Séries temporelles univariées

M1 ECAP – TD4 – Année 2024/2025

TD4 : Analyse de la non-stationarité et racines unitaires

Responsable d'enseignement : Benoît SÉVI
benoit.sevi@univ-nantes.fr

HOUSSAIS Rémi, QUINTIN DE KERCADIO Pierre

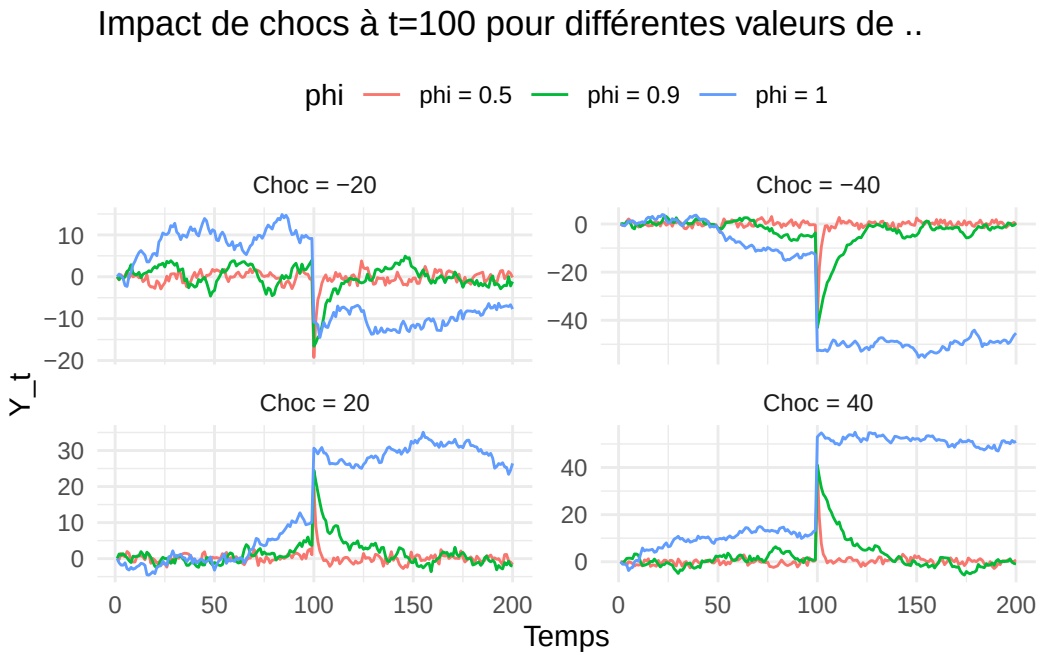
March 31, 2025

Table of contents

1 Chocs permanents vs. choc transitoires: une analyse exploratoire

```r
library(stats)
library(ggplot2)


simulate_ar1 <- function(phi, choc_value, choc_time = 100, n = 200) {
  Y <- numeric(n)
  eps <- rnorm(n, mean = 0, sd = 1)

  for (t in 2:n) {
    Y[t] <- phi * Y[t - 1] + eps[t]
    if (t == choc_time) {
      Y[t] <- Y[t] + choc_value
    }
  }
  return(Y)
}

set.seed(123)


phi_vals <- c(0.5, 0.9, 1.0)
chocs <- c(20, 40, -20, -40)

# Stocker les résultats dans une data.frame
results <- data.frame()

for (choc in chocs) {
  for (phi in phi_vals) {
    serie <- simulate_ar1(phi, choc)
    df <- data.frame(
      t = 1:200,
      Y = serie,
      phi = paste0("phi = ", phi),
      choc = paste0("Choc = ", choc)
    )
    results <- rbind(results, df)
  }
}
ggplot(results, aes(x = t, y = Y, color = phi)) +
  geom_line() +
  facet_wrap(~choc, scales = "free_y") +
  labs(title = "Impact de chocs à t=100 pour différentes valeurs de  ",
       x = "Temps", y = "Y_t") +
```

```
  theme_minimal() +
  theme(legend.position = "top")
```




```
library(ggplot2)
simulate_ar1 <- function(phi, choc_value, choc_time = 100, n = 200) {
  Y <- numeric(n)
  eps <- rnorm(n, mean = 0, sd = 1)

  for (t in 2:n) {
    Y[t] <- phi * Y[t - 1] + eps[t]
    if (t == choc_time) {
      Y[t] <- Y[t] + choc_value
    }
  }
  return(Y)
}
```

Interprétation

Lorsque ϕ=0.5 et ϕ=0.5 :

Un choc de +20 ou -20 entraîne une perturbation temporaire, mais la série retrouve rapidement son niveau d'avant-choc, en environ 10 unités de temps. Cela s'explique par le fait qu'un faible signifie une mémoire courte : les valeurs passées influencent peu les valeurs futures, et l'effet du choc s'estompe rapidement.

Lorsque ϕ=0.9 ϕ=0.9 :

Pour un même choc, le retour à l'état initial est bien plus lent, prenant environ 25 unités de temps. Avec une valeur plus élevée de , la série possède une mémoire plus longue, ce qui signifie que les effets du choc persistent plus longtemps avant de s'atténuer.

Lorsque ϕ=1.0 ϕ=1.0 (marche aléatoire) :

La série ne revient pas à une moyenne stable après le choc. Une fois l'impact survenu, la trajectoire du processus est modifiée de façon durable. Cela illustre bien le fait qu'un AR(1) avec =1 =1 est non stationnaire : il ne possède pas de force de rappel vers une moyenne fixe, et l'effet du choc se propage indéfiniment avec des fluctuations autour du nouveau niveau atteint.

Enfin, lorsque l'intensité du choc passe de 20 à 40 (ou de -20 à -40), les écarts par rapport à l'état initial sont plus marqués, et le temps nécessaire pour s'en rapprocher à nouveau augmente. Cela confirme que plus le choc est fort, plus le processus aura du mal à retrouver son équilibre, surtout pour des valeurs élevées de .

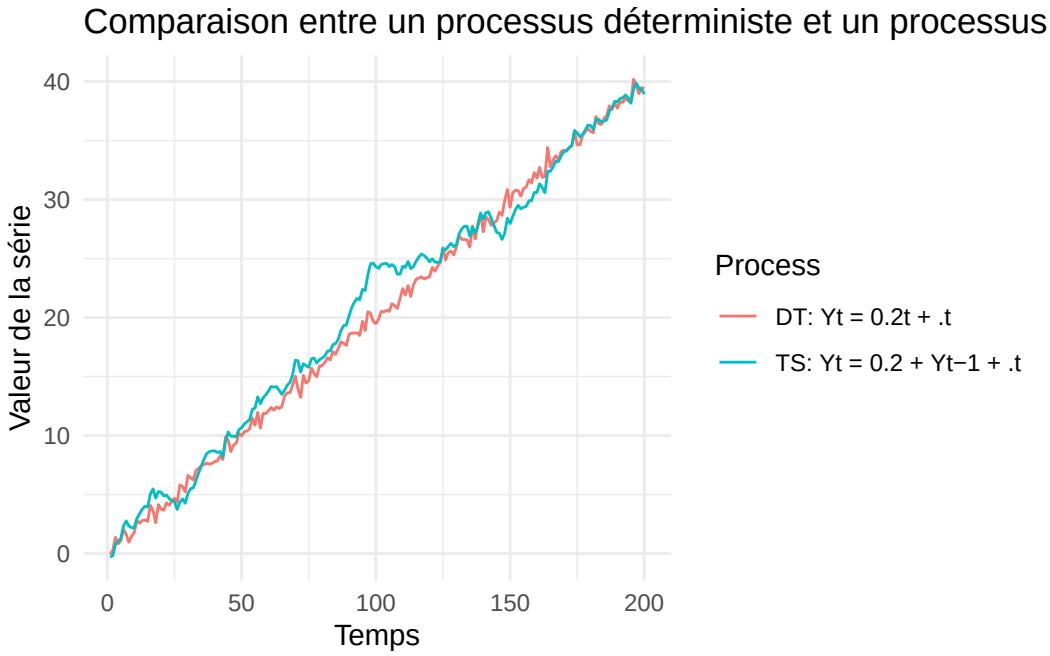
2 TS vs DS : simulation de processus

2.1 Loi $N(0,1/4)$

2.1.1 Seed(123)

```
library(ggplot2)
set.seed(123)

n <- 200
sigma2 <- 1/4

epsilon <- rnorm(n, mean = 0, sd = sqrt(sigma2))  # Génération du bruit blanc


Y_DT <- numeric(n)
Y_TS <- numeric(n)

for (t in 1:n) {
  Y_DT[t] <- 0.2 * t + epsilon[t]
  if (t == 1) {
    Y_TS[t] <- epsilon[t]
  } else {
    Y_TS[t] <- 0.2 + Y_TS[t - 1] + epsilon[t]
  }
}

# Création du DataFrame
data <- data.frame(
  Time = rep(1:n, 2),
  Value = c(Y_DT, Y_TS),
  Process = rep(c("DT: Yt = 0.2t +  t", "TS: Yt = 0.2 + Yt-1 +  t"), each = n)
)
```

```r
# Graphique
ggplot(data, aes(x = Time, y = Value, color = Process)) +
  geom_line() +
  labs(title = "Comparaison entre un processus déterministe et un processus stochasti
       x = "Temps",
       y = "Valeur de la série") +
  theme_minimal()
```



