Curso Primeros Pasos en R

Clase 2: Tipos y estructura de datos en R

Pontificia Universidad Católica de Chile

Noviembre 2021

Clase 2: Tipos y estructura de datos en R

- Tipos de datos
- Vectores
- Matrices
- Data frame
- Tibble
- Valores no disponibles
- Listas
- Operadores lógicos
- Actividad

Tipos de datos

En R existen distintas clases y tipos de datos. Estos dependen del tipo de información que contengan en su interior. Con los comandos `class()` y `typeof()` se puede preguntar qué tipo de dato es cada elemento.

Los principales tipos de datos en R son los siguientes:

Tipo de dato	Descripción	Ejemplo
`integer`	Números enteros	-1, 0, 1
`numeric`	Números reales	-0.5, 1/2, 1
`character`	Texto	"palabra", "y"
`logical`	Verdadero o Falso	'TRUE' 'FALSE'

Hay muchos otros tipos de datos en R, por ejemplo, hay datos `complex` (números complejos), `date` (fechas), `factor` (datos categóricos), etc.

Vectores

Un vector es un ordenamiento unimensional de datos. Dentro de un vector solo puede existir un tipo de dato. En R, los vectores se definen con el comando `c()` y cada elemento se separa con una coma:

Vector numérico

```
vector_1 <- c(-1/2, 1, 0.5)
class(vector_1)
```

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

Vector con letras

```
vector_2 <- c("A", "B", "C")
class(vector_2)
```

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

¿Qué tipo de dato será el vector `c(1, "dos", 3)`?

Funciones para generar vectores

En la siguiente tabla, se muestran algunas de las principales funciones usadas para facilitar la creación de vectores en R:

Función	Descripción	Ejemplo
a:b	Genera una secuencia de números naturales entre a y b	`5:10`
seq()	Genera una secuencia de números a un intervalo regular	`seq(from= -5,to=5,by= 0.5)`
rep()	Genera una secuencia de repetición de elementos de un vector base	`rep(c("a","b"),times=3)`
letters	Vector constante con las letras desde la "a" a la "z" en minuscula	`letters`
LETTERS	Vector constante con las letras desde la "A" a la "Z" en mayúscula	`LETTERS`

Funcionalidades adicionales

Con el comando `length()` se puede extraer el largo de un vector.

Se puede consultar en R si un elemento es o no un vector con el comando `is.vector()`

```
is.vector( c(1, 2, 3) )
```

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

De la misma forma, se puede transformar un objeto a un vector con el comando `as.vector()`.

Seleccionar un elemento dentro de un vector

Para seleccionar el elemento de un vector en R se utilizan los corchetes `[]`, de la siguiente forma:

`objeto[i]` extrae el i-ésimo elemento del vector llamado *objeto*

Matrices

Una matriz (`matrix`) es un arreglo de datos en una estructura bi-dimensional, entendidas como filas y columnas. En R, las matrices funcionan como una extensión de los vectores, dado que dentro de una matriz solo puede existir un tipo de dato. Hay muchas formas de generar matrices, la manera más directa es usando el comando `matrix()`:

```
matrix(vector, ncol = n, n_row = m, byrow = FALSE )
```

```
matrix(1:9, ncol = 3)
```

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

```
matrix(letters[1:9], ncol = 3)
```

```
##      [,1] [,2] [,3]
## [1,] "a"  "d"  "g"
## [2,] "b"  "e"  "h"
## [3,] "c"  "f"  "i"
```

Concatenación de vectores

Se puede transformar un conjunto de vectores en una matriz con ayuda de los comandos `cbind()` y . El primero, junta vectores de manera vertical (columna), y el segundo de manera horizontal (fila):

```
v_1 <- 1:5
v_2 <- LETTERS[1:5]
```

Concatenación por columnas (c)

```
cbind(v_1, v_2)
```

```
##      v_1 v_2
## [1,] "1" "A"
## [2,] "2" "B"
## [3,] "3" "C"
## [4,] "4" "D"
## [5,] "5" "E"
```

Concatenación por filas (r)

```
rbind(v_1, v_2)
```

```
##     [,1] [,2] [,3] [,4] [,5]
## v_1 "1"  "2"  "3"  "4"  "5"
## v_2 "A"  "B"  "C"  "D"  "E"
```

Funcionalidades adicionales

Seleccionar elementos de una matriz

En R, los arreglos bi-dimensionales funcionan siempre con el orden (filas, columnas). Utilizando esta lógica, es posible rescatar elementos desde una matriz:

`nombre_matriz[i, j]` ~ i=filas, j=columnas

Ejemplos:

- `nombre_matriz[i,]`: Selecciona la i-ésima **fila** de la matriz.
- `nombre_matriz[,j]`: Selecciona la j-ésima **columna** de la matriz.
- `nombre_matriz[i,j]`: Selecciona el j-ésimo elemento de la i-ésima fila.

