Initiation à R et découverte de la variance

Mathieu Basille

27 mai 2010

"A new statistic proves that 73% of all statistics are pure inventions." — J.J.A. Weber

Cette fiche donne quelques bases d'utilisation du logiciel R. La première section est générique et s'attache à l'utilisation proprement dite du logiciel, aux types de données[1] et aux ressources disponibles. La deuxième section expose quelques possibilités graphiques du logiciel. Enfin, une approche de l'analyse de variance est proposée. Pour un support plus détaillé, on pourra se référer par exemple à :

- An Introduction to R [en][2]
- Enseignements de statistique en biologie [fr][3]

Table des matières

[1] L'importation des données n'est pas abordée dans ce tutoriel.
[2] http://cran.r-project.org/doc/manuals/R-intro.pdf
[3] http://pbil.univ-lyon1.fr/R/enseignement.html

1 Première utilisation

1.1 L'environnement de R

R est un logiciel libre. Il est donc *librement* utilisable[4], *librement* distribuable et *librement* modifiable. R est en outre un environnement de statistiques basé sur un langage en ligne de commande.

Lorsque l'on lance le logiciel (sous Windows), une fenêtre s'ouvre avec une console, dans lequel il nous est possible de taper des commandes. Quelques menus sont disponibles, mais ils seront dans l'ensemble très peu utilisés (faire un tour dans le menu `Aide > Console` est cependant une bonne idée).

Pour bien démarrer, on vérifie dans quel répertoire de travail se place R au démarrage (*get working directory*) :

```
getwd()
```

Il existe plusieurs méthodes pour modifier ce répertoire : on peut par exemple utiliser le menu dédié (`Fichier > Changer le répertoire courant`), ou bien copier un raccourci de R dans son répertoire (en laissant vide la case "Démarrer dans") ou encore à l'aide de la ligne de commande suivante :

```
setwd("C:/Documents and Settings/Moi/Mes documents/Mon répertoire")
```

La ligne de commande accepte directement tous types de calculs, ainsi que des appels de fonctions :

```
2 + 2
pi
1:5
sqrt(26)
rnorm(10)
```

1.2 Les types de données

Les objets (données, fonctions, résultats, etc.) sont stockés directement dans le répertoire de travail, avec la suite d'assignation `<-` :

```
truc <- 2 + 2
truc
```

`truc` est un objet de type numérique. Il est de taille 1.

```
class(truc)
length(truc)
```

On peut associer plusieurs nombres dans le même objet pour former un vecteur grâce à la fonction `c` (pour "combine") :

```
truc <- c(42, 2 + 2, sqrt(26))
truc
class(truc)
length(truc)
```

Les autres grands types de données sont les matrices ("matrix"), les tableaux ("data frame") et les listes ("list") :

[4] ...et disponible gratuitement à l'adresse http://www.r-project.org/

```
mat <- matrix(1:20, nrow = 5)
mat
df <- data.frame(A = 1:5, B = seq(0, 1, length.out = 5))
df
class(df$C)
lis <- list(l1 = 1:10, l2 = seq(0, 1, length.out = 5))
lis
```

La structure de données la plus courante est le tableau. Quelques manipulations sont très précieuses pour avoir un aperçu des données exploitées[5]. On prend le cas d'un tableau avec 4 variables A, B, C et D de types différents :

```
df <- data.frame(A = 1:20, B = seq(0, 1, length.out = 20),
    C = sample(c("Grand", "Moyen", "Petit"), 20, replace = TRUE),
    D = rnorm(20))
head(df)
tail(df)
str(df)
summary(df)
names(df)
dim(df)
table(df$C)
```

On peut également stocker une fonction dans un objet :

```
carre <- function(x) print(paste("X au carré vaut",
    x^2))
```

Tous les objets sont stockés dans l'espace de travail. Il est possible de lister les objets, de les supprimer, voire de sauvegarder l'espace de travail en intégralité[6] :

```
ls()
rm(mat)
save.image(file = "Travail.RData")
```

