

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 (\hat{y} or \hat{y}) 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 ($x_1, x_2, x_3, x_4, x_5, x_6, \& x_7$). 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

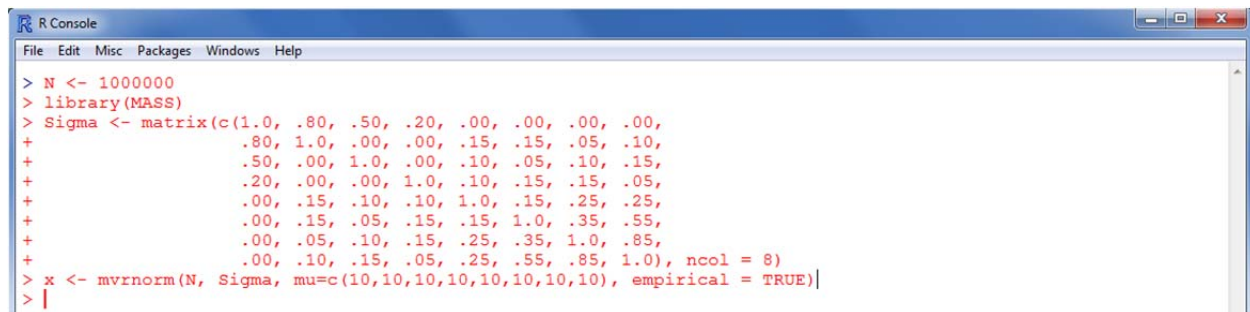
drawn and can be read in from the web (as will be shown below). The first sample contains an additional column for identifying each case of the sample (s.id).

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
File Edit Misc Packages Windows Help
> N <- 1000000
> library(MASS)
> Sigma <- matrix(c(1.0, .80, .50, .20, .00, .00, .00, .00,
+                 .80, 1.0, .00, .00, .15, .15, .05, .10,
+                 .50, .00, 1.0, .00, .10, .05, .10, .15,
+                 .20, .00, .00, 1.0, .10, .15, .15, .05,
+                 .00, .15, .10, .10, 1.0, .15, .25, .25,
+                 .00, .15, .05, .15, .15, 1.0, .35, .55,
+                 .00, .05, .10, .15, .25, .35, 1.0, .85,
+                 .00, .10, .15, .05, .25, .55, .85, 1.0), ncol = 8)
> x <- mvrnorm(N, Sigma, mu=c(10,10,10,10,10,10,10,10), empirical = TRUE)|
> |
```

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.

```
R Console
File Edit Misc Packages Windows Help

> p.id <- seq(1:N)
> population.df <- data.frame(p.id, x)
> names(population.df)[2] <- "y"
> names(population.df)[3] <- "x1"
> names(population.df)[4] <- "x2"
> names(population.df)[5] <- "x3"
> names(population.df)[6] <- "x4"
> names(population.df)[7] <- "x5"
> names(population.df)[8] <- "x6"
> names(population.df)[9] <- "x7"
> |
```

Next, two samples were drawn by randomly picking population identification numbers.

```
R Console
File Edit Misc Packages Windows Help

> n <- 100
> s.id <- seq(1:n)
> samp <- sample(population.df[,1], n, replace = FALSE)
> sample1.df <- data.frame(s.id, population.df[samp,])
> rm(n, s.id, samp)
> |
```

```
R Console
File Edit Misc Packages Windows Help

> n <- 100
> s.id <- seq(1:n)
> samp <- sample(population.df[,1], n, replace = FALSE)
> sample2.df <- data.frame(s.id, population.df[samp,])
> rm(n, s.id, samp)
> |
```

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.

```
R Console
File Edit Misc Packages Windows Help

> write.table(population.df,
+ "C:/Users/jds0282/Desktop/CrossValidation/cv_population.df.txt",
+ sep=",", col.names=TRUE, row.names=FALSE, quote=TRUE, na="NA")
> write.table(sample1.df,
+ "C:/Users/jds0282/Desktop/CrossValidation/cv_sample1.df.txt",
+ sep=",", col.names=TRUE, row.names=FALSE, quote=TRUE, na="NA")
> write.table(sample2.df,
+ "C:/Users/jds0282/Desktop/CrossValidation/cv_sample2.df.txt",
+ sep=",", col.names=TRUE, row.names=FALSE, quote=TRUE, na="NA")
> save.image("C:\\Users\\jds0282\\Desktop\\CrossValidation\\CrossValidation_003.RData")
> |
```

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.

```

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
> |

```

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).

```

R Console
File Edit Misc Packages Windows Help
> sampl.lm <- lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, sample1.df)
> summary(sampl.lm)

Call:
lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, data = sample1.df)

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

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.778977   0.169566  -16.389 < 2e-16 ***
x1           0.862742   0.008341  103.429 < 2e-16 ***
x2           0.547078   0.007593   72.050 < 2e-16 ***
x3           0.235641   0.008475   27.803 < 2e-16 ***
x4          -0.164290   0.009121  -18.013 < 2e-16 ***
x5          -0.140939   0.009475  -14.874 < 2e-16 ***
x6           0.018872   0.017262   1.093  0.277
x7          -0.081176   0.018421  -4.407 2.84e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.07509 on 92 degrees of freedom
Multiple R-squared:  0.9942,    Adjusted R-squared:  0.9938
F-statistic: 2259 on 7 and 92 DF,  p-value: < 2.2e-16

> RSE.n <- (sum((sample1.df$y - sampl.lm$fitted.values)^2))/nrow(sample1.df)
> RSE.n
[1] 0.005187536
> |

```

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.

