Paper 259-2009

Applications of the GLMSELECT Procedure for Megamodel Selection

Robert A. Cohen, SAS Institute Inc., Cary, NC

ABSTRACT

When you can select regression models from tens of thousands of effects, what possibilities for modeling are open to you? This paper explores applications of the GLMSELECT procedure in SAS/STAT® software to such problems. The GLMSELECT procedure supports a variety of model selection methods for general linear models. Examples of megamodels arising in genomic data analysis and nonparametric modeling are discussed. In addressing these examples, built-in facilities of the procedure to handle validation and test data are highlighted in addition to techniques for extending the procedure's functionality to address model selection bias by using bootstrap-based model averaging.

INTRODUCTION

When you are faced with a predictive modeling problem that has many possible predictor effects—dozens, hundreds, or even thousands—a natural question is "What subset of the effects provides the best model for the data?" Statistical model selection seeks to answer this question, using a variety of definitions of the "best" model in addition to a variety of heuristic procedures for approximating the true but computationally infeasible solution. The GLMSELECT procedure implements statistical model selection in the framework of general linear models. Methods include not only extensions to general linear models of methods long familiar in the REG procedure (forward, backward, and stepwise) but also the newer LASSO and LAR methods of Tibshirani (1996) and Efron et al. (2004), respectively.

Although the model selection question seems reasonable, trying to answer it for real data can lead to problematic pitfalls, including the following:

- Only one model is selected, and even that is not guaranteed to be the "best." Other, more parsimonious or more intuitively reasonable models that might provide nearly as good or even better models might not be found by the particular heuristic method that is used.
- Model selection might be unduly affected by outliers.
- A "selection bias" exists because a parameter is more likely to be selected if it is above its expected value than if it is below its expected value.
- Standard methods of inference for the final model are invalid in the model selection context.

To some degree these pitfalls are intrinsic, and they have even led some experts to stridently denounce model selection. However, certain features of the GLMSELECT procedure—in particular its extensive capabilities for customizing the selection and its flexibility and power in specifying complex potential effects—can partially mitigate these problems.

Model averaging approaches provide a way to make more stable inferences based on a set of models. Methods for doing this are presented in Burnham and Anderson (2002), and Bayesian approaches are discussed in Raftery, Madigan, and Hoeting (1997). However, when confronted with thousands of regressors you still have the issue of how to produce a reasonable set of candidate models to average, and this is where model selection comes into play.

This paper explores two examples of megamodel selection. The first example shows how effect selection can be used in the context of nonparametric modeling, where a large number of regressors are needed to provide the flexibility to model the unknown form of the underlying function. The second example, motivated by microarray data, shows how you can combine large scale variable selection with bootstrap methods to identify important regressors and produce averaged models whose predictive performance is competitive with other state-of-the-art methods in this field.

FEATURES OF PROC GLMSELECT

The main features of the GLMSELECT procedure are as follows:

- **Model Specification**
  - offers different parameterizations for classification effects
  - supports any degree of interaction (crossed effects) and nested effects
  - supports hierarchy among effects
  - provides for internal partitioning of data into training, validation, and testing roles
  - provides an EFFECT statement to generate spline, polynomial, multimember, and collection effects (new in SAS/STAT 9.2)
- **Selection Control**
  - provides multiple methods for effect selection
  - enables selection from a very large number of effects (tens of thousands)
  - offers selection of individual levels of classification effects
  - provides effect selection based on a variety of selection criteria
  - provides stopping rules based on a variety of model evaluation criteria
  - provides leave-one-out and k-fold cross validation
- **Display and Output**
  - produces graphical representation of the selection process
  - produces an output data set that contains predicted values and residuals
  - produces an output data set that contains the design matrix (new in SAS/STAT 9.2)
  - produces macro variables that contain selected models
  - supports parallel processing of BY groups
  - supports multiple SCORE statements

You can find detailed discussions of these features in Cohen (2006).

THE EFFECT STATEMENT

The experimental EFFECT statement, new in SAS/STAT 9.2, enables you to construct special collections of columns for the **X** matrix in your model. These collections are referred to as *constructed effects* to distinguish them from the usual model effects formed from continuous or classification variables.

The following *effect-types* are available:

COLLECTION	is a collection effect that defines one or more variables as a single effect with multiple degrees of freedom. During the selection process, the variables in a collection enter or leave as a unit.
MULTIMEMBER | MM	is a multimember classification effect whose levels are determined by one or more variables that appear in a CLASS statement. A typical example is the effect that teachers have on the performance of students, where each student has been taught by several teachers.
POLYNOMIAL | POLY	is a multivariate polynomial effect in the specified numeric variables.
SPLINE	is a regression spline effect whose columns are univariate spline expansions of one or more continuous variables. A spline expansion replaces the original variable with an expanded or larger set of new variables.

The first example in this paper uses the EFFECT statement to define spline effects. For details and examples of other uses of the EFFECT statement, see Chapter 42, “The GLMSELECT Procedure” (*SAS/STAT User’s Guide*).

NONPARAMETRIC MODELING BY USING SPLINE EFFECTS

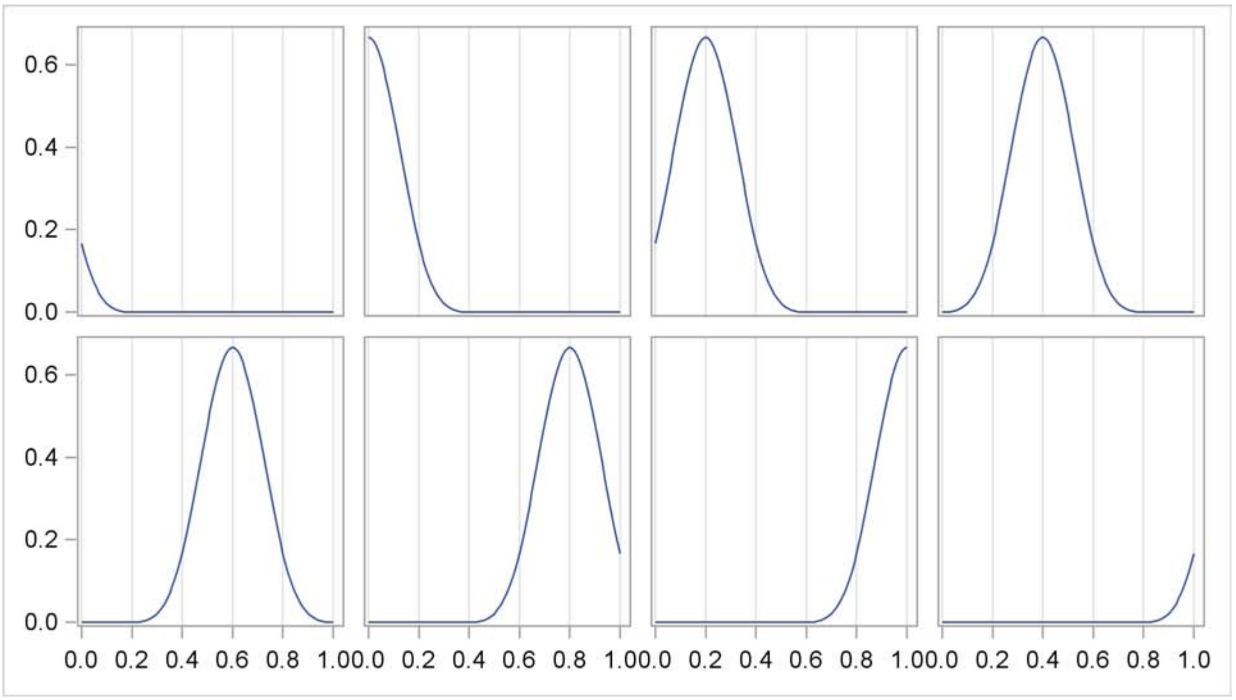
One approach to nonparametric fitting is to approximate the unknown underlying function as a linear combination of a set of basis functions. Once you specify the basis functions, then you can use least squares regression to obtain the coefficients of the linear combination. A problem with this approach is that for most data, you do not know a priori what set of basis functions to use. You need to supply a sufficiently rich set to enable the features in the data to be approximated. However, if you use too rich a set of functions, then this approach yields a fit that undersmooths the data and captures spurious features in the noise. Spline function bases provide a computationally convenient and flexible way to specify a rich set of basis functions, and variable selection can be used to obtain a parsimonious subset to prevent overfitting.

A spline function is a piecewise polynomial function where the individual polynomials have the same degree and connect smoothly at join points whose abscissa values, referred to as knots, are prespecified. By positioning an appropriate number of knots, you can use spline functions to parsimoniously fit curves to a wide variety of data. Although PROC GLMSELECT supports splines of any degree, this paper uses the cubic splines (the default) exclusively. Visually a cubic spline is a smooth curve, and it is the most commonly used spline when a smooth fit is desired.

Given a set of n distinct knots, any cubic spline can be represented as a linear combination of $n + 4$ basis functions. There are two widely used spline bases: namely, truncated power function bases and B-spline bases. Each cubic B-spline basis function has the property that its support (the set at which the function is nonzero) is an interval spanned by five adjacent knots. Hence you can associate local features in your data with particular B-spline basis functions. Truncated power bases do not share this property and are not used in this paper.

Figure 1 shows a cubic B-spline basis defined on $[0, 1]$ with four equally spaced knots at 0.2, 0.4, 0.6, and 0.8. Note that this basis consists of eight functions, each of which is nonzero over an interval that spans at most five knots.

Figure 1 Cubic B-Spline Basis with Four Equally Spaced Interior Knots



The following code simulates scatter plot data where the underlying true function is one period of the sine function $f(x) = \sin(2\pi x)$:

```
data Sine;
   do n=0 to 2048;
      x=n/2048;
      noise      = 0.5*rannor(12345);
      sine       = sin(2*constant('pi')*x);
      noisySine  = sine + noise;
      output;
   end;
run;
```

You can use an EFFECT statement to generate the columns in the design matrix that correspond to the basis shown in Figure 1. By selecting columns from this design matrix you can fit a smooth curve to the data as follows:

```
proc glmselect data=Sine;
   effect spl = spline(x/knotmethod=equal(4) split details);
   model  noisySine = spl;
   output out=sineOutCoarse p=predicted;
run;
```

The EFFECT statement in the preceding code creates a constructed effect named spl that consists of the eight cubic B-spline basis functions that correspond to the four equally spaced internal knots at 0.2, 0.4, 0.6, and 0.8 inside the range $[0, 1]$ of the variable x. Boundary knots outside the range of the data are also needed in defining this basis.

The DETAILS option in the EFFECT statement requests the knot information and the basis function information shown in Figure 2.

