ST4234: Bayesian Statistics

Tutorial 9 Solution, AY 19/20

Solutions

1.(a) The joint distribution of $(z, y_1 - z, y_2, y_3, y_4)$ is Multinomial$\left(n; \frac{1}{2}, \frac{\theta}{4}, \frac{1-\theta}{4}, \frac{1-\theta}{4}, \frac{\theta}{4}\right)$. Therefore,

$$\begin{aligned}
p(\theta, z|\boldsymbol{y}) &= \frac{p(\theta, z, \boldsymbol{y})}{p(\boldsymbol{y})} \\
&\propto p(\theta, z, \boldsymbol{y}) \\
&= p(\boldsymbol{y}, z|\theta)p(\theta) \\
&= p(z, y_1 - z, y_2, y_3, y_4|\theta)p(\theta) \\
&= \frac{n!}{z!(y_1 - z)!y_2!y_3!y_4!}\left(\frac{1}{2}\right)^z\left(\frac{\theta}{4}\right)^{y_1-z}\left(\frac{1-\theta}{4}\right)^{y_2}\left(\frac{1-\theta}{4}\right)^{y_3}\left(\frac{\theta}{4}\right)^{y_4} \cdot 1 \\
&\propto \frac{1}{z!(y_1 - z)!}\left(\frac{1}{2}\right)^z\left(\frac{1}{4}\right)^{-z}\theta^{y_1+y_4-z}(1-\theta)^{y_2+y_3} \\
&= \frac{1}{z!(y_1 - z)!}2^z\theta^{y_1+y_4-z}(1-\theta)^{y_2+y_3}.
\end{aligned}$$

(b) We find the two conditional posteriors as follows

$$\begin{aligned}
p(\theta|z, \boldsymbol{y}) &= \frac{p(\theta, z|\boldsymbol{y})}{p(z|\boldsymbol{y})} \propto p(\theta, z|\boldsymbol{y}) \\
&\propto \theta^{y_1+y_4-z}(1-\theta)^{y_2+y_3} \\
&\implies \theta|z, \boldsymbol{y} \sim \text{Beta}(y_1 + y_4 - z + 1, y_2 + y_3 + 1), \\
p(z|\theta, \boldsymbol{y}) &= \frac{p(z, \theta|\boldsymbol{y})}{p(\theta|\boldsymbol{y})} \propto p(z, \theta|\boldsymbol{y}) \\
&\propto \frac{1}{z!(y_1 - z)!}2^z\theta^{y_1+y_4-z} \\
&\propto \binom{y_1}{z}\left(\frac{2}{2+\theta}\right)^z\left(\frac{\theta}{2+\theta}\right)^{y_1-z} \\
&\implies z|\theta, \boldsymbol{y} \sim \text{Binomial}\left(y_1, \frac{2}{2+\theta}\right).
\end{aligned}$$

Therefore, we can write the Gibbs sampling algorithm to iteratively draw θ from Beta$(y_1 + y_4 - z + 1, y_2 + y_3 + 1)$ and draw z from Binomial $\left(y_1, \frac{2}{2+\theta}\right)$. The Gibbs sampler is given as follows.

(i) Initialize $\theta^{(0)}$ as a number in $(0, 1)$.

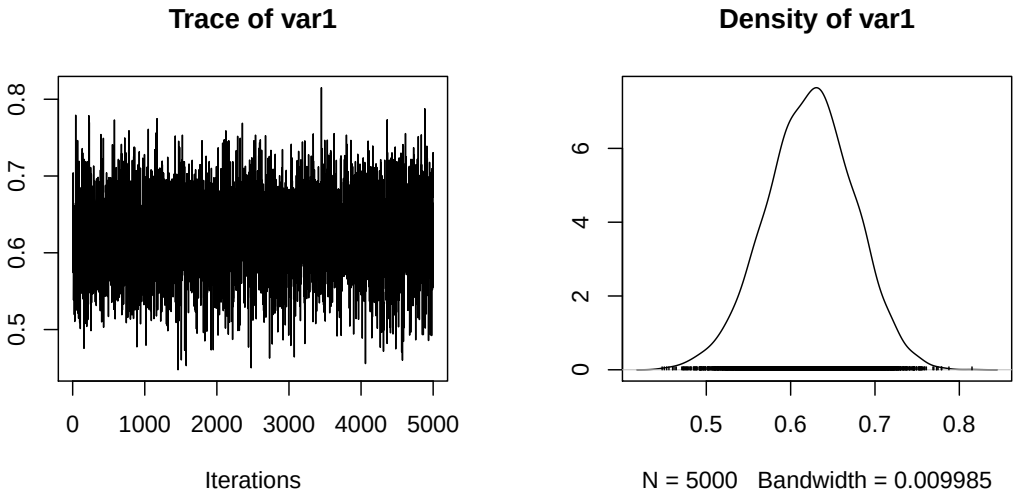
(ii) For $t = 1, \ldots, T$,

Draw $z^{(t)}$ from Binomial $\left(y_1, \frac{2}{2+\theta^{(t-1)}}\right)$.

