Funciones, Bucles y Modelos Estadísticos en Estadística R

Realizado por Juan Pablo Ramirez Correa

UNICISO
WWW.PORTALUNICISO.COM

UNICISO
WWW.PORTALUNICISO.COM

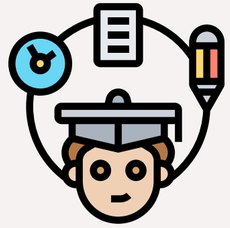
Tabla de Contenido

UNICISO
WWW.PORTALUNICISO.COM

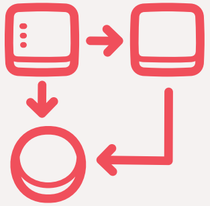
Conceptos clave: funciones, bucles y modelos en R

El lenguaje R es una de las herramientas más utilizadas en **análisis de datos, ciencia de datos y estadística y dominar funciones, bucles y modelos en R** es fundamental, ya que permite escribir código eficiente, reutilizable y escalable, lo que optimiza el flujo de trabajo en proyectos de datos.

Las funciones mejoran la modularidad y reutilización del código.



Los bucles facilitan la ejecución repetitiva de tareas con eficiencia.



Los modelos en R permiten analizar relaciones entre variables y hacer predicciones.

"R **facilita la creación de flujos de trabajo de análisis de datos reproducibles mediante scripts** que documentan cada paso del proceso. Esto es crucial para la transparencia y la verificación de los resultados de la investigación". (Gentleman, 2005).

UNICISO
WWW.PORTALUNICISO.COM

Paquetes esenciales para programación funcional y modelado

R cuenta con paquetes especializados que facilitan la implementación de funciones, el uso de bucles eficientes y la construcción de modelos estadísticos.

Base R Proporciona funciones fundamentales como apply(), lapply() y sapply().

dplyr: Facilita la manipulación de datos con funciones como mutate(), filter(), y group_by().

purrr: Introduce funciones como map(), reduce() y walk() para aplicar funciones de manera eficiente.

El conjunto de paquetes **tidymodels** permite construir, evaluar y ajustar modelos de manera estructurada.

```
install.packages("tidymodels")
library(tidymodels)
```

Cuadro de ejemplo

UNICISO
WWW.PORTALUNICISO.COM

Creación y Uso de Funciones en R

Las funciones en R son bloques de código reutilizables que ejecutan tareas específicas. Su uso permite reducir la repetición de código, mejorar la organización y aumentar la eficiencia en proyectos de análisis de datos.

- Una función recibe entradas (argumentos) y devuelve una salida.
- Ayuda a reducir la repetición de código y mejora la legibilidad.
- Permite estructurar el código en partes modulares y reutilizables.
- En R, las funciones se crean con la estructura:

1. Sintaxis básica de una función en R

En R, las funciones se crean utilizando function() {}. Son bloques de código reutilizables que ejecutan tareas específicas.

```
sumar <- function(a, b) {
  resultado <- a + b
  return(resultado)
}
# Llamar la función
sumar(5, 3)  # Devuelve 8
```

2. Argumentos y retorno de valores en funciones

Definir argumentos en una función:

Los argumentos permiten que una función reciba valores al momento de ser llamada. (X,Y)

```
multiplicar <- function(x, y) {
 return(x * y) }
```

Valores por defecto en los argumentos:

Se pueden definir valores por defecto dentro de la definición del la función. (Expotente =2)

```
elevar_potencia <- function(base, exponente = 2) {
 return(base ^ exponente)}
```

Retorno de valores en funciones:

En R se utiliza **return**() para devolver el resultado de la función.

```
calcular_cuadrado <- function(numero) {
  resultado <- numero^2
  return(resultado)
}

calcular_cuadrado(5)  # Devuelve 25
```

3. Uso de la familia apply para aplicar funciones a estructuras de datos

apply(): Aplicar funciones a filas y columnas especificas.

```
matriz <- matrix(1:9, nrow = 3, ncol = 3)
apply(matriz, 1, sum)  # Suma de cada fila
apply(matriz, 2, mean) # Promedio de cada columna
```

```
lista <- list(a = 1:5, b = 6:10)
lapply(lista, mean)  # Calcula la media de cada elemento de la lista
```

lapply(): Aplicar funciones a listas y devolver una lista.

