Examination of Cross Validation techniques and the biases they reduce.

Dr. Jon Starkweather, Research and Statistical Support consultant.

The current article continues from last month’s brief examples of how to conduct some cross validation techniques in the R statistical programming environment. This month we focus more on what cross validation does and what problems it addresses. For a slight review of the basis of cross validation techniques, consider the following paragraphs.

When building a prediction model, be it a standard regression model or a more complex model, the model should not be initially fit to the same data on which prediction error estimates are also calculated. However, due to practical limitations, it is often not feasible to collect a new data set on which to evaluate the fit and/or predictive accuracy of a model. Using the same data to assess model fit/accuracy as was used to initially build the model carries with it the bias of over-fitting. Over-fitting essentially means the model requires more information than the data can provide. Over-fitting is one aspect of the larger issue of what statisticians refer to as shrinkage (Harrell, Lee, & Mark, 1996). When over-fitting occurs, the parameter estimates will be exaggerated and prediction error estimates will be downwardly biased; meaning, they will indicate less prediction error and better model fit than is really the case. This results from evaluation of predicted values against actual outcome values which were used to build the model (initial fit). One might refer to this as double dipping. For instance, in a standard regression analysis, the predicted values (y-hat or ŷ) are subtracted from the actual values of the outcome (y) in order to create and then evaluate the residuals. Bias is introduced because the residuals do not accurately reflect an estimate of prediction error with new data. Using the actual values of the outcome (y) for both model fit and in the evaluation of prediction error provides a bias view as to how the model will perform with new values of the predictor variables.

Cross validation is a model evaluation method that is better than simply looking at the residuals. Residual evaluation does not indicate how well a model can make new predictions on cases it has not already seen. Cross validation techniques tend to focus on not using the entire data set when building a model. Some cases are removed before the data is modeled; these removed cases are often called the testing set. Once the model has been built using the cases left (often called the training set), the cases which were removed (testing set) can be used to test the performance of the model on the “unseen” data (i.e. the testing set).

The data used in this article was simulated and contains one continuous (interval/ratio) outcome variable (y) and seven other continuous (interval/ratio) variables (x1, x2, x3, x4, x5, x6, & x7). All 8 variables have an approximate mean of 10 and are (multivariate) normally distributed. Initially, a population was created (N = 1,000,000) and from it samples were randomly drawn by randomly selecting population identification numbers (p.id). The first sample (n = 100) was
drawn and can be read in from the web (as will be shown below). The first sample contains an additional column for identifying each case of the sample (s.id).

The general idea below is that we are working from a perspective of a researcher or research team with a theoretically motivated study in which we believe seven measured variables (x1 - x7) predict the outcome (y). Our goal is to use cross validation to estimate the prediction error of our model and if warranted, identify a linear model (OLS regression) which offers the lowest prediction error estimate(s).

Of course, the real goal of this article is to show various approaches to cross validation (and *true* validation); as well as showing some of the things which can (and often do) compromise the validity of model fit estimates and prediction error estimates. Keep in mind throughout; we use a simple (OLS) linear regression as an example here, but the ideas conveyed here apply to other types of modeling (e.g. GLM, HLM, SEM, etc.).

**Creating the Data**

The population ($N = 1,000,000$) was created using the ‘MASS’ package (Venables & Ripley, 2002) and the ‘mvrnorm’ function to create a multivariate normal variance / covariance matrix with each of the eight variables centered on a mean of 10. Notice, the first variable was designated the outcome and the second, third, and fourth variables are the only real predictors – they have a relationship with the outcome while the other four variables do not. Also, there is no multicollinearity among the second, third, and fourth variables; whereas, there is multicollinearity among the other four variables.

![R Console](image)

Then, a population identification variable (p.id) was created using the ‘seq’ function for sequentially labeling the cases from 1 to 1,000,000. Finally, the variables were put into a data frame and the variables were renamed.
Next, two samples were drawn by randomly picking population identification numbers.

Then, all three data frames and the workspace were saved out to the desktop and posted on the web so that they could be read in from the web and offer repeatable results.

**Examples**

Read in the first sample data file \((n = 100)\) from the web naming it "sample1.df" as below; and get the ubiquitous 'head' and 'summary' of the data to see what it looks like.
Fit the original sample 1 data to the linear model and calculate the Average Residual Squared Error for all cases in the sample (baseline [and biased] prediction error estimate: RSE.n).

These initial estimates (generated above) indicate superb model fit (R-squared & Adj. R-squared) and an extremely small prediction error estimate (RSE.n); however, they are all compromised by over-fitting. If you are not familiar with the scientific notation of R, the ‘e-00’ refers to a negative exponent and the ‘e+00’ refers to a positive exponent. For example, 5.234e-03 = 0.005234 and 5.234e+03 = 5234.00.

