Generalized Linear Models with Stan

Ben Goodrich

April 13, 2022

What about the States?

- Suppose we wanted to include an intercept for each state, rather than merely an indicator for whether the state has a Republican governor
- We could include 50 dummy variables in $\mathbf{X}$ and specify priors on those coefficients, but McElreath prefers the following approach

```
X <- as.matrix(Gabba$Trump - mean(Gabba$Trump))
group <- as.factor(Gabba$State)
nlevels(group) # size N but only 51 unique values
```

```
## [1] 51
```

- We can also utilize normal priors if we prefer with means and standard deviations as

```
m     <- c(beta = -0.5, alpha = 50, sigma = 10)
scale <- c(beta = 0.25, alpha = 10, sigma = 3)
```

```
data {               // saved as "groups.stan"
  int<lower = 0> N; // number of observations
  int<lower = 0> K; // number of predictors
  matrix[N, K] X;   // matrix of predictors
  vector[N] y;      // outcomes
  int<lower = 1> J; // number of groups
  int<lower = 1, upper = J> group[N];     // group membership
  int<lower = 0, upper = 1> prior_only;   // ignore data?
  vector[K + 2] m;                        // prior means
  vector<lower = 0>[K + 2] scale;         // prior scales
}
parameters {
  vector[K] beta;
  vector[J] alpha;
  real<lower = 0> sigma;
}
model {
  if (!prior_only) target += normal_id_glm_lpdf(y | X, alpha[group], beta, sigma);
  target += normal_lpdf(beta  | m[1:K],   scale[1:K]); // ^ important
  target += normal_lpdf(alpha | m[K + 1], scale[K + 1]);
  target += normal_lpdf(sigma | m[K + 2], scale[K + 2]); // actually half normal
}
generated quantities {
  vector[N] log_lik;
  {
    vector[N] mu = alpha[group] + X * beta;
```

Calling **stan** for the grouped model

```r
states <- stan("groups.stan", data = list(N = nrow(Gabba), K = ncol(X), y = Gabba$Vaccinated,
                                          X = X, J = nlevels(group), group = as.integer(group)
                                          prior_only = 0, m = m, scale = scale)) # ^ important

states # only 6 states could fit on the screen but all 51 intercepts were estimated
```

```
## Inference for Stan model: groups.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                   mean se_mean   sd      2.5%       25%       50%       75%     97.5%
## beta[1]          -0.46    0.00 0.01     -0.48     -0.47     -0.46     -0.46     -0.44
## alpha[1]         42.90    0.01 0.89     41.16     42.29     42.89     43.51     44.60
## alpha[2]         55.17    0.02 1.45     52.28     54.21     55.21     56.14     58.03
## alpha[3]         57.23    0.03 2.08     53.22     55.80     57.20     58.68     61.27
## alpha[4]         48.17    0.01 0.86     46.51     47.60     48.17     48.75     49.83
## alpha[5]         51.37    0.01 0.97     49.47     50.73     51.35     52.02     53.27
## alpha[6]         51.56    0.01 0.93     49.76     50.92     51.55     52.21     53.39
## alpha[7]         62.80    0.03 2.57     57.79     61.07     62.80     64.56     67.69
## alpha[8]         52.52    0.05 4.01     44.96     49.74     52.57     55.33     60.33
## alpha[9]         48.63    0.08 5.93     37.15     44.65     48.60     52.63     60.45
## alpha[10]        53.12    0.01 0.94     51.22     52.48     53.13     53.76     54.93
## alpha[11]        42.96    0.01 0.59     41.78     42.55     42.95     43.35     44.11
```

Utility Function for Predictions of Future Data

- For Bayesians, the log predictive PDF is the most appropriate utility function
- Choose the model that maximizes the expectation of this over FUTURE data

$$\mathrm{ELPD} = \mathbb{E}_Y \ln f\left(y_{N+1}, \dots, y_{2N} \mid y_1, \dots, y_N\right) =$$
$$\int \ln f\left(y_{N+1}, \dots, y_{2N} \mid \mathbf{y}\right) f\left(y_{N+1}, \dots, y_{2N} \mid \mathbf{y}\right) dy_{N+1} \dots dy_{2N} \approx$$
$$\sum_{n=1}^{N} \ln f\left(y_n \mid \mathbf{y}_{-n}\right) = \sum_{n=1}^{N} \ln \int_{\Theta} f\left(y_n \mid \boldsymbol{\theta}\right) f\left(\boldsymbol{\theta} \mid \mathbf{y}_{-n}\right) d\theta_1 d\theta_2 \dots d\theta_K$$

