

Devoir Méthodes bayésiennes : session 1

Benoit Gachet, Guillaume Mulier

21/12/2020

Table of Contents

I.	Les données	2
II.	Modèle 1	2
1.	<i>Question 1</i>	2
2.	<i>Question 2</i>	2
3.	<i>Question 3</i>	14
4.	<i>Question 4</i>	14
5.	<i>Question 5</i>	18
6.	<i>Question 6</i>	18
III.	Modèle 2.....	20
7.	<i>Question 1</i>	21
8.	<i>Question 2</i>	21
9.	<i>Question 3</i>	32
10.	<i>Question 4</i>	32
11.	<i>Question 5</i>	33
12.	<i>Question 6</i>	37
13.	<i>Question 7</i>	38
14.	<i>Question 8</i>	38
15.	<i>Question 9</i>	38
IV.	Modèle 3.....	39
16.	<i>Question 1</i>	39
17.	<i>Question 2</i>	39
18.	<i>Question 3</i>	53
19.	<i>Question 4</i>	53
20.	<i>Question 5</i>	56
21.	<i>Question 6</i>	56

I. Les données

```
sncf_machines <- tibble(  
  machine = 1:10,  
  anciennete = c(2, 14, 2, 9, 15, 7, 3, 14, 5, 2),  
  nb_pannes = c(3, 50, 7, 20, 44, 3, 1, 58, 8, 7)  
)
```

II. Modèle 1

Le modèle est le suivant :

$$y_i \sim \text{Pois}(\lambda)$$

avec $\begin{cases} y_i \text{ le nombre de pannes de la machine } i \\ \log(\lambda) = a \end{cases}$

On a $\log(\lambda) = a \Leftrightarrow \lambda = e^a$.

1. Question 1

Donner $E(y_i|a)$ d'après ce modèle en fonction de a.

$$\begin{aligned} E(y_i|a) &= E(\text{Pois}(\lambda)) \\ &= \lambda \\ &= e^a \end{aligned}$$

2. Question 2

Mettre en place ce modèle avec, comme loi a priori sur a, une loi normale d'espérance nulle et de variance 1000. Faire 30000 itérations et enlever 1000 itérations pour le temps de chauffe. D'après l'history et les autocorrélations, voyez-vous un problème de mélangeance de l'algorithme ? Si oui, résoudre ce problème en justifiant.

Réalisation du modèle avec JAGS en prenant 30000 itérations avec 1000 itérations de burn-in au début, en gardant toutes les itérations :

```
# Données à présenter sous forme d'une liste  
donnees <- as.list(sncf_machines)  
# Modèle dans Langage BUGS et pas en Langage R  
modele_1 <- function() {  
  # Modèle pour yi  
  for (i in 1:10) {  
    nb_pannes[i] ~ dpois(exp(a))  
  }  
  # Loi a priori de a  
  a ~ dnorm(0, 0.001)  
}  
# Paramètres à recueillir  
parametres_modele1 <- c("a")  
# Valeurs initiales
```

```

inits_1 <- list("a" = 0)
inits_modele1 <- list(inits_1)
n_iter <- 30000 # Nombre d'iterations
n_burn <- 1000 # Burn in

# Faire tourner le modèle avec la fonction jags
# D'abord sans thin
set.seed(1993)
modele1_fit <- jags(data = donnees,
                      inits = inits_modele1,
                      parameters.to.save = parametres_modele1,
                      n.chains = length(inits_modele1),
                      n.iter = n_iter,
                      n.burnin = n_burn,
                      n.thin = 1,
                      model.file = modele_1)
modele1_fit_mcmc <- as.mcmc(modele1_fit)
gg_modele1 <- ggs(modele1_fit_mcmc)
ess_1 <- effectiveSize(modele1_fit_mcmc)

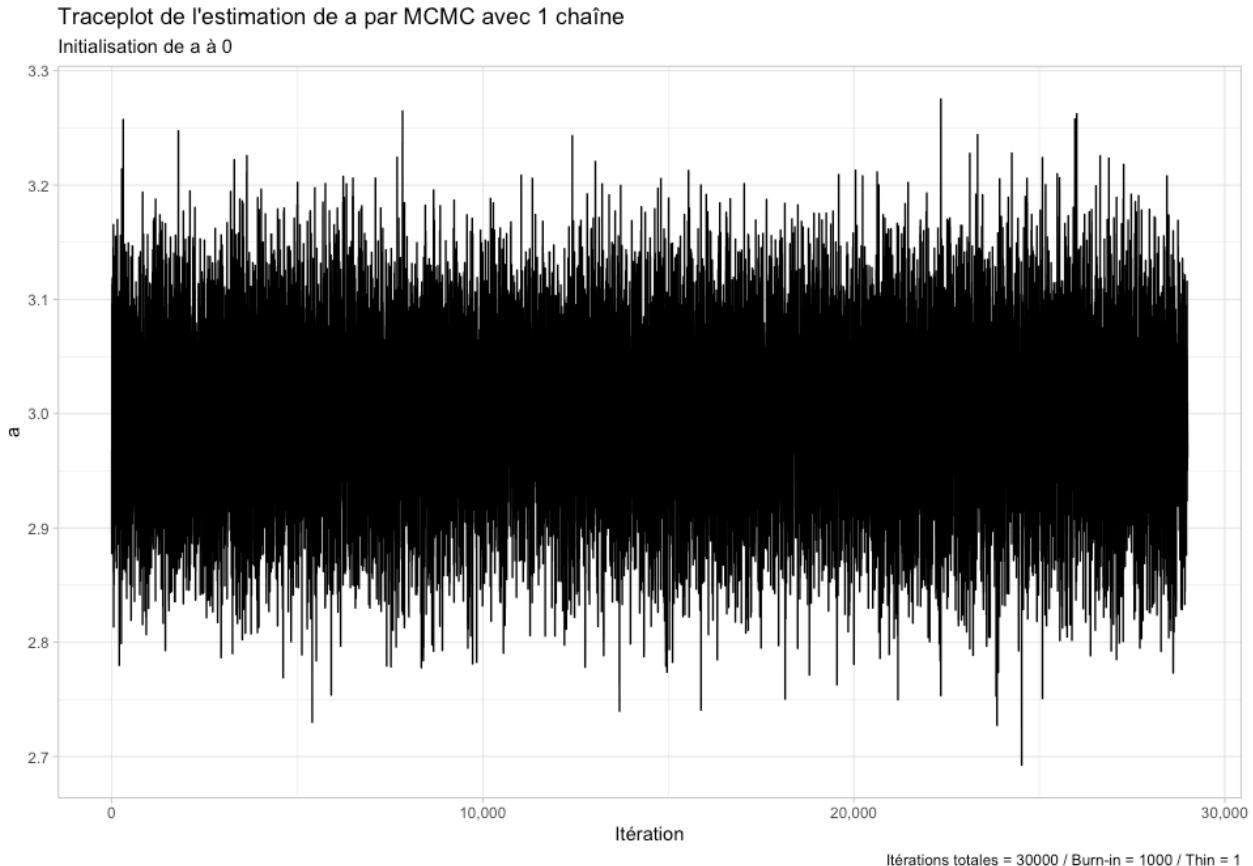
```

On regarde si le paramètre a estimé a bien convergé.

```

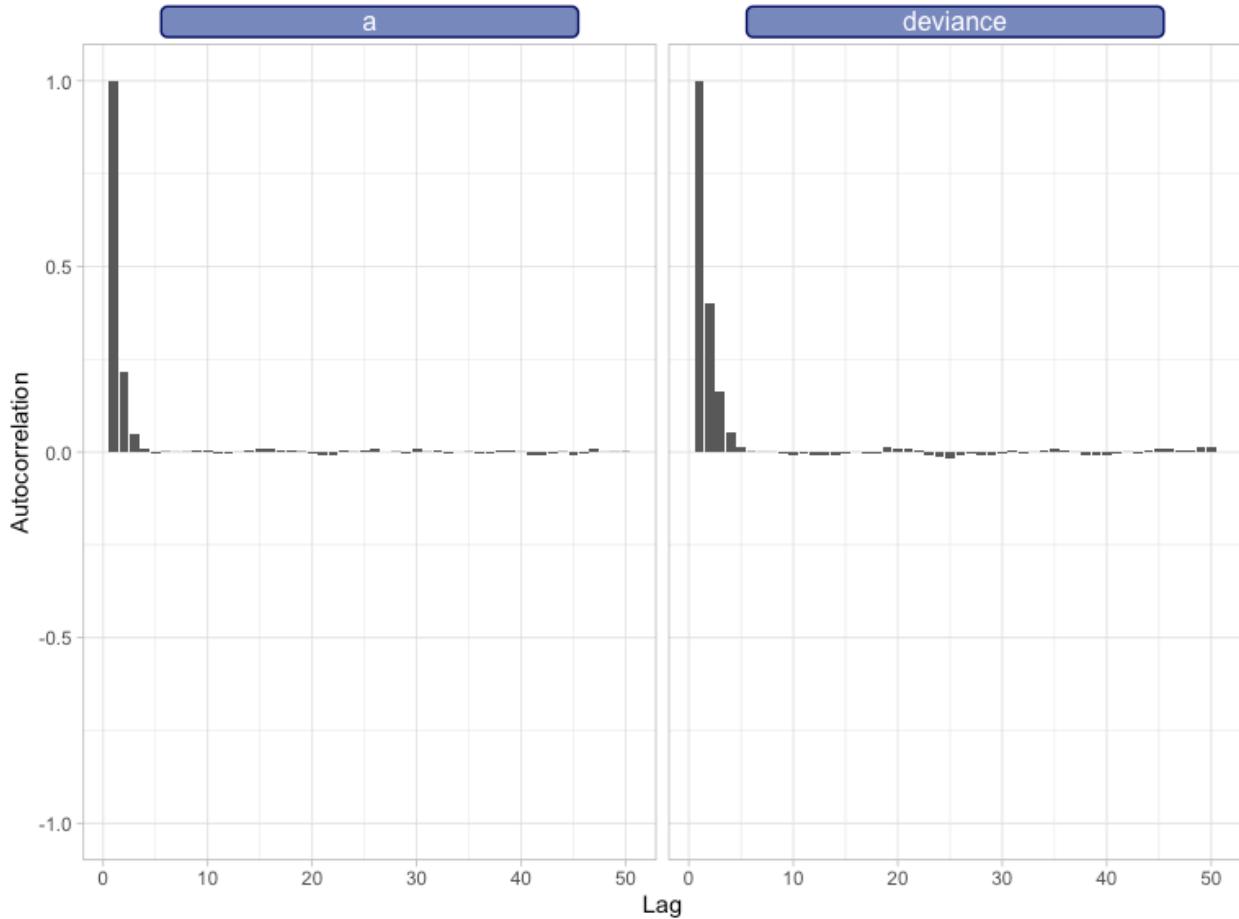
ggplot(gg_modele1 %>% filter(Parameter == "a"), aes(x = Iteration, y =
value)) +
  geom_line() +
  scale_x_continuous(labels = scales::comma_format()) +
  labs(x = "Itération",
       y = "a",
       title = "Traceplot de l'estimation de a par MCMC avec 1 chaîne",
       subtitle = "Initialisation de a à 0",
       caption = "Itérations totales = 30000 / Burn-in = 1000 / Thin = 1")

```



On voit que la valeur du paramètre a reste autour de la valeur 3 et n'a pas l'air de s'écarte beaucoup de cette valeur. De plus, le burn-in de 1000 semble suffisant car le début de la chaîne après la période de burn-in est aussi proche de 3. Nous vérifierons par la suite si cette convergence est conservée en augmentant le nombre de chaînes. Nous allons ensuite vérifier si les maillons de la chaîne sont bien indépendants les uns des autres.

```
ggs_autocorrelation(gg_modele1)
```



On voit que pour l'estimation de a , il y a corrélation jusqu'à la 3ème mesure. Pour la déviance, en revanche, cela va jusqu'à 5. Cela est confirmé par le nombre d'itérations effectives qui est franchement diminué par rapport aux 29000 itérations faites : 18628 pour l'estimation de a et 12743 pour l'estimation de la déviance du modèle.

Nous allons donc prendre un décallage de 8 pour essayer de casser cette autocorrélation. De plus, nous conserverons un burn-in de 1000 observations pour éviter de prendre des itérations qui n'ont pas encore convergé. Aussi, afin de conserver le nombre d'itérations conservées, nous multiplierons aussi par 8 le nombre total d'itération avant sélection des 1/8.

Il est à noter que par curiosité nous avons essayé de retrouver l'initialisation de la valeur de a en la faisant varier sans burn-in, mais le traceplot commençait toujours aux alentours de 3 même si nous initialisions a à 30 par exemple. Nous n'avons pas trouvé d'explication à cela.

```
n_thin <- 8
set.seed(1993)
modele1_fit_thin <- jags(data = donnees,
                           inits = inits_modele1,
                           parameters.to.save = parametres_modele1,
```

```

n.chains = length(inits_modele1),
n.iter = n_iter * n_thin,
n.burnin = n_burn,
n.thin = n_thin,
model.file = modele_1)
modele1_fit_thin_mcmc <- as.mcmc(modele1_fit_thin)
gg_modele1_thin <- ggs(modele1_fit_thin_mcmc)
ess_1_thin <- effectiveSize(modele1_fit_thin_mcmc)

```

Le modèle obtenu est le suivant :

```

tidy(modele1_fit_thin) %>%
  mutate(across(estimate:std.error, ~ round(.x, 3))) %>%
  flextable() %>%
  set_header_labels(term = "Paramètre estimé",
                    estimate = "Estimation",
                    std.error = "Ecart-Type") %>%
  autofit()

```

Paramètre estimé	Estimation	Ecart-Type
a	2.998	0.071

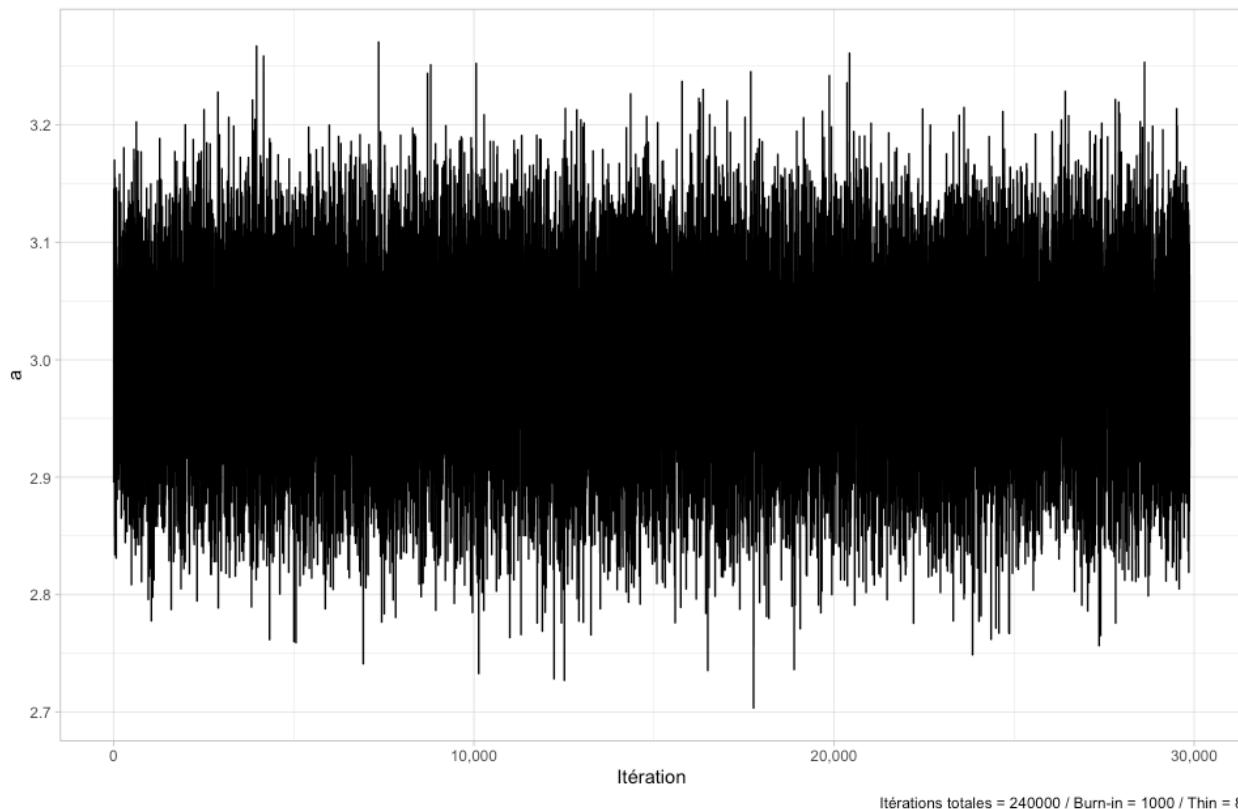
```

ggplot(gg_modele1_thin %>% filter(Parameter == "a"), aes(x = Iteration, y =
value)) +
  geom_line() +
  scale_x_continuous(labels = scales::comma_format()) +
  labs(x = "Itération",
       y = "a",
       title = "Traceplot de l'estimation de a par MCMC avec 1 chaîne",
       subtitle = "Initialisation de a à 0",
       caption = "Itérations totales = 240000 / Burn-in = 1000 / Thin = 8")

```

Traceplot de l'estimation de a par MCMC avec 1 chaîne

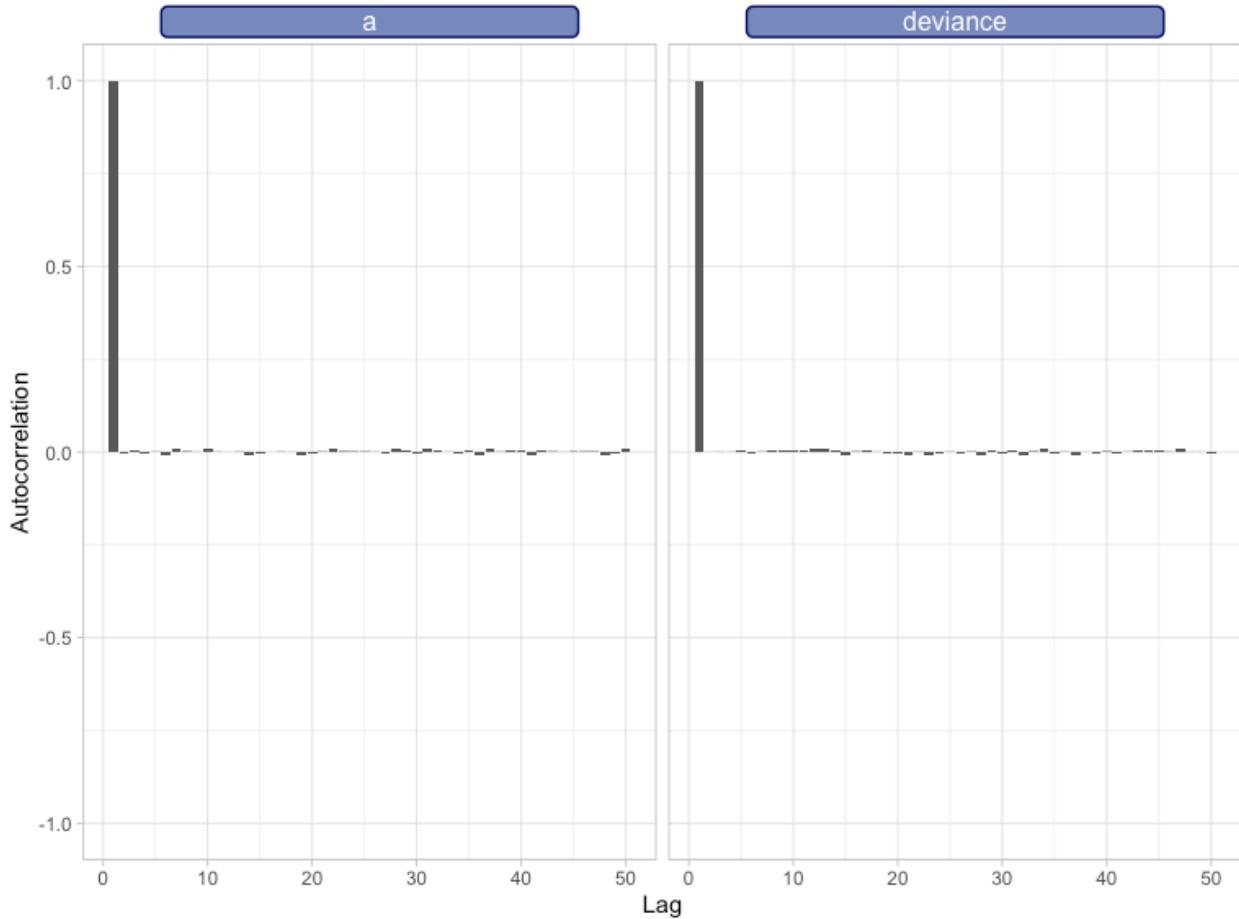
Initialisation de a à 0



Itérations totales = 240000 / Burn-in = 1000 / Thin = 8

En ne prenant qu'une observation sur 8, on voit que le traceplot reste similaire avec une bonne convergence de l'estimation de a autour de 3 après le retrait de 1000 observations de burn in.

```
ggs_autocorrelation(gg_modele1_thin)
```



Sur le graphique d'auto-corrélations, on voit que le problème a été réglé et que maintenant il n'y a plus d'auto-corrélation pour le paramètre a estimé. De plus, le nombre d'itérations effectives est maintenant de 29875 itérations : 29875 pour l'estimation de a et 29875 pour l'estimation de la déviance du modèle.