Split-half Cross Validation
Split-half cross validation (also called: split-sample or hold-out validation) involves simply dividing the data into two halves; one the training set, on which the model is fit, and one the testing set, on which the model is evaluated.

Divide the sample data into two halves, the training set and the testing set.

Specify/fit the model with the training set.

```
> model.1 <- lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, data = training.set)
> summary(model.1)
```

| Residuals: | -0.148511 -0.057115 0.039986 0.050032 0.153606 |
| Coefficients: | Estimate Std. Error t value Pr(>|t|) |
| Intercept | -2.768627 0.271758 -10.188 6.42e-13 *** |
| x1 | 0.858630 0.013399 64.131 < 2e-16 *** |
| x2 | 0.542511 0.011220 48.351 < 2e-16 *** |
| x3 | 0.244069 0.012323 19.792 < 2e-16 *** |
| x4 | -0.154081 0.012362 -12.887 5.02e-15 *** |
| x5 | -0.181493 0.017839 -10.201 1.05e-12 *** |
| x6 | -0.004515 0.029921 -0.151 0.8808 |
| x7 | -0.061335 0.031464 -1.969 0.0555 . |

```
> Apply the specified model (coefficients) to predictor values from the testing set to predict (model based) outcome values of the testing set.
```
Compare these predicted values to the actual values of the outcome in the testing set.
The results above look great; no difference between the two (paired) means, the correlation is virtually +1....perhaps a bit too good to be true?

Calculating the average cross validation sums of squares.

Compare the 'cv.ss' to the Average Residual Squared Error (RSE.n). Notice, they are virtually the same, with RSE.n slightly biased (estimating less prediction error). The difference would be
more pronounced with real data; here we have no measurement error and the effects sizes are massive.

## Leave One Out Cross Validation

The Leave One Out Cross Validation (LOOCV) strategy in its most basic form, simply takes one observation out of the data and sets it aside as the 'testing set' like what was done above. Then the model is applied to the training set of \( n - 1 \) cases (i.e. the data minus the single testing set case). The resulting coefficients are applied to the testing set case to produce a predicted value which in turn is then compared to the actual value (of y) of that single case. Below, we avoid the most basic form of LOOCV and instead iteratively conduct the procedure across all cases (i.e. each case is 'left out' once).

Setting up the initial conditions and creating an empty vector to store the values of \( y.hat \) for each iteration; then running the LOOCV iteratively with a ‘for-loop’.

Calculating the average cross validation sums of squares and comparing it to our baseline RSE.n; again, both are very close to zero; with the cv.ss indicating an even smaller estimate of prediction error.

## Bootstrapped LOOCV

Here, we create 10 bootstrapped samples (sample WITH replacement from the original sample) and apply the LOOCV from above to each of the 10 bootstrapped samples; each time saving the \( y.hat \) values and then calculating the errors (cv.ss) as well as the correlation between \( y.hat \) and
the actual values of y in each bootstrapped sample. Typically, more than 10 bootstrapped samples would be required to reach a stable estimate; for example purposes, we use only 10 here.

Creating three empty objects for storing the output values; then run the for-loop.

```r
> errors <- as.data.frame(matrix(0, ncol = 10, nrow = 50))
> cv.ss <- as.vector(0)
> corr.yhat <- as.vector(0)
> for (i in 1:10){
+   boot.id <- sample(sample1.dfs$id, 50, replace = TRUE)
+   boot.sample <- data.frame(sample1.dfs[boot.id])
+   n <- 0
+   y.hat <- as.vector(0)
+   for (k in 1:nrow(boot.sample)){
+     n <- n + 1
+     model <- lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, data = boot.sample[-n,])
+     model$coefficients[3]*boot.sample[n,5] + model$coefficients[4]*boot.sample[n,6] +
+     model$coefficients[5]*boot.sample[n,7] + model$coefficients[6]*boot.sample[n,8] +
+     model$coefficients[7]*boot.sample[n,9] + model$coefficients[8]*boot.sample[n,10]
+   }
+   errors[i,1] <- boot.samples[y - y.hat]
+   cv.ss[i] <- (sum(boot.samples[y - y.hat]^2)/50)
+   corr.yhat[i] <- cor(boot.samples[y], y.hat)
+ }
```

