Bayesian p-values for the Guns laws data

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, s the number of firearm-related deaths in 2010 in state i. This is regressed onto five potential confounders Z_{ij}.

The covariates of interest are the indicators of whether the state has certain gun laws. Let X_{il} ndicate 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 s 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("//Users/kamaladadashova/Documents/DoctoralCourses/Applied Bayesian Statistics/Lecture Notes with A
 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

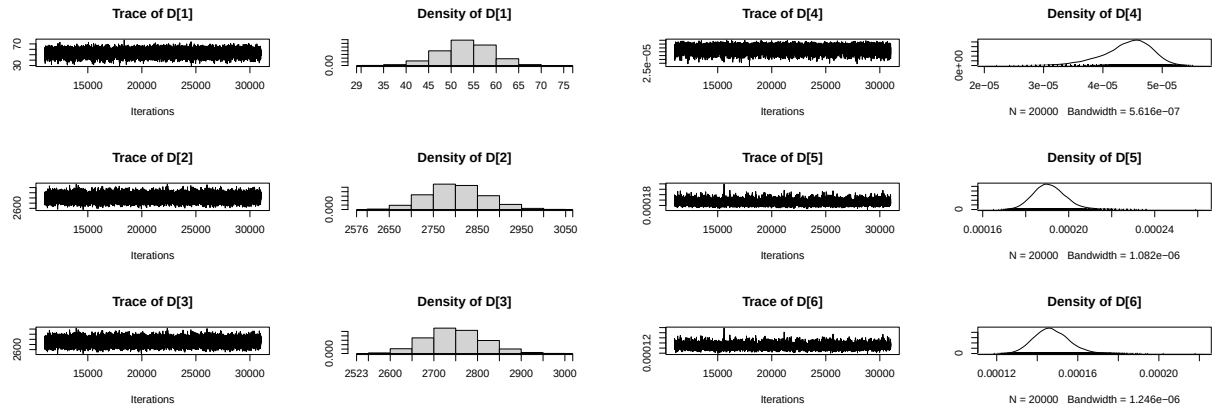
```
library(rjags)
```

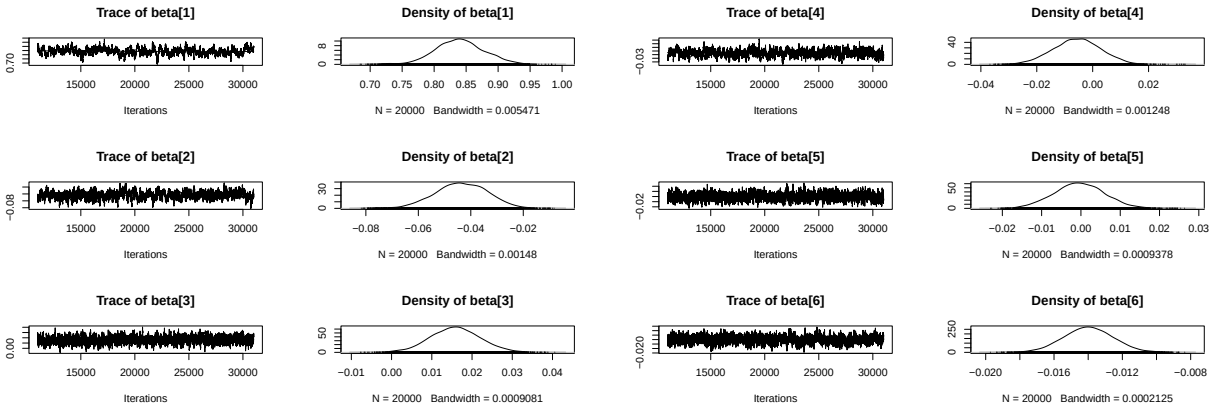
```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.1
```

```
## Loaded modules: basemod,bugs
```

```
  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)
```



