Modified: May 27, 2008

Generalized Additive Modeling (GAM) Using SCAB34S SPLINES and SCA WorkBench

Houston H. Stokes
Department of Economics
University of Illinois at Chicago

William J. Lattyak
Scientific Computing Associates Corp.

Table of Contents

Generalized Additive Modeling (GAM) Using SCAB34S SPLINES and SCA WorkBench

In this document, we discuss *Generalized Additive Models* (GAM) and estimation provided by the B34S® ProSeries Econometric System and SCAB34S SPLINES software products. We also discuss the SCA WorkBench companion product and its user interface to shell a GAM modeling and validation environment in the B34S program suite.

SCAB34S SPLINES provides a subset of the capabilities in the B34S® ProSeries Econometric System and we refer to these products interchangeably within this document. SCAB34S SPLINES runs conveniently as an integrated component to SCA WorkBench. The WorkBench product is a companion to the SCA Statistical System and SCAB34S software, providing a graphical user interface for GAM models with various link functions and error distribution settings. Within the context of GAM model validation, the predictive performance of these models may be validated by comparing the in-sample and out-of-sample predictive values to linear regression models using OLS, MINIMAX, or L1 estimation methods. Within the context of Logistic GAM model validation, the classification performance of these models may be validated by viewing the Confusion Matrices and Lift-Gains between the GAM model and a linear regression, probit, or logistic model.

The SCAB34S SPLINES product provides a number of procedures to perform common data manipulation tasks, organizational tasks, and statistical/econometric analysis tasks. It also contains a comprehensive matrix programming language that may be used to customize procedures for specialized use. No attempt will be made to cover all features of the SCAB34S product in this document nor the full range of applications that may be solved using the B34S matrix programming facilities.[1] Instead, we shall exclusively use the graphical user interface of SCA WorkBench to specify, estimate, and diagnostically test GAM models in SCAB34S SPLINES. SCA WorkBench automatically specifies the command script executed in the SCAB34S SPLINES product based on menu selections. A command file is then executed in the SCAB34S engine and the results are read back into WorkBench for examination. The user may save the program file and modify the command script to address additional analysis requirements that may arise.

A major assumption of any linear process is that the coefficients are stable across all levels of the explanatory variables and, in the case of a time series model, across all time periods. The GAM model is a very useful method of analysis when it is suspected that certain predictor variables may be nonlinear with respect to the dependent variable. There are many theoretical reasons consistent with this occurring in many different applications including energy, finance, economics, medical, social science, and manufacturing.

GAM models, at the very least, can be used as a diagnostic tool in determining potential nonlinear relationships of predictor variables with respect to the dependent variable. Here, the

[1] The text, *Specifying and Diagnostically Testing Econometric Models*, by Houston H. Stokes Greenwood Press (1997) documents the basic B34S capability. A comprehensive document covering the B34S matrix command facilities is under preparation.

user can investigate the curvature of the variable relationship that can later be used in parametric models by adding cubic terms, quadratic terms, *et cetera*, to capture the functional form of the variable. Since GAM models are not limited by imposed functional form, the data itself suggests the functional form of the predictors in the final model. GAMFIT uses nonparametric fitting based on a scatter plot smoother to fit a smooth relationship between two or more variables. The smoother summarizes the trend of the response variable as a function of the predictor variables by iteratively smoothing partial residuals in a process known as back-fitting. Degrees of freedom are approximated as penalties to keep the complexity of the nonparametric curve fitted to the data in check. By examining the curvature plots of the transformations employed by GAM on the predictor variables, the functional form of the predictor variable entering the model can be evaluated and interpreted.

Nonparametric models can have problems related to dimensionality when data is sparse and inflates the variance of the estimates. This is associated with the use of a large number of predictor variables in the model and is often cited as the "*curse of dimensionality*". Nonparametric regression methods that use kernel estimation or smoothing splines may also be difficult to interpret. Stone (1985) originally proposed additive models to help overcome these shortcomings. Hastie and Tibshirani (1990) later proposed generalized additive models where the mean of the dependent variable depends upon the additive predictor through a nonlinear link function. A GAM model replaces the coefficients that would otherwise be found in a linear regression model by a linear smoother. The smoothing technique is based on local averaging of the values of the dependent variable, grouping values of a predictor variable that are near a target value.

SCAB34S SPLINES integrates the General Public License (GPL) code from Trevor Hastie and Robert Tibshirani for Generalized Additive Models (GAM) estimation. After an overview of GAM models, some examples will be presented to illustrate the use of these procedures.

1. GAM MODELING USING SCAB34S SPLINES AND WORKBENCH

Assume a nonlinear model of the form

$$y = f(x_1, \cdots, x_m) + e \tag{1}$$

where x_i and y are one dimensional vectors, a GAM model (see Hastie-Tibshirani (1986, 1990), Faraway (2006, 240)) can be written as

$$E(y \mid x_1, x_2, ..., x_k) = \alpha_0 + \sum_{j=1}^{k} a_j(x_j) + e \tag{2}$$

where $\alpha_j(.)$ are smoothing functions standardized (to remove free constants) so that $E\alpha_j(x_j)=0$. The smoothing functions are estimated one at a time using a forward stepwise estimation method. When (2) is estimated with OLS, the expected coefficients are all 1.0.

The user sets the degree of the smoothing (DF) for each predictor variable. For example, setting DF=3 imposes a cubic fit, DF=2 imposes a quadratic fit, and DF=1 imposes a linear fit. The SCAB34S GAM model summary also provides a significance test (LIN_RES) that measures the difference of the sum of squares of the residuals for the linear restriction case and the transformed case of the GAM model for each predictor variable. An overall diagnostic test $\sigma_2^2 = e'e/(n-p)$ where p = the number of parameters in the model is also provided.

The first step in GAM estimation is to remove the means from all right hand side data and add the spline to form the smoothed series that have 0.0 expectation such that

$$x_i^* = (x_i - \overline{x}_i) + s_i \tag{3}$$

If there are n observations, s_i is an n element spline series. When OLS is applied to the model $y = f(x_1^*,\cdots,x_k^*)$, the expected value of the coefficients are 1.0 as noted above. This allows the GAM coefficients β^{gam} to be estimated using OLS in terms of the original right hand side variables such that $y^* = f(x_1,\cdots,x_k)$ where

$$y^* = y - \sum_{i=1}^{k} s_i \tag{4}$$

$$y^* = \beta_0^{gam} + \sum_{i=1}^{k} \beta_i^{gam} x_i + e_i \tag{5}$$

In effect, the nonlinear effect is removed from the *y* series to obtain the model.

After estimation the predicted left hand side values can be recovered as

$$\hat{y} = \hat{y}^* + \sum_{i=1}^{k} s_i \tag{6}$$

If out-of-sample forecasts are desired, one way to proceed is to use a polynomial regression to approximate the spline functions of the GAM model. Given the new *x* data x^{new} a new estimated spline vector s_i^* can be obtained. Using the estimated GAM coefficients β^{gam} , the forecasts y^e can be calculated as

$$y^e = x^{new}\beta^{gam} + \sum_{i=1}^{k} s_i^* \tag{7}$$

For more information on the use of polynomial regression in forecasting GAM models, refer to Stokes (2008, Chapter 14).

The GAMFIT procedure, by default, uses *Identity* as the nonlinear link function. However, other link functions may be specified depending upon the underlying problem. The types of lik functions supported in GAMFIT are *Identity*, *Inverse*, *Logit*, and *Logarithmic* and are defined below.

We begin by defining z such that

$$z = x\beta + \sum_{i=1}^{k} s_j \tag{8}$$

The linking functions are then specified as

Identity $$\hat{y} = z \tag{9}$$

Inverse $$\hat{y} = 1/z \tag{10}$$

Logit $$\hat{y} = \frac{\exp(z)}{1+\exp(z)} \tag{11}$$

Logarithmic $$\hat{y} = \exp(z) \tag{12}$$

In addition, alternative probability error distribution functions may be specified including

Gaussian (default) $$\Pr ob(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)} \tag{13}$$

Binomial $$\Pr ob(X - x) = \binom{n}{x} \alpha^x (1-\alpha)^{n-x} \quad x = 0,1,\cdots,n \tag{14}$$

Poisson $$\hat{y} = z \Pr ob(X = x) = \frac{e^{-x}\lambda^x}{x!} \tag{15}$$

Gamma $$f(x) = \frac{\lambda^P}{\Gamma(P)} e^{-\lambda x} x^{P-1} \quad x \ge 0,\ \lambda > 0,\ P > 0 \tag{16}$$

1.1 Interpreting the GAM Model Output Summaries

Based on the following GAM model,

```
call GAMFIT(Y  X1[predictor,3]
       CT1[factor,1] CT2[factor,1] CT3[factor,1]
       CT4[factor,1] CT5[factor,1] CT6[factor,1]
     :dist 'gauss'  :link 'ident' );
```

a sample of the output for the GAM model summary is presented below. The model information is summarized into nine columns.

```
Generalized Additive Models (GAM) Analysis
Reference: Generalized Additive Models by Hastie and Tibshirani. Chapman (1990)
Model estimated with GPL code obtained from CRAN.

Gaussian additive model assumed
Identity link - yhat = x*b + sum(splines)

Model  df        coef        st err     z score      nl pval      lin_res       Name     Lag
------------     ----        ------     -------      -------      -------       ----     ---
        1.      118476.       1440.       82.28                                intcpt
        3.00    607.178       19.79       30.67        1.000       0.3097E+12  X1         0
d2:     1.00    3250.64       1060.       3.066       ---          0.4208E+11  CT1        0
d2:     1.00    2561.20       1060.       2.416       ---          0.4208E+11  CT2        0
d2:     1.00    975.259       1058.      0.9221       ---          0.4208E+11  CT3        0
d2:     1.00   -1533.87       1058.      -1.450       ---          0.4208E+11  CT4        0
d2:     1.00   -11311.9       1060.      -10.67       ---          0.4208E+11  CT5        0
d2:     1.00   -17582.5       1060.      -16.58       ---          0.4208E+11  CT6        0
       -----
        10.0
```

Column 1

The first column indicates categorical variables and variables with linear constraints that have been specified in the GAM model. A categorical variable is specified in the GAMFIT procedure using the [factor,df] designation and indicated as "d2:" in the model summary. A variable that has been linearly constrained is indicated as "lin:".

Column (df)

The df column indicates the number of degrees of freedom specified for the GAM smoothing function associated with a predictor variable or categorical variable. A value of 1 indicates the variable is constrained as linear.

Column (coef)

The coef column provides the estimated coefficients of the GAM model.

Column (st err)
The “st err” column provides the standard errors for the estimated coefficients.

Column (z score)
The “z score” column provides the z-score of the estimated coefficient.

Column (nl pval)
The “nl pval” column provides the approximate probability of a variable having a nonlinear relationship with respect to the dependent variable.

Column (lin res)
The “lin res” column provides the model’s sum of squares of the residuals if the coefficient is linearly restricted in the model.

Column (Name)
The Name column provides the name of the variable associated with the coefficient.

Column (Lag)
The Lag column indicates any lag associated with the variable.

1.2 Nonlinear Testing

The Hinich (1982) test has proved invaluable in detecting whether nonlinearity has been removed from the residuals. The crucial issue is whether after the nonlinearity is removed, can the model adequately forecast? If in the process of removing the nonlinearity the model is over-fit, the forecasting performance may deteriorate. While the Hinich (1982) test may miss some types on nonlinearity that could be filtered by the GAM model, any model failing the Hinich test is certainly a candidate for GAM modeling.

Since the Hinich (1982) test only detects third-order nonlinearity, and GAMFIT can address nonlinearity of orders greater than three, the GAM model can accommodate nonlinearity in data that cannot be detected with the Hinich diagnostic test. The Cleveland and Devlin (1988) nitrous oxide dataset discussed later illustrates just such a situation. For this case the Hinich test on the OLS model does not reject linearity, yet substantial improvements in fit are possible with the GAMFIT procedures.

```
Hinich82 Nonlinear Tests  -  OLS Residuals
--------------------------------------------------
Gaussality (Mean)    :        -1.208
Linearity  (Mean)    :        -0.365
```

The Hinich test is a one-tailed test. A test value of 2.0 (or greater) indicates that nonlinearity is detected at the 5% level.

1.3 Confusion Matrix

When a logistic model or probit model is considered, information about actual and predicted values can be classified in a confusion matrix (Kohavi and Provost, 1998). The confusion matrix adopted for the GAM-Logit modeling application has the following structure:

CONFUSION MATRIX		**Predicted**		
		Negative $prob < p_n$	*Positive* $prob > p_p$	*Unclassified* $p_n > prob < p_p$
Actual	*Negative*	a	b	U1
	Positive	c	d	U2

Given the threshold values for positive classifications p_p and negative classifications p_n, a confusion matrix provides various ratios to evaluate the classification power of the model. When $p_n = p_p$ then all predicted probability values will be classified as either negative or positive instances. If threshold values are specified such that $p_n < p_p$, then there is a possibility of predicted probability values falling outside the defined range and therefore becoming unclassified. The confusion matrix reports if any cases cannot be classified.

In addition to the above table which displays a cross tabulation of the number of predicted negative/positive classifications to the number of actual negative/positive instances, several standard ratios are computed:

Accuracy Rate	$\frac{a+d}{a+b+c+d}$	The accuracy rate is the proportion of the total number of correct predictions.
True Positive Rate (TP)	$\frac{d}{c+d}$	The true positive rate is also called the recall rate and is defined as the proportion of positive cases that were correctly identified by the model.
False Positive Rate	$\frac{b}{a+b}$	The false positive rate is the proportion of negative cases that were <u>not</u> identified correctly by the model.
True Negative Rate (TN)	$\frac{a}{a+b}$	The true negative rate is the proportion of negative cases that were identified correctly by the model.
False Negative Rate	$\frac{c}{c+d}$	The false negative rate is the proportion of positive cases that were <u>not</u> classified as negative by the model.
Precision rate (P)	$\frac{d}{b+d}$	The precision rate is the proportion of predicted positive cases that were correctly classified by the model.

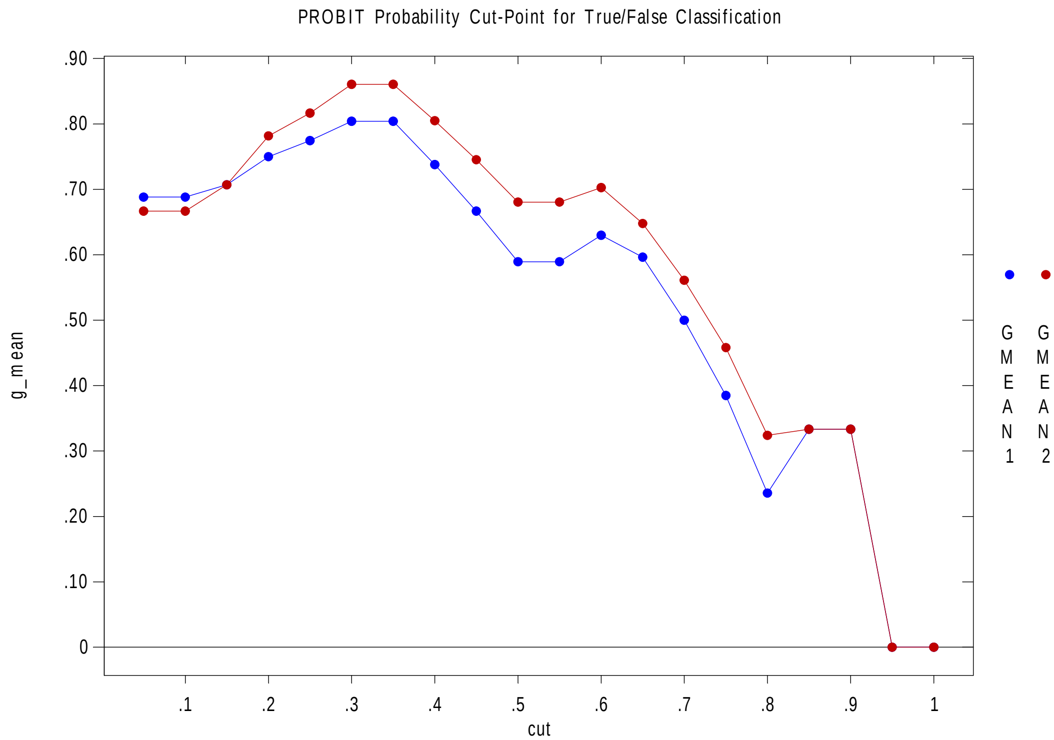
g-mean1	$\sqrt{TP \bullet P}$	When the number of negative cases is much greater than the number of positive cases, the above accuracy rate may not be a good performance measure. For example, if there are 100 cases of which 5 are positive and the remaining negative, the accuracy rate of a model that predicted all negative would be 95% even though the model failed to correctly classify a single positive case. To account for this situation, Kubat et al. (1998) suggest the geometric mean where the true positive rate is in the product.
g-mean2	$\sqrt{TP \bullet TN}$	The g-mean2 equation is based on the same principle as the g-mean1 equation. Kubat et al (1998).

The determination of probability threshold values for classification purposes is subjective. It is often set based on achieving the highest True Positive (recall) rate while minimizing/maximizing another rate such as False Positive rate or True Negative rate. The final choice of threshold cut-off is most often dictated by the purpose of the classification model.

For purposes of evaluating the impact of various probability threshold values on the classification results, the GAM application provides an extended table that computes the various ratios across the probability cut-off range of 0-1 as shown below.

```
--------------------------------------------------------------------------------
                    PROBIT True/False Probability CutOff Range
--------------------------------------------------------------------------------
   Cut  True-Pos    True-Neg  False-Pos  False-Neg  Accuracy Precision   GMean1   GMean2
 0.050     1.000       0.444      0.556      0.000     0.630     0.474    0.688    0.667
 0.100     1.000       0.444      0.556      0.000     0.630     0.474    0.688    0.667
 0.150     1.000       0.500      0.500      0.000     0.667     0.500    0.707    0.707
 0.200     1.000       0.611      0.389      0.000     0.741     0.562    0.750    0.782
 0.250     1.000       0.667      0.333      0.000     0.778     0.600    0.775    0.816
 0.300     0.889       0.833      0.167      0.111     0.852     0.727    0.804    0.861
 0.350     0.889       0.833      0.167      0.111     0.852     0.727    0.804    0.861
 0.400     0.778       0.833      0.167      0.222     0.815     0.700    0.738    0.805
 0.450     0.667       0.833      0.167      0.333     0.778     0.667    0.667    0.745
 0.500     0.556       0.833      0.167      0.444     0.741     0.625    0.589    0.680
 0.550     0.556       0.833      0.167      0.444     0.741     0.625    0.589    0.680
 0.600     0.556       0.889      0.111      0.444     0.778     0.714    0.630    0.703
 0.650     0.444       0.944      0.056      0.556     0.778     0.800    0.596    0.648
 0.700     0.333       0.944      0.056      0.667     0.741     0.750    0.500    0.561
 0.750     0.222       0.944      0.056      0.778     0.704     0.667    0.385    0.458
 0.800     0.111       0.944      0.056      0.889     0.667     0.500    0.236    0.324
 0.850     0.111       1.000      0.000      0.889     0.704     1.000    0.333    0.333
 0.900     0.111       1.000      0.000      0.889     0.704     1.000    0.333    0.333
 0.950     0.000       1.000      0.000      1.000     0.667     0.000    0.000    0.000
 1.000     0.000       1.000      0.000      1.000     0.667     0.000    0.000    0.000
```

By default, the GAM application uses a cut-off probability threshold that maximizes G-MEAN1 or G-MEAN2. In the above table, the cut-off would be computed as 0.325 = (.30+.35)/2. In addition to the table, a graph is generated to chart the G-MEANs against the cut-off probabilities.



1.4 Lift-Gain Table

A lift-gain table measures how well a classification system (e.g. logistic model) predicts the probability of a positive outcome. The data are sorted from highest probability to lowest probability and grouped into deciles. The primary statistics displayed in the Lift-Gain table are computed as follows

$$\text{Cumulative Gain}_d = \frac{\sum_{t=1}^{d} \text{Positives}_t}{\text{Total Positives}}, \quad d=1,2,\ldots,10$$

$$\text{Lift-Index}_d = \frac{\sum_{t=1}^{d} \text{Positives}_t}{d\left(\text{Total Positives}/10\right)}, \quad d=1,2,\ldots,10$$

$$\text{K-S Spread}_d = \frac{\sum_{t=1}^{d} \text{Positives}_t}{\text{Total Positives}} - \frac{\sum_{t=1}^{d} \text{Negatives}_t}{\text{Total Negatives}}, \quad d=1,2,\ldots,10$$

$$\text{Gain-over-Random}_d = 1 - \left(\frac{d}{10} \Big/ \text{Cumulative Gain}_d\right), \quad d=1,2,\ldots,10$$

The K-S (Kolmogrov-Smirnov) spread is a nonparametric statistic that tests whether two samples are from the same population. By comparing the proportion of positive and negative cases at each decile, the maximum K-S spread indicates the decile where the model classifies the greatest number of positive cases in the scoring population while minimizing the number of misclassifications. In acquisition modeling, the K-S spread is used to determine a cut-off percentage of the scoring universe to be targeted with consideration to opportunity costs. In the example below, a holdout sample of 100 cases have been scored using a logistic GAM model. From the 100 sample, there are a total of 25 positive outcomes. Based on a non-modeling approach and the assumption that the number of positive outcomes are distributed evenly across the sample, one would expect to observe 10% of the total positive outcomes to be cumulated in the first decile, 20% of the total positive outcomes to be cumulated in the second decile, and so on. This expected cumulative gain is compared against the cumulative positive outcomes obtained by the classification system for each decile. Below, the logistic GAM model classified 40% of the total positive outcomes in the first decile, 80% in the second decile, and 100% in the fourth decile. Furthermore, the K-S spread is greatest in the third decile which indicates that the classification system provides greatest return within the top 30% scored.

The lift can simply be thought of as the ratio between the cumulative gain of the classification system and the cumulative expected gain.

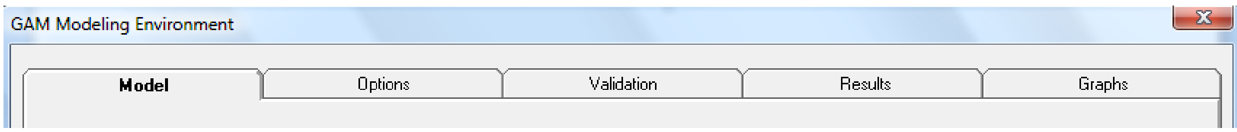
GAM Lift-Gain Table

Decile	#Obs in Decile	#Pos in Decile	%Pos in Decile	Pctg of Total Pos	Cum. #Obs	Cum. #Pos	Cum. %Pos	Cum. Gain	K_S Spread	Lift Index	Gain over Random
1	10	10	100.0%	40.0%	10	10	10.0%	40.0%	40.0%	400	75.0%
2	10	10	100.0%	40.0%	20	20	20.0%	80.0%	80.0%	400	75.0%
3	10	4	40.0%	16.0%	30	24	24.0%	96.0%	88.0%	320	68.8%
4	10	1	10.0%	4.0%	40	25	25.0%	100.0%	80.0%	250	60.0%
5	10	0	0.0%	0.0%	50	25	25.0%	100.0%	66.7%	200	50.0%
6	10	0	0.0%	0.0%	60	25	25.0%	100.0%	53.3%	166	40.0%
7	10	0	0.0%	0.0%	70	25	25.0%	100.0%	40.0%	142	30.0%
8	10	0	0.0%	0.0%	80	25	25.0%	100.0%	26.7%	125	20.0%
9	10	0	0.0%	0.0%	90	25	25.0%	100.0%	13.3%	111	10.0%
10	10	0	0.0%	0.0%	100	25	25.0%	100.0%	0.0%	100	0.0%

2. SCA WorkBench: A Graphical User Interface

SCA WorkBench provides a convenient graphical user interface to SCAB34S SPLINES for GAM modeling. The WorkBench interface builds the data loading steps and commands based on the user's menu selections. The associated commands are then organized as an SCAB34S program file and submitted to the SCAB34S engine.

The GAM modeling environment in WorkBench is organized by tabs shown below.



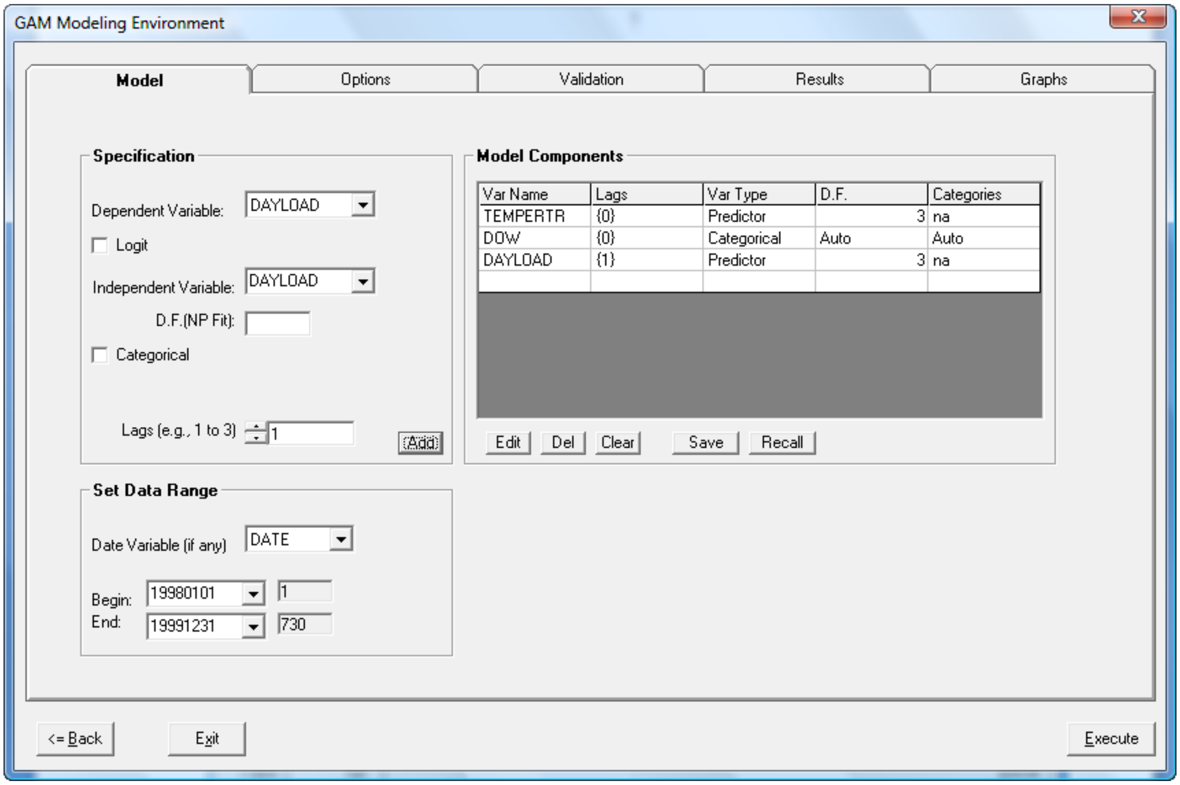
The *Model* tab is used to specify the variables, variable types, and lagged components of the GAM model. The *Options* tab sets the estimation limits placed on a GAM model, sets the linking function and error distribution type, and controls the detail of output and graphics that are produced. The *Validation* tab provides settings to evaluate the performance of GAM model prediction and to compare the results with a linear regression model (OLS, MINIMAX, L1, LOGIT, or PROBIT estimation). The *Results* tab displays the input/output from the model estimation, diagnostics, and forecasting. The *Graphs* tab displays a variety of high resolution graphics such as time series plots, residual plots, autocorrelation plots, surface plots, and others.

Once the SCAB34S program file is created by SCA WorkBench, you may save the file for future reference or make changes directly to the commands and re-execute the script from SCA WorkBench.

2.1 Model Specification Tab

This tab is central to specifying the variables and lagged components of the GAM model. Use the dropdown combo boxes to select your dependent variable and predictor variables. Click on the **Add** button to add a predictor variable component to the model. A categorical variable can be added by putting a check in the Categorical checkbox before clicking on the **Add** button. To allow a GAM model to be compared with a linear model, a categorical variable is automatically expanded into 0-1 binary variables which are then substituted in both the GAM and linear comparison model. When a variable is added into the model, the component will appear in the *Model Components* grid as they are added. In the example below, DAYLOAD is selected as the dependent variable. TEMPERTR is selected as an independent variable with a contemporaneous effect and a lag 1 effect. The lags are specified in the **Lags** textbox. Multiple lags for explanatory variable components can be specified using the word "TO" to separate contiguous lags (e.g., 0 TO 1) or commas to separate non-contiguous lags (e.g., 0, 1, 3).

A component may be deleted or modified by placing your cursor on the specific row of the Model Components grid and then by clicking on the **Del** or **Edit** buttons. If you click on **Edit**, the **Add** button will be replaced by the **Mod** button. You may make changes using the dropdown combo box for the independent variable and other components in the *Specification* frame. Click on the **Mod** button to complete the modification.



The features of the Model specification tab are presented below.

Menu Item	Description
Specification Frame	This frame organizes various controls that you may use to specify GAM model components including the dependent variable, independent variables, and lag coefficients. If a categorical variable is specified for an independent variable, the GAMFIT routine will automatically identify it as categorical when it is processed and expand it into 0-1 binary variables.
Dependent Variable	Use this drop-down list to specify the series that you wish to analyze.
Logit Checkbox	Specifies that the independent variable is a 0-1 variable. When specified, the GAM model estimates the probability of success/failure based on the independent variables in the model using the logit linking function.
Categorical Checkbox	Specifies that the dependent variable is a categorical variable. When specified, the application will automatically determine the number of categories (must be coded as integer) and expand the categorical variable into binary (0-1) variables.

Independent Variable — Use this drop-down list to specify a predictor or categorical variable components in the model.

Lags — Specifies the lag parameters associated with a random variables or categorical variables. A categorical variable may contain more than one lag parameter; however only one lag specification may be added to the model at a time. For random variables, multiple lag parameters may be added to the model as a group. Multiple lags may be specified using the "TO" keyword to separate contiguous lags. Individual lags may be separated by commas. For example, the user could specify contiguous lags as "0, 1, 3" or as "0 TO 1, 3".

D.P. (NL fit) — Specifies the number of degrees of freedom to be used on the variable for smoothing. Specifying the degrees of freedom to 1 restricts the variable as linear. The default is 3 (quadratic).

Add — Clicking on Add appends a new component to the GAM model which is displayed in the model component grid. Multiple instances of the same independent variable may be added to the model as long as the lag operators are unique. For example, in the above form, the user could specify TEMPERTR{0} and TEMPERTR{1} components separately.

Model Components Frame — The model components frame organizes form controls to display the GAM model components in a grid format, as well as to edit and delete model components.

Model Component Grid — The components of the GAM model and their attributes are displayed in this grid. The first column displays the independent variable name, the second column displays the individual or grouped lag operators within braces, the third column indicates whether the independent variable is predetermined as a predictor or categorical. The fourth column indicates the number of degrees of freedom for smoothing, and fifth column indicates that a categorical variable is specified and that the number of unique categories will be determined by the program.

Edit — The user can modify a model component by first placing the mouse cursor on the grid row of interest and then clicking on the Edit button. The Specification Frame will reflect the current attributes of the model component and the Add button will be replaced by the Mod button. Make the necessary changes in the Specification Frame and then click on the Mod button to complete the changes.

Del — The user can delete a model component by placing the mouse cursor on the grid row of interest and then clicking on the Del button.

Clear — Clears all model components from the model component grid.

