MATH48122 - Coursework 3

Student ID: 10876957

Suppose we have 2 proposed models for the lifetime distribution X of 20 bulbs. We will denote Model 1 as $X \sim Gamma(\alpha, \beta)$ and Model 2 as $X \sim Weibull(\gamma, \delta)$.

Question 1

To begin with, we will find the Jacobian of the proposed move from Model 2 to Model 1. It can be found by Equation 1.

$$\text{J=det}\begin{pmatrix} \frac{d\alpha}{d\gamma} & \frac{d\alpha}{d\delta} \\ \frac{d\beta}{d\gamma} & \frac{d\beta}{d\delta} \end{pmatrix} \tag{1}$$

Using that $\alpha = \Gamma(1+\frac{1}{\gamma})$ and $\beta = \frac{1}{\delta}$, we can find all the 4 elements in the matrix:

- $\frac{d\alpha}{d\gamma} = -\frac{1}{\gamma^2}\Gamma'(1+\frac{1}{\gamma}) = -\frac{1}{\gamma^2}\psi(1+\frac{1}{\gamma})\Gamma(1+\frac{1}{\gamma})$
- $\frac{d\alpha}{d\delta} = 0$
- $\frac{d\beta}{d\gamma} = 0$
- $\frac{d\beta}{d\delta} = -\frac{1}{\delta^2}$

Therefore, the Jacobian in terms of $\Gamma(\cdot)$ and $\psi(\cdot)$ is calculated in Equation 2:

$$J = \frac{d\alpha}{d\gamma}\cdot\frac{d\beta}{d\delta} - \frac{d\alpha}{d\delta}\cdot\frac{d\beta}{d\gamma} = \frac{1}{\gamma^2\delta^2}\Gamma(1+\frac{1}{\gamma})\psi(1+\frac{1}{\gamma}) \tag{2}$$

