The Signal and the Noise in Cost Estimating

Christian B. Smart, Ph.D.

Abstract

We seek to extract signal and eliminate noise when building models with historical data to predict future costs. One common problem is normalization, which is necessary when making comparisons but can inject noise when used in modeling. We discuss kernel smoothing and distribution fitting as ways to avoid overfitting peculiarities in historical data and the important issues of cross-validation and parsimony as ways to validate models and avoid the lure of overfitting.

Introduction

We seek to extract signal and eliminate noise when building models with historical data to predict future costs. However, there are many pitfalls in this process that can lead you to confuse signal with noise. Overfitting is a common problem that interferes with the attempt to develop accurate predictions. There is a tendency to want to use all the available data for modeling, and to include many parameters. This is appealing since it allows you to account for many different factors in a model which gives you the feeling that your predictions will be more accurate because you can account for a variety of influences – scope, technical parameters, programmatic parameters, heritage, etc. The addition of these parameters is also appealing because it makes the model easier to sell to decision makers, and it makes the model more appealing to consumers of canned models. The more inputs you include, the more the end user feels that they have control of the prediction, and they often feel more comfortable with such a model.

However actual data, like life, is often messy. The final outcome of an event, such as the actual cost of a historic program, is subject to influences that will repeat themselves in a foreseeable way in the future, but it is also subject to a great deal of noise that will not repeat itself in the future in a predictable manner. For example, the Space Shuttle Challenger disaster in the 1980s increased the cost of several satellite programs, since some programs had find other means for launch to space. Occasionally labor strikes occur at prime contractor facilities. And some of the noise in the data is pure error – reported actuals are sometimes wrong, either at the total level, or some of the lower level elements are mis-allocated. In collecting historical costs, estimators are often like forensic investigators, trying to solve a mystery and put together a story that makes sense. This involves much guesswork and requires assumptions and guesses that lead to some amount of distortion of the true historical cost. A recent paper discusses some of the challenges of collecting and validating contractor cost data (Petty et al., 2015). All these events are embedded in the cost of these programs, but are not a part of the cost that can be accurately forecasted going forward.

Despite this there is a tendency to try to explain all the variation in historical data, including the noise. This leads to too many independent variables. The famous mathematician, physicist, and computer scientist John von Neumann once said, “with four parameters I can fit an elephant, and with five I can make him wiggle his trunk.” (Dyson 2004)

This also leads to trying too many different types of equations and other approaches to estimating, all of which reduces the number of degrees of freedom. Small data sets only exacerbate this issue. In small data sets, you can find patterns where none reliably exist.

We present three solutions for avoiding overfitting: keeping the number of independent variables in your models small relative to the number of data points, splitting the data set into training and validation subsets, and cross-validation.

Another common problem with confusing the signal and the noise in cost estimating is normalization, which is necessary when making comparisons but can inject noise when used in modeling. We discuss this issue, and provide an example of how it can lead to a degradation in the quality of a model.

We also propose kernel smoothing and distribution fitting as ways to avoid overfitting errors distribution in small data sets.

Overfitting

Nate Silver calls overfitting “the most important scientific problem you’ve never heard of.” Silver defines overfitting as mistaking noise for signal. (Silver 2012). If the fit is too loose, the model is underfit, which means you are not capturing as much of the signal as possible. On the other hand, the fit can be too good to be true. This is the problem of an overfit model, which means you are fitting the noise in the data rather than discovering the underlying structure of the data. Such models will have great fit statistics, but will not predict costs of future missions well. Overfitting is attractive because it makes the fit statistics and significance tests look great; thus in practice, overfitting is more common than underfitting.

To better understand overfitting, consider the following simple example. We start with a pure signal, $y = x^{1.5}$, as shown in Figure 1.

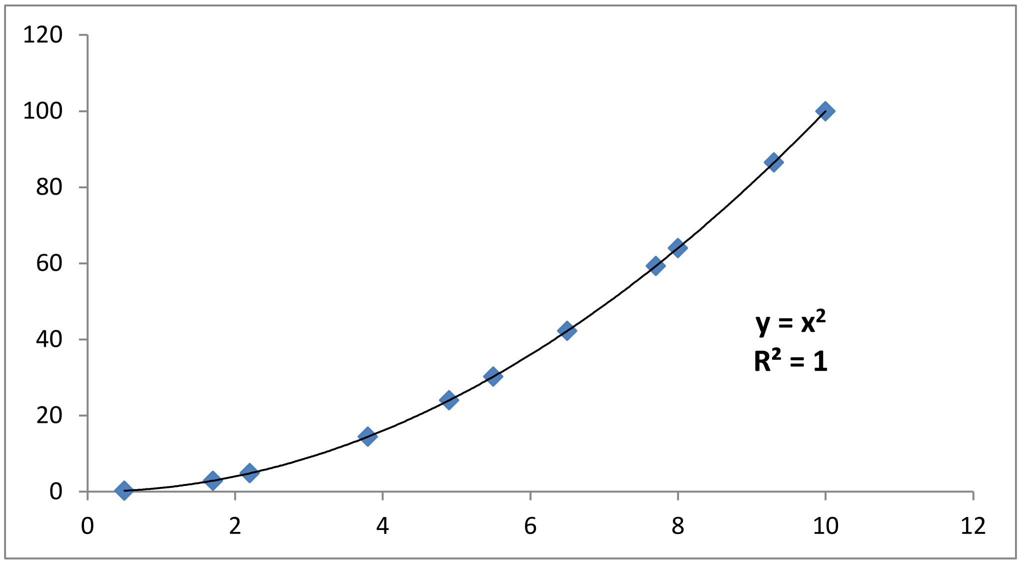

Figure 1. Example of Pure Signal.

We then add random noise to each data point. In order to better fit this noise, we add degrees to the polynomial, resulting in a sixth-degree polynomial, with a $R^2 = 98\%$. This is illustrated in Figure 2.

