Rapport sur :

Manipulation du Logiciel R

Pour analyser des séries temporelles

Réalisé Par : Rachid EL-FAJJY
Zineb ABOUHAFS
BOUAIDA Latifa
Mohamed nabil NASSAR
Mohamed IBRAHIM

Encadré Par : Mr.El Merouani

Année Universitaire : 2016/2017

Sommaire

Avant-Propos :

Le système R connaît depuis plus d'une décennie une progression remarquable dans ses fonctionnalités, dans la variété de ses domaines d'application ou, plus simplement, dans le nombre de ses utilisateurs. La documentation disponible suit la même tangente : plusieurs maisons d'édition proposent dans leur catalogue des ouvrages — voire des collections complètes— dédiés spécifiquement aux utilisations que l'on fait de R en sciences naturelles, en sciences sociales, en finance, etc. Néanmoins, peu d'ouvrages se concentrent sur l'apprentissage de R en tant que langage de programmation sous-jacent aux fonctionnalités statistiques.

Le but de ce travail est de fournir les éléments de base permettant une prise en main rapide du logiciel R afin de faciliter la compréhension pour certains et aussi bien d'appliquer nos acquis théoriques en pratique chose qui permet de joindre l'utile à l'agréable.

Introduction

R est un système d'analyse statistique et graphique développé par Ross Ihaka et Robert Gentleman. Ce logiciel constitue une alternative au logiciel S-PLUS, même si de nombreuses différences dans la conception existent. Cependant, de nombreux programmes écrits pour S-PLUS sont directement utilisables sous R.

Un point fort de R réside dans le fait que ce logiciel est distribué librement. Son installation peut être mise en œuvre à partir du site internet du Compréhensive R Archive Network qui d'une part met à disposition les exécutables et d'autres part donne des informations relatives à la procédure d'installation.

I- Logiciel R

1- Définition

R est un environnement intégré de manipulation de données, de calcul et de préparation de graphiques. Toutefois, ce n'est pas seulement un « autre » environnement statistique (comme SPSS ou SAS, par exemple), mais aussi un langage de programmation complet et autonome.

Le R est un langage principalement inspiré du S et de Scheme (Abelson et collab., 1996).

Le R est un langage particulièrement puissant pour les applications mathématiques et statistiques (et donc actuarielles) puisque précisément développé dans ce but. Parmi ses caractéristiques particulièrement intéressantes, on note :

→ Langage basé sur la notion de vecteur, ce qui simplifie les calculs mathématiques et réduit considérablement le recours aux structures itératives ;

→ Pas de typage ni de déclaration obligatoire des variables ;

→ Programmes courts, en général quelques lignes de code seulement ;

→ Temps de développement très court.

2- Stratégies de travail

Dans la mesure où R se présente essentiellement sous forme d'une invite de commande, il existe deux grandes stratégies de travail avec cet environnement statistique.

On entre des expressions à la ligne de commande pour les évaluer immédiatement :

Par exemple :

```
> 2 + 3
[1] 5
```

Lors d'une affectation, une expression est évaluée, mais le résultat est stocké dans un objet (variable) et rien n'est affiché à l'écran.

Exemple :

```
> a <- 5
```

N.B : Pour affecter le résultat d'un calcul dans un objet et simultanément afficher ce résultat, il suffit de placer l'affectation entre parenthèses.

Par exemple :

```
> (a <- 5)
[1] 5
```

Le symbole d'affectation est <-, c'est-à-dire les deux caractères < et – placés obligatoirement l'un à la suite de l'autre :

Pour affiche le résultat on écrire :

```
>a
[1] 5
```

3- Création des Objets :

Tout dans le langage R est un objet : les variables contenant des données, les fonctions, les opérateurs, même le symbole représentant le nom d'un objet est lui-même un objet. Les objets possèdent au minimum un mode et une longueur et certains peuvent être dotés d'un ou plusieurs attributs

Le mode d'un objet est obtenu avec la fonction mode :

Exemple :

```
> v <- c(1, 2, 5, 9)
>mode(v)
[1] "numeric"
```

La longueur d'un objet est obtenue avec la fonction length :

Exemple :

```
>length(v)
[1] 4
```

4- Création des Vecteurs :

Nous nous restreignons à ce type de vecteurs pour le moment.

Les fonctions de base pour créer des vecteurs sont :

- → c (concaténation) ;
- → numeric (vecteur de mode numeric) ;
- → logical (vecteur de mode logical) ;
- → character (vecteur de mode character).

Exemple :

```
> (v <- c(a = 1, b = 2, c = 5))
a b c
1 2 5
> v <- c(1, 2, 5)
```

```
>names(v) <- c("a", "b", "c")
> v
a b c
1 2 5
```

N.B : Fonction **name** sert à créer une étiquette des éléments d'un objet.

