Devoir Méthodes bayésiennes : session 1

Benoit Gachet, Guillaume Mulier

19/12/2020

Table of Contents

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 :

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

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

1. Question 1

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

$$\begin{aligned} E(y_i|a) &= E(\mathcal{P}ois(\lambda)) \\ &= \lambda \\ &= e^a \end{aligned}$$

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.

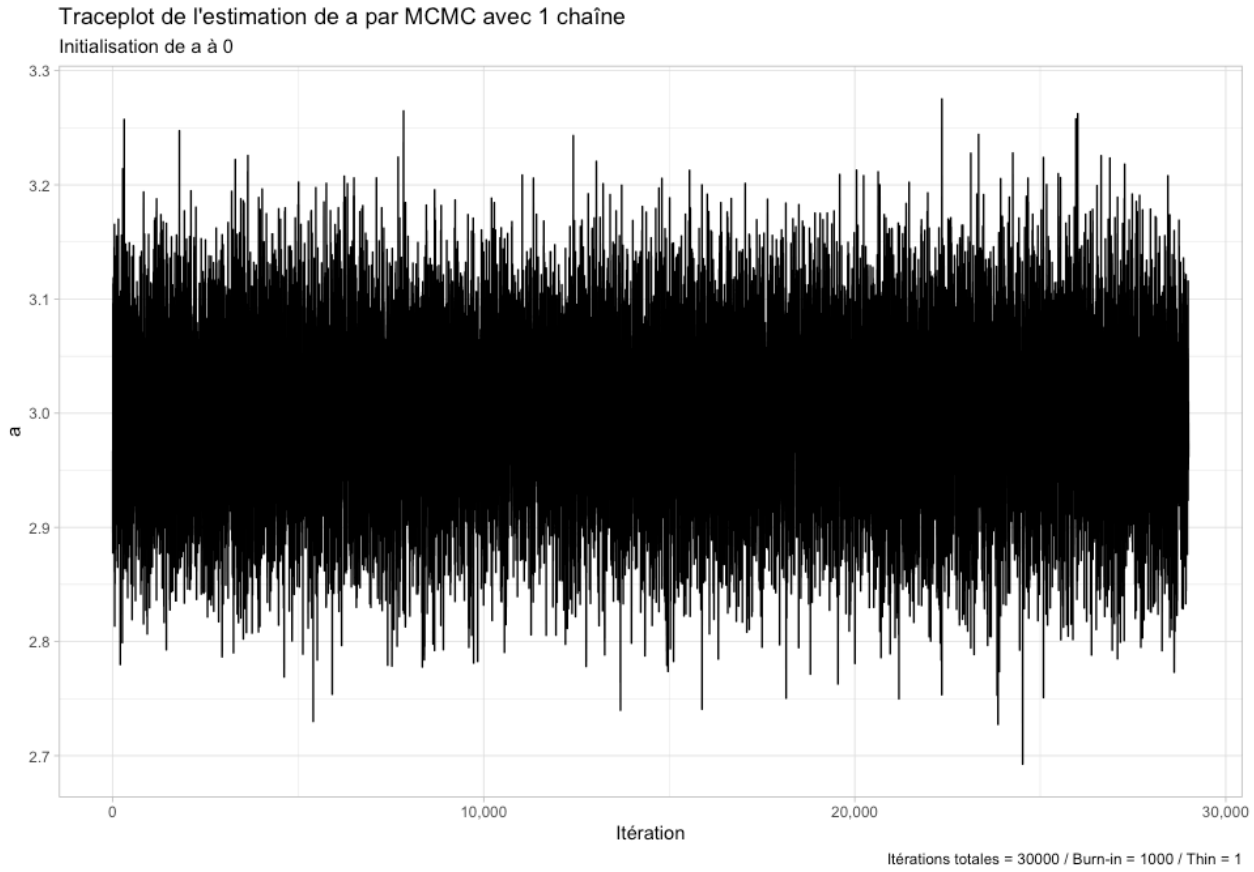
Réalisation du modèle avec JAGS en prenant 30000 itérations avec 1000 itérations de burn-in au début, en gardant toutes les itérations :

```
# 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)
```

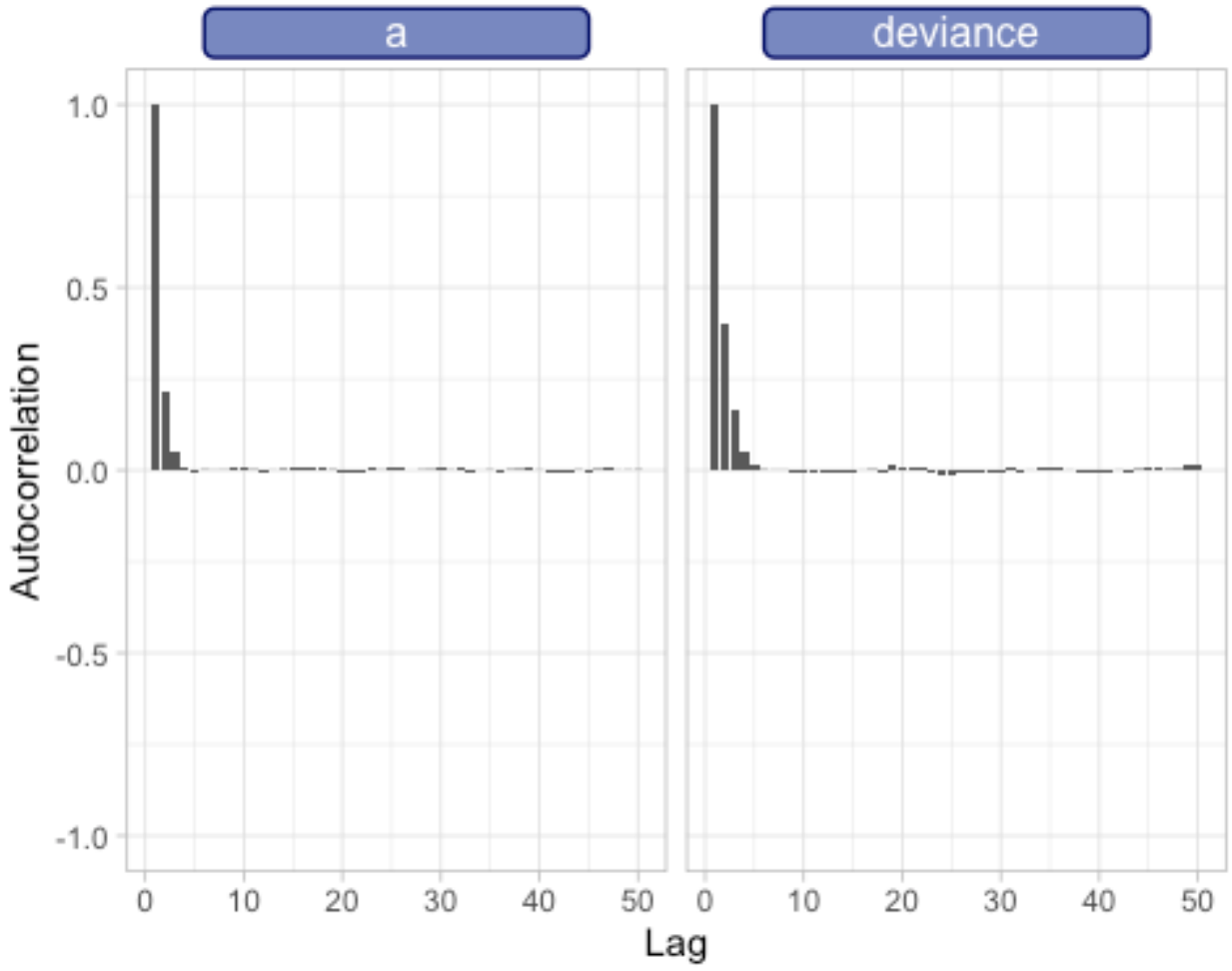
On regarde si le paramètre a estimé a bien convergé.

```
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'écarter 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 jusquà 8. Cela est confirmé par le nombre d'itérations effectives qui est franchement diminué par rapport aux 30000 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,
```

