SDS 383D: Modeling II **Spring 2017**

Section 3: Bayesian GLMs

3.1 Modeling non-Gaussian observations

So far, we've assumed real-valued observations. In this setting, our likelihood model is a univariate normal, parametrized by a mean $x_i^T\beta$ and some precision that does not directly depend on the value of x_i. In general, $x_i^T\beta$ will take values in $\mathbb{R}$

If we don't want to use a Gaussian likelihood, we typically won't be able to parametrize our data using a real-valued parameter. Instead, we must transform it via an appropriate link function. This is, in essence, the generalized linear model.

As a first step into other types of data, let's consider binary valued observations. Here, the natural likelihood model is a Bernoulli random variable; however we cannot directly parametrize this by $x_i^T\beta$. Instead, we must transform $x_i^T\beta$ to lie between 0 and 1 via some function $g^{-1} : \mathbb{R} \to (0, 1)$. We can then write a linear model as

$$\begin{aligned} y_i|p_i \sim & \text{Bernoulli}(p_i) \\ p_i = & g^{-1}(x_i^T\beta) \\ \beta|\theta \sim & \pi_\theta(\beta) \end{aligned}$$

where $\pi_\theta(\beta)$ is our choice of prior on β. Unfortunately, there is no choice of prior here that makes the model conjugate.

Let's start off with a normal prior on β. One appropriate function for g^{-1} is the CDF of the normal distribution – known as the probit function. This is equivalent to assuming our data are generated according to

$$\begin{aligned} y_i = & \begin{cases} 1 & if z > 0 \\ 0 & \text{otherwise} \end{cases} \\ z_i \sim & \text{N}(x_i^T\beta, \tau^2) \end{aligned}$$

If we put a normal-inverse gamma prior on β and τ, then we have a *latent* regression model on the (x_i, z_i) pairs, that is idential to what we had before! Conditioned on the z_i, we can easily sample values for β and τ.

Exercise 3.1 *To complete our Gibbs sampler, we must specify the conditional distribution $p(y_i|x_i, z_i, \beta, \tau)$. Write down the form of this conditional distribution, and write a Gibbs sampler to sample from the posterior distribution. Test it on the dataset* `pima.csv`*, which contains diabetes information for women of Pima indian heritage. The dataset is from National Institute of Diabetes and Digestive and Kidney Diseases, full information and explanation of variables is available at* `http://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes`*.*

Proof:

$$p(y_i|z_i) = \mathbb{1}((z_i > 0 \cap y_i = 1) \cup (z_i < 0 \cap y_i = 0))$$

$p(z|y)$ is truncated normal:

$$p(z|y=1) = \frac{p(z, y=1)}{p(y=1)} = \frac{\mathbb{1}(z>0)p(z)}{\int_{z>0} p(z)}$$

$$p(z|y=0) = \frac{p(z, y=0)}{p(y=0)} = \frac{\mathbb{1}(z<0)p(z)}{\int_{z<0} p(z)}$$

where $z \sim N(x'\beta, w)$. β and $w(\tau^2)$ has normal-gamma prior:

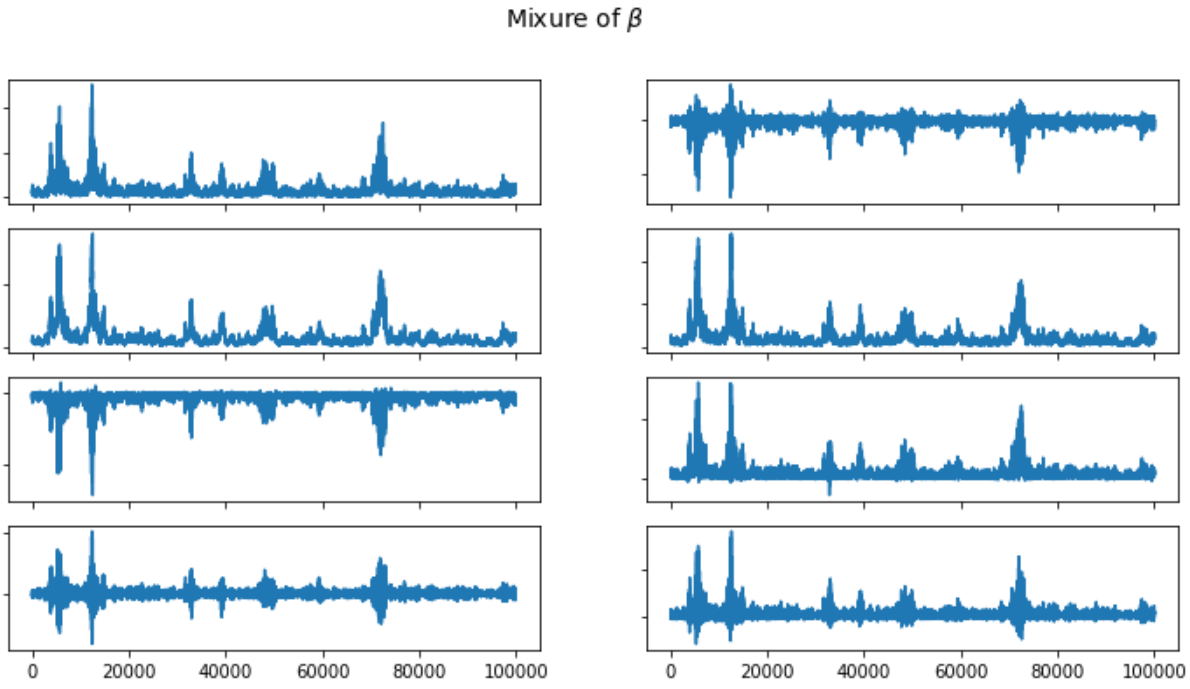
$$\beta \sim N(\mu_0, (wK)^{-1});\ w \sim \Gamma(a, b)$$

Set $a = 1, b = 0.1, \mu_0 = \mathbf{0}, K = I$. Using results from Section 2, the posterior distributions of β and w are:

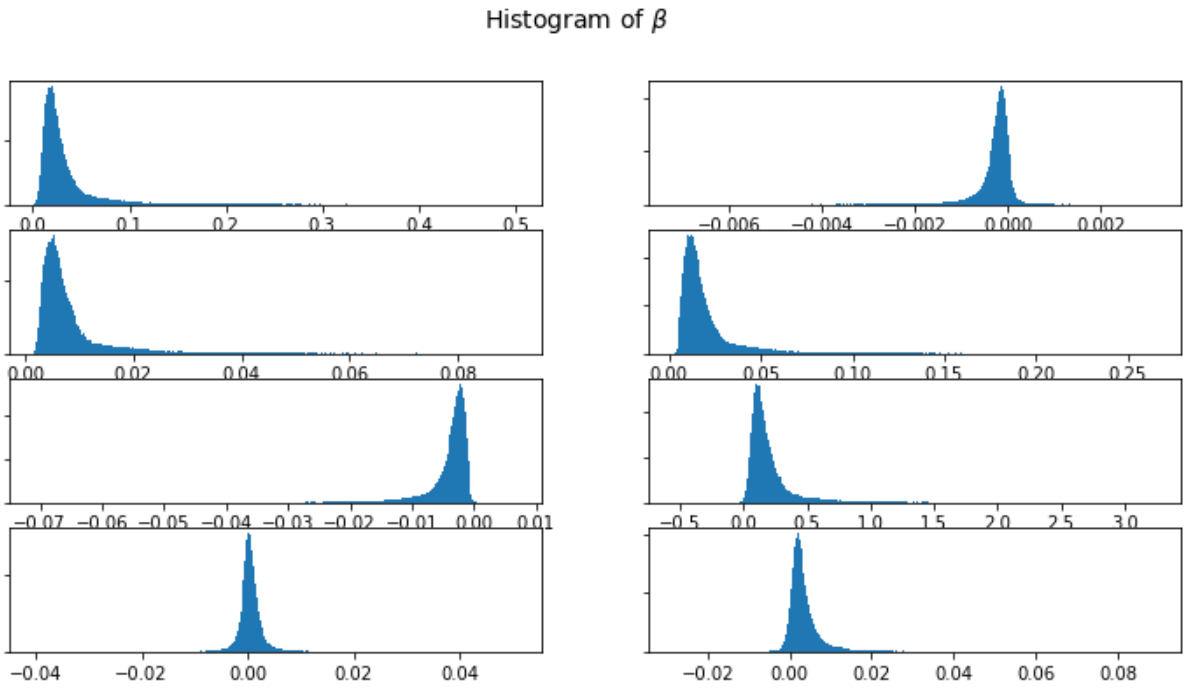
$$p(\beta|w, z) = N(\mu_n, (wG)^{-1});\ p(w|z) = \Gamma(a_n, b_n)$$

where $G = X'X + I,\ \mu_n = G^{-1}X'z,\ a_n = a + \frac{n}{2},\ b_n = b + \frac{1}{2}(z'z - \mu_n' G \mu_n)$. ■

The following figure shows that even after a long chain, the mixtures aren't stable.



