Chapter 3

3.1 Suppose that $X \sim Normal(0,1)$ and $Y = \exp(X)$.

(a) Use the change of variables technique to calculate the probability density function, mean, and variance of Y.

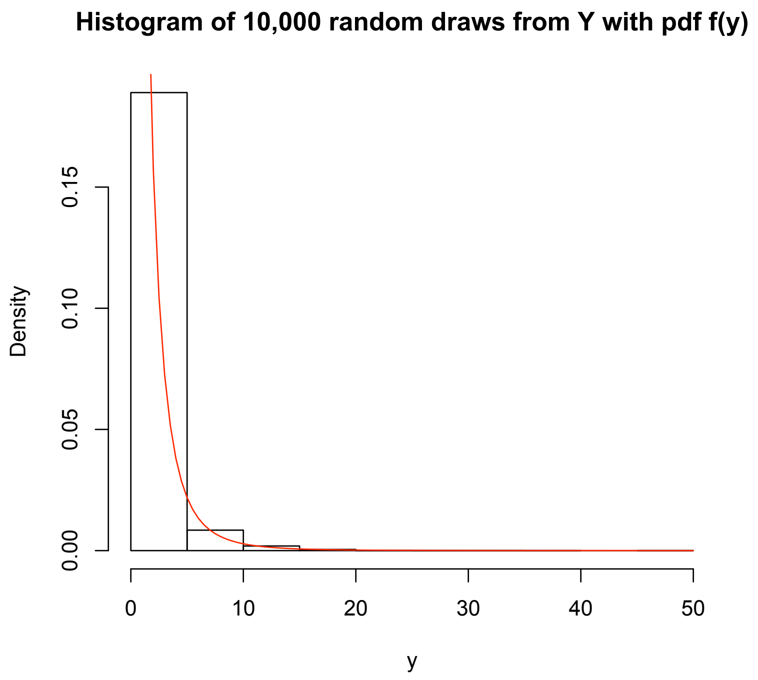
$X \sim$ Normal(0,1) $\quad f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$
$Y \sim e^X \quad g^{-1}(y) = \log(y) \quad \frac{d}{dy} g^{-1}(y) = 1/y$

$$\begin{aligned} f_Y(y) &= f_X(g^{-1}(y)) \left| \frac{d}{dy} g^{-1}(y) \right| \\ &= \frac{1}{\sqrt{2\pi} y} \exp\left[-\frac{1}{2} (\log(y))^2 \right], \quad y > 0 \end{aligned}$$

So $Y \sim LogNormal(0,1)$ with E(Y) $= e^{1/2} = 1.648$ and Var(Y) $= e^2 - e = 4.671$

(b) Draw a random sample $X_1, \ldots, X_{10{,}000}$ from a $Normal(0,1)$ distribution. Set $Y_i = \exp(X_i)$. Draw a histogram of the Y_i and overlay a plot of the probability density function of Y.

```
x <- rnorm(10000,0,1)
y <- exp(x)
f <- function(x){f <- 1/(x*sqrt(2*pi))*exp(-.5*(log(x))^2)}
hist(y, freq=F, main='Histogram of 10,000 random draws from Y
     with pdf f(y)')
curve(f, col=2, add=T)
```



(c) Estimate the probability density function, mean, and variance of Y using the random sample.

```
quantile(y,c(.025,.05,.5,.95,.975))
mean(y)
var(y)
```

			Quantiles				
Parameter	Mean	Variance	0.025	0.050	0.500	0.950	0.975
λ	1.66	4.64	0.14	0.19	0.99	5.17	7.04

3.2 Suppose we perform an experiment where the data have a $Poisson(\lambda)$ sampling density. We describe our uncertainty about λ using a *Gamma* prior density with parameters α and β. We also describe our uncertainty about α and β using independent *Gamma* prior densities.

(a) Simulate 50 observations from a Poisson distribution with parameter $\lambda = 5$.

```
x <- rpois(50,5)
```

(b) Choose diffuse prior densities for α and β.
Let $\alpha \sim Gamma(0.001, 0.001)$ and $\beta \sim Gamma(0.001, 0.001)$.

(c) Implement an MCMC algorithm to calculate posterior densities for λ, α, and β.

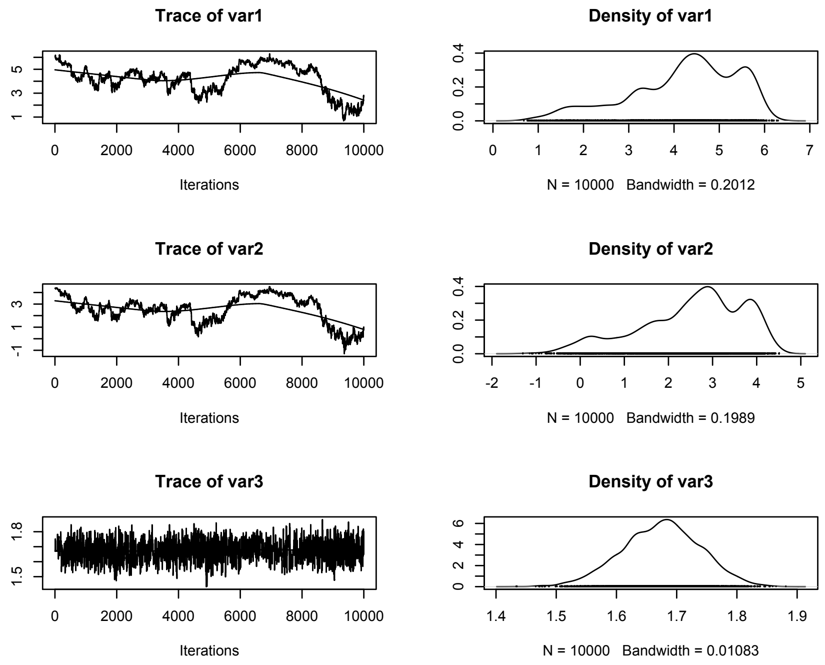
Evaluating the posterior distribution on the log scale helps to minimize numerical issues, as does transforming the parameters to real line. `logpost()` is a function that computes the log of the posterior. `logpost2()` is a function that computes the log transformed parameters.