- $f\left(y_n \mid \boldsymbol{\theta}\right)$ is just the n-th likelihood contribution, but can we somehow obtain $f\left(\boldsymbol{\theta} \mid \mathbf{y}_{-n}\right)$ from $f\left(\boldsymbol{\theta} \mid \mathbf{y}\right)$?
- Yes, assuming y_n does not have an outsized influence on the posterior

Optional `generated quantities` Block

- Can declare more endogenous knowns, assign to them, and use them
- Samples are stored
- Can reference anything except stuff in the `model` block
- Can also do this in R afterward, but primarily used for
    - Interesting functions of posterior that don’t involve likelihood
    - Posterior predictive distributions and / or functions thereof
    - The log-likelihood for each observation to pass to `loo`

PSISLOOCV

```
generated quantities { // part of groups.stan
  vector[N] log_lik;
  {
    vector[N] mu = alpha[group] + X * beta;
    for (n in 1:N) log_lik[n] = normal_lpdf(y[n] | mu[n], sigma);
  }
}
```

```
loo(states)
```

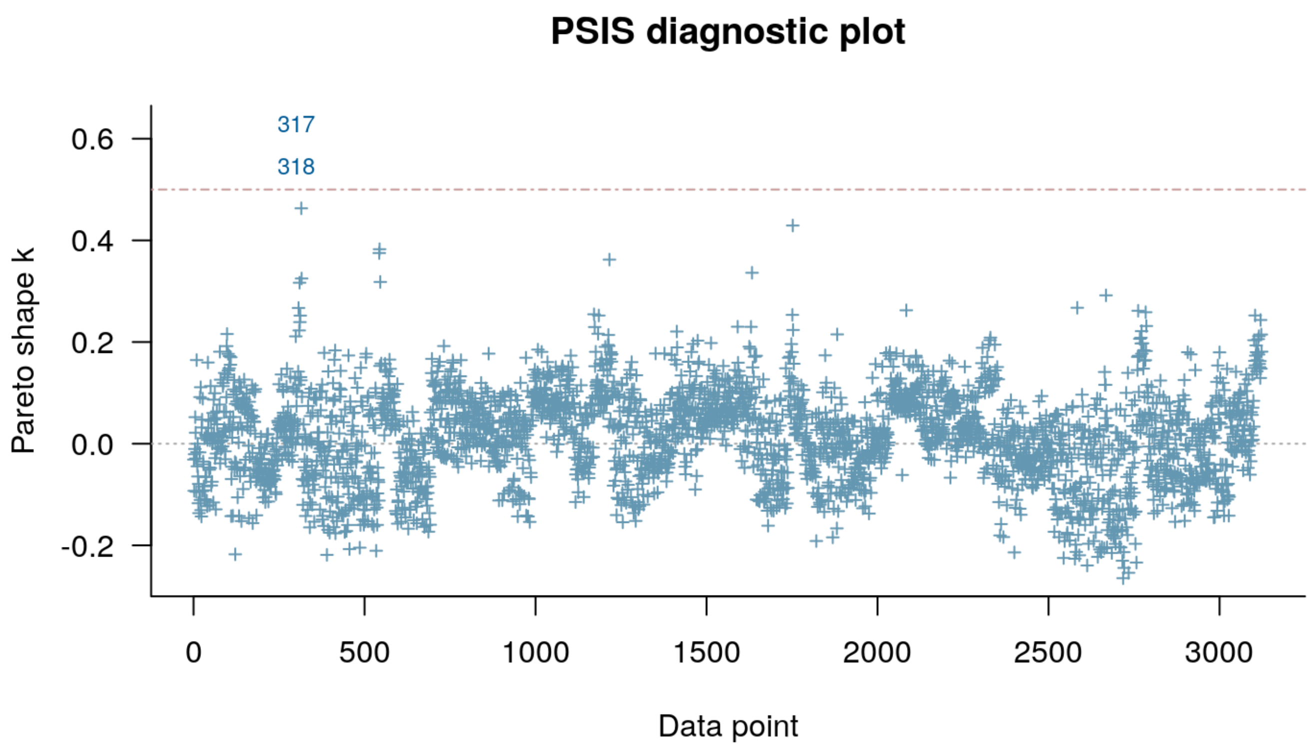
```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic')
```

```
##
## Computed from 4000 by 3125 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo -10788.1  76.7
## p_loo        51.5   2.8
## looic     21576.1 153.4
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
```