Figure 2 B-Spline Basis Information

```
                    The GLMSELECT Procedure

                  Knots for Spline Effect spl

                   Knot
                  Number    Boundary         x

                       1       *          -0.4
                       2       *          -0.2
                       3       *          -0.0
                       4                   0.2
                       5                   0.4
                       6                   0.6
                       7                   0.8
                       8       *           1.0
                       9       *           1.2
                      10       *           1.4

                    The GLMSELECT Procedure

              Basis Details for Spline Effect spl

                                          Support
                Column     --Support-     Knots

                     1      -0.4   0.2    1-4
                     2      -0.4   0.4    1-5
                     3      -0.2   0.6    2-6
                     4      -0.0   0.8    3-7
                     5       0.2   1.0    4-8
                     6       0.4   1.2    5-9
                     7       0.6   1.4    6-10
                     8       0.8   1.4    7-10
```

The SPLIT option enables the eight columns in the design matrix that correspond to this basis to enter or leave the model individually. The individual basis functions in this split effect are named spl_1, spl_2,..., spl_8, and the corresponding labels are spl:1, spl:2,..., spl:8. Because no options are specified in the MODEL statement, the default stepwise selection method is used to select effects that are shown in the selected effects table in Figure 3.

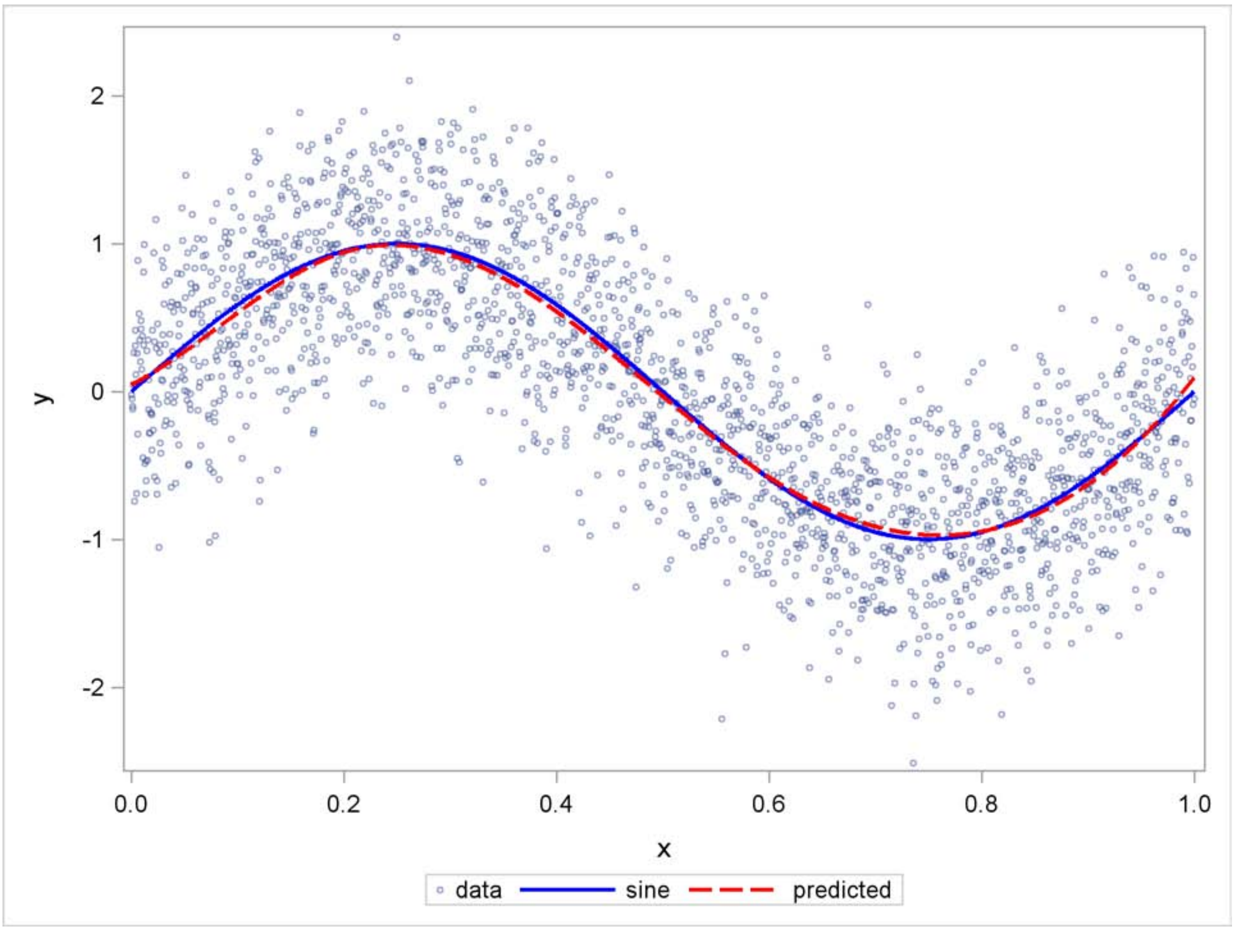
Figure 3 Selected Effects

```
Effects: Intercept spl:3 spl:4 spl:5 spl:6 spl:8
```

The OUTPUT statement captures the predicted values in an output data set. You can produce the fit plot in Figure 4 by using the SGPLOT procedure as follows:

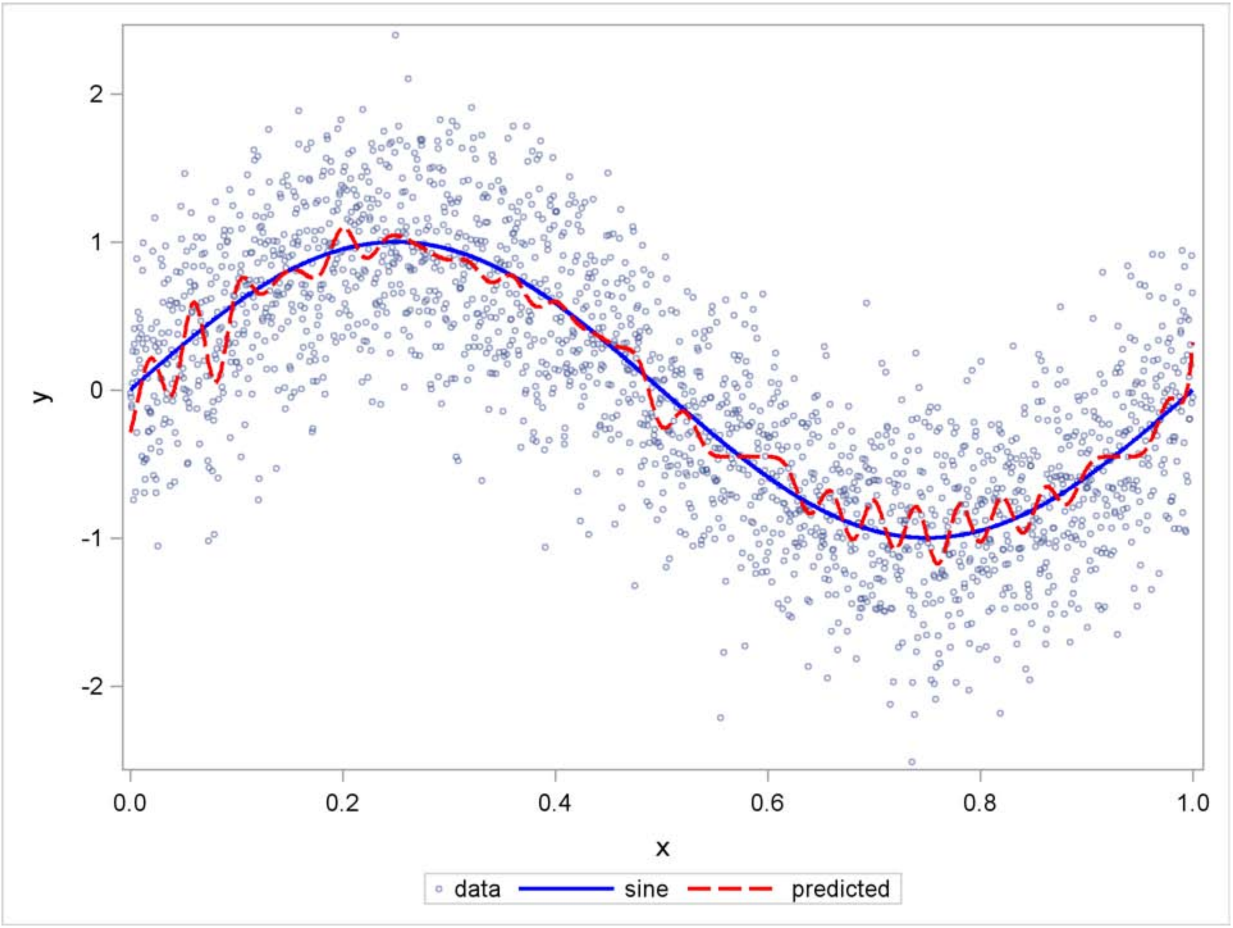
```
proc sgplot data=sineOutCoarse;
   yaxis label='y';
   scatter x=x y=noisySine/transparency=0.5
                           markerAttrs = (size=3)
                           legendLabel = 'data';
   series  x=x y=sine/lineattrs=(color=blue thickness=2);
   series  x=x y=predicted/lineattrs=(color=red thickness=2 pattern=mediumDash);
run;
```

Figure 4 Fit Plot with Four Internal Knots



You can see that selecting basis functions from the B-spline basis enables you to obtain a successful fit. Can you improve the fit by using a much finer set of knots? Figure 5 shows that the answer is no. In this case, 49 equally spaced knots are used. The distance between adjacent knots is $1/50$, and so the support width of each cubic B-spline basis is 0.08 $(4/50)$. Because each basis function is supported on such a small fraction of the range of the data, many of these basis functions are needed in the final fit; this results in the overfitting that you clearly see in Figure 5.

Figure 5 Fit Plot with 49 Internal Knots



Suppose you suspect that your data have some local features that you want to capture in a nonparametric fit based on spline approximations. If you know the locations of these local features, you can accommodate this by using unequally spaced knots with more knots concentrated around the locations of these local features. Cases where such a priori information is not available appear to be problematic. In order to capture local features whose locations are not known, you need to use many knots dispersed over the entire range of the data. But then how do you avoid the overfitting? One approach, known as the penalized B-spline method (Eilers and Marx 1996), adds a penalty to the least squares objective function that increases as the roughness of the fit increases. By appropriately choosing a smoothing parameter that adjusts the size of this roughness penalty, you can balance the goodness of the fit (bias) with the roughness (variance) of the fit. However, because only one parameter controls this bias variance tradeoff, you cannot effectively model data that have local features that occur at different scales. In such cases, effect selection from several sets of B-spline bases that correspond to different scales in the data proves to be an effective strategy.

