TP1 – Master Finance – logiciels
Introduction à R

Emeline Schmisser, `emeline.schmisser@math.univ-lille1.fr`, bureau 314 (bâtiment M3).

1 Séquences, Vecteurs, Matrice Tableaux (arrays)

Pour obtenir l'aide de `R` sur une commande, il faut taper `help("nom de la commande")`. L'aide est en anglais.
Pour rentrer un vecteur, on peut utiliser les commandes `c`, `seq` ou `rep`. Pour rentrer une matrice, on utiliser la commande `matrix(nomvecteur, nombre lignes, nombre colonnes, sens de remplissage)`.
Le logiciel `R`, comme les logiciels `Scilab` et `Matlab`, est capable de faire des opérations sur les vecteurs et les matrices. Il faut cependant faire très attention :
La commande `A*B` n'est pas une multiplication matricielle, on se contente de multiplier les coefficients entre eux. De même, quand on tape `A^2`, on ne calcule pas le carré de la matrice A, on met juste les coefficients de la matrice A au carré.
Pour faire une vraie multiplication matricielle, il faut taper `A%*%B`. La commande `solve` permet d'inverser une matrice.
La fonction `outer(u,v,'f')` permet à partir de deux vecteur u de longueur m et v de longueur n de créer la matrice

$$\begin{pmatrix} f(u(1),v(1)) & f(u(1),v(2)) & \dots & f(u(1),v(n)) \\ f(u(2),v(1)) & f(u(2),v(2)) & \dots & f(u(2),v(n)) \\ \dots & \dots & \dots & \dots \\ f(u(m),v(1)) & f(u(m),v(2)) & \dots & f(u(m),v(n)) \end{pmatrix}$$

Exercice 1 : Manipulations simples

1. Manipulation de vecteurs.
   (a) Saisir le vecteur a en tapant `a <- c(10, 5, 3, 6, 6)`. Taper a. Puis $a[2]$, $a[1,3]$ et $a[c(1,3)]$.
   (b) Générer un vecteur b de dimension 5 dont toutes les composantes sont 1 en utilisant la fonction `rep`.
   (c) Générer un vecteur c en tapant `c<-seq(from=1, to=10, by=2)`. Qu'obtient-on si on tape `c<-seq(1,10,2)` ? `d<-seq(1,10)` ? `e<-1:5` ?
   (d) Taper `2*a+b+1`. Qu'obtient-on ?
   (e) Taper `cos(a)`, `exp(a)`. Qu'obtient-on ?
   (f) Taper `a*e` puis `a%*%e`.Comparer.
   (g) Taper maintenant `a%*%t(e)`. Que se passe-t-il ? Que fait la commande `t` ?

(h) Taper `a<5` puis `(a<5)*a`. Qu'obtient-on ?

(i) Taper `c(a,b)`. Qu'obtient-on ?

(j) Que donne la commande `table(a)` ?

(k) Que donne la commande `length(a)` ?

2. Manipulation de matrices

(a) Générer une matrice diagonale `D` de dimension 3 dont les éléments de la diagonale sont 1, 5 et 9 à l'aide de la commande `diag`. Demander de l'aide sur la fonction `diag` grâce à la commande `help`.

(b) Créer la matrice

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 3 & 1 \end{pmatrix}$$

grâce à la commande `matrix`. Attention : on peut soit construire une matrice ligne par ligne, soit colonne par colonne. `R` permet de faire les deux, mais il faut lui préciser le sens de remplissages.

(c) Créer la matrice

$$B = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 \end{pmatrix}$$

grâce à la commande `matrix`. On n'est pas obligé de taper toutes les valeurs.

(d) Demander `dim(A)`. Taper `A[2,3]`, `A[,3]`, `A[2:3,]` puis `A[2:3,4]`. Qu'obtient-on ?

(e) Taper `cbind(A,D)` puis `rbind(A,D)`.

(f) Taper `t(A)`. Que fait la commande `t` ?

(g) Taper `cos(A)`, `exp(A)`. Qu'obtient-on ?

(h) Taper `A*D` puis `A%*%D`. Quelle est la différence ?

(i) Que donne la commande `dim(A)` ?

3. Manipulation de tableaux.

(a) Créer le tableau

$$C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 3 & 1 \end{pmatrix}$$

grâce à la commande `array`.

(b) Regarder l'aide sur la base de données `Titanic`. Quelles sont les dimensions de ce tableau ?

4. Manipulation de listes.

(a) Taper

```
Lst <- list(name="Fred", wife="Mary", no.children=3,child.ages=c(9,7,4)).
```

Demander `Lst`.

(b) Demander `Lst[[1]]`. Retrouver le résultat en tapant `Lst$name`

(c) Demander `Lst[[4]]`. Qu'obtient-on. Demander l'âge du troisième enfant.

