STAT440/840/CM761 Assignment # 2 SOLUTION Winter 2019 Instructor: S. Chenouri

Due: March 8th, 2019, 23:59

Instruction: Both graduate and undergraduate student must clearly mention on their submitted solution their level: "Graduate student" or "undergraduate student".

Problem 1. **[10 mark]** Consider the model given by $X \sim \text{lognormal}(0,1)$ and $\log Y = 9 + 3\log X + \epsilon$, where $\epsilon \sim N(0,1)$. We wish to estimate $\theta = E\left(\frac{Y}{X}\right)$. Compare the performance of the standard Monte Carlo estimator and the Rao-Blackwellized estimator.

SOLUTION: Note that

$$\log\left(\frac{Y}{X}\right) = 9 + \log X^2 + \epsilon \Rightarrow \frac{Y}{X} = X^2 \exp(9+\epsilon)\,.$$

Thus

$$\begin{aligned}\theta = \mathrm{E}\left[\frac{Y}{X}\right] &= \mathrm{E}\left[X^2 \exp(9+\epsilon)\right] \\ &= \mathrm{E}\left[e^9 X^2 \mathrm{E}\left[e^\epsilon \mid X\right]\right] \\ &= \mathrm{E}\left[e^9 X^2 e^{1/2}\right] = \mathrm{E}\left[e^{9.5} X^2\right]\end{aligned}$$

Standard Monte Carlo: Generate $X_i \sim \text{lognormal}(0,1)$ and $\epsilon_i \sim N(0,1)$ independently for $i = 1, \ldots, n$ and estimate θ by

$$\widehat{\theta}_{SMC} = \frac{1}{n}\sum_{i=1}^{n} X_i^2 \exp(9+\epsilon_i)\,.$$

Rao-Blackwellized Monte Carlo: Generate $X_i \sim \text{lognormal}(0,1)$ for $i = 1, \ldots, n$ and estimate θ by

$$\widehat{\theta}_{RB} = \frac{1}{n}\sum_{i=1}^{n} \exp(9.5)\, X_i^2\,.$$

The following `R` function computes these two estimates alongside other quantities for comparison.

```
A2Q1<-function(n,alpha){
epsilon<-rnorm(n,0,1)
Z<-rnorm(n,0,1)
X<-exp(Z)
deltaXe<-exp(9)*X^2*exp(epsilon)
deltaX<-exp(9.5)*X^2
Theta.hat.Xe<-mean(deltaXe)
Theta.hat.X<-mean(deltaX)
sd.deltaXe<-sd(deltaXe)
sd.deltaX<-sd(deltaX)

SE.hat.Xe<-sd.deltaXe/sqrt(n)
SE.hat.X<-sd.deltaX/sqrt(n)

LB.Xe<-Theta.hat.Xe+qnorm(alpha/2,mean=0,sd=1)*SE.hat.Xe
LB.X<-Theta.hat.X+qnorm(alpha/2,mean=0,sd=1)*SE.hat.X
```

```
UB.Xe<-Theta.hat.Xe+qnorm(1-alpha/2,mean=0,sd=1)*SE.hat.Xe
UB.X<-Theta.hat.X+qnorm(1-alpha/2,mean=0,sd=1)*SE.hat.X

  return(c(Theta.hat.Xe=Theta.hat.Xe,Theta.hat.X=Theta.hat.X,
  SE.hat.Xe=SE.hat.Xe,SE.hat.X=SE.hat.X,
  LB.Xe=LB.Xe,UB.Xe=UB.Xe,LB.X=LB.X,UB.X=UB.X))
}
```

```
> A2Q1(10000,alpha=0.05)
Theta.hat.Xe  Theta.hat.X    SE.hat.Xe     SE.hat.X        LB.Xe        UB.Xe         LB.X         UB.X
   122460.99    107573.20     21978.96     12094.19     79383.02    165538.96     83869.03    131277.37
>
> A2Q1(1000000,alpha=0.05)
Theta.hat.Xe  Theta.hat.X    SE.hat.Xe     SE.hat.X        LB.Xe        UB.Xe         LB.X         UB.X
  98861.9277   99879.1617    1046.7884     803.3743   96810.2601  100913.5953   98304.5770  101453.7463
```

From these results

$$n = 10,000, \quad \mathrm{SE}(\widehat{\theta}_{SMC}) = 21978.96 > 12094.19 = \mathrm{SE}(\widehat{\theta}_{RB})$$
$$n = 1,000,000, \quad \mathrm{SE}(\widehat{\theta}_{SMC}) = 1046.7884 > 803.3743 = \mathrm{SE}(\widehat{\theta}_{RB})\,.$$

Problem 2. [**10 mark**] The following matrix represents the transition matrix for a random walk on the integers {1, 2, 3, 4, 5}.

$$\mathbf{P} = \begin{pmatrix} 0.2 & 0.8 & 0 & 0 & 0 \\ 0.2 & 0.2 & 0.6 & 0 & 0 \\ 0 & 0.4 & 0.2 & 0.4 & 0 \\ 0 & 0 & 0.6 & 0.2 & 0.2 \\ 0 & 0 & 0 & 0.8 & 0.2 \end{pmatrix}$$

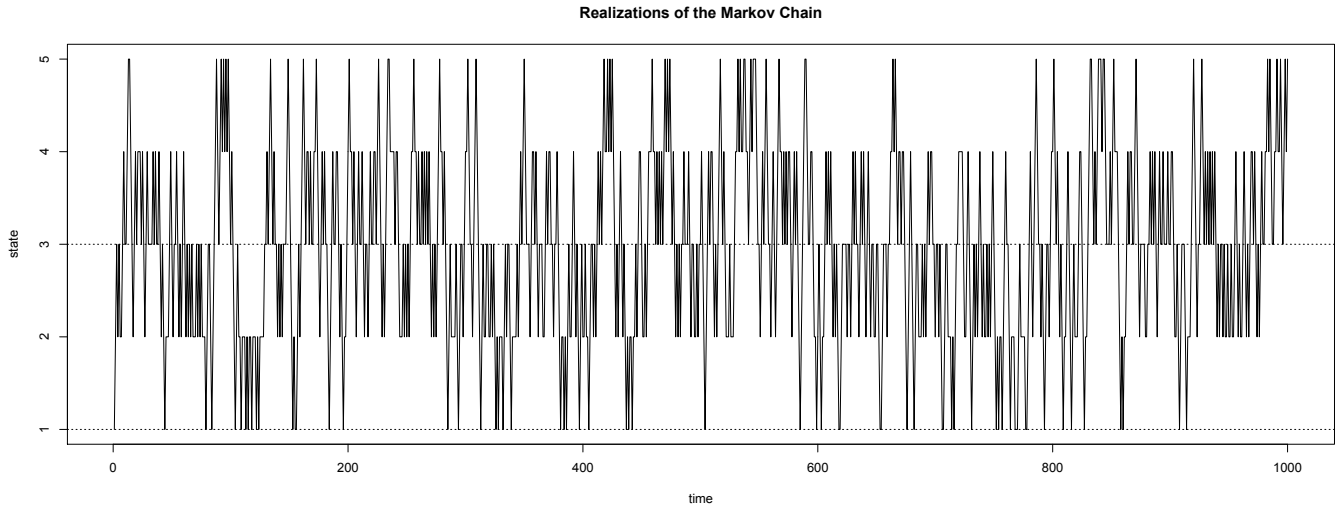
a) [**4 mark**] Suppose one starts at the state (location) 1. Simulate 1000 steps of the Markov chain using the probabilities given in the transition matrix. Store the states of the walk in a vector. You may use a plot to visualize the sample path.

SOLUTION: The following `R` function `MC.sample`, simulate `n.step` realization of the a Markov chain with transition matrix `trans.matrix` and the initial state `initial`.

```
MC.sample<-function(n.step,initial,trans.matrix){
n.states<-nrow(trans.matrix)
states<-numeric(n.step)
states[1]<-initial
for (t in 2:n.step)
{
# probability function to generate next state states[t,]
pr<-trans.matrix[states[t-1],]
# use a multinomial dist with prob. pr to generate next state states[t,]
states[t]<-which(rmultinom(1,1,pr)==1)
}
return(states)
}
```

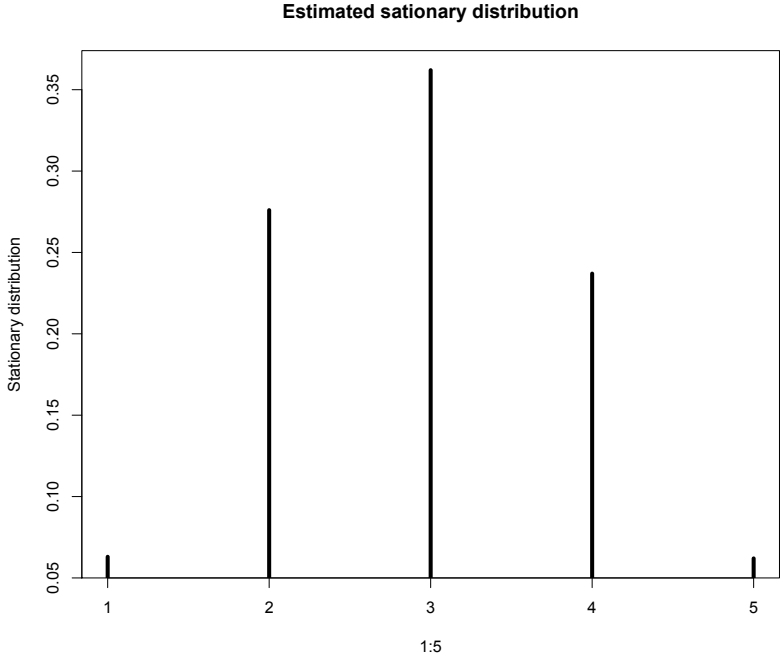
We run this function with `n.step`=1000, and the given transition matrix P and the initial value 1. The following time series plot provide the sample path or the realizations.

```
p<-c(0.2,0.8,0,0,0,0.2,0.2,0.6,0,0,0,0.4,0.2,0.4,0,0,0,0.6,0.2,0.2,0,0,0,0.8,0.2)
# One-step transition matrix
P<-matrix(p,nrow=5,ncol=5,byrow=TRUE)
# Simulating 1000 realizations of the Markov chain with transition matrix P
X<-MC.sample(1000,1,P)

# Plot these realizations vs time
matplot(X,type="l",col=1:5,ylim=c(1,5),ylab="state",xlab="time",
main="1000 Realizations of the Markov Chain")
abline(h=1,lty=3)
abline(h=3,lty=3)
```



b) **[3 mark]**Compute the relative frequencies of the walker in the five states from the simulation output. Guess at the value of the stationary distribution vector π.
SOLUTION:

```
> table(X)/1000
X
    1     2     3     4     5
0.063 0.276 0.362 0.237 0.062

PI<-c(0.063,0.276,0.362,0.237,0.062)
plot(1:5,PI,type="h",ylab="Stationary distribution",lwd=4,main="Estimated stationary distribution")
```



This suggest that $\widehat{\boldsymbol{\pi}} = (0.063, 0.276, 0.362, 0.237, 0.062)^T$.

c) [**3 mark**] Confirm that your guess is indeed the stationary distribution by the matrix computation $\mathbf{P}^T\,\boldsymbol{\pi}$.
SOLUTION:

```
PI<-matrix(c(0.063,0.276,0.362,0.237,0.062),nrow=5,ncol=1)
t(t(P)%*%PI)
       [,1]   [,2]   [,3]   [,4]   [,5]
[1,] 0.0678 0.2504 0.3802 0.2418 0.0598
# which is close to the values in vector PI
```