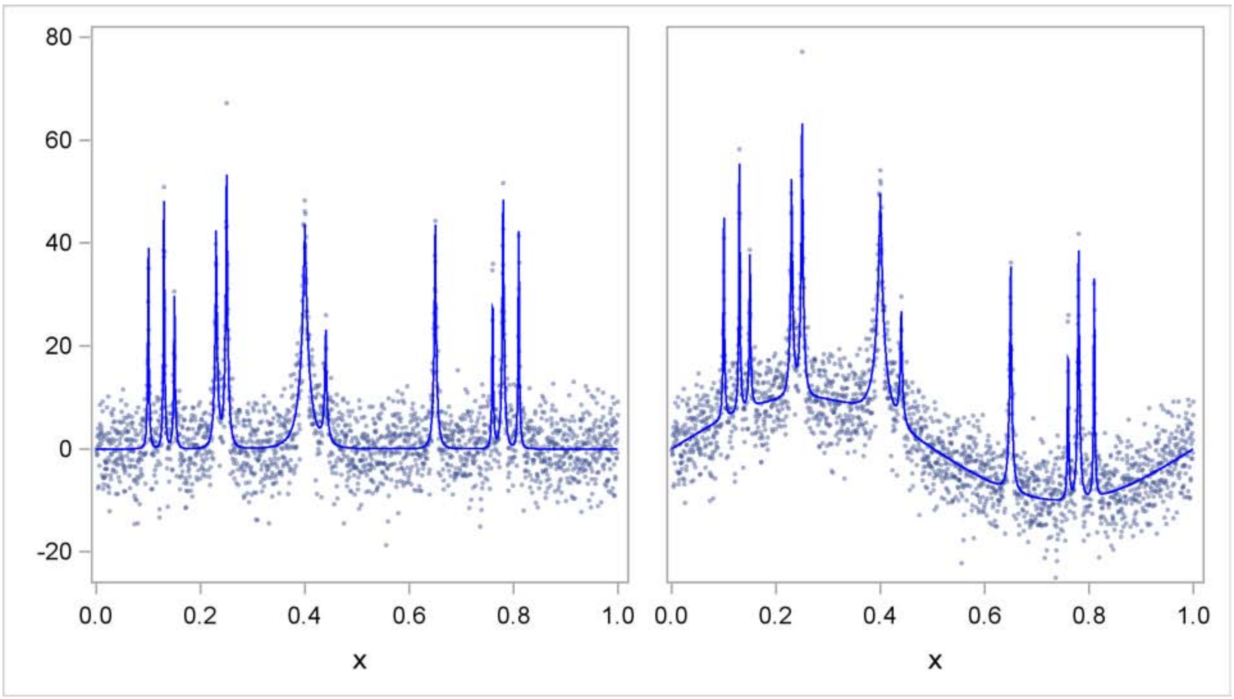
To illustrate this approach, data for this example are adapted from a set of benchmarks developed by Donoho and Johnstone (1994), which have become popular in the statistics literature. The particular benchmark used is the "Bumps" function to which random noise has been added to create the test data. The following DATA step, adapted from Sarle (2001), creates the data:

```
%let random=12345;

data DoJoBumps;
   keep x bumps noisyBumps bumpsSine noisyBumpsSine;

   do n=1 to 2048;
      x=n/2048;
      link compute;
      noisyBumps      = bumps+5*rannor(&random);
      baseSine        = 10*sin(2*constant('pi')*x);
      bumpsSine       = baseSine + bumps;
```

```
         noisyBumpsSine = baseSine + noisyBumps;
         output;
      end;
   stop;

   compute:
      array t(11) _temporary_ (.1 .13 .15 .23 .25 .4 .44 .65 .76 .78 .81);
      array b(11) _temporary_ (   4    5    3    4    5 4.2 2.1 4.3  3.1  5.1  4.2);
      array w(11) _temporary_ (.005 .005 .006 .01 .01 .03 .01 .01 .005 .008 .005);

      bumps=0;
      do i=1 to 11;
         bumps=bumps+b[i]*(1+abs((x-t[i])/w[i]))**-4;
      end;
      bumps=bumps*10.528514619;
   return;
   run;
```

The constants are chosen so that the noise-free bumps data have a standard deviation of 7. The standard deviation of the noise is 5, yielding a signal noisyBumps with a signal-to-noise ratio of 1.4 $(7/5)$. For the purpose of highlighting signals with multiple scales, this DATA step also creates a signal bumpsSine by adding a sine curve with period 1 to the original bumps curve, yielding a modified bumps function with a sinusoidal baseline for the bumps. Figure 6 shows both the original and modified bumps functions superimposed on the data.

Figure 6 Bumps Function and Sinusoidal Baseline Variant



Suppose you want to fit a smooth curve to the data in the panel on the right in Figure 6. You have seen that in order to capture the sinusoidal baseline behavior, a B-spline basis with about four internal knots is appropriate. However, such a basis does not have the resolution necessary to capture the bumps. On the other hand, if you use a much finer set of knots to define the B-spline basis, then you can resolve the bumps but at the expense of overfitting the data in the regions between the bumps. A way out of this impasse is to provide B-spline bases at multiple scales and use model selection to obtain a parsimonious subset. The following code prototype shows how you can do this:

```
proc glmselect data=DoJoBumps;
   effect spl1 = spline(x/knotmethod=equal(1) split);
   effect spl2 = spline(x/knotmethod=equal(2) split);
   effect spl4 = spline(x/knotmethod=equal(4) split);
   effect spl8 = spline(x/knotmethod=equal(8) split);

   model  noisyBumpsSine = spl1 spl2 spl4 spl8;
run;
```

This method of specifying spline bases at multiple scales can be cumbersome if you want to use a large number of scales. You can use the KNOTMETHOD=MULTISCALE option in a single EFFECT statement to accomplish this more succinctly, as illustrated in the following code:

```
proc glmselect data=DoJoBumps;
   effect spl = spline(x/knotmethod=multiscale(endscale=8) details split);
   model  noisyBumpsSine = spl;
   output out=bumpsSineOut p=predicted;
run;
```

At scale i, the associated B-spline basis corresponds to 2^i equally spaced internal knots. For each scale, a separate spline effect is generated. The name of the constructed spline effect at scale i is formed by appending _Si to the effect name you specify in the EFFECT statement. The ENDSCALE=8 option requests that the finest scale use cubic B-splines defined on 2^8 equally spaced knots in the interval $[0,1]$. Because the cubic B-splines are nonzero over five adjacent knots, at this finest scale the support of each B-spline basis function is an interval of length about 0.02 $(4/257)$, enabling the bumps in the underlying data to be resolved. The default value is ENDSCALE=7. At this scale you can still capture the bumps, but with less sharp resolution. For these data, using a value of ENDSCALE greater than 8 provides unneeded resolution, making it more likely that basis functions that fit spurious features in the noise are selected.

The DETAILS option in the EFFECT statement requests the display of spline knots and spline basis tables. These tables contain information about knots and basis functions at all scales. The results for scale 4 are shown in Figure 7 and Figure 8.

Figure 7 Spline Knots

```
                         Knots for Spline Effect spl

                                  Scale
      Knot                        Knot
    Number        Scale         Number     Boundary                x

        40            4              1        *              -0.1171
        41            4              2        *              -0.0583
        42            4              3        *               0.0005
        43            4              4                        0.0593
        44            4              5                        0.1181
        45            4              6                        0.1769
        46            4              7                        0.2357
        47            4              8                        0.2945
        48            4              9                        0.3533
        49            4             10                        0.4121
        50            4             11                        0.4708
        51            4             12                        0.5296
        52            4             13                        0.5884
        53            4             14                        0.6472
        54            4             15                        0.7060
        55            4             16                        0.7648
        56            4             17                        0.8236
        57            4             18                        0.8824
        58            4             19                        0.9412
        59            4             20        *               1.0000
        60            4             21        *               1.0588
        61            4             22        *               1.1176
```

Figure 8 Spline Details

```
                  Basis Details for Spline Effect spl
                                Scale                          Support
      Column        Scale       Column    -----Support----     Knots

          32            4            1     -0.1171  0.0593     1-4
          33            4            2     -0.1171  0.1181     1-5
          34            4            3     -0.0583  0.1769     2-6
          35            4            4      0.0005  0.2357     3-7
          36            4            5      0.0593  0.2945     4-8
          37            4            6      0.1181  0.3533     5-9
          38            4            7      0.1769  0.4121     6-10
          39            4            8      0.2357  0.4708     7-11
          40            4            9      0.2945  0.5296     8-12
          41            4           10      0.3533  0.5884     9-13
          42            4           11      0.4121  0.6472     10-14
          43            4           12      0.4708  0.7060     11-15
          44            4           13      0.5296  0.7648     12-16
          45            4           14      0.5884  0.8236     13-17
          46            4           15      0.6472  0.8824     14-18
          47            4           16      0.7060  0.9412     15-19
          48            4           17      0.7648  1.0000     16-20
          49            4           18      0.8236  1.0588     17-21
          50            4           19      0.8824  1.1176     18-22
          51            4           20      0.9412  1.1176     19-22
```

The “Dimensions” table in Figure 9 shows that at each step of the selection process, 548 effects are considered as candidates for entry or removal. Note that although the MODEL statement specifies a single constructed effect spl, the SPLIT suboption specifies that each of the parameters in this constructed effect be treated as an individual effect. You see that selecting individual basis functions from spline bases at multiple scales leads to large scale variable selection problems.

Figure 9 Dimensions

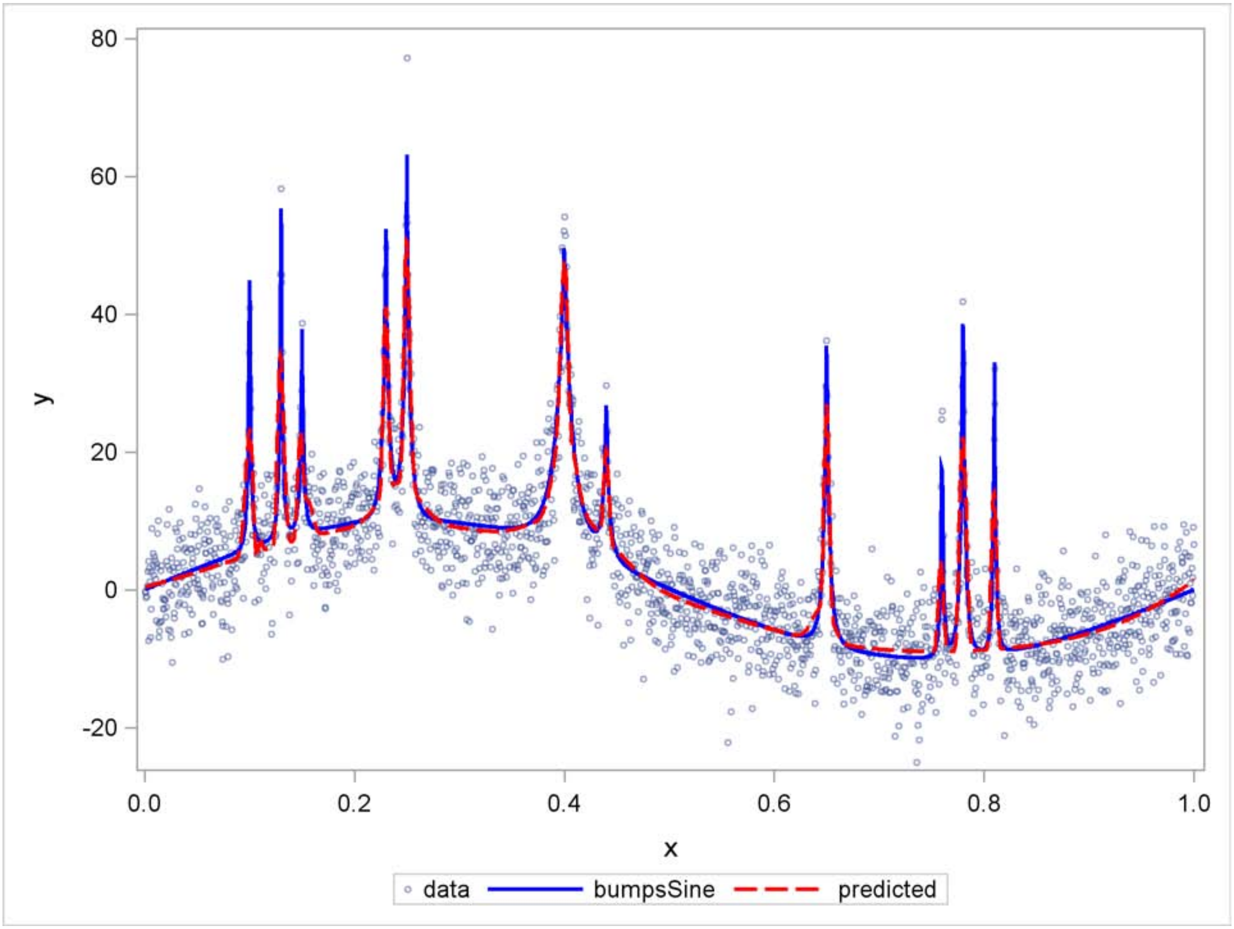
```
                               Dimensions

                   Number of Effects         548
                   Number of Parameters      548
```

Figure 10 shows the parameter estimates for the selected model. The model, selected by the default stepwise method, contains a small number of B-spline basis functions from multiple scales. The fit plot in Figure 11 shows that this model accurately captures both the low-frequency sinusoidal baseline and also the bumps, without overfitting the regions between the bumps.

