CHAPTER 3:

(9) Consider the following hierarchical changepoint model for the number of occurences Y_i of some event during time interval i:

$$Y_i \sim \begin{cases} Poisson(\theta), & i = 1, \ldots, k \\ Poisson(\lambda), & i = k+1, \ldots, n \end{cases}$$

$\theta \sim G(a_1, b_1)$, $\lambda \sim G(a_2, b_2)$, θ and λ independent,
$b_1 \sim IG(c_1, d_1)$, $b_2 \sim IG(c_2, d_2)$, b_1 and b_2 independent. where G denotes the gamma and IG denotes inverse gamma distributions.

(a). Apply this model to the data in Table 3.5, which gives counts of coal mining disasters in Great Britain by year from 1851 to 1962. (Here "disaster" is defined as an accident resulting in deaths of 10 or more miners). Set $a_1 = a_2 = 0.5$, $c_1 = c_2 = 1$, and $d_1 = d_2 = 1$ (a collection corresponding of "moderately informative" values). Also assume $k = 40$ (corresponding to the year 1890), and write a R program to obtain marginal posterior density estimates for θ, λ, and $R = \theta/\lambda$ using formula 3.9 with output from a Gibbs sampler.

Begin by deriving the forms of the full conditional distributions for each of the parameters $(\theta, \lambda, b_1, b_2)$. To do this, start with the full posterior distribution :

$$\left(IG(b_1|c_1, d_1) G(\theta|a_1, b_1) \prod_{i=1}^{k} Poisson(y_i|\theta) \right) \left(IG(b_2|c_2, d_2) G(\lambda|a_2, b_2) \prod_{i=k+1}^{n} Poisson(y_i|\lambda) \right)$$

$$\left(\frac{e^{-1/(d_1 b_1)} \theta^{a_1+1} e^{-\theta/b_1}}{\Gamma(c_1) d_1^{c_1} b_1^{c_1+1} \Gamma(a_1) b_1^{a_1}} \prod_{i=1}^{k} \frac{\theta^{y_i} e^{-\theta}}{y_i!} \right) \left(\frac{e^{-1/(d_2 b_2)} \lambda^{a_2+1} e^{-\lambda/b_2}}{\Gamma(c_2) d_2^{c_2} b_2^{c_2+1} \Gamma(a_2) b_2^{a_2}} \prod_{i=k+1}^{n} \frac{\lambda^{y_i} e^{-\lambda}}{y_i!} \right)$$

$$\propto \left(\frac{e^{-1/(d_1 b_1)} \theta^{a_1+1} e^{-\theta/b_1}}{\Gamma(c_1) d_1^{c_1} b_1^{c_1+1} \Gamma(a_1) b_1^{a_1}} \theta^{\sum_{i=1}^{k} y_i} e^{-\theta k} \right) \left(\frac{e^{-1/(d_2 b_2)} \lambda^{a_2+1} e^{-\lambda/b_2}}{\Gamma(c_2) d_2^{c_2} b_2^{c_2+1} \Gamma(a_2) b_2^{a_2}} \lambda^{\sum_{i=k+1}^{n} y_i} e^{-\lambda(n-k)} \right)$$

Now, combine like terms. Let's start with finding the conditional for θ:

$$\propto \theta^{a_1+1+\sum_{i=1}^{k} y_i} e^{-((\theta/b_1)+\theta k))}$$

which has the kernel of a Gamma distribution. Which gives us:

$$\mathsf{G}\left(\sum_{i=1}^{k} y_i + a_1, \frac{b_1}{b_1 k + 1} \right).$$

Similarly, you will find:

$$
\begin{aligned}
p(\theta|\boldsymbol{y}, b_1) &\equiv \mathsf{G}\left(\sum_{i=1}^{k} y_i + a_1, \frac{b_1}{b_1 k + 1}\right), \\
p(\lambda|\boldsymbol{y}, b_2) &\equiv \mathsf{G}\left(\sum_{i=k+1}^{n} y_i + a_2, \frac{b_2}{b_2(n-k) + 1}\right), \\
p(b_1|\theta) &\equiv \mathsf{IG}\left(a_1 + c_1, \frac{d_1}{d_1\theta + 1}\right), \text{ and} \\
p(b_1|\lambda) &\equiv \mathsf{IG}\left(a_2 + c_2, \frac{d_2}{d_2\lambda + 1}\right),
\end{aligned}
$$

where the above distributions are defined using the parametrization in the textbook. Because all of these are familiar forms from which we can readily sample in `R`, the Gibbs sampler is straight-forward.

Rao-Blackwell Hints: NOT REQUIRED
Hatfield: We can visualize the posteriors simply using histograms and kernel density estimates. However, to implement the Rao-Blackwellization of (3.9), we must average the densities.

Specifically, the Rao-Blackwellization of (3.9) states:

$$
\approx \frac{1}{m(T - t_o)} \sum_{j=1}^{m} \sum_{i=t_o+1}^{T} p(\theta_i | \theta^{(\mathbf{t})}_{\mathbf{l} \neq \mathbf{i},\mathbf{j}}, \mathbf{y})
$$

To do so for $R = \theta/\lambda$, we can be think of it as a function of either λ or θ. Taking R as a function of θ and conditioning on λ simply yields another gamma distribution, scaled by $1/\lambda$. By familiar properties of Gamma random variables, we know that

$$
p(R|\boldsymbol{y}, \lambda, b_1) \equiv \mathsf{G}\left(\sum_{i=1}^{k} y_i + a_1, \frac{b_1}{\lambda(b_1 k + 1)}\right).
$$

One numerical solution to density averaging is to consider bins along the support of the parameter, find the density value of each posterior MCMC sample in that bin, average these density values, and plot these density averages. This method works well for λ, θ and R. Code is provided in the supplemental file to produce Figure 1. The kernel (solid) and Rao-Blackwellized (dashed) density curves in Figure 1 are nearly indistinguishable, so little advantage is provided by the density smoothing approach.