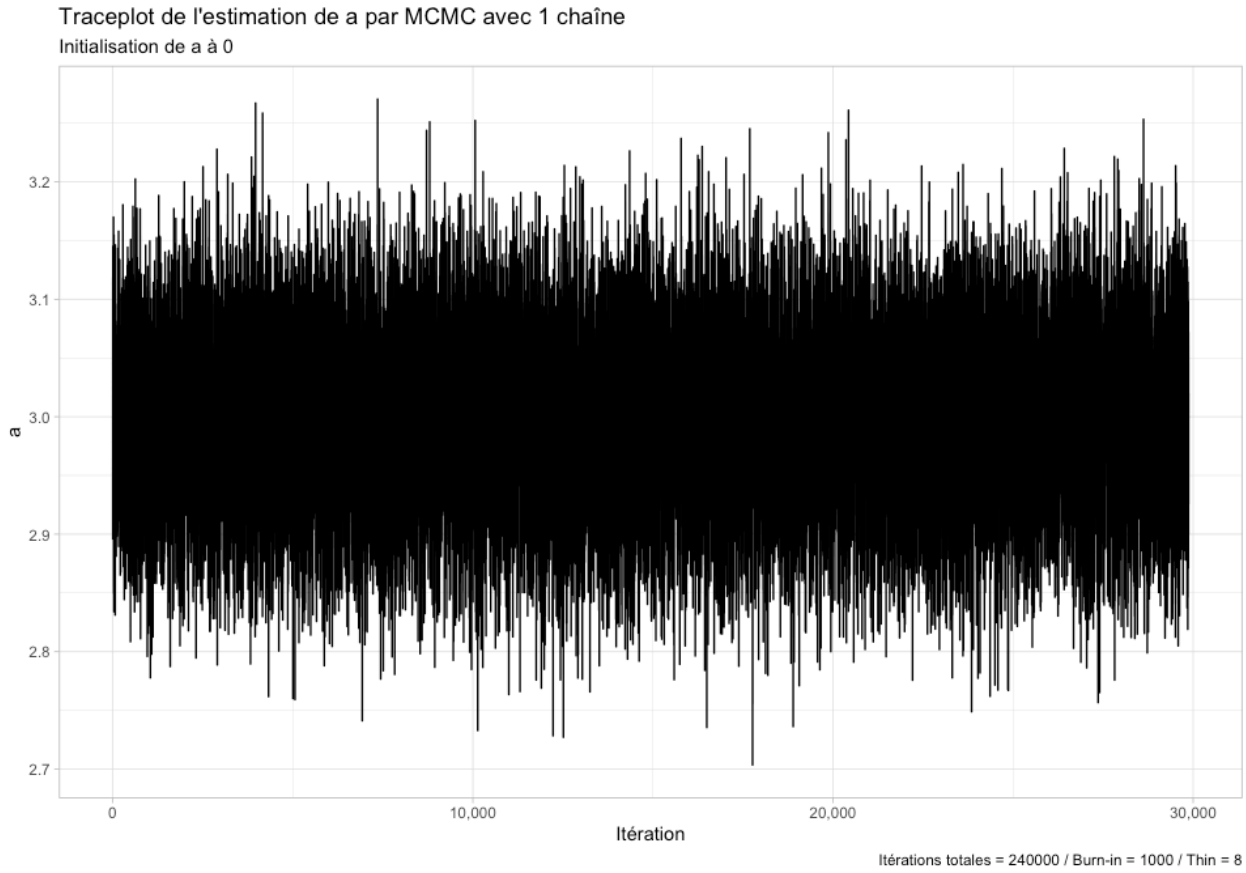
```
                          parameters.to.save = parametres_modele1,
                          n.chains = length(inits_modele1),
                          n.iter = n_iter * n_thin,
                          n.burnin = n_burn,
                          n.thin = n_thin,
                          model.file = 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()
```

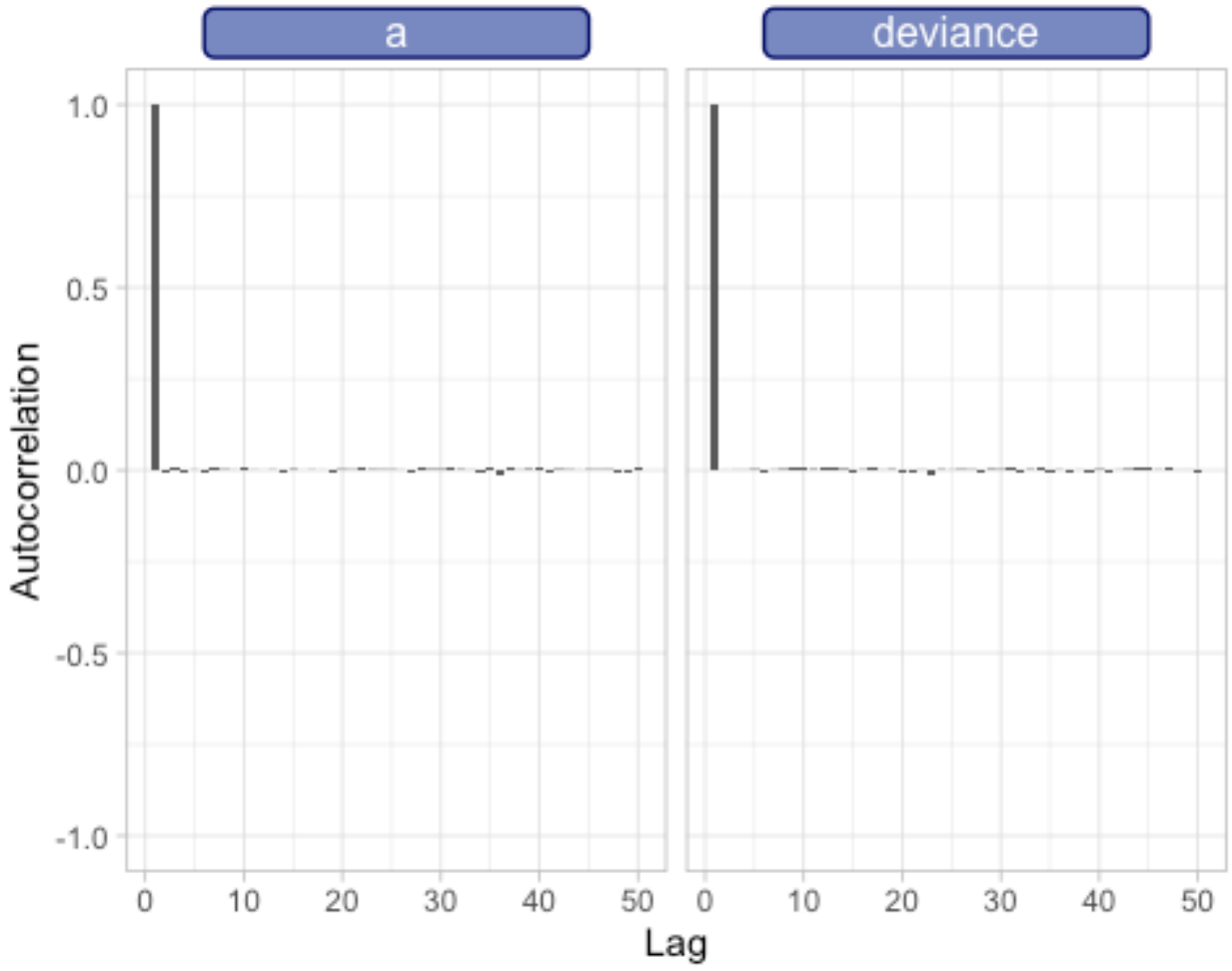
Paramètre estimé	Estimation	Ecart-Type
a	2.998	0.071

```
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")
```