Figure 10 Parameter Estimates

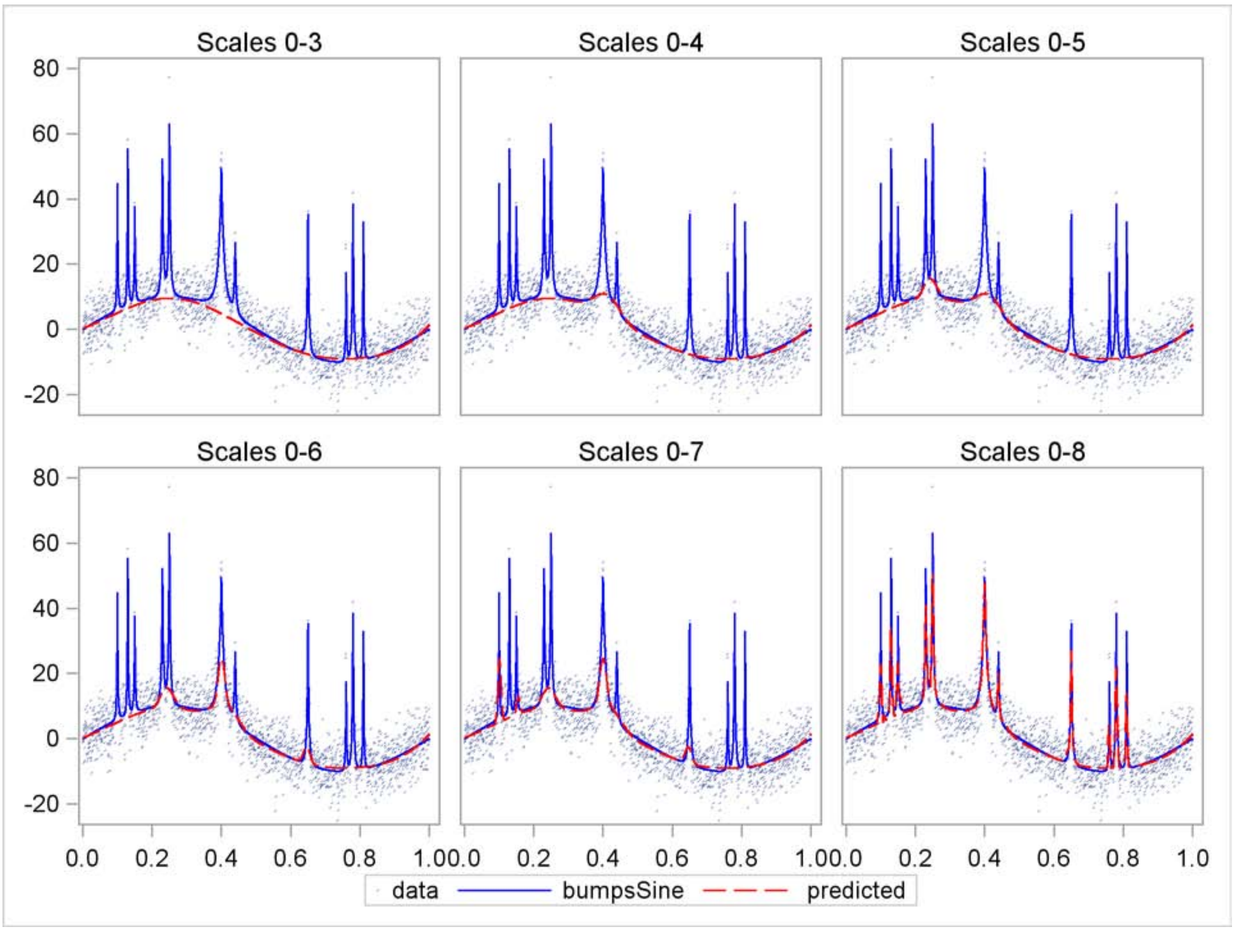
```
                          Parameter Estimates

                                                Standard
Parameter       DF          Estimate               Error    t Value

Intercept        1         -2.111956            0.768772      -2.75
spl_S0:3         1        -18.424959            2.048443      -8.99
spl_S1:3         1         27.895811            1.864830      14.96
spl_S2:3         1          5.625511            1.656240       3.40
spl_S2:8         1         39.553382            5.819788       6.80
spl_S4:9         1          9.779992            1.430574       6.84
spl_S5:10        1          8.964383            1.523382       5.88
spl_S6:28        1         20.378594            2.810779       7.25
spl_S6:44        1          7.466644            1.868956       4.00
spl_S7:15        1         28.976992            2.560599      11.32
spl_S7:22        1          7.895816            2.160341       3.65
spl_S8:29        1        -18.081751            3.541369      -5.11
spl_S8:35        1         36.898716            3.164351      11.66
spl_S8:36        1         14.734931            3.168718       4.65
spl_S8:40        1         21.064344            2.932460       7.18
spl_S8:61        1         40.378057            3.067269      13.16
spl_S8:66        1         49.899718            3.370580      14.80
spl_S8:67        1         12.269691            3.267487       3.76
spl_S8:104       1         17.032223            3.641212       4.68
spl_S8:105       1         29.080374            3.796361       7.66
spl_S8:115       1         19.432295            2.931405       6.63
spl_S8:169       1         44.868001            3.529586      12.71
spl_S8:197       1         19.733357            2.778229       7.10
spl_S8:202       1         36.232431            3.153724      11.49
spl_S8:203       1         27.185588            3.153133       8.62
spl_S8:210       1         34.145250            2.772482      12.32
```

Figure 11 Fit by Selecting B-splines



You can see how the fit is built at multiple scales in the panel in Figure 12. Each plot in the panel shows the partial fit that you obtain if you use the estimated parameters from the selected model but just from the indicated scales. You see that the basis functions at scales 0–3 accurately model the slowly varying sinusoidal baseline. The basis functions at scales 4 through 7 capture the lower portion of the wide bumps. The tops of the bumps are captured by basis functions at scale 8.

Figure 12 Multiresolution Analysis



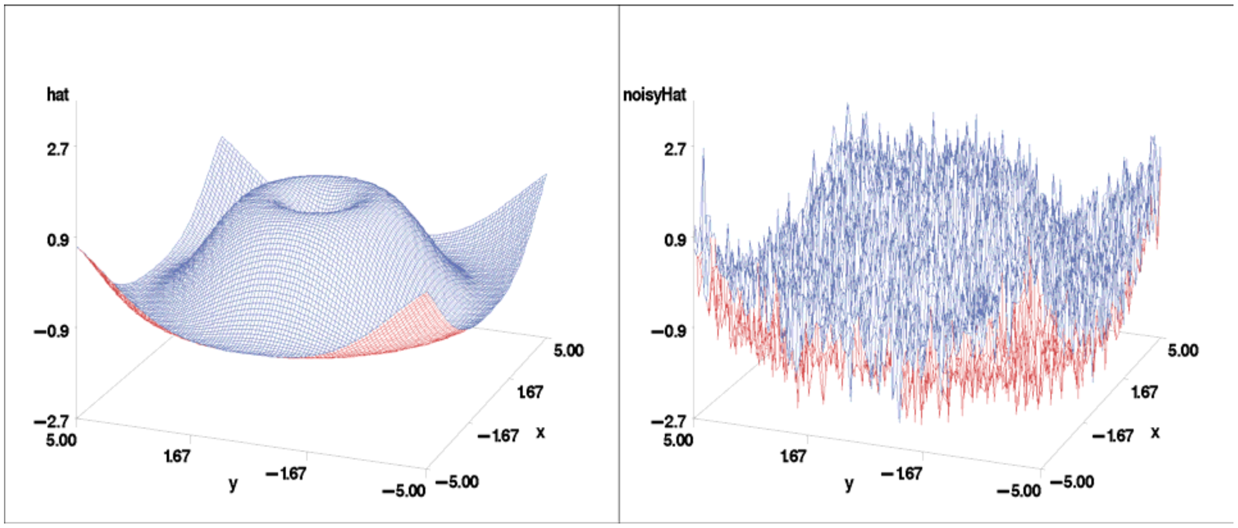
MULTIVARIATE NONPARAMETRIC FITTING

The preceding example shows how you can use variable selection with B-spline bases at multiple scales to fit functions of a single variable. The same ideas extend to multivariate functions by using products of one-dimensional B-spline bases. This section provides a two-dimensional example.

The following code generates the noisy data for the “hat” function that are displayed to the right of the true function in Figure 13:

```
data hat;
   do y=-5 to 5 by 0.1;
      do x=-5 to 5 by 0.1;
         hat      = sin(sqrt(x*x+y*y));
         noisyHat = hat + 0.5*rannor(1);
         output;
      end;
   end;
run;
```

Figure 13 Hat Function and Noisy Data



Suppose you have a basis, B_1, of m univariate functions defined on an interval $[a,b]$ and another basis, B_2, of m univariate functions defined on an interval $[c,d]$. Then you can define a basis, known as a tensor product basis, of bivariate functions defined on the rectangle $[a,b] \times [c,d]$ that consists of all mn pairwise products of one function from each basis. In particular, you can do this with the B-spline bases used in the preceding section, and you can use such a basis to fit a function to a response that depends on two regressors. If you want to fit such a response nonparametrically, you can use all pairwise products bases defined at multiple scales for each predictor together with variable selection to obtain a parsimonious subset. The following code show how you can do this to smooth the noisy hat data:

```
proc glmselect data=hat;
   effect spl_x = spline(x/knotmethod=multiscale(endscale=3) split);
   effect spl_y = spline(y/knotmethod=multiscale(endscale=3) split);
   model noisyHat = spl_x | spl_y;
   output out=outHat1 p=predHat;
run;
```

You use an effect statement that defines B-spline bases at multiple scales for each regressor. The ENDSCALE=3 option specifies that at the finest scale eight equally spaced internal knots are used. This provides sufficient resolution to model the hat function. The PROC GLM-type effect specification spl_x | spl_y is a concise way of specifying a model with both main effects spl_x and spl_y and their interaction spl_x * spl_y.