Draw $\theta^{(t)}$ from Beta$(y_1 + y_4 - z^{(t)} + 1, y_2 + y_3 + 1)$.

The following R codes draw 10^4 samples of θ and z from the posterior using Gibbs sampler. We discard the first 5000 runs as burn-in. We first set $\boldsymbol{y} = (125, 18, 20, 34)$.

```
require(LearnBayes)

## Loading required package:  LearnBayes

require(coda)

## Loading required package:  coda

y <- c(125,18,20,34)
# y <- c(14,0,1,5)
T <- 10^4
theta.trace <- numeric(T)
z.trace <- numeric(T)
theta.old <- 0.5        # initialize theta

set.seed(4234)
for (i in 1:T) {
        z.new <- rbinom(1,y[1],2/(2+theta.old))
        theta.new <- rbeta(1,y[1]-z.new+y[4]+1,y[2]+y[3]+1)
        theta.trace[i] <- theta.new
        z.trace[i] <- z.new
        theta.old <- theta.new
}
# traceplot and density plot of theta
plot(mcmc(theta.trace[5001:T]))
```

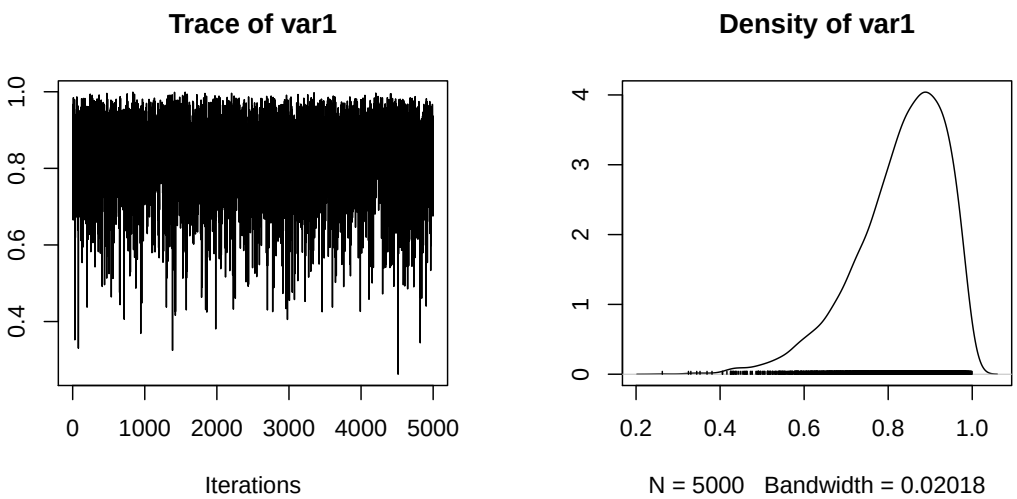


```r
# posterior mean and sd of theta
mean(theta.trace[5001:T])

## [1] 0.6230024

sd(theta.trace[5001:T])

## [1] 0.05179705
```

The posterior mean of θ is 0.6230, which matches well with the true posterior mean 0.6228 from Chapter 5 notes.

We then set $\boldsymbol{y} = (14, 0, 1, 5)$.

```r
y <- c(14,0,1,5)
T <- 10^4
theta.trace <- numeric(T)
z.trace <- numeric(T)
theta.old <- 0.5        # initialize theta

set.seed(4234)
for (i in 1:T) {
        z.new <- rbinom(1,y[1],2/(2+theta.old))
        theta.new <- rbeta(1,y[1]-z.new+y[4]+1,y[2]+y[3]+1)
        theta.trace[i] <- theta.new
        z.trace[i] <- z.new
        theta.old <- theta.new
}
# traceplot and density plot of theta
plot(mcmc(theta.trace[5001:T]))
```



```r
# posterior mean and sd of theta
mean(theta.trace[5001:T])

## [1] 0.8298638

sd(theta.trace[5001:T])

## [1] 0.1083922
```

The posterior mean of θ is 0.8299, which matches well with the true posterior mean 0.8311 from Chapter 5 notes.

