Assignment 2 Report

Computational Methods for Bayesian Statistics

Trilokinath Modi

October 2, 2020

1 Question Set 1

1.1 Q1 - a

The probability that the turbine delivers some power is

$$\begin{aligned}
\text{Prob}(P(\nu) \geq 0) &= P(13 \leq \nu \leq 30) \\
&= \int_{13}^{30} \frac{\beta^{\alpha}}{\Gamma(\alpha)} \nu^{\alpha-1} \exp(-\beta\nu) \\
&= \int_{13}^{30} \frac{0.1^{0.5}}{\Gamma(0.5)} \nu^{0.5} \exp(-0.1\nu) \\
&= \int_{0}^{30} \frac{0.1^{0.5}}{\Gamma(0.5)} \nu^{0.5} \exp(-0.1\nu) - \int_{0}^{13} \frac{0.1^{0.5}}{\Gamma(0.5)} \nu^{0.5} \exp(-0.1\nu) \\
&= 0.888 - 0.542 \\
&= 0.346
\end{aligned}$$

The R-code used for integrals is

```
alphaParameter = 1.5
betaParameter = 0.1
quantiles = c(13, 30)
probQ1a = pgamma(quantiles[2], alphaParameter, betaParameter) - pgamma(quantiles
    ↪ [1], alphaParameter, betaParameter)
probQ1a
```

1.2 Q1 - b

n samples were drawn from velocity density function i.e. from the gamma distribution using the given hyper-parameters. Windmill power was computed using the vector of velocities obtained. The experiment is repeated for different values of n. The set for number of samples is,

$$n = \{10, 100, 1000, 10000, 100000\}$$

The expected windmill power, respectively with the sample size is,

$$\bar{\nu} = \{0.2365, 0.4038, 0.3981, 0.4018, 0.3979\}$$

Table 1: *Expected wind power and Confidence Interval for different sample size*

n	$\bar{\nu}$	$I_{\bar{\nu}}$
10	0.2365	[0.0000 0.2950]
100	0.4038	[0.2675 0.4475]
1000	0.3981	[0.3518 0.4132]
10000	0.4018	[0.3877 0.4069]
100000	0.3979	[0.3933 0.3995]

We can clearly see that the expected windmill power gets relatively stable as sample size is increased. For a large sample i.e. of size $1e5$, the expected windmill power is 0.3979.

The confidence interval is then estimated by bootstrap technique and 1000 bootstrap samples were generated each for the sample size in set n. The confidence interval, expected wind power and sample size is tabulated in table 1. Since there is randomness involved, it is observed that the CI for small sample size is fairly inconsistent where as the CI in case of large sample size is fairly consistent. Considering large sample size of $1e5$, the obtained CI is [0.39330.3995].

The Rcode used here is,

```
frequencyVelocity <- c(10,100,1000,10000,100000)
windVelocities = vector(mode = "list", length = length(frequencyVelocity))
windPower = vector(mode = "list", length = length(frequencyVelocity))
expectedWindPower <- rep(0, length(frequencyVelocity))

windPowerFunction <- function(velocity){
  if(velocity >= 13 && velocity < 23){
    wP <- cos(velocity*pi/10 - 3*pi/10) + 1
    return(wP)
  }
  else if(velocity >= 23 && velocity < 30){
    wP <- (1031 + 46*velocity - velocity^2)/780
    return(wP)
  }
  else{
    return(0)
  }
}
set.seed(777)
for (iFreq in 1:length(frequencyVelocity)){
  windVelocities[[iFreq]] <- rgamma(frequencyVelocity[iFreq], alphaParameter,
     ↪ betaParameter)
  windPower[[iFreq]] <- rep(0,times = frequencyVelocity[[iFreq]])
  windPower[[iFreq]] <- sapply(windVelocities[[iFreq]], windPowerFunction)
  expectedWindPower[iFreq] <- mean(windPower[[iFreq]])
}
expectedWindPower
#print expected wind power
```

```
# Bootstrap for CI
nBootSamples <- 1000
bootStrapSamples <- vector(mode = "list", length = length(frequencyVelocity))
bootStrapSampleMean <- vector(mode = "list", length = length(frequencyVelocity))
diffMean <- vector(mode = "list", length = length(frequencyVelocity))
quantiles <- vector(mode = "list", length = length(frequencyVelocity))
confidenceInterval <- vector(mode = "list", length = length(frequencyVelocity))
set.seed(777)
for (iFreq in 1:length(frequencyVelocity)) {
  sampleValues <- sample(windPower[[iFreq]], nBootSamples*frequencyVelocity[iFreq
    ↪ ], replace = TRUE)
  bootStrapSamples[[iFreq]] <- matrix(sampleValues, nrow = frequencyVelocity[
    ↪ iFreq], ncol = nBootSamples)
  bootStrapSampleMean[[iFreq]] <- colMeans(bootStrapSamples[[iFreq]])
  diffMean[[iFreq]] <- expectedWindPower[iFreq] - bootStrapSampleMean[[iFreq]]
  quantiles[[iFreq]] <- quantile(diffMean[[iFreq]],c(0.25,0.975))
  confidenceInterval[[iFreq]] <- expectedWindPower[iFreq] - c(quantiles[[iFreq
    ↪ ]][2], quantiles[[iFreq]][1])
}
```

1.3 Q1 - c

To compute expectation using discretization, a velocity sequence is assumed. Since we know the exact distribution for velocity, we can consider 5 standard deviations for velocity.

$$\begin{aligned} \sigma &= \frac{\alpha}{\beta^2} \\ &= \sqrt{150} = 12.25 \\ \implies 5\sigma &= 12.25 * 5 = 61.25 \approx 60 \end{aligned}$$

We can consider the gamma density as prior and hence the prior density is portrayed in figure 1 Similarly, the likelihood is shown in figure 2 The posterior upto a constant can be achieved by Multiplying the posterior and prior and the expected value can be computed by area under the curve. Since we have discreticized, larger sequence will lead to more accurate results. The posterior density upto a constant is shown in figure 3. The expectation is estimated by area under the curve which is approximated as,

$$\begin{aligned} \bar{\nu} &= \text{Area under the discrete curve} \\ &\approx \left(\sum \text{posterior density}\right) \times \frac{\text{Length of sequence}}{\text{Number of points}} \\ &\approx \mathbf{0.3993} \end{aligned}$$

R code used for above computations is,

```
velocitySequence <- seq(0,60,length.out = 1000)
densityVelocity <- dgamma(velocitySequence,alphaParameter, betaParameter)
plot(velocitySequence,densityVelocity, ylab = "Density", xlab = "velocity", main
    ↪ = "Prior density")
```