On peut extraire un élément d'un vecteur par sa position ou par son étiquette :

```
>v[3]
C
5
```

```
> v["c"]
  c
  5
```

5- Manipulation des données

```
### écriture dans le répertoire de travail.
getwd(), setwd(‘‘repertoire’’)
write.table(x, file=’’foo.data’’, ...) option ascii=TRUE.

### Lecture des données dans le répertoire de travail.
read.table(‘‘mesdonnees.txt’’)
### enregistrer des données dans le répertoire de travail.
write (x,file=“donnees.txt”)
```

⇒ Écriture de données

Il est possible de créer des données directement dans R, mais il est aussi possible d'en importer comme d'en exporter

Quelle que soit l'opération choisie, il faut faire attention à la localisation du répertoire de travail. Ainsi, la commande **getwd()** permet de savoir le répertoire où l'on se trouve. S'il s'avère nécessaire d'en changer, **setwd**() peut nous y aider.

Exemple :

```
setwd(home/christinet/R) permet d'accéder au répertoire R du dossier christinet).
```

⇒ Lecture de données

Différentes fonctions permettent de lire des données. Ainsi, **read.table,** scan et **read.fwf** permettent d'accéder à des données stockées dans des fichiers de type ASCII.

D'autres for-mats peuvent être lus simplement, ils font alors appel à des fonctions qui ne sont pas de base.

La fonction **read.table** permet de lire très facilement un fichier de donner.

Par exemple :

```
Messonnees<- read.table(‘‘mesdonnees.txt’’)
```

⇒ Enregistrement des données

La façon la plus simple d'écrire des données créées dans R dans un fichier consiste à utiliser la fonction **write**. La syntaxe par défaut **write(x,file=“donnees.txt”)** permet de stocker l'objet x (un vecteur, une matrice ou un tableau) dans le fichier donnees.txt. Des options permettent d'enregistrer les données correctement, comme l'option ncol.

En ce qui concerne l'enregistrement d'objets de tout type, on peut utiliser la commande **save**.

Quelle que soit l'option retenue, il est préférable d'utiliser le format d'enregistrement ASCII (option ascii=TRUE), car cela permet une meilleure compatibilité entre les différents systèmes d'exploitation.

S'il s'avère que l'on a besoin de charger ces données, on pourra taper ultérieurement **load(nomdufichier).**

6- Les fonctions d'aide :

```
### Pour lancer l'aide :
>help(''fonction'')
>help.start()
>help.search(''word'')

### Charger une librairie :
>library(nnet)

### Exécuter un script :
>source(''file.r'')
```

7- Les packages :

Les package sont des éléments qui permet d'ajuster des nouveaux fonctions au logiciel R.

```
### Installation des packages
library(nome_de_package)

### Pour avoir de l'aide sur le package
library(help=nome_de_package)
```

Exemple des packages :

Package	Description
ctest	tests classiques (Fisher, Pearson, Bartlett, ...)
eda	méthodes décrites dans "Explorary Data Analysis" de Tukey
lqs	régression "résistante" et estimation de variance
methods	définition des méthodes et classes pour les objets R ainsi que des utilitaires pour la programmation
modreg	régression "moderne" (lissage et ajustement local)
mva	analyses multivariées
nls	régression non-linéaire
splines	représentations polynomiales
stepfun	analyse de fonctions de distributions empiriques
tcltk	fonctions pour utiliser les éléments de l'interface graphique Tcl/Tk
tools	utilitaires pour le développement de package et l'administration
ts	analyse de séries temporelles

8- Exemple des fonctions :

```
### Générer deux vecteurs de nombres pseudo-aléatoires issus
### d'une loi normale centrée réduite.
x <- rnorm(50)
y <- rnorm(x)

### Graphique des couples (x, y).
plot(x, y)

### Créer une matrice 3 x 4 à partir des valeurs de v. Remarquer que la matrice est
### remplie par colonne.
( m<- matrix(v, nrow = 3, ncol = 4) )

### Eliminer la quatrième colonne afin d'obtenir une matrice carrée.
( m<- m[,-4] )

### Transposée et inverse de la matrice m.
t(m)
solve(m)
```

II- La série temporelle :

1- Définition

Une série temporelle est une collection de données obtenues de manière séquentielle au cours du temps.

→ Modèle additif : $y_t = T_t + S_t + e_t$
→ Modèle multiplicatif : $y_t = T_t * S_t * e_t$

- T_t : tendance.
- S_t : composante saisonnière.
- e_t : composante irrégulière.

