Apprentissage statistique
ENSAE, 3ème année

Arnak Dalalyan

TP1: LOGICIEL R ET PRÉDICTION DE L'EFFLORESCENCE ALGALE

A propos du logiciel R

Le système R est un logiciel distribué gratuitement depuis le site

`http://www.r-project.org`

C'est un logiciel de type Open Source. Du point de vue juridique, le logiciel est sous une licence GPL[1]. Le système R fournit un environnement intégrant un grand nombre de fonctionnalités statistiques et graphiques qui en font un outil particulièrement adapté au traitement et à l'analyse des données. Un fichier executable permettant l'installation rapide de R sous Windows peut être télechargé depuis le site

`http://cran.us.r-project.org/bin/windows/base/`

A propos du TP1 Les étudiants qui sont familiers avec le logiciel R peuvent aller directement à la Section 4.

Tous les autres étudiants sont invités à exécuter les commandes ligne par ligne en effectuant des copier-coller à partir du fichier pdf mais en veillant à bien comprendre le but de chaque instruction. Le temps estimé à passer sur les sections 1 à 3 est d'une heure environ.

Pour lancer R sur les machines installées dans les salles informatiques de l'ENSAE, il faut aller dans `Démarrer`, ensuite `Tous les programmes`, puis `R` et `RStudio`.

Lors d'un premier lancement de `RStudio`, vous serez invités à choisir l'emplacement du logiciel `R`, il faut alors indiquer

`Ordinateur` > `SourceLogicielsSeven(L:)` > `R` > `R-3.0.2` > `bin` > `x64`

Si une mise à jour est proposée, refusez là en cliquant

`Ignore Update`.

Soyez patients, le lancement de `RStudio` peut prendre un peu de temps.

Il est fortement recommandé de sauvegarder toutes les commandes utilisées (en y ajoutant des commentaires si besoin) dans un fichier qui peut être nommé `TP1.R`.

Les exemples principaux traités lors des TP sont empruntés au livre «Data Mining with R» de Luís Torgo

`http://www.dcc.fc.up.pt/~ltorgo/DataMiningWithR/book.html`

[1] General Public License - pour plus de détails sur les licences, consulter le site `http//www.gnu.org/licenses/license-list.fr.html`.

1 Les premiers pas sous R

1. Faire quelques essais:

```
pi*sqrt(10)+exp(4)
x=3:10
x
w=rep(x, 3)
w
?rep      # aide en ligne
ls()      # liste des variables
rm(x)
x
ls()
```

"You can't just punch in 'let there be light' without writing the code underlying the user interface functions."

2. Créer et manipuler des vecteurs :

```
x = c(2,3,5,7,2,1)
y = c(10,15,12)
z = c(x,y)
x
y
z
z^2
x*x
mode(x)
x = c(4, 6, NA, 2)       # la valeur NA représente une valeur manquante
x
x = c(T, F, NA, TRUE)    # T est l'acronym de TRUE, F est celui de FALSE
x
mode(x)
x[3]
x = c(7,T, F, NA, TRUE) # tous les éléments d'un vecteur doivent être
mode(x)                  # du même type
y = as.character(x)
y
length(y)
x = c(1, 1, 3, 1, 4); print(x);
x[x>2]
x[-c(1,4)]               # exclure les coordonnées 1 et 4
```

3. Passons aux matrices:

```
x = 1:12
dim(x) = c(3,4)
?dim
x
y = matrix(1:12, nrow=3, byrow=T)
```

```
t(y)
z = matrix(1:12, nrow=2, byrow=T)
z*z
z%*%z
z[,2]
```

Il est possible de donner des noms aux lignes et aux colonnes :

```
results = matrix(c(10, 30, 40, 50, 43, 56, 21, 30), 2, 4)
colnames(results) = c("1qrt", "2qrt", "3qrt", "4qrt")
rownames(results) = c("store1", "store2")
results
results["store1", ]
```

