Sessió 6

Creació de funcions amb R i la funció apply

6.1 *Les funcions apply, sapply i lapply*

La funció **apply** s'utilitza amb data frames o amb matrius. Serveix per aplicar una funció a les files o a les columnes de la matriu o el data frame. La funció `apply` té (almenys) tres arguments:

- una matriu o un data frame.
- un 1 o un 2, depenent de si volem aplicar la funció per files (1) o per columnes (2).
- el nom de la funció .

```
> (A<-matrix(1:12,nrow=3))
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> apply(A,1,sum)    # ens dona la suma de les files de la matriu a
[1] 22 26 30   # la primera fila suma 22, la segona 26, la tercera, 30
> apply(A,2,sum)       # ens dona la suma de les columnes de la matriu a
[1]  6 15 24 33  # la primera columna suma 6, la segona 15, etc
```

Nota: Si la funció que s'aplica té més arguments, els podem posar a continuació, després del nom de la funció, per exemple

```
apply(matriu,1,sum,na.rm=TRUE)
```

Les funcions **lapply** i **sapply** s'apliquen a una llista. La primera retorna una llista i la segona un vector. L'estructura és similar a la d'apply però sense els números 1 o 2: `lapply(llista,funcio)`. En aquest cas s'aplica la funció a cada element de la llista. Podríem aplicar `lapply` i `sapply` a un vector, però aleshores R abans convertiria el vector en una llista. Les funcions `sapply` i `lapply` ens permeten fer de cop una operació que si no, hauríem de repetir. Vegem dos exemples:

- Recordem que un data frame és un cas particular de llista, i els seus elements són les variables (les columnes). Si volem descobrir el tipus de les variables d'un data frame, en comptes d'anar escrivint variable per variable `class(variable)` podem utilitzar `sapply` i ho fa de cop per a totes les variables:

  ```
  > sapply(iris,class)
  Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
     "numeric"    "numeric"    "numeric"    "numeric"     "factor"
  ```

  Fixem-nos que ens torna un vector amb les components etiquetades.

- Quan volem applicar a les variables d'un data frame una funció amb sortida més complicada, com per exemple la funció `quantile`, utilitem `lapply`

```
> lapply(iris[,1:4], quantile)
$$Sepal.Length
  0%  25%  50%  75% 100%
 4.3  5.1  5.8  6.4  7.9

$Sepal.Width
  0%  25%  50%  75% 100%
 2.0  2.8  3.0  3.3  4.4

$Petal.Length
  0%  25%  50%  75% 100%
1.00 1.60 4.35 5.10 6.90

$Petal.Width
  0%  25%  50%  75% 100%
 0.1  0.3  1.3  1.8  2.5
```

Fixem-nos que amb una sola instrucció hem obtingut els quartils i el màxim i el mínim de totes les variables numèriques del data frame `iris`. L'objecte de sortida és una llista.

Atenció: Quan volem que la sortida sigui un data frame utilitzem `lapply` (perquè un data frame és una llista).

Exemple de com fer servir lapply i sapply per arreglar un data frame importat

Carreguem el fitxer `mundo.RData` tal com està a Moodle (fitxers pràctica 6!!!).

```
load("mundo.RData")  # és al nostre directori de treball!
# mirem la capçalera per donar-li una ullada
head(mundo)
# ens fixem en el tipus de les variables
sapply(mundo,class)
# hi ha unes variables tipus factor que haurien de ser numèriques.
# podem guardar en un vector lògic si una variable és tipus factor o no.
vars.incorrectes<-sapply(mundo,class)=="factor"
# visualitzem la tercera fila d'aquestes variables
head(mundo[3,vars.incorrectes])
          pais poblac densidad    relig pib_cap calorías   sida
3     Alemania 81.200    227,0 Protest.  17.539    3.443 11.179
# Ens adonem que hi ha dues variables factor que no hem de canviar: pais i relig
vars.incorrectes["pais"]<-FALSE
vars.incorrectes["relig"]<-FALSE
# ara ja no les tenim com a TRUE a la llista de vars.incorrectes
head(mundo[3,vars.incorrectes])
  poblac densidad    relig pib_cap calorías   sida
3 81.200    227,0 Protest.  17.539    3.443 11.179
# Les passarem primer a variables tipus caràcter
mundo[,vars.incorrectes]<-lapply(mundo[,vars.incorrectes],as.character)
# fixem-nos que cal la funció lapply perquè la sortida ha de ser
# una llista (un data frame) i no un vector
```

```
# La variable densidad està malament, els decimals són comes.
# Ho arreglem amb la funció gsub que substitueix un caràcter
# o una cadena per un altre caràcter o cadena de caràcters
# exemple
> gsub(",",".","3,1416")
[1] "3.1416"
mundo$densidad<-gsub(",",".",mundo$densidad)
mundo[3,vars.incorrectes]  # ha funcionat!
# ja només queda passar-les a numèriques
mundo[,vars.incorrectes]<-lapply(mundo[,vars.incorrectes],as.numeric)
sapply(mundo,class) # arreglat!
```

