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 $\hat{y}$) 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.

```
R Console
File  Edit  Misc  Packages  Windows  Help
> sample1.df <- read.table("http://www.unt.edu/rss/class/Jon/R_SC/Module9/CrossValidation/cv_sample1.df.txt",
+    header=TRUE, sep=",", na.strings="NA", dec=".", strip.white=TRUE)
> head(sample1.df)
  s.id   p.id         y        x1        x2        x3        x4        x5       x6        x7
1    1 297497 10.239325  9.468152 10.704792 10.542263  9.014519 10.102986 10.29590  9.861752
2    2 218250 10.467680 11.457565  8.870517  9.903951 10.312773  9.714411 11.32481 11.280861
3    3 808994 11.606858  9.995756 11.690831 11.312753  9.394473  8.883531 10.00956  9.582854
4    4 352628  9.565044 10.092380  8.950241  8.487879  9.634021  8.146134  9.99382  9.561203
5    5 371676  9.904931  9.370441 10.906479 11.628177 11.768006 10.689805 10.84285 11.275871
6    6 342721  9.647289  9.130054 10.284733 10.106447  8.736938  9.837188 10.29698 10.506574
> summary(sample1.df)
      s.id            p.id              y                x1               x2               x3
 Min.   :  1.00   Min.   : 33985   Min.   : 7.770   Min.   : 7.651   Min.   : 6.607   Min.   : 7.543
 1st Qu.: 25.75   1st Qu.:299989   1st Qu.: 9.411   1st Qu.: 9.456   1st Qu.: 9.175   1st Qu.: 9.328
 Median : 50.50   Median :465996   Median :10.038   Median :10.092   Median :10.050   Median :10.069
 Mean   : 50.50   Mean   :510596   Mean   :10.092   Mean   :10.159   Mean   : 9.978   Mean   :10.027
 3rd Qu.: 75.25   3rd Qu.:710626   3rd Qu.:10.839   3rd Qu.:10.743   3rd Qu.:10.707   3rd Qu.:10.655
 Max.   :100.00   Max.   :979113   Max.   :12.019   Max.   :12.589   Max.   :12.690   Max.   :12.011
       x4               x5               x6               x7
 Min.   : 7.817   Min.   : 7.665   Min.   : 6.982   Min.   : 6.900
 1st Qu.: 9.664   1st Qu.: 9.339   1st Qu.: 9.539   1st Qu.: 9.506
 Median :10.287   Median :10.032   Median :10.031   Median :10.127
 Mean   :10.176   Mean   :10.015   Mean   :10.083   Mean   :10.117
 3rd Qu.:10.695   3rd Qu.:10.634   3rd Qu.:10.628   3rd Qu.:10.897
 Max.   :12.329   Max.   :12.392   Max.   :12.832   Max.   :12.401
>
```

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.

```
R Console
File  Edit  Misc  Packages  Windows  Help
> library(Design)
Loading required package: Hmisc
Loading required package: survival
Loading required package: splines

Attaching package: 'Hmisc'

The following object(s) are masked from 'package:survival':

    untangle.specials

The following object(s) are masked from 'package:base':

    format.pval, round.POSIXt, trunc.POSIXt, units

Design library by Frank E Harrell Jr

Type library(help='Design'), ?DesignOverview, or ?Design.Overview')
to see overall documentation.


Attaching package: 'Design'

The following object(s) are masked from 'package:Hmisc':

    strgraphwrap

The following object(s) are masked from 'package:survival':

    Surv

>
```

