Séries temporelles : TD et Examens

2024-11-24

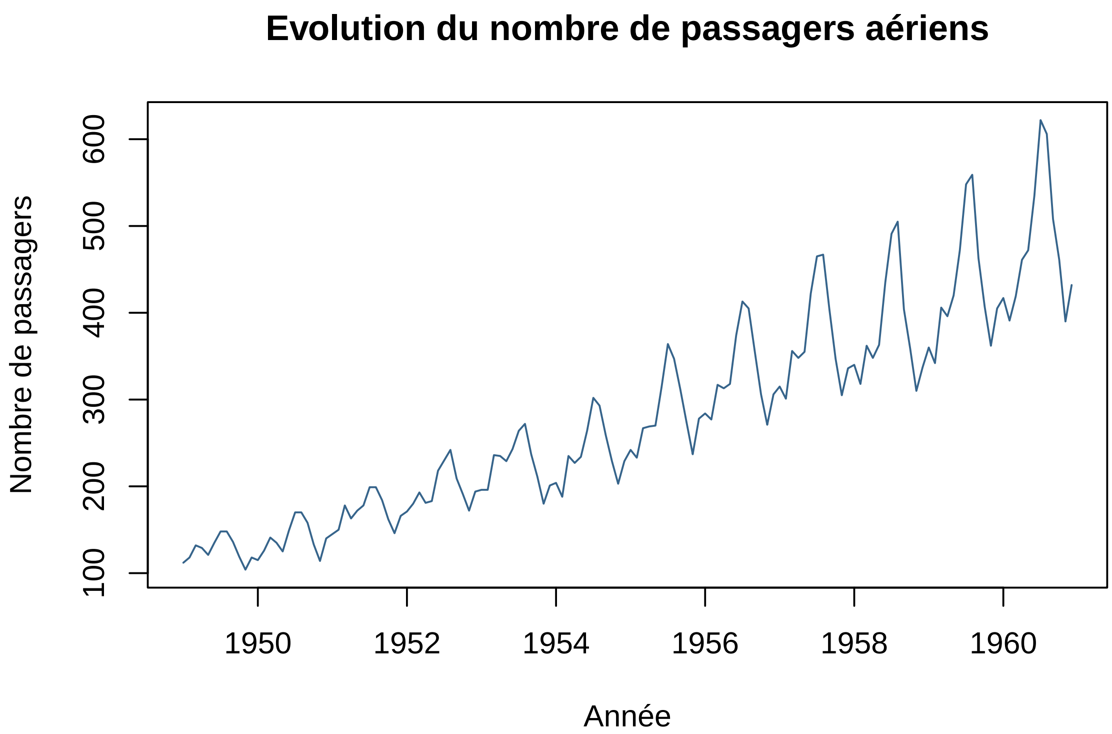
Contents

1 Avants propos, notions cours et codes associés :

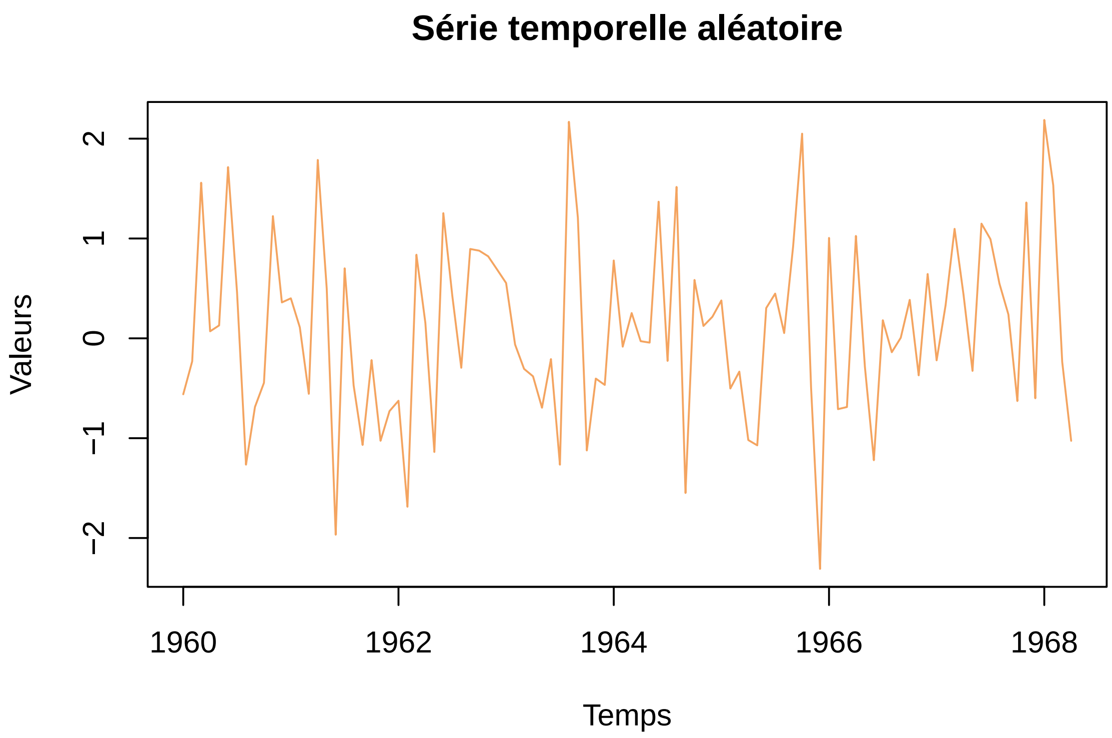
1.1 Le type ts 'time series' :

1.1.1 Exemples et création :

```r
# Exemple de série temporelle :
plot(
  AirPassengers,
  main = "Evolution du nombre de passagers aériens",
  ylab = "Nombre de passagers",
  xlab = "Année",
  type = "l",
  col = palette_couleur[1]
)
```

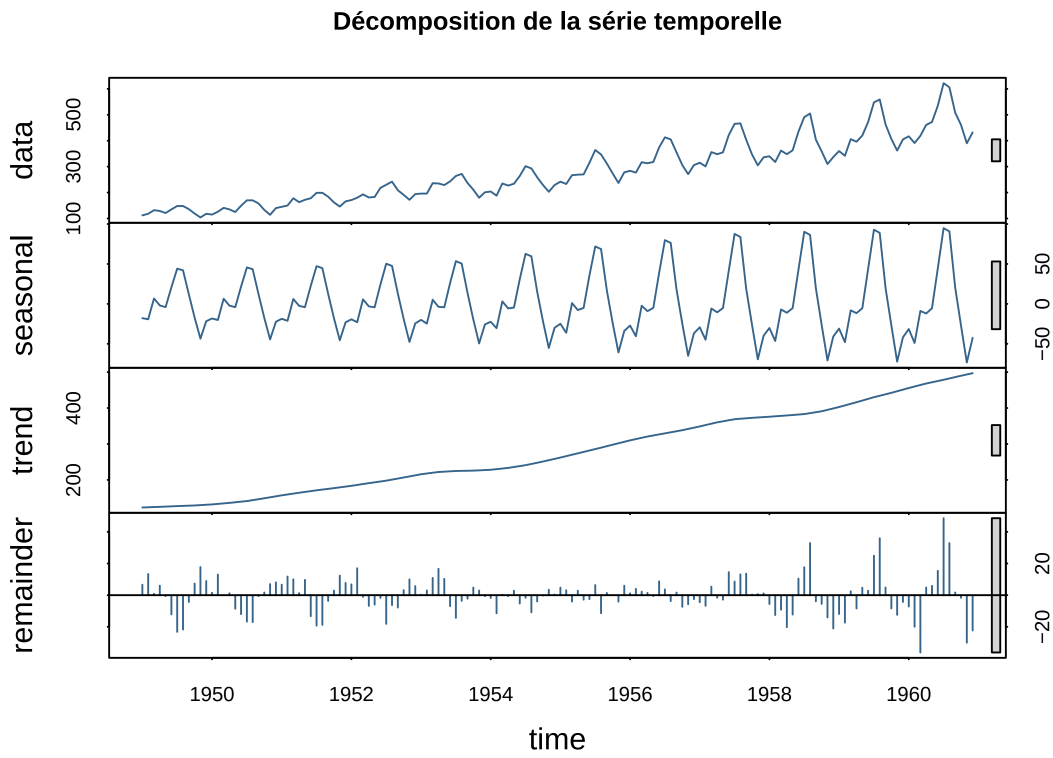



```r
# Création d'un objet time série aléatoire :

serie_temp = ts(rnorm(100), start = c(1960, 1), frequency = 12)
plot(
  serie_temp,
  main = "Série temporelle aléatoire",
  ylab = "Valeurs",
  xlab = "Temps",
  type = "l",
  col = palette_couleur[2]
)
```



1.1.2 Décomposition d'une série temporelle :

```r
plot(
  stl(AirPassengers, s.window = 12),
  main = "Décomposition de la série temporelle",
  col = palette_couleur[1],
  lwd = 1
)
```

