

Projet statistiques bayésiennes

Garance MALNOË et Matthias MAZET

2026-01-30

Contents

1	Introduction	2
2	Simulations du processus	2
2.1	<i>Variations de α</i>	3
2.2	<i>Variations de β</i>	4
2.3	<i>Variations de θ</i>	6
3	Étude de l'inférence bayésienne	8
3.1	<i>Contexte</i>	8
3.2	<i>Fonctions implémentées</i>	10
3.3	<i>Variations de β</i>	14
3.4	<i>Variations de α</i>	20
3.5	<i>Variations de θ</i>	26
3.6	<i>Variations des paramètres relatifs à l'expert</i>	32
4	Conclusion	40
5	Utilisation de l'IA	40

```
# Packages nécessaires
library(ggplot2)
library(ggpubr)
library(GGally)
library(invgamma)
```

1 Introduction

Nous souhaitons ici étudier un processus gamma X non homogène, et plus particulièrement l'inférence bayésienne de ses paramètres.

Pour cela, nous supposons que ce processus possède un *paramètre de forme* $a(t) = \alpha t^\beta$, avec $\alpha, \beta > 0$, et un *paramètre d'échelle* $\theta > 0$. Nous supposons aussi que, pour tous $t > s \geq 0$, $X(t) - X(s) \sim \mathcal{G}(a(t) - a(s), \theta)$, et nous avons alors, pour tout $x \in \mathbb{R}_+$:

$$f_{a(t)-a(s),\theta}(x) = \frac{x^{a(t)-a(s)-1}}{\theta^{a(t)-a(s)} \Gamma(a(t) - a(s))} e^{-\frac{x}{\theta}}.$$

Par la construction d'une inférence bayésienne, nous allons donc chercher à estimer les trois paramètres introduits α , β et θ .

Afin d'analyser la pertinence de l'approche bayésienne face à un problème de ce type, nous avons séparé notre analyse en deux phases : i) la simulation du processus selon différentes valeurs de α , β et θ afin d'observer leur impact respectif sur l'évolution du processus, et ii) l'étude sur l'inférence bayésienne, en la comparant notamment à l'estimation par maximum de vraisemblance et en regardant l'impact des variations des différents paramètres du modèle (α , β , θ et les paramètres relatifs à l'expert).

2 Simulations du processus

Afin de tester l'effet de chaque paramètre sur l'évolution du processus, nous regardons individuellement l'impact de chacun d'eux. Pour cela, nous définissons 2 fonctions qui vont nous permettre de réaliser les simulations tout au long de cette analyse :

- La fonction `simulations`, qui prend en entrée les paramètres α , β et θ du processus Gamma ainsi que la taille des pas de simulation et d'inspection, et qui retourne une liste des grilles de pas (simulation et inspection) et du processus simulé à partir de celles-ci (simulation et inspection).
- La fonction `plot_simulations`, qui prend en entrée la sortie de `simulations` et qui retourne le graphique du processus associé, en distinguant les points simulés des points inspectés.

```
# Fonction pour simuler un processus gamma
simulations <- function(horizon = 100, dt_sim = 0.01, dt_insp = 5, alpha, beta, theta){
  # Simulations du processus continu, avec un pas de simulation dt_sim
  grille_sim <- seq(0, horizon, by = dt_sim)
  a <- function(t) {alpha*t^beta} # val de a(t)
  shape <- diff(a(grille_sim)) # a(t) - a(s)
  increments <- rgamma(length(shape), shape=shape, scale=theta)
  X_sim <- c(0, cumsum(increments))
  # Calcul des inspections, avec un pas d'inspection dt_insp
  grille_insp <- seq(0, horizon, by = dt_insp)
  n <- dt_insp / dt_sim
  X_insp <- X_sim[seq(1, length(X_sim), by = n)]
  # Résultats
  list(
    grille_sim = grille_sim, X_sim = X_sim,      # Résultats simulés
    grille_insp = grille_insp, X_insp = X_insp  # Résultats inspectés
  )}
```

```

}

# Fonction graphique d'un processus
plot_simulations <- function(simulations, title = "") {
  ggplot() +
    geom_line(
      aes(x = simulations$grille_sim, y = simulations$X_sim, col = "Simulées")
    ) +
    geom_point(
      aes(x = simulations$grille_insp, y = simulations$X_insp, col = "Inspectées"),
      shape = 18, size = 2
    ) +
    scale_color_manual(
      name = "Type de données",
      values = c("Simulées" = "#2d0569", "Inspectées" = "#e3573e"),
    ) +
    labs(x = "t", y = expression(X[t]), title = title) +
    theme_light() +
    theme(
      axis.title = element_text(size = 10, face = "bold"),
      axis.text = element_text(size = 8),
      panel.border = element_blank(),
      axis.line = element_line(colour = "darkgrey"),
      panel.grid.minor = element_blank(),
      panel.grid.major = element_line(color = "grey85"),
      legend.title = element_text(size = 10, face = "bold"),
      legend.text = element_text(size = 8),
      legend.position = "right"
    )
}

```

2.1 Variations de α

Commençons par observer l'effet de α sur le processus. Pour cela, nous fixons les autres paramètres du modèle : nous fixons l'horizon à 100, le pas de simulation à 0.01, le pas d'inspection à 5, $\beta = 1$ et $\theta = 2$. Nous prenons ensuite $\alpha \in \{0.1, 1, 10, 20\}$ afin d'observer son effet.

```

# Seed pour la reproductibilité
set.seed(1)

# Paramètres de simulations
dt_sim <- 0.01
dt_insp <- 5
beta <- 1
theta <- 2
# Valeurs de alpha
list_alpha <- c(.1, 1, 10, 20)

# Simulations selon les valeurs de alpha de list_alpha et les paramètres fixés
sim1 <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = list_alpha[1], beta = beta, theta = theta
)
plot_sim1 <- plot_simulations(
  sim1, title = bquote(alpha == .(list_alpha[1]))
)
sim2 <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = list_alpha[2], beta = beta, theta = theta
)
plot_sim2 <- plot_simulations(

```

```

    sim2, title = bquote(alpha == .(list_alpha[2]))
)
sim3 <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = list_alpha[3], beta = beta, theta = theta
)
plot_sim3 <- plot_simulations(
  sim3, title = bquote(alpha == .(list_alpha[3]))
)
sim4 <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = list_alpha[4], beta = beta, theta = theta
)
plot_sim4 <- plot_simulations(
  sim4, title = bquote(alpha == .(list_alpha[4]))
)

ggarrange(plot_sim1, plot_sim2, plot_sim3, plot_sim4, common.legend = TRUE)

```

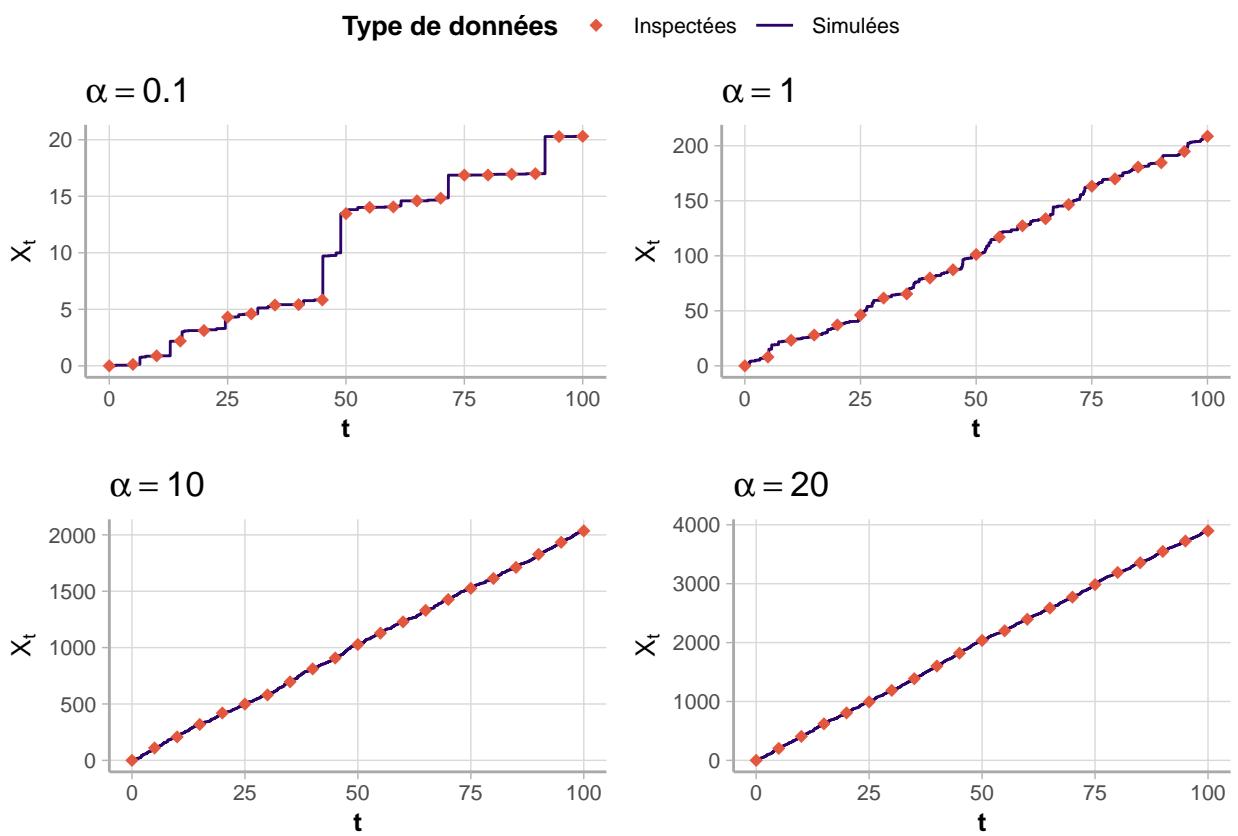


Figure 1: Simulations d'un processus gamma non-homogène ($\beta = 1$, $\theta = 2$) pour différents α

Nous constatons donc que α n'a pas d'impact sur la *forme* du processus, mais seulement sur l'intervalle des valeurs de X_t . La vitesse de dégradation du processus semble être proportionnelle (de l'ordre de 200) à α : plus le paramètre est grand, plus le processus se dégrade rapidement (1).

2.2 Variations de β

Observons maintenant l'effet de β . Comme précédemment, nous fixons les autres paramètres du modèle : nous fixons l'horizon à 100, le pas de simulation à 0.01, le pas d'inspection à 5, $\alpha = 1$ et $\theta = 2$. Nous prenons ensuite $\beta \in \{0.5, 1, 2, 3\}$ afin d'observer son effet.

```

# Seed pour la reproductibilité
set.seed(1)

```

```

# Paramètres de simulations
dt_sim <- 0.01
dt_insp <- 5
alpha <- 1
theta <- 2
# Valeurs de beta
list_beta <- c(0.5, 1, 2, 3)

# Simulations selon les valeurs de alpha de list_beta et les paramètres fixés
sim1 <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = list_beta[1], theta = theta
)
plot_sim1 <- plot_simulations(
  sim1, title = bquote(beta == .(list_beta[1])))
)
sim2 <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = list_beta[2], theta = theta
)
plot_sim2 <- plot_simulations(
  sim2, title = bquote(beta == .(list_beta[2])))
)
sim3 <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = list_beta[3], theta = theta
)
plot_sim3 <- plot_simulations(
  sim3, title = bquote(beta == .(list_beta[3])))
)
sim4 <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = list_beta[4], theta = theta
)
plot_sim4 <- plot_simulations(
  sim4, title = bquote(beta == .(list_beta[4])))
)

ggarrange(plot_sim1, plot_sim2, plot_sim3, plot_sim4, common.legend = TRUE)

```

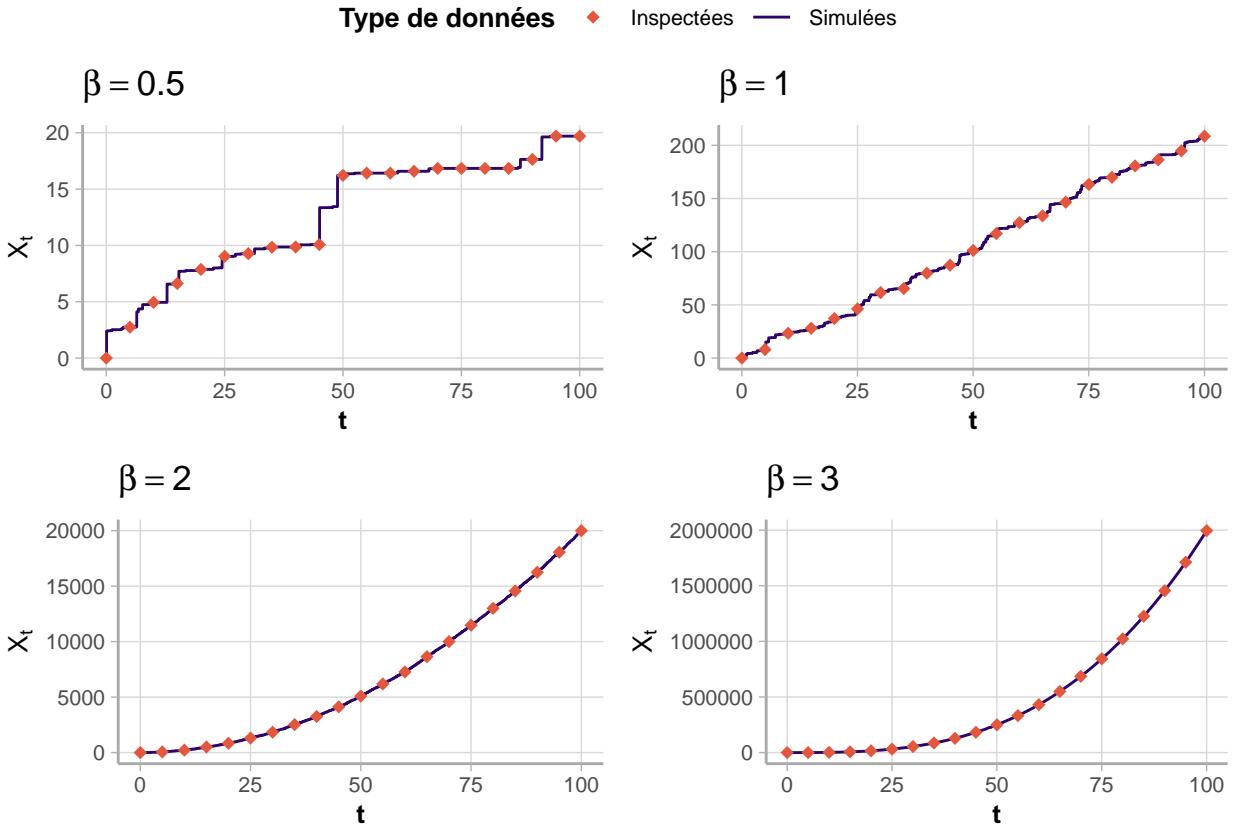


Figure 2: Simulations d'un processus gamma non-homogène ($\alpha = 1$, $\theta = 2$) pour différents β

Comme attendu, l'effet de β varie selon l'intervalle de valeurs dans lequel il se trouve :

- Pour un $\beta < 1$, la courbe est concave, i.e. la dégradation ralentit au cours du temps.
- Pour $\beta = 1$, la courbe est linéaire, i.e. la dégradation évolue toujours de la même manière au cours du temps.
- Pour un $\beta > 1$, la courbe est convexe, i.e. la dégradation s'accélère au cours du temps. Nous constatons d'ailleurs que, plus β est grand, plus l'accélération de dégradation est forte (différence entre $\beta = 2$ et $\beta = 3$) (2).

