

Guia primeros pasos en R

Jose Manuel Pinzon / Laboratorio Fintrade

15/9/2021

Índice

Tipos de datos en R.	2
R como calculadora.	2
Estructuras de datos en R.	3
Variables.	3
Vectores.	3
Creación de vectores.	4
Operaciones con vectores	5
Indexación.	5
Matrices.	6
Creación de matrices	6
Indexación	7
DataFrame.	8
Crear dataframes.	8
Características del DataFrame	8
Modificar dataframes.	8
Indexacion	9
Importar bases de datos.	10
EJERCICIOS.	12
Operaciones:	12
Vectores	12
Matrices	12
DataFrame.	13

NOTA ACLARATORIA: Esta guía aún se encuentra en construcción, no es la versión final, razón por la cual está sujeta a modificaciones.

Esta guía es exclusivamente para uso pedagógico

Tipos de datos en R.

En los lenguajes de programación se encuentran varios tipos de datos. Los mas usuales son estos:

1. Entero.
2. Decimal / Flotante / double
3. Caracter. (Texto / Palabras)
4. Factor (Transformación del caracter)
5. Lógico. (Verdadero y Falso)

Algunos comandos para conocer los diferentes atributos de los objetos son: `class()`, `dim()`, `length()`, entre otros. Si requiere cambiar la clase de un objeto basta con ejecutar `as.____()`. Por ejemplo, si usted quiere cambiar un dato de tipo caracter a tipo factor lo puede hacer de la siguiente manera. `Tipo_factor = as.factor(caracter)`

```
class(1L)
```

```
## [1] "integer"
```

```
class(2.5)
```

```
## [1] "numeric"
```

```
class("Hola mundo")
```

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

```
class(as.factor("Hola mundo"))
```

```
## [1] "factor"
```

```
class(TRUE)
```

```
## [1] "logical"
```

R como calculadora.

Un uso de R es como calculadora, a continuación verá un listado con algunos comandos de operaciones básicas que se pueden hacer con el software y posteriormente encontrará ejemplos de operaciones en R.

1. Suma (+)
2. Resta (-)
3. Multiplicación (*)
4. División (/)
5. División entera (%/ %)
6. potenciación (^)
7. Raiz Cuadrada (sqrt)
8. Logaritmos (log)
9. Multiplicación de matrices (%* %)
10. Transponer (t())

Para las operaciones entre matrices debe tener en cuenta las reglas del álgebra matricial, por ejemplo: no todas las matrices se pueden sumar entre ellas.

```
#SUMA
```

```
45 + 6 + 89
```

```

#RESTA
8 - 9 - 2
#MULTIPLICACION
10 * 5
10 * (-5)
#DIVISION
8 / 3
8 %/% 3 #Division entera
#POTENCIACION
5^2
5**3
#RAIZ CUADRADA
sqrt(25)
#LOGARITMOS
log(10)

```

Estructuras de datos en R.

Variables.

R permite crear y guardar estructuras de datos, la primera que vamos a desarrollar será la variable. Esta puede ser creada con el operador *Asignar*, el cual se denota como una flecha creada con el signo menos (-) y el simbolo mayor que (>) o menor que (<), dependiendo de la direccion en que asigna el objeto.

A continuación se tienen unos ejemplos donde se crearán objetos o variables y se les asignará un valor:

```

## Se crean los objetos "estatura" con un valor de 1.85 y "peso" con un valor de 75
estatura <- 1.85
peso <- 75
## Se crean los objetos "n" y "q"
5 -> n
sqrt(64) -> q

```

Usted también puede operar variables, el resultado le aparecerá en la consola:

```

## Operaciones
estatura / peso
estatura + peso
peso * estatura

```

Si usted desea guardar esos resultados de alguna operación lo puede hacer creando otra variable. Para saber el valor de su objeto usted puede mirar el entorno o puede llamar a la variable volviendo a escribir el nombre que le asignó:

```

## Guardar variable con un resultado
IMC <- peso/estatura^2
## Para ver el resultado en la consola
IMC

## [1] 21.91381

```

Vectores.

Un vector en R contiene una secuencia finita de valores de un solo tipo, como una secuencia de números o una secuencia de caracteres. A continuación verá dos vectores, el primero con datos de tipo numérico y el segundo con datos tipo caracter.

Creación de vectores.