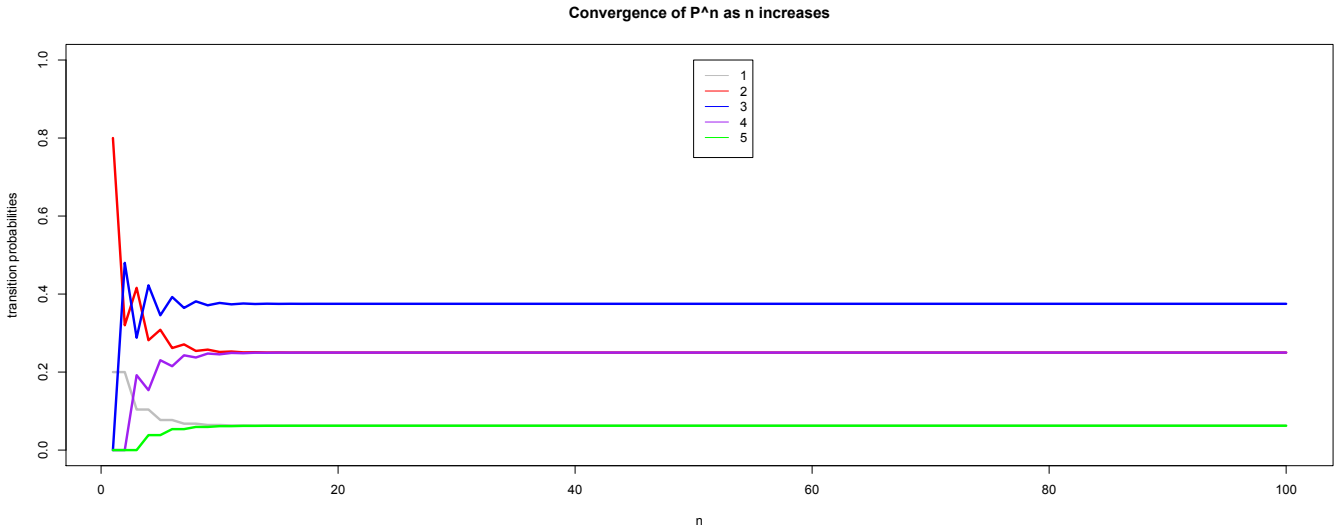
In addition, you may obtain $\boldsymbol{\pi}$ by looking at the limit $\lim_{n\to\infty} P^n$. The following `R` codes looks at P^{1000}.

```
## The folllowing function computes n.step transition matrix P
## starting with initial state and a given 1-step transition matrix
## It use the package ‘‘expm" to compute nth power of trans.matrix
library(expm)
LongRun<-function(n.step,initial,trans.matrix){
Pn<-trans.matrix%^%n.step
return(Pn)
}

> LongRun(1000,1,P)
       [,1] [,2]  [,3] [,4]   [,5]
[1,] 0.0625 0.25 0.375 0.25 0.0625
[2,] 0.0625 0.25 0.375 0.25 0.0625
[3,] 0.0625 0.25 0.375 0.25 0.0625
```

```
[4,] 0.0625 0.25 0.375 0.25 0.0625
[5,] 0.0625 0.25 0.375 0.25 0.0625
```

We also plot p_{ij}^n for $n = 1, \ldots, 100$ for $j = 1, 2, 3, 4, 5$. We see that p_{ij}^n converges to $\pi_j, \quad j = 1, 2, 3, 4, 5$ very fast as p_{ij}^n stabilizes roughly after $n = 10$.

```
plot(P1.n[,1],ylim=c(0,1),xlab="n",ylab="transition probabilities",
col="grey", type="l", lwd=3,main="Convergence of P^n as n increases")
points(P1.n[,2],type="l",col="red", lwd=3)
points(P1.n[,3],type="l",col="blue", lwd=3)
points(P1.n[,4],type="l",col="purple", lwd=3)
points(P1.n[,5],type="l",col="green", lwd=3)
legend(x=c(50,55),y=c(0.75,1),legend=1:5,col=c("grey","red","blue","purple","green"),
lty=c(1,1,1,1,1))
```



To solve the equation $\mathbf{P}^T \boldsymbol{\pi} = \boldsymbol{\pi}$, we recall that it reminds us about the eigenvalue-vector equation $\mathbf{A}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$ with $\lambda = 1$ and $\mathbf{A} = \mathbf{P}^T$ and $\boldsymbol{\alpha} = \boldsymbol{\pi}$. This means that the stationary distribution π is the eigenvector of $\mathbf{P}^T$ corresponding to the eigenvalue $\lambda = 1$. If there are multiple eigenvectors associated with the eigenvalue $\lambda = 1$, then all will be considered stationary distributions. This can happen only when the Markov chain is reducible.

```
## The following function calculates the exact long run equilibrium
## frequencies for a given transition matrix

equilib.dist<-function(trans.matrix)
{
eigen.vec <- eigen(t(trans.matrix))$vectors[,1]
eigen.vec <- eigen.vec/sum(eigen.vec)
    as.double(eigen.vec)
}

equilib.dist(P)
[1] 0.0625 0.2500 0.3750 0.2500 0.0625
```

Problem 3. **[25 mark]** The following twelve observations are from a simulated reliability study:

0.56, 2.26, 1.90, 0.94, 1.40, 1.39, 1.00, 1.45, 2.32, 2.08, 0.89, 1.68.

A Weibull model with the following density form is considered appropriate with parameters $(\alpha,\, \eta)$

$$f(x \,|\, \alpha, \eta) \propto \alpha\, \eta\, x^{\alpha-1}\, e^{-\eta\, x^{\alpha}}\,, \quad 0 < x < \infty\,.$$

Consider the prior distribution $\pi(\alpha,\, \eta) \propto e^{-\alpha}\eta^{\beta-1}e^{-c\,\eta}$

a) **[5 mark]** Write down the posterior distribution of $(\alpha,\, \eta)$ given the data $x_1,\, x_2,\, \ldots,\, x_n$.
SOLUTION: For the observed data vector $\mathbf{x} = (x_1,\, \ldots,\, x_n)$, the posterior pd.f. is

$$\begin{aligned}
\pi(\alpha,\, \eta \mid \mathbf{x}) &\propto \pi(\alpha,\, \eta)\, f(\mathbf{x} \,|\, \alpha, \eta) = \pi(\alpha,\, \eta) \prod_{i=1}^{n} f(x_i \,|\, \alpha, \eta) \\
&= e^{-\alpha}\eta^{\beta-1}e^{-c\,\eta}\, \alpha^n\, \eta^n \left(\prod_{i=1}^{n} x_i\right)^{\alpha-1} e^{-\eta \sum_{i=1}^{n} x_i^{\alpha}} \\
&= \alpha^n\, e^{-\alpha} \left(\prod_{i=1}^{n} x_i\right)^{\alpha-1} \eta^{n+\beta-1}\, \exp\left(-\left[c + \sum_{i=1}^{n} x_i^{\alpha}\right]\, \eta\right)
\end{aligned}$$

That is the posterior p.d.f $\pi(\alpha,\, \eta \mid \mathbf{x})$ up to a proportionality constant is given by

$$\pi(\alpha,\, \eta \mid \mathbf{x}) \propto\, \alpha^n\, e^{-\alpha} \left(\prod_{i=1}^{n} x_i\right)^{\alpha-1} \eta^{n+\beta-1}\, \exp\left(-\left[c + \sum_{i=1}^{n} x_i^{\alpha}\right]\, \eta\right)\,.$$

b) **[5 mark]** To get a sample from the posterior density, one may use the Metropolis-Hastings algorithm with proposal density

$$q(\alpha',\, \eta' \,|\, \alpha,\, \eta) = \frac{1}{\alpha\, \eta} \exp\left\{-\frac{\alpha'}{\alpha} - \frac{\eta'}{\eta}\right\}\,,$$

which is a product of two independent exponential distributions with means $\alpha,\, \eta$. Compute the acceptance probability $\rho(\alpha',\, \eta' \,|\, \alpha^{(t)},\, \eta^{(t)})$ at the tth step of the Metropolis-Hastings chain. What is this distribution?

SOLUTION: The acceptance probability of a new candidate sample point $(\alpha',\, \eta')$ given the observation $(\alpha^{(t)},\, \eta^{(t)})$ in the previous iteration is

$$\begin{aligned}
\rho(\alpha',\, \eta' \,|\, \alpha^{(t)},\, \eta^{(t)}) &= \min\left\{\frac{\pi(\alpha',\, \eta' \mid \mathbf{x})\, q(\alpha^{(t)},\, \eta^{(t)} \,|\, \alpha',\, \eta')}{\pi(\alpha^{(t)},\, \eta^{(t)} \mid \mathbf{x})\, q(\alpha',\, \eta' \,|\, \alpha^{(t)},\, \eta^{(t)})}\,, 1\right\} \\
&= \min\left\{\frac{(\alpha')^n\, e^{-\alpha'} \left(\prod_{i=1}^{n} x_i\right)^{\alpha'-1} (\eta')^{n+\beta-1}\, \exp\left(-\left[c + \sum_{i=1}^{n} x_i^{\alpha'}\right]\, \eta'\right) \frac{1}{\alpha'\,\eta'} \exp\left(-\frac{\alpha^{(t)}}{\alpha'} - \frac{\eta^{(t)}}{\eta'}\right)}{(\alpha^{(t)})^n\, e^{-\alpha^{(t)}} \left(\prod_{i=1}^{n} x_i\right)^{\alpha^{(t)}-1} (\eta^{(t)})^{n+\beta-1} \exp\left(-\left[c + \sum_{i=1}^{n} x_i^{\alpha^{(t)}}\right]\, \eta^{(t)}\right) \frac{1}{\alpha^{(t)}\,\eta^{(t)}} \exp\left(-\frac{\alpha'}{\alpha^{(t)}} - \frac{\eta'}{\eta^{(t)}}\right)}\,, 1\right\} \\
&= \min\left\{\left(\frac{\alpha'}{\alpha^{(t)}}\right)^{n-1} \left(\prod_{i=1}^{n} x_i\right)^{\alpha'-\alpha^{(t)}} \left(\frac{\eta'}{\eta^{(t)}}\right)^{n+\beta-2} \cdot \frac{\exp\left(-\alpha'\left[1 - \frac{1}{\alpha^{(t)}}\right]\right)}{\exp\left(-\alpha^{(t)}\left[1 - \frac{1}{\alpha'}\right]\right)} \cdot \frac{\exp\left(-\left[c + \sum_{i=1}^{n} x_i^{\alpha'} - \frac{1}{\eta^{(t)}}\right]\, \eta'\right)}{\exp\left(-\left[c + \sum_{i=1}^{n} x_i^{\alpha^{(t)}} - \frac{1}{\eta'}\right]\, \eta^{(t)}\right)}\,, 1\right\}
\end{aligned}$$

c) **[5 mark]** Explain how the chain is to be generated. Implement the algorithm.

SOLUTION: Chose a starting value for the hyper-parameters β and c. Run the following algorithm and study the convergence and efficiency (speed) of the algorithm and decide about on good values for the hyper-parameters β and c after some trial and error experiment or monitoring the acceptance probabilities for different combinations of β and c.

1. For iteration $t = 0$ start with the initial values $(\alpha^{(0)}, \eta^{(0)})$.
2. In iteration $t + 1$,
   i. Given $(\alpha^{(t)}, \eta^{(t)})$, generate $\alpha' \sim \text{Exp}(\alpha^{(t)})$ and $\eta' \sim \text{Exp}(\eta^{(t)})$ and $u \sim U(0, 1)$.
   ii. Compute the acceptance probability $\rho(\alpha', \eta' \,|\, \alpha^{(t)}, \eta^{(t)})$.
   iii. If $u \leq \rho(\alpha', \eta' \,|\, \alpha^{(t)}, \eta^{(t)})$, set $(\alpha^{(t+1)}, \eta^{(t+1)}) = (\alpha', \eta')$, else set $(\alpha^{(t+1)}, \eta^{(t+1)}) = (\alpha^{(t)}, \eta^{(t)})$.
3. Set $t \leftarrow t + 1$, goto step 2.
4. Return $\{(\alpha^{(1)}, \eta^{(1)}), \ldots, (\alpha^{(N)}, \eta^{(N)})\}$. Evaluate the sample for convergence using the time series plot, histogram, etc to and select a burn-in n. Use the sample $\{(\alpha^{(n+1)}, \eta^{(n+1)}), \ldots, (\alpha^{(N)}, \eta^{(N)})\}$ for further computations.

