THE POWER TO KNOW.

SAS/STAT® 12.1 User's Guide
The ADAPTIVEREG Procedure (Chapter)

SAS® Documentation

This document is an individual chapter from *SAS/STAT® 12.1 User's Guide*.

The correct bibliographic citation for the complete manual is as follows: SAS Institute Inc. 2012. *SAS/STAT® 12.1 User's Guide*. Cary, NC: SAS Institute Inc.

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

Electronic book 1, August 2012

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/publishing** or call 1-800-727-3228.

Chapter 24
The ADAPTIVEREG Procedure
(Experimental)

Contents

Overview: ADAPTIVEREG Procedure

The ADAPTIVEREG procedure fits multivariate adaptive regression splines as defined by Friedman (1991b). The method is a nonparametric regression technique that combines both regression splines and model selection methods. It does not assume parametric model forms and does not require specification

of knot values for constructing regression spline terms. Instead, it constructs spline basis functions in an adaptive way by automatically selecting appropriate knot values for different variables and obtains reduced models by applying model selection techniques.

PROC ADAPTIVEREG supports models with classification variables (Friedman 1991a) and offers options for improving modeling speed (Friedman 1993). PROC ADAPTIVEREG also extends the method to data with response variables that are distributed in the exponential family as suggested in Buja et al. (1991). The procedure can take advantage of multicore processors to distribute the computation to multiple threads.

SAS/STAT software offers various tools for nonparametric regression, including the GAM, LOESS, and TPSPLINE procedures. Typical nonparametric regression methods involve a large number of parameters in order to capture nonlinear trends in data; thus the model space is much larger than it is in more restricted parametric models. The fitting algorithms for nonparametric regression models are usually more complicated than for parametric regression models. Also, the sparsity of data in high dimensions often causes slow convergence or failure in many nonparametric regression methods. As the number of predictors increases, the model variance increases rapidly because of the sparsity. This phenomenon is referred as the "curse of dimensionality" (Bellman 1961). Hence, the LOESS and TPSPLINE procedures are limited to problems in low dimensions. PROC GAM fits generalized additive models with the additivity assumption. By using the local scoring algorithm (Hastie and Tibshirani 1990), PROC GAM can handle larger data sets than the other two procedures. However, the computation time for the local scoring algorithm to converge increases rapidly as data size grows, and the convergence for nonnormal distributions is not guaranteed. PROC ADAPTIVEREG uses the multivariate adaptive regression splines method, which is similar to the method used for the recursive partitioning models (Breiman et al. 1984). It creates an overfitted model first with the fast-update algorithm (Friedman 1991b); then prunes it back with the backward selection technique.

The main features of the ADAPTIVEREG procedure are as follows:

- supports classification variables with ordering options
- enables you to force effects in the final model or restrict variables in linear forms
- supports options for fast forward selection
- supports data with response variables that are distributed in the exponential family
- supports partitioning of data into training, validation, and testing roles
- provides leave-one-out and *k*-fold cross validation
- produces a graphical representation of the selection process, model fit, functional components, and fit diagnostics
- produces an output data set that contains predicted values and residuals
- produces an output data set that contains the design matrix of formed basis functions
- supports multiple SCORE statements

Getting Started: ADAPTIVEREG Procedure

This example concerns city-cycle fuel efficiency and automobile characteristics for 361 vehicle models made from year 1970 to 1982. The data can be downloaded from the UCI Machine Learning Repository (Asuncion and Newman 2007). The following DATA step creates the data set autompg:

```
title 'Automobile MPG Study';
data Autompg;
   input MPG Cylinders Displacement Horsepower Weight
         Acceleration Year Origin Name $35.;
   datalines;
18.0 8 307.0   130.0   3504   12.0   70  1  Chevrolet Chevelle Malibu
15.0 8 350.0   165.0   3693   11.5   70  1  Buick Skylark 320
18.0 8 318.0   150.0   3436   11.0   70  1  Plymouth Satellite
16.0 8 304.0   150.0   3433   12.0   70  1  AMC Rebel SST

   ... more lines ...

44.0 4 97.00   52.00   2130   24.6   82  2  VW Pickup
32.0 4 135.0   84.00   2295   11.6   82  1  Dodge Rampage
28.0 4 120.0   79.00   2625   18.6   82  1  Ford Ranger
31.0 4 119.0   82.00   2720   19.4   82  1  Chevy S-10
;
```

There are nine variables in the data set. The response variable MPG is city-cycle mileage per gallon (MPG). Seven predictor variables (Cylinders, Displacement, HorsePower, Weight, Acceleration, Year, and Origin) provide vehicle attributes. Among them, Cylinders, Year, and Origin are categorical variables. The last variable, Name, contains the specific name of each vehicle model.

The dependency of vehicle fuel efficiency on various factors might be nonlinear. There might also be redundant predictor variables as a result of dependency structures within predictors. For example, a vehicle model with more cylinders is likely to have more horsepower. The objective of this example is to explore the nonlinear dependency structure and also to produce a parsimonious model that does not overfit and thus has good predictive power. The following invocation of the ADAPTIVEREG procedure fits an additive model with linear spline terms of continuous predictors. By default, PROC ADAPTIVEREG fits a nonparametric regression model that includes two-way interaction between spline basis functions. You can try models with even higher interaction orders by specifying the MAXORDER= option in the MODEL statement. For this particular data set, the sample size is relatively small. Restricting model complexity by specifying an additive model can both improve model interpretability and reduce model variance without sacrificing much predictive power. The additive model consists of terms of nonparametric transformations of variables. The transformation of each variable and the selection of transformed terms are performed in an adaptive and automatic way.

```
ods graphics on;

proc adaptivereg data=autompg plots=all;
   class cylinders year origin;
   model mpg = cylinders displacement horsepower
               weight acceleration year origin / additive;
run;
```

PROC ADAPTIVEREG summarizes important information about the model that you are fitting in Figure 24.1.

Figure 24.1 Model Information and Fit Controls

Automobile MPG Study

The ADAPTIVEREG Procedure

Model Information

Data Set	WORK.AUTOMPG
Response Variable	MPG
Class Variables	Cylinders Year Origin
Distribution	Normal
Link Function	Identity

Fit Controls

Maximum Number of Bases	21
Maximum Order of Interaction	1
Degrees of Freedom per Knot	2
Knot Separation Parameter	0.05
Penalty for Variable Reentry	0
Missing Value Handling	Include

In addition to listing classification variables in the "Model Information" table, PROC ADAPTIVEREG displays level information about the classification variables that are specified in the CLASS statement. The table in Figure 24.2 lists the levels of the classification variables Cylinders, Year, and Origin. Although the values of Cylinders and Year are naturally ordered, they are treated as ordinary classification variables.

Figure 24.2 Class Level Information

Class Level Information

Class	Levels	Values
Cylinders	5	3 4 5 6 8
Year	13	70 71 72 73 74 75 76 77 78 79 80 81 82
Origin	3	1 2 3

The "Fit Statistics" table (Figure 24.3) lists summary statistics of the fitted regression spline model. Because the final model is essentially a linear model, several fit statistics are reported as if the model were fitted with basis functions as predetermined effects. However, because the model selection process and the determination of basis functions are highly nonlinear, additional statistics that incorporate the extra source of degrees of freedom are also displayed. The statistics include effective degrees of freedom, the generalized cross validation (GCV) criterion, and the GCV R-square value.

Figure 24.3 Fit Statistics

Fit Statistics

GCV	11.55804
GCV R-Square	0.81128
Effective Degrees of Freedom	23
R-Square	0.83161
Adjusted R-Square	0.82682
Mean Square Error	10.57977
Average Square Error	10.26079

The "Parameter Estimates" table (Figure 24.4) displays both parameter estimates for constructed basis functions and each function's construction components. The basis functions are constructed as two-way interaction terms from parent basis functions and transformations of variables. For continuous variables, the transformations are linear spline functions with knot values specified in the Knot column. For classification variables, the transformations are formed by dichotomizing the variables based on levels specified in the Levels column.

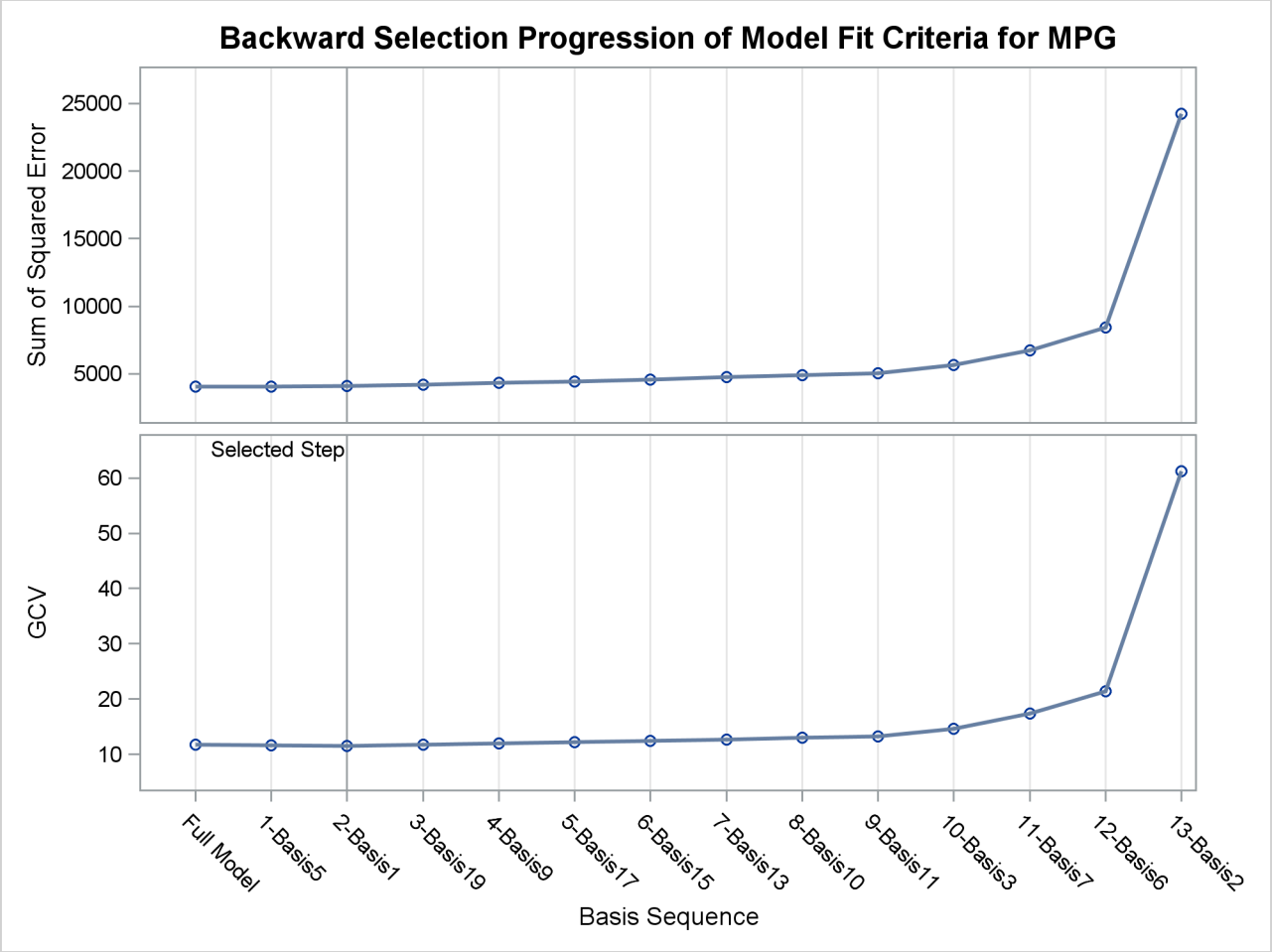
Figure 24.4 Parameter Estimates

Regression Spline Model after Backward Selection

Name	Coefficient	Parent	Variable	Knot	Levels
Basis0	29.4394		Intercept		
Basis2	0.004412	Basis0	Weight	3139.00	
Basis3	-21.2899	Basis0	Horsepower	.	
Basis6	0.1534	Basis3	Horsepower	158.00	
Basis7	2.3920	Basis3	Year		10 12 11 9 8 7 3
Basis9	1.6658	Basis0	Acceleration	21.0000	
Basis10	0.4672	Basis0	Acceleration	21.0000	
Basis11	-8.1766	Basis0	Cylinders		0 3
Basis13	-10.0976	Basis4	Origin		0
Basis15	2.1354	Basis0	Origin		2
Basis17	6.7675	Basis0	Cylinders		3
Basis19	1.4987	Basis0	Year		3 10 12 11 9

During the model construction and selection process, some basis function terms are removed. You can view the backward elimination process in the selection plot (Figure 24.5). The plot displays how the model sum of squared error and the corresponding GCV criterion change along with the backward elimination process. The sum of squared error increases as more basis functions are removed from the full model. The GCV criterion decreases at first when two basis functions are dropped and increases afterward. The vertical line indicates the selected model that has the minimum GCV value.

Figure 24.5 Selection Plot



The formed model is an additive model. Basis functions of same variables can be grouped together to form functional components. The "ANOVA Decomposition" table (Figure 24.6) shows functional components and their contribution to the final model.

Figure 24.6 ANOVA Decomposition

ANOVA Decomposition

Functional Component	Number of Bases	DF	---Change If Omitted--- Lack of Fit	GCV
Weight	1	2	299.55	0.7165
Horsepower	1	2	1324.81	3.5875
Year	2	4	1183.22	3.0358
Acceleration	2	4	287.76	0.5546
Cylinders	2	4	321.11	0.6470
Origin	2	4	316.04	0.6330

Another criterion that focuses on the contribution of each individual variable is variable importance. It is defined to be the square root of the GCV value of a submodel from which all basis functions that involve a variable have been removed, minus the square root of the GCV value of the selected model, then scaled to have the largest importance value, 100. The table in Figure 24.7 lists importance values, sorted in descending order, for the variables that compose the selected model.

Figure 24.7 Variable Importance

Variable Importance

Variable	Number of Bases	Importance
Horsepower	1	100.00
Year	2	85.46
Weight	1	21.10
Cylinders	2	19.08
Origin	2	18.67
Acceleration	2	16.38

The component panel (Figure 24.8) displays the fitted functional components against their forming variables.

