Final Homework

STAT 950

Emily Robinson

December 18, 2019

Problem 1

Modify the ESS function to also estimate a 95% HPD interval. Include your function in the printed version of the homework.

```r
ESS <- function(chain, stop = 0.1, burnin = 0.5, alpha = 0.05) {
    if (!is.matrix(chain))
        chain = matrix(chain, ncol = 1)
    if (burnin) {
        if (burnin < 1) {
            burnin = burnin * dim(chain)[1]
        }
    }
    p = dim(chain)[2]
    results = matrix(NA, p, 7)
    colnames(results) = c("mean", "lowerHPD", "upperHPD",
        "se", "sd", "L", "ESS")
    rownames(results) = colnames(chain)
    for (i in 1:p) {
        h = chain[-(1:burnin), i]
        L = length(h)
        hbar = mean(h)
        hdev = h - hbar
        hvar = crossprod(hdev)/L
        tau = 1
        k = 1
        repeat {
            rho = crossprod(hdev[-(1:k)], hdev[-((L +
                1 - k):L)])/((L - k) * hvar)
            tau = tau + 2 * rho
            if (abs(rho) < stop || k > 1000)
                break
            k = k + 1
        }
        ESS = L/tau
        h_sort = sort(h)
        I = matrix(NA, nrow = ((L - 1) - floor((1 -
```

```
            alpha) * (L - 1))), ncol = 2)
        for (j in 1:((L - 1) - floor((1 - alpha) *
            (L - 1)))) {
            I[j, 1] = h_sort[j]
            I[j, 2] = h_sort[(j + floor((1 - alpha) *
                (L - 1)))]
        }
        jstar = which.min(I[, 2] - I[, 1])
        HPD = I[jstar, ]
        results[i, ] = c(mean = hbar, lowerHPD = HPD[1],
            upperHPD = HPD[2], se = sqrt(hvar/ESS),
            sd = sqrt(hvar), L = L, ESS = ESS)
    }
    return(results)
}
```

Problem 2

Consider the following model:

$$
\begin{aligned}
y_i|\kappa &\overset{\text{ind}}{\sim} \text{exponential}(\kappa_i) \\
\kappa_i &= \prod_j \theta_j^{x_{ij}} \\
\theta_j &\overset{\text{ind}}{\sim} \text{gamma}(\alpha_j, \lambda_j)
\end{aligned}
$$

where x_{ij} are known covariates, and α_j and λ_j are known hyperparameters. In some cases the y_i are censored at time t_i so the data are the pairs (t_i, w_i) where $w_i = 1$ if t_i is an uncensored time and $w_i = 0$ if t_i is a censored time yielding

$$f_{\boldsymbol{t},\boldsymbol{w}|\boldsymbol{\kappa}}(\boldsymbol{t}, \boldsymbol{w}|\boldsymbol{\kappa}) = \prod_i \kappa_i^{w_i} e^{-t_i k_i}.$$

Note: Consistent with the book we are using the parameterization where κ_i and λ_j are rate parameters as opposed to scale parameters.

Note: For this problem you may assume that $x_i j \in \{0, 1\}$. Which implies that

$$\kappa_i = \prod_{J_i} \theta_j$$

where $J_i = \{j : x_{ij} = 1\}$. Other useful sets are $I_j = \{i : x_{ij} = 1\}$ and $K_{ij} = \{k : x_{ik} = 1 \cap k \neq j\}$.

(a) Derive the score function and Hessian matrix necessary to compute the MLE estimates of $\boldsymbol{\theta}$ using the Newton-Raphson algorithm. Note: This will not involve the prior distribution, gamma(α_j, λ_j).

Consider

$$f_{\boldsymbol{t},\boldsymbol{\omega}|\boldsymbol{\kappa}}(\boldsymbol{t},\boldsymbol{\omega}|\boldsymbol{\kappa}) = \prod_i \kappa_i^{\omega_i} e^{-t_i k_i}$$

and assume $x_{ij} \in \{0,1\}$ with $\kappa_i = \prod_i \theta_j^{x_{ij}}$. Note the useful information:

$$\begin{aligned}
\kappa_i &= \prod_j \theta_j^{x_{ij}} \\
\frac{\partial \kappa_i}{\partial \theta_j} &= \frac{x_{ij}\kappa_i}{\theta_j} \\
\frac{\partial^2 \kappa_i}{\partial \theta_j^2} &= \frac{x_{ij}(x_{ij}-1)\kappa_i}{\theta_j^2} \\
\frac{\partial^2 \kappa_i}{\partial \theta_j \theta_m} &= \frac{x_{ij}x_{im}\kappa_i}{\theta_j\theta_m}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
&& \ell(\boldsymbol{\theta}) &= \sum_{i=1}^{n} [\omega_i \log(\kappa_i) - \kappa_i t_i] \\
\implies && \frac{\partial \ell(\boldsymbol{\theta})}{\partial \theta_j} &= \left[\frac{\partial \ell(\boldsymbol{\theta})}{\partial \kappa}\right] \cdot \left[\frac{\partial \kappa}{\partial \theta_j}\right] \\
&& &= \sum_{i=1}^{n} \left[\left(\frac{\omega_i}{k_i} - t_i\right) \frac{x_{ij}\kappa_i}{\theta_j}\right] \\
\implies && \frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \theta_j^2} &= \left[\frac{\partial \ell(\boldsymbol{\theta})}{\partial \kappa}\right] \cdot \left[\frac{\partial \kappa/\partial \theta_j}{\partial \theta_j}\right] + \left[\frac{\partial \kappa}{\partial \theta_j}\right] \cdot \left[\frac{\partial \ell(\boldsymbol{\theta})/\partial \kappa}{\partial \theta_j}\right] \\
&& &= \sum_{i=1}^{n} \left[\underbrace{\left(\frac{\omega_i}{\kappa_i} - t_i\right)\left(\frac{x_{ij}(x_{ij}-1)\kappa_i}{\theta_j^2}\right)}_{0} + \left(\frac{x_{ij}\kappa_i}{\theta_j}\right)\left(-\frac{\omega_i}{\kappa_i^2}\right)\left(\frac{x_{ij}\kappa_i}{\theta_j}\right)\right] \\
&& &= \sum_{i=1}^{n} \left[-\frac{\omega_i}{\kappa_i^2}\left(\frac{x_{ij}\kappa_i}{\theta_j}\right)^2\right] \\
&& &= \sum_{i=1}^{n} \left[-\omega_i\left(\frac{x_{ij}}{\theta_j}\right)^2\right] \\
\implies && \frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \theta_j \theta_m} &= \left[\frac{\partial \ell(\boldsymbol{\theta})}{\partial \kappa}\right] \cdot \left[\frac{\partial \kappa/\partial \theta_m}{\partial \theta_j}\right] + \left[\frac{\partial \kappa}{\partial \theta_j}\right] \cdot \left[\frac{\partial \ell(\boldsymbol{\theta})/\partial \kappa}{\partial \theta_m}\right] \\
&& &= \sum_{i=1}^{n} \left[\left(\frac{\omega_i}{\kappa_i} - t_i\right)\left(\frac{x_{ij}x_{im}\kappa_i}{\theta_j\theta_m}\right) + \left(\frac{x_{ij}\kappa_i}{\theta_j}\right)\left(-\frac{\omega_i}{\kappa_i^2}\right)\left(\frac{x_{im}\kappa_i}{\theta_m}\right)\right] \\
&& &= \sum_{i=1}^{n} \left[\left(\frac{\omega_i}{\kappa_i} - t_i\right)\left(\frac{x_{ij}x_{im}\kappa_i}{\theta_j\theta_m}\right) - \omega_i\left(\frac{x_{ij}x_{im}}{\theta_j\theta_m}\right)\right].
\end{aligned}$$

(b) Write a function to compute MLE estimates of $\boldsymbol{\theta}$ along with their approximate standard errors given $\boldsymbol{t}, \boldsymbol{w}$, and $\boldsymbol{x}$. Include your function in the printed version of the homework.

```r
# Create Objective Function
logLikeCancer <- function(theta, der = 0, X, t, w) {
    p = dim(X)[2]
    n = dim(X)[1]

    kappaIndex <- matrix(NA, nrow = n, ncol = p)
    for (i in 1:n) {
        for (j in 1:p) {
            kappaIndex[i, j] <- theta[j]^X[i, j]
        }
    }
    kappa <- apply(kappaIndex, 1, prod)
    value <- sum(w * log(kappa) - kappa * t)
    if (der == 0)
        return(value)

    der1 <- matrix(NA, nrow = p, ncol = 1)
    for (j in 1:p) {
        der1[j] <- sum((w/kappa - t) * X[, j] * kappa/theta[j])
    }
    if (der == 1)
        return(list(value = value, der1 = der1))

    der2 <- matrix(NA, nrow = p, ncol = p)
    for (j in 1:p) {
        for (m in 1:p) {
            if (j == m) {
                der2[j, m] = sum(-w * (X[, j]/theta[j])^2)
            } else {
                der2[j, m] = der2[m, j] = sum((w/kappa -
                  t) * (X[, j] * X[, m] * kappa/(theta[j] *
                  theta[m])) + -w * (X[, j] * X[, m])/(theta[j] *
                  theta[m]))
            }
        }
    }
    return(list(value = value, der1 = der1, der2 = der2))

}

# Newton Function
newtonR <- function(f, xInit, maxIt = 20, relConvCrit = 1e-10,
    ...) {
    p = length(xInit)
```

```r
    results = matrix(NA, maxIt, p + 2)
    colnames(results) = c("value", paste("x", 1:p,
        sep = ""), "Conv")

    xCurrent = xInit
    for (t in 1:maxIt) {
        evalF = f(xCurrent, der = 2, ...)
        results[t, "value"] = evalF$value
        results[t, 1 + (1:p)] = xCurrent
        xNext = xCurrent - solve(evalF$der2, evalF$der1)
        Conv = sqrt(crossprod(xNext - xCurrent))/(sqrt(crossprod(xCurrent)) +
            relConvCrit)
        results[t, "Conv"] = Conv
        if (Conv < relConvCrit)
            break
        xCurrent = xNext
    }

    evalFinal <- f(xNext, der = 2, ...)

    return(list(x = xNext, se = sqrt(diag(-solve(evalFinal$der2))),
        value = evalFinal$value, convergence = (Conv <
            relConvCrit), results = results[1:t, ]))
}
```

