lab2

Lillian Clark

9/6/2021

(Spent most of lab trying to install `rstanarm`)

```
bike <- read_csv("210830_bikecrash.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   county = col_character(),
##   pop = col_double(),
##   med_hh_income = col_double(),
##   traffic_vol = col_double(),
##   pct_rural = col_double(),
##   crashes = col_double()
## )
```

```
bike_new <- bike %>%
  mutate(crashes_pc = crashes / (pop / 100000),
         high_crash_cty = case_when(
           crashes_pc >= 60 ~ 1,
           TRUE ~ 0),
         high_crash_cty = as.factor(high_crash_cty))
```

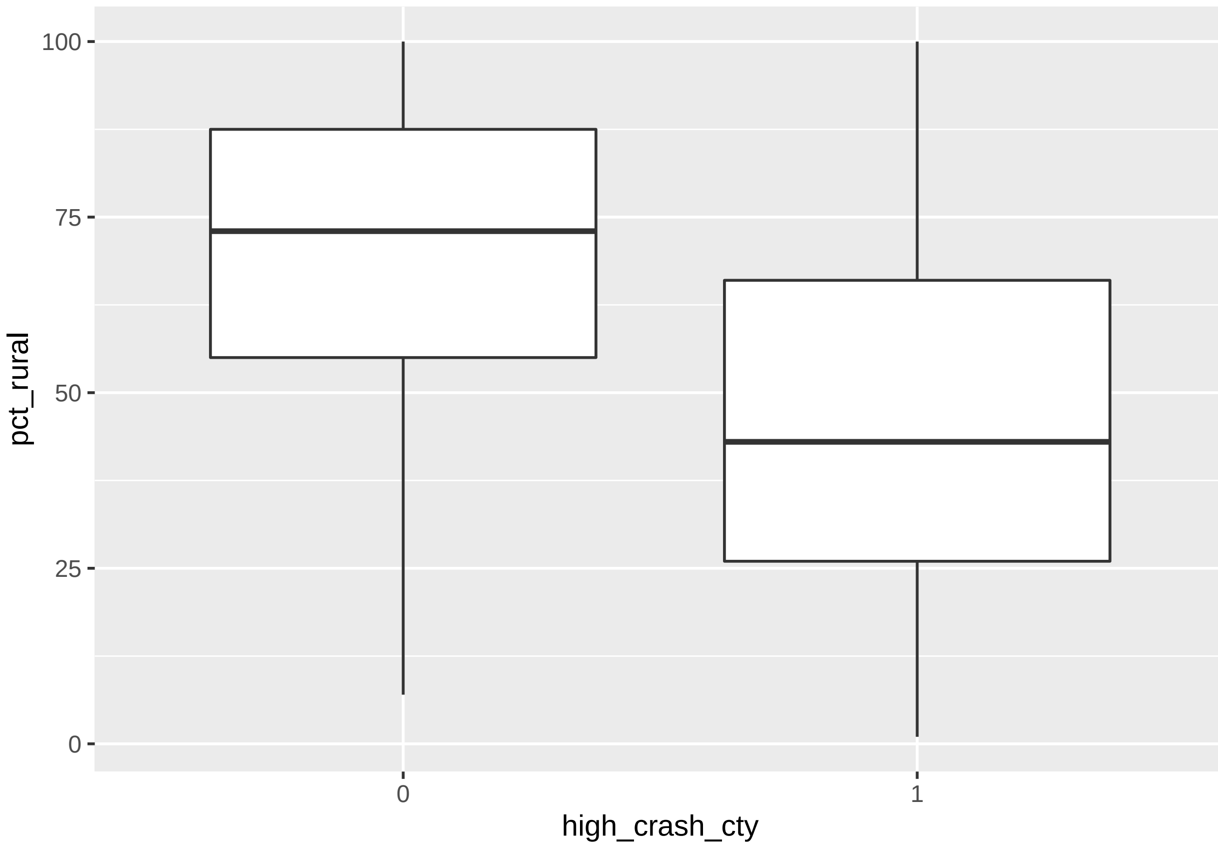
```
summary(bike_new$high_crash_cty)
```

```
##  0  1
## 63 37
```

```
summary(bike_new$pct_rural)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   42.50   62.50   61.24   85.00  100.00
```

```
ggplot(bike_new, aes(x = high_crash_cty, y = pct_rural)) +
  geom_boxplot()
```




```
m0 <- glm(high_crash_cty ~ I(pop/1000000) + pct_rural, data = bike_new, family = "binomial")
round(summary(m0)$coef, 6)
```

```
##                 Estimate Std. Error   z value Pr(>|z|)
## (Intercept)     0.994662   0.846443  1.175108 0.239951
## I(pop/1e+06)    1.162455   2.264275  0.513389 0.607679
## pct_rural      -0.028063   0.011431 -2.455100 0.014085
```