Interprétation

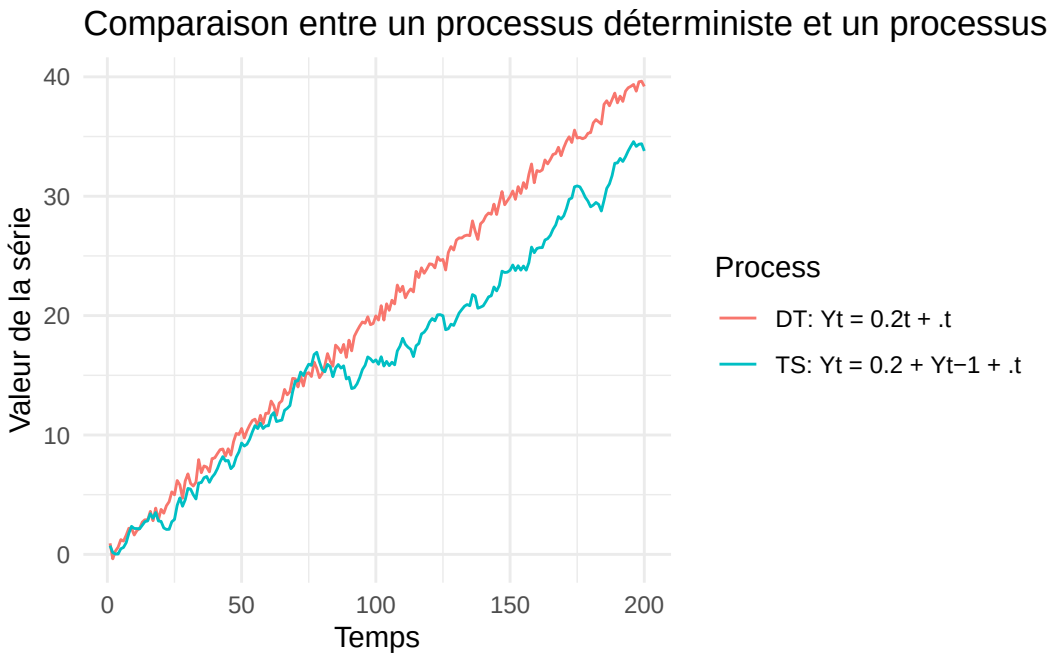
Le graphique compare deux séries soumises à un même bruit N(0,1/4): un processus déterministe ($Y_t = 0.2t + \varepsilon_t$), et un processus stochastique ($Y_t = 0.2 + Y_{t-1} + \varepsilon_t$).

La série déterministe suit une tendance linéaire régulière, perturbée temporairement par de faibles chocs : elle est stationnaire autour d'une tendance. En revanche, la série stochastique accumule les chocs dans le temps, ce qui entraîne des écarts de plus en plus marqués : elle est non stationnaire avec des effets de chocs permanents.

2.1.2 Seed(287)

```r
set.seed(287)

n <- 200
sigma2 <- 1/4

epsilon <- rnorm(n, mean = 0, sd = sqrt(sigma2))  # Génération du bruit blanc


Y_DT <- numeric(n)
```

```r
Y_TS <- numeric(n)

for (t in 1:n) {
  Y_DT[t] <- 0.2 * t + epsilon[t]
  if (t == 1) {
    Y_TS[t] <- epsilon[t]
  } else {
    Y_TS[t] <- 0.2 + Y_TS[t - 1] + epsilon[t]
  }
}

# Création du DataFrame
data <- data.frame(
  Time = rep(1:n, 2),
  Value = c(Y_DT, Y_TS),
  Process = rep(c("DT: Yt = 0.2t +  t", "TS: Yt = 0.2 + Yt-1 +  t"), each = n)
)

# Graphique
ggplot(data, aes(x = Time, y = Value, color = Process)) +
  geom_line() +
  labs(title = "Comparaison entre un processus déterministe et un processus stochasti
       x = "Temps",
       y = "Valeur de la série") +
  theme_minimal()
```



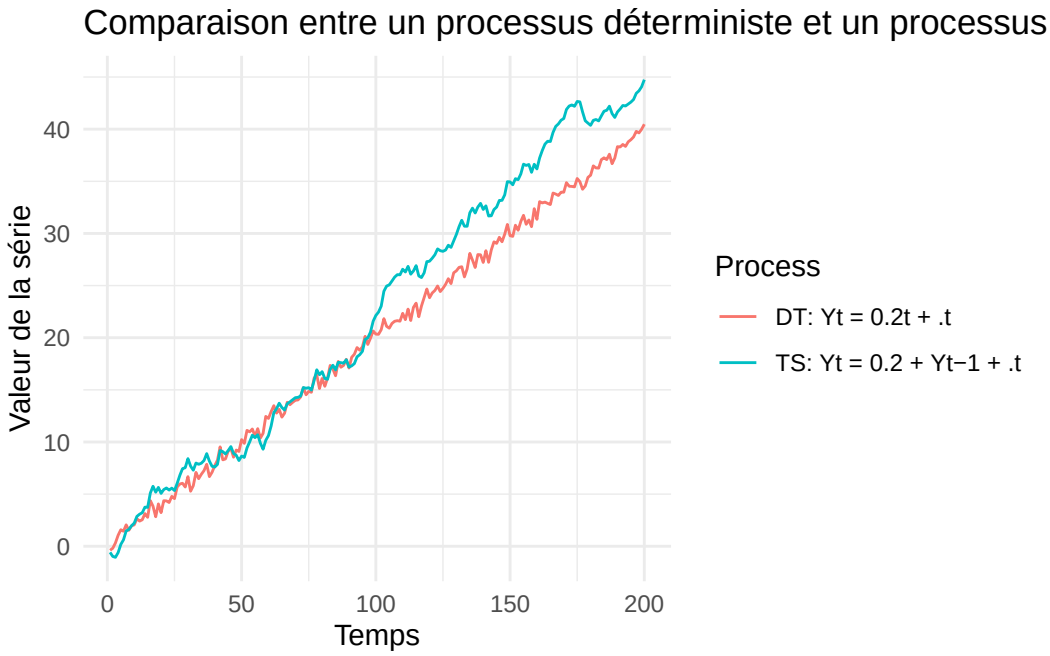
Interprétation

Avec ce nouveau tirage aléatoire, on retrouve la même dynamique générale : le processus déterministe suit une tendance régulière, alors que le processus stochastique évolue de

manière plus irrégulière. Cette fois, le DT reste au-dessus du TS sur presque toute la période, ce qui montre que la trajectoire du processus stochastique dépend fortement des chocs initiaux. Le fond reste inchangé : le DT absorbe les chocs, le TS les cumule dans le temps.

2.1.3 Seed(986)

```r
set.seed(986)

n <- 200
sigma2 <- 1/4

epsilon <- rnorm(n, mean = 0, sd = sqrt(sigma2))  # Génération du bruit blanc


Y_DT <- numeric(n)
Y_TS <- numeric(n)

for (t in 1:n) {
  Y_DT[t] <- 0.2 * t + epsilon[t]
  if (t == 1) {
    Y_TS[t] <- epsilon[t]
  } else {
    Y_TS[t] <- 0.2 + Y_TS[t - 1] + epsilon[t]
  }
}

# Création du DataFrame
data <- data.frame(
  Time = rep(1:n, 2),
  Value = c(Y_DT, Y_TS),
  Process = rep(c("DT: Yt = 0.2t +  t", "TS: Yt = 0.2 + Yt-1 +  t"), each = n)
)

# Graphique
ggplot(data, aes(x = Time, y = Value, color = Process)) +
  geom_line() +
  labs(title = "Comparaison entre un processus déterministe et un processus stochasti
       x = "Temps",
       y = "Valeur de la série") +
  theme_minimal()
```



Interprétation

Avec ce nouveau tirage aléatoire, on retrouve la même dynamique générale: le processus déterministe (rouge) suit une tendance régulière, alors que le processus stochastique (bleu) évolue de manière plus irrégulière. Cette fois, le TS dépasse nettement le DT à partir du milieu de la période, ce qui montre une nouvelle fois que la trajectoire du processus stochastique dépend fortement des chocs accumulés. Le fond reste inchangé : le DT absorbe les chocs, le TS les cumule dans le temps.

