Devoir Méthodes bayésiennes : session 1

Benoit Gachet, Guillaume Mulier

18/12/2020

Table of Contents

# 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)  
)

# Modèle 1

Le modèle est le suivant :

On a .

## Question 1

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

## Question 2

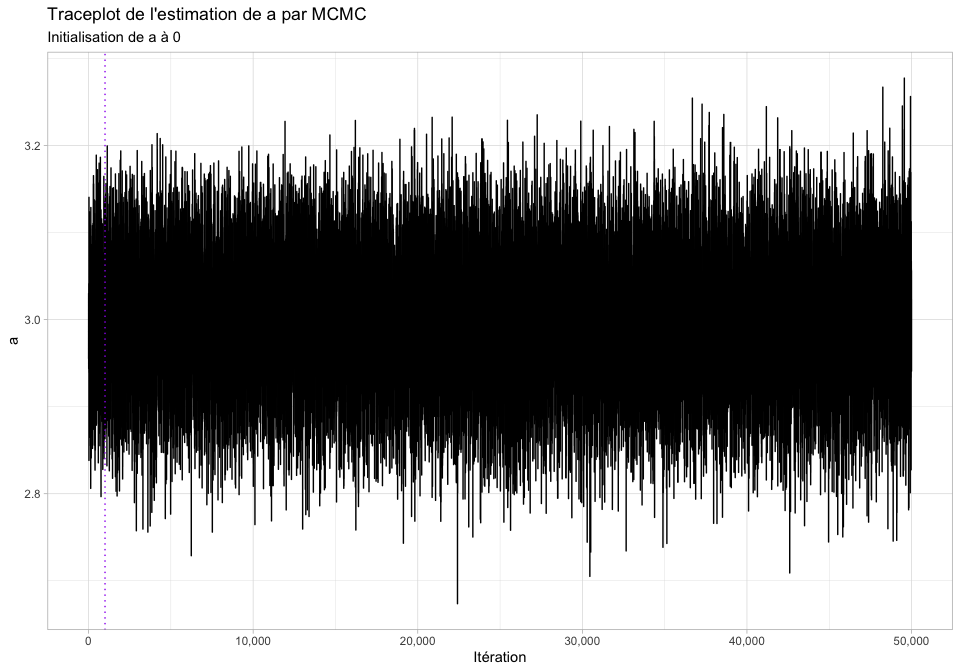
**Mettre en place ce modèle avec, comme loi a priori sur a, une loi normale d’espérance nulle et de variance 1000. Faire 30000 itérations et enlever 1000 itérations pour le temps de chauffe. D’après l’history et les autocorrélations, voyez-vous un problème de mélangeance de l’algorithme ? Si oui, résoudre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

Réalisation du modèle avec JAGS :

# 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 <- 50000 # Nombre d'iterations   
n\_burn <- 1000 # Burn in  
  
# Faire tourner le modèle avec la fonction jags  
# D'abord sans thin, ni burn in  
modele1\_fit <- jags(data = donnees,  
 inits = inits\_modele1,  
 parameters.to.save = parametres\_modele1,   
 n.chains = length(inits\_modele1),  
 n.iter = n\_iter,   
 n.burnin = 0,   
 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)

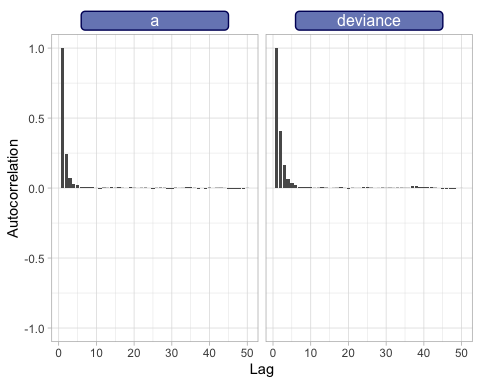
On regarde si le paramètre a estimé a bien convergé.

