Practica 1

Alejandro Cáceres
UPC - Statistics 2019/2020

Por qué R?

- Es un software libre/gratis
- Es un lenguaje de programación (orientado a objetos)
- Tiene una sintaxis flexible y codificación compacta
- Es fácil para escribir paquetes de software que usan otros paquetes (comunidad)
- Fue inicialmente diseñado como software de estadística pero desde R se pueden hacer hasta appliciones móbiles
- Es fácil para acoplar a gitHub

Por qué R?

- La idea de las prácticas es que puedan ir ejecutando el código que yo muestro en pantalla.
- Usaremos RStudio que es una interface gráfica para R
- El código lo guardaremos en un script (test.R)

Instalando R

- Ir a uno de los R mirrors
  http://cran.es.r-project.org/
- o deste el website de R
  http://www.r-project.org/ (Download CRAN packages -Spain)
- Se puede escoger el archivo binario de la plataforma
  (Se puede descargar y compilarlo)

El entorno de R

R es un programa que corre por líneas de comando.
Se escriben expresiones (2+2) y funciones (`log(34)`)
Se accede por medio de una GUI ventana (como RStudio).

- abre Rstudio
- escibe diferentes comandos:

```
> 2+2
```

```
> c(1,2,3)+2
```

El entorno de R

```
R version 2.10.1 (2009-12-14)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

R scripts

Los comandos pueden ser escriots en un archivo con formato de texto (extensión: .R).
Se pueden copiar y pegar en la ventana de comandos
Se puede unsar cualquier editor de texto (Emacs) o el que viene en RStudio.

- Abre el editor de text de RStudio
- Escribe una serie de comandos y guardalos en un archivo `test.R`.

R scripts

Por ejemplo escribe en el editor de texto:

```
#comentario: asigna valores a a
a<-c(1,2,3,4,5)
#imprimelos
a
#asigna valores a b
b<-2
#imprime la suma entre a y b
a+b
#lista todas las variables definidas
ls()
```

Notas:

- Cada línea seguida por # es ignorada (comentario), <- es la función de asignación (=). Escribiendo el nombre de la variable hace que R la muestre en pantalla. R no muestra nada que no le pidas.
- Copia y pega los comandos del archivo.
- Con el cursor sobre una línea haz [ctrl+Enter] en RStudio y te correra el comando en esa línea.
- Entra el comando: `source("test.R")` y todo el archivo (script) se correrá

Siempre usa un script

R plots

Los gráficos también se obtienene por la línea de comandos:

```
a<-1:10
a
b<-a^2
b
#para a en el eje x y b en el y
plot(a,b)
#para b como funcion de a (~=[ALTGR+4])
plot(b ~ a)
plot(b ~ a, col = "blue", pch = "A")
```

Escribe estos comandos en el archivo

`source("test.R")`

R plots

- Recobra tu trabajo, escribe:
  `source("test.R")`
- Guarda en gráfico como pdf/png/jpeg

R help

- Usa Google para preguntas genéricas como: i.e "substitute missing R" revisa en `R help archive` o cualquier otro foro (los mejores son en Inglés)
- Si quieres conocer que hace una función (por ejemplo `is.na`) escribe:
  `?is.na`
- Mira los ejemplos de la página de ayuda, copia y pega:
  `is.na(c(1, NA))`
- Manual: An Introduction to R: `http://cran.r-project.org/doc/manuals/R-intro.pdf`

vectors

- R opepra sobre estructuras de datos. La más elemental en un vector de un elementos.
- todas las operaciones son funciones del tipo `UnaFuncion()`.
- por jemplo usa la función `is.vector()` para comprobar que el número 17 es un vector `is.vector(17)`

vectors

- Los vectores son de muchas clases, pregunta la clase de un vectos con la función `class()`

```
class(17)
class("male")
class(TRUE)
class(NaN)
class(Inf)
```

vectors

- función `c()` concatena los elementos de un vector de la misma clase:

```
a <- c(10, 5, NaN, 6, Inf)
class(a)
is.vector(a)

b <- c(TRUE,FALSE,TRUE)
class(b)
is.vector(b)
```

vectors -funciones

- Una función fundamental en R es `assign()`

```
assign("a",c(1,2,3,4,5))
a
#otra forma de usar asign es con: <-
a<-1:5
a
#revisa las variables definidas
ls()
```

vectors -funciones

- Una función útil para generar vectores es `seq`

