Outils informatiques pour la statistique

C. Biernacki et R. S. Stoica

Université Lille 1
Laboratoire Paul Painlevé
59655 Villeneuve d'Ascq Cedex, France

Septembre, 2011

Avant-propos

Le cours “Outils informatiques pour la statistique” est destiné aux élèves du Master 2 Recherche Mathématiques Appliquées de l'Université Lille 1. L'objectif de ce cours est l'initiation et la familiarisation des étudiants avec le logiciel R.

J'ai effectué cet enseignement pendant quatre ans. En fonction des variations dans la maquette d'enseignement le volume horaire présentiel consacré à ce cours a été soit de 5 séances de 3 heures, soit de 8 séances de 3 heures.

Pour préparer cet enseignement j'ai travaillé avec C. Biernacki qui m'a aidé à définir la structure fondamentale de cet enseignement et qui m'a généreusement mis à disposition ses ressources pédagogiques [1]. Une autre source pédagogique très utile a été constituée par les notes de cours d'Anne Philippe [2]. Je dois mentionner et remercier également tous les collègues de France et d'ailleurs qui jouent le jeu et qui déposent sur leur pages web de nombreuses ressources d'une très grande qualité.

Le cours contient cinq grandes parties :

- préliminaires : commandes élémentaires, bibliothèques et jeux de données, manipulations élémentaires de variables
- structures de données : vecteurs (numériques, logiques, chaînes de caractères), listes, tableaux multidimensionnels, structures de données hétérogènes, matrices, facteurs
- éléments de langage - programmation : lecture de données depuis un fichier, commandes de contrôle, fonctions
- introduction aux graphiques : gestion de la fenêtre graphique, fonctions haut niveau, fonctions bas niveau
- fonctionnalités statistiques : lois de probabilité, statistiques descriptives, statistiques inférentielles, estimation et tests paramétriques classiques, estimation et tests non paramétriques classiques, régression linéaire.

J'ai choisi de faire cet enseignement en confrontant dès le début les étudiants à des problèmes et à des exercices de programmation concrets. Le nombre et le niveau des étudiants a permis de les laisser travailler à leur rythme et

en même temps de soutenir d'une manière plus appuyée ceux qui en avaient besoin. La coopération entre les étudiants a été stimulée. Pour la note finale, les étudiants ont du rendre le compte-rendu d'un projet. Afin de favoriser le travail en équipe les projets ont été traités en binôme (exceptionnellement en trinôme).

Les étudiants ont globalement tous eu des bonnes notes lors de l'évaluation. Malgré cela, j'ai pu remarquer une certaine faiblesse de leur part au niveau informatique. Les étudiants ne maîtrisent plus des notions fondamentales comme les boucles ou les tests. A mon avis, ce manque ne peut pas être compensée par une grande adaptabilité aux diverses platformes logiciels qui se présentent à eux.

En ce qui concerne les notions mathématiques, il y a eu un grand plaisir interagir avec des étudiants de niveau M2. Cependant, on a pu remarquer une certaine hétérogeneité dans leur connaissances. Ceci est du à mon avis au recrutement des étudiants provenant de divers horizons, mais aussi de certains manques difficilement évitables dans la construction des maquettes pédagogiques.

Ce document contient la plupart des exercices que les étudiants ont du travailler pendant le cours, ainsi que deux sujets de projet. Des solutions aux exercices proposés sont également fournies.

Tous les commentaires, remarques et critiques sont plus que les bienvenus. Ainsi, il est possible de contacter les auteurs directement par émail[1].

Au nom des auteurs :

Radu Stoica

[1] `christophe.biernacki@math.univ-lille1.fr` et `radu.stoica@math.univ-lille1.fr`

1 Exercices

Les premiers pas ...

Exercice 1.
a) Créer un répertoire **TPenR**. Lancer **R**.
b) Définir deux variables x et y.
c) Créer un fichier de commandes **echange** qui permutera automatiquement le contenu de x et y. Les nouvelles valeurs de x et y seront d'une part affichées à l'écran et d'autre part disponibles dans le fichier nommé **xETy**. Par aileurs, un message à l'écran devra informer l'utilisateur que le fichier de commande a été totalement exécuté.

Exercice 2. Tester les commandes suivantes :
a) history()
b) .Last.value
c) load()
d) Quit()
Indication : utiliser l'aide pour comprendre.

Exercice 3. Quitter **R** en sauvegardant l'environnement de travail dans le fichier **mon1erTP**.

Structures des données : vecteurs, matrices, listes, etc.

Exercice 4. Tester les commandes suivantes :

```
> x = c(1,5,6,11);y=c(2,3)
> z = 2*x + y + 1
> x > 3 + 10
> (x > 3)+10
> x > (3 + 10)
```

Question : qu'est-ce que vous observez ?

Exercice 5. Créer en un minimum de commandes :
a) La table de multiplication.
b) La table d'addition.
c) Une matrice 20×20 avec les nombres de 1 à 799 de deux en deux et répartis de façon croissante colonne par colonne. Les noms de lignes seront

lig. 1, lig.2, etc. et les noms des colonnes seront **col.1, col.2**, etc.
d) La matrice précedente où les multiples de 3 sont remplacés par la valeur 0. Combien y avait-il de multiples de 3 ?
e) Un vecteur contenant les valeurs non nulles de la matrice précédente.
f) Une liste à deux composantes : la première, nommée **petit**, contient les valuers inférieures strictement à la moyenne des valeurs du vecteur, la deuxième, nommée **grand**, contient les autres valeurs.
g) La liste précédente avec la composante **petit** ayant ses éléments classés par ordre décroissant.
h) Une matrice 50×50 avec des 1 partout sauf des 0 sur la diagonale (une ligne de commande suffit).