Figure 24.8 Component Panel

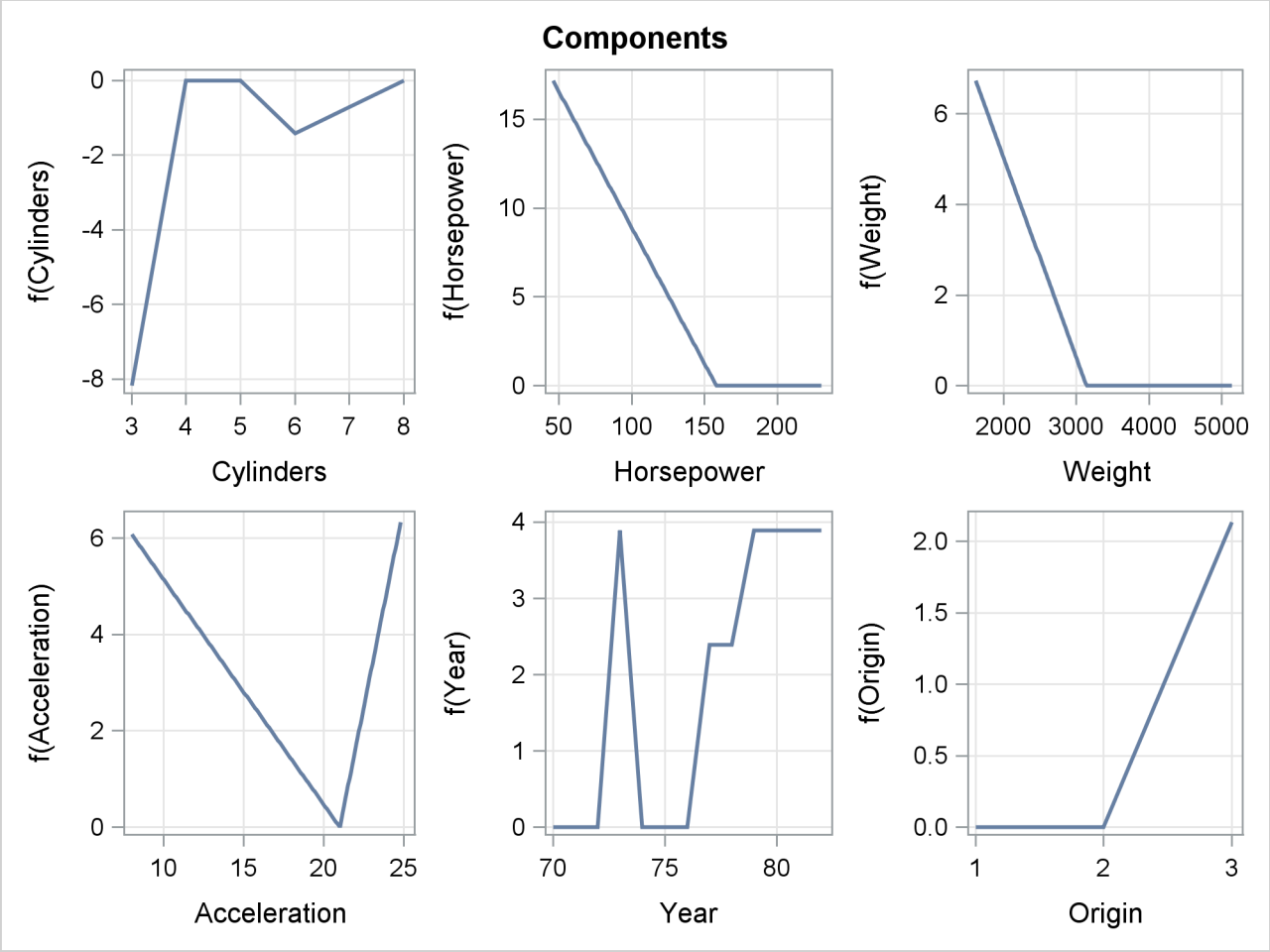
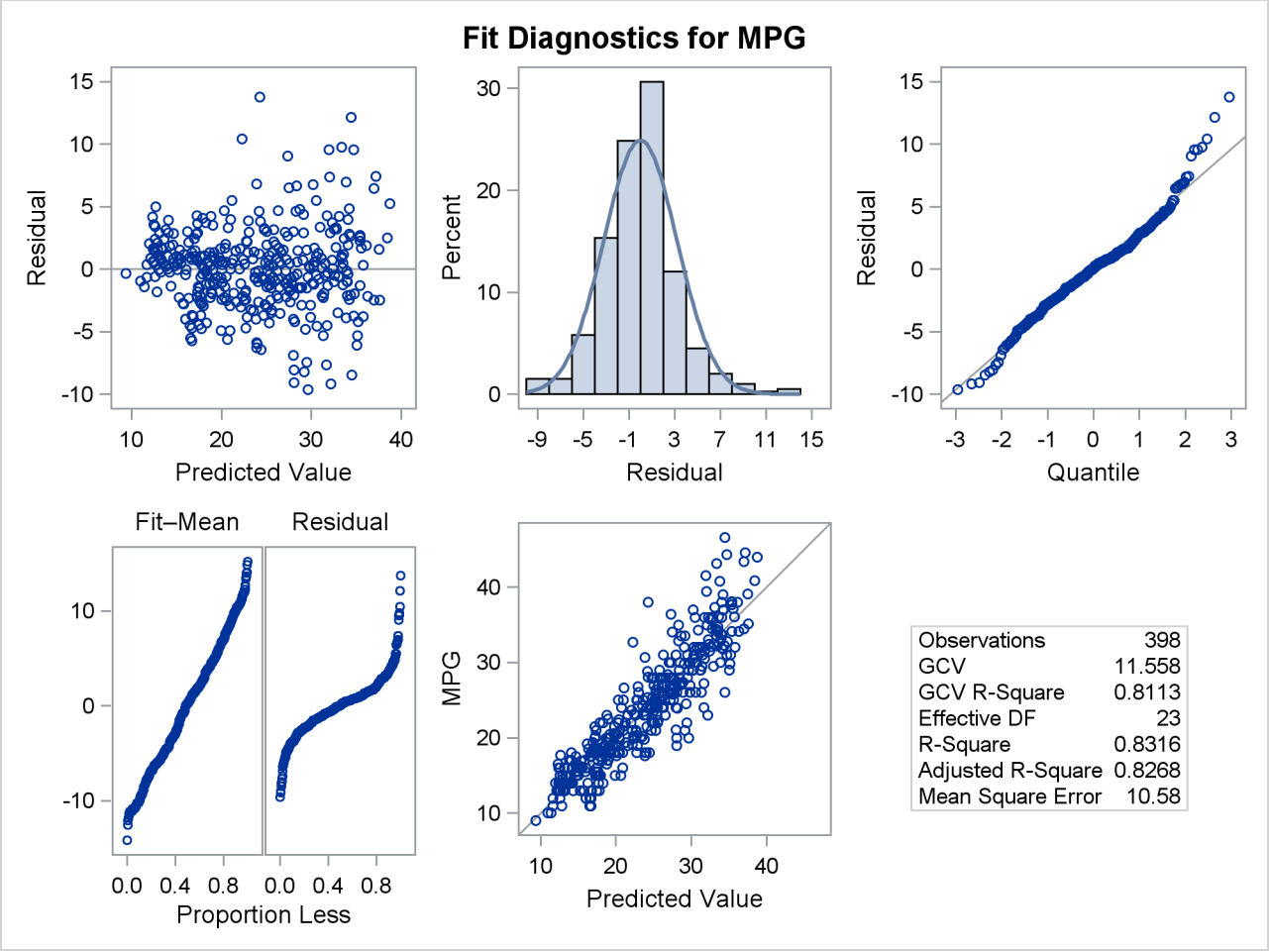


Figure 24.9 shows a panel of fit diagnostics for the selected model that indicate a reasonable fit.

PROC ADAPTIVEREG provides an adaptive way to fit parsimonious regression spline models. The nonparametric transformation of variables is automatically determined, and model selection methods are used to reduce model complexity. The final model based on piecewise linear splines is easy to interpret and highly portable. It can also be used to suggest parametric forms based on the nonlinear trend.

Figure 24.9 Diagnostics Panel



Syntax: ADAPTIVEREG Procedure

The following statements are available in the ADAPTIVEREG procedure:

PROC ADAPTIVEREG < *options* > **;**
BY *variables* **;**
CLASS *variables* < / *options* > **;**
FREQ *variable* **;**
MODEL *dependent* < *(options)* > **=** < *effects* >< / *options* > **;**
OUTPUT < **OUT=***SAS-data-set* > < *keyword* < **(***keyword-options* **)** > < **=***name* > > . . .
< *keyword* < **(***keyword-options* **)** > < **=***name* > > **;**
PARTITION < *options* > **;**
SCORE < **DATA=***SAS-data-set* > < **OUT=***SAS-data-set* >
< *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > **;**
WEIGHT *variable* **;**

The syntax of the ADAPTIVEREG procedure is similar to that of other regression procedures in the SAS System. The PROC ADAPTIVEREG and MODEL statements are required, and the MODEL statement must appear after the CLASS statement if a CLASS statement is included. The SCORE statement can appear multiple times; all other statements can appear only once.

The following sections describe the PROC ADAPTIVEREG statement and then describe the other statements in alphabetical order.

PROC ADAPTIVEREG Statement

PROC ADAPTIVEREG < *options* > **;**

The PROC ADAPTIVEREG statement invokes the procedure.

Table 24.1 summarizes the options available in the PROC ADAPTIVEREG statement.

Table 24.1 PROC ADAPTIVEREG Statement Options

Option	Description
Data Set Options	
DATA=	Specifies the input SAS data set
TESTDATA=	Names a data set that contains test data
VALDATA=	Names a data set that contains validation data
Computational Options	
NLOPTIONS	Sets optimization parameters for fitting generalized linear models
SINGULAR=	Sets the singularity tolerance
Display Options	
NAMELEN=	Sets the length of effect names in tables and output data sets
PLOTS=	Controls plots produced through ODS Graphics
DETAILS=	Displays detailed modeling information
Other Options	
NOTHREADS	Requests the computation in single-threaded mode
OUTDESIGN=	Requests a data set that contains the design matrix
SEED=	Sets the seed used for pseudo-random number generation
THREADS=	Specifies the number of threads for the computation

You can specify the following *options*.

DATA=*SAS-data-set*
specifies the SAS data set to be read by PROC ADAPTIVEREG. If you do not specify the DATA= option, PROC ADAPTIVEREG uses the most recently created SAS data set.

DETAILS< =(*detail-options***) >**
requests detailed model fitting information. You can specify the following *detail-options*:

BASES
displays the "Bases Information" table.

BWDSUMMARY
displays the "Backward Selection Summary" table.

FWDSUMMARY
displays the "Forward Selection Summary" table.

FWDPARAMS
displays the "Forward Selection Parameter Estimates" table.

If you do not specify a *detail-option*, PROC ADAPTIVEREG produces all the preceding tables by default.

NAMELEN=*number*
specifies the length to which long effect names are shortened. The default and minimum value is 20.

NLOPTIONS(*options***)**
specifies options for the nonlinear optimization methods if you are applying the multivariate adaptive regression splines algorithm to generalized linear models. You can specify the following *options*:

ABSCONV=*r*

ABSTOL=*r*
specifies an absolute function convergence criterion by which minimization stops when $f(\boldsymbol{\psi}^{(k)}) \leq r$, where $\boldsymbol{\psi}$ is the vector of parameters in the optimization and $f(\cdot)$ is the objective function. The default value of *r* is the negative square root of the largest double-precision value, which serves only as a protection against overflows.

ABSFCONV=*r*

ABSFTOL=*r*
specifies an absolute function difference convergence criterion. For all techniques except NMSIMP, termination requires a small change of the function value in successive iterations,

$$|f(\boldsymbol{\psi}^{(k-1)}) - f(\boldsymbol{\psi}^{(k)})| \leq r$$

where $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization and $f(\cdot)$ is the objective function. The same formula is used for the NMSIMP technique, but $\boldsymbol{\psi}^{(k)}$ is defined as the vertex with the lowest function value, and $\boldsymbol{\psi}^{(k-1)}$ is defined as the vertex with the highest function value in the simplex. The default value is *r*=0.

ABSGCONV=*r*

ABSGTOL=*r*
specifies an absolute gradient convergence criterion. Termination requires the maximum absolute gradient element to be small,

$$\max_j |g_j(\boldsymbol{\psi}^{(k)})| \leq r$$

where $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization and $g_j(\cdot)$ is the gradient of the objective function with respect to the *j*th parameter. This criterion is not used by the NMSIMP technique. The default value is *r* = 1E–5.

FCONV=*r*

FTOL=*r*

specifies a relative function convergence criterion. For all techniques except NMSIMP, termination requires a small relative change of the function value in successive iterations,

$$\frac{|f(\boldsymbol{\psi}^{(k)}) - f(\boldsymbol{\psi}^{(k-1)})|}{|f(\boldsymbol{\psi}^{(k-1)})|} \leq r$$

where $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization and $f(\cdot)$ is the objective function. The same formula is used for the NMSIMP technique, but $\boldsymbol{\psi}^{(k)}$ is defined as the vertex with the lowest function value, and $\boldsymbol{\psi}^{(k-1)}$ is defined as the vertex with the highest function value in the simplex. The default is $r = 10^{-\mathrm{FDIGITS}}$, where FDIGITS is by default $-\log_{10}\{\epsilon\}$ and ϵ is the machine precision.

GCONV=*r*

GTOL=*r*

specifies a relative gradient convergence criterion. For all techniques except CONGRA and NMSIMP, termination requires the normalized predicted function reduction to be small,

$$\frac{\mathbf{g}(\boldsymbol{\psi}^{(k)})'[\mathbf{H}^{(k)}]^{-1}\mathbf{g}(\boldsymbol{\psi}^{(k)})}{|f(\boldsymbol{\psi}^{(k)})|} \leq r$$

where $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, $f(\cdot)$ is the objective function, and $\mathbf{g}(\cdot)$ is the gradient. For the *CONGRA* technique (where a reliable Hessian estimate $\mathbf{H}$ is not available), the following criterion is used:

$$\frac{\| \mathbf{g}(\boldsymbol{\psi}^{(k)}) \|_2^2 \quad \| \mathbf{s}(\boldsymbol{\psi}^{(k)}) \|_2}{\| \mathbf{g}(\boldsymbol{\psi}^{(k)}) - \mathbf{g}(\boldsymbol{\psi}^{(k-1)}) \|_2 \, |f(\boldsymbol{\psi}^{(k)})|} \leq r$$

This criterion is not used by the NMSIMP technique. The default value is r = 1E–8.

HESSIAN=*hessian-options*

specifies the Hessian matrix type used in the optimization of likelihood functions, if the Newton-Raphson technique is used. You can specify the following *hessian-options*:

EXPECTED

requests that the Hessian matrix in optimization be computed as the negative of the expected information matrix.

OBSERVED

requests that the Hessian matrix in optimization be computed as the negative of the observed information matrix. For many specified distribution families and link functions, the observed information matrix is equal to the expected information matrix.

The default is HESSIAN=EXPECTED.

MAXFUNC=*n*

MAXFU=*n*

specifies the maximum number of function calls in the optimization process. The default values are as follows, depending on the optimization technique:

- TRUREG, NRRIDG, and NEWRAP: 125
- QUANEW and DBLDOG: 500
- CONGRA: 1000
- NMSIMP: 3000

The optimization can terminate only after completing a full iteration. Therefore, the number of function calls that are actually performed can exceed the number that is specified by this option. You can select the optimization technique by specifying the TECHNIQUE= option.

MAXITER=*n*

MAXIT=*n*

specifies the maximum number of iterations in the optimization process. The default values are as follows, depending on the optimization technique:

- TRUREG, NRRIDG, and NEWRAP: 50
- QUANEW and DBLDOG: 200
- CONGRA: 400
- NMSIMP: 1000

These default values also apply when *n* is specified as a missing value. You can select the optimization technique by specifying the TECHNIQUE= option.

MAXTIME=*r*

specifies an upper limit of *r* seconds of CPU time for the optimization process. The default value is the largest floating-point double representation of your computer. The time that is specified by the MAXTIME= option is checked only once at the end of each iteration. Therefore, the actual running time can be longer than the time specified by this option.

MINITER=*n*

MINIT=*n*

specifies the minimum number of iterations. The default value is 0. If you request more iterations than are actually needed for convergence to a stationary point, the optimization algorithms can behave strangely. For example, the effect of rounding errors can prevent the algorithm from continuing for the required number of iterations.

TECHNIQUE=*keyword*

specifies the optimization technique to obtain maximum likelihood estimates for nonnormal distributions. You can choose from the following techniques by specifying the appropriate *keyword*:

CONGRA	performs a conjugate-gradient optimization.
DBLDOG	performs a version of double-dogleg optimization.
NEWRAP	performs a Newton-Raphson optimization that combines a line-search algorithm with ridging.

NMSIMP	performs a Nelder-Mead simplex optimization.
NONE	performs no optimization.
NRRIDG	performs a Newton-Raphson optimization with ridging.
QUANEW	performs a dual quasi-Newton optimization.
TRUREG	performs a trust-region optimization.

The default is TECHNIQUE=NEWRAP.

For more information about these optimization methods, see the section "Choosing an Optimization Algorithm" on page 494 in Chapter 19, "Shared Concepts and Topics."

NOTHREADS
forces single-threaded execution of the analytic computations. This overrides the SAS system option THREADS | NOTHREADS. Specifying this option is equivalent to specifying the THREADS=1 option.

OUTDESIGN<(*options***)>=***SAS-data-set*
creates a data set that contains the design matrix of constructed basis functions. The design matrix column names consist of a prefix followed by an index. The default naming prefix is _X. The default output is the design matrix of basis functions after backward selection.

You can specify the following *options* in parentheses to control the content of the OUTDESIGN= data set:

BACKWARDMODEL | BACKWARD
produces the design matrix for the selected model after the backward selection.

FORWARDMODEL | FORWARD
produces the design matrix for the selected model after the forward selection.

PREFIX=*prefix*
requests that the design matrix column names consist of a prefix followed by an index.

STARTMODEL
produces the design matrix for the initial model specified in the MODEL statement.

