Prepa 1 - Introduccion a R

Alimi Garmendia

5/14/2021

Por que R?

Figure 1: R es fino

Operaciones basicas

```
# Operaciones aritmeticas basicas
10+10 # Suma y resta
```

```
## [1] 20
```

```
5*5*5 # Potenciacion y raices
```

```
## [1] 125
```

```
5^2
```

```
## [1] 25
```

```
2^(1/2)
```

```
## [1] 1.414214
```

```
sqrt(2)
```

```
## [1] 1.414214
```

```
# Orden de operaciones es  */^ luego +-
2+5*3
```

```
## [1] 17
```

```
# R es pana y sabe cuando no haz terminado (En la consola)
# 10 +
```

Asignar Variables

Podemos asignar variables usando <- , -> o = . Aunque lo acostumbrado es usar <- y -> para asignar variables y funciones y = para parametros y valores de las funciones

```
a <- 5
a
```

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

```
5*10+3/5 -> b
b
```

```
## [1] 50.6
```

```r
a = 4
a
```

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

Ahora es posible realizar operaciones con las variables creadas

```r
a * b
```

```
## [1] 202.4
```

```r
a+b
```

```
## [1] 54.6
```

```r
a/b
```

```
## [1] 0.07905138
```

Al momento de nombrar variables, es util usar nombres representativos que ayuden a entender que esta guardando dicha variable. Por ejemplo media, valorAbsoluto. Cosas como a, b, x pueden llegar a hacer que nos perdamos cuando estemos revisando nuestro codigo. RECOMENDACION = usarCamelCase

FUN with functions

Para crear una funcion usamos la siguiente plantilla

```r
a <- function(x,y = 7){
  x * 7 + y #podemos usar return, pero no es mucha la diferencia
}
a(3)
```

```
## [1] 28
```

```r
a(3,5)
```

```
## [1] 26
```

```r
a(x= 2, y = 2)
```

```
## [1] 16
```

Cuando tengamos parametros por default estos deben estar lo mas a la derecha posible (y en el ejemplo anterior). RECOMENDACION = UsarCamelCase

Si hay alguna funcion de R y queremos indagar sobre que parametros admite, o que hace podemos usar

```
?round
```

```
## starting httpd help server ... done
```

```
round(3.1416, 2)
```

```
## [1] 3.14
```

Asi podemos acceder rapidamente a la documentacion. La documentacion es Buenisima, en ocasiones hasta mejor que google o stack.

Vectores y el poder de R

Podemos crear vectores para agrupar nuestros datos, es la estructura fundamental de R y en donde comienzamos a explorar el podeR

```
sales.by.month <- c(0, 100, 200, 50, 0, 0, 0, 0, 0, 0, 0, 0)
```

Al tener nuestros datos en una sola estructura podemos usar broadcasting u operaciones vectorizadas

```
sales.by.month * 2
```

```
##  [1]   0 200 400 100   0   0   0   0   0   0   0   0
```

```
sales.by.month + 100
```

```
##  [1] 100 200 300 150 100 100 100 100 100 100 100 100
```

```
round(sales.by.month + 3.1416, digits = 2)
```

```
##  [1]   3.14 103.14 203.14  53.14   3.14   3.14   3.14   3.14   3.14   3.14
## [11]   3.14   3.14
```

```
sales.by.month /0 #Mosca con una vaina sobrino
```

```
##  [1] NaN Inf Inf Inf NaN NaN NaN NaN NaN NaN NaN NaN
```

