FACULTÉ POLYDISCIPLINAIRE DE TÉTOUAN

الكلية المتعددة التخصصات

جامعة عبد المالك السعدي
Université Abdelmalek Essaadi

Projet :

Manipulation des séries chronologiques dans le logiciel R

Réalisé par :

-Yannick Mwalaba

-Ismail Boudhaya

-Ahemed Salem Ahmed

-Fatimetou Aghrabat

-Aychetou Talebna

Encadré par :

Mr. Mohamed El marouani

Sommaire

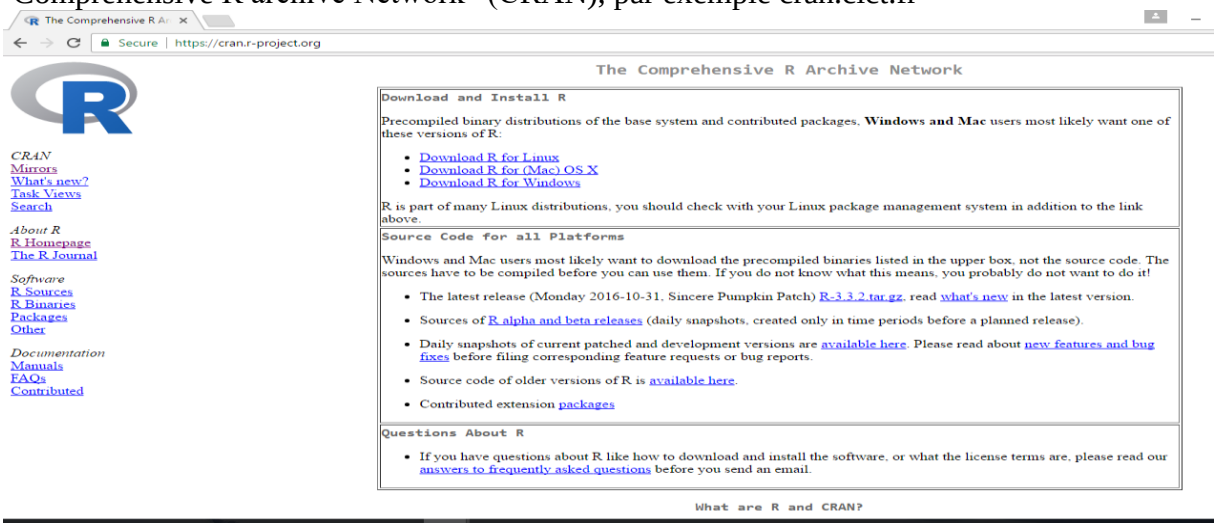
1-présentation du logiciel R

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. C'est la niche que nous tâchons d'occuper.

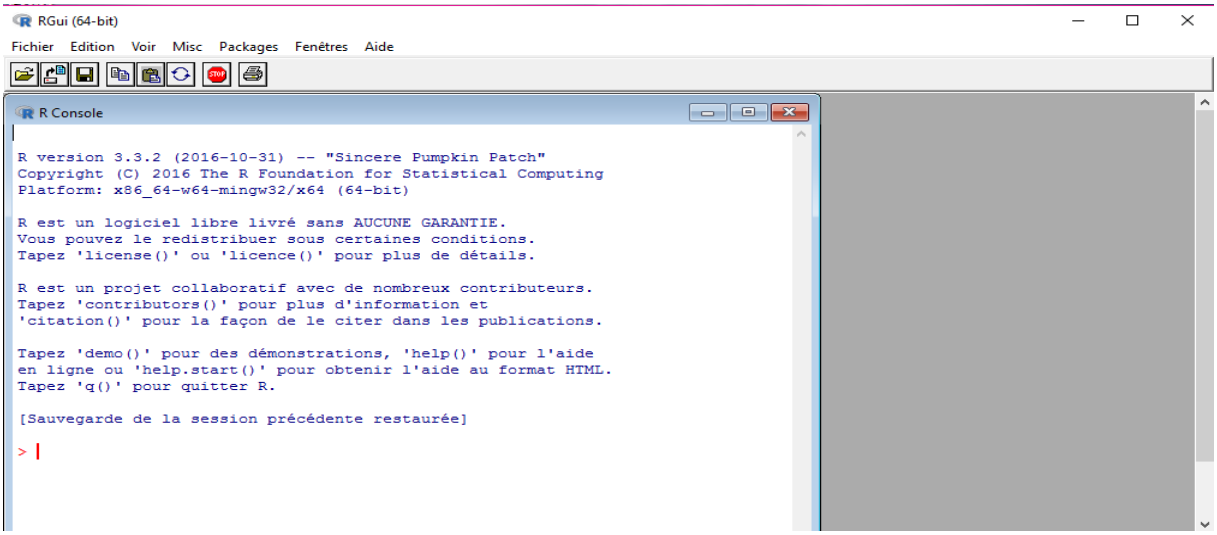
R est un environnement permettant de faire des analyses statistiques et de produire des graphiques. C'est un logiciel libre, clone d'un autre logiciel très célèbre dans la communauté statisticienne S+ (notez le clin d'œil). Il peut être téléchargé gratuitement sur www.r-project.org. Sur ce site vous trouverez également des documentations très complètes, notamment le manuel d'Emmanuel Paradis, chercheur en biologie à…Montpellier Comme tous les logiciels libres, le développement et l'amélioration de R peuvent être effectués par tout un chacun. Actuellement, R propose bien plus de fonctions statistiques que vous ne pourrez en utiliser pendant ce cours. Les développeurs insistent sur le fait que R permet un calcul vectoriel lui ouvrant des applications dans d'autres domaines que les statistiques. R est plus précisément un langage orienté-objet interprété. Outre les diverses procédures intégrées, R est très connu pour son interface graphique souple et qui permet d'exporter des graphiques de très bonne qualité à des formats variés très simplement. Enfin, contrairement à d'autres logiciels « boîtes noires », R oblige son utilisateur à une programmation minimale, ce qui le rend également plus souple que SAS ou SPSS.