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.

```
R Console
File  Edit  Misc  Packages  Windows  Help
> model.1 <- ols(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, sample1.df, x = TRUE, y = TRUE)
> model.1

Linear Regression Model

ols(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, data = sample1.df,
    x = TRUE, y = TRUE)

         n Model L.R.       d.f.         R2      Sigma
       100      515.3          7     0.9942    0.07509

Residuals:
      Min        1Q    Median        3Q       Max
-0.154608 -0.056157  0.004808  0.045365  0.186316

Coefficients:
             Value Std. Error       t  Pr(>|t|)
Intercept -2.77898   0.169566 -16.389 0.000e+00
x1         0.86274   0.008341 103.429 0.000e+00
x2         0.54708   0.007593  72.050 0.000e+00
x3         0.23564   0.008475  27.803 0.000e+00
x4        -0.16429   0.009121 -18.013 0.000e+00
x5        -0.14094   0.009475 -14.874 0.000e+00
x6         0.01887   0.017262   1.093 2.771e-01
x7        -0.08118   0.018421  -4.407 2.836e-05

Residual standard error: 0.07509 on 92 degrees of freedom
Adjusted R-Squared: 0.9938

>
```

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.

```
R Console
File  Edit  Misc  Packages  Windows  Help
> val.boot <- validate(model.1, method = "boot", B = 40, bw = FALSE, rule = "aic",
+                      type = "residual", sls = 0.05, aic = 0, pr = FALSE)
> val.boot
           index.orig   training         test      optimism index.corrected  n
R-square  0.994216123 0.99423426  0.993705437  0.0005288253     0.993687298 40
MSE       0.005187536 0.00493656  0.005645568 -0.0007090077     0.005896543 40
Intercept 0.000000000 0.00000000 -0.009016284  0.0090162843    -0.009016284 40
Slope     1.000000000 1.00000000  1.000665178 -0.0006651782     1.000665178 40
>
```

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).

```
R Console
File  Edit  Misc  Packages  Windows  Help
> val.cross <- validate(model.1, method = "crossvalidation", B = 40, bw = FALSE, rule = "aic",
+                       type = "residual", sls = 0.05, aic = 0, pr = FALSE)
> val.cross
           index.orig    training        test     optimism index.corrected  n
R-square  0.994216123 0.994228510 -0.06680937  1.061037880    -0.066821757 40
MSE       0.005187536 0.005175911  0.00653804 -0.001362129     0.006549665 40
Intercept 0.000000000 0.000000000  0.13240866 -0.132408663     0.132408663 40
Slope     1.000000000 1.000000000  0.98214115  0.017858848     0.982141152 40
>
```

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).

```
R Console
File  Edit  Misc  Packages  Windows  Help
> val.632 <- validate(model.1, method = ".632", B = 40, bw = FALSE, rule = "aic",
+                     type = "residual", sls = 0.05, aic = 0, pr = FALSE)


Weights for .632 method (ordinary bootstrap weights 0.025)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0188  0.0229  0.0255  0.0250  0.0269  0.0315
> val.632
           index.orig    training        test      optimism index.corrected  n
R-square  0.994216123 0.994488283 0.992860602  0.0008566896     0.993359434 40
MSE       0.005187536 0.004761727 0.006551268 -0.0008618789     0.006049415 40
Intercept 0.000000000 0.000000000 0.014447216 -0.0091306407     0.009130641 40
Slope     1.000000000 1.000000000 0.998938150  0.0006710891     0.999328911 40
>
```

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.

```
R Console
File  Edit  Misc  Packages  Windows  Help
> library(DAAG)
Loading required package: MASS
Loading required package: rpart
Loading required package: randomForest
randomForest 4.6-2
Type rfNews() to see new features/changes/bug fixes.

Attaching package: 'randomForest'

The following object(s) are masked from 'package:Hmisc':

    combine


Attaching package: 'DAAG'

The following object(s) are masked from 'package:MASS':

    hills

The following object(s) are masked from 'package:Design':

    vif

The following object(s) are masked from 'package:survival':

    lung

>
```

