Homework 7

PkGu

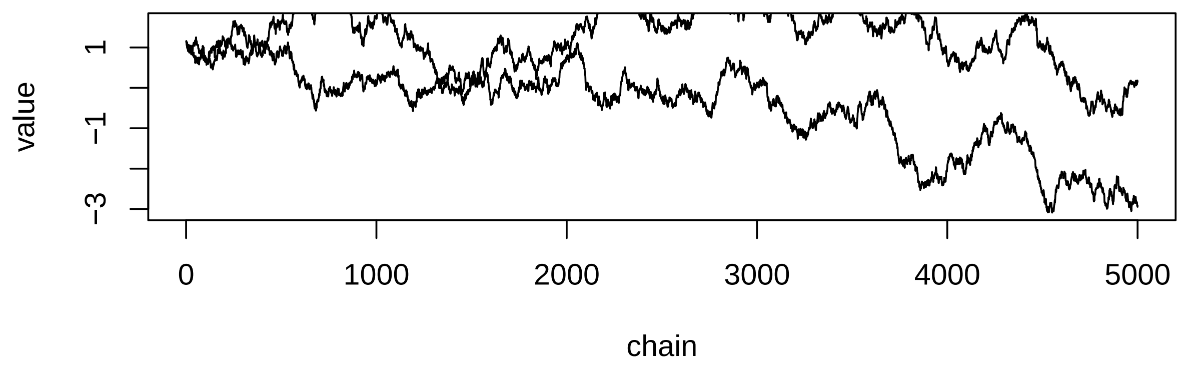
11/21/2021

P[245] 4.

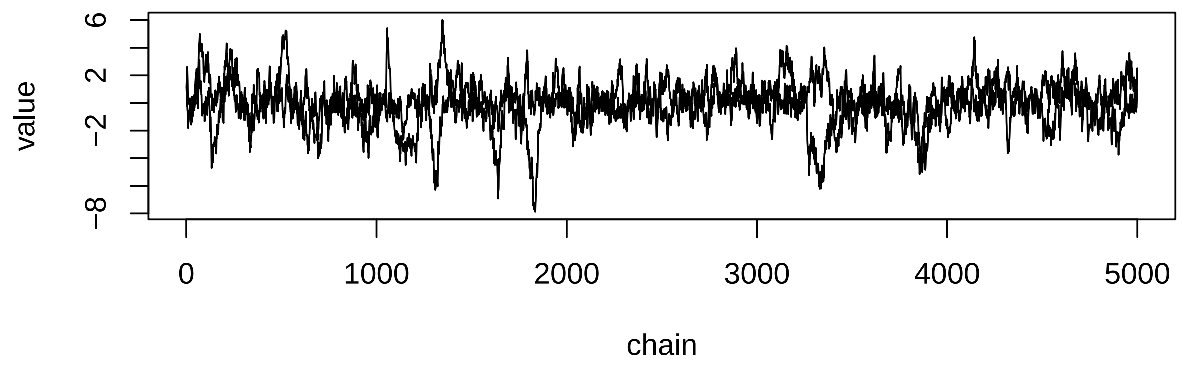
(1). Sampling from Cauchy distribution $C(0,1)$ using random walk Metropolis algorithm. The *p.d.f.* for Cauchy distribution $C(\mu,\lambda)$ is