El comando `dim()` permite obtener las dimensiones de la matriz. El primer número corresponde al número de filas, el segundo al número de columnas.

Se puede consultar si un objeto es una matriz con el comando `is.matrix()` y se puede transformar un objeto a una matriz con el comando `as.matrix()`

Data frame

Un data frame es un arreglo de datos en una estructura bi-dimensional. A diferencia de una matriz, un data frame puede tener columnas de distintos tipos de datos. En un data frame, las columnas representan variables y las filas representan a individuos u observaciones.

	Variable 1	Variable 2	...	Variable n
Observación 1	dato	dato	dato	dato
Observación 2	dato	dato	dato	dato
...	...	...	...	...
Observación m	dato	dato	dato	dato

Los data frame en R se pueden generar directamente con el comando `data.frame()` o se pueden transformar otras estructuras definidas. Por ejemplo, una matriz, con el comando `as.data.frame()`

Creación de un data frame

El comando data.frame() se puede usar de distintas formas:

Forma 1: Usar vectores ya creados. En este, todos los vectores deben ser del mismo largo. El nombre de cada vector se guardará como el nombre de cada columna del data frame.

```
var_1 <- c(a1, a1,..., an)
...
var_m <- c(z1, z2,..., zm)

base_ejemplo <- data.frame(var_1, var_2, ... , var_n)
```

Forma 2: Definir directamente las columnas dentro del dataframe

```
base_ejemplo <- data.frame(var_1 = c(a1, a1,..., an),
                           ...,
                           var_m = c(z1, z2,..., zm))
```

Ejemplo: Creación de un data frame

Considere la siguiente tabla:

Nombre	Grupo Sanguíneo	Altura (cm)
Andrea	AB	165
Bastian	O	180
Camilo	A	158
Daniela	B	170

Defina esta tabla como un data frame de las dos formas anteriores:

Forma 1

1.- Se definen las columnas de las matrices en vectores:

```
nombre <- c("Andrea", "Bastian", "Camilo", "Daniela")
grupo_s <- c("AB", "0", "A", "B")
altura_cm <- c(165, 180, 158, 170)
```

2.- Se guanra en el objeto 'tabla' el data frame:

```
tabla <- data.frame(nombre, grupo_s, altura_cm)
tabla
```

```
##    nombre grupo_s altura_cm
## 1  Andrea      AB       165
## 2 Bastian       0       180
## 3  Camilo       A       158
## 4 Daniela       B       170
```

```
names(tabla)
```

```
## [1] "nombre"    "grupo_s"   "altura_cm"
```

Forma 2

Se definen directamete los vectores dentro del comando `data.frame()`:

```
tabla <- data.frame(
  nombre = c("Andrea", "Bastian", "Camilo", "Daniela"),
  grupo_s = c("AB", "0", "A", "B"),
  altura_cm = c(165, 180, 158, 170)
)
```

```
tabla
```

```
##    nombre grupo_s altura_cm
## 1  Andrea      AB       165
## 2 Bastian       0       180
## 3  Camilo       A       158
## 4 Daniela       B       170
```

```
names(tabla)
```

```
## [1] "nombre"    "grupo_s"   "altura_cm"
```

Selección de observaciones de un data frame

Un data frame funciona igual que una matriz, ya que se puede seleccionar directamente filas y columnas específicas con el uso de corchetes `[]`.

En los data frame, es normal que las variables (columnas) tengan nombre. En R, se pueden usar tales nombres para seleccionar elementos de la base de datos:

Seleccionar variables dado nombre de variables

```
nombre_base$nombre_variable
nombre_base[["nombre_variable"]]
```

Seleccionar observaciones de un data frame

- `base[i,]`: Todas las columnas, i-ésima observación (fila).
- `base[i,j]`: i-ésima fila, j-ésima variable (columna).
- `base[v_filas, v_columnas]`: todas las filas que están en `v_filas` y columnas en `v_columnas`.

Tibble

Los tibbles son data frames, pero tienen características que hacen su manejo más fácil y más acorde con la actualidad. La librería `tibble` pertenece a la librería `tidyverse`, que se verá en profundidad en la próxima clase. Se instalará inmediatamente:

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

Para transformar un data frame o una matriz a un objeto `tibble`, basta con utilizar el comando `as_tibble()` de la siguiente forma:

```
base_tibble <- as_tibble(base_dataframe)
```

El comando `tbl_df()` de la librería `dplyr`, librería presente en `tidyverse()`, permite transformar directamente un data frame en un tibble.

Dependiendo de cómo se importan las bases de datos externas, es probable que las bases de datos ya estén con formato `tibble`.

