Projet statistiques bayésiennes

Garance MALNOË et Matthias MAZET

2026-01-28

Contents

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

1 Introduction

À partir d'un processus gamma non homogène simulé, nous voulons construire une inférence bayésienne permettant de retrouver les paramètres du dit processus. 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 posons aussi, pour tout $t > s \geq 0$, $X(t) - X(s) \sim \mathcal{G}(a(t) - a(s), \theta)$ et, pour tout $x \in \mathbb{R}_+$, $f_{a(t)-a(s),\theta}(x) = \dfrac{x^{a(t)-a(s)-1}}{\theta^{a(t)-a(s)}\Gamma(a(t)-a(s))} e^{-\frac{x}{\theta}}$.
L'inférence bayésienne va donc nous servir à retrouver 3 paramètres : 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 x phases : i) la simulation du processus selon différentes valeurs pour α, β et θ afin d'observer leur impact respectif, ii) la mise en place de l'inférence bayésienne et iii) l'étude de l'impact du pas d'inspection.
Afin de se fixer un cadre d'étude, nous restreignons les possibilités pour les lois théoriques de chaque paramètre :

- α et β suivront chacun soit une loi *lognormale*, soit une loi *Gamma*.
- θ suivra une loi *Inversegamma*.

2 Simulations

Afin de vérifier l'effet théorique de chaque paramètre sur le processus, nous regardons individuellement l'impact de chacun. Pour cela, nous définissons 2 fonctions qui vont nous permettre de réaliser les simulations tout au long de ce rapport :

- 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 (simulation et inspection) et du processus observé sur 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, subtitle = "") {
  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" = "#db2e0b"),
    ) +
    labs(x = "t", y = expression(X[t]), subtitle = subtitle) +
```

```
    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 le pas de simulation à 0.01, le pas d'inspection à 5, $\beta = 1$ et $\theta = 2$. Nous prenons ensuite $\alpha \in \{.1,\ 1,\ 10,\ 20\}$ 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, subtitle = paste(expression(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, subtitle = paste(expression(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, subtitle = paste(expression(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, subtitle = paste(expression(alpha), "=", list_alpha[4])
)

