Coursework - Computational Statistics

CID: 02091191

December 10, 2021

Abstract

In this document we are presenting the work conducted during the third Computational Statistics Coursework of the year.

Given a data set $\mathcal{D} = \{y_t\}_{t=1}^{50}$, consider the following model

$$Y_t = \alpha t + \beta \cos(\omega t) + \epsilon_t, \ \epsilon_t \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2) \text{ for } t = 1, \dots, 50 \tag{1}$$

where α and β are fixed unknown parameters, $\omega = \frac{\pi}{10}$, and $\sigma^2 = 0.1$ known. Assume the following prior distributions for α and β: α and β follow a Student's t-distribution with 5 degrees of freedom independently.

(1) Let $p(\alpha, \beta|\mathcal{D})$ be the joint posterior distribution of α and β given the data set $\mathcal{D}$. Bayes' theorem states that

$$\text{posterior} \propto \text{prior} \times \text{likelihood}$$

Thus,

$$p(\alpha, \beta|\mathcal{D}) \propto T(\alpha; 5)T(\beta; 5) \prod_{t=1}^{50} \Phi(Y_t; \alpha t + \beta \cos(\omega t), \sigma^2) := f(\alpha, \beta)$$

where $T(x; d)$ is the pdf of a Student's t-distribution with d degrees of freedom evaluated at x and $\Phi(x; \mu, \sigma^2)$ is the pdf of a normal distribution with mean μ and variance σ^2 evaluated at x.

(2) Assume we want to sample from the joint posterior distribution of α and β given the data $\mathcal{D}$ stated above. Given a current estimate of the parameters $\theta = (\alpha, \beta)$, we propose $\theta' = (\alpha', \beta')$ such that $\alpha' \sim \mathcal{N}(\alpha, \sigma_\alpha^2)$ and $\beta' \sim \mathcal{N}(\beta, \sigma_\beta^2)$, which is a simple random walk proposal with chosen variances σ_α^2 and σ_β^2. Therefore,

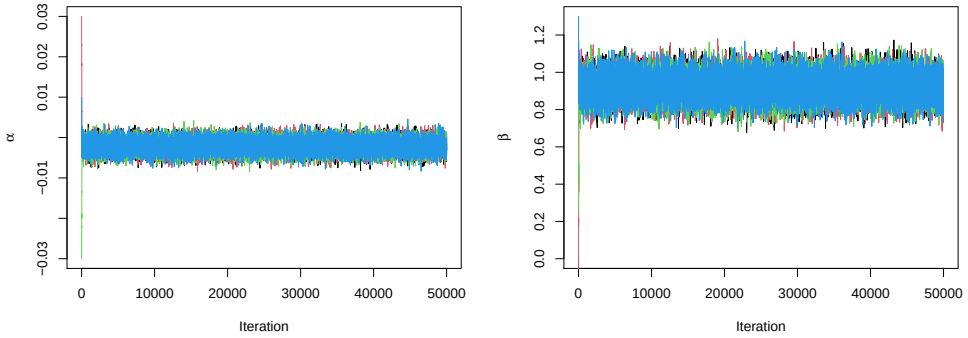
$$\frac{q(\theta', \theta)}{q(\theta, \theta')} = 1$$

as the proposal q is symmetric around the current position (q is here a bivariate normal distribution with parameters α, β, σ_α^2 and σ_β^2).

Algorithm 1: Metropolis-Hasting Algorithm

Data: starting values (α^0, β^0) ; variances $(\sigma_\alpha^2, \sigma_\beta^2)$; number of iterations n
Result: Chains $\{\alpha^t\}_{t=1}^n$ and $\{\beta^t\}_{t=1}^n$
for t = 1,...,n **do**
  Let $\alpha' \sim \mathcal{N}(\alpha, \sigma_\alpha^2)$ and $\beta' \sim \mathcal{N}(\beta, \sigma_\beta^2)$
  Let $U \sim U([0, 1])$
  if $U \leq \min\left(1, \frac{f(\alpha', \beta')}{f(\alpha, \beta)}\right)$ **then**
    $\alpha^t = \alpha', \ \beta^t = \beta'$
  else
    $\alpha^t = \alpha^{t-1}, \ \beta^t = \beta^{t-1}$
  end
end

This proposal bivariate distribution is appropriate since there is no constraint on α and β, in the sense that these unknown parameters don't have to remain positive or negative for example like it would have been the case for a variance parameter. The pseudo-code in Algorithm **??** summarises the procedure of the MCMC sampler used here. In **R**, we may tune the proposal variances $\sigma_\alpha^2, \sigma_\beta^2$ in order to get a *good* mixing of the Markov Chains. Running this sampler with $\sigma_\alpha = 0.002$, $\sigma_\beta = 0.07$ (then called the optimal proposal variances) on $n = 5 \times 10^4$ iterations gives in average an acceptance rate of 0.4842 which is a good trade-off according to the Lecture Notes [1]: indeed, for this two-dimensional sampler, such an acceptance rate would result in a *good* mixing of the chains.



(a) Trace plot for α, with $\alpha^0 \in \{0, 0.03, -0.03, 0.01\}$

(b) Trace plot for β, with $\beta^0 \in \{0, 0.25, 1, 1.3\}$

Figure 1: Trace plots for each unknown parameters generated from MCMC algorithm, for different starting values

Figure 1 shows that for different starting values, the Markov Chains for α and β converged. The mixing also looks good and it is confirmed by an acceptance rate of approximately 0.5. One can also compute the effective sample size, defined as the number of independent samples which would have been required to get the same quality of results. We obtain an effective sample size of 6737 for α and 5321 for β, which looks high enough for a Markov Chain of length 5×10^4. We tried before with different values of proposal variances in order to optimise the sampler and get a high effective sample size.

