Capítol 3

Lists

Un **list** l és un objecte **estructurat** composat per objetes anomenats components o elements. A diferència d'un vector, els components d'un list poden ser de tipus diferents. Aleshores, els lists són objectes heterogenis. De manera genèrica, un list l és de la forma:

t_1	t_2	t_3	t_4	t_5	t_6	$\ldots$	t_N

on $\forall i \in \{1, \ldots, N\}$, $t_i \in T_i$. És a dir, t_i és un valor del tipus T_i i $\forall i \neq j$, $i, j \in \{1, \ldots, N\}$ no necèssariament $T_i = T_j$.

Per exemple, considerem el següent list:

$Enric$	$Canals$	21	H	1.77	$FALSE$	66543211	$93 - 4444446$

En aquest exemple el list té components de diferents tipus: cadena de caracteres, enter, real i booleà.

Els components poden rebre un nom que facilita l'accés als components i permet un nivell d'abstracció més alt.

nom_1	nom_2	nom_3	nom_4	nom_5	nom_6	$\ldots$	nom_N
t_1	t_2	t_3	t_4	t_5	t_6	$\ldots$	t_N

En aquest list, $\forall i \in \{1, \ldots, N\}$, $t_i \in T_i$ i nom_i és l'identificador del i-éssim component. Com veurem a l'apartat 3.1, els noms d'un list poden ser dinàmicament assignats. A sota podem veure el list de l'exemple anterior amb noms per als components:

Nom	$Cognom$	$Edat$	$Sexe$	$Alçada$	$Treballa$	$Mobil$	$Telefon$
$Enric$	$Canals$	21	H	1.77	$FALSE$	66543211	$93 - 4444446$

En aquest exemple el list té components anomenats “Nom” i “Cognom” de tipus cadena de caràcters, “Edat” de tipus enter, “Sexe” de tipus caràcter (’H’/’D’), “Alçada” de tipus real, “Treballa” de tipus booleà (TRUE/FALSE) i, “Mobil” i “Telefon” que podrien ser de tipus cadena de caràcters també.

3.1 Creació de lists

Per a construir un list tenim la següent funció constructora:
`list()`

Aquest constructor ens permet crear lists buits, lists per enumeració dels seus components i lists amb *tags* o noms per als seus components, com segueix:

1. list buit

```
> l <- list()
> l
list()
>
```

2. list per enumeració

```
> l <- list("Enric", "Canals",21,"H",1.77,FALSE
    ,66543211,"93 - 4444446")
> l
[[1]]
[1] "Enric"

[[2]]
[1] "Canals"

[[3]]
[1] 21

[[4]]
[1] "H"

[[5]]
[1] 1.77

[[6]]
[1] FALSE

[[7]]
[1] 66543211

[[8]]
[1] "93 - 4444446"
>
```

3. list amb noms per als components

```
> l <- list(Nom="Enric", Cognom="Canals",Edat=21,
    Sexe="H",Alcada=1.77,Treballa=FALSE,Mobil
    =66543211,Telefon="93 - 4444446")
> l
$Nom
[1] "Enric"

$Cognom
[1] "Canals"

$Edat
[1] 21

$Sexe
[1] "H"

$Alcada
[1] 1.77

$Treballa
[1] FALSE

$Mobil
[1] 66543211

$Telefon
[1] "93 - 4444446"
```

Addicionalment, podem consultar els noms dels components d'un list mitjançant la funció `names(l)` que donat un list `l` ens retorna un vector amb els noms dels seus components:

```
> names(l)
[1] "Nom"      "Cognom"   "Edat"     "Sexe"
"Alcada"   "Treballa" "Mobil"    "Telefon"
```

Molts cops és necessari donar-li noms als components d'un list creat prèviament per enumeració. Això es pot fer assignant a `names(l)` el vector amb els noms que es vol:

```
> l<-list("Maria Perez","FIB",2011)
> l
[[1]]
[1] "Maria Perez"

[[2]]
[1] "FIB"

[[3]]
[1] 2011
```

```
11
12 #Fins aqui els components del list l no tenen
       noms. Ara li afegim els noms:
13
14 > names(l) <- c("Nom","Centre","Ingres")
15 > l
16 $Nom
17 [1] "Maria Perez"
18
19 $Centre
20 [1] "FIB"
21
22 $Ingres
23 [1] 2011
```