2.1.4 Conclusion

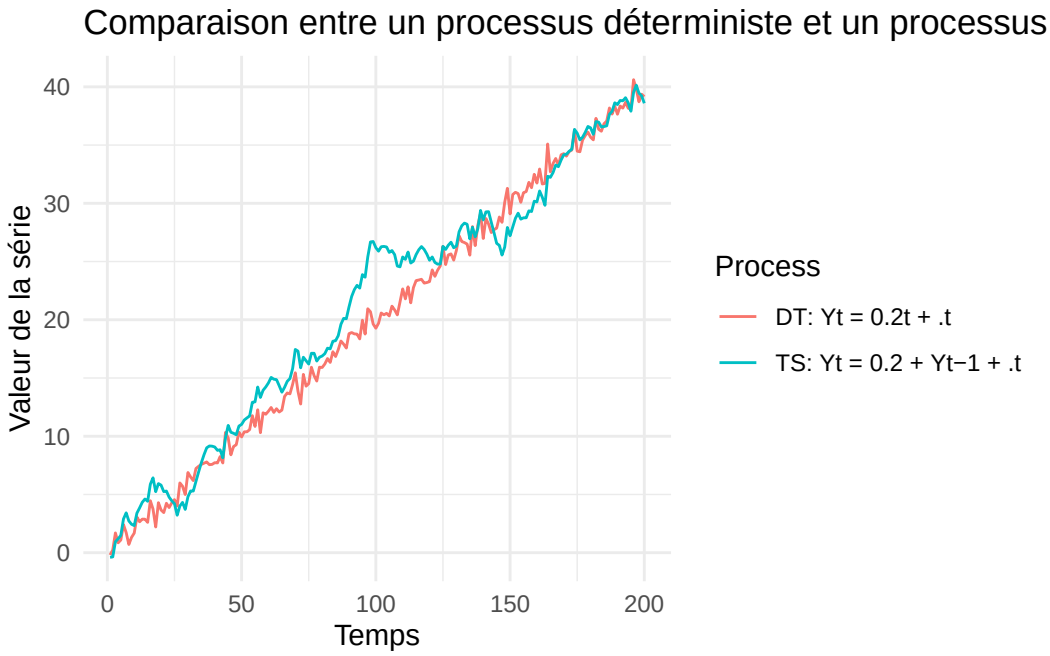
La dynamique globale reste constante : le processus déterministe suit toujours une tendance régulière perturbée temporairement par le bruit, tandis que le processus stochastique affiche une trajectoire plus irrégulière, qui s'écarte progressivement en fonction des chocs accumulés. Dans certaines simulations, le TS reste en dessous du DT, dans d'autres il le dépasse, ce qui illustre parfaitement la forte sensibilité du processus stochastique à la réalisation des chocs initiaux. Ces trois graphiques confirment que le DT est stable et prévisible, alors que le TS est instable et imprévisible à long terme, malgré des conditions de départ identiques

2.2 Loi $N(0,1/2)$

2.2.1 Seed(123)

```
set.seed(123)

# Paramètres
n <- 200  # Nombre d'observations
```

```r
sigma2 <- 1/2  # Variance du bruit

epsilon <- rnorm(n, mean = 0, sd = sqrt(sigma2))  # Génération du bruit blanc

# Simulation des processus
Y_DT <- numeric(n)
Y_TS <- numeric(n)

for (t in 1:n) {
  Y_DT[t] <- 0.2 * t + epsilon[t]
  if (t == 1) {
    Y_TS[t] <- epsilon[t]
  } else {
    Y_TS[t] <- 0.2 + Y_TS[t - 1] + epsilon[t]
  }
}

# Création du DataFrame
data <- data.frame(
  Time = rep(1:n, 2),
  Value = c(Y_DT, Y_TS),
  Process = rep(c("DT: Yt = 0.2t +  t", "TS: Yt = 0.2 + Yt-1 +  t"), each = n)
)

# Graphique
ggplot(data, aes(x = Time, y = Value, color = Process)) +
  geom_line() +
  labs(title = "Comparaison entre un processus déterministe et un processus stochasti
       x = "Temps",
       y = "Valeur de la série") +
  theme_minimal()
```



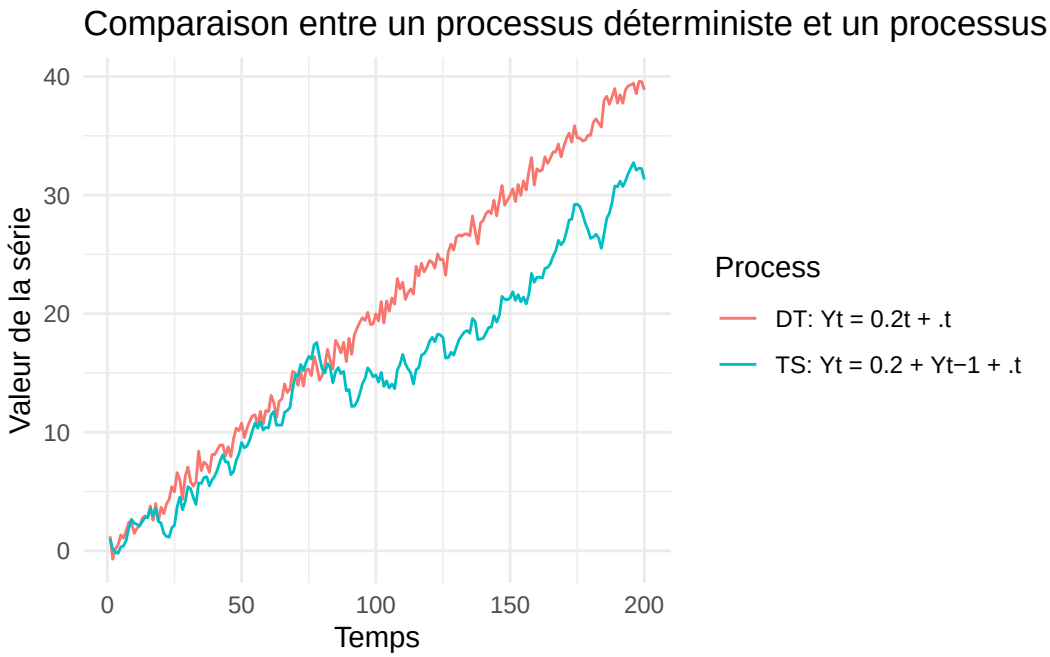
Interprétation

Avec ce nouveau tirage aléatoire et une variance du bruit plus élevée, on observe une dynamique toujours similaire : le processus déterministe progresse régulièrement selon sa tendance linéaire, tandis que le processus stochastique présente une trajectoire plus irrégulière. Ici, les deux séries restent proches sur l'ensemble de l'échantillon, mais on note que le TS dépasse légèrement le DT en fin de période. Cela montre que l'augmentation de la variance amplifie la variabilité des deux séries, mais affecte davantage le TS qui accumule les chocs. Le DT, lui, reste globalement stable autour de sa pente.