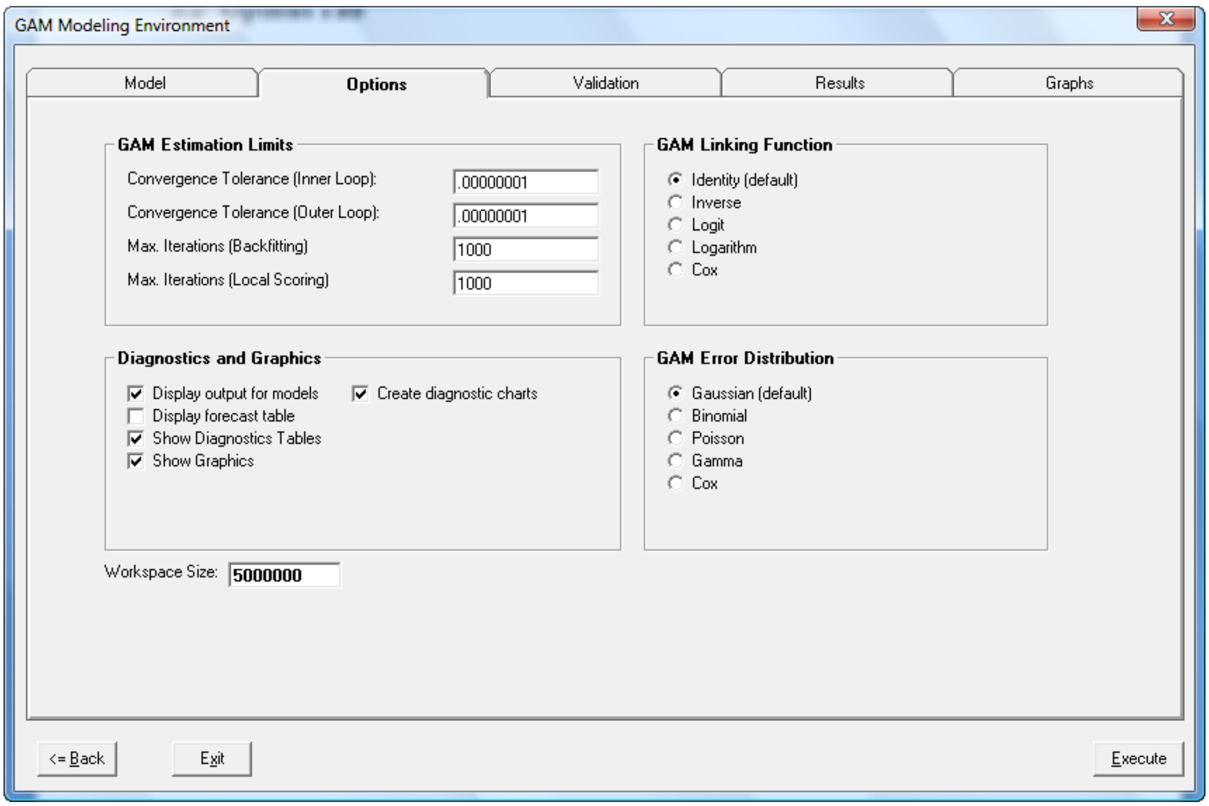
Save — Saves the information in the model component grid to a specified tab-delimited file.

Recall — Recalls the model component grid information from a specified tab-delimited file created (see Save option above).

Set Data Range Frame	This frame organizes form controls related to how the data is indexed (by date or none), and what data span is modeled and analyzed.
Date Variable	Use this drop-down list to specify the date variable associated with your series. If your SCA Data Macro contains a variable named "DATE", it is automatically assigned by SCA WorkBench. If you have an alternative index variable or date variable, you may select it from the drop-down list. If your SCA Data Macro does not contain a DATE variable, leave the dropdown list empty. WorkBench will then use the observation number as a date index. If your time series is more than 10,000 observations, WorkBench will not use your DATE variable for indexing. Instead, observation number will be used.
Begin Span	Use the Begin drop-down list to omit observations from the beginning of a time series being analyzed.
End Span	Use the End drop-down list to omit observations from the back of a time series being analyzed.
Back	Depending on the tab you are currently working in, clicking on the Back button will move you one tab to the left. If you are in the Model tab, you will move to the GAM Data Viewer dialog box where you may choose a new SCA data macro or leave the GAM Modeling Environment.
Exit	Exits the GAM modeling environment.
Execute	Executes GAM model estimation, validation, linear model comparison, diagnostics, and graphs by submitting a dynamically created program script to SCAB34S SPLINES. When completed, you will automatically be placed in the Results tab.

2.2 Options Tab

The Options tab sets the estimation limits placed on a GAM model, controls the detail of output and graphics that is produced, and allocates the workspace size of the SCAB34S SPLINES product. More estimation options are available in the GAMFIT matrix subroutine that are not exposed in this GAM Modeling Environment interface. The user may employ these other options by directly editing the B34S script generated by WorkBench.



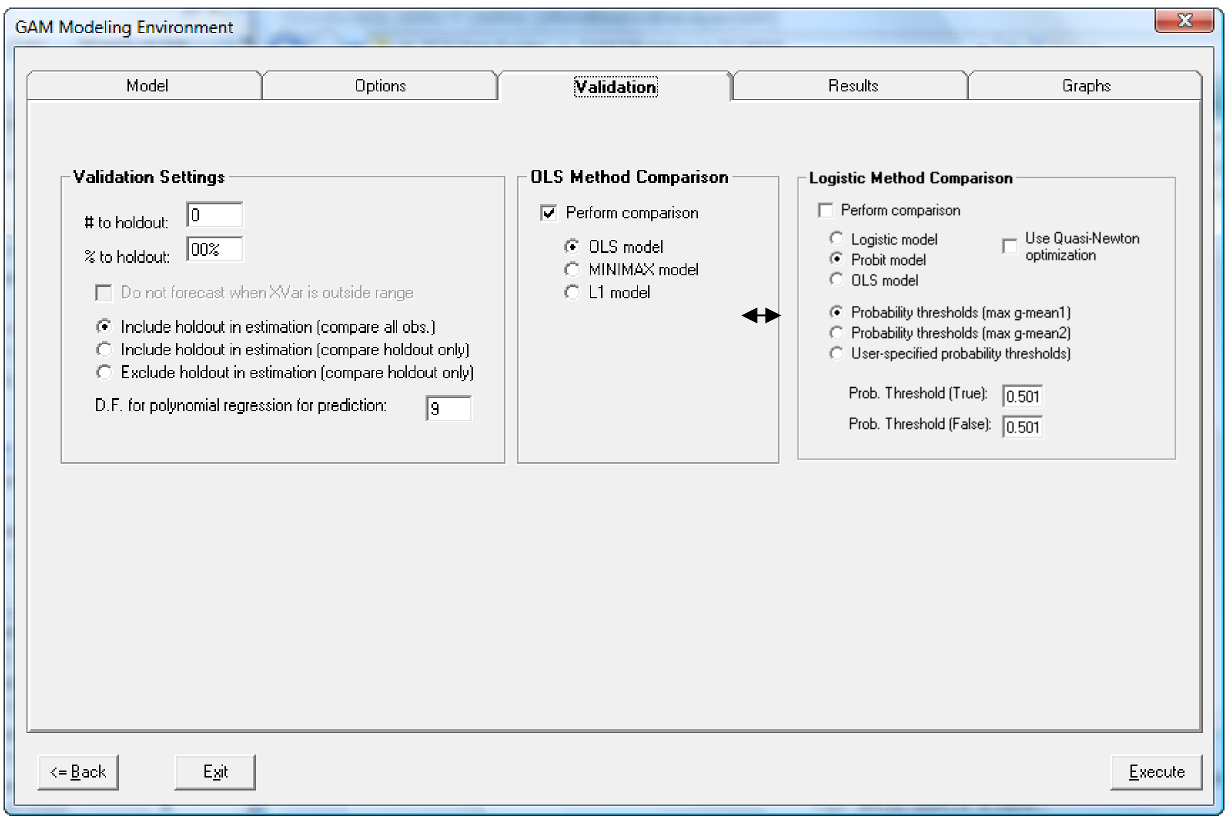
Menu Item	Description
GAM Estimation Limits Frame	This frame organizes various controls that set options in GAM model estimation. Here, the user specifies the convergence tolerance for inner and outer looping, and the maximum number of iterations for back-fitting and local scoring.
Convergence Tolerance (Inner Loop)	Set the convergence tolerance for inner looping in the GAM smoothing algorithm. The default value is 0.1D-8
Convergence Tolerance (Outer Loop)	Set the convergence tolerance for outer looping in the GAM smoothing algorithm. The default value is 0.1D-8
Max. Interactions	Set the maximum number of iterations for back-fitting. The default is 1000.

(Back-fitting)

Max. Iterations (Local scoring)	Set the maximum number of iterations for local scoring. The default is 1000.
Diagnostics and Graphics Frame	This frame organizes controls related to the amount of output produced for GAM estimation and diagnostics. The diagnostic charts option produces surface (or leverage) charts for all variables that are used in the final model.
Display Output for Model	Typically, you want to see the GAM model summary and the OLS model summary.
Display Forecast Table	The forecast table displays the original series and the predicted series for both the GAM model and OLS models. This can slow down the display of output for larger datasets.
Show Diagnostic Tables	Several diagnostics are available for the dependent variable and the residuals from the estimated models. Among the diagnostics are a statistical description tables, sample autocorrelation tables, and Hinich nonlinear testing. The Hinich test wil only be displayed for residual series greater than 50 cases.
Show Graphics	Several graphics are created including time plot of the dependent variable, Actual vs. Predicted, ACF and PACF plots, and modified Q-Statistic plot.
Workspace Size	The SCAB34S SPLINES product requires its workspace size to be set when the program is initiated. The default workspace is of 2000000 is adequate to handle moderate size datasets. The user may increase the workspace size if needed. Please note that workspace limit is imposed by the amount of available RAM memory of the computer.
GAM Linking Function Frame	The GAM model requires the specification of a nonlinear link function to declare how the mean of the dependent variable is dependent upon the additive predictor. The error distribution can also be specified.
GAM Linking Function	Specify the nonlinear link function between the mean of the dependent variable and the additive predictor. The available options are identity, inverse, logit, logarithm, and Cox. The default is identity.
GAM Error Distribution	Specify the assumed error distribution for fitting. The available options are Gaussian, Binomial, Poisson, Gamma, and Cox. The default is Gaussian.

2.3 Validation Tab

This tab allows you to evaluate the performance of GAM model prediction and validate the GAM model against a linear regression model method using simple OLS, MINIMAX, L1, Logit or Probit estimation. A common problem with most nonlinear modeling methods is over-fitting. Models that over-fit the data often perform well within the sample, but do substantially worse when predicting out of sample. Comparing in-sample fit and out-of-sample prediction performance allows the user to evaluate problems related to over-fitting. If over-fitting is suspected, the number of degrees of freedom for GAM smoothing should be reduced for one or more variables of concern. Also, since out-of-sample GAM prediction is accomplished using a polynomial regression approach to approximate the smoothing splines, the setting for number of D.F. for polynomial regression may also affect out-of-sample prediction performance. A low setting may not be able to adequately approximate the curvature whereas a high setting may cause an estimation error. A setting between 3-9 is reasonable for most situations.



The GAM modeling approach can be used effectively for both cross-sectional data and time series data. The GAM user interface offered in WorkBench leverages its utility in time series applications by allowing the dependent variable and predictor variables to be lagged.

The default validation setting compares the in-sample fit of the estimated GAM model against the in-sample fit of a simple OLS regression model. All available observations are used to evaluate fit using root mean squared error (RMSE) and mean absolute percentage error (MAPE) criteria.

Other options are available to validate the GAM model. For example, if the user is primarily interested in evaluating the fit of the model in the later part of the series, a holdout sample can be specified by typing the number of observations (or percentage) to be marked from the back of the series. After specifying the holdout, the user can evaluate in-sample fit for the "holdout period" only by setting the option "Include holdout in estimation (compare holdout only)". The user also has two choices to evaluate the prediction performance of the model where the holdout period is not used in training the model.

As another validation criterion, the user can compare the improvement of a GAM model versus a regression model with the same right-hand side variables. Diagnostics are produced for both the GAM and regression models. If the dependent variable is nonlinear in its response to the transformed (smoothed) regressor variables, the GAM model should reveal significant improvement in model fit and out-of-sample forecasting performance.

A confusion matrix is produced for the GAM-Logit model and the comparison linear model for evaluating classification power of the models. The user has a choice for determining the probability cut-off value for classification of positive and negative cases for the final confusion matrix. The user can allow the system to set the probability cut-off automatically using the maximum G-MEAN values as the criteria, or using specific cut-off values. If GMEAN1 is used, the cut-off will slightly favor True-Positive classifications and if GMEAN2 is used, the cut-off will consider equally True-Positive and True-Negative classifications. Since the determination of cut-off probability thresholds is subjective, a table of ratio statistics for a range of cut-off probability values is also provided in the output.

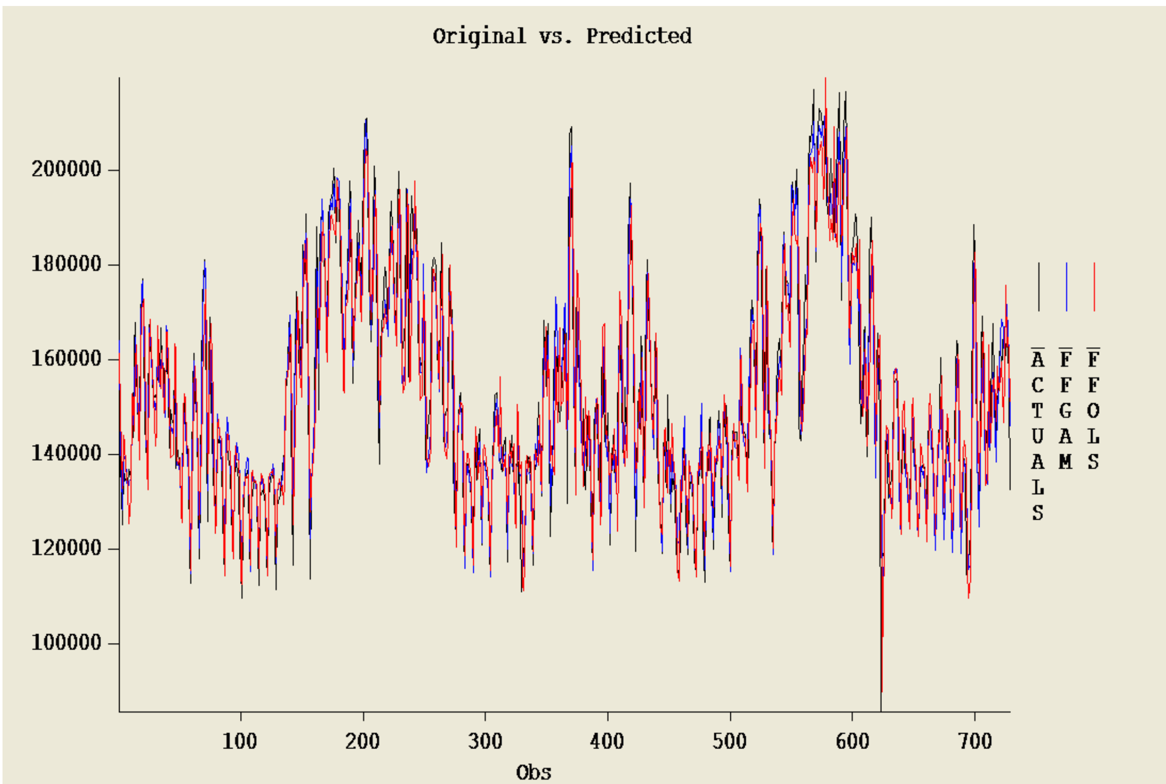
Menu Item	Description
Validation Settings Frame	This frame organizes controls for specifying a holdout sample for forecast performance and model validation. It also provides controls for the user to specify the type of validation for in-sample or out-of-sample forecasting.
# to holdout	Specifies the number of observations that are to be reserved from the back of the dependent variable for evaluating forecast performance. The percentage of the holdout sample relative to the series length is computed and is displayed in % to holdout.
% to holdout	Specifies the size of the holdout sample as a percentage of the length of the dataset. The actual number of observations reserved from the back of the series is computed and displayed in # to holdout.
Compare all obs	Evaluate the in-sample fit of the model for all observations.
Compare holdout only for in-sample fit	Evaluate the in-sample fit of the model for the defined holdout sample only
Compare holdout for out-	Evaluate the out-of-sample forecasts defined by the holdout sample. The model is estimated using observations up to the first

of-sample fit	forecast origin only
OLS Method Comparison Frame	This frame organizes controls to validate the GAM model against a regression model with the same right-hand-side variables used in the GAM model.
Logistic Method Comparsion Frame	This frame organizes controls to validate the logistic GAM model against a Logit or Probit model with the same right-hand side variables used in the logistic GAM model.
Perform comparison	By default a comparison is made to GAM using a simple OLS regression estimation method if the dependent variable is random. A comparison is not automatically performed if the dependent variable is specified as a logistical variable.
OLS model	Estimates a regression model using the ordinary least squares (OLS) method.
MINIMAX model	Estimates a regression model using the MINIMAX method which minimizes $\max\left(\text{abs}\left(Y_t - \hat{Y}_{t-1}\right)\right)$. This estimation method is more sensitive to outliers.
L1 model	Estimates a regression model using the L1 method which minimizes $\text{sum}\left(\text{abs}\left(Y_t - \hat{Y}_{t-1}\right)\right)$. This estimation method is not as sensitive to outliers as OLS or MINIMAX.
Logistic model	Estimates a logistic regression model in comparison to a logistic GAM model.
Probit model	Estimates a probit regression model in comparison to a logistic GAM model.
Probability thresholds	The threshold values for classifying a predicted case as a positive or negative instance.

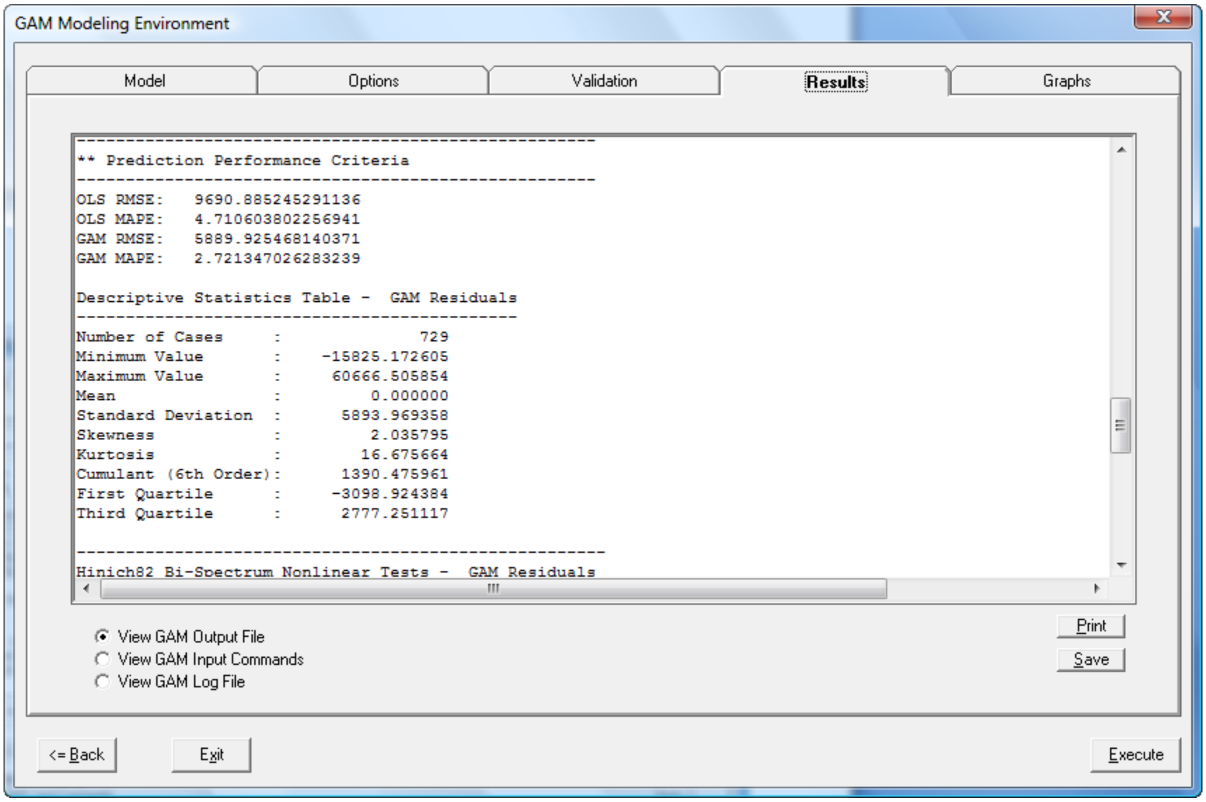
2.4 Results Tab

The results tab provides a convenient facility to view output from GAM model estimation. It also allows you to view the input commands for SCAB34S SPLINES execution. If there are errors during estimation, you can view the log file for a detailed account of all commands executed and error messages.

After the user executes the GAM model application by clicking on the **Execute** button, SCAB34S SPLINES will display a graph of the actual versus fitted data. This indicates that the GAMFIT procedure has completed. The user should click anywhere on the graph (an example is shown below) to close it.



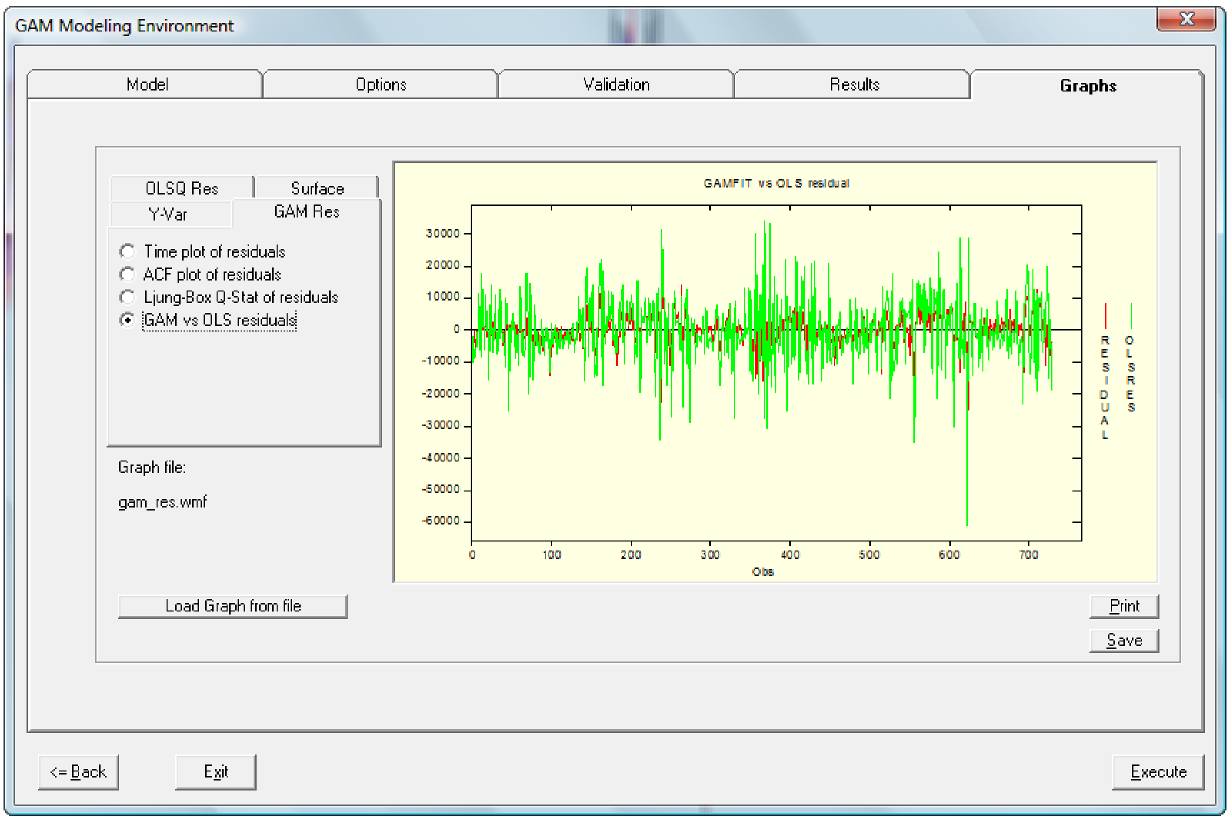
After the graph disappears, the user will be placed on the *Results* tab of the GAM Modeling environment where the output is listed.



Menu Item	Description
View GAM Output File	Displays the GAM modeling results and tabulated diagnostics.
View GAM Input Commands	Displays the input commands submitted to SCAB34S SPLINES. You can modify the commands directly in this window and submit the modified command file by clicking on the Execute button.
View GAM Log File	Displays a detailed command and error log for jobs submitted to SCAB34S SPLINES
Print	Send information displayed in the viewer to the printer.
Save	Saves the information in the viewer to a file. You may want to use this feature to save the modeling script with intentions of executing it later from the System -> Run SCA with Macro menu, or the System -> Run SCAB34S Program File menu.
Execute	While you are in the Results tab, if you click on Execute, you will send the information in the viewer to SCAB34S SPLINES for processing.

2.5 Graphs Tab

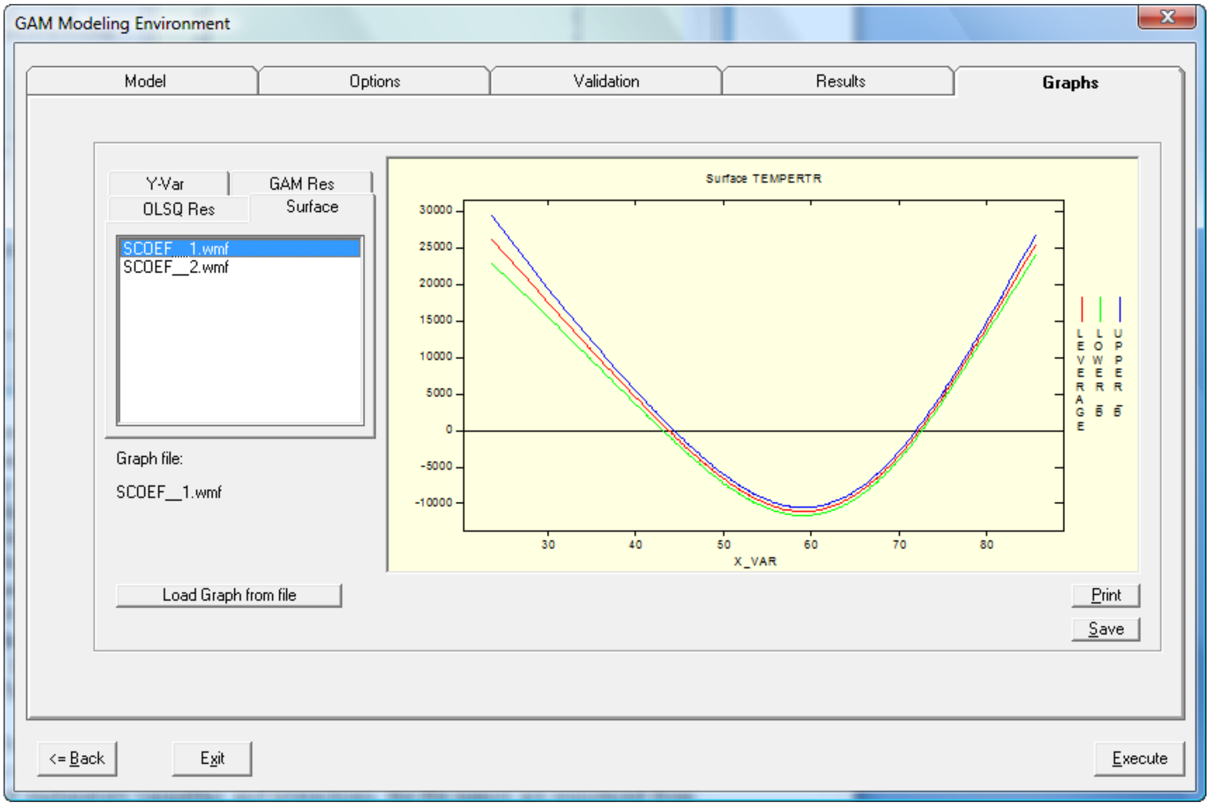
The Graphics tab provides a facility to view high-resolution plots that were generated. If you previously selected the *Show/Create Graphs* option, the individual graphs will initially be displayed on screen. When you click on the graph, the next generated graph will appear until all graphics have been created. As the graphs are displayed, they are also being saved as Windows Meta Files using fixed names such as “yvar.wmf” or “acfa.wmf”.



You can review all created graphic files by selecting the graph from the set of radio buttons provided in the small tabbed area to the left of the viewer control. In the example above, we are viewing the OLS and GAM residuals overlaid on each other. The name of the graphic file (gam_res.wmf) is displayed for reference. Since the graphs are saved to fixed file names, they are overwritten each time you generate a new set of graphs from the GAM modeling environment. If you wish to save the graphic file for future reference, please use the **Save** button on this tab to copy the file to a new name. Please do not rename the file extension because the **Save** button only renames the file. It does not convert it to a new format. You can view those renamed files by using the **Load Graph from File** facility. You may send the graph to the printer by clicking on the Print command button. If you double-click on the graph image it will load in the external program that is associated with WMF files on your computer (e.g., Windows FAX/Picture Viewer).

If you elected to create diagnostic charts, curvature plots of the transformed predictor variables in the GAM model are displayed relative to the dependent variable. Since a variable number of charts may be created based on number of explanatory variables, the file names are sequenced from

SCOEF___1 – SCOEF__## and may be viewed by selecting the file name from the list box provided. An example of a curvature chart is displayed below:



In the above graph, we are viewing the curvature of smoothed temperature. The SCOEF*.wmf files are overwritten, therefore the file should be renamed or moved to another location if the graph is to be saved for future reference.

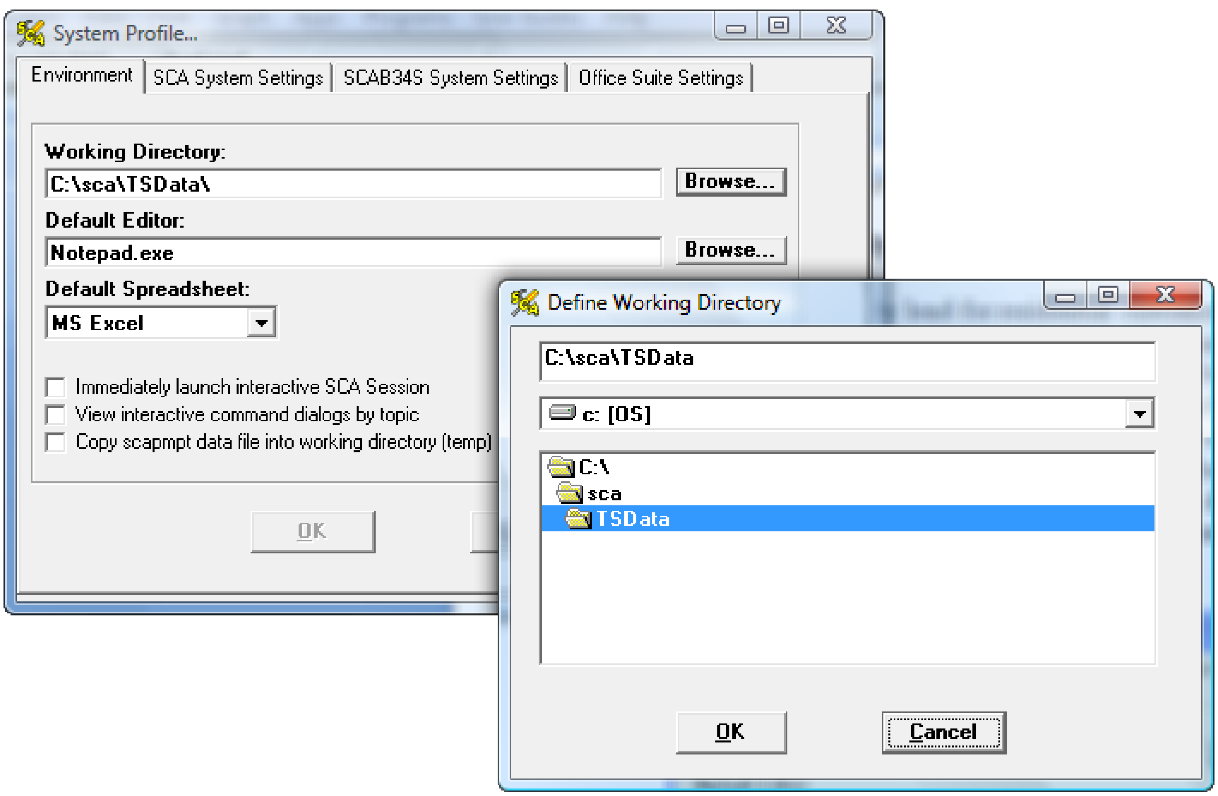
3. EXAMPLES OF GAM MODELING USING SCA WORKBENCH

This section provides various GAM modeling examples using SCA WorkBench and its interface to SCAB34S SPLINES. The first example uses GAM models to analyze the relationship of temperature to daily electricity load. A second example explores economies of scale using a production function studied by Nerlove. The third example uses a logistic GAM model to study cancer remission occurrences.