$$f(x|\mu,\lambda)=\frac{1}{(x-\mu)^2+\lambda^2}\cdot\frac{\lambda}{\pi}$$

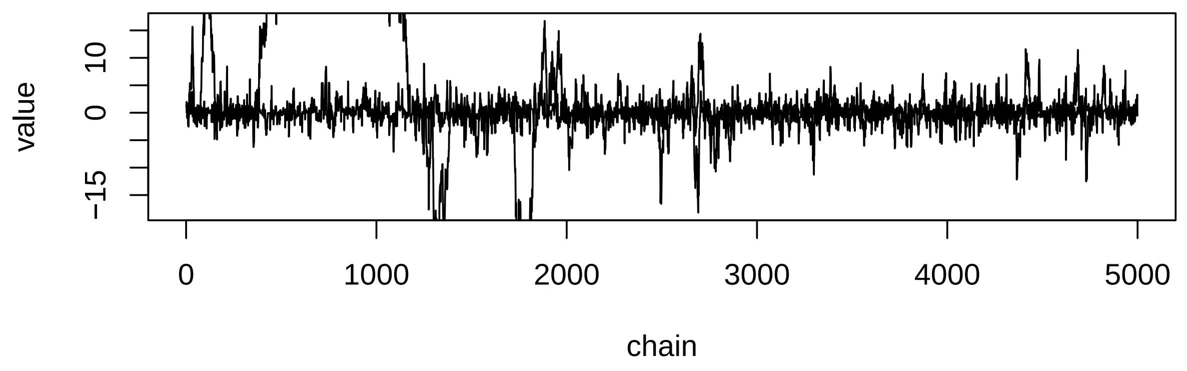
```r
rwm.cauchy<-function(mu,lambda,sigma,x0,N){
  # mu: mean of Cauchy distribution
  # lambda: scale of Cauchy distribution
  # sigma: step size
  # x0: initial value
  # N: length of chain
  x <- numeric(N)
  x[1] <- x0
  u <- runif(N)
  k <- 0
  for (i in 2:N) {
    y <- rnorm(1,x[i-1],sigma)
    if (u[i] < (((x[i-1]-mu)^2+lambda^2))/((y-mu)^2+lambda^2))
      x[i] <- y
    else {
      x[i] <- x[i-1]
      k <- k+1
    }
  }
  return (list(x=x,k=k))
}

sigma <- c(0.05,0.5,2.5,16)
N <- 5000
x0 <- runif(1,-5,5)
nseq <- 1:N
for (i in 1:4) {
  rw1 <- rwm.cauchy(0,1,sigma[i],x0,N)
  rw2 <- rwm.cauchy(0,1,sigma[i],x0,N)
  plot(nseq,rw1$x,type="l",xlab='chain',ylab='value')
  lines(nseq,rw2$x)
  print((rw1$k+rw2$k)/(2*N))
}
```



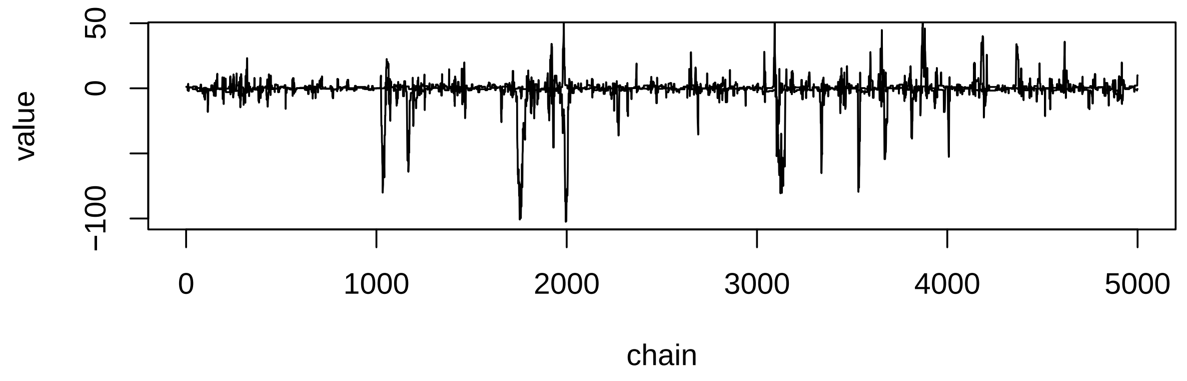
```
## [1] 0.0125
```



```
## [1] 0.1362
```



