TP Introduction à R - Bases du langage
CED - MaiMoSiNE

Février 2017

1 Session de travail

1.1 Utilisation de Rstudio

Il s'agit de se familiariser avec l'interface *RStudio* et les facilités qu'elle apporte pour l'utilisateur et le développeur.

1. Configurer le panneau d'affichage pour faire apparaître sur les différentes fenêtres les informations pertinentes.

   - Aller dans le menu **Preferences** de **RStudio** puis **Panel Layout**
   - Définissez les affichages associés aux quatre fenêtres de RStudio.

2. Les outils très utiles sont:

   - l'éditeur: écrire un script, sauvegarder, sources, ...
   - workspace: affiche les objets utilisés en cours de session,
   - plot: gestion des sorties graphiques,
   - history: historique des commandes,
   - file: gestion des fichiers et des répertoires,
   - package: gestion des packages utiles en cours de session.
   - help: affichage des help, accès aux tutoriaux,...

1.2 Aide sous R

- Utiliser le menu help de RStudio
- Accès aux tutoriaux par commande: help.start()
- Aide sur une fonction: help(plot) ou ?plot
- RSiteSearch("calcul de moyenne"): recherche sur internet dans la liste des archives des utilisateurs.

1.3 Sauvegarde de l'espace de travail

Quand vous quittez R (RStudio), il vous est posé la question:

Save workspace image ?"

Répondez *OUI*? L'ensemble des objects contenu dans l'espace de travail sera sauvegardé dans le fichier ".Rdata" par défaut.

Lorsque vous redémarrez R, vous avez le message:

[Previously saved workspace restored]

Pour nettoyez l'espace de travail: **rm(list=ls())**

1.4 Utilisation d'un package

On utilisera un certain nombre de package pendant les TPs, il sera nécessaire de les charger dans votre session.

- Charger le package *sensitivity*
- Consulter le contenu de ce package (liste des fonctions,. . .)
- Demander l'aide sur une des fonctions du package

```
# Utilisation de package
# Menu "Package" RStudio
library(sensitivity)

# Contenu du package
```

```
library(help=sensitivity)

# Aide sur une fonction du package charge
help(sobol)
```

1.5 Exécution d'un script

Charger dans l'éditeur la fontion prodVect.R stockée dans le répertoire du TP, observez les paramètres en entrée et en sortie.

Une fonction est un objet R, pour l'utiliser elle devra être présente dans votre "workspace", utilisez la commande *source* ou le menu "Source on save" de Rstudio . Écrire un script qui fera appel à la fonction *vectProd* et exécuter le.

```
# Appel de la fonction
#
v1=c(1:10)
w1 <- v1*v1

source("prodVect.R")
xx <- prodVect(v1,w1)
```

2 Manipulation des objects R

2.1 R comme machine à calculer

On peut :

- Faire toutes les opérations que l'on veut grâce aux opérateurs +,-,*,/,^
- Principales fonctions mathématiques.
  - log()/log10() Logarithme népérien/décimal
  - exp() Exponentielle.
  - cos()/sin()/tan() Cosinus/Sinus/Tangente.
  - abs() Valeur absolue.
  - sqrt() Racine carrée.

On peut manipuler des variables (scalaires, vesteurs, matrices,..)

```
a <- 2
b <- a
a*b+4
log(a)
```

2.2 Vecteur, matrice, tableau

2.2.1 Créer un vecteur

Créer les vecteurs y1, y2, y3 et y4, avec $d = 4$ et $e = 12$:

- y1 : une suite d'indices de 1 à 12.
- y2 : trois fois l'élément d, puis trois fois d au carré, puis trois fois la racine de d.
- y3 : la séquence de 1 à 20 avec un pas de deux.
- y4 : 10 chiffres compris entre 1 et 30 avec un intervalle constant.

```
#
d<-4
e=12
y1 <- 1:12
y2 <-rep(c(d,d^2,sqrt(d)),c(3,3,3))
y3 <- seq(1,20,2)
y4 <- seq(1,30,length=10)
```

2.2.2 Créer des matrices

Construisez les matrices Z_1, Z_2, Z_3 à partir de y_1

$$Z_1 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} \quad ; \quad Z_2 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} \quad ; \quad Z_3 = \begin{bmatrix} 1 & 2 & 3 & 4 & 1 & 2 & 3 \\ 5 & 6 & 7 & 8 & 5 & 6 & 7 \\ 9 & 10 & 11 & 12 & 9 & 10 & 11 \end{bmatrix}$$

Puis afficher:

- L'élément de Z_1 contenu dans la première ligne et la troisième colonne.