2- La date

Puisque la série temporelle a une relation avec le temps, le R a des fonctions qui permettent de créer des

La fonction **date()** retourne la date du jour sous forme d'une chaine de caractères :

```
> j <- date()
>j
[1] "Thu Nov 20 17:16:30 2008"
```

as.date() : permet de spécifier la forme du date :

```
> dates <- c("17/02/92", "27/02/92", "14/01/92", "28/02/92", "01/02/92", "02/01/92")
> dates <- as.Date(dates, "%d/%m/%y")
>dates
[1] "1992-02-17" "1992-02-27" "1992-01-14" "1992-02-28" "1992-02-01"
[6] "1992-01-02"
```

difftime() : permet de calculer la déférence entre les dates

```
>difftime(dates[1], dates[2])
Time difference of -10 days

>difftime(dates[1], dates[2], units = "s")
Time difference of -864000 secs
```

seq () : permet de créer une série des dates

```
> d3 <- seq(from = as.Date("01/03/09", "%d/%m/%y"), to = as.Date("30/03/09",
+ "%d/%m/%y"), length.out = 15)
>d3
[1] "2009-03-01" "2009-03-03" "2009-03-05" "2009-03-07" "2009-03-09"
[6] "2009-03-11" "2009-03-13" "2009-03-15" "2009-03-17" "2009-03-19"
[11] "2009-03-21" "2009-03-23" "2009-03-25" "2009-03-27" "2009-03-30"
```

Avec des arguments :

→ **from** et **to** : pour le d´ebut et la fin de la s´equence
→ **by** pour le pas temporel (day, month)
→ **length.out** pour la longueur de l'objet.

3- Série temporelle sous R :

Création d'une série Temporelle

La fonction **ts** permet de créer des séries temporelles.

Il y a cinq arguments principaux :

→ **data :** les données décrivant la série temporelle
→ **start :** le temps de départ
→ **end :** le temps de fin
→ **frequency :** la fréquence d'échantillonnage ou le nombre d'observations par unité de temps
→ **deltat :** la période entre deux observations successives

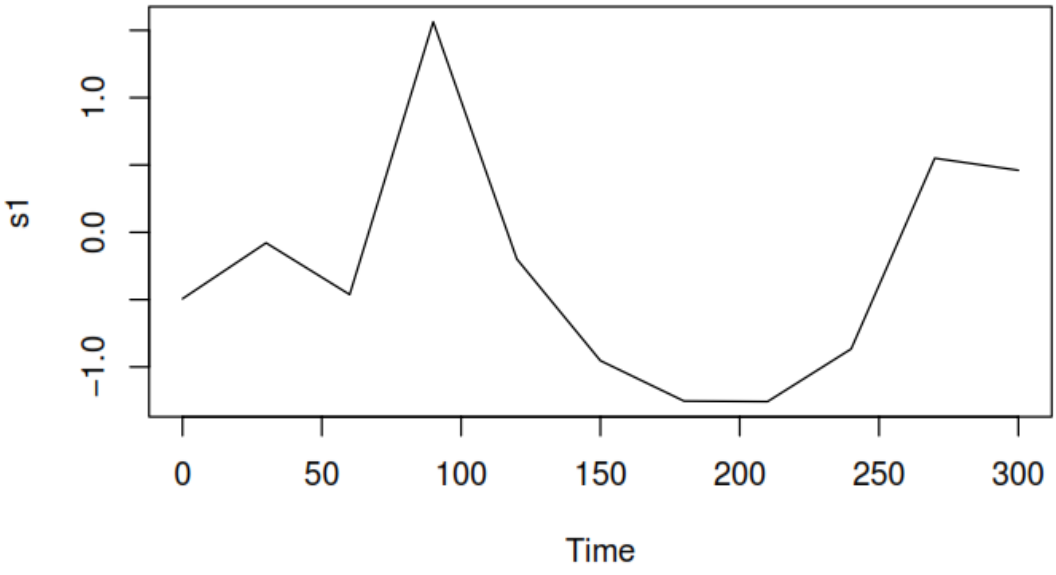
Exemple :

```
> s1 <- ts(data = rnorm(11), start = 0, end = 5 * 60, frequency = 1/30)
```

Présentation graphique d'une série temporelle :

Pour afficher la série il suffit de créer **s1**, et pour afficher en graphe en utilise la fonction **plot()**.
Par exemple : pour la série au-dessous on a :

```
>plot(s1)
```



Pour avoir des informations concernant la série temporelle on utilise les fonctions suivantes :

```
start(s3)          end(s3)          frequency(s3)      >deltat(s3)
[1]01              [1] 0 8000       [1] 8000                [1] 0.000125
```

Pour couper une partie de la série, on utilise la fonction **windwos() :**

```
> s4 <- window(s3, start = 0.25, end = 0.28)
```

Types de graphes :

plot(x)	graphe des valeurs de x (sur l'axe des y) ordonnées sur l'axe x
plot(x,y)	graphe de y en fonction de x
boxplot(x)	boîte à moustaches
²hist(x)	histogramme des fréquences de x
barplot(x)	histogramme des valeurs de x
qqnorm(x)	quantiles de x en fonction des valeurs attendues selon une loi normale
qqplot(x,y)	quantiles de y en fonction de ceux de x

