Clase de ayudantía Semana I: Nociones básicas de R y RStudio

Luis Escobar

R es un software especializado en análisis estadístico con gran presencia en el mundo académico y en el mundo laboral. La flexibilidad que se tiene con él para manipular datos ha logrado posicionarlo dentro de las herramientas más completas que se tienen para obtener *insight* de un conjunto de datos. Adicionalmente, gracias a que es *software libre*, contribuyentes de todo el mundo han logrado acumular una gran cantidad de bibliotecas con funciones especializadas para resolver todo tipo de problemas que se presenten en el proceso de análisis de datos.

Nociones básicas de R.

Una de las funciones que más se encontrarán utilizando en R es **library**, que utilizamos para cargar una biblioteca y tener al alcance en cada sesión las funciones que necesitaremos para el problema que queramos resolver.

```
suppressMessages(library(tidyverse))
```

Para definir un directorio de trabajo, usamos la función "setwd"

```
setwd("/home/luis/Desktop/Ayudantia_CienciaDeDatosMdoCapitales/Clase_15-08-2018/")
```

Para obtener la ruta en la nos encontramos trabajando, usamos "getwd"

```
getwd()
```

```
## [1] "/home/luis/Desktop/Seminario_Ciencias/Notas_Ayudantia/Semana_1"
```

Para crear un nuevo objeto (o si lo prefieren, asignarle un nombre a un objeto) en R exite un operador especial "<-", que se prefiere en lugar del clásico "=" para evitar confusiones cuando en una misma linea u operación se estén usando ambos operadores, RStudio tiene un atajo que escribe el operador y lo rodea con espacios, "Alt + -", (tecla Alt y tecla guión) para evitar presionar ambas teclas ("<"" y "-"") cada que se asigne un nombre a un nuevo objeto, y que nos ayuda a conservar las conveciones que se siguen en R para tener un código legible y ordenado.

```
x <- 10

print(x)
```

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

Estructuras de datos en R

R tiene cuatro tipos de estructuras de datos principales, que se diferencian pricipalmente en su dimensión y en la variedad de tipos datos que pueden almacenar.

La siguiente table la tomamos de (Wickham 2015)

	Homogeneous	Heterogeneous
1d	Atomic vector	List
2d	Matrix	Data frame
nd	Array	

Figure 1: Estructuras de datos en R

Coerción de datos.

Para entender mejor el funcionamiento de las estructuras de datos en R, es necesario conocer las reglas de coerción que tiene el lenguaje para mantener consistencia. La coerción de datos se refiere a la conversición del tipo de datos que hacen los lenguajes de programación para mantener consistencia en un programa. En R, tenemos cuatro tipos principales de datos que podemos almacenar en alguna de las estructuras de datos que R soporta: lógicos, enteros, dobles y caracteres (texto).

Como veremos más adelante, dado que existen estructuras homogéneas de datos, R se vale de un conjunto de reglas de coerción para evitar incosistencias en un programa; dichas reglas están basadas en una jerarquía de flexibilidad de los tipos de datos que tenemos para convertirlos a algún tipo diferente. Del menos al más flexible, la lista es:

- booleano
- entero
- doble
- caracter

Así, al momento de crear un objeto homogéneo en el que haya mezclados diferentes tipos de datos, R convertirá de acuerdo a lo anterior, por ejemplo:

```
x <- c(TRUE, FALSE, 5, 10)
print(str(x))
```

```
##  num [1:4] 1 0 5 10
## NULL
```

Vemos que los valores lógicos fueron coercionados a enteros, específicamente, TRUE se convierte a 1 y FALSE a 0. Otro ejemplo:

```
y <- c(5, 9.999, FALSE)
```

```
str(y)
```

```
##  num [1:3] 5 10 0
```

```
y[2]
```

```
## [1] 9.999
```

```
y
```

```
## [1] 5.000 9.999 0.000
```

Vemos que, a pesar imprimir el valor del doble redondeado, el valor almacenado en memoria sí es el valor que asignamos directamente al objeto. Finalmente, sin importar qué intentemos guardar en un vector, si hay una entrada que sea texto, todo el vector será convertido a texto, dado que es el tipo de dato más flexible.

```r
a <- c(TRUE, FALSE, 5, 6, 9.999, "Texto")
```

```r
str(a)
```

```
##  chr [1:6] "TRUE" "FALSE" "5" "6" "9.999" "Texto"
```

Vale la pena notar que los valores lógicos no fueron convertidos a entero antes de ser convertidos a texto, por lo que en el vector se almacena la cadena tal cual la escribimos. Casi distinto es si hacemos lo siguiente:

```r
a <- c(F, T, 9.999, 10)
```

```r
str(a)
```

```
##  num [1:4] 0 1 10 10
```

```r
a <- c(a, "Falso")
```

```r
str(a)
```

```
##  chr [1:5] "0" "1" "9.999" "10" "Falso"
```

En el caso anterior, dado que los valores lógicos primero fueron transformado a entero, se almacena el valor entero como cadena de texto, “1” y “0”.

Vector atómico.

Un vector es la estructura más simple en R, es unidimensional y puede almacenar objetos de un sólo tipo de dato, es decir, no puedes tener en un vector una entrada numérica y otra booleana. Para crear un vector, utilizamos la función c().

```r
vector_1 <- c(1:90)
```

```r
str(vector_1)
```

```
##  int [1:90] 1 2 3 4 5 6 7 8 9 10 ...
```

La función str() nos imprime un resumen con las características principales de un objeto. En el caso anterior, vemos que “vector1” es un objeto con entradas enteras (int), de dimensión 1x90 ([1:90]), e imprime las primeras diez entradas del mismo. Los vectores siempre conservan su dimensión, no importa si los anidamos:

```r
vector_2 <- c(1, 2, c(3, 4, c(5, 6)))
```

```r
str(vector_2)
```

```
##  num [1:6] 1 2 3 4 5 6
```

Lista

La lista es la siguiente estructura de datos en términos de dimensión, la diferencia primordial es que una lista es heterogénea, es decir, en sus entradas puede almacenar cualquier tipo de datos, e incluso de otras estructuras de datos. Las listas son recursivas, lo que quiere decir que una lista puede almacenar listas que estén dentro de otras listas. Creamos una lista con la función list()

```r
lista_1 <- list(1:10, letters[1:10], lm(Sepal.Length ~ Sepal.Width, data = iris))
```

```r
str(lista_1)
```

```
## List of 3
##  $ : int [1:10] 1 2 3 4 5 6 7 8 9 10
```

```
##  $ : chr [1:10] "a" "b" "c" "d" ...
##  $ :List of 12
##   ..$ coefficients : Named num [1:2] 6.526 -0.223
##   .. ..- attr(*, "names")= chr [1:2] "(Intercept)" "Sepal.Width"
##   ..$ residuals    : Named num [1:150] -0.644 -0.956 -1.111 -1.234 -0.722 ...
##   .. ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
##   ..$ effects      : Named num [1:150] -71.566 -1.188 -1.081 -1.187 -0.759 ...
##   .. ..- attr(*, "names")= chr [1:150] "(Intercept)" "Sepal.Width" "" "" ...
##   ..$ rank         : int 2
##   ..$ fitted.values: Named num [1:150] 5.74 5.86 5.81 5.83 5.72 ...
##   .. ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
##   ..$ assign       : int [1:2] 0 1
##   ..$ qr           :List of 5
##   .. ..$ qr   : num [1:150, 1:2] -12.2474 0.0816 0.0816 0.0816 0.0816 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:150] "1" "2" "3" "4" ...
##   .. .. .. ..$ : chr [1:2] "(Intercept)" "Sepal.Width"
##   .. .. ..- attr(*, "assign")= int [1:2] 0 1
##   .. ..$ qraux: num [1:2] 1.08 1.02
##   .. ..$ pivot: int [1:2] 1 2
##   .. ..$ tol  : num 1e-07
##   .. ..$ rank : int 2
##   .. ..- attr(*, "class")= chr "qr"
##   ..$ df.residual  : int 148
##   ..$ xlevels      : Named list()
##   ..$ call         : language lm(formula = Sepal.Length ~ Sepal.Width, data = iris)
##   ..$ terms        :Classes 'terms', 'formula'  language Sepal.Length ~ Sepal.Width
##   .. .. ..- attr(*, "variables")= language list(Sepal.Length, Sepal.Width)
##   .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:2] "Sepal.Length" "Sepal.Width"
##   .. .. .. .. ..$ : chr "Sepal.Width"
##   .. .. ..- attr(*, "term.labels")= chr "Sepal.Width"
##   .. .. ..- attr(*, "order")= int 1
##   .. .. ..- attr(*, "intercept")= int 1
##   .. .. ..- attr(*, "response")= int 1
##   .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. ..- attr(*, "predvars")= language list(Sepal.Length, Sepal.Width)
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. .. ..- attr(*, "names")= chr [1:2] "Sepal.Length" "Sepal.Width"
##   ..$ model        :'data.frame':    150 obs. of  2 variables:
##   .. ..$ Sepal.Length: num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##   .. ..$ Sepal.Width : num [1:150] 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##   .. ..- attr(*, "terms")=Classes 'terms', 'formula'  language Sepal.Length ~ Sepal.Width
##   .. .. .. ..- attr(*, "variables")= language list(Sepal.Length, Sepal.Width)
##   .. .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. .. ..$ : chr [1:2] "Sepal.Length" "Sepal.Width"
##   .. .. .. .. .. ..$ : chr "Sepal.Width"
##   .. .. .. ..- attr(*, "term.labels")= chr "Sepal.Width"
##   .. .. .. ..- attr(*, "order")= int 1
##   .. .. .. ..- attr(*, "intercept")= int 1
##   .. .. .. ..- attr(*, "response")= int 1
##   .. .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
```

```
##   .. .. .. ..- attr(*, "predvars")= language list(Sepal.Length, Sepal.Width)
##   .. .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. .. .. ..- attr(*, "names")= chr [1:2] "Sepal.Length" "Sepal.Width"
##   ..- attr(*, "class")= chr "lm"
```

```
length(lista_1)
```

```
## [1] 3
```

Vemos que la lista que acabamos de crear es de tamaño (dimensión) 3, y que su tercer entrada es el resultado de un ajuste de un modelo lineal sobre el famoso conjunto de datos "iris", que es a su vez una lista. Más adelante veremos los fundamentos de la *programación funcional*, en la que las listas toman un papel importante.

Atributos en vectores y listas.