1-1-Installation du système de base

Le site principal du logiciel R est www.r-project.org. Le téléchargement de R se fait a` partir d'un des sites du "Compréhensive R archive Network" (CRAN), par exemple cran.cict.fr



L'interface R :



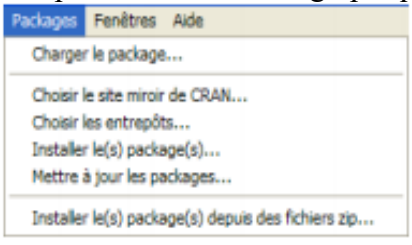
1-2-gestion des package

Un « package », ou bibliothèque, est généralement, un ensemble de fonctions R qui sont mises à la disposition des utilisateurs pour effectuer certains traitements qui n'existaient pas dans R

Les packages présents sont de deux types :

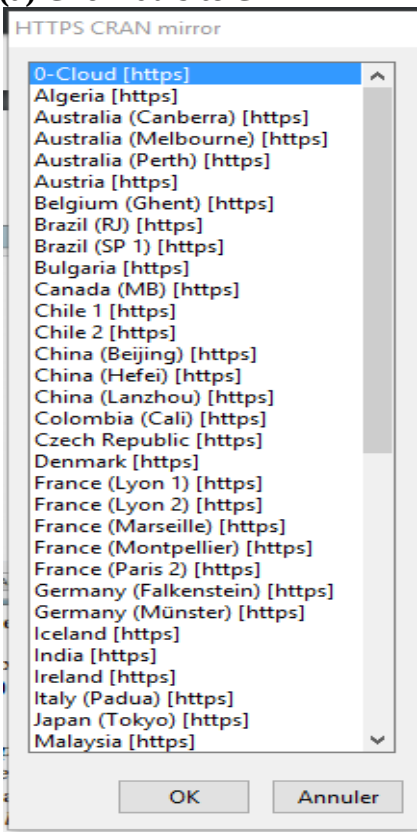
— packages de « base » au nombre de 14 : base, compiler, datasets, graphics, grDevices, grid, méthods, parralel, splines, stats, stats4, tcltk, tools, utils. Ils constituent en quelque sorte, le cœur du logiciel. Ils définissent le langage, permettent de faire les analyses courantes, donnent accès aux fonctions de base (mathématique, etc.) et aux fonctions permettant de faire les graphiques.
— packages « recommended » au nombre de 15 : boot, class, cluster, codetools, foreign, KernSmooth, lattice, MASS, Matrix, mgcv, nlme, nnet, rpart, spatial, survival. Ces packages complètent les fonctions de base. Les packages installés contiennent de très nombreuses fonctions mais évidemment pas toutes les fonctions dont on pourrait avoir besoin. Cela n'est pas un problème puisque, R étant un langage de programmation, on peut écrire ses propres fonctions et les utiliser de la même façon que les fonctions déjà présentes.
Il existe plusieurs manières d'installer un nouveau package. L'installation peut se faire :

— à partir de l'interface graphique de R : menu Packages de la fenêtre RConsole ;

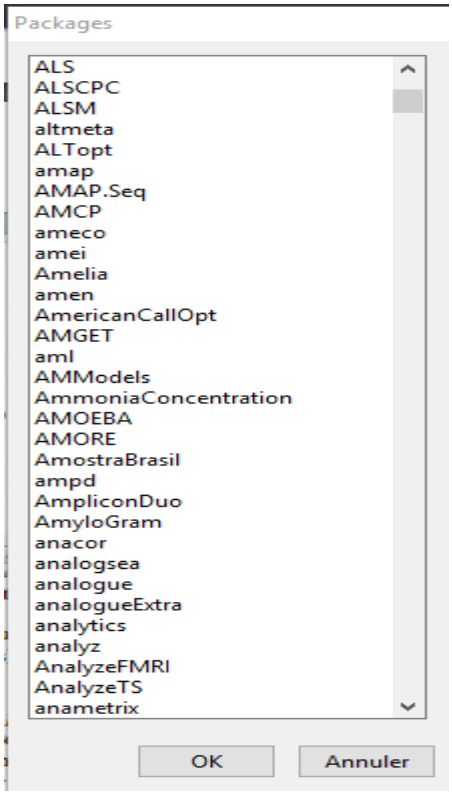


Dans ce menu, vous sélectionnez l'item Installer le(s) package(s) . . . R vous demande alors de choisir le site à partir duquel il va effectuer le téléchargement.

(a) Choix du site CRAN



(b) Package à installer



De même :

— en utilisant la fonction install.package().

Importer/exporter des données

1. Importer une suite : x=scan ("data.dat") : pour créer un vecteur `a partir de données stockées dans un fichier, ici data.dat.
2. Importer un tableau : x=read.table("data.dat") ou x=read.table("data.dat", header=TRUE) LSinstruction ˇ header=TRUE permet de préciser que la première ligne du fichier contient le nom des colonnes du tableau.
3. Exporter : write, write.table

1-3-manipulation (Graphiques et figures)

windows() ouvre une nouvelle fenêtre graphique
pdf(), png() , jpeg() , bitmap() , xfig() , pictex() , postscript() : pilote graphique produisant des sorties dans des fichiers plutôt qu'à l'écran
dev.off() ferme le pilote de sortie graphique pour clore le fichier de sortie
(mfrow=c(n,m)), on obtient alors n*m graphiques sur une même page repartis sur n lignes et m colonnes.

Commandes graphiques haut niveau

plot(x) trace les valeurs contenues dans x sur l'axe des y; s'adapte à la classe de l'objet x
plot(x, y) graphe bi varié (x sur l'axe des x, y sur l'axe des y)

hist(x) histogramme des fréquences de x
curve(expr) trace la fonction définie par l'expression exp.

2-série chronologique

2-1-Définition

Temporelles ou simplement chroniques) (en anglais : Time séries) est un ensemble d'observations enregistrées sur une période de temps. C'est une suite de données observées à intervalles de temps réguliers « réalisation d'un processus à temps discret). La fréquence des observations peut être annuelle, trimestrielle, mensuelle, journalière, ...L'analyse d'une série chronologique nous permet de prévoir la valeur d'une grandeur à l'instant en étudiant les valeurs passées de cette même grandeur .Ceci revient à étudier des méthodes statistiques de modélisation à des buts prédictifs.

2-2-Décomposition d'une série chronologique

Une série chronologique peut se décomposer en trois grandes composantes :

a. Le trend (composante tendancielle : déterministe), appelé aussi Tendance.

b. La Saison (composante déterministe)

c. Le Bruit (l'erreur : composante aléatoire ou stochastique)

2-3-Les modèles de composition

La décomposition d'une série chronologique avec un mouvement saisonnier peut s'effectuer selon deux types de modèles.

a-Modèle additif

La série chronologique xt se décompose en une tendance notée ct , des variations saisonnières st de période p (égales à s1, s2, s3, …, sp) et d'une composante accidentelle et
Le modèle le plus simple est le modèle additif, dans lequel la variation saisonnière s'ajoute simplement à la tendance : pour tout t = 1, …, T xt = ct + st + et

b-Modèle multiplicatif

Le second modèle est le modèle multiplicatif : pour tout t = 1, …, T
xt = ct (1 + st) + et

