Inférence bayésienne et méthodes MCMC

Travaux dirigés

Pierre Gloaguen

```
library(tidyverse)
```

1 Inférence bayésienne pour le modèle linéaire

Soit Y un vecteur d'observations de $\mathbb{R}^n$, β un vecteur de paramètres inconnus $\mathbb{R}^{p+1}$ (tel que $n > p+1$) et X une matrice $n \times (p+1)$ telle que la matrice X^TX soit inversible. On considère le modèle linéaire Gaussien:

$$Y = X\beta + E$$

où E est un vecteur Gaussien de loi $\mathcal{N}(0, \sigma^2 I_n)$.

1.1 Cas où σ^2 est connu

1. Dans le cas où σ^2 est connu, écrire la vraisemblance associée au modèle précédent. Montrer que cette vraisemblance est proportionelle, en tant que densité de probabilité pour le vecteur β, à la densité d'un loi $\mathcal{N}((X^TX)^{-1}X^TY, \sigma^2(X^TX)^{-1})$. En déduire la densité a posteriori sur β pour une inférence bayésienne effectuée avec un prior impropre.

$$\begin{aligned}
L(Y|\beta, X) &= \frac{1}{\sqrt{2\pi\sigma^2}^n} \exp\left\{-\frac{1}{2\sigma^2}\left(Y - X\beta\right)^T\left(Y - X\beta\right)\right\} \times \overset{Prior\ impropre}{1} \\
&\propto \exp\left\{-\frac{1}{2\sigma^2}\left(\beta^T X^T X\beta - \beta^T X^T Y - Y^T X\beta\right)\right\} \\
&\propto \exp\left\{-\frac{1}{2\sigma^2}\left(\beta^T X^T X\beta - \beta^T (X^TX)(X^TX)^{-1}X^TY - Y^TX(X^TX)^{-1}(X^TX)\beta\right)\right\} \\
&\propto \exp\left\{-\frac{1}{2\sigma^2}\left((\beta - (X^TX)^{-1}X^TY)^T X^TX(\beta - (X^TX)^{-1}X^TY)\right)\right\}
\end{aligned}$$

1.2 Cas où σ^2 est inconnu

Dans ce cas, on pose comme loi a priori que le couple (β, σ^2) suit une loi normale inverse Gamma de paramètres $\mu \in \mathbb{R}^{p+1}$, V (une matrice de variance-covariance de taille $(p+1) \times (p+1)$, a et b (deux réels positifs).

Formellement:

$$\pi(\beta, \sigma^2|\mu, \mathbf{V}, a, b) \propto \left(\frac{1}{\sigma^2}\right)^{\frac{p+1}{2}} \left(\frac{1}{\sigma^2}\right)^{a+1} \exp\left(-\frac{b}{\sigma^2}\right) \exp\left(-\frac{(\beta-\mu)^T\mathbf{V}^{-1}(\beta-\mu)}{2\sigma^2}\right).$$

Remarque Cette modélisation est en fait assez naturelle, elle correspond au cas où σ^2 suit une loi inverse $\mathcal{G}amma(a, b)$) (usuelle pour les variances) et $\beta|\sigma^2 \sim \mathcal{N}(\mu, \sigma^2 V)$.

1. Montrer que la loi de $(\beta, \sigma^2)|Y, X$ suit également une loi Normale inverse Gamma dont vous préciserez les paramètres.

Il s'agit d'un exercice d'identification des paramètres.

$$\begin{aligned}\pi(\beta,\sigma^2|Y,X) &\propto L(Y|\beta,\sigma^2,X)\pi(\beta,\sigma^2)\\ &\propto \left(\frac{1}{\sigma^2}\right)^{\frac{p+1}{2}} \left(\frac{1}{\sigma^2}\right)^{a+n/2+1} \mathrm{e}^{-\frac{1}{2\sigma^2}(Y-X\beta)^T(Y-X\beta)} \exp\left(-\frac{-b}{\sigma^2}\right) \exp\left(-\frac{(\beta-\mu)^T\mathbf{V}^{-1}(\beta-\mu)}{2\sigma^2}\right).\end{aligned}$$

