Bayesian p-values for the Guns laws data

Chapter 5.6: Posterior predictive checks

The data for this analysis come from “Firearm legislation and firearm mortality in the USA: a cross-sectional, state-level study” by Kalesan et. al. (2016). The response variable, Y_i, is the number of firearm-related deaths in 2010 in state i. This is regressed onto five potential confounders (Z_{ij}),

1. 2009 firearm death rate per 10,000 people
2. Firearm ownership rate quartile
3. Unemployment rate quartile
4. Non-firearm homicide rate quartile
5. Firearm export rate quartile

The covariates of interest are the indicators of whether the state has certain gun laws. Let X_{il} indicate that state i has law number l. In this example, we simply use the number of laws $X_i = \sum_l X_{il}$ as the covariate.

In this analysis our objective is illustrate the use of posterior predictive checks to verify that the model fits well. We compare two models:

Model 1 - Poisson regression: The first model is the Poisson model we have fit previously

$$Y_i \sim \text{Poisson}(\lambda_i) \text{ where } \lambda_i = N_i \exp(\alpha + \sum_{j=1}^{5} Z_{ij}\beta_j + X_i\beta_6)$$

and N_i is the state's population.

Model 2 - Negative-binomial regression: A limitation of the Poisson likelihood is that the variance equals the mean. To allow for a larger variance than the mean (i.e., overdispersion) we replace the Poisson likelihood with the negative binomial likelihood the same mean λ_i and over-dispersion parameter $m > 0$

$$Y_i \sim \text{NB}\left(\frac{m}{\lambda_i + m}, m\right).$$

Posterior predictive checks are performed for the following test statistics:

- $D_1(Y) = \max(Y_1, \ldots, Y_n)$
- $D_2(Y) = \min(Y_1, \ldots, Y_n)$
- $D_3(Y) = \max(Y_1, \ldots, Y_n) - \min(Y_1, \ldots, Y_n)$
- $D_4(Y) = \max(Y_1/N_1, \ldots, Y_n/N_n)$
- $D_5(Y) = \min(Y_1/N_1, \ldots, Y_n/N_n)$
- $D_6(Y) = \max(Y_1/N_1, \ldots, Y_n/N_n) - \min(Y_1/N_1, \ldots, Y_n/N_n)$

1. Load the data

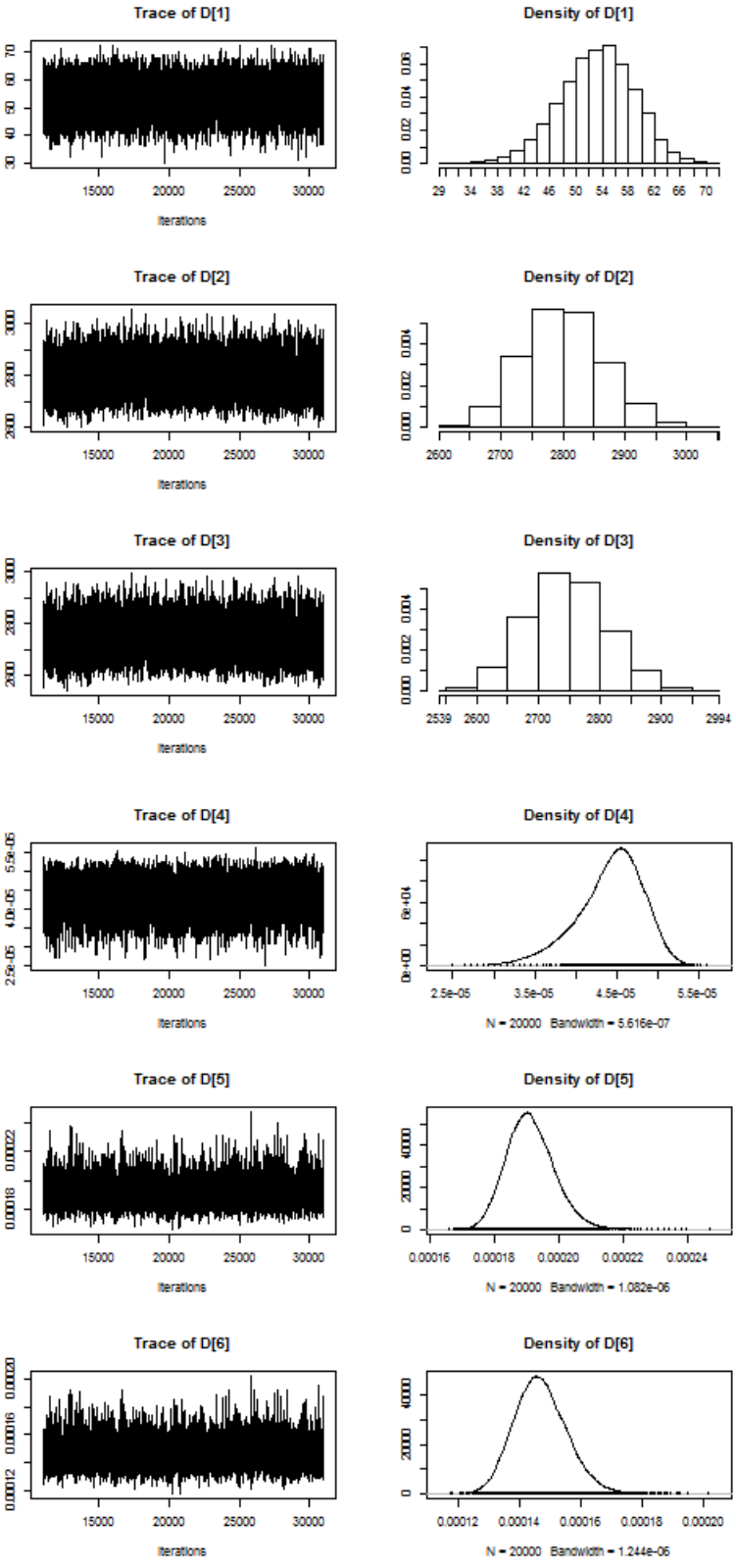
```
load("guns.RData")
X  <- rowSums(X)
n  <- length(Y)
```