Exemple d'une série temporelle en fonction de type de données :

N.B : a fonction **rnorm()** permet de générer le nombre aléatoire suivant un loi normale :

```
### données mensuelle.
x0<-ts(rnorm(60), start=c(2004,1), freq=12) ,

### données trimestrielle.
x1<-ts(rnorm(52), start=c(2004,1), freq=4) ,
```

4- Les processus AR et MA sous R

Avant d'analyse les modèles AR et MA, il faut installer le package appelé **tseries** en utilisant la fonction **library,** comme est montré :

```
library(tseries)
```

Processus AR : On dit que (Xt) est un processus auto-régressif d'ordre p (centré) s'il s'écrit sous la forme suivante :

$$X_t = \epsilon_t + \sum_{j=1}^{p} a_j X_{t-j},$$

Où (e_t) est un bruit blanc centré de variance σ

Simulation de processus AR :

Pour simuler un processus AR(1), on utilisantles fonctions suivantes :

```
### Pourcréer un processus AR(1)
ts.ar<- arima.sim(list(order = c(1,0,0), ar =0.7), n= 200)

### Pourteste la stationnarité un processus AR(1)
adf.test(ts.ar, alternative=c("stationary"), k=0)

### Pourteste l'invisibilité un processus AR(1)
adf.test(ts.ar, alternative=c("explosive"), k=0)

### présenter le processus graphiquement
ts.plot(ts.ar,type="l")
```

Calculer et tracer les auto-covariances et les autocorrélations :

Pour calculer et présenter les auto-covariances on utilise la fonction suivante :

```
###Pour calculer  et présenter les auto-covariances du processus graphiquement, en les limiter en 10
(acf(ts.ar,10))

###Pour calculer  et présenter toutes les auto-covariances du processus graphiquement,
(acf(ts.ar))
```

Pour calculer présenter les auto corrélations on utilise la fonction suivante :

```
###Pour calculer  et présenter les auto-covariances du processus graphiquement, en les limiter en 10
(pacf(ts.ar,10))

###Pour calculer  et présenter toutes les auto-covariances du processus graphiquement,
(pacf(ts.ar))
```

Processus MA : On appelle moyenne mobile (MovingAverage) d'ordre q un processus de la forme suivante :

$$X_t = \epsilon_t + b_1\epsilon_{t-1} + \ldots + b_q\epsilon_{t-q},$$

Où (e_t) est un bruit blanc centré de variance σ

Simulation de processus MA :

Pour simuler un processus MA(1), on utilisant les fonctions suivantes :

```
### Pourcréer un processus AR(1)
ts.ma<- arima.sim(list(order = c(1,0,0), ar =0.7), n= 200)

### Pourteste la stationnarité un processus AR(1)
adf.test(ts.ma, alternative=c("stationary"), k=0)

### Pourteste l'invisibilité un processus AR(1)
adf.test(ts.ma, alternative=c("explosive"), k=0)

### présenter le processus graphiquement
ts.plot(ts.ma,type="l")
```

Calculer et tracer les auto-covariances et les autocorrélations :

Pour calculer et présenter les auto-covariances on utilise la fonction suivante

```
###Pour calculer  et présenter les auto-covariances du processus graphiquement, en les limiter en 10
(acf(ts.am,10))

###Pour calculer  et présenter toutes les auto-covariances du processus graphiquement,
(acf(ts.am))
```

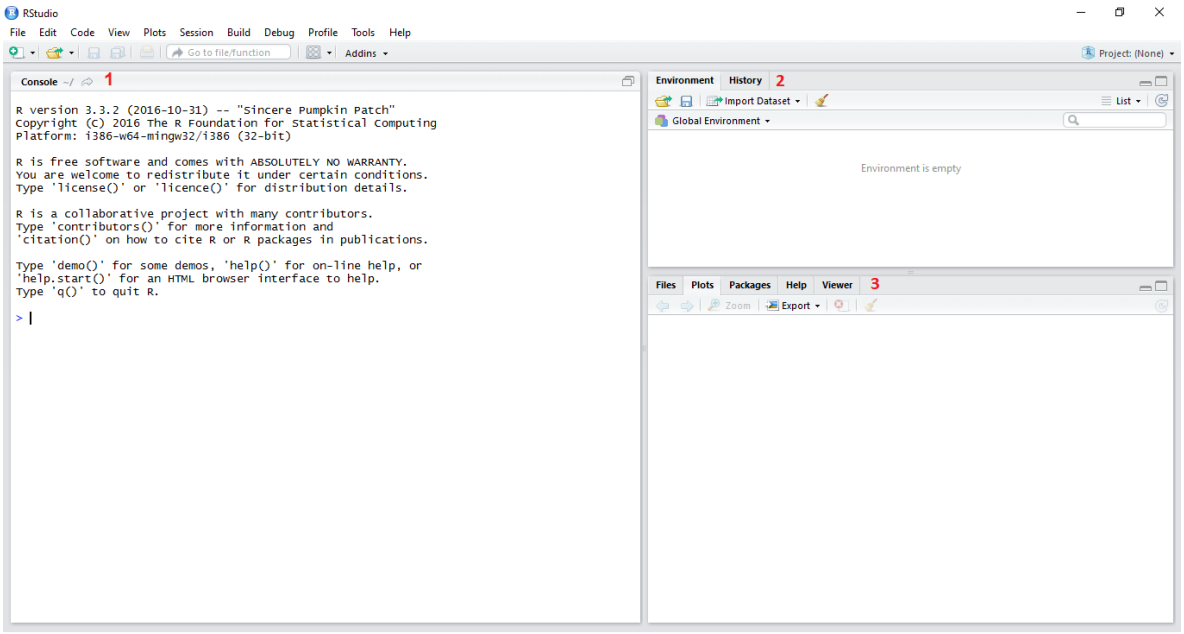
Pour calculer présenter les auto corrélations on utilise la fonction suivante :

```
###Pour calculer  et présenter les auto-covariances du processus graphiquement, en les limiter en 10
(pacf(ts.am,10))

###Pour calculer  et présenter toutes les auto-covariances du processus graphiquement,
(pacf(ts.am))
```

