ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA

EDUCACIÓN PROFESIONAL

Introducción a manejo de datos en R

Gestión de Datos

Maximiliano Arancibia

Educación Profesional - Escuela de Ingeniería

Tabla de Contenidos

ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA

¿Que es R?

R es un lenguaje de programación y entorno computacional dedicado a la estadística.

¿Quién usa R?

R es un lenguaje relativamente joven pero que ha experimentado un crecimiento acelerado en su adopción durante los últimos 10 años.

En relación popularidad, este lenguaje programación ocupa actualmente el lugar numero 16 de acuerdo al TIOBE programming community index, que es uno de los índices de más prestigio en el mundo al respecto.

Entre otros lenguajes similares podemos encontrar:

- Python (1)
- C (2)
- Java (3)
- SQL (9)

¿Cómo acceder a R?

- Instalación directa en consola del computador

¿Cómo acceder a R?

- Instalación directa en consola del computador
- Instalación de un IDE como Rstudio

¿Cómo acceder a R?

- Instalación directa en consola del computador
- Instalación de un IDE como Rstudio
- Usar alguna plataformas web que permitan el uso de R, como Kaggle y Rcolab
  $\Rightarrow$ En este curso los talleres y ejemplos de clases se verán en Kaggle.

Tabla de Contenidos

ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA

Consideraciones generales de R

- Consola y entorno interactivo
- Constantes y nombres
- Documentación
- Directorio de trabajo y sesiones

Tipos de datos

Tipo	Ejemplo	Nombre en inglés
Entero	1	integer
Numérico	1.3	numeric
Cadena de texto	“uno”	character
Lógico	TRUE	logical
Perdido	`NA`	NA
Vacio	`NULL`	null

Figura 1: Tipos de datos de uso más común en R

Tipos de datos

Funciones importantes para los tipos de dato:

- class
- as.{tipo de dato}, ej: as.integer, as.character
- is.{tipo de dato}, ej: is.integer, is.character

Operadores

Los operadores son los símbolos que le indican a R que debe realizar una tarea. Combinando datos y operadores es que logramos que R haga su trabajo.

Existen operadores específicos para cada tipo de tarea. Los tipos de operadores principales son los siguientes:

- Aritméticos
- Relacionales
- Lógicos
- De asignación

Operadores aritméticos

Podemos usar operaciones aritméticas en R, como ya hemos visto anteriormente. Estas se pueden usar con datos enteros o numéricos.

Operador	Operación	Ejemplo	Resultado
`+`	Suma	`5 + 3`	8
`-`	Resta	`5 - 3`	2
`*`	Multiplicación	`5 * 3`	18
`/`	División	`5 /3`	1.666667
`^`	Potencia	`5 ^ 3`	125
`%%`	Modulo	`5 %% 3`	2

Figura 2: Operadores aritméticos en R

Operadores relacionales

Si queremos comparar valores entre si podemos usar los operadores relacionales.

Operador	Comparación	Ejemplo	Resultado
`<`	Menor que	`5 < 3`	`FALSE`
`<=`	Menor o igual que	`5 <= 3`	`FALSE`
`>`	Mayor que	`5 > 3`	`TRUE`
`>=`	Mayor o igual que	`5 >= 3`	`TRUE`
`==`	Exactamente igual que	`5 == 3`	`FALSE`
`!=`	No es igual que	`5 != 3`	`TRUE`

Figura 3: Operadores relacionales en R

Operadores lógicos

Los operadores lógicos son usados para operaciones de álgebra Booleana, es decir, para describir relaciones lógicas, expresadas como verdadero o falso.

Operador	Comparación	Ejemplo	Resultado
`x | y`	x Ó y es verdadero	`TRUE | FALSE`	`TRUE`
`x & y`	x Y y son verdaderos	`TRUE & FALSE`	`FALSE`
`!x`	x no es verdadero (negación)	`!TRUE`	`FALSE`

Figura 4: Operadores lógicos en R

Operadores de asignación

Los operadores asignación son usados para asignar datos a variable.

Operador	Operación
`<-`	Asigna un valor a una variable
`=`	Asigna un valor a una variable

Figura 5: Operadores de asignación en R

Orden de las operaciones

Los operadores se ejecutan en un orden especifico, importante que si no recuerdan este orden usar paréntesis para ordenar las operaciones.

Orden	Operadores
1	^
2	* /
3	+ -
4	< > <= >= == !=
5	!
6	&
7	|
8	<-

Figura 6: Orden de las operaciones en R

Tabla de Contenidos

ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA

Tipos de datos en R

Las estructuras de datos contienen datos y lo que hacemos en R es manipularlos.

Las estructuras tienen diferentes características. Entre ellas, las que distinguen a una estructura de otra son su número de dimensiones y si son homogeneas o hereterogeneas.

Dimensiones	Homogeneas	Heterogeneas
1	Vector	Lista
2	Matriz	Data frame
n	Array	

Figura 7: Estructuras de datos

Revisaremos solo los vectores, listas y dataframes.