2-4-Lissage d'une série chronologique

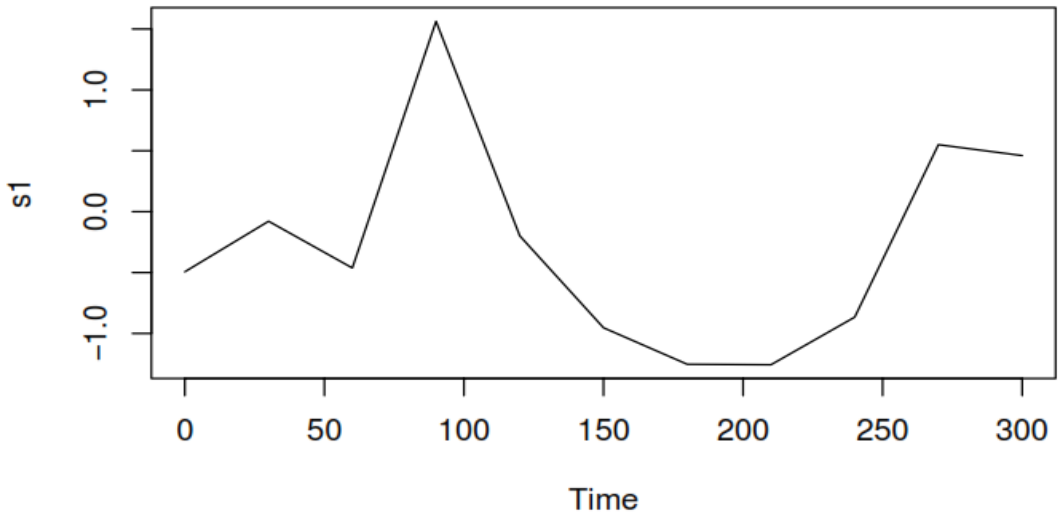
Ces méthodes constituent un moyen de prévision à partir d'observations d'une série chronologique. En voici quelques unes :

Lissage Exponentiel Simple
Lissage Exponentiel Double
Lissage de Holt-Winters

Création d'une série temporelle en R

ts est utilisée pour la création d'un objet time series dont la syntaxe est donnée par :
ts(x, start, freq).
Visualisation : plot(x)

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



2-5 les modeles AR , MA ,ARMA

Processus AR

Ces processus a tout instant t peuvent être extrapolés linéairement à partir de p valeurs précédents Xt − 1, ..., Xt − p à un bruit blanc près. X est dit autorégressif d'ordre p, si : Xp k=0 a k Xt − k = ε t (1) La fonction d'auto-corrélation partielle d'un processus AR(p) vérifie rp (k) = 0 ∀k > p.

simulation du processus AR(1)
ts.ar<- arima.sim(list(order = c(1,0,0), ar =x), n= 200)

pour la stationnarité de AR(1)
adf.test(ts.ar, alternative=c("stationary"), k=0)

pour tester l'invisibilité de AR(1)
adf.test(ts.ar, alternative=c("explosive"), k=0

présentation graphique
ts.plot(ts.ar,type="l")

Calcul des auto-covariances , les auto-corrélations et les autocorelations partielles :

Pour calculer et présenter les autocovariances
(acf(ts.ar,type="covariance"))

Pour calculer et présenter les autocorrelation
(acf(ts.ar,type="correlation"))

Pour calculer et présenter les autocorrelation partielles
(pacf(ts.ar))

Processus MA

Processus MA(q) Ces processus a tout instant t peuvent être extrapolés linéairement à partir de q résidus précédents ε t − 1, ..., ε t − p à un bruit blanc près. X est dit moyenne mobile d'ordre q, si : y t = Xq k=0 (2) θ k ε t − k La fonction d'auto-corrélation d'un processus MA(q) vérifie r q (k) = 0 ∀k > q.

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

Par exemple en veut d'analyse le processus suivant

y t = 0.7 ε t − 1 + ε t

pour la simulation du processus Ma(1)
ts.ma<- arima.sim(list(order = c(1,0,0), ar =y), n= 200)

pour tester la stationnarité de Ma(1)
adf.test(ts.ma, alternative=c("stationary"), k=0)

Pour tester l'invisibilité du processus Ma(1)
adf.test(ts.ma, alternative=c("explosive"), k=0)

Presenter le processus graphiquement
ts.plot(ts.ma,type="l")

Calcule des autocovariances les autocorrélations et les autocorrélations partielles :

Pour calculer et présenter les autocovariances
(acf(ts.ma,type="covariance"))

Pour calculer et présenter les autocorrelation
(acf(ts.ma,type="correlation"))

Pour calculer et présenter les autocorrelation partielles
(pacf(ts.ma))

Processus ARMA (p,q)

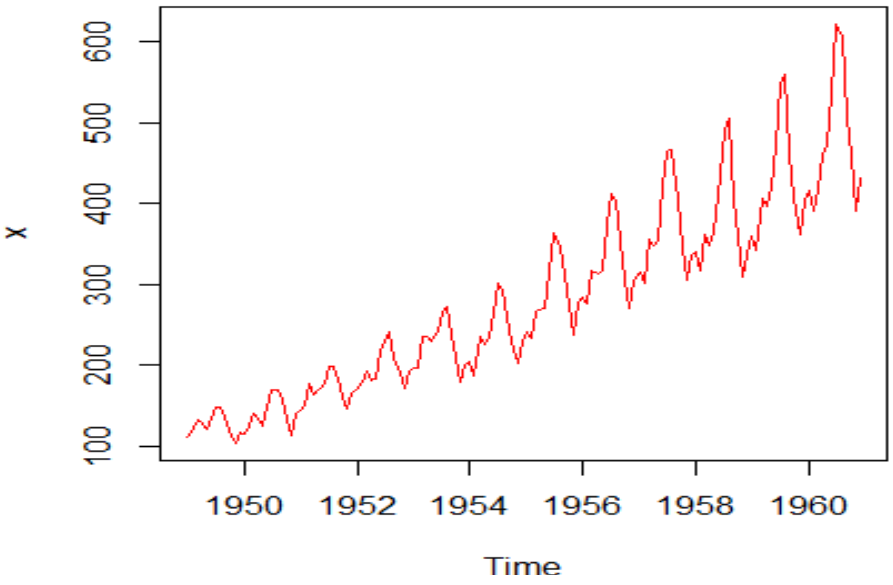
-La fonction diff différentie la série
-Les fonctions acf et la pacf calculent la corrélation et la corrélation partielle.
-La fonction arma () estime un ARMA : fit <- arma (y, order = c(p,q))
-La table d'analyse de variance s'obtient alors avec summary(fit), les graphiques de validation avec plot(fit), et les résidus avec : r <- résiduels(fit)
-Pour le contrôle graphique de la normalité des résidus : qqnorm et qqline.
Économétrie des séries temporelles avec logiciel R – p.16/22 -Pour un test de normalité :ks.test (test de Kolmogorov-Smirnov)
-La fonction predict() calcule les prévisions : fitarima <- arima(x, order=c(p,d,q)) tsdiag(fitarima) predict(fitarima, n.ahead = h) tsdiag fournit des graphiques de validation complémentaire.