III- Mise en pratique :

1- Logiciel RStudio

Pour mettre en pratique ce qu'on vient de présenter, on va utiliser le logiciel RStudio .



1- le console : ou on taper les fonction
2- angle permet de visualiser les objets de travail
3- Angle files : permet d'explore le reprtoire courant
   angle plots : permet de navigue entre les graphe qu'on a construit.
   Angle packages : permet aussi de charger et d'installer les packages.

Le logiciel RStudio il est plus pratique que logiciel R, car il fourni plusieurs fonctionnalités

2- Instalation de package :

```
library(tseries)
```

```
Console ~/rachid/
> library(tseries)

    'tseries' version: 0.10-37

    'tseries' is a package for time series analysis and computational
    finance.

    See 'library(help="tseries")' for details.

>
```

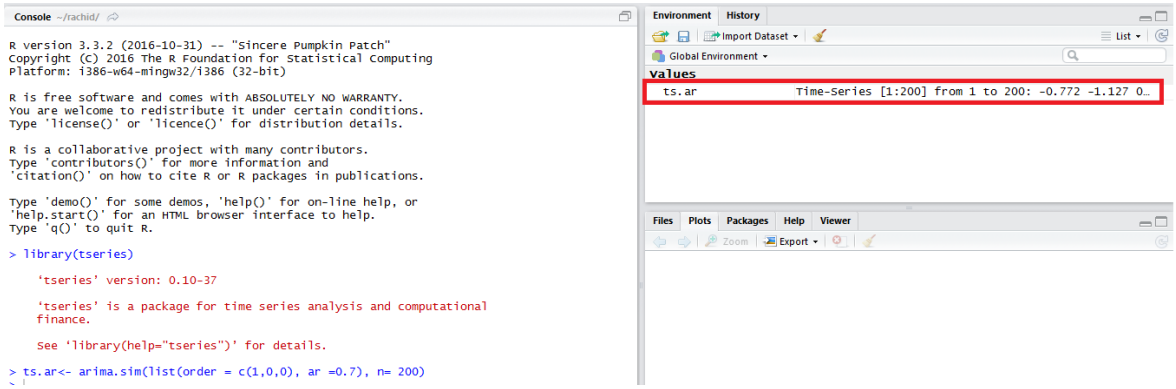
3- Analyser Le processus AR :

On prendre par exemple le processus suivant :

$$y_t = 0.7y_{t-1} + \varepsilon_t$$

Déclaration de processus :

```
ts.ar<- arima.sim(list(order = c(1,0,0), ar =0.7), n= 200)
```



Tester la stationnarité et l'inversibilité :

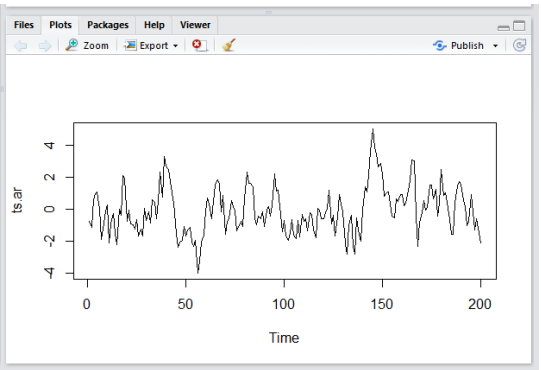
```
adf.test(ts.ar, alternative=c("stationary"), k=0)
adf.test(ts.ar, alternative=c("explosive "), k=0)
```

```
> adf.test(ts.ar, alternative=c("stationary"), k=0)

        Augmented Dickey-Fuller Test

data:  ts.ar
Dickey-Fuller = -5.5913, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary
> adf.test(ts.ar, alternative=c("explosive"), k=0)

        Augmented Dickey-Fuller Test

data:  ts.ar
Dickey-Fuller = -5.5913, Lag order = 0, p-value = 0.99
alternative hypothesis: explosive

>
```

