L'entorn estadístic R

Estructura, llenguatge i sintaxi

Daniel Liviano Solís
Maria Pujol Jover

PID_00208265

Universitat Oberta de Catalunya

Índex

Introducció

L'objectiu principal d'aquest mòdul és introduir l'estudiant en l'entorn estadístic R. Es podria definir R com un llenguatge i un entorn per a computació i gràfics estadístics. És un projecte lliure, és a dir, no requereix el pagament de cap llicència. R proporciona una àmplia varietat de tècniques estadístiques (models lineals i no lineals, proves estadístiques clàssiques, anàlisi de sèries temporals, anàlisi multivariant, etc.). A més, també ofereix una alta potencialitat a l'hora d'elaborar gràfics complexos i de qualitat.

L'ús d'R en les assignatures quantitatives que ofereix la UOC permet a l'estudiant moltes i variades possibilitats d'anàlisi numèrica i estadística. Algunes es detallen a continuació. R proporciona:

1) Facilitats per a emmagatzemar i manejar dades numèriques en diferents formats, com vectors, matrius i bases de dades.

2) Una àmplia col·lecció d'operadors i funcions matemàtiques.

3) Una llarga llista de paquets estadístics amb tota mena d'eines estadístiques i matemàtiques.

4) Facilitats gràfiques per a l'anàlisi i visualització de dades.

5) Un llenguatge de programació senzill i eficaç ben desenvolupat, que inclou condicionals, bucles i funcions recursives.

6) La possibilitat d'afegir funcionalitats addicionals mitjançant la definició de noves funcions creades per l'usuari.

7) La possibilitat d'estendre's mitjançant paquets. Hi ha prop de vuit paquests bàsics subministrats amb la distribució d'R, però n'hi ha molts més de disponibles en el CRAN (*Comprehensive R Archive Network*).

S'ha de reconèixer que començar a treballar amb R no és fàcil, sobretot per als estudiants que no estan acostumats a utilitzar paquets informàtics d'anàlisi numèrica. És a dir, hi ha una certa corba d'aprenentatge que cal superar amb la pràctica. Aquest mòdul cerca, doncs, servir de suport en aquests primers passos sempre complicats, de manera que l'estudiant es familiaritzi amb aquest paquet estadístic i en pugui aprofitar tota la potencialitat.

Per acabar, aquest mòdul introdueix l'extensió R-Commander, la qual farem servir al llarg dels diferents mòduls que formen aquests materials. Actualment, R-Commander

és una de les alternatives més viables a paquets estadístics comercials com Minitab i SPSS, de fet és la més utilitzada en docència en la majoria de les universitats del planeta. Permet executar una part de les possibilitats d'R mitjançant un entorn amb menús desplegables, ideal per a estudiants d'estadística de grau. A més, el fet que el codi subjacent es mostri permanentment en pantalla fa d'R-Commander una bona alternativa per a la primera fase d'aprenentatge del llenguatge d'R.

Objectius

1. Instal·lar correctament R en l'ordinador.

2. Aprendre a instal·lar correctament l'aplicació R-Commander.

3. Conèixer els components principals d'R-Commander.

4. Instal·lar i carregar paquets addicionals en funció de les necessitats de l'anàlisi tant amb R com amb R-Commander.

5. Utilitzar les eines d'ajuda que ofereixen R i R-Commander per a resoldre dubtes.

6. Localitzar els manuals disponibles en el web oficial del programa.

7. Identificar els tipus principals de dades i objectes admesos en R, i poder-los manejar eficientment.

8. Crear un *script* per a escriure una anàlisi de manera coherent, assignant noms a noves variables.

9. Fer operacions matemàtiques i estadístiques amb vectors i matrius.

10. Dominar els operadors lògics i les funcions recursives, cicles i condicionals.

11. En general, dur a terme tota mena d'operacions matricials.

1. Primers passos amb R

1.1. Què és R?

Aquesta és una pregunta simple que no té una resposta fàcil. En la definició més àmplia, R és un llenguatge de programació que permet a l'usuari programar algorismes i fer servir eines programades per altres. Si ens limitem a l'anàlisi estadística i quantitativa, podem afirmar sense por d'exagerar que R no té límits, és a dir, és capaç de fer qualsevol cosa que l'usuari pugui imaginar. Amb R es poden escriure funcions, fer càlculs, aplicar tècniques estadístiques complexes i implementar tècniques *ad hoc* (la manera més senzilla d'aplicar tècniques complicades és aprofitar les que ja estan disponibles en la Xarxa i, en cas necessari, optar per desenvolupar-les nosaltres mateixos), crear gràfics simples i complexos, i fins i tot escriure funcions pròpies. El fet que molts centres d'investigació i universitats el facin servir fa que hi hagi milers de contribucions disponibles per a tots els usuaris. A més, hi ha material abundant (llibres, manuals, exemples, etc.) d'alta qualitat disponibles gratuïtament. Un avantatge gens menyspreable és que, a diferència de molts programes estadístics, R no té cap cost, ja que és un projecte GNU (programari lliure).

The R Project for Statistical Computing

En la pàgina web oficial *http://www.r-project.org* trobarem tota la informació que necessitem sobre aquest programa en anglès: arxius d'instal·lació, característiques, paquets disponibles, manuals, etc. Per a trobar manuals en castellà, hem d'anar a la pàgina *http://cran.es.r-project.org/* i anar a la secció *Contributed*.

En aquest context, els avantatges principals d'R es poden resumir en la llista següent:

- És un programa distribuït lliurement sota una llicència GNU.
- Hi ha molt material de suport: manuals, programes d'aprenentatge i exemples.
- A més, hi ha llibreries que cobreixen gairebé totes les metodologies de l'estadística i les matemàtiques (optimització, sèries temporals, programació lineal, equacions diferencials, inferència, etc.).
- Permet sistematitzar anàlisis llargues i complexes en unes poques instruccions sintètiques, per la qual cosa és un programa molt eficient.

Que ningú no s'espanti! És normal que, al principi, introduir-se en el món de la programació costi una mica. Però una vegada us acostumeu a aquest nou entorn i el domineu, tot serà molt més fàcil.

No obstant això, també podem esmentar alguns inconvenients:

- Hi ha una barrera d'entrada o corba d'aprenentatge considerable.
- No és la mena de programa amb finestres i menús, com ho poden ser Minitab, E-Views o SPSS. No obstant això, hi ha extensions d'R, com R-Commander, que sí que es basen en l'ús de finestres i menús i que, per tant, minimitzen aquest inconvenient.
- Al principi, aprendre a utilitzar R requereix primer aprendre a programar.

1.2. Instal·lació

El primer pas obligat és instal·lar el programa, i per a això haurem de visitar la pàgina web del projecte:

http://www.r-project.org/

Un cop allà, veurem que hi ha diverses seccions, entre elles el centre de descàrregues *CRAN*. Si hi accedim, tindrem en pantalla una llista de *mirrors* (rèpliques), és a dir, de servidors des d'on es poden descarregar els arxius. Seleccionarem el servidor que estigui geogràficament més a prop de la nostra ubicació, i ens apareixerà una pàgina amb diverses opcions d'instal·lació. La manera més ràpida és seleccionar una de les opcions següents que dependrà del nostre sistema operatiu:

- Download R for Linux
- Download R for MacOS X
- Download R for Windows

Si suposem que tenim un sistema operatiu Windows, hi accedim i descarreguem la distribució base, a la qual s'accedeix amb l'opció *install R for the first time*. Seguim les instruccions i descarreguem l'arxiu executable, i després fem la instal·lació sense més complicacions.

Els vídeos són útils

A Internet podeu trobar multitud de vídeos en anglès i en castellà sobre com instal·lar R. Molts estan disponibles a *www.youtube.com*.