$Vector1 = (1, 2, 3, 4, 5, 6)$

$Vector2 = (arbol, casa, persona)$

Se puede utilizar la función `c` para crear vectores, ya que esta permite agrupar un conjunto de datos. Para identificar que tipo de datos contiene cada vector se utiliza la función `class`, la cual determina si el vector contiene datos numéricos, caracteres o lógicos.

```
X <- c(1,2,3,4,5,6)
Y <- c("arbol","casa","persona")
class(X)
```

```
## [1] "numeric"
```

Recuerde que los vectores permiten solo un tipo de dato, cuando se ponen varios tipos de datos R buscará la manera de transformarlos para que todo quede unificado:

```
v1 <- c(1.7, "a") ## character
v2 <- c(TRUE, 2) ## numeric
v3 <- c("a", TRUE) ## character
v4 <- 9:29 ## integer
v5 <- c(1,2,3.5) ## numeric
```

Otra manera de crear vectores es por medio de cadenas o secuencias numéricas:

```
#Creación de cadenas
1:10 #Secuencia numerica del 1 al 10
10:1 #Secuencia numerica del 10 al 1
-43:-30 #Secuencia numerica de -43 a -30
```

También se pueden crear vectores con el comando `seq`. El primer argumento será el número inicial, el segundo argumento será el número final, para el siguiente caso se establecerá una secuencia numérica del 1 al 4.

```
seq(1,4)
```

```
## [1] 1 2 3 4
```

El comando `seq` tiene argumentos adicionales que pueden ser útiles para definir el conjunto de datos de su vector. Un argumento es `len`, el cual permite establecer el número de datos que componen la secuencia, si usted coloca `len = 20`, su secuencia tendrá 20 números. Otro argumento es `by`, con el cual usted indica los saltos que va a tener la secuencia, si usted quiere que su cadena de números vaya de 5 en 5 pondrá `by = 0.5`.

```
seq(1,4, len = 20)
```

```
## [1] 1.000000 1.157895 1.315789 1.473684 1.631579 1.789474 1.947368 2.105263
## [9] 2.263158 2.421053 2.578947 2.736842 2.894737 3.052632 3.210526 3.368421
## [17] 3.526316 3.684211 3.842105 4.000000
```

```
seq(1,4, by = 0.5)
```

```
## [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0
```

Para que no se muestren tantos decimales usted puede redondear los resultados con el comando `round`. El primer argumento serán los valores que se quieren redondear y el segundo argumento serán los decimales que usted quiere mostrar. A continuación se creará el vector `a` y posteriormente se utilizará el comando `round` para redondear los decimales.

```
a <- seq(1,4, len = 20)
round(a,2)
```

```
## [1] 1.00 1.16 1.32 1.47 1.63 1.79 1.95 2.11 2.26 2.42 2.58 2.74 2.89 3.05 3.21
## [16] 3.37 3.53 3.68 3.84 4.00
```

Usted también puede sobrescribir un vector u objeto. Si quiere que el vector “a” quede guardado con las modificaciones del round puede hacer lo siguiente:

```
a <- round(a,2)
a
```

```
## [1] 1.00 1.16 1.32 1.47 1.63 1.79 1.95 2.11 2.26 2.42 2.58 2.74 2.89 3.05 3.21
## [16] 3.37 3.53 3.68 3.84 4.00
```

Operaciones con vectores

También se pueden operar vectores:

```
mi_vector <- c(2,3,6,7,8,10,11)
P <- seq(1,4, len = 7)
mi_vector + P ## Suma de dos vectores.
```

```
## [1] 3.0 4.5 8.0 9.5 11.0 13.5 15.0
```

```
mi_vector + 2
```

```
## [1] 4 5 8 9 10 12 13
```

```
mi_vector * 3
```

```
## [1] 6 9 18 21 24 30 33
```

```
mi_vector > 7
```

```
## [1] FALSE FALSE FALSE FALSE TRUE TRUE TRUE
```

Indexación.

La indexación se usa para ubicar valores en las estructuras de datos. Para los vectores es muy sencillo, lo primero es llamar el nombre del vector, luego se ponen corchetes [], y dentro del corchete se va a colocar la posición del valor que quiere buscar.

Supongamos que quiere saber qué número va en la posición 6 del objeto llamado “mi_vector”, para eso haremos lo siguiente:

```
mi_vector[6]
```

```
## [1] 10
```

El output o salida indica que el elemento 6 del vector es “10”.

También puede hacer operaciones con un número ubicado con la indexación.

```
mi_vector[2] + 3
```

```
## [1] 6
```

Ejercicio: Crear un vector que vaya desde -5 hasta 6 y que tenga 7 números. Luego de crear el objeto sumarlo con “mi_vector1” e imprima el resultado con maximo 3 decimales.

