ISYE 6420: Homework 5

aelhabr3

1. Cross-validating a Bayesian Regression.

Instructions

In this excercise…

Response

```
library(tidyverse)
```

Below we create the data according to the instructions.

```
set.seed(42)
n <- 40
x1 <- runif(n)
x2 <- floor(10 * runif(n)) + 1
y <- 2 + 6 * x1 - 0.5 * x2 + rnorm(n)
x1 <- round(x1, 4)
y <- round(y, 4)
x1
```

```
##  [1] 0.9148 0.9371 0.2861 0.8304 0.6417 0.5191 0.7366 0.1347 0.6570 0.7051
## [11] 0.4577 0.7191 0.9347 0.2554 0.4623 0.9400 0.9782 0.1175 0.4750 0.5603
## [21] 0.9040 0.1387 0.9889 0.9467 0.0824 0.5142 0.3902 0.9057 0.4470 0.8360
## [31] 0.7376 0.8111 0.3881 0.6852 0.0039 0.8329 0.0073 0.2077 0.9066 0.6118
```

```
x2
```

```
##  [1]  4  5  1 10  5 10  9  7 10  7  4  4  4  8  1  8  7  2  3  6  7 10  8
## [24]  6  9  2  3  9  7  3  1  2  3  5  2  8  1  4  6  1
```

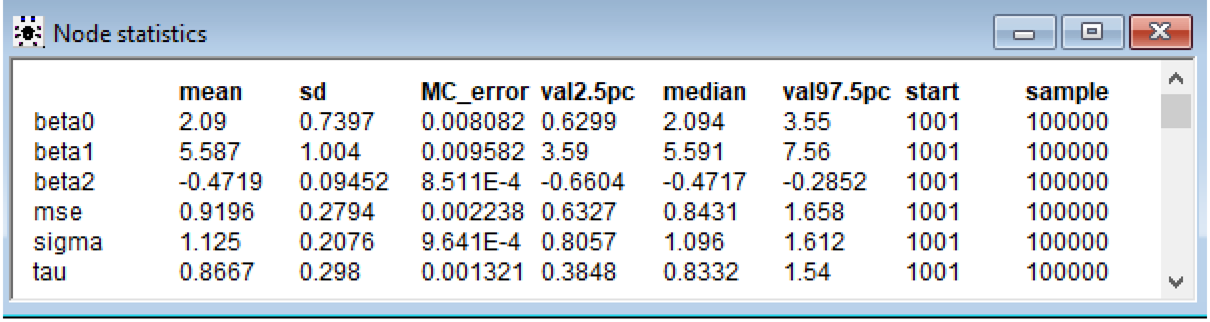
```
y
```

```
##  [1]  5.6948  4.7614  3.9750  1.2560  1.9822  0.5474  1.1081  0.7521
##  [9]  0.5105  3.3860  3.0684  3.5308  7.1838  0.1755  4.3635  3.9166
## [17]  5.0486  1.7948  0.3569  2.6469  3.5570 -1.9825  4.5152  6.0797
## [25] -2.7327  5.3878  3.1771  3.9729  2.1025  6.2369  4.8825  5.7761
## [33]  3.4522  2.6575  0.4809  3.5785  2.3122  1.7097  3.5538  4.0709
```