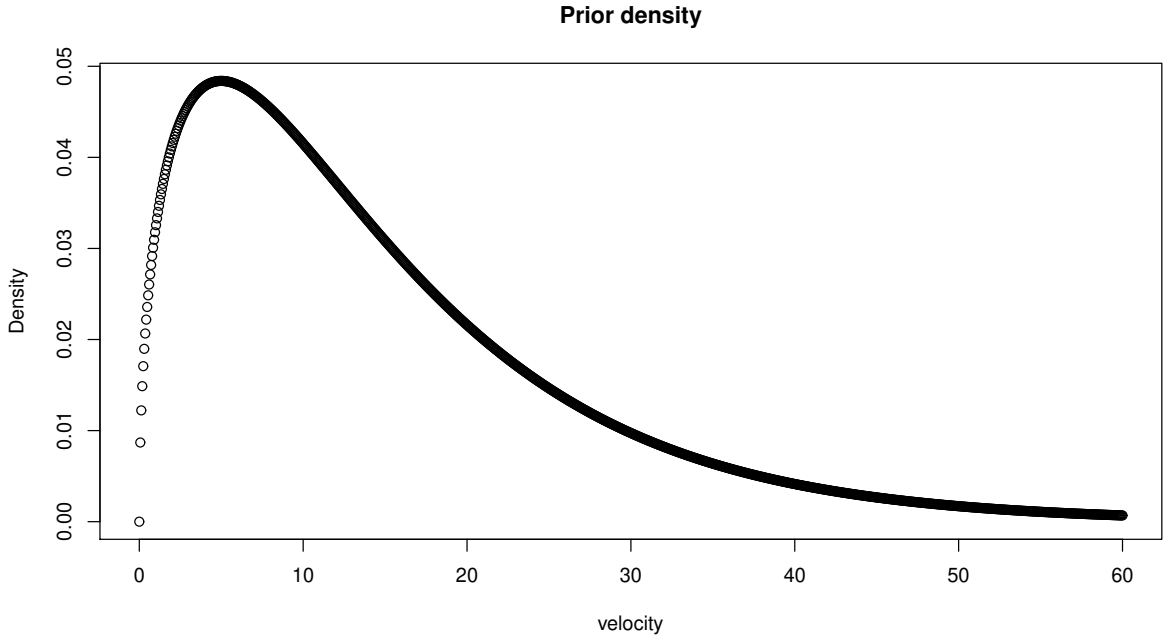

Figure 1: *Prior density*

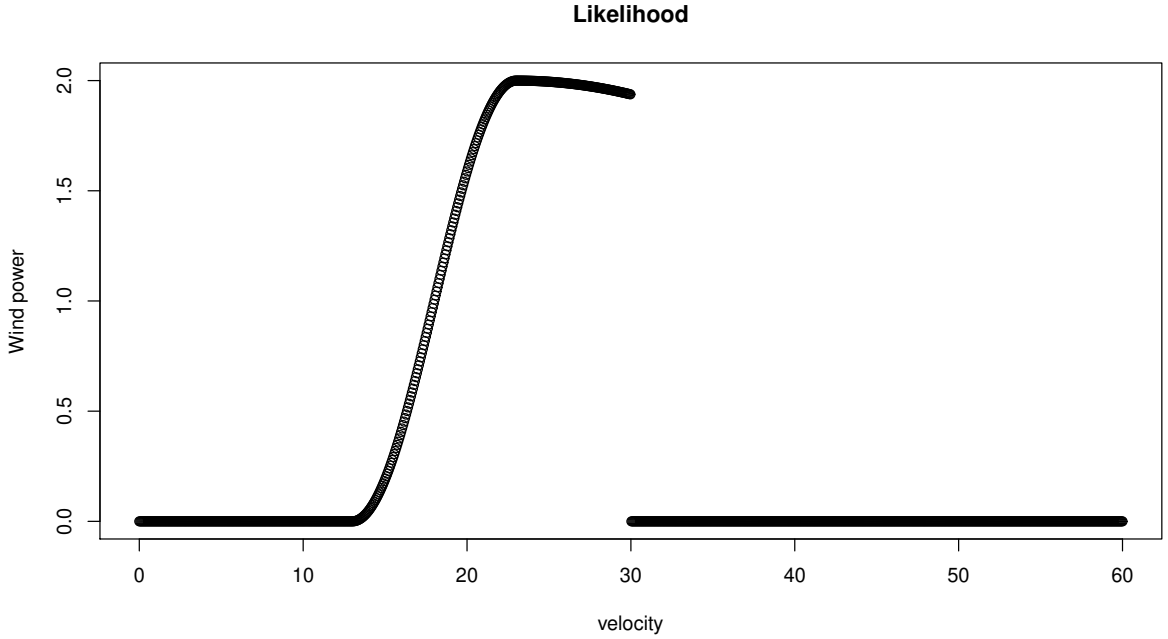

Figure 2: *Likelihood*

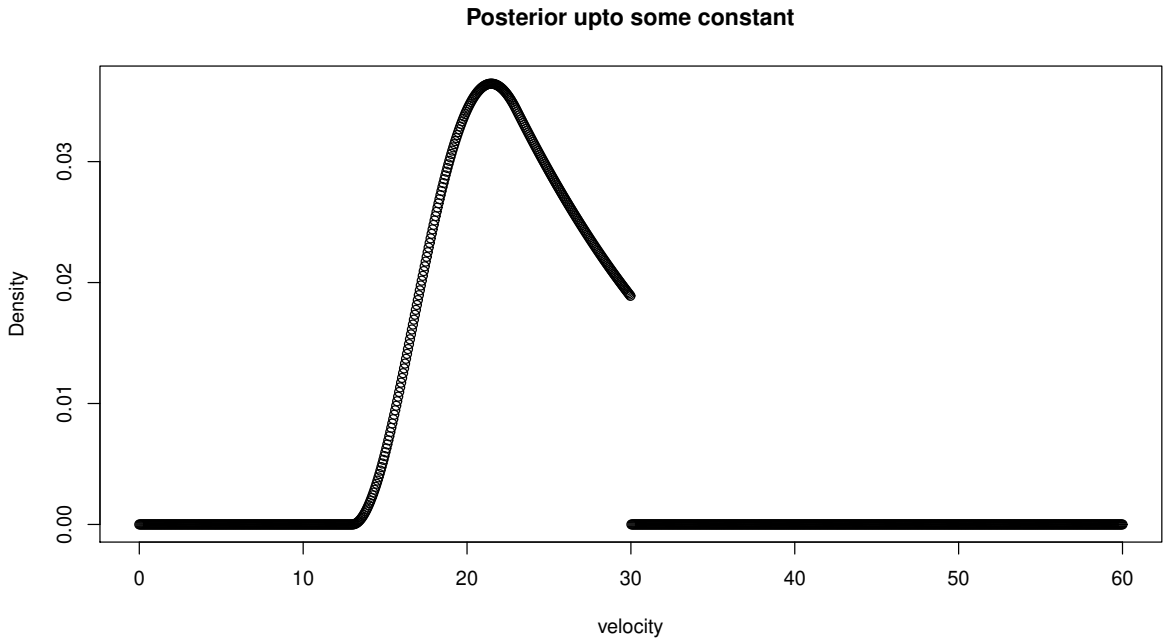

Figure 3: *The posterior density upto a constant*

```
powerSequence <- sapply(velocitySequence, windPowerFunction)
plot(velocitySequence,powerSequence, ylab = "Wind power", xlab = "velocity", main
    ↪  = "Likelihood")
posteriorSequence <- powerSequence*densityVelocity
plot(velocitySequence, posteriorSequence,, ylab = "Density", xlab = "velocity",
    ↪ main = "Posterior upto some constant")
areaUnderTheCurve <- sum(posteriorSequence) / length((posteriorSequence[which(
    ↪ posteriorSequence > 0)]))
areaUnderTheCurve
```

1.4 Q1 - d

We know that the posterior is proportional to the product of likelihood and prior and hence, the posterior can be rewritten as,

$$f(v) = \begin{cases} \left(cos\left(\frac{\pi}{10}\nu - \frac{3\pi}{10}\right) + 1\right) \frac{\beta^\alpha}{\Gamma(\alpha)} \nu^{\alpha-1} \exp\left(-\beta\nu\right) & ; 13 \leq \nu < 23 \\ \frac{1}{780}(1031 - 46\nu - \nu^2) \frac{\beta^\alpha}{\Gamma(\alpha)} \nu^{\alpha-1} \exp\left(-\beta\nu\right) & ; 23 \leq \nu < 30 \\ 0 & ; \text{otherwise} \end{cases}$$

The expected power generated can be then computed as,

$$\begin{aligned} \bar{\nu} &= \int_{13}^{30} f(\nu) \\ &= \mathbf{0.3506} \end{aligned}$$

R code used for this part of the question is,

```
windPowerIntegrateFn <- function(velocity){
  if(velocity >= 13 && velocity < 23){
    wP <- (cos(velocity*pi/10 - 3*pi/10) + 1) * (betaParameter^alphaParameter)*
       ↪ velocity^(alphaParameter - 1)*exp(-betaParameter*velocity)/gamma(
       ↪ alphaParameter)
    return(wP)
  }
  else if(velocity >= 23 && velocity < 30){
    wP <- ((1031 + 46*velocity - velocity^2)/780) * (betaParameter^alphaParameter)
