Introductions aux structures de données et aux fonctions avec R

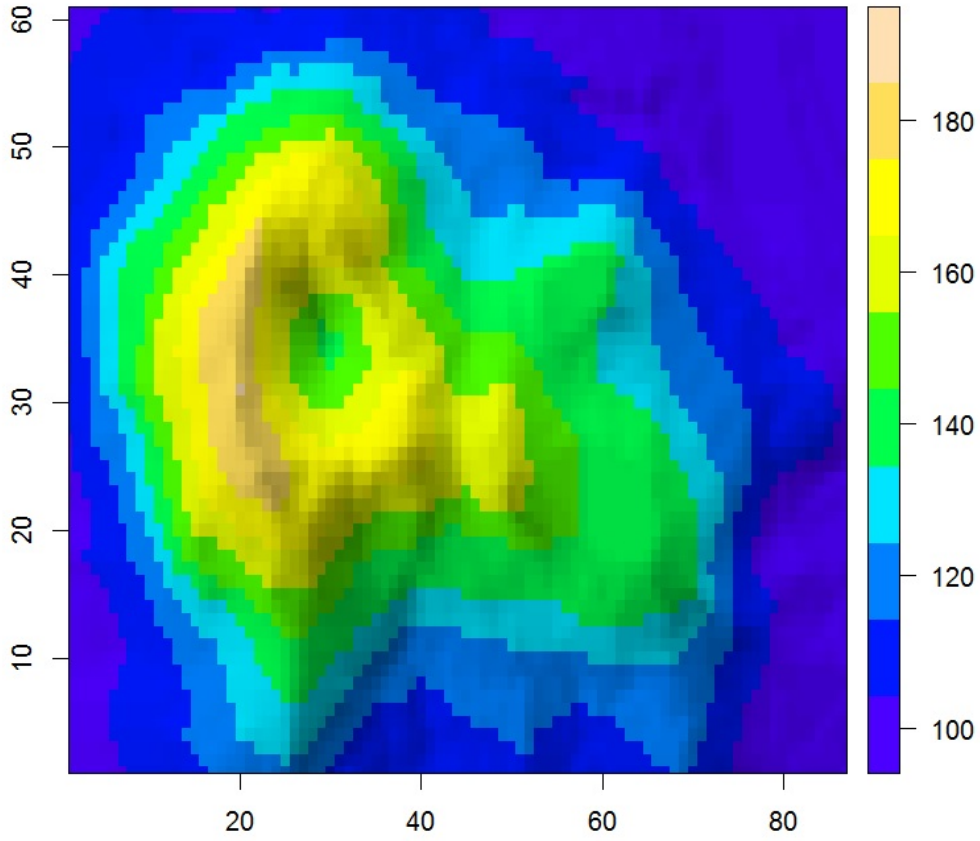
Christophe Chesneau

Université de Caen

Introductions aux structures de données et aux fonctions avec R

Christophe Chesneau

http://www.math.unicaen.fr/~chesneau/



Caen, le 26 Juin 2016

Table des matières

~ Note ~

L'objectif de ce document est de présenter les commandes associées aux structures de données offertes par le logiciel R. Contact : `christophe.chesneau@gmail.com`

Bonne lecture !

1 Structures de données

Notion de data frame

En général, les données d'une étude statistique sont présentées sous forme matricielle. Les lignes représentent les individus observés et les colonnes les variables étudiées. Le logiciel R, pour ses applications les plus évoluées, travaille avec des structures de données appelées data frame.

Construction de data frame

On peut créer une data frame en transformant une matrice numérique lorsque toutes les données sont numériques. On fait :

```
mat = matrix(1:9, ncol = 3)

as.data.frame(mat)
```

Cela renvoie :

```
  V1 V2 V3
1  1  4  7
2  2  5  8
3  3  6  9
```

On peut aussi créer une data frame en concaténant des objets de différentes natures. On fait :

```
age = c(24, 26, 22)

sexe = c("H", "H", "F")

love = c(TRUE, FALSE, TRUE)

enquete = data.frame(age, sexe, love)

enquete
```

Cela renvoie :

```
  age sexe  love
1  24    H  TRUE
2  26    H FALSE
3  22    F  TRUE
```

Désigner des facteurs

On peut souhaiter que certaines colonnes soient considérées comme des facteurs à plusieurs niveaux (ou à plusieurs modalités). Dans ce cas, on fait :

```
sexe = factor(c("H", "H", "F"))
```

Pour connaître les niveaux de ce facteur, on fait :

```
levels(sexe)
```

Cela renvoie : `[1] "F" "H"`. L'ordre ici est lexicographique.

Attribuer des labels aux colonnes

On peut mettre des labels aux colonnes comme pour une liste :

```
names(enquete) = c("Age", "Sexe", "Love")
```

Par défaut, le logiciel R met comme nom, le nom de l'objet qui est dans la data frame. Éventuellement, on peut mettre des labels aux lignes en faisant :

```
row.names(enquete) = c("Jules", "Jim", "Elsa")
```

On aurait aussi pu faire :

```
enquete = data.frame(Age = c(24, 26, 22), Sexe = c("H", "H", "F"),
Love = c(TRUE, FALSE, TRUE))
```

Une data frame peut, comme une liste, contenir d'autres listes, matrices, vecteurs... On peut aussi transformer une data frame en matrice par la commande `as.matrix`. Ceci est même conseillé pour faire des opérations arithmétiques.

Travailler avec une data.frame

On fait :

```
str(enquete)
```

Cela renvoie :

```
'data.frame':   3 obs. of  3 variables:
 $ Age : num  24 26 22
 $ Sexe: Factor w/ 2 levels "F","H": 2 2 1
 $ Love: logi  TRUE FALSE TRUE
```

On obtient ainsi une brève description des variables de `enquete`.