sapply(): Aplicar funciones a una lista y devolver un vector, una matriz o un array.

```
sapply(lista, mean)  # Retorna un vector con las medias de cada elemento
```

UNICISO
WWW.PORTALUNICISO.COM

Uso de Bucles y Estructuras de Control en R

Los bucles y estructuras de control permiten automatizar tareas repetitivas y tomar decisiones en el código.

Condicionales

if-else: Condiciones básicas.

ifelse(): Evaluación vectorizada.

switch(): Evaluación de múltiples casos.

Bucles

for: Iteración sobre elementos de un vector o lista.

while: Ejecución basada en una condición.

repeat: Repetición infinita hasta que se interrumpe.

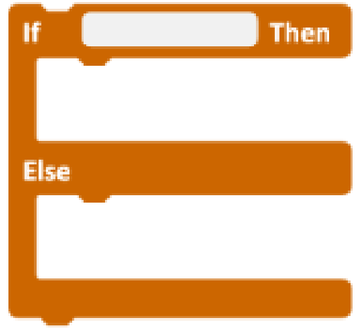
Ejemplos de Condicionales en R

```
x <- 10
if (x > 5) { print("x es mayor que 5")
} else {print("x es menor o igual a 5")}
```

El **If** permite establecer casos para los que se aplica una condición en especifico. Mientras que el **else** establece la condición para el resto de los casos.

```
valores <- c(2, 8, 5, 12)
resultado <- ifelse(valores > 5, "Mayor a 5",
"Menor o igual a 5")
print(resultado)
```

El **Ifelse** permite aplicar condiciones a cada elemento dentro de un vector.




```
opcion <- "B"
resultado <- switch(opcion,
                    "A" = "1", "B" = "2","C" = "3",
                    "Opción no válida")
print(resultado)
```

El **Switch** permite permite ejecutar diferentes acciones (condiciones) en función del valor de una variable.

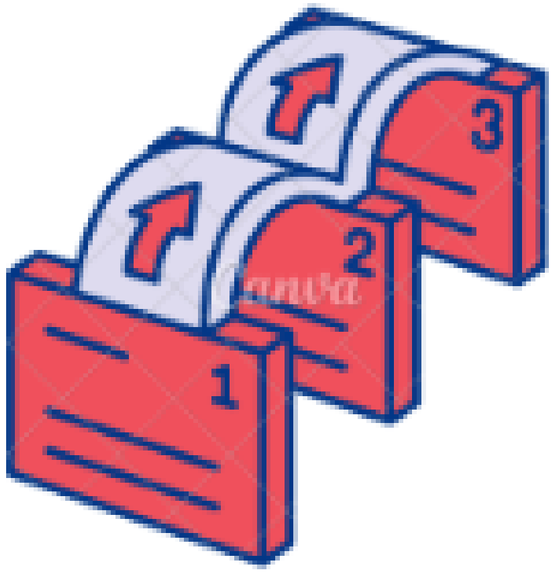
Ejemplos de Bucles en R

```
for (i in 1:5) {
  print(paste("Iteración:", i)) }
```

El **for** permite realizar una iteración (repetición de un proceso) sobre elementos de un vector o una lista.

```
contador <- 1
while (contador <= 5) {
  print(paste("Valor:", contador))
  contador <- contador + 1}
```

El **While** permite realizar una iteración basada en una condición inicial.




```
contador <- 1
repeat {
  print(contador)
  contador <- contador + 1
  if (contador > 5) break}
```

El **Repeat** permite permite ejecutar la misma accción hasta que sea interrumpida por un **break.**

UNICISO
WWW.PORTALUNICISO.COM

Programación Funcional en R

La programación funcional en R permite escribir código más conciso, modular y eficiente mediante el uso de funciones de orden superior, que aceptan otras funciones como argumentos o devuelven funciones como resultado.