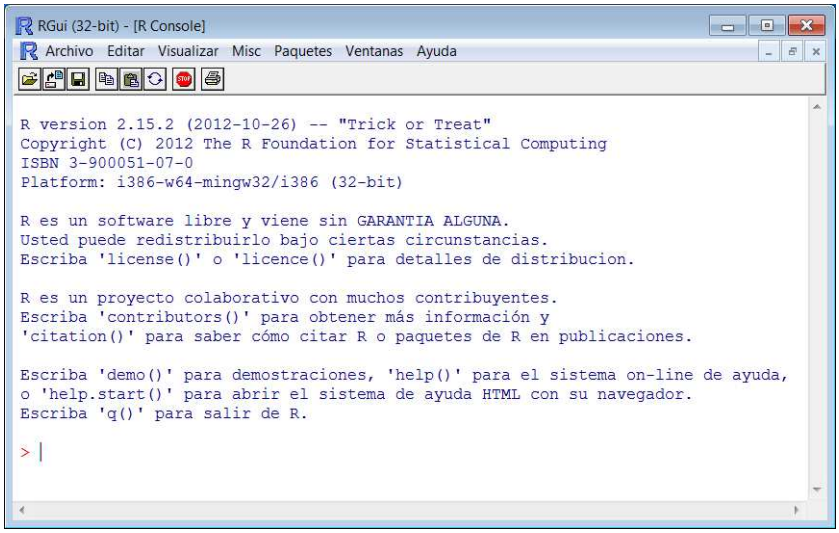
Si teniu un Mac o una distribució del Linux, haureu de tenir en compte, a més, la versió del sistema operatiu que teniu en el vostre ordinador.

1.3. Iniciar una sessió

Un cop instal·lat, ja podem obrir el programa, des de l'accés directe o des de l'arxiu executable inclòs en la carpeta d'instal·lació d'R. Sigui com sigui, en executar R ens trobarem amb la finestra següent:

És important tenir sempre instal·lada l'última versió d'R, ja que les noves versions sempre inclouen millores. A més, ens estalviarem problemes de compatibilitat amb els últims paquets que instal·lem.

Figura 1. Consola d'R



Com podem veure, les instruccions es poden introduir directament en la consola, en el cursor que apareix darrere el símbol >. Un cop introduïdes les instruccions, si premem *Enter* s'executaran els càlculs, igual que amb una calculadora:

```
> 2 * 3
[1] 6
```

Això no obstant, a l'hora de fer una anàlisi llarga, el més convenient és escriure el conjunt d'instruccions amb els càlculs que necessitem en un arxiu de text, i això es pot fer de dues maneres:

- En un *script*, la qual cosa té l'avantatge que es pot obrir i desar com un arxiu amb format *.R*, des de la barra de menús del programa. Aquesta opció, que es mostra en la figura 2, permet executar cada línia de codi o tota una selecció de línies mitjançant la instrucció *CONTROL + R*, i el resultat apareix de color blau en la consola:

Figura 2. Consola d'R amb editor de textos

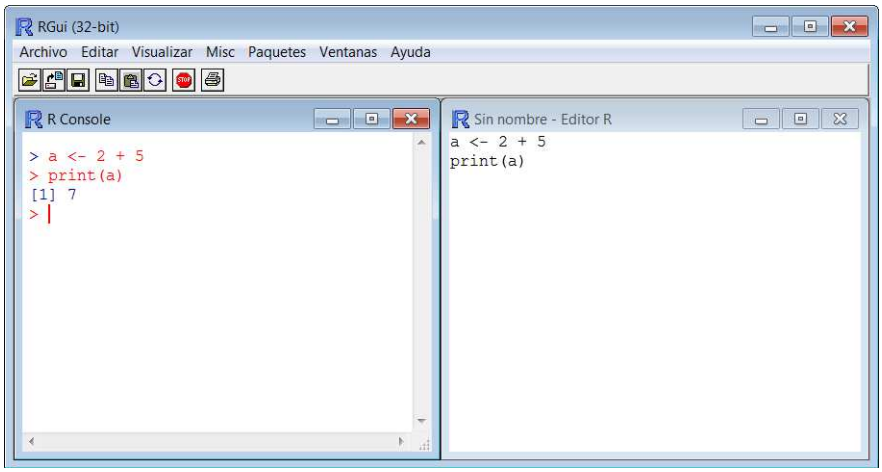


Figura 2

Fixeu-vos que R es basa en l'assignació d'objectes. Aquí veiem com el resultat de l'operació $2 + 5$ l'hem assignat a un objecte al qual hem assignat el nom de `a`.

- En un document de text a part, que un cop acabat es copia i s'enganxa en la consola. El resultat serà el mateix que el de la finestra esquerra de la figura 2, és a dir, apareixeran les instruccions i els resultats. Si triem aquesta opció, qualsevol editor de textos servirà, encara que aquí recomanem l'editor *gedit*, ja que destaca la sintaxi de les instruccions.

gedit

gedit és un editor de textos compatible amb Linux, Mac OS X i Windows. Està dissenyat com un editor de textos de propòsit general, i emfatitza la simplicitat i facilitat d'ús. A més, inclou eines per a l'edició de codi font i de textos estructurats. Es pot descarregar lliurement des de l'adreça *projects.gnome.org/gedit*

1.4. Gestió de paquets

El programa R és modular, és a dir, es basa en diferent mòduls o paquets (*packages*) que els usuaris han elaborat. Aquests paquets són unitats específiques que proporcionen funcions i dades per a posar en pràctica anàlisis específiques. Actualment hi ha tants paquets que seria inviable carregar-los tots simultàniament en una sessió per motius de memòria. La llista completa de paquets es troba en la pàgina del *CRAN*, en l'apartat *Packages*. Allà podrem visualitzar una llista dels més de 4.000 paquets dis-

ponibles per ordre alfabètic amb una breu descripció. És molt útil saber que tenim a la nostra disposició una llista per categories: matemàtiques, estadística, econometria, etc. Accedint a cada paquet obtindrem més informació sobre aquest, i també diferents manuals d'ús amb les funcionalitats que inclouen.

En aquest sentit, quan s'obre una sessió d'R les funcionalitats bàsiques (denominades *base*) es carreguen automàticament, i si l'usuari està interessat en una metodologia específica ha d'instal·lar el paquet corresponent i carregar-lo. Això ho podem fer des del menú de la consola d'R que es mostra en la figura 1, en l'opció *Paquetes*. Aquesta opció ens permet seleccionar un paquet de la llista que apareix, instal·lar-lo i/o carregar-lo. Una altra opció és fer això manualment utilitzant la consola. Si sabem el nom del paquet, per a instal·lar-lo introduirem:

Abans d'intentar implementar nosaltres mateixos un procediment numèric o estadístic complex, val la pena comprovar si algú ja ha fet la feina. Per això mateix, és molt recomanable visitar el repositori de paquets d'R (http://cran.r-project.org/), en el qual podem buscar paquets per ordre alfabètic o per categories.

```
> install.packages("paquete")
```

Un cop instal·lat, al principi de cada sessió en què necessitem aquest paquet serà suficient introduir:

Un cop s'ha instal·lat un paquet, aquest ja queda emmagatzemat en el nostre ordinador, de manera que no cal tornar-lo a instal·lar. Només l'haurem de carregar en les sessions en què el necessitem.

```
> library(paquete)
```

Una anàlisi ben planificada ha d'incloure al principi una llista de tots els paquets que s'han de carregar per a fer les anàlisis que volem.