2.2.2 Seed(287)

```r
set.seed(287)

# Paramètres
n <- 200  # Nombre d'observations
sigma2 <- 1/2  # Variance du bruit

epsilon <- rnorm(n, mean = 0, sd = sqrt(sigma2))  # Génération du bruit blanc

# Simulation des processus
Y_DT <- numeric(n)
Y_TS <- numeric(n)

for (t in 1:n) {
  Y_DT[t] <- 0.2 * t + epsilon[t]
  if (t == 1) {
    Y_TS[t] <- epsilon[t]
  } else {
```

```r
    Y_TS[t] <- 0.2 + Y_TS[t - 1] + epsilon[t]
  }
}

# Création du DataFrame
data <- data.frame(
  Time = rep(1:n, 2),
  Value = c(Y_DT, Y_TS),
  Process = rep(c("DT: Yt = 0.2t +  t", "TS: Yt = 0.2 + Yt-1 +  t"), each = n)
)

# Graphique
ggplot(data, aes(x = Time, y = Value, color = Process)) +
  geom_line() +
  labs(title = "Comparaison entre un processus déterministe et un processus stochasti
       x = "Temps",
       y = "Valeur de la série") +
  theme_minimal()
```

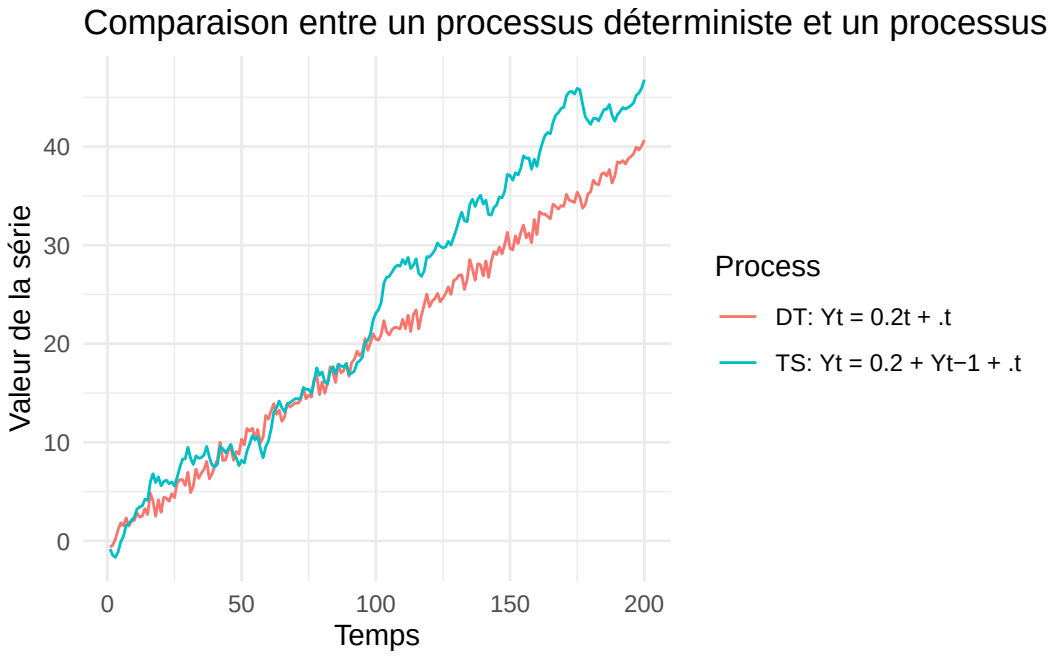


Interprétation

Avec ce nouveau tirage, on observe une configuration différente : le processus déterministe dépasse clairement le processus stochastique sur presque toute la période. Cela illustre à nouveau que, bien que les deux séries partagent la même pente moyenne (0.2), le comportement du TS est fortement influencé par les chocs initiaux. Ici, ces chocs ont freiné la progression de la série stochastique, tandis que la série déterministe, insensible à la mémoire, poursuit une trajectoire plus régulière. Le contraste est net en fin de période, où le DT atteint des niveaux beaucoup plus élevés que le TS.

2.2.3 Seed(986)

```r
set.seed(986)

# Paramètres
n <- 200  # Nombre d'observations
sigma2 <- 1/2  # Variance du bruit

epsilon <- rnorm(n, mean = 0, sd = sqrt(sigma2))  # Génération du bruit blanc

# Simulation des processus
Y_DT <- numeric(n)
Y_TS <- numeric(n)

for (t in 1:n) {
  Y_DT[t] <- 0.2 * t + epsilon[t]
  if (t == 1) {
    Y_TS[t] <- epsilon[t]
  } else {
    Y_TS[t] <- 0.2 + Y_TS[t - 1] + epsilon[t]
  }
}

# Création du DataFrame
data <- data.frame(
  Time = rep(1:n, 2),
  Value = c(Y_DT, Y_TS),
  Process = rep(c("DT: Yt = 0.2t +  t", "TS: Yt = 0.2 + Yt-1 +  t"), each = n)
)

# Graphique
ggplot(data, aes(x = Time, y = Value, color = Process)) +
  geom_line() +
  labs(title = "Comparaison entre un processus déterministe et un processus stochasti
       x = "Temps",
       y = "Valeur de la série") +
  theme_minimal()
```



Interprétation

Avec ce troisième tirage, le processus stochastique est nettement au dessus sur le déterministe dès la moitié de la période. On observe ici une forte montée du TS, liée à une succession de chocs positifs qui, une fois cumulés, entraînent un écart important par rapport au DT. À l'inverse, le processus déterministe reste fidèle à sa tendance linéaire, avec des fluctuations modérées. Cette configuration montre à quel point le processus stochastique peut diverger rapidement selon les chocs, même si les deux séries partagent la même pente moyenne.

2.2.4 Conclusion

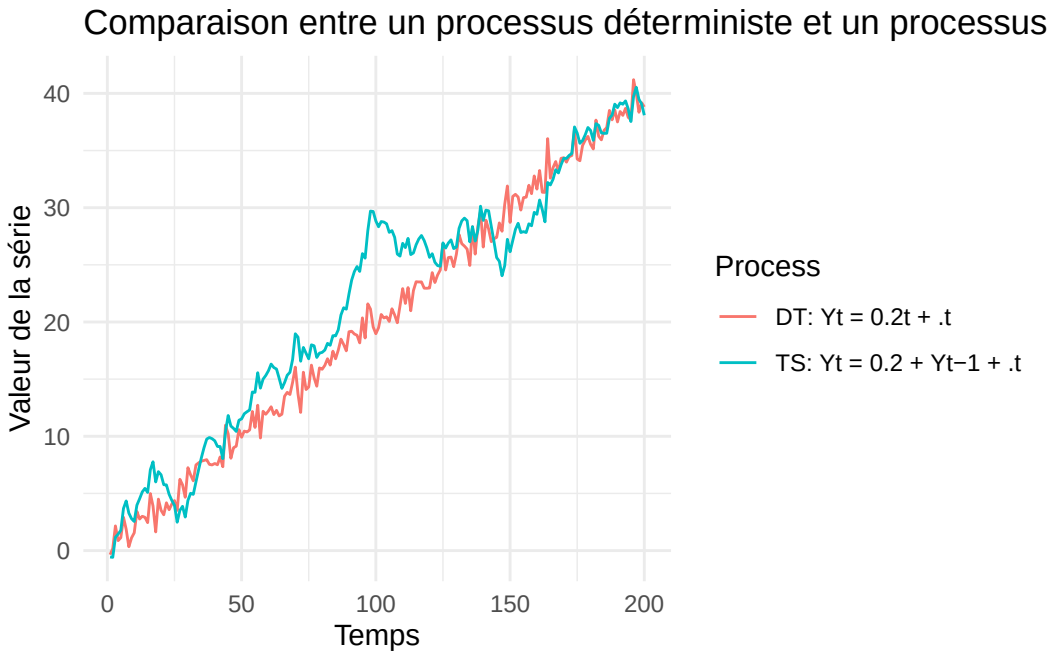
Pour une loi $N(0,1/2)$ nous pouvons remarquer que la dynamique reste constante comme sur la loi $N(0,1/4)$. En effet, le processus déterministe suit toujours une tendance régulière, seulement perturbée de manière temporaire par un bruit modéré. En revanche, le processus stochastique présente une trajectoire plus irrégulière, qui s'écarte progressivement à mesure que les chocs s'accumulent dans le temps. Dans certaines simulations, le TS reste en dessous du DT, dans d'autres il le dépasse nettement, ce qui illustre parfaitement la forte sensibilité du processus stochastique à la réalisation des chocs aléatoires. Ces trois graphiques confirment que le DT est stable, tandis que le TS reste instable.