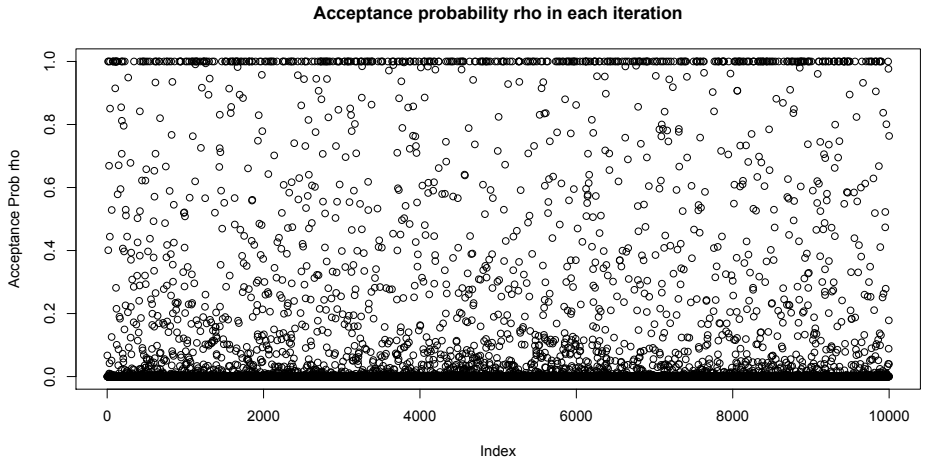
```
A2Q3.MH<-function(x,N,ini,beta,c){
n<-length(x)
alpha<-ini[1]
eta<-ini[2]
t<-1
rho<-NA
Test<-NA
while(t<=N){
# generate new points
alpha.prim<-rexp(1,rate=1/alpha[t])
eta.prim<-rexp(1,rate=1/eta[t])
u<-runif(1,min=0,max=1)
# Compute acceptance prob.
A<-(alpha.prim/alpha[t])^n
B<-prod(x)^(alpha.prim-alpha[t])
C<-(eta.prim/eta[t])^(n+beta-2)
D<-exp(-alpha.prim*(1-1/alpha[t]))/exp(-alpha[t]*(1-1/alpha.prim))
E<-exp(-(c+sum(x^alpha.prim)-1/eta[t])*eta.prim)/exp(-(c+sum(x^alpha[t])-1/eta.prim)*eta[t])
rho[t]<-min(c(A*B*C*D*E,1))
# Test the new points
    if (u<=rho[t]){
Test[t]<-1 #Accept
alpha[t+1]<-alpha.prim
eta[t+1]<-eta.prim
}
else{
Test[t]<-0 # Reject
alpha[t+1]<-alpha[t]
eta[t+1]<-eta[t]
}
# set t to t+1 goto step 2.
t<-t+1
}
# return samples and the acceptance probabilities in every iterations
alpha.eta.sample<-cbind(alpha,eta)
```

```
return(list(alpha.eta.sample,rho,Test))
}

# Here are acceptance rate for a few values for beta and c
> OUT<-A2Q3.MH(x,10000,c(1,1),1,0.01)
> mean(OUT[[3]])
[1] 0.0829

> OUT<-A2Q3.MH(x,10000,c(1,1),1,4)
> mean(OUT[[3]])
[1] 0.0815

> OUT<-A2Q3.MH(x,10000,c(1,1),0.1,4)
> mean(OUT[[3]])
[1] 0.0779

> OUT<-A2Q3.MH(x,10000,c(1,1),0.1,10)
> mean(OUT[[3]])
[1] 0.0665

> OUT<-A2Q3.MH(x,10000,c(1,1),0.1,0.1)
> mean(OUT[[3]])
[1] 0.0738

> OUT<-A2Q3.MH(x,10000,c(1,1),4,0.1)
> mean(OUT[[3]])
[1] 0.0799

> OUT<-A2Q3.MH(x,10000,c(1,1),10,0.1)
> mean(OUT[[3]])
[1] 0.0716
```

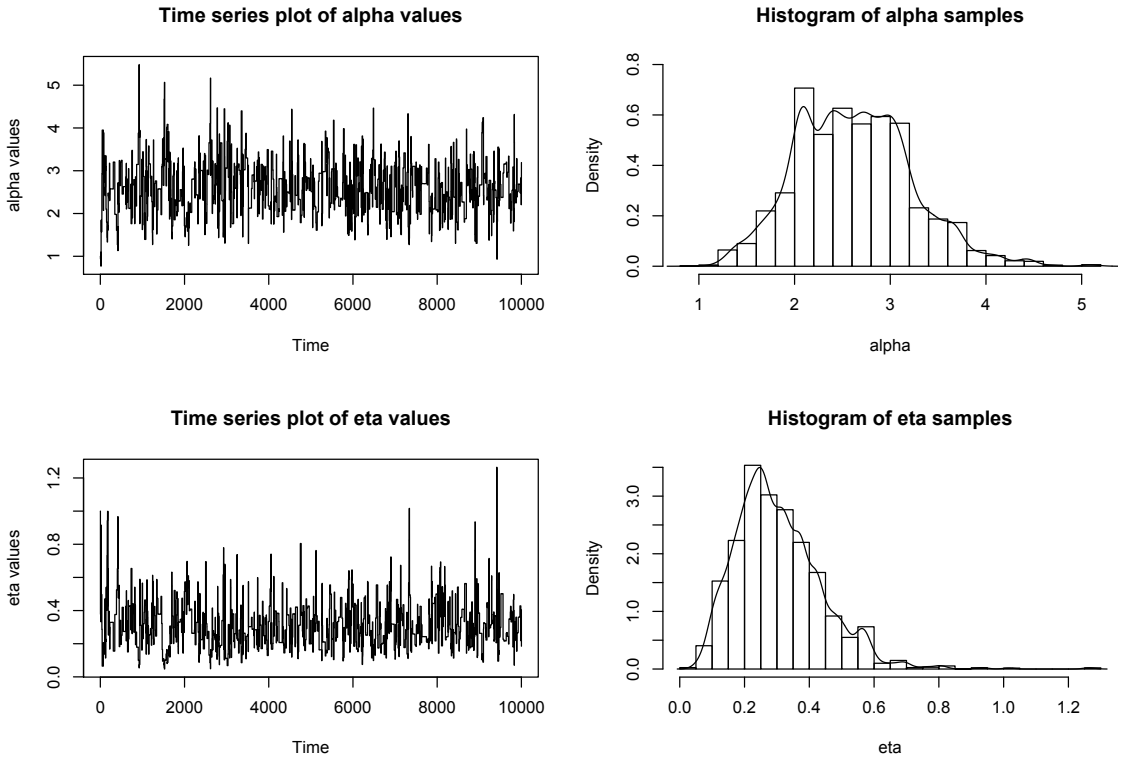
Looks like the Metropolis-Hastings algorithm with the suggested proposal has a very small acceptance rate. I examined several β and c values and rate are not too different.

```
> OUT<-A2Q3.MH(x,10000,c(1,1),1,0.01)
> mean(OUT[[3]])
[1] 0.083

plot(OUT[[2]],ylab="Acceptance Prob rho", main="Acceptance probability rho in each iteration")

par(mfrow=c(2,2))
ts.plot(OUT[[1]][,1],ylab="alpha values",main="Time series plot of alpha values")
hist(OUT[[1]][-c(1:1000),1],breaks=20,freq=FALSE,ylim=c(0,0.8),
main="Histogram of alpha samples",xlab="alpha")
lines(density(OUT[[1]][-c(1:1000),1]))
```

```
ts.plot(OUT[[1]][,2],ylab="eta values",main="Time series plot of eta values")
hist(OUT[[1]][-c(1:1000),2],breaks=20,freq=FALSE,ylim=c(0,3.5),
main="Histogram of eta samples",xlab="eta")
lines(density(OUT[[1]][-c(1:1000),2]))
```



The majority of acceptance probabilities over iterations are very small. The time series plots of the generated $\alpha^{(t)}$ and $\eta^{(t)}$ are given below.



d) **[5 mark]** Use part (c) to approximate the Bayes estimates of α, and η under the squared error loss (i.e. posterior means of α, and η).

SOLUTION: We use the burn-in $n = 1000$ and obtain the Bayes estimates of α and η using the remaining $N - n = 9000$ realizations of the Markov chains.

```
# Bayes estimate of alpha
mean(OUT[[1]][,1])
[1] 2.61806

# Bayes estimate of eta
mean(OUT[[1]][,2])
[1] 0.3093825
```

So the Bayes estimates under the squared error loss as the posterior mean are $\widehat{\alpha} = 2.61806$ and $\widehat{\eta} = 0.3093825$.

e) [**5 mark**] Obtain the posterior modes and 95% quantile-based confidence intervals for α, and η. Show your steps, or your R codes.

SOLUTION:

```
# The following function estimate the mode of a distribution based on a dataset x
MCmode<-function(x){
d<-density(x)
return(d$x[which(d$y==max(d$y)[1])])
}

# posterior mode estimate for alpha
MCmode(OUT[[1]][-c(1:1000),1])
[1] 2.093988

# posterior mode estimate for eta
MCmode(OUT[[1]][-c(1:1000),2])
[1] 0.2467673

# 95% confidence interval for alpha
> quantile(OUT[[1]][-c(1:1000),1],  probs=c(0.025,0.975))
    2.5%    97.5%
1.556087 3.913722

# 95% confidence interval for eta

quantile(OUT[[1]][-c(1:1000),2],  probs=c(0.025,0.975))
     2.5%     97.5%
0.1026983 0.5877482
```

Problem 4. [**15 mark**] Suppose X and Y are two random variables having the following joint density function

$$\pi(x, y) \propto x^2 \exp\left(-xy^2 - y^2 + 2y - 4x\right), \quad x > 0, \ -\infty < y < \infty,$$

a) **[6 mark]** Identify the full conditionals of the bivariate density $\pi(x, y)$.
SOLUTION: We start with noting that for both conditional distributions

$$\pi(x \mid y) \propto \pi(x, y) \quad \text{and} \quad \pi(y \mid x) \propto \pi(x, y)\,.$$

For $\pi(x \mid y)$ we have

$$\begin{aligned}\pi(x \mid y) &\propto x^2 \exp\left(-xy^2 - y^2 + 2y - 4x\right) \\ &\propto x^2 \exp\left(-xy^2 - 4x\right) \\ &= x^2 \exp\left(-(y^2+4)x\right)\,.\end{aligned}$$

From this we observe that the conditional distribution of X given Y is Gamma($\alpha = 3$, $\beta = ((y^2+4)^{-1})$, that is

$$\pi(x \mid y) = \frac{(y^2+4)^3}{\Gamma(3)}\, x^2\, \exp\left(-(y^2+4)x\right), \quad x > 0,\ -\infty < y < \infty\,.$$

For $\pi(y \mid x)$ we have

$$\begin{aligned}\pi(y \mid x) &\propto x^2 \exp\left(-xy^2 - y^2 + 2y - 4x\right) \\ &\propto \exp\left(-(x+1)\left[y^2 - 2\frac{1}{x+1}\, y\right]\right) \\ &\propto \exp\left(-(x+1)\left[y^2 - 2\frac{1}{x+1}\, y + \frac{1}{(x+1)^2}\right]\right) \\ &= \exp\left(-(x+1)\left[y - \frac{1}{(x+1)}\right]^2\right)\end{aligned}$$

From this we observe that the conditional distribution of Y given x is $\mathrm{N}\left(\frac{1}{(x+1)},\ \frac{1}{2\,(x+1)}\right)$, that is

$$\pi(y \mid x) = \sqrt{\frac{x+1}{\pi}}\, \exp\left(-(x+1)\left[y - \frac{1}{(x+1)}\right]^2\right), \quad x > 0,\ -\infty < y < \infty\,.$$

b) **[3 mark]** Construct a Gibbs sampler which has this stationary distribution $\pi(x, y)$.
SOLUTION:

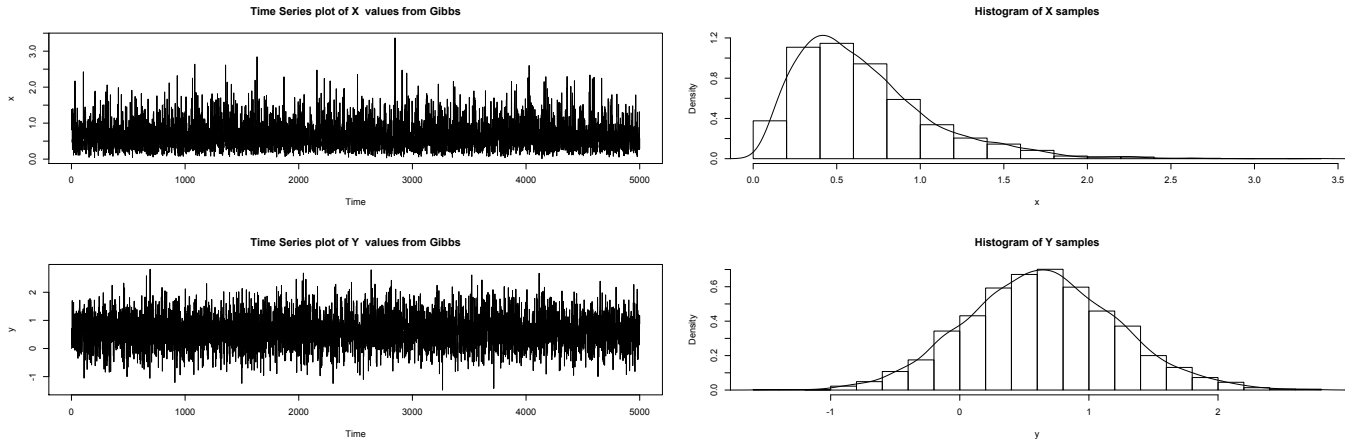
1) Initialize for $n = 0$ with $(x_{(0)},\, y_{(0)})$ such that $x_{(0)} > 0$ and $\infty < y_{(0)} < \infty$.
2) For $n \geq 1$, generate $x_{(n)}$ from Gamma($\alpha = 3$, $\beta = ((y^2_{(n-1)} + 4)^{-1})$
3) Generate $y_{(n)}$ from Beta($x_{(n)} + \alpha$, $m - x_{(n)} + \beta$),
4) Return $\{(x_{(0)}, y_{(0)}), \ldots, (x_{(N)}, y_{(N)})\}$.

