Analyse des systèmes intégrés et cointégrés des séries temporelles

*Henri Makika**

Junho 1, 2019

Introduction

Dans ce travail, nous analysons des systèmes intégrés et cointégrés en utilisant le langage R, version R Markdown. Premièrement, nous présentons la série univariée avec toute ses caractéristiques et en second lieu, nous présentons une analyse multivariée de séries temporelles intégrant les modèles VAR, SVAR, Cointégration, VECM et SVECM. Nous terminerons notre analyse avec les modèles ARCH et GARCH.

Séries temporelles univariées

Une série temporelle discrète est définie comme une séquence ordonnée de nombres aléatoires par rapport au temps. Plus formellement, un tel processus stochastique peut être écrit comme suit:

$$y(s,t), s \in \varsigma, t \in \nu$$

Où $t \in \nu, y(.,t)$ est une variable aléatoire dans l'espace ς et une réalisation de ce processus stochastique est donné par $y(s,.)$ pour chaque $s \in \varsigma$ en ce qui concerne un moment donné $t \in \nu$.

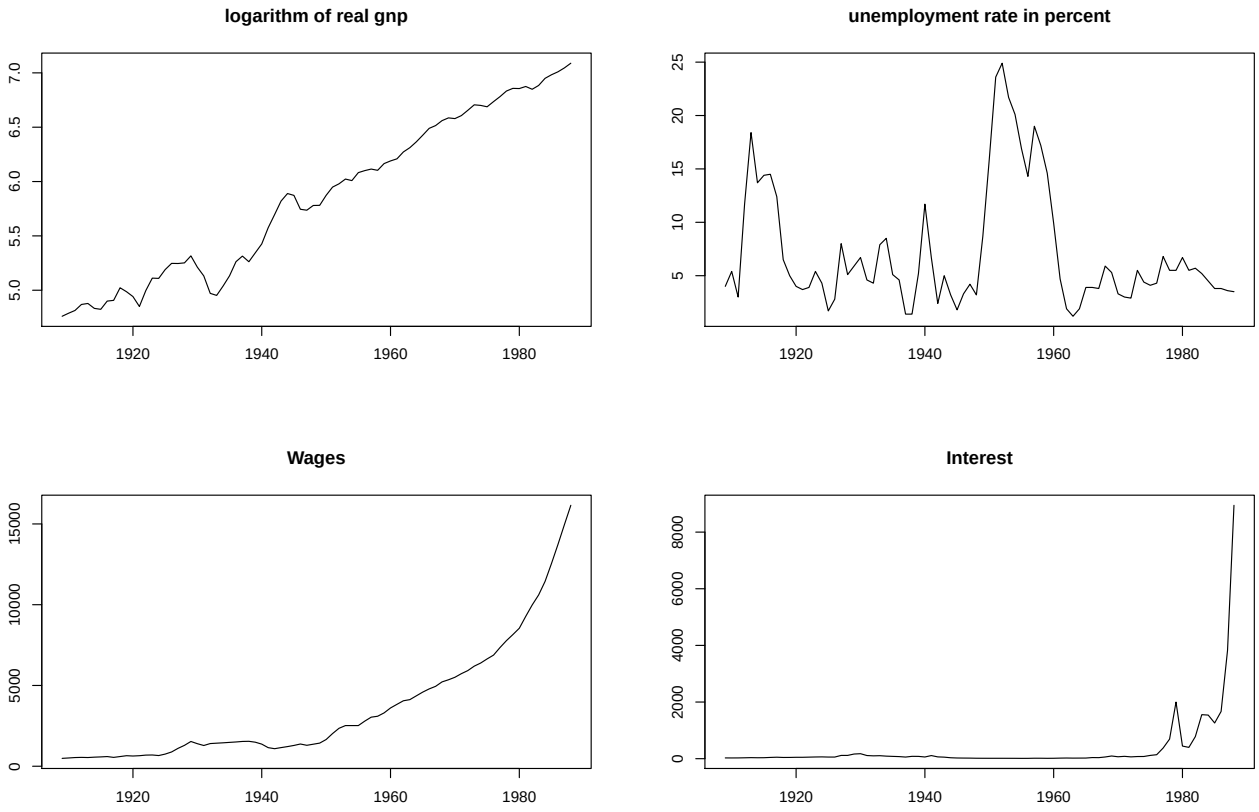
1 Présentation graphique d'un processus stochastique

```
library(urca)
data(npext)
attach(npext)
head(npext)
```

```
##   year      cpi employmt gnpdefl nomgnp interest    indprod gnpperca
## 1 1860 3.295837       NA      NA     NA       NA -0.1053605       NA
## 2 1861 3.295837       NA      NA     NA       NA -0.1053605       NA
## 3 1862 3.401197       NA      NA     NA       NA -0.1053605       NA
## 4 1863 3.610918       NA      NA     NA       NA  0.0000000       NA
## 5 1864 3.871201       NA      NA     NA       NA  0.0000000       NA
## 6 1865 3.850148       NA      NA     NA       NA  0.0000000       NA
##   realgnp wages realwag sp500 unemploy velocity  M
## 1      NA    NA      NA    NA       NA       NA NA
## 2      NA    NA      NA    NA       NA       NA NA
## 3      NA    NA      NA    NA       NA       NA NA
## 4      NA    NA      NA    NA       NA       NA NA
## 5      NA    NA      NA    NA       NA       NA NA
## 6      NA    NA      NA    NA       NA       NA NA
```

*University of Campinas, São Paulo. Email : hd.makika@gmail.com

```r
x = ts(na.omit(npext$realgnp), start = 1909, end = 1988, frequency = 1)
y = ts(exp(na.omit(npext$unemploy)), start = 1909, end = 1988, frequency = 1)
z = ts(exp(na.omit(npext$wages)), start = 1909, end = 1988, frequency = 1)
w = ts(exp(na.omit(npext$interest)), start = 1909, end = 1988, frequency = 1)
par(mfrow = c(2,2))
plot(x, xlab = "", ylab = "", main = "logarithm of real gnp")
plot(y, xlab = "", ylab = "", main = "unemployment rate in percent")
plot(z, xlab = "", ylab = "", main = "Wages")
plot(w, xlab = "", ylab = "", main = "Interest")
```



2 Stationnarité

1. Faible stationnarité : La forme améliorée d'un processus stationnaire est appelée faiblement stationnaire et est définie comme suit:

$$E[y_t] = \mu < \infty, \forall t \in \varsigma$$
$$E[(y_t - \mu)(y_{t-j} - \mu)] = \gamma_t, \forall t \in \varsigma$$

Comme seuls les deux premiers moments théoriques du processus stochastique doivent être définis et qu'ils sont constants et finis dans le temps, ce processus est également appelé stationnaire ou covariance du second ordre.

2. Stationnarité stricte : Le concept de processus strictement stationnaire est défini comme suit:

$$F[y_1, y_2, ..., y_t, ..., y_T] = F[y_{1+j}, y_{2+j}, ..., y_{t+j}, ..., y_{T+j}]$$

Où $F[.]$ est la fonction de distribution conjointe et $\forall t, j \in \varsigma$.

Par conséquent, si un processus est strictement stationnaire avec des moments finis, alors il doit également être stationnaire. Bien que les processus stochastiques puissent être configurés pour être stationnaires par covariance, il ne faut pas que ce soit un processus strictement stationnaire. Ce serait le cas, par exemple, si la moyenne et les autocovariances n'étaient pas des fonctions du temps mais des moments plus élevés.

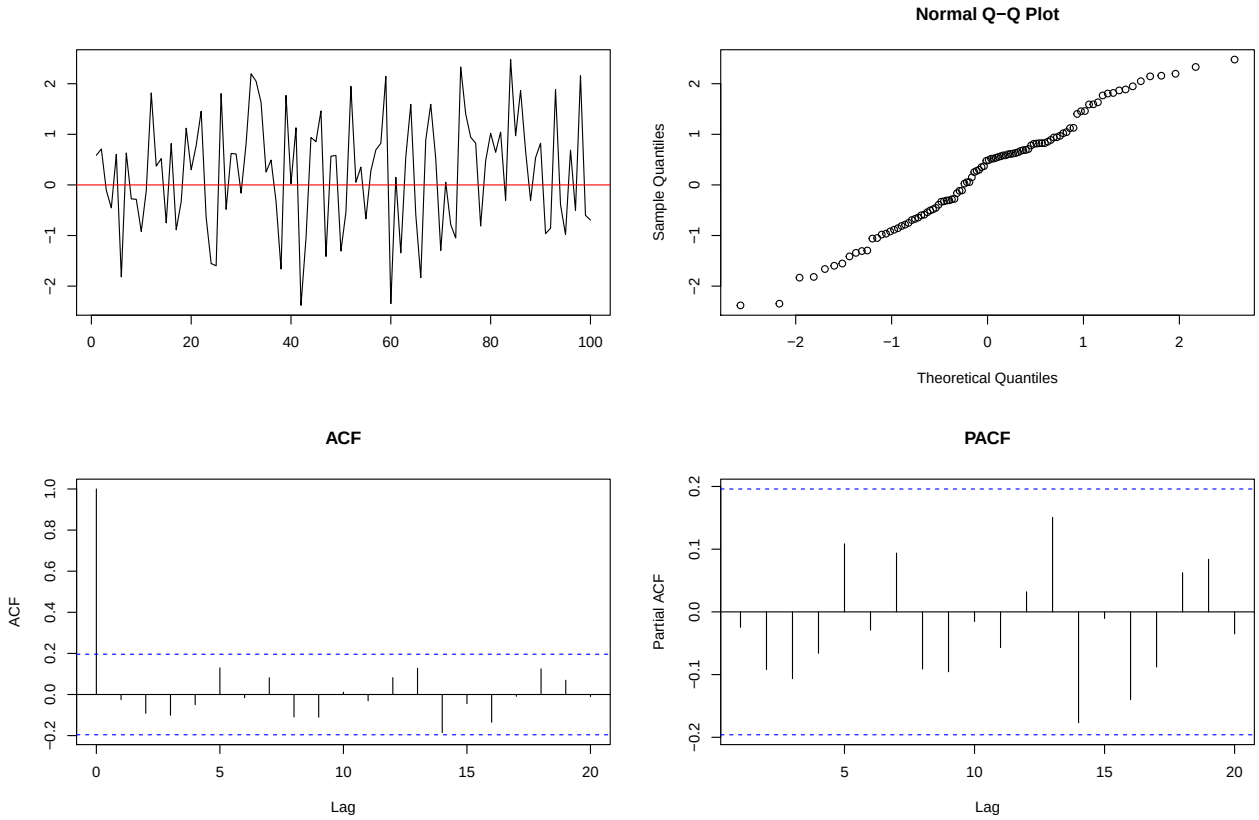
3 Bruit blanc

Un processus est dit *bruit blanc* lorsqu'il est défini comme:

$$E(\varepsilon_t) = 0$$
$$E(\varepsilon_t^2) = \delta^2$$
$$E(\varepsilon_t, \varepsilon_\tau) = 0, \forall t \neq \tau$$

On suppose que ε_t est normalement distribué: $\varepsilon_t \sim N(0, \delta^2)$. Si les hypothèses 1 et 2 sont modifiées par cette hypothèse, alors le processus est dit processus bruit blanc normal ou gaussien. De plus, la dernière hypothèse est parfois remplacée par la plus forte hypothèse d'indépendance. Si tel est le cas, le processus est dit processus bruit blanc indépendant. Veuillez noter que pour les variables aléatoires normalement distribuées, la non corrélation et l'indépendance sont équivalentes. Autrement, l'indépendance est suffisante pour le découplage mais pas l'inverse. Voici l'exemple d'un processus bruit blanc :

```
set.seed(12345)
gwn = rnorm(100)
layout(matrix(1:4, ncol = 2, nrow = 2))
plot.ts(gwn, xlab = "", ylab = "")
abline(h = 0, col = "red")
acf(gwn, main = "ACF")
qqnorm(gwn)
pacf(gwn, main = "PACF")
```



4 Ergodicité

Ergodicité fait référence à un type d'indépendance asymptotique. Plus formellement, l'indépendance asymptotique peut être définie comme:

$$|F(y_1, y_2, ..., y_T, y_{j+1}, y_{j+2}, ..., y_{j+T}) - F(y_1, y_2, ..., y_T)F(y_{j+1}, y_{j+2}, ..., y_{j+T})| \to 0$$

Où $j \to \infty$. La distribution conjointe de deux sous-séquences d'un processus stochastique $[y_t]$ est égale au produit des fonctions de distribution marginales, plus les deux sous-séquences sont éloignées l'une de l'autre. Un processus stochastique stationnaire est ergodique si :

$$\lim_{T \to 0} \frac{1}{T} \sum_{j=1}^{T} E[y_t - \mu][y_{t+j} - \mu] = 0, tient.$$

Cette équation serait satisfaite si les autocovariances tendent à zéro avec l'augmentation de j.

L'indépendance asymptotique signifie que deux réalisations d'une série chronologique se rapprochent de plus en plus de l'indépendance, plus elles se séparent par rapport au temps.

5 Théorème

Toute série temporelle stationnaire de covariance y_t peut être représentée sous la forme:

$$y_t = \mu + \sum_{j=0}^{\infty} \psi_j \varepsilon_{t-j}, \varepsilon_t \sim WN(0, \delta^2)$$

$$\psi_0 = 1, \sum_{j=0}^{\infty} \psi_j^2 < \infty$$

Avec comme caractéristiques :

i. Moyenne fixe : $E[y_t] = \mu$
ii. Variance finie : $\gamma_0 = \delta^2 \sum_{j=0}^{\infty} \psi_j^2 < 0$

6 Méthodologie Box et Jenkins

i. Modèles à moyenne mobile autorégressifs (ARMA)
ii. Forme approximative d'une série temporelle stationnaire par un modèle paramétrique parcimonieux
iii. Modèle ARMA (p, q) se présente comme suit:

$$y_t - \mu = \phi_1(y_{t-1} - \mu) + ... + \phi_p(y_{t-p} - \mu) + \varepsilon_t + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q}$$
$$\varepsilon_t \sim WN(0, \delta^2)$$

Extension pour les séries temporelles intégrées est la classe de modèle ARIMA(p,d,q).

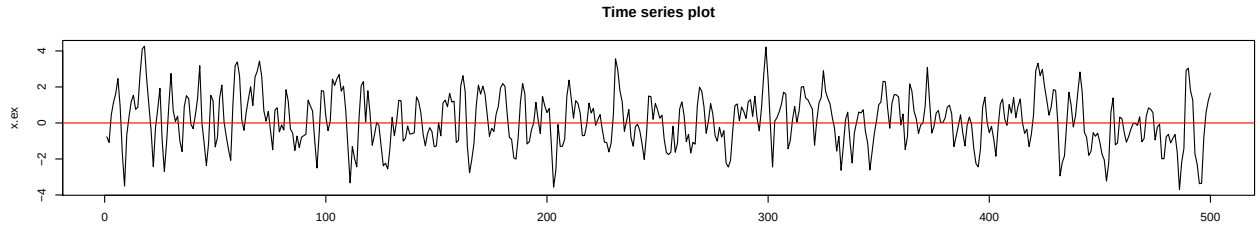
7 Procédure

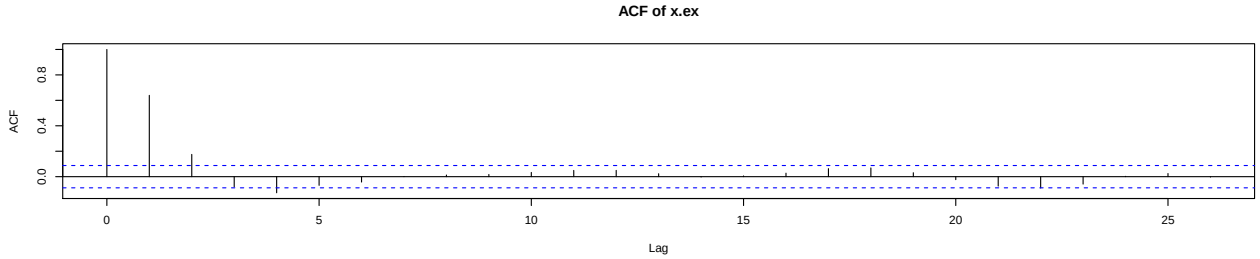
i. Si cela est nécessaire, transformez les données, de manière à obtenir la stationnarité de covariance;
ii. Inspectez, les graphiques d'autocorrélation, ACF et d'autocorrélation partielle, PACF pour les suppositions initiales de p et q;
iii. Estimez le modèle proposé;
iv. Vérifiez les résidus (tests de diagnostic) et la stationnarité du processus;
v. Si le point iv échoue, passez au point ii et recommencez. En cas de doute, choisissez la spécification de modèle la plus parcimonieuse.

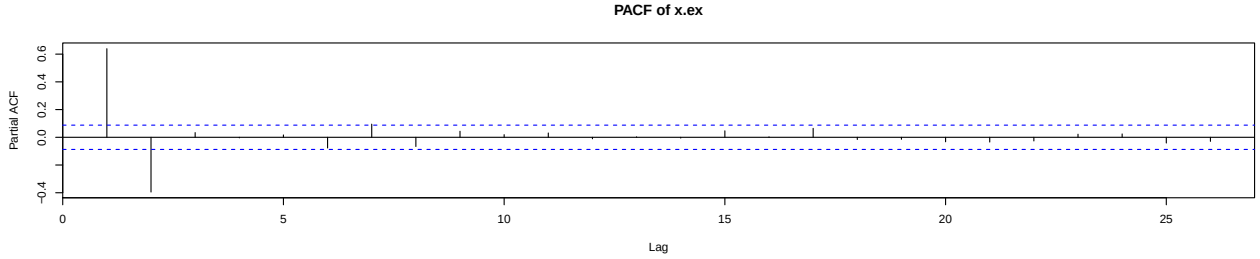
```
library(dse1)
library(forecast)
library(stats)
```

Exemple du modèle ARMA(2,0)

```
set.seed(12345)
x.ex = arima.sim(n = 500, list(ar = c(0.9, -0.4)))
layout(matrix(1:3, nrow = 3, ncol = 1))
plot(x.ex, xlab = "", main = "Time series plot")
abline(h = 0, col = "red")
acf(x.ex, main = "ACF of x.ex")
pacf(x.ex, main = "PACF of x.ex")
```








```
arma20 = arima(x.ex, order = c(2, 0, 0), include.mean = FALSE)
result = matrix(cbind(arma20$coef, sqrt(diag(arma20$var.coef))), nrow = 2)
rownames(result) = c("ar1", "ar2")
colnames(result) = c("estimate", "s.e.")
result
```

```
##       estimate       s.e.
## ar1  0.8955016 0.04109141
## ar2 -0.3917359 0.04111454
```

Processus non stationnaire

De nombreuses séries chronologiques économiques ou financières présentent un comportement de tendance. D'où il faut déterminer la forme la plus appropriée de cette tendance. Une série temporelle est dit stationnaire, c'est lorsque les moments sont invariants dans le temps. En distinction au processus non stationnaire qui, ce dernier leurs moments sont dépendants du temps (principalement moyenne et variance).

Décomposition en cycle de tendance

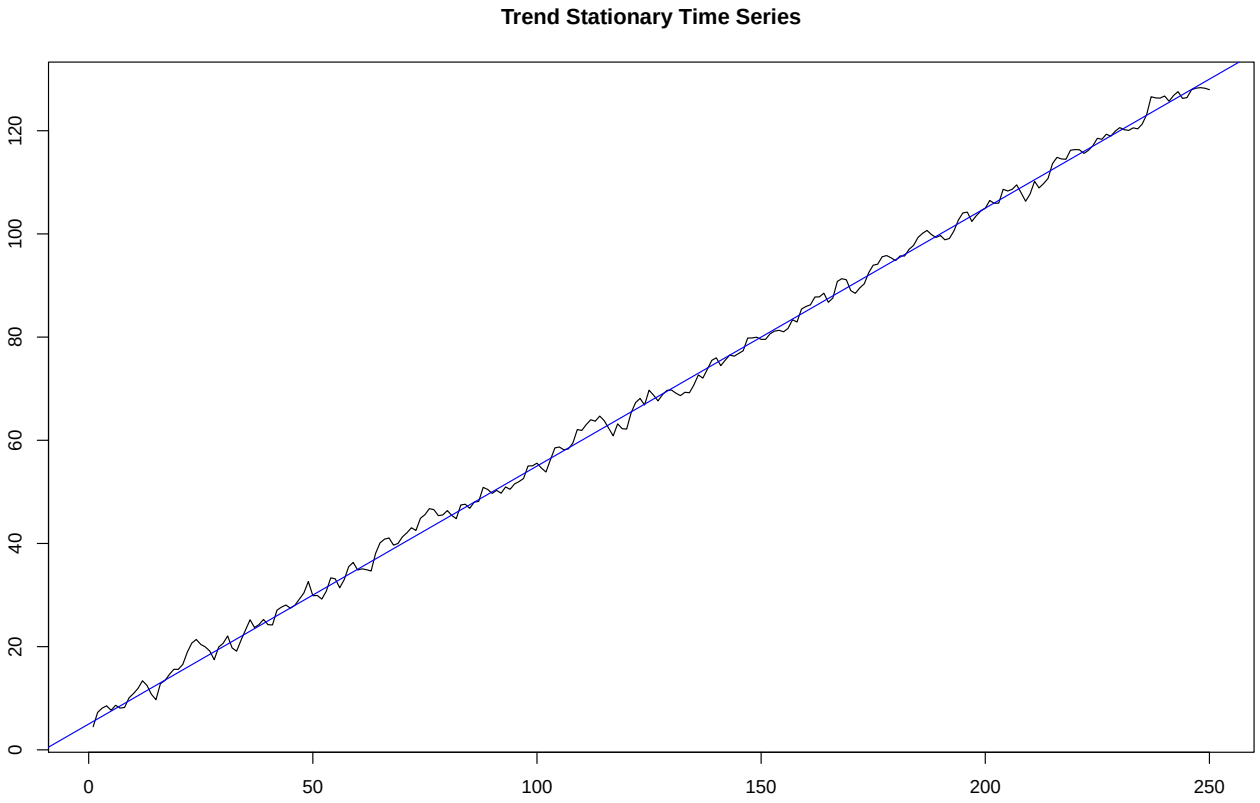
$$y_t = TD_t + Z_t$$
$$TD_t = \beta_1 + \beta_2.t$$
$$\phi(L)Z_t = \theta(L)\varepsilon_t, \varepsilon_t \sim WN(0, \delta^2)$$
$$\phi(L) = 1 - \phi_1 L - ... - \phi_p L^p$$
$$\theta(L) = 1 + \theta_1 L + ... + \theta_q L^q$$

En supposant que $\phi(Z) = 0$ a au plus une racine unitaire complexe autour du cercle et $\theta(Z) = 0$ a toutes les racines unitaire en dehors du cercle.

Série Temporelle Stationnaire en tendance

La série y_t est stationnaire si les racines de $\phi(z) = 0$ sont en dehors du cercle unitaire. Nous donnons ici l'exemple d'une série temporelle stationnaire en tendance.

