Computational Statistics

Homework 4

Romane PERSCH

December 9, 2016

7.1 Accept-Reject and Metropolis-Hastings comparison to generate Gamma variables

The goal of the Problem is to approximate the mean of a Gamma(4.3, 6.2)

(a) Accept-Reject with a Gamma(4,7) candidate

Gamma distribution There are two different ways to specify the parameters for Gamma distributions. In this Problem, we will necessarily consider that a Gamma(α, β) corresponds to the probability density function (see Appendix for an explanation of this specification):

$$\forall x > 0, \quad f_{\alpha,\beta}(x) = \frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} e^{-\frac{x}{\beta}}$$

How to simulate from the proposal, a Gamma(4,7) ? As $4 \in \mathbb{N}$, a Gamma(4,7) can be represented as the sum of 4 independent exponential random variables $\epsilon_i \sim \mathcal{E}(\frac{1}{\beta})$, which are easy to simulate since $\epsilon_i = -\log(U_i)\beta$ (using Inverse Transform method).

Accept-Reject upper bound computation Let Gamma(α, β) be the target distribution (i. e : $\alpha = 4.3$ and $\beta = 6.2$) and Gamma(a, b) be the proposal distribution (i.e : $a = 4$ and $b = 7$). In order to use Accept-Reject, we need to find M such that $\forall x > 0, \quad f_{\alpha,\beta}(x) \leq M f_{a,b}(x)$. Therefore, we need to find an upper bound to the ratio :

$$w(x) = \frac{f_{\alpha,\beta}(x)}{f_{a,b}(x)} = \frac{\Gamma(a)b^a}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-a} e^{-(\frac{x}{\beta}-\frac{x}{b})}$$

$$w(x) = \frac{f_{\alpha,\beta}(x)}{f_{a,b}(x)} = \frac{\Gamma(a)b^a}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-a} e^{\frac{\beta-b}{b\beta}x}$$

Note that is our case $\alpha - a > 0$ and $\frac{\beta-b}{b\beta} < 0$. Therefore, $\lim_{x\to+\infty} w(x) = 0$ and $\lim_{x\to 0^+} w(x) = 0$.

w is differentiable on $\mathbb{R}_+^*$ and :

$$w'(x) = \frac{\Gamma(a)b^a}{\Gamma(\alpha)\beta^\alpha} \left[(\alpha - a)x^{\alpha-a-1} e^{\frac{\beta-b}{b\beta}x} + x^{\alpha-a}\frac{\beta-b}{b\beta} e^{\frac{\beta-b}{b\beta}x}\right]$$

$$w'(x) = \frac{\Gamma(a)b^a}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-a-1} e^{\frac{\beta-b}{b\beta}x} \left[(\alpha - a) + \frac{\beta-b}{b\beta}x\right]$$

Hence :

$$w'(x) \geq 0 \quad \Leftrightarrow \quad \alpha - a \geq \frac{b-\beta}{b\beta}x$$

Since $b - \beta = 0.8 > 0$ and $b\beta > 0$:

$$w'(x) \geq 0 \quad \Leftrightarrow \quad \frac{(\alpha - a)b\beta}{b-\beta} \geq x$$

x	0	$\frac{(\alpha-a)b\beta}{b-\beta}$	$+\infty$
$w(x)$	0 ↗	M	↘ 0

Where $M = w(\frac{(\alpha-a)b\beta}{b-\beta}) = \frac{\Gamma(a)b^a}{\Gamma(\alpha)\beta^\alpha}(\frac{(\alpha-a)b\beta}{b-\beta})^{\alpha-a}e^{a-\alpha}$

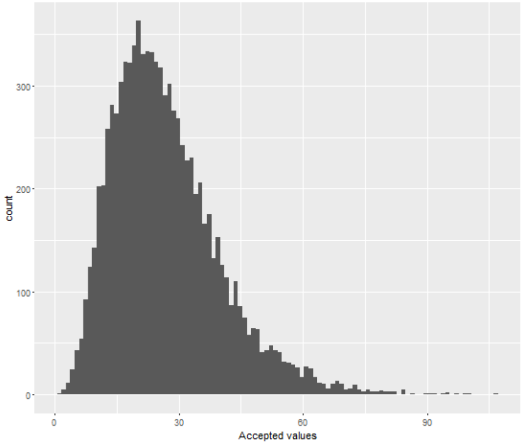
Therefore, we found M such that $\forall x > 0, \quad f_{\alpha,\beta}(x) \leq M' f_{a,b}(x)$.

Note that in practice we do not know $\Gamma(\alpha)$ and $\Gamma(a)$ exactly. However, they simplify when we compute the conditional acceptance probability :

$$\frac{f(y)}{Mg(y)} = \frac{w(y)}{M} = \frac{\Gamma(a)b^a}{\Gamma(\alpha)\beta^\alpha}y^{\alpha-a}e^{\frac{\beta-b}{b\beta}y} \times \frac{\Gamma(\alpha)\beta^\alpha}{\Gamma(a)b^a}(\frac{b-\beta}{(\alpha-a)b\beta})^{\alpha-a}e^{\alpha-a}$$
$$= y^{\alpha-a}e^{\frac{\beta-b}{b\beta}y}(\frac{b-\beta}{(\alpha-a)b\beta})^{\alpha-a}e^{\alpha-a}$$

R Code The corresponding R code is available in Appendix.

Figure 1 – Distribution of the accepted simulations : Gamma(4.3,6.2) distribution



Results I runned 10000 simulations from the instrumental distribution. The empirical acceptance rate was : 0.9198 which is indeed close to $\frac{1}{M} \approx 0.9178$ (the unconditional acceptance probability). Note that this is a very high acceptance rate and this is more generally explained by the fact that the instrumental distribution is very close here to the target distribution.

The final value of the Gamma(4.3,6.2) mean using Accept-Reject was 26.89. Figure 1 displays the histogram of the accepted simulations, which indeed looks like a Gamma distribution.

The convergence speed of the mean will be compared to other methods in question (c).

(b) Metropolis-Hastings with a Gamma(4,7) candidate

Reminder - Theorem 7.8 Let f be the target distribution and g be the instrumental distribution. The Independent Metropolis-Hastings algorithm produces an uniformly ergodic chain if there exists a constant M such that :

$$\forall x \in Supp(f), \quad f(x) \leq Mg(x)$$

On the other hand, if for every M there exists a set of positive measure where this inequation does not hold, then the resulting Markov chain is not even geometrically ergodic.

Applying Theorem 7.8 to our case In our case where $f = f_{\alpha,\beta}$ and $g = f_{a,b}$, we showed in (a) that there exists M' such that $\forall x \in Supp(f), \quad f(x) \leq Mg(x)$. Hence, the Metropolis-Hastings algorithm will necessarily produce a uniformly ergodic chain, meaning that the convergence will be "very fast".

Acceptance probability in this Independent Metropolis-Hastings Using the same notations as previously, the acceptance probability in this Metropolis-Hastings algorithm will be :

$$\rho(x,y) = \min\Big(1, \frac{f(y)g(x)}{f(x)g(y)}\Big)$$

$$\rho(x,y) = \min\Big(1, \frac{f_{\alpha,\beta}(y)f_{a,b}(x)}{f_{\alpha,\beta}(x)f_{a,b}(y)}\Big)$$

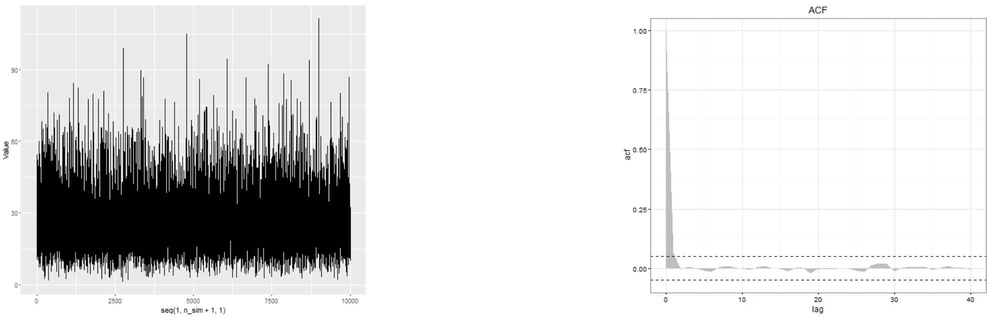
$$\rho(x,y) = \min\Big(1, \frac{w(y)}{w(x)}\Big)$$

$$\rho(x,y) = \min\Big(1, \frac{\Gamma(a)b^a}{\Gamma(\alpha)\beta^\alpha} y^{\alpha-a} e^{\frac{\beta-b}{b\beta}y} \times \frac{\Gamma(\alpha)\beta^\alpha}{\Gamma(a)b^a} \frac{1}{x^{\alpha-a} e^{\frac{\beta-b}{b\beta}x}}\Big)$$

$$\rho(x,y) = \min\Big(1, \frac{y^{\alpha-a} e^{\frac{\beta-b}{b\beta}y}}{x^{\alpha-a} e^{\frac{\beta-b}{b\beta}x}}\Big)$$

R Code The corresponding R Code is available in Appendix.