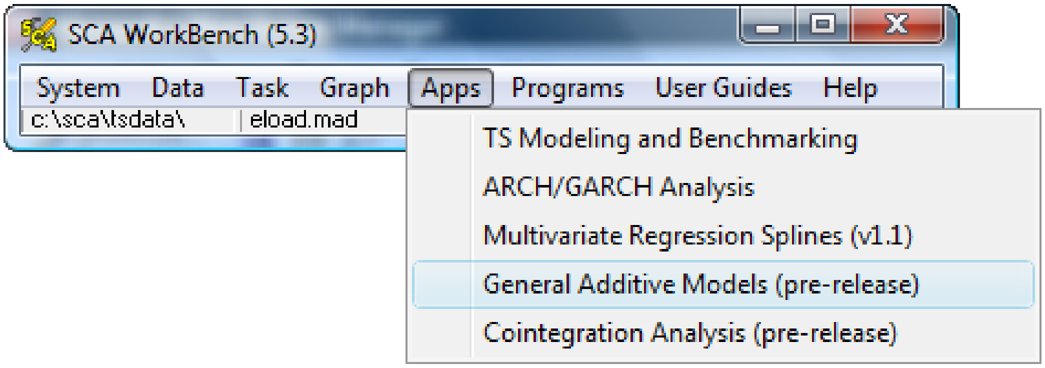
The data files used for the examples discussed in this section are available under the WorkBench installation folder in a sub-directory named **TSDATA**. The command files built by SCA WorkBench for the illustrated examples are presented later in Section 8 of this document.

3.1 Modeling Daily Electricity Load Using GAM

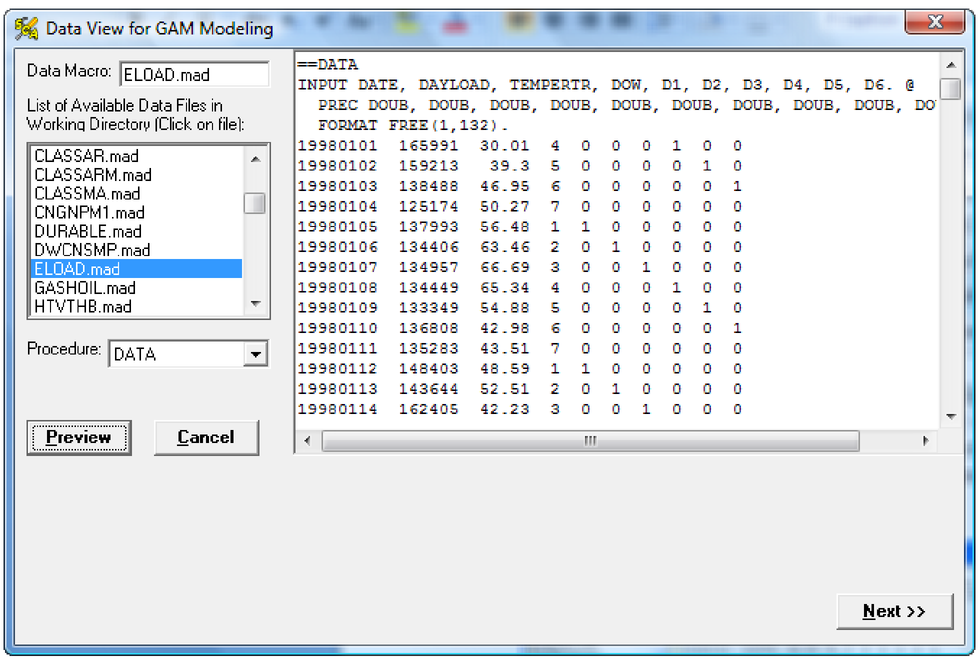
Daily total electricity load for residential customers is used to demonstrate generalized additive models (GAM). If you are working through this example, the first step is to set the working directory by selecting the *System Profile* item under the *System* menu of WorkBench. In the *Environment* tab of the *System Profile* dialog box, click on the *Browse* button associated with the working directory text box. Using the *Define Working Directory* dialog box, move to the **C:\SCA\TSData** directory that contains various sample data sets distributed with WorkBench. If WorkBench is installed under a different directory, the **TSData** subdirectory is located under the SCA installation directory. An example of the dialog boxes associated with modifying the working directory is shown below.



After the working directory is modified and the new profile is saved, click on the *General Additive Models* item under the *Apps* menu to enter the graphical user interface for GAM modeling as demonstrated below.



Once you click on the *General Additive Modeling* item, it is necessary to select the data to analyze. The *Data View for GAM Modeling* dialog box will automatically pop-up and display all SCA data macro files in the working directory. The daily electricity load data (DAYLOAD series) is located in the **ELOAD** data macro file under the **DATA** procedure. Please select this data set as illustrated below.



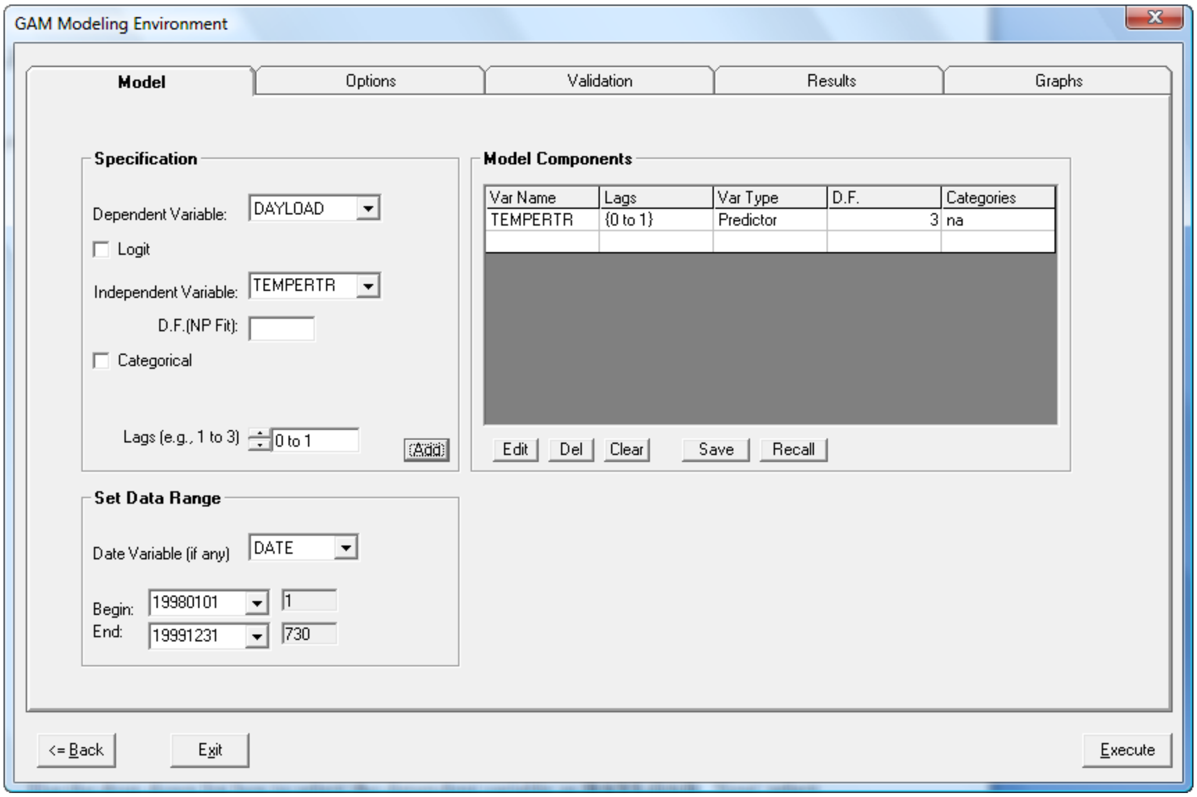
The data may be viewed by clicking on the ***Preview*** button. Here, we see that the ELOAD data macro contains ten variables:

DATE	→	YYYYMMDD
DAYLOAD	→	Daily total electricity load for residential customers
TEMPERTR	→	Average daily temperature
DOW	→	Day of week categorical variable (1=Monday, 2=Tuesday, etc.)
D1 to D6	→	Dummy variables for day of week (D1=Monday, D2=Tuesday, etc.)

In this example, we specify a GAM model for the DAYLOAD time series using average daily temperature as a predictor variable and the DOW categorical variable. The DOW variable is automatically expanded into separate 0-1 binary variables to accommodate linear model comparisons. Click on the *Next* button to enter the *GAM Modeling Environment*.

3.1.1 Specification of the GAM Model for Daily Electricity Load

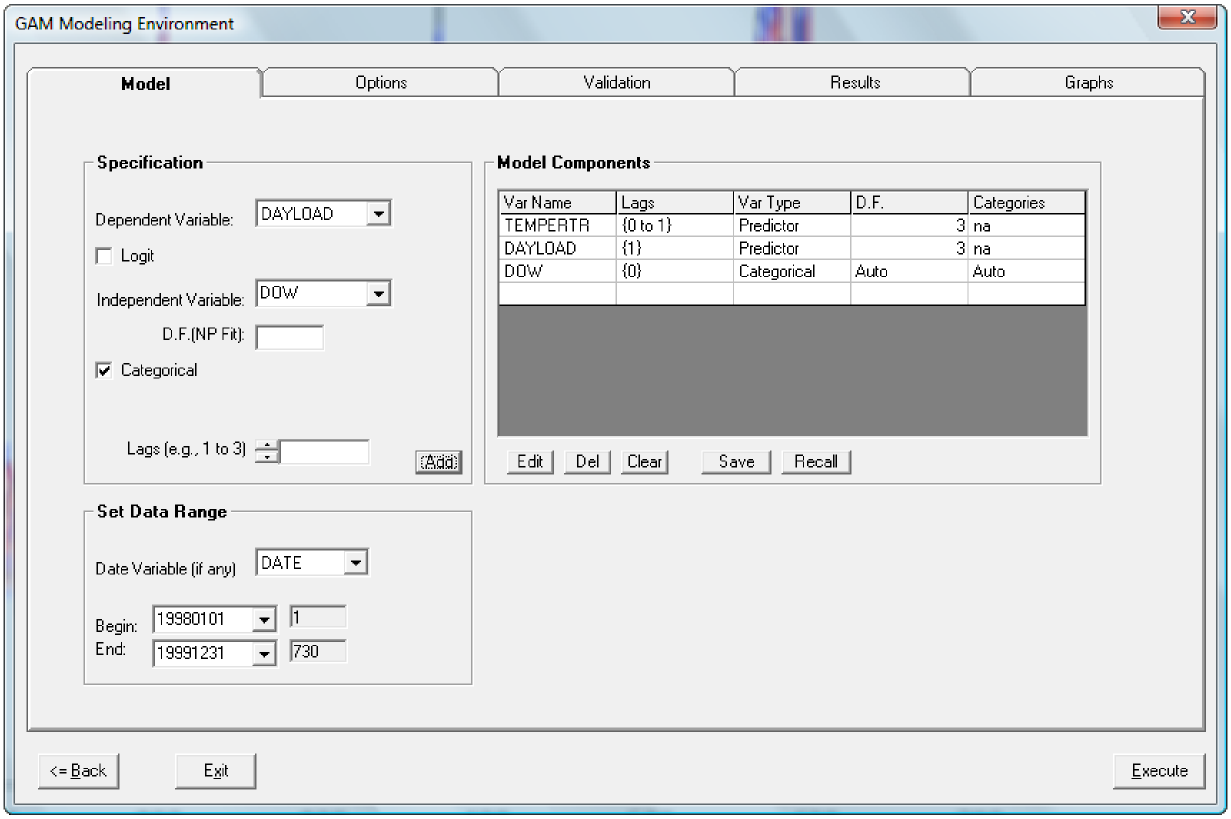
Once in the *GAM Modeling Environment*, click on the *Model* tab as shown below to specify a GAM model specification.



Use the drop down list box to select the dependent variable as **DAYLOAD**. Next, select temperature (**TEMPERTR**) as an independent variable. It is conjectured that temperature not only has a contemporaneous effect on electricity load, temperature from the previous day may also have some effect on electricity load. To accommodate a contemporaneous effect and lag effect, we specify "0 to 1" in the lags text box provided. Multiple lags may also be specified using commas. After specifying the **TEMPERTR** model component, click on the ***Add*** button to include it in the model. The model components are displayed in the Model Components grid. To modify or delete an existing model component, place your cursor on the grid row, and click on the ***Edit*** or ***Del*** buttons.

In addition to temperature, **DAYLOAD** at lag one is specified in the GAM model to partially offset the existence of autocorrelation from day to day electricity load. Finally, the **DOW** (day-of-week) categorical variable is included since day of week has a strong inference on electricity load (i.e., weekday and weekend patterns for electricity load differ considerably). Be sure to check the *Categorical* checkbox when adding the DOW variable to the model.

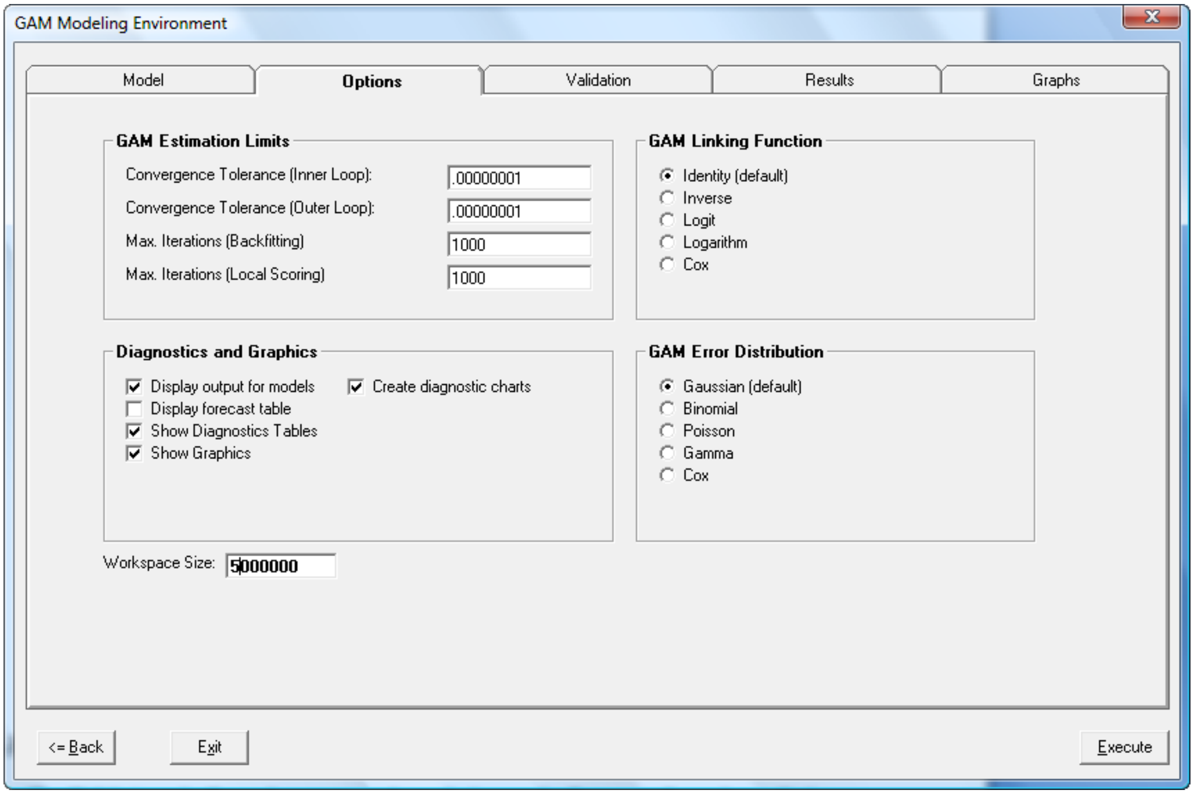
After specifying the GAM model for daily electricity load, the *Model* tab should like the one below:



3.1.2 GAM Model Options for the Daily Electricity Load Example

After specifying the GAM model, click on the *Options* tab. Here, frequently used options to set estimation criteria in GAM model estimation are provided. This tab also allows the user to select the level of detail for estimation summaries, diagnostics, and graphics.

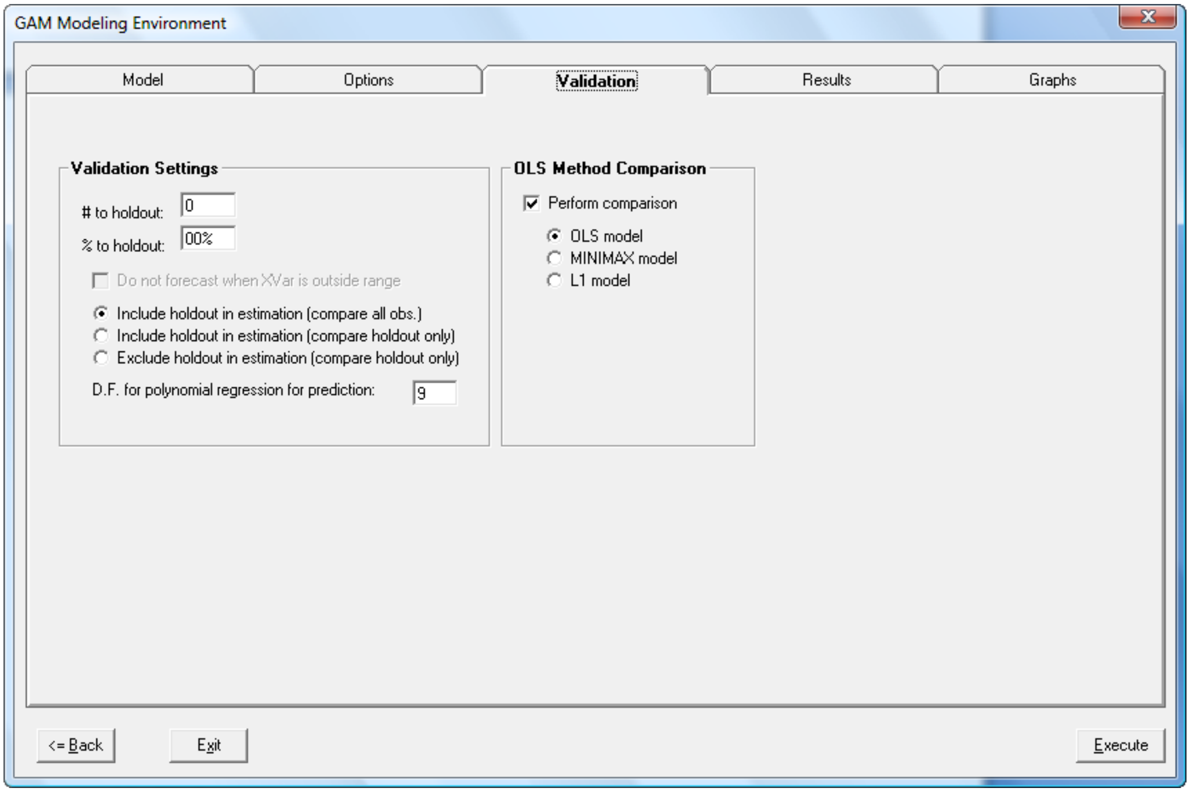
Typically, the default settings for the GAM estimation limits are adequate for most applications. You do not need to modify these settings unless an estimation error is encountered. We will use the *Identity* link function and the default *Gaussian* error distribution for the ELOAD example.



3.1.3 GAM Model Validation for the Daily Electricity Load Example

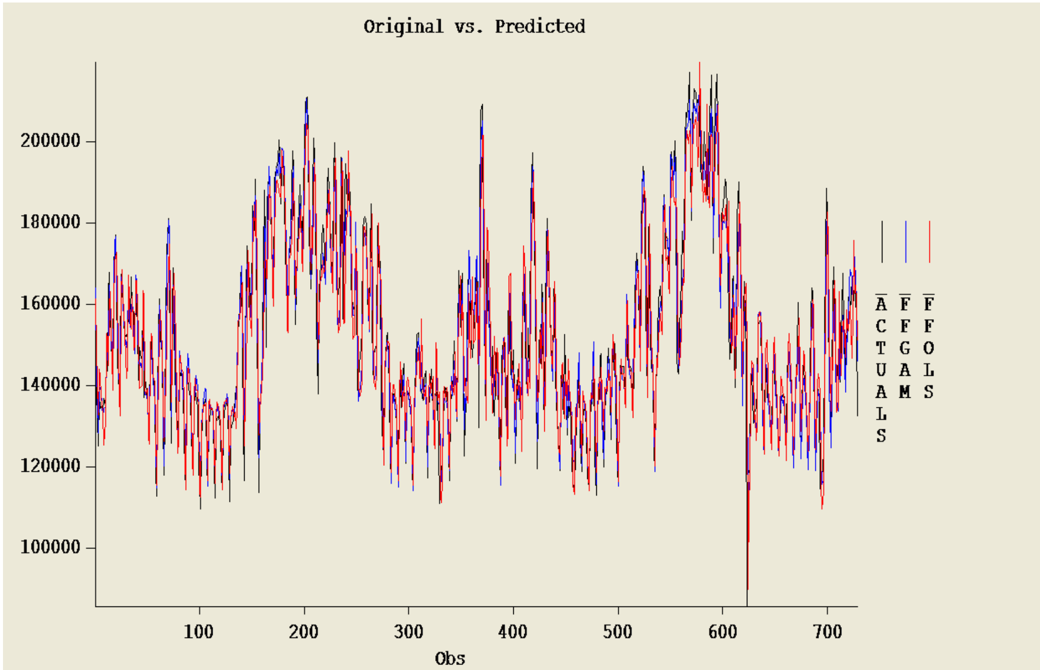
The *Validation* tab allows the user to evaluate the in-sample fit or out-of-sample predictive power of the GAM model compared to a linear regression model. As a first step, the overall in-sample fit of the GAM model will be examined and compared against a regression model that uses ordinary least squares (OLS) estimation.

Select the radio button that compares all observations and set the number of observation in the holdout period to zero as shown below. Next, click on the *Execute* button to run the analysis. The program script file will be automatically generated by SCA WorkBench and submitted to the SCAB34S engine for execution.



Once the SCAB34S engine is activated and the program script is loaded, an information box will appear on the computer screen indicating that calculations are in progress. When the analysis has finished, the information box will be replaced by a graph of the actual versus fitted values for both the GAM and OLS regression model.

An example of the graph is shown below. Click on the graph and it will disappear. However, this graph and other diagnostic graphs will be created in windows metafile format (WMF) and be saved to the user's working directory. The file names for the graphs are static and will be overwritten for each execution run. The plots can be viewed, printed, or saved under the *Graphs* tab.

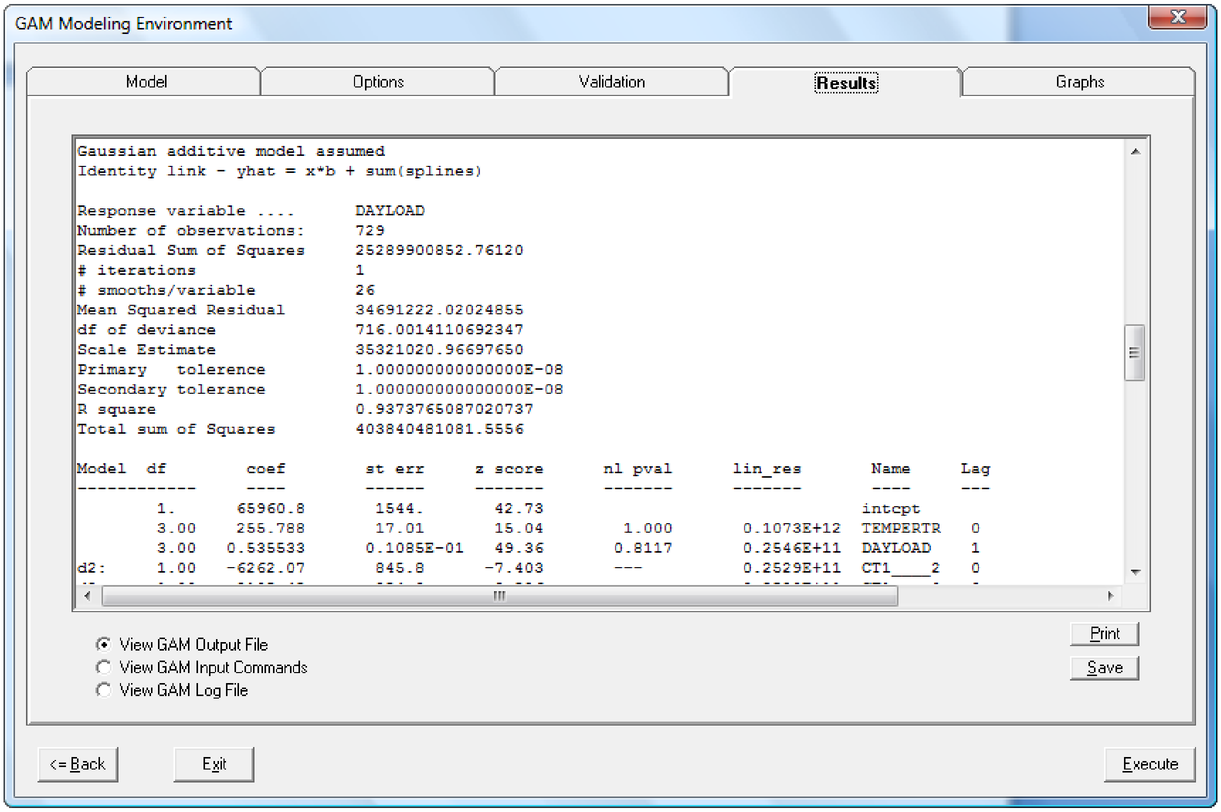


The _FFGAM series is the fitted values from the GAM model and the _FFOLS series is the fitted values from the linear regression model. It looks like both models have a decent fit, but it is not clear whether one model is much better than the other, at least from this graph. After clicking on the graph, the user will automatically be placed in the *Result* tab.

3.1.4 GAM Model Results for the Daily Electricity Load Example

The Results tab shows the output for the analysis. The user can also view the SCAB34S commands that were built by the SCA WorkBench interface. If desired, the input commands can be modified directly from the Results tab and re-submitted to SCAB34S SPLINES by clicking on the *Execute* button in the *Results* tab. The modified script can then be saved to a new file which can be executed from the System->Run SCAB34S Applet Program File menu.

Clicking on the *Execute* button from any other tab will trigger SCA WorkBench to rebuild the SCAB34S command script based on current settings in the *Model*, *Options*, and *Validation* tabs.



The output generated for a GAM analysis may be reduced by appropriate settings provided in the *Options* tab. The output always begins by summarizing the data series that has been brought into the workspace. This is followed by the OLS regression model summary, the GAM model details, followed by various diagnostic tests on the residuals (summary statistics, autocorrelation testing, nonlinear testing, etc.).

Below are segments of the output generated from the above analysis. Comments are also provided to help explain some of the results obtained. The first segment is a model summary of the OLS regression model based on the same regressor variables included in the GAM model. The linear regression model is used as a benchmark in evaluating the performance gain of a non-linear GAM model. It also provides insight regarding the importance of the parameters being considered.

The DOW categorical variable has been automatically expanded into individual 0-1 binary variables. The DOW variable is made up of seven categories (1-7) that correspond to the day-of-week (Mondays=1, Tuesdays=2, …, Sundays=7). Based on these unique values, seven binary variables were generated automatically by the program (CT1____1 to CT1____7). The first category was dropped from the model by default.

```
Categorical variable  DOW      split into 0-1 variables

Categorical  Value   Reference    Cases-On
 DOW          1.0     CT1____1        *** dropped in model
 DOW          2.0     CT1____2        104
 DOW          3.0     CT1____3        104
 DOW          4.0     CT1____4        105
 DOW          5.0     CT1____5        105
 DOW          6.0     CT1____6        104
 DOW          7.0     CT1____7        104
```

The model summary for OLS model estimation follows which is a benchmark comparison to the GAM model. Here, we see all variables included in the regression model are significant after estimation.

```
Final Model Estimation Summary -  OLS
Dependent Variable:               DAYLOAD
----------------------------------------------------------------
  # Variable   Lag     Coefficient      std.Err      t-value     Comments
----------------------------------------------------------------
  1 TEMPERTR     0        -394.922       70.404       -5.609
  2 TEMPERTR     1         446.461       69.130        6.458
  3 DAYLOAD      1           0.897        0.017       51.765
  4 CT1____2     0      -13094.057     1350.394       -9.696
  5 CT1____3     0      -15689.850     1364.489      -11.499
  6 CT1____4     0      -17482.286     1363.351      -12.823
  7 CT1____5     0      -18937.460     1353.272      -13.994
  8 CT1____6     0      -25767.282     1345.041      -19.157
  9 CT1____7     0      -22905.939     1319.950      -17.354
 10 CONSTANT     0       28844.713     2478.529       11.638

Number of observations:                      729
Adjusted R-Square:                         0.838
Sum of Squared Residuals:              0.647E+11
Schwartz Information Criteria:         15483.108
```

The GAM model summary is displayed next. The z-score of the coefficients provide support that all variables are significant. We also see evidence that TEMPERTR is nonlinear based on the approximated probability of nonlinearity, ***nl pval***, and the increase in residual sum of squares of the model when TEMPERTR is restricted to be linear, ***lin_res***.