Exercice 6. On veut résoudre le système linéaire $Ax = b$ avec $A = \begin{pmatrix} 1 & 1 \\ 1 & 10^{12} \end{pmatrix}$ et $b = (1, 2)'$.
a) Créer la matrice A.
b) Essayer deux commandes différentes pour résoudre le système.

Exercice 7.
a) Charger le jeux de données **iris**.
b) À quelle structure de données appartient-t-il ?
c) À quelle structure de données appartiennent chacune de ses colonnes ? De ses lignes ?
d) Proposer une méthode qui évitera dans l'avenir de retaper à chaque fois le mot**iris** dès qu'on veut accéder à la structure de données.
e) Donner, en une seule commande, le nom de toutes les espèces de fleur présentes dans le jeux de données.

Un peu de programmation ...

Exercice 8. Programmer la fonction factorielle sous forme d'une fonction **R** récursive (c'est à dire qui s'appele elle-même).

Exercice 9.
a) Écrire une fonction qui échange le contenu des valeurs a et b.
b) Écrire une fonction qui calcul le plus grand diviseur commun de deux entiers naturels.
c) Écrire une fonction qui donne la forme irréductible d'une fraction.
d) Créer l'opérateur binaire %sur% qui effectue la mêeme opération.

Exercice 10. On s'intéresse à la série $S_n = 1 + 2^2 + 3^2 + \ldots + n^2$.
a) Calculer S_n de deux façons différentes : avec boucle puis sans aucune boucle.
b) Pour $n = 10^5$ par exemple, mesurer avec une commmande **R** la différence de temps de calcul entre les deux méthodes. Conclusions.

Introduction aux graphiques.

Exercice 11. Tester les commandes suivantes :

```
> x = rnorm(20) ; y =rexp(20)
> plot(x,y)
> points(x+.1,y+.1,pch=2)
> lines(sort(x),y,lty=2)
> text(1,2,"text")
> abline(h=0.75)
> title("texte")
```

Commentaire : bon, la première commande devance un peu le sujet du tp ...

Exercice 12.
a) Afficher une projection du jeu de données **iris** sur toutes les paires d'axes dans une même fenêtre graphique.
b) Faites plusieurs constatations visuelles.
c) Écrire une fonction projetant sur une paire d'axes les individus en affectant un symbole et une couleur différent selon leur espèce d'appartenance. Cette fonction ajoutera aussi le nom des axes et une légende. Par défaut, cette fonction devra sélectionner les deux premiers axes.

Exercice 13. Programmer deux fonctions qui affichent les fonctions sinus et cosinus sur $[0, 2\pi]$:
a) - superposées dans le même graphe (fonction **sin.cos**)
b) - dans deux graphes différents se partageant la même fenêtre graphique (fonction **sin.cos.2**).

Voila les stats ...

Exercice 14. Statistiques descriptives. On désire faire des statistiques descriptives sur les données `iris`.
a) En une seule commande, obtenir tous les indicateurs numériques de base

associés à chacune des variables quantitatives et qualitatives.
b) Tracer sur un même graphique chacune des "boîtes à moustaches" associée à chaque variable quantitative (une commande est encore suffisante).
c) Tracer un camembert puis un *bar plot* de la variable qualitative.
d) Tracer sur un premier graphique la droite de Henry pour la variable `Sepal.Length`. Sur un autre graphique, faire de même avec uniquement les individus de l'espèce `versicolor`. Conclusion ?

Exercice 15. Loi des grands nombres. Il s'agît d'illustrer la loi forte des grands nombres : si $X_1, X_2, \ldots, X_n$ est une suite de variables aléatoires i.i.d. avec $\mathbb{E}[X_1] < \infty$, alors

$$\frac{1}{n}\sum_{i=1}^{n} X_i \xrightarrow{p.s.} \mathbb{E}[X_1].$$

a) Générer une réalisation $x_1, \ldots, x_n$ de la suite $X_1, \ldots, X_n$ avec $n = 50000$ dans le cas d'une loi normale $X_1 \sim \mathcal{N}(0, 1)$.
b) Calculer $\bar{x}_m = \frac{1}{m}\sum_{i=1}^{m} x_i$ pour $m = 1, \ldots, n$.
c) Tracer $\bar{x_m}$ en fonction de m. Conclusion ?
d) Procéder de même avec la loi de Cauchy. Conclusion ?

Exercice 16. Théorème central limite. Il s'agît d'illustrer maintenant le théorème central limite : si $X_1, X_2, \ldots, X_n$ est une suite de variables aléatoires i.i.d. avec $\mathbb{E}[X_1] < \infty$ et $\mathbb{E}[X_1^2] < \infty$, alors

$$\sqrt{n}\frac{\frac{1}{n}\sum_{i=1}^{n} X_i - \mathbb{E}[X_1]}{\sqrt{Var[X_1]}} \xrightarrow{\mathcal{D}} \mathcal{N}(0, 1)$$

a) Générer R réalisations $x_{1,r}, \ldots, x_{n,r}$ de la suite $X_1, \ldots, X_n$ dans le cas d'une loi gamma $X_1 \sim \mathcal{G}(1, 1)$, $(r = 1, \ldots, R)$. On a donc $\mathbb{E}[X_1] = 1$ et $Var[X_1] = 1$. On prendra $n = 1000$ et $R = 500$.
b) Calculer $\bar{x}_{n,r} = \frac{1}{n}\sum_{i=1}^{n} x_{i,r}$ pour chaque r.
c) Tracer (1) un histogramme de $\sqrt(n)(\bar{x}_{n,r} - 1)$ sur le même graphique que la fonction de densité de la loi $\mathcal{N}(0, 1)$ et (2) la fonction de répartition empirique de cette suite sur le même graphique que la fonction de répartition de la loi $\mathcal{N}(0, 1)$. Essayer avec d'autres valeurs de n. Conclusion ?
d) Procéder de même avec la loi binomiale $\mathcal{B}(10, 0.8)$.