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)
```

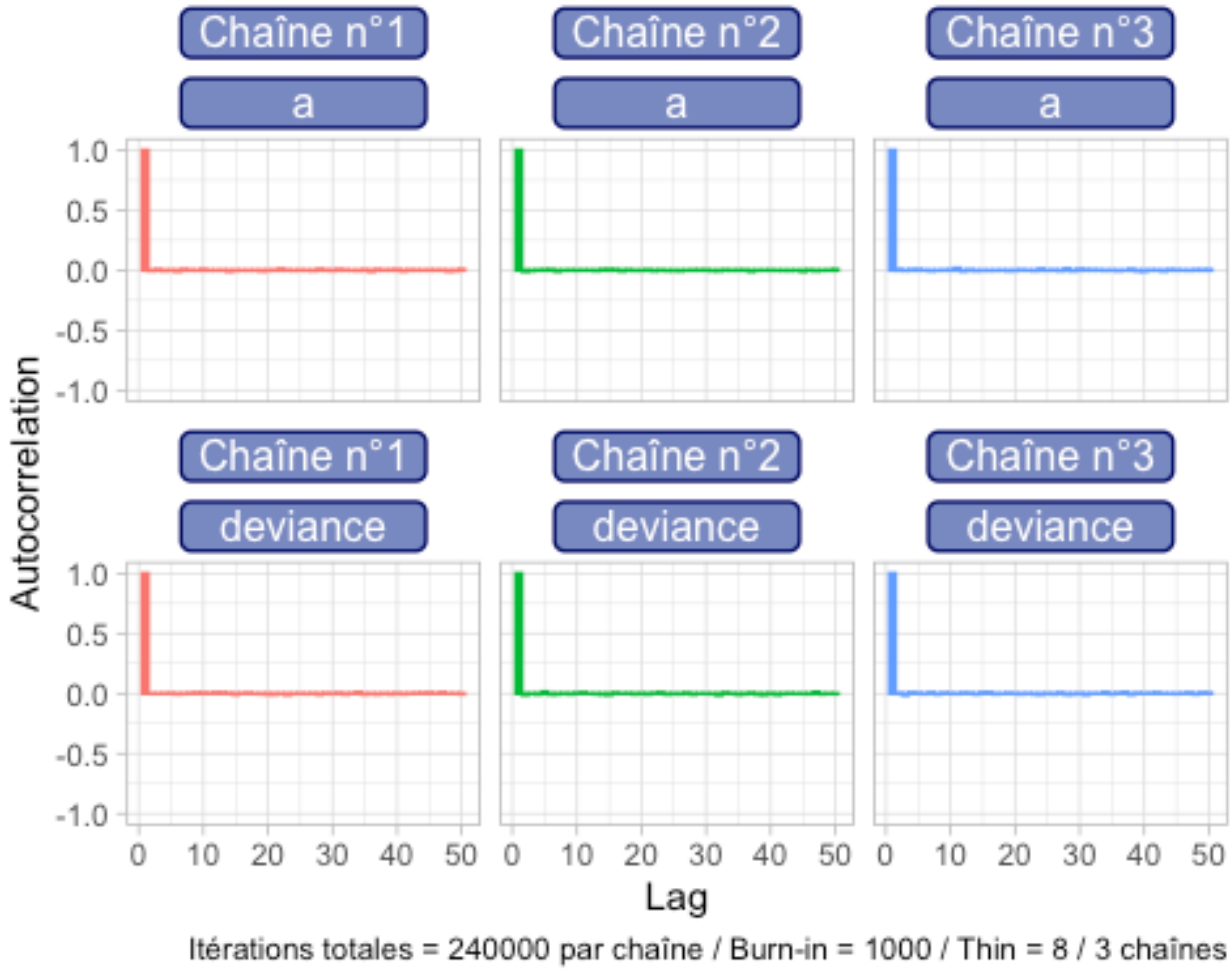


Sur le graphique d'auto-corrélations, on voit que le problème a été réglé et que maintenant il n'y a plus d'auto-corrélation pour le paramètre a estimé. De plus, le nombre d'itérations effectives est maintenant autour des 29875 itérations faites : 29875 pour l'estimation de a et 29875 pour l'estimation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

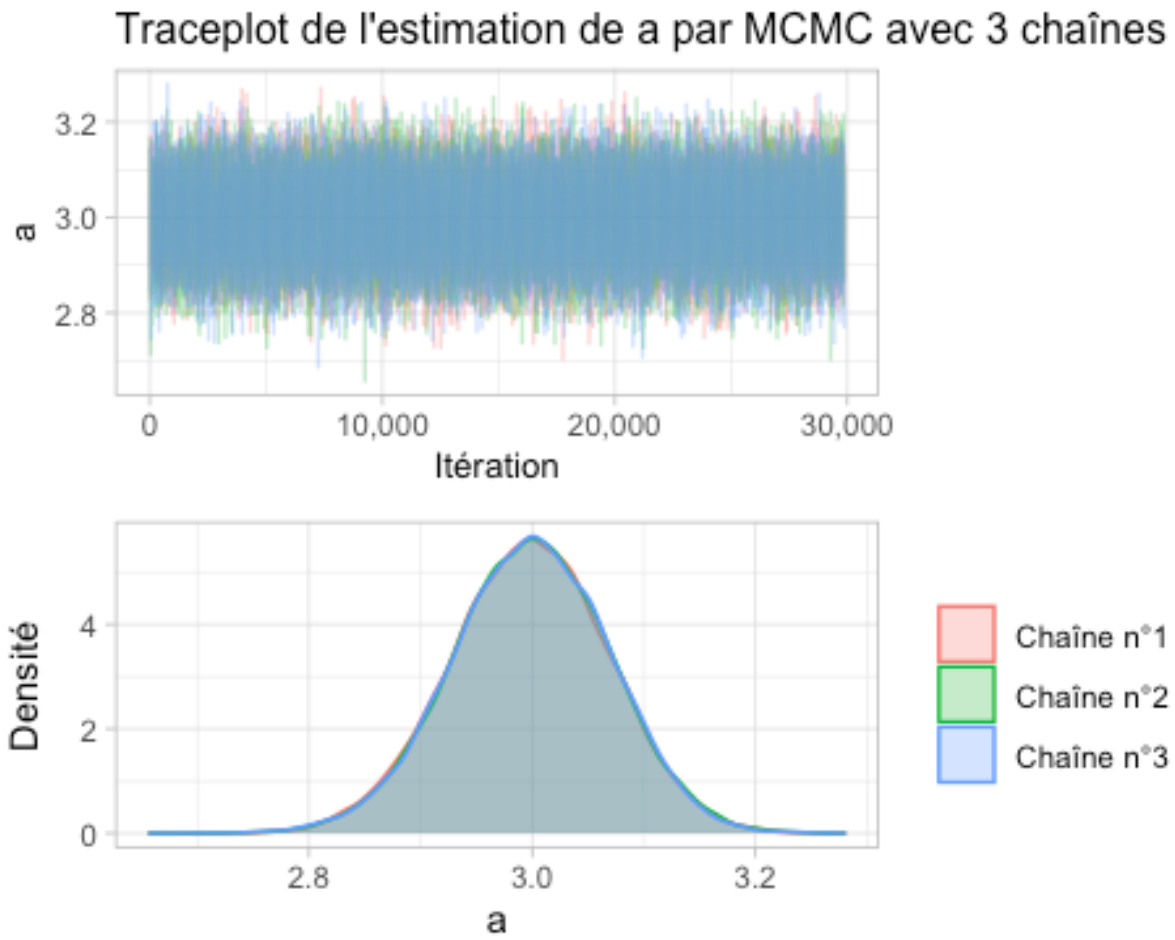
```
inits_2 <- list("a" = 5)
inits_3 <- list("a" = -5)
inits_modele1_mult <- list(inits_1, inits_2, inits_3)
set.seed(1993)
modele1_fit_mult <- jags(data = donnees,
                         inits = inits_modele1_mult,
                         parameters.to.save = parametres_modele1,
                         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")
```



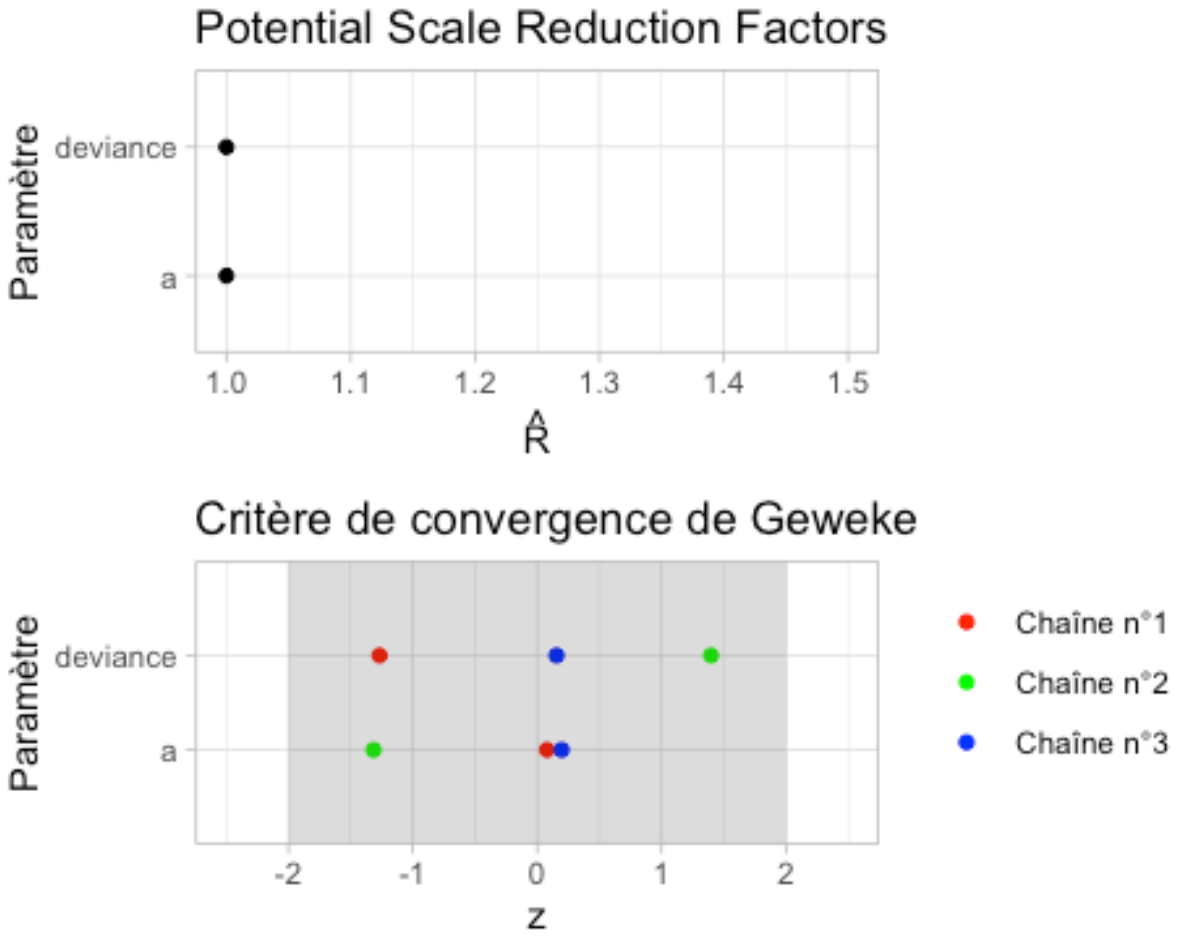
Pour les 3 chaînes, on ne voit pas d'auto-corrélation dans notre modèle. Aussi, le nombre d'itérations effectives est resté autour des 89625 itérations faites : 89625 pour l'estimation de a et 89625 pour l'estimation de la déviance du modèle (à noter que pour la déviance, l'effective sample size dépasse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