```
seq(1,10)
#o tambien se puede escribir
1:10
#una funcion dentro de otra funcion
assign("b",seq(1,10))
#en notacion mas standard
b<-1:10
```

vectors -funciones

- los elementos de un vector se seleccionan con: `[ ]`

```
b<-seq(1,100,2)
b[5]
b[5]<-999
b

#selecciona y quita elementos
b[c(1,3,4)]
b[1:10]

b[-c(1,3,4)]
b
```

vectors -funciones

- Se puede hacer aritmética con vectores

```
a<-rep(2,10)
a
b<-a+1
b
b*a
exp(a)
```

Matrices

Un arreglo de 2 dimensiones de vectores es una matriz, organizada por filas y columnas.

- usando la función `matrix()`

```
mat<-matrix(c(1,2,3,4,5,6,7,8),
ncol=2,nrow=4)

mat

mat<-matrix(1:8,ncol=4,nrow=2)

mat
```

Matrices-funciones

Pregunta:

- Qué hacen las funciones `ncol` and `nrow`?

Matrices-funciones

Los elementos de una matriz pueden ser seleccionados con:

`[,]`

- Simula una matriz de números aleatorios con `rnorm`

```
x <- matrix(rnorm(100, 1), ncol = 5)
colnames(x)<-c("a","b","c","d","e")
x
```

- Explora la matriz con las funciones head, tail, dim, image
- seleccional el elemento (1,2)

Matrices-funciones

- Selecciona las columnas b, c, d (también se pueden usar nombres)
- Selecciona las filas pares.
- Remplaza la primera y segunda columnas con 0.

Matrices-funciones

La operaciones sobre matrices se pueden hacer sobre toda la matriz o partes de ella

- se puede sumar o multiplicar por un número, etc...
- Funciones útiles: `colSums()`, `rowSums()`, `t()` and `diag()`
- Qué hace el siguiente comando?

  `(x[-1,1:3])[1,1]<-0`

data.frame

los data.frames son un tipo especial de matrices que permiten vectores de diferentes classes en sus columnas

```
v1<-c("a","b","c")
v2<-c(1,2,3)
matFrame<-data.frame(letters=v1,
                     numbers=v2)
class(matFrame[,2])
class(matFrame[,1])
```

data.frame

Cómo se puede crear un `data.frame` de una matriz?

```
matFrame<-as.data.frame(x)
matFrame
```

data.frame

Hay unos datos de ejemplo en R que se llaman: `airquality`

```
> airquality
```

exploremos la base de datos con:

```
> dim(airquality)
> head(airquality)
> class(airquality)
```

data.frame -funciones

- Se pueden seleccionar partes del `data.frame` de varias formas. `data.frame` es una lista de variables (columnas). Por nombre:

```
names(airquality)
airquality$Ozone
airquality$Day
airquality[c("Ozone","Day")]
```

data.frame -funciones

- `data.frame` también es una matriz asi que se puede seleccionar con los corchetes `[]`

  `airquality[1:10,c(2,5)]`
- también por criterio: datos desde Agosto

  `sel<-airquality$Month>=8`
- sel es un vector lógico

  `AugustNovAir<-airquality[sel,]`
- selecciona las filas para las cuales sel es TRUE

Listas

Las listas son colecciones de variables de diferente clase

- crea un lista con diferentes objetos

```
mat<-cbind(c(2,4,6),c(1,2,3))
vec<-c(TRUE,FALSE)
chr<-"hello"
L<-list(M=mat,V=vec,C=chr)
L
```

Listas

- Los elementos de una lista se pueden seleccionar con $

```
names(L)
L$M
L$C
```

- o con corchetes dobles [[]]

```
L[[1]]
L[[2]]
```

- Cambia los nombres de la lista de elementos

```
names(L)<-c("data","output","message")
L
```

Importar y exportar datos

Generalmente los datos se encuentran en archivos como: txt, STATA, EXCEL, etc..

- los datos de texto se leen con `read.table()`

```
dat<-read.table("dat1.txt")
dat[1:5,]
names(dat)
class(dat)
#cambiar las opciones para read.table
?read.table
dat<-read.table("dat1.txt",header=TRUE)
dat
names(dat)
```

Importar y exportar datos

- Se puede exportar datos a un txt

```
write.table(dat,file="myData.txt")
?write.table
write.table(dat,file="myData.txt",
                    quote=FALSE)
```

Importar y exportar datos

- Los datos se puden guardar con `save` en un archivo binario para una futura sesión de R y cargarlos con `load`

```
ls()
save(dat,file="myResults.RData")
#load your previous results
load("myResults.RData")
ls()
dat
```