Présentation graphique

```
ts.plot(ts.ar,type="l")
```

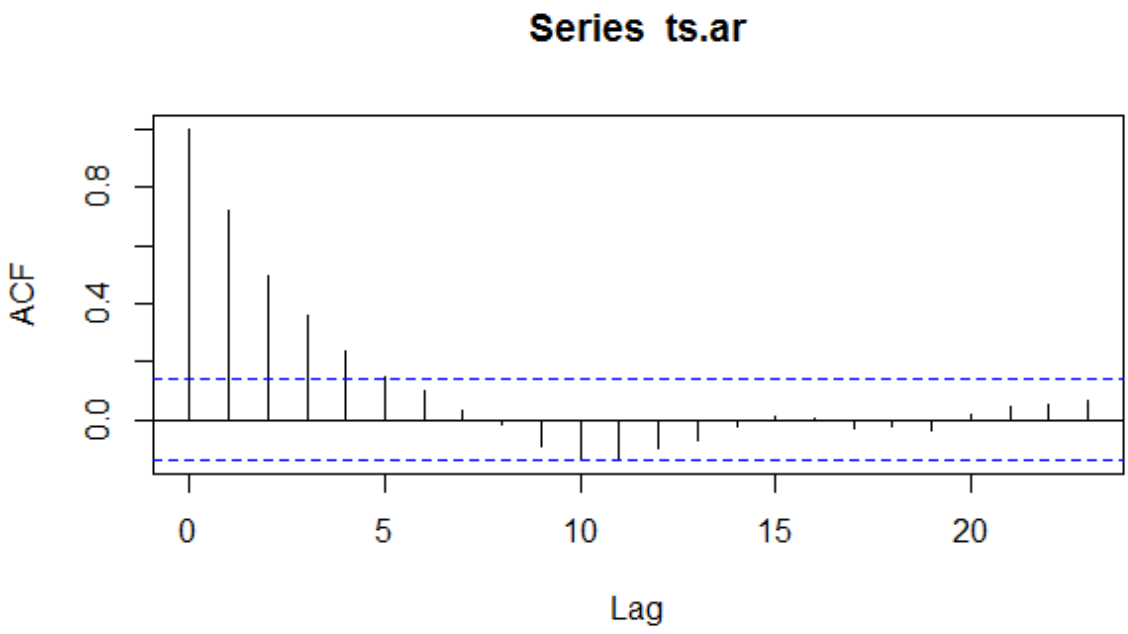
```
    See 'library(help="tseries")' for details.

> ts.ar<- arima.sim(list(order = c(1,0,0), ar =0.7), n= 200)
> adf.test(ts.ar, alternative=c("stationary"), k=0)

        Augmented Dickey-Fuller Test

data:  ts.ar
Dickey-Fuller = -5.5913, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(ts.ar, alternative = c("stationary"), k = 0) :
  p-value smaller than printed p-value
> adf.test(ts.ar, alternative=c("explosive"), k=0)

        Augmented Dickey-Fuller Test

data:  ts.ar
Dickey-Fuller = -5.5913, Lag order = 0, p-value = 0.99
alternative hypothesis: explosive

Warning message:
In adf.test(ts.ar, alternative = c("explosive"), k = 0) :
  p-value smaller than printed p-value
> ts.plot(ts.ar,type="l")
>
```