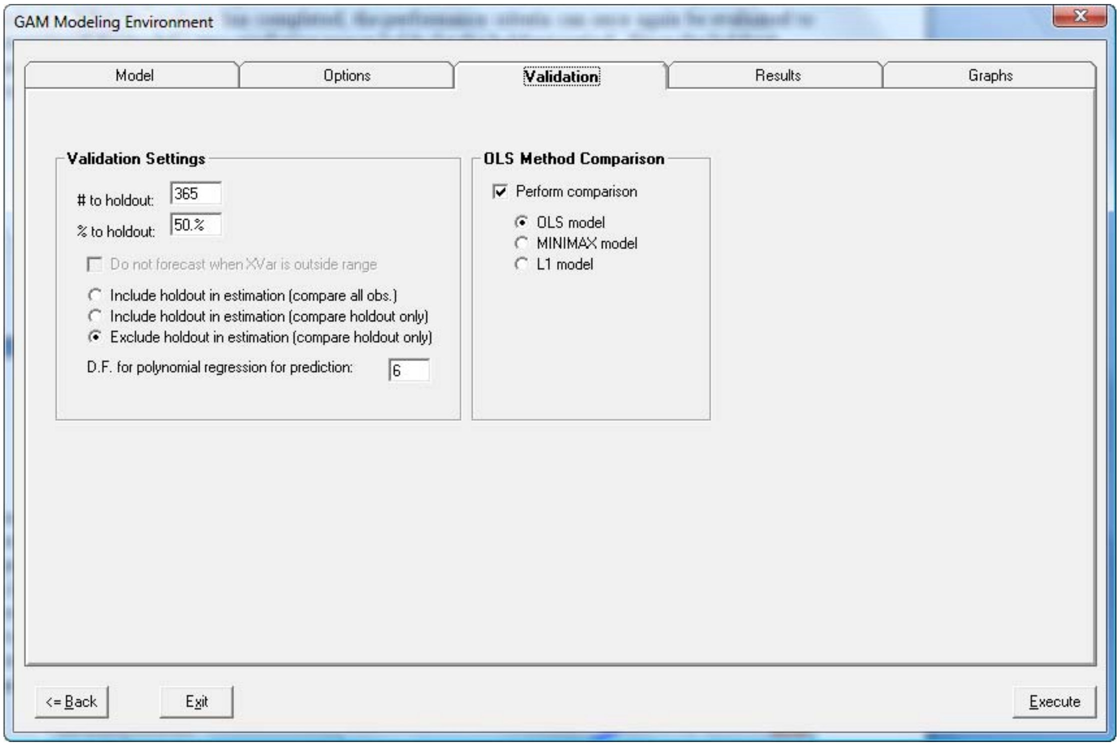
```
Generalized Additive Models (GAM) Analysis
Reference: Generalized Additive Models by Hastie and Tibshirani. Chapman (1990)
Model estimated with GPL code obtained from R.

Gaussian additive model assumed
Identity link - yhat = x*b + sum(splines)

Response variable ....      DAYLOAD
Number of observations:     729
Residual Sum of Squares     25289900852.76120
# iterations                1
# smooths/variable          26
Mean Squared Residual       34691222.02024855
df of deviance              716.0014110692347
Scale Estimate              35321020.96697650
Primary   tolerence         1.000000000000000E-08
Secondary tolerance         1.000000000000000E-08
R square                    0.9373765087020737
Total sum of Squares        403840481081.5556

Model  df       coef        st err     z score      nl pval      lin_res       Name     Lag
------------    ----        ------     -------      -------      -------       ----     ---
        1.      65960.8       1544.      42.73                                intcpt
        3.00    255.788       17.01      15.04        1.000       0.1073E+12  TEMPERTR   0
        3.00   0.535533     0.1085E-01   49.36       0.8117       0.2546E+11  DAYLOAD    1
d2:     1.00   -6262.07       845.8      -7.403       ---         0.2529E+11  CT1____2   0
d2:     1.00   -8135.45       854.0      -9.526       ---         0.2529E+11  CT1____3   0
d2:     1.00   -9870.34       854.0     -11.56        ---         0.2529E+11  CT1____4   0
d2:     1.00   -11679.8       847.6     -13.78        ---         0.2529E+11  CT1____5   0
d2:     1.00   -19733.7       842.4     -23.43        ---         0.2529E+11  CT1____6   0
d2:     1.00   -20766.7       826.9     -25.11        ---         0.2529E+11  CT1____7   0
       -----
        13.0
```

We now can explore the in-sample fit of the GAM model in comparison to the benchmark regression model. The table below provides the root-mean-squared-error (RMSE) and mean-absolute-percentage-error (MAPE) are two measures of prediction performance.

```
--------------------------------------------------
** Prediction Performance Criteria
--------------------------------------------------
OLS RMSE:   9690.885245291136
OLS MAPE:   4.710603802256941
GAM RMSE:   5889.925468140371
GAM MAPE:   2.721347026283239

Descriptive Statistics Table -  GAM Residuals
--------------------------------------------
Number of Cases     :              729
Minimum Value       :    -15825.172605
Maximum Value       :     60666.505854
Mean                :         0.000000
Standard Deviation  :      5893.969358
Skewness            :         2.035795
Kurtosis            :        16.675664
Cumulant (6th Order):      1390.475961
First Quartile      :     -3098.924384
Third Quartile      :      2777.251117

---------------------------------------------------
Hinich82 Bi-Spectrum Nonlinear Tests -  GAM Residuals
---------------------------------------------------
Gaussality  :       27.281   Fail
Linearity   :       11.003   Fail
```

It is evident that the GAM model exhibits a significantly better fit compared to the OLS model. Under the section of the output labeled as “Prediction Performance Criteria”, the root mean square error (RMSE) of the residuals for the GAM model is much smaller than the OLS regression model. This may be deceptive though since we are evaluating the in-sample fit of the data and it is quite possible that the GAM model is over-fitting the data.

To examine the out-of-sample predictive performance of the models, a percentage of the data should be retained for post-sample forecast comparison. To accomplish this, go back to the *Validation* tab. The last half of the data will be retained as the post-sample period.

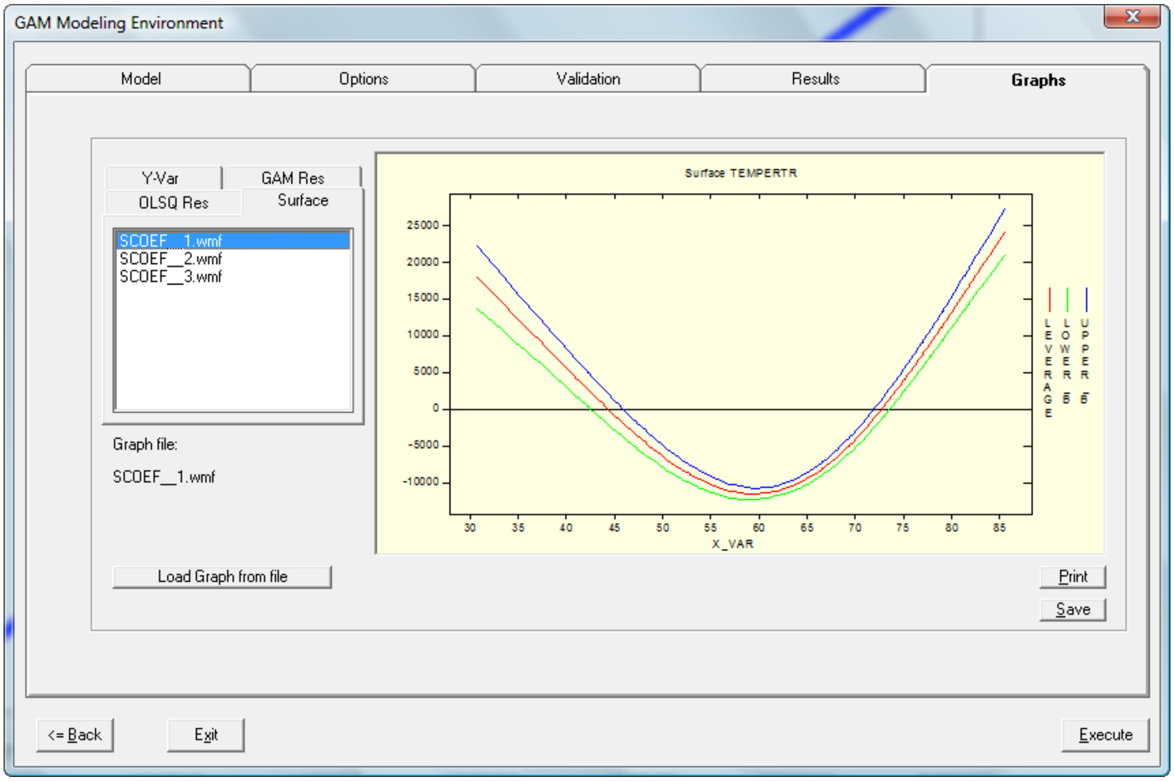


After changing the settings on the *Validation* tab, click on the *Execute* button to perform the analysis. After the analysis has completed, the performance criteria can once again be evaluated to determine if the model's true predictive power holds for the holdout period. Since the holdout sample is rather large in this example, we can be assured that outliers in the data will not severely skew our conclusions. The excerpt from the output is displayed below.

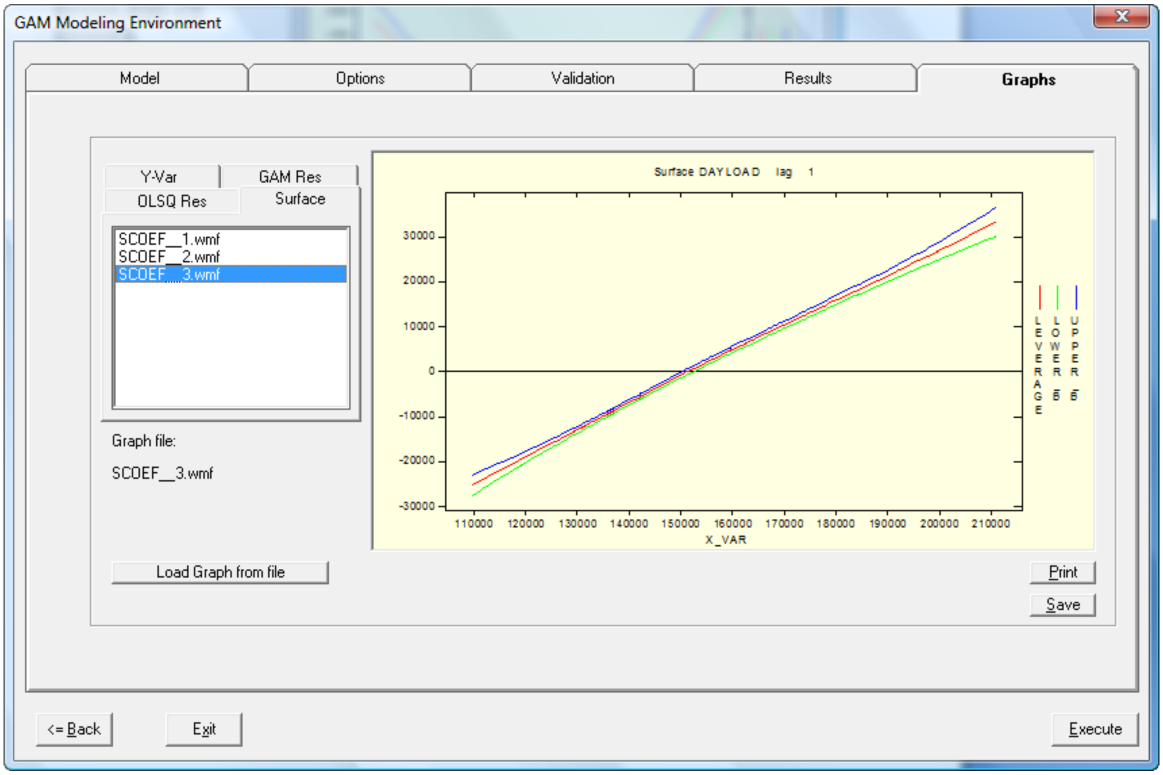
```
-----------------------------------------------------
** Prediction Performance Criteria
-----------------------------------------------------
OLS RMSE:   10236.36655180456
OLS MAPE:   4.949096693159521
GAM RMSE:   6476.742212420853
GAM MAPE:   2.977841294260566
```

We can conclude that the GAM modeling approach is a viable approach to predict daily electricity load based on changing levels of temperature, day of week, and observed load from the previous period. The model can be further refined to include dummy variables for seasons (Winter, Spring, Summer and Fall) and known events (holidays, power grid failures, etc.). An important tool to lend direction to possible model improvements is to examine residual graphs and diagnostics. Several diagnostic tables have been presented in the above output summary. By clicking on the *Graph* tab, the user can examine various graphs including a time plot of the dependent variable, actual versus predicted, time plots of residuals, residual autocorrelation function, Q-Stat charts, and various curvature charts. Since GAM models provide important information on the possible nonlinear relationships between the predictor variables and the dependent variable, we display two curvature charts next.

The first chart, SCOEF___1, shows the nonlinear curvature of the TEMPERTR variable with respect to DAYLOAD. Here, we can see a U-Shape relationship between temperature and electricity load indicating that when temperature increases (or decreases) from the minima (approximately 58 degrees), electricity load increases. We can conclude that this variable is definitely nonlinear and may be addressed using a quadratic term in a regression model.



The second chart, SCOEF___3, shows the nonlinear curvature of the DAYLOAD at lag order 1 with respect to DAYLOAD. Here, we can see that if nonlinearity exists it is slight and may be treated as linear without dire consequences. The approximated probability that the lagged DAYLOAD is nonlinear was 0.8117 which is not significant at the 95% level. Another intuitive method of determining whether a variable is nonlinear is to attempt to draw a line between the confidence bands on the chart. If the line does not intersect with a confidence band, a linear approximation is sufficient. In addition, if a constant function fits between the confidence bands, there is evidence that the variable is not a significant predictor.


GAM Modeling Environment
Model
Options
Validation
Results
Graphs
Y-Var
GAM Res
OLSQ Res
Surface
SCOEF__1.wmf
SCOEF__2.wmf
SCOEF__3.wmf
Graph file:
SCOEF__3.wmf
Load Graph from file
Surface DAYLOAD lag 1
X_VAR
LEVERAGE
LOWER_B
UPPER_B
Print
Save
<= Back
Exit
Execute

3.2 Modeling Production Cost Economies of Scale Using GAM

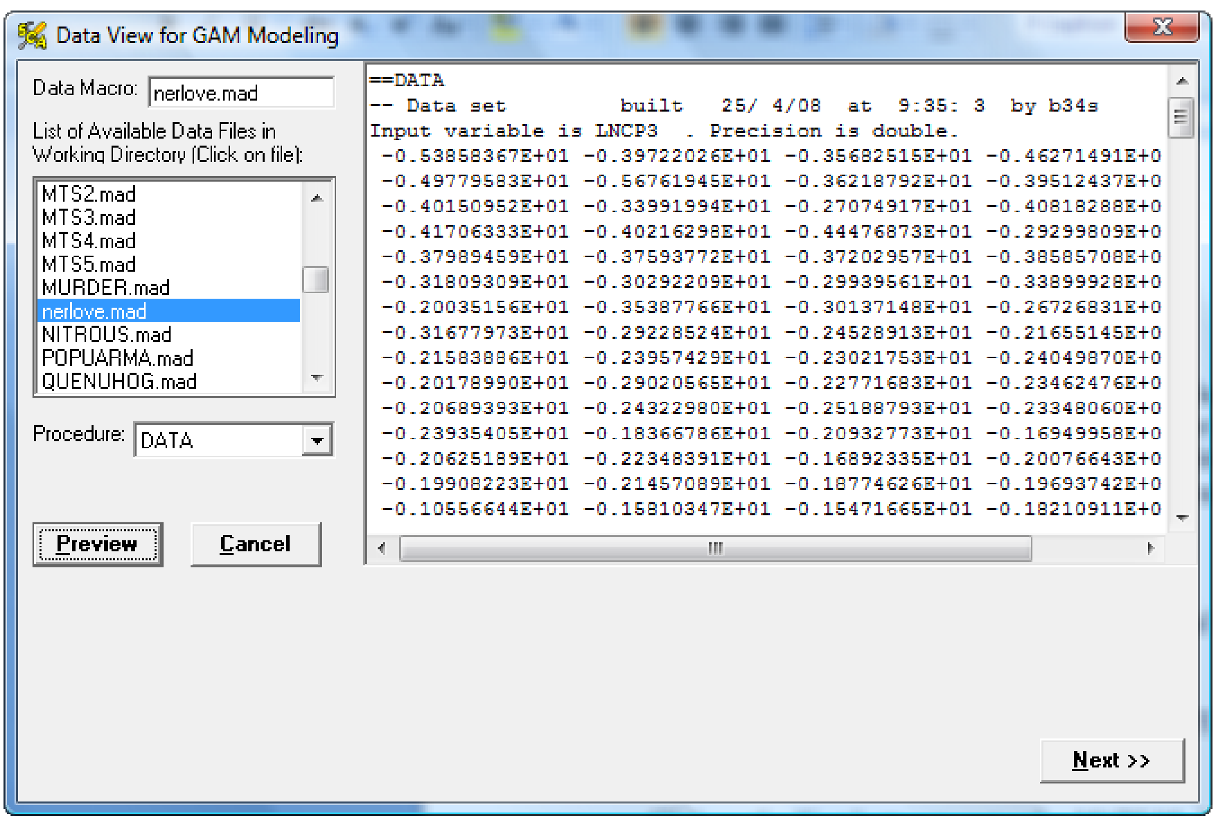
In this example, we use generalized additive models to study economies of scale for 145 U.S. companies generating electricity. The data is taken from Nerlove (1963) which is known to have nonlinearity. In that landmark study, Nerlove uses the log-linear cost function

$$\ln\left(\text{Costs} / \text{PF}\right) = \text{CNST} + \ln\left(\text{PL} / \text{PF}\right) + \ln\left(\text{PK} / \text{PF}\right) + \ln\left(\text{KWH}\right) + a_t$$

where,

COSTS → Total Production Costs in Millions of $
KWH → Kilowatt hours of output, in billions
PL → The wage rate per hour
PF → Price of fuels in cents per million BTU
PK → Rental price index of capital

The Nerlove data is located in the NERLOVE data macro file under the DATA procedure. This file is found under the TSData subdirectory under the SCA System installation folder (e.g., C:\SCA\TSData\). To work through this example, please set your SCA WorkBench working directory to C:\SCA\TSData. Next, launch the Generalized Additive Models environment and select the NERLOVE data macro file as illustrated below.



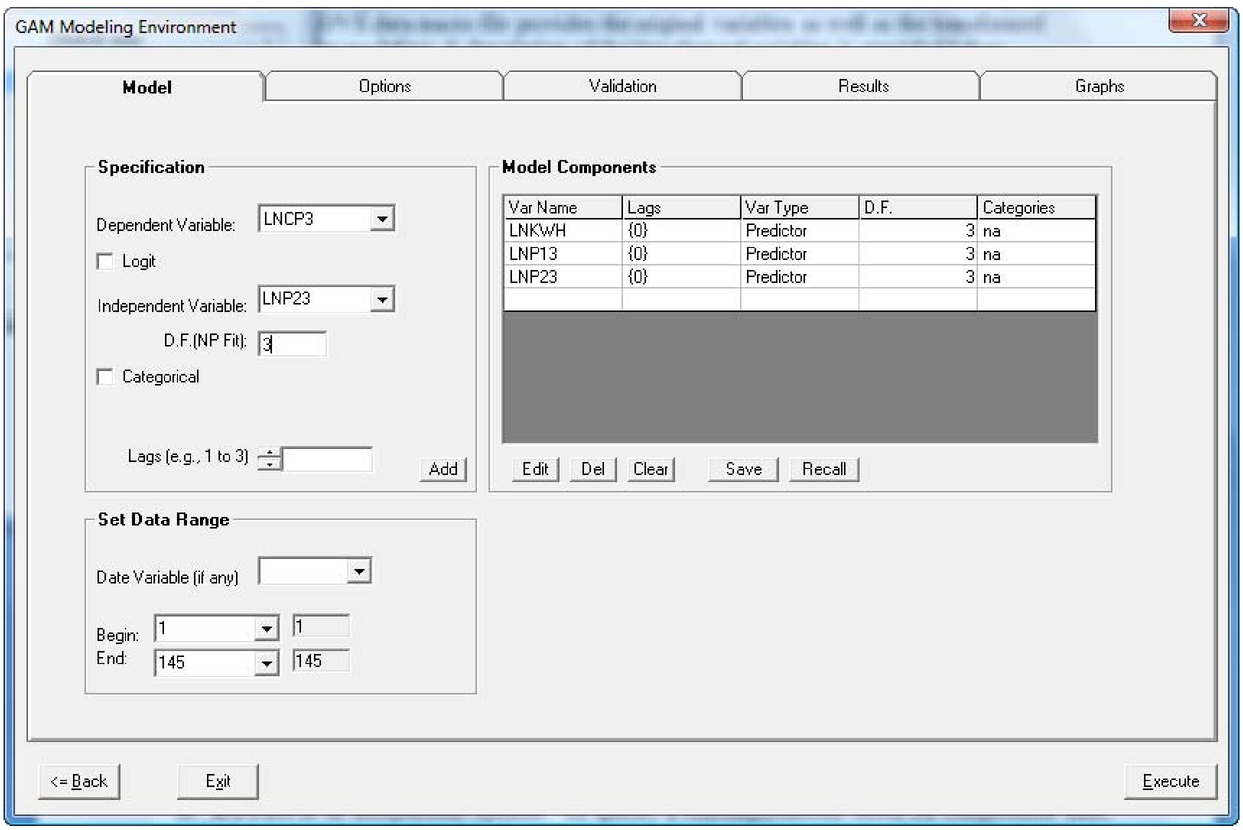
The NERLOVE data macro file provides the original variables as well as the transformed variables used for modeling. A description of the transformed variables is provided below.

LNCP3	=	ln(COSTS/PF)
LNP13	=	ln(PL/PF)
LNP23	=	ln(PK/PF)
LNKWH	=	ln(KWH)

Click on the *Next* button to enter the *GAM Modeling Environment*.

3.2.1 Specification of the GAM Model for the Nerlove Data

Once in the *GAM Modeling Environment*, click on the *Model* tab to specify the model components.

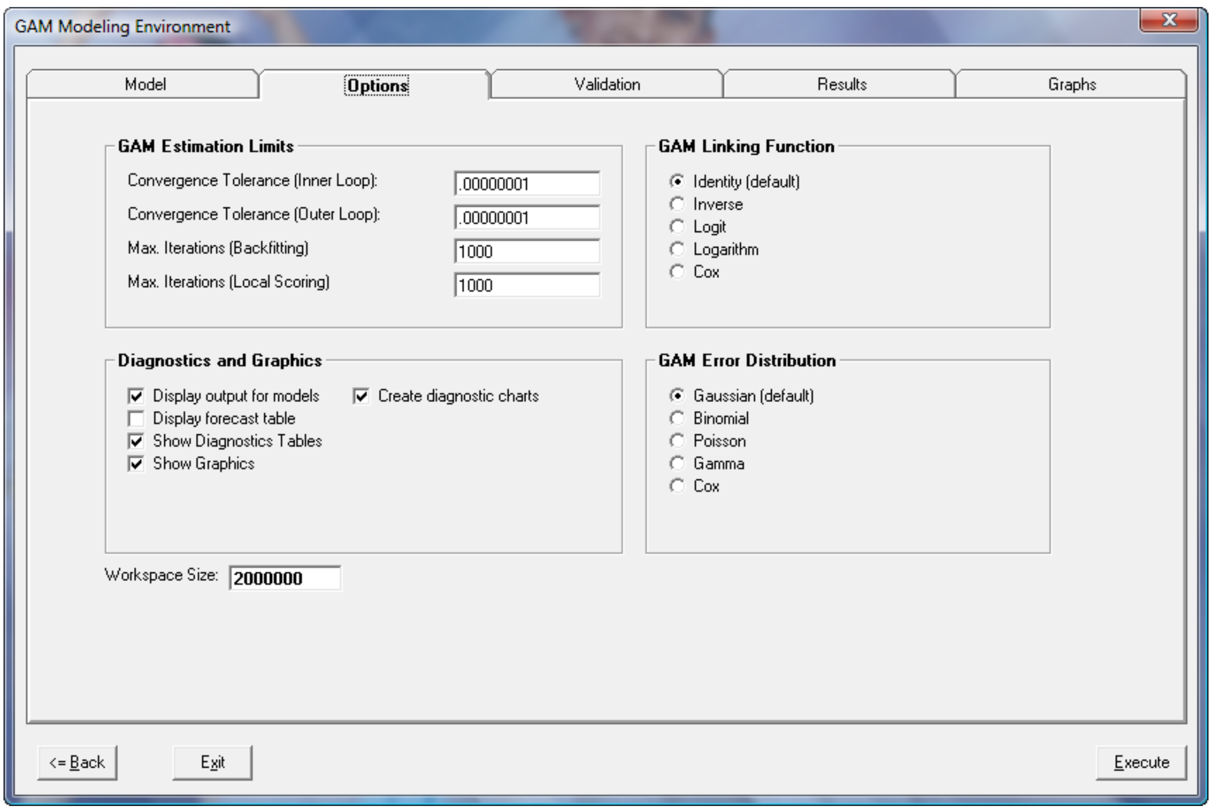


Use the drop down list box to select the dependent variable as **LNCP3**. Next, select **LNKWH** as an independent variable. To specify a contemporaneous effect for compression ratio, type “0” in the *Lags* text box provided. If you leave the *Lags* text box empty a contemporaneous effect will be assumed. Click on the *Add* button to include the component in the model displayed in the Model Components grid. To modify or delete an existing model component, place your cursor on the Model Components grid row, and click on the *Edit* or *Del* buttons.

In addition to compression ratio, specify **LNP13** and **LNP23** in the same manner. After specifying all GAM model components, the *Model* tab should like the one above.

3.2.2 GAM Model Options for Nerlove Data Example

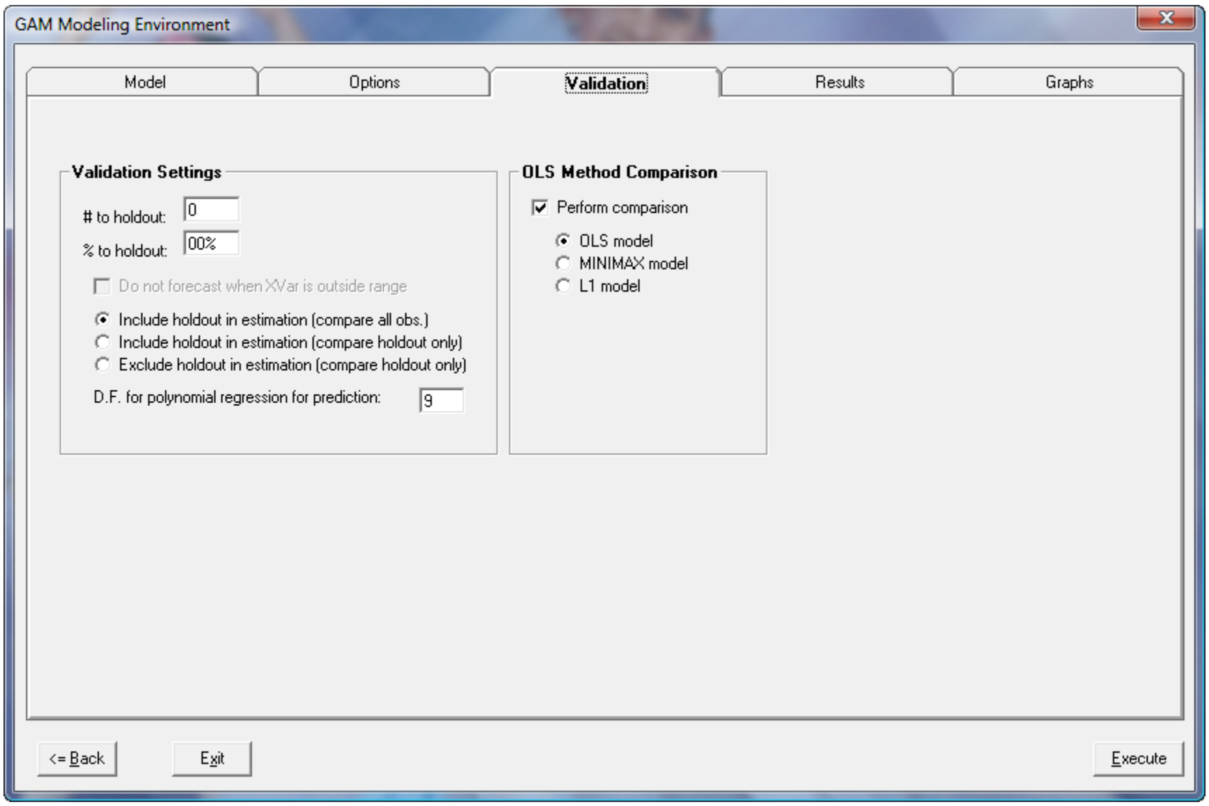
After specifying the GAM model, click on the *Options* tab to review the various options available. Here, the default options are accepted including the linking function and assumed error distribution.



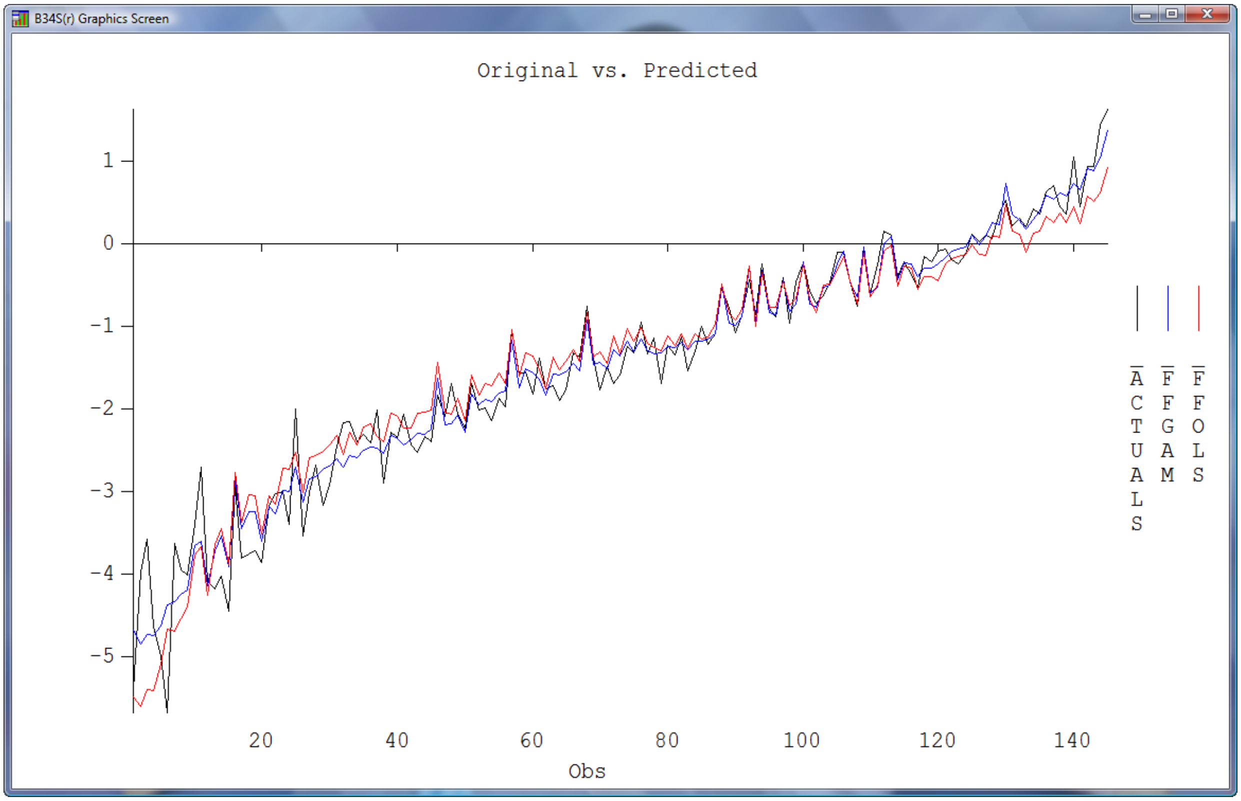
3.2.3 GAM Model Validation for the Nerlove Data Example

The *Validation* tab allows the user to evaluate the in-sample fit or out-of-sample predictive power of the GAM model compared to a linear regression model. As a first step, the overall in-sample fit of the GAM model will be examined and compared against a regression model that uses ordinary least squares (OLS) estimation.