Question 2

The reversible jump algorithm to move between Models 1 and 2 can be done by the code shown in Listing 1:

```r
library(LambertW)

# This function computes the inverse of the gamma function
invgamma <- function(x, k) {
  c <- sqrt(2*pi)/exp(1) - gamma(k)
  L <- log((x+c)/sqrt(2*pi))
  W1 <- W(L/exp(1))
  igamma <- L/W1 + 1/2
  return(igamma)
```

```
}
#Define the function for the RJMCMC algorithm
rjlife <- function(x, model, gamma, delta, asig, bsig, gsig, dsig, k, run)
{
  n <- length(x)
  output <- matrix(0, ncol = 3, nrow = run)
 {
#Initialise alpha and beta based on initial values for gamma and delta
v <- 1 + 1/gamma
alpha <- gamma(v)
beta <- 1 / delta
}
  for(ii in 1:run)
{
    if (model == 1)
{
mod=1
{
#RWM algorithm for updating alpha and beta in Model 1

alphanew <- rnorm(1, alpha, asig)
betanew <- rnorm(1, beta, bsig)
if(alphanew<0) alphanew=abs(alphanew)
if(betanew<0) betanew=abs(betanew)

if (alphanew > 0.89 && betanew > 0) {
liknew <- prod(dgamma(x, alphanew, betanew)) #numerator of the likelihood ratio
lik <- prod(dgamma(x, alpha, beta)) #denominator of the likelihood ratio
ratio=liknew/lik
ratio=ratio*exp(alpha-alphanew) #proposals for alpha
ratio=ratio*exp(beta-betanew) #proposals for beta

u <- runif(1)
if (u < ratio)
{
alpha=alphanew
beta=betanew
} # If proposed value is accepted, set alpha and beta equal to the proposed values
}
}
    {
    # Proposal to move from Model 1 to Model 2
      gamma=1/(invgamma(alpha,k)-1)
      delta=1/beta
      v <- 1 + 1/gamma
      jacob = (1/(delta^2 * gamma^2)) * gamma(v) * digamma(v) # Jacobian
      likr = (prod(dweibull(x, gamma, delta)))/(prod(dgamma(x, alpha, beta))) #Likelihood ratio
      priors= (exp(-gamma-delta)) / (exp(-alpha-beta)) #Proposal ratio of Model 2 over Model 1
      ratio = likr*priors/(jacob)
}
u=runif(1)
if (u < ratio) mod=2 #If accepted move to Model 2
}
    if (model == 2)
{
```

```r
65  mod=2
66  {
67  #RWM algorithm for updating gamma and delta in Model 2
68  gammanew <- rnorm(1, gamma, gsig)
69  deltanew <- rnorm(1, delta, dsig)
70  if(gammanew<0) gammanew=abs(gammanew)
71  if(deltanew<0) deltanew=abs(deltanew)
72  if (gammanew > 1.46 && deltanew > 0) {
73  liknew2<- prod(dweibull(x, gammanew, deltanew)) #numerator of the likelihood ratio
74  lik2<- prod(dweibull(x, gamma, delta)) #denominator of the likelihood ratio
75  ratio=liknew2/lik2
76  ratio=ratio*exp(gamma-gammanew) #proposals for gamma
77  ratio=ratio*exp(delta-deltanew) #proposals for delta
78  u <- runif(1)
79  if (u < ratio)
80  {
81  gamma=gammanew
82  delta=deltanew
83  } # If proposed value is accepted, set gamma and delta equal to the proposed values
84  }
85  }
86  {
87  # Proposal to move from Model 2 to Model 1
88  v <- 1 + 1/gamma
89  alpha=gamma(v)
90  beta=1/delta
91  jacob = (1/(delta^2 * gamma^2)) * gamma(v) * digamma(v) #Jacobian
92  likr = (prod(dweibull(x, gamma, delta))) / (prod(dgamma(x, alpha, beta)))
93  priors= (exp(-gamma-delta)) / (exp(-alpha-beta))
94  ratio = (1/likr)*(1/priors)*(jacob) #Inverse of the likelihood and prior ratio from before
95        }
96  u=runif(1)
97  if(u < ratio) mod=1 #If accepted, move back to Model 1
98  }
99      model=mod
100     if (mod == 1) output[ii,] <- c(1, alpha, beta)
101     if (mod == 2) output[ii,] <- c(2, gamma, delta)
102   }
103   output
104 }
```

Listing 1: R code for RJMCMC algorithm

Code implementation: On lines 4 to 10 in Listing 1 we define a function for computing the inverse of the gamma function. On line 12 we define a function for performing the RJMCMC algorithm, containing the data, the model which we will start from, the parameters in Model 2, the proposal standard deviations for the parameters in both models, k from above, and the number of times we will run the algorithm for, respectively. On line 15 we define the output to be with 3 columns - the first one contains 1 or 2, depending on which model was chosen, and the other two are the parameter estimates from the according model. On lines 18 to 20 we set initial values for α and β based on γ and δ. First, we will consider moving from Model 1 to Model 2. Therefore, we should use a RWM algorithm for updating α and β in Model 1 /Lines 30 to 46/. Then, we calculate the acceptance ratio needed to move to Model 2. /Lines 52 to 61/. It is found by dividing the posterior distribution of Model 2 over the one of Model 1 and then we divide this ratio by the Jacobian we found in Question 1 /i.e. multiply it by its inverse/, since here we move from Model 1 to Model 2. We should note that here $q(1|2) = q(2|1) = 1$ and also there are no auxiliary variables because of the bijection. That is why this becomes our final acceptance ratio. From Line 63 to 97 we use a similar approach to calculate the acceptance ratio to move from Model 2 to Model 1. However, this time the ratio is calculated by taking the product of the inverses of the likelihood ratio and the priors ratio we used above /i.e. we divide the posterior distribution of Model 1 over the one of Model 2/. Since we already found the Jacobian for moving from Model 2 to Model 1 in Question 1, we simply multiply the ratio by it. Finally, on Lines 99 to 101 we set the algorithm to output the estimates of α and β if Model 1 is chosen and these of γ and δ if Model 2 is chosen.