2.3 Variations de θ

Observons finalement l'effet de θ . Comme précédemment, nous fixons les autres paramètres du modèle : nous fixons l'horizon à 100, le pas de simulation à 0.01, le pas d'inspection à 5, $\alpha = \beta = 1$. Nous prenons ensuite $\theta \in \{0.5, 1, 2, 5\}$ afin d'observer son effet.

```
# Seed pour la reproductibilité
set.seed(1)

# Paramètres de simulations
dt_sim <- 0.01
dt_insp <- 5
alpha <- 1
beta <- 1
# Valeurs de theta
list_theta <- c(0.5, 1, 2, 5)

# Simulations selon les valeurs de alpha de list_beta et les paramètres fixés
sim1 <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = beta, theta = list_theta[1])
)
plot_sim1 <- plot_simulations(
  sim1, title = bquote(theta == .(list_theta[1])))
```

```

)
sim2 <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = beta, theta = list_theta[2]
)
plot_sim2 <- plot_simulations(
  sim2, title = bquote(theta == .(list_theta[2]))
)
sim3 <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = beta, theta = list_theta[3]
)
plot_sim3 <- plot_simulations(
  sim3, title = bquote(theta == .(list_theta[3]))
)
sim4 <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = beta, theta = list_theta[4]
)
plot_sim4 <- plot_simulations(
  sim4, title = bquote(theta == .(list_theta[4]))
)

ggarrange(plot_sim1, plot_sim2, plot_sim3, plot_sim4, common.legend = TRUE)

```

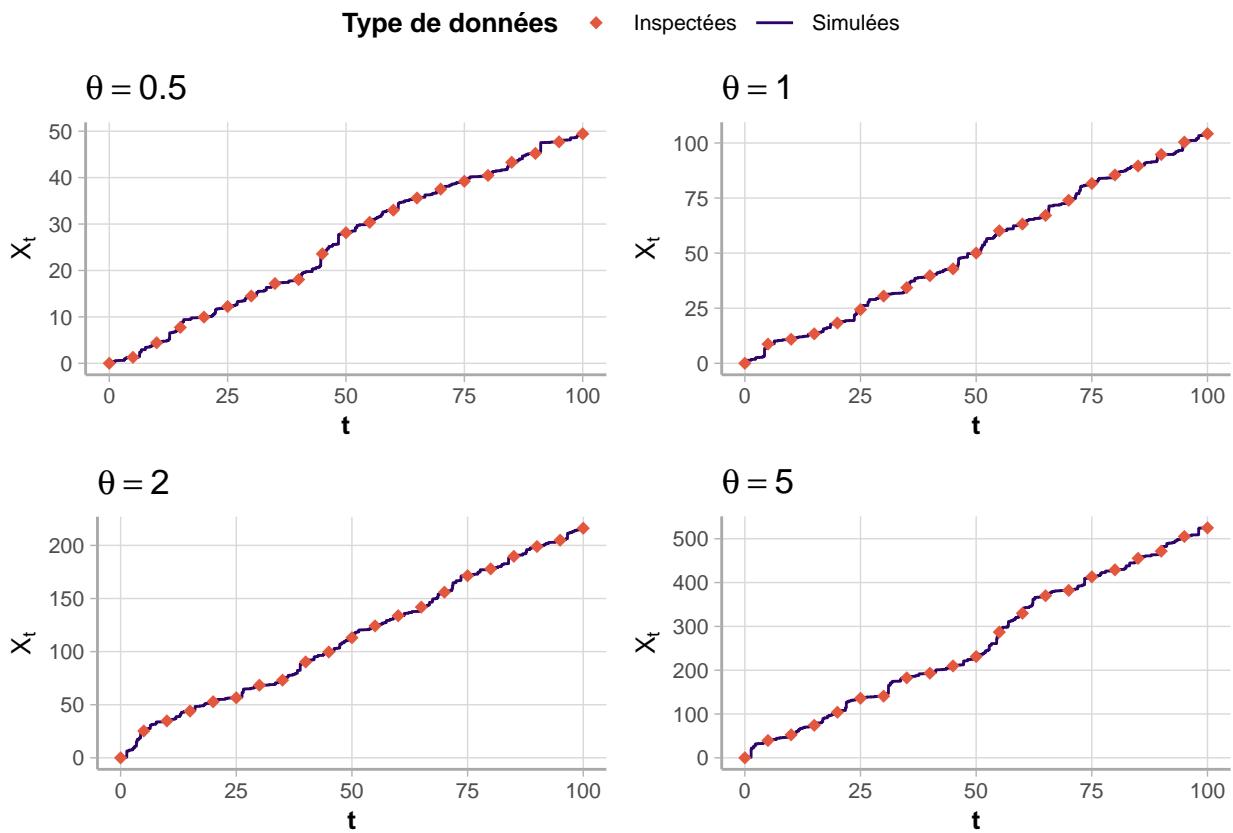


Figure 3: Simulations d'un processus gamma non-homogène ($\alpha = 1$, $\beta = 1$) pour différents θ

Comme pour α , θ ne semble pas avoir d'impact sur la *forme* du processus, mais seulement sur la vitesse de dégradation du processus. Les deux semblent encore une fois être proportionnels : plus θ est grand, plus le processus se dégrade rapidement. Toutefois, l'influence de θ semble moins forte que celle de α , le coefficient de proportionnalité étant ici de l'ordre de 100, contre 200 pour α (3).

3 Étude de l'inférence bayésienne

3.1 Contexte

Nous supposons maintenant que nous disposons d'un échantillon d'inspection du niveau de dégradation $x = \{x_0, \dots, x_n\}$, obtenu à différents temps d'inspection $t = \{t_0, \dots, t_n\}$ et issus d'un processus gamma non-homogène de paramètres α , β et θ . Nous souhaitons étudier la qualité de l'inférence bayésienne multidimensionnelle pour l'estimation de ces trois paramètres lorsque nous faisons varier les valeurs des paramètres, la qualité de la prédiction de l'expert ainsi que la confiance accordée à l'expert.

Puisque nous sommes dans un cas multidimensionnel, nous allons donc passer par les algorithmes de Gibbs et de Metropolis-Hastings, en testant différentes combinaisons de loi a priori pour les 3 paramètres. Les trois paramètres étant positifs, nous restreignons le choix des lois théoriques de chaque paramètre à un certain nombre de lois à support positif :

- Pour α , nous testerons soit une *loi log-normale* $\mathcal{L}\mathcal{N}(\mu, \sigma)$, soit une *loi gamma* $\mathcal{G}(k, s)$.
- Pour β nous testerons soit une *loi log-normale* $\mathcal{L}\mathcal{N}(\mu, \sigma)$, soit une *loi uniforme* $\mathcal{U}(a, b)$.
- Pour θ , nous utiliserons toujours une *loi inverse Gamma* $\mathcal{I}\mathcal{N}\mathcal{G}(a, b)$.

Nous supposons également que l'expert nous indique toujours une estimation μ_{exp} du paramètre à estimer (et non directement les paramètres de sa loi a priori) à laquelle nous associons une certaine confiance σ_{exp} .

Pour l'étude de l'impact des variations des paramètres α , β et θ sur la qualité de l'inférence, nous choisissons μ_{exp} et σ_{exp} en partant du principe que nous avons un bon expert à qui nous faisons confiance. Nous choisissons donc $\mu_{exp} = \frac{6}{5}\mu$, avec μ la véritable valeur du paramètre, et $\sigma_{exp} = 0.2 \times \mu_{exp}$.

Pour l'étude de l'impact des paramètres relatifs à l'expert, nous choisissons $\mu_{exp} \in \{\frac{6}{5}\mu, 2\mu\}$ pour un expert bon ou mauvais respectivement et $\sigma_{exp} \in \{0.2\mu_{exp}, 0.8\mu_{exp}\}$ pour une confiance élevée ou faible respectivement.

Afin de juger de la pertinence de l'approche bayésienne, nous allons comparer ses estimations à celles obtenues par l'approche du maximum de vraisemblance. Nous calculons donc la vraisemblance du modèle.

Pour l'ensemble des lois, nous allons plutôt travailler avec les incrémentations $y_i = x_i - x_{i-1} = x_{(t_i)} - x_{(t_{i-1})}$, $i \in \{1, \dots, n\}$.

Nous posons également $k_i(\alpha, \beta) = a_{(t_i)} - a_{(t_{i-1})} = \alpha(t_i^\beta - t_{i-1}^\beta)$ afin d'alléger les calculs suivants.

Par hypothèse, nous avons $X(t_i) - X(t_{i-1}) = Y_i \sim \mathcal{G}(k_i(\alpha, \beta), \theta)$, et donc :

$$f(y_i|k_i(\alpha, \beta), \theta) = \frac{1}{\Gamma(k_i(\alpha, \beta))\theta^{k_i(\alpha, \beta)}} y_i^{k_i(\alpha, \beta)-1} e^{\frac{y_i}{\theta}}$$

Puisque les incrémentations sont supposées indépendantes, la vraisemblance des données y sachant $\alpha, \beta, \theta > 0$ est donc :

$$\begin{aligned} L(y|\alpha, \beta, \theta) &= \prod_{i=1}^n f(y_i|k_i(\alpha, \beta), \theta) \\ &= \prod_{i=1}^n \frac{1}{\Gamma(k_i(\alpha, \beta))\theta^{k_i(\alpha, \beta)}} y_i^{k_i(\alpha, \beta)-1} e^{\frac{y_i}{\theta}} \end{aligned}$$

Afin de réaliser l'inférence bayésienne, nous calculons maintenant les différentes lois a posteriori obtenues à partir des différentes lois a priori que nous allons tester dans notre analyse.

3.1.1 Calcul de la loi a posteriori pour θ

Nous avons choisi comme loi a priori pour θ une $\mathcal{I}\mathcal{N}\mathcal{G}(a, b)$. Ainsi :

$$\begin{aligned} p(\theta|y, \alpha, \beta) &= L(y|\alpha, \beta, \theta) \times p(\theta) \\ &\propto \theta^{\sum k_i(\alpha, \beta)} e^{-\sum \frac{y_i}{\theta}} \theta^{-(a+1)} e^{-\frac{b}{\theta}} \\ &\propto \theta^{-(\sum k_i(\alpha, \beta)+a+1)} e^{-\sum \frac{y_i+\beta}{\theta}} \end{aligned}$$

Nous pouvons reconnaître, à une constante de normalisation près, la densité d'une $\mathcal{I}\mathcal{N}\mathcal{G}(a + \sum k_i(\alpha, \beta), b + \sum y_i)$. Nous pourrons donc estimer θ avec un algorithme de Gibbs.

En supposant que les paramètres d'expert μ_θ et σ_θ correspondent à $\mu_\theta = \mathbb{E}[\mathcal{I}\mathcal{N}\mathcal{G}(a, b)] = \frac{b}{a-1}$ et $\sigma_\theta = \text{Var}(\mathcal{I}\mathcal{N}\mathcal{G}(a, b)) = \frac{b^2}{(a-1)^2(a-2)}$, nous avons alors :

$$\begin{cases} a = \frac{\mu_\theta^2}{\sigma_\theta^2} + 2 \\ b = (a-1)\mu_\theta \end{cases} .$$

3.1.2 Calcul des lois a posteriori pour α

1. Calcul pour une $\text{Log-N}(\mu, \sigma)$.

Nous avons :

$$\begin{aligned} p(\alpha|y, \beta, \theta) &= L(y|\alpha, \beta, \theta) \times p(\alpha) \\ &\propto \frac{1}{\alpha \sigma \sqrt{2\pi}} e^{-\frac{(ln(\alpha) - \mu)^2}{2\sigma^2}} \prod_i \frac{1}{\Gamma(k_i(\alpha, \beta)) \theta^{k_i(\alpha, \beta)}} y_i^{k_i(\alpha, \beta)} \end{aligned}$$

Comme nous ne reconnaissions pas de forme de loi usuelle, nous allons utiliser l'algorithme de Metropolis-Hastings. Nous choisissons comme distribution instrumentale pour α une loi normale centrée en la valeur précédente de α dans la chaîne de Markov $\alpha^{(k-1)}$ afin de nous ramener à un rapport des lois a posteriori entre la nouvelle valeur $\tilde{\alpha}$ et l'ancienne valeur $\alpha^{(k-1)}$. Nous notons τ_α la variance de la normale et nous avons choisi de fixer sa valeur à 0.3 pour l'ensemble de nos analyses. Pour faciliter le calcul numérique, nous regardons $\ln(p(\tilde{\alpha}|y, \beta, \theta)) - \ln(p(\alpha^{(k-1)}|y, \beta, \theta))$ plutôt que le rapport $\frac{p(\tilde{\alpha}|y, \beta, \theta)}{p(\alpha^{(k-1)}|y, \beta, \theta)}$. Nous avons donc :

$$\ln(p(\alpha|y, \beta, \theta)) = -\ln(\alpha) - \frac{(ln(\alpha) - \mu)^2}{2\sigma^2} + \sum_{i=1}^n (k_i(\alpha, \beta) - 1)\ln(y_i) - k_i(\alpha, \beta)\ln(\theta) - \ln(\Gamma(k_i(\alpha, \beta))) + \text{cste}$$

En supposant que les paramètres d'expert μ_α et σ_α correspondent à $\mu_\alpha = \mathbb{E}[\text{Log-N}(\mu, \sigma)] = e^{\frac{\mu+\sigma^2}{2}}$ et $\sigma_\alpha = \text{Var}(\text{Log-N}(\mu, \sigma)) = (e^{\sigma^2} - 1)e^{2\mu+\sigma^2}$, nous avons alors :

$$\begin{cases} \mu = \ln(\mu_\alpha) - \frac{\ln(1 + \frac{\sigma_\alpha}{\mu_\alpha^2})}{2} \\ \sigma = \sqrt{\ln(\frac{\sigma_\alpha}{\mu_\alpha^2} + 1)} \end{cases} .$$