```
R Console
File Edit Misc Packages Windows Help

> nrow(sample1.df)
[1] 100
> training.set <- sample1.df[1:50,]
> head(training.set)
  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
> testing.set <- sample1.df[51:100,]
> head(testing.set)
  s.id  p.id      y      x1      x2      x3      x4      x5      x6      x7
51   51 183190  9.532576  9.803391  9.898863  9.591129  9.510771 11.179013 10.527221 11.149856
52   52 767464  9.171071  8.798694  9.751533 10.122447  9.713885  8.816620  9.676893  9.025118
53   53 144760 10.753408 10.584856 11.020712 11.188528 12.294643 11.255245 12.831842 11.743193
54   54 704173 11.021633 10.555549 11.191503 11.271303 10.580588 11.475620 10.903804 10.929445
55   55 679179 11.031378 10.494769 11.158354 10.454756 10.892952  9.890623  8.584673  9.004999
56   56 382746  9.847144 10.825381  8.316025 10.048572 10.086709 10.110882 11.503425 11.061784
> |
```

Specify/fit the model with the training set.

```
R Console
File Edit Misc Packages Windows Help

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

Call:
lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, data = training.set)

Residuals:
    Min       1Q   Median       3Q      Max
-0.148511 -0.057115  0.009986  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   0.013389   64.131 < 2e-16 ***
x2           0.542511   0.011220   48.351 < 2e-16 ***
x3           0.244069   0.012332   19.792 < 2e-16 ***
x4          -0.154081   0.012962  -11.887 5.05e-15 ***
x5          -0.148143   0.014783  -10.021 1.05e-12 ***
x6          -0.004515   0.029921   -0.151  0.8808
x7          -0.061135   0.031044   -1.969  0.0555 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.07938 on 42 degrees of freedom
Multiple R-squared:  0.9931,    Adjusted R-squared:  0.992
F-statistic: 865.1 on 7 and 42 DF,  p-value: < 2.2e-16

> |
```

Apply the specified model (coefficients) to predictor values from the testing set to predict (model based) outcome values of the testing set.

```
R Console
File Edit Misc Packages Windows Help

> attach(testing.set)
> model.l$coefficients
(Intercept)      x1      x2      x3      x4      x5      x6      x7
-2.768627071  0.858629942  0.542510968  0.244069326 -0.154081005 -0.148142918 -0.004514858 -0.061134844
> y.hat <- model.l$coefficients[1] + model.l$coefficients[2]*x1 + model.l$coefficients[3]*x2 +
+ model.l$coefficients[4]*x3 + model.l$coefficients[5]*x4 + model.l$coefficients[6]*x5 +
+ model.l$coefficients[7]*x6 + model.l$coefficients[8]*x7
> |
```

Compare these predicted values to the actual values of the outcome in the testing set.

```
R Console
File Edit Misc Packages Windows Help

> t.test(y.hat, y, paired = TRUE)