- La première ligne de Z_1.

Soit:

$$W = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} \quad \text{et le vecteur} \quad V = \begin{bmatrix} 1 \\ 5 \\ 9 \end{bmatrix}$$

- Effectuer le produit matriciel $Z_1 * W$
- Exécuter les commande $Z_1 * W$, $Z_1 + W$ et observez le résultat.
- Exécutez les commandes $Z_1 * V$ et $Z_1 \% * \% V$ et observez les résultats.

2.3 Les listes et les fonctions

Écrire une fonction qui prend deux vecteurs de même taille en entrées et fournit la somme et le produit terme à terme de ces deux vecteurs. On utilesera un objet de type *list* pour les sorties.

Testez sur deux vecteurs et observez les résulats.

```
# Fonction et liste
prodsom <- function(V1,V2)
{Vprod <- V1*V2
 Vsomme <- V1 + V2
 res <- list(prod=Vprod,somme=Vsomme)
 return(res)
}
```

3 Manipuler les données

3.1 Fichiers de données (data.frame)

Lire le fichier texte "poidsTaille.txt" stocké dans le répertoire du TP R, ce fichier est un fichier texte contenant les données d'un étude du CHU de Toulouse, concernant le poids, la taille, le sexe d'enfants de 4 à 7ans. Ce fichier contient sur sa première ligne le nom des variables observées.

- Quel est la classe de la variable ainsi crée? Quelle est la taille de l'échantillon? Quel le mode et la classe des variables contenu dans cet échantillon.

- Calculer les statistiques de base à l'aide de la fonction *summary*
- Calculer séparément la moyenne, la médiane, l'étendue et les quantiles pour les variables *Poids* et *Taille* à l'aide de commandes appropriées.
- Calculer la variance (fonction *var*) des variables *Poids* et *Taille*, la variance calculée et la variance avec biais. Écrire une fonction qui calcule la variance sans biais et l'écart-type.
- Tracer un histogramme en n classes de ces deux variables, on fera varier n.
- Extraire :
  - la deuxième ligne
  - la troisième colonne
  - les lignes 1, 2 et 4 avec une seule commande c()
  - les lignes 3 à 6 avec la commande :
  - tout sauf les colonnes 1 et 2.
  - toutes les lignes ayant une AGE supérieure à 70mois

3.2 Reperer les individus manquants

Les données manquantes sont représentées sous R par NA (Not Available). Pour les retrouver, on utilise la fondtion **is.na** qui renvoie **TRUE** si la valeur vaut NA et **False** sinon.

Construire une variable mesurée sur 15 individus en effectuant un tirage selon une loi normale $\mathcal{N}(0,1)$, on affecte des données manquantes pour les individus 7,8,14, il s'agit de:

- repérer les individus qui ont des données manquantes,
- éliminer les individus qui ont des données manquantes.

```
# Echantillon
set.seed(23)
var <- rnorm(15,mean=0,sd=1)
var[c(7,8,14)] <- NA
#
# Identifier les donnees manquantes
```

```
select <- is.na(var)
which(select)
# Eliminer les individus ayant une donnee manquante
var2 <- var[!select]
```

4 Distribution et représentations graphiques

$X_1, X_2, \ldots X_p$: p variables indépendantes qui suivent une loi normale. $X_1^2 + X_2^2 \ldots X_p^2$ suit une loi du χ^2 à p degrés de liberté.