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.

```
R Console
File  Edit  Misc  Packages  Windows  Help
> val.daag <- CVlm(df = sample1.df, m = 10, form.lm = formula(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7))
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value  Pr(>F)
x1         1   48.7    48.7  8631.8 < 2e-16 ***
x2         1   30.8    30.8  5471.0 < 2e-16 ***
x3         1    5.1     5.1   900.1 < 2e-16 ***
x4         1    1.8     1.8   312.4 < 2e-16 ***
x5         1    2.5     2.5   438.3 < 2e-16 ***
x6         1    0.2     0.2    41.2 6.0e-09 ***
x7         1    0.1     0.1    19.4 2.8e-05 ***
Residuals 92    0.5     0.0
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


fold 1
Observations in test set: 3 8 16 18 39 42 45 62 63 100
               X3      X8     X16    X18     X39     X42     X45    X62    X63    X100
Predicted 11.5141  9.9671  9.3781 8.8386  9.4042  9.7155 10.0094  9.277  9.079  9.4779
y         11.6069  9.9490  9.3040 8.8825  9.3647  9.6782 10.0593  9.178  8.966  9.4293
Residual   0.0928 -0.0182 -0.0741 0.0439 -0.0395 -0.0373  0.0499 -0.099 -0.113 -0.0487

Sum of squares = 0.047    Mean square = 0.0047    n = 10

fold 2
Observations in test set: 4 9 10 19 30 47 55 78 82 93
              X4     X9     X10     X19     X30    X47     X55     X78    X82    X93
Predicted 9.5047 11.785 10.0616 10.3095 10.7081 10.776 11.0915  8.2624  8.021 8.7906
y         9.5650 11.829 10.0821 10.3614 10.6913 10.612 11.0314  8.2295  7.909 8.8210
Residual  0.0604  0.044  0.0206  0.0519 -0.0168 -0.163 -0.0601 -0.0329 -0.112 0.0304

Sum of squares = 0.054    Mean square = 0.0054    n = 10

fold 3
Observations in test set: 35 54 56 73 74 77 87 88 89 99
             X35     X54   X56     X73   X74   X77     X87     X88    X89     X99
Predicted 11.287 11.0791 9.713 11.2339 9.395 9.108 10.8626  9.7615 11.293  9.7556
y         11.169 11.0216 9.847 11.2197 9.421 9.301 10.7678  9.7507 11.155  9.7173
Residual  -0.118 -0.0575 0.134 -0.0142 0.026 0.193 -0.0948 -0.0108 -0.139 -0.0383

Sum of squares = 0.1    Mean square = 0.01    n = 10

fold 4
Observations in test set: 2 21 24 41 50 64 71 75 80 95
               X2      X21    X24     X41   X50    X64     X71     X75     X80     X95
Predicted 10.5363 10.00998 10.274  8.8239 10.88 9.4221  9.8130 10.1583 10.3260 11.0276
y         10.4677 10.01757 10.417  8.7942 10.84 9.4606  9.7438 10.1024 10.3483 11.0132
Residual  -0.0686  0.00758  0.142 -0.0297 -0.04 0.0385 -0.0692 -0.0559  0.0223 -0.0144
```

Some output (folds) has been omitted.