Exercice 17. Intervalles de confiance.
a) Générer $n = 10$ i.i.d d'une v.a. $\mathcal{N}(0, 1)$. Donner un intervalle de confiance

bilatéral de niveau 99% sur la moyenne, la variance étant supposée inconnue elle aussi.
b) Estimer le niveau du test par le niveau empirique. Le niveau empirique correspond à la fréquence de présence du vrai centre (ici 0) dans l'intervalle de confiance. Conclusion ?

Exercice 18. Tests d'adequation.
a) Réecrire le test d'adéquation du χ^2 à une loi normale en considérant les deux paramètres moyenne et variance comme inconnus.
b) On considère un échantillon $x_1, x_2, \ldots, x_n$ de $\mathcal{N}(0,1)$.
· Tester l'hypothèse d'adéquation à une loi normale. On utilisera trois tests différents : χ^2, Kolomogorov-Smirnov et Shapiro-Wilk. On prendra n+100 et un risque $\alpha = 1\%$. On pourra prendre le découpage en classes suivant pour le test du χ^2 : $(-\infty,-1)$,$[-1,0)$,$[0,1)$ et $[1,\infty)$.
· Quel est le risque empirique associé à chaque test ?
c) On considère maintenant un échantillon $x_1, \ldots, x_n$ de $\mathcal{U}_{[-2,2]}$. Dans le cadre de l'hypothèse d'adéquation à une loi normale, estimer la puissance (pour n=100) associée à chaque test quand $\alpha = 5\%$. Classer alors les trois tests par ordre de préférence.

Exercice 19. Régression. On cosidère le modèle de régression $Y = 10 + x + 5\sin(x) + \epsilon$ avec $\epsilon \sim \mathcal{N}(0,4)$.
a) Pour des valeurs $x_i = i$, $(i = 1, \ldots, 30)$, générer un échantillon $y_1, \ldots, y_{30}$ i.i.d. à partir du modèle précédent. Tracer les couples de points (x_i, y_i) ainsi que la vraie courbe.
b) On suppose maintenant ignorer le modèle qui a généré les données.
· Estimer par moindres carrés les paramètres du modèle $Y = a + bx + \epsilon$. Tracer la courbe estimée sur le graphique précédent. Vérifier graphiquement la normalité des résidus. Que peut-on supposer ?
· Estimer par moindres carrés les paramètres du modèle $Y = a + bx + c\sin(x) + dx^2 + \epsilon$. Ajouter la courbe estimée au graphique précédent. Faire le test de nulité des coefficients. Que laisse supposer ce test ?
· Estimer par moindres carrés les paramètres du modèle $Y = a + bx + c\sin(x) + \epsilon$. Ajouter la courbe estimée au graphique précédent.
· Sélectionner le meilleur modèle au sens du critère AIC. Conclusion ? Aurait-on pu choisir le modèle avec le critère du R^2 ? Du $\bar{R}^2$? Essayer.
· Vérifier graphiquement la normalité des résidus de ce modèle, les résidus studentisés, la distance de Cook.

Exercice 20. Efficacité asymptotique et maximum de vraisemblance. On

considère un échantillon $X_1, \ldots, X_n$ i.i.d de $\mathcal{N}(\theta, 1)$, θ étant un paramètre inconnu.
a) On s'intèresse dans un 1er temps à l'estimation de θ par maximum de vraisemblance (MV).
· Donner l'estimateur $\hat{\theta}_n$ du MV de θ ainsi que sa loi. Quels sont le biais et l'efficacité de cet estimateur ?
· Visualiser le biais et l'efficacité de cet estimateur pour différentes tailles d'échantillon.
b) On s'intéresse maintenant à l'estimation de $P(X < \theta)$. Répondre alors aux mêmes questions qu'auparavant. Comparer aussi les résultats avec le cas précédent.

Exercice 21. Estimation non paramétrique par noyaux.
a) Écrire la fonction `get.sample` qui génére un échantillon de taille $n = 100$ dont la moitié provient d'une loi uniforme entre 0 et 2 et l'autre moitié provient d'une loi normale de centre 5 et de variance 1.
b) Écrire la fonction `show.all` qui trace trois densités sur le même graphique : (i) la vraie densité, (ii) la densité estimée avec noyau uniforme et (iii) la densité estimée avec noyau normal. Pour commencer, on prendra une valeur de fenêtre $h = 0.5$.
c) Faire varier n et h pour observer leur influence sur la qualité de l'estimation. Utiliser une fonction **R** qui calcule une fenêtre optimale et visualiser la solution proposée.