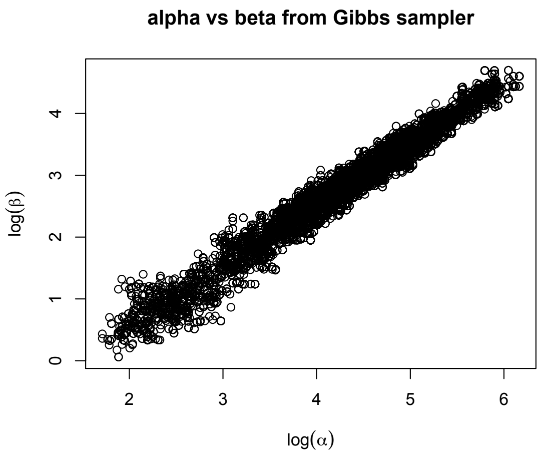
```
library(coda)
library(MASS)
library(LearnBayes)
library(MCMCpack)

logpost <- function(theta, data){
alpha = theta[1]
beta = theta[2]
lambda = theta[3]
n = length(data)
sumd = sum(data)
val = (sumd + alpha -1)*log(lambda) - lambda*(n+beta)
      + alpha*log(beta) - lgamma(alpha) + (.001-1)*log(alpha*beta)
      - .001*(alpha+beta)
return(val)
}

logpost2 <- function(theta, data){ #theta is the log of above theta
a = theta[1]
b = theta[2]
l = theta[3]
return(logpost(c(exp(a), exp(b), exp(l)), data) + a + b + l)
#remember the Jacobian!
}

mu.x <- mean(x)
theta <- c(1,1,mu.x); theta <- log(theta)
fit <- laplace(logpost2, theta, x)
```

```
sample <- gibbs(logpost2, fit$mode, 10000, c(.13,.13,.13), data=x)
sample$accept
plot(as.mcmc(sample$par))
plot(sample$par[,1], sample$par[,2], xlab=expression(alpha),
     ylab=expression(beta), main='alpha vs beta from Gibbs sampler')

sample2 <- rwmetrop(logpost2, list(var=fit$var, scale=2), fit$mode,
                    10000, x)
sample2$accept
plot(as.mcmc(sample2$par))
plot(sample2$par[,1], sample2$par[,2])
```

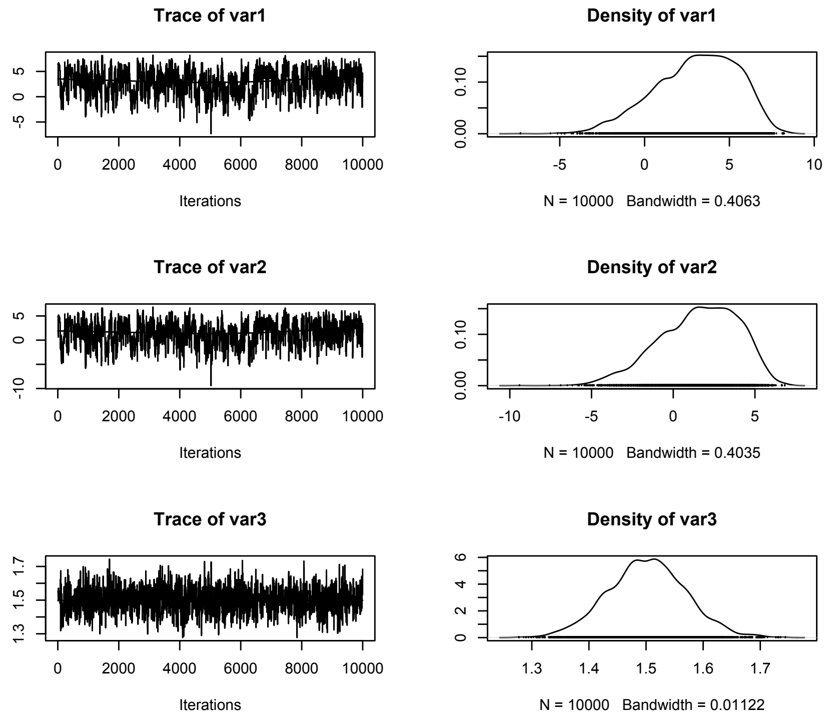
`gibbs()` and `rwmetrop()` are part of the LearnBayes package and `as.mcmc()` is in the MCMCpack package. `rwmetrop` is a random walk Metropolis algorithm, and `gibbs` is a Metropolis-within-Gibbs algorithm. If you would like addition reading and examples with these functions I recommend Jim Albert's *Bayesian Computation with R* (Springer).

The Gibbs sampler has problems mixing, probably due to the high correlation between α and β.





By sampling α and β together, the Metropolis algorithm mixes better than the Gibbs in this problem.



(d) Is $\lambda = 5$ contained in a 90% posterior credible interval for λ?

```
quantile(exp(sample2$par[,3]), c(.05,.95))
```

Yes. Remember that the above graphs are based on the log of the parameters. A 90% posterior credible interval for λ is (4.06, 5.05).

3.3 Consider again the fluid breakdown times introduced in Sect. 2.5. Two models were proposed for these data. The first incorporated a normal likelihood function and a noninformative prior density; the second, a normal likelihood function and a conjugate inverse-gamma/normal prior density. Now suppose that the properties of the manufacturing process were controlled when these samples of lubricant were produced so that it is known that the true mean of the sample values must lie between 6.0 and 7.4 (on the original measurement scale). No further information is available concerning the value of the variance parameter σ^2. Assume that the joint prior density for (μ, σ^2) is proportional to $1/\sigma^2$ whenever $\mu \in (\log(6.0), \log(7.4))$, and is 0 otherwise.