PLOTS <(*global-plot-options***)> <= *plot-request* <(***options***)> >**

PLOTS <(*global-plot-options***)> <= (***plot-request* <(***options***)> <... *plot-request* <(***options***)> >>)>**
controls the plots produced through ODS Graphics. When you specify only one *plot-request*, you can omit the parentheses around the *plot-request*. For example:

```
plots=all
plots=components(unpack)
plots(unpack)=(components diagnostics)
```

ODS Graphics must be enabled before plots can be requested. For example:

```
ods graphics on;

proc adaptivereg plots=all;
   model y=x1 x2;
run;

ods graphics off;
```

For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" on page 600 in Chapter 21, "Statistical Graphics Using ODS."

You can specify the following *global-plot-option*, which applies to all plots that the ADAPTIVEREG procedure generates:

UNPACK | UNPACKPANEL
suppresses paneling. By default, multiple plots can appear in some output panels. Specify UNPACK to get each plot individually. You can also specify UNPACK as a suboption with COMPONENTS and DIAGNOSTICS.

You can specify the following *plot-requests* and their options:

ALL
requests that all default plots be produced.

COMPONENTS < (*component-options*) >
plots a panel of functional components of the fitted model. You can specify the following *component-options*:

COMMONAXES
specifies that the functional component plots use a common vertical axis except for contour plots. This enables you to visually judge relative effect size.

UNPACK | UNPACKPANEL
displays the component plots individually.

DIAGNOSTICS < (UNPACK | UNPACKPANEL) >
produces a summary panel of fit diagnostics that consists of the following:

- residuals versus the predicted values
- a histogram of the residuals
- a normal quantile plot of the residuals
- a residual-fit (RF) plot that consists of side-by-side quantile plots of the centered fit and the residuals
- response values versus the predicted values

You can request the five plots in this panel as individual plots by specifying the UNPACK suboption. The fit diagnostics panel is not produced for dependent variable with nonnormal distributions.

FIT <(NODATA | NOOBS)>
produces a plot of the predicted values against the variables that form the selected model. By default, a scatter plot of the input data is overlaid. You can suppress the scatter plot by specifying the NODATA | NOOBS option.

The plot is not produced if the number of variables in the selected model exceeds two. The plot is not produced for dependent variables with nonnormal distributions.

NONE
suppresses all plots.

SELECTION<(*selection-panel-options*)>
plots a panel of model fit criteria. The panel consists of two plots. The upper plot shows the progression of the model lack-of-fit criterion as the selection process proceeds. The lower plot shows the progression of the model validation criterion as the selection process proceeds. By default, the selection panel shows the progression for the backward selection process. You can specify the following *selection-panel-options*:

BACKWARDMODEL | BACKWARD
displays the progression of model fit criteria for the backward selection process.

FORWARDMODEL | FORWARD
displays the progression of model fit criteria for the forward selection process.

SEED=*number*
specifies an integer used to start the pseudorandom number generator for random cross validation and random partitioning of data for training, testing, and validation. If you do not specify a seed, or if you specify a value less than or equal to 0, the seed is generated from the time of day, which is read from the computer's clock.

SINGULAR=*number*

EPSILON=*number*
sets the tolerance for testing singularity of the $\mathbf{X}'\mathbf{W}\mathbf{X}$ matrix that is formed from the design matrix $\mathbf{X}$. Roughly, the test requires that a pivot be at least this number times the original diagonal value. By default, *number* is 10^7 times the machine epsilon. The default *number* is approximately 10^{-9} on most machines.

TESTDATA=*SAS-data-set*
names a SAS data set that contains test data. This data set must contain all the variables specified in the MODEL statement. Furthermore, when a BY statement is used and the TESTDATA=data set contains any of the BY variables, then the TESTDATA= data set must also contain all the BY variables sorted in the order of the BY variables. In this case, only the test data for a specific BY group are used with the corresponding BY group in the analysis data. If the TESTDATA= data set contains none of the BY variables, then the entire TESTDATA = data set is used with each BY group of the analysis data.

If you specify a TESTDATA= data set, then you cannot also specify a PARTITION statement to reserve observations for testing.

THREADS=*n*

specifies the number of threads for analytic computations and overrides the SAS system option THREADS | NOTHREADS. If you do not specify the THREADS= option or if you specify THREADS=0, the number of threads is determined based on the data size and the number of CPUs on the host on which the analytic computations execute. If the specified number of threads is more than the number of actual CPUs, PROC ADAPTIVEREG by default sets the value to the number of actual CPUs.

VALDATA=*SAS-data-set*

names a SAS data set that contains validation data. This data set must contain all the variables specified in the MODEL statement. Furthermore, when a BY statement is used and the VALDATA= data set contains any of the BY variables, then the VALDATA= data set must also contain all the BY variables sorted in the order of the BY variables. In this case, only the validation data for a specific BY group are used with the corresponding BY group in the analysis data. If the VALDATA= data set contains none of the BY variables, then the entire VALDATA = data set is used with each BY group of the analysis data.

If you specify a VALDATA= data set, then you cannot also specify a PARTITION statement to reserve observations for validation.

BY Statement

BY *variables* **;**

You can specify a BY statement with PROC ADAPTIVEREG to obtain separate analyses of observations in groups that are defined by the BY variables. When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables. If you specify more than one BY statement, only the last one specified is used.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data by using the SORT procedure with a similar BY statement.
- Specify the NOTSORTED or DESCENDING option in the BY statement for the ADAPTIVEREG procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.
- Create an index on the BY variables by using the DATASETS procedure (in Base SAS software).

For more information about BY-group processing, see the discussion in *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see the discussion in the *Base SAS Procedures Guide*.

CLASS Statement

CLASS *variables* < / *options* > **;**

The CLASS statement names the classification *variables* to be used in the analysis. Typical class variables are Treatment, Sex, Race, Group, and Replication. If the CLASS statement is used, it must appear before the MODEL statement.

Classification variables can be either character or numeric. Class levels are determined from the formatted values of the *variables*. Thus, you can use formats to group values into levels. See the discussion of the FORMAT procedure in the *Base SAS Procedures Guide* and the discussions of the FORMAT statement and SAS formats in *SAS Formats and Informats: Reference*.

You can specify the following *options* for classification *variables*:

DESCENDING
DESC
reverses the sort order of the classification variable. If you specify both the DESCENDING and ORDER= options, PROC ADAPTIVEREG orders the categories according to the ORDER= option and then reverses that order.

ORDER=*order-type*
specifies the sort order for the categories of categorical variables. This ordering determines which parameters in the model correspond to each level in the data. When the default ORDER=FORMATTED is in effect for numeric variables for which you have supplied no explicit format, the levels are ordered by their internal values. Table 24.2 shows how PROC ADAPTIVEREG interprets values of the ORDER= option.

Table 24.2 Sort Order for Categorical Variables

order-type	**Levels Sorted By**
DATA	Order of appearance in the input data set
FORMATTED	External formatted value, except for numeric variables with no explicit format, which are sorted by their unformatted (internal) value
FREQ	Descending frequency count; levels with the most observations come first in the order
FREQDATA	Order of descending frequency count, and within counts by order of appearance in the input data set when counts are tied
FREQFORMATTED	Order of descending frequency count, and within counts by formatted value (as above) when counts are tied
FREQINTERNAL	Order of descending frequency count, and within counts by unformatted value when counts are tied
INTERNAL	Unformatted value

For the FORMATTED and INTERNAL values, the sort order is machine-dependent. If you specify the ORDER= option in the MODEL statement and the ORDER= option in the CLASS statement, the former takes precedence.

For more information about sort order, see the chapter on the SORT procedure in the *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

FREQ Statement

FREQ *variable* **;**

The FREQ statement names a variable that provides frequencies for each observation in the DATA= data set. Specifically, if n is the value of the FREQ variable for a given observation, then that observation is used n times.

The analysis produced using a FREQ statement reflects the expanded number of observations. You can produce the same analysis without the FREQ statement by first creating a new data set that contains the expanded number of observations. For example, if the value of the FREQ variable is 5 for the first observation, the first five observations in the new data set are identical. Each observation in the old data set is replicated n_i times in the new data set, where n_i is the value of the FREQ variable for that observation.

If the value of the FREQ variable is missing or is less than 1, the observation is not used in the analysis. If the value is not an integer, only the integer portion is used.

MODEL Statement

MODEL *dependent* < *(options)* >**=**< *effects* > < / *options* > **;**

MODEL *events/trials* **=** < *effects* > < / *options* > **;**

The MODEL statement names the response variable and the explanatory effects, including covariates, main effects, interactions, and nested effects; see the section "Specification of Effects" on page 3324 for more information. If you omit the explanatory effects, the procedure fits an intercept-only model. You must specify exactly one MODEL statement.

You can specify two forms of the MODEL statement. The first form, referred to as *single-trial* syntax, is applicable to binary, ordinal, and nominal response data. The second form, referred to as *events/trials* syntax, is restricted to binary response data. You use the *single-trial* syntax when each observation in the DATA= data set contains information about only a single trial, such as a single subject in an experiment. When each observation contains information about multiple binary response trials, such as the counts of the number of observed subjects and the number of subjects who respond, then you can use the *events/trials* syntax.

In the *events/trials* syntax, you specify two variables that contain count data for a binomial experiment. These two variables are separated by a slash. The value of the first variable, *events*, is the number of positive responses (or events). The value of the second variable, *trials*, is the number of trials. The values of both *events* and (*trials*–*events*) must be nonnegative and the value of *trials* must be positive for the response to be valid.

In the single-trial syntax, you specify one variable (on the left side of the equal sign) as the response variable. This variable can be character or numeric. You can specify variable options specific to the response variable immediately after the response variable with parentheses around them.

For both forms of the MODEL statement, explanatory *effects* follow the equal sign. Variables can be either continuous or classification variables. Classification variables can be character or numeric, and they must be declared in the CLASS statement. When an *effect* is a classification variable, the procedure inserts a set of coded columns into the design matrix instead of directly entering a single column that contains the values of the variable.

Table 24.3 summarizes the *options* available in the MODEL statement.

Table 24.3 MODEL Statement Options

Option	Description
Response Variable Options	
DESCENDING	Reverses the order of the response categories
EVENT=	Specifies the event category for the binary response
ORDER=	Specifies the sort order for the categorical response
REFERENCE=	Specifies the reference category for the categorical response
Statistical Modeling Options	
ADDITIVE	Requests an additive model
ALPHA	Controls the knot selection
CVMETHOD=	Specifies how subsets for cross validation are formed
DFPERBASIS	Specifies degrees of freedom per basis function
DIST=	Specifies the distribution family
FAST	Controls the fast-forward selection algorithm
FORWARDONLY	Requests that the backward selection process be skipped
KEEP=	Specifies effects to be included in the final model
LINEAR=	Specifies linear effects to be examined in model selection
LINK=	Specifies the link function
MAXBASIS=	Specifies the maximum number of basis functions allowed
MAXORDER=	Specifies the maximum order of interactions allowed
NOMISS	Requests removal of missing values from modeling
OFFSET=	Specifies an offset for the linear predictor
VARPENALTY=	Specifies the penalty for variable reentry

You can specify the following *options* in the MODEL statement.

Response Variable Options

Response variable options determine how the ADAPTIVEREG procedure models probabilities for binary data. You can specify the following response variable options by enclosing them in parentheses after the response variable.

DESCENDING

DESC

reverses the order of the response categories. If both the DESCENDING and ORDER= options are specified, PROC ADAPTIVEREG orders the response categories according to the ORDER= option and then reverses that order.

EVENT='*category*' | **FIRST** | **LAST**

specifies the event category for the binary response model. PROC ADAPTIVEREG models the probability of the event category. You can specify one of the following values for this option:

'*category*' specifies the formatted value of the reference category.

FIRST designates the first ordered category as the event.

LAST designates the last ordered category as the event.

The default is EVENT=FIRST.

One of the most common sets of response levels is $\{0, 1\}$, with 1 representing the event for which the probability is to be modeled. Consider the example where Y takes the value 1 for event and 0 for nonevent, and X is the explanatory variable. To specify the value 1 as the event category, use the following MODEL statement:

```
model Y (event='1') = X;
```

ORDER=*order-type*

specifies the sort order for the categories of categorical variables. This ordering determines which parameters in the model correspond to each level in the data. When the default ORDER=FORMATTED is in effect for numeric variables for which you have supplied no explicit format, the levels are ordered by their internal values. Table 24.4 shows how PROC ADAPTIVEREG interprets values of the ORDER= option.

Table 24.4 Sort Order for Categorical Variables

order-type	**Levels Sorted By**
DATA	Order of appearance in the input data set
FORMATTED	External formatted value, except for numeric variables with no explicit format, which are sorted by their unformatted (internal) value
FREQ	Descending frequency count; levels with the most observations come first in the order
FREQDATA	Order of descending frequency count, and within counts by order of appearance in the input data set when counts are tied
FREQFORMATTED	Order of descending frequency count, and within counts by formatted value (as above) when counts are tied
FREQINTERNAL	Order of descending frequency count, and within counts by unformatted value when counts are tied
INTERNAL	Unformatted value

For the FORMATTED and INTERNAL values, the sort order is machine-dependent. If you specify the ORDER= option in the MODEL statement and the ORDER= option in the CLASS statement, the former takes precedence.

For more information about sort order, see the chapter on the SORT procedure in the *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

REFERENCE='*category*' | **FIRST** | **LAST**

REF='*category*' | **FIRST** | **LAST**

specifies the reference category for the binary or multinomial response model. For the binary response model, specifying one response category as the reference is the same as specifying the other response category as the event category. You can specify one of the following values for this option:

'*category*' specifies the formatted value of the reference category.

FIRST designates the first ordered category as the reference.

LAST designates the last ordered category as the reference.

The default is REFERENCE=LAST.

Model Options

You can specify the following model *options*.

ADDITIVE

requests an additive model for which only main effects are included in the fitted model. If you do not specify the ADDITIVE option, PROC ADAPTIVEREG fits a model that has both main effects and two-way interaction terms.

ALPHA=*number*

specifies the parameter that controls the number of knots considered for each variable. Friedman (1991b) uses the following as the number of observations between interior knots:

$$-\frac{2}{5}\log_2\left[-\frac{\log(1-\alpha)}{pn_m}\right]$$

Friedman also uses the following as the number of observations between extreme knots and the corresponding variable boundary values,

$$3-\log_2\frac{\alpha}{p}$$

where p is the number of variables and n_m is the number of observations for which a parent basis $\mathbf{B}_m > 0$. The value of α should be greater than 0 and less than 1. The default is ALPHA=0.05.

CVMETHOD=RANDOM <(*n***)>**

CVMETHOD=INDEX (*variable***)**

specifies the method for subdividing the training data into *n* parts when you request *n*-fold cross validation when you do backward selection. CVMETHOD=RANDOM assigns each training observation randomly to one of the *n* parts. CVMETHOD=INDEX(*variable*) assigns observations to parts based on the formatted value of the named *variable*. This input data set variable is treated as a classification variable, and the number of parts *n* is the number of distinct levels of this variable. By optionally naming this variable in a CLASS statement, you can use the ORDER= option in the CLASS statement to control how this variable is levelized.

The value of *n* defaults to 5 with CVMETHOD=RANDOM.

DFPERBASIS=*d*

DF=*d*

specifies the degrees of freedom (*d*) that are "charged" for each basis function that is used in the lack-of-fit function for backward selection. Larger values of *d* lead to fewer spline knots and thus smoother function estimates. The default is DFPERBASIS=2.

DIST=*distribution-id*

specifies the distribution family used in the model.

If you do not specify a *distribution-id*, the ADAPTIVEREG procedure defaults to the normal distribution for continuous response variables and to the binary distribution for classification or character variables, unless the *events/trial* syntax is used in the MODEL statement. If you choose the *events/trial* syntax, the ADAPTIVEREG procedure defaults to the binomial distribution.

Table 24.5 lists the values of the DIST= option and the corresponding default link functions. For generalized linear models with these distributions, you can find expressions for the log-likelihood functions in the section "Log-Likelihood Functions" on page 2802.