For some $n < N$, we discard $\{(x_{(0)}, y_{(0)}), \ldots, (x_{(n)}, y_{(n)})\}$, and consider $\{(x_{(n+1)},\, y_{(n+1)}), \ldots, (x_{(N)}, y_{(N)})\}$ as a sample from $\pi(x, y)$ which does not depend on the initial pair $(x_{(0)},\, y_{(0)})$.

c) **[3 mark]** Implement your Gibbs sampler in part (b). Monitor the convergence.
SOLUTION:

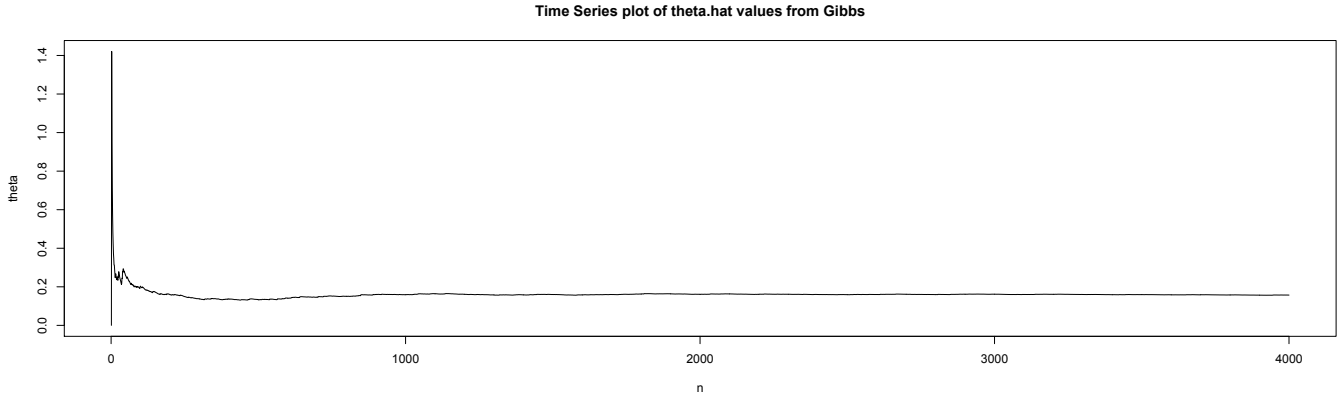
```
A1Q4.Gibbs<-function(N,x0,y0){
X<-c(x0)
Y<-c(y0)
for ( i in 2:N){
```

```
X[i]<-rgamma(1,shape=3,scale=1/(Y[i-1]^2+4))
Y[i]<-rnorm(1,mean=1/(X[i]+1),sd=1/sqrt(2*(X[i]+1)))
}
return(cbind(X,Y))
}
# Generate a sample of size 5000
Samp.XY<-A1Q4.Gibbs(5000,1,0)

# The time series plots of X and Y and their histograms
# with kernel density estimates overlaid.
par(mfrow=c(2,2))
ts.plot(Samp.XY[,1],ylab="x",main="Time Series plot of X  values from Gibbs")
hist(Samp.XY[-c(1:1000),1],breaks=20,freq=FALSE,ylim=c(0,1.2),main="Histogram of X samples",xlab="x")
lines(density(Samp.XY[-c(1:1000),1]))

ts.plot(Samp.XY[,2],ylab="y",main="Time Series plot of Y  values from Gibbs")
hist(Samp.XY[-c(1:1000),2],breaks=20,freq=FALSE,main="Histogram of Y samples",xlab="y")
lines(density(Samp.XY[-c(1:1000),2]))
```



d) **[3 mark]** Estimate $E_\pi\left[X^2\,Y^3\exp(-X^2)\right]$.
SOLUTION:

```
Delta<-Samp.XY[-c(1:1000),1]^2*Samp.XY[-c(1:1000),2]^3*exp(-Samp.XY[-c(1:1000),1]^2)

theta.hat<-mean(Delta)

> theta.hat
[1] 0.1571514
```



Problem 5. **[15 mark]** A random variable Z has an *inverse Gaussian distribution* if it has density

$$f_Z(z \,|\, \theta_1,\, \theta_2) \propto z^{-3/2} \exp\left(-\theta_1\, z - \frac{\theta_2}{z} + 2\sqrt{\theta_1\, \theta_2} + \log(\sqrt{2\,\theta_2})\right), \qquad z > 0$$

where $\theta_1 > 0$ and $\theta_2 > 0$ are parameters. It can be shown that

$$E(Z) = \sqrt{\frac{\theta_2}{\theta_1}} \quad \text{and} \quad E\left(\frac{1}{Z}\right) = \sqrt{\frac{\theta_1}{\theta_2}} + \frac{1}{2\theta_2}\,.$$

a) **[8 mark]** Let $\theta_1 = 1.5$ and $\theta_2 = 2$. Draw a sample of size 1000 using the independent-Metropolis-Hastings algorithm. Use a Gamma distribution as the proposal density. To assess the accuracy, compare the mean of Z and $1/Z$ from the sample to the theoretical means. Try different Gamma distributions to see if you can get an accurate sample.

SOLUTION: Let the proposal distribution be Gamma(α, β) with p.d.f.

$$g(z) = \frac{1}{\Gamma(\alpha)\,\beta^\alpha} z^{\alpha-1} \exp\left(-\frac{z}{\beta}\right) \quad z > 0\,.$$

Thus the acceptance probability is

$$\alpha(z,\, z') = \min\left\{\frac{f_Z(z' \,|\, \theta_1,\, \theta_2)\, g(z)}{f_Z(z \,|\, \theta_1,\, \theta_2)\, g(z')},\ 1\right\}.$$

```
fz<-function(z,theta1,theta2){
d<-(z^(-3/2))*exp(-theta1*z-theta2/z+2*sqrt(theta1*theta2)+log(sqrt(2*theta2)))
return(d)
}
###############
A2Q3.IMH<-function(N,z0,theta1,theta2,alpha,beta){
Z<-z0
t<-1
accept.prob<-NA
Test<-NA
while(t<=N){
# generate new points
z.prim<-rgamma(1,shape=alpha,scale=beta)
```

```
u<-runif(1,min=0,max=1)

# Compute acceptance prob.
accept.prob[t]<-min((fz(z.prim,theta1,theta2)*dgamma(Z[t],shape=alpha,scale=beta))/
(fz(Z[t],theta1,theta2)*dgamma(z.prim,shape=alpha,scale=beta)),1)

# Test the new points
    if (u<=accept.prob[t]){
Test[t]<-1 # Accept
Z[t+1]<-z.prim
}
else{
Test[t]<-0 # Reject
Z[t+1]<-Z[t]
}
# set t to t+1 goto step 2.
t<-t+1
}
# return samples and the acceptance probabilities in every iterations
return(list(Z,accept.prob,Test))
}

out1<-A2Q3.IMH(2000,2,1.5,2,1,1.5)
m1<-mean(out1[[1]][-c(1:1000)])
inv.m1<-mean(1/out1[[1]][-c(1:1000)])

out2<-A2Q3.IMH(2000,2,1.5,2,1,3)
m2<-mean(out2[[1]][-c(1:1000)])
inv.m2<-mean(1/out2[[1]][-c(1:1000)])

out3<-A2Q3.IMH(2000,2,1.5,2,1,5)
m3<-mean(out3[[1]][-c(1:1000)])
inv.m3<-mean(1/out3[[1]][-c(1:1000)])

out4<-A2Q3.IMH(2000,2,1.5,2,1.5,1.5)
m4<-mean(out4[[1]][-c(1:1000)])
inv.m4<-mean(1/out4[[1]][-c(1:1000)])

out5<-A2Q3.IMH(2000,2,1.5,2,1.5,3)
m5<-mean(out5[[1]][-c(1:1000)])
inv.m5<-mean(1/out5[[1]][-c(1:1000)])

out6<-A2Q3.IMH(2000,2,1.5,2,1.5,5)
m6<-mean(out6[[1]][-c(1:1000)])
inv.m6<-mean(1/out6[[1]][-c(1:1000)])
```

```
theta1<-1.5
theta2<-2
mtrue<-rep(sqrt(theta2/theta1),6)
inv.mtrue<-rep(sqrt(theta1/theta2)+1/(2*theta2),6)
alpha<-c(1,1,1,1.5,1.5,1.5)
beta<-c(1.5,3,5,1.5,3,1.5)
m<-c(m1,m2,m3,m4,m5,m6)
inv.m<-c(inv.m1,inv.m2,inv.m3,inv.m4,inv.m5,inv.m6)
dif.m<-m-mtrue
dif.inv.m<-inv.m-inv.mtrue
cbind(alpha,beta,mtrue,m,inv.mtrue,inv.m,dif.m,dif.inv.m)
```

```
     alpha beta    mtrue        m inv.mtrue    inv.m        dif.m    dif.inv.m
[1,]   1.0  1.5 1.154701 1.135511  1.116025 1.143813 -0.019189360  0.027787214
[2,]   1.0  3.0 1.154701 1.213390  1.116025 1.113584  0.058689133 -0.002441547
[3,]   1.0  5.0 1.154701 1.157939  1.116025 1.086060  0.003238925 -0.029965682
[4,]   1.5  1.5 1.154701 1.147491  1.116025 1.102820 -0.007209740 -0.013205596
[5,]   1.5  3.0 1.154701 1.069105  1.116025 1.158519 -0.085595369  0.042493611
[6,]   1.5  1.5 1.154701 1.075951  1.116025 1.180788 -0.078749308  0.064762177
```

Note that for $\theta_1 = 1.5$ and $\theta_2 = 2$ the theoretical means are

$$E(Z) = \sqrt{\frac{\theta_2}{\theta_1}} = 1.154701 \qquad \text{and} \qquad E\left(\frac{1}{Z}\right) = \sqrt{\frac{\theta_1}{\theta_2}} + \frac{1}{2\theta_2} = 1.199359\,.$$

b) **[7 mark]** Draw a sample of size 1000 using the random walk-Metropolis-Hastings algorithm. Since $z > 0$ we cannot just use a Normal density. One strategy is this: let $W = \log(Z)$, find the density of W. Use the random walk-Metropolis-Hastings algorithm to get a sample $W_1, \ldots, W_N$ and let $Z_i = e^{W_i}$. Assess the accuracy of the simulation as in part (a).

SOLUTION: For $w = \log(z) \in \mathbb{R}$, $z = e^w$ and the Jacobian of the transformation is $\frac{dz}{dw} = e^w$.

$$\begin{aligned} f_{_W}(w \,|\, \theta_1, \theta_2) &= f_{_Z}(e^w \,|\, \theta_1, \theta_2) \mid J \mid \\ &\propto \exp\left(-\frac{3}{2}w\right) \exp\left(-\theta_1\, e^w - \theta_2 e^{-w} + 2\sqrt{\theta_1\,\theta_2} + \log(\sqrt{2\,\theta_2})\right) \exp(w) \\ &= \exp\left(-\frac{1}{2}w\right) \exp\left(-\theta_1\, e^w - \theta_2 e^{-w} + 2\sqrt{\theta_1\,\theta_2} + \log(\sqrt{2\,\theta_2})\right) \\ &= \exp\left(-\frac{1}{2}w - \theta_1\, e^w - \theta_2 e^{-w} + 2\sqrt{\theta_1\,\theta_2} + \log(\sqrt{2\,\theta_2})\right), \quad w \in \mathbb{R}\,. \end{aligned}$$

```
##########
##########
# Part b)
fw<-function(w,theta1,theta2){
d<-exp(-0.5*w-theta1*exp(w)-theta2*exp(-w)+2*sqrt(theta1*theta2)+log(sqrt(2*theta2)))
return(d)
}
A2Q3.RW.MH<-function(N,w0,theta1,theta2,sig){
```

```
W<-w0
t<-1
accept.prob<-NA
Test<-NA
while(t<=N){
# generate new points
e.prim<-rnorm(1,mean=0,sd=sig)
w.prim<-W[t]+e.prim
u<-runif(1,min=0,max=1)

# Compute acceptance prob.
accept.prob[t]<-min(fw(w.prim,theta1,theta2)/
fw(W[t],theta1,theta2),1)

# Test the new points
    if (u<=accept.prob[t]){
Test[t]<-1 # Accept
W[t+1]<-w.prim
}
else{
Test[t]<-0 # Reject
W[t+1]<-W[t]
}
# set t to t+1 goto step 2.
t<-t+1
}
Z<-exp(W)
# return samples and the acceptance probabilities in every iterations
return(list(Z,accept.prob,Test))
}

outw1<-A2Q3.RW.MH(2000,2,1.5,2,0.1)
mw1<-mean(outw1[[1]][-c(1:1000)])
inv.mw1<-mean(1/outw1[[1]][-c(1:1000)])

outw2<-A2Q3.RW.MH(2000,2,1.5,2,0.5)
mw2<-mean(outw2[[1]][-c(1:1000)])
inv.mw2<-mean(1/outw2[[1]][-c(1:1000)])

outw3<-A2Q3.RW.MH(2000,2,1.5,2,1)
mw3<-mean(outw3[[1]][-c(1:1000)])
inv.mw3<-mean(1/outw3[[1]][-c(1:1000)])

outw4<-A2Q3.RW.MH(2000,2,1.5,2,2)
mw4<-mean(outw4[[1]][-c(1:1000)])
```

```
inv.mw4<-mean(1/outw4[[1]][-c(1:1000)])

outw5<-A2Q3.RW.MH(2000,2,1.5,2,4)
mw5<-mean(outw5[[1]][-c(1:1000)])
inv.mw5<-mean(1/outw5[[1]][-c(1:1000)])

outw6<-A2Q3.RW.MH(2000,2,1.5,2,6)
mw6<-mean(outw6[[1]][-c(1:1000)])
inv.mw6<-mean(1/outw6[[1]][-c(1:1000)])

theta1<-1.5
theta2<-2
mtrue<-rep(sqrt(theta2/theta1),6)
inv.mtrue<-rep(sqrt(theta1/theta2)+1/(2*theta2),6)
sig<-c(0.1,0.5,1,2,4,6)
mw<-c(mw1,mw2,mw3,mw4,mw5,mw6)
inv.mw<-c(inv.mw1,inv.mw2,inv.mw3,inv.mw4,inv.mw5,inv.mw6)
dif.mw<-m-mtrue
dif.inv.mw<-inv.mw-inv.mtrue
cbind(sig,mtrue,mw,inv.mtrue,inv.mw,dif.mw,dif.inv.mw)

     sig    mtrue       mw inv.mtrue   inv.mw       dif.mw  dif.inv.mw
[1,] 0.1 1.154701 0.971069  1.116025 1.295893 -0.019189360  0.17986741
[2,] 0.5 1.154701 1.131542  1.116025 1.088480  0.058689133 -0.02754510
[3,] 1.0 1.154701 1.127818  1.116025 1.134159  0.003238925  0.01813376
[4,] 2.0 1.154701 1.120322  1.116025 1.130765 -0.007209740  0.01473987
[5,] 4.0 1.154701 1.212946  1.116025 1.063709 -0.085595369 -0.05231604
[6,] 6.0 1.154701 1.124592  1.116025 1.150806 -0.078749308  0.03478106
```