```r
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
```



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
```



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-corrélations négatives pour notre modèle, mais nous avons considéré que cela montrait que les itérations étaient indépendantes.

4. Question 4

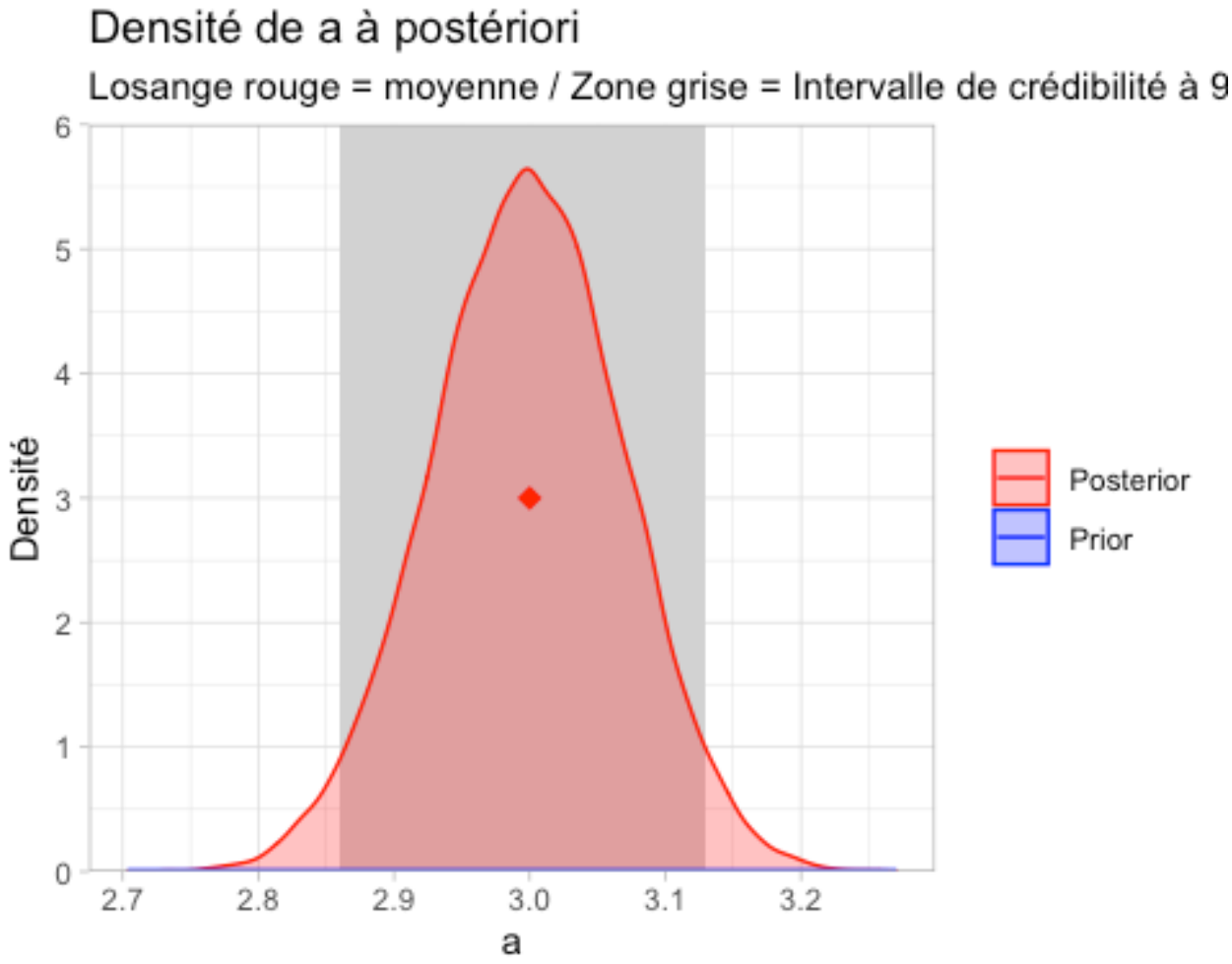
Donnez la moyenne a posteriori et l'intervalle de crédibilité à 95% de a.

```
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.

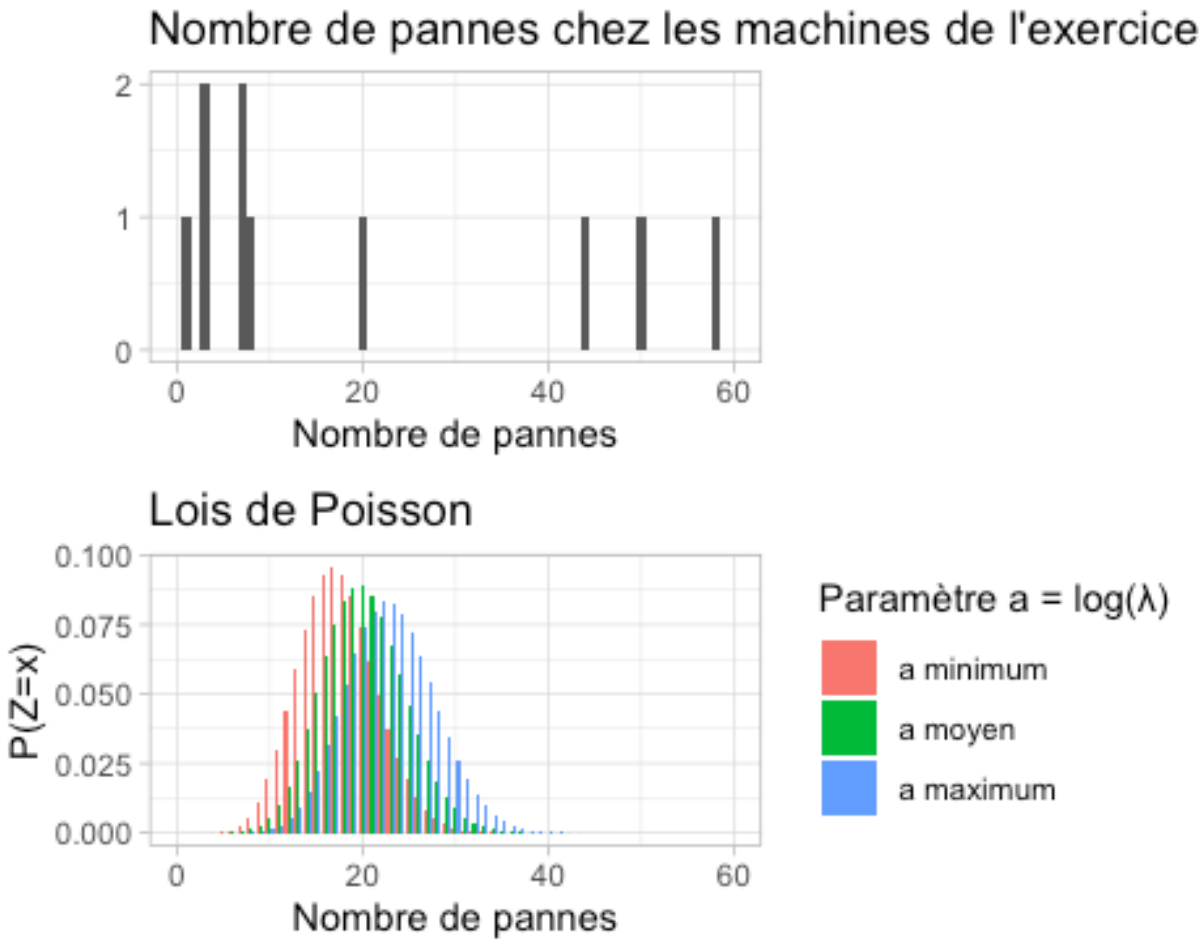
```
ggplot(gg_modele1_thin %>% filter(Parameter == "a")) +
  geom_rect(aes(xmin = 2.86, xmax = 3.13, ymin = 0, ymax = 6), fill =
"lightgrey", alpha = 0.2) +
  geom_density(aes(x = value, color = "Posterior"), fill = "red" , alpha =
```