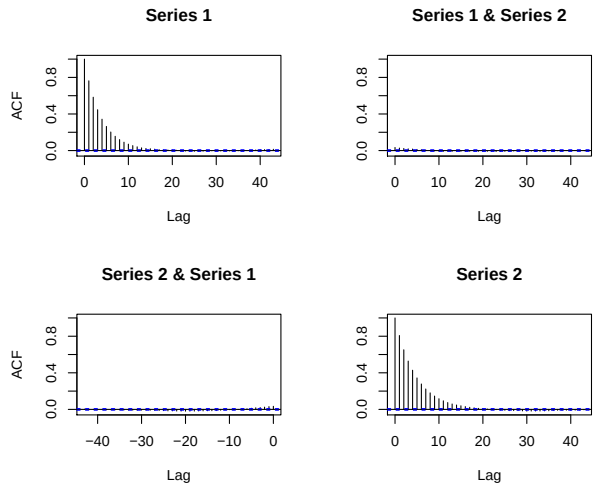
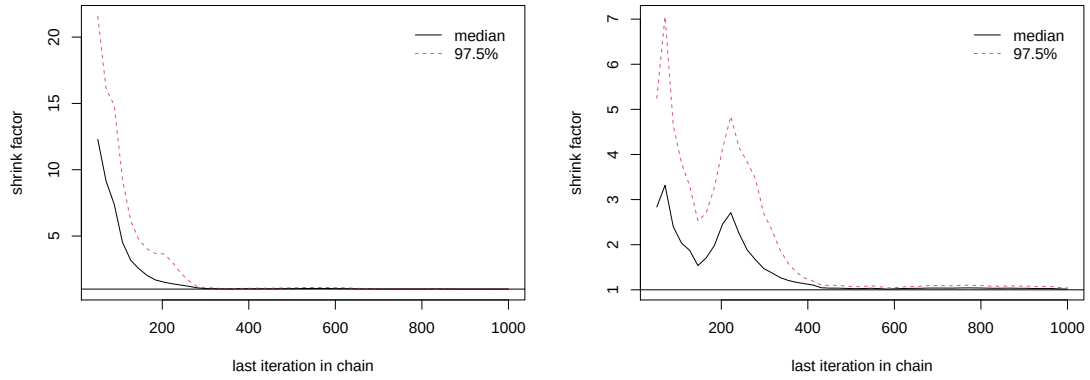


Figure 2: ACF plots of the generated Markov Chains for α and β (Series 1: α, Series 2: β), with the optimised proposal variances

Figure 2 shows that the auto-correlation decreases quickly as the lag increases, which assess a good

mixing of the chain.



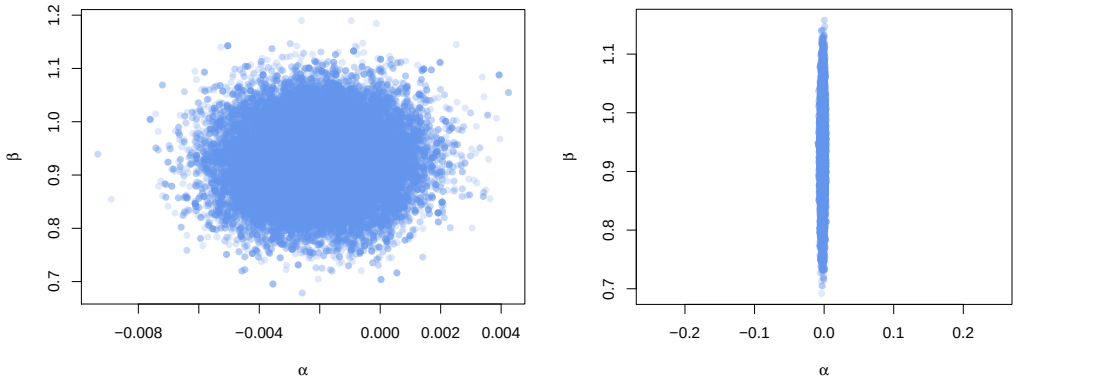
(a) Gelman plot for α

(b) Gelman plot for β

Figure 3: Gelman plots for each unknown parameter, with 5 different random starting values for each chain, using the optimised proposal variances

Figure 3 shows the Gelman plots for both chains seem to have reached convergence, as the ratio R verifies $\sqrt{\hat{R}} < 1.1$ after a certain number of iterations. Moreover, the Gelman diagnostics give for both α and β a point estimate of 1 and an upper confidence interval of 1.01. Hence, we can assess that the Markov Chains have reached convergence and a relatively *good* mixing of the chains.

(3) We may represent the joint distribution $p(\alpha, \beta|\mathcal{D})$ obtained from the sampler in Figure 4a.



(a) Empirical posterior joint distribution $p(\alpha, \beta|\mathcal{D})$, with different scale for each axis

(b) Empirical posterior joint distribution $p(\alpha, \beta|\mathcal{D})$, with the same scale for each axis

Figure 4: Empirical posterior joint distribution $p(\alpha, \beta|\mathcal{D})$

Figure 4a shows that values for α are very narrowed around 0. If we used the same scales for the α-axis and the β-axis, we would have seen (roughly) a straight vertical line, as shown on Figure 4b

(4) Assume we are now interested in computing the posterior mean for each parameter. These posterior means can be estimated using MCMC algorithm as follows

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

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

where n_0 is a burn-in period. In **R**, using a Markov Chain of length 10^5 and a burn-in period of 10^3, we obtain $\mathbb{E}[\alpha|\mathcal{D}] \approx -0.0021$ and $\mathbb{E}[\beta|\mathcal{D}] = 0.9261$. We may also notice that the posterior mean obtained for α is very close to 0. In fact, fitting a linear model for $y = \alpha t + \beta \cos(\omega t)$ gives the estimates $\alpha = -0.0021$ and $\beta = 0.9290$, which are very close to the estimates obtained using MCMC algorithm. Furthermore, the t-tests associated to the α parameter reports a p-value for 0.163, which would make us fail to reject the null hypothesis that $\alpha = 0$. Hence, we may consider α to be not statistically significant in this model. This will be discussed in deep in question (6).

