C_{n,k} = (n k) = n! / (k! (n-k)!) —> Nombre combinatori

Pràctica 1

Introducció al programa R

1.1 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.

El software R és lliure, la instal·lació es realitza a través del CRAN (Comprehensive R Achive Network), podeu trobar tota la informació sobre R a http://www.r-project.org. Un manual introductori sobre R que us pot ser útil el trobareu a http://www.math.csi.cuny.edu/Statistics/R/simpleR/.

Us recomanem treballar sempre amb un Script. Per fer-ho heu d'anar:

Archivo → Nuevo script,

així tindreu una finestra on podeu anar escrivint les comandes, amb F5 les podeu executar (una o vàries). Si voleu fer comentaris poseu davant el signe # i no s'executaran. Després només l'heu de guardar amb el nom que vulgueu. Més endavant podeu tornar a executar les comandes o afegir-ne.

1.2 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 (així si ho escrivim, `3 + 2 i` es produeix un missatge d'error).

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

Les variables categòriques (o qualitatives) es codifiquen com a `factors`, que prenen valors que són etiquetes, aquestes 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 el <-

```
a<-3
```

que també serveix cap a la dreta:

```
2 -> x
```

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

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)
```

En el cas de cadenes de caràcters, el resultat de `c( )` és un vector de cadenes de caràcters, però 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()`:

```
paste("A", 1:6, sep = "")
paste("Today is", date())
```

Operacions aritmètiques

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

Exercici. Proveu:

```
z<- x+u
1/x
```

Què passa si són vectors de diferent longitud?

```
z<- x+u
x/u
```

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 poder-la 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 sense parèntesis 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 i comandes més usuals

Per fer el màxim, el mínim, l'arrel quadrada, el valor absolut, el logaritme o l'exponencial, les funcions que necessitem són:

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

Tenim també les funcions trigonomètriques:

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

En combinatòria pels factorials tenim `factorial` i pels nombres combinatoris `choose`. Per exemple, per obtenir 5! i $\binom{10}{2}$ faríem:

```
factorial(5)
choose(10,2)
```

I les següents comandes serveixen per construïr successions regulars:

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

Gràfiques

La funció genèrica que fa gràfiques és `plot()`. Per exemple:

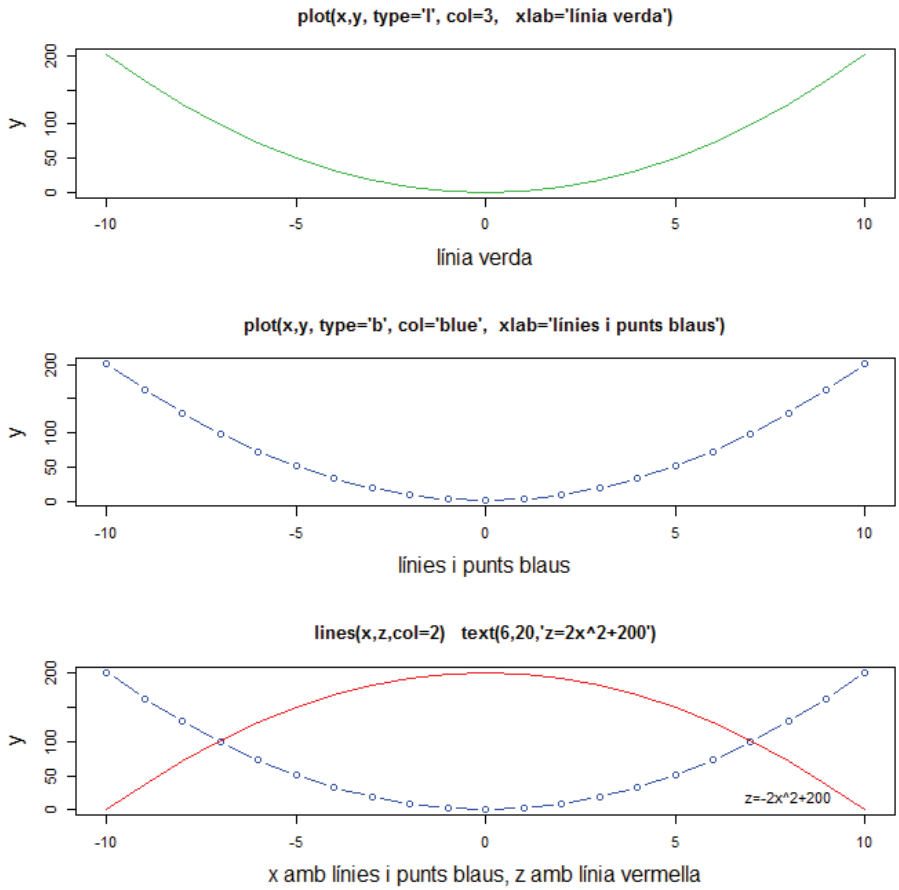
```
x<--10:10
y<-2*x^2+1
plot(x,y)
```

La comanda `plot` té moltes opcions referents a tota mena d'aspectes del gràfic. (Vegeu el help per la sintaxi.)

Per exemple podem fer línies en lloc de punts, podem posar colors, títols, noms als eixos.... A més amb les comandes `lines, points, legend, text` podem afegir línies, punts, títols o textos a un gràfic.

Aquí teniu tres exemples: Si voleu fer aquests gràfics recordeu primer asignar les variables x,y,z.

