Actividad 1: Ana Sofia Santos Tedim Sousa Pedrosa

Ana P. Tedim

2025-12-11

Actividad 1 - Iniciación a R: Programación básica y lógica

7. Crea las siguientes variables:

7.1. Vector10. Esta variable debe ser un vector que almacene los diez primeros números enteros. Indica 3 formas distintas de definir esta variable, explorando las funciones seq() y rep().

```
# Método A - usando el operador :
Vector10_A <- 1:10
print(Vector10_A)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
# Método B - usando seq()
Vector10_B <- seq(from = 1, to = 10, by = 1)
print(Vector10_B)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
# Método C - usando rep()
Vector10_C <- rep(1:10, times = 1)
print(Vector10_C)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

7.2. Vector3. Esta variable debe ser un vector que almacene una variable de tipo carácter, una de tipo numérico y una de tipo Booleano. ¿De qué clases son los valores almacenados? Ayuda: explorar la función class().

```
Vector3 <- c("Introducción a Programación", 41, TRUE)
print(Vector3)
```

```
## [1] "Introducción a Programación" "41"
## [3] "TRUE"
```

```r
class(Vector3)
```

```
## [1] "character"
```

Para ver el tipo de dato se usó la función class(). Que en este caso nos devuelve el resultado de character, aun que sabemos que hay otras clases de datos que hemos introducido en el vector. Sin embargo, los vectores en R son estructuras homogéneas, es decir que almacenan el mismo tipo de dato. Cuando se combinan diferentes tipos de datos (como es el caso), R los convierte a la clase más flexible que es siempre character (de menor a mayor flexibilidad: logical < integer < numeric < character). En este caso 41 pasaría a "41" y TRUE a "TRUE"

7.3. Lista3. Esta variable debe ser una lista que almacene una variable de tipo carácter, una de tipo numérico y una de tipo Booleano. ¿De qué clases son los valores almacenados? Ayuda: explorar la función class().

```r
Lista3 <- list("Introducción a Programación", 41, TRUE)

class(Lista3)
```

```
## [1] "list"
```

```r
sapply(Lista3, class)
```

```
## [1] "character" "numeric"   "logical"
```

Para crear una lista con elementos de tipos de datos diferentes se usó la función list() que crea una lista de 3 elementos en que el primero es de tipo "character" el segundo del tipo "numeric" y el tercero del tipo "logical" o booleano.

Si usamos directamente la función class(Lista3) nos devuelve únicamente que lo que hemos creado es una lista. Para verificar el tipo de dato de cada elemento de la lista tenemos que usar sapply() en la Lista3 indicando que lo que queremos saber es la class(). Esto nos devuelve el tipo de dato de cada elemento de la lista creada.

7.4. Matriz_Resultado. Esta matriz debe almacenar el resultado de sumar la matrizA y la matrizB (ambas son matrices de 4 filas x 3 columnas).

```r
matrizA <- matrix(1:12, nrow = 4, ncol = 3)
matrizB <- matrix(12:23, nrow = 4, ncol = 3)
Matriz_Resultado <- matrizA + matrizB
print(Matriz_Resultado)
```

```
##      [,1] [,2] [,3]
## [1,]   13   21   29
## [2,]   15   23   31
## [3,]   17   25   33
## [4,]   19   27   35
```

8. Crea el vector vector_random que almacene 50 números aleatorios. Ayuda: para obtener valores random puedes usar las funciones runif(), rnorm(), rexp(), sample().

• Encuentra el valor máximo y mínimo del vector. • Calcula la media y la mediana. • Determina la desviación estándar. • Ordena los valores del vector en orden ascendente y descendente. • Calcula la suma y el producto de todos los elementos del vector.

```
# Usamos la función runif() (uniforme) como ejemplo de como se podria crear este tipo de vector
set.seed(10)  # para reproducibilidad
vector_random <- runif(50, min = 10, max = 200)  # 50 valores entre 0 y 100

# Para encontrar el valor máximo y mínimo usamos:
max_vector <- max(vector_random)
min_vector<- min(vector_random)
cat("El número más elevado del vector es:", max_vector, "\n")
```

```
## El número más elevado del vector es: 191.3842
```

```
cat( "El número más pequeño del vector es:", min_vector, "\n")
```

```
## El número más pequeño del vector es: 12.74244
```

```
# Para calcular la media y la mediana del vector_random usamos:
media_vector <- mean(vector_random)
mediana_vector <- median(vector_random)
cat("La média de este vector es:", media_vector, "\n")
```

```
## La média de este vector es: 97.1342
```

```
cat("La mediana de este vector es:", mediana_vector, "\n")
```

```
## La mediana de este vector es: 91.29312
```

```
# Para calcular la desviación estándar del vector_random usamos:
sd_vector <- sd(vector_random)
cat("La desviación estándar del vector es:", sd_vector, "\n")
```

```
## La desviación estándar del vector es: 50.37533
```

```
# Para ordenar los valores por orden ascendente usamos:
orden_asc_vector <- sort(vector_random, decreasing = FALSE)
cat("Los primeros 5 valores de vector por orden arcendente son:", orden_asc_vector[1:5], "\n")
```