Results Figure 2 (a) shows the evolution of the Metropolis Markov chain after 10 000 candidate simulations. The acceptance rate was 0.9457, which is very high. It can here also be explained by the fact that the candidate distribution is very close to the target distribution. The value of the mean of the generated Markov chain , which should approximate the mean of a Gamma(4.3,6.2) due to the convergence of the chain to its stationary distribution, is 26.75. This is thus similar to the result we obtained in (a). Figure 2 (b) is the plot of the Autocorrelation Function of the Metropolis Markov chain. We can see here that autocorrelation disappears after a very low number of simulations (about 1 or 2), so the mixing is very good. This is in line with a very high acceptance rate, which shows the chain does not stay too much time "blocked" at the very exact same state.



(a) Parameter a

(b) Autocorrelation Function (ACF)

Figure 2 – Results of the Metropolis-Hastings 1 (with a Gamma(4,7) candidate)

(c) Metropolis-Hastings with a Gamma(5,6) candidate

Conditions to use the Independent Metropolis-Hastings algorithm Part 7.4.1 of the book tells us that the resulting chain $(X^{(t)})$ is irreducible and aperiodic (thus ergodic using Corollary 7.5) if and only if g is almost everywhere positive on the support of f. As, in our case, the support of f and the support of g are the same (equal to $\mathbb{R}_+^*$), this condition is necessarily verified. **Hence, the convergence properties of the resulting chain are guaranteed.**

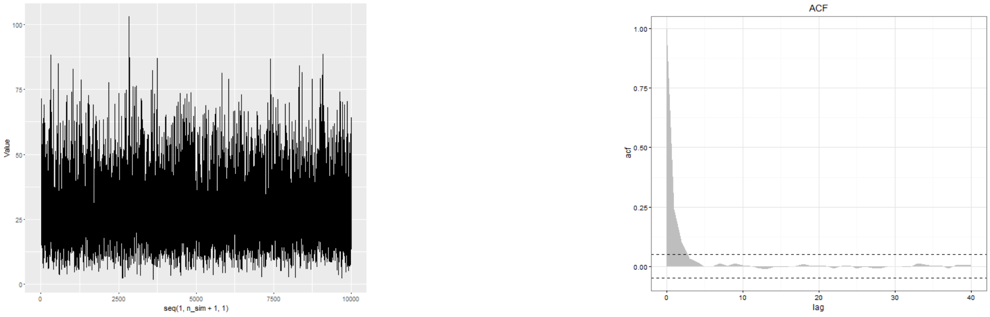
Applying Theorem 7.8 to our case In our case, where $f = f_{\alpha,\beta}$ and $g = f_{a,b}$, we showed in (a) that the ratio $\frac{f}{g}$ is equal to :

$$w(x) = \frac{f_{\alpha,\beta}(x)}{f_{a,b}(x)} = \frac{\Gamma(a)b^a}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-a} e^{\frac{\beta-b}{b\beta}x}$$

Since here $b = 6 < \beta = 6.2$, $\lim\limits_{x\to+\infty} w(x) = +\infty$ even if $a = 5 > \alpha = 4.3$ (because exponential is "stronger"). It is thus impossible to find an upper bound to the ratio. Following Theorem 7.8, the resulting Markov chain will not even be geometrically ergodic. **We should therefore observe a pretty much slower convergence here, with the Gamma(5,6) candidate, than in (b) with the Gamma(4,7) candidate.**

R Code The corresponding R Code is available in Appendix.

Results Figure 3 (a) shows the evolution of the resulting Markov chain after 10 000 candidate simulations. The acceptance rate was 0.8428, which is still very high but lower than the one obtained with the previous Metropolis-Hastings. It can be explained by the fact that the candidate distribution is not as close to the target distribution as the candidate in (b), as Figure 4 shows it. The value of the mean of the generated Markov chain is 26.50, which is also in line with the results obtained in (a) and in (b). Figure 3 (b) shows us that the mixing is still very good, but not as good as when using the previous candidate distribution, since the autocorrelations disappear after a bit more steps than previously. This is in line with the obtained lower acceptance rate.

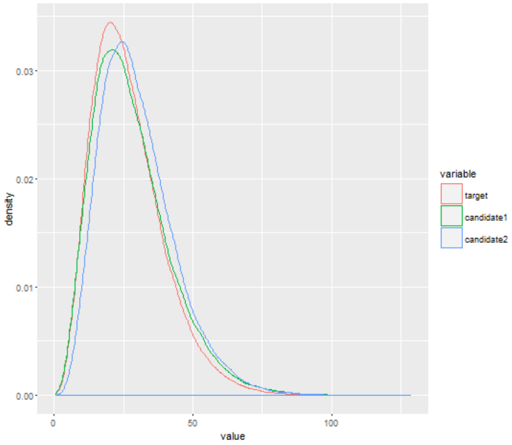


(a) Parameter a

(b) Autocorrelation Function (ACF)

Figure 3 – Results of the Metropolis-Hastings 2 (with a Gamma(5,6) candidate)

Figure 4 – Comparison of the target density (drawn directly using rgamma in R) and both candidate densities



Comparison between the 3 methods

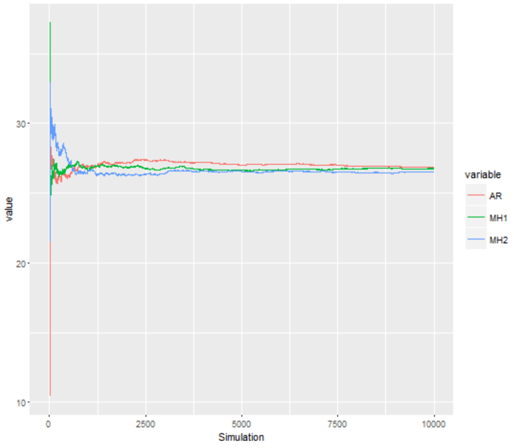
Figure 5 compares the convergence of the 3 estimators to the mean of a Gamma(4.3, 6.2). We see that the 3 algorithms tend to deliver similar results only after about 1250 candidate simulations. The second Metropolis-Hastings "MH2" (Metropolis-Hastings with candidate Gamma(5,6)) seems to have a slower convergence than the first one (with candidate Gamma(4,7)) and this seems in line with Theorem 7.8. However, this difference is actually not always visible when we run the code several times. Therefore, the only empirical difference we "always" observe is a lower acceptance rate for "MH2", which is of course still an important sign of slower convergence. Accept-Reject and Metropolis-Hastings with candidate Gamma(4,7) seem to deliver very similar results in terms of convergence speed.

7.4 Gaussian transition kernel

Let the transition kernel :

$$X^{(t+1)}|x^{(t)} \sim \mathcal{N}(\rho x^{(t)}, \tau^2)$$

Figure 5 – Mean Convergence using AR (Accept-Reject), MH1 (Metropolis-Hastings with Gamma(4,7) candidate) and MH2 (Metropolis-Hastings with Gamma(5,6) candidate)



This means that the transition kernel is more explicitly :

$$K(x^{(t)},x)=\frac{1}{\sqrt{2\pi}\tau}e^{-\frac{(x-\rho x^{(t)})^2}{2\tau^2}}$$

Sufficient condition for the stationary distribution π to exist Theorem 6.46 tells us that if a Markov Chain with transition kernel K satisfies the detailed balance condition with π a probability density function (i.e : there exists π a probability density function such that $\forall(x,y) \quad \pi(x)K(x,y)=\pi(y)K(y,x)$) then the density π is the invariant density of the chain.

Hence, a sufficient condition for the stationary distribution to exist is that there exists a probability density function π verifying the detailed balance condition. More precisely, if there exists a probability density function π verifying the detailed balance condition, then : 1) the stationary distribution does exist 2) the stationary distribution is actually π.

