Lenguaje R: una herramienta versátil para el análisis de datos

Isaías Pérez Ramírez, Hugo Francisco Rincón López,

Rudesindo Acuña Sánchez

Universidad Nacional de Colombia - sede La Paz
isperezr@unal.edu.co, hufrinconlo@unal.edu.co, ruacunas@unal.edu.co

June 13, 2022

Abstract

En 1970 los laboratorios Bell crearon un lenguaje de programación llamado S, que tenía como propósito brindar un enfoque alternativo y más interactivo en el campo de análisis de datos. Más adelante, dicho lenguaje derivó a lo que hoy se conoce como lenguaje de programación R. Este entorno de programación es una poderosa herramienta para análisis estadísticos y creación de gráficos. R tiene la ventaja de ser gratis, tener numerosos paquetes de mucha ayuda para multiples cálculos y tener en su página web muchas guías y ayudas para la correcta implementación de sus comandos. En el siguiente documento se abordarán aspectos generales para la instalación y el uso del lenguaje R.

In 1970 Bell Labs created a programming language called S, which was intended to provide an alternative and more interactive approach to data analysis. This language later evolved into what is known today as the R programming language. This programming environment is a powerful tool for statistical analysis and graphing. R has the advantage of being free, having numerous packages of much help for multiple calculations and having in its web page many guides and aids for the correct implementation of its commands. The following document will address general aspects for the installation and use of the R language.

I. Instalación

Antes de la instalación, es importante tener en cuenta que se debe descargar tanto R como Rstudio. Rstudio funciona como un IDE que brinda un entorno más agradable del lenguaje R. Si bien no es obligatorio descargar ambos programas, se recomienda hacerlo para facilitar la visualización de los datos.

i. Instalación de R

El primer paso para la instalación del lenguaje R consiste en ingresar a la página web de R (http://www.r-project.org/). A partir de ahí, se tienen en cuenta los siguientes pasos:

1. click download CRAN in the left bar.
2. choose a download site.
3. choose Windows as target operation system.
4. click base
5. choose Download R 3.0.3 for Windows and choose default answers for all questions

ii. Instalación de Rstudio

Al igual que la instalación del lenguaje R, para la descarga del Rstudio se realiza un protocolo semejante. El primer paso consiste en ingresar a la pagina web de Rstudio (http://www.rstudio.org/) y por último se efectúan los siguientes pasos.

1. click Download RStudio
2. click Download RStudio Desktop

3. click Recommended For Your System.
4. download the .exe file and run it (choose default answers for all questions)

De forma alternativa, Rstudio también cuenta con un servidor en línea que se puede usar en lugar de descargar el programa. Dicho programa recibe el nombre de Rstudio Cloud (https://rstudio.cloud/), y se puede usar de forma gratuita, aunque existe una versión paga con mejores características y más capacidad de almacenado. Para su ingreso, es necesario vincular una cuenta de usuario.

II. Estructuras de datos básicas de R

Las estructura de datos se conocen como las formas de organización y manipulación de datos para posterior análisis y presentación. El lenguaje R maneja tres principales estructuras de datos: vectores, matrices y data frames.

i. Vectores

Para definir un vector en lenguaje R es necesario hacer uso de la función c(). Si se desea crear un vector con los números del 1 al 10, el vector debe ser escrito de la siguiente forma:

```
x = c(1,2,3,4,5,6,7,8,9,19)
print(x)
```

La primer línea de código indica que al vector lo hemos almacenado en una variable llamada x. Además, con la función c() creamos un vector que va de 1 a 10. Con la siguiente línea de código indicamos que imprima el vector en consola.
Es importante resaltar que dentro de un vector pueden estar números, valores logicos (booleanos), cadenas de caracteres y valores complejos. Incluso pueden estar diferentes tipos de datos dentro de un mismo vector.