Sacar informacion de los vectores

Podemos acceder a los valores de un vector a traves de su indice, MOSCA ES 1-BASED INDEXING

```
sales.by.month[1]
```

```
## [1] 0
```

```
sales.by.month[1:4]
```

```
## [1]   0 100 200  50
```

```
sales.by.month[]
```

```
##  [1]   0 100 200  50   0   0   0   0   0   0   0   0
```

```
sales.by.month[-2]
```

```
##  [1]   0 200  50   0   0   0   0   0   0   0   0
```

```
sales.by.month[c(1,3,5)]
```

```
## [1]   0 200   0
```

```
#Podemos asignarle nombres a las variables
primerTrimestre <- sales.by.month[1:3]
names(primerTrimestre) <- c("Enero","Febrero","Marzo")

primerTrimestre['Enero']
```

```
## Enero
##     0
```

Podemos alterar los valores de los vectores de la misma forma que asignariamos una variable

```
sales.by.month[1] <- 25
sales.by.month
```

```
##  [1]  25 100 200  50   0   0   0   0   0   0   0   0
```

Finalmente, podemos obtener informadcion del vector:

```
length(sales.by.month)
```

```
## [1] 12
```

```
max(sales.by.month)
```

```
## [1] 200
```

```
min(sales.by.month)
```

```
## [1] 0
```

```r
sum(sales.by.month)
```

```
## [1] 375
```

```r
sort(sales.by.month)
```

```
##  [1]   0   0   0   0   0   0   0   0  25  50 100 200
```

```r
#Tambien funciona con vectores de characteres
 months.names <- c("January", "February", "March", "April", "May", "June","July", "August", "September"

 sort(months.names)
```

```
##  [1] "April"     "August"    "December"  "February"  "January"   "July"
##  [7] "June"      "March"     "May"       "November"  "October"   "September"
```

```r
 names(sales.by.month) <- months.names
sales.by.month
```

```
##   January  February     March     April       May      June      July    August
##        25       100       200        50         0         0         0         0
## September   October  November  December
##         0         0         0         0
```

Booling

En R podemos usar valores de verdadero usando TRUE y FALSE. Con ellos podemos usar condicionales, realizar comparaciones y algunas otras cosa.

```r
5 > 6
```

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

```r
5 == 12
```

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

```r
5 >= 1
```

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

```r
a = 25

if (a > 12) {

  print("El condicional funciona")

}
```

```
## [1] "El condicional funciona"
```

```
sales.by.month > 12 # Usar comparacion de un vector nos retorna un vector con la comparacion de cada el
```

```
##   January  February     March     April       May      June      July    August
##      TRUE      TRUE      TRUE      TRUE     FALSE     FALSE     FALSE     FALSE
## September   October  November  December
##     FALSE     FALSE     FALSE     FALSE
```

Podemos usar expresiones mucho mas complejas usando operadores logicos (& | !)

```
 !(5 > 3)
```

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

```
 5 > 3 | 2 <4
```

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

```
 5 > 3 & 2 <4
```

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

Disclaimer! OJO con usar T o F

```
T
```

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

```
F
```

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

Aunque funcionan y es muy usado en argumentos de funciones hay que tener cuidado, pues tanto T como F pueden cambiarles su valor al asignarle un nuevo valor

```
T <- 7

T
```

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

Como vimos en ejemplos anteriores podemos obtener un vector de booleanos al realizar comparaciones usando un vector. Esto nos puede ayudar a filtrar datos

```
sales.by.month > 20 # Esto nos dira cuales meses tuvieron un valor  mayor a 20
```

```
##   January  February     March     April       May      June      July    August
##      TRUE      TRUE      TRUE      TRUE     FALSE     FALSE     FALSE     FALSE
## September   October  November  December
##     FALSE     FALSE     FALSE     FALSE
```

```
# Ahora podemos filtrar cuales meses cumplen esa condicion

sales.by.month[sales.by.month > 20]
```

```
##  January February    March    April
##       25      100      200       50
```

