STAT 4010 Bayesian Learning

Tutorial 10

Spring 2023

Cheuk Hin (Andy) CHENG (Email | Homepage)

Di SU (Email | Homepage)

1 Gibbs Sampler

We want to draw d-dimensional sample $\theta_j = (\theta_{j1}, \ldots, \theta_{jd})^{\mathrm{T}}$ from $\pi(\theta)$.

Algorithm 1: Gibbs Sampler

Input: (i) number of iteration J; (ii) conditional PDF $\pi^{(k|-k)}(\cdot \mid \theta_{-k})$ for $k = 1, \ldots, d$; and (iii) initialization PDF $\pi_{\text{initial}}(\cdot)$.

begin
- (1) Generate $\theta_0 \sim \pi_{\text{initial}}(\cdot)$
- (2) Set $\vartheta \leftarrow \theta_0$
- (3) **for** j *in* $\{1, \ldots, J\}$ **do**
  - **for** k *in* $\{1, \ldots, d\}$ **do**
    - Generate $\theta_{jk} \sim \pi^{(k|-k)}(\cdot \mid \vartheta_{-k})$
    - Update the k th component of ϑ as $\vartheta_k \leftarrow \theta_{jk}$.
  - **end**
- **end**

end

Output: $\theta_{1:J}$

Theorem 1.1. *Gibbs sampler is a composition of d MH algorithm with acceptance probabilities in each step always equals to 1.*

When the full conditionals are not easy to sample from, we can make use of MH algorithm.

Algorithm 2: MH-within-Gibbs Sampler

Input: (i) number of iteration J; (ii) proposal PDF $p^{(k)}(\theta_k \mid \theta_{-k})$ for $k = 1, \ldots, d$; (iii) conditional PDF $\pi_u^{(k|-k)}(\cdot \mid \theta_{-k})$ for $k = 1, \ldots, d$; and (iv) initialization PDF $\pi_{\text{initial}}(\cdot)$.

begin

(1) Generate $\theta_0 \sim \pi_{\text{initial}}(\cdot)$

(2) Set $\vartheta \leftarrow \theta_0$

(3) **for** j *in* $\{1, \ldots, J\}$ **do**

for k *in* $\{1, \ldots, d\}$ **do**

Generate $\tilde{\theta}_{jk} \sim p^{(k)}(\cdot \mid \vartheta)$

Generate $U_{jk} \sim \text{Unif}(0, 1)$

Compute the acceptance probability

$$a_{jk} = \min\left\{1, \frac{\pi_{\text{u}}^{(k|-k)}\left(\widetilde{\theta}_{jk} \mid \vartheta_{-k}\right) p_{\text{u}}^{(k)}\left(\theta_{j-1,k} \mid \tilde{\theta}_{jk}, \vartheta_{-k}\right)}{\pi_{\text{u}}^{(k|-k)}\left(\theta_{j-1,k} \mid \vartheta_{-k}\right) p_{\text{u}}^{(k)}\left(\widetilde{\theta}_{jk} \mid \theta_{j-1,k}, \vartheta_{-k}\right)}\right\}$$

Compute $\theta_{jk} = \tilde{\theta}_{jk}\mathbb{1}(U_{jk} \leq a_{jk}) + \theta_{j-1,k}\mathbb{1}(U_{jk} > a_{jk})$

Update the k th component of ϑ as $\vartheta_k \leftarrow \theta_{jk}$

end

end

end

Output: $\theta_{1:J}$

Remark 1.1. Notice that the unnormalized density $\pi_{\text{u}}^{(k|-k)}\left(\widetilde{\theta}_{jk} \mid \vartheta_{-k}\right)$ is not conditioned on $\theta_{j-1,k}$.

2 Examples

Example 2.1. (Exercise 6.2 A6 2021) Let $x_1, \ldots, x_n$ be the numbers of reported COVID-19 cases in Hong Kong from 1 February 2020 to 10 April 2020 (i.e., $n = 70$), respectively. The dataset can be downloaded from the HKSAR government dataset (click here). Some people believe that the distribution of $x_1, \ldots, x_{\tau-1}$ is different from that of $x_\tau, \ldots, x_n$ for some $2 \leq \tau \leq n$. Suppose that the data are modeled by a change-point model as follows:

$$\begin{aligned}
[x_i \mid \tau, \theta_1, \theta_2] &\overset{\text{IID}}{\sim} \begin{cases} \text{Po}(\theta_1) & \text{if } i = 1, \ldots, \tau - 1; \\ \text{Po}(\theta_2) & \text{if } i = \tau, \ldots, n; \end{cases} \\
\theta_1, \theta_2 &\overset{\text{ID}}{\sim} \text{Ga}(\alpha)/\beta \\
\tau &\sim \text{Unif}\{2, \ldots, n\},
\end{aligned}$$

where $\alpha, \beta > 0$ are non-random and are suitably chosen by you. The goals of this exercise are (i) to perform statistical inference on τ, θ_1, θ_2; and (ii) to learn from data and give statistically grounded suggestions.

1. Derive the conditional densities of $[\tau \mid \theta_1, \theta_2, x_{1:n}]$, $[\theta_1 \mid \tau, \theta_2, x_{1:n}]$, and $[\theta_2 \mid \tau, \theta_1, x_{1:n}]$.
2. Use a suitable MCMC method to draw posterior samples of τ, θ_1, θ_2 with $J = 2^{13}$ iterations. Discard the first half as burn-in.
3. Visualize your MCMC sample produced in part 2. Comment the quality of your MCMC sample.

Solution:

1. Note that

$$\begin{aligned} f(\tau \mid \theta_1, \theta_2, x_{1:n}) &\propto f(x_{1:n} \mid \tau, \theta_1, \theta_2)\, f(\tau \mid \theta_1, \theta_2) \\ &\propto \left(\prod_{i=1}^{\tau-1} e^{-\theta_1}\theta_1^{x_i}\right)\left(\prod_{i=\tau}^{n} e^{-\theta_2}\theta_2^{x_i}\right) \mathbb{1}(\tau \in \{2, \ldots, n\}) \\ &= \exp\left\{-\tau\theta_1 - (n-\tau)\theta_2 + \left(\sum_{i=1}^{\tau-1} x_i\right)\ln\theta_1 + \left(\sum_{i=\tau}^{n} x_i\right)\ln\theta_2\right\} \mathbb{1}(\tau \in \{2, \ldots, n\}) \end{aligned}$$

Similarly, we have

$$\begin{aligned} f(\theta_1 \mid \tau, \theta_2, x_{1:n}) &\propto f(x_{1:n} \mid \tau, \theta_1, \theta_2)\, f(\theta_1 \mid \tau, \theta_2) \\ &\propto \left(\prod_{i=1}^{\tau-1} e^{-\theta_1}\theta_1^{x_i}\right) \theta_1^{\alpha-1} e^{-\beta\theta_1} \mathbb{1}(\theta_1 > 0) \\ &\sim \mathrm{Ga}\left(\alpha + \sum_{i=1}^{\tau-1} x_i, \beta + \tau\right) \qquad \text{and} \\ f(\theta_2 \mid \tau, \theta_1, x_{1:n}) &\sim \mathrm{Ga}\left(\alpha + \sum_{i=\tau}^{n} x_i, \beta + n - \tau\right) \end{aligned}$$