```
x<--10:10
y<-2*x^2+1
z<--2*x^2+200
```



Per obtenir la gràfica d'una funció ($f(x) = x^2 \sin(1/x)$, per exemple) es pot fer amb aquestes instruccions (amb més punts si calgués)

```
x<-seq(-0.5,0.5,by=0.005)
y<-x**2*sin(1/x)
plot(x,y, type= "l")
```

Podem veure més d'una gràfica alhora utilitzant la funció `par`. Per exemple si volem posar dues gràfiques de costat fem `par(mfrow=c(1,2))`, si volem dues gràfiques una sobre l'altre podem fer `par(mfrow=c(2,1))`. En l'exemple anterior de les tres gràfiques s'ha utilitzat `par(mfrow=c(3,1))`.

Exercici: Feu la gràfica de la funció $f(x) = \frac{1}{x}e^{2x^2}$ entre $-\frac{1}{2}$ i $\frac{1}{2}$ posant títols al gràfic i a cada un dels eixos.

Matrius i arrays de més dimensions

Una matriu o, en general, un array de dimensió més gran és un vector de valors, amb informació addicional: un vector `dim` d'enters positius, amb producte igual a la longitud del vector de valors.

Generem un vector, sense més estructura

```
u <- 1:12
```

L'operador `dim(u)` retorna `NULL`. Li podem assignar a `u` una estructura de matriu (vector) fila 1×12,

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

o bé, li assignem estructura de matriu (vector) columna 12×1,

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

o bé, li assignem estructura de matriu 3×4,

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

o bé, li assignem estructura de matriu tridimensional $2 \times 2 \times 3$,

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

També, 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))
```

Observem que el primer argument pot ser de longitud inferior a la deduïda del 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,

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

Seccions d'una matriu

La fila 1 d'una matriu: `u[1,]`.
La columna 1 d'una matriu: `u[,1]`
La segona secció 2×2 d'un array de dimensió: `c(2,2,3)`: `u[,,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)]
```

Operacions amb matrius

Producte de matrius Si `a` i `b` són dues matrius compatibles pel producte, aquest s'obté:

```
a %*% b
```

transposta Per obtenir la transposta d'una matriu fem:

```
t(a)
```

inversa Per obtenir la inversa d'una matriu quadrada:

```
solve(a).
```

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 tenen) utilitzant 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.

1.3 Llegir dades externes

La instrucció

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

llegirà un fitxer ASCII que és al disc d carpeta "usuaris" contenint dades numèriques, ordenades en files (individus) i columnes (variables). A vegades és pràctic canviar el directori en el programa R (Archivo, Cambiar dir...) i que llegeixi directament de la carpeta on tenim els fitxers de dades. Es poden posar els noms de les variables en la primera línia, per llegir-los es fa:

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

Hi ha moltes opcions, per tal 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. També es pot especificar el símbol o símbols que al fitxer indica un valor faltant *(missing)*.

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

Exercici: Carregar les dades del fitxer `peixos.txt` que teniu al campus.

1.4 Escriptura de noves funcions

La sintaxi d'una nova funció en R és

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

Per exemple, suposem que volem una funció que intercanviï la primera i l'última component d'un vector:

```
f<-function(x){
n<-length(x)
aux<-x[1]
x[1]<-x[n]
x[n]<-aux
x
}
```

Un altre exemple podria ser una funció que donades les probabilitats dels diferents punts (ordenats) d'una variable discreta retorni el valor de la Funció de Distribució en aquests punts, és a dir, la probabilitat acumulada:

```
Fdd <- function (x) {
n<-length(x)
fdd <- rep(0,n)
fdd[1]<-x[1]
for (i in 2:n) fdd[i] <- x[i] + fdd[i-1]
fdd
}
```

Exercici: Proveu ara de fer:

```
Fdd(c(2/3,1/6,1/6))
Fdd(c(2,3,4,5))
```

Això també ho podriem fer amb la funció `cumsum`, pero cap de les dues controla si els valors del vector són positius ni que sumin 1. Podem imposar també aquestes condicions:

```
Fdd <- function (x) {
if (sum(x)!=1| sum(x<0)>0) {"Error, no és un vector de probabilitats"}
else{
n<-length(x)
fdd <- rep(0,n)
fdd[1]<-x[1]
for (i in 2:n) fdd[i] <- x[i] + fdd[i-1]
fdd}
 }
```

Exercici: Què passa ara si executeu les dues instruccions de l'exercici anterior?

El llenguatge de R no està optimitzat per `for`, sinó que està pensat per utilitzar càlcul matricial. Podríem canviar la funció Fdd per:

```
Fdd<-function(x){
if (sum(x)!=1| sum(x<0)>0) {"Error, no és un vector de probabilitats"}
else{
n<-length(x)
mat<-matrix(rep(1,n*n), nrow=n)
mat[lower.tri(mat)]<- 0
fdd<-x%*%mat
fdd} }
```

i, per tenir-la a punt per més endavant, podem gravar-la com un fitxer `Fdd.R`.

1.5 Problemes

1. Representeu, en un gràfic de punts, dues variables del fitxer `peixos`. Poseu títol al gràfic i canvieu el nom dels eixos.

2. Entreu les matrius

$$A = \begin{pmatrix} 0 & 1 & -1 & 1 \\ -1 & 1 & 0 & -1 \end{pmatrix} \quad \text{i} \quad B = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix}$$

   i multipliqueu-les. Quin és el segon element de la segona columna?

3. Escriviu una funció de R que donat un enter x calculi:

$$F(x) = \sum_{n=0}^{|x|} \frac{2^n n!}{(n+14)}.$$

4. Escriviu una funció en R que donats dos vectors x i y de longitud n, calculi

$$F(x,y) = \sum_{k=1}^{n} \frac{(x(k) - 2^{x(k)})y(k)}{k!}.$$

5. Escriviu una funció en R que donat un real x calculi

$$F(x) = \sum_{k=0}^{100} \frac{x^k + x^{2k}}{k!}$$