Find a density π such that the detailed balance condition is verified

$$\begin{aligned}
& \forall(x,y) \quad \pi(x)K(x,y)=\pi(y)K(y,x)\\
\Leftrightarrow \quad & \forall(x,y) \quad \frac{\pi(x)}{\pi(y)}=\frac{K(y,x)}{K(x,y)}\\
\Leftrightarrow \quad & \forall(x,y) \quad \frac{\pi(x)}{\pi(y)}=\frac{\frac{1}{\sqrt{2\pi}\tau}e^{-\frac{(x-\rho y)^2}{2\tau^2}}}{\frac{1}{\sqrt{2\pi}\tau}e^{-\frac{(y-\rho x)^2}{2\tau^2}}}\\
\Leftrightarrow \quad & \forall(x,y) \quad \frac{\pi(x)}{\pi(y)}=e^{\frac{(y-\rho x)^2}{2\tau^2}-\frac{(x-\rho y)^2}{2\tau^2}}\\
\Leftrightarrow \quad & \forall(x,y) \quad \frac{\pi(x)}{\pi(y)}=e^{\frac{1}{2\tau^2}[y^2-2\rho xy+\rho^2x^2-x^2+2\rho xy-\rho^2y^2]}\\
\Leftrightarrow \quad & \forall(x,y) \quad \frac{\pi(x)}{\pi(y)}=e^{\frac{1}{2\tau^2}[(1-\rho^2)y^2-(1-\rho^2)x^2]}\\
\Leftrightarrow \quad & \forall(x,y) \quad \frac{\pi(x)}{\pi(y)}=e^{\frac{1-\rho^2}{2\tau^2}[y^2-x^2]}\\
\Leftrightarrow \quad & \forall(x,y) \quad \frac{\pi(x)}{\pi(y)}=\frac{e^{-\frac{1-\rho^2}{2\tau^2}x^2}}{e^{-\frac{1-\rho^2}{2\tau^2}y^2}}
\end{aligned}$$

Hence, if we take $\pi(x)=\frac{\sqrt{1-\rho^2}}{\sqrt{2\pi}\tau}e^{-\frac{1-\rho^2}{2\tau^2}x^2}$, i.e we take π equal to the density of a $\mathcal{N}(0,\frac{\tau^2}{1-\rho^2})$, the detailed balance condition will be verified. This π is actually well defined as a probability density distribution if and only if $\tau^2>0$ and $1-\rho^2>0 \quad \Leftrightarrow \quad |\rho|<1$.

Conclusion : sufficient condition on ρ and τ for the stationary distribution to exist and value of the stationary distribution in this case We thus showed that a sufficient condition for the stationary distribution π to exist is $\begin{cases} |\rho| < 1 \\ \tau^2 > 0 \end{cases}$. If this condition is verified, the density of the normal distribution $\mathcal{N}(0, \frac{\tau^2}{1-\rho^2})$ will indeed verify the detailed balance condition. Using Theorem 6.46, this has 2 consequences : 1) The stationary distribution does exist 2) The stationary distribution is the normal distribution $\mathcal{N}(0, \frac{\tau^2}{1-\rho^2})$.

Show that in this case, (7.4) occurs, i.e : show that the probability of $\{X^{(t+1)} = X^{(t)}\}$ is zero

$$\begin{aligned} P\{X^{(t+1)} = X^{(t)}\} &= E(\mathbb{1}_{X^{(t+1)}-X^{(t)}=0}) \\ &= E\big(E(\mathbb{1}_{X^{(t+1)}-X^{(t)}=0}|X^{(t)} = x^{(t)})\big) \\ &= E\big(P\{X^{(t+1)} = x^{(t)}|X^{(t)} = x^{(t)}\}\big) \end{aligned}$$

Since $X^{(t+1)}|x^{(t)}$ has a continuous distribution :

$$= E(0) = 0$$

Therefore (7.4) occurs. In the Metropolis-Hastings context, i.e if $(X^{(t)})$ is the Metropolis-chain resulting from a Metropolis-Hastings algorithm, this means that we **always** accept the proposal Y_t.

Why does (7.4) occurs in this example ? This comes from the fact that the transition Kernel we are considering in this exercise is continuous, so it does not involve any Dirac delta function, or in other words it does not put any mass on the point $x^{(t)}$.

7.9 Using Metropolis-Hastings to correct an Accept-Reject algorithm applied to a ratio without any uniform bound (Tierney 1994)

Consider a version based on a "bound" M on $\frac{f}{g}$ that is not a uniform bound; that is, $\frac{f(x)}{g(x)} > M$ for some $x \in A$, where A is of strictly positive measure.

N.B : Since we divided by g, it has been implicitly assumed that g is strictly positive on the support of f. In other words, $Supp(f) \subset Supp(g)$.

(a) Accept-Reject result

Assume an Accept-Reject algorithm is used with candidate g and acceptance probability $\frac{f(y)}{Mg(y)}$. What will then be the resulting distribution ?

We will show that a carefully chosen Accept-Reject algorithm enabling us to generate variables from $\tilde{f} \propto \min(f, Mg)$ is exactly the same as this one. Therefore, we will be able to conclude that the resulting variables from this Accept-Reject are generated from $\tilde{f}$.

Step 1 : Find an upper bound and a candidate distribution to be able to use Accept-Reject in order to simulate from $\tilde{f}$

$$\forall y, \quad \min(f(y), Mg(y)) \leq Mg(y)$$

Accept-Reject is still valid when the density of interest is only known up to a multiplicative factor (see my Homework 2 for the detailed proof or see Chapter 2 of the book). Hence, we can implement an Accept-Reject in order to simulate from $\tilde{f}$ by using the candidate distribution g and the probability of acceptance $\frac{\min(f(y),Mg(y))}{Mg(y)} = \min(\frac{f(y)}{Mg(y)}, 1)$.

Step 2 : See that this leads to the desired Accept-Reject algorithm The Accept-Reject found in Step 1 to simulate from $\tilde{f}$ is more explicitly :

$$Y \sim g \quad \text{and} \quad U|Y = y \sim U(0, 1),$$

$$\text{until} \quad 0 < u < \min(\frac{f(y)}{Mg(y)}, 1) \quad \Leftrightarrow \quad \text{until} \quad 0 < u < \frac{f(y)}{Mg(y)}, \quad \text{since } u \text{ is always lower than } 1$$

Therefore, we recognize the desired Accept-Reject algorithm mentioned at the beginning of the exercise.

Conclusion : The Accept-Reject algorithm with candidate g and acceptance probability $\frac{f(y)}{Mg(y)}$ generates variables following the distribution $\tilde{f} \propto \min(f, Mg)$.

(b) Metropolis-Hastings correction

Let us use $\tilde{f}$ as the instrumental distribution (since we are able to generate such variables easily with the Accept-Reject mentioned in (a)) in an Independent Metropolis-Hastings algorithm with target distribution f. This will enable us to obtain a sample from f, so to finally correct the error.

Convergence conditions Condition (7.5) is verified, i.e $Supp(f) \subset Supp(\tilde{f})$ since $Supp(f) \subset Supp(g)$ and $\tilde{f} = min(f, Mg)$. Condition (7.4) follows from the fact that there exist a set of strictly positive measure where $f(x) > Mg(x)$. Hence, our Metropolis-Hastings algorithm should converge.

Independent Metropolis-Hastings with target f and instrumental distribution $\tilde{f}$: The algorithm is :

Given $x^{(t)}$

1. Generate $Y_t \sim \tilde{f}$ (using the Accept-Reject algorithm of question (a))

2. Take :

$$X^{(t+1)} = \begin{cases} Y_t \text{ with probability } \min(1, \frac{f(Y_t)\tilde{f}(x^{(t)})}{f(x^{(t)})\tilde{f}(Y_t)}) \\ x^{(t)} \text{ otherwise} \end{cases}$$

Since $\tilde{f} \propto \min(f, Mg)$, there exists a constant c such that : $\tilde{f} = c\min(f, Mg)$. So the acceptance probability can be written :

$$\begin{aligned} \min(1, \frac{f(Y_t)\tilde{f}(x^{(t)})}{f(x^{(t)})\tilde{f}(Y_t)}) &= \min(1, \frac{f(Y_t)c\min(f(x^{(t)}), Mg(x^{(t)}))}{f(x^{(t)})c\min(f(Y_t), Mg(Y_t))}) \\ &= \min(1, \frac{f(Y_t)\min(f(x^{(t)}), Mg(x^{(t)}))}{f(x^{(t)})\min(f(Y_t), Mg(Y_t))}) \end{aligned}$$

There are therefore 2 cases : either $f(Y_t) > Mg(Y_t)$ (case 1) or $f(Y_t) \leq Mg(Y_t)$ (case 2)

