STAT 656: Bayesian Data Analysis
Fall 2024
Homework 2

Juanwu Lu*

```
library("bayesplot")
library("ggplot2")
library("patchwork")
library("rstan")
options(repr.plot.width = 6, repr.plot.height = 4)
bayesplot_theme_set(theme_default(base_size = 12, 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) With the provided data, perform a Bayesian analysis on the parameters of the model above to decide **which terms in the expression for $f(x)$ you think are important**. State clearly what your prior over β is, and how you arrived at your conclusion, including any useful figures (especially of the posterior distribution). You can use Stan.

   Solution:

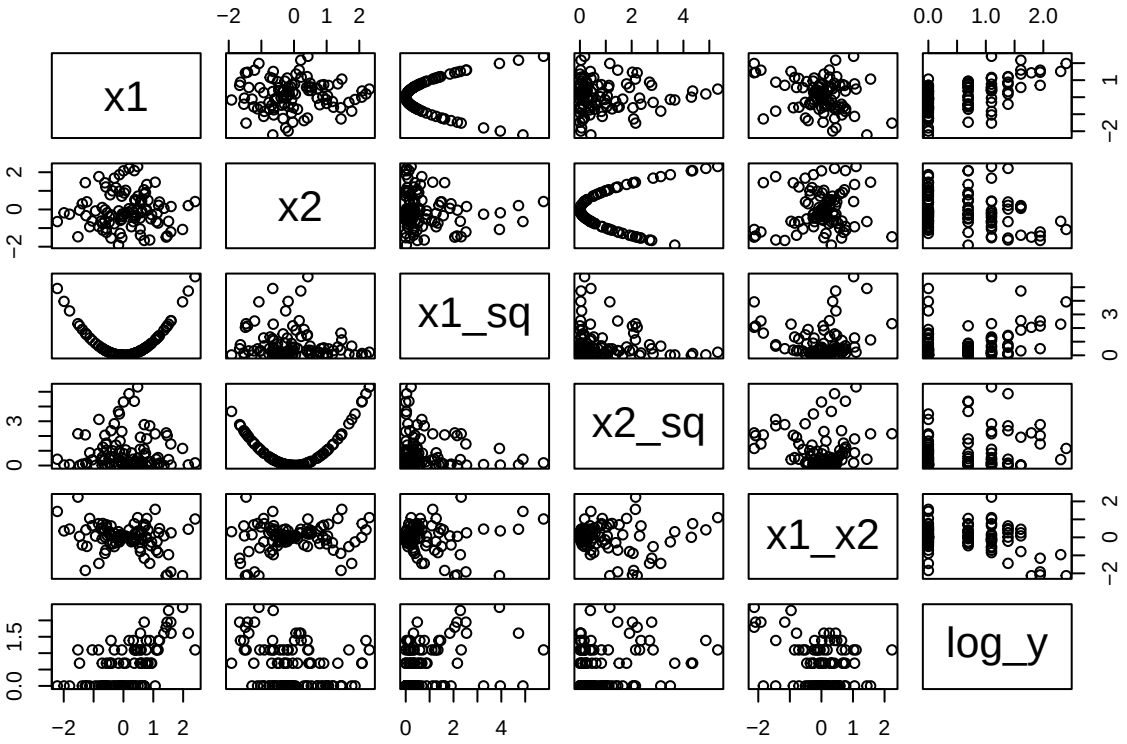
   ```
   # Read the data
   if (file.exists("data/hw2_synthetic.csv")) {
     data <- read.csv("data/hw2_synthetic.csv", header = TRUE)
   } else {
     stop("FileNotFound: data file not found at 'data/hw2_synthetic.csv'.")
   }
   summary(data)
   ```

   ```
   ##        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
   ```

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

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 the 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.