II-plan pratique :

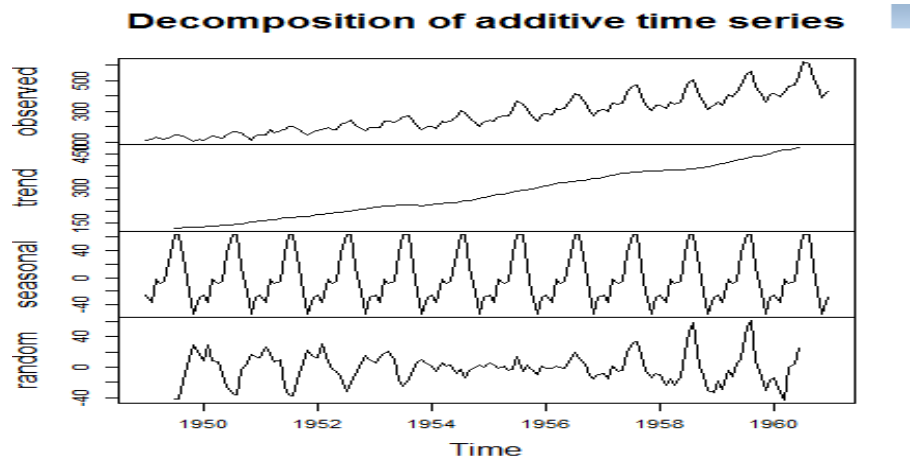
1-Simulation et manipulations d'une série chronologique

La série x représente le nombre de personnes en millier ayant pris un avion entre 1949 et 1960

```
R Console
> set.seed(123)
> data("AirPassengers")
> x=AirPassengers
> x
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
> plot(x,col="red")
```

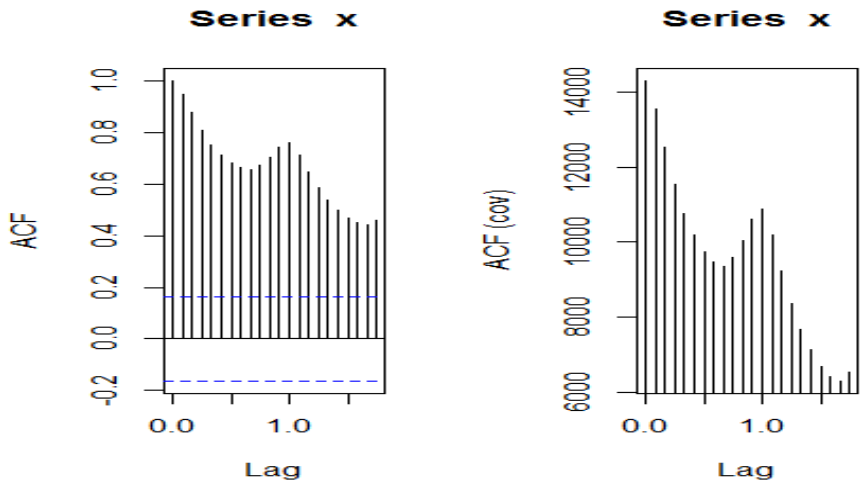


Décomposition de la série :



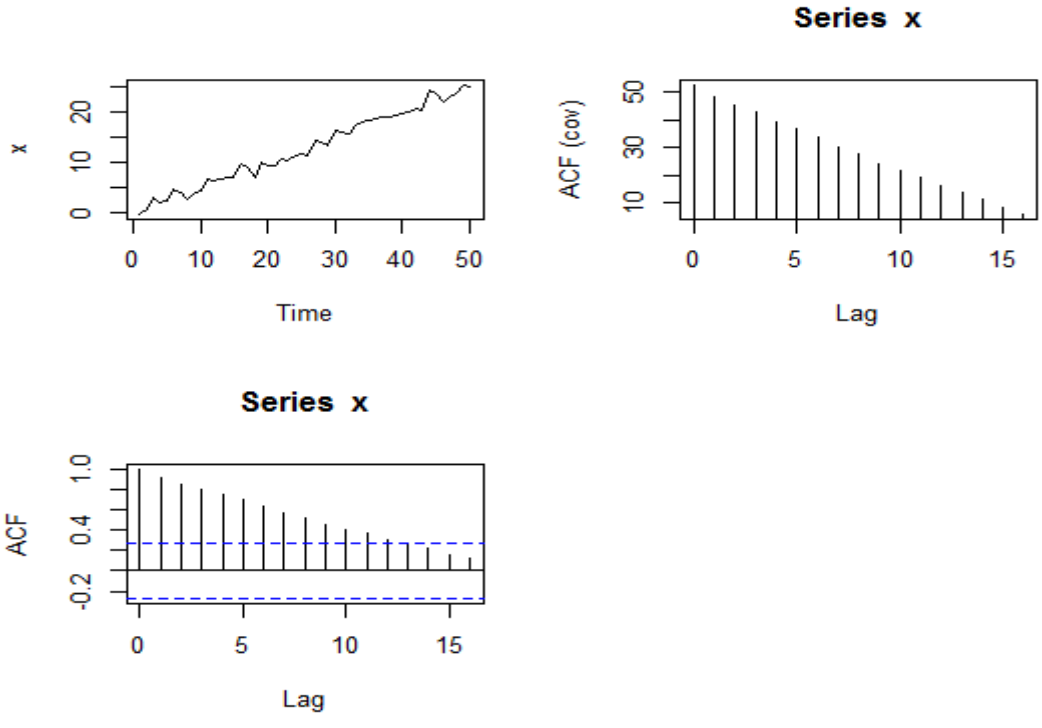
Auto-covariances et auto-corrélations de la série :

```
R Console
> set.seed(123)
> data("AirPassengers")
> x=AirPassengers
> x
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
> par(mfrow=c(1,2))
> acf(x,type="correlation")
> acf(x,type="covariance")
>
```



Simmulation d'une serie chronologique :

```
R Console
> set.seed(123)
> bb=rnorm(50)
> x=0.5*t+bb
> par(mfrow=c(2,2))
> plot.ts(x)
> acf(x,type="covariance")
> acf(x,type="correlation")
>
```



