Bayesian GLM

Linda Tang

9/6/2021

Question Formulation:

Bayesian Logistic regression: The response is binary (high crash vs. not high crash). The response will be denoted as 1 if it has > 60 crashes per 100,000 residents per year.

Sampling Model: $Y \mid \theta \sim Bern(\theta)$. We chose the logit link function $g(\theta) = log(\frac{\theta}{1-\theta}) = \eta$ and the systematic component $\eta = x^T\beta$. Overall the likelihood of a single observation is :

$$p(Y \mid \theta) = \binom{n}{y}\theta^y(1-\theta)^{n-y} = \binom{n}{y}(\frac{e^\eta}{1+e^\eta})^y(\frac{1}{1+e^\eta})^{n-y}$$

Prior: Since we have little prior belief about which predictor will be significant and the relative magnitude/direction of their influence, we want to chose a weakly informative prior. We will use a t-prior for all coefficients β_k and the intercept β_o.

$$\beta_1 ... \beta_N, \beta_o \sim \text{t}(7, 0)$$

Data Manipulation

I created a binary response variable. I noticed that the population and pct_rural are on quite different scales, so I scaled them and saved it into the original dataframe.

```
bike <- bike %>%
  mutate(high_crashes = ifelse(crashes*100000/pop > 60, 1, 0)) %>%
  mutate(pop = scale(pop)) %>%
  mutate(pct_rural = scale(pct_rural))
```

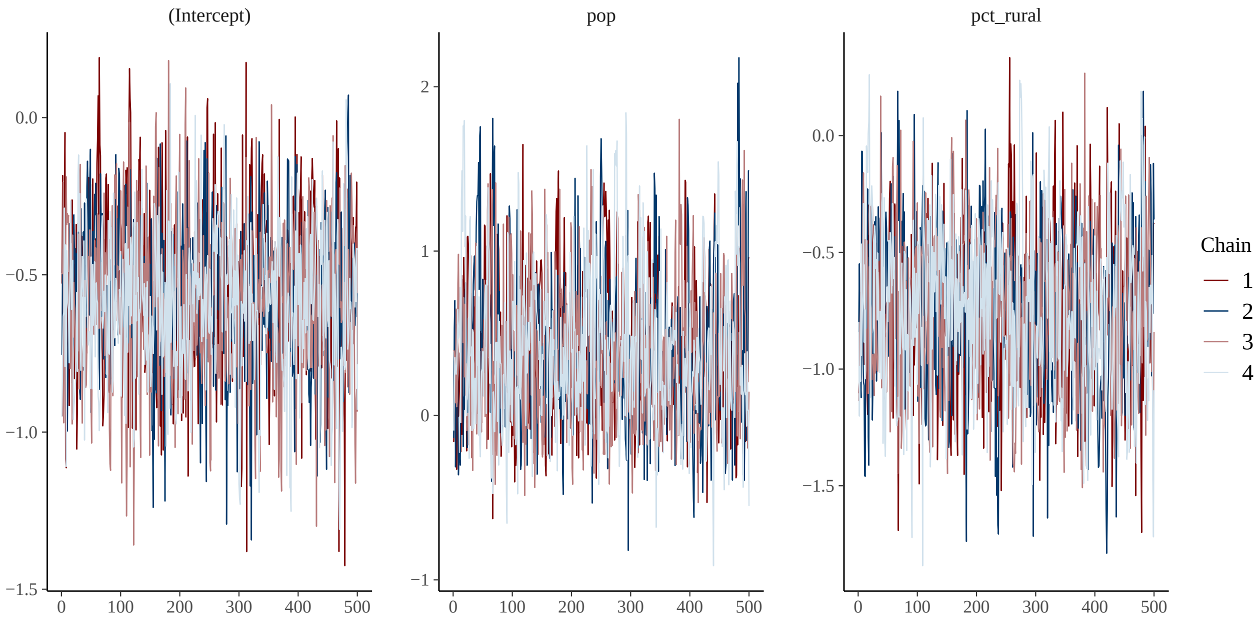
Fitting the model

```
model <- stan_glm(high_crashes ~ pop + pct_rural, # only used these 2 predictors
                  data = bike,
                  family = binomial(link = "logit"),
                  prior = student_t(df = 7, 0),
                  prior_intercept = student_t(df = 7, 0),
                  cores = 4,
                  chains = 4, # run 4 MCMC chains
                  iter = 1000,
                  seed = 12345)
```