R permite agregar metadatos a los objetos que creamos, esto con la finalidad de tener más y mejores formas de trabajar con ellos, pues nos permiten crear referencias a los objetos dentro de dichas estructuras a partir de esos metadatos. Uno de los atributos más utilizados es el nombre. Como veremos adelante, cuando estamos recuperando información específica dentro de una estructura, referenciarla directamente con su nombre facilita la recuperación de la misma. Podemos agregar nombres a las entradas de un vector desde su creación:

```
vector_con_nombres_1 <- c(a = 1, b = 2, c = 3)
```

```
str(vector_con_nombres_1)
```

```
##  Named num [1:3] 1 2 3
##  - attr(*, "names")= chr [1:3] "a" "b" "c"
```

O podemos agregar un nombre después de crear el vector:

```
vector_con_nombres_2 <- c(1, 2, 3)
```

```
str(vector_con_nombres_2)
```

```
##  num [1:3] 1 2 3
```

```
attr(x = vector_con_nombres_2, "Name") <- c("a", "b", "c")
```

```
str(vector_con_nombres_2)
```

```
##  num [1:3] 1 2 3
##  - attr(*, "Name")= chr [1:3] "a" "b" "c"
```

O de manera más directa, con la función names()

```
vector_con_nombres_3 <- c(1, 2, 3)
```

```
str(vector_con_nombres_3)
```

```
##  num [1:3] 1 2 3
```

```
names(vector_con_nombres_3) <- c("a", "b", "c")
```

```
str(vector_con_nombres_3)
```

```
##  Named num [1:3] 1 2 3
##  - attr(*, "names")= chr [1:3] "a" "b" "c"
```

Podemos hacer lo mismo con una lista:

```
lista_con_nombres <- list(VecNum = 1:5, VecLet = letters[1:27],
                          LinMod = lm(Sepal.Length ~ Sepal.Width, iris))
```

```r
str(lista_con_nombres)
```

```
## List of 3
##  $ VecNum: int [1:5] 1 2 3 4 5
##  $ VecLet: chr [1:27] "a" "b" "c" "d" ...
##  $ LinMod:List of 12
##   ..$ coefficients : Named num [1:2] 6.526 -0.223
##   .. ..- attr(*, "names")= chr [1:2] "(Intercept)" "Sepal.Width"
##   ..$ residuals    : Named num [1:150] -0.644 -0.956 -1.111 -1.234 -0.722 ...
##   .. ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
##   ..$ effects      : Named num [1:150] -71.566 -1.188 -1.081 -1.187 -0.759 ...
##   .. ..- attr(*, "names")= chr [1:150] "(Intercept)" "Sepal.Width" "" "" ...
##   ..$ rank         : int 2
##   ..$ fitted.values: Named num [1:150] 5.74 5.86 5.81 5.83 5.72 ...
##   .. ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
##   ..$ assign       : int [1:2] 0 1
##   ..$ qr           :List of 5
##   .. ..$ qr   : num [1:150, 1:2] -12.2474 0.0816 0.0816 0.0816 0.0816 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:150] "1" "2" "3" "4" ...
##   .. .. .. ..$ : chr [1:2] "(Intercept)" "Sepal.Width"
##   .. .. ..- attr(*, "assign")= int [1:2] 0 1
##   .. ..$ qraux: num [1:2] 1.08 1.02
##   .. ..$ pivot: int [1:2] 1 2
##   .. ..$ tol  : num 1e-07
##   .. ..$ rank : int 2
##   .. ..- attr(*, "class")= chr "qr"
##   ..$ df.residual  : int 148
##   ..$ xlevels      : Named list()
##   ..$ call         : language lm(formula = Sepal.Length ~ Sepal.Width, data = iris)
##   ..$ terms        :Classes 'terms', 'formula'  language Sepal.Length ~ Sepal.Width
##   .. .. ..- attr(*, "variables")= language list(Sepal.Length, Sepal.Width)
##   .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:2] "Sepal.Length" "Sepal.Width"
##   .. .. .. .. ..$ : chr "Sepal.Width"
##   .. .. ..- attr(*, "term.labels")= chr "Sepal.Width"
##   .. .. ..- attr(*, "order")= int 1
##   .. .. ..- attr(*, "intercept")= int 1
##   .. .. ..- attr(*, "response")= int 1
##   .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. ..- attr(*, "predvars")= language list(Sepal.Length, Sepal.Width)
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. .. ..- attr(*, "names")= chr [1:2] "Sepal.Length" "Sepal.Width"
##   ..$ model        :'data.frame':    150 obs. of  2 variables:
##   .. ..$ Sepal.Length: num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##   .. ..$ Sepal.Width : num [1:150] 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##   .. ..- attr(*, "terms")=Classes 'terms', 'formula'  language Sepal.Length ~ Sepal.Width
##   .. .. .. ..- attr(*, "variables")= language list(Sepal.Length, Sepal.Width)
##   .. .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. .. ..$ : chr [1:2] "Sepal.Length" "Sepal.Width"
##   .. .. .. .. .. ..$ : chr "Sepal.Width"
##   .. .. .. ..- attr(*, "term.labels")= chr "Sepal.Width"
```

```
##   .. .. .. ..- attr(*, "order")= int 1
##   .. .. .. ..- attr(*, "intercept")= int 1
##   .. .. .. ..- attr(*, "response")= int 1
##   .. .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. .. ..- attr(*, "predvars")= language list(Sepal.Length, Sepal.Width)
##   .. .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. .. .. ..- attr(*, "names")= chr [1:2] "Sepal.Length" "Sepal.Width"
##   ..- attr(*, "class")= chr "lm"
```

Después de crear la lista:

```
lista_con_nombres_2 <- list(1:5, letters[1:27], lm(Sepal.Length ~ Sepal.Width,
                                                   iris))
names(lista_con_nombres_2) <- c("VecNum", "VecLet", "LinMod")
```

```
str(lista_con_nombres_2)
```

```
## List of 3
##  $ VecNum: int [1:5] 1 2 3 4 5
##  $ VecLet: chr [1:27] "a" "b" "c" "d" ...
##  $ LinMod:List of 12
##   ..$ coefficients : Named num [1:2] 6.526 -0.223
##   .. ..- attr(*, "names")= chr [1:2] "(Intercept)" "Sepal.Width"
##   ..$ residuals    : Named num [1:150] -0.644 -0.956 -1.111 -1.234 -0.722 ...
##   .. ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
##   ..$ effects      : Named num [1:150] -71.566 -1.188 -1.081 -1.187 -0.759 ...
##   .. ..- attr(*, "names")= chr [1:150] "(Intercept)" "Sepal.Width" "" "" ...
##   ..$ rank         : int 2
##   ..$ fitted.values: Named num [1:150] 5.74 5.86 5.81 5.83 5.72 ...
##   .. ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
##   ..$ assign       : int [1:2] 0 1
##   ..$ qr           :List of 5
##   .. ..$ qr   : num [1:150, 1:2] -12.2474 0.0816 0.0816 0.0816 0.0816 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:150] "1" "2" "3" "4" ...
##   .. .. .. ..$ : chr [1:2] "(Intercept)" "Sepal.Width"
##   .. .. ..- attr(*, "assign")= int [1:2] 0 1
##   .. ..$ qraux: num [1:2] 1.08 1.02
##   .. ..$ pivot: int [1:2] 1 2
##   .. ..$ tol  : num 1e-07
##   .. ..$ rank : int 2
##   .. ..- attr(*, "class")= chr "qr"
##   ..$ df.residual  : int 148
##   ..$ xlevels      : Named list()
##   ..$ call         : language lm(formula = Sepal.Length ~ Sepal.Width, data = iris)
##   ..$ terms        :Classes 'terms', 'formula'  language Sepal.Length ~ Sepal.Width
##   .. .. ..- attr(*, "variables")= language list(Sepal.Length, Sepal.Width)
##   .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:2] "Sepal.Length" "Sepal.Width"
##   .. .. .. .. ..$ : chr "Sepal.Width"
##   .. .. ..- attr(*, "term.labels")= chr "Sepal.Width"
##   .. .. ..- attr(*, "order")= int 1
##   .. .. ..- attr(*, "intercept")= int 1
##   .. .. ..- attr(*, "response")= int 1
```

```
##   .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. ..- attr(*, "predvars")= language list(Sepal.Length, Sepal.Width)
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. .. ..- attr(*, "names")= chr [1:2] "Sepal.Length" "Sepal.Width"
##   ..$ model        :'data.frame':    150 obs. of  2 variables:
##   .. ..$ Sepal.Length: num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##   .. ..$ Sepal.Width : num [1:150] 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##   .. ..- attr(*, "terms")=Classes 'terms', 'formula'  language Sepal.Length ~ Sepal.Width
##   .. .. .. ..- attr(*, "variables")= language list(Sepal.Length, Sepal.Width)
##   .. .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. .. ..$ : chr [1:2] "Sepal.Length" "Sepal.Width"
##   .. .. .. .. .. ..$ : chr "Sepal.Width"
##   .. .. .. ..- attr(*, "term.labels")= chr "Sepal.Width"
##   .. .. .. ..- attr(*, "order")= int 1
##   .. .. .. ..- attr(*, "intercept")= int 1
##   .. .. .. ..- attr(*, "response")= int 1
##   .. .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. .. ..- attr(*, "predvars")= language list(Sepal.Length, Sepal.Width)
##   .. .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. .. .. ..- attr(*, "names")= chr [1:2] "Sepal.Length" "Sepal.Width"
##   ..- attr(*, "class")= chr "lm"
```

Importante: los atributos de los objetos son guardados en una lista, por lo que un mismo objeto puede tener multiples atributos.

```
str(attributes(lista_con_nombres_2))
```

```
## List of 1
##  $ names: chr [1:3] "VecNum" "VecLet" "LinMod"
```

```
attr(lista_con_nombres_2, "Posicion") <- c(1, 2, 3)
```

```
str(attributes(lista_con_nombres_2))
```

```
## List of 2
##  $ names   : chr [1:3] "VecNum" "VecLet" "LinMod"
##  $ Posicion: num [1:3] 1 2 3
```

Algunas de las funciones que son de gran utilidad para la manipulación de los vectores y las listas son length(), que nos devuelve el tamaño (longitud), names(), que nos devuelve los nombres de las entradas de los objetos, si es que lo tienen.

```
names(lista_con_nombres)
```

```
## [1] "VecNum" "VecLet" "LinMod"
```

```
names(vector_con_nombres_1)
```

```
## [1] "a" "b" "c"
```

```
print(length(lista_con_nombres_2))
```

```
## [1] 3
```

```
print( length(vector_con_nombres_3))
```

```
## [1] 3
```

Podemos combinar vectores y listas con la función c():

```r
vector_combinado <- c(vector_con_nombres_1, vector_con_nombres_2)
```

```r
vector_combinado
```

```
## a b c
## 1 2 3 1 2 3
```

Vemos que los nombres deben ser únicos, por lo que los nombres del segundo vector son ignorados.