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

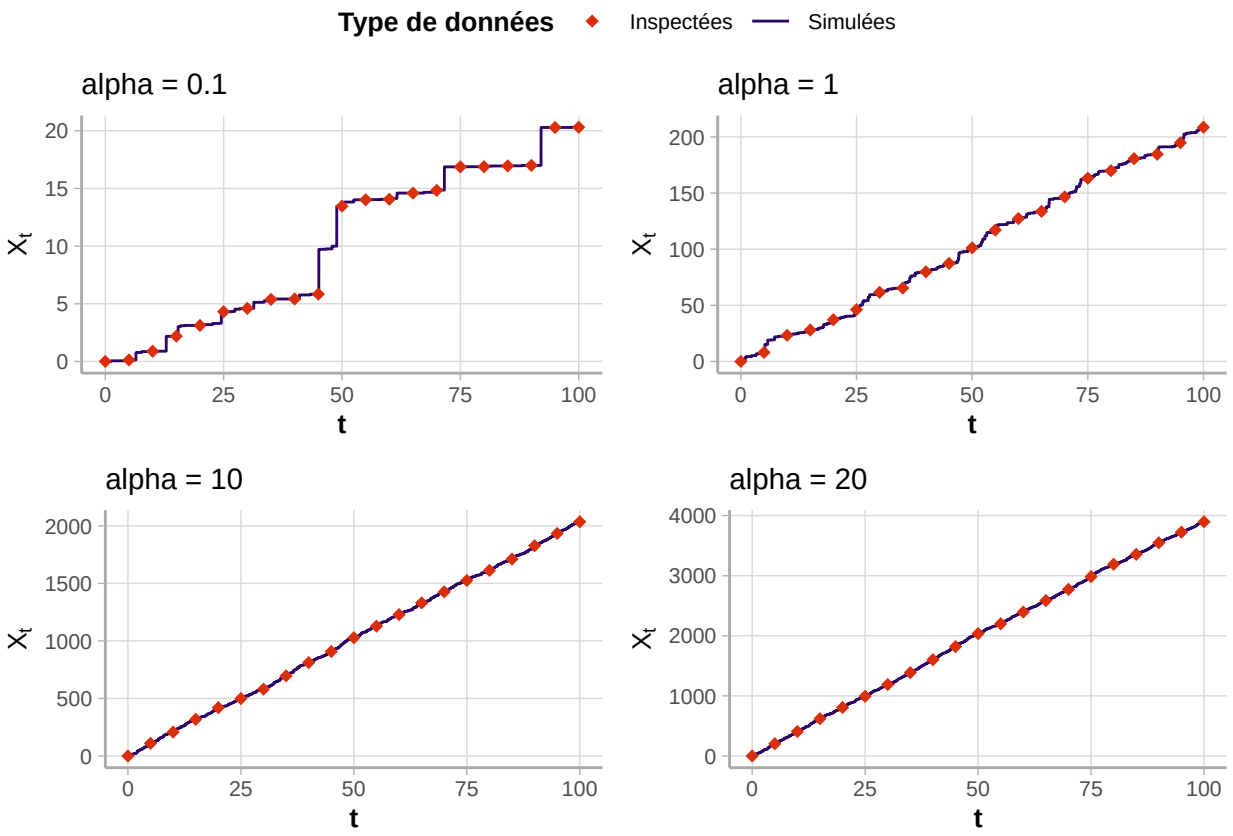

Figure 1: Simulations d'un processus gamma non-homogène ($\beta = 1$, $\theta = 2$) pour différents α, avec un pas de simulation de 0.01 et un pas d'inspection de 5

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 proportionnel (de l'ordre de 20 000) à α : plus il est grand, plus le processus se dégrade rapidement **(1)**.

Commentaire : Pas d'impact sur la forme de la courbe seulement sur les valeurs prises par Xt. Proportionnel à alpha : 0.1 -> 2000 1 -> 20.000 10 -> 200.000 20 -> 400.000 Faire le lien avec les maths / la théorie. Dégradations + rapide avec alpha + grand.

2.2 *Variations de* β

Observons maintenant l'effet de β. Comme précédemment, nous fixons les autres paramètres du modèle : nous fixons 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, subtitle = paste(expression(beta), "=", list_beta[1])
)
```

```r
sim2 <- simulations(
  dt_sim = dt_sim, dt_insp = dt_insp, alpha = alpha, beta = list_beta[2], theta = theta
)
plot_sim2 <- plot_simulations(
  sim2, subtitle = paste(expression(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, subtitle = paste(expression(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, subtitle = paste(expression(beta), "=", list_beta[4])
)

