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 fcil 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 fcil 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 esta en el archivo scriptsPart1.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 lineas de comando.
Se escriben expresiones (2+2) y funciones (`log(34)`)
Se accese 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 text(extension .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:

```
#assign value to a
a<-c(1,2,3,4,5)
#print value of a
a
#assign value to b
b<-2
#print (show) sum of a and b
a+b
#list all variables defined in the session
ls()
```

Notas:

- Cada linea 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.
- En una line haz ctrl+Enter in RStudio y te correra el comando en esa línea.
- escribe: `> 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

- type:

```
a<-1:10
a
b<-a^2
b
plot(a,b)
plot(a,b,col="blue",pch="A")
```

Escribe estos comandos en el archivo `source(samplePlot.R)`

R plots

- Recobra tu trabajo, escribe: `source(samplePlot.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`

Libraries

- Busca en Google: "read stata file in r", cual es el comando o función para hacer esto?
- escribe: `?read.dta`
- Mira la librería en el que está la función
- escribe:

```
library(foreign)
?read.dta
```

- descarga el paquete `ade4` y corre el ejemplo de la función `dudi.acm`

vectors

- R oepra sobre estructuras de datos. La mas elemental en un vector de un elementos.
- todas las operaciones son funciones del tipo `oneFunction()`.
- 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:

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

c(TRUE,FALSE,TRUE)
class(c(TRUE,FALSE,TRUE))
is.vector(c(TRUE,FALSE,TRUE))
```

vectors -functions

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

```
assign("a",c(1,2,3,4,5))
a
#another way of using assign is with <-
a<-1:5
a
#check the variables you have defined
ls()
```

vectors -functions

- Una función útil para generar vectores es `seq`

```
seq(1,10)
#or
1:10
#a function within function
assign("b",seq(1,10))
#or a more standard use
b<-1:10
```

vectors -functions

- los elementos de un vector se seleccionan con [ ]

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

#select and remove many elements
b[c(1,3,4)]
b[1:10]

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

vectors -functions

- Se puede hacer aritmetica con vectores

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

Matrix

Un arreglo de 2 dimensiones de vectores es una matriz, organizada por filas y columnas.
Las matrices pueden ser construidas por medio de la concatenación de vectores (y asignadas a una variable)

- concatena columnas `cbind()`

```
matR<-cbind(c(1,2,3,4),c(2,4,6,8))
matR
```

- concatena filas `rbind()`

```
matC<-rbind(c(1,2,3,4),c(2,4,6,8))
matC
```

Matrix

Las matrices también se contruyen dándole el formato adecuado a una serie de datos

- usando la función `matrix()`

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

mat

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

mat
```

Matrix-functions

Ejercicio:

- Que hacen las functiones `colnames()` y `rownames()`?
- Nombra las filas `mat<-matrix(1:10,5,2)` con las letras del alfabeto

otras funciónes útiles son `ncol` and `nrow`

Matrix-functions

Specific elements or subsets of the matrix can be selected with `[,]`

- use `rnorm` to simulate a data set

```
x <- matrix(rnorm(100, 1), ncol = 5)
colnames(x)<-c("a","b","c","d","e")
x
#explore the matrix...
head(x)
tail(x)
dim(x)
image(x)
#select one element
x[1,2]
```

Matrix-functions

- selecciona las columnas b,c,d
- selecciona las filas pares.
- Remplaza la primera y segunda columnas con 0.

Matrix-functions

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

- se puede sumar o multiplicar por un número, etc...
- Functiones 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

```
mat<-cbind(c("a","b","c"),c(1,2,3))
colnames(mat)<-c("letters","numbers")
mat
mat[,2]
#numbers get coerced into characters
class(mat[,2])
class(mat)
```

data.frame

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

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

```
newCol<-c(rep("yes",130),rep("maybe",10),
rep("no",13))
```

data.frame

```
newAir<-cbind(airquality,newCol)
class(newAir[,3])
class(newAir[,7])
#en data.frame los vectores
#se convierten a factores
newAir[,7]
```

data.frame

Como se puede crear un `data.frame`?

- conviertiendo la clase

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

data.frame

- También se puede directamente con la función data.frame

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

data.frame -functions

- 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 -functions

- `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 ($Month >= 8$)

```
sel<-airquality$Month>=8
#whatever matches TRUE rows-wise
#will be selected

AugustNovAir<-airquality[sel,]
```

factor

- los factores son vectores que contienen variables categoricas

```
newCol<-c(rep("yes",130),rep("maybe",10),
          rep("no",13))
facCol<-as.factor(newCol)
```

- Los niveles del factor se obtienen como

```
levels(facCol)
#the levels can be renamed
levels(facCol)<-c("a","b","c")
facCol
```

factor

- una función útile para factores es

  `table(airquality$Month,facCol)`
- la media para cada factor se puede calcular como

  `tapply(airquality$Temp,facCol,mean)`

List

Las listas son colecciones de variables de diferente clase, de hecho un `data.frame` en una lista

- 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
```

List

- 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
```

Importing/exporting data

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)
#change options for the read
?read.table
dat<-read.table("dat1.txt",header=TRUE)
dat
names(dat)
```

Importing/exporting data

- Se puede exportar datos a un txt

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

Importing/exporting data

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

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