2. Since we have derived all conditional densities, we can use the Gibbs sampler. While $f(\tau \mid \theta_1, \theta_2, x_{1:n})$ is not a named distribution, note that it is a PMF and so we can use the sample function. For simplicity, we take $\alpha = \beta = 1$.

```r
# Import and extract data
data = read.csv("enhanced_sur_covid_19_eng.csv")
t0 = as.numeric(as.Date("01/02/2020","%d/%m/%Y"))
t1 = as.numeric(as.Date("10/04/2020","%d/%m/%Y"))
x = t = rep(NA,t1-t0+1)
for (i in t0:t1){
  x[i-t0+1] = sum(as.Date(data$Report.date,"%d/%m/%Y")==i)
  t[i-t0+1] = as.character(as.Date("01/02/2020","%d/%m/%Y")+i-t0)
}
names(x) = t

# Ex6.2.2: sample from posterior
sim_tau <- function(theta1,theta2,x,n){
  tau = 2:n
  cs = cumsum(x)
  lp = -tau*theta1 -(n-tau)*theta2 +cs[1:(n-1)]*log(theta1) +(sum(x)-cs
      [1:(n-1)])*log(theta2)
  p = exp(lp-max(lp))
```

```r
  sample(tau, 1, prob=p/sum(p))
}

##initial proposal
#tau ~ Unif{2,..,n}
#theta1, theta2 ~iid Ga(1)/1

gibbs21 <- function(J,x,burn_frac = 0.5, keep = F, a = 1,b = 1){
  nsim = ceiling(J/burn_frac)+1
  n = length(x)
  theta = array(NA,c(nsim,3))
  colnames(theta) = c('tau','theta1','theta2')
  ##sample from the initial proposal
  theta[1,] = c(sample(2:n,1),rgamma(2,a,b))
  for (j in 2:nsim) {
    theta_use = theta[j-1,] ##current info
    ##Update tau
    theta_use[1] = sim_tau(theta_use[2],theta_use[3],x,n)
    ##Update theta1
    theta_use[2] = rgamma(1, a+sum(x[1:(theta_use[1]-1)]), b+theta_use[1])
    ##Update theta2
    theta_use[3] = rgamma(1, a+sum(x[theta_use[1]:n]), b+n-theta_use[1])

    ##update theta
    theta[j,] = theta_use
  }
  if (keep) {
    return(theta)
  }else{
    return(tail(theta,J)) ##burn the first fraction
  }

}

J_sim = 2^13
theta_name = c('tau','theta1','theta2')
theta_sim = gibbs21(J_sim,x,keep=TRUE)
theta_sim = theta_sim[-1,]#remove initialization
iUse = (J_sim/2+1):J_sim
```

3. The plots show that the MCMC sample produced in part 2 looks stationary and converge quickly. We can further compare the auto-correlation plots which shows that the burn-in has effectively reduced the stickiness of the chain as the cross-correlations are eliminated (we omit the 3-by-3 auto-correlation plots here for compactness).

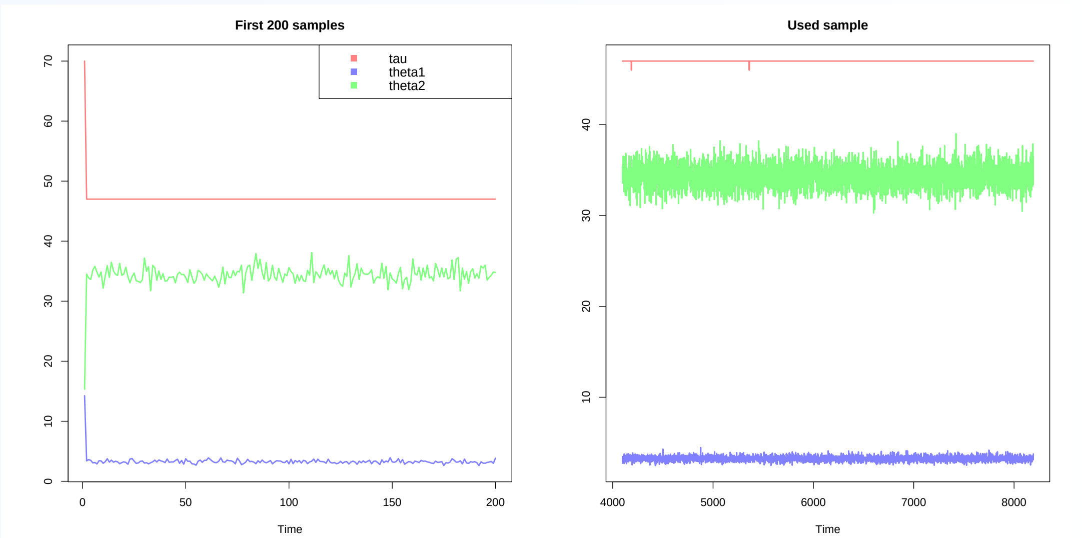
```r
transCol = function(color, percent=50) {
  v = col2rgb(color)
  newCol = rgb(v[1],v[2],v[3],max=255,alpha=(1-percent/100)*255)
  invisible(newCol)
}
par(mfrow=c(1,2), mar=c(4.5,5,3,2))
col = c(transCol("red", percent=50),
        transCol("blue", percent=50),
        transCol("green", percent=50))
matplot(1:200, theta_sim[1:200,], col=col, lwd=2, type="l", lty=1,
        ylab="", xlab="Time", main="First 200 samples")
legend("topright", c("tau","theta1","theta2"), col=col, pch=15, cex=1.2)
```

```
matplot(iUse, theta_sim[iUse,], col=col, lwd=2, type="l", lty=1,
        ylab="", xlab="Time", main="Used sample")
acf(theta_sim, mar=c(3,2.5,2,0.5))
acf(theta_sim[iUse,], mar=c(3,2.5,2,0.5))

##MCMC Diagnostic
##1) Density plot
windows(height=15,width = 15)
par(mfrow = c(3,3))
for (i in 1:3) {
  plot(density(theta_sim[,i], kernel="epanechnikov"),main = paste(' Density plot for',theta_name[i]),
       ylab = 'Density',xlab = expression(theta))
}
##2) ACF plot
for (i in 1:3) {
  acf = acf(theta_sim[,i],plot=F)
  plot(acf,main = paste('ACF of',theta_name[i]),xlab = 'Lag')
}

##3) Trace plot
for (i in 1:3) {
  plot((J_sim-500+1):J_sim,tail(theta_sim[,i],500),type = 'l',
       main = paste('Trace plot of',theta_name[i]),ylab = expression(theta),xlab = 'j')
}

##Cp analysis
cp = round(mean(theta_sim[,1]),0)
cp
n = length(x)
plot(1:n,x,type='l')
abline(v=cp,col = 'green')
```