       ↪ *velocity^(alphaParameter - 1)*exp(-betaParameter*velocity)/gamma(
       ↪ alphaParameter)
    return(wP)
  }
  else{
    return(0)
  }
}
integrate(windPowerIntegrateFn,lower = 13, upper = 30)
```

1.5 Q1 - e

The plot for choosing a proposal density is the plot for $f(\nu)$. Figure 4 contains the required plot. The motivation for choosing the proposal function as gamma is the prior being gamma. The expectation and variance are assumed by the fact that the first local maxima for likelihood is obtained at velocity = 23 implying we can assume the expectation for proposal gamma density function as 23. And the standard deviation for proposal gamma density can be assumed as 5. Having expectation and standard deviation, we can find the parameters for proposal gamma density as,

$$\alpha = \frac{\text{Expectation}^2}{Variance} = 469/25$$
$$\beta = \frac{Expectation}{Variance} = 23/25$$

Hence the chosen proposal function is

$$q(\nu) = \text{Gamma}\left(\nu\ ;\ \frac{469}{25}, \frac{23}{25}\right)$$

The R code used for this part of the question is,

```
windPowerPosterior <- sapply(velocitySequence, windPowerIntegrateFn)
plot(velocitySequence, windPowerPosterior, ylab = "Expected␣wind␣power␣upto␣a␣
   ↪ constant", xlab = "velocity", main = "Posterior␣and␣proposal␣fucntion")
points(velocitySequence, dgamma(velocitySequence, 469/25, 23/25)/3, col = "red")
legend(50, 0.03, legend=c("Posterior", "Proprosal"),
       col=c("black", "red"), pch = 1, cex=0.8)
```

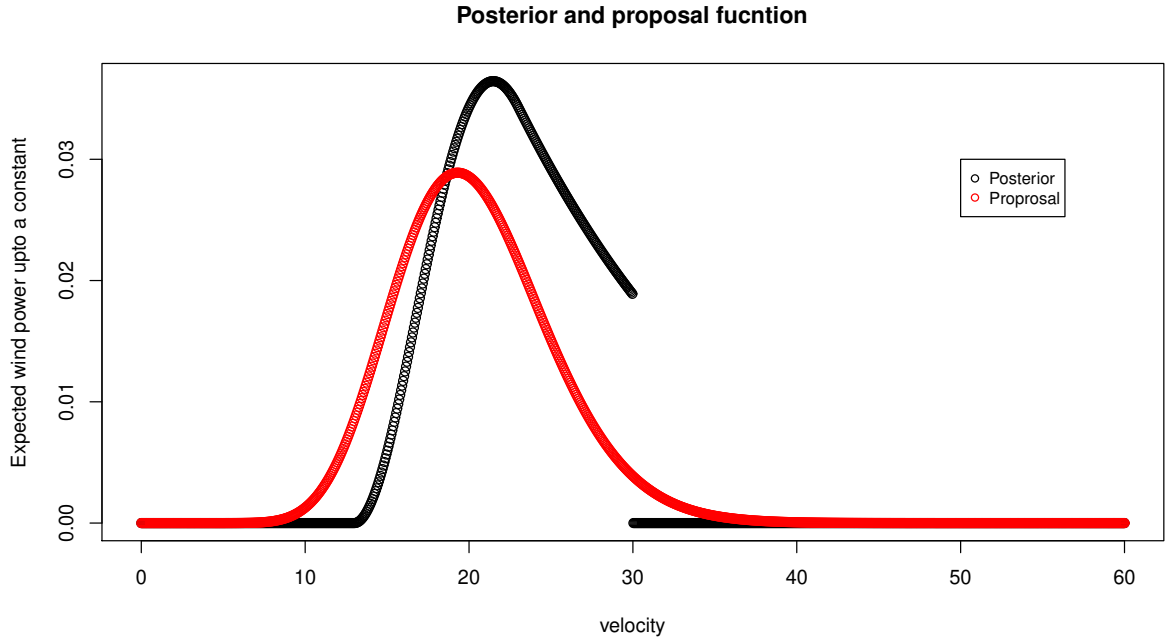

Figure 4: *Posterior density and proposal function*

1.6 Q1 - f

Using importance sampling, expectation can be found by first taking a sample from proposal function and then average over $\frac{Posterior}{proposal} = \frac{f(\nu)}{q(\nu)}$.