2.(a) If we transform the parameters such that $\theta_A = \log \lambda_A$, then $\lambda_A = \mathrm{e}^{\theta_A}$ and $\frac{\mathrm{d}\lambda_A}{\mathrm{d}\theta_A} = \mathrm{e}^{\theta_A}$. Similarly, if $\theta_B = \log \lambda_B$, then $\lambda_B = \mathrm{e}^{\theta_B}$ and $\frac{\mathrm{d}\lambda_B}{\mathrm{d}\theta_B} = \mathrm{e}^{\theta_B}$. Therefore, applying the change-of-variable technique, the induced prior is

$$p(\theta) = p(\theta_A, \theta_B) \propto \frac{1}{\mathrm{e}^{\theta_A}\mathrm{e}^{\theta_B}} \cdot \mathrm{e}^{\theta_A}\mathrm{e}^{\theta_B} = 1.$$

The joint posterior is thus

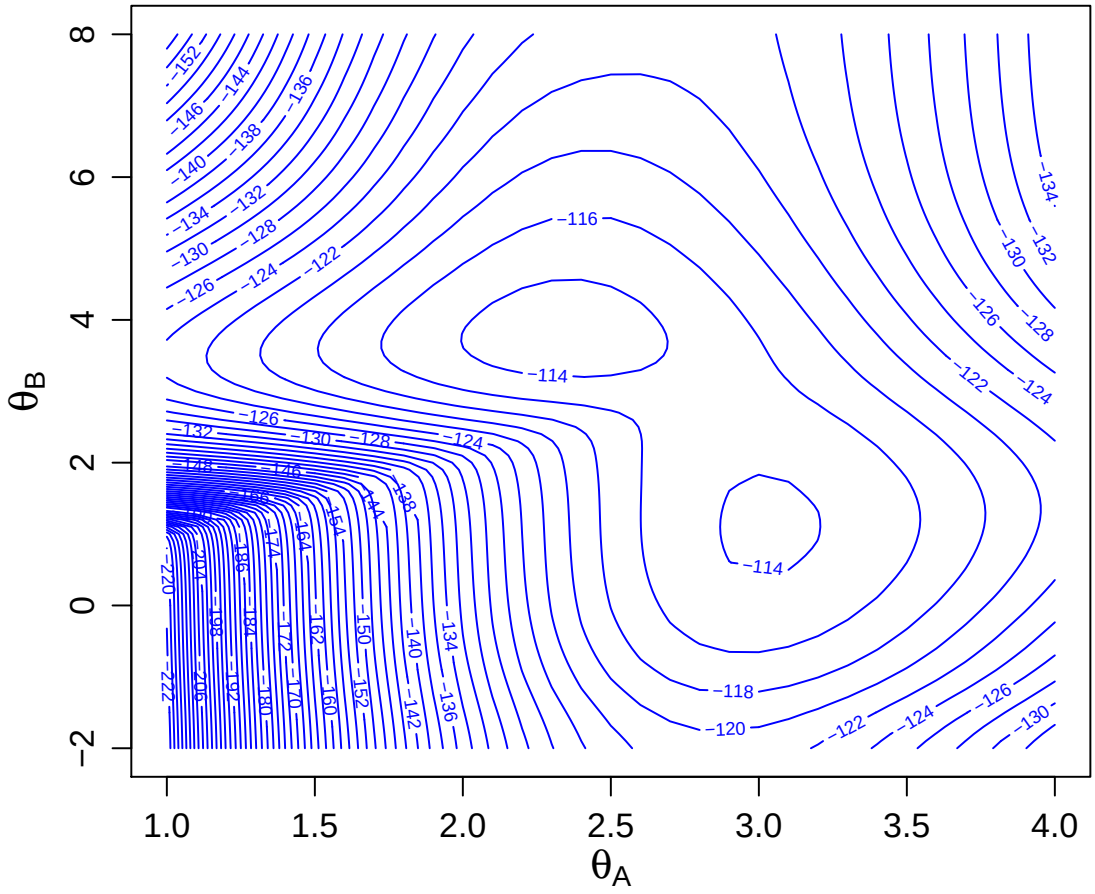
$$\begin{aligned} p(\theta|\boldsymbol{y}) &\propto p(\boldsymbol{y}|\theta)p(\theta) \\ &\propto \prod_{i=1}^{n} p(y_i|\theta_A, \theta_B) \\ &\propto \prod_{i=1}^{n} \left[0.8\frac{\exp(-y_i/\lambda_A)}{\lambda_A} + 0.2\frac{\exp(-y_i/\lambda_B)}{\lambda_B}\right]. \\ \Rightarrow \log p(\theta|\boldsymbol{y}) &= \underbrace{\sum_{[i=1}^{n} \log \left[0.8\frac{\exp(-y_i/\lambda_A)}{\lambda_A} + 0.2\frac{\exp(-y_i/\lambda_B)}{\lambda_B}\right]}_{\ell(\theta)} + C. \end{aligned}$$

