STAT 656: Bayesian Data Analysis
Fall 2024
Homework 2

Juanwu Lu*

```
library("bayesplot")
library("ggplot2")
library("rstan")
options(repr.plot.width = 6, repr.plot.height = 4)
bayesplot_theme_set(theme_default(base_size = 24, base_family = "sans"))
```

Synthetic Data

The file `hw2_synthetic.csv` is a dataset of count-valued measurements $\boldsymbol{y} = \{y_1, \ldots, y_n\}$, with $y_i \in 0, 1, \ldots$. Each output y_i has an associated $x_i = (x_{i,1}, x_{i,2}) \in \mathbb{R}^2$, and write $\boldsymbol{x} = \{x_1, \ldots, x_n\}$'s as $\boldsymbol{x}$. We model y_i as

$$y_i|\beta \sim \text{Poisson}(e^{f(x_i,\beta)}).$$

Here, the exponential is to ensure the Poisson rate is always positive, and the function $f(x_i, \beta) = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \beta_3 x_{i,1}^2 + \beta_4 x_{i,2}^2 + \beta_5 x_{i,1} x_{i,2}$.

1. (25 points) **Solution**:

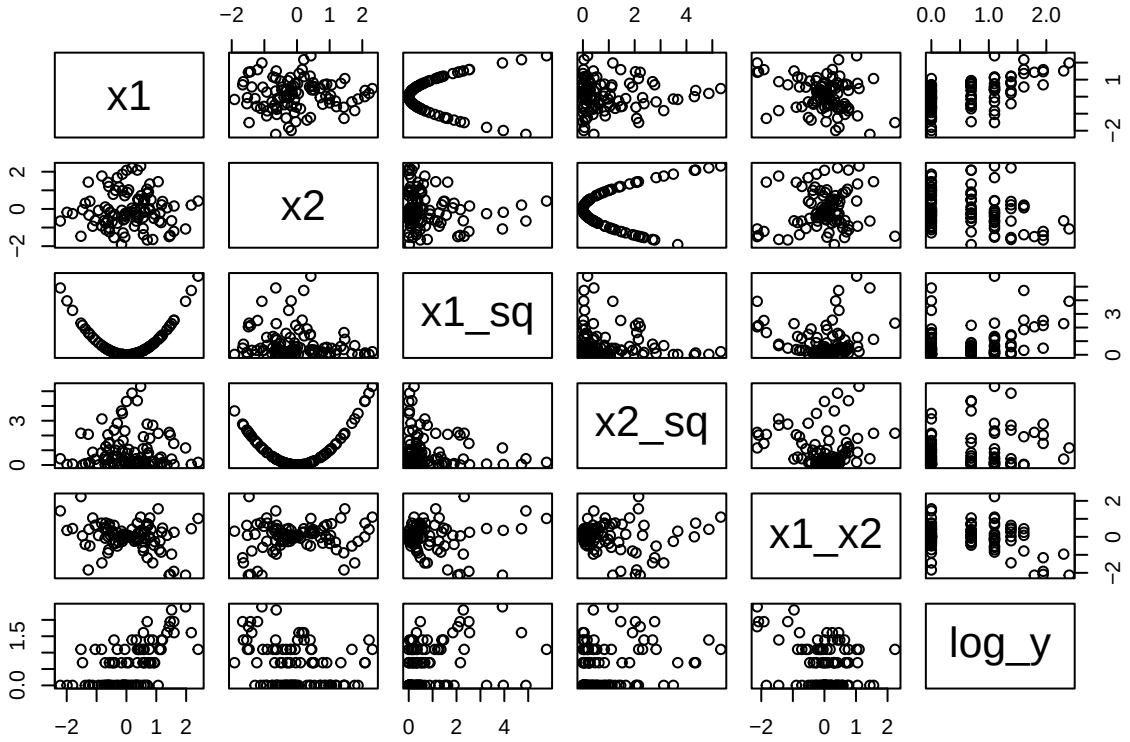
```
# Read the data
if (file.exists("data/hw2_synthetic.csv")) {
  data <- read.csv("data/hw2_synthetic.csv", header = TRUE)
} else {
  stop("FileNotFound: data file not found at 'data/hw2_synthetic.csv'.")
}
summary(data)
```

```
##        x1                 x2                  y          
##  Min.   :-2.2147   Min.   :-1.91436   Min.   : 0.00  
##  1st Qu.:-0.4942   1st Qu.:-0.65105   1st Qu.: 0.00  
##  Median : 0.1139   Median :-0.17722   Median : 1.00  
##  Mean   : 0.1089   Mean   :-0.03781   Mean   : 1.29  
##  3rd Qu.: 0.6915   3rd Qu.: 0.50090   3rd Qu.: 2.00  
##  Max.   : 2.4016   Max.   : 2.30798   Max.   :10.00
```