(a) Find an expression for a function that is proportional to the joint posterior density.

We can write the joint prior of μ and σ^2 with the use of an indicator function. i.e., $p(\mu, \sigma^2) \propto \frac{1}{\sigma^2} I[\log(6.0) < \mu < \log(7.4)]$. Then

$$p(\mu, \sigma^2 \,|\, \mathbf{y}) \propto (\sigma^2)^{-\frac{n}{2}-1} \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\mu)^2\right] I[\log(6.0) < \mu < \log(7.4)].$$

(b) Describe a hybrid Gibbs/Metropolis-Hastings algorithm for sampling from the joint posterior density.

$Z \sim Normal(0,1)$ and c_1 and c_2 are specified constants for μ and σ respectively.

0. Initialize $j = 0$, and starting vales $\mu^{(j)}$, $\sigma^{2(j)}$.

1. Generate μ^* from $\mu^{(j)} + c_1 Z$.

2. Compute $r = \frac{(\sigma^{2(j)})^{-\frac{n}{2}-1}\exp\left[-\frac{1}{2\sigma^{2(j)}}\sum_{i=1}^{n}(y_i-\mu^*)^2\right] I[\log(6.0)<\mu^*<\log(7.4)]}{(\sigma^{2(j)})^{-\frac{n}{2}-1}\exp\left[-\frac{1}{2\sigma^{2(j)}}\sum_{i=1}^{n}(y_i-\mu^{(j)})^2\right] I[\log(6.0)<\mu^{(j)}<\log(7.4)]}$.

3. Draw u from a $Uniform(0,1)$ distribution.

4. If $u \le r$, set $\mu^{(j+1)} = \mu^*$. Otherwise, set $\mu^{(j+1)} = \mu^{(j)}$.

5. Generate σ^{2*} from $\sigma^{2(j)} + c_2 Z$.

6. Compute $r = \frac{(\sigma^{2*})^{-\frac{n}{2}-1}\exp\left[-\frac{1}{2\sigma^{2*}}\sum_{i=1}^{n}(y_i-\mu^{(j)})^2\right] I[\log(6.0)<\mu^{(j)}<\log(7.4)]}{(\sigma^{2(j)})^{-\frac{n}{2}-1}\exp\left[-\frac{1}{2\sigma^{2(j)}}\sum_{i=1}^{n}(y_i-\mu^{(j)})^2\right] I[\log(6.0)<\mu^{(j)}<\log(7.4)]}$.

7. Draw u from a $Uniform(0,1)$ distribution.

8. If $u \le r$, set $\sigma^{2(j+1)} = \sigma^{2*}$. Otherwise, set $\sigma^{2(j+1)} = \sigma^{2(j)}$.

9. Increment j and return to 1.

3.4 Implement the algorithm in Fig. 3.1. Use batch means to compute the simulation error.

```
m = 10000
pi = array(0, dim=c(m,1))
```

```
pi0 = 0.5
pi1 = pi0
for (j in 1:m)
{
pi.star = runif(1,.1,.9)
r = (pi.star^3* (1-pi.star)^8)/(pi1^3*(1-pi1)^8)
u = runif(1) <= r
pi1 = pi.star*(u==1) + pi1*(u==0)
pi[j] = pi1
}
```

To calculate the simulation error via batch means:

```
# 1. Get the means of the batches
b <- 80
v <- m/b; v  #make sure v is a whole number
means <- array(0, dim=c(b,1))
for(i in 1:b){
    means[i,1] = mean(pi[(v*i - (v-1)):(v*i)])
}

# 2. Check that lag 1 autocorrelation of batch means is less than 0.1
library(coda)
autocorr(as.mcmc(means), 1)

# 3. Compute an estimate of the simulation error
mu <- mean(means)
stderror <- sd(means)/sqrt(b)
```

The simulation standard error is 0.0019. You can also get this using `summary(as.mcmc(pi))` from the coda package.

3.5 Implement the algorithm in Fig. 3.4. Calculate the autocorrelation for the chain.

```
data <- read.table('http://www.bayesianreliability.com/wp-content/
        uploads/2009/06/table23.txt', header=T)
y <- log(data[,1])
n = length(y)
```

```
m = 5000
par = array(0, dim=c(m,2))
par1 = c(0,0.5)
s1 = 0.5; s2 = 1
for(j in 1:m){
    mu.star = rnorm(n=1, mean=par1[1], sd=s1)
    r = exp(-sum((y-mu.star)^2)/(2*par1[2]))/
       exp(-sum((y-par1[1])^2)/(2*par1[2]))
    u = runif(1) <= r
    par1[1] = mu.star*(u==1) + par1[1]*(u==0)
    nu = rnorm(1,0,sd=s2)
    var.star = par1[2]*exp(nu)
    r = (var.star^(-n/2)*exp(-sum((y-par1[1])^2)/(2*var.star)))/
        (par1[2]^(-n/2)*exp(-sum((y-par1[1])^2)/(2*par1[2])))
    u = runif(1) <= r
    par1[2] = var.star*(u==1) + par1[2]*(u==0)
    par[j,] = par1
}

# Calculate the autocorrelation
autocorrelation <- function(par, lag){
    l = lag
    A = par[1:(m-l),]
    B = par[(1+l):m,]
    mu1 = mean(par[,1])
    mu2 = mean(par[,2])
    cor1 = (sum((A[,1]-mu1)*(B[,1]-mu1))/(m-l))/
           (sum((par[,1]-mu1)^2)/m)
    cor2 = (sum((A[,2]-mu2)*(B[,2]-mu2))/(m-l))/
           (sum((par[,2]-mu2)^2)/m)
    return(c(cor1, cor2))
}
autocorrelation(par, 1)
```