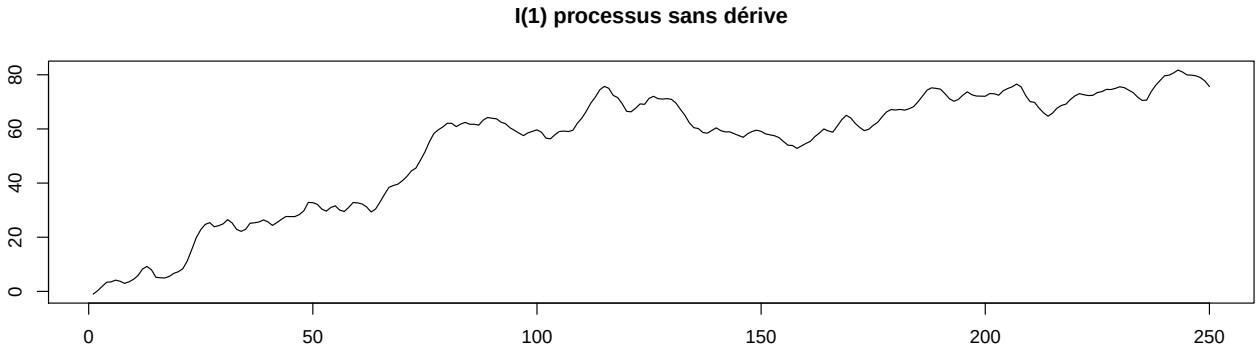
```
set.seed(12345)
x.ar2 = 5 + 0.5 * seq(250) + arima.sim(list(ar = c(0.8, -0.2)), n = 250)
plot(x.ar2, ylab = "", xlab = "", main = "Trend Stationary Time Series")
abline(a = 5, b = 0.5, col = "blue")
```

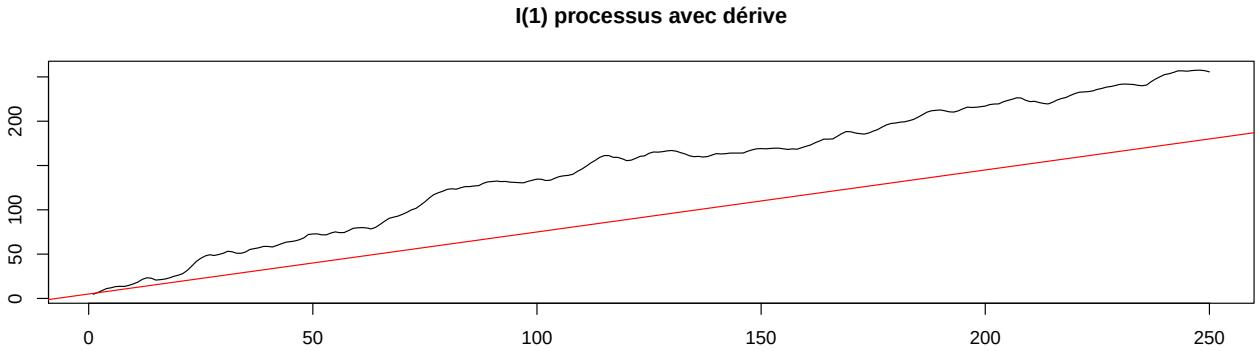


Séries Temporelles Stationnaires en Différence

La série y_t est stationnaire en différence si $\phi(z) = 0$ a une racine unitaire sur le cercle des unités et les autres sont en dehors du cercle des unités. Nous présentons ici la stationnarité en différence d'une série temporelle.

```
set.seed(12345)
u.ar2 <- arima.sim(list(ar = c(0.8, -0.2)), n = 250)
x1 <- cumsum(u.ar2)
TD <- 5.0 + 0.7 * seq(250)
x1.d <- x1 + TD
layout(matrix(1:2, nrow = 2, ncol = 1))
plot.ts(x1, main = "I(1) processus sans dérive", ylab="", xlab = "")
plot.ts(x1.d, main = "I(1) processus avec dérive", ylab = "", xlab = "")
abline(a = 5, b = 0.7, col = "red")
```





Considérons la décomposition suivante en cycle de tendance d'une série temporelle y_T:

$$y_t = TD_t + Z_t = TD_t + TS_t + C_t$$

Où TD_t signifie la tendance déterministe, TS_t est la tendance stochastique et C_t est un composant stationnaire.

i. Test de racine unitaire:$H_0 : TS_t \neq 0, H_1 : TS_t = 0, y_t \sim I(1) et y_t \sim I(0)$;
ii. Test de stationnarité : $H_0 : TS_t = 0, H_1 : TS_t \neq 0, y_t \sim I(0) et y_t \sim I(0)$.

Les tests sont basés sur le cadre suivant :

$$y_t = \phi y_{t-1} + \mu_t, \mu_t \sim I(0)$$

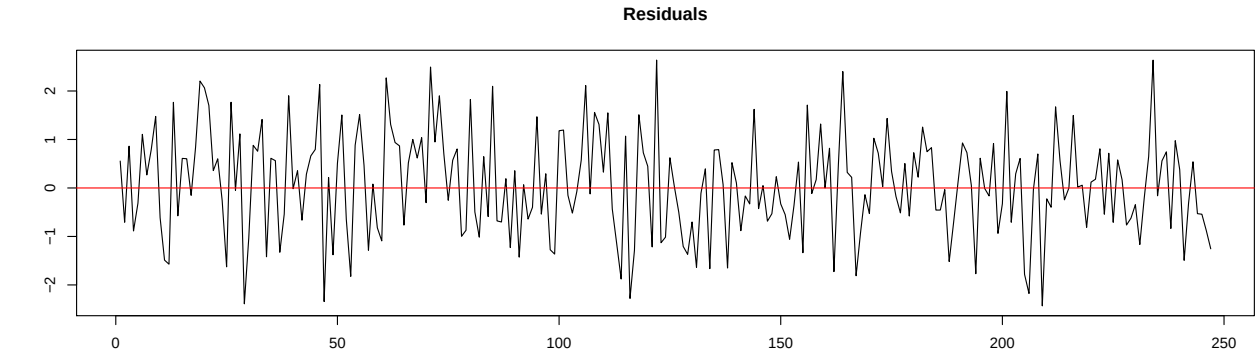
Avec comme :

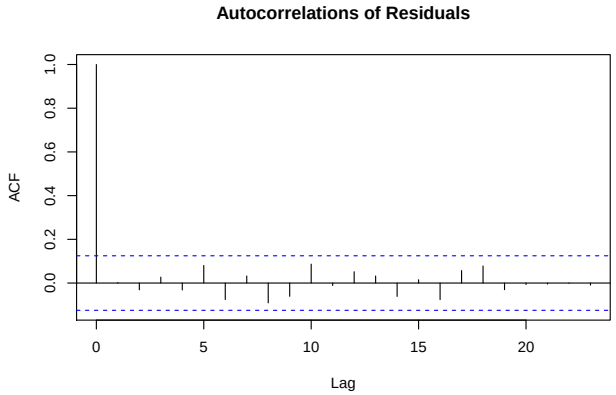
i. $H_0 : \phi = 1, H_1 : |\phi| < 1$
ii. Test de ADF : La corrélation sérial dans μ_t est capturée par structure paramétrique autorégressive du test.
iii. PP test : test non paramétrique, basée sur la variance estimée à long terme de Δy_t.

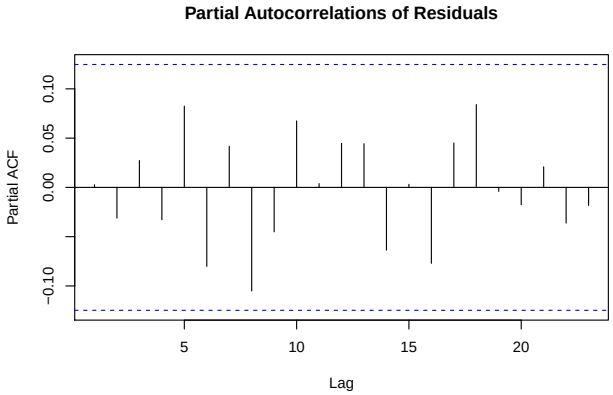
Test de racine unitaire

1. ADF test

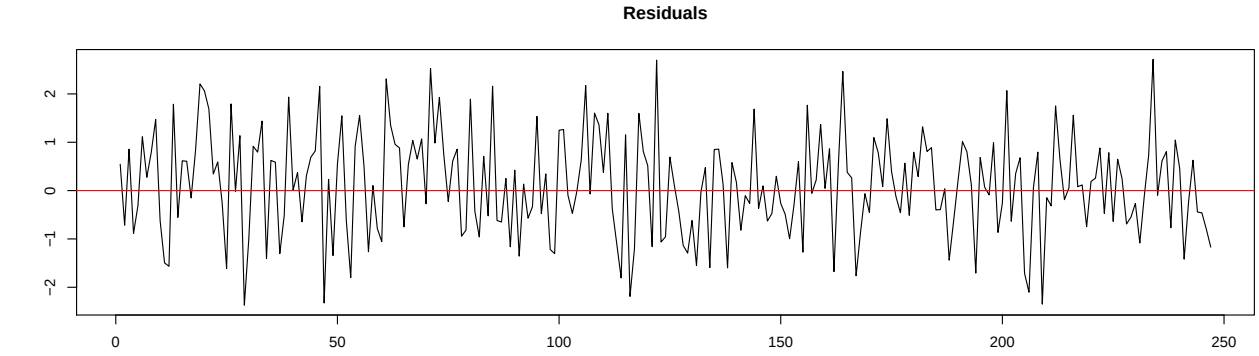
```
x1.adf = ur.df(x1, type = "none", lags = 2)
dx1.adf <- ur.df(diff(x1), type = "none", lags = 1)
plot(x1.adf)
```

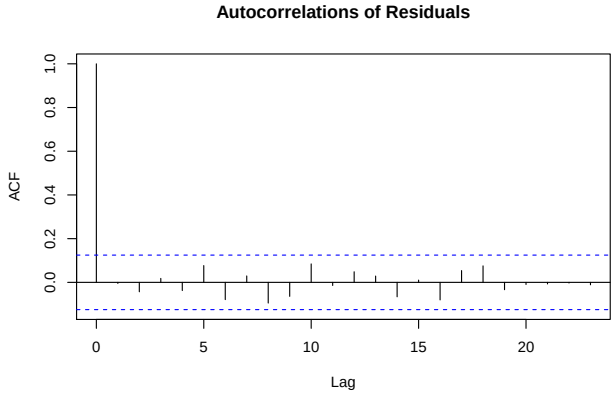

Residuals
2
1
0
−1
−2
0
50
100
150
200
250

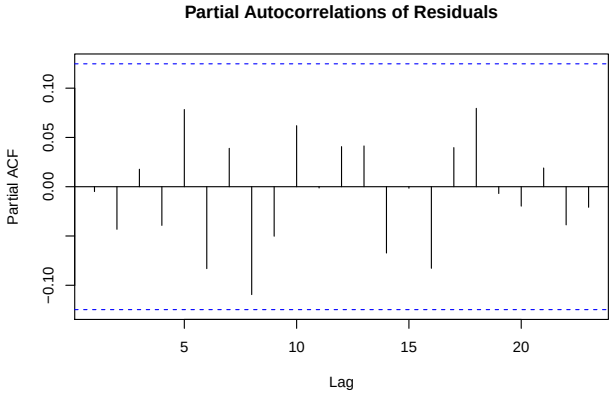

Autocorrelations of Residuals
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
Lag


Partial Autocorrelations of Residuals
Partial ACF
0.10
0.05
0.00
−0.10
5
10
15
20
Lag

```
plot(dx1.adf)
```


Residuals
2
1
0
−1
−2
0
50
100
150
200
250


Autocorrelations of Residuals
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
Lag


Partial Autocorrelations of Residuals
Partial ACF
0.10
0.05
0.00
−0.10
5
10
15
20
Lag

```
summary(x1.adf)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.43193 -0.65470  0.05685  0.74682  2.63766
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## z.lag.1      0.001010   0.001186   0.852 0.395321
## z.diff.lag1  0.802214   0.062460  12.844  < 2e-16 ***
## z.diff.lag2 -0.231898   0.062690  -3.699 0.000267 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.062 on 244 degrees of freedom
## Multiple R-squared:  0.4606, Adjusted R-squared:  0.454
## F-statistic: 69.45 on 3 and 244 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: 0.8515
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

```
summary(dx1.adf)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3710 -0.6189  0.1153  0.8045  2.7144
##
## Coefficients:
```

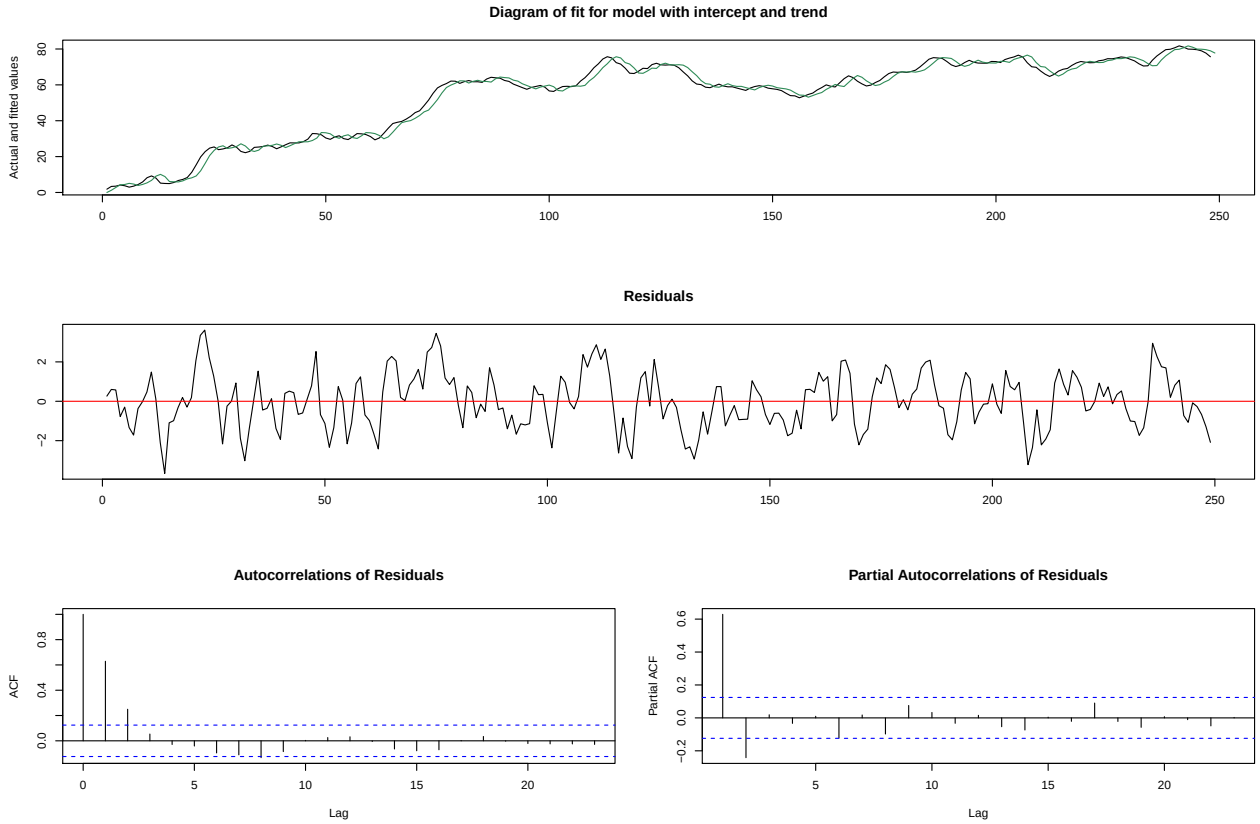
```
##             Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.42131    0.05173  -8.144 1.95e-14 ***
## z.diff.lag  0.22647    0.06233   3.633 0.000341 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.061 on 245 degrees of freedom
## Multiple R-squared:  0.2132, Adjusted R-squared:  0.2068
## F-statistic:  33.2 on 2 and 245 DF,  p-value: 1.742e-13
##
##
## Value of test-statistic is: -8.144
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

2. PP test

Ce test de Phillips et Perron est basé sur :

$$\Delta y_t = \beta^{"} D_t + \pi y_{t-1} + \mu_t, \mu_t \sim I(0)$$

```
x1.pp.ts <- ur.pp(x1, type = "Z-tau", model = "trend", lags = "short")
dx1.pp.ts <- ur.pp(diff(x1), type = "Z-tau", model = "trend", lags = "short")
plot(x1.pp.ts)
```



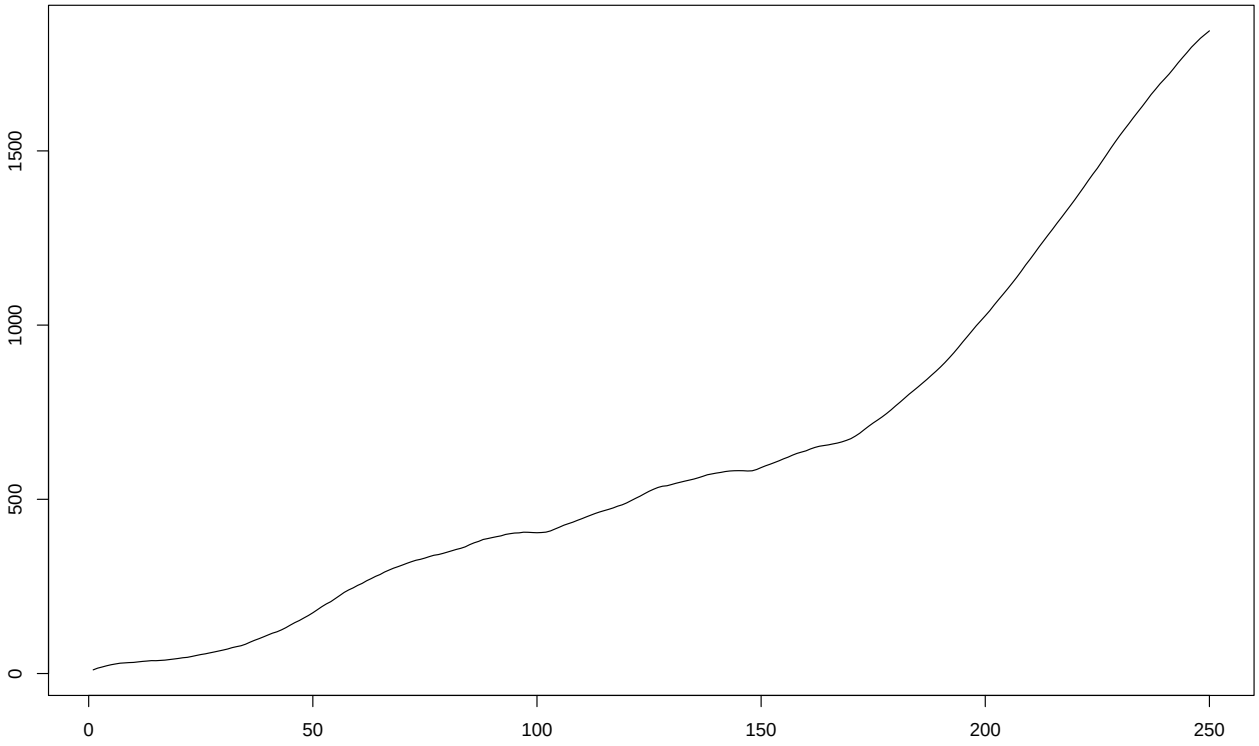
```
#plot(dx1.pp.ts)
#summary(x1.pp.ts)
#summary(dx1.pp.ts)
```

Les tests ADF et PP sont asymptotiquement équivalents. Le test de PP a de meilleures propriétés d'échantillonnage que l'ADF. Les deux ont une faible puissance contre $I(0)$ alternatives proches $I(1)$ processus. La puissance des tests diminue à mesure que les termes déterministes sont ajouté à la régression.

3. Elliot, Rothenberg et Stock

$$y_t = d_t + \mu_t$$
$$\mu_t = au_{t-1} + v_t$$

```
set.seed(12345)
u.ar1 <- arima.sim(list(ar = 0.99), n = 250)
TD <- 5.0 + 0.7 * seq(250)
x1.ni <- cumsum(u.ar1) + TD
x1.ers <- ur.ers(x1.ni, type = "P-test", model = "trend", lag = 1)
x1.adf <- ur.df(x1.ni, type = "trend")
plot.ts(x1.ni, xlab = "", ylab = "")
```




```
summary(x1.ers)
```

```
##
## ###############################################
## # Elliot, Rothenberg and Stock Unit Root Test #
## ###############################################
##
## Test of type P-test
## detrending of series with intercept and trend
```

```
##
## Value of test-statistic is: 33.7996
##
## Critical values of P-test are:
##                 1pct 5pct 10pct
## critical values 3.96 5.62  6.89
```

```
summary(x1.adf)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.83495 -0.70551  0.06655  0.69792  2.73869
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.1190123  0.1581242  -0.753   0.4524
## z.lag.1     -0.0006921  0.0004932  -1.403   0.1618
## tt           0.0060322  0.0028634   2.107   0.0362 *
## z.diff.lag   0.9747618  0.0210545  46.297   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.011 on 244 degrees of freedom
## Multiple R-squared:  0.9677, Adjusted R-squared:  0.9673
## F-statistic:  2436 on 3 and 244 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -1.4033 2.3554 2.63
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.99 -3.43 -3.13
## phi2  6.22  4.75  4.07
## phi3  8.43  6.49  5.47
```

4. Test de Schmidt et Phillips

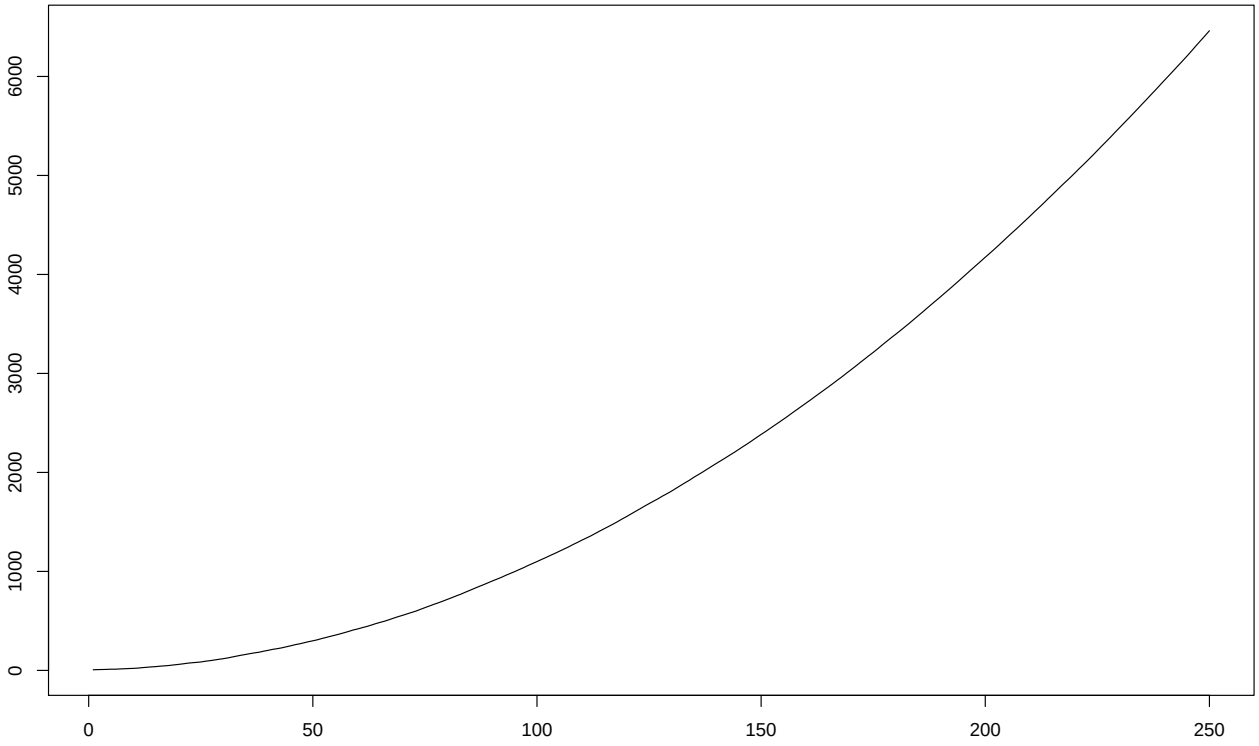
Problème des tests de DF type: les paramètres de nuisance, c'est-à-dire les coefficients des régresseurs déterministes, ne sont pas définis ou ont une interprétation différente sous l'hypothèse alternative de la stationnarité.

