PASPE 2024: R básico

Tarea 4

Rossana Torres Alvarez

August 23, 2025

# Problema 1. Funciones e if else statements

1. Explica lo que hace esta función y muestra un ejemplo de su uso

f3 <- function(x, y) {  
 rep(y, length.out = length(x))  
}

# La funcion repite el elemento y, tantas veces como la longitud del objeto x. Demostracion:  
  
# x es un vector de 3 elementos  
x <- c(1:3)  
# Verifico  
length(x)

## [1] 3

# y es una variable numerica con valor 1   
y = 1  
  
# Mi funcion me debe imprimir 1 tres veces:  
f3(x = c(1:3), y = 1)

## [1] 1 1 1

1. Escribe una función llamada vectorType, que tenga como input un vector y devuelva la frase: “El vector es numerico” o “El vector es caracter” o “El vector no es ni numerico ni caracter” (Hint: Usa las funciones is.numeric(), is.character())

vectorType = function(vector){  
 if(is.numeric(vector)) {  
 print("El vector es numerico")  
 } else if (is.character(vector)){  
 print("El vector es caracter")  
 } else {  
 print("El vector no es ni numerico ni caracter")  
 }  
}

1. Prueba tu función con los siguientes vectores

x = 1:10  
y = c("Copito", "Firulais")  
z = c(TRUE, FALSE, NA)

1. Explica la diferencia entre un if else statement y la función ifelse() que usamos en el laboratorio 2

# La condición dentro de un if else statement nos devuelve un solo valor logico. En cambio, la condición dentro de la funcion ifelse nos devuelve vectores de valores logicos. Por lo tanto, la funcion ifelse nos permite aplicar tests logicos a vectores, es decir, esta funcion esta vectorizada.

## Problema 2. For loops

1. Escribir un for loop para calcular la media de cada columna en el dataframe mtcars (precargada).

medias <- c()  
print(medias)

## NULL

for (col in 1:ncol(mtcars)) {  
  
 medias[col] <- (mean(mtcars[,col]))  
   
}  
  
# Verificar  
print(medias)

## [1] 20.090625 6.187500 230.721875 146.687500 3.596563 3.217250  
## [7] 17.848750 0.437500 0.406250 3.687500 2.812500

1. Instala el paquete nycflights13. Al cargar el paquete, tendrás disponible una base de datos llamada flights. Verifica qué clase de objeto es la base flights. Si es necesario, conviértela a data.frame. Determina el tipo de cada columna en la base usando un for loop. Si la consola te da un warning lee con cuidado y verifica tu resultado.

#install.packages("nycflights13") # Esto solo se hace una vez en la vida!!  
library(nycflights13)  
  
# Inspecciono  
class(flights)

## [1] "tbl\_df" "tbl" "data.frame"

# Cambio a dataframe  
flights\_df <- data.frame(flights)  
  
tipo <- c()  
  
print(tipo)

## NULL

for (col in 1:ncol(flights\_df)) {  
  
 tipo[col] <- (class(flights\_df[,col]))[1]  
   
}  
  
# Verificar  
print(tipo)

## [1] "integer" "integer" "integer" "integer" "integer" "numeric"   
## [7] "integer" "integer" "numeric" "character" "integer" "character"  
## [13] "character" "character" "numeric" "numeric" "numeric" "numeric"   
## [19] "POSIXct"

1. Escribir un for loop que calcule el número de valores únicos en cada columna del dataframe iris (precargada en R). Si la consola te da un warning lee con cuidado y verifica tu resultado.

unicos <- c()  
  
print(unicos)

## NULL

for (col in 1:ncol(iris)) {  
  
 unicos[col] <- length(unique(iris[,col]))  
   
}  
  
# Verificar  
print(unicos)

## [1] 35 23 43 22 3