ggplot(gg\_modele1 %>% filter(Parameter == "a"), aes(x = Iteration, y = value)) +  
 geom\_line() +  
 geom\_vline(xintercept = 1000, color = "purple", linetype = "dotted") +  
 scale\_x\_continuous(labels = scales::comma\_format()) +  
 labs(x = "Itération",  
 y = "a",  
 title = "Traceplot de l'estimation de a par MCMC",  
 subtitle = "Initialisation de a à 0")



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. 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 diminué par rapport aux 50000 itérations faites : 28341.6451653 pour l’estimation de a et 20569.3247511 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 appliquerons un burn-in de 1000 observations pour éviter de prendre des itérations qui n’ont pas encore convergé.

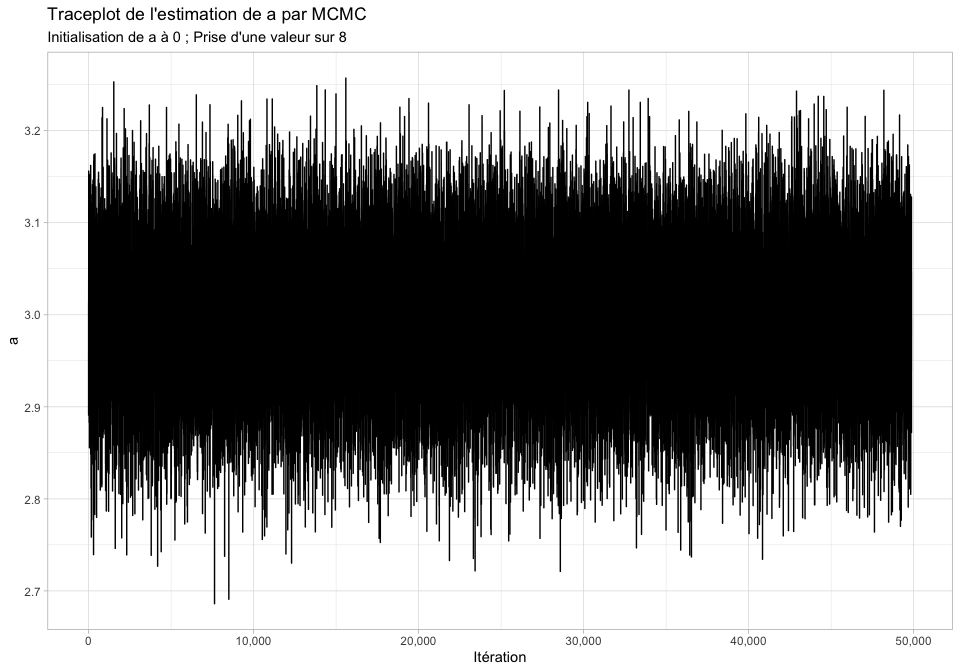
n\_thin <- 8  
modele1\_fit\_thin <- jags(data = donnees,  
 inits = inits\_modele1,  
 parameters.to.save = parametres\_modele1,   
 n.chains = length(inits\_modele1),  
 n.iter = n\_iter \* n\_thin,   
 n.burnin = n\_burn,   
 n.thin = n\_thin,  
 model.file = modele\_1)   
modele1\_fit\_thin\_mcmc <- 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()

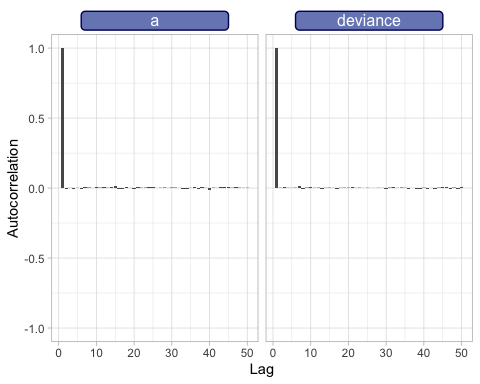
| Paramètre estimé | Estimation | Ecart-Type |
| --- | --- | --- |
| a | 2.999 | 0.07 |