Para las listas:

```r
lista_combinada <- c(lista_con_nombres, lista_con_nombres_2)
```

```r
str(lista_combinada)
```

```
## List of 6
##  $ VecNum: int [1:5] 1 2 3 4 5
##  $ VecLet: chr [1:27] "a" "b" "c" "d" ...
##  $ LinMod:List of 12
##   ..$ coefficients : Named num [1:2] 6.526 -0.223
##   .. ..- attr(*, "names")= chr [1:2] "(Intercept)" "Sepal.Width"
##   ..$ residuals    : Named num [1:150] -0.644 -0.956 -1.111 -1.234 -0.722 ...
##   .. ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
##   ..$ effects      : Named num [1:150] -71.566 -1.188 -1.081 -1.187 -0.759 ...
##   .. ..- attr(*, "names")= chr [1:150] "(Intercept)" "Sepal.Width" "" "" ...
##   ..$ rank         : int 2
##   ..$ fitted.values: Named num [1:150] 5.74 5.86 5.81 5.83 5.72 ...
##   .. ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
##   ..$ assign       : int [1:2] 0 1
##   ..$ qr           :List of 5
##   .. ..$ qr   : num [1:150, 1:2] -12.2474 0.0816 0.0816 0.0816 0.0816 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:150] "1" "2" "3" "4" ...
##   .. .. .. ..$ : chr [1:2] "(Intercept)" "Sepal.Width"
##   .. .. ..- attr(*, "assign")= int [1:2] 0 1
##   .. ..$ qraux: num [1:2] 1.08 1.02
##   .. ..$ pivot: int [1:2] 1 2
##   .. ..$ tol  : num 1e-07
##   .. ..$ rank : int 2
##   .. ..- attr(*, "class")= chr "qr"
##   ..$ df.residual  : int 148
##   ..$ xlevels      : Named list()
##   ..$ call         : language lm(formula = Sepal.Length ~ Sepal.Width, data = iris)
##   ..$ terms        :Classes 'terms', 'formula'  language Sepal.Length ~ Sepal.Width
##   .. .. ..- attr(*, "variables")= language list(Sepal.Length, Sepal.Width)
##   .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:2] "Sepal.Length" "Sepal.Width"
##   .. .. .. .. ..$ : chr "Sepal.Width"
##   .. .. ..- attr(*, "term.labels")= chr "Sepal.Width"
##   .. .. ..- attr(*, "order")= int 1
##   .. .. ..- attr(*, "intercept")= int 1
##   .. .. ..- attr(*, "response")= int 1
##   .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. ..- attr(*, "predvars")= language list(Sepal.Length, Sepal.Width)
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
```

```
##   .. .. .. ..- attr(*, "names")= chr [1:2] "Sepal.Length" "Sepal.Width"
##   ..$ model        :'data.frame':    150 obs. of  2 variables:
##   .. ..$ Sepal.Length: num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##   .. ..$ Sepal.Width : num [1:150] 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##   .. ..- attr(*, "terms")=Classes 'terms', 'formula'  language Sepal.Length ~ Sepal.Width
##   .. .. .. ..- attr(*, "variables")= language list(Sepal.Length, Sepal.Width)
##   .. .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. .. ..$ : chr [1:2] "Sepal.Length" "Sepal.Width"
##   .. .. .. .. .. ..$ : chr "Sepal.Width"
##   .. .. .. ..- attr(*, "term.labels")= chr "Sepal.Width"
##   .. .. .. ..- attr(*, "order")= int 1
##   .. .. .. ..- attr(*, "intercept")= int 1
##   .. .. .. ..- attr(*, "response")= int 1
##   .. .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. .. ..- attr(*, "predvars")= language list(Sepal.Length, Sepal.Width)
##   .. .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. .. .. ..- attr(*, "names")= chr [1:2] "Sepal.Length" "Sepal.Width"
##   ..- attr(*, "class")= chr "lm"
##  $ VecNum: int [1:5] 1 2 3 4 5
##  $ VecLet: chr [1:27] "a" "b" "c" "d" ...
##  $ LinMod:List of 12
##   ..$ coefficients : Named num [1:2] 6.526 -0.223
##   .. ..- attr(*, "names")= chr [1:2] "(Intercept)" "Sepal.Width"
##   ..$ residuals    : Named num [1:150] -0.644 -0.956 -1.111 -1.234 -0.722 ...
##   .. ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
##   ..$ effects      : Named num [1:150] -71.566 -1.188 -1.081 -1.187 -0.759 ...
##   .. ..- attr(*, "names")= chr [1:150] "(Intercept)" "Sepal.Width" "" "" ...
##   ..$ rank         : int 2
##   ..$ fitted.values: Named num [1:150] 5.74 5.86 5.81 5.83 5.72 ...
##   .. ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
##   ..$ assign       : int [1:2] 0 1
##   ..$ qr           :List of 5
##   .. ..$ qr   : num [1:150, 1:2] -12.2474 0.0816 0.0816 0.0816 0.0816 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:150] "1" "2" "3" "4" ...
##   .. .. .. ..$ : chr [1:2] "(Intercept)" "Sepal.Width"
##   .. .. ..- attr(*, "assign")= int [1:2] 0 1
##   .. ..$ qraux: num [1:2] 1.08 1.02
##   .. ..$ pivot: int [1:2] 1 2
##   .. ..$ tol  : num 1e-07
##   .. ..$ rank : int 2
##   .. ..- attr(*, "class")= chr "qr"
##   ..$ df.residual  : int 148
##   ..$ xlevels      : Named list()
##   ..$ call         : language lm(formula = Sepal.Length ~ Sepal.Width, data = iris)
##   ..$ terms        :Classes 'terms', 'formula'  language Sepal.Length ~ Sepal.Width
##   .. .. ..- attr(*, "variables")= language list(Sepal.Length, Sepal.Width)
##   .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:2] "Sepal.Length" "Sepal.Width"
##   .. .. .. .. ..$ : chr "Sepal.Width"
##   .. .. ..- attr(*, "term.labels")= chr "Sepal.Width"
##   .. .. ..- attr(*, "order")= int 1
```

```
##   .. .. ..- attr(*, "intercept")= int 1
##   .. .. ..- attr(*, "response")= int 1
##   .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. ..- attr(*, "predvars")= language list(Sepal.Length, Sepal.Width)
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. .. ..- attr(*, "names")= chr [1:2] "Sepal.Length" "Sepal.Width"
##   ..$ model        :'data.frame':    150 obs. of  2 variables:
##   .. ..$ Sepal.Length: num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##   .. ..$ Sepal.Width : num [1:150] 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##   .. ..- attr(*, "terms")=Classes 'terms', 'formula'  language Sepal.Length ~ Sepal.Width
##   .. .. .. ..- attr(*, "variables")= language list(Sepal.Length, Sepal.Width)
##   .. .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. .. ..$ : chr [1:2] "Sepal.Length" "Sepal.Width"
##   .. .. .. .. .. ..$ : chr "Sepal.Width"
##   .. .. .. ..- attr(*, "term.labels")= chr "Sepal.Width"
##   .. .. .. ..- attr(*, "order")= int 1
##   .. .. .. ..- attr(*, "intercept")= int 1
##   .. .. .. ..- attr(*, "response")= int 1
##   .. .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. .. ..- attr(*, "predvars")= language list(Sepal.Length, Sepal.Width)
##   .. .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. .. .. ..- attr(*, "names")= chr [1:2] "Sepal.Length" "Sepal.Width"
##   ..- attr(*, "class")= chr "lm"
```

En la lista, vemos que los nombres sí se repiten, y tenemos 6 entradas de la lista, sin embargo, no es la mejor de las prácticas tener dos o más objetos con el mismo nombre dentro de una lista, pues esto puede generar resultados inesperados.

Matrices y Arreglos

Las matrices son arreglos de dos dimensiones, utilizados para la mayoria de las funciones de uso estadístico (la especialidad de R). Como se ve en la imagen inicial, las matrices son una estructura de datos homogénea, es decir, no es posible definir una matriz cuyas entradas sean de distintos tipos de datos. Existen tres maneras de definir una matriz: función matrix(), función array() y dándo dimensión a un vector atómico con la función dim()

```
matriz_1 <- matrix(1:9, nrow = 3, ncol = 3)
```

```
matriz_1
```

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

Si queremos que los número se acomoden en la matriz siguiendo las columnas y no los renglones, usamos un argumento más de la función matrix:

```
matriz_2 <- matrix(1:9, ncol = 3, byrow = T)
```

```
matriz_2
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

```
## [3,]    7    8    9
```

Dando dimensión a un vector:

```
matriz_vector <- c(1:9)
```

```
matriz_vector
```

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

```
dim(matriz_vector) <- c(3, 3)
```

```
matriz_vector
```

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

Utilizando la función array:

```
matriz_array <- array(1:9, c(3, 3, 1))
matriz_array
```

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

En el ejemplo anterior, vemos que las dimensiones del arreglo que queremos crear se especifican dentro de un vector que sirve como parámetro de la función array(). Podemos crear matrices con la función array(), pues las matrices son un caso especial de una estructura de n dimensiones. Los arreglons multidimensionales son poco utilizados en R, sin embargo, no está de más conocerlos.

```
arreglo_1 <- array(1:12, c(2, 3, 2))
arreglo_1
```

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

Igual que con las listas y los vectores, es posible asignar atributos a las matrices y los arreglos. Las funciones de tamaño y nombres de los objetos anteriores se generalizan para tomar en cuenta las dos dimensiones de las matrices, así:

```
rownames(matriz_1) <- letters[1:3]
colnames(matriz_1) <- LETTERS[1:3]
matriz_1
```

```
##   A B C
```

```
## a 1 4 7
## b 2 5 8
## c 3 6 9
```

```
ncol(matriz_1)
```

```
## [1] 3
```

```
nrow(matriz_1)
```

```
## [1] 3
```

Para un arreglo:

```
dim(arreglo_1)
```

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

```
dimnames(arreglo_1) <- list(c("row1", "row2"), c("col1", "col2", "col3"),
                            c("A", "B"))
arreglo_1
```

```
## , , A
##
##      col1 col2 col3
## row1    1    3    5
## row2    2    4    6
##
## , , B
##
##      col1 col2 col3
## row1    7    9   11
## row2    8   10   12
```

El output de la función str() para matrices y arreglos es diferente, sin importar si tienen las mismas dimensiones (sólo posible en el caso de un arreglo con "tercera" dimensión 1).

```
str(matriz_1)
```

```
##  int [1:3, 1:3] 1 2 3 4 5 6 7 8 9
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:3] "a" "b" "c"
##   ..$ : chr [1:3] "A" "B" "C"
```

```
str(arreglo_1)
```

```
##  int [1:2, 1:3, 1:2] 1 2 3 4 5 6 7 8 9 10 ...
##  - attr(*, "dimnames")=List of 3
##   ..$ : chr [1:2] "row1" "row2"
##   ..$ : chr [1:3] "col1" "col2" "col3"
##   ..$ : chr [1:2] "A" "B"
```

Finalmente, podemos combinar matrices con las funciones rbind() y cbind(), dependiendo de la manera que queramos hacerlo:

```
matriz_combinada_col <- cbind(matriz_1, matriz_2)
matriz_combinada_col
```

```
##   A B C
## a 1 4 7 1 2 3
## b 2 5 8 4 5 6
```

```
## c 3 6 9 7 8 9
```

```
matriz_combinada_ren <- rbind(matriz_1, matriz_2)
matriz_combinada_ren
```

```
##   A B C
## a 1 4 7
## b 2 5 8
## c 3 6 9
##   1 2 3
##   4 5 6
##   7 8 9
```

De nuevo, los nombres repetidos son ignorados.

Data Frames

Los Data Frames son arreglos heterogéneos de dos dimensiones, es decir, pueden contener multiples tipos de datos, **siempre y cuando** las columnas del data frame sean homogéneas cada una. Los data frames son una de las estructuras más usadas y más flexibles al momento de hacer análisis de datos. Un data frame es en realidad una lista formada de vectores de la misma dimensión, de ahí que las columnas de cada data frame son homogéneas. Para crear un data frame, utilizamos la función data.frame(), a la que es necesario pasar como argumentos vectores con nombres, que servirán como nombres de las columnas del data frame.

```
dataframe_1 <- data.frame(a = 1:3, b = letters[1:3])
dataframe_1
```

```
##   a b
## 1 1 a
## 2 2 b
## 3 3 c
```

Vemos que la columna b es convertida automáticamente a factor, lo que puede ocasionar muchos problemas al momento de automatizar una tarea, por lo que es mejor evitar dicho comportamiento (muchas de las opciones de carga de datos en R tienen argumentos extra para evitarlo), y sólo convertir a factores cuando sea necesario/deseado.