Since the split option is used on both EFFECT statements, each column in the design matrix can enter or leave the model individually. Furthermore, because the number of columns in the interaction is the product of the number of basis functions for each regressor, fitting nonparametric models by selecting from interactions of spline bases at multiple scales leads to large model selection problems. The dimension table in Figure 14 shows that this model yields a variable selection problem with 1,024 regressors.

Figure 14 Dimensions

```
                The GLMSELECT Procedure

                      Dimensions

          Number of Effects        1024
          Number of Parameters     1024
```

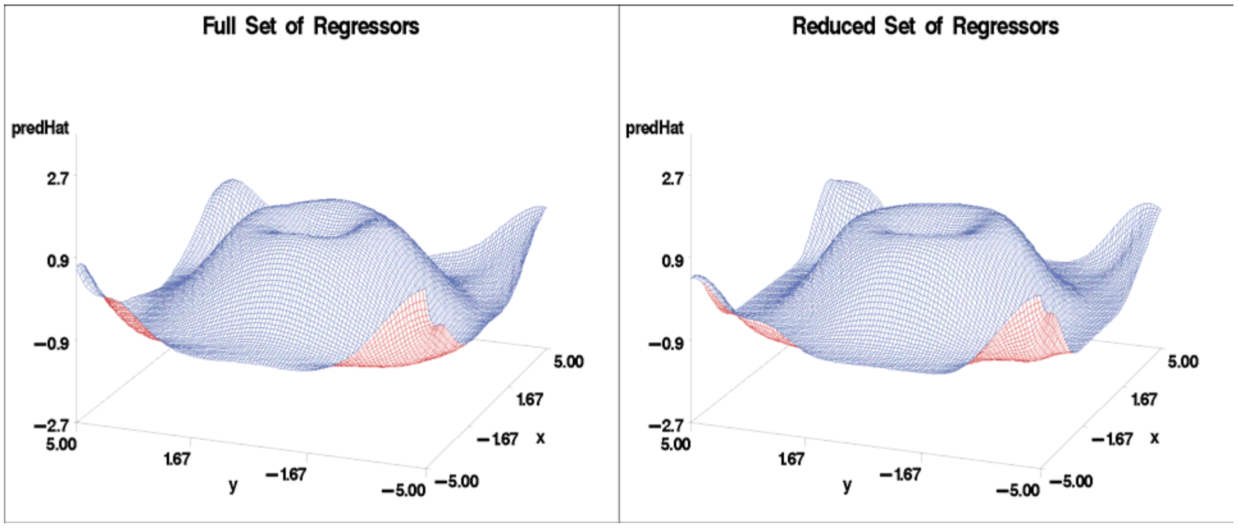
The fit is displayed in the left panel of Figure 15. You see that the default stepwise selection successfully smooths the noisy data and recovers a smooth approximation of the underlying true hat function.

You can extend this approach to more than two dimensions. However, such a fully nonparametric approach in higher dimensions generates a huge variable selection problem. Usually, you can use prior knowledge to make an informed choice of a more appropriate set of basis functions than simply taking tensor products of univariate B-spline bases at multiple scales. For example, you might know that local features in your data are radially symmetric. In this case, using only radially symmetric basis functions at the finer scales that capture local features is appropriate. In the context of tensor product spline bases, you obtain locally radially symmetric basis functions by using only interactions of splines that are defined with the same equal spacing of knots for each regressor. Another popular method is to use radial basis functions instead of tensor products of splines. You can apply the preceding reasoning in fitting the data in this example. The following code shows how you do this:

```
proc glmselect data=hat;
   effect spl_x  = spline(x/knotmethod=multiscale(endscale=2) split);
   effect spl_x8 = spline(x/knotmethod=equal(8) split);

   effect spl_y  = spline(x/knotmethod=multiscale(endscale=2) split);
   effect spl_y8 = spline(y/knotmethod=equal(8) split);

   model noisyHat = spl_x | spl_y  spl_x8 * spl_y8;

   output out=outHat2 p=predHat;
run;
```

Note that basis functions at the finest scale are defined separately as constructed effects spl_x8 and spl_y8. These two constructed effects interact with each other but not with basis functions at the coarser scales that are defined by constructed effects spl_x and spl_y. This produces at total of 544 regressors from which 47 are selected to yield the fit shown the right panel in Figure 15. Note that the second GLMSELECT step chooses from about half the number of regressors as the original GLMSELECT step in this example.

Figure 15 Fits by Selecting B-Splines



You see that both fitted surfaces in Figure 15 are successful in capturing the underlying hat function from the noisy data.

MODEL SELECTION FOR MICROARRAY DATA

Microarrays are arrangements of microscopic spots of short segments of genes on glass or silicon chip substrates, with the number of spots ranging from the hundreds to tens of thousands. A probe is formed by attaching fluorescent labels to the DNA of a sample being investigated and letting this material attach to the genetic material in the microarray. The intensity of the fluorescence at each spot in the microarray yields expression levels of how much of specific DNA fragments is present in the sample. A typical experiment consists of using samples from two classes of cells that correspond to the presence or absence of a property under investigation. Statistical challenges are to identify the genes that are predictive of this property and to build a reliable predictive model that can be used to classify new samples based on their gene expression profiles.

Usually the data for an experiment consist of a few hundred samples that form the training data used for building the model and an equal or greater number of samples that comprise the test data used for validating the model. Hence, you are confronted with challenging statistical problems that have just hundreds of observations but tens of thousands of regressors.

The second phase of the "MicroArray Quality Control (MAQC) Project," a large collaborative effort between the FDA, industry, and academia, focuses on developing best practices for the development and validation of predictive signatures and classifiers for microarray data that the FDA aims to publish by the end of 2009. You can find details about this project and references to related publications and meetings at the Web site http://www.fda.gov/nctr/science/centers/toxicoinformatics/maqc/. As part of this project, Russ Wolfinger, director of scientific discovery and genomics at SAS, and many other groups have investigated a wide variety of different statistical approaches for building predictive classifiers that use the data sets from the MAQC-II initiative. Wolfinger and his team have shown that variable selection performed with the GLMSELECT procedure can produce predictive classifiers whose performance is competitive with other state-of-the-art techniques (Wolfinger, 2008).

The following example shows how you can use the GLMSELECT procedure to build such classifiers, and it uses bootstrap techniques to extend the analyses carried out by Wolfinger. The data sets used in the MAQC-II study are protected by confidential disclosure agreements, and so instead this paper uses simulated data that is designed to have properties and dimensionality consistent with the data in this study.

This example is organized into the following steps:

1. Create the simulated data.
2. Build a classifier with an optimal logistic model.
3. Build a classifier with stepwise selection.
4. Build a classifier with bootstrap-based model averaging.
5. Build a classifier with stepwise selection and validation.

Create the Simulated Data

The following DATA step creates the simulated training and test data sets, each with 300 observations and 35,000 explanatory variables. The binary response y, coded as 0 or 1, is constructed to depend in decreasing order of importance on just the first 10 regressors, which are collectively referred to as the "germane" regressors. The values of y are chosen to be consistent with the probabilities computed by applying an inverse logit transform to a linear predictor formed from the germane regressors.

```
%let nTest  =   300;
%let nTrain =   300;
%let nVars  = 35000;

data train test;
  array x{&nVars};
  do obsNum=1 to &nTest+&nTrain;

     /*--- randomly assign the regressors ---*/
     do j=1 to &nVars;
       x{j}=ranuni(1);
     end;

     /*--- Form a linear predictor that   ---*/
     /*--- depends on x1-x10              ---*/
     linp =  11*x1 - 10*x2 + 9*x3 - 8*x4 + 7*x5
             -6*x6 + 5*x7  - 4*x8 + 3*x9 - 2*x10;

     /*--- Use an inverse logit transform to assign    ---*/
     /*--- the probability that the response will be 1 ---*/
     TrueProb = 1/(1+exp(-linp));

     /*--- Assign the binary response to be ---*/
     /*--- consistent with the probability  ---*/
     if ranuni(1) < TrueProb then y=1;
                             else y=0;
```

```
         /*--- Output: nTrain observations to train data ---*/
         /*---         nTest  observations to test data  ---*/
         if obsNum<= &nTrain then output train;
                             else output test;
      end;
    run;
```

The following code creates a macro classify that classifies observations in an input data set into classes 0 or 1 based on whether a probability is less than or greater than 0.5, respectively. Then it uses the FREQ procedure to count how many times this predicted classification agrees with the observed binary response.

```
    %macro classify(source,var,probability);
       data &source;
          length &var $9;
          set &source;
          if &probability<= 0.5 then yPred = 0;
                                else yPred = 1;

          if y = yPred then &var = 'Correct';
                       else &var = 'Incorrect';
       run;

       proc freq data=&source;
          table &var / nocum;
       run;
    %mend;
```

Figure 16 shows the results of applying this macro to the train and test data sets as follows:

```
    %classify(train,Train,TrueProb);
    %classify(test,Test,TrueProb);
```

You see that the response corresponds to its most likely value for about 95% of the observations in the training data and 92% of the observations in the test data. These percentages are bounds on the predictive accuracy that a good statistical model for these data should achieve.

Figure 16 Training and Test Set Classification for True Probability

```
                        The FREQ Procedure

               Train        Frequency     Percent
               ------------------------------------
               Correct           284        94.67
               Incorrect          16         5.33

               Test         Frequency     Percent
               ------------------------------------
               Correct           277        92.33
               Incorrect          23         7.67
```

Classification with an Optimal Logistic Model

Generalized linear models are often used to model data with a binary response. The special case of a linear logistic model assumes that the response follows a binomial distribution where the expected value μ_i of the response y_i is related to a linear combination of the explanatory variables by

$$g(\mu_i) = x_i'\beta$$

where g is the logit link function defined by $g(\mu) = \log\left(\frac{\mu}{1-\mu}\right)$. The response in this example is related to a linear combination of the germane regressors via an inverse logit transform, and so a linear logistic model that uses just the germane regressors is an optimal model.

In contrast to logistic models, the GLMSELECT procedure performs model selection under the usual linear model assumptions where the link function is the identity function and the error distribution is Gaussian. Using the GLMSELECT procedure for classification with a binary response violates these assumptions. Nevertheless, as the following analyses show, careful use of linear variable selection can yield models whose predictive performance is competitive with the optimal logistic model. This approach of doing classification by using variable selection in very large linear models has also been used successfully in areas other than microarray analysis. For example, Foster and Stine (2004) use a modified version of linear stepwise selection to build a predictive model for bankruptcy from over 67,000 possible predictors and show that this yields a model whose predictions compare favorably with other recently developed data mining tools.