(5) Let $p(y_t^*|\mathcal{D}, t)$ be the posterior predictive distribution of y_t^* for a given t, defined as follows

$$p(y_t^*|\mathcal{D}, t) = \iint p(y_t^*|\alpha, \beta, t) p(\alpha, \beta|\mathcal{D}) d\alpha d\beta$$

(a) Let $t = 55$ fixed. Then, we can estimate $p(y_t^*|\mathcal{D}, t = 55)$ which is a function of y_t^* using Monte Carlo integration and the Markov Chains generated previously for α and β. The idea is to generate a grid of values for y_t^* and compute the probability to observe this value of y_t^* when $t = 55$ given the data $\mathcal{D}$, defined by

$$\begin{aligned} p(y_t^*|\mathcal{D}, t) &\approx \frac{1}{n - n_0} \sum_{i=n_0+1}^{n} p(y_t^*; \alpha^i, \beta^i, t) \\ &= \frac{1}{n - n_0} \sum_{i=n_0+1}^{n} \Phi(y_t^*; \alpha^i t + \beta^i \cos(\omega t), \sigma^2) \end{aligned}$$

where n_0 is a burn-in period. In **R**, using $n_0 = 10^3$, $n = 10^5$, $t = 55$ and the chains for α and β obtained with the optimal proposal variances, we obtain the following plot:

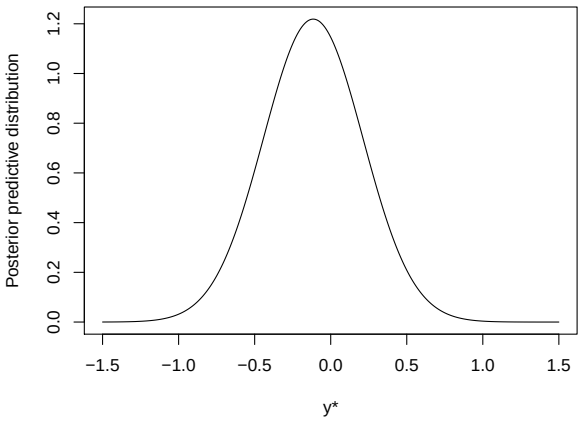


Figure 5: Posterior predictive distribution of $y_{t=55}^*$

Figure 5 shows $p(y_t^*|\mathcal{D}, t)$ as a function of y_t^*. The obtained curve is maximised in $y_{t=55}^* = -0.115$. Hence, this would mean that according to the chains of α and β obtained using MCMC algorithm, the predicted value of Y_t where $t = 55$ would be -0.115. To compare, we can re-use the linear model fitted in (4) in order to compare the predicted value it would gave given: we obtain -0.1129, which is close to the value obtained using MCMC.

(b) Let Y_t^* be a random variable with pdf $p(y_t^*|\mathcal{D}, t)$. Then, as described in the linear model,

$$Y_t \sim \mathcal{N}(\alpha t + \beta \cos(\omega t), \sigma^2)$$

Thus,

$$\mathbb{E}[Y_t] = \alpha t + \beta \cos(\omega t)$$

Hence, similarly we could use the MCMC algorithm sampling the two chains $\{\alpha^i\}_{i=1}^n$ and $\{\beta^i\}_{i=1}^n$, and then estimate

$$\begin{aligned}\mathbb{E}[Y_t^*] &\approx \frac{1}{n-n_0}\sum_{i=n_0+1}^{n} \alpha^i t + \beta^i \cos(\omega t) \\ &= \underbrace{\left(\frac{1}{n-n_0}\sum_{i=n_0+1}^{n}\alpha^i\right)}_{\approx \bar{\alpha}} t + \underbrace{\left(\frac{1}{n-n_0}\sum_{i=n_0+1}^{n}\beta^i\right)}_{\approx \bar{\beta}} \cos(\omega t)\end{aligned}$$

which is a function of t, where n_0 is a burn-in period. Therefore, we can plot $\mathbb{E}[Y_t^*]$ as a function of t for $t \in [0, 60]$.

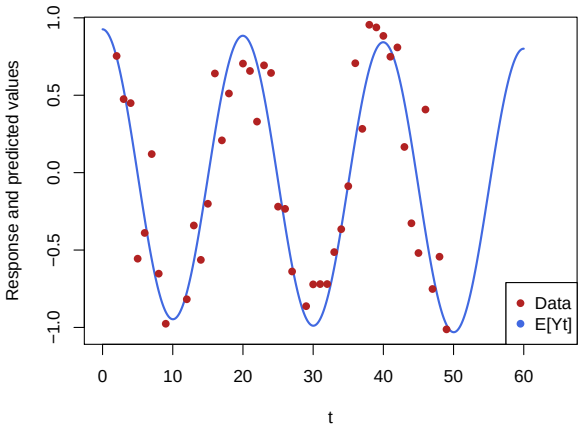


Figure 6: Comparison plot between the predicted values $\mathbb{E}[Y_t^*]$ and the data

The comparison plot obtained in Figure 6 shows that our estimated coefficients of α and β using MCMC fit relatively well to the actual data on $t \in [0, 50]$. These estimates can also be used to make predictions, as it is shown for example on the same figure for $t \in [50, 60]$. To discuss more about the goodness of fit, we should focus more in deep on the residuals or the R^2 coefficient, which is not required here.