2 Idées programmées en R pour les solutions des exercices

```
#
# Les premiers pas ...
#

#
# Exercice 1 :
#

#
# a) Creer un repertoire nomme ...
# sous Windows il y a une option "Start In" ...
#
# b) Definir ...
#

#
# c) Creer un fichier de commandes ...
#
x=3
y=4

temp=y
y=x
x=temp

print(x)
print(y)

save(x,y,file="tp1.data")
#rm(lis=ls())
#load("tp1.data")

sink("tp1.sorties")
print(x)
print(y)
print("Programme execute")
```

```
sink();

print("Programme execute")


#
# Exercice 2 ...
#

#
# Exercice 3 ...
#

#
#  Structures des données~: vecteurs, matrices, listes, etc.
#

#
# Exercice 4 : Etudiez l'effet de lignes de commande suivantes
#

#
# Exercice 5 : Creer en un minimum de commandes ...
#

#a) la table de multiplication
a=1:10
b=1:5
c=outer(a,b,'*')

#b) la table d'addition
a=1:10
b=1:5
c=outer(a,b'+')

#c) une matrice 20 x 20 ...
a=matrix(seq(from=1,to=800,by=2),nrow=20,byrow=T)
index=as.character(1:20)
rownames(a)=paste("lig. ",index)
colnames(a)=paste("col. ",index)
```

```
#d) ... utiliser ?"+" ...
a[a%%3==0]=0
b=sum(as.numeric(a==0))

#e)
c=a[a!=0]

#f)
mc=mean(c)
c1=c[c<mc];
c2=c[c>=mc];
lst=list(petit=c1,grand=c2)

#g)
lst$petit=sort(lst$petit,decreasing=T)

#h)
a=matrix(1,10,10)-diag(10)
a=matrix(as.numeric(!diag(10)),nrow=10)

#
# Exercice 6 : On veut resoudre le systeme lineaire ...
#

#a)
a=matrix(c(1,1,1,1e12),ncol=2)

#b)
b=matrix(c(1.1,1.9),ncol=1)

#c)
x=solve(a,b)

p=eigen(a)$vectors
d=diag(eigen(a)$values)

#a1=p%*%d%*%solve(p)
a2=p%*%solve(d)%*%solve(p)
```

```
x2=a2%*%b

# ... ou bien
a=matrix(c(10,7,8,7,7,5,6,5,8,6,10,9,7,5,9,10),ncol=4)
b=matrix(c(32,23,33,31),ncol=1)

x=solve(a,b)

b1=matrix(c(32.1,22.9,33.1,30.9),ncol=1)
x1=solve(a,b1)

#
# Exercice 7 :
#

#a) Charger les donnees iris ...
data(iris)

#b) ... c'est une data.frame ...

#c) ... vecteurs numeriques et facteurs pour
#les colonnes, et listes pour les lignes ...

#d) ... proposer ...
attach(iris)
#detach(iris)

#e) Donner en une seule commande

levels(iris$Species)

#
# Exercice 8 : Programmer la fonction factorielle ...
#

fact = function(n)
{
 if(n>=0)
   {
     if (n==0){ f=1 }
```

```
     else { f=n*fact(n-1) }
   }
 else
   {
    print("ERREUR : valeur de n incorrecte !")
    f=-1
   }
 f
}

y=fact(6)

#
# Exercice 9 :
#

#a) Ecrire une fonction qui echange ...

echange=function(a,b)
{
 temp=a;
 a=b;
 b=temp;

 list("a"=a,"b"=b)
}

a=2
b=3

res=echange(a,b)

res$a

res$b

#b) Le plus grand diviseur commun de deux
# entiers naturels.

# Algorithme d'Euclide ...
```

```
pgcd=function(a,b)
{
 if((a<0)|(b<0))
   {
     print("ERREUR : a et/ou b doivent etre des entiers naturels")
     p=-1
   }
 else
   {
     if((a==b)|(b==0)){ p=a }
     else
       {
         if(a>b)
           {
             temp=echange(a,b)
             a=temp$a;b=temp$b
           }
         p=pgcd(a,b%%a)
       }
   }
 p

}

#c)  Forme irreductible d'une fraction.

irred=function(a,b)
{
 ta=abs(a)
 tb=abs(b)

 p=pgcd(ta,tb)

 c(a/p,b/p)
 #list("a"=a/p,"b"=b/p)
}

irred(4,6)
x=irred(4,6)
```

```
#d) Creer l'operateur binaire ...

"%sur%"=function(a,b)
{
 irred(a,b)
}

4%sur%84
25%sur%48

#
# Exercice 10 : On s'interesse a la serie ...
#

#a)

n=5
x=1:n
sn=x%*%x

sn=0
for(i in 1:n)
  {
    sn=sn+(i*i)
  }

#b)Indication : sur le site web r-project.org regarder
# dans le Search ...

# proc.time()
ptm=proc.time()

n=10^5;x=1:n;sn=x%*%x

proc.time()-ptm

#system.time(n=10^5;x=1:n;sn=x%*%x) ... voir le help pour lui ...

# proc.time()
ptm=proc.time()
```

```
n=10^5
sn=0
for(i in 1:n)
  {
    sn=sn+(i*i)
  }
proc.time()-ptm

# ... 3eme valeur la bonne ; l'unite de mesure ?

#
# Exercice 11 : Tester les commandes suivantes ...
#

x = rnorm(20) ; y =rexp(20)
plot(x,y)
points(x+.1,y+.1,pch=2)
lines(sort(x),y,lty=2)
text(1,1,"text")
abline(h=0.75)
title("texte")

#
# Exercice 12 :
#

#a) Afficher une projection du jeu de donnees iris ...

data(iris)
pairs(iris[1:4])

x11()
pairs(iris[1:4], main = "Iris Data",
      pch = c("+","*","o")[sort(iris$Species)],
      col = c("red", "green3", "blue")[sort(iris$Species)])

x11()
pairs(iris[1:4], main = "Iris Data",
      pch = c("+","*","o")[unclass(iris$Species)],
      col = c("red", "green3", "blue")[unclass(iris$Species)])
```

```
x11()
pairs(iris[1:4], main = "Iris Data",
      pch = 21, bg = c("red", "green3", "blue")[sort(iris$Species)])

x11()
pairs(iris[1:4], main = "Iris Data",
      pch = 21, bg = c("red", "green3", "blue")[unclass(iris$Species)])

#b) Faites plusieurs constatations visuelles

#c) Ecrire une fonction ...

afficherAxesIris= function(x,y)