Pràctica: • Fes una matriu de 1000 files i 2 columnes que contingui tirades de daus aleatòries (necessitaràs la funció `sample`). Cada fila representarà el llençament de dos daus. Hem fet el llençament de dos daus 1000 cops. Es tracta de calcular les freqüències dels possibles resultats de la suma.
Amb la funció `apply` calcula la suma dels dos daus en cada tirada (en cada fila).
Fes una representació gràfica de l'experiment (suma de dos daus, amb 1000 tirades).
• Recupera l'script utilitzat per obtenir el data frame DESPESA de la pràctica 5 (Despesa en lleure de 15 estudiants durant els 12 mesos de l'any). Pots obtenir-lo directament del Moodle (tot l'script de la pràctica, `sess5-requadr-pract.R`). L'objecte DESPESA conté les despeses de 15 alumnes (per files) els 12 mesos de l'any (per columnes). Es tracta d'afegir a la matriu 15×12 una fila i una columna. La fila ha d'incloure les mitjanes dels mesos i la columna les mitjanes de despesa de cada alumne. A la component 16,13 hi poses la despesa mitjana de tots els alumnes i tots els mesos.

6.2 Creació de funcions

Fins ara hem utilitzat funcions de R, com per exemple `mean(x)`, `sd(x)`, `plot(x, y, ...)`, `matrix(...)`

Fixem-nos que:

- Les funcions poden tenir més d'un argument o input, per exemple, l'argument de `mean()` és un vector.
- A vegades el nombre d'arguments pot ser variable, com per exemple `mean(x,na.rm=T)` i `mean(x)`, o bé `matrix(1:10,nrow=2)` o `matrix(1:10,nrow=2, byrow=T)`
- Les funcions també tenen un conjunt d'outputs o sortides, per exemple la sortida podria ser un número, com a `mean` o bé una matriu, com a `matrix(1:10,nrow=2)` o bé una llista . És el que ens apareix a la consola quan executem la funció.
- Algunes funcions no estan carregades quan arrenquem R, i s'han de carregar abans d'utilitzar-les. Per exemple `read.spss` no funciona si no carreguem abans el paquet `foreign`

Però a vegades volem fer un càlcul o un procediment que no està definit a cap funció de les ja existents a R (pot ser que vulguem fer unes quantes funcions de R seguides). Si el càlcul o procediment el volem repetir més d'una vegada, amb diferents conjunts de dades, haurem de **crear una funció pròpia**.

Comencem la construcció de noves funcions. Una funció té l'estructura següent:

```
nom_funcio <- function(arg1, arg2, ...){
  instruccions
  output
}
```

Donem un nom a la funció, indiquem els seus arguments, programem les tasques o instruccions que fa la funció i finalment donem l'output.

Veurem com definir funcions amb exemples.

```
f1 <- function(x){
  x^2
  }
```

Obsevem que:

- Anomenem `f1` la funció.
- La commanda `function` seveix per crear la funció.
- Dins del parèntesi de `function` hi posem els objectes que seran arguments de la funció. Els hi donem un nom per treballar-hi. En aquest exemple hi ha un sol argument, que anomenem `x`.
- La variable `x` dins de la funció és LOCAL (podem tenir a l'entorn de treball un objecte que es digui `x`, que no es veurà afectat per les modificacions que fem dins de la funció de la variable x).
- Ara ja podem utilitzar la funció `f1` que hem creat:

```
> f1(3)
[1] 9
> f1(1:5)
[1]  1  4  9 16 25
> x<- 100   # a l'entorn de treball hi posem un objecte anomenat x
f1(6)
[1] 36   # quan apliquem la funció a 6, dins de la funció x val 6,
x
[1] 100  #  però fora de la funció el valor de x segueix sent 100.
```