```
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).

```r
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(&lambda;)") +
  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
```



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 a 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 $\tau = \frac{1}{\sigma^2} = \frac{1}{10000} = 0.0001$

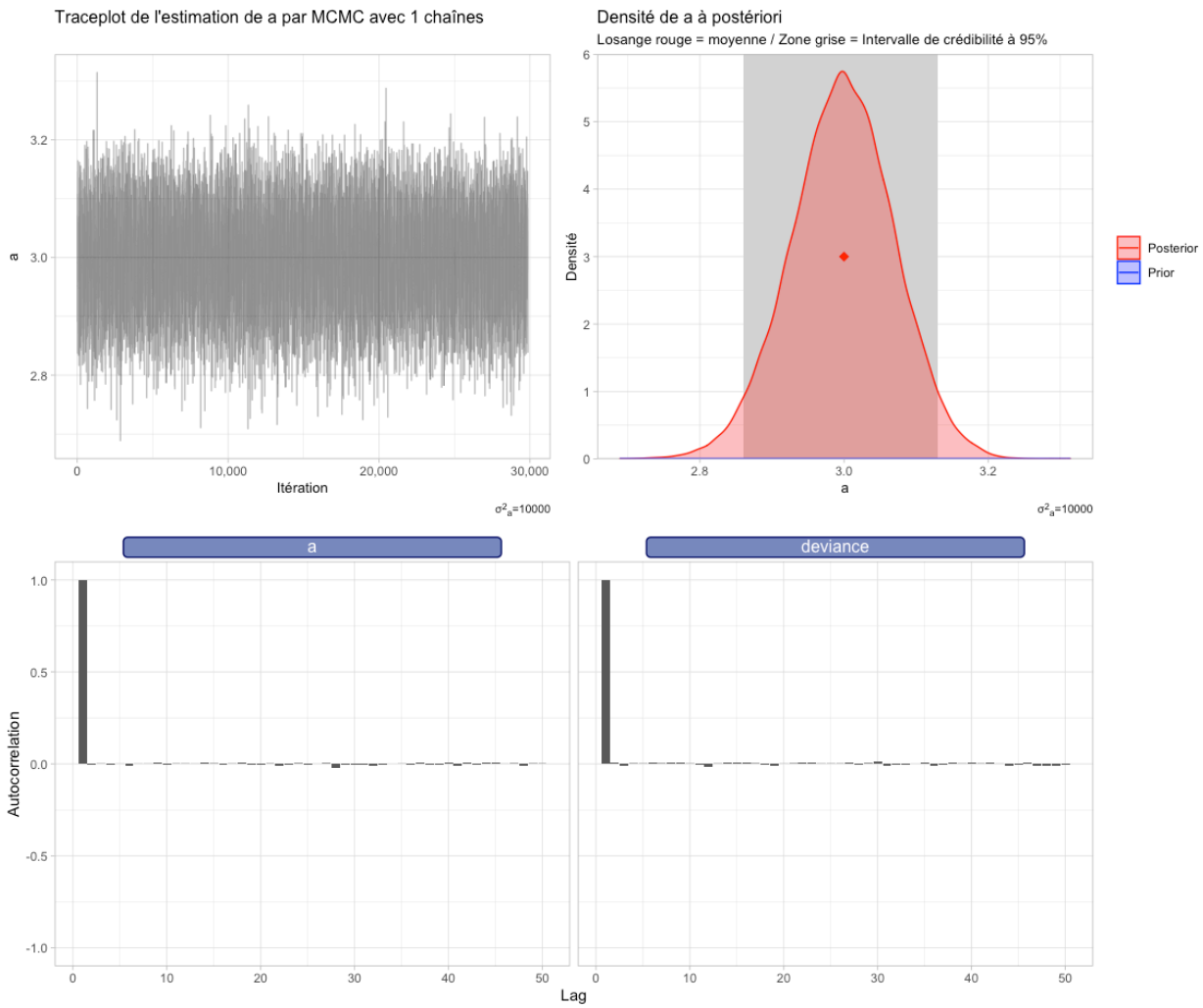
```
modele_1_sens <- function() {
  # Modèle pour yi
  for (i in 1:10) {
    nb_pannes[i] ~ dpois(exp(a))
  }
  # Loi a priori de a
  a ~ dnorm(0, 0.0001)
}
set.seed(1993)
modele1_fit_sens <- jags(data = donnees,
                         inits = inits_modele1,
                         parameters.to.save = 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 résultat du modèle apparaît très similaire, 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 précédant. 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 :

$$y_i \sim \mathcal{P}ois(\lambda_i)$$

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

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 $E(yi|a0, b0, xi)$ d'après ce modèle en fonction de a0, b0 et de xi ? Si b0=0, que cela signifie-t-il ? Même question si b0 est supérieur à 0 ou si b0 est inférieur à 0 ?*

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

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 a0 et b0 , 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 considèrera que c'est suffisant.

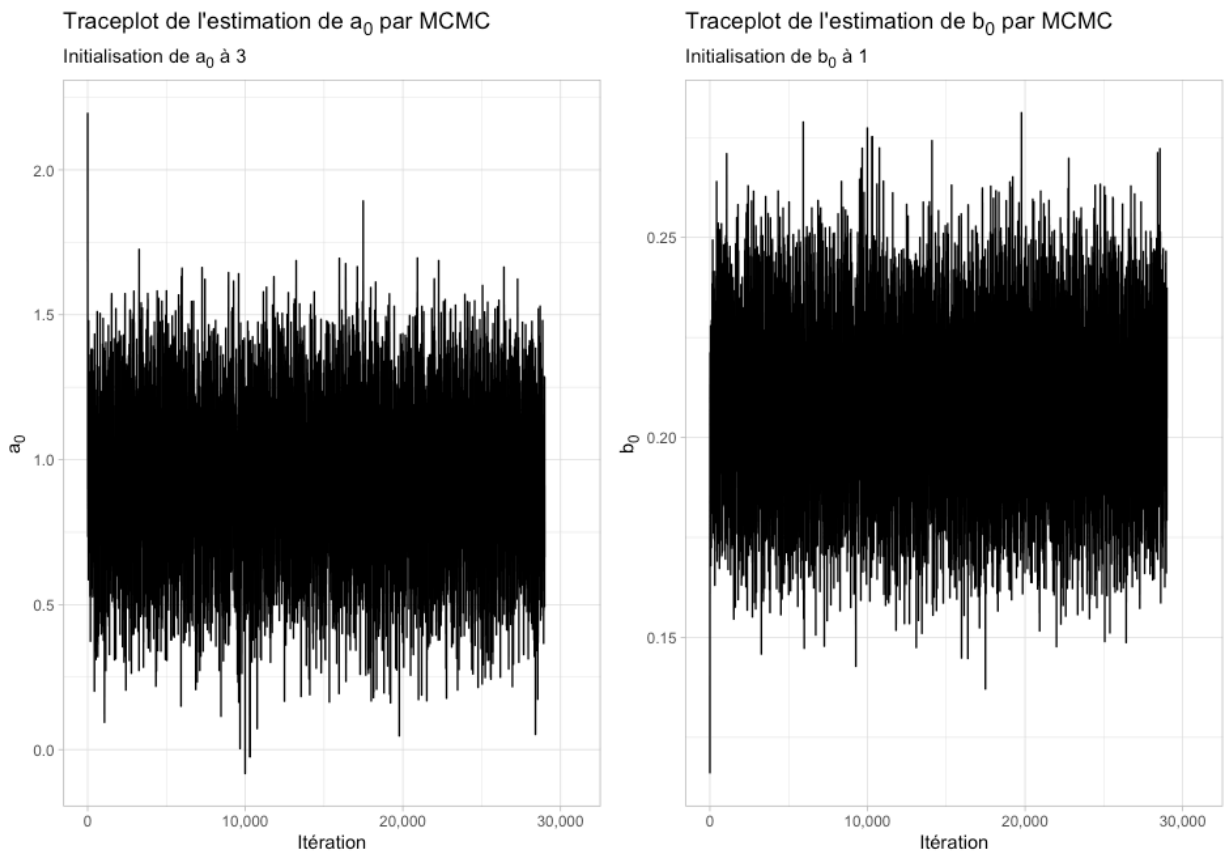
Réalisation du modèle avec JAGS en prenant 30000 itérations avec 1000 itérations de burn-in au début, en gardant toutes les itérations :

