statComp_hw8

凌浩东

12/14/2021

9.6

Suppose the number of animals in four categories are $Y = (y_1, y_2, y_3, y_4)$ respectively. Known that the probabilities of the corresponding multinomial distribution is $(\frac{1}{2} + \frac{\theta}{2}, \frac{1-\theta}{4}, \frac{1-\theta}{4}, \frac{\theta}{4})$. Then the likelihood function is

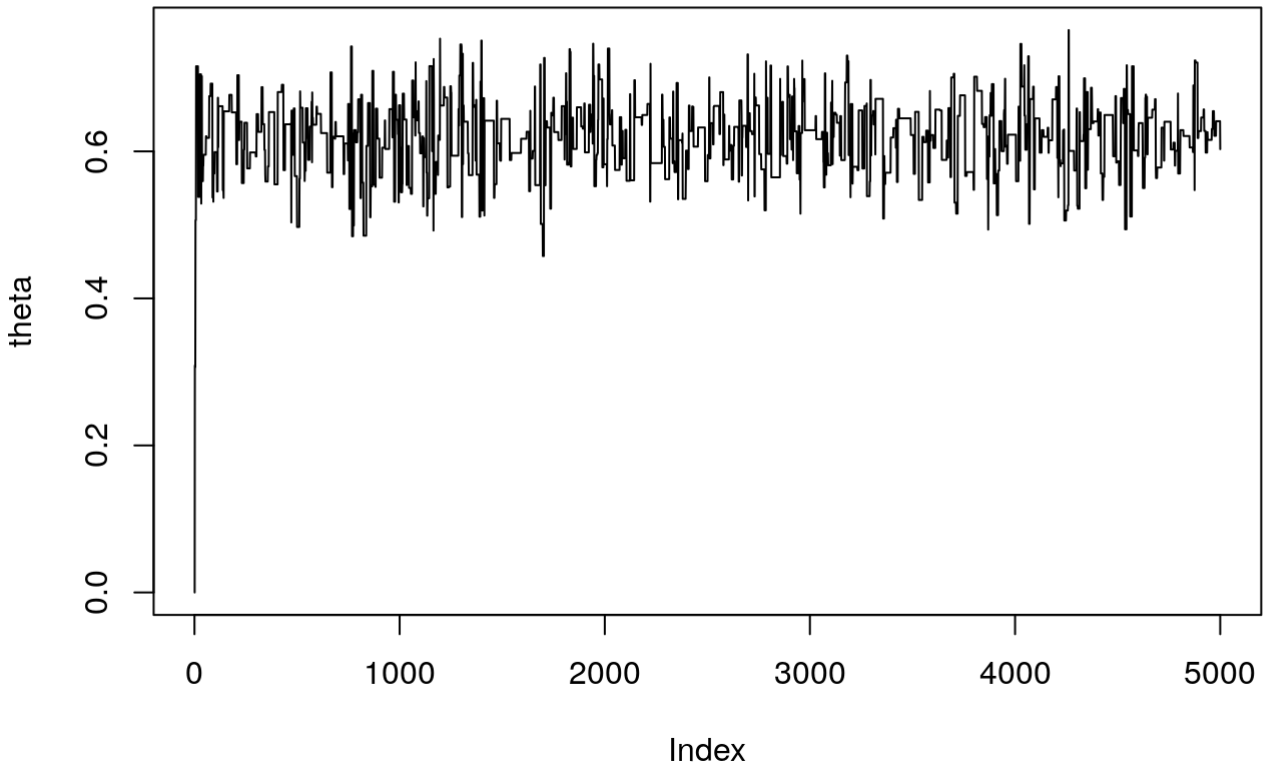
$$L(Y|\theta) = \left(\frac{1}{2} + \frac{\theta}{2}\right)^{y_1} \left(\frac{1-\theta}{4}\right)^{y_2} \left(\frac{1-\theta}{4}\right)^{y_3} \left(\frac{\theta}{4}\right)^{y_4}$$

Suppose the prior is uniform distribution between 0 and 1, i.e. $\pi(\theta) = 1$. Thus the posterior is

$$p(\theta|Y) \propto \pi(\theta) L(Y|\theta) = L(Y|\theta)$$

Now we use the Random Walk sampler to sample date.

```
set.seed(42)
# the length of the chain
N <- 5000
# initiate the chain
x0 <- 0

likelihood <- function(y) {
    if (y < 0 || y >= 1)
      return (0)
    return( (0.5+y/4)^125*(0.25-y/4)^38*(y/4)^34 )
  }

# random walk sampler
rw.Metropolis <- function(sigma, x0, N, f) {
  # the target function
  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] <= (f(y) / f(x[i-1])))
      x[i] <- y
    else {
      x[i] <- x[i-1]
      k <- k + 1
    }
  }
  return(x)
}

x <- rw.Metropolis(0.5,x0,N,likelihood)
plot(x, type = "l",ylab = "theta",main = "random walk markov chain")
```



