Sessió 2

Més vectors. Condicions lògiques. Matrius.

2.1 Construcció de vectors de caràcters

- La funció `paste` serveix per crear *strings* o cadenes de caràcters

```
 > paste("X","Y","Zabc",sep="")  # sense espai de separació
[1] "XYZabc"
 > paste("X",1:4, sep="")   # com que 1:4 és un vector, crea un vector
[1] "X1"  "X2"  "X3"  "X4"
 > paste("Pacient",2:6, sep=".")
[1] "Pacient.2" "Pacient.3" "Pacient.4" "Pacient.5" "Pacient.6"
 > paste(c("X,Y"),1:4, sep="")
[1] "X,Y1" "X,Y2" "X,Y3" "X,Y4"
 > paste(c("X","Y"),1:4, sep="")
[1] "X1" "Y2" "X3" "Y4"
 > rep(c("X","Y"),rep(3,2))
[1] "X" "X" "X" "Y" "Y" "Y"
 > paste(rep(c("X","Y"),rep(3,2)),c(1:3,1:3),sep="")
[1] "X1" "X2" "X3" "Y1" "Y2" "Y3"
```

- L'argument `collapse` serveix per enganxar les components del vector de caràcters generat per `paste` en una sola cadena

```
 paste(c("X"),1:4, sep="",collapse="+")
 'X1+X2+X3+X4'
```

- Podem assignar **noms** a les components dels vectors.

  Amb la concatenació `c` i la funció `names` podem posar noms als elements d'un vector

```
 x1<-c(19, 176, 82.7)
 names(x1)
NULL    # les components del vector x1 no tenen noms
 names(x1)<-c("edat", "alt", "pes")  # ara els hi assignem noms
```

```
                # fixem-nos que names(x1) és un vector
 x1                    # en fer sortir x1 per pantalla, apareixen
edat   alt   pes       # els noms de les components
19.0 176.0  82.7
```

Pràctica:
• Fent servir `paste` i els dos punts (`:`), crea la variable identificador `id` de la forma `id1,...,id10`
• Genera un vector que contingui el codi de 10 pacients dones (codificades amb "D") i 5 homes (codificats amb "H"). Els codis de les dones han d'estar numerats del valor 206 al 215 i el dels homes del 201 al 205. A més, el separador entre la lletra i el número ha de ser un guió "-". Per exemple la primera dona hauria de ser `D-206`
• Considerem la variable temperatura màxima mitjana dels mesos de gener fins a desembre:
15, 15, 17, 20, 23, 27, 29, 29, 26, 23, 18, 15.
Crea un vector `temp` amb les temperatures i després amb la funció `names` aplicada al vector, posar com a noms els mesos de l'any abreujats (recorda la funció `month.abb`).

2.2 Operacions amb vectors numèrics i algunes funcions

R està preparat per operar amb vectors numèrics component a component. Mirem què passa amb els exemples següents:

```
x<- 1:10          # vector numèric de longitud 10
y<- 45:54         # vector numèric de longitud 10
x*y               # resultat: vector numèric de longitud 10
x+y               # resultat: vector numèric de longitud 10
x^2               # resultat: vector numèric de longitud 10
sqrt(x)           # idem
1/x               # idem
```

Què passa si sumem dos vectors de diferent longitud?

```
10*x+1          # vector més número dona vector numèric de longitud 10
x1<-c(10,20,30)     # vector longitud 3
x2<-c(4,5,1,3,4,5)  # vector longitud 6
x1+x2               # suma c(x1,x1) amb x2 (és a dir repeteix dos cops x1)
[1] 14 25 31 13 24 35
x3<-c(20,10)        # vector longitud 2
x1+x3             # dona error
Warning message:
In x1 + x2 :
  longer object length is not a multiple of shorter object length
```

