GLM Poisson

Chris Parrish

August 18, 2016

Contents

GLM Poisson, 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)

3.3. Poisson GLM in R and WinBUGS for modeling time series of counts

3.3.1. Generation and analysis of simulated data

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

data

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_Poisson.data.R")
```

```r
## Initial values
inits <- function() list(alpha = runif(1, -2, 2),
                         beta1 = runif(1, -3, 3))

## Parameters monitored
params <- c("alpha", "beta1", "beta2", "beta3", "lambda")

## MCMC settings
ni <- 2000
nt <- 1
nb <- 1000
nc <- 4
```

model

stan model

GLM_Poisson.stan

```
data {
  int<lower=0> n;       // Number of years
  int<lower=0> C[n];    // Count
  vector[n] year;       // Year
}

transformed data {
  vector[n] year_squared;
  vector[n] year_cubed;

  year_squared = year .* year;
  year_cubed = year .* year .* year;
}

parameters {
  real<lower=-20,upper=20> alpha;
  real<lower=-10,upper=10> beta1;
  real<lower=-10,upper=10> beta2;
  real<lower=-10,upper=10> beta3;
}

transformed parameters {
  vector[n] log_lambda;

  log_lambda = alpha +
               beta1 * year +
               beta2 * year_squared +
               beta3 * year_cubed;
}

model {
  // Priors
  alpha ~ uniform(-20, 20);
  beta1 ~ uniform(-10, 10);
```

```
  beta2 ~ uniform(-10, 10);
  beta3 ~ uniform(-10, 10);

  // Likelihood
  C ~ poisson_log(log_lambda);
}

generated quantities {
  vector[n] lambda;

  lambda = exp(log_lambda);
}
```

call Stan from R

```
fit <- stan("GLM_Poisson.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_Poisson.
## 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%
## alpha          4.29    0.00 0.03     4.23     4.27     4.28     4.30
## beta1          1.25    0.00 0.04     1.16     1.22     1.25     1.27
## beta2          0.07    0.00 0.02     0.02     0.05     0.07     0.08
## beta3         -0.23    0.00 0.02    -0.27    -0.24    -0.23    -0.21
## lambda[1]     32.13    0.07 3.07    26.33    30.05    32.05    34.10
## lambda[2]     29.97    0.05 2.45    25.32    28.31    29.92    31.58
## lambda[3]     28.44    0.04 2.00    24.63    27.09    28.43    29.76
## lambda[4]     27.44    0.03 1.67    24.28    26.31    27.43    28.55
## lambda[5]     26.89    0.03 1.44    24.12    25.91    26.87    27.86
## lambda[6]     26.73    0.02 1.30    24.21    25.83    26.72    27.60
## lambda[7]     26.94    0.02 1.22    24.57    26.11    26.93    27.76
## lambda[8]     27.49    0.02 1.19    25.17    26.68    27.49    28.30
## lambda[9]     28.39    0.03 1.20    26.04    27.58    28.39    29.21
## lambda[10]    29.65    0.03 1.24    27.24    28.82    29.64    30.49
## lambda[11]    31.27    0.03 1.30    28.75    30.38    31.25    32.15
## lambda[12]    33.28    0.03 1.37    30.67    32.37    33.26    34.23
## lambda[13]    35.73    0.04 1.45    32.93    34.77    35.69    36.72
## lambda[14]    38.63    0.04 1.54    35.68    37.62    38.60    39.68
## lambda[15]    42.06    0.04 1.63    38.91    40.98    42.01    43.14
## lambda[16]    46.04    0.04 1.72    42.75    44.91    45.98    47.18
```