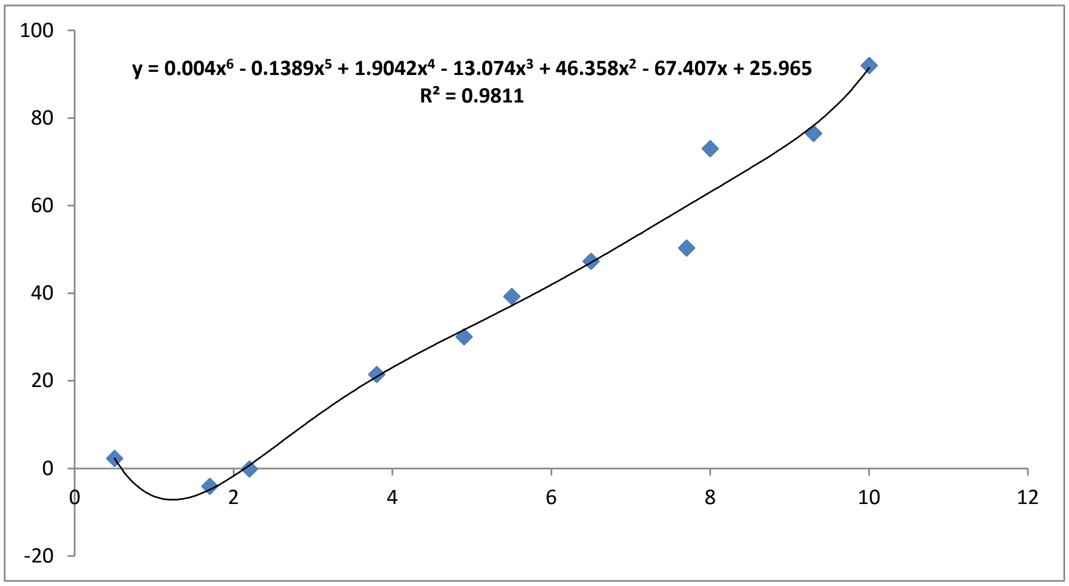

Figure 2. Sixth-degree polynomial fit to noisy data.

But how well does this sixth-degree polynomial mimic the true signal, versus a simpler second-degree polynomial fit? The sixth-degree polynomial better fits the noisy data, but what more closely represents the true signal? To do that, we compare the fits on the noisy data with the true underlying signal. This is illustrated in Figure 3. Just doing a visual comparison, the simpler model fits the underlying true signal better. The second-degree fit is closer to the true signal for

10 of 11 data points. The Pearson's R^2 between the second-degree polynomial and the true underlying signal is much better as well – 98% versus 88% for the sixth-degree polynomial.

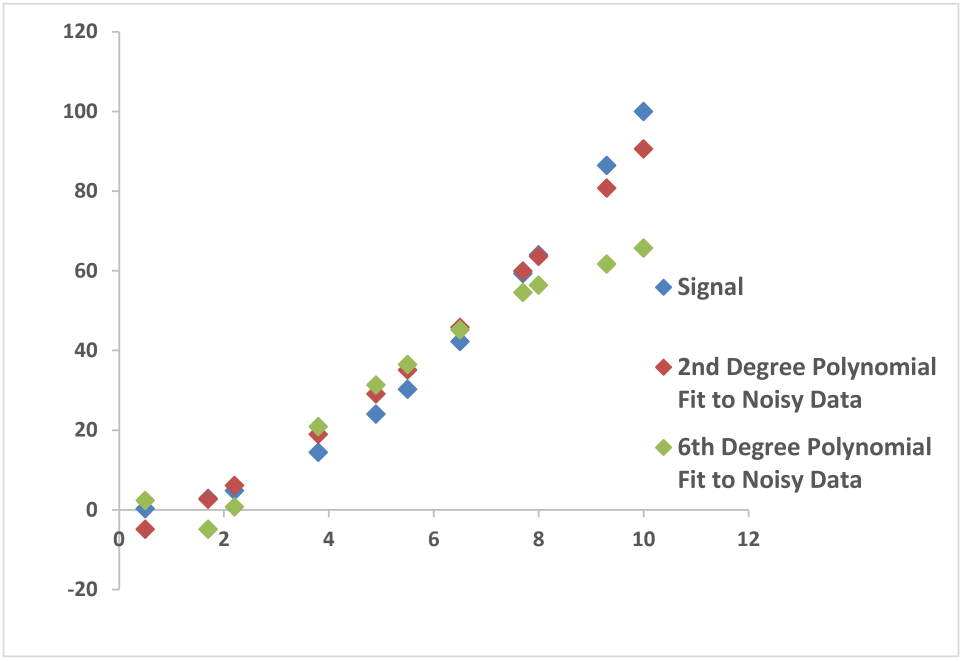

Figure 3. Comparing the fits to noisy data with the underlying signal.

If we only had access to the true signal, we would not be misled by the noisy data. Alas, we can only see the noisy data. As Nassim Taleb writes in *Fooled by Randomness* (2004), in the real world, we have to work by induction, which means we have to infer the structure from the available evidence. Taleb, as well as Nate Silver in The Signal and the Noise, note that you are most likely to overfit a model when the data are limited and noisy and when your understanding of the fundamental relationships is poor. (Silver 2012)

When there is a small number of data points is that it might be possible to think that you have found a signal when there is only noise. In Figure 4 below, there is a strong correlation between the variate and its covariate.

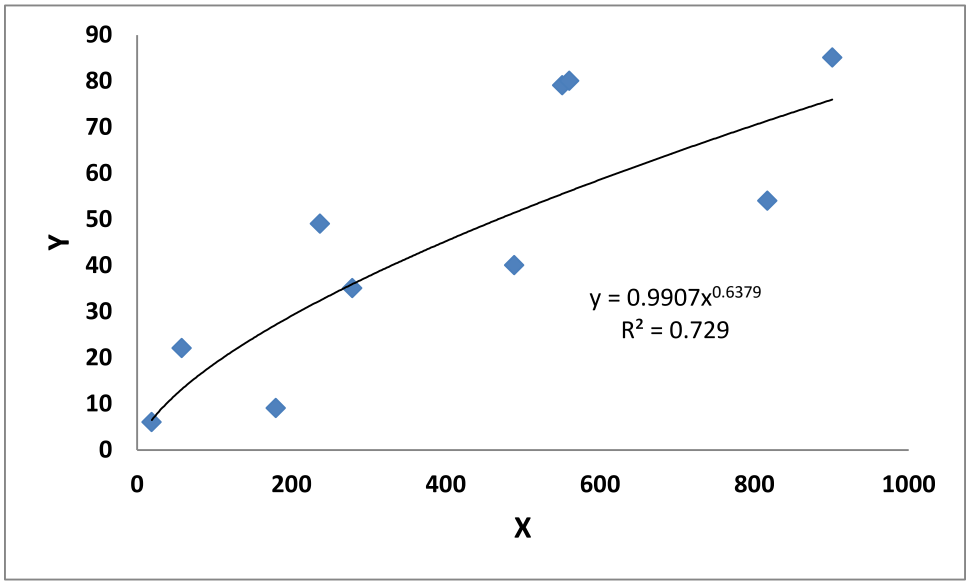

Figure 4. There is a strong correlation between these two variables.