We use the estimate and standard error from a frequentist GLM as our prior for the intercept in our Bayesian regression.

```
m1 <- stan_glm(high_crash_cty ~ I(pop/1000000) + pct_rural,
                         data = bike_new,
                         family = binomial(link = "logit"),
                         prior_intercept = normal(.995, .846),
                         prior = normal(0, 100, autoscale = T),
                         chains = 2, iter = 10000, seed = 9763,
                         prior_PD = F)
```

```
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000132 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.32 seconds.
## Chain 1: Adjust your expectations accordingly!
```

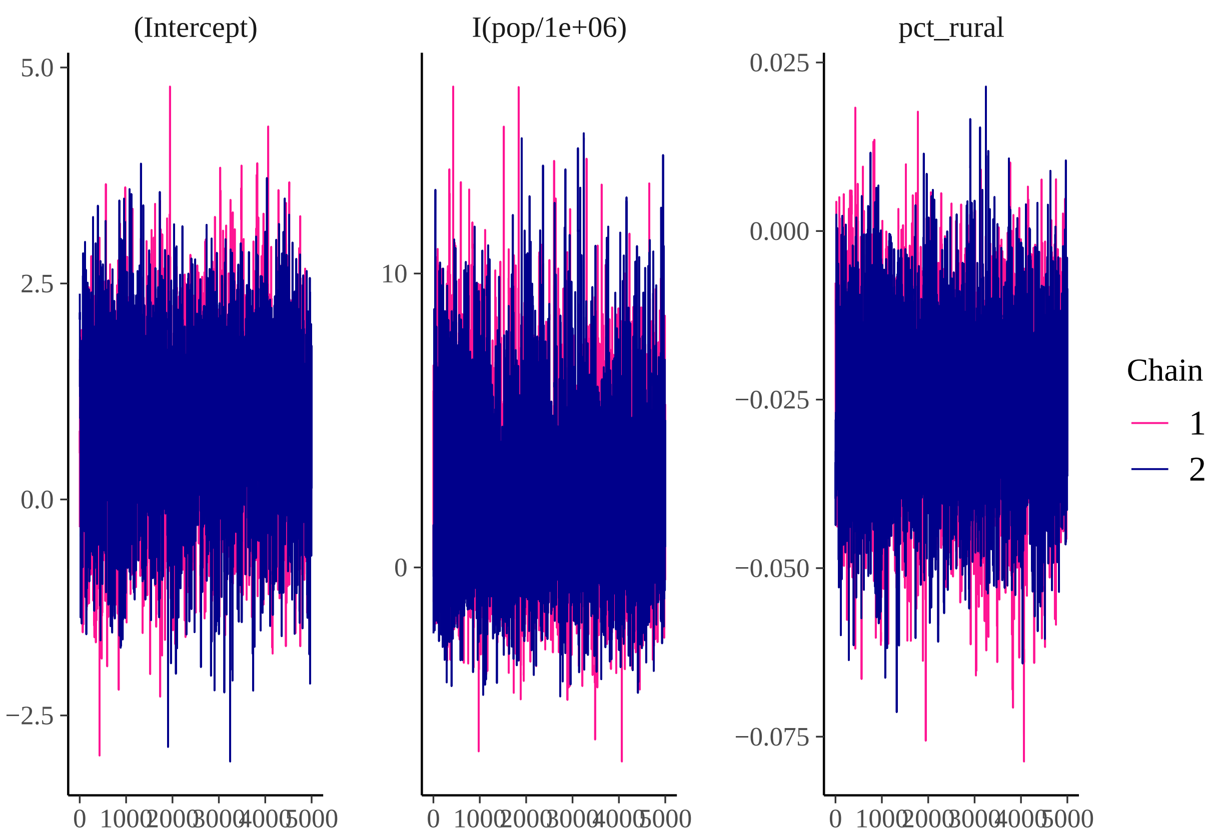
```
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 1: Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 1: Iteration: 2000 / 10000 [ 20%]  (Warmup)
## Chain 1: Iteration: 3000 / 10000 [ 30%]  (Warmup)
## Chain 1: Iteration: 4000 / 10000 [ 40%]  (Warmup)
## Chain 1: Iteration: 5000 / 10000 [ 50%]  (Warmup)
## Chain 1: Iteration: 5001 / 10000 [ 50%]  (Sampling)
## Chain 1: Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 1: Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 1: Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 1: Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 1: Iteration: 10000 / 10000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.399126 seconds (Warm-up)
## Chain 1:                0.491079 seconds (Sampling)
## Chain 1:                0.890205 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2.3e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.23 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 2: Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 2: Iteration: 2000 / 10000 [ 20%]  (Warmup)
## Chain 2: Iteration: 3000 / 10000 [ 30%]  (Warmup)
## Chain 2: Iteration: 4000 / 10000 [ 40%]  (Warmup)
## Chain 2: Iteration: 5000 / 10000 [ 50%]  (Warmup)
## Chain 2: Iteration: 5001 / 10000 [ 50%]  (Sampling)
## Chain 2: Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 2: Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 2: Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 2: Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 2: Iteration: 10000 / 10000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.409604 seconds (Warm-up)
## Chain 2:                0.37632 seconds (Sampling)
## Chain 2:                0.785924 seconds (Total)
## Chain 2:
```

```r
prior_summary(m1)
```