La funció següent té dos arguments, x i p i té com a output x^p:

```
f2<-function(x,p){
  x^p
}
f2(2,10)
f2(5,2)
f2(1:5,3)
```

La funció `f3` ens calcula la mitjana i la mediana d'una variable. La sortida és una cadena de caràcters:

```
f3 <- function(x){
  mitjana<-round(mean(x),2)
  mediana<-round(median(x),2)
  paste("mitjana=",mitjana,", mediana=",mediana)
}
```

- Posem diferents comandes separades per línies dins de les claus (`{ ... }`).
- El que volem que sigui la sortida de la funció ho posem a la darrera línia.
- La idea és que una funció conté diferents línies d'un escript que executem seguides.

Fixem-nos que ara podem aplicar la funció `f3` a un vector o també a una matriu. I, és clar, també usar-la dins d'un `apply`:

```
f3(sample(1:6,10,replace=T))  # f3 aplicada a un vector
DESPESA <- matrix(sample(seq(6,200,by=0.01),180,rep=T),15,12)
f3(DESPESA)  # mitjana i mediana de tota la matriu DESPESA
apply(DESPESA,2,f3) # mitjana i mediana de cada  columna de DESPESA
```

En les funcions podem donar un **valor per defecte a algun argument**, com passa amb les funcions de R ja existents (per exemple, la funció `sum` té per defecte `na.rm=FALSE`)

```
potencia <- function(x, p=2){
  x^p
}
```

Si cridem la funció `potencia` amb dos arguments, és com la funció `f2` si no, la funció `potencia` eleva al quadrat.

```
potencia(4, 3)  # ara li hem dit p = 3
[1] 64
potencia(p=5, x=2) # podem desordenar els arguments si posem els seus noms
[1] 32
potencia(4) # si no posem el segon argument, per això calcula el quadrat
[1] 16
```

En canvi la funció `f2` no tenia tenia valor de `p` per defecte i si la cridem amb un sol argument, es queixa.

```
> f2(2)
Error in f2(2) : argument "p" is missing, with no default
```

La sortida d'una funció pot ser qualsevol tipus d'objecte (llista, gràfic,...). En la propera funció la sortida és una llista:

```
resum <- function(x)
{
 # x és un vector
 # calcularem la mitjana, la desviació, el mínim i el màxim
  mitj <- mean(x)
  desv <- sd(x)
  mi <- min(x)
  ma <- max(x)
  return(list(mitjana=mitj,desvicació=desv,
          mínim=mi, màxim=ma))
}
resum(DESPESA)
```

Exemple d'utilització de `if` dins d'una funció: les mitjanes generalitzades

• La *mitjana geomètrica* d'un vector `x` de longitud n es defineix com

$$g = (x_1 x_2 \cdots x_n)^{1/n}.$$

Però per calcular-la no s'utlitiza el producte perquè provoca molts errors numèrics i, de fet, R no permet aplicar la funció `prod` a vectors molt llargs. Per això s'apliquen logaritmes a la definició anterior de g i s'obté la fórmula següent per calcular-la:

$$\log(g) = \frac{\sum_{i=1}^{n} \log(x_i)}{n}, \quad \text{i} \quad g = \exp(\log(g)) \tag{6.1}$$

A continuació definim una funció de R que calculi la mitjana geomètrica utilitzant la darrera fórmula i que no faci res, només donar un missatge d'avís si el vector `x` té alguna component negativa o zero.

```
mitjgeom<- function(x)  # x ha de ser un vector de valors positius
  {
  if (any(x<=0)) stop("mitjana geom no es pot calcular, algun x <=0!")
  else
    {
    n<-length(x)
    lg<-sum(log(x))/n
    exp(lg)
  }
}
```

Un cop hem creat la funció `mitjgeom`, ja la podem utilitzar.

```
dades<- 1:10
mitjgeom(dades)
[1] 4.528729
dad1<-c(-1,2,4)
mitgeom(dad1)
[1] Error in mitjgeom(dad1) : mitjana geom no es pot calcular, X<=0!
```

• La *mitjana generalitzada d'ordre* p de la variable X es defineix com

$$M_{p,X} = \left(\frac{1}{n} \sum_{i=1}^{n} x_i^p \right)^{\frac{1}{p}} \tag{6.2}$$