Menos código repetitivo: Se evita escribir loops manuales.

Mayor claridad y legibilidad: Se presenta un código mas estructurado y de mejor comprensión.

Eficiencia : Mejor rendimiento en operaciones con grandes volúmenes de datos.

Ejemplo tradicional vs. funcional

Modelo tradicional de bucle

```
numeros <- c(1, 2, 3, 4, 5)
resultado <- numeric(length(numeros))

for (i in seq_along(numeros)) {
  resultado[i] <- numeros[i]^2
}

print(resultado)  # Retorna c(1, 4, 9, 16, 25)
```

Método funcional (sin bucles)

```
resultado <- sapply(numeros, function(x) x^2)
print(resultado)
```

En este caso, ambos códigos buscan realizar el mismo proceso de aplicar la operación numérica del exponente a todos los números presentes dentro de “**numeros”.** Sin embargo el codigo funcional utilizando **sapply** permite reducir la longitud del Código.

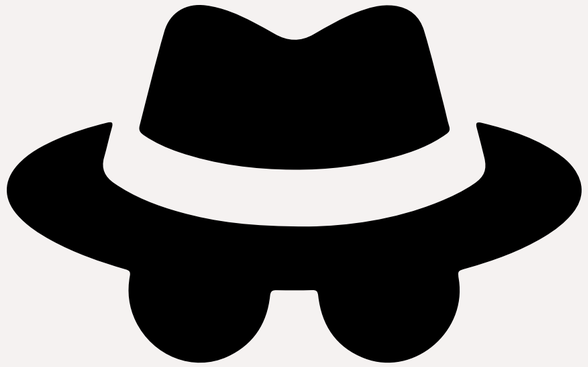
Funciones Anónimas y Aplicación en Datos

UNICISO
WWW.PORTALUNICISO.COM

Las funciones anónimas, son aquellas que no tiene un identificador asociado. Permiten definir operaciones rápidas sin necesidad de crear funciones nombradas.

```
doble <- function(x) x * 2  # Función tradicional
doble(5)  # Retorna 10

# Función anónima:
(function(x) x * 2)(5)  # También retorna 10
```



Ejemplo con programación funcional

```
numeros <- c(5, 10, 15)
resultado <- sapply(numeros, function(x) x * 2)
print(resultado)  # Retorna c(10, 20, 30)
```

Para este caso no se define la función anteriormente antes de implementarla con el "**sapply"** por lo que resulta una función anonima.

Uso del paquete purrr: Transformaciones más eficientes

El paquete **purrr** es parte de la familia **tidyverse** y nos permite trabajar con funciones de manera más eficiente en R. Permite tener un código mas simple y fácil de leer.

Principales usos para purrr:

Escenario	Función de purrr	Ejemplo
Aplicar una función a cada elemento de una lista o vector	map()	map(numeros, ~ .x + 2)
Obtener un vector numérico como resultado	map_dbl()	map_dbl(numeros, ~ .x * 2)
Obtener un vector de texto	map_chr()	map_chr(nombres, toupper)
Filtrar valores lógicos (TRUE/FALSE)	map_lgl()	map_lgl(numeros, ~ .x > 5)

Para verificar el tipo de datos en R se utiliza el código: **class("variable"),** siendo "variable" la variable que se quiere analizar.

Ejemplo de uso de purrr en R

Definición de variables

```
library(purrr)
# Datos de ejemplo
temperaturas <- c(0, 10, 20, 30, 40)
radios <- c(1, 3, 5, 7)
nombres <- c("ana", "juan", "maria", "carlos")
numeros <- c(1, 2, 3, 4, 5, 6)
```

Aplicación del código purrr

```
# Aplicamos funciones con purrr
resultado <- list(
  "Temperaturas °F" = map_dbl(temperaturas, ~ .x * 9/5 + 32),
  "Áreas círculos" = map_dbl(radios, ~ pi * .x^2),
  "Nombres mayúsculas" = map_chr(nombres, toupper),
  "Números pares" = map_lgl(numeros, ~ .x %% 2 == 0)
)

# Mostramos los resultados
resultado
```

Construcción de Modelos Básicos en R

Un modelo es una herramienta matemática que permite explicar y predecir fenómenos basándose en datos. Se construye a partir de una muestra representativa de la población y busca identificar la relación entre una variable respuesta (predecir o explicar) y una o más variables explicativas (factores que pueden influir en la variable respuesta). (Dereck Corcoran, 2020).