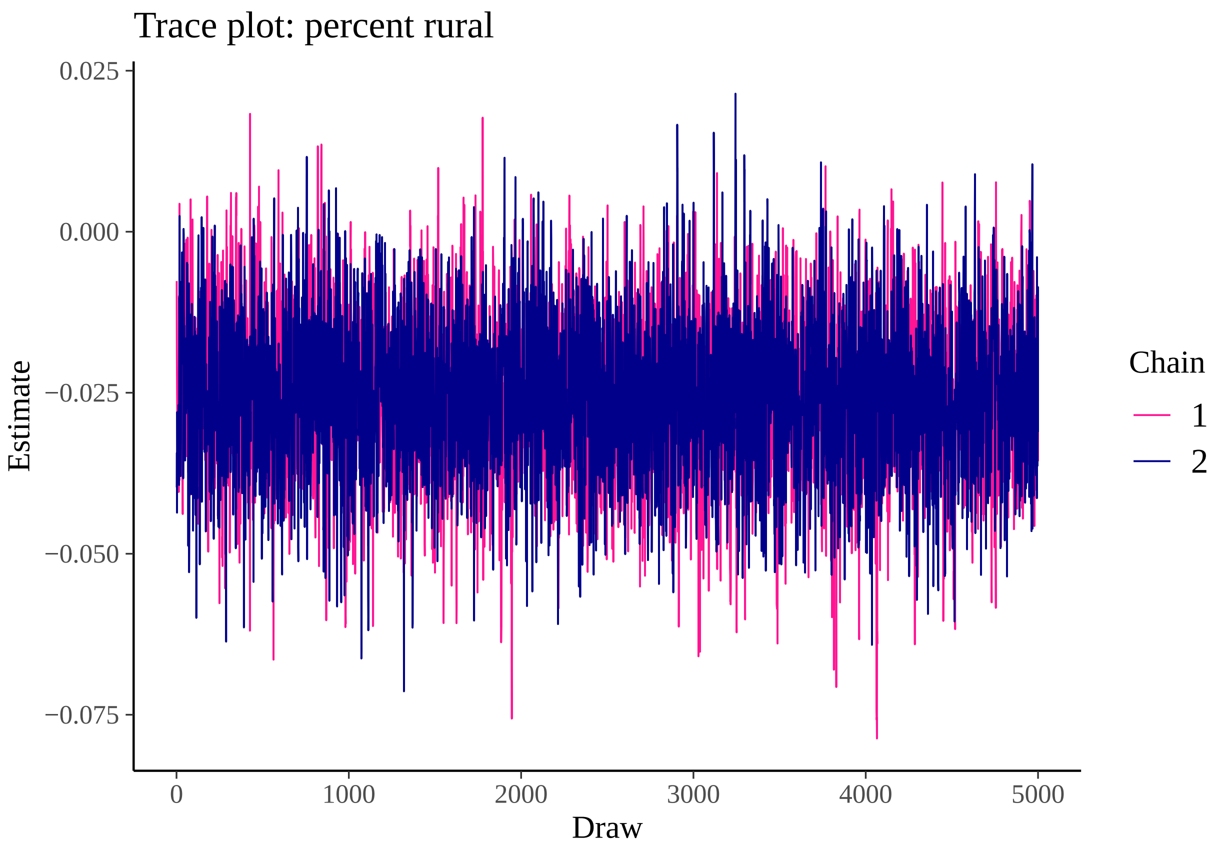
```
## Priors for model 'm1'
## ------
## Intercept (after predictors centered)
##  ~ normal(location = 0.99, scale = 0.85)
##
## Coefficients
##   Specified prior:
```

```
##     ~ normal(location = [0,0], scale = [100,100])
##   Adjusted prior:
##     ~ normal(location = [0,0], scale = [597.13,  3.55])
## ------
## See help('prior_summary.stanreg') for more details
```

```r
color_scheme_set(c("darkblue", "darkred", "darkgray",
                   "deepskyblue", "deeppink", "darkgreen"))
plot(m1, "trace")
```