(c) Derive the conditional distributions necessary to implement the Gibbs sampler for $\boldsymbol{\theta}$.

The full joint distribution is

$$
\begin{aligned}
p(\boldsymbol{\theta}|\boldsymbol{t},\boldsymbol{\omega},X,\boldsymbol{\alpha},\boldsymbol{\lambda}) &\propto p(\boldsymbol{t}|\boldsymbol{\theta})\prod_j p(\theta_j) \\
&= \prod_{i=1}^n \kappa_i^{\omega_i} e^{-\kappa_i t_i} \prod_{j=1}^p \frac{\lambda_j^{\alpha_j}}{\Gamma(\alpha_j)}\theta_j^{\alpha_j-1}e^{-\lambda_j\theta_j} \\
&= \prod_{j=1}^p \theta_j^{\sum_{i=1}^n x_{ij}\omega_i} e^{-\sum_{i=1}^n [t_i \prod_{j=1}^p \theta_j^{x_ij}]} \prod_{j=1}^p \left[\frac{\lambda_j^{\alpha_j}}{\Gamma(\alpha_j)}\theta_j^{\alpha_j-1}e^{-\lambda_j\theta_j}\right] \\
&= \prod_{j=1}^p \left[\theta_j^{\sum_{i=1}^n x_{ij}\omega_i \frac{\lambda_j^{\alpha_j}}{\Gamma(\alpha_j)}\theta_j^{\alpha_j-1}} e^{-\lambda_j\theta_j}\right] e^{-\sum_{i=1}^n\left[t_j \prod_{j=1}^p \theta_j^{x_{ij}}\right]} \\
&\propto \prod_{j=1}^p \left[\theta_j^{\sum_{i=1}^n [x_{ij}\omega_i]+\alpha_j-1} e^{-\lambda_j\theta_j}\right] e^{-\sum_{i=1}^n\left[t_j \prod_{j=1}^p \theta_j^{x_{ij}}\right]}.
\end{aligned}
$$

Let $I_j = \{i : x_{ij} = 1\}$ and $m \neq j$. Note: $x_ij = 0 \cup 1$. Therefore, we can pull $\theta_j^{x_{ij}}$ out and use only observations where $x_{ij} = 1$. Therefore,

$$p(\theta_j|\boldsymbol{\theta}_{-j}, \boldsymbol{t}, \boldsymbol{\omega}, X, \boldsymbol{\alpha}, \boldsymbol{\lambda}) \propto \underbrace{\theta_j^{\left(\sum_{I_j} \omega_w x_{ij} + \alpha_j\right) - 1} e^{-\left(\lambda_j + \sum_{I_j}\left[t_i \prod \theta_m^{x_i m}\right]\right)\theta_j}}_{\text{looks like Gamma}\left(\sum_{I_j} \omega_w x_{ij} + \alpha_j, \lambda_j + \sum_{I_j}\left[t_i \prod \theta_m^{x_i m}\right]\right)} .$$