In order to obtain a reference point for comparing models produced by using PROC GLMSELECT with an optimal model for the simulated data in this example, the following code fits a logistic model that uses the unrealistic prior knowledge of the germane regressors:

```
proc logistic data=train;
   model y(event='1') = x1-x10;
   output          out=trainOut(keep=y BestEst) p=BestEst;
   score data=test out=testOut (rename=(P_1=BestEst) keep=y P_1);
run;
%classify(trainOut,Train,BestEst);
%classify(testOut,Test,BestEst);
```

The classification results in Figure 17 show that this optimal logistic model correctly classifies 95% of the training data and 90.33% of the test data, consistent with the expected accuracy bounds for this simulated data.

Figure 17 Training and Test Set Classification Errors for Optimal Logistic Model

```
                  The FREQ Procedure

       Train         Frequency     Percent
       ------------------------------------
       Correct             285       95.00
       Incorrect            15        5.00

       Test          Frequency     Percent
       ------------------------------------
       Correct             271       90.33
       Incorrect            29        9.67
```

For real microarray data, prior knowledge of what regressors to use is not available. In such cases, a simple approach that you might consider is to use a logistic model with the subset of the regressors that have the highest pairwise correlations with the response. There are several problems associated with this simple approach:

1. Regressors that are important in a multivariate analysis might have small pairwise correlations with the response.
2. Even after ranking regressors by their correlation with the response, you still need to determine how many of these regressors to use.
3. With the small number of observations and the very large number of regressors, this method often selects many regressors that have very little out-of-sample predictive utility but by chance have relatively large correlations with the response in the training data.

Multivariate variable selection methods such as stepwise selection partially address the first problem but are impacted by the multivariate analogues of the latter two problems. You still need to determine an appropriate stopping rule, and you have to deal with multiple comparison issues at each step of the selection process. A further difficulty is that traditional implementations of variable selection techniques for both linear and logistic regression start by assembling the full $\mathbf{X}'\mathbf{X}$ matrix, where $\mathbf{X}$ is the design matrix. With 35,000 regressors, the computational cost of forming and storing this matrix is very large, making such techniques impractical on most current-generation computers.

Classification with Linear Stepwise Selection

The GLMSELECT procedure has features that enable you to mitigate all of the preceding problems. You can use validation and data resampling to address issues of stopping rules and selection bias. For wide data, the GLMSELECT procedure builds the relevant portions of the $\mathbf{X}'\mathbf{X}$ matrix as the selection process proceeds, at the expense of requiring a pass through the data at each step. When the number of regressors that are selected is a small fraction of the total number of regressors (as occurs for microarray data), this approach yields substantial memory and computational savings, making such analyses tractable on even modest hardware.

The following code provides a starting point for the analysis:

```
ods graphics on;
proc glmselect data=train plots=(criteria ase) testdata=test seed=1;
   model y = x1-x&nVars / selection=stepwise(choose=cv maxsteps=30)
                          cvMethod=random(5);
   output          out=trainOut(keep=y predGls)  p=predGls;
   score data=test out=testOut (keep=y predGls)  p=predGls;
run;
ods graphics off;
%classify(trainOut,Train,predGls);
%classify(testOut,Test,predGls);
```

The default stepwise selection method with variable selection and stopping based on the Schwarz Bayesian information criterion (SBC) is used. If the SBC values decrease monotonically at each of the first 30 steps, the MAXSTEPS=30 option stops the stepwise selection process at step 30. The CHOOSE=CV option, in concert with the CVMETHOD=RANDOM(5) option, specifies that among the models at each step, the selected model is the one that yields the lowest predicted residual sum of squares obtained by using fivefold cross validation. This statistic is computed by randomly splitting the data into five approximately equal-sized parts. One of these parts is held out for validation, and the model is fit on the remaining four parts. This fitted model is used to compute the predicted residual sum of squares on the omitted part, and this process is repeated for each of five parts. The sum of the five predicted residual sum of squares so obtained is the CVPRESS statistic.

When you enable ODS Graphics and request the PLOTS=CRITERIA option in the PROC GLMSELECT statement, you obtain the panel of fit criteria shown in Figure 18. You can see that the CVPRESS criterion and all the other fit criteria shown are still decreasing at step 30, which thus becomes the selected step.

Figure 18 Panel of Fit Criteria

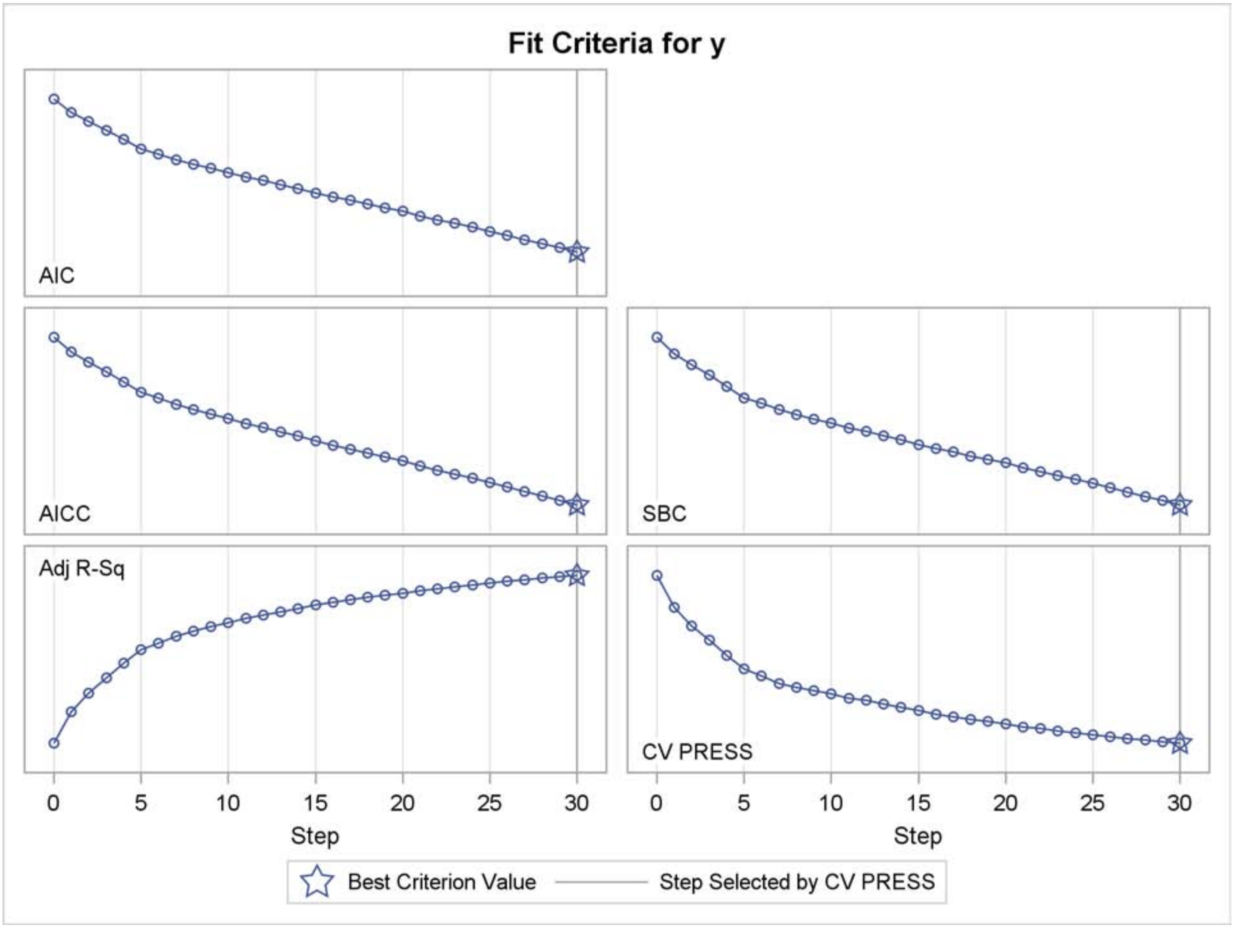


Figure 19 shows the effects in the selected model. You see that although all but two of the germane regressors are selected, most of the regressors selected are not systematically related to the response. This indicates that the selected model overfits the training data and so generalizes poorly to new data.

Figure 19 Selected Effects

```
Effects: Intercept x1 x2 x3 x4 x5 x6 x7 x9 x2291 x4062 x4356 x4745 x4996 x6028
         x6096 x9026 x11002 x11126 x11735 x13702 x14969 x16976 x17543 x18477
         x24775 x25042 x25982 x32233 x32762 x33126
```

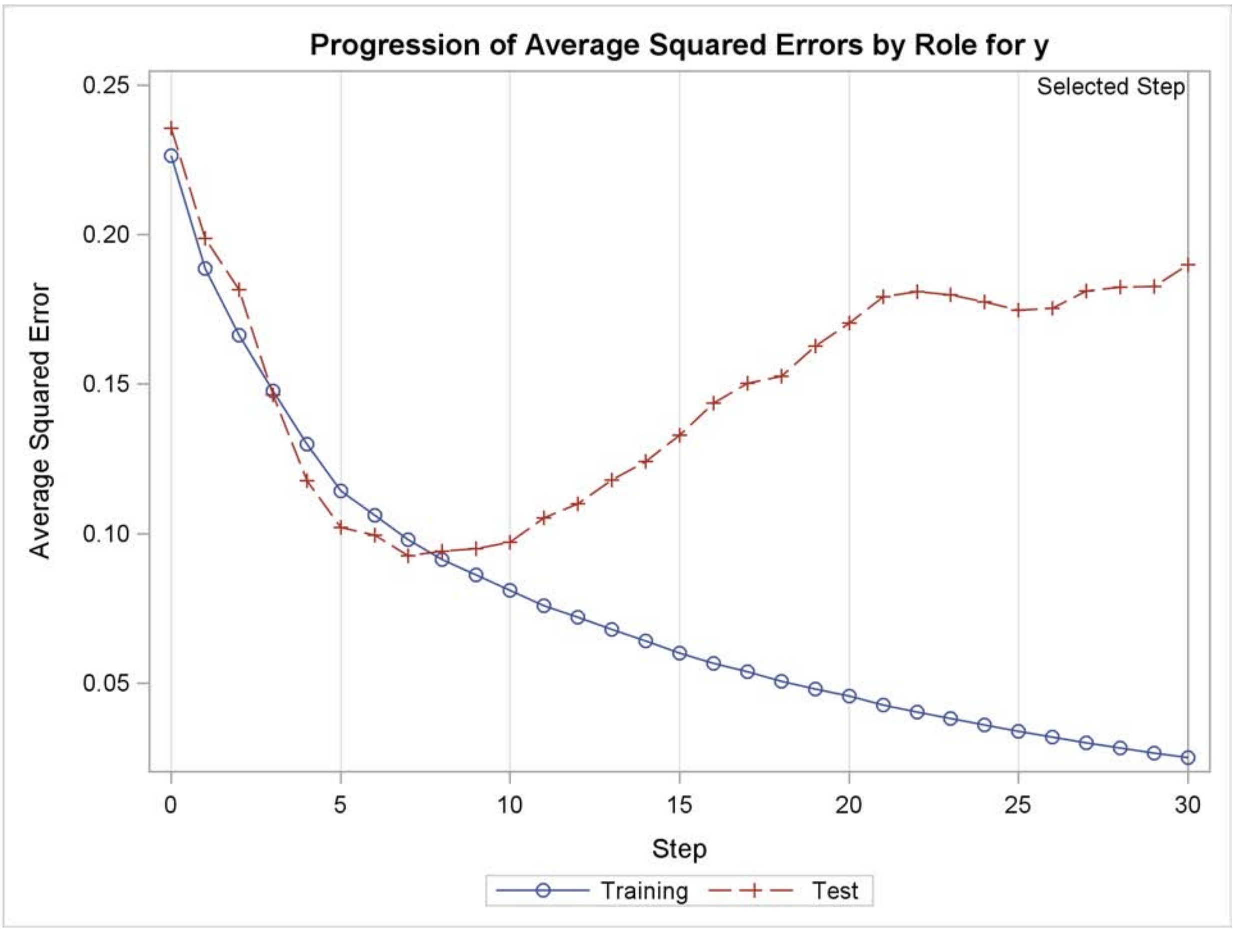
You can see the effects of this overfitting in the classification results in Figure 20; these results show that 100% of the observations in the training data but just 76% of the observations in the test data are correctly classified. Note that the predicted responses produced by PROC GLMSELECT are not probabilities. Classification is done by using 0.5 (the midpoint of the response coding values) as the critical value and assigning observations to classes 0 or 1 depending on which side of the critical value the predicted response falls.