Summary of the errors for each of the 10 iterations (loops); each vector corresponds to each loop/bootstrapped sample.

```r
> summary(errors)
          V1         V2         V3         V4         V5
Min. -0.2107229 Min. : -0.434039 Min. : -0.630777 Min. : -0.219415 Min. : -0.219193
1st Qu. -0.0576941 1st Qu.: -0.031617 1st Qu.: -0.040772 1st Qu.: -0.065178 1st Qu.: -0.064804
Median -0.0008719 Median : -0.001018 Median : -0.008932 Median : 0.007586 Median : 0.029720
Mean -0.0014962 Mean : -0.001436 Mean : -0.001562 Mean : -0.003047 Mean : -0.003012
3rd Qu. 0.1978080 3rd Qu.: 0.042858 3rd Qu.: 0.077144 3rd Qu.: 0.077336 3rd Qu.: 0.068355
Max. 0.1978662 Max. : 0.173699 Max. : 0.215780 Max. : 0.174627 Max. : 0.192277
V6         V7         V8         V9         V10
Min. -0.145993 Min. : -0.2107229 Min. : -0.434039 Min. : -0.434039 Min. : -0.151049
1st Qu. -0.057849 1st Qu.: -0.0700161 1st Qu.: -0.053289 1st Qu.: -0.052643 1st Qu.: -0.046621
Median 0.0021889 Median : 0.0132802 Median : 0.005144 Median : 0.005949 Median : 0.017861
Mean 0.0003277 Mean : 0.0007376 Mean : 0.0006105 Mean : 0.0001666 Mean : 0.0002541
3rd Qu. 0.044317 3rd Qu.: 0.0561143 3rd Qu.: 0.0419220 3rd Qu.: 0.0227421 3rd Qu.: 0.0476740
Max. 0.140738 Max. : 0.1537669 Max. : 0.1187669 Max. : 0.1642784 Max. : 0.1357129
```

Comparison of the average cross validation sums of squares for each of the 10 loops (cv.ss) to the Average Residual Squared Error (RSE.n) from each iteration and the bootstrapped average of the cv.ss across iterations (an average of the averages).

```r
> RSE.n
(1) 0.005187536
> cv.ss
> mean(cv.ss)
(1) 0.0001307674
```

All 10 bootstrapped estimates (and their average) are lower than the RSE.n. The average of the averages across iterations provides the more robust estimate of prediction error.
Correlations between \( y \) and \( \hat{y} \) for each of the 10 iterations; all of which are all nearly 1.0.

**Bootstrapped (cross validation): Estimates of Prediction Error (Efron & Tibshirani, 1993).**

The bootstrapped cross validation approach represents a robust method of estimating the prediction error for a model. Each bootstrapped sample (sampling WITH replacement from the original sample) has the model fit to it and the subsequent coefficients are then used to predict outcome scores of the original sample as well as the outcome scores of the bootstrapped sample. Naturally, the errors associated with the original sample estimates will be larger than the errors associated with the bootstrapped sample estimates; because, the bootstrapped sample was used to fit the model and generate the coefficients. Then, using the DIFFERENCE between the original sample errors and the bootstrapped sample errors provides us with an estimate of bias or "optimism" (Efron & Tibshirani, 1993, p. 248). Finally, the average optimism (average of each optimism from all the bootstrapping) can be added to the original sample RSE.n; which corrects it and provides a more accurate estimate of average prediction error.

Setting initial conditions and creating an empty data frame to store results. The little ‘n’ sets the number of cases in each bootstrapped sample and the capital ‘N’ sets the number of bootstrapped samples to draw. The ‘prediction.error’ data frame stores the original sample errors, the bootstrapped sample errors, and the optimism value from each iteration or loop.

Run the bootstrapped cross validation loop.
The 'prediction.errors' data frame contains the following: the original sample-based prediction error estimate (average cross validation sums of squares), the bootstrapped sample-based prediction error estimate (average cross validation sums of squares), and the optimism estimate (difference between the previous two); 'prediction.errors' contains all 3 for each N = 200 iterations (bootstraps).

Averages of each and a comparison to the RSE.n.

The improved bootstrapped prediction error estimate, which adds a bias correction to the original RSE.n; because, RSE.n is biased downwardly (i.e. predicted error estimate is smaller [less error] than it really should be due to overfitting). See Efron, 1993, p. 237 - 249.

**Real Cross Validation**

'Real cross validation' = collecting another sample of data from the same population and using the new data (measured with the same instruments) to "validate" the model's accuracy. Of course, in actual research it is often impossible to do this, because of funding, time constraints, etc. Collecting new data: here, this is very easy because we have simulated the data by first generating a simulated population and then sampling from it to build the model. This is why our estimates are very 'unbiased' above; most of the prediction error estimates (regardless of cross validation technique) have produced very similar estimates (virtually the same near-zero estimates of prediction error).
Collecting (drawing) a new sample from the population. Here; read in the second sample from the web, naming it "sample2.df".