where $\lambda_A = \mathrm{e}^{\theta_A}$, $\lambda_B = \mathrm{e}^{\theta_B}$, and C is an additive constant not depending on θ.

(b) The R-code below defines the function `logpost` for computing $\ell(\theta)$, and constructs the contour plot.

```
y <- c(9.3,4.9,3.5,26.0,0.6,1.0,3.5,26.9,2.6,20.4,1.0,10.0,1.7,11.3,7.7,
      14.1,24.8,3.8,8.4,1.1,24.5,90.7,16.4,30.7,8.5,5.9,14.7,0.5,99.5,35.2)
n <- length(y)

logpost <- function(theta,y){
        lambdaA <- exp(theta[1])
        lambdaB <- exp(theta[2])
        return(sum(log(0.8*exp(-y/lambdaA)/lambdaA +
               0.2*exp(-y/lambdaB)/lambdaB)))
}

# contour plot
theta1 <- seq(from=1, to=4, by=0.1)
theta2 <- seq(from=-2, to=8, by=0.1)
z <- matrix(0,length(theta1),length(theta2))
for (i in 1:length(theta1)){
        for (j in 1:length(theta2)){
                theta <- c(theta1[i], theta2[j])
                z[i,j] <- logpost(theta,y)
}}
```

```
par(mar=c(3.5,3.5,1,1))
par(mgp=c(2.1,0.8,0))
contour(x=theta1,y=theta2,z,col="blue",nlevels=40,
        xlab=expression(theta[A]),ylab=expression(theta[B]),
        cex.axis=1.1,cex.lab=1.3)
```



(c) With a starting guess of $(\theta_A, \theta_B) = (3, 0)$, the posterior mode is $(3.03, 1.10)$.

```
(out1 <- optim(par=c(3,0),fn=logpost,hessian=TRUE,control=list(fnscale=-1),y=y))
```

```
## $par
## [1] 3.029745 1.097967
##
## $value
## [1] -113.68
##
## $counts
## function gradient
##       51       NA
##
```

```
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##            [,1]      [,2]
## [1,] -21.268899  0.112227
## [2,]   0.112227 -1.331868
```

(d) With a starting guess of $(\theta_A, \theta_B) = (2, 4)$, the posterior mode is (2.33, 3.78).

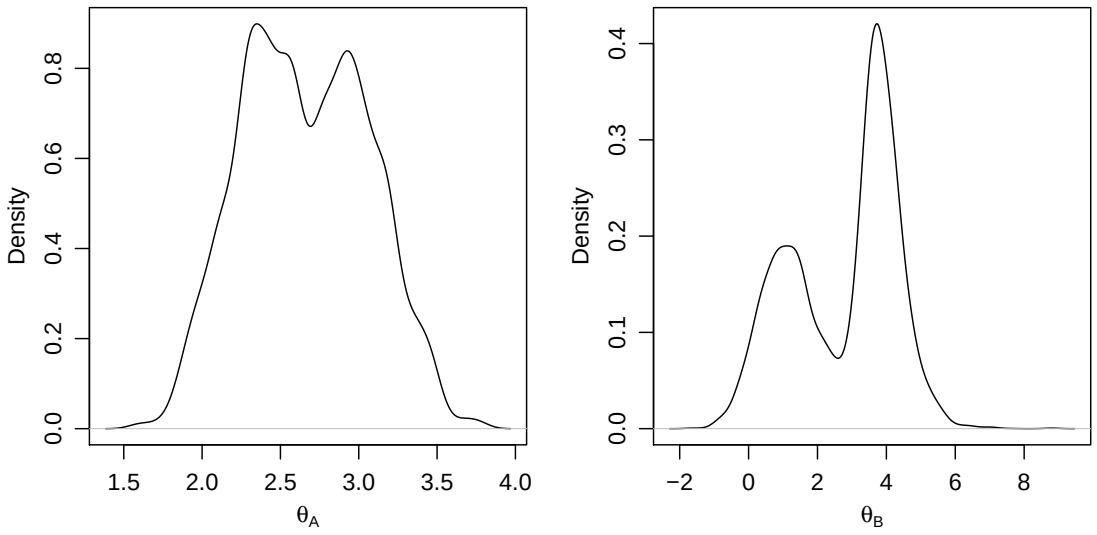
```
(out2 <- optim(par=c(2,4),fn=logpost,hessian=TRUE,control=list(fnscale=-1),y=y))

## $par
## [1] 2.330166 3.776908
##
## $value
## [1] -113.0161
##
## $counts
## function gradient
##       45       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##             [,1]       [,2]
## [1,] -17.1893784  0.1178333
## [2,]   0.1178333 -4.5828566
```