The OpenBUGs code looks as follows.

```
model{
for(i in 1:k) {
  mu[i] <- beta0 + beta1 * x1[i] + beta2 * x2[i]
  y[i] ~ dnorm(mu[i], tau)
}
for(i in k+1:n) {
  ypred[i] <- beta0 + beta1 * x1[i] + beta2 * x2[i]
  error[i] <- ypred[i] - y[i]
  se[i] <- error[i] * error[i]
}
  mse <- mean(se[k+1:n])
  beta0 ~ dnorm(0, 0.001)
  beta1 ~ dnorm(0, 0.001)
  beta2 ~ dnorm(0, 0.001)
  tau ~ dgamma(0.001, 0.001)
  sigma <- 1 / sqrt(tau)
}

# data
list(
  n = 40,
  k = 20,
  x1 = c(0.9148, 0.9371, 0.2861, 0.8304, 0.6417, 0.5191, 0.7366, 0.1347, 0.657, 0.7051, 0.4577, 0.7191, 0.93
47, 0.2554, 0.4623, 0.94, 0.9782, 0.1175, 0.475, 0.5603, 0.904, 0.1387, 0.9889, 0.9467, 0.0824, 0.5142, 0.39
02, 0.9057, 0.447, 0.836, 0.7376, 0.8111, 0.3881, 0.6852, 0.0039, 0.8329, 0.0073, 0.2077, 0.9066, 0.6118),
  x2 = c(4, 5, 1, 10, 5, 10, 9, 7, 10, 7, 4, 4, 4, 8, 1, 8, 7, 2, 3, 6, 7, 10, 8, 6, 9, 2, 3, 9, 7, 3, 1, 2,
3, 5, 2, 8, 1, 4, 6, 1),
  y = c(5.6948, 4.7614, 3.975, 1.256, 1.9822, 0.5474, 1.1081, 0.7521, 0.5105, 3.386, 3.0684, 3.5308, 7.1838,
0.1755, 4.3635, 3.9166, 5.0486, 1.7948, 0.3569, 2.6469, 3.557, -1.9825, 4.5152, 6.0797, -2.7327, 5.3878, 3.1
771, 3.9729, 2.1025, 6.2369, 4.8825, 5.7761, 3.4522, 2.6575, 0.4809, 3.5785, 2.3122, 1.7097, 3.5538, 4.0709)
)

# inits
list(beta0 = 0, beta1 = 0, beta2 = 0, tau = 1)
```

The results are as follows.



```
          mean     sd       MC_error  val2.5pc  median   val97.5pc  start   sample
beta0     2.09     0.7397   0.008082  0.6299    2.094    3.55       1001    100000
beta1     5.587    1.004    0.009582  3.59      5.591    7.56       1001    100000
beta2     -0.4719  0.09452  8.511E-4  -0.6604   -0.4717  -0.2852    1001    100000
mse       0.9196   0.2794   0.002238  0.6327    0.8431   1.658      1001    100000
sigma     1.125    0.2076   9.641E-4  0.8057    1.096    1.612      1001    100000
tau       0.8667   0.298    0.001321  0.3848    0.8332   1.54       1001    100000
```

We observe that the Bayesian estimates of β_0, β_1, and β_2 are relatively close to their true values—2.09 compared to 2 for β_0; 5.587 compared to 6 for β_2; and -0.4719 compared to -0.5 for β_2. Also, the 95% credible set for these variables include the true values. The Bayesian estimate of σ is also relatively close to its true value—1.12 compared to 0.8—although it's not quite as close.

2. Body Fat from Linear Regression.

Instructions

Excess adiposity is a…

Response

a

Per section *14.7.4 Bayesian Model Selection in Multiple Regression* of Brani's book (http://statbook.gatech.edu/statb4.pdf),

> *"Laud and Ibrahim (1995) argue that agreement of model-simulated predictions and original data should be used as a criterion for model selection. If for y_i responses $\hat{y}_{i,new}$'s are hypothetical replications according to the posterior predictive distribution of competing model parameters, then $\mathrm{LI} = \sum_{i=1}^{n} \left(\mathbb{E}\hat{y}_{i,\text{ new}} - y_i\right)^2 + \mathrm{Var}\left(\hat{y}_{i,\text{ new}}\right)$ measures the discrepancy between the observed and model-predicted data. A smaller LI is better."*

The following code implements Laud–Ibrahim (LI) Bayesian model selection. The first model—having coefficients `b1[1]`, …, `b1[6]`—is the "full" model using all predictors. This constitutes our first "suggested" model. The other five models are univariate linear regressions for each variable individually. Our model selection process, in which we will disregard the full model, will guide us in our choice for our second suggested model.

