TP Statistique n°1

L'objectif de ce TP est de vous familiariser avec le logiciel Matlab, contraction de Matrix Laboratory. Matlab n'est pas à la base un langage de calcul formel comme Mathematica ou Maple. Il s'agit d'un interpréteur de commandes écrites en langage Matlab. Ce logiciel a été conçu pour faciliter les opérations sur les vecteurs, matrices et tableaux. Le logiciel libre "octave" devrait fonctionner de manière identique, et normalement les fichiers pour l'un devraient être compatibles avec l'autre. Pour les probabilités, on utilisera les fonctions du package gratuit "stixbox".
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/stat.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

Cela peut être une bonne idée de commencer par créer un répertoire où seront stockés tous vos fichiers matlab, et de travailler dans ce répertoire.
Lancer matlab : >matlab &

Pour sauver : "Save workspace as" dans le menu "File" ne sauve que les valeurs des variables. L'historique des commandes dans la fenêtre "Command history" est de toute manière automatiquement sauvé par matlab, et sera visible lors de la session suivante.
Pour sauver les commandes dans un fichier, on peut sélectionner des entrées dans cette fenêtre (on peut par exemple sélectionner tout l'historique d'une session) faire un clic droit et sauver dans un fichier ".m" à l'aide de "Create m-file".

3. Notions de base

Product help dans Menu help : Ouverture de la fenêtre d'aide Matlab.
Si on veut de l'aide sur une commande on peut la chercher ici, ou taper directement " `help` nom de la commande" dans la fenêtre de commande ("command window").

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

`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 : par défaut, `i` et `j` sont deux variables qui contiennent le nombre imaginaire i. Mais vous pouvez les utiliser quand même et changer leur valeur...

`who`	Liste les variables Matlab utilisées dans la session.
`clear b`	Destruction de la variable b
`who`	Vérification que la variable b n'existe plus.
`sqrt(a)`	Calcule la racine carrée de la variable a
`A=[1 2 3;4 5 6; 7 8 9]`	Création « à la main » de la matrice A.

Les colonnes sont séparées par des espaces (ou des virgules), les lignes par des point virgule.
`A(2,3)` désigne l'élément de la deuxième ligne et de la troisième colonne de A.

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

`X=[1:20]`	X est le vecteur ligne contenant les entiers de 1 à 20.
`Y=[1:2:20]`	Y est le vecteur ligne contenant les entiers $1, 3, \cdots, 19$.

1. Essayer [1:-0.5:-2].
2. A votre avis que font les deux commandes suivantes ? Vérifier.

```
1+1:5
1+(1:5)
```

`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 .

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

3.3. **Opérations sur les matrices (faire help elmat).**

`length(A)`	donne la taille de A
`C=A'`	C est la matrice transposée de A
`A+C,A*C`	Addition et multiplication matricielles classiques
`n=4;I4=eye(n)`	Création de la matrice identité d'ordre 4
`B=zeros(n)`	B est la matrice nulle d'ordre 4
`D=ones(n)`	D est la matrice carrée d'ordre 4 ne contenant que des 1 .
`p=2; A^p`	Calcul de A^2
`det(A) ,trace(A) ,eig(A)`	Déterminant, trace et valeurs propres de A
`ans`	La variable ans contient la dernière réponse Matlab non affectée.

4. Obtenir une matrice 3×3 *dont tous les éléments valent 5.*
5. Calculer le produit scalaire des deux premiers vecteurs colonnes de A (penser à utiliser la transposée)

La plupart des fonctions matlab 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>2`*, qui renvoie une matrice ne contenant que des 0 et des 1.*
6. Construire un vecteur contenant les racines carrées des entiers pairs de 2 à 10.

3.4. **Utilisation du point (.)** L'opérateur point (.) permet de transformer une commande matricielle en une commande élément par élément. *Comparer par exemple* `A^2` *et* `A.^2`*,* `A*C` *et* `A.*C`
7. Construire (sans faire de boucle) un vecteur contenant les carrés des entiers de 1 à 10.
8. Construire (sans faire de boucle) un vecteur contenant les inverses des entiers de 1 à 10.

3.5. **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
```

9. Construire à l'aide de deux boucles la matrice 3×3 *dont l'élément* (i, j) *est* $\frac{1}{i+j-1}$*. Calculer son déterminant, et son inverse.*

Vous vous en êtes peut-être rendu compte, il est assez pénible de taper des boucles directement dans la fenêtre de commande. La manière la plus pratique dès qu'on commence à avoir des suites d'instructions à taper est de créer un fichier ".m" avec ces instructions (un "script"); on peut ouvrir un tel fichier dans le menu "file", qui ouvrira l'éditeur de matlab.
10. Mettre les instructions de la question précédente dans un fichier matTP1.m (vous pouvez l'appeler autrement). Si ensuite on tape matTP1 dans la fenêtre de commande, les instructions de ce fichier sont exécutées.

3.6. **Représentations graphiques.** Si `x` et `y` sont deux vecteurs de même taille, la commande `plot(x,y)` fournit le graphe formé à partir des points $[x_i, y_i]$. Ainsi :

`t=0:1/10:3; plot(t,exp(t)+3))`	graphe de la fonction $t \mapsto \exp(t) + 3$.
`t=0:1/10:3; plot(t,exp(t),t,1+t)`	graphes de deux fonctions.

Si on souhaite rajouter une (ou des) courbe(s) sur la même figure, on utilise
`hold on` avant la commande `plot` suivante.

`clf`	efface le graphe.
`hold off`	annule la commande `hold on`
`figure`	ouvre une nouvelle fenêtre pour une nouvelle figure

11. Obtenir sur le même graphe les courbes représentatives de cos, sin et $x \mapsto x^2$*.*
Si on dispose d'une fonction, on peut aussi obtenir son graphe à l'aide de `fplot`.
`fplot('cos',[-pi pi])` graphe de la fonction cos.
Mais pour cela, il faut disposer de fonctions...

3.7. **Les fonctions.** Un certain nombre de commandes de matlab nécessitent d'avoir défini des fonctions : c'est le cas de `fplot`, mais aussi de `quad` qui calcule numériquement des intégrales, de `fzero` et `fsolve` qui cherche numériquement des zéros de fonctions.
Il y a deux manières de définir une fonction.
Première manière : si elle va resservir, il est conseiller de la créer dans un fichier .m, qui devra porter le nom de la fonction. On peut par exemple mettre les commandes suivantes dans un fichier "toto.m" :