2-Analyse des processus AR, MA, ARMA

2-1-Analyse du processus AR

soit :
AR(1) : $X_t = 0.7X_{t-1} + \varepsilon_t$

simulation du processus

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

Teste de la stationnarité

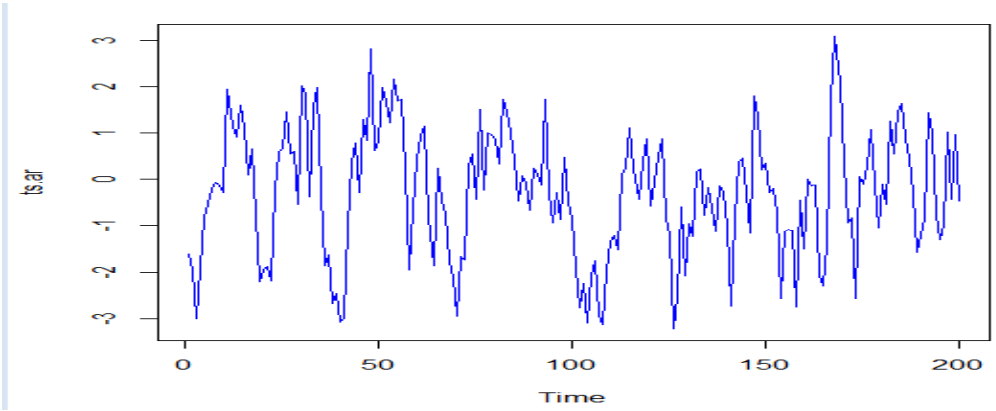
```
> adf.test(ts.ar, alternative=c("stationary"), k=0)

        Augmented Dickey-Fuller Test

data:  ts.ar
Dickey-Fuller = -6.5908, 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
```

Teste de l'invisibilité

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

        Augmented Dickey-Fuller Test

data:  ts.ar
Dickey-Fuller = -6.5908, 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
```

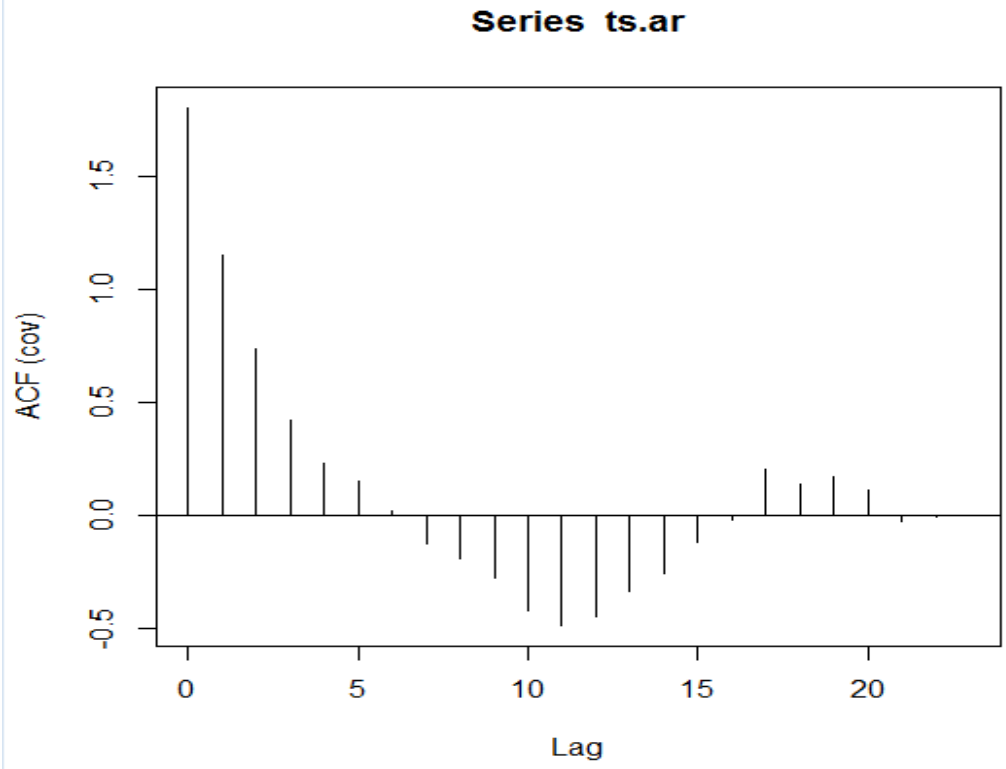
présentation graphique

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



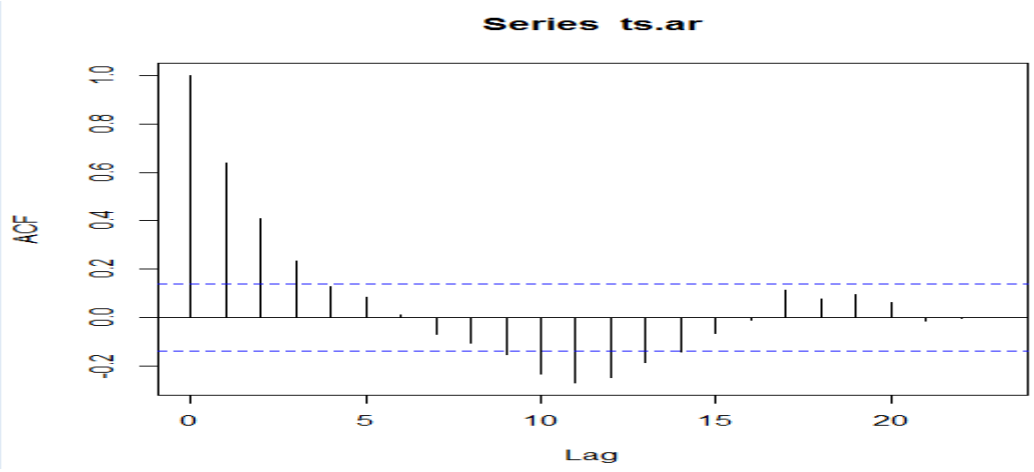
test d'auto-covariance

```
> (acf(ts.ar,type="covariance"))

Autocovariances of series 'ts.ar', by lag

      0       1       2       3       4       5       6       7       8       9
 1.8043  1.1557  0.7372  0.4216  0.2323  0.1496  0.0215 -0.1280 -0.1893 -0.2781
     10      11      12      13      14      15      16      17      18      19
-0.4235 -0.4871 -0.4468 -0.3356 -0.2586 -0.1198 -0.0224  0.1998  0.1388  0.1731
     20      21      22      23
 0.1136 -0.0280 -0.0108 -0.0027
> |
```



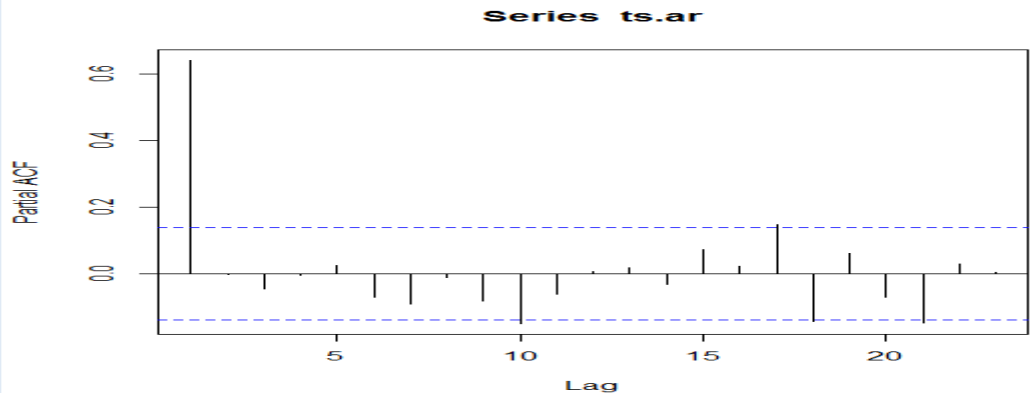
test d'auto-corrélation

