STAT 447 Assignment 8

Bayesian Workflow

Caden Hewlett

2024-03-14

Question 1: Calibration Analysis via Cross-Validation

One way to approximate the outcome of an experiment is via cross-validation. In particular, this exercise will teach you how to use leave-one-out cross-validation to assess the calibration of the model for the Hubble dataset.

Let's begin by loading and formatting the data again.

```
df = read.csv("data/hubble.csv")[1:24, c(3:4)]
colnames(df) = c("distance", "velocity")
velocity = df$velocity/1000
distance = df$distance

# get the data frame size
N_obs = nrow(df)
```

Part 1

Now, we can build our modified Stan file, called `hubble_predict`.

```
// The input data are the distances 'd' of length 'N'.
data {
  int<lower=0> N;
  vector<lower = 0>[N] d;
  vector[N] v;
  real x_pred; // held-out point
}

// The parameters accepted by the model. Our model
// accepts two parameters 'beta' which is the slope
// and 'sigma', which is the variance (heteroskedastic)
parameters {
  real beta;
  real<lower=0> sigma;
}

// The model to be estimated.
// We model the velocity 'v' to be normally distributed
```

```
// with mean 'beta * d_i'and standard deviation 'sigma'.

model {
  beta ~ student_t(3, 0, 100);
  sigma ~ exponential(0.001);
  for (i in 1:N){
    v ~ normal(beta * d[i], sigma);
  }
}


// Values Produced by the model
generated quantities {
   real y_pred = normal_rng(beta*x_pred, sigma);
}
```

Part 2

Then, we can run `sampling()` with this object, using MCMC methodology to find $\mathbb{E}\big(y_n \mid x_n, \{(x_i, y_i)\}_{i=1}^{n-1}\big)$.

```
fit = sampling(
  hubble_predict,
  seed = 1990,
  data = list(
    N  = N_obs-1,
    d = df$distance[-N_obs],
    v = df$velocity[-N_obs],
    x_pred = df$distance[N_obs]
  )
)

q1_info = extract(fit)
```

Note that the true value y_n was $1,090$km/s. The reported value of $\mathrm{C}(y_{\neg n}, \alpha = 0.80)$ is given below:

```
ci_1 = round(c(quantile( q1_info$y_pred, 0.10 ),
          quantile( q1_info$y_pred, 0.90 )), 3)

kable(ci_1, caption = "80% Credible Interval")
```

Table 1: 80% Credible Interval

	x
10%	1.049
90%	1028.611

Part 3

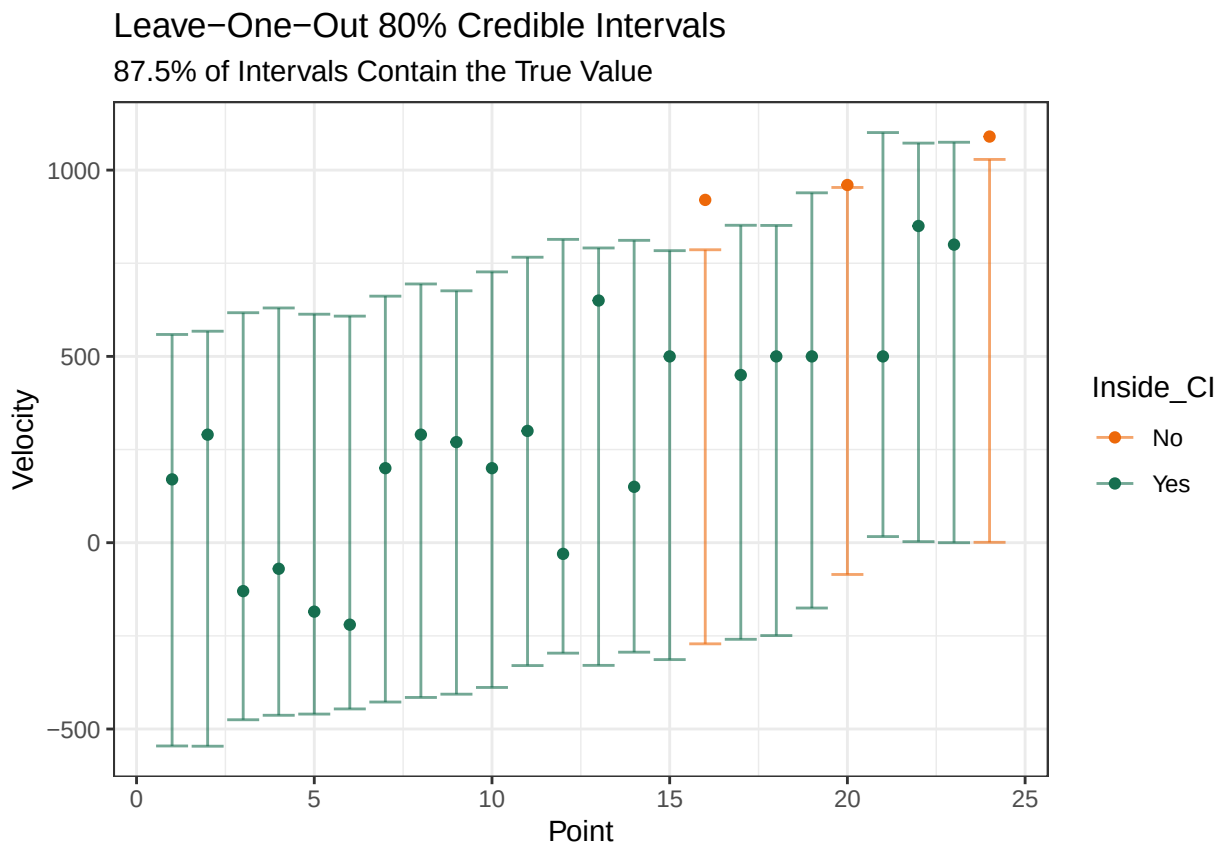
Now, we repeat this leave-one-out process for all observations. We will construct a matrix of `nrow(df)` rows and 2 columns called `ci_limits`, where the i-th row contains an 80% credible interval for the i-th observation left out.