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

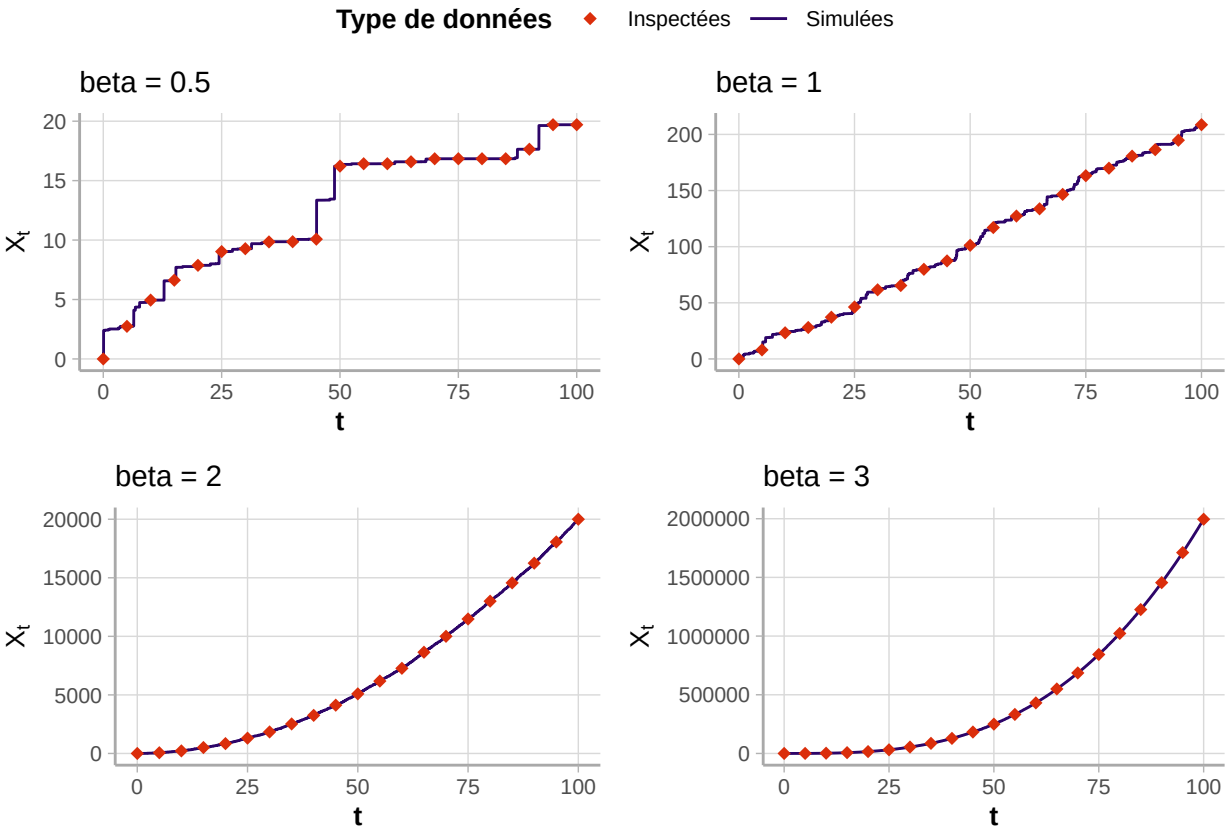

Figure 2: Simulations d'un processus gamma non-homogène ($\alpha = 1$, $\theta = 2$) pour différents β, avec un pas de simulation de 0.01 et un pas d'inspection de 5

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

- Pour un $\beta < 1$, la courbe et concave, i.e. la dégradation ralenti 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 concave, 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 **(2)**.

2.3 *Variations de* θ

Observons maintenant l'effet de θ. Comme précédemment, nous fixons les autres paramètres du modèle : nous fixons 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, subtitle = paste(expression(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, subtitle = paste(expression(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, subtitle = paste(expression(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, subtitle = paste(expression(theta), "=", list_theta[4])
)