Problem 6. (Graduate Students Only) [15 mark] Carlin and Louis (2000) describe the use of a generalized logit model to fit dose-mortality data from Bliss (1935). The following table records the number of adult flour beetles killed after five hours of exposure to various levels of gaseous carbon disulphide. The number of insects killed y_i under dose w_i is assumed binomial $\text{Bin}(n_i, p_i)$, where the probability p_i of death is given by

$$p_i = \left(\frac{\exp(x_i)}{1 + \exp(x_i)} \right)^{m_1} ,$$

where $x_i = (w_i - \mu)/\sigma$. The prior distribution for μ, σ, and m_1 are assumed independent, where μ is assigned a uniform prior, σ is assigned a prior proportional to $1/\sigma$, and m_1 is gamma with parameters $a_o = 0.25$ and $b_o = 4$. If one transforms μ, σ, and m_1 to the real-valued parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3) = (\mu, \log(\sigma), \log(m_1))$, then Carlin and Louis (2000) show the posterior density is given by

$$g(\boldsymbol{\theta} \,|\, \text{data}) \propto \prod_{i=1}^{8} \left[p_i^{y_i} \, (1 - p_i)^{n_i - y_i} \right] \exp(a_o \, \theta_3 - e^{\theta_3} / b_o).$$

Flour beetle mortality data

Dosage w_i	Number Killed y_i	Number Exposed n_i
1.6907	6	59
1.7242	13	60
1.7552	18	62
1.7842	28	56
1.8113	52	63
1.8369	53	59
1.8610	61	62
1.8839	60	60

a) **[7 mark]** Write an R function that defines the log posterior of $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$.
SOLUTION:

```
remove(list=ls())

log.post<-function(theta){
a0<-0.25
b0<-4
w<-c(1.6907,1.7242,1.7552,1.7842,1.8113,1.8369,1.8610,1.8839)
y<-c(6,13,18,28,52,53,61,60)
n<-c(59,60,62,56,63,59,62,60)
mu<-theta[1]
sig<-exp(theta[2])
m1<-exp(theta[3])
x<-(w-mu)/sig
p1<-(exp(x)/(1+exp(x)))^m1
lg<-sum(y*log(p1)+(n-y)*log(1-p1))+a0*theta[3]-exp(theta[3])/b0
return(lg)
}

A2Q6.RW.MH<-function(N,Theta0){
Theta<-matrix(NA,nrow=N+1,ncol=3)
Theta[1,]<-Theta0
t<-1
accept.prob<-NA
Test<-NA
while(t<=N){
# generate new points
e.prim<-c(rnorm(1,mean=0,sd=0.00012),rnorm(1,mean=0,sd=0.033),rnorm(1,mean=0,sd=0.10))
Theta.prim<-Theta[t,]+e.prim
u<-runif(1,min=0,max=1)

# Compute acceptance prob.
accept.prob[t]<-min(exp(log.post(Theta.prim))/
exp(log.post(Theta[t,])),1)
```

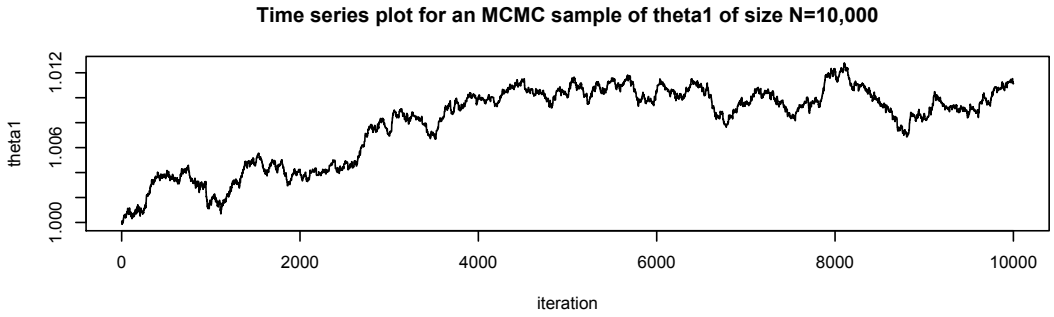
```
# Test the new points
    if (u<=accept.prob[t]){
Test[t]<-1 # Accept
Theta[t+1,]<-Theta.prim
}
else{
Test[t]<-0 # Reject
Theta[t+1,]<-Theta[t,]
}
# set t to t+1 goto step 2.
t<-t+1
}
# return samples and the acceptance probabilities in every iterations
return(list(Theta,accept.prob,Test))
}
```

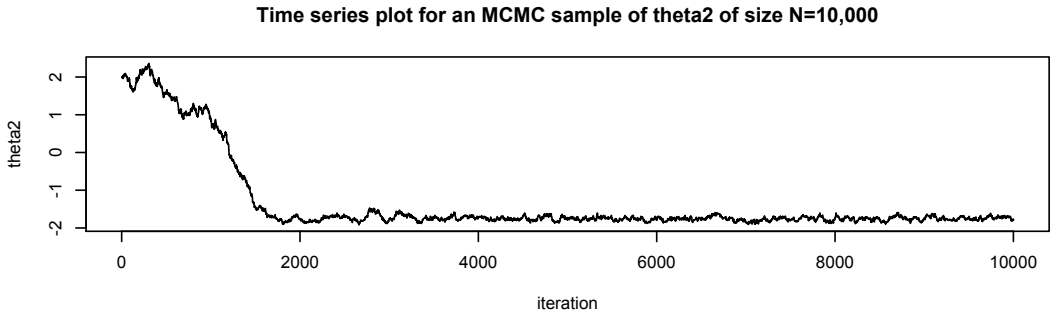
b) [**8 mark**] Carlin and Louis (2000) suggest running a Metropolis random walk chain with a multivariate normal proposal density where the variance covariance matrix is diagonal with elements 0.00012, 0.0.033, and 0.10. Implement a Metropolis algorithm as an R function to run this chain for 10,000 iterations. Compute the acceptance rate and the 5th and 95th percentiles for each parameter.
SOLUTION:

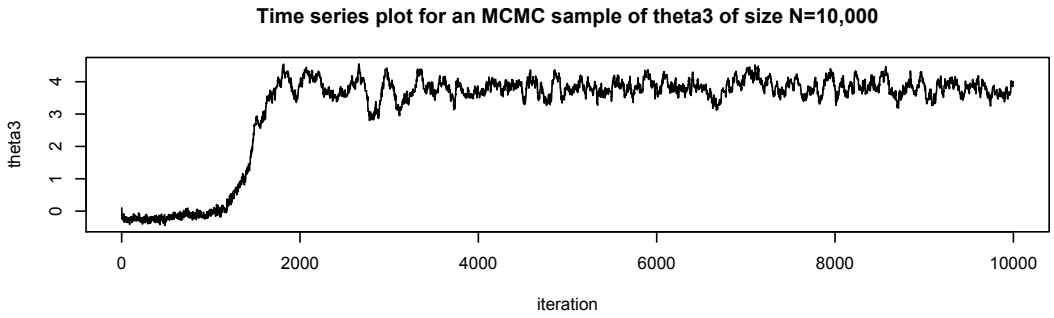
```
OUT<-A2Q6.RW.MH(10000,c(1,5,0.1))
par(mfrow=c(3,1))
ts.plot(OUT[[1]][,1],xlab="iteration",ylab="theta1",main="Time series plot for an MCMC sample of thet
ts.plot(OUT[[1]][,2],xlab="iteration",ylab="theta2",main="Time series plot for an MCMC sample of thet
ts.plot(OUT[[1]][,3],xlab="iteration",ylab="theta3",main="Time series plot for an MCMC sample of thet
##


# The acceptance rate is
mean(OUT[[2]])
0.451234
```

Looking at the time series plots in Figure below , we decide to exclude the first 3000 draws and keep 7000 sample points for the 5th and 95th percentiles for each parameter. Below is the calculations.

```
Out.burn.in<-OUT[[1]][-c(1:3000),]

# theta 1
c(quantile(Out.burn.in[,1],0.05), quantile(Out.burn.in[,1],0.95))
      5%      95%
1.007868 1.011488

# theta 2
c(quantile(Out.burn.in[,2],0.05), quantile(Out.burn.in[,2],0.95))
       5%       95%
```


Time series plot for an MCMC sample of theta1 of size N=10,000
theta1
1.000
1.006
1.012
0
2000
4000
6000
8000
10000
iteration


Time series plot for an MCMC sample of theta2 of size N=10,000
theta2
-2
-1
0
1
2
0
2000
4000
6000
8000
10000
iteration


Time series plot for an MCMC sample of theta3 of size N=10,000
theta3
0
1
2
3
4
0
2000
4000
6000
8000
10000
iteration

```
-1.847222 -1.654833

# theta 3
c(quantile(Out.burn.in[,3],0.05), quantile(Out.burn.in[,3],0.95))
      5%      95%
3.399637 4.231727
```

Problem 7. **[15 mark]** Let X_1, X_2 be a random sample from Cauchy$(\theta, 1)$. Suppose we observe $x_1 = 1.3$ and $x_2 = 15$. Consider an improper prior $\pi(\theta) = 1, \quad \theta \in \mathbb{R}$. Note that in this case the prior does not integrate to 1 (in fact the integral diverges) but the posterior will be a proper distribution.

a) **[3 mark]** Graph the posterior density.
SOLUTION: The data distribution is

$$f(x \mid \theta) = \frac{1}{\pi\,(1 + (x - \theta)^2)} \quad \text{for} \quad -\infty < x < \infty \quad \text{and} \quad -\infty < \theta < \infty\,.$$

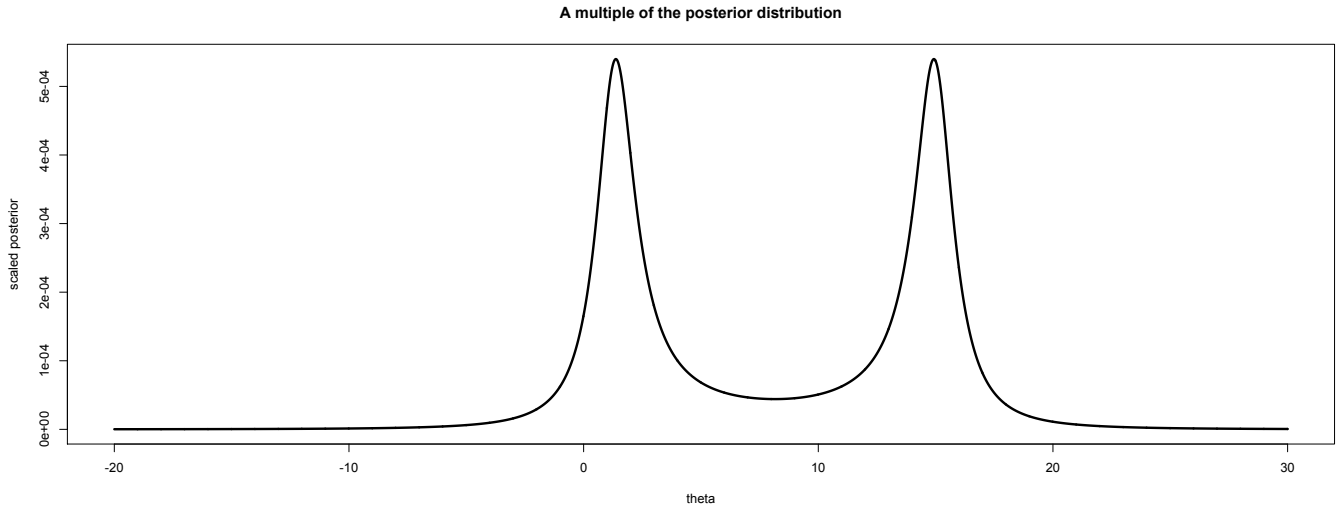
Thus the joint distribution of X_1 and X_2 is

$$f(x_1, x_2 \mid \theta) = \frac{1}{\pi^2\,(1 + (x_1 - \theta)^2)\;(1 + (x_2 - \theta)^2)}$$

