Les séquences : listes

Algo & Prog avec R

A. Malapert, B. Martin, M. Pelleau, et J.-P. Roy

4 décembre 2025

Université Côte d'Azur, CNRS, I3S, France
firstname.lastname@univ-cotedazur.fr

Qu'est-ce qu'une séquence ?

Séquence
Une séquence est une suite finie de valeurs numérotées, distinctes ou pas, de n'importe quel type.

Vecteur
Un vecteur est une séquence d'éléments du même type.

```
 > 1:6
[1] 1 2 3 4 5 6
> c('foo', 'bar')
[1] "foo" "bar"
```

Liste
Une liste est une séquence d'éléments de type quelconque.

```
> list(1:6, c('foo', 'bar'))
[[1]]
[1] 1 2 3 4 5 6

[[2]]
[1] "foo" "bar"
```

Mutabilité
Les listes et vecteurs sont mutables, i.e. on peut les modifier.

De l'utilité des listes ...

Comment représenter n points de $\mathbb{R}^2$?
On numérote les points qui appartiendront en fait à $\mathbb{F}^2$.

Liste avec 2 vecteurs de taille n (Version I)

- Le vecteur `x` contient les abscisses des points.
- Le vecteur `y` contient les ordonnées des points.

Un carré tourné à 45 degrés

```
> li <- list(c(-1, 0, 1, 0), c(0, -1, 0, 1))
```

Liste de n vecteurs de taille 2 (Version II)
Chaque vecteur contient l'abscisse et l'ordonnée d'un point.

Un carré tourné à 45 degrés

```
> li <- list(c(-1, 0), c(0, -1), c(1, 0), c(0, 1))
```

Les listes sont partout !
Les structures de données à plusieurs dimensions sont souvent des listes décorées.

Accès par rang aux éléments d'une liste

Attention aux différences entre vecteurs et listes

- La notation indexée par double crochets.
- Les indices commencent à 1.
- R renvoie une erreur quand les indices sont négatifs.
- L'accès par rang devient différent de l'extraction par tranches.

```
> li <- list(c(-1, 0), c(0, -1), c(1, 0), c(0, 1))
> li[[2]] # renvoie un vecteur !
[1]  0 -1
> li[[-2]]
Error in li[[-2]] :
  attempt to select more than one element in get1index <real
      >
> li[[2:3]]
Error in li[[2:3]] : indice hors limites
```

Construction d'une nouvelle liste

Construction en extension

On utilise surtout la fonction `list`.

```
> li <- list(42, "foo", TRUE, 1:10)
```

Concaténation de deux séquences

On peut coller côte à côte (concaténer) deux séquences.

```
> li1 <- list(42)
> li2 <- list("foo")
```

Concaténation

```
> append(li1, li2)
[[1]]
[1] 42

[[2]]
[1] "foo"
```

Création d'une liste de listes

```
> list(li1, li2) # création
    d'une liste de listes
[[1]]
[[1]][[1]]
[1] 42

[[2]]
[[2]][[1]]
[1] "foo"
```

Extraction d'une tranche d'une liste

La notation des tranches (slices) est valide pour toute séquence.

- Comme pour les vecteurs, on utilise les crochets simples.
- Le résultat est une liste.

```
> li <- list(c(-1, 0), c(0, -1), c(1, 0), c(0, 1))
```

Vecteur numérique d'indices
Avec duplication.

```
> li[c(1,3,3)]
[[1]]
[1] -1  0

[[2]]
[1] 1 0

[[3]]
[1] 1 0
```

Vecteur logique d'indices
Avec recyclage.

```
> li[c(TRUE, FALSE)]
[[1]]
[1] -1  0

[[2]]
[1] 1 0
```

Comme pour les vecteurs, on peut utiliser les fonctions d'extraction `head` et `tail`.

Autres opérations sur les listes

Modification d'une liste

Avec l'opérateur crochet, la méthode append ou les indices négatifs.

```
> li <- list(c(-1, 0), c(0, -1), c(1, 0), c(0, 1))
> li[c(-1, -3, -4)] # suppression d'éléments
[[1]]
[1]  0 -1
```

Arithmétique, conditions et tests vectorisés ? Non !

On procédera d'autres manières, par exemple par itérations.

```
> li + li
Error in li + li : argument non numérique pour un opérateur
    binaire
> li == li
Error in li == li : comparaison de ces types non implémentée
> li & li
Error in li & li :
  ces opérations ne sont possibles que pour des types numé
      riques, logiques ou complexes
```

Boîte englobante pour un nuage de points

Comment représenter n points de $\mathbb{R}^2$?
On numérote les points qui appartiendront en fait à $\mathbb{F}^2$.

1. Liste avec 2 vecteurs de taille n.
   - Le vecteur `x` contient les abscisses points.
   - Le vecteur `y` contient les ordonnées des points.
2. Liste de taille n dont chaque élément est un vecteur de taille 2.
   - Chaque vecteur contient l'abscisse et l'ordonnée d'un point.

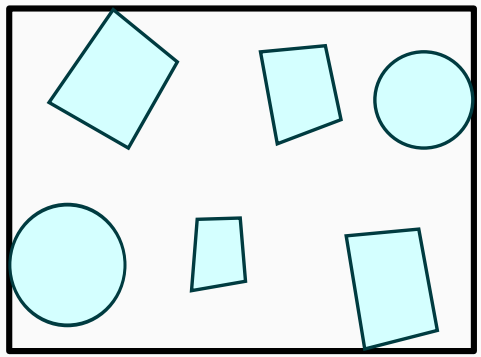

Figure 1: Qu'est-ce qu'une boîte englobante ? Merci Wikipedia.

Boîte englobante (Version I)

```
CalculerBoiteEnglobante <- function(pts) {
  ## pts est une liste avec 2 vecteurs
  return(list(
   range(pts[[1]]),
   range(pts[[2]])
   ))
}
```

```
> pts <- list(c(-1, 0, 1, 0), c(0, -1, 0, 1))
> CalculerBoiteEnglobante(pts)
[[1]]
[1] -1  1

[[2]]
[1] -1  1
```

Boîte englobante (Version II)

```
CalculerBoiteEnglobante <- function(pts) {
  ## pts est une liste de n vecteurs
  CalculerIntervalleEnglobant <- function(pts, d) {
    ## on reprogramme la fonction range ...
    a <- Inf
    b <- -Inf
    for(p in pts) {
      if(p[d] < a) a <- p[d]
      if(p[d] > b) b <- p[d]
    }
    return(c(a,b))
  }
  x <- CalculerIntervalleEnglobant(pts, 1)
  y <- CalculerIntervalleEnglobant(pts, 2)
  ## on met en forme le résultat
  return(list(c(x[1], y[1]), c(x[2], y[2])))
}
```

Questions?

Retrouvez ce cours sur le site web

`www.i3s.unice.fr/~malapert/R`