



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

El uso de apuntes de clases estará reservado para finalidades académicas. La reproducción total o parcial de los mismos por cualquier medio, así como su difusión y distribución a terceras personas no está permitida, salvo con autorización del autor.

Tabla de Contenidos

Introducción a R

Funciones

Operador pipe

Estructuras de control

If, elseif, else

Lógica en R

For loop

While loop

Estructuras de datos en R

Transformaciones de dataframes



Tabla de Contenidos

Introducción a R

Funciones

Estructuras de control

Estructuras de datos en R



¿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 13 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)



Consideraciones generales de R

- Consola y entorno interactivo



Consideraciones generales de R

- Consola y entorno interactivo
- Constantes y nombres



Consideraciones generales de R

- Consola y entorno interactivo
- Constantes y nombres
- Documentación



Consideraciones generales de R

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



Tabla de Contenidos

Introducción a R

Funciones

Operador pipe

Estructuras de control

Estructuras de datos en R



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)  
}
```



Entre algunas de las ventajas tenemos:

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



Entre algunas de las ventajas tenemos:

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

Pregunta:

¿Cuándo 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



Operador pipe

Importante entender el **operador pipe de dplyr** (`%>%`):

Sirve para concatenar operaciones de dplyr.

Ejemplo, digamos que necesitamos aplicar mas de una función, la instrucción seria:

tercero(segundo(primero(data)))

Usando el operador pipe nos permite escribir una secuencia de funciones de izquierda a derecha. Si es que alguna de las funciones tuviera algún parametro de entrada lo ponemos en el parentesis correspondiente:

primero(data) %>% segundo(param₂) %>% third(param₃)



Introducción a R

Funciones

Estructuras de control

If, elseif, else

Lógica en R

For loop

While loop

Estructuras de datos en R



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)

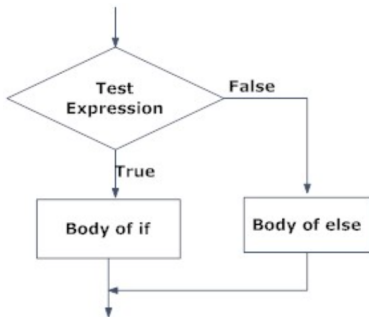


Figura 1: Esquema de operador *if*



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)



Operadores relacionales

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

Operador	Comparación	Ejemplo	Resultado
<code>x y</code>	x Ó y es verdadero	<code>TRUE FALSE</code>	<code>TRUE</code>
<code>x & y</code>	x Y y son verdaderos	<code>TRUE & FALSE</code>	<code>FALSE</code>
<code>!x</code>	x no es verdadero (negación)	<code>!TRUE</code>	<code>FALSE</code>

Figura 2: Operadores relacionales en R



Operaciones lógicas

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
<code>x y</code>	x Ó y es verdadero	<code>TRUE FALSE</code>	<code>TRUE</code>
<code>x & y</code>	x Y y son verdaderos	<code>TRUE & FALSE</code>	<code>FALSE</code>
<code>!x</code>	x no es verdadero (negación)	<code>!TRUE</code>	<code>FALSE</code>

Figura 3: Operadores lógicos en R



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.

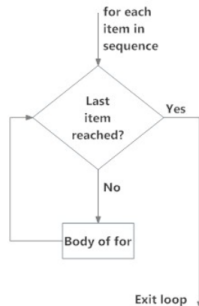


Figura 4: Esquema de operador *for*



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  
}
```



While loop

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

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

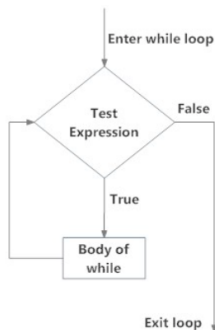


Figura 5: Esquema de operador *while*



Tabla de Contenidos

Introducción a R

Funciones

Estructuras de control

Estructuras de datos en R

Transformaciones de dataframes



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 homogéneas o heterogéneas.

Dimensiones	Homogéneas	Heterogéneas
1	Vector	Lista
2	Matriz	Data frame
n	Array	

Figura 6: Estructuras de datos



Revisaremos solo los vectores, listas y dataframes.