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

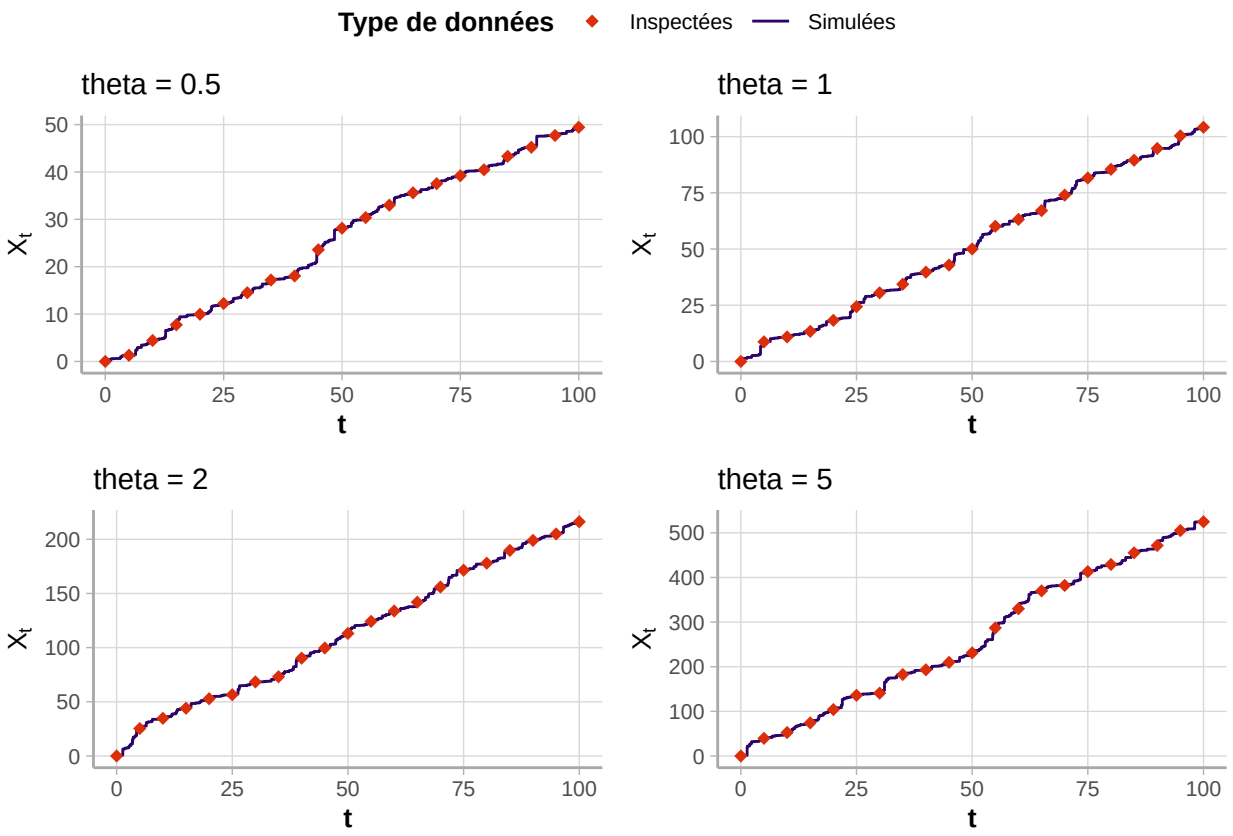

Figure 3: Simulations d'un processus gamma non-homogène ($\alpha = 1$, $\beta = 1$) pour différents θ, avec un pas de simulation de 0.01 et un pas d'inspection de 5

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 être proportionnels : plus θ est grand, plus le processus se dégrade rapidement. Toutefois, l'influence de θ semble moins fort que celle de α, le coefficient de proportionnalité étant ici de l'ordre de 100, contre 20 000 pour α **(3)**.

Commentaire : Pas d'impact sur la forme de la courbe seulement sur les valeurs prises par Xt. Proportionnel à theta : 0.5 -> 50 1 -> 100 2 -> 200 Dégradation + rapide avec theta + grand Faire le lien avec les maths / la théorie.

3 Etude de l'inférence bayésienne

3.1 Contexte, choix des lois a priori et calcul des lois a posteriori conditionnelles

Nous disposons d'un échantillon d'inspections du niveau de dégradation $x = \{x_0, \dots, x_n\}$ à différents temps d'inspection $t = \{t_0, \dots, t_n\}$ qui sont issus d'un processus gamma non-homogène de paramètres α, β et θ où les incréments sont positifs et indépendants. [Mettre le reste de l'énoncé].

Nous souhaitons réaliser une estimation bayésienne des trois paramètres α, β et θ, nous sommes donc dans le cas d'une estimation multidimensionnelle. Pour réaliser l'estimation, nous passons donc par les algorithmes de Gibbs et de Metropolis-Hasting en testant différentes combinaisons de loi a priori pour α, β et θ.

