1 L'entorn estadístic R

R és un entorn de tractament estadístic de dades, construït al voltant d'un llenguatge de programació, dotat d'una interfície gràfica amb capacitat interactiva.

El llenguatge de programació és orientat a objectes i interpretat, anàleg a d'altres llenguatges dels anomenats *de scripting*, com el **Perl**, el **Python**.

La finestra principal del programa permet entrar ordres, que s'escriuen a partir de l'indicador >.

L'interfície gràfica s'integra en la dels sistemes operatius, com **Windows** o **Linux**, i té un menú amb tasques bàsiques. Apart de les comunes a tots els programes, en comentarem algunes d'específiques més en detall.

File Ens anirà bé adreçar **Change dir..** a la carpeta on hi tenim els nostres fitxers. **Source R code** serveix per executar fitxers, típicament amb extensió `.R`, que contenen funcions o altre codi. Qualsevol ordre o llista d'ordres que s'escrigui interactivament a l'indicador d'ordres pot igualment escriure's a un fitxer `.R` i, així, es té per més vegades.

Packages Els *packages* són col·leccions de funcions, jocs de dades i documentació associada. Hi ha *packages* molt generals, que contenen centenars de funcions i d'altres molt especialitzats. Són contribucions de la comunitat d'usuaris de R. **Load package** carrega en memòria les funcions i jocs de dades d'un package ja instal·lat. Prèviament s'ha d'haver instal·lat, sigui en directe, d'un node CRAN, o bé a partir d'un fitxer `.zip` local.

1.1 Variables i operacions

Escalars

En general són nombres reals de punt flotant. Pot operar amb complexos, que s'escriuen com `3 + 2i`, sense espai en blanc (aixi, `3 + 2~i` produeix un missatge d'error).

Hi ha quantitats booleanes o lògiques, amb els valors `TRUE`, `FALSE` i cadenes de caràcters.

Les variables categòriques, qualitatives, es codifiquen com a `factors`, que prenen valors que són etiquetes, que poden ser numèriques o cadenes de caràcters.

Operador d'assignació

En R és possible fer servir el signe = com a operador d'assignació,

```
a=3
```

però és més propi la fletxa <-

```
a<-3
```

que també serveix cap a la dreta:

```
2 -> x
```

El signe = es fa servir, com veurem més endavant, per donar als paràmetres opcionals de les funcions valors diferents dels que tenen per defecte.

Per poder veure el contingut d'un objecte només cal posar el seu nom:

```
x<-10
x<-x+12
x
```

cal anar en compte de no trepitjar variables que ens interessen ja que no ens avisa.

Operador de concatenació

Permet formar vectors a partir d'escalars o altres vectors

```
x <- c(1,2,3,4)
```

```
y <- c(4,3,2,1)
```

```
u <- c(x,4,x,y)
```

En el cas de cadenes de caràcters, el resultat de `c( )` és un vector de cadenes de caràcters, que no és la mateixa cosa que la cadena de caràcters més llarga obtinguda enganxant les cadenes operands. Per aquest propòsit hi ha un operador específic, `paste()`.

Proveu per exemple:

```
b1<-paste(‘‘A’’, 1:6)
b2<-paste( 1:6, ‘‘A’’)
a<-paste(‘‘avui és’’, date())
```

Operacions aritmètiques

Les usuals `+ - * /`. Per calcular potències podem posar `**` o bé `^`.

```
z<- x+y    z<- x-y   z<- x*y   z<- x^2  z<-x**2  z<-1/x
```

Amb vectors de diferent longitud, com per exemple si sumem u=(1,2,3,4,4,1,2,3,4,4,3,2,1) i x=(1,2,3,4), repetirà els elements del vector x tantes vegades com calgui.

```
z<- x+u
z<-x+c(2,3)
```

Proveu a veure que fa

```
x/u  i  x/2
```

Funcions

En R les funcions s'escriuen amb un nom, seguit de parèntesis, entre els quals es posen els arguments. Cada funció té 0 o més arguments obligatoris, i potser alguns d'opcionals.

