Lab 8: Bayesian generalized linear regression models

Turning in solutions

This lab is part of Homework 6. Solutions to the exercises, as well as the non-lab homework exercises are to be written up and uploaded to Gradescope as a PDF.

Getting started

You will need the following R packages. If you do not already have them installed, please do so first using the `install.packages` function.

```
require(tidyverse)
require(rstanarm)
require(magrittr)
library(tidyverse)
library(ggplot2)
require(loo)
require(bayesplot)
require(caret)
library(rstan)
require(HSAUR3)
```

Linear Regression

An experiment was run where clouds were seeded with silver iodide to examine whether increased rainfall would occur after the seeding. In the experiment, the "treatment variable" is whether or not the cloud was seeded, and the response is the amount of rainfall. The study data also include other covariates: the suitability criterion (sne), the percentage cloud cover in the experimental area, the pre-wetness/total rainfall in the target area one hour before seeding, the echo-motion (stationary or moving), and the number of days after the first day of the experiment (time).

```
data("clouds", package = "HSAUR3")
head(clouds)
```

```
##   seeding time sne cloudcover prewetness echomotion rainfall
## 1      no    0 1.8       13.4      0.274 stationary     12.8
## 2     yes    1 2.7       37.9      1.267     moving      5.5
## 3     yes    3 4.1        3.9      0.198 stationary      6.3
## 4      no    4 2.4        5.3      0.526     moving      6.1
## 5     yes    6 4.2        7.1      0.250     moving      2.5
## 6      no    9 1.6        6.9      0.018 stationary      3.6
```

Frequentist Approach

We will use a linear regression model to predict rainfall. We will include interactions of all covariates with seeding with the exception of the time variable. In the usual frequentist setting, recall that the OLS predictor for β is $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$, where $\boldsymbol{X}$ is the design matrix of predictors. We can fit the model as follows:

```
ols <- lm(rainfall ~ seeding * (sne + cloudcover + prewetness + echomotion) + time,
          data = clouds)
summary(ols)
```

```
##
## Call:
## lm(formula = rainfall ~ seeding * (sne + cloudcover + prewetness +
##     echomotion) + time, data = clouds)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -2.53  -1.15  -0.27   1.04   4.39
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      -0.3462     2.7877   -0.12   0.9031
## seedingyes                       15.6829     4.4463    3.53   0.0037
## sne                               0.4198     0.8445    0.50   0.6274
## cloudcover                        0.3879     0.2179    1.78   0.0984
## prewetness                        4.1083     3.6010    1.14   0.2745
## echomotionstationary              3.1528     1.9325    1.63   0.1268
## time                             -0.0450     0.0251   -1.80   0.0959
## seedingyes:sne                   -3.1972     1.2671   -2.52   0.0254
## seedingyes:cloudcover            -0.4863     0.2411   -2.02   0.0648
## seedingyes:prewetness            -2.5571     4.4809   -0.57   0.5780
## seedingyes:echomotionstationary  -0.5622     2.6443   -0.21   0.8349
##
## Residual standard error: 2.2 on 13 degrees of freedom
## Multiple R-squared:  0.716,  Adjusted R-squared:  0.497
## F-statistic: 3.27 on 10 and 13 DF,  p-value: 0.0243
```

---

1. Interpret the significant coefficients (at the 0.05 significance level).

Bayesian Approach

To run this model using Bayesian estimation, we can use the `rstanarm` package. We already covered estimation and writing the Gibbs sampler in class but let's see how to do the same using Stan. The `rstanarm` function equivalent of `lm()` is `stan_lm()`, however, we can also use the `stan_glm()` function.