Pour résoudre ce problème, la solution est d'effectuer le test de type LM, qui a le même ensemble de paramètres de nuisance sous l'hypothèse nulle et alternative. Les polynômes plus élevés qu'une tendance linéaire sont alors autorisés. Le modèle se présente ainsi :

$$y_t = \alpha + Z_t\delta + x_t, x_t = \pi x_{t-1} + \varepsilon_t$$

Le test de regréssion est alors : $\Delta y_t = \Delta Z_t\gamma + \phi S_{t-1} + v_t$

```
set.seed(12345)
y1 <- cumsum(rnorm(250))
TD <- 5.0 + 0.7 * seq(250) + 0.1 * seq(250)^2
y1.d <- y1 + TD
plot.ts(y1.d, xlab = "", ylab = "")
```




```
y1.d.sp <- ur.sp(y1.d, type = "tau", pol.deg = 2, signif = 0.05)
summary(y1.d.sp)
```

```
##
## ###################################
## # Schmidt-Phillips Unit Root Test #
## ###################################
##
##
## Call:
## lm(formula = sp.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.43771 -0.67937  0.00849  0.68706  2.73219
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.963666   0.206578   4.665 5.08e-06 ***
## y.lagged    0.939952   0.022784  41.255  < 2e-16 ***
```

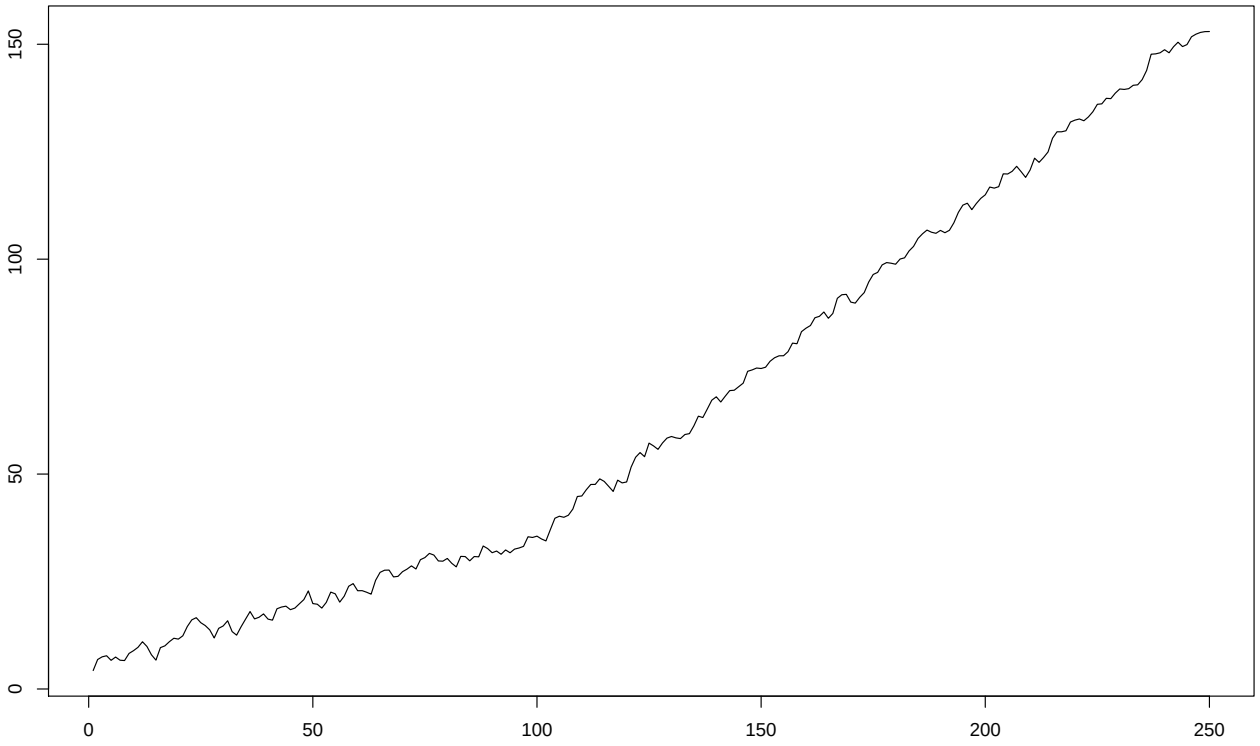
```
## trend.exp1  0.245991   0.018581  13.239  < 2e-16 ***
## trend.exp2  0.005966   0.002262   2.638  0.00888 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.043 on 245 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 2.827e+08 on 3 and 245 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -2.5292
## Critical value for a significance level of 0.05
## is: -3.55
```

5. Test de Zivot et Andrews

Problème: il est difficile de distinguer statistiquement une série $I(1)$ d'une série stable $I(0)$ qui est contaminée par un décalage structurel. Si le point de rupture est connu: on utilise les tests de Phillips et Perron et Vogelsang. Mais risque d'exploration de données si le point de rupture est déterminé de manière exogène.

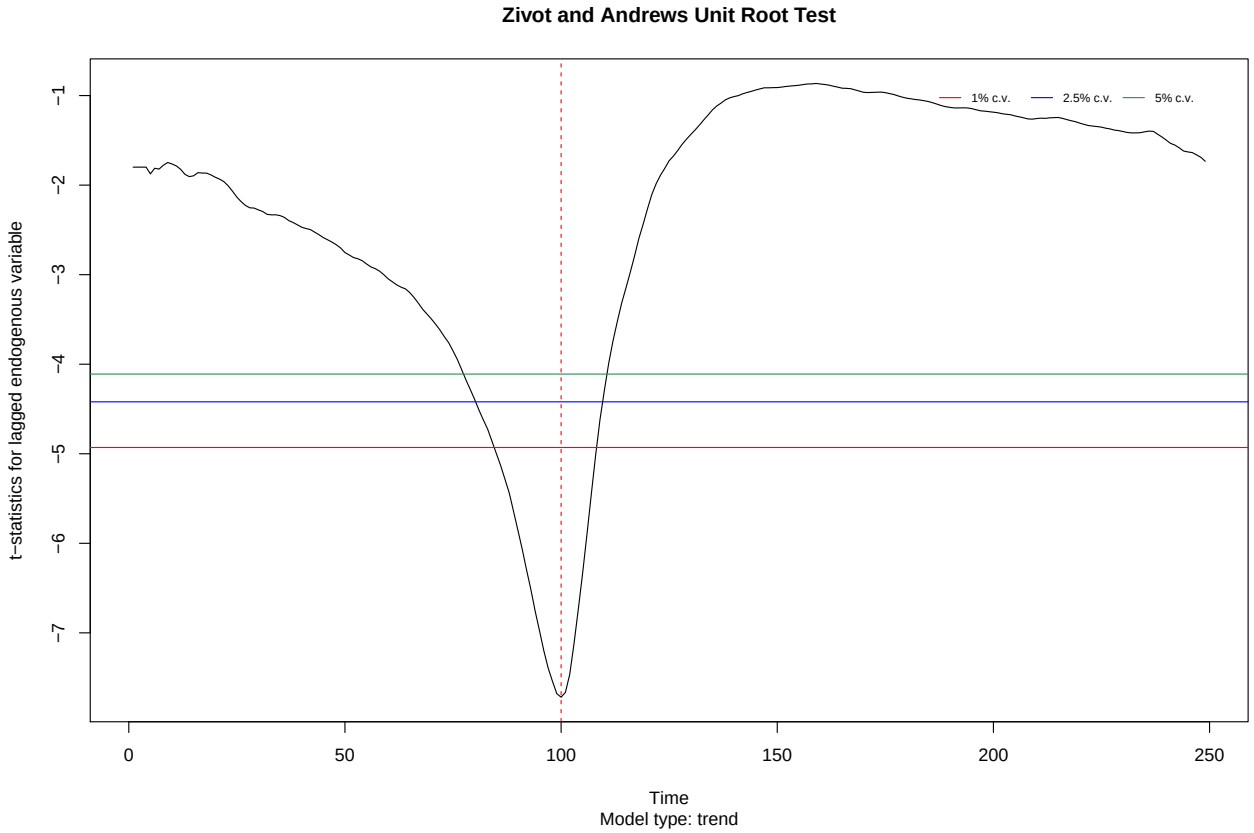
Solution: il faut donc déterminer de manière endogène le point de rupture potentiel; d'où on fait appel au test Zivot et Andrews.

```
set.seed(12345)
u.ar2 <- arima.sim(list(ar = c(0.8, -0.2)), n = 250)
TD1 <- 5 + 0.3 * seq(100)
TD2 <- 35 + 0.8 * seq(150)
TD <- c(TD1, TD2)
y1.break <- u.ar2 + TD
plot.ts(y1.break, xlab = "", ylab = "")
```

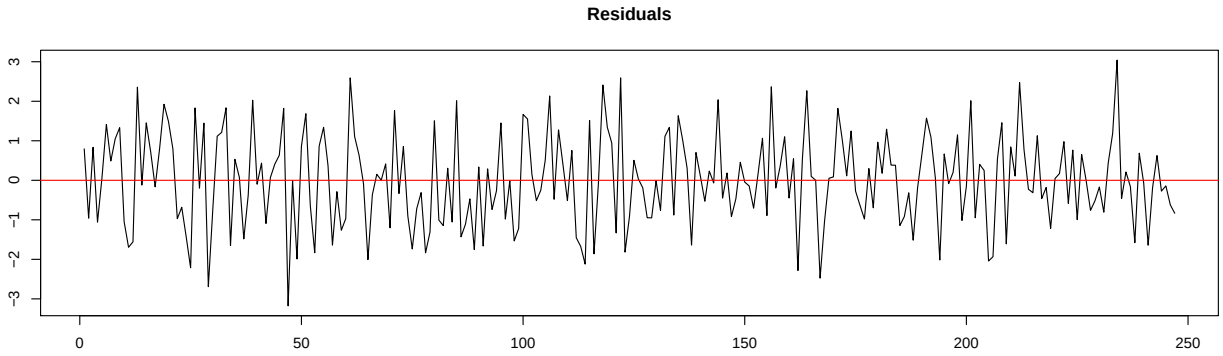


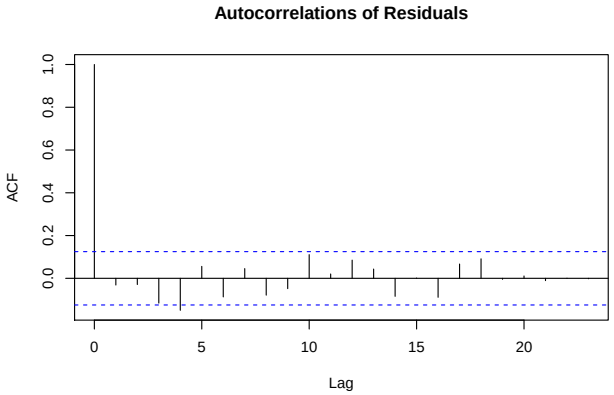

```
y1.break.za <- ur.za(y1.break, model = "trend", lag = 2)
```

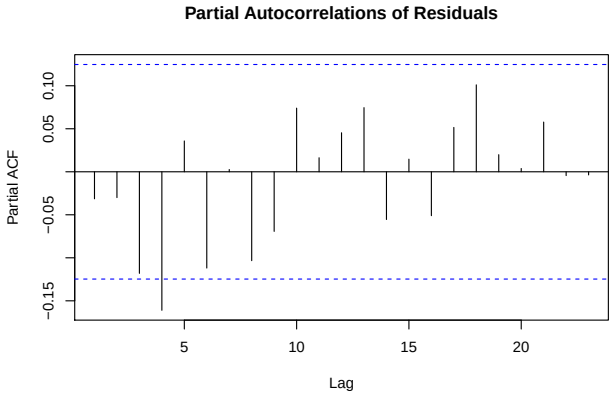
```
plot(y1.break.za)
```




```
y1.break.df <- ur.df(y1.break, type = "trend", lags = 2)
plot(y1.break.df)
```







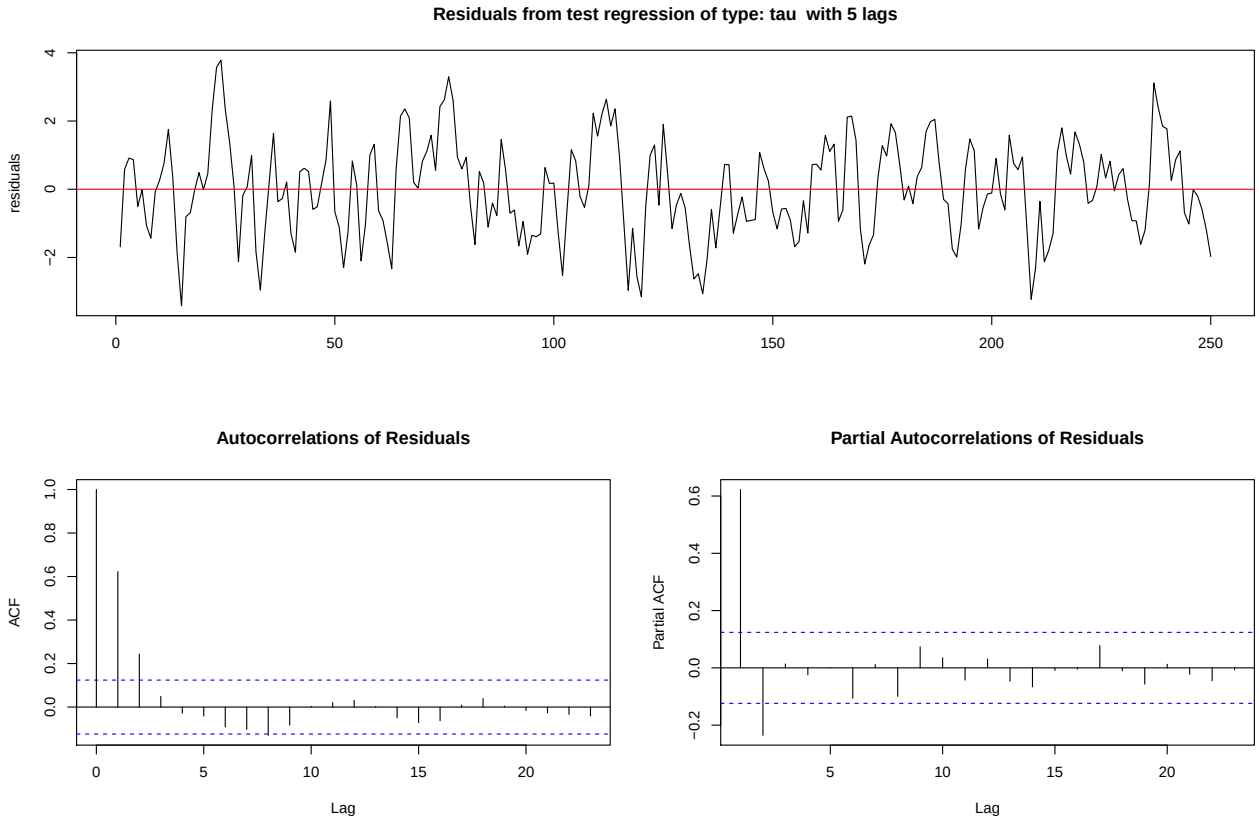
Test de stationnarité

Le modèle se présente comme suit :

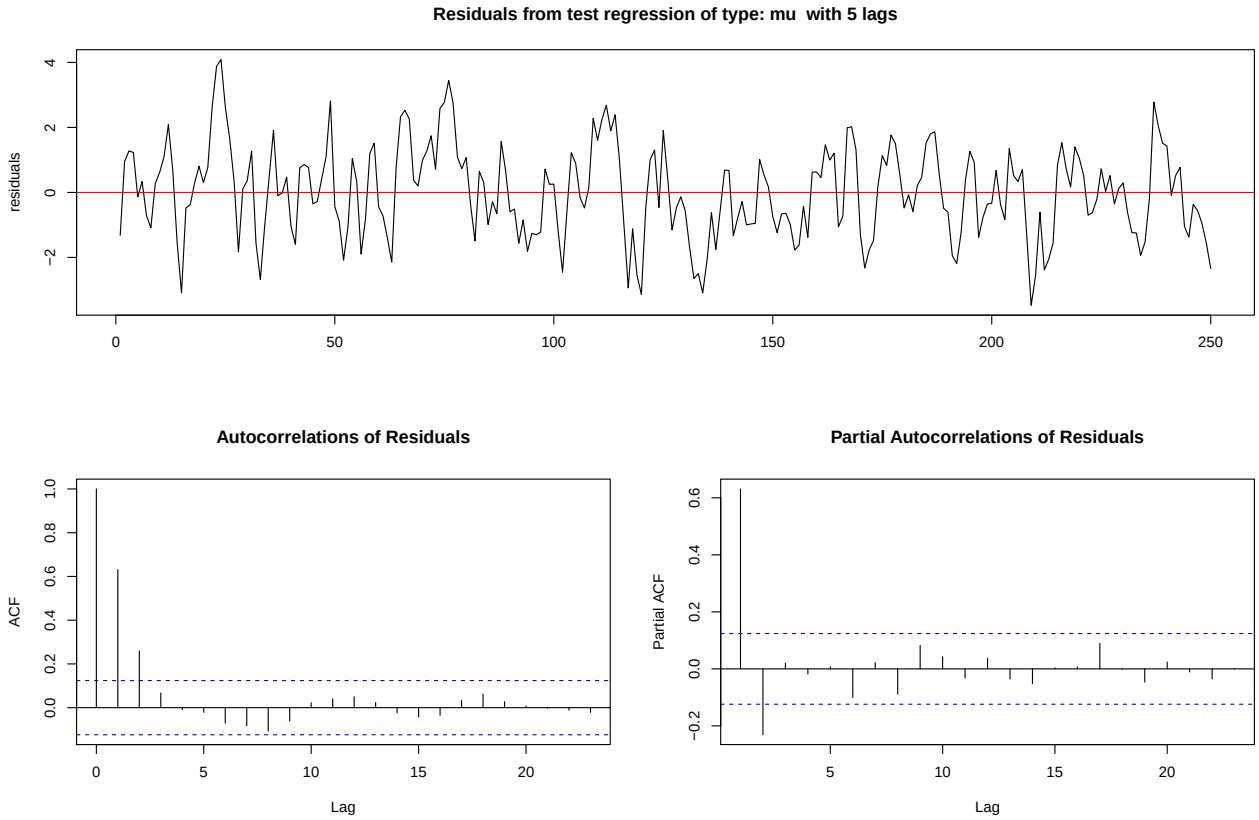
$$y_t = \beta D_t + \mu_t, \mu_t \sim I(0)$$
$$\mu_t = \mu_{t-1} + \varepsilon_t, \varepsilon_t \sim WN(0, \delta^2)$$

Avec comme hypothèse nulle et alnertative : $H_0 : \delta_\varepsilon^2 = 0, H_1 : \delta_\varepsilon^2 > 0$

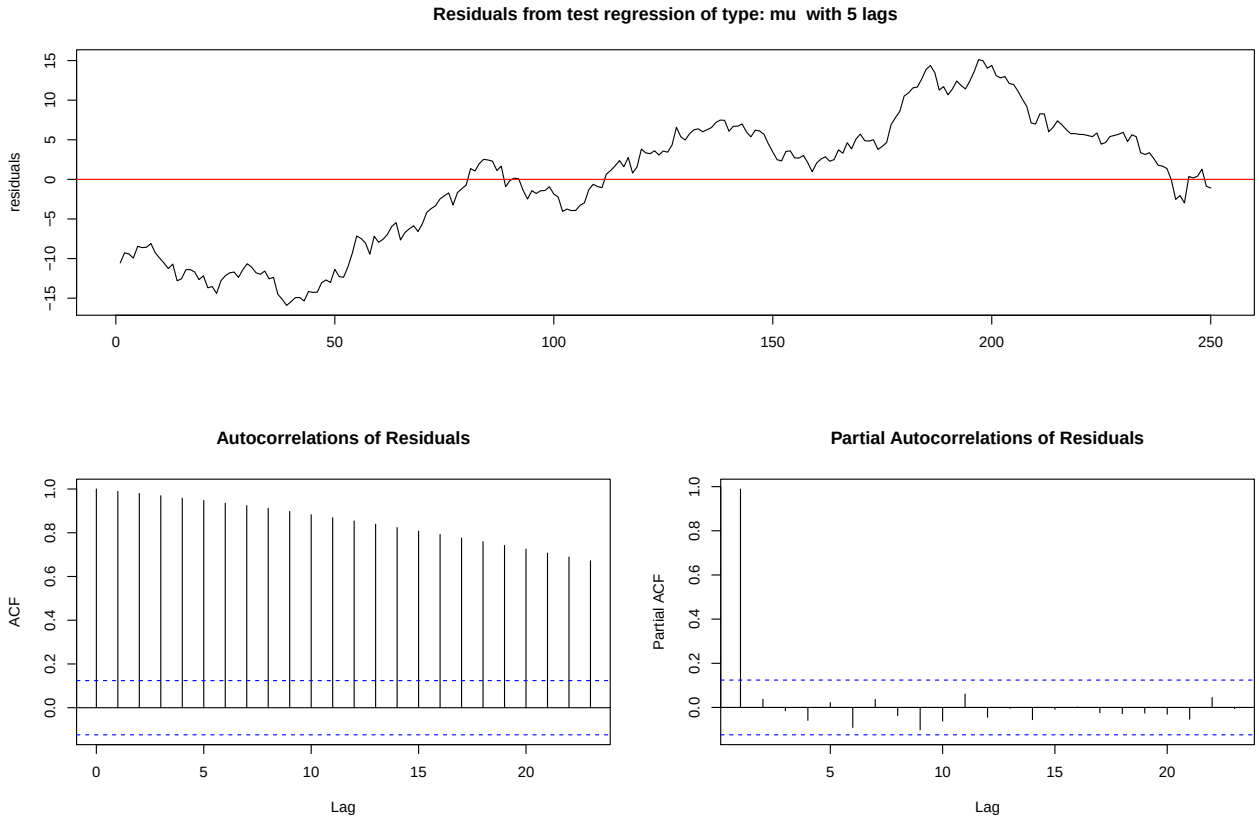
```
set.seed(12345)
u.ar2 = arima.sim(list(ar = c(0.8, -0.2)), n = 250)
TD1 = 5 + 0.3 * seq(250)
TD2 = rep(3, 250)
y1.td1 = u.ar2 + TD1
y1.td2 = u.ar2 + TD2
y2.rw = cumsum(rnorm(250))
y1td1.kpss = ur.kpss(y1.td1, type = "tau")
y1td2.kpss = ur.kpss(y1.td2, type = "mu")
y2rw.kpss = ur.kpss(y2.rw, type = "mu")
plot(y1td1.kpss)
```




```
plot(y1td2.kpss)
```



```
plot(y2rw.kpss)
```



Il est aussi possible d'effectuer les testsde normalité de résidus et le test d'hétéroscédasticité. Mais nous le verrons plus tard dans les séries multivariées.

Séries temporelles multivariées

i. Stationary VAR(p)-models
ii. SVAR models
iii. Cointegration: Concept, models and methods
iv. SVEC models

1 Modèle VAR

Le processus VAR(p) est défini comme:

$$y_t = A_1 y_{t-1} + ... + A_p y_{t-p} + CD_t + \mu_t$$

Où A_1 est la matrice de coefficients pour $i = 1, 2, ..., p$, μ_t est le processus *bruit blanc* de dimension K avec matrice de covariance définie positive invariante dans le temps $E(\mu_t \mu_t^{"}) = \sum_{\mu}$, C est la matrice de coefficients de régresseurs potentiellement déterministes et D_t un vecteur colonne tenant les régresseurs déterministe approprié régresseurs.