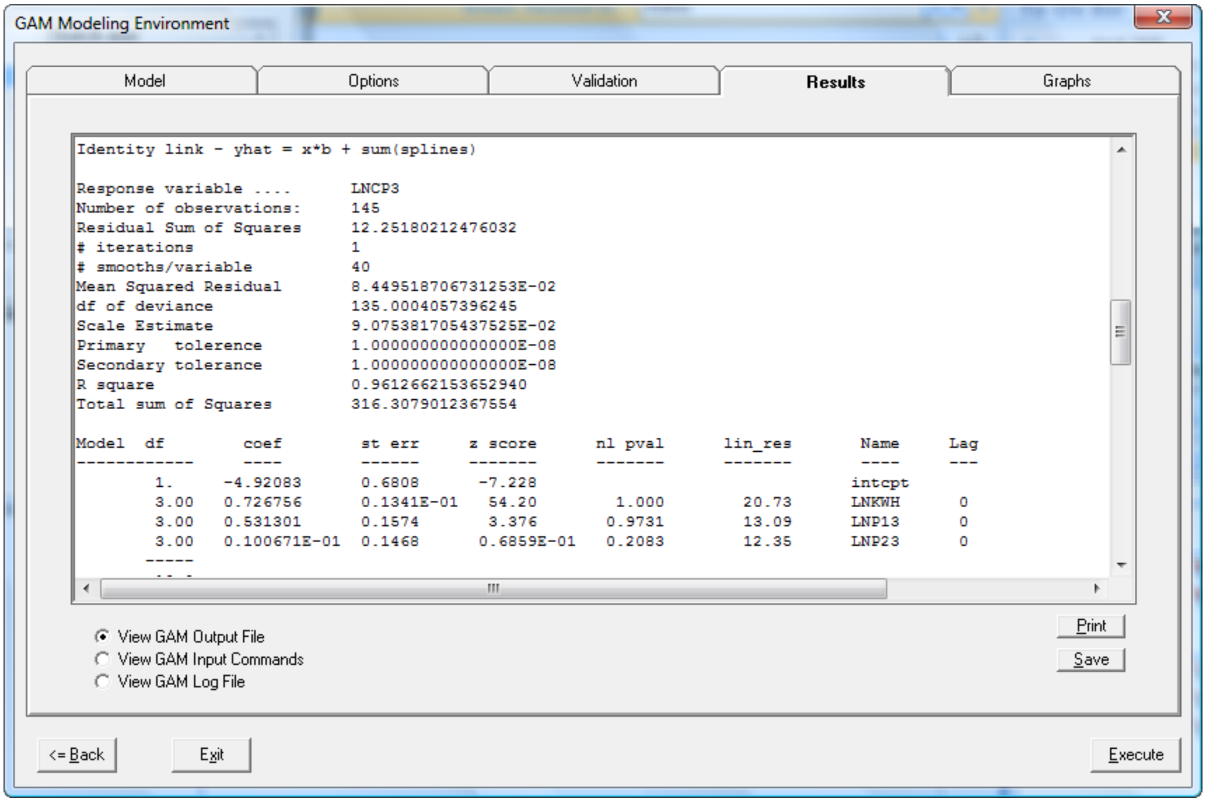
Select the radio button that compares all observations and set the number of observation in the holdout period to zero as shown below.



Next, click on the *Execute* button to run the analysis. The program script file will be automatically generated by SCA WorkBench and submitted to the SCAB34S engine for execution. When the GAM estimation completes, a graph showing the actual and fitted values for both OLS and GAM will be displayed.



Examining this graph there is a hint of a structural change in the data and/or that one or more of the predictor variables are nonlinearly related to the **LNCP3** variable. The OLS predicted values seem to be over-estimated near the beginning of the series and consistently under-estimated near the end of the series whereas the GAM predicted values seem to fit more uniformly across the entire data. To view the detailed output, first click on the graph to close it. The user will automatically be taken to the *Results* tab as shown below.



Use the scroll bar on the right-hand side of the text box to review the detailed output from the execution run. The sample output below has been reduced for brevity.

```
-------------------------------------------------------
** Analysis Performed on Variable: LNCP3
-------------------------------------------------------
```

The benchmark regression model estimated with OLS is shown below in the output. All variables except the LNP23 variable are significant. The adjusted R^2 also looks good indicating that we have a decent fitting model.

```
Final Model Estimation Summary -  OLS
Dependent Variable:               LNCP3
----------------------------------------------------------------
  # Variable    Lag     Coefficient      std.Err       t-value     Comments
----------------------------------------------------------------
  1 LNKWH        0            0.721        0.017        41.335
  2 LNP13        0            0.594        0.205         2.903
  3 LNP23        0           -0.008        0.191        -0.044
  4 CONSTANT     0           -4.686        0.885        -5.293

Number of observations:                145
Adjusted R-Square:                       0.930
Sum of Squared Residuals:               21.6
Schwartz Information Criteria:         160.536
```

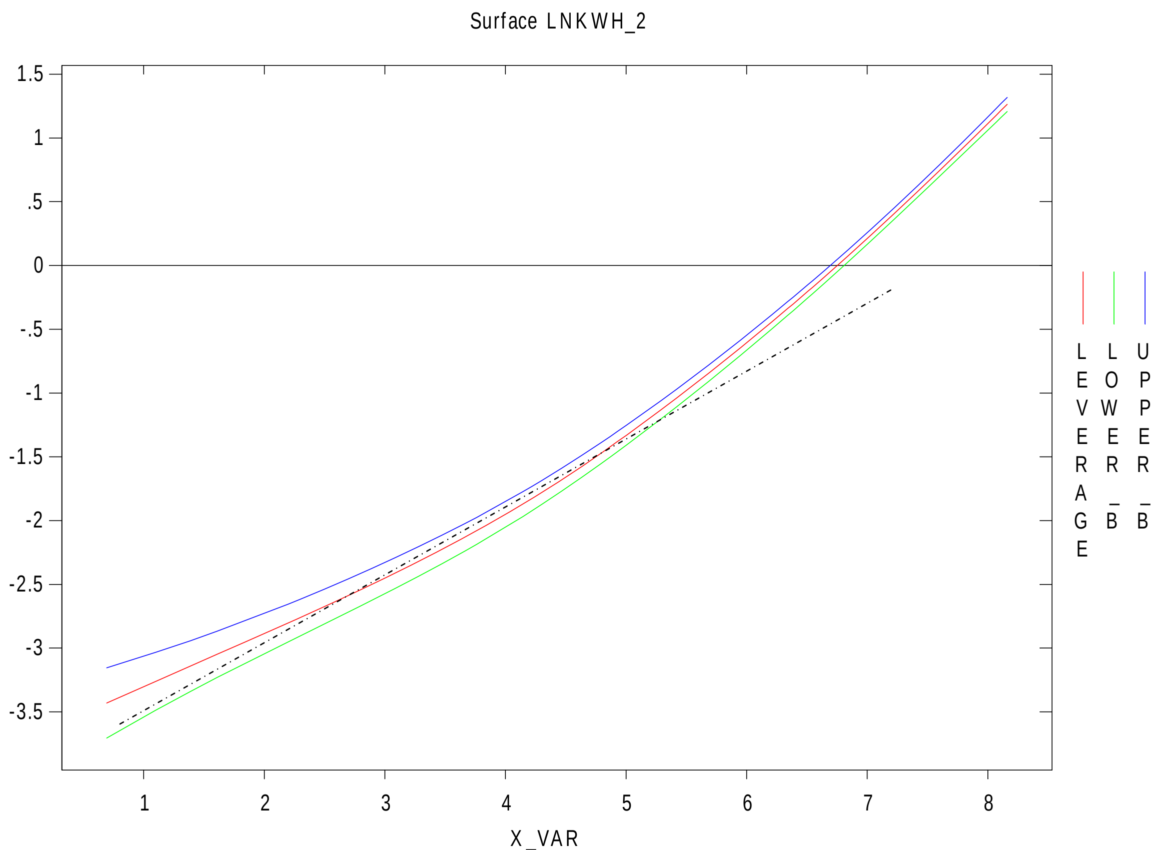
We now examine the GAM model summary and diagnostics to evaluate if our assumptions regarding linearity are founded, and whether improvements are possible.

```
Generalized Additive Models (GAM) Analysis

Gaussian additive model assumed
Identity link - yhat = x*b + sum(splines)

Response variable ....      LNCP3
Number of observations:     145
Residual Sum of Squares     12.25180212476032
# iterations                1
# smooths/variable          40
Mean Squared Residual       8.449518706731253E-02
df of deviance              135.0004057396245
Scale Estimate              9.075381705437525E-02
Primary   tolerence         1.000000000000000E-08
Secondary tolerance         1.000000000000000E-08
R square                    0.9612662153652940
Total sum of Squares        316.3079012367554

Model  df        coef        st err     z score      nl pval      lin_res       Name     Lag
------------     ----        ------     -------      -------      -------       ----     ---
        1.     -4.92083      0.6808      -7.228                                intcpt
        3.00   0.726756      0.1341E-01   54.20        1.000        20.73      LNKWH      0
        3.00   0.531301      0.1574       3.376        0.9731       13.09      LNP13      0
        3.00   0.100671E-01  0.1468      0.6859E-01   0.2083        12.35      LNP23      0
       -----
        10.0
```

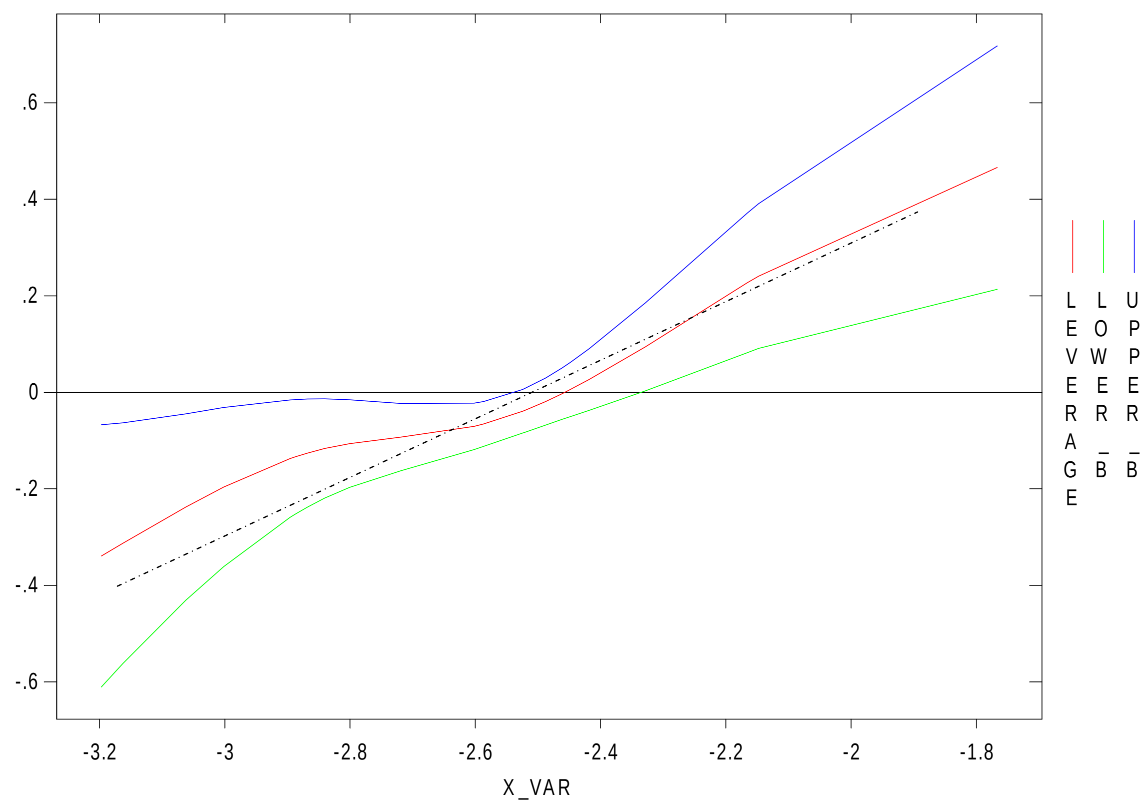
The GAM model reveals that LNKWH is nonlinear (nl pval=1), LNP13 may also be nonlinear(nl pval=.9731), and LNP23 remains insignificant (z-score=.2083). Furthermore, if we examine the prediction performance table below, significant improvement is once again possible through GAM.

```
-----------------------------------------------------
** Prediction Performance Criteria
-----------------------------------------------------
OLS RMSE:   0.3862880004134601
OLS MAPE:   35.63870474450533
GAM RMSE:   0.2906805584611956
GAM MAPE:   22.65559437168299

Descriptive Statistics Table -  GAM Residuals
---------------------------------------------
Number of Cases     :             145
Minimum Value       :       -1.158634
Maximum Value       :        1.304020
Mean                :        0.000000
Standard Deviation  :        0.291688
Skewness            :       -0.138578
Kurtosis            :        4.182627
Cumulant (6th Order):       21.031025
First Quartile      :       -0.141374
Third Quartile      :        0.145510


Descriptive Statistics Table -  OLS Residuals
---------------------------------------------
Number of Cases     :             145
Minimum Value       :       -1.818976
Maximum Value       :        1.012116
Mean                :        0.000000
Standard Deviation  :        0.387627
Skewness            :       -1.238098
Kurtosis            :        4.511577
Cumulant (6th Order):       22.337364
First Quartile      :       -0.160462
Third Quartile      :        0.217891
```

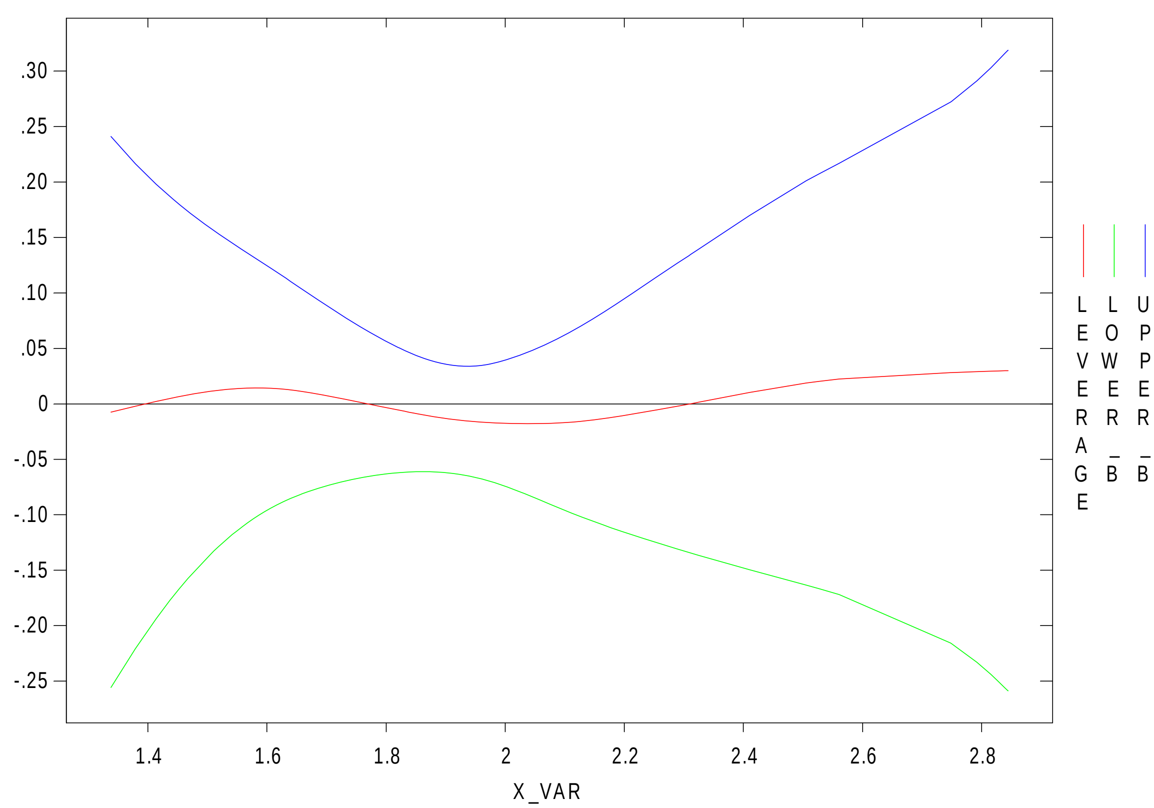
The curvature of the transformations applied the predictor variables are displayed below. A dotted line is super-imposed on the graphs in an attempt to draw the line inside the upper and lower confidence bands as a empirical method of determining the degree of nonlinearity associated with the variable. Here, we conclude that the LNKWH variable is nonlinear, since we are unable to draw a line within the confidence bands. We cannot be as confident regarding the LNP13 variable based on this empirical test but there is still a reasonable chance that it is nonlinear. The user can restrict the variable as linear in the GAM model and re-estimate to evaluate the improvement gained. Also, note that a constant function fits between the confidence bands for the LNP23 variable which provides confirming evidence that this variable is statistically insignificant in the model. These diagnostic plots are also accessible from the Graphs tab of the GAM Modeling Environment for all predictor variables.



Surface LNP13_2



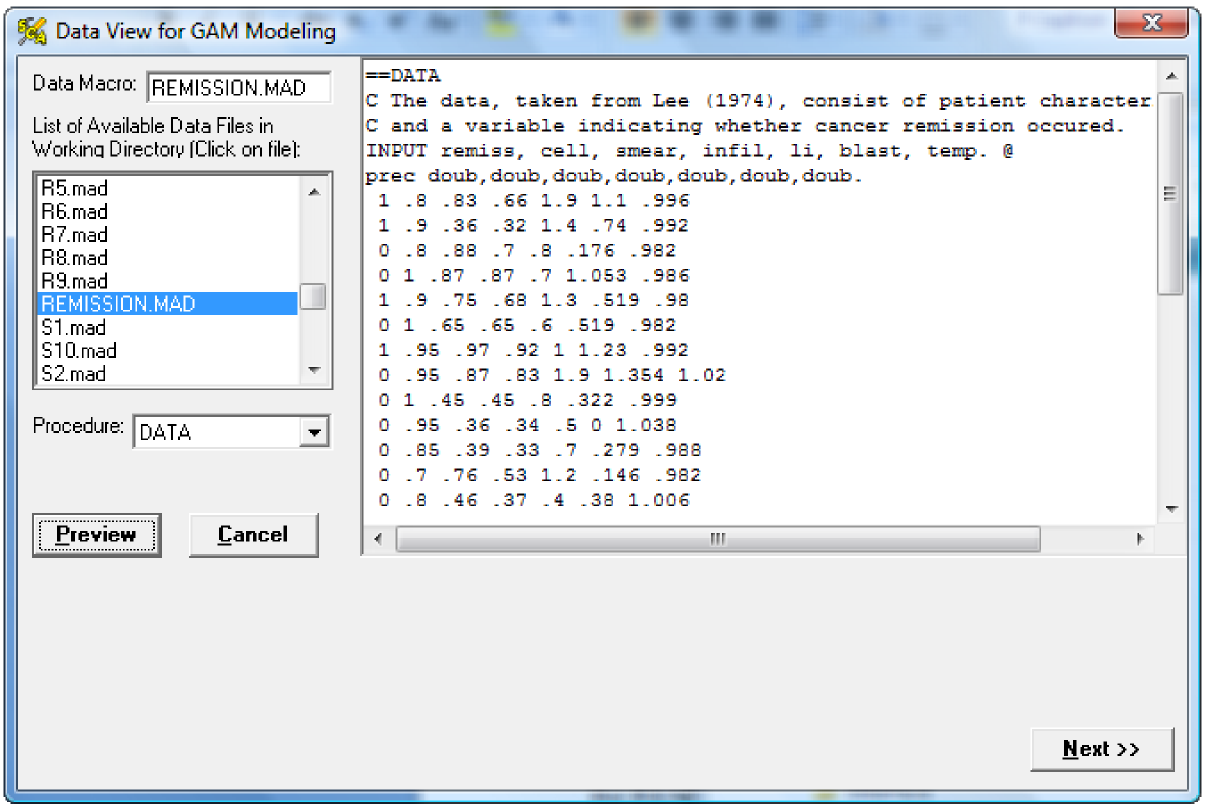
Surface LNP23_2



3.3 Modeling Cancer Remission Using Logistic GAM

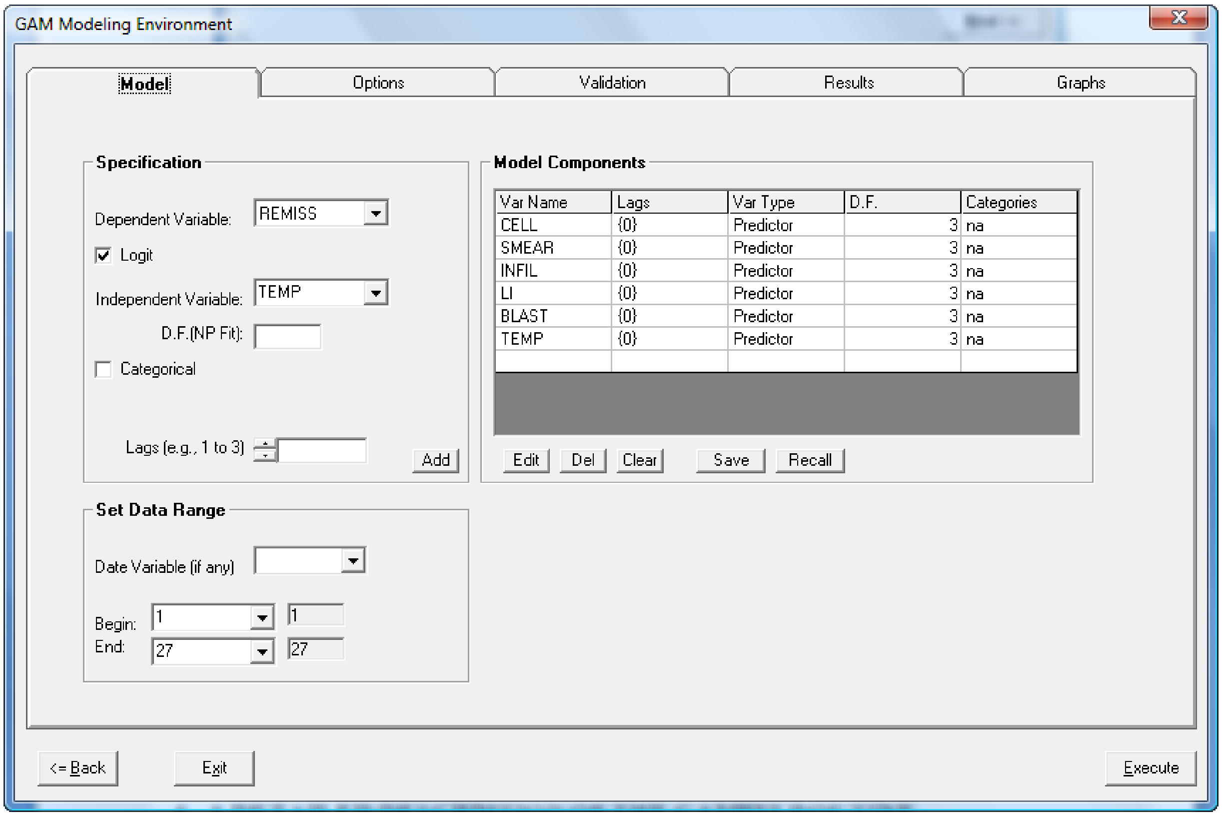
This example uses a Logistic GAM to model remission occurrences in cancer patients. The data is taken from Lee (1974). The data consists of the remission indicator variable, REMISS, and six other risk factors (CELL, SMEAR, INFIL, LI, BLAST, and TEMP) considered to have an effect on cancer remission.

The Remission data is located in the REMISSION data macro file under the DATA procedure. This file is found under the TSData subdirectory under the SCA System installation folder (e.g., C:\SCA\TSData\). To work through this example, please set your SCA WorkBench working directory to C:\SCA\TSData. Next, launch the Generalized Additive Models environment and select the REMISSION data macro file as illustrated below.



3.3.1 Specification of the GAM Model for Cancer Remission

Enter the *GAM Modeling Environment* and specify the model components in the *Model* tab.

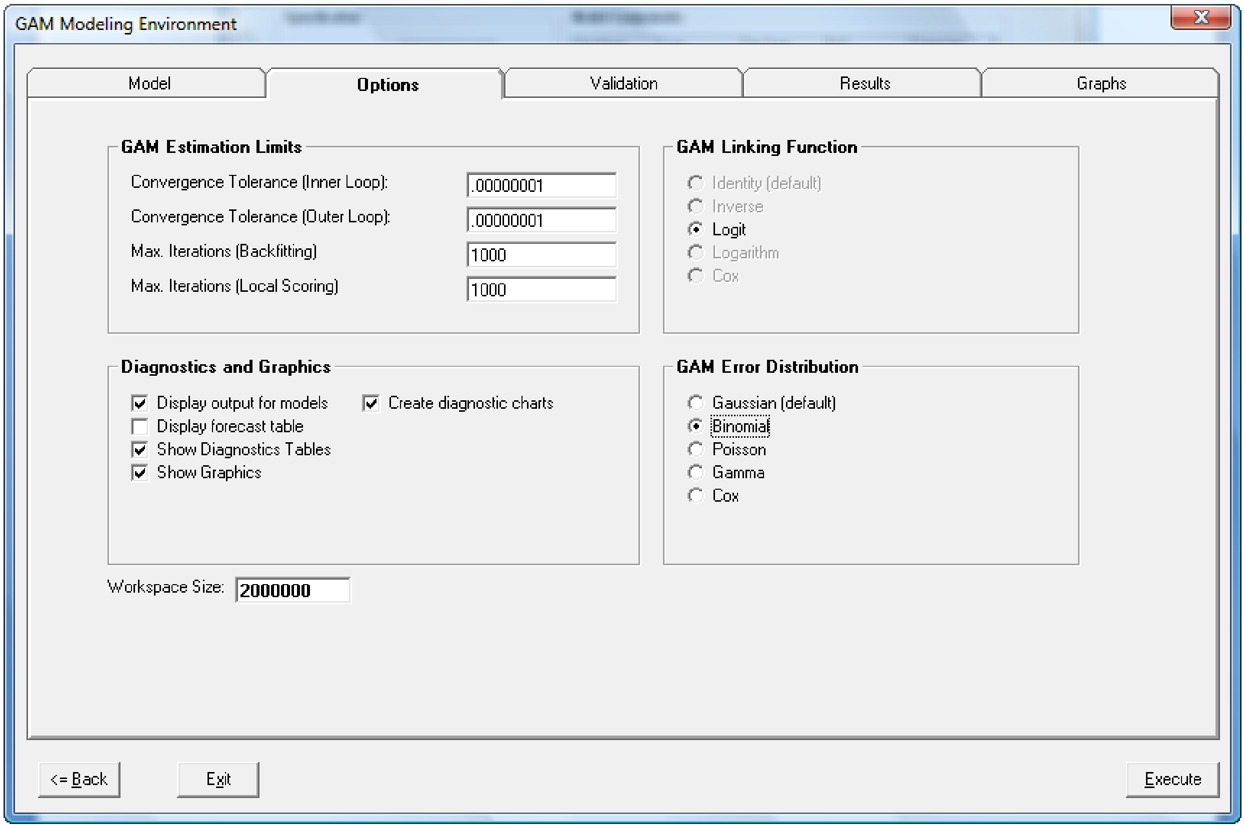


Use the drop down list to select the dependent variable as **REMISS**. Next, add the predictor variables to the model by selecting them one at a time from the *Independent Variable* drop down list and clicking on the *Add* button. To modify of delete a component from the model, click on the variable name on the *Model Components* grid, and then on the *Edit (or Delete)* button. After specifying the model components, the *Model* tab should look like the example above.

3.3.2 GAM Model Options for the Cancer Remission Example

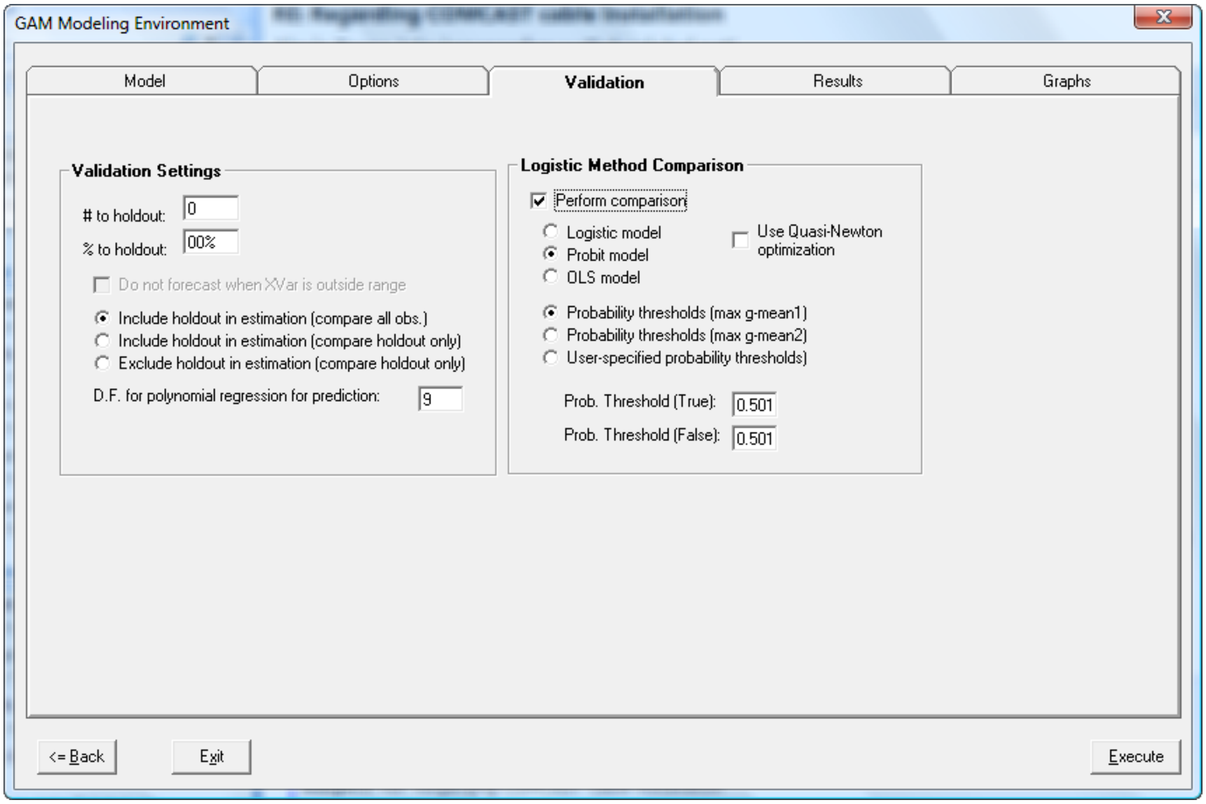
After specifying the GAM model, go to the Options tab. The linking function should already be set to Logit. If not, return to the *Model* tab, and put a check mark in the *Logit* box under the *Dependent* variable drop down list, and then return to the Options tab. We know have several options regarding the assumed error distribution. Select the *Poisson* option for the *GAM Error Distribution*. An alternative error distribution can also be entertained, if desired.

