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 and libraries

```
X  <- rowSums(X)
n  <- length(Y)
```

2. 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

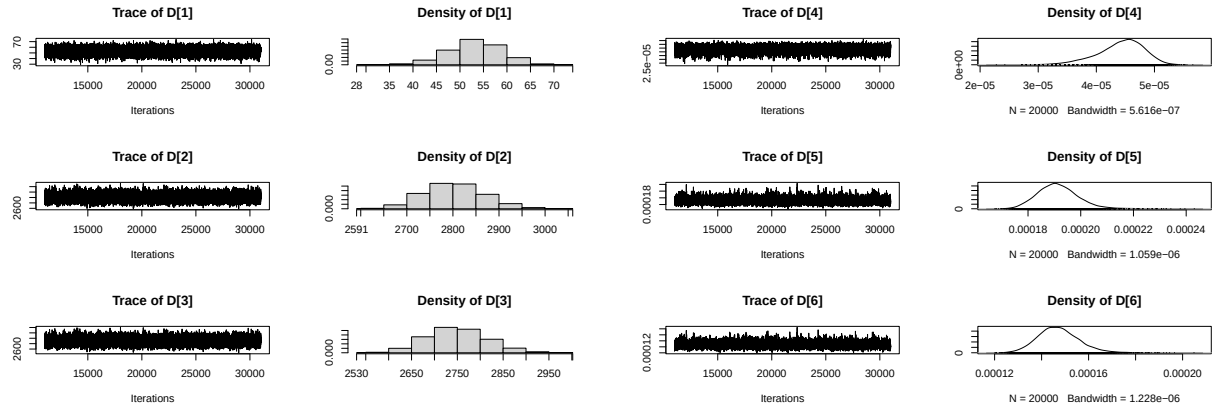
```
library(rjags)
```

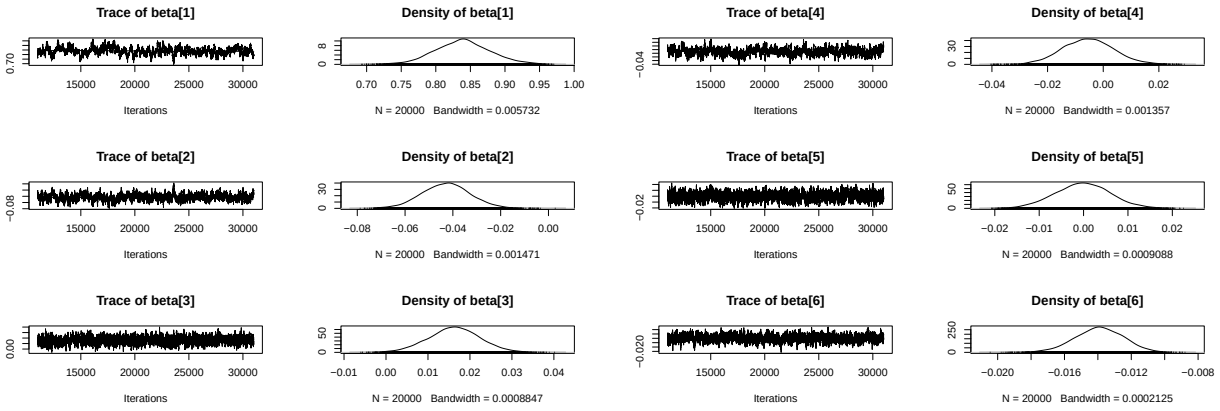
```
## Loading required package: coda
```

```
## 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)
```