Probability of acceptance in case 1 If $f(Y_t) > Mg(Y_t) \quad \Leftrightarrow \quad \frac{f(Y_t)}{g(Y_t)} > M$, then :

$$\min(f(Y_t), Mg(Y_t)) = Mg(Y_t)$$

Therefore, the probability of acceptance is :

$$\begin{aligned} \min\left(1, \frac{f(Y_t)}{f(x^{(t)})Mg(Y_t)}\min(f(x^{(t)}), Mg(x^{(t)}))\right) &= \min\left(1, \min(\frac{f(Y_t)}{Mg(Y_t)}, \frac{f(Y_t)g(x^{(t)})}{f(x^{(t)})g(Y_t)})\right) \\ &= \min\left\{1, \frac{f(Y_t)}{Mg(Y_t)}, \frac{f(Y_t)g(x^{(t)})}{f(x^{(t)})g(Y_t)}\right\} \end{aligned}$$

Since $\frac{f(Y_t)}{Mg(Y_t)} > 1$ (this is the hypothesis of case 1) :

$$= \min\left(1, \frac{f(Y_t)g(x^{(t)})}{f(x^{(t)})g(Y_t)}\right)$$

Probability of acceptance in case 2 If $f(Y_t) \leq Mg(Y_t) \quad \Leftrightarrow \quad \frac{f(Y_t)}{g(Y_t)} \leq M$, then :

$$\min(f(Y_t), Mg(Y_t)) = f(Y_t)$$

Therefore, the probability of acceptance is :

$$\begin{aligned}
\min\Big(1, \frac{f(Y_t)}{f(x^{(t)})f(Y_t)}\min(f(x^{(t)}), Mg(x^{(t)}))\Big) &= \min\Big(1, \frac{1}{f(x^{(t)})}\min(f(x^{(t)}), Mg(x^{(t)}))\Big) \\
&= \min\Big(1, \min(\frac{f(x^{(t)})}{f(x^{(t)})}, \frac{Mg(x^{(t)})}{f(x^{(t)})})\Big) \\
&= \min\Big(1, \min(1, \frac{Mg(x^{(t)})}{f(x^{(t)})})\Big) \\
&= \min\Big\{1, 1, \frac{Mg(x^{(t)})}{f(x^{(t)})}\Big\} \\
&= \min\Big(1, \frac{Mg(x^{(t)})}{f(x^{(t)})}\Big)
\end{aligned}$$

Conlusion The final Metropolis correction algorithm is therefore :

Given $x^{(t)}$

1. Generate $Y_t \sim \tilde{f}$ (using the Accept-Reject algorithm of question (a))

2. Accept with probability :

$$P(X^{(t+1)} = y_t | x^{(t)}, y_t) = \begin{cases} \min\Big(1, \frac{f(y_t)g(x^{(t)})}{f(x^{(t)})g(y_t)}\Big) \text{ if } f(y_t) > Mg(y_t) \\ \min\Big(1, \frac{Mg(x^{(t)})}{f(x^{(t)})}\Big) \text{ if } f(y_t) \leq Mg(y_t) \end{cases}$$

7.21 Posterior sampling on the "braking data" of Tukey using Metropolis-Hastings

The goal of the exercise is to simulate from the posterior distribution of the parameters of the following quadratic model applied to the "braking data" :

$$y_{ij} = a + bx_i + cx_i^2 + \epsilon_{ij} \quad \text{where } \epsilon_{ij} \sim \mathcal{N}(0, \sigma^2) \text{ i.i.d}, \quad i = 1, ..., k \quad j = 1, ..., n_i$$

The likelihood function of this model is thus easily computable[1].

Preliminary explanation : How can Metropolis-Hasting be used to sample from the posterior ? What is the link with the likelihood ? There are indeed several ways to understand the statement "We can view this likelihood function as a posterior of a, b, c and σ^2 and we can sample from it with a Metropolis–Hastings algorithm". I chose the following classic Bayesian linear regression approach : my goal was to simulate the parameters from the posterior after carefully choosing a proper prior which provides a "realistic knowledge" on the parameters (Purpose of questions (a) and (b) as I understood them, since the Inverse-Gamma distribution is a classic prior in such linear regression models with gaussian error terms). I will show that using this approach, if we also use the prior distribution as the candidate distribution, the link with the likelihood is clear, since the acceptance probability of the Metropolis-Hastings algorithm will only depend on the likelihood of the candidate parameters.

However, I recognize that a second possible approach would have been to consider the likelihood directly as the distribution of interest of the parameters, after checking if it is integrable wrt the parameters (otherwise we cannot consider it as a proper probability distribution). This is equivalent to consider an improper prior on the vector of parameters constant equal to 1 and a posterior equal to the likelihood. The good thing in this approach is that this prior is uninformative. The drawback to me is that it is harder to interpret, since we are not considering "true" probability distributions.

Nevertheless, both approaches deliver the same form of results : we get an idea of the distribution of the possible parameters which could fit the data well whereas in a standard MLE approach (like in the linear regression of question (a)), we only get a point estimate. In both cases, we should find that the MLE region has a high "posterior" probability using both definitions of "posterior".

[1] There is a typo in the exercise : a minus is missing in the exponential of the likelihood

Model and corresponding Metropolis-Hastings algorithm Let $\theta = (a, b, c, \sigma^2)$. We first need to choose a prior $\pi(\theta)$ on these parameters with hyperparameters that are realistic according to the observed data (this is the purpose of question (a) and (b)).
The main idea is then to use the prior $\pi(\theta)$ as the instrumental distribution in a Metropolis-Hasting algorithm and the posterior $\pi(\theta|y)$ as the target (since it is our goal to sample from the posterior). There will then be some simplifications, and we will be able to simulate from this target by using only the form of the likelihood. We will therefore only accept candidates depending on the likelihood.
In order to understand, let us write this Metropolis-Hasting algorithm precisely (where y is the observed data) :

Given $\hat{\theta}^{(t)}$

1. Generate $U_t \sim \pi(\theta)$ (the chosen prior)

2. Take :

$$\hat{\theta}^{(t+1)} = \begin{cases} U_t \text{ with probability } \rho(\hat{\theta}^{(t)}, U_t) = \min(1, \frac{\pi(U_t|y)\pi(\hat{\theta}^{(t)})}{\pi(\hat{\theta}^{(t)}|y)\pi(U_t)}) \\ \hat{\theta}^{(t)} \text{ otherwise} \end{cases}$$

Using Bayes' theorem :

$$\pi(\theta|y) = \frac{\pi(\theta)L(y;\theta)}{\pi(y)} \text{ where } L(y;\theta) \text{ is the likelihood of the model}$$

Therefore, we can write the acceptance probability as :

$$\begin{aligned} \rho(\hat{\theta}^{(t)}, U_t) &= \min\left(1, \frac{\pi(U_t)L(y;U_t)\pi(y)\pi(\hat{\theta}^{(t)})}{\pi(\hat{\theta}^{(t)})L(y;\hat{\theta}^{(t)})\pi(y)\pi(U_t)}\right) \\ &= \min\left(1, \frac{L(y;U_t)}{L(y;\hat{\theta}^{(t)})}\right) \end{aligned}$$

As we know the likelihood exactly and are able to compute it easily, we thus showed how to simulate from the posterior using Metropolis-Hastings.

Note that the advantage of this algorithm is to have a very easy and straightforward interpretation : we accept the candidate parameter if it fits the observed data better than the previously generated parameter (i.e : if the likelihood is higher). If not, we rather keep the previous one. The drawback, as we will see after, is that it is hard to calibrate, since the choice of the prior-candidate has a big impact of the acceptance rate.

(a) First parameter estimation using linear regression

Table 1 displays the estimates of the linear regression.

Table 1 – Results of the linear regression

	a	b	c	σ^2
Estimate	2.47	0.913	0.0999	230

R code The R code to obtain the linear regression result is :

```
################################# Problem 7.21 #################################
###############################################################################

#Create the dataframe containing the braking data
x ← c(rep(4,2), rep(7,2), 8, 9, rep(10, 3), rep(11,2), rep(12,4),rep(13,4),
       rep(14,4), rep(15,3), rep(16,2), rep(17,3), rep(18,4), rep(19,3),
       rep(20, 5), 22, 23, rep(24,4), 25)
y← c(2,10, 4,22, 16, 10, 18,26, 34, 17,28, 14,20, 24,28, 26,34, 34,46, 26,36,
   60,80, 20,26, 54,
     32,40, 32,40, 50, 42,56, 76,84, 36,46, 68, 32,48, 52,56,64, 66, 54, 70, 92,
        93, 120, 85)

braking_data ← data.frame(x = x, y = y, x2 = x**2)

```

```
#### Question (a) : Linear regression

lin_reg ← lm(y ∼ x + x2, data = braking_data)
summary(lin_reg)

coefficients_vector ← coefficients(lin_reg)
sigma ← summary(lin_reg)$sigma

theta_est_linreg ← c(coefficients_vector,sigma**2)
theta_est_linreg
```

(b) Candidate distribution selection (i.e Prior selection in our case) using the previous estimation

The linear regression model assumes that there is a "true parameter", or, from a Bayesian point of view, that the parameters are constant. In this Problem, we will relax this assumption and, as explained before, assume that the parameters are random variables which are not constant, our goal being to sample from their posterior distribution.