It's only 10 data points but surely there is a strong connection between these two variables, right? There is a clear upward pattern. But the truth is that I randomly generated these 10 points. You wouldn't likely see this in larger data sets, but it is easier to find these in small data sets.

This is only the case for two variables, but adding more randomly generated variables in a small data set allows for even better ostensible fits. Consider the data set displayed in Table 1.

Y	X_1	X_2	X_3	X_4
36	328	482	3	18%
51	124	351	2	70%
41	210	264	3	40%
17	822	99	2	27%
5	255	373	1	92%
11	554	457	7	32%
98	373	24	8	25%
35	551	350	3	6%
46	180	80	9	74%
70	88	250	3	45%

Table 1. 10 randomly generated data points.

The data in Table 1 were randomly generated between specific values. The "Y" values are random numbers between 1 and 100, for example. Regressing Y on X_1 results in an R^2 of only 16%, but adding more variables significantly improves this fit.

Adding four variables with 10 data points drives the R^2 up to 92% and the standard error down to 39%.

Variables	R^2	SE
X_1	17%	87%
X_1, X_2	44%	76%
X_1, X_2, X_3	53%	75%
X_1, X_2, X_3, X_4	92%	39%

Table 2. Fitting random noise can result in a good fit.

Again this is random data, but the fit looks great. This is due to the fact that we have a small sample size, and we have used four variables. Indeed with nine variables we could use up all the degrees of freedom and explain all the variation in the data.

In a large set of 100 random data points of pairs of variables that are uncorrelated there are several small 10-point samples that have clear patterns, as in Figure 5.

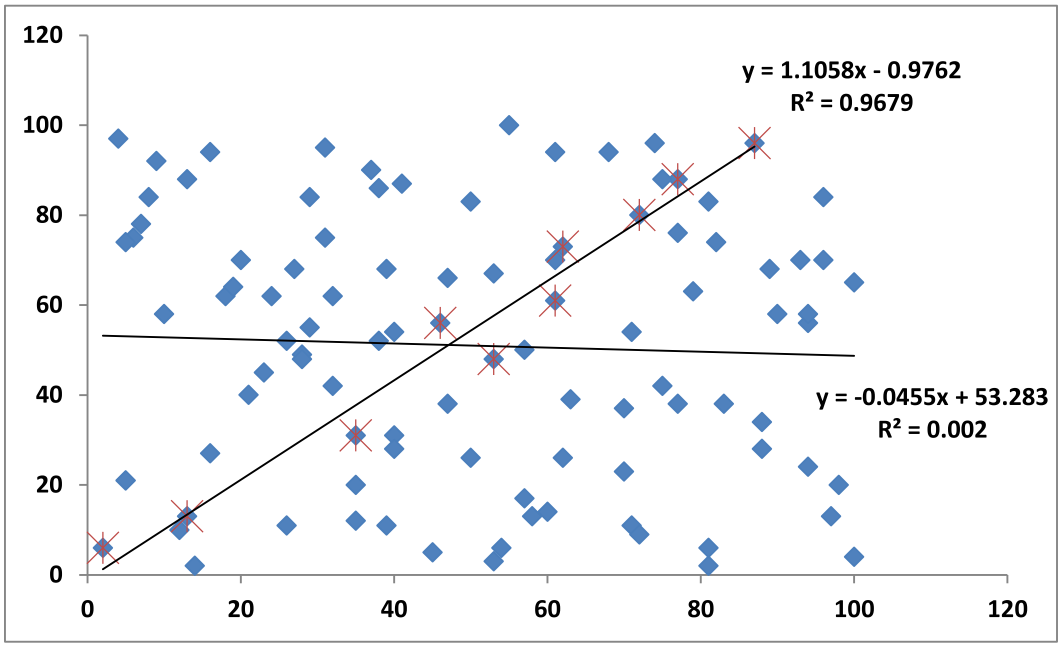

Figure 5. A population of 100 random data points, with a small sample that displays a "trend."

In Figure 5, the overall correlation for all 100 data points is 0.2%, but it is easy to find a sample of 10 data points that are strongly correlated.

Correlations between variables that have no connection are referred to as "spurious correlations." It is easy to find spurious correlations for small data sets. Tyler Vigen has built a website and has published a book devoted to the subject. It shows that there are high correlations between seemingly unrelated statistics, such as the yearly number of math PhDs awarded and the amount of uranium stored at US nuclear power plants. There is a 95% correlation between these two from 1996 to 2008. There is a 95% correlation between per capita cheese consumption and the number of people who died by becoming entangled in their bedsheets between 2000 and 2009 (Vigen 2015). What these spurious correlations all have in common is that they are small data sets, tend to be around 10-15 data points.

Another popular example of spurious correlation is that when the National Football Conference team wins the Super Bowl, the stock market tends to have an up year, and when the America Football Conference team wins the Super Bowl, the stock market tends to have a down year.

In *Fooled by Randomness*, Nassim Taleb (2004) notes that we are hard wired to look for patterns, even where none exist. He posits that this is due to our hunter-gatherer ancestors where finding patterns in small data sets about potential prey could mean the difference between life and death.

As a practical example, given a set of cost data points, a simple model would be to take the average of those values and use that as a model. For example, for 72 historical attitude control systems for NASA and Air Force satellites, the plot of weight and cost is shown in Figure 6. The data ranges from $74,000 for simple earth-orbiting systems to $800 million for crewed spacecraft. The average of these 72 data points is $24 million. This simple model of using the average is definitely not overfit to the data but it is very likely underfit since it underestimates many of the historical data points by a wide margin and overestimates most of the data. We can do better.

The most common independent variable in government project cost estimating relationships is weight. Weight is not a true cost driver, but it is a very good proxy for the scope of a project. Indeed the term “cost driver” is a problematic term. As Andy Price points out in his recent paper “The Dangers of Parametrics,” “one of the worst terms in parametric cost modeling is ‘cost driver.’ Because our parameters are most often associate or scaling, they no more ‘drive’ cost than your dog can drive your car.” (Prince 2016)