Leer de archivos

Durante el curso estaremos trabajando con datos almacenados en archivos, la manera mas sencilla para obtenerlos es

```
books <- read.csv('booksales.csv')
books
```

```
##     ï..Month Days Sales Stock_Levels
## 1     January   31     0         high
## 2    February   28   100         high
## 3       March   31   200          low
## 4       April   30     0          out
## 5         May   31     0          out
## 6        June   30     0         high
## 7        July   31     0         high
## 8      August   31     0         high
## 9   September   30     0         high
## 10    October   31     0         high
## 11   November   30     0         high
## 12   December   31     0         high
```

Es importante ubicar nuestro working directory pues la funcion read.csv toma el path relativo a nuestro working directory

Es importante darle un vistazo a nuestros archivos antes de cargarlos, asi sabremos como cargarlos

```
bookst <- read.csv('booksales.txt')
bookst
```

```
##    Month.Days.Sales.Stock_Levels
## 1              January\t31\t0\thigh
## 2           February\t28\t100\thigh
## 3               March\t31\t200\tlow
## 4               April\t30\t0\tout
## 5                 May\t31\t0\tout
## 6               June\t30\t0\thigh
## 7               July\t31\t0\thigh
## 8              August\t31\t0\thigh
## 9           September\t30\t0\thigh
## 10            October\t31\t0\thigh
## 11           November\t30\t0\thigh
## 12           December\t31\t0\thigh
```

Como podemos ver este archivo no es CSV es separado por tabs ”͡ para ellos usamos

```
bookst <- read.delim('booksales.txt',sep = '\t')
bookst
```

```
##        Month Days Sales Stock_Levels
## 1    January   31     0         high
## 2   February   28   100         high
## 3      March   31   200          low
## 4      April   30     0          out
## 5        May   31     0          out
## 6       June   30     0         high
## 7       July   31     0         high
## 8     August   31     0         high
## 9  September   30     0         high
## 10   October   31     0         high
## 11  November   30     0         high
## 12  December   31     0         high
```

Lo que leimos es guardado en una estructura llamada data frame, que agrupa nuestros datos con columnas con nombres y las observaciones se agrupan por filas.

Podemos resumir nuestros Data Frames, dandoles un sneak peak usando summary().

```
summary(books)
```

```
##    ï..Month              Days           Sales     Stock_Levels
##  Length:12          Min.   :28.00   Min.   :  0   Length:12
##  Class :character   1st Qu.:30.00   1st Qu.:  0   Class :character
##  Mode  :character   Median :31.00   Median :  0   Mode  :character
##                     Mean   :30.42   Mean   : 25
##                     3rd Qu.:31.00   3rd Qu.:  0
##                     Max.   :31.00   Max.   :200
```

Podemos acceder a los datos de las siguientes maneras:

```
books$Days
```

```
##  [1] 31 28 31 30 31 30 31 31 30 31 30 31
```

```
books[,1]
```

```
##  [1] "January"   "February"  "March"     "April"     "May"       "June"
##  [7] "July"      "August"    "September" "October"   "November"  "December"
```

```
books[,-1]
```

```
##    Days Sales Stock_Levels
## 1    31     0         high
## 2    28   100         high
## 3    31   200          low
```

```
## 4    30     0          out
## 5    31     0          out
## 6    30     0         high
## 7    31     0         high
## 8    31     0         high
## 9    30     0         high
## 10   31     0         high
## 11   30     0         high
## 12   31     0         high
```

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

```
##     ï..Month Sales
## 1    January     0
## 2   February   100
## 3      March   200
## 4      April     0
## 5        May     0
## 6       June     0
## 7       July     0
## 8     August     0
## 9  September     0
## 10   October     0
## 11  November     0
## 12  December     0
```

```
books[books$Days  < 30 | books$Sales > 0  ,]
```

```
##   ï..Month Days Sales Stock_Levels
## 2 February   28   100         high
## 3    March   31   200          low
```

Referencias

1. https://learningstatisticswithr.com/lsr-0.6.pdf - Danielle Navarro
2. Imagenes e Ilustraciones de Allison Horst - https://github.com/allisonhorst/stats-illustrations