Computer class 5 exercises

Richard Wilkinson

Solution 1

The prior is

$$\pi(p) = \begin{cases} 1 & \text{if } p \in [0,1] \\ 0 & \text{otherwise} \end{cases}$$

and the likelihood is

$$\begin{aligned}\pi(X|p) &= \prod_{i=1}^{10} \binom{n}{x_i} p^{x_i}(1-p)^{n-x_i} \\ &\propto p^{\sum x_i}(1-p)^{10n-\sum x_i}\end{aligned}$$

To do rejection sampling, we are told to use $g(x) = 1$ on $[0,1]$. We need to calculate

$$M = \sup \frac{f(p)}{g(p)}$$

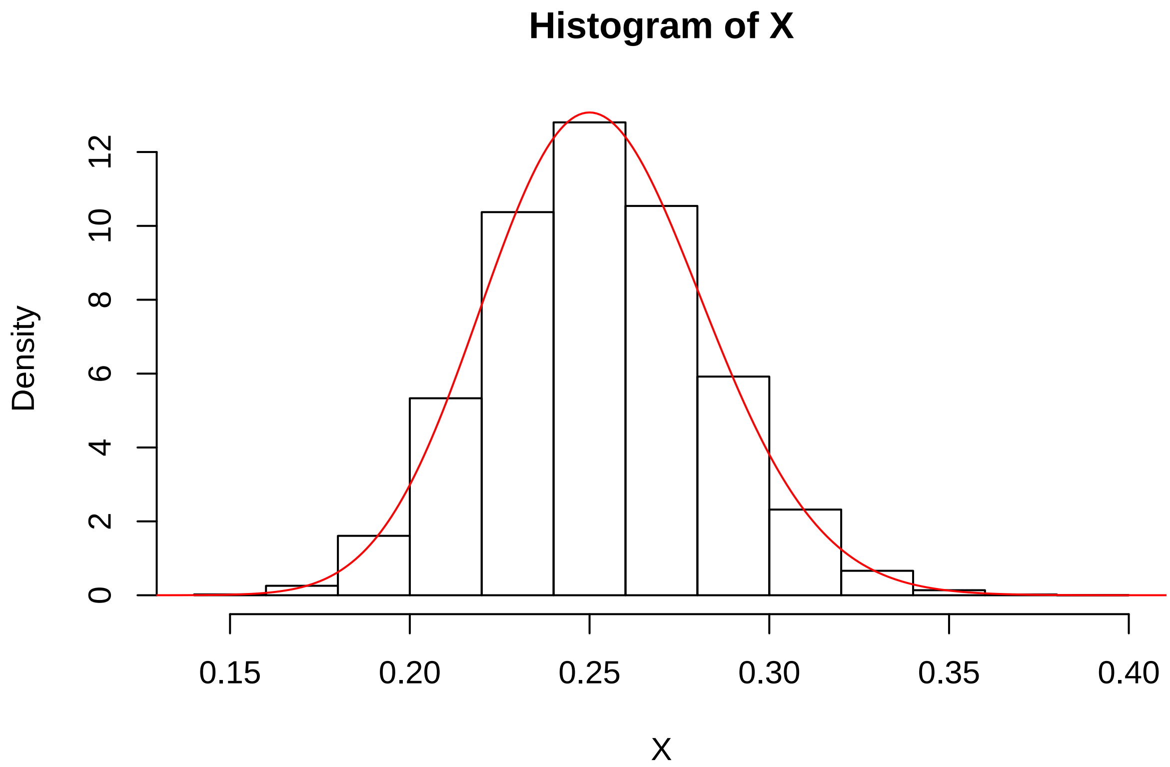
which obviously occurs at the MLE $\hat{p} = \frac{\sum x_i}{200}$ giving

$$M = \hat{p}^{50}(1-\hat{p})^{150}$$

```
phat <- 50/200
M <- phat^{50}*(1-phat)^{150}
```

```
Y <- runif(10^6)
U <- runif(10^6)
keep <- U <= Y^50*(1-Y)^150/M
X <- Y[keep]
```

```
hist(X, probability=TRUE)
xvals <- seq(0,1,0.001)
lines(xvals, dbeta(xvals, shape1=51, shape2=151), col=2)
```