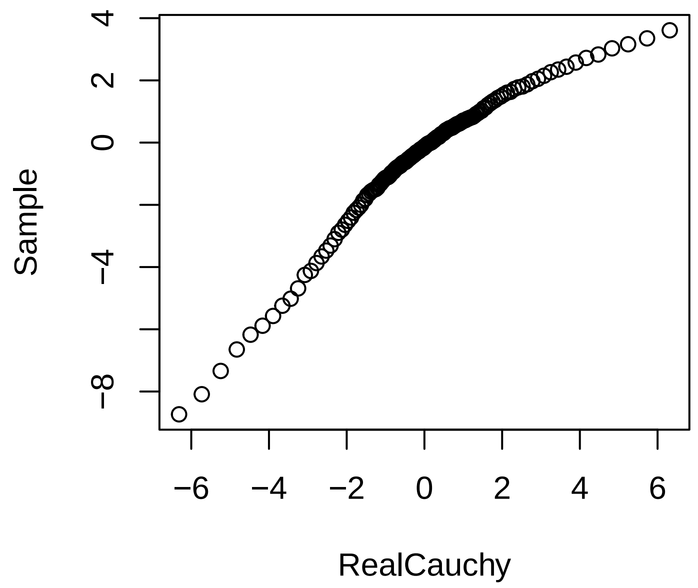
```
## [1] 0.407
```



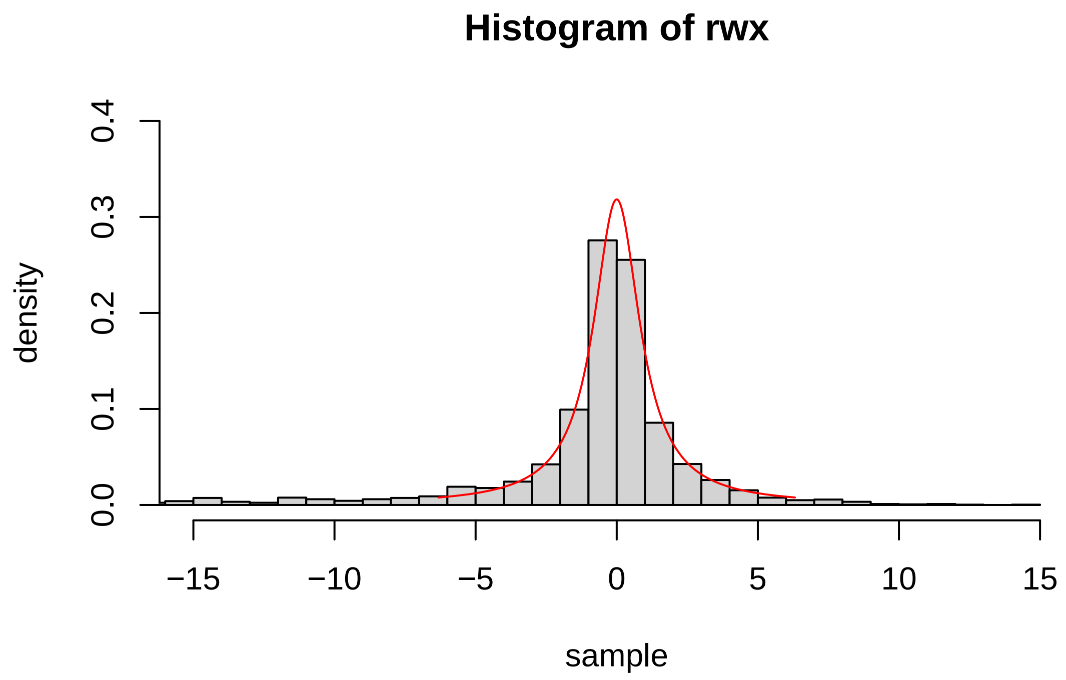
```
## [1] 0.8
```

Only for $\sigma = 2.5$ the rejection ratio is contained in $[0.15, 0.5]$.

```
rw <- rwm.cauchy(0,1,2.5,x0,N)
rwx <- rw$x[2001:N]
a <- seq(0.05,0.95,0.005)
Q <- quantile(rwx,a)
QR <- tan(pi/2*(2*a-1))
qqplot(QR,Q,xlab='RealCauchy',ylab='Sample')
```



```
hist(rwx,breaks='scott',xlab='sample',ylab='density',xlim=c(-15,15),ylim=c(0,0.4),freq=FALSE)
lines(QR,1/((QR^2+1)*pi),col='red')
```



(2).

```
N=3000
x0 <- runif(1,-5,5)
a <- c(0.05,seq(0.1,0.9,0.1),0.95)
QR <- tan(pi/2*(2*a-1))
rw1 <- (rwm.cauchy(0,1,sigma[1],x0,N))$x
rw2 <- (rwm.cauchy(0,1,sigma[2],x0,N))$x
rw3 <- (rwm.cauchy(0,1,sigma[3],x0,N))$x
rw4 <- (rwm.cauchy(0,1,sigma[4],x0,N))$x
rwx <- cbind(rw1, rw2, rw3, rw4)
rw <- rwx[1001:N, ]
Q <- apply(rw,2,function(x) quantile(x,a))
round(cbind(QR,Q),3)
```

```
##         QR    rw1    rw2    rw3    rw4
## 5%  -6.314 -0.440 -1.927 -4.409 -6.623
## 10% -3.078 -0.237 -1.332 -3.122 -2.835
## 20% -1.376 -0.084 -0.796 -1.565 -1.199
## 30% -0.727  0.060 -0.420 -0.871 -0.528
## 40% -0.325  0.179 -0.094 -0.420 -0.110
## 50%  0.000  0.301  0.167 -0.004  0.059
## 60%  0.325  0.502  0.414  0.320  0.309
## 70%  0.727  0.725  0.721  0.662  0.651
## 80%  1.376  0.906  1.078  1.108  1.206
## 90%  3.078  1.135  1.831  2.378  2.350
## 95%  6.314  1.465  2.662  4.459  4.976
```