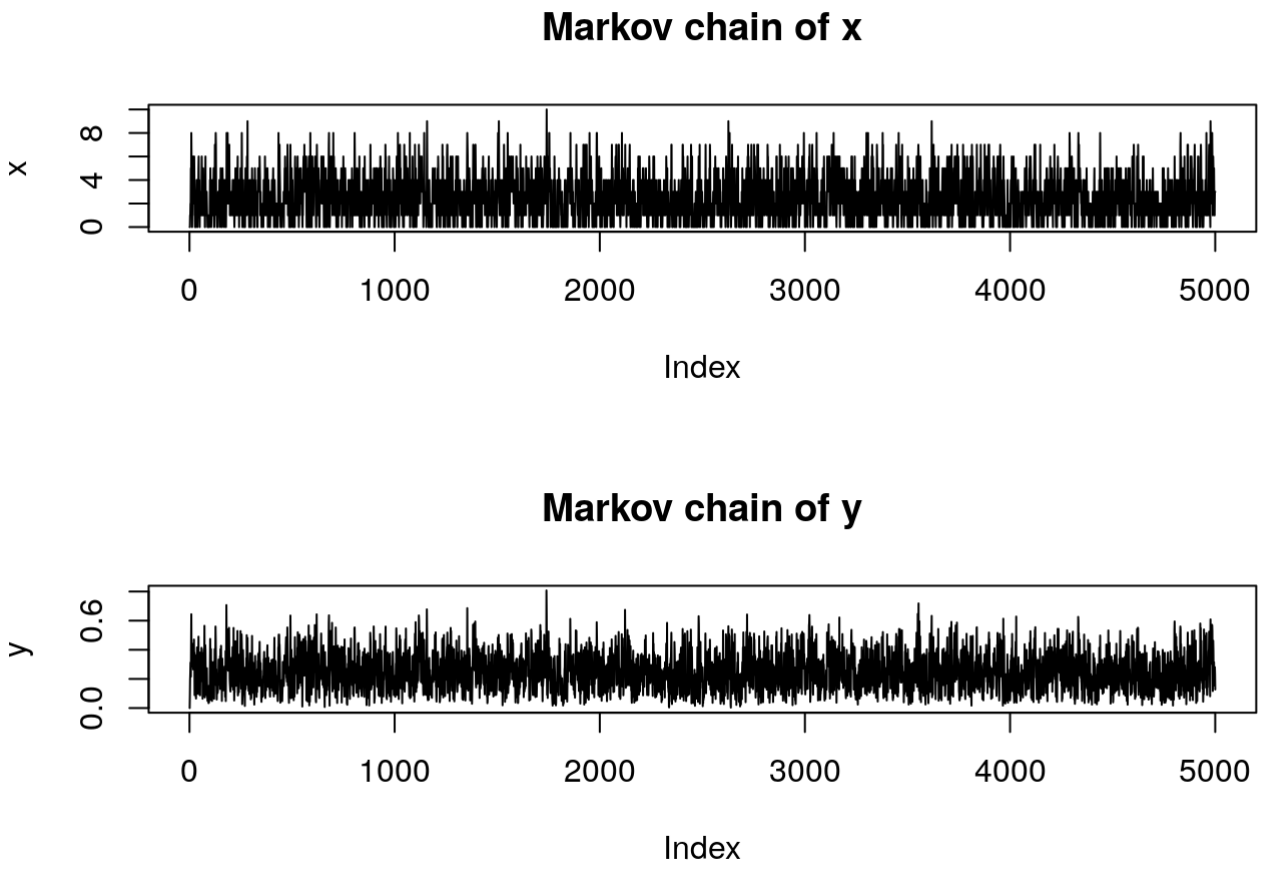

```
burnin <- 1000
mean(x[burnin:N])
```

```
## [1] 0.6195926
```