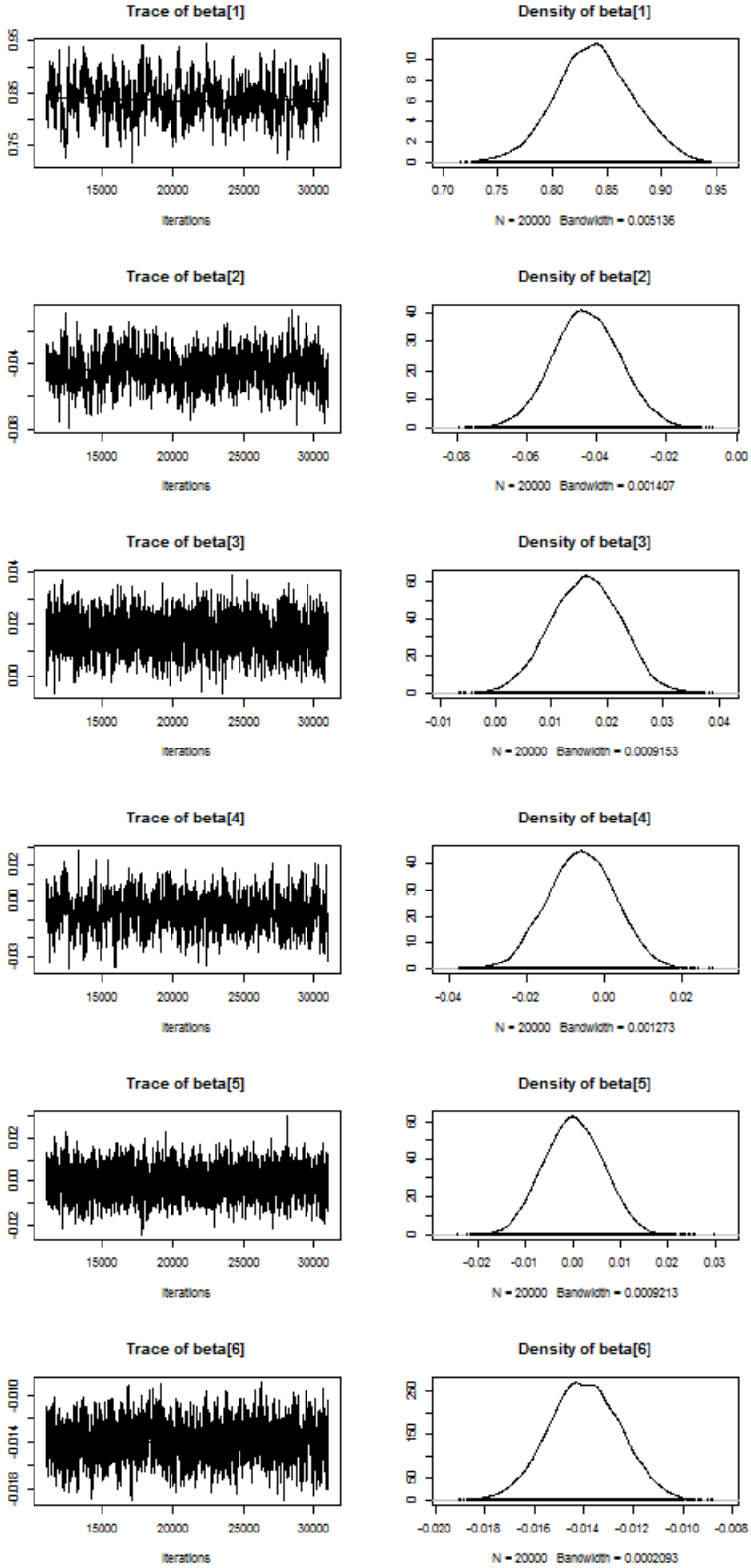
2. Specify two competing models:

```
# (1) Poisson regression

model_string1 <- "model{

 # Likelihood
 for(i in 1:n){
   Y[i]           ~ dpois(lambda[i])
   log(lambda[i]) <- log(N[i]) + alpha + inprod(Z[i,],beta[1:5]) + X[i]*beta[6]
 }

 #Priors
  for(j in 1:6){
     beta[j] ~ dnorm(0,0.1)
  }
  alpha ~ dnorm(0,0.1)


 # Posterior preditive checks
 for(i in 1:n){
   Y2[i]    ~ dpois(lambda[i])
   rate[i] <- Y2[i]/N[i]
 }
 D[1] <- min(Y2[])
 D[2] <- max(Y2[])
 D[3] <- max(Y2[])-min(Y2[])
 D[4] <- min(rate[])
 D[5] <- max(rate[])
 D[6] <- max(rate[])-min(rate[])

}"


# (2) Over-dispersed Poisson
model_string2 <- "model{

 # Likelihood (note hierarchical centering)
 for(i in 1:n){
   Y[i]            ~ dnegbin(q[i],m)
   q[i]           <- m/(m+N[i]*lambda[i])
   log(lambda[i]) <- alpha + inprod(Z[i,],beta[1:5]) + X[i]*beta[6]
 }

 #Priors
  for(j in 1:6){
     beta[j] ~ dnorm(0,0.1)
  }
  alpha ~ dnorm(0,0.1)
  m     ~ dgamma(0.1,0.1)

 # Posterior preditive checks
 for(i in 1:n){
   Y2[i]    ~ dnegbin(q[i],m)
   rate[i] <- Y2[i]/N[i]
 }

 D[1] <- min(Y2[])
 D[2] <- max(Y2[])
 D[3] <- max(Y2[])-min(Y2[])
 D[4] <- min(rate[])
 D[5] <- max(rate[])
 D[6] <- max(rate[])-min(rate[])

}"
```