Afin de nous assurer de la convergence du modèle, nous avons réalisé 3 chaînes avec des départ pour des valeurs différentes. Nous avons initié a à -5, 0 et 5 et regardé comment se comportait le modèle.

```

inits_2 <- list("a" = 5)
inits_3 <- list("a" = -5)
inits_modele1_mult <- list(inits_1, inits_2, inits_3)
set.seed(1993)
modele1_fit_mult <- jags(data = donnees,
                           inits = inits_modele1_mult,
                           parameters.to.save = parametres_modele1,
                           n.chains = length(inits_modele1_mult),
                           n.iter = n_iter * n_thin,
                           n.burnin = n_burn,
                           n.thin = n_thin,
                           model.file = modele_1)
modele1_fit_mult_mcmc <- as.mcmc(modele1_fit_mult)

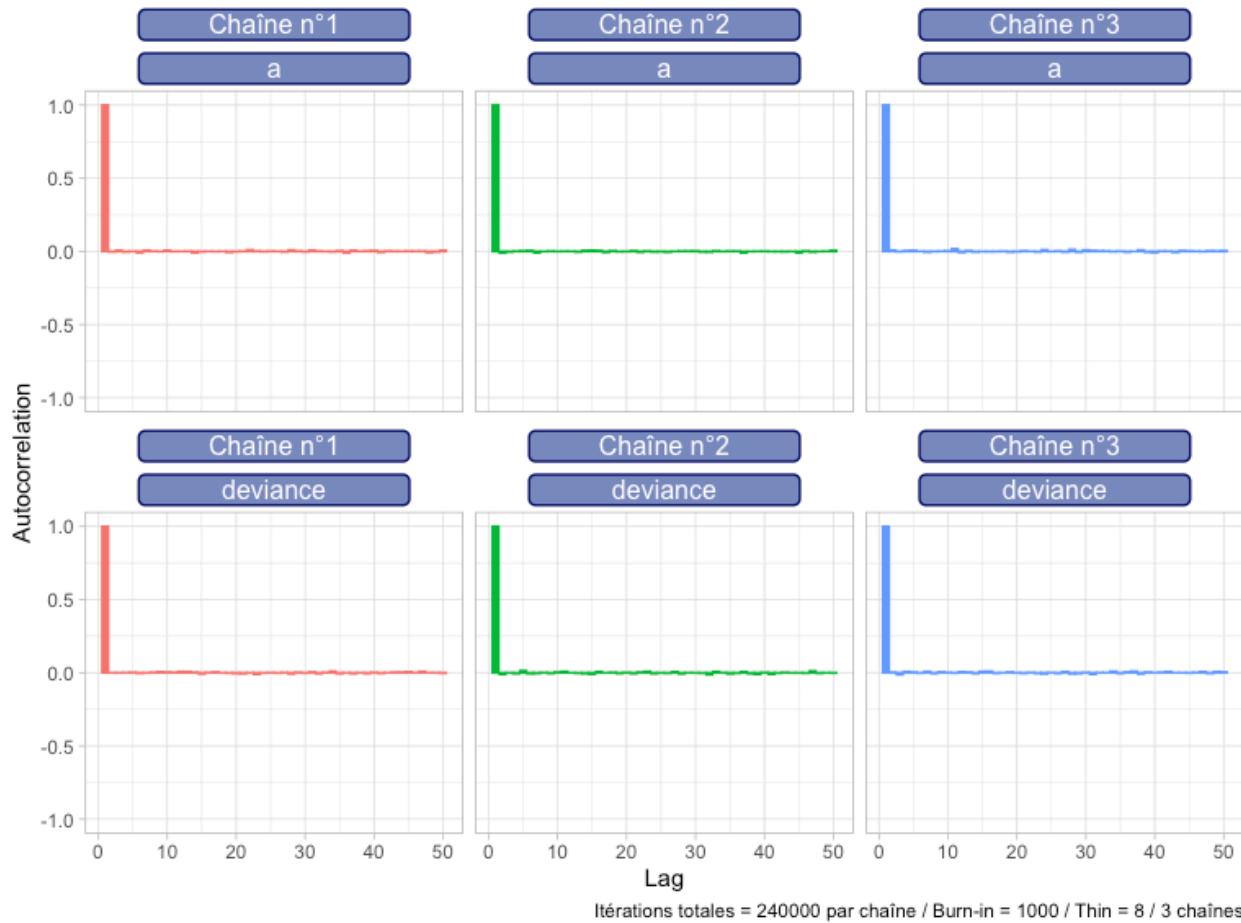
```

```

gg_modele1_mult <- ggs(modele1_fit_mult_mcmc)
ess_1_mult <- effectiveSize(modele1_fit_mult)

ggs_autocorrelation(gg_modele1_mult %>% mutate(Chain = paste0("Chaîne n°",
Chain))) +
facet_wrap(~ Chain + Parameter, ncol = 3, dir = "v") +
theme(legend.position = "none") +
labs(caption = "Itérations totales = 240000 par chaîne / Burn-in = 1000 /
Thin = 8 / 3 chaînes")

```



Pour les 3 chaînes, on ne voit pas d'auto-corrélation dans notre modèle. Aussi, le nombre d'itérations effectives est resté autour des 89625 itérations faites : 89625 pour l'estimation de a et 89625 pour l'estimation de la déviance du modèle (à noter que l'effective sample size égale ici le nombre d'itérations réellement faites. Cela est peut-être du à l'indépendantes des itérations les unes des autres).

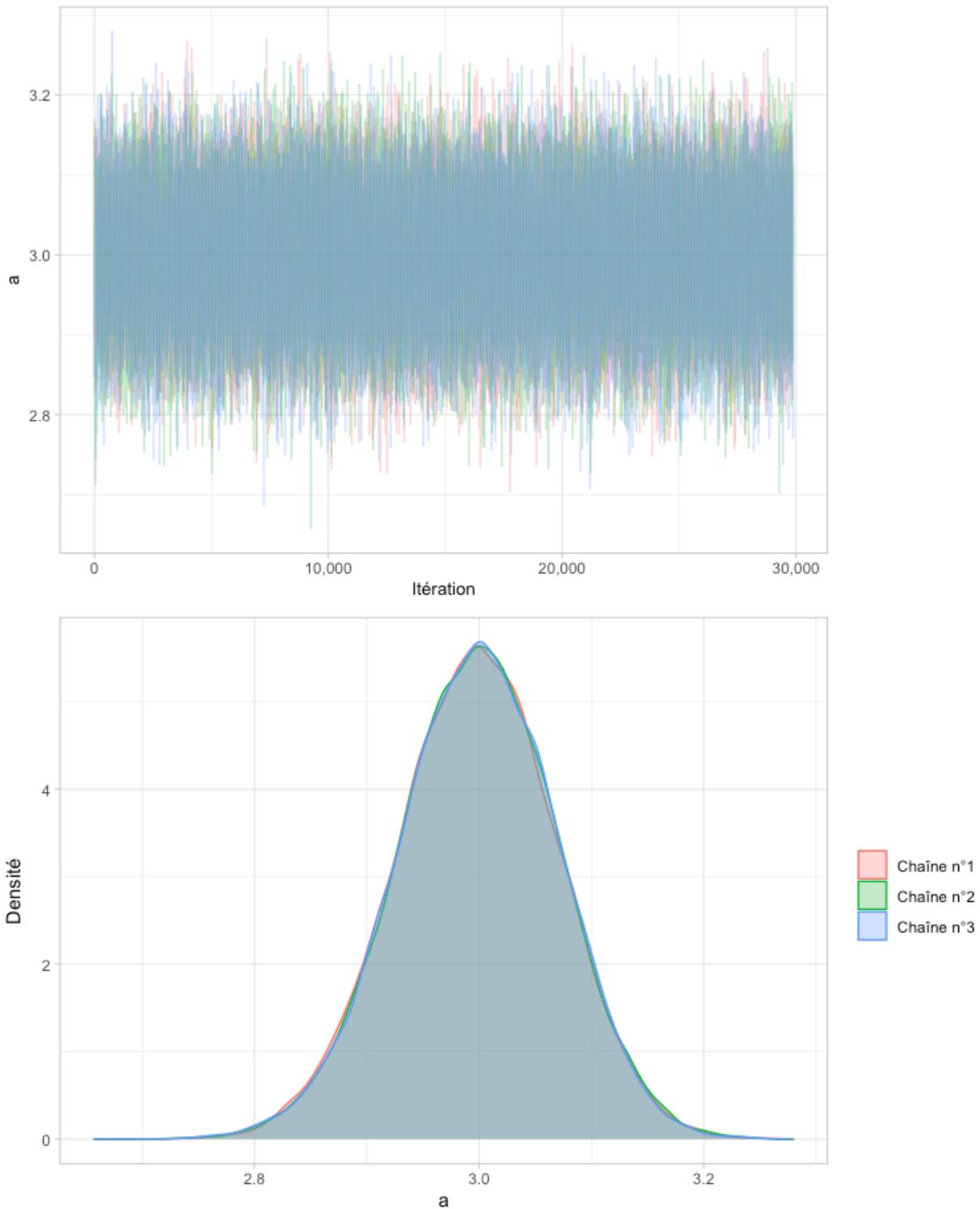
```

plot1 <- ggplot(gg_modele1_mult %>% filter(Parameter == "a"), aes(x =
Iteration, y = value)) +
geom_line(aes(color = as.factor(Chain)), alpha = 0.3) +
scale_x_continuous(labels = scales::comma_format()) +
labs(x = "Itération",
y = "a",

```

```
    title = "Traceplot de l'estimation de a par MCMC avec 3 chaînes") +
  theme(legend.position = "none",
        title = element_markdown(size = 10))
plot2 <- ggplot(gg_modele1_mult %>% filter(Parameter == "a") %>% mutate(Chain
= paste0("Chaîne n°", Chain)), aes(x = value, color = as.factor(Chain), fill
= as.factor(Chain))) +
  geom_density(alpha = 0.3) +
  scale_color_discrete(name = NULL) +
  scale_fill_discrete(name = NULL) +
  labs(x = "a",
       y = "Densité")
plot1 / plot2
```

Traceplot de l'estimation de a par MCMC avec 3 chaînes

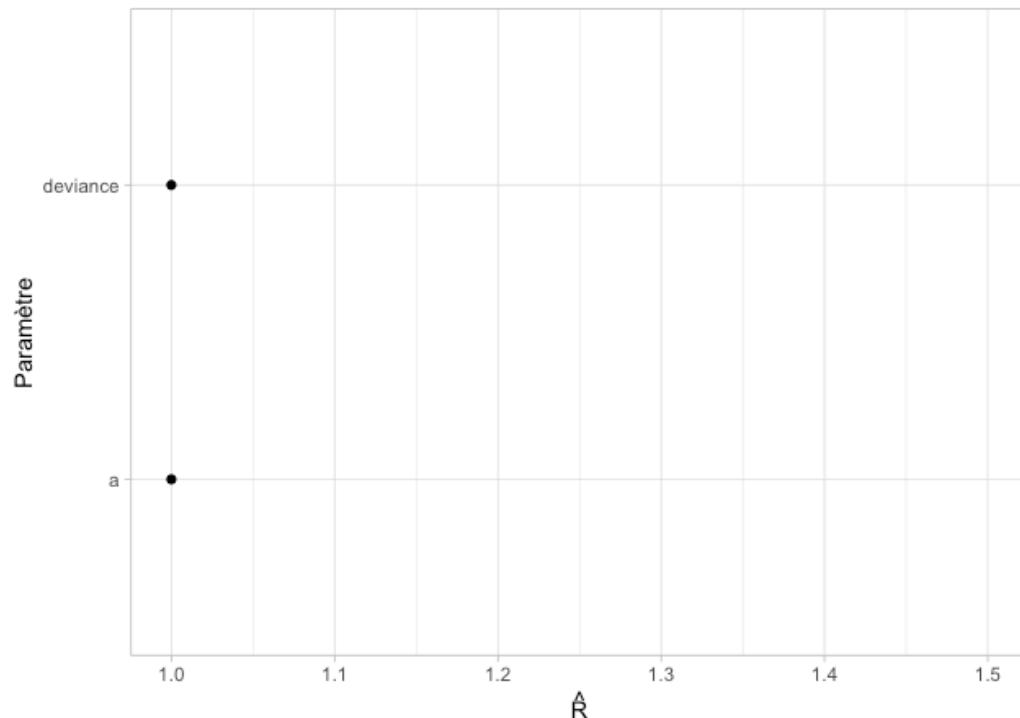


On peut voir que les 3 chaînes convergent bien vers la même valeur pour 3 initialisations différentes du paramètre a . Cela se voit sur le traceplot qui montre que les estimations de a

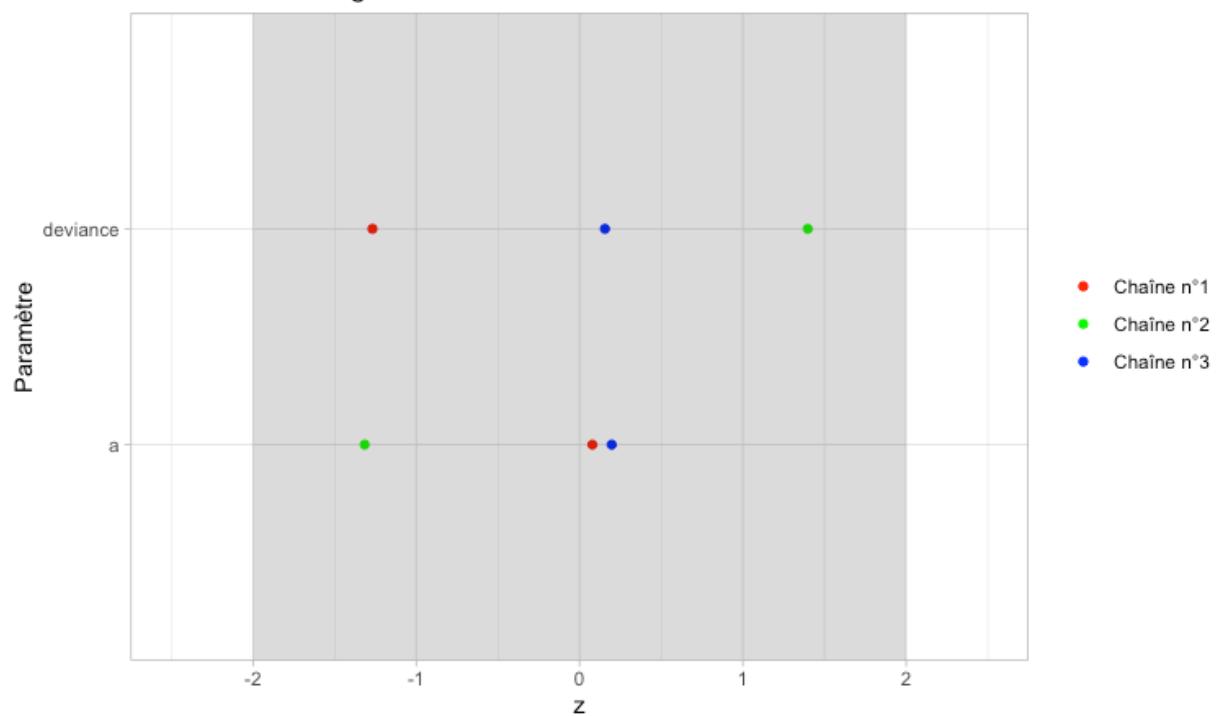
restent autour de la même valeur, mais aussi sur le graphique des densités de a pour chaque chaîne qui montre des densités superposables pour les 3 chaînes.

```
gelman <- ggs_Rhat(gg_modele1_mult) +
  labs(y = "Paramètre")
geweke <- ggs_geweke(gg_modele1_mult) +
  labs(y = "Paramètre",
       title = "Critère de convergence de Geweke") +
  scale_color_manual(name = NULL,
                     values = c("red", "green", "blue", NA),
                     labels = function(x) ifelse(x == "black", "", 
paste0("Chaîne n°", x)))
gelman / geweke
```

Potential Scale Reduction Factors



Critère de convergence de Geweke



Les critères de convergence de Gelman et Geweke sont bien respectés : le \hat{R} de Gelman est bien autour de 1 (il est même à 1) et le Z-score de Geweke reste entre -2DS et 2DS pour toutes les chaînes.

Ainsi, voici notre estimation du paramètre a pour notre modèle avec 3 chaînes, 240000 itérations par chaîne avec 1000 itérations de burn-in et la conservation d'une itération sur 8 :

```
tidy(modele1_fit_mult) %>%
  mutate(across(estimate:std.error, ~ round(.x, 3))) %>%
  flextable() %>%
  set_header_labels(term = "Paramètre estimé",
                    estimate = "Estimation",
                    std.error = "Ecart-Type") %>%
  autofit()
```

Paramètre estimé	Estimation	Ecart-Type
a	2.999	0.071

Cette estimation est très proche à celle faite sur une chaîne précédemment qui sera notre résultat principal.

3. Question 3

Que vaut le nombre d'itérations pour les calculs ? Que vaut le nombre d'itérations « effectif » ?

Pour le modèle avec 1 chaîne, 1000 itérations de burn-in et la conservation d'une itération sur 8, on avait 29875 itérations de faites, et le nombre effectif était de 29875 itérations.

A titre indicatif, le modèle avec 3 chaînes avait 89625 itérations de faites avec un nombre effectif de 89625 itérations. Ce nombre est plus grand que le nombre d'itérations réalisées peut-être en raison d'auto-corrélation négatives pour notre modèle, mais nous avons considéré que cela montrait que les itérations étaient indépendantes.

4. Question 4

Donnez la moyenne a posteriori et l'intervalle de crédibilité à 95% de a .

```
resm <- summary(modele1_fit_thin_mcmc)[[1]] %>% as.data.frame()
resq <- summary(modele1_fit_thin_mcmc)[[2]] %>% as.data.frame()
ic <- paste0(round(resm[["Mean"]][1], 2), "[", round(resq[["2.5%"]][1], 2),
";", round(resq[["97.5%"]][1], 2), "]")
```

L'estimation de a nous donne la moyenne et l'intervalle de crédibilité à 95% suivant : 3[2.86;3.13]. Nous avons représenté la densité à postériori de a sur le graphique suivant. Nous avons aussi intégré la loi à priori qui est une loi normale très plate et semble donc bien une loi non informative.

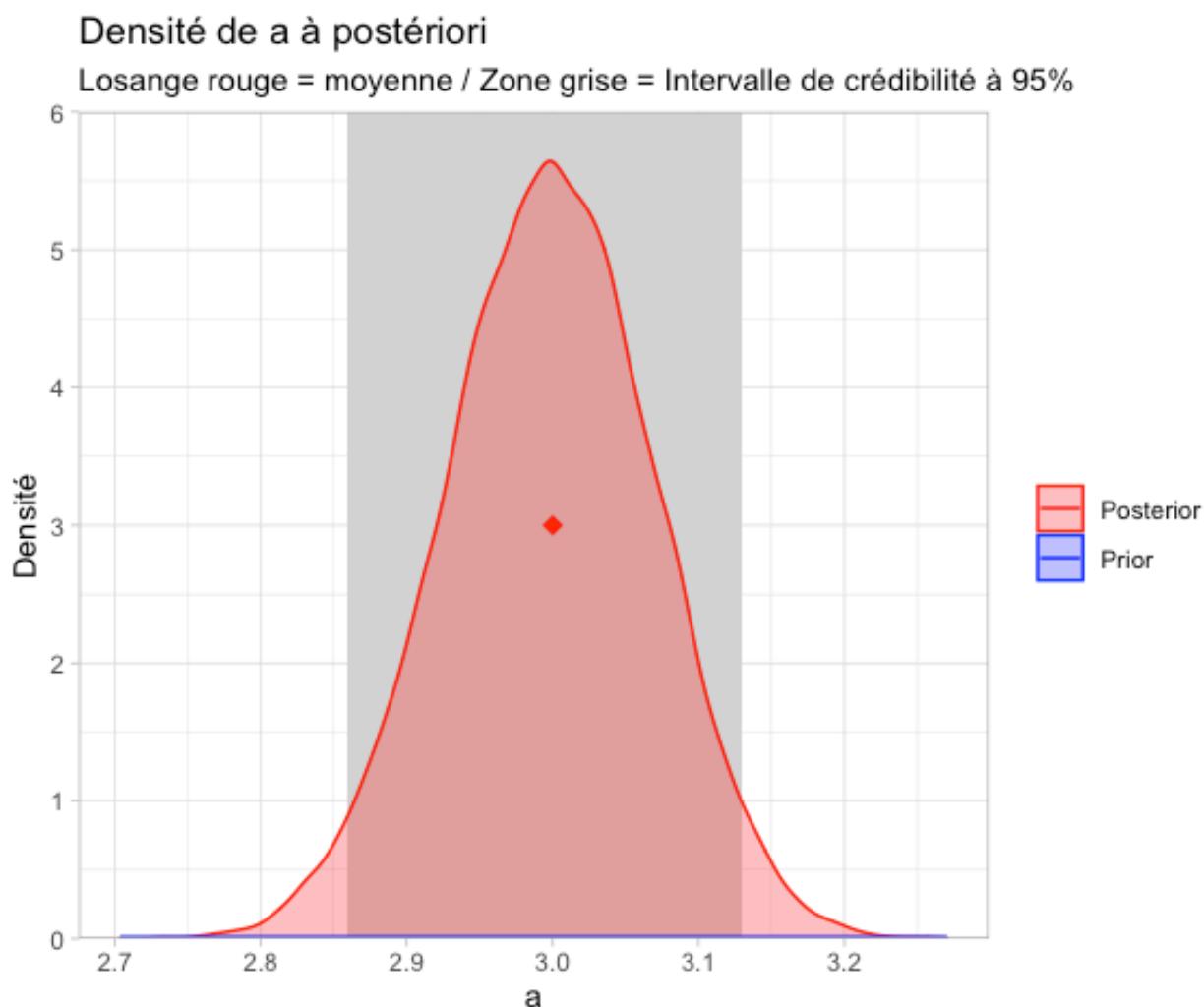
```
ggplot(gg_modele1_thin %>% filter(Parameter == "a")) +
  geom_rect(aes(xmin = 2.86, xmax = 3.13, ymin = 0, ymax = 6), fill =
"lightgrey", alpha = 0.2) +
  geom_density(aes(x = value, color = "Posterior"), fill = "red" , alpha =
0.3) +
```

```

annotate("point", x = 3, y = 3, color = "#FF0000", shape = 18, size = 3,
alpha = 1) +
  geom_function(aes(color = "Prior"), fun = ~ dnorm(.x, mean = 0, sd =
sqrt(1000))) +
  scale_color_manual(name = NULL, values = c("red", "blue")) +
  scale_fill_manual(name = NULL) +
  scale_y_continuous(expand = expansion(mult = c(0, 0))) +
  guides(color = guide_legend(override.aes = list(fill = c("red", "blue"))))

+ labs(x = "a",
      y = "Densité",
      title = "Densité de a à postériori",
      subtitle = "Losange rouge = moyenne / Zone grise = Intervalle de
crédibilité à 95%")