3.2 Accés als components d'un list

Existeixen diferents maneres d'accedir directament a cada component d'un list. En efecte, mitjaçant **operadors d'accés directe**, que tenen com a paràmetre o bé el nom d'un component o bé un índex que indica la posició del component que es desitja visitar, es pot accedir a qualsevol component del list de manera directa. Les especificacions d'aquests operadors són les següents:

1. $\$: list \times id \rightarrow T_{id}$

   Essent l un list, id el nom d'un component de l i T_{id} el tipus del component amb nom id. La crida a aquest operador té la següent sintaxi

   ```
   l$id
   ```

   ```
   1 > l<-list(Nom="Maria Perez",Centre="FIB",Ingres
         =2011)
   2 > l$Nom
   3 [1] "Maria Perez"
   4 > l$Centre
   5 [1] "FIB"
   6 > l$Ingres
   7 [1] 2011
   8 >
   ```

2. $[[\quad]] : list \times enter \rightarrow T$

   En aquest operador d'accés, els lists són tractats de manera semblant als vectors i fem servir la crida

   ```
   l[[exp]]
   ```

   per referir-nos al component del list l que es troba a la posició que resulti d'avaluar l'expressió entera exp. Cal notar, però, una diferència important

amb els vectors, i és que amb el list, l'operador és un doble gafet (`[[ ]]`) i cal parar atenció a aquesta diferència.

De la mateixa manera que passava amb els vectors, els accessos als components d'un list han de ser necessàriament posicions permeses en el list, i per tant, cal tenir la possibilitat de conèixer la dimensió d'un list. Per fer-ho disposem d'una funció especificada com segueix:

$$length : list \rightarrow enter$$

Així doncs, si un list l té N components, la crida

$$length(l)$$

ens tornarà N.

```
> l<-list(Nom="Maria Perez",Centre="FIB",Ingres
    =2011)
> l
$Nom
[1] "Maria Perez"

$Centre
[1] "FIB"

$Ingres
[1] 2011

> length(l)
[1] 3
>
> l1 <-list()
> length(l1)
[1] 0
```

Ara podem demanar que

$$1 \leq avaluacio(exp) \leq length(l) \qquad (2)$$

```
> l<-list(Nom="Maria Perez",Centre="FIB",Ingres
    =2011)
> l[[1]]
[1] "Maria Perez"
> l[[2]]
[1] "FIB"
> l[[3]]
[1] 2011
> l[[1+1]]
[1] "FIB"
> l[[2*1]]
[1] "FIB"
```

Com en el cas dels vectors, és molt important vigilar que es satisfaci la restricció (2). En cas que no la respectem tindrem un problema en intentar accedir a una posiciò il·legal dins del list. En aquests casos, l'R ens respòn con segueix

```
> l<-list(Nom="Maria Perez",Centre="FIB",Ingres
    =2011)
> l[[4]]
Error en l[[4]] : subindice fuera de  los limites
>
```

S'ha de tenir en compte que existeix una bijecció del conjunt de noms d'un list l i la posició en que aquest nom apareix al list. Per exemple, al list l obtingut fent la instrucció

```
l<-list(Nom="Maria Perez",Centre="FIB",Ingres
    =2011):
```

tindríem la bijecció següent:

		posicio
Nom	$\rightarrow$	1
Centre	$\rightarrow$	2
Ingres	$\rightarrow$	3

3. Entre les dues opcions anteriors per accedir als components d'un list l, també existeix una altra manera d'indexar-lo amb la crida següent:

```
l[[quote(id)]]
```

on $quote(id)$ és el nom corresponent al id dins de doble cometes:

```
> l<-list(Nom="Maria Perez",Centre="FIB",Ingres
    =2011)
> l[["Centre"]]
[1] "FIB"
>
```

Com a conseqüencia, tenim tres maneres d'accedir al mateix component:

```
> l<-list(Nom="Maria Perez",Centre="FIB",Ingres
    =2011)
>
> l[["Centre"]]
[1] "FIB"
>
> l$Centre
[1] "FIB"
>
> l[[2]]
[1] "FIB"
>
```

3.3 Afegir i eliminar elements d'un list

Els lists poden créixer i decréixer dinàmicament. És a dir, podem afegir nous components a un list i eliminar qualsevol dels components que existeixin.