Les auto-covariances :

```
(acf(ts.ar))
```

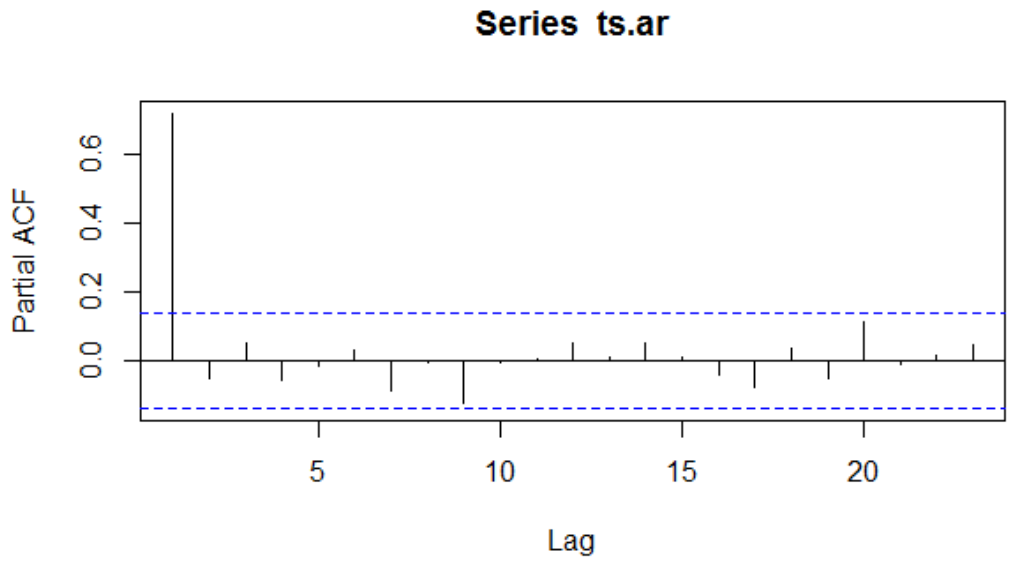
```
> (acf(ts.ar))

Autocorrelations of series 'ts.ar', by lag

     0      1      2      3      4      5      6      7      8      9     10     11
 1.000  0.719  0.492  0.359  0.238  0.144  0.102  0.031 -0.016 -0.092 -0.134 -0.136
    12     13     14     15     16     17     18     19     20     21     22     23
-0.101 -0.072 -0.023  0.009  0.006 -0.034 -0.026 -0.035  0.014  0.043  0.052  0.067
>
```



Les autocorrélations :

(pacf(ts.ar))

```
> (pacf(ts.ar))

Partial autocorrelations of series 'ts.ar', by lag

     1      2      3      4      5      6      7      8      9     10     11     12
 0.719 -0.053  0.052 -0.056 -0.018  0.032 -0.090 -0.003 -0.126 -0.005  0.005  0.051
    13     14     15     16     17     18     19     20     21     22     23
 0.011  0.048  0.012 -0.041 -0.080  0.037 -0.050  0.112 -0.010  0.015  0.044
>
```



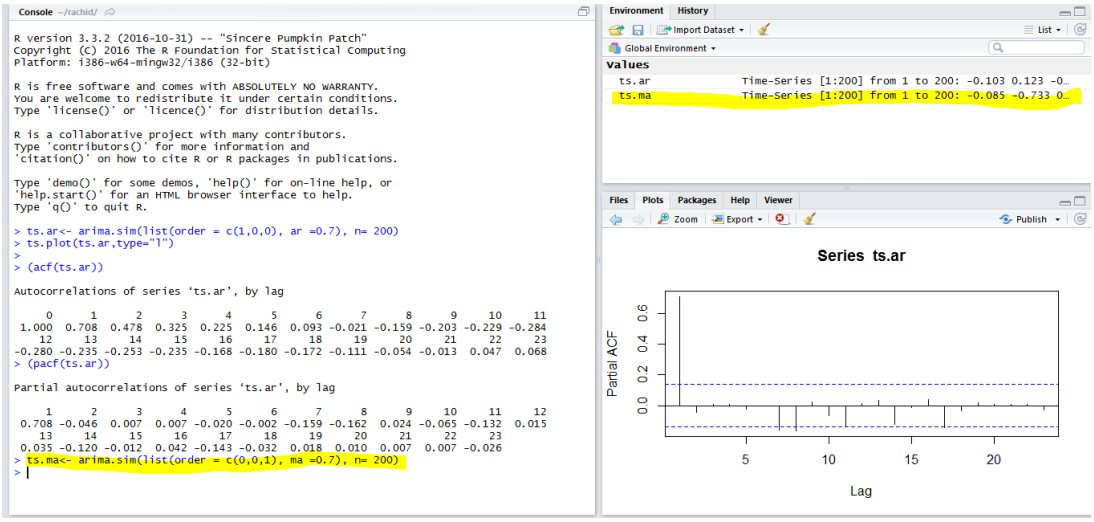
4- Analyser Le processus MA :

On prendre par exemple le processus suivant :

$$y_t = 0.7\varepsilon_{t-1} + \varepsilon_t$$

Déclaration de processus :

```
ts.ma<- arima.sim(list(order = c(0,0,1), ma =0.7), n= 200)
```