On peut travailler sur une data frame comme sur une matrice.

On considère les commandes :

```
enquete[1:2, 2:3]
```

Cela renvoie :

```
  Sexe  Love
1    H  TRUE
2    H FALSE
```

On a ainsi extrait une une sous-data frame contenant les 1-ère et 2-ème lignes et 2-ème et 3-ème colonnes de la data frame `enquete`.

On considère les commandes :

```
summary(enquete)
```

Cela renvoie :

```
     Age      Sexe      Love
Min.   :22   F:1       Mode :logical
1st Qu.:23   H:2       FALSE:1
Median :24             TRUE :2
Mean   :24              NA's :0
3rd Qu.:25
Max.   :26
```

On obtient alors plusieurs paramètres statistiques des variables considérées.

Une variable peut être appelée en faisant :

```
enquete$Age
```

Cela renvoie : `[1] 24 26 22`

On associe les labels aux colonnes d'une data frame en faisant :

```
attach(enquete)
```

On peut alors appeler directement les variables par leur nom :

```
Age
```

Cela renvoie : `[1] 24 26 22`

On détache les labels des colonnes en faisant :

```
detach(enquete)
```

Construction de tableaux de contingence avec une data frame

Dans la data frame `enquete`, les variables `Sexe` et `Love` sont considérées comme des facteurs.

On considère alors les commandes :

```
attach(enquete)

table(Sexe)
```

Cela renvoie :

```
F  H
1  2
```

On obtient ainsi les effectifs de l'échantillon pour chaque modalité du facteur `Sexe`.

On fait :

```
table(Sexe, Love)
```

Cela renvoie :