Hi ha funcions primitives i d'altres, escrites en el propi llenguatge i que es troben en fitxers amb extensió `.R` Qualsevol successió de comandes que s'escriu de manera interactiva pot gravar-se en un tal fitxer, per repetir en una altra ocasió.

Atenció!
Quan s'invoca una funció cal posar-hi els parèntesis, fins i tot si la funció no té cap argument. El nom a soles produeix que R volqui a la pantalla la definició de la funció. Per exemple

```
ls()
```

és una funció sense arguments, que produeix la llista d'objectes presents en memòria. En canvi, podeu veure què passa si entrem només

```
ls
```

Funcions simples

```
max(), min(), sqrt(), abs(), log(), exp()
```

Funcions trigonomètriques

```
sin(), cos(), tan()
```

Funcions especials

```
choose(n,k), gamma(), beta()
```

Per saber que fa una funció podeu utilitzar la funció help, per exemple `help(max)`.

Successions regulars

Proveu les comandes següents:

```
1:10
seq(-5,5,by=0.4)
seq(-5,4,by=0.4)
seq(-5,5,length=150)
```

Gràfiques

La funció genèrica que fa dibuixos és `plot()` Per exemple si volem dibuixar la paràbola $y = x^2$ entre -15 i 15 (com vam fer amb l'EXCEL) farem:

```
x<-seq(-15,15, by=0.05)
y<-x**2
plot(x,y, type= "p")
plot(x,y, type="l")
```

la opció type= "p" produeix un diagrama de punts, en canvi, type= "l" produeix un diagrama amb línies contínues. Hi ha més tipus de diagrames, que podeu explorar a partir del help.

Per obtenir la gràfica d'una funció es pot fer com en aquest exemple, amb més punts si cal

```
x<- seq(-5,5,by=0.005)
y<- sin(x)
plot(x,y,type="l")
```

La funció `plot()` té moltes opcions, referents a tota mena d'aspectes del gràfic. Vegeu el help per la sintaxi. Per exemple `col="red"` dibuixarà de color vermell.

Podrieu fer la gràfica de dues funcions a la vegada:

```
y2<- sin(x+2)
par(new=T)
plot(x,y2,col="red")
```

Matrius i arrays de més dimensions

Una matriu bidimensional (o, en general, un array de dimensió més gran), és un vector amb informació adicional mitjançant un altre vector anomenat `dim` (de dimensió) i que conté el nombre de columnes i files.

Generem un vector, sense més estructura

```
u<- c(1:12)
```

L'operador `dim(u)` retorna `NULL`. Assignem a `u` una estructura de matriu de tres files i quatre columnes ($3 \times 4 = 12$):

```
dim(u)<- c(3,4)
```

Podem assignar-li estructura de vector vertical (1 columna i 12 files) o de vector horitzontal (1 filea, 12 columnes):

```
dim(u)<- c(12,1)
dim(u)<- c(1,12)
```

Assignem estructura de matriu (array) tridimensional $2 \times 2 \times 3$:

```
dim(u) <- c(2,2,3)
```

Mireu com queda u.

Es pot crear directament una matriu (2-dimensional) amb

```
u<- matrix(1:12, nrow=3, ncol=4)
```

o, en cas de més de dues dimensions, amb

```
u<- array(1:12,dim=c(2,3,2))
```

El primer argument pot ser de longitud inferior a la deduïda pel vector `dim`, igual al producte dels seus elements. En aquest cas s'anirà repetint fins a omplir el total d'elements del vector de valors. Proveu

```
u <- array(c(1,2),dim=c(3,4))
```

Seccions d'una matriu

Construiu la matriu

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix}$$

Per transposar una matriu (canviar files per columnes): `t(A)`.

