Bayesian Statistical Methods

Partial solutions

Chapter 3: Computational approaches

Jump to probem: 1, 3, 5, 7, 9, 11, 13, 15, 17

(1) MAP: Fast but doesn't quantify uncertainty

Numerical integration: Accurate but fails in high dimensions

CLT: Fast but requires a large sample size

Gibbs: Accurate but requires conjugate priors

M-H: Flexible but requires tuning

(3a) The conjugate prior is $\lambda \sim \text{Gamma}(a,b)$ and the posterior is

$$\begin{aligned} p(\lambda \mid Y) &\propto \left[\prod_{i=1}^{n} f(Y_i \mid \lambda)\right]\pi(\lambda) \\ &\propto \left[\prod_{i=1}^{n} \lambda^{Y_i}\exp(-N_i\lambda)\right]\cdot[\lambda^{a-1}\exp(-\lambda b)] \\ &\propto \lambda^{(\sum_{i=1}^{n} Y_i + a)-1}\exp\left[-\lambda\left(\sum_{i=1}^{n} N_i + b\right)\right] \end{aligned}$$

and so $\lambda \mid Y \sim \text{Gamma}(\sum_{i=1}^n Y_i + a, \sum_{i=1}^n N_i + b)$.

(3b) For $\lambda \in (0,20)$ the posterior is $p(\lambda \mid Y) \propto \prod_{i=1}^n f(Y_i \mid \lambda)$. Therefore the log posterior is a constant plus

$$l(\lambda) = \sum_{i=1}^{n}[-N_i\lambda + Y_i\log(\lambda)].$$

Taking the derivative and setting to zero gives

$$l'(\lambda) = -\sum_{i=1}^{n} N_i + [\sum_{i=1}^{n} Y_i]/\lambda = 0$$

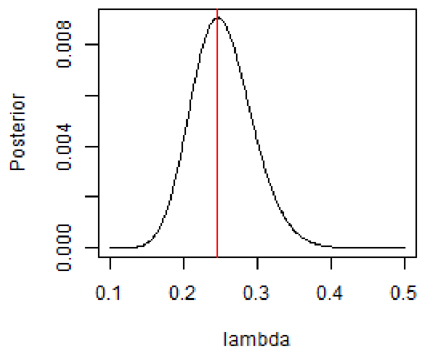
and so the MAP estimate is $\hat{\lambda} = [\sum_{i=1}^n Y_i]/[\sum_{i=1}^n N_i]$ (unless this is greater than 20, in which case the MAP is 20).

(3c)

```
lambda <- seq(0.1,0.5,0.001)
post   <- dpois(12,50*lambda)*dpois(25,100*lambda)
MAP    <- (12+25)/(50+100)
MAP
```

```
## [1] 0.2466667
```

```
plot(lambda,post,type="l",ylab="Posterior")
abline(v=MAP,col=2)
```

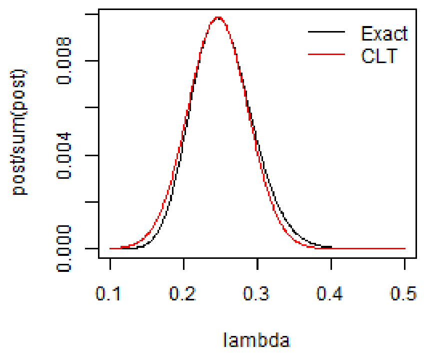


(3d) Taking the second derivative gives

$$\frac{d^2 l(\lambda)}{d\lambda^2} = -[\sum_{i=1}^{n} Y_i]/\lambda^2$$

and so the approximate variance is $\hat{\lambda}^2/[\sum_{i=1}^n Y_i] = [\sum_{i=1}^n Y_i]/[\sum_{i=1}^n N_i]^2$.

```
lambda <- seq(0.1,0.5,0.001)
MAP    <- (12+25)/(50+100)
VAR    <- MAP*MAP/(12+25)
clt    <- dnorm(lambda,MAP,sqrt(VAR))
plot(lambda,post/sum(post),type="l")
lines(lambda,clt/sum(clt),col=2)
legend("topright",c("Exact","CLT"),lty=1,col=1:2,bty="n")
```



(5a) Perhaps Y_i is the number of crimes in year i and a new police policy is put in place in year n so the objective is to test whether the mean number of crimes is affected by the new policy.

(5b) Let $\tau = 1/\sigma^2$.

$$\mu \,|\, \text{rest} \sim \text{Normal}\left(\frac{\tau\sum_{i=1}^{N} Y_i + \tau\sum_{i=n+1}^{n+m}(Y_i - \delta)}{\tau(n+m) + 1/100^2}, \frac{1}{\tau(n+m) + 1/100^2}\right).$$

Similarly,

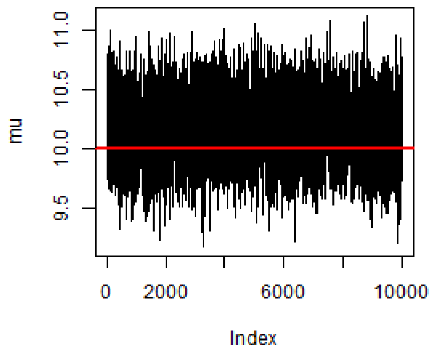
$$\delta \,|\, \text{rest} \sim \text{Normal}\left(\frac{\tau\sum_{i=n+1}^{n+m}(Y_i - \mu)}{\tau m + 1/100^2}, \frac{1}{\tau m + 1/100^2}\right).$$