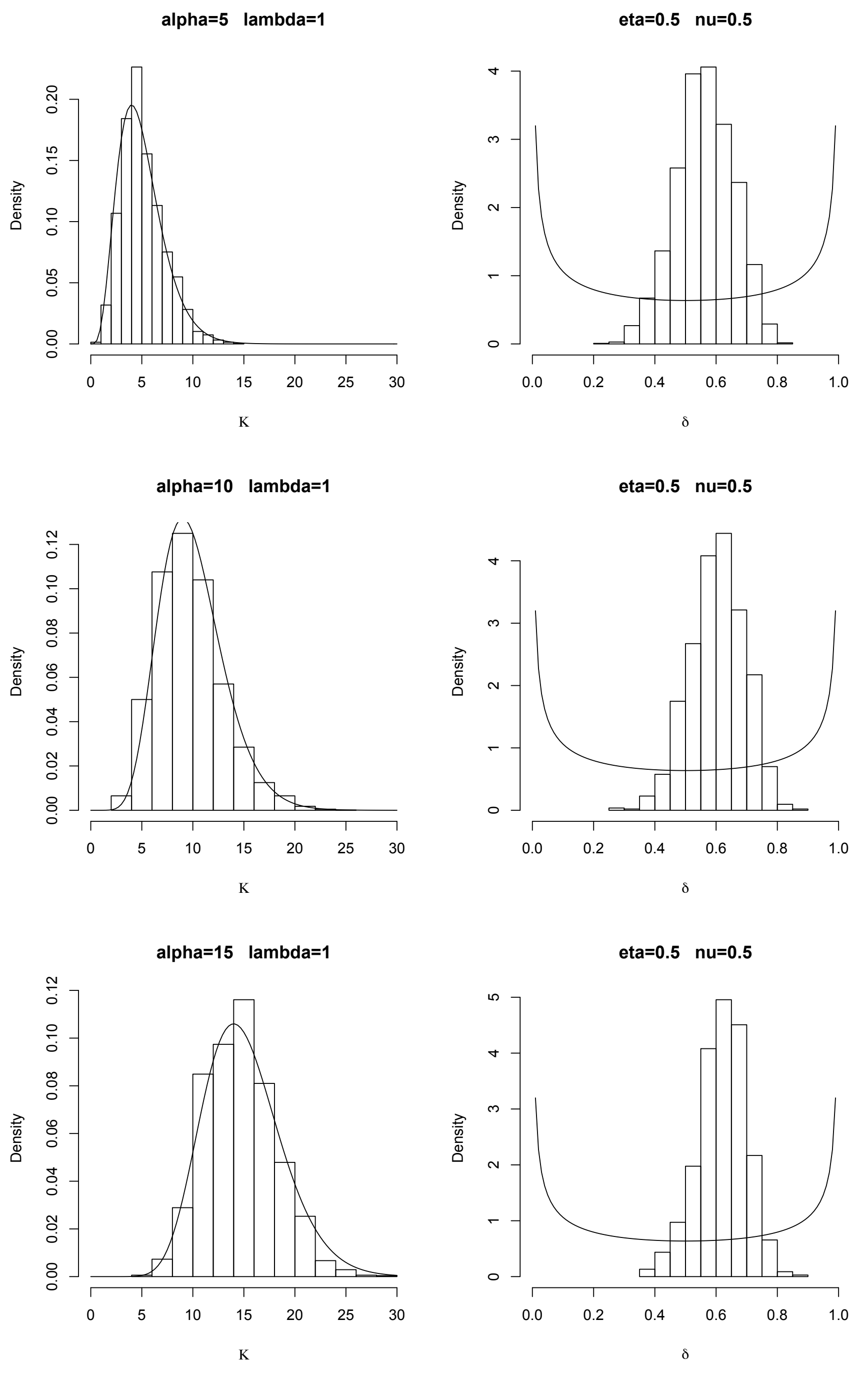
The autocorrelation for the chain (i.e. lag 1) is 0.8565168 and 0.7419024 for μ and σ^2. You can also get this using `autocorr.diag(as.mcmc(par))` from the coda package.

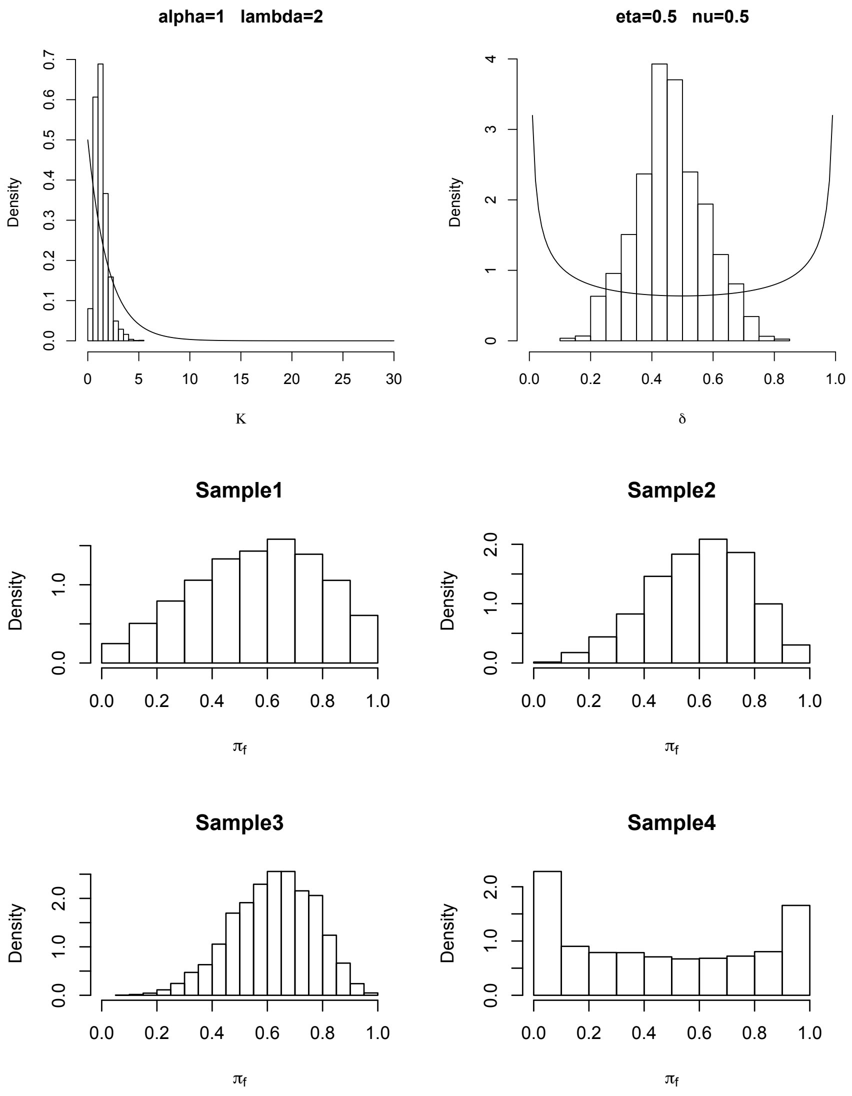
3.6 In the analysis of the launch vehicle success probabilities described in