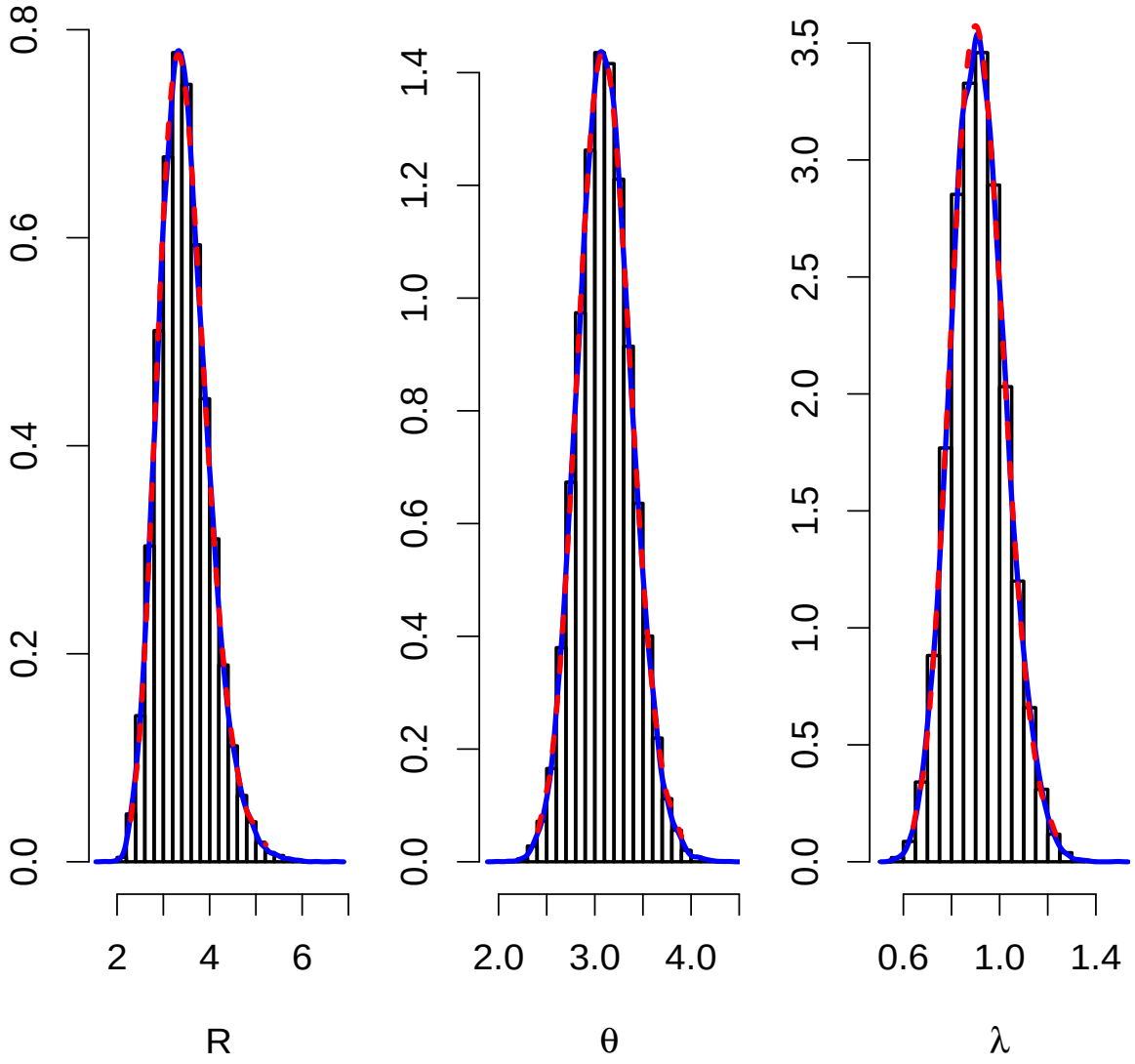

Figure 1: Histograms and density estimates for posteriors in coal model with k known using Gibbs sampler in R. Solid blue lines are kernel smoothers, dashed red lines are Rao-Blackwellized densities.

```
coal <- read.table("coal_data.txt",header=FALSE)
names(coal) <- c('i','y'); attach(coal)

###Part A: Using R
N = dim(coal)[1]; K = 40; a1 = a2 = .5; c1 = c2 = 1; d1 = d2 = 1
G = 20000 #Number of posterior samples
b1 = b2 = 1 #Initial values (just need them for b1 & b2)

#Create object to store posterior samples
post = matrix(NA,nrow=G,ncol=5)
colnames(post) = c("R","theta","lambda","b1","b2")

#Gibbs Sampler
for (g in 1:G){
#Sample theta given K & b1
theta = rgamma(1,shape=a1 + sum(y[1:K]),scale=1/(K+1/b1))
#Sample lambda given K & b2
lambda = rgamma(1,shape=a2 + sum(y[(K+1):N]),scale=1/(N-K+1/b2))
#Sample b1 given theta
b1 = 1/rgamma(1,shape=a1+c1,scale=1/(theta + 1/d1))
#Sample b2 given lambda
b2 = 1/rgamma(1,shape=a2+c2,scale=1/(lambda + 1/d2))
#Calculate R
R = theta/lambda
post[g,] = rbind(R,theta,lambda,b1,b2) #Save samples
}

#Check Traceplots
plot(post[,1],type='l')
plot(post[,2],type='l')
plot(post[,3],type='l')
plot(post[,4],type='l')
plot(post[,5],type='l')
#Looks Great

#Posterior Summaries
res = round(t(apply(post,2,function(x) c(mean(x),sd(x),quantile(x,c(0.025,.5,0.975))))),2)
rownames(res) = c("R","theta","lambda","b1","b2")
colnames(res) = c("Mean","SD","2.5%","50%","97.5%")
print(res)

#Calculate Rao-Blackwellized Posterior Density Estimates
#For theta
sorted <- cbind(post[order(post[,"theta"]),],bins=rep(seq(1,G,length.out=100),each=G/100))
RB.theta <- tapply(sorted[,"theta"],sorted[,'bins'],function(x)
mean(dgamma(x,shape=sum(y[1:K])+a1,scale=1/(K+1/sorted[,'b1'']))))
midpoints.theta <- tapply(sorted[,"theta"],sorted[,'bins'],mean)
out.theta <- cbind(x=midpoints.theta,y=RB.theta)

#For lambda
sorted <- cbind(post[order(post[,"lambda"]),],bins=rep(seq(1,G,length.out=100),each=G/100))
RB.lambda<- tapply(sorted[,"lambda"],sorted[,'bins'],function(x)
mean(dgamma(x,shape=sum(y[(K+1):N])+a2,scale=1/(N-K+1/sorted[,'b2']))))
midpoints.lambda<- tapply(sorted[,"lambda"],sorted[,'bins'],mean)
out.lambda<- cbind(x=midpoints.lambda,y=RB.lambda)

#For R
sorted <- cbind(post[order(post[,"R"]),],bins=rep(seq(1,G,length.out=100),each=G/100))
RB.R <- tapply(sorted[,"R"],sorted[,'bins'],function(x)
mean(dgamma(x,shape=sum(y[1:K])+a1,scale=1/(sorted[,'lambda']*(K+1/sorted[,'b1'])))))
midpoints.R <- tapply(sorted[,"R"],sorted[,'bins'],mean)
out.R <- cbind(x=midpoints.R,y=RB.R)

##Plot Densities for various posterior density estimators
# Plot histograms and kernel density estimates
pdf("coal_1_hist.pdf")
```

```
par(mfrow=c(1,3),lwd=2,cex=1.3,mar=c(5,2,1,1))
hist(post[,"R"],breaks=G/1000,freq=FALSE,main="",ylab="",xlab=expression(R))
lines(density(post[,"R"]),lwd=3,col='blue')
lines(out.R,lty=2,lwd=3,col='red')

hist(post[,"theta"],breaks=G/1000,freq=FALSE,main="",ylab="",xlab=expression(theta))
lines(density(post[,"theta"]),lwd=3,col='blue')
lines(out.theta,lty=2,lwd=3,col='red')

hist(post[,"lambda"],breaks=G/1000,freq=FALSE,main="",ylab="",xlab=expression(lambda))
lines(density(post[,"lambda"]),lwd=3,col='blue')
lines(out.lambda,lty=2,lwd=3,col='red')
dev.off()
```

(b). Re-do this problem in Winbugs. Are your answers comparable?
Hint: We recommend doing this in R2Winbugs or RJAGS. Create a table comparing the mean, median, sd, 2.5, and 97.5 quantiles of each parameter.

We recommend starting with a model code something like this:

```
for (i in 1:K){
 Y[i] ~ dpois(theta)
}
for (i in K+1:N){
 Y[i] ~ dpois(lambda)
}
theta ~ dgamma(a[1],inv.b1) #Second parameter is
#1/b1 in the BUGS parameterization
#Continue to enter other priors
}
```

Then, you may think of using estimates similar to what you found in (a) for initial values.

ANSWER
Now fitting the model in `JAGS` using `R2jags` we see very similar results. Figure 2 displays dashed and dotted lines for the R kernel and R-B density curves, respectively, while the solid line is the kernel density based on `JAGS` posterior samples. Table 2 displays summary statistics from the `R` sampler and from the `JAGS` output. The results are very similar for all parameters.

```
# Part B: Using BUGS/JAGS
write("model{
for (i in 1:K){
 Y[i] ~ dpois(theta)
}
for (i in K+1:N){
 Y[i] ~ dpois(lambda)
}
theta ~ dgamma(a[1],inv.b1) #Second parameter is 1/b1 in the BUGS parameterization
lambda ~ dgamma(a[2],inv.b2) #Second parameter is 1/b2 in the BUGS parameterization
inv.b1 ~ dgamma(c[1],1/d[1]) #Second parameter is 1/d1 in the BUGS parameterization
inv.b2 ~ dgamma(c[2],1/d[2]) #Second parameter is 1/d2 in the BUGS parameterization
b1 <- 1/inv.b1
b2 <- 1/inv.b2
R <- theta/lambda
}","coal_model.txt")
```