Tipos de Modelos:

- **Regresión**:Predice valores numéricos (Ejemplo: Precio de casas).
- **Clasificación**: Predice categorías (Ejemplo: Spam o no spam).

Paquetes útiles:

Paquete	Descripción
stats	Modelos estadísticos básicos en R.
caret	Herramientas para entrenamiento y validación de modelos.
tidymodels	Conjunto de paquetes modernos para modelado en R.

UNICISO
WWW.PORTALUNICISO.COM

Pasos para Construir un Modelo en R

R cuenta con paquetes especializados que facilitan la implementación de funciones, el uso de bucles eficientes y la construcción de modelos estadísticos.

1. Preparación de los Datos:

- **Dividir datos:** en conjunto de entrenamiento (80%) y prueba (20%).
- **Manejar valores faltantes:** y transformar variables si es necesario.
- **Normalizar variables** si los modelos son sensibles a escalas.

2. Selección del Modelo

- **Modelos lineales:** lm() (Regresión Lineal), glm() (Regresión Logística).
- **Modelos más avanzados:** randomForest (bosques aleatorios), xgboost (boosting).

3. Evaluación del modelo

- **Ajustar el modelo** con train() o lm().
- **Evaluar precisión con métricas como:** Error Cuadrático Medio, o Área bajo la Curva ROC
- **Interpetar los resultados:** Coeficientes y significancia.

Introducción a la Regresión Lineal en R

La regresión lineal es un modelo estadístico que describe la relación entre una variable dependiente (respuesta) y una o más variables independientes (predictoras). Se representa por la siguiente ecuación:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

Donde: Y es la variable respuesta, **X** es la variable predictora, **β0** es la intersección, **β1** es la pendiente y **ϵ** es el error.

```
# Cargar datos
data(mtcars)
# Ajustar modelo de regresión lineal
modelo <- lm(mpg ~ hp, data = mtcars)
# Resumen del modelo
summary(modelo)
```

Interpretación de resultados:

- **Coeficiente de la pendiente (β1):** Muestra el cambio esperado en Y por cada unidad de cambio en X.
- **Valor p:** Indica si la relación entre las variables es estadísticamente significativa.
- **R^2:** Mide qué tan bien el modelo explica la variabilidad de los datos (valores cercanos a 1 indican un buen ajuste).

Validación de Modelos en R

La validación es un proceso fundamental en cualquier modelo ya que permite evaluar la precisión y la confiabilidad del modelo y entender si los resultados obtenidos son representativos de la problemática planteada.

Principales Métricas de Evaluación:

Métrica	Descripción	Función en R
RMSE	Error cuadrático medio (menor es mejor).	RMSE() (caret)
MAE	Error absoluto medio (interpretable en unidades de Y).	MAE() (caret)
R^2	Proporción de varianza explicada por el modelo.	summary(lm())
AUC	Área bajo la curva ROC (para clasificación).	roc() (pROC)

Interpretación de resultados:

- **RMSE bajo:** Buen modelo.
- **MAE bajo:** Predicciones cercanas a los valores reales.
- **R^2 alto:** Explicación adecuada de la variabilidad en los datos.

```
# Cargar paquete
library(caret)
# Dividir datos en entrenamiento:
set.seed(123)
trainIndex <- createDataPartition(mtcars$mpg,
p = 0.8, list = FALSE)
trainData <- mtcars[trainIndex, ]
testData  <- mtcars[-trainIndex, ]
# Ajustar modelo de regresión lineal
modelo <- lm(mpg ~ hp, data = trainData)
# Predecir y calcular métricas
predicciones <- predict(modelo, newdata =
testData)
RMSE(predicciones, testData$mpg)
MAE(predicciones, testData$mpg)
summary(modelo)$r.squared  # R2
```

UNICISO
WWW.PORTALUNICISO.COM

Modelos No Lineales en R

Los modelos no lineales en R se utilizan cuando la relación entre las variables no puede ser representada con una simple línea recta. Son útiles en situaciones donde los datos muestran patrones curvos o límites de decisión no lineales. (Crecimiento Poblacional, probabilidades, fenómenos físicos).