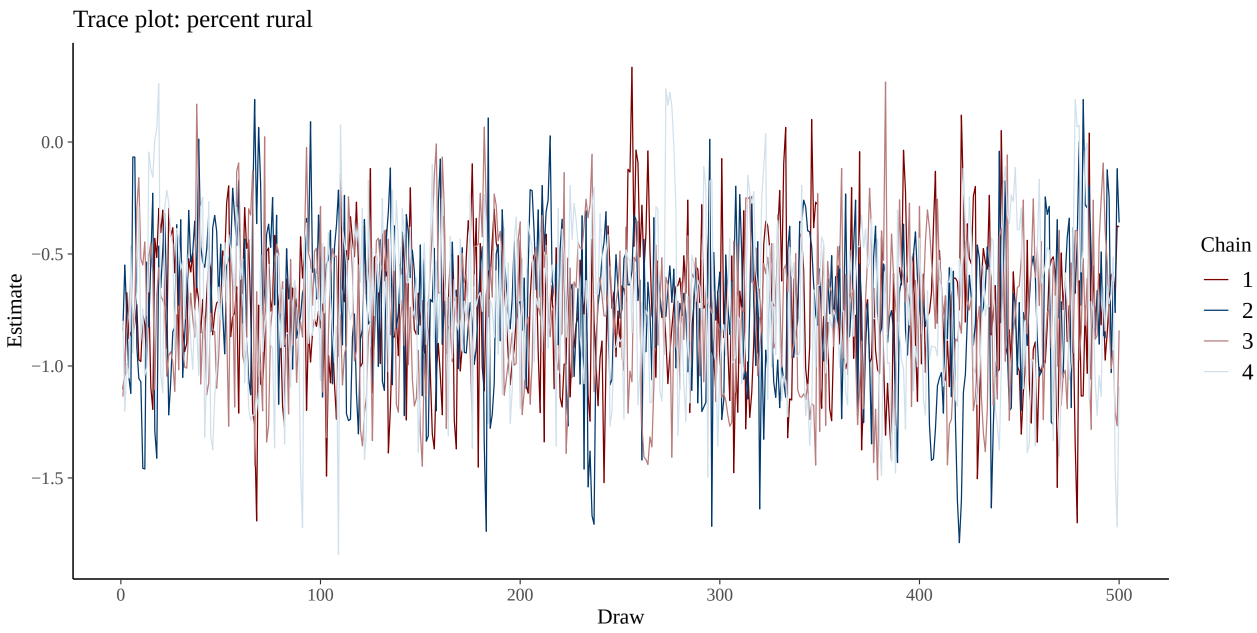
Model Diagnoistics

Some model diagnostic plots are shown below:

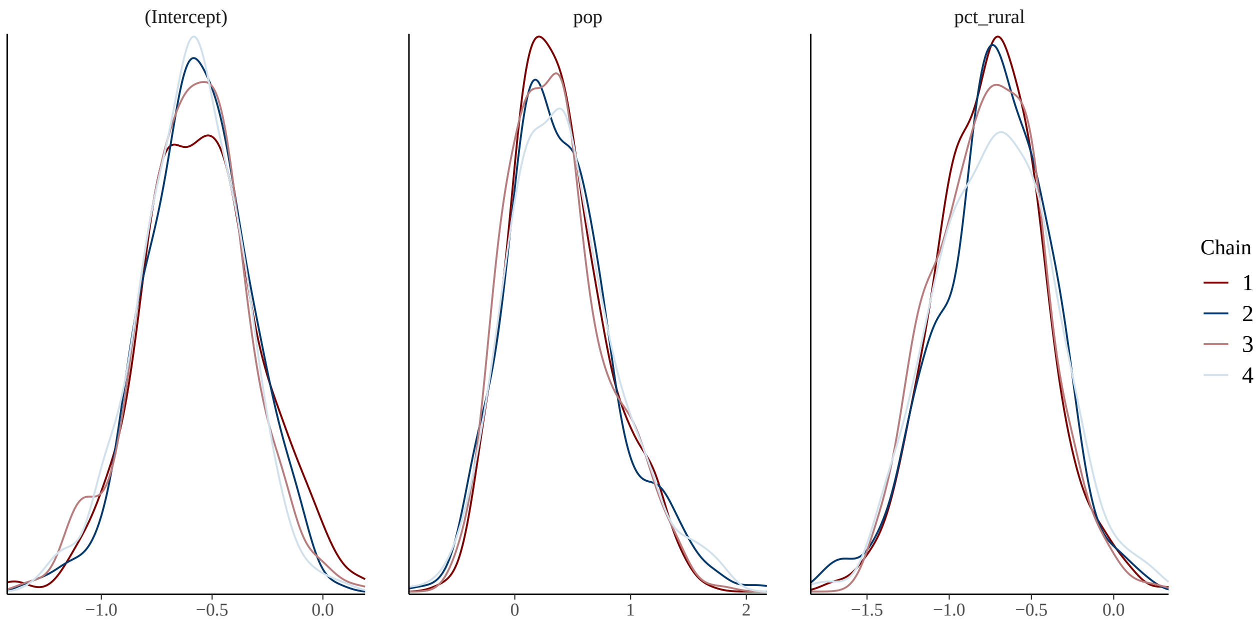
```r
# all chains converged & mixed well
color_scheme_set("mix-blue-red")
plot(model, "trace")
```



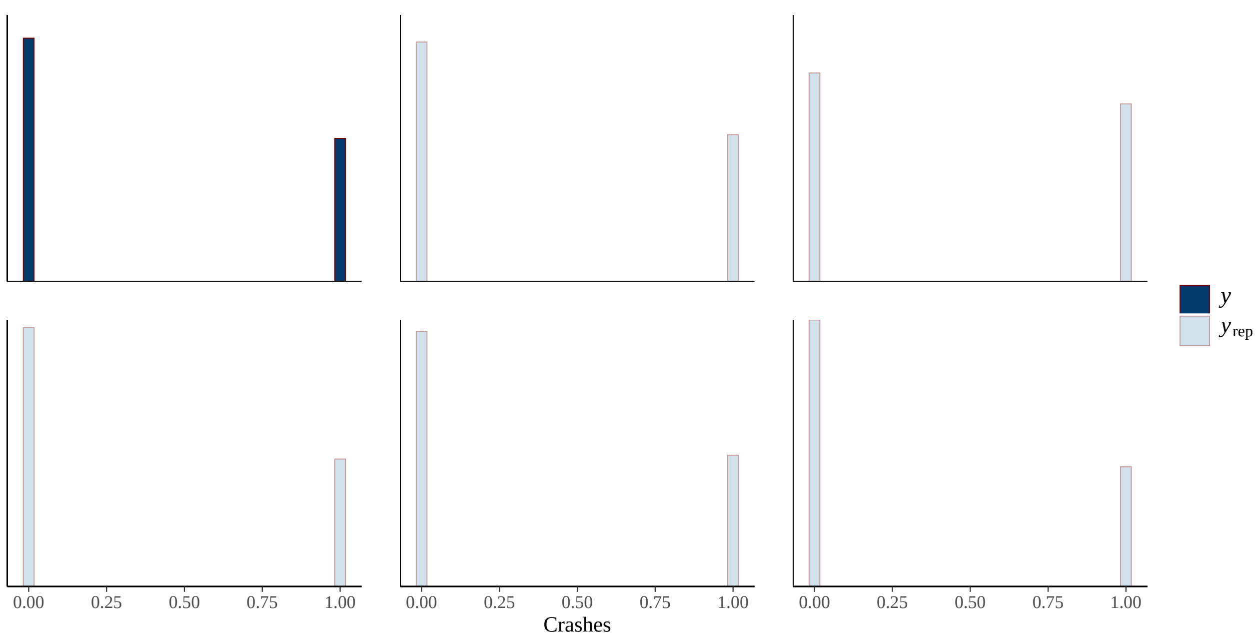

```r
# all chains converged & mixed well
plot(model, "trace", pars = "pct_rural", ylab = "Asdf") +
  labs(title = "Trace plot: percent rural",
       y = "Estimate", x = "Draw")
```



```r
# density from the 4 chain seems to overlap well
# suggest convergence and good mixing
plot(model, "dens_overlay")
```