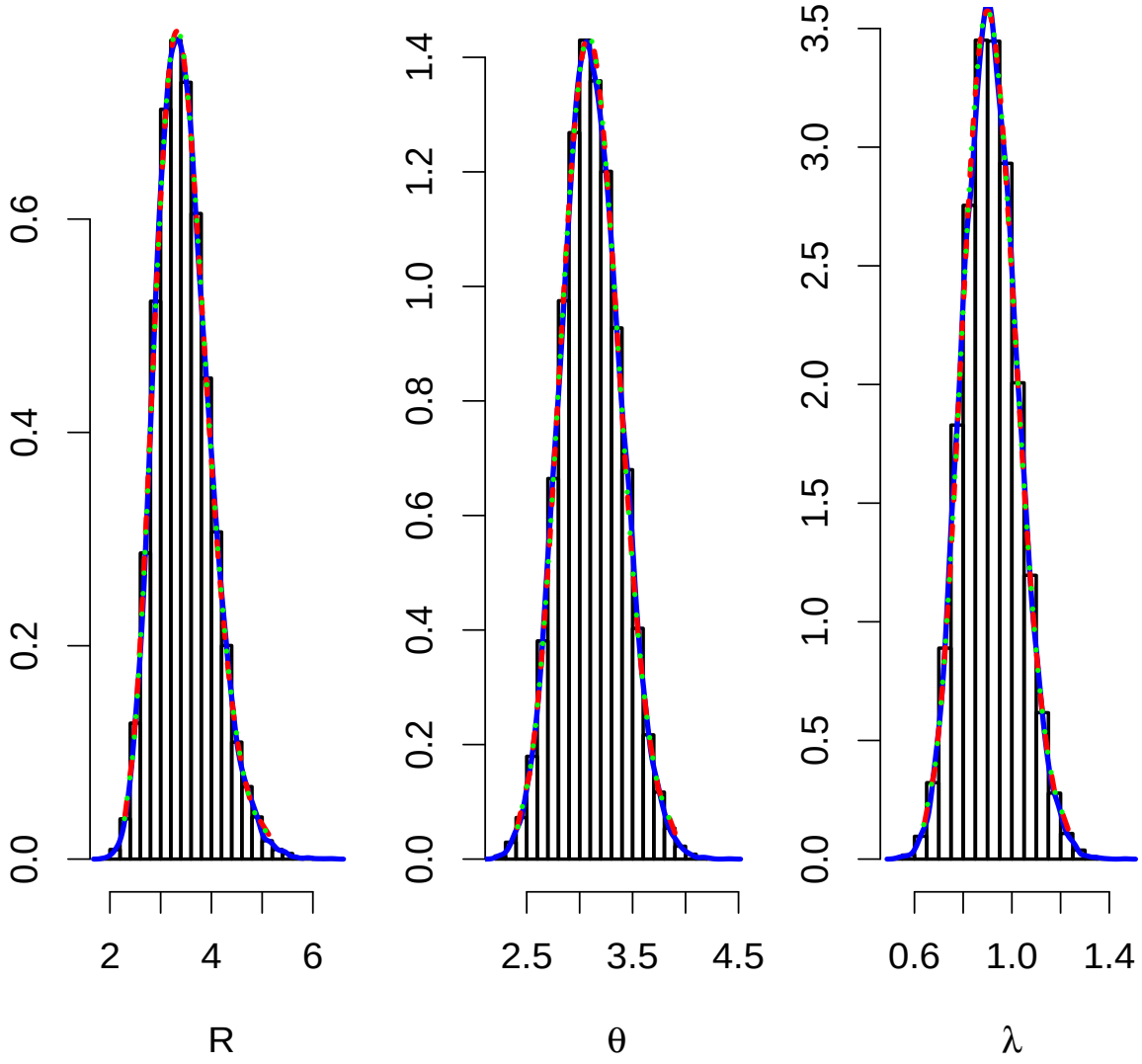

Figure 2: Histograms and density estimates for posteriors in coal model with k known using Gibbs sampler in `R` compared to Gibbs sampler in `JAGS`. Solid blue lines are kernel smoothers of the `JAGS` results, dashed red lines are R-B estimates from `R`, and dotted lines are R-B estimates from `JAGS`.

Table 1: Question 9 Posterior Results

	Post Mean	Post SD	Post 2.5%	Post 50%	Post 97.5%
R Results					
R	3.45	0.54	2.53	3.41	4.64
θ	3.11	0.28	2.59	3.10	3.67
λ	0.91	0.11	0.70	0.91	1.15
b_1	7.98	42.14	0.87	3.49	37.10
b_2	3.70	18.52	0.41	1.58	16.76
JAGS Results					
R	3.46	0.53	2.54	3.41	4.64
θ	3.11	0.28	2.58	3.10	3.67
λ	0.91	0.11	0.71	0.91	1.14
b_1	8.20	77.83	0.79	3.49	38.73
b_2	3.99	48.49	0.37	1.60	17.54

```
#Initialize at good values since we know them
ints <- function(){
list(theta=mean(post[,"theta"]),lambda=mean(post[,"lambda"]),
inv.b1=1/mean(post[,c("b1")]), inv.b2=1/mean(post[,c("b2")]))
}
dta = list(Y=coal[,'y'],N=dim(coal)[1],K=40,a=c(.5,.5),c=c(1,1),d=c(1,1))
pars = c('R','theta','lambda','b1','b2')

#BUGS
#coal.fit <- bugs(data=dta,inits=ints,parameters.to.save=pars,model.file="coal_model.txt",
# n.chains = 2, n.iter = 10200, n.burnin = 200, n.thin=1, DIC=TRUE)

#JAGS
coal.fit <- jags(data=dta,inits=ints,parameters=pars,model.file="coal_model.txt",
 n.chains = 2, n.iter = 10200, n.burnin = 200, n.thin=1, DIC=TRUE)

post.jags <- coal.fit$BUGSoutput$sims.matrix

#Calculate Rao-Blackwellized Posterior Density Estimates
#For theta
sorted <- cbind(post.jags[order(post.jags[,"theta"]),],
bins=rep(seq(1,G,length.out=100),each=G/100))
RB.theta <- tapply(sorted[,"theta"],sorted[,'bins'],function(x)
mean(dgamma(x,shape=sum(y[1:K])+a1,scale=1/(K+1/sorted[,'b1']))))
midpoints.theta <- tapply(sorted[,"theta"],sorted[,'bins'],mean)
out.theta.jags <- cbind(x=midpoints.theta,y=RB.theta)

#For lambda
sorted <- cbind(post.jags[order(post.jags[,"lambda"]),],
bins=rep(seq(1,G,length.out=100),each=G/100))
RB.lambda <- tapply(sorted[,"lambda"],sorted[,'bins'],function(x)
mean(dgamma(x,shape=sum(y[(K+1):N])+a2,scale=1/(N-K+1/sorted[,'b2']))))
midpoints.lambda <- tapply(sorted[,"lambda"],sorted[,'bins'],mean)
out.lambda.jags <- cbind(x=midpoints.lambda,y=RB.lambda)

#For R
sorted <- cbind(post.jags[order(post.jags[,"R"]),],
bins=rep(seq(1,G,length.out=100),each=G/100))
RB.R <- tapply(sorted[,"R"],sorted[,'bins'],function(x)
```

```
mean(dgamma(x,shape=sum(y[1:K])+a1,scale=1/(sorted[,'lambda']*(K+1/sorted[,'b1'])))))
midpoints.R <- tapply(sorted["R"],sorted[,'bins'],mean)
out.R.jags <- cbind(x=midpoints.R,y=RB.R)

# Plot histograms and kernel density estimates
pdf("coal_2_hist.pdf")
par(mfrow=c(1,3),lwd=2,cex=1.3,mar=c(5,2,1,1))
hist(post.jags[,"R"],breaks=G/1000,freq=FALSE,main="",ylab="",xlab=expression(R))
lines(density(post.jags[,"R"]),lwd=3,col='blue')
lines(out.R,lty=2,lwd=3,col='red')
lines(out.R.jags,lty=3,lwd=3,col='green')

hist(post.jags[,"theta"],breaks=G/1000,freq=FALSE,main="",ylab="",xlab=expression(theta))
lines(density(post.jags[,"theta"]),lwd=3,col='blue')
lines(out.theta,lty=2,lwd=3,col='red')
lines(out.theta.jags,lty=3,lwd=3,col='green')

hist(post.jags[,"lambda"],breaks=G/1000,freq=FALSE,main="",ylab="",xlab=expression(lambda))
lines(density(post.jags[,"lambda"]),lwd=3,col='blue')
lines(out.lambda,lty=2,lwd=3,col='red')
lines(out.lambda.jags,lty=3,lwd=3,col='green')
dev.off()

#Compile Results from 2 Approaches
res.gibbs = res
res.jags1 = t(apply(post.jags,2,function(x) c(mean(x),sd(x),quantile(x,c(.025,.5,.975)))))
res.jags = round(rbind(res.jags1[1,],res.jags1[6:5,],res.jags1[2:3,]),2)
res.coal = rbind(res.gibbs,rep(NA,5),res.jags)
colnames(res.coal) = c("Post Mean","Post SD","Post 2.5%","Post 50%","Post 97.5%")
rownames(res.coal) = c(c("R","theta","lambda","b_1","b_2"),NA,c("R","theta","lambda","b_1","b_2"))
print(res.coal)
```

(11) In problem 9, replace the third-stage prior given above with
$b_1 \sim G(c_1, d_1)$, $b_2 \sim G(c_2, d_2)$, b_1 and b_2 independent,
thus destroying the conjugacy for these two complete conditionals. Resort to rejection or Metropolis-Hastings subsampling for these two components instead, choosing appropriate values for c_1,c_2,d_1, and d_2. What is the effect on the posterior for b_1 and b_2? For R? For k?