```
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)
```

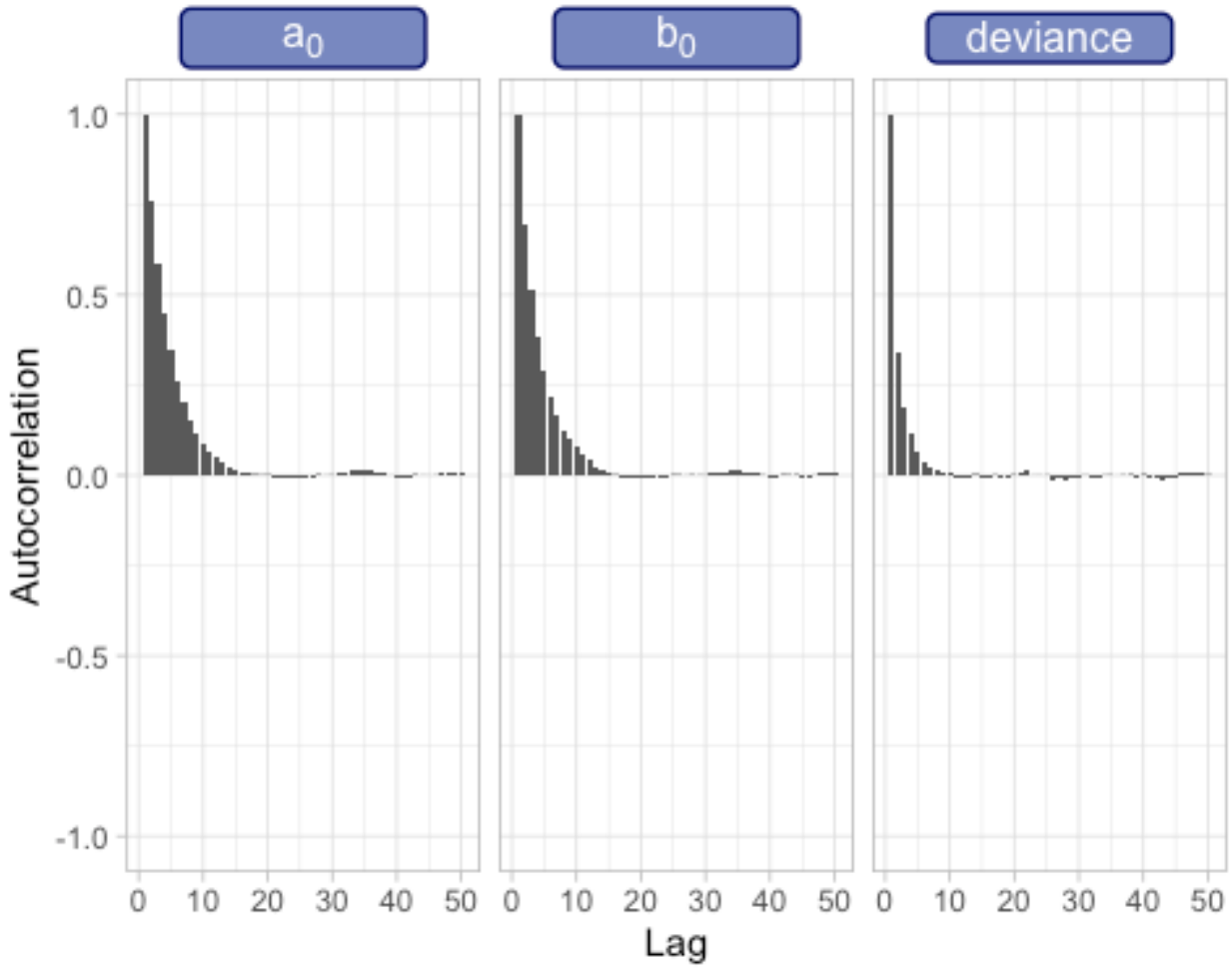
On regarde si les paramètres estimés ont bien convergé.

```
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
```



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'écarter 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>",
                                                   Parameter == "b0" ~
"b<sub>0</sub>",
                                                   TRUE ~
as.character(Parameter))))
```

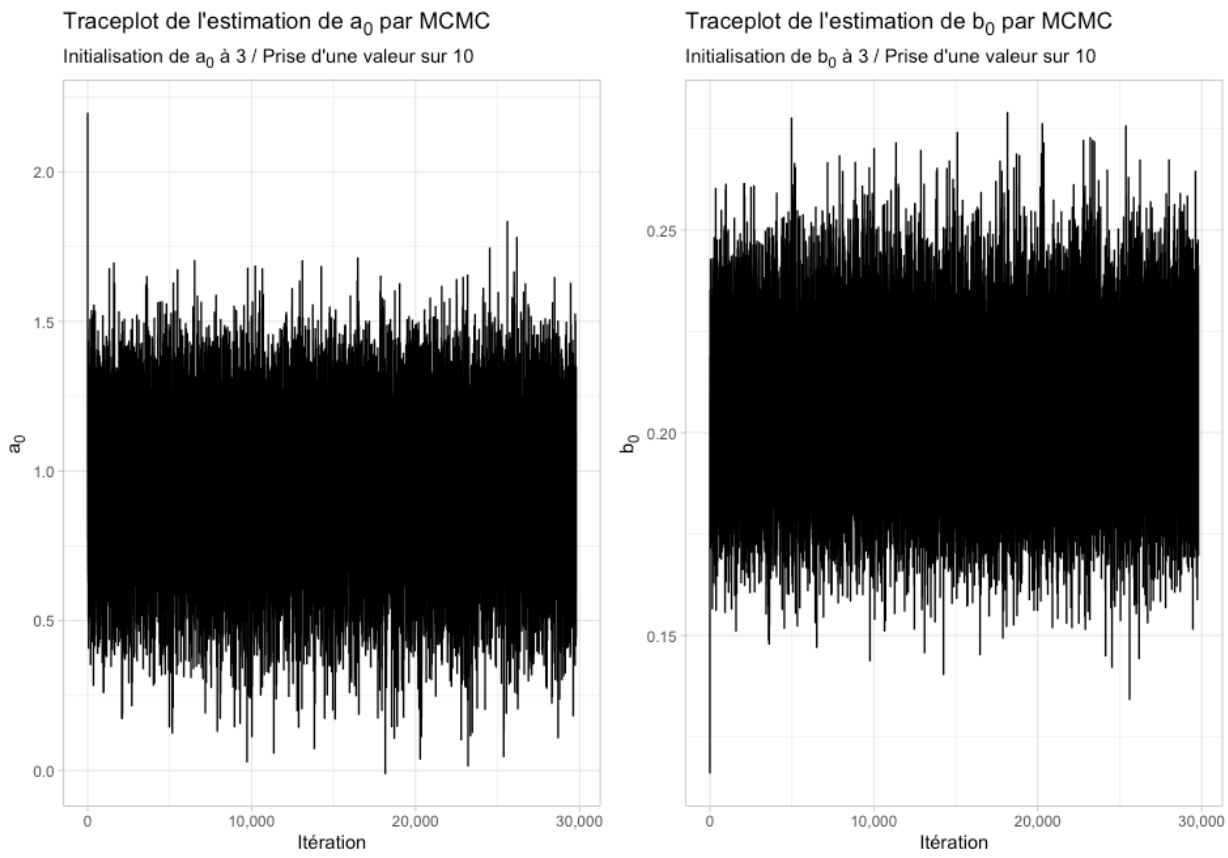


On voit que pour l'estimation de a_0 et b_0, il y a auto-corrélation jusqu'à la 15ème mesure. Pour la déviance, en revanche, cela va jusquà 10. De plus, le nombre d'itérations effectives est bien en-dessous du nombre d'itérations réalisées : nous avons fait 30000 itérations, et le nombre d'itérations 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.