2.3 Loi $N(0,1)$

2.3.1 Seed(123)

```
set.seed(123)

# Paramètres
```

```r
n <- 200  # Nombre d'observations
sigma2 <- 1  # Variance du bruit

epsilon <- rnorm(n, mean = 0, sd = sqrt(sigma2))  # Génération du bruit blanc

# Simulation des processus
Y_DT <- numeric(n)
Y_TS <- numeric(n)

for (t in 1:n) {
  Y_DT[t] <- 0.2 * t + epsilon[t]
  if (t == 1) {
    Y_TS[t] <- epsilon[t]
  } else {
    Y_TS[t] <- 0.2 + Y_TS[t - 1] + epsilon[t]
  }
}

# Création du DataFrame
data <- data.frame(
  Time = rep(1:n, 2),
  Value = c(Y_DT, Y_TS),
  Process = rep(c("DT: Yt = 0.2t +  t", "TS: Yt = 0.2 + Yt-1 +  t"), each = n)
)

# Graphique
ggplot(data, aes(x = Time, y = Value, color = Process)) +
  geom_line() +
  labs(title = "Comparaison entre un processus déterministe et un processus stochasti
       x = "Temps",
       y = "Valeur de la série") +
  theme_minimal()
```



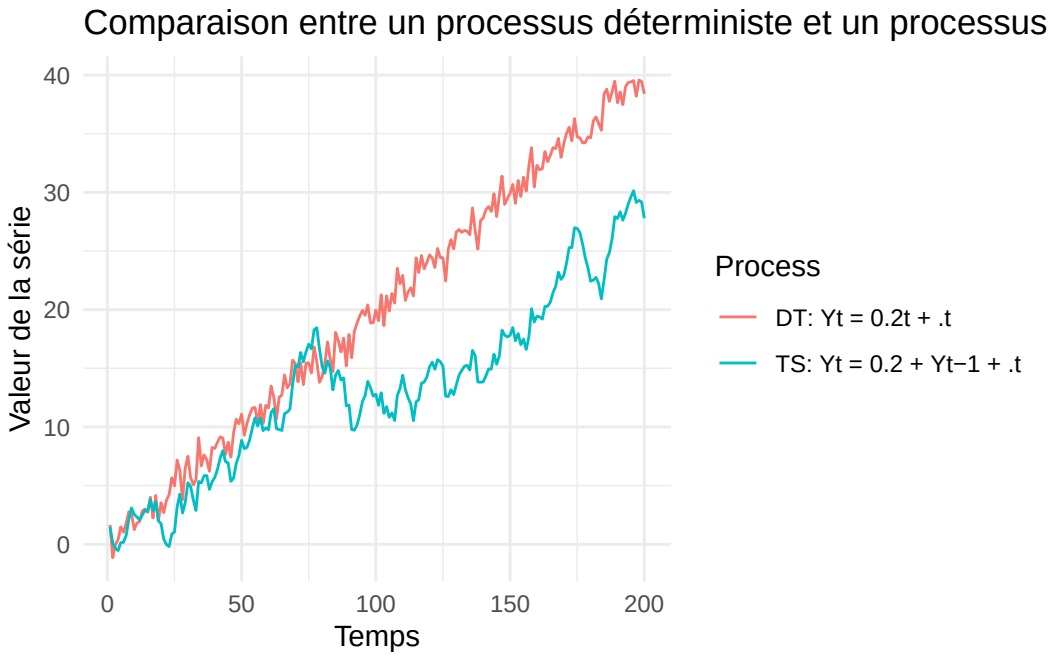
Interprétation

Pour cette loi $N(0,1)$, la dynamique reste la même observée que les deux autres lois. En effet,le processus déterministe suit une trajectoire croissante régulière, alors que le processus stochastique fluctue davantage et s'en écarte par moments. Ici, le TS dépasse nettement le DT autour de t=100. avant de redescendre et de se rapprocher en fin de période.

2.3.2 Seed(287)

```r
set.seed(287)

# Paramètres
n <- 200  # Nombre d'observations
sigma2 <- 1  # Variance du bruit

epsilon <- rnorm(n, mean = 0, sd = sqrt(sigma2))  # Génération du bruit blanc

# Simulation des processus
Y_DT <- numeric(n)
Y_TS <- numeric(n)

for (t in 1:n) {
  Y_DT[t] <- 0.2 * t + epsilon[t]
  if (t == 1) {
    Y_TS[t] <- epsilon[t]
  } else {
    Y_TS[t] <- 0.2 + Y_TS[t - 1] + epsilon[t]
  }
```

```r
}

# Création du DataFrame
data <- data.frame(
  Time = rep(1:n, 2),
  Value = c(Y_DT, Y_TS),
  Process = rep(c("DT: Yt = 0.2t +  t", "TS: Yt = 0.2 + Yt-1 +  t"), each = n)
)

# Graphique
ggplot(data, aes(x = Time, y = Value, color = Process)) +
  geom_line() +
  labs(title = "Comparaison entre un processus déterministe et un processus stochasti
       x = "Temps",
       y = "Valeur de la série") +
  theme_minimal()
```



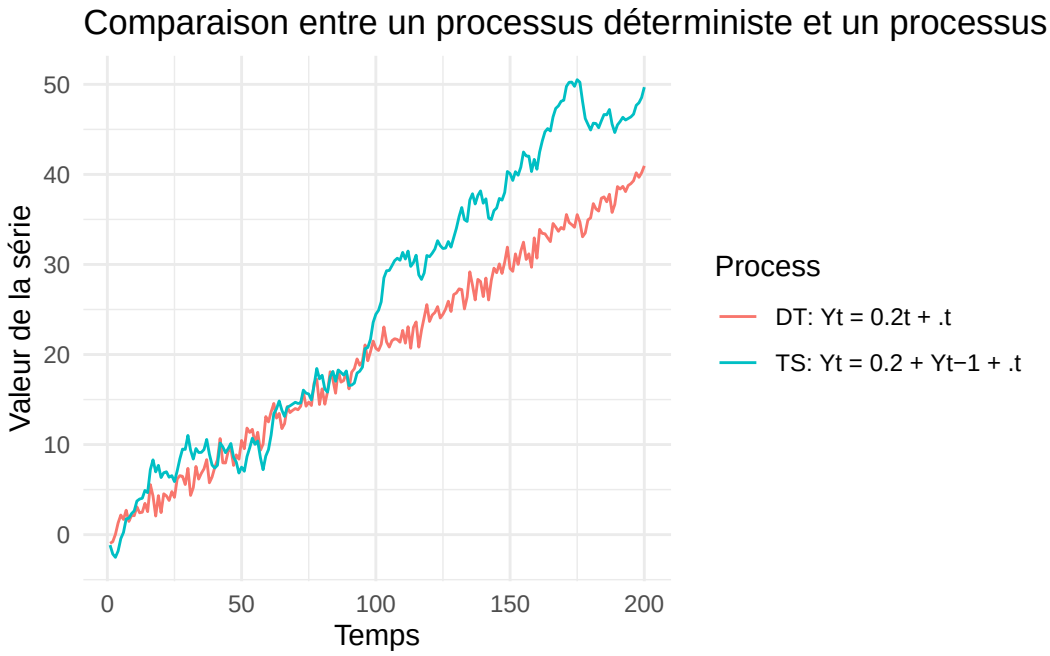
Interprétation

Avec ce nouveau tirage aléatoire, ,nous pouvons remarquer le processus déterministe est toujours stable dans le temps et est au dessus du stochastique sur l'ensemble de la période. Nous pouvons remarquer, que à partiur de T=90, TS chute brutalement et reste en dessous du processus déterministe.

2.3.3 Seed(986)

```r
set.seed(986)

# Paramètres
```

```r
n <- 200  # Nombre d'observations
sigma2 <- 1  # Variance du bruit

epsilon <- rnorm(n, mean = 0, sd = sqrt(sigma2))  # Génération du bruit blanc

# Simulation des processus
Y_DT <- numeric(n)
Y_TS <- numeric(n)

for (t in 1:n) {
  Y_DT[t] <- 0.2 * t + epsilon[t]
  if (t == 1) {
    Y_TS[t] <- epsilon[t]
  } else {
    Y_TS[t] <- 0.2 + Y_TS[t - 1] + epsilon[t]
  }
}

# Création du DataFrame
data <- data.frame(
  Time = rep(1:n, 2),
  Value = c(Y_DT, Y_TS),
  Process = rep(c("DT: Yt = 0.2t +  t", "TS: Yt = 0.2 + Yt-1 +  t"), each = n)
)

# Graphique
ggplot(data, aes(x = Time, y = Value, color = Process)) +
  geom_line() +
  labs(title = "Comparaison entre un processus déterministe et un processus stochasti
       x = "Temps",
       y = "Valeur de la série") +
  theme_minimal()
```



Interprétation

Avec ce nouveau tirage aléatoire, ,nous pouvons remarquer le processus déterministe est toujours stable dans le temps et est en dessous du stochastique sur l'ensemble de la période. Nous pouvons remarquer, que à partiur de T=90, TS monte brutalement et reste au dessus du processus déterministe.

2.3.4 Conclusion

Pour une loi $N(0,1)$, la dynamique globale reste constante : le processus déterministe suit toujours une tendance régulière perturbée temporairement par le bruit, tandis que le processus stochastique affiche une trajectoire plus irrégulière, qui s'écarte progressivement en fonction des chocs accumulés. Avec une variance plus élevée, ces écarts deviennent encore plus marqués. Le TS peut rester en retrait comme dans le deuxième graphique, coller au DT comme dans le premier, ou bien le dépasser largement comme dans le troisième. Cette variabilité illustre parfaitement la forte sensibilité du processus stochastique à la réalisation des chocs, qui s'amplifie avec la taille des perturbations.