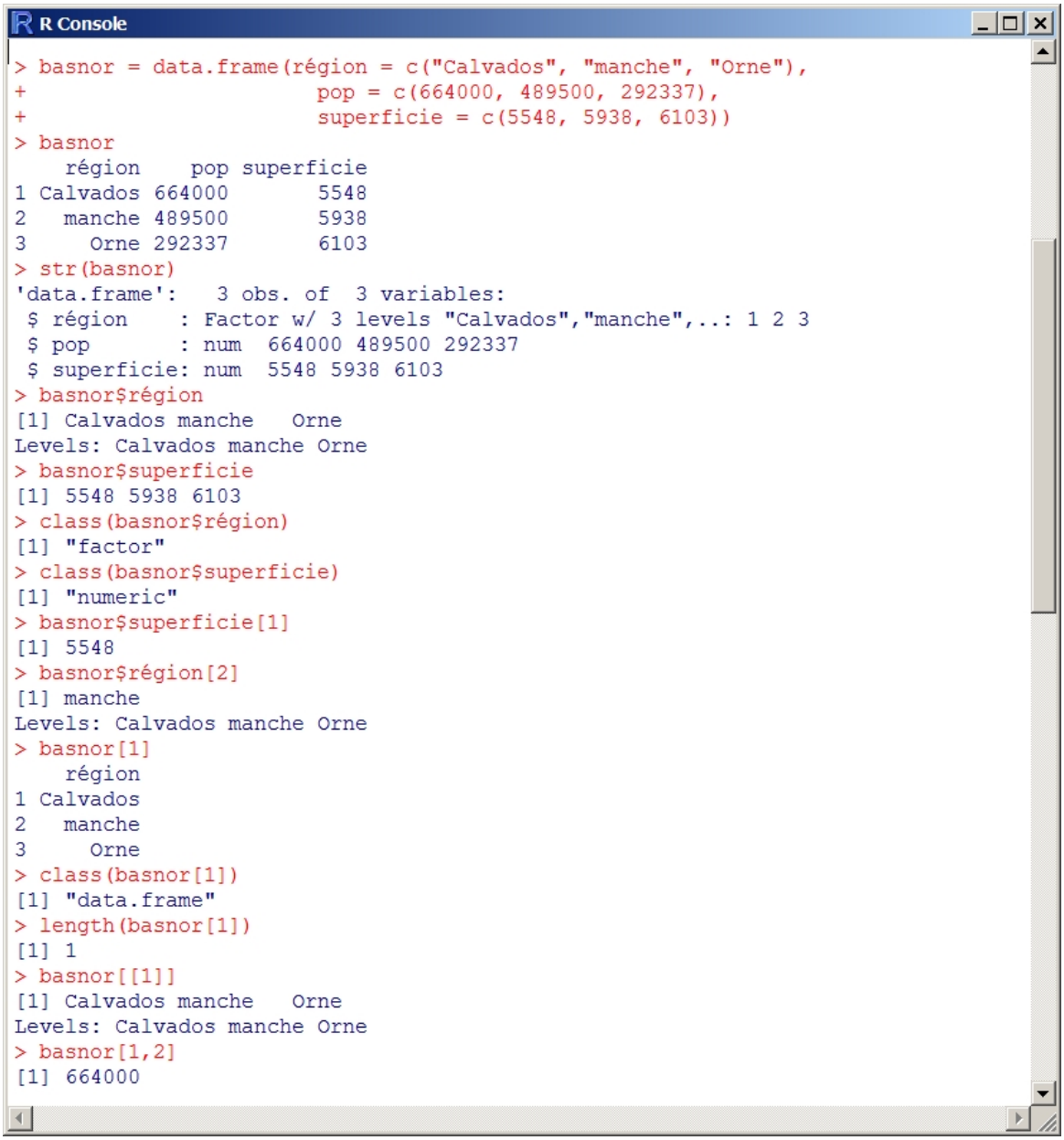
```
  F T
F 0 1
H 1 1
```

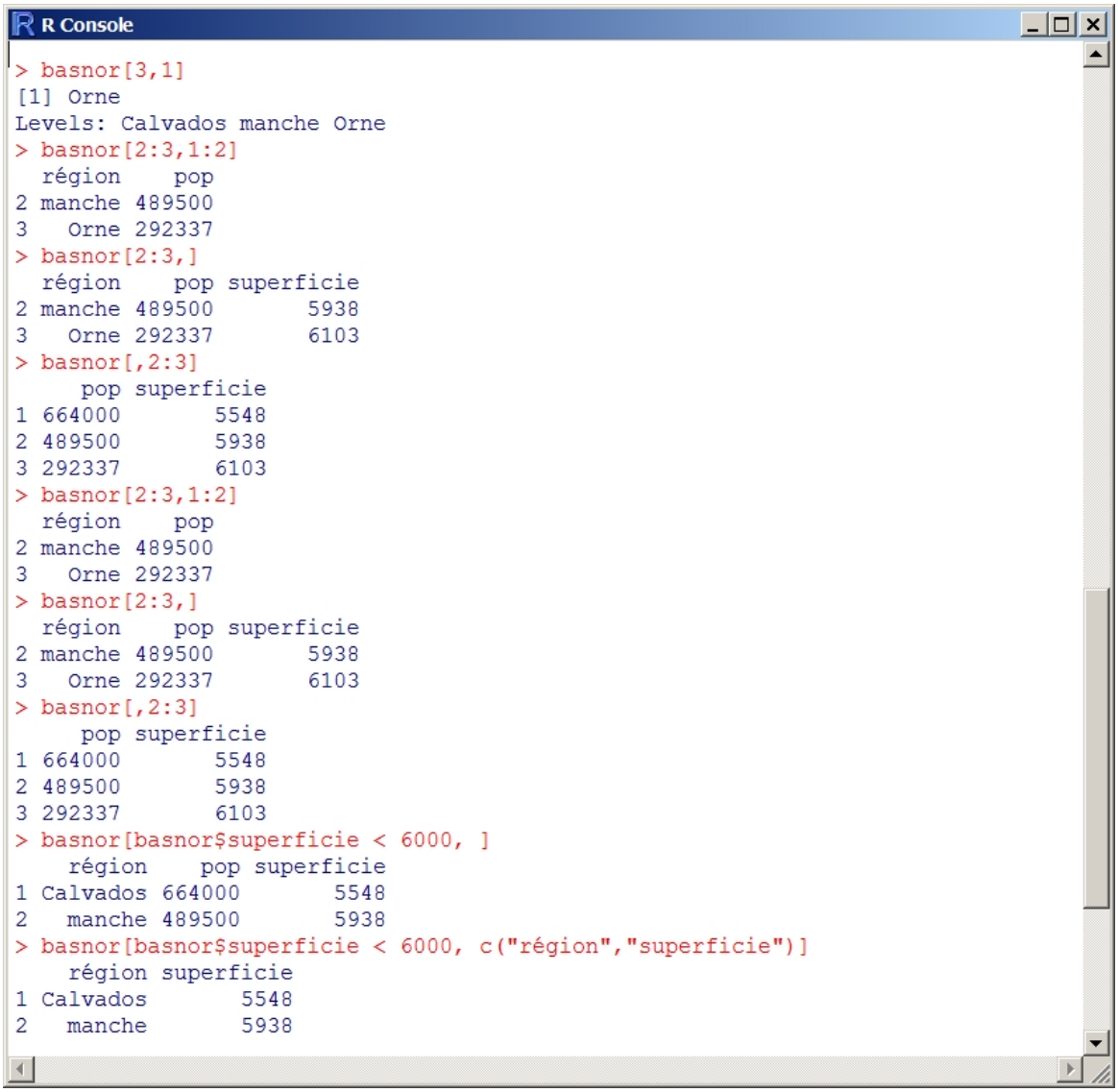
On obtient ainsi les effectifs de l'échantillon pour chaque couple de modalités.

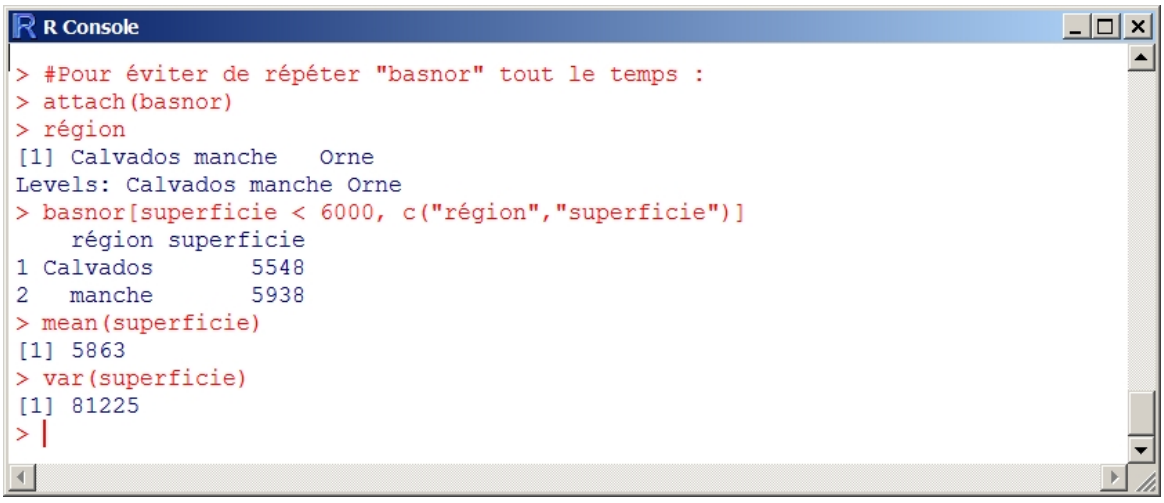
Exemple : data frame "Basse-Normandie"

```
basnor = data.frame(région = c("Calvados", "manche", "Orne"),
pop = c(664000, 489500, 292337), superficie = c(5548, 5938, 6103))
basnor
str(basnor)
basnor$région
basnor$superficie
class(basnor$région)
class(basnor$superficie)
basnor$superficie[1]
basnor$région[2]
basnor[1]
class(basnor[1])
length(basnor[1])
basnor[[1]]
basnor[1,2]
basnor[3,1]
basnor[2:3,1:2]
basnor[2:3, ]
basnor[, 2:3]
basnor[basnor$superficie < 6000, ]
basnor[basnor$superficie < 6000, c("région", "superficie")]
#Pour éviter de répéter "basnor", on aurait pu attacher la data frame :
attach(basnor)
région
basnor[superficie < 6000, c("région", "superficie")]
mean(superficie)
var(superficie)
```

