Activity 4: Out-of-sample cross validation

In this activity, we will conduct an out-of-sample cross-validation for our Gaussian linear regression. The same procedure can be used for any type of analysis. The basic idea is that we withhold some data from model training and use the model to make predictions for those sample units so that we can assess accuracy (often referred to as "training" and "test" data sets).

Our primary aim is to provide an example of how to do a cross validation in a Bayesian context, focusing on a new "generated quantities" model block in the Stan code and new diagnostics specifically for cross-validation in the R script.

Cross validation is the gold standard for assessing how well a model performs and to compare model performance across several models, particularly when predicting the outcome y is the primary objective. See Hooten and Hobbs (2015, Ecological Monographs; available in./readings/fundamentals/) for an in-depth treatment of the topic.

Source files

1. Stan model:
   ./practicals/4_cross_validation/4_model.stan
2. R script to simulate data and run MCMC:
   ./practicals/4_cross_validation/4_script.R

The model

Our Gaussian linear regression model will take the same form as before:

$$y_i \sim Normal(\mu_i, \sigma)$$
$$\mu_i = \alpha + \sum_{k=1}^{K} \beta_k x_{k,i}$$

$$\alpha \sim Normal(0, 10)$$
$$\beta_k \sim Normal(0, 10)$$
$$\sigma \sim Uniform(0, 1e4)$$

To perform an out-of-sample cross validation, we hold out some data for a subset of sample units j from the model fitting process, and we then use the model to make predictions for those data $\hat{y}_j$ so that we can compare to the known values.

$$\hat{y}_j \sim Normal(\mu_j, \sigma)$$
$$\mu_j = \alpha + \sum_{k=1}^{K} \beta_k x_{k,j}$$

Notice that we are simply drawing our predicted values for $\hat{y}_j$ from the likelihood distribution by applying the fitted parameter values from the main fitted model to covariate values for the new sample units x_j. It is important to understand that $\hat{y}_j$ is not provided to the model as data.

Stan model

Let's take a look at how to write this model in Stan, focusing on how to implement the out-of-sample predictions. See .../4_model.stan. This model is identical to the Gaussian regression from activity 3, but we have now added a new model block called "generated quantities".

generated quantities{}

The generated quantities block is used to derive predictions from the likelihood distribution (or other stochastic quantities) from the fitted parameters in the model, often integrating new data as well. In this case, we are using it to generate predicted values $\hat{y}_j$ for our out-of-sample test data. Notice that we are using Stan's random number generator for the normal distribution *y_hat = normal_rng()* rather than applying a stochastic distribution like in the "model" block (i.e. *y ~ normal()*). This is because the generated quantities block only supports deterministic functions—this is NOT part of the statistical model, it is derived after-the-fact.

Simulate data

We provided code to simulate data for our cross-validation analysis in .../4_script.R.

This is setup like in the previous activity, except now we are producing a training dataset (like before) and a separate test dataset that we can use for cross validation.

Run MCMC and evaluate results

See .../4_script.R. The MCMC code is exactly like before, so we won't focus on it here. The model evaluation code contains an "out-of-sample cross validation" section which is our primary focus now.

We get a predicted value for each sample unit *j* for every iteration of the MCMC, so our predicted value is actually a vector of values (i.e. drawn from the posterior). This is a key distinction between Bayesian approaches and frequentist approaches. The distribution of these predicted values for a given sample unit provides a complete accounting of prediction uncertainty, and we want to make good use of this.

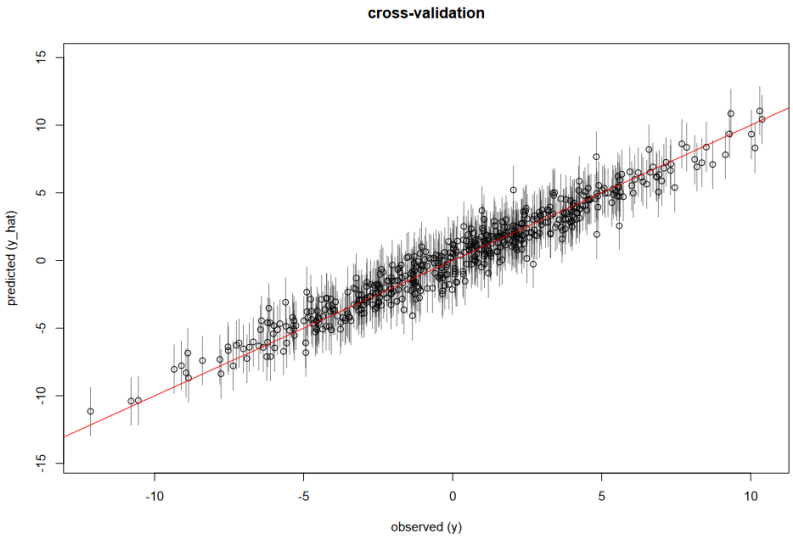
We first extract the *yhat* values as a data.frame that contains a column for each sample unit *j* in our test data and a row for each MCMC iteration.

Cross-validated bias, imprecision, inaccuracy

We calculate the residuals as $\delta_j = \frac{\hat{y}_j - y_j}{y_j}$. This gives us the prediction error for each sample unit as a proportion of the true value. Remember, the residuals ALSO contain a value for every MCMC iteration, so each one represents a probability distribution. We calculate bias $mean(\delta_j)$, imprecision $sd(\delta_j)$, and inaccuracy $mean(|\delta_j|)$ from the residuals, and we plot their distributions (and/or quantify their credible intervals). Many other fit statistics could be calculated, but these are strong general-purpose metrics.

Observed vs predicted plot

We generate an observed vs predicted plot to visually asses the fit of out-of-sample predictions. This is often the first thing to look at to quickly assess model fit.



A good fit is evident when 95% of the credible intervals (calculated for 0.95 probability) overlap the red one-to-one line. A perfect fit would be if all of the mean predictions (the dots) were exactly on the one-to-one line.

Bias would be indicated if the points and their credible intervals tended to systematically fall above or below the line.

Imprecision would be indicated if the points were often far from the line, but 95% of their credible intervals still overlapped the lines.

Coverage of prediction intervals

We can easily calculate the “coverage” of prediction intervals, and we provide this code in the script. We expect 95% of the sample units (no more, no less) to have 95% credible intervals that contain the true value. If this is not the case, then the error structure in our model is mis-specified (e.g. the prior on the standard deviation in our likelihood is too strong; we’ve used an inapropriate likelihood distribution, etc).