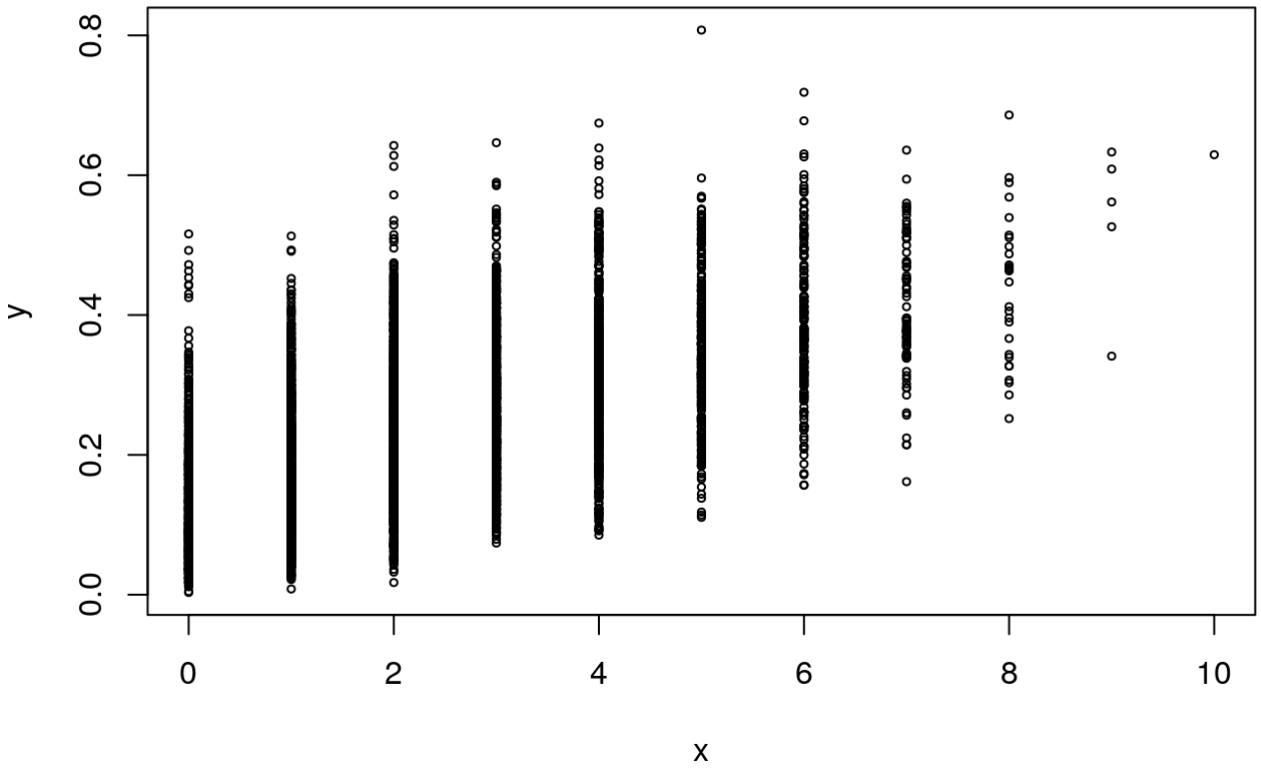
Thus the posterior of θ is 0.620.

9.8

```
set.seed(42)
# initiate constants and parameters
N <- 5000 # length of the chain
burn <- 1000 # burn-in length
X <- matrix(0, N, 2) # the chain
n <- 10
a <- 2
b <- 3

# gibbs sampler
X[1,] <- c(0,0)

for (i in 2:N) {
  x2 <- X[i-1, 2]
  X[i,1] <- rbinom(1, n, x2)
  x1 <- X[i, 1]
  X[i, 2] <- rbeta(1, x1+a, n-x+b)
}

par(mfrow = c(2,1))
plot(X[,1], type = "l", ylab = "x", main = "Markov chain of x")
plot(X[,2], type = "l", ylab = "y", main = "Markov chain of y")
```




```
b <- burn + 1
x <- X[b:N,]
plot(x, main = "joint distribution of x and y",, xlab = "x",
     ylab = "y", cex = 0.5)
```