2. Calcul pour une $\mathcal{G}(k, s)$.

Nous avons :

$$\begin{aligned} p(\alpha|y, \beta, \theta) &= L(y|\alpha, \beta, \theta) \times p(\alpha) \\ &\propto \frac{1}{\Gamma(k)s^k} \alpha^{k-1} e^{-\alpha/s} \prod_i \frac{1}{\Gamma(k_i(\alpha, \beta)) \theta^{k_i(\alpha, \beta)}} y_i^{k_i(\alpha, \beta)} \end{aligned}$$

Encore une fois, nous ne reconnaissions pas de forme de loi usuelle. Nous allons donc utiliser l'algorithme de Metropolis-Hastings. Nous choisissons comme distribution instrumentale pour α une loi normale centrée en la valeur précédente de α dans la chaîne de Markov $\alpha^{(k-1)}$ afin de nous ramener à un rapport des lois a posteriori entre la nouvelle valeur $\tilde{\alpha}$ et l'ancienne valeur $\alpha^{(k-1)}$. Nous notons τ_α la variance de la normale et nous avons choisi de fixer sa valeur à 0.3 pour l'ensemble de nos analyses. Pour faciliter le calcul numérique, nous regardons $\ln(p(\tilde{\alpha}|y, \beta, \theta)) - \ln(p(\alpha^{(k-1)}|y, \beta, \theta))$ plutôt que le rapport $\frac{p(\tilde{\alpha}|y, \beta, \theta)}{p(\alpha^{(k-1)}|y, \beta, \theta)}$. Nous avons donc :

$$\ln(p(\alpha|y, \beta, \theta)) = (k-1)\ln(\alpha) - \frac{\alpha}{s} + \sum_{i=1}^n (k_i(\alpha, \beta) - 1)\ln(y_i) - k_i(\alpha, \beta)\ln(\theta) - \ln(\Gamma(k_i(\alpha, \beta))) + \text{cste}$$

En supposant que les paramètres d'expert μ_α et σ_α correspondent à $\mu_\alpha = \mathbb{E}[\mathcal{G}(k, s)] = \frac{k}{s}$ et $\sigma_\alpha = \text{Var}(\mathcal{G}(k, s)) = \frac{k}{s^2}$, nous avons alors :

$$\begin{cases} k = \frac{\mu_\alpha^2}{\sigma_\alpha^2} \\ s = \frac{\sigma_\alpha}{\mu_\alpha} \end{cases} .$$

3.1.3 Calcul des lois a posteriori pour β

1. Calcul pour une $\text{Log-N}(\mu, \sigma)$.

Par inversion des rôles entre α et β , nous retrouvons les résultats déjà exhibés plus haut. Nous avons donc :

$$\begin{cases} \mu = \ln(\mu_\beta) - \frac{\ln(1 + \frac{\sigma_\beta}{\mu_\beta^2})}{2} \\ \sigma = \sqrt{\ln(\frac{\sigma_\beta}{\mu_\beta^2} + 1)} \end{cases} .$$

Nous choisissons comme distribution instrumentale pour β une loi normale centrée en la valeur précédente de β dans la chaîne de Markov $\beta^{(k-1)}$ afin de nous ramener à un rapport des lois a posteriori entre la nouvelle valeur $\tilde{\beta}$ et l'ancienne valeur $\beta^{(k-1)}$. Nous notons τ_β la variance de la normale et nous avons choisi de fixer sa valeur à 0.3 pour l'ensemble de nos analyses. Comme pour α , nous regarderons le logarithme des lois a posteriori conditionnelles pour faciliter les calculs numériques.

2. Calcul pour une $\mathcal{U}(a, b)$.

Nous avons :

$$\begin{aligned} p(\beta|y, \alpha, \theta) &= L(y|\alpha, \beta, \theta) \times p(\beta) \\ &\propto \mathbf{1}_{\beta \in [a, b]} \prod_i \frac{1}{\Gamma(k_i(\alpha, \beta)) \theta^{k_i(\alpha, \beta)}} y_i^{k_i(\alpha, \beta)} \end{aligned}$$

Encore une fois, nous ne reconnaissions pas de forme de loi usuelle. Nous allons donc utiliser l'algorithme de Metropolis-Hastings. Nous choisissons comme distribution instrumentale pour β une loi normale centrée en la valeur précédente de β dans la chaîne de Markov $\beta^{(k-1)}$ afin de nous ramener à un rapport des lois a posteriori entre la nouvelle valeur $\tilde{\beta}$ et l'ancienne valeur $\beta^{(k-1)}$. Nous notons τ_β la variance de la normale et nous avons choisi de fixer sa valeur à 0.3 pour l'ensemble de nos analyses. Pour faciliter le calcul numérique, nous regardons $\log(p(\tilde{\beta}|y, \alpha, \theta)) - \log(p(\beta^{(k-1)}|y, \alpha, \theta))$ plutôt que le rapport $\frac{p(\tilde{\beta}|y, \alpha, \theta)}{p(\beta^{(k-1)}|y, \alpha, \theta)}$. Nous avons donc :

$$\ln(p(\beta|y, \alpha, \theta)) = \begin{cases} \sum_{i=1}^n (k_i(\alpha, \beta) - 1) \ln(y_i) - k_i(\alpha, \beta) \ln(\theta) - \ln(\Gamma(k_i(\alpha, \beta))) + \text{cste} & \text{si } \beta \in [a, b], \\ -\infty & \text{sinon} \end{cases}$$

En supposant que les paramètres d'expert μ_β et σ_β correspondent à $\mu_\beta = \mathbb{E}[\mathcal{U}(a, b)] = \frac{a+b}{2}$ et $\sigma_\beta = \text{Var}(\mathcal{U}(a, b)) = \frac{(a-b)^2}{12}$, nous avons alors :

$$\begin{cases} a = \mu_\beta - \sqrt{3\sigma_\beta} \\ b = \mu_\beta + \sqrt{3\sigma_\beta} \end{cases} .$$

3.2 Fonctions implémentées

Afin de fluidifier notre analyse, nous avons implémenté un certain nombre de fonctions utiles à l'étude.

3.2.1 Lois a posteriori

Nous avons implémenté une fonction pour chacune des lois a posteriori conditionnelles de α et β calculées précédemment. Nous avons aussi implémenté une fonction pour calculer les $k_i(\alpha, \beta)$ introduits précédemment.

```
# Fonction de calcul des k_i
func_list_ki <- function(alpha, beta, t_insp) {
  n <- length(t_insp) - 1
  alpha * (t_insp[2:(n+1)]^beta - t_insp[1:n]^beta)
}

# Lois a posteriori pour alpha
### Log-normale(mu,sigma)
logpost_alpha_lognormale <- function(alpha, y, t_insp, beta, theta, mu_alpha, sigma_alpha) {
  mu <- log(mu_alpha) - 0.5 * log(1 + sigma_alpha/(mu_alpha^2))
  sigma <- sqrt(log(1 + sigma_alpha/(mu_alpha^2)))
  list_ki <- func_list_ki(alpha, beta, t_insp)
  -log(alpha) - ((log(alpha)-mu)^2)/(2*sigma^2) +
    sum((list_ki - 1)*log(y) - list_ki*log(theta) - lgamma(list_ki))
}

### Gamma(k,s)
logpost_alpha_gamma <- function(alpha, y, t_insp, beta, theta, mu_alpha, sigma_alpha) {
  k <- mu_alpha^2 / sigma_alpha
  s <- mu_alpha / sigma_alpha
```

```

list_ki <- func_list_ki(alpha, beta, t_insp)
(k-1)*log(alpha) - alpha/s +
  sum((list_ki - 1)*log(y) - list_ki*log(theta) - lgamma(list_ki))
}

# Lois a posteriori pour beta
### Log-normale(mu,sigma)
logpost_beta_lognormale <- function(beta, y, t_insp, alpha, theta, mu_beta, sigma_beta) {
  mu <- log(mu_beta) - 0.5 * log(1 + sigma_beta/(mu_beta^2))
  sigma <- sqrt(log(1 + sigma_beta/(mu_beta^2)))
  list_ki <- func_list_ki(alpha, beta, t_insp)
  -log(beta) - ((log(beta)-mu)^2)/(2*sigma^2) +
    sum((list_ki - 1)*log(y) - list_ki*log(theta) - lgamma(list_ki))
}
### Uniforme(a,b)
logpost_beta_unif <- function(beta, y, t_insp, alpha, theta, mu_beta, sigma_beta) {
  a <- mu_beta - sqrt(3*sigma_beta)
  b <- mu_beta + sqrt(3*sigma_beta)
  if (beta < a || beta > b) {
    -Inf
  } else {
    list_ki <- func_list_ki(alpha, beta, t_insp)
    sum((list_ki - 1)*log(y) - list_ki*log(theta) - lgamma(list_ki))
  }
}

```

3.2.2 Algorithme de Gibbs-Metropolis-Hastings (GMH)

La fonction `algo_GMH` implémente un algorithme MCMC de type Gibbs–Metropolis–Hastings (GMH) permettant d'obtenir K estimations bayésiennes de chaque paramètre (α , β et θ). Elle prend en entrée les arguments suivants :

- `logpost_par` la fonction utilisée pour calculer `log(posteriori(par))`, avec `par` un des paramètres du modèle.
- `mu_par` la moyenne de l'expert pour le paramètre `par`.
- `sigma_par` la confiance en l'expert pour le paramètre `par`.
- `tau_par` la variance de la normale (loi instrumentale) pour le paramètre `par` (fixé à 0.3).
- `par0` l'initialisation du paramètre `par` pour la chaîne de markov.
- `data` les x_1, \dots, x_n observations (à transformer en y_1, \dots, y_n) les incrémentums.
- `t_insp` les t_1, \dots, t_n temps d'inspection.
- `K` le nombre d'itérations des algorithmes.
- `burnin` le nombre d'itérations à supprimer du début de la liste des résultats, le “temps de chauffe” de l'algorithme.

```

algo_GMH <- function(
  logpost_alpha, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
) {
  # Transformation des données en incrémentums  $y_i = x(t_i) - x(t_{i-1})$ 
  y <- data[2:length(data)] - data[1:length(data)-1]
  n <- length(y)
  # Calcul du a et b pour simuler theta
  a_theta <- mu_theta^2 / sigma_theta + 2
  b_theta <- (a_theta-1) * mu_theta
  # Initialisation des vecteurs de résultats
  res_alpha <- numeric(K+1)
  res_alpha[1] <- alpha0
  res_beta <- numeric(K+1)
  res_beta[1] <- beta0
  res_theta <- numeric(K+1)
  res_theta[1] <- theta0
  # Boucle sur k pour calculer de  $\alpha^{(k)}$ ,  $\beta^{(k)}$  et  $\theta^{(k)}$ 
  for(k in 2:(K+1)) {

```

```

# 1. Simulation de theta^(k)
list_ki <- func_list_ki(res_alpha[k-1], res_beta[k-1], t_insp)
a_bis <- a_theta + sum(list_ki)
b_bis <- b_theta + sum(y)
theta_hat_moment <- sum(y) / sum(list_ki)
res_theta[k] <- 1 / rgamma(1, shape = a_bis, rate = b_bis)
# 2. Simulation de alpha^(k) (MH)
alpha_tilde <- rnorm(1, mean = res_alpha[k-1], sd = tau_alpha)
if (alpha_tilde <= 0) {
  res_alpha[k] <- res_alpha[k-1]
} else {
  first_log <- logpost_alpha(
    alpha_tilde, y, t_insp, res_beta[k-1], res_theta[k], mu_alpha, sigma_alpha
  )
  second_log <- logpost_alpha(
    res_alpha[k-1], y, t_insp, res_beta[k-1], res_theta[k], mu_alpha, sigma_alpha
  )
  log_r <- first_log - second_log
  u <- runif(1)
  res_alpha[k] <- ifelse(log(u) <= log_r, alpha_tilde, res_alpha[k-1])
}
# 3. Simulation de beta^(k) (MH)
beta_tilde <- rnorm(1, mean = res_beta[k-1], sd = tau_beta)
if (beta_tilde <= 0) {
  res_beta[k] <- res_beta[k-1]
} else {
  first_log <- logpost_beta(
    beta_tilde, y, t_insp, res_alpha[k], res_theta[k], mu_beta, sigma_beta
  )
  second_log <- logpost_beta(
    res_beta[k-1], y, t_insp, res_alpha[k], res_theta[k], mu_beta, sigma_beta
  )
  log_r <- first_log - second_log
  u <- runif(1)
  res_beta[k] <- ifelse(log(u) <= log_r, beta_tilde, res_beta[k-1])
}
}

# Résultats avec burnin
return(list(
  res_theta = tail(res_theta, n = K-burnin),
  res_alpha = tail(res_alpha, n = K-burnin),
  res_beta = tail(res_beta, n = K-burnin))
)
}
}

```

3.2.3 Estimateurs du maximum de vraisemblance (MLE) par optimisation

Afin d'obtenir les MLE, nous cherchons à minimiser $-\ln(Vraisemblance)$. Nous implémentons donc d'abord une fonction permettant de retourner cet objet (`neg_vraisemblance`), puis nous implémentons une fonction permettant d'exhiber les MLE obtenus (`mle`).

```

# -ln(Vraisemblance)
neg_vraisemblance <- function(vect_param, y, t_insp) {
  alpha <- vect_param[1]
  beta <- vect_param[2]
  theta <- vect_param[3]
  list_ki <- func_list_ki(alpha, beta, t_insp)
  if (any(!is.finite(list_ki)) || any(list_ki <= 0) || !is.finite(theta) || theta <= 0) {
    1e100
  } else if (any(!is.finite(y)) || any(y <= 0)) {
    1e100
  } else {

```

```

        -(sum((list_ki - 1) * log(y) - (y/theta) - list_ki * log(theta) - lgamma(list_ki)))
    }
}

# Calcul des MLE via optimisation
mle <- function(alpha0, beta0, theta0, data, t_insp) {
  y <- data[2:length(data)] - data[1:(length(data)-1)]
  init <- c(alpha0, beta0, theta0)
  fit <- optim(
    par = init,
    fn = neg_vraisemblance,
    y = y,
    t_insp = t_insp,
    method = "BFGS",
    control = list(maxit = 5000, reltol = 1e-10)
  )
  list(
    alpha = fit$par[1],
    beta = fit$par[2],
    theta = fit$par[3],
    converged = (fit$convergence == 0)
  )
}

