

Université Paris I, Panthéon - Sorbonne

M1 M.A.E.F. 2012 – 2013

TP de Séries Temporelles n^0 1 : Manipulation et simulation d'une série temporelle

L'objectif de ce TP est de vous familiariser avec les objets et les commandes du logiciels R, notamment de tout ce qui a trait aux séries chronologiques. Le logiciel R est un logiciel de mathématiques appliquées, spécialisés en statistiques, qui a la particularité d'être libre (donc gratuit) et qui s'enrichie au fur et à mesure de l'apport gracieux de chercheurs du monde entier. Il est très proche d'un logiciel commercialisé, peut-être plus convivial, Splus. On peut également télécharger ce logiciel à partir des sites <http://www.r-project.org/> ou <http://cran.cict.fr/> pour des systèmes d'exploitation Windows, Mac ou Linux. .

On commence par se connecter à l'ordinateur, puis à ouvrir le logiciel en "cliquant" sur l'icône correspondante. Ensuite, et cela pourra être un réflexe utile dans toute la suite, on "clique" sur *Help*; deux formes d'aide sont alors proposées. En premier lieu, quand on ne connaît pas de commande, on peut "cliquer" sur *Search help* et rechercher des commandes à partir de mots-clés (essayer avec *time series* ou *regression*). Ensuite, quand le nom d'une commande est connue, on peut avoir tous les détails la concernant en "cliquant" sur *R functions*, puis en écrivant le nom de cette commande (essayer avec *aov* ou *monthplot*).

Dans la suite, sont écrites à gauche des commandes à taper, et à droite des commentaires sur ces commandes ou des questions relatives. Vous pouvez donc ainsi apprendre les commandes du logiciel par la pratique.

Une façon intéressante d'utiliser le logiciel est d'ouvrir des "Scripts" dans lesquels vous allez pouvoir écrire l'ensemble des commandes. Il sera alors possible d'exécuter une partie des commandes tapées. Ce sera donc très pratique plutôt que de retaper les commandes à chaque fois. Il faut donc aller dans la colonne *Fichier*, et cliquer sur nouveau *script*. Puis, après avoir taper les commandes choisies, avec la souris vous sélectionner la ou les commandes particulières que vous désirez faire tourner, vous allez ensuite dans la colonne *Edition* et vous cliquer sur *Executer une selection*.

Manipulations

<code>x = c()</code>	Définit x comme un vecteur (c comme concatener).
<code>x = matrix(nrow = n, ncol = d)</code>	Définit x comme une matrice de n ligne, d colonnes.
<code>x = matrix(c(1,2,-1,5), nrow = 2 ncol = 2)</code>	Génère x comme une matrice de 2 ligne, 2 colonnes, colonne par colonne.
<code>y = x * x</code>	Produit de Hadamard.
<code>y</code>	Résultat.
<code>y = x%%*%x</code>	Produit matriciel.
<code>x = c("a", "B")</code>	Génère une chaîne de caractères.
<code>is.character(x)</code>	x est-il une série de caractères ?
<code>x = c(0, 1)</code>	Génère une chaîne de chiffres.
<code>is.numeric(x)</code>	x est-il une série numérique ?
<code>y = as.character(x)</code>	On transforme x en caractère.
<code>x = c(4,3,7,1.7,3,8.4,5,12,9,12)</code>	Génère un vecteur de 10 données prédéfinies.
<code>x[i]</code>	Donne le i ème point du vecteur x (essayer $x = 3$ et $x = 12$).
<code>x[1] + x[2]</code>	résultat ?
<code>x[1 :3]</code>	Résultat ?
<code>x[seq(1, length(x), by = 2)]</code>	que fait la fonction "seq" ? La fonction "length" ?
<code>x[seq(1, length(x), length = 2)]</code>	que fait la fonction "seq" ?
<code>is.ts(x)</code>	x est-il une "time series" (série chronologique) ?
<code>y = as.ts(x)</code>	On transforme x en un objet R time series y .
<code>y</code>	Changements apparents ?
<code>time(y)</code>	Permet de consulter le descriptif de cette série temporelle.
<code>?time</code>	Donne l'aide (en ligne) de la fonction "time".
<code>is.ts(y)</code>	Vérification.
	Plus généralement, le logiciel R fonctionne avec différents types d'objets :
	les vecteurs (commandes <i>is.vector</i> et <i>as.vector</i>) ;
	les matrices (commandes <i>is.matrix</i> et <i>as.matrix</i>) ;
	les tableaux de données (commandes <i>is.data.frame</i> et <i>as.data.frame</i>) ;
	les séries chronologiques (commandes <i>is.ts</i> et <i>as.ts</i>) ;
	les listes (commandes <i>is.list</i> et <i>as.list</i>) ;
	Associe à la série y une série temporelle t (par défaut de 1 en 1, fréquence 1).
<code>t = tsp(y)</code>	Tracé de la série chronologique (relie les données).
<code>plot.ts(y)</code>	Marque les différents points de la série.
<code>plot.ts(y, type = "b")</code>	Divise la série en trimestre (marche aussi en divisant en mois donc
<code>z = ts(y, freq = 4)</code>	<i>freq = 12 ...</i>).
<code>z</code>	Vérification.
<code>time(z)</code>	Temps associé ?
<code>z = ts(x, freq = 4, 1991+1/4, 1993)</code>	On renouvelle la série des temps, avec une autre année de début et de fin.
<code>time(z)</code>	
<code>plot.ts(z)</code>	Nouveau tracé.
<code>frequency(z)</code>	Détermine la fréquence de la série des temps.
<code>length(z)</code>	Longueur de la série z .
<code>m = tapply(z, cycle(z), mean)</code>	Permet de calculer la moyenne par cycle.
<code>m</code>	Vérification. Dans le cas présent, est-ce une évaluation d'un éventuel saisonnier ?