Lastly, put a check in the *Create diagnostic charts* box under the *Diagnostics and Graphics* frame. This will generate the curvature charts to evaluate the nonlinear attributes of the predictor variables in relation to the dependent variable. Note, a comparison model such as OLS, Logit, or Probit must also be specified on the validation tab for the charts to be created.

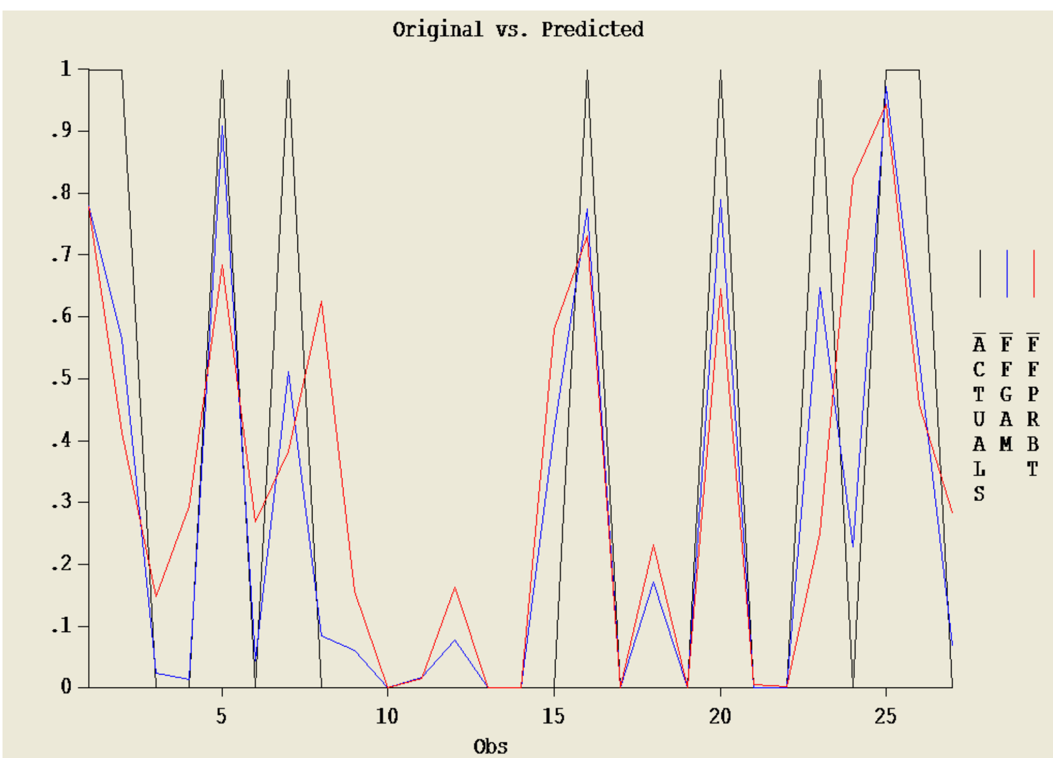


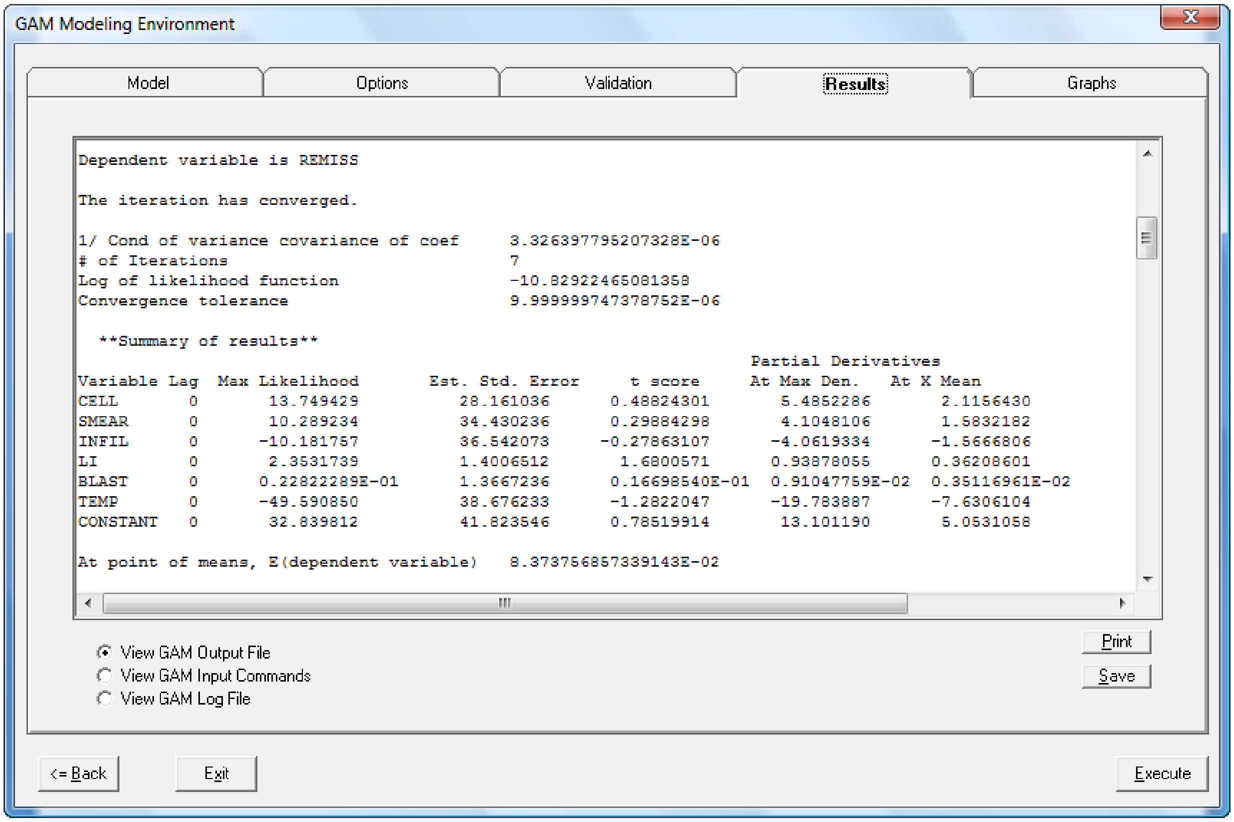
3.3.3 GAM Model Validation for the Cancer Remission Example

The *Validation* tab allows the user to evaluate the in-sample and out-of-sample predictive performance of a GAM model. A linear comparison model can also selected as a benchmark for performance gains. Since we only have 27 observations, we will compare the in-sample fit of the models without holding back any observations. Furthermore, we will specify a *Probit model* to be used for comparison purposes. Be sure of put a check mark in the *Perform Comparison* box to activate the model comparison feature and generate the diagnostic charts.



Clicking on the *Next* button will run the analysis. The program script will automatically be generated by SCA WorkBench and executed in the SCAB34S engine. When the execution has completed, a graph of actual and fitted values is displayed. Click anywhere on the graph to enter the *Results* tab.





Use the scroll bar on the right-hand side of the text box to view the detailed output. The output is shown below along with comments regarding findings. We begin with the Probit model summary that is used as our benchmark in this example.

```
-------------------------------------------------------
** Analysis Performed on Variable: REMISS
-------------------------------------------------------

Multivariate Probit Analysis (December 2004).

1/ Cond of variance covariance of coef     3.326397795207328E-06
# of Iterations                            7
Log of likelihood function                 -10.82922465081358
Convergence tolerance                      9.999999747378752E-06

                                                                    Partial Derivatives
Variable Lag  Max Likelihood      Est. Std. Error    t score       At Max Den.   At X Mean
CELL       0       13.749429         28.161036     0.48824301      5.4852286       2.1156430
SMEAR      0       10.289234         34.430236     0.29884298      4.1048106       1.5832182
INFIL      0      -10.181757         36.542073    -0.27863107     -4.0619334      -1.5666806
LI         0       2.3531739         1.4006512      1.6800571     0.93878055      0.36208601
BLAST      0    0.22822289E-01       1.3667236     0.16698540E-01  0.91047759E-02  0.35116961E-02
TEMP       0      -49.590850         38.676233     -1.2822047     -19.783887      -7.6306104
CONSTANT   0       32.839812         41.823546     0.78519914      13.101190       5.0531058

At point of means, E(dependent variable)   8.373756857339143E-02

# of observations                          27
# limits(=0)                               18
# nonlimits(=1)                            9
(-2.0) times the log likelihood ratio      12.71331578629275
Distributed as Chi squared with DF         6
Significance of Chi squared statistic      0.9521787372759888
```

Final Model Estimation Summary - PROBIT
Method => Quasi-Newton (CMAXF2)
Functional Value= 12.713

#	Variable	Lag	Coefficient	std.Err	t-value
1	CELL	0	13.7	28.2	0.488
2	SMEAR	0	10.3	34.4	0.299
3	INFIL	0	-10.2	36.5	-0.279
4	LI	0	2.35	1.40	1.68
5	BLAST	0	0.228E-01	1.37	0.167E-01
6	TEMP	0	-49.6	38.7	-1.28
7	CONSTANT	0	32.8	41.8	0.785

When the dependent variable is specified as Logistic, a Confusion Matrix and Lift-Gain table is generated. The GMEAN1 was used to compute the probability cut-off of .325 and based on this setting, the Probit model has a precision rate of 72.7% and was successful in classifying a remission case correctly 88.9% of the time.

PROBIT True/False Probability CutOff Range

Cut	True-Pos	True-Neg	False-Pos	False-Neg	Accuracy	Precision	GMean1	GMean2
0.050	1.000	0.444	0.556	0.000	0.630	0.474	0.688	0.667
0.100	1.000	0.444	0.556	0.000	0.630	0.474	0.688	0.667
0.150	1.000	0.500	0.500	0.000	0.667	0.500	0.707	0.707
0.200	1.000	0.611	0.389	0.000	0.741	0.562	0.750	0.782
0.250	1.000	0.667	0.333	0.000	0.778	0.600	0.775	0.816
0.300	0.889	0.833	0.167	0.111	0.852	0.727	0.804	0.861
0.350	0.889	0.833	0.167	0.111	0.852	0.727	0.804	0.861
0.400	0.778	0.833	0.167	0.222	0.815	0.700	0.738	0.805
0.450	0.667	0.833	0.167	0.333	0.778	0.667	0.667	0.745
0.500	0.556	0.833	0.167	0.444	0.741	0.625	0.589	0.680
0.550	0.556	0.833	0.167	0.444	0.741	0.625	0.589	0.680
0.600	0.556	0.889	0.111	0.444	0.778	0.714	0.630	0.703
0.650	0.444	0.944	0.056	0.556	0.778	0.800	0.596	0.648
0.700	0.333	0.944	0.056	0.667	0.741	0.750	0.500	0.561
0.750	0.222	0.944	0.056	0.778	0.704	0.667	0.385	0.458
0.800	0.111	0.944	0.056	0.889	0.667	0.500	0.236	0.324
0.850	0.111	1.000	0.000	0.889	0.704	1.000	0.333	0.333
0.900	0.111	1.000	0.000	0.889	0.704	1.000	0.333	0.333
0.950	0.000	1.000	0.000	1.000	0.667	0.000	0.000	0.000
1.000	0.000	1.000	0.000	1.000	0.667	0.000	0.000	0.000

Confusion Matrix
Performance Evaluation for PROBIT

	Predicted		
	Negative	Positive	Unclassified
Negative	15	3	0
Positive	1	8	0

Total Number Cases	:	27
Total Classified Cases	:	27
Unclassified Positive Cases	:	0
Unclassified Negative Cases	:	0
Cut-off for True (>)	:	0.325
Cut-off for False (<)	:	0.325
Accuracy Rate	:	0.852
True Positive Rate	:	0.889
False Positive Rate	:	0.167
True Negative Rate	:	0.833
False Negative Rate	:	0.111
Precision Rate	:	0.727
g-mean1	:	0.804
g-mean2	:	0.861

PROBIT Lift-Gain Table

Decile	#Obs in Decile	#Pos in Decile	%Pos in Decile	Pctg of Total Pos	Cum. #Obs	Cum. #Pos	Cum. %Pos	Cum. Gain	K_S Spread	Lift Index	Gain over Random
1	3	2	66.7%	22.2%	3	2	7.4%	22.2%	16.7%	222	55.0%
2	2	2	100.0%	22.2%	5	4	14.8%	44.4%	38.9%	222	55.0%
3	3	1	33.3%	11.1%	8	5	18.5%	55.6%	38.9%	185	46.0%
4	3	3	100.0%	33.3%	11	8	29.6%	88.9%	72.2%	222	55.0%
5	3	0	0.0%	0.0%	14	8	29.6%	88.9%	55.6%	177	43.8%
6	2	1	50.0%	11.1%	16	9	33.3%	100.0%	61.1%	166	40.0%
7	3	0	0.0%	0.0%	19	9	33.3%	100.0%	44.4%	142	30.0%
8	3	0	0.0%	0.0%	22	9	33.3%	100.0%	27.8%	125	20.0%
9	2	0	0.0%	0.0%	24	9	33.3%	100.0%	16.7%	111	10.0%
10	3	0	0.0%	0.0%	27	9	33.3%	100.0%	0.0%	100	0.0%

The GAM model summary is shown next. We see that several of the predictor variables are on the borderline of being identified as nonlinear. Caution is required due to the number of observations since they are few. However, the BLAST predictor has the highest probability of being nonlinear (nl pval = .9405).

```
Generalized Additive Models (GAM) Analysis
Reference: Generalized Additive Models by Hastie and Tibshirani. Chapman (1990)
Model estimated with GPL code obtained from R.

Poisson additive model assumed
Logit link - yhat = exp(z)/(1.0+exp(z)).
where:       z    = x*b + sum(splines)

Response variable ....      REMISS
Number of observations:     27
Residual Sum of Squares     3.724899289693494
# iterations                544
# smooths/variable          16751
Mean Squared Residual       0.1379592329516109
df of deviance              8.648902240870296
Scale Estimate              0.4306788521775078
Primary   tolerence         1.000000000000000E-08
Secondary tolerance         1.000000000000000E-08
R square                    0.8116361417381410
Total sum of Squares        19.77502119602597

Model  df        coef        st err     z score      nl pval      lin_res       Name      Lag
------------     ----        ------     -------      -------      -------       ----      ---
        1.      174.955       30.90       5.662                                 intcpt
        2.77   -1.62949       13.25      -.1230       0.8719        6.022       CELL        0
        2.88   -21.2814       25.81      -.8247       0.8648        6.043       SMEAR       0
        2.91    27.5458       28.68      0.9605       0.5587        4.846       INFIL       0
        2.76    6.56756       2.500       2.627       0.8664        5.971       LI          0
        3.15   0.469946       2.546      0.1846       0.9405        7.032       BLAST       0
        2.88   -187.488       29.41      -6.376       0.8657        6.046       TEMP        0
       -----
        18.4

Residual Sum of Squares for Original Data  1.207089552211462
```

The Confusion Matrix and Lift-Gain table associated with the GAM model are presented below. The GAM model has correctly classified all remission cases. It has also correctly identified all cases not in remission. We need to be cognoscente of the possibility the model is over-fitted.

GAM True/False Probability CutOff Range

Cut	True-Pos	True-Neg	False-Pos	False-Neg	Accuracy	Precision	GMean1	GMean2
0.050	1.000	0.444	0.556	0.000	0.630	0.474	0.688	0.667
0.100	1.000	0.556	0.444	0.000	0.704	0.529	0.728	0.745
0.150	1.000	0.722	0.278	0.000	0.815	0.643	0.802	0.850
0.200	1.000	0.833	0.167	0.000	0.889	0.750	0.866	0.913
0.250	1.000	0.889	0.111	0.000	0.926	0.818	0.905	0.943
0.300	1.000	0.944	0.056	0.000	0.963	0.900	0.949	0.972
0.350	1.000	0.944	0.056	0.000	0.963	0.900	0.949	0.972
0.400	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000
0.450	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000
0.500	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000
0.550	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000
0.600	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000
0.650	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000
0.700	0.778	1.000	0.000	0.222	0.926	1.000	0.882	0.882
0.750	0.667	1.000	0.000	0.333	0.889	1.000	0.816	0.816
0.800	0.556	1.000	0.000	0.444	0.852	1.000	0.745	0.745
0.850	0.556	1.000	0.000	0.444	0.852	1.000	0.745	0.745
0.900	0.222	1.000	0.000	0.778	0.741	1.000	0.471	0.471
0.950	0.111	1.000	0.000	0.889	0.704	1.000	0.333	0.333
1.000	0.000	1.000	0.000	1.000	0.667	0.000	0.000	0.000

Confusion Matrix
Performance Evaluation for GAM

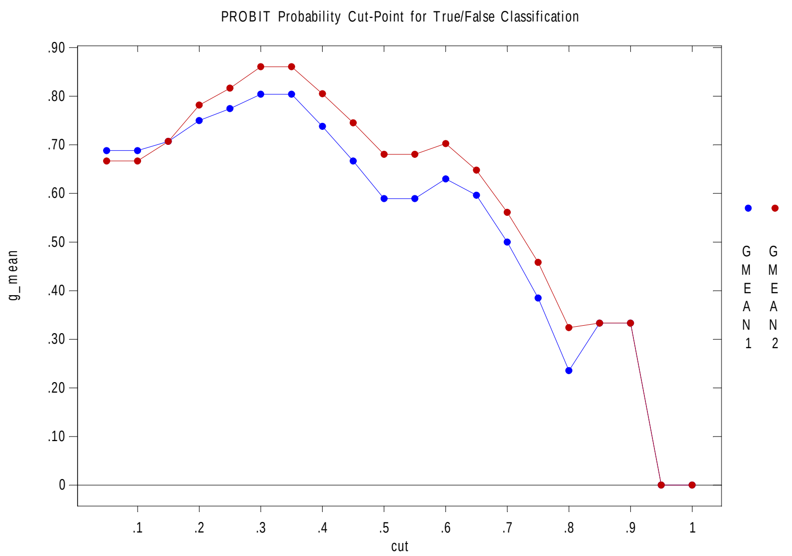
	Negative	Predicted Positive	Unclassified
Negative	18	0	0
Positive	0	9	0

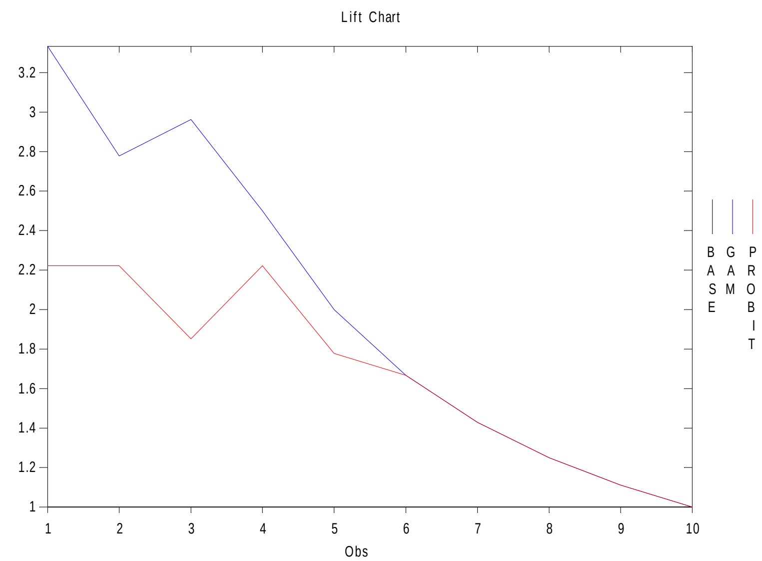
Total Number Cases	:	27
Total Classified Cases	:	27
Unclassified Positive Cases	:	0
Unclassified Negative Cases	:	0
Cut-off for True (>)	:	0.525
Cut-off for False (<)	:	0.525
Accuracy Rate	:	1.000
True Positive Rate	:	1.000
False Positive Rate	:	0.000
True Negative Rate	:	1.000
False Negative Rate	:	0.000
Precision Rate	:	1.000
g-mean1	:	1.000
g-mean2	:	1.000

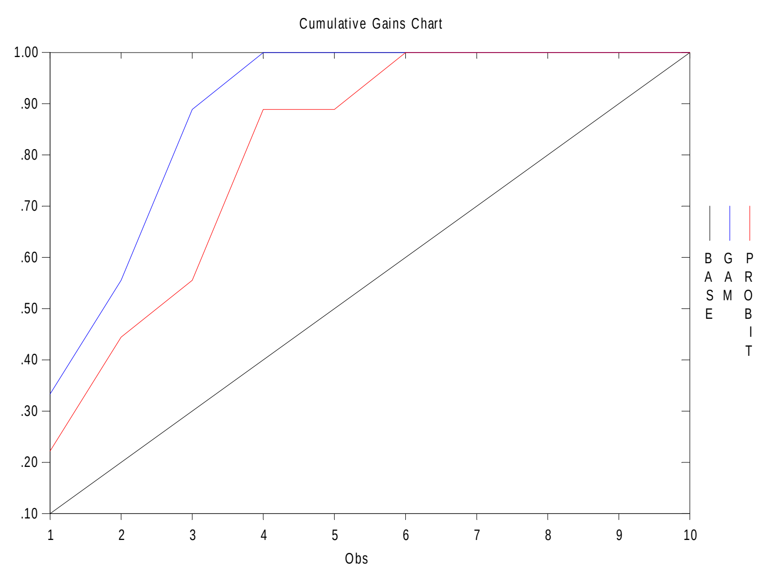
GAM Lift-Gain Table

Decile	#Obs in Decile	#Pos in Decile	%Pos in Decile	Pctg of Total Pos	Cum. #Obs	Cum. #Pos	Cum. %Pos	Cum. Gain	K_S Spread	Lift Index	Gain over Random
1	3	3	100.0%	33.3%	3	3	11.1%	33.3%	33.3%	333	70.0%
2	2	2	100.0%	22.2%	5	5	18.5%	55.6%	55.6%	277	64.0%
3	3	3	100.0%	33.3%	8	8	29.6%	88.9%	88.9%	296	66.2%
4	3	1	33.3%	11.1%	11	9	33.3%	100.0%	88.9%	250	60.0%
5	3	0	0.0%	0.0%	14	9	33.3%	100.0%	72.2%	200	50.0%
6	2	0	0.0%	0.0%	16	9	33.3%	100.0%	61.1%	166	40.0%
7	3	0	0.0%	0.0%	19	9	33.3%	100.0%	44.4%	142	30.0%
8	3	0	0.0%	0.0%	22	9	33.3%	100.0%	27.8%	125	20.0%
9	2	0	0.0%	0.0%	24	9	33.3%	100.0%	16.7%	111	10.0%
10	3	0	0.0%	0.0%	27	9	33.3%	100.0%	0.0%	100	0.0%

In addition to the Lift-Gain and cut-off probability evaluation tables, the corresponding Lift, Cumulative Gains , and G-Mean evaluation charts are accessible from the Graphs tab. The charts show the predictive power of the GAM model for this example.


PROBIT Probability Cut-Point for True/False Classification
g_mean
cut
GMEAN1
GMEAN2


Lift Chart
Obs
BASE
GAM
PROBIT


Cumulative Gains Chart
Obs
BASE
GAM
PROBIT

4 A DETAILED DESCRIPTION OF THE GAMFIT ROUTINE

The SCAB34S SPLINES product integrates the GPL subroutines for Generalized Additive Models (GAM) estimation developed by Hastie and Tribshirani. A detailed description of the **GAMFIT** subroutine is listed next and should be studied prior to any model building.

Usage:

```
CALL GAMFIT(Y X1[vtype,df] X2[vtype,df] :OPTIONS);
```

Where vtype should be substituted with a keyword (predictor or function) and df should be substituted with the integer value for the degrees of freedom (df=1 implies linear).

Controls estimation of GAM models using GPL code developed by Hastie and Tribshirani. Lags can be entered as

```
X[vtype,df]{1} or  X[vtype,df]{1 to 20}
```

Required subroutine arguments:

Arguments	Intent	Comments
Y	*Input*	Dependent variable in the GAM model. The dependent variable may be specified as a random variable or a binary variable.
X[vtype,df]	*Input*	One or more regressors in the GAM model. The regressor-variables may be specified as predictor (vtype=predictor) or categorical (vtype=function) with the associated degrees of freedom (df=1 is linear assumption, df=3 is quadratic assumption). If a lagged regressor variable is included in the model, the lags must be specified within braces as shown in the usage section above.

Optional keywords and associated arguments:

Keyword	# Args	Type(s)	Intent	Comments
:LINK	1	*Character keyword*	*input*	Specifies the nonlinear link function between the mean of the dependent variable and the additive predictor variable. The link functions supported are IDENT → Identity (default) INVER → Inverse LOGIT → Logistic LOGAR → Logarithmic COX → Cox

:DIST	1	*Character keyword*	*input*	Specifies the assumed error response probability distribution. The distribution types supported are GAUSS → Gaussian (default) BINOM → Binomial POISS → Poisson GAMMA → Gamma COX → Cox
:PRINT	1	*Logical*	*input*	Print model header and minimal output for the GAM model
:NOINT	1	*Logical*	*input*	Exclude an intercept term in the estimated GAM model
:INFO	1	*Logical*	*input*	Displays the estimation summary at each iteration
:SAMPLE	1	*Real number array*	*input*	Specifies a mask real*8 variable where mask= 0.0 drops that observation. Unless the mask is the number of observations after any lags, an error message will be generated. The sample variable must be used with great caution when there are lags. A much better choice is the :holdout option when lagged regression variables are used.
:HOLDOUT	1	*Integer scalar*	*input*	Specifies the number of observations to hold out from the back of the time series for evaluating prediction power of the model. The :holdout option can not be used when the :sample option is employed.
:PUNCH_RES	1	*Logical*	*input*	Creates a summary file of fitted values and residuals for each coefficient in the GAM model using a proprietary SCA FSAVE format. This file is later used to create surface plots using the GAMPLOT user-developed subroutine. The names of the individual files begin with the root name SCOEF_____. The information saved to these files are File Member: GAM_RES obsnum → Observation number y → Dependent variable yhat → Independent variable residual → residuals obsnum → Observation number x_var → Actual predictor variable series smooth_x → Smoothed x(j) lower → s(x)-1.96*se upper → s(x)+1.96*se

:PUNCH_SUR	1	*Logical*	*input*	Creates a summary file for each predictor variable in the GAM model using a proprietary SCA FSAVE format. This file is later used to create surface plots using the GAMPLOT user-developed subroutine. The names of the individual files begin with the root name SVAR_____. The information saved to these files are obsnum → Observation number x_var → Actual predictor variable series smooth_x → Smoothed x(j) lower → s(x)-1.96*se upper → s(x)+1.96*se partial_res → partial residual (smooth_x+res) spline → spline(i,j)
:FILENAME	1	*Character scalar*	*input*	Specifies the file name if output is requested. Unit 44 is used to save file. Default file name is 'gamfit.fsv'
:TOL	2	*Real number array*	*input*	Sets the inner and outer loop convergence. The default is array(:0.1D-8,0.1D-8)
:MAXIT	2	*Integer array*	*input*	Sets the maximum number of iterations for backfitting and local scoring respectively. The default is array(:1000,1000)
:SAVEX	1	*Logical*	*Input*	Saves the X matrix with internal storage name %x, the smoothed X matrix in %SMOOTHX, and the spline values in %SPLINE.

<u>Variables created in GAMFIT subroutine call:</u>

Variable	Comments
%YVAR	*Character scalar; output only* Name of the dependent variable in the GAM model.
%NAMES	*Character array; output only* Names of the independent variables in the GAM model.
%VARTYPE	*Character array; output only* *The variable type (predictor, function) for the independent variables in the GAM model*
%DF	*Integer array; output only* The Degrees of Freedom for each independent variable in the GAM model
%LINK	*Character scalar; output only* The link function used in the GAM model
%DIST	*Character scalar; output only* The error probability distribution assumed in the GAM model
%NOB	*Integer scalar; output only* The number of observations used in the estimated model
%LAG	*Integer array; output only* Lags of the independent variables.
%COEF	*Real number array; output only* Estimates of the final GAM model. Constant is in location one. Size of the array is nk+1.

%NL_P	*Real number array; output only* The probability of the coefficient possessing nonlinear tendencies %NL_P=CHISQPROB(testnl,%dof) Where testnl=(%SS_REST - %RSS)/%SIGMA2
%Z	Real number array; output only The z-score for the GAM coefficients (s.e. =%coef/%z)
%RES	*Real number array; output only* Residuals of the estimated GAM model.
%RSS	*Real number scalar; output only* Residual sum of squares value
%SS_REST	*Real number scalar; output only* Restricted residual sum of squares value
%TSS	*Real number scalar; output only* Total residual sum of squares value
%YHAT	*Real number array; output only* Fitted values of the dependent variable.
%Y	*Real number array; output only* *Actual values of the dependent variable having the same number of observations as %YHAT*
%SIGMA2	Real number scalar; output only Scaling factor
%XFOBS	*Integer array; output only* Observation number of the holdout sample (if specified)
%XFUTURE	*Real number matrix; output only* The X-Matrix for the holdout sample (if specified)

5 A DETAILED DESCRIPTION OF THE OLSQ COMMAND

The OLSQ command is a linear regression capability using OLS, L1, and MINIMAX estimation methods. The variables in the model can be real*8 or real*16. Real*16 capability is designed to handle difficult problems. Cholesky factorization is used to perform calculations although the QR (Quantile Regression) approach is an option. Recursive residuals can be optionally calculated.

Usage:

```
CALL OLSQ(Y X1 X2 :OPTIONS);
```

Lags can be entered as

```
X{1} or  X{1 to 20}
```

Required subroutine arguments:

Arguments	Intent	Comments
Y	*Input*	Dependent variable in the regression model. The dependent variable may be a random variable.
X	*Input*	One or more regressor variables in the regression model. The regressor variables may be specified also be random. If a lagged regressor variable is included in the model, the lags must be specified within braces as shown in the usage section above.

Optional keywords and associated arguments:

Keyword	# Args	Type(s)	Intent	Comments
:NOINT	1	*Logical*	*input*	Estimate the model without an intercept term.
:PRINT	1	*Logical*	*input*	Print estimation results for OLSQ
:DIAG	1	*Logical*	*input*	Print/save extensive diagnostics for the estimated model
:L1	1	*Logical*	*input*	Estimates a regression model using the L1 method which minimizes $\mathrm{sum}\left(\mathrm{abs}\left(Y_t - \hat{Y}_{t-1}\right)\right)$. This estimation method is not as sensitive to outliers as OLS or MINIMAX.
:MINIMAX	1	*Logical*	*input*	Estimates a regression model using the MINIMAX method which minimizes $\max\left(\mathrm{abs}\left(Y_t - \hat{Y}_{t-1}\right)\right)$. This estimation method is more sensitive to outliers.

:QR	1	*Logical*	*input*	Uses the quantile regression method to obtain ordinary least squares (OLS) estimates. This option is slower and more computationally intensive than the default Cholesky decomposition method. Use this option when more accuracy is needed.
:WHITE **:WHITE1** **:WHITE2** **:WHITE3**	1	*Logical*	*input*	Computes the standard errors and t-values using method introduced by White (:WHITE) or its variants (:WHITE1, :WHITE2, :WHITE3). Results are saved in temporary variables, %whitese and %whitet. For details on alternative formulas see Davidson-MacKinnon (2004) page 199-200. See also Greene (2003) page 220 for added detail.
:SAVEX	1	*Logical*	*input*	Saves the X matrix in %X. This is useful for threshold autoregressive (TAR) modeling.
:SAMPLE	1	*Real number array*	*input*	Specifies a mask real*8 variable that if = 0.0 drops that observation from estimation. The sample variable must be used with great caution when there are lags specified in the model. If lags are specified, a much better option is the :holdout option.
:HOLDOUT	1	*Integer scalar*	*input*	Sets the number of observations to omit from the back of the data during estimation. When this option is used, the %XFUTURE temporary variable is created that saved the holdout portion of the data as an X-Matrix for post-sample forecasting purposes.
:RR	1	*Integer scalar*	*input*	Calculates recursive residuals for up to a specified maximum order. When this option is employed, several temporary variables are created. These temporary variables are indicated in the next table. The :RR option is computationally intensive. The user may consider using the RR command in the B34S ProSeries Econometric System for datasets greater than 10,000 observations.