Solution 2

In the previous computer class, we considered a simple Bayesian inference problem where we wanted to learn about an unknown parameter p, to which we assigned a U[0,1] distribution. We were given data $x_1, \ldots, x_{10}$ which were independent $Bin(20, p)$ random variables, and we were told that $\sum_{i=1}^{10} x_i = 50$. Recall that the likelihood times the prior for this problem was proportional to

$$f_1(p) = p^{\sum x_i}(1-p)^{200-\sum_i x_i}.$$

- Using a $U[0,1]$ distribution as the importance distribution, use importance sampling to generate a weighted sample

$$\{p_i, w_i\}_{i=1}^N$$

of particles and weights that approximates the posterior distribution.

```
N <- 10000
f1 <- function(p){
  p^50*(1-p)^150
}
g <- 1
p <- runif(N)
w <- f1(p)/g
```

- Calculate the posterior mean of p. Note that we can approximate any integral by a weighted sum. So for example,

$$E(\theta|x) = \int \theta\pi(\theta|x)d\theta \approx \frac{\sum w_i\theta_i}{\sum w_i}.$$

Alternatively, we can use the weighted version of statistical estimators in the Hmisc library, for example, `wtd.mean`. You may need to install Hmisc the first time you use it (`install.packages('Hmisc')`)

```r
W <- w/sum(w)
sum(W*p)
```

```
## [1] 0.252756
```

```r
library(Hmisc)
```

```
## Warning: replacing previous import by 'ggplot2::unit' when loading 'Hmisc'
## Warning: replacing previous import by 'ggplot2::arrow' when loading 'Hmisc'
## Warning: replacing previous import by 'scales::alpha' when loading 'Hmisc'
```

```r
wtd.mean(p, weights=W)
```

```
## [1] 0.252756
```

- We can resample the particles to get an unweighted sample of particles. To do this, first convert the weights into probabilities,

$$W_i = \frac{w_i}{\sum w_i}$$

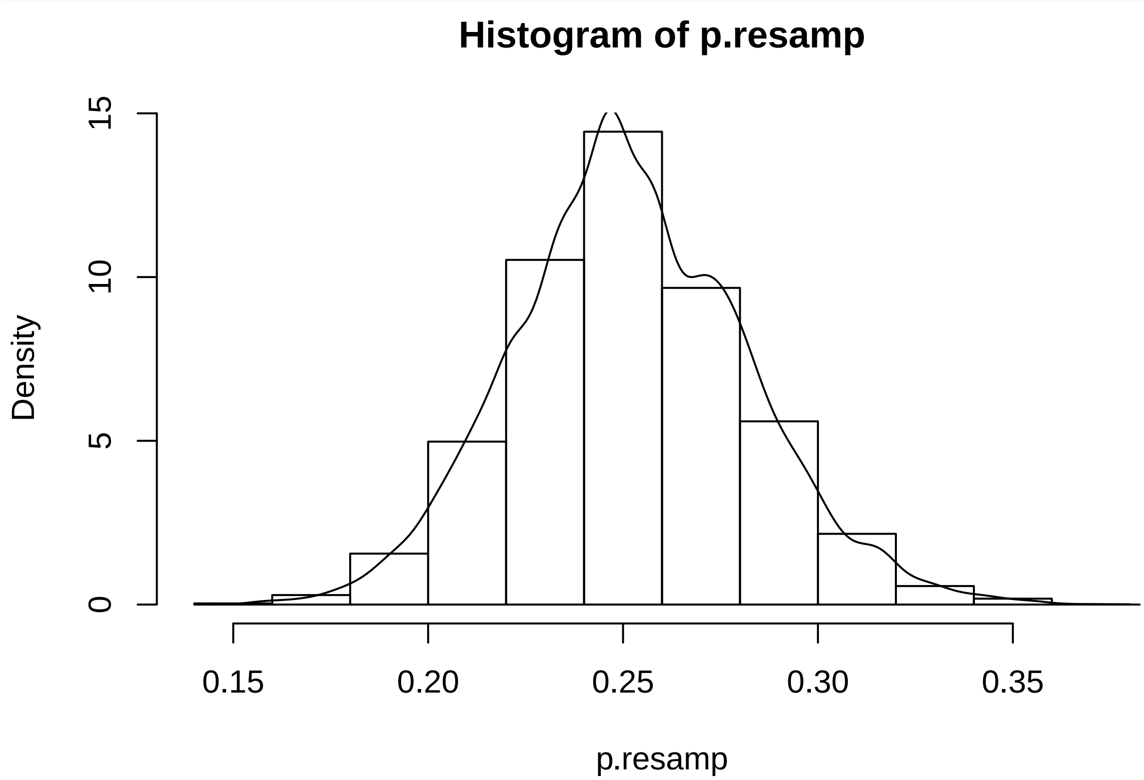
and then sample from $\{p_i\}_{i=1}^N$ with replacement, picking particle i with probability W_i. Calculate the number of unique particles in your unweighted sample.

```r
p.resamp <- sample(p, size=N, replace=TRUE, prob=W)
length(unique(p.resamp))
```

```
## [1] 1492
```