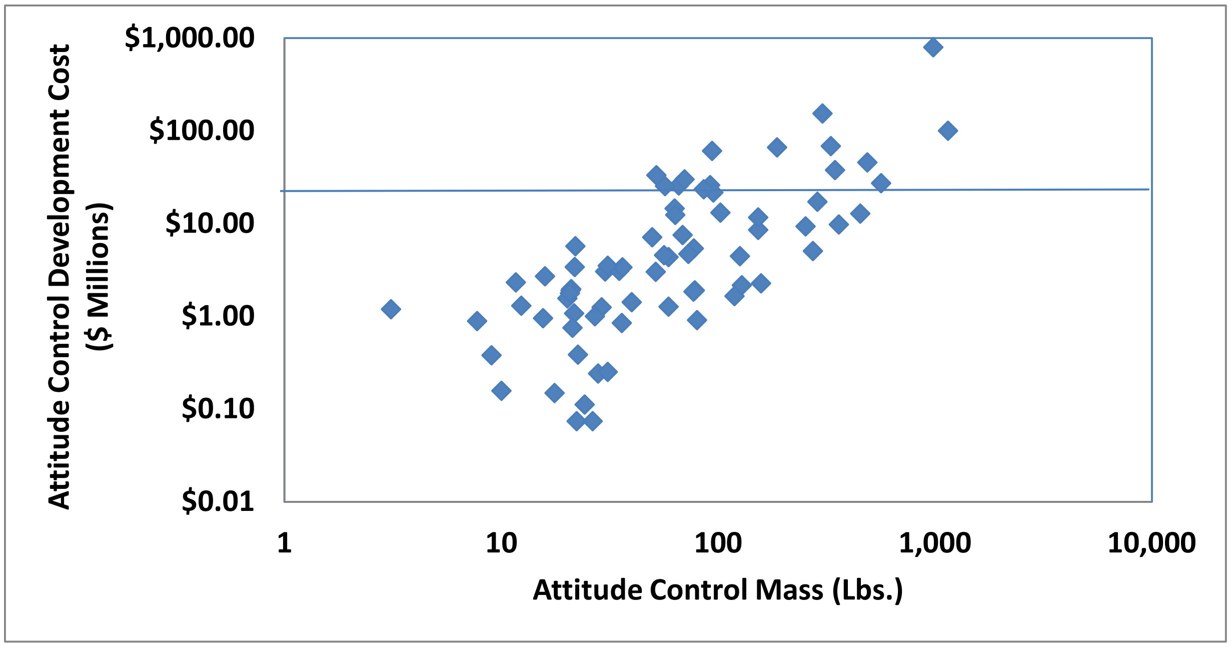

Figure 6. Weight/Cost for 72 historical attitude control systems for NASA and AF spacecraft.

As is evident in Figure 7, going from a simple arithmetic average to a regression model based on weight seems promising. It has a decent R^2 (greater than 50%) and seems to follow the general trend of the data.

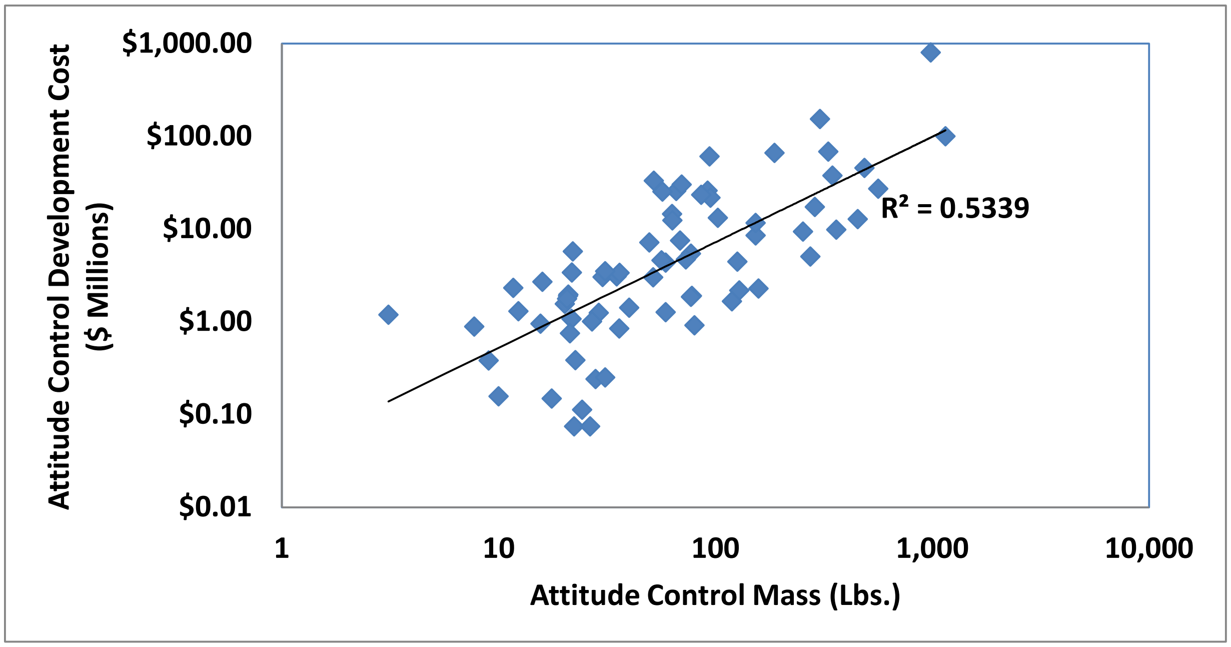

Figure 7. Using a simple weight-based cost estimating relationship for 72 historical attitude control systems for NASA and AF spacecraft.

But can we do better? One option would be to add additional variables. There are many from which to choose and many methods for adding variables to a regression equation. A popular and traditional method is stepwise automated selection of variables. This approach entails starting with the best single variable. One logical criterion for selecting the variable is to choose the one with the highest correlation with the output. We then check if this variable is significant. If not, we quit and adopt the average cost of the data set as our model. Then the variable that improves the goodness of fit the most (e.g. the R^2 or the partial F-value) is added. This process is repeated until the selected goodness-of-fit value fails to increase by a significant amount. This is a long-standing and popular method endorsed by authors of well-known text books such as Draper and Smith's *Applied Regression Analysis* (1998), and has been implemented in commercial statistical software tools including MINITAB. However, this method leads to overfitting. The issue is that each time we sift through the data, which occurs at each step in the stepwise regression, we lose degrees of freedom. Studies of stepwise regression indicate that 30-70% of the predictions included in stepwise regressions are *pure noise*. (Babyak 2004)

Another way to select variables for use in regression is to look at those that are highly correlated with cost and then include only those variables that are above a certain threshold in a multivariate model. This univariate pre-screening of variables is also a form of stepwise regression.

Babyak (2004) describes overfitting as "asking too much from the available data" and "capitalizing on the idiosyncratic characteristics of the sample at hand." The problem with overfitting yields overly optimistic model results – findings that appear in an overfitted model do not exist in the population and will not replicate. Virtually any data-driven decision about modeling will lead to an overly-optimistic model.