Beneficios de usar `tibble`

Impresión en la consola

En comparación a un data frame, los tibble se imprimen de una manera mucho más limpia en la consola, representando un resumen claro de la base de datos:

```
cars
```

```
##    speed dist
## 1      4    2
## 2      4   10
## 3      7    4
## 4      7   22
## 5      8   16
## 6      9   10
## 7     10   18
## 8     10   26
## 9     10   34
## 10    11   17
## 11    11   28
## 12    12   14
## 13    12   20
## 14    12   24
## 15    12   28
## 16    13   26
## 17    13   34
## 18    13   34
## 19    13   46
```

```
cars_tibble <- as_tibble(cars)
cars_tibble
```

```
## # A tibble: 50 x 2
##    speed  dist
##    <dbl> <dbl>
##  1     4     2
##  2     4    10
##  3     7     4
##  4     7    22
##  5     8    16
##  6     9    10
##  7    10    18
##  8    10    26
##  9    10    34
## 10    11    17
## # ... with 40 more rows
```

Los beneficios de impresión `tibble` son más notorios cuando se tienen bases de datos amplias, dado que solo muestran las primeras 10 filas y solo aquellas columnas que se pueden visualizar en el ancho de la pantalla.

```
iris_tibble <- as_tibble(iris)
iris_tibble
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # ... with 140 more rows
```

```
iris
```

```
##     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
## 4            4.6         3.1          1.5         0.2     setosa
## 5            5.0         3.6          1.4         0.2     setosa
## 6            5.4         3.9          1.7         0.4     setosa
## 7            4.6         3.4          1.4         0.3     setosa
## 8            5.0         3.4          1.5         0.2     setosa
## 9            4.4         2.9          1.4         0.2     setosa
## 10           4.9         3.1          1.5         0.1     setosa
## 11           5.4         3.7          1.5         0.2     setosa
## 12           4.8         3.4          1.6         0.2     setosa
## 13           4.8         3.0          1.4         0.1     setosa
## 14           4.3         3.0          1.1         0.1     setosa
## 15           5.8         4.0          1.2         0.2     setosa
## 16           5.7         4.4          1.5         0.4     setosa
## 17           5.4         3.9          1.3         0.4     setosa
## 18           5.1         3.5          1.4         0.3     setosa
## 19           5.7         3.8          1.7         0.3     setosa
## 20           5.1         3.8          1.5         0.3     setosa
## 21           5.4         3.4          1.7         0.2     setosa
## 22           5.1         3.7          1.5         0.4     setosa
## 23           4.6         3.6          1.0         0.2     setosa
## 24           5.1         3.3          1.7         0.5     setosa
## 25           4.8         3.4          1.9         0.2     setosa
## 26           5.0         3.0          1.6         0.2     setosa
## 27           5.0         3.4          1.6         0.4     setosa
## 28           5.2         3.5          1.5         0.2     setosa
## 29           5.2         3.4          1.4         0.2     setosa
## 30           4.7         3.2          1.6         0.2     setosa
## 31           4.8         3.1          1.6         0.2     setosa
```

Creación de un tibble

Los tibble se definen de la misma forma que un data frame.
Volviendo al ejemplo anterior:

```
tabla_tibble <- tibble(
  nombre = c("Andrea", "Bastian", "Camilo", "Daniela"),
  grupo_s = c("AB", "0", "A", "B"),
  altura_cm = c(165, 180, 158, 170)
)

tabla_tibble
```

```
## # A tibble: 4 x 3
##   nombre  grupo_s altura_cm
##   <chr>   <chr>       <dbl>
## 1 Andrea  AB            165
## 2 Bastian 0             180
## 3 Camilo  A             158
## 4 Daniela B             170
```

De la misma manera, la forma de selección de variables y observaciones de un data frame funcionan en los tibble (`$`, `[[]]`, `[]`).

Valores no disponibles (NA)

En R, los valores no disponibles (celdas vacías, valores perdidos, etc) se representan con `NA`. Al tener elementos nulos dentro de un objeto de R **no** afecta la clase de este, por ejemplo:

```
vec_ejemplo <- c(1, NA, 3, NA, 5)
class(vec_ejemplo)
```

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

```
tib_ejemplo <- tibble(
  nombre = c("Claudio", "Javiera", "Elias", NA, "Camila"),
  valor = c(10, NA, 7, NA, 15)
)
str(tib_ejemplo)
```

```
## tibble [5 x 2] (S3: tbl_df/tbl/data.frame)
##  $ nombre: chr [1:5] "Claudio" "Javiera" "Elias" NA ...
##  $ valor : num [1:5] 10 NA 7 NA 15
```