Table 24.5 Values of the DIST= Option

distribution-id	**Aliases**	**Distribution**	**Default Link Function**
BINOMIAL		Binomial	Logit
GAMMA	GAM, G	Gamma	Reciprocal
GAUSSIAN	NORMAL, N, NOR	Normal	Identity
IGAUSSIAN	IG	Inverse Gaussian	Inverse squared (power(–2))
NEGBIN	NB	Negative binomial	Log
POISSON	POI	Poisson	Log

FAST<(*fast-options***)>**

improves the speed of the modeling. Because of the computation complexity in the original multivariate adaptive regression splines algorithm, Friedman (1993) proposes modifications to improve the speed by tuning several parameters. See the section "Fast Algorithm" on page 874 for more information about the improvement of the multivariate adaptive regression splines algorithm. You can specify the following *fast-options*:

BETA=*beta*

specifies the "aging" factor in the priority queue of candidate parent bases. Larger values of *beta* result in low-improvement parents rising fast into top list of candidates. The default value is BETA=1.

H=*h*

specifies the parameter that controls how often the improvement is recomputed for a parent basis $\mathbf{B}_m$ over all candidate variables. Larger values of *h* cause fewer computations of improvement. The default value is H=1.

K=*k*

specifies the number of top candidates in the priority queue of parent bases for selecting new bases. Larger values of *k* cause more parent bases to be considered. The default is to use all eligible parent bases at every iteration.

FORWARDONLY

skips the backward selection step after forward selection is finished.

KEEP=*effects*

specifies a list of variables to be included in the final model.

LINEAR=*effects*

specifies a list of variables to be considered without nonparametric transformation. They should appear in the linear form if they are selected.

LINK=*keyword*

specifies the link function in the model. Not all link functions are available for all distribution families. The *keywords* and expressions for the associated link functions are shown in Table 24.6.

Table 24.6 Link Functions in MODEL Statement of the ADAPTIVEREG Procedure

keyword	**Alias**	**Link Function**	$g(\mu) = \eta =$
CLOGLOG	CLL	Complementary log-log	$\log(-\log(1-\mu))$
IDENTITY	ID	Identity	μ
LOG		Log	$\log(\mu)$
LOGIT		Logit	$\log(\mu/(1-\mu))$
POWERMINUS2		Power with exponent –2	$1/\mu^2$
PROBIT	NORMIT	Probit	$\Phi^{-1}(\mu)$
RECIPROCAL	INVERSE	Reciprocal	$1/\mu$

MAXBASIS=*number*

specifies the maximum number of basis functions ($M_{\max}$) that can be used in the final model. The default value is the larger value between 21 and one plus two times the number of nonintercept effects specified in the MODEL statement.

MAXORDER=*number*

specifies the maximum interaction levels for effects that could potentially enter the model. The default value is MAXORDER=2.

NOMISS

excludes all observations with missing values from the model fitting. By default, the ADAPTIVEREG procedure takes the missingness into account when an explanatory variable has missing values. For more information about how PROC ADAPTIVEREG handles missing values, see the section "Missing Values" on page 875.

OFFSET=*variable*

specifies an offset for the linear predictor. An offset plays the role of a predictor whose coefficient is known to be 1. For example, you can use an offset in a Poisson model when counts have been obtained in time intervals of different lengths. With a log link function, you can model the counts as Poisson variables with the logarithm of the time interval as the offset variable. The offset variable cannot appear in the CLASS statement or elsewhere in the MODEL statement.

VARPENALTY= γ

specifies the incremental penalty γ for increasing the number of variables in the adaptive regression model. To discourage a model with too many variables, at each iteration of the forward selection the model improvement is reduced by a factor of $(1-\gamma)$ for any new variable that is introduced.

For highly collinear designs, the VARPENALTY= option helps PROC ADAPTIVEREG produce models that are nearly equivalent in terms of residual sum of squares but have fewer independent variables. Friedman (1991b) suggests the following values for γ:

0.0	no penalty (default value)
0.05	moderate penalty
0.1	heavy penalty

The best value depends on the specific situation. Some experimenting with different values is usually required. You should use this option with care.

OUTPUT Statement

OUTPUT < **OUT=***SAS-data-set* > < *keyword* < **(***keyword-options* **)** > < **=***name* > > . . .
< *keyword* < **(***keyword-options* **)** > < **=***name* > > **;**

The OUTPUT statement creates a new SAS data set to contain diagnostic measures that are calculated for the selected model. If you do not specify a *keyword*, then the only diagnostic included is the predicted response.

All the variables in the original data set are included by the new data set, along with variables created in the OUTPUT statement. These new variables contain the values of a variety of statistics and diagnostic measures that are calculated for each observation in the data set. If you specify a BY statement, then a variable _BY_ that indexes the BY groups is included. For each observation, the value of _BY_ is the index of the BY group to which this observation belongs.

If you have requested n-fold cross validation, then a variable _CVINDEX_ is included in the output data set. For each observation that is used for model training, the value of _CVINDEX_ is i if that observation is omitted in forming the ith subset of the training data. See the CVMETHOD= for additional details. The value of _CVINDEX_ is 0 for all observations in the input data set that are not used for model training.

If you have partitioned the input data by using a PARTITION statement, then a character variable _ROLE_ is included in the output data set. For each observation the value of _ROLE_ is as follows:

ROLE	Observation Role
TEST	Testing
TRAIN	Training
VALIDATE	Validation

If you want to create a permanent SAS data set, you must specify a two-level name. For more information about permanent SAS data sets, see *SAS Language Reference: Concepts*.

Details about the specifications in the OUTPUT statement follow.

keyword < **(***keyword-options***)** >< **=***name* >
specifies the statistics to include in the output data set and optionally names the new variables that contain the statistics. You can use the *keyword-options* to control which type of a particular statistic to compute for generalized linear models. You can specify the following *keyword-options* for associated statistics:

ILINK	computes the prediction on the scale of the data $\hat{\mu} = g^{-1}(\hat{\eta})$.
RAW	requests the raw residual value $r = y - \hat{\eta}$.
PEARSON	requests the Pearson residual value $r = (y - \hat{\eta})/\sqrt{V(\hat{\mu})}$.
DEVIANCE	requests the deviance residual value $r = \text{sign}(y - \hat{\mu})\sqrt{\hat{d}^2}$.

You can specify a *keyword* for each desired statistic (see the following list of keywords), followed optionally by an equal sign, and a variable to contain the statistic.

If you specify *keyword*=*name*, the new variable that contains the requested statistic has the specified name. If you omit the optional =*name* after a *keyword*, then the new variable name is formed by default names.

You can specify the following *keywords* for the corresponding statistics:

PREDICTED | PRED | P requests predicted values. The default name is Pred.

RESIDUAL | RESID | R requests residuals, calculated as ACTUAL – PREDICTED. The default name is Resid.

OUT=*SAS-data-set*

specifies the name of the new data set to contain the diagnostic measures. If the OUT= option is omitted, the procedure uses the DATA*n* convention to name the output data set.

PARTITION Statement

PARTITION < *options* > **;**

The PARTITION statement specifies how observations in the input data set are logically partitioned into disjoint subsets for model training, validation, and testing. Either you can designate a variable in the input data set and a set of formatted values of that variable to determine the role of each observation, or you can specify proportions to use for random assignment of observations for each role.

An alternative to using a PARTITION statement is to provide a variable named _ROLE_ in the input data set to define roles of observations in the input data. If you specify a PARTITION statement, then the _ROLE_ variable is ignored if it is present in the input data set. If you do not specify a PARTITION statement and the input data do not contain a variable named _ROLE_, then all observations in the input data set are assigned to model training.

You can specify the following mutually exclusive *options*:

ROLEVAR=*variable* **(< TEST='***value***' > < TRAIN='***value***' > < VALIDATE='***value***' >)**
ROLE=*variable* **(< TEST='***value***' > < TRAIN='***value***' > < VALIDATE='***value***' >)**

names the *variable* in the input data set whose values are used to assign roles to each observation. The formatted values of this variable that are used to assign observations roles are specified in the TEST=, TRAIN=, and VALIDATE= suboptions. If you do not specify the TRAIN= suboption, then all observations whose role is not determined by the TEST= or VALIDATE= suboptions are assigned to training. If you specify a TESTDATA= data set in the PROC ADAPTIVEREG statement, then you cannot also specify the TEST= suboption in the PARTITION statement. If you specify a VALDATA= data set in the PROC ADAPTIVEREG statement, then you cannot also specify the VALIDATE= suboption in the PARTITION statement.

FRACTION(< TEST=*fraction* **> < VALIDATE=***fraction* **>)**
randomly assigns training and validation roles to the observations in the input data according to the proportions that are specified by the fraction values in the TEST= and VALIDATE= suboptions. If you specify both the TEST= and VALIDATE= suboptions, then the sum of the specified fractions must be less than 1 and the remaining fraction of the observations are assigned to the training role. If you specify a TESTDATA= data set in the PROC ADAPTIVEREG statement, then you cannot also specify the TEST= suboption in the PARTITION statement. If you specify a VALDATA= data set in the PROC ADAPTIVEREG statement, then you cannot also specify the VALIDATE= suboption in the PARTITION statement.

SCORE Statement

SCORE < **DATA=***SAS-data-set* > < **OUT=***SAS-data-set* >
< *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > **;**

The SCORE statement creates a new SAS data set to contain predicted values and optionally residuals for data in a new data set that you name. If you do not specify a DATA= data set, then the input data are scored. If you want to predict multiple data sets, you can specify multiple SCORE statements. If you want to create a SAS data set in a permanent library, you must specify a two-level name. For more information about permanent libraries and SAS data sets, see *SAS Language Reference: Concepts*.

When you specify a BY statement, the DATA= data set must either contain all the BY variables sorted in the order of the BY variables or contain none of the BY variables. If the DATA= data set contains all the BY variables, then the model that is selected for a given BY group is used to score just the matching observations in that data set. If the DATA= set contains none of the BY variables, then the entire data set is scored for each BY group.

All observations in the DATA= data set are retained in the output data set. All the variables in the input data set are included in the output data set, along with variables that contain predicted values and optionally residuals.

You can specify the following arguments in the SCORE statement:

DATA=*SAS data set*
names the data set to be scored. If you omit this option, then the input data set that is named in the DATA= option in the PROC ADAPTIVEREG statement is scored.

keyword < **=***name* >
specifies the statistics to include in the output data set and optionally names the new variables that contain the statistics. Specify one of the following *keyword* for each desired statistic, followed optionally by an equal sign, and a variable to contain the statistic.

If you specify *keyword***=***name*, the new variable that contains the requested statistic has the specified name. If you omit the optional **=***name* after a *keyword*, then the new variable name is formed by using a prefix of one or more characters that identify the statistic, followed by an underscore (_), followed by the dependent variable name.

You can specify the following keywords, which represent the statistics shown:

PREDICTED | PRED | P includes predicted values in the output data set. The prefix for the default name is Pred.

RESIDUAL | RESID | R includes residuals (which are calculated as ACTUAL – PREDICTED), in the output data set. The prefix for the default name is Resid.

OUT=*SAS data set*
specifies the name of the new output data set. By default, PROC ADAPTIVEREG uses the DATA*n* convention to name the new data set.

WEIGHT Statement

WEIGHT *variable* **;**

When you specify a WEIGHT statement, each observation in the input data set is weighted by the value of *variable*. The value of *variable* can be nonintegral. Observations that have a negative, zero, or missing value for the WEIGHT variable are not used in model fitting.

Details: ADAPTIVEREG Procedure

Fitting Algorithms

The multivariate adaptive regression splines algorithm (Friedman 1991b) is a predictive modeling algorithm that combines nonparametric variable transformations with a recursive partitioning scheme.

The algorithm originates with Smith (1982), who proposes a nonparametric method that applies the model selection method (stepwise regression) to a large number of truncated power spline functions, which are evaluated at different knot values. This method constructs spline functions and selects relevant knot values automatically with the model selection method. However, the method is applicable only to problems in low dimensions. For multiple variables, the number of tensor products between spline basis functions is too large to fit even a single model. The multivariate adaptive regression splines algorithm avoids this situation by using forward selection to build the model gradually instead of using the full set of tensor products of spline basis functions.

Like the recursive partitioning algorithm, which has "growing" and "pruning" steps, the multivariate adaptive regression splines algorithm contains two stages: forward selection and backward selection. During the forward selection process, bases are created from interactions between existing parent bases and nonparametric transformations of continuous or classification variables as candidate effects. After the model grows to a certain size, the backward selection process begins by deleting selected bases. The deletion continues until the null model is reached, and then an overall the best model is chosen based on some goodness-of-fit criterion. The next three subsections give details about the selection process and methods of nonparametric transformation of variables. The fourth subsection describes how the multivariate adaptive regression splines algorithm is applied to fit generalized linear models. The fifth subsection describes the fast algorithm (Friedman 1993) for speeding up the fitting process.

Forward Selection

The forward selection process in the multivariate adaptive regression splines algorithm is as follows:

1. Initialize by setting $\mathbf{B}_0 = \mathbf{1}$ and $M = 1$.
2. Repeat the following steps until the maximum number of bases $M_{\max}$ has been reached or the model cannot be improved by any combination of $\mathbf{B}_m$, $\mathbf{v}$, and t.
   a) Set the lack-of-fit criterion $\mathrm{LOF}^* = \infty$.
   b) For each selected basis: $\mathbf{B}_m, m \in \{0, \ldots, M-1\}$ do the following for each variable $\mathbf{v}$ that $\mathbf{B}_m$ does not consist of $\mathbf{v} \notin \{\mathbf{v}(k,m) | 1 \le k \le K_m\}$
      i. For each knot value (or a subset of categories) t of $\mathbf{v} : t \in \{\mathbf{v} | \mathbf{B}_m > 0\}$, form a model with all currently selected bases $\sum_{i=0}^{M-1} \mathbf{B}_i$ and two new bases: $\mathbf{B}_m\mathbf{T}_1(\mathbf{v}, t)$ and $\mathbf{B}_m\mathbf{T}_2(\mathbf{v}, t)$.
      ii. Compute the lack-of-fit criterion for the new model LOF.
      iii. If $\mathrm{LOF} < \mathrm{LOF}^*$, then update $\mathrm{LOF}^* = \mathrm{LOF}$, $m^* = m$, $\mathbf{v}^* = \mathbf{v}$, and $t^* = t$.
   c) Update the model by adding two bases that improve the most $\mathbf{B}_{m^*}\mathbf{T}_1(\mathbf{v}^*, t^*)$ and $\mathbf{B}_{m^*}\mathbf{T}_2(\mathbf{v}^*, t^*)$.
   d) Set $M = M + 2$.

The essential part of each iteration is to search a combination of $\mathbf{B}_m$, $\mathbf{v}$, and t such that adding two corresponding bases most improve the model. The objective of the forward selection step is to build a model that overfits the data. The lack-of-fit criterion for linear models is usually the residual sum of squares (RSS).

Backward Selection

The backward selection process in the multivariate adaptive regression splines algorithm is as follows:

1. Initialize by setting the overall lack-of-fit criterion: $\mathrm{LOF}^* = \infty$.
2. Repeat the following steps until the null model is reached. The final model is the best one that is found during the backward deletion process.
   a) For a selected basis $\mathbf{B}_m, m \in \{1, \ldots, M\}$:
      i. Compute the lack-of-fit criterion, LOF, for a model that excludes $\mathbf{B}_m$.
      ii. If $\mathrm{LOF} < \mathrm{LOF}^*$, save the model as the best one. Let $m^* = m$.
      iii. Delete $\mathbf{B}_{m^*}$ from the current model.
   b) Set $M = M - 1$.

The objective of the backward selection is to "prune" back the overfitted model to find the best model that has good predictive performance. So the lack-of-fit criteria that characterize model loyalty to original data are not appropriate. Instead, the multivariate adaptive regression splines algorithm uses a quantity similar to the generalized cross validation criterion. See the section "Goodness-of-Fit Criteria" on page 873 for more information.

Variable Transformations

The type of transformation depends on the variable type:

- For a continuous variable, the transformation is a linear truncated power spline,

$$\mathbf{T}_1(\mathbf{v},t) = (v-t)_+ = \begin{cases} v-t, & \text{if } v > t \\ 0, & \text{otherwise} \end{cases}$$

$$\mathbf{T}_2(\mathbf{v},t) = [-(v-t)]_+ = \begin{cases} 0, & \text{if } v > t \\ t-v, & \text{otherwise} \end{cases}$$

  where t is a knot value for variable $\mathbf{v}$ and v is an observed value for $\mathbf{v}$. Instead of examining every unique value of $\mathbf{v}$, a series of knot values with a minimum span are used by assuming the smoothness of the underlying function. Friedman (1991b) uses the following formula to determine a reasonable number of counts between knots (span size). For interior knots, the span size is determined by

$$-\frac{2}{5}\log_2\left[-\frac{\log(1-\alpha)}{pn_m}\right]$$

  For boundary knots, the span size is determined by

$$3-\log_2\frac{\alpha}{p}$$

  where α is the parameter that controls the knot density, p is the number of variables, and n_m is the number of observations that a parent basis $\mathbf{B}_m > 0$.