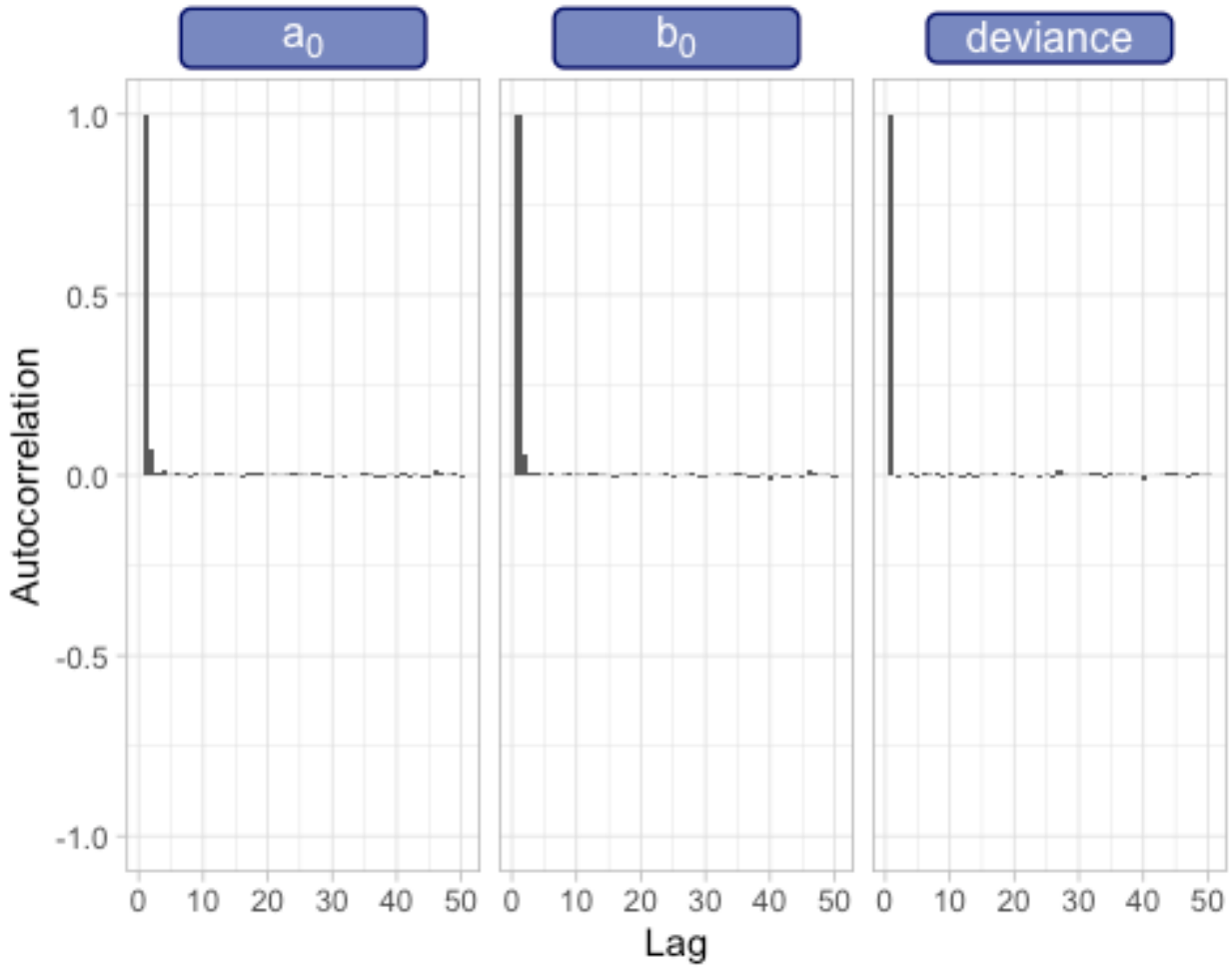
```
n_thin2.1 <-  10
n_burn2 <- 2000
set.seed(1993)
modele2_fit_thin <- jags(data = donnees,
                         inits = inits_modele2,
                         parameters.to.save = parametres_modele2,
                         n.chains = length(inits_modele2),
                         n.iter = n_iter2 * 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 3 / Prise d'une valeur
sur 10") +
  theme(plot.title = element_markdown())
plot_a0 + plot_b0
```



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" ~
"b<sub>0</sub>",
                                                   TRUE ~
as.character(Parameter))))
```



Sur le graphique d'auto-corrélations, 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 29000 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 :

```
tidy(modele2_fit) %>%
  mutate(across(estimate:std.error, ~ round(.x, 3))) %>%
  flextable() %>%
  set_header_labels(term = "Paramètre estimé",
                    estimate = "Estimation",
                    std.error = "Ecart-Type") %>%
  autofit()
```

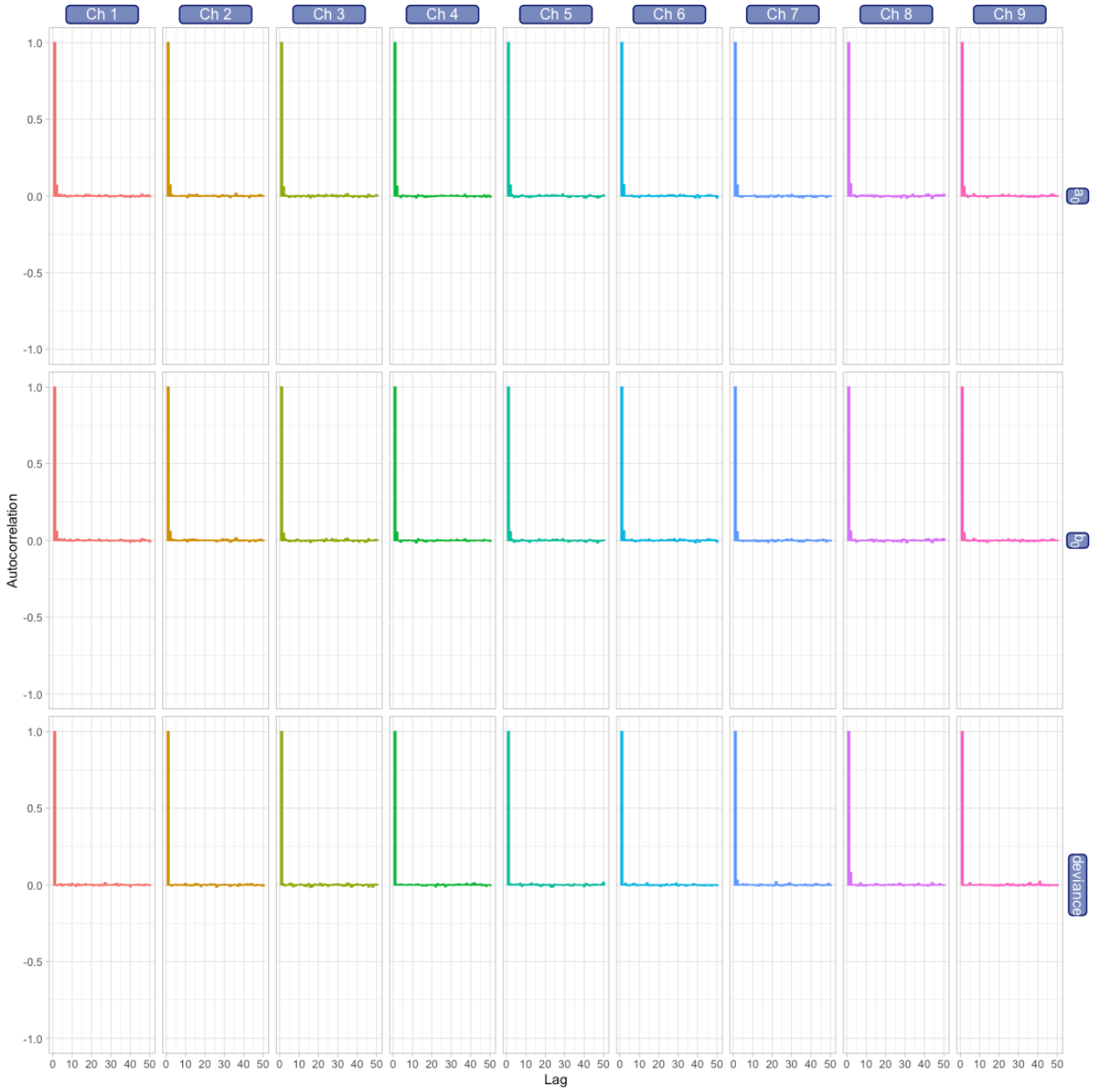
Paramètre estimé	Estimation	Ecart-Type

Paramètre estimé	Estimation	Ecart-Type
a0	0.944	0.230
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* à 5, -5 et 0, *b0* a 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")
```