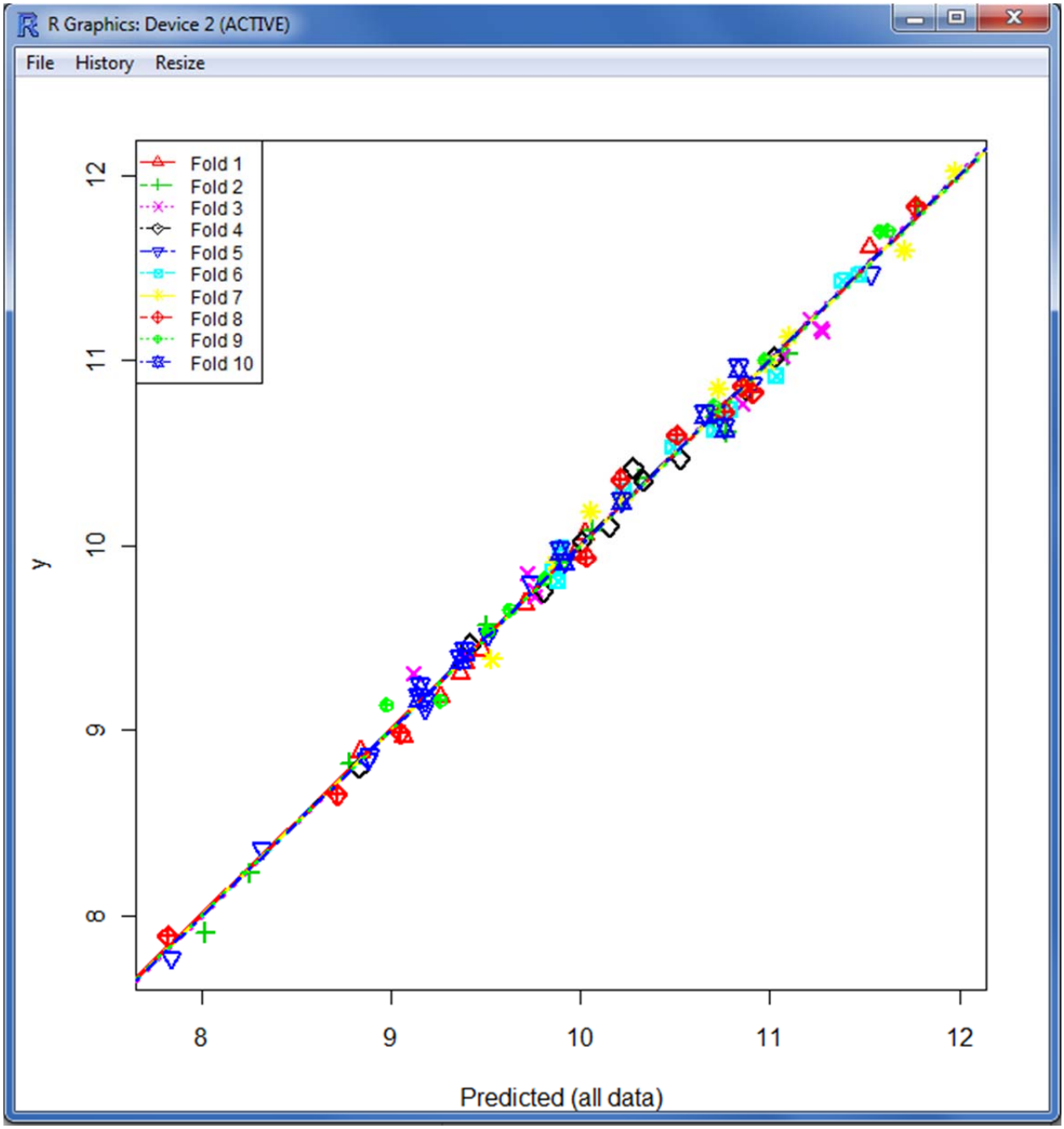
```
fold 9
Observations in test set: 6 11 22 34 37 44 51 53 65 79
              X6   X11    X22    X34     X37  X44    X51     X53     X65    X79
Predicted 9.6250 11.57  9.265 9.8877 10.9623 8.97 9.5057 10.6938 11.6039 9.8000
y         9.6473 11.70  9.154 9.9035 11.0038 9.14 9.5326 10.7534 11.6992 9.8147
Residual  0.0223  0.13 -0.111 0.0158  0.0415 0.17 0.0269  0.0596  0.0953 0.0147

Sum of squares = 0.074    Mean square = 0.0074    n = 10

fold 10
Observations in test set: 1 20 29 32 38 52 69 72 81 91
               X1    X20   X29    X32     X38    X52    X69    X72     X81    X91
Predicted 10.2170 10.765 9.353 9.8876 9.90237 9.1390 9.1459 9.3839 10.6522 10.824
y         10.2393 10.632 9.379 9.9687 9.90626 9.1711 9.2300 9.4207 10.7054 10.955
Residual   0.0223 -0.133 0.026 0.0811 0.00389 0.0321 0.0841 0.0369  0.0532  0.131

Sum of squares = 0.055    Mean square = 0.0055    n = 10
Overall ms
   0.00624
>
```