- Use the resampled particles to plot a histogram of the posterior distribution.

```r
hist(p.resamp, prob=T)
lines(density(p.resamp))
```

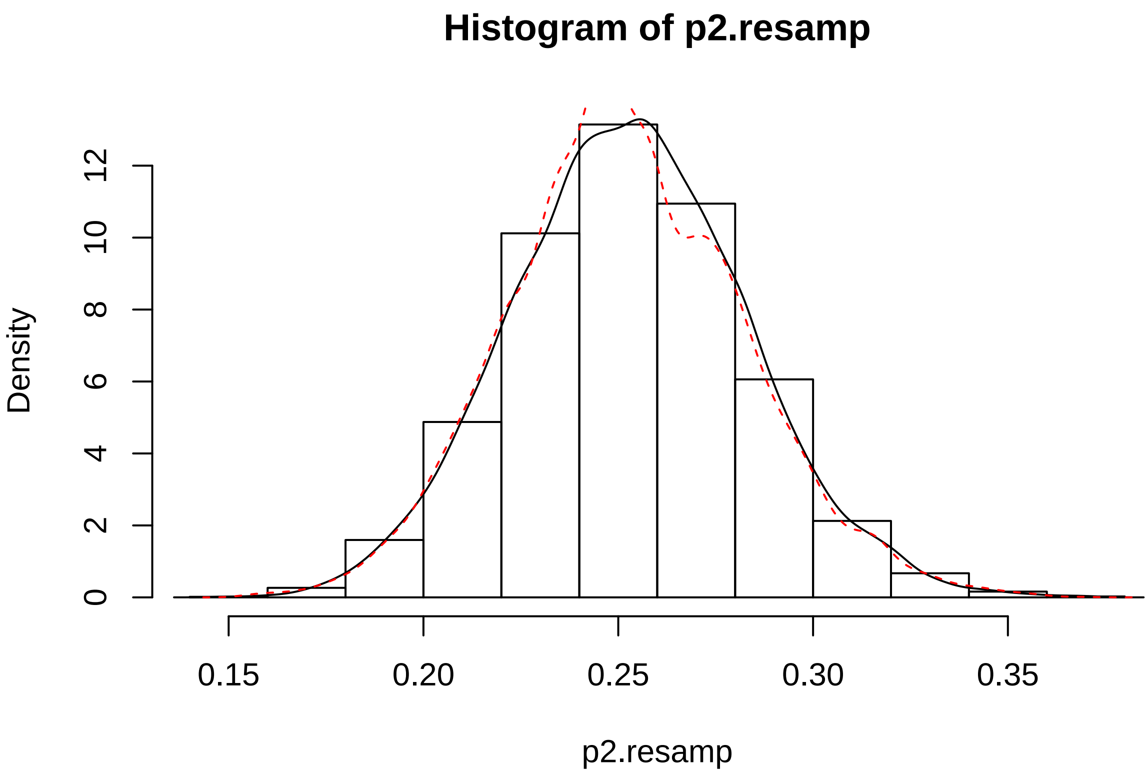


- Repeat the steps above using a $Beta(10, 30)$ distribution as the importance distribution.

```
p2 <- rbeta(N, 10,30)
g2 <- dbeta(p2, 10,30)
w2 <- f1(p2)/g2
W2 <- w2/sum(w2)
wtd.mean(p2, weights=W2)
```

```
## [1] 0.2527932
```

```
p2.resamp <- sample(p2, size=N, replace=TRUE, prob=W2)
hist(p2.resamp, prob=T)
lines(density(p2.resamp))
lines(density(p.resamp), col=2, lty=2)
```