Exercice 2 : Résolution d'un système linéaire et inversion d'une matrice carrée
Soient :

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 3 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}.$$

1. Saisir la matrice A (par colonnes) et le vecteur b. La matrice A est-elle inversible ? Calculer son déterminant grace à la commande `det(A)`.
2. Regarder l'aide de la commande `solve` et résoudre $Ax = b$.
3. Utiliser `solve` pour calculer A^{-1}.

Exercice 3 : Manipulation de outer
1. Créer la table d'addition.
2. Créer la table de multiplication.

2 Lois de probabilités

R connaît la plupart des lois de probabilités usuelles. On y accède en chargeand le package stats (`library(stats)`). Il peut :
- Simuler une variable suivant cette loi grâce à la commande `rloi`
- Donner la densité (pour une variable à densité) ou les probabilités marginales (pour une variable discrète) : `dloi`
- Donner la fonction de répartition : `ploi`
- Donner la fonction quantile : `qloi`.

Loi usuelles :
- Loi normale : `norm`
- loi exponentielle : `exp`
- loi uniforme : `unif`
- loi binomiale : `binom`
- loi de Poisson : `pois`

Par exemple, la fonction `dunif(0.3,0,2)` calcule la densité de la loi uniforme sur l'intervalle $[0, 2]$ au point 0.3. Il faut penser à regarder l'aide pour chacune de ces fonctions, car les paramètres dépendent de la loi. Le logiciel `R` a de plus beaucoup de fonctions qui permettent d'étudier les données : par exemple `mean`, `var` (qui calcule la variance corrigée, ce qui est légèrement différent de la variance empirique), `sd` (écart-type), `cov`, `cor`, `median`, `mad` :

Exercice 4 : Probabilités
1. Quelle est la probabilité qu'une loi de Poisson de paramètre 2 vaille 3 ?
2. Calculez la loi de probabilité d'une loi binomiale $\mathcal{B}(25, 0.25)$. Pour cela, il faut calculer les probabilités marginales pour toutes les valeurs que peut prendre une variable de loi binomiale $\mathcal{B}(25, 0.5)$.
3. Quelle est la probabilité qu'une loi binomiale de paramètres $(10, 0.7)$ soit plus grande que 8 ?
4. Quelle est la probabilité qu'une loi exponentielle de paramètre 1 soit plus petite que 5 ?
5. Quelle est la probabilité qu'une loi normale $\mathcal{N}(0, 1)$ soit plus petite que 3 en valeur absolue ?

6. Quel est le 95ème centile d'une loi exponentielle de paramètre 0.5 ?
7. Quel est le 1er quartile d'une loi binomiale de paramètres $(20, 0.2)$?
8. Simuler 100 variables de loi $\mathcal{B}(10, 0.5)$ et donner la table de contingence grâce à la commande `table`. Calculez ensuite la table de probabilité pour cette loi. Est-ce que les valeurs observées sont proches des valeurs théoriques.
9. Simuler 100 variables de loi $\mathcal{B}(1000, 0.001)$ et donner la table de contingence. Comparer ensuite les valeurs observées avec la table de probabilité d'une loi de Poisson de paramètre approprié.
10. Simuler 1000 variables de loi normale $\mathcal{N}(0, 2)$. Calculer la moyenne et la variance empirique (corrigé de ces variables. Est-ce que les résultats obtenus vous paraissent logiques ?

3 Graphiques

Il existe différents type de graphiques sous `R` : les graphes classiques, les histogrammes, les boîtes à moustaches . . . .

Pour les graphes les plus simples, on utilise les trois commandes `plot(x,y,options)`, `points(x,y,options)` et `lines(x,y,options)`. Ces trois commandes permettent de tracer y en fonction de x.
On utilise toujours `plot` pour commencer un graphique. Les options les plus utilisées sont :

- type ='p' par défaut (ne trace que des points), ='l' si on veut tracer une ligne, ='s' pour une fonction en escalier.
- col ='black' par défaut (couleurs possibles : red, blue, green, yellow, orange, magenta, pink, ...)
- pch (c'est le style de point utilisé, il peut être égal à 1,2,3,...)
- lty (c'est le style de ligne utilisé, il peut être égal à 1,2,3,...)
- ylim=c(a,b) (option très utile si on veut superposer plusieurs graphiques. Par défaut, `R` affiche les y entre min(y) et max(y), on peut forcer l'affichage d'autres valeurs). Ne fonctionne qu'avec la commande `plot`.
- xlim=c(a,b) (même chose que ylim, mais pour x. Peut-être moins utile).

Pour superposer un deuxième graphique, on utilise soit `points` (pour tracer des points) soit `lines` (pour tracer des lignes).