- For a classification variable, the transformation is an indicator function,

$$\mathbf{T}_1(\mathbf{v},t) = \begin{cases} 1, & \text{if } v \in \{c_1,\ldots,c_t\} \\ 0, & \text{otherwise} \end{cases}$$

$$\mathbf{T}_2(\mathbf{v},t) = \begin{cases} 0, & \text{if } v \in \{c_1,\ldots,c_t\} \\ 1, & \text{otherwise} \end{cases}$$

  where $\{c_1,\ldots,c_t\}$ is a subset of all categories of variable $\mathbf{v}$. The smoothing is applied to categorical variables by assuming that subsets of categories tend to have similar properties, analogous to the assumption that a local neighborhood has close predictions for continuous variables.

  If a categorical variable has k distinct categories, then there are a total of $2^{k-1}-1$ possible subsets to consider. The computation cost is equal to all-subsets selection in regression, which is expensive for large k values. The multivariate adaptive regression splines algorithm use the stepwise selection method to select categories to form the subset $\{c_1,\ldots,c_t\}$. The method is still greedy, but it reduces computation and still yields reasonable final models.

Goodness-of-Fit Criteria

Like other nonparametric regression procedures, the multivariate adaptive regression splines algorithm can yield complicated models that involve high-order interactions in which many knot values or subsets are considered. Besides the basis functions, both the forward selection and backward selection processes are also highly nonlinear. Because of the trade-off between bias and variance, the complicated models that contain many parameters tend to have low bias but high variance. To select models that achieve good prediction performance, Craven and Wahba (1979) propose the widely used generalized cross validation criterion (GCV),

$$\mathrm{GCV} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{f}_i}{1 - \mathrm{trace}(\mathbf{S})/n}\right)^2 = \frac{\mathrm{RSS}}{n(1 - \mathrm{trace}(\mathbf{S})/n)^2}$$

where y is the response, $\hat{f}$ is an estimate of the underlying smooth function, and $\mathbf{S}$ is the smoothing matrix such that $\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}$. The effective degrees of freedom for the smoothing spline can be defined as $\mathrm{trace}(\mathbf{S})$. In the multivariate adaptive regression splines algorithm, Friedman (1991b) uses a similar quantity as the lack-of-fit criterion,

$$\mathrm{LOF} = \frac{\mathrm{RSS}}{n(1 - (M + d(M-1)/2)/n)^2}$$

where d is the degrees-of-freedom cost for each nonlinear basis function and M is total number of linearly independent bases in the model. Because any candidate model that is evaluated at each step of the multivariate adaptive regression splines algorithm is a linear model, M is actually the trace of the hat matrix. The only difference between the GCV criterion and the LOF criterion is the extra term $d(M-1)$. The corresponding effective degrees of freedom is defined as $M + d(M-1)/2$. The quantity d takes into account the extra nonlinearity in forming new bases, and it operates as a smoothing parameter. Larger values of d tend to result in smoother function estimates. Based on many practical experiments and some theoretic work (Owen 1991), Friedman suggests that the value of d is typically in the range of $[2, 4]$. For data that have complicated structures, the value of d could be much larger.

Alternatively, you can use the cross validation as the goodness-of-fit criterion or use a separate validation data set to select models and a separate testing data set to evaluate selected models.

Generalized Linear Models

Friedman (1991b) applies the multivariate adaptive regression splines algorithm to a logistic model by using the squared error loss between the response and inversely linked values in the goodness-of-fit criterion:

$$\sum_{i=1}^{n}\left(y_i - \frac{1}{1 + \exp(\mathbf{x}'\boldsymbol{\beta})}\right)^2$$

When a final model is obtained, the ordinary logistic model is fitted on selected bases. Some realizations of the multivariate adaptive regression splines algorithm ignore the distributional properties and derive model bases that are based on the least squares criterion. The reason to ignore the distributional properties or use least squares approximations is that examining the lack-of-fit criterion for each combination of $\mathbf{B}_m, \mathbf{v}$, and t is computationally formidable, because one generalized linear model fit involves multiple steps of weighted least squares. The ADAPTIVEREG procedure extends the multivariate adaptive regression splines algorithm to generalized linear models as suggested by Buja et al. (1991).

In the forward selection process, the ADAPTIVEREG procedure extends the algorithm in the following way. Suppose there are $(2k + 1)$ bases after the kth iteration. Then a generalized linear model is fitted against the data by using the selected bases. Then the weighted least squares method uses the working weights and working response in the last step of the iterative reweighted least squares algorithm as the weight and response for selecting new bases in the $(k + 1)$th iteration. Then the residual chi-square statistic is used to select two new bases. This is similar to the forward selection scheme that the LOGISTIC procedure uses. For more information about the score chi-square statistic, see the section "Testing Individual Effects Not in the Model" on page 4247 in Chapter 54, "The LOGISTIC Procedure."

In the backward selection process, the ADAPTIVEREG procedure extends the algorithm in the following way. Suppose there are M bases in the selected model. The Wald chi-square statistic is used to determine which basis to delete. After one basis is selected for deletion, a generalized linear model is refitted with the remaining bases. This is similar to the backward deletion scheme that the LOGISTIC procedure uses. For more information about the Wald chi-square statistic, see the section "Testing Linear Hypotheses about the Regression Coefficients" on page 4262 in Chapter 54, "The LOGISTIC Procedure."

Accordingly, the lack-of-fit criterion in the forward selection for generalized linear models is the score chi-square statistic. For the lack-of-fit criterion in the backward selection process for generalized linear models, the residual sum of squares term is replaced by the model deviance.

Fast Algorithm

The original multivariate adaptive regression splines algorithm is computationally expensive. To improve the computation speed, Friedman (1993) proposes the fast algorithm. The essential idea of the fast algorithm is to reduce the number of combinations of $\mathbf{B}$, $\mathbf{v}$, and t that are examined at each step of forward selection.

Suppose there are $(2k + 1)$ bases that are formed after the kth iteration, where a parent basis $\mathbf{B}_m$ is selected to construct two new bases. Consider a queue with bases as its elements. At the top of the queue are the two newly constructed bases, $\mathbf{B}_{2k}$ and $\mathbf{B}_{2k+1}$. The rest of the queue is sorted based on the minimum lack-of-fit criterion for each basis:

$$J(\mathbf{B}_i) = \min_{\substack{\text{for all eligible } \mathbf{v} \\ \text{for all knot } t}} \mathrm{LOF}(\mathbf{v}, t|\mathbf{B}_i),\ i = 1, \ldots, 2k-1$$

When k is not small, there are a relatively large number of bases in the model, and adding more bases is unlikely to dramatically improve the fit. Thus the ranking of the bases in the priority queue is not likely to change much during adjacent iterations. So the candidate parent bases can be restricted to the top K ones in the queue for $(k + 1)$th iteration. After the kth iteration, the top bases have new $J(\mathbf{B}_i)$ values, whereas the values of the bottom bases are unchanged. The queue is reordered based on $J(\mathbf{B}_i)$ values. This corresponds to the K= option value for the FAST option in the MODEL statement.

To avoid losing the candidate bases that are ranked at the bottom of the queue and to allow them to rise back to the top, a natural "aging" factor is introduced into each basis. This is accomplished by defining the priority for each basis function to be

$$P(\mathbf{B}_i) = R(\mathbf{B}_i) + \beta(k_c - k_r)$$

where $R(\mathbf{B}_i)$ is the rank of ith basis in the queue, k_c is the current iteration number, and k_r is the number of the iteration where the $J(\mathbf{B}_i)$ value was last computed. The top K candidate bases are then sorted again based on this priority. Large β values cause bases that have low improvement during previous iterations to rise faster to the top of the list. This corresponds to the BETA= value for the FAST option in the MODEL statement.

For a candidate basis in the top of the priority queue, the minimum lack-of-fit criterion $J(\mathbf{B}_i)$ is recomputed for all eligible variables $\mathbf{v}$ for the $(k+1)$ iteration. An optimal variable is likely to be the same as the one that was found during the previous iteration. So the fast multivariate adaptive regression splines algorithm introduces another factor H to save the computation cost. The factor specifies how often $J(\mathbf{B}_i)$ should be recomputed for all eligible variables. If $H = 1$, then optimization over all variables is done at each iteration when a parent basis is considered. If $H = 5$, the complete optimization is done after five iterations. For an iteration count less than the specified H, the optimization is done only for the optimal variable found in the last complete optimization. The only exceptions are the top three candidates, $\mathbf{B}_{2k-1}$ (which is the parent basis $\mathbf{B}_m$ used to construct two new bases) and two new ones, $\mathbf{B}_{2k}$ and $\mathbf{B}_{2k+1}$. The complete optimization for them is performed at each iteration. This corresponds to the H= option value for the FAST option in the MODEL statement.

Missing Values

When fitting a model, the ADAPTIVEREG procedure excludes observations that have missing values for the response variable, weight variable, or frequency variable. It also excludes observations with invalid response, weight, or frequency values. For observations that have valid response, weight, and frequency values but missing predictor values, the ADAPTIVEREG procedure can either include them in model fitting or exclude them.

By default, observations with missing values in the predictor variables are included in the model fitting. Suppose a variable $\mathbf{v}$ contains missing values. The ADAPTIVEREG procedure automatically forms two candidate bases, $\mathbf{B}_m$ and $\mathbf{B}_{m+1}$, in the forward selection step when variable $\mathbf{v}$ is considered. When v is missing, $\mathbf{B}_{m+1} = I(v \text{ is missing})$. When v is not missing, $\mathbf{B}_m = I(v \text{ is not missing})$. $I(\cdot)$ is a scalar-valued indicator function that returns a 1 when the argument is true and a 0 when the argument is false.

If the transformation of $\mathbf{v}$ with a parent basis $\mathbf{B}_i$ and a knot (or a subset) t turns out to be the best one during this iteration, then two more bases are added to the model:

$$\mathbf{B}_{m+2} = \mathbf{B}_i \mathbf{B}_m \mathbf{T}_1(v - t)$$

$$\mathbf{B}_{m+3} = \mathbf{B}_i \mathbf{B}_{m+1} \mathbf{T}_2(v - t)$$

The indicator function does not contribute to the interaction order of the constructed bases. This approach assumes that the missingness in the training data is representative of missingness in future data to be predicted.

Alternatively, you can specify the NOMISS option in the MODEL statement to exclude from the model fitting all observations that have missing values in the predictor variables.

ANOVA Decomposition

The model that is produced by the multivariate adaptive regression splines algorithm can be formed as

$$\begin{aligned}\hat{f}(\mathbf{x}) &= \beta_0 + \sum_{m=1}^{M} \beta_m \mathbf{B}_m \\ &= \beta_0 + \sum_{m=1}^{M} \beta_m \prod_{k=1}^{K_m} \mathbf{T}_m(\mathbf{x}_{k,m}, t_{k,m})\end{aligned}$$

Here $\hat{f}$ is the nonparametric estimate of the response variable in linear models and of the linked response variable in generalized linear models. M is the number of nonconstant bases. For each formed basis, K_m is the order of interaction, $\mathbf{T}_m$ is the variable transformation function that depends on the variable type, $\mathbf{x}_{k,m}$ is variable for the kth component of the basis, and $t_{k,m}$ is the corresponding knot value or subset categories for the variable.

The function estimate can be recast into the form

$$\hat{f}(\mathbf{x}) = \beta_0 + \sum_{i:K_m=1} f_i(\mathbf{x}_i) + \sum_{i,j:K_m=2} f_{ij}(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i,j,k:K_m=3} f_{ijk}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) + \cdots$$

where f_i represents the sum of bases that involve a single variable $\mathbf{x}_i$, f_{ij} represents the sum of bases that involve two-way interactions between transformations of two variables, and so on. The univariate function f_i is a linear regression spline for variable $\mathbf{x}_i$, which represent the univariate contribution of $\mathbf{x}_i$ to the model. Let

$$f_{ij}^* = f_i(\mathbf{x}_i) + f_j(\mathbf{x}_j) + f_{ij}(\mathbf{x}_i, \mathbf{x}_j)$$

Then this bivariate function is a tensor product regression spline that represents the joint contribution by both $\mathbf{x}_i$ and $\mathbf{x}_j$. Multivariate functions can be formed similarly if higher-order interaction terms are present in the model. Because of its similarity to the analysis of variance for contingency tables, this is referred as the ANOVA decomposition of the multivariate adaptive regression splines model.

Computational Resources

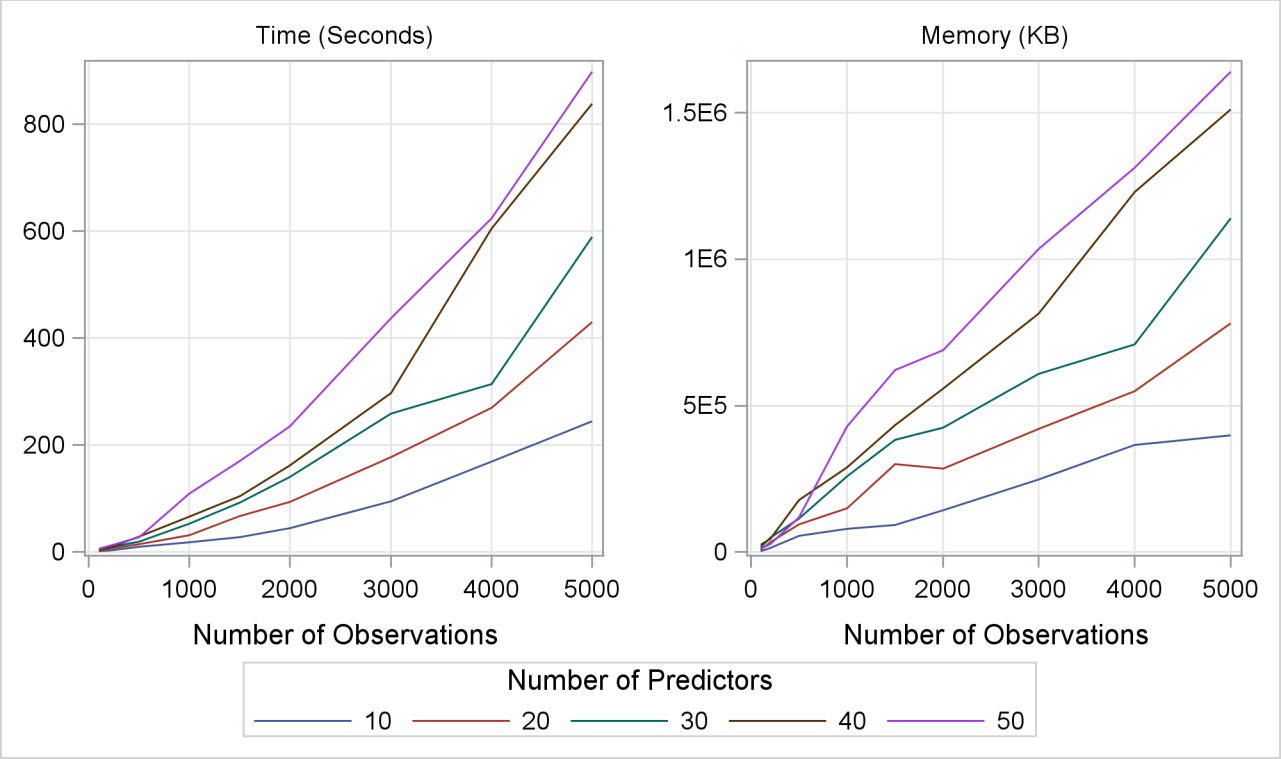
The multivariate adaptive regression splines algorithm is computationally intensive and requires a significant amount of memory for large data sets. However, the core algorithm is fairly scalable, so you might expect performance improvement if you use multicore machines. You can even further improve the fitting speed by carefully tuning the parameters of the FAST option in the MODEL statement.