2.4 Conlusion générale

Pour conclure sur cette partie, nous pouvons remarquer que pour chaque série c'était plus ou moins la m^meme tendance. En effet, le processus déterministe a toujours suivi la tendnace quelque soit la loi N ou le seed, et le processus stochastique a toujours été plus sensible aux fluctuations.

3 Régressions fallacieuses

3.1 $n = 200$ et $N = 5000$

```
set.seed(123)


n <- 200
N <- 5000
rejets <- numeric(N)

for (i in 1:N) {

  X <- cumsum(rnorm(n))
  Y <- cumsum(rnorm(n))


  model <- lm(Y ~ X)
  p_value <- summary(model)$coefficients[2, 4]


  rejets[i] <- ifelse(p_value < 0.05, 1, 0)
}

pourcentage_rejets <- mean(rejets) * 100
cat("Pourcentage de rejets de H0 ( 1 = 0) au seuil de 5% :", round(pourcentage_rejets,
```

```
Pourcentage de rejets de H0 ( 1 = 0) au seuil de 5% : 83.52 %
```

Interprétation Avec ce premeir test avec n=200 et N=500, nous obtenons un taux de rejet de HO de 83.52%, par conséquent même si les séries sont indépendantes, nous rejetons HO très souvent. Cela s'explique par le fait que les marches aléatoires sont non stationnaires, en raison de la présence d'une racine unitaire dans leur dynamique.

Pour pouvoir observer les differnces nousn allons varier n et N

3.2 Variations du nombre d'observations,n, et du nombre de séries générée, N.

3.2.1 Augmentation et diminution de n

3.2.1.1 $n = 50$ et $N = 5000$

```
set.seed(238)
```

```r
n <- 50
N <- 5000
rejets <- numeric(N)

for (i in 1:N) {

  X <- cumsum(rnorm(n))
  Y <- cumsum(rnorm(n))


  model <- lm(Y ~ X)
  p_value <- summary(model)$coefficients[2, 4]


  rejets[i] <- ifelse(p_value < 0.05, 1, 0)
}

pourcentage_rejets <- mean(rejets) * 100
cat("Pourcentage de rejets de H0 ( 1 = 0) au seuil de 5% :", round(pourcentage_rejets,
```

```
Pourcentage de rejets de H0 ( 1 = 0) au seuil de 5% : 66.14 %
```

Interprétation

Lorsqu'on diminue le nombre d'observation n à 50, nous pouvons remarquer que nous obtenons un taux de rejet de HO de 66.14%. Ce qui est inférieur à celui précédent. Par conséquent, quand le nombre d'observation diminue alors on rejete moins à tort.

3.2.1.2 $n = 500$ et $N = 5000$

```r
set.seed(238)


n <- 500
N <- 5000
rejets <- numeric(N)

for (i in 1:N) {

  X <- cumsum(rnorm(n))
  Y <- cumsum(rnorm(n))


  model <- lm(Y ~ X)
  p_value <- summary(model)$coefficients[2, 4]
```

```r
  rejets[i] <- ifelse(p_value < 0.05, 1, 0)
}

pourcentage_rejets <- mean(rejets) * 100
cat("Pourcentage de rejets de H0 ( 1 = 0) au seuil de 5% :", round(pourcentage_rejets,
```

```
Pourcentage de rejets de H0 ( 1 = 0) au seuil de 5% : 89.2 %
```

Interprétation

Lorsqu'on augmente le nombre d'observation n à 50°, nous pouvons remarquer que nous obtenons un taux de rejet de HO de 89.2%. Ce qui est supérieur aux deux précédents. Par conséquent, quand le nombre d'observation augmente alors on rejete plus à tort.

3.2.2 Augmentation et diminution de N

3.2.2.1 $n = 200$ et $N = 2000$

```r
set.seed(238)


n <- 200
N <- 2000
rejets <- numeric(N)

for (i in 1:N) {

  X <- cumsum(rnorm(n))
  Y <- cumsum(rnorm(n))


  model <- lm(Y ~ X)
  p_value <- summary(model)$coefficients[2, 4]


  rejets[i] <- ifelse(p_value < 0.05, 1, 0)
}

pourcentage_rejets <- mean(rejets) * 100
cat("Pourcentage de rejets de H0 ( 1 = 0) au seuil de 5% :", round(pourcentage_rejets,
```

```
Pourcentage de rejets de H0 ( 1 = 0) au seuil de 5% : 83.6 %
```

Interprétation

Lorsqu'on diminue le nombre de séries générées N à 2000, nous pouvons remarquer que nous obtenons un taux de rejet de HO de 83.6%. Ce qui est légérement au dessus du modèle initial.

3.2.2.2 $n = 200$ et $N = 10000$

```r
set.seed(238)


n <- 200
N <- 10000
rejets <- numeric(N)

for (i in 1:N) {

  X <- cumsum(rnorm(n))
  Y <- cumsum(rnorm(n))


  model <- lm(Y ~ X)
  p_value <- summary(model)$coefficients[2, 4]


  rejets[i] <- ifelse(p_value < 0.05, 1, 0)
}

pourcentage_rejets <- mean(rejets) * 100
cat("Pourcentage de rejets de H0 ( 1 = 0) au seuil de 5% :", round(pourcentage_rejets,
```

```
Pourcentage de rejets de H0 ( 1 = 0) au seuil de 5% : 83.49 %
```

Interprétation

Lorsqu'on augmente le nombre de séries générées N à 10000, nous pouvons remarquer que nous obtenons un taux de rejet de HO de 83.49%. Ce qui est légérement en dessous du modèle initial.

3.3 Conclusion générale

Par conséquent, le fait de changer le nombre de séries générées ne varie pas énormément le taux de rejet de H0. Alors que, le fait de changer à lla baisse ou à la hausse le nombre d'observations fait varier énormément le taux de rejet de H0

4 Distribution de la statistique de test de Dickey-Fuller pour le modèle sans constante ni tendance via la méthode de Monte Carlo

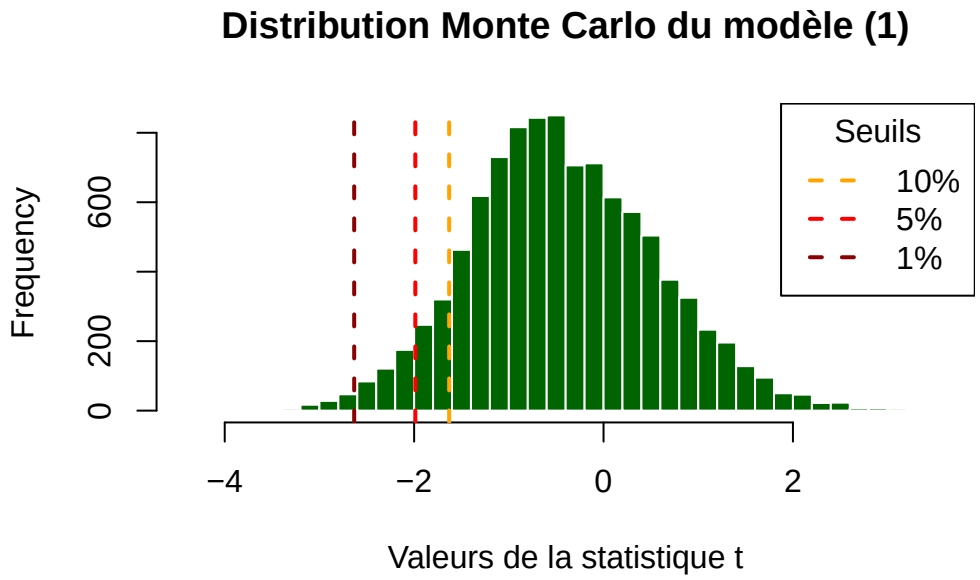
4.1 Distribution du modèle (1) sans constante ni tendance $Y_t = Y_{t-1} + \varepsilon_t$

```
set.seed(123)

# Paramètres
n <- 100          # Longueur de la série
N <- 10000        # Nombre de simulations
t_stats <- numeric(N)

for (i in 1:N) {
  eps <- rnorm(n)
  Y <- cumsum(eps)  # Y_t = Y_{t-1} +  _t

  dY <- diff(Y)             # ΔY_t
  Y_lag <- Y[-n]            # Y_{t-1}

  model <- lm(dY ~ 0 + Y_lag)  # régression sans constante
  t_stats[i] <- summary(model)$coefficients[1, 3]  # t-statistique
}

# Valeurs critiques empiriques
quantiles <- quantile(t_stats, probs = c(0.10, 0.05, 0.01))
print(round(quantiles, 3))
```

```
   10%     5%     1%
-1.630 -1.988 -2.633
```

```
# Histogramme
hist(t_stats, breaks = 50, col = "darkgreen", border = "white",
     main = "Distribution Monte Carlo du modèle (1)",
     xlab = "Valeurs de la statistique t")

# Ajout des lignes verticales pour les quantiles critiques
abline(v = quantiles[1], col = "orange", lwd = 2, lty = 2)
abline(v = quantiles[2], col = "red", lwd = 2, lty = 2)
abline(v = quantiles[3], col = "darkred", lwd = 2, lty = 2)

# Ajout d'une légende
legend("topright", legend = c("10%", "5%", "1%"),
```

```
       col = c("orange", "red", "darkred"),
       lty = 2, lwd = 2, title = "Seuils")
```