Since the model is a Poisson regression, the logarithm of the response variable is assumed to be a linear combination of the kernel features. The visualization below shows the relationship between te logarithm of the response variable and the kernel features. It is shown that $\log y$ is roughly positively correlated with x_1 and $x_1 x_2$, and roughly negatively correlated with x_2, x_1^2, and x_2^2.

*College of Engineering, Purdue University, West Lafayette, IN, USA

```r
# Preprocess data to create the kernel terms
data$x1_sq <- data$x1^2
data$x2_sq <- data$x2^2
data$x1_x2 <- data$x1 * data$x2
data$log_y <- log(data$y + 1)
data <- data[, c("x1", "x2", "x1_sq", "x2_sq", "x1_x2", "y", "log_y")]
plot(data[, c("x1", "x2", "x1_sq", "x2_sq", "x1_x2", "log_y")])
```



Therefore, the weights β_i can be both positive and negative. Given no prior knowledge on the features, I choose a non-informative prior for the weights, *i.e.,* isotropic normal distribution centered at 0: $\mathcal{N}(0, \sigma^2\mathbf{I})$. The Stan model is implemented as follows:

```
linreg_poisson_code <- "
    // Input arguments to the model
    data {
        int<lower=0> n;          // Number of observations
        int<lower=0> k;          // Number of features
        real<lower=0> pr_std;    // Prior coefficients standard deviation
        matrix[n, k] x;          // Observation matrix
        int<lower=0> y[n];                  // Integer response vector
    }

    // Latent paramters of interests
    parameters {
        vector[k] beta;          // Coefficients
    }

    // Transformed parameters for MCMC sampling
    transformed parameters {
        vector[n] lambda = exp(x * beta);    // Poisson rate
    }
```

```
    // PoissoXn Linear Regression Model
    model {
        beta ~ normal(0, pr_std);           // Prior on the coefficients
        y ~ poisson(lambda);                // Poisson emission
    }

    // Retrieve MCMC samples
    generated quantities {
        real y_hat[n];
        y_hat = poisson_rng(lambda);
    }
"
linreg_poisson_model <- stan_model(
  model_name = "poisson_regression",
  model_code = linreg_poisson_code
)
x <- data[, c("x1", "x2", "x1_sq", "x2_sq", "x1_x2")]
x[, "intercept"] <- 1
y <- data$y
reg_data <- list(n = nrow(x), k = ncol(x), pr_std = 1.0, x = x, y = y)
samples <- sampling(
  linreg_poisson_model,
  data = reg_data,
  iter = 10000,
  warmup = 2000,
  chains = 4,
  seed = 42,
  show_messages = FALSE,
)
```

```
##
## SAMPLING FOR MODEL 'poisson_regression' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.12 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: 2001 / 10000 [ 20%]  (Sampling)
## Chain 1: Iteration: 3000 / 10000 [ 30%]  (Sampling)
## Chain 1: Iteration: 4000 / 10000 [ 40%]  (Sampling)
## Chain 1: Iteration: 5000 / 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: 0.092 seconds (Warm-up)
## Chain 1:                0.406 seconds (Sampling)
```

```
## Chain 1:                0.498 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'poisson_regression' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 8e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.08 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: 2001 / 10000 [ 20%]  (Sampling)
## Chain 2: Iteration: 3000 / 10000 [ 30%]  (Sampling)
## Chain 2: Iteration: 4000 / 10000 [ 40%]  (Sampling)
## Chain 2: Iteration: 5000 / 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: 0.091 seconds (Warm-up)
## Chain 2:                0.444 seconds (Sampling)
## Chain 2:                0.535 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'poisson_regression' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 5e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.05 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 3: Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 3: Iteration: 2000 / 10000 [ 20%]  (Warmup)
## Chain 3: Iteration: 2001 / 10000 [ 20%]  (Sampling)
## Chain 3: Iteration: 3000 / 10000 [ 30%]  (Sampling)
## Chain 3: Iteration: 4000 / 10000 [ 40%]  (Sampling)
## Chain 3: Iteration: 5000 / 10000 [ 50%]  (Sampling)
## Chain 3: Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 3: Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 3: Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 3: Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 3: Iteration: 10000 / 10000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.091 seconds (Warm-up)
## Chain 3:                0.401 seconds (Sampling)
## Chain 3:                0.492 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'poisson_regression' NOW (CHAIN 4).
```

```
## Chain 4:
## Chain 4: Gradient evaluation took 1.3e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.13 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 4: Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 4: Iteration: 2000 / 10000 [ 20%]  (Warmup)
## Chain 4: Iteration: 2001 / 10000 [ 20%]  (Sampling)
## Chain 4: Iteration: 3000 / 10000 [ 30%]  (Sampling)
## Chain 4: Iteration: 4000 / 10000 [ 40%]  (Sampling)
## Chain 4: Iteration: 5000 / 10000 [ 50%]  (Sampling)
## Chain 4: Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 4: Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 4: Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 4: Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 4: Iteration: 10000 / 10000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.089 seconds (Warm-up)
## Chain 4:                0.431 seconds (Sampling)
## Chain 4:                0.52 seconds (Total)
## Chain 4:
```