Here are the density histograms for the 8 betas besides intercept:



Exercise 3.2 *Sadly, the posterior isn't in a "known" form. As a starting point, let's find the maximum a posteriori estimator (MAP). The dataset "titantic.csv" contains survival data from the Titanic; we're going to look at probability of survival as a function of age. For now, we're going to assume the intercept of our regression is zero – i.e. that β is a scalar. Write a function (that can use a black-box optimizer! No need to reinvent the wheel. It shouldn't be a long function) to estimate the MAP of β. Note that the MAP corresponds to the frequentist estimator using a ridge regularization penalty.*

Another choice for $g^{-1}(\theta)$ might be the logit function, $\frac{1}{1+e^{-x^T\beta}}$. In this case, it's less obvious to see how we can construct an auxilliary variable representation (it's not impossible! See citetPolScoWin2013. But for now, we'll assume we haven't come up with something). So, we're stuck with working with the posterior distribution over β.

Proof: Let $p_i \equiv g^{-1}(x_i'\beta) = (1+e^{-x_i'\beta})^{-1}$, and apply previous assumptions of $\mu_0 = \mathbf{0}, K = I$:

$$\begin{aligned} p(\beta|y,x,w) = \frac{p(\beta,y|w,x)}{p(y|w,x)} &\propto p(y|\beta,w,x)p(\beta|w,x) \\ &\propto exp\left(-0.5(\beta-\mu_0)'(wK)(\beta-\mu_0)\right)\prod_i p_i^{y_i}(1-p_i)^{1-y_i} \\ &\propto exp\left(-\frac{w}{2}\beta^2\right)\prod_i\left(\frac{1}{1+e^{-x_i\beta}}\right)^{y_i}\left(\frac{e^{-x_i\beta}}{1+e^{-x_i\beta}}\right)^{1-y_i} \end{aligned}$$

take log:

$$ln(p(\beta|y,w)) \propto -\frac{w\beta^2}{2} - \sum_i\left((1-y_i)x_i\beta + ln(1+e^{-x_i\beta})\right)$$

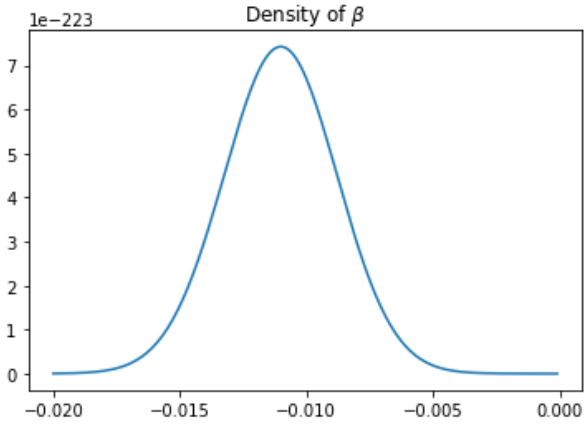
the gradient function is:

$$\frac{d(-ln(p(\beta))}{d\beta} = w\beta + (1-\mathbf{y})'\mathbf{x} - \sum_i \frac{x_i e^{-x_i\beta}}{1+e^{-x_i\beta}}$$

Assume $w = 1$, the MAP estimate for β is -0.011. ■

Exercise 3.3 *OK, we don't know how to sample from the posterior, but we can at least look at it. Write a function to calculate the posterior pdf $p(\beta|\mathbf{x}, \mathbf{y}, \mu, \sigma^2)$, for some reasonable hyperparameter values μ and θ (up to a normalizing constant is fine!). Plot over a reasonable range of β (your MAP from the last question should give you a hint of a reasonable range).*

Proof: Assume $w = 1, \mu_0 = 0$: ■



The Laplace approximation is a method for approximating a distribution with a Gaussian, by matching the mean and variance at the mode.[1] Let P^* be the (unnormalized) PDF of a distribution we wish to approximate. We start by taking a Taylor expansion of the log (unnormalized) PDF at the global maximizing value x^*

$$\log P^*(x) \approx \log P^*(x^*) - \frac{c}{2}(x - x^*)^2$$

where $c = -\frac{\delta^2}{\delta x^2} \log P^*(x)\Big|_{x=x^*}$.