It seems $\sigma = 16$ performs better according to quantile examination.

P [245] 8.

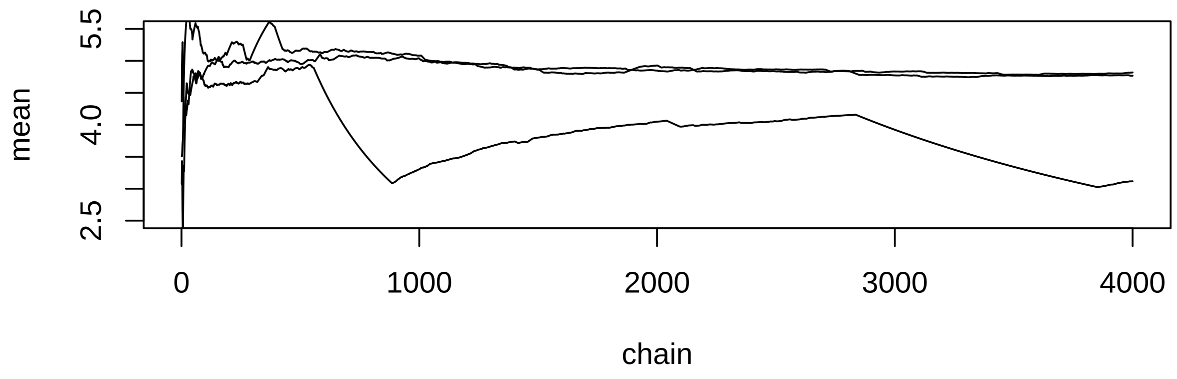
(1). By definition, the density of combined normal distribution $0.2N(0,1) + 0.8N(5,1)$ is

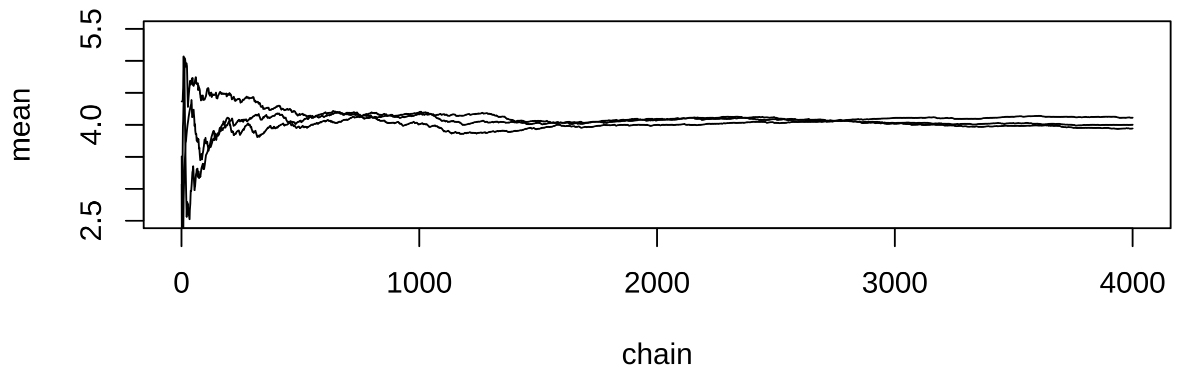
$$f(x) \propto 0.2e^{-\frac{x^2}{2}} + 0.8e^{-\frac{(x-5)}{2}}$$