Simulation d'une série chronologique

qnorm(0.95)	Aller dans l'aide et taper le mot-clé "distributions".
pnorm(2)	le préfixe r est utilisé pour la génération de nombre aléatoire, d pour la densité, p la fonction de répartition et q pour le quantile.
	Quantile à 95% pour la gaussienne centrée réduite.
	Fonction de répartition en 2.
	Générer 100 variables gaussiennes indépendantes de loi $\mathcal{N}(-1, 3)$.
	Calculer la moyenne, l'écart-type et tracer un histogramme.
x = rnorm(30, mean = 0, sd = 1)	On génère un bruit blanc de taille 30 de moyenne = 0 et écart type = 1.
par(mfrow = c(2,2))	On découpe la fenêtre graphique en 4.
plot.ts(x)	Représentation du bruit blanc.
hist(x, nclass = 6)	On précise le nombre de classes.
qqnorm(x)	On trace les valeurs de x en fonction des quantiles théoriques d'une loi $\mathcal{N}(0, 1)$.
qqline(x)	Tracé de la droite d'Henry. Conclusion ?
	Recommencer en modifiant la taille de x et conclure.
x = rbinom(30,10,.3)	Génération de variables binomiales. Observer les caractéristiques de x .
	Centrer et normaliser cette série. On note y la nouvelle série.
y = as.ts(x)	Transformer y en série chronologique.
sort(y)	On ordonne les éléments de y . Tracer la série ordonnée.
plot.ts(x,z)	Pour tracer dans l'ordre les z en fonction des x .

Exercice

Simuler une série $X_2 = (X_2(k))_k$ trimestrielle commençant le deuxième trimestre 1980 et finissant le troisième trimestre 2000, telle que $X_2(k) = a(k) + S(k) + \varepsilon_k$ (k étant le numéro du trimestre), où :

- La tendance est $a(k) = 0.02 \times k + 3$.
- Le saisonnier S est une fonction périodique de période 1 an, telle que $S = 3$ au premier trimestre, $S = 5$ au second, $S = -1$ au troisième et $S = -7$ au quatrième.
- ε est un bruit blanc gaussien de variance 1.5.

Quelle est la partie déterministe de X_2 ? Les variables $X_2(k)$ sont-elles indépendantes ?

Estimation de la tendance d'une série temporelle

L'objectif est de mettre en pratique différentes procédures de régression permettant d'estimer la tendance éventuelle d'une série chronologique. On commence par simuler une série chronologique avec tendance (connue) puis on estime cette tendance comme si on ne la connaissait pas...

Simulation d'une série avec tendance

t = 1 :120	Le temps.
a = 10*sqrt(t+200/log(t+2))	La tendance.
eps = 10*rnorm(120)	Le bruit.
X = a + eps	La série chronologique simulée

Régressions polynomiales

On commence par estimer l'éventuelle tendance de la série chronologique par des régressions linéaires de degrés de plus en plus élevé.

 Régression linéaire simple

<code>reg1 = lm(X ~ t)</code>	Régression linéaire de X par t .
<code>plot(reg1)</code>	Pour voir.
<code>names(reg1)</code>	Différents composants associés à <i>res</i> .
<code>reg1\$coef</code>	Coefficients de <i>res</i> .
<code>abline(reg1)</code>	Droite de régression s'ajoutant à la série chronologique.
<code>summary(reg1)</code>	Résumé des résultats de la régression.
<code>segments(t, reg1\$fit, t, X)</code>	Visualisation des résidus.
<code>reg1\$res</code>	Valeurs des résidus.
<code>win.graph()</code>	On crée une nouvelle fenêtre graphique.
<code>par(mfrow = c(2,2))</code>	On divise cette fenêtre graphique en (2, 2).
<code>plot(reg1, las = 1)</code>	Trace 4 graphiques permettant d'analyser le résultat de la régression.

 Régression polynomiale (degré 2)

<code>reg2 = lm(X ~ poly(t,2))</code>	Ajustement polynomial (degré 2).
<code>plot.ts(X)</code>	Graphique de la série chronologique.
<code>lines(t, reg2\$fit)</code>	Grappe du polynôme associé.
<code>summary(reg2)</code>	Analyse détaillée de la régression.
<code>segments(t, reg2\$fit, t, X)</code>	Visualisation des résidus.