The average wind power estimated using importance sampling is 0.3963 which is a very close to the expected wind power computed in Question 1-b (1.2), when the sample size considered was 100000. Hence, this justifies that the chosen proposal function does a good approximation for sampling in this case.

The R code used for this part of the section is,

```
alphaImportance <- 469/25
betaImportance <- 23/25
importanceSamplingFn <- function(velocity){
  if(velocity >= 13 && velocity < 23){
    wP <- (cos(velocity*pi/10 - 3*pi/10) + 1) * (betaParameter^alphaParameter)*
       ↪ velocity^(alphaParameter - 1)*exp(-betaParameter*velocity)/gamma(
       ↪ alphaParameter)
    wP <- wP/((betaImportance^alphaImportance)*velocity^(alphaImportance - 1)*exp
       ↪ (-betaImportance*velocity)/gamma(alphaImportance))
    return(wP)
  }
  else if(velocity >= 23 && velocity < 30){
    wP <- ((1031 + 46*velocity - velocity^2)/780) * (betaParameter^alphaParameter)
       ↪ *velocity^(alphaParameter - 1)*exp(-betaParameter*velocity)/gamma(
       ↪ alphaParameter)
    wP <- wP/((betaImportance^alphaImportance)*velocity^(alphaImportance - 1)*exp
```

```
      ↪ (-betaImportance*velocity)/gamma(alphaImportance))
    return(wP)
  }
  else{
    return(0)
  }
}
sampleImportance <- rgamma(10000,alphaImportance, betaImportance)
imporanceFunctionValue <- sapply(sampleImportance, importanceSamplingFn)
mean(imporanceFunctionValue)
```

2 Question set 2

2.1 Q2 - a

The likelihood function for the poisson density can be written as,

$$L(\lambda\ ;\ Y_0, ..., Y_9) = \prod_{i=0}^{9} \left(\frac{\lambda^i e^{-\lambda}}{i!} \right)^{Y_i}$$

If the data was given for each day, then the likelihood function doesn't change. Infact, the likelihood formed here is assuming the data is collected each day and a histogram was made to form the likelihood function in a straightforward manner.

2.2 Q2 - b

To maximize the likelihood, we can consider log-likelihood and differentiating it.

$$logL = \sum_{i=0}^{9} log \left(\left(\frac{\lambda^i e^{-\lambda}}{i!} \right)^{Y_i} \right)$$

$$\implies \frac{dL}{d\lambda} = \sum_{i=0}^{9} \left(\frac{iY_i}{\lambda} - Y_i \right) = 0$$

$$\implies \lambda = \frac{\sum_{i=0}^{9} iY_i}{\sum_{i=0}^{9} Y_i} = \mathbf{2.156}$$

The plot for observed and expected counts is shown in figure 5. The R code used for this part of the question is,

```
dayCount <- c(162,267,271,185,111,61,27,8,3,1)
deathNotices <- c(0:9)
lambda <- sum(deathNotices*dayCount)/sum(dayCount)
predictedCounts <- sum(dayCount)*dpois(deathNotices,lambda)
predictedCounts
plot(deathNotices, y = dayCount, col = "black", pch = 20,cex = 1.5,xlab = "Death␣
   ↪ notices", ylab = "Frequency", main = "Observed␣and␣Expected␣counts")
points(deathNotices, y = predictedCounts, col = "red", pch = 20, cex = 1.5)
legend(x = 7.5, y= 225, legend = c("Observed", "Predicted"),col = c("black","red"
   ↪ ), pch = c(20,20))
```