Funcions bàsiques amb vectors

Quan tenim dos vectors, podem enganxar-los (**concatenar-los**) amb `c()`, construint així un nou vector que tingui longitud igual a la suma de les seves longituds.

```
x1<-c(10,20,30)      # vector longitud 3
x2<-c(4,5,1,3,4,5)   # vector longitud 6
(x4<-c(x1,x2))       # vector de longitud 9
```

Dues funcions bàsiques per treballar amb vectors són `sum` (suma de les components) i `length` (nombre de components):

```
sum(x2)
length(x4)
```

Hem vist que podem assignar noms a les components dels vectors amb la funció `names`

```
names(x1)<- c("id1","id2","id3")
```

Pràctica:
• Calcula la mitjana aritmètica dels números 3,5,6,7,8,9, creant un vector amb aquests números i després utilitzant les funcions `sum` i `length`.
• Les sortides 3, 4, 5, 6 de l'autopista estan als km 68, 123, 157 i 190. Calcula la distància en metres de les sortides 3, 4, 5 i 6 a l'àrea de servei que està al km 233.
• L'avaluació del curs consisteix en: primer parcial, segon parcial, primera prova d'ordinador, segona prova d'ordinador i lliuraments setmanals, amb pesos respectius de 20%, 25%, 20%, 25% i 10% en la nota final.
Les notes dels 5 primers alumnes són:
Parcial.1 (79, 45, 83, 100, 62)
Parcial.2 (85, 55, 75, 98, 82)
Ordinador.1 (100, 63, 55, 90, 51)
Ordinador.2 (85, 37, 88, 95, 77)
Lliuraments (100, 0, 80, 90, 80)
Quants aprovats hi ha? (Crea els vectors i el càlcul de la nota final.)

2.3 Vectors lògics i expressions lògiques

A **R** s'utilitzen molt les condicions lògiques, per seleccionar trossos de vectors (això darrer ho veurem més endavant).

```
> 3>5
FALSE
> x<-c(10,2,5,6,10,3,11)
> x>=5   # per a cada component, mira si és major o igual que 5
[1]  TRUE FALSE  TRUE  TRUE  TRUE FALSE TRUE
> x == rep(10,7)  # la condició 'és igual a' s'escriu amb == ,
[1]  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE
                    # hem comparat dos vectors de longitud 7
> x == 10  # és equivalent a l'anterior:
[1]  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE
```

Exemple: Si volem saber quantes components del vector `x` són més grans que 4, podem utilitzar la funció `sum`, que suma les components d'un vector numèric aplicada, però, a un vector de tipus lògic:

```
> sum(c(1,3,4))
8
> sum(c(TRUE,TRUE,FALSE))  # converteix els TRUE a 1 i els FALSE a 0
2
> x<-c(10,2,5,6,10,3,11)
> sum(x>4)
5
```

• Ja hem vist que un valor especial és el `NA` (valor no disponible o perdut). Podem detectar-lo amb la funció `is.na()`.

```
> u<- c(2,3,4,NA,6,NA,100)
> is.na(u)
[1] FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE
> sum(is.na(u))   # quants valors perduts té el vector u?
2
```

Expressions de comparació i operadors lògics

Els operadors de relació són:

`<`	menor	`>`	més gran
`<=`	menor o igual	`>=`	més gran o igual
`==`	igual	`!=`	diferent

Els operadors lògics són:

- `!` negació
- `&` i lògic (han de ser certes les dues condicions)
- `|` o lògic (ha de ser certa almenys una de les condicions)

Fixeu-vos en l'avaluació de les expressions lògiques següents:

```
 !(3<10)         # com que 3<10 dona TRUE, la negació és FALSE
FALSE
 (3==6) & (3<10)  # les dues comparacions han de ser certes
FALSE
 (3==6) | (3<10)  # alguna de les dues comparacions ha de ser certa
TRUE
```

La funció %in% (pertany)

```
>  x<-c(1:4,10,1:3)
>  2 %in% x   # 2 pertany al vector x?
[1] TRUE      # si
>  8 %in% x  # 8 pertany al vector x?
[1] FALSE     # no
> y<-c(10,11)
> y %in% x     # 10 i 11 pertanyen al vector x?
[1]  TRUE FALSE   # 10 sí i 11 no
```

Funcions que s'apliquen a vectors lògics

Les funcions `ANY` i `ALL` actuen sobre vectors lògics i ens diuen en el primer cas si el vector té algun `TRUE`, i en el segon, si totes les components del vector són `TRUE`.