(e) From the contour plot in Part (b), the posterior distribution has more than one mode. Hence which mode we converge to in our numerical optimization depends on the starting value in `optim()`.

(f) We first use the posterior mode (3.03, 1.10) as the starting value and the covariance matrix of the normal approximation about this mode as the covariance matrix in

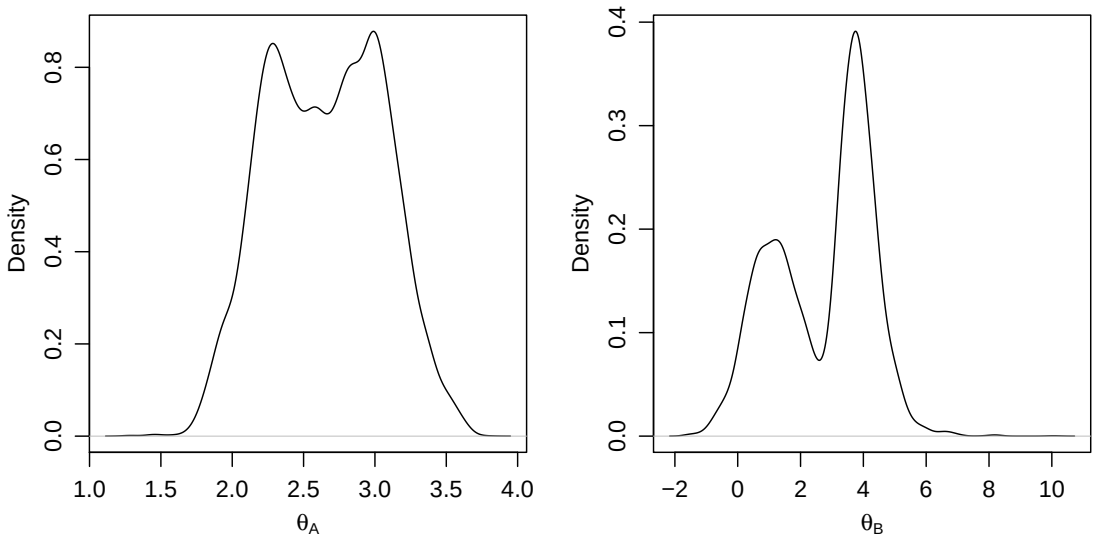
the multivariate normal proposal density. The scale is set as 3 to encourage random walk with longer ranges. Using `rwmetrop`, we run the chain for 10^4 iterations. Density estimates for θ_A and θ_B are obtained from the simulated draws using the `density` function and plotted in the codes below, with the first 5000 draws discarded as burn-in. The accpetance rate is about 0.20.

```
# starting at the 1st mode
post.mode <- out1$par
post.cov <- -solve(out1$hessian)

T <- 10^4
start <- post.mode
proposal <- list(var=post.cov,scale=3)
set.seed(4234)
fit1.1<- rwmetrop(logpost,proposal,start,T,y)
fit1.1$accept

## [1] 0.1984

# density plot
par(mar=c(3.5,3.5,1,1))
par(mgp=c(2.1,0.8,0))
par(mfrow=c(1,2))
plot(density(fit1.1$par[5001:T,1]),main="",xlab=expression(theta[A]))
plot(density(fit1.1$par[,2]),main="",xlab=expression(theta[B]))
```



We then use the posterior mode $(2.33, 3.78)$ as the starting value and redo the random walk Metropolis. The acceptance rate is about 0.25.

```r
# starting at the 2nd mode
post.mode <- out2$par
post.cov <- -solve(out2$hessian)

T <- 10^4
start <- post.mode
proposal <- list(var=post.cov,scale=3)
set.seed(4234)
fit1.2 <- rwmetrop(logpost,proposal,start,T,y)
fit1.2$accept

## [1] 0.2499

# density plot
par(mar=c(3.5,3.5,1,1))
par(mgp=c(2.1,0.8,0))
par(mfrow=c(1,2))
plot(density(fit1.2$par[,1]),main="",xlab=expression(theta[A]))
plot(density(fit1.2$par[,2]),main="",xlab=expression(theta[B]))
```

