HAI718 Probabilité et statistiques

Exploration de quelques fonctions de R

1 Quelques fonctions et structures de contrôles utiles

Exercice 1 *Tester les fonctions* `c`*, l'opérateur* `:`*, les fonctions* `seq` *et* `rep` *:*

```
> c(1,4,5)
> 1:6
> seq(1,5,by=0.5)
> rep(3,5)
```

Produire les séquences suivantes à la main et en utilisant les opérations précédentes :

```
1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
10   3   4   5   6  10 100 100  10  20  30  40
```

Exercice 2 *Consulter l'aide de la fonction* `vector` *(rappel, on demande l'aide avec help ou en mettant un ? devant la fonction surlaquelle on demande l'aide). Tester :*

```
a <- vector(length=10, mode="numeric")
```

Comment produit-on un vecteur de caractères de taille 50 ?

Exercice 3 *Sélection d'éléments dans un vecteur : tester les commandes suivantes*

```
> x<-10:15
> x[2]
> x[c(2,4)]
> x[-4]
```

1. *Créer un vecteur x contenant 1, 4 et 5*
2. *Créer un vecteur y contenant les chiffres impairs de 1 à 9*
   (a) *afficher le deuxième élément de y*
   (b) *afficher tous les éléments de y sauf le deuxième*
3. *Créer un vecteur xy contenant le 1er, le 4ème et le 5ème élément de y (utiliser le vecteur x pour cela)*
4. *Afficher les éléments 2 à 4 de y*

Exercice 4 *Tester :*

```
> x <- 1:3
> x**2
> x/c(2,4,6)
> y <- 1:5
> x*y
```

Dans la dernière instruction, que fait R et à quoi correspondent les chiffres donnés ?

Exercice 5 *Les fonctions* `sort, rev, order, rank`*.*

1. *Lire les aide correspondant à ces fonctions.*
2. *Comment afficher en ordre décroissant les éléments du vecteur* `1:10`*, avec chacune des trois premières fonctions ?*

Exercice 6 *Aggrégation :* `cbind, rbind`*. Tester :*

```
> a <- 1:2
> b <- 3:4
> rbind(a,b)
> cbind(a,b)
```

Il est possible de filtrer des données dans un vecteur en utilisant la syntaxe suivante : `nom_vecteur[condition_a_verifier]`. Exemple :

```
> x <- 1:10
> y <- x[x>7]
> y
[1]  8  9 10
```

Ainsi, si on veut compter le nombre d'éléments positifs d'un vecteur, on utilise un filtre et la fonction de comptage `length`.

Exercice 7

1. *Engendrer un vecteur x de 100 éléments suivant la loi normale centrée réduite*
2. *Compter son nombre d'éléments strictement positifs*
3. *Prendre le logarithme de ce vecteur, dans un autre vecteur y (fonction* `log`*). Que constatez-vous ?*
4. *Les données manquantes sont détectées par la fonction* `is.na`*. Compter le nombre de données manquantes de y*
5. *Prendre la moyenne de y avec la fonction* `mean`*. Quelle est l'option de cette fonction permettant de ne pas tenir compte des données manquantes ?*

2 Bibliothèques de données de R

R possède des bibliothèques prédéfinies de données statistiques. Elles peuvent être pratique lorsque l'on écrit un programme et que l'on veut disposer de jeux de données pour tester ce programme. On va s'en servir ici pour découvrir quelques fonctions sur les jeux de données (dataframes) en R. On travaillera sur le jeu de données `airquality`, disponible de base dans R. Stockez ce jeu de données dans une variable `air1`.

2.1 Manipulations de base sur les dataframes

Pour récupérer une colonne, on utilise le caractère `$` : `air1$Ozone` produit le vecteur des données de la colonne Ozone. On peut aussi référencer une colonne par son numéro ou son nom : `air1[,1]`, `air1[,"Ozone"]`, idem pour une ligne : `air1[3,]`.

L'extraction d'une sous-base se fait avec la fonction `subset` qui prend en paramètre le jeu de donnée et le critère de sélection. Exemple :Le jeu de donnée extrait dont la colonne Temp a une valeur > 92 est produit grâce à :

```
> subset(air1,Temp>92)
    Ozone Solar.R Wind Temp Month Day
42     NA     259 10.9   93     6  11
120    76     203  9.7   97     8  28
121   118     225  2.3   94     8  29
122    84     237  6.3   96     8  30
123    85     188  6.3   94     8  31
126    73     183  2.8   93     9   3
127    91     189  4.6   93     9   4
```

ou, de la même manière que pour les vecteurs (notez que la sélection se fait sur les lignes ici, le critère sur la colonne étant laissé vide) :

```
> air1[air1$Temp>92,]
    Ozone Solar.R Wind Temp Month Day
42     NA     259 10.9   93     6  11
120    76     203  9.7   97     8  28
121   118     225  2.3   94     8  29
122    84     237  6.3   96     8  30
123    85     188  6.3   94     8  31
126    73     183  2.8   93     9   3
127    91     189  4.6   93     9   4
```

Pour transformer les données d'un jeu de données, on utilise la fonction `transform`, qui permet par exemple d'ajouter une colonne. Voici trois façons équivalentes de rajouter une colonne `logtemp` contenant le log de la température :

```
> air2 <- transform(air1,logTemp=log(Temp))
> air3 <- air1
> air3$logTemp <- log(air3$Temp)
> air4 <- cbind(air1,logTemp=log(air1$Temp))
```