Variables created specific to OLSQ OPTIONS specified:

Variable	If (option)	Comments
%YVAR		*Character scalar; output only* Name of the dependent variable in the model
%NAMES		*Character array; output only* Names of the regressor variables in the model
%LAG		*Integer array; output only* Lag operator associated with the regressor variable
%COEF		*Real number array; output only* The estimated OLS coefficients in the model
%SE		*Real number array; output only* The standard errors of the OLS coefficient estimates
%T		*Real number array; output only* The t-values for the OLS coefficient estimates
%RSS		*Real number scalar; output only* The residual sum of squares value for OLS estimation
%SUMRE		*Real number scalar; output only*

		The sum of absolute residuals for OLS estimation
%REMAX		*Real number scalar; output only* The maximum absolute residual value for OLS estimation
%RESVAR		*Real number scalar; output only* Residual variance for OLS estimation
%RSQ		*Real number scalar; output only* Centered R-Square value for OLS estimation
%RCOND		*Real number scalar; output only* Inverse of the condition of the X’X matrix
%NOB		*Integer scalar; output only* Number of observations used in the model
%K		*Integer scalar; output only* Number of regressor variables in the model
%XPXINV		*Real number matrix; output* Inverse X’X matrix
%YHAT		*Real number array; output only* The OLS fitted values for the dependent variable
%Y		*Real number array; output only* The dependent variable
%RES		*Real number array; output only* The residuals from the OLS model estimation
%ALMXK		*Real number scalar; output only* -2.0 * log maximum likelihood function for OLS estimation
%AICSTAT		*Real number scalar; output only* Akaike Information Criteria for OLS estimation
%SICSTAT		*Real number scalar; output only* Schwartz Information Criteria for OLS estimation
%FPETEST		*Real number scalar; output only* Akaike Finite Prediction Error (1970) for OLS estimation
%GVCTEST		*Real number scalar; output only* Generalized cross validation test for OLS estimation
%HGTEST		*Real number scalar; output only* Hannan - Quinn (1979) for OLS estimation
%SHTEST		*Real number scalar; output only* Shibata test for OLS estimation
%RICETST		*Real number scalar; output only* Rice test for OLS estimation
%XFUTURE		*Real number matrix; output only* The X-Matrix for the holdout sample (if specified)
%XFOBS		*Integer array; output only* The observation number of the holdout sample (if specified)
%WHITESE	*:WHITE or :WHITE1 or :WHITE2 or :WHITE3*	*Real number array; output only* White standard errors for estimates
%WHITET	*:WHITE or :WHITE1 or :WHITE2 or :WHITE3*	*Real number array; output only* White t-values for estimates
%X	*:SAVEX*	*Real number matrix; output only* The X-Matrix
%L1COEF	*:L1*	*Real number array; output only* The estimated coefficients using L1 estimation
%L1SUMRE	*:L1*	*Real number scalar; output only* The sum of absolute errors for L1 estimation
%L1RES	*:L1*	*Real number array; output only* The residuals from L1 model estimation

%L1YHAT	*:L1*	*Real number array; output only* The fitted values from L1 estimation
%L1RSS	*:L1*	*Real number scalar; output only* Residual sum of squares from L1 estimation
%L1REMAX	*:L1*	*Real number scalar; output only* The maximum absolute residual from L1 estimation
%MMCOEF	*:MINIMAX*	*Real number array; output only* The estimated coefficients using MINIMAX estimation
%MMSUMRE	*:MINIMAX*	*Real number scalar; output only* The sum of absolute errors for MINIMAX estimation
%MMRES	*:MINIMAX*	*Real number array; output only* The residuals from MINIMAX model estimation
%MMYHAT	*:MINIMAX*	*Real number array; output only* The fitted values from MINIMAX estimation
%MMRSS	*:MINIMAX*	*Real number scalar; output only* Residual sum of squares from MINIMX estimation
%MMREMAX	*:MINIMAX*	*Real number scalar; output only* The maximum absolute residual from MINIMAX estimation
%RROBS	*:RR*	*Integer array; output only* Recursive residual observation base
%RR	*:RR*	*Real number matrix; output only* Recursive residuals matrix of dimension (N-K) by maximum order specified for the RR option
%RRSTD	*:RR*	*Real number matrix; output only* Standardized recursive residuals matrix of dimension (N-K) by maximum order specified for the RR option
%RRCOEF	*:RR*	*Real number matrix; output only* Coefficients used for recursive residuals of dimension (N-K) by maximum order specified for the RR option
%RRCOEFT	*:RR*	*Real number matrix; output only* t-Values of coefficients used for recursive residuals of dimension (N-K) by maximum order specified for the RR option
%SSR1	*:RR*	*Real number array; output only* Sumof square residuals going forward. N-2K obs
%SSR2	*:RR*	*Real number array; output only* Sum of squares residuals backward. N-2K obs

6 A DETAILED DESCRIPTION OF THE PROBIT COMMAND

The PROBIT capability estimates the probability of a dichotomous (0-1) response variable based on the cumulative normal probability distribution. The log-likelihood function is

$$\ln \mathrm{L} = \sum w_j \ln \Phi(x_j b) + \sum w_j \ln(1-\Phi(x_j b))$$

The variables in the model can be real*8 and the X-variables may be specified as a matrix if desired.

Usage:

```
CALL PROBIT(Y X1 X2 :OPTIONS);
```

Lags can be entered as

```
X{1} or  X{1 to 20}
```

Required subroutine arguments:

Arguments	Intent	Comments
Y	*Input*	Dependent variable in the probit model. The dependent variable must contain values of 0 and 1 only.
X	*Input*	One or more regressor variables in the probit model. The regressor variables may be specified as random or as dummies. If a lagged regressor variable is included in the model, the lags must be specified within braces as shown in the usage section above.

Optional keywords and associated arguments:

Keyword	# Args	Type(s)	Intent	Comments
:PRINT	1	*Logical*	*input*	Print estimation results for probit
:PRINTVCV	1	*Logical*	*input*	Print Variance-Covariance matrix
:SECD	1	*Logical*	*input*	Print second derivatives matrix
:NSTRT	1	*Integer Scalar*	*input*	Beginning observation for output of predicted and actual dependent variable and its density.
:NSTOP	1	*Integer Scalar*	*input*	Ending observation for output of predicted and actual dependent variable and its density.
:TOLA	1	*Real Number Scalar*	*input*	Convergence tolerance Default = .00001
:IITLK	1	*Logical*	*Input*	Print the Log-likelihood function after each iteration
:IIESK	1	*Logical*	*Input*	Print estimated coefficients after each iteration
:SAVEX	1	*Logical*	*input*	Saves the X-Variable matrix in the temporary variable %X
:HOLDOUT	1	*Integer scalar*	*input*	Sets the number of observations to omit from the back of the data during estimation. When this option is used, the %XFUTURE temporary variable is created that saved the holdout portion of the data as an X-Matrix for post-sample forecasting purposes.

Variables created specific to PROBIT OPTIONS specified:

Variable	If (option)	Comments
%YVAR		*Character scalar; output only* Name of the dependent variable in the model
%Y		*Real number array; output only* The dependent variable
%YHAT		*Real number array; output only* The probit fitted values for the dependent variable
%NAMES		*Character array; output only* Names of the regressor variables in the model
%LAG		*Integer array; output only* Lag operator associated with the regressor variable
%COEF		*Real number array; output only* The estimated coefficients in the probit model
%SE		*Real number array; output only* The standard errors of the coefficient estimates
%T		*Real number array; output only* The t-values for the coefficient estimates
%FUNC		*Real number scalar; output only* -2.0 times Log-likelihood function
%FUNCSIG		*Real number scalar; output only* Statistical significance of %FUNC
%DFFUNC		*Real number scalar; output only* Degrees of freedom for %FUNC
%LIMITS		*Integer scalar; output only* The number of 0's in the dependent variable
%RCOND		*Real number scalar; output only* 1 / condition of the Variance-Covariance matrix
%HESSIAN		*Real number matrix; output only* Hessian matrix
%XFUTURE		*Real number matrix; output only* The X-Matrix for the holdout sample (if specified)
%XFOBS		*Integer array; output only* The observation number of the holdout sample (if specified)
%XPXINV		*Real number matrix; output* Inverse X'X matrix

7 A DETAILED DESCRIPTION OF CUSTOMIZABLE SUBROUTINES USED IN THE GAM APPLICATION

The SCAB34S SPLINES application uses some customized user subroutines that were developed in the B34S matrix language. These subroutines perform similar to a macro procedure and are stored in the STAGING.MAC and WBSUPPL.MAC files located in the B34SLM installation folder. Since these subroutines are loaded and called from the script generated by SCA WorkBench for GAM modeling, the user can modify the scripts to handle some specific needs, as long as the nature of the arguments and calling sequence of the arguments is not changed.

For example, the P_L_EST subroutine performs LOGIT and PROBIT model estimation using two available optimization routines. The user can conceivably modify the functional forms of the model being optimized or include additional sequences inside the subroutine and have them automatically employed in the GAM Modeling application.

It is highly recommended that you make a backup of the original files before attempting to customize the subroutines.

7.1 P_L_EST User Subroutine

The P_L_EST routine estimates a Logit (or Probit) model using the CMAXF2 (Quasi-Newton) or MAXF2 which is a safeguarded quadratic interpolation method written by M.J.D. Powell of Cambridge University. The Logit estimation procedure in this user subroutine is much slower than the Probit matrix subroutine. It is recommended that the Probit option be used for faster execution.

By default, the GAM interface application will use the MAXF2 routine when estimating a Logit model as a validation against the GAM method. The standard errors for the estimated model parameters may differ between the CMAXF2 and MAXF2 routines, however since prediction accuracy is of primary interest in validating the GAM model, the MAXF2 routine seems to converge faster and with more reliability using default convergence criteria. The Probit option is not used by the GAM interface since a more comprehensive and faster Probit routine is available directly as a B34S matrix subroutine.

Usage:

```
CALL LOAD(P_L_EST  :STAGING);
CALL P_L_EST(TYPEMOD,Y,X,COEF,SE,T,YHAT,IPRINT,ROUTINE);
```

Required subroutine arguments:

Arguments	Type(s)	Intent	Comments
TYPEMOD	*Character keyword*	*input*	Sets the model type as a Logit or a Probit model. LOGIT => Logit Model PROBIT => Probit model
Y	*Real number vector*	*input*	The name of the dependent variable (0-1) in the model

X	*Real number matrix*	*input*	The name of the X-Matrix of regressor variables in the model. If lags are used, the regressor variables must be already pre-lagged in the X-Matrix. Use the LAGMATRIX routine to perform the lagging operation.
COEF	*Real number vector*	*output*	The name of the variable that the model coefficients are to be stored after estimation.
SE	*Real number vector*	*output*	The name of the variable that the standard errors of the coefficients are to be stored after estimation.
T	*Real number vector*	*output*	The name of the variable that the t-values of the model coefficients are to be stored after estimation.
YHAT	*Real number vector*	*output*	The name of the variable that the fitted values are to be stored after estimation.
IPRINT	*keyword*	*input*	Sets whether to print the output from the estimation or run silent. The keywords are PRINT => print estimation NOPRINT => silent
ROUTINE	*Integer key*	*input*	Sets the optimization routine to use for estimating the model. The integer keys are 0 => CMAXF2 (Quasi-Newton) 1 => MAXF2 (Powell)

Example:

```
call p_l_est('LOGIT',_ytmp, _xlogit,_func,_logparm,_logse,_logt,
_fflogt,'noprint',1);
```

7.2 DISP_LGT User Subroutine

The DISP_LGT subroutine displays a formatted model summary of parameter estimates, standard errors, and t-values for the Logit and Probit model estimated using the P_L_EST routine. An example of the information it generates is shown below:

```
Final Model Estimation Summary -  LOGIT
Method =>   Powell Quadratic Interpolation (MAXF2)
Functional Value=       -7.2955
----------------------------------------------------------------
  # Variable   Lag     Coefficient      std.Err      t-value
----------------------------------------------------------------
  1 Y           0             14.1    0.321E-01       438.00
  2 M           0             1.54        0.909         1.69
  3 LF          0           -0.743        0.110        -6.77
  4 NW          0             60.0         3.12         19.2
  5 TM          0        0.316E-01    0.104E-01         3.04
  6 CONSTANT    0             3.48         4.30        0.810
```

Usage:

```
CALL LOAD(DISP_LGT   :WBSUPPL);
CALL DISP_LGT(PNAMES,LAGS,COEF,SE,TVAL,FUNC,MDLDESC,METHOD);
```

Required subroutine arguments:

Arguments	Type(s)	Intent	Comments
PNAMES	*Character array*	*input*	The names of the parameters in the same order as the coefficients, standard errors, and t-values.
LAGS	*Integer array*	*input*	The lags associated with the parameters in the model.
COEF	*Real number vector*	*input*	The estimated coefficients in the model.
SE	*Real number vector*	*input*	The standard errors of the estimated coefficients in the model.
TVAL	*Real number vector*	*input*	The t-values of the estimated coefficients in the model.
FUNC	*Real number scalar*	*input*	The functional value of the estimated model.
MDLDESC	*Character string*	*input*	One or two words describing the model (e.g., LOGIT)
METHOD	*Integer key*	*input*	An integer key value assigned to the estimation method used. This should match the ROUTINE integer key in the P_L_EST routine. 0 => CMAXF2 (Quasi-Newton) 1 => MAXF2 (Powell)

Example:

```
call disp_lgt(%lmatvar,%lmatlag,_logparm,_logse,_logt,_func,'LOGIT',1);
```

7.3 DISP_OLS User Subroutine

The DISP_OLS subroutine displays a formatted model summary of parameter estimates, standard errors, and t-values for the linear regression model estimated using the OLSQ routine. An example of the information it generates is provided below.

```
Final Model Estimation Summary -  OLS
Dependent Variable:               DAYLOAD
----------------------------------------------------------------
  # Variable   Lag     Coefficient      std.Err      t-value     Comments
----------------------------------------------------------------
  1 TEMPERTR     0          70.929       34.874        2.034
  2 DAYLOAD      1           0.835        0.021       39.507
  3 CONSTANT     0       20776.779     3121.907        6.655

Number of observations:                    729
Adjusted R-Square:                       0.726
Sum of Squared Residuals:            0.110E+12
Schwartz Information Criteria:       15825.765
```

Usage:

```
CALL LOAD(COINT2       :WBSUPPL);
CALL DISP_OLS(PNAMES,LAGS,COEF,SE,TVAL,NUMOBS,ADJR2,SIC,RSS,
  MDLDESC,CMNT,ESTTYP);
```

Required subroutine arguments:

Arguments	Type(s)	Intent	Comments
PNAMES	*Character array*	*input*	The names of the parameters in the same order as the coefficients, standard errors, and t-values.
LAGS	*Integer array*	*input*	The lags associated with the parameters in the model.
COEF	*Real number vector*	*input*	The estimated coefficients in the model.
SE	*Real number vector*	*input*	The standard errors of the estimated coefficients in the model.
TVAL	*Real number vector*	*input*	The t-values of the estimated coefficients in the model.
NUMOBS	*Integer scalar*	*input*	Number of observations used in estimation
ADJR2	*Real number scalar*	*input*	The Adjusted R-Square value
SIC	*Real number scalar*	*input*	The Schwartz information criteia
RSS	*Real number scalar*	*input*	Residual Sum of Squares
MDLDESC	*Character string*	*input*	One or two words describing the model (e.g., LOGIT)
CMNT	*Character*1 array*	*input*	Row comments for a coefficient
ESTTYP	*Integer key*	*input*	An integer key value assigned to the estimation

			method used. 1 => OLS 2 => L1 3 => MiniMax

Example:

```
call olsq(DAYLOAD argument(_rxmdl)  :diag);
ccomment=c1array(norows(%coef):);
call character(vnamea,'DAYLOAD');
call disp_ols(vnamea,%names,%lag,%coef,%se,%t,%nob,
%adjrsq,%sicstat,%rss,'OLS',ccomment,1);
_ffols=goodrow(vfam(%yhat));
```

7.4 DSP_TBL User Subroutine

The DISP_TBL subroutine displays a formatted numeric table. An example of the information it generates is shown below:

```
Obs         GAM_          OLS_        DAYLOAD
  2   165338.4039   162142.7985   159213.0000
  3   150036.4320   157026.7541   138488.0000
  4   136420.1296   139959.8565   125174.0000
  5   128691.0769   129285.0584   137993.0000
  6   131868.7082   140482.1561   134406.0000
  7   133968.5805   137716.6297   134957.0000
  8   132311.4424   138080.8810   134449.0000
  9   132753.3339   136914.8574   133349.0000
 10   146452.7486   135152.4616   136808.0000
```

Usage:

```
CALL LOAD(DISP_TBL     :WBSUPPL);
CALL DISP_TBL(DMATRIX,NNAMES,ISPACE,IPREC,DESC);
```

Required subroutine arguments:

Arguments	Type(s)	Intent	Comments
DMATRIX	*Real number matrix (n,p)*	*input*	A rectangular matrix of numeric data
NNAMES	*Character array (p)*	*input*	The labels for the columns of data
ISPACE	*Integer scalar*	*Input*	The size of each column of data
IPREC	*Integer scalar*	*Input*	The number of decimals of precision displayed
DESC	*Character*1 array*	*Input*	The title description of the table

Example:

```
call dsp_tbl(_dmat,nnames,_nbegn,14,4,_desc);
```

7.5 TLGTR User Subroutine

The TLGTR subroutine produces the ratio statistics of a confusion matrix over a range of probability cut-off values. This subroutine also calls the TLOGIT subroutine to display the final Confusion Matrix to evaluate the performance of a logistic model. An example of the information it generates is shown below:

```
----------------------------------------------------------------------------------------
                      PROBIT True/False Probability CutOff Range
----------------------------------------------------------------------------------------
   Cut  True-Pos    True-Neg  False-Pos  False-Neg  Accuracy Precision   GMean1    GMean2
 0.050     1.000       0.444      0.556      0.000     0.630     0.474    0.688     0.667
 0.100     1.000       0.444      0.556      0.000     0.630     0.474    0.688     0.667
 0.150     1.000       0.500      0.500      0.000     0.667     0.500    0.707     0.707
 0.200     1.000       0.611      0.389      0.000     0.741     0.562    0.750     0.782
 0.250     1.000       0.667      0.333      0.000     0.778     0.600    0.775     0.816
 0.300     0.889       0.833      0.167      0.111     0.852     0.727    0.804     0.861
 0.350     0.889       0.833      0.167      0.111     0.852     0.727    0.804     0.861
 0.400     0.778       0.833      0.167      0.222     0.815     0.700    0.738     0.805
 0.450     0.667       0.833      0.167      0.333     0.778     0.667    0.667     0.745
 0.500     0.556       0.833      0.167      0.444     0.741     0.625    0.589     0.680
 0.550     0.556       0.833      0.167      0.444     0.741     0.625    0.589     0.680
 0.600     0.556       0.889      0.111      0.444     0.778     0.714    0.630     0.703
 0.650     0.444       0.944      0.056      0.556     0.778     0.800    0.596     0.648
 0.700     0.333       0.944      0.056      0.667     0.741     0.750    0.500     0.561
 0.750     0.222       0.944      0.056      0.778     0.704     0.667    0.385     0.458
 0.800     0.111       0.944      0.056      0.889     0.667     0.500    0.236     0.324
 0.850     0.111       1.000      0.000      0.889     0.704     1.000    0.333     0.333
 0.900     0.111       1.000      0.000      0.889     0.704     1.000    0.333     0.333
 0.950     0.000       1.000      0.000      1.000     0.667     0.000    0.000     0.000
 1.000     0.000       1.000      0.000      1.000     0.667     0.000    0.000     0.000

-----------------------------------------------------
                  Confusion Matrix
         Performance Evaluation for PROBIT
-----------------------------------------------------
                           Predicted
               Negative    Positive    Unclassified
Negative             15           3           0
Positive              1           8           0

Total Number Cases          :         27
Total Classified Cases      :         27
Unclassified Positive Cases :          0
Unclassified Negative Cases :          0
Cut-off for True  (>)       :      0.325
Cut-off for False (<)       :      0.325
Accuracy Rate               :      0.852
True  Positive Rate         :      0.889
False Positive Rate         :      0.167
True  Negative Rate         :      0.833
False Negative Rate         :      0.111
Precision Rate              :      0.727
g-mean1                     :      0.804
g-mean2                     :      0.861
```

Usage:

```
CALL LOAD(TLOGIT    :STAGING);
CALL TLGTR(YVAL,YHAT,NCUT, PTRUER,PFALSER, PFPOS, PFFALS,
ACCURACY, PRECISION, GMEAN1, GMEAN2, RPROB1, RPROB2,
UPPER,LOWER,DESC,IPROBSET,IMDL,IPRINT);
```

Required subroutine arguments:

Arguments	Type(s)	Intent	Comments
YVAL	*Real number array*	*Input*	The dependent variable (0-1) containing the actual series.
YHAT	*Real number array*	*Input*	The predicted series to be evaluated corresponding to the dependent variable.
NCUT	*Integer scalar*	*Input*	The number of probability cut-off values to evaluate between the range 0-1
PTRUER	*Real number array*	*Output*	True positive ratio
PFALSER	*Real number array*	*Output*	True negative ratio
PFPOS	*Real number array*	*Output*	False positive ratio
PFFALS	*Real number array*	*Output*	False negative ratio
ACCURACY	*Real number array*	*Output*	Accuracy rate
PRECISION	*Real number array*	*Output*	Precision rate
GMEAN1	*Real number array*	*Output*	g-mean 1 => SQRT(PTRUER*PRECISION)
GMEAN2	*Real number array*	*Output*	g-mean 2 => SQRT(PTRUER*PFALSER)
RPROB1	*Real number scalar*	*Output*	Max(GMEAN1) value
RPROB2	*Real number scalar*	*Output*	Max(GMEAN2) value
UPPER	*Real number scalar*	*Input/Output*	The threshold value used to classify a prediction as a positive instance. YHAT >= UPPER
LOWER	*Real number scalar*	*Input/Output*	The threshold value used to classify a prediction as a negative instance. YHAT < LOWER
DESC	*Character string*	*Input*	A character string that provides a description for the confusion matrix.
IPROBSET	*Integer scalar*	*Input*	Controls what cut-off probability values are used for the confusion matrix 0 ➔ UPPER and LOWER (user specified) 1 ➔ RPROB1 is used (max(gmean1)) 2 ➔ RPROB2 is used (max(gmean2))
IPRINT	*Integer key*	*Input*	Specify whether or not to print the confusion matrix Key =0 => no print Key =1 => print Key =2 => print and graph

Example:

```
_fflogt=afam(_fflogt);
_iprint=2;
_iprobset=1;
call tlgtr(_actuals,_fflogt,_ncut,_ptruer,_pfalser,_pFPos,_pFFals,
           _acc,_pprecise,_gmean1,_gmean2,rprob1,rprob2,_upper,_lower,
           _desc,_iprobSet,_iprobset,_iprint);
```

7.6 TLOGIT User Subroutine

The TLOGIT subroutine produces a confusion matrix (Kohavi and Provost, 1998) that compares actual to predicted classifications from a classification system. TLOGIT presents information to evaluate the predictive performance of a logistic model. An example of the information it generates is shown below:

```
-----------------------------------------------------
                  Confusion Matrix
          Performance Evaluation for LOGIT
-----------------------------------------------------
                                Predicted
               Negative    Positive    Unclassified
Negative              7           2           0
Positive              1          34           0

Total Number Cases          :         44
Total Classified Cases      :         44
Unclassified Positive Cases :          0
Unclassified Negative Cases :          0
Cut-off for True  (>=)      :      0.500
Cut-off for False (<)       :      0.500
Accuracy Rate               :      0.932
True  Positive Rate         :      0.971
False Positive Rate         :      0.222
True  Negative Rate         :      0.778
False Negative Rate         :      0.222
Precision Rate              :      0.944
g-mean1                     :      0.958
g-mean2                     :      0.869
```

Usage:

```
CALL LOAD(TLOGIT    :STAGING);
CALL TLOGIT(YVAL,YHAT,UPPER,LOWER,DESC,NTRUER,NTRUEP,NFALSER,
  NFALSEP,NUNCLEAR,PTRUER,PFALSER,IPRINT);
```

Required subroutine arguments:

Arguments	Type(s)	Intent	Comments
YVAL	*Real number array*	*Input*	The dependent variable (0-1) containing the actual series.
YHAT	*Real number array*	*Input*	The predicted series to be evaluated corresponding to the dependent variable.
UPPER	*Real number scalar*	*Input*	The threshold value used to classify a prediction as a positive instance. YHAT >= UPPER
LOWER	*Real number scalar*	*Input*	The threshold value used to classify a prediction as a negative instance. YHAT < LOWER
DESC	*Character string*	*Input*	A character string that provides a description for the confusion matrix.
NTRUER	*Real number*	*Output*	The number of positive cases correctly classified

	scalar		
NTRUEP	*Real number scalar*	*Output*	The total number of predicted positive cases
NFALSER	*Real number scalar*	*Output*	The number of negative cases correctly classified
NFALSEP	*Real number scalar*	*Output*	The total number of predicted negative cases
NUNCLEAR	*Real number scalar*	*Output*	The total number of cases that could not be classified. This may occur if there is a gap in the upper and lower threshold values
PTRUER	*Real number scalar*	*Output*	True positive ratio
PFALSER	*Real number scalar*	*Output*	True negative ratio
IPRINT	*Integer key*	*Input*	Specify whether or not to print the confusion matrix Key =0 => no print Key<>0 => print

Example:

```
_fflogt=afam(_fflogt);
_iprint=1;
call tlogit(_actuals,_fflogt,_upper,_lower,_desc2,_ntruer,_ntruep
   _nfalser,_nfalsep,_nunclear,_ptruer,_pfalser,_iprint);
```

7.7 LIFTGAIN User Subroutine

The LIFTGAIN subroutine displays a formatted table of the cumulative gains and lift of a classification system. LIFTGAIN presents information to evaluate the predictive performance of a logistic model compared to base level expectations. An example of the information it generates is shown below:

```
-----------------------------------------------------------------------------------------------------------------
                                           GAM Lift-Gain Table
-----------------------------------------------------------------------------------------------------------------
        #Obs in    #Pos in      %Pos in     Pctg of       Cum.       Cum.       Cum.       Cum.        K_S     Lift  Gain over
Decile   Decile     Decile       Decile   Total Pos       #Obs       #Pos       %Pos       Gain     Spread    Index     Random
     1       10         10       100.0%       40.0%         10         10      10.0%      40.0%      40.0%      400      75.0%
     2       10         10       100.0%       40.0%         20         20      20.0%      80.0%      80.0%      400      75.0%
     3       10          4        40.0%       16.0%         30         24      24.0%      96.0%      88.0%      320      68.8%
     4       10          1        10.0%        4.0%         40         25      25.0%     100.0%      80.0%      250      60.0%
     5       10          0         0.0%        0.0%         50         25      25.0%     100.0%      66.7%      200      50.0%
     6       10          0         0.0%        0.0%         60         25      25.0%     100.0%      53.3%      166      40.0%
     7       10          0         0.0%        0.0%         70         25      25.0%     100.0%      40.0%      142      30.0%
     8       10          0         0.0%        0.0%         80         25      25.0%     100.0%      26.7%      125      20.0%
     9       10          0         0.0%        0.0%         90         25      25.0%     100.0%      13.3%      111      10.0%
    10       10          0         0.0%        0.0%        100         25      25.0%     100.0%       0.0%      100       0.0%
```

Usage:

```
CALL LOAD(LIFTGAIN     :WBSUPPL);
CALL LIFTGAIN(PREDICT1,ACTUALS1,DESC,NPT,NET,CPGAIN,CPLIFT);
```

Required subroutine arguments:

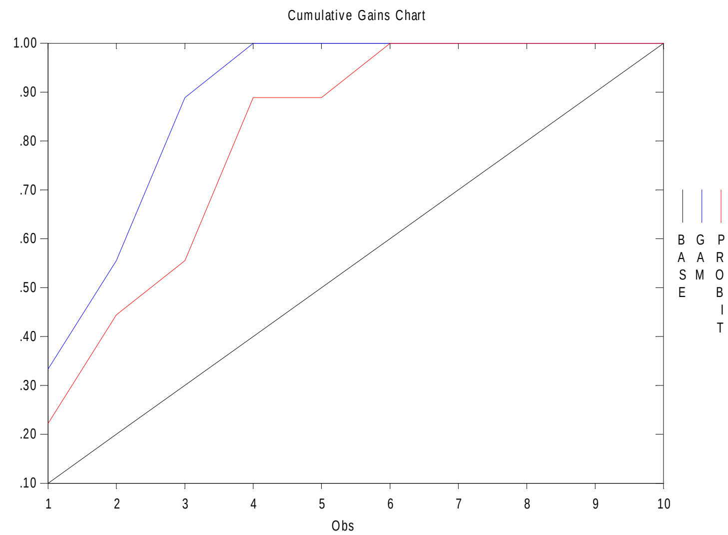
Arguments	Type(s)	Intent	Comments
PREDICT1	*Real number array*	*Input*	The predicted probabilities of positive outcomes for the sample period.
ACTUALS1	*Real number array*	*Input*	The actuals (0/1) for the sample period.
DESC	*Character string*	*Input*	A description of the table or method being displayed. For example, "GAM Lift-Gains".
NPT	*Integer array*	*Output*	The number of positive outcomes classified by the classification system by deciles.
NET	*Integer array*	*Output*	The number of expected positive outcomes by deciles.
CPGAIN	*Real number array*	*Output*	The cumulative gains of the classification system by deciles
CPLIFT	*Real number array*	*Output*	The cumulative lifts of the classification system by deciles

Example:

```
call character(_desc1b, 'GAM Lift-Gain Table');
call liftgain(_ffgam,_actuals,_desc1b,npt,nt,margain,marlift);
```

7.8 GRFLIFT User Subroutine

The GRFLIFT subroutine creates the cumulative gains chart and lift chart using the results of the LIFTGAIN subroutine. The subroutine will charts up to two comparative methods on the same chart. The charts are saved to windows-metafile files under the names GAINS.WMF and LIFT.WMF respectively. An example of the gains chart is shown below:



Usage:

```
CALL LOAD(LIFTGAIN    :WBSUPPL);
CALL GRFLIFT(GAIN1,GAIN2,LIFT1,LIFT2,NAME1,NAME2,M);
```

Required subroutine arguments:

Arguments	Type(s)	Intent	Comments
GAIN1	*Real number array*	*Input*	The cumulative gains by deciles for method 1.
GAIN2	*Real number array*	*Input*	The cumulative gains by deciles for method 2. If only one method is being displayed, pass the same array as the GAIN1 argument.
LIFT1	*Real number array*	*Input*	The cumulative lift by deciles for method 1.
LIFT2	*Real number array*	*Input*	The cumulative lift by deciles for method 2. If only one method is being displayed, pass the same array as the LIFT1 argument.
NAME1	*Integer Keyword*	*Input*	1=MARS, 2=LOGIT, 3=PROBIT, 4=OLS, 5=GAM
NAME2	*Integer Keyword*	*Input*	1=MARS, 2=LOGIT, 3=PROBIT, 4=OLS, 5=GAM
M	*Integer*	*Input*	1=>Graph Lift/Gain for method 1 only 2=>Graph Lift/Gain for method 1 and method 2

Example:

```
call grflift(margain,marlift,lgtgain,lgtlift,1,2,2);
```

7.9 DISP_HIN User Subroutine

The DISP_HIN subroutine displays a formatted table of the Hinich (1982) nonlinearity tests(subroutine hinich82()). An example of the information it generates is shown below:

```
---------------------------------------------------
Hinich82 Bi-Spectrum Nonlinear Tests -  GAM Residuals
---------------------------------------------------
Gaussality  :       -0.370
Linearity   :       -0.363
```

Usage:

```
CALL LOAD(DISP_HIN    :WBSUPPL);
CALL DISP_HIN(GAUSS,LINEAR,MDLTYPE);
```

Required subroutine arguments:

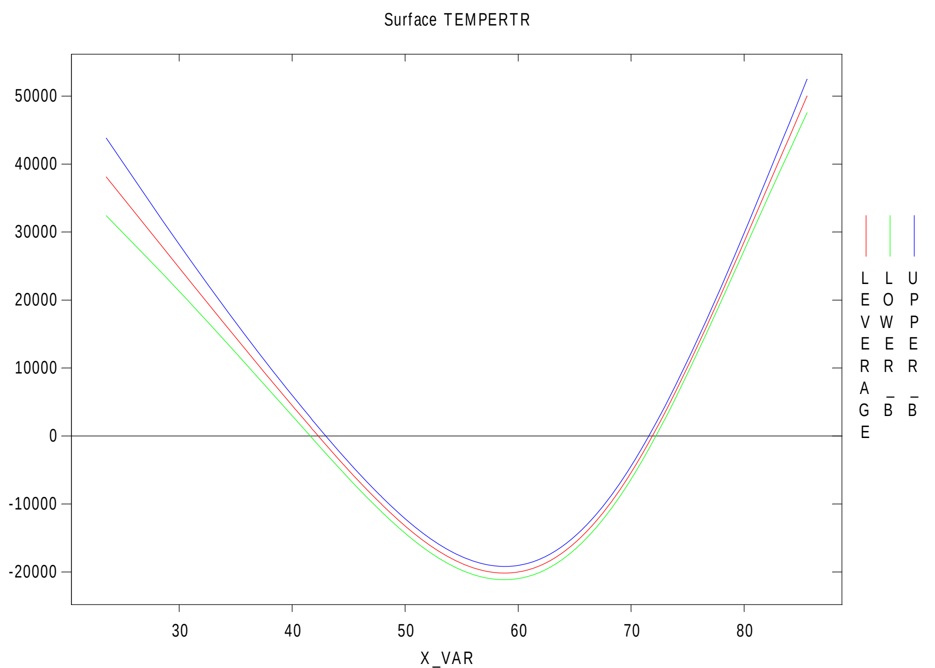
Arguments	Type(s)	Intent	Comments
GAUSS	*Real number array*	*Input*	An array(2) that contains the Hinich Gaussality tests from the HINICH82 subroutine.
LINEAR	*Real number array*	*Input*	An array(2) that contains the Hinich Linearity tests from the HINICH82 subroutine.
MDLTYPE	*Character Scalar*	*Input*	A character description of the variable that is being tested (e.g., GAM Residuals).