Pour α, nous testerons une $\mathcal{L}\wr\}\backslash\wr\nabla\Updownarrow\dashv\updownarrow\rceil(\mu,\sigma)$ et une $\mathcal{G}\dashv\Updownarrow\Updownarrow\dashv(k,s)$. Pour β, nous testerons une $\mathcal{L}\wr\}\backslash\wr\nabla\Updownarrow\dashv\updownarrow\rceil(\mu,\sigma)$ et une $\mathcal{U}\backslash\}\{\wr\nabla\Updownarrow\rceil(a,b)$. Pour θ, nous testerons une $\mathcal{I}\backslash\sqsubseteq\}\dashv\Updownarrow\Updownarrow\dashv(a,b)$. Toutes ces lois ont des supports positifs, ce qui convient bien pour l'estimation de α, β et $\theta > 0$.

Pour l'ensemble des lois, nous allons plutôt travailler avec les incréments $y_i = x_i - x_{i-1} = x(t_i) - x(t_{i-1})$, $i \in \{1, \dots, n\}$. Nous allons également poser $k_i(\alpha, \beta) = a(t_i) - a(t_{i-1}) = \alpha(t_i^\beta - t_{i-1}^\beta)$ afin d'alléger les calculs suivants.

Nous avons par hypothèse que $X(t_i) - X(t_{i-1}) = Y_i \sim Gamma(k_i(\alpha,\beta), \theta)$ 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 incréments sont supposés indépendants, la vraisemblance des données y sachant α, β et $\theta > 0$ est donc :

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

Dans cette analyse, nous comparerons les estimations bayésiennes avec les estimations obtenues par maximum de vraisemblance, ces dernières seront obtenues en réalisant une optimisation (maximisation) de la fonction de vraisemblance pour les trois paramètres.

3.1.1 Calcul des lois a posteriori pour θ

Nous choisissons comme loi a priori pour θ une $\mathcal{I}\backslash\sqsubseteq\}\dashv\Updownarrow\Updownarrow\dashv(a,b)$, ainsi :

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

Nous pouvons reconnaitre, à une constante de normalisation prêt, la densité d'une $\mathcal{I}\backslash\sqsubseteq\}\dashv\Updownarrow\Updownarrow\dashv(a+\sum k_i(\alpha,\beta),\ b+\sum y_i)$. Nous pourrons estimer θ avec un algorithme de *Gibbs*. Nous supposons également que l'expert ne nous donnerai pas directement a et b pour l'Invgamma mais plutôt une estimation du paramètre θ : μ_θ associé à une certaine confiance : σ_θ. En supposant que $\mu_\theta = \mathbb{E}[\mathcal{I}\backslash\sqsubseteq\}\dashv\Updownarrow\Updownarrow\dashv(a,b)] = \frac{b}{a-1}$ et $Var(\mathcal{I}\backslash\sqsubseteq\}\dashv\Updownarrow\Updownarrow\dashv(a,b)) = \sigma_\theta = \frac{b^2}{(a-1)^2(a-2)}$ on a alors : $a = \frac{\mu_\theta^2}{\sigma_\theta} + 2$ et $b = (a-1)\mu_\theta$. [Je peux détailler les calculs si besoin mais bon, c'est pas super central ici...]