Prior Specification: β Coefficients

The `stan_glm()` function allows us to fit a linear model if we specify the `family` parameter to be `gaussian()`. In class we used a normal prior for the vector of coefficients β. As an alternative, here we will use the Cauchy distribution which has a taller peak than the normal and also has fatter tails that decay more slowly. Can you think of why that might be useful or desirable? The following code places independent Cauchy priors on the intercept and remaining predictors via the `prior_intercept` and `prior` arguments respectively.

```
beta0.prior <- cauchy()
beta.prior <- cauchy()
stan.glm <- stan_glm(data = clouds,
                     formula = rainfall ~ seeding*(sne + cloudcover + prewetness + echomotion) + time,
                     family = gaussian(),
```

```
                  prior = beta.prior,
                  prior_intercept = beta0.prior,
                  refresh = 0,
                  refresh = 0)
```

```
## Warning: There were 12 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quanti
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

---

2. How do the estimated coefficients compare in this glm model to those from the model fit using `lm`? You can use the `summary` function on the `stan.glm` object to see the posterior summaries.
3. How do the credible intervals and standard errors of the coefficients compare to the confidence intervals and standard errors from the model fit using `lm`?

Logistic Regression

Now that we've used the `stan_glm()` function, you might be wondering if you can fit other GLMs with a Bayesian model. Yes we can! The `stan_glm()` function supports every link function that `glm()` supports. We will fit a logistic regression model in the next example.

Suppose we are interested in how an undergraduate student's GRE, GPA, and college prestige ranking affect their admission into graduate school. The response variable is whether or not the student was admitted to graduate school.

```
seed <- 196
admissions <- read.csv("https://stats.idre.ucla.edu/stat/data/binary.csv")
## view the first few rows of the data
head(admissions)
```

```
##   admit gre gpa rank
## 1     0 380 3.6    3
## 2     1 660 3.7    3
## 3     1 800 4.0    1
## 4     1 640 3.2    4
## 5     0 520 2.9    4
## 6     1 760 3.0    2
```

```
admissions$rank <- factor(admissions$rank)
admissions$admit <- factor(admissions$admit)
admissions$gre <- scale(admissions$gre)
p <- 5
n <- nrow(admissions)
```

Frequentist Approach

```
freq.mod <- glm(admit ~. , data = admissions,
                family = binomial())
```

```
summary(freq.mod)
```

```
##
## Call:
## glm(formula = admit ~ ., family = binomial(), data = admissions)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -2.659      1.166   -2.28   0.0225
## gre            0.262      0.126    2.07   0.0385
## gpa            0.804      0.332    2.42   0.0154
## rank2         -0.675      0.316   -2.13   0.0328
## rank3         -1.340      0.345   -3.88   0.0001
## rank4         -1.551      0.418   -3.71   0.0002
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 458.52  on 394  degrees of freedom
## AIC: 470.5
##
## Number of Fisher Scoring iterations: 4
```

---

4. Interpret the significant coefficients (at the 0.05 significance level).

Weakly Informative Prior: Normal

We have the choice of a logit or probit link. You should already know what probit and logistic regressions are. If you do not, please do a quick review of the frequentist versions. We should cover Bayesian logistic regression next week but we may or may not get to Bayesian probit regression. With `stan_glm()`, binomial models with a logit link function can typically be fit slightly faster than the identical model with a probit link because of how the two models are implemented in Stan. In the following code, we simply specify the chosen link, and set priors for the intercept and the predictor coefficients.

