1. $Y_1, \cdots, Y_k | \theta \overset{iid}{\sim} \text{Poisson}(\theta)$

$Y_{k+1}, \cdots, Y_n | \lambda \overset{iid}{\sim} \text{Poisson}(\lambda)$

$$\left.\begin{array}{l} \theta \sim \text{Gamma}(a_1, \frac{1}{b_1}) \\ \lambda \sim \text{Gamma}(a_2, \frac{1}{b_2}) \end{array}\right\} \text{indep}$$

$b_1 \sim \text{IGamma}(c_1, \frac{1}{d_1})$

$b_2 \sim \text{IGamma}(c_2, \frac{1}{d_2})$

$$P(\theta, \lambda, b_1, b_2 | \underset{\sim}{y}) \propto \prod_{i=1}^{k} P(y_i|\theta) \times \prod_{i=k+1}^{n} P(y_i|\lambda) \times P(\theta|b_1) P(\lambda|b_2) \times P(b_1) P(b_2)$$

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

Full conditionals are

$$P(\theta | \underset{\sim}{y}, \lambda, b_1, b_2) \propto e^{-k\theta} \theta^{\sum_{i=1}^{k} y_i} \theta^{a_1-1} e^{-\frac{1}{b_1}\theta} = \theta^{\sum_{i=1}^{k} y_i + a_1 - 1} e^{-(k + \frac{1}{b_1})\theta}$$

$$\Rightarrow \text{Gamma}\left(\sum_{i=1}^{k} y_i + a_1, \ k + \frac{1}{b_1}\right)$$

$$P(\lambda | \underset{\sim}{y}, \theta, b_1, b_2) \propto e^{-(n-k)\lambda} \lambda^{\sum_{i=k+1}^{n} y_i} \lambda^{a_2-1} e^{-\frac{1}{b_2}\lambda} = \lambda^{\sum_{i=k+1}^{n} y_i + a_2 - 1} e^{-(n-k+\frac{1}{b_2})\lambda}$$

$$\Rightarrow \text{Gamma}\left(\sum_{i=k+1}^{n} y_i + a_2 - 1, \ n - k + \frac{1}{b_2}\right)$$

$$P(b_1 | \underset{\sim}{y}, \theta, \lambda, b_2) \propto \left(\frac{1}{b_1}\right)^{a_1} e^{-\frac{\theta}{b_1}} b_1^{-(c_1+1)} e^{-\frac{1}{d_1 b_1}} = b_1^{-(a_1+c_1+1)} e^{-(\theta + \frac{1}{d_1})\frac{1}{b_1}}$$

$$\Rightarrow \text{IGamma}\left(a_1 + c_1, \ \theta + \frac{1}{d_1}\right)$$

$$P(b_2 | \underset{\sim}{y}, \theta, \lambda, b_1) \propto \left(\frac{1}{b_2}\right)^{a_2} e^{-\frac{\lambda}{b_2}} b_2^{-(c_2+1)} e^{-\frac{1}{d_2 b_2}} = b_1^{-(a_2+c_2+1)} e^{-(\lambda + \frac{1}{d_2})\frac{1}{b_2}}$$

$$\Rightarrow \text{IGamma}\left(a_2 + c_2, \ \lambda + \frac{1}{d_2}\right).$$

```
library(invgamma)

setwd('C://Users/yunhy/Desktop/data')
data = read.table('coal_data.txt')

y = data[,2]

gibbs <- function(n.sims, beta.start, alpha, gamma, delta, y, k, burnin, thin) {

  theta.draws <- rep(NA, n.sims-burnin)
  lambda.draws <- rep(NA, n.sims-burnin)
  beta1.cur <- beta.start
  beta2.cur <- beta.start

  theta.update <- function(alpha, beta, y) {
    rgamma(length(y), shape = sum(y[1:k]) + alpha, scale = beta/(k*beta+1))
  }

  lambda.update <- function(alpha, beta, y) {
    rgamma(length(y), shape = sum(y[(k+1):length(y)]) + alpha,
           scale = beta/((length(y)-k)*beta+1))
  }

  beta.update <- function(alpha, gamma, delta, theta, y) {
    rinvgamma(length(y), shape = alpha + gamma, scale = delta/(theta*delta+1))
  }

  for (i in 1:n.sims) {
    theta.cur <- theta.update(alpha = alpha, beta = beta1.cur, y = y)
    beta1.cur <- beta.update(alpha = alpha, gamma = gamma, delta = delta,
                             theta = theta.cur, y = y)
    lambda.cur <- lambda.update(alpha = alpha, beta = beta2.cur, y = y)
    beta2.cur <- beta.update(alpha = alpha, gamma = gamma, delta = delta,
                             theta = lambda.cur, y = y)

    if (i > burnin & (i - burnin)%%thin == 0) {
      theta.draws[(i - burnin)/thin] <- theta.cur
      lambda.draws[(i - burnin)/thin] <- lambda.cur
    }
  }

  return(list(theta.draws = theta.draws, lambda.draws = lambda.draws))
}

set.seed(0)
draws = gibbs(n.sims = 10000, beta.start = 1, alpha = 0.5, gamma = 1, delta = 1,
              y = y, k=40, burnin = 1000, thin = 1)

theta = draws$theta.draws
lambda = draws$lambda.draws
mean(theta); var(theta)

## [1] 3.106735

## [1] 0.07586734

mean(lambda); var(lambda)

## [1] 0.9158031

## [1] 0.01230237

mean(theta/lambda); var(theta/lambda)

## [1] 3.443558

## [1] 0.2785831
```

2. See HW3-Problem2.R

(a) First, note for this problem I have used `set.seed()` in the R code. This sets the seed of the random number generator, so that each time I run the code, it gives the same random values. Isn't that sneaky? But it helps me give some consistent answers I guess. Anyway:

```
[1] 1.018692
[1] 0.2114948
```

```
SHOULD have mean 1, and variance 1/5
```

Nothing to lose sleep over.
(b) I made a grid, and used `image()` to create a little display of the posterior, see Figure 4. More to the point:

```
P( ALPHA < 5) =  0.6874
P( BETA < 5) =  0.7217
```

(c) Alright:

```
Normalizing constant:  0.006529946
```

Small! But not to worry, this is related to the joint distribution of 100 variables, and any particular combination will be rather unlikely. Actually any combination will be a probability zero event, so the statement of the previous sentence is just a way to think about it but it is in no way true.