Interprétation

Pour ce modèle, sans constante ni tendance nous pouvons observer plusieurs indications. En effet, les valeurs critiques issues de la simulation du modèle (1) sont très proches de celles de la table théorique de Dickey-Fuller :

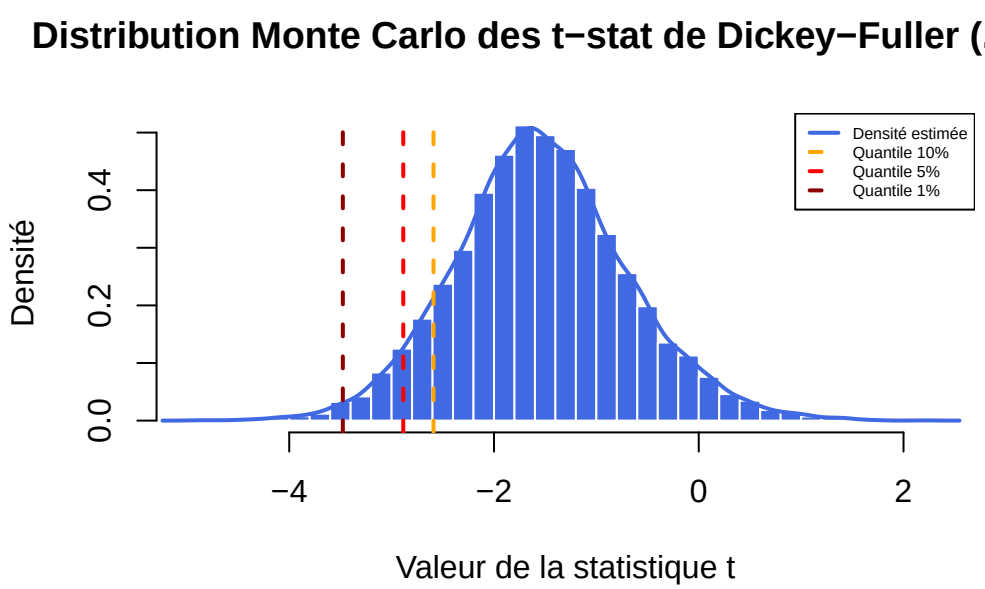
Seuil	Monte Carlo	Table DF (n = 100)
10%	-1.630	-1.61
5%	-1.988	-1.95
1%	-2.633	-2.60

En simulant 10 000 séries de 100 observations d'un processus à racine unitaire sans constante ni tendance, nous obtenons une distribution empirique de la statistique t du test de Dickey-Fuller. Les seuils de rejet à 10 %, 5 % et 1 % obtenus sont respectivement de -1.630, -1.988 et -2.633. Et lorsque l'on compare ces valeurs à celles issues de la table théorique de Dickey-Fuller, on observe une très bonne similitude avec des valeurs respectives de -1.61, -1.95 et -2.60 pour les mêmes seuils de rejet.

4.2 Distribution du modèle (2) avec $\delta_0 = 5$

```
set.seed(123)

n_simulations <- 10000
n_obs <- 100
```

```r
# Fonction pour générer une marche aléatoire avec constante
generate_random_walk_avec_const <- function(n, delta0) {
  Yt <- numeric(n)
  Yt[1] <- delta0  # Initialisation avec la constante
  for (t in 2:n) {
    Yt[t] <- Yt[t-1] + rnorm(1)
  }
  return(Yt)
}

# Fonction pour exécuter la simulation et afficher les résultats
simulation_cst <- function(delta0) {
  set.seed(123)  # Réinitialisation de la graine pour reproductibilité

  t_stats <- numeric(n_simulations)  # Réinitialisation à chaque appel

  for (i in 1:n_simulations) {
    Yt <- generate_random_walk_avec_const(n_obs, delta0)

    dYt <- diff(Yt)
    Yt_lag <- Yt[-n_obs]

    model <- lm(dYt ~ Yt_lag)  # Régression avec constante par défaut
    t_stats[i] <- summary(model)$coefficients[2, 3]  # Récupération du t-stat
  }

  # Calcul des quantiles (valeurs critiques)
  quantiles <- quantile(t_stats, c(0.10, 0.05, 0.01))

  # Affichage graphique
  hist(t_stats, breaks = 50, probability = TRUE, col = "royalblue", border = "white",
       main = paste("Distribution Monte Carlo des t-stat de Dickey-Fuller ( 0 =", delt
       xlab = "Valeur de la statistique t", ylab = "Densité")
  lines(density(t_stats), col = "royalblue", lwd = 2)
  abline(v = quantiles, col = c("orange", "red", "darkred"), lwd = 2, lty = 2)
  legend("topright", legend = c("Densité estimée", "Quantile 10%", "Quantile 5%", "Qu
         col = c("royalblue", "orange", "red", "darkred"), lwd = 2, cex = 0.5, lty =

  # Affichage des valeurs critiques
  cat("Valeurs critiques obtenues pour  0 =", delta0, ":\n")
  print(quantiles)
}


simulation_cst(delta0 = 5)
```




```
Valeurs critiques obtenues pour  0 = 5 :
      10%        5%        1%
-2.590458 -2.885931 -3.476393
```

```
#simulation_cst(delta0 = 10)
```

Interprétation

En introduisant une constante $\delta_0 = 5$ dans le processus simulé, nous observons que la distribution empirique de la statistique t est nettement déplacée vers la gauche par rapport à la distribution théorique attendue pour le test de Dickey-Fuller avec constante.

Seuil	Monte Carlo	Table DF (n = 100)
10%	-2.590	-1.61
5%	-2.885	-1.95
1%	-3.476	-2.60

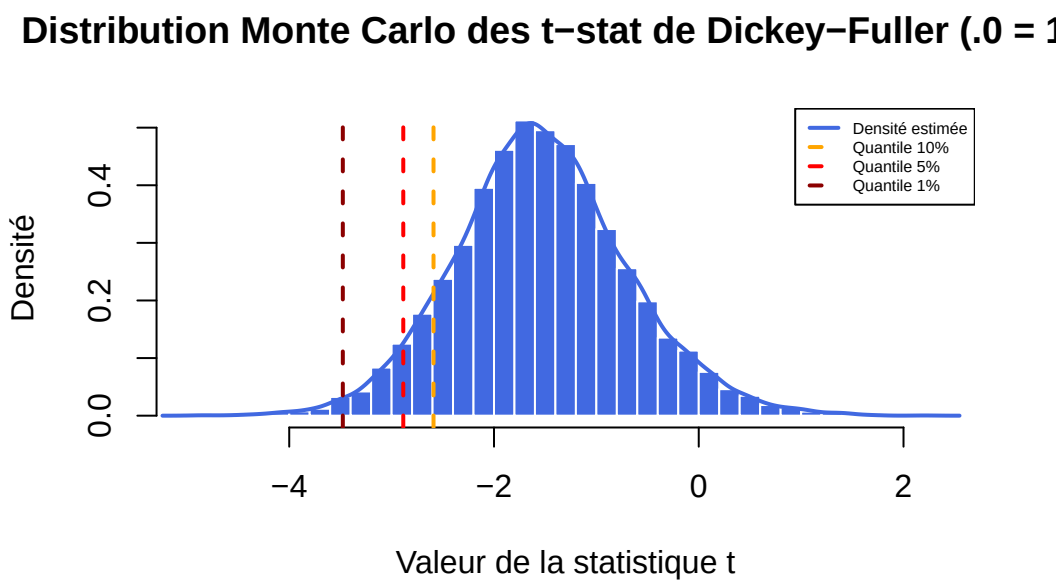
En effet, nous pouvons observer que pour $\delta_0 = 5$ les seuils de rejet s'éloignent de ceux de la table de Dickey-Fuller.