(6) Assume now we want to compare the model defined in Equation 1 noted as $\mathcal{M}_1$ to another model $\mathcal{M}_2$ defined as

$$Y_t = \beta \cos(\omega t) + \epsilon_t, \ \epsilon_t \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2) \text{ for } t = 1, \ldots, 50 \tag{2}$$

where $\omega = \frac{\pi}{10}$ and $\sigma^2 = 0.1$, assuming the same prior for β as in $\mathcal{M}_1$.

(a) We may notice that $\mathcal{M}_2$ is a sub-model of $\mathcal{M}_1$ where $\alpha = 0$. Using the MCMC sampler designed in (2), we can compute a simple 95% confidence interval for α by looking at the 0.025 and 0.975 empirical quantiles of the chain obtained for α, using a burn-in period. Hence, we obtain $\alpha \in [-0.0051, 0.0009]$ at a 95% confidence level. We may notice that this confidence interval seems very tightened around 0: then, we can wonder if α was significant in $\mathcal{M}_1$ or not. As discussed in (4), the t-test provided by fitting a linear model to the data would reveal that α is not statistically significant in the first model $\mathcal{M}_1$, i.e. we fail to reject $\alpha = 0$. The idea of the following question (6) (b) is to design and implement a reversible jump Markov Chain Monte Carlo algorithm between

these two models $\mathcal{M}_1$ and $\mathcal{M}_2$ in order to proceed to model selection.

(b) Assume we now want to design a Reversible Jump MCMC algorithm to estimate $\mathbb{P}[\mathcal{M}_1|\mathcal{D}] = 1 - \mathbb{P}[\mathcal{M}_2|\mathcal{D}]$ and proceed to model selection. Recall that $\mathcal{M}_2$ is a sub-model of $\mathcal{M}_1$ where $\alpha = 0$. First derive the likelihood of each model.

$$L_1(y|\alpha,\beta) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{t=1}^{50}(y_t - \alpha t - \beta\cos(\omega t))^2\right) \text{ for } \mathcal{M}_1$$

$$L_2(y|\beta) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{t=1}^{50}(y_t - \beta\cos(\omega t))^2\right) \text{ for } \mathcal{M}_2$$

Then, assume the following priors for each model

$$\mathbb{P}[\mathcal{M}_1] = \mathbb{P}[\mathcal{M}_2] = \frac{1}{2}$$

And that both parameters α and β have a Student's t-distribution prior with 5 degrees of freedom, denoted t_5.

Thus, using Bayes' theorem (posterior $\propto$ prior $\times$ likelihood), we can write the posterior distribution $f(k,\theta_k)$ where $k = 1, 2$ denotes the model $\mathcal{M}_k$ and θ_k denotes the parameter estimate for model $\mathcal{M}_k$, i.e. $\theta_k = (\alpha,\beta)$ for $\mathcal{M}_1$ and $\theta_k = (\beta)$ for $\mathcal{M}_2$.

$$f(k,\theta_k) = \begin{cases} \frac{1}{2}\mathbb{P}((\alpha,\beta)|k=1)L(y|(\alpha,\beta),k=1), & k=1 \\ \frac{1}{2}\mathbb{P}((\beta)|k=2)L(y|(\beta),k=2), & k=2 \end{cases}$$

Then, f is the target distribution of this Reversible Jump MCMC algorithm. As we proceed to model selection and we do not want to favour a model over another, we use equal probability of proposing a move from $\mathcal{M}_1$ to $\mathcal{M}_2$ and a move from $\mathcal{M}_2$ to $\mathcal{M}_1$. Hence,

$$\kappa(m,\cdot) = \begin{cases} 1 & \text{with probability } \frac{1}{2} \\ 2 & \text{with probability } \frac{1}{2} \end{cases}$$

We then need to derive the acceptance probability of between-models moves. First, consider a move from $\mathcal{M}_2$ to $\mathcal{M}_1$. Then, let the current state be $x = (2,\theta)$ and the proposal $x' = (1,\theta')$. We generate $u \sim \mathcal{N}(0,\sigma_{12}^2)$ with $\sigma_{12}^2 > 0$. Then, $\theta' = (\alpha',\beta') = T_{12}(u,\beta) = (u,\beta)$ (T is chosen to be the identity). Hence, the Jacobian of T_{12} is

$$\left|\frac{\partial T_{12}(u,\theta)}{\partial(u,\theta)}\right| = 1$$

Hence, the acceptance probability for a proposed move from $\mathcal{M}_2$ to $\mathcal{M}_1$ is

$$\alpha_{21} = \min\left\{\frac{f(1,(u,\beta))}{f(2,(\beta))} \cdot \frac{1}{\Phi(u;0,\sigma_{12}^2)},\ 1\right\}$$

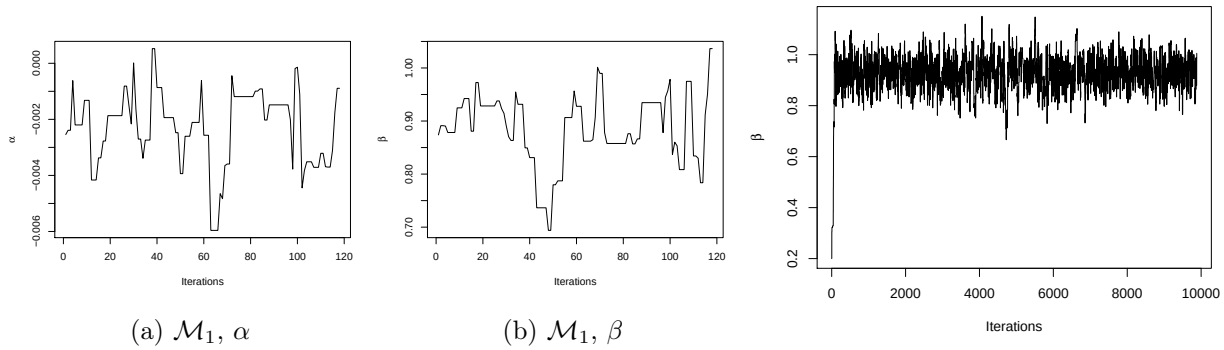
Now consider a move from $\mathcal{M}_2$ to $\mathcal{M}_1$. As we reduce the dimension, we may only delete the α parameter, which would give