La fila 1 l'obtenim: `A[1,]`

La columna 2 d'A `A[,2]`

Una matriu, sense la fila 3

```
a <- matrix(1:20,nrow=4)
a[-3,]
```

Una matriu, sense les columnes 1 a 3 `a[,-(1:3)]`

Donem ara dimensió tres $2 \times 2 \times 3$ a la matriu A:

```
dim(A)<- c(2,2,3)
```

La segona secció 2×2 d'A seria: `A[,,2]`

Operacions amb matrius

Producte exterior *(diàdic, tensorial)*

```
a %o% b
```

Un array amb vector de dimensió igual al resultat de concatenar els dos vectors de dimensió dels operands, i que conté, ordenadament, tots els productes de cada element de `a` amb tots els de `b`.

Proveu per

$$a = \begin{pmatrix} 1 & 2 & 1 \\ 3 & 0 & 1 \end{pmatrix} \qquad b = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$$

veureu que obtenim un array de dimensió $3 \times 2 \times 2 \times 2$

Aquesta operació també es pot escriure

```
outer(a,b,"*")
```

Més en general, es pot posar qualsevol altre operador binari en comptes del de multiplicar; com per exemple "+". Proveu

```
outer(a,b,"+")
```

Producte de matrius Si `a` i `b` són dues matrius compatibles per al producte, aquest s'obté: `a %*% b`

Proveu per

$$a = \begin{pmatrix} 1 & 2 \\ 3 & 0 \end{pmatrix} \qquad b = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$$

Llistes

Estructures de dades formades per llistes de components de caràcter heterogeni.
Aquests components poden tenir noms. Per exemple,

```
x<-list("valor"=1,"retol"="exemple","vector"=c(1,2,3))
```

construeix un objecte `x` amb tres components, que tenen, respectivament, els noms `valor`, `retol` i `vector`. Podem adreçar-nos als components d'una llista pel número (amb doble parèntesi quadrat)

```
x[[1]]
x[[2]]
x[[3]]
```

o bé, pel nom, si en si tenen, amb el separador $

```
x$valor
x$retol
x$vector
```

Encara que en general no necessitarem construir aquests objectes, sí que els farem servir implícitament.

Per exemple, els `data.frame` són els objectes bàsics on emmagatzemar dades estadístiques. Una "matriu de dades" $n \times p$ amb informació de p variables observades sobre n individus anirà bé tenir-la com un `data.frame` amb p components, les p variables, on cada component és un vector de longitud n.

1.2 Distribucions de probabilitat univariants

Per a cadascuna de les distribucions de probabilitat llistades a la columna de l'esquerra de la taula 1.1, hi ha 4 funcions, els noms de les quals es forma a partir del sufix de la taula, amb diferents prefixs. Si `X` és el sufix, els noms de les funcions són com s'explica a la taula 1.2.

1.3 Escriure noves funcions

La sintaxi d'una nova funció en R és

Taula 1.1: Distribucions de probabilitat

Distribució	Sufix del nom	Paràmetres addicionals
beta	`beta`	`shape1, shape2, ncp`
binomial	`binom`	`size, prob`
Cauchy	`cauchy`	`location, scale`
Khi quadrat	`chisq`	`df, ncp`
exponencial	`exp`	`rate`
F	`f`	`df1, df1, ncp`
gamma	`gamma`	`shape, scale`
geomètrica	`geom`	`prob`
hypergeomètrica	`hyper`	`m, n, k`
log-normal	`lnorm`	`meanlog, sdlog`
logística	`logis`	`location, scale`
binomial negativa	`nbinom`	`size, prob`
normal	`norm`	`mean, sd`
Poisson	`pois`	`lambda`
t de Student	`t`	`df, ncp`
uniforme	`unif`	`min, max`
Weibull	`weibull`	`shape, scale`
Wilcoxon	`wilcox`	`m, n`

Taula 1.2: Funcions per calcular distribucions de probabilitat

`dX`	=	la funció de densitat de probabilitat,
`pX`	=	la funció de distribució de probabilitat,
`qX`	=	la funció quantila, és a dir, la (pseudo)inversa de la funció de distribució de probabilitat,
`rX`	=	nombres aleatoris segons la distribució de probabilitat donada.