We approximate P^* with an unnormalized Gaussian, with the same mean and variance as P^*:

$$Q^*(x) = P^*(x^*) \exp\left\{-\frac{c}{2}(x - x^*)^2\right\}$$

Exercise 3.4 *Find the mean and precision of a Gaussian that can be used in a Laplace approximation to the posterior distribution over β.*

[1] More generally, the Laplace approximation is used to approximate integrands of the form $\int_A e^{Nf(x)}dx...$ but for our purposes we will always be working with PDFs.

Proof: The mean is $x*$ and precision is c. Using information from exercise 3.2:

$$c = -\frac{\delta^2}{\delta\beta^2}\log P^*(\beta)\Big|_{\beta=\beta^*} = -\frac{\delta}{\delta\beta}\left(-w\beta + \sum \frac{x_i}{1+e^{x_i\beta}}\right)$$
$$= w + \sum_i \frac{x_i^2 e^{x_i\beta}}{(1+e^{x_i\beta})^2}$$

For our example, $x^* = -0.011, c = 200894.805$. ■

Exercise 3.5 *That's all well and good... but we probably have a non-zero intercept. We can extend the Laplace approximation to multivariate PDFs. This amounts to estimating the precision matrix of the approximating Gaussian using the negative of the Hessian – the matrix of second derivatives*

$$H_{ij} = \frac{\delta^2}{\delta x_i \delta x_j}\log P^*(x)\Big|_{x=x^*}$$

Use this to approximate the posterior distribution over β. Give the form of the approximating distribution, plus 95% marginal credible intervals for its elements.

Proof: If X has dimension higher than one:

$$ln(p(\beta|y,w)) \propto -\frac{w\beta'\beta}{2} - \sum_i \left((1-y_i)x_i'\beta + ln(1+e^{-x_i'\beta})\right)$$

$$\frac{\delta ln(p(\beta))}{\delta\beta} \propto -w\beta - \sum (1-y_i)x_i + \sum \frac{1}{1+e^{x_i'\beta}}x_i = -w\beta - X'(1-y) + X'\left(\frac{1}{1+e^{X\beta}}\right)$$

$$\frac{\delta^2 ln(p(\beta))}{\delta\beta\delta\beta'} \propto -wI - \sum_i \frac{e^{x_i'\beta}}{(1+e^{x_i'\beta})^2}x_i x_i'$$

Using Python's Scipy optimizer:

$$\beta_{map} = x^* = [-0.07905, -0.00886]$$

$$C = \begin{bmatrix} 183.726 & 5495.036 \\ 5495.036 & 201782.325 \end{bmatrix} \qquad C^{-1} = \Sigma = \begin{bmatrix} 2.9340e^{-2} & -7.9901e^{-4} \\ -7.9901e^{-4} & 2.6715e^{-5} \end{bmatrix}$$

95% C.I. for $\beta_0 = (-0.4148, 0.2567)$. 95% C.I. for $\beta_1 = (-0.0190, 0.0013)$. ■

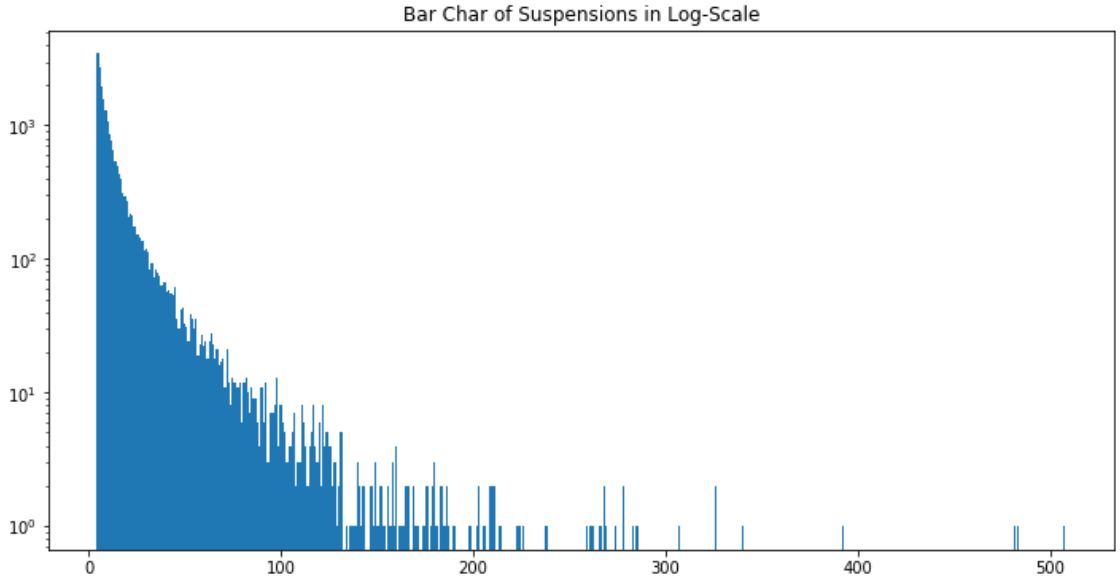
Let's try the same thing with a Poisson likelihood. Here, the obvious transformation is to let $g^{-1}(\theta) = e^\theta$, i.e.