```



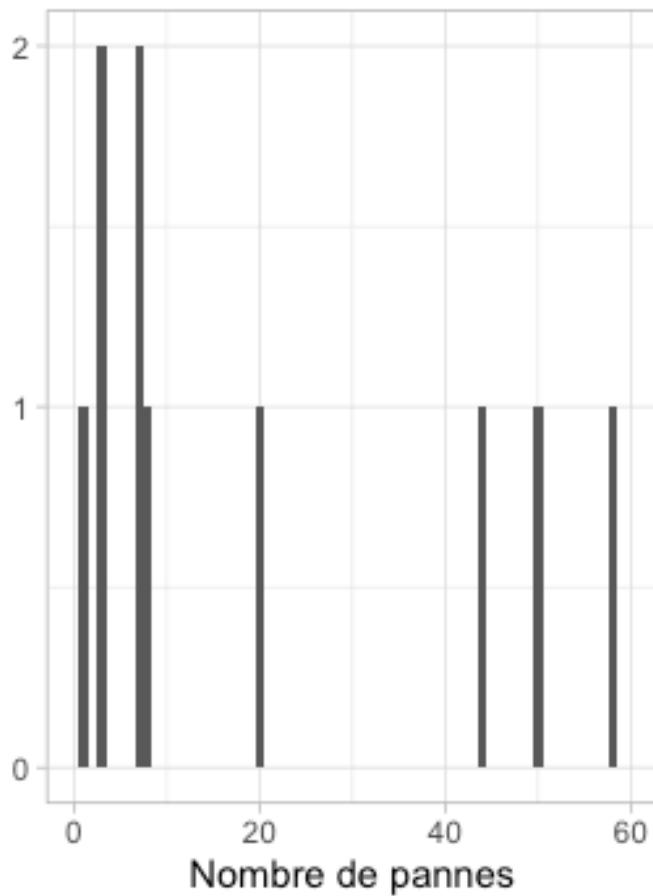
On peut représenter les lois de Poisson associées à ces différents paramètres a (le a moyen représente la loi associé à la moyenne de a , a minimum et maximum représentent les a des bornes de l'intervalle de crédibilité).

```

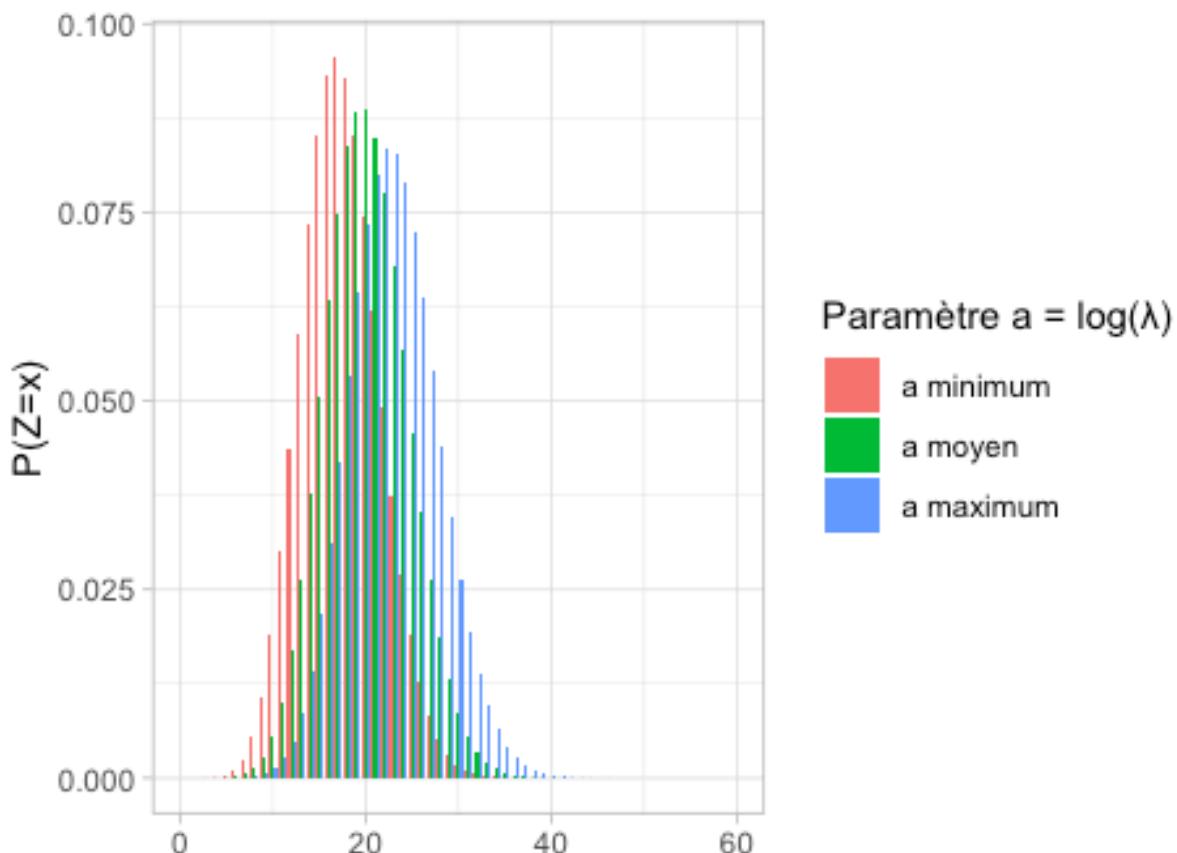
poisson_dens <- tibble(
  x = 0:60,
  poiss_min = (exp(2.86) ^ x) * (exp(-exp(2.86)) / factorial(x)),
  poiss_moy = (exp(3) ^ x) * (exp(-exp(3)) / factorial(x)),
  poiss_max = (exp(3.13) ^ x) * (exp(-exp(3.13)) / factorial(x))
)
dens_poiss <- ggplot(poisson_dens %>% pivot_longer(-x) %>%
  mutate(name = factor(name, levels = c("poiss_min", "poiss_moy",
  "poiss_max")),
         labels = c("a minimum", "a moyen", "a
maximum")))) +
  geom_col(aes(x = x, y = value, fill = name), position = position_dodge()) +
  scale_fill_discrete(name = "Paramètre a = log(&lambd;a)") +
  labs(x = "Nombre de pannes",
       y = "P(Z=x)",
       title = "Lois de Poisson") +
  theme(legend.title = element_markdown()) +
  xlim(c(0, 60))
dens_pannes <- ggplot(sncf_machines, aes(x = nb_pannes)) +
  geom_bar() +
  xlim(c(0, 60)) +
  scale_y_continuous(breaks = 0:2) +
  labs(x = "Nombre de pannes",
       y = "",
       title = "Nombre de pannes chez les machines de l'exercice")
dens_pannes / dens_poiss

```

Nombre de pannes chez les machines de l'exercice



Lois de Poisson



On peut voir que les lois de Poisson décrivent mal la survenue de pannes pour les machines. Si la moyenne de la loi de Poisson pour le moyen est très proche de la moyenne dans l'échantillon de 10 machines (20.04 vs 20.1), la loi ne décrit pas de façon très satisfaisante la distribution du nombre de pannes. Il faut sûrement estimer plus de paramètres.

5. Question 5

Que vaut le DIC ? Que vaut l'estimation de la complexité du modèle ? Vous semble-t-elle logique ?

```
dic <- round(modele1_fit_thin$BUGSoutput$DIC, 4)
complexite <- round(modele1_fit_thin$BUGSoutput$pD, 4)
```

Le DIC du modèle vaut 253.367.

Pour estimer la complexité du modèle, le pD est estimé et représente le nombre effectif de paramètres estimés. Pour notre modèle, il vaut 0.9995. Ce chiffre est voisin de 1, ce qui est en accord avec le modèle car nous n'estimons qu'un paramètre : a qui vaut $\log(\lambda)$ et est unique pour toutes les machines.

6. Question 6

Refaire tourner ce modèle (30000 itérations et enlever 1000 itérations pour le temps de chauffe) mais avec cette fois-ci comme loi a priori sur a , une loi normale d'espérance nulle et de variance 10000. Donnez la moyenne a posteriori et l'intervalle de crédibilité à 95% de a et commentez.

Pour la loi à priori de a , on prendra une variance plus élevée à 10000 au lieu de 1000, et donc $\tau = \frac{1}{\sigma^2} = \frac{1}{10000} = 0.0001$

```
modele_1_sens <- function() {
  # Modèle pour yi
  for (i in 1:10) {
    nb_pannes[i] ~ dpois(exp(a))
  }
  # Loi a priori de a
  a ~ dnorm(0, 0.0001)
}
set.seed(1993)
modele1_fit_sens <- jags(data = donnees,
                           inits = inits_modele1,
                           parameters.to.save = parametres_modele1,
                           n.chains = length(inits_modele1),
                           n.iter = n_iter * n_thin,
                           n.burnin = n_burn,
                           n.thin = n_thin,
                           model.file = modele_1_sens)
modele1_fit_sens_mcmc <- as.mcmc(modele1_fit_sens)
gg_modele1_sens <- ggs(modele1_fit_sens_mcmc)
ess_1_sens <- effectiveSize(modele1_fit_sens)
```

```

resm_sens <- summary(modele1_fit_sens_mcmc)[[1]] %>% as.data.frame()
resq_sens <- summary(modele1_fit_sens_mcmc)[[2]] %>% as.data.frame()
ic_sens <- paste0(round(resm_sens[["Mean"]][1], 2), "[",
round(resq_sens[["2.5%"]][1], 2), ";", round(resq_sens[["97.5%"]][1], 2),
")")
tidy(modele1_fit_sens) %>%
  mutate(across(estimate:std.error, ~ round(.x, 3))) %>%
  flextable() %>%
  set_header_labels(term = "Paramètre estimé",
                    estimate = "Estimation",
                    std.error = "Ecart-Type") %>%
  autofit()

```

	Paramètre estimé	Estimation	Ecart-Type
a		2.999	0.071

Les résultats du modèle apparaissent très similaires, avec un a moyen et son intervalle de crédibilité à 95% de 3[2.85;3.14].

Le comportement pour la mélangeance du modèle était très similaire au modèle précédent. Ainsi, on n'avait pas de problème de mélangeance dans notre modèle avec pas d'auto-corrélations et une bonne convergence.

```

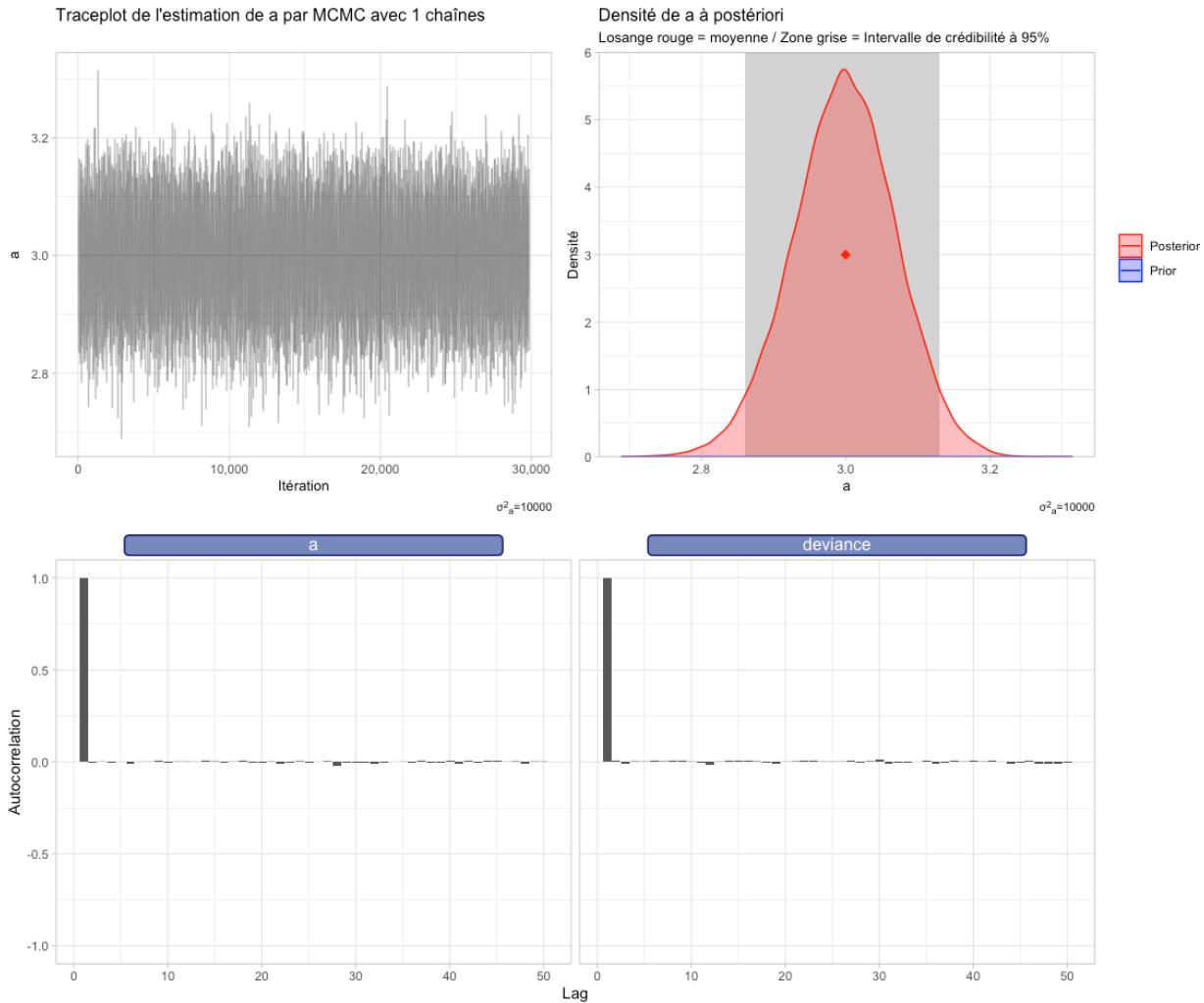
plot1 <- ggplot(gg_modele1_sens %>% filter(Parameter == "a"), aes(x =
Iteration, y = value)) +
  geom_line(alpha = 0.3) +
  scale_x_continuous(labels = scales::comma_format()) +
  labs(x = "Itération",
       y = "a",
       title = "Traceplot de l'estimation de a par MCMC avec 1 chaînes",
       caption = "&sigma;<sup>2</sup><sub>a</sub>=10000") +
  theme(legend.position = "none",
        title = element_markdown(size = 10))
plot2 <- ggplot(gg_modele1_sens %>% filter(Parameter == "a")) +
  geom_rect(aes(xmin = 2.86, xmax = 3.13, ymin = 0, ymax = 6), fill =
"lightgrey", alpha = 0.2) +
  geom_density(aes(x = value, color = "Posterior"), fill = "red" , alpha =
0.3) +
  annotate("point", x = 3, y = 3, color = "#FF0000", shape = 18, size = 3,
alpha = 1) +
  geom_function(aes(color = "Prior"), fun = ~ dnorm(.x, mean = 0, sd =
sqrt(10000))) +
  scale_color_manual(name = NULL, values = c("red", "blue")) +
  scale_fill_manual(name = NULL) +
  scale_y_continuous(expand = expansion(mult = c(0, 0))) +
  guides(color = guide_legend(override.aes = list(fill = c("red", "blue")))) +
  labs(x = "a",
       y = "Densité",

```

```

    title = "Densité de a à postériori",
    subtitle = "Losange rouge = moyenne / Zone grise = Intervalle de
crédibilité à 95%",
    caption = "&sigma;<sup>2</sup><sub>a</sub>=10000") +
theme(title = element_markdown(size = 10))
plot3 <- ggs_autocorrelation(gg_modele1_sens)
(plot1 + plot2) / plot3

```



En conclusion, en ayant pris une loi à priori sur le paramètre a encore plus plate et donc moins informative, nous avons les mêmes résultats. Cela confirme donc bien que la première loi à priori que nous avions choisie était bien non informative.

III. Modèle 2

Le modèle est le suivant :

$$y_i \sim \text{Pois}(\lambda_i)$$

avec $\begin{cases} y_i \text{ le nombre de pannes de la machine } i \\ \log(\lambda_i) = a_0 + b_0 \times x_i \\ x_i \text{ l'ancienneté de la machine } i \end{cases}$

On a $\log(\lambda) = a_0 + b_0 \times x_i \Leftrightarrow \lambda = e^{a_0 + b_0 \times x_i}$.

7. Question 1

Donner $E(y_i|a_0, b_0, x_i)$ d'après ce modèle en fonction de a_0 , b_0 et de x_i ? Si $b_0=0$, que cela signifie-t-il? Même question si b_0 est supérieur à 0 ou si b_0 est inférieur à 0 ?*

$$\begin{aligned} E(y_i|a_0, b_0, x_i) &= E(\text{Pois}(\lambda_i)) \\ &= \lambda_i \\ &= e^{a_0 + b_0 \times x_i} \end{aligned}$$

Si $b_0 = 0$ cela signifie que nous sommes dans le modèle 1 avec $E(y_i|a_0, b_0, x_i) = e^{a_0} (= e^a)$ et que le nombre de panne ne dépend pas de l'ancienneté de la machine.

Si $b_0 \neq 0$, on a $E(y_i|a_0, b_0, x_i) = e^{a_0 + b_0 \times x_i}$ et donc le nombre de pannes varie avec le vieillissement de la machine i. Si $b_0 < 0$, $a_0 + b_0 \times x_i$ diminue en fonction de l'ancienneté croissante de la machine i et donc le nombre de pannes diminue avec l'ancienneté de la machine (car la loi exponentielle est monotone croissante sur $]-\infty; +\infty[$). De manière analogue, si $b > 0$, $a_0 + b_0 \times x_i$ augmente en fonction de l'ancienneté croissante de la machine i et donc le nombre de pannes augmente avec l'ancienneté de la machine.

8. Question 2

Mettre en place ce modèle avec, comme loi a priori sur a_0 et b_0 , une loi normale d'espérance nulle et de variance 1000. Faire 30000 itérations et enlever 1000 itérations pour le temps de chauffe. D'après l'history et les autocorrélations, voyez-vous un problème de mélangeance de l'algorithme? Si oui, mettre un thin à 10. Cela a-t-il amélioré la mélangeance? On considérera que c'est suffisant.

Réalisation du modèle avec JAGS en prenant 30000 itérations avec 1000 itérations de burn-in au début, en gardant toutes les itérations :

```
modele_2 <- function() {
  for (i in 1:length(nb_pannes)) {
    lam[i] <- exp(a0 + b0 * anciennete[i])
    nb_pannes[i] ~ dpois(lam[i])
  }
  a0 ~ dnorm(0, 1.0E-3)
  b0 ~ dnorm(0, 1.0E-3)
}

# Paramètres
parametres_modele2 <- c("a0", "b0")

# Inits
```

```

inits2 <- list("a0" = 3, "b0" = 1)
inits_modele2 <- list(inits2)

# Nombre d'iterations
n_burn2 <- 1000
n_iter2 <- 30000
n_thin2 <- 1

# Modélisation
set.seed(1993)
modele2_fit <- jags(
  data = donnees,
  inits = inits_modele2,
  parameters.to.save = parametres_modele2,
  n.chains = length(inits_modele2),
  n.iter = n_iter2,
  n.burnin = n_burn2,
  n.thin = n_thin2,
  model.file = modele_2)
modele2_fit_mcmc <- as.mcmc(modele2_fit)
gg_modele2 <- ggs(modele2_fit_mcmc)
ess_2 <- effectiveSize(modele2_fit_mcmc)

```

On regarde si les paramètres estimés ont bien convergé.

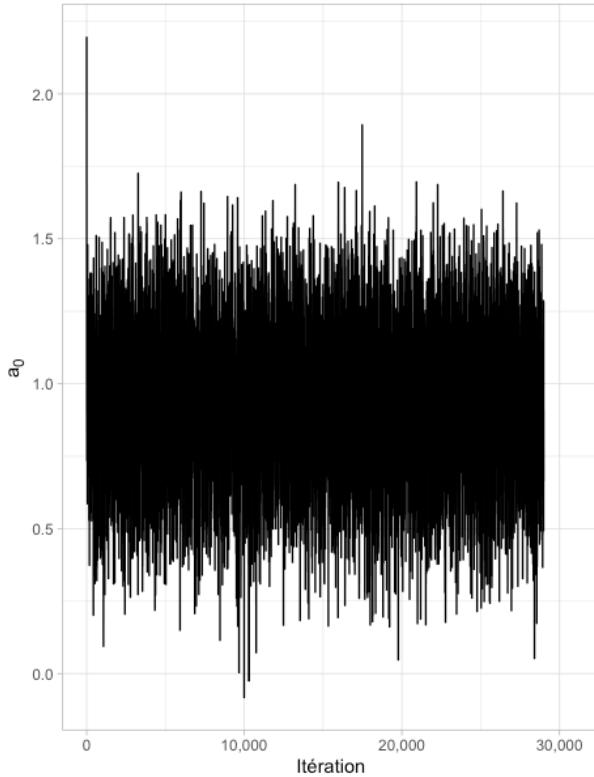
```

plot_a0 <- ggplot(gg_modele2 %>% filter(Parameter == "a0"), aes(x =
Iteration, y = value)) +
  geom_line() +
  scale_x_continuous(labels = scales::comma_format(), lim = c(0, 31000)) +
  labs(x = "Itération",
       y = "a<sub>0</sub>",
       title = "Traceplot de l'estimation de a<sub>0</sub> par MCMC",
       subtitle = "Initialisation de a<sub>0</sub> à 3") +
  theme(plot.title = element_markdown())
plot_b0 <- ggplot(gg_modele2 %>% filter(Parameter == "b0"), aes(x =
Iteration, y = value)) +
  geom_line() +
  scale_x_continuous(labels = scales::comma_format(), lim = c(0, 31000)) +
  labs(x = "Itération",
       y = "b<sub>0</sub>",
       title = "Traceplot de l'estimation de b<sub>0</sub> par MCMC",
       subtitle = "Initialisation de b<sub>0</sub> à 1") +
  theme(plot.title = element_markdown())
plot_a0 + plot_b0

```

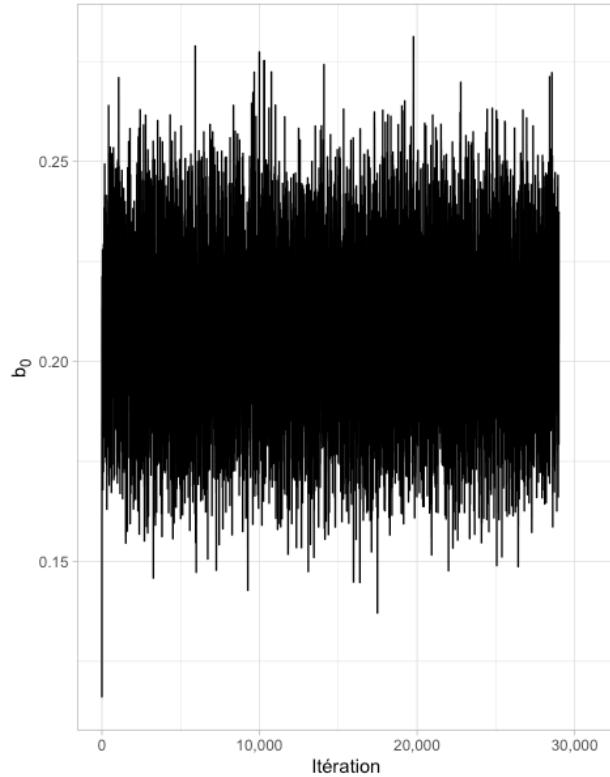
Traceplot de l'estimation de a_0 par MCMC

