Initiation aux Probabilités sous Scilab
MA101

October 10, 2017

Contents

1 Introduction à Scilab

1.1 Description

Scilab (contraction de Scientific Laboratory) est un logiciel libre, développé conjointement par l'INRIA et l'ENPC. Il est téléchargeable gratuitement à partir de

http://scilabsoft.inria.fr/

C'est un environnement de calcul numérique qui permet d'effectuer rapidement toutes les résolutions et représentations graphiques couramment rencontrées en mathématiques appliquées.

Scilab (qui ressemble beaucoup à Matlab) est basé sur le principe que tout calcul, programmation ou tracé graphique peut se faire à partir de matrices rectangulaires. En Scilab, tout est matrice : les scalaires sont des matrices, les vecteurs lignes des matrices, les vecteurs colonnes des matrices.

1.2 Quelques commandes utiles pour commencer

1.2.1 B.A.-Ba

Dans une ligne de commande, tout ce qui suit `//` est ignoré, ce qui est utile pour les commentaires. Les commandes que nous proposons sur des lignes successives sont supposées être séparées par des retours-chariots.

```
A=[1,2,3;4,5,6;7,8,9],// définit une matrice 3X3
```

Ajouter un point virgule en fin de ligne supprime l'affichage du résultat (le calcul est quand même effectué). Ceci évite les longs défilements à l'écran, et s'avère vite indispensable.

```
x=ones(1,100);// rien n'apparaît
x,// le vecteur x a bien été défini
```

Les résultats sont affectés par défaut à la variable `ans` ("answer"), qui contient donc le résultat du dernier calcul non affecté. Toutes les variables d'une session sont globales et conservées en mémoire. Des erreurs proviennent souvent de confusions avec des noms de variables déjà affectés. Il faut penser à ne pas toujours utiliser les mêmes noms, ou à libérer les variables par `clear`. Les variables courantes sont accessibles par `who` et `whos`.

```
a=[1,2];A=[1,2;3,4];// affecte a et A
1+1,// affecte ans
//who;                    // toutes les variables
whos(),// les détails techniques
clear a
who ;// a disparaît
clear
who ;// a, A et ans disparaissent
xbasc(),// efface le contenu de la fenêtre  graphique active
```

La commande `stacksize` permet de connaître la taille de la pile utilisée par Scilab pour stocker les variables. Si cette taille est trop faible, on peut l'ajuster grâce `stacksize(n)` où `n` est un entier. Pour plus de détails, faire `help stacksize`.

L'aide en ligne est appelée par `help`. La commande `apropos` permet de retrouver les rubriques d'aide quand on ignore le nom exact d'une fonction.

```
help help
help apropos
apropos matrix          // rubriques dont le titre contient "matrix"
help matrix             // aide de la fonction "matrix"
```

1.2.2 Opérations sur les matrices

Toutes les opérations sont matricielles. Tenter une opération entre matrices de tailles non compatibles retournera en général un message d'erreur, sauf si une des matrices est un scalaire. Dans ce cas, l'opération (addition, multiplication, puissance) s'appliquera terme à terme.

```
A=[1,2,3;4,5,6]
A+ones(1,3),// erreur
A+ones(A)
A+10
A*10
A*ones(A),// erreur
A*ones(A')
A'*ones(A)
```

Les opérations terme à terme sur les matrices s'effectuent grâce à l'opérateur classique `+ -` et `* / ^` précédé d'un point.

En résumé les différentes opérations matricielles sont :
`+, -` addition, soustraction
`* , ^` multiplication, puissance (matricielles)
`.* ,. ^` multiplication terme à terme, puissance terme à terme
`A\b` solution de A*x=b
`b/A` solution de x*A=b
`./` division terme à terme

Cette brève présentation de Scilab est inspirée des manuels "Scilab: une introduction" de Jean-Philippe Chancelier et "Démarrer en Scilab" de Bernard Ycart.