```
# 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.,* the isotropic normal distribution: $\mathcal{N}(0, 10^2\mathbf{I})$. 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 parameters 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 = 10.0, x = x, y = y)
nfit <- sampling(
  linreg_poisson_model,
  data = reg_data,
  iter = 10000,
  warmup = 2000,
  chains = 1,
  seed = 42,
  show_messages = FALSE,
)
```

The visualization below shows the posterior distributions of the coefficients $\boldsymbol{\beta} = \{\beta_0, \ldots, \beta_5\}$ with 95% intervals. Based on the results, the coefficients of x_1^2, and $x_1 x_2$ are close to zero (*i.e.,* with zero within its 95% posterior interval), while the coefficients of x_1, x_2, and x_2^2 are significantly different from zero. Therefore, the terms x_1, x_2, and x_2^2 are important in the expression for $f(x)$.

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

2. (25 points) Having decided which terms in f are important, keep only those and discard the rest, resulting in a possibly simpler model. Now perform a Bayesian analysis over the parameters of this model. Note that you are using the data twice, once to select the model and next to fit it, but we will not worry about that. Compare the posteriors for both models.

   Solution:

   Based on the analysis in the previous question, the terms x_1, x_2, and x_2^2 are kept in the model. The

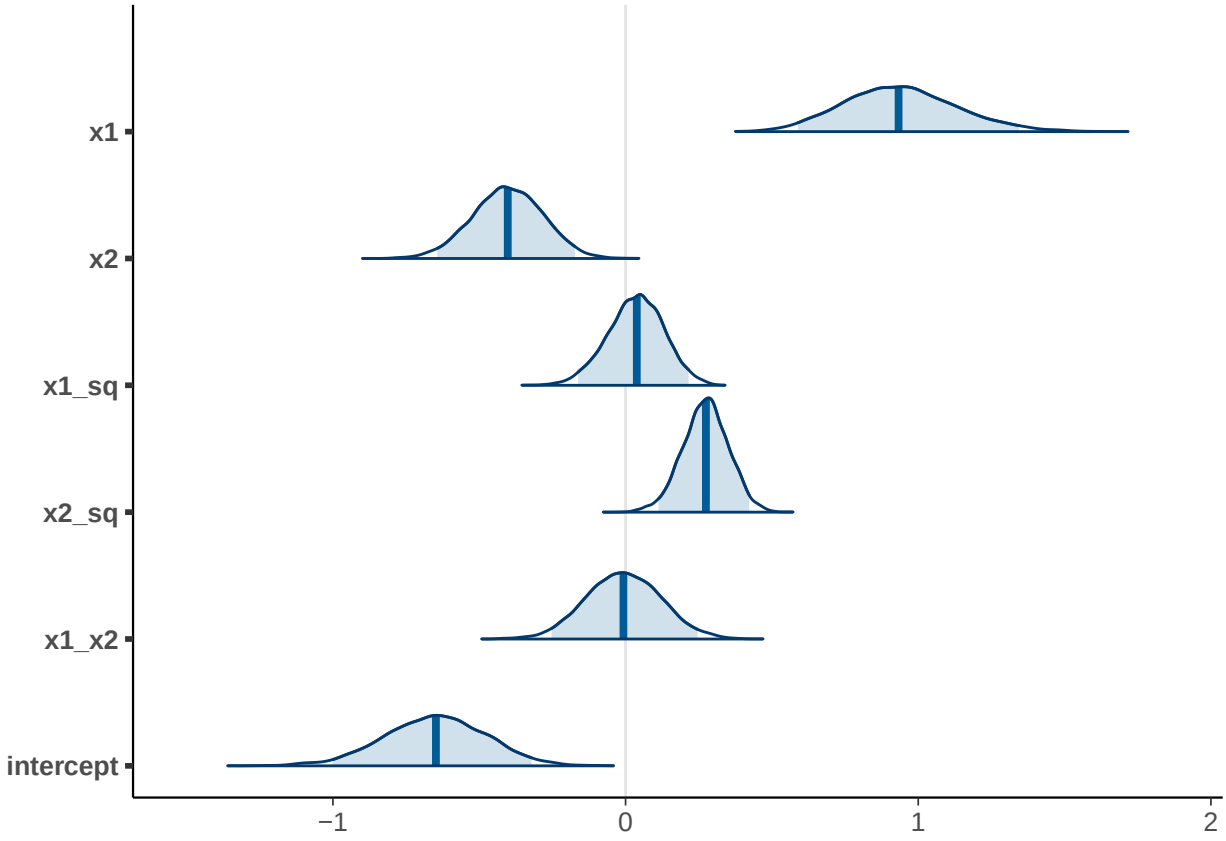


Figure 1: Posterior Distributions of the Coefficients.

new stan model is implemented as follows:

```
x_simple <- x[, c("x1", "x2", "x2_sq", "intercept")]
reg_data_simple <- list(
  n = nrow(x_simple),
  k = ncol(x_simple),
  pr_std = 10.0,
  x = x_simple,
  y = y
)
nfit_simple <- sampling(
  linreg_poisson_model,
  data = reg_data_simple,
  iter = 10000,
  warmup = 2000,
  chains = 1,
  seed = 42,
  show_messages = FALSE,
)
```

The visualization below shows the posterior distributions of the coefficients $\boldsymbol{\beta}' = \{\beta_0, \beta_1, \beta_2, \beta_4\}$ with 95% intervals. The results show that the coefficients of x_1, x_2, and x_2^2 are all significantly different from zero (*i.e.,* with zero outside its 95% posterior interval).