Exemples:

```
 x<- (-1):10
 any(x<0)
[1] TRUE
 all(x>=0)
[1] FALSE
y<-c(3,4,NA,6,5)
any(is.na(y))
[1] TRUE
```

En l'anterior codi hem generat primer el vector x igual a $(-1, 0, 1, 2, 3, \ldots, 10)$. Quan escrivim `x<0` genera el vector lògic
`TRUE FALSE FALSE ... FALSE`
Ara apliquem a aquest vector lògic la funció `any` que compta si hi ha algun `TRUE` en el vector lògic anterior.

La funció `all` només dona com a resultat TRUE si totes les components del vector lògic al que s'aplica són TRUE. En aquest cas, el vector generat amb `x>=0` és
`FALSE TRUE TRUE TRUE ...`
per tant el resultat d'avaluar `all(x>=0)` és `FALSE`.

Procedim de manera anàloga per detectar si un vector té algun valor perdut (`NA`).

Pràctica: Seguim amb un vector de notes corresponents a un parcial (notes de 0 a 10 on s'aprova amb un 5 o més, notable a partir de 7, excel·lent a partir de 9 i NA per als no presentats).
Crea les instruccions en R que responguin a les preguntes que hi ha a continuació tenint en compte que el vector de notes és:

```
notes <- c(9.5, 8, NA, NA, 5.7, 5, 6.1, 5.2, 3.7, 0.8, 9.3, 6, 2, 8.3,
6.4,2.9, 8.9, 4.8, 3.9, 0, 8.8, 9.4, 5.2, 9.3, 8.3, 8.7, 3.1, 5.8, 3.1,
NA, 6.9, NA, 0.3, 5, 5, 7.6, 2.8, 9, 7, 7.3, 2.2, 6, 9.2, 1.9, 0, 3.8, 6.9,
9.2, NA, 8.8, 5, NA, 6.4, 2.5, 1.3, 0, 7.2, 5, NA, 3.6)
```

(a) Quants/es alumnes hi ha a classe?
(b) S'han presentat tots/es els/les alumnes?
(c) Quants/es alumnes no s'han presentat?
(d) Quants/es alumnes han tret una nota diferent de 0?
(Ajuda: potser hauràs de requerir que els valors no siguin NA i que siguin > 0)
(e) Quants/es alumnes han suspès?
(f) Quants/es alumnes han tret un 5 o un 6?
(g) Quants/es alumnes han tret un notable?
(h) Quants/es alumnes han tret un excel·lent?

2.4 Matrius $(n \times m)$ i k-arrays $(n_1 \times \cdots \times n_k)$

Una matriu és un conjunt d'elements d'un mateix tipus (mode) organitzats en files i columnes.

- Les matrius es creen a partir de la funció `matrix`. És habitual crear primer el vector i després transformar el vector en matriu especificant les dimensions (amb l'argument `dim`) i la manera d'omplir la matriu és per columnes o per files (per defecte és per columnes, és a dir, `byrow=F`, cal especificar `byrow=T` si es vol fer per files):

```
m<-c(1,1,1,2,2,2) # comencem amb un vector d'elements
(M<-matrix(m,nrow=2)) # el transformem en una matriu de dues files
(M<-matrix(m,ncol=3)) # igual que l'anterior
(M<-matrix(m,2,3))            # fa el mateix
dim(m)<-c(2,3); m  ## idem, hem perdut el vector "m", ara és matriu
      # fixeu-vos que, per defecte, omple per columnes !!
M<-matrix(m,nrow=2,byrow=T); M  # forcem a omplir per files
```