$$T_{21}(\alpha',\beta') = (u,\beta) = (\alpha',\beta')$$

where $u \sim \mathcal{N}(0,\sigma_{12}^2)$. In that case, the acceptance probability is

$$\alpha_{12} = \min\left\{\frac{f(2,(\beta))}{f(1,(\alpha',\beta'))} \cdot \frac{\Phi(\alpha';0,\sigma_{12}^2)}{1},\ 1\right\}$$

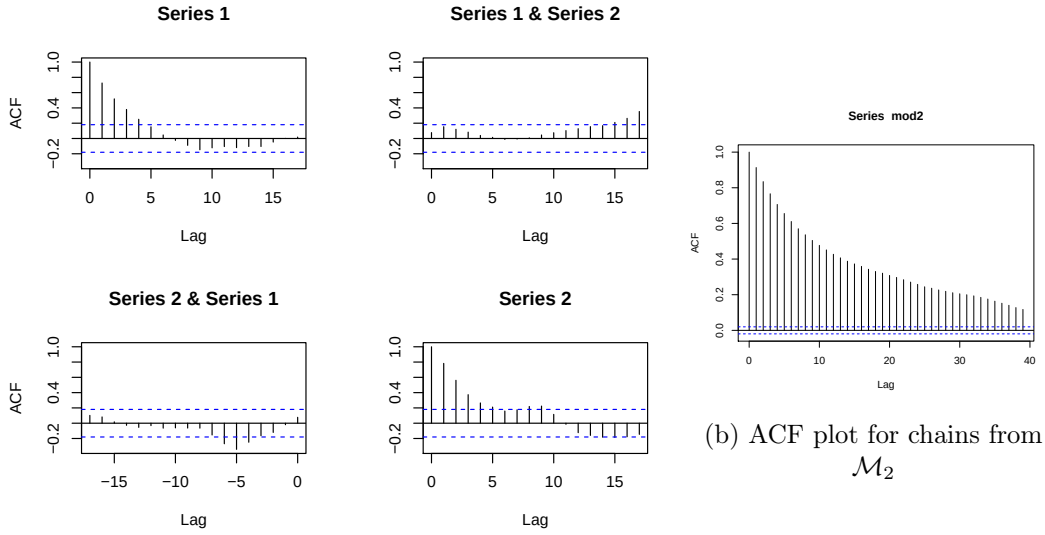
Implementing this Reversible Jump MCMC algorithm in `R` using $\sigma_\alpha = 0.001$, $\sigma_\beta = 0.07$, $\sigma_{12} = 2$, an initial value in $\mathcal{M}_1$ ($\alpha_0 = 0.2$) over $n = 10^4$ iterations gives a global acceptance rate (for all moves) of 0.3374 which is relatively good for a two-dimensional MCMC algorithm.



(a) $\mathcal{M}_1$, α (b) $\mathcal{M}_1$, β (c) $\mathcal{M}_2$, β

Figure 7: Trace plots of α and β chains generated by the Reversible Jump MCMC sampler, for each model $\mathcal{M}_1$ and $\mathcal{M}_2$

Figure 7 shows the evolution of the Markov Chains for each parameter in each model. We may notice that $\mathcal{M}_1$ contains much less observations than $\mathcal{M}_2$: $\mathcal{M}_1$ contains 118 observations whereas $\mathcal{M}_2$ contains 9882 observations, which explains the look of plots 7a and 7b.



(a) ACF plot for chains from $\mathcal{M}_1$

(b) ACF plot for chains from $\mathcal{M}_2$

Figure 8: ACF plots for each model

Figure 8 shows the auto-correlation plots of each Markov Chain generated using the sampler. Note that the auto-correlation in samples from $\mathcal{M}_1$ is decreasing as well as the one of samples from $\mathcal{M}_2$. Therefore, we may assess that the MCMC algorithm has reached convergence. The obtained effective sample sizes for each chain are 19 for α from $\mathcal{M}_1$, 18 for β from $\mathcal{M}_1$ and 276 for β from $\mathcal{M}_2$. We would prefer higher values here, but these are the best we could have tuning all the parameters of the model.

We may now estimate $\mathbb{P}[\mathcal{M}_1|\mathcal{D}] = 1 - \mathbb{P}[\mathcal{M}_2|\mathcal{D}]$ using the proportion of samples from each model in the output of the MCMC sampler. Hence, we obtain