{

  #data(iris)

  x11()
  plot(iris[,x],iris[,y],
       xlab=names(iris)[x],
       ylab=names(iris)[y],
       pch = c("+","o","*")[sort(iris$Species)],
       col = c("red", "green3", "blue")[sort(iris$Species)])

  legend("bottomright",levels(iris$Species),
         col = c("red", "green3", "blue"),
         pch=c("+","o","*"),bg="white")
}

#
# Exercice 13 :
#

#a) - superposees ...

sin.cos = function()

{
  x11()
```

```
  x=seq(from=0,to=2*pi,by=0.1)
  plot(x,sin(x),type="l",xlab="x",ylab="f(x)")

  x=seq(from=0,to=2*pi,by=0.2)
  points(x,cos(x),pch=20)

  legend("bottomleft",c("sin(x)","cos(x)"),
        lty=c(1,-1),pch=c(-1,20),bg="white")
}

# ... comment rajouter un point

#b) - deux graphes differentes ...
sin.cos.2 = function()
{
  x11()
  par(mfrow=c(2,1))
  x=seq(from=0,to=2*pi,by=0.1)
  plot(x,sin(x),type="l",xlab="x",ylab="sin(x)")

  x=seq(from=0,to=2*pi,by=0.2)
  plot(x,cos(x),pch=20,xlab="x",ylab="cos(x)")
}

#
# Exercice 14 : statistiques descriptives
#

#a) En une seule commande ...
data(iris)
summary(iris)

#b) Tracer sur ...
boxplot(iris[1:4])

#c) Tracer un camembert ...
attach(iris)
x11()
barplot(table(Species))
x11()
```

```
pie(table(Species))

#d) Tracer sur un premier graphique ...
x11()
qqnorm(Sepal.Length,main="All the Species")
qqline(Sepal.Length)

x=Sepal.Length[Species=="versicolor"]
x11()
qqnorm(x,main="Versicolor Only")
qqline(x)

#
# Exercice 15 : loi de grands nombres
#

#a) Generer une realisation ...
n=50000
x=rnorm(n,0,1)

#b) Calculer ...
y=cumsum(x)/(1:n)

#c) Tracer ...
plot(y,type="l",col="blue")
abline(h=0)

#d) Proceder de meme ...
x11()
x=rcauchy(n,0,1)
y=cumsum(x)/(1:n)
plot(y,type="l",col="blue")
abline(h=0)

#
# Exercice 16 : theoreme central limite
#

#a) Generer R realisations ...
n=1000
```

```
R=500
a=matrix(1,nrow=R,ncol=n)
for( i in 1 : R)
  {
   temp=rgamma(n,1,1)
   a[i,]=temp
  }

#b) Calculer ...
x=apply(a,1,mean)

#c) Tracer (1) un histogramme ...

x11()
y=sqrt(n)*(x-1)
hist(y,proba=T,col="blue")
z=seq(min(y),max(y),0.01)
lines(z,dnorm(z,0,1),lty=1,col="red")

x11()
Fn=ecdf(y)
plot(Fn,verticals=T,do.p=F)
lines(z,pnorm(z,0,1),lty=1,col="red")

#d) Proceder de meme ... E[X] = 10 * 0.8, Var[X] = 10*0.8*0.2
n=1000
R=500
a=matrix(1,nrow=R,ncol=n)
for( i in 1 : R)
  {
   temp=rbinom(n,10,0.8)
   a[i,]=temp
  }

x=apply(a,1,mean)

mu=10*0.8
sigma=sqrt(10*0.8*0.2)

x11()
```

```
y=sqrt(n)*(x-mu)/sigma
hist(y,proba=T,col="blue")
z=seq(min(y),max(y),0.01)
lines(z,dnorm(z,0,1),lty=1,col="red")

x11()
Fn=ecdf(y)
plot(Fn,verticals=T,do.p=F)
lines(z,pnorm(z,0,1),lty=1,col="red")

#
# Exercice 17 : intervalles de confiance
#

#a) Generer n = 10 ...
n=10
x=rnorm(n,0,1)

res=t.test(x,mu=0,alternative="two.sided",conf.level=0.99)
res$conf.int

#b) Estimer le niveau empirique du test ...
K=20000
a=1:K
for( i in 1 : K)
  {
   x=rnorm(n,0,1)
   res=t.test(x,mu=0,alternative="two.sided",conf.level=0.99)
   a[i]=(res$conf.int[1]<0)&(res$conf.int[2]>0)
  }

alpha=sum(a)/K

#
# Exercice 18 : tests d'adequation
#

#a) Reecrire le test d'adequation ...
chi2.adequation = function(x,intervals)
{
```

```
  # taille echantillon
  nx=length(x)
  # parametres loi
  mx=mean(x)
  sdx=sd(x)

  # nombre classes
  k=length(intervals)-1

  # initialise effectif empirique
  n.empi=1:(length(intervals)-1)

  # initialise effectif theorique
  n.theo=1:(length(intervals)-1)

  # calcul de la statistique
  for(i in 1: k)
    {
      n.empi[i]=sum(x>=intervals[i]&x<intervals[i+1])
      n.theo[i]=nx*(pnorm(intervals[i+1],mx,sdx)
                -pnorm(intervals[i],mx,sdx))
    }

  if(any(n.theo<5))
    {
      printf("WARNING : effectif theorique trop petit");
      break
    }

  # calcul de la statistique
  D=sum((n.empi-n.theo)^2/n.theo)

  # retourner la p-value
  list(p.value=1-pchisq(D,k-1-2))

}

# b) Tester l'hypothese d'adequation ...

# generer les variables normales et les intervalles ...
```

```
x=rnorm(100)
intervals=c(-Inf,-1 : 1, Inf)

# ... tester

chi2.adequation(x,intervals)$p.value
ks.test(x,"pnorm",0,1)$p.value
shapiro.test(x)$p.value

#  Quel est le risque empirique ?
risque.empirique = function(nessai)
{
 rejet.chi2=0
 rejet.ks=0
 rejet.shapiro=0

 for(i in 1:nessai)
   {
    x=rnorm(100)

    pvalue.chi2=chi2.adequation(x,c(-Inf,-1 : 1, Inf))$p.value
    if(pvalue.chi2 < 0.05)
      rejet.chi2=rejet.chi2+1

    pvalue.ks=ks.test(x,"pnorm",0,1)$p.value
    if(pvalue.ks < 0.05)
      rejet.ks=rejet.ks+1

    pvalue.shapiro=shapiro.test(x)$p.value
    if(pvalue.shapiro < 0.05)
      rejet.shapiro=rejet.shapiro+1
   }