Paired t-test

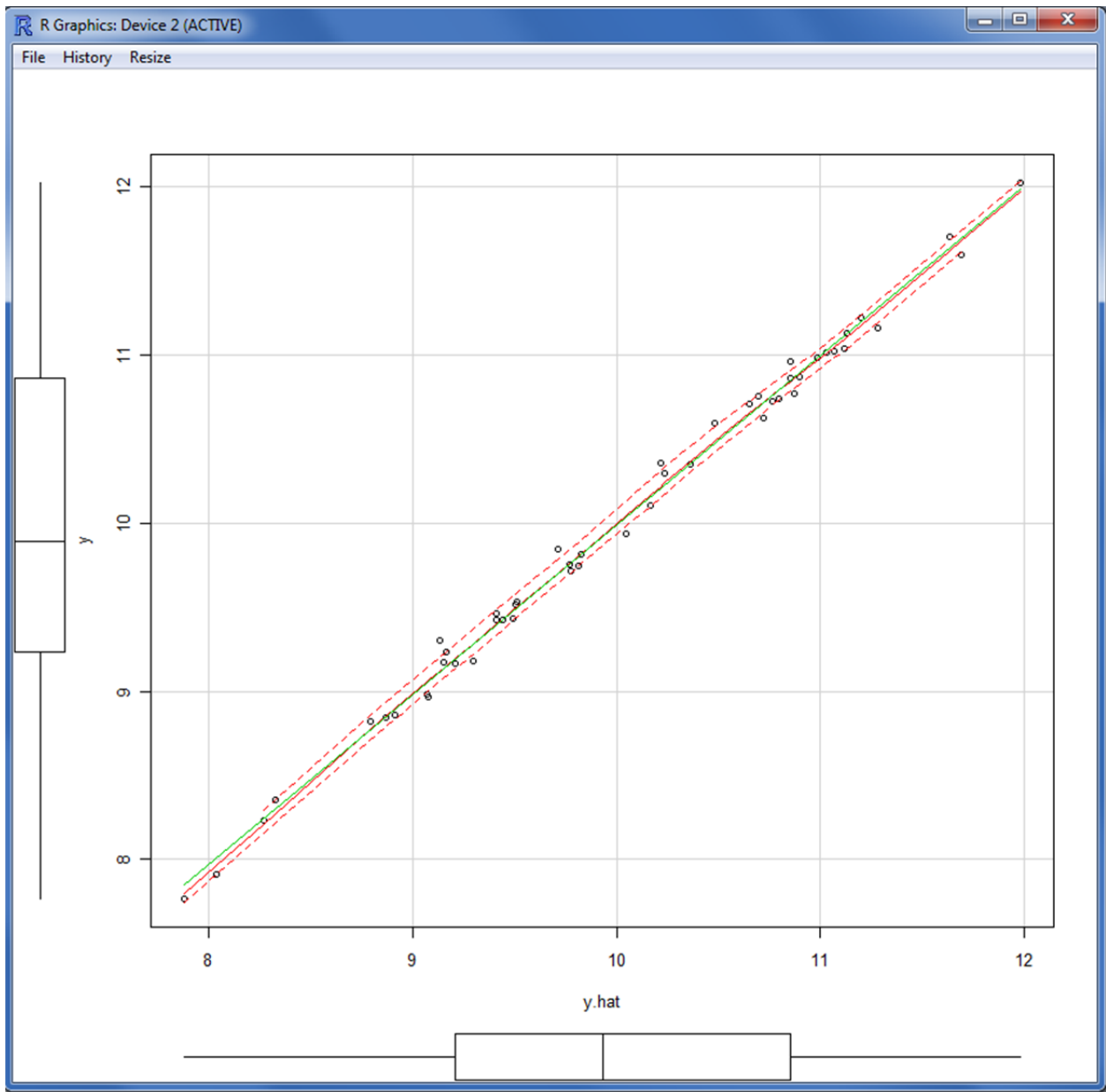
data: y.hat and y
t = 1.1791, df = 49, p-value = 0.244
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.008740101  0.033560446
sample estimates:
mean of the differences
      0.01241017

> cor.test(y.hat, y)

Pearson's product-moment correlation

data: y.hat and y
t = 94.4683, df = 48, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9952602 0.9984871
sample estimates:
      cor
0.9973215

> library(car)|
Loading required package: MASS
Loading required package: nnet
Loading required package: survival
Loading required package: splines
> scatterplot(y.hat, y)
> |
```



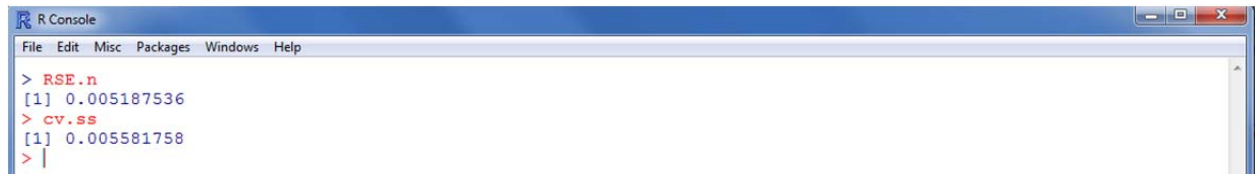
The results above look great; no difference between the two (paired) means, the correlation is virtually +1....perhaps a bit too good to be true?

Calculating the average cross validation sums of squares.

```
R Console
File Edit Misc Packages Windows Help
> cv.ss <- sum((y - y.hat)^2)/length(y)
> cv.ss
[1] 0.005581758
> |
```

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.

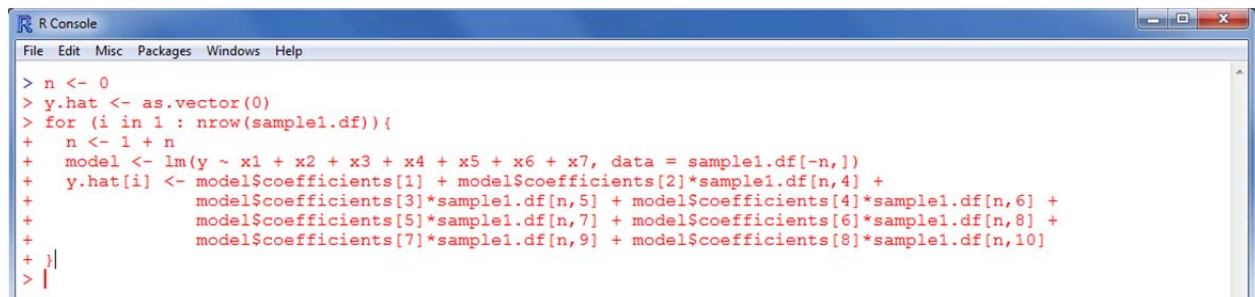


```
R Console
File Edit Misc Packages Windows Help
> RSE.n
[1] 0.005187536
> cv.ss
[1] 0.005581758
> |
```

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'.