En el caso de que se quiera generar un vector con valores aleatorios, hacemos uso de la función rnorm(). Dicha función hace parte del paquete que tiene R para estudiar distribuciones normales. Con rnorm() obtenemos valores aleatorios dentro una media y varianza establecidas. La función se usa de la siguiente manera:

```
y = rnorm(100)
print(y)
```

ii. matrices

La matrices en R son objetos donde los datos son almacenados en un conjunto rectangular bidimensional. La matrices son muy útiles para la organización y posterior análisis de datos.

Para la creación de matrices en lenguaje R se usa la función:

```
matrix(data, nrow, ncol, byrow=F)
```

Donde el parametro "data" hace referencia a los valores estarán dentro de la matriz, "nrow" hace referencia al número de filas de la matriz, "ncol" es el número de columnas de la matriz y, por último, "byrow" sirve para indicar si los datos se colocan por filas o columnas. Como ejemplo de la función de matrix(data, nrow, ncol, byrow=F) tenemos el siguiente:

```
x <- matrix(c(2, 7, 1, 3, 6, 1),
ncol = 2, byrow = TRUE)
```

Además de la función Matriz() existen otros comandos de mucha ayuda para el manejo de las matrices. Dichas funciones son:

- dim(): Devuelve las dimesiones de la matriz.
- dimnames(): Devuelve el nombre de las dimensiones de la matriz.
- colnmaes(): Devuelven el nombre de las columnas de la matriz.
- rownames(): Devuelve el nombre de las filas de la matriz.
- length(): Devuelve la cantidad de elementos de la matriz.
- apply(): Aplica una función sobre las filas o columnas de una matriz.
- cbind(): Añade una columna a la matriz.
- rbind(): Añade una fila a la matriz.

iii. Data frame

Un dataframe es una estructura de datos similar a una matriz en la que las columnas son vectores y tienen un nombre, parecido a una tabla. Es muy utilizada para trabajar con cantidades de datos grandes. A diferencia de las matrices, un dataframe puede tener distintos tipos de datos en cada columna. Las operaciones ariméticas se vectorizan y aplican a todo el dataframe.

Para crear un dataframe se puede hacer de la siguiente manera:

```
df <- data.frame(
"entero" = 1:4,
"factor" = c("a", "b", "c", "d"),
"numero" = c(1.2, 3.4, 4.5, 5.6),
"cadena" = as.character(c("a","b"))
)
```

En este caso el comando `data.frame` recibe el nombre de la columna entre comillas y un vector que ubica en dicha columna, estos vectores deben ser de igual tamaño.

Cuando se quiere acceder a una columna del dataframe se hace de la siguiente manera.

```
df$col_name
```

Usando el signo dólar y colocando el nombre de la columna. Este devuelve todos los datos de esta columna.

III. Visualizaciones

Al ser R un lenguaje de programación estadístico dispone de visualizaciones como histogramas, lineplots, barplots, boxplots, scatter plots, heatmaps, entre otras.

A continuación se enseñarán algunas gráficas muy usadas para exploración de datos. En las gráficas se usan hasta tres vectores data1, data2 y data3, conformados por 100 valores aleatorios cada uno.

Hay ciertos parámetros dentro de los gráficos que son comunes en los códigos, algunos de esos son:

main Se utiliza para asignar un nombre al gráfico, este saldrá encima y en grande.

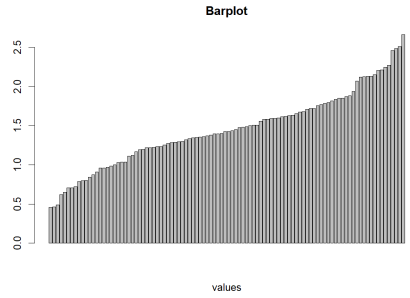
xlab Se utiliza para asignar un nombre al eje x.

ylab Se utiliza para asignar un nombre al eje y.

col permite cambiar el color de los gráficos.

horiz Este parámetro se utiliza en algunos casos para cambiar la orientación de algunos gráficos como los de barras o boxplots.