```
samples_simple <- as.data.frame(nfit_simple)
colnames(samples_simple)[seq_len(ncol(x_simple))] <- colnames(x_simple)
mcmc_areas(
  samples_simple[, seq_len(ncol(x_simple))],
  pars = colnames(x_simple),
  prob = 0.95,
)
```

Compared to the model from the previous question, the posterior distributions of the coefficients in this simpler model are more concentrated around its mean, which indicates that the simpler model is more confident about the estimation.

3. (25 points) Perform posterior predictive checks for both models, being sure to explain what you are doing. Which model do you think fits the data better?

Solution:

The posterior predictive checks (PPC) are performed to evaluate the goodness-of-fit of the models. The PPC is done by comparing the observed data with the data simulated from the posterior predictive distribution. Figure 3 shows the posterior predictive distributions of the two models. The results show that the simpler model fits data better than the full model since the posterior predictive distribution of the simpler model is more concentrated around the observed data. As a result, the observed data should have a higher posterior predictive density under the simpler model.

```
y_hat <- extract(nfit)$y_hat
y_hat_simple <- extract(nfit_simple)$y_hat
plot1 <- ppc_dens_overlay(
  yrep = log(y_hat + 1), y = log(data$y + 1)
) + ggplot2::labs(
  title = "Full Model", x = expression("log(y + 1)"), y = "Density"
)
plot2 <- ppc_dens_overlay(
  yrep = log(y_hat_simple + 1), y = log(data$y + 1)
```

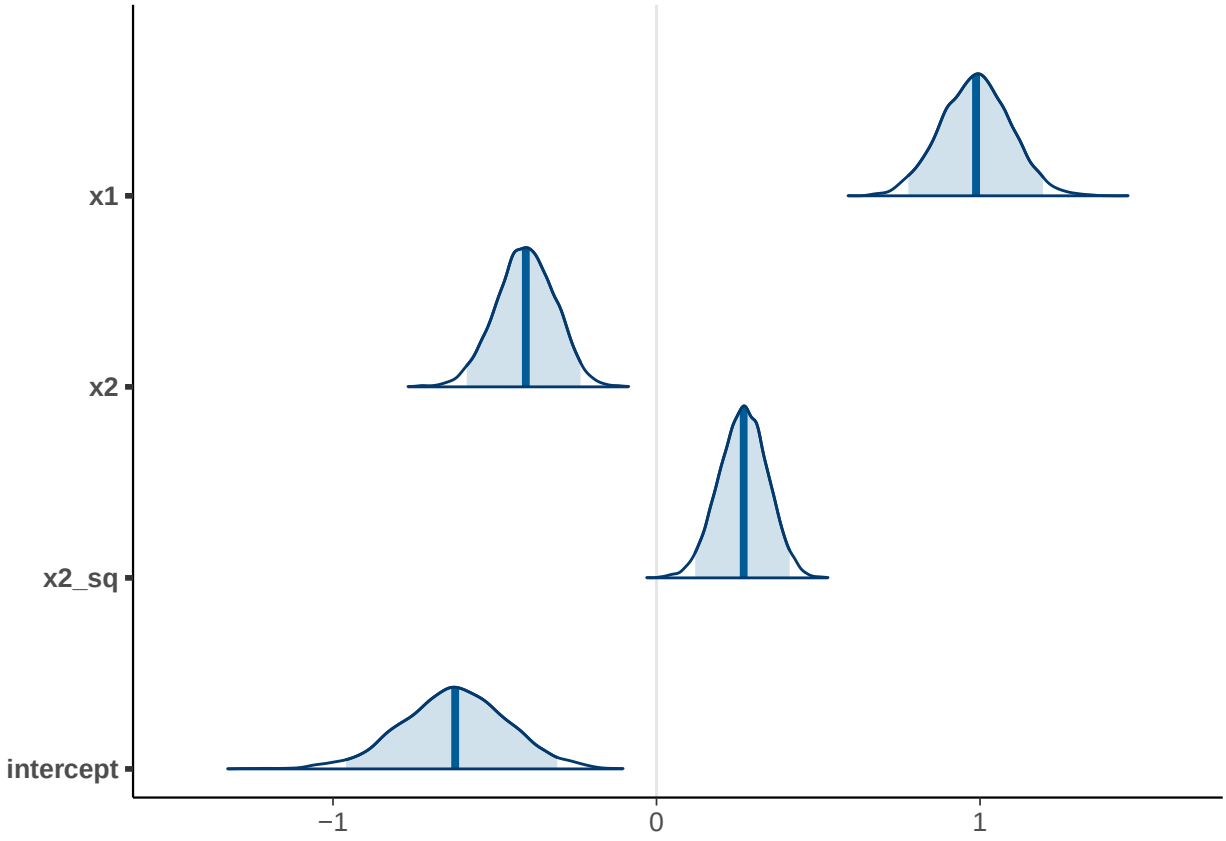


Figure 2: Posterior Distributions of the Coefficients.

