Préparation Agrégation Bordeaux 1

Année 2013 -2014

TP Statistique n°1

L'objectif de ce TP est de vous familiariser avec le logiciel scilab (qui est libre). Scilab n'est pas à la base un langage de calcul formel comme Maple. Il s'agit d'un interpréteur de commandes écrites en langage scilab. Ce logiciel a été conçu pour faciliter les opérations sur les vecteurs, matrices et tableaux.
Pour les probabilités, on utilisera les fonctions du module "stixbox" que l'on peut facilement télécharger.
Vous pouvez aussi aller travailler au CREMI quand vous voulez, et même vous connecter à distance sur les machines du CREMI depuis chez vous, en suivant les consignes sur l'intranet du site du CREMI.
Après les TPs, les corrigés seront sur ma page
http://www.math.u-bordeaux1.fr/~chabanol/TP_ agreg.html

1. Fonctionnement des TP

La commande suivante à taper dans un terminal vous ouvrira une fenêtre, dans laquelle vous aurez à tout instant une copie de mon écran :

```
krdc vnc://nom_du_serveur
```

(Je vous communiquerai le nom_du_serveur à chaque séance)

2. Pour commencer

Commencer par créer un répertoire où seront stockés tous vos fichiers scilab... On travaillera ensuite dans ce répertoire.

A VERIFIER AU CREMI Lorsqu'on lance scilab une fenêtre s'ouvre, comportant la console où sont tapées les instructions et où s'affichent les résultats. On peut aussi, si elles ne sont pas déjà visibles, à partir de l'onglet "Applications", faire apparaître les fenêtres avec l"'historique des commandes", avec le "navigateur de variables" (qui montre les variables utilisées), ainsi que l'éditeur "Scinotes" ... qui servira beaucoup.
Enfin on peut changer si nécessaire le répertoire courant avec l'onglet "Fichier".
Vérifiez que vous êtes bien dans le bon répertoire.

On peut afficher l'aide à l'aide de F1. De manière générale, "help nom" fournit de l'aide sur le mot clé "nom".
1. Trouver une fonction qui donne les valeurs propres d'une matrice (en anglais, valeur propre se dit eigenvalue).

Scilab conserve l'historique des commandes. Il est donc possible de récupérer des instructions déjà saisies : en double cliquant sur une de ces instructions passées, on la réexécute.

Enfin, pour sauver certaines commandes de l'historique dans un fichier, on peut sélectionner des entrées dans cette fenêtre faire un clic droit et les ouvrir dans l'éditeur.

↑, ↓, →, ← permet de se déplacer dans les lignes de commandes tapées dans la console. Cela évite de retaper des instructions !
La tabulation permet quand on a commencé de taper une commande de rechercher une complétion automatique.

ATTENTION l'historique peut dépendre de l'ordinateur sur lequel on travaille. Le jour de l'agreg, SAUVER toutes les commandes que vous voulez montrer au jury dans un fichier...

3. Notions de base

3.1. **Créations et Manipulations de variables sous Scilab.**

`a=4;b=%pi` La valeur 4 est affectée à a et la valeur π à b.
Le point virgule a pour effet que matlab n'affiche pas le résultat. Remarque : %pi et %i contiennent respectivement la constante π et le nombre imaginaire i. (⚠ce n'était pas les mêmes commandes avec matlab)

a	
`A=[1 2 3;4 5 6; 7 8 9]`	Création « à la main » de la matrice A.
`C=A'`	C est la matrice adjointe de A

Les colonnes sont séparées par des espaces (ou des virgules), les lignes par des point virgule, ou par des retours à la ligne.
`A(2,3)` désigne l'élément de la deuxième ligne et de la troisième colonne de A.
2. A votre avis que font les commandes suivantes ? Vérifier.
```
[A,[0 0 0]']
[A;[0,0,0]]
```

3.2. **Utilisation des deux points (:)**

`X=[1:20]`	X est le vecteur ligne contenant les entiers de 1 à 20.
`X=[X,21]`	rajoute un élément au vecteur X.
`Y=[1:2:20]`	Y est le vecteur ligne contenant les entiers $1, 3, \cdots, 19$.
`Z=linspace(1,5,50)`	Z est un vecteur ligne contenant 50 valeurs régulièrement réparties entre 1 et 5.

