T.P. 1 de Data Mining

Lors de nos T.P. de Data Mining, nous utiliserons le logiciel R. C'est un logiciel « libre », téléchargeable gratuitement à l'adresse `www.r-project.org`.

Le but de la première partie du T.P. 1 est d'apprendre à se familiariser avec le logiciel R et de présenter un certain nombre de commandes de base.

La seconde partie est consacrée à l'Analyse en Composantes Principales.

Première partie

Utilisation de R

1 Une calculatrice

R est un logiciel de calcul numérique orienté vers l'analyse des données en statistique. Il s'agit avant tout d'une « grosse calculatrice ».

Ainsi, R permet de faire les opérations de *calcul* élémentaire. Essayer les commandes suivantes :

```
> 3+5
> 2*4
> 3-8
> 2/3
> 2.1-6*2.5
> 3*2-5*(2-4)/6.02
> 2^2
```

R permet aussi de faire des calculs plus élaborés, à l'aide de plusieurs fonctions prédéfinies. Que font les fonctions suivantes ?

```
> sqrt(4)
> abs(-4)
> log(4)
> sin(0)
> exp(1)
```

R connaît la valeur de quelques constantes mathématiques :

```
> pi
> cos(2*pi)
```

R manipule des opérateurs et valeurs *logiques* :

`<` : inférieur
`>` : supérieur
`<=` : inférieur ou égal
`>=` : supérieur ou égal
`==` : égal
`!=` : différent
`!` : non
`&` ou `&&` : et
`|` ou `||` : ou
`xor(,)` : ou exclusif

Essayer les commandes :

```
> 1<2
> 1==3
> TRUE
> F
```

Pour définir une *chaîne de caractères*, on emploie des guillemets.

```
> 'a'
> 'essai'
```

Pour une *affectation*, on utilise les commandes `<-` ou `=`.

Remarque 1.

- Si une ligne contient plusieurs instructions, elles doivent être séparées par « ; ».
- R distingue minuscules et majuscules.
- Les lignes de commande commençant par « # » ne sont pas exécutées par R.
- `ls()` affiche la liste des objets créés et installés dans l'environnement courant.

```
> # Un joli calcul
> a=52
> b <- 50
> A=0
> A-b ; a-b
```

2 L'aide

Lorsque vous recherchez une fonction, que vous en avez oublié le nom ou la syntaxe, l'aide du logiciel vous apportera sûrement toutes les informations souhaitées. S'il ne vous est pas demandé de retenir par coeur toutes les commandes possibles et imaginables de R, il est en revanche *très important de savoir utiliser l'aide* pour les retrouver. Celle-ci doit vous permettre d'acquérir une réelle autonomie. On peut y accéder de différentes manières.

- `help.start()` lance l'aide en ligne.
- L'instruction `?obj` (ou `help(obj)`) affiche la documentation associée à l'objet `obj`.
- `apropos('mot')` affiche la liste des fonctions et objets qui contiennent le mot-clé `mot` dans leur nom, tandis que `??mot` affiche cette liste dans l'aide en ligne.

Essayer les commandes :

```
> ?plot
> help(plot)
> help.start()
> apropos('plot')
> ??plot
```

3 Vecteurs, matrices, tableaux

L'objet de base en R est le *vecteur*. Un vecteur peut contenir des valeurs numériques, mais aussi des valeurs logiques (`TRUE` ou `FALSE`, `T` ou `F`) ou encore des chaînes de caractères...

La commande permettant de créer un vecteur en énumérant ses valeurs est `c`, ce qui signifie *concaténation*. Plusieurs vecteurs peuvent d'ailleurs être concaténés à l'aide de cette même commande.

```
> v1=c(1,2,3,4,5)
> v1
> v2= c(3.14, 2.71, 1.414, 1.732)
> v2
```

```
> v3= c(2,4, 3.14)
> v3
> v4=c(v1,v2,v3)
> v4
> length(v1); length(v2); length(v3); length(v4)
> v5=c(T, T, F, F)
> v5
> v6=c('a', 'b')
> v6
```