ANSWER: Only one method is required

Hatfield: **RJAGS or Winbugs**
We begin by using Gamma priors on b_1 and b_2 in `JAGS`, noting that the program will use Metropolis sampling, due to the non-conjugate priors. We choose $c_1 = c_2 = 1$ and $d_1 = d_2 = 100$ so that $E(b_i|c_i, d_i) = 100$, $\sigma(b_i|c_i, d_i) = 100$. Note that `JAGS` and `WinBUGS` uses rate (i.e., 1/scale) as the second parameter of a Gamma distribution.

Compared to using conjugate IG priors above, the posteriors for k and R have changed little. In contrast, the posterior distributions of b_1 and b_2 are more sensitive to the choice of prior. The new distributions are much more sharply peaked, reflecting the fact that the values for c_i and d_i in the previous problem resulted in hyperpriors having limiting means and variances equal to infinity. The current hyperpriors have finite mean and variance, so the posteriors of b_1 and b_2 are correspondingly tighter. Changing the (arbitrarily selected) hyperparameters could change this result, of course.

OR: METROPOLIS WITHIN R
We can similarly modify our `R` sampler to incorporate a block Metropolis step for updating b_1 and b_2. Considering only the terms involving b_1 and b_2, the unnormalized likelihood $\times$ prior is

$$h(b_1, b_2) = \exp\left(-\theta/b_1 - \lambda/b_2 - b_1/d_1 - b_2/d_2\right)\ b_1^{c_1-a_2-1}\ b_2^{c_2-a_2-1}$$

Then we make the change of variable $u = \log(b_1)$; $v = \log(b_2)$ so that $h()$ is defined on $\mathscr{R}^2$. The Jacobian of the transformation is $J = \exp(u + v)$ so that the unnormalized likelihood $\times$ prior follows as

$$g(u, v) = h(e^u, e^v)e^{u+v}.$$

We draw candidate values (u^*, v^*) from $\mathscr{N}\left\{(\log(b_1), \log(b_2))^T, \sigma^2 I\right\}$, where $\sigma^2 = 6$ after preliminary investigation to provide an approximately 30% acceptance ratio. See subsequent `R` code for further details.

Examining Table 4, we can see that the Metropolis in Gibbs method produces very similar parameter estimates as `JAGS` in the case of R and k, but not b_1 and b_2. This is further evidence of the robustness of this model to prior specification. That is, even when parameters at the third stage of the model differ, the second-stage

parameter posterior distributions remain similar. (It also makes me question the validity of the M-H normal candidate density, perhaps another sampling approach would be more suited for this type of full conditional.)

Table 2: Question 11 Posterior Results, k unknown with non-conjugate priors

	Post Mean	Post SD	Post 2.5%	Post 50%	Post 97.5%
R Metropolis within Gibbs Posterior Results					
R	3.42	0.53	2.51	3.38	4.58
θ	3.13	0.29	2.59	3.12	3.74
λ	0.93	0.12	0.71	0.92	1.17
b_1	64.69	72.49	2.90	39.98	264.58
b_2	58.80	72.20	1.38	32.57	256.76
K	39.93	2.43	36.00	40.00	46.00
JAGS Results					
R	3.46	0.54	2.53	3.41	4.64
θ	3.09	0.29	2.55	3.08	3.68
λ	0.91	0.12	0.69	0.90	1.15
b_1	0.50	0.43	0.03	0.38	1.69
b_2	1.69	1.47	0.12	1.29	5.64
k	40.05	2.36	36.00	40.00	46.00

```
write("model{
for (i in 1:N){
 Y[i] ~ dpois(mu[i])
 mu[i] <- gamma[J[i]] #gamma[1] = theta and gamma[2] = lambda
 J[i] <- 1 + step(i - K - 0.5) # J=1 if i <= K and J=2 if i > K
 punif[i] <- 1/N #Uniform probabilities
}
K ~ dcat(punif[]) #Discrete Uniform
gamma[1] ~ dgamma(a[1],inv.b1) # previously theta
gamma[2] ~ dgamma(a[2],inv.b2) # previously lambda
inv.b1 <- b1
inv.b2 <- b2
b1 ~ dgamma(c[1],1/d[1])
b2 ~ dgamma(c[2],1/d[2])
R <- gamma[1]/gamma[2]
}","coal_model3.txt")

#Initialize at good values since we know them
ints3 <- function(){
list(gamma=as.vector(colMeans(post2[,c("theta","lambda")])),
b1=mean(post2[,c("b1")]), b2=mean(post2[,c("b2")]))
}
dta3 = list(Y=coal[,'y'],N=dim(coal)[1],a=c(.5,.5),c=c(1,1),d=c(100,100))
pars3 = c('R','gamma','b1','b2','K')

#BUGS
#coal.fit3 <- bugs(data=dta3,inits=ints3,parameters.to.save=pars3,model.file="coal_model3.txt",
# n.chains = 2, n.iter = 10200, n.burnin = 200, n.thin=1, DIC=TRUE)

#JAGS
coal.fit3 <- jags(data=dta3,inits=ints3,parameters=pars3,model.file="coal_model3.txt",
 n.chains = 2, n.iter = 10200, n.burnin = 200, n.thin=1, DIC=TRUE)
```

```
traceplot(coal.fit3) #Looks Great!

post3.jags <- coal.fit3$BUGSoutput$sims.matrix

#Look at the density for b1, b2, K, and R
#For b1, compared to the conjugate setting
plot(density(post3.jags[,"b1"]),lwd=2,xlim=c(0,50))
lines(density(post2.jags[,"b1"]),lwd=2,lty=2)

#For b2, compared to the conjugate setting
plot(density(post3.jags[,"b2"]),lwd=2,xlim=c(0,50))
lines(density(post2.jags[,"b2"]),lwd=2,lty=2)

#For K, compared to the conjugate setting
post.prob = sapply(1:N,function(x) mean(post3.jags[,"K"]==x))
plot(post.prob,pch=20) #Highly concentrated about 40
post.probz = sapply(1:N,function(x) mean(post2.jags[,"K"]==x))
points(post.probz,pch=20,col='blue')
#Very little difference!

#For R, compared to conjugate setting
plot(density(post3.jags[,"R"]),lwd=2)
lines(density(post2.jags[,"R"]),lwd=2,lty=2)
#Very little difference

#Compile Results
res3.jags = round(t(apply(post3.jags[,c("R","gamma[1]","gamma[2]","b1","b2","K")],2,function(x)
c(mean(x),sd(x),quantile(x,c(.025,.5,.975))))),2)
colnames(res3.jags) = c("Post Mean","Post SD","Post 2.5%","Post 50%","Post 97.5%")
rownames(res3.jags) = c("R","theta","lambda","b1","b2","K")
print(res3.jags)

##By Hand using Gibbs with M-H substeps for b1 and b2
N = dim(coal)[1]; a1 = a2 = .5; c1 = c2 = 1; d1 = d2 = 100
G = 20000 #Number of posterior samples
K = 40; b1 = b2 = 1 #Initial values (just need them for b1 & b2)

##Create object to store posterior samples
post3 = matrix(NA,nrow=G,ncol=7)
colnames(post3) = c("R","theta","lambda","b1","b2","K","Accept")
##M-H candidate density variance tuner (set so that acceptance is ~30%)
sig.star = 6

#Gibbs Sampler
for (g in 1:G){
#Sample theta given K & b1
theta = rgamma(1,shape=a1 + sum(y[1:K]),scale=1/(K+1/b1))
#Sample lambda given K & b2
lambda = rgamma(1,shape=a2 + sum(y[(K+1):N]),scale=1/(N-K+1/b2))
#Calculate R
R = theta/lambda

#Calculate full conditional of K given theta & lambda
numer = rep(NA,N) #Object to fill
for (i in 1:(N-1)){
 numer[i] = exp(sum(dpois(y[1:i],theta,log=TRUE))+sum(dpois(y[(i+1):N],lambda,log=TRUE)))
}
numer[N] = exp(sum(dpois(y[1:N],theta,log=TRUE)))
#Sample K given theta & lambda
K = sample(1:N,1,prob=numer/sum(numer))

#Multivariate M-H block substep
log.b.star = mvrnorm(1,log(c(b1,b2)),sig.star*diag(2)) #Draw Candidate Value
u = runif(1) #Draw Uniform
#Calculate Ratio
r = exp( (c1-a1)*(log.b.star[1]-log(b1)) + (c2-a2)*(log.b.star[2]-log(b2)) + (b1-exp(log.b.star[1]))/d1 +
```

```
theta*(1/b1-1/exp(log.b.star[1])) + (b2-exp(log.b.star[2]))/d2 + lambda*(1/b2-1/exp(log.b.star[2])))
if(r >= u){ b1 = exp(log.b.star[1]); b2 = exp(log.b.star[2]); accept = 1 } #Decision Rule
#(if r >= u accept, o.w. retain b1)
if(r < u) accept = 0 #Leave b1, b2 alone

#Save gth samples
post3[g,] = rbind(R,theta,lambda,b1,b2,K,accept) #Save samples
}

#Check Acceptance Rate of M-H substep(s)
mean(post3[,"Accept"])

#Check Traceplots
plot(post3[,1],type='l')
plot(post3[,2],type='l')
plot(post3[,3],type='l')
plot(post3[,4],type='l')
plot(post3[,5],type='l')
plot(post3[,6],type='l')
#Looks Okay

#Look at the density for K, R, theta, and lambda
post.prob = sapply(1:N,function(x) mean(post3[,"K"]==x))
plot(post.prob) #Highly concentrated about 40

#For R, compared to conjugate setting
plot(density(post3[,"R"]),lwd=2)
lines(density(post2[,"R"]),lwd=2,lty=2)

#For theta, compared to conjugate setting
plot(density(post3[,"theta"]),lwd=2)
lines(density(post2[,"theta"]),lwd=2,lty=2)

#For lambda, compared to conjugate setting
plot(density(post3[,"lambda"]),lwd=2)
lines(density(post2[,"lambda"]),lwd=2,lty=2)
#All Very Similar

#Compile Results
res3 = round(t(apply(post3[,-c(7,8)],2,function(x) c(mean(x),sd(x),quantile(x,c(.025,.5,.975))))),2)
colnames(res3) = c("Post Mean","Post SD","Post 2.5%","Post 50%","Post 97.5%")
rownames(res3) = c("R","theta","lambda","b1","b2","K")
print(res3)
```