Après transformation, on trouve que

$$\pi(\beta,\sigma^2|Y,X) \sim \mathcal{N}ormInvGamma(\mu_{post}, V_{post}, a_{post}, b_{post})$$

où

$$\begin{aligned}V_{post} &= (X^TX + V^{-1})^{-1},\\ \mu_{post} &= V_{post}^{-1}(X^TY + V^{-1}\mu)\\ a_{post} &= a + \frac{n}{2}\\ b_{post} &= b + \frac{1}{2}(y^Ty + \mu^TV^{-1}\mu - \mu_{post}^TV_{post}^{-1}\mu_{post})\end{aligned}$$

2. Interprétez les paramètres en terme "d'apprentissage bayésien", c'est à dire en distinguant le poids du prior et des données.

2 Modèle probit avec covariables

On reprend l'exemple vu en cours et dans l'exercice 5 du TD3 sur l'estimation de covariables corrélées à la présence d'oiseaux.

2.1 Notations et modèle

On note $y_1, \ldots, y_n$ les observations de présence (1 si on observe un oiseau, 0 sinon) sur les sites 1 à n.

On note x_{ij} la valeur de la j-ème ($1 \leq j \leq 3$) covariable sur le i-ème site.

On suppose que les $y_1, \ldots, y_n$ sont les réalisations de variables aléatoires $Y_1, \ldots, Y_n$ telles que

$Y_i \sim \mathcal{B}ern(p_i)$ où

$$p_i = \phi(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3}) = \phi(\mathbf{x}_i^T\theta)$$

où $\theta = (\beta_0, \ldots, \beta_3)^T$ et ϕ la fonction de répartition d'une $\mathcal{N}(0,1)$. L'objectif est d'estimer le vecteur θ dans un cadre bayésien.

2.1.1 Vraisemblance et posterior

1. Rappelez l'expression de la vraisemblance d'un paramètre θ pour vecteur d'observations $\mathbf{y}$ ainsi que l'expression du posterior associée à un prior $\mathcal{N}(0, 4I_4)$.

Le posterior est donc donné par (voir cours et poly):

$$\pi(\theta|y_{1:n}) \propto \pi(\theta)L(y_{1:n}|\theta) \propto \mathrm{e}^{-\frac{1}{8}\theta^T\theta} \prod_{k=1}^{n} \phi(\mathbf{x}_k^T\theta)^{y_k}(1-\phi(\mathbf{x}_k^T\theta))^{1-y_k}$$

2.2 Algorithme de Metropolis Hastings

2. On se propose d'approcher la loi *a posteriori* en utilisant un algorithme MCMC. Plus précisemment, on se propose de générer une chaîne de Markov $(\theta_n)_{n\geq 0}$ dont l'unique loi stationnaire est le posterior défini plus haut. Pour cela, on utilisera un algorithme de Metropolis Hastings dont le noyau de transition est une marche aléatoire de loi normale $\mathcal{N}(0, \sigma^2 I_4)$ où I_4 est la matrice identité 4×4. Définir l'algorithme de Metropolis Hastings pour un jeu de données $\mathbf{y}$.

On note $q(x, y)$ la densité d'une loi normale $\mathcal{N}(x, \sigma^2 I_4)$.

On construit la chaîne de Markov $\{X_t\}_{t\in\mathbb{N}}$ de la manière suivante.

1. Choisir X_0
2. Pour $t \geq 0$
   (a) Tirer $Z \sim \mathcal{N}(X_t, \sigma^2 I_4)$
   (b) Tirer (indépendemment) $U \sim \mathcal{U}[0, 1]$
   (c) Calculer
   $$\alpha(X_t, Z) = \frac{\pi(Z|y_{1:n})q(Z, X_t)}{\pi(X_t|y_{1:n})q(X_t, Z)}$$
   (d) Si $U \leq \alpha(X_t, Z)$, on pose $X_{t+1} = Z$, sinon, on pose $X_{t+1} = X_t$

Ici, comme la marche aléatoire indexée par q permet de visiter tout $\mathbb{R}^4$, et qu'elle n'est pas périodique, la chaîne de Markov a pour loi invariante la loi a posteriori que l'on cible.