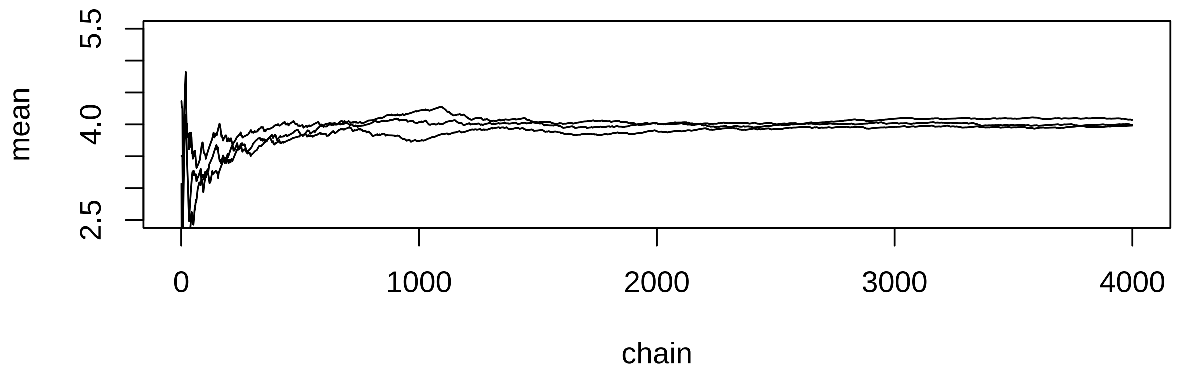
If we use independent Metropolis sampling method through normal distribution $Y \sim N(X_t, \sigma^2)$, it's reasonable to fix $X_t = 4$ and choose σ from $\{1, 3, 6, 10\}$.

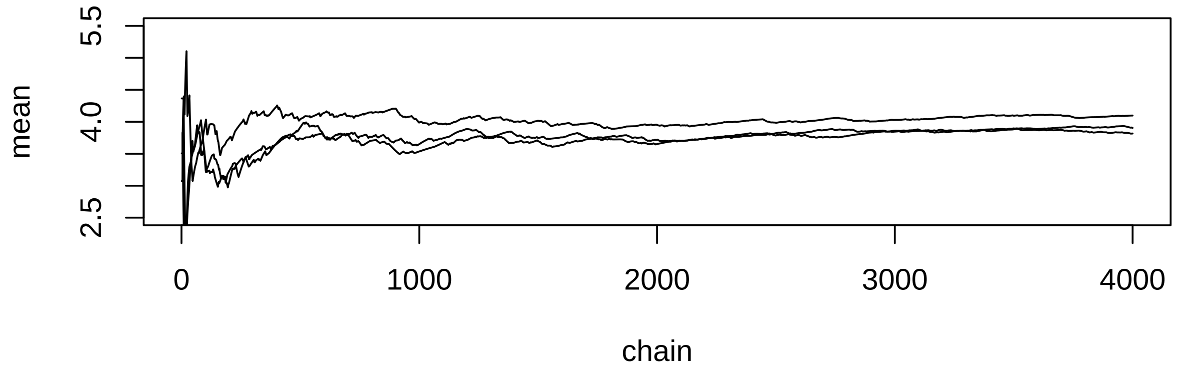
```
idm.245p8 <- function (sigma,x0,N){
  # sigma: step size
  # x0: initial value
  # N: length of chain
  x <- numeric(N)
  x[1] <- x0
  u <- runif(N)
  k <- 0
  for (i in 2:N){
    y <- rnorm(1,4,sigma)
    fy <- 0.2*exp(-(y^2)/2)+0.8*exp(-((y-5)^2)/2)
    gx <- exp(-(x[i-1]-4)^2/(2*sigma^2))
    fx <- 0.2*exp(-(x[i-1]^2)/2)+0.8*exp(-((x[i-1]-5)^2)/2)
    gy <- exp(-(y-4)^2/(2*sigma^2))
    if (u[i]<(fy*gx)/(fx*gy))
```

```r
      x[i] <- y
    else {
      x[i] <- x[i-1]
      k <- k+1
    }
  }
  return (list(x=x,k=k))
}

sigma <- c(1,3,6,10)
N <- 4000
x0 <- runif(3,0,5)
nseq <- 1:N
rejr <- numeric(4)
expc <- numeric(4)
var.GR <- numeric(4)
for (i in 1:4) {
  rw1 <- idm.245p8(sigma[i],x0[1],N)
  rw2 <- idm.245p8(sigma[i],x0[2],N)
  rw3 <- idm.245p8(sigma[i],x0[3],N)

  cmean1 <- numeric(N)
  for (j in 1:N)
    cmean1[j] <- mean(rw1$x[1:j])
  cmean2 <- numeric(N)
  for (j in 1:N)
    cmean2[j] <- mean(rw2$x[1:j])
  cmean3 <- numeric(N)
  for (j in 1:N)
    cmean3[j] <- mean(rw3$x[1:j])

  plot(nseq,cmean1,type="l",xlab='chain',ylab='mean',ylim=c(2.5,5.5))
  lines(nseq,cmean2)
  lines(nseq,cmean3)

  rw <- rbind(rw1$x[1500:N],rw2$x[1500:N],rw3$x[1500:N])
  rmean <- apply(rw,1,mean)
  expc[i] <- mean(rmean)
  rvar <- apply(rw,1,var)
  var.wth <- mean(rvar)
  var.bt <- var(rmean)
  var.GR[i] <- var.wth*(N-1500-1)/(N-1500) + var.bt
  rejr[i] <- (rw1$k+rw2$k+rw3$k)/(3*N)
}
```