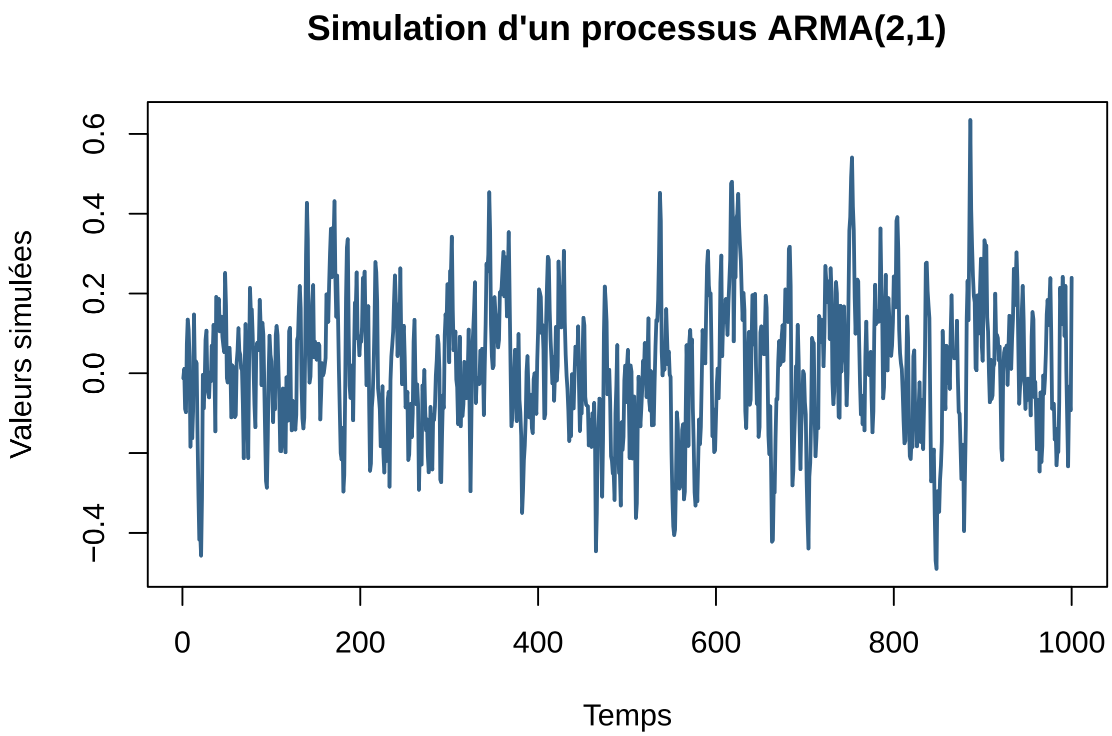



```
# s.window = 12 : période de saisonnalité (nb de mois)
```

1.2 Simulation de processus :

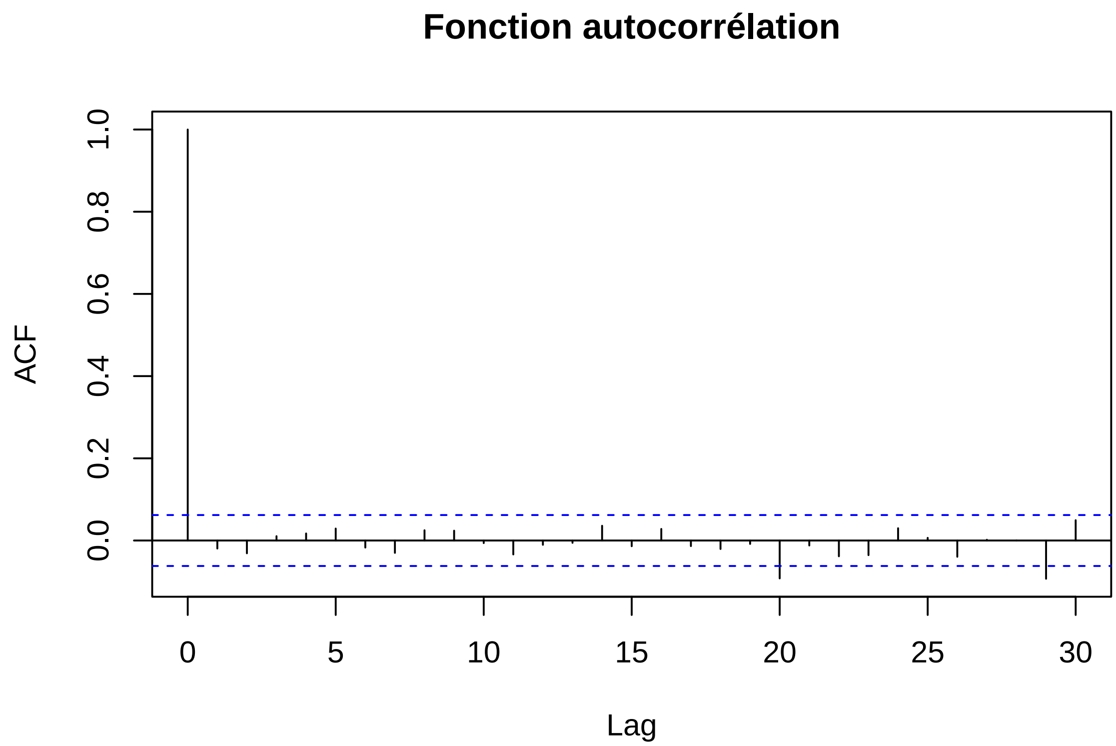
1.2.1 Simulation ARMA(2,1) :

```
alpha = c(.5, .2) #paramètres AR
beta = .5 #paramètres MA
sig = .1 #écart-type du bruit
Time = 1000 #longueur de la série temporelle
x = arima.sim(model = list(ar = alpha, ma = beta),
              sd = sig,
              n = Time) #simulation modèle ARMA
plot(x, type = "l", col = palette_couleur[1], lwd = 2,
     main = "Simulation d'un processus ARMA(2,1)",
     ylab = "Valeurs simulées", xlab = "Temps")
```

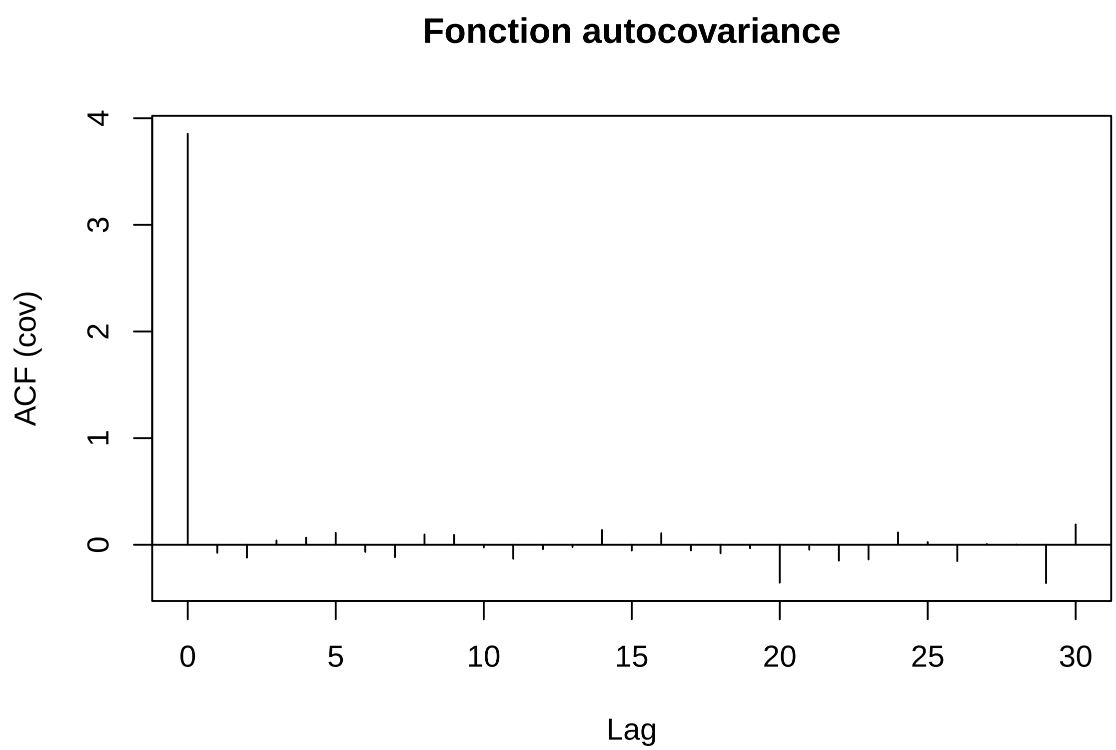


1.3 Fonctions d'autocorrélation et d'autocovariance :

```
Time = 1000
x = 2 * rnorm(Time) #simulation d'un bruit blanc gaussien
acf(x, main = "Fonction autocorrélation")
```



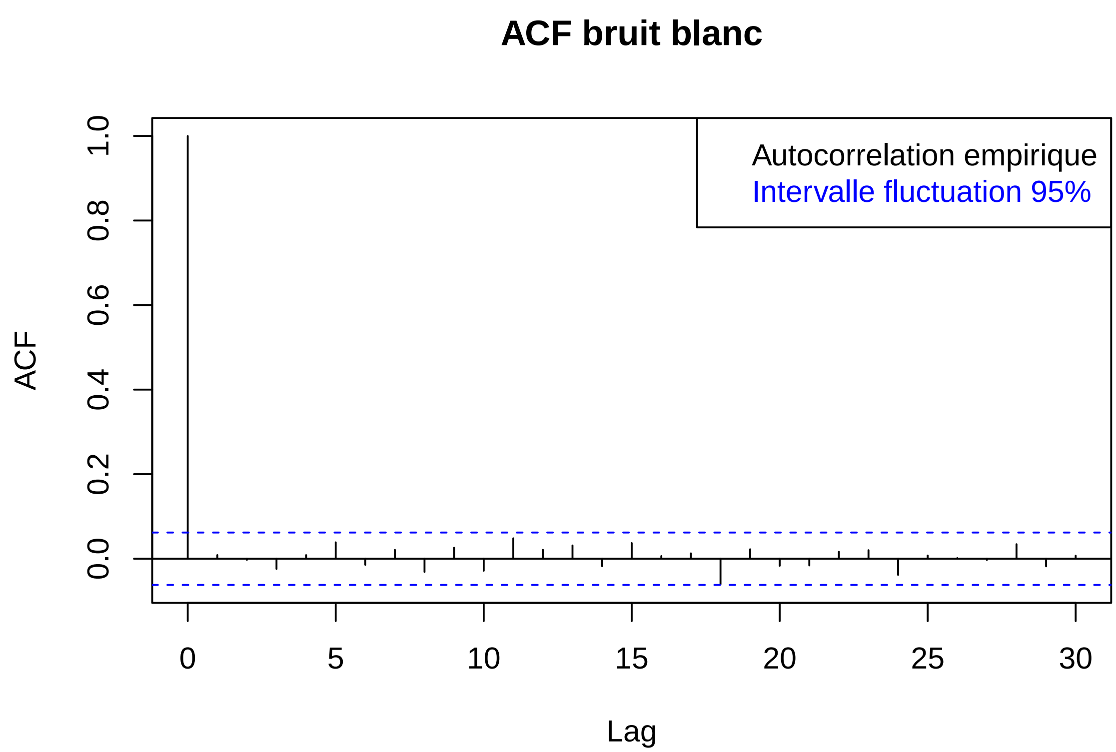

```r
acf(x, type = 'covariance', main = "Fonction autocovariance")
```