Omisión de NAs

En algunas ocasiones, se necesitará eliminar las observaciones `NA` de algunos elementos, debido a que trabajar con datos incompletos, genera problemas computacionales. El comando `na.omit()` limpia aquellos objetos de valores faltantes. Por ejemplo:

```
na.omit(vec_ejemplo)
```

```
## [1] 1 3 5
## attr(,"na.action")
## [1] 2 4
## attr(,"class")
## [1] "omit"
```

```
na.omit(tib_ejemplo)
```

```
## # A tibble: 3 x 2
##   nombre  valor
##   <chr>   <dbl>
## 1 Claudio    10
## 2 Elias       7
## 3 Camila     15
```

Omisión de NAs

Algunas funciones contienen argumentos que permiten omitir las observaciones NA de los elementos para realizar sus calculos. Por ejemplo, en la función `sum()` existe el argumento `na.rm` que elimina los NA del cálculo de la suma:

```
sum(vec_ejemplo)
```

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

```
sum(vec_ejemplo, na.rm = TRUE)
```

```
## [1] 9
```

Listas

Una lista es una estructura de datos unidimensional, con la diferencia que en cada elemento de la lista, pueden haber elementos de distintas clases. Estas se definen con el comando `list()` de la misma forma que se define un data frame o un tibble.

```
vec <- LETTERS[1:5]
mat <- matrix(1:9, ncol = 3)

lista <- list("Elemento_1" = vec,
              "Elemento_2" = mat)
lista
```

```
## $Elemento_1
## [1] "A" "B" "C" "D" "E"
##
## $Elemento_2
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

Seleccionar elementos de listas

Se pueden seleccionar elementos de listas usando `$` o `[[]]`, seleccionando con el nombre del elemento o la posición:

```
lista$Elemento_1
```

```
## [1] "A" "B" "C" "D" "E"
```

```
lista[[2]]
```

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

¿Cómo se puede obtener la tercera fila del segundo elemento de la lista?

Operadores logicos

En R hay operadores lógicos. Estos permiten realizar preguntas a R, las que serán respondidas como verdadero (`TRUE`) o falso (`FALSE`). Algunos operadores básicos son:

Operador	Código
Menor que	<
Mayor que	>
Menor o igual que	<=
Mayor o igual que	>=
Igual que	==
Distinto que	!=
Intersección	&
Unión	|
Dentro de	%in%

```
10 > 5
```

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

```
"Gato" == "gato"
```

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

```
c("a", "B", "c") %in% letters
```

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

Usos de operadores lógicos

Los operadores tienen múltiples usos. Principalmente, se usan para generar condiciones en la creación de funciones y filtrar bases de datos y/o vectores. Utilizando el ejemplo anterior:

```
tabla
```

```
##    nombre grupo_s altura_cm
## 1  Andrea      AB       165
## 2 Bastian       0       180
## 3  Camilo       A       158
## 4 Daniela       B       170
```

Suponga que se quiere seleccionar aquellas filas en donde la altura sea menor a 170 cm. Esto, se puede realizar de la siguiente forma:

```
tabla[tabla$altura_cm < 170, ]
```

```
##   nombre grupo_s altura_cm
## 1 Andrea      AB       165
## 3 Camilo       A       158
```

Actividad

Actividad

La base de datos `viviendasRM.xlsx` contiene avisos de viviendas usadas que se venden en la Región Metropolitana de Chile recolectados de Chile Propiedades en mayo 2020. La base de datos contiene 1139 observaciones de 13 variables (una descripción de esta base se encuentra en la siguiente slide). **Fuente:** Kaggle.

Mediante selección de elementos del dataframe y operadores logicos, indique:

a. Toda la información del cuarto registro.

b. El número de baños de la vivienda del registro 8.

c. El numero de estacionamientos de la vivienda 14.

d. Total de superficie M2 y valor en UF, para las viviendas desde la quinta a la décima posición.

e. Listado de superficie M2, corredor y valor en UF de viviendas cuya superficie M2 es igual a 100 m2.

Descripción de la base de datos:

Variable	Descripción
Comuna	Comuna de la región en la cual se encuentra la casa.
Link	Enlace al aviso de la vivienda.
Tipo_Vivienda	Tipo de vivienda.
N_Habitaciones	Número de habitaciones.
N_Banos	Número de baños.
N_Estacionamientos	Número de estacionamientos.
Total_Superficie_M2	Total de superficie en mt2.
Superficie_Construida_M2	Superficie construida en mt2: metros de la construcción habitable.
Valor_UF	Valor en Unidades de fomentos de la propiedad declarado en el portal.
Valor_CLP	Valor en pesos chilenos.
Direccion	Dirección de la casa del aviso.
Quien_Vende	Nombre de la persona que vende la vivienda.
Corredor	Indica nombre de la empresa que vende.

¡Gracias!

María Constanza Prado
mcprado@uc.cl

Emiliano Romero
emilianomm@uc.cl