La respuesta debería salir así:

$Vector = (-3.000, -0.167, 4.667, 7.500, 10.333, 14.167, 17.000)$

Matrices.

Una matriz es realmente un vector con un atributo adicional (`dim`) el cual a su vez es un vector numerico de longitud 2, que define el numero de filas y columnas de la matriz. Una matriz se puede crear con la funcion **matrix**.

Creación de matrices

En la figura 1 se muestra un ejemplo de una matriz numérica.

$$\mathbf{X} = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}$$

Figura 1: Matriz 1

Para hacerla en R se puede utilizar el siguiente código. El primer argumento es la secuencia de numeros o valores que se quieren tener en la matriz, seguido por el comando **nrow** con el cual se especifican el número de filas, y por último está el argumento **ncol**, este permite establecer el numero de columnas de la matriz.

```
matrix(1:12, nrow = 3, ncol = 4)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

Figura 2: Matriz 2

Para organizar los valores por filas y que queden como en la figura 2 se va a utilizar el argumento **byrow**. Mire el siguiente código.

```
matrix(1:12, nrow = 3, ncol = 4, byrow = TRUE)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
```

Otra manera de crear una matriz es dando los valores apropiados al atributo **dim** (que inicialmente tiene valor NULL):

```
g <- 1:15 #vector
dim(g)
```

```
## NULL
```

```
dim(g) <- c(5, 3) #se crea la matriz especificando filas y columnas
g
```

```
##      [,1] [,2] [,3]
## [1,]    1    6   11
## [2,]    2    7   12
## [3,]    3    8   13
## [4,]    4    9   14
## [5,]    5   10   15
```

También se pueden crear matrices uniendo varios vectores. A continuación se crearán 4 vectores.

```
vector_1 <- 1:4
vector_2 <- 5:8
vector_3 <- 9:12
vector_4 <- 13:16
```

Ahora se va a utilizar el comando *cbind*, con este comando se van a unir los vectores usando cada uno de ellos como columnas:

```
matriz_v <- cbind(vector_1,vector_2,vector_3,vector_4)
matriz_v
```

```
##      vector_1 vector_2 vector_3 vector_4
## [1,]        1        5        9        13
## [2,]        2        6       10        14
## [3,]        3        7       11        15
## [4,]        4        8       12        16
```

O también se puede usar el comando *rbind* para usar los vectores como filas:

```
matriz_v2 <- rbind(vector_1,vector_2,vector_3,vector_4)
matriz_v2
```

```
##      [,1] [,2] [,3] [,4]
## vector_1    1    2    3    4
## vector_2    5    6    7    8
## vector_3    9   10   11   12
## vector_4   13   14   15   16
```

Indexación

Para ubicar valores dentro de una matriz, lo primero que se hace es llamar la matriz, seguido de [], en estos corchetes se colocará el número de fila y de columna respectivamente.

```
index <- matrix(1:12, nrow = 3, ncol = 4)
index
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
index[1,3] ## Para ubicar el valor 7, fila numero 1 y columna numero 3
```

```
## [1] 7
```

```
index[3,2] ## Para ubicar el valor 6, fila numero 3 y columna numero 2
```

```
## [1] 6
```

DataFrame.

Los dataframe son tipos especiales de objetos en R diseñados para conjuntos de datos que se parecen a las matrices, pero a diferencia de las matrices, las columnas de un marco de datos pueden ser diferentes tipos, como numéricos o de caracteres. Se ha visto que un dataframe se crea de manera implícita con la función `read.table`; también es posible hacerlo con la función `data.frame`, los vectores incluidos como argumentos deben ser de la misma longitud.

Crear dataframes.

Un DataFrame puede ser creado uniendo vectores. A continuación se crearán dos vectores y luego se van a unir con la función **data.frame**. Y se utilizará la función `class` para verificar que el objeto creado es un DataFrame.