Question 3

Let us now apply the RJMCMC to the given data. Then we will estimate the model parameters and the posterior probability of Model 1 and 2. This is done by the code shown in Figure 1:

```r
x <- c(0.8, 8.8, 1.4, 3.8, 6.7, 0.8, 0.3, 2.1, 3.7, 1.2, 2.7, 2.7, 4.5, 1.5, 10.1, 1.4, 10.3, 1.2, 8.7, 5.3) #store the data in vector x
result <- rjlife(x, 1,1,2, 0.5, 0.2, 1, 1, 1.46, 1000) #apply RJMCMC algorithm to the data x

mean(result[result[,1]==1,2]) #estimate for alpha from Model 1
mean(result[result[,1]==1,3]) #estimate for beta from Model 1

mean(result[result[,1]==2,2]) #estimate for gamma from Model 2
mean(result[result[,1]==2,3]) #estimate for delta from Model 2

var(result[result[,1]==1,2]) #posterior variance for alpha from Model 1
var(result[result[,1]==1,3]) #variance for beta from Model 1

var(result[result[,1]==2,2]) #posterior variance for gamma from Model 2
var(result[result[,1]==2,3]) #posterior variance for delta from Model 2

sum(result[,1]==1)/1000 #posterior probability of Model 1
sum(result[,1]==2)/1000 #posterior probability of Model 2
```

Figure 1: R code for applying RJMCMC to the data and estimating the model parameters and the posterior probability of Model 1 and 2

The code output we get is presented in Figure 2.

From the output, we can observe that the estimates for α and β are 1.385192 and 0.3505895, respectively /Lines 2 and 4/. The estimates for γ and δ are 1.159241 and 4.110237 /Lines 7 and 9/. Moreover, the posterior variances for α and β are 0.09437035 and 0.01196399, respectively /Lines 12 and 14/, whereas the ones for γ and δ are 0.04992123 and 0.7581651, respectively /Lines 17 and 19/. Since the variances are really low for α, β and γ, the estimates our algorithm made are quite accurate. The δ parameter has a bit higher variance compared to the other parameters, but since it is not excessively large, we can conclude that the estimate is still quite good. The estimated posterior probabilities for the two models are 0.823 and 0.177 for Models 1 and 2, respectively /Lines 22 and 24/. That is, we know with certainty that Model 1 is the superior model for this data set.

```
> mean(result[result[,1]==1,2]) #estimate for alpha from Model 1
[1] 1.385192
> mean(result[result[,1]==1,3]) #estimate for beta from Model 1
[1] 0.3505895
>
> mean(result[result[,1]==2,2]) #estimate for gamma from Model 2
[1] 1.159241
> mean(result[result[,1]==2,3]) #estimate for delta from Model 2
[1] 4.110237
>
> var(result[result[,1]==1,2]) #posterior variance for alpha from Model 1
[1] 0.09437035
> var(result[result[,1]==1,3]) #variance for beta from Model 1
[1] 0.01196399
>
> var(result[result[,1]==2,2]) #posterior variance for gamma from Model 2
[1] 0.04992123
> var(result[result[,1]==2,3]) #posterior variance for delta from Model 2
[1] 0.7581651
>
> sum(result[,1]==1)/1000 #posterior probability of Model 1
[1] 0.823
> sum(result[,1]==2)/1000 #posterior probability of Model 2
[1] 0.177
```

Figure 2: R code for applying RJMCMC to the data and estimating the model parameters and the posterior probability of Model 1 and 2

Question 4

Since we have chosen Model 1 in Question 3, we will now compute the batch mean estimates for the variances of α and β. This is done by the R code shown in Figure 3:

```
alphav=result[result[,1]==1,2] #vector of all estimates of alpha
betav=result[result[,1]==1,3] #vector of all estimates of beta

batch=function(x,b) #function to compute a batch mean estimate of the paramater variance
{
n=length(x)
m=floor(n/b)
y=0
for(i in 1:m) y[i]=mean(x[(((i-1)*b)+1):(i*b)])
sigma=sum((y-mean(x))**2)
sigma2=b*sigma/(m-1)
sigma2
}

batch(alphav,20) #apply the function to the output for alpha
batch(betav,20) #apply the function to the output for beta
```

Figure 3: R code for computing the batch mean estimates for the variances of α and β.