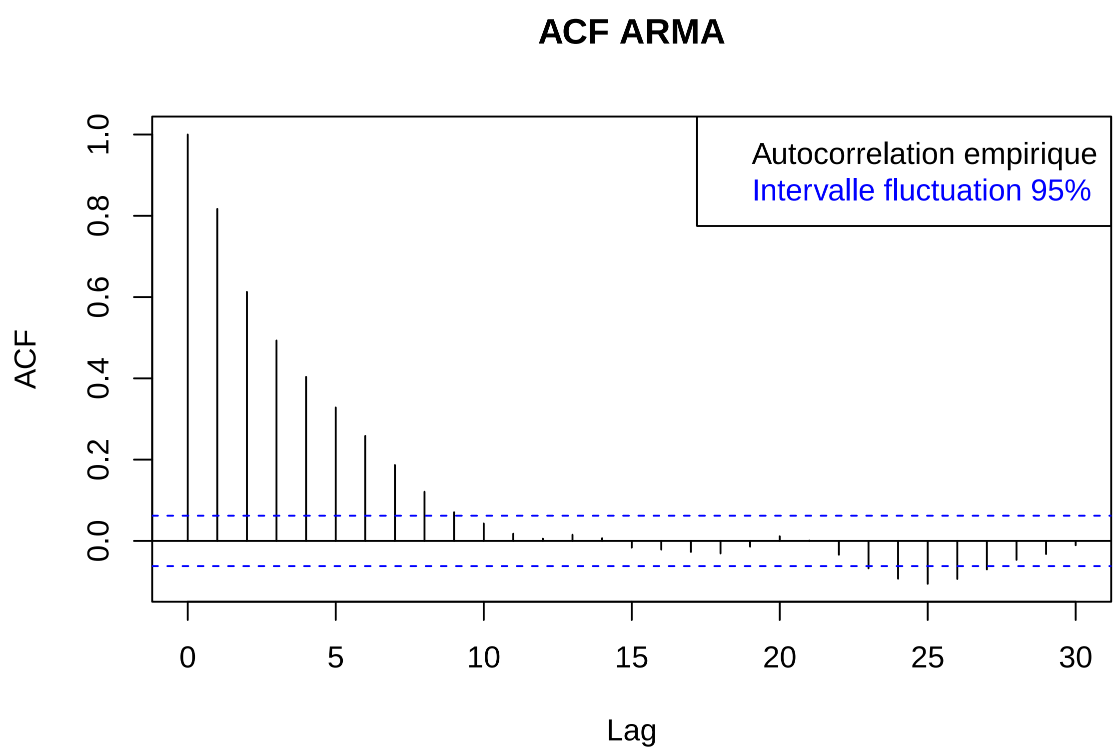
1.4 Etude d'un bruit blanc :

1.4.1 Fonctions d'autocorrélation et d'autocovariance :

```
# Simulation bruit blanc gaussien :
Time = 1000
bbg = rnorm(Time)

acf(bbg, main = 'ACF bruit blanc')
legend(
  'topright',
  c('Autocorrelation empirique', 'Intervalle fluctuation 95%'),
  text.col = c('black', 'blue'))
```




```
x = arima.sim(model = list(ar = c(.5, .2), ma = .5),
              sd = .1,
              n = Time)
acf(x, main = 'ACF ARMA')
legend(
  'topright',
  c('Autocorrelation empirique', 'Intervalle fluctuation 95%'),
  text.col = c('black', 'blue'))
```



1.4.2 Test statistiques sur le bruit blanc :

Le test de Box-Pierce permet de tester l'hypothèse nulle d'indépendance des observations d'une série temporelle.

Le test de Ljung-Box est une généralisation du test de Box-Pierce qui permet de tester l'indépendance des observations d'une série temporelle sur plusieurs retards.

$$\begin{cases} H_0 : \rho(1) = 0, \text{ X est un bruit blanc} \\ H_1 : \rho(1) \neq 0, \text{X n'est pas un bruit blanc} \end{cases}$$

```
Time = 1000
x = rnorm(Time) #simulation d'un bruit blanc gaussien
ar21 = arima.sim(model = list(ar = c(.5, .2), ma = .5),
              sd = .1, n = Time)

p1 = Box.test(x) #(test uniquement sur rho(1))
p2 = Box.test(x, lag = 10) #(test sur rho(1),...,rho(10))
p3 = Box.test(x, lag = 10, type = 'Ljung-Box') #(test de Ljung-Box)
p4 = Box.test(ar21) #(test uniquement sur rho(1))

p_value = round(c(p1$p.value, p2$p.value, p3$p.value, p4$p.value),4)

data.frame(
  Test = c(
    "BB Box rho(1)",
    "BB Box rho(1:10)",
```

```
    "BB Lunj-Box rho(1:10)",
    "Arma21 rho(1)"),
  p_value = p_value,
  Bruit_Blanc = p_value > .05
  )
```

```
                   Test p_value Bruit_Blanc
1         BB Box rho(1)  0.2375        TRUE
2      BB Box rho(1:10)  0.3597        TRUE
3 BB Lunj-Box rho(1:10)  0.3544        TRUE
4         Arma21 rho(1)  0.0000       FALSE
```

2 Travaux dirigés

2.1 Exercice 1: Simulation de différents procesus :

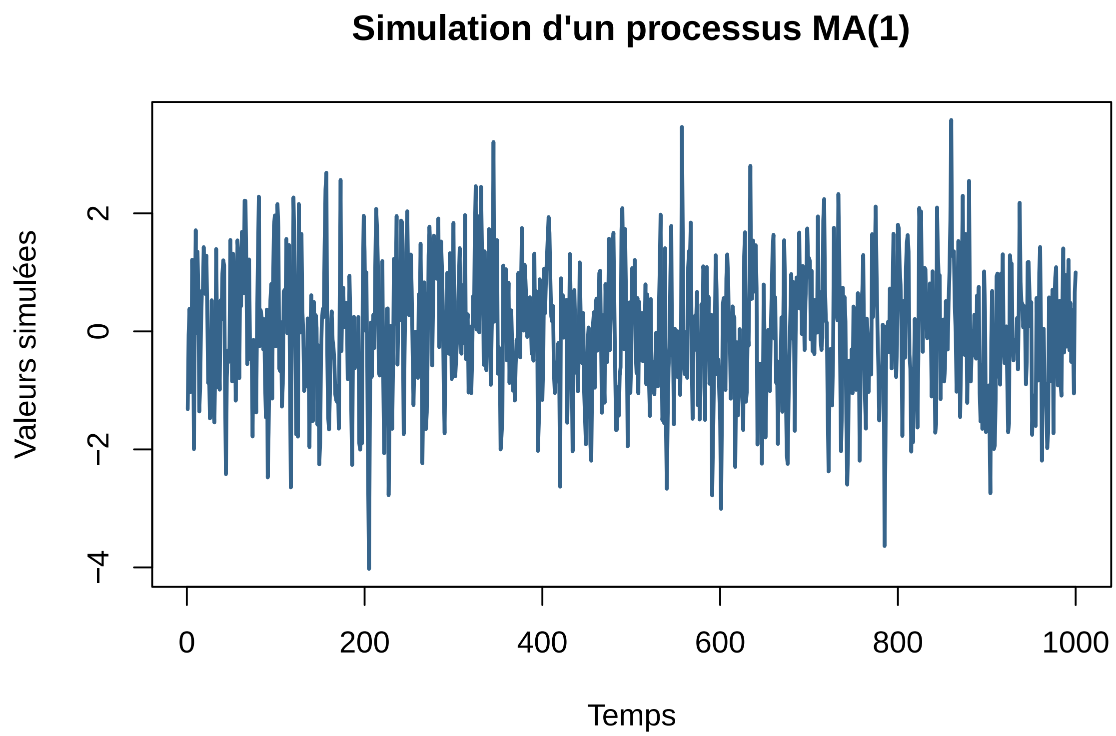
2.1.1 Simulation MA(1) :

$$A_t = \epsilon_t + \beta\epsilon_{t-1} \text{ avec } \epsilon_t \sim \mathcal{N}(0,1)$$