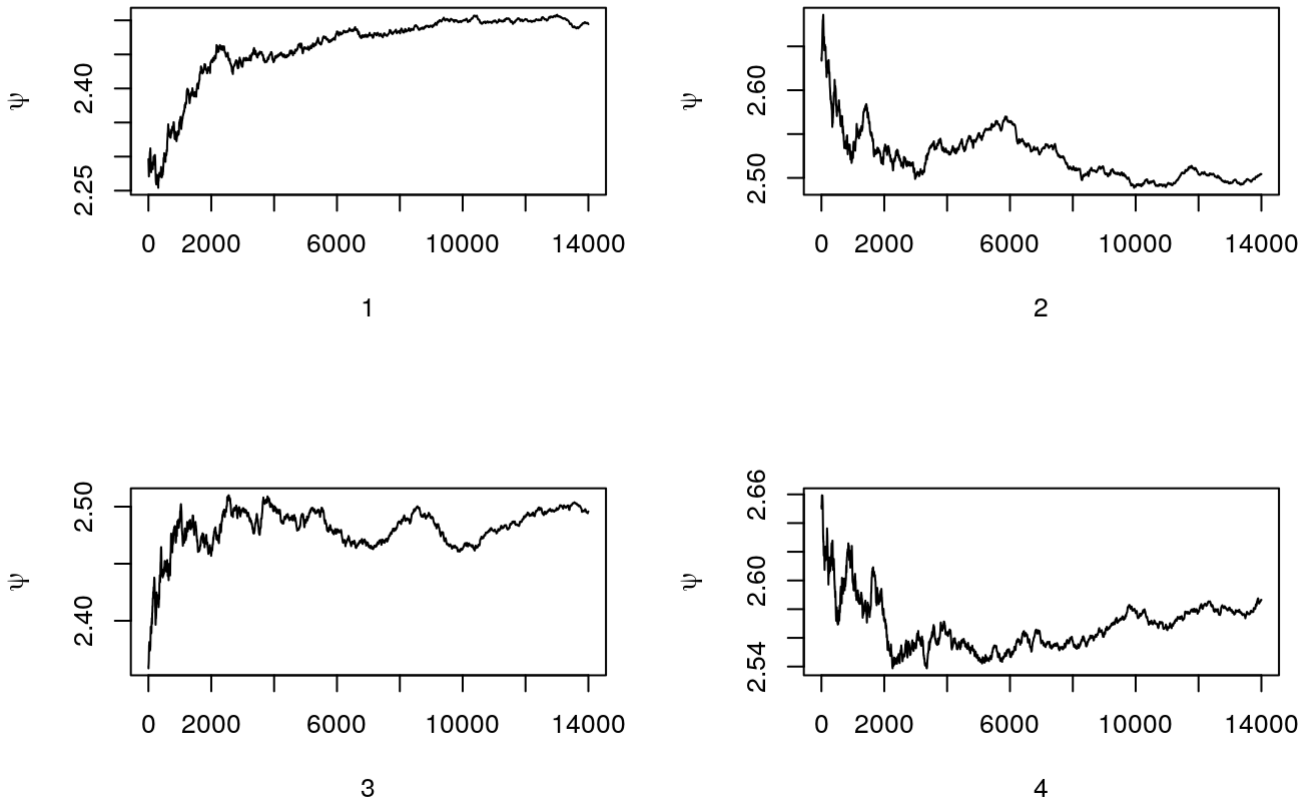
9.10

```r
Gelman.Rubin <- function(psi) {
  # psi[i,j] is the statistic psi(X[i,1:j])
  # for chain in i-th row of X
  psi <- as.matrix(psi)
  n <- ncol(psi)
  k <- nrow(psi)
  psi.means <- rowMeans(psi) #row means
  B <- n * var(psi.means) #between variance est.
  psi.w <- apply(psi, 1, "var") #within variances
  W <- mean(psi.w) #within est.
  v.hat <- W*(n-1)/n + (B/n) #upper variance est.
  r.hat <- v.hat / W #G-R statistic
  return(r.hat)
}

rayleigh.chain <- function(sigma, N, X1) {
  # generate a Metropolis chain for Rayleigh(sigma)
  # with chisq(Xt) proposal distribution
  # and starting value X1

  # the Rayleigh distribution pdf
  f <- function(x, sigma) {
    if (any(x < 0)) return (0)
    stopifnot(sigma > 0)
    return ((x / sigma^2) * exp(-x^2 / (2*sigma^2)))
  }

  # initiate the MC
  x <- numeric(N)
  # chi-square as proposal
  x[1] <- X1
  # generate uniform numbers
  u <- runif(N)

  for (i in 2:N) {
    xt <- x[i-1]
    y <- rchisq(1, df = xt)
    # numerator
    num <- f(y, sigma) * dchisq(xt, df = y)
    # denominator
    den <- f(xt, sigma) * dchisq(y, df = xt)
    if (u[i] <= num/den) x[i] <- y else {
      x[i] <- xt
    }
  }
  return (x)
}
```