(Note that the code below is relatively different from that in the provided "BFReg.odc" file. In particular,

- `mu` is modified to be matrices instead of vectors;
- the priors for th2 `b*` coefficients are modified to be vectors instead of scalars;

- the notation of the `b*` coefficients is changed such that `b1` refers to the coefficients of the first model, `b2` to the coefficients of the second model, etc.;
- code to compute the LI statistic and compare this value between models has been added.)

```
model{
  for(i in 1:N){
    # 6 models
    BB[i] <- BAI[i] * BMI[i]
    mu[1, i] <- b1[1] + b1[2] * Age[i] + b1[3] * BAI[i] + b1[4] * BMI[i] + b1[5] * BB[i] + b1[6] * Gender[i]
    mu[2, i] <- b2[1] + b2[2] * Age[i]
    mu[3, i] <- b3[1] + b3[2] * BAI[i]
    mu[4, i] <- b4[1] + b4[2] * BMI[i]
    mu[5, i] <- b5[1] + b5[2] * BB[i]
    mu[6, i] <- b6[1] + b6[2] * Gender[i]
  }

  for(i in 1:6) {
    tau[i] ~ dgamma(0, 0.001)
    LI[i] <- sqrt(sum(D2[i, ]) + pow(sd(BF.new[i, ]), 2))

    for(j in 1:N) {
      BF2[i, j] <- BF[j]
      BF2[i, j] ~ dnorm(mu[i, j], tau[i])
      BF.new[i, j] ~ dnorm(mu[i, j], tau[i])
      D2[i, j] <- pow(BF[j] - BF.new[i, j], 2)
    }
  }

  # Compare predictive criteria between models i and j
  # Comp[i,j] is 1 when LI[i]<LI[j], i-th model better.
  for (i in 1:5) {
    for (j in i+1:6) {
      Comp[i, j] <- step(LI[j] - LI[i])
    }
  }

 # priors
  for(i in 1:6) {
    b1[i] ~ dnorm(0, 0.001)
  }
  for(i in 1:2) {
    b2[i] ~ dnorm(0, 0.001)
    b3[i] ~ dnorm(0, 0.001)
    b4[i] ~ dnorm(0, 0.001)
    b5[i] ~ dnorm(0, 0.001)
    b6[i] ~ dnorm(0, 0.001)
  }
}

# DATA
list(N=3200)

 BFData

# INITS (initialize by loading one set  of tau's
#              and generating the rest of the parameters)
list(tau=c(1, 1, 1, 1, 1, 1))
```

The results are as follows.

Note that the posterior median values are the most important for model selection. Disregarding the first model—which would clearly be the best—we see that third model—based on `BAI` —has the lowest posterior median. And, when compared to the other univariate models via the `LI` calculation, this `BAI` model “wins out”. (Note that the median `LI` ’s for the third model compared to the fourth, fifth, and sixth —i.e., `LI[3, 4]` , `LI[3, 5]` , and `LI[3, 6]` —and that the `LI` of the second model when compared to the third model—i.e. `LI[1, 3]` — has a median of 0.

Thus, we conclude that the `BAI` variable is the single best predictor (according to the LI statistic).

Aside

We could have done several other things to determine the single best predictor

1. We might have also calculated and compared the R^2 statistic for each univariate model, choosing the model having the largest R^2.

2. We could have used a “classical” approach where we perform (non-Bayesian) step-wise regression (https://en.wikipedia.org/wiki/Stepwise_regression).