```

3.2.4 Fonction de visualisation

Nous implémentons finalement une fonction nous permettant de visualiser les résultats obtenus simultanément (`vis_res`). Elle nous permettra notamment de comparer les vraies données aux estimations bayésiennes et par maximum de vraisemblance.

```

vis_res <- function(res_GMH, res_MLE, sim_og, title = "") {
  # Récupération des hyperparamètres de la simulation originale en jeu
  horizon <- sim_og$grille_sim[length(sim_og$grille_sim)]
  dt_sim <- sim_og$grille_sim[2] - sim_og$grille_sim[1]
  dt_insp <- sim_og$grille_insp[2] - sim_og$grille_insp[1]
  # Simulation du processus avec les paramètres estimés par GMH
  alpha_GMH <- mean(res_GMH$res_alpha)
  beta_GMH <- mean(res_GMH$res_beta)
  theta_GMH <- mean(res_GMH$res_theta)
  simulations_GMH <- simulations(
    horizon, dt_sim, dt_insp, alpha = alpha_GMH, beta = beta_GMH, theta = theta_GMH
  )
  # Simulation du processus avec les paramètres estimés par maximum de vraisemblance
  alpha_MLE <- res_MLE$alpha
  beta_MLE <- res_MLE$beta
  theta_MLE <- res_MLE$theta
  simulations_MLE <- simulations(
    horizon, dt_sim, dt_insp, alpha = alpha_MLE, beta = beta_MLE, theta = theta_MLE
  )
  # Dataframe des résultats au format "long" (pour ggplot)
  df_long <- data.frame(
    grille_sim = sim_og$grille_sim,
    X_sim = sim_og$X_sim,
    X_sim_GMH = simulations_GMH$X_sim,
    X_sim_MLE = simulations_MLE$X_sim
  )
  # Sous-titre
  subt <- bquote(
    atop(
      MLE : alpha == .(round(alpha_MLE, 3)) * "," ~
        beta == .(round(beta_MLE, 3)) * "," ~
        theta == .(round(theta_MLE, 3)),
      .(round(theta_GMH, 3))
    )
  )
}
```

```

GMH : alpha == .(round(alpha_GMH, 3)) * "," ~
      beta == .(round(beta_GMH, 3)) * "," ~
      theta == .(round(theta_GMH, 3))
    )
}

# Visualisation des résultats sur une même figure
ggplot() +
  geom_line(aes(
    x = df_long$grille_sim, y = df_long$X_sim,
    col = "Processus original"
  )) +
  geom_line(aes(
    x = df_long$grille_sim, y = df_long$X_sim_GMH,
    col = "GMH"
  )) +
  geom_line(aes(
    x = df_long$grille_sim, y = df_long$X_sim_MLE,
    col = "MLE"
  )) +
  scale_color_manual(
    name = "Données",
    values = c(
      "Processus original" = "#e3573e",
      "GMH" = "#49a86c",
      "MLE" = "#2d0569"
    ),
  ) +
  labs(x = "t", y = expression(X[t]), title = title, subtitle = subt) +
  theme_light() +
  theme(
    axis.title = element_text(size = 10, face = "bold"),
    plot.subtitle = element_text(size = 7),
    axis.text = element_text(size = 8),
    panel.border = element_blank(),
    axis.line = element_line(colour = "darkgrey"),
    panel.grid.minor = element_blank(),
    panel.grid.major = element_line(color = "grey85"),
    legend.title = element_text(size = 10, face = "bold"),
    legend.text = element_text(size = 8),
    legend.position = "right"
  )
}

```

3.3 Variations de β

D'après nos constats précédents (partie **Simulations du processus**), β module la forme de la courbe selon trois cas : $\beta < 1$, $\beta = 1$ et $\beta > 1$. Nous allons donc distinguer ces trois cas afin d'analyser et comparer les estimations bayésiennes et par maximum de vraisemblance. Afin de garder une cohérence entre les résultats, nous fixons l'ensemble des autres paramètres de simulations à une valeur fixe :

```

# Paramètres de simulation
dt_sim <- 0.01
dt_insp <- 5
alpha <- 1
theta <- 2
# Paramètres de l'algorithme GMH
K <- 10000
burnin <- 1000

```

Nous fixons aussi les données relatives à l'expert pour α et θ . Nous considérons le cas où il est bon et où nous lui faisons confiance :

```
# Bonne confiance en un expert bon
### Expertise sur alpha
mu_alpha <- 6*alpha/5
sigma_alpha <- mu_alpha*0.2
tau_alpha <- 0.3
alpha0 <- mu_alpha
### Expertise sur theta
mu_theta <- 6*theta/5
sigma_theta <- mu_theta*0.2
theta0 <- mu_theta
```

3.3.1 $\beta < 1$, cas concave

Commençons par regarder le cas où $\beta < 1$, par exemple $\beta = 0.5$. Comme pour α et θ , nous considérons le cas où l'expert est bon et où nous lui faisons confiance pour l'expertise de β .

```
# Seed pour la reproductibilité
set.seed(1)

# 1. Simulation du processus
beta <- 0.5
sim_og <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = beta, theta = theta
)
data <- sim_og$X_insp
t_insp <- sim_og$grille_insp
### Expertise sur beta : bonne confiance en un expert bon
mu_beta <- 6*beta/5
sigma_beta <- beta*0.2
tau_beta <- 0.3
beta0 <- mu_beta
### Estimations par maximum de vraisemblance
res_MLE <- mle(alpha0, beta0, theta0, data, t_insp)

# 2. alpha ~ Log-normale et beta ~ Log-normale
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot1 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Gamma et beta ~ Lognormale
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot2 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Log-N")
)

# 4. alpha ~ Log-normale et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
```

```

logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot3 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Uniforme")
)

# 5. alpha ~ Gamma et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot4 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Uniforme")
)

# Visualisation finale
ggarrange(plot1, plot2, plot3, plot4, common.legend = TRUE)

```

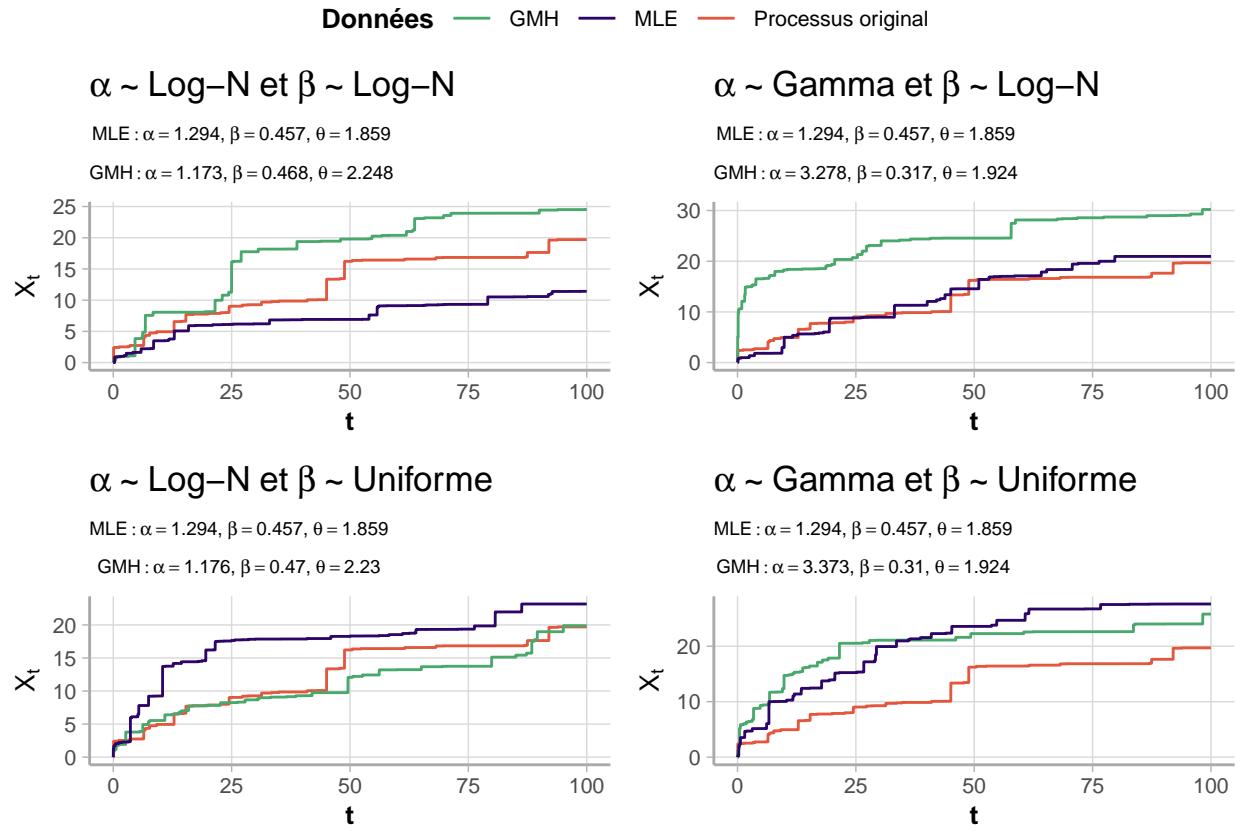


Figure 4: Résultats d'estimations bayésiennes et par maximum de vraisemblance lorsque $\beta = 0.5$, $\alpha = 1$ et $\theta = 2$

Dans ce contexte, la meilleure combinaison de lois a priori en termes d'estimation par GMH des trois paramètres semble donc être soit $\alpha \sim \mathcal{L}\text{-N}$ et $\beta \sim \mathcal{L}\text{-N}$, soit $\alpha \sim \mathcal{L}\text{-N}$ et $\beta \sim \mathcal{U}$. En effet, lorsque $\alpha \sim \mathcal{G}$, l'algorithme tend à sous-évaluer β et sur-évaluer α . L'algorithme MLE tend également à surestimer α , plus que l'algorithme GMH pour $\alpha \sim \mathcal{L}\text{-N}$. Suivant le choix des lois et le paramètre considéré, GMH peut être meilleur ou moins bon que MLE : le choix de l'un ou l'autre dépend donc de la justesse des lois a priori choisies et de quel (4).

3.3.2 $\beta = 1$, cas linéaire

Regardons maintenant le cas où $\beta = 1$. Comme précédemment, nous considérons le cas où l'expert est bon et où nous lui faisons confiance pour l'expertise de β et reprenons les mêmes valeurs de α et θ .

```
# Seed pour la reproductibilité
set.seed(1)

# 1. Simulation du processus
beta <- 1
sim_og <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = beta, theta = theta
)
data <- sim_og$X_insp
t_insp <- sim_og$grille_insp
### Expertise sur beta : bonne confiance en un expert bon
mu_beta <- 6*beta/5
sigma_beta <- beta*0.2
tau_beta <- 0.3
beta0 <- mu_beta
### Estimations par maximum de vraisemblance
res_MLE <- mle(alpha0, beta0, theta0, data, t_insp)

# 2. alpha ~ Log-normale et beta ~ Log-normale
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot1 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Gamma et beta ~ Lognormale
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot2 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Log-normale et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot3 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "log-N" ~ "et" ~ beta %~% "Uniforme")
)

# 4. alpha ~ Gamma et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot4 <- vis_res(
```

```

res_GMH, res_MLE, sim_og,
title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Uniforme")
)

# Visualisation finale
ggarrange(plot1, plot2, plot3, plot4, common.legend = TRUE)

```

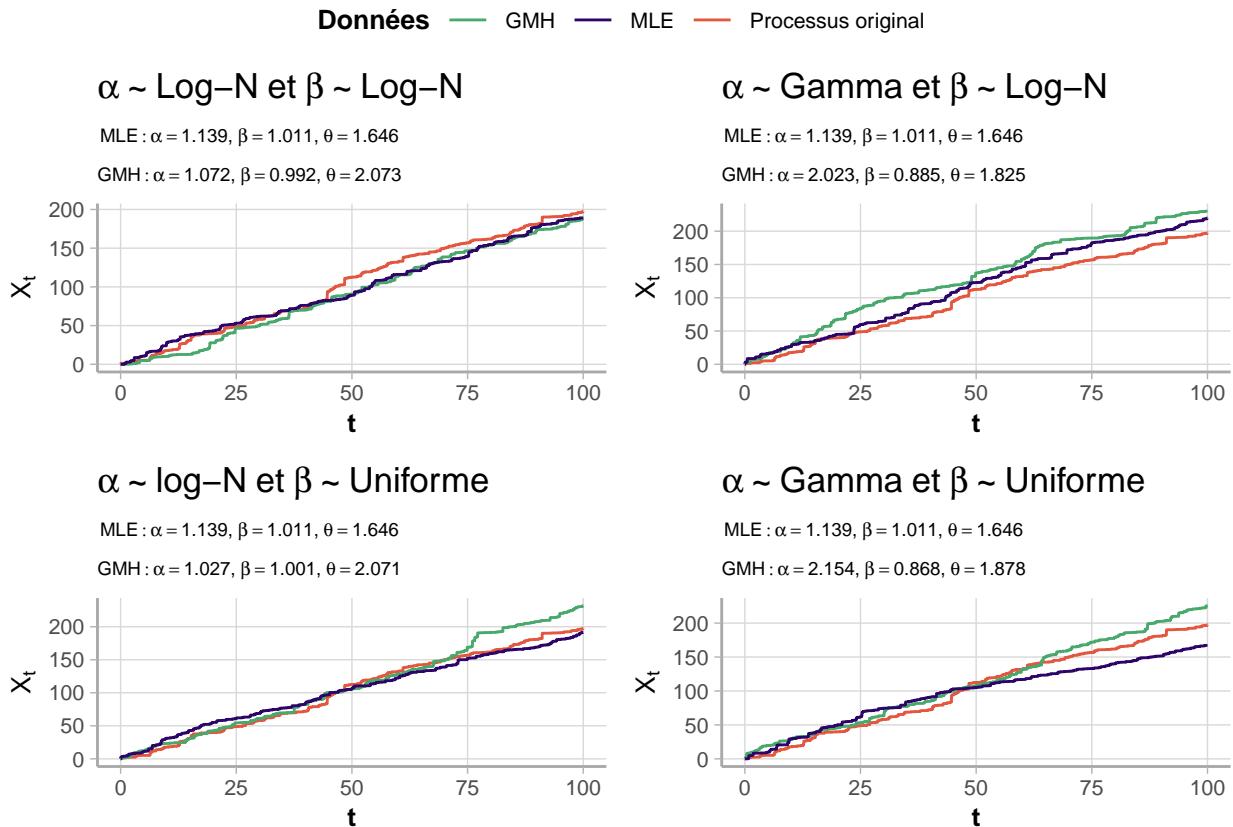


Figure 5: Résultats d'estimations bayésiennes et par maximum de vraisemblance lorsque $\beta = 1$, $\alpha = 1$ et $\theta = 2$

Encore une fois, les meilleures combinaisons de lois en termes d'estimation GMH *des trois paramètres* semblent être soit $\alpha \sim \mathcal{L}\text{-N}$ et $\beta \sim \mathcal{L}\text{-N}$, soit $\alpha \sim \mathcal{L}\text{-N}$ et $\beta \sim \mathcal{U}$. Nous constatons aussi que MLE sous-estime θ et sur-estime légèrement α . Ainsi, dans ce cas là, le meilleur choix d'estimations semble être le bayésien, avec une des deux combinaisons citées (5).