Finally,

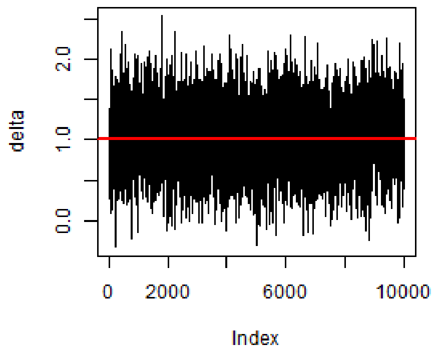
$$\sigma^2 \mid \text{rest} \sim \text{InvGamma}\left(a + (n+m)/2, b + \sum_{i=i}^{n}(Y_i - \mu)^2/2 + \sum_{i=n+1}^{n+m}(Y_i - \mu - \delta)^2/2\right).$$

(5c)

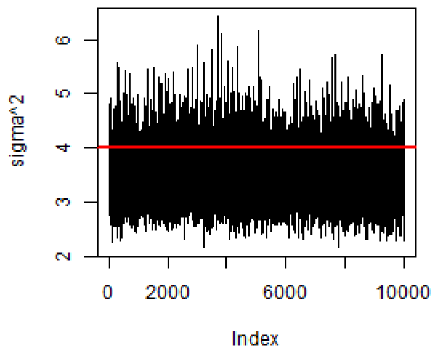
```r
# Generate data
n          <- 50
m          <- 50
mu_true    <- 10
delta_true <- 1
sigma_true <- 2
set.seed(919)
Y1         <- rnorm(n,mu_true,sigma_true)
Y2         <- rnorm(m,mu_true+delta_true,sigma_true)
Y          <- c(Y1,Y2)

# Prep for Gibbs sampling
S       <- 10000   # number of iterations
pri_var <- 100^2   # priors
eps     <- 0.01
keep    <- matrix(0,S,3)
mu      <- mean(Y) # initial values
delta   <- 0
sigma2  <- var(Y)

# Go!
for(iter in 1:S){
  # mu
  prec  <- (n+m)/sigma2 + 1/pri_var
  mn    <- sum(Y[1:n])/sigma2 + sum(Y[1:m+n]-delta)/sigma2
  mu    <- rnorm(1,mn/prec,1/sqrt(prec))

  # delta
  prec  <- m/sigma2 + 1/pri_var
  mn    <- sum(Y[1:m+n]-mu)/sigma2
  delta <- rnorm(1,mn/prec,1/sqrt(prec))

  # sigma2
  SS     <- sum((Y[1:n]-mu)^2) +
            sum((Y[1:m+n]-mu-delta)^2)
  sigma2 <- 1/rgamma(1,eps+(n+m)/2,SS/2+eps)

  keep[iter,] <- c(mu,delta,sigma2)
}

plot(keep[,1],type="l",ylab="mu")
abline(mu_true,0,lwd=2,col=2)
```



```r
plot(keep[,2],type="l",ylab="delta")
abline(delta_true,0,lwd=2,col=2)
```

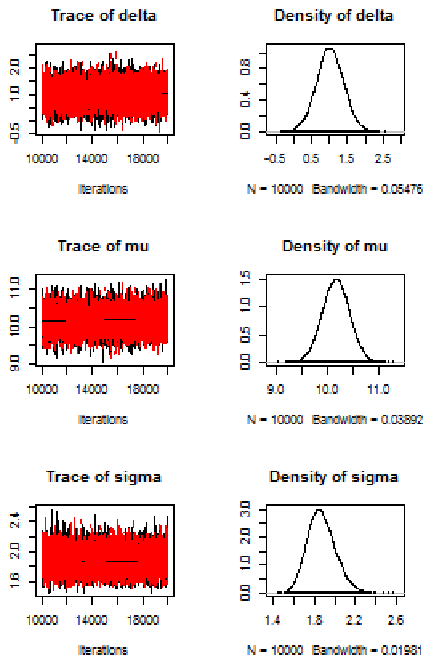


```r
plot(keep[,3],type="l",ylab="sigma^2")
abline(sigma_true^2,0,lwd=2,col=2)
```



(5d)

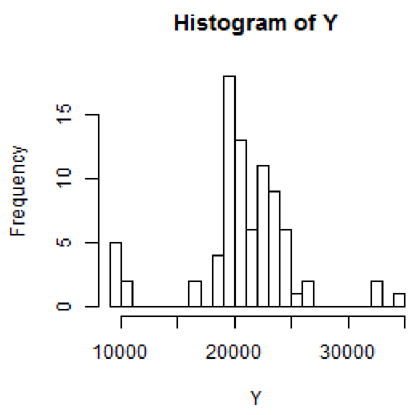
```
 library(rjags)
 data <- list(n=n,m=m,Y1=Y[1:n],Y2=Y[n+1:m])

 model_string <- textConnection("model{

  # Likelihood
  for(i in 1:n){
    Y1[i] ~ dnorm(mu,tau)
  }
  for(i in 1:m){
    Y2[i] ~ dnorm(mu+delta,tau)
  }

  # Priors
  mu    ~  dnorm(0, 0.0001)
  delta ~  dnorm(0, 0.0001)
  tau   ~  dgamma(0.1, 0.1)
  sigma <- 1/sqrt(tau)
}")

inits <- list(mu=mean(Y),delta=0,tau=1/var(Y))
model <- jags.model(model_string,data = data, inits=inits, n.chains=2,quiet=TRUE)
update(model, 10000, progress.bar="none")
params  <- c("mu","delta","sigma")
samples <- coda.samples(model,
           variable.names=params,
           n.iter=10000, progress.bar="none")
plot(samples)
```



Convergence looks great and the posterior approximations in parts c and d are very similar.