3. Fit the two models

```
library(rjags)


model1 <- jags.model(textConnection(model_string1),
                     data = list(Y=Y,N=N,n=n,X=X,Z=Z),n.chains=1,
                     quiet=TRUE)
update(model1, 10000, progress.bar="none")
samps1 <- coda.samples(model1,
          variable.names=c("D","beta"),
          n.iter=20000, progress.bar="none")
plot(samps1)
```

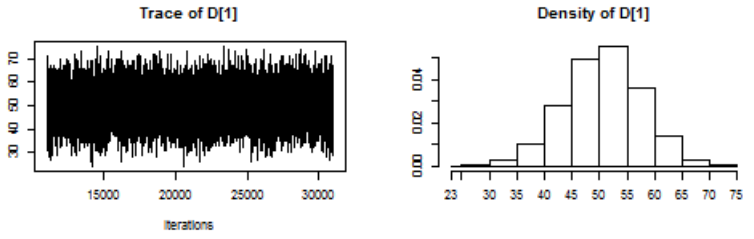




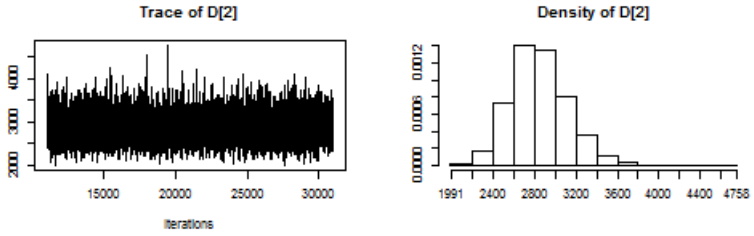

```
print(summary(samps1))
```

```
##
## Iterations = 11001:31000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 20000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean        SD  Naive SE Time-series SE
## D[1]     5.372e+01 5.577e+00 3.944e-02      5.359e-02
## D[2]     2.801e+03 6.431e+01 4.548e-01      9.672e-01
## D[3]     2.747e+03 6.465e+01 4.571e-01      1.013e+00
## D[4]     4.421e-05 4.045e-06 2.860e-08      6.365e-08
## D[5]     1.912e-04 7.726e-06 5.463e-08      1.389e-07
## D[6]     1.470e-04 8.693e-06 6.147e-08      1.702e-07
## beta[1]  8.381e-01 3.535e-02 2.500e-04      2.885e-03
## beta[2] -4.299e-02 9.620e-03 6.802e-05      5.021e-04
## beta[3]  1.614e-02 6.259e-03 4.425e-05      2.369e-04
## beta[4] -5.941e-03 8.703e-03 6.154e-05      4.342e-04
## beta[5]  1.472e-04 6.299e-03 4.454e-05      2.071e-04
## beta[6] -1.397e-02 1.431e-03 1.012e-05      6.288e-05
##
## 2. Quantiles for each variable:
##
##               2.5%        25%        50%        75%      97.5%
## D[1]     4.200e+01  5.000e+01  5.400e+01  5.800e+01  6.400e+01
## D[2]     2.681e+03  2.756e+03  2.800e+03  2.844e+03  2.931e+03
## D[3]     2.627e+03  2.702e+03  2.746e+03  2.790e+03  2.878e+03
## D[4]     3.455e-05  4.190e-05  4.484e-05  4.705e-05  5.072e-05
## D[5]     1.778e-04  1.860e-04  1.906e-04  1.959e-04  2.076e-04
## D[6]     1.318e-04  1.410e-04  1.465e-04  1.524e-04  1.653e-04
## beta[1]  7.677e-01  8.147e-01  8.377e-01  8.617e-01  9.084e-01
## beta[2] -6.175e-02 -4.948e-02 -4.312e-02 -3.651e-02 -2.370e-02
## beta[3]  3.788e-03  1.185e-02  1.620e-02  2.046e-02  2.817e-02
## beta[4] -2.283e-02 -1.180e-02 -5.900e-03 -3.486e-05  1.094e-02
## beta[5] -1.197e-02 -4.152e-03  1.147e-04  4.431e-03  1.230e-02
## beta[6] -1.680e-02 -1.493e-02 -1.398e-02 -1.299e-02 -1.115e-02
```