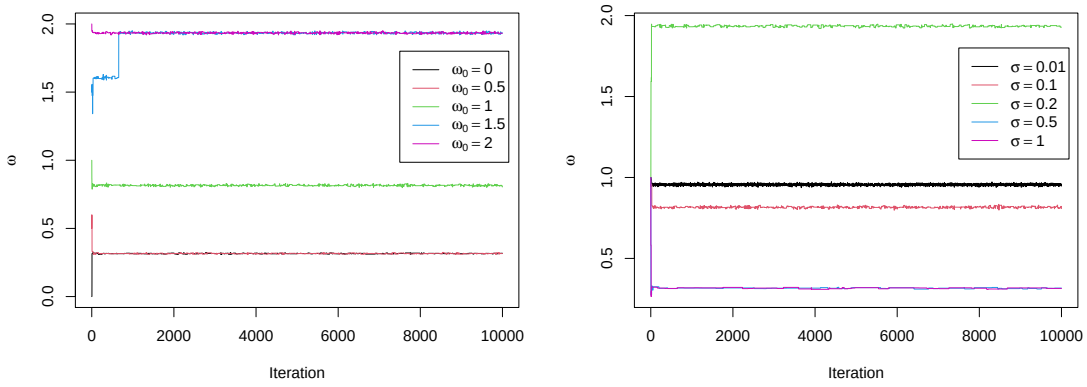
$$\mathbb{P}[\mathcal{M}_1|\mathcal{D}] = \frac{118}{10^4} = 0.0118 \text{ and } \mathbb{P}[\mathcal{M}_2|\mathcal{D}] = 1 - \frac{118}{10^4} = 0.9882$$

Therefore, the model $\mathcal{M}_2$ seems much more appropriate to the data than the model $\mathcal{M}_1$. This result is consistent to what we had noticed in 6 (a) computing a 95% confidence interval for α

which was very narrowed around 0. Moreover, as discussed in (4), fitting the linear model $\mathcal{M}_1$ to the data, and proceeding to inference on α parameter via a t-test shows that α is not statistically significant to $\mathcal{M}_1$, i.e. $\alpha = 0$. Finally, looking at Figure 6 reveals no global linear trend in the data or in the fitted model. Therefore, we would prefer using $\mathcal{M}_2$.

(7) Suppose now we are interested in inferring the parameter ω in the model $\mathcal{M}_1$, assuming a uniform prior distribution between 0 and π for this parameter. We now consider the parameters α and β as fixed, and equal to the values obtained in (4): $\alpha = -0.0021$ and $\beta = 0.9261$. We then implement a simple random-walk Metropolis-Hastings algorithm to sample from $p(\omega|\alpha, \beta, \mathcal{D})$ the same way we designed it in (1). We may re-write the posterior distribution as

$$p(\omega|\alpha, \beta, \mathcal{D}) \propto U(\omega; 0, \pi) \prod_{t=1}^{50} \Phi(Y_t; \alpha t + \beta \cos(\omega t), \sigma^2) := f(\omega)$$



(a) Different initial values $\omega_0 \in \{0, 0.5, 1, 1.5, 2\}$, same proposal variance $\sigma_\omega = 0.1$

(b) Same initial values $\omega_0 = 1$, different proposal variances $\sigma_\omega \in \{0.01, 0.1, 0.2, 0.5, 1\}$

Figure 9: Comparison of MCMC output sampling from $p(\omega|\alpha, \beta, \mathcal{D})$ with either different initial values or different proposal variances

Figure 9a shows the trace plot of ω for different initial values with same proposal variance. We may notice that the chains do not converge to the same value: hence, the MCMC algorithm seems sensitive to initial conditions. Note that since ω is a pulse (frequency) in $\mathcal{M}_1$, ω could have been defined modulo-2π. However, even considering that, we obtain different values of ω which are not the same modulo-2π. Figure 9b shows the trace plot of ω for different proposal variances but with the same starting value. We may notice here that, again, the chains do not converge to the same value even after a high number of iterations. This is due to the different proposal variances which allow or not *high* jumps in the Markov Chain.

To get a better idea of the domain of convergence, we may focus on the ω estimate for different starting values and different proposal variances. Hence, we construct a grid of possible starting values ω_0 from 0 to π and proposal variances σ_ω from 0 to 1. Each component contains 40 elements, so we construct a grid containing $40^2 = 1600$ different initial conditions. Then, for each value of σ_ω and ω_0 in the grid, we generate 3 Markov Chains for ω of length 3000 using the MCMC sampler implemented previously, using a burn-in period of 500, and compute the average of the (converged) ω estimate. Therefore, we obtain a grid a 1600 ω estimates for different ω_0 and σ_ω.

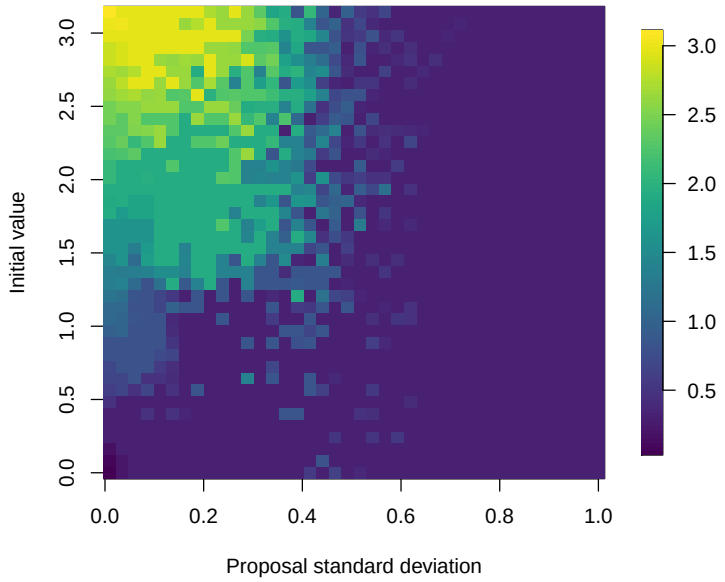


Figure 10: Grid of ω estimates from multiple MCMC samplers for different values of ω_0 and σ_ω

Figure 10 allows to see to which ω all these MCMC samplers converge to for each chosen initial conditions. Notice that for *low* proposal variances σ_ω (between 0.01 and 0.6) and for high starting value ω_0 (above 1 approximately), the generated Markov Chains seem to converge to different values. Indeed, as the variance is small, the Markov Chain remains around the initial value and seems stuck. Moreover, for *low* starting value for ω around $\frac{\pi}{10}$, the Markov Chain seems to converge almost always to the "real" value $\frac{\pi}{10}$ (in purple on this grid), and same for *high* proposal variances (greater than 0.6). Hence, when trying to infer on ω parameter, one needs to be careful before drawing any conclusion using a simple random-walk Metropolis-Hastings algorithm as this parameter has multiple modes.