1.3 Aides en tous genres

Enfin, pour chercher de l'aide, on peut soit demander au prof (avec l'incertitude que cela comporte), soit utiliser les ressources propres de R :

```
help("sqrt")
`?`(sqrt)
apropos("test")
help.search("Linear Model")
RSiteSearch("An Introduction to R")
```

`help` et `?` sont des fonctions équivalentes pour obtenir la fiche d'aide détaillée d'une fonction (avec généralement des exemples). `apropos` recherche toutes les fonctions qui contiennent l'élément entre guillemets dans leur nom. `help.search` recherche dans les fiches d'aide les fonctions qui correspondent à la requête. `RSiteSearch` envoie la requête au navigateur pour effectuer une recherche sur le contenu en ligne de R(listes de discussion, fiches d'aides, documents divers ; en anglais).

[5] **Remarque :** ces fonctions sont généralement valables pour tous les types de données mais ne seront illustrées ici que pour les tableaux.

[6] Cela vous est de toutes façons proposé lorsque vous quittez la session.

2 Représentations graphiques

2.1 Nuages de points

La représentation graphique la plus simple est le nuage de points en 2 dimensions. On prépare 20 abscisses aléatoires à partir d'un tirage dans une loi normale centrée réduite. Les ordonnées correspondants sont construites sur le modèle :

$$x = 2 \times y + \epsilon$$

avec ϵ étant un bruit lui-même tiré d'une loi normale centrée mais d'écart type 0.5.

```
XX <- rnorm(20)
YY <- XX * 2 + rnorm(20, 0, 0.5)
plot(XX, YY)
abline(0, 2)
```

On a rajouté ici une ligne droite à l'aide de la fonction `abline`, avec comme ordonnée à l'origine 0 et coefficient directeur 2, ce qui correspond au modèle qui a servi a générer les données. À l'aide d'une régression linéaire, on peut toutefois vérifier l'adéquation des données observées au modèle :

```
lm1 <- lm(YY ~ XX)
abline(lm1, lty = 2, lwd = 2)
```

2.2 Boîtes à moustaches

Les boîtes à moustache sont très simplement appelées à l'aide de la fonction `boxplot` :

```
boxplot(df$D)
boxplot(df$D ~ df$C)
```

2.3 Histogrammes

Soit un échantillon aléatoire de 1000 tirages au sein d'une loi normale de moyenne 0 et d'écart type 1. Il est très facile de créer un histogramme de cette distribution :

```
tir <- rnorm(1000)
hist(tir)
hist(tir, main = "Loi normale centrée réduite", br = -50:50/10)
hist(tir, main = "Loi normale centrée réduite", br = c(-5,
    -2, -1, -0.25, 0, 0.5, 1.5, 3, 5))
hist(tir, main = "Loi normale centrée réduite (densité)",
    br = -50:50/10, freq = FALSE)
lines(-50:50/10, dnorm(-50:50/10), lwd = 3, col = "red",
    lty = 2)
```

2.4 Graphique croisé

À quoi correspond le résultat de l'appel suivant ?

```
plot(df)
```

NB : voici un indice pour mieux comprendre ce qu'il se passe :

```
unclass(df$C)
```

3 La variance chez les iris

3.1 Présentation

Nous allons maintenant examiner un des jeux de données les plus célèbres de la statistique : les *iris de Fisher*[7]. Ce jeu de données est disponible directement dans dans l'objet `iris`. Vérifier tout d'abord le type d'objet, puis examinez-en la structure.

```
class(iris)
head(iris)
str(iris)
```

Ce jeu de données contient différentes informations mesurées sur 50 fleurs de 3 espèces d'iris différentes.

```
table(iris$Species)
```

Toutes les informations sont détaillées dans la fiche d'aide du jeu de données :

```
`?`(iris)
```