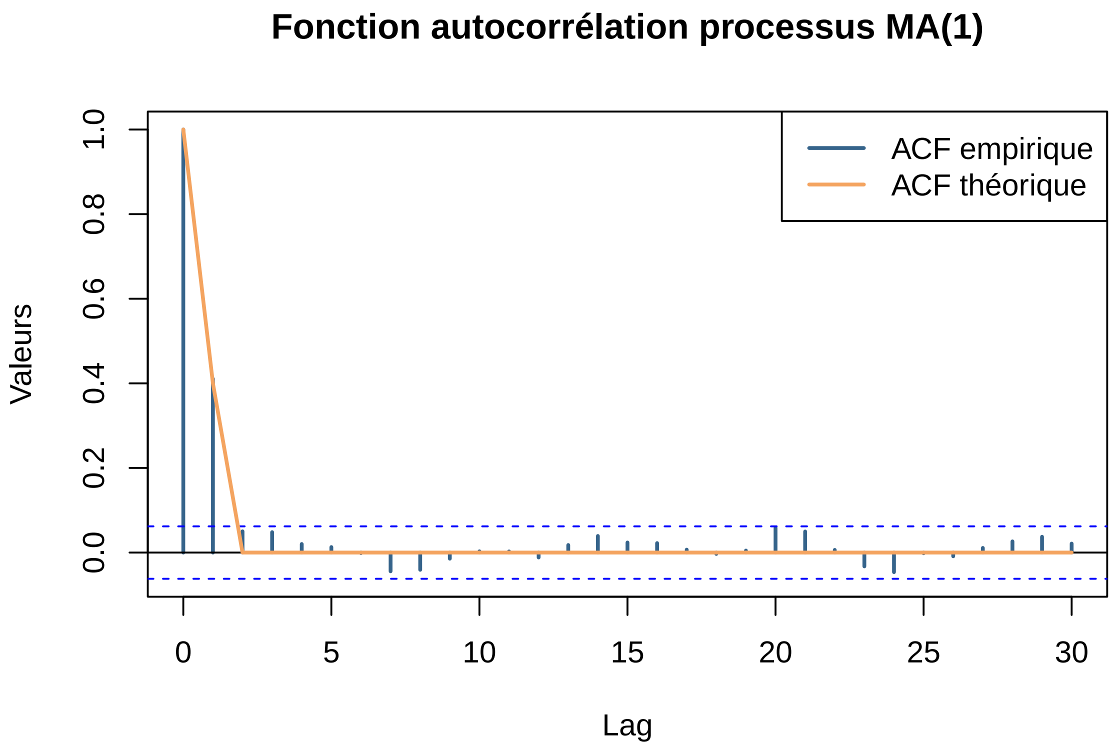
```
# Paramètres de la simulation :
Time = 1000
sigma = 1
beta = 0.5

# Simulation par formule :
esp = rnorm(Time +1, sd = sigma)
A = esp[2:(Time+1)] + beta * esp[1:Time]

# Simulation par fonction intégrée :
A2 = arima.sim(model = list(ma = beta), n = Time, sd = sigma)

# Acf Théorique d'un' processus MA(1) :
ARMAacf(ma = beta, lag.max = 20)
```

```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
1.0 0.4 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 20
0.0
```

```
# Représentation graphique :
plot(
  A,
  type = 'l',
  main = 'Simulation d\'un processus MA(1)',
  ylab = 'Valeurs simulées',
  xlab = 'Temps',
  col = palette_couleur[1],
  lwd = 2
)
```




```
ac = acf(A,
         main = "Fonction autocorrélation processus MA(1)",
         ylab = "Valeurs",
         col = palette_couleur[1],
         lwd = 2)   #acf empirique
lines(ac$lag, c(1, beta / (1 + beta ^ 2), rep(0, length(ac$lag) - 2)),
      col = palette_couleur[2],
      lwd = 2)  #acf théorique
#Alternative :
# lines(ac$lag,
#       ARMAacf(ma = beta, lag.max = ac$lag[length(ac$lag)]),
#       col = palette_couleur[3],
#       lwd = 2)  #acf alternative
legend(
  'topright',
  c('ACF empirique', 'ACF théorique'),
  col = palette_couleur[1:2],
  lty = 1,
  lwd = 2)
```



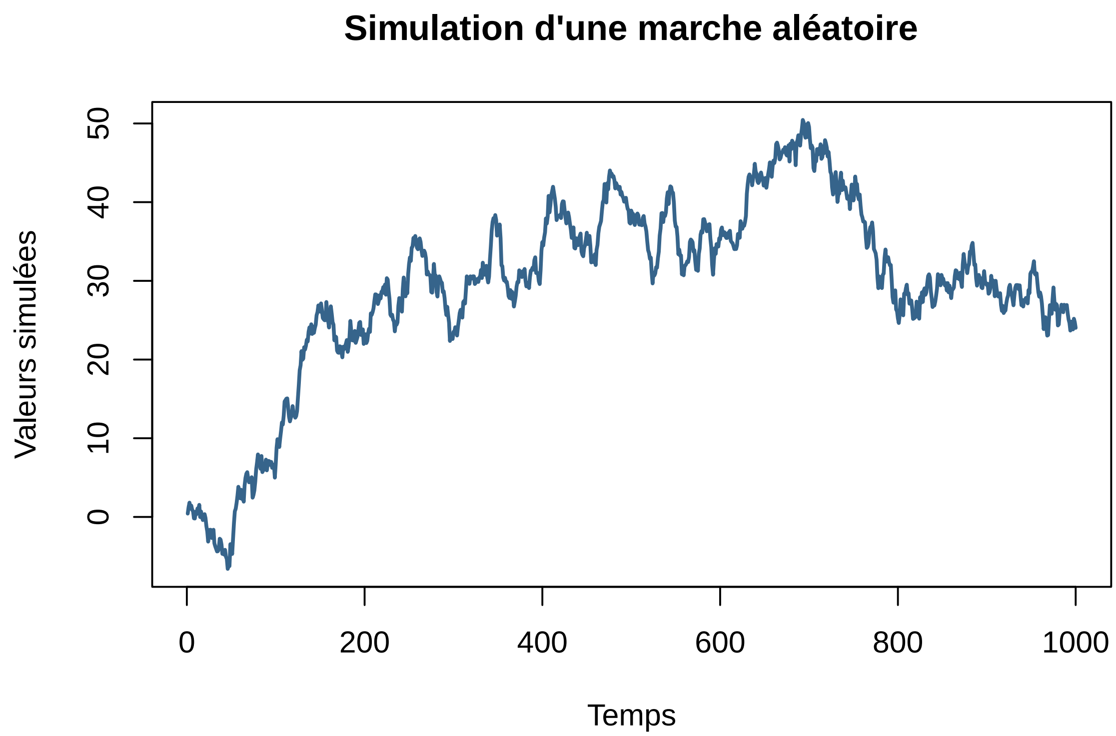
Commentaires :

L'acf (acf=fonction d'autocorrélation) empirique doit être proche de l'acf théorique si T est grand (estimateur consistant) L'acf montre l'existence d'une dépendance entre les valeurs successives, X_t et X_{t+h} sont non corrélées si $h > 1$. Les bornes de l'intervalle bleu sont égales à +-1.96/$T^{0.5}$ Pour un bruit blanc, on doit avoir l'acp empirique dans l'intervalle bleu avec un proba de 95%. On retrouve que $\hat{\rho}(1)$ est (significativement) plus grand que ce qu'on attend pour un bruit blanc.

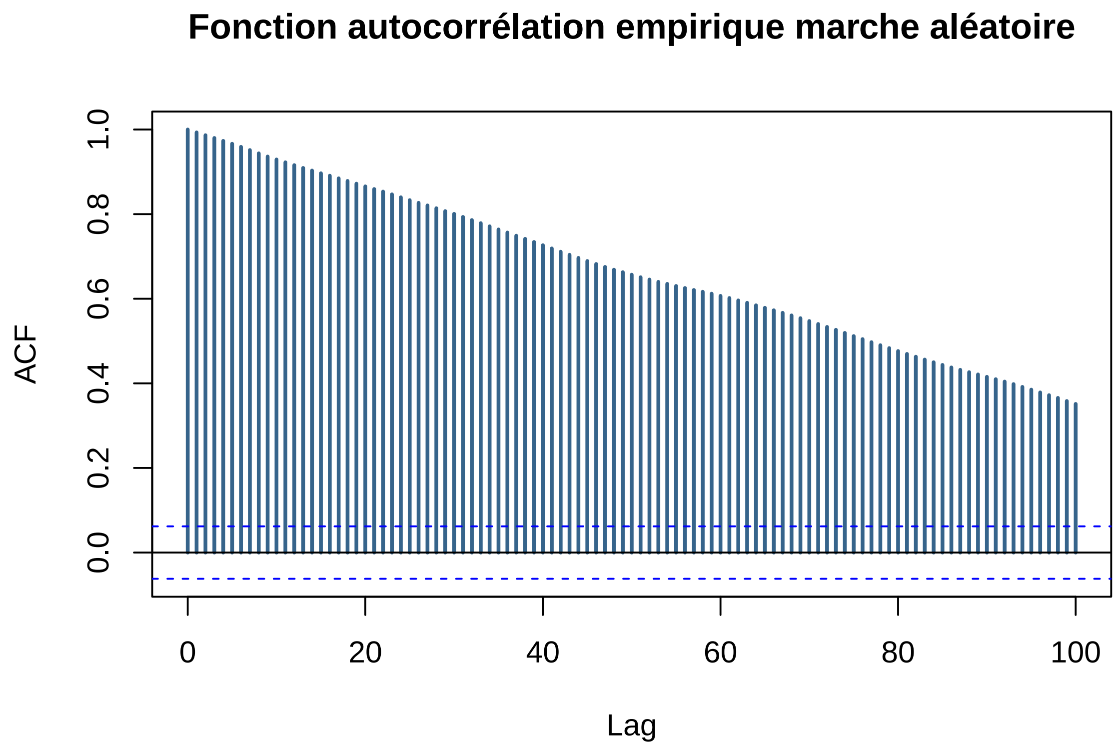
2.1.2 Simulation d'une marche aléatoire :

$$B_t = B_{t-1} + \epsilon_t \text{ avec } \epsilon_t \sim \mathcal{N}(0, 1)$$

```
sig = 1
tau = 1
Time = 1000
eps = rnorm(Time, sd = sig) #bruit blanc
B = rnorm(1, sd = tau) + cumsum(eps[1:Time])  #alternative : boucle
plot(
  B,
  type = 'l',
  main = 'Simulation d\'une marche aléatoire',
  ylab = 'Valeurs simulées',
  xlab = 'Temps',
  col = palette_couleur[1],
  lwd = 2
)
```




```r
acf(
  B,
  lag.max = 100,
  main = "Fonction autocorrélation empirique marche aléatoire",
  col = palette_couleur[1],
  lwd = 2
)
```