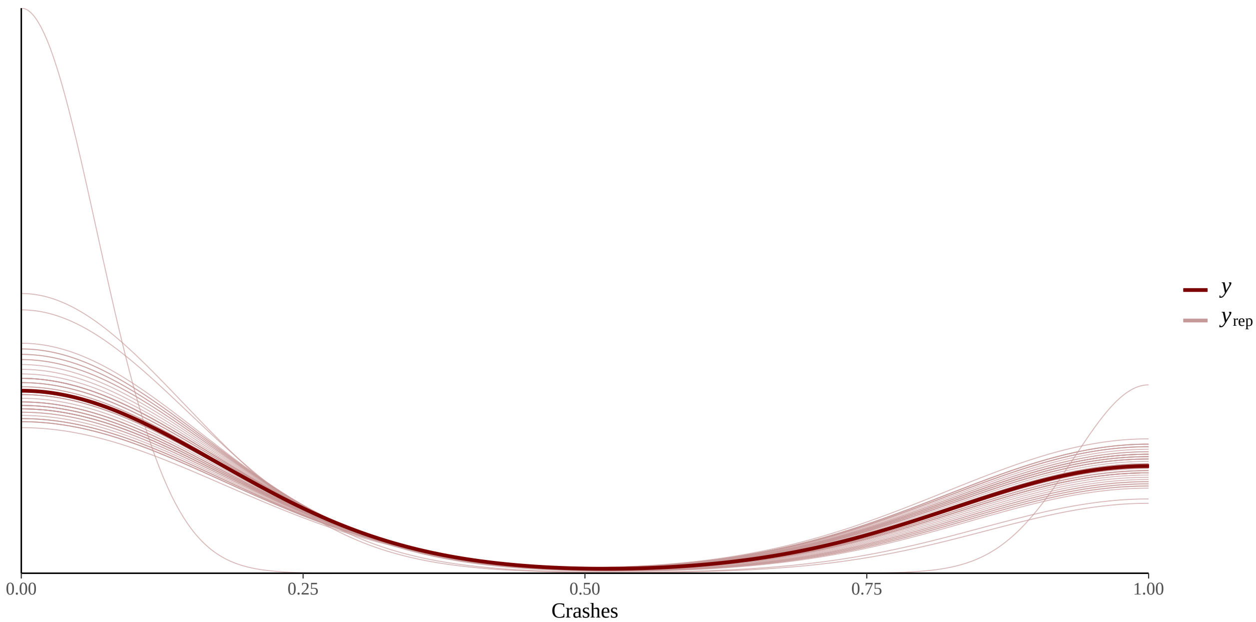



```r
# posterior predictive checks
# draws from the posterior predictive dist. seems to be similar to actual data
pp_check(model, plotfun = "hist", nreps = 5) +
  xlab("Crashes")
```