(d) Write a function that to implement your Gibbs sampler given $\boldsymbol{t}, \boldsymbol{w}, \boldsymbol{x}, \boldsymbol{\alpha}$, and $\boldsymbol{\lambda}$. Include your function in the printed version of the homework.

```
breastCancerGibbs <- function(X, t, w, alpha, lambda,
    nSamples = 10^4) {
    p = dim(X)[2]
    n = dim(X)[1]

    # Initial parameters
    theta = rep(NA, p)
    for (j in 1:p) {
        theta[j] = rgamma(1, alpha[j], lambda[j])
    }
    chain = matrix(NA, nSamples + 1, p)
    rownames(chain) = 0:nSamples
    colnames(chain) = c(paste("theta", 1:p, sep = "_"))
    chain[1, ] = theta
    for (s in 1:nSamples) {

        kappaIndex <- matrix(NA, nrow = n, ncol = p)
        for (i in 1:n) {
            for (j in 1:p) {
                kappaIndex[i, j] <- theta[j]^X[i, j]
            }
        }
        kappa <- apply(kappaIndex, 1, prod)

        for (j in 1:p) {
            index = which(X[, j] == 1)
            theta[j] = rgamma(1, sum(w[index] * X[index,
                j]) + alpha[j], lambda[j] + sum(t[index] *
                kappa[index]/kappaIndex[index, j]))
        }
        chain[s + 1, ] = theta
    }
    return(chain)
}
```

(e) Using the data and model described in problem 7.4.

i. Run your MLE function to obtain maximum likelihood estimates and approximate standard errors of $\boldsymbol{\theta}$.

```
breastCancer <- read.table("breastcancer.dat", header = T)
ans = newtonR(logLikeCancer, c(0.01, 1), X = model.matrix(~breastCancer$treatment),
    t = breastCancer$recurtime, w = !breastCancer$censored,
    relConvCrit = 1e-16)
results <- cbind(ans$x, ans$se)
colnames(results) = c("Estimate", "StdErr")
rownames(results) = c("theta", "tau")
kable(round(results, 3))
```

	Estimate	StdErr
theta	0.008	0.003
tau	1.212	0.495

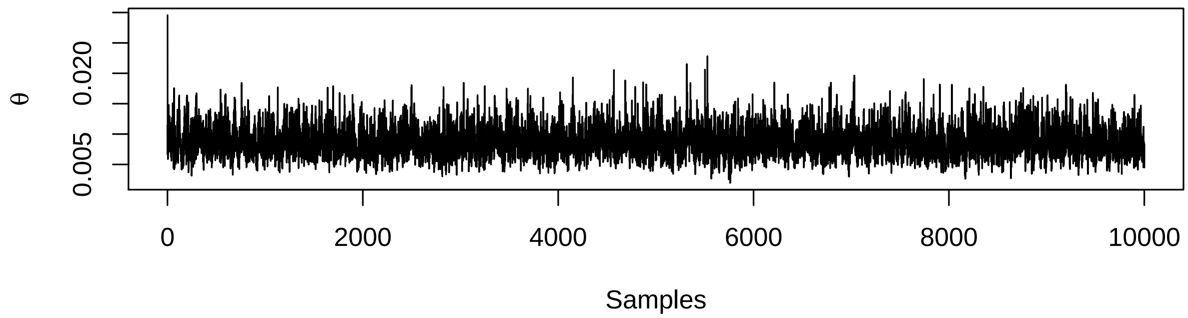
ii. Run and evaluate the performance of your Gibbs sampler using a single chain.

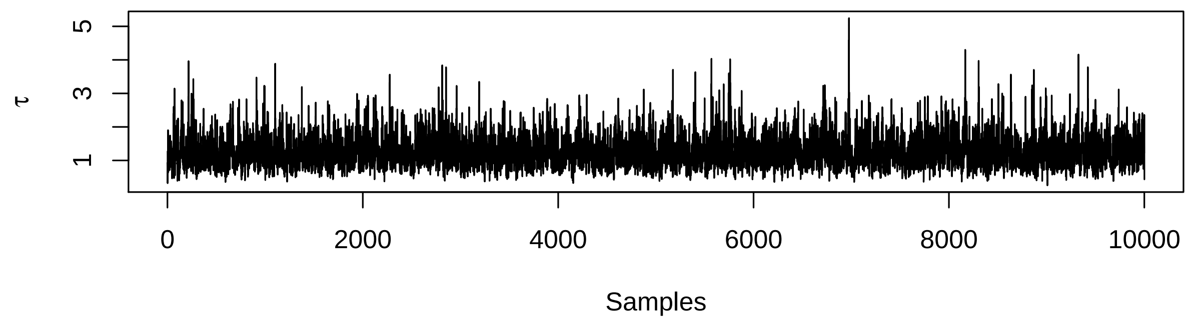
One chain ran in ≈ 3.75 seconds. The performance of the chain appears to be stable since the plots of the 10001 samples are "fuzzy catepillars" indicating that the estimates are being sampled around the same values. In addition the effective sample size is around 5001 (length of the end of our chain).

```
startTime = Sys.time()
breastCancerChain = breastCancerGibbs(X = model.matrix(~breastCancer$treatment),
    t = breastCancer$recurtime, w = !breastCancer$censored,
    alpha = c(2, 2), lambda = c(60, 1), nSamples = 10^4)
endTime = Sys.time()
endTime - startTime
```

```
## Time difference of 7.886317 secs
```

```
par(mfrow = c(2, 1))
plot(breastCancerChain[, 1], xlab = "Samples", ylab = expression(theta),
    type = "l")
plot(breastCancerChain[, 2], xlab = "Samples", ylab = expression(tau),
    type = "l")
```






```
par(mfrow = c(1, 1))
kable(round(ESS(breastCancerChain), 3))
```