4. Les facteurs : on les utilise pour manipuler des variables catégorielles / qualitatives, prenant un nombre fini de valeurs appelées des «levels» (catégories).

```
g = c("f","m","m","f","f","m")
g
g = factor(g)
g
table(g)              # permet d'afficher le tableau de contingences
a = factor(c("adulte","adulte","enfant","enfant","adulte","adulte"))
table(a,g)
```

On peut également afficher les proportions :

```
t = table(a, g)
prop.table(t)
```

5. Les listes : une liste de R est une collection ordonnée d'autres objets de R (vecteurs, matrices, facteurs, etc.) que l'on appelle composants de la liste. Ces composants ne sont pas nécessairement du même type.

```
my.lst = list(stud.id=3445, stud.name="John",stud.marks=c(14.3,12,15))
my.lst
my.lst[1]     # permet d'accéder à la composante no 1 de la liste
my.lst[[1]]   # permet d'accéder à la valeur de la composante no 1
my.lst[[3]]
names(my.lst)
unlist(my.lst)
```

6. Tableaux de données (`dataframe`) : sorte de matrice où chaque colonne peut avoir son type (contrairement à une matrice dont tous les éléments doivent avoir le même type). Les colonnes représentent les variables et les lignes les individus.

```
mes.donnees = data.frame(site=c("A","B","A","A","B"),
  season=c("Winter","Summer","Summer","Spring","Fall"),
  pH = c(7.4,6.3,8.6,7.2,8.9))
mes.donnees
```

Les différentes façons d'accéder aux éléments d'un tableau de données :

```
mes.donnees$pH
mes.donnees[3, 2]
mes.donnees[mes.donnees$pH > 7, ]
mes.donnees[mes.donnees$season == "Summer", c("site", "pH")]
```

Différentes fonctions applicables à un dataframe :

```
names(mes.donnees)
nrow(mes.donnees)
head(mes.donnees)
ncol(mes.donnees)
edit(mes.donnees)
```

7. Graphiques :

```
x = runif(50, 0, 2)
y = runif(50, 0, 2)
par(fg="blue", bg="cornsilk", col="darkred")
plot(x, y, main="Titre", xlab="abscisse", ylab="ordonnée")
abline(h=.6,v=.6)
text(.6,.6, "placer un commentaire")
colors()
```

On peut également modifier le caractère représentant les points (option «pch») ainsi que la méthode utilisée pour relier les points (option «type») :

```
x = sort(x)
par(pch=16, col="navy", mfrow=c(2,3))
plot(x, y, main="type=points", xlab="absc", ylab="ord", type="p")
plot(x, y, main="type=overplotted", xlab="absc", ylab="ord", type="o")
plot(x, y, main="type=lines", xlab="absc", ylab="ord", type="l")
plot(x, y, main="type=both", xlab="absc", ylab="ord", type="b")
plot(x, y, main="type=only lines of both", xlab="absc", ylab="ord", type="c")
plot(x, y, main="thick lines", xlab="absc", ylab="ord", type="b", lwd=2)
```

8. Définition de fonctions:

- exemple très simple :

```
carre = function(x) x^2
carre(3)
carre
```

- exemple plus élaboré :

```
hist.norm=function(n, couleur)
{
 x = rnorm(n)
 h = hist(x, plot=F)
```

```
s = sd(x)
m = mean(x)
a = max(h$density)
b = 1/(s*sqrt(2*pi))
ylimits = c(0,1.2*max(a,b))
xtitre = "Histogramme et approximation par une loi normale"
ylab = " "
titre = paste("Echantillon gaussien : n=",n)
hist(x, freq=F, ylim=ylimits, xlab=xtitre, col=couleur, main=titre)
curve(dnorm(x,m,s), add=T, lwd=2)
}
```

Cette fonction peut maintenant être utilisée :

```
op=par(mfcol=c(1,3))
hist.norm(200 , "yellow")
hist.norm(800 , "darkgoldenrod")
hist.norm(3200, "blue")
par(op)
```