Initialisation de a_0 à 3



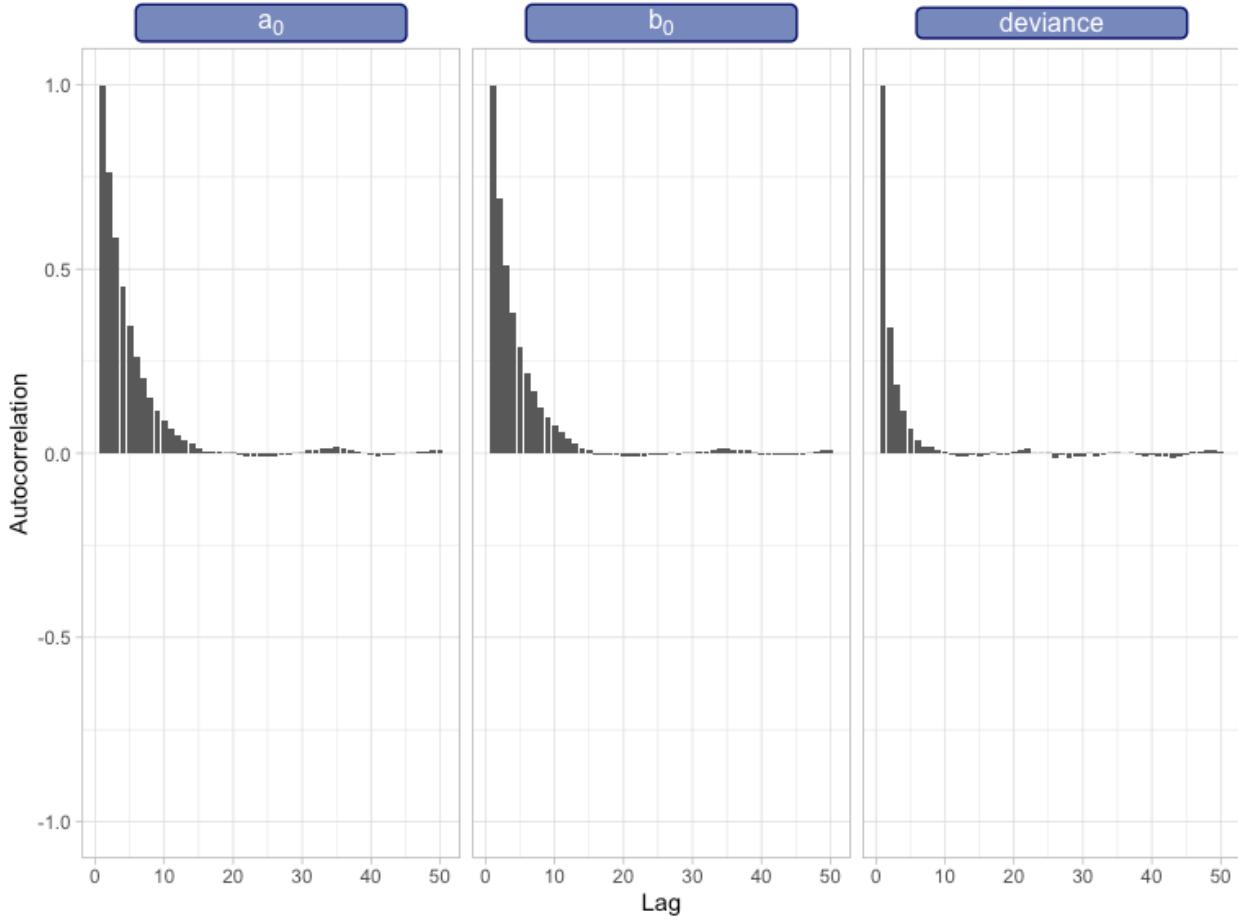
Traceplot de l'estimation de b_0 par MCMC

Initialisation de b_0 à 1



On voit que les valeurs des paramètres a_0 et b_0 restent respectivement autour de la valeur 1 et 0.2 et n'ont pas l'air de s'écarte beaucoup de cette valeur. Nous vérifierons par la suite si cette convergence est conservée en augmentant le nombre de chaînes. De plus, nous avons l'impression qu'il reste un peu de non-convergence des paramètres au début et allons augmenter un peu de burn-in en conséquence. Nous allons ensuite vérifier si les valeurs des paramètres estimés n'ont pas d'auto-corrélation.

```
ggs_autocorrelation(gg_modele2 %>%
  mutate(Parameter = case_when(Parameter == "a0" ~
    "a<sub>0</sub>",
    "b<sub>0</sub>",
    TRUE ~
    as.character(Parameter))))
```



On voit que pour l'estimation de a_0 et b_0 , il y a auto-corrélation jusqu'à la 15ème mesure. Pour la déviance, en revanche, cela va jusqu'à 10. De plus, le nombre d'itérations effectives est bien en-dessous du nombre d'itérations réalisées : nous avons fait 29000 itérations, et le nombre d'itérations effectives n'est que de 3809 pour a_0 et de 4546 pour b_0 . Ce nombre diminué indique une grande auto-corrélation des valeurs estimées entre les itérations.

Nous allons donc prendre une estimation sur 10 pour essayer de casser cette auto-corrélation en augmentant le nombre d'itérations en conséquence pour ne pas perdre le nombre d'itérations effectives.

```
n_thin2.1 <- 10
n_burn2 <- 2000
set.seed(1993)
modele2_fit_thin <- jags(data = donnees,
                           inits = inits_modele2,
                           parameters.to.save = parametres_modele2,
                           n.chains = length(inits_modele2),
                           n.iter = n_iter2 * n_thin2.1,
                           n.burnin = n_burn2,
                           n.thin = n_thin2.1,
                           model.file = modele_2)
modele2_fit_thin_mcmc <- as.mcmc(modele2_fit_thin)
```

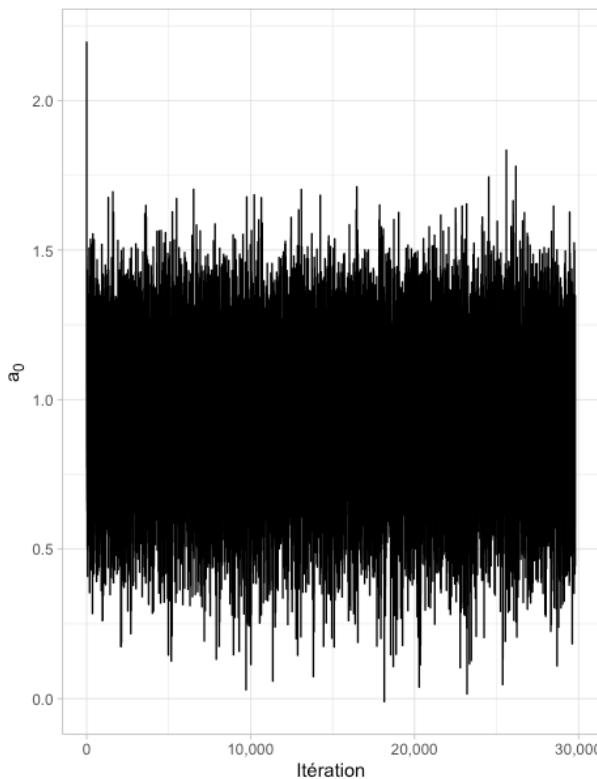
```

gg_modele2_thin <- ggs(modele2_fit_thin_mcmc)
ess_2_thin <- effectiveSize(modele2_fit_thin_mcmc)

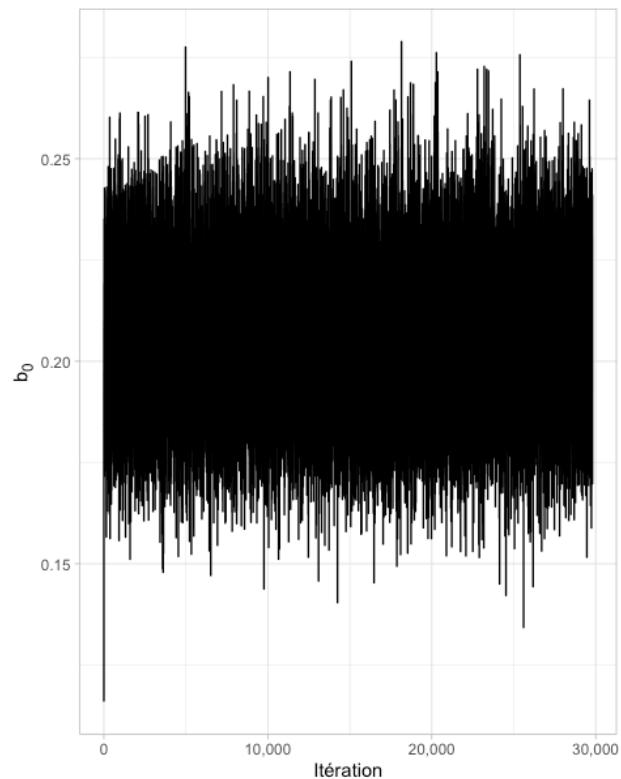
plot_a0 <- ggplot(gg_modele2_thin %>% filter(Parameter == "a0"), aes(x =
Iteration, y = value)) +
  geom_line() +
  scale_x_continuous(labels = scales::comma_format()) +
  labs(x = "Itération",
       y = "a<sub>0</sub>",
       title = "Traceplot de l'estimation de a<sub>0</sub> par MCMC",
       subtitle = "Initialisation de a<sub>0</sub> à 3 / Prise d'une valeur
sur 10") +
  theme(plot.title = element_markdown())
plot_b0 <- ggplot(gg_modele2_thin %>% filter(Parameter == "b0"), aes(x =
Iteration, y = value)) +
  geom_line() +
  scale_x_continuous(labels = scales::comma_format()) +
  labs(x = "Itération",
       y = "b<sub>0</sub>",
       title = "Traceplot de l'estimation de b<sub>0</sub> par MCMC",
       subtitle = "Initialisation de b<sub>0</sub> à 1 / Prise d'une valeur
sur 10") +
  theme(plot.title = element_markdown())
plot_a0 + plot_b0

```

Traceplot de l'estimation de a_0 par MCMC
Initialisation de a_0 à 3 / Prise d'une valeur sur 10

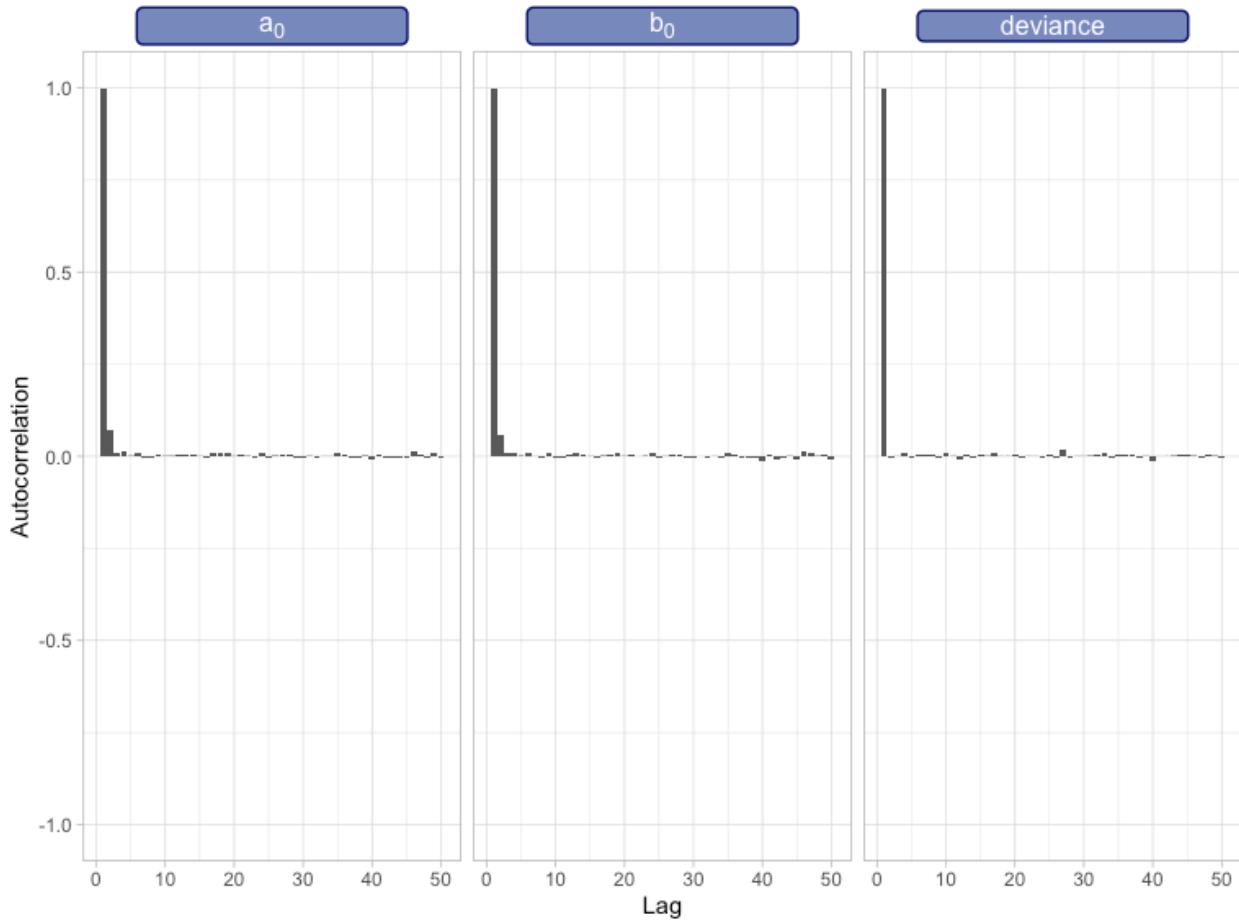


Traceplot de l'estimation de b_0 par MCMC
Initialisation de b_0 à 1 / Prise d'une valeur sur 10



En ne prenant qu'une observation sur 10, on voit que le traceplot reste similaire avec une bonne convergence de l'estimation de a_0 et de b_0 autour des même valeurs après le retrait de 2000 observations de burn in. On remarque aussi que le reste de non-convergence au début n'a pas été totalement réglé avec l'augmentation du burn-in, mais cela n'apparaît que sur une itération on dirait et en augmentant le burn in nous obtenons le même résultat, donc nous allons garder ce burn-in de 2000.

```
ggs_autocorrelation(gg_modele2_thin %>%
  mutate(Parameter = case_when(Parameter == "a0" ~
    "a<sub>0</sub>",
    Parameter == "b0" ~
    TRUE ~
  as.character(Parameter))))
```



Sur le graphique d'auto-corrélation, on voit que le problème a été amélioré. En regardant le nombre d'itérations effectives, nous pouvons remarquer que ça s'est amélioré aussi : nous avons fait 29800 itérations et le nombre d'itérations effectives est de 25905 pour a_0 et de 26482 pour b_0 . Ces chiffres sont encore un peu en-dessous du nombre d'itérations réalisées, mais l'amélioration est conséquente. Pour un lag de 1 (a_0 et b_0), il reste un peu d'auto-

corrélation, mais comme demandé dans l'énoncé, nous conserverons notre thin à 10 et considérerons que les itérations sont indépendantes les unes des autres.

Les résultats de notre modèle pour les estimations de a_0 et b_0 sont les suivants :

```
modele2_fit_thin$BUGSoutput$summary[1:2, 1:2] %>%
  as.data.frame() %>% rownames_to_column() %>%
  mutate(across(where(is.numeric), ~ round(.x, 3))) %>%
  flextable() %>%
  set_header_labels(rowname = "Paramètre estimé",
                    mean = "Estimation",
                    sd = "Ecart-Type") %>%
  autofit()
```

Paramètre estimé	Estimation	Ecart-Type
a0	0.943	0.228
b0	0.206	0.018

On a donc un nombre moyen de pannes de base représenté par e^{a_0} qui est donc positif, et un b_0 positif indiquant que le nombre de pannes augmente avec l'ancienneté de la machine.

Afin de nous assurer de la convergence du modèle, nous avons réalisé 3 chaînes avec des départ pour des valeurs différentes. Nous avons initié a_0 à 5, -5 et 0, b_0 à 5, -5 et 0 et regardé comment se comportait le modèle.

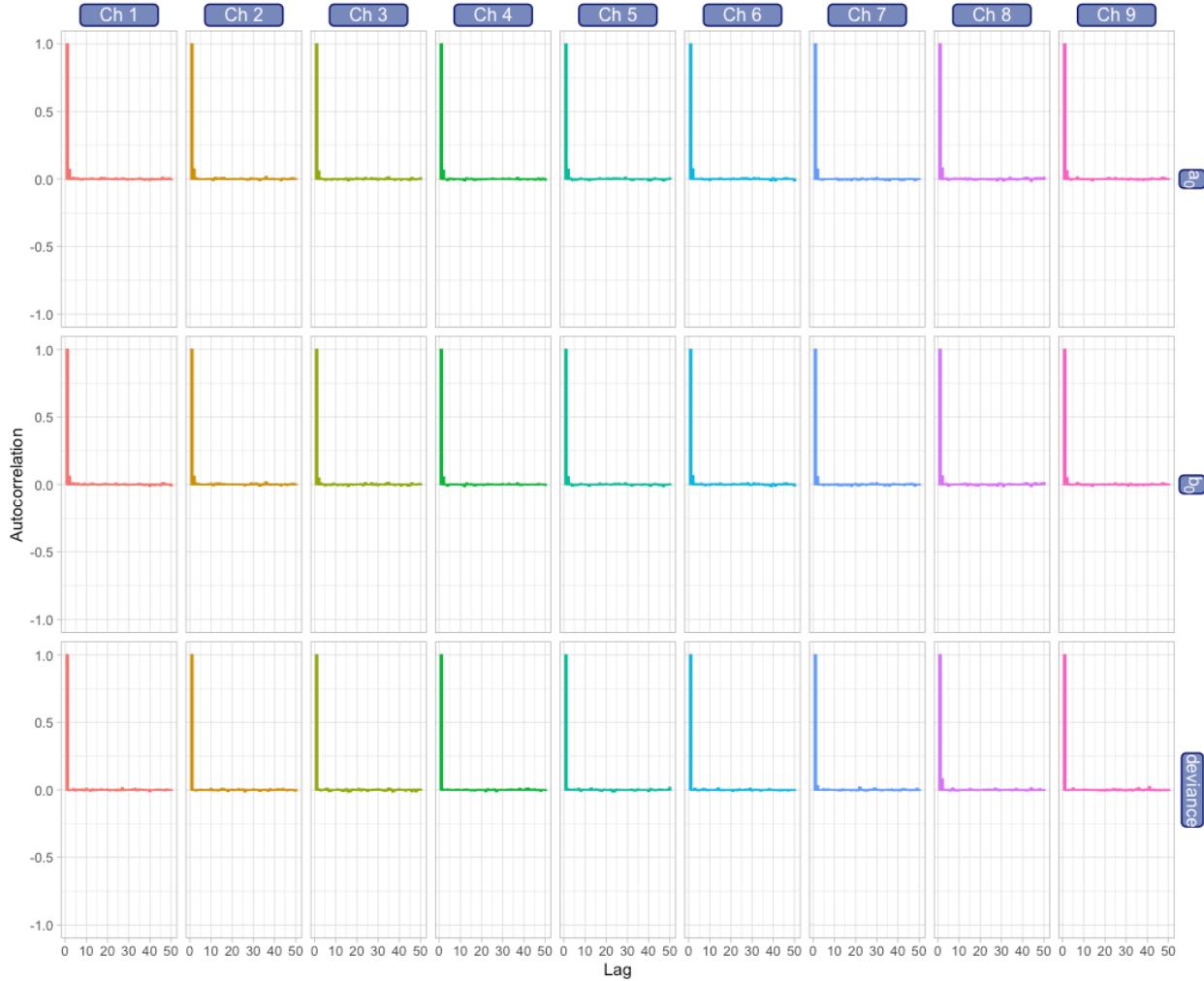
```
grille <- expand.grid(a0 = c(5, 0, -5), b0 = c(5, 0, -5)) %>%
  as.data.frame()
inits_modele2_mult <- list()
for (i in seq_len(nrow(grille))) {
  inits_modele2_mult[[i]] <- list(a0 = grille[i, "a0"], b0 = grille[i, "b0"])
}
set.seed(1993)
modele2_fit_mult <- jags(data = donnees,
                           inits = inits_modele2_mult,
                           parameters.to.save = parametres_modele2,
                           n.chains = length(inits_modele2_mult),
                           n.iter = n_iter2 * n_thin2.1,
                           n.burnin = n_burn2,
                           n.thin = n_thin2.1,
                           model.file = modele_2)
modele2_fit_mult_mcmc <- as.mcmc(modele2_fit_mult)
gg_modele2_mult <- ggs(modele2_fit_mult_mcmc)
ess_2_mult <- effectiveSize(modele2_fit_mult_mcmc)

ggs_autocorrelation(gg_modele2_mult %>%
  mutate(Chain = paste0("Ch ", Chain),
        Parameter = case_when(Parameter == "a0" ~
          "a<sub>0</sub>",
                               Parameter == "b0" ~
```

```

"b<sub>0</sub>",
TRUE ~
as.character(Parameter))) +
theme(legend.position = "none")

