STAT 447 Assignment 8

Bayesian Workflow

Caden Hewlett

2024-03-13

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.

```r
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`.

```stan
// 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.

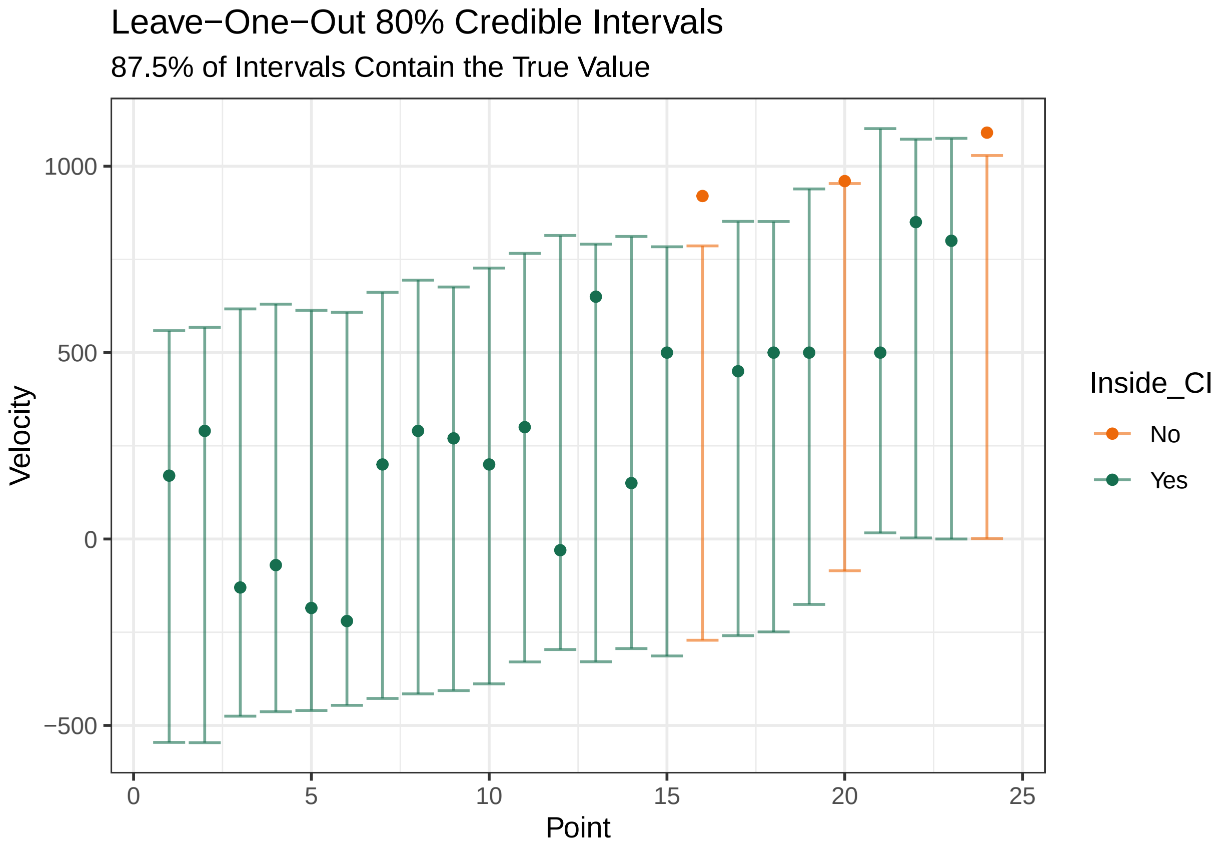
To do this, we define a function called `CrI` which takes in an index and repeats the previous question's process:

```
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."