GLM Binomial

Chris Parrish

August 18, 2016

Contents

GLM Binomial, BPA, chap 03

references:
- notice of Ito's translations of BPA models to stan, Gelman
- BPA example models in stan by Ito
- Stan case study by Daniel C. Furr

3. Introduction to the generalized linear model (GLM)

3.5. Binomial GLM for modeling bounded counts or proportions

3.5.1. Generation and analysis of simulated data

```
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
set.seed(123)
```

data

The data generation code is in bpa-code.txt, available at http://www.vogelwarte.ch/de/projekte/publikationen/bpa/complete-code-and-data-files-of-the-book.html

```
stan_data <- read_rdump("GLM_Binomial.data.R")
```

model

stan model

GLM_Binomial.stan

```
data {
  int<lower=0> nyears;          // Number of Years
  int<lower=0> C[nyears];       // Counts
  int<lower=0> N[nyears];       // Binomial Totals
  vector[nyears] year;          // Year covariates
}

transformed data {
  vector[nyears] year_squared;

  year_squared = year .* year;
}

parameters {
  real alpha;
  real beta1;
  real beta2;
}

transformed parameters {
  vector[nyears] logit_p;

  // Linear predictor
  logit_p = alpha +
            beta1 * year +
            beta2 * year_squared;
}

model {
  // Priors
  alpha ~ normal(0, 100);
  beta1 ~ normal(0, 100);
  beta2 ~ normal(0, 100);

  // Likelihood
  // Distribution for random part
  C ~ binomial_logit(N, logit_p);
}

generated quantities {
  real<lower=0,upper=1> p[nyears];

  for (i in 1:nyears)
    p[i] = inv_logit(logit_p[i]);
}
```

Initial values

```
inits <- function() list(alpha = runif(1, -1, 1),
                         beta1 = runif(1, -1, 1),
                         beta2 = runif(1, -1, 1))
```

Parameters monitored

```
params <- c("alpha", "beta1", "beta2", "p")
```

MCMC settings

```
ni <- 2000
nt <- 1
nb <- 1000
nc <- 4
```

call Stan from R

```
fit <- stan("GLM_Binomial.stan", data = stan_data,
            init = inits, pars = params,
            chains = nc, thin = nt, iter = ni, warmup = nb,
            seed = 1,
            open_progress = FALSE)
```