Example 3.4, the hyperparameters α and λ were assigned values of 5 and 1, respectively.

(a) Perform a sensitivity analysis for α and λ by varying their values over a suitable range.

```
data <- read.table('http://www.bayesianreliability.com/wp-content/
    uploads/2009/06/table31.txt', header=T)
m <- data[,3]
y <- data[,2]
launch.mcmc <- function(m, y, size, alpha, lambda, eta, nu){
    K1 = alpha/lambda
    D1 = eta/(eta+nu)
    n = length(m)
    par1 = array(0, dim=c(size,2))
    par2 = array(0, dim=c(n,1))
    arateK = 0; arateD = 0
    # We use the log of the posterior for a more stable algorithm
    logpost = function(K,D){
        val=0
        for(i in 1:n){
            val = val + (y[i]+K*D-1)*log(par2[i]) +
                    (m[i]-y[i]+K-K*D-1)*log(1-par2[i])}
        val = val + n*lgamma(K) - n*lgamma(K*D) - n*lgamma(K-K*D) +
            (alpha-1)*log(K) - lambda*K + (eta-1)*log(D) + (nu-1)*log(1-D)
        return(val)
    }
    for (j in 1:size){
        for (i in 1:n){
            a = y[i] + K1*D1
            b = m[i] - y[i] + K1 - K1*D1
            par2[i] = rbeta(1,a,b)
        }
        z = rnorm(1)
        K.star = K1*exp(z)
        # use log(r) because we want to use the log of the posterior
        logr = logpost(K.star,D1) + log(K.star) - (logpost(K1,D1) + log(K1))
        u = runif(1) <= exp(logr)
        arateK = arateK + u
        K1 = K.star*(u==1) + K1*(u==0)
        c = mean(par2)
        D.star = rbeta(1, K1*c, K1*(1-c))
        logr = (logpost(K1,D.star) + (K1*c-1)*log(D1/D.star)) -
                (logpost(K1,D1) + (K1*(1-c)-1)*log((1-D.star)/(1-D1)))
        u = runif(1) <= exp(logr)
        arateD = arateD + u
        D1 = D.star*(u==1) + D1*(u==0)
        par1[j,] = c(K1,D1)
    }
    arate = c(arateK, arateD); arate = array(arate/size, dim=c(1,2))
    colnames(arate) = c("Kappa", "Delta")
    pif <- array(0, dim=c(size,1))
    for(i in 1:size){
        Kj = par1[i,1]
        Dj = par1[i,2]
        pif[i] = rbeta(1, Kj*Dj, Kj*(1-Dj))
    }
par = cbind(par1, pif)
colnames(par) = c("Kappa", "Delta", "Pif")
ans = list(par = par, accept = arate)
return(ans)
}

sample1 <- launch.mcmc(m,y,size=10000, alpha=5, lambda=1, eta=.5, nu=.5)
sample1$par <- sample1$par[-c(1:50),]
mu <- array(apply(sample1$par, 2, mean), dim=c(3,1))
st.dev <- array(apply(sample1$par, 2, sd), dim=c(3,1))
quant <- rbind(quantile(sample1$par[,1], c(.025,.05,.5,.95,.975)),
    quantile(sample1$par[,2], c(.025, .05, .5, .95, .975)),
    quantile(sample1$par[,3], c(.025, .05, .5, .95, .975)))
summary = array(c(mu, st.dev, quant), dim=c(3,7))
colnames(summary) = c("Mean", "Std Dev", "2.5%",  "5%", "50%",
    "95%", "97.5%")
rownames(summary) = c("Kappa", "Delta", "Pif")
summary
par(mfrow=c(1,2))
hist(sample1$par[,1], freq=F, xlim=c(0,30), xlab=expression(Kappa),
    main='alpha=5   lambda=1')
curve(dgamma(x, shape=5, scale=1), add=T)
hist(sample1$par[,2], xlab=expression(delta), xlim=c(0,1), freq=F,
    main='eta=0.5   nu=0.5')
curve(dbeta(x, .5,.5), add=T)

sample2 <- launch.mcmc(m,y,size=5000, alpha=10, lambda=1, eta=.5, nu=.5)
sample2$par <- sample2$par[-c(1:50),]
mu <- array(apply(sample2$par, 2, mean), dim=c(3,1))
st.dev <- array(apply(sample2$par, 2, sd), dim=c(3,1))
quant <- rbind(quantile(sample2$par[,1], c(.025,.05,.5,.95,.975)),
    quantile(sample2$par[,2], c(.025, .05, .5, .95, .975)),
    quantile(sample2$par[,3], c(.025, .05, .5, .95, .975)))
summary = array(c(mu, st.dev, quant), dim=c(3,7))
colnames(summary) = c("Mean", "Std Dev", "2.5%",  "5%", "50%",
    "95%", "97.5%")
rownames(summary) = c("Kappa", "Delta", "Pif")
summary
par(mfrow=c(1,2))
hist(sample2$par[,1], freq=F, xlim = c(0,30), xlab=expression(Kappa),
    main='alpha=10   lambda=1')
curve(dgamma(x, shape=10, scale=1), add=T)
hist(sample2$par[,2], xlab=expression(delta), xlim=c(0,1), freq=F,
    main='eta=0.5   nu=0.5')
curve(dbeta(x, .5,.5), add=T)

sample3 <- launch.mcmc(m,y,size=5000, alpha=15, lambda=1, eta=.5, nu=.5)
sample3$par <- sample3$par[-c(1:50),]
mu <- array(apply(sample3$par, 2, mean), dim=c(3,1))
st.dev <- array(apply(sample3$par, 2, sd), dim=c(3,1))
quant <- rbind(quantile(sample3$par[,1], c(.025,.05,.5,.95,.975)),
    quantile(sample3$par[,2], c(.025, .05, .5, .95, .975)),
    quantile(sample3$par[,3], c(.025, .05, .5, .95, .975)))
summary = array(c(mu, st.dev, quant), dim=c(3,7))
colnames(summary) = c("Mean", "Std Dev", "2.5%",  "5%", "50%",
    "95%", "97.5%")
rownames(summary) = c("Kappa", "Delta", "Pif")
summary
par(mfrow=c(1,2))
hist(sample3$par[,1], freq=F, xlim = c(0,30), xlab=expression(Kappa),
    main='alpha=15   lambda=1')
curve(dgamma(x, shape=15, scale=1), add=T)
hist(sample3$par[,2], xlab=expression(delta), xlim=c(0,1), freq=F,
    main='eta=0.5   nu=0.5')
curve(dbeta(x, .5,.5), add=T)

sample4 <- launch.mcmc(m,y,size=5000, alpha=1, lambda=2, eta=.5, nu=.5)
sample4$par <- sample4$par[-c(1:50),]
mu <- array(apply(sample4$par, 2, mean), dim=c(3,1))
st.dev <- array(apply(sample4$par, 2, sd), dim=c(3,1))
quant <- rbind(quantile(sample4$par[,1], c(.025,.05,.5,.95,.975)),
    quantile(sample4$par[,2], c(.025, .05, .5, .95, .975)),
    quantile(sample4$par[,3], c(.025, .05, .5, .95, .975)))
summary = array(c(mu, st.dev, quant), dim=c(3,7))
colnames(summary) = c("Mean", "Std Dev", "2.5%",  "5%", "50%",
    "95%", "97.5%")
rownames(summary) = c("Kappa", "Delta", "Pif")
summary
par(mfrow=c(1,2))
hist(sample4$par[,1], freq=F, xlim = c(0,30), xlab=expression(Kappa),
    main='alpha=1   lambda=2')
curve(dgamma(x, shape=1, scale=2), add=T)
hist(sample4$par[,2], xlab=expression(delta), xlim=c(0,1), freq=F,
    main='eta=0.5   nu=0.5')
curve(dbeta(x, .5,.5), add=T)

par(mfrow=c(2,2))
hist(sample1$par[,3], freq=F, xlim=c(0,1), xlab=expression(pi[f]),
    main='Sample1')
hist(sample2$par[,3], freq=F, xlim=c(0,1), xlab=expression(pi[f]),
    main='Sample2')
hist(sample3$par[,3], freq=F, xlim=c(0,1), xlab=expression(pi[f]),
    main='Sample3')
hist(sample4$par[,3], freq=F, xlim=c(0,1), xlab=expression(pi[f]),
    main='Sample4')
```


