Statistiques avec le logiciel R - TD2

Objectifs

— Connaître les structures de données de **R**.
— Connaître les principales fonctions liées aux vecteurs.
— Savoir faire des requêtes sur un vecteur.

1 Les structures de données de R

1.1 Les vecteurs

Les données en **R** peuvent être organisées sous différentes formes, notamment sous forme de vecteurs, de matrices, de tables de données, ou de listes. Le vecteur est l'objet de base. Il représente une suite de données du même type. La fonction permettant de créer un vecteur est la fonction `c()` pour collection ou concaténation. D'autres fonctions plus spécifiques existent comme la fonction `seq()` utilisée pour obtenir une séquence de réels, la fonction `rep(x,n)` pour répéter `n` fois la valeur `x`. On peut donner des types différents de données lors de la création du vecteur, mais **R** se chargera de les transformer vers le type le plus général.

```
> vec1=c('tomate', 'carotte', 'courgette')
> vec1
[1] "tomate"    "carotte"   "courgette"
> vec2=seq(1,5)
> vec2
[1] 1 2 3 4 5
> vec3=c(vec1,vec2)
> vec3
[1] "tomate"    "carotte"   "courgette" "1"         "2"         "3"
[7] "4"         "5"
```

La fonction `length(x)` donne la taille du vecteur `x`. On peut aussi faire appel à un ou plusieurs éléments d'un vecteur en précisant leur place entre crochets `[]`.

```
> length(vec3)
[1] 8
> vec3[1]
[1] "tomate"
> vec3[c(1,3,5)]
[1] "tomate"    "courgette" "2"
```

1.2 La matrice (`matrix`)

C'est une généralisation de la notion de vecteur, puisqu'il s'agit d'une suite de données **du même type** à doubles indices. Une matrice peut-être créée à partir d'un vecteur en précisant le nombre de colonnes et le nombre de lignes.

```
> vec=1:12
> mat=matrix(vec,nrow=3,ncol=4)
> mat
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> class(mat)
[1] "matrix"
```

Avec, l'instruction `x[i,j]`, on accède à l'élément sur la i^e ligne et la j^e colonne de la matrice `x`. `x[i,]` permet d'obtenir la i^e ligne, et `x[,j]` la j^e colonne. Enfin, la fonction `dim` renvoie la dimension de la matrice, c'est-à-dire le nombre de lignes et de colonnes.

```
> mat[2,3]
[1] 8
> mat[1,]
[1]  1  4  7 10
> mat[,1]
[1] 1 2 3
> dim(mat)
[1] 3 4
```

Le tableau `array` est une extension de la matrice au cas de plus de deux dimensions.

1.3 Les facteurs (`factor`)

Cette structure de données est particulièrement utile pour définir un vecteur de données qualitatives. Ces données seront organisées de manière plus astucieuse qu'avec un simple vecteur de chaînes de caractères. De plus, de nombreuses fonctions se comporteront différemment si la donnée est reconnue comme un vecteur ou un facteur. Cette structure est créée avec la fonction `factor`, ou converti avec la fonction `as.factor`, la fonction `levels` renvoie les niveaux de la variable qualitative.

```
> vec=c('vert', 'bleu', 'rouge', 'jaune', 'vert', 'jaune')
> fac=factor(c('vert', 'bleu', 'rouge', 'jaune', 'vert', 'jaune'))
> class(fac)
[1] "factor"
> levels(fac)
[1] "bleu"  "jaune" "rouge" "vert"
```

1.4 La liste (`list`)

C'est à la fois la structure la plus souple et la plus riche du langage **R**. La liste permet de rassembler sous une même structure des données de types différents sans les altérer. Une liste est composée d'une liste de structures de **R**, qui peuvent être un vecteur, une matrice, un facteur ou bien même une liste. On peut créer une liste avec la fonction `list` en nommant et définissant chaque élément de la liste.

```
> MaListe=list(elem1=TRUE, elem2=1:5, elem3='Une chaine de caractères')
> MaListe
$elem1
[1] TRUE

$elem2
[1] 1 2 3 4 5

$elem3
[1] "Une chaine de caractères"
> class(MaListe)
[1] "list"
```

On peut accéder aux différents éléments d'une liste en utilisant directement leur nom avec l'instruction `NomDeLaListe$NomDeL'élément`

```
> MaListe$elem2
[1] 1 2 3 4 5
```

ou en utilisant les doubles crochets `[[]]`.

```
> MaListe[[1]]
[1] TRUE
```

La fonction `str`, permet d'accéder à la structure d'un objet. Cela est particulièrement utile si l'objet est complexe (liste de listes).

```
> str(MaListe)
List of 3
 $ elem1: logi TRUE
 $ elem2: int [1:5] 1 2 3 4 5
 $ elem3: chr "Une chaine de caractères"
```

1.5 Le tableau individus×variables (`data.frame`)

Cette structure est particulièrement utile en statistiques, puisqu'il s'agit du tableau dans lequel les individus sont rangés sur les lignes et les variables sur les colonnes. Tous les éléments d'une même colonne sont du même type. Voici un exemple de `data.frame`

```
> produits=data.frame(Nom=c('Poele', 'Seau', 'Ballon', 'Assiette', 'Trompette'),
+                     Reference=c('2.3.4', '4.2.1', '1.7.3', '5.2.0', '9.1.9'),
+                     Disponibilite= c(T,T,F,T,F),
+                     Prix=c(12.50, 3.89, 7.00, 2.50, 149.00))
> produits
        Nom Reference Disponibilite   Prix
1     Poele     2.3.4          TRUE  12.50
2      Seau     4.2.1          TRUE   3.89
3    Ballon     1.7.3         FALSE   7.00
4  Assiette     5.2.0          TRUE   2.50
5 Trompette     9.1.9         FALSE 149.00
```

On remarque que le tableau `produits` est formé de quatre variables, `Nom`, `Reference`, `Disponibilite` et `Prix`, dont deux sont de type chaîne de caractères, une est de type logique, et la dernière est de type numérique.