```
R Console
File Edit Misc Packages Windows Help
> n <- 0
> y.hat <- as.vector(0)
> for (i in 1 : nrow(sample1.df)){
+   n <- 1 + n
+   model <- lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, data = sample1.df[-n,])
+   y.hat[i] <- model$coefficients[1] + model$coefficients[2]*sample1.df[n,4] +
+     model$coefficients[3]*sample1.df[n,5] + model$coefficients[4]*sample1.df[n,6] +
+     model$coefficients[5]*sample1.df[n,7] + model$coefficients[6]*sample1.df[n,8] +
+     model$coefficients[7]*sample1.df[n,9] + model$coefficients[8]*sample1.df[n,10]
+ }
> |
```

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.



```
R Console
File Edit Misc Packages Windows Help
> cv.ss <- sum((sample1.df$y - y.hat)^2)/1000
> cv.ss
[1] 0.0006070758
> RSE.n
[1] 0.005187536
> |
```

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 Console
File Edit Misc Packages Windows Help
> 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.df$s.id, 50, replace = TRUE)
+   boot.sample <- data.frame(sample1.df[boot.id,])
+   n <- 0
+   y.hat <- as.vector(0)
+   for (k in 1:nrow(boot.sample)){
+     n <- 1 + n
+     model <- lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, data = boot.sample[-n,])
+     y.hat[k] <- model$coefficients[1] + model$coefficients[2]*boot.sample[n,4] +
+       model$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$coefficients[8]*boot.sample[n,10]
+   }
+   errors[i,] <- boot.sample$y - y.hat
+   cv.ss[i] <- (sum(boot.sample$y - y.hat)^2)/50
+   corr.yhat[i] <- cor(boot.sample$y, y.hat)
+ }
> |
```

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

```
R Console
File Edit Misc Packages Windows Help
> summary(errors)
      V1          V2          V3          V4          V5
Min. :-0.1764885  Min. :-0.166096  Min. :-0.163785  Min. :-0.219415  Min. :-0.211913
1st Qu.:-0.0712985 1st Qu.:-0.042399  1st Qu.:-0.047569 1st Qu.:-0.065178 1st Qu.:-0.064804
Median :-0.0008719 Median :-0.010176  Median :-0.008932 Median : 0.007586  Median :-0.029720
Mean  :-0.0014968 Mean  :-0.001436  Mean  :-0.001562 Mean  :-0.003047 Mean  :-0.003012
3rd Qu.: 0.0714475 3rd Qu.: 0.049994  3rd Qu.: 0.077336 3rd Qu.: 0.068355 3rd Qu.: 0.067220
Max.   : 0.1986662 Max.   : 0.173689  Max.   : 0.215780 Max.   : 0.174627 Max.   : 0.192277

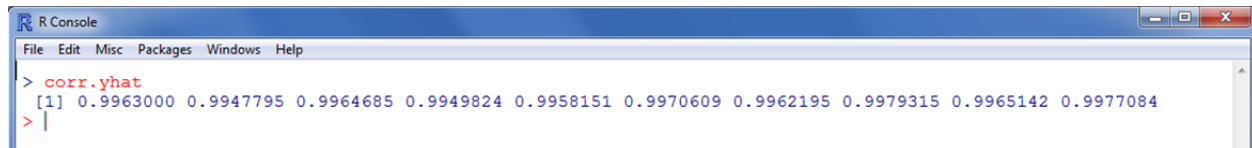
      V6          V7          V8          V9          V10
Min. :-0.145993  Min. :-0.2107229  Min. :-0.1405947  Min. :-0.1554108  Min. :-0.1510409
1st Qu.:-0.057849 1st Qu.:-0.0700161 1st Qu.:-0.0532896 1st Qu.:-0.0256436 1st Qu.:-0.0446621
Median : 0.001898  Median : 0.0132802  Median : 0.0051424  Median :-0.0094489  Median : 0.0197861
Mean  : 0.000227  Mean  :-0.0007376  Mean  :-0.0006105  Mean  :-0.0001666  Mean  :-0.0002541
3rd Qu.: 0.043717 3rd Qu.: 0.0561143 3rd Qu.: 0.0413290 3rd Qu.: 0.0227421 3rd Qu.: 0.0476740
Max.   : 0.140738 Max.   : 0.1537669  Max.   : 0.1167966  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 Console
File Edit Misc Packages Windows Help
> RSE.n
[1] 0.005187536
> cv.ss
 [1] 1.120235e-04 1.030956e-04 1.219923e-04 4.640630e-04 4.534687e-04 2.577242e-06 2.719903e-05 1.863842e-05
 [9] 1.387626e-06 3.228684e-06
> 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.

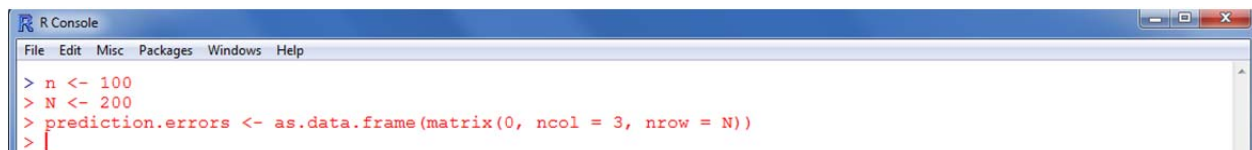