- The variance of the importance weights is a useful measure of how successful a given importance distribution will be - we want the variance to be as small as possible. A related quantity that is often used is the effective sample size (ESS)

$$ESS = \frac{1}{\sum W_i^2}$$

where the W_i are the normalised weights ($\sum W_i = 1$). If all the weights are the same (i.e. they have zero variance), then the ESS = N, i.e. the sample is as effective as a sample of N unweighted particles. Whereas in the worst case where all the weights are 0 except for one which has $W = 1$, then the ESS=1, i.e., the sample is equivalent to a single unweighted sample. Calculate the ESS for your two importance distributions to see which gives a better sample.

```
1/sum(W^2)
```

```
## [1] 1090.688
```

```
1/sum(W2^2)
```

```
## [1] 5967.854
```

- What choice of importance distribution would give the best possible ESS?

The optimal importance distribution is the posterior distribution, which we showed previously to be

$$\pi(p|x_1, \ldots, x_{10}) = \text{Beta}(51, 151).$$

This would give an ESS=N, as each weight would be 1/N.

Solution 3

This problem is described in the notes. Here we will work through the details.

Patients suffering from leukaemia are given a drug, 6-mercaptopurine (6-MP), and the number of days x_i until freedom from symptoms is recorded for patient i:

$$6^*, 6, 6, 6, 7, 9^*, 10^*, 10, 11^*, 13, 16, 17^*, 19^*, 20^*, 22, 23, 25^*, 32^*, 32^*, 34^*, 35^*,$$

where a * denotes censored observation. The time x to the event of interest follows a *Weibull* distribution:

$$f(x|\alpha, \beta) = \alpha\beta(\beta x)^{\alpha-1} \exp\{-(\beta x)^\alpha\}$$

for $x > 0$. For censored observations, we can show that

$$P(x > t|\alpha, \beta) = \exp\{-(\beta t)^\alpha\}.$$

We want to estimate the posterior mean of θ, and the posterior 5th and 95th percentiles.

- Briefly explain what calculation you would to do to do this analytically.

Define d to be the number of uncensored observations and $\sum_u \log x_i$ to be the sum of logs of all uncensored observations. Writing $\theta = (\alpha, \beta)^T$, show that the log likelihood is given by

$$\log f(x|\theta) = d \log \alpha + \alpha d \log \beta + (\alpha - 1) \sum_u \log x_i - \beta^\alpha \sum_{i=1}^n x_i^\alpha.$$

We will use the following prior distributions for α and β

$$\begin{aligned} f(\alpha) &= 0.001 \exp(-0.001\alpha), \\ f(\beta) &= 0.001 \exp(-0.001\beta). \end{aligned}$$

- Obtain the posterior mode of θ, i.e., maximise the log posterior

$$h(\theta) = d \log \alpha + \alpha d \log \beta + (\alpha - 1) \sum_u \log x_i - \beta^\alpha \sum_{i=1}^n x_i^\alpha - 0.001\alpha - 0.001\beta + K,$$

  for some constant K. You can do this in R by writing a function to evaluate h and then using the `optim` command. Note that optim does minimization by default.

```
# Define data
alldata<-c(6,6,6,6,7,9,10,10,11,13,16,17,19,20,22,23,25,32,32,34,35)
uncensored<-c(6,6,6,7,10,13,16,22,23)
d <- length(uncensored)

#log posterior function.
```

```r
weibulllogposterior<-function(theta){
# theta is log of parameters, to ensure alpha and beta are positive
  alpha<-exp(theta[1])
  bet<-exp(theta[2])
  logprior<- -(0.001*alpha+0.001*bet)
  tmp <- logprior+d*log(alpha)+alpha*d*log(bet)+
      (alpha-1)*sum(log(uncensored))-bet^alpha*sum(alldata^alpha)
  return(tmp)
}

# Find mode
# Should experiment with alternative starting values
m<-exp(optim(c(log(1),log(1)),function(x) -weibulllogposterior(x))$par)
m
```

```
## [1] 1.35405025 0.02960048
```