To do this, we define a function called `CrI` which takes in an index and repeats the previous question's process:

```
CrI <- function(index){
  # compute LOO MCMC
  one_out = sampling(
    hubble_predict,
    seed = 1990,
    data = list(
      N  = N_obs-1,
      d = df$distance[-index],
      v = df$velocity[-index],
      x_pred = df$distance[index]
    ))
  # extract and compute interval
  info = extract(one_out)
  return(c(
    quantile(info$y_pred, 0.10),
    quantile(info$y_pred, 0.90)
  ))
}

## use this to make the matrix
all_loo = sapply(1:N_obs, function(n){CrI(n)})
# populate matrix
ci_limits = matrix(all_loo, nrow = nrow(df), ncol = 2, byrow = T)
```

Then the plot is below:

```
merged_df = df %>%
  bind_cols(data.frame(CI_L = ci_limits[, 1], CI_R = ci_limits[, 2])) %>%
  mutate(Inside_CI = (velocity >= CI_L & velocity <= CI_R))
plot = merged_df %>%
  ggplot(aes(
    x = 1:N_obs,
    y = velocity,
    ymin = CI_L,
    ymax = CI_R,
    color = Inside_CI
  )) +
  geom_point() +
  geom_errorbar(alpha = 0.6) +
  scale_color_manual(values = c("#ed6809", "#166e4f"),
                     labels = c("No", "Yes")) +
  theme_bw() +
  labs(
    x = "Point",
    y = "Velocity",
    title = "Leave-One-Out 80% Credible Intervals",
    subtitle = paste((sum(merged_df$Inside_CI)/N_obs)*100,
                     "% of Intervals Contain the True Value", sep = "")
  )
print(plot)
```



As the above plot shows, roughly 87.5% of the credible intervals contain the true value.

By the properties of well-specified confidence intervals, we know that:

$$\text{Well Specified Model} \implies \mathbb{P}\big(Y_n \in \mathrm{C}(Y_{\neg n})\big) = 1 - \alpha$$

Where in our case, $(1 - \alpha) = 0.80$.

Our observed proportion of intervals $\mathrm{C}(y_{\neg n})$ containing the true y_n value was 0.875. Since $0.875 \approx 0.80$, we don't currently have significant evidence to suggest the model is not well-specified. Using the language of hypothesis testing, we would fail to reject "H_0: The model is well-specified."

Question 2: Prior Predictive Checks

In this exercise we will investigate the effect of having fixed independent priors on the coefficients $\{\beta_i\}_{i=1}^{n_{\text{pred}}}$ as n_{pred} grows.

Part 1

To properly design this function, we will introduce some nomenclature. Let $\mathbf{X}$ be defined as follows:

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n_{\text{pred}}} \\ x_{2,1} & x_{2,2} & \dots & x_{2,n_{\text{pred}}} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n_{\text{obs}},1} & x_{n_{\text{obs}},2} & \dots & x_{n_{\text{obs}},n_{\text{pred}}} \end{bmatrix}_{n_{\text{obs}} \times n_{\text{pred}}}$$

Then, the vector of random normal pulls is denoted $\vec{\beta}$, where:

$$\vec{\beta} = \left\{ \beta_i \mid \beta_i \overset{\text{iid}}{\sim} N(0,1), \text{ for } i \in [1, n_{\text{pred}}] \right\}$$

