Clase de programación en R - $N^0.4$

Victor Augusto Lizcano Sandoval

November 23, 2021

Entradas de usuario

Para recibir alguna entrada o información de un usuario hacemos uso la función *readline()* y del operador *prompt*. La función *readline()* retornará un elemento de tipo carácter. Si se desea una salida en números, toca hacer la conversión.

Entradas de usuario - Ejemplo

```
print("Hola, ¿cómo estás?")
cat("\n")
Nombre = readline(prompt ="¿Cómo te llamas? ")
cat("\n")
print(paste("Mucho gusto ", Nombre, " Me llamo Darth
Vader")) cat("\n")
Edad = as.integer(readline(prompt ="¿Por cierto ...
¿cuál es tu edad ? "))
cat("\n")
print(paste("Tienes ",Edad, " ?", " Wow, soy mucho
mayor que tu" ))
```

Entradas de usuario - Taller

Un método matemático muy famoso para estimar el número pi (π) son las series infinitas de Euler (o problema de Basilea). Este método consiste en sumar los inversos cuadrados de números enteros positivos para obtener un valor equivalente a $\frac{\pi}{6}$.

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \cdots = \frac{\pi^2}{6} \tag{1}$$

$$\pi = \sqrt{6\left(\sum_{n=1}^{\infty} \frac{1}{n^2}\right)} \tag{2}$$

Entradas de usuario - Taller

Estimar el número π con R empleando el problema de Basilea. Para ello debe crear una función vacia y la función *readline()* para pedir la extensión la longitud de la serie.

Entradas de usuario - Taller

```
options(digits=15)

calcularPI = function(n){
n = as.integer(readline(prompt="Ingrese un número
entero positivo que abarque la totalidad de los valores de
la sumatoria Pi: "))
c = rep(0,n)
for( i in 1:n){
c[i] = 1/(i^2)}
PIValor = sqrt(6*sum(c))
return(PIValor)}

calcularPI()
```

Función switch()

La función *switch ()* en R evalua una expresión con elementos de una lista. Si el valor evaluado de la expresión coincide con el elemento de la lista, se devuelve el valor correspondiente.

Función switch() - Ejemplo

switch(2,"Rojo","Verde", "Amarillo", "Azul")

switch("Color", "Color" = "Rojo", "Forma" = "Circulo", "Longitud" = 15)

Función switch() - Ejemplo

```
val1 = 6
val2 = 7
val3 = "s"
resultado = switch(
val3,
"s"= cat("Suma =", val1 + val2),
"r"= cat("Resta =", val1 - val2),
"d"= cat("Division = ", val1 / val2),
"M"= cat("Multiplicacion =", val1 * val2),
"m"= cat("Modulo =", val1 %% val2),
"p"= cat("Potencia =", val1 ∧ val2)
)
print(result)
```

Vectores

Un vector es simplemente una lista de elementos que son del **mismo tipo**.

frutas = c("Peras", "Manzanas", "Piñas", "Bananos")
ValLog = c(TRUE, FALSE,TRUE)
Secuencias = seq(1,10,0.5)
Numeros = c(1,3,4,6,3,2,7,8,9)

Vectores - Filtrar

```
Nombres = c("Ana", "Maria", "Juan", "Pedro", "Eva")
Nombres[3]
length(Nombres)
```

Listas

Una lista es una colección de datos ordenados y modificables de **diferentes tipos**.

frutas = list("Peras", "Manzanas", "Piñas", "Bananos")
frutas[3]

Listas

```
Listas = list(Frutas=c("Peras", "Manzanas", "Piñas",
"Bananos"), Nombres = c("Juan", "Pedro", "Ana"),
Numeros = c(1:15))

Listas[[1]][[2]]
length(Listas)
```

Listas - Añadir elementos

```
Frutas = list("Pera","Manzana", "Banano")
append(Frutas, "Naranja")
Frutas = append(Frutas, "Mandarina", after=2)

Listas = list(Frutas=c("Peras", "Manzanas", "Piñas",
"Bananos"), Nombres = c("Juan", "Pedro", "Ana"),
Numeros = c(1:15))
Listas = append(Listas[[2]], "Maria", after=2)
```

Listas - remover elementos

```
Frutas = list("Pera","Manzana","Mandarina",
"Banano","Naranja")
Frutas= Frutas[-2]
Listas = list(Frutas=c("Peras", "Manzanas", "Piñas",
"Bananos"), Nombres = c("Juan", "Pedro", "Ana"),
Numeros = c(1:15))
Listas = Listas[[3]][-10]
```