1.2.3 Commandes statistiques

Lire et exécuter les lignes suivantes.

initiation.sce

```
xbasc();
clear

n=1000;
```

```
Y=rand(n,1);// Renvoie un vecteur colonne dont les entrées sont n
// réalisations pseudo-aléatoires  selon la loi uniforme
// sur [0,1] et pseudo-indépendantes

moyenne=sum(Y)/length(Y),//  Moyenne arithmétique =
//  (Y(1)+...+Y(n))/n

mediane=median(Y),// Médiane

plot2d([1:n],Y),// Tracé du vecteur Y. Dans ce cas [1:n] est facultatif

xset('window',1)
histplot(20,Y),// Histogramme à 20 classes
```

2 Loi Forte des Grands Nombres

Soit $(Y_1, \ldots, Y_n)$ un n-échantillon de variables aléatoires de loi uniforme sur $[0,1]$. En regardant l'aide sur la fonction `cumsum` (tapez `help cumsum`), calculez le vecteur $\bar{Y} = (\bar{Y}_1, \ldots, \bar{Y}_n)$ des moyennes empiriques où $\bar{Y}_i = \frac{1}{i}\sum_{k=1}^{i} Y_k$ et tracez l'évolution de la moyenne empirique $i \longmapsto \bar{Y}_i$ à l'aide de la fonction `plot2d`.

3 Théorème central limite

On considère un n-échantillon $(Z_1, \ldots, Z_n)$ où les variables aléatoires Z_i sont i.i.d de même loi que $\sqrt{12p}\left(\frac{1}{p}\sum_{i=1}^{p} U_i - \frac{1}{2}\right)$, les variables aléatoires $(U_i, i \leq p)$ étant i.i.d de loi uniforme sur [0,1].

Utiliser le programme suivant pour tracer l'histogramme à nc classes de $(Z_i, i \leq n)$. Faites varier n, p, nc. Qu'observez-vous pour $p = 1$, $p = 12$, nc grand et nc petit? On choisira n de l'ordre de 1000.

TCL.sce

```
function tcl(n,p,nc)
  xbasc();
  X=rand(n,p);
  Z=sqrt(12/p)*(sum(X,'c')-p/2);// somme des colonnes de X, centrage et
  // renormalisation
  histplot(nc,Z)
  C=[-5:1/1000:5];
  plot2d(C,exp(-C .^2/2)/sqrt(2*%pi),3);// densité de la loi N(0,1)
endfunction
```

4 Cas des vecteurs gaussiens

On veut simuler des variables aléatoires indépendantes suivant la loi gaussienne centrée réduite $\mathcal{N}(0,1)$.

1. Soient R et θ deux variables aléatoires indépendantes. On pose $X = R\cos(\theta)$ et $Y = R\sin(\theta)$. Montrer que si R^2 suit une loi exponentielle de paramètre $\frac{1}{2}$ et θ est uniformément répartie sur $[0, 2\pi]$, alors (X, Y) a pour densité $\frac{1}{2\pi}e^{-\frac{x^2+y^2}{2}}$.

2. Montrer que si U suit la loi uniforme sur $[0,1]$ alors $-\frac{1}{\lambda}\log(U)$ suit la loi exponentielle de paramètre λ.

3. En déduire que si (U_1, U_2) sont deux variables aléatoires indépendantes de loi uniforme sur $[0,1]$ alors $(\sqrt{-2\log(U_1)}\cos(2\pi U_2), \sqrt{-2\log(U_1)}\sin(2\pi U_2))$ est un couple de variables aléatoires indépendantes de loi $\mathcal{N}(0,1)$.

On veut maintenant simuler un vecteur gaussien centré de matrice de covariance $\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$. Montrer que si $\begin{pmatrix} G_1 \\ G_2 \end{pmatrix}$ suit la loi $\mathcal{N}\left(0, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right)$ alors

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \rho & \sqrt{1-\rho^2} \end{pmatrix} \begin{pmatrix} G_1 \\ G_2 \end{pmatrix}$$

suit la loi $\mathcal{N}\left(0, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}\right)$.

Remarque. La matrice $\mathcal{L} = \begin{pmatrix} 1 & 0 \\ \rho & \sqrt{1-\rho^2} \end{pmatrix}$ est en fait la décomposition de Cholesky de $M = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$ (i.e. $\mathcal{L}\mathcal{L}^T = M$ et $\mathcal{L}$ triangulaire inférieure). Cette méthode permet de simuler un vecteur gaussien dès que l'on a calculé la décomposition de Cholesky de sa matrice de covariance. Sous Scilab la fonction `chol` permet de calculer la décomposition de Cholesky d'une matrice symétrique. Cette fonction renvoie $\mathcal{L}^T$. Une version de l'algorithme de Cholesky est proposée en annexe.