The given $x_1 = 1.3$ and $x_2 = 15$ the posterior distribution is

$$\begin{aligned}\pi(\theta \mid x_1 = 1.3,\, x_2 = 15) &\propto f(x_1, x_2 \mid \theta)\,\pi(\theta) \\ &= \frac{1}{\pi^2\,(1 + (1.3 - \theta)^2)\,(1 + (15 - \theta)^2)}\end{aligned}$$

```
theta<-seq(-20,30,0.01)
post<-1/(pi^2*(1+(1.3-theta)^2)*(1+(15-theta)^2))
plot(theta,post,ylab="scaled posterior",type="l",lwd=3,
main="A multiple of the posterior distribution")
```



b) **[5 mark]** Write R codes for a Metropolis algorithm for this problem using a symmetric Cauchy proposal distribution. Turn the scale parameter of this proposal distribution appropriately.
SOLUTION: The posterior distribution is

$$\pi(\theta \mid x_1 = 1.3,\, x_2 = 15) \propto \frac{1}{\pi^2\,(1+(1.3-\theta)^2)\,(1+(15-\theta)^2)} \qquad -\infty < \theta < \infty\,.$$

The proposal distribution is

$$q(\theta,\, \theta') = \frac{1}{\pi\,\sigma\,\left(1+\left(\frac{\theta'-\theta}{\sigma}\right)^2\right)}$$

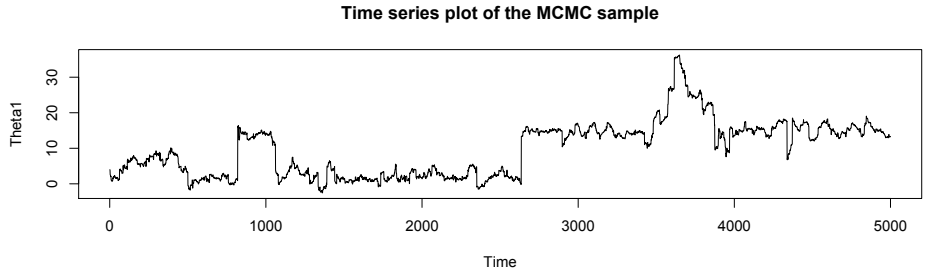
The steps of the algorithm are

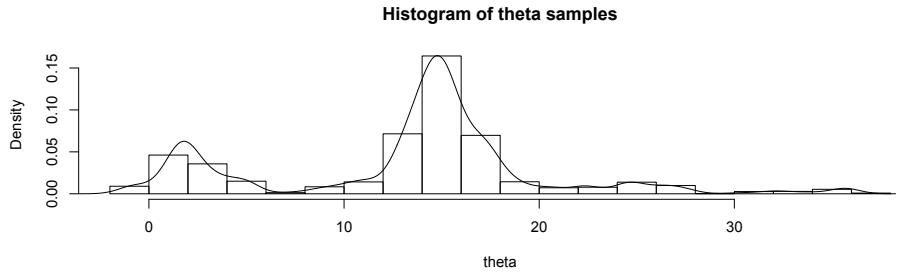
1. For $n = 0$, start with initial value θ_0
2. At step n, generate θ^* from Cauchy$(\theta_n,\, \sigma)$.
3. Calculate the ratio

$$r = \frac{\pi(\theta^* \mid x_1 = 1.3,\, x_2 = 15)}{\pi(\theta_n \mid x_1 = 1.3,\, x_2 = 15)} = \frac{(1+(1.3-\theta_n)^2)\,(1+(15-\theta_n)^2)}{(1+(1.3-\theta^*)^2)\,(1+(15-\theta^*)^2)}\,.$$

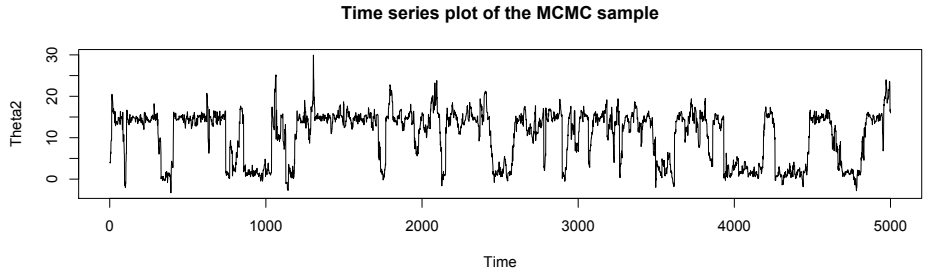
4. If $r \geq 1$ set $\theta_{n+1} = \theta^*$, otherwise generate $u \sim U(0,1)$. If $u < r$ set $\theta_{n+1} = \theta^*$, else set $\theta_{n+1} = \theta_n$.
5. Set $n = n+1$, goto step 2.

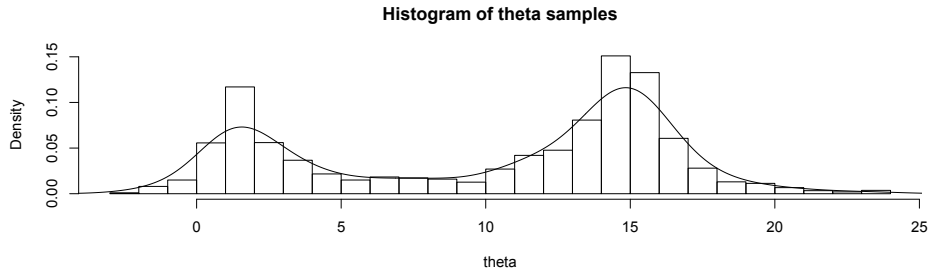
```
# Part b)
M.H.Cauchy<-function(N,theta0,sigma){
theta<-c(theta0)
for (i in 2:N){
theta.prim<-rcauchy(1,location=theta[i-1],scale=sigma)
r<-(post(theta.prim)*dcauchy(theta[i-1],location=theta.prim,scale=sigma))/
(post(theta[i-1])*dcauchy(theta.prim,location=theta[i-1],scale=sigma))
if (r>=1){theta[i]<-theta.prim}
if (runif(1,min=0,max=1)<r){theta[i]<-theta.prim}
else theta[i]<-theta[i-1]
}
return(theta)
}
Theta1<-M.H.Cauchy(5000,4,0.1)
par(mfrow=c(2,1))
ts.plot(Theta1,main="Time series plot of the MCMC sample")
Theta1<-Theta1[-c(1:2000)]
hist(Theta1,breaks=20,freq=FALSE,main="Histogram of theta samples",xlab="theta")
lines(density(Theta1))
```





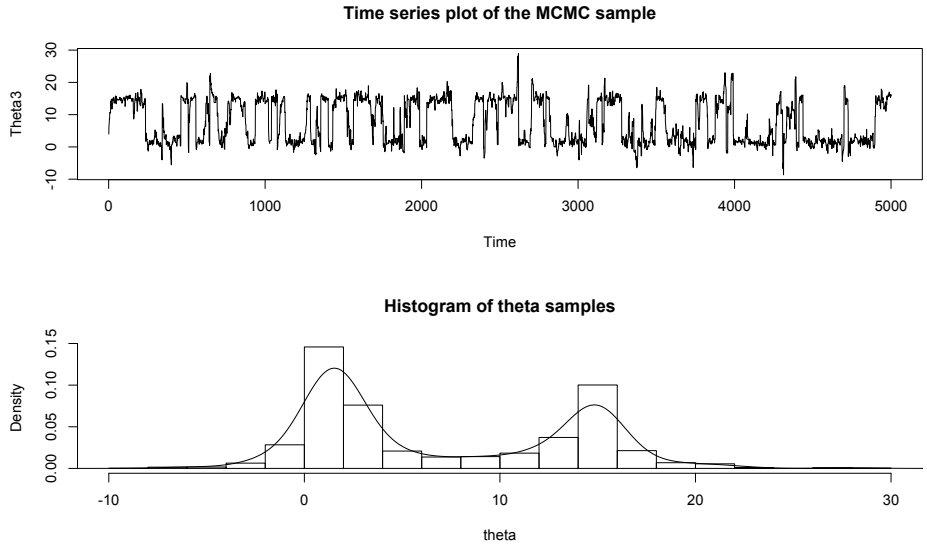

```
Theta2<-M.H.Cauchy(5000,4,0.5)
par(mfrow=c(2,1))
ts.plot(Theta2,main="Time series plot of the MCMC sample")
Theta2<-Theta2[-c(1:2000)]
hist(Theta2,breaks=20,freq=FALSE,main="Histogram of theta samples",xlab="theta")
lines(density(Theta2))
```



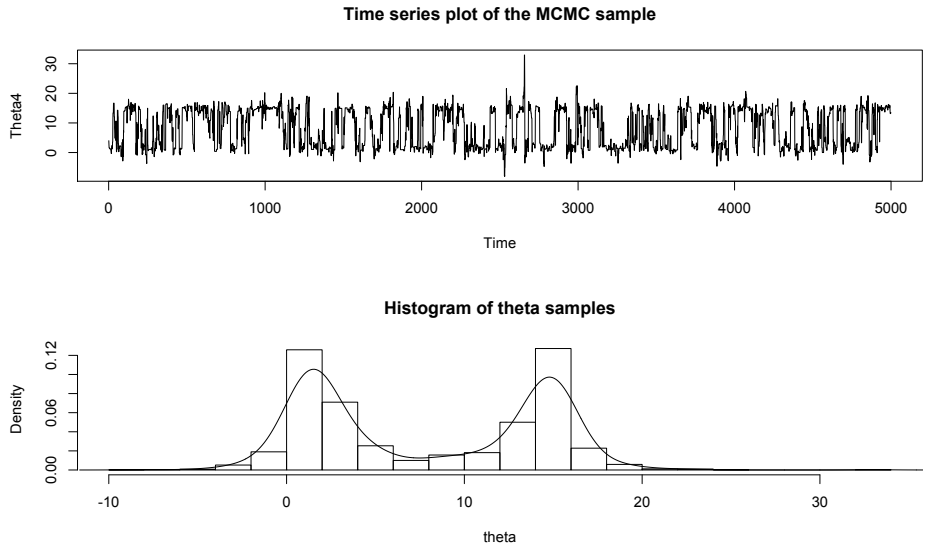



```
Theta3<-M.H.Cauchy(5000,4,1)
par(mfrow=c(2,1))
ts.plot(Theta3,main="Time series plot of the MCMC sample")
Theta3<-Theta3[-c(1:2000)]
```

```
hist(Theta3,breaks=20,freq=FALSE,main="Histogram of theta samples",xlab="theta")
lines(density(Theta3))
```

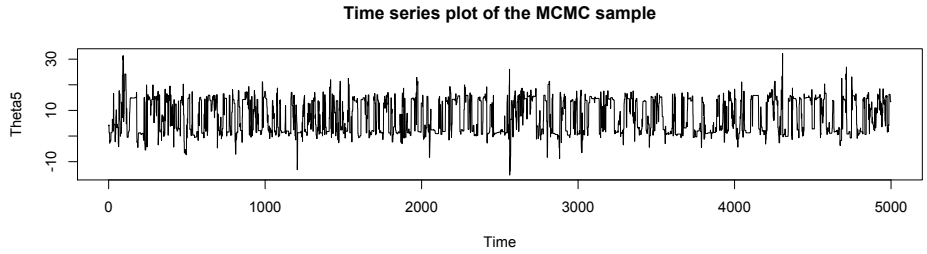


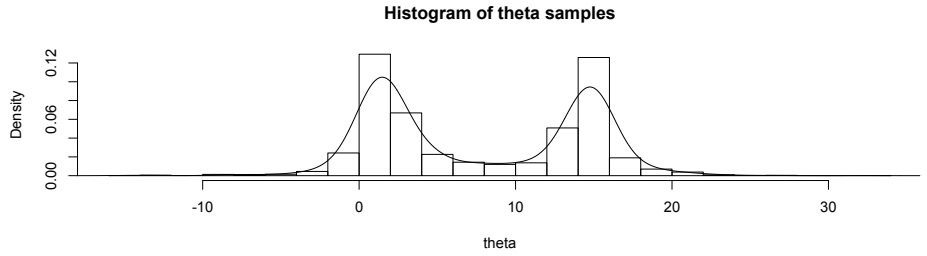

```
Theta4<-M.H.Cauchy(5000,4,3)
par(mfrow=c(2,1))
ts.plot(Theta4,main="Time series plot of the MCMC sample")
Theta4<-Theta4[-c(1:2000)]
hist(Theta4,breaks=20,freq=FALSE,main="Histogram of theta samples",xlab="theta")
lines(density(Theta4))
```




```
Theta5<-M.H.Cauchy(5000,4,6)
par(mfrow=c(2,1))
ts.plot(Theta5,main="Time series plot of the MCMC sample")
```

```
Theta5<-Theta5[-c(1:2000)]
hist(Theta5,breaks=20,freq=FALSE,main="Histogram of theta samples",xlab="theta")
lines(density(Theta5))
```