```
dataframe_2 <- data.frame(a = 1:3, b = letters[1:3], stringsAsFactors = F)
dataframe_2
```

```
##   a b
## 1 1 a
## 2 2 b
## 3 3 c
```

Podemos combinar data frames con las mismas funciones, cbind() y rbind(). Sin embargo, hay que tener en cuenta que el resultado de dichas funciones sólo será un data frame si los argumentos que le pasamos son ya data frames; si usamos la función cbind() con dos vectores, nos devolverá una matriz en su lugar.

Operador "<-" vs "="

Hay sútiles diferencias entre los dos operadores que pueden ocasionar problemas en nuestro código, por lo que es importante conocer qué los hace diferentes. El ejemplo más claro es el "alcance" (*scope*) de cada uno, por ejemplo:

```
# Borramos x si existe
rm(x)
```

```
mean(x = 1:10)
```

```
## [1] 5.5
```

```
x
```

```
## Error in eval(expr, envir, enclos): object 'x' not found
```

Vemos que el operador "=" sólo funciona dentro del scope de la función, pero no tiene ningún efecto en el ambiente globlal de nuestra sesión.

Ahora, intentamos lo mismo con el operador "<-"

```
rm(x)
```

```
## Warning in rm(x): object 'x' not found
```

```
mean(x <- 1:10)
```

```
## [1] 5.5
```

```
x
```

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

```
y <- mean(x)
```

```
y
```

```
## [1] 5.5
```

Por lo anterior, es recomendable siempre utilizar el operador "<-" para asignar un nombre a un objeto, pues al utilizarlo dentro de la llamada de una función corremos el riesgo de sobreescribir el valor de una variable en el ambiente global de la sesión, y si hay algún otro proceso dentro de nuetro código que depende del msimo objeto, podemos terminar con resultados muy distintos a los esperados.

Convenciones para escribir código

Cada quien tiene su estilo para escribir código, sin embargo, existen muchas convenciones que se utilizan primordialmente para facilitar la lectura del código que produzcamos. Esto se agredece bastante cuando el código se comparte y lo utilizan muchas personas, por ejemplo, en un grupo de trabajo.

Nombrar objetos

Una de las prácticas principales es hacer que los nombres de nuestros objetos sean lo más explicitos posibles, y sigan siempre un mismo patrón de nombramiento, con lo cual le estaremos haciendo la vida mucho más fácil a nuestros colaboradores. Las tres principales que se sugieren en (Wickham and Grolemund 2017) son:

Camel case

Combinación de letras mayúculas y minúsculas, siempre empezando con minúscula, y si es necesario un número. Por ejemplo:

- variableParaIterar1
- coeficienteRegresionB1

Snake case

La convención snake case nos dice que separemos las palabras que formas el nombre de nuestro objeto con guiones bajos.

- variable_para_iterar_1
- coeficiente_regresion_B1

Puntos

Similar al caso anterior, pero utilizando un punto en lugar de un guión bajo:

- variable.para.iterar.1
- coeficiente.regresion.B1

Indentado

Actualmente, la gran mayoria de los IDEs y de los editores de texto tienen implementando el indentado automático de nuestro código. El indentado resulta bastante útil visualmente cuando tenemos un código sumamente complejo, nos ayuda a localizar rápidamente la falta de algún paréntesis o llave de apertura o cierre, localizar parámetros de funciones dentro de una llamada a una función demasiados de ellos, etc.

```
# Es fácil anidas un ciclo for, pues el indentado lo hace automáticamente
# RStudio

# Código indentado
for (i in 1:length(seq(5))) {
  cat("Iteración ", i, " de ", length(seq(5)), " nivel 1.", "\n")

  for (j in 1:length(seq(5))) {
    cat("\tIteración ", j, " de ", length(seq(5)), " nivel 2.", "\n")
  }

}
```

```
## Iteración  1  de  5  nivel 1.
##  Iteración  1  de  5  nivel 2.
##  Iteración  2  de  5  nivel 2.
##  Iteración  3  de  5  nivel 2.
##  Iteración  4  de  5  nivel 2.
##  Iteración  5  de  5  nivel 2.
## Iteración  2  de  5  nivel 1.
##  Iteración  1  de  5  nivel 2.
##  Iteración  2  de  5  nivel 2.
##  Iteración  3  de  5  nivel 2.
##  Iteración  4  de  5  nivel 2.
##  Iteración  5  de  5  nivel 2.
## Iteración  3  de  5  nivel 1.
##  Iteración  1  de  5  nivel 2.
##  Iteración  2  de  5  nivel 2.
##  Iteración  3  de  5  nivel 2.
##  Iteración  4  de  5  nivel 2.
##  Iteración  5  de  5  nivel 2.
## Iteración  4  de  5  nivel 1.
```

```
##  Iteración  1  de  5  nivel 2.
##  Iteración  2  de  5  nivel 2.
##  Iteración  3  de  5  nivel 2.
##  Iteración  4  de  5  nivel 2.
##  Iteración  5  de  5  nivel 2.
## Iteración  5  de  5  nivel 1.
##  Iteración  1  de  5  nivel 2.
##  Iteración  2  de  5  nivel 2.
##  Iteración  3  de  5  nivel 2.
##  Iteración  4  de  5  nivel 2.
##  Iteración  5  de  5  nivel 2.
```

```
# Código sin indentar
for (i in 1:length(seq(5))) {
cat("Iteración ", i, " de ", length(seq(5)), " nivel 1.", "\n")
for (j in 1:length(seq(5))) {
cat("\tIteración ", j, " de ", length(seq(5)), " nivel 2.", "\n")
}
}
```

```
## Iteración  1  de  5  nivel 1.
##  Iteración  1  de  5  nivel 2.
##  Iteración  2  de  5  nivel 2.
##  Iteración  3  de  5  nivel 2.
##  Iteración  4  de  5  nivel 2.
##  Iteración  5  de  5  nivel 2.
## Iteración  2  de  5  nivel 1.
##  Iteración  1  de  5  nivel 2.
##  Iteración  2  de  5  nivel 2.
##  Iteración  3  de  5  nivel 2.
##  Iteración  4  de  5  nivel 2.
##  Iteración  5  de  5  nivel 2.
## Iteración  3  de  5  nivel 1.
##  Iteración  1  de  5  nivel 2.
##  Iteración  2  de  5  nivel 2.
##  Iteración  3  de  5  nivel 2.
##  Iteración  4  de  5  nivel 2.
##  Iteración  5  de  5  nivel 2.
## Iteración  4  de  5  nivel 1.
##  Iteración  1  de  5  nivel 2.
##  Iteración  2  de  5  nivel 2.
##  Iteración  3  de  5  nivel 2.
##  Iteración  4  de  5  nivel 2.
##  Iteración  5  de  5  nivel 2.
## Iteración  5  de  5  nivel 1.
##  Iteración  1  de  5  nivel 2.
##  Iteración  2  de  5  nivel 2.
##  Iteración  3  de  5  nivel 2.
##  Iteración  4  de  5  nivel 2.
##  Iteración  5  de  5  nivel 2.
```

Subsetting

Hay tres operadores para obtener subconjuntos de datos de nuestros objetos:

- [
- [[
- $

Los tres operadores funcionan de manera diferente para cada estrucutra de las ya definidas. La principal diferencia es la forma en la que regresan los datos, además de la manera en la que los utilizamos, y del tipo de datos con el que los utilizamos.

Vector

Con un vector, sólo funciona el primer operador, [, y su comportamiento depende del tipo de dato que utilicemos para realizar el subsetting. Tenemos los siguientes casos:

Enteros positivos

Al pasar valores enteros al operador, éste los interpreta como posiciones, por lo que devolvera los objetos que se encuentren en las posiciones del valor que usamos dentro de los corchetes. Recordemos que, a diferencia de otros lenguajes, en R el índice dentro de una estructura inicia en 1, no en 0. Así:

```
vector_subsetting <- c(1:90)
vector_subsetting[56]
```

```
## [1] 56
```

Si pasamos un vector, devuelve todos los objetos en las posiciones especificadas

```
vector_subsetting[c(1, 50, 89)]
```

```
## [1]  1 50 89
```

Pasar valores duplicados, duplica el valor devuelto:

```
vector_subsetting[c(1, 1, 2, 2)]
```

```
## [1] 1 1 2 2
```

Enteros negativos

El operador interpreta también como posiciones los enteros negativos, sin embargo, devuelve el vector original **sin** las entradas especificadas en el vector/número negativo que se le pase:

```
vector_subsetting[-c(1:85)]
```

```
## [1] 86 87 88 89 90
```

No es posile mezclar enteros positivos y negativos:

```
vector_subsetting[c(1, -2)]
```

```
## Error in vector_subsetting[c(1, -2)]: only 0's may be mixed with negative subscripts
```

Valores lógicos

Los valores lógicos son quizá los más útiles en combinación con los operadores para obtener subconjuntos de nuestros datos. Al pasar un vector de valores lógicos, obtendremos un subvector con los objetos en las posiciones donde el vector lógico es TRUE.

```
vector_subsetting_2 <- c(1:6)
vector_subsetting_2[c(T, F, T, F, T, F)]
```

```
## [1] 1 3 5
```

Si el vector lógico que se pasa al operador es de menor longitud, se recicla, es decir, se repite hasta igualar la longitud del vector sobre el que se está haciendo la operación.

```
vector_subsetting_2[c(T, F)]
```

```
## [1] 1 3 5
```

Como mencionamos, resultan de gran utilidad para obtener subconjutos de nuestras estructuras que cumplen cierto criterio, por ejemplo:

```
# Creamos un vector lógico:
vec_log <- vector_subsetting %% 2 == 0 # %% es la función modulo
```

```
vec_log
```

```
##  [1] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE
## [12]  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE
## [23] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE
## [34]  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE
## [45] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE
## [56]  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE
## [67] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE
## [78]  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE
## [89] FALSE  TRUE
```

```
# Lo usamos para obtener un subconjuto de los números pares del vector_subsetting
vector_subsetting[vec_log]
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46
## [24] 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90
```

Si el vector tiene nombres, podemos usarlos para obtener subconjuntos:

```
vector_con_nombres_1["b"]
```

```
## b
## 2
```

```
vector_con_nombres_1[c("a", "b")]
```

```
## a b
## 1 2
```

Lista

En una lista se aplican los operadores de la misma forma en la que se aplican con un vector atómico. Sin embargo, en una lista sí es posible utilizar los otros dos operadores, $ y [[. Cada uno tiene un comportamiento diferente en la lista.

[siempre devolvera una lista:

```
str(lista_con_nombres[1])
```

```
## List of 1
##  $ VecNum: int [1:5] 1 2 3 4 5
```

[[y $ devuelven el objeto mismo en la posición o con el nombre que se use con el operador:

```
str(lista_con_nombres[[1]])
```

```
##  int [1:5] 1 2 3 4 5
```

```
lista_con_nombres$VecNum
```

```
## [1] 1 2 3 4 5
```

Matrices y Data Frames

En las matrices y data frames, podemos utilizar un sólo vector, dos vectores, uno para cada dimensión, o incluso sacar provecho de los nombres, si los hay.

```
matriz_1
```

```
##   A B C
## a 1 4 7
## b 2 5 8
## c 3 6 9
```

```
matriz_1[1, 3]
```

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

```
matriz_1[c(1, 2), c("A", "C")]
```

```
##   A C
## a 1 7
## b 2 8
```

```
matriz_1[c(F, T, F), c(F, F, T)]
```

```
## [1] 8
```

En los data frames se pueden obtener subconjuntos de la misma forma, pero se obtiene más provecho con los otros dos operadores.