(d) The sampler lies within the R file, the simple answer lie below:

```
POSTERIOR BETA MEAN =  4.916263
POSTERIOR BETA VARIANCE =  0.04912721
P(BETA < 5 | alpha=5, Y) =  0.655
```

You may start to panic here, as the probability seems not to line up with our grid approximation, but that's because we've fixed α here.
(e) The sampler lies within the R file, the simple answers lie below:

```
POSTERIOR ALPHA MEAN =  5.062664
POSTERIOR ALPHA VARIANCE =  0.04606942
P(ALPHA < 5 | beta=5, Y) =  0.401
```

Figure 5 displays some of the details and diagnostics of the Metropolis sampler.

About the proposal distribution: we know that in any case, the likelihood is the dominant factor in the distribution $p(\alpha|\beta, Y)$. Since the data has mean near 1, we know that whatever β is equal to, α should also have about that value. Also, we can guess that maybe the standard deviation should also be about the same as the data. So, we could try:

```
proposal <- rnorm(1, currentValue[2] * mean(Y), propSd)
```

Where in the above, `currentValue[2]` denotes the current value of β, and `propSd` is `sd(Y)`.
(f) The sampler lies within the R file, the simple answers lie below:

```
POSTERIOR ALPHA, BETA MEAN =  4.711935 4.624612
POSTERIOR ALPHA, BETA VARIANCE =  0.4214031 0.4590967
P(ALPHA < 5 | Y) =  0.687
```

In the end, the probability lines up quite nicely with that obtained with the grid approximation.

Figure 6 displays some of the details and diagnostics of the Metropolis sampler. I chose a normal proposal distribution, with standard deviation one. This worked really badly, I had to burn every 180 draws. That is really terrible, but I guess it worked OK in the end.

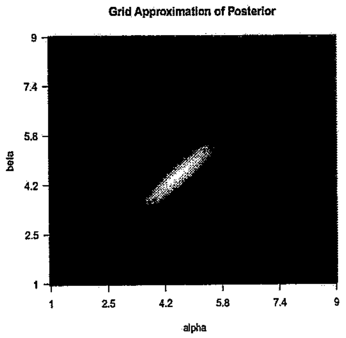

Figure 4: Image of the posterior density, approximated with a grid. We see a mode at roughly 5, 5. Thank god, because this are the parameters of the distribution the data comes from. It appears unimodal, and the two coordinates are highly correlated. Unrelated to anything that matters, it resembles a galaxy.

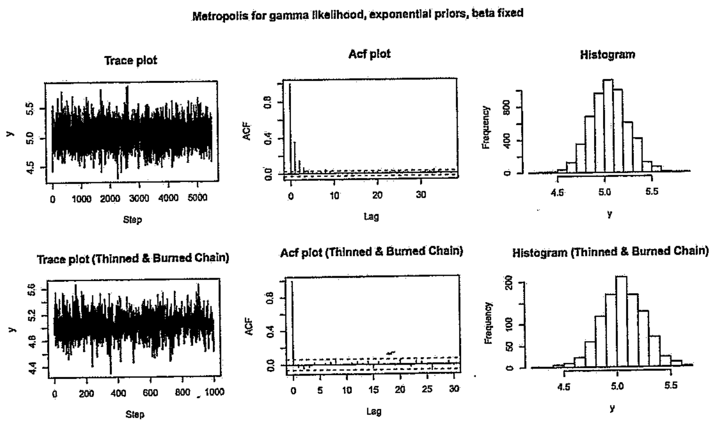

Figure 5: Diagnostics for the β only Metropolis sampler. The final chain burned 500 and used every 13th draw.

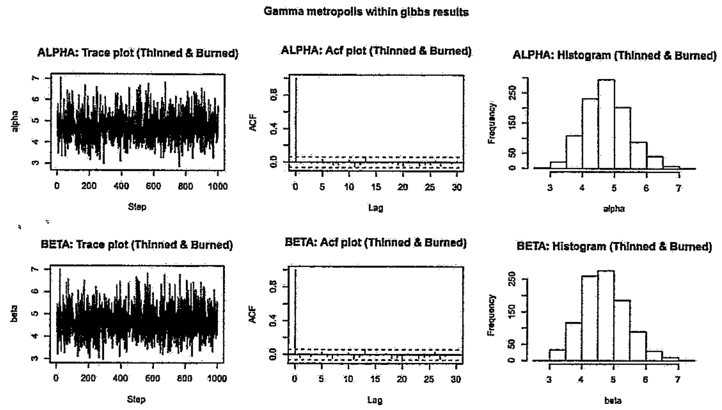

Figure 6: Diagnostics for the Metropolis-within-Gibbs sampler for (α, β). The final chain burned 500 and used every 180th (yikes!) draw. The bad autocorrelation suggest some serious issues with the proposal distribution.