A general formula does not exist for predicting amount of memory that is required for PROC ADAPTIVEREG. The procedure uses logical utility files to store values that are associated with observations. If sufficient random access memory (RAM) is available, the utility files reside in RAM to allow fast access. Otherwise, the utility files are stored on hard drives, which have slower read/write speed.

Because of the model selection nature, the multivariate adaptive regression splines algorithm essentially fits a large number of candidate models with different sets of basis functions. The original prototype requires computation that is proportional to $pNM_{\max}^4$. The implemented algorithm that takes advantage of the special structure of linear truncated power functions to reduce the computation to be proportional to $pNM_{\max}^3$. With the fast algorithm, the computational can be reduced even further.

To provide a feel for how the number of variables and the number of observations affect the fitting performance, a series of simulations are carried out on a server with a 12-way 2.6GHz AMD Opteron processor and 32GB of RAM. Data sets are created in different sizes with the number of variables ranging from 10 to 50 and the number of observations ranging from 100 to 5,000. For each data set, the true model is the same as the one used in Example 24.1. At each data size, the experiment is repeated three times to measure minimum running times and corresponding memory consumption. PROC ADAPTIVEREG sets the maximum number of basis function to 50 and uses all other default options. Figure 24.10 displays the results of the simulations.

Figure 24.10 Time and Memory Used

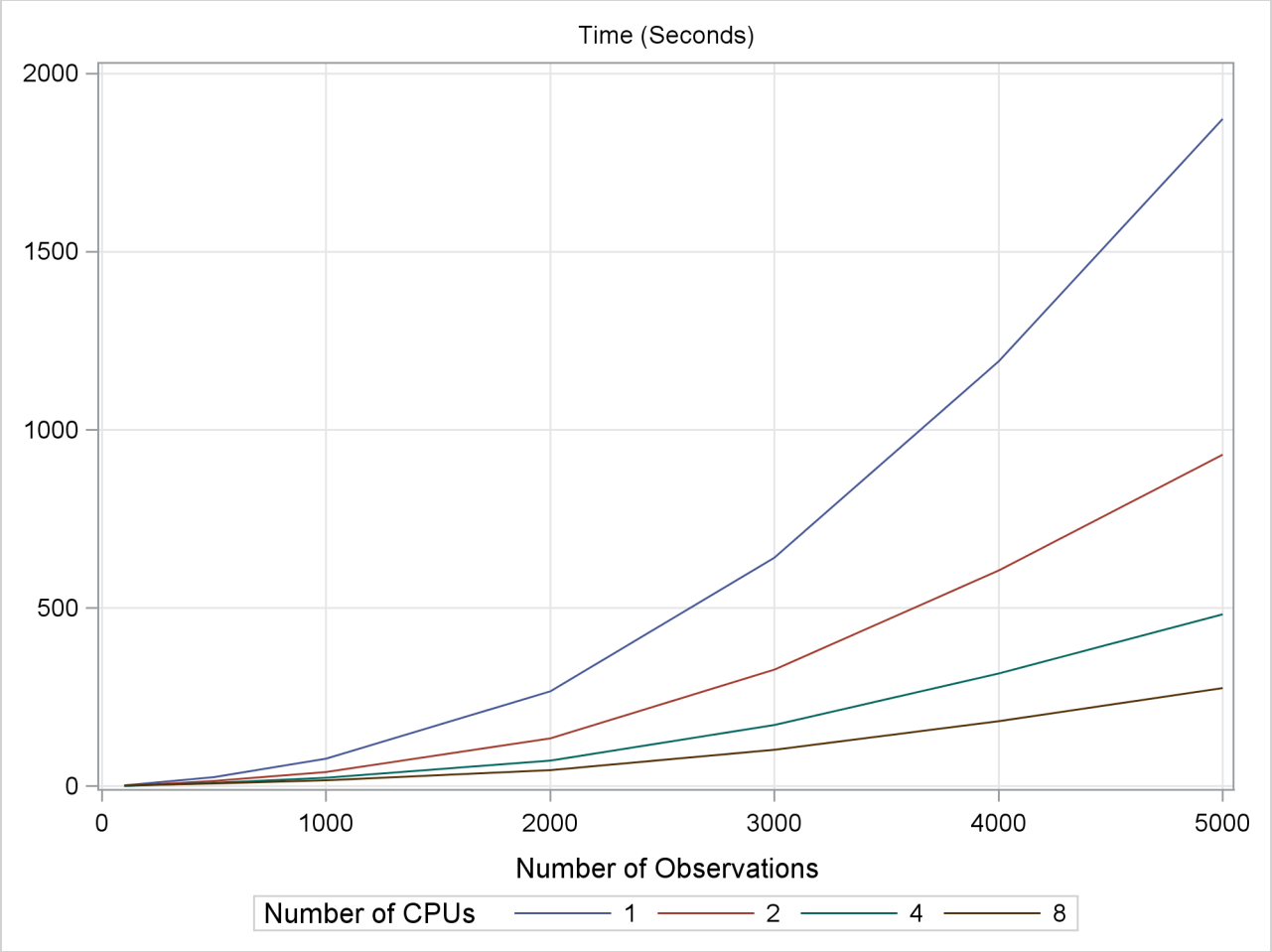


The graphs in Figure 24.10 show that computational times grow with respect to the number of observations at a speed that is slightly faster than linear. The growth of the memory consumption is close to linear. Also, the increment with respect to number of variables is approximately in fixed ratios.

To show how PROC ADAPTIVEREG scales as the number of CPUs grows, another series of experiments is performed with the following settings. SAS DATA steps use the same mechanism as in previous simulations to generate data sets. The number of observations range from 100 to 5,000, and the number of variables is 10. PROC ADAPTIVEREG fits models to these data sets with the maximum number of basis functions set to 50. Each fitting uses four threading settings with 1, 2, 4, and 8 CPUs.

Figure 24.11 displays the simulation results. The computation times scale well with the number of CPUs. The more CPUs you have, the less time you need to fit a model by using PROC ADAPTIVEREG.

Figure 24.11 Time Used with Different Number of Threads



ODS Table Names

PROC ADAPTIVEREG assigns a name to each table that it creates. You can use these names to refer to the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 24.7. For more information about ODS, see Chapter 20, "Using the Output Delivery System."

Table 24.7 ODS Tables Produced by PROC ADAPTIVEREG

ODS Table Name	Description	Statement	Option
ANOVA	ANOVA functional decomposition	PROC	Default
Bases	Bases transformation information	PROC	DETAILS=BASES
BWDParams	Parameter estimates after backward selection	PROC	Default
ClassInfo	Classification variable levels information	CLASS	Default
FitControls	Fit control parameters	PROC	Default

Table 24.7 *continued*

ODS Table Name	Description	Statement	Option
FitStatistics	Model fit statistics	PROC	Default
FWDParams	Parameter estimates after forward selection	PROC	DETAILS=FWDPARAMS
FWDSummary	Forward selection summary	PROC	DETAILS=FWDSUMMARY
ModelInfo	Model information	PROC	Default
NObs	Number of observations	PROC	Default
SelectionSummary	Backward selection summary	PROC	DETAILS=BWDSUMMARY
VarImp	Variable importance information	PROC	Default

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS."

Before you create graphs, ODS Graphics must be enabled (for example, by specifying the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" on page 600 in Chapter 21, "Statistical Graphics Using ODS."

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" on page 599 in Chapter 21, "Statistical Graphics Using ODS."

You must also specify the PLOTS= option in the PROC ADAPTIVEREG statement.

PROC ADAPTIVEREG assigns a name to each graph that it creates using ODS. You can use these names to refer to the graphs when using ODS. The names are listed in Table 24.8.

Table 24.8 Graphs Produced by PROC ADAPTIVEREG

ODS Graph Name	Plot Description	PLOTS Option
BoxPlot	Box plot of the response at each level of one categorical predictor, overlaid with a plot of predicted values	FIT
ComponentPanel	Panel of partial prediction curves for components that contain up to two predictors	COMPONENTS
ContourPlot	Contour plot of the fitted surface by two continuous predictors overlaid on scatter plot of data	FIT
DiagnosticPanel	Panel of fit diagnostics	DIAGNOSTICS
FitPlot	Plot of fitted values by single continuous predictor (or with one categorical variable) overlaid on scatter plot of data	FIT
IntPlot	Plot of fitted values by two categorical predictors overlaid on scatter plot of data	FIT
ObservedByPredicted	Dependent variable versus fitted values	DIAGNOSTICS(UNPACK)

Table 24.8 *continued*

ODS Graph Name	Plot Description	PLOTS Option
QQPlot	Normal quantile plot of residuals	DIAGNOSTICS(UNPACK)
ResidualByPredicted	Residuals versus fitted values	DIAGNOSTICS(UNPACK)
ResidualHistogram	Histogram of fit residuals	DIAGNOSTICS(UNPACK)
RFPlot	Side-by-side plots of quantiles of centered fit and residuals	DIAGNOSTICS(UNPACK)
SelectionPlot	Model fit criteria by step	SELECTION

Examples: ADAPTIVEREG Procedure

Example 24.1: Surface Fitting with Many Noisy Variables

This example shows how you can use PROC ADAPTIVEREG to fit a surface model from a data set that contains many nuisance variables.

Consider a simulated data set that contains a response variable and 10 continuous predictors. Each continuous predictor is sampled independently from the uniform distribution $U(0, 1)$. The true model is formed by x_1 and x_2:

$$y = \frac{40 \exp\left(8\left((x_1 - 0.5)^2 + (x_2 - 0.5)^2\right)\right)}{\exp\left(8\left((x_1 - 0.2)^2 + (x_2 - 0.7)^2\right)\right) + \exp\left(8\left((x_1 - 0.7)^2 + (x_2 - 0.2)^2\right)\right)}$$

The values of the response variable are generated by adding errors from the standard normal distribution $N(0, 1)$ to the true model. The generating mechanism is adapted from Gu et al. (1990). There are 400 generated observations in all. The following statements create an artificial data set:

```
data artificial;
   drop i;
   array X{10};
   do i=1 to 400;
      do j=1 to 10;
         X{j} = ranuni(1);
      end;
      Y = 40*exp(8*((X1-0.5)**2+(X2-0.5)**2))/
            (exp(8*((X1-0.2)**2+(X2-0.7)**2))+
             exp(8*((X1-0.7)**2+(X2-0.2)**2)))+rannor(1);
     output;
   end;
run;
```

The standard deviation for the response without noise is 3, whereas the standard deviation for the error term is 1. So the response variable Y has a signal-to-noise ratio of 3. When eight more variables are introduced, it is harder to search for the true model because of the extra variability that the nuisance variables create. The objective is to fit a nonparametric surface model that can well approximate the true model without experiencing much interference from the nuisance variables.

The following statements invoke the ADAPTIVEREG procedure to fit the model:

```
ods graphics on;

proc adaptivereg data=artificial plots=fit;
   model y=x1-x10;
run;
```

The PLOTS=FIT option in the PROC ADAPTIVEREG statement requests a fit plot. PROC ADAPTIVEREG might not produce the fit plot because the number of predictors in the final model is unknown. If the final model has no more than two variables, then the fit can be graphically presented.

PROC ADAPTIVEREG selects the two variables that form the true model (X1, X2) and does not include other nuisance variables. The "Fit Statistics" table (Output 24.1.1) lists summary statistics of the fitted surface model. The model has 27 effective degrees of freedom and 14 basis functions formed by X1 or X2 or both. The fit statistics suggest that this is a reasonable fit.

Output 24.1.1 Fit Statistics

The ADAPTIVEREG Procedure

Fit Statistics

GCV	1.55656
GCV R-Square	0.86166
Effective Degrees of Freedom	27
R-Square	0.87910
Adjusted R-Square	0.87503
Mean Square Error	1.40260
Average Square Error	1.35351

Output 24.1.2 lists both parameter estimates and construction components (parent basis function, new variable, and optimal knot for the new variable) for the basis functions.

Output 24.1.2 Parameter Estimates

Regression Spline Model after Backward Selection

Name	Coefficient	Parent	Variable	Knot
Basis0	12.3031		Intercept	
Basis1	13.1804	Basis0	X1	0.05982
Basis3	-23.4892	Basis0	X2	0.1387
Basis4	-171.03	Basis0	X2	0.1387
Basis5	-86.1867	Basis3	X1	0.6333
Basis7	-436.86	Basis4	X1	0.5488
Basis8	397.18	Basis4	X1	0.5488
Basis9	11.4682	Basis1	X2	0.6755
Basis10	-19.1796	Basis1	X2	0.6755
Basis13	126.84	Basis11	X1	0.6018
Basis14	40.8134	Basis11	X1	0.6018
Basis15	22.2884	Basis0	X1	0.7170
Basis17	-53.8746	Basis12	X1	0.2269
Basis19	598.89	Basis4	X1	0.2558

Output 24.1.3 shows all the ANOVA functional components that form the final model. The function estimate consists of two basis functions for each of X1 and X2 and nine bivariate functions of both variables. Because the true model contains the interaction between X1 and X2, PROC ADAPTIVEREG automatically selects many interaction terms.

Output 24.1.3 ANOVA Decomposition

ANOVA Decomposition

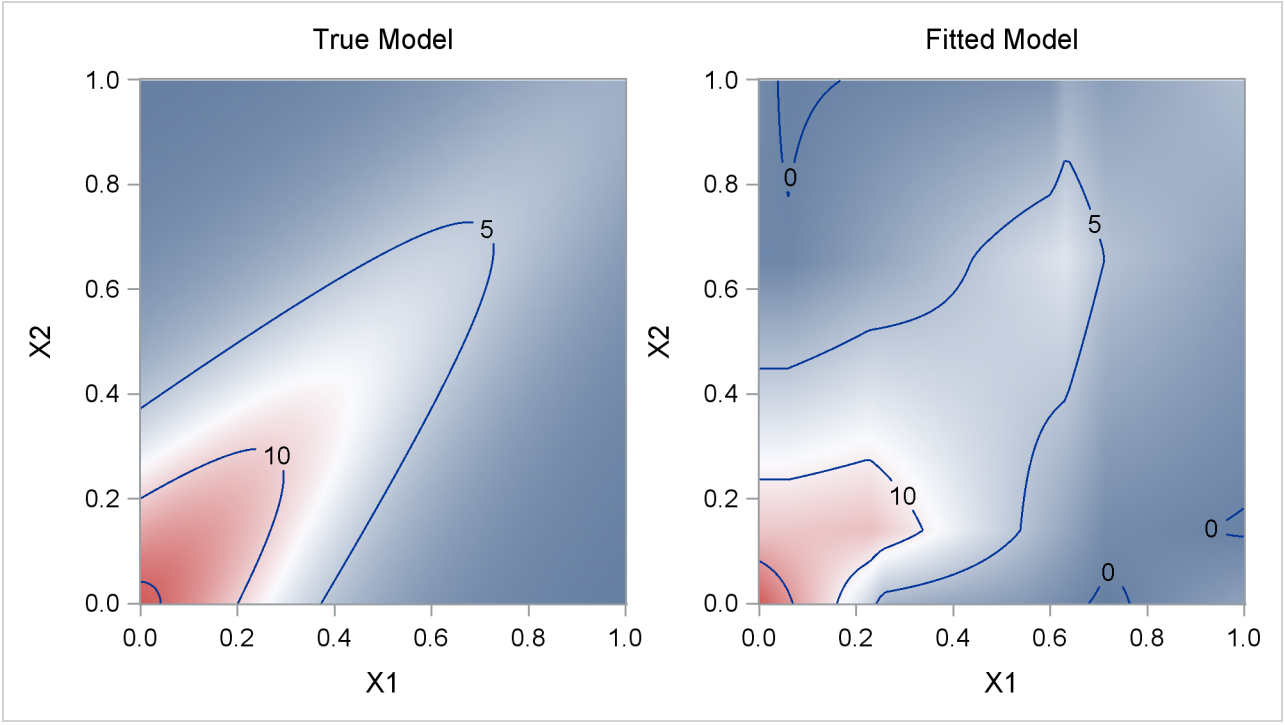
Functional Component	Number of Bases	DF	---Change If Omitted--- Lack of Fit	GCV
X1	2	4	405.18	1.1075
X2	2	4	947.87	2.6348
X2 X1	9	18	2583.21	6.6187

To compute predictions for the contour plot of the fitted model, you can use the SCORE statement. The following statements produce the graph that shows both the true model and the fitted model:

```
data score;
   do X1=0 to 1 by 0.01;
      do X2=0 to 1 by 0.01;
         Y=40*exp(8*((X1-0.5)**2+(X2-0.5)**2))/
             (exp(8*((X1-0.2)**2+(X2-0.7)**2))+
              exp(8*((X1-0.7)**2+(X2-0.2)**2)));
         output;
      end;
   end;
run;

proc adaptivereg data=artificial;
   model y=x1-x10;
   score data=score out=scoreout;
run;

proc template;
   define statgraph surfaces;
   begingraph / designheight=360px;
      layout lattice/columns=2;
         layout overlay/xaxisopts=(offsetmin=0 offsetmax=0)
                        yaxisopts=(offsetmin=0 offsetmax=0);
            entry "True Model" / location=outside valign=top
                                 textattrs=graphlabeltext;
            contourplotparm z=y y=x2 x=x1;
         endlayout;
         layout overlay/xaxisopts=(offsetmin=0 offsetmax=0)
                        yaxisopts=(offsetmin=0 offsetmax=0);
            entry "Fitted Model" / location=outside valign=top
                                   textattrs=graphlabeltext;
            contourplotparm z=pred y=x2 x=x1;
         endlayout;
      endlayout;
   endgraph;
end;

proc sgrender data=scoreout template=surfaces;
run;
```

Output 24.1.4 displays surfaces for both the true model and the fitted model. The fitted model approximates the underlying true model well.

