S3 Extendida

Juan José Merino Zarco

28/6/2021

Dataframes

> Nota: cuando estamos visualizando dataframes en RStudio nos lo muestra por fragmentos, sin embargo, cuando compilamos a un pdf u otra salida el programa imprime todas las observaciones, por lo que si queremos compilar podemos usar el comando "head()" que solo visualiza las primeras observaciones. Alternativamente podemos usar el comando "tail()" que visualiza las ultimas observaciones.

```
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
```

```
tail(iris)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 145          6.7         3.3          5.7         2.5 virginica
## 146          6.7         3.0          5.2         2.3 virginica
## 147          6.3         2.5          5.0         1.9 virginica
## 148          6.5         3.0          5.2         2.0 virginica
## 149          6.2         3.4          5.4         2.3 virginica
## 150          5.9         3.0          5.1         1.8 virginica
```

Explorando un dataframe Revisamos las dimensiones del dataframe iris.

```
dim(iris)
```

```
## [1] 150   5
```

Revisamos el largo del dataframe, que corresponderá al numero de columnas que tenga.

```
length(iris)
```

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

Para obtener las estadísticas básicas usamos: summary()

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

Para revisar cuantas observaciones tiene, numero de variables, etc., usamos: str()

```
str(summary)
```

```
## function (object, ...)
```

Manipulando el dataframe Podemos realizar operaciones con un dataframe como lo hicimos con una matriz, sin embargo, este puede contener datos no numericos, por lo que al realizar la operacion con un dato del tipo texto inmediatamente arrojara *NA*

```
q <- iris+2
```

```
## Warning in Ops.factor(left, right): '+' not meaningful for factors
```

```
head(q)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          7.1         5.5          3.4         2.2      NA
## 2          6.9         5.0          3.4         2.2      NA
## 3          6.7         5.2          3.3         2.2      NA
## 4          6.6         5.1          3.5         2.2      NA
## 5          7.0         5.6          3.4         2.2      NA
## 6          7.4         5.9          3.7         2.4      NA
```

Subconjuntos de una estructura de datos

Indices

Un indice representa una posicion.

> Nota : Los indices en R empiezan en 1, lenguajes como Python empiezan en 0.

Vectores

```
color <- c("rojo", "azul", "verde", "amarillo", "morado")
color
```

```
## [1] "rojo"     "azul"     "verde"    "amarillo" "morado"
```

Verificamos el largo

```
length(color)
```

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

Buscamos extraer elementos especificos:

- Primer elemento

```
color[1]
```

```
## [1] "rojo"
```

- Quinto elemento

```
color[5]
```

```
## [1] "morado"
```

- Primeros 4 elementos

```
color[1:4]
```

```
## [1] "rojo"     "azul"     "verde"    "amarillo"
```

- 1 y 3 elemento

```
color[c(1,3)]
```

```
## [1] "rojo"  "verde"
```

- Eliminar el primer elemento

```
color[-1]
```

```
## [1] "azul"     "verde"    "amarillo" "morado"
```

- Eliminar el primer y tercer elemento

```
color[c(-1, -3)]
```

```
## [1] "azul"     "amarillo" "morado"
```

Dataframes

Usaremos el dataframe Iris.

```
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
```

Verificamos nuevamente su dimensión

```
dim(iris)
```

```
## [1] 150   5
```

Notemos que el primer dígito corresponde al numero de renglones y el segundo al numero de columnas, lo cual nos servirá para identificar la posición de los elementos tal cual lo haríamos con una matriz.

Ahora buscamos extraer:

- El elemento que esta en el renglón 1 y columna 3

```
iris[1,3]
```

```
## [1] 1.4
```

- Solo el renglón 1

```
iris[1,]
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
```

- Solo la columna 3

```
head(iris[,3])
```

```
## [1] 1.4 1.4 1.3 1.5 1.4 1.7
```

- Columnas 1 y 3

```
head(iris[c(1,3)])
```

```
##   Sepal.Length Petal.Length
## 1          5.1          1.4
## 2          4.9          1.4
## 3          4.7          1.3
## 4          4.6          1.5
## 5          5.0          1.4
## 6          5.4          1.7
```

```
head(iris[,c(1,3)])
```

```
##   Sepal.Length Petal.Length
## 1          5.1          1.4
## 2          4.9          1.4
## 3          4.7          1.3
## 4          4.6          1.5
## 5          5.0          1.4
## 6          5.4          1.7
```

