of the integers from 1 to n (inclusive). Use the function to determine the sum of integers from 1 to 5,000.

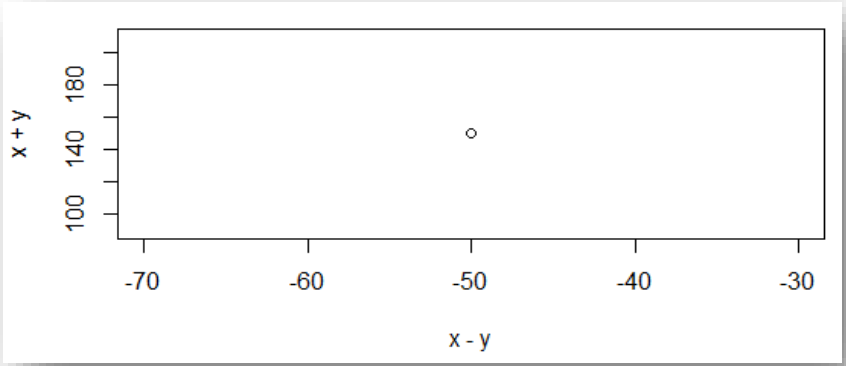
```
33  sum_n <- function(x){
34    sum(1:x)
35  }
36  # Determinar la suma de los números enteros del 1 al 5000
37  sum_n(5000)
```

5. Create a function altman_plot that takes two arguments, x and y, and plots the difference against the sum.

```
51  altman_plot <- function(x,y){
52    plot(x-y,x+y)
53  }
54  altman_plot(100,50)
```




```
58  altman_plot <- function(x,y){
59    plot(x-y,x+y)
60  }
61  altman_plot(50,100)
```



6. After running the code below, what is the value of x?

```
x <- 3
my_func <- function(y){
  x <- 5
  y+5
}
#  y luego `print (x)` . Tendrás la respuesta
print (x)
        #No se usa x en la función, por lo tanto x=3.
```

7. Write a function compute_s_n that for any given **n** computes the sum $S_n = 1^2 + 2^2 + 3^2 + 4^2 + \ldots + n^2$ Report the value of the sum when n=10.

```
compute_s_n <- function(n){
  x<-1:n
  sum(x^2)
}
compute_s_n(10)
```

Resultado: 385

8. Define an empty numerical vector `s_n` of size 25 using `s_n <- vector("numeric", 25)` and store in the results of $S_1, S_2, \ldots S_{25}$ using a for-loop

```
s_n <-vector("numeric",25)
for (i in 1:25) {
  s_n[i] <-i
}
s_n
```

Resultado:

```
[1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
17 18 19 20 21
[22] 22 23 24 25
```

9. Repeat exercise 8, but this time use sapply.

```
s_n <-vector("numeric",25)
for (i in 1:25) {
  s_n[i]<-sapply(i,sqrt)
}
s_n
```

Resulatdo:

```
[1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751
 [8] 2.828427 3.000000 3.162278 3.316625 3.464102 3.605551 3.741657
[15] 3.872983 4.000000 4.123106 4.242641 4.358899 4.472136 4.582576
[22] 4.690416 4.795832 4.898979 5.000000
```

10. Repeat exercise 8, but this time use map_dbl.

```
install.packages("purrr")
library(purrr)
s_n <-vector("numeric",25)
for (i in 1:25) {
  s_n[i]<-map_dbl(i,sqrt)
}
s_n
```