Vérifiez que la longueur des sépales augmente avec la longueur des pétales, quelle que soit l'espèce.

```
plot(iris$Sepal.Length, iris$Petal.Length)
```

Est-ce que ce résultat est également vrai pour chaque espèce ?

```
plot(iris$Sepal.Length, iris$Petal.Length, col = as.numeric(iris$Species))
coplot(iris$Petal.Length ~ iris$Sepal.Length | iris$Species,
    columns = 3)
```

3.2 Comparaison de moyennes

Nous nous intéressons désormais à la largeur des pétales uniquement (`Petal.Width`). Vérifiez visuellement que ces largeurs sont différentes selon l'espèce.

```
boxplot(iris$Petal.Width ~ iris$Species)
```

Nous nous intéressons ici aux moyennes :

```
mean(iris$Petal.Width[iris$Species == "setosa"])
by(iris$Petal.Width, iris$Species, mean)
iris$Mean <- rep(by(iris$Petal.Width, iris$Species, mean),
    each = 50)
```

Par le calcul, ces moyennes paraissent clairement différentes. En première approche, on peut tester 2 à 2 ces différences à l'aide de tests t (le paramètre `var.equal = TRUE` indique que les variances sont égales) :

```
t.test(iris$Petal.Width[1:50], iris$Petal.Width[51:100],
    var.equal = TRUE)
t.test(iris$Petal.Width[51:100], iris$Petal.Width[101:150],
    var.equal = TRUE)
t.test(iris$Petal.Width[1:50], iris$Petal.Width[101:150],
    var.equal = TRUE)
```

Bien que les différences soient significatives, nous sommes limités ici par l'accumulation de tests (si on avait 10 classes, il faudrait faire 45 tests !) avec des risques d'erreurs qui se multiplient ! L'approche développée par l'analyse de variance (ANOVA) permet d'effectuer cette comparaison de moyennes en un seul test.

[7] Aussi appelés *iris d'Anderson...*

3.3 Décomposition de la variance

En première année, nous avons vu comment une comparaison de moyennes s'appuyait sur une étude de variance. Nous avons également vu comment décomposer la variance totale en variance intra et variance inter. Nous nous intéressons d'abord à la variance totale :

```
VT <- 1/nrow(iris) * sum((iris$Petal.Width - mean(iris$Petal.Width))^2)
```

Comparez ce résultat avec celui de la fonction `var` (aidez-vous de la fiche d'aide de `var` pour mieux comprendre) :

```
var(iris$Petal.Width)
VT * nrow(iris)/(nrow(iris) - 1)
```

Nous aurions pu créer une fonction pour calculer la variance d'un échantillon sur N plutôt que sur $N - 1$:

```
varN <- function(x) 1/length(x) * sum((x - mean(x))^2)
varN(iris$Petal.Width)
```

Nous pouvons ensuite procéder à la décomposition de la variance :

```
VB <- 1/nrow(iris) * sum((iris$Mean - mean(iris$Petal.Width))^2)
VW <- 1/nrow(iris) * sum((iris$Petal.Width - iris$Mean)^2)
```

Puis vérifier que la somme est bien égale à la variance totale :

```
VB + VW
```

Il existe en fait une fonction dédiée à la décomposition de la variance : la fonction `aov` qui vous donne immédiatement les sommes des carrés des écarts inter (expliquée par le modèle : la variable qualitative) et intra (résiduelle).

```
aov(iris$Petal.Width ~ iris$Species)
```

À partir des sommes des carrés des écarts, retrouvez les variances intra et inter :

```
80.41333/150
6.1566/150
```

3.4 Conclusions et tests

Le rapport d'explication est donnée par $\eta^2 = \frac{V_B}{V_T}$:

```
VB/VT
```

Conclusion ?

Le test d'ANOVA proprement dit, que nous verrons plus tard en cours, se fait avec la fonction `anova` sur le modèle linéaire que nous voulons tester :

```
anova(lm(iris$Petal.Width ~ iris$Species))
```

Conclusion ?