```
> (acf(ts.ar,type="correlation"))

Autocorrelations of series 'ts.ar', by lag

     0      1      2      3      4      5      6      7      8      9     10
 1.000  0.641  0.409  0.234  0.129  0.083  0.012 -0.071 -0.105 -0.154 -0.235
    11     12     13     14     15     16     17     18     19     20     21
-0.270 -0.248 -0.186 -0.143 -0.066 -0.012  0.111  0.077  0.096  0.063 -0.016
    22     23
-0.006 -0.001
>
```



test d'auto-corrélation partielle

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

Autocorrelations of series 'ts.ar', by lag

     0      1      2      3      4      5      6      7      8      9     10
 1.000  0.641  0.409  0.234  0.129  0.083  0.012 -0.071 -0.105 -0.154 -0.235
    11     12     13     14     15     16     17     18     19     20     21
-0.270 -0.248 -0.186 -0.143 -0.066 -0.012  0.111  0.077  0.096  0.063 -0.016
    22     23
-0.006 -0.001
>
```



2-2-Analyse du processus MA

soit :
MA(1) :$Xt = \varepsilon t + 0.7\varepsilon t-1$

déclaration du processus

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

test de stationnarité

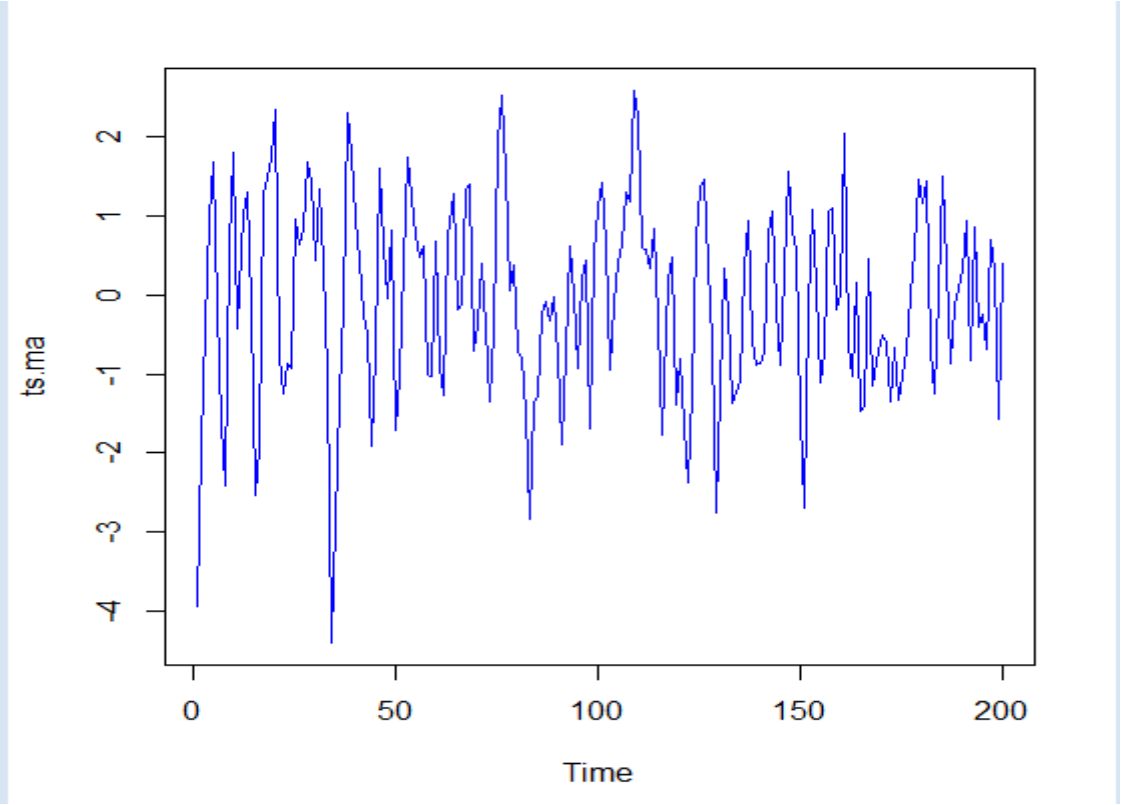
```
> adf.test(ts.ma, alternative=c("stationary"), k=0)

        Augmented Dickey-Fuller Test

data:  ts.ma
Dickey-Fuller = -8.8369, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary

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

test d'invisibilité

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

        Augmented Dickey-Fuller Test

data:  ts.ma
Dickey-Fuller = -8.8369, Lag order = 0, p-value = 0.99
alternative hypothesis: explosive

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

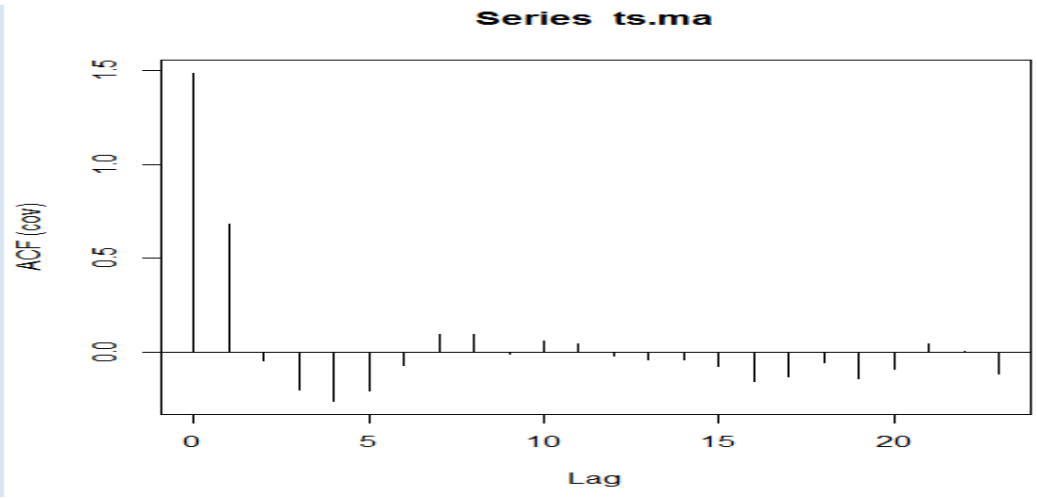
présentation graphique