Leverage Diagnostic Plot

```
plot(loo(states), label_points = TRUE) # not too bad, 318 is D.C.
```

```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic')
```

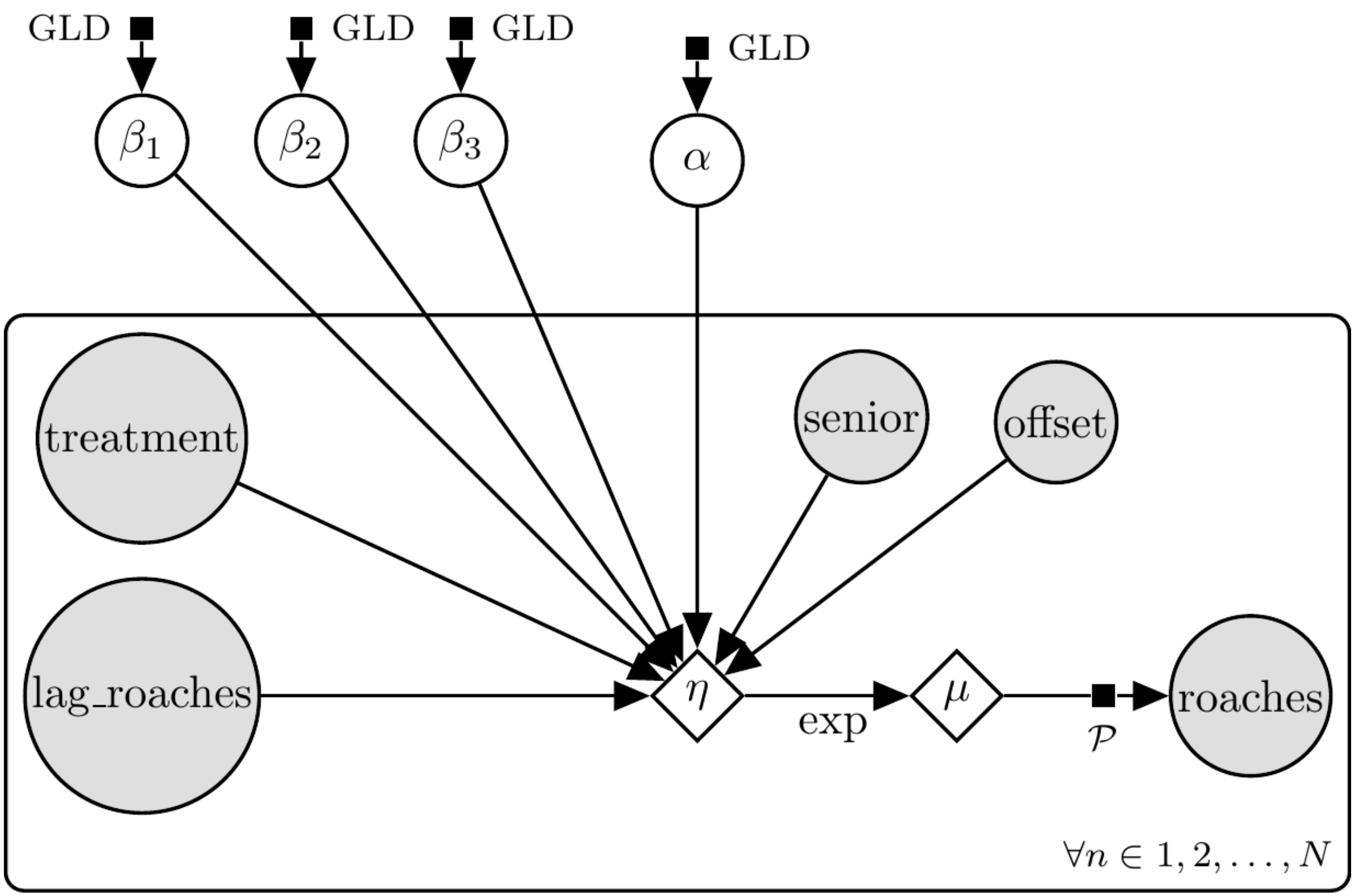


Roach Data in NYC Experiment

```
ROOT <- "https://raw.githubusercontent.com/avehtari/ROS-Examples/master/"
FILE <- "Roaches/data/roaches.csv"
roaches <- readr::read_csv(paste0(ROOT, FILE), col_types = "_idiid")
library(dplyr)
roaches <- filter(roaches, roach1 > 0)
glimpse(roaches)
```

```
## Rows: 202
## Columns: 5
## $ y         <int> 153, 127, 7, 7, 0, 73, 24, 2, 2, 0, 21, 0, 179, 136, 104, 2, 5, 1, 203…
## $ roach1    <dbl> 308.00, 331.25, 1.67, 3.00, 2.00, 70.00, 64.56, 1.00, 14.00, 138.25, 1…
## $ treatment <int> 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ senior    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ exposure2 <dbl> 0.8000000, 0.6000000, 1.0000000, 1.0000000, 1.1428571, 0.8000000, 1.14…
```

Prior Predictive Distribution for Roach Study



Roach Model

Prior Predictive Distribution in Symbols

$$\alpha \sim GLD$$
$$\beta_1 \sim GLD$$
$$\beta_2 \sim GLD$$
$$\beta_3 \sim GLD$$
$$\forall n : \eta_n \equiv \alpha + OFFSET_n + \beta_1 \times \log LAG_n + \beta_2 \times SENIOR_n + \beta_3 \times T_n$$
$$\forall n : \mu_n \equiv e^{\eta_n}$$
$$\forall n : Y_n \sim \mathcal{P}\left(\mu_n\right)$$

- In this case, the inverse link function mapping the linear predictor η_n on $\mathbb{R}$ to the outcome's conditional expectation μ_n on $\mathbb{R}_+$ is the antilog function.
- An "offset" is a predictor whose coefficient is fixed to be 1

Generalized Lambda Distribution Priors

- What do you believe about β_1, the coefficient on the logarithm of roaches in the previous period?
- What do you believe about β_2, the coefficient on whether the building is a senior living facility?
- What do you believe about β_3, the coefficient on the treatment variable?
