

# Leave one out Method

*David Leslie*

*Monday, March 24, 2015*

The purpose of the code is to evaluate the effectiveness of the leave one out method for determining the predictive power of each model constructed. In order to accomplish this, the model will be constructed by reducing the data by one sample. The sample that is left out will then be used as a new data point to be predicted by the model constructed. By looking at the differences in the predicted value by the model and the actual value recorded, a greater insight will be gained as to how well the model performs when predicting new data. Let's first start by generating a population, which we will sample from.

```
# Create predictor variables
set.seed(10)
pop = 9000
x1 = rnorm(pop)
x2 = rnorm(pop)
x3 = rnorm(pop)

# Create noise
noise = rnorm(pop,0,3)

# Generate response: additive model plus noise, intercept = 0
y = 2*x1 + x2 + 3*x3 + noise

# Organize predictors and response in data frame
pop_data = data.frame(y, x1, x2, x3)

head(pop_data)
```

```
##           y           x1           x2           x3
## 1 -0.9350841  0.01874617 -0.97718623 -0.9620658
## 2 -5.0348547 -0.18425254 -1.15456052 -0.7386388
## 3 -3.3519495 -1.37133055 -0.05577223  0.4370131
## 4 -6.7564424 -0.59916772  0.61778059 -0.1895484
## 5  1.8298094  0.29454513  1.38595893  0.2462712
## 6  1.6258103  0.38979430  1.72930724 -0.1739482
```

Now that we have our population, let's take a sample of our data and verify that the results generated by the model are reasonable estimates of the true model's coefficients (Remembering that they should be close to the expected values of:  $x_1=2$ ,  $x_2=1$ , and  $x_3=3$ ).

```
# Create sample population
n = 375
samp = pop_data[sample(nrow(pop_data), n), ]
# Create model(s)
mod = lm(y ~ x1 + x2 + x3, data=samp)

summary(mod)
```

```
##
```

```
## Call:
## lm(formula = y ~ x1 + x2 + x3, data = samp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7.8807 -1.9683  0.0068  2.0143  8.3742
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.2321     0.1596  -1.454   0.147
## x1             1.9598     0.1501  13.057 < 2e-16 ***
## x2             1.2881     0.1576   8.172 4.83e-15 ***
## x3             2.9242     0.1590  18.394 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.083 on 371 degrees of freedom
## Multiple R-squared:  0.6039, Adjusted R-squared:  0.6007
## F-statistic: 188.5 on 3 and 371 DF, p-value: < 2.2e-16
```

Looking at the model summary, it appears that calculated coefficients are fairly close to actual coefficients. It also appears that approximately 60% of the variability can be explained by the predictor variables. This R-squared value seems reasonable since there is an extra error term incorporated into the model (noise). Now that we feel confident about the sample taken, let's see how good the models generated will be at predicting new data. In order to do this, let's apply the leave one out method to the data. The two things to note for this method are:

1. How close are the predicted values to the actual values.
2. How much the coefficients vary between models

```
# Altered leaveOneOut script from original
LeaveOneOut = function(data, model) {
  actual = data[,1]
  matrixCoef = matrix(, nrow=0, ncol=ncol(data))
  predicted = NULL
  for(i in 1:nrow(data)) {
    trainSet = data[-i,]
    testSet = data[i,]
    trainMod = update(lm(model, data=trainSet), data=trainSet)
    assign(paste("modelCoef", i, sep=""), coef(trainMod))
    matrixCoef = rbind(get(paste("modelCoef", i, sep="")), matrixCoef)
    predicted[i] = predict(trainMod, newdata = testSet)
  }
  finalMatrix = cbind(matrixCoef, predicted, actual)
  return(finalMatrix)
}

resultsMatrix = LeaveOneOut(samp, y ~ x1 + x2 + x3)

head(resultsMatrix)
```

```
##      (Intercept)      x1      x2      x3 predicted      actual
```

```
## [1,] -0.2203477 1.962363 1.285973 2.908068 0.7098677 -0.85620190
## [2,] -0.2366035 1.955742 1.285119 2.933857 1.2452626 6.39325405
## [3,] -0.2214554 1.960618 1.281139 2.925090 -1.5478888 0.44202209
## [4,] -0.2383431 1.965600 1.288650 2.933715 1.9310070 0.03740653
## [5,] -0.2343209 1.962633 1.290189 2.922827 4.6747159 4.42426709
## [6,] -0.2221818 1.959027 1.296770 2.941593 -2.4242752 -3.96262376
```

From the results, it appears that the coefficients are not varying too much, which would suggest that the models are not overfitting the data. However, the differences in the predicted and actual values do vary significantly with each model prediction. Given this variability, model selection may be more difficult than originally expected.