MATH97125 – Computational Statistics

Group coursework – Autumn 2021
Comments and pointers towards solutions

In the following, we provide some guidelines, comments and answers to the questions. These are not meant to be full or model solutions. For some questions, there are many correct answers. In this report, a particular dataset (provided to one of the groups) is used. Results will be different for other groups. All the results will be rounded with 4 digits.

Let $\mathcal{D} = \{Y_i, X_{1i}, X_{2i}\}_{i=1}^{50}$ be the dataset of interest. Consider the following linear model

$$Y_i = \beta_1 X_{1i} + \beta_2 X_{2i} + \epsilon_i, \quad \epsilon_i \sim_{\text{i.i.d}} \mathcal{N}(0, \omega) \quad \text{for } i = 1, \dots 50 \,. \tag{1}$$

Assume the following prior distributions $\beta_1, \beta_2 \sim \mathcal{N}(0, 5)$, $\omega \sim$Gamma(1,1).

1. *(5 marks)* Let $\theta = (\beta_1, \beta_2, \omega)$. The posterior distribution of interest is

$$p(\theta|\mathcal{D}) \propto g(\omega; 1, 1)\Phi(\beta_1; 0, 5)\Phi(\beta_2; 0, 5) \prod_{i=1}^{50} \Phi(Y_i; \beta_1 X_{1i} + \beta_2 X_{2i}, \omega) := f(\beta_1, \beta_2, \omega)$$

   where $g(\omega; s, r)$ is the pdf of a gamma distribution with shape parameter s and rate parameter r, and $\Phi(y; m, \omega)$ is the pdf of a normal distribution with mean m and variance ω.

2. *(10 marks – 5 marks for the explanations, 5 marks for a correct implementation and diagnostic plots)* There are different correct answers for this question. Here, we have chosen to use the following proposal: given the current estimate of the parameter $\theta = (\beta_1, \beta_2, \omega)$, we propose $\theta' = (\beta_1', \beta_2', \omega')$ such as $\beta_1' \sim \mathcal{N}(\beta_1, \sigma_1^2)$, $\beta_2' \sim \mathcal{N}(\beta_2, \sigma_2^2)$ and $\omega' = \omega \exp(\gamma)$ with $\gamma \sim \mathcal{N}(0, \sigma_m^2)$. Therefore

$$\frac{q(\theta', \theta)}{q(\theta, \theta')} = \frac{q_m(\omega', \omega)}{q_m(\omega, \omega')}$$

   where q_m is defined as follows:

$$q_m(\omega, \omega') = \frac{1}{\omega' \sigma_a \sqrt{2\pi}} \exp(-\frac{(\log \omega - \log \omega')^2}{2\sigma_a^2}) \,. \tag{2}$$

Algorithm 1: Metropolis-Hastings algorithm

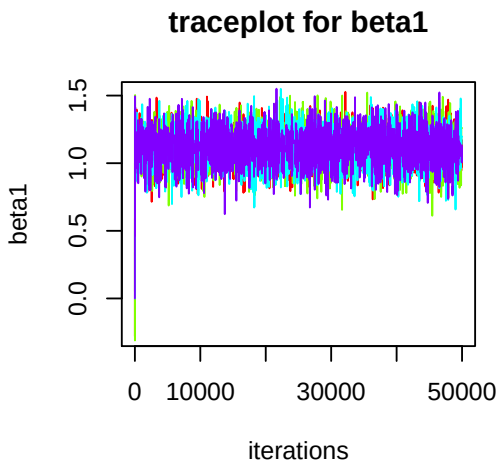
input: starting value $(\beta_1^0, \beta_2^0, \omega^0)$; variance of the proposals σ_1^2,σ_2^2,σ_m^2; number of iterations n
for $t = 1, 2, \dots n$ **do**
  Let $\beta_1' \sim \mathcal{N}(\beta_1^{t-1}, \sigma_1^2)$;
  Let $\beta_2' \sim \mathcal{N}(\beta_2^{t-1}, \sigma_2^2)$;
  Let $\omega' = \omega^{t-1} \exp(\gamma)$ with $\gamma \sim \mathcal{N}(0, \sigma_m^2)$;
  Let $U \sim U(0, 1)$;
  if $U \leq \min\left(\frac{f(\beta_1', \beta_2', \omega') q_m(\omega', \omega)}{f(\beta_1^{t-1}, \beta_2^{t-1}, \omega^{t-1}) q_m(\omega, \omega')}, 1\right)$ **then**
    $\beta_1^t = \beta_1', \quad \beta_2^t = \beta_2',$ and $\omega^t = \omega'$;
  else
    $\beta_1^t = \beta_1^{t-1}, \quad \beta_2^t = \beta_2^{t-1},$ and $\omega^t = \omega^{t-1}$;
Return $(\beta_1^t, \beta_2^t, \omega^t)_{t=1}^n$