CHAPTER 4:

(6) Consider the data displayed in Table 4.6, originally collected by Treloar (1974) and reproduced in Bates and Watts (1988). These data record the "velocity" y_i of an enzymatic reaction (in counts/min/min) as a function of substrate concentration x_i (in ppm), where the enzyme has been treated with puromycin.
A common model for analyzing biochemical kinetics data of this type is the *Michaelsis-Menten* model, wherein we adopt the mean structure

$$\mu_i = \gamma + \alpha x_i/(\theta + x_i),$$

where $\alpha, \gamma \in \Re$ and $\theta \in \Re^+$. IN the nomenclature of Example 4.6, we have design matrix ...
For this model, obtain estimates of the marginal posterior density of the parameter α, and also the marginal posterior density of the mean velocity at $X = 0.5$, a concentration not represented in the original data, assuming that the error density is:
(a) normal
(b) Student's t with 2 degrees of freedom
(c) double exponential

HINT:
To get around the duplication within `BUGS` issue, just duplicate the data outside of `BUGS` and supply it as data. You can use the "NA" trick with $X = 0.5$, or just add a node to your code to predict for a $Y|X = 0.5$.

There are a few ways to code this problem:
Method 1:

```
X = c(rep(c(.02,.06,.11,.22,.56,1.10),each=2),.5)
Y = c(76,47,97,107,123,139,159,152,191,201,207,200,NA)
Y = matrix(rep(Y,3),13,3)
N = length(X)

model.function = function(){
 for(m in 1:3){ #Model Loop
  for(i in 1:N){
   Y[i,m] ~ dnorm(mu[i,m],tau[m]/lambda[i,m]) #Model m
   mu[i,m] <- gamma[m] + alpha[m]*X[i]/(theta[m]+X[i])
  }
  #Vague Priors for All Parameters in All Models
  ...
 }
 #Priors for lambda[i]'s
 for(i in 1:N){
  lambda[i,1] <- 1 #M1 = e_i ~ N(0,sigma[1]^2)
  lambda[i,2] <- inv.lambda2[i] # So that lambda[i,2] ~ IG
  inv.lambda2[i] ~ dgamma(1,1) #M2 = e_i ~ t_2(0,sigma[2]^2)
  lambda[i,3] ~ dexp(0.5) #M3 = e_i ~ DE(0,sigma[3]^2)
 }
}
```

where DE is double exponential. Here we are allowing you do model all three error densities at the same time. RJAGS is nice to do this in.

Method 2:
You could also fit t2 and DE directly as follows

```
model.function = function(){
  for(i in 1:N){
   Y[i,1] ~ dnorm(mu[i,1],tau[1]) #Model 1
   mu[i,1] <- gamma[1] + alpha[1]*X[i]/(theta[1]+X[i])
   Y[i,2] ~ dt(mu[i,2],tau[2],2) #Model 2
   mu[i,2] <- gamma[2] + alpha[2]*X[i]/(theta[2]+X[i])
   Y[i,3] ~ ddexp(mu[i,3],tau[3]) #Model 3
   mu[i,3] <- gamma[3] + alpha[3]*X[i]/(theta[3]+X[i])
  }
  #Vague Priors for All Parameters in All Models
 for(m in 1:3){
...
 }
}
```

ANSWER:

Table 3: Posterior Mean and 95% HPD CIs

	α	$Y\|X = 0.5$
Normal	188.6 (161.3, 216.7)	187.7 (162.5, 212.1)
t_2	184.4 (156.7, 218.9)	187.1 (156.0, 217.6)
DE	185.3 (157.4, 221.3)	187.2 (156.6, 216.9)

Table 1 contains the posterior estimates for α and $Y|X = 0.5$. The point estimates are lower for both these parameters for the t_2 and DE errors, though only hardly at all for the predictions. The major difference are the much wider 95% HPD intervals for non-normal error models. (The question actually seems to ask for the posterior of $E(Y|X = 0.5)$, which naturally has tighter credible intervals but the same point estimates). It is also worth noting that these estimates are quite sensitive to the prior on α, if you used `dflat()` in `WinBUGS` the results change quite substantially. α is very large here, so the `dnorm(0,0.0001)` priors that I've used are actually moderately informative!

```
###############
# Problem 4.6 #
###############
X = c(rep(c(.02,.06,.11,.22,.56,1.10),each=2),.5)
Y = c(76,47,97,107,123,139,159,152,191,201,207,200,NA)
Y = matrix(rep(Y,3),13,3)
N = length(X)

model.function = function(){
 for(m in 1:3){ #Model Loop
  for(i in 1:N){
   Y[i,m] ~ dnorm(mu[i,m],tau[m]/lambda[i,m]) #Model m
   mu[i,m] <- gamma[m] + alpha[m]*X[i]/(theta[m]+X[i])
  }
  #Vague Priors for All Parameters in All Models
  gamma[m] ~ dnorm(0,0.0001)
  alpha[m] ~ dnorm(0,0.0001)
  theta[m] ~ dgamma(0.01,0.01)
  tau[m] <- 1/(sigma[m]*sigma[m])
  sigma[m] ~ dunif(0.01,100)
 }
```

```
 #Priors for lambda[i]'s
 for(i in 1:N){
  lambda[i,1] <- 1 #M1 = e_i ~ N(0,sigma[1]^2)
  lambda[i,2] <- inv.lambda2[i] # So that lambda[i,2] ~ IG
  inv.lambda2[i] ~ dgamma(1,1) #M2 = e_i ~ t_2(0,sigma[2]^2)
  lambda[i,3] ~ dexp(0.5) #M3 = e_i ~ DE(0,sigma[3]^2)
 }
}

jags.dat = list("N", "Y", "X")
jags.pars = c("alpha","gamma","theta","Y[13,1]","Y[13,2]","Y[13,3]")
jags.fit6abc = jags(data=jags.dat,inits=NULL,model.file=model.function,
parameters=jags.pars,n.chains=2,n.iter=40000,n.burnin=10000,n.thin=1,jags.seed=123)

post.alphas = apply(jags.fit6abc$BUGSoutput$sims.matrix[,c("alpha[1]","alpha[2]","alpha[3]")],2,
function(x) c(mean(x),quantile(x,c(0.025,0.975))))
post.alphas #t_2 and DE are similar, lower point estimates and wider CIs than N model

post.Y13s = apply(jags.fit6abc$BUGSoutput$sims.matrix[,c("Y[13,1]","Y[13,2]","Y[13,3]")],2,
function(x) c(mean(x),quantile(x,c(0.025,0.975))))
post.Y13s #t_2 and DE are similar, slightly lower point estimates and much wider CIs than N model

#You could also fit t2 and DE directly as follows
model.function = function(){
  for(i in 1:N){
   Y[i,1] ~ dnorm(mu[i,1],tau[1]) #Model 1
   mu[i,1] <- gamma[1] + alpha[1]*X[i]/(theta[1]+X[i])
   Y[i,2] ~ dt(mu[i,2],tau[2],2) #Model 2
   mu[i,2] <- gamma[2] + alpha[2]*X[i]/(theta[2]+X[i])
   Y[i,3] ~ ddexp(mu[i,3],tau[3]) #Model 3
   mu[i,3] <- gamma[3] + alpha[3]*X[i]/(theta[3]+X[i])
  }
  #Vague Priors for All Parameters in All Models
 for(m in 1:3){
  gamma[m] ~ dnorm(0,0.0001)
  alpha[m] ~ dnorm(0,0.0001)
  theta[m] ~ dgamma(0.01,0.01)
  tau[m] <- 1/(sigma[m]*sigma[m])
  sigma[m] ~ dunif(0.01,100)
 }
}

jags.dat = list("N", "Y", "X")
jags.pars = c("alpha","gamma","theta","Y[13,1]","Y[13,2]","Y[13,3]")
jags.fit6abc2 = jags(data=jags.dat,inits=NULL,model.file=model.function,
parameters=jags.pars,n.chains=2,n.iter=50000,n.burnin=10000,n.thin=1,jags.seed=123)
traceplot(jags.fit6abc2)

post.alphas = apply(jags.fit6abc2$BUGSoutput$sims.matrix[,c("alpha[1]","alpha[2]","alpha[3]")],2,
function(x) c(mean(x),quantile(x,c(0.025,0.975))))
post.alphas #t_2 and DE are similar, lower point estimates and wider CIs than N model

post.Y13s = apply(jags.fit6abc2$BUGSoutput$sims.matrix[,c("Y[13,1]","Y[13,2]","Y[13,3]")],2,
function(x) c(mean(x),quantile(x,c(0.025,0.975))))
post.Y13s #t_2 and DE are similar, slightly lower point estimates and much wider CIs than N model
#The t2 model is a bit sketchy in terms of convergence...so be careful
```