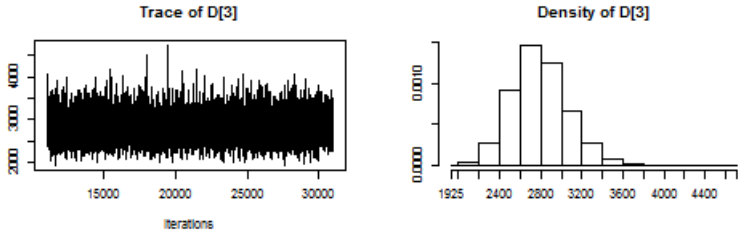
```
D1  <- samps1[[1]]


model2 <- jags.model(textConnection(model_string2),
                  data = list(Y=Y,N=N,n=n,X=X,Z=Z),n.chains=1,
                  quiet=TRUE)
update(model2, 10000, progress.bar="none")
samps2 <- coda.samples(model2,
          variable.names=c("D","beta"),
          n.iter=20000, progress.bar="none")
plot(samps2)
```


Trace of D[1]
30 40 50 60 70
15000 20000 25000 30000
Iterations

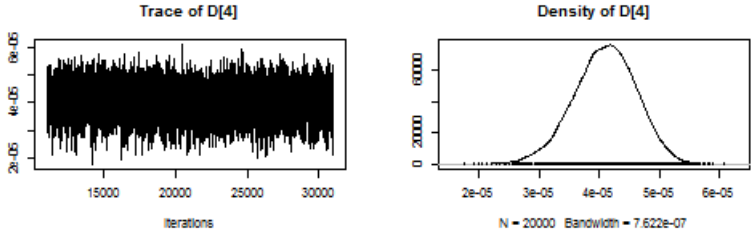
Density of D[1]
0.00 0.02 0.04
23 30 35 40 45 50 55 60 65 70 75


Trace of D[2]
2000 3000 4000
15000 20000 25000 30000
Iterations

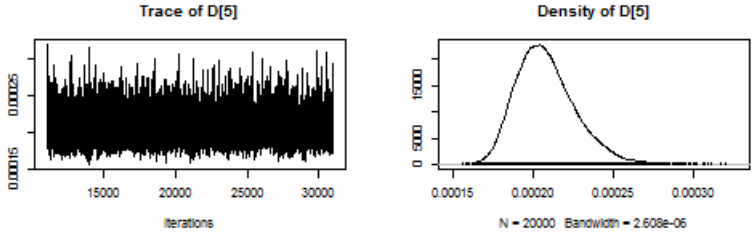
Density of D[2]
0.0000 0.0006 0.0012
1991 2400 2800 3200 3600 4000 4400 4758


Trace of D[3]
2000 3000 4000
15000 20000 25000 30000
Iterations

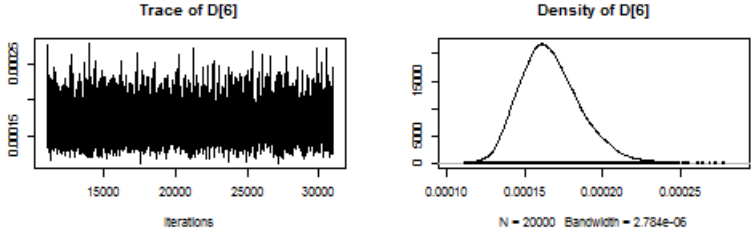
Density of D[3]
0.0000 0.0010
1925 2400 2800 3200 3600 4000 4400