1 Sélection empirique de l'ordre de décalage

$$AIC(p) = log\det(\sum_{\mu}(p)) + \frac{2}{T}pK^2$$

$$HQ(p)log\det(\sum_{\mu}(p)) + \frac{2log(log(T))}{T}pK^2$$

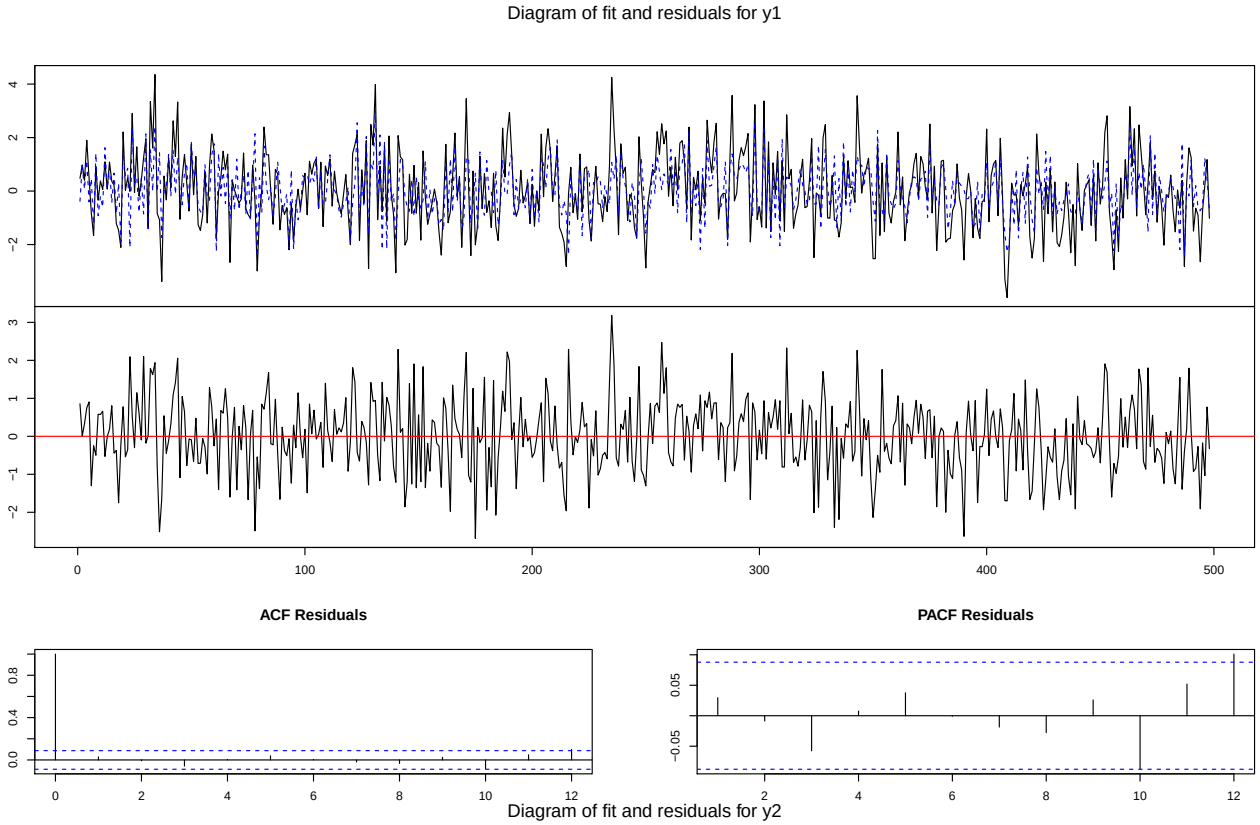
$$SC(p) = log\det(\sum_{\mu}(p)) + \frac{log(T)}{T}pK^2$$

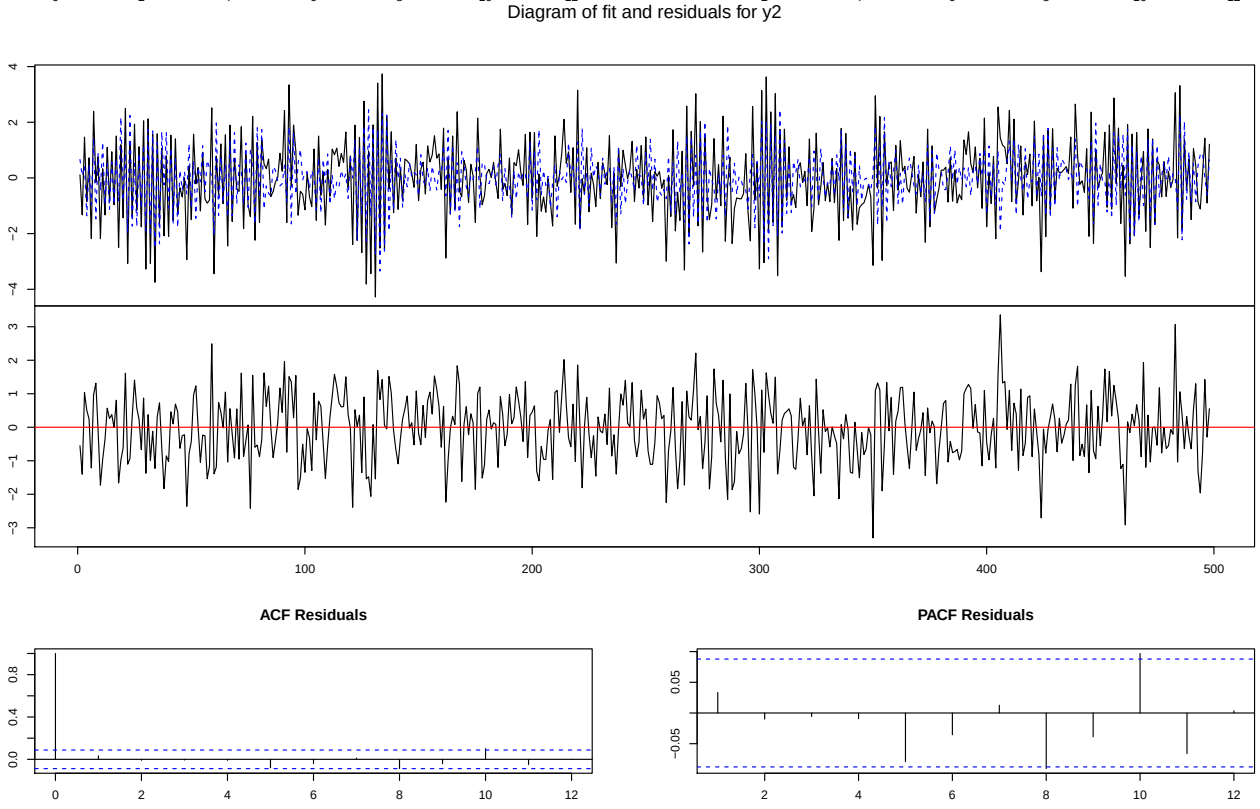
$$FPE(p) = (\frac{T+p^*}{T-p^*})\det(\sum_{\mu}(p))$$

Avec $\sum_{\mu}(p) = T^{-1}\sum_{t=1}^{T}\hat{\mu_t}\hat{\mu_t'}$ est le nombre total de paramètres dans chaque équation et p attribue l'ordre de décalage.

La simulation de processus VAR avec le package dse1 de R et l'estimation de processus VAR avec les packages dse1 et vars.

```
library(dse1)
library(vars)
Apoly = array(c(1.0, -0.5, 0.3, 0, 0.2, 0.1, 0, -0.2, 0.7, 1, 0.5, -0.3) , c(3, 2, 2))
B = diag(2)
var2 <- ARMA(A = Apoly, B = B)
varsim <- simulate(var2, sampleT = 500, noise = list(w = matrix(rnorm(1000),
                                                                nrow = 500, ncol = 2)),
                   rng = list(seed = c(123456)))
vardat = matrix(varsim$output, nrow = 500, ncol = 2)
colnames(vardat) = c("y1", "y2")
infocrit <- VARselect(vardat, lag.max = 3, type = "const")
varsimest <- VAR(vardat, p = 2, type = "none")
roots <- roots(varsimest)
plot(varsimest)
```






```
summary(varsimest)
```

```
##
```

```
## VAR Estimation Results:
## ========================
## Endogenous variables: y1, y2
## Deterministic variables: none
## Sample size: 498
## Log Likelihood: -1408.104
## Roots of the characteristic polynomial:
## 0.8446 0.6629 0.5681 0.5681
## Call:
## VAR(y = vardat, p = 2, type = "none")
##
##
## Estimation results for equation y1:
## ===================================
## y1 = y1.l1 + y2.l1 + y1.l2 + y2.l2
##
##       Estimate Std. Error t value Pr(>|t|)
## y1.l1  0.49544    0.03656  13.552  < 2e-16 ***
## y2.l1  0.14658    0.04037   3.631 0.000312 ***
## y1.l2 -0.27875    0.03639  -7.660 9.92e-14 ***
## y2.l2 -0.75704    0.04550 -16.638  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.9864 on 494 degrees of freedom
## Multiple R-Squared: 0.5167,  Adjusted R-squared: 0.5128
## F-statistic:   132 on 4 and 494 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation y2:
## ===================================
## y2 = y1.l1 + y2.l1 + y1.l2 + y2.l2
##
##       Estimate Std. Error t value Pr(>|t|)
## y1.l1 -0.20760    0.03749  -5.538 4.99e-08 ***
## y2.l1 -0.48993    0.04140 -11.834  < 2e-16 ***
## y1.l2 -0.11444    0.03732  -3.067  0.00228 **
## y2.l2  0.33754    0.04666   7.234 1.80e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 1.011 on 494 degrees of freedom
## Multiple R-Squared: 0.5275,  Adjusted R-squared: 0.5237
## F-statistic: 137.9 on 4 and 494 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##         y1      y2
## y1 0.97283 0.00246
## y2 0.00246 1.02185
##
```

```
## Correlation matrix of residuals:
##          y1       y2
## y1 1.000000 0.002467
## y2 0.002467 1.000000
```

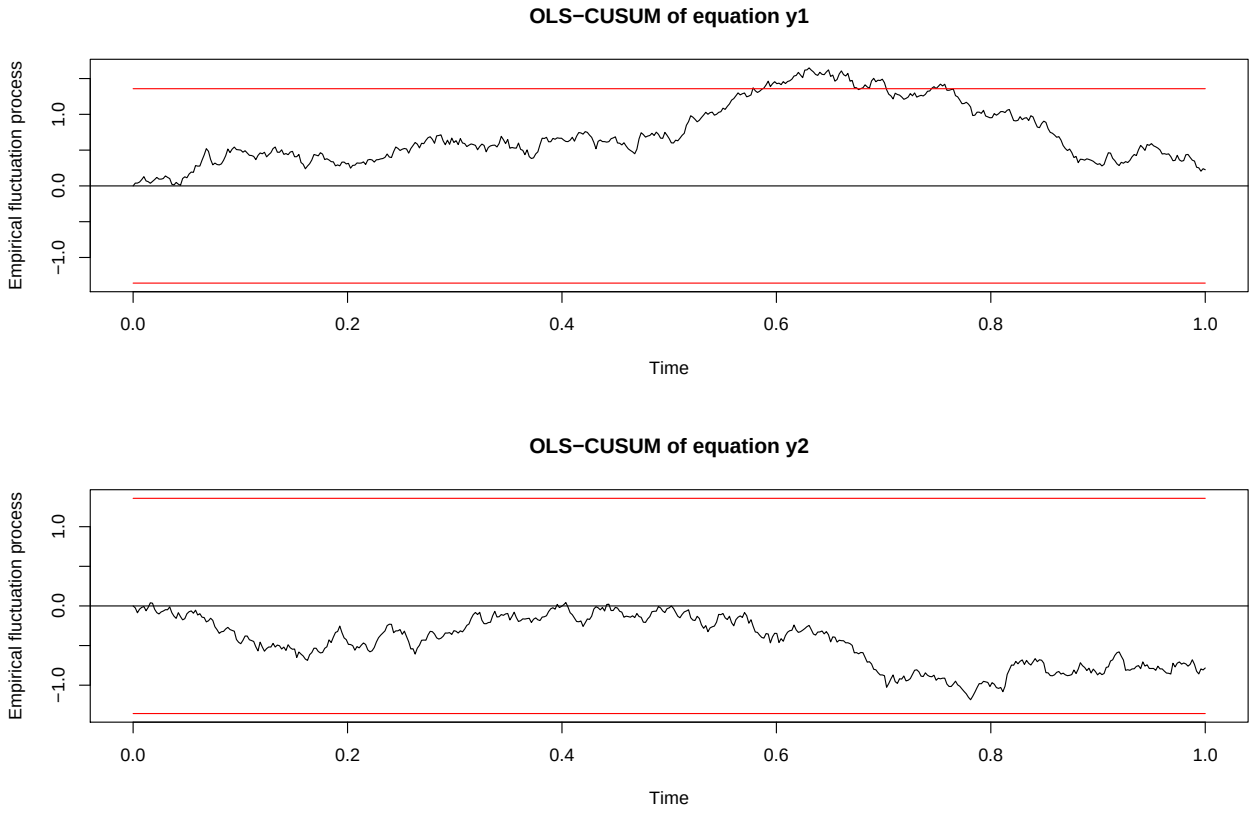
Le tableau ci-haut nous donne les résultats du modèle VAR(2) simulé (voir graphique) et du VAR estimé (voir tableau). Pour les deux estimations (y1 et y2) on remarque les coefficients des modèles estimés sont statistiquement significatifs. Plus bas du tableau des résultats, on a la matrice de covariance et la matrice de corrélations résiduelles.

```
print(infocrit)
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      3      2      2      3
##
## $criteria
##                1          2          3
## AIC(n) 0.6115307 0.02014260 0.01922390
## HQ(n)  0.6314727 0.05337937 0.06575539
## SC(n)  0.6623386 0.10482248 0.13777573
## FPE(n) 1.8432512 1.02034821 1.01941367
```

Ci-haut représente le tableau de critère c'est-à-dire le choix du modèle à estimer et de selection empirique du décalage.

```
varsimest.stability = stability(varsimest, type = "OLS-CUSUM")
plot(varsimest.stability)
```



2 Diagnostique de tests

i. Corrélation de série: test de Portmanteau, Breusch et Godfrey;
ii. Hétéroscédasticité: test de ARCH;
iii. Normalité: tests de Jarque et Bera, de Skewness et de Kurtosis;
iv. Stabilité structurelle: tests de EFP, CUSUM, CUSUM-of-Squares, Test de fluctuation, etc.

Nous allons à présent montrer comment le diagnostique des tests est obtenu sur R avec l'estimation du modèle VAR(2) ci-haut.

```
var2c.serie = serial.test(varsimest)
print(var2c.serie)
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object varsimest
## Chi-squared = 52.673, df = 56, p-value = 0.6016
```

```
## $serial
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object varsimest
## Chi-squared = 52.673, df = 56, p-value = 0.6016
```

```
var2c.arch = arch.test(varsimest)
print(var2c.arch)
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object varsimest
## Chi-squared = 45.005, df = 45, p-value = 0.4717
```

```
## $arch.mul
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object varsimest
## Chi-squared = 45.005, df = 45, p-value = 0.4717
```

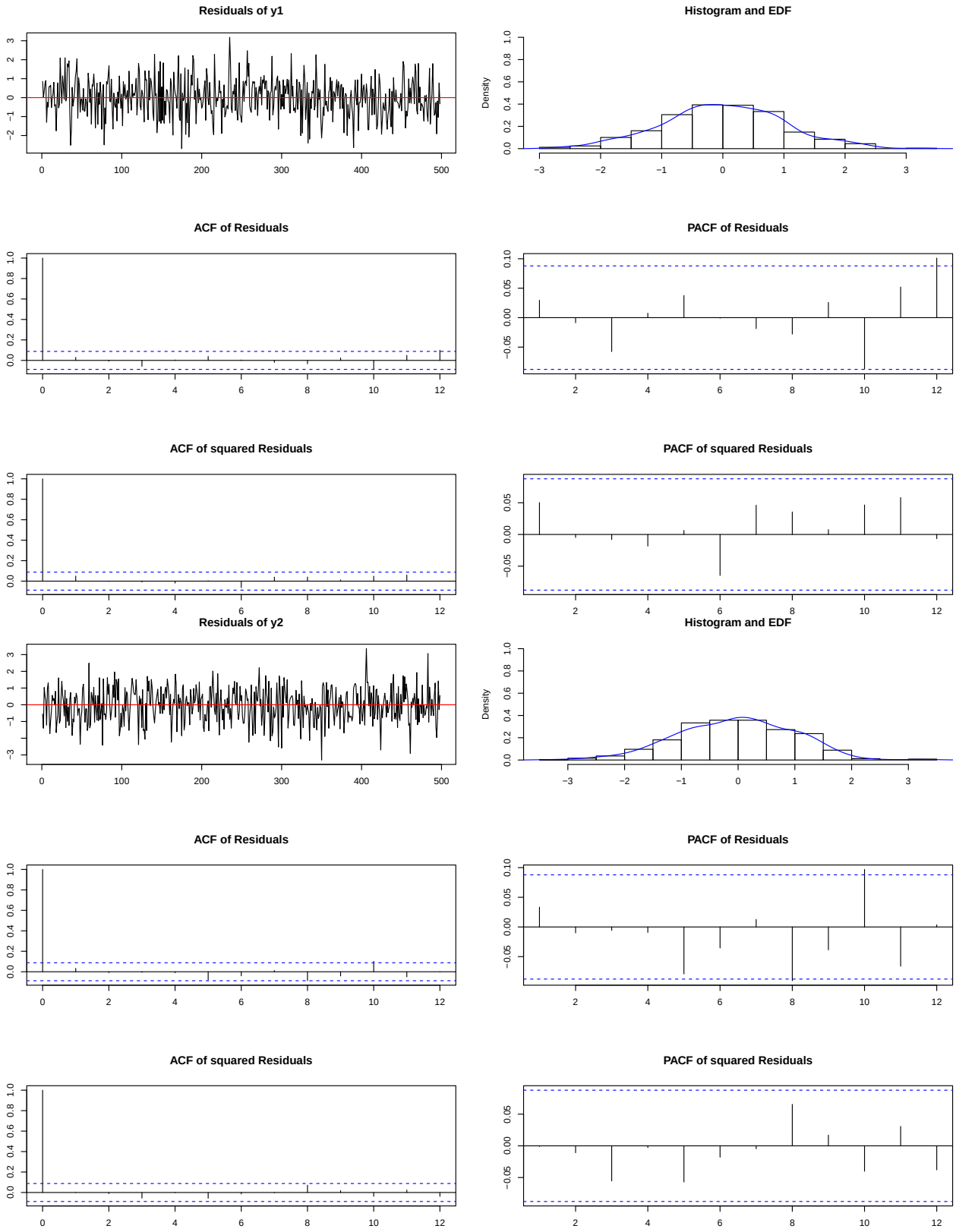
```
var2c.norm = normality.test(varsimest)
print(var2c.norm)
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object varsimest
## Chi-squared = 1.3687, df = 4, p-value = 0.8496
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object varsimest
```

```
## Chi-squared = 1.3397, df = 2, p-value = 0.5118
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object varsimest
## Chi-squared = 0.028939, df = 2, p-value = 0.9856
```

```
## $jb.mul
## $jb.mul$JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object varsimest
## Chi-squared = 1.3687, df = 4, p-value = 0.8496
##
##
## $jb.mul$Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object varsimest
## Chi-squared = 1.3397, df = 2, p-value = 0.5118
##
##
## $jb.mul$Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object varsimest
## Chi-squared = 0.028939, df = 2, p-value = 0.9856
```

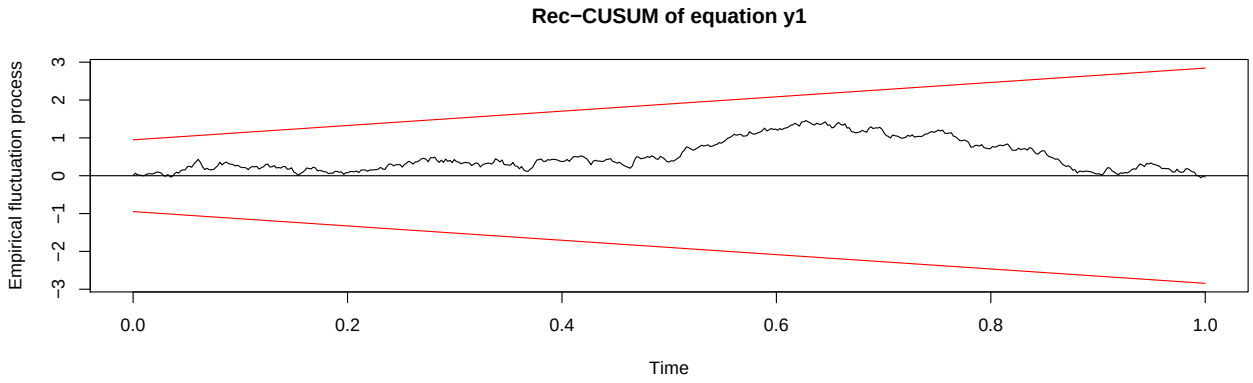
```
plot(var2c.serie)
```

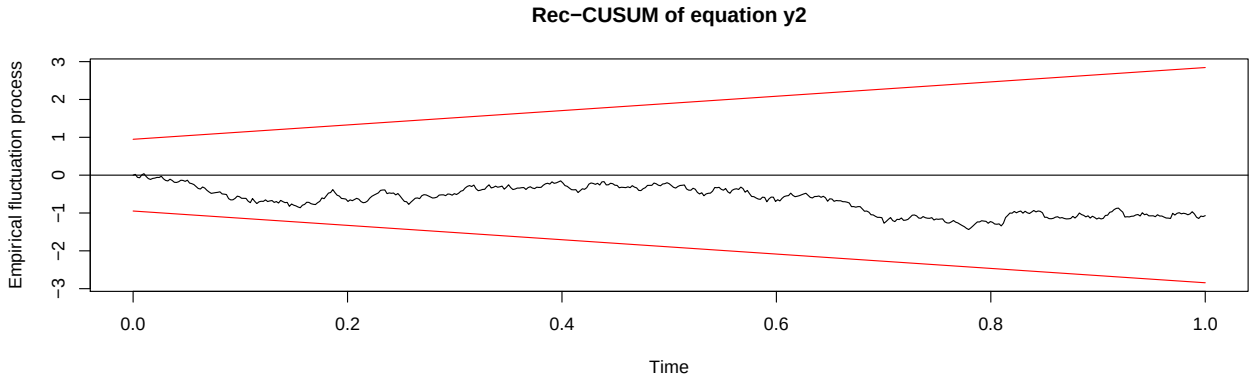


Ci-aut, sont présentés les tests de corrélation de séries, suivi du test d'hétéroscédasticité et enfin le test de normalité résiduelle. Notre estimation montre que les erreurs sont normalement distribuées et que l'hypothèse nulle ne peut être rejetée. Encore l'hypothèse nulle d'homoscédasticité est maintenue. Les p-values de nos

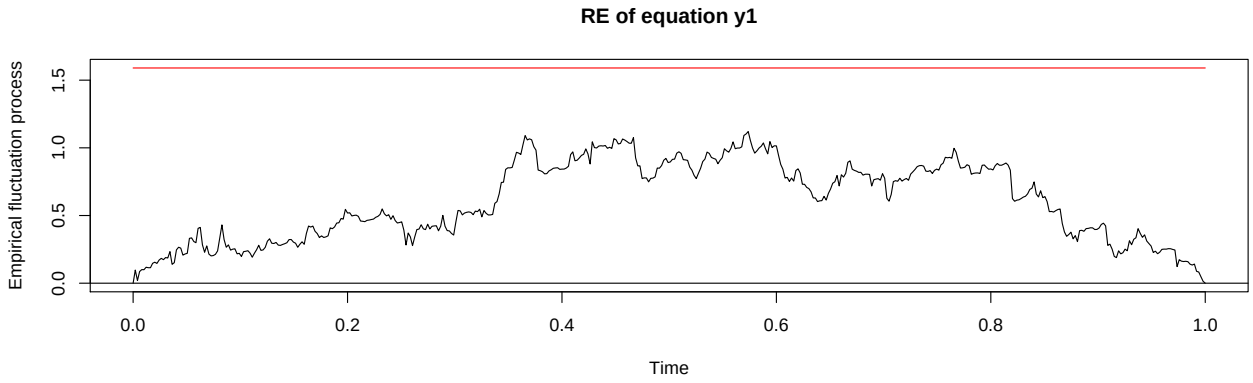
estimations sont bien supérieures au seuil de 5%. À présent, nous présentons le test de stabilité structurelle.

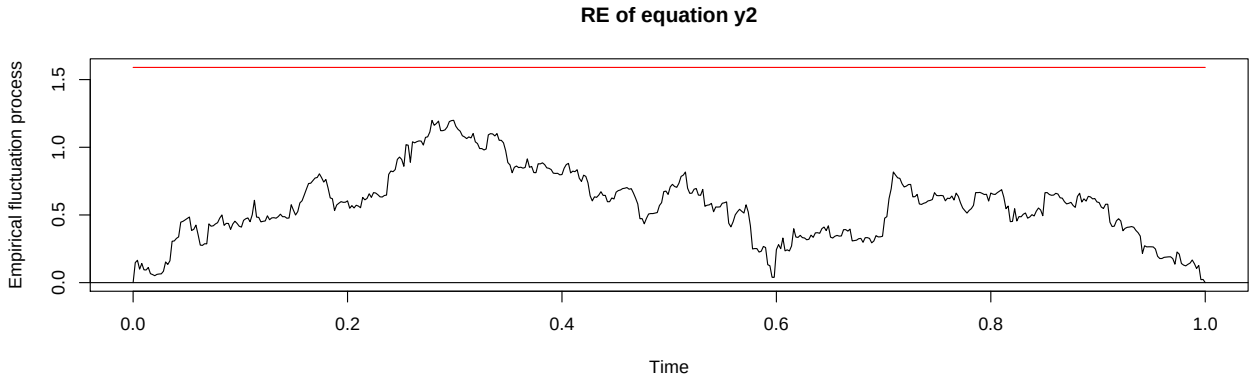
```
reccusum.stab = stability(varsimest, type = "Rec-CUSUM")
plot(reccusum.stab)
```






```
fluctuation.stab = stability(varsimest, type = "fluctuation")
plot(fluctuation.stab)
```





3 Causalités

Nous avons la causalité au sens de Granger et la causalité instantanée.

1. Causalité Granger :

$$y_{i,t} = \sum_{i=1}^{p} \alpha_{it} y_{i,t-1} + CD_t + \mu_{it}$$

Hypothèse nulle: le sousvecteur y_{1t} ne cause pas au sens de Granger y_{2t}, qui est défini par $\alpha_{21}, i = 0$ pour $i = 1, 2, ..., p$.

L'hypothèse alternative est: $\exists \alpha_{21}, i \neq 0$ pour $i = 1, 2, ..., p$.