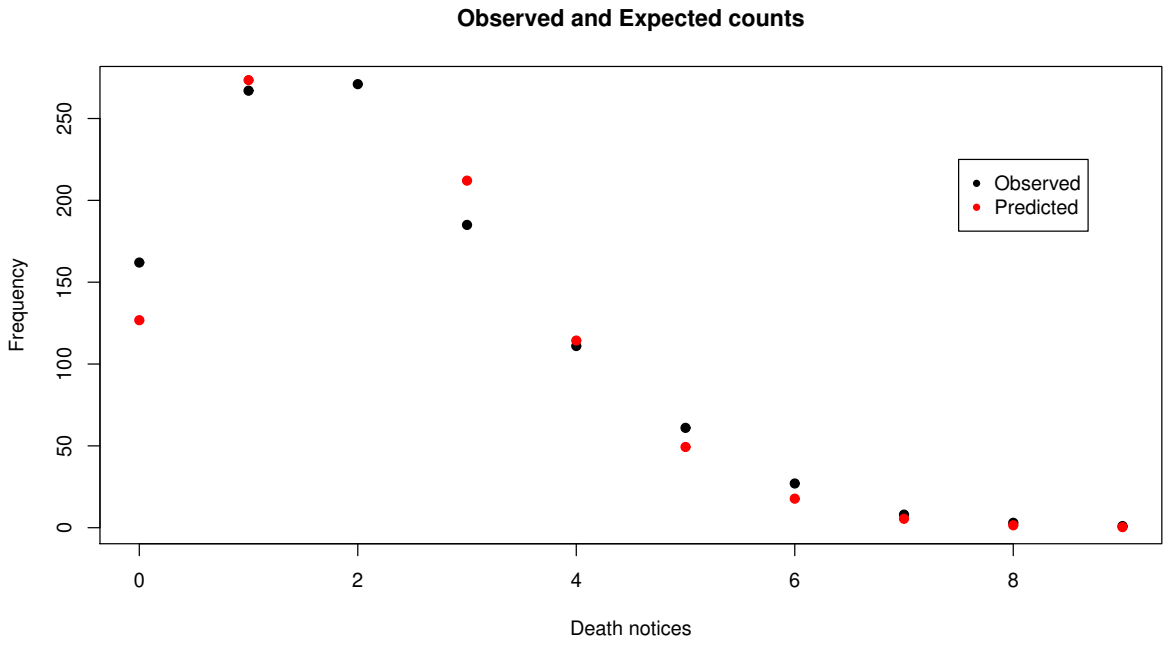

Figure 5: *Observed vs Expected counts*

2.3 Q2 - c

The required likelihood function will be,

$$\pi(Y_0, ..., Y_9, Z_0, ..., Z_9 \mid \lambda_1, \lambda_2, p) = \prod_{i=0}^{9} \left\{ p^{Z_i} \left(\frac{e^{-\lambda_1} \lambda_1^i}{i!} \right)^{Z_i} (1-p)^{Y_i - Z_i} \left(\frac{e^{-\lambda_2} \lambda_2^i}{i!} \right)^{Y_i - Z_i} \right\}$$

2.4 Q2 - d

The posterior for parameter p given all others can be written as,

$$\begin{aligned}
\pi(p \mid Y_0, ..., Y_9, Z_0, ..., Z_9, \lambda_1, \lambda_2) &= \pi(Y_0, ...Y_9, Z_0, ..., Z_9, \lambda_1, \lambda_2 \mid p)\, \pi(p) \\
&= \pi(Y_0, ...Y_9, Z_0, ..., Z_9 \mid p, \lambda_1, \lambda_2)\, \pi(p)\, \pi(\lambda_1)\, \pi(\lambda_2) \\
&= C\, \pi(Y_0, ...Y_9, Z_0, ..., Z_9 \mid p, \lambda_1, \lambda_2) \\
&= C \prod_{i=0}^{9} \left\{ p^{Z_i} \left(\frac{e^{-\lambda_1} \lambda_1^i}{i!} \right)^{Z_i} (1-p)^{Y_i - Z_i} \left(\frac{e^{-\lambda_2} \lambda_2^i}{i!} \right)^{Y_i - Z_i} \right\} \\
&= C \prod_{i=0}^{9} p^{Z_i} (1-p)^{(Y_i - Z_i)} \\
&= C\, p^{\sum_{i=0}^{9} Z_i} (1-p)^{\sum_{i=0}^{9} (Y_i - Z_i)} \\
\pi(p \mid Y_0, ..., Y_9, Z_0, ..., Z_9, \lambda_1, \lambda_2) &= C\, \text{Beta}\left(p\, ;\, \sum_{i=0}^{9} Z_i + 1\, ,\, \sum_{i=0}^{9} (Y_i - Z_i) + 1 \right)
\end{aligned}$$

2.5 Q2 - e

The posterior for λ_1 given all others can be written as,

$$\begin{aligned}
\pi(\lambda_1 \mid Y_0, ..., Y_9, Z_0, ..., Z_9, \lambda_2, p) &= \pi(Y_0, ...Y_9, Z_0, ..., Z_9 \mid p, \lambda_1, \lambda_2)\, \pi(p)\, \pi(\lambda_2)\, \pi(\lambda_1) \\
&= C\pi(Y_0, ...Y_9, Z_0, ..., Z_9 \mid p, \lambda_1, \lambda_2)\, \pi(\lambda_1) \\
&= C\ \mathrm{Gam}(\lambda_1\ ;\ \alpha, \beta) \prod_{i=0}^{9} \left\{ p^{Z_i} \left(\frac{e^{-\lambda_1}\lambda_1^i}{i!} \right)^{Z_i} (1-p)^{Y_i - Z_i} \left(\frac{e^{-\lambda_2}\lambda_2^i}{i!} \right)^{Y_i - Z_i} \right\} \\
&= C\ \mathrm{Gam}(\lambda_1\ ;\ \alpha, \beta) \prod_{i=0}^{9} \left\{ \left(\frac{e^{-\lambda_1}\lambda_1^i}{i!} \right)^{Z_i} \left(\frac{e^{-\lambda_2}\lambda_2^i}{i!} \right)^{Y_i - Z_i} \right\} \\
&= C\ \mathrm{Gam}(\lambda_1\ ;\ \alpha, \beta) \prod_{i=0}^{9} \lambda_1^{i\, Z_i}\ \exp\left(-\lambda_1 Z_i\right) \\
&= C\ \mathrm{Gam}(\lambda_1\ ;\ \alpha, \beta)\ \lambda_1^{\sum_{i=0}^{9} i\, Z_i}\ \exp\left(-\lambda_1 \sum_{i=0}^{9} Z_i\right) \\
\pi(\lambda_1 \mid Y_0, ..., Y_9, Z_0, ..., Z_9, \lambda_2, p) &= C\ \mathrm{Gam}\left(\lambda_1\ ; \alpha + \sum_{i=0}^{9} i\, Z_i\ ,\ \beta + \sum_{i=0}^{9} Z_i\right)
\end{aligned}$$

Note that, a Gamma-Poisson conjugacy is used in the derivation.

The posterior for λ_2 given all others can be written as,

$$\begin{aligned}
\pi(\lambda_2 \mid Y_0, ..., Y_9, Z_0, ..., Z_9, \lambda_1, p) &= \pi(Y_0, ...Y_9, Z_0, ..., Z_9 \mid p, \lambda_1, \lambda_2)\, \pi(p)\, \pi(\lambda_2)\, \pi(\lambda_1) \\
&= C\pi(Y_0, ...Y_9, Z_0, ..., Z_9 \mid p, \lambda_1, \lambda_2)\, \pi(\lambda_2) \\
&= C\ \mathrm{Gam}(\lambda_2\ ;\ \alpha, \beta) \prod_{i=0}^{9} \lambda_2^{i\,(Y_i - Z_i)}\ \exp\left(-\lambda_2 (Y_i - Z_i)\right. \\
&= C\ \mathrm{Gam}(\lambda_2\ ;\ \alpha, \beta)\ \lambda_2^{\sum_{i=0}^{9} i\,(Y_i - Z_i)}\ \exp\left(-\lambda_2 \sum_{i=0}^{9} (Y_i - Z_i)\right) \\
\pi(\lambda_2 \mid Y_0, ..., Y_9, Z_0, ..., Z_9, \lambda_1, p) &= C\ \mathrm{Gam}\left(\lambda_2\ ; \alpha + \sum_{i=0}^{9} i\,(Y_i - Z_i)\ ,\ \beta + \sum_{i=0}^{9} (Y_i - Z_i)\right)
\end{aligned}$$