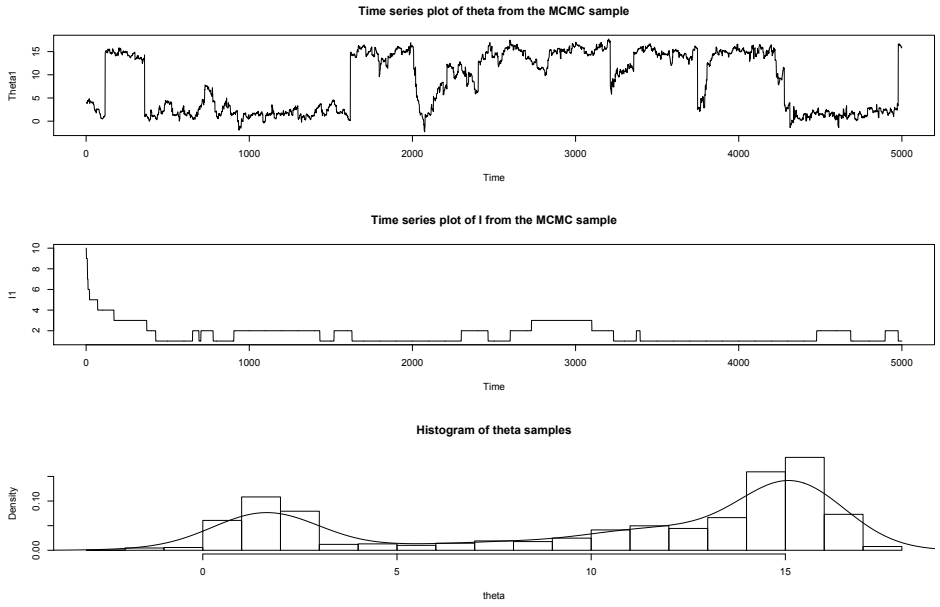


c) [**4 mark**] Write R codes for a simulated tempering with ladder of 10 inverse-temperatures 0.1, 0.2, . . . , 1.

```
# Part c) Simulated Tempering
# Tempreature
Temp<-1/(seq(1,0.1,by=-0.1))
# propsal transition probabilities for i
qij<-function(i,j){
if ((i==1 && j==2) || (i==10 && j==9)){
q<-1
}
else if (abs(i-j)==1 && i>=2 && i<= 9){
q<-1/2
}
else { q<-0}
return(q)
}
# marginal uniform dist. for i
p.i<-rep(1/10,10)
#
SimTemp<-function(N,theta0,i0,Temp,sig){
theta<-c(theta0)
I<-c(i0)
for (n in 2:N){
    # generate theta.prim from the tempered target distribution
    theta.prim<-rcauchy(1,location=theta[n-1],scale=sig)
r1<-((post(theta.prim)^Temp[I[n-1]])*dcauchy(theta[n-1],location=theta.prim,scale=sig))/
((post(theta[n-1])^Temp[I[n-1]])*dcauchy(theta.prim,location=theta[n-1],scale=sig))
if (r1>=1){theta[n]<-theta.prim}
else if (runif(1,min=0,max=1)<r1){theta[n]<-theta.prim}
else {theta[n]<-theta[n-1]}

# generate I[n] from proposal q
if (I[n-1]==1) {
j<-2
}
else if (I[n-1]==10){
j<-9
}
```

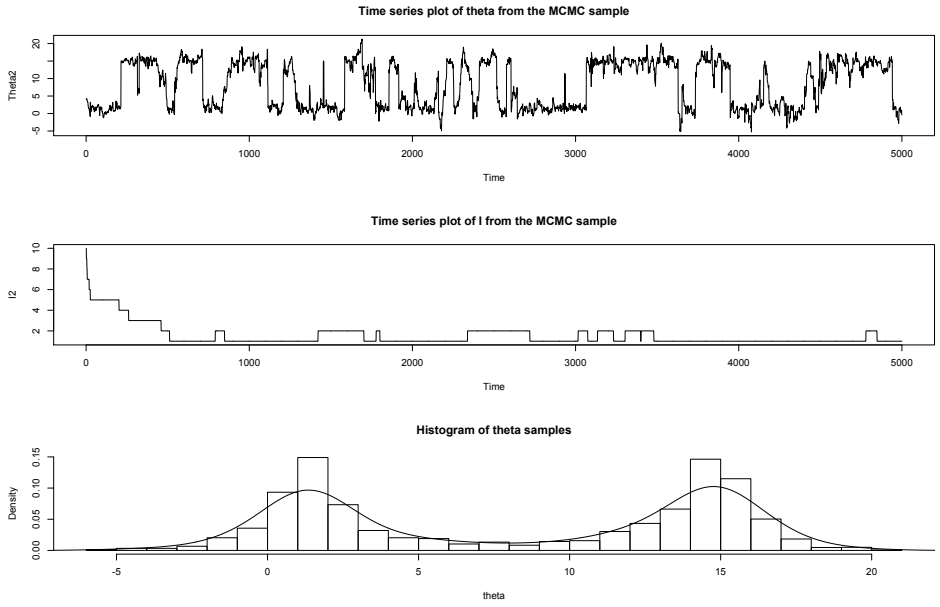
```
else {
u<-runif(1,0,1)
if (u<0.5){ j<-I[n-1]-1}
else { j<-I[n-1]+1}
}
# compute MH ratio for I[n-1], I[n] and theta[n]
r2<-(post(theta[n])^Temp[j])*p.i[j]*qij(I[n-1],j)/(post(theta[n])^Temp[I[n-1]])*p.i[I[n-1]]*qij(j,I[n-1])
if (r2>=1){I[n]<-j}
else if (runif(1,min=0,max=1)<r2){I[n]<-j}
else {I[n]<-I[n-1]}
}
return(cbind(theta,I))
}
```

```
Out1<-SimTemp(N=5000, theta0=4, i0=10, Temp, sig=0.1)
Theta1<-Out1[,1]
I1<-Out1[,2]
par(mfrow=c(3,1))
ts.plot(Theta1,main="Time series plot of theta from the MCMC sample")
ts.plot(I1,main="Time series plot of I from the MCMC sample")
Theta1.1<-Theta1[-c(1:2000)]
hist(Theta1.1,breaks=20,freq=FALSE,main="Histogram of theta samples",xlab="theta")
lines(density(Theta1.1))
```

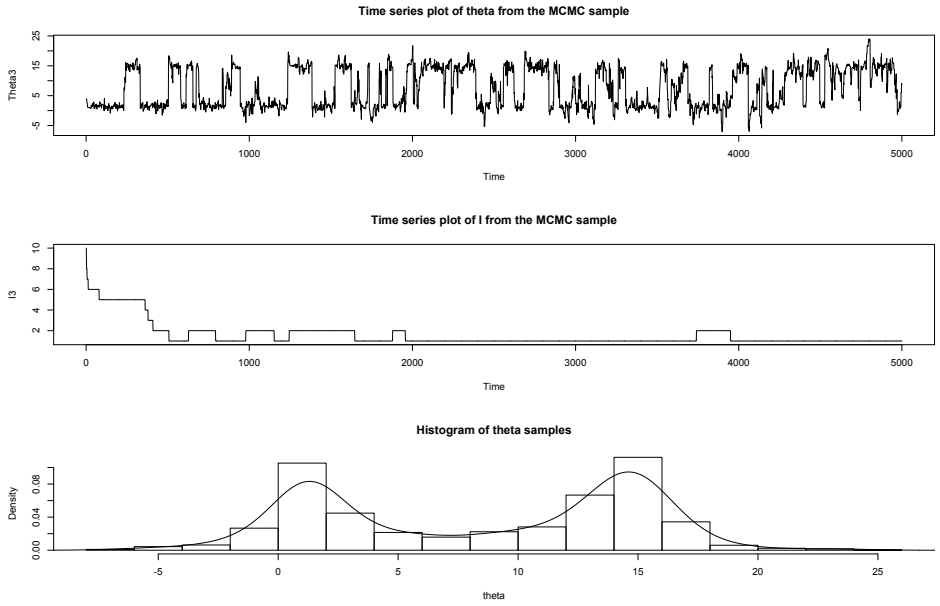



```
Out2<-SimTemp(N=5000, theta0=4, i0=10, Temp, sig=0.5)
Theta2<-Out2[,1]
I2<-Out2[,2]
par(mfrow=c(3,1))
ts.plot(Theta2,main="Time series plot of theta from the MCMC sample")
ts.plot(I2,main="Time series plot of I from the MCMC sample")
Theta2.1<-Theta2[-c(1:2000)]
hist(Theta2.1,breaks=20,freq=FALSE,main="Histogram of theta samples",xlab="theta")
```

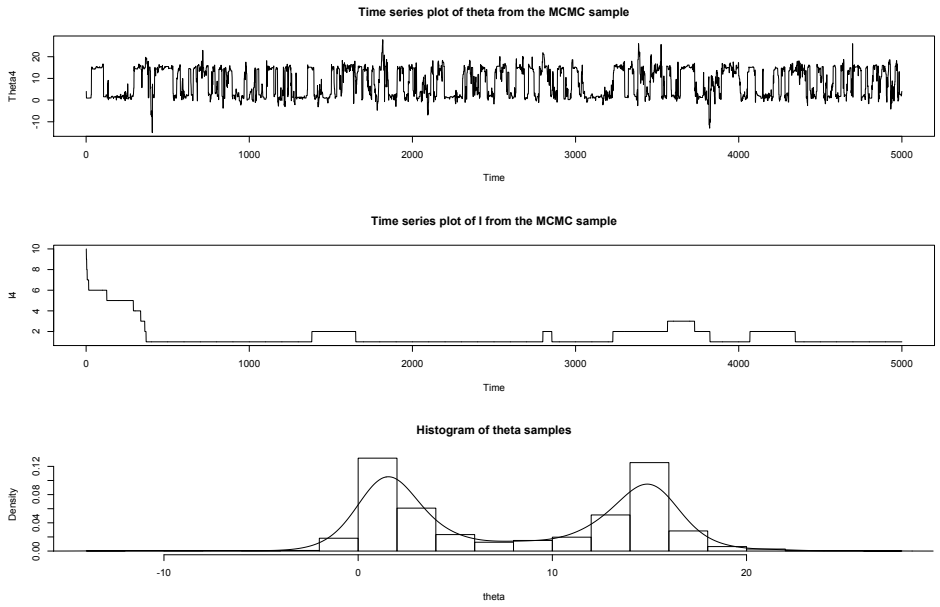
```
lines(density(Theta2.1))
```



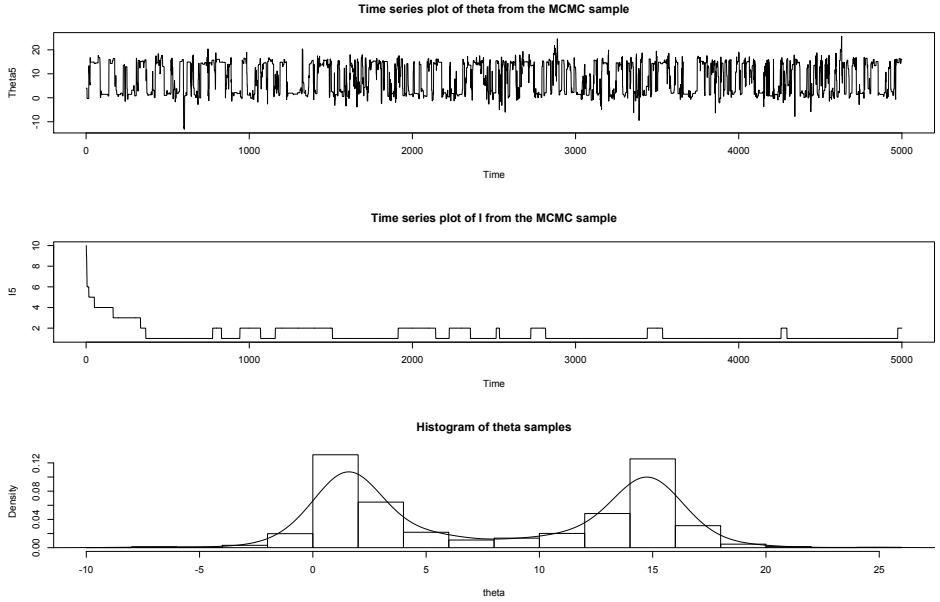

```
Out3<-SimTemp(N=5000, theta0=4, i0=10, Temp, sig=1)
Theta3<-Out3[,1]
I3<-Out3[,2]
par(mfrow=c(3,1))
ts.plot(Theta3,main="Time series plot of theta from the MCMC sample")
ts.plot(I3,main="Time series plot of I from the MCMC sample")
Theta3.1<-Theta3[-c(1:2000)]
hist(Theta3.1,breaks=20,freq=FALSE,main="Histogram of theta samples",xlab="theta")
lines(density(Theta3.1))
```



```
Out4<-SimTemp(N=5000, theta0=4, i0=10, Temp, sig=3)
Theta4<-Out4[,1]
I4<-Out4[,2]
par(mfrow=c(3,1))
ts.plot(Theta4,main="Time series plot of theta from the MCMC sample")
ts.plot(I4,main="Time series plot of I from the MCMC sample")
Theta4.1<-Theta4[-c(1:2000)]
hist(Theta4.1,breaks=20,freq=FALSE,main="Histogram of theta samples",xlab="theta")
lines(density(Theta4.1))
```




```
Out5<-SimTemp(N=5000, theta0=4, i0=10, Temp, sig=6)
Theta5<-Out5[,1]
I5<-Out5[,2]
par(mfrow=c(3,1))
ts.plot(Theta5,main="Time series plot of theta from the MCMC sample")
ts.plot(I5,main="Time series plot of I from the MCMC sample")
Theta5.1<-Theta5[-c(1:2000)]
hist(Theta5.1,breaks=20,freq=FALSE,main="Histogram of theta samples",xlab="theta")
lines(density(Theta5.1))
```



d) **[3 mark]** Compare your answers in (b) and (c) to the graph (a).
SOLUTION: Overall the samples obtained from simulated tempering captures the shape of the posterior better than the samples from Metropolis algorithm. This is specially the case when the scale parameter is larger the 0.1.