$$y_i|p_i \sim \text{Poisson}(\lambda_i)$$
$$\lambda_i = e^{x_i^T\beta}$$

We're going to work with the dataset `tea_discipline_oss.csv`, a dataset gathered by Texas Appleseed, looking at the number of out of school suspensions (ACTIONS) accross schools in Texas. The data is censored for privacy reasons – data points with fewer than 5 actions are given the code "-99". For now, we're going to exclude these data points.

Exercise 3.6 *We're going to use a Poisson model on the counts. Ignoring the fact that the data is censored, why is this not quite the right model? Hint: there are several answers to this – the most fundamental involve considering the support of the Poisson.*

Proof: Looking at the bar chart for number of suspensions, I find the right tail to be extremely heavy. There are a few pal whose scores are very high.



■

Exercise 3.7 *Let's assume our only covariate of interest is GRADE*[2] *and put a normal prior on* β*. Using a Laplace approximation and an appropriately vague prior, find 95% marginal credible intervals for the entries of* β*. You'll probably want to use an intercept.*

Proof: Let $p_i \equiv g^{-1}(x_i'\beta) = e^{x_i'\beta}$, and $p(\beta|w) = N(0, (wI)^{-1})$:

$$\begin{aligned} p(\beta|y,x,w) = \frac{p(\beta,y|w,x)}{p(y|w,x)} &\propto p(y|\beta,w,x)p(\beta|w,x) \\ &\propto exp\left(-\frac{w}{2}\beta'\beta)\right)\prod_i \frac{\lambda_i^{y_i}e^{-\lambda_i}}{y_i!} \\ &\propto exp\left(-\frac{w}{2}\beta'\beta)\right)\prod_i \frac{e^{y_i x_i'\beta}e^{-e^{x_i'\beta}}}{y_i!} \end{aligned}$$

Taking log:

$$l(\beta|y,x,w) \propto -\frac{w}{2}\beta'\beta + \sum_i y_i x_i'\beta - e^{x_i'\beta}$$

Taking first and second derivatives:

$$\begin{aligned} \frac{\delta l(\beta)}{\delta\beta} &\propto -w\beta + \sum y_i x_i - \sum e^{x_i'\beta}x_i = -w\beta + (y - e^{X'\beta})'X \\ \frac{\delta^2 l(\beta)}{\delta\beta\delta\beta'} &\propto -wI - \sum_i e^{x_i'\beta}x_i x_i' \end{aligned}$$

[2] I have manually replaced Kindergarten and Pre-K with Grades 0 and -1, respectively.

Assume $w = 1$. Using Python's Scipy optimizer:

$$\beta_{map} = x^* = [2.3894, 0.0503]$$

$$C = \begin{bmatrix} 333177.6 & 2576058.9 \\ 2576058.9 & 22480443.9 \end{bmatrix} \qquad C^{-1} = \Sigma = \begin{bmatrix} 2.6327e^{-5} & -3.0168e^{-6} \\ -3.0168e^{-6} & 3.9018e^{-7} \end{bmatrix}$$

95% C.I. for $\beta_0 = (2.37930334, 2.39941633)$. 95% C.I. for $\beta_1 = (0.04903924, 0.05148781)$. ■

Exercise 3.8 (Optional) *Repeat the analysis using a set of variables that interest you.*