3.1.2 Calcul des lois a posteriori pour α

1. $\mathcal{L}\wr\}\mathcal{N}\wr\nabla\Updownarrow\dashv\updownarrow\rceil(\mu,\sigma)$

$$p(\alpha|y,\beta,\theta) = L(y|\alpha,\beta,\theta)\times p(\alpha)p(\alpha|y,\beta,\theta) \propto \frac{1}{\alpha\sigma\sqrt{2\pi}} exp(-\frac{(ln(\alpha)-\mu)^2}{2\sigma^2}) \prod \frac{1}{\Gamma(k_i(\alpha,\beta))\theta^{k_i(\alpha,\beta)}} y_i^{k_i(\alpha,\beta)}$$

Nous ne reconnaissons pas de forme de loi usuelle, nous allons donc utiliser l'algorithme de Metropolis-Hasting. Pour faciliter le calcul numérique, nous plutôt que de regarder le rapport $\frac{p(\tilde{\alpha}|y,\beta,\theta)}{p(\alpha^{(k-1)}|y,\beta,\theta)}$ nous allons regarder $log(p(\tilde{\alpha}|y,\beta,\theta)) - log(p(\alpha^{(k-1)}|y,\beta,\theta))$.

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

Nous supposons également que l'expert ne nous donnerai pas directement μ et σ pour la lognormale mais plutôt une estimation du paramètre α : μ_α associé à une certaine confiance : σ_α. En supposant que $\mu_\alpha = \mathbb{E}[\mathcal{L}\wr\}\mathcal{N}\wr\nabla\Updownarrow\dashv\updownarrow\rceil(\mu,\sigma)] = e^{\mu+\sigma^2/2}$ et $Var(\mathcal{L}\wr\}\mathcal{N}\wr\nabla\Updownarrow\dashv\updownarrow\rceil(\mu,\sigma)) = \sigma_\alpha = (e^{\sigma^2}-1)e^{2\mu+\sigma^2}$ on a alors : $\mu = ln(\mu_\alpha) - \frac{ln(1+\frac{\sigma_\alpha}{\mu_\alpha^2})}{2}$ et $\sigma = \sqrt{ln(\frac{\sigma_\alpha}{\mu_\alpha^2}+1)}$.

2. $\mathcal{G}\dashv\Updownarrow\Updownarrow\dashv(k,s)$

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

Nous ne reconnaissons pas de forme de loi usuelle, nous allons donc utiliser l'algorithme de Metropolis-Hasting. Pour faciliter le calcul numérique, nous plutôt que de regarder le rapport $\frac{p(\tilde{\alpha}|y,\beta,\theta)}{p(\alpha^{(k-1)}|y,\beta,\theta)}$ nous allons regarder $log(p(\tilde{\alpha}|y,\beta,\theta)) - log(p(\alpha^{(k-1)}|y,\beta,\theta))$.

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

Nous supposons également que l'expert ne nous donnerai pas directement k et s pour la gamma mais plutôt une estimation du paramètre α : μ_α associé à une certaine confiance : σ_α. En supposant que $\mu_\alpha = \mathbb{E}[\mathcal{G}\dashv\Updownarrow\Updownarrow\dashv(k,s)] = \frac{k}{s}$ et $Var(\mathcal{G}\dashv\Updownarrow\Updownarrow\dashv(k,s)) = \sigma_\alpha = \frac{k}{s^2}$ on a alors : $k = \frac{\mu_\alpha^2}{\sigma_\alpha}$ et $s = \frac{\sigma_\alpha}{\mu_\alpha}$.

3.1.3 Calcul des lois a posteriori pour β

1. $\mathcal{L}\wr\}\mathcal{N}\wr\nabla\Updownarrow\dashv\updownarrow\rceil(\mu,\sigma)$

$$p(\beta|y,\alpha,\theta) = L(y|\beta,\alpha,\theta) \times p(\beta)p(\beta|y,\alpha,\theta) \propto \frac{1}{\beta\sigma\sqrt{2\pi}} exp(-\frac{(ln(\beta)-\mu)^2}{2\sigma^2}) \prod \frac{1}{\Gamma(k_i(\alpha,\beta))\theta^{k_i(\alpha,\beta)}} y_i^{k_i(\alpha,\beta)}$$

Nous ne reconnaissons pas de forme de loi usuelle, nous allons donc utiliser l'algorithme de Metropolis-Hasting. Pour faciliter le calcul numérique, nous plutôt que de regarder le rapport $\frac{p(\tilde{\beta}|y,\alpha,\theta)}{p(\beta^{(k-1)}|y,\alpha,\theta)}$ nous allons regarder $log(p(\tilde{\beta}|y,\alpha,\theta)) - log(p(\beta^{(k-1)}|y,\alpha,\theta))$.