Una funció que calcula la mitjana generalitzada d'ordre p amb $p \geq 1$ és

```
 mitjana.ordre.positiu <- function(x,p){
  if (any(x<=0)){stop("X<=0")}
  if (p<=0){stop ("p<=0")}
  else mean(x^p)^(1/p)
}
```

Podem comprovar que funciona, calculant la mitjana d'ordre 3, directament i amb la funció:

```
set.seed(1)
y<-runif(100,min=1,max=2) # generem números aleatoris a l'interval [1,2]
mitjana.ordre.positiu(y,3)
[1] 1.562964
(sum(y^3)/100)^(1/3)
 [1] 1.562964
```

• Fixem-nos que si $p = -1$ la fórmula (6.2), de la mitjana generalitzada dóna justament la *mitjana harmònica*:

$$M_{-1,X} = \left(\frac{1}{n}\sum_{i=1}^{n} x_i^{-1}\right)^{-1} = \frac{1}{\frac{1}{n}\sum_{i=1}^{n}\frac{1}{x_i}} = \frac{1}{\frac{1}{H_X}} = H_X$$

Per a $p = 0$ es demostra que "passant al límit" que la mitjana d'ordre p és la mitjana geomètrica, de manera que

$$M_{0,X} = G = e^{\log(g)}, \qquad \log(g) = \frac{1}{n}\sum_{i=1}^{n}\log(x_i)$$

La propera funció calcula la mitjana generalitzada d'ordre p, incloent $p = 0$ i $p = -1$.

```
mitjana.ordre.p <- function(x,p){
  if (any(x<=0)){stop("X<=0")}  # si algun x és <=0 la funció para
  if (p==0){return(exp(mean(log(x))))} # si p=0, mitjana gemoètrica
  else mean(x^p)^(1/p) # en cas contrari s'usa fórmula (7.2)
}
```

Optatiu: I si volem calcular unes quantes mitjanes generalitzades de la variable X de cop? Podem utilitzar la funció `sapply`.

```
# mitjanes d'ordre 2,3 i 4 del vector y del requadre anterior
sapply(c(2,3,4), function(p) mitjana.ordre.p(y,p))
[1] 1.541021 1.562964 1.583494
# si volem posar l'ordre p de la mitjana arrodonida:
sapply(c(2,3,4), function(p) paste("ordre",p, "=",mitjana.ordre.p(y,p)))
"mitjana ordre 2 = 1.54102086359341" "mitjana ordre 3 = 1.56296399646648"
[3] "mitjana ordre 4 = 1.58349357346897"
```

A l'exemple anterior hem utilitzat el que s'anomena *funció anònima*, que és una funció que no s'associa a una variable sinó que només s'utilitza dins d'una altra funció, com per exemple dins d'`apply`.

Pràctica:

• Defineix una funció que agafi tres números, els elevi al cub per separat i ens retorni la suma.

• Defineix una funció que ens calculi la mitjana ponderada. Si $x = (x_1, \ldots, x_k)$ són els valors de la variable i $w = (n_1, \ldots, n_k)$ són els pesos (podrien ser les freqüències absolutes) aleshores la mitjana ponderada és

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{k} x_i n_i, \qquad \text{amb } n = \sum_{i=1}^{k} n_i.$$

Com a arguments de la funció, posa-hi el vector x i el vector w de pesos.

• Recupera l'script utilitzat per obtenir el data frame `PerTur17` de la pràctica 4 (Enquesta sobre la percepció del turisme pels ciutadans de Barcelona ciutat). Pots obtenir-lo directament del Moodle (tot l'script de la pràctica, sess4-practs, o bé només el data frame `PerTur17.RData`). Crea una funció que faci la taula amb les freqüències absolutes i els percentatges d'una variable categòrica. Utilitzant la funció `apply()` aplica-la a totes les variables `FREQ_VISIT_...` del data frame `PerTur17` (Ajuda: pots fer un sub data frame amb les variables que vols treballar).

• Recupera l'script utilitzat per obtenir el data frame DESPESA de la pràctica 5 (Despesa en lleure de 15 estudiants durant els 12 mesos de l'any). Pots obtenir-lo directament del Moodle (tot l'script de la pràctica, sess5-practs).

a) Utilitzant la funció apply, dibuixa en una sola pantalla els box-plot de la despesa dels 15 alumnes durant tot l'any, tots escalats de 6 a 200.

b) Fes el mateix per a la despesa de tots els alumnes durant cada un dels 12 mesos.