```



Au niveau des auto-corrélations, nous avons un résultat similaire à celui que nous avions eu avec une chaîne : cela semble satisfaisant avec peut-être un peu de corrélation pour un lag de 1. De plus, l'effective sample size reste proche du nombre d'itérations conservées : nous avons conservé 268200 itérations, et les nombres effectifs sont de 234519 pour a_0 et de 240142 pour b_0 . Nous avons considéré que les itérations étaient donc suffisamment indépendantes les unes des autres.

```

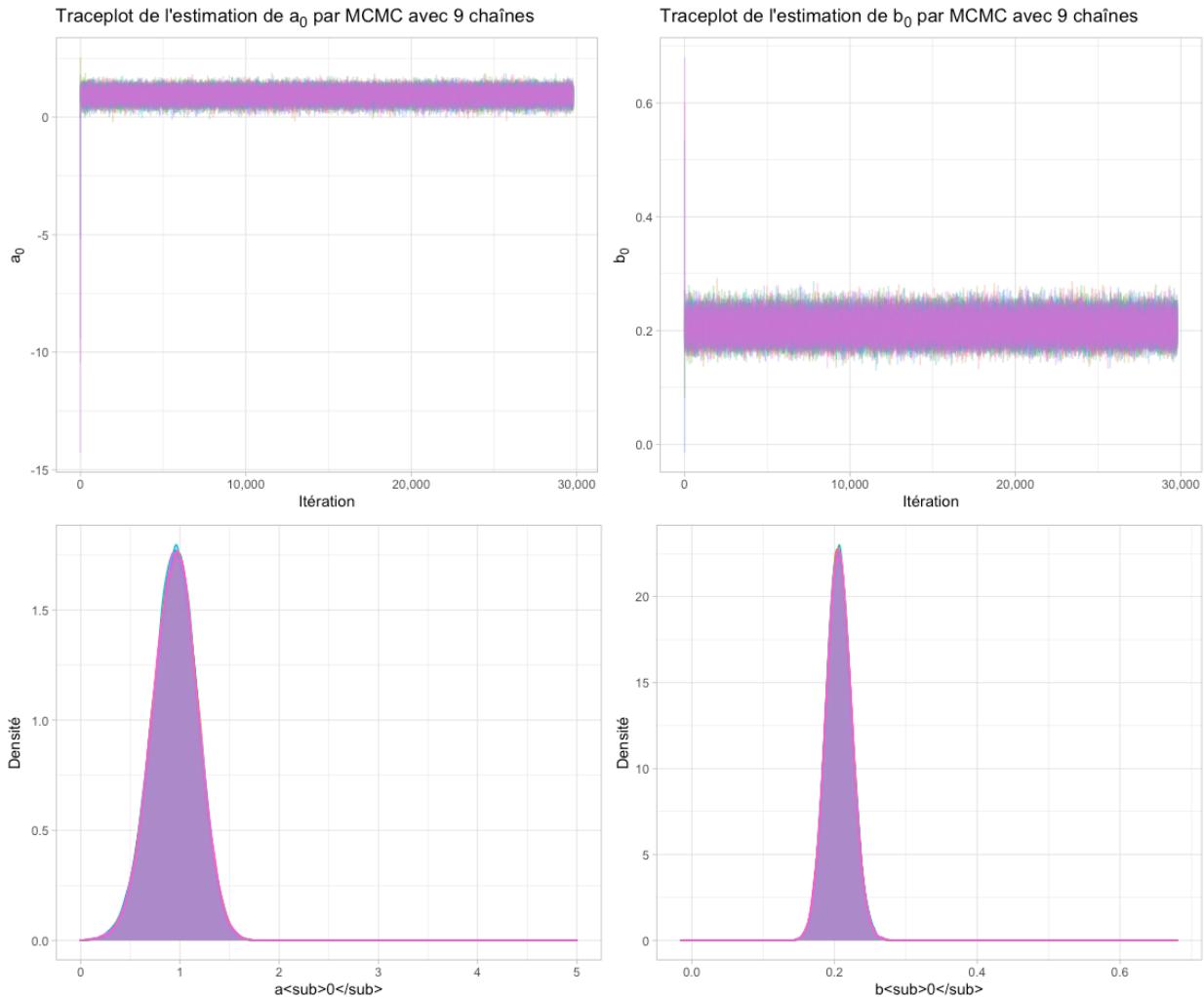
plot_peignea <- ggplot(gg_modele2_mult %>% filter(Parameter == "a0"), aes(x =
Iteration, y = value)) +
geom_line(aes(color = as.factor(Chain)), alpha = 0.3) +
scale_x_continuous(labels = scales::comma_format()) +
labs(x = "Itération",
y = "a<sub>0</sub>",
title = "Traceplot de l'estimation de a<sub>0</sub> par MCMC avec 9

```

```

chaines") +
  theme(legend.position = "none",
        plot.title = element_markdown())
plot_peigneb <- ggplot(gg_modele2_mult %>% filter(Parameter == "b0"), aes(x =
Iteration, y = value)) +
  geom_line(aes(color = as.factor(Chain)), alpha = 0.3) +
  scale_x_continuous(labels = scales::comma_format()) +
  labs(x = "Itération",
       y = "b<sub>0</sub>",
       title = "Traceplot de l'estimation de b<sub>0</sub> par MCMC avec 9
chaines") +
  theme(legend.position = "none",
        plot.title = element_markdown())
plot_densa <- ggplot(gg_modele2_mult %>% filter(Parameter == "a0"), aes(x =
value, color = as.factor(Chain), fill = as.factor(Chain))) +
  geom_density(alpha = 0.3) +
  labs(x = "a<sub>0</sub>",
       y = "Densité") +
  theme(legend.position = "none") +
  xlim(c(0, 5))
plot_densb <- ggplot(gg_modele2_mult %>% filter(Parameter == "b0"), aes(x =
value, color = as.factor(Chain), fill = as.factor(Chain))) +
  geom_density(alpha = 0.3) +
  labs(x = "b<sub>0</sub>",
       y = "Densité") +
  theme(legend.position = "none")
(plot_peignea + plot_peigneb) / (plot_densa + plot_densb)

```



On peut voir que les 9 chaînes convergent bien vers la même valeur pour 3 initialisation différentes du paramètre a_0 et du paramètre b_0 . Cela se voit sur le traceplot qui montre que les estimations de a_0 restent autour de la même valeur d'environ 1 (et 0.2 pour b_0), mais aussi sur le graphique des densités de a_0 et de b_0 pour chaque chaîne qui montre des densités superposables pour les 9 chaînes. Il y a sur les traceplots un peu de variabilité sur les 1ère itérations, mais en augmentant le burn-in, cela n'a pas changé ce phénomène, et il n'a l'air que très localisé au début et nous ne pensons pas que cela a impacté les résultats. Nous avons aussi réalisé des diagnostics de convergence de Geweke et Gelman. Nous pouvons voir sur les représentations graphiques suivantes que le \hat{R} de Gelman reste à 1 et que le Z-score de Geweke est compris entre -2DS et 2DS pour toutes les chaînes, signes d'une bonne convergence pour tous les paramètres du modèle.

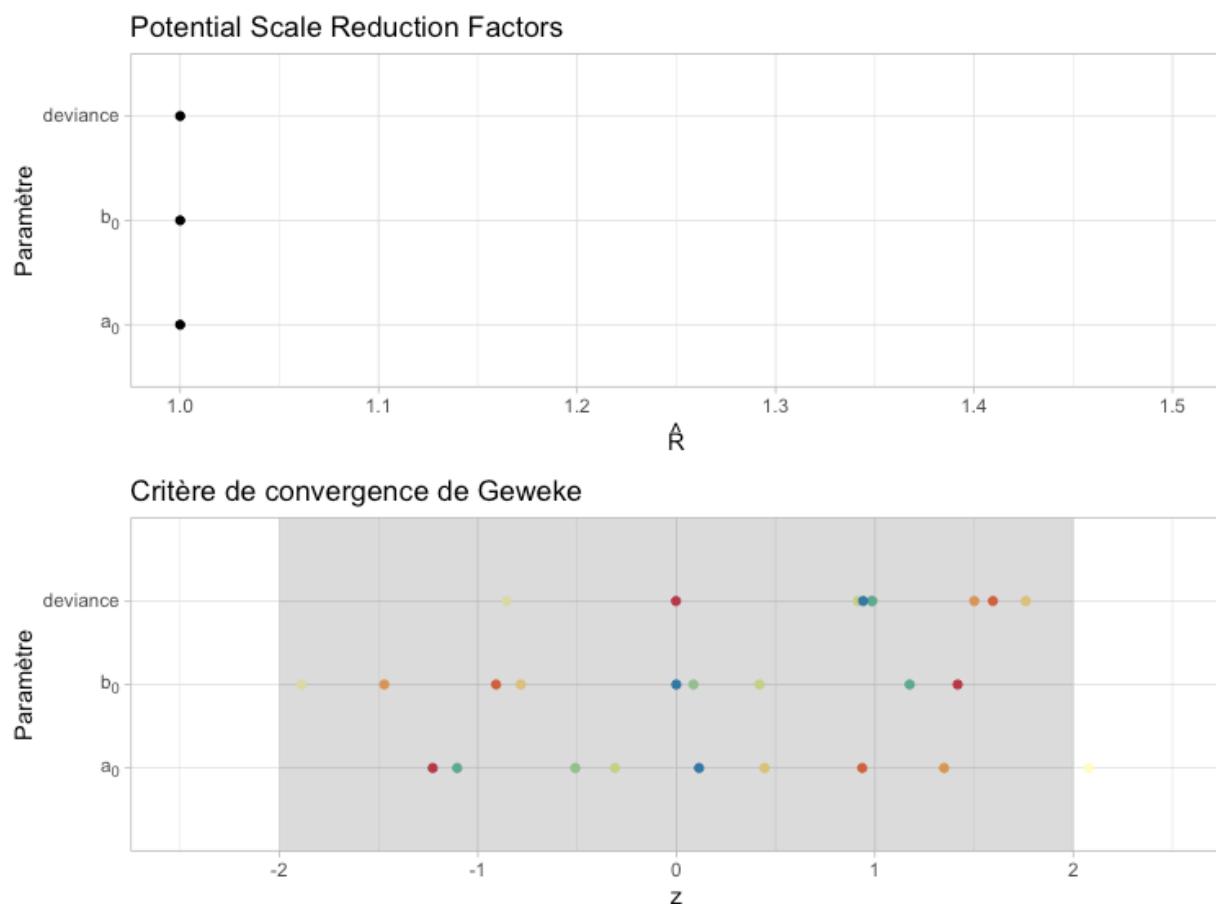
```
gelman <- ggs_Rhat(gg_modele2_mult %>% mutate(Parameter = case_when(Parameter == "a0" ~ "a<sub>0</sub>", Parameter == "b0" ~ "b<sub>0</sub>",
  TRUE ~ as.character(Parameter)))) +
```

```

  labs(y = "Paramètre",
        x = expression(hat("R")))) +
  theme(axis.text.y = element_markdown())
geweke <- ggs_geweke(gg_modele2_mult %>% mutate(Parameter =
  case_when(Parameter == "a0" ~ "a<sub>0</sub>",
            Parameter == "b0" ~
  TRUE ~
  as.character(Parameter)))) +
  labs(y = "Paramètre",
        title = "Critère de convergence de Geweke") +
  scale_color_manual(name = NULL,
                     values = c(brewer.pal(n = 9, name = "Spectral"), NA),
                     labels = function(x) ifelse(x == "black", "",

paste0("Chaîne n°", x))) +
  theme(axis.text.y = element_markdown(),
        legend.position = "none")
gelman / geweke

```



Par ailleurs, les résultats de notre modèle avec 9 chaînes sont très proches des résultats du modèle à 1 chaîne :

```

modele2_fit_mult$BUGSoutput$summary[1:2, 1:2] %>%
  as.data.frame() %>% rownames_to_column() %>%
  mutate(across(where(is.numeric), ~ round(.x, 3))) %>%
  flextable() %>%
  set_header_labels(rownname = "Paramètre estimé",
                    mean = "Estimation",
                    sd = "Ecart-Type") %>%
  autofit()

```

Paramètre estimé	Estimation	Ecart-Type
a0	0.944	0.231
b0	0.206	0.018

9. Question 3

Que vaut le nombre d'itérations pour les calculs ? Que vaut le nombre d'itérations « effectif » ?

Le nombre d'itérations pour notre calcul est de 300000 mais nous avons mis un thin de 10, ce qui fait que nous avions gardé 29800 itérations. Le nombre d'itérations effectif est de 25905 pour le paramètre a_0 et de 26482 pour le paramètre b_0 . Cela représente le nombre d'itérations qui ont apportée de l'information utile à notre modèle.

10. Question 4

Si ce n'est pas le cas, refaire tourner votre modèle pour que le nombre effectif d'itérations soit au moins de 10000.

Nous avions anticipé la diminution du nombre d'itérations avec l'augmentation du thin. Nous sommes donc supérieur à 10 000 itération effectives. Cependant pour atteindre ce chiffre nous pouvons diminuer notre nombre total d'itérations. Le nombre d'itération nécessaire pour atteindre 10 000 itérations effectives avoisine 120 000.

```

set.seed(1993)
modele2_fit_thinopti <- jags(data = donnees,
                                inits = inits_modele2,
                                parameters.to.save = parametres_modele2,
                                n.chains = length(inits_modele2),
                                n.iter = 120000,
                                n.burnin = n_burn2,
                                n.thin = n_thin2.1,
                                model.file = modele_2)

effectiveSize(modele2_fit_thinopti)

##   a0   b0 deviance
## 10206.90 10397.84 11800.00

```

11. Question 5

Donnez la moyenne a posteriori et l'intervalle de crédibilité à 95% de a_0 et b_0 .

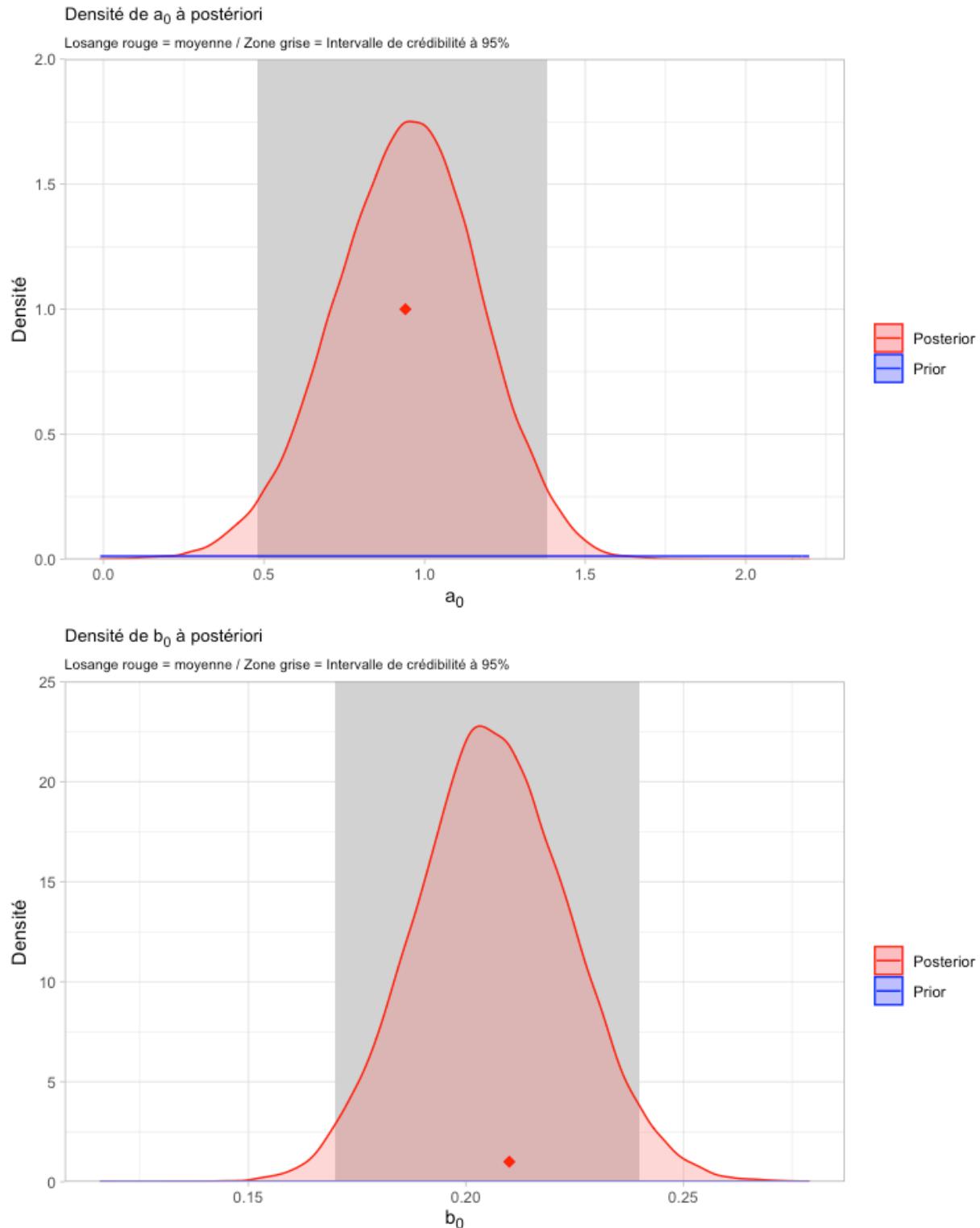
```
resm2 <- summary(modele2_fit_thin_mcmc)[[1]] %>% as.data.frame()
resq2 <- summary(modele2_fit_thin_mcmc)[[2]] %>% as.data.frame()
ic2a <- paste0(round(resm2[["Mean"]][1], 2), "[", round(resq2[["2.5%"]][1],
2), ";", round(resq2[["97.5%"]][1], 2), "]")
ic2b0 <- paste0(round(resm2[["Mean"]][2], 2), "[", round(resq2[["2.5%"]][2],
2), ";", round(resq2[["97.5%"]][2], 2), "]")
```

L'estimation de a_0 nous donne la moyenne et l'intervalle de crédibilité à 95% suivant : 0.94[0.48;1.38]. L'estimation de b_0 nous donne la moyenne et l'intervalle de crédibilité à 95% suivant : 0.21[0.17;0.24].

Nous avons représenté ces estimations sur les graphes de densité de a_0 et b_0 .

```
plot_a0 <- ggplot(gg_modele2_thin %>% filter(Parameter == "a0")) +
  geom_rect(aes(xmin = 0.48, xmax = 1.38, ymin = 0, ymax = 2), fill =
"lightgrey", alpha = 0.2) +
  geom_density(aes(x = value, color = "Posterior", fill = "Posterior"), alpha
= 0.3) +
  annotate("point", x = 0.94, y = 1, color = "#FF0000", shape = 18, size = 3,
alpha = 1) +
  geom_function(aes(color = "Prior"), fun = ~ dnorm(.x, mean = 0, sd =
sqrt(1000))) +
  scale_color_manual(name = NULL, values = c("red", "blue")) +
  scale_fill_discrete(name = NULL, labels = "") +
  scale_y_continuous(expand = expansion(mult = c(0, 0))) +
  guides(color = guide_legend(override.aes = list(fill = c("red", "blue"))),
        fill = guide_legend(override.aes = list(fill = NA, color = NA))) +
  labs(x = "a<sub>0</sub>",
       y = "Densité",
       title = "Densité de a<sub>0</sub> à postériori",
       subtitle = "Losange rouge = moyenne / Zone grise = Intervalle de
crédibilité à 95%") +
  theme(plot.title = element_markdown(size = 10),
        plot.subtitle = element_markdown(size = 8),
        axis.title.x = element_markdown())
plot_b0 <- ggplot(gg_modele2_thin %>% filter(Parameter == "b0")) +
  geom_rect(aes(xmin = 0.17, xmax = 0.24, ymin = 0, ymax = 25), fill =
"lightgrey", alpha = 0.2) +
  geom_density(aes(x = value, color = "Posterior", fill = "Posterior"), alpha
= 0.3) +
  annotate("point", x = 0.21, y = 1, color = "#FF0000", shape = 18, size = 3,
alpha = 1) +
  geom_function(aes(color = "Prior"), fun = ~ dnorm(.x, mean = 0, sd =
sqrt(1000))) +
  scale_color_manual(name = NULL, values = c("red", "blue")) +
  scale_fill_discrete(name = NULL, labels = "") +
```

```
scale_y_continuous(expand = expansion(mult = c(0, 0))) +  
guides(color = guide_legend(override.aes = list(fill = c("red", "blue"))),  
      fill = guide_legend(override.aes = list(fill = NA, color = NA))) +  
labs(x = "b<sub>0</sub>",  
     y = "Densité",  
     title = "Densité de b<sub>0</sub> à postériori",  
     subtitle = "Losange rouge = moyenne / Zone grise = Intervalle de  
crédibilité à 95%") +  
theme(plot.title = element_markdown(size = 10),  
      plot.subtitle = element_markdown(size = 8),  
      axis.title.x = element_markdown())  
plot_a0 / plot_b0
```



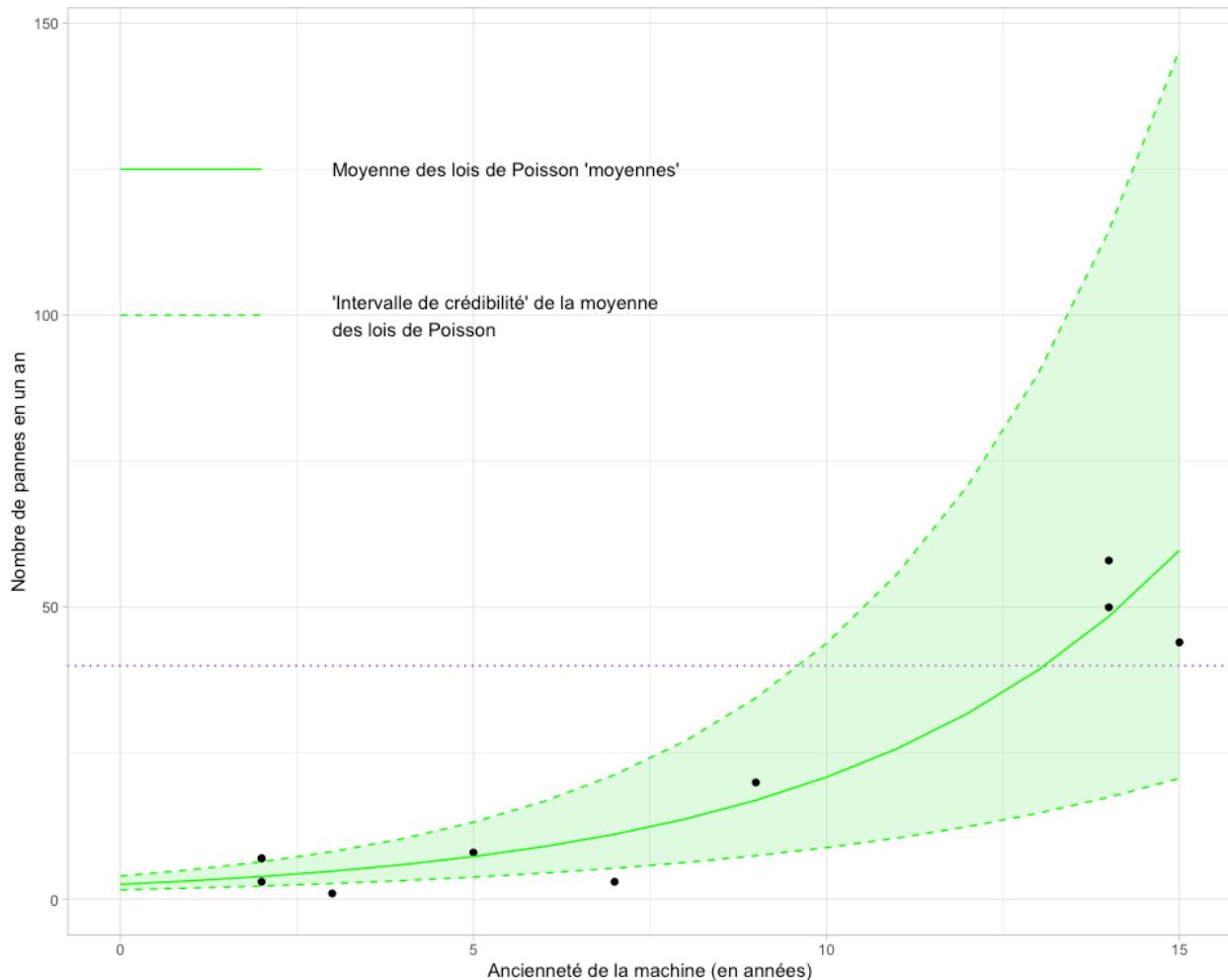
Nous avons représenté le nombre de pannes en fonction de l'âge de la machine. A cette représentation, nous avons essayé de représenter les moyennes des lois de Poisson

estimées. Pour ces lois, nous estimons les coefficients a_0 et b_0 permettant d'obtenir le paramètre λ des lois de Poisson.