2 Outils de visualisation des données

1. Histogramme :

```
x = rnorm(1000)
hist(x, breaks=20)
hist(x, breaks=20, freq=F, col="cyan")
curve(dnorm(x), add=T,col="darkblue")
x = rnorm(50)
h = hist(x, plot=F)
h$breaks
h$counts
```

2. Boxplot (boîte à moustache) :

```
op = par(mfcol=c(2,2),bg="lightcyan")
boxplot(x)
boxplot(x,horizontal=T)
boxplot(x, col="red")
boxplot(x, col="orange",border="darkblue",lwd=2)
par(op)
```

3. Boxplots en parallèle :

```
y = (rnorm(400))^2-1
z= rnorm(50)^3
par(bg="lightcyan", lwd=1.5)
boxplot(x, y, z, col=c("blue","white","red"), border=c("black","darkblue"))
```

4. QQ-plots (diagramme quantile-quantile):

```
x = rnorm(100)
y = (rnorm(400))^2-1
z = rnorm(200,m=4,sd=5)
op = par(bg="lightcyan",mfrow=c(2,2))
qqplot(x,y,pch=21,bg="red",fg="darkblue",lwd=2)
qqplot(x,z,pch=21,bg="red",fg="darkblue",lwd=2)
qqnorm(y,pch=21,bg="orange",fg="darkblue",lwd=2)
qqline(y,pch=21,col="blue",lwd=2)
qqnorm(z,pch=21,bg="orange",fg="darkblue",lwd=2)
qqline(z,pch=21,col="blue",lwd=2)
par(op)
```

3 Lecture des données contenues dans un fichier

1. Les jeux de données sont généralement stockés dans des fichiers externes. La commande `read.table` permet de lire ce type de données. Pour tester cette commande,

   - téléchargez le fichier `AirQuality.data` de la page web (en cliquant avec le bouton droit de la souris et choisissant `enregistrer le lien sous`)
     `http://certis.enpc.fr/~dalalyan/StatNum.html`
     et placez-le dans votre répertoire de travail,
   - saisissez les commandes

   ```
   Donnees = read.table("AirQuality.data")   # lire les données
   summary(Donnees)                          # résumer les données
   hist(Donnees$Ozone, col="gold") # histogramme de la variable Ozone
   attach(Donnes)        # permet d ommettre le nom du jeu de données
   hist(Ozone, freq=F, col="gold")
   detach(Donnees)
   hist(Ozone,freq=F,col="gold")   # erreur!
   ```

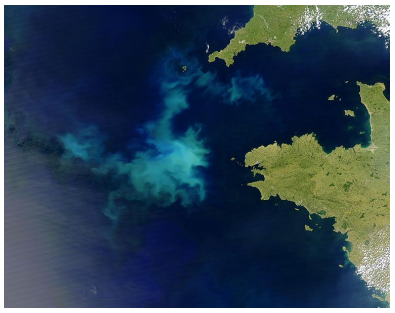
2. Un certain nombre de jeux de données sont fournis avec le logiciel R.

   ```
   data()    # afficher la liste des jeux de données
   ?WWWusage # description des données WWWusage
   plot(WWWusage)
   ```

Utiliser la commande `quit()` pour quitter `R`.

4 PRÉDICTION DE L'EFFLORESCENCE ALGALE

4.1 Présentation générale du problème (source : wikipedia.org)



Une efflorescence planctonique au large de la Bretagne, vue depuis l'espace. Surface : plus de 20 000 km^2 (cliché NASA)