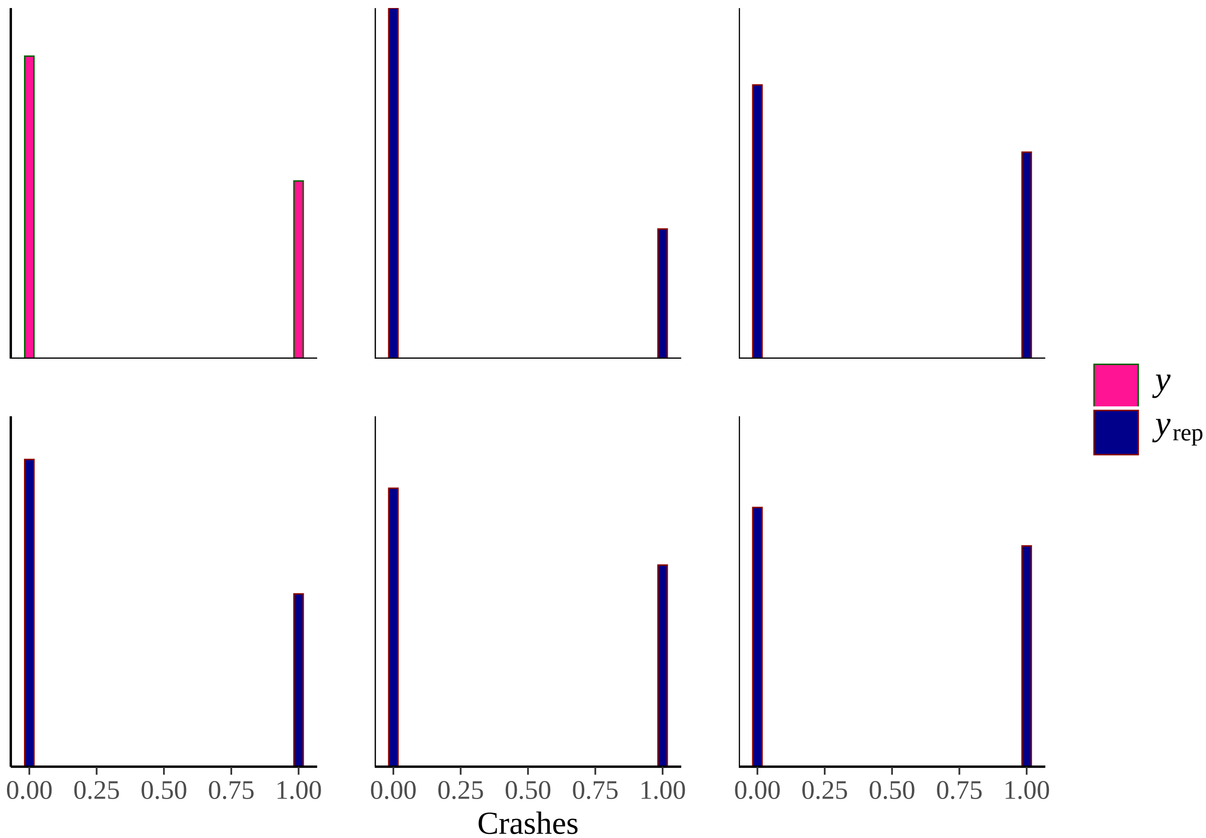



```r
plot(m1, "trace", pars = "pct_rural") +
  labs(title = "Trace plot: percent rural",
       y = "Estimate", x = "Draw")
```