Listas - remover elementos

```
Frutas = list("Pera","Manzana","Mandarina",
"Banano","Naranja")
Frutas= Frutas[-2]
Listas = list(Frutas=c("Peras", "Manzanas", "Piñas",
"Bananos"), Nombres = c("Juan", "Pedro", "Ana"),
Numeros = c(1:15))
Listas = Listas[[3]][-10]
```

Listas - rangos

```
Listas = list(Frutas=c("Peras", "Manzanas", "Piñas",
"Bananos"), Nombres = c("Juan", "Pedro", "Ana"),
Numeros = c(1:15))
(Listas)[[2]][1:3]
```

Listas - concatenar

```
Listas = list(Frutas=c("Peras", "Manzanas", "Piñas",
"Bananos"), Nombres = c("Juan", "Pedro", "Ana"),
Numeros = c(1:15))
Listas2 = list(Animales=c("Perro","Gato","Conejo"))
NuevaLista = c(Listas, Listas2)
```

Listas - blucles

```
Listas = list(Frutas=c("Peras", "Manzanas", "Piñas",
"Bananos"), Nombres = c("Juan", "Pedro", "Ana"),
Numeros = c(1:15))

for (x in Listas[[2]]) {
print(x)
}
```

Matrices

Una matriz es un conjunto de datos (**de un solo tipo**) bidimensionales con columnas y filas.

Matrices-Ejemplo

LaMatriz = matrix(1:12, ncol=3, nrow=4, byrow= TRUE)

Matrices-Filtrar

```
LaMatriz[2,2] #Elementos
LaMatriz[2,] # Filas
LaMatriz[,2] # Columnas
LaMatriz[c(1,3),] # Filas especificas
LaMatriz[,c(1,3)] # Columnas especificas
```

Matrices-agregar filas y columnas

Lamatriz = cbind(Lamatriz, c(4,7,10,13)) #Agregar columna
Lamatriz = rbind(Lamatriz, c(13,14,15,16)) #Agregar fila

Matrices-remover filas y columnas

Lamatriz = LaMatriz[-c(1),-c(1)] #Remover fila 1 y columna 1

Matrices-Chequear la existencia de algún elemento

8%in%LaMatriz

Matrices-dimensiones y longitudes de elementos

dim(LaMatriz)
nrow(LaMatriz)
ncol(LaMatriz)
length(LaMatriz)

Matrices - bucles en matrices

```
for (filas in 1:nrow(LaMatriz)) {
for (columnas in 1:ncol(LaMatriz)) {
print(LaMatriz[filas, columnas])
}
}
```

Matrices - Operaciones (Transpuesta)

```
A = matrix(c(10, 8, 5, 12), ncol = 2, byrow = TRUE)
B = matrix(c(5, 3,15, 6), ncol = 2, byrow = TRUE)
t(A)
t(B)
```

Matrices - Operaciones (suma-resta)

A+B
A-B

Matrices - Operaciones (Multiplicación)

```
2*A # Multiplicar por un escalar
A*B # Multiplicar elemento a elemento
A%*%B # Multiplicación matricial
crossprod(A, B) # Producto cruzado t(A)%*% B
tcrossprod(A, B) # Producto cruzado A%*%t(B)
```

Matrices - Operaciones (potencia)

Para ello instalamos la siguiente libreria:

```
install.packages("expm", dependencies=TRUE)
library(expm)
A %^% 2
```

Matrices - Operaciones (determinante)

det(A)
det(B)

Matrices - Operaciones (inversa)

solve(A)
solve(B)
solve(A, B) # Resuelve un sistema de ecuaciones A%*%
X =B

Matrices - Operaciones (diagonal)

```
diag(A)
diag(B)
diag(4) # Genera una matriz identidad
```

Matrices - Operaciones (autovalores)

```
eigen(A)$values
eigen(B)$values
eigen(A)$vectores
eigen(B)$vectores
```

Arreglos

Son matrices con más de dos dimensiones.

```
Arreglo = c(1:24)
Arreglo = array(Arreglo, dim = c(4, 3, 2))
Arreglo[,,2]
```

Dataframes

Son datos que se muestran en formato de tabla. A diferencia de las matrices, sus columnas (o campos), pueden contener datos de diferente tipo.