	mean	lowerHPD	upperHPD	se	sd	L	ESS
theta_1	0.009	0.004	0.014	0.000	0.003	5001	4552.925
theta_2	1.281	0.442	2.187	0.007	0.487	5001	4513.736

iii. Using the `doParallel` and `foreach` packages run multiple chains of your Gibbs sampler and evaluate the performance of your Gibbs sampler.

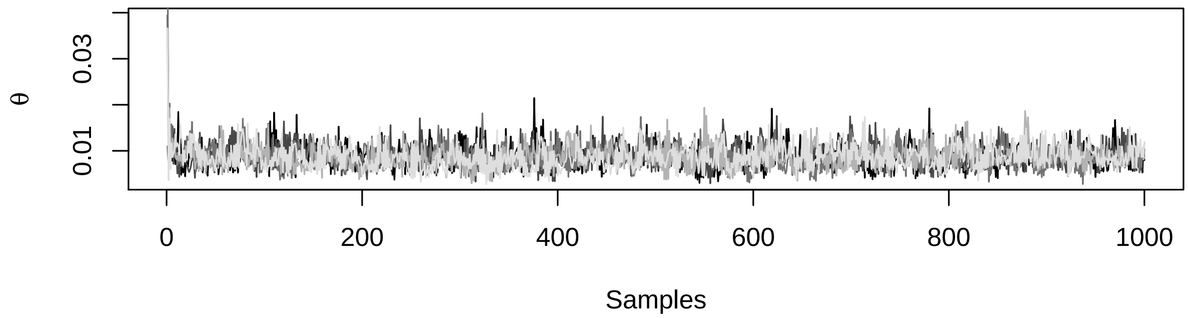
```
Five chains ran in $\approx 12.5$ seconds (~ 3 times as long as one chain). The performa
```

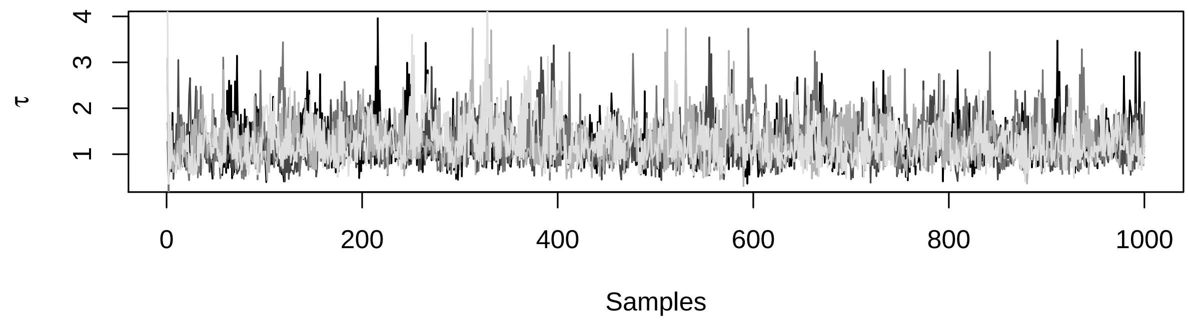
```
# Parallel Computing
require(parallel)
require(doParallel)

nCores = detectCores()
nChains = 5
ncl = min(nChains, nCores - 1)
```

```r
registerDoParallel(ncl)
startTime = Sys.time()
chains = foreach(c = 1:nChains) %dopar% {
    breastCancerChain = breastCancerGibbs(X = model.matrix(~breastCancer$treatment),
        t = breastCancer$recurtime, w = !breastCancer$censored,
        alpha = c(2, 2), lambda = c(60, 1), nSamples = 10^4)
}
endTime = Sys.time()
endTime - startTime
```

```
## Time difference of 18.58399 secs
```

```r
stopImplicitCluster()

par(mfrow = c(2, 1))
plot(chains[[1]][1:1000, 1], xlab = "Samples", ylab = expression(theta),
    col = "black", type = "l")
lines(chains[[2]][1:1000, 1], col = "gray28")
lines(chains[[3]][1:1000, 1], col = "gray45")
lines(chains[[4]][1:1000, 1], col = "gray70")
lines(chains[[5]][1:1000, 1], col = "gray87")

plot(breastCancerChain[1:1000, 2], xlab = "Samples",
    ylab = expression(tau), col = "black", type = "l")
lines(chains[[2]][1:1000, 2], col = "gray28")
lines(chains[[3]][1:1000, 2], col = "gray45")
lines(chains[[4]][1:1000, 2], col = "gray70")
lines(chains[[5]][1:1000, 2], col = "gray87")
```






```
par(mfrow = c(1, 1))
```