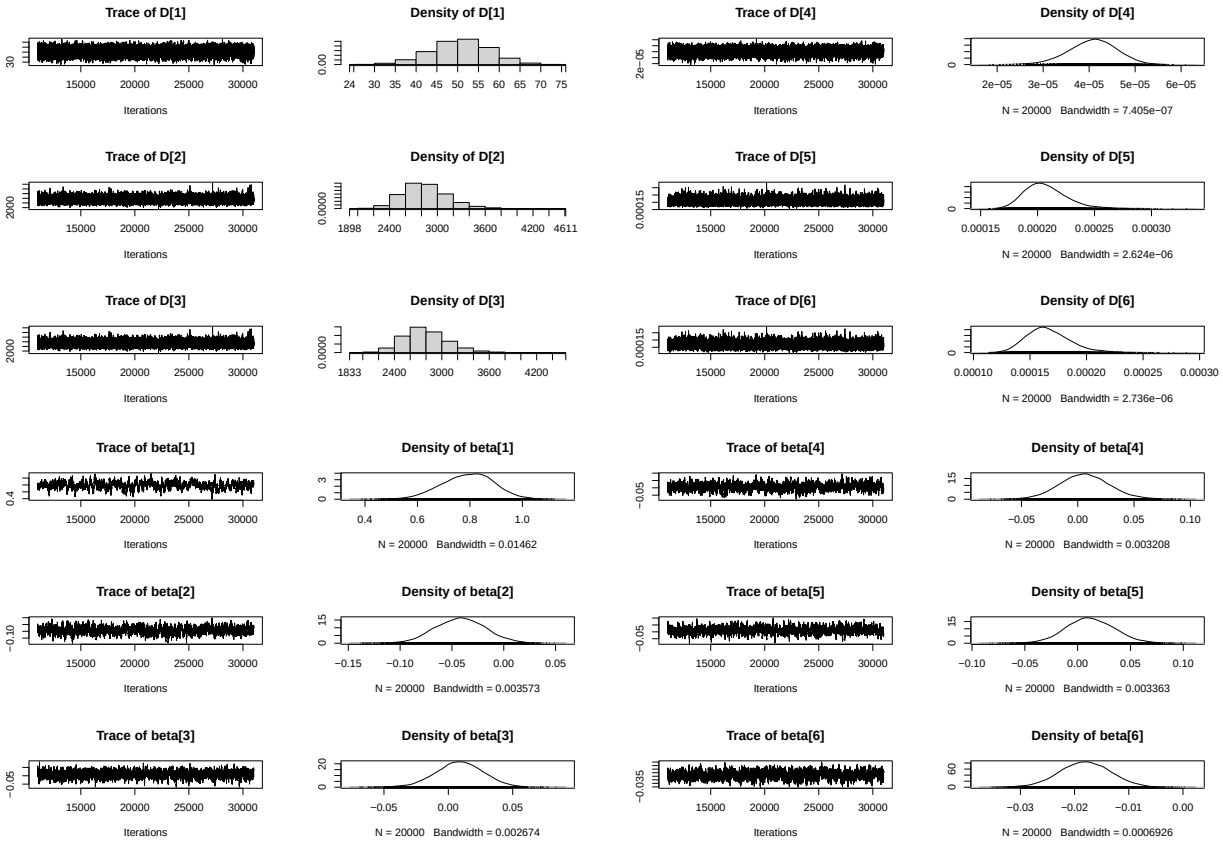



```
print(summary(samps1))
```

```
##
## 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]     5.367e+01 5.582e+00 3.947e-02      5.377e-02
## D[2]     2.802e+03 6.488e+01 4.588e-01      1.005e+00
## D[3]     2.749e+03 6.525e+01 4.614e-01      1.033e+00
## D[4]     4.408e-05 4.107e-06 2.904e-08      5.847e-08
## D[5]     1.912e-04 7.760e-06 5.487e-08      1.435e-07
## D[6]     1.472e-04 8.761e-06 6.195e-08      1.876e-07
## beta[1]  8.401e-01 3.855e-02 2.726e-04      3.318e-03
## beta[2] -4.366e-02 1.012e-02 7.155e-05      5.828e-04
## beta[3]  1.586e-02 6.209e-03 4.390e-05      2.287e-04
## beta[4] -5.639e-03 8.535e-03 6.035e-05      4.214e-04
## beta[5] -4.114e-04 6.437e-03 4.552e-05      2.152e-04
## beta[6] -1.399e-02 1.469e-03 1.039e-05      5.862e-05
##
## 2. Quantiles for each variable:
##
##              2.5%        25%        50%        75%      97.5%
## D[1]     4.200e+01  5.000e+01  5.400e+01  5.700e+01  6.400e+01
## D[2]     2.681e+03  2.757e+03  2.800e+03  2.845e+03  2.933e+03
## D[3]     2.626e+03  2.703e+03  2.746e+03  2.792e+03  2.880e+03
## D[4]     3.455e-05  4.190e-05  4.475e-05  4.705e-05  5.072e-05
## D[5]     1.776e-04  1.860e-04  1.908e-04  1.959e-04  2.078e-04
## D[6]     1.317e-04  1.411e-04  1.466e-04  1.525e-04  1.656e-04
## beta[1]  7.655e-01  8.146e-01  8.394e-01  8.647e-01  9.166e-01
## beta[2] -6.377e-02 -5.026e-02 -4.356e-02 -3.657e-02 -2.476e-02
## beta[3]  3.673e-03  1.164e-02  1.583e-02  1.998e-02  2.825e-02
## beta[4] -2.247e-02 -1.133e-02 -5.575e-03  1.060e-04  1.065e-02
## beta[5] -1.290e-02 -4.714e-03 -4.577e-04  3.878e-03  1.235e-02
```

```
## beta[6] -1.686e-02 -1.496e-02 -1.400e-02 -1.302e-02 -1.111e-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)
```




```
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:
##
```