- Renglón 1 y 3

```
iris[c(1,3),]
```

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

- Renglón 1 y 2, columna 3

```
iris[1:2,3]
```

```
## [1] 1.4 1.4
```

Extraer por nombre

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
head(iris["Sepal.Length"])
```

```
##   Sepal.Length
## 1          5.1
## 2          4.9
## 3          4.7
## 4          4.6
## 5          5.0
## 6          5.4
```

```
head(iris[c("Sepal.Length", "Species")])
```

```
##   Sepal.Length Species
## 1          5.1  setosa
## 2          4.9  setosa
## 3          4.7  setosa
## 4          4.6  setosa
## 5          5.0  setosa
## 6          5.4  setosa
```

Condicionales

Para utilizar condicionales usaremos la siguiente estructura:

> Data_frame[condicion, columnas_devueltas]

Uso de condicionales

Queremos obtener un subconjunto del dataframe iris, donde solo tengamos las observaciones en que la variable "Petal.Length" sea mayor o igual que 5.5.

Primero obtenemos la variable Petal.Length del dataframe iris

> dataframe$columna
>
> Nota2: Analogo al comando "head()" podemos usar "tail()" para visualizar las ultimas 6 observaciones.

```
tail(iris$Petal.Length)
```

```
## [1] 5.7 5.2 5.0 5.2 5.4 5.1
```

Ahora, aplicamos la condicional

```
tail(iris$Petal.Length >= 5.5)
```

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

Ahora obtendremos el subconjunto deseado del dataframe.

```
head(iris[iris$Petal.Length >= 5.5, ])
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 101          6.3         3.3          6.0         2.5 virginica
## 103          7.1         3.0          5.9         2.1 virginica
## 104          6.3         2.9          5.6         1.8 virginica
## 105          6.5         3.0          5.8         2.2 virginica
## 106          7.6         3.0          6.6         2.1 virginica
## 108          7.3         2.9          6.3         1.8 virginica
```

Buscamos un subconjunto que contenga las columnas 1, 3 y 5

```
head(iris[iris$Petal.Length >= 5.5, c(1,3,5)])
```

```
##     Sepal.Length Petal.Length   Species
## 101          6.3          6.0 virginica
## 103          7.1          5.9 virginica
## 104          6.3          5.6 virginica
## 105          6.5          5.8 virginica
## 106          7.6          6.6 virginica
## 108          7.3          6.3 virginica
```

Podemos conjuntar varias condiciones usando operadores lógicos.

```
head(iris[iris$Petal.Length >= 5.5 & iris$Petal.Width >= 1, ])
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 101          6.3         3.3          6.0         2.5 virginica
## 103          7.1         3.0          5.9         2.1 virginica
## 104          6.3         2.9          5.6         1.8 virginica
## 105          6.5         3.0          5.8         2.2 virginica
## 106          7.6         3.0          6.6         2.1 virginica
## 108          7.3         2.9          6.3         1.8 virginica
```

```
head(iris[!(iris$Petal.Length < 5), ])
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 78           6.7         3.0          5.0         1.7 versicolor
## 84           6.0         2.7          5.1         1.6 versicolor
## 101          6.3         3.3          6.0         2.5  virginica
## 102          5.8         2.7          5.1         1.9  virginica
## 103          7.1         3.0          5.9         2.1  virginica
## 104          6.3         2.9          5.6         1.8  virginica
```

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Dplyr

El manejo de datos que realizamos hasta ahora fue hecho con funciones base de R, sin embargo, ahora existen otras librerías como Dyplir que permiten realizar transformaciones "complejas" de una manera sencilla, el único inconveniente es que tendríamos que aprender la sintaxis propia de la librería.

Para quien tenga interés por esta alternativa puede revisar el siguiente enlace Tutorial de Dplyr.

Funciones

La estructura de la función es la siguiente

```
nombre <- function(argumentos){
operaciones
}
```

Area de un cuadrado

Creemos la función