Etant donné qu'on a 2 lois à postériori pour estimer le paramètre d'une loi, nous avons représenté la moyenne des lois de Poisson. Ainsi, la courbe pleine représente la loi de Poisson 'moyenne', c'est à dire qu'on a pris le a_0 et le b_0 moyen. Pour l'intervalle de crédibilité de la moyenne des lois de Poisson, nous avons pris dans le cas de la borne basse et haute, les bornes basses et hautes simultanément de a_0 et b_0 . Les moyennes obtenues sont reportées sur les courbes en pointillées.

On a un résumé par notre modèle qui est plus fidèle aux données que le modèle 1.

```
poissons_moy <- tibble(
  anciennete = 0:15,
  poiss_moymin = exp(0.48 + anciennete * 0.17),
  poiss_moymean = exp(0.94 + anciennete * 0.21),
  poiss_moymax = exp(1.38 + anciennete * 0.24)
)
ggplot() +
  geom_hline(yintercept = 40, linetype = "dotted", color = "purple") +
  geom_ribbon(data = poissons_moy, aes(x = anciennete, ymin = poiss_moymin,
  ymax = poiss_moymax), fill = "lightgreen", alpha = 0.3) +
  geom_point(data = sncf_machines, aes(anciennete, nb_pannes)) +
  geom_line(data = poissons_moy, aes(x = anciennete, y = poiss_moymean),
  color = "green") +
  geom_line(data = poissons_moy, aes(x = anciennete, y = poiss_moymin), color
  = "green", linetype = "dashed") +
  geom_line(data = poissons_moy, aes(x = anciennete, y = poiss_moymax), color
  = "green", linetype = "dashed") +
  annotate("segment", x = 0, xend = 2, y = 125, yend = 125, color = "green") +
  annotate("segment", x = 0, xend = 2, y = 100, yend = 100, linetype =
  "dashed", color = "green") +
  annotate("text", x = 3, y = 125, label = "Moyenne des lois de Poisson
'moyennes'", hjust = 0) +
  annotate("text", x = 3, y = 100, label = "'Intervalle de crédibilité' de la
moyenne\ndes lois de Poisson", hjust = 0) +
  labs(x = "Ancienneté de la machine (en années)",
  y = "Nombre de pannes en un an")
```



Les moyennes des lois de Poisson avec le λ_i estimé estiment bien les données dont nous disposons. Toutefois, pour construire cette figure, nous avons émis l'hypothèse que a_0 et b_0 varient selon leur intervalle de crédibilité de façon simultanée et identique, ce qui n'est pas forcément vrai.

12. Question 6

Pensez-vous que la variable x doit être prise en compte ? Donnez rapidement une interprétation du résultat (par exemple, pour deux machines ayant une différence d'ancienneté de 1 an, que représente $\exp(b_0)$?).

La variable x_i (ancienneté de la machine) doit être prise en compte car elle apporte de l'information à notre modèle. En effet, l'ancienneté est associé à un sur-risque de panne comme en témoigne le moyenne de b_0 supérieure à 0 et son intervalle de crédibilité à 95% qui ne recouvre pas 0. En raison de l'exclusion de 0 de cet intervalle de crédibilité, nous pensons qu'il faut bien prendre en compte l'ancienneté dans le modèle.

Prenons 2 machines i et j , avec $x_j = x_i + 1$. On peut écrire donc $\lambda_i = e^{a_0+b_0 \times x_i}$ et $\lambda_j = e^{a_0+b_0 \times x_j}$. On peut faire le rapport du nombre moyen de pannes entre ces 2 machines puisque ce nombre moyen vaut λ_i et λ_j .

$$\begin{aligned}\frac{\lambda_j}{\lambda_i} &= \frac{e^{a_0+b_0 \times x_j}}{e^{a_0+b_0 \times x_i}} \\ &= e^{a_0+b_0 \times x_j - (a_0+b_0 \times x_i)} \\ &= e^{a_0+b_0 \times (x_i+1) - a_0 - b_0 \times x_i} \\ &= e^{a_0 - a_0 + b_0 \times (x_i+1 - x_i)} \\ &= e^{b_0}\end{aligned}$$

Donc, e^{b_0} représente le facteur d'augmentation du nombre moyen de pannes pour une augmentation d'un an d'ancienneté : si on a une machine qui a un an de plus, elle aura en moyenne e^{b_0} fois plus de pannes.

13. Question 7

Que vaut le DIC ? Que vaut l'estimation de la complexité du modèle ? Vous semble-t-elle logique ?

```
dic2 <- round(modele2_fit_thin$BUGSoutput$DIC, 4)
complexite2 <- round(modele2_fit_thin$BUGSoutput$pD, 4)
```

Le DIC du modèle vaut 69.4381.

Pour estimer la complexité du modèle, le pD est estimé et représente le nombre effectif de paramètres estimés. Pour notre modèle, il vaut 2.0429. Ce chiffre est voisin de 2. Cela est cohérent puisque nous avions 2 paramètres dans notre modèle (a_0 et b_0).

14. Question 8

D'après le DIC, quel modèle choisissez-vous entre M1 et M2 ?

En relevant les DIC de nos deux modèles, nous choisissons le modèle M2. Ce dernier à un DIC nettement inférieur à celui du modèle 1, 69.4 contre 253. L'apport de la variable x (ancienneté de la machine) est donc associée une meilleure estimation du modèle. Le DIC vient apporter un argument de plus dans l'utilité d'ajouter l'ancienneté de la machine dans le modèle discutée dans la question 6 vis-à-vis du coefficient b_0 .

15. Question 9

Selon le modèle que vous avez retenu, dire quelles machines (s'il y en a) ont une moyenne du nombre de pannes probablement (c'est à dire à 97,5%) supérieure à 40 pannes (valeur pour laquelle la machine sera remplacée) ?

Sur la dernière figure de la question 5, on peut voir que la borne inférieure de l'intervalle de crédibilité à 95% reste toujours en-dessous de la valeur seuil 40. Ainsi, sur nos données, il n'y a aucune machine qui a un nombre moyen de pannes supérieur à 40.

On peut déterminer le nombre d'année d'ancienneté avant qu'une machine doive être remplacée :

$$\begin{aligned}
 e^{0.48+x_i \times 0.17} &> 40 \\
 0.48 + 0.17 \times x_i &> \log(40) \\
 0.17 \times x_i &> \log(40) - 0.48 \\
 x_i &> \frac{\log(40) - 0.48}{0.17} \\
 x_i &> 18.9
 \end{aligned}$$

Les machines ayant 19 ans et plus d'ancienneté seront donc à remplacer.

IV. Modèle 3

Le modèle est le suivant :

$$\begin{aligned}
 y_i &\sim \text{Pois}(\lambda_i) \\
 \text{avec } \begin{cases} y_i \text{ le nombre de pannes de la machine } i \\ \log(\lambda_i) = a_0 + b_0 \times x_i + \varepsilon_i \\ x_i \text{ l'ancienneté de la machine } i \\ \varepsilon_i \sim \mathcal{N}(0, \sigma^2) \end{cases}
 \end{aligned}$$

16. Question 1

Selon vous, quel est l'intérêt de l'ajout de la composante aléatoire ε_i ?

La composante aléatoire ε_i représente un "bruit" autour de la valeur de la moyenne de la machine i . En faisant cela, on ajoute l'information que 2 machines de même ancienneté sont différentes les unes des autres indépendamment de leur caractéristiques. Cette part d'aléatoire représente la part de variabilité inter-individuelle. Si on ne l'ajoute pas, nous pensons que cela surestime la corrélation entre le nombre de pannes par an et l'ancienneté de la machine, à la manière de l'imputation par régression dans le problème de l'imputation de données manquantes.

17. Question 2

Mettre en place ce modèle avec, comme loi a priori sur a_0 et b_0 , une loi normale d'espérance nulle et de variance 1000 et sur $\tau = \frac{1}{\sigma^2}$ une loi gamma de paramètres 0.01 et 0.01. Faire 30000 itérations et enlever 1000 itérations pour le temps de chauffe. D'après l'history et les autocorrélations, voyez-vous un problème de mélangeance de l'algorithme ? Si oui, mettre un thin à 10. Cela a-t-il amélioré la mélangeance ? Si non, rajoutez des itérations et augmentez le thin afin d'avoir des résultats raisonnables.

```

modele_3 <- function() {
  for (i in 1:length(nb_pannes)) {
    epsilon[i] ~ dnorm(0, tau)
    lam[i] <- exp(a0 + b0 * anciennete[i] + epsilon[i])
  }
}
  
```

```

    nb_pannes[i] ~ dpois(lam[i])
}
a0 ~ dnorm(0, 1.0E-3)
b0 ~ dnorm(0, 1.0E-3)
tau ~ dgamma(0.01, 0.01)
}

# Paramètres
parametres_modele3 <- c("a0", "b0", "tau")

# Inits
inits3 <- list("a0" = 1, "b0" = 1, "tau" = 1)
inits_modele3 <- list(inits3)

# Nombre d'iterations
n_burn3 <- 1000
n_iter3 <- 300000
n_thin3 <- 1

# Modélisation
set.seed(1993)
modele3_fit <- jags(
  data = donnees,
  inits = inits_modele3,
  parameters.to.save = parametres_modele3,
  n.chains = length(inits_modele3),
  n.iter = n_iter3,
  n.burnin = n_burn3,
  n.thin = n_thin3,
  model.file = modele_3)
modele3_fit_mcmc <- as.mcmc(modele3_fit)
gg_modele3 <- ggs(modele3_fit_mcmc)
ess_3 <- effectiveSize(modele3_fit_mcmc)

```

Nous allons regarder la convergence des différents paramètres estimés.

```

vec_par <- c("a0", "b0", "tau")
vec_nom <- c("a<sub>0</sub>", "b<sub>0</sub>", "&sigma;<sup>2</sup>")
for (i in vec_par) {
  assign(x = paste0("plot_", i),
         value = ggplot(gg_modele3 %>%
                           filter(Parameter == i) %>%
                           mutate(value = ifelse(Parameter == "tau", 1 /
value, value)),
                     aes(x = Iteration, y = value)) +
    geom_line() +
    scale_x_continuous(labels = scales::comma_format(), lim = c(0,
31000)) +
    labs(x = "Itération",
         y = paste0(vec_nom[match(i, vec_par)])),

```

```
            title = paste0("Traceplot de l'estimation de ",
vec_nom[match(i, vec_par)], " par MCMC"),
            subtitle = paste0("Initialisation de ", vec_nom[match(i,
vec_par)], " à 1")) +
            theme(plot.title = element_markdown()))
}
plot_a0 / plot_b0 / plot_tau

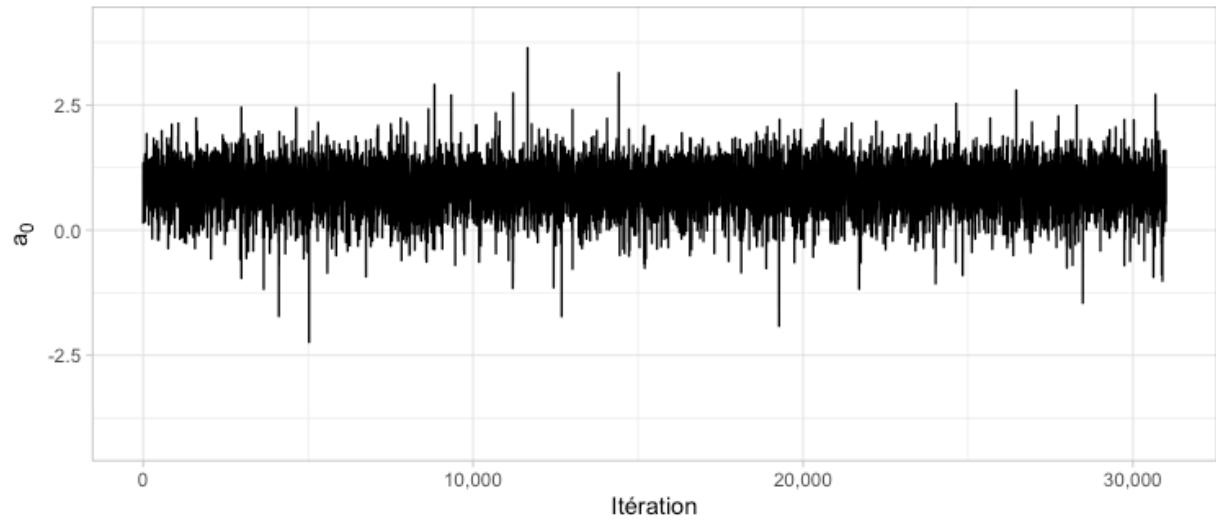
## Warning: Removed 268000 row(s) containing missing values (geom_path).

## Warning: Removed 268000 row(s) containing missing values (geom_path).

## Warning: Removed 268000 row(s) containing missing values (geom_path).
```

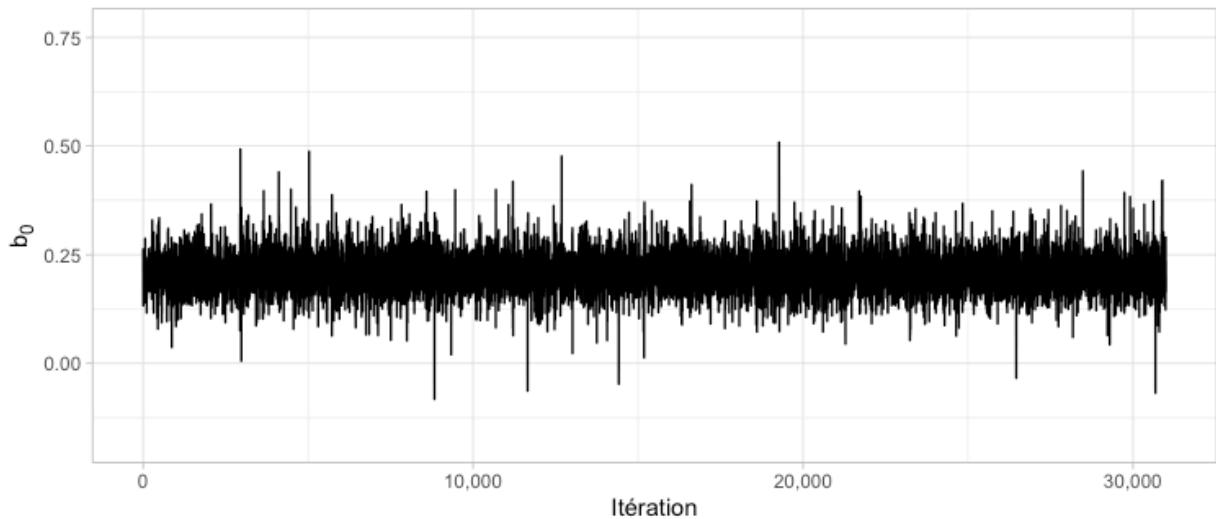
Traceplot de l'estimation de a_0 par MCMC

Initialisation de a_0 à 1



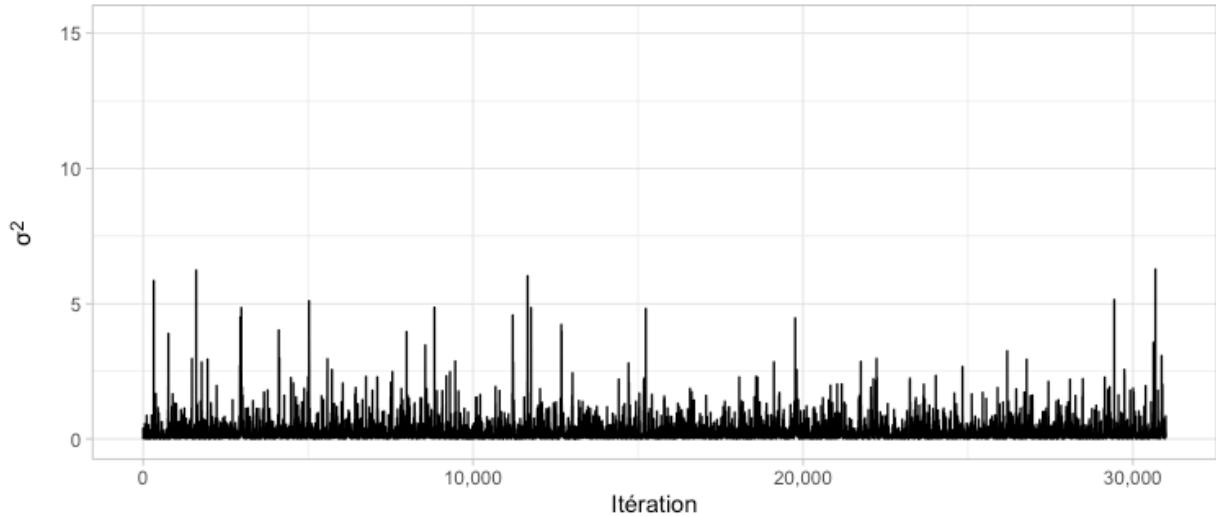
Traceplot de l'estimation de b_0 par MCMC

Initialisation de b_0 à 1



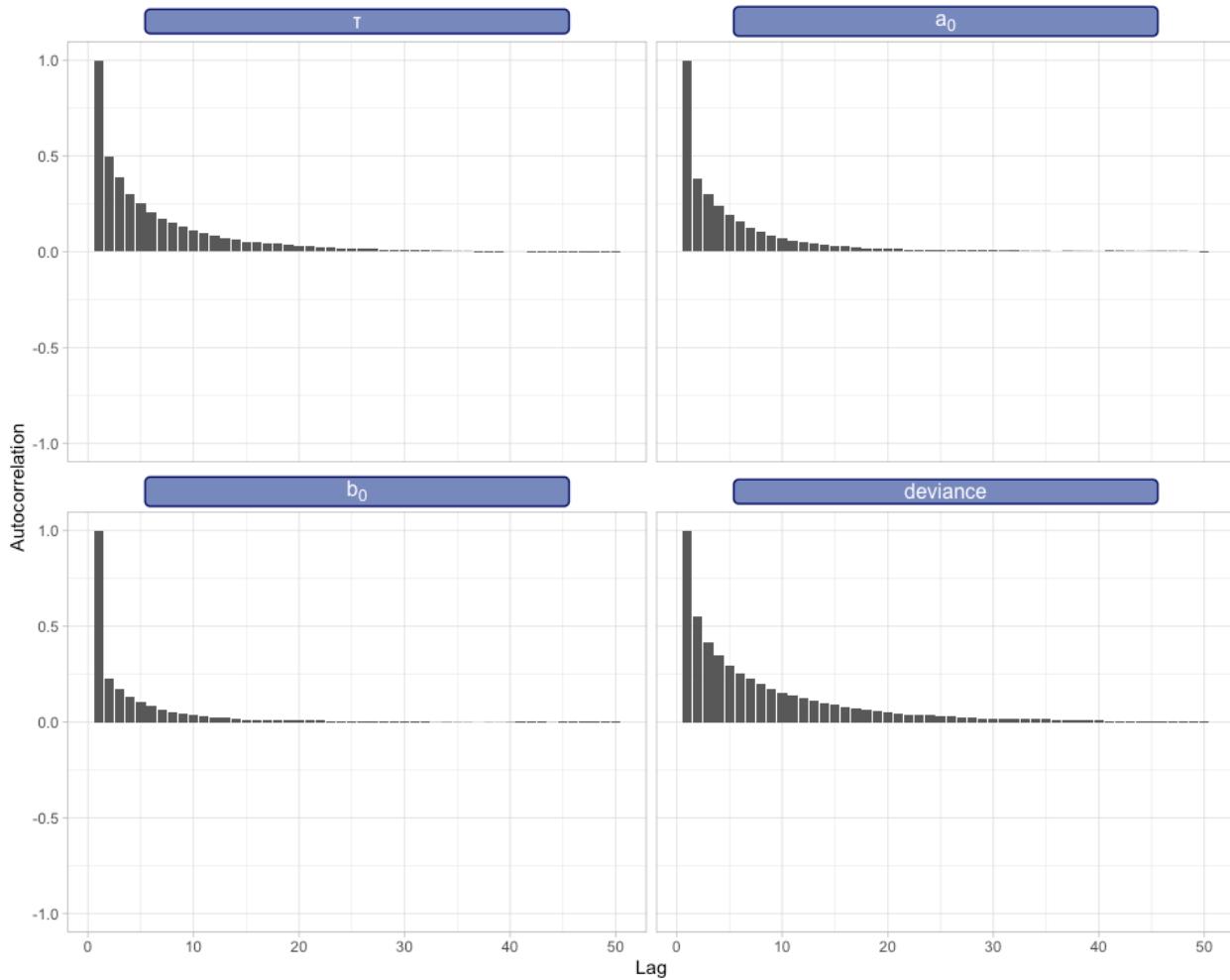
Traceplot de l'estimation de σ^2 par MCMC

Initialisation de σ^2 à 1



Au niveau des traceplot, les valeurs de a_0 et b_0 tournent autour d'une valeur : un peu moins de 1 pour a_0 et 0.2 pour b_0 . Il y a un peu plus de variations que pour les modèles 1 et 2, mais cela ne s'écarte pas trop. En revanche, pour σ^2 , beaucoup de valeurs sont plus dispersées.

```
ggs_autocorrelation(gg_modele3 %>%
  mutate(Parameter = case_when(Parameter == "a0" ~
    "a<sub>0</sub>",
    Parameter == "b0" ~
    "b<sub>0</sub>",
    Parameter == "tau" ~
    "&tau;",
    TRUE ~
    as.character(Parameter))))
```