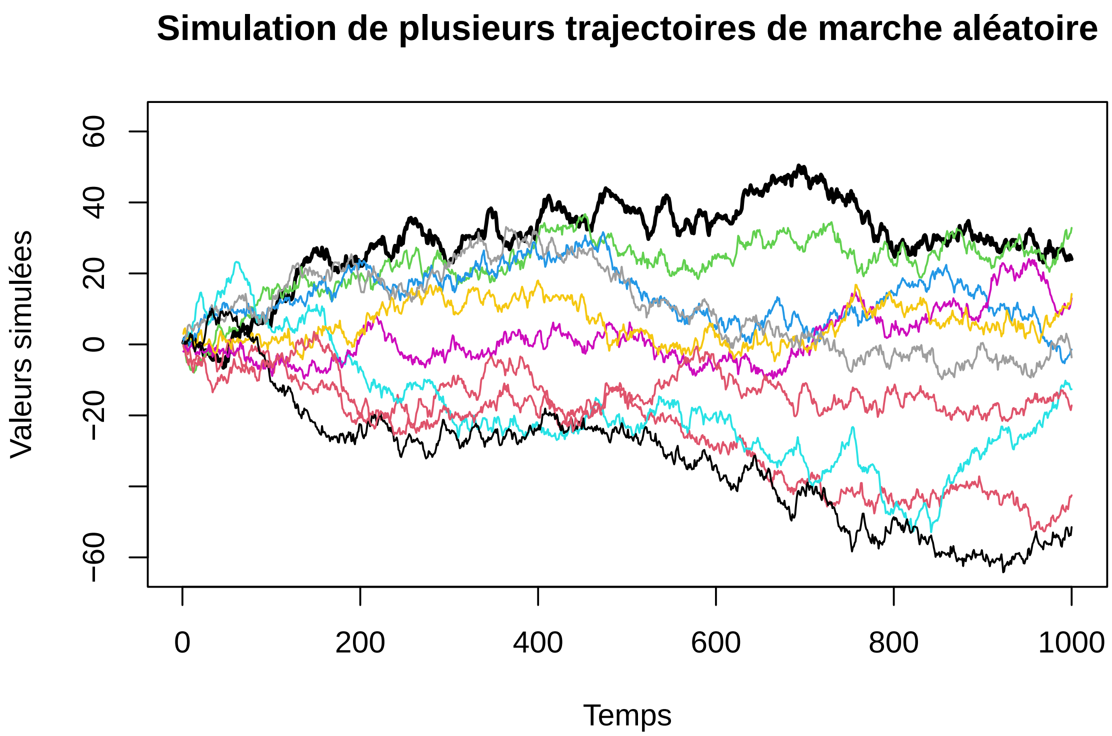


Commentaires :

Pprocessus non-stationnaire, ACF théorique pas définie, décroissance lente vers 0 de l'ACP empirique.

```
Time = 1000
plot(
  B,
  type = 'l',
  main = 'Simulation de plusieurs trajectoires de marche aléatoire',
  ylab = 'Valeurs simulées',
  xlab = 'Temps',
  col = 1,
  ylim = c(-2 * sig * sqrt(Time), 2 * sig * sqrt(Time)),
  lwd = 2
)
for (i in 2:10) {
  eps = rnorm(Time + 1, sd = sig)
  B = rnorm(1, sd = tau) + cumsum(eps[1:Time]) #on prend tau=sigma
  lines(B, col = i)
}
```

2.1.2.1 Simulation de plusieurs trajectoires de marche aléatoire :



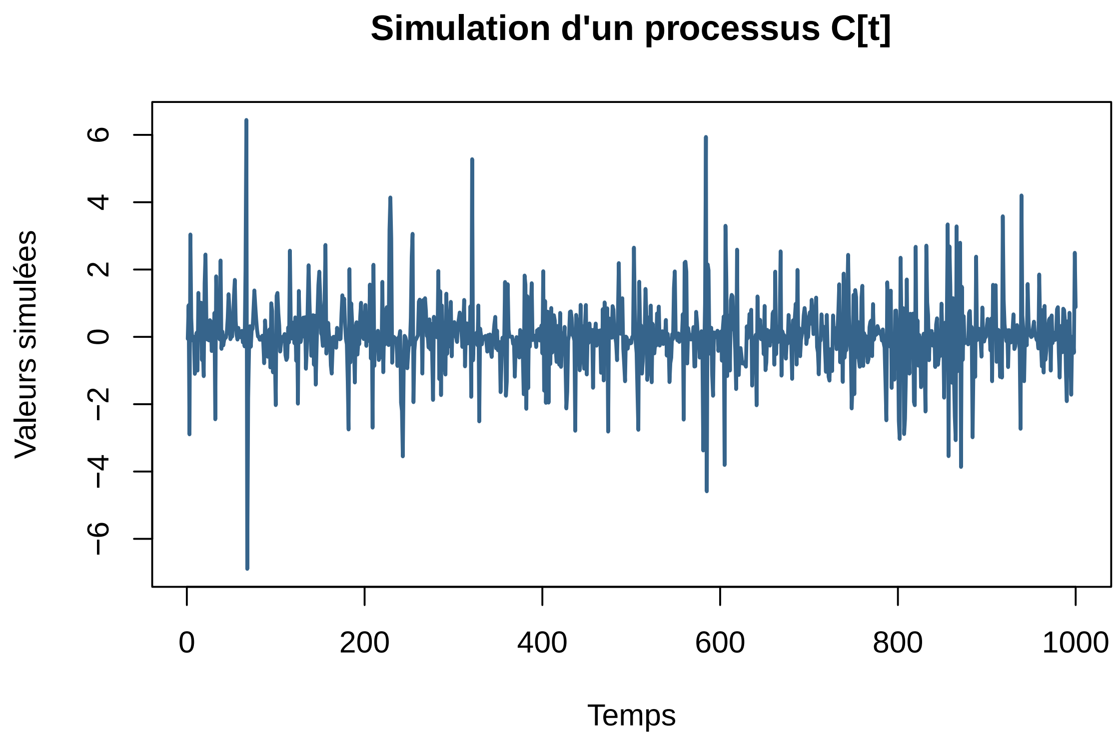
Commentaire :

On retrouve que la variance augmente avec le temps, processus non-stationnaire

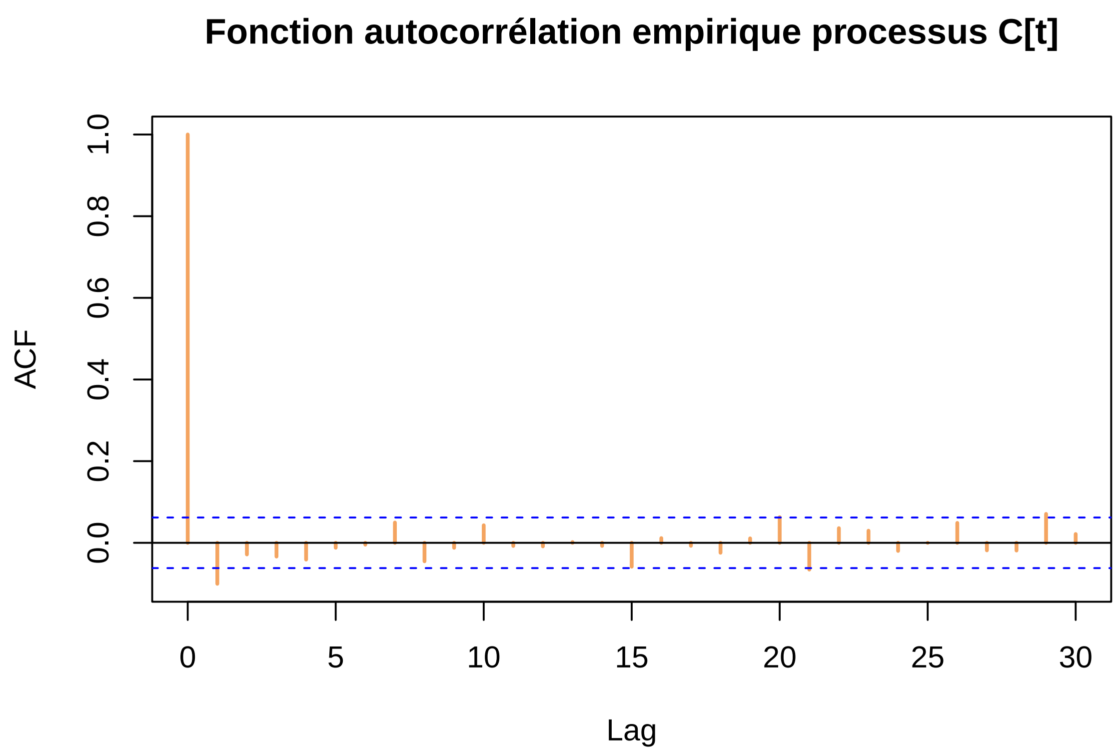
2.1.3 Simulation du troisième processus :

$$C_t = \epsilon_{t-1} \times \epsilon_t \text{ avec } \epsilon_t \sim \mathcal{N}(0,1)$$

```
Time = 1000
sigma = 1
mu = 0
esp = rnorm(Time + 1, sd = sigma , mean = mu)
C = esp[1:Time] * esp[2:(Time + 1)]

plot(C,
     main = "Simulation d'un processus C[t]",
     ylab = "Valeurs simulées",
     xlab = "Temps",
     type = "l",
     col = palette_couleur[1],
     lwd = 2)
```




```r
acf(C,
    main = "Fonction autocorrélation empirique processus C[t]",
    col = palette_couleur[2],
    lwd = 2)
```