```
head(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
```

```
iris$Sepal.Length
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4
##  [18] 5.1 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5
##  [35] 4.9 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0
```

```
##  [52] 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8
##  [69] 6.2 5.6 5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4
##  [86] 6.0 6.7 6.3 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8
## [103] 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7
## [120] 6.0 6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7
## [137] 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8 6.7 6.7 6.3 6.5 6.2 5.9
```

```
iris[1]
```

```
##     Sepal.Length
## 1            5.1
## 2            4.9
## 3            4.7
## 4            4.6
## 5            5.0
## 6            5.4
## 7            4.6
## 8            5.0
## 9            4.4
## 10           4.9
## 11           5.4
## 12           4.8
## 13           4.8
## 14           4.3
## 15           5.8
## 16           5.7
## 17           5.4
## 18           5.1
## 19           5.7
## 20           5.1
## 21           5.4
## 22           5.1
## 23           4.6
## 24           5.1
## 25           4.8
## 26           5.0
## 27           5.0
## 28           5.2
## 29           5.2
## 30           4.7
## 31           4.8
## 32           5.4
## 33           5.2
## 34           5.5
## 35           4.9
## 36           5.0
## 37           5.5
## 38           4.9
## 39           4.4
## 40           5.1
## 41           5.0
## 42           4.5
## 43           4.4
## 44           5.0
## 45           5.1
```

```
## 46          4.8
## 47          5.1
## 48          4.6
## 49          5.3
## 50          5.0
## 51          7.0
## 52          6.4
## 53          6.9
## 54          5.5
## 55          6.5
## 56          5.7
## 57          6.3
## 58          4.9
## 59          6.6
## 60          5.2
## 61          5.0
## 62          5.9
## 63          6.0
## 64          6.1
## 65          5.6
## 66          6.7
## 67          5.6
## 68          5.8
## 69          6.2
## 70          5.6
## 71          5.9
## 72          6.1
## 73          6.3
## 74          6.1
## 75          6.4
## 76          6.6
## 77          6.8
## 78          6.7
## 79          6.0
## 80          5.7
## 81          5.5
## 82          5.5
## 83          5.8
## 84          6.0
## 85          5.4
## 86          6.0
## 87          6.7
## 88          6.3
## 89          5.6
## 90          5.5
## 91          5.5
## 92          6.1
## 93          5.8
## 94          5.0
## 95          5.6
## 96          5.7
## 97          5.7
## 98          6.2
## 99          5.1
```

```
## 100          5.7
## 101          6.3
## 102          5.8
## 103          7.1
## 104          6.3
## 105          6.5
## 106          7.6
## 107          4.9
## 108          7.3
## 109          6.7
## 110          7.2
## 111          6.5
## 112          6.4
## 113          6.8
## 114          5.7
## 115          5.8
## 116          6.4
## 117          6.5
## 118          7.7
## 119          7.7
## 120          6.0
## 121          6.9
## 122          5.6
## 123          7.7
## 124          6.3
## 125          6.7
## 126          7.2
## 127          6.2
## 128          6.1
## 129          6.4
## 130          7.2
## 131          7.4
## 132          7.9
## 133          6.4
## 134          6.3
## 135          6.1
## 136          7.7
## 137          6.3
## 138          6.4
## 139          6.0
## 140          6.9
## 141          6.7
## 142          6.9
## 143          5.8
## 144          6.8
## 145          6.7
## 146          6.7
## 147          6.3
## 148          6.5
## 149          6.2
## 150          5.9
```

```
iris[[1]]
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4
```

	Simplifying	Preserving
Vector	`x[[1]]`	`x[1]`
List	`x[[1]]`	`x[1]`
Factor	`x[1:4, drop = T]`	`x[1:4]`
Array	`x[1, ]` **or** `x[, 1]`	`x[1, , drop = F]` **or** `x[, 1, drop = F]`
Data frame	`x[, 1]` **or** `x[[1]]`	`x[, 1, drop = F]` **or** `x[1]`

Figure 2: Características de los operadores de subsetting

```
##  [18] 5.1 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5
##  [35] 4.9 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0
##  [52] 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8
##  [69] 6.2 5.6 5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4
##  [86] 6.0 6.7 6.3 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8
## [103] 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7
## [120] 6.0 6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7
## [137] 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8 6.7 6.7 6.3 6.5 6.2 5.9
```

Como vemos, los resultados de dos de los operadores difieren del otro.

Subsetting con preservación y sin preservación

Los resultados de los tres operadores devuelven los datos en formas/estructuras distintas, por lo que se definen dos tipos de subsetting: con preservación y sin preservación. Como su nombre lo dice, el subsetting con preservación devuelve un objeto con las mismas características del objeto sobre el que se aplica el operador con preservación, mientras que el subsetting sin preservación devuelve un objeto con características distintos al objeto originial. La siguiente tabla también sale de (Wickham 2015):

Así, veamos las diferencias:

```
str(lista_con_nombres_2[1])
```

```
## List of 1
##  $ VecNum: int [1:5] 1 2 3 4 5
```

```
str(lista_con_nombres_2[[1]])
```

```
##  int [1:5] 1 2 3 4 5
```

En un data frame:

```
str(iris[1])
```

```
## 'data.frame':    150 obs. of  1 variable:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
```

```
str(iris[[1]])
```

```
##  num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
```

```r
str(iris$Sepal.Length)
```

```
##  num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
```

Es importante tener en cuenta lo anterior cuando queramos hacer operaciones con objetos que extraigamos de algun otra estrucutra, pues podemos terminar con resultados inesperados o tener un error de ejecución.

Estructuras de control

Al igual que en el resto de los lenguajes de programación, en R podemos construir estructuras de control que nos ayudarán a repetir instrucciones o a controlar en qué momento queremos detener la ejecución de nuestras funciones, etc.

For

```r
for(i in 1:10) {
  print(i)
}
```

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

If

```r
int <- 15
for (i in 1:20) {
  if (i == int) {
    cat("Esta iteración es especial\n")
  } else {
    cat("Iteracion ", i, " de 100\n")
  }
}
```

```
## Iteracion  1  de 100
## Iteracion  2  de 100
## Iteracion  3  de 100
## Iteracion  4  de 100
## Iteracion  5  de 100
## Iteracion  6  de 100
## Iteracion  7  de 100
## Iteracion  8  de 100
## Iteracion  9  de 100
```

```
## Iteracion  10  de 100
## Iteracion  11  de 100
## Iteracion  12  de 100
## Iteracion  13  de 100
## Iteracion  14  de 100
## Esta iteración es especial
## Iteracion  16  de 100
## Iteracion  17  de 100
## Iteracion  18  de 100
## Iteracion  19  de 100
## Iteracion  20  de 100
```

While

```
int <- 15
cont <- 1
while(cont <= int) {
  if (cont == int) {
    cat("Llegamos al valor deseado")
    break
  } else {
    cat("El control es ", cont, "\n")
    cont <- cont + 1
  }
}
```

```
## El control es  1
## El control es  2
## El control es  3
## El control es  4
## El control es  5
## El control es  6
## El control es  7
## El control es  8
## El control es  9
## El control es  10
## El control es  11
## El control es  12
## El control es  13
## El control es  14
## Llegamos al valor deseado
```

Vemos que dentro del if en el ejemplo anterior tenemos la función "break", en un ciclo while es importante tener una forma de detener un ciclo, de lo contrario se ejecutará de manera indefinidad y hará que nuestro programa falle.

Una probada del tidyverse: *dplyr*

El paquete *dplyr* presenta un nuevo paradigma de manipulación de datos en R, basándose en 5 verbos (funciones) que, al combinarlos con un gran número de funciones auxiliares nos brindan un framework sumamente flexible. Dichos verbos son: select, filter, mutate, arrange y summarize. Al combinar la

flexibilidad de dplyr con otros paquetes presentes en el tidyverse (ver (Wickham and Grolemund 2017) para más detalle del tidyverse), obtenemos un conjunto de herramientas que convierten a R en una herramienta de gran poder para el análisis de datos.

Select

La función select nos ayuda a obtener un subconjuno de nuestros datos que contenga sólo las columnas que seleccionamos con la función (o sin ellas, si las precedemos por un guión).

Usando el data frame *iris* para ejemplificar:

```
head(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
```

```
iris %>% select(Sepal.Length, Sepal.Width, Species)
```

```
##     Sepal.Length Sepal.Width    Species
## 1            5.1         3.5     setosa
## 2            4.9         3.0     setosa
## 3            4.7         3.2     setosa
## 4            4.6         3.1     setosa
## 5            5.0         3.6     setosa
## 6            5.4         3.9     setosa
## 7            4.6         3.4     setosa
## 8            5.0         3.4     setosa
## 9            4.4         2.9     setosa
## 10           4.9         3.1     setosa
## 11           5.4         3.7     setosa
## 12           4.8         3.4     setosa
## 13           4.8         3.0     setosa
## 14           4.3         3.0     setosa
## 15           5.8         4.0     setosa
## 16           5.7         4.4     setosa
## 17           5.4         3.9     setosa
## 18           5.1         3.5     setosa
## 19           5.7         3.8     setosa
## 20           5.1         3.8     setosa
## 21           5.4         3.4     setosa
## 22           5.1         3.7     setosa
## 23           4.6         3.6     setosa
## 24           5.1         3.3     setosa
## 25           4.8         3.4     setosa
## 26           5.0         3.0     setosa
## 27           5.0         3.4     setosa
## 28           5.2         3.5     setosa
## 29           5.2         3.4     setosa
## 30           4.7         3.2     setosa
## 31           4.8         3.1     setosa
```

```
## 32          5.4         3.4     setosa
## 33          5.2         4.1     setosa
## 34          5.5         4.2     setosa
## 35          4.9         3.1     setosa
## 36          5.0         3.2     setosa
## 37          5.5         3.5     setosa
## 38          4.9         3.6     setosa
## 39          4.4         3.0     setosa
## 40          5.1         3.4     setosa
## 41          5.0         3.5     setosa
## 42          4.5         2.3     setosa
## 43          4.4         3.2     setosa
## 44          5.0         3.5     setosa
## 45          5.1         3.8     setosa
## 46          4.8         3.0     setosa
## 47          5.1         3.8     setosa
## 48          4.6         3.2     setosa
## 49          5.3         3.7     setosa
## 50          5.0         3.3     setosa
## 51          7.0         3.2 versicolor
## 52          6.4         3.2 versicolor
## 53          6.9         3.1 versicolor
## 54          5.5         2.3 versicolor
## 55          6.5         2.8 versicolor
## 56          5.7         2.8 versicolor
## 57          6.3         3.3 versicolor
## 58          4.9         2.4 versicolor
## 59          6.6         2.9 versicolor
## 60          5.2         2.7 versicolor
## 61          5.0         2.0 versicolor
## 62          5.9         3.0 versicolor
## 63          6.0         2.2 versicolor
## 64          6.1         2.9 versicolor
## 65          5.6         2.9 versicolor
## 66          6.7         3.1 versicolor
## 67          5.6         3.0 versicolor
## 68          5.8         2.7 versicolor
## 69          6.2         2.2 versicolor
## 70          5.6         2.5 versicolor
## 71          5.9         3.2 versicolor
## 72          6.1         2.8 versicolor
## 73          6.3         2.5 versicolor
## 74          6.1         2.8 versicolor
## 75          6.4         2.9 versicolor
## 76          6.6         3.0 versicolor
## 77          6.8         2.8 versicolor
## 78          6.7         3.0 versicolor
## 79          6.0         2.9 versicolor
## 80          5.7         2.6 versicolor
## 81          5.5         2.4 versicolor
## 82          5.5         2.4 versicolor
## 83          5.8         2.7 versicolor
## 84          6.0         2.7 versicolor
## 85          5.4         3.0 versicolor
```