```
R Console
File Edit Misc Packages Windows Help
> corr.yhat
[1] 0.9963000 0.9947795 0.9964685 0.9949824 0.9958151 0.9970609 0.9962195 0.9979315 0.9965142 0.9977084
> |
```

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

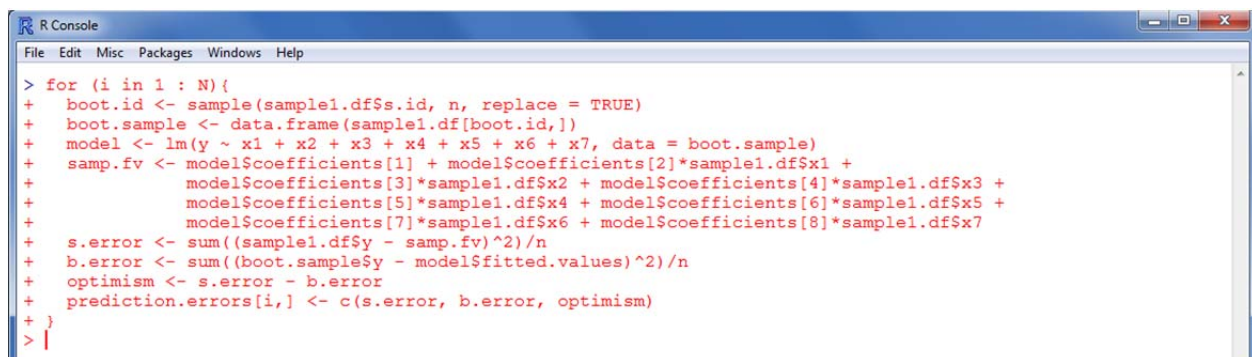
The bootstrapped cross validation approach represents a robust method of estimating the prediction error for a model. Each bootstrapped sample (sampling WITH replacement from the original sample) has the model fit to it and the subsequent coefficients are then used to predict outcome scores of the original 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.errors' data frame stores the original sample errors, the bootstrapped sample errors, and the optimism value from each iteration or loop.



```
R Console
File Edit Misc Packages Windows Help
> n <- 100
> N <- 200
> prediction.errors <- as.data.frame(matrix(0, ncol = 3, nrow = N))
> |
```

Run the bootstrapped cross validation loop.



```
R Console
File Edit Misc Packages Windows Help
> for (i in 1 : N){
+   boot.id <- sample(sample1.df$id, n, replace = TRUE)
+   boot.sample <- data.frame(sample1.df[boot.id,])
+   model <- lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, data = boot.sample)
+   samp.fv <- model$coefficients[1] + model$coefficients[2]*sample1.df$x1 +
+     model$coefficients[3]*sample1.df$x2 + model$coefficients[4]*sample1.df$x3 +
+     model$coefficients[5]*sample1.df$x4 + model$coefficients[6]*sample1.df$x5 +
+     model$coefficients[7]*sample1.df$x6 + model$coefficients[8]*sample1.df$x7
+   s.error <- sum((sample1.df$y - samp.fv)^2)/n
+   b.error <- sum((boot.sample$y - model$fitted.values)^2)/n
+   optimism <- s.error - b.error
+   prediction.errors[i,] <- c(s.error, b.error, optimism)
+ }
> |
```

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).

```
R Console
File Edit Misc Packages Windows Help
> names(prediction.errors)[1] <- "Sample.Error.Estimates"
> names(prediction.errors)[2] <- "Boot.Error. Estimates"
> names(prediction.errors)[3] <- "Optimism"
> summary(prediction.errors)
Sample.Error.Estimates Boot.Error. Estimates Optimism
Min. :0.005213 Min. :0.003000 Min. : -0.0012029
1st Qu.:0.005453 1st Qu.:0.004396 1st Qu.: 0.0003085
Median :0.005567 Median :0.004838 Median : 0.0007595
Mean :0.005617 Mean :0.004847 Mean : 0.0007696
3rd Qu.:0.005726 3rd Qu.:0.005281 3rd Qu.: 0.0012385
Max. :0.006751 Max. :0.006684 Max. : 0.0026862
> |
```

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

```
R Console
File Edit Misc Packages Windows Help
> avg.s.pe <- mean(prediction.errors[,1])
> avg.s.pe
[1] 0.005616914
> avg.b.pe <- mean(prediction.errors[,2])
> avg.b.pe
[1] 0.004847358
> avg.optimism <- mean(prediction.errors[,3])
> avg.optimism
[1] 0.0007695554
> RSE.n
[1] 0.005187536
> |
```

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.

```
R Console
File Edit Misc Packages Windows Help
> avg.optimism + RSE.n
[1] 0.005957091
> |
```

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".