1.5. Ajuda!

Per a un usuari novell, la interfície d'R, la sintaxi i en general el mode d'ús pot resultar complex, especialment en comparació amb altres programes estadístics. Això no obstant, R disposa de molts manuals i eines de suport a l'abast dels usuaris. Per començar, en la pàgina del *CRAN* des de la qual hem descarregat l'arxiu d'instal·lació hi ha una secció anomenada *Manuals*. Si hi accedim tindrem accés als manuals oficials d'R, el primer dels quals és *An Introduction to R*, un manual completíssim molt ben estructurat que cobreix tots els aspectes elementals del programa. Si volem manuals en altres idiomes, en l'enllaç *contributed documentation* disposem d'una llista ordenada per idiomes, entre els quals el castellà.

A més, hi ha la possibilitat d'obtenir ajuda específica per a cada paquet i funció. Simplement cal introduir en la consola la instrucció `help()`, amb el nom del paquet o la funció sobre la qual necessitem ajuda entre els símbols de parèntesis, i se'ns obrirà una finestra amb una pàgina completa d'ajuda amb descripcions i exemples.

Compte amb les majúscules!

Cal tenir en compte que el llenguatge d'R distingeix entre majúscules i minúscules. D'aquesta manera, la instrucció `HELP()` resultaria en un missatge d'error.

2. Sintaxi i programació

2.1. Tipus de dades i objectes

És important tenir en compte que R interpreta de la mateixa manera les cometes simples 'x' i les dobles "x". Per tant, podem introduir les cadenes de text de qualsevol de les dues maneres.

R és un llenguatge orientat a objectes o estructures de dades diferenciades amb característiques específiques. Els tres tipus bàsics de dades són:

1) *Dades numèriques:* nombres reals i complexos $(-1, 2, 3)$.

2) *Caràcters:* cadenes de text representades entre cometes ("x", "y", "z").

3) *Dades lògiques:* les que únicament prenen els valors vertader o fals ($TRUE$, $FALSE$).

Aquesta orientació a objectes fa indispensable que coneguem els diferents tipus d'objectes amb què podem treballar, ja que cada un té característiques úniques. Això no obstant, com veurem en aquest mòdul, R ens permet que en transformem alguns en d'altres (per exemple, que convertim vectors en matrius i viceversa). Les classes principals d'objectes estan enumerades i definides en la taula 1.

Taula 1. Classes d'objectes

Objecte	Descripció
Data frame	Estructures de dades bidimensionals, en què es recullen diferents variables per columnes.
Vector	Col·lecció ordenada de dades amb una longitud determinada.
Array	Conjunt de dades indexat amb dos o més índexs, caracteritzat per tenir dimensió.
Matrix	Cas particular d'*array* quan hi ha dos índexs, és a dir, dues dimensions (files i columnes).
List	Objecte que recull diferents tipus d'elements (components), que poden ser de classes diferents.
Function	Peça de codi que pot ser un algorisme, una funció matemàtica o una estructura lògica.
Factor	Vector que especifica una classificació discreta dels elements d'un altre vector de la mateixa longitud.

A continuació aprofundirem en cada una d'aquestes estructures.

2.2. Vectors

2.2.1. Emmagatzemar dades en vectors

Hi ha dues maneres d'introduir la informació en el programa. Si introduïm informació mitjançant expressions sense assignar-les a cap objecte, estem utilitzant el programa

com una simple calculadora. En canvi, si l'emmagatzemem i hi donem un nom estem desant el resultat per a utilitzar-lo posteriorment. Aquestes dues funcionalitats bàsiques queden explicades a continuació:

1) *Expressió.* La informació s'introdueix en la consola, R l'avalua i la mostra en pantalla, però no l'emmagatzema en la memòria.

```
> 4 + 3
[1] 7
```

2) *Assignació.* En aquest cas, l'expressió s'avalua però no es mostra, i s'emmagatzema amb el nom que escrivim abans del símbol d'assignació <−. Per a veure'n el valor en pantalla, haurem d'introduir el nom que hem donat al vector, que és l'estructura de dades més simple[1]. La manera més immediata d'introduir els valors en un vector és amb la instrucció c, que significa *concatenar*.

Qüestió de memòria

Quan assignem un nom a un objecte, aquest queda desat en la memòria temporal. Això és, podem recuperar la informació d'aquests objectes sempre que no tanquem la sessió d'R.

```
> a <- c(1,2,3,4,5,6)
> a
[1] 1 2 3 4 5 6
```

Si introduïm una assignació entre parèntesis, a més de quedar desada en la memòria temporal també es mostrarà en pantalla.

```
> (b <- c(3,6,9))
[1] 3 6 9
```

Un cop tenim uns quants vectors creats, els podem combinar d'una manera molt flexible. Vegem-ne un exemple:

L'espai no existeix

En el llenguatge d'R els espais no s'interpreten. És a dir, les expressions `a <- 6` i `a<-6` són anàlogues, i no hi ha cap diferència entre elles.

```
> a <- c(1,2,3)
> b <- c(7,8,9)
> ab <- c(a,4,5,6,b)
> ab
[1] 1 2 3 4 5 6 7 8 9
```

2.2.2. Operacions bàsiques

En la taula 2 es recullen les operacions aritmètiques bàsiques prenent com a exemple els vectors v1 i v2. Aquestes operacions estan vectoritzades, és a dir, s'apliquen element a element.

[1] La funció print també mostra el resultat en pantalla.

Taula 2. Operacions aritmètiques bàsiques (I)

```
v1 <- c(1,2,3)
v2 <- c(2,4,6)
```

Descripció	Instruccions	Resultat
Concatenació	`c(v1,v2)`	`1,2,3,4,5,6`
Suma	`v1+v2`	`3,6,9`
Resta	`v1-v2`	`-1,-2,-3`
Multiplicació	`v1*v2`	`2,8,18`
Divisió	`v1/v2`	`0.5,0.5,0.5`
Potència	`v1^v2`	`1,16,729`

Cal assenyalar que en R les comes queden reservades per a la separació dels diferents valors o components d'un mateix objecte o bé per als diferents paràmetres d'una funció.

Quan s'intenten fer operacions matemàtiques no definides, com per exemple l'arrel d'un nombre negatiu, apareixerà el valor `NaN`, que significa *Not a Number*. A més, els valors ± `Inf.` apareixeran quan el resultat tendeixi a $\pm\infty$. Vegem més operacions matemàtiques elementals aplicades element a element en la taula 3, prenent com a exemple el vector `v3`:

Taula 3. Operacions aritmètiques bàsiques (II)

```
v3 <- c(-1,0,1,2)
```

Descripció	Instruccions	Resultat
Arrel quadrada	`sqrt(v3)`	`NaN 0.00 1.00 1.41`
Valor absolut	`abs(v3)`	`1 0 1 2`
Exponencial	`exp(v3)`	`0.36 1.00 2.71 7.38`
Log. base *e*	`log(v3)`	`NaN -Inf 0.00 0.69`
Log. base 10	`log10(v3)`	`NaN -Inf 0.00 0.30`
Factorial (!)	`factorial(v3)`	`NaN 1 1 2`

Taula 3

És fonamental tenir en compte que, en R, els decimals s'expressen amb un punt, i no una coma com es fa a Espanya. Confusions en aquest sentit poden provocar errors.

Amb R podem fer les operacions estadístiques bàsiques. En la taula 4 es recullen les més usuals, prenent com a exemple els vectors `v4` i `v5`.