(7a)

```
library(MASS)
Y<-galaxies
n<-length(Y)
hist(Y,breaks=25)
```



```
mean(Y);var(Y)
```

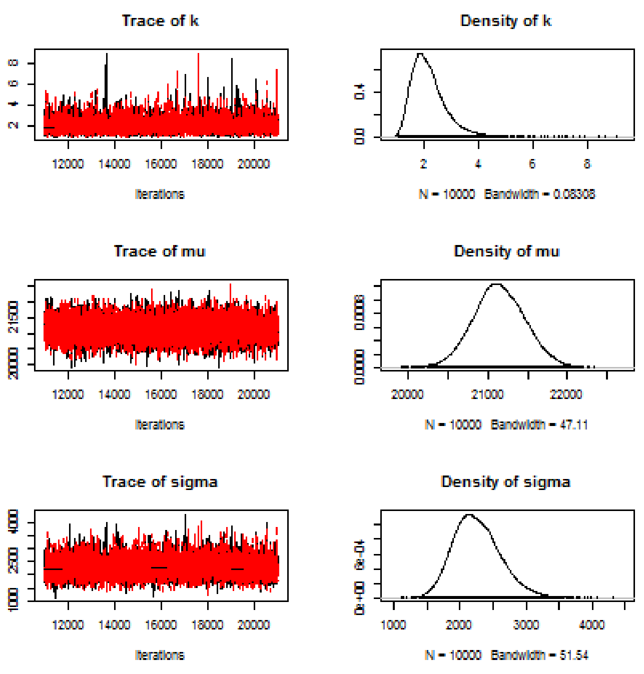
```
## [1] 20828.17
```

```
## [1] 20827887
```

One reasonable initial value is to start at the approximately Gaussian model with $k = 30$ and mean and variance set to the sample mean and variance (above).

(7b)

```
 library(rjags)
 data <- list(n=n,Y=Y)

 model_string <- textConnection("model{

  # Likelihood
  for(i in 1:n){
    Y[i] ~ dt(mu,tau,k)
  }

  # Priors
  mu    ~  dnorm(0, 0.0000000001)
  tau   ~  dgamma(0.01, 0.01)
  k     ~  dunif(1, 30)
  sigma <- 1/sqrt(tau)
}")

inits <- list(mu=mean(Y),tau=1/var(Y),k=30)
model <- jags.model(model_string,data = data, inits=inits, n.chains=2,quiet=TRUE)
update(model, 10000, progress.bar="none")
params  <- c("mu","sigma","k")
samples <- coda.samples(model,
           variable.names=params,
           n.iter=10000, progress.bar="none")
plot(samples)
```



```r
summary(samples)
```

```
##
## Iterations = 11001:21000
## Thinning interval = 1
## Number of chains = 2
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##             Mean      SD Naive SE Time-series SE
## k          2.158   0.682 0.004823         0.0111
## mu     21140.856 323.627 2.288385         2.9208
## sigma   2262.255 359.784 2.544059         4.9864
##
## 2. Quantiles for each variable:
##
##             2.5%       25%       50%       75%     97.5%
## k          1.247     1.701     2.034     2.462     3.792
## mu     20515.753 20925.426 21133.193 21357.048 21788.130
## sigma   1645.335  2008.107  2230.688  2480.372  3053.795
```

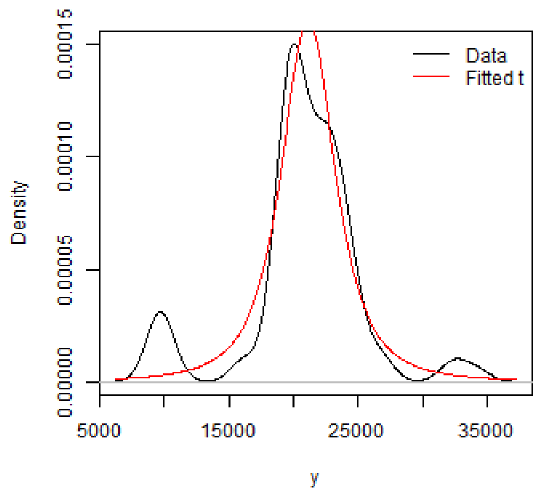
```r
mn <- summary(samples)$statistics[,1]
mn
```

```
##            k           mu        sigma
##     2.158073 21140.856281  2262.254839
```

Convergence looks good.

(7c) The code below compares the fitted t PDF with the kernel density estimate (an alternative to the histogram) of the data.

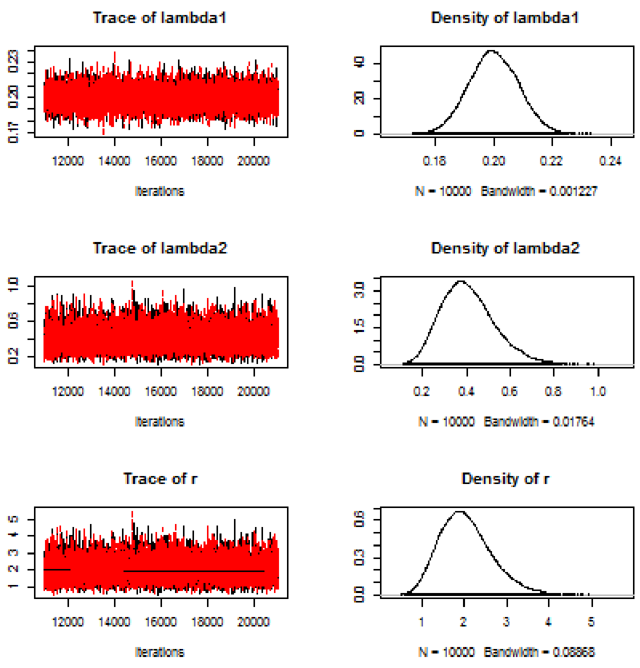
```r
 kde <- density(Y)
 pdf <- dt((kde$x-mn[2])/mn[3],df=mn[1])
 pdf <- sum(kde$y)*pdf/sum(pdf)  # This makes both plots on the same scale

 plot(kde,xlab="y",ylab="Density",main="")
 lines(kde$x,pdf,col=2)
 legend("topright",c("Data","Fitted t"),lty=1,col=1:2,bty="n")
```



The t density misses the humps on the left and right and isn't a great fit.