 list(alpha.chi2=rejet.chi2/nessai,alpha.ks=rejet.ks/nessai,
      alpha.shapiro=rejet.shapiro/nessai)
}

risque.empirique(2000)

#
```

```
# c) On considere maintenant un echantillon ...
#

#
# On utilise la meme function avec une loi uniforme que l'on appele
# puissance.empirique ...
#
puissance.empirique = function(nessai)
{
 rejet.chi2=0
 rejet.ks=0
 rejet.shapiro=0

 for(i in 1:nessai)
   {
    x=runif(100,min=-2,max=2)

    pvalue.chi2=chi2.adequation(x,c(-Inf,-1 : 1, Inf))$p.value
    if(pvalue.chi2 < 0.05)
      rejet.chi2=rejet.chi2+1

    pvalue.ks=ks.test(x,"pnorm",0,1)$p.value
    if(pvalue.ks < 0.05)
      rejet.ks=rejet.ks+1

    pvalue.shapiro=shapiro.test(x)$p.value
    if(pvalue.shapiro < 0.05)
      rejet.shapiro=rejet.shapiro+1
   }

 list(pi.chi2=rejet.chi2/nessai,pi.ks=rejet.ks/nessai,
      pi.shapiro=rejet.shapiro/nessai)
}

puissance.empirique(1000)

#
# Exercice 19 : regression
#
```

```
#a)

# generation de l'echantillon et affichage du nuage de points
x=1:30
y=10+x+5*sin(x)+2*rnorm(30)
plot(x,y)

# afficher la vraie courbe
xall=seq(0,30,0.01)
yall=10+xall+5*sin(xall)
lines(xall,yall)

#b)

# estimer les parametres du modele
reg.simple=lm(y~x)

# estimer la droite de regression
attach(reg.simple)
y.simple=coefficients["(Intercept)"]+coefficients["x"]*xall
lines(xall,y.simple,lty=2)

# les residus on les voit avec ... plot(reg.simple)

# estimer les parametres du modele
reg.complex=lm(y~x+sin(x)+I(x^2))

# estimer la droite de regression
attach(reg.complex)
y.complex=coefficients["(Intercept)"]+coefficients["x"]*xall
         +coefficients["sin(x)"]*sin(xall)
         +coefficients["I(x^2)"]*xall^2
lines(xall,y.complex,lty=3)

summary(reg.complex)

# estimer les parametres du modele
reg.ok=lm(y~x+sin(x))

# estimer la droite de regression
```

```
attach(reg.ok)
y.ok=coefficients["(Intercept)"]+coefficients["x"]*xall
     +coefficients["sin(x)"]*sin(xall)
lines(xall,y.ok,lty=4)

summary(reg.ok)

#
# ... les criteres AIC choisissent le modele correct, alors que R^2 choisit
# le modele le plus complique ; R bar ^2 peut eventuellement convenir ...
#

AIC(reg.simple)
AIC(reg.complex)
AIC(reg.ok)

# ... pour verifier les residus
plot(reg.ok)

#
# Exercice 20 : efficacite asymptotique et maximum de vraisemblance
#

#a) On s'interesse dans un 1er temps ...

voir.biais.efficacite = function(tab.n)
{
 # on considere une loi N(0,1)

 tab.biais=1:length(tab.n)
 tab.variance=1:length(tab.n)

 for (i in tab.n)
   {
     theta.n=1:1000

     for(j in 1:1000)
       {
         x=rnorm(tab.n[i])
         theta.n[j]=mean(x)
```

```
      }

    tab.biais[i]=mean(theta.n)
    tab.variance[i]=var(theta.n)
  }


 split.screen(c(2,1))

 screen(1)
 plot(tab.n,tab.biais,type="l",col="blue",main="Biais")
 lines(tab.n,0*tab.n,lty="dotted",col="red")

 screen(2)
 plot(tab.n,tab.n*tab.variance,type="l",col="blue",main="Efficacite")
 lines(tab.n,0*tab.n+1,lty="dotted",col="red") # borne Cramer-Rao ...
}

voir.biais.efficacite(1:20)

# b) On s'interesse maintenant a l'estimation ...

voir.biais.efficacite = function(tab.n)
{
 # on considere une loi N(0,1)

 tab.biais=1:length(tab.n)
 tab.variance=1:length(tab.n)

 for (i in tab.n)
   {
     estim.n=1:1000

     for(j in 1:1000)
       {
         x=rnorm(tab.n[i])
         estim.n[j]=pnorm(mean(x))
       }

     tab.biais[i]=mean(estim.n)-0.5
```

```
    tab.variance[i]=var(estim.n)
   }


 split.screen(c(2,1))

 screen(1)
 plot(tab.n,tab.biais,type="l",col="blue",main="Biais")
 lines(tab.n,0*tab.n,lty="dotted",col="red")

 screen(2)
 plot(tab.n,2*pi*tab.n*tab.variance,type="l",col="blue",main="Efficacite")
 lines(tab.n,0*tab.n+1,lty="dotted",col="red") # borne Cramer-Rao ...
}

voir.biais.efficacite(1:50)

#
# Exercice 21 : estimation non parametrique par noyaux
#

#a) Ecrire la fonction get.sample ...

get.sample = function(n)
{
 #generer un echantillon uniforme sur [0,2] de taille n/2
 x.unif=runif(n/2,max=2)

 #generer un echantillon N(5,1) de taille n/2
 x.norm=rnorm(n/2,mean=5)

 c(x.unif,x.norm)
}


#b) Ecrire la fonction show all ...

show.all = function(x,h)
{
 # tracer la densite estimee avec un noyau uniforme
```

```
 f.unif=density(x,bw=h,kernel="rectangular")
 plot(f.unif$x,f.unif$y,type="l")

 # tracer la densite estimee avec noyau gaussien
 f.norm=density(x,bw=h)
 lines(f.norm$x,f.norm$y,type="l",col="blue")

 # tracer la vraie densite
 f.true=0.5*dunif(f.norm$x,max=2)+0.5*dnorm(f.norm$x,mean=5)
 lines(f.norm$x,f.true,type="l",col="red")
}

x=get.sample(1000)
show.all(x,0.1)

#c) Fenetre optimale
h=bw.nrd0(x)
```