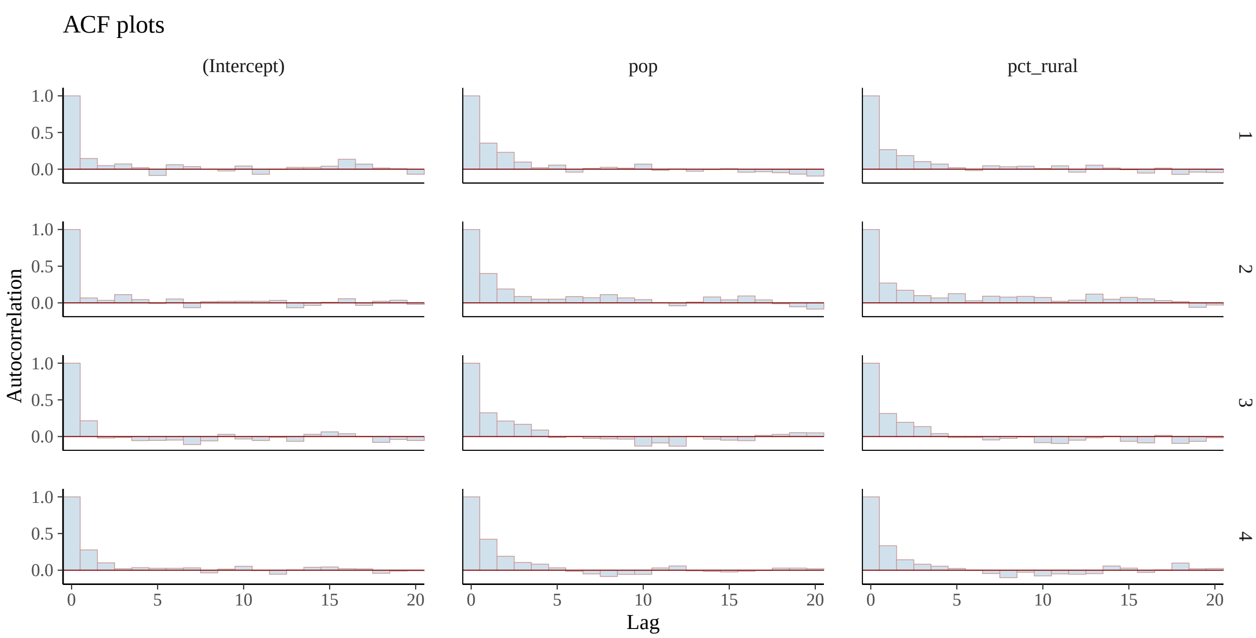



```r
# posterior predictive checks
# draws from the posterior predictive dist. seems to be similar to actual data
# although there's one outlier?
```

```r
pp_check(model) +
  xlab("Crashes")
```




```r
# relatively low autocorrelation
plot(model, "acf_bar") +
  labs(title = "ACF plots")
```