- Crear una nova matriu a partir d'altres amb les funcions `rbind` i `cbind`, si són compatibles

```
M1<-matrix(m,2,3,byrow=T); M2<-matrix(m,3,2,byrow=F)
M3<-matrix(c(1,1,1,2,2,2,3,3,3),3,3,byrow=T)
M1;M2;M3
M<-rbind(M1,M3); W<-cbind(M2,M3); M; W
```

- Assignem noms a files i columnes de la matriu `W`

```
rownames(W)<-paste(c("W"),1:3, sep="")
colnames(W)<-1:5; W
dim(W)
 ## [1] 3 5
dimnames(W)
 ## [[1]] [1] "W1" "W2" "W3"          #  noms 1a dim. (files)
 ## [[2]] [1] "1" "2" "3" "4" "5"     #  noms 2a dim. (cols.)
colnames(W); rownames(W)
```

- Demanem més atributs de la matriu

```
length(W)  # nombre d'elements nrow x ncol
 ## [1] 15
mode(W)   # referit al tipus d'elements que conté
 ## [1] "numeric"
class(W)  # referit al tipus d'objecte que és
 ## [1] "matrix"
```

- Atenció quan el nombre d'elements no lliga amb les dimensions

```
B<-matrix(1:18,4,5)           # n'hi falten 2, torna a començar   !
B<-matrix(c(1:18,NA,NA),4,5)  # forcem 2 "missings"
```

- Una matriu és un 2-***array***. Un 3-***array*** es pot pensar com una col·lecció de matrius una darrera l'altra formant "capes" o fulls, totes els fulls amb el mateix nombre de files i columnes. Definim amb la funció `array` un d'aquests objectes i demanem els seus atributs:

```
A<-array(1:32, dim=c(2,4,4))     # elements i dimensions
dim(A)
 ## [1] 2 4 4   ## 2 files, 4 columnes, 4 capes (o fulls)
length(A) # nombre total d'elements
mode(A); class(A)
```

Pràctica:
• Crea una matriu $5\times$ 6 amb nombres aleatoris de l'1 al 5. Anomena-la `A1`. [Ajuda: per crear números aleatoris del 1 al 5 pots utilitzar la funció `sample`. Escrivint `?sample` trobaràs exemples sobre com utilitzar-la.]
• Crea l'script necessari per generar la matriu següent:

$$A2 = \begin{pmatrix} 10 & 10 & 10 & 10 & 10 & 10 \\ 20 & 20 & 20 & 20 & 20 & 20 \\ 30 & 30 & 30 & 30 & 30 & 30 \end{pmatrix}$$

• Ajunta les matrius `A1` i `A2` en una nova matriu `A3`.
• Anomena les fileres de la matriu `A3` amb una 'F' seguida dels números correlatius a partir de 101 i les columnes amb números correlatius a partir de 201.
• Quins atributs té A3?

2.5 Llibreries o paquets

Les llibreries o paquets són grups de funcions[1]. Les llibreries més bàsiques s'instal·len en instal·lar el programa, altres es poden baixar del web de `CRAN` (Comprehensive R Archive Network):

`http://cran.r-project.org` .

`CRAN` és un repositori, però n'hi ha d'altres, com ara `bioconductor`.

Les llibreries instal·lades a l'ordinador en principi no es carreguen quan engeguem el **`RStudio`**, si les volem utilitzar les hem de *carregar*. Quan una llibreria no es troba al nostre ordinador, hem de fer dos passos per utilitzar-la: instal·lar-la i després carregar-la.

- Informació de llibreries

```
library()                # llista de paquets disponibles
library(help=splines)    # documentació sobre una llibreria
search()                 # llista de llibreries carregades
ls(4)                    # funcions del 4rt paquet carregat
```

- Per carregar una llibreria es pot fer

[1]Una funció s'aplica a uns paràmetres o arguments, els valors dels quals s'especifiquen dins del parèntesi `( )`. Si no s'especifiquen paràmetres, es prenen els valors per omissió que s'hauran definit a la funció. Per exemple, `exp(10); exp(10,base=10); exp(10,base=2)`

```
  library(splines)            # o també
  require(splines)
# comprovem que s'ha carregat ...
  search()                    # la veiem carregada
  detach('package:splines')   # la traiem
  search()                    # no hi és
```

- A vegades necessitarem paquets que no estan instal·lats al nostre ordinador (no apareixen al llistat de paquets carregables quan executem `library()`).