`nom<-function(`⟨*arguments*⟩`){`⟨*codi*⟩`}`

Per exemple, si volem escriure una funció per a calcular la variància empírica s^2, anàloga a la `VARP` de l'Excel, podem escriure:

```
varp<-function(x){
v<-var(x)
n<-length(x)
v*(n-1)/n}
```

i, per tenir-la a punt per més endevant, podem gravar-ho com un fitxer `varp.R`,deixar-la en el mateix directori de treball i carregar-la amb `source("varp.r")`.

Exercici 1. Escriviu una funció que entri dos enters `n`, `p`, i generi una matriu $n \times p$ d'aleatoris normals de mitjana i desviació estàndard donades.

Exercici 2. Escriviu una funció que estandarditzi una matriu $n \times p$, de dues maneres, de forma que:

(a) Les columnes tinguin mitjana 0 i desviació estàndard 1.

(b) Les columnes tinguin mitjana 0 i els valors estiguin compresos entre -1 i $+1$.

Exercici 3. Escriviu una funció que calculi la matriu de variàncies i covariàncies d'una matriu $n \times p$.

Exercici 4. Donada una matriu $\boldsymbol{X}$ de mida $n \times p$, escriviu una funció que calculi la matriu $n \times n$ de les distàncies euclidianes (ℓ^2) entre les files de $\boldsymbol{X}$.

1.4 Dades

1.4.1 Dades en packages del R

En el package `datasets` hi ha molts conjunts de dades. Un d'ells, que és un clàssic en problemes de classificació, és el de les flors *iris*, de Fisher. La instrucció:

```
data(iris)
```

produeix un `data.frame` que conté aquestes dades.

Instal·leu i carregueu el package `ElemStatLearn`.

Un conjunt de dades que farem servir sovint:

```
data(zip.train)
data(zip.test)
```

1.4.2 Llegir fitxers de dades

La instrucció:

```
x <- read.table("d:/usuaris/fitxer.txt")
```

llegirà un fitxer ASCII contenint dades numèriques, ordenades en files (individus) i columnes (variables).

El resultat és un objecte del tipus `data.frame`.

Si es vol posar una primera línia que contingui els noms de les variables,

```
x <- read.table("d:/usuaris/fitxer.txt", header = TRUE)
```

Hi ha moltes opcions, a fi de tenir en compte les diverses possibilitats de formats de fitxers, per exemple amb valors separats per comes, de formats numèrics, especificant el caràcter separador de decimals. Una variant sovint útil és:

```
x <- read.csv("d:/usuaris/fitxer.csv")
```

També es pot especificar el símbol o símbols que al fitxer indica un valor faltant *(missing)*.

Proveu-ho amb els fitxers `Vida.txt` i `Steam.txt`, que teniu al Campus Virtual:

```
Vida<- read.table("Vida.txt", header=TRUE)
Steam<- read.table("Steam.txt")
```

1.4.3 Fonts externes de dades.

Repositoris de dades que seràn útils:

UCI Machine Learning Repository

DASL: The Data and Story Library

1.5 Estadística descriptiva

Freqüències

Podeu obtenir les freqüències absolutes d'una variable discreta amb la funció `table(variable)`, les freqüències acumulades amb `cunsum(variable)`. Per obtenir les freqüències relatives i les freqüències acumulades relatives només cal dividir pel nombre de valors: `length(variable)`

Per exemple, calculem 200 nombres aleatoris d'una Poisson(5) i calculem les freqüències:

```
x<-rpois(200,5)
# freqüències absolutes
t<-table(x)
# freqüències acumulades
c<-cumsum(t)
# freqüències relatives
t2<-table(x)/length(x)
# freqüències relatives acumulades
c2<-cumsum(x)/length(x)
```

Mitjana

Per obtenir la mitjana aritmètica feu servir `mean`. Posant l'argument opcional `a`, un nombre entre 0 i 0.5, produeix la mitjana *retallada* (*trimmed*), descartant una fracció `a` de dades a cada extrem del vector ordenat. Per `a=0` és la mitjana aritmètica ordinària.

```
anys <- vida$anys
mean(anys)
```

```
    mean(anys,trim=a),  weighted.mean((anys,pesos)
```

Mediana

```
median()
```

Quantiles

```
quantile(x, probs=c(0.25,0.5,0.75))
```

Canviant el vector probs obtenim els **percentils**.