```
) + ggplot2::labs(
  title = "Simple Model", x = expression("log(y + 1)"), y = "Density"
)
plot1 + plot2
```

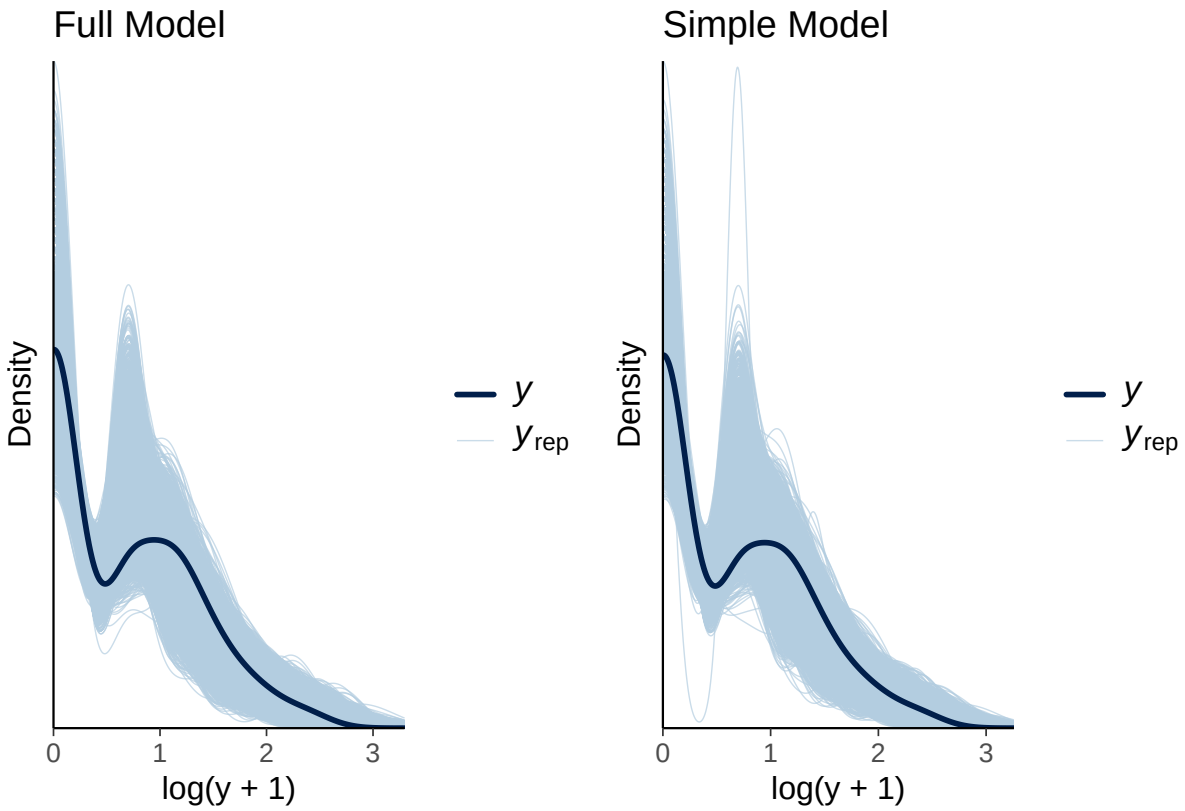


Figure 3: Overlay of the Posterior Predictive Distributions.

4. (25 points) Use the results from the second model to create a contour plot showing the log average Poisson intensity as a function of x. In other words, plot $\log \mathbb{E}_{p(\beta|\boldsymbol{x},\boldsymbol{y})}[\exp(f(\boldsymbol{x},\boldsymbol{\beta}))]$ as a function of the two components of x (you can restrict the component ranges from -10 to $+10$).

Solution:

The code below generates the contour plot as requested.

```
log_avg_intensity <- function(x1, x2, weights) {
  intensity <- weights[, 1] * x1 +
    weights[, 2] * x2 +
    weights[, 3] * x2^2 +
    weights[, 4]
  out <- log(mean(exp(intensity)))
  return(out)
}

x1 <- seq(-10, 10, 0.5)
x2 <- seq(-10, 10, 0.5)
grid <- expand.grid(x1 = x1, x2 = x2)
weight_samples <- samples_simple[, 1:4]
z <- mapply(
```

```
  function(x1, x2) log_avg_intensity(x1, x2, weight_samples),
  grid$x1,
  grid$x2
)
z <- matrix(z, nrow = length(x1), ncol = length(x2))
col <- hcl.colors(10, "YlOrRd")
contour(
  x1,
  x2,
  z,
  col = col,
  xlab = "x1",
  ylab = "x2",
  main = "Log Average Intensity"
)
```