3. A votre avis que font les commandes suivantes ? Vérifier.
```
2+1:5
2+(1:5)
[2:1/10:%pi]
```

`b=A(2,:)`	b est le vecteur ligne contenant la deuxième ligne de A.
`c=A(:,3)`	c est le vecteur colonne contenant la troisième colonne de A.
`B=A(1:2,1:2)`	B est la première sous-matrice carrée d'ordre 2 de A .

4. Obtenir la matrice contenant les deux premières colonnes de A.

3.3. **Opérations élémentaires sur les matrices et les vecteurs, opérateurs logiques.**

`length(A)`	donne le nombre d'éléments de A (surtout utile si A est un vecteur)
`size(A)`	donne la taille de A
`A+C,A*C,2*A`	Addition et multiplications matricielles classiques
`A+1`	ajoute 1 à tous les éléments de A.
`n=4;I4=eye(n,n)`	Création de la matrice identité d'ordre 4
`B=zeros(n,n)`	B est la matrice nulle 4×4 (⚠Sous matlab, on pouvait utiliser zeros(n))
`D=ones(3,2)`	D est la matrice 3×2 ne contenant que des 1 .
`p=2; A^p`	Calcul de A^2
`det(A),trace(A),spec(A)`	Déterminant, trace et valeurs propres de A (⚠Sous matlab, c'était eig(A))
`sum(X)`	Somme des éléments d'un vecteur
`cumsum(X)`	Vecteur $[X_1, X_1 + X_2, \ldots, X_1 + \ldots X_n]$ (sommes cumulées) TRES UTILE en proba-stat !

Effectuez les questions ci-dessous sans faire de boucle :
5. Obtenir un vecteur colonne de taille 10 dont tous les éléments valent 5.
6. Calculer le produit scalaire des deux premiers vecteurs colonnes de A (penser à utiliser la transposée)
7. Calculer la somme des 50 premiers entiers pairs
La plupart des fonctions scilab peuvent s'appliquer à des matrices ou à des vecteurs, et s'effectuent élément par élément : c'est le cas par exemple de `cos`, `sin`, `sqrt` et `exp`. Attention : si on veut calculer l'exponentielle d'une matrice il faut utiliser `expm`.
Cela marche aussi avec les opérateurs logiques : *essayer* `(A<3)|(A>7)`*, qui renvoie une matrice ne contenant que des F et des T.*
(⚠Sous matlab on obtenait directement des 0 et des 1) Pour transformer les "F" et "T" en 0 et 1, la méthode la plus rapide consiste en fait... à multiplier par 1. Essayer `1*(A>2)`.
Remarque : `sum` ajoute directement les "F" et les "T" comme si c'étaient des 0 et des 1.
8. Construire un vecteur contenant les racines carrées des entiers pairs de 2 à 10.
9. Construire un vecteur contenant les cosinus des entiers de 1 à 100. Utiliser `sum` *et un opérateur logique pour savoir combien d'entiers de 1 à 100 ont un cosinus supérieur à 0.5.*

3.4. **Utilisation du point (.)** L'opérateur point (`.`) permet de transformer une commande matricielle (typiquement, une opération "puissance") en une commande élément par élément. *Comparer par exemple* `A^2` *et* `A.^2`*,* `A*C` *et* `A.*C`
Remarque : si X est un vecteur, `X^2` effectue directement la multiplication coordonnée par coordonnée. (⚠ce n'était pas le cas avec matlab)
ATTENTION : un entier suivi d'un point est compris comme une valeur réelle. Donc si on veut que le point soit compris comme ayant un lien avec l'opération qui suit, il faut un espace ! ⚠Ce n'était pas le cas avec matlab ! EXEMPLE : *comparer* `1./A` *et* `1 ./A`.
10. Construire (sans faire de boucle) un vecteur contenant les carrés des entiers de 1 à 10.
11. Construire (sans faire de boucle) un vecteur contenant les inverses des entiers de 1 à 10.
12. Construire (sans faire de boucle) un vecteur contenant les sommes harmoniques $1, 1+\frac{1}{2}, \ldots, 1+\frac{1}{2}+\ldots\frac{1}{10}$. *(penser à* `cumsum`*)*