```
## Los primeros 5 valores de vector por orden arcendente son: 12.74244 13.2381 19.86163 26.17583 27.686
```

```
# Para ordenar los valores por orden descendente usamos:
orden_desc_vector <- sort(vector_random, decreasing = TRUE)
cat("Los primeros 5 valores de vector por orden descendente son:",orden_desc_vector[1:5], "\n")
```

```
## Los primeros 5 valores de vector por orden descendente son: 191.3842 180.9682 174.297 169.2747 168.8
```

```r
# Para determinar la suma de todos los elementos del vector usamos:
suma_vector <- sum(vector_random)
cat("La suma total del vector es:", suma_vector, "\n")
```

```
## La suma total del vector es: 4856.71
```

```r
# Para determinar la suma de todos los elementos del vector usamos:
producto_vector <- prod(vector_random)
cat("La producto total del vector es:", producto_vector, "\n")
```

```
## La producto total del vector es: 2.482526e+95
```

9. ¿De qué tipo son las variables almacenadas hasta el momento, globales o locales? Indica cómo se puede eliminar una variable en concreto (ejemplo: vector_random) y cómo todas las variables a la vez.

```r
#Para eliminar únicamente la variable vector_random
rm(vector_random)

#Para eliminar todas la variables creadas hasta el momento y dejar el Global environment limpio
rm(list = ls())
```

Todas la variables que hemos creado en esta actividad hasta el momento son variables globales, ya que ha sido declaradas fuera de cualquier función y son visibles y accesibles en todas las secciones del script. Las variables locales solo son visibles y accesibles dentro de la función en la que se definen, que no es el caso aqui

10. Desarrolla, utilizando sentecias if – else if - else, un bloque de código que lea un nucleótido ingresado por el usuario y determine si es Adenina (A), Timina (T), Citosina (C) o Guanina (G). Si el nucleótido ingresado no es válido, debe informar al usuario del error.

```r
nucleotido_inicial <- readline(prompt = "Introduce un nucleótido (A, T, C, G): ") #Muestra un texto ent
```

```
## Introduce un nucleótido (A, T, C, G):
```

```r
nucleotido <- toupper(nucleotido_inicial) # haz con que el texto introducido por el usuario se convierta
if (nucleotido == "A") {
    print("Es Adenina (A).")
} else if (nucleotido == "T") {
    print("Es Timina (T).")
} else if (nucleotido == "C") {
    print("Es Citosina (C).")
} else if (nucleotido == "G") {
    print("Es Guanina (G).")
} else {
    print("Error: El nucleótido ingresado no es válido (A, T, C, o G).")
}
```

```
## [1] "Error: El nucleótido ingresado no es válido (A, T, C, o G)."
```

11. Desarrolla, utilizando bucles for o while, un bloque de código que calcule la suma de todos los números comprendidos entre 50 y 100 (ambos inclusive).

```
#Suma con for
suma_for <- 0
for (i in 50:100) {
    suma_for <- suma_for + i
}
cat("Suma usando for:", suma_for, "\n")
```

```
## Suma usando for: 3825
```

```
#Suma con while
Suma_while <- 0
i <- 50
while (i <= 100) {
    Suma_while <- Suma_while + i
    i <- i + 1
}
cat("Suma usando while:", Suma_while)
```

```
## Suma usando while: 3825
```

12. Desarrolla, utilizando bucles for o while, un bloque de código que calcule la suma y el promedio de todos los números pares comprendidos entre 1 y 50 (ambos inclusive).

```
suma_pares <- 0      # permite acumular la suma de los números pares
contador_pares <- 0        # cuenta cuántos números pares hay entre 1 y 50

for (i in 1:50) {
  if (i %% 2 == 0) {   # %% calcula el resto: si i es par, i %% 2 == 0 en caso afirmativo suma el número
    suma_pares <- suma_pares + i
    contador_pares <- contador_pares + 1
  }
}

promedio_pares <- suma_pares/contador_pares

cat("La suma de todos los números pares comprendidos entre 1 y 50 es:", suma_pares, "\n")
```

```
## La suma de todos los números pares comprendidos entre 1 y 50 es: 650
```

```
cat("La promedio de todos los números pares comprendidos entre 1 y 50 es:", promedio_pares)
```

```
## La promedio de todos los números pares comprendidos entre 1 y 50 es: 26
```

13. Crea y utiliza una función llamada Deteccion_Nucleotido que realice el bloque de código del Ejercicio 10.

```r
Deteccion_Nucleotido <- function(nucleotido_input) {
    nucleotido_upper <- toupper(nucleotido_input)
    if (nucleotido_upper == "A") {
        return("Es Adenina (A).")
    } else if (nucleotido_upper == "T") {
        return("Es Timina (T).")
    } else if (nucleotido_upper == "C") {
        return("Es Citosina (C).")
    } else if (nucleotido_upper == "G") {
        return("Es Guanina (G).")
    } else {
        return("Error: El nucleótido ingresado no es válido (A, T, C, o G).")
    }
}

cat("El nucleotido introducido es:", Deteccion_Nucleotido("G"))
```

```
## El nucleotido introducido es: Es Guanina (G).
```