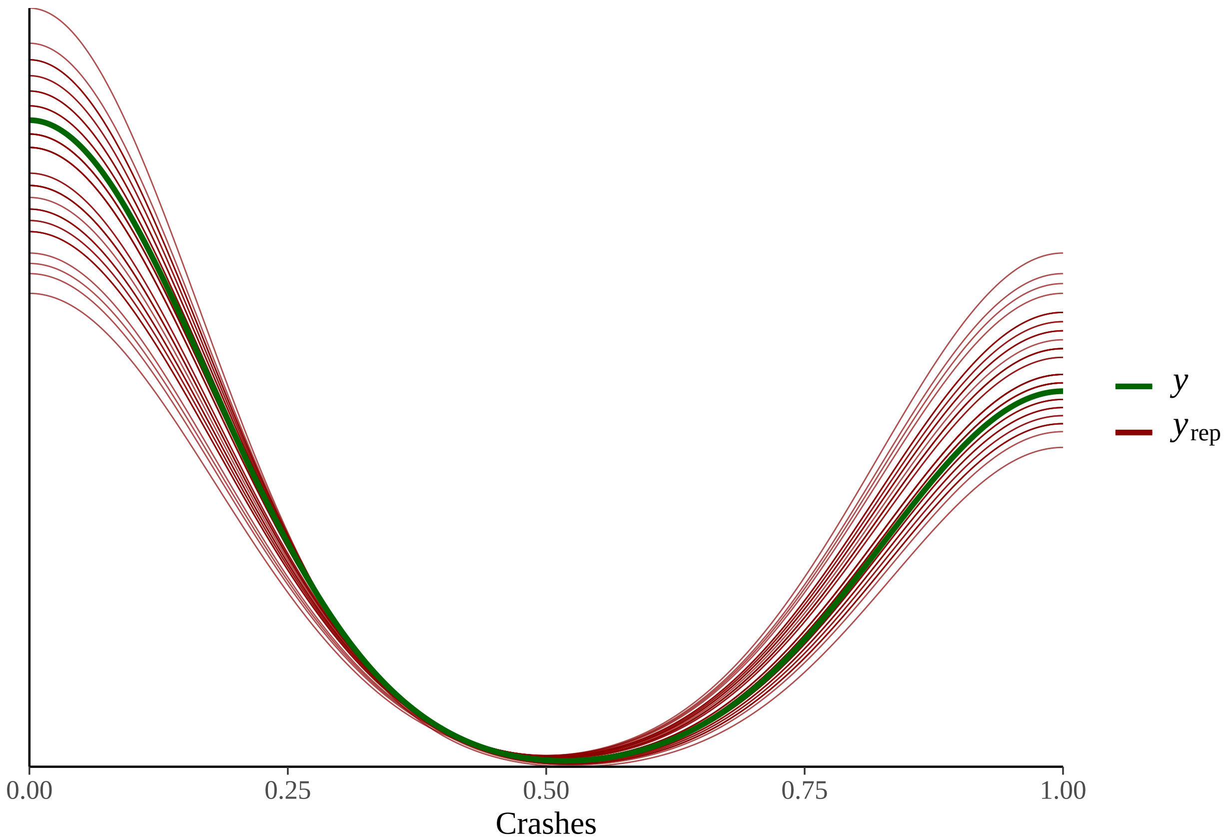



```r
pp_check(m1, plotfun = "hist", nreps = 5) +