Therefore, we can view the linear regression estimates as a first estimation of their posterior expectation. A natural choice for the prior is thus to take these estimates as the mean of the prior. Hence :

- We will use the linear regression estimates as the mean parameter of a Normal distribution for a, b and c. The variance will be first fixed to 1, but we will then adjust it if the acceptance probability is far too low or far too high.
- We will view the linear regression estimate of σ^2 as the mean of an Inverse Gamma distribution (Note that the Inverse Gamma distribution has a positive support, so it makes sense to use it as a prior for a variance parameter). Since the mean of an Inverse Gamma(α,rate $=\beta$) is $\frac{\beta}{\alpha-1}$, we will thus use $\alpha = 1 + \frac{\beta}{\hat{\sigma}^2_{OLS}}$ where $\hat{\sigma}^2_{OLS}$ is the linear regression estimate, and adjust β in order to get simulations that are rather close to this mean. Note that simulating from an Inverse Gamma distribution can be done easily by simulating from a Gamma distribution and take the inverse since $X \sim \text{Gamma}(\alpha, \text{rate} = \beta) \quad \Leftrightarrow \quad \frac{1}{X} \sim \text{Inv-Gamma}(\alpha, \text{rate} = \beta)$.

Remark first that the linear regression estimate of σ^2 is very high (equal to 230), so it will be tricky to get simulations from an Inverted Gamma that often fall close to this value due to the form of this distribution. Since the linear regression estimates maximize the likelihood and since our acceptance rate only depends on the likelihood, we will therefore have troubles to achieve a good acceptance rate if we cannot simulate points that are close to the linear regression estimates. In practice, I had a 0 acceptance when I ran the algorithm for the first time using default hyperparameters $\sigma_a^2 = 1$, $\sigma_b^2 = 1$, $\sigma_c^2 = 1$ and rate $\beta = 1$! I took a larger rate parameter in order to get a large tail, and hence to get simulations with a higher value. However, this came at a cost : many simulations are also very far from 230 (see Table 2).

Table 2 – Candidate simulations for σ^2 using $\beta = 250$

Using $\beta = 250$	Min	1st Quantile	Median	Mean	3rd Quantile	Max
Candidate simulations for σ^2	16.7	89.6	141.7	233	244.4	65740

I then adjusted the other variance parameters σ_a^2, σ_b^2 and σ_c^2 until I was not able to improve the acceptance rate.

My final candidate distribution / prior was therefore :

- $a \sim \mathcal{N}(\hat{a}_{OLS}, 0.5)$
- $b \sim \mathcal{N}(\hat{b}_{OLS}, 0.05)$
- $c \sim \mathcal{N}(\hat{c}_{OLS}, 0.01)$
- $\sigma^2 \sim \text{Inv-Gamma}(1 + \frac{250}{\hat{\sigma}^2_{OLS}}, \text{rate} = 250)$

Note that the choice of a Normal-Inverse Gamma prior is actually a classic one, since it is a conjugate prior in this model. In practice, we are thus here simulating a posterior that we could entirely derive explicitly.

R code of the Metropolis-Hastings algorithm The corresponding R code is :

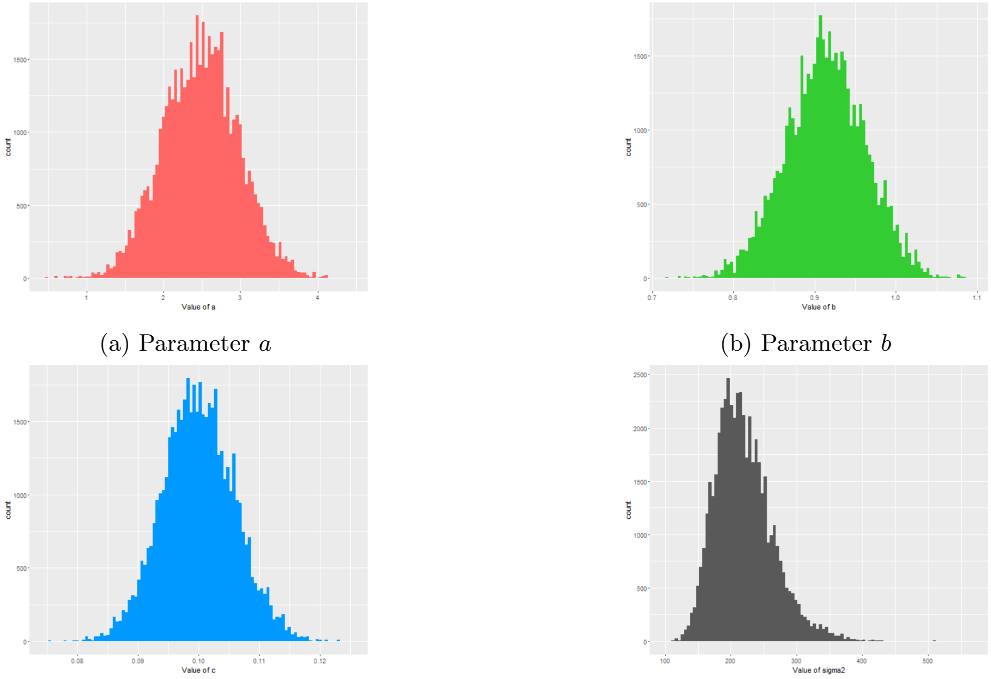
```r
### Question (b) & (c) : Metropolis-Hastings

#compute the log likelihood on the braking data for a given parameter theta (since the likelihood is too small for R !)
log_likelihood ← function(theta){
  a ← theta[1]
  b ← theta[2]
  c ← theta[3]
  sigma2 ← theta[4]
  N = dim(braking_data)[1]
  vector_for_sum ← (braking_data$y - a - b*braking_data$x -c*braking_data$x2)**2
  result = (N/2)*log(1/sigma2) -(1/(2*sigma2))*sum(vector_for_sum)
  return(result)
}

#compute the acceptance probability
rho_accept_log ← function(x,y){
  quotient ← log_likelihood(y) - log_likelihood(x)
  result ← min(1, quotient)
  return(result)
}


#Independent Metropolis Hastings with candidate = prior (here 3 Normal distributions and 1 Inverse gamma)
independent_metropolis_hastings_posterior ←function(n_sim, theta_est_linreg, var_a, var_b, var_c, beta_sigma2){
  a ← rnorm(n_sim, theta_est_linreg[1], var_a)
  b ← rnorm(n_sim, theta_est_linreg[2], var_b)
  c ← rnorm(n_sim, theta_est_linreg[3], var_c)
  shape_inv_gamma = 1 + (beta_sigma2/theta_est_linreg[4])
  sigma2 ← 1/rgamma(n_sim, shape = shape_inv_gamma, rate = beta_sigma2)
  y ←cbind(a,b,c,sigma2)
  #arbitrary initialization
  x ← t(as.matrix(theta_est_linreg))
  acceptance ← 0
  for (k in 1:n_sim){
    u ← runif(1,0,1)
    rho ← rho_accept_log(x[k,],y[k,])
    if (log(u) < rho){
      x ← rbind(x,y[k,])
      acceptance ← acceptance + 1
    } else{
      x←rbind(x, x[k,])
    }
  }
  acceptance ← acceptance / n_sim
  return(list(x, acceptance))
}
```

(c) Metropolis-Hastings results

As explained previously, I had trouble to reach a reasonable acceptance rate, in particular due to the choice of using an Inverted Gamma as a prior for σ^2. However, I finally reached a 18.40 % acceptance rate, which is enough to achieve convergence in a reasonable time.

Figure 6 displays the histograms of the posterior distributions after running 50 000 candidate simulations. Figure 7 shows the resulting Metropolis Markov chain for each parameter. As expected, the posterior distribution are centered on values that are close to the OLS estimates. This is not surprising, since the OLS estimators of the linear regression coefficients are also Maximum Likelihood Estimators in this model and since the Maximum Likelihood Estimator for σ^2 is the unadjusted sample variance of the residuals, which is pretty close to the OLS estimator (the adjusted sample variance of the residuals).