Dataframes - ejemplo

```
Df = data.frame(Genero =
c("Hombre","Mujer","Mujer","Mujer", "Hombre"), Edad
= c(25,28,26,22,27), Ciudad=
c("Cali","Bogotá","Medellín","Barranquilla","Bucaramanga"
)
```

Dataframes - ejemplo (resumen estadístico y estructura)

```
summary(Df)
str(Df)
```

Dataframes - ejemplo (Filtros por columna)

Df[[1]]
Df[1]
Df[["Genero"]]
Df["Genero"]
Df$Genero
Los demás filtros y funciones empleadas en matrices aplican para los dataframes.

Dataframes - ejemplo (Operaciones)

Df$EdadPor2 = Df$*2

Dataframes - ejemplo (Nombre Columna)

colnames(Df)[4] = c("Edad duplicada")

Dataframes - ejemplo (Nombre fila)

rownames(Df) = c("Dato 1", "Dato 2","Dato 3","Dato 4","Dato 5")

Dataframes - ejemplo (Cambiar valores)

```
Df$Edad[Df$Edad==28]=25
Df$Edad[1]=29
```

Factores

Los factores se utilizan para categorizar datos.

Factores-Ejempo

```
GeneroMusical <- factor(c("Jazz", "Rock", "Clasica",
"Clasica", "Pop", "Jazz", "Rock", "Jazz"))
levels(GeneroMusical)
```

Fechas-Ejempo

```
navidad=as.Date("2013-12-25")
navidad=as.Date("25/12/2013",format="%d/%m/%Y")
navidad=as.Date("25-dec-13",format="%d-%b-%y")
navidad=as.Date("25 December 2013",format="%d %B
%Y")
```

Fechas-Ejempo

Table: Formatos de fecha en R

Simbolo	Significado
%d	día (numérico, de 0 a 31)
%a	día de la semana abreviado a tres letras
%A	día de la semana (nombre completo)
%m	mes (numérico de 0 a 12)
%b	mes (nombre abreviado a tres letras)
%B	mes (nombre completo)
%y	año (con dos dígitos)
%Y	año (con cuatro dígitos)

Fechas-Ejempo

```
Fecha1 = seq(as.Date("2020-01-01"),
as.Date("2020-12-31"), by="days")
Fecha1 = seq(as.Date("2020-01-01"),
as.Date("2020-12-31"), by="weeks")
Fecha1 = seq(as.Date("2020-01-01"),
as.Date("2020-12-31"), by="2 weeks")
Fecha2 = seq(as.Date("2020-01-01"),
as.Date("2020-12-31"), by="months")
Fecha3 = seq(as.Date("2020-01-01"),
as.Date("2020-12-31"), by="quarters")
Fecha4 = seq(as.Date("2000-01-01"),
as.Date("2020-12-31"), by="years")
```

Fechas-Ejempo

```
format(Fecha1, "%Y")
format(Fecha1, "%y")
format(Fecha1, "%m")
format(Fecha1, "%d")
Sys.Date()
```

Fechas-Ejempo

```
dia1=as.Date("25/12/2012",format="%d/%m/%Y")
dia2=as.Date("20/1/2013",format="%d/%m/%Y")
dia3=as.Date("25/12/2013",format="%d/%m/%Y")
dia3-dia1
dia3-dia2
dia2-dia1
```

Fechas-Ejempo

```
difftime(dia3, dia1, units = "weeks")
difftime(dia3, dia1, units = "days")
difftime(dia3, dia1, units = "hours")
difftime(dia3, dia1, units = "mins")
difftime(dia3, dia1, units = "secs")
dia3+10
dia3+42
```

Fechas-Ejempo

```
dias=as.Date(c("1/10/2005","2/2/2006",
"3/4/2006","6/8/2006"),format="%d/%m/%Y")
diff(dias)
```

Calcula la diferencia, en días, entre los términos sucesivos de un vector de fechas

Fechas-Ejempo

seq(dia1,dia3,length=10)
seq(dia1,dia3,by=15)

Fechas-Ejempo

```
FechaHora = as.POSIXct("01/10/1983
22:10:00",format="%d/%m/%Y %H:%M:%S")
FechaHora = as.POSIXct("01/10/1983
22:10:00",format="%d/%m/%Y %H:%M:%S",
tz="PDT")
Sys.time()
```

Objetos temporales-Ejempo

miST <- ts(c(1:72), start=c(2009, 1), end=c(2014, 12), frequency=12)
en *frequency* 12 son meses, 4 son trimestres, 1 son años.

Objetos temporales-Ejempo

```
miST2 <- stl(miST, s.window="period")
miST3 <- window(miST, start=c(2014, 6), end=c(2014,
12))
```