Vectores

Un vector es la estructura de datos más sencilla en R. Un vector es una colección de uno o más datos del mismo tipo.

- Todos los valores atomicos se consideran vectores.

Vectores

Un vector es la estructura de datos más sencilla en R. Un vector es una colección de uno o más datos del mismo tipo.

- Todos los valores atomicos se consideran vectores.
- podemos crear vectores más largos con la funcion *c*() (combinar).

Vectores

Un vector es la estructura de datos más sencilla en R. Un vector es una colección de uno o más datos del mismo tipo.

- Todos los valores atomicos se consideran vectores.
- podemos crear vectores más largos con la funcion *c*() (combinar).
- Muchas operaciones se pueden aplicar a vectores, las cuales se ejecutan componente a componente (vectorizacion).

Listas

Las listas, al igual que los vectores, son estructuras de datos unidimensionales, sólo tienen largo, pero a diferencia de los vectores cada uno de sus elementos puede ser de diferente tipo o incluso de diferente clase, por lo que son estructuras heterogéneas.

- Para crear una lista usamos la función list(), donde los argumentos son los elementos de la lista
- No es posible vectorizar las operaciones en la lista

Dataframes

Los dataframes son estructuras de datos de dos dimensiones (rectangulares) que pueden contener datos de diferentes tipos, por lo tanto, son heterogéneas. Esta estructura de datos es la más usada para realizar análisis de datos.

- Para crear un data frame usamos la función data.frame(), lo cual nos pedirá los vectores que queremos usar como columnas. Todos deben tener el mismo tamaño

Dataframes

Los dataframes son estructuras de datos de dos dimensiones (rectangulares) que pueden contener datos de diferentes tipos, por lo tanto, son heterogéneas. Esta estructura de datos es la más usada para realizar análisis de datos.

- Para crear un data frame usamos la función data.frame(), lo cual nos pedirá los vectores que queremos usar como columnas. Todos deben tener el mismo tamaño
- Un data frame está compuesto por vectores en sus columnas por lo que cada una tiene un tipo definido.

Dataframes

Uno de nuestros objetivos dentro del curso y tambien la ciencia de datos es llevar nuesto conjunto de datos a un formato ordenado. Con esto nos referimos principalmente a:

- Cada variable debe tener su propia columna.
- Cada observación debe tener su propia fila.
- Cada valor debe tener su propia celda

Dataframes

Este set de reglas se puede visualizar más fácilmente:

Figura 8: Reglas que hacen que un conjunto de datos sea ordenado: las variables están en columnas, las observaciones en filas y los valores en celdas.

Tabla de Contenidos

ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA

Funciones

Las funciones que tiene la base de R instalado dan para hacer gran cantidad de operaciones y análisis, incluso cosas mas complejas como mostrar información, reportar e incluso crear paginas web.

Sin embargo, les pasará que no encontrarán funciones especificas para hacer alguna tarea, o les gustaría agrupar una secuencia de código en una sola linea y para esto podemos crear funciones de manera sencilla en R

Funciones

Las funciones te permitirán automatizar algunas tareas comunes de una forma más poderosa y general que copiar-y-pegar.

Una función se ve como lo siguiente:

```
nombre_de_funcion <- function(arg1,arg2, ...) {
  ...
  #cuerpo
  ...
  return(resultado)
}
```

Funciones

Entre algunas de las ventajas tenemos:

- Facilidad de comprensión en tu código.
- Facilidad de edición del código.
- Eliminar posibles errores en código.

Funciones

Entre algunas de las ventajas tenemos:

- Facilidad de comprensión en tu código.
- Facilidad de edición del código.
- Eliminar posibles errores en código.

Pregunta:

¿Cuando deberías escribir una función?

Consideraciones para las funciones

- Utilizar nombres simples, descriptivos y entendibles

Consideraciones para las funciones

- Utilizar nombres simples, descriptivos y entendibles
- Tener cuidado con las coincidencias de nombres

Consideraciones para las funciones

- Utilizar nombres simples, descriptivos y entendibles
- Tener cuidado con las coincidencias de nombres
- Utilizar convenciones de nombres

Consideraciones para las funciones

- Utilizar nombres simples, descriptivos y entendibles
- Tener cuidado con las coincidencias de nombres
- Utilizar convenciones de nombres
- Documentar

Tabla de Contenidos

ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA

Estructuras de control

Como su nombre lo indica, las estructuras de control nos permiten controlar la manera en que se ejecuta el código.

Las estructuras de control establecen condicionales en nuestros código. Por ejemplo, qué condiciones deben cumplirse para realizar una operación o qué debe ocurrir para ejecutar una función.