3 Projets

3.1 Autour des processus ponctuels

Première partie : notions de base sur les processus ponctuels.

Les processus ponctuels ont comme réalisations des configurations aléatoires de points dans un espace mesuré $(S, \mathcal{A}, \lambda)$. Dans ce projet nous allons travailler sur des processus qui se déroulent dans un éspace fini $\lambda(S) < \infty$ et dont les configurations ont toujours un nombre fini de points.

Exercice I. En utilisant l'article "Likelihood inference for spatial point processes" de Charles Geyer, répondez aux questions suivantes :
a) Précisez les densités de probabilité $h(\boldsymbol{x})$ d'un processus ponctuel de Poisson, un processus ponctuel de Strauss et un processus ponctuel hard-core.
b) Calculez l'intensité conditionnelle $h(\boldsymbol{x} \cup \{\eta\})/h(\boldsymbol{x})$ pour chacun de ces modèles.
c) Calculez la constante de normalisation d'un processus ponctuel de Poisson. Peut-t-on calculer analytiquement la constante de normalisation d'un processus ponctuel de Strauss ? Et celle d'un processus ponctuel de type hard-core ? Justifiez votre réponse.

Deuxième partie : simulation des processus ponctuels.

Les densités de probabilités des processus ponctuels ont souvent des constantes de normalisation qui ne peuvent pas être calculées analytiquement. Par conséquent, pour simuler de tels processus il faut faire appel à des techniques Monte Carlo par chaînes de Markov.

Exercice II. Algorithme de Metropolis-Hastings En utilisant toujours le même article, répondez aux questions suivantes :
a) Précisez brièvement ce que c'est une chaîne de Markov $\varphi-$irréductible, récurrente au sens de Harris et géometriquement ergodique. Quelle est l'importance pratique de ces propriétés ?
b) Programmez en R l'Algorithme A présenté à la page 23 de l'article.

Troisième partie : inférence statistique.

Une fois, que nous avons construit des modèles si nous pouvons les simuler, alors nous pouvons faire de l'inférence statistique. Tous les processus seront

simulés dans un carré $S = [0,1] \times [0,1]$. Nous souhaitons calculer des intégrales du type :

$$\mathbb{E}_\theta [g(X)] = \int_\Omega g(\boldsymbol{x}) h_\theta(\boldsymbol{x}) \mu(d\boldsymbol{x})$$

avec $g(\boldsymbol{x}) : \Omega \to \mathbb{R}$, une fonction définie sur l'espace des configurations Ω et prenant des valeurs en $\mathbb{R}$. $h_\theta(\boldsymbol{x})$ est la densité de probabilité du processus ponctuel considéré.

Exercice III. Soit un processus ponctuel de Poisson avec $\theta = \ln(100)$. La statistique du modèle est $t(\boldsymbol{x}) = n(\boldsymbol{x})$ qui répresente le nombre de points dans le carré S.
a) Simulez $m = 1000$ réalisations $X_1, X_2, \ldots, X_m$ du processus ponctuel de Poisson considéré.
b) Affichez le histogramme des $t(X_1), t(X_2), \ldots, t(X_m)$ et l'évolution de leur moyennes cumulées.
c) Faites un test statistique pour vérifier si l'on peut vraiment affirmer que les $t(X_i)$ avec $i = 1, \ldots, m$ suivent une loi de Poisson $\mathcal{P}(100)$.
d) Utilisant des méthodes Monte Carlo proposez un estimateur de $\mathbb{E}_\theta [t(X)]$ ainsi que de $Var_\theta [t(X)]$. Comparer les résultats obtenus avec les résultats théoriques.
e) Calculez la probabilité $P(t(X) > 100)$ analytiquement et empiriquement.

Exercice IV. Soit un processus ponctuel de Strauss avec $\theta = \langle \ln(100), \ln(0.5) \rangle$ et $r = 0.05$. Les statistiques du modèle sont $t(\boldsymbol{x}) = \langle n(\boldsymbol{x}), s(\boldsymbol{x}) \rangle$ qui représentent le nombre de points dans le carré S et le nombre de paires de points plus proches que la distance r, respectivement.
a) Simulez $m = 1000$ réalisations $X_1, X_2, \ldots, X_m$ du processus ponctuel de Strauss considéré.
b) Affichez les histogrammes de chaque composante de $t(X_1), t(X_2), \ldots, t(X_m)$ ainsi que l'évolution de leur moyennes cumulées, respectivement.
c) Calculez l'espérance $\mathbb{E}_\theta [t(X)]$ et la probabilité $P(n(X) > 100)$ empiriquement.

Exercice V. Soit un processus ponctuel hard-core avec $\theta = ln(100)$ et $r = 0.05$. La statistiques du modèle est $t(\boldsymbol{x}) = n(\boldsymbol{x})$ qui représente le nombre de points dans le carré S.
a) Simulez $m = 1000$ réalisations $X_1, X_2, \ldots, X_m$ du processus ponctuel de Strauss considéré.
b) Affichez les histogrammes de chaque composante de $t(X_1), t(X_2), \ldots, t(X_m)$

ainsi que l'évolution de leur moyennes cumulées, respectivement.
c) Calculez l'espérance $\mathbb{E}_\theta[t(X)]$ et la probabilité $P(n(X) > 100)$ empiriquement.
d) Pour la suite des valeurs des paramètres

$$\exp(\theta) = \{10, 50, 100, 200, 500, 1000, 5000, 10000, 100000, 1000000\},$$

calculez les $\mathbb{E}_\theta[t(X)]$ correspondants. Qu'est-ce que vous observez ? Justifiez votre réponse. Est-ce que les points ont tendance à s'organiser d'une façon particulière ?