Per a afegir un nou element directament li assignem un valor al component i això ja crea el component:

```
> l<-list(Nom="Maria Perez",Centre="FIB",Ingres
    =2011)
> l
$Nom
[1] "Maria Perez"

$Centre
[1] "FIB"

$Ingres
[1] 2011

# Afegim l'element "Universitat":

> l$Universitat <- "UPC"
> l
$Nom
[1] "Maria Perez"

$Centre
[1] "FIB"

$Ingres
[1] 2011

$Universitat
[1] "UPC"
>
```

Per a eliminar un element el que farem és assignar-li el valor `NULL` al component que volem eliminar:

```
> l<-list(Nom="Maria Perez",Centre="FIB",Ingres
    =2011,Universitat="UPC")
> l
$Nom
[1] "Maria Perez"

$Centre
[1] "FIB"

$Ingres
[1] 2011
```

```
11
12 $Universitat
13 [1] "UPC"
14
15 # Eliminem l'element "Centre":
16
17 > l$Centre<-NULL
18 > l
19 $Nom
20 [1] "Maria Perez"
21
22 $Ingres
23 [1] 2011
24
25 $Universitat
26 [1] "UPC"
27 >
```

3.4 Accés a sublists

Per obtenir un sublist a partir d'un list, es fa servir la següent forma d'indexada:

$$[\ \ ] : list \times interval \rightarrow list$$

on *interval* és un interval de la forma [*exp*] o [*exp*1 : *exp*2]. La crida es fa com segueix:

```
l[exp]
```

```
1 > l<-list(Nom="Maria Perez",Centre="FIB",Ingres =2011,
      Universitat="UPC")
2 > l[1]
3 $Nom
4 [1] "Maria Perez"
5 >
6 > l[[1]]
7 [1] "Maria Perez"
8 >
```

S'ha de notar la diferència de significat quan es fa servir gafet simple i quan es fa servir doble gafet. En el primer cas torna un list mentre que en el segon cas torna l'ièsim component.

També podem fer un sublist amb més d'un element del list original usant un subrang per a l'índex:

```
l[exp1:exp2]
```

```
> l[1:3]
$Nom
[1] "Maria Perez"

$Centre
[1] "FIB"

$Ingres
[1] 2011

> l[3:4]
$Ingres
[1] 2011

$Universitat
[1] "UPC"
>
> l[3:1]
$Ingres
[1] 2011

$Centre
[1] "FIB"

$Nom
[1] "Maria Perez"
>
> l[(2+1):(3+1)]
$Ingres
[1] 2011

$Universitat
[1] "UPC"
>
```

3.5 Recorreguts i cerques

Hi ha molts problemes on s'han de recórrer els components d'un list i també problemes on és necessari fer servir l'esquema de cerca. Aquestes dues famílies de problemes les hem caracteritzat al Capítol 1, quan parlàvem de vectors, a les seccions 1.3 i 1.4. Com a exemple, aquí podem veure una funció que escriu un list:

```
escriure_list <- function(l){
   n <- names(l)
   for (i in 1:length(l)){
      cat(n[i],": ",l[[i]],"\n")
   }
}
```

que fa un recorregut del list i que es pot utilitzar d'aquesta forma:

```
1 > l <- list(Nom="Joan",Cognom="Martinez",Edat=23,Grau=
    "Informatica",Universitat="UPC")
2 > escriure_list(l)
3 Nom : Joan
4 Cognom : Martinez
5 Edat : 23
6 Grau : Informatica
7 Universitat : UPC
8 >
```