Esto es de gran utilidad para determinar la lógica y el orden en que ocurren las operaciones, en especial al definir funciones. Las principales estructuras de control son:

- *if-else*
- *for*
- *while*

If, elseif, else

Una sentencia *if* (si) te permite ejecutar un código condicional, en otras palabras ejecutar una secuencia de código distinto a partir del valor de verdad de una condición. Un *if* se ve de la siguiente manera:

Si en el cuerpo del *else* deseamos encadenar otro *if* podemos usar el comando *ifelse (condición2)*

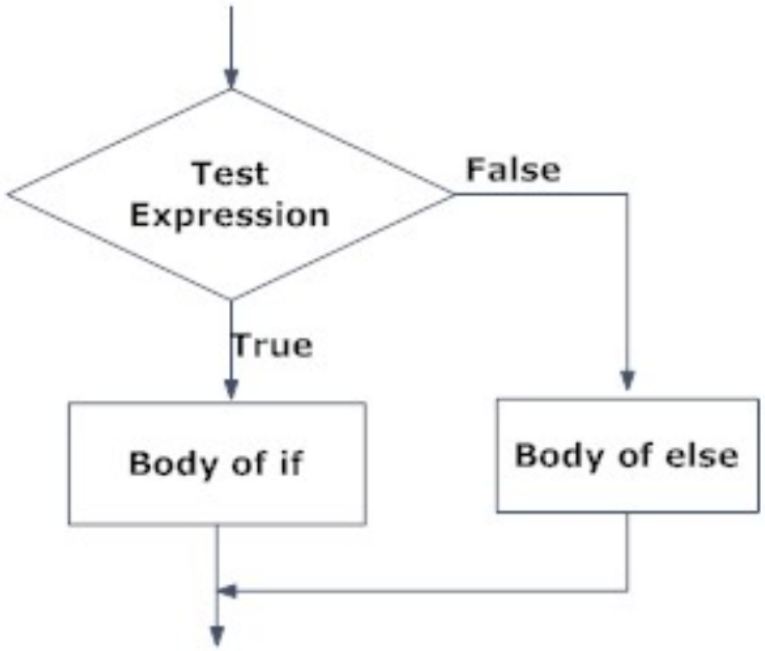

Figura 9: Esquema de operador *if*

If, elseif, else

Una sentencia *if* (si) te permite ejecutar un código condicional. Un *if* se ve de la siguiente manera:

```
if (condicion) {
    # el código que se ejecuta cuando
    la condición es verdadera (TRUE)
} else {
    # el código que se ejecuta cuando
    la condición es falsa (FALSE)
}
```

Si en el cuerpo del *else* deseamos encadenar otro *if* podemos usar el comando *ifelse (condición2)*

ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA

For loop

La estructura *for* nos permite ejecutar un bucle (loop), realizando una operación o secuencia de código para cada elemento de un conjunto de datos.

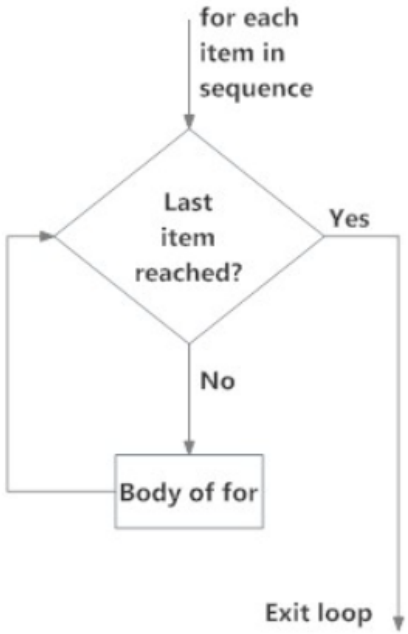

Figura 10: Esquema de operador *for*

For loop

Podemos identificar 2 partes principales en un *for*

- Secuencia: vector sobre el cual el elemento va tomando sus valores
- Cuerpo: código que se ejecuta y depende del valor del elemento

```
for(elemento in secuencia) {
  #cuerpo
}
```

ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA

While loop

Este es un tipo de bucle que ocurre mientras una condición es verdadera. La operación se realiza hasta que se se llega a cumplir un criterio previamente establecido. El modelo de *while* se puede expresar como

```
while(condicion) {
  operaciones
}
```

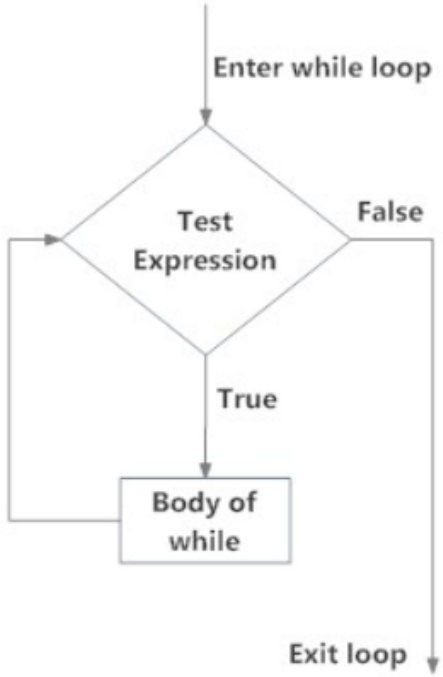

Figura 11: Esquema de operador *while*