```
## 86          6.0         3.4 versicolor
## 87          6.7         3.1 versicolor
## 88          6.3         2.3 versicolor
## 89          5.6         3.0 versicolor
## 90          5.5         2.5 versicolor
## 91          5.5         2.6 versicolor
## 92          6.1         3.0 versicolor
## 93          5.8         2.6 versicolor
## 94          5.0         2.3 versicolor
## 95          5.6         2.7 versicolor
## 96          5.7         3.0 versicolor
## 97          5.7         2.9 versicolor
## 98          6.2         2.9 versicolor
## 99          5.1         2.5 versicolor
## 100         5.7         2.8 versicolor
## 101         6.3         3.3  virginica
## 102         5.8         2.7  virginica
## 103         7.1         3.0  virginica
## 104         6.3         2.9  virginica
## 105         6.5         3.0  virginica
## 106         7.6         3.0  virginica
## 107         4.9         2.5  virginica
## 108         7.3         2.9  virginica
## 109         6.7         2.5  virginica
## 110         7.2         3.6  virginica
## 111         6.5         3.2  virginica
## 112         6.4         2.7  virginica
## 113         6.8         3.0  virginica
## 114         5.7         2.5  virginica
## 115         5.8         2.8  virginica
## 116         6.4         3.2  virginica
## 117         6.5         3.0  virginica
## 118         7.7         3.8  virginica
## 119         7.7         2.6  virginica
## 120         6.0         2.2  virginica
## 121         6.9         3.2  virginica
## 122         5.6         2.8  virginica
## 123         7.7         2.8  virginica
## 124         6.3         2.7  virginica
## 125         6.7         3.3  virginica
## 126         7.2         3.2  virginica
## 127         6.2         2.8  virginica
## 128         6.1         3.0  virginica
## 129         6.4         2.8  virginica
## 130         7.2         3.0  virginica
## 131         7.4         2.8  virginica
## 132         7.9         3.8  virginica
## 133         6.4         2.8  virginica
## 134         6.3         2.8  virginica
## 135         6.1         2.6  virginica
## 136         7.7         3.0  virginica
## 137         6.3         3.4  virginica
## 138         6.4         3.1  virginica
## 139         6.0         3.0  virginica
```

```
## 140          6.9         3.1  virginica
## 141          6.7         3.1  virginica
## 142          6.9         3.1  virginica
## 143          5.8         2.7  virginica
## 144          6.8         3.2  virginica
## 145          6.7         3.3  virginica
## 146          6.7         3.0  virginica
## 147          6.3         2.5  virginica
## 148          6.5         3.0  virginica
## 149          6.2         3.4  virginica
## 150          5.9         3.0  virginica
```

```
iris %>% select(-Petal.Length, -Petal.Width)
```

```
##     Sepal.Length Sepal.Width    Species
## 1            5.1         3.5     setosa
## 2            4.9         3.0     setosa
## 3            4.7         3.2     setosa
## 4            4.6         3.1     setosa
## 5            5.0         3.6     setosa
## 6            5.4         3.9     setosa
## 7            4.6         3.4     setosa
## 8            5.0         3.4     setosa
## 9            4.4         2.9     setosa
## 10           4.9         3.1     setosa
## 11           5.4         3.7     setosa
## 12           4.8         3.4     setosa
## 13           4.8         3.0     setosa
## 14           4.3         3.0     setosa
## 15           5.8         4.0     setosa
## 16           5.7         4.4     setosa
## 17           5.4         3.9     setosa
## 18           5.1         3.5     setosa
## 19           5.7         3.8     setosa
## 20           5.1         3.8     setosa
## 21           5.4         3.4     setosa
## 22           5.1         3.7     setosa
## 23           4.6         3.6     setosa
## 24           5.1         3.3     setosa
## 25           4.8         3.4     setosa
## 26           5.0         3.0     setosa
## 27           5.0         3.4     setosa
## 28           5.2         3.5     setosa
## 29           5.2         3.4     setosa
## 30           4.7         3.2     setosa
## 31           4.8         3.1     setosa
## 32           5.4         3.4     setosa
## 33           5.2         4.1     setosa
## 34           5.5         4.2     setosa
## 35           4.9         3.1     setosa
## 36           5.0         3.2     setosa
## 37           5.5         3.5     setosa
## 38           4.9         3.6     setosa
## 39           4.4         3.0     setosa
## 40           5.1         3.4     setosa
```

```
## 41          5.0         3.5     setosa
## 42          4.5         2.3     setosa
## 43          4.4         3.2     setosa
## 44          5.0         3.5     setosa
## 45          5.1         3.8     setosa
## 46          4.8         3.0     setosa
## 47          5.1         3.8     setosa
## 48          4.6         3.2     setosa
## 49          5.3         3.7     setosa
## 50          5.0         3.3     setosa
## 51          7.0         3.2 versicolor
## 52          6.4         3.2 versicolor
## 53          6.9         3.1 versicolor
## 54          5.5         2.3 versicolor
## 55          6.5         2.8 versicolor
## 56          5.7         2.8 versicolor
## 57          6.3         3.3 versicolor
## 58          4.9         2.4 versicolor
## 59          6.6         2.9 versicolor
## 60          5.2         2.7 versicolor
## 61          5.0         2.0 versicolor
## 62          5.9         3.0 versicolor
## 63          6.0         2.2 versicolor
## 64          6.1         2.9 versicolor
## 65          5.6         2.9 versicolor
## 66          6.7         3.1 versicolor
## 67          5.6         3.0 versicolor
## 68          5.8         2.7 versicolor
## 69          6.2         2.2 versicolor
## 70          5.6         2.5 versicolor
## 71          5.9         3.2 versicolor
## 72          6.1         2.8 versicolor
## 73          6.3         2.5 versicolor
## 74          6.1         2.8 versicolor
## 75          6.4         2.9 versicolor
## 76          6.6         3.0 versicolor
## 77          6.8         2.8 versicolor
## 78          6.7         3.0 versicolor
## 79          6.0         2.9 versicolor
## 80          5.7         2.6 versicolor
## 81          5.5         2.4 versicolor
## 82          5.5         2.4 versicolor
## 83          5.8         2.7 versicolor
## 84          6.0         2.7 versicolor
## 85          5.4         3.0 versicolor
## 86          6.0         3.4 versicolor
## 87          6.7         3.1 versicolor
## 88          6.3         2.3 versicolor
## 89          5.6         3.0 versicolor
## 90          5.5         2.5 versicolor
## 91          5.5         2.6 versicolor
## 92          6.1         3.0 versicolor
## 93          5.8         2.6 versicolor
## 94          5.0         2.3 versicolor
```

```
## 95          5.6         2.7 versicolor
## 96          5.7         3.0 versicolor
## 97          5.7         2.9 versicolor
## 98          6.2         2.9 versicolor
## 99          5.1         2.5 versicolor
## 100         5.7         2.8 versicolor
## 101         6.3         3.3  virginica
## 102         5.8         2.7  virginica
## 103         7.1         3.0  virginica
## 104         6.3         2.9  virginica
## 105         6.5         3.0  virginica
## 106         7.6         3.0  virginica
## 107         4.9         2.5  virginica
## 108         7.3         2.9  virginica
## 109         6.7         2.5  virginica
## 110         7.2         3.6  virginica
## 111         6.5         3.2  virginica
## 112         6.4         2.7  virginica
## 113         6.8         3.0  virginica
## 114         5.7         2.5  virginica
## 115         5.8         2.8  virginica
## 116         6.4         3.2  virginica
## 117         6.5         3.0  virginica
## 118         7.7         3.8  virginica
## 119         7.7         2.6  virginica
## 120         6.0         2.2  virginica
## 121         6.9         3.2  virginica
## 122         5.6         2.8  virginica
## 123         7.7         2.8  virginica
## 124         6.3         2.7  virginica
## 125         6.7         3.3  virginica
## 126         7.2         3.2  virginica
## 127         6.2         2.8  virginica
## 128         6.1         3.0  virginica
## 129         6.4         2.8  virginica
## 130         7.2         3.0  virginica
## 131         7.4         2.8  virginica
## 132         7.9         3.8  virginica
## 133         6.4         2.8  virginica
## 134         6.3         2.8  virginica
## 135         6.1         2.6  virginica
## 136         7.7         3.0  virginica
## 137         6.3         3.4  virginica
## 138         6.4         3.1  virginica
## 139         6.0         3.0  virginica
## 140         6.9         3.1  virginica
## 141         6.7         3.1  virginica
## 142         6.9         3.1  virginica
## 143         5.8         2.7  virginica
## 144         6.8         3.2  virginica
## 145         6.7         3.3  virginica
## 146         6.7         3.0  virginica
## 147         6.3         2.5  virginica
## 148         6.5         3.0  virginica
```

```
## 149          6.2         3.4  virginica
## 150          5.9         3.0  virginica
```

filter

La función filter nos ayuda a seleccionar un conjunto de renglones de un data frame, que cumplan algún criterio de interés.

```
iris %>% filter((Species == "setosa") & (Sepal.Length > 5))
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1           5.1         3.5          1.4         0.2  setosa
## 2           5.4         3.9          1.7         0.4  setosa
## 3           5.4         3.7          1.5         0.2  setosa
## 4           5.8         4.0          1.2         0.2  setosa
## 5           5.7         4.4          1.5         0.4  setosa
## 6           5.4         3.9          1.3         0.4  setosa
## 7           5.1         3.5          1.4         0.3  setosa
## 8           5.7         3.8          1.7         0.3  setosa
## 9           5.1         3.8          1.5         0.3  setosa
## 10          5.4         3.4          1.7         0.2  setosa
## 11          5.1         3.7          1.5         0.4  setosa
## 12          5.1         3.3          1.7         0.5  setosa
## 13          5.2         3.5          1.5         0.2  setosa
## 14          5.2         3.4          1.4         0.2  setosa
## 15          5.4         3.4          1.5         0.4  setosa
## 16          5.2         4.1          1.5         0.1  setosa
## 17          5.5         4.2          1.4         0.2  setosa
## 18          5.5         3.5          1.3         0.2  setosa
## 19          5.1         3.4          1.5         0.2  setosa
## 20          5.1         3.8          1.9         0.4  setosa
## 21          5.1         3.8          1.6         0.2  setosa
## 22          5.3         3.7          1.5         0.2  setosa
```

```
iris %>% filter((Species == "versicolor"))
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1           7.0         3.2          4.7         1.4 versicolor
## 2           6.4         3.2          4.5         1.5 versicolor
## 3           6.9         3.1          4.9         1.5 versicolor
## 4           5.5         2.3          4.0         1.3 versicolor
## 5           6.5         2.8          4.6         1.5 versicolor
## 6           5.7         2.8          4.5         1.3 versicolor
## 7           6.3         3.3          4.7         1.6 versicolor
## 8           4.9         2.4          3.3         1.0 versicolor
## 9           6.6         2.9          4.6         1.3 versicolor
## 10          5.2         2.7          3.9         1.4 versicolor
## 11          5.0         2.0          3.5         1.0 versicolor
## 12          5.9         3.0          4.2         1.5 versicolor
## 13          6.0         2.2          4.0         1.0 versicolor
## 14          6.1         2.9          4.7         1.4 versicolor
## 15          5.6         2.9          3.6         1.3 versicolor
## 16          6.7         3.1          4.4         1.4 versicolor
## 17          5.6         3.0          4.5         1.5 versicolor
## 18          5.8         2.7          4.1         1.0 versicolor
```