Figure 20 Training and Test Set Classification Errors for GLMSELECT Model

```
                              The FREQ Procedure

                         Train      Frequency     Percent
                         --------------------------------
                         Correct         300      100.00

                       Test         Frequency     Percent
                       ----------------------------------
                       Correct            228       76.00
                       Incorrect           72       24.00
```

You can also clearly see the impact of the overfitting in the average squared error plot in Figure 21. This plot shows how the average squared error on the training and test data changes at the steps of the selection process. You see that although this error decreases monotonically on the training data, it starts growing after step 7 on the test data. Note that the test data is being used to evaluate how the models at each step of the selection process generalize to new data. You cannot use this data as part of the selection process. In studies such as the MAQC-II project, researchers are provided with only the training data, and the models they produce are evaluated by their performance on the unseen test data.

Figure 21 Average Squared Error Progression



Classification with Bootstrap-Based Model Averaging

Suppose you repeated the preceding model selection for multiple sets of training data that represent the same underlying biology. Each set of training data would yield different sets of selected regressors that each overfit their respective training data. However, the germane regressors would appear frequently in the selected models, but few of the non-predictive regressors would appear multiple times. Hence the average prediction from all these models would give greater weight to the germane regressors and thus would produce more accurate classification on new data. For real microarray experiments you do not have multiple sets of training data, but you can simulate this situation by repeatedly sampling your training data. In the bootstrap method, you generate a sample with the same number of observations as the training data by randomly drawing with replacement from the training data. Bootstrap predictions are obtained by fitting models to these bootstrap samples and averaging the predictions that you obtain.

The following code shows how you can use the SURVEYSELECT procedure to produce the bootstrap samples:

```
%let nSamples = 100;

proc surveyselect data=train out=trainBoot
    outall
    reps     = &nSamples
    sampsize = &nTrain
    method   = urs
    seed     = 1;
run;
```

The output data set trainBoot contains all the bootstrap samples of the input data, indexed by a variable named Replicate. The OUTALL option specifies that all the observations in the input data set appear in each sample. A variable named NumberHits holds the number of times that each observation is selected within a sample. The METHOD=URS option specifies that the sampling is done with replacement.

By using Replicate as a BY variable and NumberHits as a FREQ variable, the following code selects models for all the bootstrap samples in a single invocation of the GLMSELECT procedure. The macro bootstrapAverage forms the average predictions across the BY groups and uses these average predictions for classification.

```
proc glmselect data=trainBoot;
   freq NumberHits;
   by   Replicate;

   model y=x1-x&nVars/maxsteps=20;
   output          out=trainOut(keep=y obsNum predGls)  p=predGls;
   score data=test out=testOut (keep=y obsNum predGls)  p=predGls;
   ods output parameterEstimates = pEst;
run;

%macro bootstrapAverage(source);
   proc sort data=&source.Out;
     by  obsNum;
   run;

   proc means data=&source.Out noprint;
      by obsNum;
      var predGls;
      output out=predOut(keep=pMean) mean=pMean;
   run;

   data results;
      length  &source $9;
      set &source;
      set predOut;
      if pMean<= 0.5 then yPred = 0;
                     else yPred = 1;
      if y = yPred then &source = 'Correct';
                   else &source = 'Incorrect';
   run;

   proc freq data=results;
       table &source / nocum;
   run;
%mend;

%bootstrapAverage(Train);
%bootstrapAverage(Test);
```

Figure 22 shows the classification accuracy you achieve using the average predictions from the bootstrap models. You see that although the average of the bootstrap predictions still overfits the training data, the average predictions yield classification accuracy on the test data that rivals the accuracy obtained with the optimal logistic model, which uses a priori knowledge of the germane regressors.

Figure 22 Training and Test Set Classification Errors for Bootstrap GLMSELECT Model

```
                    The FREQ Procedure

             Train      Frequency     Percent
             --------------------------------
             Correct         300      100.00

            Test         Frequency      Percent
            ----------------------------------
            Correct            267        89.00
            Incorrect           33        11.00
```

You can also form an average model whose parameter values are the averages of the parameter values in the N bootstrap models. If a parameter is not selected for a given sample, then it assigned the value 0 for that sample and it is included when forming the average parameter estimates. With this definition of the average model, the predictions you obtain by using the average parameter estimates to score a data set are the same as the predictions you obtain by scoring this data set by using each of the bootstrap models and then averaging the predictions. To show this, denote by $\beta^{(i)}$ the parameter estimates for the bootstrap sample i where $\beta_j^{(i)} = 0$ if parameter j is not the selected model for sample i. Then the predicted values $\hat{y}^{(i)}$ for bootstrap model i are given by

$$\hat{y}^{(i)} = \mathbf{X}\beta^{(i)}$$

where **X** is the design matrix of the data to be scored. Forming averages gives

$$\hat{y}^{\text{ave}} = \frac{\sum_{i=1}^{N} \hat{y}^{(i)}}{N} = \frac{\sum_{i=1}^{N} \mathbf{X}\beta^{(i)}}{N} = \mathbf{X}\frac{\sum_{i=1}^{N} \beta^{(i)}}{N} = \mathbf{X}\beta^{\text{ave}}$$

where for parameter j, $\beta_j^{\text{ave}} = \frac{1}{N}\sum_{i=1}^{N} \beta_j^{(i)}$

You can see that if a parameter estimate is nonzero in just a few of the bootstrap models, then averaging the estimates for this parameter shrinks this estimate towards zero. It is this shrinkage that ameliorates the bias that a parameter is more likely to be selected if it is above its expected value rather than below it. This reduction in bias results in the improved predictions on new data that you obtain with the average model.

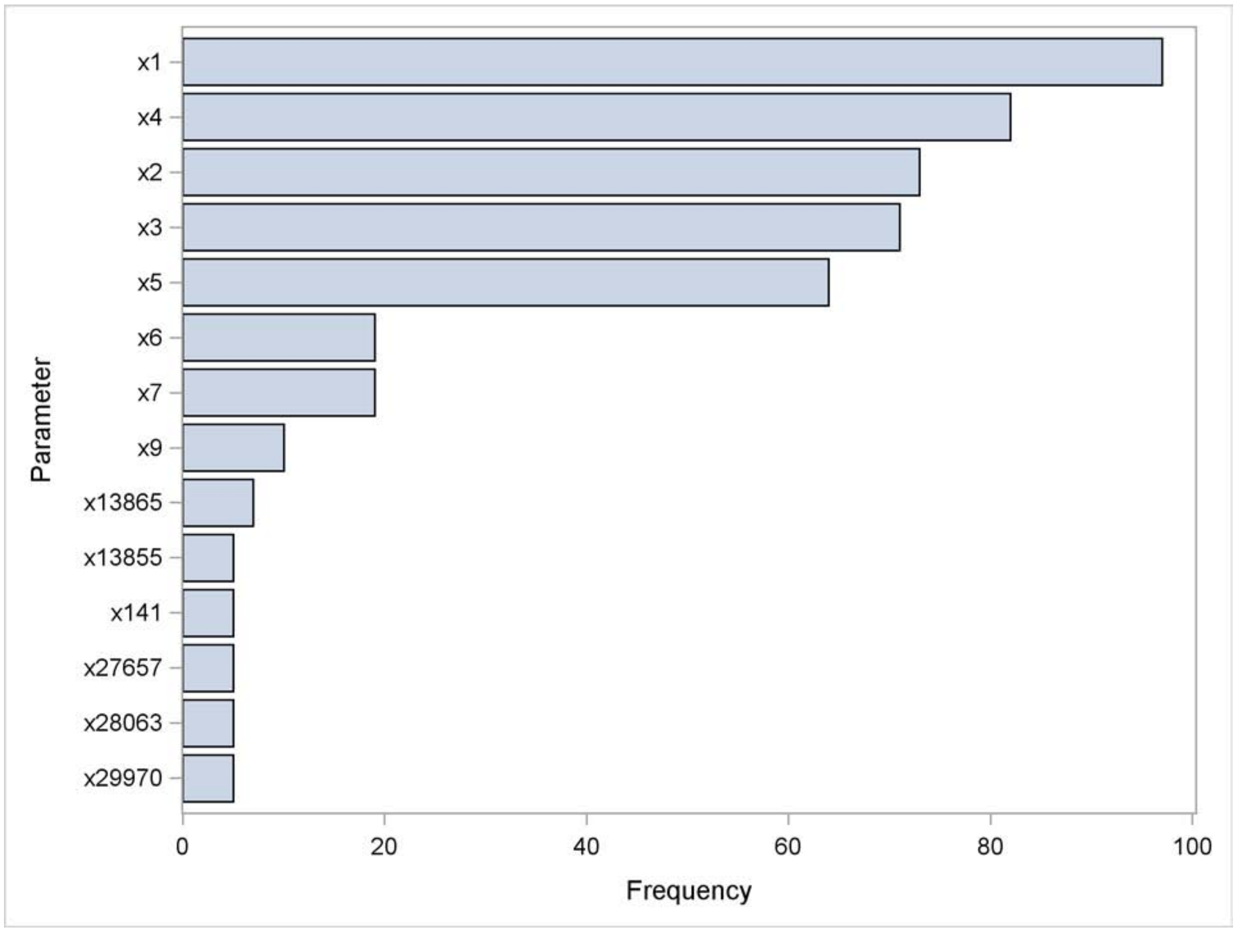
However, the average model is not parsimonious in that it can have nonzero estimates for all parameters that are selected in any bootstrap sample. In the example of this section, 1,344 of the 35,000 regressors are nonzero in the average model. However, 1,165 of these 1,344 regressors are selected exactly once and just 14 regressors are selected in at least 5% of the bootstrap models. One way to try to achieve a more parsimonious model is to average over the subset of the bootstrap models that are selected multiple times. This strategy fails completely in this example because no two bootstrap models contain identical sets of regressors. Another approach to obtaining a more parsimonious model is to ignore parameters that are selected in less than some prespecified percentage of the bootstrap models. This strategy is not reliable because although a parameter might appear in just a small percentage of models, it might be an essential parameter in those models. A simple scenario where this can occur is when two regressors are highly correlated and any appropriate model must contain one of these regressors. If just a few bootstrap models contain one of these two correlated regressors, then ignoring this parameter when forming the average model is inappropriate.