Des consoles R illustrant cet exemple sont présentées ci-dessous :




R Console
> basnor[3,1]
[1] Orne
Levels: Calvados manche Orne
> basnor[2:3,1:2]
région pop
2 manche 489500
3 Orne 292337
> basnor[2:3,]
région pop superficie
2 manche 489500 5938
3 Orne 292337 6103
> basnor[,2:3]
pop superficie
1 664000 5548
2 489500 5938
3 292337 6103
> basnor[2:3,1:2]
région pop
2 manche 489500
3 Orne 292337
> basnor[2:3,]
région pop superficie
2 manche 489500 5938
3 Orne 292337 6103
> basnor[,2:3]
pop superficie
1 664000 5548
2 489500 5938
3 292337 6103
> basnor[basnor$superficie < 6000,]
région pop superficie
1 Calvados 664000 5548
2 manche 489500 5938
> basnor[basnor$superficie < 6000, c("région","superficie")]
région superficie
1 Calvados 5548
2 manche 5938


R Console
> #Pour éviter de répéter "basnor" tout le temps :
> attach(basnor)
> région
[1] Calvados manche Orne
Levels: Calvados manche Orne
> basnor[superficie < 6000, c("région","superficie")]
région superficie
1 Calvados 5548
2 manche 5938
> mean(superficie)
[1] 5863
> var(superficie)
[1] 81225
>

2 Lire des données

Considérons des données présentées dans un fichier texte intitulé `donnees.txt`, lequel se trouve dans un dossier de travail intitulé `TP`. Dans un premier temps, on dirige le logiciel R vers le dossier `TP` avec la commande `setwd`. Voici un exemple :

```
setwd("C:/Users/christophe/Desktop/TP")
```

Si les données sont présentées sous la forme d'un vecteur, alors on fait :

```
donnees = scan("donnees.txt")
```

Si les données sont présentées sous forme matricielle avec en colonnes les variables et en lignes les individus, alors on fait :

```
donnees = read.table("donnees.txt")
```

On obtient alors une data frame. Les colonnes qui sont numériques restent numériques tandis que les colonnes qui sont constituées de chaînes de caractères sont transformées en facteurs.

Si le fichier texte comprend un label au-dessus de chaque colonne, on fait :

```
donnees = read.table("donnees.txt", header = T)
```

Ainsi, les colonnes auront le nom adéquat.

On peut aussi lire un fichier de données en ligne sur internet :

```
donnees = read.table("http://www.math.unicaen.fr/~chesneau/donnees.txt",
header = T)
```

On peut aussi lire un fichier de données disponible dans R :

```
donnees = data(airquality)
```

Les commandes `scan` et `read.table` s'adaptent à toutes les mises en forme des données grâce à de nombreuses options de lecture (`sep`, `dec`, `skip`...). Voir : `help("scan")` et `help("read.table")`.

3 Valeurs manquantes

L'absence de données est indiquée par `NULL`. Une valeur manquante est indiquée par `NA` (pour `Not Available`). On fait :

```
u = c(31, 43, NA, 36, NA)

is.na(u)
```

Cela renvoie : `[1] FALSE FALSE TRUE FALSE TRUE`

Les valeurs manquantes sont alors précisées par des `TRUE`.

On considère les commandes :

```
mean(u, na.rm = TRUE)
```

Cela renvoie : `[1] 36.66667`

On a ainsi fait la moyenne des valeurs des éléments `u` en ignorant les valeurs manquantes.

Si le fichier texte `donnees` contient des données manquantes, on ignore les individus correspondants en utilisant la commande `na.strings` :

```
donnees = read.table("donnees.txt", header = T, na.strings = "NA")
```

Autrement, on peut identifier les lignes présentant des données manquantes en faisant :

```
donnees[!complete.cases(donnees), ]
```

On peut également créer un nouveau jeu de données sans valeur manquante en faisant :

```
donnees2 = na.omit(donnees)
```

4 Les outils pour créer des fonctions

Dorénavant, les commandes étant nombreuses et liées, il est préférable de les écrire dans un fichier texte, puis copier-coller celles-ci dans la fenêtre "R Console".

Les instructions conditionnelles

– La syntaxe de la commande `if` est :

`if ( `*condition*` ) { `*instruction1*` } else { `*instruction2*` }`

Notons que *condition* qui peut valoir `TRUE` ou `FALSE`.

Si on a seulement `if ( `*condition*` ) { `*instruction1*` }`, alors *instruction1* est exécuté si *condition* vaut `TRUE` et il ne se passe rien si condition vaut `FALSE`.

Voir aussi la commande `ifelse` qui s'applique à un vecteur logique.