- What do you believe about α, the expected logarithm of roaches for a building with average predictors?

Calling **stan** for the Poisson Model

```
stan_data <- list(N = nrow(roaches), offset = log(roaches$exposure2),
                  K = 3, y = roaches$y,
                  X = with(roaches, cbind(log(roach1) - mean(log(roach1)),
                                          senior - mean(senior),
                                          treatment - mean(treatment))),
                  prior_only = FALSE, m = m, r = r, a = a, s = s)

post_poisson <- stan("poisson.stan", data = stan_data)

## Trying to compile a simple C file
```

ShinyStan

```
y <- roaches$y
shinystan::launch_shinystan(post_poisson) # opens in a web browser
```

Numerical Assessment of Calibration

```r
y_rep <- rstan::extract(post_poisson, "y_rep")[[1]]; dim(y_rep)
```

```
## [1] 4000  202
```

```r
lower <- apply(y_rep, MARGIN = 2, FUN = quantile, probs = 0.25)
upper <- apply(y_rep, MARGIN = 2, FUN = quantile, probs = 0.75)
mean(roaches$y > lower & roaches$y < upper) # bad fit
```

```
## [1] 0.04950495
```

- Overall, the model is fitting the data poorly in this case, although overfitting can be a concern in other situations

Adding Overdispersion

$$\alpha \sim GLD$$
$$\beta_1 \sim GLD$$
$$\beta_2 \sim GLD$$
$$\beta_3 \sim GLD$$
$$\forall n : \eta_n \equiv \alpha + OFFSET_n + \beta_1 \times \log LAG_n + \beta_2 \times SENIOR_n + \beta_3 \times T_n$$
$$\forall n : \mu_n \equiv e^{\eta_n}$$
$$\phi \sim GLD$$
$$\forall n : \epsilon_n \sim \mathcal{G}\left(\phi, \phi\right)$$
$$\forall n : Y_n \sim \mathcal{Poisson}\left(\epsilon_n \mu_n\right)$$

* The conditional distribution of Y_n given $\epsilon_n \mu_n$ is Poisson, but the conditional distribution of Y_n given μ_n irrespective of ϵ_n is negative binomial with expectation μ_n and variance $\mu_n + \mu_n^2 / \phi$ * What are your beliefs about ϕ?