One way to avoid overfitting is to limit the number of parameters to a percentage of the sample data size. A typical statistical rule of thumb is 50 data points, plus one parameter for every 10 data points after that. So for one parameter, 50+10*1 = 60 data points are needed. For two parameters, 70 data points are required; for three parameters, 80 data points; and so on. This is the idea of parsimony in modeling – to paraphrase Einstein, keep the model as simple as possible (but no simpler).

In my experience there are not many data sets for government programs that have 50 applicable data points, meaning that in many cases we will need to turn to alternate methods, such a Bayesian regression, to help circumvent this issue (Smart 2014). In our attitude control example, this means we could add a second variable to our number of variables since we have 72 data points. Adding a heritage factor increases the R^2 significantly to 90%. New design is a subjective variable, unlike weight. The use of subjective parameters has its own set of pitfalls, as pointed out in a recent paper by Prince (2016).

Using the stepwise process we can add variables for management rating, mission type, technology maturity, launch year, spin type, and some other miscellaneous parameters to bring the total to 11 significant parameters, all of which are statistically significant as measured by the ratio of the variable's coefficient to its standard error. The overall R^2 increases slightly to 96% but at the potential cost of fitting the model to much of the noise.

In addition to adding variables to a regression model, it is also tempting to try out many different types of models, equation forms, and ways to model. Each attempt to improve the fit of a model, whether it is the addition of a variable, trying a different equation form, or a different approach (traditional regression vice say a neural network), we reduce the number of degrees of freedom available to us, so this also leads to overfitting.

Some ways to avoid overfitting include: collecting more data; combining correlated predictors; clustering; fixing some regression coefficients - e.g., always include weight and heritage in a cost estimating relationship; use of shrinkage and penalization techniques; splitting the data into training and validation sets; and cross-validation. We next discuss these latter two approaches, and apply it to the attitude control subsystem example.

Training, Testing, and Cross-Validation

A popular way to increase the predictive accuracy of a model is to split the data into a training set and a testing set. One widely used rule of thumb is to split one-third of the available data for validation, and use the other two-thirds for training (Mitchell 1998). The model is fit to the training data, and then tested on the validation set. The idea is that the more training you do to try to find the perfect fit for the historical data (adding variables, exploring model forms) the lower your training error will be. But at some point the real-world generalization and predictability of the model reaches an optimum level, and then gets worse as you enter the overfitting zone. This is shown graphically in Figure 8. As can be seen from the graph there is a sweet spot where the complexity is high enough to minimize the testing error, but the model is not so complex that we are overfitting the noise. The key is having the discipline to stop at the sweet spot. Splitting the data into training and testing sets is one way to do that.

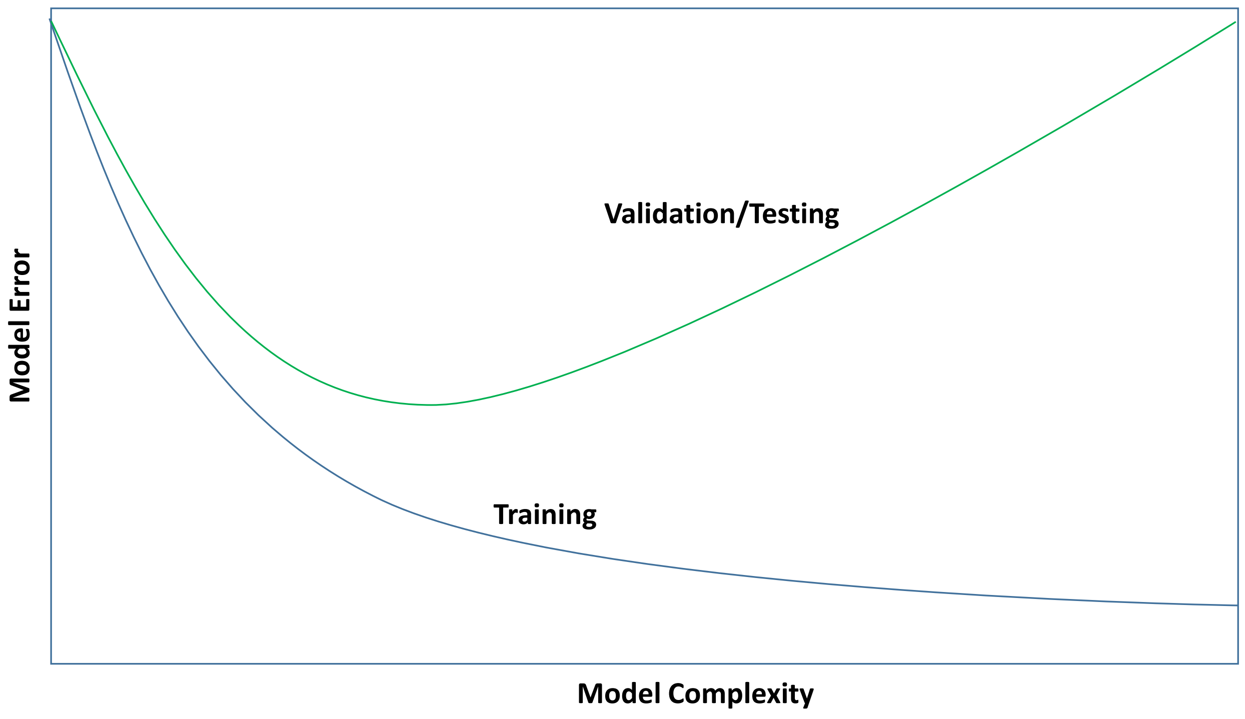

Figure 8. Training vs. testing error as a function of model complexity.

Re-visiting our attitude control example, if we split the data set into a training and a testing/validation set, we use 48 data points for training, and 24 for testing the model fit.

The standard percent error versus the number of independent variables in the regression model for the two sets is shown in Figure 9.

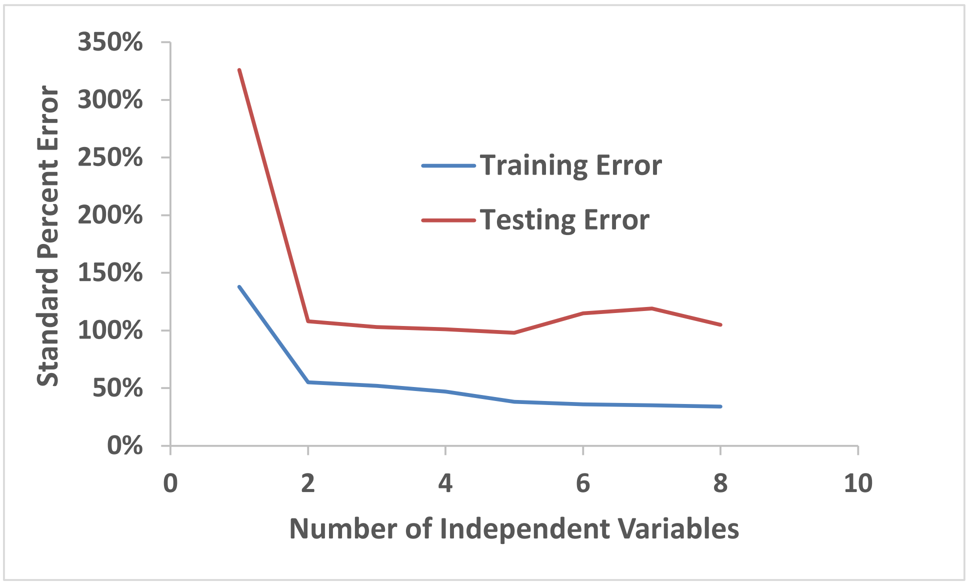

Figure 9. Training and testing error as a function of the number of variables.