```
> ts.plot(ts.ma,type="l",col="blue")
```



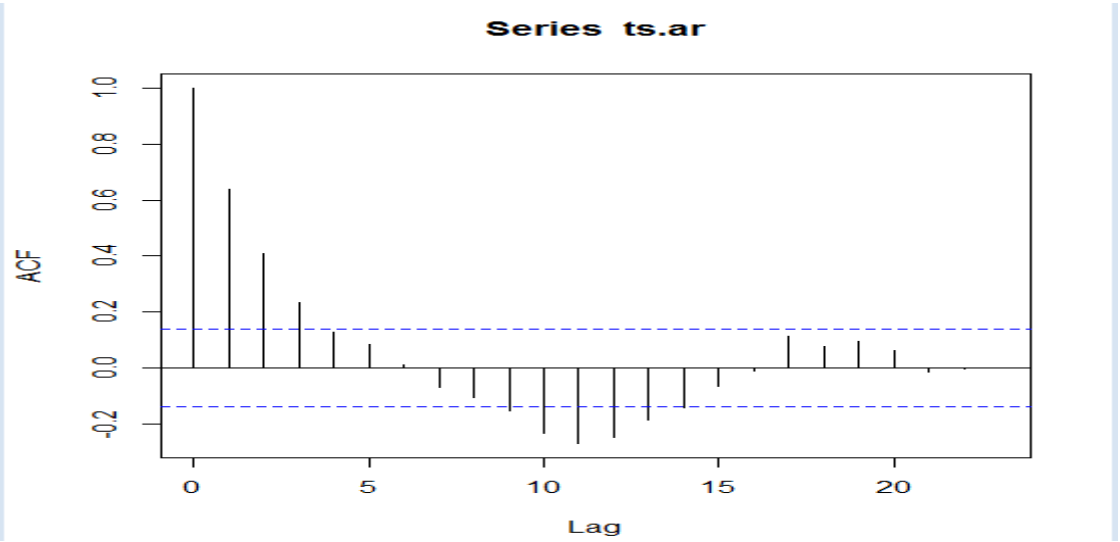
Test d'auto-covariance

```
> (acf(ts.ma,type="covariance"))

Autocovariances of series 'ts.ma', by lag

       0        1        2        3        4        5        6        7
 1.48563  0.68122 -0.04690 -0.19958 -0.26020 -0.20626 -0.07249  0.09551
       8        9       10       11       12       13       14       15
 0.09634 -0.00945  0.05829  0.04277 -0.02200 -0.04252 -0.04316 -0.07845
      16       17       18       19       20       21       22       23
-0.15823 -0.12991 -0.05552 -0.14327 -0.08946  0.04390  0.00654 -0.11836
```



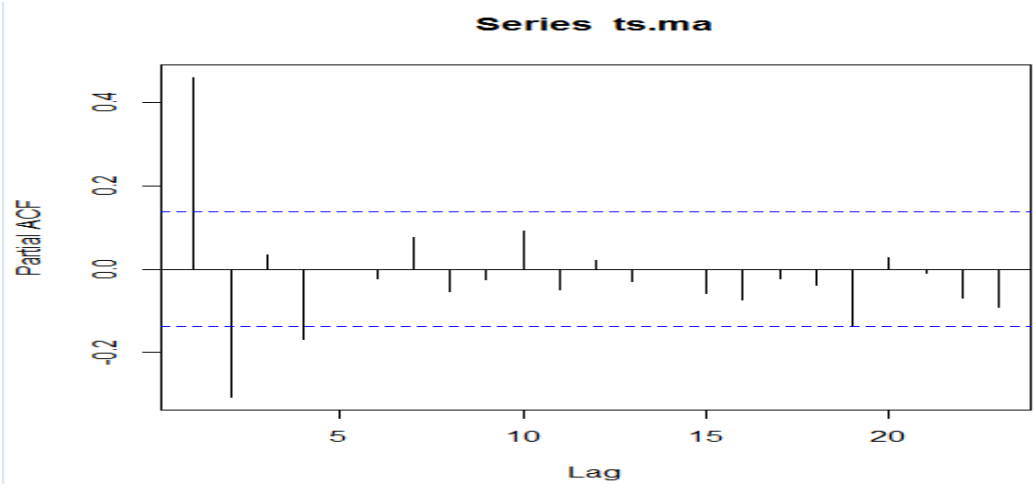
Test d'auto-corrélation

```
> (acf(ts.ma,type="correlation"))

Autocorrelations of series 'ts.ma', by lag

     0      1      2      3      4      5      6      7      8      9     10
 1.000  0.459 -0.032 -0.134 -0.175 -0.139 -0.049  0.064  0.065 -0.006  0.039
    11     12     13     14     15     16     17     18     19     20     21
 0.029 -0.015 -0.029 -0.029 -0.053 -0.107 -0.087 -0.037 -0.096 -0.060  0.030
    22     23
 0.004 -0.080
```



test d'auto-corrélation partiel

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

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

     1      2      3      4      5      6      7      8      9     10     11
 0.459 -0.306  0.035 -0.167 -0.002 -0.023  0.076 -0.053 -0.025  0.092 -0.049
    12     13     14     15     16     17     18     19     20     21     22
 0.020 -0.029 -0.001 -0.059 -0.074 -0.024 -0.037 -0.135  0.028 -0.011 -0.069
    23
-0.092
>
```