```
R Console
File Edit Misc Packages Windows Help
> sample2.df <- read.table("http://www.unt.edu/rss/class/Jon/R_SC/Module9/CrossValidation/cv_sample2.df.txt",
+ header=TRUE, sep=",", na.strings="NA", dec=".", strip.white=TRUE)
> head(sample2.df)
  s.id  p.id      y      x1      x2      x3      x4      x5      x6      x7
1    1 582317 10.501239 9.992421 10.287485 9.812357 9.248713 8.622629 7.979130 7.829725
2    2 209790 10.522875 10.005652 10.278167 10.955029 10.173412 9.961691 7.876121 8.218935
3    3 248466 10.729997 11.288404 10.135821 9.367200 10.407756 11.996042 8.507795 9.236061
4    4 932425 10.830438 11.150997 9.922778 9.234003 10.843946 9.300424 8.399428 8.460402
5    5 431021 10.117095 9.960406 10.558703 10.214714 10.614445 10.637828 8.977879 9.691200
6    6 719027 9.806795 9.262890 11.454777 9.304795 9.736196 10.629906 11.759719 11.921484
> summary(sample2.df)
      s.id      p.id      y      x1      x2      x3
Min.   : 1.00   Min.   : 2272   Min.   : 6.482   Min.   : 6.549   Min.   : 7.024   Min.   : 7.590
1st Qu.: 25.75  1st Qu.:209173   1st Qu.: 9.333   1st Qu.: 9.627   1st Qu.: 9.479   1st Qu.: 9.289
Median : 50.50  Median :386911   Median :10.227   Median :10.135   Median :10.090   Median : 9.844
Mean   : 50.50  Mean   :443510   Mean   :10.030   Mean   :10.048   Mean   : 9.960   Mean   : 9.998
3rd Qu.: 75.25  3rd Qu.:662813   3rd Qu.:10.757   3rd Qu.:10.629   3rd Qu.:10.533   3rd Qu.:10.573
Max.   :100.00  Max.   :994411   Max.   :12.115   Max.   :12.355   Max.   :12.359   Max.   :13.202
      x4      x5      x6      x7
Min.   : 6.951   Min.   : 7.453   Min.   : 7.113   Min.   : 7.416
1st Qu.: 9.366   1st Qu.: 9.275   1st Qu.: 9.325   1st Qu.: 9.209
Median :10.084   Median :10.018   Median : 9.936   Median :10.091
Mean   : 9.993   Mean   : 9.978   Mean   : 9.955   Mean   : 9.978
3rd Qu.:10.812   3rd Qu.:10.677   3rd Qu.:10.683   3rd Qu.:10.697
Max.   :13.145   Max.   :12.163   Max.   :12.328   Max.   :12.234
> |
```

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.

```
R Console
File Edit Misc Packages Windows Help
> summary(samp1.lm)

Call:
lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, data = sample1.df)

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

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.778977    0.169566  -16.389 < 2e-16 ***
x1           0.862742    0.008341  103.429 < 2e-16 ***
x2           0.547078    0.007593   72.050 < 2e-16 ***
x3           0.235641    0.008475   27.803 < 2e-16 ***
x4          -0.164290    0.009121  -18.013 < 2e-16 ***
x5          -0.140939    0.009475  -14.874 < 2e-16 ***
x6           0.018872    0.017262    1.093  0.277
x7          -0.081176    0.018421   -4.407 2.84e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.07509 on 92 degrees of freedom
Multiple R-squared:  0.9942,    Adjusted R-squared:  0.9938
F-statistic: 2259 on 7 and 92 DF,  p-value: < 2.2e-16

> samp1.lm$coefficients
(Intercept)      x1      x2      x3      x4      x5      x6      x7
-2.77897747  0.86274221  0.54707821  0.23564062 -0.16428995 -0.14093930  0.01887163 -0.08117551
> |
```

Applying the coefficients to the new predictor variable data to create the 'predicted' values of the outcome (y.hat).

```

R Console
File Edit Misc Packages Windows Help
> y.hat <- samp1.lm$coefficients[1] + samp1.lm$coefficients[2]*sample2.df$x1 + samp1.lm$coefficients[3]*samp1$
+   samp1.lm$coefficients[4]*sample2.df$x3 + samp1.lm$coefficients[5]*sample2.df$x4 +
+   samp1.lm$coefficients[6]*sample2.df$x5 + samp1.lm$coefficients[7]*sample2.df$x6 +
+   samp1.lm$coefficients[8]*sample2.df$x7
> summary(y.hat)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 6.39   9.33   10.18   10.02  10.79   11.97
> |

```

Comparison of the 'predicted' values (y.hat) to the actual (new) values of the outcome (sample2.df\$y).