Problem 8. [15 mark] If $\widehat{\theta}$ is the MLE for θ, the following asymptotic approximation is often used for statistical inference:

$$\sqrt{n}\,(\widehat{\theta} - \theta) \sim N(0, I^{-1}(\widehat{\theta})),$$

where $I(\widehat{\theta})$ is the Fisher information evaluated at the MLE.
Now consider the model $Y_i \sim Poisson(\beta_0 + \beta_1\, x_i)$ for the following data set

y	40	10	21	16	11	7	4	4	2	3
x	30.11	13.91	10.63	8.68	7.14	5.99	5.23	4.37	4.09	3.54

a) **[5 mark]** Find the asymptotic variance of $\widehat{\beta} = (\widehat{\beta}_0, \widehat{\beta}_1)^T$ in this model.
SOLUTION: You may have done this in one of the following ways. I accept both methods.
Method 1. You have assumed that $\mathrm{E}\,[Y_i \mid x_i] = \beta_0 + \beta_1\, x_i$. In this case the likelihood function is

$$L(\beta_0, \beta_1) = \prod_{i=1}^{n} \frac{(\beta_0 + \beta_1\, x_i)^{y_i}\ \exp(-\beta_0 - \beta_1\, x_i)}{y_i!}\,,$$

and thus the log-likelihood is

$$\ell(\beta_0, \beta_1) = \sum_{i=1}^{n} \left[\, y_i\, \log(\beta_0 + \beta_1\, x_i) - (\beta_0 + \beta_1\, x_i) - \log(y_i!)\,\right]\,.$$

The following codes minimizes the negative log-likelihood function. Note that we have removed $\log(y_i!)$ as it does not affect the optimization.

```
prlike<-function(theta){
a<-theta[1]
```

```
b<-theta[2]
x<-c(30.11,13.91,10.63,8.68,7.14,5.99,5.23,4.37,4.09,3.54)
y<-c(40,10,21,16,11,7,4,4,2,3)
nL<- -sum(y*log(a+b*x)-(a+b*x))
return(nL)
}
# We tried many intial values and concluded that the
# the global minimum or MLE is attained in
optim(c(0.1,2),prlike)
$par
[1] -2.309691  1.506086
$value
[1] -210.1964
$counts
function gradient
     109       NA
$convergence
[1] 0
$message
NULL
```

So $\widehat{\beta} = (\widehat{\beta}_0, \widehat{\beta}_1)^T = (-2.309691, 1.506086)^T$. To find the fisher information we need to obtain the second derivatives of the log-likelihood and take expectation with respect to the random variables Y_i. The first derivatives are

$$\frac{\partial \ell(\beta_0, \beta_1)}{\partial \beta_0} = \sum_{i=1}^{n} \left[\frac{y_i}{\beta_0 + \beta_1 x_i} - 1 \right]$$
$$\frac{\partial \ell(\beta_0, \beta_1)}{\partial \beta_1} = \sum_{i=1}^{n} \left[\frac{y_i x_i}{\beta_0 + \beta_1 x_i} - x_i \right]$$

The second derivatives are

$$\frac{\partial^2 \ell(\beta_0, \beta_1)}{\partial \beta_0^2} = -\sum_{i=1}^{n} \left[\frac{y_i}{(\beta_0 + \beta_1 x_i)^2} \right]$$
$$\frac{\partial^2 \ell(\beta_0, \beta_1)}{\partial \beta_1^2} = -\sum_{i=1}^{n} \left[\frac{y_i x_i^2}{(\beta_0 + \beta_1 x_i)^2} \right]$$
$$\frac{\partial^2 \ell(\beta_0, \beta_1)}{\partial \beta_0 \beta_1} = -\sum_{i=1}^{n} \left[\frac{y_i x_i}{(\beta_0 + \beta_1 x_i)^2} \right]$$

Using $\mathrm{E}[Y_i \mid x_i] = \beta_0 + \beta_1 x_i$, the entries of the Fisher information are

$$\mathrm{E}\left[-\frac{\partial^2 \ell(\beta_0, \beta_1)}{\partial \beta_0^2} \right] = \sum_{i=1}^{n} \left[\frac{1}{(\beta_0 + \beta_1 x_i)} \right]$$
$$\mathrm{E}\left[-\frac{\partial^2 \ell(\beta_0, \beta_1)}{\partial \beta_1^2} \right] = \sum_{i=1}^{n} \left[\frac{x_i^2}{(\beta_0 + \beta_1 x_i)} \right]$$
$$\mathrm{E}\left[-\frac{\partial^2 \ell(\beta_0, \beta_1)}{\partial \beta_0 \beta_1} \right] = \sum_{i=1}^{n} \left[\frac{x_i}{(\beta_0 + \beta_1 x_i)} \right]$$

Substituting $\widehat{\beta} = (\widehat{\beta}_0,\, \widehat{\beta}_1)^T = (-2.309691,\, 1.506086)^T$, and data in these entries, the estimated Fisher information matrix is obtained using the following `R` codes

```
prFisher<-function(theta){
        a<-theta[1]
b<-theta[2]
x<-c(30.11,13.91,10.63,8.68,7.14,5.99,5.23,4.37,4.09,3.54)
y<-c(40,10,21,16,11,7,4,4,2,3)
prF11<--sum(1/(a+b*x))
prF22<--sum((x^2)/(a+b*x))
prF12<-prF21<--sum(x/(a+b*x))
prF<-matrix(c(prF11,prF12,prF21,prF22),nrow=2,byrow=TRUE)
return(prF)
}

# Fisher information
FisherInfo<-prFisher(optim(c(0.1,2),prlike)$par)
FisherInfo
         [,1]      [,2]
[1,] 1.514571  8.962429
[2,] 8.962429 75.952127

# Asymptotic variance-covariance matrix
AsyVar<-solve(FisherInfo)
AsyVar
           [,1]        [,2]
[1,]  2.1882088 -0.25821090
[2,] -0.2582109  0.04363534
```

Method 2. You may assume that $\mathrm{E}\,[Y_i \mid x_i] = \exp(\beta_0 + \beta_1\, x_i)$. This is usually referred as the Poisson regression in the literature of the the generalized linear models (GLM) and the `glm` function in `R` produces the estimate and fits the model. The likelihood in this model is

$$L(\beta_0, \beta_1) = \prod_{i=1}^{n} \frac{\exp(y_i(\beta_0 + \beta_1\, x_i))\, \exp(-\exp(\beta_0 + \beta_1\, x_i))}{y_i!}\,,$$

and thus the log-likelihood is

$$\ell(\beta_0, \beta_1) = \sum_{i=1}^{n} \left[\, y_i(\beta_0 + \beta_1\, x_i) - \exp(\beta_0 + \beta_1\, x_i) - \log(y_i!)\,\right]\,.$$

```
# Method 2
x<-c(30.11,13.91,10.63,8.68,7.14,5.99,5.23,4.37,4.09,3.54)
y<-c(40,10,21,16,11,7,4,4,2,3)
fit1<-glm(y~x,family="poisson")
summary(fit1)
Call:
```

```
glm(formula = y ~ x, family = "poisson")

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0851  -1.2333  -0.6902   0.6619   2.8506

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 1.583448   0.161761   9.789   <2e-16 ***
x           0.072181   0.008334   8.661   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 86.825  on 9  degrees of freedom
Residual deviance: 23.728  on 8  degrees of freedom
AIC: 67.123

Number of Fisher Scoring iterations: 5

# Estimated coefficients
fit1$coefficients
(Intercept)           x
 1.58344796  0.07218113

# Estimated Fisher information
summary(fit1)$cov.scaled
             (Intercept)             x
(Intercept)  0.026166756 -1.108533e-03
x           -0.001108533  6.945693e-05
```

Note that Model 1 and Model 2 produce totally different estimates as they are totally different models. Model 2 is more realistic as it allows the parameters to vary in $\mathbb{R}$ without any restriction as in this model $\mathrm{E}(Y \mid x) = \exp(\beta_0 + \beta_1 x)$. For model 1 though, $\mathrm{E}(Y \mid x) = \beta_0 + \beta_1 x$ may produce negative values for some values of x and β_0 and β_1 while $\mathrm{E}(Y \mid x)$ must be alway positive. In part (b) and (c) I will only assume model 2, but it is acceptable if you have used model 1.

b) [**5 mark**] Since the sample size is quite small here, the asymptotic variance may not be a good approximation. Produce an alternative estimate for the variance of $\widehat{\beta}$ using the Bootstrap (both parametric and nonparametric).

SOLUTION:

```
## Part b
# Model 2: Parametric bootstrap
B<-1000
```

```
x<-c(30.11,13.91,10.63,8.68,7.14,5.99,5.23,4.37,4.09,3.54)
y<-c(40,10,21,16,11,7,4,4,2,3)
fit1<-glm(y~x,family="poisson")
Boot.Theta.p<-matrix(NA,nrow=B,ncol=2)
theta.hat<-fit1$coefficients
for(b in 1:B){
xb<-sample(x,length(x),replace=TRUE)
lamb.hat<-theta.hat[1]+theta.hat*xb
yb<-NA
for(i in 1:length(x)){
yb[i]<-rpois(1,lamb.hat[i])
}
Boot.Theta.p[b,]<-glm(yb~xb,family="poisson")$coefficients
}
```

The parametric bootstrap estimate of the variance-covariance matrix of $\widehat{\beta}$ in Model 2 is

```
Sigma.p<-var(Boot.Theta.p)
> Sigma.p
            [,1]         [,2]
[1,]  0.37990702 -0.045528239
[2,] -0.04552824  0.006100186
```

```
# Method 2: Nonparametric bootstrap
B<-1000
x<-c(30.11,13.91,10.63,8.68,7.14,5.99,5.23,4.37,4.09,3.54)
y<-c(40,10,21,16,11,7,4,4,2,3)
fit1<-glm(y~x,family="poisson")
Boot.Theta.np<-matrix(NA,nrow=B,ncol=2)
theta.hat<-fit1$coefficients
for(b in 1:B){
I<-sample(1:length(x),length(x),replace=TRUE)
xb<-x[I]
yb<-y[I]
Boot.Theta.np[b,]<-glm(yb~xb,family="poisson")$coefficients
}
Sigma.np<-var(Boot.Theta.np)
```

The non-parametric bootstrap estimate of the variance-covariance matrix of $\widehat{\beta}$ in Model 2 is

```
Sigma.np<-var(Boot.Theta.p)
> Sigma.np
            [,1]         [,2]
[1,]  0.30726398 -0.039268527
[2,] -0.03926853  0.005702187
```

c) [**5 mark**] Compute the 95% bootstrap confidence intervals for β_1 (both parametric and nonparametric).
SOLUTION: We obtain the normal confidence intervals based on parametric and non-pramateric bootstraps for β_1.

```
# Parametric
c(theta.hat[2]-qnorm(0.975)*sd(Boot.Theta.p[,2]),theta.hat[2]+qnorm(0.975)*sd(Boot.Theta.p[,2]))
     LB            UB
-0.08089928  0.22526155

# Nonparametric
c(theta.hat[2]-qnorm(0.975)*sd(Boot.Theta.np[,2]),theta.hat[2]+qnorm(0.975)*sd(Boot.Theta.np[,2]))

     LB          UB
-0.07582129  0.22018355
```