Example:

```
call hinich82(res2,_mols,_gols,_lols :meanonly);
call disp_hin(_gols,_lols,'OLS Residuals');
```

7.10 GAMPLOT User Subroutine

The GAMPLOT subroutine creates surface charts for all explanatory variables. The charts are saved in Windows-Meta-File (WMF) format. The number of charts created is dependent upon the number of explanatory variables in the model. Here, the contribution variable is plotted on the horizontal axis over its minimum to maximum range in the training data. An example of a contribution chart is provided below:



Usage:

```
CALL LOAD(GAMPLOT);
CALL GAMPLOT(VNAMES,VLAGS,FSV_FILE,OLS_Fitted,OLS_Resid,IPRINT);
```

Required subroutine arguments:

Arguments	Type(s)	Intent	Comments
VNAMES	*Character array(,p)*	*input*	The names of the predictor variables in the model. GAMFIT produces the %NAMES internal storage variable with the predictor variable names.
VLAGS	*Integer array(,p)*	*input*	The lags associated with the predictor variables in the model. GAMFIT produces the %LAG internal storage variable with the lags.
FSV_FILE	*Character scalar*	*input*	The name of the SCA FSAVE file that contains the information produced by GAMFIT (:PUNCH_RES and :PUNCH_SUR options). The default file created by GAMFIT is named 'gamdata.fsv'.

OLS_Fitted	*Real number array*	*input*	The fitted values from the OLS, LOGIT, or PROBIT linear model.
OLS_Resid	*Real number array*	*input*	The residuals from the OLS, LOGIT, or PROBIT linear model.
IPRINT	*Integer scalar*	*input*	Sets the level of graphics and controls display. 0 → Display leverage graphs only 1 → Save leverage graph as SCOEF__n.wmf 2 → Save and display leverage graphs 10 → Same as 0 with addition of raw and smoothed series 11 → Same as 1 with addition of raw and smoothed series 12 → Same as 2 with addition of raw and smoothed series

Example:

```
call gamplot(%names, %lag,'gamdata.fsv',_ffols,_olsres,2);
```

7.11 GAMFORE User Subroutine

The GAMFORE subroutine uses polynomial regression to perform out-of-sample forecasting with GAM models. More details about the rationale is documented in Stokes (2008, Chapter 14).

Usage:

```
CALL LOAD(POLYFIT);
CALL LOAD(POLYVAL);
CALL LOAD(GAMFORE);
CALL GAMFORE(SPLINE,XMAT,HFUTURE,DEC_MOD,COEF,FORE_GAM);
```

Required subroutine arguments:

Arguments	Type(s)	Intent	Comments
SPLINE	*Real number matrix(n1,p)*	*input*	The matrix of estimated splines associated with the predictor variables. GAMFIT produces the %SPLINE internal storage variable with spline information.
XMAT	*Real number matrix(n1,p)*	*input*	The original X-Matrix containing the predictor variables used for estimation. GAMFIT produces the %X internal storage variable with the X-Matrix.
HFUTURE	*Real number matrix(n2,p)*	*input*	The holdout X-Matrix containing the predictor variables for forecasting. GAMFIT produces the %XFUTURE internal storage variable with the holdout X-Matrix.
DEG_MOD	*Integer scalar*	*input*	The degree of polynomial fitting to be used for forecasting purposes. This setting should be set between 3 and 10 for most applications. A higher setting will capture more curvature but may be susceptible to convergence errors.
COEF	*Real number array(p)*	*input*	The estimated coefficients of the GAM model to be used for forecasting. GAMFIT produces the %COEF internal storage variable with the estimated coefficients.
FORE_GAM	*Real number array(n2)*	*output*	The forecasted series produced by GAMFORE

Example:

```
call gamfore(%spline, %x, %xfuture, 9, %coef, _ffgam);
```

8 EXAMPLES OF SCAB34S COMMAND FILES FOR GAM MODELING

SCA WorkBench was used to automatically generate the command files for the SCAB34S engine for GAM modeling, OLS modeling, Logit-Probit modeling, estimation, diagnostics, and graphics. This section presents the command files generated by WorkBench for the examples used in this document.

8.1 SCAB34S Commands Generated for the Electricity Load Example

```
b34sexec matrix;
call echooff;
call getsca('ELOAD.mad' :mad :member DATA);
call load(disp_hin :wbsuppl);
call load(dsp_acf  :wbsuppl);
call load(coint2   :wbsuppl);
call load(dsp_tbl  :wbsuppl);
call load(contrib  :wbsuppl);
call load(catbuild :wbsuppl);
call load(polyfit);
call load(polyval);
call load(gamplot);
call load(gamfore);
call print('-------------------------------------------------------':);
call print('** Analysis Performed on Variable: DAYLOAD':);
call print('-------------------------------------------------------':);

call dspdscrb('DAYLOAD Dependent Variable',DAYLOAD);

/$ **************************************************************$/
/$ Set span for sample: DAYLOAD
/$ **************************************************************$/
iorigins=integers(1, 730);
DAYLOAD = DAYLOAD(iorigins);
DATE = DATE(iorigins);
TEMPERTR = TEMPERTR(iorigins);
DOW = DOW(iorigins);
D1 = D1(iorigins);
D2 = D2(iorigins);
D3 = D3(iorigins);
D4 = D4(iorigins);
D5 = D5(iorigins);
D6 = D6(iorigins);

/$ *******************************************************
/$ _holdout is the number of forecasts
/$ _maxlag is the longest lag in model
/$ _nlags is number of lags for diagnostics
/$ _ffgam holds the GAM forecasts
/$ _degmod is the D.F. for polynomial regression used in prediction
/$ _distr specifies the error distribiton for GAM
/$ _linkf specifies the linking function for GAM
/$ _ffols holds the OLS forecasts for comparison
/$ _olsres holds the OLS residuals for comparison
/$ _actuals holds the actuals for comparison
/$ _rxmdl holds the Predictor X-variable model components for GAM models
/$ _rxols holds the Predictor X-variable model components for OLS models
/$ _cxmdl# holds the categorical X-variable model components for GAM models
/$ _cxols# holds the categorical X-variable model components for OLS models
/$ *******************************************************
call character(_linkf,'ident');
call character(_distr,'gauss');
_holdout=0;
_nlags=12;

if((_holdout.le._nlags).and.(_holdout.gt.0)) _nlags=_holdout-1;
```

```
_maxlag=1;
imax=1;
if(imax .le. _maxlag) imax=_maxlag+1;
 nn=integers(imax,norows(DAYLOAD));
 nnact=integers(imax,norows(DAYLOAD));
_actuals=vfam(DAYLOAD(nnact));


/$ ****************************************************************
/$  Specify model components using character string variables
/$ ****************************************************************

call character(_rxols,
'TEMPERTR{ 0 TO 1 }
DAYLOAD{ 1 }'
);

call character(_rxmdl,
'TEMPERTR[Predictor,3]{ 0 TO 1 }
DAYLOAD[Predictor,3]{ 1 }'
);

call fprint(:clear);
/$ ****************************************************************
/$  Expand categorical values into indicator variables: DOW
/$ ****************************************************************
catmat1=matrix(2,2:);
catnam1=array(2:);

call character(chlags,'{0}');
call ialen(chlags,ilen);
call catbuild(DOW ,'DOW ',1,catmat1,catnam1);
ipos=1;
ipos2=10;
k=0;
call fprint(:clear);
do i=2,nocols(catmat1);
  call copy(catmat1(,i),argument(catnam1(i)));
  call fprint(:col ipos :display catnam1(i)
              :col ipos2 :display chlags
              :save catlbl1);
  ipos=ipos2+ilen+1;
  ipos2=ipos+10;
  if((ipos.gt.120).or.(i.eq.nocols(catmat1)))then;
    ibuff=integers(1,130);
    jbuff=ibuff+(k*130);
    _cxols1 (jbuff)=catlbl1(ibuff);
    catlbl1=' ';
    call fprint(:clear);
    ipos=1;
    ipos2=10;
    k=k+1;
  endif;
enddo;


ipos=1;
ipos2=10;
ipos3=23;
k=0;
call fprint(:clear);
do i=2,nocols(catmat1);
  call copy(catmat1(,i),argument(catnam1(i)));
  call fprint(:col ipos :display catnam1(i)
              :col ipos2 :display '[factor,1]'
              :col ipos3 :display chlags
              :save catlbl1);
  ipos=ipos3+ilen+9;
  ipos2=ipos+10;
```

```
  ipos3=ipos2+13;
  if((ipos.gt.106).or.(i.eq.nocols(catmat1)))then;
    ibuff=integers(1,130);
    jbuff=ibuff+(k*130);
    _cxmdl1 (jbuff)=catlbl1(ibuff);
    catlbl1=' ';
    call fprint(:clear);
    ipos=1;
    ipos2=10;
    ipos3=23;
    k=k+1;
  endif;
enddo;
call fprint(:clear);

/$ ****************************************************************
/$  Specify/Estimate the OLS model
/$ ****************************************************************

call olsq(DAYLOAD argument(_rxols)  argument(_cxols1)
 :diag);
ccomment=c1array(norows(%coef):);
call character(vnamea,'DAYLOAD');

/$ ****************************************************************
/$  Formatted display of OLS model estimation results
/$ ****************************************************************
call disp_ols(vnamea,%names,%lag,%coef,%se,%t,%nob,
%adjrsq,%sicstat,%rss,'OLS',ccomment,1);

/$ ****************************************************************
/$  Store the OLS predicted values and residuals
/$ ****************************************************************
_ffols=goodrow(vfam(%yhat));
_olsres=goodrow(vfam(%res));


/$ ******************************************************
/$ Specify/Estimate GAM model
/$ ******************************************************

call GAMFIT(DAYLOAD argument(_rxmdl)  argument(_cxmdl1)
 :print :tol array(2:.00000001,.00000001) :maxit index(1000,1000)
 :punch_res :punch_sur :filename 'gamdata.fsv' :savex
 :dist argument(_distr) :link argument(_linkf)  );

/$ ****************************************************************
/$  Store the GAM model predicted values
/$ ****************************************************************
_ffgam=goodrow(vfam(%yhat));

/$ **************************************************************$/
/$ Compute performance criteria for prediction
/$ **************************************************************$/

_n01=sum(DAYLOAD .eq. 0.0 .or. DAYLOAD .eq. 1.0);
if(_n01 .lt. dfloat(norows(DAYLOAD))) then;

gamsq=sumsq(_actuals-_ffgam);
m_rmse=dsqrt(mean(afam(_actuals-_ffgam)**2.));
m_mape=mean(afam(dabs(_actuals-_ffgam))/afam(dabs(_actuals)))*100.;
olsssq=sumsq(_actuals-_ffols);
o_rmse=dsqrt(mean(afam(_actuals-_ffols)**2.));
o_mape=mean(afam(dabs(_actuals-_ffols))/afam(dabs(_actuals)))*100.;

call print('------------------------------------------------------':);
call print('** Prediction Performance Criteria':);
call print('------------------------------------------------------':);
call print('OLS RMSE: ', o_rmse :format '(f12.3)');
```

```
call print('OLS MAPE: ', o_mape :format '(f12.3)');
call print('GAM RMSE: ', m_rmse :format '(f12.3)');
call print('GAM MAPE: ', m_mape :format '(f12.3)');
endif;

res1=_ffgam - _actuals;
res2=_ffols - _actuals;
call graph(_actuals,_ffgam,_ffols :file 'yfit.wmf'
:heading 'Original vs. Predicted' :nolabel
    :colors black black bblue bred);
call dspdscrb('GAM Residuals',res1);

.
. The commands related to building graphics in SCAB34S omitted here for brevity. It is
. a common script that will be presented later in this section.
.
.

call gamplot(%names,%lag,'gamdata.fsv',_ffols,_olsres,1);
b34srun;
```

8.2 SCAB34S Commands for the Nerlove Data Example

```
b34sexec matrix;
call echooff;
call getsca('NERLOVE.MAD' :mad :member DATA);
call load(disp_hin :wbsuppl);
call load(dsp_acf  :wbsuppl);
call load(coint2   :wbsuppl);
call load(dsp_tbl  :wbsuppl);
call load(contrib  :wbsuppl);
call load(catbuild :wbsuppl);
call load(polyfit);
call load(polyval);
call load(gamplot);
call load(gamfore);
call print('------------------------------------------------------':);
call print('** Analysis Performed on Variable: LNCP3':);
call print('------------------------------------------------------':);

call dspdscrb('LNCP3 Dependent Variable',LNCP3);

/$ *******************************************************
/$ _holdout is the number of forecasts
/$ _maxlag is the longest lag in model
/$ _nlags is number of lags for diagnostics
/$ _ffgam holds the GAM forecasts
/$ _degmod is the D.F. for polynomial regression used in prediction
/$ _distr specifies the error distribiton for GAM
/$ _linkf specifies the linking function for GAM
/$ _ffols holds the OLS forecasts for comparison
/$ _olsres holds the OLS residuals for comparison
/$ _actuals holds the actuals for comparison
/$ _rxmdl holds the Predictor X-variable model components for GAM models
/$ _rxols holds the Predictor X-variable model components for OLS models
/$ *******************************************************
call character(_linkf,'ident');
call character(_distr,'gauss');
_holdout=0;
_nlags=12;
if((_holdout.le._nlags).and.(_holdout.gt.0)) _nlags=_holdout-1;
_maxlag=0;
imax=1;
if(imax .le. _maxlag) imax=_maxlag+1;
 nn=integers(imax,norows(LNCP3));
 nnact=integers(imax,norows(LNCP3));
_actuals=vfam(LNCP3(nnact));

/$ ****************************************************************
/$  Specify model components using character string variables
/$ ****************************************************************

call character(_rxols,
'LNKWH
LNP13
LNP23 '
);

call character(_rxmdl,
'LNKWH[Predictor,3]
LNP13[Predictor,3]
LNP23[Predictor,3] '
);
```

```
/$ ****************************************************************
/$  Specify/Estimate the OLS model
/$ ****************************************************************

call olsq(LNCP3 argument(_rxols)  :diag);
ccomment=c1array(norows(%coef):);
call character(vnamea,'LNCP3');

/$ ****************************************************************
/$  Formatted display of OLS model estimation results
/$ ****************************************************************
call disp_ols(vnamea,%names,%lag,%coef,%se,%t,%nob,
%adjrsq,%sicstat,%rss,'OLS',ccomment,1);

/$ ****************************************************************
/$  Store the OLS predicted values and residuals
/$ ****************************************************************
_ffols=goodrow(vfam(%yhat));
_olsres=goodrow(vfam(%res));


/$ *******************************************************
/$ Specify/Estimate GAM model
/$ *******************************************************

call GAMFIT(LNCP3 argument(_rxmdl)  :print :tol
array(2:.00000001,.00000001) :maxit index(1000,1000)
 :punch_res :punch_sur :filename 'gamdata.fsv' :savex
 :dist argument(_distr) :link argument(_linkf)  );

/$ ****************************************************************
/$  Store the GAM model predicted values
/$ ****************************************************************
_ffgam=goodrow(vfam(%yhat));

/$ **************************************************************$/
/$ Compute performance criteria for prediction
/$ **************************************************************$/

_n01=sum(LNCP3 .eq. 0.0 .or. LNCP3 .eq. 1.0);
if(_n01 .lt. dfloat(norows(LNCP3))) then;

gamsq=sumsq(_actuals-_ffgam);
m_rmse=dsqrt(mean(afam(_actuals-_ffgam)**2.));
m_mape=mean(afam(dabs(_actuals-_ffgam))/afam(dabs(_actuals)))*100.;
olsssq=sumsq(_actuals-_ffols);
o_rmse=dsqrt(mean(afam(_actuals-_ffols)**2.));
o_mape=mean(afam(dabs(_actuals-_ffols))/afam(dabs(_actuals)))*100.;

call print('-----------------------------------------------------':);
call print('** Prediction Performance Criteria':);
call print('-----------------------------------------------------':);
call print('OLS RMSE: ', o_rmse :format '(f12.3)');
call print('OLS MAPE: ', o_mape :format '(f12.3)');
call print('GAM RMSE: ', m_rmse :format '(f12.3)');
call print('GAM MAPE: ', m_mape :format '(f12.3)');
endif;

res1=_ffgam - _actuals;
res2=_ffols - _actuals;
call graph(_actuals,_ffgam,_ffols :file 'yfit.wmf'
:heading 'Original vs. Predicted' :nolabel
    :colors black black bblue bred);
call dspdscrb('GAM Residuals',res1);

/$ **************************************************************$/
/$ Compute ACF, PACF, Q-stats for model residuals
/$ **************************************************************$/
call dsp_acf('GAM Residuals',res1,_nlags,acfa,pacfa,sea,
```

```
mqa,pmqa,-1,0,0);
if(norows(res1).gt.50)then;
call hinich82(res1,_mgam,_ggam,_lgam :meanonly);
call disp_hin(_ggam,_lgam,'GAM Residuals');
endif;
call dspdscrb('OLS Residuals',res2);

/$ **************************************************************$/
/$ Compute ACF, PACF, Q-stats for model residuals
/$ **************************************************************$/
call dsp_acf('OLS Residuals',res2,_nlags,acfb,pacfb,seb,
mqb,pmqb,-1,0,0);
if(norows(res2).gt.50)then;
call hinich82(res2,_mols,_gols,_lols :meanonly);
call disp_hin(_gols,_lols,'OLS Residuals');
endif;

.
. The commands related to building graphics in SCAB34S omitted here for brevity. It is
. a common script that will be presented later in this section.
.
.

call gamplot(%names,%lag,'gamdata.fsv',_ffols,_olsres,1);
b34srun;
```

8.3 SCAB34S Commands for the Cancer Remission Example

```
b34sexec matrix;
call echooff;
call getsca('REMISSION.MAD'  :mad :member DATA);
call load(disp_hin :wbsuppl);
call load(dsp_acf  :wbsuppl);
call load(coint2   :wbsuppl);
call load(dsp_tbl  :wbsuppl);
call load(contrib  :wbsuppl);
call load(catbuild :wbsuppl);
call load(tlogit   :staging);
call load(liftgain :wbsuppl);
call load(disp_lgt :wbsuppl);
call load(polyfit);
call load(polyval);
call load(gamplot);
call load(gamfore);

call print('-------------------------------------------------------':);
call print('** Analysis Performed on Variable: REMISS':);
call print('-------------------------------------------------------':);

call dspdscrb('REMISS Dependent Variable',REMISS);

/$ *****************************************************
/$ _holdout is the number of forecasts
/$ _maxlag is the longest lag in model
/$ _nlags is number of lags for diagnostics
/$ _ffgam holds the GAM forecasts
/$ _degmod is the D.F. for polynomial regression used in prediction
/$ _distr specifies the error distribiton for GAM
/$ _linkf specifies the linking function for GAM
/$ _ffprbt holds the PROBIT forecasts for comparison
/$ _actuals holds the actuals for comparison
/$ _rxmdl holds the Predictor X-variable model components for GAM models
/$ _rxols holds the Predictor X-variable model components for OLS models
/$ *****************************************************
_holdout=0;
_nlags=12;
if((_holdout.le._nlags).and.(_holdout.gt.0)) _nlags=_holdout-1;
_maxlag=0;
_upper=.501;
_lower=.501;
call character(_desc1,
'Performance Evaluation for GAM');
call character(_linkf,'logit');
call character(_distr,'poiss');
call character(_desc2,'Performance Evaluation for PROBIT');

imax=1;
if(imax .le. _maxlag) imax=_maxlag+1;
 nn=integers(imax,norows(REMISS));
 nnact=integers(imax,norows(REMISS));
_actuals=vfam(REMISS(nnact));

/$ **************************************************************
/$  Specify model components using character string variables
/$ **************************************************************

call character(_rxols,
'TEMP BLAST LI INFIL SMEAR CELL ');


call character(_rxmdl,
'TEMP[Predictor,3]  BLAST[Predictor,3] LI[Predictor,3]
 INFIL[Predictor,3] SMEAR[Predictor,3] CELL[Predictor,3] ');
```

```
call probit(REMISS argument(_rxols)  :print );
%lmatvar=%names;
%lmatlag=%lag;
_logparm=%coef;
_logse=%se;
_logt=%t;
_func=%func;

_ffprbt=goodrow(vfam(%yhat));
_olscoef=%coef;
_olsyhat=%yhat;
_olsres =%res;
call disp_lgt(%lmatvar,%lmatlag,_logparm,_logse,
_logt,_func,'PROBIT',0);

_ffprbt=vfam(_ffprbt);
_iprint=2;
call tlgtr(_actuals,_ffprbt,_ncut,_ptruer,_pfalser,_pFPos,_pFFals,
           _acc,_pprecise, _gmean1,_gmean2,rprob1,rprob2,_upper,_lower,
           _desc2,_iprobSet,3,_iprint);

call character(_desc2b, 'PROBIT Lift-Gain Table');
call liftgain(_ffprbt,_actuals,_desc2b,npt,nt,lgtgain,lgtlift);


/$ *******************************************************
/$ Specify/Estimate GAM model
/$ *******************************************************

call GAMFIT(REMISS argument(_rxmdl)  :print :tol
array(2:.00000001,.00000001) :maxit index(1000,1000)
 :punch_res :punch_sur :filename 'gamdata.fsv' :savex
 :link argument(_linkf) :dist argument(_distr) );

/$ ****************************************************************
/$  Store the GAM model predicted values
/$ ****************************************************************
_ffgam=goodrow(vfam(%yhat));
_iprint=2;
call tlgtr(_actuals,_ffgam,_ncut,_ptruer,_pfalser,_pFPos,_pFFals,
           _acc,_pprecise, _gmean1,_gmean2,rprob1,rprob2,_upper,_lower,
           _desc1,_iprobSet,5,_iprint);

call character(_desc2b, 'GAM Lift-Gain Table');
call liftgain(_ffgam,_actuals,_desc2b,npt,nt,gamgain,gamlift);


/$ *************************************************************$/
/$ Compute performance criteria for prediction
/$ *************************************************************$/

_n01=sum(REMISS .eq. 0.0 .or. REMISS .eq. 1.0);
if(_n01 .lt. dfloat(norows(REMISS))) then;

gamsq=sumsq(_actuals-_ffgam);
m_rmse=dsqrt(mean(afam(_actuals-_ffgam)**2.));
m_mape=mean(afam(dabs(_actuals-_ffgam))/afam(dabs(_actuals)))*100.;
olsssq=sumsq(_actuals-_ffprbt);
o_rmse=dsqrt(mean(afam(_actuals-_ffprbt)**2.));
o_mape=mean(afam(dabs(_actuals-_ffprbt))/afam(dabs(_actuals)))*100.;

call print('----------------------------------------------------':);
call print('** Prediction Performance Criteria':);
call print('----------------------------------------------------':);
call print('OLS RMSE: ', o_rmse :format '(f12.3)');
call print('OLS MAPE: ', o_mape :format '(f12.3)');
call print('GAM RMSE: ', m_rmse :format '(f12.3)');
call print('GAM MAPE: ', m_mape :format '(f12.3)');
```

```
endif;

res1=vfam(_ffgam) - vfam(_actuals);
res2=vfam(_ffprbt)- vfam(_actuals);
call graph(_actuals,_ffgam,_ffprbt :file 'yfit.wmf'
:heading 'Original vs. Predicted' :nolabel
    :colors black black bblue bred);
call grflift(gamgain,gamlift,lgtgain,lgtlift,5,3,2);

.
. The commands related to building graphics in SCAB34S omitted here for brevity. It is
. a common script that will be presented later in this section.
.
.

call gamplot(%names,%lag,'gamdata.fsv',_ffprbt,_olsres,1);

b34srun;
```

8.4 SCAB34S Commands Used to Display Graphs in the Examples

```
/$ *******************************************************
/$ Display graphics for Dependent Variable
/$ *******************************************************
call graph(DAYLOAD :file 'yvar.wmf' :noshow
     :pspaceon
     :pgyscaleright 'i'
     :pgborder
     :pgxscaletop 'i'
     :nocontact
     :colors black bblue
     :heading 'Plot of DAYLOAD');

/$*******************************************************
/$ Display graphics for GAM Residuals
/$ *******************************************************
call graph(res1  :file 'resa.wmf' :noshow
     :pspaceon
     :pgyscaleright 'i'
     :pgborder
     :pgxscaletop 'i'
     :nocontact
     :colors black bblue
     :heading 'Plot of GAM Model Residuals');
call graph(acfa, sea  :file 'acfa.wmf' :noshow
     :pspaceon
     :pgyscaleright 'i'
     :pgborder
     :pgxscaletop 'i'
     :histscale integers(0,_nlags,2)
     :overlay acfplot
     :colors black bblue bred
     :heading 'ACF of GAM Model Residuals');
call graph(mqa  :file 'mqa.wmf' :noshow
     :pspaceon
     :pgyscaleright 'i'
     :pgborder
     :pgxscaletop 'i'
     :nocontact
     :colors black bblue
     :heading 'Q-Stats from GAM Model Residuals');

/$*******************************************************
/$ Display graphs for OLSQ model residuals
/$ *******************************************************
call graph(res2 :file 'resb.wmf' :noshow
     :pspaceon
     :pgyscaleright 'i'
     :pgborder
     :pgxscaletop 'i'
     :nocontact
     :colors black bblue
     :heading 'Plot of OLSQ Model Residuals');
call graph(acfb, seb :file 'acfb.wmf' :noshow
     :pspaceon
     :pgyscaleright 'i'
     :pgborder
     :pgxscaletop 'i'
     :histscale integers(0,_nlags,2)
     :overlay acfplot
     :colors black bblue bred
     :heading 'ACF of OLSQ Residuals');
call graph(mqb  :file 'mqb.wmf' :noshow
     :pspaceon
     :pgyscaleright 'i'
```

```
 :pgborder
 :pgxscaletop 'i'
 :nocontact
 :colors black bblue
 :heading 'Q-Stats from OLSQ Residuals');\

/$ *************************************************************$/
/$ Create contribution charts for righthand-side variables
/$ *************************************************************$/
call lagmatrix( argument(_rxmdl)
               :noint :matrix tmat);
_medians=array(nocols(tmat):);
_means=_medians;

do i=1,norows(_medians);
 call describe(tmat(,i));
 _medians(i)=%median;
 _means(i)=%mean;
enddo;

call contrib(_medians,_means,3);
```

REFERENCES

Faraway, J. (2006). *Extending the Linear Model with R*, New York: Chapman & Hall/CRC

Hastie, T. and Tibshirani, R. (1990). *Generalized Additive Models*. Chapman & Hall

Hinich, M. (1982). "Testing for Gaussianity and Linearity of a Stationary Time Series". *Journal of Time Series Analysis* 3: 169-176

Kubat, M. and Matwin, S. (1997). "Addressing the Curse of Imbalanced Training Sets: One Sided Selection". In *Proceedings of the Fourteenth International Conference on Machine Learning*: 179–186. Nashville, Morgan Kaufmann.

Kubat, M. and Holte, R. and Matwin, S. (1998). "Machine Learning for the Detection of Oil Spills in Satellite Radar Images". *Machine Learning* 30: 195–215.

Lattyak, W.J. and Liu, L-M. (2005). *SCA WorkBench User's Guide*. Scientific Computing Associates Corp.

Ljung, G.M. and Box, G.E.P. (1978). "On a Measure of Lack of Fit in Time Series Models". *Biometrika* 66: 265-270

Lee, E.T. (1974), "A Computer Program for Linear Logistic Regression Analysis," Computer Programs in Biomedicine, 80–92.

Nerlove, M. (1963), "Returns to Scale in Electricity Supply", In *Measurement in Economics - Studies in Mathematical Economics and Econometrics in Memory of Yehuda Grunfeld*, edt. by Christ, C. Stanford University Press.

Provost, F., Fawcett, T. and Kohavi, R. (1998). "The Case Against Accuracy Estimation for Comparing Induction Algorithms". In *Proceedings of the Fifteenth International Conference on Machine Learning*: 445–453. San Francisco, Morgan Kaufmann.

Stokes, H.H. (2004) *The B34S ProSeries Econometric System* (software).

Stokes, H.H. (1997). *Specifying and Diagnostically Testing Econometric Models*. Second Edition. New York, Quorum Books.

Stokes, H.H. (2008) "Chapter 14: Nonlinear Model Building using Spline and Related Methods " in *Specifying and Diagnostically Testing Econometric Models, Volume II*. (manuscript).

Stokes, H.H. (2002) "Chapter 16: Programming in the Matrix Command" in *Specifying and Diagnostically Testing Econometric Models*, *Volume II*. (manuscript).

Stone, C.J. (1985),"Additive Regression and Other Nonparametric Models", Annals of Statistics, 13, 689-705.