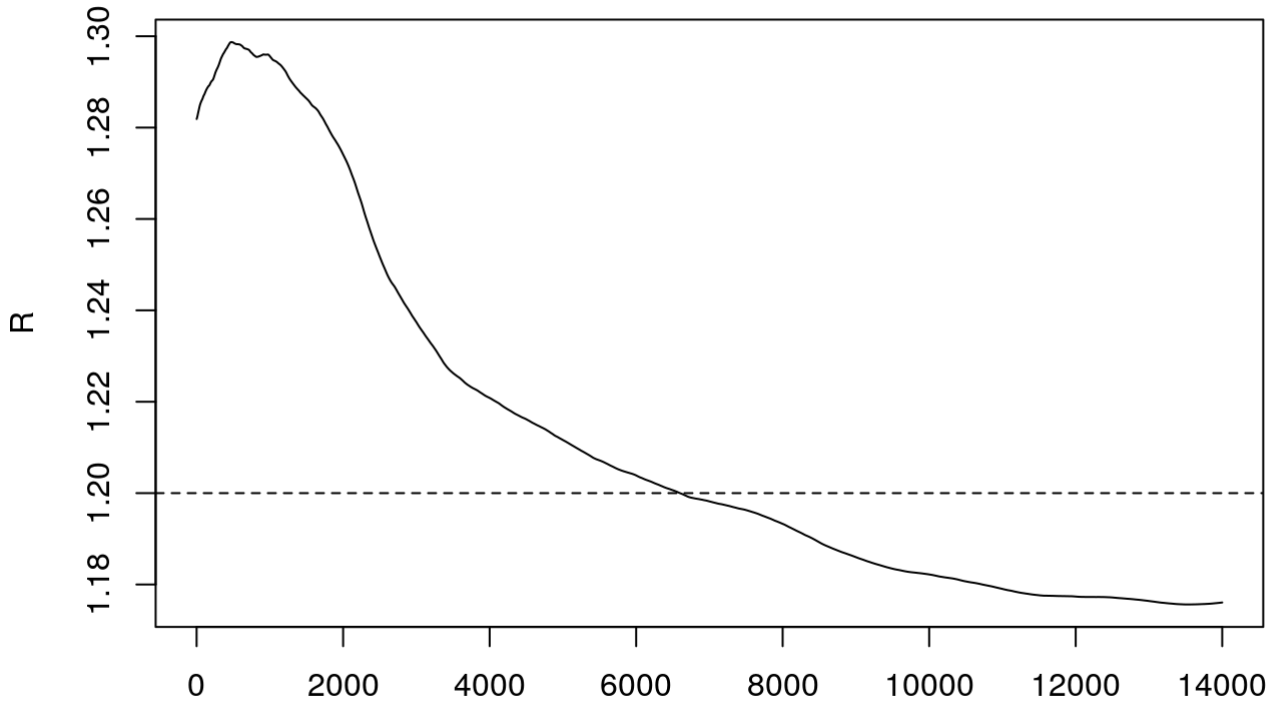
```r
set.seed(42)
# parameter for rayleigh distribution
sigma <- 2
# number of chains
k <- 4
# length of chains
N <- 15000
# burn in length
b <- 1000

# choose overdispersed initial values
x0 <- c(1,5,9,13)

# generate the chains
X <- matrix(0, nrow = k, ncol = N)
for (i in 1:k)
  X[i,] <- rayleigh.chain(sigma, N, x0[i])

# compute the diagnostic statistics
psi <- t(apply(X, 1, cumsum))
for (i in 1:nrow(psi)) {
  psi[i,] <- psi[i,] / (1:ncol(psi))
}
print(Gelman.Rubin(psi))
```

```
## [1] 1.176063
```

```r
#plot psi for the four chains
par(mfrow=c(2,2))
for (i in 1:k)
  plot(psi[i, (b+1):N], type="l",
       xlab=i, ylab=bquote(psi))
```




```
par(mfrow=c(1,1)) #restore default

#plot the sequence of R-hat statistics
rhat <- rep(0, N)
for (j in (b+1):N)
  rhat[j] <- Gelman.Rubin(psi[,1:j])
plot(rhat[(b+1):N], type="l", xlab="", ylab="R")
abline(h=1.2, lty=2)
```