Test statistique: $F(pK_1K_2, KT - n')$, avec n' égal au total de nombre de paramètres dans le processus VAR(p) ci-dessus, y compris les régresseurs déterministes.

2. Causalité instantanée :

L'hypothèse nulle pour la causalité non instantanée est définie comme suit: $C_\delta = 0$, où C est une matrice $(NxK(K+1)/2)$ de rang N, qui selectionne les co-variances pertinentes de μ_{1t} et μ_{2t}.

```
var.cause = causality(varsimest, cause = "y2")
print(var.cause)
```

```
## $Granger
##
##  Granger causality H0: y2 do not Granger-cause y1
##
## data:  VAR object varsimest
```

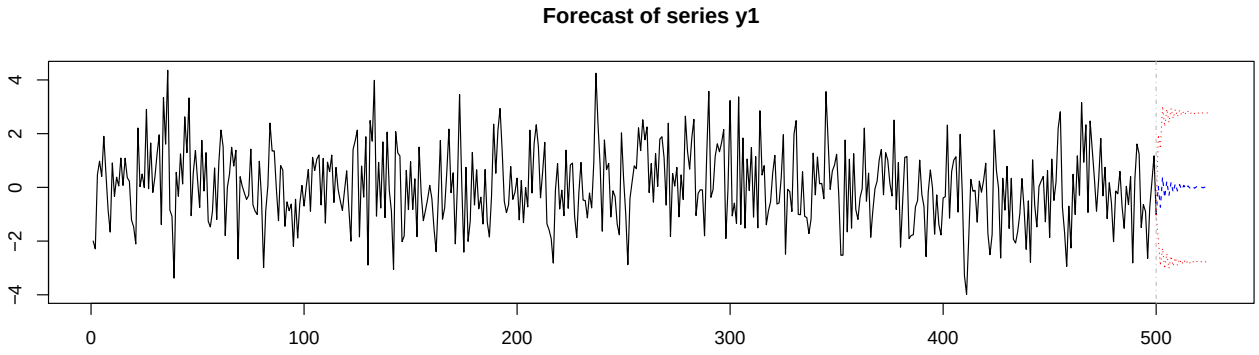
```
## F-Test = 254.53, df1 = 2, df2 = 988, p-value < 2.2e-16
##
##
## $Instant
##
##  H0: No instantaneous causality between: y2 and y1
##
## data:  VAR object varsimest
## Chi-squared = 0.0022115, df = 1, p-value = 0.9625
```

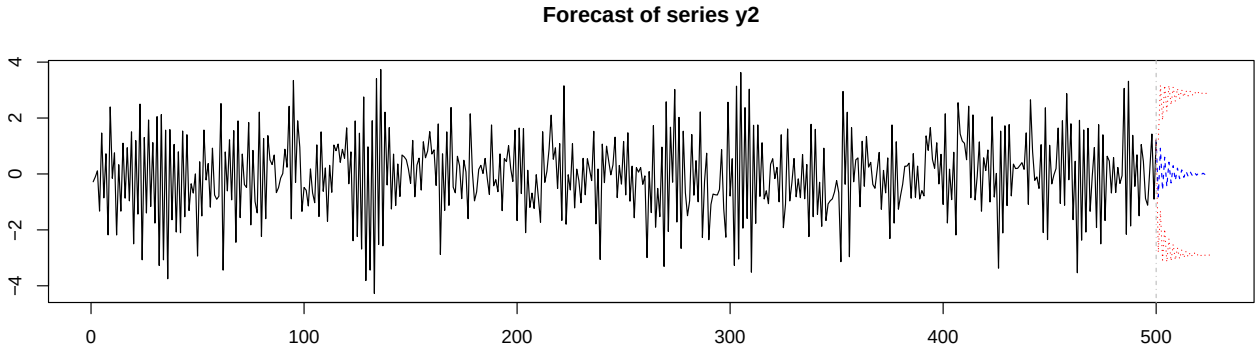
4 Prédiction

Les prédictions récursives et la matrice de covariance d'erreur prévue. Les prédictions récursives sont présentées selon l'équation :

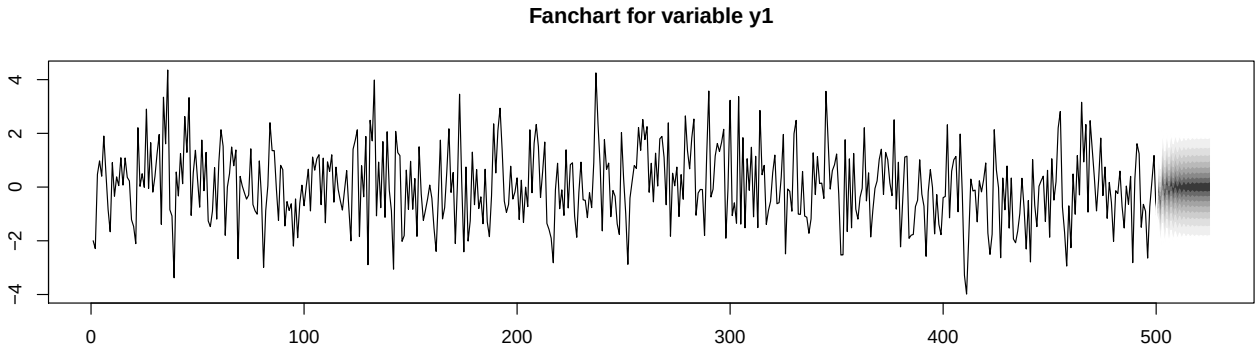
$$y_{T+1|T} = A_1 y_T + ... + A_p y_{T+1-p} + CD_{T+1}$$

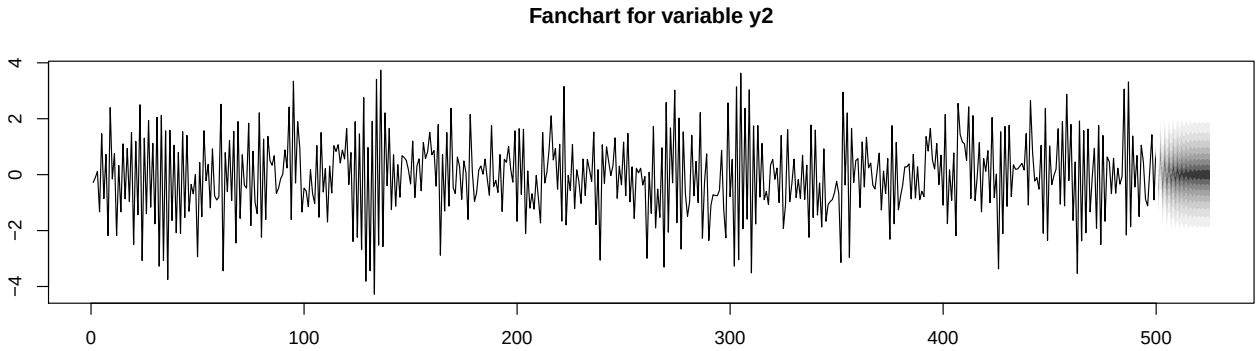
```
prediction = predict(varsimest, n.ahead = 25)
plot(prediction)
```





```
fanchart(prediction)
```



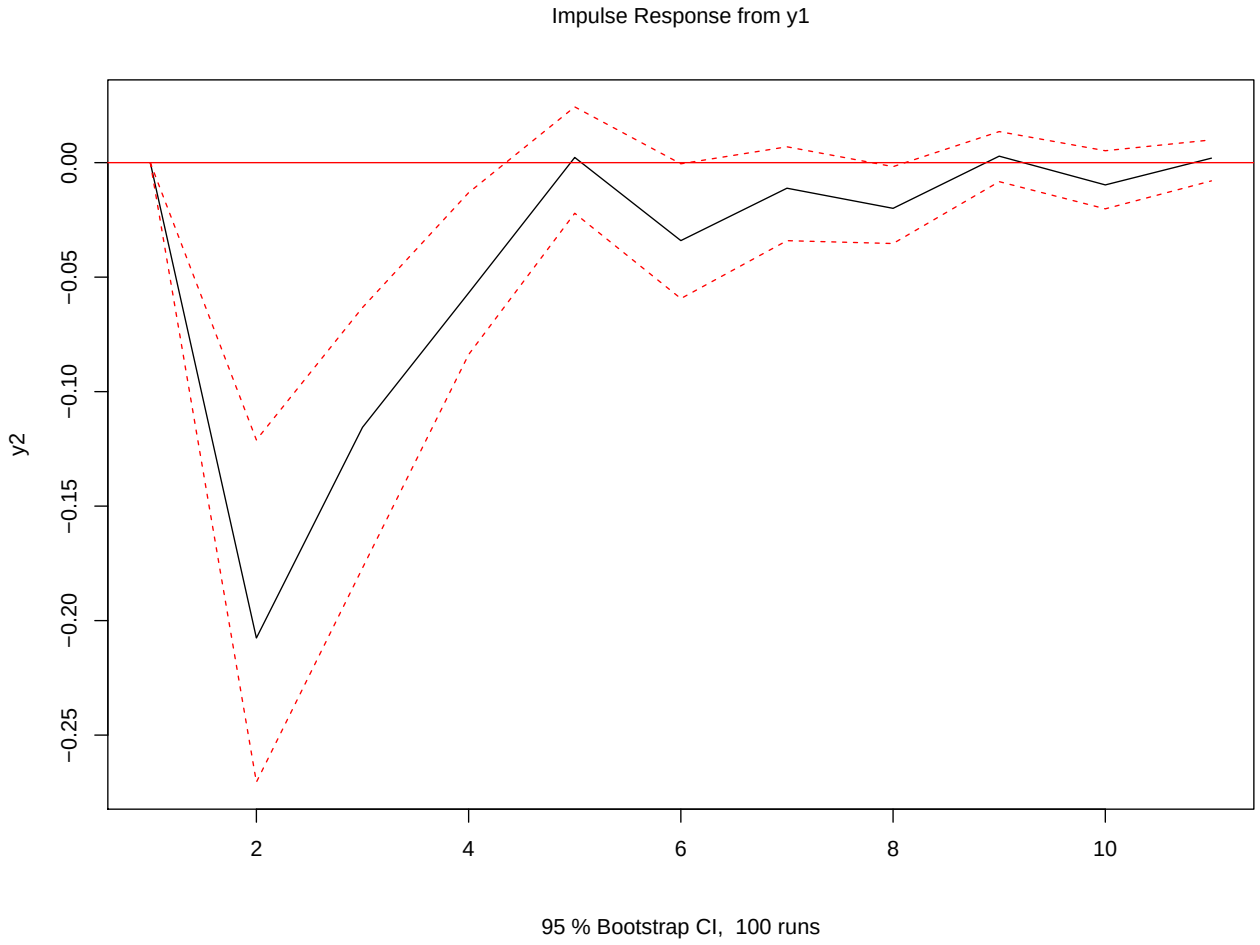


5 Fonction de réponse impulsionnelle, IRF

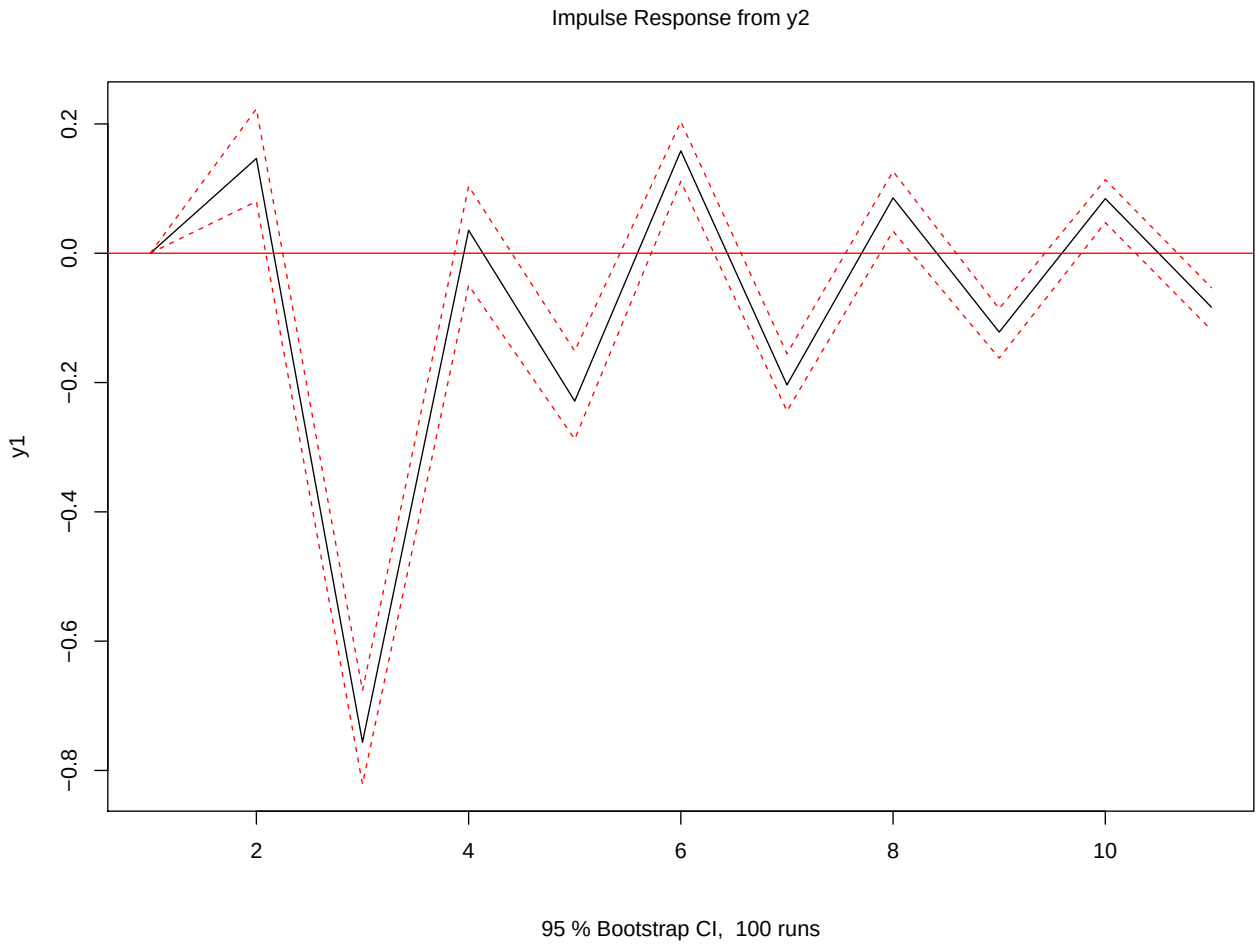
i. Basé sur la décomposition de *Wold* d'un VAR(p) stable;
ii. Étudier les interactions dynamiques entre le système endogène des variables;
iii. Les coefficients (i, j) des matrices Θ_s sont ainsi interprétés comme la réponse attendue de la variable $y_{i,t+s}$ à un changement d'unité de la variable y_{jt};
iv. Peut être cumulé dans le temps $s = 1, 2, ...$: l'impact d'un changement d'unité de la variable j sur la variable i au temps s;
v. Réponses impulsionnelles orthogonalisées: les chocs sous-jacents sont moins susceptibles de se produire de manière isolée (dérivée de la décomposition de Cholesky).

Sur R, voyons la réponse de la variable y1 sur y2 et vice versa.

```
irf.y1 = irf(varsimest, impulse = "y1", response = "y2", n.ahead = 10, ortho = FALSE,
cumulative = FALSE, boot = TRUE, seed = 12345)
irf.y2 = irf(varsimest, impulse = "y2", response = "y1", n.ahead = 10, ortho = FALSE,
cumulative = FALSE, boot = TRUE, seed = 12345)
par(mfrow = c(2,1))
plot(irf.y1)
```


Impulse Response from y1
y2
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
2
4
6
8
10
95 % Bootstrap CI, 100 runs

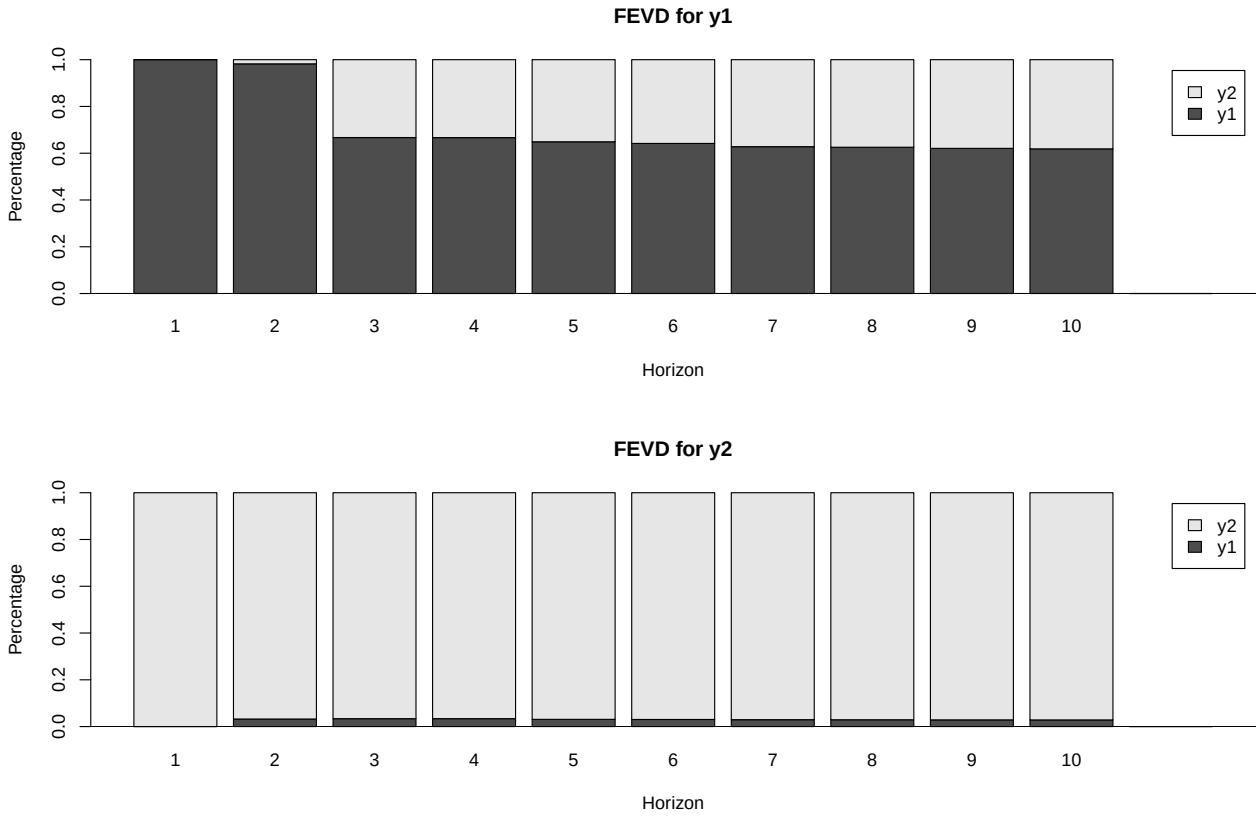
```
plot(irf.y2)
```



6 Décomposition de la variance d'erreur prévue, FEVD

i. FEVD est basé sur des matrices de coefficients de réponse impulsionnelle orthogonalisées Ψ_n;
ii. Analyser la contribution de la variable j à la prévision de variance d'erreur de la variable k.

```
fevd.var2 = fevd(varsimest, n.ahead = 10)
plot(fevd.var2)
```



2 Modèle SVAR

Le modèle VAR peut être considéré comme un modèle de forme réduite. Le modèle SVAR est sa forme structurelle et est défini comme:

$$Ay_t = A'_1 y_{t-1} + ... + A'_p y_{t-p} + B\varepsilon_t$$

Erreurs structurelles: ε_t est un bruit blanc. Matrices de coefficients: A'_i pour $i = 1, ..., p$, sont des coefficients structurels qui pourraient différer de leurs équivalents sous forme réduite.

Le modèle SVAR est utilisé pour identifier les chocs et les retracer par IRF et, ou FEVD en imposant des restrictions aux matrices A et, ou B.

Les résidus de forme réduite peuvent être extraits d'un modèle SVAR en $\mu_t = A^{-1}B\varepsilon_t$ et sa matrice de variance-covariance par $\sum_\mu = A^{-1}BB'A^{-1'}$.

Dans le modèle A, B est défini sur I_K (nombre minimal de restrictions pour l'identification $K(K-1)/2$). Dans le modèle B, A est défini sur I_K (nombre minimal de restrictions pour l'identification $K(K-1)/2$). Le modèle AB, des restrictions peuvent être imposées aux deux matrices (le nombre minimal de restrictions d'identification est $K^2 + K(K-1)/2$).

L'estimation est réalisée directement en minimisant le négatif du log-vraisemblance :

$$lnL_c(A, B) = -\frac{KT}{2}ln(2\pi) + \frac{T}{2}ln|A|^2 - \frac{T}{2}ln|B|^2 - \frac{T}{2}tr(A'B^{-1'}B'A\sum_\mu)$$

L'algorithme de score a été proposé par Amisano et Giannini (1997).

```
data("Canada")
A <- diag(4)
diag(A) = NA
A[1, 2] = NA
A[1, 3] = NA
A[3, 2] = NA
A[4, 1] = NA
varest = VAR(Canada, p = 2, type = "none")
svarA = SVAR(varest, estmethod = "direct", Amat = A, Bmat = NULL, hessian = TRUE,
             method = "BFGS")
svarA
```

```
##
## SVAR Estimation Results:
## ========================
##
##
## Estimated A matrix:
##          e    prod     rw     U
## e    2.694 -0.0629 0.1814 0.000
## prod 0.000  1.5149 0.0000 0.000
## rw   0.000 -0.1996 1.3009 0.000
## U    2.691  0.0000 0.0000 4.736
```

```
svarB = SVAR(varest, estmethod = "scoring", Amat = A, Bmat = NULL, hessian = TRUE,
             method = "BFGS")
svarB
```

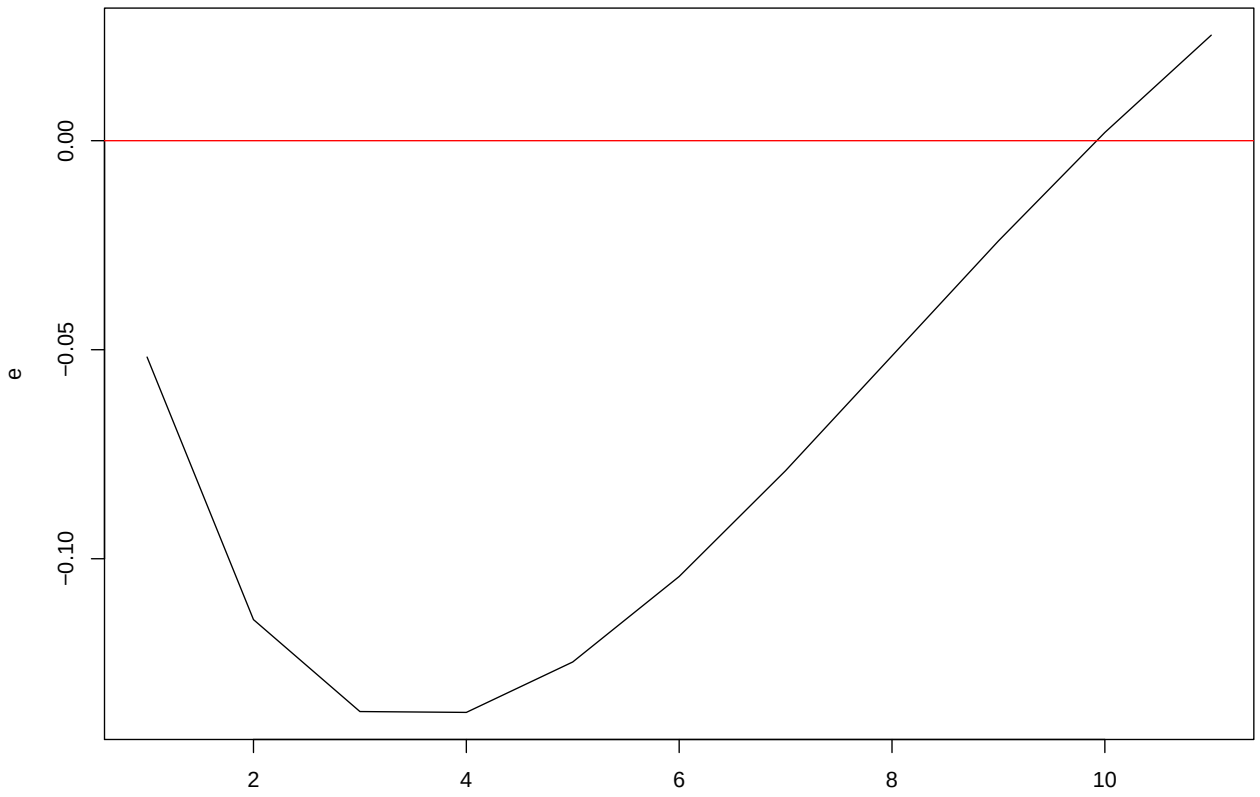
```
##
## SVAR Estimation Results:
## ========================
##
##
## Estimated A matrix:
##          e     prod     rw     U
## e    2.694 -0.06285 0.1814 0.000
## prod 0.000  1.51477 0.0000 0.000
## rw   0.000 -0.19967 1.3008 0.000
## U    2.691  0.00000 0.0000 4.736
```