Trace of D[4]
2e-05 4e-05 6e-05
15000 20000 25000 30000
Iterations

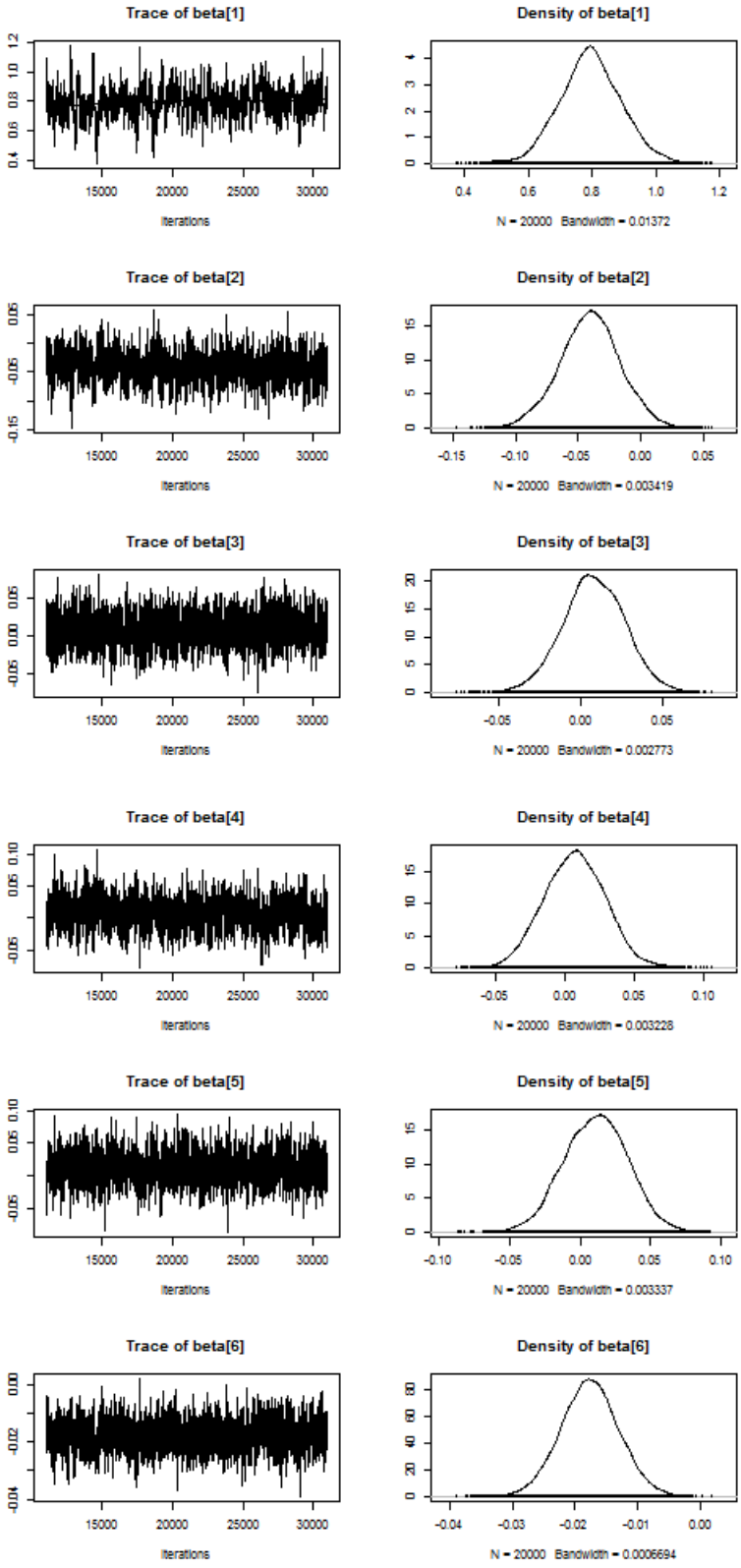
Density of D[4]
0 20000 60000
2e-05 3e-05 4e-05 5e-05 6e-05
N = 20000 Bandwidth = 7.622e-07


Trace of D[5]
0.00015 0.00025
15000 20000 25000 30000
Iterations

Density of D[5]
0 5000 15000
0.00015 0.00020 0.00025 0.00030
N = 20000 Bandwidth = 2.608e-06


Trace of D[6]
0.00015 0.00025
15000 20000 25000 30000
Iterations

Density of D[6]
0 5000 15000
0.00010 0.00015 0.00020 0.00025
N = 20000 Bandwidth = 2.784e-06