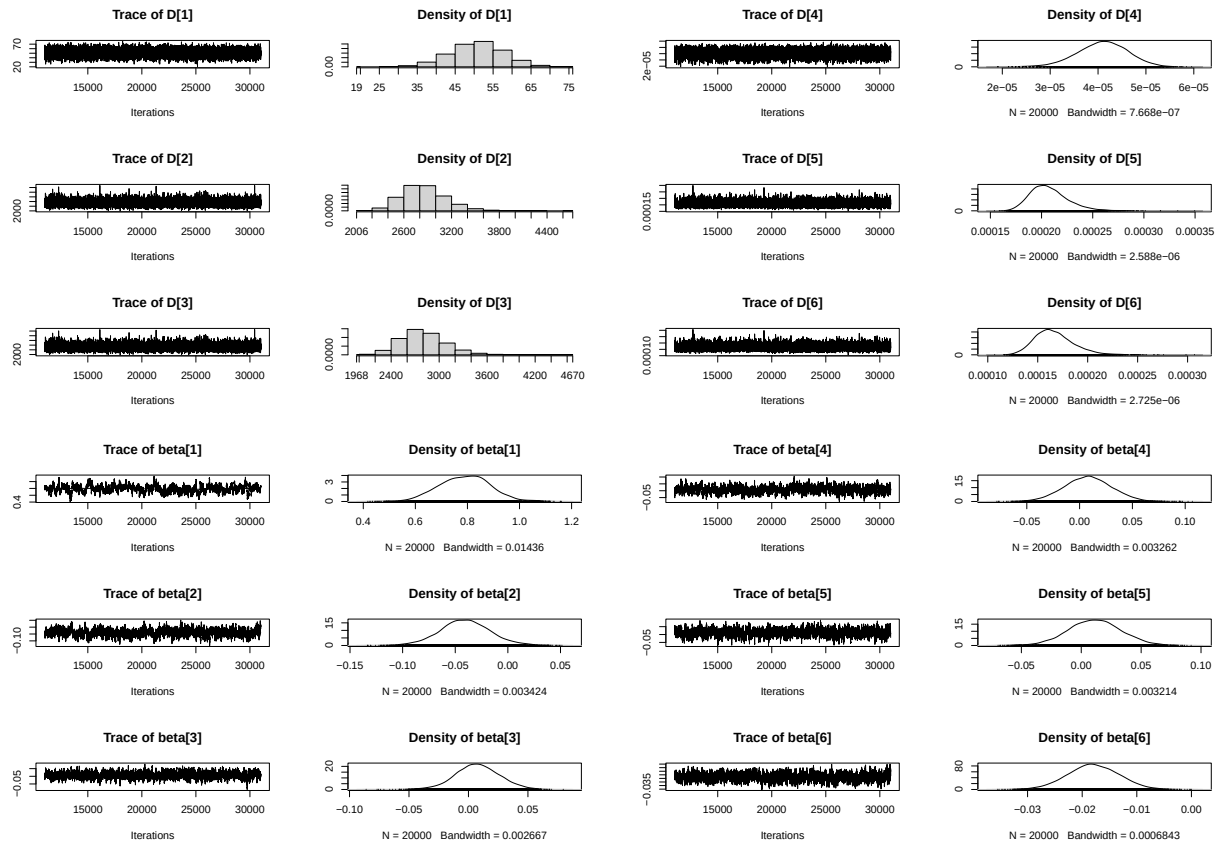



```
print(summary(samps1))
```

```
##
## 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]     5.368e+01 5.612e+00 3.969e-02      5.289e-02
## D[2]     2.803e+03 6.440e+01 4.554e-01      1.119e+00
## D[3]     2.749e+03 6.476e+01 4.579e-01      1.146e+00
## D[4]     4.416e-05 4.016e-06 2.840e-08      6.880e-08
## D[5]     1.913e-04 7.701e-06 5.446e-08      1.458e-07
## D[6]     1.472e-04 8.723e-06 6.168e-08      1.920e-07
## beta[1]  8.392e-01 4.026e-02 2.847e-04      3.719e-03
## beta[2] -4.282e-02 1.013e-02 7.163e-05      5.739e-04
## beta[3]  1.626e-02 6.136e-03 4.339e-05      2.274e-04
## beta[4] -5.416e-03 9.279e-03 6.561e-05      5.149e-04
## beta[5] -1.926e-04 6.214e-03 4.394e-05      2.081e-04
## beta[6] -1.393e-02 1.453e-03 1.027e-05      6.514e-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]     2.683e+03  2.757e+03  2.801e+03  2.846e+03  2.935e+03
## D[3]     2.629e+03  2.704e+03  2.747e+03  2.793e+03  2.882e+03
## D[4]     3.529e-05  4.190e-05  4.475e-05  4.705e-05  5.072e-05
## D[5]     1.778e-04  1.862e-04  1.908e-04  1.959e-04  2.076e-04
## D[6]     1.315e-04  1.413e-04  1.467e-04  1.525e-04  1.660e-04
## beta[1]  7.608e-01  8.126e-01  8.388e-01  8.651e-01  9.219e-01
## beta[2] -6.290e-02 -4.965e-02 -4.273e-02 -3.617e-02 -2.259e-02
## beta[3]  4.112e-03  1.223e-02  1.631e-02  2.033e-02  2.838e-02
## beta[4] -2.342e-02 -1.179e-02 -5.462e-03  7.294e-04  1.325e-02
## beta[5] -1.245e-02 -4.366e-03 -1.418e-04  4.022e-03  1.220e-02
```

```
## beta[6] -1.688e-02 -1.489e-02 -1.390e-02 -1.293e-02 -1.117e-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]     5.122e+01 6.969e+00 4.928e-02      7.789e-02
```