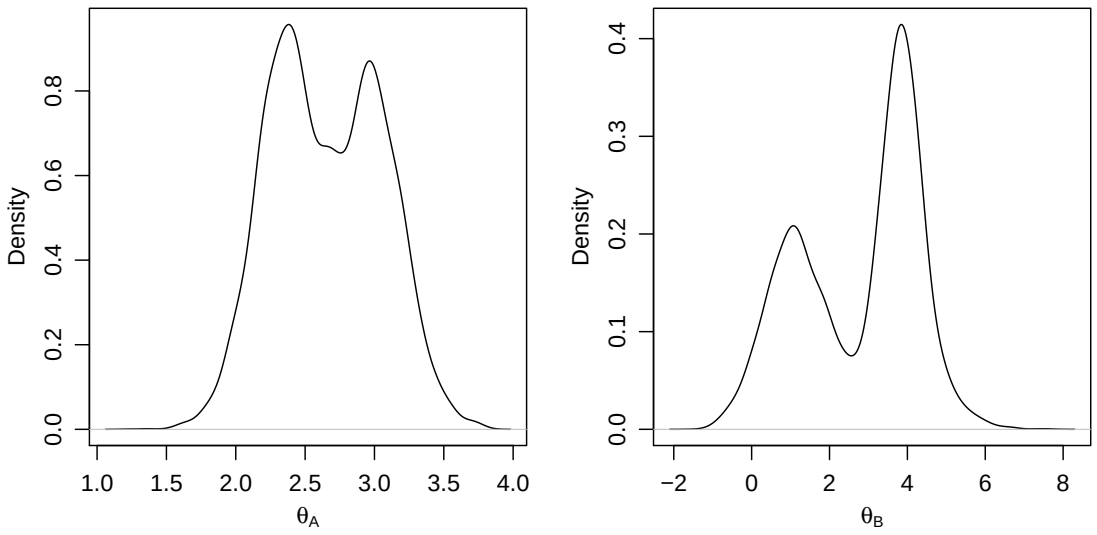


No matter which mode we start with, the marginal posterior densities of both θ_A and θ_B appear to be bimodal. However, the shapes are very different, indicating that at least one of the two chains have not converged to the stationary distribution. We will determine the convergence of these two Markov chains in Part (h).

(g) To construct the Metropolis within Gibbs sampler, we first use the posterior mode $(3.03, 1.10)$ as the starting value and set the scale as 2 times the standard deviation of the parameters in the normal approximation. Using the function `gibbs`, we run the chain for 10^4 iterations. Density estimates for θ_A and θ_B are obtained from the

simulated draws using the `density` function and plotted in the codes below, with the first 5000 draws discarded as burn-in. The accpetance rate for θ_A and θ_B are 0.53 and 0.45, respectively.

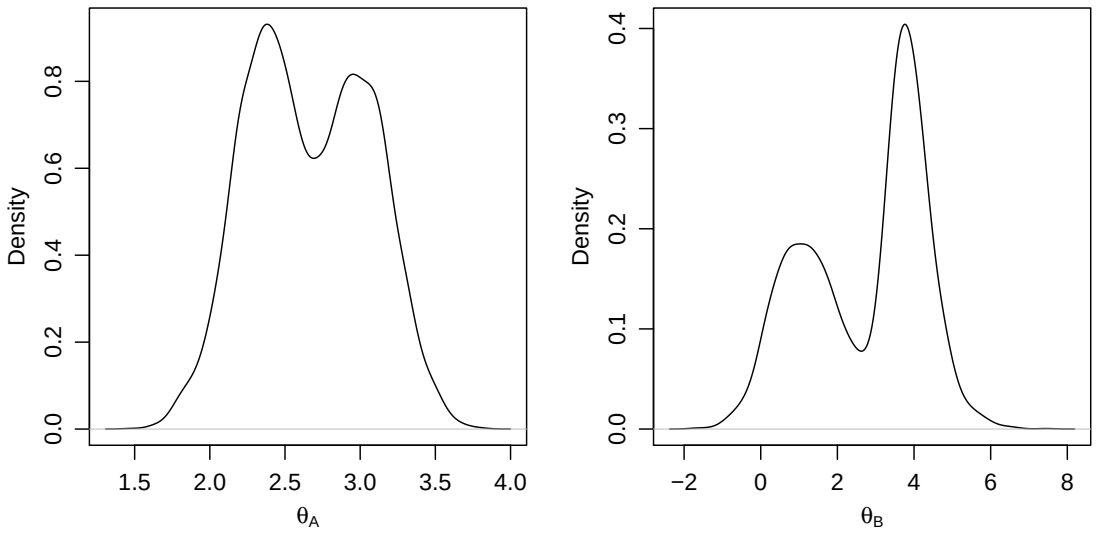
```
# starting at the 1st mode
post.mode <- out1$par
post.cov <- -solve(out1$hessian)
post.sd <- sqrt(diag(post.cov))
set.seed(4234)
fit2.1 <- gibbs(logpost,start,T,2*post.sd,y)
fit2.1$accept

##         [,1]   [,2]
## [1,] 0.5271 0.4528

# density plot
par(mar=c(3.5,3.5,1,1))
par(mgp=c(2.1,0.8,0))
par(mfrow=c(1,2))
plot(density(fit2.1$par[,1]),main="",xlab=expression(theta[A]))
plot(density(fit2.1$par[,2]),main="",xlab=expression(theta[B]))
```