3.3.3 $\beta > 1$, cas convexe

Regardons finalement le cas où $\beta > 1$, par exemple $\beta = 3$. Comme précédemment, nous considérons le cas où l'expert est bon et où nous lui faisons confiance pour l'expertise de β et reprenons les mêmes valeurs de α et θ .

```

# Seed pour la reproductibilité
set.seed(1)

# 1. Simulation du processus
beta <- 3
sim_og <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = beta, theta = theta
)
data <- sim_og$X_insp
t_insp <- sim_og$grille_insp
### Expertise sur beta : bonne confiance en un expert bon
mu_beta <- 6*beta/5
sigma_beta <- beta*0.2
tau_beta <- 0.3

```

```

beta0 <- mu_beta
### Estimations par maximum de vraisemblance
res_MLE <- mle(alpha0, beta0, theta0, data, t_insp)

# 2. alpha ~ Log-normale et beta ~ Log-normale
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot1 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Gamma et beta ~ Lognormale
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot2 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Log-normale et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot3 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "log-N" ~ "et" ~ beta %~% "Uniforme")
)

# 4. alpha ~ Gamma et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot4 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Uniforme")
)

# Visualisation finale
ggarrange(plot1, plot2, plot3, plot4, common.legend = TRUE)

```

Données — GMH — MLE — Processus original

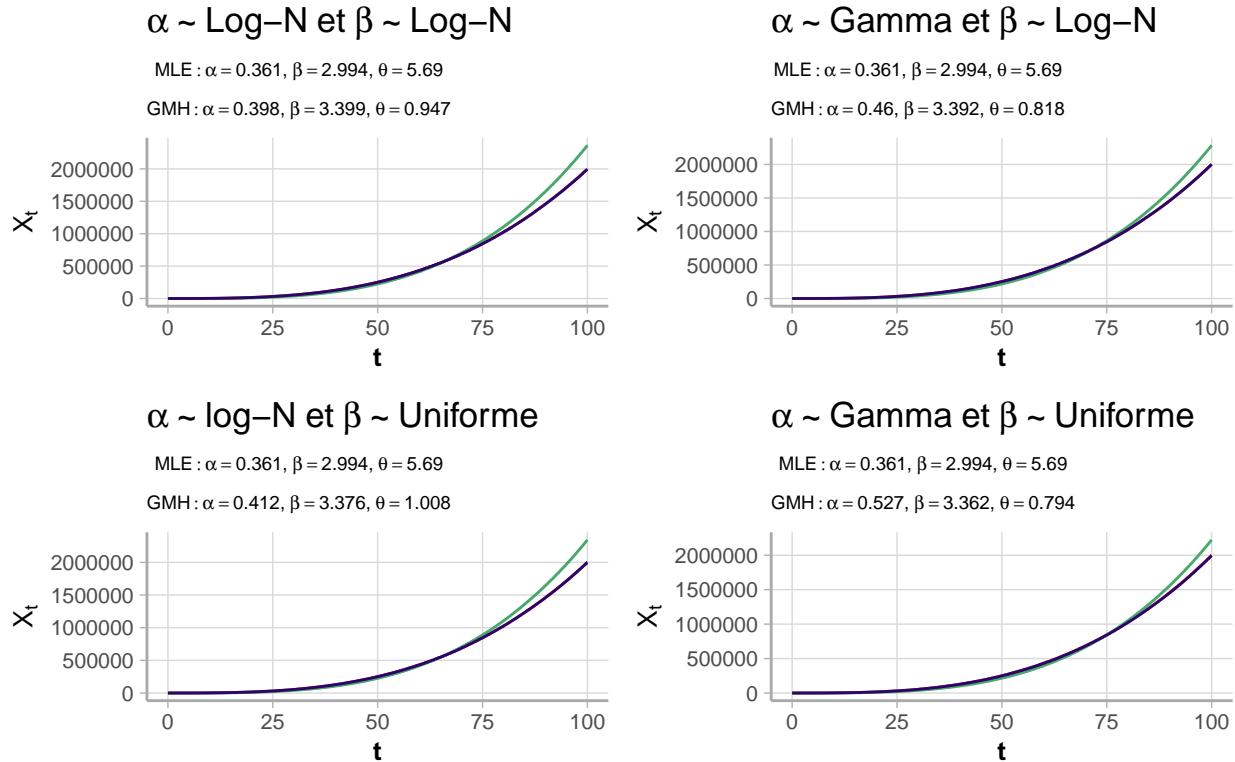


Figure 6: Résultats d'estimations bayésiennes et par maximum de vraisemblance lorsque $\beta = 3$, $\alpha = 1$ et $\theta = 2$

Visuellement, nous constatons que MLE semble être très bon pour approcher le processus, les deux courbes étant confondues. GMH semble plutôt le surestimer lorsque $t > 75$. Toutefois, en regardant de plus près les estimations de chaque paramètre, nous remarquons que seule l'estimation de β par MLE est bonne. L'algorithme MLE sur-estime largement θ et sous-estime α . Ainsi, aucune des deux estimations ne semble bien adaptée. Le choix portera donc plutôt sur l'objectif final d'estimation, i.e. retrouver les vrais paramètres ou retrouver la forme du processus (6).

3.4 Variations de α

Nous regardons maintenant l'impact des variations de α sur la qualité de l'inférence bayésienne. Nous avons constaté que α agissait comme un coefficient de proportionnalité sur X_t , c'est pourquoi nous allons observer les variations pour $\alpha \in \{1, 10, 100\}$. Nous fixons $\beta = 2$ et $\theta = 2$ et gardons les mêmes paramètres que précédemment pour le pas de simulations, d'inspection, d'horizon ainsi que pour la qualité de l'estimation de l'expert μ_{exp} et la confiance accordée σ_{exp} .

```
# Paramètres de simulation
dt_sim <- 0.01
dt_insp <- 5
beta <- 2
theta <- 2
# Paramètres de l'algorithme GMH
K <- 10000
burnin <- 1000
```

Nous fixons aussi les données relatives à l'expert pour β et θ . Nous considérons le cas où il est bon et où nous lui faisons confiance :

```
# Bonne confiance en un expert bon
### Expertise sur beta
mu_beta <- 6*beta/5
sigma_beta <- mu_beta*0.2
```

```

tau_beta <- 0.3
beta0 <- mu_beta
### Expertise sur theta
mu_theta <- 6*theta/5
sigma_theta <- mu_theta*0.2
theta0 <- mu_theta

```

3.4.1 $\alpha = 1$

Commençons par regarder le cas où $\alpha = 1$. Comme pour β et θ , nous considérons le cas où l'expert est bon et où nous lui faisons confiance pour l'expertise de α .

```

# Seed pour la reproductibilité
set.seed(1)

# 1. Simulation du processus
alpha <- 1
sim_og <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = beta, theta = theta
)
data <- sim_og$X_insp
t_insp <- sim_og$grille_insp
### Expertise sur beta : bonne confiance en un expert bon
mu_alpha <- 6*alpha/5
sigma_alpha <- alpha*0.2
tau_alpha <- 0.3
alpha0 <- mu_alpha
### Estimations par maximum de vraisemblance
res_MLE <- mle(alpha0, beta0, theta0, data, t_insp)

# 2. alpha ~ Log-normale et beta ~ Log-normale
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot1 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Gamma et beta ~ Lognormale
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot2 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Log-normale et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot3 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Uniforme")
)

```

```

# 4. alpha ~ Gamma et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot4 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Uniforme")
)

# Visualisation finale
ggarrange(plot1, plot2, plot3, plot4, common.legend = TRUE)

```

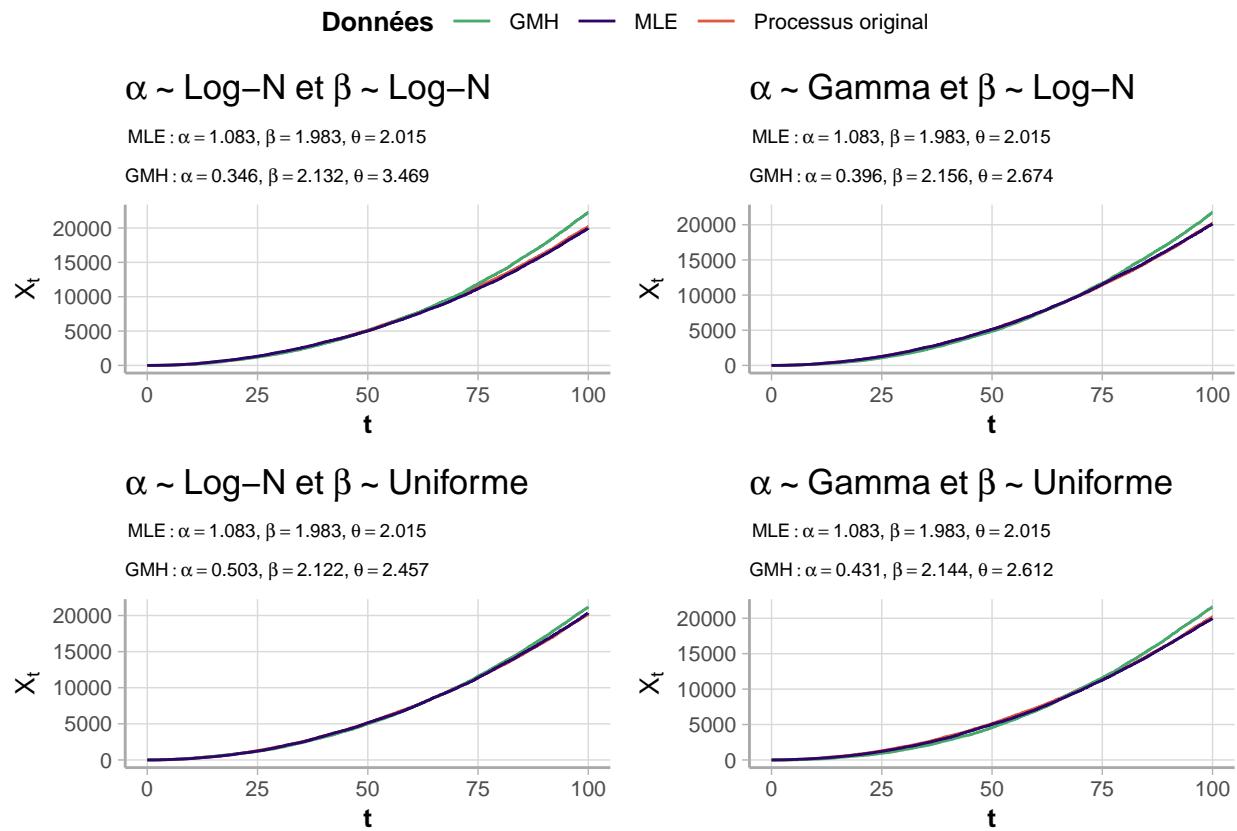


Figure 7: Résultats d'estimations bayésiennes et par maximum de vraisemblance lorsque $\alpha = 1$, $\beta = 2$ et $\theta = 2$

Peu importe la combinaison de lois a priori choisies, GMH est toujours mauvais pour estimer les paramètres du processus, et il se trompe toujours de la même manière : il sur-estime θ et sous-estime α . Cette erreur peut nous faire penser au rôle similaire que jouent ces deux paramètres sur la forme de la courbe et les valeurs du processus, i.e. de coefficient de proportionnalité. À l'inverse, MLE donne une bonne estimation de chaque paramètre. Cependant, nous constatons sur la figure que les deux courbes estimées restent toujours assez proches du vrai processus. Il semble donc préférable d'utiliser MLE ici (7).

3.4.2 $\alpha = 10$

Nous regardons ensuite le cas $\alpha = 10$, toujours avec les mêmes conditions de départ pour les autres paramètres.

```

# Seed pour la reproductibilité
set.seed(1)

# 1. Simulation du processus
alpha <- 10

```

```

sim_og <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = beta, theta = theta
)
data <- sim_og$X_insp
t_insp <- sim_og$grille_insp
### Expertise sur alpha : bonne confiance en un expert bon
mu_alpha <- 6*alpha/5
sigma_alpha <- alpha*0.2
tau_alpha <- 0.3
alpha0 <- mu_alpha
### Estimations par maximum de vraisemblance
res_MLE <- mle(alpha0, beta0, theta0, data, t_insp)

# 2. alpha ~ Log-normale et beta ~ Log-normale
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot1 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Gamma et beta ~ Lognormale
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot2 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Log-normale et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot3 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Uniforme")
)

# 4. alpha ~ Gamma et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot4 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Uniforme")
)

# Visualisation finale
ggarrange(plot1, plot2, plot3, plot4, common.legend = TRUE)

```

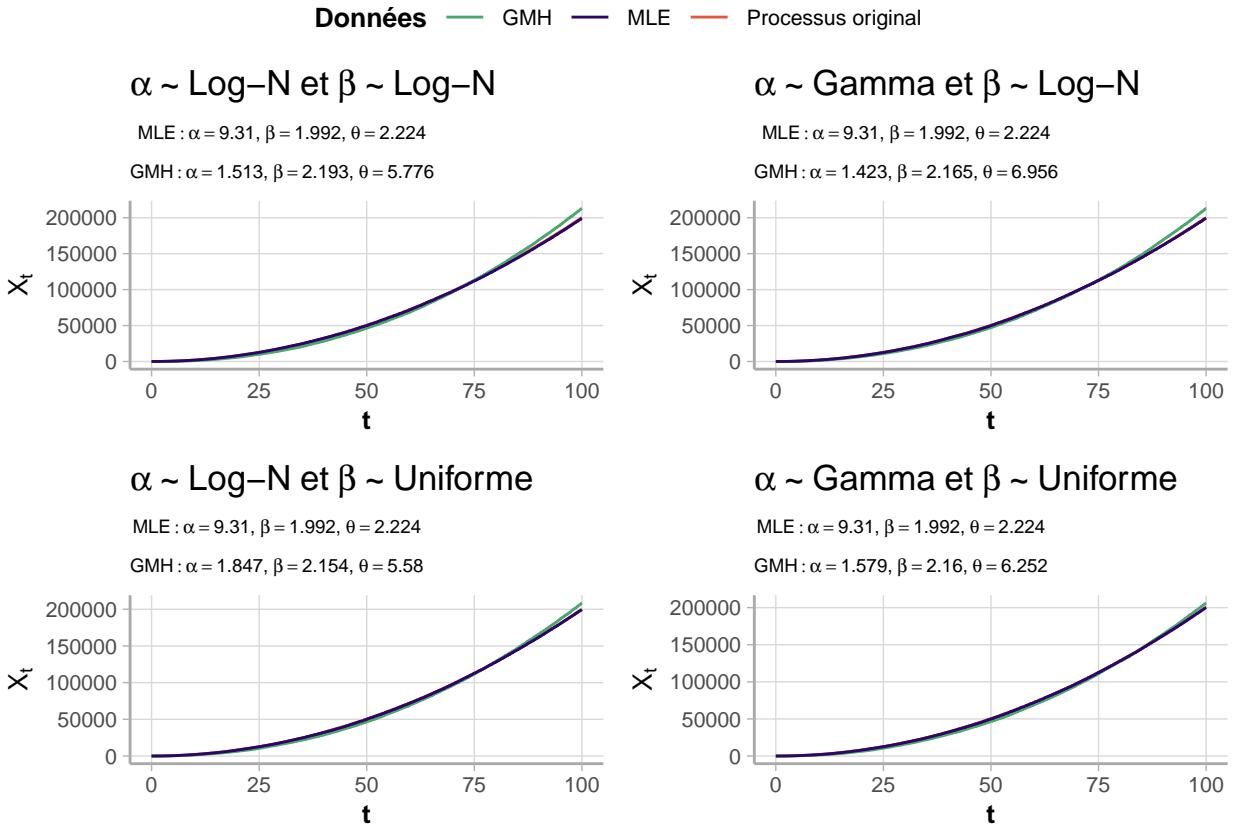


Figure 8: Résultats d'estimations bayésiennes et par maximum de vraisemblance lorsque $\alpha = 10$, $\beta = 2$ et $\theta = 2$

Encore une fois, GMH sous-estime dans tous les cas α et sur-estime θ . MLE reste encore assez performant, bien que nous puissions constater une dégradation de ses estimations (elles sont moins proches des vraies valeurs que précédemment). Les deux courbes estimées restent encore une fois toujours assez proches du vrai processus même si on observe un léger décalage pour GMH pour les 4 combinaisons (8).