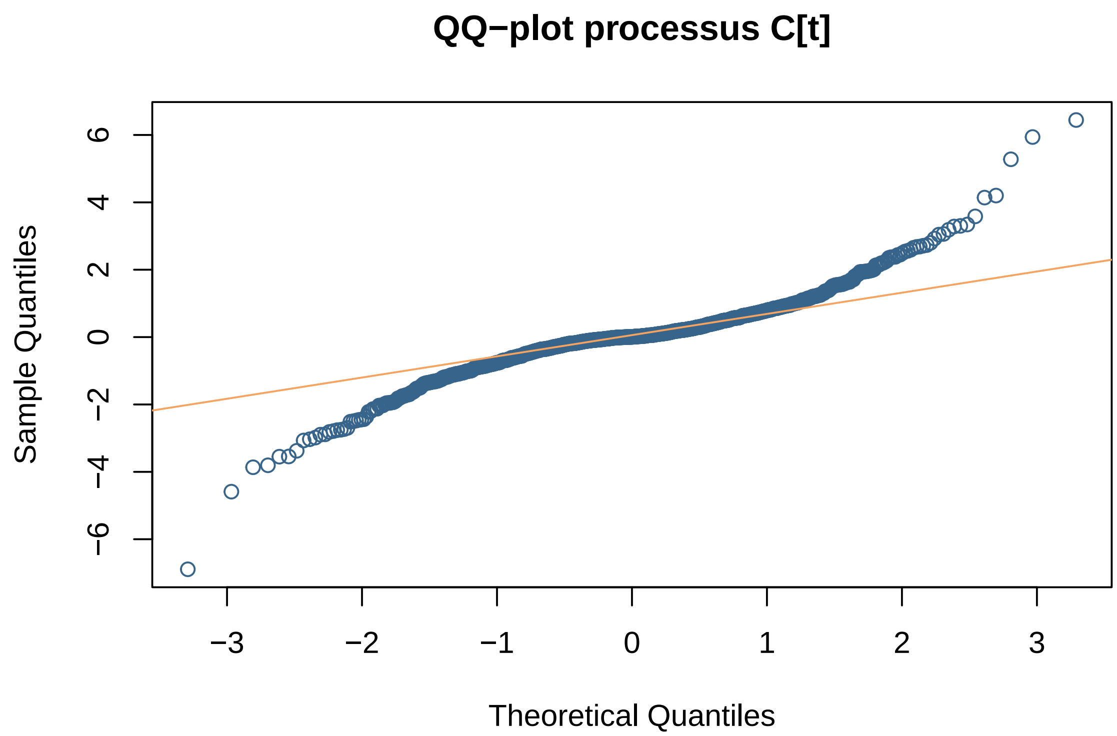
2.1.3.1 Test de normalité des résidus : On peut tester la normalité des résidus avec le test de Shapiro-Wilk.

Le test de shapiro test l'existence de normalité des résidus.

$$\begin{cases} H_0 : \text{Les résidus suivent une loi normale} \\ H_1 : \text{Les résidus ne suivent pas une loi normale} \end{cases}$$

On peut aussi tracer un QQ-plot pour vérifier la normalité des résidus.

```
#Etude de la distribution des valeurs simulées :
qqnorm(C,
       main = "QQ-plot processus C[t]",
       col = palette_couleur[1])
qqline(C, col = palette_couleur[2])
```



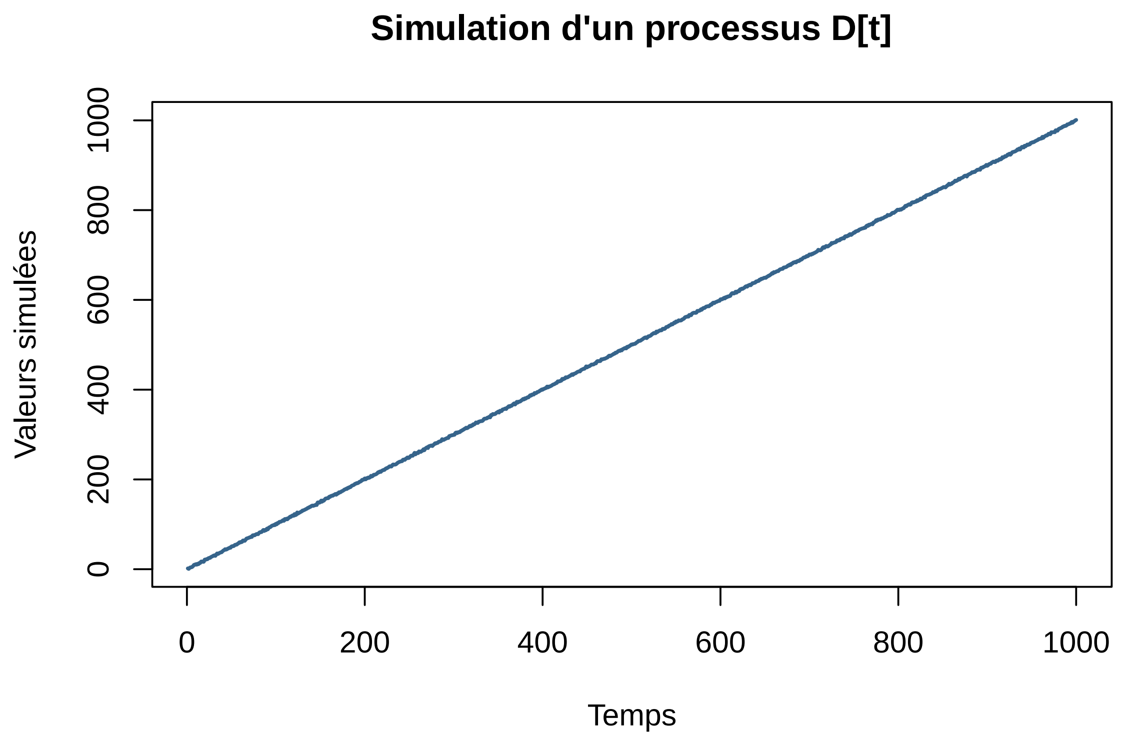

```r
shapiro.test(C) #test de normalité des résidus
```

```
    Shapiro-Wilk normality test

data:  C
W = 0.914, p-value < 2.2e-16
```

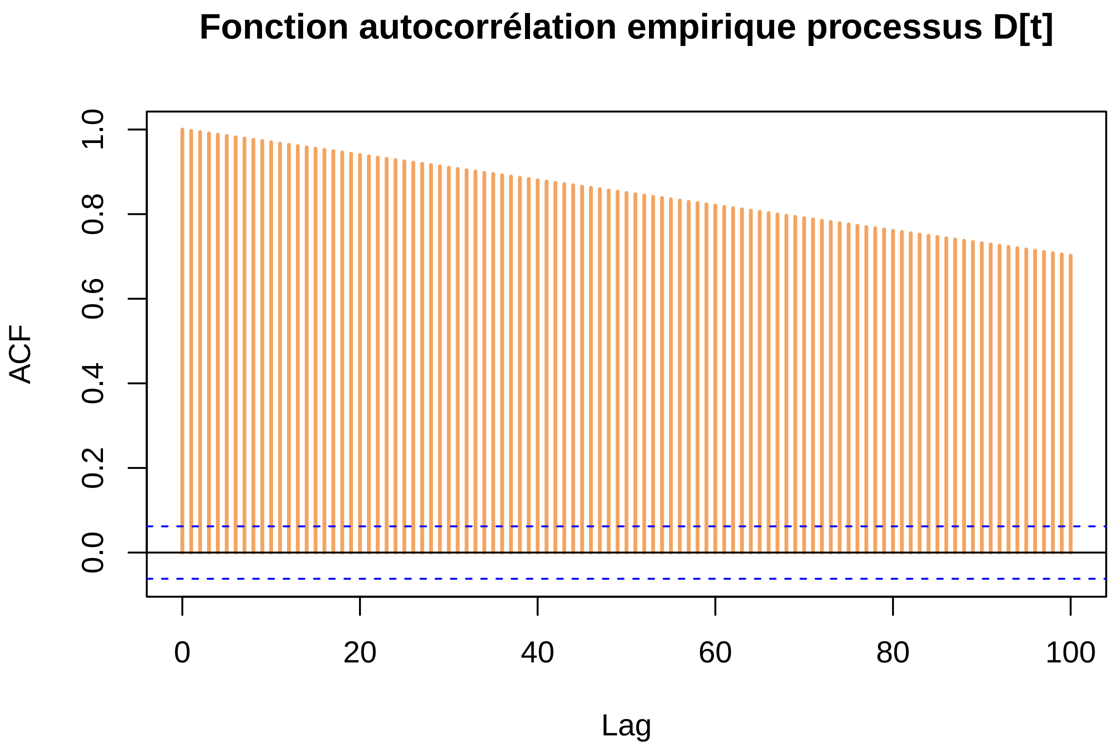
2.1.4 Simulation du processus D :

$$D_t = t + \epsilon_t \text{ avec } \epsilon_t \sim \mathcal{N}(0,1)$$

```r
sigma = 1
Time = 1000
esp = rnorm(Time, sd = sigma)

D = 1:Time + esp[1:Time]

plot(D,
     main = "Simulation d'un processus D[t]",
     ylab = "Valeurs simulées",
     xlab = "Temps",
     type = "l",
     col = palette_couleur[1],
     lwd = 2)
```




```r
acf(D,
    main = "Fonction autocorrélation empirique processus D[t]",
    col = palette_couleur[2],
    lwd = 2,
    lag.max = 100)
```



Commentaires :