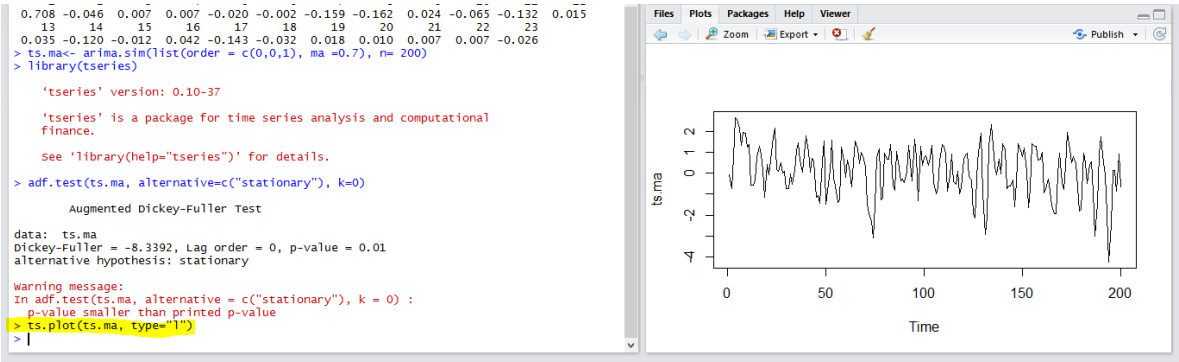
Test la stationnarité et l'inversible :

```
adf.test(ts.ma, alternative=c("stationary"), k=0)
adf.test(ts.ma, alternative=c("explosive "), k=0)
```

```
> adf.test(ts.ma, alternative=c("stationary"), k=0)

        Augmented Dickey-Fuller Test

data:  ts.ma
Dickey-Fuller = -6.7911, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary

> adf.test(ts.ma, alternative=c("explosive"), k=0)

        Augmented Dickey-Fuller Test

data:  ts.ma
Dickey-Fuller = -6.7911, Lag order = 0, p-value = 0.99
alternative hypothesis: explosive

> |
```

Présentation graphique

ts.plot(ts.ma,type="l")



Les auto-covariances :

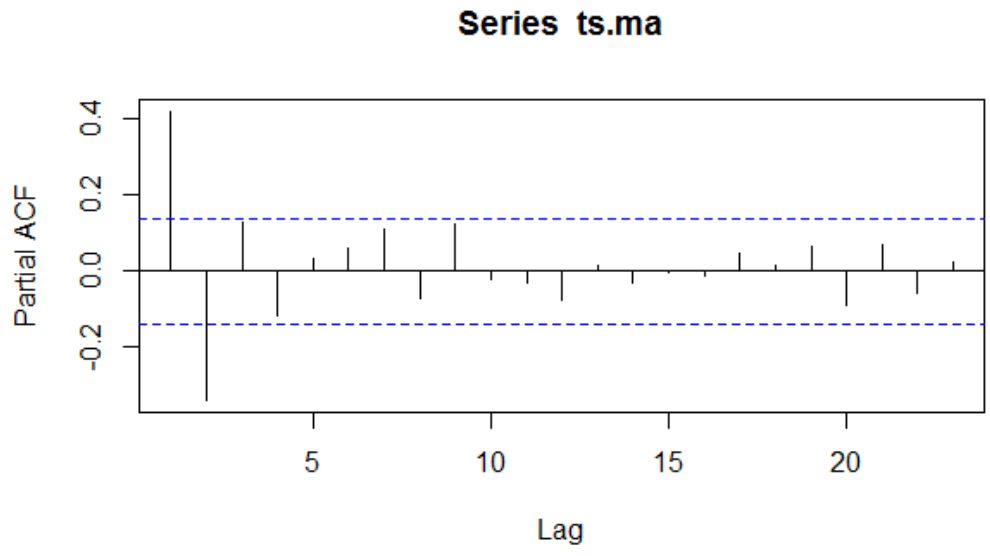
(acf(ts.ma))

```
> (acf(ts.ma))

Autocorrelations of series 'ts.ma', by lag

     0      1      2      3      4      5      6      7      8      9     10     11
 1.000  0.420 -0.103 -0.108 -0.055 -0.030  0.056  0.152  0.062  0.028  0.044 -0.029
    12     13     14     15     16     17     18     19     20     21     22     23
-0.110 -0.064  0.011  0.006  0.005  0.045  0.047  0.039 -0.029 -0.038 -0.025 -0.014
> |
```

Les autocorrélations :

(pacf(ts.ma))

```
> (pacf(ts.ma))

Partial autocorrelations of series 'ts.ma', by lag

     1      2      3      4      5      6      7      8      9     10     11     12
 0.420 -0.340  0.127 -0.119  0.035  0.061  0.109 -0.070  0.125 -0.023 -0.033 -0.075
    13     14     15     16     17     18     19     20     21     22     23
 0.016 -0.030 -0.005 -0.011  0.048  0.016  0.065 -0.090  0.069 -0.057  0.024
>
```



Conclusion

Bref ,le logiciel R présente de nombreux atouts mais aussi des faiblesses. Il semble important de savoir, si celles-ci seront comblées. Rappelons que le projet R est un projet de type cooperatif et il n'existe donc pas de prévisionnel concernant telle ou telle amélioration. Cependant, l'étendue des améliorations entre la première version de R et la version actuelle laisse à penser que de nombreuses améliorations seront réalisées et que ce logiciel constitue un choix d'avenir..