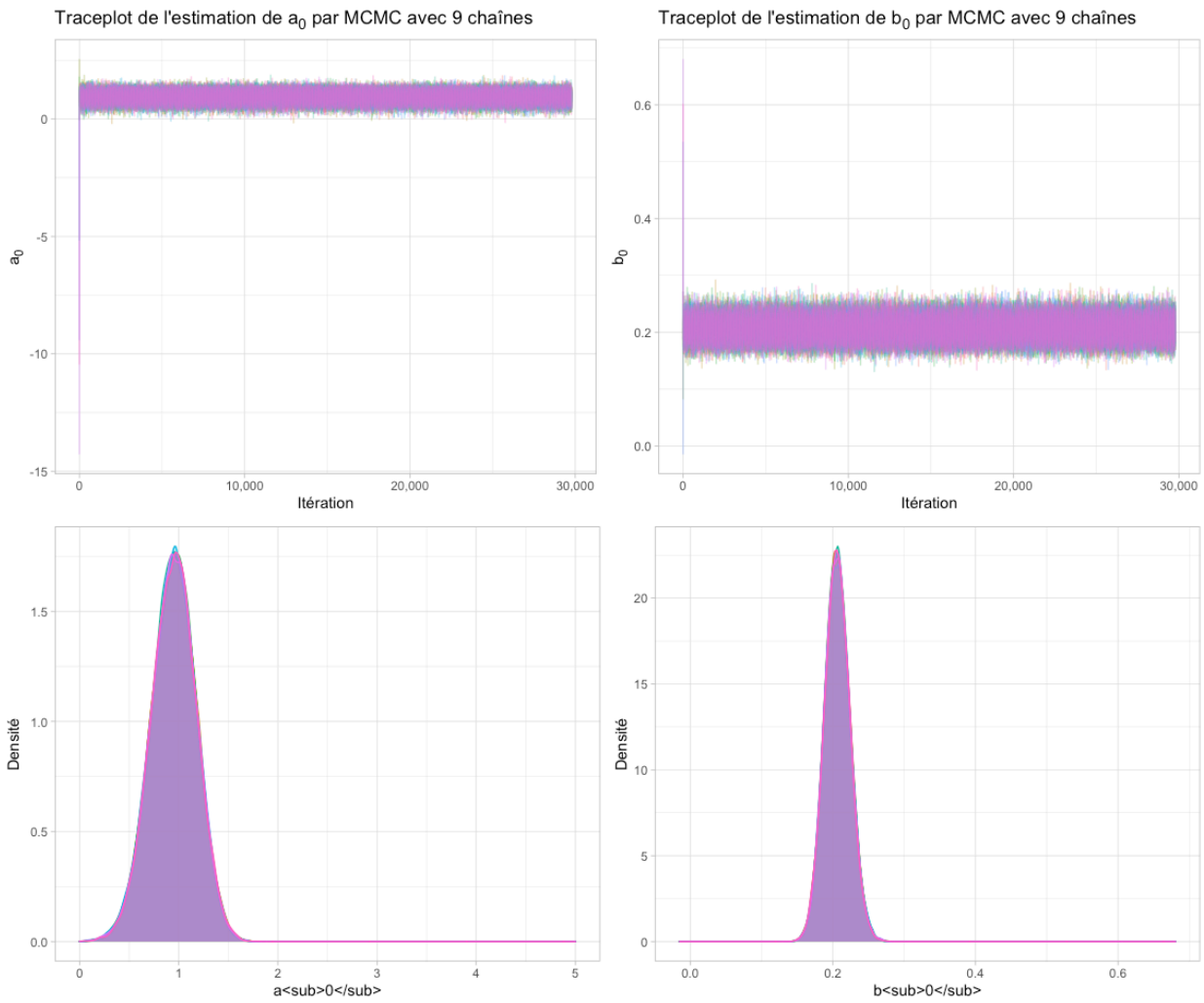


Au niveau des auto-corrélations, nous avons un résultat similaire à celui que nous avions eu avec une chaîne : cela semble satisfaisant avec peut-être un peu de corrélation pour un lag de 1. De plus, l'effective sample size reste proche du nombre d'itérations conservées : nous avons conservé 268200 itérations, 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
chaînes") +
  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
chaînes") +
  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)
```

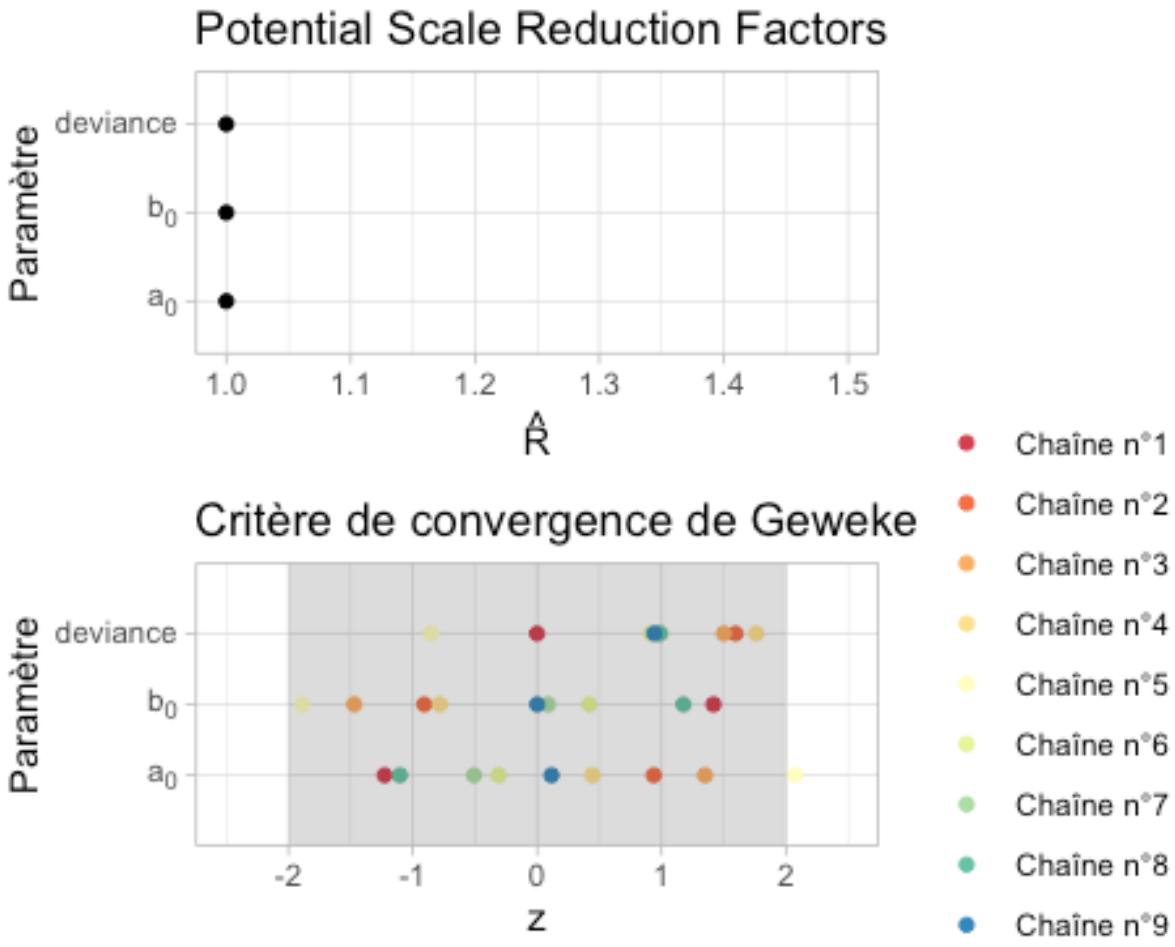
```
## Warning: Removed 21 rows containing non-finite values (stat_density).
```



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" ~
"b<sub>0</sub>",
                                                 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())
gelman / geweke
```



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

```
tidy(modele2_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
a0	0.950	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'itéartions 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.

```
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)

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 10
##    Unobserved stochastic nodes: 2
##    Total graph size: 45
```

```
##
## Initializing model

effectiveSize(modele2_fit_thinopti)

##      a0      b0  deviance
##  9630.188  9718.937 11800.000
```