Par conséquent, d'après cette première analyse nous pouvons observer que l'instauration d'un delta est importante dans les simulations. Nous allons le vérifier avec l'instauration de delta à 10.

4.3 Distribution du modèle (3) avec une constante $\delta_0 = 10$

```r
set.seed(123)

n_simulations <- 10000
n_obs <- 100

# Fonction pour générer une marche aléatoire avec constante
generate_random_walk_avec_const <- function(n, delta0) {
  Yt <- numeric(n)
  Yt[1] <- delta0  # Initialisation avec la constante
  for (t in 2:n) {
    Yt[t] <- Yt[t-1] + rnorm(1)
  }
  return(Yt)
}

# Fonction pour exécuter la simulation et afficher les résultats
simulation_cst <- function(delta0) {
  set.seed(123)  # Réinitialisation de la graine pour reproductibilité

  t_stats <- numeric(n_simulations)  # Réinitialisation à chaque appel

  for (i in 1:n_simulations) {
    Yt <- generate_random_walk_avec_const(n_obs, delta0)

    dYt <- diff(Yt)
    Yt_lag <- Yt[-n_obs]

    model <- lm(dYt ~ Yt_lag)  # Régression avec constante par défaut
    t_stats[i] <- summary(model)$coefficients[2, 3]  # Récupération du t-stat
  }

  # Calcul des quantiles (valeurs critiques)
  quantiles <- quantile(t_stats, c(0.10, 0.05, 0.01))

  # Affichage graphique
  hist(t_stats, breaks = 50, probability = TRUE, col = "royalblue", border = "white",
       main = paste("Distribution Monte Carlo des t-stat de Dickey-Fuller ( 0 =", delt
       xlab = "Valeur de la statistique t", ylab = "Densité")
  lines(density(t_stats), col = "royalblue", lwd = 2)
  abline(v = quantiles, col = c("orange", "red", "darkred"), lwd = 2, lty = 2)
  legend("topright", legend = c("Densité estimée", "Quantile 10%", "Quantile 5%", "Qu
         col = c("royalblue", "orange", "red", "darkred"), lwd = 2, cex = 0.5, lty =

  # Affichage des valeurs critiques
  cat("Valeurs critiques obtenues pour  0 =", delta0, ":\n")
  print(quantiles)
}
```

```r
#simulation_cst(delta0 = 5)
simulation_cst(delta0 = 10)
```




```
Valeurs critiques obtenues pour  0 = 10 :
      10%        5%        1%
-2.590458 -2.885931 -3.476393
```

Interprétation

En introduisant une constante $\delta_0 = 10$ dans le processus simulé, nous observons que la distribution empirique de la statistique t est nettement déplacée vers la gauche par rapport à la distribution théorique attendue pour le test de Dickey-Fuller avec constante.

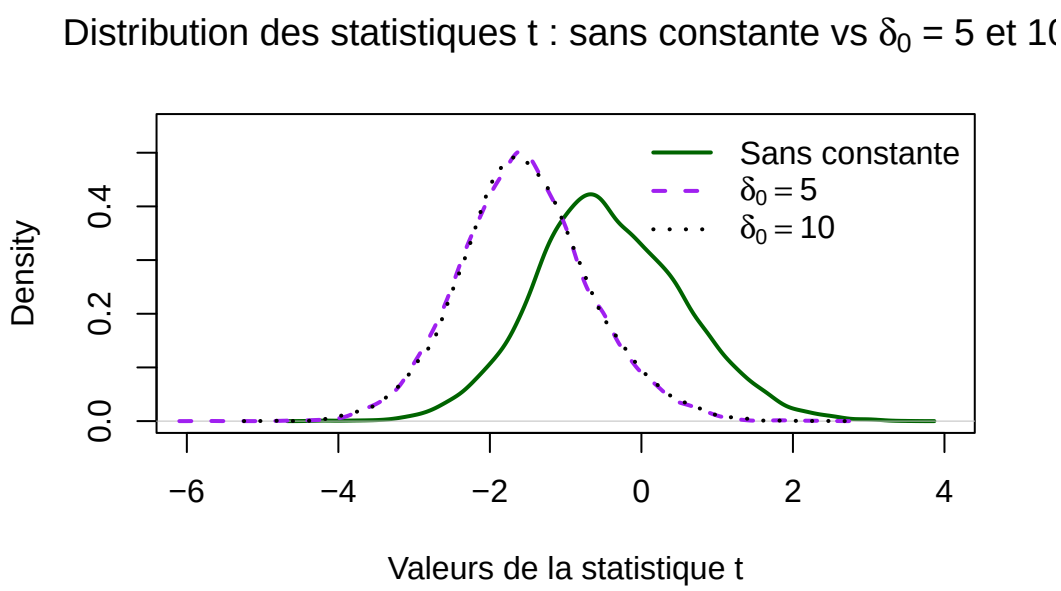
Seuil	Monte Carlo	Table DF (n = 100)
10%	-2.590	-1.61
5%	-2.885	-1.95
1%	-3.476	-2.60

Pour cette secponde simulation avec constante, nous pouvons également remarquer que le fait d'introduire une constante influence grandement grandement les résultats de la simulation.

4.4 Conclusion générale

```r
set.seed(123)

n <- 100
N <- 10000

# Initialisation
t_none <- numeric(N)
t_d5 <- numeric(N)
t_d10 <- numeric(N)

# Simulation : modèle sans constante
for (i in 1:N) {
  eps <- rnorm(n)
  Y <- cumsum(eps)
  dY <- diff(Y)
  Y_lag <- Y[-n]
  model <- lm(dY ~ 0 + Y_lag)
  t_none[i] <- summary(model)$coefficients[1, 3]
}

# Simulation : modèle avec constante   = 5
for (i in 1:N) {
  eps <- rnorm(n)
  Y <- numeric(n)
  Y[1] <- 5
  for (t in 2:n) {
    Y[t] <- Y[t - 1] + eps[t]
  }
  dY <- diff(Y)
  Y_lag <- Y[-n]
  model <- lm(dY ~ Y_lag)
  t_d5[i] <- summary(model)$coefficients[2, 3]
}

# Simulation : modèle avec constante   = 10
for (i in 1:N) {
  eps <- rnorm(n)
  Y <- numeric(n)
  Y[1] <- 10
  for (t in 2:n) {
    Y[t] <- Y[t - 1] + eps[t]
  }
  dY <- diff(Y)
  Y_lag <- Y[-n]
  model <- lm(dY ~ Y_lag)
  t_d10[i] <- summary(model)$coefficients[2, 3]
}
```

```r
# Densités
dens_none <- density(t_none)
dens_5 <- density(t_d5)
dens_10 <- density(t_d10)

# Graphique final
plot(dens_none, col = "darkgreen", lwd = 2,
     main = expression("Distribution des statistiques t : sans constante vs " * delta
     xlab = "Valeurs de la statistique t", ylab = "Density",
     xlim = c(-6, 4), ylim = c(0, 0.55))
lines(dens_5, col = "purple", lwd = 2, lty = 2)
lines(dens_10, col = "black", lwd = 2, lty = 3)

legend("topright",
       legend = c("Sans constante", expression(delta[0] == 5), expression(delta[0] ==
       col = c("darkgreen", "purple", "black"),
       lty = c(1, 2, 3), lwd = 2, bty = "n")
```



Interprétation

Concernant le graphqiue ci dessus, nous observer comme indiqué plus haut, une certaine différence entre le fait de rajouter une constante et de pas en avoir. En effet, avoir une constance entraine que la courbe se décalle vers la gauche.

Néanmoins, concernnat la différence de $\delta_0 = 5$ et $\delta_0 = 10$, il n'existe pas dedifférence visuelle. Les deux courbes se suivent.

Tableau de comparaison

Seuil	Sans constante	$\delta_0 = 5$	$\delta_0 = 10$	Table DF (n = 100)
10%	-1.630	-2.590	-2.590	-1.61
5%	-1.988	-2.885	-2.885	-1.95
1%	-2.633	-3.476	-3.476	-2.60

Comme nous pouvons l'observer avec ce tableau, il existe une réélle différence entre le fait d'avoir une constant et non. En effet, avec une constante on s'éloigne des résultats de la table de Dickey-Fuller.

Néanmoins, entre $\delta_0 = 5$ et $\delta_0 = 10$ il n'existe pas de différence donc la différence se fait suelement sur le fait d'ajouter une constante ou non.