- Find the Hessian (matrix of second derivatives) of $h(\theta)$ at $\theta = m$, either by deriving it analytically, or estimating it using numerical differentiation (the `hessian` command in the numDeriv package works well),

$$M = \begin{pmatrix} \frac{\partial^2}{\partial\alpha^2}h(\theta) & \frac{\partial^2}{\partial\alpha\partial\beta}h(\theta) \\ \frac{\partial^2}{\partial\alpha\partial\beta}h(\theta) & \frac{\partial^2}{\partial\beta^2}h(\theta) \end{pmatrix}.$$

Using the expressions given in lectures

```r
Hessian<-function(alpha,bet,d,alldata){
    v11 <- -d/alpha^2 - sum((bet*alldata)^alpha*log(bet*alldata)^2)
    v22<- 1/bet^2*(bet^alpha*alpha*(1-alpha)*sum(alldata^alpha)-alpha*d)
    v12<- 1/bet*(d-bet^alpha*log(bet)*alpha*sum(alldata^alpha)-
                   bet^alpha*sum(alldata^alpha)-
                   bet^alpha*alpha*sum(alldata^alpha*log(alldata)) )
    M<- matrix(c(v11,v12,v12,v22),nrow=2,ncol=2)
    return(M)
}

(M<-Hessian(m[1],m[2],d,alldata))
```

```
##            [,1]        [,2]
## [1,]  -8.67542    175.8999
## [2,] 175.89994 -18828.5531
```

Or, we can use a numerical estimate of the Hessian

```r
library(numDeriv)

f <- function(x){
  alpha <- x[1]
  beta <- x[2]
  -0.001*alpha-0.001*beta+d*log(alpha) +alpha*d*log(beta)+(alpha-1)*sum(log(uncensored))-
    beta^alpha*sum(alldata^alpha)
}
hessian(f, m)
```

```
##            [,1]        [,2]
## [1,]  -8.67542    175.8999
## [2,] 175.89994 -18828.5531
```

- Use an importance sampling algorithm to estimate the posterior mean and 5th and 95th percentiles of this distribution. Use a multivariate Gaussian distribution as your proposal, with mean m and covariance matrix $V = -M$. To simulate from a multivariate normal, you can either use the Cholesky decomposition of V, or use the mvtnorm package in R (you may need to install it using `install.packages('mvtnorm')` the first time you use this). Note that you will need to use `wtd.quantile` or resample the particles and use `quantile` to get the quantiles.

```
(V <- -solve(M))
```

```
##             [,1]         [,2]
## [1,] 0.142204444 1.328501e-03
## [2,] 0.001328501 6.552194e-05
```

```
N<-10000

library(mvtnorm)
#Sample from importance density
X<-rmvnorm(N,m,V)
```

We need to discard negative values as we know the Weibull distribution is constrained to be positive. This is equivalent to using a truncated Gaussian as the proposal. Although this changes the proposal density g, it is only by a multiplicative constant and so this will cancel when we divide by the sum of the weights.

```
#discard negative values
check<-apply(X,1,min)
X<-X[check>0,]
dim(X)
```

```
## [1] 9999    2
```

```
Npos <- dim(X)[1]
#Evaluate importance weights

flogdensity <- apply(X,1, function(x) f(x))
glogdensity<-dmvnorm(X,m,V, log=TRUE)
w <- exp(flogdensity-glogdensity)
W <- w/sum(w)
```

It is a good idea to check that none of the importance weights are too large - if they are, this is called degeneracy, and means that the final estimate will be dominated by just a few of the random samples. If we observe this, we need to find a better proposal g.