  xlab("Crashes")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

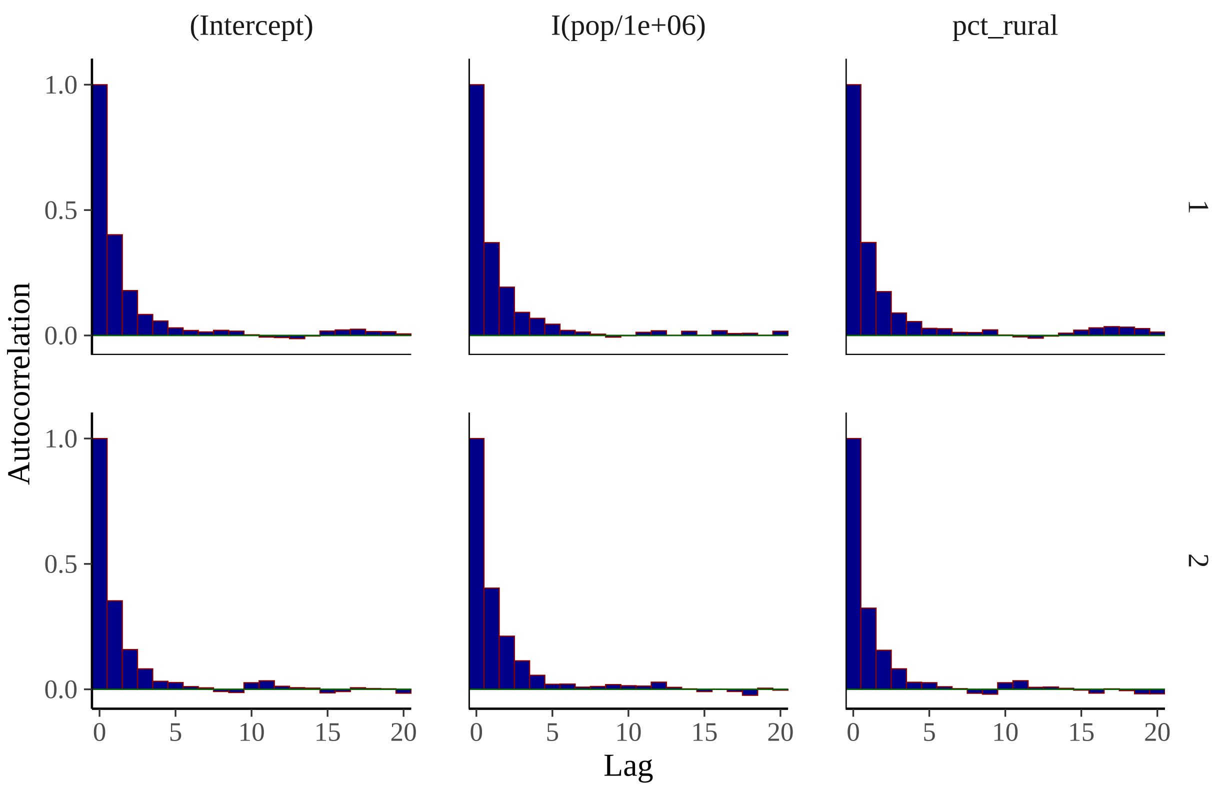