Au passage, apprenons à nous servir d'une structure de contrôle fort utile en R, à savoir l'alternative `ifelse` :

```
> x <- 6:-4
> x
 [1]  6  5  4  3  2  1  0 -1 -2 -3 -4
> sqrt(x)
 [1] 2.449490 2.236068 2.000000 1.732051 1.414214 1.000000 0.000000      NaN
 [9]      NaN      NaN      NaN
Warning message:
NaNs produced in: sqrt(x)
> sqrt(ifelse(x>=0,x,NA))
 [1] 2.449490 2.236068 2.000000 1.732051 1.414214 1.000000 0.000000       NA
 [9]       NA       NA       NA
```

Exercice 8 *Manipulation du dataframe* `airquality`

1. *Créer à partir de* `air1` *un jeu de données* `air2` *dans lequel :*
   — *Il n'y a plus de données manquante dans la colonne Ozone,*
   — *La température est* ≤ 94 *degrés Fahrenheit.*
2. *Créer à partir de* `air1` *un jeu de données* `air3` *dans lequel la donnée Ozone n'est pas manquante. Rajouter à* `air3` *une colonne dont la valeur est 1 si*
   — *on est au mois de juin (6),*
   — *et la température est* > 78,
   et 0 sinon. (L'égalité se teste avec `==` *et l'opérateur logique "et" se traduit par le symbole* `&`*)*
3. *Combien de lignes donnent la valeur 1 (obtenir l'information avec la fonction* `length` *et une sélection) ?*

Pour ordonner selon une ou plusieurs variables, on utilise la fonction `order` :

```
> air1[order(air1$Month,air1$Day),]
```

Exercice 9 *Sur le jeu de données* `iris`*, ordonner les données selon la longueur des sépales, puis la longueur des pétales.*

La fonction `match` permet de tester si une donnée est présente dans le jeu de données.

```
> month.name
 [1] "January"   "February"  "March"     "April"     "May"       "June"
 [7] "July"      "August"    "September" "October"   "November"  "December"
> match(c("Mai","May"),month.name,nomatch=0)
[1] 0 5
```

Exercice 10

1. *Que produit la commande suivante :*

   ```
   > month.name[match(air1$Month,1:12)]
   ```

2. *Rajouter à* `air1` *une colonne avec le nom du mois.*

2.2 Fonctions statistiques

2.2.1 Un peu de graphiques

La fonction `pairs` permet de représenter les variables 2 à 2.

```
> data(airquality)
> is(airquality)
> air <- airquality
> names(air)
> help(air)
> air <- na.omit(air)
> pairs(air, panel = panel.smooth, main="airquality",col="blue",pch=3*3)
```

On pourrait représenter quatre variables sur un même graphique. On utilisera l'axe des abscisses et des ordonnées pour représenter les variables Ozone et Wind, la taille des points pour représenter la variable Temperature et la couleur pour la variable Month.

La taille des points d'un graphique est généralement comprise entre 0.1 et 2. Pour se mettre dans cette gamme et mieux visualiser l'effet des températures on effectue l'opération suivante. Observer le raccourci `air$T` pour `air$Temp`...

```
> air$T <- (air$T - min(air$T) +1)/10
```

On peut alors lancer le graphique après avoir changé la palette des couleurs.

```
> palette()
> palette(rainbow(5))
> palette()
> plot(air$O,air$W,cex=air$T,col=air$M-4, pch=16)
> text(air$O,air$W,lab=air$M,cex=0.5)
> title("Parametres atmospheriques (New-York, 1973)")
  legend(130,20,legend=c("Mai","Juin","Juillet","Aout","Sept."),fill=palette())
```

2.2.2 Moyenne

La moyenne d'un vecteur s'obtient avec la fonction `mean`. La fonction `tapply` permet de calculer la moyenne selon certains paramètres.

Exercice 11 *Lire l'aide de cette fonction, et reprendre le jeu de données* `iris`*. Calculer la moyenne de la longueur des sépales pour chaque catégorie d'espèce.*

La fonction `mean` permet de gérer les données manquantes d'une façon différente de l'élimination pure et simple des lignes qui en contiennent : on peut remplacer ces données par la moyenne des autres données.

Exercice 12 *Réaliser cette opération sur la colonne Ozone de* `airquality` *(on utilisera la fonction* `apply` *simple).*

2.2.3 Ecart-type

La fonction `sd` permet d'obtenir l'écart-type d'un échantillon.

Exercice 13

1. *Produire un vecteur de 100 données suivant la loi normale de moyenne 4 et d'écart-type 0,5. Calculer la moyenne et l'écart-type sur cet échantillon.*
2. *Calculer l'écart relatif entre la moyenne et la moyenne empirique, ainsi que le rapport entre l'écart-type et l'écart-type empirique.*

2.2.4 Boucles

On veut généraliser l'expérience précédente et stocker les données dans un dataframe. Celui-ci devra comporter 6 colonnes : Moyenne, EcartType, MoyenneEmp, EcartTypeEmp, EcartRelMoyenne, RapportEcartType. Pour engendrer le vecteur Moyenne et le vecteur EcartType, on utilisera `runif`, en mutlipliant les valeurs par 10 pour la moyenne. On veut obtenir 50 lignes. Pour créer les 4 autres vecteurs, on utilise la fonction `vector`, cf. tout au début du TP.

Enfin, pour calculer les valeurs de chaque ligne, on utilise une boucle `for` :

```
for(i in 1:50){
   u<- rnorm(100);
   MoyenneEmp[i]<-mean(u);
   EcartTypeEmp[i] <- sd(u)
}
```

Exercice 14

1. *Créer tous les vecteurs demandés.*
2. *Réaliser le dataframe demandé (utiliser la fonction* `data.frame`*).*
3. *Sauver le dataframe dans un fichier (exemple :* `write.table(blob,"blob.dat")` *).*