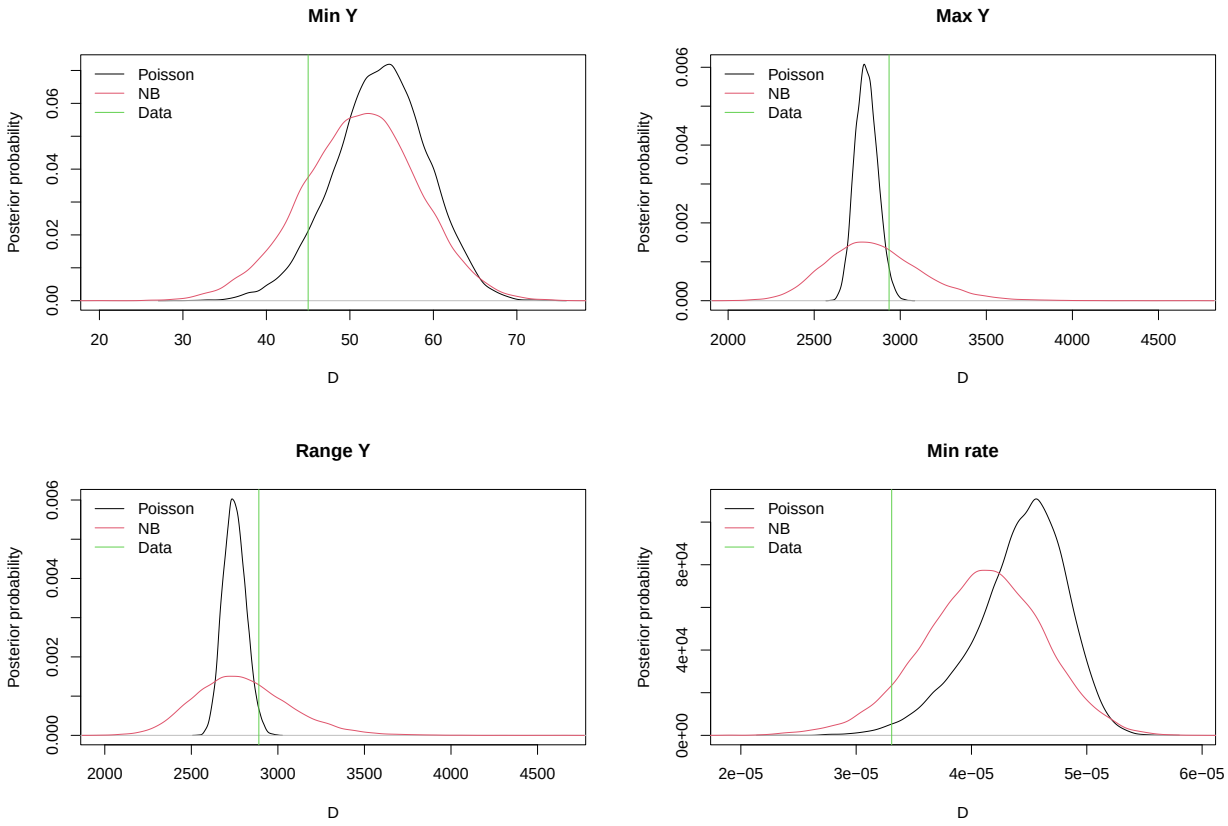
```
## D[2]     2.848e+03 2.775e+02 1.963e+00      3.101e+00
## D[3]     2.797e+03 2.776e+02 1.963e+00      3.105e+00
## D[4]     4.092e-05 5.243e-06 3.707e-08      7.408e-08
## D[5]     2.074e-04 1.854e-05 1.311e-07      3.181e-07
## D[6]     1.665e-04 1.954e-05 1.381e-07      3.880e-07
## beta[1]  7.898e-01 9.818e-02 6.943e-04      7.304e-03
## beta[2] -4.058e-02 2.402e-02 1.698e-04      1.067e-03
## beta[3]  7.554e-03 1.853e-02 1.310e-04      6.331e-04
## beta[4]  7.879e-03 2.243e-02 1.586e-04      9.021e-04
## beta[5]  1.129e-02 2.197e-02 1.554e-04      8.730e-04
## beta[6] -1.797e-02 4.679e-03 3.308e-05      1.851e-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]     2.369e+03  2.656e+03  2.827e+03  3.016e+03  3.452e+03
## D[3]     2.317e+03  2.604e+03  2.776e+03  2.964e+03  3.402e+03
## D[4]     3.014e-05  3.749e-05  4.117e-05  4.459e-05  5.072e-05
## D[5]     1.774e-04  1.943e-04  2.052e-04  2.180e-04  2.497e-04
## D[6]     1.339e-04  1.529e-04  1.643e-04  1.778e-04  2.102e-04
## beta[1]  6.012e-01  7.222e-01  7.917e-01  8.562e-01  9.790e-01
## beta[2] -8.837e-02 -5.629e-02 -4.086e-02 -2.492e-02  8.315e-03
## beta[3] -2.951e-02 -4.699e-03  7.320e-03  1.973e-02  4.441e-02
## beta[4] -3.598e-02 -7.113e-03  7.849e-03  2.278e-02  5.187e-02
## beta[5] -3.178e-02 -3.633e-03  1.122e-02  2.585e-02  5.443e-02
## beta[6] -2.713e-02 -2.109e-02 -1.800e-02 -1.480e-02 -8.837e-03
```