Finally, the likelihood is given by:

$$\vec{\mathbf{y}} \mid \vec{\beta} \sim \text{bern}\Big(\text{logistic}\big(\mathbf{X}\vec{\beta}\big)\Big)$$

Then, letting $\vec{\mathbf{1}}$ be the vector of 1s, we let the mean of $\vec{y}$ be

$$\overline{\mathbf{y}} := n_{\text{obs}}^{-1}(\vec{\mathbf{1}} \cdot \vec{\mathbf{y}}) = \frac{1}{n_{\text{obs}}} \sum_{\{y \,:\, y \in \vec{\mathbf{y}}\}} y$$

Now that we've clarified the model, let's proceed with the design of the `logistic_regression` function.

Given $\mathbf{X}$, the function will sample $\vec{\beta}$ first before computing $\text{logistic}\big(\mathbf{X}\vec{\beta}$ and then conducting n_{obs} Bernoulli trials to create $\vec{\mathbf{y}}$ and then computing the mean of $\vec{\mathbf{y}}$. (NOTE: we know there are n_{obs} trials due to the fact that $\mathbf{X}_{n_{\text{obs}} \times n_{\text{pred}}} \times \vec{\beta}_{n_{\text{pred}} \times 1} = \vec{\mathbf{y}}_{1 \times n_{\text{obs}}}$.)

We will denote this function as follows:

$$\text{Logistic Regression} : f : \mathbb{R}^{n_{\text{obs}} \times n_{\text{pred}}} \to \mathbb{R}, \text{ where } f(\mathbf{X}) = \overline{\mathbf{y}}$$

We attempt to inherit the terminology and process described above.

```
logistic_regression <- function(X){
  # get the matrix sizes
  n_pred = ncol(X)
  n_obs = nrow(X)

  # simulate beta values
  beta = rnorm(n_pred)

  # compute the p parameter of the bernoulli distribution
  p = plogis(X%*%beta) # 1 / (1 + exp(-X%*%beta))

  # then the y values
  y = rbinom(n_obs, size = 1, prob = p)

  # then take the mean
```

```
  ybar = mean(y)

  # and return it
  return(ybar)
}
```

Part 2

Now, we let $n_{\text{obs}} = 100$ and consider $n_{\text{pred}} \in \{2, 4, 15\}$.

We will create a set of data matrices $\mathcal{X}$ defined by the following, letting $x_{i,j}$ be the (i, j) cell of $\mathbf{X}$

$$\mathcal{X} = \left\{ \mathbf{X}_{100\times n}, \text{ where } x_{i,j} \overset{\text{iid}}{\sim} \text{bern}(0.9), \text{ for } i \in [1, 100], j \in [1, n] \text{ and } n \in \{2, 4, 15\} \right\} \tag{1}$$

Then, for $M = 100,000$, we complete the simulation process, letting $\mathbf{X}_k \subset \mathcal{X}$ be the matrix in $\mathcal{X}$ with k columns. We can think of the resulting process creating a $M \times K$ matrix, where $K = |\mathcal{X}|$ denoted as follows:

$$\overline{\mathbf{Y}}_{M,K} = \begin{bmatrix} \bar{\mathbf{y}}_{1,1} & \bar{\mathbf{y}}_{1,2} & \cdots & \bar{\mathbf{y}}_{1,K} \\ \bar{\mathbf{y}}_{2,1} & \bar{\mathbf{y}}_{2,2} & \cdots & \bar{\mathbf{y}}_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{\mathbf{y}}_{M,1} & \bar{\mathbf{y}}_{M,2} & \cdots & \bar{\mathbf{y}}_{M,K} \end{bmatrix}_{M\times K} \tag{2}$$

$$\text{Where } \bar{\mathbf{y}}_{m,k} = f(\mathbf{X}_k) \text{ for } \mathbf{X}_k \in \mathcal{X}, k \in [1, K], \text{ and } m \in [1, M] \tag{3}$$

Then we plot a histogram for each column of $\overline{\mathbf{Y}}_{M,K}$. In our example there are only three columns, but the language here is designed to be extended upon if desired. The tags on the above equations have been added to direct the reader through the code.