(15) In the previous problem, implement both models in WinBUGS, and use DIC instead of Bayes factors to choose between them. Is your preference between the two models materially altered by this change? What are the advantages and disadvantages of using DIC instead of Bayes factor here?

HINT:
You do not need to re-calculate Bayes factor. You can use the solution in the back of the book for question 14. However, you should calculate DIC. Use a code similar to this:

```
model{
 for (i in 1:42){
  y[i] ~ dnorm(mu[i],tau)
  mu[i] <- alpha+beta*(x[i]-mean(x[])) #Model a
#  mu[i] <- alpha+beta*(z[i]-mean(z[])) #Model b
 }
 alpha ~ dnorm(3000,1.0E-6)
 beta ~ dnorm(185,1.0E-4)
 tau ~ dgamma(3,180000)
}
```

Note: Here I am giving you the model. You should understand how this model was obtained based on problem 14. You should make a table of the DIC values and Bayes Factors. Make sure you answer the questions!

ANSWER:
Model 2 fits the data substantially better with a much smaller $\bar{D}$. Since the size of the two models are the same, the DIC for the second model is much lower. If you go through the effort of computing the Bayes factor, you'll find that they provide essentially the same information. The Bayes factor requires much more work with DIC being readily available in all the MCMC programs we've been using.

Table 4: Model Summaries

	$\bar{D}$	p_D	DIC
Model 1	609.49	2.77	612.26
Model 2	592.44	2.85	595.29

Using the answer for the Bayes factor in problem 14 from the back of the book and the following identity, we can readily verify that DIC and the Bayes factor provide the same information in this setting.

$$-16.98 = -2\log(4862) = -2\log(BF) = 2\log(f(y|M_2)) - 2\log(f(y|M_1)) \approx$$

$$\bar{D}_2 - \bar{D}_1 \approx DIC_2 - DIC_1 = 595.33 - 612.31 = -16.98$$

```
###############
#Question 4.15#
###############
dat = read.table("http://www.biostat.umn.edu/~brad/data/pine_data.txt",header=FALSE)
colnames(dat) = c("y","x","z")
y=dat$y;x=dat$x;z=dat$z
```

```
model.function = function(){
 for (i in 1:42){
  y[i] ~ dnorm(mu[i],tau)
  mu[i] <- alpha+beta*(x[i]-mean(x[])) #Model a
#  mu[i] <- alpha+beta*(z[i]-mean(z[])) #Model b
 }
 alpha ~ dnorm(3000,1.0E-6)
 beta ~ dnorm(185,1.0E-4)
 tau ~ dgamma(3,180000)
}

jags.dat = list("y", "x")
jags.pars = c("alpha", "beta", "tau")

jags.fit15a = jags(data=jags.dat,inits=NULL,model.file=model.function,
parameters=jags.pars,n.chains=2,n.iter=12000,n.burnin=2000,n.thin=1,jags.seed=123)

#Comment in relevant line in model.function above
jags.dat = list("y", "z")
jags.fit15b = jags(data=jags.dat,inits=NULL,model.file=model.function,
parameters=jags.pars,n.chains=2,n.iter=12000,n.burnin=2000,n.thin=1,jags.seed=123)

dics = cbind(c(jags.fit15a$BUGSoutput$mean$deviance,
jags.fit15a$BUGSoutput$DIC,jags.fit15a$BUGSoutput$pD),
c(jags.fit15b$BUGSoutput$mean$deviance,
jags.fit15b$BUGSoutput$DIC,jags.fit15b$BUGSoutput$pD))
rownames(dics) = c("D.bar","DIC","pD")
colnames(dics) = c("Mod a","Mod b")
dics
#Model b fits better!

#BF = 4862;
-2*log(4862)
dic[1,2]-dic[1,1]
#Pretty consistent!
```

(16) Consider again the stack loss data originally presented in Example 2.16. Suppose we wish to compare the assumptions of normal, t_4, and DE errors for these data.

(a). Repeat the outlier analysis in Example 4.7 for these data. Do the λ_i posteriors for the non-normal models effectively identify outliers?
(b). Use the Comparison tool to obtain a side-by-side comparison of the boxplots of the posterior median θ across the three error models, similar to Figure 4.9. Then do a similar comparison of the width of the central 95% credible interval for the errors, $q_{0.975} - q_{0.025}$ and explain any differences.
(c). Repeat the DIC analysis in Example 4.8 for these data. What model is DIC-best? Are negative p_D values again a problem here?
HINT:
You will be creating at least 3 of the 5 models with these specifications on lambda or direct specification:
1). Normal

```
lambda[i] <- 1
```

2). t_4 scale mixture:

```
lambda[i] <- 1/inv.lambda[i]
 inv.lambda[i] ~ dgamma(2,2)
```

3). DE scale mixture:

```
lambda[i] ~ dexp(0.5)
```

4). t_4 Direct Fit:

```
Y[i] ~ dt(mu[i],tau,4)
```

5. DE Direct Fit

```
Y[i] ~ ddexp(mu[i],tau)
```

We would recommend standardizing Xs outside in R rather than in BUGS: To Do that use the following:

```
Y<-c(42, 37, 37, 28, 18, 18, 19, 20, 15, 14, 14, 13, 11, 12, 8, 7, 8, 8, 9, 15, 15)
N<-length(Y); p<-3
x = matrix(c(80, 27, 89, 80, 27, 88, 75, 25, 90, 62, 24, 87, 62, 22, 87,
62, 23, 87, 62, 24, 93, 62, 24, 93, 58, 23, 87, 58, 18, 80, 58, 18, 89, 58, 17, 88,
58, 18, 82, 58, 19, 93, 50, 18, 89, 50, 18, 86, 50, 19, 72, 50, 19, 79, 50, 20, 80,
56, 20, 82, 70, 20, 91),ncol=3,byrow=TRUE)

#Standardize X's In R rather than BUGS
z = apply(x,2,function(x) (x-mean(x))/sd(x))
```