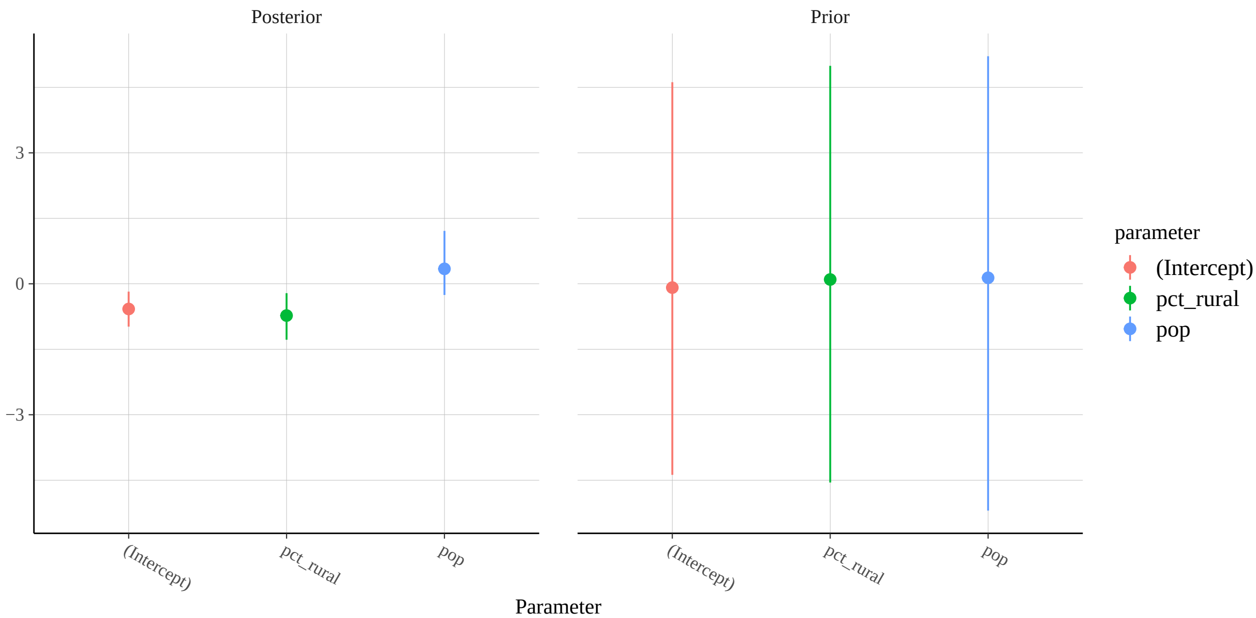



```r
# Compare prior and posterior dist.
# Seems like the posterior distribution is much narrower
prior_summary(model)
```

```
## Priors for model 'model'
```

```
## ------
## Intercept (after predictors centered)
##  ~ student_t(df = 7, location = 0, scale = 2.5)
##
## Coefficients
##  ~ student_t(df = [7,7], location = [0,0], scale = [2.5,2.5])
## ------
## See help('prior_summary.stanreg') for more details
```

```r
posterior_vs_prior(model)
```



Inference

Output of model coefficients:

```r
summary(model)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       binomial [logit]
##  formula:      high_crashes ~ pop + pct_rural
##  algorithm:    sampling
##  sample:       2000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 100
##  predictors:   3
##
## Estimates:
##               mean   sd   10%   50%   90%
## (Intercept) -0.6    0.2 -0.9   -0.6  -0.3
## pop          0.4    0.4 -0.1    0.3   1.0
```

```
## pct_rural   -0.7    0.3 -1.2  -0.7  -0.3
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 0.4    0.1  0.3   0.4   0.4
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##               mcse Rhat n_eff
## (Intercept)   0.0  1.0  1281
## pop           0.0  1.0   762
## pct_rural     0.0  1.0   834
## mean_PPD      0.0  1.0  1696
## log-posterior 0.0  1.0   718
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

A more precise output of model coefficients:

```
round(posterior_interval(model, prob = 0.95), 3)
```

```
##               2.5%  97.5%
## (Intercept) -1.084 -0.097
## pop         -0.338  1.379
## pct_rural   -1.387 -0.093
```

Based on the model output above, the 95% confidence interval of the coefficient of pct_rural is from -1.387 to - 0.093, which doesn't cross 0. So after accounting for population, pct_rural is still important in predicting whether a county is high crash.

Interpretation: for one standard deviation increase in pct_rural (which is around 28%), the odds of a county being high crash by multiply be a factor of 0.4965853, holding population constant.

(*Note: Initially I build the modeling without scaling the variables and I found pct_rural to be not significant after accounting for population. However, after I scaled the data, pct_rural is significant).