1 Analyse de réponse impulsionnelle

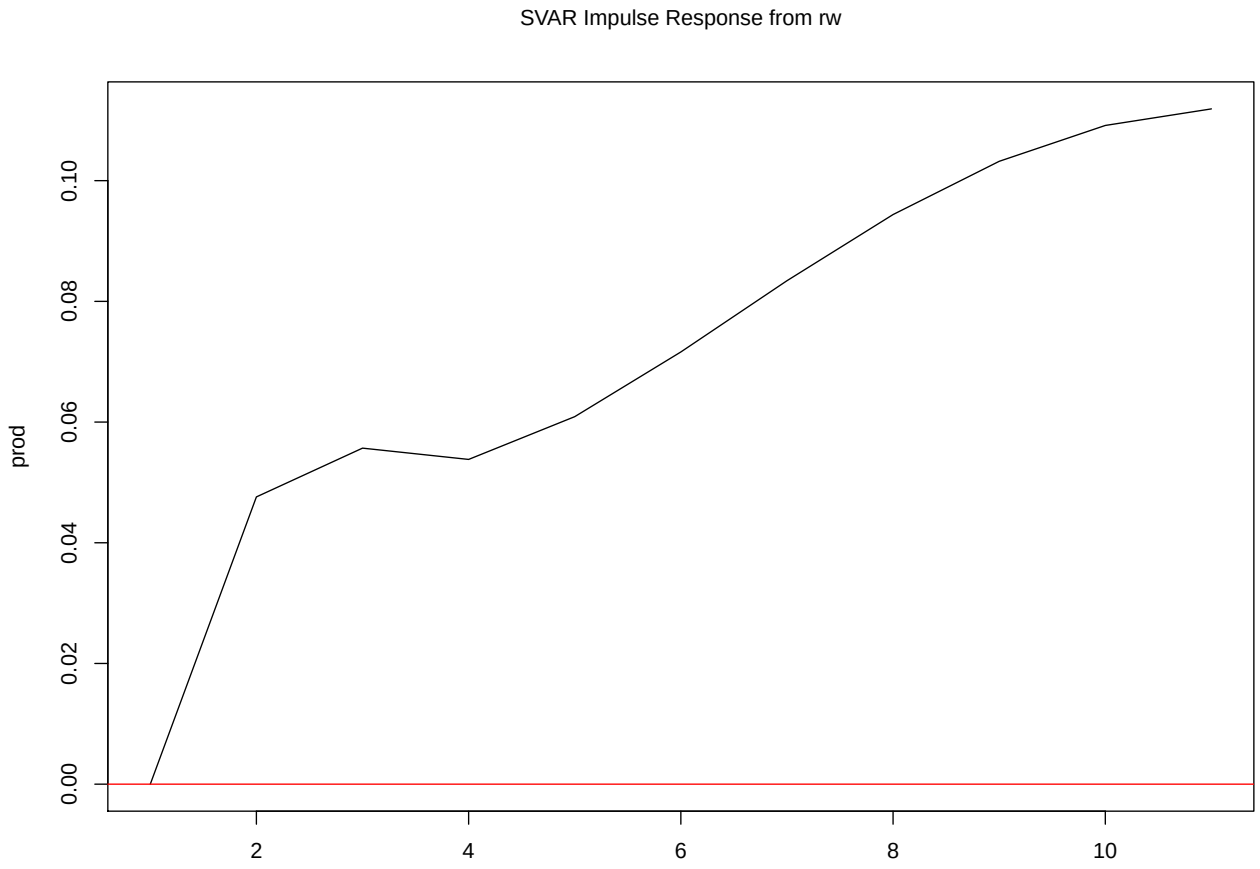
Les coefficients de réponse impulsionnelle pour le modèle SVAR sont tirés de :

$$\Theta_i = \phi_i A^{-1} B, i = 1, 2, ..., n$$

```
irf.svar.rw = irf(svarA, impulse = "rw", response = "e", n.ahead = 10, cumulative = FALSE,
             boot = FALSE, seed = 12345)
irf.svar.prod = irf(svarB, impulse = "rw", response = "prod", n.ahead = 10, cumulative = FALSE,
             boot = FALSE, seed = 12345)
plot(irf.svar.rw)
```



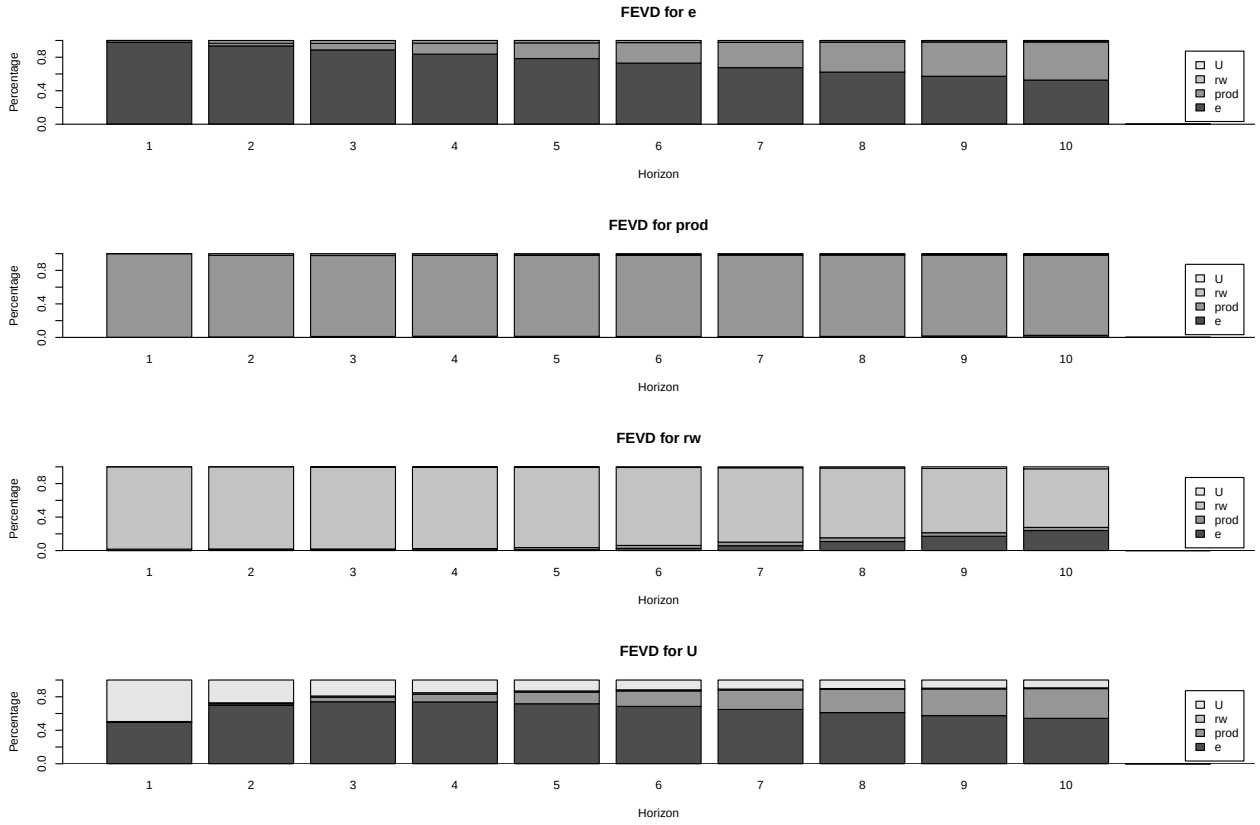

```
plot(irf.svar.prod)
```



2. Décomposition de la variance d'erreur prévue

Les erreurs de prévision de $y_{T+h|T}$ sont dérivées des réponses impulsionnelles de SVAR et la dérivation de la décomposition de la variance d'erreur de prévision est similaire à celle décrite pour les VAR.

```
fevd.svarb = fevd(svarA, n.ahead = 10)
plot(fevd.svarb)
```

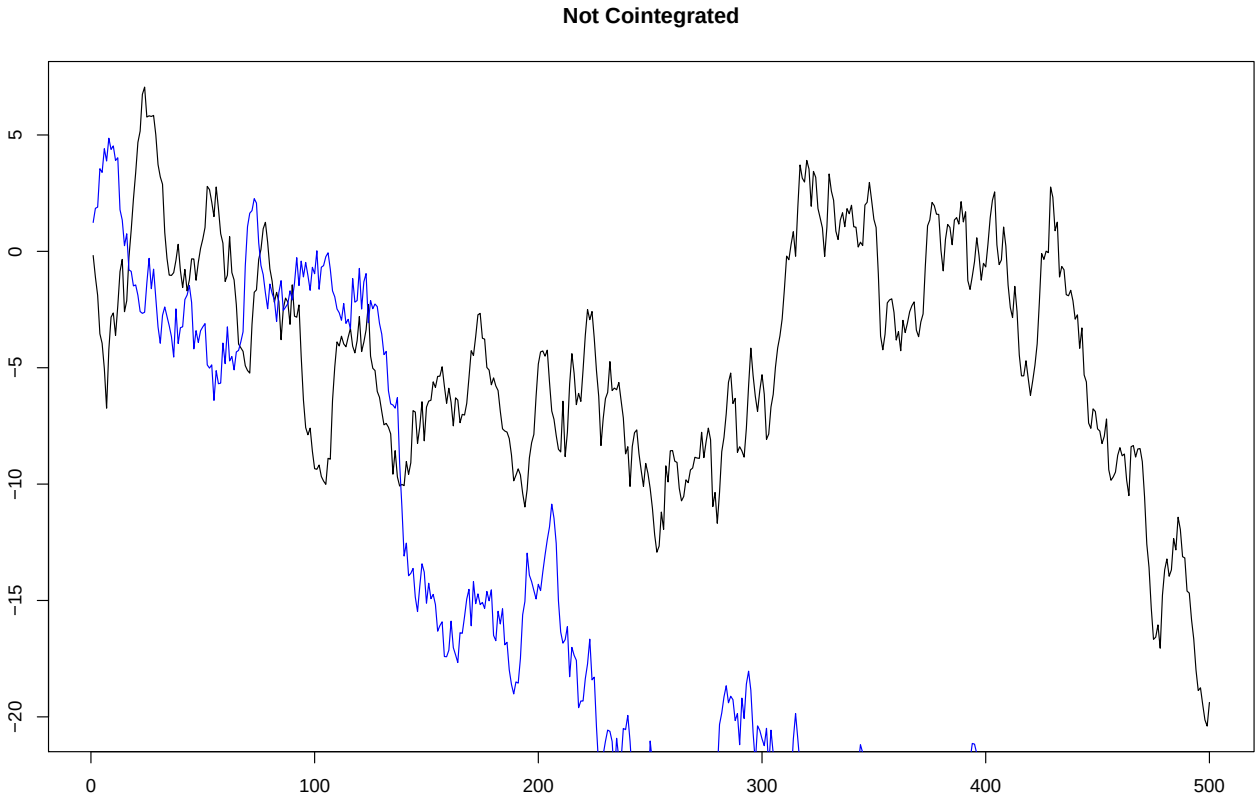


3 Cointrégation

1 Problème de régression fausse

$I(1)$ les variables non cointégrées sont régressées sur chacune autre. Les coefficients de pente de régression ne convergent pas en probabilité zéro. Les t-statistiques divergent de plus ou moins l'infinie. Le coefficient de détermination tant vers l'unité. Il faut toujours être prudent lorsque le coefficient de déterminantion est supérieur à la statistique DW.

```
library(lmtest)
set.seed(54321)
e1 = rnorm(500)
e2 = rnorm(500)
y1 = cumsum(e1)
y2 = cumsum(e2)
sr.reg1 = lm(y1 ~ y2)
sr.dw = dwtest(sr.reg1)
sr.reg2 = lm(diff(y1) ~ diff(y2))
plot.ts(y1, xlab = "", ylab = "", main = "Not Cointegrated")
lines(y2, col = "blue")
```



Le graphique ci-haut nous montre le comportement de deux variables non cointégrés. Les statistiques des tests sont visualisées en faisant la commande *summary(sr.reg1)* pour la régression simple et *summary(sr.reg2)* pour la régression en différence.

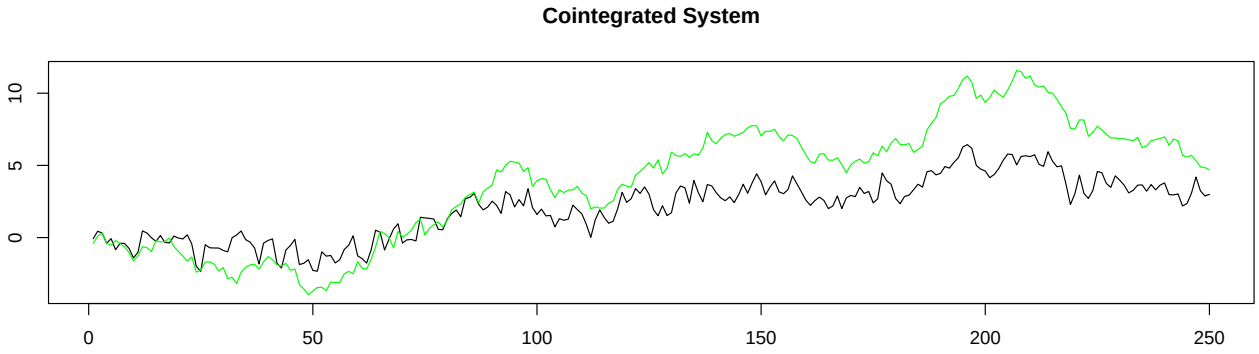
On dit que les composantes d'un vecteurs y_t sont cointégrées de l'ordre d, b noté $y_t \sim CI(d, b)$, si (a) toutes les composantes de y_t sont $I(d)$ et b un vecteur $\beta(\neq 0)$ existe pour que $z_t = \beta' y_t \sim I(d-b)$, $b > 0$. Le vecteur β est appelé *vecteur de cointégration*.

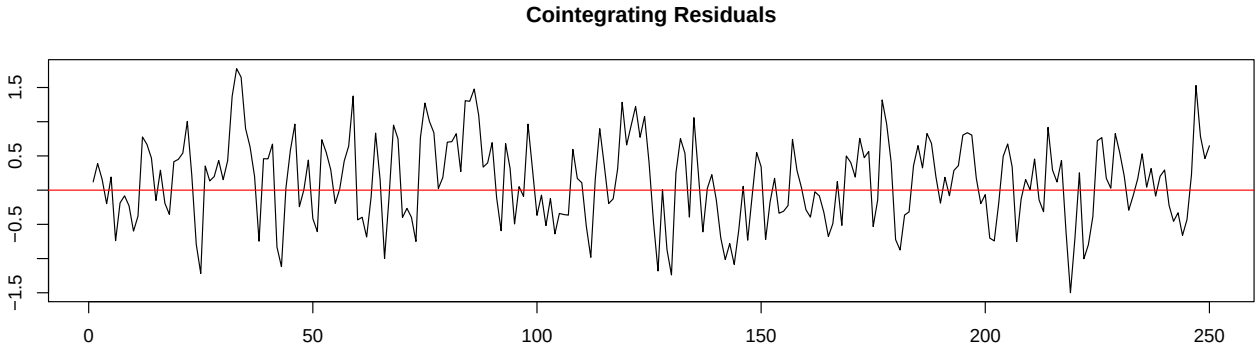
Si le vecteur y_t, où $(nx1)$ est cointégré avec $0 < r < n$ vecteurs de cointégration, il existe $n - r$ tendances stochastiques courantes notée $I(1)$.

```
set.seed(12345)
e1 = rnorm(250, mean = 0, sd = 0.5)
e2 = rnorm(250, mean = 0, sd = 0.5)
u.ar3 = arima.sim(model = list(ar = c(0.6, -0.2, 0.1)), n = 250, innov = e1)
y2 = cumsum(e2)
y1 = u.ar3 + 0.5*y2
ymax = max(c(y1, y2))
ymin = min(c(y1, y2))

layout(matrix(1:2, nrow = 2, ncol = 1))
plot(y1, xlab = "", ylab = "", ylim = c(ymin, ymax), main = "Cointegrated System")
lines(y2, col = "green")

plot(u.ar3, ylab = "", xlab = "", main = "Cointegrating Residuals")
abline(h = 0, col = "red")
```





Nous voyons bien sur le graphique que nos deux systèmes sont cointégrés.

2 Cointégration et Modèle à correction d'erreur, ECM

Soit le système cointégré ci-après : $I(1), y_t = (y_{1,t}, y_{2,t})$ avec vecteur de cointégration $\beta = (1, -\beta_2)'$, un ACM existe sous la forme :

$$\Delta y_{1,t} = \alpha_1 + \gamma_1(y_{1,t-1} - \beta_2 y_{2,t-1}) + \sum_{i=1}^{K} \Psi_{1,i}\Delta y_{1,t-i} + \sum_{i=1}^{L} \Psi_{2,i}\Delta y_{2,t-i} + \varepsilon_{1,t}$$

$$\Delta y_{2,t} = \alpha_2 + \gamma_2(y_{1,t-1} - \beta_2 y_{2,t-1})_{t-1} + \sum_{i=1}^{K} \eta_{1,i}\Delta y_{1,t-i} + \sum_{i=1}^{L} \eta_{2,i}\Delta y_{2,t-i} + \varepsilon_{2,t}$$

i. Cointégration à la Engle et Granger : Procédure en deux étapes

Estimez la relation à long terme, c'est-à-dire la régression en niveaux et les résidus de test pour $I(0)$. On prend ensuite les résidus de la première étape et on les utilisent dans la régression ECM.

```
library(dynlm)

reg0 = lm(y1 ~ y2)
ect = resid(reg0)[1:249]
dy1 = diff(y1)
dy2 = diff(y2)
ecmdat = cbind(dy1, dy2, ect)
ecm = dynlm(dy1 ~ L(ect, 1) + L(dy1, 1) + L(dy2, 1) , data = ecmdat)
coeftest(ecm)
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  0.006412   0.037576  0.1706 0.8646488
## L(ect, 1)   -0.621609   0.072485 -8.5757 1.152e-15 ***
## L(dy1, 1)   -0.423467   0.070264 -6.0268 6.133e-09 ***
## L(dy2, 1)    0.317131   0.091067  3.4824 0.0005888 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Le tableau ci-haut présente les résultats de ECM selon la méthode proposée par Engle et Granger, voir Engle et Yoo (1987).

ii. Cointégration à la Phillips et Ouliaris

Tests basés sur les résidus: statistique de test et de trace du rapport de variance. Le test basé sur la régression est représenté comme :

$$z_t = \Gamma z_{t-1} + \xi_t$$

Où z_t est partitionné comme $z_t = (y_t, x_t')$ avec une dimmension de x_t qui est égale à $[m = n + 1]$.

L'hypothèse nulle est la non cointégration. Voir Phillips et Ouliaris (1990).

```
z = cbind(y1, y2)
test.po.pu = ca.po(z, demean = "none", type = "Pu")
test.po.pz = ca.po(z, demean = "none", type = "Pz")
#summary(test.po.pu)
#summary(test.po.pz)
```

Pour voir les résutats des tests statistiques, il faut tout simplement écrire *summary(test.po.pu)* et *summary(test.po.pz)*.

4 Vecteur à correction d'erreur, VECM

Nous montrons ici comment on peut transformer un modèle du type *vecteur autorégressif, VAR* à un modèle VECM.

Soit le modèle VAR ci-après :

$$y = A_1 y_{t-1} + ... + A_p y_{t-p} + CD_t + \varepsilon_t$$

La transformation pour le modèle VECM est donnée sous la forme :

$$\Delta y_t = \Gamma_1 \Delta y_{t-1} + ... + \Gamma_{K-1} \Delta y_{t-p+1} + \Pi y_{t-1} + CD_t + \varepsilon_t$$
$$\Gamma_i = -(A_{i+1} + ... + A_p), i = 1, 2, ..., p-1$$
$$\Pi = -(I - A_1 - ... - A_p)$$

À long-terme le VECM prend la forme :

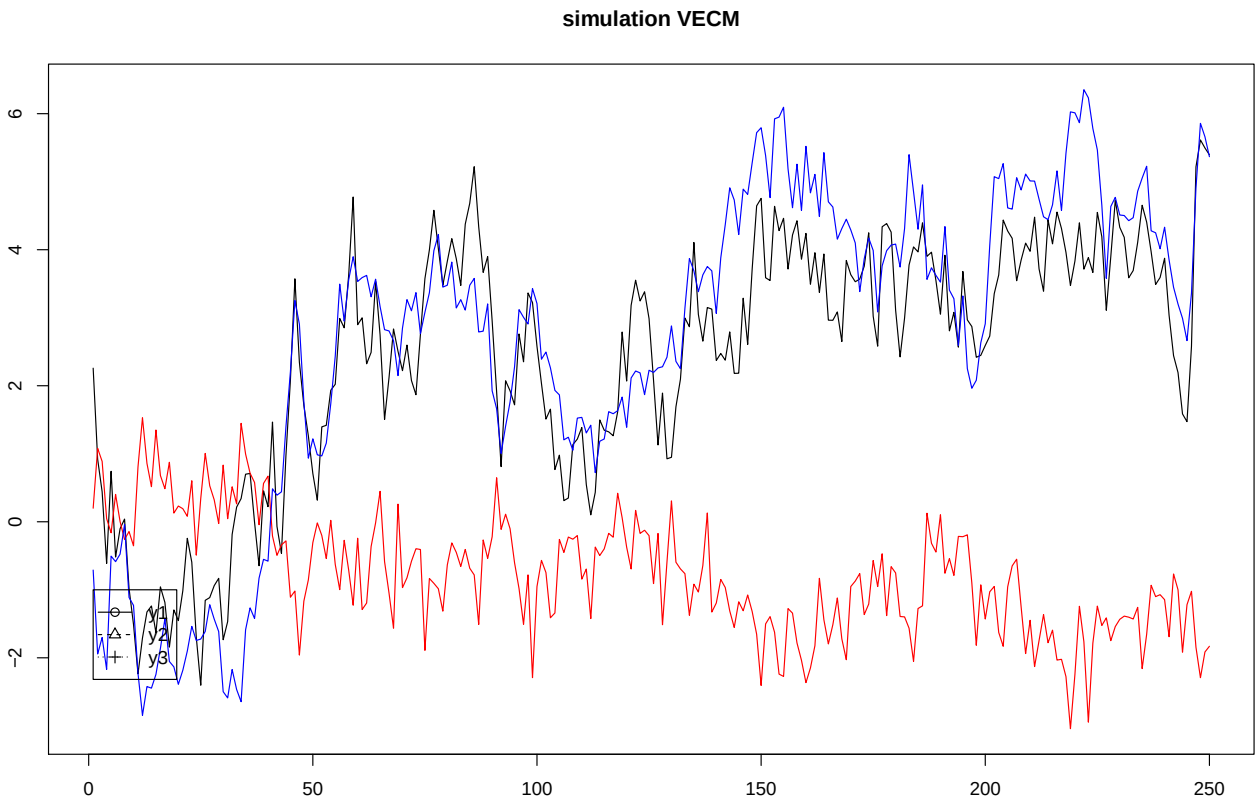
$$\Delta y_t = \Gamma_1 \Delta y_{t-1} + ... + \Gamma_{p-1} \Delta y_{t-p+1} + \Pi y_{t-p} + CD_t + \varepsilon_t$$

$$\Gamma_i = -(I - A_1 - ... - A_i), i = 1, 2, ..., p-1$$
$$\Pi = -(I - A_1 - ... - A_p)$$

ci-bas, l'exemple d'un VECM

```
set.seed(12345)
e1 = rnorm(250, 0, 0.5)
e2 = rnorm(250, 0, 0.5)
e3 = rnorm(250, 0, 0.5)
u1.ar1 = arima.sim(model = list(ar = 0.75), innov = e1, n = 250)
u2.ar1 <- arima.sim(model = list(ar = 0.3), innov = e2, n = 250)
y3 = cumsum(e3)
y1 = 0.8 * y3 + u1.ar1
y2 = -0.3 * y3 + u2.ar1
ymax <- max(c(y1, y2, y3))
ymin <- min(c(y1, y2, y3))

plot(y1, ylab = "", xlab = "", main = "simulation VECM", ylim = c(ymin, ymax))
lines(y2, col = "red")
lines(y3, col = "blue")
legend(1, -1, c("y1", "y2", "y3"), pch = c(1,2,3), lty = c(1,2,3))
```



L'inférence statistique est Basée sur les corrélations canoniques entre y_t et Δy_t avec différences décalées. Ces corrélations se présentent comme :

$$H_{00} = \frac{1}{T}\sum_{t=1}^{T} \hat{\mu}_t \hat{\mu}_t', H_{01} = H_{10} = \sum_{t=1}^{T} \hat{\mu}_t \hat{v}_t', H_{11} = \frac{1}{T}\sum_{t=1}^{T} \hat{v}_t \hat{v}_t'$$

Et les valeurs propres sont données comme étant : $|\lambda H_{11} - H_{10} H_{00} H_{01}| = 0$.