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",  
 subtitle = "Initialisation de a à 0 ; Prise d'une valeur sur 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)

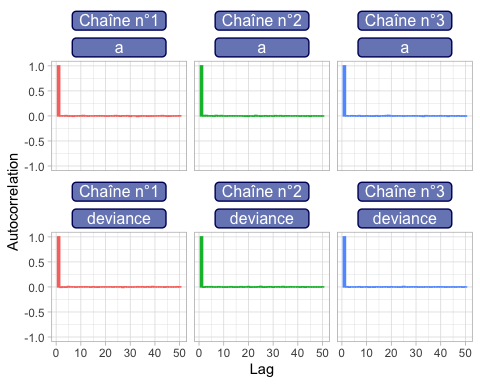


Sur le graphique d’auto-corrélations, on voit que le problème de mélangeance a été réglé et que maintenant il n’y a plus d’auto-corrélation pour le paramètre a estimé. De plus, le nombre d’itérations effectives est maintenant autour des 49875 itérations faites : 50547 pour l’estimation de a et 49875 pour l’estimation de la déviance du modèle (nous pensons que le chiffre dépassant 50000 pour a est soit lié au burn in de 1000, soit au fait que les observations soient très décorrélées).

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.

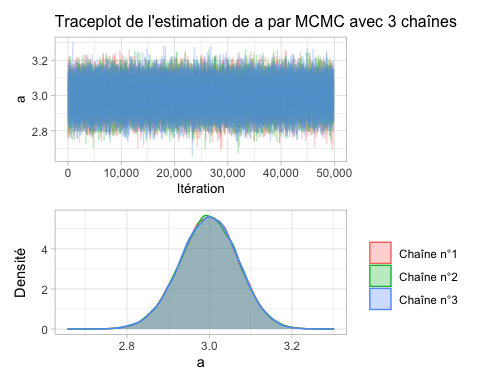
inits\_2 <- list("a" = 5)  
inits\_3 <- list("a" = -5)  
inits\_modele1\_mult <- list(inits\_1, inits\_2, inits\_3)  
modele1\_fit\_mult <- jags(data = donnees,  
 inits = inits\_modele1\_mult,  
 parameters.to.save = parametres\_modele1,   
 n.chains = length(inits\_modele1\_mult),  
 n.iter = n\_iter \* 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")



Pour les 3 chaînes, on ne voit pas d’auto-corrélation dans notre modèle. Aussi, le nombre d’itérations effectives est resté autour des 50000 itérations faites par chaîne, soit 149625 itérations en tout après retrait des 1000 itérations de burn-in : 145470 pour l’estimation de a et 149625 pour l’estimation de la déviance du modèle (ici, on ne doit plus avoir aucune corrélation vu que le nombre effectif est égal au nombre brut).

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\_text(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 initialisation 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.

Ainsi, voici notre estimation du paramètre a pour notre modèle avec 3 chaînes, 400000 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.

## Question 3

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

Pour le modèle final retenu avec 3 chaînes, il y avait 49875 itérations faites par chaîne, soit 149625 itérations au total. Le nombre d’itérations effectif était le même : 149625.

Pour le modèle avec 1 chaîne, 1000 itérations de burn-in et la conservation d’une itération sur 8, on avait 49875 itérations de faite, et le nombre effectif était plus élevé : 50547.

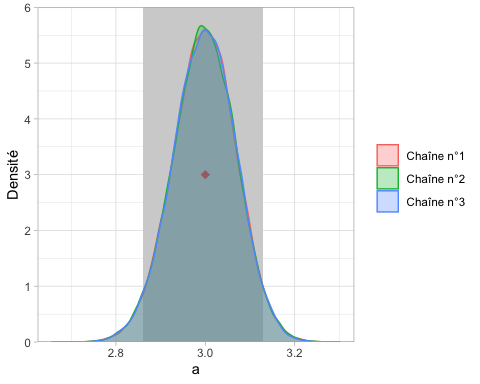
## Question 4

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

resm <- summary(modele1\_fit\_mult\_mcmc)[[1]] %>% as.data.frame()  
resq <- summary(modele1\_fit\_mult\_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].