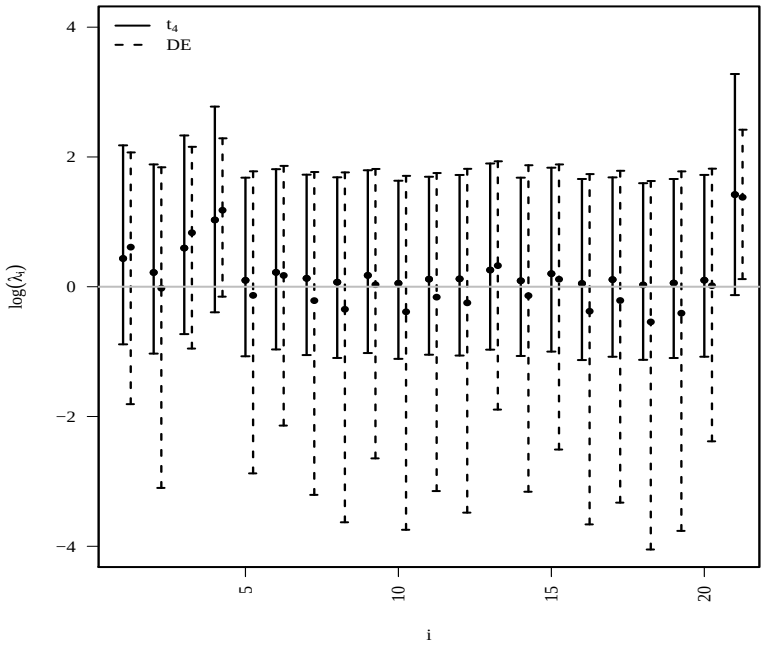

Figure 3: Posterior 95% HPD CIs of $\log(\lambda_i)$'s under t_4 (solid) and double exponential (dashed) error distributions

ANSWER:
(a) Figure 2 displays posterior whisker plots of the λ_i's on the log-scale from the two mixture models. Observation 4 and 21 appear to be a marginal outliers, especially under the double exponential error model. There is slightly less evidence that these observations are outliers under the t_4 error model. Recall, in the exact CPO analysis from before, observation 21 really stood out. It is less obvious here, so this method may not be quite as effective at identifying outliers, but it is certainly less burdensome than looping over the same model N times.

(b) This question was a bit misleading as the original example did not require any covariate adjustments, so there was just a single θ. Now we have many θ_i's (I call these μ_i's in the code below), so I just take a look at all of them as well as the regression coefficients. There isn't really any discernible pattern in Figure 3 across the three models with respect to the θ_i's. Perhaps an interesting difference is that the fatter tailed distributions result in tighter intervals for the persons with relatively low estimates, and wider intervals for the persons with relatively high estimates. We do see some changes in the behavior with respect to the regression coefficients across models, most notably the tighter posteriors of β_3 for the t_4 and double exponential models.

Lastly, to compare the standard deviation estimates from the three models, we convert the Gibbs samples from the σ parameters under each (direct fit) model to the standard deviation scale. For the normal, this is simply σ since σ^2 is the

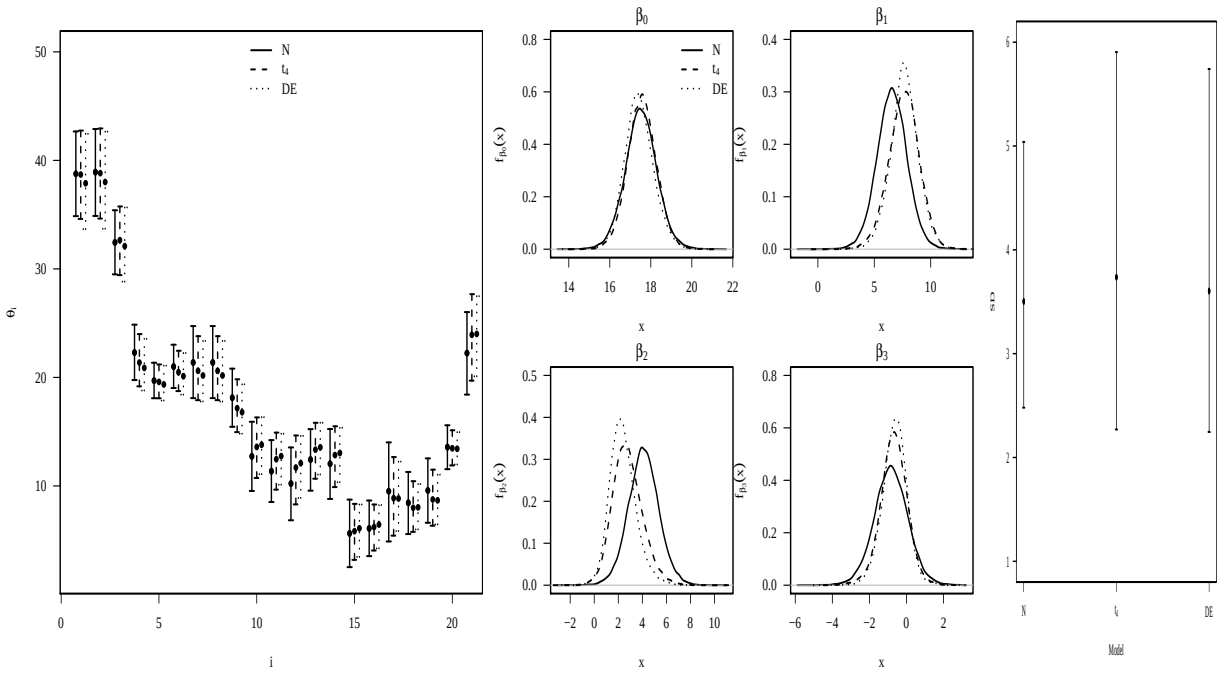

Figure 4: The left panel contains posterior densities of θ_{11} (left side of left panel) and θ_{21} (right side of left panel) under normal (left), t_4 (middle) and double exponential (right) error distributions. The right panel contains posterior densities of the regression coefficients β_p's.

variance for the normal distribution. For the t_4 and double exponential however, the variance is given by $\frac{\nu}{(\nu-2)\tau}$ and $2\tau^{-2}$, respectively. Thus, plugging in $\nu = 4$, the standard deviation for the t_4 model is $\sqrt{2\sigma^2} = \sqrt{2}\sigma$ and for the DE model we have $\sqrt{2}\sigma^2$. Transforming the Gibbs samples for σ to the standard deviation scale using these identities and looking at the posterior means and 95% credible intervals along with their widths, we see in the right panel of Figure 3 that the heavier tailed distributions have a slightly larger standard deviation estimate as well as less certainty than the normal error model. The point estimates aren't terribly different, the major difference is the increased width of the 95% credible intervals.

(c) Finally, comparing the model summaries in Table 3 we no longer see negative p_D values, but these values seem to be inflated for the mixture fits relative to their direct counterparts. Because of this, the DIC's for models 2 and 3 should not be trusted. Comparing the direct fit summaries, we see that the double exponential and t_4 error distribution models are slightly preferred over the conventional normal errors model.