```
function resultat=toto(t);
% Ligne de commentaire éventuelle
resultat=t.^2-3;
% Attention si on ne met pas le ; la fonction affichera deux fois le résultat.
end
```

`toto(2)` renvoie alors la valeur 1. (le `end` indiquant la fin de la fonction n'est pas nécessaire.)
ATTENTION : toujours faire en sorte que la fonction puisse s'appliquer à un vecteur (d'où le "point" pour le carré).
Une fonction peut aussi renvoyer plusieurs valeurs sous forme d'un vecteur. Dans ce cas la syntaxe serait par exemple
`function [f1,f2]=toto2(x,y)` pour une fonction de $\mathbb{R}^2$ dans $\mathbb{R}^2$.

`fplot(@toto,[-1 1])`	dessine le graphe de la fonction toto.
ou `fplot('toto',[-1 1])`	
`quad(@toto,0,1)`	calcule une valeur approchée de $\int_0^1 (t^2 - 3)dt$
`fzero(@toto,1)`	donne une valeur approchée d'une racine de $X^2 - 3$

12. Créer une fonction dans un fichier, puis obtenir le graphe de $x \mapsto \frac{2}{1+x^2} - 1$ *à l'aide de la commande* **`fplot`**. *Obtenir une valeur approchée de son intégrale sur* $[0,1]$ *à l'aide de* **`quad`**, *comparer à la vraie valeur.*

Deuxième manière : si on ne souhaite pas créer de fichier, on peut aussi définir une fonction directement à l'aide de `@`. Ainsi

`toto2=@(x) cos(x).^2 - sin(x).^2;`	définit une fonction toto2
`quad(toto2,0,pi)`	calcule $\int_0^\pi (\cos^2(x) - \sin^2(x))dx$
`fzero(@(x) x.^3-3,1)`	résout $x^3 = 3$ "en partant de 1"

13. Obtenir une valeur approchée d'une solution de $\exp(x) = 3x$

4. Premiers pas en statistiques

`rand`	Générateur aléatoire associé à la loi uniforme $\mathcal{U}([0,1])$.
`randn`	Générateur aléatoire associé à la loi normale $\mathcal{N}(0,1)$
`X=rand(1000,1);`	X est un vecteur colonne contenant $n = 1000$ réalisations i.i.d. $\mathcal{U}([0,1])$
`s=sum(X)`	Calcule la somme des composantes de X .
`m=mean(X)`	Calcule la moyenne empirique de X
`v=var(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 $n-1$)

14. Comparer la moyenne empirique de X^3 *avec sa valeur théorique.*

4.1. **Représentations graphiques : Histogrammes.** Vous avez sûrement déjà vu un histogramme. Un histogramme peut être normalisé ou non, selon que les hauteurs des colonnes sont calculées de telle sorte qu'il soit d'aire 1, ou que les hauteurs sont les effectifs de chaque classe.
Pour comparer un histogramme à une densité de probabilités, il vaut mieux qu'il soit normalisé...
Si X est un vecteur, `histo(X)` trace un histogramme non normalisé de X. Pour le normaliser, il faut donner 4 paramètres, par exemple `histo(X,10,0,1)`. Le deuxième paramètre est le nombre de classes (ou de colonnes, en anglais "bins", de l'histogramme), le troisième paramètre vaut 0 ou 1 (en général 0 pour une loi continue, 1 pour une discrète mais ce n'est pas crucial), et le quatrième vaut 1 pour indiquer que l'histogramme est normalisé.
15. Construire plusieurs fois X *en faisant varier le nombre de réalisations* n *de* 100 *à* 100 000 . *Tracer à chaque fois l'histogramme de* X, *en faisant éventuellement varier le nombre de classes de l'histogramme.*
Remarque : il n'est pas évident en règle générale de savoir combien de classes il est pertinent de prendre; en général on prend de l'ordre de $\sqrt{n}$ où n est le nombre de réalisations. 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 de ce choix; c'est pourquoi on travaillera plus tard avec la "fonction de répartition empirique", qui a de bonnes propriétés.
16. Générer un échantillon de 1000 réalisations d'une loi normale à l'aide de **`randn`**. *Obtenir son histogramme, et tracer sa densité sur le même graphe. Comparer sa moyenne empirique et son espérance, ainsi que son écart-type empirique et théorique*