iv. Compute summary statistics of the estimated joint posterior distribution of the $\boldsymbol{\theta}$ along with the mean remission times for control and treated patients, including marginal means, standard deviations, and 95% probability intervals.

The estimated remission time for a patient in the treatment group is about 2 weeks earlier than the remission time for a patient in the control group.

```
breastCancerChain2 <- cbind(breastCancerChain, remissionC = 1/breastCancerChain[,
    1], remissionT = 1/(breastCancerChain[, 1] * breastCancerChain[,
    2]))
kable(round(ESS(breastCancerChain2), 3))
```

	mean	lowerHPD	upperHPD	se	sd	L	ESS
theta_1	0.009	0.004	0.014	0.000	0.003	5001	4552.925
theta_2	1.281	0.442	2.187	0.007	0.487	5001	4513.736
remissionC	125.112	62.444	201.127	0.580	39.335	5001	4593.959
remissionT	111.142	28.806	216.555	0.548	54.822	5001	10009.917

Problem 3

Using the data and change point model for problem 7.6. For the prior assume $\lambda_i \sim \text{Gamma}(\gamma_1, \alpha)$ for $i = 1, 2$ and $\alpha \sim \text{Gamma}(\gamma_2, \gamma_3)$ where $\gamma_1, \gamma_2, \gamma_3$ are known hyperparameters.

(a) Derive the conditional distributions necessary to implement a change point model Gibbs sampler.

Consider the full joint distribution,

$$
\begin{aligned}
p(\lambda_1, \lambda_2, \alpha, \theta | \boldsymbol{x}) &\propto p(\boldsymbol{x}_{1:\theta}) p(\boldsymbol{x}_{\theta+1:N}) p(\lambda_1|\alpha) p(\lambda_2|\alpha) p(\alpha) p(\theta) \\
&= prod_{i=1}^{\theta} p(x_i|\lambda_1) \prod_{i=\theta+1}^{N} p(x_i|\lambda_2) p(\lambda_1|\alpha) p(\lambda_2|\alpha) p(\alpha) p(\theta) \\
\implies \qquad \log p(\lambda_1, \lambda_2, \alpha, \theta | \boldsymbol{x}) &= \sum_{i=1}^{\theta} [\log p(x_i|\lambda_1)] \\
&+ \sum_{i=\theta+1}^{N} [\log p(x_i|\lambda_2)] \\
&+ \log p(\lambda_1|\alpha) \\
&+ p(\lambda_2|\alpha) \\
&+ \log p(\alpha) \\
&+ \log p(\theta) \\
&= \sum_{i=1}^{\theta} [x_i \log \lambda_1 - \lambda_1 - \log x_i!] \\
&+ \sum_{\theta+1}^{N} [x_i \log \lambda_2 - \log x_i!] \\
&+ \gamma_1 \log \alpha - \log \Gamma(\gamma_1) \\
&+ (\gamma_1 - 1) \log \lambda_1 - \alpha \lambda_1 \\
&+ \gamma_1 \log \alpha - \log \Gamma(\gamma_1) \\
&+ (\gamma_1 - 1) \log \lambda_2 - \alpha \lambda_2 \\
&+ \gamma_2 \log \gamma_3 - \log \Gamma(\gamma_2) \\
&+ (\gamma_2 - 1) \log \alpha - \gamma_3 \alpha \\
&- \log(N - 1).
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
p(\lambda_1|\lambda_2,\alpha,\theta,\boldsymbol{x},\gamma_1,\gamma_2,\gamma_3) &= \sum_{i=1}^{\theta}[x_i\log\lambda_1-\lambda_1]+(\gamma_1-1)\log\lambda_1-\alpha\lambda_1\\
&= \underbrace{\left(\gamma_1+\sum_{i=1}^{\theta}[x_i]-1\right)\log\lambda_1-(\theta+\alpha)\lambda_1}_{\text{look like logGamma}(\gamma_1+\sum_{i=1}^{\theta}[x_i],\theta+\alpha)}\\
p(\lambda_2|\lambda_1,\alpha,\theta,\boldsymbol{x},\gamma_1,\gamma_2,\gamma_3) &= \sum_{i=\theta+1}^{N}[x_i\log\lambda_2-\lambda_2]+(\gamma_1-1)\log\lambda_2-\alpha\lambda_2\\
&= \underbrace{\left(\gamma_2+\sum_{i=\theta+1}^{N}[x_i]-1\right)\log\lambda_2-(N-\theta+\alpha)\lambda_2}_{\text{look like logGamma}(\gamma_1+\sum_{i=\theta+1}^{N}[x_i],N-\theta+\alpha)}\\
p(\alpha|\lambda_1,\lambda_2,\theta,\boldsymbol{x},\gamma_1,\gamma_2,\gamma_3) &= \gamma_1\log\alpha-\alpha\lambda_1+\gamma_1\log\alpha-\alpha\lambda_2+(\gamma_2-1)\log\alpha-\gamma_3\alpha\\
&= \underbrace{(2\gamma_1+\gamma_2-1)\log\alpha-(\lambda_1+\lambda_2+\gamma_3)\alpha}_{\text{looks like logGamma}(2\gamma_1+\gamma_2,\lambda_1+\lambda_2+\gamma_3)}\\
p(\theta|\lambda_1,\lambda_2,\alpha,\boldsymbol{x},\gamma_1,\gamma_2,\gamma_3) &= \sum_{i=1}^{\theta}[x_i\log\lambda_1-\lambda_1-\log x_i!]+\sum_{i=\theta+1}^{N}[x_i\log\lambda_2-\lambda_2-\log x_i!].
\end{aligned}
$$