La fonction `hist` trace l'histogramme d'un vecteur. La commande `hist(x,breaks,freq=TRUE)` donne le nombre de variables dans chaque intervalle, et la commande `hist(x,breaks,freq=FALSE)` donne la proportion de variables dans chaque intervalle.

Attention : sauf dans le cas d'une série temporelle, il faut toujours préciser les deux vecteurs !.

La fonction `boxplot` trace la boîte à moustache d'un vecteur. Elle permet de tracer soit la boîte à moustache d'un seul vecteur, soit la boîte à moustache de plusieurs vecteurs côte à côte. La fonction `qqplot` permet de tracer le graphe quantile/quantile de deux vecteurs (X, Y). Les deux vecteurs n'ont pas besoin d'être de même longueur. Avec `qqline`, on peut ajouter la droite qui passe par les quartiles $Q1$ et $Q3$. La fonction `qqnorm` trace les quantiles empiriques d'un vecteur X par rapport aux quantiles théoriques de la loi normale $\mathcal{N}(0, 1)$.

Exercice 5.

1. Loi normale et densité.
   (a) Tracer la densité d'une loi normale centrée réduite sur $[-5, 5]$. Pour cela :
      i. Créer un vecteur x qui varie doucement de -5 à 5.

ii. Créer un vecteur y qui donne la densité de la loi normale centrée réduite pour tous les points de x.

iii. Tracer le graphique. Attention, il faut tracer une courbe, pas des points côte à côte.

(b) Superposer la densité d'une loi normale centrée de variance 0.5. Attention : on veut voir entièrement les deux densités.

(c) Superposer la densité d'une loi normale centrée de variance 2.

2. Loi normale et fonction de répartition.
   (a) Tracer la fonction de répartition d'une loi normale centrée réduite sur $[-5, 5]$.
   (b) Superposer la fonction de répartition d'une loi normale centrée de variance 0.5.
   (c) Superposer la fonction de répartition d'une loi normale centrée de variance 2.
3. Loi normale et histogramme.
   (a) Simuler 100 variables de loi normale $\mathcal{N}(0, 1)$. Tracer l'histogramme des proportions.
   (b) Superposer en rouge la densité théorique.
4. Boîtes à moustaches.
   (a) Simuler 100 variables aléatoires de loi de Poisson de paramètre 2 et les stocker dans un vecteur X.
   (b) Simuler 100 variables aléatoires de loi normale d'espérance 2 et de variance 2 et les stocker dans un vecteur Y.
   (c) Calculer la moyenne et la variance empirique de X et de Y.
   (d) Tracer sur le même graphique les diagrammes en boîtes à moustache pour X et pour Y.
5. Loi binomiale et fonction de répartition empirique.
   (a) Simuler 20 variables aléatoires de loi binomiale $\mathcal{B}(10, 0.2)$. Les stocker dans un vecteur X.
   (b) On veut tracer la fonction de répartition empirique F_n. On a que :

$$F_n(x) = \sum_{i=1}^{n} \mathbf{1}_{X_i \leq x}$$

   On introduit les statistiques d'ordre $X_{(1)} \leq X_{(2)} \leq \ldots \leq X_{(n)}$ (c'est le vecteur X trié). On a alors que

$$F_n(x) = 0\mathbf{1}_{x<X_{(1)}} + \frac{1}{n}\mathbf{1}_{X_{(1)}\leq x<X_{(2)}} + \ldots + \mathbf{1}_{x>X_{(n)}}$$

   Utiliser la fonction `sort` pour trier le vecteur X puis tracer la fonction de répartition empirique.

   (c) Superposer la fonction de répartition théorique.

4 Bases de données et séries temporelles

Le logiciel `R` est capable de gérer des séries temporelles ou des fichiers de données. Il comporte aussi beaucoup de jeux données déjà implémentés. Si on veut utiliser un fichier excel, il faut d'abord le convertir en .csv puis utiliser la fonction `read.table("adresse du fichier")`, `read.csv` (pour les fichiers codés à la manière anglaise) et `read.csv2` (pour les fichiers codés à la manière française). On peut stocker des tableaux dans un fichier grâce à la commande `write.table`.

Exercice 6 : Manipulation d'une base de données

1. Regarder l'aide pour la fonction `read.table`.
2. Ouvrir le fichier `sondages.csv` grâce à la commande `sondages<-read.csv("sondages.csv")`.
3. Afficher les 5 premières lignes de `sondages`.
4. À quoi correspondent les différentes colonnes ?
5. Grâce à la fonction `cbind`, créer une matrice M contenant les sondages pour les 6 premiers candidats.
6. Que fait la fonction `rowSums(M)` ?
7. Tracer une boîte à moustache pour les 6 premiers candidats.