- Écrire une fonction *normchi* qui calcul la somme des carrés d'une variable normale. Utiliser p= 10
- Générer à l'aide de la fonction *normchi*, une table de 1000 répétitions comme suit :

  tapply(1 :1000,as.factor(1 :1000),normchi)
- Tracer un histogramme de 20 classes et superposer dessus la loi du χ^2 théorique.
- Tracer, sur le même graphique, des densités pour différentes degrés de liberté : 3, 5, 10, 15 et 20.

```
Distribution
# -------------

normchi<-function(x) {sum((rnorm(10))^2)}
tt<-tapply(1:1000,as.factor(1:1000),normchi)
#
hist(tt,proba=T,nclass=20,main='Somme de va normales = chi-deux',xlab='somme')
x0<-seq(0,30,length=100)
lines(x0,dchisq(x0, df=10),col=4)

#
y1<-dchisq(x0,3)
y2<-dchisq(x0,5)
y3<-dchisq(x0,10)
y4<-dchisq(x0,15)
y5<-dchisq(x0,20)
```

```
#
X11()
plot(x0,y1,type="n",xlab="x",ylab="Chi-2 densite")
lines(x0,y1,lty=1)
lines(x0,y2,lty=2)
lines(x0,y3,lty=3)
lines(x0,y4,lty=4)
lines(x0,y5,lty=5)
lego<-c("Chi-deux(3)","Chi-deux(5)","Chi-deux(10)","Chi-deux(15)","Chi-deux(20)")
legend(18,0.23,lego,lty=1:5)
```

5 Quelques méthodes statistiques

On reprend le fichier de données "poidsTaille.txt".

5.1 Ajustement de loi

On s'intéresse à la variable **POIDS**

1. Visualiser l'histogramme, le boxplot et la fonction de répartition empirique de X1 (avec *hist*, *boxplot* et *plot(ecdf())*)

2. Tester plusieurs types de lois (loi lognormale lnorm et loi normale norm par exemple)

   - Estimer les paramètres de ces lois par maximum vraisemblance (avec *fitdistr* de la librairie **MASS**)
     On rappelle que l'on peut utiliser la fonction *names* pour savoir quelles quantités sont attachées à un objet et pour extraire ces quantités, on peut utiliser $.
   - Regarder graphiquement l'ajustement en superposant les densités de probabilités testées sur l'histogramme (ou les fonctions de répartition sur la fonction de répartition empirique)

```
# ----------Ajustement de la loi

x11()
hist(POIDS,pr=T,col="lightblue",border="white",main="Histogramme de POIDS",c
lines(density(X1),col="blue",lwd=2)

# ---------------------- Estimation
```

```
# Loi normale
fit_N <- fitdistr(POIDS,"normal")
names(fit_N)
param_N <- fit_N$estimate

# ---------------------- Ajustement graphique
x11()
hist(POIDS,pr=T,col="lightblue",border="white",ylim=c(0,0.45),main="Histogramme de PO
lines(seq(0,10,le=100),dlnorm(seq(0,10,le=100),param_LN[1],param_LN[2]),lwd=2,col="bl
lines(seq(0,10,le=100),dnorm(seq(0,10,le=100),param_N[1],param_N[2]),lwd=2,col="red")
legend("topright",col=c("blue","red"),lty=c(1,1),lwd=c(2,2),c("loi lognormale","loi n
```

5.2 Régression linéaire simple

On se propose ici d'estimer par la méthode des moindres carrés les coefficients de régression linéaire entre les variables *POIDS* et *TAILLE* .

$$POIDS = aTAILLE + b + \epsilon$$

.

(a) Visuliser le graphe (TAILLE,POIDS)

(b) Procéder à la régression linéaire (avec lm)

(c) Utiliser les commandes *summary* et *plot* sur les résultats de la regression linéaires.

```
# ------------------ Regression lineaire
# Lecture des donnees
don <-read.table("poidsTaille.txt",header=TRUE)[,2:5]

POIDS <- as.numeric(don$POIDS)
TAILLE <- as.numeric(don$TAILLE)

formule1 <- POIDS ~ TAILLE
fit1 <- lm(formule1)
summary(fit1)
x11()
par(mfrow = c(2, 2))
```

```
plot(fit1)

x11()
plot(TAILLE,POIDS)
points(TAILLE,predict(fit1),col="blue")
abline(0,1)
```