On fait :

```
a = numeric()
vec = c(1, 3, 2, 4, 7, 5, 6, 9, 8)
if (mean(vec) > 6) { a = 1:9 } else { a = rev(1:9) }
a
```

Cela renvoie : `[1] 9 8 7 6 5 4 3 2 1`

– La syntaxe de `for` est :

`for (`*variable* `in` *vecteur*`) { `*instruction*` }`

avec *instruction* dépendant de la variable. On considère les commandes :

```
b = numeric()
for(i in 1:8) { b[i] = vec[i] + vec[i + 1] }
b
```

Cela renvoie : `[1] 4 5 6 11 12 11 15 15.3`

– La syntaxe de la commande `while` est :

`while ( `*condition*` ) { `*instruction*` }`.

On fait :

```
while (mean(vec) < 10) { vec = vec + 1 }

vec
```

Cela renvoie : `[1] 6 8 7 9 12 10 11 14 13`

– La syntaxe de la commande `repeat` est :

`repeat { `*instruction*` }`

L'instruction est répétée tant que la commande `break` n'est pas exécutée.

On fait :

```
t = 1

repeat { t = 3 * t + 1

if (t >= 15.3) break

}

t
```

Cela renvoie : `[1] 40`

– Voir aussi la commande `next` qui oblige à passer à la boucle suivante.

Construire ses propres fonctions

Syntaxe de la commande `function`. Si on veut, par exemple, construire la fonction `newfonction` qui s'applique à 2 arguments `arg1` et `arg2`, la syntaxe de la commande `function` est :

`newfonction = function (arg1, arg2) { `*instruction*` }`

Ensuite on fait : `newfonction(val1, val2)`

On considère les commandes :

```
arrangement = function(n, k) { factorial(n) / factorial(n - k) }
```

Ensuite on fait :

```
arrangement(7, 3)
```

Cela renvoie : `[1] 210`

On a ainsi calculer l'arrangement : $A_n^k = n!/((n-k)!)$ avec $n = 7$ et $k = 3$.

On fait :

```
courbe = function(x) { sqrt(x + exp(x)) }

courbe(1)
```

Cela renvoie : `[1] 1.928285`

On a ainsi calculer la valeur de la fontion : $f(x) = \sqrt{x + e^x}$ en $x = 1$.

Exemple avancé : fonction `moyennemat`.

1. On construit une fonction `moyennemat` qui dépend d'un argument `x` qui sera une matrice.
2. On ouvre une accolade.
3. On initialise un objet `a` qui sera un vecteur numérique.
4. On déclare un objet `nbcol` qui sera le nombre de colonnes de la matrice ; cet objet aura pour valeur la deuxième dimension de la matrice.
5. On fait un boucle qui part du numéro de la première colonne jusqu'au dernier numéro `nbcol` ; pour un numéro `i`, on affecte à `a[i]` la valeur de la moyenne de la colonne numéro `i`.
6. On retourne l'objet `a` qui est un vecteur.
7. On ferme la fonction avec une accolade.

Ceci donne :

```
moyennemat = function(x) {

a = numeric()

nbcol = dim(x)[2]

for (i in 1:nbcol) { a[i] = mean(x[,i]) }

return(a)

}
```

On fait :

```
mat = matrix(1:9, ncol = 3)

moyennemat(mat)
```

Cela renvoie : `[1] 2 5 8`

Ainsi, la fonction `moyennemat` construit un vecteur de longueur égale au nombre de colonnes de la matrice et dont les éléments seront les moyennes de colonnes.

Exemple avancé : fonction `compter`.

1. On construit une fonction `compter` qui dépend de 2 arguments `a` et `b`.
2. On ouvre une accolade.
3. On initialise un vecteur `d` numérique.
4. On fait une boucle qui part de 1 jusqu'au nombre qui mesure la longueur de `a`.

   Dans cette boucle, on affecte au i-ème élément de `d` la valeur calculée ainsi : on construit un vecteur logique de même longueur que `b` et dont le j-ème élément vaudra `TRUE` s'il est égale à `a[i]` et `FALSE` sinon. On applique la fonction `sum` à ce vecteur logique. Le résultat sera le nombre de `TRUE` dans ce vecteur logique.
5. On donne un nom aux éléments du vecteur `d`.
6. On retourne le vecteur `d`.
7. On ferme la fonction avec une accolade.

Ceci donne :

```
compter = function(a, b) {
d = numeric()
for(i in 1:length(a)) { d[i] = sum(b == a[i]) }
names(d) = as.character(a)
return(d)
}
```

On fait :

```
couleur = c("rose", "vert", "jaune")
fleurs = c("rose", "rose", "vert", "rose", "rose", "jaune", "jaune")
compter(couleur, fleurs)
```

Cela renvoie :

```
 rose  vert jaune
    4     1     2
```

Ainsi, la fonction `compter` donne le nombre de fois où les éléments d'un vecteur apparaissent dans un autre vecteur.

Fonctions s'appliquant à des objets plus complexes

Application d'une fonction aux lignes ou colonnes d'une matrice. On fait :

```
apply(mat, 1, mean)
```

Cela renvoie : `[1] 4 5 6`

On a alors un vecteur de longueur le nombre de lignes de `mat` et dont les élémens sont ici les moyennes des lignes.

On considère les commandes :

```
apply(mat, 2, mean)
```

Cela renvoie : `[1] 2 5 8`

Ainsi, on obtient un vecteur de longueur le nombre de colonnes de `mat` et dont les éléments sont ici les moyennes des colonnes.

On peut remplacer `mean` par n'importe quelle fonction de `R` prédéterminée ou construite, qui s'applique à un vecteur.