3.4.3 $\alpha = 100$

Nous regardons finalement $\alpha = 100$, toujours avec les mêmes conditions de départ pour les autres paramètres.

```
# Seed pour la reproductibilité
set.seed(1)

# 1. Simulation du processus
alpha <- 100
sim_og <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = beta, theta = theta
)
data <- sim_og$X_insp
t_insp <- sim_og$grille_insp
### Expertise sur alpha : bonne confiance en un expert bon
mu_alpha <- 6*alpha/5
sigma_alpha <- alpha*0.2
tau_alpha <- 0.3
alpha0 <- mu_alpha
### Estimations par maximum de vraisemblance
res_MLE <- mle(alpha0, beta0, theta0, data, t_insp)

# 2. alpha ~ Log-normale et beta ~ Log-normale
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
```

```

)
plot1 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Gamma et beta ~ Lognormale
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot2 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Log-normale et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot3 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Uniforme")
)

# 4. alpha ~ Gamma et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot4 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Uniforme")
)

# Visualisation finale
ggarrange(plot1, plot2, plot3, plot4, common.legend = TRUE)

```

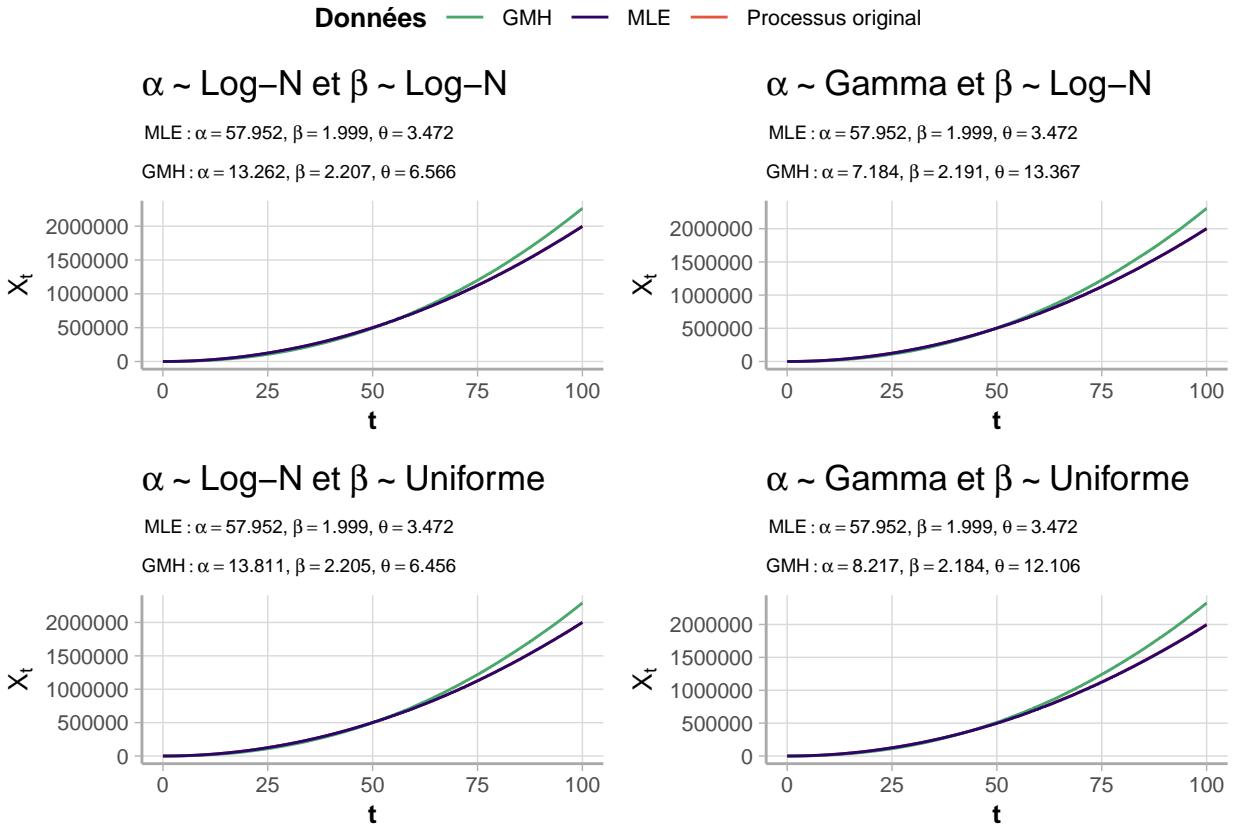


Figure 9: Résultats d'estimations bayésiennes et par maximum de vraisemblance lorsque $\alpha = 100$, $\beta = 2$ et $\theta = 2$

Les observations suivent la ligne de conduite observée précédemment : GMH reste mauvais pour estimer α et θ , les estimations de MLE continuent de se dégrader et les courbes estimées pour MLE restent assez proches du vrai processus tandis que celles de GMH se décalent encore davantage (9).

3.5 Variations de θ

Nous regardons maintenant l'impact des variations de θ sur la qualité de l'inférence bayésienne. Nous avions constaté que θ , comme α , agissait comme un coefficient de proportionnalité sur X_t , c'est pourquoi nous allons observer les variations pour $\alpha \in \{1, 10, 100\}$. Nous fixons $\alpha = 1$ et $\beta = 2$ et gardons les mêmes paramètres que précédemment pour le pas de simulations, d'inspection, d'horizon ainsi que pour la qualité de l'estimation de l'expert μ_{exp} et la confiance accordée σ_{exp} .

```
# Paramètres de simulation
dt_sim <- 0.01
dt_insp <- 5
beta <- 2
alpha <- 1
# Paramètres de l'algorithme GMH
K <- 10000
burnin <- 1000
```

Nous fixons aussi les données relatives à l'expert pour β et θ . Nous considérons le cas où il est bon et où nous lui faisons confiance :

```
# Bonne confiance en un expert bon
### Expertise sur beta
mu_beta <- 6*beta/5
sigma_beta <- mu_beta*0.2
tau_beta <- 0.3
beta0 <- mu_beta
```

```

### Expertise sur alpha
mu_alpha <- 6*alpha/5
sigma_alpha <- mu_alpha*0.2
alpha0 <- mu_alpha

```

3.5.1 $\theta = 1$

Commençons par regarder le cas où $\alpha = 1$. Comme pour β et α , nous considérons le cas où l'expert est bon et où nous lui faisons confiance pour l'expertise de α .

```

# Seed pour la reproductibilité
set.seed(1)

# 1. Simulation du processus
theta <- 1
sim_og <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = beta, theta = theta
)
data <- sim_og$X_insp
t_insp <- sim_og$grille_insp
### Expertise sur theta : bonne confiance en un expert bon
mu_theta <- 6*theta/5
theta0 <- mu_theta
### Estimations par maximum de vraisemblance
res_MLE <- mle(alpha0, beta0, theta0, data, t_insp)

# 2. alpha ~ Log-normale et beta ~ Log-normale
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot1 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Gamma et beta ~ Lognormale
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot2 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Log-normale et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot3 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Uniforme")
)

# 4. alpha ~ Gamma et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,

```

```

logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot4 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Uniforme")
)

# Visualisation finale
ggarrange(plot1, plot2, plot3, plot4, common.legend = TRUE)

```

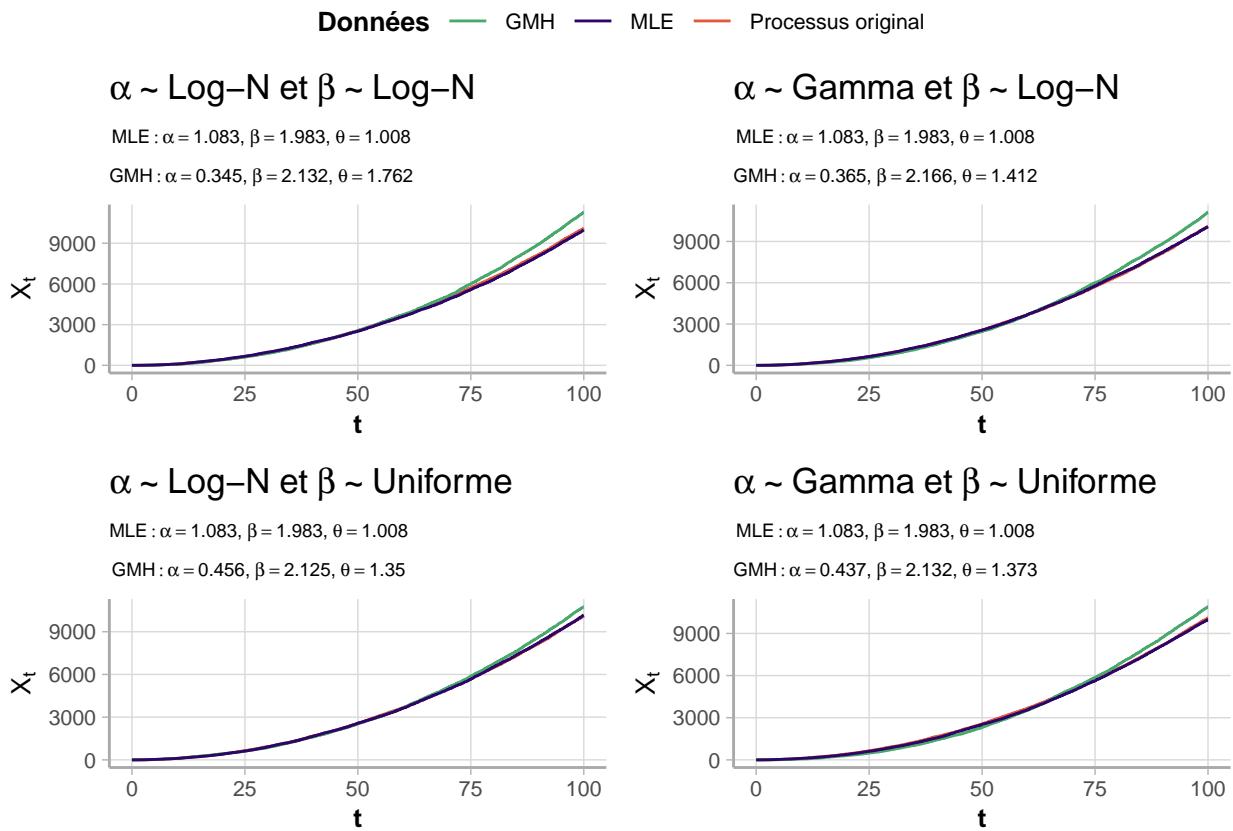


Figure 10: Résultats d'estimations bayésiennes et par maximum de vraisemblance lorsque $\theta = 1$, $\beta = 2$ et $\alpha = 1$

Nous constatons que GMH estime assez bien β mais sur-estime θ et sous-estime α . À l'inverse, MLE estime bien les trois paramètres. Les courbes estimées restent par contre toujours assez proches du vrai processus. Comme pour les analyses sur les variations de α , nous pouvons supposer que ces mauvaises estimations sont entraînées par le rôle similaire de α et θ sur X_t . Nous pouvons aussi supposer que, lorsque nous allons augmenter θ , GMH sera toujours mauvais et la qualité de MLE va se dégrader (**10**).