Utilisez le programme suivant pour simuler un vecteur gaussien centré et faites varier ρ.

vecteur_gaussien.sce

```
function gaussian_vector(rho)
  if abs(rho) > 1 then
    disp('la corrélation doit être comprise entre -1 et 1!')
    disp('aborting...')
    return
  end
```

```
  xbasc();
  n=1000;
  u1=rand(1,n);
  u2=rand(1,n);
  g1=ZZ;// à compléter
  g2=ZZ;// à compléter
  A=[1,0;rho,sqrt(1-rho^2)];
  z=A*[g1;g2];
  x=z(1,:);
  y=z(2,:);
  plot2d(x,y,-1)
endfunction
```

5 Estimateurs à noyaux

On cherche à estimer la densité de la loi d'un échantillon. La première approche est de tracer l'histogramme renormalisé des valeurs obtenues.

Une autre est d'estimer la densité de l'échantillon simulé par la méthode des noyaux décrite ci-après.

Soit f la densité de probabilité à estimer. Soit $(X_1,...,X_n)$ un échantillon de la v.a. i.i.d de loi de densité f. La mesure empirique

$$\Pi_n = \frac{1}{n}\sum_{i=1}^{n} \delta_{X_i}$$

de l'échantillon, où δ_x désigne la mesure de Dirac au point x, "converge" vers la mesure μ. Mais cette mesure empirique n'admet pas de densité par rapport à la mesure de Lebesgue. C'est pourquoi, on "régularise par convolution" Π_n avec une suite de noyaux $(K_h)_{h>0}$ qui vérifie :

$$\begin{cases} K_h(x) \geq 0 & \text{pour tout} \quad h>0 \quad \text{et tout} \quad x \in \mathbb{R} \\ \int_{\mathbb{R}} K_h(x)dx = 1 & \text{pour tout} \quad h>0 \\ K_h \xrightarrow[h\to 0]{} \delta_0. \end{cases}$$

On peut ainsi considérer la suite de noyaux $K_h(x) = K(x/h)$ où K peut par exemple désigner le noyau gaussien

$$K(x) = \frac{1}{\sqrt{2\pi}} \exp(-x^2/2)$$

ou le noyau d'Epanechnikov

$$K(x) = \frac{3}{4}(1-x^2)\mathrm{I}_{]-1,1[}(x).$$

On estime alors la densité f par la fonction

$$\hat{f}_n(x) = \frac{1}{nh_n}\sum_{i=1}^{n} K\left(\frac{x-X_i}{h_n}\right).$$

La suite $\hat{f}_n$ converge vers f. On admettra qu'un choix judicieux pour la suite $(h_n)_n$ est de prendre $(h_n)_n$ de l'ordre de $= n^{-1/5}$.

On considère X et Y deux variables aléatoires gaussiennes centrées, réduites et indépendantes. On définit une nouvelle variable aléatoire Z telle que Z vaut X avec probabilité $\frac{1}{3}$ et $aY + b$ avec probabilité $\frac{2}{3}$ où $a > 0$ et $b \in \mathbb{R}$. Vérifiez que la densité f de Z s'écrit

$$f(x) = \frac{2}{3a\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-b}{a}\right)^2} + \frac{1}{3\sqrt{2\pi}} e^{-\frac{x^2}{2}}.$$

Utilisez le programme suivant pour voir comment évolue l'estimation de la densité en fonction de h. La vraie densité apparaît en rouge sur le graphique.

noyau.sce

```
clear ;

// nc est le nombre de classes dans l'histogramme
// n la taille de l'échantillon
// h le pas de l'ordre de n^(-1/5)
function estim_noyau(n,nc,hn)
  xbasc();
  a=1;
  b=3;
  // 2 variables alétoires uniformes
  U=rand(n,1);
  V=rand(n,1);

  // 2 gaussiennes centrées réduites indépendantes
  X=sqrt(-2*log(U)) .*cos(2*%pi*V);
  Y=sqrt(-2*log(U)) .*sin(2*%pi*V);

  // Z = X avec proba 1/3 et aY+b avec proba 2/3
  epsilon=rand(n,1);
  Z=(a*Y+b) .*(epsilon > 1/3)+X .*(epsilon <= 1/3);

  // histogramme de la variable simulée E, nc=nombre de classes
  histplot(nc,Z)

  // estimation de la densité de la loi de E par la méthode des noyaux,
  // noyau Epanechnikov :
  C=[min(Z)-1:1/n:max(Z)+1];
  for i=1:length(C) do
    B(i)=1/(n*hn)*3/4*sum((1-((C(i)-Z)/hn) .^2) .*(-1 < (C(i)-Z)/hn) .*((C(i)-Z)/hn < 1)
```

```
  end

  plot2d(C,B,2)

  //tracé de la vraie densité
  f=(2/(3*a)*exp(-((C-b)/a) .^2/2)+1/3*exp(-C .^2/2))/sqrt(2*%pi);
  plot2d(C,f,5);
endfunction
```