Par exemple, on fait :

```
apply(mat, 1, function(x) { sum(x > 5)} )
```

Cela renvoie : `[1] 1 1 2`

On obtient ainsi un vecteur de longueur le nombre de lignes de `mat` dont la valeur des éléments est le nombre d'éléments de la ligne correspondante ayant des valeurs strictement supérieures à 5.

Application d'une fonction aux éléments d'une liste. On considère les commandes :

```
liste = list(c(1, 2), c(3, 1, 2))

sapply(liste, mean)
```

Cela renvoie : `[1] 1.5 2.0`

On a ainsi un vecteur de longueur le nombre d'éléments de `liste` et dont la valeur de chaque élément est ici la moyenne de chaque élément de `liste`.

On considère les commandes :

```
lapply(liste, sd)
```

Cela renvoie :

```
[[1]]
[1] 0.707
[[2]]
[1] 1
```

On obtient ainsi une liste ayant le même nombre d'éléments que `liste` et dont la valeur de chaque élément est ici le résultat de l'application de la fonction `sd` aux éléments de `liste`.

Application d'une fonction aux éléments d'une data.frame. On fait :

```
tapply(Age, Sexe, mean)
```

Cela renvoie :

```
F  H
22 25
```

On obtient ainsi un vecteur constitué de la moyenne des âges pour chaque modalité `F`, `H` de la variable `Sexe`. On peut remplacer `mean` par n'importe quelle fonction de `R` prédéterminée ou construite, qui s'applique à un vecteur.

Fonction qui construit une liste.

On considère les commandes :

```
split(Age, Sexe)
```

Cela renvoie :

```
$F
[1] 22

$H
[1] 24 26
```

On a ainsi construit une liste à partir de deux arguments, le premier étant un vecteur et le second un facteur. Les éléments de la liste sont les niveaux du facteur. Chaque élément contient un vecteur dont les valeurs correspondent aux valeurs du vecteur passé comme premier argument ayant le niveau de l'élément considéré.

5 Exercices

Exercice 1. Créer, avec la commande adéquate, le vecteur numérique `x` dont les éléments sont les valeurs disponibles ici :

`http://www.math.unicaen.fr/~chesneau/norm.txt`

Préciser les paramètres statistiques élémentaires associés à `x`.

Exercice 2. On considère la matrice :

$$M = \begin{pmatrix} 2 & 4 & 8 \\ 2 & 5 & 9 \\ 2 & 3 & 8 \\ 1 & 4 & 7 \\ 1 & 5 & 1 \\ 1 & 2 & 3 \end{pmatrix}.$$

1. Créer la matrice M dans R.
2. Créer une nouvelle matrice `mat` obtenu en ajoutant une nouvelle colonne à M d'éléments : $8, 7, 1, 2, 1, 2$.
3. Transformer votre matrice en data frame.
4. Attribuer aux colonnes les noms : cas, mois, semaine, valeur et aux lignes, les 6 premières lettres de l'alphabet.
5. Extraire la 2-ème, 4-ème et 6-ème lignes de `mat`.
6. Ordonner les lignes de `mat` suivant les valeurs des éléments de `valeur`.

Exercice 3. On travaille avec le jeu de données `airquality`, disponible dans R.

1. Charger les données et comprendre d'où elles émanent.

2. Afficher les noms des variables considérées.
3. Afficher le nombre de lignes et de colonnes.
4. Calculer les paramètres statistiques de base à l'aide de la commande `summary`.
5. Calculer la variance et l'écart-type pour les valeurs de la variable `Temp`.
6. Extraire du jeu de données :
   (a) la 2-ème ligne,
   (b) la 3-ème colonne,
   (c) les lignes 1, 2 et 4 avec une seule commande `c()`,
   (d) les lignes 3 à 6 avec la commande `:`,
   (e) tout sauf les colonnes 1 et 2,
   (f) toutes les lignes ayant une température supérieure à 90.
7. Insérer une nouvelle variable `TooWindy` dans le jeu de données `airquality` qui affiche `TRUE` pour chaque valeur de `Wind` supérieure à 10, et `FALSE` sinon. Puis supprimer cette nouvelle variable.
8. Créer un nouveau jeu de données correspondant à `airquality` privée des lignes de `Ozone` contenant des valeurs manquantes.
9. Installer le package `ggplot2`. Reproduire et analyser les commandes suivantes :

```
library(ggplot2)
qplot(Temp, Ozone, data = airquality, colour = Month)
```

Exercice 4. Le fichier `donneesconsommation.txt` contient 300 résultats bruts d'une enquête sur la consommation des ménages en 1995 dont le résumé se trouve dans le document de l'INSEE "Enquête sur la consommation des ménages en 1995". Pour simplifier le territoire a été divisé en 3 grandes régions (GR1, GR2 et GR3) correspondant au regroupement de plusieurs Z.E.A.T. (Zones d'Étude et d'Aménagement du Territoire) suivant la nomenclature de l'INSEE :

- GR1 : Région parisienne, Bassin parisien,
- GR2 : Nord, Ouest, Sud-Ouest,
- GR3 : Est, Centre Est, Méditerranée.

On a déterminé par ailleurs deux catégories de ménage TA1 et TA2 :

- TA1 : ménage dont la personne de référence a un âge inférieur à 55 ans.
- TA2 : ménage dont la personne de référence a un âge supérieur ou égal à 55 ans.

Dans chacune des 3 grandes régions, on a interrogé 50 ménages appartenant à TA1 et 50 ménages appartenant à TA2. À chaque ligne du fichier correspond un ménage.