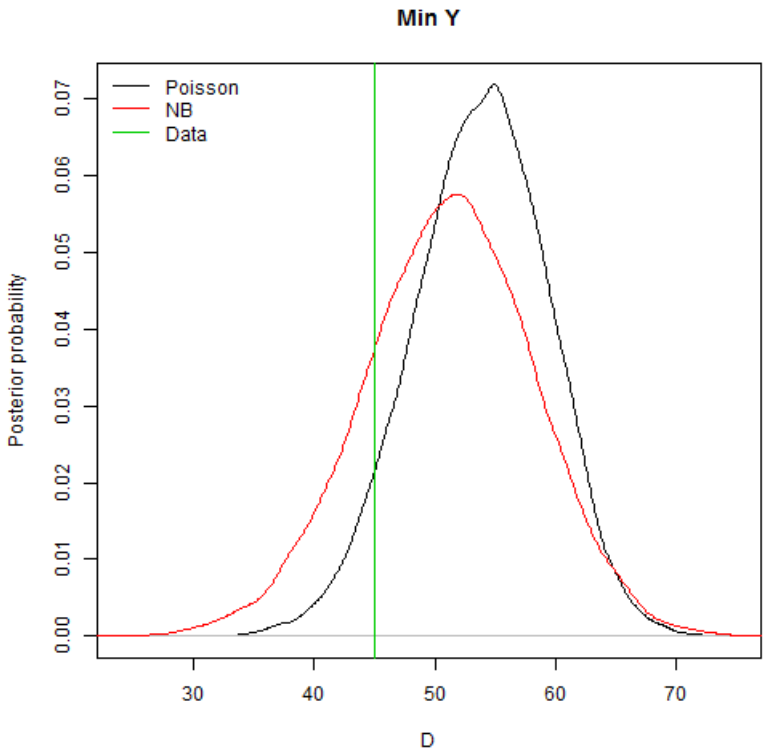

```
print(summary(samps2))
```

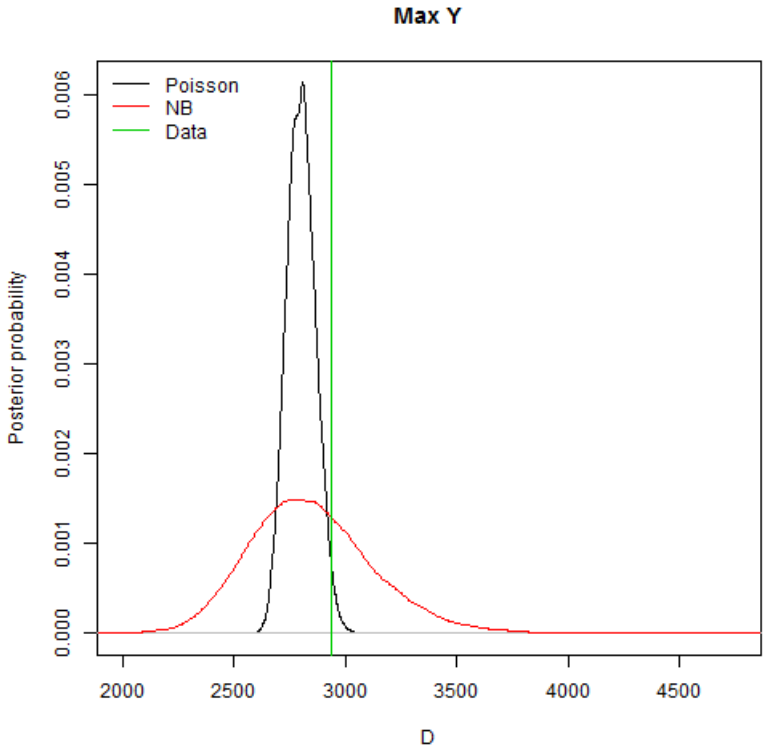
```
##
## Iterations = 11001:31000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 20000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean        SD  Naive SE Time-series SE
## D[1]     5.110e+01 7.046e+00 4.982e-02      7.436e-02
## D[2]     2.846e+03 2.778e+02 1.964e+00      3.177e+00
## D[3]     2.795e+03 2.778e+02 1.964e+00      3.177e+00
## D[4]     4.089e-05 5.235e-06 3.702e-08      7.085e-08
## D[5]     2.078e-04 1.879e-05 1.328e-07      3.266e-07
## D[6]     1.669e-04 1.975e-05 1.396e-07      3.862e-07
## beta[1]  7.955e-01 9.910e-02 7.007e-04      7.381e-03
## beta[2] -4.060e-02 2.416e-02 1.708e-04      1.057e-03
## beta[3]  7.993e-03 1.909e-02 1.350e-04      6.671e-04
## beta[4]  6.788e-03 2.207e-02 1.560e-04      9.827e-04
## beta[5]  1.125e-02 2.282e-02 1.613e-04      9.442e-04
## beta[6] -1.767e-02 4.688e-03 3.315e-05      1.929e-04
##
## 2. Quantiles for each variable:
##
##               2.5%        25%        50%        75%      97.5%
## D[1]     3.700e+01  4.600e+01  5.100e+01  5.600e+01  6.500e+01
## D[2]     2.360e+03  2.654e+03  2.826e+03  3.015e+03  3.451e+03
## D[3]     2.309e+03  2.603e+03  2.775e+03  2.963e+03  3.399e+03
## D[4]     3.014e-05  3.749e-05  4.117e-05  4.448e-05  5.071e-05
## D[5]     1.774e-04  1.946e-04  2.058e-04  2.185e-04  2.502e-04
## D[6]     1.341e-04  1.531e-04  1.648e-04  1.786e-04  2.111e-04
## beta[1]  6.040e-01  7.329e-01  7.951e-01  8.586e-01  9.941e-01
## beta[2] -8.948e-02 -5.628e-02 -4.038e-02 -2.495e-02  6.588e-03
## beta[3] -2.994e-02 -4.410e-03  7.870e-03  2.100e-02  4.522e-02
## beta[4] -3.557e-02 -8.307e-03  6.875e-03  2.167e-02  4.965e-02
## beta[5] -3.373e-02 -4.309e-03  1.174e-02  2.683e-02  5.536e-02
## beta[6] -2.690e-02 -2.076e-02 -1.768e-02 -1.463e-02 -8.421e-03
```