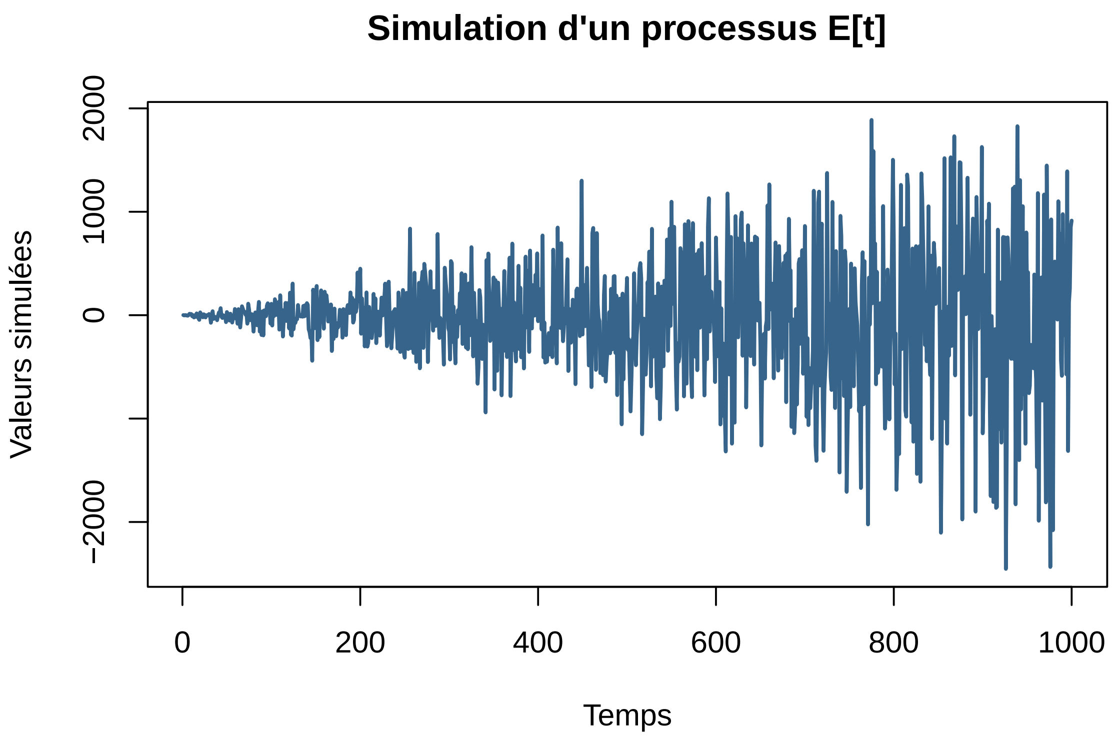
Le proccesus n'est pas stationnaire, il y a une tendance dans la variance.

En considérant $t \in R$ fixé on peut dire que le processus est gaussien.

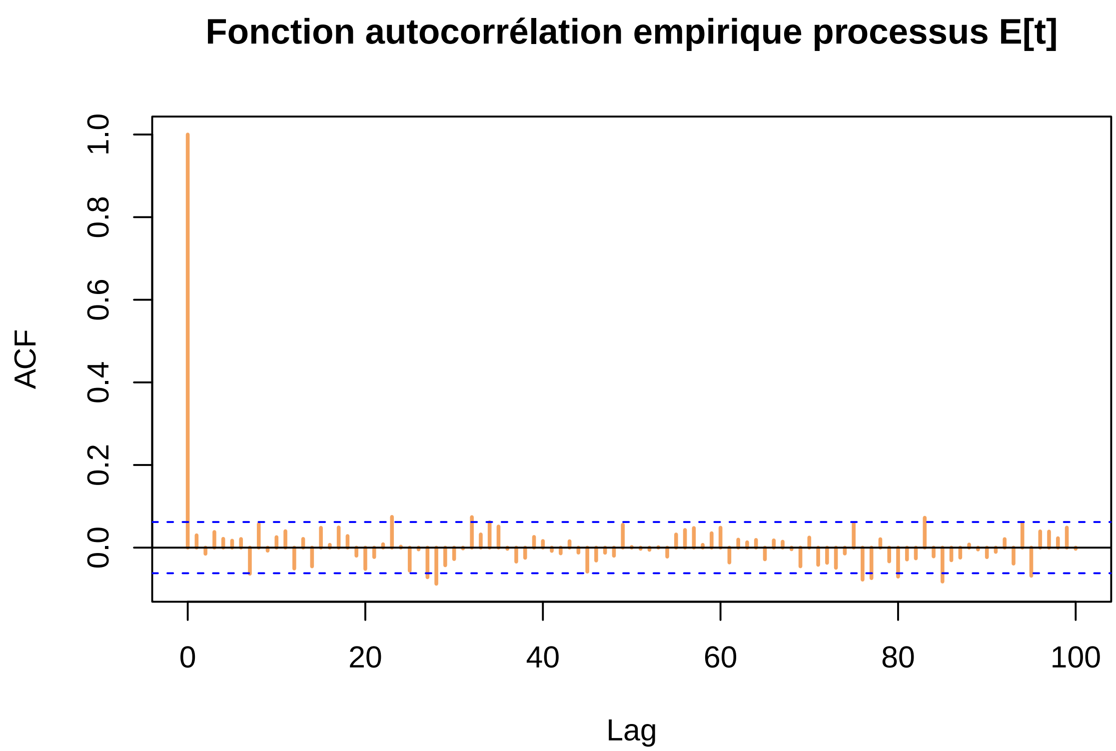
2.1.5 Simulation du processus E :

$$E_t = t \times \epsilon_t \text{ avec } \epsilon_t \sim \mathcal{N}(0, 1)$$

```
E = (1:Time) * esp[1:Time]

plot(E,
     main = "Simulation d'un processus E[t]",
     ylab = "Valeurs simulées",
     xlab = "Temps",
     type = "l",
     col = palette_couleur[1],
     lwd = 2)
```




```r
acf(E,
    main = "Fonction autocorrélation empirique processus E[t]",
    col = palette_couleur[2],
    lwd = 2,
    lag.max = 100)
```



Commentaire :

Le processus n'est pas stationnaire il y a une tendance dans la variance.

2.2 Exercice 2 :

$$X_t = \epsilon_t + \beta_1 * \epsilon_{t-1} + \beta_2 * \epsilon_{t-2} + \beta_3 * \epsilon_{t-3} \text{ avec } \epsilon_t \sim \mathcal{N}(0,1)\beta_1 = 1, \beta_2 = 0.5, \beta_3 = -0.5$$

2.2.1 Fonction autocorrélation théorique :

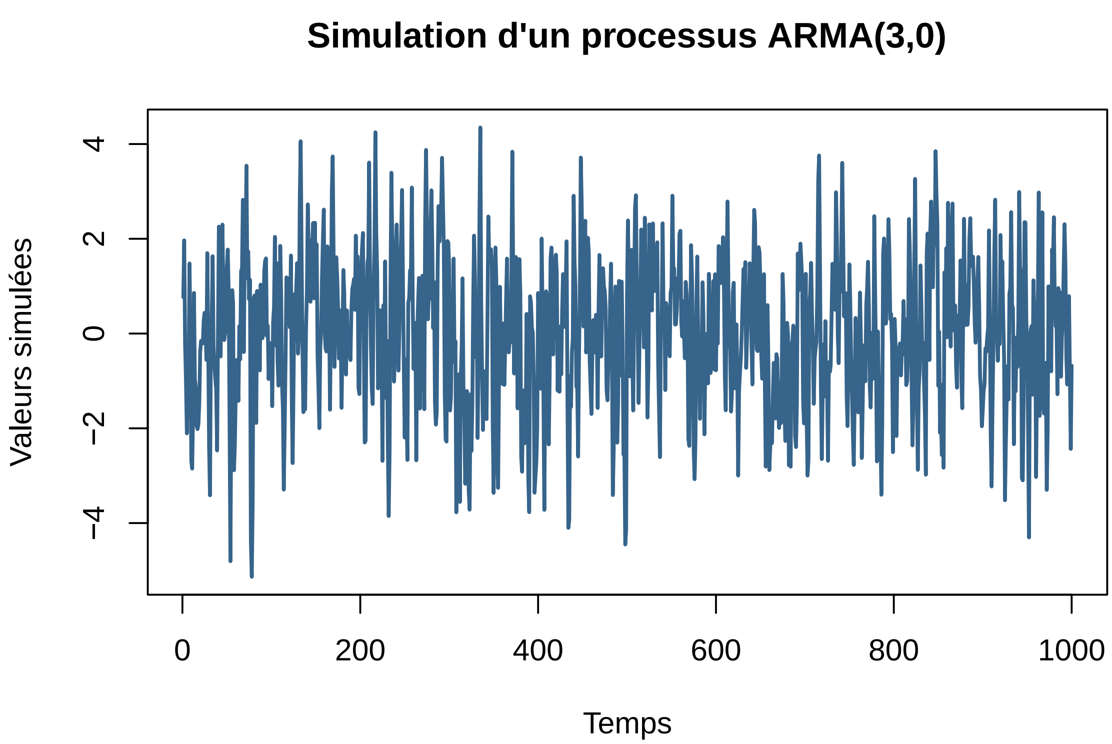
```
# Paramètres du modèle :
beta = c(1, .5, -.5)
rho = ARMAacf(ma = beta)
# Formule théoriques :
rho_1 <- (beta[1]+beta[1]*beta[2]+beta[2]*beta[3])/(1+sum(beta^2))
rho_2 <- (beta[2]+beta[1]*beta[3])/(1+sum(beta^2))
rho_3 <- (beta[3])/(1+sum(beta^2))

# Affichage des résultats :

data.frame(
  Lag = 1:3,
  Formule = c(rho_1, rho_2, rho_3),
  Fonction = rho[2:4]
)
```

```
  Lag Formule Fonction
1   1     0.5      0.5
```

```
2   2     0.0      0.0
3   3    -0.2     -0.2
```