Calling `stan` for the Negative Binomial Model

```r
stan_data$m <- m; stan_data$r <- r; stan_data$a <- a; stan_data$s <- s

post_nb <- stan("negative_binomial.stan", data = stan_data)

print(post_nb, pars = c("alpha", "beta", "phi"))
```

```
## Inference for Stan model: negative_binomial.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## alpha    2.82    0.00 0.10  2.61  2.74  2.81  2.88  3.02  4472    1
## beta[1]  0.71    0.00 0.06  0.58  0.67  0.71  0.75  0.83  4070    1
## beta[2] -0.18    0.00 0.24 -0.63 -0.33 -0.18 -0.01  0.28  4086    1
## beta[3] -0.49    0.01 0.23 -0.93 -0.65 -0.50 -0.33 -0.01  2118    1
## phi      0.47    0.00 0.05  0.38  0.44  0.47  0.51  0.58  3870    1
##
## Samples were drawn using NUTS(diag_e) at Wed Apr 13 11:42:31 2022.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Model Comparison

```r
library(loo)
loo_compare(loo(post_poisson), loo(post_nb)) # warnings about high Pareto k values
```

```
##        elpd_diff se_diff
## model2     0.0       0.0
## model1 -3386.5     510.8
```

```r
loo_list <- list(loo(post_poisson, moment_match = TRUE), loo(post_nb, moment_match = TRUE))
loo_compare(loo_list)
```

```
##        elpd_diff se_diff
## model2     0.0       0.0
## model1 -3400.9     515.9
```

```r
loo_model_weights(loo_list)
```

```
## Method: stacking
## ------
##        weight
## model1 0.000
## model2 1.000
```

A Binomial Model for Romney vs Obama in 2012

```r
poll <- readRDS("GooglePoll.rds") # WantToWin is coded as 1 for Romney and 0 for Obama
collapsed <- filter(poll, !is.na(WantToWin)) %>%
             group_by(Region, Gender, Urban_Density, Age, Income) %>%
             summarize(Obama = sum(grepl("Obama", WantToWin)), n = n()) %>%
             na.omit
glimpse(collapsed)
```

```
## Rows: 516
## Columns: 7
## Groups: Region, Gender, Urban_Density, Age [143]
## $ Region        <fct> MIDWEST, MIDWEST, MIDWEST, MIDWEST, MIDWEST, MIDWEST, MIDWEST, MID…
## $ Gender        <fct> Female, Female, Female, Female, Female, Female, Female, Female, Fe…
## $ Urban_Density <fct> Rural, Rural, Rural, Rural, Rural, Rural, Rural, Rural, Rural, Rur…
## $ Age           <ord> 18-24, 18-24, 25-34, 25-34, 35-44, 45-54, 45-54, 45-54, 55-64, 55-…
## $ Income        <ord> "25,000-49,999", "50,000-74,999", "25,000-49,999", "50,000-74,999"…
## $ Obama         <int> 6, 5, 4, 3, 8, 10, 8, 0, 0, 18, 4, 10, 1, 6, 21, 13, 1, 1, 0, 40, …
## $ n             <int> 12, 7, 7, 4, 15, 28, 12, 1, 1, 44, 8, 22, 3, 8, 31, 19, 1, 1, 1, 5…
```

Prior Predictive Distribution in Symbols

- Here is how McElreath does many hierarchical binomial models
- Suppose a categorical predictor x_k has K levels

$$\sigma \sim \mathcal{E}(r)$$
$$\forall k : \beta_k \sim \mathcal{N}(m_k, \sigma)$$
$$\forall k : \mu_k = \frac{1}{1 + e^{-\beta_k}}$$
$$\forall k : y_k \sim \text{Binomial}(n_k, \mu_k)$$

- Aggregating Bernoulli random variables with a common success probability to binomial random variables is much more computationally efficient

Calling `stan`

```
stan_data <- with(collapsed, list(N = length(Obama), y = Obama, n = n,
                                  J = c(nlevels(Region), nlevels(Gender),
                                        nlevels(Urban_Density), nlevels(Age), nlevels(Income)),
                                  Region = as.integer(Region), Gender = as.integer(Gender),
                                  Urban_Density = as.integer(Urban_Density),
                                  Age = as.integer(Age), Income = as.integer(Income),
                                  prior_only = FALSE, m = rep(0, 5), r = rep(1, 5)))
post_binomial <- stan("binomial.stan", data = stan_data)
```