On peut aussi fabriquer une suite à l'aide de `seq` en donnant les valeurs minimale et maximale et le pas ou la longueur. Si le pas est 1, il suffit même d'écrire `min : max`. La fonction `rep` donne un vecteur dans lequel un nombre ou une suite de nombres est répété autant de fois que désiré. Enfin, la fonction `scan('')`, qui permet de saisir directement le contenu d'un vecteur, peut s'avérer très pratique.

```
> seq(1,10,2)
> seq(1,10,by=2)
> seq(1,9,length.out=5)
> seq(1,10,by=.1)
> 1:10
> rep(55,10)
> rep(c(1,2), 10)
> notes=scan('')
1: 5 12 14 3 6 10 12 6 5 11 10 9 2 6.5 9 17 8 9.5 10
> notes
```

La fonction `vector(mode=,length=)` crée un vecteur avec des entrées de type et longueur donnés.

```
> vector('numeric',6)
> vector('logical',5)
> vector('character',3)
```

La fonction `factor` donne à un vecteur le statut de variable qualitative ou facteur. On peut aussi construire un facteur en spécifiant le nombre de modalités et le nombre d'occurrences de la même modalité à l'aide de `gl`.

```
> factor(c('brun','blond','roux','blond','roux','brun','brun'))
> factor(c(1,2,3,2,3,1,1),labels=c('brun','blond','roux'))
```

On accède aux éléments d'un vecteur en mettant entre crochets l'indice ou un vecteur d'indices :

```
> a=rnorm(50)
> a[1]
> a[2]
> a[c(1,3,5)]
```

On peut donner des noms aux éléments d'un vecteur :

```
> v=c(1,2,3,4)
> names(v)=c('alpha','beta','gamma','delta')
> v['beta']
> a='alpha'; b='beta'; c='gamma'; d='delta'
> v[b]
```

La fonction `matrix`, prenant en argument un vecteur, le nombre de colonnes et le nombre de lignes, permet d'obtenir une *matrice* (tableau à deux dimensions). En mettant les indices entre crochets, on peut extraire un élément, une colonne, une ligne ou une sous-matrice. Il est possible de donner des noms aux lignes et colonnes de la matrice avec `colnames` et `rownames`. Les fonctions `cbind` et `rbind` permettent de réunir deux matrices.

```
> mat=matrix(1:20, 4, 5)
> mat1=matrix(1:20, 4, 5, byrow=TRUE)
> id=diag(5)
> mat[1,5]
> mat[1,]
> mat[,5]
> mat[1:2,1:3]
> colnames(mat)=c('C1', 'C2', 'C3', 'C4', 'C5')
> rownames(mat)=c('L1', 'L2', 'L3', 'L4')
> cbind(mat,mat1)
> rbind(mat,mat1)
```

La longueur d'un vecteur est donnée par `length` et la dimension d'une matrice par `dim`. La fonction `which` retourne le vecteur des indices pour lesquels la condition logique donnée en argument est vraie. L'option `arr.ind=T` permet de traiter le cas des matrices.

```
> length(v)
> dim(mat)
> which(notes==10)
```

D'autres objets de R :
- `list` : liste, ensemble de vecteurs.
- `array` : vecteur avec dimensions ; il peut y en avoir une (vecteur), deux (matrice) ou plus.
- `data.frame` : tableau de données, matrice dont chaque colonne, qui possède un nom, correspond à une variable, les lignes correspondant aux individus.

```
> a=array(1:20,dim=c(4,5))
> a[2,4]
>
> b=list(alpha=1:3, beta=c('a','b','c','d'))
> names(b)
> b$alpha
> b$beta
> b[1]
> b[2]
>
> c=data.frame(a=gl(2,5,10), b=1:10, x=seq(1,20,2))
> c
> c$a ; c[,1]
> c$b ; c[,2]
> c$x ; c[,3]
> names(c)
> rownames(c)
```

On peut faire afficher le type d'un objet et convertir un objet en un objet de type différent.

```
> class(mat)
> is.data.frame(mat)
> as.vector(mat)
```

Pour lire un fichier de données, on utilise la fonction `read.table`. L'option `header` permet de d'indiquer si la première ligne du fichier correspond aux noms des colonnes, tandis que `sep` sert à spécifier le type de séparateur.