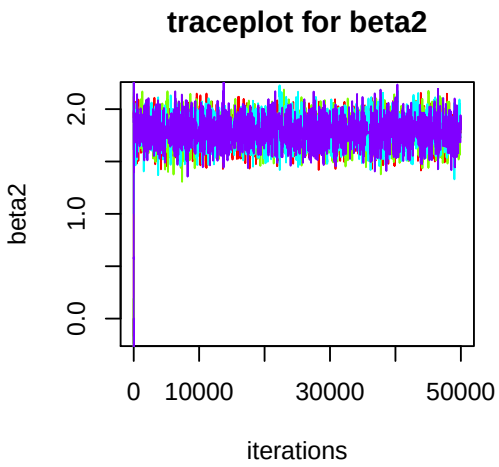
Other proposals could have been used. MH within Gibbs would also work well.

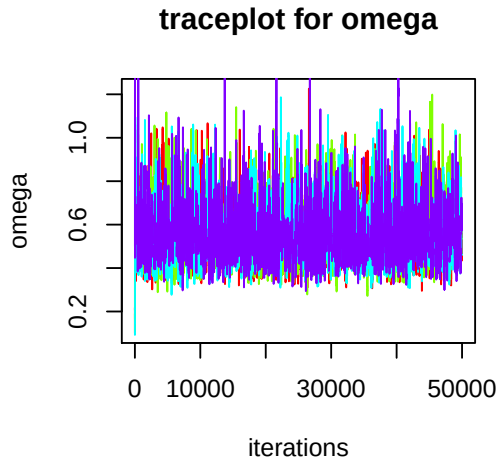
Below is the R implementation of this code:

```
> logpost <- function(theta) sum(dnorm(Y,X[,1]*theta[1]+X[,2]*theta[2],
+   sd=sqrt(theta[3]), log=T))+dgamma(theta[3],1,1, log=T)+
+   dnorm(theta[1],0, sqrt(5), log=T)+dnorm(theta[2],0, sqrt(5), log=T)
> sigma_b1 = 0.5 #0.2
> sigma_b2 = 0.5 #0.2
> sigma_o = 0.5 #0.3
> qe <- function(x,y) dnorm(log(y),log(x),sigma_o)/y
> rqe <- function(x) x*exp(rnorm(1,0,sigma_o))
> # Function tailored for this example
> MHpost <- function(n,init){
+   xall <- matrix(NA,ncol=n, nrow=3)
+   x <- init
+   xall[,1] <- x
+   for (t in seq(2,n)){
+     y <- c(rnorm(1,x[1],sigma_b1), rnorm(1,x[2],sigma_b2), rqe(x[3]))
+     if (runif(1)<=(exp(logpost(y)-logpost(x))*qe(y[3],x[3])/qe(x[3],y[3]))){
+       x<-y
+     }
+     xall[,t] <- x
+   }
+   return(xall)
+ }
```

Running the MCMC sampler described in the previous question for 5×10^4 iterations with $\sigma_1 = 0.2$, $\sigma_2 = 0.2$ and $\sigma_m = 0.3$ four times with different initial conditions, we obtain the following traceplots:



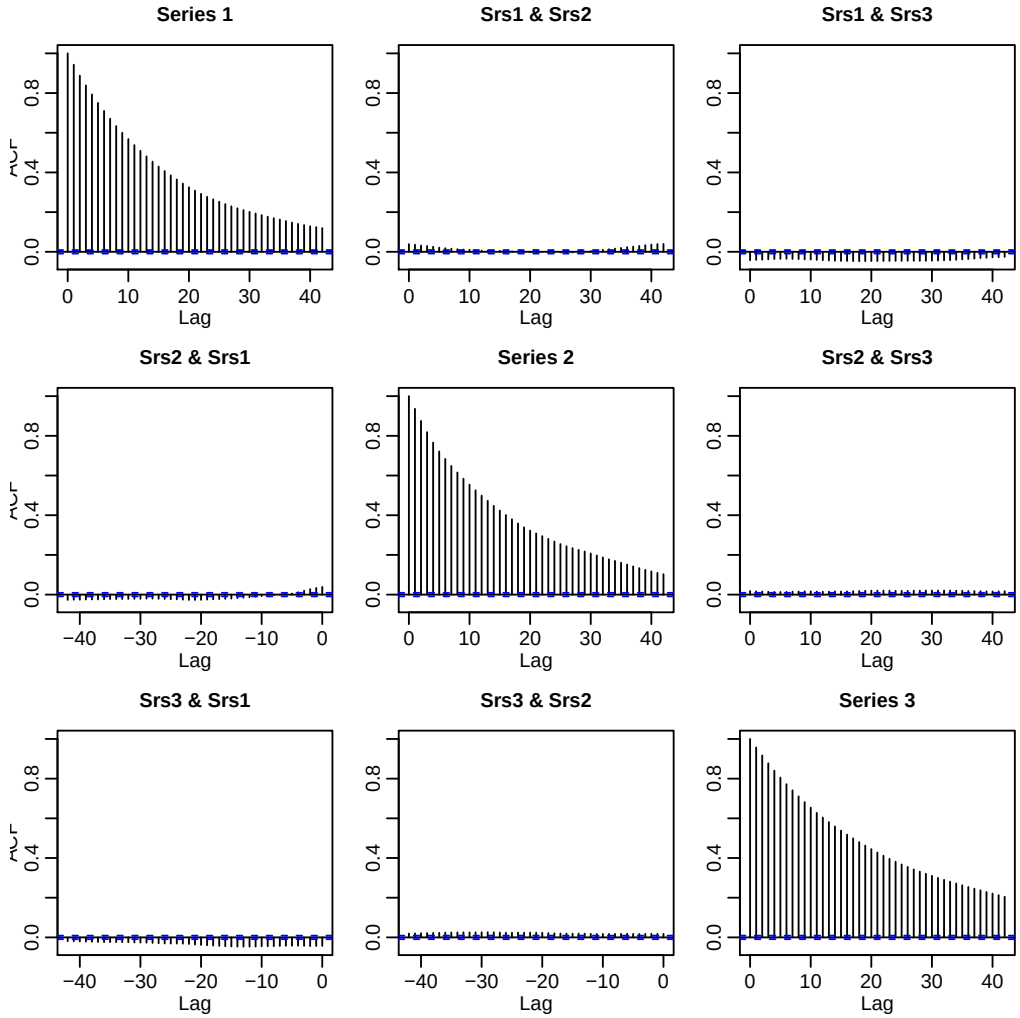




We also computed the Gelman Rubin diagnostic and obtained a point etimate and an upper confident interval equal to 1 for each of the three parameters. The traceplot and the result of the Gelman Rubin diagnostic both suggest that the MCMC algorithm has reached convergence.

In addition we observe from the auto-correlation plot below that the chain mixes well.

```
> acf(t(chains1))
```



We tried various combinations of values for the variances of the three proposal random walks. The combination used above was the one resulting to a Markov Chain with the highest effective sample size and showing a relatively good mixing.

3. (5 marks) The posterior mean of β_1 is given by $E(\beta_1|\mathcal{D}) = \int \beta_1 p(\beta_1|\mathcal{D}) d\beta 1$. The posterior mean can be estimated from an output $(\beta_1^t, \beta_2^t, \omega^t)_t$ of the MCMC algorithm as follows

$$E(\beta_1|\mathcal{D}) = \int \beta_1 p(\beta_1|\mathcal{D}) d\beta 1 \approx \frac{1}{n - n_0} \sum_{t=n_0+1}^{n} \beta_1^t$$

where n_0 is the burn-in period. Therefore, $E(\beta_1|\mathcal{D})$ is estimated by computing the empirical mean of the $\beta_1^{n_0+1}, \ldots \beta_1^n$.

```
> sm <- sapply(seq(3), function(i) mean(chains1[i,-seq(1000)]))
```

By considering a burn-in period of 1000, we obtain $E(\beta_1|\mathcal{D})$ ≈1.1103, $E(\beta_2|\mathcal{D})$ ≈1.7839, and $E(\omega|\mathcal{D})$ ≈0.5542.