6 Méthode de Monte-Carlo

La méthode de Monte-Carlo permet le calcul de valeurs approchées d'intégrales multiples en utilisant des réalisations i.i.d. de loi uniforme. Si $(X_n)_{n\geq 1}$ est une suite de v.a. i.i.d. de loi uniforme sur $[0,1]^m$ et si $f : [0,1]^m \to \mathbb{R}$ est une fonction mesurable, alors la loi des grands nombres appliquée à la suite de v.a.r. i.i.d. $(f(X_n))_{n\geq 1}$ entraîne la convergence presque-sûre suivante :

$$\frac{1}{n}\left(f(X_1)+\cdots+f(X_n)\right) \underset{n\to+\infty}{\longrightarrow} \mathbb{E}[f(X_1)] = \int_{[0,1]^m} f(x)\,dx,$$

Si la variance de $f(X_1)$ existe, le TCL assure que la convergence à lieu en $1/\sqrt{n}$. Cette vitesse est relativement lente en petite dimension comparée aux méthodes déterministes. En revanche elle devient très pertinente en dimension élevée. De plus cette méthode ne demande, à priori, aucune régularité particulière sur f.

Vérifiez que les trois intégrales suivantes sont bien égales à π. Utilisez le programme suivant pour comprendre comment approximer la valeur de π. Quelle est selon vous la plus efficace?

$$\int_{[0,+1]} 4\sqrt{1-x^2}\,dx$$

$$\int_{[-1,+1]^2} \mathrm{I}_{(x^2+y^2\leq 1)}\,dxdy \qquad \text{et} \qquad \int_{[-1,+1]^3} \frac{3}{4}\mathrm{I}_{(x^2+y^2+z^2\leq 1)}\,dxdydz.$$

MonteCarlo.sce

```
function MC_pi(n)
  x=4*(1-rand(1,n) .^2) .^(1/2);
  X=cumsum(x) ./(1:n);
  var_x=cumsum(x .^2) ./(1:n)-X .^2;

  y=(4*(rand(1,n) .^2+rand(1,n) .^2 <= 1));
  Y=cumsum(y) ./(1:n);
  var_y=cumsum(y .^2) ./(1:n)-Y .^2;

  z=(3*2*(rand(1,n) .^2+rand(1,n) .^2+rand(1,n) .^2 <= 1));
```

```
  Z=cumsum(z) ./(1:n);
  var_z=cumsum(z .^2) ./(1:n)-Z .^2;

  PI=%pi*ones(1,n);

  xset('window',0)
  xbasc(0);
  plot2d([1:n],PI,1)

  plot2d([1:n],X,3)
  plot2d([1:n],X+1.96*sqrt(var_x) ./sqrt([1:n]),2);//Intervalle de confiance
  plot2d([1:n],X-1.96*sqrt(var_x) ./sqrt([1:n]),2);//à 95%

  xset('window',1)
  xbasc(1);
  plot2d([1:n],PI,1)

  plot2d([1:n],Y,4)
  plot2d([1:n],Y+1.96*sqrt(var_y) ./sqrt([1:n]),2);//Intervalle de confiance
  plot2d([1:n],Y-1.96*sqrt(var_y) ./sqrt([1:n]),2);//à 95%

  xset('window',2)
  xbasc(2);
  plot2d([1:n],PI,1)

  plot2d([1:n],Z,5)
  plot2d([1:n],Z+1.96*sqrt(var_z) ./sqrt([1:n]),2);//Intervalle de confiance
  plot2d([1:n],Z-1.96*sqrt(var_z) ./sqrt([1:n]),2);//à 95%
endfunction
```

A Algorithme de Cholesky

Voici l'algorithme utilisé pour calculer la décomposition de Cholesky $\mathcal{L}$ d'une matrice symétrique positive A.

$$\begin{aligned}
&\text{pour } k = 1..size \text{ faire} \\
&\quad \mathcal{L}_{k,k} = \sqrt{A_{k,k} - \sum_{j<k} \mathcal{L}_{k,j}^2}, \\
&\quad \text{pour } i = k+1..size \text{ faire} \\
&\quad\quad \mathcal{L}_{i,k} = \frac{A_{i,k} - \sum_{j<k} \mathcal{L}_{k,j}\mathcal{L}_{i,j}}{\mathcal{L}_{k,k}}.
\end{aligned}$$