3.5. **Premières commandes statistiques.**

`rand`	Générateur aléatoire associé à la loi uniforme $\mathcal{U}([0,1])$.
`X=rand(1,1000);`	X est un vecteur ligne contenant $n = 1000$ réalisations i.i.d. $\mathcal{U}([0,1])$
`D=rand(2,3);`	D est une matrice dont chaque élément est aléatoire uniforme sur $[0,1]$
`m=mean(X)`	Calcule la moyenne empirique (i.e. arithmétique) de X
`v=variance(X)`	Calcule la variance empirique de X (avec une division par $n-1$)
`sigma=std(X)`	Calcule l'écart-type empirique de X (avec une division par $n-1$)

Un vecteur comportant n réalisations indépendantes d'une même loi est aussi appelé n-échantillon.
13. Générer un 2000-échantillon X de loi uniforme sur $[0,1]$. Comparer sa moyenne empirique avec $E[X]$.
14. Comparer la moyenne empirique de X^3 avec $E[X^3]$
15. Générer un 20-échantillon de loi uniforme sur $[0,1]$ et déterminer le nombre N d'éléments de votre échantillon qui sont inférieurs à 0.3. N est donc aléatoire. Quelle est sa loi ?

`grand(n,m,'nom',paramètres)` est LA commande qui permet de simuler les lois usuelles.
⚠Cette commande n'existait pas sous matlab; elle permet d'éviter le recours au module stixbox dans de nombreux cas

16. Faire help grand. Trouver comment simuler une loi normale, une loi exponentielle, une loi de Poisson.
17. Générer un 100 échantillon d'une loi normale centrée de variance 4. Obtenir sa moyenne empirique et sa variance empirique **ATTENTION** Le dernier paramètre de la loi normale pour scilab est l'écart-type, pas la variance !
18. Générer un 100 échantillon d'une loi de Poisson de paramètre 2. Obtenir sa moyenne empirique et sa variance empirique.

3.6. **Boucles.** Toutes ces commandes permettent bien souvent de ne pas avoir à faire de boucle. Néanmoins ce n'est pas toujours possible... La syntaxe pour une boucle est par exemple

```
 for k=1:10
INSTRUCTIONS
end
```

Il est assez pénible de taper des boucles directement dans la console. La manière la plus pratique dès qu'on commence à avoir des suites d'instructions à taper est d'ouvrir un fichier dans l'éditeur avec ces instructions (le fichier sera un "script" ".sce"). On peut mettre des commentaires (c'est très utile quand on doit montrer son programme par exemple à un jury) en commençant une ligne par // ⚠(avec matlab c'était %)
19. Créer un script blabla.sce qui construise la matrice 3×3 dont l'élément (i,j) est $\frac{1}{i+j-1}$ (utiliser deux boucles), calcule son déterminant, et son inverse.
Pour exécuter le script, il suffit de sélectionner F5 dans l'éditeur, ou de sélectionner "exécuter" dans l'un des menus ou dans la barre des taches.
Par défaut, exécuter fonctionne comme s'il y avait des point virgule partout. (Donc on ne voit rien). Pour changer cela, sélectionner "exécuter avec echo", ou utiliser `disp` pour faire afficher les variables.

3.7. **Représentations graphiques.** Si `x` et `y` sont deux vecteurs de même taille, la commande `plot2d(x,y)` fournit le graphe formé à partir des points $[x_i, y_i]$. On peut mettre ensuite des options : en particulier, les entiers correspondent à des couleurs, -1 affiche des croix(⚠la commande

matlab était plot; elle marche toujours, et les options des couleurs sont par exemple 'r' pour rouge) Ainsi, par exemple :

`t=linspace(-2,2,50); plot2d(t,exp(t)+3,5)`	graphe de la fonction $t \mapsto \exp(t)+3$ en rouge.
`xtitle(''graphe'', ''temps'', ''fn'')`	Titre et légendes des axes

Si on fait plusieurs "plot2d" à la suite, ils seront sur la même figure, à moins de l'effacer avant avec `clf`.(⚠Le hold on est automatique avec scilab!) `figure`, ou encore mieux `scf(2)` ouvre une nouvelle fenêtre pour une nouvelle figure (si la fenêtre 2 n'existe pas; sinon scf(2) utilise la fenêtre 2 pour la figure)

20. Obtenir sur le même graphe les courbes représentatives de cos, et $x \mapsto 1-\frac{x^2}{2}$. sur $[-\pi, \pi]$. Essayez d'avoir des couleurs différentes. Rajouter des légendes aux axes, Obtenir encore sur le même graphe une droite horizontale d'équation $y = 0.5$ et une droite verticale $x = \frac{\pi}{2}$.