Now we can use the original sample's linear model coefficients, applied to the new (2nd) sample's data (predictor variables), to 'predict' values on the outcome variable (y) of the new (2nd) sample data set. Then we can compare the 'predicted' values (y.hat) with the actual values (y) from the new data. Recall the first sample's linear model.

Applying the coefficients to the new predictor variable data to create the 'predicted' values of the outcome (y.hat).
Comparison of the 'predicted' values (\(y_{\text{hat}}\)) to the actual (new) values of the outcome (\(\text{sample2.df}y\)).

The following creates two histograms (in one graphics window); showing \(y_{\text{hat}}\) and \(y\).
So, clearly our model (based on the original sample) is doing an excellent job of predicting new values of 'y' based on new values of all the predictors. This is due to a few advantages of simulating or creating the data in the manner we did: the variables have zero measurement error and the cases were randomly sampled from a defined (and constant) population (i.e. zero sampling error/bias); although, because this data is simulated, there is a slight chance the same case(es) may appear in multiple samples. However, that was not the case here; meaning, each case is unique to sample 1 or sample 2. This is largely due to the overwhelmingly large population size ($N = 1000000$) in relation to sample sizes ($n = 100$).

Furthermore, our samples have greater than a 10 to 1 ratio of cases to variables. Simply put, our sample sizes ($n = 100$ each), though not large, are adequate because; for each of our 8 variables
we have at least 10 cases -- it would obviously be better to have a 20 to 1 (or even higher) ratio; but, 10 to 1 is about the lowest acceptable ratio. However, one should also remember that the sample size in relation to the population size is important. The smaller the sample (in comparison to the population), the more likely the sample, even a random sample, will contain bias.

In a 'real-world' setting, it is not uncommon to have violations of one, or more of the above conditions -- any one of which would bias the estimate of RSE.n; meaning the model would have more prediction error than the RSE.n would indicate.

Two other threats to the accuracy of estimating prediction errors for a model are model specification error and model search error. Model specification error can happen in three ways; (1) using the wrong model 'form' (e.g. using a linear model rather than a quadratic model when the quadratic more accurately represents the data), (2) errors of omission (leaving crucial variables out of the model), and (3) errors of inclusion (including non-essential variables in the model). Model search error is related to the first type of model specification error mentioned above. Model search error refers to the bias associated with the method of selecting a model form. In other words, model search error refers to the uncertainty of picking the model form you did; which can contribute to the prediction error. The term model search error is used as a reflection of the search for the best model or best set of models (e.g. Bayesian Model Averaging). It is important to note that cross validation techniques do not address the problem of model search nor do they address model specification error of the first type (i.e. using the wrong model form). Errors of inclusion can increase multicollinearity and cause bias in model fit estimates, prediction error estimates, and individual parameter estimates (i.e. coefficients). To clarify this last point, notice that throughout the above results, our model has performed almost perfectly with respect to model fit (R-squared & Adj. R-squared), small residuals (errors), and very small predicted error estimates. However, closer inspection of the sample(s) data (and the population data) will reveal two related faults with our model.

First, we have model specification errors of inclusion; four of our predictors are not related to the outcome (AT ALL!), which means they have no business being included in the model. The only thing they do is artificially increase fit measures (R-squared & Adj. R-squared). Second, these four variables are even more disruptive because they bring with them multicollinearity (intercorrelations among themselves and to a lesser extent with the actual 3 predictors); a condition which decreases the validity of the coefficients in terms of interpreting variable importance.

To ‘see’ the relationships referred to above, simply take a look at the correlation matrix of ‘y’ and all the ‘x’ variables. Below, both sample correlation matrices are displayed with rounding two places after the decimal (if interested in seeing the matrices without rounding use this: cor(sample1.df[,3:10]) and this: cor(sample2.df[,3:10])
If we examine the true population values we see that indeed, only x1, x2, and x3 are related to y; the other four variables (x4, x5, x6, & x7) do not contribute to y and they do add multicollinearity to our model (further confusing the results). However, in order to review the population, we must read in the original work space (because the internet connection will time out if attempting to read in the data directly due to the large number of cases; $N = 1,000,000$).

To compare the ‘true’ population model with the mis-specified model, run both on the population. Be advised, this takes a minute or so due to the 1,000,000 cases.
Based on the results of the ‘bad.model’ it appears superior; smaller residuals, larger R-squared and Adj. R-squared, and smaller residual standard errors. But, we ‘know’ those are biased because the correlation matrix(-ces) shows us what is *truly* related to the outcome (y) and what is not. Of course, the p-values cannot be relied upon due to the enormous number of cases.

References & Resources