References

[1] Dr Sarah Filippi (Autumn 2021) *MATH70093 - Computational Statistics lecture notes*, Imperial College London MSc Statistics resources

[2] Kenneth Lange (2010) *Numerical analysis for statisticians*, New York: Springer

A R code

```
data <- read.csv("02091191.csv")
library("stringi")
Y <- data$y
t <- data$t
omega <- pi/10
sigma <- sqrt(0.1)

### Question 2

logpost <- function(theta){
  dt(theta[1], 5, log=TRUE) + dt(theta[2], 5, log=TRUE) +
    sum(dnorm(Y, theta[1]*t + theta[2]*cos(omega*t), sigma, log=TRUE))
}

sigma.alpha <- 0.002
sigma.beta <- 0.07
MHpost <- function(n, init){
  accept <- 0
  xall <- matrix(NA,ncol=n, nrow=2)
  x <- init
  xall[,1] <- x
  for (t in seq(2,n)){
    y <- c(rnorm(1, x[1], sigma.alpha), rnorm(1, x[2], sigma.beta))
    if(runif(1)<=(exp(logpost(y)-logpost(x)))){
      x <- y
      accept <- accept + 1/n
    }
    xall[,t] <- x
  }
  print(accept)
  return(xall)
}

chain <- MHpost(5e4, c(0, 1))
chain1 <- MHpost(5e4, c(0, 1))
chain2 <- MHpost(5e4, c(0.03, 0))
chain3 <- MHpost(5e4, c(-0.03, 0.25))
chain4 <- MHpost(5e4, c(0.01, 1.3))

# Traceplots
plot(chain[1,], type="l")
plot(chain[2,], type="l")

plot(chain1[1,], type="l", col=1, ylim=c(-0.03, 0.03), xlab="Iteration", ylab=expression(alpha)
lines(chain2[1,], col=2)
lines(chain3[1,], col=3)
lines(chain4[1,], col=4)

plot(chain1[2,], type="l", col=1, ylim=c(0, 1.3), xlab="Iteration", ylab=expression(beta))
lines(chain2[2,], col=2)
lines(chain3[2,], col=3)
```

```
lines(chain4[2,], col=4)

# Effective sample size
library(coda)
effectiveSize(chain[1,])
effectiveSize(chain[2,])

# ACF plots
acf(t(chain))

# Gelman plots
X0s <- cbind(rnorm(5, 0, 0.1), rnorm(5, 0, 0.1))
chains <- lapply(1:5, function(i) mcmc(MHpost(init=X0s[i,], n=1e3)[2,]))
chains <- mcmc.list(chains)
gelman.diag(chains)
gelman.plot(chains)


### Question 3
library(scales)
plot(chain[1,], chain[2,], pch=16, col=alpha("cornflowerblue", 0.2), xlab=expression(alpha),
     ylab=expression(beta), xlim=c(-0.25, 0.25))


### Question 4
mean(chain[1,][-c(1:1000)])
mean(chain[2,][-c(1:1000)])
mylm <- lm(y~t+cos(omega*t)-1, data=data)
summary(mylm)


### Question 5
tpred <- 55
p <- function(y, tpred) mean((dnorm(y, chain[1,]*tpred+chain[2,]*cos(omega*tpred), sigma))[-c(
y <- seq(-1.5, 1.5, by=0.001)
pr <- Vectorize(p)(y, tpred)
plot(y, pr, type="l", xlab="y*", ylab="Posterior predictive distribution")
y[which(pr == max(pr))]
predict(mylm, newdata=data.frame(t=55))

tt <- seq(0, 60, by=.1)
pred <- mean(chain[1,][-c(1:1000)])*tt + mean(chain[2,][-c(1:1000)])*cos(omega*tt)
plot(tt, pred, pch=16, col="royalblue", type="l", lwd=2,
     xlab="t", ylab="Response and predicted values", xlim=c(0, 65))
points(data$t, data$y, pch=16, col="firebrick")
legend("bottomright", legend=c("Data", "E[Yt]"), col=c("firebrick", "royalblue"),
       pch=c(16, 16))


### Question 6
# a
quantile(chain[1,][-c(0:1000)], c(0.025, 0.975))
```

```
# b
# Model M1
sigma <- sqrt(0.1)
w <- pi/10
log.like.1 <- function(theta){ # log-likelihood of M1
  alpha <- theta[1]
  beta <- theta[2]
  lp.beta <- dt(beta, 5, log = TRUE)
  lp.alpha <- dt(alpha, 5, log = TRUE)
  lp.xy.theta <- dnorm(
    Y,
    mean = alpha * t + beta * cos(w*t),
    sd = sigma,
    log = TRUE
  )
  lp.beta + lp.alpha + sum(lp.xy.theta)
}

# Model M2
log.like.2 <- function(beta){ # log-likelihood of M2
  lp.beta <- dt(beta, 5, log = TRUE)
  lp.xy.theta <- dnorm(
    Y,
    mean = beta * cos(w*t),
    sd = sigma,
    log = TRUE
  )
  lp.beta + sum(lp.xy.theta)
}

set.seed(1111111)
# Parameters to tune for a good acceptance rate
sigma.alpha <- 0.001
sigma.beta <- 0.07
sigma.trans = 2
# Equal probability for each model
r1 <- 0.5
r2 <- 0.5

pos <- c(0, 0.2) # Initial in M1
path <- list()
accept <- 0
n <- 1e4

for (i in 1:n){
  if (pos[1]==0){ # 1D
    if (runif(1)>r1){ # 1D -> 1D with prob 1-r1
      y <- rnorm(1,pos[2],sigma.beta)
