Reforzamiento programación (parte 2)

Ejemplo adivinanza

```
adivinanza <- 999
numero_secreto <- sample(c(1:10), 1)
print(numero_secreto)

while (numero_secreto != adivinanza) {
  adivinanza <-  as.integer(readline("Adivina el número: "))

  if (numero_secreto == adivinanza) {
    print("¡acertaste!")
  }

}
```

Búsqueda binaria

Modifiquemos el juego anterior

1. El rango de valores para buscar será todos los enteros entre 1 a 100
2. Cuando el usuario no acierte, debemos indicar si el número secreto es mayor o menor al número ingresado
3. Debemos guardar los intentos realizados y cuando el usuario adivine, debemos escribir "Felicitaciones, acertaste después de n intentos"
4. Si el usuario escribe "stop", el programa se detiene y envía el siguiente mensaje: "Gracias por participar en este apasionante juego"

```
adivinanza <- 999
numero_secreto <- sample(c(1:100), 1)
print(numero_secreto)

contar <- 0
while (numero_secreto != adivinanza) {
  # Identificar si el usuario escribe stop
  instruccion <-  readline("Adivina el número secreto: ")

  if (instruccion == "stop") {
    print("Gracias por participar en este apasionante juego")
    break
  } else {
    adivinanza <- as.integer(instruccion)
  }

  contar <- contar + 1
```

```
  if (numero_secreto == adivinanza) {
    print(paste("¡Felicitaciones! Acertaste después de", contar, "intentos"))
  } else if (numero_secreto > adivinanza) {
    print("El número secreto es mayor")
  } else if (numero_secreto < adivinanza) {
    print("El número secreto es menor")
  }


}
```

Suma 10

Trabajaremos sobre la siguiente lista de números

```
numeros <- list(1, 5, 9, 34, 98, 76)
```

La idea es iterar sobre la lista numeros, sumar 10 a cada elemento y reemplazar el valor original por el resultado de la operación

La función `length` puede ser de utilidad

```
for (i in 1:length(numeros)) {
  numeros[[i]] <- numeros[[i]] + 10
}
```

Suma vector

Queremos una función que reciba un vector numérico y retorne la suma de todos sus elementos.

No está permitido el uso de `sum`

Por ejemplo, si la función recibe el vector `c(1, 2 , 2)`, debería devoler 5

Pista: Puede usar un `for` para sumar los elementos del vector

```
sumar <- function(vector) {
  suma <- 0
  for (i in vector) {
    suma <- suma + i
  }
  return(suma)
}

sumar(runif(10))
```

```
## [1] 5.305706
```

Coeficiente de variación

Vamos a construir una función para calcular el coeficiente de variación.

Interpretación: Relación entre la desviación estándar de un estimador y su media.

No está permitido usar la función cv, pero sí la función sd y la función mean

```r
calcular_cv <- function(vector) {
  cv <- sd(vector) / mean(vector) * 100
  return(cv)
}

calcular_cv(c(1:100))
```

```
## [1] 57.4485
```

Desviación estándar

La idea es no usar sd

Puedes probar tu función con el siguiente vector `rnorm(10000)`

Pista: algunas funciones útiles son:

```r
get_sd <- function(vector) {
  suma_cuadrados <- sum((vector - mean(vector))**2)
  varianza <- suma_cuadrados/ (length(vector) - 1)
  return(sqrt(varianza))

}

set.seed(123)
vec <- rnorm(10000)
get_sd(vec)
```

```
## [1] 0.9986366
```

```r
sd(vec)
```

```
## [1] 0.9986366
```