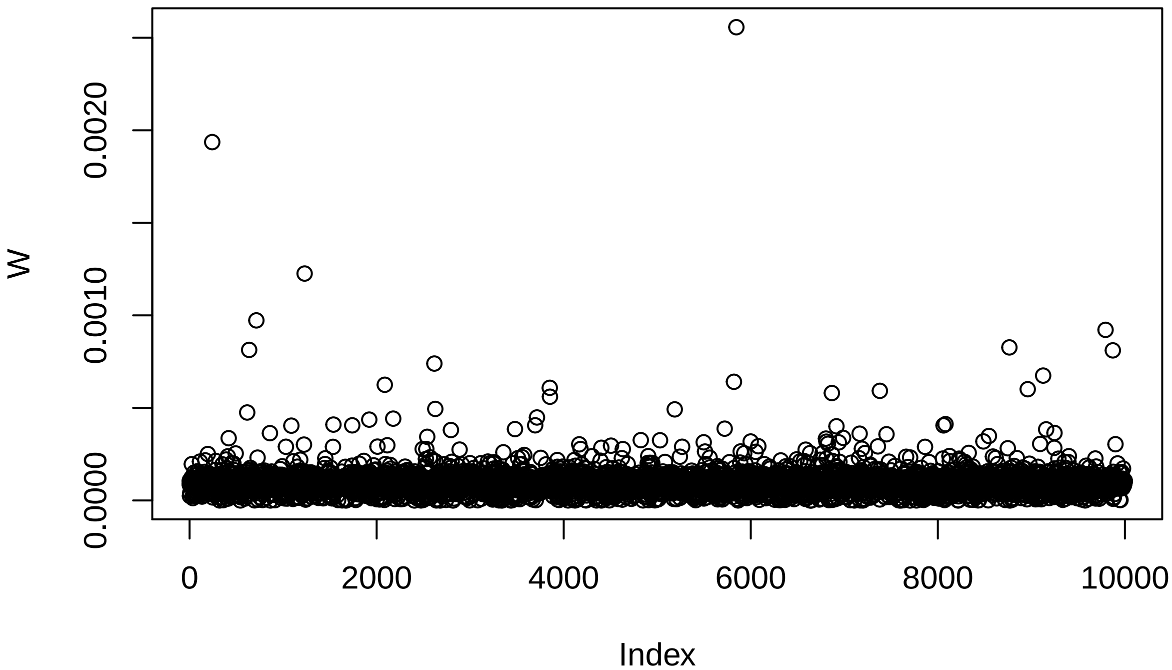
```
# Examine importance weights (none too large)
var(W)
```

```
## [1] 2.835643e-09
```

```
1/sum(W^2) # ESS
```

```
## [1] 7790.543
```

```
plot(W)
```



Finally, we can estimate the posterior mean. Any integral is approximated by a weighted sum. So for example,

$$E(\theta|x) = \int \theta\pi(\theta|x)d\theta \approx W_i\theta_i$$

You can alternatively use the weighted version of statistical estimators in the Hmisc library.

```
#Estimate posterior mean
t(X)%*%W
```

```
##             [,1]
## [1,] 1.38210739
## [2,] 0.03067852
```

```
library(Hmisc)
wtd.mean(X[,1], weights = W)
```

```
## [1] 1.382107
```

```
wtd.mean(X[,2], weights = W)
```

```
## [1] 0.03067852
```

To estimate the quantiles, we need to use the weighted quantile functions.

```
library(Hmisc)
wtd.quantile(X[,1], weights = W, probs =c(0.05, 0.95), normwt=TRUE)
```

```
##        5%       95%
## 0.8281793 2.0423928
```

```
wtd.quantile(X[,2], weights = W, probs =c(0.05, 0.95), normwt=TRUE)
```

```
##         5%        95%
## 0.01764572 0.04358096
```

- Resample the particles to get an unweighted sample, and plot the posterior distribution of α and β.

```
#Resample
index<-sample(1:Npos, replace=TRUE , prob=W)
length(unique(index)) # how many distinct values resampled?
```

```
## [1] 6050
```

```
apply(X[index,],2, mean)
```

```
## [1] 1.37948603 0.03062571
```

```
quantile(X[index,1], probs=c(0.05, 0.95))
```

```
##        5%       95%
## 0.8308395 2.0349042
```