After running the code, we get the output shown in Figure 4:

```
1 > batch(alphav,20)
2 [1] 0.9030867
3 > batch(betav,20)
4 [1] 0.1045755
```

Figure 4: R code output for the batch mean estimates for the variances of α and β.

From Figure 4, we find that the batch mean estimate for the variance of α is 0.9030867 /Line 2/ and the one of β is 0.1045755 /Line 4/.

Let us do the acf plots for the both parameters, presented in Figures 5 and 6:

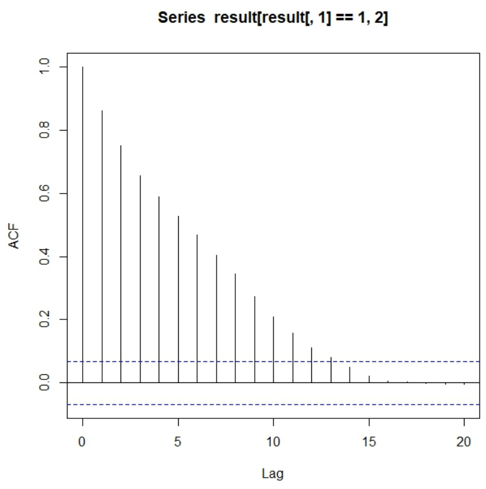

Figure 5: Acf plot for α

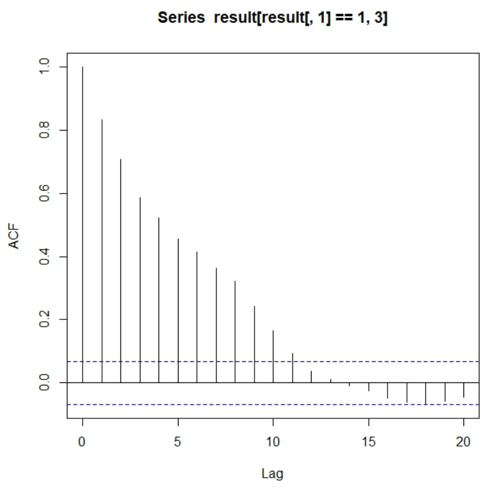

Figure 6: Acf for β

Therefore, after observing the graphs, we may consider a smaller batch size for more accuracy. If we choose the size of 15, we get more accurate results, as shown in Figure 7:

```
1 > batch(alphav,15) #apply the function to the output for alpha
2 [1] 0.8617692
3 > batch(betav,15) #apply the function to the output for beta
4 [1] 0.09644118
```

Figure 7: R code output for the batch mean estimates for the variances of α and β.

In Figure 7 we see that this time the batch mean estimate for the variance of α is 0.8617692 /Line 2/ and the one of β is 0.09644118 /Line 4/. This suggests that in our case the variability in the estimates becomes lower when we have a smaller size of the batches and therefore they become more precise.

The batch size affects the bias and variance in batch mean estimation. Smaller batch sizes reduce bias, but increase variance, as we can see when comparing with the posterior variances we found on lines 12 and 14 in Figure 2. However, it is important to find the balance between precision and computational efficiency - we cannot simply use a very large sample size. In this example, after looking at Figures 5 and 6, we conclude that 15 may be our compromise since the batch mean estimates for the variances of α and β seem to be reasonable /especially the one for β, since it is approximately 0.1/ and they are not excessively large.