```
entero <- c(1:7)
decimal <- c(1.2,2.5,3.2,4.5,5.6,5,3.4)
caracter <- c("Juan","Jose","Laura","Valentina","Valentina","Valentina", "Juana")
mi_df <- data.frame(entero,decimal,caracter)
class(mi_df)
```

```
## [1] "data.frame"
```

Características del DataFrame

Si usted quiere tener una idea de que datos tiene su dataframe puede usar los códigos **head** y **tail**. El comando `head` le permitirá ver los primeros datos de su base de datos, el primer argumento es el nombre de su dataframe y el segundo elemento es la cantidad de observaciones o filas que desea ver. El comando `tail` funciona igual que `head`, pero en vez de mostrar los primeros datos, le imprime las últimas observaciones.

Para ver los nombres de las columnas de su DataFrame utilice el comando **names**, y entre parentesis colocará el nombre de su DataFrame.

Explore los siguientes comandos:

```
names(mi_df)
head(mi_df,3)
tail(mi_df,2)
unique(mi_df$caracter) #Valores que puede tener una columna
length(mi_df) # Numero de columnas
nrow(mi_df) # Numero de filas
dim(mi_df) #Numero de filas y de columnas
str(mi_df) # Atributos de cada variable / columna
```

Si usted desea ver información de alguna variable o columna en específico, puede vincular la base de datos y la variable con el signo `$`. En primer lugar usted colocará el dataframe que quiere trabajar, seguido del signo `$` y luego el nombre de la variable que quiere seleccionar:

```
min(mi_df$entero) #Numero máximo de la variable seleccionada
```

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

```
max(mi_df$decimal) #Numero mínimo de la variable seleccionada
```

```
## [1] 5.6
```

Modificar dataframes.

Se pueden agregar nuevas variables o vectores a un dataframe ya creado. A continuación se mostrará un ejemplo:

Lo primero que se va a hacer es crear un vector con ciudades. Recuerde que para unir vectores tienen que tener la misma longitud, es decir, el mismo número de filas.

Para agregar el vector al dataframe: Lo primero es llamar al DF, luego se pondrá el signo \$ y posteriormente se escribirá el nombre de la columna que usted quiere crear, por último le asignará el vector creado.

```
## Crear vector
Ciudades <- c("Bogota","Medellin","Cartagena", "Cali", "Leticia", "Pereira", "Barranquilla")
## Agregar al DF
mi_df$CIUDADES <- Ciudades
mi_df
```

```
##      entero decimal  caracter    CIUDADES
## 1         1      1.2      Juan     Bogota
## 2         2      2.5      Jose    Medellin
## 3         3      3.2     Laura   Cartagena
## 4         4      4.5 Valentina    Cali
## 5         5      5.6 Valentina   Leticia
## 6         6      5.0 Valentina   Pereira
## 7         7      3.4     Juana Barranquilla
```

Cambiar nombre de columnas: Para cambiar el nombre de las columnas se emplea el código **names** de la siguiente manera.

```
## Cambiar todos los nombres de las columnas.
names(mi_df) = c("Columna1","Columna2","Columna3","Columna4")
mi_df
```

```
##      Columna1 Columna2  Columna3    Columna4
## 1          1      1.2      Juan     Bogota
## 2          2      2.5      Jose    Medellin
## 3          3      3.2     Laura   Cartagena
## 4          4      4.5 Valentina    Cali
## 5          5      5.6 Valentina   Leticia
## 6          6      5.0 Valentina   Pereira
## 7          7      3.4     Juana Barranquilla
```

```
## Cambiar el nombre de alguna columna en específico.
names(mi_df)[1] = "Entero"
mi_df
```

```
##      Entero Columna2  Columna3    Columna4
## 1         1      1.2      Juan     Bogota
## 2         2      2.5      Jose    Medellin
## 3         3      3.2     Laura   Cartagena
## 4         4      4.5 Valentina    Cali
## 5         5      5.6 Valentina   Leticia
## 6         6      5.0 Valentina   Pereira
## 7         7      3.4     Juana Barranquilla
```

Indexacion

La indexación para los dataframes funciona exactamente igual que para las matrices. Se ponen corchetes [] después del nombre de su data frame y dentro de ellos coloca el número de fila y columna del valor que quiere seleccionar.

Si usted quiere seleccionar una fila completa deja el primer argumento vacío, si quiere mirar todas las columnas deja el segundo argumento vacío.

Explore los siguientes comandos:

```
mi_df[1,3]
mi_df[,4] #Todas las filas
mi_df[2,] #Todas las columnas
mi_df[, -4] #Quitar columnas
mi_df[-3,] #Quitar filas
```

Importar bases de datos.

Por defecto, R ya trae instaladas algunas bases de datos o DataFrames que pueden ser utilizadas por los usuarios. Para acceder a estas librerías simplemente basta con escribir el nombre del dataset. Si usted desea conocer cuales son las bases de datos que puede manejar desde R puede llamar al paquete de la siguiente manera **datasets::**, de inmediato le aparecerá una lista con todos los archivos que puede mirar. Si usted desea guardar uno de estos objetos en el entorno, puede hacerlo con el comando *data()*, entre parentesis colocará el nombre del dataset. Otra manera de hacerlo es asignandole un nombre seguido por el simbolo de asignación.