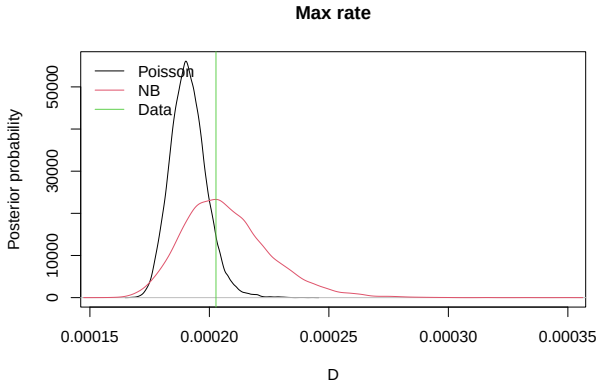
```
D2  <- samps2[[1]]
```

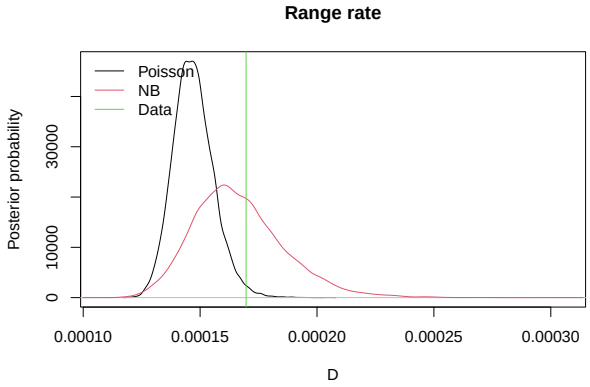
4. Bayesian p-values

```
# Compute rate as the ratio of Y to N
rate <- Y / N

# Create a vector with min, max, and range for Y and rate
D0 <- c(min(Y), max(Y), diff(range(Y)),
        min(rate), max(rate), diff(range(rate)))

# Define names for the vector D0
Dnames <- c("Min Y", "Max Y", "Range Y", "Min rate", "Max rate", "Range rate")

# Initialize vectors for p-values
pval1 <- numeric(6)
names(pval1) <- Dnames
pval2 <- pval1

# Loop through each element to compute test statistics and plot densities
for (j in 1:6) {
  # Plot density for D1
  plot(density(D1[, j]), xlim = range(c(D0[j], D1[, j], D2[, j])),
       xlab = "D", ylab = "Posterior probability", main = Dnames[j])
```

```r
  # Add density line for D2
  lines(density(D2[, j]), col = 2)

  # Add a vertical line for the value in D0
  abline(v = D0[j], col = 3)

  # Add a legend to the plot
  legend("topleft", legend = c("Poisson", "NB", "Data"), lty = 1, col = 1:3, bty = "n")

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








```
# Print p-values for inspection
print(pval1)
```

```
##      Min Y      Max Y    Range Y   Min rate   Max rate Range rate
##    0.92330    0.02415    0.01850    0.98765    0.06820    0.01210
```

```
print(pval2)
```

```
##      Min Y      Max Y    Range Y   Min rate   Max rate Range rate
##     0.7954     0.3448     0.3368     0.9233     0.5563     0.3920
```

Conclusion

In this analysis, two statistical models were evaluated to assess their fit to the data on firearm-related deaths in the USA in 2010. The models compared were a Poisson regression model and a negative binomial regression model.

- **Poisson Regression Model**: The Poisson model assumes that the number of firearm-related deaths follows a Poisson distribution, where the rate parameter λ_i is determined by the state's population and other covariates. The limitation of this model is that it assumes the variance is equal to the mean, which may not be realistic for the data at hand.
- **Negative Binomial Regression Model**: The negative binomial model addresses the overdispersion present in the data by allowing for a variance greater than the mean. This model incorporates an additional over-dispersion parameter m to better fit the data.

Posterior Predictive Checks

Posterior predictive checks were performed using several test statistics, including the minimum and maximum of the observed counts and rates, as well as their ranges. The results of these checks were used to compute Bayesian p-values for both models.

- **Poisson Regression Model**: The Poisson model showed several p-values near zero or one, indicating a poor fit to the data. This suggests that the Poisson model may not adequately capture the variability in the number of firearm-related deaths.

- **Negative Binomial Regression Model**: The negative binomial model provided a better fit to the data, with p-values indicating a more reasonable match between the observed and predicted values. This model is more flexible in accounting for the overdispersion in the data.