Table 5: Model Fit Summary Statistics

Model	$\bar{D}$	$\hat{D}$	p_D	DIC
M1 (Normal)	110.62	103.23	7.39	118.01
M2 (t_4, normal scale mixture)	103.51	83.05	20.46	123.98
M3 (DE, normal scale mixture)	99.30	75.81	23.49	122.79
M4 (t_4, direct fit)	109.08	101.56	7.52	116.60
M5 (DE, direct fit)	108.11	99.99	8.12	116.22

```
################
#Question 4.16#
################
Y<-c(42, 37, 37, 28, 18, 18, 19, 20, 15, 14, 14, 13, 11, 12, 8, 7, 8, 8, 9, 15, 15)
N<-length(Y); p<-3
x = matrix(c(80, 27, 89, 80, 27, 88, 75, 25, 90, 62, 24, 87, 62, 22, 87,
62, 23, 87, 62, 24, 93, 62, 24, 93, 58, 23, 87, 58, 18, 80, 58, 18, 89, 58, 17, 88,
58, 18, 82, 58, 19, 93, 50, 18, 89, 50, 18, 86, 50, 19, 72, 50, 19, 79, 50, 20, 80,
56, 20, 82, 70, 20, 91),ncol=3,byrow=TRUE)

#Standardize X's In R rather than BUGS
z = apply(x,2,function(x) (x-mean(x))/sd(x))

#Comment in/out relevant lines, and re-run for 5 different models
model.function = function(){
 for (i in 1:N){
  mu[i] <- beta[1] + beta[2]*z[i, 1] +  beta[3]*z[i, 2] + beta[4]*z[i, 3]
## Models a-c ##
  Y[i] ~ dnorm(mu[i],tau/lambda[i])
## Model a ##
#  lambda[i] <- 1 # Normal
## Model b ##
#  lambda[i] <- 1/inv.lambda[i]
#  inv.lambda[i] ~ dgamma(2,2) # t_4, scale mixture
## Model c ##
#  lambda[i] ~ dexp(0.5) # DE, scale mixture
```

```
## Model b2 ##
#  Y[i] ~ dt(mu[i],tau,4) # t_4, direct fit
## Model c2 ##
#  Y[i] ~ ddexp(mu[i],tau) # DE, direct fit
}
 for(p in 1:4){
  beta[p] ~ dnorm(0,0.00001)
 }
 tau <- 1/(sigma0*sigma0)
 sigma0 ~ dunif(0.01,100)
## Model a ##
 sigma <- sigma0 #N standard deviation
## Model b2 ##
# sigma <- sqrt(2)*sigma0 # t_4 & DE standard deviation
## Model c2 ##
# sigma <- sqrt(2)*pow(sigma0,2) # t_4 & DE standard deviation
}

jags.dat = list("Y", "N", "z")
jags.pars = c("beta", "sigma", "lambda", "mu")

#Normal Mixture Formulation
jags.fit16a = jags(data=jags.dat,inits=NULL,model.file=model.function,
parameters=jags.pars,n.chains=2,n.iter=12000,n.burnin=2000,n.thin=1,jags.seed=123)

jags.pars = c("beta", "lambda","mu")

#t4 Mixture Formulation
jags.fit16b = jags(data=jags.dat,inits=NULL,model.file=model.function,
parameters=jags.pars,n.chains=2,n.iter=12000,n.burnin=2000,n.thin=1,jags.seed=123)

#DE Mixture Formulation
jags.fit16c = jags(data=jags.dat,inits=NULL,model.file=model.function,
parameters=jags.pars,n.chains=2,n.iter=12000,n.burnin=2000,n.thin=1,jags.seed=123)

jags.pars = c("beta", "sigma")

#t4 Direct Formulation
jags.fit16b2 = jags(data=jags.dat,inits=NULL,model.file=model.function,
parameters=jags.pars,n.chains=2,n.iter=12000,n.burnin=2000,n.thin=1,jags.seed=123)

#DE Direct Formulation
jags.fit16c2 = jags(data=jags.dat,inits=NULL,model.file=model.function,
parameters=jags.pars,n.chains=2,n.iter=12000,n.burnin=2000,n.thin=1,jags.seed=123)

#Plot lambda_i's from Models 2 & 3 on log scale
pdf("Lambdas.pdf",family="serif")
par(mfrow=c(1,1),lwd=2,cex=1,mar=c(4.8,4.8,1,1),las=2)
post.lambdas = apply(log(jags.fit16b$BUGSoutput$sims.matrix[,sapply(1:N,function(i)
paste("lambda[",i,"]",sep=""))]),2,function(x) c(mean(x),quantile(x,c(0.025,0.975))))
plotCI(1:N, post.lambdas[1,], li=post.lambdas[2,],
ui=post.lambdas[3,],sfrac=.005,gap=0,pch=20,ylab=expression(log(lambda[i])),xlab="i",ylim=c(-4,4))
post.lambdas = apply(log(jags.fit16c$BUGSoutput$sims.matrix[,sapply(1:N,function(i)
paste("lambda[",i,"]",sep=""))]),2,function(x) c(mean(x),quantile(x,c(0.025,0.975))))
plotCI(1:N+.25, post.lambdas[1,], li=post.lambdas[2,],
ui=post.lambdas[3,],sfrac=.005,gap=0,pch=20,add=TRUE,lty=2)
abline(h=0,col='grey')
legend("topleft",c(expression(t[4]),"DE"),lty=1:2,bty='n')
dev.off()

#mu Comparisons
pdf("Thetas.pdf",family="serif")
par(mfrow=c(1,1),lwd=2,cex=1,mar=c(4.2,4.2,1.5,0),las=1)
post.mus = apply(jags.fit16a$BUGSoutput$sims.matrix[,sapply(1:N,function(i)
paste("mu[",i,"]",sep=""))],2,function(x) c(mean(x),quantile(x,c(0.025,0.975))))
plotCI(1:N - 0.25, post.mus[1,], li=post.mus[2,],
```

```
ui=post.mus[3,],sfrac=.005,gap=0,pch=20,ylab=expression(theta[i]),xlab="i",ylim=c(2,50))
post.mus = apply(jags.fit16b$BUGSoutput$sims.matrix[,sapply(1:N,function(i)
paste("mu[",i,"]",sep=""))],2,function(x) c(mean(x),quantile(x,c(0.025,0.975))))
plotCI(1:N, post.mus[1,], li=post.mus[2,], ui=post.mus[3,],
sfrac=.005,gap=0,pch=20,lty=2,add=TRUE)
post.mus = apply(jags.fit16c$BUGSoutput$sims.matrix[,sapply(1:N,function(i)
paste("mu[",i,"]",sep=""))],2,function(x) c(mean(x),quantile(x,c(0.025,0.975))))
plotCI(1:N + 0.25, post.mus[1,], li=post.mus[2,], ui=post.mus[3,],
sfrac=.005,gap=0,pch=20,lty=3,add=TRUE)
legend("top",c("N",expression(t[4]),"DE"),lty=1:3,bty='n')
dev.off()

#Beta Comparisons
pdf("Betas.pdf",family="serif")
par(mfrow=c(2,2),lwd=2,cex=1,mar=c(4.2,4.2,1.5,0),las=1)
plot(density(jags.fit16a$BUGSoutput$sims.matrix[,"beta[1]"]),ylim=c(0,.8),
xlab="x",ylab=expression(f[beta[0]](x)),main=expression(beta[0]))
lines(density(jags.fit16b$BUGSoutput$sims.matrix[,"beta[1]"]),lty=2)
lines(density(jags.fit16c$BUGSoutput$sims.matrix[,"beta[1]"]),lty=3)
legend("topright",c("N",expression(t[4]),"DE"),lty=1:3,bty='n')

plot(density(jags.fit16a$BUGSoutput$sims.matrix[,"beta[2]"]),ylim=c(0,.4),
xlab="x",ylab=expression(f[beta[1]](x)),main=expression(beta[1]))
lines(density(jags.fit16b$BUGSoutput$sims.matrix[,"beta[2]"]),lty=2)
lines(density(jags.fit16c$BUGSoutput$sims.matrix[,"beta[2]"]),lty=3)

plot(density(jags.fit16a$BUGSoutput$sims.matrix[,"beta[3]"]),ylim=c(0,.5),
xlab="x",ylab=expression(f[beta[2]](x)),main=expression(beta[2]))
lines(density(jags.fit16b$BUGSoutput$sims.matrix[,"beta[3]"]),lty=2)
lines(density(jags.fit16c$BUGSoutput$sims.matrix[,"beta[3]"]),lty=3)

plot(density(jags.fit16a$BUGSoutput$sims.matrix[,"beta[4]"]),ylim=c(0,.8),
xlab="x",ylab=expression(f[beta[3]](x)),main=expression(beta[3]))
lines(density(jags.fit16b$BUGSoutput$sims.matrix[,"beta[4]"]),lty=2)
lines(density(jags.fit16c$BUGSoutput$sims.matrix[,"beta[4]"]),lty=3)
dev.off()

#Standard Deviation Comparisons
sds = cbind(jags.fit16a$BUGSoutput$sims.matrix[,'sigma'],
jags.fit16b2$BUGSoutput$sims.matrix[,'sigma'],
jags.fit16c2$BUGSoutput$sims.matrix[,'sigma'])
post.sds = apply(sds,2,function(x) c(mean(x),quantile(x,c(0.025,0.975))))

pdf("SDs.pdf",family="serif")
par(mfrow=c(1,1),lwd=2,cex=1,mar=c(4.8,4.8,1,1),las=1)
plotCI(1:3, post.sds[1,], li=post.sds[2,], ui=post.sds[3,],sfrac=.005,gap=0,
pch=20,ylab="SD",xlab="Model",ylim=c(1,6),xaxt='n')
axis(1,lab=c("N",expression(t[4]),"DE"),at=1:3,las=1)
dev.off()

#DIC Comparison
dics = cbind(c(jags.fit16a$BUGSoutput$DIC,jags.fit16a$BUGSoutput$pD),
c(jags.fit16b2$BUGSoutput$DIC,jags.fit16b2$BUGSoutput$pD),
c(jags.fit16c2$BUGSoutput$DIC,jags.fit16c2$BUGSoutput$pD))
colnames(dics) = c("N","t4","DE")
rownames(dics) = c("DIC","pD")
dics
#Double Exponential Fits a Little Better
```