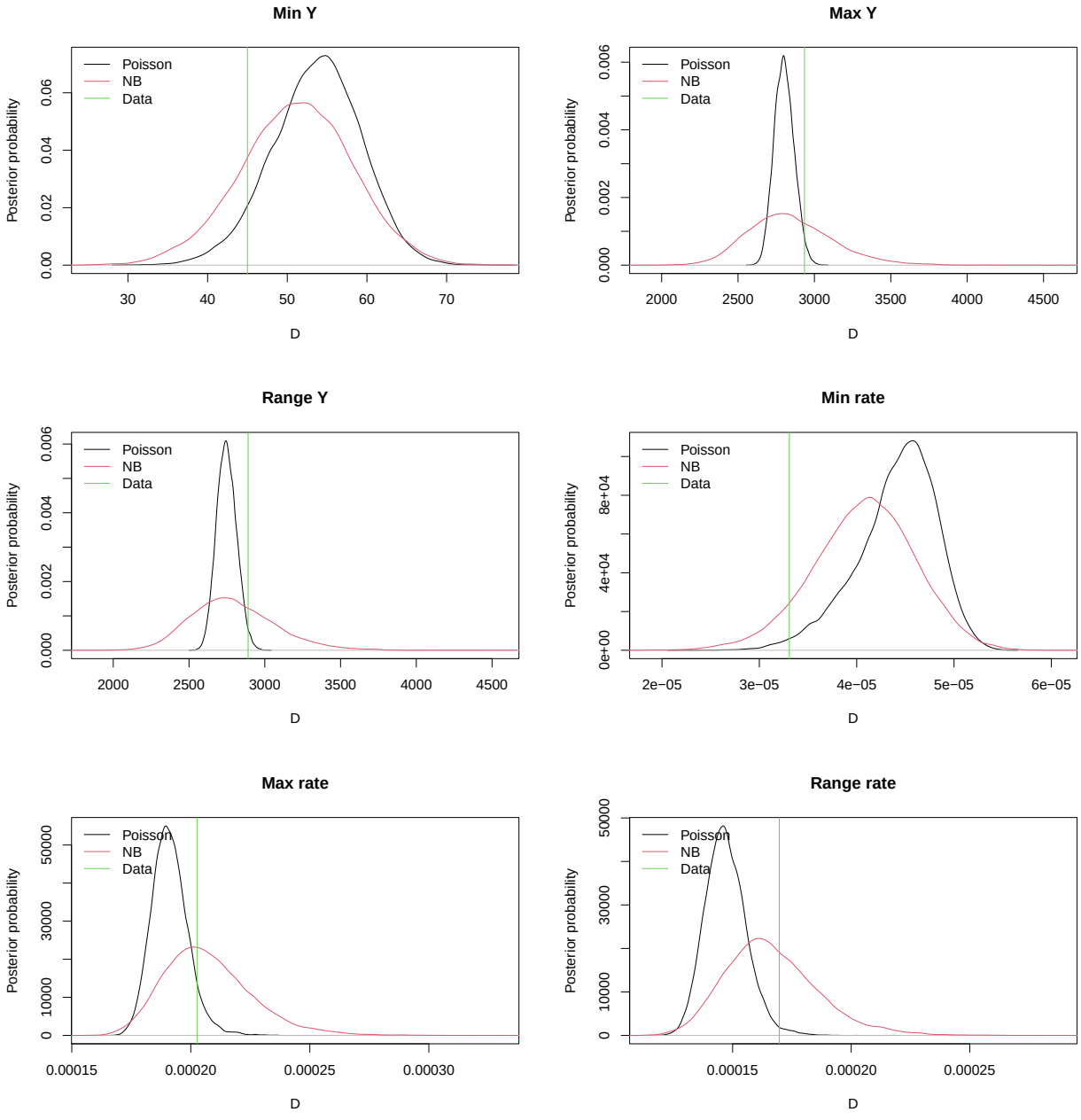
```
##               Mean        SD  Naive SE Time-series SE
## D[1]     5.113e+01 7.008e+00 4.956e-02      7.575e-02
## D[2]     2.843e+03 2.775e+02 1.962e+00      3.581e+00
## D[3]     2.792e+03 2.775e+02 1.962e+00      3.583e+00
## D[4]     4.074e-05 5.256e-06 3.717e-08      7.474e-08
## D[5]     2.072e-04 1.857e-05 1.313e-07      3.318e-07
## D[6]     1.665e-04 1.957e-05 1.384e-07      3.820e-07
## beta[1]  7.915e-01 9.999e-02 7.071e-04      7.582e-03
## beta[2] -4.193e-02 2.443e-02 1.728e-04      1.093e-03
## beta[3]  8.632e-03 1.871e-02 1.323e-04      6.392e-04
## beta[4]  6.220e-03 2.230e-02 1.577e-04      8.966e-04
## beta[5]  1.188e-02 2.300e-02 1.626e-04      9.678e-04
## beta[6] -1.806e-02 4.736e-03 3.349e-05      1.974e-04
##
## 2. Quantiles for each variable:
##
##               2.5%        25%        50%        75%      97.5%
## D[1]     3.700e+01  4.700e+01  5.100e+01  5.600e+01  6.500e+01
## D[2]     2.366e+03  2.650e+03  2.820e+03  3.012e+03  3.451e+03
## D[3]     2.316e+03  2.599e+03  2.769e+03  2.961e+03  3.401e+03
## D[4]     2.993e-05  3.749e-05  