Exercise 1

1. Découvrez comment utiliser les fonctions suivantes : `runif`, `rnorm` et `sample`.

2. Créez une matrice M de taille 10×3 telle que les entrées de la première colonne soient $\mathcal{N}(0, 10)$ indépendante, les entrées de la deuxième colonne soient $\mathcal{U}[-1, 1]$ indépendantes et les entrées de la dernière colonne soient choisies uniformément de {0,1,...,100}.
3. Créez une matrice N obtenue en mettant tous les éléments non positifs de M à 0, puis calculez une liste L de `min`, `max` et `mean` des lignes et des colonnes de N (L est une liste de 6 vecteurs).

4 Quelques paramètres graphiques

La fonction principale permettant de tracer un *graphe* est `plot`. Elle possède de nombreuses options concernant le type de tracé, la couleur, les axes, le titre du graphique, ou encore la légende.

`type` : points, ligne, les deux, bâtons, escalier...
`lwd` : épaisseur du trait
`lty` : type de trait
`cex` : taille des points
`pch` : forme des points
`col` : couleur
`main` : titre du graphique
`xlab, ylab` : titres des axes
`asp` : échelle y/x
`xlim, ylim` : limites des axes

Les fonctions `lines` et `points` sont utilisées pour ajouter des lignes ou des points à un graphe existant. On peut ajouter une légende à l'aide de `legend`, fonction possédant elle aussi plusieurs options. Plus généralement, du texte peut être ajouté sur le graphique grâce à `text`. Lorsqu'il s'agit d'une expression mathématique, il est préférable utiliser la fonction `expression` plutôt qu'une chaîne de caractère. `x11()` ouvre une nouvelle fenêtre graphique, tandis que `dev.cur()` donne le numéro de la fenêtre courante et `dev.list()` la liste des fenêtres ouvertes. `dev.off()` ferme la fenêtre courante ou celle dont le numéro est donné en argument et `graphics.off()` les ferme toutes. On peut diviser la fenêtre graphique grâce à `par(mfrow=)`.

```
> x=1:10
> y=(1:10)^2
> a=cbind(x,y)
> par(mfrow=c(2,2))
> plot(a,cex=1.5)
> plot(a,type='l')
> plot(a,type='b',lwd=2,pch=2)
> plot(a,type='l',lty='dotted',col=2,main='Titre du graphe',xlab='Abscisses',
  ylab='Ordonnées')
> lines(x,x,col=3)
> points(x,y+2*rnorm(10),col=4,pch=16)
> legend('topleft',expression(y==x^2,y==x,'Données'),inset=0.05,
  lty=c('dotted','solid',NA),pch=c(NA,NA,16),col=c(2,3,4 ))
```

Les méthodes graphiques les plus modernes dans R sont probablement `ggplot` et `plotly`. Pour plus d'informations :

— `ggplot` : `https://ggplot2-book.org/index.html`
— plotly : `https://plotly.com/r/`

Exercise 2

— Simulez 1000 points de `Drifted Simple Random Walk` i.e., $Y_n = T(n) + \sum_{i=1}^{n} X_i$, où $X_i \in \{-1, 1\}$ indépendante avec $\mathbb{P}(X_i = -1) = \mathbb{P}(X_i = 1) = 1/2$, et dans ce cas $T(n)$ est choisi pour être $\sin(\pi n/50)$. Ajoutez un titre et une couleur.
— Répétez la question précédente 50 fois et tracez toutes les courbes dans un graphique.
— Ajoutez la série temporelle moyenne des 50 courbes ci-dessus. Commenter.

5 Fonctions

Remarque 2. Pour afficher le code d'une fonction de R, il suffit d'en entrer le nom dans la console.

On peut définir ses propres fonctions, avec la syntaxe suivante :

`mafonction <- function(x) {mes instructions}`.

En écrivant une fonction, on peut être amené à utiliser une boucle. La syntaxe d'une boucle « for » (pour i variant de 1 à n, faire telle action) est par exemple

`for (i in indices) {action}`.

Il peut être utile aussi de manipuler une condition « if », ce qui se fait avec la syntaxe

`if (condition) {action 1} else {action 2}`.

Lorsque l'on écrit une fonction, il est préférable de la rédiger dans un fichier séparé plutôt que dans la console R. Un fichier contenant un ensemble de fonctions et commandes R constitue un programme, qui peut être enregistré au format « .R » de `Rscript`, ou « .Rmd » de `Rmarkdown`.

