<https://sites.duke.edu/teachingr/files/2013/06/CrossValidation1_JDS_May2011.pdf>

Cross Validation techniques in **R**: A brief overview of some methods, packages, and functions for assessing prediction models.

Dr. Jon Starkweather, Research and Statistical Support consultant

This month’s article focuses on an initial review of techniques for conducting cross validation in **R**. Next month, a more in-depth evaluation of cross validation techniques will follow. Cross validation is useful for overcoming the problem of over-fitting. Over-fitting is one aspect of the larger issue of what statisticians refer to as shrinkage (Harrell, Lee, & Mark, 1996). Over-fitting is a term which refers to when the model requires more information than the data can provide. For example, over-fitting can occur when a model which was initially fit with the same data as was used to assess fit. Much like exploratory and confirmatory analysis should not be done on the same sample of data, fitting a model and then assessing how well that model performs on the same data should be avoided. When we speak of assessing how well a model performs, we generally think of fit measures (e.g. *R*2, adj. *R*2, AIC, BIC, RMSEA, etc.); but, what we really would like to know is how well a particular model *predicts* based on new information. This really gets at the goals of science and how we go about them; observation yields description, experimentation yields explanation, and all of those utilize statistical models with the goal of explanation and/or prediction. When predictions are confirmed, evidence is born for supporting a theory. When predictions fail, evidence is born for rejecting a theory.

Fit measures, whether in the standard regression setting or in more complex settings, are biased by over-fitting – generally indicating better fit, or less prediction error than is really the case. Prediction error refers to the discrepancy or difference between a predicted value (based on a model) and the actual value. In the standard regression situation, prediction error refers to how well our regression equation predicts the outcome variable scores of new cases based on applying the model (coefficients) to the new cases’ predictor variable scores. When dealing with a single sample, typically the residuals are a reflection of this prediction error; where the residuals are specifically how discrepant the predicted values (y-hat or ŷ) are from the actual values of the outcome (y). However, because of over-fitting, these errors or residuals will be biased downward (less prediction error) due to the actual outcome variable values being used to create the regression equation (i.e. the prediction model). Cross validation techniques are one way to address this over-fitting bias.

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 examples below are meant to show how some common cross validation techniques can be implemented in the statistical programming language environment **R**. The examples below focus on standard multiple regression situations using a sample drawn from a simulated population of *true scores*. Next month’s article will show how the population was generated and how each sample was drawn, as well as a more in-depth exploration of how cross validation techniques address the over-fitting problem.

**Example Data**

The examples below were designed to be representative of a typical modeling strategy, where the researcher has theorized a model based on a literature review (and other sources of information) and has collected a sample of data. The setting for the examples below concerns a model with seven hypothesized predictors (x1, x2, x3, x4, x5, x6, & x7), each interval/ratio scaled, and one interval/ratio outcome variable (y). All variables have an approximate mean of 10. The sample contains two additional columns, one which identifies cases sequentially in the sample (s.id) and one which identifies cases sequentially in the population from which it was drawn (p.id). The sample contains 100 cases randomly sampled from a defined population of 1,000,000 individuals.

First, read in the sample data from the web, naming it ‘sample1.df’ (df = data.frame), and getting the ubiquitous ‘head’ and ‘summary’ to get an idea of what the data looks like.

**The ‘Design’ Package**

Next, we specify the model. Typically, we would use the ‘lm’ function from the base ‘stats’ package to specify an Ordinary Least Squares (OLS) regression model. However, here we will use the ‘ols’ function in the ‘Design’ package (Harrell, 2009). So, first we must load the ‘Design’ package, which has several dependencies.

Now, we can use the ‘ols’ function to specify the model and get a summary of it. Make sure to set the optional arguments ‘x = TRUE’ and ‘y = TRUE’ as these will save a design matrix of predictors and a vector of outcome values. These two objects will be used in the cross validation techniques below. 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.

Next, we can begin exploring cross validation techniques. The 'validate' function in the 'Design' package "does resampling validation of a regression model, with or without backward step-down variable deletion" (Harrell, 2009, p. 187). Here, our examples focus on OLS regression, but the 'validate' function can hand a logistic model as well; as long as the model is fit with the 'lrm' function (Logistic Regression Model) in the 'Design' package. The key part of the output for this function is the 'index.corrected' measures of fit -- which corrects for over-fitting. We start with the default values/arguments for 'validate' which uses the 'boot' method (bootstrapped validation; Efron, 1983; Efron & Tibshirani, 1993). Bootstrapped validation takes B number of samples of the original data, with replacement, and fits the model to this training set. Then, the original data is used as the testing set for validation.

Notice in the output above the index corrected estimates are all marginally worse in terms of fit and / or prediction error. In other words, the index corrected measures do not reflect the shrinkage caused by over-fitting. The “optimism” (Efron & Tibshirani, 1993, p. 248) is the difference between the training set estimates and the test set estimates and can be thought of as the amount of optimism of each initial estimate (e.g. how much the training estimates are biased).

Next, we can explore the ‘crossvalidation’ method, which uses B number of observations as the testing set (testing or validating the model) and the rest of the sample for the training set (building the model).

Next, we can take a look at the “.632” bootstrapped method which corrects for the bias in prediction error estimates “based on the fact that bootstrap samples are supported on approximately .632*n* of the original data points” (Efron, 1983; Efron & Tibshirani, 1997, p. 552).

**The ‘DAAG’ package**

Another package which is capable of performing cross validation is the Data Analysis And Graphing (‘DAAG’) package (Maindonald & Braun, 2011) which also has several dependent packages.