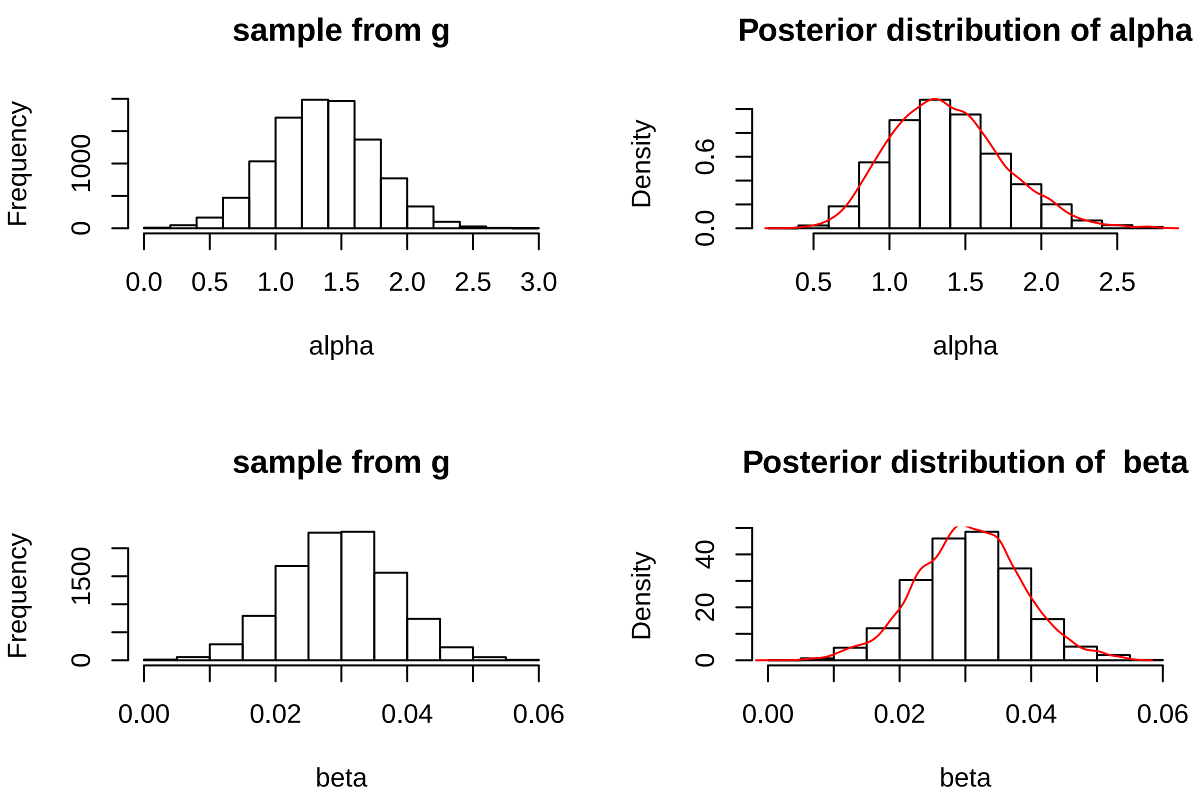
```
quantile(X[index,2], probs=c(0.05, 0.95))
```

```
##         5%         95%
## 0.01774452 0.04354689
```

Finally, let's plot the posterior distributions using the unweighted sample, and for comparison, we'll also plot the sample generated from g alongside

```
par(mfrow=c(2,2))
hist(X[,1],xlab="alpha",main="sample from g")
hist(X[index,1],xlab="alpha",main="Posterior distribution of alpha", prob=TRUE)
lines(density(X[index,1]), col=2)

hist(X[,2],xlab="beta",main="sample from g")
hist(X[index,2],xlab="beta",main="Posterior distribution of  beta", prob=TRUE)
lines(density(X[index,2]), col=2)
```



We can see that the Laplace approximation we used to generate g was a very good approximation to the posterior, hence the high ESS value.