```
pp_check(m1) +
  xlab("Crashes")
```

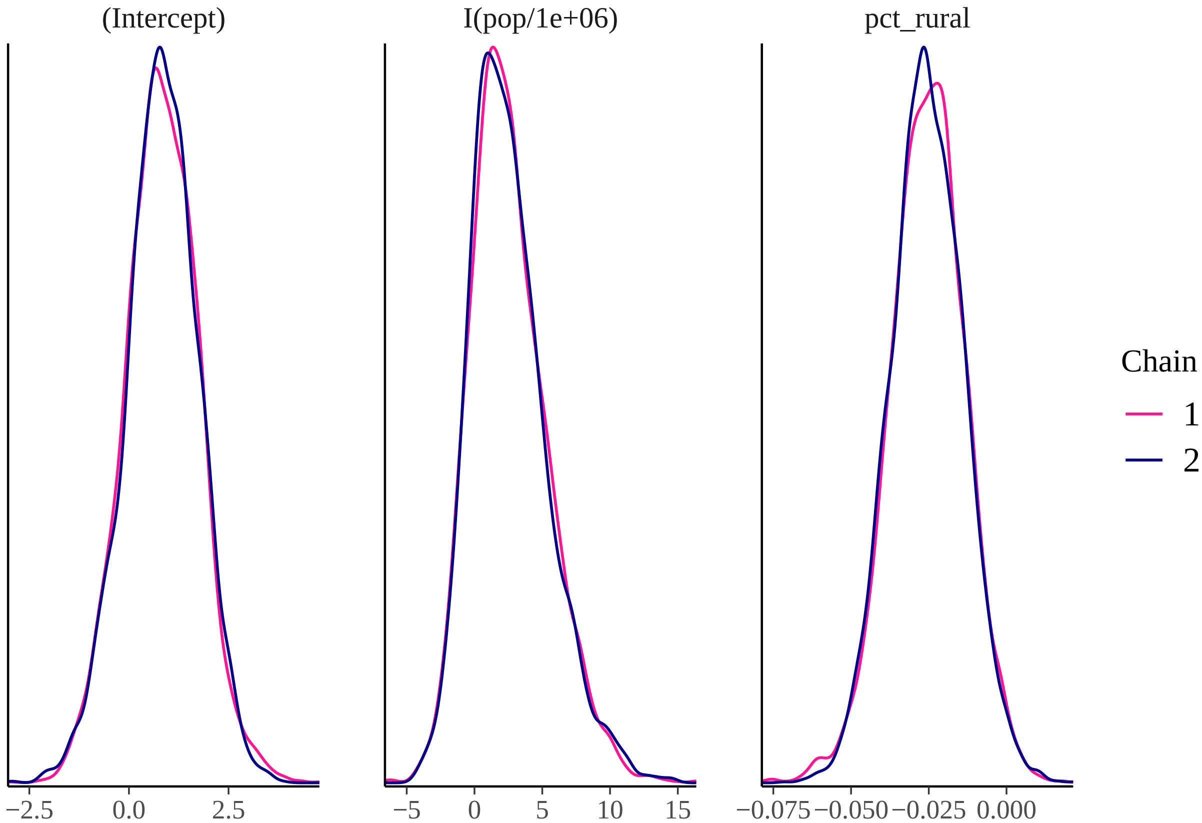