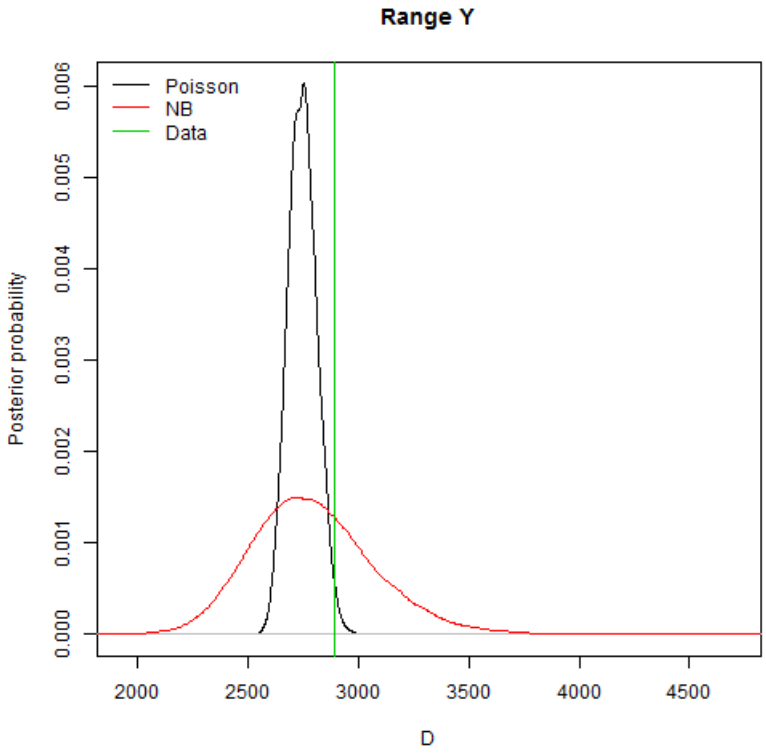
```
D2  <- samps2[[1]]
```

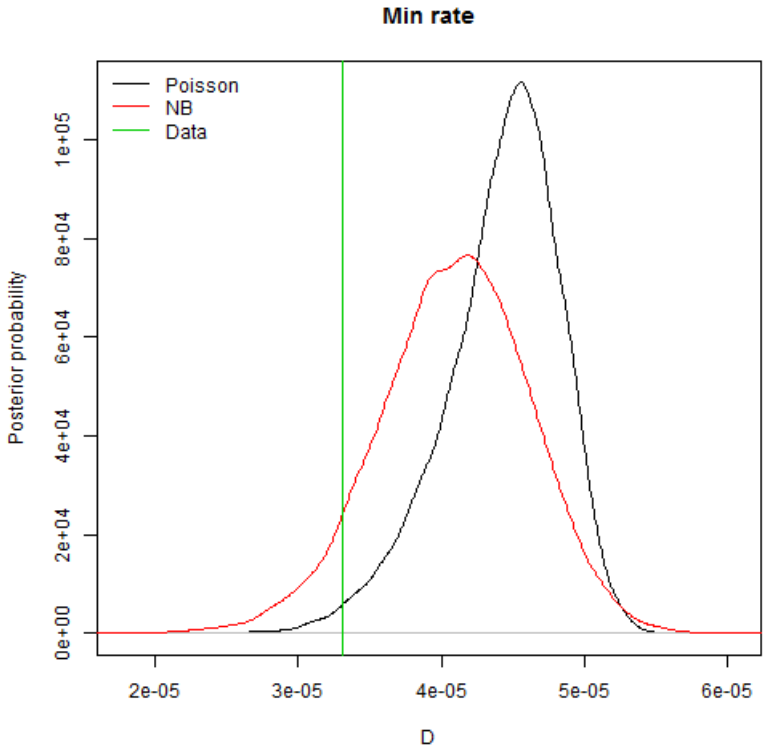
4. Compute the Bayesian p-values

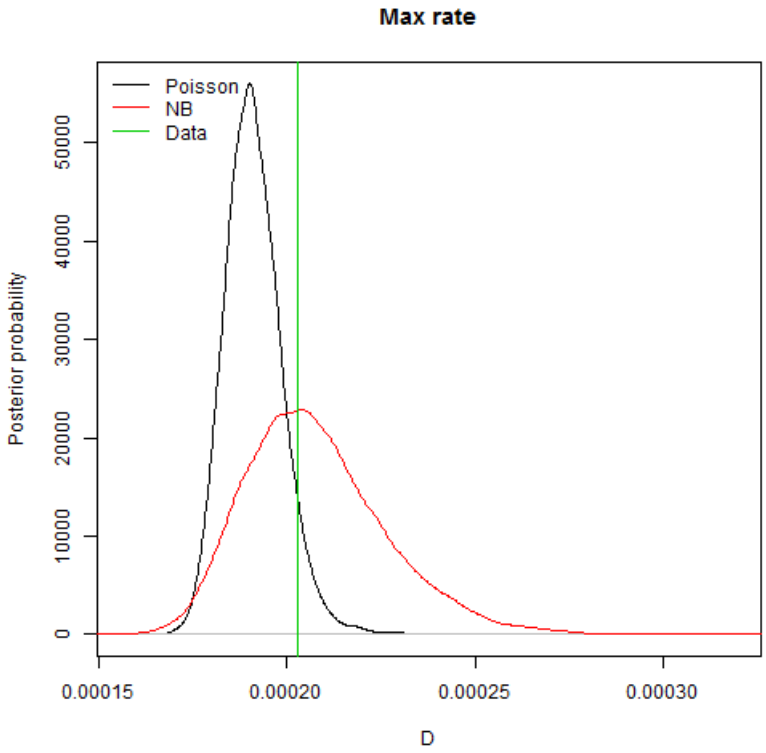
```r
# Compute the test stats for the data
rate <- Y/N
D0   <- c(   min(Y),   max(Y),      max(Y)-min(Y),
          min(rate),max(rate),max(rate)-min(rate))
Dnames <- c("Min Y", "Max Y", "Range Y", "Min rate", "Max rate", "Range rate")

# Compute the test stats for the models

pval1 <- rep(0,6)
names(pval1)<-Dnames
pval2 <- pval1

for(j in 1:6){
  plot(density(D1[,j]),xlim=range(c(D0[j],D1[,j],D2[,j])),
                       xlab="D",ylab="Posterior probability",
                       main=Dnames[j])
 lines(density(D2[,j]),col=2)
 abline(v=D0[j],col=3)
 legend("topleft",c("Poisson","NB","Data"),lty=1,col=1:3,bty="n")

 pval1[j] <- mean(D1[,j]>D0[j])
 pval2[j] <- mean(D2[,j]>D0[j])
}
```

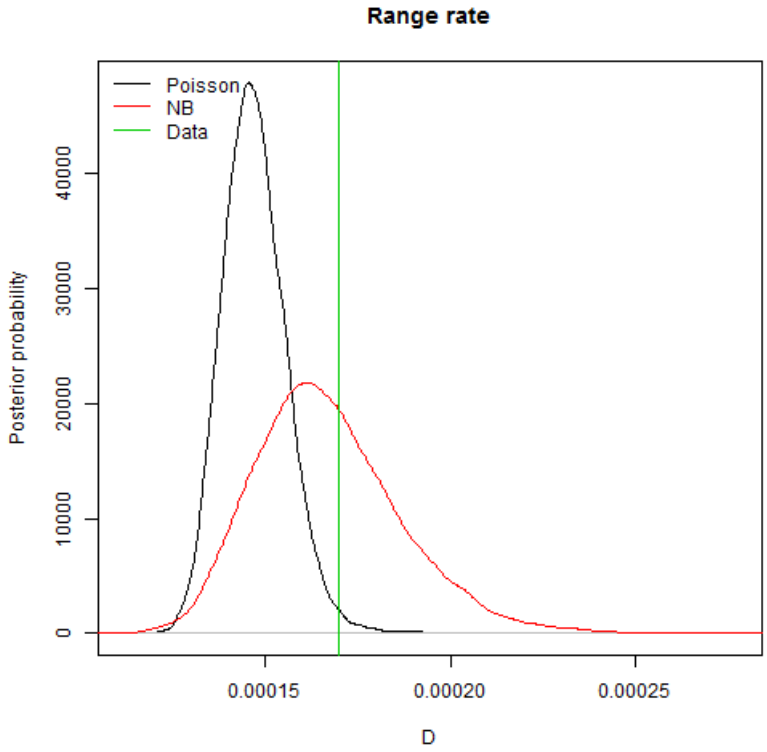



Max Y
Poisson
NB
Data
Posterior probability
0.000
0.001
0.002
0.003
0.004
0.005
0.006
2000
2500
3000
3500
4000
4500
D


Range Y
Poisson
NB
Data
Posterior probability
0.000
0.001
0.002
0.003
0.004
0.005
0.006
2000
2500
3000
3500
4000
4500
D


Min rate
Poisson
NB
Data
Posterior probability
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
2e-05
3e-05
4e-05
5e-05
6e-05
D


Max rate
Poisson
NB
Data
Posterior probability
0
10000
20000
30000
40000
50000
0.00015
0.00020
0.00025
0.00030
D



5. Results

```
pval1
```

```
##      Min Y      Max Y    Range Y   Min rate   Max rate Range rate
##    0.92440    0.02100    0.01610    0.98635    0.06800    0.01125
```

```
pval2
```

```
##      Min Y      Max Y    Range Y   Min rate   Max rate Range rate
##    0.79130    0.34375    0.33545    0.92460    0.56570    0.40190
```

The regular Poisson model has several p-values near zero or one, so it doesn't seem to fit well. The negative binomial model fits better.