Example 2.2 (Option pricing under double exponential model)**.** Consider the following

model,

$$
\begin{aligned}
r_{1:n} \mid \lambda_0, \lambda_1, p &\overset{\text{IID}}{\sim} f(r \mid \lambda_0, \lambda_1, p) = p\frac{1}{\lambda_1}e^{-r/\lambda_1}\mathbb{1}(r \geq 0) + (1-p)\frac{1}{\lambda_0}e^{r/\lambda_0}\mathbb{1}(r < 0) \\
\lambda_0, \lambda_1 &\overset{\text{IID}}{\sim} \text{InvGamma}(\mu = 5\%, \sigma = 1\%) \\
p &\sim Beta(\mu = 0.5, \sigma = 0.1),
\end{aligned}
$$

where r can be thought of log return for each unit of time. Note that the common representations for the parameters of Inverse-Gamma and Beta distribution are

$$
\begin{aligned}
\text{InvGamma}(\mu, \sigma) &= k/Ga(h) \\
h &= \mu^2/\sigma^2 + 2 \\
k &= \mu(h-1) \\
Beta(\mu, \sigma) &= Beta(\alpha, \beta) \\
\alpha &= \frac{\mu^2(1-\mu)}{\sigma^2} - \mu \\
\beta &= \alpha(1/\mu - 1).
\end{aligned}
$$

Define $P_n = \sum_{i:r_i \geq 0} r_i$, $N_n = \sum_{i:r_i<0} r_i$, n_1 be the number of positive $r_{1:n}$ and $n_0 = n - n_1$. From the data, we have $n = 100$, $n_1 = 55$, $P_n = 2.2$ and $N_n = -2.7$. Let $S_0 = 373$ and $K = 380$. Using MH-within-Gibbs sampler, estimate

$$
\mathsf{E}[\max(S_T - k, 0) \mid r_{1:n}] = \mathsf{E}[\max(S_0 e^{r_{n+1}} - k, 0) \mid r_{1:n}].
$$

Solution: Note that the joint sampling distribution is

$$
f(r_{1:n} \mid \lambda_0, \lambda_1, p) = p^{n_1}(1-p)^{n0}\lambda_1^{-n_1}\lambda_0^{-n_0}e^{-P_n/\lambda_1}e^{N_n/\lambda_0}.
$$

We first find the conditional distributions for the parameters.

$$
\begin{aligned}
f(\lambda_1 \mid r_{1:n}, \lambda_0, p) &\propto f(\lambda_1 \mid \lambda_0, p) f(r_{1:n} \mid \lambda_1, \lambda_0, p) \\
&= f(\lambda_1) f(r_{1:n} \mid \lambda_1, \lambda_0, p) \\
&\propto \lambda_1^{-h-1} e^{-k/\lambda_1}\left[p^{n_1}(1-p)^{n0}\lambda_1^{-n_1}\lambda_0^{-n_0}e^{N_n/\lambda_1}\right] \\
&= p^{n_1}(1-p)^{n_0}\lambda_1^{-h-n_1-1}\lambda_0^{-n_0}e^{-(P_n+k)/\lambda_1}e^{N_n/\lambda_0}\mathbb{1}(\lambda_1 > 0).
\end{aligned}
$$

