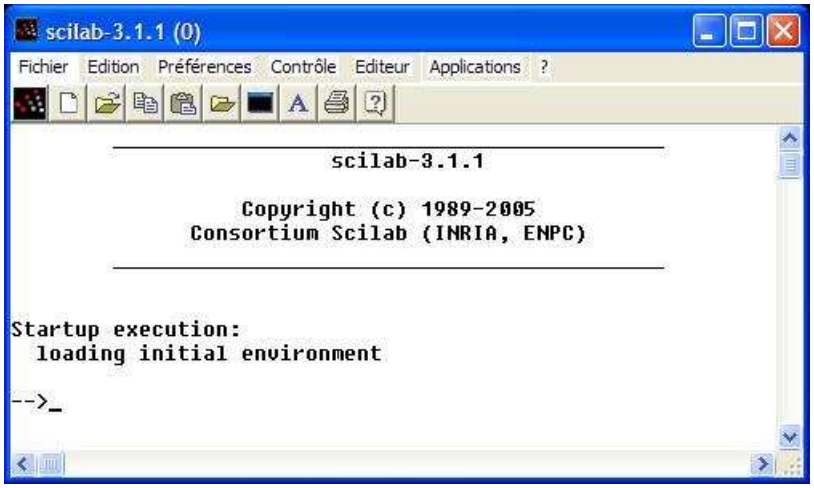
Université de Nice, Département de mathématiques , année 2005-2006

TP de Statistique (projet)

0. Lancer Scilab



1. Simulation de points aléatoire dans un carré

1.1. Taper l'instruction

```
rand()
```

puis valider. Exécuter plusieurs fois cette commande *(la touche <curseur haut> fait revenir les instructions données à Scilab, dans l'ordre de la dernière à la première)* . Qu'observe t-on ?

1.2. On va définir une suite a(n) indexée par les entiers de 1 à 1000 et prenant les valeurs successive donnée par rand() : Taper et valider l'instruction

```
a=1:1000;for i=1:1000,a(i)=rand(),end
```

puis 'n' à la question intempestive [More (y or n) ?] posée.

Pour vérifier que la suite ainsi générée est uniformément répartie dans le segment [0,1] on peut afficher dans un graphique la suite des points de coordonnées (0,a(i)), i décrivant 1..30. Taper (et valider) l'instruction

```
xbasc();for i=1:30,plot2d(a(i),0,0,rect=[-.2,-.5,1.2,.5]),end
```

Qu'observe t-on ? Changer le nombre 30 en 200 puis en 1000. Est-ce concluant ?

Une autre méthode pour controler la répartition des valeurs des a(i) dans le segment [0,1] est de découper le segment en 20 intervalles et de calculer pour chacun de ces intervalles le nombre d'indices i tels que a(i) tombe dans l'intervalle. On affiche le résultat (les 20 nombres obtenus) sous une forme graphique qu'on appelle histogramme. La commande suivante fait tout ce travail :

```
xbasc();histplot(20,a)
```

1.3 Modifier l'instruction définissant a pour obtenir une suite de nombre aléatoire uniformément répartie dans le segment [-1,1] puis définir une suite b par la même instruction. On obtient alors une suite de couples (a(i),b(i)) qu'on va représenter graphiquement avec l'instruction

```
xbasc();for i=1:1000,plot2d(a(i),b(i),0,rect=[-1.2,-1.2,1.2,1.2]),end
```

La suite de points semble t-elle uniformément répartie dans le carré [-1,1]x[-1,1] ? Comparer avec ce qu'on obtient si on redéfinit la suite a par l'instruction

```
for i=1:1000,a(i)=-1+2*rand()^2,end
```

Changer de nouveau a en la suite demandée au début de 1.3

2. Approximation de pi

2.1 On va compter le nombre d'indices i tel que le point de coordonnées (a(i),b(i)) définit en début de 1.3 soit dans le disque de centre 0 et de rayon 1.
Essayer d'abord les instructions

```
bool2s(1<=2)
```

et

```
bool2s(2<1)
```

Ecrire une instruction dépendant de deux paramètres x et y qui rende 1 si le point (x,y) est dans le disque de centre (0,0) et de rayon 1, qui rende 0 sinon, et tester la.
Ecrire une instruction qui définisse une suite c avec c(i)=1 si le point d'indice i est dans le disque, c(i)=0 sinon.
On obtient maintenant le nombre de points dans le disque avec l'instruction

```
sum(c)
```

2.2 Si la répartition des points (a(i),b(i)), i décrivant 1,..,1000 est uniforme sur le carré, le nombre de points dans le disque de centre (0,0) et de rayon 1 doit être proche du quotient de l'aire du disque par celle du carré.
Transformer l'instruction en fin de 2.1 pour obtenir une approximation de pi.

2.3 Recommencer les instructions définissant a,b,c jusqu'à obtenir de nouvelles approximations de pi. Quelles variations observe t-on ?

3. Création d'une série de 400 valeurs approchées de pi.

On va créer une suite d(i), i décrivant les entiers de 1 à 400, contenant des valeurs approchées de pi obtenues par les instructions de 2.3
Ecrire une instruction sous forme de boucle (une instruction commençant par 'for i=1:400') répétant 2.3 et affectant à d(i) le résultat.
Calculer la moyenne des valeurs prises par les d(i) avec l'instruction

```
mean(d)
```

La valeur obtenue est elle plus proche de pi que les valeurs obtenues en 2.2 ?