```
## 19          6.2         2.2          4.5         1.5 versicolor
## 20          5.6         2.5          3.9         1.1 versicolor
## 21          5.9         3.2          4.8         1.8 versicolor
## 22          6.1         2.8          4.0         1.3 versicolor
## 23          6.3         2.5          4.9         1.5 versicolor
## 24          6.1         2.8          4.7         1.2 versicolor
## 25          6.4         2.9          4.3         1.3 versicolor
## 26          6.6         3.0          4.4         1.4 versicolor
## 27          6.8         2.8          4.8         1.4 versicolor
## 28          6.7         3.0          5.0         1.7 versicolor
## 29          6.0         2.9          4.5         1.5 versicolor
## 30          5.7         2.6          3.5         1.0 versicolor
## 31          5.5         2.4          3.8         1.1 versicolor
## 32          5.5         2.4          3.7         1.0 versicolor
## 33          5.8         2.7          3.9         1.2 versicolor
## 34          6.0         2.7          5.1         1.6 versicolor
## 35          5.4         3.0          4.5         1.5 versicolor
## 36          6.0         3.4          4.5         1.6 versicolor
## 37          6.7         3.1          4.7         1.5 versicolor
## 38          6.3         2.3          4.4         1.3 versicolor
## 39          5.6         3.0          4.1         1.3 versicolor
## 40          5.5         2.5          4.0         1.3 versicolor
## 41          5.5         2.6          4.4         1.2 versicolor
## 42          6.1         3.0          4.6         1.4 versicolor
## 43          5.8         2.6          4.0         1.2 versicolor
## 44          5.0         2.3          3.3         1.0 versicolor
## 45          5.6         2.7          4.2         1.3 versicolor
## 46          5.7         3.0          4.2         1.2 versicolor
## 47          5.7         2.9          4.2         1.3 versicolor
## 48          6.2         2.9          4.3         1.3 versicolor
## 49          5.1         2.5          3.0         1.1 versicolor
## 50          5.7         2.8          4.1         1.3 versicolor
```

mutate

Con la función mutate somos capaces de crear nuevas columnas en un data frame a partir de los valores de otras columnas existentes en el mismo, **para cada renglón/observación**.

```
iris %>% mutate(Area = as.numeric(Sepal.Length)*as.numeric(Sepal.Width))
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species  Area
## 1            5.1         3.5          1.4         0.2     setosa 17.85
## 2            4.9         3.0          1.4         0.2     setosa 14.70
## 3            4.7         3.2          1.3         0.2     setosa 15.04
## 4            4.6         3.1          1.5         0.2     setosa 14.26
## 5            5.0         3.6          1.4         0.2     setosa 18.00
## 6            5.4         3.9          1.7         0.4     setosa 21.06
## 7            4.6         3.4          1.4         0.3     setosa 15.64
## 8            5.0         3.4          1.5         0.2     setosa 17.00
## 9            4.4         2.9          1.4         0.2     setosa 12.76
## 10           4.9         3.1          1.5         0.1     setosa 15.19
## 11           5.4         3.7          1.5         0.2     setosa 19.98
## 12           4.8         3.4          1.6         0.2     setosa 16.32
## 13           4.8         3.0          1.4         0.1     setosa 14.40
```

```
## 14          4.3         3.0          1.1         0.1     setosa 12.90
## 15          5.8         4.0          1.2         0.2     setosa 23.20
## 16          5.7         4.4          1.5         0.4     setosa 25.08
## 17          5.4         3.9          1.3         0.4     setosa 21.06
## 18          5.1         3.5          1.4         0.3     setosa 17.85
## 19          5.7         3.8          1.7         0.3     setosa 21.66
## 20          5.1         3.8          1.5         0.3     setosa 19.38
## 21          5.4         3.4          1.7         0.2     setosa 18.36
## 22          5.1         3.7          1.5         0.4     setosa 18.87
## 23          4.6         3.6          1.0         0.2     setosa 16.56
## 24          5.1         3.3          1.7         0.5     setosa 16.83
## 25          4.8         3.4          1.9         0.2     setosa 16.32
## 26          5.0         3.0          1.6         0.2     setosa 15.00
## 27          5.0         3.4          1.6         0.4     setosa 17.00
## 28          5.2         3.5          1.5         0.2     setosa 18.20
## 29          5.2         3.4          1.4         0.2     setosa 17.68
## 30          4.7         3.2          1.6         0.2     setosa 15.04
## 31          4.8         3.1          1.6         0.2     setosa 14.88
## 32          5.4         3.4          1.5         0.4     setosa 18.36
## 33          5.2         4.1          1.5         0.1     setosa 21.32
## 34          5.5         4.2          1.4         0.2     setosa 23.10
## 35          4.9         3.1          1.5         0.2     setosa 15.19
## 36          5.0         3.2          1.2         0.2     setosa 16.00
## 37          5.5         3.5          1.3         0.2     setosa 19.25
## 38          4.9         3.6          1.4         0.1     setosa 17.64
## 39          4.4         3.0          1.3         0.2     setosa 13.20
## 40          5.1         3.4          1.5         0.2     setosa 17.34
## 41          5.0         3.5          1.3         0.3     setosa 17.50
## 42          4.5         2.3          1.3         0.3     setosa 10.35
## 43          4.4         3.2          1.3         0.2     setosa 14.08
## 44          5.0         3.5          1.6         0.6     setosa 17.50
## 45          5.1         3.8          1.9         0.4     setosa 19.38
## 46          4.8         3.0          1.4         0.3     setosa 14.40
## 47          5.1         3.8          1.6         0.2     setosa 19.38
## 48          4.6         3.2          1.4         0.2     setosa 14.72
## 49          5.3         3.7          1.5         0.2     setosa 19.61
## 50          5.0         3.3          1.4         0.2     setosa 16.50
## 51          7.0         3.2          4.7         1.4 versicolor 22.40
## 52          6.4         3.2          4.5         1.5 versicolor 20.48
## 53          6.9         3.1          4.9         1.5 versicolor 21.39
## 54          5.5         2.3          4.0         1.3 versicolor 12.65
## 55          6.5         2.8          4.6         1.5 versicolor 18.20
## 56          5.7         2.8          4.5         1.3 versicolor 15.96
## 57          6.3         3.3          4.7         1.6 versicolor 20.79
## 58          4.9         2.4          3.3         1.0 versicolor 11.76
## 59          6.6         2.9          4.6         1.3 versicolor 19.14
## 60          5.2         2.7          3.9         1.4 versicolor 14.04
## 61          5.0         2.0          3.5         1.0 versicolor 10.00
## 62          5.9         3.0          4.2         1.5 versicolor 17.70
## 63          6.0         2.2          4.0         1.0 versicolor 13.20
## 64          6.1         2.9          4.7         1.4 versicolor 17.69
## 65          5.6         2.9          3.6         1.3 versicolor 16.24
## 66          6.7         3.1          4.4         1.4 versicolor 20.77
## 67          5.6         3.0          4.5         1.5 versicolor 16.80
```

```
## 68          5.8         2.7          4.1         1.0 versicolor 15.66
## 69          6.2         2.2          4.5         1.5 versicolor 13.64
## 70          5.6         2.5          3.9         1.1 versicolor 14.00
## 71          5.9         3.2          4.8         1.8 versicolor 18.88
## 72          6.1         2.8          4.0         1.3 versicolor 17.08
## 73          6.3         2.5          4.9         1.5 versicolor 15.75
## 74          6.1         2.8          4.7         1.2 versicolor 17.08
## 75          6.4         2.9          4.3         1.3 versicolor 18.56
## 76          6.6         3.0          4.4         1.4 versicolor 19.80
## 77          6.8         2.8          4.8         1.4 versicolor 19.04
## 78          6.7         3.0          5.0         1.7 versicolor 20.10
## 79          6.0         2.9          4.5         1.5 versicolor 17.40
## 80          5.7         2.6          3.5         1.0 versicolor 14.82
## 81          5.5         2.4          3.8         1.1 versicolor 13.20
## 82          5.5         2.4          3.7         1.0 versicolor 13.20
## 83          5.8         2.7          3.9         1.2 versicolor 15.66
## 84          6.0         2.7          5.1         1.6 versicolor 16.20
## 85          5.4         3.0          4.5         1.5 versicolor 16.20
## 86          6.0         3.4          4.5         1.6 versicolor 20.40
## 87          6.7         3.1          4.7         1.5 versicolor 20.77
## 88          6.3         2.3          4.4         1.3 versicolor 14.49
## 89          5.6         3.0          4.1         1.3 versicolor 16.80
## 90          5.5         2.5          4.0         1.3 versicolor 13.75
## 91          5.5         2.6          4.4         1.2 versicolor 14.30
## 92          6.1         3.0          4.6         1.4 versicolor 18.30
## 93          5.8         2.6          4.0         1.2 versicolor 15.08
## 94          5.0         2.3          3.3         1.0 versicolor 11.50
## 95          5.6         2.7          4.2         1.3 versicolor 15.12
## 96          5.7         3.0          4.2         1.2 versicolor 17.10
## 97          5.7         2.9          4.2         1.3 versicolor 16.53
## 98          6.2         2.9          4.3         1.3 versicolor 17.98
## 99          5.1         2.5          3.0         1.1 versicolor 12.75
## 100         5.7         2.8          4.1         1.3 versicolor 15.96
## 101         6.3         3.3          6.0         2.5  virginica 20.79
## 102         5.8         2.7          5.1         1.9  virginica 15.66
## 103         7.1         3.0          5.9         2.1  virginica 21.30
## 104         6.3         2.9          5.6         1.8  virginica 18.27
## 105         6.5         3.0          5.8         2.2  virginica 19.50
## 106         7.6         3.0          6.6         2.1  virginica 22.80
## 107         4.9         2.5          4.5         1.7  virginica 12.25
## 108         7.3         2.9          6.3         1.8  virginica 21.17
## 109         6.7         2.5          5.8         1.8  virginica 16.75
## 110         7.2         3.6          6.1         2.5  virginica 25.92
## 111         6.5         3.2          5.1         2.0  virginica 20.80
## 112         6.4         2.7          5.3         1.9  virginica 17.28
## 113         6.8         3.0          5.5         2.1  virginica 20.40
## 114         5.7         2.5          5.0         2.0  virginica 14.25
## 115         5.8         2.8          5.1         2.4  virginica 16.24
## 116         6.4         3.2          5.3         2.3  virginica 20.48
## 117         6.5         3.0          5.5         1.8  virginica 19.50
## 118         7.7         3.8          6.7         2.2  virginica 29.26
## 119         7.7         2.6          6.9         2.3  virginica 20.02
## 120         6.0         2.2          5.0         1.5  virginica 13.20
## 121         6.9         3.2          5.7         2.3  virginica 22.08
```

```
## 122          5.6         2.8          4.9         2.0  virginica 15.68
## 123          7.7         2.8          6.7         2.0  virginica 21.56
## 124          6.3         2.7          4.9         1.8  virginica 17.01
## 125          6.7         3.3          5.7         2.1  virginica 22.11
## 126          7.2         3.2          6.0         1.8  virginica 23.04
## 127          6.2         2.8          4.8         1.8  virginica 17.36
## 128          6.1         3.0          4.9         1.8  virginica 18.30
## 129          6.4         2.8          5.6         2.1  virginica 17.92
## 130          7.2         3.0          5.8         1.6  virginica 21.60
## 131          7.4         2.8          6.1         1.9  virginica 20.72
## 132          7.9         3.8          6.4         2.0  virginica 30.02
## 133          6.4         2.8          5.6         2.2  virginica 17.92
## 134          6.3         2.8          5.1         1.5  virginica 17.64
## 135          6.1         2.6          5.6         1.4  virginica 15.86
## 136          7.7         3.0          6.1         2.3  virginica 23.10
## 137          6.3         3.4          5.6         2.4  virginica 21.42
## 138          6.4         3.1          5.5         1.8  virginica 19.84
## 139          6.0         3.0          4.8         1.8  virginica 18.00
## 140          6.9         3.1          5.4         2.1  virginica 21.39
## 141          6.7         3.1          5.6         2.4  virginica 20.77
## 142          6.9         3.1          5.1         2.3  virginica 21.39
## 143          5.8         2.7          5.1         1.9  virginica 15.66
## 144          6.8         3.2          5.9         2.3  virginica 21.76
## 145          6.7         3.3          5.7         2.5  virginica 22.11
## 146          6.7         3.0          5.2         2.3  virginica 20.10
## 147          6.3         2.5          5.0         1.9  virginica 15.75
## 148          6.5         3.0          5.2         2.0  virginica 19.50
## 149          6.2         3.4          5.4         2.3  virginica 21.08
## 150          5.9         3.0          5.1         1.8  virginica 17.70
```

arrange

Con arrange, podemos ordenar las observaciones de un data frame de acuerdo a los valores de una o más columnas, sin importar si son de distintos tipos de datos.