3.8. **Histogrammes.** Vous avez sûrement déjà vu un histogramme. L'histogramme fourni par scilab est par défaut normalisé, c'est-à-dire que les hauteurs des colonnes sont calculées de telle sorte qu'il soit d'aire 1. (Cela permet de mieux le comparer à une densité de probabilités)

Si X est un vecteur et m un entier `histplot(m,X)` trace un histogramme normalisé de X avec m classes. En général on prend de l'ordre de $\sqrt{n}$ classes, où n est le nombre de réalisations. (⚠Pas la peine d'utiliser histo comme avec matlab, donc, et l'histogramme est normalisé par défaut!)

Remarque : pour changer la couleur de l'histogramme, on peut utiliser par exemple `histplot(m,X,style=3)`

21. Construire plusieurs échantillons d'une loi normale centrée réduite en faisant varier le nombre de réalisations n entre 100 *à* 100 000 *. Tracer à chaque fois l'histogramme de X, en faisant éventuellement varier le nombre de classes de l'histogramme.*

18. Grâce au module stixbox, la fonction `dnorm` *permet d'obtenir la fonction densité de la loi normale.*

Obtenir sur la même figure que précédemment le graphe de la densité de la loi normale.

On voit de plus que la forme de l'histogramme (et le fait qu'on reconnaît ou non la densité sousjacente) peut dépendre pas mal des choix du nombre de classes; c'est pourquoi on travaillera plus tard avec la "fonction de répartition empirique", qui a de bonnes propriétés.

3.9. **Les fonctions.** Une fonction scilab est une procédure qui peut prendre en entrée des arguments et peut ressortir un ou des résultats en sortie. Une fonction s'écrit soit dans un script, soit dans un fichier .sci à part. Un même fichier peut comporter plusieurs fonctions, et peut s'appeler comme on le souhaite... (⚠Ce n'était pas le cas avec matlab!) On peut par exemple mettre les commandes suivantes dans un fichier "truc.sci" :

```
function y=toto(t);
// Ligne de commentaire éventuelle
y=t^2-3;
endfunction
```

Il faut ensuite exécuter le fichier qui contient la fonction, par exemple avec F5 (ou avec le menu Exécuter) ⚠⚠Cette étape n'était pas nécessaire avec matlab Même si on n'a pas mis de point virgule dans la fonction, scilab fait comme s'il y en avait. Si on veut forcer la fonction à afficher une valeur pendant son exécution, il faut utiliser `disp nomdelavariable`. (⚠ce n'était pas le cas sous matlab). Ici `toto(2)` renvoie la valeur 1. L'argument de toto peut aussi être un vecteur : essayer `toto([1:10])`

Une fonction peut aussi renvoyer plusieurs variables. Pour une fonction qui doit renvoyer deux réels, on a deux solutions :

- Soit on la déclare sous la forme `function [y1,y2]=bidule(x)`. Dans ce cas là, si on effectue `u=bidule(1)` u recevra uniquement la valeur y1. Pour obtenir les deux valeurs il faut effectuer [u,v]=bidule(1).
- On peut aussi déclarer `function y=bidule(x)`, mais en faisant en sorte (dans la fonction) que y soit bien égal au vecteur [y1,y2]. Dans ce cas la fonction ne renvoie qu'une variable, qui est un vecteur. Si on effectue u=bidule(1), u reçoit le vecteur [y1,y2].

23. Créer une fonction dans un fichier qui prend en entrée un réel x et un réel l et calcule $l\exp(-l*x)$.
24. Comparer sur le même graphique la courbe de la densité de la loi exponentielle de paramètre 2 et un histogramme d'un échantillon de 1000 réalisations de cette loi.