(9)

```
 library(rjags)
 data <- list(N1=2820,Y1=563,N2=27,Y2=10)

 model_string <- textConnection("model{

  # Likelihood
   Y1 ~ dpois(N1*lambda1)
   Y2 ~ dpois(N2*lambda2)

  # Priors
   lambda1 ~  dunif(0, 10)
   lambda2 ~  dunif(0, 10)
   r       <- lambda2/lambda1
}")

inits <- list(lambda1=0.1,lambda2=0.2)
model <- jags.model(model_string,data = data, inits=inits, n.chains=2,quiet=TRUE)
update(model, 10000, progress.bar="none")
params  <- c("lambda1","lambda2","r")
samples <- coda.samples(model,
           variable.names=params,
           n.iter=10000, progress.bar="none")
plot(samples)
```



```r
summary(samples)
```

```
##
## Iterations = 11001:21000
## Thinning interval = 1
## Number of chains = 2
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean       SD  Naive SE Time-series SE
## lambda1 0.2000 0.008387 5.931e-05      7.531e-05
## lambda2 0.4079 0.122202 8.641e-04      1.148e-03
## r       2.0428 0.617442 4.366e-03      5.730e-03
##
## 2. Quantiles for each variable:
##
##           2.5%    25%    50%    75%  97.5%
## lambda1 0.1839 0.1943 0.1998 0.2057 0.2166
## lambda2 0.2044 0.3200 0.3953 0.4816 0.6819
## r       1.0171 1.6016 1.9809 2.4141 3.4344
```

```r
effectiveSize(samples)
```

```
##  lambda1  lambda2        r
## 12402.53 11332.00 11627.71
```

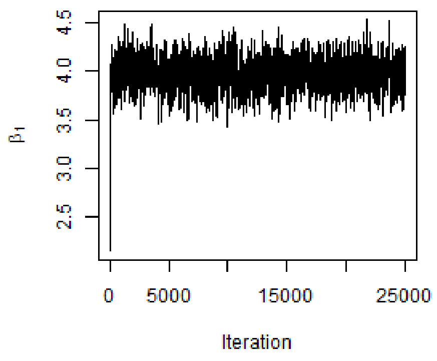
```r
gelman.diag(samples)
```

```
## Potential scale reduction factors:
##
##         Point est. Upper C.I.
## lambda1          1          1
## lambda2          1          1
## r                1          1
##
## Multivariate psrf
##
## 1
```

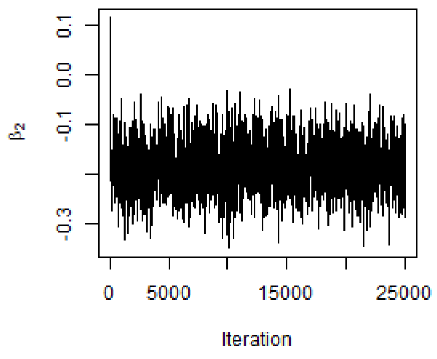
The trace plots look great, the effective sample sizes are all large (over 1,000), and the Gelman-Rubin statistics are 1.0. Therefore, the chains have clearly converged.

(11) The code below uses Gaussian candidate distributions with standard deviation 0.2 and 0.05 (selected by trial and error to give acceptance probability around 0.4 for each parameter).

```
Y      <- c(64,13,33,18,30,20)
t      <- 1:6

S      <- 25000
beta   <- c(2,0)
cansd  <- c(0.2,0.05)
keep   <- matrix(0,S,2)

post   <- function(Y,t,beta,pri.sd=10){
    mn <- exp(beta[1] + t*beta[2])
    l  <- prod(dpois(Y,mn))
    p  <- prod(dnorm(beta,0,pri.sd))
return(l*p)}

for(iter in 1:S){
   for(j in 1:2){
     can     <- beta
     can[j]  <- rnorm(1,beta[j],cansd[j])
     R       <- post(Y,t,can)/post(Y,t,beta)
     if(runif(1)<R){
        beta <- can
     }
   }
   keep[iter,] <- beta
}

plot(keep[,1],type="l",ylab=expression(beta[1]),xlab="Iteration")
```



```
plot(keep[,2],type="l",ylab=expression(beta[2]),xlab="Iteration")
```



```
acc_rate <- colMeans(keep[-1,]!=keep[-S,])
acc_rate
```

```
## [1] 0.4033761 0.4568983
```

The trace plots look great and the acceptance rates are around 0.4, which is acceptable.