Suma

```
sum()
```

Variància (corregida)

```
var(x)
```

Atenció, aquesta funció retorna la variància corregida $\tilde{s}^2$, amb denominador $n-1$. Si necessitem la variància empírica s, amb denominador n, podeu fer servir, per exemple, la funció de la secció 8.

Covariància

```
cov(x,y)
```

Desviació estàndar

```
sd(x)
```

Coeficient de correlació

```
cor(x,y)
```

La funció

```
summary(x)
```

produeix una llista dels estadístics més corrents.

Coeficient d'asimetria

Per a aquesta funció cal, prèviament, carregar el package `e1071`

```
skewness(x)
```

Coeficient de curtosi

Per a aquesta funció cal, prèviament, carregar el package `e1071`

```
kurtosis(x)
```

1.6 Diagrames univariants

Diagrama de caixa

```
boxplot(x)
```

Histograma

```
hist(x)
```

Paràmetres optatius de la funció `hist()`:

Per posar un nombre de classes fixat, per exemple 12 classes:

```
hist(y, nclass=12)
```

Per posar les vores dels rectangles en posicions concretes. En particular, això permet tenir un histograma amb intervals desiguals:

```
hist(y, breaks=c(0,0.5,1,2,4))
```

Per fer que les alçades dels rectangles siguin les proporcions (freqüències relatives), en comptes de les freqüències absolutes,

```
hist(y, freq=FALSE)
```

Aquesta opció és útil si volem comprovar en quina mesura l'histograma s'aproxima a la funció de densitat de probabilitat. Si, com és el cas amb aquests exemples, hem generat dades d'una llei coneguda, podem comparar-les directament:

```
y<-rexp(200,rate=2)
hist(y, freq=FALSE)
x<-seq(0,7,by=0.05)
y1<-dexp(x,rate=2)
lines(x,y1)
```

La funció `lines()` afegeix una gràfica (amb línies) a la figura ja existent, resultat de l'histograma.

En un cas real, potser coneixem o podem suposar que la llei de probabilitat de les dades és d'una família donada, però desconeixem algun paràmetre que obtenim per *estimació puntual*,

```
y<-rnorm(200,mean=3,sd=1.1)
hist(y, freq=FALSE)
```

Suposem que no sabem la mitjana i variància de la població, i les aproximem amb les corresponents quantitats empíriques:

```
m<-mean(y)
s<-sd(y)
```

Dibuixem a sobre de l'histograma la funció de densitat de probabilitat normal amb paràmetres iguals als obtinguts per estimació:

```
x<-seq(m-3*s,m+3*s,by=0.05)
y1<-dnorm(x,mean=m,sd=s)
lines(x,y1)
```

Estimació no paramètrica de la densitat

Si no ens és coneguda una família paramètrica de lleis de probabilitat a la qual podem ajustar les observacions, podem proposar una *estimació no paramètrica de la funció de densitat de probabilitat*, per exemple, amb la funció `density()` que, essencialment, fa un suavitzat de l'histograma

```
lines(density(y))
```

El paràmetre `bw`, *l'ample de banda (bandwidth)* regula el grau de suavitzat.

```
lines(density(y,bw=0.8))
```

1.7 Diagrama de dispersió i recta de regressió

Els coeficients a b de la recta de regressió de Y respecte de X ($Y = a + b\,X$) a partir d'unes dades que ja tenim en dos vectors `x` (predictor) i `y` (resposta) s'obtenen fent:

```
lm(y~x)
```

podem dibuixar la recta de regressió sobre el diagrama de dispersió obtingut amb `plot(x,y)` amb la funció `abline()`.