3.2 Statistiques et un peu d'astronomie aussi ...

Première partie : un peu d'ordre ...

Soit $Z_1, \ldots, Z_n$ une suite de variables aléatoires aléatoires indépendantes identiquement distribuées, et soit $F(z) = \mathbb{P}(Z \leq z)$ la fonction de répartition associée. F est considérée continue et le quantile d'ordre p, z_p est défini par $F^{-1}(p)$.

La statistique d'ordre de l'échantillon $(Z_1, \ldots, Z_n)$ est donnée par le réarrangement croissant de l'échantillon $(Z_{(1)}, \ldots, Z_{(n)})$. L'approximation des quantiles est une application immédiate de la statistique d'ordre.

Théorème 1. *Soit $(k(n), n \geq 1)$ une suite d'entiers telle que $1 \geq k(n) \geq n$ et $\lim_{n\to\infty} \frac{k(n)}{n} = p$. Alors la suite des quantiles empiriques $(Z_{(k(n))})$ converge presque sûrement vers z_p quand $n \to \infty$.*

Théorème 2. *Si Z_1 a une densité de probabilité telle que $f(z_p) > 0$ quand $p \in (0, 1)$ et si $k(n) = np + o(\sqrt{(}n))$, alors $Z_{(k(n))}$ converge en loi vers z_p de la manière suivante :*

$$\sqrt{(}n)(Z_{(k(n))} - z_p) \xrightarrow{\mathcal{L}} \mathcal{N}\left(0, \frac{p(1-p)}{f(z_p)^2}\right).$$

Exercice I. a) En fonction de F, précisez les fonctions de répartition de $M = \max_{i=1}^n Z_i$ et de $N = \min_{i=1}^n Z_i$.
b) Si Z_1 suit une loi uniforme dans $[0, 1]$ donnez les densités de probabilité de M et N, respectivement.

Exercice II a) Considérons $n = 3$ et Z_1 qui suit une loi uniforme sur $[0, 1]$. Tracez les histogrammes de M et N.

b) Sur les résultats obtenus superposez les courbes théoriques des densités de probabilité correspondantes.
c) Faites un test de Kolmogorov-Smirnov et un test χ^2 pour vérifier les résultats obtenus.
d) Nous souhaitons estimer $z_{0.99}$. En utilisant le Théorème 2, proposer un test statistique qui permet de déterminer la taille d'échantillon garantissant une estimation d'une qualité suffisante.
e) Dans les limites du possible, faites le même exercice dans le cas ou Z_1 suit une loi exponentielle de paramètre $\lambda = 0.5$.

Deuxième partie : ... des plans sur les comètes ...
Les jeux des données dont nous disposons représentent des perturbations planétaires qui sont appliquées aux comètes du nuage d'Oort. Il est tout à fait possible de regarder ces données comme des objets dans un espace $K \times M$. K est donnée par le rectangle $[0, 15] \times [-1, 1]$. $(q, \cos i)$, un point dans K représente la distance périhélique et l'angle d'inclinaison de la trajectoire elliptique d'une comète quelconque. M représente l'espace des perturbations $\triangle z$, qui prennent des valeurs dans $\mathbb{R}$.

En très simple, la dynamique des comètes est étudiée de la manière suivante. Premièrement, les perturbations planétaires sont calculées à l'aide d'un intégrateur numérique. Ces sont nos données. Ensuite, la perturbation $\triangle z$ qui se trouve en $(q, \cos i)$ est appliquée au demi-grand axe de la trajectoire de la comète ayant les mêmes périhélie et angle d'inclinaison que la perturbation considérée. Il en résulte que plus on a des perturbations calculées, plus précis sera l'étude de la dynamique des comètes.

Cependant le calcul des perturbations est vraiment très, très coûteux ... D'ou l'interêt des statistiques ...

Télécharghez les deux jeux des données disponibles à partir de ma page web :

`http://math.univ-lille1.fr/~stoica/`

Le premier jeu de données est localisé dans le domaine $K_1 = [0, 7] \times [-1, 1]$, alors que le deuxième est localisé dans $K_2 = [7, 15] \times [-1, 1]$. Les valeurs $q = 5.20$ et $q = 9.55$ correspondent aux demi-grand axes de Jupiter et Saturne. Cela veut dire que la comète ayant un périhélie proche peut croiser l'orbite d'une de ces grandes planètes. Ainsi, on attend que les perturbations

dans ces régions soit plus grandes ... statistiquement parlant ... Malgré le fait, que les perturbations soient uniformément réparties dans K.

L'objectif de cette partie du projet est de caractériser statistiquement ces perturbations ...

Exercice III.
a) Est-ce que ces données sont stationnaires ? Localement ? Dans tout K ? Argumentez.
b) Faites une analyse exploratoire de ces deux jeux de données. Reflechissez au choix de vos statistiques. Argumentez ce choix.
c) Est-ce que vous observez des structures particulières ?
d) Avez-vous des indications concernant la modélisation par une loi de probabilité de ces perturbations ?
e) Est-ce que vous pouvez proposer un test qui rejette certaines lois concernant ces perturbations. Exemple : loi normale, exponentielle bilatérale, Cauchy ou autre ? Si oui, il faudrait expliquer comment vous estimez les paramètres des lois testées ...

References

[1] C. Biernacki. *Logiciel R - Tutorial avec applications statistiques.* Cours Polycopié Université Lille 1, 2007.

[2] A. Philippe. *Notes de cours sur le logiciel R.* Cours Polycopié Université de Nantes, 2011.