We measure the standard percent error for both the training set and the testing set. But we only use the data in the training set to fit the model. The testing set gives you an idea of how well the model will generalize in the real world. The sweet spot in this case is five variables. At this point, the standard percent error for the testing set is 98%, still much higher than the training error at 38%. This is expected - the training error should be lower than the testing error. The testing error is what you are more likely see when using the model in the real world.

This is just a simple example, we have used many more variables than stipulated by the statistical rule of thumb we discussed earlier. The standard would not allow you to use any variables in a 48-element model. Recall this is based on simulation experiments of random data and how many data points you need to have confidence that you are not just fitting a model to noise, so this is a serious consideration. There is a significant chance that you could just be fooling yourself. And our testing set is only a couple of dozen data points, which makes the testing set even more likely to be prone to noise. You could just be lucky in finding a set of variables that fit the testing set well. This is one of those times that you need to balance the science of modeling with the artistic aspect and make a judgment call about what makes the most sense. Using the testing and training split approach does help with avoiding some amount of overfitting, even for relatively small data sets such as the example.

An alternative when the data set is small is to perform cross-validation instead of separating the data into training and testing sets. The idea is simple – split the data set into multiple partitions and do the testing over multiple small partitions and average the results. This has the advantage that you can save more of the data for training.

For the attitude control subsystem example, we use six-fold cross validation. This means we will split the data into six sets of 12 data points, fit the model on 60 data points, and validate on 12, and do this process six times. We then average the results over the six validation sets.

The results of six-fold cross validation are shown in Figure 10. The figure shows the standard percent error for each fold as a function of the number of variables, along with the overall average. The average reaches a minimum on the third variable. This is a little more severe, but probably more realistic, than splitting the data set into a training and validation set.

Once cross-validation has helped you decide to not use more than three variables in your regression model, you can go back and fit the final model using all the data, keeping in mind that the predictive accuracy in practice will be worse than the fit on the sample space. Another option is to notice that for each set of variables, six-fold cross validation has produced six different models. One option would be to average the coefficients from the six different models and use the average model to make predictions. This is a simple form of bagging, a powerful technique for variance reduction. (Wasserman 2005)

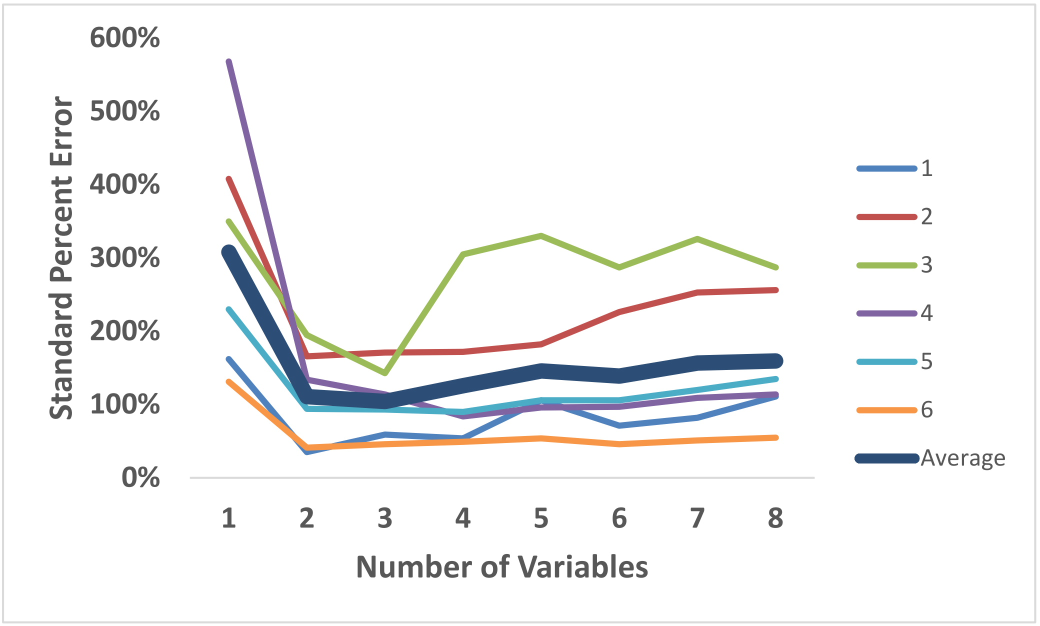

Figure 10. Results of six-fold cross validation.

Cross-validation is a modern technique, and a lot of research has been conducted in this area in the last two decades. There are other novel techniques, such as bootstrap validation, that allow you to use the entire data set for training and still do validation. (Harell 2010)

Normalization vs. Noise-ification

Normalizing data is the process of manipulating raw data to make it comparable with each other. While intended for just comparing data points, it is typically the case that estimators model with normalized data, rather than raw data. This normalization is a source of noise if normalized data are used in modeling. Examples of this include learning, test hardware, and inflation. We begin with a set of data, we then apply some type of linear or nonlinear transformation, and then run a regression on this transformed data. To get back to the original data we then have to apply the transformation in reverse. For inflation, we begin with real or "then year" data, normalize to a constant base year, and develop a model in base year dollars. Then in order to budget we have to convert the model back to real of "then year" cost. In the end we have to convert the model back to "then year." The modeling process does not need the transformation - instead, the information can be used in the model as a variable.

For example, consider inflation. If we wish to compare the cost of a missile designed and built in the 1960s with a missile designed and built in the 2000s, we need to normalize the data to a common base year. The effect of inflation across the decades makes the comparison meaningless otherwise. For example, the average price of a house built in 1950 was less than $9,000 (http://www.thepeoplehistory.com/1950s.html) while in 2016 the average price is $355,000 (https://www.census.gov/construction/nrs/pdf/uspricemon.pdf). To have a meaningful comparison we have to consider inflation, as well as taking into account other changes, such as the fact that the average home today is much bigger than a house built in the 1950s, and has much different amenities.