```
#Para ver el menu de datasets.
```

```
datasets::
```

```
#Para guardarlo en el entorno
```

```
data("airquality")
head(airquality,3)
```

```
##      Ozone Solar.R Wind Temp Month Day
## 1      41      190  7.4   67     5   1
## 2      36      118  8.0   72     5   2
## 3      12      149 12.6   74     5   3
```

```
base <- datasets::iris
head(base,3)
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1           5.1           3.5           1.4           0.2  setosa
## 2           4.9           3.0           1.4           0.2  setosa
## 3           4.7           3.2           1.3           0.2  setosa
```

Otra ventaja de R es la integración de Api's. Las cuales se vinculan con otras páginas o servidores para descargar información.

Para este caso vamos a descargar de la página *yahoo finance* información de la acción de apple desde el primero de enero del 2015. Para esto es necesario llamar la librería **quantmod**, si no la tiene, por favor descárgela y la instala.

Para obtener los datos se usa la función **getsymbols**, el primer argumento será el ticket o nombre que represente la acción, para este caso es AAPL, con argumento *src* indicará la fuente o página de donde quiere descargar la información, si lo que quiere descargar tiene periodicidad colóquelo con el argumento *periodicity*, de lo contrario omítalo, y por último tiene los argumentos *from* y *end*, con los cuales podrá definir fecha de inicio y fecha final respectivamente.

```
library(quantmod) ## Se llama a la librería
```

```
getSymbols("AAPL",src = "yahoo", periodicity = "daily", from = "2015/01/01")
```

```
## [1] "AAPL"
```

```
head(AAPL,3)
```

```
##           AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
## 2015-01-02  27.8475  27.8600  26.8375   27.3325  212818400    24.78211
## 2015-01-05  27.0725  27.1625  26.3525   26.5625  257142000    24.08396
## 2015-01-06  26.6350  26.8575  26.1575   26.5650  263188400    24.08623
```

R también permite trabajar con varios tipos de archivo, se pueden importar archivos provenientes de programas como SPSS, SAS, STATA, Excel, entre otros. Para importar estos archivos debe dirigirse al entorno, allí encontrará una opción que dice *Import Dataset*, al seleccionarla le aparecerá una lista, con algunos de los formatos de archivo que puede importar. Mire la Figura 3.

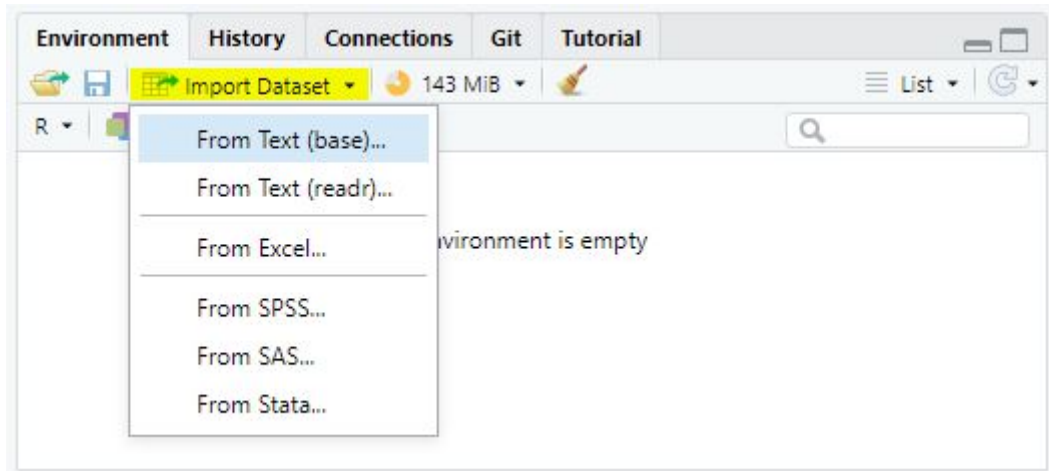


Figura 3: Menu Import Dataset.

Las dos primeras opciones, por lo general se usan para importar archivos **txt** y **csv**, la tercera opción es para importar archivos de excel o archivos **.xlsx**.

Para importar los archivos seleccione la opción correspondiente, a continuación le aparecerá una nueva ventana, mire la figura 4. Lo primero que debe hacer es darle al botón **Browse...**, luego deberá buscar y seleccionar lo archivo. Cuando lo haya seleccionado R le mostrará una vista previa de los datos del archivo. Cuando confirme que toda la información está correcta copie el código que aparece en la parte inferior derecha (Puede hacerlo dando click en el logo resaltado), le dará importar y por último, pegará el código copiado en su script de R