Regarding (2), see the implementation below. We add variables to a trivial model (with no predictors) according to which variable improves the Akaike information criterion (https://en.wikipedia.org/wiki/Akaike_information_criterion) (AIC) of the model the most. We observe from the first iteration in the output from the call to `fit_step_f_partial()` that the `BAI` variable is added first. This agrees with our finding above using the LI statistic.

```
data_q2 <-
  'q2-data.xlsx' %>%
  readxl::read_excel() %>%
  janitor::clean_names() %>%
  mutate(bb = bai * bmi)
data_q2
```

```
## # A tibble: 3,200 x 6
##      age gender   bai   bmi    bf    bb
##    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
##  1    42      1 22.81  19    17.4 433.4
##  2    23      1 21.78  17.6  26   383.3
##  3    25      1 21.53  18.8  28.7 404.8
##  4    31      1 25.55  19.1  29.2 488.0
##  5    57      1 23.18  19.9  22.8 461.3
##  6    49      1 22.78  18.8  18.8 428.3
##  7    27      1 22.78  16.8  28.8 382.7
##  8    23      1 24.49  17.9  17.8 438.4
##  9    32      1 24.09  19.7  18   474.6
## 10    26      1 23.9   18.6  25.2 444.5
## # ... with 3,190 more rows
```

```r
fmla_step_l_1 <- formula(bf ~ 1)
fmla_step_u_1 <- formula(bf ~ .)
fit_step_l_1 <- lm(fmla_step_l_1, data = data_q2)
fit_step_u_1 <- lm(fmla_step_u_1, data = data_q2)
fit_step <-
  function(fit_l,
           fit_u,
           direction = c('both', 'backward', 'forward'),
           fit = NULL,
           ...) {
    direction <- match.arg(direction)
    if (is.null(fit)) {
      if (direction == 'forward') {
        fit <- fit_l
      } else {
        fit <- fit_u
      }
    }
    step(fit, scope = list(lower = fit_l, upper = fit_u), ...)
  }

fit_step_partial <-
  purrr::partial(
    fit_step,
    fit_l = fit_step_l_1,
    fit_u = fit_step_u_1,
    ... =
  )

fit_step_f_partial <-
  purrr::partial(
    fit_step_partial,
    trace = TRUE,
    direction = 'forward',
    ... =
  )

fit_step_f_1 <- fit_step_l_1 %>% fit_step_f_partial()
```

```
## Start:  AIC=13439.96
## bf ~ 1
##
##          Df Sum of Sq    RSS   AIC
## + bai     1    116672  96588 10907
## + bb      1    101506 111754 11374
## + bmi     1     63142 150118 12318
## + gender  1     49388 163872 12599
## + age     1     17998 195262 13160
## <none>                213260 13440
##
## Step:  AIC=10907.37
## bf ~ bai
##
##          Df Sum of Sq    RSS   AIC
## + gender  1     10984  85604 10523
## + age     1      3539  93049 10790
## + bmi     1      1833  94755 10848
## + bb      1       757  95831 10884
## <none>                 96588 10907
## - bai     1    116672 213260 13440
##
## Step:  AIC=10523.04
## bf ~ bai + gender
##
##          Df Sum of Sq    RSS     AIC
## + bmi     1     27670  57933  9275.6
## + bb      1     16309  69294  9848.7
## + age     1      5497  80107 10312.7
## <none>                 85604 10523.0
## - gender  1     10984  96588 10907.4
## - bai     1     78268 163872 12599.0
##
## Step:  AIC=9275.64
## bf ~ bai + gender + bmi
##
##          Df Sum of Sq   RSS     AIC
## + bb      1      3347 54586  9087.2
## + age     1      2901 55032  9113.3
## <none>                57933  9275.6
## - bai     1       408 58341  9296.1
## - bmi     1     27670 85604 10523.0
## - gender  1     36822 94755 10848.1
##
## Step:  AIC=9087.22
## bf ~ bai + gender + bmi + bb
##
##          Df Sum of Sq   RSS     AIC
## + age     1      1758 52828  8984.5
## <none>                54586  9087.2
## - bb      1      3347 57933  9275.6
## - bai     1      3700 58287  9295.1
## - bmi     1     14708 69294  9848.7
## - gender  1     38390 92976 10789.4
##
## Step:  AIC=8984.48
## bf ~ bai + gender + bmi + bb + age
##
##          Df Sum of Sq   RSS     AIC
## <none>                 52828  8984.5
## - age     1      1758 54586  9087.2
## - bb      1      2204 55032  9113.3
## - bai     1      2499 55327  9130.4
## - bmi     1     11365 64193  9606.0
## - gender  1     38190 91018 10723.3
```

```
fit_step_f_1
```

```
##
## Call:
## lm(formula = bf ~ bai + gender + bmi + bb + age, data = data_q2)
##
## Coefficients:
## (Intercept)          bai       gender          bmi           bb
##   -32.81633      0.76747     10.60182      1.88812     -0.02379
##         age
##     0.07368
```

```
fit_step_f_1 %>% summary()
```