Taula 4. Operacions estadístiques bàsiques

```
v4 <- c(-2,3,5,0,0,3)
v5 <- c(20,14,51,76,21,30)
```

Descripció	Instruccions	Resultat
Longitud	`length(v4)`	`6`
Màxim	`max(v4)`	`5`
Mínim	`min(v4)`	`-2`
Suma	`sum(v4)`	`9`
Producte	`prod(v4)`	`0`
Mitjana	`mean(v4)`	`1.5`
Mediana	`median(v4)`	`1.5`
Desviació estàndard	`sd(v4)`	`2.58`
Variància	`var(v4)`	`6.7`
Covariància	`cov(v4,v5)`	`5.8`
Correlació	`cor(v4,v5)`	`0.09`
Producte escalar	`sum(v4,v5)`	`347`

2.2.3. Seqüències

Una eina molt útil és la creació de seqüències, que es pot fer de diferents maneres, entre les quals hi ha les funcions `seq` (*sequence*), `rev` (*reverse*) i `rep` (*repeat*). La taula 5 ho il·lustra amb alguns exemples, partint del vector `v6`:

Taula 5. Seqüències i repeticions

```
v6 <- c(1,2,3)
```

Instruccions	Resultat
`rep(v6,2)`	`1 2 3 1 2 3`
`rep(v6,each=2)`	`1 1 2 2 3 3`
`rep(v6,each=2,times=2)`	`1 1 2 2 3 3 1 1 2 2 3 3`
`rep(v6,c(2,3,4))`	`1 1 2 2 2 3 3 3 3`
`1:10`	`1 2 3 4 5 6 7 8 9 10`
`seq(1,10)`	`1 2 3 4 5 6 7 8 9 10`
`10:1`	`10 9 8 7 6 5 4 3 2 1`
`seq(10,1)`	`10 9 8 7 6 5 4 3 2 1`
`seq(4,length=8)`	`4 5 6 7 8 9 10 11`
`seq(0,40,by=5)`	`0 5 10 15 20 25 30 35 40`
`seq(0,1,length=4)`	`0.00 0.33 0.66 1.00`
`seq(from=2,to=8,by=3)`	`2 5 8`
`rev(1:5)`	`5 4 3 2 1`
`2*1:4`	`2 4 6 8`

Taula 5

El domini d'aquestes instruccions ens estalviarà moltes línies de programació, i ens permetrà programar procediments numèrics complexos d'una manera simple, compacta i elegant.

2.2.4. Operadors lògics

R també ens permet operar amb vectors lògics que poden prendre els valors `TRUE` (vertader) i `FALSE` (fals). Els operadors principals es mostren en la taula 6:

Taula 6. Operadors lògics

Descripció	Operador
Més petit	`<`
Més gran	`>`
Igual	`==`
No igual	`!=`
Més petit o igual	`<=`
Més gran o igual	`>=`
Intersecció	`a & b`
Unió	`a|b`
Negació	`!a`

Taula 6

Podem interpretar l'operador lògic intersecció com a a i b, mentre que la unió seria interpretable com a a o b.

Vegem alguns exemples de l'ús d'aquests operadors:

```
> a <- c(-2,-1,0,1,2)
> a > 2
[1] FALSE FALSE FALSE FALSE FALSE
> a <= -1
[1] TRUE TRUE FALSE FALSE FALSE
> a > 0 & abs(a) == 2
[1] FALSE FALSE FALSE FALSE TRUE
> a > 0 | abs(a) == 2
[1] TRUE FALSE FALSE TRUE TRUE
```

Els símbols = i ==

Quan fem servir l'operador lògic igualtat, és important tenir en ment que s'expressa amb un doble símbol d'igualtat (==), diferent del símbol d'igualtat simple utilitzat en assignacions.

2.2.5. Indexació

Per a accedir als elements d'un vector hi ha la possibilitat d'*indexar*, és a dir, accedir als valors d'un vector especificant-lo en una expressió entre claudàtors `[]`. Això ho fem indicant la posició dels elements o bé fent servir operadors lògics. Un exemple considerant el vector `v7` es recull en la taula 7:

Taula 7. Indexació de vectors

`v7 <- c(-3,-2,-1,0,1,2,3)` o de manera més compacta `v7 <- c(-3:3)`

Expressió	Resultat	Descripció
`v7[1]`	`-3`	Primer element
`v7[2:4]`	`-2 -1 0`	Del segon al quart element
`v7[-(2:length(v7))]`	`-3`	Tots menys aquells des del segon fins a l'últim element
`v7[c(2,4,6)]`	`-2 0 2`	Elements segon, quart i sisè
`v7[v7>0]`	`1 2 3`	Elements més grans que 0
`v7[v7!=0]`	`-3 -2 -1 1 2 3`	Elements diferents de 0
`v7[abs(v7)==3]`	`-3 3`	Elements amb valor absolut 3
`v7[v7>0 & v7!=2]`	`1 3`	Elements positius **i** que no siguin 3
`v7[v7<0 | v7==3]`	`-3 -2 -1 3`	Elements negatius **o** que siguin 3

A més, la funció `which` ens permet extreure la posició ordinal dels elements descrits en la condició que s'ha introduït. En l'exemple següent, en el primer cas els elements vector `a` iguals a zero són el primer, tercer i sisè. En el segon cas, l'únic element més gran que zero (`a>0`) i més petit o igual que dos (`a<=2`) és el quart.

```
> a <- c(0,3,0,1,3,0)
> which(a==0)
[1] 1 3 6
> which(a>0 & a<=2)
[1] 4
```

De la mateixa manera, les funcions `which.min` i `which.max` ens donen la posició dels valors mínims i màxims del vector, respectivament.

2.3. Matrius

2.3.1. Els objectes `array` i `matrix`

Sovint tindrem la necessitat d'emmagatzemar les dades en estructures més complexes que simples vectors. Això és, podem necessitar tenir una estructura de files i columnes, i poder fer operacions matricials. En R, hi ha dues estructures (tipus d'objectes) que satisfan aquesta necessitat:

1) Variable indexada (`array`). Estructura que organitza les dades en k dimensions, i és molt útil per a tractar amb dades amb diverses classificacions simultànies. Per exemple, la variable v_{ijt} pot representar les vendes d'una empresa en què i és el sector, j la província i t l'any, i s'emmagatzemaria en una estructura `array` amb dimensió $k = 3$.

2) Matriu (`matrix`). Es tracta d'un cas particular d'*array* quan hi ha dues dimensions ($k = 2$), essent la primera dimensió les files i la segona dimensió, les columnes.

En molts casos no hi ha diferència entre els dos tipus d'estructures. Això no obstant, algunes operacions com la transposició i alguns tipus d'operacions algebraiques estan reservades només a les matrius. Per simplicitat, a continuació ens centrarem en la creació i manipulació d'estructures bidimensionals (matrius). La manera més immediata de crear una matriu és assignant una dimensió a un vector, ja que, com hem vist, els vectors tenen longitud i no dimensió:

```
> a <- 1:8
> a
[1] 1 2 3 4 5 6 7 8
> length(a)
[1] 8
> dim(a)
NULL
```

El resultat `NULL`

Sempre que un objecte estigui buit apareixerà el resultat `NULL`. En aquest cas, en ser `a` un vector i no una matriu, la seva dimensió no existirà.

2.3.2. Creació de matrius

Suposem que volem convertir el vector `a` en una matriu de dimensió 4×2, és a dir, una estructura amb 4 files i 2 columnes. En aquest cas, hi podem assignar una dimensió:

```
> dim(a)<-c(4,2)
> a
```

```
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
> dim(a)
[1] 4 2
```

És important destacar que, per defecte, R sempre comença a col·locar els valors de dalt a baix i d'esquerra a dreta, com es pot observar en l'exemple anterior.