Similarly,

$$
f(\lambda_0 \mid r_{1:n}, \lambda_1, p) \propto p^{n_1}(1-p)^{n_0}\lambda_1^{-n_1}\lambda_0^{-h-n_0-1}e^{-P_n/\lambda_1}e^{-(k-N_n)/\lambda_0}\mathbb{1}(\lambda_0 > 0).
$$

Moreover,

$$
f(p \mid r_{1:n}, \lambda_1, \lambda_0) \propto p^{n_1+\alpha-1}(1-p)^{n_0+\beta-1}\lambda_1^{-n_1}\lambda_0^{-n_0}e^{-P_n/\lambda_1}e^{N_n/\lambda_0}\mathbb{1}(p \in (0,1)).
$$

The discussion for the proposals are as followed.

- The conditional densities for λ_1 and λ_0 are proportional to inverse-gamma kernel. Therefore, we would set $\tilde{\lambda}_\cdot \mid \lambda_\cdot \sim \text{InvGamma}(\mu = \lambda_\cdot, \sigma = 0.01)$.
- The conditional density for p is proportional to beta kernel. Therefore, we would set $\tilde{p} \mid p \sim \text{Beta}(\mu = p, \sigma = 0.05)$.

Note that the conditional distributions are not exactly the commonly known distributions. We implement the MH algorithm to generate samples.

```r
library(invgamma)

log_pi_lambda1 <- function(Pn,Nn,n1,n0,lambda1,lambda0,p,h,k){
  n1*log(p)+n0*log(1-p)-(h+n1+1)*log(lambda1)-n0*log(lambda0)-(Pn+k)/lambda1+
      Nn/lambda0
}

log_pi_lambda0 <- function(Pn,Nn,n1,n0,lambda1,lambda0,p,h,k){
  n1*log(p)+n0*log(1-p)-n1*log(lambda1)-(h+n0+1)*log(lambda0)-Pn/lambda1-(k-Nn
      )/lambda0
}

log_pi_p <- function(Pn,Nn,n1,n0,lambda1,lambda0,p,a,b){
  (n1+a-1)*log(p)+(n0+b-1)*log(1-p)-n1*log(lambda1)-n0*log(lambda0)-Pn/lambda1
      +Nn/lambda0
}

get_invgamma_para <- function(mu,sd){
  h = mu^2/sd^2+2
  k = mu*(h-1)
  c(h,k)
}

get_beta_para <- function(mu,sd){
  a = mu^2*(1-mu)/sd^2-mu
  b = a*(1/mu-1)
  c(a,b)
}

##MH
MH_lambda1 <- function(Pn,Nn,n1,n0,lambda1,lambda0,p,sd_lambda1 = 0.01){
  ##represent the parameters
  ##parameters for numerator in proposal odd
  para = get_invgamma_para(lambda1,sd_lambda1)
  h = para[1]
  k = para[2]

  lambda1_p = rinvgamma(1,h,k)
  log_target_odd = log_pi_lambda1(Pn,Nn,n1,n0,lambda1 = lambda1_p,lambda0,p,h,
      k)-log_pi_lambda1(Pn,Nn,n1,n0,lambda1 = lambda1,lambda0,p,h,k)

  ##parameters for denominator in proposal odd
  para = get_invgamma_para(lambda1_p,sd_lambda1)
  h_d = para[1]
  k_d = para[2]
  log_proposal_odd = log(dinvgamma(lambda1_p,h,k)) - log(dinvgamma(lambda1,h_d
      ,k_d))
  accept_prob = exp(min(0,log_target_odd-log_proposal_odd))
  u = runif(1)
  lambda1_p*(u<=accept_prob)+lambda1*(u>accept_prob)
}

MH_lambda0 <- function(Pn,Nn,n1,n0,lambda1,lambda0,p,sd_lambda0 = 0.01){
  para = get_invgamma_para(lambda0,sd_lambda0)
  h = para[1]
  k = para[2]
```

```
  lambda0_p = rinvgamma(1,h,k)
  log_target_odd = log_pi_lambda0(Pn,Nn,n1,n0,lambda1,lambda0 = lambda0_p,p,h,k)-log_pi_lambda0(Pn,Nn,n1,n0,lambda1,lambda0,p,h,k)
  para = get_invgamma_para(lambda0_p,sd_lambda0)
  h_d = para[1]
  k_d = para[2]
  log_proposal_odd = log(dinvgamma(lambda0_p,h,k)) - log(dinvgamma(lambda0,h_d,k_d))
  accept_prob = exp(min(0,log_target_odd-log_proposal_odd))
  u = runif(1)
  lambda0_p*(u<=accept_prob)+lambda0*(u>accept_prob)
}

MH_p <- function(Pn,Nn,n1,n0,lambda1,lambda0,p,sd_p = 0.1){
  para = get_beta_para(p,sd_p)
  a = para[1]
  b = para[2]
  p_p = rbeta(1,a,b)
  log_target_odd = log_pi_p(Pn,Nn,n1,n0,lambda1,lambda0,p = p_p,a,b)-log_pi_p(Pn,Nn,n1,n0,lambda1,lambda0,p,a,b)
  para = get_beta_para(p_p,sd_p)
  a_d = para[1]
  b_d = para[2]
  log_proposal_odd = log(dbeta(p_p,a,b)) - log(dbeta(p,a_d,b_d))
  accept_prob = exp(min(0,log_target_odd-log_proposal_odd))
  u = runif(1)
  p_p*(u<=accept_prob)+p*(u>accept_prob)
}
```