Remarques

1. Ici, on ne connaît pas $\pi(\theta|y_{1:n})$, mais seulement $\tilde{\pi}(\theta|y_{1:n})$. Ce n'est pas grave car:
   $$\frac{\pi(Z|y_{1:n})}{\pi(X_t|y_{1:n})} = \frac{\tilde{\pi}(Z|y_{1:n})}{\tilde{\pi}(X_t|y_{1:n})}$$
2. On remarque de plus que dans notre cas où le noyau de Markov est symmétrique, $q(x, y) = q(y, x)$, donc finalement
   $$\alpha(X_t, Z) = \frac{\tilde{\pi}(Z|y_{1:n})}{\tilde{\pi}(X_t|y_{1:n})}$$

3. Le fichier `donnees_presence_complet.txt` contient les observations de 300 sites sur lesquels la présence d'oiseaux a été constatée, ainsi que différentes variables environnementales mesurées. Ecrire un programme `R` codant l'algorithme de Metropolis Hastings précédent pour ce jeu de données. Vous testerez plusieurs valeurs de σ^2 pour la variance de la marche aléatoire, et choisirez celle qui vous semble la meilleure.

On créée le vecteur $\mathbf{y}$ et la matrice de design $\mathbf{X}$ à partir des données.

```
design_matrix <- donnees_presence_complet %>%
  select(-presence) %>%
  as.matrix() %>%
```

```r
  cbind(intercept = rep(1, nrow(.)), .)
y_vector <- donnees_presence_complet %>%
  pull(presence) %>%
  as.character() %>%
  as.numeric()
```

Ensuite, on crée les fonction importantes (notamment l'evaluation du posterior)

```r
get_likelihood <- function(beta_vec, X, y, log = FALSE){
  phis <- pnorm(as.numeric(X %*% beta_vec), log.p = T)
  log_likelihood <- rep(NA, nrow(X))
  log_likelihood[y == 1] <- phis[y == 1]
  log_likelihood[y == 0] <- log(1 - exp(phis[y == 0]))
  if(log){
    return(sum(log_likelihood))
  }
  else
    return(exp(sum(log_likelihood)))
}
get_prior <- function(beta_vec, log = FALSE){
  log_prior <- sum(dnorm(length(beta_vec), 0, 4, log = TRUE))
  if(log){
    return(log_prior)
  }
  else{
    return(exp(log_prior))
  }
}
```

```r
get_posterior <- function(beta_vec, X, y, log = FALSE){
  log_posterior <- get_prior(beta_vec, log = TRUE)  +
    get_likelihood(beta_vec, X, y, log = TRUE)
  if(log){
    return(log_posterior)
  }
  else{
    return(exp(log_posterior))
  }
}
```

On a désormais tous les ingrédients pour coder notre fonction de Metropolis Hastings

```r
get_metropolis_sampling <- function(beta_init, # Première valeur de beta
                                    n_step, # Nombre d'iterations
                                    sigma2, # parametre de la marche aleatoire
                                    X, y # Parametre supplémentaires (données)
                                    ){
  beta_dim <- length(beta_init)
  # On initialise notre sortie, qui sera une matrice
  out <- matrix(ncol = beta_dim, nrow = n_step + 1,
                dimnames = list(NULL, paste0("beta_", 0:(beta_dim - 1))))
  # On nomme chaque colonne de la matrice beta_0, beta_1, ....
  out[1, ] <- beta_init # Valeur initiale de la chaîne
  my_sigma <- sqrt(sigma2) # Passage a l'ecart type (pour rnorm)
```