2-3-Analyse de processus ARMA

soit :
ARMA(1) : $X_t = 0.8X_{t-1} + 0.6\varepsilon_{t-1} + \varepsilon_t$

déclaration des processus

```
> library(stats)
>  ts.arma<- arima.sim(list(order = c(1,0,1), ar
+ = 0.8,ma=0.6), n = 200)
```

test de stationnarité

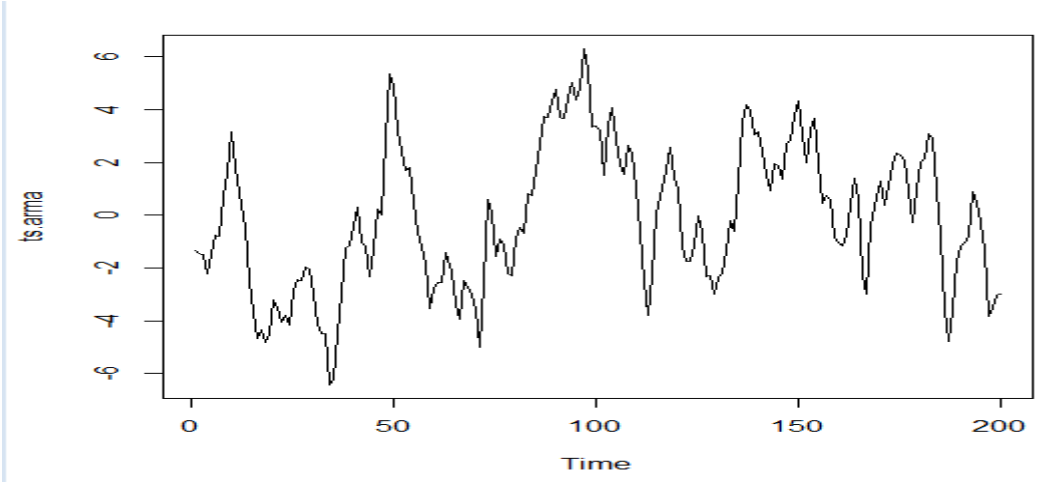
```
>  adf.test(ts.arma, alternative=c("stationary"), k=0)

        Augmented Dickey-Fuller Test

data:  ts.arma
Dickey-Fuller = -2.9042, Lag order = 0, p-value = 0.1974
alternative hypothesis: stationary
```

teste d'invisibilité

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

        Augmented Dickey-Fuller Test

data:  ts.arma
Dickey-Fuller = -2.9042, Lag order = 0, p-value = 0.8026
alternative hypothesis: explosive

>
```

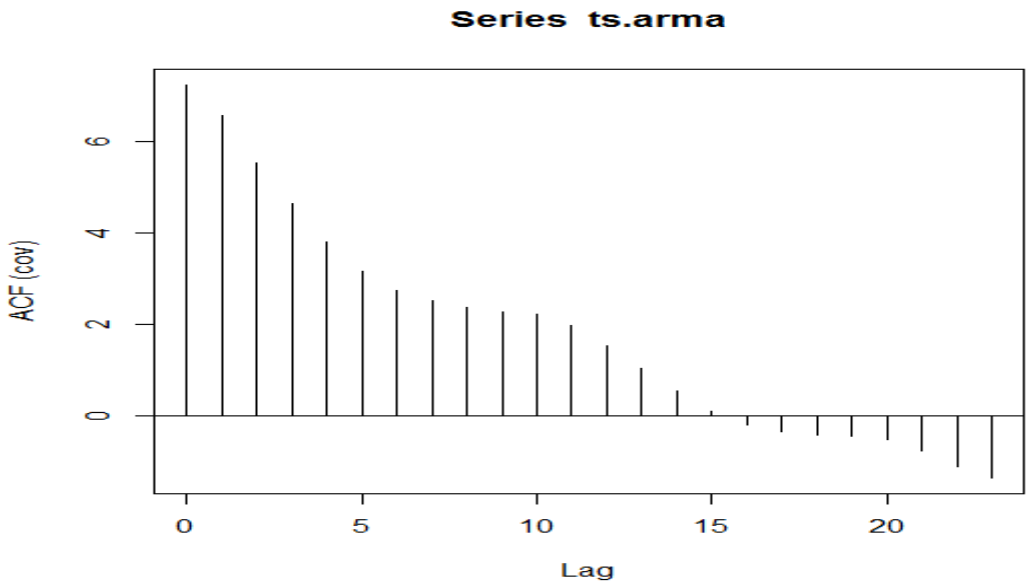
présentation graphique

```
>  ts.plot(ts.arma,type="l")
>
```



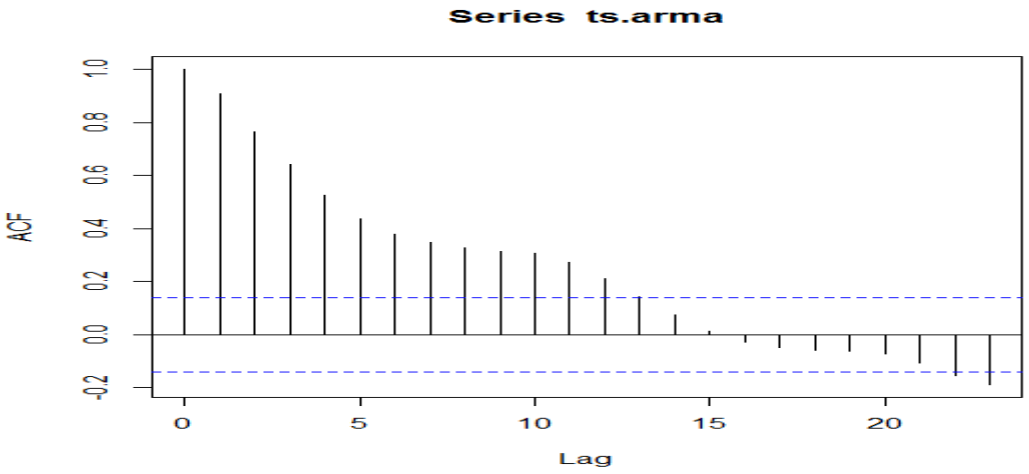
test d'auto-covariance

```
> (acf(ts.arma,type="covariance"))

Autocovariances of series 'ts.arma', by lag

      0       1       2       3       4       5       6       7       8       9
 7.2353  6.5793  5.5356  4.6375  3.8046  3.1646  2.7390  2.5159  2.3634  2.2667
     10      11      12      13      14      15      16      17      18      19
 2.2108  1.9650  1.5315  1.0426  0.5381  0.0982 -0.2014 -0.3423 -0.4159 -0.4580
     20      21      22      23
-0.5163 -0.7678 -1.1171 -1.3530
>
```



Test d'auto-corrélaltion

```
> (acf(ts.arma,type="correlation"))

Autocorrelations of series 'ts.arma', by lag

     0      1      2      3      4      5      6      7      8      9     10
 1.000  0.909  0.765  0.641  0.526  0.437  0.379  0.348  0.327  0.313  0.306
    11     12     13     14     15     16     17     18     19     20     21
 0.272  0.212  0.144  0.074  0.014 -0.028 -0.047 -0.057 -0.063 -0.071 -0.106
    22     23
-0.154 -0.187
>
```



test d'auto-corrélation partiel

```
> (pacf(ts.arma))

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

     1      2      3      4      5      6      7      8      9     10     11
 0.909 -0.357  0.143 -0.132  0.136  0.003  0.114 -0.052  0.088 -0.014 -0.138
    12     13     14     15     16     17     18     19     20     21     22
-0.061 -0.049 -0.046 -0.003  0.012  0.014 -0.037  0.005 -0.070 -0.145  0.007
    23
 0.058
>
```