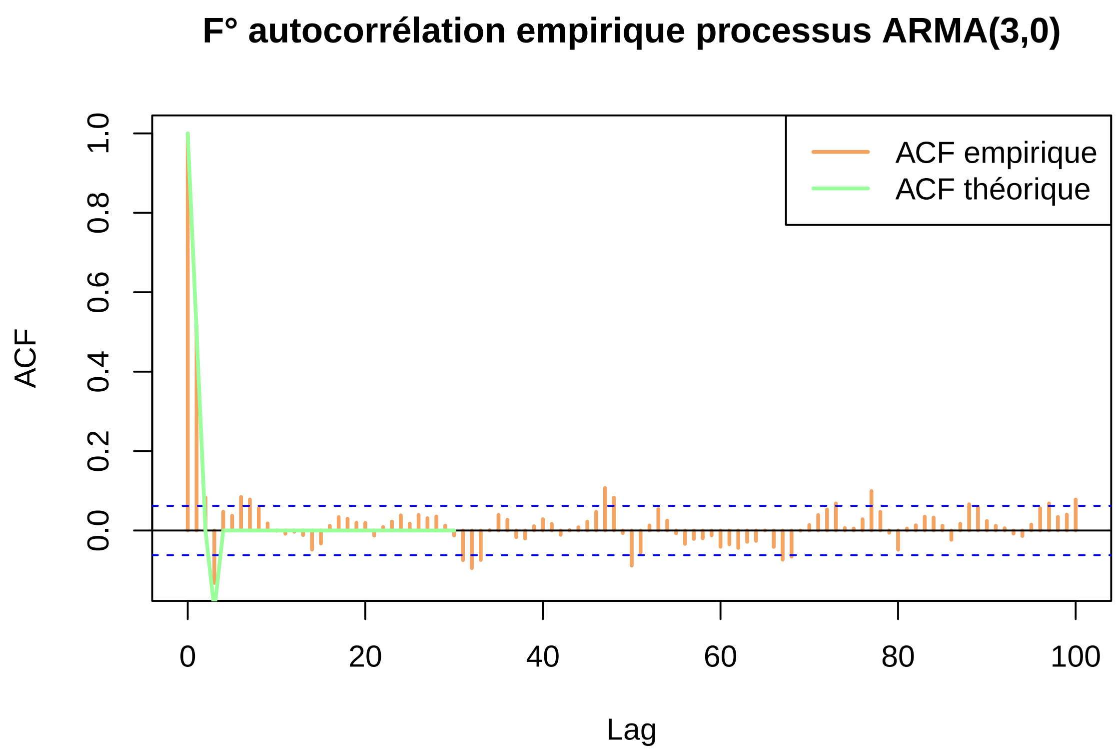
2.2.2 Simulation du processus en utilisant la fonction intégrée :

```
x = arima.sim(model = list(ma = beta), n = 1000)
plot(x,
     main = "Simulation d'un processus ARMA(3,0)",
     ylab = "Valeurs simulées",
     xlab = "Temps",
     type = "l",
     col = palette_couleur[1],
     lwd = 2)
```




```
acf(x,
    main = "F° autocorrélation empirique processus ARMA(3,0)",
    col = palette_couleur[2],
    lwd = 2,
    lag.max = 100)

rho = ARMAacf(ma = c(1, .5, -.5), lag.max = 30)
lines(0:30, rho, col = palette_couleur[3], lwd = 2)
legend(
  'topright',
  c('ACF empirique', 'ACF théorique'),
  col = palette_couleur[2:3],
  lty = 1,
  lwd = 2
```

```
)
```



A retenir :

Pour un MA(q), on a rho(h)=0 pour h>q.

2.3 Exercice 3 :

$$X_t = \alpha X_{t-1} + \epsilon_t \text{ avec } \epsilon_t \sim \mathcal{N}(0,1)$$

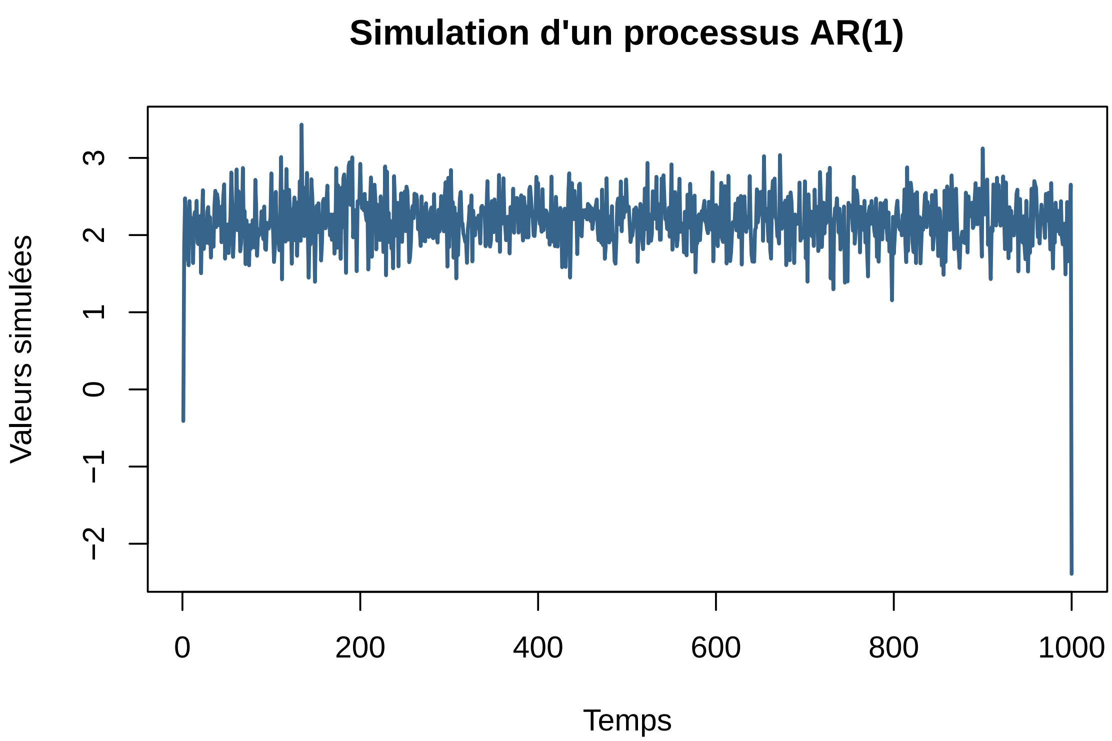
2.3.1 Fonction de simulation d'un processus Ar(1):

```
ar1 = function(alpha, sigma = 1, Time= 1000) {
  x[1] = rnorm(1, sd = sqrt(sigma ^ 2 / (1 - alpha ^ 2)))
  for (t in 2:Time) {
    x[t] = alpha * x[Time- 1] + rnorm(1, sd = sigma)
  }
  return(x)
}
```

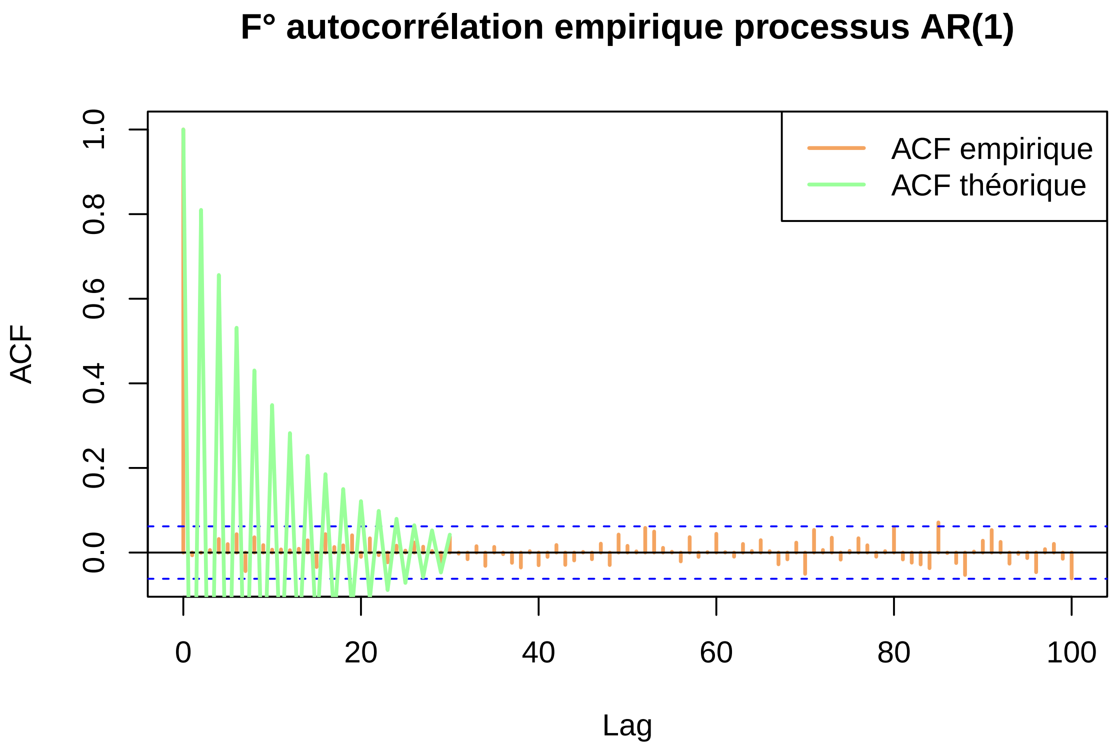
2.3.2 Simulation du processus :

```
alpha = -0.9
# alpha = 0.9
sigma = sqrt(0.1)
Time = 1000
x = ar1(alpha, sigma, Time)
plot(x,
```

```
     main = "Simulation d'un processus AR(1)",
     ylab = "Valeurs simulées",
     xlab = "Temps",
     type = "l",
     col = palette_couleur[1],
     lwd = 2,
     xlim = c(1, Time))
```




```
acf(x,
    main = "F° autocorrélation empirique processus AR(1)",
    col = palette_couleur[2],
    lwd = 2,
    lag.max = 100)
tab = 0:30
rho = alpha ^ tab
lines(tab, rho, col = palette_couleur[3], lwd = 2)
legend(
  'topright',
  c('ACF empirique', 'ACF théorique'),
  col = palette_couleur[2:3],
  lty = 1,
  lwd = 2
)
```



2.4 Exercice 5 :