4. Comparaison de la série avec une distribution normale

4.1 On va visualiser la suite d à l'aide d'un histogramme comme on la fait pour la suite a en 1.2. Cette fois on découpe le segment [2.97,3.31] en intervalles de longueur 0.02. On crée la suite x des extrémités de ces intervalles par l'instruction

```
x=2.97:0.02:3.31
```

Comme certaines valeurs de d peuvent sortir du segment [2.97,3.31] on extrait de d les seules valeurs tombant dans ce segment par l'instruction

```
de=d(2.97<=d & d<=3.31)
```

On fabrique maintenant l'histogramme par l'instruction

```
xbasc();histplot(x,de)
```

La commande histplot est telle que la suite des intervalles de [2.97,3.31] dans lesquels on compte le nombre d'occurence d'un élément de d est [x(0),x(1)],]x(1),x(2)],]x(2),x(3)],...
L'histogramme est normalisé : cela signifie que la graduation sur l'axe des y a été choisie par défaut de telle sorte que la somme des aires des rectangles vaut 1.

4.2 Pour comparer avec une loi normale ayant l'espérance et l'écart-type idéals de la série (c'est à dire correspondant à une suite d'approximation de pi de longueur infinie) on va dessiner dans la même fenêtre le graphe de la densité de la loi normale

$$f(x) = \frac{1}{s\sqrt{2\pi}}\,\mathrm{e}^{-\frac{(x-m)^2}{2s^2}}$$

où m est l'espérance (idéale) et s l'écart-type (idéal).
Que vaut m ? Définir m dans Scilab par une instruction.

On donne

$$s = \sqrt{\frac{m(4-m)}{1000}}$$

ce dont on instruit Scilab par l'instruction

```
s=sqrt(m*(4-m)/1000)
```

On définirait dans Scilab la fonction g(x)=cos(2pi*x) par la suite d'instructions

```
function y=g(x);
y=cos(2*%pi*x);
endfunction
```

Modifier cette suite d'instructions pour définir la fonction f ci-dessus(dépendant des paramètres m et s) .
(Les variables m et s devront être définies avant tout appel à la fonction f. Elles pourront éventuellement être redéfinies avant chaque appel à f.)
On trace le graphe de f dans la même fenètre que l'histogramme par l'instruction

```
fplot2d(2.97:0.001:3.31,f)
```

Qu'observe t-on ?

5. Pourcentage de réussite

Les valeurs prises par la suite d(i), i=1..400 (définie en 3.) sont des valeurs approchées du nombre pi. On va caractériser la dispersion de cette série par son écart-type se et on va calculer le pourcentage d'indices i tel que pi soit dans l'intervalle]d(i)-se,d(i)+se]
5.1 Calculer l'écart-type effectif de la série d avec l'instruction

```
se=sqrt(variance(d))
```

5.2 Ecrire une instruction analogue à celle de 2.1 définissant une suite e(i), i=1..400, telle que pour tout i e(i)=1 si pi est dans l'intervalle]d(i)-se,d(i)+se] et 0 sinon. Calculer la somme des valeurs prises par e puis le pourcentage d'indices cherché.
5.3 Refaire ce calcul en remplaçant se par 2*se.

*5.4 Pouvez vous écrire une fonction Scilab donnant les deux pourcentages réactualisés à chaque appel de la fonction *(voir l'aide en ligne de Scilab en appuyant sur <F1> dans la fenêtre de commande, chercher dans l'aide le mot clef 'function'.)*

6. Compte-rendu de séance

Lancez le programme Wordpad ou OpenOffice ou un éditeur html avec lequel vous allez écrire les instructions données à Scilab, recopier les résultats obtenus et écrire vos observation. Imprimer votre document quand vous aurez fini.
On peut sélectionner les instructions tapées dans la fenêtre de commande de Scilab ou les résultats fournis par Scilab et les copier ('Ctrl C' ou menu 'édition', 'copier') pour les coller (Ctrl V) dans l'éditeur que vous avez choisi pour votre compte-rendu. Lorsqu'une instruction dans Scilab ouvre une fenêtre graphique, on peut sauvegarder le graphique dans le presse-papier (dans la fenêtre "Scilab Graphic", sélectionner le menu 'Fichier' 'Copier dans le presse papier' 'EnhMetafile') et coller le résultat dans l'éditeur. On peut aussi sauvegarder le graphique dans un fichier au format BMP (menu 'Fichier' 'Exporter') et le réutiliser plus tard pour le compte-rendu (menu 'Insertion' ou un simple "glisser-déposer" dans l'éditeur si c'est possible).

Au boulot !

Adapté de la feuille de TP 2 de Statistiques en Licence MI/MP 2ème année, 2004-2005 "Méthode de Monte Carlo".
Page fabriquée avec l'éditeur Nvu, une commande de saisie de fenêtre au format jpg (touche 'Prt Scrn' sous Windows xp, la commande 'import' de ImageMagick sous Linux), Latex, la commande 'convert' de ImageMagick

F-X Dehon, 20 décembre 2005