Une efflorescence algale (en anglais : algal bloom ou algae bloom) est une augmentation relativement rapide de la concentration d'une (ou de quelques) espèce(s) de phytoplancton dans un système aquatique. Cette augmentation de concentration se traduit généralement par une coloration de l'eau (rouge, brun-jaune ou vert). Ce phénomène peut concerner des eaux douces ou marines. Le phénomène peut être naturel ou favorisé par des pollutions terrigènes. Dans ces derniers cas, des proliférations intenses et longues peuvent conduire à des zones mortes, en raison d'une consommation de la totalité de l'oxygène dissout dans l'eau la nuit et/ou d'émissions de toxines par certaines espèces de plancton.
Les efflorescences algales peuvent alors localement déséquilibrer la chaîne alimentaire, voire entraîner des déséquilibres écologiques plus durables (émissions de gaz à effet de serre, mortalité de poissons et crustacés), sur de vastes zones (la plus grande a atteint 22 000 km^2 en 2007, au large de l'estuaire du Mississippi).

4.2 Problem Description and Objectives

High concentrations of certain harmful algae in rivers constitute a serious ecological problem. Being able to monitor and perform an early forecast of algae blooms is essential to improving the quality of rivers.

With the goal of addressing this prediction problem, several water samples were collected in different European rivers at different times during a period of approximately 1 year. For each water sample, different chemical properties were measured as well as the frequency of occurrence of seven harmful algae. Some other characteristics of the water collection process were also stored, such as the season of the year, the river size, and the river speed.

One of the main motivations behind this application lies in the fact that chemical monitoring is cheap and easily automated, while the biological analysis of the samples to identify the algae that are present in the water involves microscopic examination, requires trained manpower, and is therefore both expensive and slow. As such, obtaining models that are able to accurately predict the algae frequencies based on chemical properties would facilitate the creation of cheap and automated systems for monitoring harmful algae blooms.

Another objective of this study is to provide a better understanding of the factors influencing the algae frequencies. Namely, we want to understand how these frequencies are related to certain chemical attributes of water samples as well as other characteristics of the samples (like season of the year, type of river, etc.).

4.3 Data Description

There are two main datasets for this problem. The first consists of data for 200 water samples. To be more precise, each observation in the available datasets is in effect an

aggregation of several water samples collected from the same river over a period of 3 months, during the same season of the year. Each observation contains information on 11 variables. Three of these variables are nominal and describe the season of the year when the water samples to be aggregated were collected, as well as the size and speed of the river in question. The eight remaining variables are values of different chemical parameters measured in the water samples forming the aggregation, namely:

- Maximum pH value
- Minimum value of O_2 (oxygen)
- Mean value of Cl (chloride)
- Mean value of NO_3^- (nitrates)
- Mean value of NH_4^+ (ammonium)
- Mean of PO_4^{3-} (orthophosphate)
- Mean of total PO_4 (phosphate)
- Mean of chlorophyll

Associated with each of these parameters are seven frequency numbers of different harmful algae found in the respective water samples. No information is given regarding the names of the algae that were identified.

The second dataset contains information on 140 extra observations. It uses the same basic structure but it does not include information concerning the seven harmful algae frequencies. These extra observations can be regarded as a kind of test set. The main goal of our study is to predict the frequencies of the seven algae for these 140 water samples. This means that we are facing a predictive data mining task. This is one among the diverse set of problems tackled in data mining. In this type of task, our main goal is to obtain a model that allows us to predict the value of a certain target variable given the values of a set of predictor variables. This model may also provide indications on which predictor variables have a larger impact on the target variable; that is, the model may provide a comprehensive description of the factors that influence the target variable.

Recommandation: Créer un nouveau fichier `R` pour sauvegarder les commandes qui suivent, car celles de la partie précédente ne sont pas utiles pour le compte rendu.

4.4 Charger, résumer et visualiser les données

1. Pour charger les données, on a besoin d'appeler la librairie DMwR :

   ```
   library(DMwR)
   head(algae)
   ```

   Ceux qui travaillent sur leur portable, doivent installer cette librairie en utilisant la commande `install.packages("DMwR")`.