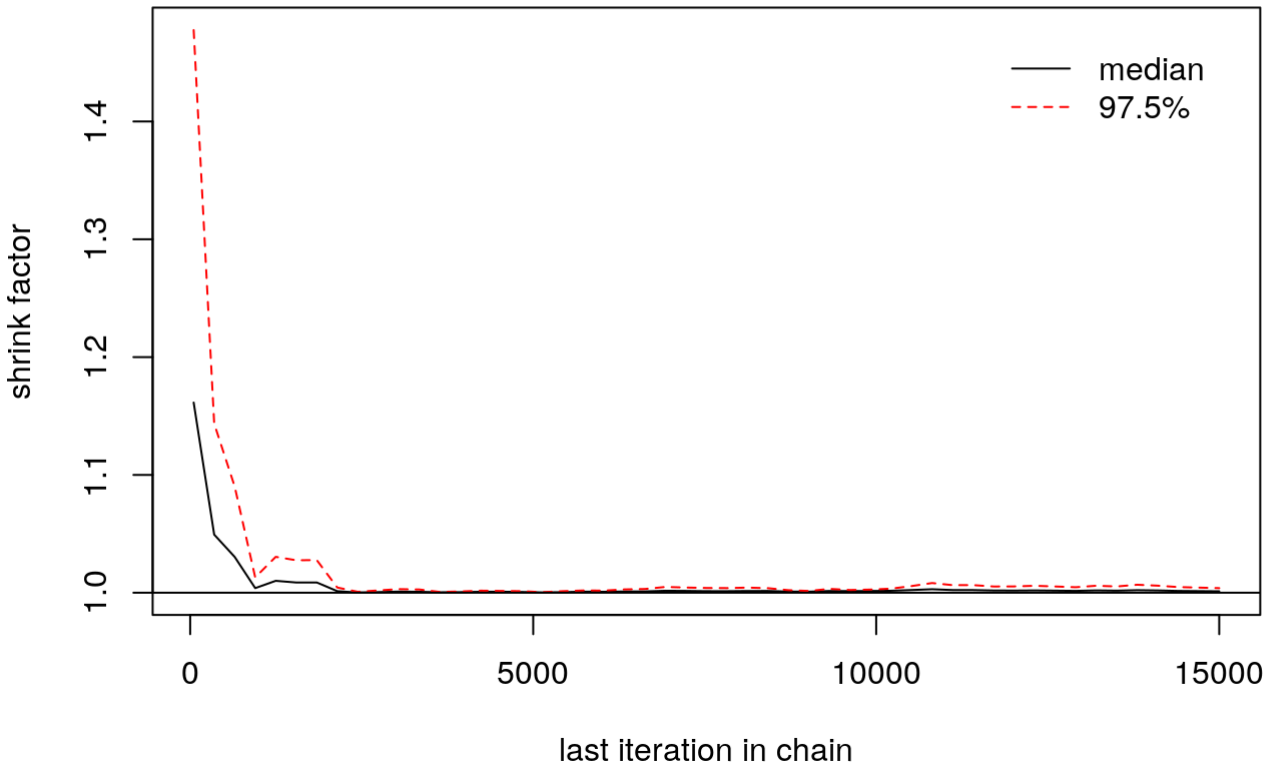


After aroud 6300 iteration, the Gelman-Rubin $\hat{R}$ is less than 1.2. Using package `coda` , we can also see that chain converges well.

```
x1.mcmc <- mcmc(X[1,])
x2.mcmc <- mcmc(X[2,])
x3.mcmc <- mcmc(X[3,])
x4.mcmc <- mcmc(X[4,])
x.mcmc <- mcmc.list(x1.mcmc, x2.mcmc, x3.mcmc, x4.mcmc)

gelman.diag(x.mcmc)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```

```
gelman.plot(x.mcmc)
```

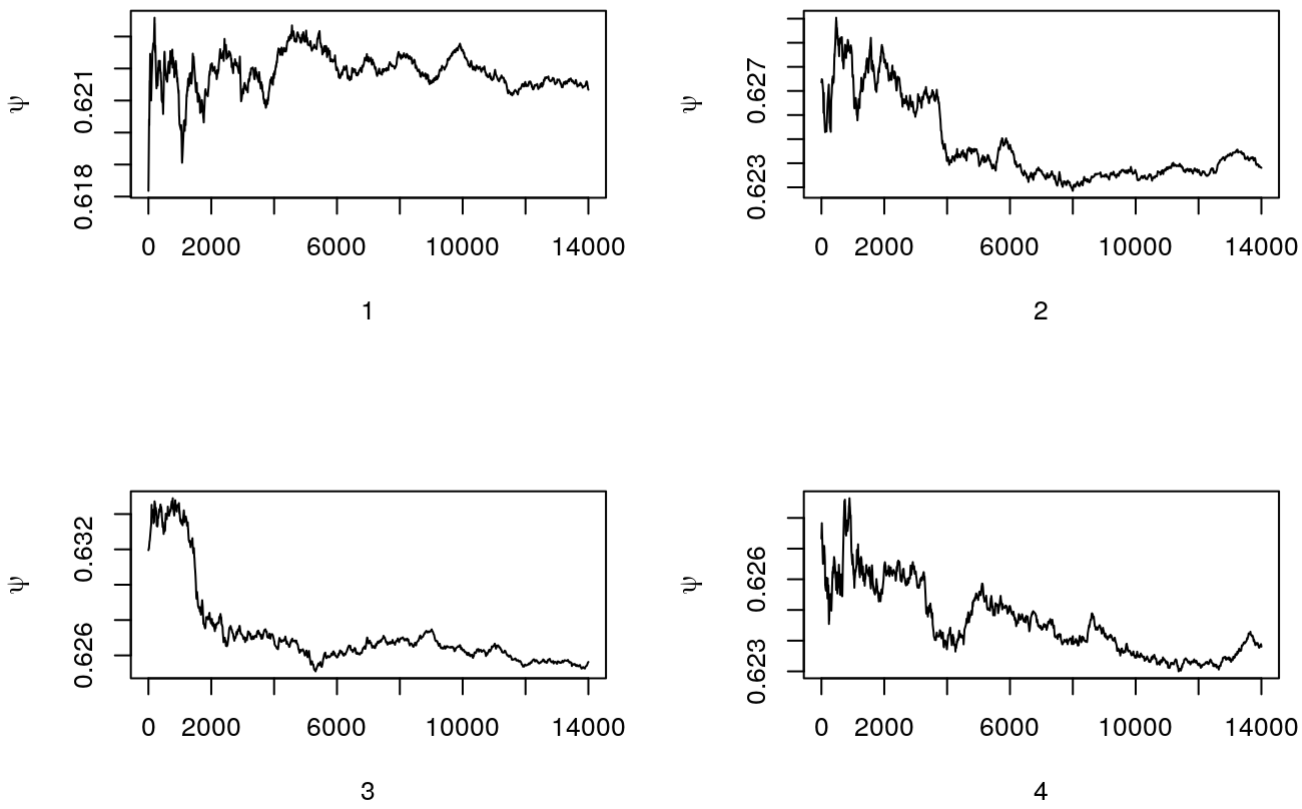


9.12

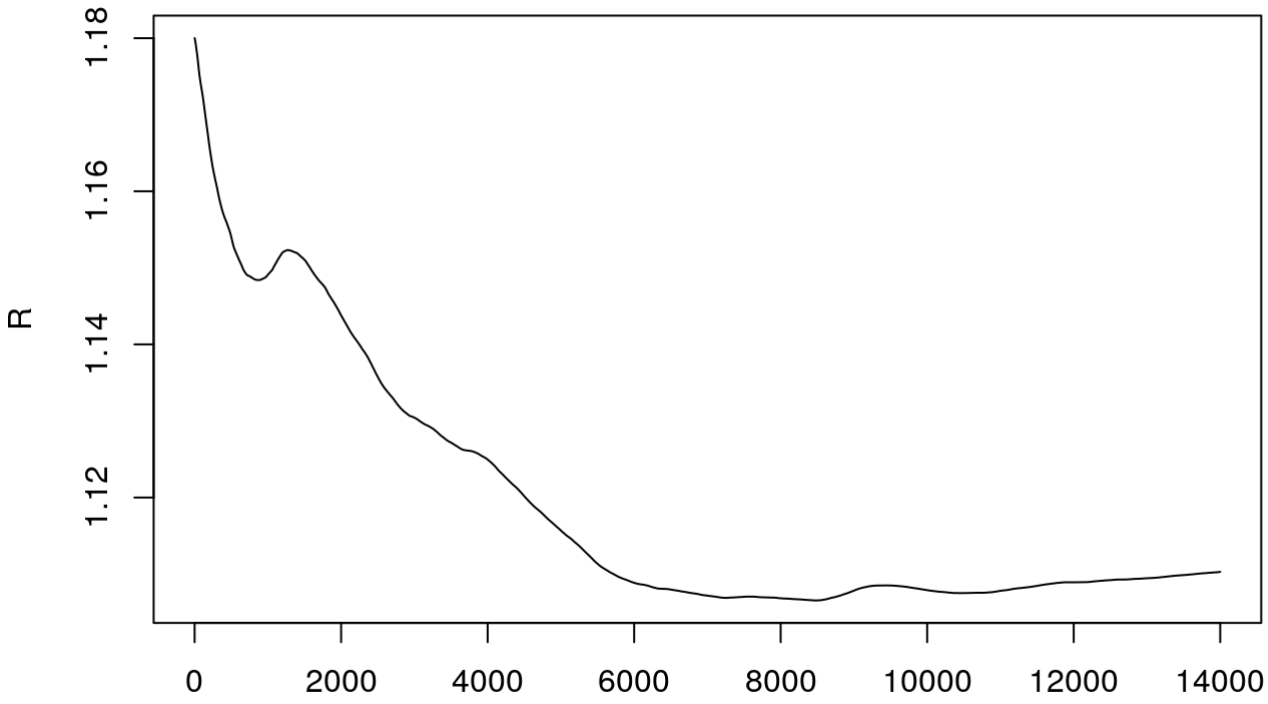
```
set.seed(42)
# number of chains
k <- 4
# length of chains
N <- 15000
# burn in length
b <- 1000