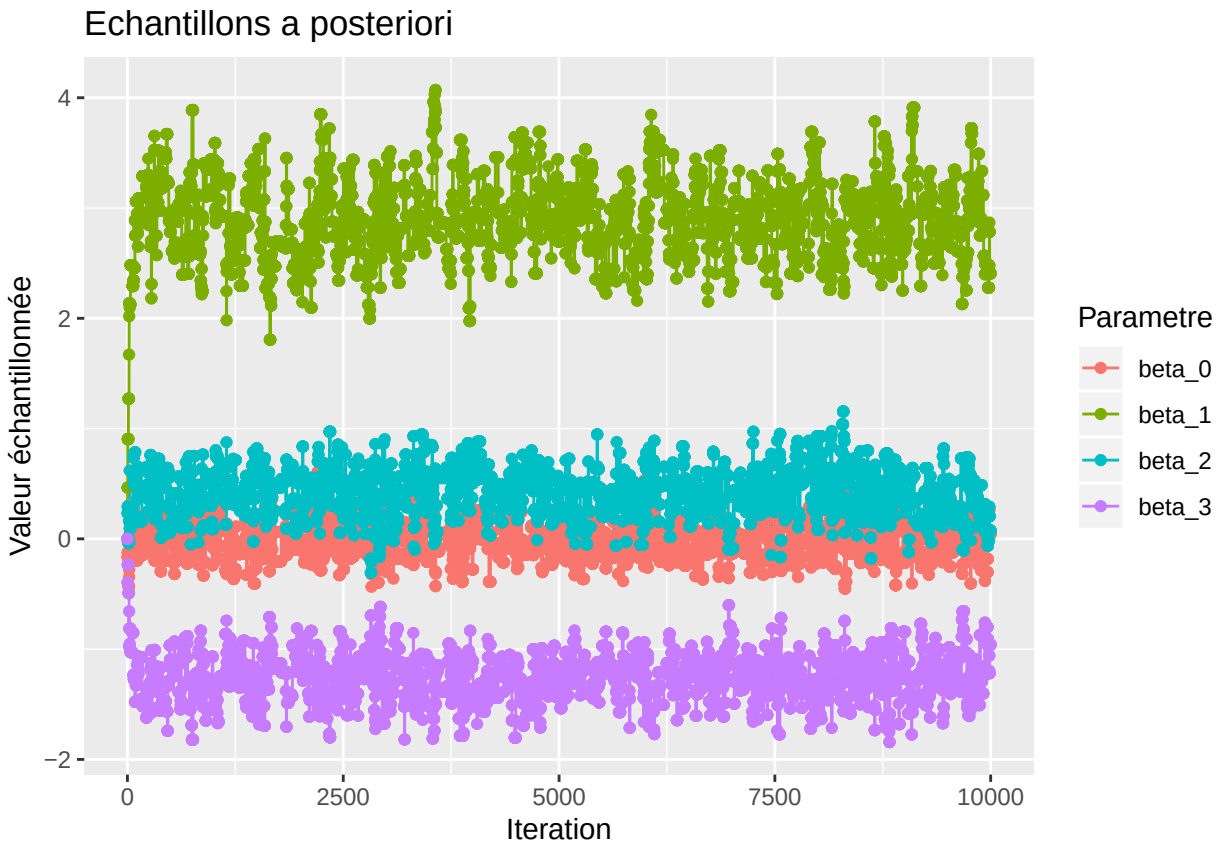
```
  accepted <- rep(NA, n_step + 1) # On va garder ça en mémoire
  log_posterior <- rep(NA, n_step + 1) # On va garder ça en mémoire
  # Il est souvent conseillé de travailler en logarithme
  log_posterior[1] <- get_posterior(beta_init, X, y, log = TRUE)
  if(is.infinite(log_posterior[1])){
    # Si mon point de départ initial est numériquement trop loin
    # pour éviter les problèmes
    stop("First log posterior value is infinite, change beta_init")
  }
  for(i in 1:n_step){
    candidate <- rnorm(beta_dim, out[i, ], sd = my_sigma) # On tire Z
    # Calcul de pi(Z | X, y)
    candidate_log_posterior <- get_posterior(candidate, X, y, log = TRUE)
    log_u <- log(runif(1)) # En log aussi!
    accepted[i + 1] <- log_u < (candidate_log_posterior - log_posterior[i])
    if (accepted[i + 1]) {
      # Si on accepte
      out[i + 1, ] <- candidate
      log_posterior[i + 1] <- candidate_log_posterior
    }
    else {
      # Si on refuse
      out[i + 1, ] <- out[i, ] # La chaine est encore actualisée!
      log_posterior[i + 1] <- log_posterior[i]
    }
  }
  # On sort sous forme de tableau
  tibble(iteration = 0:n_step) %>%
    bind_cols(as_tibble(out)) %>%
    mutate(log_posterior = log_posterior,
           accepted = accepted,
           sigma2 = sigma2) %>%
    return()
}
```