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

Nous supposons également que l'expert ne nous donnerai pas directement μ et σ pour la lognormale mais plutôt une estimation du paramètre β : μ_β associé à une certaine confiance : σ_β. En supposant que $\mu_\beta = \mathbb{E}[\mathcal{L}\wr\}\mathcal{N}\wr\nabla\Updownarrow\dashv\updownarrow\rceil(\mu,\sigma)] = e^{\mu+\sigma^2/2}$ et $Var(\mathcal{L}\wr\}\mathcal{N}\wr\nabla\Updownarrow\dashv\updownarrow\rceil(\mu,\sigma)) = \sigma_\beta = (e^{\sigma^2}-1)e^{2\mu+\sigma^2}$ on a alors : $\mu = ln(\mu_\beta) - \frac{ln(1+\frac{\sigma_\beta}{\mu_\beta^2})}{2}$ et $\sigma = \sqrt{ln(\frac{\sigma_\beta}{\mu_\beta^2}+1)}$.

2. $\mathcal{U}\backslash\}\{\wr\nabla\Updownarrow\rceil(a,b)$

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

Nous ne reconnaissons pas de forme de loi usuelle, nous allons donc utiliser l'algorithme de Metropolis-Hasting. Pour faciliter le calcul numérique, nous plutôt que de regarder le rapport $\frac{p(\tilde{\beta}|y,\alpha,\theta)}{p(\beta^{(k-1)}|y,\alpha,\theta)}$ nous allons regarder $log(p(\tilde{\beta}|y,\alpha,\theta)) - log(p(\beta^{(k-1)}|y,\alpha,\theta))$.

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

Nous supposons également que l'expert ne nous donnerai pas directement a et b pour la uniforme mais plutôt une estimation du paramètre β : μ_β associé à une certaine confiance : σ_β. En supposant que $\mu_\beta = \mathbb{E}[\mathcal{U}\backslash\}\{\wr\nabla\Updownarrow\rceil(a,b)] = \frac{a+b}{2}$ et $Var(\mathcal{U}\backslash\}\{\wr\nabla\Updownarrow\rceil(a,b)) = \sigma_\beta = \frac{(a-b)^2}{12}$ on a alors : $a = \mu_\beta - \sqrt{3\sigma_\beta}$ et $b = \mu_\beta + \sqrt{3\sigma_\beta}$.

3.1.4 Remarque

Pour α et pour β, nous faisons le choix dans l'algorithme de Metropolis-Hasting de choisir comme distribution instrumentale une loi normale centrée en $\alpha^{(k-1)}$ ou $\beta^{(k-1)}$ afin de se ramener à un rapport des lois a posteriori entre la nouvelle valeur $\tilde{\alpha}$ ou $\tilde{\beta}$ et l'ancienne valeur $\alpha^{(k-1)}$ ou $\beta^{(k-1)}$, avec τ_α et τ_β comme variance pour chacune des normales.

3.2 Fonctions

```
library(invgamma)
```

```
## Warning: le package 'invgamma' a été compilé avec la version R 4.4.3
```

3.2.1 Fonctions utiles, calcul du log des loi a posteriori

```
# Fonction de calcul des ki
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)
}


# Fonctions a posteriori pour alpha
## Lognormale(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))
}

# Fonctions a posteriori pour beta
## Lognormale(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) return(-Inf)
  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 Fonction algorithme de Gibbs-Metropolis-Hastings

[Mettre pseudo code de la fonction] avec explication “mathématiques”.