2.6 Q2 - f

We know,

$$\begin{aligned}
&\pi(Y_0, ..., Y_9, Z_0, ..., Z_9 \mid \lambda_1, \lambda_2, p) = \pi(Z_0, ...Z_9 \mid Y_0, ..., Y_9, p, \lambda_1, \lambda_2)\pi(Y_0, ..., Y_9 \mid p, \lambda_1, \lambda_2) \\
\implies\ &\pi(Z_0, ..., Z_9 \mid Y_0, ..., Y_9, p, \lambda_1, \lambda_2) \propto \pi(Z_0, ..., Z_9, Y_0, ..., Y_9 \mid p, \lambda_1, \lambda_2)
\end{aligned}$$

But for each of Z_i, if we know Y_i, p, λ_1, λ_2 then the values of $Y_j\ ; j \neq i$ doesn't affect the value of Z_i. Hence we can write it as,

$$\begin{aligned}
&\pi(Z_i \mid Y_i, p, \lambda_1, \lambda_2) \propto \pi(Z_i, Y_i \mid p, \lambda_1, \lambda_2) \\
\implies\ &\pi(Z_i \mid Y_i, p, \lambda_1, \lambda_2) \propto p^{Z_i} \left(\frac{e^{-\lambda_1}\lambda_1^i}{i!} \right)^{Z_i} (1-p)^{Y_i - Z_i} \left(\frac{e^{-\lambda_2}\lambda_2^i}{i!} \right)^{Y_i - Z_i} \\
&\propto \left(p\lambda_1^i e^{-\lambda_1}\right)^{Z_i} \left((1-p)\lambda_2^i e^{-\lambda_2}\right)^{(Y_i - Z_i)}
\end{aligned}$$

Assuming,

$$m = p\lambda_1^i e^{-\lambda_1}$$
$$n = (1-p)\lambda_2^i e^{-\lambda_2}$$

We have,

$$\begin{aligned}\pi(Z_i \mid Y_i, p, \lambda_1, \lambda_2) &\propto m^{Z_i} n^{Y_i - Z_i} \\ &\propto \frac{m^{Z_i}}{(m+n)^{Z_i}} \frac{n^{Y_i - Z_i}}{(m+n)^{Y_i - Z_i}} (m+n)^{Y_i} \\ &\propto \left(\frac{m}{m+n}\right)^{Z_i} \left(\frac{n}{m+n}\right)^{Y_i - Z_i}\end{aligned}$$

Substituting m and n, we get,

$$\pi(Z_i \mid Y_i, p, \lambda_1, \lambda_2) \propto \left(\frac{p\lambda_1^i e^{-\lambda_1}}{p\lambda_1^i e^{-\lambda_1} + (1-p)\lambda_2^i e^{-\lambda_2}}\right)^{Z_i} \left(\frac{(1-p)\lambda_2^i e^{-\lambda_2}}{p\lambda_1^i e^{-\lambda_1} + (1-p)\lambda_2^i e^{-\lambda_2}}\right)^{(Y_i - Z_i)}$$

$$\implies \pi(Z_i \mid Y_i, p, \lambda_1, \lambda_2) \propto \text{Bin}\left(Y_i\,,\, \frac{p\lambda_1^i e^{-\lambda_1}}{p\lambda_1^i e^{-\lambda_1} + (1-p)\lambda_2^i e^{-\lambda_2}}\right)$$

2.7 Q2 - g

Gibbs sampling can be implemented here and the algorithm to do that is,

```
1. Find a good starting point for parameters. This can be done by sampling a
   ↪ value from their respective prior.
2. Estimate Z's using the binomial distribution with Y's and starting point of
   ↪ parameters.
3. Update parameters using respective distributions along with Y's and estimated
   ↪ Z's.
4. Update Z's using binomial distribution with Y's and updated parameters.
5. Iterate steps 3 and 4 for large number of times.
```

A burn-in was observed for the parameters and hence, initial 2000 values of the parameters were excluded in estimating their expectations. The probability parameter p quickly converges to a small value implying the distribution of Y is largely dependent on the distribution offered by λ_2. Although, the experiment was run for $1e5$ times, a plot is made for initial $1e4$ iterations for clarity. And it appears as the convergence is achieved by $1e4$ iterations.

The trace plots for λ_1, λ_2 and p is shown in figure 6, 7 and 8 respectively. Histogram for λ_1 and λ_2 is shown in figure 9 and 10 respectively.