# choose overdispersed initial values
x0 <- c(0, 0.3, 0.6, 0.9)

# generate the chains
X <- matrix(0, nrow = k, ncol = N)
for (i in 1:k)
  X[i,] <- rw.Metropolis(0.5, x0[i], N, likelihood)

# compute the diagnostic statistics
psi <- t(apply(X, 1, cumsum))
for (i in 1:nrow(psi)) {
  psi[i,] <- psi[i,] / (1:ncol(psi))
}
print(Gelman.Rubin(psi))
```

```
## [1] 1.110296
```

```
#plot psi for the four chains
par(mfrow=c(2,2))
for (i in 1:k)
  plot(psi[i, (b+1):N], type="l",
       xlab=i, ylab=bquote(psi))
```



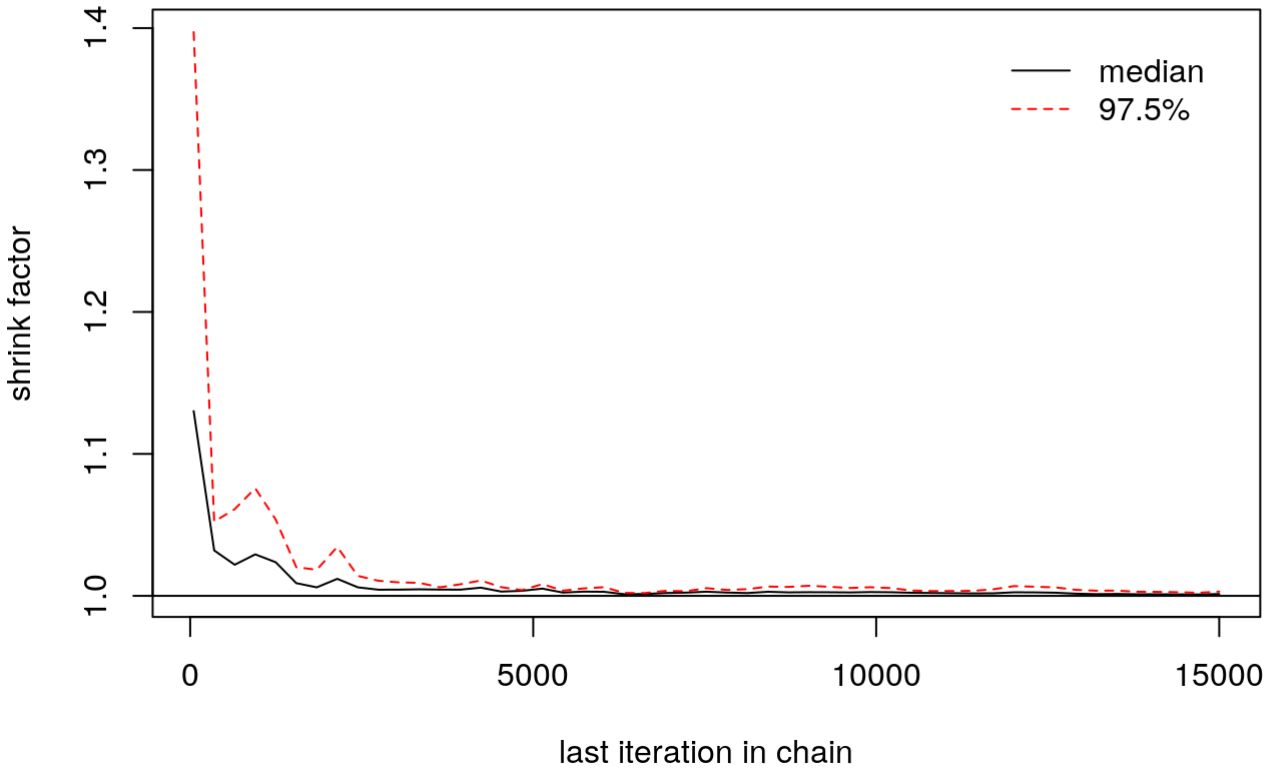