Therefore, we sample from the following distributions,

$$
\begin{aligned}
\lambda_1 &\sim \text{Gamma}(\gamma_1+\sum_{i=1}^{\theta}[x_i],\theta+\alpha)\\
\lambda_2 &\sim \text{Gamma}(\gamma_1+\sum_{i=\theta+1}^{N}[x_i],N-\theta+\alpha)\\
\alpha &\sim \text{Gamma}(2\gamma_1+\gamma_2,\lambda_1+\lambda_2+\gamma_3)
\end{aligned}
$$

and use the proportional conditional distribuiton of θ to sample from a multinomial since $\theta \in \{1, ..., 111\}$.

(b) Write a function that to implement a change point model Gibbs sampler given $\boldsymbol{X}$ and the three gamma distribution hyperparameters. Include your function in the printed version of the homework.

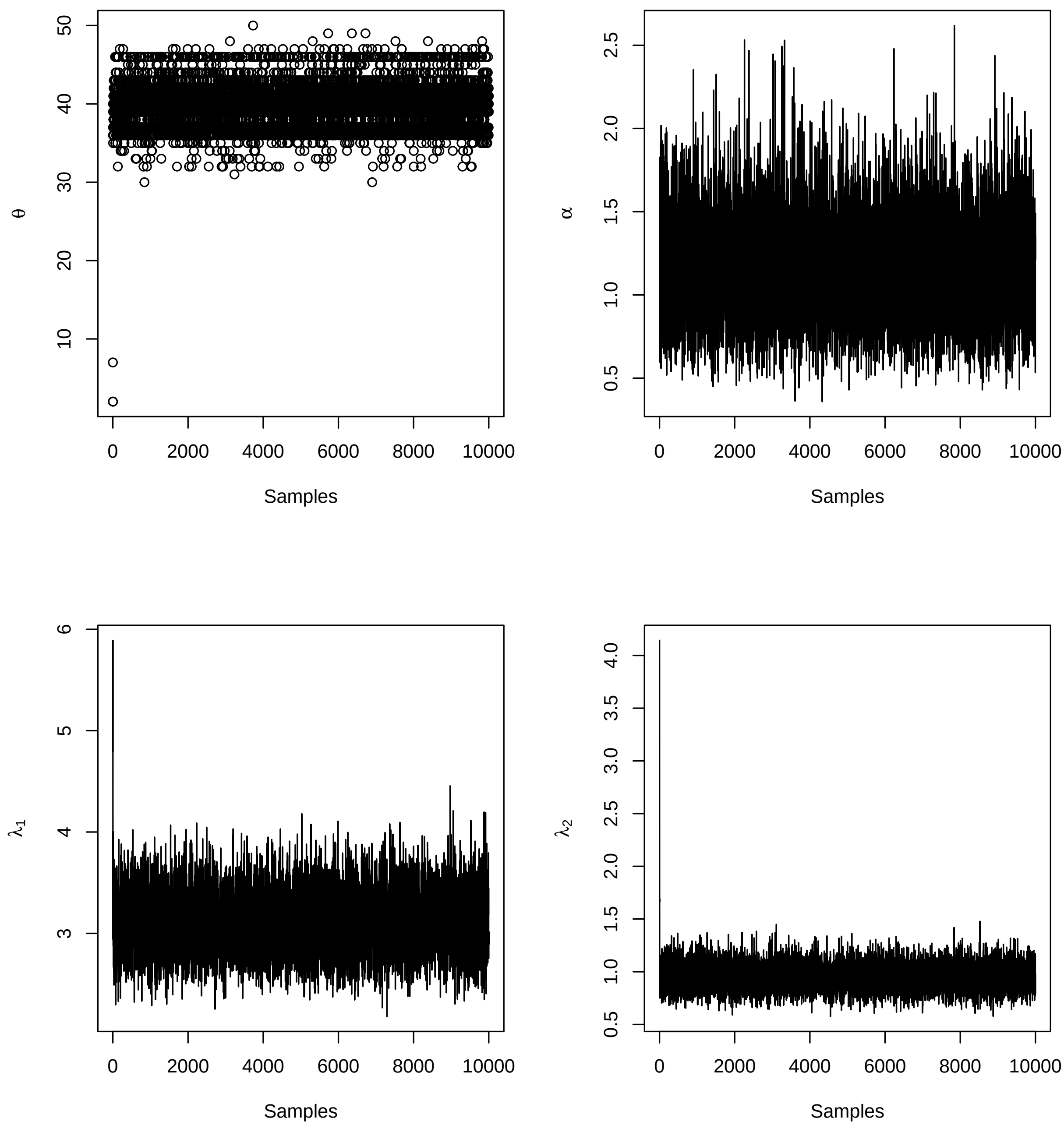
```
changePointGibbs <- function(x, gamma, nSamples = 10^4) {
    x = as.matrix(x)
    N = dim(x)[1]
    # Initial parameters
    theta = sample(1:N - 1, 1)
    alpha = rgamma(1, gamma[2], gamma[3])
    lambda1 = rgamma(1, gamma[1], alpha)
    lambda2 = rgamma(1, gamma[1], alpha)
```

```
    chain = matrix(NA, nSamples + 1, 4)
    rownames(chain) = 0:nSamples

    colnames(chain) = c("theta", "alpha", "lambda_1",
        "lambda_2")
    chain[1, ] = c(theta, alpha, lambda1, lambda2)

    for (s in 1:nSamples) {
        ptheta <- rep(NA, (N - 1))
        for (n in 1:(N - 1)) {
            ptheta[n] = sum(x[1:n]) * log(lambda1) -
                n * lambda1 + sum(x[(n + 1):N]) * log(lambda2) -
                (N - n) * lambda2
        }
        prob = exp(ptheta - max(ptheta))/sum(exp(ptheta -