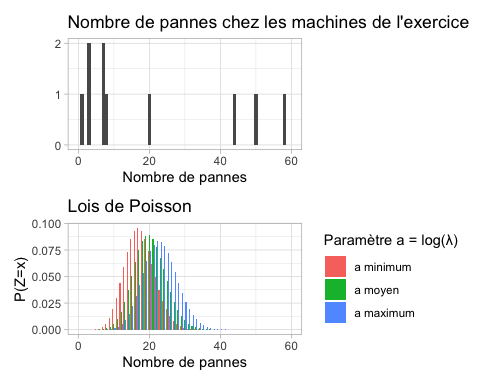
ggplot(gg\_modele1\_mult %>% filter(Parameter == "a") %>% mutate(Chain = paste0("Chaîne n°", Chain))) +  
 geom\_rect(aes(xmin = 2.86, xmax = 3.13, ymin = 0, ymax = 6), fill = "lightgrey", alpha = 0.2) +  
 annotate("point", x = 3, y = 3, color = "#FF0000", shape = 18, size = 3, alpha = 1) +  
 geom\_density(aes(x = value, color = as.factor(Chain), fill = as.factor(Chain)), alpha = 0.3) +  
 scale\_color\_discrete(name = NULL) +  
 scale\_fill\_discrete(name = NULL) +  
 scale\_y\_continuous(expand = expansion(mult = c(0, 0))) +  
 labs(x = "a",  
 y = "Densité")



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(&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

## Warning: Removed 4 rows containing missing values (geom\_col).



On peut voir que les loi 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 (20.09 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.

## Question 5

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

dic <- round(modele1\_fit\_mult$BUGSoutput$DIC, 4)  
complexite <- round(modele1\_fit\_mult$BUGSoutput$pD, 4)

Le DIC du modèle vaut 253.3654. Pour estimer la complexité du modèle, le pD est estimé et représente le nombre effectif de paramètres estimés. Pour notre modèle, il vaut 1.0027. Ce chiffre est voisin de 1, ce qui est en accord avec le modèle car nous n’estimons qu’un paramètre : qui vaut et est unique pour toutes les machines.

## 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

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)  
}  
modele1\_fit\_sens <- 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\_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)