 Régression polynomiale (degré 9)

<code>reg9 = lm(X ~ poly(t,9))</code>	On utilise maintenant un ajustement polynomial de degré 9 Refaire les mêmes démarches que précédemment.
---------------------------------------	--

 Choix du degré du polynôme avec les critères AIC et BIC

<code>library(MASS)</code>	On charge le package MASS
<code>long = length(t)</code>	
<code>kmax = 10</code>	Degré maximum du polynôme
<code>Z = matrix(nrow=long, ncol=kmax+1)</code>	On définit une matrice Z
<code>for (i in 1 :long)</code>	On va définir la matrice Z élément par élément
<code>for (j in 1 :(kmax+1))</code>	
<code>Z[i,j] = i ^ (j-1)</code>	
<code>ZZ = as.data.frame(Z)</code>	On transforme Z en objet "data.frame"
<code>x.lm = lm(X~.,data=ZZ)</code>	
<code>x.AIC = stepAIC(x.lm,k=2)</code>	On utilise une commande qui minimise le critère AIC
<code>x.BIC = stepAIC(x.lm,k=log(long),trace=FALSE)</code>	Même chose mais avec le critère BIC
	Commentaires sur les résultats ?

Régressions non-paramétriques pour estimer la tendance

Pour obtenir une meilleure adaptation de la régression aux variations locales de la série, on peut procéder avec des régressions locales (loess et lowess : "lo" comme local, "ss" comme scatterplot smoothing) qui fonctionnent avec une fenêtre (dont on peut directement choisir la longueur dans le cas lowess), "coulissant" successivement le long des temps de la série. Dans chaque fenêtre, on fait une régression polynomiale de degré en général 0, 1 ou 2, par moindres carrés pondérés, sur les données de la série, et on obtient ainsi un point lissé pris en général au milieu de la fenêtre. Les poids de la pondération sont obtenus suivant la règle suivante : les poids sont plus élevés plus on est proche du point à lisser. Dans le cas de la régression lowess, on opère de manière itérative, une nouvelle pondération (d'où le "w" pour weighted), en fonction des écarts aux valeurs initiales.

 Régression loess

```
regl1 = loess(X ~ t)
names(regl1)
ts.plot(X)
lines(t, regl1$fit)
points(t, X)
title("Résultats de loess(X ~ t)")
R2=1-var(regl1$res)/var(X)
```

régression "locale" standard.

Différents résultats de la régression loess.

On trace à nouveau les variations de la série.

Graphique de la régression locale associée.

Représentation des points.

On met une légende au graphique.

Calcul direct du R2. Conclusion ?

 Régression par lissage lowess

```
regl2 = lowess(t, X, f=0.5)
names(regl2)
```

Lissage par lowess de la série X , avec un paramètre de lissage fixé

Différents résultats de la régression lowess.

Noter que lowess n'a pas la syntaxe et les options identiques aux autres régressions.

```
low2 = ts(regl2$y)
ts.plot(X, low2)
regl3 = lowess(t, X, f=0.1)
low3 = ts(regl3$y)
ts.plot(X, low3)
ts.plot(low2, low3)
```

On forme à nouveau une série chronologique.

Tracé de la série et de la série lissée.

Lissage par lowess de la série X , avec un autre paramètre de lissage fixé

On forme la série chronologique.

Tracé de la série X et de la série lissée.

Pour comparer les deux lissages ; expliquer la différence entre les deux lissages.

Quelle est le paramètre que vous choisiriez ?

 Régression par l'estimateur à noyaux

```
ks1=ksmooth(t, X, "normal", bandwidth = 2)
ks1
plot.ts(X)
lines(ks1, col=2)
ks2 = ksmooth(t, X, "normal", bandwidth=10)
lines(ks2, col=3)
library(lokern)
kerada = glkerns(t, X)
plot(kerada)
```

On commence par choisir la taille de fenêtre de l'estimateur.

Pour voir...

A comparer avec la série

Autre choix.

A comparer avec la série et l'autre régression

On installe un nouveau package

Régression par noyau avec choix adaptatif de la taille de fenêtre

Pour voir...

Devoir maison

Installer et charger le package "mda". Télécharger le cours du S&P 500 sur Yahoo Finance en données journalières du 01/01/2010 au 22/02/2012.

```
spy = read.csv("chemin/fichier.csv", dec=".", sep=";", header=TRUE)
```

Importe le "fichier" de données.

utilité de "dec" ? "sep" ? "header" ?

```
Cours = spy[,"Close"]
Cours = rev(Cours)
plot(Cours, type='l')
Time = 1 :length(Cours)
?mars
Fit = mars(time, Cours)
fit = Fit$fitted.values
plot(Cours, type='l')
lines(fit, col=2)
```

Définie "Cours" comme le cours de fermeture du S&P

Inverse l'ordre des données.

Grappe des données.

On définit un vecteur de "temps".

Aide pour le code du modèle MARS.

On "calcule" le modèle MARS.

On récupère les données de la régression.

Grappe du cours du S&P 500 (à la fermeture).

Résultat de la régression via MARS.

```
Fit = mars(time, Cours, nk = ??)
```

Commenter !

Faire varier le paramètre "nk", commenter !