Introducción a R

Extraído de Secciones 2 a 6 de “An Introdution to R”: https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf

Vectores y asignaciones

- **Realice las siguientes operaciones:**

```
> x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
> x
> assign("x", c(10.4, 5.6, 3.1, 6.4, 21.7))
> c(10.4, 5.6, 3.1, 6.4, 21.7) -> x
> 1/x
> y <- c(x, 0, x)
```

Aritmética con vectores

- Realice:

  > x<-c(1,2,3); y<-c(x,0,x)

  > v <- 2*x + y + 1

  > v

- Otras funciones: +, -, *, /, ^, log, exp, sin, cos, tan, sqrt
- Realice:

  > c(min(x), max(x), length(x), sum(x), prod(x), mean(x), var(x))

  - var(x) es equivalente a: sum((x-mean(x))^2)/(length(x)-1)

Secuencias

- Secuencias equivalentes:
  - 1:30 ; seq(1,30); seq(from=1, to=30); seq(from=1, to=30)
  - seq(-5, -2, by=.2) ; seq(length=16, from=-5, by=.2)
- Formas simples de rep():
  - rep(x, times=5)
  - rep(x, each=5)

Vectores lógicos

> x = 1:10

> x > 7

- Otros operadores lógicos: <, <=, >, >=, ==, !=
- and & , or | , y not ! x>7 & x<=9 ;!(x>7 & x<=9) ; x>7 | x<=9

Valores inexistentes

- Not avaliable: NA z <- c(1:3,NA); is.na(z)
  - Ojo, es diferente z==NA y is.na(z)
- Not a Number: NaN 0/0 ; Inf - Inf
- zz <- c(z,NaN); is.na(zz) ; is.nan(zz)

Caracteres

Prueben:

```
>"Hello" ; 'world!' ;

>c("Hello" , 'world! ')

>paste("Hello" , 'world!')

>paste(c("X","Y"), 1:10)

>paste(c("X","Y"), 1:10, sep="")
```

Otras estructuras en R

matrices: generalización multidimensional de vectores (p.18).

lists: es una generalización de vectores, donde varios elementos no tienen por qué ser del mismo tipo (p.26).

data frames: Son como matrices, pero donde las columnas pueden ser de distinto tipo (p.27).

functions: Objetos de R que permiten extender los comandos de R (p.42).

factors: proveen formas compactas de manejar datos categóricos (p.16).

Otras estructuras en R: matrices

```
>  dim(z) <- c(3,5,100)    # matriz vacía de dimensiones 3x5x100

> x <- array(1:20, dim=c(4,5)) ; x ; x[,1]; x[2,]          #vean el orden en que ponen los números
  1:20

> i <- array(c(1:3,3:1), dim=c(3,2)) ; i ; x[i]

> x[i] <- 0 ; x

> y <- x; x*y ; x %*% t(y)    # vean la diferencia entre los productos    (t(y)  transpone la matriz y)

> diag(y) ; diag(diag(y))      # diag de un vector genera una matriz cuadrada y vacía con el vector
                                 en la diagonal
> solve(A,b)      # resolución de sistemas lineales
```

Otras estructuras en R: listas

```
> Lst <- list(name="Fred", wife="Mary", no.children=3,child.ages=c(4,7,9)) ; Lst

> Lst$wife ; Lst[2] ; Lst[[2]] ; List[["wife"]]                    x<-"wife"; Lst[[x]]

> Lst$child.ages[1] ; Lst$[[4]][1] ;

>  Lst[5] <- list(matrix = array(c(1,2,3,4),dim=c(2,2)))

> Lstt <- c(Lst,Lst)
```

Otras estructuras en R: Data frames

```
> x=c(1,2,3,4) ; y=("ene","feb","mar","abr") ; z=list(Juan = c(1,2,3,4),Pedro =
  c("a","b","c","d"))

> DFrm <- data.frame(Dato1=x, Dato2=y, Dato3=z)

> DFrm$u <- 2*DFrm$Dato1

> attach(DFrm)

> search()

> detach(“DFrm”)
```

Entrada y salida de información

Extraído de https://rstudio-education.github.io/hopr/dataio.html

Importar datos: read.table() function

> getwd() #Conocer el directorio de trabajo

> dir.create("workR")

> setwd("./workR")

Ir a notepad o notepad++ y crear el archivo houses.data con la siguiente información:

```
Price Floor Area Rooms Age Cent.heat
52.00 111.0 830 5 6.2 no
54.75 128.0 710 5 7.5 no
57.50 101.0 1000 5 4.2 no
57.50 131.0 690 6 8.8 no
59.75 93.0 900 5 1.9 yes
```

> HousePrice <- read.table("houses.data") ; HousePrice

> HousePrice <- read.table("houses.data", header=TRUE) ; HousePrice #observe la diferencia

Importar datos: read.table() function

Ir a notepad o notepad++ y crear el archivo `houses.csv` con la siguiente información:

```
Price,Floor,Area,Rooms,Age,Cent.heat
52.00,111.0,830,5,6.2,no
54.75,128.0,710,5,7.5,no
57.50,101.0,1000,5,4.2,no
57.50,131.0,690,6,8.8,no
59.75,93.0,900,5,1.9,yes
```

```
> HPCSV <- read.table("houses.csv") ; HPCSV

> HPCSV <- read.table("houses.csv", header=TRUE) ; HPCSV    #observe la diferencia

> HPCSV <- read.table("houses.csv",  sep = ",",header=TRUE) ; HPCSV
```

Importar y crear datos: scan() y edit()

Ir a notepad o notepad++ y crear el archivo `input.dat` con la siguiente información:

```
52.00 111.0 830 no
54.75 128.0 710 no
57.50 101.0 1000 no
57.50 131.0 690 no
59.75 93.0 900 yes
```

```
> inp <- scan("input.dat",list(0,0,0, "")) ; inp
```

```
> inp <- scan("input.dat", list(Price=0, Floor=0, Area=0, Cent.heat="")); inp
```

```
> inp$Price ; inp$Cent.heat
```

```
> xnew <- edit(HousePrice)
```

```
> xnew <- edit(data.frame())
```

Generar archivos: write.csv()

Almacenar un archivo en formato .CSV

```
> write.csv(HPCSV, "HPSCV.csv")

> write.csv(HPCSV, "HPSCV2.csv", row.names = FALSE)  #Elimina la primer columna con
  nro de fila
```

Comprimir un archivo .CSV

```
> write.csv(HPCSV, file = bzfile("HPSCV2.csv.zip"), row.names = FALSE)   #ábralo con
  winzip o algún otro

> read.csv("HPSCV2.csv.zip")
```

Salvar variables: save() y save.image

Almacenar variables de trabajo.

> xx<-HPCSV; save(xx, HousePrice, file = "stuff.RData");

> rm(xx,HousePrice); xx ; HousePrice

> load("stuff.RData") ; xx; HousePrice

Salvar todo el entorno de trabajo (Global Environment).

> save.image(file="TodosLosDatos.Rdata");

```
> rm(list = ls()) ;   #borra todas las variables
> load("TodosLosDatos.RData")
```