You can get a parsimonious model by using the number of times that regressors are selected as an indication of importance and then refit a new model that uses just the regressors you deem to be most important. This approach is not without its own risks. One possible problem is that you might have several regressors that for purposes of prediction can be used as surrogates for one another. In this case it possible that none of these regressors individually appears in a large enough percentage of the bootstrap models to be deemed important, even though every model contains at least one of these regressors. Despite such potential problems, this strategy is often successful.

The following code uses the output data set that contains the parameter estimates from the bootstrap models to identify the most frequently selected regressors, which are displayed in the bar chart in Figure 23:

```
proc sort data=pEst;by parameter;run;

proc freq data=pEst order=freq noprint;
  tables parameter/nocum out=freqOut(drop=percent);
run;

data freqOut;
  set freqOut;
  rename count=nTimesSelected;
  PercentSelected=100*count/&nSamples;
run;

proc sgplot data=freqout(where=(nTimesSelected>=5 and
                                Parameter ^= "Intercept"));
  yaxis discreteOrder = data;
  hbar Parameter/freq=nTimesSelected;
run;
```

Figure 23 Regressors Selected At Least Five Times



You can now use this reduced set of regressors for further exploratory data analysis that would be intractable using all the original regressors. For example, you might try to see whether interactions among these regressors are important, and you can investigate whether nonlinear transformations of these regressors might be appropriate. As with the original variable selection, you need to be extremely careful that by performing these additional steps you are not simply overfitting the training data.

Classification by Using Stepwise Selection with Validation

Another way to get a parsimonious model directly with PROC GLMSELECT is to mimic the use of a test data set to stop the model selection before overfitting the training data starts to occur. Although you cannot use the test data during the selection process, provided that you have a sufficient number of training observations, you can split the input data into smaller training and validation subsets and use the validation data as a surrogate for the test data. You can use a PARTITION statement to do this as follows:

```
ods graphics on;

proc glmselect data=train plots=ase testdata=test seed=1;
   partition fraction(validate=0.3333);

   model y = x1-x&nVars / selection=stepwise(choose=validate maxsteps=30);
   output           out=trainOut(keep=y predGls)  p=predGls;
   score data=test out=testOut (keep=y predGls)  p=predGls;
run;

ods graphics off;

%classify(trainOut,Train,predGls);
%classify(testOut,Test,predGls);
```

The PARTITION statement specifies that about 100 (1/3 of 300) randomly selected observations from the input data set are reserved as validation data and that the remaining 200 observations are used as the training data. The CHOOSE=VALIDATE option specifies that the selected model is the model at the step that yields the lowest average error sum of squares on the validation data.

Figure 24 shows the selected effects and Figure 25 shows classification errors on the training and test data sets. You can see that this model underfits the training data but still yields good predictions on the test data.

Figure 24 Selected Effects

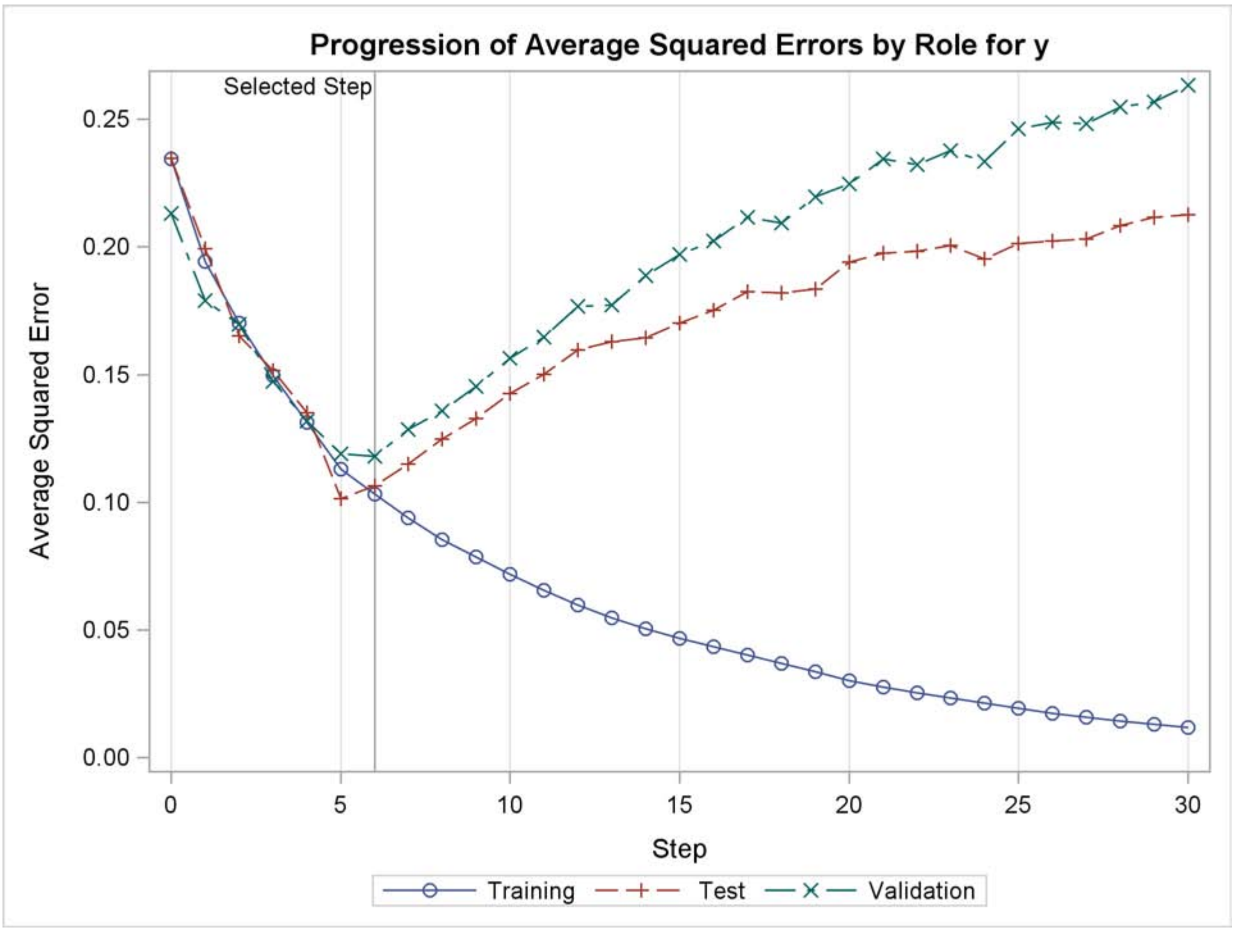
```
                The GLMSELECT Procedure
                    Selected Model

         Effects: Intercept x1 x2 x3 x4 x5 x9
```

Figure 25 Training and Test Set Classification Errors for GLMSELECT Model with Validation

```
                 The FREQ Procedure

      Train        Frequency      Percent
      ----------------------------------
      Correct            265        88.33
      Incorrect           35        11.67

      Test         Frequency      Percent
      ----------------------------------
      Correct            270        90.00
      Incorrect           30        10.00
```

Figure 26 compares the progression of the average squared errors on the training, validation, and test data. You see that the average squared error on the validation data behaves similarly to the average squared error on the test data and prevents the overfitting of the training data.

Figure 26 Average Squared Error Progression



You might also consider using a bootstrap together with a randomly chosen validation data set with each bootstrap sample. However, as the preceding results have shown, you can expect most such bootstrap models to underfit the training data and omit some important regressors. Because the average predictions of the bootstrap models correspond to an average model where the estimates of infrequently selected parameters are shrunk, the bootstrap predictions will amplify the effect of underfitting the training data. One approach to address this is to form an average model where nonselected parameters are not counted when forming the average estimates. See Burnham and Anderson (2002) for details about this strategy.

FUTURE WORK

You have seen how applying bootstrap analysis can be used to address the problem of model selection bias that arises when a single model is produced by using variable selection. In order to facilitate such bootstrap analyses, future releases of the GLMSELECT procedure will include a BOOTSTRAP statement that will enable you to perform bootstrap analysis and obtain appropriately averaged models as a built-in feature. For larger problems, bootstrap analyses are computationally demanding in that the same model selection process needs to be repeated hundreds or even thousands of times. Because the model selection on each bootstrap sample is done independently, multiple samples can be processed in parallel. To exploit this high degree of parallelism, PROC GLMSELECT will support distributed grid processing of the bootstrap analyses.

CONCLUSIONS

This paper demonstrates how you can use the GLMSELECT procedure to perform model selection where the number of regressors is large. Microarray experiments provide one important class of such problems. Other examples include nonparametric fitting by selecting from a large set of suitable basis functions. The paper also highlights the use of the new experimental EFFECT statement to generate spline basis functions, and it demonstrates how you can perform bootstrap analyses to produce models that address problems that arise with single model selection.

REFERENCES

Burnham, K. P. and Anderson, D. R. (2002), *Model Selection and Multimodel Inference,* Second Edition, New York: Springer-Verlag.

Cohen, R, (2006), “Introducing the GLMSELECT Procedure for Model Selection,” *Proceedings of the Thirty-First Annual SAS Users Group International Conference.*

Donoho, D. L. and Johnstone, I. M. (1994), “Ideal Spatial Adaptation via Wavelet Shrinkage,” *Biometrika*, 81, 425–455.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004), “Least Angle Regression (with Discussion),” *Annals of Statistics*, 32, 407–499.

Eilers, P. H. C. and Marx, B. D. (1996), “Flexible Smoothing with B-Splines and Penalties (with Discussion),” *Statistical Science*, 11, 89–121.

Foster, D. P. and Stine, R. A. (2004), “Variable Selection in Data Mining: Building a Predictive Model for Bankruptcy,” *Journal of the American Statistical Association*, 99, 303–313.

Harrell, F. E. (2001), *Regression Modeling Strategies,* New York: Springer-Verlag.

Hastie, T., Tibshirani, R., and Friedman, J. (2001), *The Elements of Statistical Learning,* New York: Springer-Verlag.

Miller, A. (2002), *Subset Selection in Regression,* Second Edition, New York: Chapman & Hall/CRC.

Raftery, A. E., Madigan, D., and Hoeting, J. A. (1997), “Bayesian Model Averaging for Linear Regression Models,” *Journal of the American Statistical Association*, 92, 179–191.

Sarle, W. S. (2001) “Donoho-Johnstone Benchmarks: Neural Net Results,”
ftp://ftp.sas.com/pub/neural/dojo/dojo.html: last accessed January 6, 2009.

Tibshirani, R. (1996), “Regression Shrinkage and Selection via the Lasso,” *Journal of the Royal Statistical Society Series B*, 58, 267–288.

Wolfinger, R. (2008), *Private Communication*.

CONTACT INFORMATION

Robert Cohen
SAS Institute Inc.
500 SAS Campus Drive
Cary, NC 27513
E-mail: Robert.Cohen@sas.com
Web: `http://www.sas.com`