Pour accéder à une colonne du tableau, comme pour les listes, nous pourrons directement utiliser son nom.

```
> produits$Prix
[1]  12.50   3.89   7.00   2.50 149.00
```

Chaque colonne du tableau est un vecteur, on peut ainsi accéder à un élément de la colonne choisie en utilisant les crochets.

```
> class(produits$Prix)
[1] "numeric"
> produits$Prix[1]
[1] 12.5
```

Comme pour les matrices, il est aussi possible de définir une colonne ou une ligne avec sa position. Cependant, une ligne d'un `data.frame` reste de nature `data.frame`, puisqu'elle est composée d'éléments de types différents, alors qu'une colonne d'un `data.frame` est un vecteur.

```
> produits[,3]
[1]  TRUE  TRUE FALSE  TRUE FALSE
> produits[1,]
    Nom Reference Disponibilite Prix
1 Poele     2.3.4          TRUE 12.5
> class(produits[1,])
[1] "data.frame"
```

La fonction `dim` donne les dimensions (nombres de lignes et de colonnes) du tableau.

```
> dim(produits)
[1] 5 4
```

2 Faire des requêtes sur les lignes d'un tableau

La fonction `which`, qui prend comme argument un vecteur logique, permet de faire des requêtes sur la table de données. En pratique, on obtiendra un vecteur logique en écrivant une assertion logique à l'aide d'opérateurs logiques (cf. TD1).

```
> which(c(T,T,T,F,F))
[1] 1 2 3
> produits$Prix>30
[1] FALSE FALSE FALSE FALSE  TRUE
> which(produits$Prix>30)
[1] 5
> produits[which(produits$Prix>30),]
        Nom Reference Disponibilite Prix
5 Trompette     9.1.9         FALSE  149
> produits[which(produits$Prix>10 & produits$Disponibilite),]
```

```
    Nom Reference Disponibilite Prix
1 Poele     2.3.4          TRUE 12.5
```

3 Exercices

Exercice 1

1. Que renvoie l'instruction `1:6` ?
2. Donner deux autres instructions différentes qui renvoient au même résultat que l'instruction `1:6`.
3. Que renvoie l'instruction `1:6*2` ?
4. Que renvoie l'instruction `1:(6*2)` ?
5. Qu'en concluez-vous sur la règle de priorité des opérateurs `:` et `*` ?

Exercice 2

On veut créer un vecteur avec les noms de couleurs 'vert', 'bleu', 'rouge', 'jaune', 'noir' et 'blanc'. On appellera ce vecteur `couleurs`.

1. Donner l'instruction qui permet de créer un tel vecteur.
2. De quel type de données est le vecteur `couleurs`. Donner l'instruction qui permet de le vérifier.
3. Donner l'instruction pour obtenir la taille du vecteur `couleurs`.
4. Quelle instruction permet de créer un vecteur ne contenant que les couleurs 'bleu', 'rouge' et 'jaune'. Nommer ce vecteur `coul.prim`. Vérifier sa taille.
5. Donner l'instruction pour concaténer au vecteur `couleurs` la couleur 'rose'.

Exercice 3

Le tableau ci-dessous donne le prix du pain moyen annuel pour la période 1992-2011, obtenu à partir des données INSEE `http://www.insee.fr/fr/bases-de-donnees/bsweb/serie.asp?idbank=000442423`

Prix	2.15	2.23	2.30	2.34	2.39	2.42	2.45	2.50	2.56	2.63
Année	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001
Prix	2.73	2.84	2.95	3.00	3.07	3.18	3.32	3.35	3.35	3.42
Année	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011

1. Enregistrer les prix dans un vecteur `prix`. Faire de même avec les années. On nommera le vecteur `annees`.
2. Vérifier le type de données des vecteurs `prix` et `annees`, ainsi que leur taille.
3. Calculer le prix moyen du pain, ainsi que sa variance. Pour cela, utilisez les fonctions `mean` et `var`.
4. Créer un tableau de données individus×variables, à partir des deux vecteurs `prix` et `annees`.
5. Faire une requête sur les lignes du tableau pour sélectionner les années pour lesquelles le prix du pain était inférieur à 2.50. Combien y a t-il d'années ?
6. Donner le numéro des lignes du tableau pour lesquelles l'année est supérieure ou égale à 2000 et le prix est inférieur à 3.10.

Exercice 4

1. Charger l'environnement de travail `Titanic.RData` à l'aide de la fonction `load()`. La description de cette table est donnée au début du fichier `Titanic.dat2.csv`.
2. Combien de passagers du Titanic sont recensés dans cette table ?
3. Combien de classes y a t-il dans la variable `Class` ? Que représentent-elles ?
4. Quel est le type de données de la variable `Class` ? Proposez un autre type de données plus approprié pour cette donnée et convertissez-là en créant une nouvelle variable. Faire de même pour les variables `Age`, `Sex` and `Survived`.
5. Combien y avait-il de passagers de 1re, 2e et 3e classes ? Combien y avait-il de membres d'équipages ?
6. Quelle était la proportion d'enfants sur le Titanic.
7. Combien d'enfants sont morts sur le Titanic ?
8. Quelle était la probabité de survivre pour un homme adulte ? Pour un homme adulte de première classe ? De deuxième classe ? De troisième classe ? Membre d'équipage ?
9. Un homme de première classe avait-il plus de chance de survivre qu'une femme de troisième classe ?