The ‘DAAG’ package contains three functions for *k* – fold cross validation; the ‘cv.lm’ function is used for simple linear regression models, the ‘CVlm’ function is used for multiple linear regression models, and the ‘CVbinary’ function is used for logistic regression models. The *k* – fold method randomly removes *k* – folds for the testing set and models the remaining (training set) data. Here we use the commonly accepted (Harrell, 1998) 10 – fold application.

Some output (folds) has been omitted.

Here, at the bottom of the output we get the cross validation residual sums of squares (Overall MS); which is a corrected measure of prediction error averaged across all folds. The function also produces a plot (below) of each fold’s predicted values against the actual outcome variable (y); with each fold a different color.

**The ‘boot’ package**

Lastly, we can use the ‘boot’ package (Ripley, 2010) for cross validation of generalized linear models (e.g. binomial, Gaussian, poisson, gamma, etc.). Bootstrapping can be used to correct for some of the bias associated with the other cross validation techniques.

First, we must fit the model. Our example below is *really* an OLS regression model, but if we specify ‘family = gaussian’ then it is the same as using ‘lm’. If we had a logistic model, then we would specify ‘family = binomial(link = logit)’ to fit the logistic model.

The ‘cv.glm’ function “estimates the *k* – fold cross validation prediction error for generalized linear models” (Ripley, 2010). If *k* – fold is set to the number of cases (rows), then a complete Leave One Out Cross Validation (LOOCV) is done. The LOOCV method is intuitively named; essentially, one case is *left out* as the testing set and the rest of the data is used as the training set. If this process is repeated so that each case is given a chance as the testing case, then we have the complete LOOCV method. The 'cv.glm' function returns a 'delta' which shows (first) the raw cross-validation estimate of prediction error and (second) the adjusted cross-validation estimate. The adjustment is designed to compensate for the bias introduced by not using leave-one-out cross-validation. The default for ‘cv.glm’ is complete LOOCV.

First, we run the common 10 – fold cross validation. Below, the majority of seed information is cut off the end of the figure.

Next, we run the complete LOOCV method, specifying *k* as the number of rows in the sample data (nrow). Again, below the majority of the seed numbers have been left off the figure.

Obviously the delta numbers match because we used the LOOCV method. Recall, the first delta value is the raw cross validation estimate of prediction error and the second is the adjusted cross validation estimate; which is supposed to adjust for the bias of not using the LOOCV method.

**Conclusions**

Three packages were employed to demonstrate some relatively simple examples of conducting cross validation in the **R** programming language environment. Cross validation refers to a group of methods for addressing the some over-fitting problems. Over-fitting refers to a situation when the model requires more information than the data can provide. One way to induce over-fitting is by specifying the model with the same data on which one assesses fit or prediction error. The examples here were conducted using simulated data. Rather strikingly, you may have noticed, the estimates of prediction error were not terribly different from the full sample (over-fitted) estimates, even though this sample was considerable small (*n* = 100) in comparison to its parent population (*N* = 1,000,000). These results might lead one to think cross validation and over-

fitting are not things one needs to be concerned with. However, there are a few reasons our estimates here were not more starkly different than the full sample estimates and you might be surprised to find that some of our predictor variables are not at all related to the outcome variable. Next month’s article will reveal the *secrets* behind those statements. However, cross validation and over-fitting are serious concerns when dealing with real data and should be considered in each study involving modeling.

References & Resources

Chernick, M. R. (2008). *Bootstrap methods: A guide for practitioners and Researchers* (2nd ed.). Hoboken, NJ: John Wiley & Sons, Inc.

Efron, B. (1983). Estimating the error rate of a prediction rule: Some improvements on cross- validation. *Journal of the American Statistical Association, 78*, 316 - 331.

Efron, B, & Tibshirani, R. (1993). *An introduction to the bootstrap*. New York: Chapman & Hall.

Efron, B., & Tibshirani, R. (1997). Improvements on cross-validation: The .632+ bootstrap method. *Journal of the American Statistical Association, 92*(438), 548 - 560.

Harrell, F., Lee, K., & Mark, D. (1996). Tutorial in Biostatistics: Multivariable prognostic models: Issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in Medicine, 15*, 361 – 387. Available at: http://www.yaroslavvb.com/papers/steyerberg-application.pdf

Harrell, F. (1998). Comparisons of strategies for validating binary logistic regression models. Available at: http://biostat.mc.vanderbilt.edu/twiki/pub/Main/RmS/logistic.val.pdf

Harrell, F. E. (2001). *Regression modeling strategies: With applications to linear models, logistic regression, and survival analysis*. New York: Springer-Verlag, Inc.

Harrell, F. E. (2009). Package 'Design'. Available at CRAN:http://cran.r- project.org/web/packages/Design/index.html

Maindonald, J., & Braun, W. J. (2011). Package 'DAAG'. Available at CRAN: http://cran.r- project.org/web/packages/DAAG/index.html

Moore, A. (2008). Cross-Validation: tutorial slides. Available at: http://www.autonlab.org/tutorials/overfit.html

Ripley, B. (2010). Package 'boot'. Available at CRAN: http://cran.r- project.org/web/packages/boot/index.html

Schneider, J. (1997). Cross validation. Availabe at: http://www.cs.cmu.edu/~schneide/tut5/node42.html

Wikipedia. (2011). Cross-validation (statistics). Available at: http://en.wikipedia.org/wiki/Cross- validation\_%28statistics%29

Tune in next time, *Same bat channel, same bat time...*