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.



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).



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).



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



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



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.



Uno de nuestros objetivos dentro del curso y también la ciencia de datos es llevar nuestro 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



Este set de reglas se puede visualizar más fácilmente:

pais	anio	casos	poblacion
Afganistán	1999	745	19987071
Afganistán	2000	2666	20595360
Brasil	1999	37737	172006362
Brasil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

variables

pais	anio	casos	poblacion
Afganistán	1999	745	19987071
Afganistán	2000	2666	20595360
Brasil	1999	37737	172006362
Brasil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

observaciones

pais	anio	casos	poblacion
Afganistán	1999	745	19987071
Afganistán	2000	2666	20595360
Brasil	1999	37737	172006362
Brasil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

valores

Figura 7: Reglas que hacen que un conjunto de datos sea ordenado: las variables están en columnas, las observaciones en filas y los valores en celdas.



Al enfrentarse a un set de datos es importante:

1. Entender cuales son las observaciones y cuales las variables



Al enfrentarse a un set de datos es importante:

1. Entender cuales son las observaciones y cuales las variables
2. Resolver los problemas:



Al enfrentarse a un set de datos es importante:

1. Entender cuales son las observaciones y cuales las variables
2. Resolver los problemas:
 - Una variable se extiende por varias columnas



Al enfrentarse a un set de datos es importante:

1. Entender cuales son las observaciones y cuales las variables
2. Resolver los problemas:
 - Una variable se extiende por varias columnas
 - Una observación está dispersa entre múltiples filas.



El primer caso lo resolvemos con la función *pivot_longer*.

pais	anio	casos
Afganistán	1999	745
Afganistán	2000	2666
Brasil	1999	37737
Brasil	2000	80488
China	1999	212258
China	2000	213766

pais	1999	2000
Afganistán	745	2666
Brasil	37737	80488
China	212258	213766

Tabla 4

Figura 8:



El el segundo lo resolvemos con la función *pivot_wide*.

pais	anio	tipo	casos
Afganistán	1999	casos	745
Afganistán	1999	población	19987071
Afganistán	2000	casos	2666
Afganistán	2000	población	20595360
Brasil	1999	casos	37737
Brasil	1999	población	172006362
Brasil	2000	casos	80488
Brasil	2000	población	174504898
China	1999	casos	212258
China	1999	población	1272915272
China	2000	casos	213766
China	2000	población	1280428583

pais	anio	casos	poblacion
Afganistán	1999	745	19987071
Afganistán	2000	2666	20595360
Brasil	1999	37737	172006362
Brasil	2000	80488	17504898
China	1999	212258	1272915272
China	2000	213766	1280428583


Tabla 2

Figura 9:



Separar y unir

Puede ser que alguna de nuestras columnas tenga mas de un dato y por esto nos conviene separarlo en dos o mas variables. Esto lo logramos con la función *separate*, su inverso es la funcion *unite*



The diagram illustrates the process of separating a single column into two columns. A curved arrow points from the 'tasa' column of the first table to the 'casos' and 'poblacion' columns of the second table, indicating the transformation.

pais	anio	tasa
Afganistán	1999	745 / 19987071
Afganistán	2000	2666 / 20595360
Brasil	1999	37737 / 172006362
Brasil	2000	80488 / 17504898
China	1999	212258 / 1272915272
China	2000	213766 / 1280428583

pais	anio	casos	poblacion
Afganistán	1999	745	19987071
Afganistán	2000	2666	20595360
Brasil	1999	37737	172006362
Brasil	2000	80488	17504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Tabla 3

Figura 10:



Valores faltantes

Cambiar la forma de representar los datos puede llevarnos a descubrir una mayor cantidad de valores faltantes o NA (Not Available).

Existen dos formas en las que puede aparecer un valor NA:

- Explícita, esto es, aparece como NA.
- Implícita, esto es, simplemente no aparece en los datos.

anio	trimestre	retorno
<dbl>	<dbl>	<dbl>
2015	1	1.88
2015	2	0.59
2015	3	0.35
2015	4	NA
2016	2	0.92
2016	3	0.17
2016	4	2.66

Figura 11: Tabla con valores faltantes