Proof: Set x_0 to be intercept, x_1 to be GRADE, x_2 to be dummy variable if male, x_3 to be dummy variable if Hispanic or African American. Assume $w = 1$. Here are my results:

$$\beta_{map} = x^* = [1.6599, 0.0559, 0.1494, 0.6783]$$

$$diag(\Sigma) = [0.00817687, 0.00062906, 0.00392903, 0.00558541]$$

95% C.I. for $\beta_0 = (1.644, 1.676), \beta_1 = (0.055, 0.057), \beta_2 = (0.142, 0.157), \beta_3 = (0.667, 0.689)$. ■

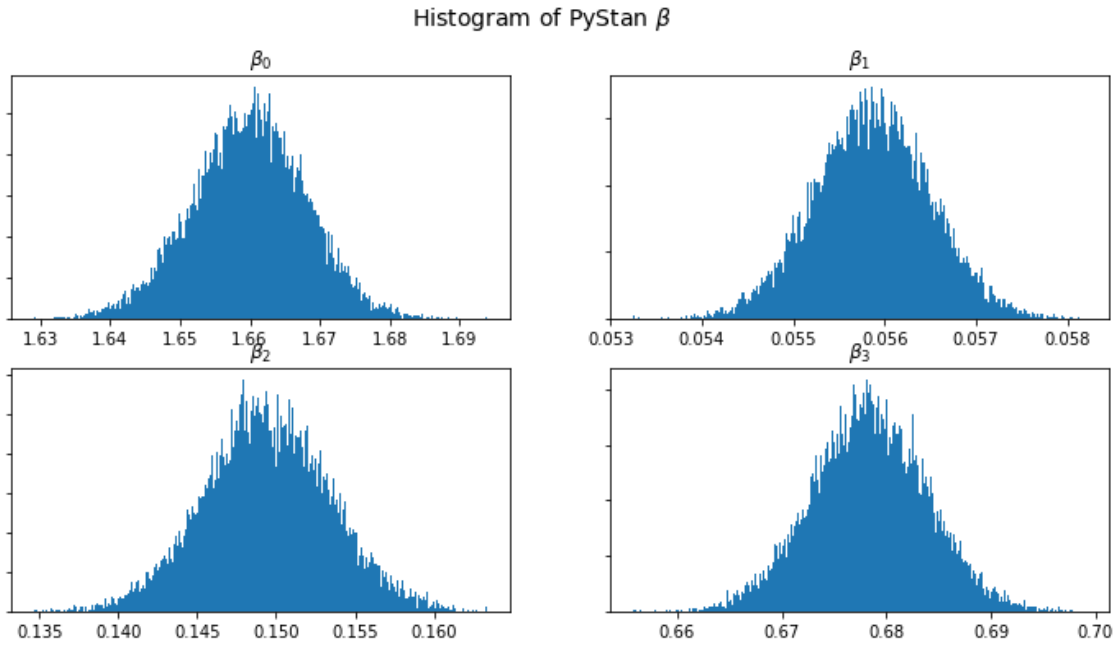
Even though we don't have conjugacy, we can still use MCMC methods – we just can't use our old friend the Gibbs sampler. Since this isn't an MCMC course, let's use STAN, a probabilistic programming language available for R, python and Matlab. I'm going to assume herein that we're using RStan, and give appropriate scripts; it should be fairly straightforward to use if you're an R novice, or if you want to use a different language, there are hints on translating to PyStan at `http://pystan.readthedocs.io/en/latest/differences_pystan_rs` and info on MatlabStan (which seems much less popular) at `http://mc-stan.org/users/interfaces/matlab-stan`.

Exercise 3.9 *Download the sample STAN script* ***`poisson.stan`*** *and corresponding R script* ***`run_poisson_stan.R`****. The R script should run the regression vs GRADE from earlier (feel free to change the prior parameters). Run it and see how the results differ from the Laplace approximation. Modify the script to include more variables, and present your results.*

Proof: Implement the same model and identical priors as in exercise 3.8. The resulting posterior mean and variance for β by STAN simulation is almost identical with results from exercise 3.8.

$$\bar{\beta} = [1.6599, 0.0559, 0.1494, 0.6784]$$

$$\text{sampling } \sigma = [0.00811663, 0.0006311, 0.00398475, 0.00548853]$$



■

Exercise 3.10 *Consider ways you might improve your regression (still, using the censored data) - while staying in the GLM framework. Ideas might include hierarchical error modeling (as we looked at in the last set of exercises), interaction terms... or something else! Looking at the data may give you inspiration. Implement this in STAN.*