This is all well and good for comparing historical data points. But it doesn't mean we should model the data after it has been normalized for inflation. Instead of normalizing the data before we model, we should add a variable that accounts for the year or years in which the project was executed and model the impact directly.

For example, applying and modeling the cost of reaction control subsystems for 62 NASA and Air Force missions with weight as the independent variable on normalized data results in the equation $0.17*Weight^{0.74}$. Modeling the non-normalized data with the mid-point of design added as a variable yields the equation $0.07*Weight^{0.85}*Yr\ of\ Tech^{-0.12}$, where *Yr. of Tech = Year of Technology* is defined as the *mid-point of project design – 1960*. The first equation produces a cost in a constant base year, whereas the second equation produces cost in real year dollars, based on the year input variable. Note that the value of the year variable is negative. Why is this, when we know there is a strong and steady uptick in prices every year? The time coefficient reflects the overall real productivity growth over time, which on average exceeds inflation, reducing net costs overall over time, everything else being equal (ceteris paribus). When we deflate the normalized data and compare it to the original, raw data, we find a Pearson's R^2 equal to 30%. When we compare the model on un-normalized data with the raw actual data we find a Pearson's R^2 equal to 39%, a big improvement over the normalized model. The standard error of the normalized data is 358% vs. 278% for the non-normalized model.

The process of normalization when applied to modeling should be called "noise-ification" since it is better to model the raw data directly. Much of this is due to the nonlinearity of the data – if the coefficient of the power equation were equal to 1 then applying a linear filter to the data before and after modeling will have little to no impact. But the application of a linear filter in the presence of nonlinearities, as seen with this example, when introduce noise and error into the equation, reducing predictive capability.

Similarly, manipulating raw data to produce a "theoretical" first unit cost is highly sensitive to assumptions about the learning and rate percentages. A small difference between what is assumed and what is experienced in practice make a big difference in the estimate of cost for production units.

Test hardware has a similar impact. It is often the case that the data are manipulated to adjust for test hardware quantities, but this can be modeled directly as an input, or even better, estimated separately for even better predictive accuracy.

Figure 11 compares the common practice with what should be done in practice to avoid adding more error to the modeling process. The model step in the two approaches is a little more complicated since it involves another variable, but that is at it should be. You should let the data tell you how to account for inflation and productivity, rather than rely on a separate normalization process.

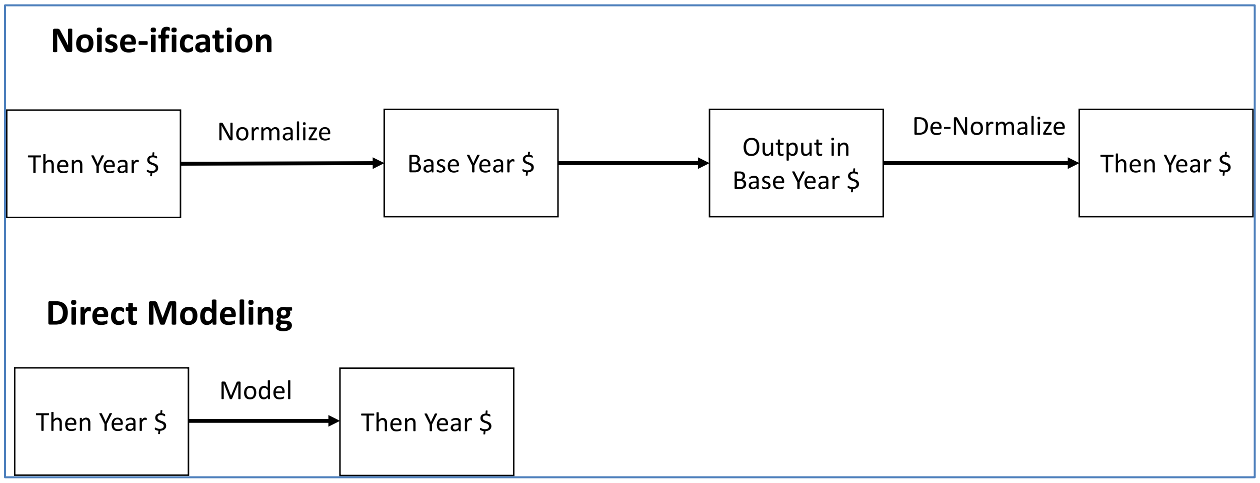

Figure 11. Illustrating the differences between the common noise-ification approach to modeling versus directly modeling inflation in the model.

Boostrap vs. Kernel Smoothing and Distribution Fitting

When you have sufficient data, the bootstrap is a good method for simulating actuals. The bootstrap method, so called because it is akin to “pulling yourself up by your own bootstraps” repeatedly draws samples from a given data set to provide alternate outcomes. It is a method used to compute standard errors and confidence intervals. (Wasserman 2010, Foussier 2006)

For example this has been proposed as a way to develop prediction intervals for non-parametric CER methods, such as the Zero-percent bias Minimum Percent Error (ZMPE) method (Feldman and Springer 2006). However, when there is a small amount of data, there are large gaps in the data that are not realistic when trying to develop prediction intervals. I argue that the use of bootstrapping with small data sets is a form of overfitting to the data. In such cases, the use of kernel smoothing and data fitting is more realistic.

For example, consider the histogram of CER percentage errors for a small data set as shown in Figure 12. For this case, a bootstrap would only simulate from the given data. Even though there is no reason why we could not see a percentage error between 0% and 50%, but the bootstrap would ignore this possibility.