The Gibbs sampler can be implemented as followed.

```
gibbs <- function(J,Pn,Nn,n1,n0,burn_frac = 0.5, keep = F, ##general para representing Gibbs
                  sd_lambda1 = 0.01,sd_lambda0 = 0.01,sd_p = 0.1, ##para for step size
                  mu_lambda_ini = 0.05,sd_lambda_ini = 0.01, ##para for initial proposal
                  mu_p_ini = 0.5,sd_p_ini = 0.1){
  nsim = ceiling(J/burn_frac)+1
  theta = array(NA,c(nsim,3))
  colnames(theta) = c('lambda1','lambda0','p')
  ##sample from the initial proposal
  para = get_invgamma_para(mu_lambda_ini,sd_lambda_ini)
  h = para[1]
  k = para[2]
  para = get_beta_para(mu_p_ini,sd_p_ini)
  a = para[1]
  b = para[2]
  theta[1,] = c(rinvgamma(2,h,k),rbeta(1,a,b))
  for (j in 2:nsim) {
    theta_use = theta[j-1,]
    ##update lambda1
    theta_use[1] = MH_lambda1(Pn,Nn,n1,n0,
                              theta_use[1],theta_use[2],theta_use[3],sd_lambda1)
    ##update lambda0
    theta_use[2] = MH_lambda0(Pn,Nn,n1,n0,
                              theta_use[1],theta_use[2],theta_use[3],sd_lambda0)
```