Output 24.1.4 True Model and Fitted Model



For high-dimensional data sets with complex underlying data-generating mechanisms, many different models can almost equally approximate the true mechanisms. Because of the sequential nature of the selection mechanism, any change in intermediate steps due to perturbations from local structures might yield completely different models. Therefore, PROC ADAPTIVEREG might find models that contain noisy variables. For example, if you change the random number seed in generating the data (as in the following statements), PROC ADAPTIVEREG might return different models with more variables. You can use the information from the variable importance table (Output 24.1.5) to aid further analysis.

```
data artificial;
   drop i;
   array x{10};
   do i=1 to 400;
      do j=1 to 10;
         x{j} = ranuni(12345);
      end;
      y = 40*exp(8*((x1-0.5)**2+(x2-0.5)**2))/
            (exp(8*((x1-0.2)**2+(x2-0.7)**2))+
             exp(8*((x1-0.7)**2+(x2-0.2)**2)))+rannor(1);
      output;
   end;
run;

proc adaptivereg data=artificial;
   model y=x1-x10;
run;
```

Output 24.1.5 shows that the variables X1 and X2 are two dominating factors for predicting the response, whereas the relative importance of the variable X8 compared to the other two is negligible. You might want to remove the variable if you fit a new model.

Output 24.1.5 Variable Importance

The ADAPTIVEREG Procedure

Variable Importance

Variable	Number of Bases	Importance
x1	13	100.00
x2	12	98.58
x8	2	0.22

Example 24.2: Fitting Data with Mixture Structures

This example shows how you can use PROC ADAPTIVEREG to fit a model from a data set that contains mixture structures. It also demonstrates how to use the CLASS statement.

Consider a simulated data set that contains a response variable and two predictors, one continuous and the other categorical. The continuous predictor is sampled from the uniform distribution $U(0, 1)$, and the classification variable is sampled from $U(0, 3)$ and then rounded to integers. The response variable is constructed from three different models that depend on the CLASS variable levels, with error sampled from the standard normal distribution.

$$y = \begin{cases} \exp(5(x-0.3)^2), & \text{if } c = 0 \\ \log(x - x^2), & \text{if } c = 1 \\ 7x, & \text{if } c = 2 \end{cases}$$

The following statements create the artificial data set Mixture:

```
data Mixture;
   drop i;
   do i=1 to 1000;
      X1 = ranuni(1);
      C1 = int(3*ranuni(1));
      if C1=0 then Y=exp(5*(X1-0.3)**2)+rannor(1);
      else if C1=1 then Y=log(X1*(1-X1))+rannor(1);
      else Y=7*X1+rannor(1);
      output;
   end;
run;
```

The standard deviation for the response without noise is 3.14. So the response variable Y in the data set Mixture has a signal-to-noise ratio of 3.14. With a classification variable and a continuous variable in the data set, the objective is to fit a nonparametric model that can reveal the underlying three different data-generating processes. The following statements use the ADAPTIVEREG procedure to fit the data:

```
ods graphics on;
proc adaptivereg data=Mixture plots=fit;
   class c1;
   model y=c1 x1;
run;
```

Because the data contain two explanatory variables, graphical presentation of the fitted model is possible. The PLOTS=FIT option in the PROC ADAPTIVEREG statement requests the fit plot. The CLASS statement specifies that C1 is a classification variable.

Output 24.2.1 displays the parameter estimates for the 13 selected basis functions after backward selection. For Basis1, the coefficient estimate is -4.3871. It is constructed from the intercept and the classification variable C1 at levels 0 and 1.

Output 24.2.1 Parameter Estimates

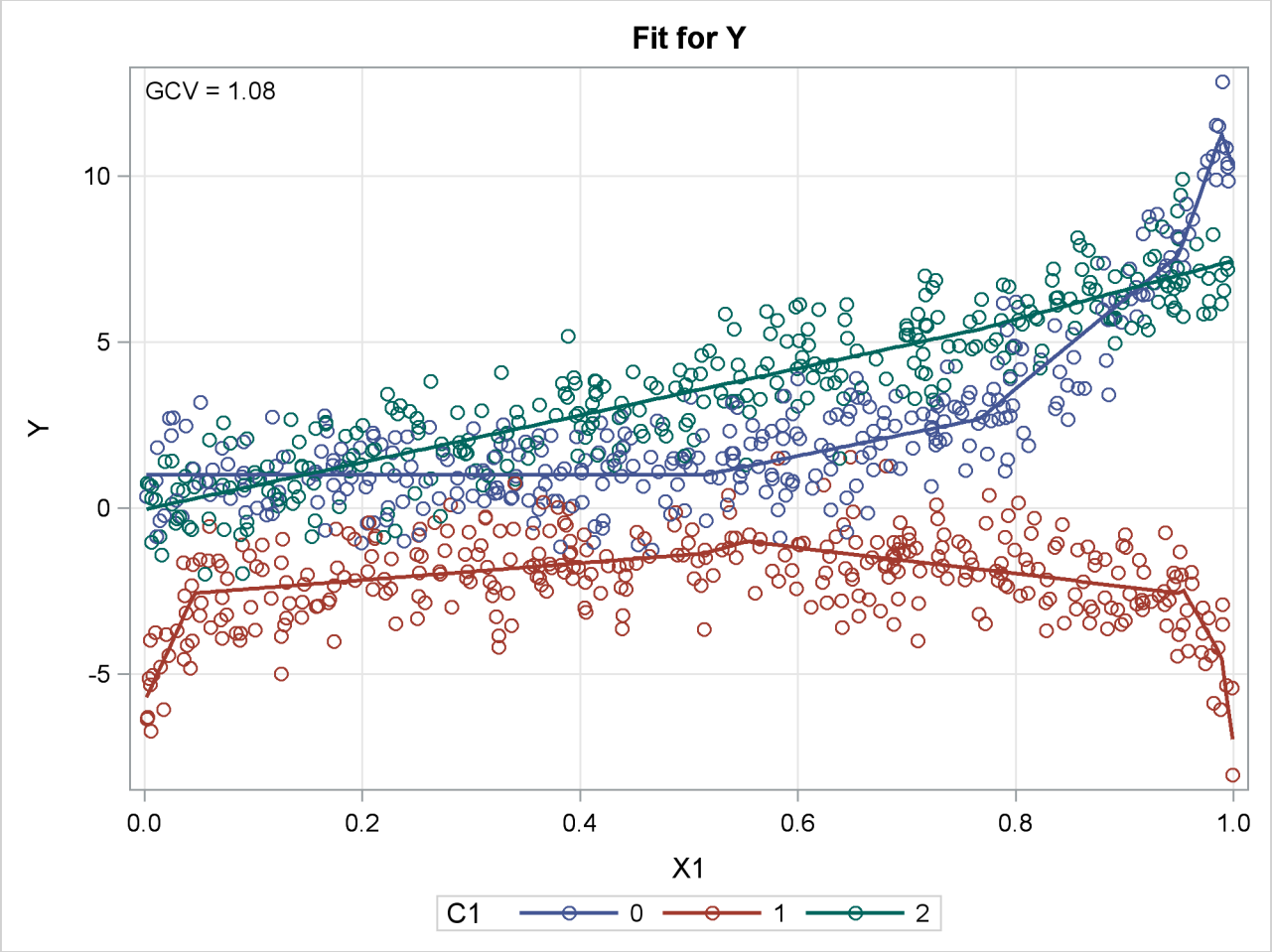
The ADAPTIVEREG Procedure

Regression Spline Model after Backward Selection

Name	Coefficient	Parent	Variable	Knot	Levels
Basis0	5.3829		Intercept		
Basis1	-4.3871	Basis0	C1		1 0
Basis3	32.7761	Basis0	C1		1
Basis5	20.2859	Basis4	X1	0.7665	
Basis7	-11.4183	Basis2	X1	0.7665	
Basis8	-7.0758	Basis2	X1	0.7665	
Basis9	58.4911	Basis3	X1	0.5531	
Basis10	-71.6388	Basis3	X1	0.5531	
Basis11	-69.0764	Basis3	X1	0.04580	
Basis13	-119.71	Basis3	X1	0.9526	
Basis15	66.5733	Basis1	X1	0.9499	
Basis17	6.6681	Basis1	X1	0.5143	
Basis19	-185.21	Basis1	X1	0.9890	

Output 24.2.2 displays the fitted linear splines overlaid with the original data. PROC ADAPTIVEREG captures the three underlying data-generating processes. For observations with C1 at level 0, the shape of the fitted splines is quite similar to the exponential function. For observations with C1 at level 1, the shape of the fitted spline suggests a symmetric function along X1 with a symmetry point approximately equal to 0.5. The function at each side of the symmetry point is analogous to the logarithmic transformation. For the rest of the observations with C1 at level 2, PROC ADAPTIVEREG suggests a strict linear model. The fitted model is very close to the true model. PROC ADAPTIVEREG fits the model in an automatic and adaptive way, except that it needs the CLASS statement to name the classification variable.

Output 24.2.2 Raw Data and Fitted Model



You might notice that some basis functions have their parent basis functions not listed in the parameter estimates table (Output 24.2.1). This is because their parent basis functions are dropped during the model selection process. You can view the complete set of basis functions used in the model selection by specifying the DETAILS=BASES option in the PROC ADAPTIVEREG statement, as in the following statements:

```
proc adaptivereg data=Mixture details=bases;
   class c1;
   model y=c1 x1;
run;
```

Output 24.2.3 Basis Function Information

The ADAPTIVEREG Procedure

Basis Information

Name	Transformation
Basis0	None
Basis1	Basis0*(C1 = 1 OR C1 = 0)
Basis2	Basis0*NOT(C1 = 1 OR C1 = 0)
Basis3	Basis0*(C1 = 1)
Basis4	Basis0*NOT(C1 = 1)
Basis5	Basis4*MAX(X1 - 0.7665019053,0)
Basis6	Basis4*MAX(0.7665019053 - X1,0)
Basis7	Basis2*MAX(X1 - 0.7665019053,0)
Basis8	Basis2*MAX(0.7665019053 - X1,0)
Basis9	Basis3*MAX(X1 - 0.5530566455,0)
Basis10	Basis3*MAX(0.5530566455 - X1,0)
Basis11	Basis3*MAX(X1 - 0.045800759,0)
Basis12	Basis3*MAX(0.045800759 - X1,0)
Basis13	Basis3*MAX(X1 - 0.9526330293,0)
Basis14	Basis3*MAX(0.9526330293 - X1,0)
Basis15	Basis1*MAX(X1 - 0.9499325226,0)
Basis16	Basis1*MAX(0.9499325226 - X1,0)
Basis17	Basis1*MAX(X1 - 0.5142821095,0)
Basis18	Basis1*MAX(0.5142821095 - X1,0)
Basis19	Basis1*MAX(X1 - 0.9889635476,0)
Basis20	Basis1*MAX(0.9889635476 - X1,0)

You can produce a SAS DATA step for scoring new observations by using the information provided in the parameter estimate table and the basis information table, as shown in the following statements:

```
data New;
   basis1  = (c1=1 OR c1=0);
   basis3  = (c1=1);
   basis5  = NOT(c1=1)*MAX(x1-0.7665019053,0);
   basis7  = NOT(c1=1 OR c1=0)*MAX(x1-0.7665019053,0);
   basis8  = NOT(c1=1 OR c1=0)*MAX(0.7665019053-x1,0);
   basis9  = (c1=1)*MAX(x1-0.5530566455,0);
   basis10 = (c1=1)*MAX(0.5530566455-x1,0);
   basis11 = (c1=1)*MAX(x1-0.045800759,0);
   basis13 = (c1=1)*MAX(x1-0.9526330293,0);
   basis15 = (c1=1 OR c1=0)*MAX(x1-0.9499325226,0);
   basis17 = (c1=1 OR c1=0)*MAX(x1-0.5142821095,0);
   basis19 = (c1=1 OR c1=0)*MAX(x1-0.9889635476,0);
   pred    = 5.3829 - 4.3871*basis1 + 32.7761*basis3 +
             20.2859*basis5 - 11.4183*basis7 - 7.0758*basis8 +
             58.4911*basis9 - 71.6388*basis10 - 69.0764*basis11 -
             119.71*basis13 + 66.5733*basis15 + 6.6681*basis17 -
             185.21*basis19;
run;
```

Example 24.3: Predicting E-Mail Spam

This example shows how you can use PROC ADAPTIVEREG to fit a classification model for a data set with a binary response. It illustrates how you can use the PARTITION statement to create subsets of data for training and testing purposes. It also demonstrates how to use the OUTPUT statement. Finally, it shows how you can improve the modeling speed by changing some default settings.

This example concerns a study on classifying whether an e-mail is junk e-mail (coded as 1) or not (coded as 0). The data were collected in Hewlett-Packard labs and donated by George Forman. The data set contains 4,601 observations with 58 variables. The response variable is a binary indicator of whether an e-mail is considered spam or not. The 57 variables are continuous variables that record frequencies of some common words and characters in e-mails and lengths of uninterrupted sequences of capital letters. The data set is publicly available at the UCI Machine Learning repository (Asuncion and Newman 2007).

The following DATA step downloads the data from the UCI Machine Learning repository and creates a SAS data set called spambase:

```
%let base = http://archive.ics.uci.edu/ml/machine-learning-databases;
data spambase;
   infile "&base/spambase/spambase.data" device=url dsd dlm=',';
    input Make Address All _3d Our Over Remove Internet Order Mail Receive
       Will People Report Addresses Free Business Email You Credit Your Font
       _000 Money Hp Hpl George _650 Lab Labs Telnet _857 Data _415 _85
       Technology _1999 Parts Pm Direct Cs Meeting Original Project Re Edu
       Table Conference Semicol Paren Bracket Bang Dollar Pound Cap_Avg
       Cap_Long Cap_Total Class;
   run;
```

This example shows how you can use PROC ADAPTIVEREG to build a model with good predictive power and then use it to classify observations in independent data sets. PROC ADAPTIVEREG enables you to partition your data into subsets for training, validation, and testing. The training set is used to build models, the validation set is used to estimate prediction errors and select models, and the testing set is used independently to evaluate the final model. When the sample size is not large enough, sample reusing approaches are used instead, such as bootstrap and cross validation. For this data set, the sample size is sufficient to support a random partitioning. Because the GCV model selection criterion itself serves as an estimate of prediction error, this data set is split into two separate subsets. The training set is used to build the classification model, and the test set is used to evaluate the model. The PARTITION statement performs the random partitioning for you, as shown in the following statements:

```
proc adaptivereg data=spambase seed=10359;
   class Class;
   model class = _000          _85          _415          _650          _857
                 _1999         _3d           address       addresses     all
                 bang          bracket       business      cap_avg       cap_long
                 cap_total     conference    credit        cs            data
                 direct        dollar        edu           email         font
                 free          george        hp            hpl           internet
                 lab           labs          mail          make          meeting
                 money         order         original      our           over
                 paren         parts         people        pm            pound
                 project       re            receive       remove        report
                 semicol       table         technology    telnet        will
                 you           your   / additive dist=binomial;
   partition fraction(test=0.333);
   output out=spamout p(ilink);
run;
```

The FRACTION option in the PARTITION statement specifies that 33.3% of observations in the spambase data set are randomly selected to form the testing set while the rest of the data form the training set. If you want to use the same partitioning for further analysis, you can specify the seed for the random number generator so that the exact same random number stream can be duplicated. For the preceding statements, the seed is 10359, which is specified in the PROC ADAPTIVEREG statement. The response variable is a two-level variable, which is specified in the CLASS statement. The ADDITIVE option specifies that this is an additive model without interactions between spline basis functions; this option makes the predictive model more interpretable. The DIST=BINOMIAL option specifies the distribution of the response variable. The ILINK option in the OUTPUT statement requests predicted probabilities for each observation.