results

```
## Summarize posteriors
print(fit)
```

```
## Inference for Stan model: GLM_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%
## alpha     1.02    0.00 0.06     0.89     0.97     1.02     1.06     1.15
## beta1    -0.17    0.00 0.07    -0.32    -0.22    -0.17    -0.12    -0.03
## beta2    -0.93    0.00 0.14    -1.22    -1.02    -0.93    -0.82    -0.64
## p[1]      0.58    0.00 0.03     0.53     0.57     0.58     0.60     0.64
## p[2]      0.60    0.00 0.02     0.56     0.59     0.60     0.62     0.65
## p[3]      0.62    0.00 0.02     0.58     0.61     0.62     0.64     0.66
## p[4]      0.64    0.00 0.02     0.60     0.62     0.64     0.65     0.68
## p[5]      0.65    0.00 0.02     0.62     0.64     0.65     0.66     0.69
## p[6]      0.66    0.00 0.02     0.63     0.65     0.66     0.67     0.69
## p[7]      0.68    0.00 0.01     0.65     0.67     0.68     0.69     0.70
## p[8]      0.69    0.00 0.01     0.66     0.68     0.69     0.70     0.71
## p[9]      0.70    0.00 0.01     0.67     0.69     0.70     0.70     0.72
## p[10]     0.70    0.00 0.01     0.68     0.70     0.70     0.71     0.73
## p[11]     0.71    0.00 0.01     0.69     0.70     0.71     0.72     0.74
## p[12]     0.72    0.00 0.01     0.70     0.71     0.72     0.73     0.74
## p[13]     0.72    0.00 0.01     0.70     0.72     0.72     0.73     0.75
## p[14]     0.73    0.00 0.01     0.70     0.72     0.73     0.74     0.75
## p[15]     0.73    0.00 0.01     0.71     0.72     0.73     0.74     0.75
## p[16]     0.73    0.00 0.01     0.71     0.73     0.73     0.74     0.76
```

```
## p[17]     0.73    0.00 0.01     0.71     0.73     0.74     0.74     0.76
## p[18]     0.74    0.00 0.01     0.71     0.73     0.74     0.74     0.76
## p[19]     0.74    0.00 0.01     0.71     0.73     0.74     0.74     0.76
## p[20]     0.73    0.00 0.01     0.71     0.73     0.73     0.74     0.76
## p[21]     0.73    0.00 0.01     0.71     0.72     0.73     0.74     0.76
## p[22]     0.73    0.00 0.01     0.70     0.72     0.73     0.74     0.75
## p[23]     0.72    0.00 0.01     0.70     0.72     0.73     0.73     0.75
## p[24]     0.72    0.00 0.01     0.70     0.71     0.72     0.73     0.74
## p[25]     0.71    0.00 0.01     0.69     0.71     0.71     0.72     0.74
## p[26]     0.71    0.00 0.01     0.68     0.70     0.71     0.72     0.73
## p[27]     0.70    0.00 0.01     0.68     0.69     0.70     0.71     0.72
## p[28]     0.69    0.00 0.01     0.67     0.68     0.69     0.70     0.71
## p[29]     0.68    0.00 0.01     0.65     0.67     0.68     0.69     0.70
## p[30]     0.67    0.00 0.01     0.64     0.66     0.67     0.68     0.69
## p[31]     0.66    0.00 0.01     0.63     0.65     0.66     0.66     0.68
## p[32]     0.64    0.00 0.01     0.61     0.63     0.64     0.65     0.67
## p[33]     0.63    0.00 0.02     0.60     0.62     0.63     0.64     0.66
## p[34]     0.61    0.00 0.02     0.58     0.60     0.61     0.62     0.64
## p[35]     0.59    0.00 0.02     0.55     0.58     0.59     0.60     0.63
## p[36]     0.57    0.00 0.02     0.53     0.56     0.57     0.59     0.61
## p[37]     0.55    0.00 0.02     0.50     0.53     0.55     0.57     0.60
## p[38]     0.53    0.00 0.03     0.48     0.51     0.53     0.55     0.58
## p[39]     0.50    0.00 0.03     0.45     0.48     0.50     0.52     0.56
## p[40]     0.48    0.00 0.03     0.42     0.46     0.48     0.50     0.55
## lp__  -1628.35    0.03 1.26 -1631.48 -1628.89 -1628.02 -1627.45 -1626.95
##       n_eff Rhat
## alpha  1942    1
## beta1  2207    1
## beta2  1832    1
## p[1]   2648    1
## p[2]   2767    1
## p[3]   2904    1
## p[4]   3209    1
## p[5]   4000    1
## p[6]   4000    1
## p[7]   4000    1
## p[8]   4000    1
## p[9]   3286    1
## p[10]  3022    1
## p[11]  2759    1
## p[12]  2533    1
## p[13]  2355    1
## p[14]  2220    1
## p[15]  2121    1
## p[16]  2050    1
## p[17]  2001    1
## p[18]  1969    1
## p[19]  1951    1
## p[20]  1946    1
## p[21]  1953    1
## p[22]  1972    1
## p[23]  2006    1
## p[24]  2059    1
## p[25]  2134    1
```

```
## p[26]  2239    1
## p[27]  2383    1
## p[28]  2574    1
## p[29]  2813    1
## p[30]  3084    1
## p[31]  3339    1
## p[32]  4000    1
## p[33]  4000    1
## p[34]  4000    1
## p[35]  4000    1
## p[36]  3101    1
## p[37]  2920    1
## p[38]  2764    1
## p[39]  2542    1
## p[40]  2443    1
## lp__   1915    1
##
## Samples were drawn using NUTS(diag_e) at Thu Aug 18 19:52:38 2016.
## 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).
##  The estimated Bayesian Fraction of Missing Information is a measure of
##  the efficiency of the sampler with values close to 1 being ideal.
##  For each chain, these estimates are
##  1.1 0.8 1 0.9
```

posterior parameters

This code is from a Stan case study by Daniel C. Furr (see the link at the top of this page).

```
fit.summary <- as.data.frame(summary(fit)[[1]])
wanted_pars <- c("alpha", "beta1", "beta2",
                 paste0("p[", 1:40, "]"))
estimated_values <- fit.summary[wanted_pars, c("mean", "2.5%", "97.5%")]
```

data frame

```
# Assemble a data frame to pass to ggplot()
sim_df <- data.frame(parameter = factor(wanted_pars, rev(wanted_pars)),
                     row.names = NULL)
sim_df$middle <- estimated_values[, "mean"]
sim_df$lower <- estimated_values[, "2.5%"]
sim_df$upper <- estimated_values[, "97.5%"]
str(sim_df)
```

```
## 'data.frame':    43 obs. of  4 variables:
##  $ parameter: Factor w/ 43 levels "p[40]","p[39]",..: 43 42 41 40 39 38 37 36 35 34 ...
##  $ middle   : num  1.016 -0.171 -0.925 0.585 0.603 ...
##  $ lower    : num  0.893 -0.315 -1.216 0.53 0.555 ...
##  $ upper    : num  1.1458 -0.0304 -0.6441 0.639 0.6524 ...
```

display posterior parameters

```
# Plot
ggplot(sim_df) +
  aes(x = parameter, y = middle, ymin = lower, ymax = upper) +
  scale_x_discrete() +
  labs(y = "95% interval", x = NULL) +
  geom_abline(intercept = 0, slope = 0, color = "white") +
  geom_linerange(color = "steelblue") +
  geom_point(color = "darkred") +
  theme_gray() +
  coord_flip()
```