```
post1 <- stan_glm(admit ~ ., data = admissions,
                  family = binomial(link = "logit"),
                  prior = normal(0,1), prior_intercept = normal(0,1),
                  seed = seed,
                  refresh = 0)
```

---

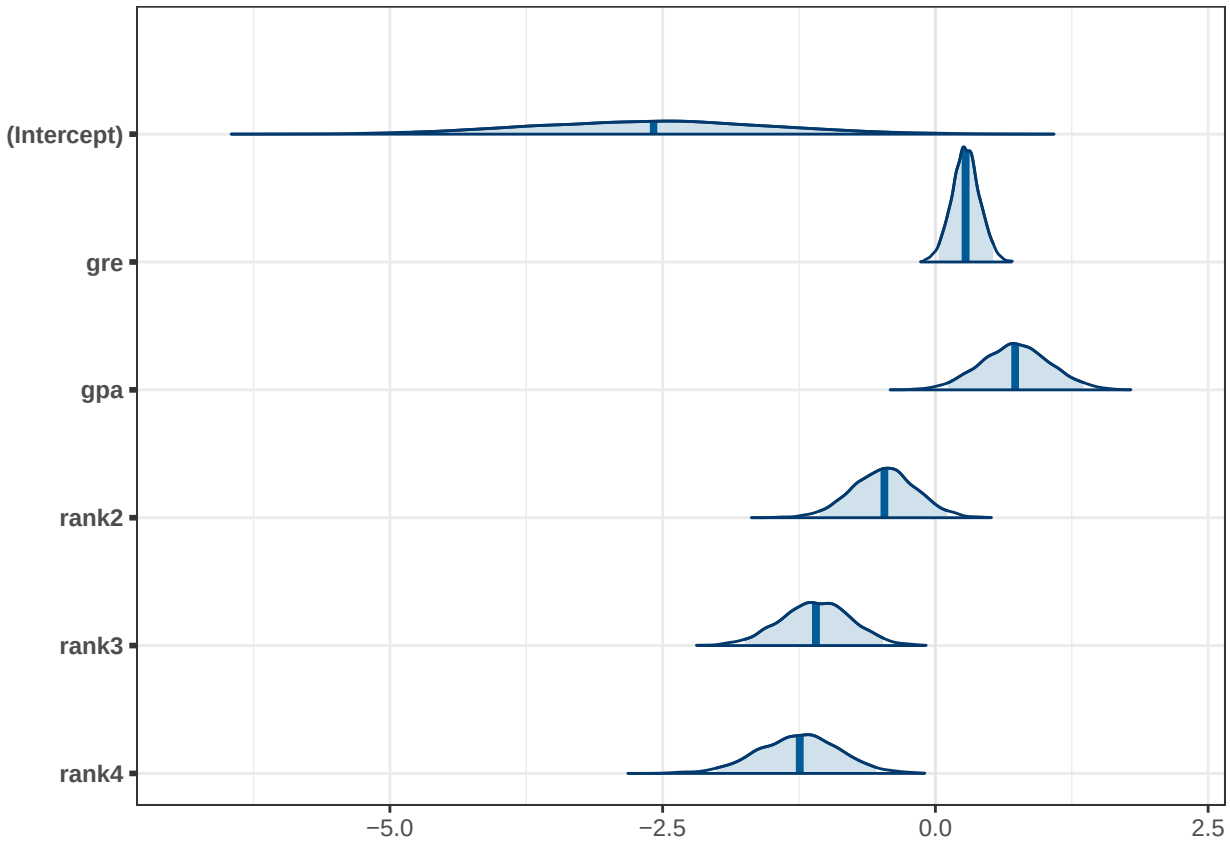
5. What do our choice of priors say about our beliefs? How do we interpret these normal priors?

As always, it is good practice to run diagnostics to check model convergence. Here is a nice function that will allow you to do this without having to call many different functions:

```
launch_shinystan(post1)
```

Now we can look at posterior densities and estimates for the coefficients.

```
mcmc_areas(as.matrix(post1), prob = 0.95, prob_outer = 1)
```




```
round(coef(post1), 3)
```

```
## (Intercept)         gre         gpa       rank2       rank3       rank4
##       -2.58        0.28        0.73       -0.47       -1.09       -1.24
```

```
round(posterior_interval(post1, prob = 0.95), 3)
```

```
##                2.5% 97.5%
## (Intercept) -4.787 -0.43
## gre          0.029  0.53
## gpa          0.114  1.36
## rank2       -1.061  0.11
## rank3       -1.748 -0.48
## rank4       -2.009 -0.54
```

---

6. How do the estimated coefficients compare in this model to those from the model fit using `glm`?
7. How do the credible intervals and standard errors of the coefficients compare to the confidence intervals and standard errors from the model fit using `glm`?

Posterior Predictive Checks

```
(loo1 <- loo(post1, save_psis = TRUE))
```

```
##
## Computed from 4000 by 400 log-likelihood matrix.
```

```
##
##          Estimate   SE
## elpd_loo   -235.3  8.6
## p_loo         5.6  0.3
## looic       470.6 17.1
## ------
## MCSE of elpd_loo is 0.0.
## MCSE and ESS estimates assume independent draws (r_eff=1).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