Finally note that we can indeed recognize the form of 3 normal distributions and an Inverse-Gamma for σ^2. As explained before, this is not surprising, since we chose a conjugate prior.



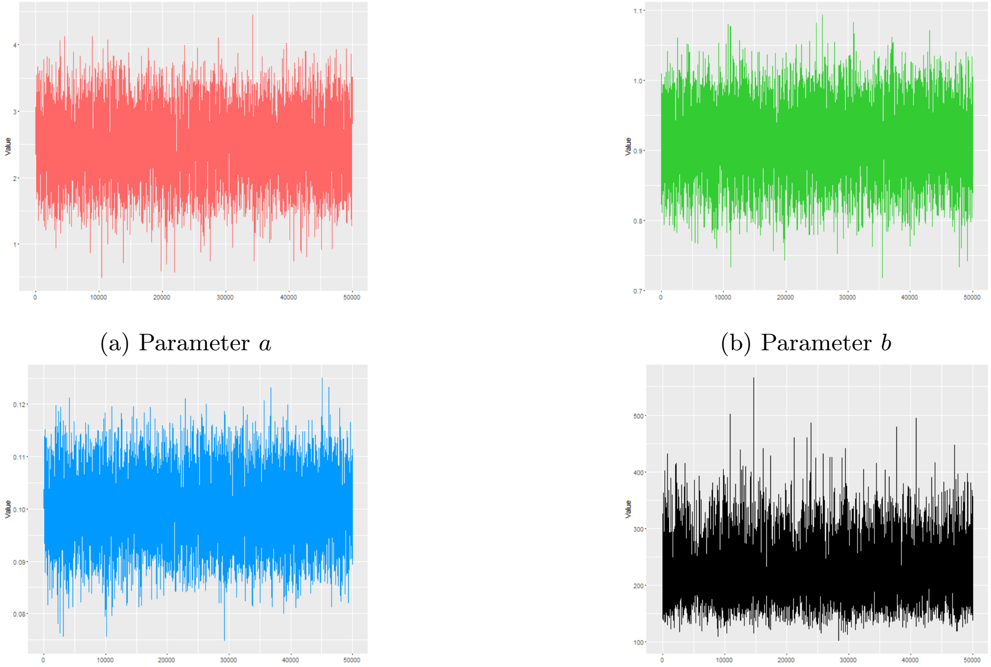
(a) Parameter a

(b) Parameter b

(c) Parameter c

(d) Parameter σ^2

Figure 6 – Histogram of posterior distributions



(a) Parameter a

(b) Parameter b

(c) Parameter c

(d) Parameter σ^2

Figure 7 – Plot of the Metropolis Markov Chain of each parameter

R code to get the results The R code to run the algorithm and plot the results is :

```
n_sim= 50000
results ← independent_metropolis_hastings_posterior(n_sim = n_sim,
    theta_est_linreg = theta_est_linreg, var_a= 0.5, var_b= 0.05, var_c = 0.01,
    beta_sigma2 = 250)
theta_markov ← matrix(unlist(results[1]), ncol = 4, byrow = FALSE)
acceptance_rate ← unlist(results[2])
print(acceptance_rate)

library(ggplot2)
```

```
#Plot the posterior
#Posterior of a
qplot( theta_markov[,1], geom = 'histogram', bins = 100, fill = I("#FF6666"), xlab = 'Value of a')
qplot(seq(1,n_sim + 1,1), theta_markov[,1], geom = 'line', ylab = 'Value' ,xlab = "", colour =I("#FF6666"))
#Posterior of b
qplot( theta_markov[,2], geom = 'histogram', bins = 100, fill = I("#33CC33"), xlab = 'Value of b' )
qplot(seq(1,n_sim + 1,1), theta_markov[,2], geom = 'line', ylab = 'Value', xlab = "", colour =  I("#33CC33")  )
#Posterior of c
qplot( theta_markov[,3], geom = 'histogram', bins = 100, fill = I("#0099FF"), xlab = 'Value of c' )
qplot(seq(1,n_sim + 1,1), theta_markov[,3], geom = 'line', ylab = 'Value', xlab = "", colour = I("#0099FF") )
#Posterior of sigma2
qplot( theta_markov[,4], geom = 'histogram', bins = 100 , xlab = 'Value of sigma2' )
qplot(seq(1,n_sim + 1,1), theta_markov[,4], geom = 'line', ylab = 'Value', xlab = "" )
```

(d) Modification of the error distribution assumption in order to improve robustness

R code Here is the R code, with the best parameters I found for each Metropolis-Hastings (with Normal Inverse Gamma prior and with t and half-t prior) :

```
### Question (d) : Metropolis-Hastings for the second model

#compute the log likelihood on the braking data for a given parameter theta (since the likelihood is too small for R !)
log_likelihood2 ← function(theta, nu = 4){
  a ← theta[1]
  b ← theta[2]
  c ← theta[3]
  sigma2 ← theta[4]
  N = dim(braking_data)[1]
  vector_for_sum ← 1 + ((braking_data$y - a - b*braking_data$x -c*braking_data$x2)**2)/nu
  result = (N/2)*log(1/sigma2) -((nu + 1)/2)*sum(sapply(vector_for_sum, log))
  return(result)
}

log_likelihood2(theta_est_linreg)

#compute the acceptance probability
rho_accept_log2 ← function(x,y){
  quotient ← log_likelihood2(y) - log_likelihood2(x)
  result ← min(1, quotient)
  return(result)
}


#Independent Metropolis Hastings with candidate = prior
#NORMAL INVERSE GAMMA PRIOR
independent_metropolis_hastings_posterior2 ←function(n_sim, theta_est_linreg, var_a, var_b, var_c, beta_sigma2){
  a ← rnorm(n_sim, theta_est_linreg[1], var_a)
  b ← rnorm(n_sim, theta_est_linreg[2], var_b)
  c ← rnorm(n_sim, theta_est_linreg[3], var_c)
  shape_inv_gamma = 1 + (beta_sigma2/theta_est_linreg[4])
  sigma2 ← 1/rgamma(n_sim, shape = shape_inv_gamma, rate = beta_sigma2)
  y ←cbind(a,b,c,sigma2)
  #arbitrary initialization
  x ← t(as.matrix(theta_est_linreg))
  acceptance ← 0
  for (k in 1:n_sim){
    u ← runif(1,0,1)
    rho ← rho_accept_log2(x[k,],y[k,])
    if (log(u) < rho){
      x ← rbind(x,y[k,])
      acceptance ← acceptance + 1
    } else{
      x←rbind(x, x[k,])
    }
  }
  acceptance ← acceptance / n_sim
  return(list(x, acceptance))
}


#We try to adjust the parameters...
n_sim= 10000
results ← independent_metropolis_hastings_posterior2(n_sim = n_sim, theta_est_linreg = theta_est_linreg, var_a= 2, var_b= 1, var_c = 0.25, beta_sigma2 = 250)
theta_markov ← matrix(unlist(results[1]), ncol = 4, byrow = FALSE)
acceptance_rate ← unlist(results[2])
print(acceptance_rate)

 ######################

#Independent Metropolis Hastings with candidate = prior
#t and half-t candidates
#Note that here we don't not use the linear regression anymore to choose the prior parameters since the mean of a t-distribution is 0
#We only use the OLS estimators as an initialization
independent_metropolis_hastings_posterior3 ←function(n_sim, theta_est_linreg, nu_a, nu_b, nu_c, nu_sigma2){
  a ← rt(n_sim, nu_a)
  b ← rt(n_sim, nu_b)
  c ← rt(n_sim, nu_c)
  sigma2 ← sapply(rt(n_sim, nu_sigma2), abs)
  y ←cbind(a,b,c,sigma2)
  #arbitrary initialization
  x ← t(as.matrix(theta_est_linreg))
  acceptance ← 0
  for (k in 1:n_sim){
    u ← runif(1,0,1)
    rho ← rho_accept_log2(x[k,],y[k,])
    if (log(u) < rho){
      x ← rbind(x,y[k,])
      acceptance ← acceptance + 1
    } else{
      x←rbind(x, x[k,])
    }
  }
  acceptance ← acceptance / n_sim
  return(list(x, acceptance))
}


#We try to adjust the parameters...
n_sim= 15000
results ← independent_metropolis_hastings_posterior3(n_sim = n_sim, theta_est_linreg = theta_est_linreg, nu_a = 3.5, nu_b= 10, nu_c= 50, nu_sigma2 = 0.95)
theta_markov ← matrix(unlist(results[1]), ncol = 4, byrow = FALSE)
acceptance_rate ← unlist(results[2])
print(acceptance_rate)
mean(theta_markov[,4])
mean(theta_markov[,3])
mean(theta_markov[,2])
mean(theta_markov[,1])
```

I tried both types of prior on this second model : normal and inverted gamma or t and half-t distributions. However, I was not able to achieve a reasonable acceptance rate : I did not even reach 0.5 % ! I find 2 explanations to this :

- **Normal-Inverted Gamma prior** : In this case, the problem probably still comes from the Inverted Gamma prior on the σ^2 parameter. It is however reinforced here by the fact that the posterior of the coefficients parameters a, b and c may have a larger variance as the error tails of the model are heavier, which introduces more "uncertainty".
- **t and half-t prior** : In this case, the problem seems to rather come from the prior on the coefficients a, b and c. The Student t-distribution has indeed a mean equal to 0, and thus leads to simulations that have a too small likelihood and are thus rejected. In we adjust the number of degrees of freedom ν we can obtain heavier tails but most of the simulations will remain close to 0.