Proof: The graph from exercise 3.6 shows logged value of suspensions has distribution akin to exponential distribution. Thus, my model is:

$$z \equiv ln(y), \qquad f(z_i) = \theta_i e^{-\theta_i z_i}, \qquad \theta_i = e^{x_i'\beta}$$

$$E_Y[y] = E_Z[e^z] = \frac{\theta}{\theta - 1} \qquad \text{if } \theta > 1 \ (x'\beta > 0)$$

Here are the parameters that I got:

$$\bar{\beta} = [-0.56011803, -0.01831332, -0.04181387, -0.18344612]$$

Thus, $\theta_i < 1 \ \forall \ i$. This shows that y has a fat-tail distribution with $E[y] = \infty$. Instead of letting $\hat{y}_i = E[y_i]$, let $\hat{y}_i = e^{E[z_i]}$. This way, we can have tractable statistics. Compare with this model from Exercise 3.8 (and 3.9), the RMSE is higher. However, when y is log-transformed, i.e., use z, the exponential model has lower RMSE [3].

Model Type	y RMSE	ln(y) RMSE
Poisson	3048.7	114.3
Exponential	3135.6	103.5

[3]Note that for the Poisson model, the "correct" way is to compute $E[ln(y_i)]$, but I computed $ln(E[y_i])$ instead.

Here is the STAN code:

```
data {
  // Define variables in data
  // Number of observations (an integer)
  int<lower=0> N;
  // Covariates
  int <lower=0, upper=1> intercept[N];
  int <lower=-1, upper=12> grade[N];
  int <lower=0, upper=1> male[N];
  int <lower=0, upper=1> race[N];
  // Count outcome
  real <lower=0> y[N]; // logged value of y
}
parameters {
  // Define parameters to estimate
  real beta[4];
}
transformed parameters  {
  real theta[N];
  real<lower=0> alpha[N];
  for (i in 1:N) {
    theta[i] = beta[1] + beta[2]*grade[i] + beta[3]*male[i] + beta[4]*race[i];
    alpha[i] = exp(theta[i]);
  }
}
model {
  beta[1]~normal(0,1);
  beta[2]~normal(0,1);
  beta[3]~normal(0,1);
  beta[4]~normal(0,1);
  y ~ exponential(alpha);
}
```

■

Exercise 3.11 *We are throwing away a lot of information by not using the censored data. Come up with a strategy, and write down how you would alter your model/sampler. Bonus points for actually implementing it in STAN (hint: look up the section on censored data in the STAN manual).*

Proof: Integrating out censored values. Let M be the number of censored observations, L=4 be the upper bound for censored values, and use the same Poisson regression model as in exercise 3.8 and 3.9:

$$ln\left(\prod_{i}^{M} P(y_i <= L)\right) = \sum_{i} \text{Poisson logged cdf}(L|\theta_i)$$

Add the value to target likelihood in STAN. After including the censored data, I got different estimates of β:

$$\bar{\beta} = [0.4830, 0.0844, 0.2568, 0.8631]$$

$$\text{sampling } \sigma = [0.00630206, 0.00047321, 0.00330757, 0.00428372]$$

The intercept coefficient is now much lower and coefficients for grade, male, and race are now higher.

```
data {
  // Define variables in data
  // Number of observations (an integer)
  int<lower=0> N;
  int<lower=0> N_cut;
  // Covariates
  int <lower=-1, upper=12> grade[N];
  int <lower=0, upper=1> male[N];
  int <lower=0, upper=1> race[N];
  // Count outcome
  int <lower=0> y[N-N_cut]; // logged value of y
  int <lower=0, upper=4> L;
}
parameters {
  real beta[4];
}
transformed parameters  {
  real theta[N];
  real<lower=0> alpha[N];
  real<lower=0> a1[N-N_cut];
  real<lower=0> a0[N_cut];
  for (i in 1:N) {
    theta[i] = beta[1] + beta[2]*grade[i] + beta[3]*male[i] + beta[4]*race[i];
    alpha[i] = exp(theta[i]);
  }
  a0 = alpha[1:N_cut];
  a1 = alpha[N_cut+1:N];
}
model {
  beta[1]~normal(0,1);
  beta[2]~normal(0,1);
  beta[3]~normal(0,1);
  beta[4]~normal(0,1);
  y ~ poisson(a1);
  for (i in 1:N_cut){
      target += poisson_lcdf(L | a0[i]);
  }
}
```

■