2. Pour afficher les statistiques basiques des différentes variables :

   ```
   summary(algae)
   ```

3. Pour afficher l'histogramme, l'estimateur à noyau de la densité et le QQ-plot :

```
library(car)
op = par(mfrow=c(1,2))
hist(algae$mxPH, prob=T, xlab="",
  main="Histogram of maximum pH value",ylim=0:1)
lines(density(algae$mxPH,na.rm=T))
rug(jitter(algae$mxPH))
qqnorm(algae$mxPH,main="Normal QQ plot of maximum pH")
par(op)
```

Pour tracer des boxplot conditionnelles où le conditionnement est fait par rapport à une variable catégorielle (ici `size`) :

```
library(lattice)
bwplot(size ~ a1, data=algae, ylab="River Size",xlab="Algal A1")
```

4.5 Comment traiter les données manquantes ?

Certains échantillons d'eau contiennent des valeurs manquantes. Cette situation, très fréquente en pratique, peut rendre impossible l'utilisation de certaines techniques qui ne sont pas capables de gérer des données manquantes. Nous présentons ci-dessous 3 méthodes permettant de contourner cette difficulté. Mais avant d'utiliser l'une de ces méthodes, il est raisonnable de jeter un oeuil sur les observations contenant des données manquantes ou, au moins, calculer le nombre de ce type d'observations.

```
algae[!complete.cases(algae),]
nrow(algae[!complete.cases(algae),])
```

1. Supprimer les observations avec des valeurs manquantes :

```
algae1 = na.omit(algae)
nrow(algae)
nrow(algae1)
```

2. Remplir avec les valeurs les plus fréquentes :

```
algae2 = centralImputation(algae)
nrow(algae[!complete.cases(algae),])
nrow(algae2[!complete.cases(algae2),])
```

La fonction `centralImputation()`, fournie par le package `DMwR`, remplace les valeurs manquantes d'une variable quantitative par la médiane de cette variable et celles d'une variable catégorielle par sa valeur la plus fréquente.

3. Remplir les valeurs manquantes en explorant la similitude entre les observations :

```
algae3 = knnImputation(algae, k =10, meth = "median")
nrow(algae[!complete.cases(algae),])
nrow(algae3[!complete.cases(algae3),])
```

La fonction `knnImputation()`, fournie par le package `DMwR`, remplace les valeurs manquantes d'une variable quantitative par la médiane des 10 valeurs de cette variable correspondantes aux observations les plus proches de celle qui contient la valeur manquante. Pour une variable catégorielle, la médiane est remplacée par sa valeur la plus fréquente parmi les 10 observations les plus proches.

4.6 Proposer un modèle prédictif

Rappelons que le but est de prévoir l'efflorescence algale, c'est-à-dire construire un modèle de prévision pour les variables `a1,...,a7` du jeu de données `algae3`. Nous verrons ici deux méthodes simples permettant de construire ce type de modèle : la régression linéaire multiple et les arbres de décision.

1. Régression linéaire multiple :

   (a) Calcul des coefficients de la régression :

   ```
   algae = knnImputation(algae, k =10, meth = "median")
   lm.a1 <- lm(a1 ~ ., data = algae[, 1:12])
   summary(lm.a1)
   ```

   Question : Comment la commande `lm` de `R` gère-t-elle les variables catégorielles ?
   Question : Quelle est la valeur de la quantité qui sert à mesurer la qualité d'ajustement par un modèle linéaire ?

   (b) Pour choisir les variables pertinentes :

   ```
   anova(lm.a1)
   ```

   Question : Y a-t-il des variables clairement inutiles pour prévoir la variable `a1` ?

   (c) Pour déterminer un sous-modèle qui ne contient pas de variable inutile :

   ```
   final.lm = step(lm.a1)
   ```

   Question : Quelles sont les variables retenues pour le modèle final ?
   Question : En utilisant la commande `summary(final.lm)`, déterminer si oui ou non la qualité d'ajustement a augmentée par rapport au modèle initial contenant toutes les variables. Expliquer.