```
library(readxl)
Departamentos <- read_excel("C:/Users/JosePinzon/Downloads/Departamentos.xlsx")
```

```
## New names:
## * `` -> ...1
```

```
head(Departamentos)
```

```
## # A tibble: 6 x 10
##   ...1 DEPARTAMENTO TDEP  TBM  TGF  EVN  PIBPC PNBI  DEJV  GINI
##   <dbl> <chr>      <dbl> <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl>
## 1     1 Antioquia    495.  5.89  63.8  75.1 11815358. 23.0  26.5 0.561
## 2     2 Atlántico    525.  5.29  68.6  75.5  8966238. 24.7  40.0 0.496
## 3     3 Bogotá      450.  4.47  56.8  78.0 18791838.  9.20 19.2 0.527
## 4     4 Bolívar     605.  5.63  83.3  74.2 10887813. 46.6  61.7 0.514
## 5     5 Boyacá      623.  6.93  74.7  75.3 11774525. 30.8  40.6 0.539
## 6     6 Caldas     521.  7.19  65.6  74.5  8700434. 17.8  22.1 0.534
```

Import Excel Data

File/URL:
C:/Users/JosePinzon/Downloads/Departamentos.xlsx Browse...

Data Preview:

...1 (double)	DEPARTAMENTO (character)	TDEP (double)	TBM (double)	TGF (double)	EVN (double)	PIBPC (double)	PNBI (double)	DEFV (double)	GINI (double)
1	Antioquia	495.2311	5.892037	63.8	75.07	11815358	22.959642	26.48	0.560599
2	Atlántico	525.3218	5.292990	68.6	75.49	8966238	24.737478	39.97	0.495800
3	Bogotá	450.2722	4.471373	56.8	78.01	18791838	9.203009	19.15	0.526600
4	Bolívar	605.1489	5.625398	83.3	74.15	10887813	46.604706	61.74	0.513700
5	Boyacá	622.9729	6.928872	74.7	75.32	11774525	30.765119	40.56	0.539200
6	Caldas	520.5367	7.185238	65.6	74.52	8700434	17.763779	22.11	0.534100
7	Cauquetá	678.5941	6.582215	98.1	69.48	5201366	41.723837	55.21	0.450800
8	Cauca	610.2243	6.720995	83.0	71.89	6043423	46.615119	56.68	0.570100

Previewing first 50 entries.

Import Options:

Name: Departamentos Max Rows: ☒ First Row as Names

Sheet: Default Skip: 0 ☒ Open Data Viewer

Range: A1:D10 NA:

Code Preview:

```
library(readxl)
Departamentos <- read_excel("C:/Users/JosePinzon/Downloads/Departamentos.xlsx")
view(Departamentos)
```

? Reading Excel files using readxl Import Cancel

Figura 4: Menu Import Dataset.

EJERCICIOS.

Cree un script o archivo en R y dentro de el realice los siguientes ejercicios.

Operaciones:

1. Construya un código que realice las siguientes operaciones:

- $10 + 5$
- $4 * 8$
- $3/5$
- 4^2
- Raíz cuadrada de 81

Vectores

2. Guarde un vector que contenga los numeros 1,3,6 y numero favorito
3. Guarde un vector que contenga los números del 1 al 100, dando saltos de 5 en 5.
4. Construya un vector que contenga las letras de su nombre.
5. Cree dos vectores de longitud 30 cada uno y súmelos.
6. Construya un código que le sume 1 a elemento del punto 2 y lo sobrescriba.
7. Elimine el vector del punto 4 con el comando rm.

Matrices

8. Cree las siguientes matrices.

$$\mathbf{A} = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

9. Cree otro objeto con la suma de las dos matrices creadas en el punto anterior.
10. Haga un código que ubique el número 5 de las dos matrices.

DataFrame.

11. Importe un dataframe de R y mire sus características (Tipo de variables, numero de columnas, numero de filas, nombre de las columnas)

NOTA ACLARATORIA: Esta guía aún se encuentra en construcción, no es la versión final, razón por la cual está sujeta a modificaciones.

Esta guía es exclusivamente para uso pedagógico