```r
y.daframe = data.frame(y1, y2, y3)
vecm1 = ca.jo(y.daframe, type = "eigen", spec = "transitory")
vecm2 = ca.jo(y.daframe, type = "trace", spec = "transitory")
vecm.r2 = cajorls(vecm1, r = 2)

print(vecm.r2)
```

```
## $rlm
##
## Call:
## lm(formula = substitute(form1), data = data.mat)
##
## Coefficients:
##           y1.d       y2.d       y3.d
## ect1      -0.331293   0.064612   0.012682
## ect2       0.094473  -0.709385  -0.009165
## constant   0.168371  -0.027019   0.025255
## y1.dl1     0.103616  -0.037599   0.055476
## y2.dl1     0.049979  -0.006223   0.049652
## y3.dl1    -0.147211  -0.034070  -0.063251
##
##
## $beta
##                ect1      ect2
## y1.l1  1.000000e+00 0.0000000
## y2.l1 -2.775558e-17 1.0000000
## y3.l1 -7.328534e-01 0.2951962
```

```r
summary(vecm1)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.27036254 0.15474942 0.01884032
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 2 |  4.72  6.50  8.18 11.65
## r <= 1 | 41.69 12.91 14.90 19.19
## r = 0  | 78.17 18.90 21.07 25.75
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##           y1.l1      y2.l1     y3.l1
## y1.l1  1.000000  1.0000000 1.0000000
## y2.l1 -4.732436  0.2273774 0.1513858
## y3.l1 -2.129850 -0.6657324 2.3153224
```

```
##
## Weights W:
## (This is the loading matrix)
##
##             y1.l1       y2.l1        y3.l1
## y1.d -0.034235501 -0.29705774 -0.008294582
## y2.d  0.145988517 -0.08137695  0.003335110
## y3.d  0.002429191  0.01025326 -0.010523045
```

```
summary(vecm2)
```

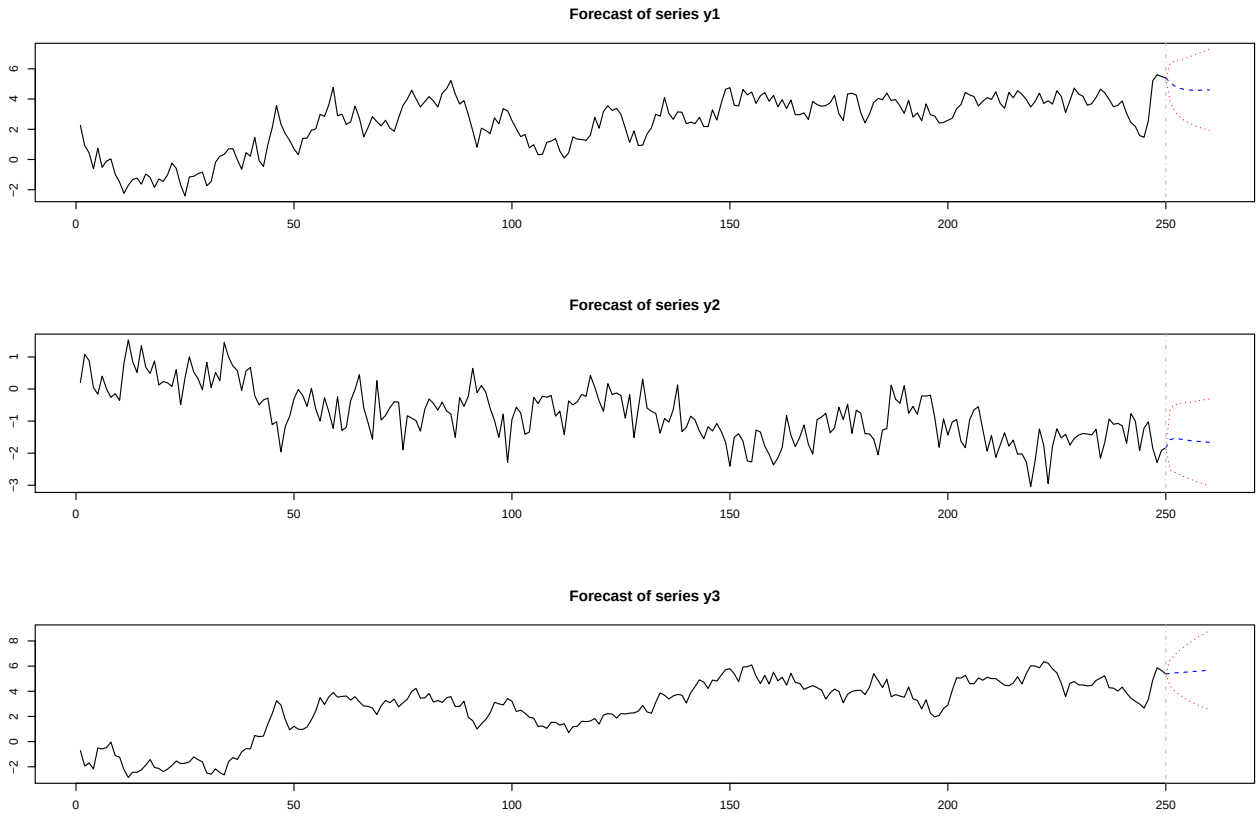
```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.27036254 0.15474942 0.01884032
##
## Values of teststatistic and critical values of test:
##
##            test 10pct  5pct  1pct
## r <= 2 |   4.72  6.50  8.18 11.65
## r <= 1 |  46.41 15.66 17.95 23.52
## r = 0  | 124.58 28.71 31.52 37.22
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##           y1.l1      y2.l1     y3.l1
## y1.l1  1.000000  1.0000000 1.0000000
## y2.l1 -4.732436  0.2273774 0.1513858
## y3.l1 -2.129850 -0.6657324 2.3153224
##
## Weights W:
## (This is the loading matrix)
##
##             y1.l1       y2.l1        y3.l1
## y1.d -0.034235501 -0.29705774 -0.008294582
## y2.d  0.145988517 -0.08137695  0.003335110
## y3.d  0.002429191  0.01025326 -0.010523045
```

Les tableaux ci-dessus nous donnent toutes les statistiques des tests pour la prise de décision concernant le modèle à correction d'erreur cointégré.

1 prédiction

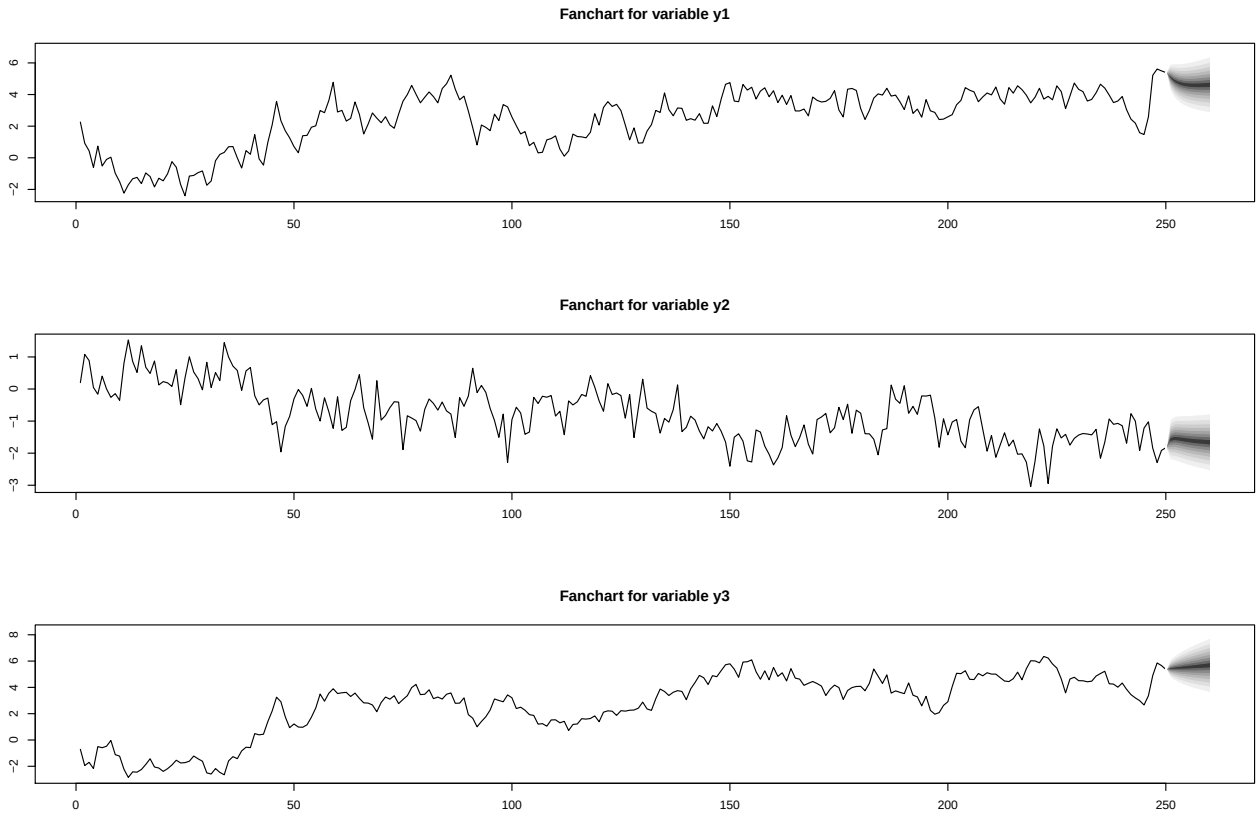
Conversion de VECM restreint en VAR de niveau. La vérification des prévisions, de l'IRF, du FEVD et du diagnostic s'applique également aux modèles $VAR(p)$ fixes.

```
vecm.level = vec2var(vecm1, r = 2)
vecm.prediction = predict(vecm.level, n.ahead = 10)
plot(vecm.prediction)
```



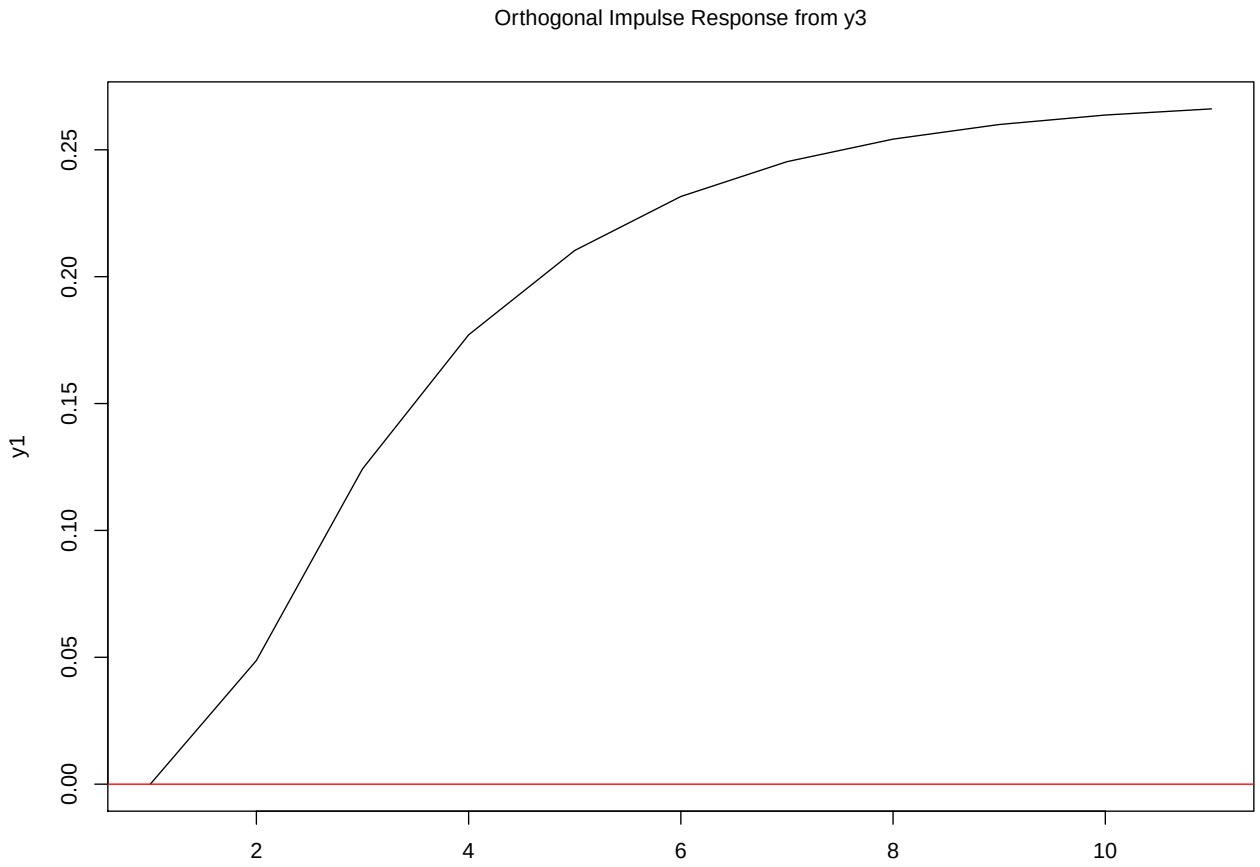
Ci-haut le grphiques de prévision pour les variables y1, y2 et y3.

```
fanchart(vecm.prediction)
```



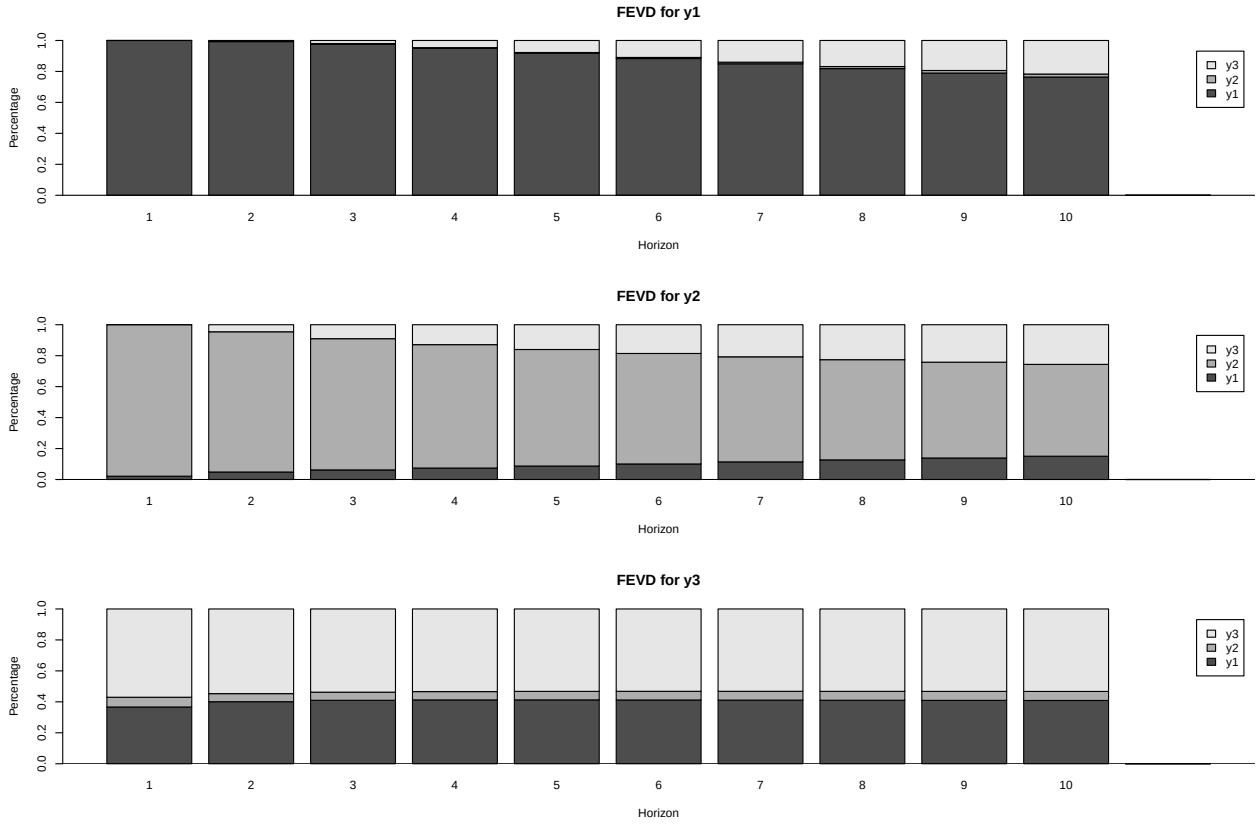
2 Réponse impulsionnelle, IRF et FEVD

```
vecm.irf = irf(vecm.level, impulse = "y3", response ="y1", boot = FALSE)
plot(vecm.irf)
```



La réponse de la variable y1 face un choc de la variable y3.

```
vecm.fevd = fevd(vecm.level)
plot(vecm.fevd)
```



3 Statistiques de tests

```
vecm.norm = normality.test(vecm.level)
print(vecm.norm)
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object vecm.level
## Chi-squared = 9.1317, df = 6, p-value = 0.1663
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object vecm.level
## Chi-squared = 5.8516, df = 3, p-value = 0.1191
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object vecm.level
```

```
## Chi-squared = 3.2801, df = 3, p-value = 0.3504
```

```
## $jb.mul
## $jb.mul$JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object vecm.level
## Chi-squared = 9.1317, df = 6, p-value = 0.1663
##
##
## $jb.mul$Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object vecm.level
## Chi-squared = 5.8516, df = 3, p-value = 0.1191
##
##
## $jb.mul$Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object vecm.level
## Chi-squared = 3.2801, df = 3, p-value = 0.3504
```

Les tests de normalité des résidus nous montre que les erreurs sont normalement distribuées.

```
vecm.arch = arch.test(vecm.level)
print(vecm.arch)
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object vecm.level
## Chi-squared = 173.46, df = 180, p-value = 0.6231
```

```
## $arch.mul
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object vecm.level
## Chi-squared = 173.46, df = 180, p-value = 0.6231
```

Le test ARCH d'hétéroscédasticité nous indique que l'hypothèse nulle d'homoscédasticité ne peut pas être réjetée. Nous sommes pas en présence d'hétéroscédasticité des résidus.

```
vecm.serial = serial.test(vecm.level)
print(vecm.serial)
```

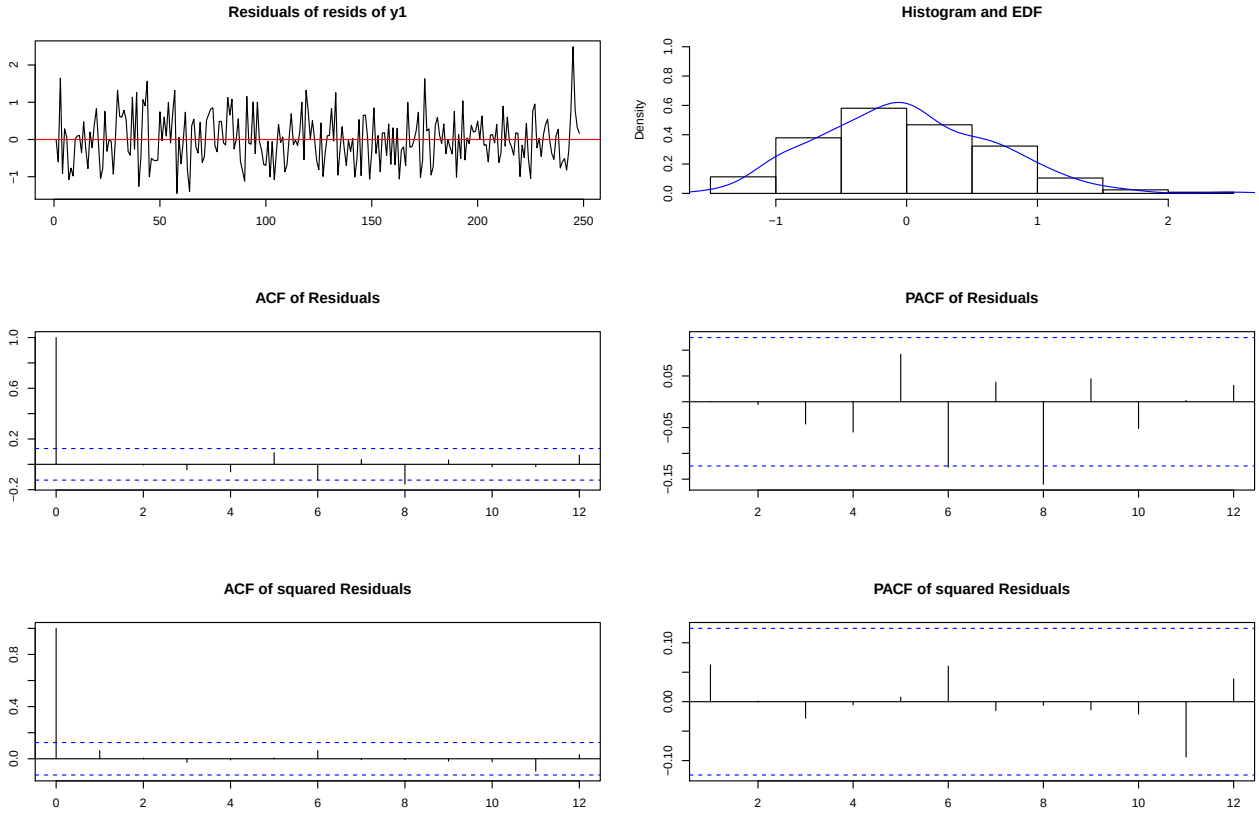
```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object vecm.level
## Chi-squared = 120.92, df = 129, p-value = 0.6817
```

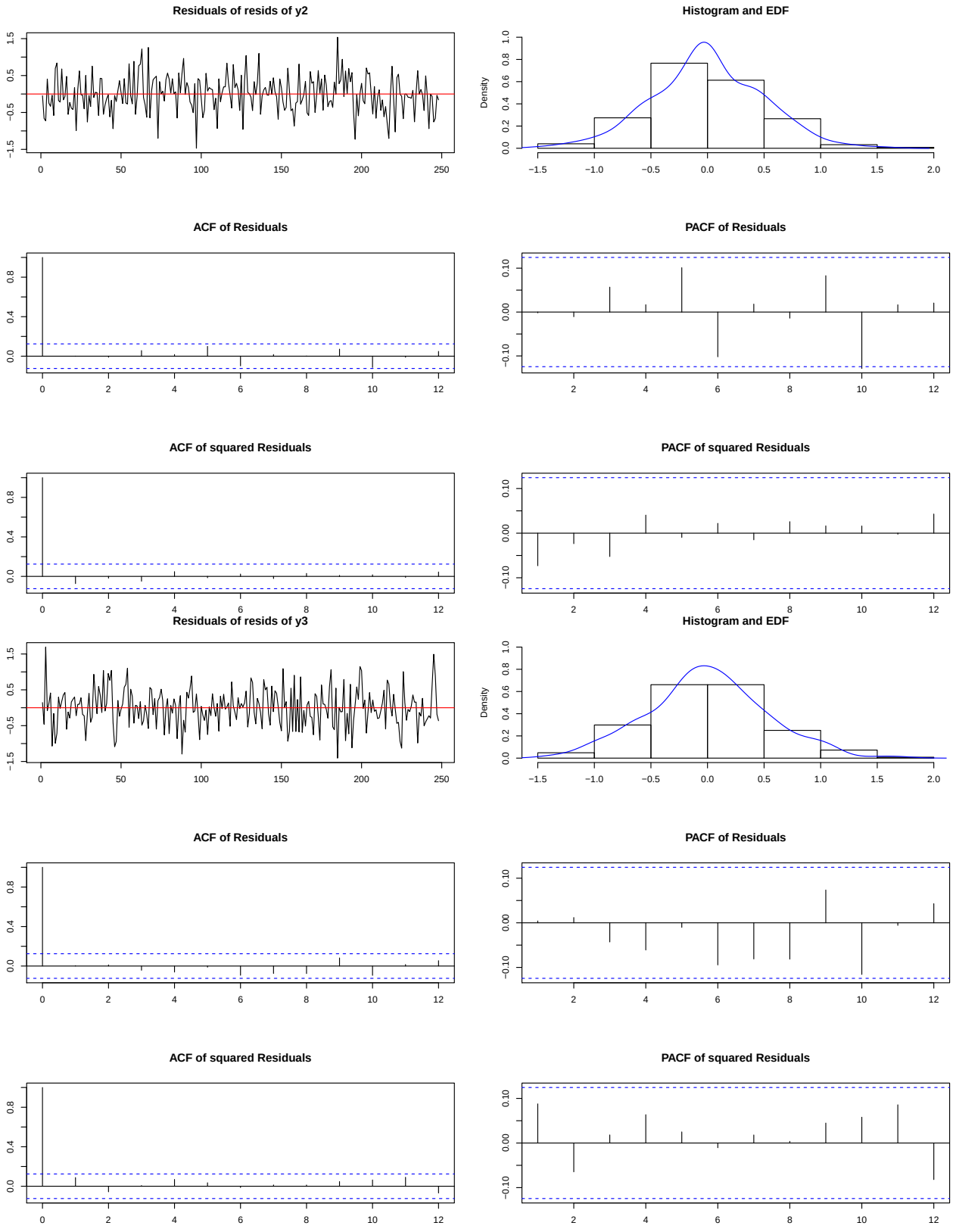
```
## $serial
##
```

```
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object vecm.level
## Chi-squared = 120.92, df = 129, p-value = 0.6817
```

Le test de Portmanteau nous indique que le modèle estimé sous VECM n'a pas de correlation de série. On peut voir aussi ce comportement sur un graphique pour nos trois variables.