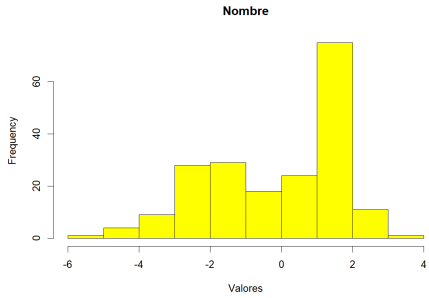
i. Gráfico de barras



Estos son usados para analizar cambios en el tiempo de los datos y observar tendencias. Para crearlo se usando la función barplot().

```
barplot(data1, main='Barplot',
    xlab='values', col = 'gray',
    horiz = F)
```

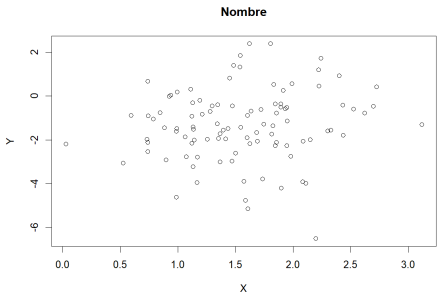
ii. Histograma



Un histograma permite ver la frecuencia de los valores en ciertos rangos y se crea con la función hist()

```
hist(c(data1,data2),
 main ="Nombre", xlab ="Valores",
 col ="yellow",
 )
```

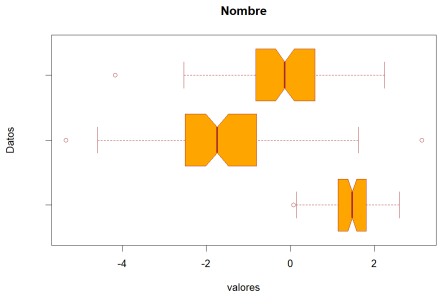
iii. Gráfico de dispersión



Esta visualización permite observar la dispersión de los datos, es útil para encontrar agrupaciones y patrones. Se crea usando el comando `plot`

```
plot(data1, data2, main ="Nombre",
 xlab = "X", ylab = "Y"
 )
```

iv. Boxplot

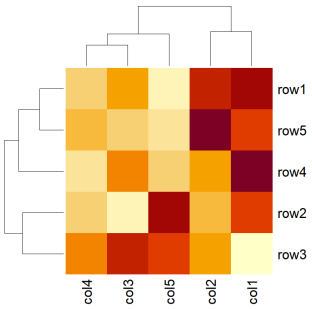


En el boxplot se puede observar un resumen de los datos. El máximo, el mínimo, la mediana, cuartiles y valores atípicos de los datos. Para generar uno se utiliza el comando `boxplot`

```
boxplot(data1, data2, data3,
    main = "Nombre",xlab="valores",
    ylab = "Datos", col = "orange",
    border = "brown",
    horizontal = T, notch = T)
```

En este caso se puede observar que la gráfica está en modo horizontal, el comando `notch` activa la forma diferente del gráfico y `border` cambia el color de los marcos para hacerlos más notables.

v. Mapa de calor



Los mapas de calor sirven para ver diferentes valores en una tabla usando colores, son de gran utilidad para comparar matrices de correlación. Para generarla se usa el código `heatmap`

```
data <- matrix(rnorm(25, 0, 5),
    nrow = 5, ncol = 5)
colnames(data) <-paste0("col", 1:5)
rownames(data) <-paste0("row", 1:5)
heatmap(data)
```

En este caso se utiliza una matriz 5x5 de valores aleatorios. Se le asignan nombres de col1 hasta col5 a las columnas y row1 hasta row5 a las filas.

References

[bhartirishika, 2022] *Data Visualization in R*. Geeksforgeeks. https://www.geeksforgeeks.org/data-visualization-in-r/

[RDocumentation] https://www.rdocumentation.org/

[Figueredo and Wolf, 2009] Figueredo, A. J. and Wolf, P. S. A. (2009). Assortative pairing and life history strategy - a cross-cultural study. *Human Nature*, 20:317–330.