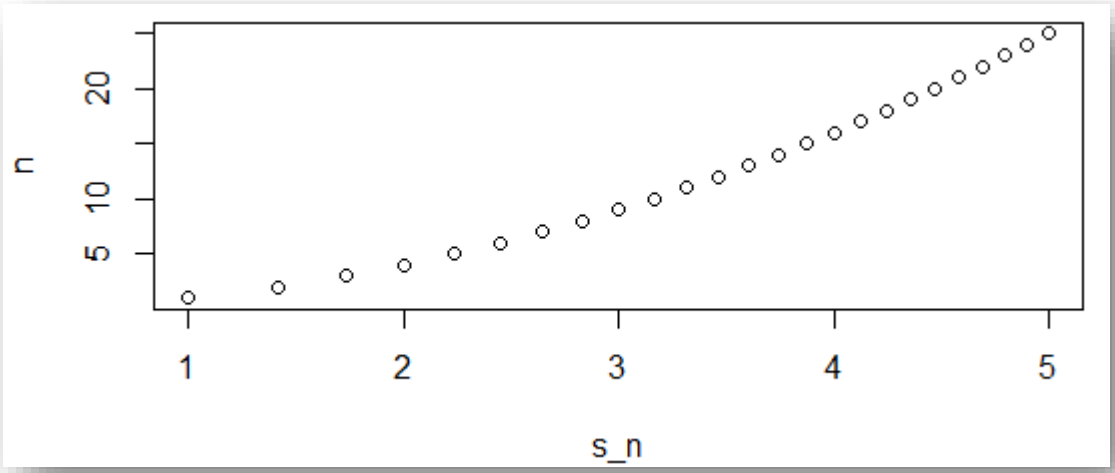
Resulatdo:

```
 [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751
 [8] 2.828427 3.000000 3.162278 3.316625 3.464102 3.605551 3.741657
[15] 3.872983 4.000000 4.123106 4.242641 4.358899 4.472136 4.582576
[22] 4.690416 4.795832 4.898979 5.000000
```

11. Plot Sn versus n. Use points defined by n=1,...,25.

```r
s_n <-vector("numeric",25)
for (i in 1:25) {
  s_n[i]<-map_dbl(i,sqrt)
}
n<-1:25
plot(s_n,n)
```

Resultado:



12. Confirm that the formula for this sum is Sn=n(n+1)(2n+1)/6.

```r
#  Definir la función
compute_s_n <- function(n){
  x <- 1:n
  sum(x^2)
}
# Definir el vector de n
n <- 1:25
# Definir el vector para almacenar datos
s_n <- vector("numeric", 25)
for(i in n){
  s_n[i] <- compute_s_n(i)
}
#  Verifique que s_n sea idéntico a la fórmula dada en las instrucciones
identical (s_n, n*(n+1)*(2*n+1)/6)
```

Resultado: True

6. INVESTIGACIÓN COMPLEMENTARIA

Investigar acerca de las funciones `lapply`, `tapply`, `mapply`, `vapply`, y `replicate`

Function Lapply()

La función lapply () es útil para realizar operaciones en objetos de lista y devuelve un objeto de lista de la misma longitud del conjunto original. lappy () devuelve una lista de la misma longitud que el objeto de lista de entrada, cada uno de los cuales es el resultado de aplicar FUN al elemento correspondiente de la lista. lapply () toma la lista, el vector o el marco de datos como entrada y da salida en la lista. [1]

```
149  s_n <-vector("numeric",5)
150  for (i in 1:5) {
151    s_n[i]<-lapply(i,sqrt)
152  }
153  s_n
```

Resulatdo:

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

[[2]]
[1] 1.414214

[[3]]
[1] 1.732051

[[4]]
[1] 2

[[5]]
[1] 2.236068
```

FunctionTapply()

Tapply () calcula una medida (media, mediana, min, max, etc.) o una función para cada variable de factor en un vector. Es una función muy útil que le permite crear un subconjunto de un vector y luego aplicar algunas funciones a cada uno de los subconjuntos.
Parte del trabajo de un científico de datos o investigadores es calcular resúmenes de variables. Por ejemplo, mida el promedio o datos grupales basados en una característica. La mayoría de los datos se agrupan por ID, ciudad, países, etc. Resumir sobre el grupo revela patrones más interesantes

Para entender cómo funciona, usemos el conjunto de datos de iris. Este conjunto de datos es muy famoso en el mundo del aprendizaje automático. El propósito de este conjunto de datos es predecir la clase de cada una de las tres especies de flores: Sepal, Versicolor, Virginica. El conjunto de datos recopila información para cada especie sobre su longitud y ancho [2]

```
159  # FUCTION TAPPLY IN R
160  attach(iris)
161  # longitud media del sépalo por especie
162  tapply(iris$Sepal.Length, Species, mean)
```

El primer argumento de la función tapply toma el vector para el que necesitamos realizar la función. el segundo argumento es un vector por el cual necesitamos realizar la función y el tercer argumento es la función, aquí es malo. Entonces la salida será:

Resultado:

```
    setosa versicolor  virginica
     5.006      5.936      6.588
```

Function Mapply

Mapply es una versión multivariada de sapply. mapply aplica FUN a los primeros elementos de cada (...) argumento, los segundos elementos, los terceros elementos, etc.

es decir, para cuando tiene varias estructuras de datos (por ejemplo, vectores, listas) y desea aplicar una función a los primeros elementos de cada uno, y luego a los segundos elementos de cada uno, etc., coaccionando el resultado a un vector / matriz como en sapply

Esto es multivariante en el sentido de que su función debe aceptar múltiples argumentos.

Ejemplos:

```
165  # FUCTION Mapply in R
166  mapply(sum, 1:4, 1:4, 1:4)
167
```

Mapply resume todos los primeros elementos (1 + 1 + 1), resume todos los segundos elementos (2 + 2 + 2) y así sucesivamente para que el resultado sea:

Salida o Resultado:

```
[1]  3  6  9 12
```

Function Vapply

La función vapply en R es similar a sapply, pero tiene un tipo de valor de retorno previamente especificado, por lo que puede ser más seguro (y a veces más rápido) de usar.

```
169  # FUCTION VAPPLY in R
170  vapply(1:5, sqrt, 1i)
```

La salida será:

```
[1] 1.000000+0i 1.414214+0i 1.732051+0i 2.000000+0i 2.236068+0i
```

Function Replicate

Rep() is the function in R that replicates the values. [3]

La sintaxis básica para crear un ciclo de repetición en R es:

1. rep(value,number_of_times)
2. rep(sequence,each,number_of_times)

Ejemplos:

```
173  #Ejemplo_1 replicar valores con un número específico de veces
174  rep(1,10)
```

```
Salida: [1] 1 1 1 1 1 1 1 1 1 1
```

En el ejemplo anterior 2 se replica 10 veces

```
177  # Ejemplo_ 2  Función replicar en R con atributo de longitud
178  rep(1:4,len=20)
```

Salida: [1] 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4

En el ejemplo anterior, la secuencia 1 a 4 se replica hasta que la longitud alcanza los 20 elementos.

7. DISCUSIÓN

Mediante el estudio del lenguaje R podemos aprender métodos para realizar análisis de datos, pero sobre todo representar gráficamente esos análisis de dato.

En este cuarto laboratorio se realiza una investigación complementaria de funciones apply (). La función apply () es la más básica de todas las colecciones. También aprenderemos sapply (), lapply () y tapply (). La colección de aplicación se puede ver como un sustituto del bucle.

Para manejar correctamente R se necesitan unos conocimientos previos en programación, pero cuenta con la ventaja de que está formado por un conjunto de herramientas muy flexibles que pueden ampliarse fácilmente mediante paquetes, librerías o definiendo nuestras propias funciones.

8. CONCLUSIONES

- Al haber concluido esta práctica se tiene conocimiento de las herramientas básicas para utilizar R como un lenguaje de programación. Desde los conceptos más básicos, hasta la definición de funciones, además R nos permite el cálculo de predicciones basados en estadísticas, no solo mediante código sino gráficamente, lo que en otros lenguajes de programación resultaría un código muy extenso y complejo.
- Se llega a concluir que R un lenguaje potente con un gran objetivo orientado a la estadística es por eso que si nos profundizamos en el podremos conocerlo y analizarlo más, para así poder aplicarlo en nuestra carrera.

9. RECOMENDACIONES

- Revisar la interfaz gráfica del programa para así poder conocerlo y poder compartir opiniones y experiencias en clases, interactuando entre compañeros e ingeniero
- Para programar en R, necesitamos conocer el uso de sus funciones, el tipo objeto al que puede aplicárseles, sus argumentos, así como los valores que requieren ser definidos.

10. BIBLIOGRAFÍA:

[1] D. Science, «Apply Function in R – apply vs lapply vs sapply vs mapply vs tapply vs rapply vs vapply,» [En línea]. Available: http://www.datasciencemadesimple.com/apply-function-r/. [Último acceso: 11 Diciembre 2019].

[2] Guru, «Función apply (), lapply (), sapply (), tapply () en R con ejemplos,» [En línea]. Available: https://www.guru99.com/r-apply-sapply-tapply.html#2. [Último acceso: 12 Diciembre 2019].

[3] DataScience, «The world of Analytics and Data Science: Replicate Function in R,» [En línea]. Available: http://www.datasciencemadesimple.com/repeat-and-replicate-function-in-r/. [Último acceso: 12 Diciembre 2019].

Firma del Presidente de Curso de Sexto A

Ing. Marlon Santiago Viñan Ludeña Mg. Sc

DOCENTE CIS