On peut ainsi regarder un premier résultat.

```
premier_mcmc <- get_metropolis_sampling(beta_init = rep(0, 4),
                                        X = design_matrix, y = y_vector,
                                        n_step = 1e4, sigma2 = 0.1)
mean(premier_mcmc$accepted, na.rm = T)
```

```
[1] 0.125
```

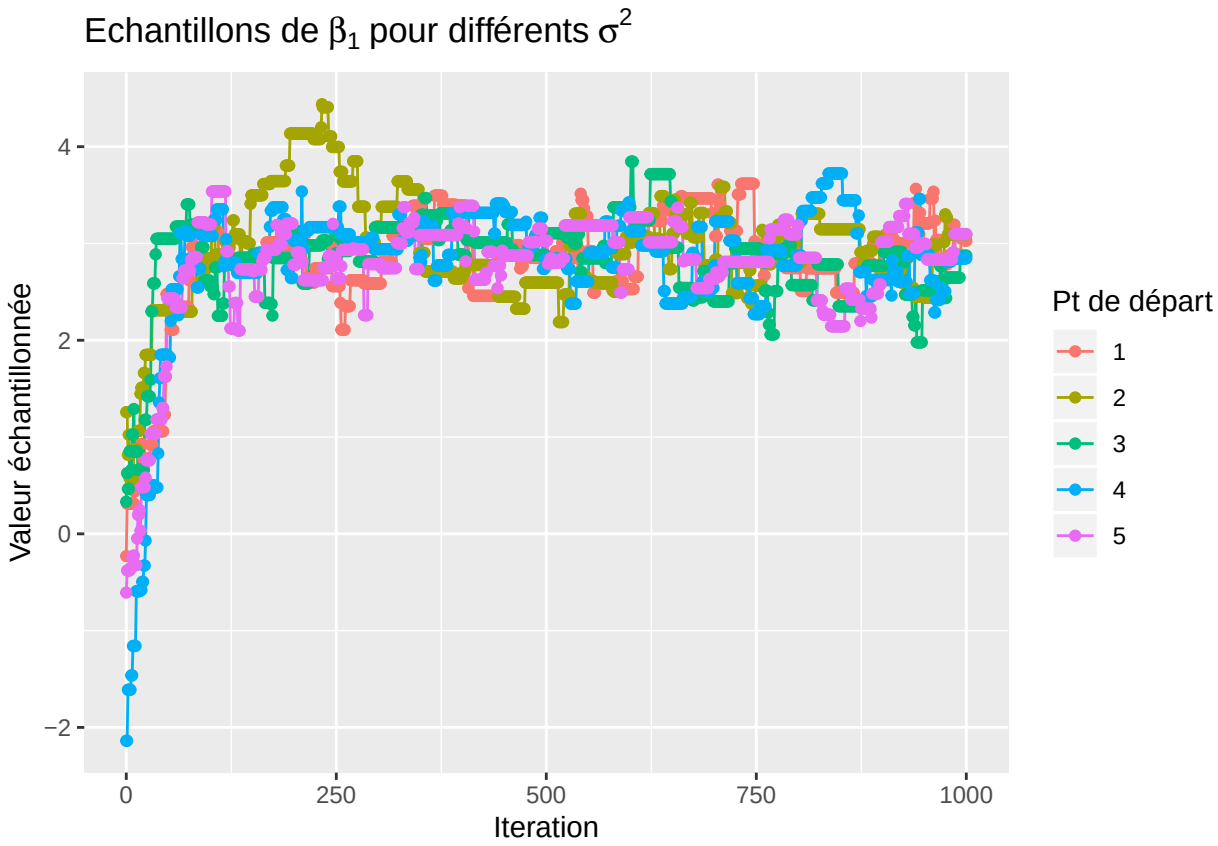
```
premier_mcmc %>%
  select(-log_posterior, -accepted, -sigma2) %>% # On vire des colonnes
  gather(-iteration, key = "Parametre",
         value = "Sample", factor_key = TRUE) %>%
  ggplot(aes(x = iteration, y = Sample, colour = Parametre)) +
  geom_line() +
  geom_point() +
  labs(y = "Valeur échantillonnée", x = "Iteration",
       title = "Echantillons a posteriori")
```