Exercice 7 : Manipulation simple d'une série temporelle

1. Enregistrer la série temporelle `EuStockMarket` dans une variable E. À quoi correspond cette série ?
2. À quoi correspondent les commandes `start(E)`, `end(E)`, `frequency(E)` ?
3. Afficher les dix premières lignes de E. À quoi correspondent les colonnes ?
4. Que fait la commande `summary(E)` ?
5. Que se passe-t-il si on fait `plot(E)` ?
6. Stocker la troisième colonne de E dans une variable CAC.

On voudrait savoir si les variables du CAC suivent une loi normale. Pour cela, on va utiliser 3 méthodes graphiques différentes.

7. Tracer la boîte à moustache de ces données. Commenter.
8. Utiliser la fonction `qqnorm`. Est-ce que les valeurs sont distribuées suivant une droite ? Commenter.
9. Tracer un histogramme de ces valeurs, puis superposer la densité d'une loi normale de même moyenne et même variance. Commenter.

Exercice 8 : Créer une série temporelle

Les températures maximales observées à Lille du 1er au 18 juin 2013 sont : 16°C, 17°C, 15°C, 18°C, 23°C, 26°C, 27°C, 23°C, 19°C, 18°C, 21°C, 23°C, 19°C, 20°C, 19°C, 19°C, 23°C, 21°C.

1. Regarder l'aide de la fonction `ts` et créer la série temporelle des températures à Lille.
2. Visualiser les résultats à l'aide de la fonction `plot`.
3. Est-ce que les données suivent une loi normale ?

5 Un peu de programmation

– Structure d'une boucle for :

```
for (k in 1:n)
{
lignes codes
}
```

Les accolades ne sont pas nécessaires s'il n'y a qu'une seule ligne de code.

- Conditions

```
if (condition)
{
lignes codes
}
else
{
lignes codes
}
```

On n'est pas obligé de mettre un `else`. `R` ne sait comparer pas utiliser la condition `if` sur un vecteur. La structure `if/then/else` s'utilise quand on a qu'une seule variable à tester. Pour faire des opérations conditionnelles sur un vecteur, on peut simplement utiliser les fonctions `<` ou `==`.

- Boucles while

```
while (condition)
{
lignes commandes
}
```

- Fonctions

```
nomfonction <- function (nomvariables)
 {
une ligne de code
}
```

ou si la fonction est plus compliquée :

```
nomfonction<- function (nomvariables)
{
lignes code
nomfonction<- expression
}
```

Dans ce dernier cas, le résultat de la fonction ne s'affichera pas automatiquement.

Exercice 9 : Fonctions

1. construire un vecteur $v = (10, 12, 8, 11, 9)$ et un vecteur $c = (2, 4, 1, 4, 1)$.
2. Créer une fonction `moyenne` qui calcule la moyenne d'un vecteur v. On pourra utiliser les fonctions `sum` et `length`. La tester sur le vecteur v.
3. Créer une fonction `moyenne_pond` qui calcule une moyenne pondérée (les notes sont dans le vecteur v, les coefficients dans le vecteur c. La tester.
4. Créer une fonction `indicatrice` qui prend en entrée un vecteur u et renvoie un autre vecteur v : $v(i) = 1$ si $u(i) \in [0, 1]$ et $v(i) = 0$ sinon. La fonction ne doit pas comporter de boucles.

Exercice 10 : Pourquoi il faut éviter les boucles lorsque c'est possible

La fonction `system.time` nous donne le temps mis pour exécuter une fonction.

1. Créer un vecteur séquence x qui varie de 0 à 1 et de longueur 1000. On va calculer $sin(x)$ de deux manières différentes.
2. Taper

```
 system.time(
for (i in 1:n) {y[i]<-sin(x[i])})
```

3. Comparer avec le résultat de

```
system.time(z<-sin(x))
```

Exercice 11 : Jeu

On modifie un nombre comme suit :
Si n est pair, on le divise par 2, si n est impair, on le multiplie par 3 et on lui ajoute 1. On répète ensuite l'opération.

1. Appliquer ces opérations sur différents nombres : 4, 5, 8 par exemple. Que peut-on remarquer ?
2. Créer une fonction `jeu` qui, étant donné un nombre n, affiche la suite de ses 10 premières transformations. On pourra utiliser la commande `n%%p` qui donne le reste de la division de n par p.
3. Créer une fonction `jeu2` qui, étant donné un nombre n, affiche la suite de ses transformations jusquq'à ce qu'on atteigne le nombre 1.
4. Créer une fonction `nombre` qui, étant donné un nombre n, donne le nombre d'opérations que l'on doit faire avant d'atteindre le nombre 1.
5. Chercher, parmis les cent premiers nombres entiers, celui pour lequel la fonction nombre atteint son maximum. Quel est ce maximum ? (On pourra utiliser les fonctions `max` et `which.max`).