      if (runif(1) <= min(1,exp(log.like.2(y)-log.like.2(pos[2])))){
        pos[2] <- y
        accept <- accept + 1/n
      }
    }
```

```
    else{ # 1D -> 2D with prob r1
      u <- rnorm(1,mean=0,sd=sigma.trans)
      alpha <- min(1,exp(log.like.1(c(u,pos[2])) - log.like.2(pos[2]))/dnorm(u,mean=0,sd=sigma
      if (runif(1)<alpha){
        pos <- c(1,u, pos[2])
        accept <- accept + 1/n
      }
    }
  }
  else{ # 2D
    if (runif(1)>r2){ # 2D -> 2D with prob 1-r2
      y <- c(rnorm(1,pos[2],sigma.alpha),rnorm(1,pos[3],sigma.beta))
      if (runif(1) <= min(1,exp(log.like.1(y)-log.like.1(pos[c(2,3)])))){
        pos[c(2,3)] <- y
        accept <- accept + 1/n
      }
    }
    else{ # 2D -> 1D with prob r2
      alpha <- min(1,exp(log.like.2(pos[3]) - log.like.1(c(pos[2],pos[3])))*dnorm(pos[2],mean=0
      if (runif(1)<alpha){
        pos <- c(0,pos[3])
        accept <- accept + 1/n
      }
    }
  }
  path[[length(path)+1]] <- pos
}
accept

mod1 <- c()
mod2 <- c()
for(i in 1:length(path)){
  if(length(path[[i]]) == 2){
    mod2 <- c(mod2, path[[i]][-c(1)])
  }
  else{
    mod1 <- rbind(mod1, path[[i]][-c(1)])
  }
}

# Number of samples in each model
dim(mod1)[1]
length(mod2)
dim(mod1)[1]/n
length(mod2)/n

# Trace plots M1
plot(mod1[,1], type="l", xlab="Iterations", ylab=expression(alpha))
plot(mod1[,2], type="l", xlab="Iterations", ylab=expression(beta))
# Trace plot M2
plot(mod2, type="l", xlab="Iterations", ylab=expression(beta))

# Estimates
```

```
mean(mod1[,1])
mean(mod1[,2])
mean(mod2)

# Mixing
acf(mod1)
effectiveSize(mcmc(mod1))

acf(mod2)
effectiveSize(mcmc(mod2))

### Question 7
alpha <- mean(chain[1,][-c(1:1000)])
beta <- mean(chain[2,][-c(1:1000)])

logpost <- function(w) sum(dnorm(Y, alpha*t+beta*cos(w*t), sd=sigma, log=TRUE)) +
  dunif(w, 0, pi, log=TRUE)

set.seed(42)
sigma.om <- seq(0.01, 1, length.out=40)
start <- seq(0, pi, length.out=40)
values <- matrix(NA, nrow=40, ncol=40)
cpt <- 0
i <- 0
j <- 0
for(sigma.w in sigma.om){
  i <- i + 1
  j <- 0
  for(init in start){
    cpt <- cpt + 1
    j <- j + 1
    cat(cpt, c(i,j), "\n")
    MHpost <- function(n, init){
      xall <- matrix(NA,ncol=n, nrow=1)
      x <- init
      xall[,1] <- x
      for (t in seq(2,n)){
        y <- rnorm(1, x, sigma.w)
        if(runif(1)<=(exp(logpost(y)-logpost(x)))){
          x <- y
        }
        xall[,t] <- x
      }
      return(xall)
    }
    omega.chain1 <- MHpost(3e3, init)[1,][-c(1:500)]
    omega.chain2 <- MHpost(3e3, init)[1,][-c(1:500)]
    omega.chain3 <- MHpost(3e3, init)[1,][-c(1:500)]
    values[i,j] <- mean(c(omega.chain1, omega.chain2, omega.chain3))
  }
}
library("fields")
image.plot(x=sigma.om, y=start, z=values, col=viridis(100), xlab="Proposal standard deviation"
```

```
           ylab="Initial value")



MHpost <- function(n, init, sd){
  sigma.w <- sd
  xall <- matrix(NA,ncol=n, nrow=1)
  x <- init
  xall[,1] <- x
  for (t in seq(2,n)){
    y <- rnorm(1, x, sigma.w)
    if(runif(1)<=(exp(logpost(y)-logpost(x)))){
      x <- y
    }
    xall[,t] <- x
  }
  return(xall)
}

omega.chain1 <- MHpost(1e4, 0, 0.1)[1,]
omega.chain2 <- MHpost(1e4, 0.5, 0.1)[1,]
omega.chain3 <- MHpost(1e4, 1, 0.1)[1,]
omega.chain4 <- MHpost(1e4, 1.5, 0.1)[1,]
omega.chain5 <- MHpost(1e4, 2, 0.1)[1,]
full.chain <- c(omega.chain1, omega.chain2, omega.chain3, omega.chain4, omega.chain5)

plot(omega.chain1, type="l", col=1, ylim=c(min(full.chain), max(full.chain)),
     xlab="Iteration", ylab=expression(omega))
lines(omega.chain2, col=2)
lines(omega.chain3, col=3)
lines(omega.chain4, col=4)
lines(omega.chain5, col=6)
legend(7500, 1.8, legend=c(expression(omega[0]==0),
                           expression(omega[0]==0.5),
                           expression(omega[0]==1),
                           expression(omega[0]==1.5),
                           expression(omega[0]==2)),
       col=c(1,2,3,4,6), lty=1)
```