```
plot(vecm.serial)
```





Nous pouvons toute fois faire une analyse comparative des deux économies où, nous testions de tendance linéaire. Par exemple, testez si la tendance linéaire dans VAR est existante. Cela correspond à l'inclusion d'une constante dans le terme de correction d'erreur.

```
data(denmark)
sjd = as.matrix(denmark[, c("LRM", "LRY", "IBO", "IDE")])
sjd.vecm = ca.jo(sjd, ecdet = "const", type = "eigen", K = 2, spec = "longrun", season = 4)
lttest.1 = lttest(sjd.vecm, r = 1)
```

```
## LR-test for no linear trend
##
## H0: H*2(r<=1)
## H1: H2(r<=1)
##
## Test statistic is distributed as chi-square
## with 3 degress of freedom
##         test statistic p-value
## LR test           1.98    0.58
```

Le test statistique pour le Danmark nous dit qu'il n'a donc pas une tendance linéaire dans le modèle VECM estimé. Pouvons-nous aussi pour la Finland.

```
data(finland)
sjf = as.matrix(finland)
sjf.vecm = ca.jo(sjf, ecdet = "none", type = "eigen", K = 2, spec = "longrun", season = 4)
lttest.2 = lttest(sjf.vecm, r=3)
```

```
## LR-test for no linear trend
##
## H0: H*2(r<=3)
## H1: H2(r<=3)
##
## Test statistic is distributed as chi-square
## with 1 degress of freedom
##         test statistic p-value
## LR test           4.78    0.03
```

Il est aussi conseillé de faire le test *d'exogénéité* de la série. Le test de l'exogenité, c'est-à-dire que certaines variables n'entrent pas dans la ou les relations de cointégration. Dans ce cas, on test le rapport de vraisemblance pour l'hypothèse: $H_4 : \alpha = A\Psi$ avec $r(K - m)$ degrés de liberté.

```
data(UKpppuip)
attach(UKpppuip)
head(UKpppuip)
```

```
##         p1       p2       e12         i1         i2      doilp0
## 1 3.399837 3.846749 -4.899593 0.04707441 0.05059805 0.000000000
## 2 3.412952 3.856261 -4.894152 0.04678816 0.04859967 0.006686741
## 3 3.430709 3.864228 -4.832260 0.05987140 0.05543471 0.000000000
## 4 3.452861 3.881285 -4.803663 0.07250669 0.05808021 0.002213712
## 5 3.465303 3.913175 -4.785582 0.07890360 0.07622003 0.102026935
## 6 3.469929 3.953242 -4.785582 0.07148310 0.08296152 0.132782319
##        doilp1
## 1 0.000000000
## 2 0.000000000
## 3 0.006686741
## 4 0.000000000
## 5 0.002213712
## 6 0.102026935
```

```r
dat1 = cbind(p1, p2, e12, i1, i2)
dat2 = cbind(doilp0, doilp1)
H1 = ca.jo(dat1, K = 2, season = 4, dumvar = dat2)
A1 = matrix(c(1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1), nrow = 5, ncol = 4)
A2 = matrix(c(1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0), nrow = 5, ncol = 4)
H4.1 = summary(alrtest(z = H1, A = A1, r = 2))
H4.2 = summary(alrtest(z = H1, A = A2, r = 2))
print(H4.1)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Estimation and testing under linear restrictions on beta
##
## The VECM has been estimated subject to:
## beta=H*phi and/or alpha=A*psi
##
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    0    0    0    0
## [3,]    0    1    0    0
## [4,]    0    0    1    0
## [5,]    0    0    0    1
##
## Eigenvalues of restricted VAR (lambda):
## [1] 0.4002 0.2854 0.1674 0.0880 0.0000
##
## The value of the likelihood ratio test statistic:
## 0.66 distributed as chi square with 2 df.
## The p-value of the test statistic is: 0.72
##
## Eigenvectors, normalised to first column
## of the restricted VAR:
##
##              [,1]     [,2]
## RK.p1.l2   1.0000   1.0000
## RK.p2.l2  -0.9249  -1.1420
## RK.e12.l2 -0.9690  -3.2247
## RK.i1.l2  -3.4812  34.6544
## RK.i2.l2  -1.7651 -32.3316
##
## Weights W of the restricted VAR:
##
##         [,1]    [,2]
## [1,] -0.0645  0.0012
## [2,]  0.0000  0.0000
## [3,]  0.1259 -0.0005
## [4,]  0.0403 -0.0043
## [5,]  0.0566  0.0085
```

Pour detecter l'exogénéité, on regarde la valeur du test statistique qui est de 0.66 et la p-value de 0.72.

```r
print(H4.2)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Estimation and testing under linear restrictions on beta
##
## The VECM has been estimated subject to:
## beta=H*phi and/or alpha=A*psi
##
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    0    1    0    0
## [3,]    0    0    1    0
## [4,]    0    0    0    1
## [5,]    0    0    0    0
##
## Eigenvalues of restricted VAR (lambda):
## [1] 0.3870 0.2560 0.1945 0.0856 0.0000
##
## The value of the likelihood ratio test statistic:
## 4.38 distributed as chi square with 2 df.
## The p-value of the test statistic is: 0.11
##
## Eigenvectors, normalised to first column
## of the restricted VAR:
##
##               [,1]    [,2]
## RK.p1.l2   1.0000  1.0000
## RK.p2.l2  -0.8518 -1.4443
## RK.e12.l2 -0.9525  2.1238
## RK.i1.l2  -4.6381 -0.7205
## RK.i2.l2  -1.0922  1.3813
##
## Weights W of the restricted VAR:
##
##         [,1]    [,2]
## [1,] -0.0624 -0.0022
## [2,] -0.0157 -0.0101
## [3,]  0.1029 -0.0418
## [4,]  0.0363 -0.0049
## [5,]  0.0000  0.0000
```

Les tests ne dépendent pas de la normalisation de β. Les tests sont des tests de vraisemblance, similaires pour les restrictions de test sur α. Cela exige de réaliser les test des restrictions pour toutes les relations de cointégration. r_1 les relations de cointégration sont supposées être connues et r_2 les relations cointégrées doivent être estimées, l'ensemble on a $r = r_1 + r_2$. r_1 les relations de cointégration sont estimées avec restrictions et r_2 les relations de cointégration sont estimées sans contraintes, $r = r_1 + r_2$.

```r
H.3.1 = matrix(c(1,-1,-1,0,0,0,0,0,1,0,0,0,0,0,1), c(5,3))
H.3.2 = matrix(c(1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,-1), c(5,4))
H31 = blrtest(z = H1, H = H.3.1, r = 2)
summary(H31)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Estimation and testing under linear restrictions on beta
##
## The VECM has been estimated subject to:
## beta=H*phi and/or alpha=A*psi
##
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]   -1    0    0
## [3,]   -1    0    0
## [4,]    0    1    0
## [5,]    0    0    1
##
## Eigenvalues of restricted VAR (lambda):
## [1] 0.3855 0.2776 0.0895
##
## The value of the likelihood ratio test statistic:
## 2.76 distributed as chi square with 4 df.
## The p-value of the test statistic is: 0.6
##
## Eigenvectors, normalised to first column
## of the restricted VAR:
##
##         [,1]     [,2]    [,3]
## [1,]  1.0000   1.0000  1.0000
## [2,] -1.0000  -1.0000 -1.0000
## [3,] -1.0000  -1.0000 -1.0000
## [4,] -2.6141  28.7802  2.5168
## [5,] -2.0949 -26.2994 -0.1954
##
## Weights W of the restricted VAR:
##
##          [,1]    [,2]    [,3]
## p1.d  -0.0577  0.0012 -0.0115
## p2.d   0.0004 -0.0030 -0.0013
## e12.d  0.1323 -0.0121 -0.0353
## i1.d   0.0321 -0.0069 -0.0200
## i2.d   0.0683  0.0094 -0.0136
```

```
H32 = blrtest(z = H1, H = H.3.2, r = 2)
summary(H32)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Estimation and testing under linear restrictions on beta
##
```

```
## The VECM has been estimated subject to:
## beta=H*phi and/or alpha=A*psi
##
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    0    1    0    0
## [3,]    0    0    1    0
## [4,]    0    0    0    1
## [5,]    0    0    0   -1
##
## Eigenvalues of restricted VAR (lambda):
## [1] 0.2857 0.2542 0.1458 0.0927
##
## The value of the likelihood ratio test statistic:
## 13.71 distributed as chi square with 2 df.
## The p-value of the test statistic is: 0
##
## Eigenvectors, normalised to first column
## of the restricted VAR:
##
##          [,1]    [,2]    [,3]    [,4]
## [1,]   1.0000  1.0000  1.0000  1.0000
## [2,]  -1.1268 -1.2754 -1.6080  1.7357
## [3,]  -2.3411  1.1611 -0.0469 -5.6116
## [4,]  20.7157  2.9055 -0.7435  7.3346
## [5,] -20.7157 -2.9055  0.7435 -7.3346
##
## Weights W of the restricted VAR:
##
##          [,1]    [,2]    [,3]    [,4]
## p1.d   0.0013 -0.0026 -0.0460 -0.0029
## p2.d   0.0001 -0.0138  0.0181 -0.0008
## e12.d  0.0004 -0.0542 -0.0558 -0.0018
## i1.d  -0.0065 -0.0101  0.0077 -0.0053
## i2.d   0.0139  0.0046  0.0502 -0.0049
```

Dans H31: toutes les relations de cointégration ne peuvent être rejetées. Dans H32 : identifiant dans toutes les relations de cointégration doivent être rejetées.

5 Modèle SVEC OU SVECM

Le modèle SVECM est un modèle B avec $\mu_t = B\varepsilon_t$ et $\sum_\mu = BB'$. Pour l'identification unique de B, au moins les restrictions $\frac{1}{2}K(K-1)$ sont obligatoires. Ainsi, le théorème de Granger se présente comme suit :

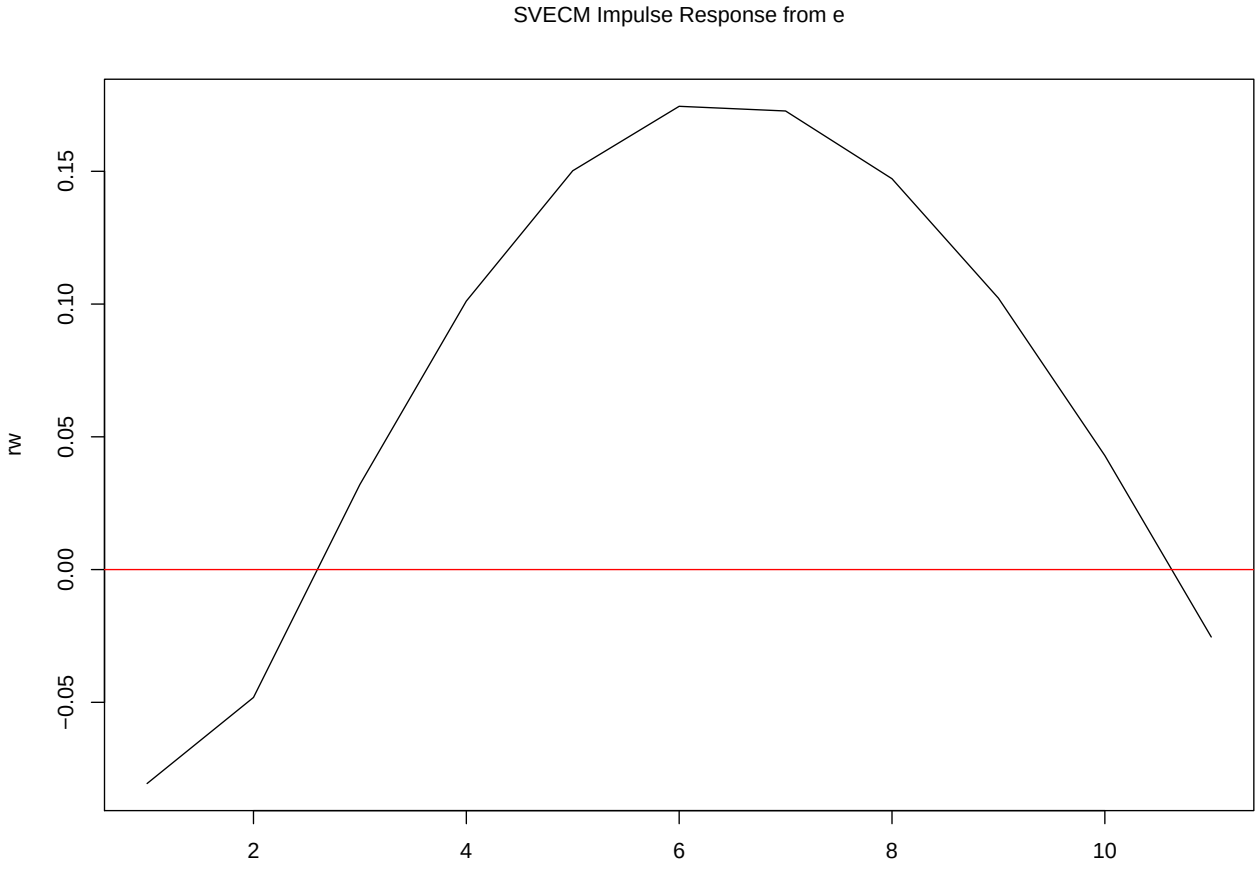
$$y_t = \Xi \sum_{i=1}^{t} \mu_i + \sum_{j=0}^{\infty} \Xi_j \mu_{t-j} + y_0$$

```
data(Canada)
vec.canada = ca.jo(Canada, K = 2, spec = "transitory", season = 4)
LR = matrix(0, nrow = 4, ncol = 4)
LR[, c(1, 2)] = NA
SR = matrix(NA, nrow = 4, ncol = 4)
SR[3, 4] = 0
```

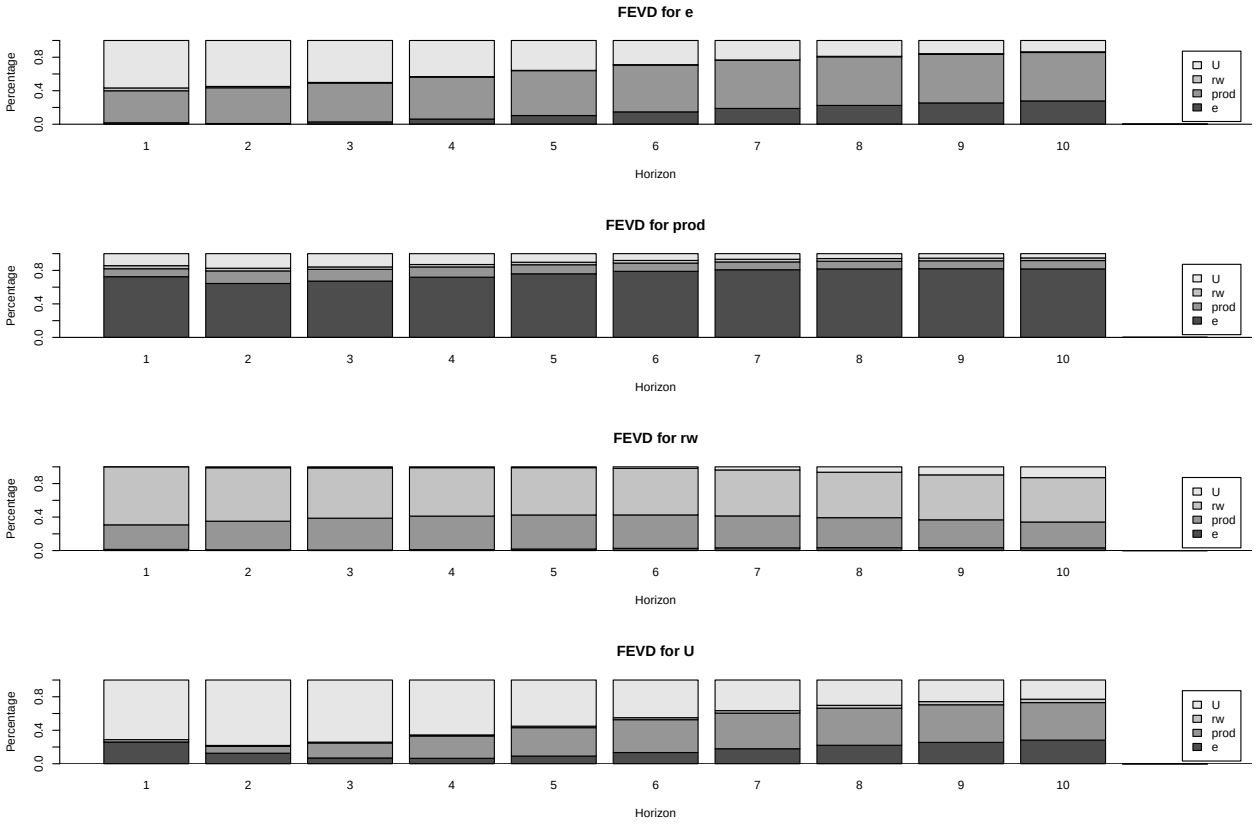
```
SR[4, 2] = 0
svecm.canada = SVEC(vec.canada, r = 2, LR = LR, SR = SR, max.iter = 200, lrtest = TRUE,
             boot = FALSE)
```

```
## Warning in SVEC(vec.canada, r = 2, LR = LR, SR = SR, max.iter = 200, lrtest
## = TRUE, : The SVEC is just identified. No test possible.
```

```
svecm.irf = irf(svecm.canada, impulse = "e", response = "rw", boot = FALSE, cumulative = FALSE,
                runs = 100)
plot(svecm.irf)
```




```
svecm.fevd = fevd(svecm.canada)
plot(svecm.fevd)
```



Modélisation de la volatilité

1 Modèles ARCH et GARCH

Les modèles hétéroscedastiques conditionnels autocorrélés (ARCH) a été introduite dans la littérature par Engle (1982). Ce type de modèle a depuis été modifié et étendu de plusieurs manières. Les articles d'Engle et Bollerslev (1986), Bollerslev et al. (1992) et Bera et Higgins (1993) donnent un aperçu des extensions de modèle au cours de la décennie suivant le document original. Aujourd'hui, les modèles ARCH ne sont pas seulement bien établis dans la littérature universitaire, ils sont également largement appliqués dans le domaine de la modélisation des risques. Dans cette partie, le terme *ARCH* est utilisé à la fois pour le modèle ARCH spécifique et pour ses extensions et modifications. Mais soulignons qu'à même que, le point de départ des modèles ARCH est une équation d'attente qui ne s'écarte que de la régression linéaire classique en ce qui concerne l'hypothèse d'erreurs indépendantes et identiquement normalement distribuées:

$$y_t = x_t'\beta + \varepsilon_t$$
$$\varepsilon_t|\Psi_{t-1} \sim N(0, h_t)$$

La deuxième composante des modèles ARCH est l'équation de la variance. Dans un modèle ARCH d'ordre q, la variance conditionnelle est expliquée par l'historique des erreurs carrées jusqu'au décalage temporel q:

$$h_t = \alpha_0 + \alpha_1\varepsilon_{t-1}^2 + ... + \alpha_q\varepsilon_{t-q}^2$$

Où $\alpha_0 > 0, \alpha_i \geqslant 0, i = 1, 2, ..., q$. Ces restrictions de paramètres garantissent une variance conditionnelle positive. L'inclusion des informations disponibles jusqu'au temps $t-1$ est évidente à partir de:

$$\varepsilon_{t-i} = y_{t-1} - x'_{t-i}\beta, i = 1, 2, ..., q$$

On peut déjà en déduire pourquoi cette classe de modèles peut capturer le fait stylisé (stylised facts) du clustering de volatilité: la variance conditionnelle est expliquée par les erreurs des périodes passées. Si ces erreurs sont importantes en valeur absolue, il en résulte une valeur élevée pour la variance conditionnelle, et inversement.

Après avoir discuté du modèle ARCH de base, l'accent est maintenant mis sur sa modification et ses extensions. Bollerslev (1986) a introduit le modèle *GARCH(p,q)* dans la littérature. Cela diffère du modèle ARCH en ce qui concerne l'inclusion de variables endogènes retardées dans l'équation de la variance. En d'autres termes, la variance conditionnelle dépend non seulement des erreurs carrées passées, mais également des variances conditionnelles retardées:

$$h_t = \alpha_0 + \alpha_1\varepsilon_{t-1}^2 + ... + \alpha_q\varepsilon_{t-q}^2 + \beta_1 h_{t-1} + ... + \beta_p h_{t-p}$$

avec comme restrictions $\alpha_0 > 0, \alpha_i \geqslant 0, i = 1, 2, ..., q$ et $\beta_i \geqslant 0, j = 1, 2, ..., p$ de sorte que le processus de variance conditionnelle soit strictement positif.

Nelson (1991) a aussi élargi le modèle ARCH pour permettre de prendre en compte cet effet. Il a proposé la classe de modèles de GARCH (EGARCH) exponentiels pour capturer des asymétries des certaines informations. La modélisation des effets asymétriques peut être justifiée d'un point de vue économique par des effets de levier, en particulier lorsqu'on étudie les rendements des actions. Pour ces derniers, une relation négative entre la volatilité et les rendements passés est postulée (voir Black 1976). L'équation de variance prend maintenant la forme :

$$log(h_t) = \alpha_0 + \sum_{i=1}^{q} \alpha_i g(\eta_{t-i}) + \sum_{j=1}^{p} \beta_j log(h_{t-j})$$

où la spécification alternative des modèles ARCH introduite par Bollerslev (1986) est utilisée et où la fonction $g(\eta_t)$ est définie comme suit:

$$g(\eta_t) = \theta\eta_t + \gamma[|\eta_t| - E(|\eta_t|)]$$

Dans R, le package *bayesGARCH* implémente l'estimation bayésienne des modèles *GARCH(1,1)* avec les innovations de Student (voir par exemple, Ardia 2008, 2009, 2015; Ardia et Hoogerheide 2010). Nous reprenons ici l'exemple de Stock et Watson. Cet ensemble de données est un complément à la monographie de Stock et Watson (2007) et figure dans le package AER (voir Kleiber et Zeileis 2008). Il s'agit donc d'un *back-test* du déficit attendu à un niveau de confiance de 99% pour les rendements quotidiens de la Bourse de New York (NYSE).

```
knitr::opts_chunk$set(echo = FALSE)
library(AER)
library(timeSeries)
library(fGarch)
data("NYSESW")
head(NYSESW)
```

```
## 1990-01-02 1990-01-03 1990-01-04 1990-01-05 1990-01-08 1990-01-09
##    2093.60    2091.48    2075.52    2058.07    2065.37    2044.43
```