  Per instalar paquets podem fer-ho des de la consola:

```
  install.packages('combinat')  # instal.lem el paquet combinat
  library(combinat)             # ara el carreguem
```

Gestió de paquets amb **RStudio**

Podem gestionar els paquets amb l'ajuda de **RStudio**, a la pestanya **Packages** de la finestra de baix a la dreta.

- Quan anem a la pestanya de paquets, ens surt una llista de tots els paquets instal·lats a l'ordinador.
- Quan marquem el quadradet del costat del nom d'un paquet, es carrega (marcar el quadrat de `combinat` equival a escriure `library('combinat')`)
- En clicar el nom d'un paquet, obtenim informació del paquet a la pestanya **Help** (equival a `help(package='combinat')`
- Per instal·lar un paquet cliquem al botó **Install**.

Pràctica:
• Treu una llista els paquets o llibreries instal·lats a l'ordinador en què treballes.
• Instal·la i carrega el paquet `vioplot` mitjançant les comandes corresponents.
• Comprova que realment s'ha carregat i busca en la documentació sobre aquesta llibreria quin és el seu títol i la seva descripció.
• Quines funcions té aquest paquet?

2.6 Ajudes

Hi ha diverses maneres de demanar ajuda:

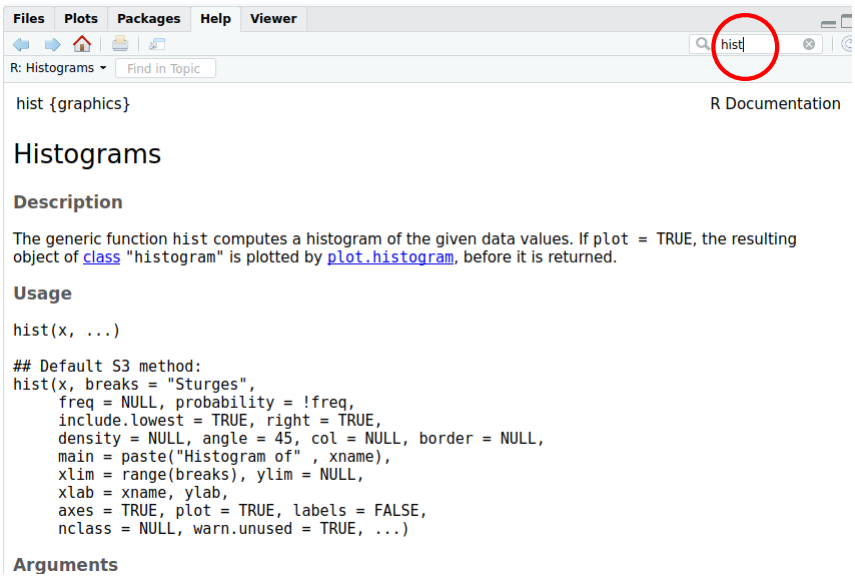
- Ajuda genèrica

```
  apropos('text')    # llista de comandes amb la paraula text
  ??'save'           # llibreries que inclouen la paraula save
```

- Ajuda d'una comanda específica

```
help(save)
?save            # idem
help('if')       # entre ' '  paraules reservades: for, function,...
```

- Una altra manera d'aconseguir ajuda amb **RStudio** és escriure una comanda al nostre script i pitjar la tecla **F1**. Apareix l'ajuda a la pestanya de help (cantonada inferior dreta)



També podeu accedir a l'ajuda directament a través del requadre que hi ha per introduir la consulta (cercle vermell). Al final de la pantalla d'ajuda hi ha exemples que ajuden a fer anar correctament la funció.