Principales Métricas de Evaluación:

Modelo	Uso	Función en R
Regresión Logística	Predicción de categorías (Ejemplo: sí/no).	glm(family = binomial)
Regresión Polinómica	Captura curvaturas en la relación X-Y.	lm(y ~ poly(x, grado))
Modelos No Lineales	Ajuste a formas específicas de datos.	nls()

Interpretación de resultados:

- **Regresión logística:** Devuelve probabilidades en lugar de valores numéricos.
- **Regresión polinómica:** Captura mejor relaciones curvas en los datos. (al usar un polinomio se aproxima mas a las posibles curvas presentes en los datos.

```
# Cargar librerías
library(ggplot2)
# Crear datos simulados
set.seed(123)
df <- data.frame(x = runif(100, 0, 10))
df$y <- ifelse(df$x + rnorm(100, 0, 2) > 5, 1, 0)  # Variable binaria
# Ajustar modelo logístico
modelo_log <- glm(y ~ x, data = df, family = binomial)
# Resumen del modelo
summary(modelo_log)
```

UNICISO
WWW.PORTALUNICISO.COM

Ajuste y Comparación de Modelos

La comparación de modelos en R permite identificar cuál ofrece el mejor ajuste y precisión para los datos. Cada modelo captura las relaciones de manera distinta, por lo que evaluarlos ayuda a tomar decisiones informadas y evitar problemas como el sobreajuste o la falta de representatividad.

```
# Cargar librerías
library(ggplot2)
library(caret)
# Modelo lineal
modelo_lin <- lm(mpg ~ hp, data = mtcars)
# Modelo polinómico
modelo_poly <- lm(mpg ~ poly(hp, 2), data = mtcars)
# Comparar R²
summary(modelo_lin)$r.squared
summary(modelo_poly)$r.squared
# Comparar AIC
AIC(modelo_lin)
AIC(modelo_poly)
```

Principales Métricas de Evaluación:

Métrica	Uso	Función en R
R^2 (Coeficiente de determinación)	Evalúa qué porcentaje de la variabilidad es explicada por el modelo.	summary(modelo)$r.squared
RMSE (Error Cuadrático Medio)	Indica la diferencia promedio entre valores predichos y reales.	sqrt(mean((real - predicho)^2))
AIC/BIC (Criterios de Información)	Comparación de modelos penalizando la complejidad.	AIC(modelo), BIC(modelo)
Curva ROC/AUC	Evalúa modelos de clasificación.	roc(response, predictor)

Interpretación de resultados:

- Si un modelo tiene menor **RMSE** y mayor **R^2,** es mejor para predicción.
- Si **AIC/BIC** es menor en un modelo, indica un mejor balance entre precisión y simplicidad.
- Para clasificación, la curva **ROC** y el **AUC** ayudan a evaluar rendimiento.

Validación de Modelos Métricas y Diagnóstico

UNICISO
WWW.PORTALUNICISO.COM

La validación garantiza que el modelo funcione correctamente en datos nuevos y no solo en los datos de entrenamiento. También permite detectar problemas como el sobreajuste o el subajuste.

```
# Cargar librería
library(caret)
# Dividir los datos
set.seed(123)
trainIndex <- createDataPartition(mtcars$mpg, p = 0.8, list = FALSE)
trainData <- mtcars[trainIndex, ]
testData  <- mtcars[-trainIndex, ]
# Ajustar modelo lineal y predecir
modelo <- lm(mpg ~ hp + wt, data = trainData)
predicciones <- predict(modelo, newdata = testData)
# Calcular RMSE
sqrt(mean((testData$mpg - predicciones)^2))
```