Il est **extrêmement important** de vérifier que le point de départ n'influe pas sur les valeurs échantillonnées (au moins au bout d'un certain temps).

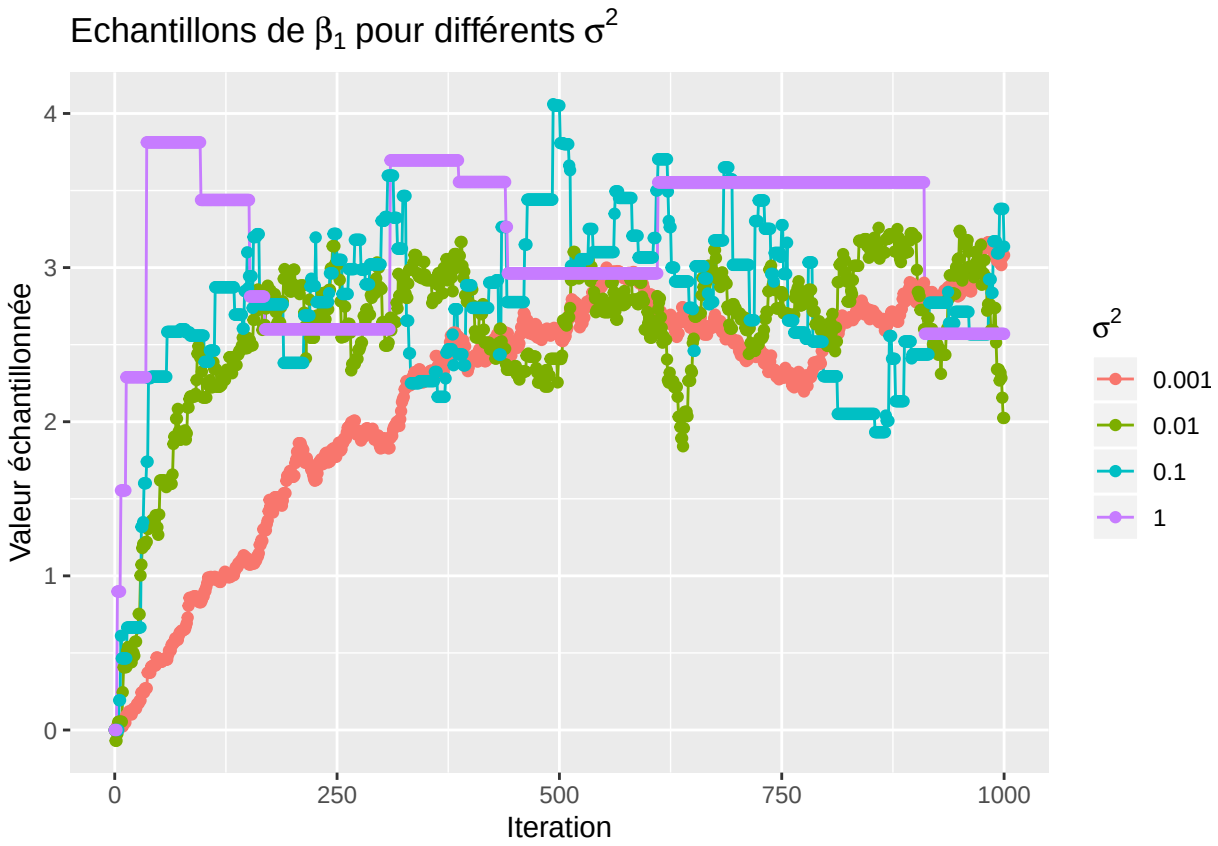
Cela revient à s'assurer que, quelque soit le point de départ, on a bien atteint la loi stationnaire. Ici, on vérifie que pour la valeur de β_1, il n'y a pas d'influence du point de départ.

```
set.seed(123)
mcmc_multiple_start <- rerun(5,
                             get_metropolis_sampling(rnorm(4),
                                                     X = design_matrix,
                                                     y = y_vector,
                                                     n_step = 1e3, sigma2 = 0.1)) %>%
  bind_rows(.id = "Replicate")
ggplot(mcmc_multiple_start) +
  aes(x = iteration, y = beta_1, colour = Replicate) +
  geom_line() +
  geom_point() +
  labs(y = "Valeur échantillonnée", x = "Iteration",
       title = expression("Echantillons de"~beta[1]~"pour différents"~sigma^2),
       color = "Pt de départ")
```