Hence, a good solution to this would be to use a regular Metropolis-Hastings with initialization at the linear regression estimates rather than an Independent Metropolis-Hastings. We would then not use the prior as the candidate distribution but rather use a random walk. We would therefore be able to sample candidates which are close to the linear regression estimates and which remain in a high likelihood region.

7.37 and 7.38 : The Metropolis-Hastings algorithm is a special case of a more general class of algorithms

Hastings has defined a more general class of algorithms based on any arbitrary positive symmetric function s such that $\rho(x,y) \leq 1$:

Given $x^{(t)}$

1. Generate $Y_t \sim q(y|x^{(t)})$
2. Take :
$$X^{(t+1)} = \begin{cases} Y_t \text{ with probability } \rho(x^{(t)}), Y_t) \\ x^{(t)} \text{ otherwise} \end{cases}$$

Where the acceptance probability $\rho(x,y) = \frac{s(x,y)}{1+\frac{f(x)q(y|x)}{f(y)q(x|y)}}$ (7.20).

7.37 - Show that the transition kernel associated with the acceptance probability (7.20), i.e the acceptance probability of this more general class of algorithms, also leads to f as invariant distribution for every symmetric function s

Let an arbitrary positive symmetric function s such that $\rho(x,y) \leq 1$.

We will show that the generated Markov chain $(X^{(t)})$ satisfies the detailed balance condition with f. Using Theorem 6.46, this will imply that f is the invariant distribution.

What is the corresponding transition kernel ? The transition kernel $K(\cdot,\cdot)$ of the generated Markov chain $(X^{(t)})$ is :

$$K(x,y) = \rho(x,y)q(y|x) + (1 - r(x))\delta_x(y)$$

Where $r(x) = \int \rho(x,y)q(y|x)dy$.

Step 1 - Show that $\rho(x,y)q(y|x)f(x) = \rho(y,x)q(x|y)f(y)$

$$\begin{aligned}
\rho(y,x)q(x|y)f(y) &= \frac{s(y,x)}{1+\frac{f(y)q(x|y)}{f(x)q(y|x)}} q(x|y)f(y) \\
&= \frac{s(y,x)f(x)q(y|x)}{f(x)q(y|x)+f(y)q(x|y)} q(x|y)f(y) \\
&= \frac{s(y,x)f(y)q(x|y)}{f(x)q(y|x)+f(y)q(x|y)} f(x)q(y|x) \\
&= \frac{s(y,x)}{\frac{f(x)q(y|x)}{f(y)q(x|y)}+1} q(y|x)f(x)
\end{aligned}$$

Since s is symmetric, $s(y,x) = s(x,y)$:

$$= \frac{s(x,y)}{\frac{f(x)q(y|x)}{f(y)q(x|y)} + 1} q(y|x)f(x)$$

$$\boxed{\rho(y,x)q(x|y)f(y) = \rho(x,y)q(y|x)f(x)}$$

Step 2 - Show that $(1 - r(x))\delta_x(y)f(x) = (1 - r(y))\delta_y(x)f(y)$ First note that :

$$\delta_x(y) = \begin{cases} 1 \text{ if } y = x \\ 0 \text{ if } y \neq x \end{cases} = \delta_y(x)$$

Therefore :

$$\begin{aligned} (1 - r(x))\delta_x(y)f(x) &= (1 - r(x))\delta_y(x)f(x) \\ &= \begin{cases} 0 \text{ if } y \neq x \\ (1 - r(x))f(x) \text{ if } y = x \end{cases} \\ &= \begin{cases} 0 \text{ if } y \neq x \\ (1 - r(y))f(y) \text{ if } y = x \end{cases} \end{aligned}$$

$$\boxed{(1 - r(x))\delta_x(y)f(x) = (1 - r(y))\delta_y(x)f(y)}$$

Step 3 - Conclusion It follows from Step 1 and Step 2 that :

$$K(x,y)f(x) = K(y,x)f(y)$$

The detailed balance condition is thus verified with f. Hence, (using Theorem 6.46), f is the invariant distribution of the generated Markov chain.

$\boxed{\text{Conclusion : for any s, this more general class of algorithms leads to f as invariant distribution.}}$

7.38 - Show that the Metropolis-Hastings algorithm is a special case of this more general class of algorithms

Let $\tilde{\rho}(x,y)$ the acceptance probability in Metropolis-Hastings.

Step 1 - Condition on s so as the Metropolis-Hastings algorithm to be a special case of this more general class of algorithm. The Metropolis-Hastings algorithm belongs to this more general class of algorithm, if and only if there exists s positive symmetric such that the acceptance probabilities are the same, i.e :

$$\tilde{\rho}(x,y) = \rho(x,y)$$

$$\Leftrightarrow$$

$$\min\left(1, \frac{f(y)q(x|y)}{f(x)q(y|x)}\right) = \frac{s(x,y)}{1 + \frac{f(x)q(y|x)}{f(y)q(x|y)}}$$

$$\Leftrightarrow$$

$$f(y)q(x|y)\min\left(\frac{1}{f(y)q(x|y)}, \frac{1}{f(x)q(y|x)}\right) = f(y)q(x|y)\frac{s(x,y)}{f(x)q(y|x) + f(y)q(x|y)}$$

$$\Leftrightarrow$$

$$\min\left(\frac{1}{f(y)q(x|y)}, \frac{1}{f(x)q(y|x)}\right) = \frac{s(x,y)}{f(x)q(y|x) + f(y)q(x|y)}$$

$$\Leftrightarrow$$

$$\min\left(\frac{f(x)q(y|x) + f(y)q(x|y)}{f(y)q(x|y)}, \frac{f(x)q(y|x) + f(y)q(x|y)}{f(x)q(y|x)}\right) = s(x,y)$$

$$\Leftrightarrow$$

$$\min\left(1 + \frac{f(x)q(y|x)}{f(y)q(x|y)}, 1 + \frac{+f(y)q(x|y)}{f(x)q(y|x)}\right) = s(x,y)$$

$$\Leftrightarrow$$

$$s(x,y) = 1 + \min\left(\frac{f(x)q(y|x)}{f(y)q(x|y)}, \frac{f(y)q(x|y)}{f(x)q(y|x)}\right)$$

Step 2 - Verify that if s has this form, s is positive symmetric

1 - s is positive If $s(x,y) = 1 + \min\left(\frac{f(x)q(y|x)}{f(y)q(x|y)}, \frac{f(y)q(x|y)}{f(x)q(y|x)}\right)$, s is obviously positive since f and q are positive functions (as they are densities).

2 - s is symmetric If $s(x,y) = 1 + \min\left(\frac{f(x)q(y|x)}{f(y)q(x|y)}, \frac{f(y)q(x|y)}{f(x)q(y|x)}\right)$:

$$s(y,x) = 1 + \min\left(\frac{f(y)q(x|y)}{f(x)q(y|x)}, \frac{f(x)q(y|x)}{f(y)q(x|y)}\right)$$

Since min is symmetric :

$$s(y,x) = 1 + \min\left(\frac{f(x)q(y|x)}{f(y)q(x|y)}, \frac{f(y)q(x|y)}{f(x)q(y|x)}\right)$$
$$s(y,x) = s(x,y)$$

Conclusion The Metropolis-Hastings algorithm is indeed a special case of this more general class of algorithm where : $\boxed{s(x,y) = 1 + \min\left(\frac{f(x)q(y|x)}{f(y)q(x|y)}, \frac{f(y)q(x|y)}{f(x)q(y|x)}\right)}$.

Appendix

Appendix to 7.1 (a) : Which Gamma parametrization should we consider : shape and scale or shape and rate ?