Métodos de validación

Método	Descripción	Función en R
Train-Test Split	Dividir los datos en entrenamiento (80%) y prueba (20%).	createDataPartition() (caret)
Cross-validation	Entrena el modelo varias veces con diferentes particiones.	trainControl(method = "cv")
Bootstrapping	Extrae muestras repetidas con reemplazo para mayor robustez.	boot() (boot package)

Diagnóstico de Modelos:

Diagnóstico	Qué mide	Función en R
Errores de predicción	Diferencia entre valores reales y predichos.	residuals(modelo)
Normalidad de errores	Supuesto de regresión lineal.	shapiro.test(resid(modelo))
Autocorrelación	Si los errores están correlacionados.	durbinWatsonTest(modelo)
Multicolinealidad	Si hay alta correlación entre variables.	vif(modelo) (car package)

Errores comunes y Mejores Prácticas

El éxito de un modelo depende tanto de su correcta implementación como de evitar errores frecuentes.

Error	Descripción	Solución
Sobreajuste (Overfitting)	El modelo se ajusta demasiado a los datos de entrenamiento y no generaliza bien.	Validar con Train-Test Split o Cross-Validation. Reducir complejidad del modelo si es necesario.
Subajuste (Underfitting)	El modelo es demasiado simple y no captura patrones en los datos.	Agregar más variables explicativas o probar modelos más complejos.
Multicolinealidad	Variables altamente correlacionadas afectan la interpretación del modelo.	Usar vif() (Variance Inflation Factor) y eliminar variables redundantes.
Suposiciones de regresión no verificadas	En modelos lineales, los residuos deben ser normales y homocedásticos.	Usar shapiro.test() para normalidad y plot(modelo) para diagnóstico visual.
Elección inadecuada del modelo	Aplicar un modelo lineal cuando la relación es no lineal.	Revisar patrones en los datos y probar modelos polinómicos o logísticos.

UNICISO
WWW.PORTALUNICISO.COM

Referencias

- Casal (ruben.fcasal@udc.es), R. F., Roca-Pardiñas (roca@uvigo.es), J., Bouzas (julian.costa@udc.es), J. C., & Fuente (manuel.oviedo@udc.es), M. O. de la. (2023). 11.3 Bucles y vectorización | Introducción al Análisis de Datos con R. https://rubenfcasal.github.io/intror/bucles-y-vectorizaci%C3%B3n.html
- Corcoran, D. (2019). Capítulo 5 Modelos en R | Manipulación de datos e investigación reproducible en R. https://bookdown.org/content/3515/modelos.html
- datdata. (2025). R para el análisis de datos. https://www.datdata.com/blog/r-para-el-analisis-de-datos
- Ihaka, R., & Gentleman, R. (1996). R: A Language for Data Analysis and Graphics. Journal of Computational and Graphical Statistics, 5(3), 299–314. https://doi.org/10.2307/1390807
- Keilor Rojas-Jimenez. (2022). Ciencia de Datos para Ciencias Naturales. https://bookdown.org/keilor_rojas/CienciaDatos/
- r4ds. (2023, June 6). R Para Ciencia de Datos. https://es.r4ds.hadley.nz/
- Río, F. M. del. (2024). Tema 11 Funciones | Programación con R. https://www4.ujaen.es/~fmartin/R/funciones.html
- VIU Universidad Online. (2022, October 4). Lenguaje de programación R: Qué es, características e importancia en el Big Data. VIU Universidad Online. https://www.universidadviu.com/es/actualidad/nuestros-expertos/lenguaje-de-programacion-r-que-es-caracteristicas-e-importancia-en-el-big-data
- Williams, G. (2011). Data Mining with Rattle and R: The Art of Excavating Data for Knowledge Discovery. Springer. https://doi.org/10.1007/978-1-4419-9890-3

CITA DE LA GUÍA

Ramirez, J.P. (2025). Funciones, bucles y modelos en estadística R. UNICISO. Disponible en: www.portaluniciso.com

SÍGUENOS:

UNICISO
WWW.PORTALUNICISO.COM