```
## lambda[17]    50.66    0.05 1.81    47.22    49.48    50.61    51.85
## lambda[18]    55.96    0.05 1.89    52.36    54.71    55.92    57.22
## lambda[19]    62.01    0.05 1.98    58.27    60.69    61.98    63.32
## lambda[20]    68.88    0.05 2.06    64.93    67.49    68.86    70.24
## lambda[21]    76.61    0.05 2.15    72.55    75.16    76.60    78.02
## lambda[22]    85.27    0.05 2.25    80.98    83.75    85.26    86.79
## lambda[23]    94.88    0.05 2.37    90.25    93.26    94.86    96.47
## lambda[24]   105.45    0.05 2.51   100.56   103.75   105.40   107.17
## lambda[25]   116.96    0.06 2.70   111.62   115.18   116.92   118.84
## lambda[26]   129.36    0.06 2.94   123.68   127.36   129.32   131.42
## lambda[27]   142.55    0.07 3.22   136.38   140.37   142.55   144.78
## lambda[28]   156.36    0.08 3.53   149.62   153.94   156.32   158.84
## lambda[29]   170.58    0.09 3.87   163.24   167.93   170.50   173.26
## lambda[30]   184.91    0.10 4.18   176.88   182.08   184.91   187.81
## lambda[31]   199.02    0.11 4.46   190.46   195.93   198.96   202.10
## lambda[32]   212.48    0.10 4.68   203.50   209.29   212.39   215.67
## lambda[33]   224.85    0.10 4.81   215.61   221.58   224.81   228.13
## lambda[34]   235.62    0.10 4.89   226.16   232.35   235.56   238.92
## lambda[35]   244.30    0.08 4.96   234.64   240.98   244.29   247.65
## lambda[36]   250.42    0.08 5.18   240.20   246.97   250.46   253.93
## lambda[37]   253.54    0.09 5.69   242.37   249.74   253.57   257.50
## lambda[38]   253.34    0.10 6.63   240.57   248.81   253.36   257.92
## lambda[39]   249.63    0.13 7.99   234.27   244.15   249.62   255.04
## lambda[40]   242.36    0.18 9.68   224.24   235.59   242.12   248.90
## lp__       17480.41    0.04 1.43 17476.93 17479.72 17480.70 17481.45
##               97.5% n_eff Rhat
## alpha          4.34  1687    1
## beta1          1.33  1298    1
## beta2          0.11  1633    1
## beta3         -0.18  1302    1
## lambda[1]     38.32  2049    1
## lambda[2]     34.91  2391    1
## lambda[3]     32.45  2611    1
## lambda[4]     30.83  2811    1
## lambda[5]     29.79  2914    1
## lambda[6]     29.33  2871    1
## lambda[7]     29.33  2630    1
## lambda[8]     29.81  2305    1
## lambda[9]     30.75  2019    1
## lambda[10]    32.07  1813    1
## lambda[11]    33.80  1669    1
## lambda[12]    35.99  1575    1
## lambda[13]    38.62  1516    1
## lambda[14]    41.78  1482    1
## lambda[15]    45.43  1467    1
## lambda[16]    49.62  1468    1
## lambda[17]    54.41  1484    1
## lambda[18]    59.87  1517    1
## lambda[19]    66.07  1568    1
## lambda[20]    73.06  1642    1
## lambda[21]    80.93  1741    1
## lambda[22]    89.73  1867    1
## lambda[23]    99.57  2010    1
## lambda[24]   110.49  2148    1
```

```
## lambda[25]   122.29  2250    1
## lambda[26]   135.12  2207    1
## lambda[27]   148.78  2172    1
## lambda[28]   163.22  1953    1
## lambda[29]   178.14  1865    1
## lambda[30]   193.22  1805    1
## lambda[31]   207.75  1788    1
## lambda[32]   221.57  2079    1
## lambda[33]   234.26  2229    1
## lambda[34]   245.16  2543    1
## lambda[35]   254.10  4000    1
## lambda[36]   260.50  4000    1
## lambda[37]   264.64  4000    1
## lambda[38]   266.16  4000    1
## lambda[39]   265.48  4000    1
## lambda[40]   262.03  2821    1
## lp__       17482.18  1421    1
##
## Samples were drawn using NUTS(diag_e) at Thu Aug 18 19:52:14 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 0.9 1.1
```

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", "beta3",
                 paste0("lambda[", 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':    44 obs. of  4 variables:
##  $ parameter: Factor w/ 44 levels "lambda[40]","lambda[39]",..: 44 43 42 41 40 39 38 37 36 35 ...
##  $ middle   : num  4.285 1.2455 0.0691 -0.2295 32.1284 ...
##  $ lower    : num  4.2286 1.1599 0.0219 -0.2732 26.3253 ...
##  $ upper    : num  4.343 1.331 0.115 -0.184 38.324 ...
```

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()
```