The visualization below shows the

```r
samples <- as.data.frame(samples)
colnames(samples)[seq_len(ncol(x))] <- colnames(x)
mcmc_areas(
  samples[, seq_len(ncol(x))],
  pars = colnames(x),
  prob = 0.95
)
```

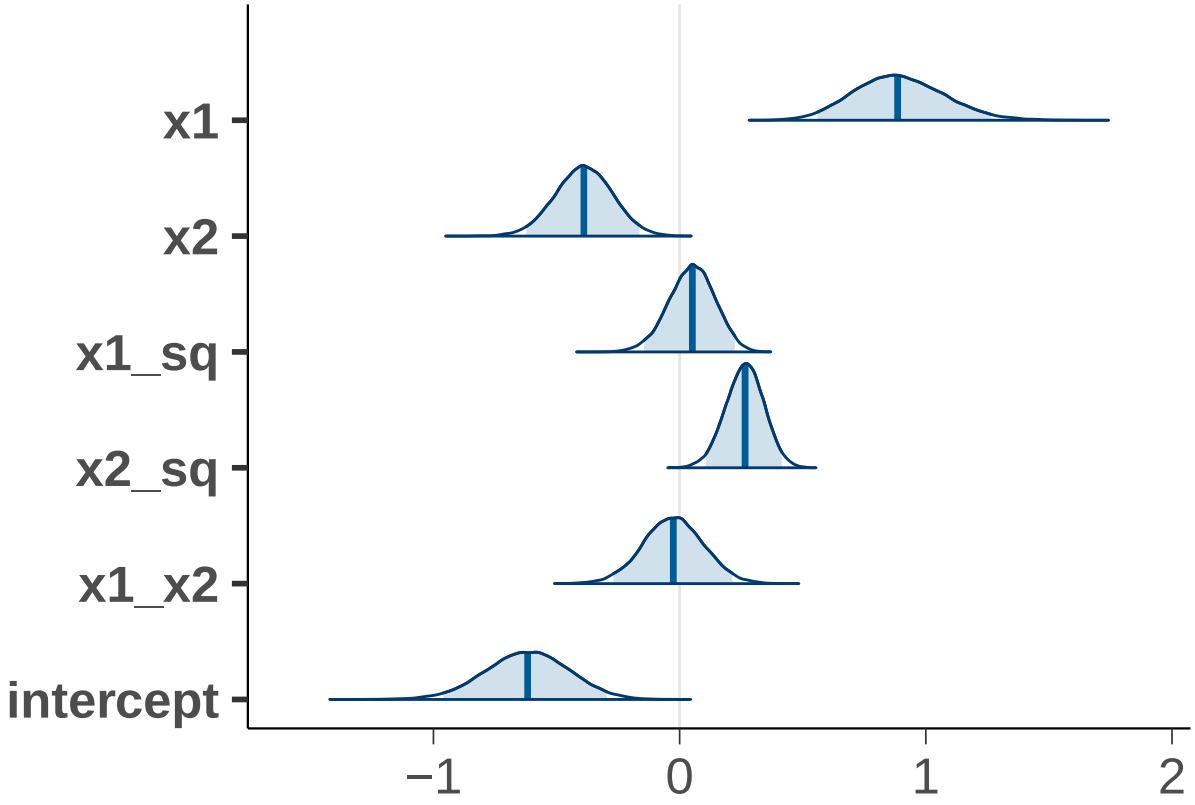


Figure 1: Posterior Distributions of the Coefficients.