- També amb **RStudio**, en les funcions se'ns ofereix autocompletació i apareix un petit quadre d'ajuda sobre la funció, just quan hem acabat d'escriure-la.

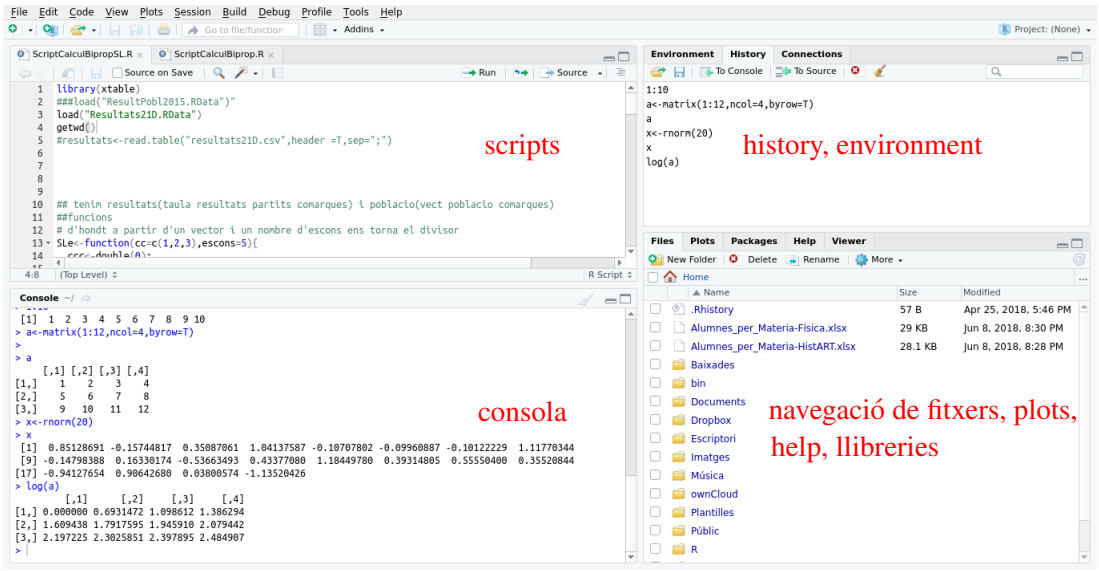
- Altres ajudes, estructurades des del més general fins als exemples

```
help.start()              # info sobre R
help('hist')     # termes relacionats amb 'hist'
example('hist')         # exemples d'aquest diagrama
```

Coneixeu el diagrama de caixa? Més endavant treballarem els diagrames de caixa, però ara podem fer una primera interpretació intuitiva amb l'ajuda del professor de l'exemple *"Guinea pigs' tooth growth"*, que és el sisè exemple que apareix.

Les pestanyes Environment i History de RStudio

La disposició habitual de les finestres de RStudio és:



A la cantonada superior dreta de **RStudio** hi ha dues pestanyes molt útils per treballar:

History

- **History** recorda totes les comandes que hem executat. Fixem-nos que encara que treballem amb un script, quan fem **Control+Enter** la comanda apareix a la finestra consola.
- Podem recuperar una instrucció ja executada, anant a history i clicant al botó **ToSource** (al script) o al botó **ToConsole**.
- Quan sortim de **RStudio** ens demana si volem guardar el `workspace`. Si contestem que sí, en el directori de treball apareixen dos fitxers, un d'ells anomenat `.Rhistory`, que guarda tot l'historial de comandes i es carrega si engeguem **RStudio** un altre cop des del mateix driectori. Aleshores veurem totes les comandes executades en la sessió anterior.

Environment

- A la pestanya **Environment** hi apareix la llista de tots els objectes de **R** que hem creat a la sessió (les variables).
- Veiem quina llargada tenen, de quin tipus són, etc.
- Quan feu una nova assignació, automàticament apareix a la llista.