```
par(mfrow=c(1,1)) #restore default

#plot the sequence of R-hat statistics
rhat <- rep(0, N)
for (j in (b+1):N)
  rhat[j] <- Gelman.Rubin(psi[,1:j])
plot(rhat[(b+1):N], type="l", xlab="", ylab="R")
abline(h=1.2, lty=2)
```




```
x1.mcmc <- mcmc(X[1,])
x2.mcmc <- mcmc(X[2,])
x3.mcmc <- mcmc(X[3,])
x4.mcmc <- mcmc(X[4,])
x.mcmc <- mcmc.list(x1.mcmc, x2.mcmc, x3.mcmc, x4.mcmc)

gelman.diag(x.mcmc)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```

```
gelman.plot(x.mcmc)
```



2

In the example Change Point Analysis, fill in the detailed derivation for the posterior distributions on page 273.

$$
\begin{aligned}
f(k|Y,\mu,\lambda,b_1,b_2) &= \frac{f(Y,k,\mu,\lambda,b_1,b_2)}{f(Y,\mu,\lambda,b_1,b_2)} \\
&= \frac{f(Y|k,\mu,\lambda)f(\mu|b_1)f(\lambda|b_2)f(k)f(b_1)f(b_2)}{\sum_{k=1}^{n} f(Y|k,\mu,\lambda)f(\mu|b_1)f(\lambda|b_2)f(k)f(b_1)f(b_2)} \\
&= \frac{f(Y|k,\mu,\lambda)f(k)}{\sum_{k=1}^{n} f(Y|k,\mu,\lambda)f(k)} \\
&= \frac{f(Y|k,\mu,\lambda)\cdot\frac{1}{n}}{\sum_{k=1}^{n} f(Y|k,\mu,\lambda)\cdot\frac{1}{n}} \\
&= \frac{f(Y|k,\mu,\lambda)}{\sum_{k=1}^{n} f(Y|k,\mu,\lambda)}
\end{aligned}
$$

$$
\begin{aligned}
f(Y|k,\mu,\lambda) &= \prod_{i=1}^{k} f(y_i|\mu) \cdot \prod_{i=k+1}^{n} f(y_i|\lambda) \\
&= \prod_{i=1}^{k} \frac{\mu^{y_i}}{y_i!} e^{-\mu} \prod_{i=k+1}^{n} \frac{\lambda^{y_i}}{y_i!} e^{-\lambda} \\
&= e^{k(\lambda-\mu)} e^{-n\lambda} \frac{\mu^{S_k} \lambda^{S_n - S_k}}{\prod_{i=1}^{n} y_i!} \\
&= e^{k(\lambda-\mu)} \left(\frac{\mu}{\lambda}\right)^{S_k} \cdot \frac{e^{-n\lambda} \lambda^{S_n}}{\prod_{i=1}^{n} y_i!}
\end{aligned}
$$

The fraction in the right part contains no k, thus would be eliminated with the same factor in the denominator. Hence,

$$
f(k|Y,\mu,\lambda,b_1,b_2) = \frac{L(Y;k,\mu,\lambda)}{\sum_j L(Y;j,\mu,\lambda)}
$$

where $L(Y;k,\mu,\lambda) = e^{k(\lambda-\mu)} \left(\frac{\mu}{\lambda}\right)^{S_k}$