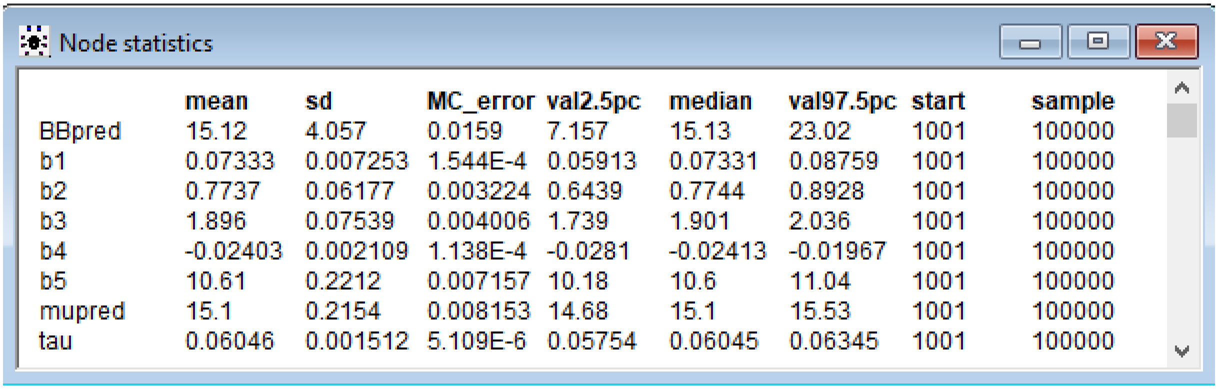
```
##
## Call:
## lm(formula = bf ~ bai + gender + bmi + bb + age, data = data_q2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.5014  -2.4774  -0.0063   2.4988  22.7272
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -32.816326   1.770358  -18.54   <2e-16 ***
## bai           0.767466   0.062440   12.29   <2e-16 ***
## gender       10.601818   0.220635   48.05   <2e-16 ***
## bmi           1.888124   0.072030   26.21   <2e-16 ***
## bb           -0.023790   0.002061  -11.54   <2e-16 ***
## age           0.073683   0.007148   10.31   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.067 on 3194 degrees of freedom
## Multiple R-squared:  0.7523, Adjusted R-squared:  0.7519
## F-statistic:  1940 on 5 and 3194 DF,  p-value: < 2.2e-16
```

b

The OpenBUGs code to make the prediction for the full model is as follows. (Note that this code is much closer to what is provided to use in “BFReg.odc”; only, the last two lines in the model scope (for prediction) are added.)

```
model{
  for(i in 1:N){
    BF[i] ~ dnorm(mu[i], tau)
    BB[i] <- BAI[i] * BMI[i]
    mu[i] <- b0 + b1 * Age[i] + b2*BAI[i] + b3*BMI[i] + b4*BB[i] + b5* Gender[i]
  }

 # priors
  b0 ~ dnorm(0, 0.001)
  b1 ~ dnorm(0, 0.001)
  b2 ~ dnorm(0, 0.001)
  b3 ~ dnorm(0, 0.001)
  b4 ~ dnorm(0, 0.001)
  b5 ~ dnorm(0, 0.001)
  tau ~ dgamma(0.001, 0.001)

  # prediction
  mupred <- b0 + b1 * (35) + b2 * (26) + b3 * (20) + b4 * (520) + b5 * (0)
  BBpred ~ dnorm(mupred, tau)
}

# DATA
list(N=3200)

 BFData

# INITS
list(b0=1, b1=0, b2=0, b3=0, b4=0, b5=0, tau=1)
```

See the results below.