   Recommendation : Sauver les graphiques sous le format `pdf` ou `png`. Pour cela, si vous utilisez `RStudio`, allez dans le menu `plots` et cliquez sur `Save plot as pdf` ou `Save plot as image`.

2. Arbres de décision :

   (a) Calcul du modèle :

   ```
   algae = knnImputation(algae, k =10, meth = "median")
   library(rpart)
   rt.a1 = rpart(a1 ~ ., data = algae[, 1:12])
   rt.a1
   ```

   (b) Pour afficher l'arbre de décision obtenu :

```
par(lwd=2, col="red")
plot(rt.a1, compress=TRUE)
text(rt.a1, use.n=TRUE,col="blue")
```

ou encore

```
par(lwd=2, bg="lemonchiffon3")
prettyTree(rt.a1,col="navy",bg="lemonchiffon")
```

(c) Pour évaluer la qualité de prévision :

```
lm.predictions.a1 = predict(final.lm, algae)
rt.predictions.a1 = predict(rt.a1, algae)
regr.eval(algae[, "a1"], rt.predictions.a1, train.y = algae[,"a1"])
regr.eval(algae[, "a1"], lm.predictions.a1, train.y = algae[,"a1"])
```

(d) Pour afficher les erreurs, on peut tracer la courbe des valeurs prédites contre les valeurs observées :

```
par(mfrow = c(1, 2), col="navy", bg="lemonchiffon1")
plot(lm.predictions.a1, algae[, "a1"], main = "Linear Model",
xlab = "Predictions", ylab = "True Values", xlim=c(-15,62))
abline(0, 1, lty = 2)
plot(rt.predictions.a1, algae[, "a1"], main = "Regression Tree",
xlab = "Predictions", ylab = "True Values", xlim=c(-15,62))
abline(0, 1, lty = 2)
```

On remarque que dans les deux cas, les résultats ne sont pas terribles.

(e) Pour calculer les prévision sur les 140 observations de test qui se trouvent dans le tableau `test.algae` :

```
summary(test.algae)
test.algae =                 # completer la commande dont le but est de
                             # nettoyer les donnees manquantes
lm.predictions.a1 = predict(., .)  # completer ces 2 commandes
rt.predictions.a1 = predict(., .)  # qui ont pour but de calculer
                                       # les predictions en utilisant
                                       # le modele de regression
                                       # lineaire et celui d'arbre de
                                       # décision
```

(f) Sachant que les vraies valeurs des variables `a1,...,a7` correspondantes aux 140 nouvelles observations se trouvent dans le tableau `algae.sols`, évaluer la qualité des prédictions fournies par les deux modèles ci-dessus. On pourra utiliser la commande `regr.eval(true,pred,train.y)` vue ci-dessus.

A faire pour le compte-rendu du TP1

Utilisez votre éditeur de texte préféré (Word, LaTeX, Open Office) pour rédiger le compte-rendu. Convertissez le fichier final en **pdf** avant de l'envoyer à votre chargé de TD.

Il ne faut pas envoyer vos codes, vous pouvez les inclure dans votre compte-rendu sous forme d'annexe (ou directement dans le texte).

1. Répondez aux questions posées dans la partie 4.6.
2. Expliquez comment vous avez complété les commandes de la partie (e) ci-dessus. Quel résultat avez-vous obtenu ?
3. Insérez le code correspondant à la partie (f) ainsi que le résultat obtenu.
4. Faites ce qui est demandé dans la question (f) pour les variables `a2,...,a7` et insérez les résultats dans le compte-rendu.
5. Restez concis. Le rapport ne doit pas dépasser les 4-5 pages.
6. Il est fortement recommandé de rédiger le compte-rendu en binôme. N'oubliez pas d'indiquer vos noms et vos prénoms dans le rapport ainsi que dans le nom du fichier pdf (exemple: `ENSAE2014_TP1_Dalalyan_Hebiri.pdf`).
7. Soignez la présentation!

La date limite pour l'envoi du compte-rendu est le 31 octobre 2016.