Choix de σ^2 On peut regarder l'évolution de la chaîne selon la valeur de σ^2 choisie.

```
set.seed(123)
mcmc_multiple_sigma <- map_dfr(c(1e-3, 1e-2, 1e-1, 1),
                               function(my_sigma)
                                 get_metropolis_sampling(rep(0, 4),
                                                         X = design_matrix,
                                                         y = y_vector,
                                                         n_step = 1e3,
                                                         sigma2 = my_sigma))
ggplot(mcmc_multiple_sigma) +
  aes(x = iteration, y = beta_1, colour = factor(sigma2)) +
  geom_line() +
  geom_point() +
  labs(y = "Valeur échantillonnée", x = "Iteration",
       title = expression("Echantillons de"~beta[1]~"pour différents"~sigma^2),
       color = expression(sigma^2))
```



Pour les deux cas extrêmes, on voit deux comportements problématiques.

Pour $\sigma^2 = 1$, on voit que la chaîne reste bloquée sur les mêmes valeurs la plupart du temps. ce qui implique en pratique que tout estimateur d'espérance selon la loi de $\sigma^2|\mathbf{y}, \mathbf{X}$ aura une très grande variance.

Pour $\sigma^2 = 0.001$, on voit que la chaîne progresse lentement, cette chaîne présente donc une forte autocorrelation. Ceci implique également en pratique que tout estimateur d'espérance selon la loi de $\sigma^2|\mathbf{y}, \mathbf{X}$ aura une très grande variance. Ici, $\sigma^2 = 0.01$ semble un bon compromis.

4. Pour le σ^2 choisi quelle est la probabilité d'acceptation empirique?

On obtient les taux d'acceptations suivants

```
mcmc_multiple_sigma %>%
  group_by(sigma2) %>%
  summarise(taux_acceptation = mean(accepted, na.rm = TRUE)) %>%
  knitr::kable(col.names = c("$\\sigma^2$", "Taux d'acceptation"))
```

σ^2	Taux d'acceptation
0.001	0.781
0.010	0.515
0.100	0.147
1.000	0.013

On peut regarder également l'autocorrélation de la chaîne (pour β_1, ici)

```
mcmc_multiple_sigma %>%
  group_by(sigma2) %>%
  summarise(autocorrel = cor(beta_1[-1], beta_1[-n()])) %>%
  knitr::kable(col.names = c("$\\sigma^2$", "Autocorrelation"))
```

σ^2	Autocorrelation
0.001	0.9993751
0.010	0.9917581
0.100	0.9835038
1.000	0.9864433

5. Quelle est la valeur réalisée de l'estimateur Bayésien $\mathbb{E}[\theta|\mathbf{Y}]$?

On a désormais accès aux tirages de la chaîne de Markov qui a pour loi stationnaire notre loi cible. En pratique, pour estimer des espérances selon cette loi, on utilisera des méthodes de Monte Carlo standard, i.e., supposant que chaque tirage est i.i.d. de la loi voulue.