On voit, sur le graphique des auto-corrélation qu'on a des auto-corrélations importantes, ce qui est confirmé par l'effective sample size qui est autour de 5000 pour les 3 paramètres estimés ($56714/94191/42522$), ce qui est bien inférieur aux 299000 itérations réellement faites. Nous allons donc mettre un thin à 10 afin d'essayer de casser ces auto-corrélations, tout en augmentant le nombre total d'itérations, en espérant que la variabilité d'estimation des paramètres sera réduite.

```

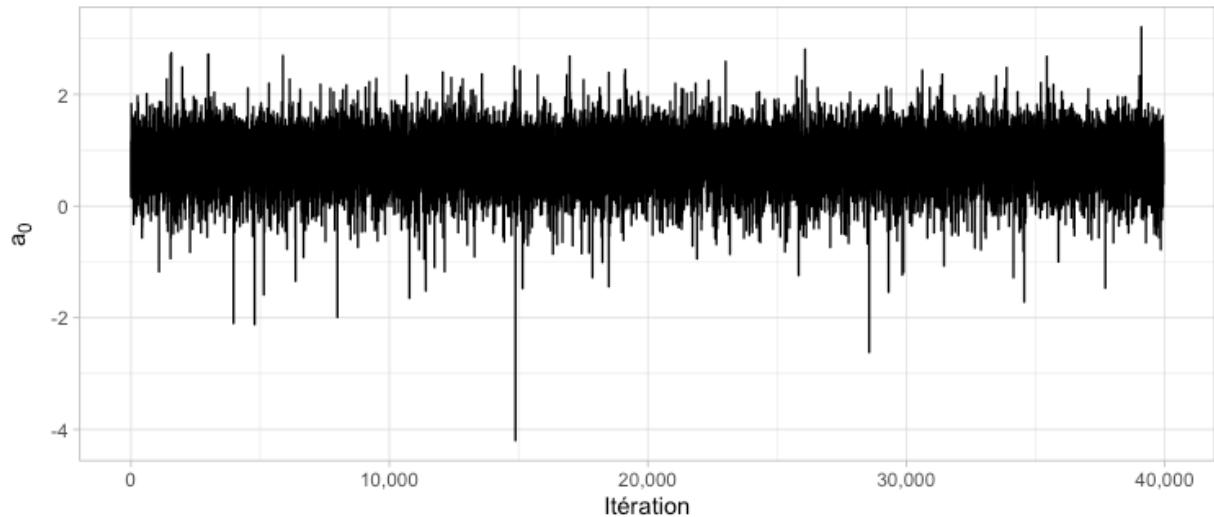
n_iter3 <- 40000
n_thin3 <- 20
set.seed(1993)
modele3_fit_thin_ch1 <- jags(
  data = donnees,
  inits = inits_modele3,
  parameters.to.save = parametres_modele3,
  n.chains = length(inits_modele3),
  n.iter = n_iter3 * n_thin3,
  n.burnin = n_burn3,
  n.thin = n_thin3,
  model.file = modele_3)
modele3_fit_thin_ch1_mcmc <- as.mcmc(modele3_fit_thin_ch1)
gg_modele3_thin_ch1 <- ggs(modele3_fit_thin_ch1_mcmc)
ess_3_thin_ch1 <- effectiveSize(modele3_fit_thin_ch1_mcmc)

vec_par <- c("a0", "b0", "tau")
vec_nom <- c("a0", "b0", "&sigma;2")
for (i in vec_par) {
  assign(x = paste0("plot_", i),
         value = ggplot(gg_modele3_thin_ch1 %>%
                           filter(Parameter == i) %>%
                           mutate(value = ifelse(Parameter == "tau", 1 /
value, value)),
                     aes(x = Iteration, y = value)) +
    geom_line() +
    scale_x_continuous(labels = scales::comma_format()) +
    labs(x = "Itération",
         y = paste0(vec_nom[match(i, vec_par)]),
         title = paste0("Traceplot de l'estimation de ",
vec_nom[match(i, vec_par)], " par MCMC"),
         subtitle = paste0("Initialisation de ", vec_nom[match(i,
vec_par)], " à 1")) +
    theme(plot.title = element_markdown()))
}
plot_a0 / plot_b0 / plot_tau

```

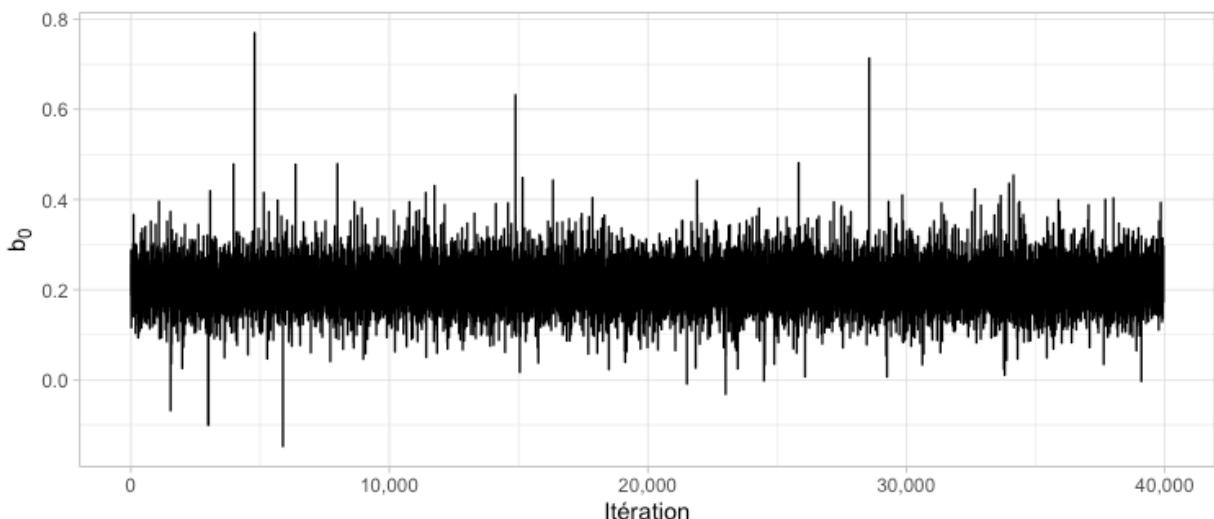
Traceplot de l'estimation de a_0 par MCMC

Initialisation de a_0 à 1



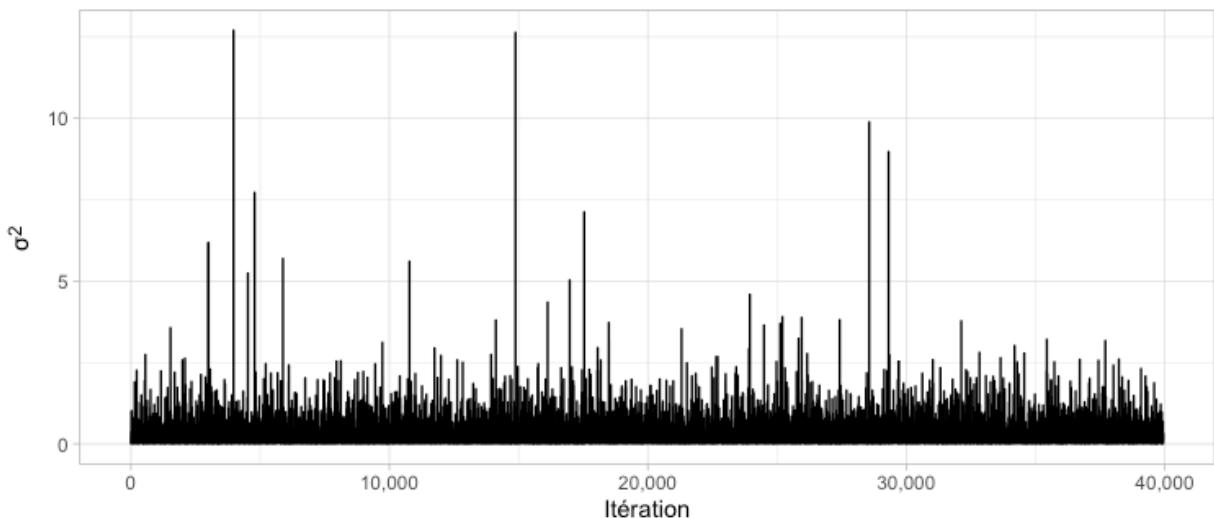
Traceplot de l'estimation de b_0 par MCMC

Initialisation de b_0 à 1



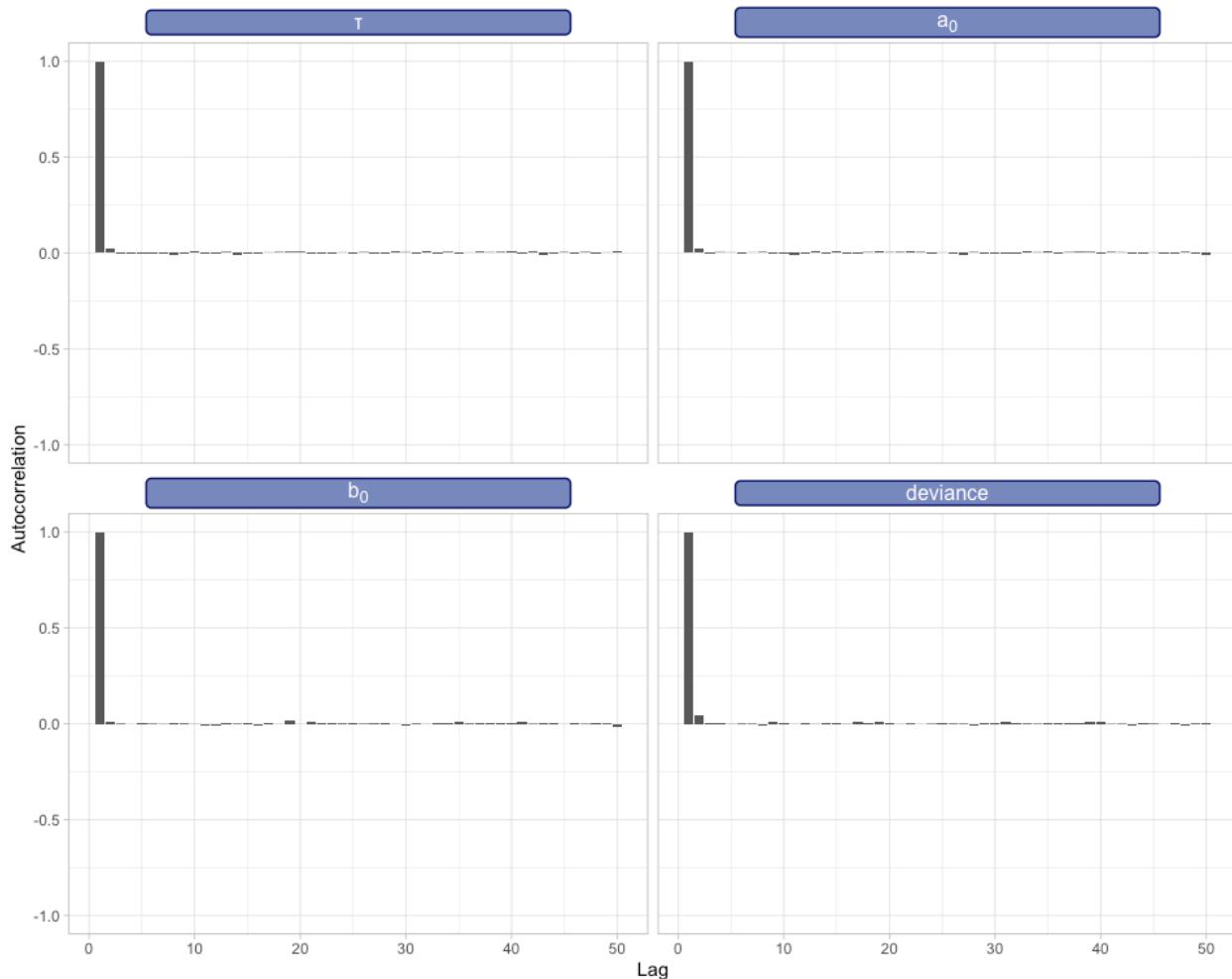
Traceplot de l'estimation de σ^2 par MCMC

Initialisation de σ^2 à 1



Nous avons tenté de faire un thin de 10, mais cela ne nous semblait pas suffisant sur le plot des auto-corrélations. Nous avons donc augmenté notre thin à 20. Sur le traceplot pour chaque paramètre estimé, les valeurs restent autour des mêmes zones avec un peu de variabilité résiduelle, mais nous considérons que ce la a bien convergé.

```
ggs_autocorrelation(gg_modele3_thin_ch1 %>%
  mutate(Parameter = case_when(Parameter == "a0" ~
    "a<sub>0</sub>",
    Parameter == "b0" ~
    "Parameter == "b0" ~
    "&tau;;",
    TRUE ~
    as.character(Parameter))))
```



Sur le plot des auto-corrélations, nous remarquons que nos itérations sont ben indépendantes les unes des autres. Cela est corroboré par les effective sample sizes pour a_0 , b_0 , la déviance et τ (38351/39318/36632/37854) qui sont proches des 39950 itérations effectuées.

Pour nous assurer de la bonne convergence de nos paramètres, nous avons tenté plusieurs initiations des 3 paramètres. Nous avons initié a_0 et b_0 à -5, 0 et 5, et τ à 0.05, 0.5, 1 et 2.

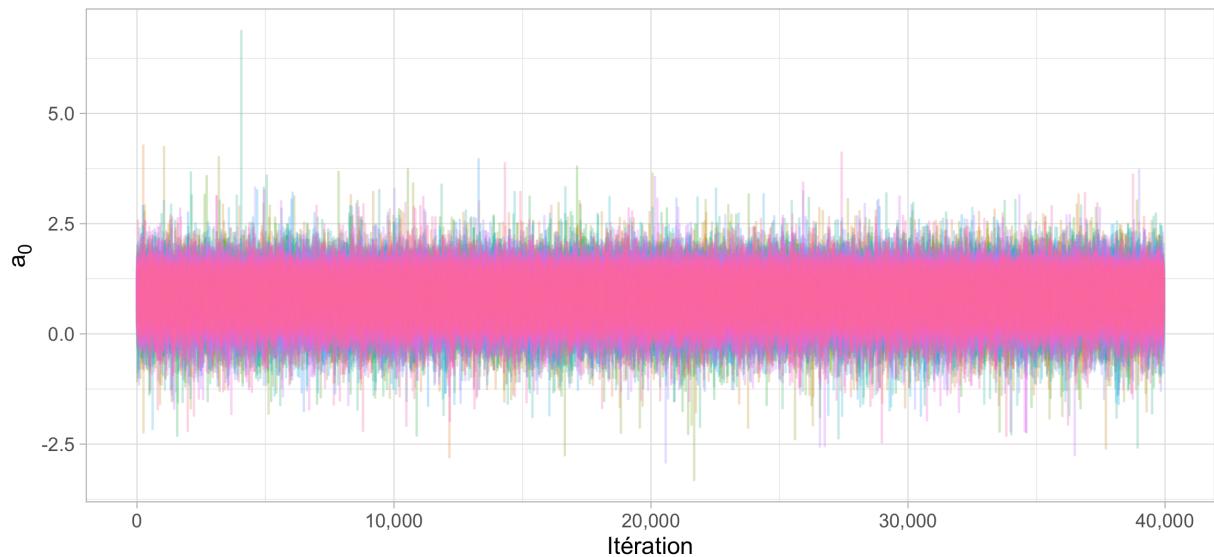
(Les codes ont été commenté dans le Rmd du fait du temps pour obtenir le modèle et les graphes. Nous avons sauvegardé des objets R et des images pour rendre les résultats.)

```
# grille <- expand.grid(a0 = c(5, 0, -5), b0 = c(5, 0, -5), tau = c(0.05, 0.5, 1, 2)) %>%
#   as.data.frame()
# inits_modele3_mult <- list()
# for (i in seq_len(nrow(grille))) {
#   inits_modele3_mult[[i]] <- list(a0 = grille[i, "a0"], b0 = grille[i, "b0"], tau = grille[i, "tau"])
# }
# set.seed(1993)
# modele3_fit_thin <- jags(
#   data = donnees,
#   inits = inits_modele3_mult,
#   parameters.to.save = parametres_modele3,
#   n.chains = length(inits_modele3_mult),
#   n.iter = n_iter3 * n_thin3,
#   n.burnin = n_burn3,
#   n.thin = n_thin3,
#   model.file = modele_3)
# save(modele3_fit_thin,
#       file = "modele3_thin.Rdata")
load("modele3_thin.Rdata")
# modele3_fit_thin_mcmc <- as.mcmc(modele3_fit_thin)
# gg_modele3_thin <- ggs(modele3_fit_thin_mcmc)
# ess_3_thin <- effectiveSize(modele3_fit_thin_mcmc)

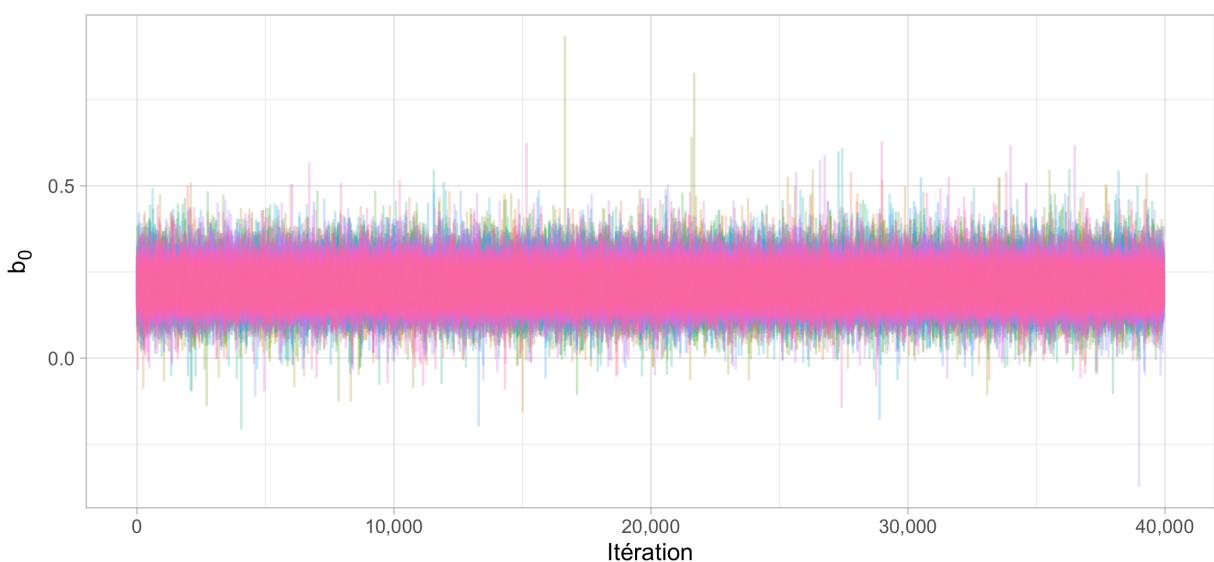
# vec_par <- c("a0", "b0", "tau")
# vec_nom <- c("a<sub>0</sub>", "b<sub>0</sub>", "&sigma;<sup>2</sup>")
# for (i in vec_par) {
#   assign(x = paste0("plot_", i),
#          value = ggplot(gg_modele3_thin %>%
#                           filter(Parameter == i) %>%
#                           mutate(value = ifelse(Parameter == "tau", 1 /
# value, value)),
#          aes(x = Iteration, y = value, color =
# as.factor(Chain))) +
#     geom_line(show.Legend = FALSE, alpha = 0.3) +
#     scale_x_continuous(labels = scales::comma_format()) +
#     Labs(x = "Itération",
#          y = paste0(vec_nom[match(i, vec_par)]),
#          title = paste0("Traceplot de l'estimation de ",
# vec_nom[match(i, vec_par)], " par MCMC avec 36 chaînes")) +
#     theme(plot.title = element_markdown()))
# }
# plot_abt <- plot_a0 / plot_b0 / plot_tau
```

```
# save(plot_a0, plot_b0, plot_tau, plot_abt,
#       file = "donnee_graphiques.Rdata")
# ggsave(filename = "plot_abt_36c.png",
#         plot = plot_abt,
#         device = "png",
#         width = 8,
#         height = 12)
```

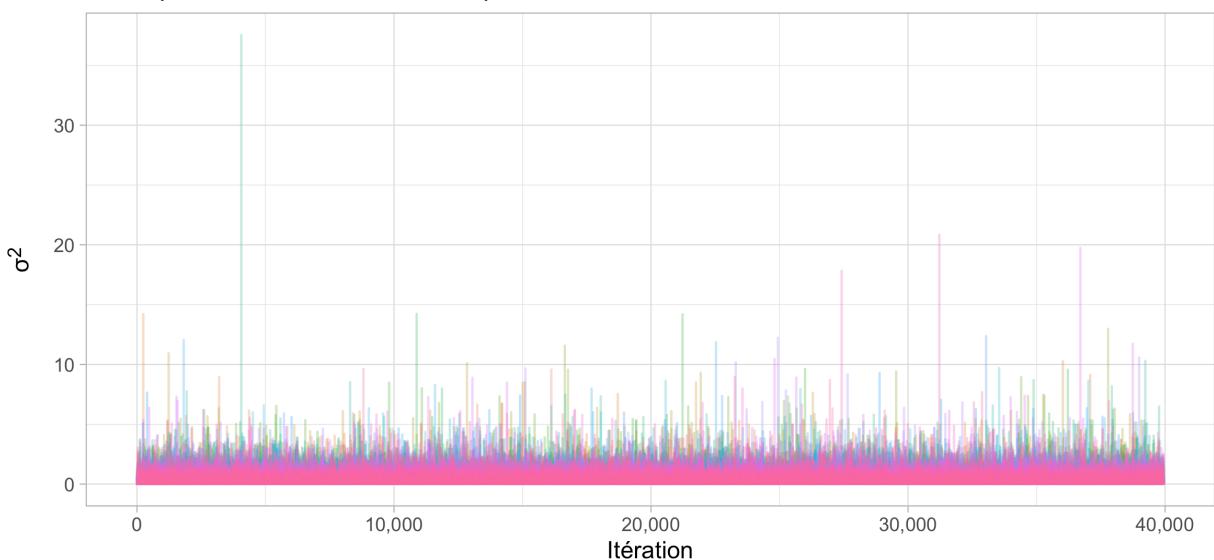
Traceplot de l'estimation de a_0 par MCMC avec 36 chaînes