- 1-ère colonne : Age = TA1 ou TA2 (tranche d'âge),
- 2-ème colonne : Region = GR1, GR2 ou GR3 (grande région),
- 3-ème colonne : Revenu = revenu annuel du ménage (en euros),
- 4-ème colonne : Loisirs = dépense annuelle en loisirs (en euros),
- 5-ème colonne : Pates = dépense annuelle en pâtes alimentaires (en euros),
- 6-ème colonne : Cafe = AC, BC ou MC marque de café préférée du ménage (BC = "Bon café", MC = "Mon café", AC= "Autre café"),
- 7-ème colonne : Enfants = 0,1,2,... = nombre d'enfants de la personne de référence du ménage (mais ne vivant pas obligatoirement avec les ressources du ménage),
- 8-ème colonne : Magnetoscope = 0,1 (0 si le ménage ne possède pas de magnétoscope, 1 sinon),
- 9-ème colonne : Lavevaisselle = 0,1 (0 si le ménage ne possède pas de lave-vaisselle 1 sinon).

1. Construire une data frame `conso.dat` en faisant :

```
conso.dat = read.table("http://www.math.unicaen.fr/~chesneau/
donneesconsommation.txt", header = T)
```

2. Reproduire et commenter les commandes suivantes :

```
attach(conso.dat)
str(conso.dat)
summary(conso.dat)
```

3. Extraire les données des individus dont la marque de café préférée est AC et ayant un revenu annuel inférieur à 15000 euros.

4. Extraire les données des individus ayant plus de 3 enfants, une dépense annuelle en pâtes alimentaires supérieure à 60 euros et un revenu annuel inférieur à 20000 euros.

Exercice 5. Créer dans R la fonction $f : (\mathbb{N}^*)^2 \to \mathbb{N}^*$ définie par :

$$f(m,n) = \sum_{k=1}^{n} k^m.$$

Exercice 6. Créer dans R la fonction $f : \mathbb{R}^n \to \mathbb{R}^n$ définie par :

$$f(x_1,\dots,x_n) = \left(x_1, \frac{x_2^2}{2}, \dots, \frac{x_n^n}{n}\right).$$

Exercice 7. Créer dans R la fonction $f : \mathbb{R}^n \to \mathbb{R}^{n-2}$ définie par :

$$f(x_1,\dots,x_n) = \left(\frac{x_1+x_2+x_3}{3}, \frac{x_2+x_3+x_4}{3}, \dots, \frac{x_{n-2}+x_{n-1}+x_n}{3}\right).$$

Exercice 8. Créer dans R la fonction $f : \mathbb{R} \to [0,\infty[$ définie par :

$$f(x) = \begin{cases} \dfrac{1}{10} e^{-\frac{x}{10}} & \text{si } x \geq 0, \\ 0 & \text{sinon.} \end{cases}$$

Exercice 9. Créer dans R la fonction $f : \mathbb{N}^2 \times [0,1] \to [0,1]$ définie par :

$$f(k,n,p) = \binom{n}{k} p^k (1-p)^{n-k}.$$

Exercice 10. Créer dans R la fonction $f : \mathbb{N} \times [0,\infty[\to [0,1]$ définie par :

$$f(k,\lambda) = e^{-\lambda} \frac{\lambda^k}{k!}.$$

Exercice 11. On considère la suite de réel $(u_n)_{n\in\mathbb{N}^*}$ telle que $u_0 = 0.5$ et, pour tout $n \geq 1$,

$$u_n = 1 + 0.25u_{n-1}^2.$$

Construire une fonction `Loop` d'argument `m` qui retourne un vecteur contenant les `m` premières valeurs de la suite : u_1, u_2, ..., $u_{\texttt{m}}$. Utiliser cette fonction pour appuyer le fait que $(u_n)_{n\in\mathbb{N}^*}$ converge vers 2.

6 Solutions

Solution 1. On fait :

```
x = scan("http://www.math.unicaen.fr/~chesneau/norm.txt")
```

```
summary(x)
```

Cela renvoie :