alpha=5 lambda=1
eta=0.5 nu=0.5
alpha=10 lambda=1
eta=0.5 nu=0.5
alpha=15 lambda=1
eta=0.5 nu=0.5
Density
K
δ


alpha=1   lambda=2
Density
K
eta=0.5   nu=0.5
Density
δ
Sample1
Density
π_f
Sample2
Density
π_f
Sample3
Density
π_f
Sample4
Density
π_f

				Quantiles	
	Parameter	Mean	Std Dev	0.05	0.95
sample1	$\kappa = 5$	5.12	2.12	2.25	9.04
	$\delta = 0.5$	0.56	0.09	0.40	0.71
	π_f	0.56	0.23	0.16	0.91
sample2	$\kappa = 10$	9.73	3.21	5.26	15.54
	$\delta = 0.5$	0.60	0.09	0.46	0.74
	π_f	0.60	0.18	0.28	0.87
sample3	$\kappa = 15$	14.62	3.49	9.40	20.82
	$\delta = 0.5$	0.62	0.08	0.48	0.74
	π_f	0.62	0.15	0.36	0.85
sample4	$\kappa = 0.5$	1.32	0.64	0.53	2.49
	$\delta = 0.5$	0.46	0.12	0.27	0.66
	π_f	0.46	0.35	0.00	0.99

An alternative to running an MCMC algorithm several times, as done above, is to do sampling importance resampling (SIR). For more information see Albert's *Bayesian Computation with R* section 5.10 (Springer). The idea is that we can take a weighted bootstrap sample (with replacement) from the current posterior distribution to get a sample from the new posterior distribution. The weights are computed by calculating $\frac{NewPosterior}{OldPosterior}$. Then convert the weights to probabilities by normalizing them to add to 1. Lastly, resample the sample from the OLD posterior using these probabilities to get a sample the NEW posterior.