```

R Console
File Edit Misc Packages Windows Help
> summary(y.hat)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 6.39   9.33   10.18   10.02  10.79   11.97
> summary(sample2.df$y)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 6.482  9.333  10.230  10.030  10.760  12.110
> mean(y.hat)
[1] 10.02455
> mean(sample2.df$y)
[1] 10.03038
> var(y.hat)
[1] 1.069014
> var(sample2.df$y)
[1] 1.069242
> sd(y.hat)
[1] 1.033932
> sd(sample2.df$y)
[1] 1.034042
> t.test(y.hat, sample2.df$y, paired = TRUE)

    Paired t-test

data:  y.hat and sample2.df$y
t = -0.8159, df = 99, p-value = 0.4165
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.020007830  0.008348458
sample estimates:
mean of the differences
 -0.005829686

> cor.test(y.hat, sample2.df$y)

    Pearson's product-moment correlation

data:  y.hat and sample2.df$y
t = 142.9945, df = 98, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9964470 0.9983956
sample estimates:
cor
0.9976122
> |

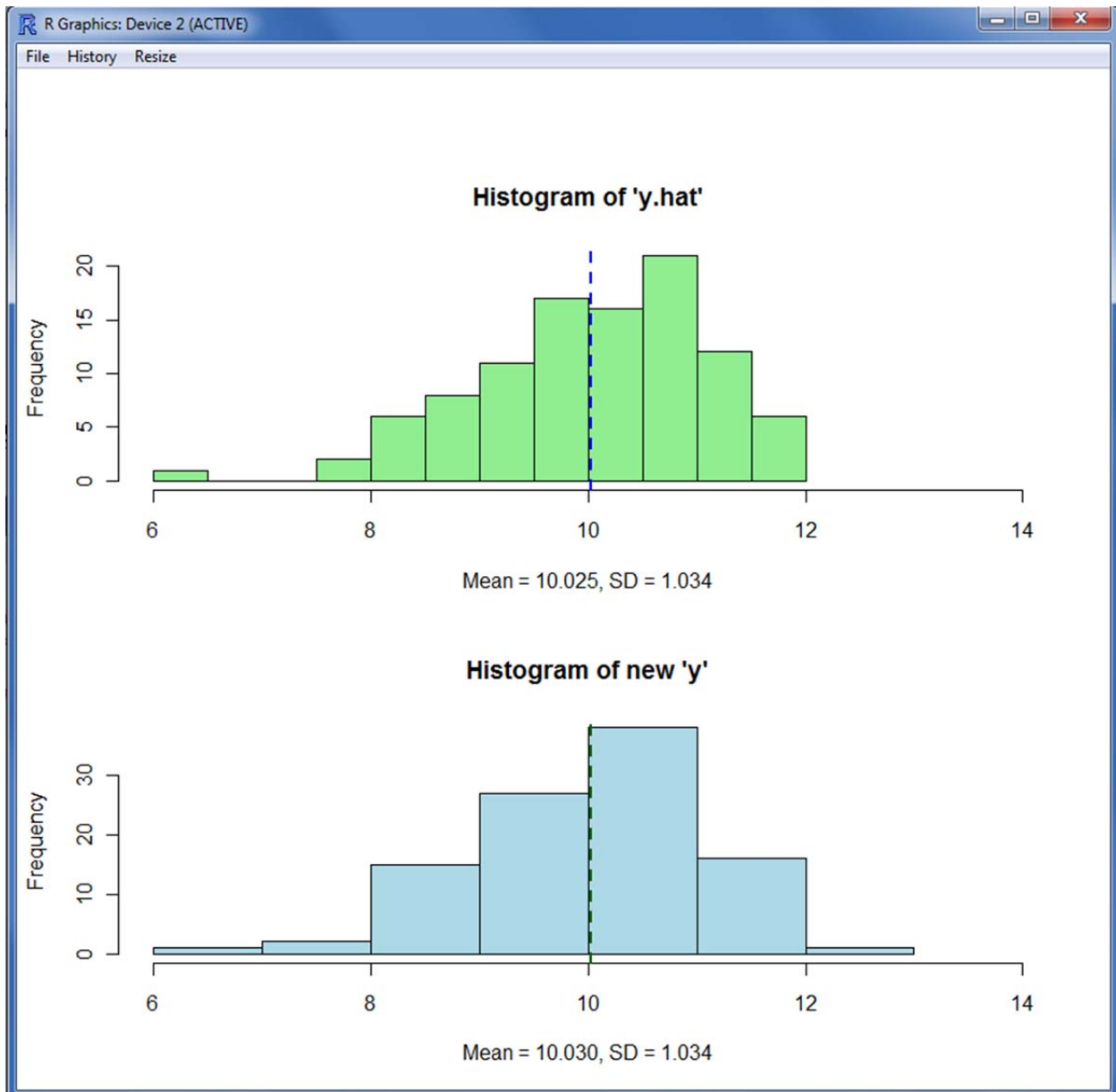
```

The following creates two histograms (in one graphics window); showing y.hat and y.

```

R Console
File Edit Misc Packages Windows Help
> oldpar <- par(oma=c(0,0,3,0), mfrow=c(2,1))
> hist(y.hat, col = "lightgreen", xlim = c(6,14), main = "Histogram of 'y.hat'", xlab = "Mean = 10.025, SD = $
> abline(v = mean(y.hat), col = "blue", lwd = 2, lty = "dashed")
> hist(sample2.df$y, col = "lightblue", xlim = c(6,14), main = "Histogram of new 'y'", xlab = "Mean = 10.030,$
> abline(v = mean(sample2.df$y), col = "darkgreen", lwd = 2, lty = "dashed")
> par(oldpar)
> |

```



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])`)