We then use the posterior mode $(2.33, 3.78)$ as the starting value and redo the Metropolis within Gibbs as above. The accpetance rate for θ_A and θ_B are 0.50 and 0.62, respectively.

```
# starting at the 2nd mode
post.mode <- out2$par
post.cov <- -solve(out2$hessian)
post.sd <- sqrt(diag(post.cov))
```

```
set.seed(4234)
fit2.2 <- gibbs(logpost,start,T,2*post.sd,y)
fit2.2$accept

##         [,1]  [,2]
## [1,] 0.4972 0.624

# density plot
par(mar=c(3.5,3.5,1,1))
par(mgp=c(2.1,0.8,0))
par(mfrow=c(1,2))
plot(density(fit2.2$par[,1]),main="",xlab=expression(theta[A]))
plot(density(fit2.2$par[,2]),main="",xlab=expression(theta[B]))
```

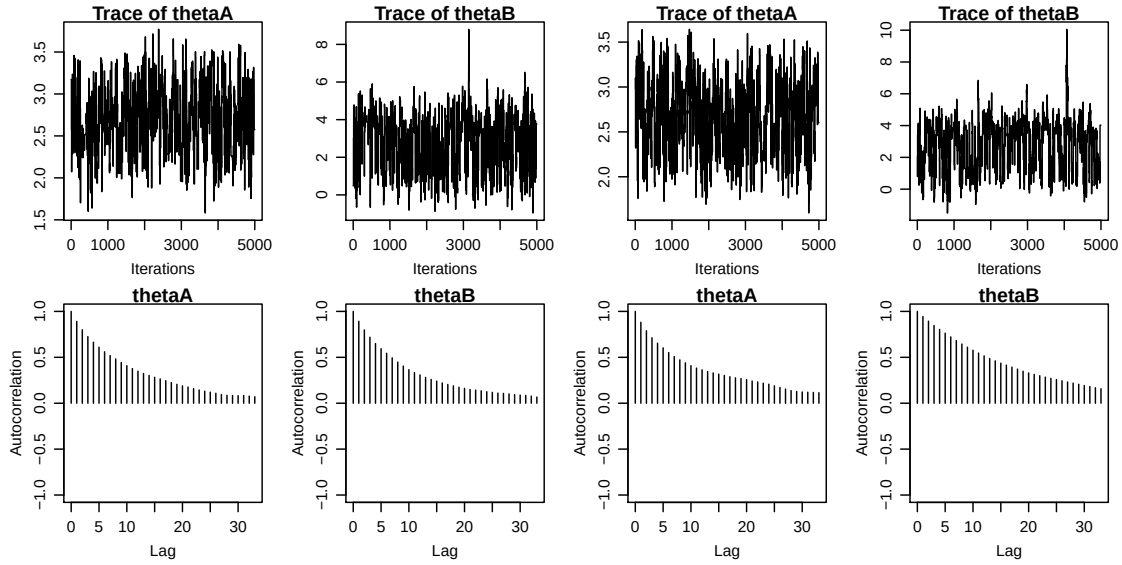


No matter which mode we start with, the marginal posterior densities of both θ_A and θ_B appear to be bimodal. The shapes are similar to each other, unlike the random walk Metropolis algorithm in Part (f).

(h) We plot all the trace plots and autocorrelation plots from the algorithms in Part (f) and Part (g) as below. First, the two chains from random walk Metropolis with two different starting values are plotted as follows.