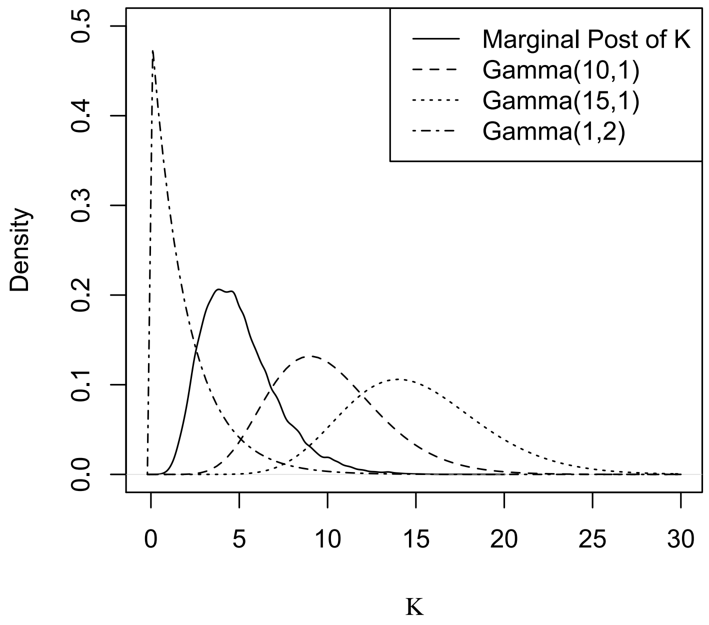
In this exercise, we are on the log scale. Since we are only interested in how changes to the prior on K affects the posterior distribution,

$$\log\left(\frac{NewPosterior}{OldPosterior}\right) = \log\left(\frac{p(K \mid \alpha, \lambda)_{NEW}}{p(K \mid \alpha, \lambda)_{OLD}}\right).$$

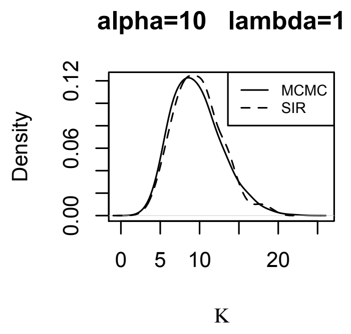
Then the weights to get an approximation to `sample2` above are

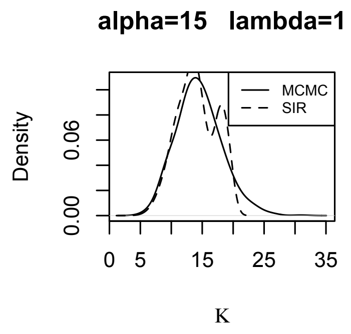
$$\begin{aligned}\log\left(\frac{p(K \mid \alpha, \lambda)_{NEW}}{p(K \mid \alpha, \lambda)_{OLD}}\right) &\propto \log\left(\frac{K^9 e^{-K}}{K^4 e^{-K}}\right) \\ &= 5 \log K.\end{aligned}$$

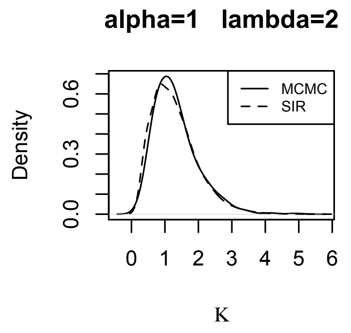
For this particular example, SIR works well for getting an approximation to `sample2` and `sample4` above. However, for `sample3`, SIR gives a poor approximation. The graph below shows the marginal posterior that we will be sampling from, along with the three new priors on K that we used for samples 2 through 4 above. With SIR, be careful that you have enough points in the tails that will be heavily weighted in the resampling. To get a decent approximation to `sample2` I had to increase my MCMC sample size to 100,000 in order to get sufficient data in the upper tail. We can also see that for `sample3`, our samples for the marginal posterior do not extend nearly far enough to get a good approximation for that distribution.



The graphs below show how the three SIR approximations (resampled from the 100,000 size MCMC) compared to the three additional MCMC runs called `sample2,` `sample3` and `sample4` above.







Here is the code used for SIR

```
# Sampling Importance Resampling (SIR)
sample1 <- launch.mcmc(m,y,size=100000, alpha=5, lambda=1, eta=.5, nu=.5)
sample1$par <- sample1$par[-c(1:50),]
lw <- 5*log(sample1$par[,1])
lw <- lw - max(lw)   #so we don't exponentiate anything too large or small
wt <- exp(lw)/sum(exp(lw))  #normalizing
ind <- sample(1:99950, replace=T, prob=wt)  #easier to sample the
s2 <- sample1$par[ind,]                     #indices of sample1

lw <- 10*log(sample1$par[,1])
lw <- lw - max(lw)
wt <- exp(lw)/sum(exp(lw))
ind <- sample(1:99950, replace=T, prob=wt)
s3 <- sample1$par[ind,]

lw <- -4*log(sample1$par[,1]) - sample1$par[,1]
lw <- lw - max(lw)
wt <- exp(lw)/sum(exp(lw))
ind <- sample(1:99950, replace=T, prob=wt)
s4 <- sample1$par[ind,]
```

(b) Report how changes in the values assumed for λ and α impact the posterior means of other model parameters.

The choice of K has a small effect on the marginal posterior distribution of δ. As we increase K, the mean of the marginal posterior of δ increases slightly. The posterior predictive mean for π_f is roughly equal to the posterior mean of δ, so our choice of K has the same effect on the posterior mean of π_f as that of δ.