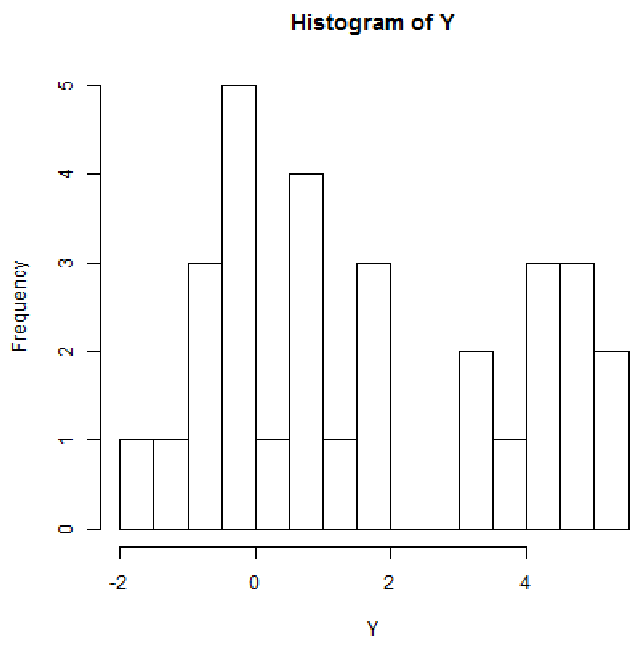
(13a) The joint distribution of B and Y is $f(Y,B) = g(Y|B)h(B)$ where $g(Y|B) = \phi(Y - B\theta)$ is the normal PDF and h is the Bernoulli PMF. The marginal distribution of Y is then

$$f(Y) = \sum_{b=0}^{1} g(Y|B)h(B) = \phi(Y)\frac{1}{2} + \phi(Y-\theta)\frac{1}{2}$$

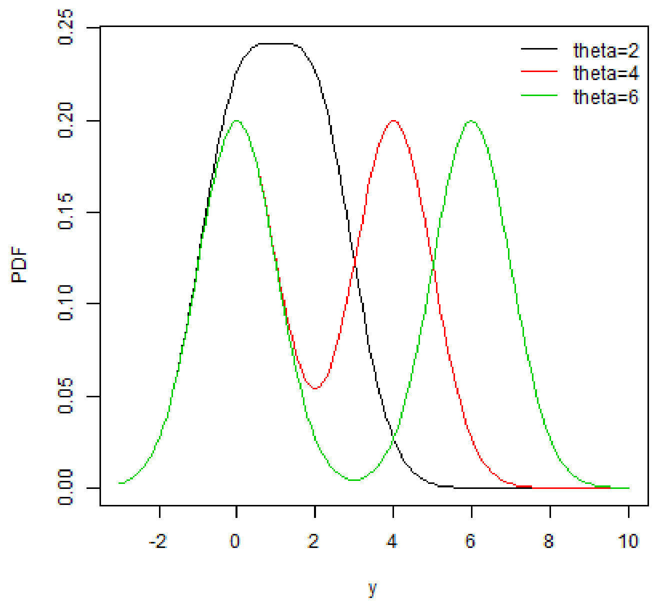
as desired.

(13b)

```
set.seed(27695)
theta_true <- 4
n          <- 30
B          <- rbinom(n,1,0.5)
Y          <- rnorm(n,B*theta_true,1)
hist(Y,breaks=25)
```



```
y           <- seq(-3,10,0.1)
plot(y,0.5*dnorm(y,0,1) + 0.5*dnorm(y,2,1),type="l",ylab="PDF")
lines(y,0.5*dnorm(y,0,1) + 0.5*dnorm(y,4,1),col=2)
lines(y,0.5*dnorm(y,0,1) + 0.5*dnorm(y,6,1),col=3)
legend("topright",c("theta=2","theta=4","theta=6"),
                 lty=1,col=1:3,bty="n")
```



(13c)

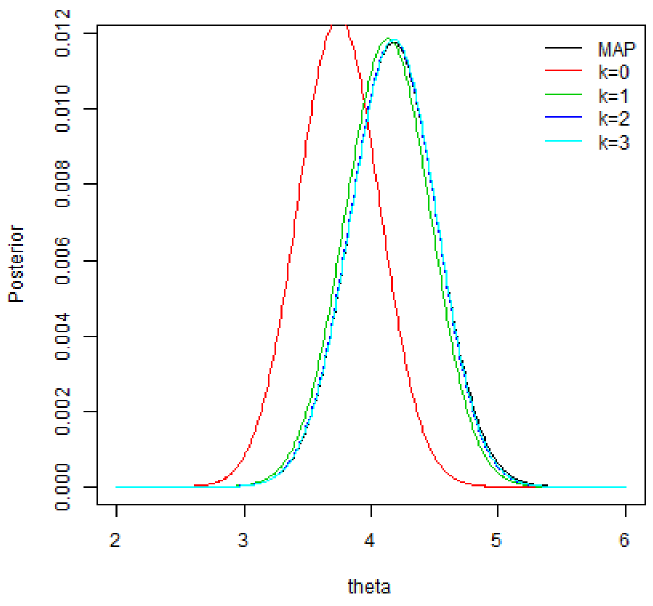
```
library(stats4)
nlp <- function(theta,Y){
   like <- 0.5*dnorm(Y,0,1)+
           0.5*dnorm(Y,theta,1)
   prior <- dnorm(theta,0,10)
   neg_log_post <- -sum(log(like))-log(prior)
return(neg_log_post)}

map_est <- mle(nlp,start=list(theta=1),
                   fixed=list(Y=Y))
sd      <- sqrt(vcov(map_est))
map_est; sd
```

```
##
## Call:
## mle(minuslogl = nlp, start = list(theta = 1), fixed = list(Y = Y))
##
## Coefficients:
##       theta          Y1          Y2          Y3          Y4          Y5
##  4.17919603 -1.77844991  1.60305490  0.63529137  4.67157916 -0.43544347
##          Y6          Y7          Y8          Y9         Y10         Y11
## -0.12584690 -0.38585384  4.66977750  1.12469067  1.79842288  3.85616261
##         Y12         Y13         Y14         Y15         Y16         Y17
##  3.12904919 -1.38070620  4.07364670  3.35284203  0.52304572 -0.75516014
##         Y18         Y19         Y20         Y21         Y22         Y23
## -0.52569447  0.76633658  5.04733582  0.11272642  4.78594654  1.69632947
##         Y24         Y25         Y26         Y27         Y28         Y29
##  4.05891960  0.58024782  5.23736037  4.44583893 -0.09848674 -0.36530670
##         Y30
## -0.80020385
##            theta
## theta 0.3389466
```

```
map     <- 4.17919603
```