There are two different ways to specify the parameters for Gamma distributions. The first one, which is used in Problem 7.1, is to consider that a Gamma(α, β) corresponds to the probability density function :

$$\forall x > 0, \quad f_{\alpha,\beta}(x) = \frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} e^{-\frac{x}{\beta}}$$

The second one is to consider that a Gamma(α, β) corresponds to the probability density function :

$$\forall x > 0, \quad \tilde{f}_{\alpha,\beta}(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$$

However, in our case, we will use Accept-Reject to simulate from the target distribution Gamma(α, β) using the proposal distribution Gamma(a, b). Therefore, the following ratio must be bounded :

$$\tilde{w}(x) = \frac{\tilde{f}_{\alpha,\beta}(x)}{\tilde{f}_{a,b}(x)} = \frac{\Gamma(a)\beta^\alpha}{\Gamma(\alpha)b^a} x^{\alpha-a} e^{-(\beta-b)x}$$

In Problem 7.1, $a = 4 < \alpha = 4.3$ but $b = 7 > \beta = 6.2$. Hence $-(\beta - b) > 0$ and $\lim\limits_{x\to+\infty} \tilde{w}(x) = +\infty$. It is thus impossible to use Accept-Reject in this case, as the ratio cannot be bounded.

Therefore, the only possible density to consider in Problem 7.1 is necessarily the first one.

Appendix to 7.1 (a) : R Code

```
################################# Problem 7.1 #################################
###############################################################################
library(ggplot2)


#### Question (a) : Accept-Reject

#Gamma simulation with alpha an integer
transform_exp <- function(u, beta){
  epsilon <- - log(u)*beta
  return(epsilon)
}

gamma_simul_integer <- function(n_sim = 500, alpha,beta){
  gamma_simul = rep(0, n_sim)
  for (k in 1:alpha){
    u ← runif(n_sim, 0, 1)
    epsilon ← sapply(u,transform_exp, beta = beta)
    gamma_simul ← gamma_simul + epsilon
  }
  return(gamma_simul)
}


quotient_f_g ← function(y,alpha, beta, a, b){
  result = y**(alpha - a) * exp(alpha - a)
  result = result * exp(y * ((beta - b)/ (b*beta)))
  result = result * (((b - beta) /((alpha - a)*(b*beta)) )**(alpha - a))
  return(result)
}



n_sim = 10000


y ← gamma_simul_integer(n_sim = n_sim, alpha = 4, beta = 7)
#Check if it really looks like a Gamma density:
qplot(y , geom = 'density')

unif ← runif(n_sim, min = 0, max = 1)
accepted ← ifelse(unif ≤ sapply(y,quotient_f_g,alpha = 4.3, beta = 6.2, a = 4, b = 7),TRUE, FALSE)
print(length(accepted))
summary(accepted)

#Check if the empirical acceptance rate is close to the unconditional acceptance probability (equal to 1 / M)
compute_inv_M ← function(alpha,beta,a,b){
  result ← exp(alpha - a) * (((b - beta) /((alpha - a)*(b*beta)) )**(alpha - a))
  result ← result * gamma(alpha) * (beta**(alpha)) * (1 / (gamma(a)*(b**(a)) ))
  return(result)
}

compute_inv_M(alpha = 4.3, beta = 6.2, a = 4, b = 7)

qplot(y[accepted], geom = 'histogram', xlab='Accepted values', bins = 100)
print(length(y[accepted]))

#Compute the mean and monitor the convergence
est_vector_AR = c()
est_mean = 1 #arbitrary value initialization
for (k in 1:n_sim){
  est_mean ← ifelse(accepted[k] == TRUE,mean(y[1:k][accepted[1:k]]),est_mean)
  est_vector_AR ← c(est_vector_AR, est_mean)
}


q_AR ← qplot(seq(1,n_sim,1),est_vector_AR[1:n_sim], geom = 'line', ylab = 'Mean of Gamma(4.3,6.2)', xlab = 'Overall number of simulations')
q_AR
qplot(seq(1,length(accepted[accepted == TRUE]),1),est_vector_AR[accepted == TRUE], geom = 'line', ylab = 'Mean of Gamma(4.3,6.2)', xlab = 'Number of simulations')

#final value of the mean
mean(y[accepted])
```

Appendix to 7.1 (b) : R Code

```
#### Question (b) : Metropolis-Hastings with candidate = Gamma(4,7)
```

```

rho_accept ← function(x,y, alpha, beta, a ,b){
  quotient ← ((y**(alpha-a))/(x**(alpha-a))) * exp(((beta -b)/(b*beta))*(y - x))
  return(min(1,quotient))
}

independent_metropolis_hastings_gamma ←function(n_sim, alpha, beta, a, b){
  y ← gamma_simul_integer(n_sim, a, b)
  #arbitrary initialization
  x ← c(20)
  acceptance ← 0
  for (k in 1:n_sim){
    u ← runif(1,0,1)
    rho ← rho_accept(x[k],y[k], alpha, beta, a ,b)
    if (u < rho){
      x ← c(x,y[k])
      acceptance ← acceptance + 1
    } else{
      x←c(x, x[k])
    }
  }
  acceptance ← acceptance / n_sim
  return(list(x, acceptance))
}

n_sim= 10000
results ← independent_metropolis_hastings_gamma(n_sim, alpha = 4.3, beta=6.2, a=4,
     b=7)
x_markov ← unlist(results[1])
acceptance_rate ← unlist(results[2])
print(acceptance_rate)

#Plot the autocorrelations to monitor convergence towards the stationary
    distribution
bacf1 ← acf(x_markov, plot = FALSE)
bacfdf1 ← with(bacf1, data.frame(lag, acf))
q1 ← ggplot(data=bacfdf1, mapping=aes(x=lag, y=acf)) +
  geom_area(fill="grey") +
  geom_hline(yintercept=c(0.05, -0.05), linetype="dashed") +
  theme_bw() + ggtitle("ACF")

q1

#Plot the Markov chain
qplot(seq(1,n_sim + 1,1), x_markov, geom = ’line’, ylab = ’Value’ )

#Compute the mean and monitor the convergence
est_vector_MH1 = c()
for (k in 1:(n_sim+1)){
  est_mean ← mean(x_markov[1:k])
  est_vector_MH1 ← c(est_vector_MH1, est_mean)
}
q_MH1 ← qplot(seq(1,n_sim,1),est_vector_MH1[2:(n_sim + 1)], geom = ’line’, ylab =
    ’Mean of Gamma(4.3,6.2)’, xlab = ’Number of simulations’)
q_MH1

#Final value of the mean
est_vector_MH1[(n_sim + 1)]
```

Appendix to 7.1 (c) : R Code

```
#### Question (c) : Metropolis-Hastings with candidate = Gamma(5,6)
n_sim= 10000
results ← independent_metropolis_hastings_gamma(n_sim, alpha = 4.3, beta=6.2, a=5,
     b=6)
x_markov ← unlist(results[1])
acceptance_rate ← unlist(results[2])
```

```
print(acceptance_rate)

#Plot the autocorrelations to monitor convergence towards the stationary distribution
bacf2 ← acf(x_markov, plot = FALSE)
bacfdf2 ← with(bacf2, data.frame(lag, acf))
q2 ← ggplot(data=bacfdf2, mapping=aes(x=lag, y=acf)) +
  geom_area(fill="grey") +
  geom_hline(yintercept=c(0.05, -0.05), linetype="dashed") +
  theme_bw() + ggtitle("ACF")

q2

#Plot the Markov chain
qplot(seq(1,n_sim + 1,1), x_markov, geom = ’line’, ylab = ’Value’ )

#Compute the mean and monitor the convergence
est_vector_MH2 = c()
for (k in 1:(n_sim+1)){
  est_mean ← mean(x_markov[1:k])
  est_vector_MH2 ← c(est_vector_MH2, est_mean)
}
q_MH2 ← qplot(seq(1,n_sim,1),est_vector_MH2[2:(n_sim + 1)], geom = ’line’, ylab = ’Mean of Gamma(4.3,6.2)’, xlab = ’Number of simulations’)
q_MH2

#Final value of the mean
est_vector_MH2[(n_sim + 1)]

#Compare algorithms
mdf ← data.frame(AR = est_vector_AR, MH1 = est_vector_MH1[2:(n_sim + 1)], MH2 = est_vector_MH2[2:(n_sim + 1)] , Simulation = seq(1,n_sim,1))

library("reshape2")
mdf2 ← melt(mdf, id="Simulation")
ggplot(data=mdf2,
       aes(x=Simulation, y=value, colour=variable)) +
  geom_line()
```