```r
plot(m1, "acf_bar") +          # compare to "acf"
  labs(title = "ACF plots")
```

ACF plots




```r
plot(m1, "dens_overlay")
```




```r
round(as.data.frame(summary(m1)), 2)
```

```
##                  mean mcse   sd    10%    50%    90% n_eff Rhat
## (Intercept)      0.82 0.01 0.91  -0.36   0.83   1.95  4022    1
## I(pop/1e+06)     2.56 0.05 2.84  -0.73   2.22   6.38  3652    1
## pct_rural       -0.03 0.00 0.01  -0.04  -0.03  -0.01  4185    1
## mean_PPD         0.39 0.00 0.06   0.31   0.39   0.47  7221    1
## log-posterior -63.47 0.02 1.31 -65.15 -63.12 -62.20  3372    1
```

```r
bike_new %>%
  filter(county == "Durham")
```

```
## # A tibble: 1 x 8
##   county    pop med_hh_income traffic_vol pct_rural crashes crashes_pc
##   <chr>   <dbl>         <dbl>       <dbl>     <dbl>   <dbl>      <dbl>
## 1 Durham 316739          59.3         395         6     340       107.
## # ... with 1 more variable: high_crash_cty <fct>
```

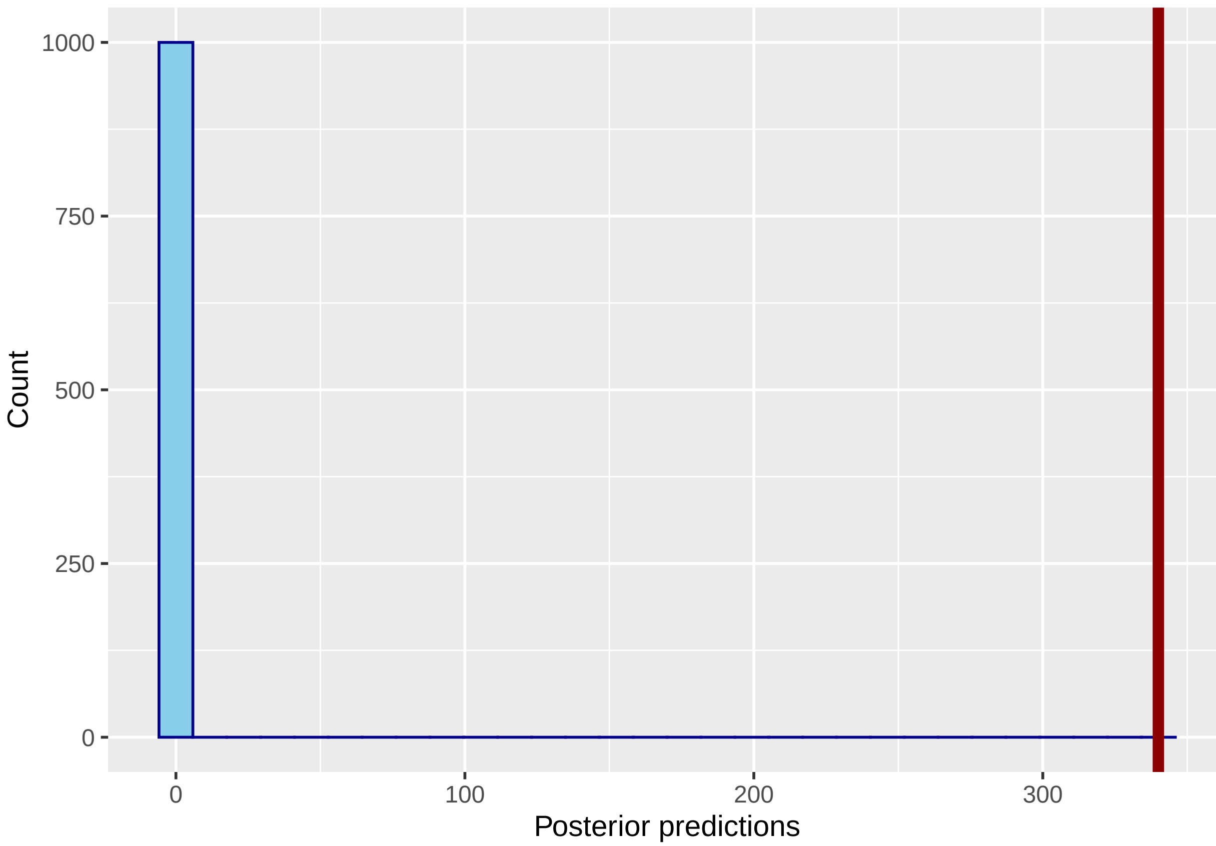
```r
durham <- posterior_predict(m1,
                            newdata = data.frame(pop = 316739,
                                                 pct_rural = 6),
                            draws = 1000)
head(durham)
```

```
##      1
```

```
## [1,] 1
## [2,] 1
## [3,] 0
## [4,] 1
## [5,] 1
## [6,] 1
```

```r
ggplot(as.data.frame(durham), aes(x = durham)) +
  geom_histogram(color = "darkblue", fill = "skyblue") +
  labs(x = "Posterior predictions", y = "Count") +
  geom_vline(xintercept = 340, color = "darkred", lwd = 2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Is there evidence of an association between rurality and being a high-crash county, after adjusting for population? Explain, and quantify any variability in your estimates. Be sure to evaluate model convergence.

asks the beta coefficient for pct_rural variability in beta_1