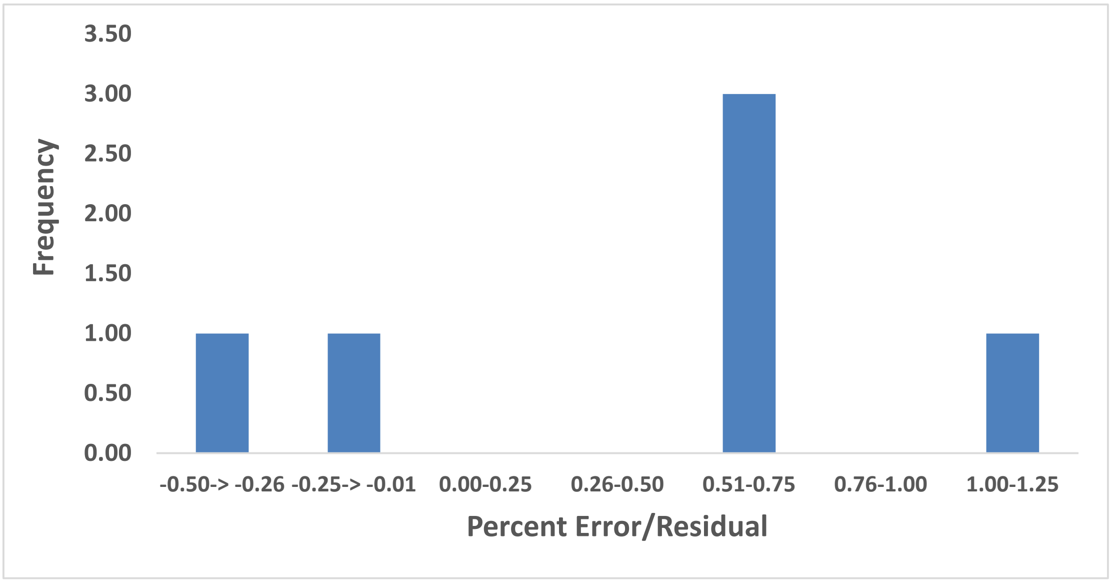

Figure 12. Example of CER percentage errors for a small data set.

Kernel smoothing is a way to spread this histogram to peanut-butter spread some of the error, based on the available data at hand. This produces a continuous distribution that fills in the gaps left by the original discrete histogram. A simple uniform kernel with bandwidth equal to 0.50 produces a density function shown in Figure 13. While simple, and with some jaggedness, this illustrates a way to generalize the data to fill in the gaps left by a small data set. Using the mean and standard deviation of the sample to fit a continuous distribution is also an appealing choice for a small data set.

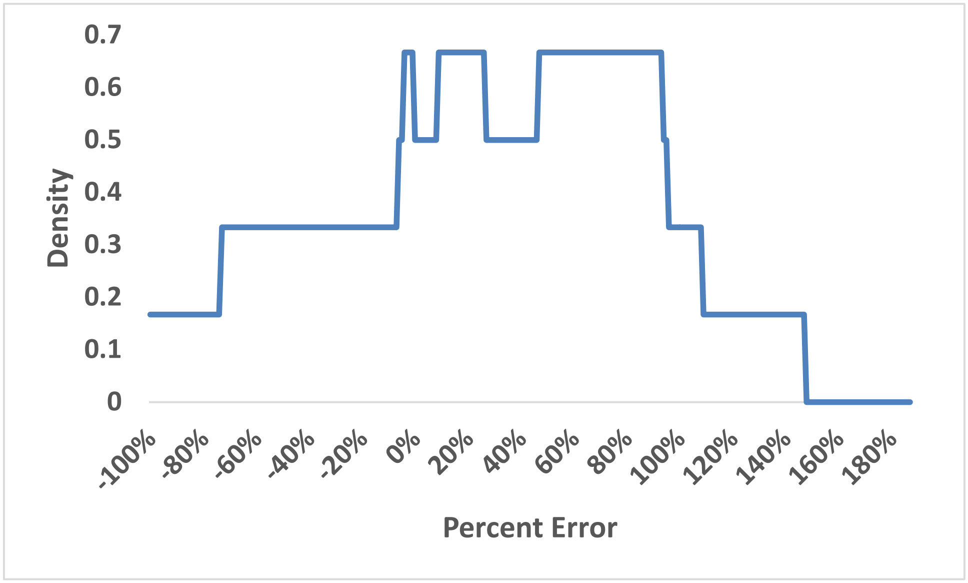

Figure 13. Uniform kernel smoothing with bandwidth = 0.5.

Summary

Prediction is a perilous business. As we have discussed in this paper, the process of developing a model is filled with pitfalls. The lure of overfitting is very powerful, as it is a natural human tendency to want to explain past actual behavior completely. However, the actual data is a combination of signal and noise, and if we are not careful we fit both the noise along with the signal, making our predictions less accurate and reliable. For small data sets, it is easy to confuse noise and signal. We have discussed ways to overcome this, such as keeping the number of independent variables to a minimum, splitting data into training and validation sets, and cross-validation. Cross-validation is particularly powerful, and it should be part of every parametrician's tool kit.

We have also discussed a couple of topics related to overfitting, such as the fact that normalizing the data prior to modeling can lead injecting to additional noise that could be avoided by directly modeling the phenomenon using the data. Another topic is the potential misuse of bootstrapping on small data sets. In such cases we have shown the using kernel smoothing or using the moments to define a continuous distribution can be a better approach.

References

1. Babyak, M.A., "What You See May Not Be What You Get: A Brief, Nontechnical Introduction to Overfitting in Regression-Type Models," Psychosomatic Medicine 66 (Feb. 19, 2004).

2. Draper, N.R., and H. Smith, *Applied Regression Analysis*, 3rd Ed., 1998, John Wiley and Sons, New York.

3. Dyson, F., "A Meeting with Enrico Fermi," Nature 427 (22 January 2004), page 297.

4. Feldman, D., and Springer, S., "Algebraic Formulas for Prediction Bounds on CER-Based Estimates," presented at the 2006 Society of Cost Estimating and Analysis Annual Conference, Tyson's Corner, VA, June 2006.

5. Foussier, P., *From Product Description to Cost: A Practical Approach, Volume 2: Building a Specific Model*, 2006, Springer-Verlag, London.

6. Harrell. F.E., *Regression Modeling Strategies*, 2010, Springer-Verlag, New York.

7. Mitchell, T., *Machine Learning*, 1997, McGraw-Hill, Boston, Massachusetts.

8. Petty, C., C. Smart, and J. Lawlor, "Seven Degrees of Separation," presented at the International Cost Estimating and Analysis Association Annual Conference, June 2015, San Diego, CA.

9. Prince, A., “The Dangers of Parametrics,” presented at the International Cost Estimating and Analysis Association Annual Conference, June 2016, Atlanta, GA.

10. Silver, N., *The Signal and the Noise: Why So Many Predictions Fail – But Some Don’t*, 2012, Penguin Books, New York.

11. Smart, “Bayesian Parametrics: How to Develop a CER with Limited Data and Even Without Data,” presented at the International Cost Estimating and Analysis Association Annual Conference, June 2014, Denver, CO.

12. Taleb, N.N., *Fooled by Randomness*, 2nd ed., 2004, TEXERE, New York.

13. Vygen, T., *Spurious Correlations*, 2015, Hachette Books, New York, and http://www.tylervigen.com/spurious-correlations

14. Wasserman, L., *All of Statistics: A Concise Course in Statistical Inference*, 2005, Springer, New York.