(13d)

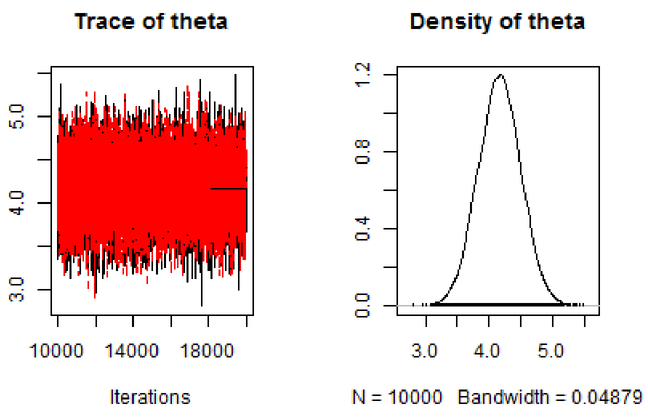
```
posterior <- function(theta,Y,k){
   post <- dnorm(theta,0,sqrt(10^k))
   for(i in 1:length(Y)){
      post<-post*(0.5*dnorm(Y[i],0,1)+
                  0.5*dnorm(Y[i],theta,1))
   }
return(post/sum(post))}

theta <- seq(2,6,0.01)
map   <- dnorm(theta,map,sd)
plot(theta,map/sum(map),type="l",ylab="Posterior")
lines(theta,posterior(theta,Y,0),col=2)
lines(theta,posterior(theta,Y,1),col=3)
lines(theta,posterior(theta,Y,2),col=4)
lines(theta,posterior(theta,Y,3),col=5)
legend("topright",c("MAP","k=0","k=1","k=2","k=3"),
       col=1:5,lty=1,bty="n")
```



Even though an improper prior is problematic in this case, the normal prior with large variance performs similar to the Gaussian approximation.

(13e)

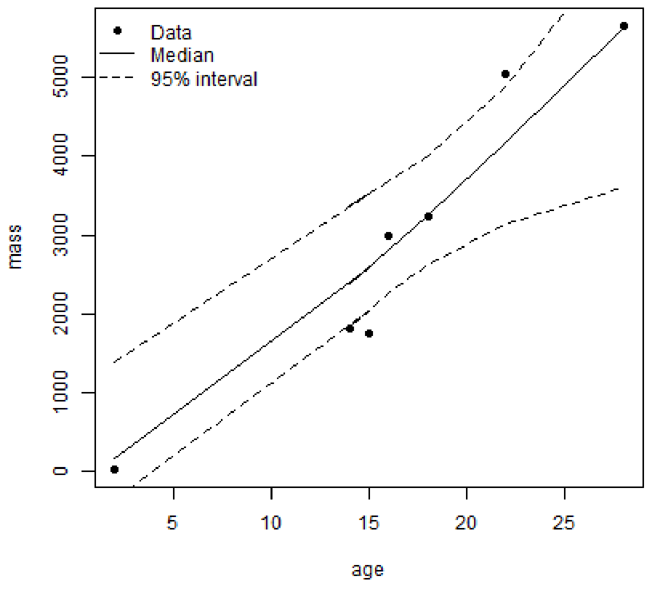
```
 library(rjags)
 data <- list(n=n,Y=Y)

 model_string <- textConnection("model{

  # Likelihood
  for(i in 1:n){
    Y[i] ~ dnorm(B[i]*theta,1)
    B[i] ~ dbern(0.5)
  }

  # Priors
  theta ~  dnorm(0, 0.01)
}")

inits <- list(theta=1)
model <- jags.model(model_string,data = data, inits=inits, n.chains=2,quiet=TRUE)
update(model, 10000, progress.bar="none")
params  <- c("theta")
samples <- coda.samples(model,
           variable.names=params,
           n.iter=10000, progress.bar="none")
plot(samples)
```



Convergence looks good and the results are similar to part d.