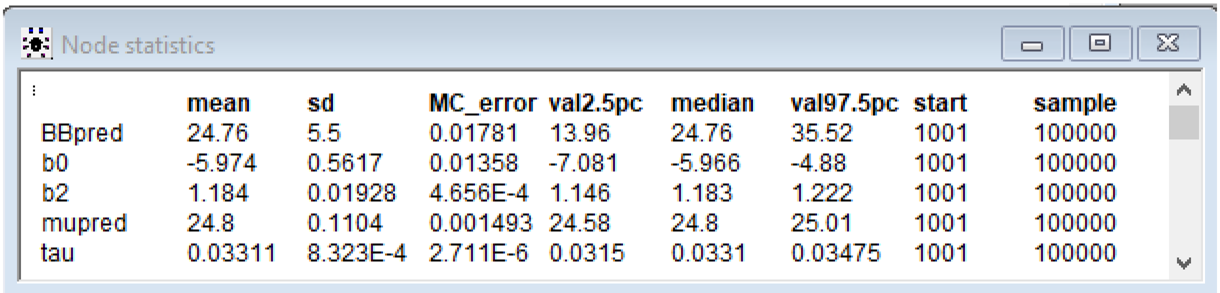


```
          mean      sd        MC_error  val2.5pc  median    val97.5pc start  sample
BBpred    15.12     4.057     0.0159    7.157     15.13     23.02     1001   100000
b1        0.07333   0.007253  1.544E-4  0.05913   0.07331   0.08759   1001   100000
b2        0.7737    0.06177   0.003224  0.6439    0.7744    0.8928    1001   100000
b3        1.896     0.07539   0.004006  1.739     1.901     2.036     1001   100000
b4        -0.02403  0.002109  1.138E-4  -0.0281   -0.02413  -0.01967  1001   100000
b5        10.61     0.2212    0.007157  10.18     10.6      11.04     1001   100000
mupred    15.1      0.2154    0.008153  14.68     15.1      15.53     1001   100000
tau       0.06046   0.001512  5.109E-6  0.05754   0.06045   0.06345   1001   100000
```

The OpenBUGs for the univariate model using `BAI` as the lone predictor is as follows. (Note that the code is nearly identical to that above; only the `mu` definitions are changed.)

```
model{
  for(i in 1:N){
    BF[i] ~ dnorm(mu[i], tau)
    mu[i] <- b0 + b2*BAI[i]
  }

 # priors
  b0 ~ dnorm(0, 0.001)
  b2 ~ dnorm(0, 0.001)
  tau ~ dgamma(0.001, 0.001)

  # prediction
  mupred <- b0 + b2 * (26)
  BBpred ~ dnorm(mupred, tau)
}

# DATA
list(N=3200)

 BFData

# INITS
list(b0=1, b2=0, tau=1)
```

See the results below.



```
         mean     sd         MC_error  val2.5pc  median   val97.5pc  start  sample
BBpred   24.76    5.5        0.01781   13.96     24.76    35.52      1001   100000
b0       -5.974   0.5617     0.01358   -7.081    -5.966   -4.88      1001   100000
b2       1.184    0.01928    4.656E-4  1.146     1.183    1.222      1001   100000
mupred   24.8     0.1104     0.001493  24.58     24.8     25.01      1001   100000
tau      0.03311  8.323E-4   2.711E-6  0.0315    0.0331   0.03475    1001   100000
```

We observe that the predictions—see the posterior `mean` of the `BBpred` variable in the tables above—are relatively different. The full model predicts 15.12 and the univariate `BAI` model predicts 24.76.

3. Shocks.

Instructions

An experiment was conducted…

Response

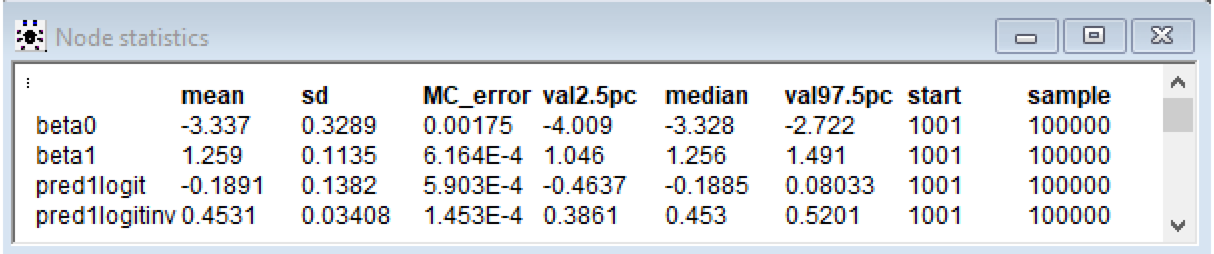
Below is OpenBUGs code for this problem. Note that:

- “noninformative priors” is implemented as `dnorm(0.0, 0.00001)` ;

- `pred1logit` is the un-transformed prediction;
- `pred1logitinv` is the transformed prediction (i.e. a value between 0 and 1);
- `x` and `response` (in the data list) is supplied “explicitly” as vectors having `n` x length(`x`) values (i.e. 70 x 6 = 420 values). The values of `response` are inferred from the number of responses y and number of trials n from the provided table.

```
model{
  for(i in 1:n) {
    response[i] ~ dbern(p[i])
    logit(p[i]) <- beta0 + beta1 * x[i]
  }
  beta0 ~ dnorm(0.0, 0.00001)
  beta1 ~ dnorm(0.0, 0.00001)

  pred1logit <- beta0 + beta1 * (2.5)
  pred1logitinv <- 1 / (1 + exp(-pred1logit))
}

# data
list(
  n = 420,
  x = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5),
  response = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0)
)

# inits
list(beta0 = 0, beta1 = 0)
```

The results are as follows.



	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
beta0	-3.337	0.3289	0.00175	-4.009	-3.328	-2.722	1001	100000
beta1	1.259	0.1135	6.164E-4	1.046	1.256	1.491	1001	100000
pred1logit	-0.1891	0.1382	5.903E-4	-0.4637	-0.1885	0.08033	1001	100000
pred1logitinv	0.4531	0.03408	1.453E-4	0.3861	0.453	0.5201	1001	100000

`pred1logitinv` represents our prediction for the proportion of responses after a shock of 2.5 milliamps. Its posterior mean is 0.4531 and its 95% credible set is [0.3861, 0.5201]. As a check on these values, we note that the posterior mean and credible set for this proportion fall between the observed proportions for 2 and 3 milliamps (0.300 and 0.671 respectively).

Aside

We can carry out the equivalent calculations using `R` and verify that our results are nearly identical.

```
.x <- 0:5
.y <- c(0, 9, 21, 47, 60, 63)
..n <- 70
.n <- rep(..n, length(.x))
.diff <- .n - .y
```

```
ones <- .y %>% purrr::map(~rep(1L, .x))
zeros <- .diff %>% purrr::map(~rep(0L, .x))
data_explicit <-
  purrr::map2(ones, zeros, c) %>%
  tibble(x = .x, response = .) %>%
  unnest(response)
```

```
fit_glm <- glm(formula(response ~ x), data = data_explicit, family = binomial())
fit_glm
```

```
##
## Call:  glm(formula = formula(response ~ x), family = binomial(), data = data_explicit)
##
## Coefficients:
## (Intercept)            x
##      -3.301        1.246
##
## Degrees of Freedom: 419 Total (i.e. Null);  418 Residual
## Null Deviance:       581.3
## Residual Deviance: 340.2     AIC: 344.2
```

```
data_new <- tibble(x = 2.5)
pred_glm_link <- predict(fit_glm, newdata = data_new, type = 'link')
pred_glm_link
```

```
##          1
## -0.1861914
```

```
pred_glm <- predict(fit_glm, newdata = data_new, type = 'response')
pred_glm
```

```
##         1
## 0.4535862
```

```
confint_pred_glm <- ciTools::add_ci(data_new, fit_glm)
confint_pred_glm
```

```
## # A tibble: 1 x 4
##       x   pred LCB0.025 UCB0.975
##   <dbl>  <dbl>    <dbl>    <dbl>
## 1   2.5 0.4536   0.3882   0.5207
```

`pred_glm` (0.4536) is nearly identical to `pred1logitinv` (0.4531) from the OpenBUGs output. Likewise, the 95% credible set defined by the `LCB0.025` and `UCB0.975` components of `confint_pred_glm` ([0.3882, 0.5207]) is nearly identical to that found in the OpenBUGs output ([0.3861, 0.5201]).