Una manera més directa de crear matrius és amb la funció `matrix`. Aquesta funció té tres arguments: el conjunt de valors que compondran la matriu, el nombre de files i el nombre de columnes:

```
> matrix(1:8,2,4)
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
```

En cas de necessitar una estructura de dades amb més de dues dimensions, caldrà fer servir la funció `array`. La diferència principal respecte a la funció `matrix` és que el nombre de dimensions s'introdueix mitjançant un vector. A tall d'exemple, crearem una estructura $2 \times 2 \times 2$:

```
> array(1:4,c(2,2,2))

, , 1
     [,1] [,2]
[1,]    1    3
[2,]    2    4

, , 2
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

El reciclatge és important

Fixeu-vos que, quan s'acaben de col·locar tots els elements del vector `1:4` en l'estructura resultant i aquesta encara no està completa, es torna a començar fins que l'estructura està completa. Aquesta funcionalitat es denomina *reciclatge*.

Una característica interessant de la construcció de matrius és el **reciclatge**, que consisteix en el fet que si el nombre total d'elements de la matriu (files per columnes) és superior al nombre de valors inicials, aquests es repeteixen fins a completar la matriu. En l'exemple anterior, hem volgut transformar un vector amb 4 elements en una estructura amb $2 \times 2 \times 2 = 8$ elements, això és, el doble. Per això, R ha hagut de repetir els elements del vector inicial dues vegades per a completar l'estructura final. Vegem un altre exemple de reciclatge:

```
> matrix(1:3,3,3)
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    2    2    2
[3,]    3    3    3
```

A més hi ha l'opció que la matriu es construeixi invertint l'ordre, és a dir, d'esquerra a dreta i de dalt a baix. S'aconsegueix afegint una opció al final de la funció:

```
> matrix(1:8,2,4,byrow=TRUE)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
```

Per files o columnes?

Utilitzant la funció `byrow`, invertim l'ordre de la construcció de matrius, començant per files en lloc de per columnes, que és l'opció per defecte.

També és possible especificar únicament el nombre de files i/o columnes de la matriu que s'ha de construir:

```
> matrix(1:6,nrow=2)
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> matrix(1:4,ncol=2)
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

És molt fàcil combinar escalars, vectors i matrius. La funció `rbind` uneix diverses estructures per files, mentre que `cbind` fa el mateix per columnes. Vegem com es fa:

```
> a<-matrix(0,2,2)
> a
     [,1] [,2]
[1,]    0    0
[2,]    0    0
> b<-matrix(1,1,2)
> b
     [,1] [,2]
[1,]    1    1

> rbind(a,b)
     [,1] [,2]
[1,]    0    0
[2,]    0    0
[3,]    1    1
```

Com hem vist, aquestes dues funcions ens ofereixen moltes possibilitats:

```
> cbind(1,1:4)
     [,1] [,2]
[1,]    1    1
[2,]    1    2
[3,]    1    3
[4,]    1    4
```

Un altre exemple de reciclatge

En aquest cas, veiem que unir un escalar i un vector fa que el valor de l'escalar es repeteixi tantes vegades com la longitud del vector. Això fa que l'estructura resultant tingui dues columnes i tantes files com elements del vector.

La matriu identitat es crea amb la funció `diag`, especificant el nombre de files i columnes:

```
> diag(2)
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

La matriu identitat

Recordem que la matriu identitat sempre és quadrada, això és, té el mateix nombre de files i de columnes.

A més, la funció `diag` també es pot fer servir per a accedir als elements de la diagonal principal d'una matriu:

```
> a<-matrix(5*1:4,2,2)
> a
     [,1] [,2]
[1,]    5   15
[2,]   10   20
> diag(a)
[1]  5 20
```

2.3.3. Submatrius i indexació

De manera anàloga al cas dels vectors, també és possible accedir a subconjunts de matrius mitjançant claudàtors `[]`. Aquests es fan servir amb tres propòsits:

1) Tenir accés a files i columnes específiques.

2) Crear submatrius.

3) Modificar directament parts de una matriu.

Això ens ofereix una gran potencialitat per a fer tota mena d'operacions amb matrius, i fer que aquestes siguin fàcilment replicables, modificables i combinables. A partir de la matriu `a`, la taula 8 mostra exemples de creació de submatrius.

```
> a <- matrix(1:9*3,3,3)
> a
     [,1] [,2] [,3]
[1,]    3   12   21
[2,]    6   15   24
[3,]    9   18   27
```

Taula 8. Indexació de matrius

Expressió	Resultat	Descripció
`a[,3]`	`21 24 27`	Totes les files, 3a. columna
`a[2,]`	`6 15 24`	2a. fila, totes les columnes
`a[-1,-3]`	`6 15` `9 18`	Totes les files menys la 1a. i totes les columnes menys la 3a.
`a[2:3,1]`	`6 9`	De la 2a. a la 3a. fila, primera columna

És interesant comprovar que les submatrius creades, en cas de tenir una sola fila o columna, es degraden a vectors, amb la qual cosa perden la dimensió[2]. Perquè aquestes submatrius continuïn tenint dimensió, ho hem d'especificar desactivant l'opció `drop`:

```
> a
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> b <- a[,1,drop=FALSE]
> b
     [,1]
[1,]    1
[2,]    2
> dim(b)
[1] 2 1
```

Vegem ara com es faria per a introduir elements nous en una matriu. Suposem, per exemple, que volem substituir la primera columna de valors per una columna de zeros:

```
> a<-matrix(1:8,2,4)
> a[,1]<-0
> a
     [,1] [,2] [,3] [,4]
[1,]    0    3    5    7
[2,]    0    4    6    8
```

Modificant una matriu

En aquest cas, hem utilitzat `a[,1]<-0` per a transformar la primera columna, inicialment composta pels elements 1 i 2, en una columna amb dos zeros.

[2] En anglès, aquest fet es denomina *deprecation*.

2.3.4. Operacions matricials bàsiques

R ofereix moltes possibilitats per a aplicar operacions i funcions a vectors i matrius[3]. En la taula 9 s'inclouen les funcions bàsiques aplicables a matrius:

Taula 9. Funcions amb matrius

Descripció	Funció
Dimensions	`dim`
Transposada	`t`
Determinant	`det`
Inversa	`solve`
Valors i vectors propis	`eigen`
Valors singulars	`svd`
Factorització de Choleski	`chol`
Factorització QR	`qr`
Rang	`rank`
Nombre de files	`nrow`
Nombre de columnes	`ncol`
Suma de les files	`rowSums`
Suma de les columnes	`colSums`
Mitjana de les files	`rowMeans`
Mitjana de les columnes	`colMeans`

Per acabar l'apartat dedicat a matrius, val la pena esmentar que hi ha algunes funcions que permeten, d'una banda, identificar objectes i, de l'altra, transformar-los en objectes d'altres classes. Aquest primer exemple és il·lustratiu:

```
> a <- c(1,3,6)
> is.matrix(a)
[1] FALSE
> is.vector(a)
[1] TRUE
> class(a)
[1] 'numeric'
```

La família de funcions `is.` respon a la pregunta de si es tracta d'un objecte determinat per un valor lògic, mentre que la funció `class` torna la classe de l'objecte. Finalment, la família de funcions `as.` es fa servir per a transformar un tipus d'objecte en un altre. L'exemple següent ho mostra:

```
> (A <- matrix(1,2,2))
     [,1] [,2]
[1,]    1    1
[2,]    1    1
> (a <- as.vector(A))
[1] 1 1 1 1
```

Convertir matrius en vectors

En aquest cas, hem aconseguit transformar la matriu `A` en el vector `a`.

[3] El mòdul 2 d'aquest manual inclou un apartat d'àlgebra vectorial i matricial dedicat a operacions algebraiques específiques.

2.4. Funcions

Una eina molt útil i poderosa és l'ordre `function`, que també és una classe d'objecte. Podem crear des de simples funcions matemàtiques a algorismes complexos amb els quals podem fer multitud de càlculs d'una manera estructurada i sintètica. La forma de funció més senzilla té l'estructura següent:

```
nom <- function (arguments) expressió
```

Com a exemple, suposem que volem estudiar la funció matemàtica $y = x^3$ i avaluar-la en el vector de valors inicials $x_0 = (-4, -3, \ldots, 3, 4)$. Els farem creant la nostra pròpia funció `cubic` i avaluant-la en el vector de valors inicials `x0`:

```
> cubic <- function(x) x^3
> x0 <- (-4:4)
> cubic(x0)
[1] -64 -27  -8  -1   0   1   8  27  64
```

Un exemple d'una funció amb més d'un argument és el de la fórmula financera de l'interès simple $C_f = C_i\,(1 + rt)$. Segons aquesta fórmula, el capital final (C_f) que s'ha de retornar en una operació financera amb un import inicial C_i depèn de la duració de l'operació en anys (t) i del tipus d'interès en tant per un (r). Quin import final resulta d'una operació a nou mesos ($t = 0,75$) amb un tipus nominal del 5% ($r = 0,05$) i un capital inicial $C_i = 1000$?

```
> op.fin <- function(Ci,r,t) Ci*(1+r*t)
> op.fin(1000,0.05,0.75)
[1] 1037.5
```

Per a funcions més complexes sovint és necessària més d'una línia per a introduir les fórmules. En aquest cas, suposant que necessitem n línies, l'estructura serà la següent:

```
nom <- function(arguments){
expressió 1
...
expressió n
return(resultat)
}
```

A fi d'il·lustrar aquest tipus d'estructura crearem una funció per calcular la covariància mostral. Això és una mica innecessari, podríem fer servir la funció `cov` que ens

proporciona R. De tota manera, ho fem perquè serveixi d'exemple de funcions més elaborades:

$$Cov(X,Y) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})$$

```
> mi.covar <- function(x,y){
+    N <- length(x)
+    demean.x <- x-mean(x)
+    demean.y <- y-mean(y)
+    sumat <- sum(demean.x*demean.y)
+    return(sumat/(N-1))
+ }
```

Funcions multiusos

Crear funcions pròpies ens serà molt útil per a dur a terme un conjunt de càlculs que hàgim de fer servir més d'una vegada, de manera estructurada i ordenada.

2.5. Cicles i condicionals

R ofereix una sèrie d'eines per a fer assignacions múltiples i condicionals. `ifelse` permet fer operacions element per element a un vector subjecte a una condició. Suposem que volem aplicar la transformació següent a un vector:

$$f(x) = \begin{cases} log(x) & \text{si} \quad x > 0 \\ 0 & \text{si} \quad x \leq 0 \end{cases}$$

```
> a <- c(-1,0,2,5)
> ifelse(a>0,log(a),0)
[1] 0.0000000 0.0000000 0.6931472 1.6094379
```

Cicles i condicionals

Per a fer servir eficientment cicles i condicionals és fonamental dominar els operadors lògics, explicats en aquest mòdul.

Un dels components fonamentals de qualsevol llenguatge de programació és l'ús d'estructures iteratives, és a dir, de funcions que repeteixen una o més expressions en un cicle. L'ordre principal que fa aquesta funció és `for`. L'estructura general d'aquesta funció és, per a un cicle de *n* iteracions:

```
for (i in 1:n){
expressió(ns)
...
}
```

Un exercici senzill per a il·lustrar un cicle és la suma acumulativa. Partint d'un vector $\vec{v} = (v_1, \cdots, v_n)$, crearem un vector $\vec{s} = (s_1, \cdots, s_n)$ en què $s_j = \sum_{i=1}^{j} v_i$. Abans d'implementar el cicle crearem un vector s buit que anirem completant iterativament:

Cicles iteratius

Utilitzarem els cicles quan vulguem treballar amb bucles o fer càlculs iteratius.

```
> v <- 1:10
> n <- length(v)
> s <- rep(0,n)

> for (i in 1:n){
+     p <- v[1:i]
+     s[i] <- sum(p)
+ }

> print(s)
 [1]  1  3  6 10 15 21 28 36 45 55
```

Òbviament, en una anàlisi real no farem servir aquest càlcul iteratiu, ja que la funció implementada `cumsum` ja ens ofereix aquesta funcionalitat d'una manera més eficient.

2.6. La família `apply`

En la pràctica, és important no fer un ús excessiu dels cicles iteratius, ja que no són eficients quant a l'ús de la memòria i poden alentir el temps d'anàlisi que es requereixi. Per això, la norma és que es crearan estructures iteratives sempre que no hi hagi alternatives, i el grup de funcions `apply` n'és una.

Per il·lustrar aquest concepte, suposem que tenim una matriu A en què les files corresponen a observacions i les columnes són les variables X, Y i Z:

$$A = \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{pmatrix}$$

La funció `apply` ens permet aplicar qualsevol operació a les files i a les columnes, independentment. Suposem que volem aplicar l'operació $\sqrt{x_i^2 + y_i^2 + z_i^2}$ a cada observació (és a dir, a cada fila). El primer pas serà definir aquesta operació (que anomenarem `hipot`), i el segon aplicar-la a les files de la matriu A:

La funció `apply`

La instrucció `apply(A,1,hipot)` es llegeix com *aplica a la dimensió 1 (files) de la matriu A l'operació* `hipot`. El resultat serà un vector amb n elements.

```
> hipot <- function(x) sqrt(sum(x*x))
> apply(A,1,hipot)
```

Si volguéssim un vector amb la mitjana aritmètica de cada columna, és a dir $(\bar{x}, \bar{y}, \bar{z})$, hauríem d'introduir:

```
> apply(A,2,mean)
```

Per a explicar la funcionalitat de la funció de la mateixa família `tapply`, és necessari introduir els **factors**, que són una classe d'objecte que estableix una classificació discreta d'una o més variables. Suposem que estem estudiant l'estatura dels alumnes d'una escola, per a la qual cosa disposem de dos vectors: un amb l'estatura de cada estudiant i un altre amb el gènere (masculí *m* i femení *f*):

```
> alçada <- c(131,125,126,140,152,119)
> genere <- c('m','m','f','m','f','f')
```

Introducció de caràcters

A l'hora d'introduir un vector de caràcters, com `gènere`, els valors del vector han d'anar entre cometes, per a la qual cosa són vàlids els símbols `"` i `'` indistintament.

En aquest cas és convenient convertir el vector `genere` en un factor amb dues categories: masculina i femenina. Per a això, fem servir la funció `factor` per a obtenir un vector de factors amb les categories $f = 1$ i $m = 2$, que anomenarem `f.genere`. Vegem com són objectes de diferent classe:

```
> f.genere <- factor(genere)
> class(genere)
[1] "character"
> class(f.genere)
[1] "factor"
```

La instrucció `levels` torna les categories que el vector inclou:

```
> levels(f.genere)
[1] 'f' 'm'

> cbind(f.genere,alçada)
     f.genere alçada
[1,]        2    131
[2,]        2    125
[3,]        1    126
[4,]        2    140
[5,]        1    152
[6,]        1    119
```

Una anàlisi estadística de la variable *alçada* haurà de diferenciar entre *f* i *m*, per a la qual cosa es fa servir la funció `tapply`. Calcularem la mitjana aritmètica i la desviació estàndard de l'alçada diferenciant per gènere:

```
> tapply(alçada,f.genere,mean)
       f        m
132.3333 132.0000

> tapply(alçada,f.genere,sd)
        f         m
17.387735  7.549834
```

La funció `tapply`

Utilitzarem la funció `tapply` quan vulguem aplicar un càlcul a una variable diferenciant per grups o segments, això és, segons les categories d'un factor.

2.7. Bases de dades

Una base de dades (*data frame*) és similar a una matriu, ja que és una estructura bidimensional en què les files són les observacions i les columnes són les variables, cada una amb un nom específic. És molt recomanable tenir les variables emmagatzemades en aquest format per diversos motius:

- És molt fàcil extreure subgrups (*subsets*) de les variables utilitzant operadors lògics.
- Moltes funcions estadístiques, com els estimadors economètrics, només admeten dades en aquest format.
- És possible extreure variables de la base de dades mitjançant la funció `attach`.

Hi ha diferents maneres de crear una base de dades. La primera és important dades externes d'un arxiu de text net[4]. La segona és agrupant vectors existents, com mostrem en l'exemple següent (continuació de l'anterior):

```
> g <- c('m','m','f','m','f','f')   # Gènere
> v1 <- c(131,125,126,140,152,119)  # Alçada
> v2 <- c(48,53,45,40,49,50)        # Pes
>
> dades <- data.frame(genere=g,alçada=v1,pes=v2)
> dades
  genere alçada  pes
1      m    131   48
2      m    125   53
3      f    126   45
4      m    140   40
5      f    152   49
6      f    119   50
```

Incorporar comentaris mitjançant el símbol #

Molt sovint ens interessarà introduir aclariments i comentaris en les nostres línies de codi per a entendre'l millor. Perquè R no els interpreti i ens doni un missatge d'error, els introduirem després del símbol #.

Com veiem, en la primera columna s'identifica cada element de la mostra, mentre que cada columna està encapçalada pel nom de la variable. És important destacar que, alternativament, la funció `data.frame` també admet una matriu.

[4] En el mòdul dedicat a l'anàlisi estadística descriptiva s'explica aquest procediment amb detall.

Suposem que volem fer una anàlisi només dels nois (*m*) que pesen menys de 50 kilos. Per a això, és molt pràctic crear una altra base de dades que sigui un subgrup de la base de dades original. Anomenarem aquest subgrup `dades2`, i com veiem només inclou dues observacions, les que satisfan les condicions establertes:

```
> dades2<-subset(dades,genere=="m" & pes<50)
> dades2
  sexe alçada  pes
1    m    131   48
4    m    140   40
```

La funció `subset`

La funció `subset` ens permet crear una nova base de dades que contingui una selecció condicional d'observacions de la base de dades original.

Si volem operar amb les variables incloses en una base de dades, les hem de referenciar amb el símbol $, ja que les variables no són en l'espai de treball individualment, només com a elements d'una base de dades. Per exemple, si necessitem la correlació entre l'alçada i el pes, hem d'introduir la instrucció següent:

```
> cor(dades$alçada,dades$pes)
[1] -0.3202351
```

Això pot arribar a ser força pesat. Per això, si la intenció és fer operacions amb les variables de la base de dades, és recomanable *abocar-les* prèviament en l'espai de treball mitjançant la funció `attach`, amb la qual cosa ja podrem fer referència a les variables individualment:

```
> attach(dades)
> cor(alçada,pes)
[1] -0.3202351
```

La funció `attach`

Aquesta funció és aplicable quan volem fer operacions i càlculs individuals amb les variables incloses en una base de dades.

2.8. Llistes

Les llistes (*lists*) són comparables a un calaix de sastre, ja que permeten emmagatzemar objectes de classes diferents en una mateixa estructura. Això és molt útil quan una mateixa anàlisi té com a resultat diferents objectes dispars, per exemple vectors o matrius de longituds o dimensions diferents. Suposem que volem emmagatzemar en una estructura la matriu *A*, la seva descripció, la seva inversa i el determinant:

```
> descr <- c('Matriu A')
> A <- matrix(c(3,5,6,1),2,2)
> inv <- solve(A)
> deter <- det(A)
```

Les llistes

Aquesta classe d'objecte ens permetrà agrupar en una sola estructura elements que són, alhora, objectes de classes diferents.

La manera d'emmagatzemar-lo és mitjançant la funció `list`. Per a accedir als diferents elements d'una llista pel seu nom, farem servir el símbol $ de la manera següent:

```
> MatA <- list(descripcio=descr,
+              matriu=A,
+              inversa=inv,
+              deter=deter)

> MatA$descripcio
[1] 'Matriu A'

> MatA$matriu
     [,1] [,2]
[1,]    3    6
[2,]    5    1

> MatA$inversa
            [,1]       [,2]
[1,] -0.03703704  0.2222222
[2,]  0.18518519 -0.1111111

> MatA$deter
[1] -27
```

Els símbols > i + en la consola

Com hem vist, en la consola hem d'introduir cada línia de codi després del símbol >. Això no obstant, algunes instruccions de codi són tan llargues que ocupen més d'una línia, com en el cas de la llista `MatA`. En aquest cas, en la línia següent, R ens mostrarà el símbol + per recordar-nos que la instrucció de la línia prèvia està incompleta.

També es pot accedir als elements d'una llista no pel nom, sinó ordinalment. Per a això farem servir el símbol [[]]. Vegem una comparació de totes dues maneres d'accedir a elements de la llista:

```
> MatA$matriu
     [,1] [,2]
[1,]    3    6
[2,]    5    1
> MatA[[4]]
[1] -27
```

Diferents maneres d'accedir als elements d'una llista

Fixeu-vos que, en aquest exemple, les instruccions `MatA$matriu` i `MatA[[2]]` donarien el mateix resultat.

3. L'extensió R-Commander

3.1. Introducció

Un dels problemes principals als quals ha fet front R des del seu naixement és la important corba d'aprenentatge que l'usuari troba per a poder-lo utilitzar. De fet, això es un problema a l'hora d'implementar aquest programa en els primers cursos d'estadística de grau, ja que a la dificultat d'aprendre el contingut de l'assignatura se suma la complicació del llenguatge d'R. Per sort, aquest programa és tan flexible i mal·leable que s'han creat diverses aplicacions basades en R però que són més fàcils d'utilitzar. Una de les aplicacions amb més èxit ha estat R-Commander, que consisteix en una interfície gràfica d'usuari (GUI, en les sigles en anglès). El seu creador, el professor John Fox, el va desenvolupar amb l'objectiu de crear un programa amb *menús desplegables*, i que fos fàcil d'utilitzar per a estudiants d'estadística. Es pot considerar R-Commander com una alternativa viable als paquets estadístics comercials (com Minitab o SPSS).

RStudio

Per als usuaris d'R avançats, l'extensió RStudio és una eina potent que consisteix en un entorn de desenvolupament integrat (IDE), en el qual es visualitzen el document amb el codi, la finestra de resultats, una llista amb els objectes creats i un espai per als gràfics. En aquest sentit, és similar a l'entorn de treball que ofereix el programa d'anàlisi matemàtica Matlab.

Un altre aspecte destacable d'R-Commander és que és molt útil per als estudiants que aprenen a utilitzar R. Això és així perquè, en efectuar l'anàlisi per mitjà del menú desplegable, el codi en R subjacent també apareix en pantalla juntament amb el resultat, de manera que, al final, l'estudiant acabarà introduint el codi directament.

3.2. Instal·lació

Per a instal·lar R-Commander hem de tenir instal·lada una versió d'R. En general, com per a tots els paquets hi ha dues opcions d'instal·lació:

1) Anar al menú desplegable d'R, seleccionar *Paquetes* i *Instalar paquetes*. Un cop aquí, seleccionem un servidor (per proximitat podríem escollir Spain (Madrid), France (Toulouse) o Portugal), i premem *Ok*. A continuació ens apareixerà una llista de totes les llibreries (paquets o *packages* en anglès) que podem instal·lar, i seleccionem el paquet *Rcmdr*.

2) Com faríem amb qualsevol altre paquet que volguéssim instal·lar, podríem anar directament a la consola d'R i introduir la instrucció `install.packages("Rcmdr")`.

Executar el programa més d'una vegada

Si tanquem el programa R-Commander, tornarem a la consola d'R. Si aleshores volem tornar a carregar R-Commander sense haver de tancar la sessió d'R i tornar-la a obrir, haurem d'introduir la instrucció `Commander()` en la consola d'R.

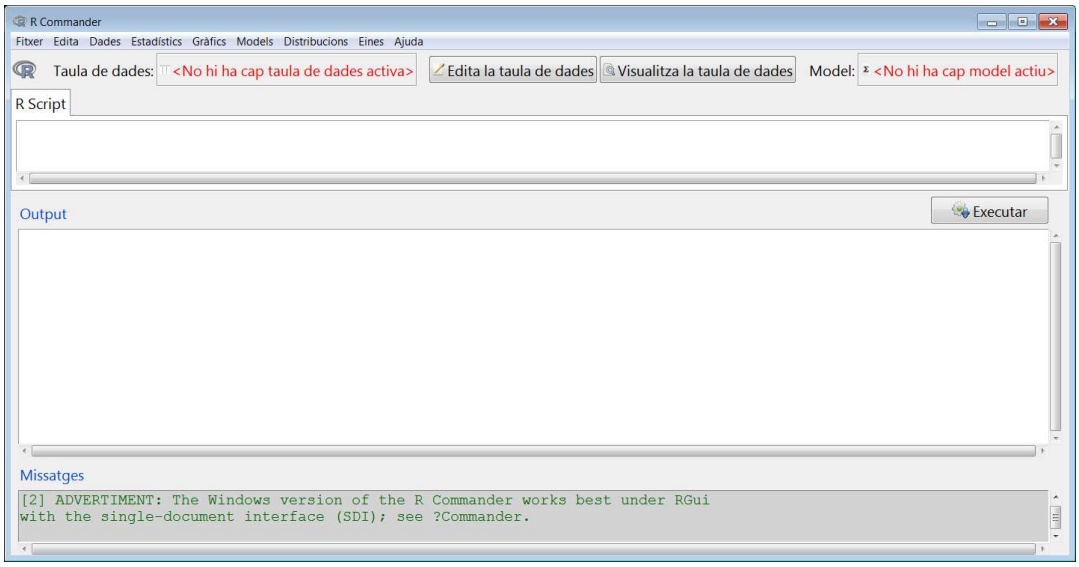
Un cop acabat el procés de descàrrega, la instal·lació es completa introduint en la consola la instrucció `library(Rcmdr)`. Aleshores apareixerà una finestra d'avís que ens indicarà que cal instal·lar alguns paquets addicionals. Premem *Aceptar* i esperem que s'acabi el procés. Si tot ha anat bé, en acabar ens apareixerà en pantalla la finestra

d'R-Commander, a punt per a fer servir. En general, sempre que vulguem utilitzar R-Commander haurem d'introduir la instrucció `library(Rcmdr)` en la consola.

3.3. Components

La finestra principal d'R-Commander es mostra a continuació en la figura 3:

Figura 3. La interfície gràfica R-Commander



Al llarg dels mòduls d'aquest material analitzarem cada un dels elements d'aquest programa. Això no obstant, en aquesta secció es presenta una visió general de tots els components.

3.3.1. Barra de menús

Aquesta barra es troba en la línia superior, i incorpora les funcionalitats principals del programa. A continuació se'n descriuen els elements:

1) Fitxer. Instruccions per a crear, carregar i desar documents amb codi (*scripts*), resultats i espais de treball (*workspaces*), com també per a sortir del programa.

2) Edita. Opcions de tallar, copiar i enganxar, i en general per a editar el contingut de les finestres d'instruccions i de resultats.

3) Dades. Submenús que contenen elements de menú per a importar, llegir i manipular dades i variables.

4) Estadístics. Submenús que contenen elements de menú per a una varietat d'anàlisis estadístiques bàsiques.

5) Gràfics. Elements de menú per a crear gràfics estadístics simples.

6) Models. Elements per a obtenir tota mena d'informació sobre l'estimació de models estadístics: resums numèrics, intervals de confiança i contrastos d'hipòtesi, diagnòstics, anàlisis de residus i gràfics derivats d'un model estadístic.

7) Distribucions. Conté les distribucions principals de probabilitat discretes i contínues, incloent-hi probabilitats, quantils, gràfics i generació de dades aleatòries.

8) Eines. Permet carregar paquets addicionals que no vénen en la distribució bàsica d'R-Commander, i es poden carregar com a extensions si es necessita fer una anàlisi específica. A més, en aquesta part del menú també hi ha les opcions de visualització (mida de lletra, etc.).

9) Ajuda. Permet accedir a diversos documents d'ajuda d'R-Commander molt complets, de lectura recomanada als no iniciats en el programa.

Un element fonamental que s'ha de tenir en compte és que R-Commander només inclou una petitíssima part de les potencialitats d'R. És a dir, R-Commander s'ha dissenyat per a satisfer les necessitats d'estudiants de cursos d'estadística introductoris i intermedis. Els usuaris que necessitin fer anàlisis estadístiques i quantitatives més avançades necessitaran més funcions de les que s'inclouen en la barra de menús.

3.3.2. Barra d'eines

Aquesta barra inclou dos elements bàsics: el conjunt de dades i el model.

1) Taula de dades. La taula inclou una sèrie de dades, distribuïdes en variables per columnes. Hi ha dos botons que permeten *editar* i *visualitzar* la taula de dades. Un aspecte que cal tenir en compte és que es pot treballar amb diferents taules de dades simultàniament, però només hi pot haver *una taula de dades activa* en cada moment. En l'espai de la figura 3 on es llegeix *<No hi ha cap taula de dades activa>* apareixerà el menú desplegable amb les diferents taules de dades existents, entre les quals escollirem, en cada moment, la taula de dades activa que vulguem analitzar.

2) Model. El funcionament d'aquest espai és anàleg a l'anterior. Això és, podem estimar diferents models en cada sessió de treball, però només hi pot haver un model actiu en cada moment, ja que la funcionalitat del menú desplegable s'aplica sempre sobre el model actiu.

3.3.3. Finestres d'instruccions, resultats i missatges

La **finestra d'instruccions** té com a objectiu desplegar les instruccions en codi d'R. És important explicar-ne el funcionament: quan fem una anàlisi mitjançant les opcions disponibles en el menú desplegable, el codi corresponent apareixerà igualment en la finestra d'instruccions, i el resultat apareixerà en la **finestra de resultats**. Ara bé, també és possible introduir les instruccions *directament* en la finestra d'instruccions sense

necessitat d'acudir al menú desplegable, seleccionar les línies introduïdes i prémer el botó **Executar**, situat a la dreta, amb la qual cosa el resultat apareixerà igualment en la finestra de resultats. Això ho farem si coneixem el codi i preferim escriure'l en comptes d'acudir al menú desplegable, però sobretot si volem implementar un càlcul o tècnica que no està disponible en el menú desplegable.

Per acabar, veurem una finestra inferior de **Missatges**, en la qual apareixeran bàsicament missatges d'error que ens informaran de què hem fet de manera incorrecta.

Bibliografia

Gibernans Bàguena, J.; Gil Estallo, À. J.; Rovira Escofet, C. (2009). *Estadística*. Barcelona: Material didàctic UOC.