- Repeat the analysis but using a different importance distribution, for example, a $Exp(1)$ distribution. Does this work as well?

```
X2<-cbind(rexp(N,1), rexp(N,1))

flogdensity <- apply(X2,1, function(x) f(x))
glogdensity<-dexp(X2[,1],1, log=TRUE)+dexp(X2[,2],1, log=TRUE)
w<-exp(flogdensity-glogdensity)
W2 <- w/sum(w)
1/sum(W2^2)
```

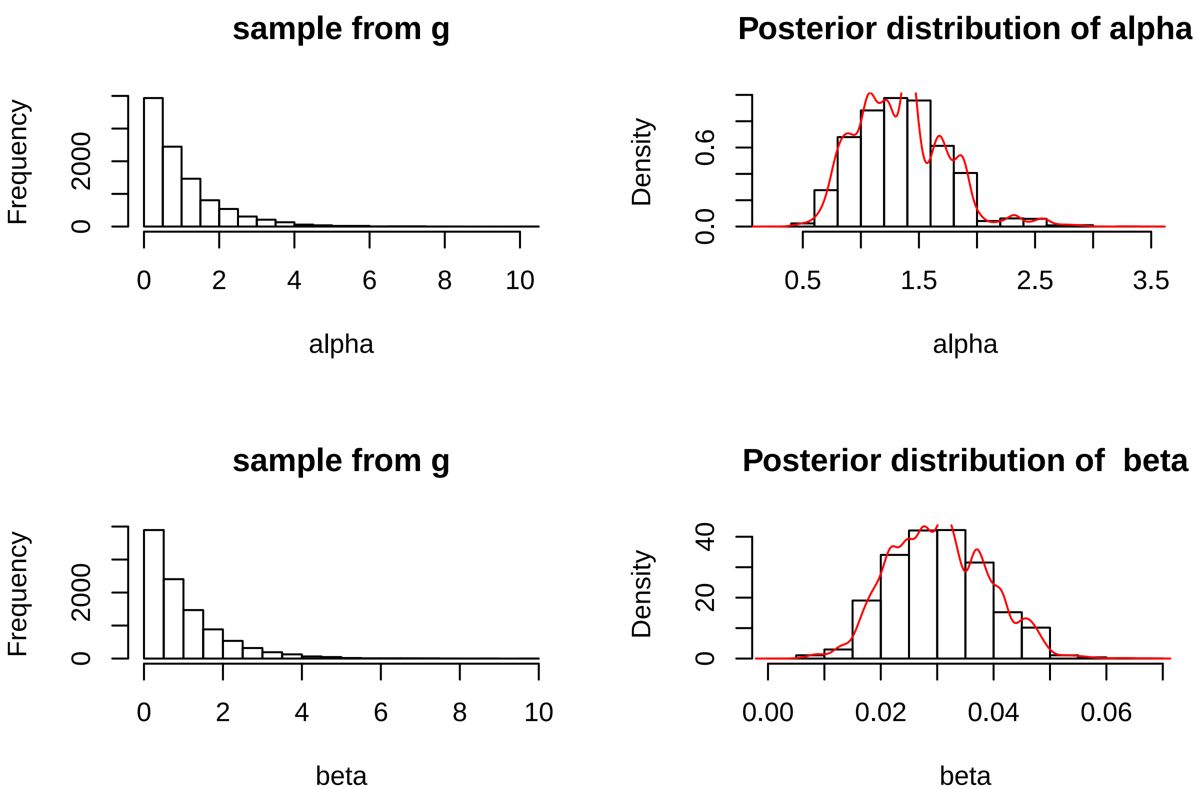
```
## [1] 82.52757
```

```
index<-sample(1:N, replace=TRUE , prob=W2)
length(unique(index)) # how many distinct values resampled?
```

```
## [1] 276
```

```
par(mfrow=c(2,2))
hist(X2[,1],xlab="alpha",main="sample from g")
hist(X2[index,1],xlab="alpha",main="Posterior distribution of alpha", prob=TRUE)
lines(density(X2[index,1]), col=2)

hist(X2[,2],xlab="beta",main="sample from g")
hist(X2[index,2],xlab="beta",main="Posterior distribution of  beta", prob=TRUE)
lines(density(X2[index,2]), col=2)
```



It is easy to see that this has worked much less well than using the Laplace approximation to build a bespoke choice for g.