Exercise 3

1. Nous voulons vérifier le TCL avec certaines distributions bien connues.
   - Ecrire une fonction `simulMat(m,n)` qui simule une matrice G de taille $m \times n$ avec des éléments $G_{ij} \sim \mathcal{P}(\lambda)$ indépendante, pour votre libre choix de $\lambda > 0$.
   - Calculer un vecteur z de taille m où $z_j = \sqrt{\frac{n}{\lambda}}(\sum_{j=1}^{n} G_{ij}/n - \lambda)$ pour $j = 1, ..., m$. Quelle est la distribution attendue de z_j ?
   - Pour $m = 500$ et $n = 1000$, calculez z et tracez la distribution du vecteur z. Proposer un test de normalité de la distribution de z. Conclure.
   - Faites la même chose, mais utilisez des distributions exponentielles et géométriques.
2. Lemme de Johnson-Lindenstrauss. Ce lemme est une technique de réduction dimensionnelle qui vise à projeter les points de données d'un espace de grande dimension $\mathbb{R}^m$ dans un sous-espace de dimension plus petit d ($d << m$), en préservant leurs normes euclidiennes avec grande probabilité.
   - Simulez deux vecteurs u et v de taille $m = 1000$ à partir d'une distribution de votre choix. On peut considérer ces deux vecteurs comme deux points de données de dimension m.
   - Ecrire une fonction `randMat(m,d)` qui simule une matrice R de taille $m \times d$ avec des éléments $R_{ij} \sim \mathcal{N}(0, 1/d)$ indépendante. Calculer le rapport :

$$r = \frac{\|u^t R - v^t R\|^2}{\|u - v\|^2}$$

     .
   - Pour u et v fixes, simulez 500 valeurs du rapport r avec différentes valeurs de la matrice R. Tracez la densité de ce vecteur $(r_j)_{j=1}^{500}$ de dimension 500.
   - On s'intéresse à la distribution du vecteur $(r_j)_{j=1}^{500}$ précédent pour différentes valeurs de d. Avec $d = 1, 2, 3, 5, 10, 20, 50, 100, 500, 750$, calculez une matrice $K = (r_{ji})$ de taille 500×10, où chaque colonne i contient $(r_j)_{j=1}^{500}$ pour chaque valeur de $d = d_i$.
   - Tracez la distribution de chaque colonne de cette matrice. Testez la normalité de chaque colonne de K. Commenter.
   - Répétez la même chose mais avec une matrice aléatoire d'entrées uniformes $\mathcal{U}[-1, 1]$. Commenter.

Deuxième partie

Exemples d'ACP

Utiliser la fonction `prcomp` de R ou la bibliothèque ade4 et en particulier la fonction dudi.pca() pour réaliser une analyse en composantes principales normée dans les deux cas suivants. Interpréter.

1 Exemple élémentaire

Six étudiants ont obtenu les notes suivantes dans trois matières.

Etudiant	M1	M2	M3
Victorine	9	12	10
Eugène	15	9	10
Delphine	5	10	8
Jacques	11	13	14
Anastasie	11	13	8
Honoré	3	15	10

2 Données BigMac2003

Les données `BigMac2003`, disponibles dans la bibliothèque alr3, fournissent des informations sur le niveau de vie dans 69 villes du monde. Les variables sont :

— `BigMac` : nombre de minutes de travail nécessaires pour pouvoir s'acheter un Big Mac.
— `Bread` : nombre de minutes de travail pour acheter 1kg de pain.
— `Rice` : nombre de minutes de travail pour acheter 1kg de riz.
— `FoodIndex` : indice des prix des aliments (base 100 : Zürich).
— `Bus` : prix en dollars d'un aller simple pour un voyage jusqu'à 10 km.
— `Apt` : loyer d'un appartement de 3 chambres.
— `TeachGI` : revenu brut annuel d'un enseignant d'école primaire, en milliers de dollars.
— `TeachNI` : revenu net annuel d'un enseignant d'école primaire, en milliers de dollars.
— `TaxRate` : taux d'imposition d'un enseignant d'école primaire.
— `TeachHours` : nombre d'heures de travail par semaine d'un enseignant d'école primaire.