The expected values are,

$$\lambda_1 = 2.996156$$
$$\lambda_2 = 2.996853$$
$$p = 2.547 \times 10^{-7}$$

The R code for this part of the question is,

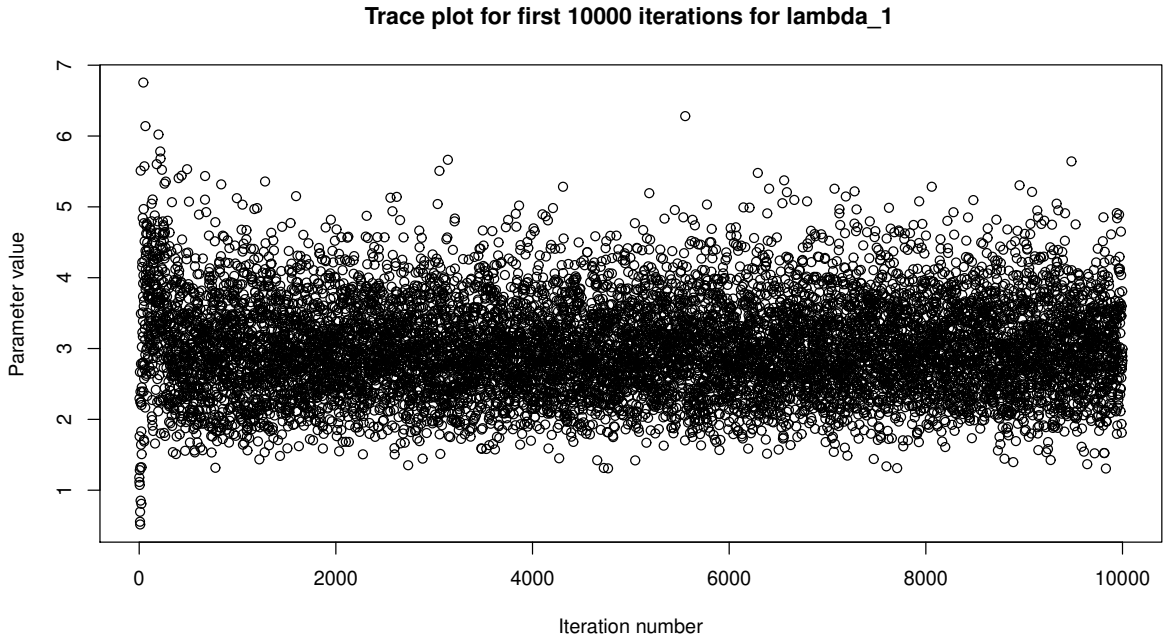

Figure 6: *Trace plot for parameter* λ_1

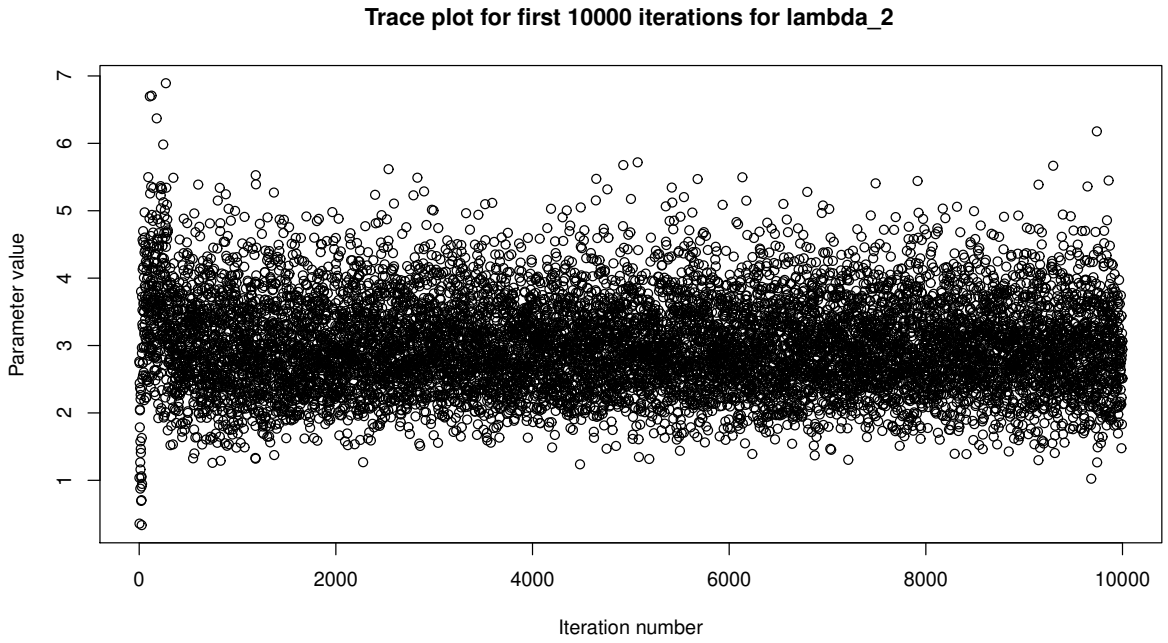

Figure 7: *Trace plot for parameter* λ_2

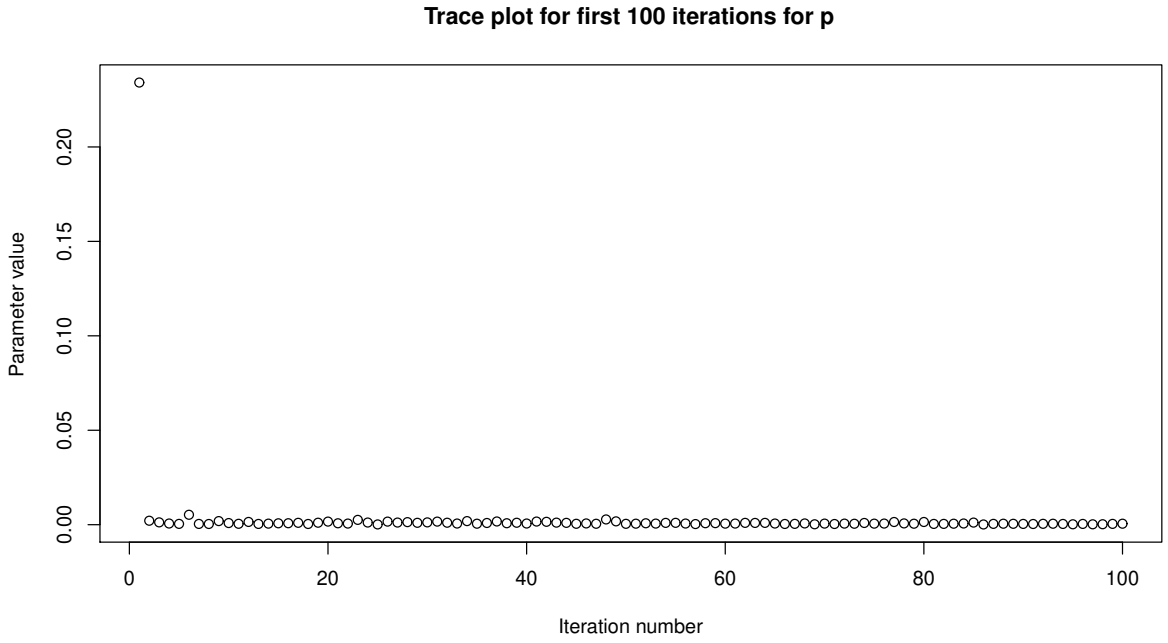

Figure 8: *Trace plot for parameter p*

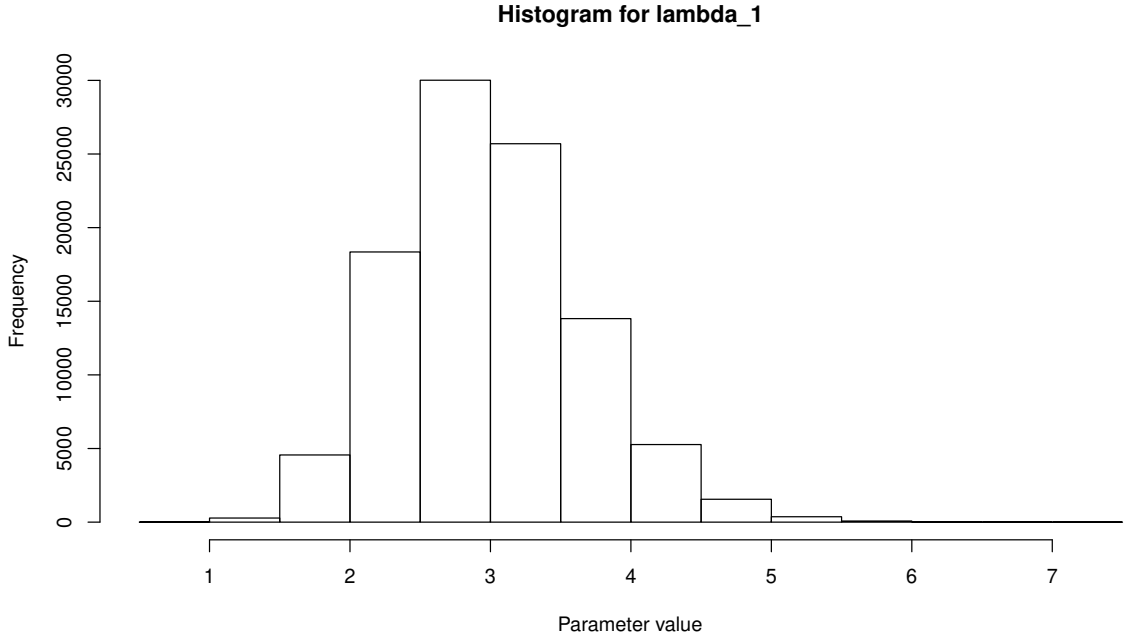

Figure 9: *Histogram for λ_1*

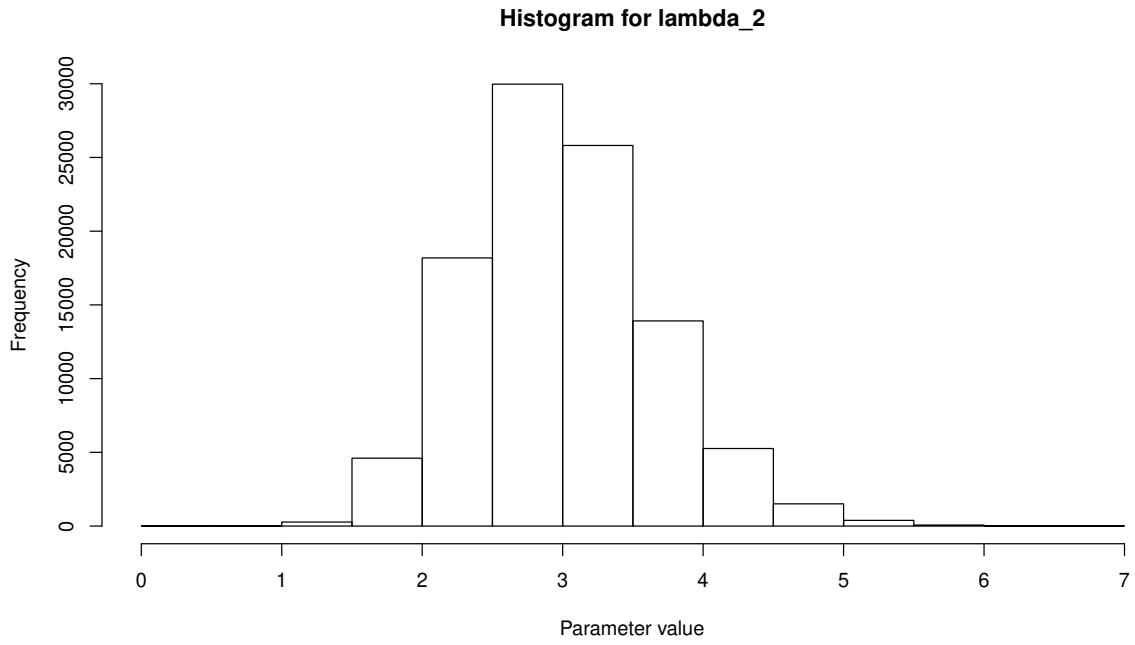

Figure 10: *Histogram for* λ_2

```
lengthPrior <- 1
N <- 100000

posteriorLambda1 <- rep(0, times = N)
posteriorLambda2 <- rep(0, times = N)
p <- rep(0, times = N)

posteriorLambda1[1] <- rgamma(lengthPrior,1,1)
posteriorLambda2[1] <- rgamma(lengthPrior,1,1)
p[1] <- runif(lengthPrior)

Zs <- function(Y,i, p, l1, l2){
  numerator = p*(l1^i)*exp(-l1)
  denom = p*(l1^i)*exp(-l1) + (1-p)*(l2^i)*exp(-l2)
  prob = numerator/denom
  return(rbinom(1, Y, prob))
}

estimatedZ <- rep(0,times = 10)
iVector <- c(0:9)

for (i in 2:N){
  for (iZ in 1:length(deathNotices)){
    estimatedZ[i] <- Zs(dayCount[iZ], iZ - 1, p[i-1],posteriorLambda1[i - 1],