```
area_cuadrado <- function(lado){
lado*lado
}
```

Probemos que funcione

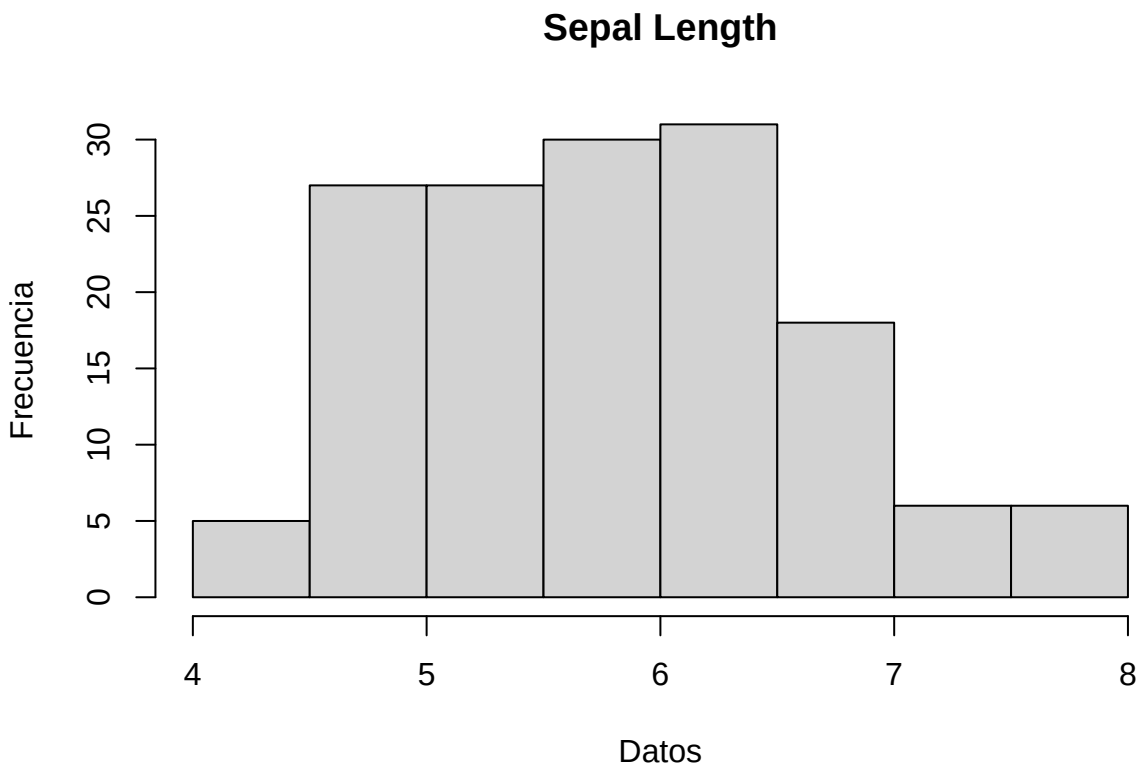
```
area_cuadrado(lado = 2)
```

```
## [1] 4
```

Histograma con estadisticos

```
histograma <- function(Variable, name){
hist(Variable, main = name, xlab = "Datos", ylab = "Frecuencia")
summary(Variable)
}
```

```
histograma(Variable = iris$Sepal.Length, name = "Sepal Length")
```




```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.300   5.100   5.800   5.843   6.400   7.900
```

Estructuras de control

ESTRUCTURA DE CONTROL	DESCRIPCIÓN
if, else	si, de otro modo
for	Para cada uno en

Existen otros como *while*, *break* y *next*, que no veremos.

If y Else

La estructura para operar un if y else, es la siguiente:

```
if(condicion){
operacion si la condicion es TRUE
}else{
Operacion si la condicion es FALSE
}
```

```r
if(5>3){
  "Verdadero"
}else{
  "Falso"
}
```

```
## [1] "Verdadero"
```

Caso: Verificar si un alumno de posgrado mantiene o no la beca de CONACYT

```r
promedio <- function(calificaciones){
  media <- mean(calificaciones)

  if(media >= 8){
    "Conserva la beca"
  } else {
    "Pierde la beca"
  }
}
```

```r
alumno1 <- promedio(c(7, 8.8, 8.5,8.6))
alumno1
```

```
## [1] "Conserva la beca"
```

```r
alumno2 <- promedio(c(9,9,7,6))
alumno2
```

```
## [1] "Pierde la beca"
```

for

La estructura es la siguiente:

```
for(elemento *in* objeto){
operacion con cada elemento
}
```

Ejemplo:

```r
d <- 1:6
for(elemento in d){
  elemento + 3
}
```

```r
d <- 1:6

## Para este caso, como buscamos ir agregando elementos a un vector dup
## Primero, necesitamos crear un vector vacio (NULL)
## al cual le agregaremos los elementos que creemos.
```

```r
dup <- NULL

for(elemento in d){
  dup[elemento] <- elemento + 3
}

print(dup)
```

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

Recursos adicionales

R para principiantes

https://bookdown.org/jboscomendoza/r-principiantes4/datos-mas-comunes.html

Tutorial de Dplyr

https://swcarpentry.github.io/r-novice-gapminder-es/13-dplyr/