Or cette hypothèse est violée ici pour deux raisons:

1. Les échantillons ne sont pas indépendants car ils sont issus d'une chaîne de Markov. Cependant, pour deux valeurs distantes de la chaîne, on peut penser que cette hypothèse d'indépendance sera valable.
2. Les échantillons ne sont pas tirés selon la loi cible. Asymptotiquement, on pourra le supposer, mais le point de départ n'est pas dans cette loi cible. On peut supposer cependant qu'au bout d'un certain temps, la chaîne a atteint la loi stationnaire.

Afin de palier ces deux problèmes, on va sous échantillonné la chaîne de Markov selon deux principes, le *thinning* et le *burn-in*.

1. On choisit un écart temporel (le *thin*) au delà duquel on suppose que toutes les valeurs de la chaîne sont indépendantes. En pratique, en gardera un point sur 10, un point sur 100, un point sur 1000,... selon l'autocorrelation initiale de la chaîne.
2. On choisit un pas de temps (le *burn*) en deça duquel on jettera tous les échantillons, car on suppose que la chaîne n'a pas encore atteint sa loi stationnaire. De même, le choix de ce pas dépendra des données, et devra être justifié au moins graphiquement.

Ici, on garde $\sigma^2 = 0.01$, on voit sur les graphes précédents qu'un *burn-in* de 200 semble raisonnable (il faudrait vérifier sur les autres paramètres!). On gardera un point sur 100. Afin d'avoir un nombre surffisant de points pour l'approximation de l'espérance par méthode de Monte Carlo, on augmentera le nombre d'itérations.

On pourra vérifier ici que l'autocorrelation diminue en deça de 0.1 avec ces valeurs.

```
set.seed(123)
my_mcmc_sample <- get_metropolis_sampling(rep(0, 4), # On fait un metropolis
                                          X = design_matrix,
                                          y = y_vector,
                                          n_step = 5e4, sigma2 = 0.01) %>%
  # Puis on ne garde qu'un sous ensemble des points
  dplyr::filter(iteration >= 200, # Burn-in
                (iteration %% 100) == 0) # Thinning
```

On pourra supposer que cet échantillon est i.i.d. de notre loi cible. On peut alors estimer l'espérance a posteriori de manière classique, en utilisant la moyenne empirique de cet échantillon.

Paramètre	Espérance a posteriori
β_0	-0.023
β_1	2.906
β_2	0.412
β_3	-1.265

6. Donner un intervalle de crédibilité à 95% pour chacun des paramètres.

De la même manière, un intervalle de crédiilité sera donné par les quantiles de niveaux correspondants.

```
my_mcmc_sample %>%
  select(beta_0, beta_1, beta_2, beta_3) %>%
  gather(key = "Parametre", value = "Echantillon", factor_key = TRUE) %>%
  group_by(Parametre) %>%
  summarise(IC_inf = paste0("[",
                            paste(quantile(Echantillon, probs = c(0.025,0.975)) %>%
                                    round(3),
                                  collapse = ", "),
                            "]")) %>%
  mutate(Parametre = factor(Parametre,
                            labels = paste0("$\\beta_", 0:3,"$"))) %>%
  knitr::kable(col.names = c("Paramètre", "Intervalle de crédibilité à 95 \\%"))
```

Paramètre	Intervalle de crédibilité à 95 %
β_0	[-0.317, 0.24]
β_1	[2.324, 3.63]
β_2	[-0.013, 0.836]
β_3	[-1.709, -0.901]