            max(ptheta)))

        theta = which.max(rmultinom(1, 1, prob))
        alpha = rgamma(1, 2 * gamma[1] + gamma[2],
            lambda1 + lambda2 + gamma[3])
        lambda1 = rgamma(1, gamma[1] + sum(x[1:theta]),
            theta + alpha)
        lambda2 = rgamma(1, gamma[1] + sum(x[(theta +
            1):N]), N - theta + alpha)
        chain[s + 1, ] = c(theta, alpha, lambda1, lambda2)
    }
    return(chain)
}
```

(c) Run your Gibbs sample using the coal mine data and compute summary statistics of the estimated joint posterior distribution of the θ, λ_1, and λ_2, including marginal means, standard deviations, and 95% probability intervals.

```
coal <- read.table("coal.dat", header = T)
coalChain = changePointGibbs(coal$disasters, gamma = c(3,
    10, 10), nSamples = 10^4)

par(mfrow = c(2, 2))
plot(coalChain[, 1], xlab = "Samples", ylab = expression(theta))
plot(coalChain[, 2], xlab = "Samples", ylab = expression(alpha),
    type = "l")
plot(coalChain[, 3], xlab = "Samples", ylab = expression(lambda[1]),
    type = "l")
plot(coalChain[, 4], xlab = "Samples", ylab = expression(lambda[2]),
```

```r
    type = "l")
```




```r
par(mfrow = c(1, 1))
kable(round(ESS(coalChain), 3))
```

	mean	lowerHPD	upperHPD	se	sd	L	ESS
theta	39.830	34.000	44.000	0.041	2.470	5001	3643.922

	mean	lowerHPD	upperHPD	se	sd	L	ESS
alpha	1.134	0.600	1.706	0.004	0.286	5001	4762.338
lambda_1	3.108	2.525	3.662	0.004	0.294	5001	4448.778
lambda_2	0.950	0.726	1.188	0.002	0.118	5001	4267.788