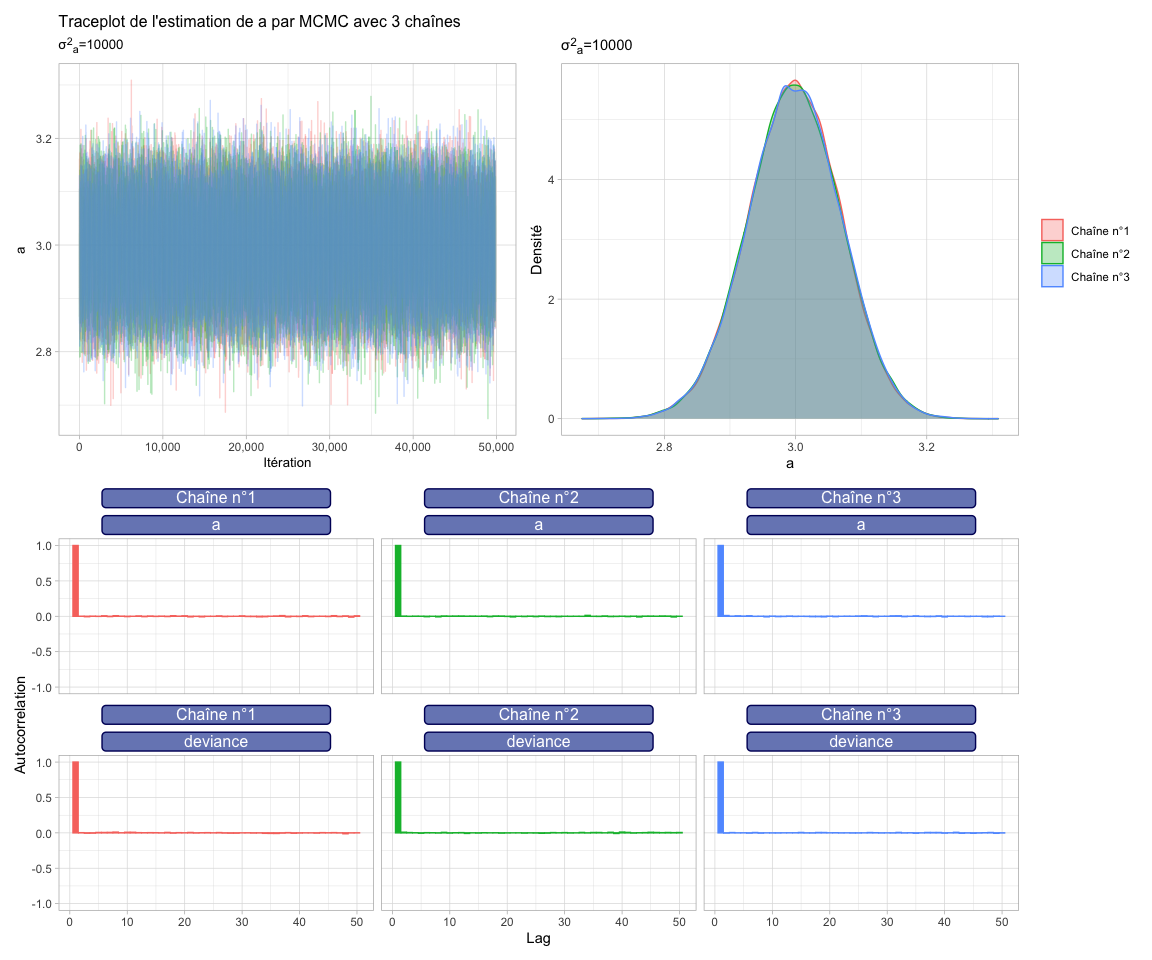
resm\_sens <- summary(modele1\_fit\_sens\_mcmc)[[1]] %>% as.data.frame()  
resq\_sens <- summary(modele1\_fit\_sens\_mcmc)[[2]] %>% as.data.frame()  
ic\_sens <- paste0(round(resm\_sens[["Mean"]][1], 2), "[", round(resq\_sens[["2.5%"]][1], 2), ";", round(resq\_sens[["97.5%"]][1], 2), "]")  
tidy(modele1\_fit\_sens) %>%   
 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 | 2.998 | 0.071 |

Les résultat du modèle apparaît très similaire, avec un a moyen et son intervalle de crédibilité à 95% de 3[2.86;3.13].

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(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",  
 subtitle = "&sigma;<sup>2</sup><sub>a</sub>=10000") +  
 theme(legend.position = "none",  
 title = element\_text(size = 10),  
 plot.subtitle = element\_markdown())  
plot2 <- ggplot(gg\_modele1\_sens %>% 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",  
 subtitle = "&sigma;<sup>2</sup><sub>a</sub>=10000")+  
 theme(plot.subtitle = element\_markdown())  
plot3 <- ggs\_autocorrelation(gg\_modele1\_sens %>% mutate(Chain = paste0("Chaîne n°", Chain))) +  
 facet\_wrap(~ Chain + Parameter, ncol = 3, dir = "v") +  
 theme(legend.position = "none")  
(plot1 + plot2) / plot3



En conclusion, en ayant pris une loi à priori sur le paramètre 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.