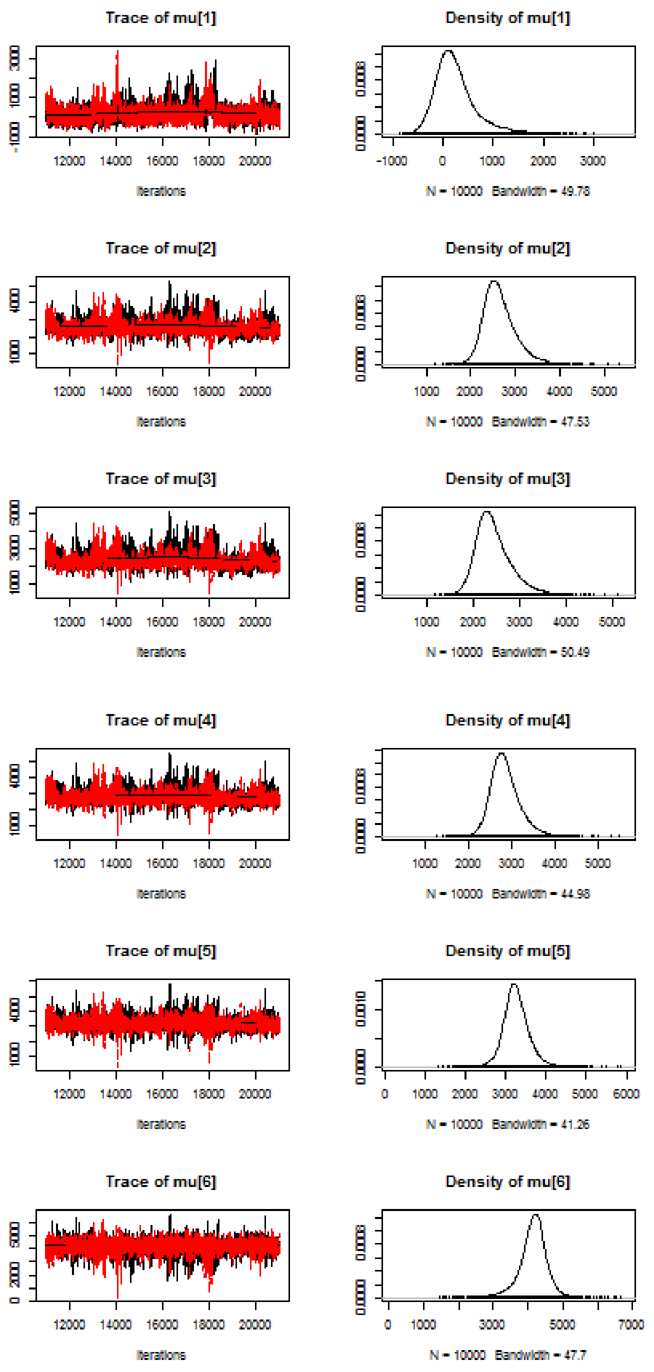
(15a)

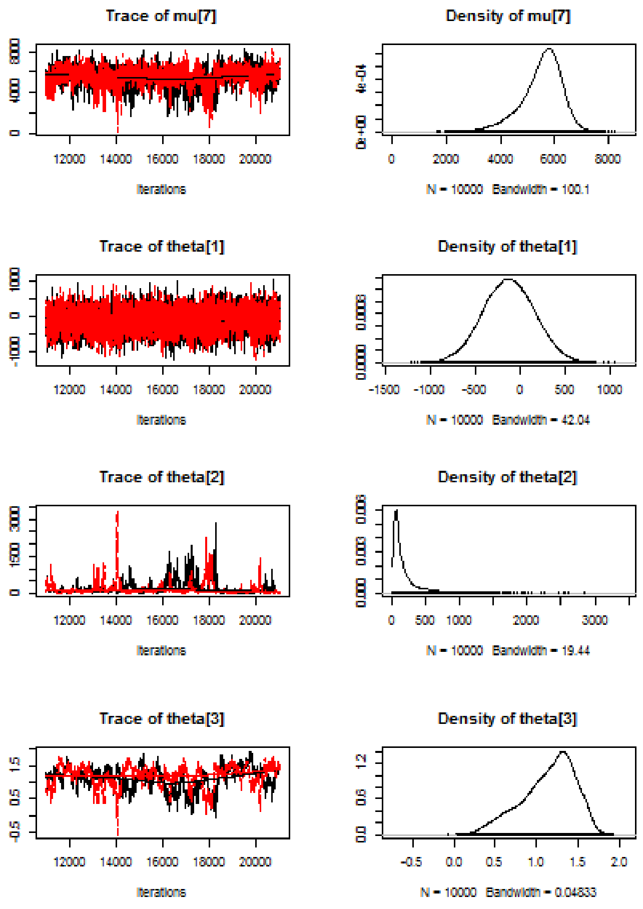
```r
 library(rjags)
mass <- c(29.9, 1761, 1807, 2984, 3230, 5040, 5654)
age  <- c(2, 15, 14, 16, 18, 22, 28)
n    <- length(age)
data <- list(mass=mass,age=age,n=n)

model_string <- textConnection("model{

  # Likelihood
   for(i in 1:n){
     mass[i]  ~ dnorm(mu[i],tau)
     mu[i]   <- theta[1] + theta[2]*pow(age[i],theta[3])
   }

  # Priors
   theta[1] ~ dnorm(0, 0.00001)
   theta[2] ~ dunif(0, 10000)
   theta[3] ~ dnorm(0,1)
   tau      ~ dgamma(0.01,0.01)
}")

model  <- jags.model(model_string,data = data, n.chains=2,quiet=TRUE)
update(model, 10000, progress.bar="none")
params  <- c("mu","theta")
samples <- coda.samples(model,
           variable.names=params,
           n.iter=10000, progress.bar="none")

q <- summary(samples)$quantiles

plot(age,mass,pch=19)
lines(age,q[1:n,3])         # Posterior medians
lines(age,q[1:n,1],lty=2)  # Posterior 95% interval
lines(age,q[1:n,5],lty=2)
legend("topleft",c("Data","Median","95% interval"),
       lty=c(NA,1,2),pch=c(19,NA,NA),bty="n")
```