```

R Console
File Edit Misc Packages Windows Help
> round(cor(sample1.df[,3:10]), 2)
      y      x1      x2      x3      x4      x5      x6      x7
y  1.00  0.74  0.56  0.13  0.07  0.07 -0.01  0.00
x1  0.74  1.00 -0.04 -0.20  0.16  0.25  0.15  0.22
x2  0.56 -0.04  1.00  0.09  0.12  0.06 -0.11 -0.06
x3  0.13 -0.20  0.09  1.00  0.01 -0.02  0.14 -0.05
x4  0.07  0.16  0.12  0.01  1.00  0.01  0.13  0.06
x5  0.07  0.25  0.06 -0.02  0.01  1.00  0.25  0.43
x6 -0.01  0.15 -0.11  0.14  0.13  0.25  1.00  0.87
x7  0.00  0.22 -0.06 -0.05  0.06  0.43  0.87  1.00
> round(cor(sample2.df[,3:10]), 2)
      y      x1      x2      x3      x4      x5      x6      x7
y  1.00  0.80  0.49  0.29 -0.07  0.02 -0.12 -0.09
x1  0.80  1.00 -0.05  0.17  0.06  0.19  0.01  0.07
x2  0.49 -0.05  1.00 -0.01  0.01 -0.06 -0.16 -0.08
x3  0.29  0.17 -0.01  1.00  0.23  0.21  0.26  0.16
x4 -0.07  0.06  0.01  0.23  1.00 -0.05  0.18  0.07
x5  0.02  0.19 -0.06  0.21 -0.05  1.00  0.25  0.48
x6 -0.12  0.01 -0.16  0.26  0.18  0.25  1.00  0.84
x7 -0.09  0.07 -0.08  0.16  0.07  0.48  0.84  1.00
> |

```

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; $N = 1,000,000$).

```

R Console
File Edit Misc Packages Windows Help
> connection <- url("http://www.unt.edu/rss/class/Jon/R_SC/Module9/CrossValidation/CrossValidation_003.RData")
> load(connection)
> round(cor(population.df[,2:9]), 2)
      y      x1      x2      x3      x4      x5      x6      x7
y  1.0  0.80  0.50  0.20  0.00  0.00  0.00  0.00
x1  0.8  1.00  0.00  0.00  0.15  0.15  0.05  0.10
x2  0.5  0.00  1.00  0.00  0.10  0.05  0.10  0.15
x3  0.2  0.00  0.00  1.00  0.10  0.15  0.15  0.05
x4  0.0  0.15  0.10  0.10  1.00  0.15  0.25  0.25
x5  0.0  0.15  0.05  0.15  0.15  1.00  0.35  0.55
x6  0.0  0.05  0.10  0.15  0.25  0.35  1.00  0.85
x7  0.0  0.10  0.15  0.05  0.25  0.55  0.85  1.00
Warning message:
closing unused connection 3 (gzcon(http://www.unt.edu/rss/class/Jon/R_SC/Module9/CrossValidation/CrossValidat$
> |

```

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.


```

R Console
File Edit Misc Packages Windows Help
> pop.lm <- lm(y ~ x1 + x2 + x3, population.df)
> summary(pop.lm)

Call:
lm(formula = y ~ x1 + x2 + x3, data = population.df)

Residuals:
    Min       1Q   Median       3Q      Max
-1.28465 -0.17852 -0.00006  0.17850  1.31010

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -5.0000000  0.0045902 -1089.3  <2e-16 ***
x1           0.8000000  0.0002646  3023.7  <2e-16 ***
x2           0.5000000  0.0002646  1889.8  <2e-16 ***
x3           0.2000000  0.0002646   755.9  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2646 on 999996 degrees of freedom
Multiple R-squared:  0.93,    Adjusted R-squared:  0.93
F-statistic: 4.429e+06 on 3 and 999996 DF,  p-value: < 2.2e-16

> bad.model <- lm(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, population.df)
> summary(bad.model)

Call:
lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7, data = population.df)

Residuals:
    Min       1Q   Median       3Q      Max
-0.293255 -0.043691  0.000008  0.043602  0.291613

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.666e+00  1.290e-03 -2066.1  <2e-16 ***
x1           8.522e-01  6.610e-05 12891.7  <2e-16 ***
x2           5.344e-01  6.581e-05  8121.0  <2e-16 ***
x3           2.344e-01  6.792e-05  3451.6  <2e-16 ***
x4          -1.711e-01  6.799e-05 -2517.0  <2e-16 ***
x5          -1.224e-01  8.282e-05 -1477.5  <2e-16 ***
x6           4.094e-02  1.325e-04   309.0  <2e-16 ***
x7          -1.018e-01  1.487e-04  -684.5  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06468 on 999992 degrees of freedom
Multiple R-squared: 0.9958,    Adjusted R-squared: 0.9958
F-statistic: 3.4e+07 on 7 and 999992 DF,  p-value: < 2.2e-16

> |

```

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

- 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. Available at: <http://www.cs.cmu.edu/~schneide/tut5/node42.html>
- Venables, W. N., & Ripley, B. D. (2002). Package 'MASS'. Available at CRAN: <http://cran.r-project.org/web/packages/MASS/index.html>
- Wikipedia. (2011). Cross-validation (statistics). Available at: http://en.wikipedia.org/wiki/Cross-validation_%28statistics%29