3.5.2 $\theta = 10$

Nous regardons maintenant l'impact pour $\theta = 10$, toujours avec les mêmes conditions de départ pour les autres paramètres.

```

# Seed pour la reproductibilité
set.seed(1)

# 1. Simulation du processus
theta <- 10
sim_og <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = beta, theta = theta
)
data <- sim_og$X_insp

```

```

t_insp <- sim_og$grille_insp
### Expertise sur theta : bonne confiance en un expert bon
mu_theta <- 6*theta/5
sigma_theta <- theta*0.2
theta0 <- mu_theta
### Estimations par maximum de vraisemblance
res_MLE <- mle(alpha0, beta0, theta0, data, t_insp)

# 2. alpha ~ Log-normale et beta ~ Log-normale
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot1 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Gamma et beta ~ Lognormale
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot2 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Log-normale et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot3 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Uniforme")
)

# 4. alpha ~ Gamma et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot4 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Uniforme")
)

# Visualisation finale
ggarrange(plot1, plot2, plot3, plot4, common.legend = TRUE)

```

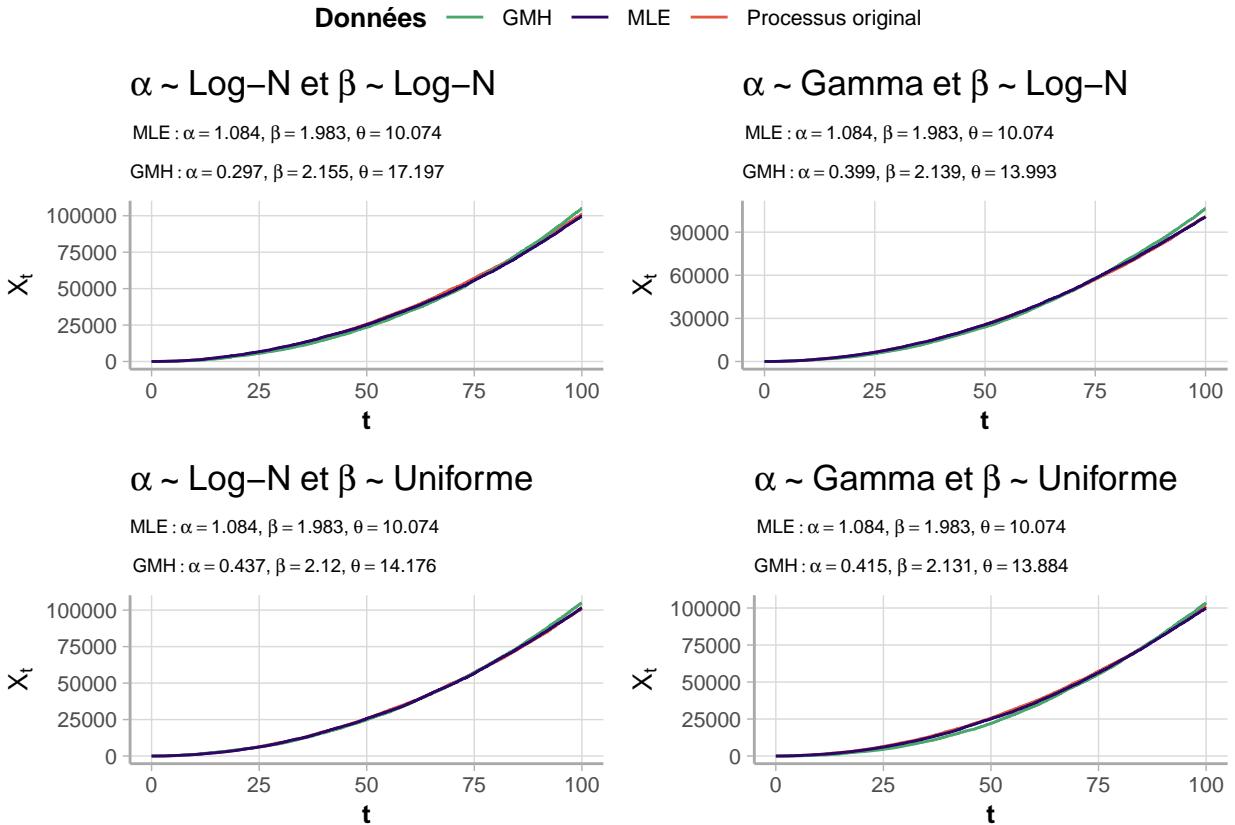


Figure 11: Résultats d'estimations bayésiennes et par maximum de vraisemblance lorsque $\theta = 10$, $\beta = 2$ et $\alpha = 1$

Contrairement à ce que nous avions intuité, la qualité d'estimation de MLE ne semble pas s'être dégradée ici. Par contre, GMH reste toujours mauvais pour estimer α et θ . Encore une fois, les courbes d'estimations restent toujours proches du vrai processus (11).

3.5.3 $\theta = 100$

Nous regardons finalement l'impact pour $\theta = 100$, toujours avec les mêmes conditions de départ pour les autres paramètres.

```
# Seed pour la reproductibilité
set.seed(1)

# 1. Simulation du processus
theta <- 100
sim_og <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = beta, theta = theta
)
data <- sim_og$X_insp
t_insp <- sim_og$grille_insp
### Expertise sur theta : bonne confiance en un expert bon
mu_theta <- 6*theta/5
sigma_theta <- theta*0.2
theta0 <- mu_theta
### Estimations par maximum de vraisemblance
res_MLE <- mle(alpha0, beta0, theta0, data, t_insp)

# 2. alpha ~ Log-normale et beta ~ Log-normale
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
```

```

plot1 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Gamma et beta ~ Lognormale
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot2 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Log-normale et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot3 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Uniforme")
)

# 4. alpha ~ Gamma et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot4 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Uniforme")
)

# Visualisation finale
ggarrange(plot1, plot2, plot3, plot4, common.legend = TRUE)

```

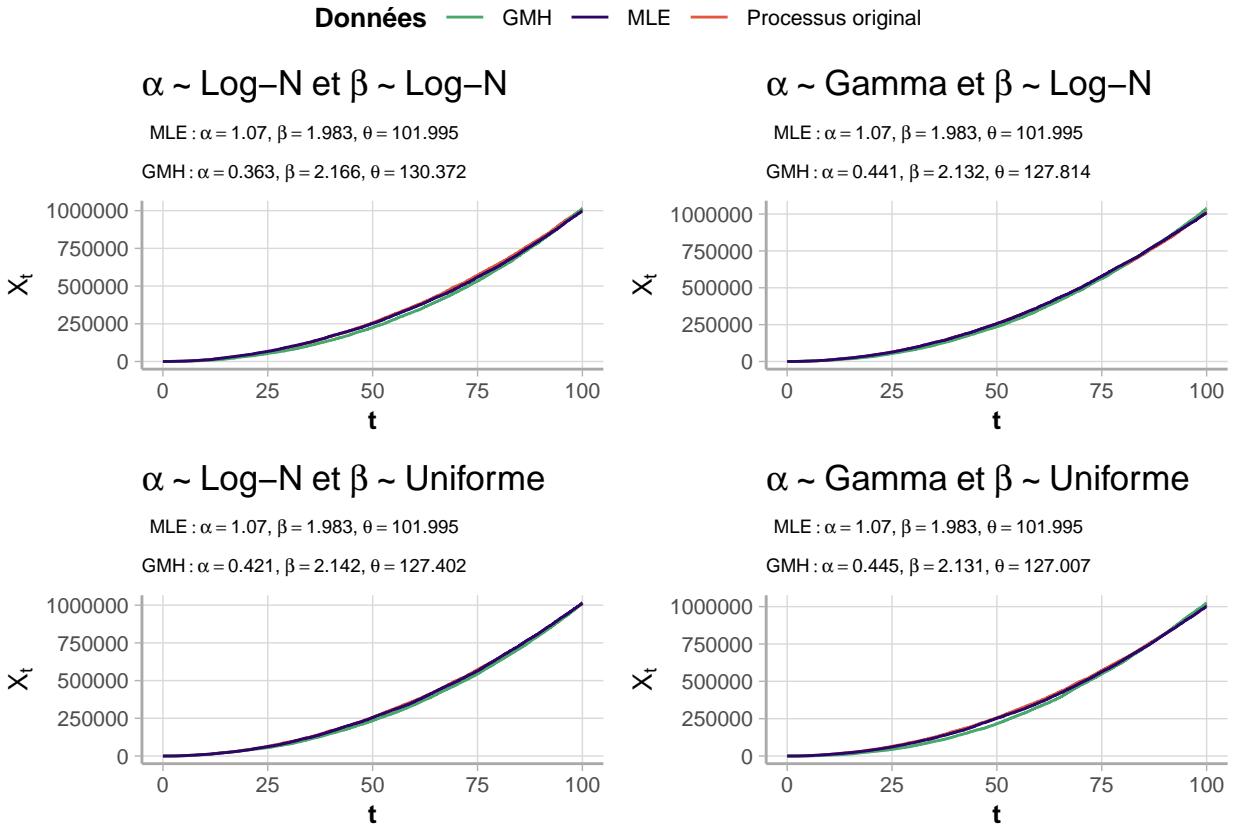


Figure 12: Résultats d'estimations bayésiennes et par maximum de vraisemblance lorsque $\theta = 100$, $\beta = 2$ et $\alpha = 1$

Nous obtenons toujours le même constat qu'avec $\theta = 10$. Par contre, pour faire le parallèle avec les variations de α , les deux algorithmes d'estimations (MLE ou GMH) semblent toujours vouloir augmenter la valeur de θ lorsque l'un ou l'autre augmente (12).

3.6 Variations des paramètres relatifs à l'expert

Nous nous intéressons maintenant à l'impact des paramètres d'expert sur la qualité d'estimation. Comme GMH semblait être le plus efficace avec $\beta = 1$, nous conservons cette valeur. De plus, nous refixons les paramètres dans le même cadre que celui utilisé lors des variations de β .

```
# Paramètres de simulation
dt_sim <- 0.01
dt_insp <- 5
alpha <- 1
beta <- 1
theta <- 2
# Paramètres de l'algorithme GMH
K <- 10000
burnin <- 1000
```

3.6.1 Expert bon et confiance élevée

```
# Seed pour la reproductibilité
set.seed(1)

# 1. Simulation du processus
sim_og <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = beta, theta = theta
)
```

```

data <- sim_og$X_insp
t_insp <- sim_og$grille_insp

### Expertise sur les params : bonne confiance en un expert bon
mu_theta <- 6*theta/5
sigma_theta <- theta*0.2
theta0 <- mu_theta

mu_alpha <- 6*alpha/5
sigma_alpha <- alpha*0.2
tau_alpha <- 0.3
alpha0 <- mu_alpha

mu_beta <- 6*beta/5
sigma_beta <- mu_beta*0.2
tau_beta <- 0.3
beta0 <- mu_beta
### Estimations par maximum de vraisemblance
res_MLE <- mle(alpha0, beta0, theta0, data, t_insp)

# 1. alpha ~ Log-normale et beta ~ Log-normale
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot1 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Log-N")
)

# 2. alpha ~ Gamma et beta ~ Lognormale
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot2 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Log-normale et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot3 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Uniforme")
)

# 4. alpha ~ Gamma et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)
plot4 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Uniforme")
)

```

```

)
# Visualisation finale
ggarrange(plot1, plot2, plot3, plot4, common.legend = TRUE)

```

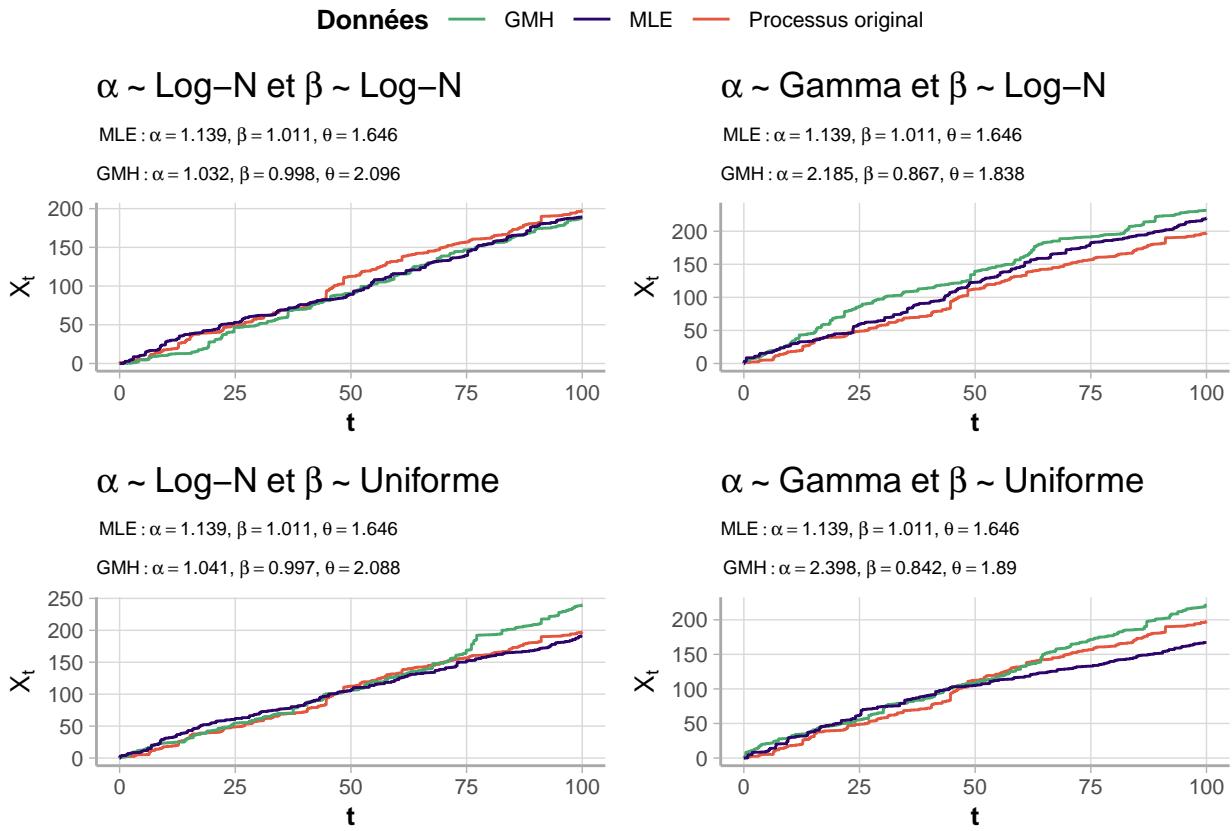


Figure 13: Résultats pour un bon expert à qui nous faisons confiance, avec $\alpha = 1$, $\beta = 1$ et $\theta = 2$

Nous avions déjà observé ce cas, les performances de GMH sont assez bonnes lorsque $\alpha \sim \mathcal{L}\text{og-}\mathcal{N}$ et $\beta \sim \mathcal{L}\text{og-}\mathcal{N}$ ou $\alpha \sim \mathcal{L}\text{og-}\mathcal{N}$ et $\beta \sim \mathcal{U}$. Par contre, lorsque $\alpha \sim \mathcal{G}$, GMH tend à sur-estimer α (13).