```
algo_Gibbs_Metropolis_Hastings <- 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){

  # logpost_alpha <- fonction pour le calcul de log(posteriori(alpha~))
  # mu_alpha <- moyenne de l'expert
  # sigma_alpha <- confiance en l'expert
  # tau_alpha <- variation de la normale
  # alpha0 <- initialisation de alpha pour chaine de markov

  # logpost_beta <- fonction pour le calcul de log(posteriori(beta~))
  # mu_beta <- moyenne de l'expert
  # sigma_beta <- confiance en l'expert
  # tau_beta <- variation de la normale
  # beta0 <- initialisation de beta pour chaine de markov
```

```r
# mu_theta <- moyenne de l'expert
# sigma_theta <- confiance expert
# theta0 <- initialisation de theta pour la chaine de markov

# data <- x1,...,xn observations (à transformer en y1,...,yn) les incréments
# t_insp <- t1,...,tn les temps d'inspection
# K <- nombre d'itérations des algorithmes
# burnin <- nombre d'itérations à supprimer au début

# Transformation des données en incréments 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 de l'inverse gamma pour simuler theta à partir de esp_theta et de var_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 in 2,...,K+1 avec calcul de alpha^(k),beta^(k) et theta^(k)
for(k in 2:(K+1)){
  # 1. Simulation de theta^(k)
    # Calcul des k_i = alpha(ti^beta - ti-1^beta) avec alpha=alpha^(k-1) et beta=beta^(k-1)
  list_ki <- res_alpha[k-1]*(t_insp[2:(n+1)]^res_beta[k-1]-t_insp[1:n]^res_beta[k-1])
    # Calcul de a_bis = a + somme ki
  a_bis <- a_theta+sum(list_ki)
    # Calcul de b_bis = b + somme yi
  b_bis <- b_theta+sum(y)
  res_theta[k]  <- rinvgamma(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_alp
    second_log <- logpost_alpha(res_alpha[k-1], y, t_insp, res_beta[k-1], res_theta[k], mu_alpha, sigma_
    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])
  }
```

```
  }

  # Résultat 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 Fonction calcul des estimateurs de vraisemblance par optimisation

```
# - vraisemblance à minimiser pour trouver les MLE.
negative_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) return(1e100)
  if(any(!is.finite(y)) || any(y <= 0)) return(1e100)

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

}

# Fonction calculant les 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  = negative_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)
  )
}
```

3.3 Résultats

```
set.seed(1)
# Simulations 1
dt_sim <- 0.01
dt_insp <- 5
alpha <- 1
beta <- 2
theta <- 2
simulations1 <- simulations(dt_sim=dt_sim,dt_insp=dt_insp,alpha=alpha,beta=beta,theta=theta)
```

```
logpost_alpha <- logpost_alpha_gamma
mu_alpha <- alpha+0.01
sigma_alpha <- 1
tau_alpha <- 0.1
alpha0 <- mu_alpha
```

```r
logpost_beta <- logpost_beta_lognormale
mu_beta <- beta+0.01
sigma_beta <- 1
tau_beta <- 0.1
beta0 <- mu_beta

mu_theta <- theta+0.01
sigma_theta <- 1
theta0 <- mu_theta

data <- simulations1$X_insp
t_insp <- simulations1$t_insp
K <- 10000
burnin <- 100

res_GMH <- algo_Gibbs_Metropolis_Hastings(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)

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

Visualisations : Fixer un alpha, beta et theta. Pour chaque coombinaison de lois a priori Faire grille avec chaque cas de figure (expert bon ou mauvais, confiance faible ou élevée). (ggarance 2 cols 2 lignes) Visualisation avec titre commun les vrais paramètres et lois a priori choisies, titre perso avec cas de figure expert et simulations avec vrais paramètres et simulations avec paramètres estimés bayes et simulations avec MLE.

Commentaire : Il est important de vérifier la convergence de GMH et de MLE (GMH en faisant le plot de res_param et voir si bcp oscillations ou pas et pour MLE avec converged).

4 Conclusion