mean
chain
2.5
4.0
5.5
0
1000
2000
3000
4000


mean
chain
2.5
4.0
5.5
0
1000
2000
3000
4000


mean
chain
2.5
4.0
5.5
0
1000
2000
3000
4000



```
print(round(cbind(rejr,expc,var.GR),3))
```

```
##       rejr  expc var.GR
## [1,] 0.578 4.069  4.841
## [2,] 0.473 4.024  4.989
## [3,] 0.681 4.035  4.876
## [4,] 0.799 4.010  4.945
```

It seems of $\sigma = 3$ the convergence rate is the fastest and reject ratio is the lowest, while the estimated mean and variance can be found in the second row of the previous table.

(2). If the proposal distribution is substituted by Cauchy distribution, the performance of MCMC should be improved where the improvement should be ascribed to the heavy-tail property of Cauchy distribution.

P[246] 11.

(1). The prior of λ

$$\pi(\lambda) \propto e^{-\lambda}$$

and the prior for single $\theta_i|\lambda$

$$\pi(\theta_i|\lambda) \propto e^{-\lambda(\theta_i)}$$

the marginal distribution of $\boldsymbol{\theta}$

$$\begin{aligned}\pi(\boldsymbol{\theta}) &= \int_\Lambda \pi(\boldsymbol{\theta}|\lambda)\pi(\lambda)d\lambda \\ &= \int_0^\infty \lambda^k e^{-\lambda\sum_1^k \theta_i} \cdot e^{-\lambda} d\lambda \\ &\propto (1+\sum_1^k \theta_i)^{-(1+k)}\end{aligned}$$

(2).

$$\begin{aligned}\pi(\boldsymbol{\theta}|\lambda,\mathbf{x}) &\propto \pi(\boldsymbol{\theta}|\lambda)p(\mathbf{x}|\boldsymbol{\theta})\\ &\propto e^{-\lambda\sum_1^k\theta_i}\cdot e^{-\sum_1^k\theta_i}\prod_1^k\theta_i^{x_i}\\ &= e^{-(1+\lambda)\sum_1^k\theta_i}\prod_1^k\theta_i^{x_i}\end{aligned}$$

$$\begin{aligned}\pi(\theta_j|\boldsymbol{\theta}_{-j},\lambda,\mathbf{x}) &= \frac{\pi(\boldsymbol{\theta}|\lambda,\mathbf{x})}{\pi(\boldsymbol{\theta}_{-j}|\lambda,\mathbf{x})}\\ &\propto e^{-(1+\lambda)\theta_j}\theta_j^{x_j}\end{aligned}$$

So $\theta_j|\boldsymbol{\theta}_{-j},\lambda,\mathbf{x}\sim\Gamma(x_j+1,\lambda+1)$.

$$\begin{aligned}\pi(\lambda|\boldsymbol{\theta},\mathbf{x}) &\propto \pi(\lambda|\boldsymbol{\theta})\\ &\propto \pi(\boldsymbol{\theta}|\lambda)\cdot\pi(\lambda)\\ &\propto \lambda^k e^{-\lambda\sum_1^k\theta_i}\cdot e^{-\lambda}\\ &= \lambda^k e^{-\lambda(1+\sum_1^k\theta_i)}\end{aligned}$$

So $\lambda|\boldsymbol{\theta},\mathbf{x}\sim\Gamma(1+k,1+\sum_1^k\theta_i)$.