3.7 Derive the conditional densities described in Example 3.4 for the random effects model.

The joint posterior distribution is

$$p(\mu, \sigma^2, \kappa, \beta \,|\, \mathbf{y}) \propto \kappa^{-9.5}(\sigma^2)^{-31}\exp\left[-\frac{0.25}{\kappa} - \frac{1}{2\sigma^2\kappa}\sum_{j=1}^{10}\beta_j^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{5}\sum_{j=1}^{10}(y_{ij} - \beta_j - \mu)^2\right].$$

So the full conditional densities are

$$\begin{aligned}
p(\mu \,|\, \beta, \sigma^2, \kappa, \mathbf{y}) &\propto \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{5}\sum_{j=1}^{10}(y_{ij}-\beta_j-\mu)^2\right] \\
&\propto \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{5}\sum_{j=1}^{10}(\mu-(y_{ij}-\beta_j))^2\right] \\
&\propto \exp\left[-\frac{1}{2\sigma^2}\left(50\mu^2-2\mu\sum_{i=1}^{5}\sum_{j=1}^{10}(y_{ij}-\beta_j)\right)\right] \\
&\propto \exp\left[-\frac{1}{2(\sigma^2/50)}\left(\mu-\frac{1}{50}\sum_{i=1}^{5}\sum_{j=1}^{10}(y_{ij}-\beta_j)\right)^2\right] \\
&\propto \frac{1}{\sqrt{2\pi(\sigma^2/50)}}\exp\left[-\frac{1}{2(\sigma^2/50)}\left(\mu-\frac{1}{50}\sum_{i=1}^{5}\sum_{j=1}^{10}(y_{ij}-\beta_j)\right)^2\right] \\
&\sim \mathit{Normal}\left(\frac{1}{50}\sum_{i=1}^{5}\sum_{j=1}^{10}(y_{ij}-\beta_j), \frac{\sigma^2}{50}\right)
\end{aligned}$$

$$
\begin{aligned}
p(\beta_j \,|\, \beta_{\mathbf{i}\neq\mathbf{j}}, \mu, \sigma^2, \kappa, \mathbf{y}) &\propto \exp\left[-\frac{1}{2\sigma^2\kappa}\sum_{j=1}^{10}\beta_j^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{5}(y_{ij}-\beta_j-\mu)^2\right] \\
&\propto \exp\left[-\frac{1}{2\sigma^2}\left(\frac{1}{\kappa}\beta_j^2 + \sum_{i=1}^{5}(\beta_j-(y_{ij}-\mu))^2\right)\right] \\
&\propto \exp\left[-\frac{1}{2\sigma^2}\left(\frac{1}{\kappa}\beta_j^2 + 5\beta_j^2 - 2\beta_j\sum_{i=1}^{5}(y_{ij}-\mu) + \sum_{i=1}^{5}(y_{ij}-\mu)^2\right)\right] \\
&\propto \exp\left[-\frac{5+1/\kappa}{2\sigma^2}\left(\beta_j - \frac{\sum_{i=1}^{5}(y_{ij}-\mu)}{5+1/\kappa}\right)^2\right] \\
&\propto \frac{1}{\sqrt{2\pi(\frac{1}{5/\sigma^2+1/(\kappa\sigma^2)})}}\exp\left[-\frac{\frac{5}{\sigma^2}+\frac{1}{\kappa\sigma^2}}{2\sigma^2}\left(\beta_j - \frac{\sum_{i=1}^{5}(y_{ij}-\mu)/\sigma^2}{\frac{5}{\sigma^2}+\frac{1}{\kappa\sigma^2}}\right)^2\right] \\
&\sim Normal\left(\frac{\sum_{i=1}^{5}(y_{ij}-\mu)/\sigma^2}{5/\sigma^2+1/(\kappa\sigma^2)}, \frac{1}{5/\sigma^2+1/(\kappa\sigma^2)}\right)
\end{aligned}
$$

$$
\begin{aligned}
p(\sigma^2 \quad | \quad & \beta, \mu, \kappa, \mathbf{y}) \\
\propto \quad & (\sigma^2)^{-31}\exp\left[-\frac{1}{2\sigma^2\kappa}\sum_{j=1}^{10}\beta_j^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{5}\sum_{j=1}^{10}(y_{ij}-\beta_j-\mu)^2\right] \\
\propto \quad & \frac{\left[\frac{1}{2}\sum_{i=1}^{5}\sum_{j=1}^{10}(y_{ij}-\beta_j-\mu)^2 + \frac{1}{2\sigma^2}\sum_{j=1}^{10}\frac{\beta_j^2}{\kappa}\right]^{30}}{\Gamma(30)}(\sigma^2)^{-(30+1)} \\
& \times\exp\left[-\frac{1}{\sigma^2}\left(\frac{1}{2}\sum_{i=1}^{5}\sum_{j=1}^{10}(y_{ij}-\beta_j-\mu)^2 + \frac{1}{2}\sum_{j=1}^{10}\frac{\beta_j^2}{\kappa}\right)\right] \\
\sim \quad & InverseGamma\left(30,\ \frac{1}{2}\sum_{i=1}^{5}\sum_{j=1}^{10}(y_{ij}-\beta_j-\mu)^2 + \frac{1}{2}\sum_{j=1}^{10}\frac{\beta_j^2}{\kappa}\right)
\end{aligned}
$$

$$
\begin{aligned}
p(\kappa\,|\,\beta,\mu,\sigma^2,\mathbf{y}) &\propto \kappa^{-(8.5+1)}\exp\left[-\frac{0.25}{\kappa}-\frac{0.5}{\sigma^2\kappa}\sum_{j=1}^{10}\beta_j^2\right]\\
&\propto \kappa^{-(8.5+1)}\exp\left[-\frac{1}{\kappa}\left(0.25+0.5\sum_{j=1}^{10}\beta_j^2/\sigma^2\right)\right]\\
&\sim InverseGamma\left(8.5, 0.25+0.5\sum_{j=1}^{10}\beta_j^2/\sigma^2\right)
\end{aligned}
$$