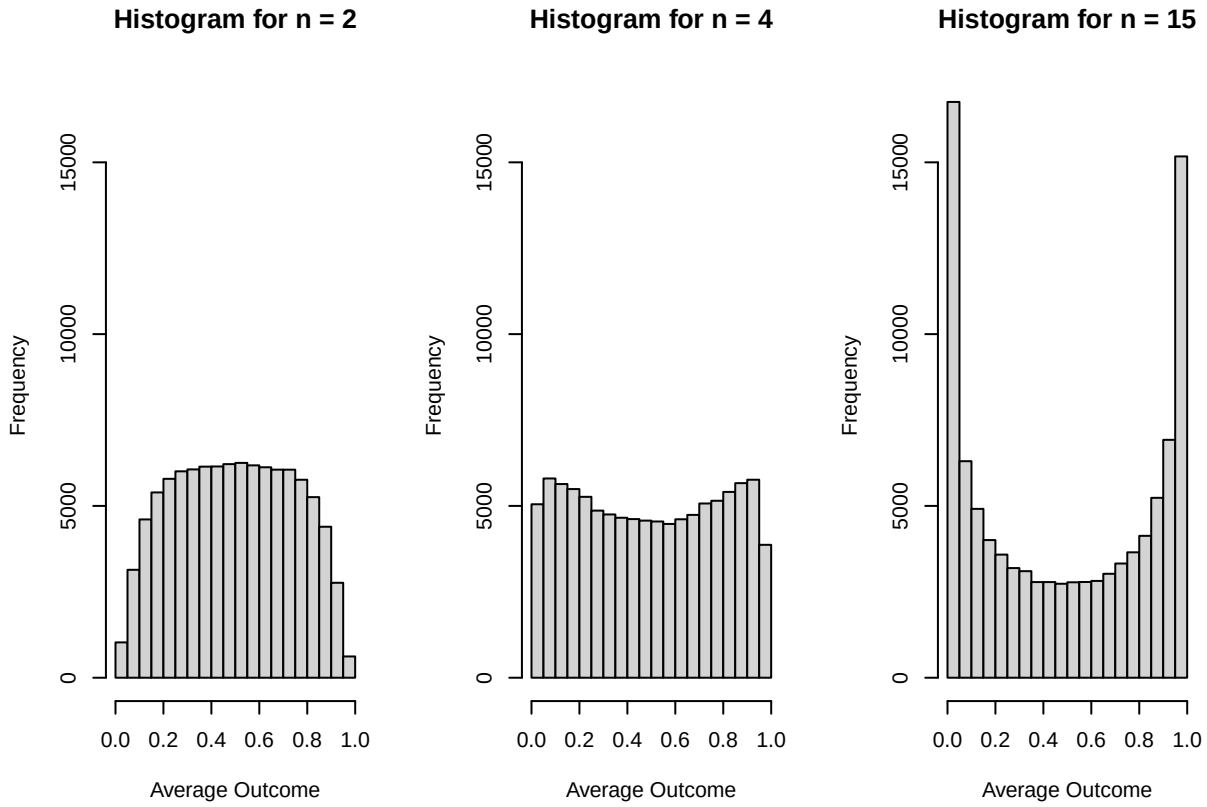
The entire simulation study is conducted below, with the $\overline{\mathbf{Y}}_{M,K}$ matrix previewed at the end.

```
set.seed(1928)
# declare parameters
M = 100000
p = 0.9
m = 100
n_pred = c(2, 4, 15)
K = length(n_pred)
# rename the logistic_regression to match the terminology provided
f <- logistic_regression
remove(logistic_regression)

# create the set of matrices described earlier: (1)
X <- sapply(n_pred, function(n){
  matrix(data = rbinom(n*m, size = 1, prob = p),
         nrow = m, ncol = n)
})

# then, create Y: k in [1, K], m in [1, M] : (2)
Y = matrix(rep(1:K, each = M), nrow = M, ncol = K, byrow = F)
# finally, populate this matrix: (3)
Ybar = apply(Y,  MARGIN = c(1,2), function(k){ f(X[[k]]) })
head(Ybar)
```

```
##      [,1] [,2] [,3]
## [1,] 0.17 0.28 0.43
## [2,] 0.74 0.84 0.12
## [3,] 0.80 0.83 0.03
## [4,] 0.42 0.24 0.99
## [5,] 0.24 0.77 0.40
## [6,] 0.43 0.06 1.00
```

Then, we can make the plots as follows (note, I added $n = 1$):

```
par(mfrow=c(1, K))
for (k in 1:K) {
  hist(Ybar[, k],  breaks=20, ylim=c(0,17000),
       main=paste("Histogram for n =", n_pred[k]), xlab="Average Outcome", ylab="Frequency")
}
```



Part 3

Do you observe any change in the distributions obtained in step 2? Yes, we see that as n_{pred} increases, the plot of average outcomes becomes increasingly concave. In this sense, there is higher frequency for the $[0.00, 0.05]$ and $[0.95, 1.00]$ as $n_{\text{pred}} \uparrow$. In running repeated tests for different n_{pred}, I found that the concavity becomes evident for $n_{\text{pred}} \geq 4$

Do you think this is desirable in a model?

If yes, why? If not, how would you fix this?