(3). Given $k=10$ and data $x=(3,1,4,2,5,3,2,2,0,4)$,

```
gb.246p11 <- function (datax,x0,N){
  # x0: initial value (\theta_1,...,\theta_10,\lambda)
  # N: length of chain
  if ((length(x0)!=11)|(length(datax)!=10))
    return ()
  x <- array(dim=c(N,11))
  x[1,] <- x0
  for (i in 2:N){
    for (j in 1:10)
      x[i,j] <- rgamma(1,datax[j]+1,x[i-1,11]+1)
    temp <- sum(x[i-1,1:10])+1
    x[i,11] <- rgamma(1,11,temp)
  }
  return (x)
}

x0 <- c(2,2,2,2,2,2,2,2,2,2,1)
datax <- c(3,1,4,2,5,3,2,2,0,4)
N <- 3000
n <- 1000

mc0 <- gb.246p11(datax,x0,N)
mc <- mc0[n:N,]
print(round(apply(mc,2,mean),3))
```

```
##  [1] 2.856 1.461 3.525 2.118 4.304 2.871 2.087 2.130 0.712 3.552 0.424
```

```r
print(round(var(mc),2))
```

```
##       [,1] [,2] [,3] [,4]  [,5]  [,6]  [,7] [,8] [,9] [,10] [,11]
##  [1,] 2.18 0.01 0.16 0.05  0.12  0.14  0.11 0.00 0.03  0.05  0.00
##  [2,] 0.01 1.06 0.00 0.01  0.09  0.01  0.06 0.03 0.03  0.07  0.00
##  [3,] 0.16 0.00 2.58 0.07  0.10  0.16  0.06 0.07 0.07  0.18  0.00
##  [4,] 0.05 0.01 0.07 1.70  0.04  0.03  0.08 0.03 0.01  0.05  0.01
##  [5,] 0.12 0.09 0.10 0.04  3.38  0.03  0.09 0.10 0.03 -0.02  0.01
##  [6,] 0.14 0.01 0.16 0.03  0.03  2.16 -0.01 0.03 0.07  0.13  0.00
##  [7,] 0.11 0.06 0.06 0.08  0.09 -0.01  1.54 0.03 0.00  0.12  0.00
##  [8,] 0.00 0.03 0.07 0.03  0.10  0.03  0.03 1.55 0.03  0.09  0.00
##  [9,] 0.03 0.03 0.07 0.01  0.03  0.07  0.00 0.03 0.53  0.07  0.00
## [10,] 0.05 0.07 0.18 0.05 -0.02  0.13  0.12 0.09 0.07  2.68  0.00
## [11,] 0.00 0.00 0.00 0.01  0.01  0.00  0.00 0.00 0.00  0.00  0.02
```

```r
print(round(cor(mc),2))
```

```
##       [,1] [,2] [,3] [,4]  [,5]  [,6]  [,7] [,8] [,9] [,10] [,11]
##  [1,] 1.00 0.01 0.07 0.03  0.05  0.07  0.06 0.00 0.03  0.02  0.01
##  [2,] 0.01 1.00 0.00 0.01  0.05  0.01  0.05 0.02 0.03  0.04  0.00
##  [3,] 0.07 0.00 1.00 0.04  0.03  0.07  0.03 0.03 0.06  0.07  0.01
##  [4,] 0.03 0.01 0.04 1.00  0.02  0.02  0.05 0.02 0.02  0.02  0.03
##  [5,] 0.05 0.05 0.03 0.02  1.00  0.01  0.04 0.04 0.02 -0.01  0.02
##  [6,] 0.07 0.01 0.07 0.02  0.01  1.00 -0.01 0.02 0.06  0.06  0.00
##  [7,] 0.06 0.05 0.03 0.05  0.04 -0.01  1.00 0.02 0.00  0.06  0.02
##  [8,] 0.00 0.02 0.03 0.02  0.04  0.02  0.02 1.00 0.03  0.04  0.01
##  [9,] 0.03 0.03 0.06 0.02  0.02  0.06  0.00 0.03 1.00  0.05  0.00
## [10,] 0.02 0.04 0.07 0.02 -0.01  0.06  0.06 0.04 0.05  1.00  0.00
## [11,] 0.01 0.00 0.01 0.03  0.02  0.00  0.02 0.01 0.00  0.00  1.00
```