4.108e-05  4.428e-05  5.055e-05
## D[5]     1.769e-04  1.941e-04  2.052e-04  2.182e-04  2.502e-04
## D[6]     1.338e-04  1.529e-04  1.645e-04  1.780e-04  2.116e-04
## beta[1]  5.905e-01  7.253e-01  7.961e-01  8.611e-01  9.777e-01
## beta[2] -8.958e-02 -5.843e-02 -4.178e-02 -2.531e-02  5.838e-03
## beta[3] -2.928e-02 -3.345e-03  8.839e-03  2.116e-02  4.493e-02
## beta[4] -3.733e-02 -8.675e-03  5.931e-03  2.072e-02  5.186e-02
## beta[5] -3.287e-02 -3.657e-03  1.148e-02  2.722e-02  5.796e-02
## beta[6] -2.744e-02 -2.124e-02 -1.805e-02 -1.484e-02 -8.810e-03
```

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

4. Compute the Bayesian p-values

```
# 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])
 }
```



5. Results

```
pval1
```

```
##      Min Y      Max Y    Range Y   Min rate   Max rate Range rate
##    0.92465    0.02290    0.01805    0.98500    0.06565    0.01320
```

```
pval2
```

```
##      Min Y      Max Y    Range Y   Min rate   Max rate Range rate
##    0.79300    0.33945    0.33170    0.91800    0.55245    0.39320
```

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.s