(15b)

```r
plot(samples)
```


Trace of mu[1]
Density of mu[1]
Iterations
N = 10000 Bandwidth = 49.78
Trace of mu[2]
Density of mu[2]
Iterations
N = 10000 Bandwidth = 47.53
Trace of mu[3]
Density of mu[3]
Iterations
N = 10000 Bandwidth = 50.49
Trace of mu[4]
Density of mu[4]
Iterations
N = 10000 Bandwidth = 44.96
Trace of mu[5]
Density of mu[5]
Iterations
N = 10000 Bandwidth = 41.26
Trace of mu[6]
Density of mu[6]
Iterations
N = 10000 Bandwidth = 47.7



```
summary(samples)
```

```
##
## Iterations = 11001:21000
## Thinning interval = 1
## Number of chains = 2
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean       SD Naive SE Time-series SE
## mu[1]     256.082 435.2737 3.077850       33.62855
## mu[2]    2639.986 371.5540 2.627284       30.46857
## mu[3]    2437.227 388.9419 2.750234       36.94359
## mu[4]    2845.571 357.2971 2.526472       24.12871
## mu[5]    3264.930 345.8699 2.445669       10.73452
## mu[6]    4134.656 430.8688 3.046702       13.72068
## mu[7]    5510.568 794.6578 5.619079       70.75311
## theta[1] -121.491 287.4275 2.032419        4.98096
## theta[2]  215.479 292.6183 2.069124       26.57504
## theta[3]    1.139   0.3305 0.002337        0.04447
##
## 2. Quantiles for each variable:
##
##               2.5%       25%      50%      75%   97.5%
## mu[1]    -363.8349  -23.5302  178.593  432.571 1386.68
## mu[2]    2041.2965 2397.3442 2590.581 2832.836 3521.57
## mu[3]    1829.0203 2176.9567 2377.629 2639.582 3372.65
## mu[4]    2257.2552 2618.8173 2806.125 3030.927 3669.95
## mu[5]    2622.9779 3069.7937 3246.819 3447.838 4012.44
## mu[6]    3138.4372 3942.2632 4173.945 4379.307 4875.10
## mu[7]    3591.0847 5116.0648 5635.422 6032.957 6754.45
## theta[1] -686.4429 -316.9708 -121.804   73.388  437.62
## theta[2]   26.0520   59.4941  108.357  237.609 1083.00
## theta[3]    0.4038    0.9362    1.196    1.381    1.64
```

```
effectiveSize(samples)
```

```
##      mu[1]      mu[2]      mu[3]      mu[4]      mu[5]      mu[6] 
##  172.45780  156.28386  115.72326  234.80936 1095.44094 1043.56345 
##      mu[7]   theta[1]   theta[2]   theta[3] 
##  125.61708 3515.13968  126.57898   55.19019
```

```
gelman.diag(samples,multivariate=F)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## mu[1]          1.01       1.03
## mu[2]          1.01       1.05
## mu[3]          1.01       1.06
## mu[4]          1.01       1.04
## mu[5]          1.00       1.02
## mu[6]          1.00       1.00
## mu[7]          1.01       1.03
## theta[1]       1.00       1.00
## theta[2]       1.01       1.04
## theta[3]       1.02       1.07
```

Convergence is fine for θ_1, but poor for θ_2 and θ_3 which have effective sample size less than 100.

(15c) One approach to improving convergence is to simply run the chains longer. This will likely work here because convergence isn't hopeless and the code is fast. A second approach is to simplify the model. The log-linear model

$$\log(mass) = \beta_0 + \beta_1 \log(age) + \epsilon$$

is one possibility. Finally, θ_2 and θ_3 have posterior correlation -0.88 and so updating them in a block might improve convergence.

(17a) Convergence could be slow because there are more parameters than observations and so it is likely that not all parameters are identifiable.

(17b)

```r
library(rjags)
mod <- "model{

  # Likelihood
   for(i in 1:n){
     Y[i]   ~ dnorm(mu[i],tau[i])
     mu[i]  ~ dnorm(0,theta[1])
     tau[i] ~ dgamma(theta[2],theta[3])
   }

  # Priors
   theta[1] ~ dgamma(eps, eps)
   theta[2] ~ dgamma(eps, eps)
   theta[3] ~ dgamma(eps, eps)
}"

params  <- c("theta")

model_string <- textConnection(mod)
data  <- list(Y=1:5,n=5,eps=0.1)
model <- jags.model(model_string,data = data, n.chains=2,quiet=TRUE)
update(model, 10000, progress.bar="none")
s <- coda.samples(model,
      variable.names=params,
      n.iter=20000, progress.bar="none")
ESS1 <- effectiveSize(s)


model_string <- textConnection(mod)
data  <- list(Y=1:25,n=25,eps=0.1)
model <- jags.model(model_string,data = data, n.chains=2,quiet=TRUE)
update(model, 10000, progress.bar="none")
s <- coda.samples(model,
      variable.names=params,
      n.iter=20000, progress.bar="none")
ESS2 <- effectiveSize(s)


model_string <- textConnection(mod)
data  <- list(Y=1:5,n=5,eps=10)
model <- jags.model(model_string,data = data, n.chains=2,quiet=TRUE)
update(model, 10000, progress.bar="none")
s <- coda.samples(model,
      variable.names=params,
      n.iter=20000, progress.bar="none")
ESS3 <- effectiveSize(s)



model_string <- textConnection(mod)
data  <- list(Y=1:25,n=25,eps=10)
model <- jags.model(model_string,data = data, n.chains=2,quiet=TRUE)
update(model, 10000, progress.bar="none")
s <- coda.samples(model,
      variable.names=params,
      n.iter=20000, progress.bar="none")
ESS4 <- effectiveSize(s)

ESS1 # n = 5  and eps = 0.1
```

```
##  theta[1]  theta[2]  theta[3]
## 1399.1240  268.1794  625.9386
```

```r
ESS2 # n = 25 and eps = 0.1
```

```
##   theta[1]   theta[2]   theta[3]
## 30802.3032    57.2700   195.5023
```

```
ESS3 # n = 5  and eps = 10
```

```
## theta[1] theta[2] theta[3]
## 16901.53 12979.49 27441.12
```

```
ESS4 # n = 35 and eps = 10
```

```
## theta[1] theta[2] theta[3]
## 10533.82 18625.49 29523.03
```

In the first case with small sample size and uninformative prior convergence isn't great. In the second case the sample size increases, but so does the number of parameters, and so convergence remains problematic. However, increasing the prior information improves convergence for the last two fits.