In the code chunk above, we assessed the strength of our model via its posterior predictive LOOCV. However as we know, this accuracy rate is quite meaningless unless we have something to compare it to. So let's create a baseline model with no predictors to compare to this first model:

```
post0 <- stan_glm(admit ~ 1, data = admissions,
                  family = binomial(link = "logit"),
                  prior = normal(0,1), prior_intercept = normal(0,1),
                  seed = seed,
                  refresh = 0)
(loo0 <- loo(post0, save_psis = T))
```

```
##
## Computed from 4000 by 400 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo   -250.9  7.1
## p_loo         0.9  0.0
## looic       501.8 14.2
## ------
## MCSE of elpd_loo is 0.0.
## MCSE and ESS estimates assume independent draws (r_eff=1).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
rstanarm::loo_compare(loo0, loo1)
```

```
##       elpd_diff se_diff
## post1   0.0       0.0
## post0 -15.6       5.6
```

---

8. Which model is better? Why?

Below, we compute posterior predictive probabilities of the linear predictor via the `posterior_linpred()` function provided in the rstanarm package. This function will extract posterior draws from the linear predictor. If we used a link function, then specifying the transform argument as `TRUE` will return the predictor as transformed via the inverse-link.

```
preds <- posterior_linpred(post1, transform=TRUE)
```

```
## Instead of posterior_linpred(..., transform=TRUE) please call posterior_epred(), which provides equi
```

```
pred <- colMeans(preds)
```

We calculate these posterior predictive probabilities in order to determine the classification accuracy of our model. If the posterior probability of success for an individual is greater or equal to 0.5, then we would predict that observation to be a success (and similarly for less than 0.5). For each observation, we can compare the posterior prediction to the actual observed value. The proportion of times we correctly predict an individual (i.e. [prediction = 0 and observation = 0] or [prediction = 1 and observation = 1]) is our classification accuracy.

```
pr <- as.integer(pred >= 0.5)
round(mean(xor(pr,as.integer(admissions$admit==0))),3)
```

```
## [1] 0.7
```

The Horseshoe Prior

In the case when we have more variables than observations, it will be difficult to achieve good estimates of the coefficients. To address this hurdle, we may consider alternative priors on the βs which place higher prior density on 0, effectively saying that those predictors should not be included in our final model. The horseshoe prior (`hs()`) is one such "shrinkage" prior. We will not spend time on the horseshoe prior beyond this, so consider this a very brief introduction. If you would like to know more, let the instructor know. In this dataset we have many samples and few covariates, so the horseshoe is not necessary. However, we will examine its effect on posterior inference of the β coefficients.