Traceplot de l'estimation de b_0 par MCMC avec 36 chaînes

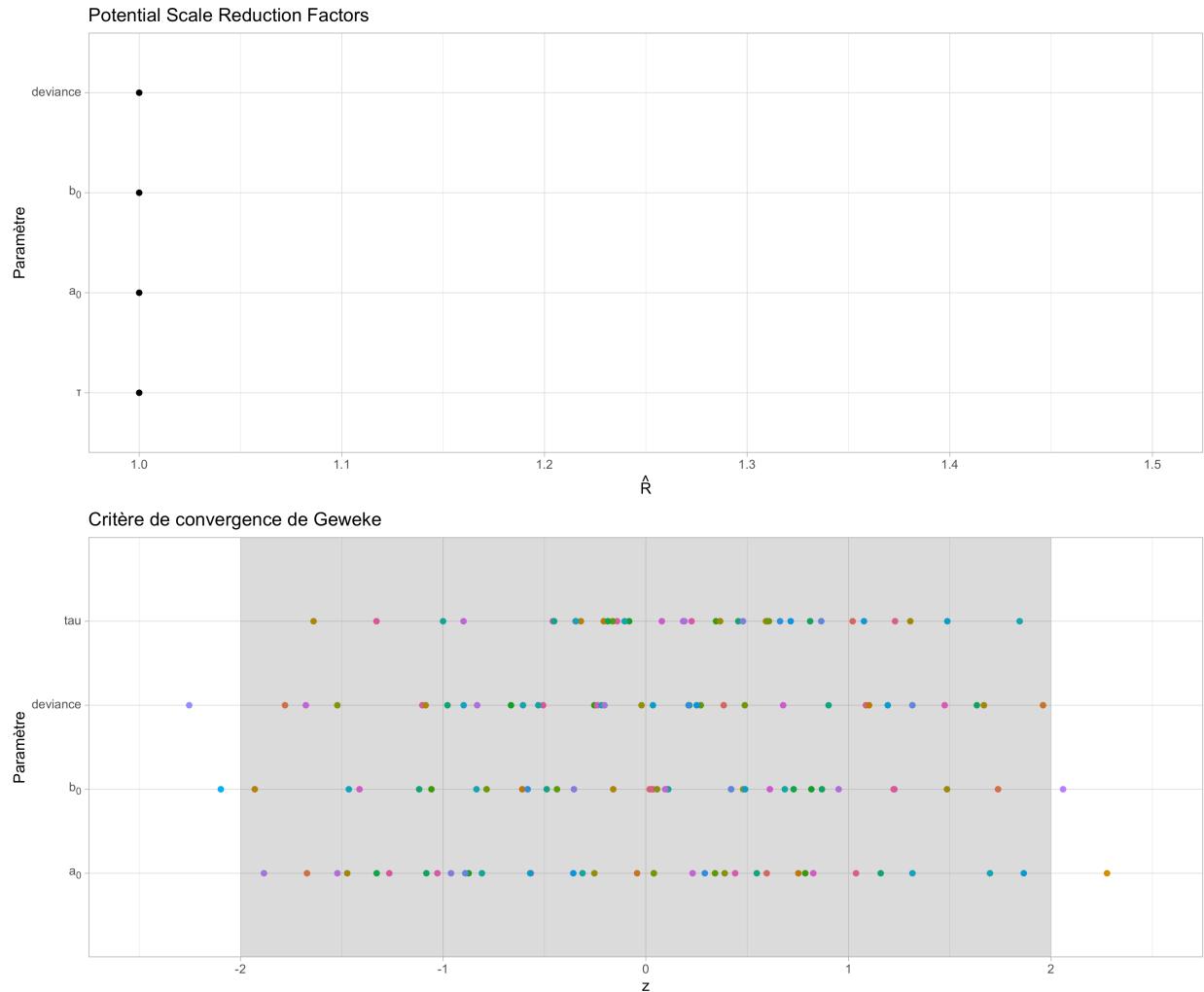


Traceplot de l'estimation de σ^2 par MCMC avec 36 chaînes



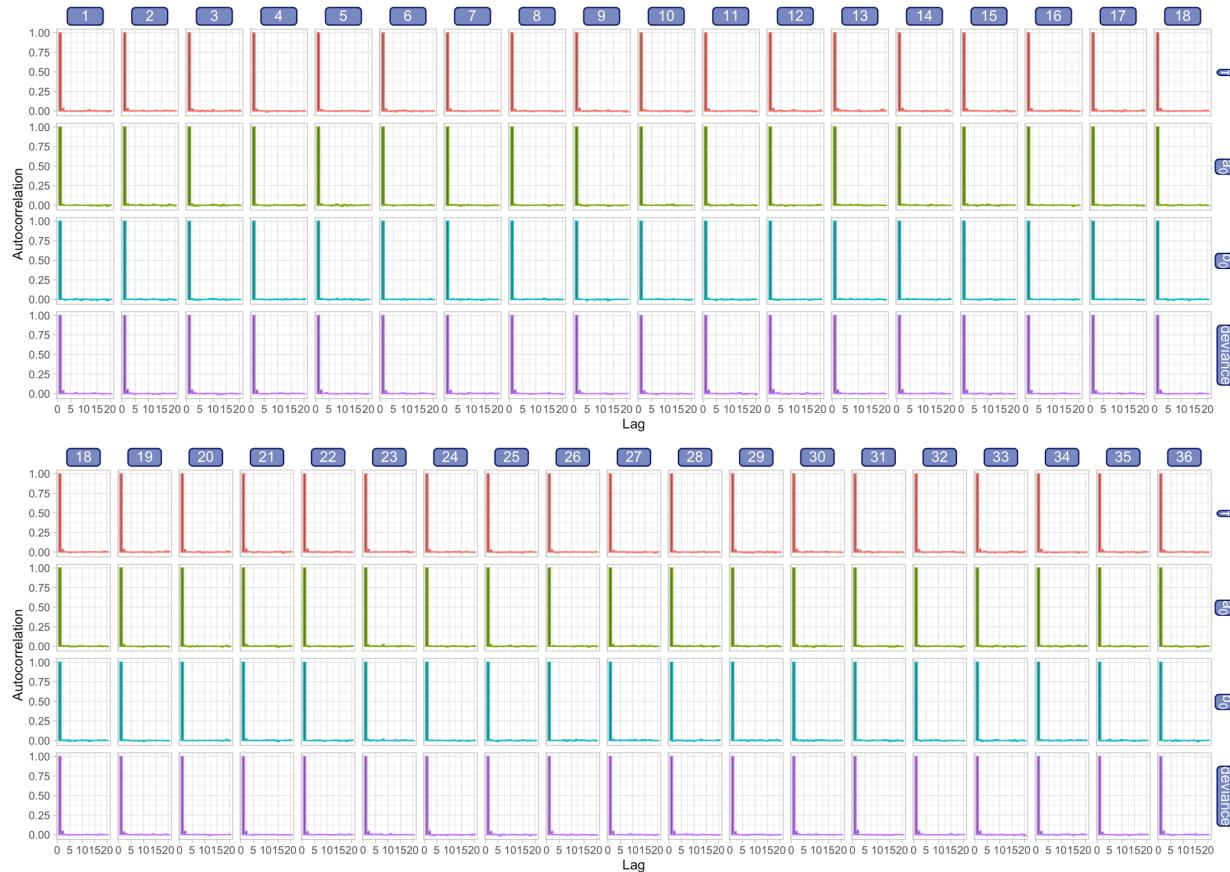
Sur les traceplots, les 3 paramètres ont l'air de converger vers une valeur identique à ce qui était vu sur le modèle pour 1 chaîne. La valeur de $\sigma^2 = \frac{1}{\tau}$ est sujette à plus de variabilité cependant mais ne s'écarte pas trop. Les critères de convergence de Geweke et Gelman nous confirme cette convergence. Le \hat{R} de Gelman est à 1, et les critère de convergence de Geweke reste entre -2DS et 2DS pour la quasi-totalité des chaînes. Nous considérons donc que le modèle a bien convergé.

```
# gelman3 <- ggs_Rhat(gg_modele3_thin %>% mutate(Parameter =
  case_when(Parameter == "aθ" ~ "a<sub>θ</sub>",
#                                         Parameter == "bθ" ~
# "b<sub>θ</sub>",
#                                         Parameter == "tau" ~
# "&tau;",
#                                         TRUE ~
#   as.character(Parameter))) +
#   Labs(y = "Paramètre",
#         x = expression(hat("R")) +
#   theme(axis.text.y = element_markdown())
# geweke3 <- ggs_geweke(gg_modele3_thin %>% mutate(Parameter =
  case_when(Parameter == "aθ" ~ "a<sub>θ</sub>",
#                                         Parameter == "bθ" ~
# "b<sub>θ</sub>",
#                                         TRUE ~
#   as.character(Parameter))) +
#   Labs(y = "Paramètre",
#         title = "Critère de convergence de Geweke") +
#   theme(axis.text.y = element_markdown(),
#         legend.position = "none")
# gelgew <- gelman3 / geweke3
# ggsave(filename = "plot_gelman_geweke.png",
#        device = "png",
#        plot = gelgew,
#        height = 10, width = 12)
```



```
# autocor <- ggs_autocorrelation(gg_modele3_thin %>%
#                                     mutate(Parameter = case_when(Parameter == "a0" ~
# "a<sub>0</sub>",
#                                     Parameter == "b0" ~
# "b<sub>0</sub>",
#                                     Parameter == "tau" ~
# "&tau;",
#                                     TRUE ~
# as.character(Parameter))))
# plot_1 <- autocor$data %>%
#   filter(Lag < 21, Chain %in% c(1:18)) %>%
#   ggplot(aes(x = Lag, y = Autocorrelation, color = Parameter)) +
#   geom_col(show.legend = FALSE) +
#   facet_grid(Parameter ~ Chain)
# plot_2 <- autocor$data %>%
#   filter(Lag < 21, Chain %in% c(18:36)) %>%
#   ggplot(aes(x = Lag, y = Autocorrelation, color = Parameter)) +
#   geom_col(show.legend = FALSE) +
#   facet_grid(Parameter ~ Chain)
```

```
# plot_tot_autocor <- plot_1 / plot_2
# ggsave(plot = plot_tot_autocor,
#         filename = "plot_autocor.png",
#         device = "png",
#         width = 14, height = 10)
```



Le graphique des auto-corrélations nous montre quant à lui qu'il n'y a pas d'auto-corrélation résiduelle sur nos chaînes.

Les résultats pour les 2 modèles (1 chaîne vs 36 chaînes) sont présentés dans le tableau suivant. On peut remarquer que les résultats sont presque identiques, ce qui nous rassure pour prendre le résultat du modèle à 1 chaîne comme résultat principal. Par ailleur, on remarque que l'écart-type de τ est très élevé malgré le grand nombre d'itérations. Cela peut nous indiquer que la variable ε_i de chaque machine suivant une loi normale de même variance pour toutes les machines n'est pas forcément pertinente dans notre modèle. La variabilité n'est peut-être pas tout à fait identique entre chaque machine et cela pourrait empêcher l'estimation d'être précise.

```
modele3_fit_thin_ch1$BUGSoutput$summary[c(1:2, 4), 1:2] %>%
  as.data.frame() %>% rownames_to_column() %>%
  left_join(modele3_fit_thin$BUGSoutput$summary[c(1:2, 4), 1:2] %>%
    as.data.frame() %>% rownames_to_column(),
            by = "rowname") %>%
```

```

mutate(across(where(is.numeric), ~ round(.x, 3))) %>%
flextable() %>%
set_header_labels(rowname = "Paramètre estimé",
                  mean.x = "Estimation",
                  sd.x = "Ecart-Type",
                  mean.y = "Estimation",
                  sd.y = "Ecart-Type") %>%
add_header_row(values = c(" ", "1 chaîne", " ", "36 chaînes", " ")) %>%
merge_at(i = 1, j = 2:3, part = "header") %>%
merge_at(i = 1, j = 4:5, part = "header") %>%
theme_booktabs() %>%
align(i = 1:2, j = 1:5, align = "center", part = "header") %>%
autofit()

```

Paramètre estimé	1 chaîne		36 chaînes	
	Estimation	Ecart-Type	Estimation	Ecart-Type
a0	0.864	0.368	0.867	0.360
b0	0.209	0.036	0.209	0.035
tau	14.996	24.584	14.965	24.302

Nous prenons donc les résultats de notre modèle à 1 chaîne avec 800000 itérations dont 1000 itérations de burn-in et un thin de 20.

18. Question 3

Que vaut le nombre d'itérations pour les calculs? Que vaut le nombre d'itérations "effectif" ?

Le nombre d'itérations faites et conservées est de 39950.

Les nombres d'itérations effectives pour les différents paramètres estimés sont :

- pour a_0 38351;
- pour b_0 39318;
- pour τ 37854;
- pour la déviance 36632.

Ces nombres effectifs proches du nombre d'itérations gardées montrent l'indépendance de nos itérations.

19. Question 4

Donnez la moyenne a posteriori et l'intervalle de crédibilité à 95% de a0, b0 et tau.

```

resm3 <- summary(modele3_fit_thin_ch1_mcmc)[[1]] %>% as.data.frame()
resq3 <- summary(modele3_fit_thin_ch1_mcmc)[[2]] %>% as.data.frame()
ic3a <- paste0(round(resm3[["Mean"]][1], 2), "[", round(resq3[["2.5%"]][1],
2), ";", round(resq3[["97.5%"]][1], 2), "]")
ic3b0 <- paste0(round(resm3[["Mean"]][2], 2), "[", round(resq3[["2.5%"]][2],

```

```

2), ";", round(resq3[["97.5%"]][2], 2), "]")
ic3tau <- paste0(round(resm3[["Mean"]][4], 2), "[", round(resq3[["2.5%"]][4],
2), ";", round(resq3[["97.5%"]][4], 2), "]")
ic3sig <- paste0(round(1/resm3[["Mean"]][4], 2), "[",
round(1/resq3[["97.5%"]][4], 2), ";", round(1/resq3[["2.5%"]][4], 2), "]")

```

L'estimation de a_0 nous donne la moyenne et l'intervalle de crédibilité à 95% suivant : 0.86[0.08;1.53]. L'estimation de b_0 nous donne la moyenne et l'intervalle de crédibilité à 95% suivant : 0.21[0.14;0.28]. L'estimation de τ nous donne la moyenne et l'intervalle de crédibilité à 95% suivant : 15[1.03;78.32], soit un intervalle de crédibilité pour σ^2 de 0.07[0.01;0.97].

Nous avons tenté de représenter ces moyennes de lois de Poisson sur les données dont nous disposons. Les lignes "moyenne", "minimum" et "maximum" représentent les moyennes des lois de Poisson lorsque a_0 et b_0 sont respectivement toutes les 2 à leur valeur moyenne, à la borne inférieure ou supérieure de leur intervalle de crédibilité à 95%. De plus, autour de ces estimations de moyenne, nous avons représenté la variabilité inter-machine ε_i qui est assumée comme l'intervalle de confiance à 95% de la loi normale suivie par $\varepsilon_i(\mathcal{N}(0, \sigma^2))$ pour le σ^2 moyen estimé.

```

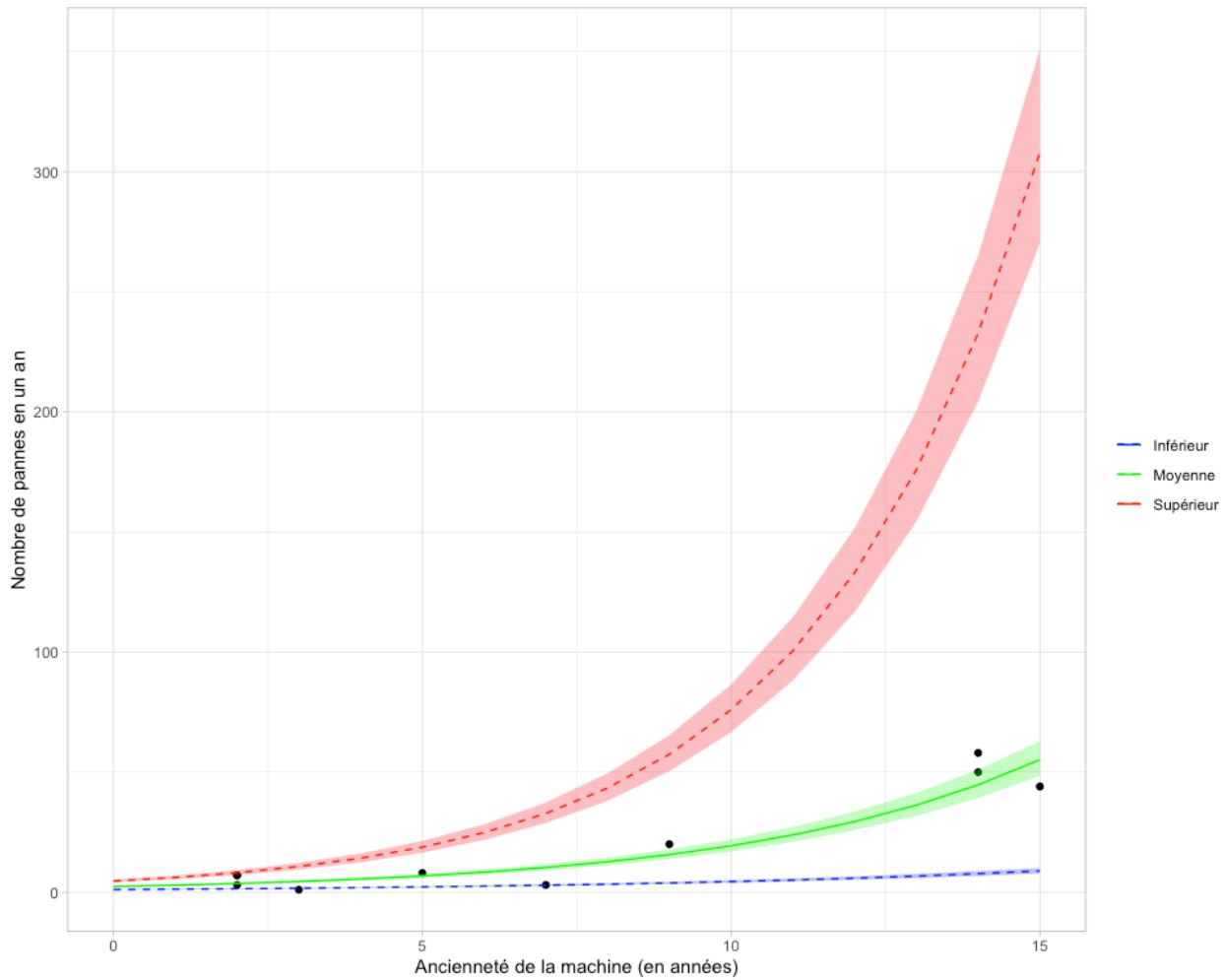
poissons_moy <- tibble(
  anciennete = 0:15,
  poiss_moymin = exp(0.08 + anciennete * 0.14),
  poiss_moymininf = exp(0.08 + anciennete * 0.14 + qnorm(0.025, mean = 0, sd
= 1/15)),
  poiss_moyminsup = exp(0.08 + anciennete * 0.14 + qnorm(0.975, mean = 0, sd
= 1/15)),
  poiss_moymean = exp(0.86 + anciennete * 0.21),
  poiss_moymeaninf = exp(0.86 + anciennete * 0.21 + qnorm(0.025, mean = 0, sd
= 1/15)),
  poiss_moymeansup = exp(0.86 + anciennete * 0.21 + qnorm(0.975, mean = 0, sd
= 1/15)),
  poiss_moymax = exp(1.53 + anciennete * 0.28),
  poiss_moymaxinf = exp(1.53 + anciennete * 0.28 + qnorm(0.025, mean = 0, sd
= 1/15)),
  poiss_moymaxsup = exp(1.53 + anciennete * 0.28 + qnorm(0.975, mean = 0, sd
= 1/15))
)
ggplot() +
  geom_point(data = sncf_machines, aes(anciennete, nb_pannes)) +
  geom_ribbon(data = poissons_moy, aes(x = anciennete, ymin =
poiss_moymeaninf, ymax = poiss_moymeansup), fill = "green", alpha = 0.3) +
  geom_line(data = poissons_moy, aes(x = anciennete, y = poiss_moymean, color
= "Moyenne")) +
  geom_ribbon(data = poissons_moy, aes(x = anciennete, ymin =
poiss_moymininf, ymax = poiss_moyminsup), fill = "blue", alpha = 0.3) +
  geom_line(data = poissons_moy, aes(x = anciennete, y = poiss_moymin, color
= "Inférieur"), linetype = "dashed") +
  geom_ribbon(data = poissons_moy, aes(x = anciennete, ymin =

```

```

poiss_moymaxinf, ymax = poiss_moymaxsup), fill = "red", alpha = 0.3) +
  geom_line(data = poissons_moy, aes(x = anciennete, y = poiss_moymax, color
= "Supérieur"), linetype = "dashed") +
  scale_color_manual(name = NULL,
                     values = c("blue", "green", "red")) +
  labs(x = "Ancienneté de la machine (en années)",
       y = "Nombre de pannes en un an")

```



Nous pouvons voir sur la représentation que le modèle englobe maintenant toutes les machines. Cependant, on remarque aussi que l'incertitude devient plus grande pour les bornes. La droite des moyennes des lois de Poissons "moyennes" a l'air de bien représenter nos données, mais la borne haute s'envole beaucoup plus haut que le dernier modéle à cause du coefficient b_0 plus haut.

De plus, tout comme le modèle 2, nous avons fait l'hypothèse que a_0 et b_0 suivent leur intervalle de crédibilité à 95% de façon parallèle, et nous n'avons pris le τ que moyen.

20. Question 5

Que vaut le DIC ? Que vaut l'estimation de la complexité du modèle ? Vous semble-t-elle logique ?

```
dic3 <- round(modele3_fit_thin_ch1$BUGSoutput$DIC, 4)
complexite3 <- round(modele3_fit_thin_ch1$BUGSoutput$pD, 4)
```

Le DIC du modèle vaut 68.7502.

Pour estimer la complexité du modèle, le pD est estimé et représente le nombre effectif de paramètres estimés. Pour notre modèle, il vaut 13.463.

Nous avions 3 paramètres à estimer : a_0 , b_0 et τ , donc nous nous attendions à avoir environ un pD de 3. Mais comme évoqué plus tôt, le paramètre τ a un écart-type élevé et nous avions émis l'hypothèse que les machines n'avaient peut-être pas toutes la même variabilité inter-individuelle. Ici, on a l'impression que cela va dans le sens de cette hypothèse, avec un nombre effectif de paramètres estimés environ augmenté de 10 par rapport à ce que nous pensions trouver sur un échantillon de 10 machines. Nous pensons donc que le modèle spécifié n'est pas le plus adapté aux données.

21. Question 6

D'après le DIC, quel modèle choisissez-vous entre M1, M2 et M3 ?

Le DIC le plus bas est celui du modèle 2 donc c'est le modèle que nous choisissons au final.

De plus, même si le DIC du modèle 3 est voisin, il aurait pu être encore plus proche de celui du modèle 2 voire un peu inférieur que nous aurions choisi le modèle 2 car le modèle est beaucoup plus complexe comme l'indique le nombre effectif de paramètres estimés. On gagne un peu en qualité de représentativité des données mais au prix d'un modèle trop complexe.