```
iris$Species <- as.character(iris$Species)
```

```
iris %>% arrange(desc(Species), Sepal.Length)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1            4.9         2.5          4.5         1.7  virginica
## 2            5.6         2.8          4.9         2.0  virginica
## 3            5.7         2.5          5.0         2.0  virginica
## 4            5.8         2.7          5.1         1.9  virginica
## 5            5.8         2.8          5.1         2.4  virginica
## 6            5.8         2.7          5.1         1.9  virginica
## 7            5.9         3.0          5.1         1.8  virginica
## 8            6.0         2.2          5.0         1.5  virginica
## 9            6.0         3.0          4.8         1.8  virginica
## 10           6.1         3.0          4.9         1.8  virginica
## 11           6.1         2.6          5.6         1.4  virginica
## 12           6.2         2.8          4.8         1.8  virginica
## 13           6.2         3.4          5.4         2.3  virginica
## 14           6.3         3.3          6.0         2.5  virginica
```

```
## 15          6.3         2.9          5.6         1.8  virginica
## 16          6.3         2.7          4.9         1.8  virginica
## 17          6.3         2.8          5.1         1.5  virginica
## 18          6.3         3.4          5.6         2.4  virginica
## 19          6.3         2.5          5.0         1.9  virginica
## 20          6.4         2.7          5.3         1.9  virginica
## 21          6.4         3.2          5.3         2.3  virginica
## 22          6.4         2.8          5.6         2.1  virginica
## 23          6.4         2.8          5.6         2.2  virginica
## 24          6.4         3.1          5.5         1.8  virginica
## 25          6.5         3.0          5.8         2.2  virginica
## 26          6.5         3.2          5.1         2.0  virginica
## 27          6.5         3.0          5.5         1.8  virginica
## 28          6.5         3.0          5.2         2.0  virginica
## 29          6.7         2.5          5.8         1.8  virginica
## 30          6.7         3.3          5.7         2.1  virginica
## 31          6.7         3.1          5.6         2.4  virginica
## 32          6.7         3.3          5.7         2.5  virginica
## 33          6.7         3.0          5.2         2.3  virginica
## 34          6.8         3.0          5.5         2.1  virginica
## 35          6.8         3.2          5.9         2.3  virginica
## 36          6.9         3.2          5.7         2.3  virginica
## 37          6.9         3.1          5.4         2.1  virginica
## 38          6.9         3.1          5.1         2.3  virginica
## 39          7.1         3.0          5.9         2.1  virginica
## 40          7.2         3.6          6.1         2.5  virginica
## 41          7.2         3.2          6.0         1.8  virginica
## 42          7.2         3.0          5.8         1.6  virginica
## 43          7.3         2.9          6.3         1.8  virginica
## 44          7.4         2.8          6.1         1.9  virginica
## 45          7.6         3.0          6.6         2.1  virginica
## 46          7.7         3.8          6.7         2.2  virginica
## 47          7.7         2.6          6.9         2.3  virginica
## 48          7.7         2.8          6.7         2.0  virginica
## 49          7.7         3.0          6.1         2.3  virginica
## 50          7.9         3.8          6.4         2.0  virginica
## 51          4.9         2.4          3.3         1.0 versicolor
## 52          5.0         2.0          3.5         1.0 versicolor
## 53          5.0         2.3          3.3         1.0 versicolor
## 54          5.1         2.5          3.0         1.1 versicolor
## 55          5.2         2.7          3.9         1.4 versicolor
## 56          5.4         3.0          4.5         1.5 versicolor
## 57          5.5         2.3          4.0         1.3 versicolor
## 58          5.5         2.4          3.8         1.1 versicolor
## 59          5.5         2.4          3.7         1.0 versicolor
## 60          5.5         2.5          4.0         1.3 versicolor
## 61          5.5         2.6          4.4         1.2 versicolor
## 62          5.6         2.9          3.6         1.3 versicolor
## 63          5.6         3.0          4.5         1.5 versicolor
## 64          5.6         2.5          3.9         1.1 versicolor
## 65          5.6         3.0          4.1         1.3 versicolor
## 66          5.6         2.7          4.2         1.3 versicolor
## 67          5.7         2.8          4.5         1.3 versicolor
## 68          5.7         2.6          3.5         1.0 versicolor
```

```
## 69          5.7         3.0          4.2         1.2 versicolor
## 70          5.7         2.9          4.2         1.3 versicolor
## 71          5.7         2.8          4.1         1.3 versicolor
## 72          5.8         2.7          4.1         1.0 versicolor
## 73          5.8         2.7          3.9         1.2 versicolor
## 74          5.8         2.6          4.0         1.2 versicolor
## 75          5.9         3.0          4.2         1.5 versicolor
## 76          5.9         3.2          4.8         1.8 versicolor
## 77          6.0         2.2          4.0         1.0 versicolor
## 78          6.0         2.9          4.5         1.5 versicolor
## 79          6.0         2.7          5.1         1.6 versicolor
## 80          6.0         3.4          4.5         1.6 versicolor
## 81          6.1         2.9          4.7         1.4 versicolor
## 82          6.1         2.8          4.0         1.3 versicolor
## 83          6.1         2.8          4.7         1.2 versicolor
## 84          6.1         3.0          4.6         1.4 versicolor
## 85          6.2         2.2          4.5         1.5 versicolor
## 86          6.2         2.9          4.3         1.3 versicolor
## 87          6.3         3.3          4.7         1.6 versicolor
## 88          6.3         2.5          4.9         1.5 versicolor
## 89          6.3         2.3          4.4         1.3 versicolor
## 90          6.4         3.2          4.5         1.5 versicolor
## 91          6.4         2.9          4.3         1.3 versicolor
## 92          6.5         2.8          4.6         1.5 versicolor
## 93          6.6         2.9          4.6         1.3 versicolor
## 94          6.6         3.0          4.4         1.4 versicolor
## 95          6.7         3.1          4.4         1.4 versicolor
## 96          6.7         3.0          5.0         1.7 versicolor
## 97          6.7         3.1          4.7         1.5 versicolor
## 98          6.8         2.8          4.8         1.4 versicolor
## 99          6.9         3.1          4.9         1.5 versicolor
## 100         7.0         3.2          4.7         1.4 versicolor
## 101         4.3         3.0          1.1         0.1     setosa
## 102         4.4         2.9          1.4         0.2     setosa
## 103         4.4         3.0          1.3         0.2     setosa
## 104         4.4         3.2          1.3         0.2     setosa
## 105         4.5         2.3          1.3         0.3     setosa
## 106         4.6         3.1          1.5         0.2     setosa
## 107         4.6         3.4          1.4         0.3     setosa
## 108         4.6         3.6          1.0         0.2     setosa
## 109         4.6         3.2          1.4         0.2     setosa
## 110         4.7         3.2          1.3         0.2     setosa
## 111         4.7         3.2          1.6         0.2     setosa
## 112         4.8         3.4          1.6         0.2     setosa
## 113         4.8         3.0          1.4         0.1     setosa
## 114         4.8         3.4          1.9         0.2     setosa
## 115         4.8         3.1          1.6         0.2     setosa
## 116         4.8         3.0          1.4         0.3     setosa
## 117         4.9         3.0          1.4         0.2     setosa
## 118         4.9         3.1          1.5         0.1     setosa
## 119         4.9         3.1          1.5         0.2     setosa
## 120         4.9         3.6          1.4         0.1     setosa
## 121         5.0         3.6          1.4         0.2     setosa
## 122         5.0         3.4          1.5         0.2     setosa
```

```
## 123          5.0         3.0          1.6         0.2     setosa
## 124          5.0         3.4          1.6         0.4     setosa
## 125          5.0         3.2          1.2         0.2     setosa
## 126          5.0         3.5          1.3         0.3     setosa
## 127          5.0         3.5          1.6         0.6     setosa
## 128          5.0         3.3          1.4         0.2     setosa
## 129          5.1         3.5          1.4         0.2     setosa
## 130          5.1         3.5          1.4         0.3     setosa
## 131          5.1         3.8          1.5         0.3     setosa
## 132          5.1         3.7          1.5         0.4     setosa
## 133          5.1         3.3          1.7         0.5     setosa
## 134          5.1         3.4          1.5         0.2     setosa
## 135          5.1         3.8          1.9         0.4     setosa
## 136          5.1         3.8          1.6         0.2     setosa
## 137          5.2         3.5          1.5         0.2     setosa
## 138          5.2         3.4          1.4         0.2     setosa
## 139          5.2         4.1          1.5         0.1     setosa
## 140          5.3         3.7          1.5         0.2     setosa
## 141          5.4         3.9          1.7         0.4     setosa
## 142          5.4         3.7          1.5         0.2     setosa
## 143          5.4         3.9          1.3         0.4     setosa
## 144          5.4         3.4          1.7         0.2     setosa
## 145          5.4         3.4          1.5         0.4     setosa
## 146          5.5         4.2          1.4         0.2     setosa
## 147          5.5         3.5          1.3         0.2     setosa
## 148          5.7         4.4          1.5         0.4     setosa
## 149          5.7         3.8          1.7         0.3     setosa
## 150          5.8         4.0          1.2         0.2     setosa
```

summarize

La función summarize nos ayuda a obtener valores estadísticos de las columnas que forman un data frame. Por ejemplo, queremos los valores promedio de las cuatro columnas de iris por cada especie.

```
iris %>% group_by(Species) %>% summarise(PromSepal.Length = mean(Sepal.Length),
                                         PromSepal.Width = mean(Sepal.Width),
                                         PromPetal.Length = mean(Petal.Length),
                                         PromPetal.Width = mean(Petal.Width))
```

```
## # A tibble: 3 x 5
##   Speci~ PromSepal.Length PromSepal.Width PromPetal.Length PromPetal.Width
##   <chr>             <dbl>           <dbl>            <dbl>           <dbl>
## 1 setosa             5.01            3.43             1.46           0.246
## 2 versi~             5.94            2.77             4.26           1.33
## 3 virgi~             6.59            2.97             5.55           2.03
```

Referencias

Wickham, Hadley. 2015. *Advanced R*. Chapman & Hall/CRC.

Wickham, Hadley, and Garret Grolemund. 2017. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. 1st ed. O'Reilly Media, Inc.