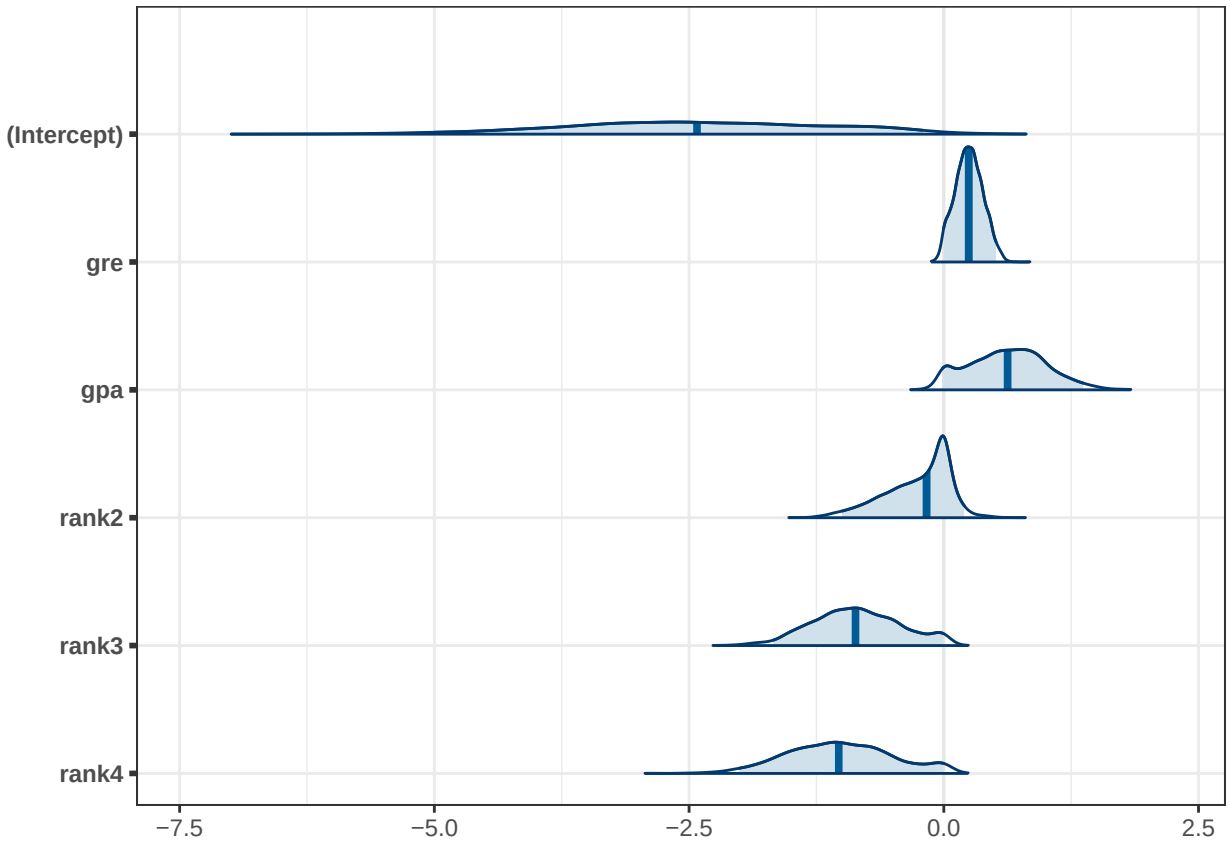
```
p0 <- 2 # prior guess for the number of relevant variables
tau0 <- p0/(p-p0) * 1/sqrt(n) # recommended by Pilronen and Vehtari (2017)
hs_prior <- hs(df=1, global_df=1, global_scale=tau0)
post2 <- stan_glm(admit ~ ., data = admissions,
                  family = binomial(link = "logit"),
                  prior = hs_prior, prior_intercept = normal(0,1),
                  seed = seed,
                  refresh = 0)

round(coef(post2), 3)
```

```
## (Intercept)         gre         gpa       rank2       rank3       rank4
##       -2.42        0.24        0.63       -0.17       -0.87       -1.03
```

```
round(posterior_interval(post2, prob = 0.95), 3)
```

```
##                2.5%  97.5%
## (Intercept) -4.995 -0.209
## gre         -0.001  0.512
## gpa         -0.013  1.367
## rank2       -0.998  0.200
## rank3       -1.673 -0.007
## rank4       -2.007 -0.002
```

```
mcmc_areas(as.matrix(post2), prob = 0.95, prob_outer = 1)
```



9. How does posterior inference for the coefficients compare to when we used the weakly informative Normal prior above?
10. How do the two models compare in terms of predictive performance? Consider using the `loo` function as we have been doing.

```
(loo2 <- loo(post2, save_psis = T))
```

```
##
## Computed from 4000 by 400 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo   -238.2  8.2
## p_loo         7.3  0.4
## looic       476.4 16.3
## ------
## MCSE of elpd_loo is 0.0.
## MCSE and ESS estimates assume independent draws (r_eff=1).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
rstanarm::loo_compare(loo1, loo2)
```

```
##       elpd_diff se_diff
## post1  0.0       0.0
## post2 -2.9       1.0
```

Acknowledgement

This lab was adapted from tutorial-1 and tutorial-2 by Jordan Bryan and Becky Tang.