       ↪ posteriorLambda2[i - 1] )
```

```
  }
  p[i] <- rbeta(1, sum(estimatedZ) + 1, sum(dayCount - estimatedZ) + 1)
  posteriorLambda1[i] <- rgamma(1,sum(iVector*estimatedZ) + 1, sum(estimatedZ) +
     ↪ 1)
  posteriorLambda2[i] <- rgamma(1,sum(iVector*estimatedZ) + 1, sum(estimatedZ) +
     ↪ 1)
}
hist(p)
hist(posteriorLambda1, ylab = "Frequency", xlab = "Parameter␣value", main = "
   ↪ Histogram␣for␣lambda_1")
hist(posteriorLambda2, ylab = "Frequency", xlab = "Parameter␣value", main = "
   ↪ Histogram␣for␣lambda_2")

plot(posteriorLambda1[1:10000], ylab = "Parameter␣value", xlab = "Iteration␣
   ↪ number", main = "Trace␣plot␣for␣first␣10000␣iterations␣for␣lambda_1")
plot(posteriorLambda2[1:10000], ylab = "Parameter␣value", xlab = "Iteration␣
   ↪ number", main = "Trace␣plot␣for␣first␣10000␣iterations␣for␣lambda_2")
plot(p[1:100], ylab = "Parameter␣value", xlab = "Iteration␣number", main = "Trace
   ↪ ␣plot␣for␣first␣100␣iterations␣for␣p")

avgP <- mean(p[2001:N])
lambda1 <- mean(posteriorLambda1[2001:N])
lambda2 <- mean(posteriorLambda2[2001:N])
avgP
lambda1
lambda2
```