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-3.19300 -0.59960  0.03010  0.03493  0.74610  3.02300
```

Solution 2.

1. On fait :

   ```
   M = matrix(c(2, 2, 2, 1, 1, 1, 4, 5, 3, 4, 5, 2, 8, 9, 8, 7, 1, 3), ncol = 3)
   ```

2. On fait : `mat = cbind(M, c(8, 7, 1, 2, 1, 2))`

3. On fait : `mat = as.data.frame(mat)`

4. On fait :

   ```
   colnames(mat) = c("cas", "mois", "semaine", "valeur")
   rownames(mat) = letters[1:6]
   mat
   ```

5. On fait : `mat[c(2, 4, 6), ]`

6. On fait : `mat = mat[order(mat$valeur), ]`

Solution 3.

1. On fait :

   ```
   data(airquality)
   ?airquality
   ```

2. On fait : `names(airquality)`

3. On fait : `dim(airquality)`

4. On fait : `summary(airquality)`

5. On fait :

   ```
   var(airquality$Temp)
   ```

   ```
   sqrt(var(airquality$Temp))
   ```

6. (a) On fait : `airquality[2, ]`

   (b) On fait : `airquality[ ,3]`

   (c) On fait : `airquality[c(1, 2, 4), ]`

   (d) On fait : `airquality[3:6, ]`

   (e) On fait : `airquality[ ,-c(1, 2)]`

   (f) On fait : `airquality[airquality$Temp>90, ]`

7. On fait :

   ```
   airquality$TooWindy = airquality$Wind >= 10.
   ```

   ```
   airquality$TooWindy = NULL
   ```

8. On fait :

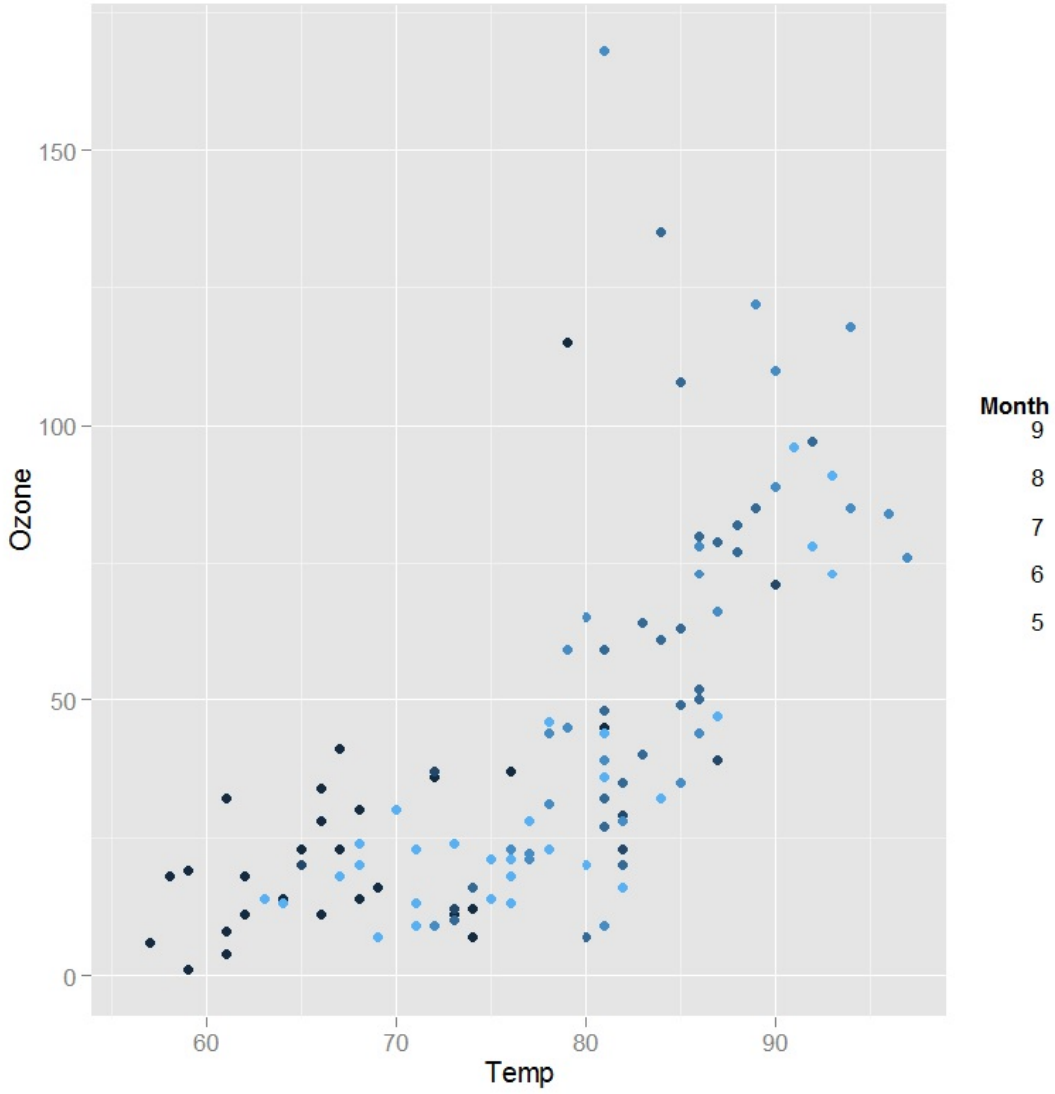
   ```
   aq = airquality[!is.na(airquality$Ozone),]
   ```

9. On fait :

   ```
   library(ggplot2)
   ```

   ```
   qplot(Temp, Ozone, data = airquality, colour = Month)
   ```

Cela renvoie :



Solution 4.

1. On fait :

   ```
   conso.dat = read.table("http://www.math.unicaen.fr/~chesneau/
   donneesconsommation.txt", header = T)
   ```

2. On fait :

   ```
   attach(conso.dat)
   ```

   Cela attache les labels aux colonnes.

On fait :

```
str(conso.dat)
```

On obtient ainsi une brève description des variables.

On fait :

```
summary(conso.dat)
```

On obtient ainsi des éléments statistiques pour les variables considérées.

3. On fait : `conso.dat[Cafe=="AC" & Revenu <15000, ]`
4. On fait : `conso.dat[Enfants >3 & Pates >60 & Revenu <20000, ]`

Solution 5. On fait :

```
fonc1 = function(m, n){
x = 1:n
y = x^m
sum(y)
}
```

Solution 6. On fait :

```
fonc2 = function(x){
n = length(x)
x^(1:n) / (1:n)
}
```

Solution 7. On fait :

```
fonc3 = function(x){
n = length(x)
( x[1:(n - 2)] + x[2:(n - 1)] + x[3:n] ) / 3
}
```

Solution 8. On fait :

```
fonc4 = function(x) {
if (x<0)
0
else
(1 / 10) * exp(-x / 10)
}
```

ou plus simplement :

```
fonc4 = function(x) {
ifelse(x >=0, (1 / 10) * exp(-x / 10), 0)
}
```

Solution 9. On fait :

```
fonc5 = function(k, n, p) {
choose(n, k) * p^k * (1 - p)^(n - k)
}
```

Solution 10. On fait :

```
fonc6 = function(k, lambda)  {
exp(-lambda) * (lambda^k / factorial(k))
}
```

Solution 11. On fait :

```
Loop = function(n) {
x = numeric()
x[1] = 0.5
for(j in 2:n)
x[j] = 1 + 0.25 * x[j-1]^2
x
}
```

On fait, par exemple :

```
Loop(500)
```

On remarque alors que la suite se stabilise vers une valeur très proche de 2.