3.6.2 Expert mauvais et confiance élevée

```

# Seed pour la reproductibilité
set.seed(1)

# 1. Simulation du processus
sim_og <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = beta, theta = theta
)
data <- sim_og$X_insp
t_insp <- sim_og$grille_insp

### Expertise sur les params : bonne confiance en un expert mauvais
mu_theta <- 2*theta
sigma_theta <- theta*0.2
theta0 <- mu_theta

mu_alpha <- 2*alpha
sigma_alpha <- alpha*0.2
tau_alpha <- 0.3
alpha0 <- mu_alpha

```

```

mu_beta <- beta + beta
sigma_beta <- mu_beta*0.2
tau_beta <- 0.3
beta0 <- mu_beta
### Estimations par maximum de vraisemblance
res_MLE <- mle(alpha0, beta0, theta0, data, t_insp)

# 1. alpha ~ Log-normale et beta ~ Log-normale
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)

plot1 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Log-N")
)

# 2. alpha ~ Gamma et beta ~ Lognormale
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)

plot2 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Log-normale et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)

plot3 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Uniforme")
)

# 4. alpha ~ Gamma et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)

plot4 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Uniforme")
)

# Visualisation finale
ggarrange(plot1, plot2, plot3, plot4, common.legend = TRUE)

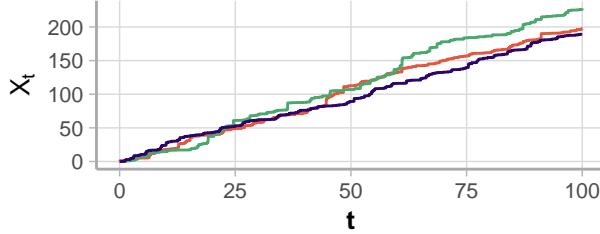
```

Données — GMH — MLE — Processus original

$\alpha \sim \text{Log-N}$ et $\beta \sim \text{Log-N}$

MLE : $\alpha = 1.139, \beta = 1.011, \theta = 1.646$

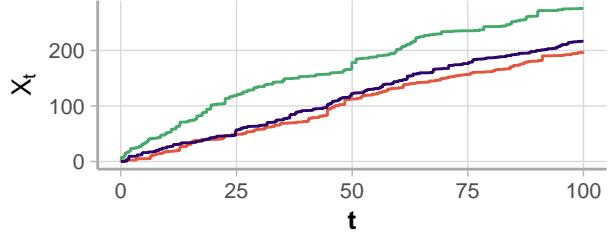
GMH : $\alpha = 1.456, \beta = 0.84, \theta = 3.309$



$\alpha \sim \text{Gamma}$ et $\beta \sim \text{Log-N}$

MLE : $\alpha = 1.139, \beta = 1.011, \theta = 1.646$

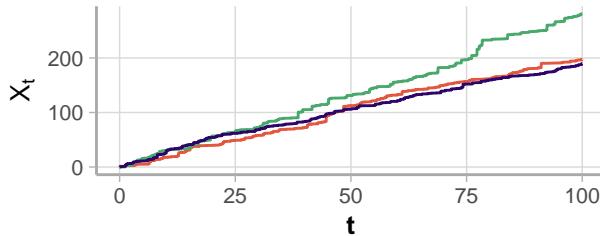
GMH : $\alpha = 3.904, \beta = 0.652, \theta = 3.078$



$\alpha \sim \text{Log-N}$ et $\beta \sim \text{Uniforme}$

MLE : $\alpha = 1.139, \beta = 1.011, \theta = 1.646$

GMH : $\alpha = 1.122, \beta = 0.925, \theta = 3.022$



$\alpha \sim \text{Gamma}$ et $\beta \sim \text{Uniforme}$

MLE : $\alpha = 1.139, \beta = 1.011, \theta = 1.646$

GMH : $\alpha = 1.364, \beta = 0.916, \theta = 2.731$

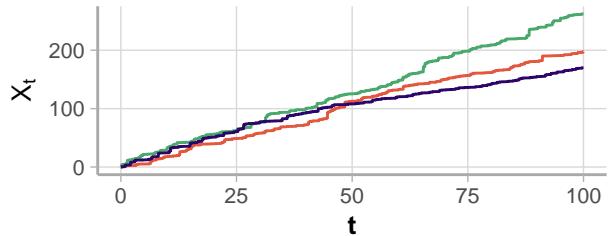


Figure 14: Résultats pour un mauvais expert à qui nous faisons confiance, avec $\alpha = 1, \beta = 1$ et $\theta = 2$

Bien qu'assez faible, nous observons une dégradation des estimations de GMH, notamment pour θ et α . Les valeurs initiales données par l'expert peuvent donc entraîner de grandes variations par rapport aux vraies valeurs (14).

3.6.3 Expert bon et faible confiance

```
# Seed pour la reproductibilité
set.seed(1)

# 1. Simulation du processus
sim_og <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = beta, theta = theta
)
data <- sim_og$X_insp
t_insp <- sim_og$grille_insp

### Expertise sur les params : bonne confiance en un expert bon
mu_theta <- 6*theta/5
sigma_theta <- theta*0.8
theta0 <- mu_theta

mu_alpha <- 6*alpha/5
sigma_alpha <- alpha*0.8
tau_alpha <- 0.3
alpha0 <- mu_alpha

mu_beta <- 6*beta/5
sigma_beta <- mu_beta*0.8
tau_beta <- 0.3
beta0 <- mu_beta

### Estimations par maximum de vraisemblance
res_MLE <- mle(alpha0, beta0, theta0, data, t_insp)
```

```

# 1. alpha ~ Log-normale et beta ~ Log-normale
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)

plot1 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Log-N")
)

# 2. alpha ~ Gamma et beta ~ Lognormale
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)

plot2 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Log-normale et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)

plot3 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Uniforme")
)

# 4. alpha ~ Gamma et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)

plot4 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Uniforme")
)

# Visualisation finale
ggarrange(plot1, plot2, plot3, plot4, common.legend = TRUE)

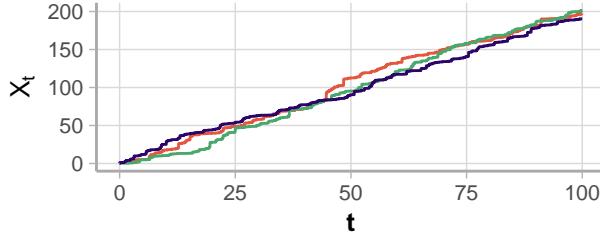
```

Données — GMH — MLE — Processus original

$\alpha \sim \text{Log-N}$ et $\beta \sim \text{Log-N}$

MLE : $\alpha = 1.139, \beta = 1.011, \theta = 1.646$

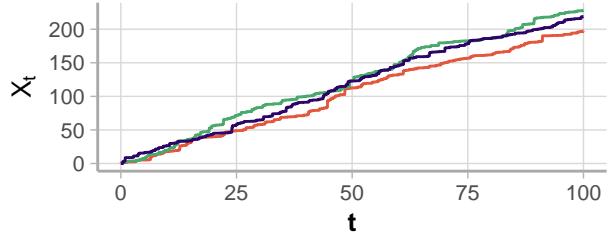
GMH : $\alpha = 1.048, \beta = 1.015, \theta = 1.967$



$\alpha \sim \text{Gamma}$ et $\beta \sim \text{Log-N}$

MLE : $\alpha = 1.139, \beta = 1.011, \theta = 1.646$

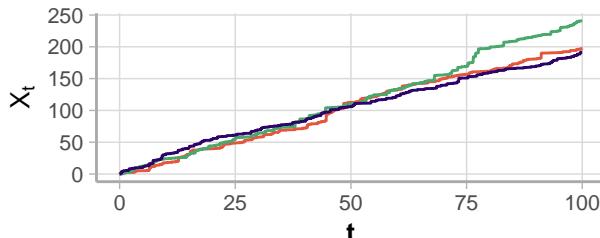
GMH : $\alpha = 1.207, \beta = 0.996, \theta = 1.849$



$\alpha \sim \text{Log-N}$ et $\beta \sim \text{Uniforme}$

MLE : $\alpha = 1.139, \beta = 1.011, \theta = 1.646$

GMH : $\alpha = 1.056, \beta = 1.012, \theta = 2.009$



$\alpha \sim \text{Gamma}$ et $\beta \sim \text{Uniforme}$

MLE : $\alpha = 1.139, \beta = 1.011, \theta = 1.646$

GMH : $\alpha = 1.352, \beta = 0.97, \theta = 1.897$

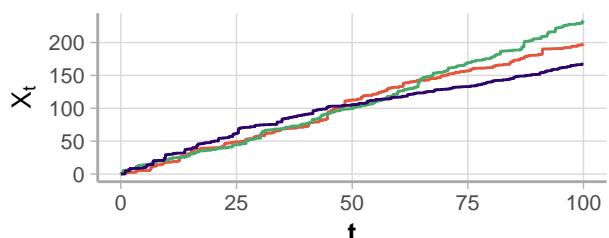


Figure 15: Résultats pour un bon expert à qui nous ne faisons pas confiance, avec $\alpha = 1, \beta = 1$ et $\theta = 2$

Bien que nous n'accordions pas une grande confiance à l'expert, le fait qu'il soit bon permet quand même d'obtenir de bonnes estimations des paramètres via GMH. L'impact de la confiance accordée semble donc moins important que la qualité de l'expert (15).

3.6.4 Expert mauvais et faible confiance

```
# Seed pour la reproductibilité
set.seed(1)

# 1. Simulation du processus
sim_og <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = beta, theta = theta
)
data <- sim_og$X_insp
t_insp <- sim_og$grille_insp

### Expertise sur les params : bonne confiance en un expert bon
mu_theta <- 2*theta
sigma_theta <- theta*0.8
theta0 <- mu_theta

mu_alpha <- 2*alpha
sigma_alpha <- alpha*0.8
tau_alpha <- 0.3
alpha0 <- mu_alpha

mu_beta <- 2*beta
sigma_beta <- mu_beta*0.8
tau_beta <- 0.3
beta0 <- mu_beta

### Estimations par maximum de vraisemblance
```

```

res_MLE <- mle(alpha0, beta0, theta0, data, t_insp)

# 1. alpha ~ Log-normale et beta ~ Log-normale
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)

plot1 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Log-N")
)

# 2. alpha ~ Gamma et beta ~ Lognormale
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_lognormale, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)

plot2 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Log-N")
)

# 3. alpha ~ Log-normale et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_lognormale, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)

plot3 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Log-N" ~ "et" ~ beta %~% "Uniforme")
)

# 4. alpha ~ Gamma et beta ~ Uniforme
res_GMH <- algo_GMH(
  logpost_alpha_gamma, mu_alpha, sigma_alpha, tau_alpha, alpha0,
  logpost_beta_unif, mu_beta, sigma_beta, tau_beta, beta0,
  mu_theta, sigma_theta, theta0, data, t_insp, K, burnin
)

plot4 <- vis_res(
  res_GMH, res_MLE, sim_og,
  title = expression(alpha %~% "Gamma" ~ "et" ~ beta %~% "Uniforme")
)

# Visualisation finale
ggarrange(plot1, plot2, plot3, plot4, common.legend = TRUE)

```

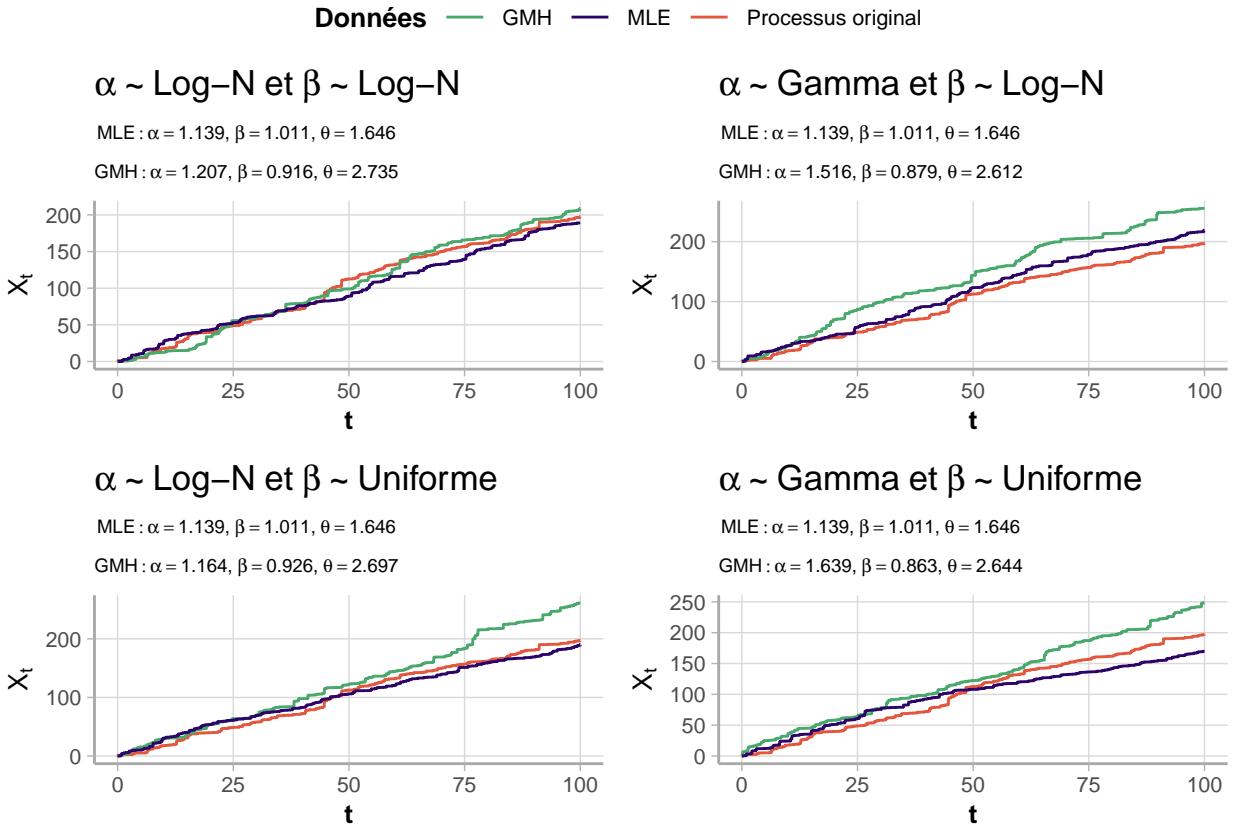


Figure 16: Résultats pour un mauvais expert à qui nous faisons confiance, avec $\alpha = 1, \beta = 1$ et $\theta = 2$

Une faible confiance permet de corriger en partie les mauvaises valeurs de l’expert, notamment quand nous comparons les résultats au cas où nous lui accordons une grande confiance. Toutefois, cela ne suffit pas pour corriger entièrement les écarts aux vraies valeurs (16).

4 Conclusion

Nous avons constaté dans cette analyse que des variations sur au moins un des trois paramètres du processus (α , β et θ) entraînent de nombreux changements dans les estimations via l’algorithme GMH. Ainsi, suivant les vraies valeurs choisies, la meilleure combinaison de lois a priori peut varier. Toutefois, la meilleure loi a priori pour α reste toujours une log-normale, le choix d’une loi a priori gamma ayant montré une baisse de la qualité d’estimation de manière quasiment systématique. En comparant aux estimations faites par maximum de vraisemblance, nous avons aussi constaté que l’estimation bayésienne n’était pas toujours la plus adaptée suivant les cas pour un temps de calcul similaire.

Nous pouvons également remarquer que ces estimations (MLE et GMH) semblent mieux adaptées pour retrouver la forme du processus plutôt que la valeur de ces paramètres.

Comme attendu au niveau des paramètres relatifs à l’expert, les estimations sont dégradées lorsque nous avons un mauvais expert, et le degré de confiance que nous lui accordons alors peut en partie corriger ses erreurs.

Pour aller plus loin, nous pourrions diversifier encore plus les cas d’expert, en ayant par exemple une bonne expertise pour β mais une mauvaise pour les deux autres paramètres et regarder l’impact sur l’estimation faite par l’algorithme GMH.

5 Utilisation de l’IA

Nous avons utilisé ChatGPT-5 principalement pour corriger les erreurs de nos fonctions implémentées, pour nous aider dans le choix des lois a priori et pour nous aider à obtenir les visuels voulus.