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.

```
R Console
File  Edit  Misc  Packages  Windows  Help
> library(boot)

Attaching package: 'boot'

The following object(s) are masked from 'package:survival':

    aml

> |
```

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.

```
R Console
File  Edit  Misc  Packages  Windows  Help
> model.2 <- glm(y ~ x1 + x2 + x4 + x5 + x6 + x7, sample1.df, family = gaussian)
> summary(model.2)

Call:
glm(formula = y ~ x1 + x2 + x4 + x5 + x6 + x7, family = gaussian,
    data = sample1.df)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.5173  -0.1635   0.0071   0.1343   0.4584

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.6058     0.4589   -1.32  0.19006
x1            0.8186     0.0250   32.78  < 2e-16 ***
x2            0.5777     0.0229   25.21  < 2e-16 ***
x4           -0.1748     0.0278   -6.29  1.0e-08 ***
x5           -0.1014     0.0286   -3.55  0.00061 ***
x6            0.2171     0.0479    4.53  1.8e-05 ***
x7           -0.2745     0.0520   -5.28  8.6e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.0524)

    Null deviance: 89.6896  on 99  degrees of freedom
Residual deviance:  4.8774  on 93  degrees of freedom
AIC: -2.269

Number of Fisher Scoring iterations: 2

> .
```

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.

```
R Console
File  Edit  Misc  Packages  Windows  Help
> val.10.fold <- cv.glm(data = sample1.df, glmfit = model.2, K = 10)
> val.10.fold
$call
cv.glm(data = sample1.df, glmfit = model.2, K = 10)

$K
[1] 10

$delta
     1      1
0.0572 0.0567

$seed
  [1]          403          100   1118745441   1531885299      2763106   2140198868   1789174487   -845059891
  [9] -1092788124   -695153078   -320332875   1768072975   1625781270    388718272    642795651   1378608289
 [17]  -456324464   -949634706    376543001    762448811   -707143270   1098218844   1309676639   -302363595
 [25] -1069787188  -1394539742    237698429   1290899063   -814237794   -266332584   2139062203  -2033086119
 [33]    25004776   -705146090  -1756965743   2102593859  -1873443758  -1694296348  -1047834329  -1442872963
 [41]  -707344268   2118199898    966616741  -1075465025    229050182    559293904  -1608157197  -1169305839
```

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.

```
R Console
File  Edit  Misc  Packages  Windows  Help
> val.loocv <- cv.glm(data = sample1.df, glmfit = model.2, K = nrow(sample1.df))
> val.loocv
$call
cv.glm(data = sample1.df, glmfit = model.2, K = nrow(sample1.df))

$K
[1] 100

$delta
     1      1
0.0567 0.0567

$seed
  [1]          403          200   1118745441   1531885299      2763106   2140198868   1789174487   -845059891
  [9] -1092788124   -695153078   -320332875   1768072975   1625781270    388718272    642795651   1378608289
 [17]  -456324464   -949634706    376543001    762448811   -707143270   1098218844   1309676639   -302363595
 [25] -1069787188  -1394539742    237698429   1290899063   -814237794   -266332584   2139062203  -2033086119
 [33]    25004776   -705146090  -1756965743   2102593859  -1873443758  -1694296348  -1047834329  -1442872963
 [41]  -707344268   2118199898    966616741  -1075465025    229050182    559293904  -1608157197  -1169305839
 [49]   -16478400  -1680050402   1531439081  -2105537253  -1848449238  -1822459828    699682831  -1586685115
 [57]  -1104835332   1457453906  -1691345715   2114509959  -1991874706  -1192748728   -549462709   1936425129
```

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…*