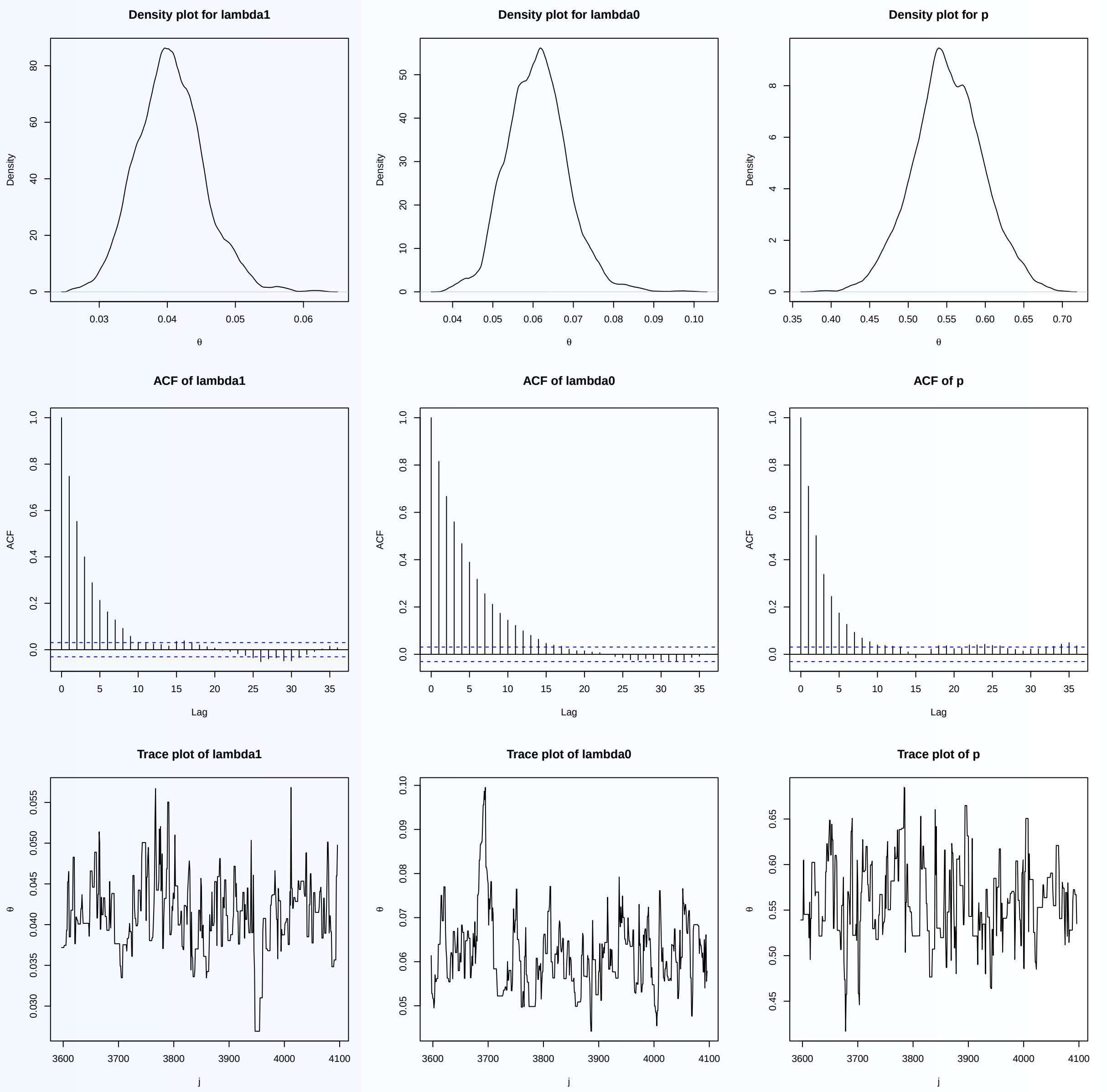
```
    ##update p
    theta_use[3] = MH_p(Pn,Nn,n1,n0,
                                  theta_use[1],theta_use[2],theta_use[3],sd_p)
    theta[j,] = theta_use
  }
  if (keep) {
    return(theta)
  }else{
    return(tail(theta,J)) ##burn the first fraction
  }
}

##sample parameters from Gibbs
set.seed(410)
J_sim = 2^12
n1 = 55
n0 = 45
Pn = 2.2
Nn = -2.7
theta_sim = gibbs(J_sim,Pn,Nn,n1,n0,keep=F)
theta_name = c('lambda1','lambda0','p')
colnames(theta_sim) = theta_name
```

After running Gibbs, some diagnostic can be performed.

```
##MCMC Diagnostic
##1) Density plot
windows(height=15,width = 15)
par(mfrow = c(3,3))
for (i in 1:3) {
  plot(density(theta_sim[,i], kernel="epanechnikov"),main = paste('Density
      plot for',theta_name[i]),
        ylab = 'Density',xlab = expression(theta))
}
##2) ACF plot
for (i in 1:3) {
  acf = acf(theta_sim[,i],plot=F)
  plot(acf,main = paste('ACF of',theta_name[i]),xlab = 'Lag')
}

##3) Trace plot
for (i in 1:3) {
  plot((J_sim-500+1):J_sim,tail(theta_sim[,i],500),type = 'l',
        main = paste('Trace plot of',theta_name[i]),ylab = expression(theta),
            xlab = 'j')
}
```



Finally, use the posterior samples to generate posterior predictive samples and estimate the call price. Note that $\max(A, 0) = A \cdot \mathbb{1}(A > 0)$. The estimated option price is \$6.00.

```
##generate posterior predictive samples
set.seed(4010)
u = rbinom(J_sim,1,theta_sim[,'p'])
r = rexp(J_sim,1/theta_sim[,'lambda1'])*u-rexp(J_sim,1/theta_sim[,'lambda0'])*
    (1-u)
##Compute option payoff
S0 = 373
K = 380
c = S0*exp(r)-K
c = c*(c>0)
mean(c)
[1] 5.99987
```