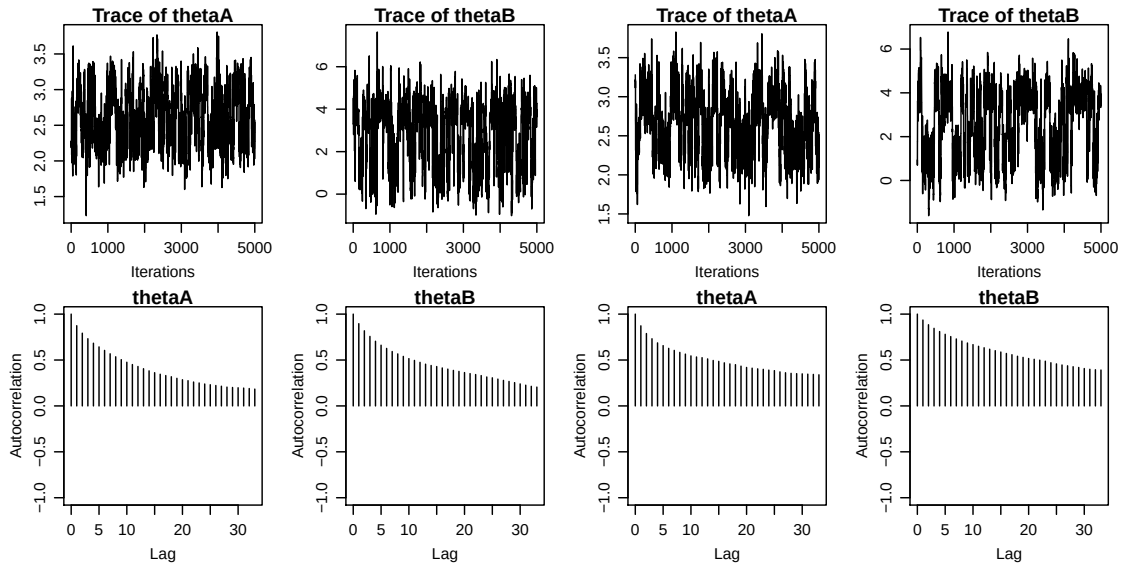
```
mcmcobj1.1 <- mcmc(fit1.1$par[5001:T,])
mcmcobj1.2 <- mcmc(fit1.2$par[5001:T,])
mcmcobj2.1 <- mcmc(fit2.1$par[5001:T,])
mcmcobj2.2 <- mcmc(fit2.2$par[5001:T,])
colnames(mcmcobj1.1) <- c("thetaA","thetaB")
colnames(mcmcobj1.2) <- c("thetaA","thetaB")
colnames(mcmcobj2.1) <- c("thetaA","thetaB")
colnames(mcmcobj2.2) <- c("thetaA","thetaB")
```

```
par(mar=c(3.5,3.5,1,1))
par(mgp=c(2.1,0.8,0))
par(mfrow=c(2,4))
traceplot(mcmcobj1.1)
traceplot(mcmcobj1.2)
autocorr.plot(mcmcobj1.1,auto.layout=FALSE)
autocorr.plot(mcmcobj1.2,auto.layout=FALSE)
```



The convergence performances are similar for these two random walk Metropolis chains. The autocorrelation plots show that both chains have decaying autocorrelation in both θ_A and θ_B. But the decaying in θ_B for the 2nd chain seems to be slower than the 1st chain.

```
par(mar=c(3.5,3.5,1,1))
par(mgp=c(2.1,0.8,0))
par(mfrow=c(2,4))
traceplot(mcmcobj2.1)
traceplot(mcmcobj2.2)
autocorr.plot(mcmcobj2.1,auto.layout=FALSE)
autocorr.plot(mcmcobj2.2,auto.layout=FALSE)
```



The convergence performances are also similar for these two Metropolis within Gibbs chains. The autocorrelation plots show that both chains have decaying autocorrelation in both θ_A and θ_B. But the decaying in θ_A and θ_B for the 2nd chain seems to be slower than the 1st chain.

Finally we compare the two panels of plots. The trace plots of the random walk Metropolis sampler resemble more of random noise while the Metropolis within Gibbs sampler tends to show slightly more visible trends. It is possible that the Metropolis within Gibbs has explored one mode for a while before moving to the other mode. The sample values tend to get "stuck" in certain regions and does not move between the two modes as frequently.

This example shows that it is generally challenging for MCMC algorithms to explore multimodal posterior distributions.