The "Model Information" table in Output 24.3.1 includes the distribution, link function, and the random number seed.

Output 24.3.1 Model Information

The ADAPTIVEREG Procedure

Model Information

Data Set	WORK.SPAMBASE
Response Variable	Class
Distribution	Binomial
Link Function	Logit
Random Number Seed	10359

The "Number of Observations" table in Output 24.3.2 lists the total number of observations used. It also lists number of observations for the training set and the test set.

Output 24.3.2 Number of Observations

Number of Observations Read	4601
Number of Observations Used	4601
Number of Observations Used for Training	3028
Number of Observations Used for Testing	1573

The response variable is a binary classification variable. PROC ADAPTIVEREG produces the "Response Profile" table in Output 24.3.3. The table shows the response level frequencies for the training set and the probability that PROC ADAPTIVEREG models.

Output 24.3.3 Response Profile

Response Profile

Ordered Value	Class	Total Frequency
1	0	2788
2	1	1813

Probability modeled is Class='0'.

The "Fit Statistics" table in Output 24.3.3 shows that the final model for the training set contains large effective degrees freedom.

Output 24.3.4 Fit Statistics

Fit Statistics

GCV	0.23427
GCV R-Square	0.82508
Effective Degrees of Freedom	173
Log Likelihood	-315.30998
Deviance (Train)	630.61996
Deviance (Test)	806.74112

To classify e-mails from the test set, the following rule is used. For each observation, the e-mail is classified as spam if the predicted probability of Class = '0' is greater than the predicted probability of Class = '1', and ham (a good e-mail) otherwise. Because the response is binary, you can classify an e-mail as spam if the predicted probability of Class = '0' is less than 0.5. The following statements evaluate classification errors:

```
data test;
   set spamout(where=(_ROLE_='TEST'));
   if ((pred>0.5 & class=0) | (pred<0.5 & class=1))
   then Error=0;
   else error=1;
run;

proc freq data=test;
   tables class*error/nocol;
run;
```

Output 24.3.5 shows the misclassification errors for all observations and observations of each response category. Compared to the results from other statistical learning algorithms that use different training subsets (Hastie, Tibshirani, and Friedman 2001), these results from PROC ADAPTIVEREG are competitive.

Output 24.3.5 Crosstabulation Table for Test Set Prediction

The FREQ Procedure

Table of Class by Error

Class	Error		
Frequency | Percent | Row Pct	0	1	Total
0	885 56.26 93.75	59 3.75 6.25	944 60.01
1	592 37.64 94.12	37 2.35 5.88	629 39.99
Total	1477 93.90	96 6.10	1573 100.00

It takes approximately 3GB of memory and about five and a half minutes to fit the model on a workstation with a 12-way 2.6GHz AMD Opteron processor. The following analyses illustrate how you can change some default settings to improve the modeling speed without sacrificing much predictive capability. As discussed in the section "Computational Resources" on page 876, the computation cost for PROC ADAPTIVEREG is proportional to $pNM^3_{\max}$. For the same data set, you can significantly increase the modeling speed by reducing the maximum number of basis functions that are allowed for the forward selection.

PROC ADAPTIVEREG uses 115 as the maximum number of basis functions. Suppose you want to set the maximum number to 61, which is approximately half the default value. The following program fits a multivariate adaptive regression splines model with MAXBASIS= set to 61. The same random number seed is used to get the exact same data partitioning.

```
proc adaptivereg data=spambase seed=10359;
   class Class;
   model class = _000         _85         _415         _650         _857
                 _1999        _3d          address      addresses    all
                 bang         bracket      business     cap_avg      cap_long
                 cap_total    conference   credit       cs           data
                 direct       dollar       edu          email        font
                 free         george       hp           hpl          internet
                 lab          labs         mail         make         meeting
                 money        order        original     our          over
                 paren        parts        people       pm           pound
                 project      re           receive      remove       report
                 semicol      table        technology   telnet       will
                 you          your  / maxbasis=61 additive dist=binomial;
   partition fraction(test=0.333);
   output out=spamout2 p(ilink);
run;
```

The "Fit Statistics" table in Output 24.3.6 displays summary statistics for the second model. The log likelihood of the second model is smaller than that of the first model, which is expected because the effective degrees of freedom is 95, much smaller than the effective degrees of freedom of the first model. This means that the fitted model is much simpler than the first model. Both the GCV and GCV R-square values show that the estimated prediction capability of the second model is slightly less than the first model.

Output 24.3.6 Fit Statistics

The ADAPTIVEREG Procedure

Fit Statistics

GCV	0.27971
GCV R-Square	0.79115
Effective Degrees of Freedom	95
Log Likelihood	-397.32916
Deviance (Train)	794.65833
Deviance (Test)	682.79427

By predicting observations in the test set, the second model has an overall misclassification error of 5.28%, which is slightly lower that that of the first model. This shows that the predictive power of the second model is actually greater than of the first model due to reduced model complexity. The computation takes around 80 seconds on the same workstation and consumes approximately 170MB of memory. This is a significant improvement in both computation speed and memory cost.

You can further improve the modeling speed by using the FAST option in the MODEL statement. The FAST option avoids evaluating certain combinations of parent basis functions and variables. For example, you can specify the FAST(K=20) option so that in each forward selection iteration, PROC ADAPTIVEREG uses only the top 20 parent basis functions (based on their maximum improvement from the previous iteration) to construct and evaluate new basis functions. The underlying assumption, as discussed in the section "Fast Algorithm" on page 874, is that parent basis functions that offer low improvement at previous steps are less likely to yield new basis functions that offer large improvement at the current step. The following statements illustrate the FAST option:

```
proc adaptivereg data=spambase seed=10359;
   class Class;
   model class = _000         _85          _415         _650         _857
                 _1999        _3d           address      addresses    all
                 bang         bracket      business     cap_avg      cap_long
                 cap_total    conference   credit       cs           data
                 direct       dollar       edu          email        font
                 free         george       hp           hpl          internet
                 lab          labs         mail         make         meeting
                 money        order        original     our          over
                 paren        parts        people       pm           pound
                 project      re           receive      remove       report
                 semicol      table        technology   telnet       will
                 you          your  / maxbasis=61 fast(k=20) additive dist=binomial;
   partition fraction(test=0.333);
   output out=spamout3 p(ilink);
run;
```

The fitted model is the same as the second model. The computation time reduces further to 70 seconds. You should tune the parameters for the FAST option with care because the underlying assumption does not always hold.

With this investigation, the second model can serve as a good classifier. It contains 26 variables. The "Variable Importance" table (Output 24.3.7) lists all variables and their importance values in descending order. Two variables in the model, George and Hp, are important factors in classifying e-mails as not spam. George Forman, the donor of the original data set, collected e-mails from filed work and personal e-mails at Hewlett-Packard labs. Thus these two variables are strong indicators of e-mails that are not spam. This confirms the results from the fitted multivariate adaptive regression splines model by PROC ADAPTIVEREG.

Output 24.3.7 Variable Importance

Variable Importance

Variable	Number of Bases	Importance
George	1	100.00
Hp	1	78.35
Edu	3	61.25
Remove	2	49.21
Bang	3	44.14
Free	2	34.18
Meeting	3	32.57
_1999	2	29.71
Dollar	2	28.30
Money	3	26.39
Cap_Long	3	24.41
Our	2	19.46
Semicol	2	14.98
Re	2	13.52
Business	3	13.48
Over	3	12.63
Cap_Total	3	12.50
Will	1	10.81
Pound	2	9.73
Internet	1	5.88
_000	1	4.57
You	2	3.17

Example 24.4: Nonparametric Poisson Model for Mackerel Egg Density

This example demonstrates how you can use PROC ADAPTIVEREG to fit a nonparametric Poisson regression model.

The example concerns a study of mackerel egg density. The data are a subset of the 1992 mackerel egg survey conducted over the Porcupine Bank west of Ireland. The survey took place in the peak spawning area. Scientists took samples by hauling a net up from deep sea to the sea surface. Then they counted the number of spawned mackerel eggs and used other geographic information to estimate the sizes and distributions of spawning stocks. The data set is used as an example in Bowman and Azzalini (1997).

The following SAS DATA step creates the data set Mackerel. This data set contains 634 observations and five variables. The response variable Egg_Count is the number of mackerel eggs collected from each sampling net. Longitude and Latitude are the location values in degrees east and north, respectively, of each sample station. Net_Area is the area of the sampling net in square meters. Depth records the sea bed depth in meters at the sampling location. And Distance is the distance in geographic degrees from the sample location to the continental shelf edge.

```
title 'Mackerel Egg Density Study';
data Mackerel;
   input Egg_Count Longitude Latitude  Net_Area Depth Distance;
   datalines;
   0      -4.65      44.57      0.242     4342     0.8395141177
   0      -4.48      44.57      0.242     4334     0.8591926336
   0       -4.3      44.57      0.242     4286     0.8930152895
   1      -2.87      44.02      0.242     1438     0.3956408691
   4      -2.07      44.02      0.242      166     0.0400088237
   3      -2.13      44.02      0.242      460     0.0974234463
   0      -2.27      44.02      0.242      810     0.2362566569

   ... more lines ...

  22      -4.22      46.25       0.19      205     0.1181120828
  21      -4.28      46.25       0.19      237      0.129990854
   0      -4.73      46.25       0.19     2500     0.3346500536
   5      -4.25      47.23       0.19      114      0.718192582
   3      -3.72      47.25       0.19      100     0.9944669778
   0      -3.25      47.25       0.19       64     1.2639918431
;
```

The response values are counts, so the Poisson distribution might be a reasonable model. The study of interest is the mackerel egg density, which can be formed as

$$\text{density} = E(\text{count})/\text{net\_area}$$

This is equivalent to a Poisson regression with the response variable Egg_Count and an offset variable $\log(\text{net\_area})$ and other covariates.

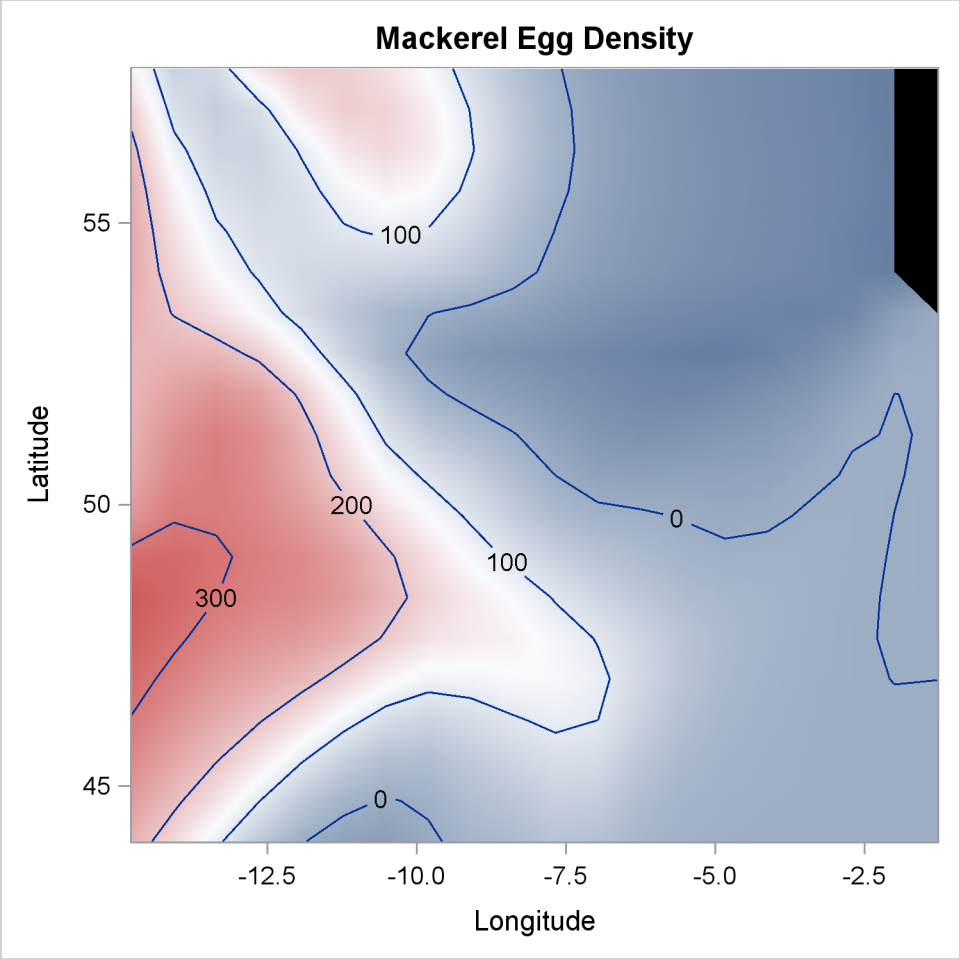
The following statements produce the plot of the mackerel egg density with respect to the sampling station location:

```
data temp;
   set mackerel;
   density = egg_count/net_area;
run;

proc template;
   define statgraph surface;
      dynamic _title _z;
      begingraph / designwidth=defaultDesignHeight;
      entrytitle _title;
      layout overlay / xaxisopts=(offsetmin=0 offsetmax=0
                                  linearopts=(thresholdmin=0 thresholdmax=0))
                       yaxisopts=(offsetmin=0 offsetmax=0
                                  linearopts=(thresholdmin=0 thresholdmax=0));
         contourplotparm z=_z y=latitude x=longitude / gridded=FALSE;
      endlayout;
      endgraph;
   end;
run;
```

```
ods graphics on;
proc sgrender data=temp template=surface;
   dynamic _title='Mackerel Egg Density'
           _z='density';
run;
```

Output 24.4.1 displays the mackerel egg density in the sampling area. The black hole in the upper right corner is due to missing values in that area.

Output 24.4.1 Mackerel Egg Density



In this example, the dependent variable is the mackerel egg counts, the independent variables are the geographical information about each of the sampling stations, and the logarithm of the sampling area is the offset variable. The following statements fit the nonparametric Poisson regression model:

```
data mackerel;
   set mackerel;
   log_net_area = log(net_area);
run;
```

```
proc adaptivereg data=mackerel;
   model egg_count = longitude latitude depth distance
                     / offset=log_net_area dist=poisson;
   output out=mackerelout p(ilink);
run;
```

Output 24.4.2 lists basic model information such as the offset variable, distribution, and link function.

Output 24.4.2 Model Information

Mackerel Egg Density Study

The ADAPTIVEREG Procedure

Model Information

Data Set	WORK.MACKEREL
Response Variable	Egg_Count
Offset Variable	log_net_area
Distribution	Poisson
Link Function	Log

Output 24.4.3 lists fit statistics for the final model.

Output 24.4.3 Fit Statistics

Fit Statistics

GCV	6.94340
GCV R-Square	0.79204
Effective Degrees of Freedom	29
Log Likelihood	-2777.21279
Deviance	4008.60601

The final model consists of basis functions and interactions between basis functions of three geographic variables. Output 24.4.4 lists seven functional components of the final model, including three one-way spline transformations and four two-way spline interactions.

Output 24.4.4 ANOVA Decomposition

ANOVA Decomposition

Functional Component	Number of Bases	DF	---Change If Omitted--- Lack of Fit	GCV
Longitude	3	6	2035.77	3.3216
Depth	1	2	420.59	0.6780
Latitude	1	2	265.05	0.4104
Longitude Latitude	2	4	199.17	0.2496
Depth Distance	3	6	552.75	0.8030
Depth Latitude	2	4	680.45	1.0723
Depth Longitude	2	4	415.77	0.6198

The "Variable Importance" table in Output 24.4.5 displays the relative variable importance among the four variables. Longitude is the most important one.

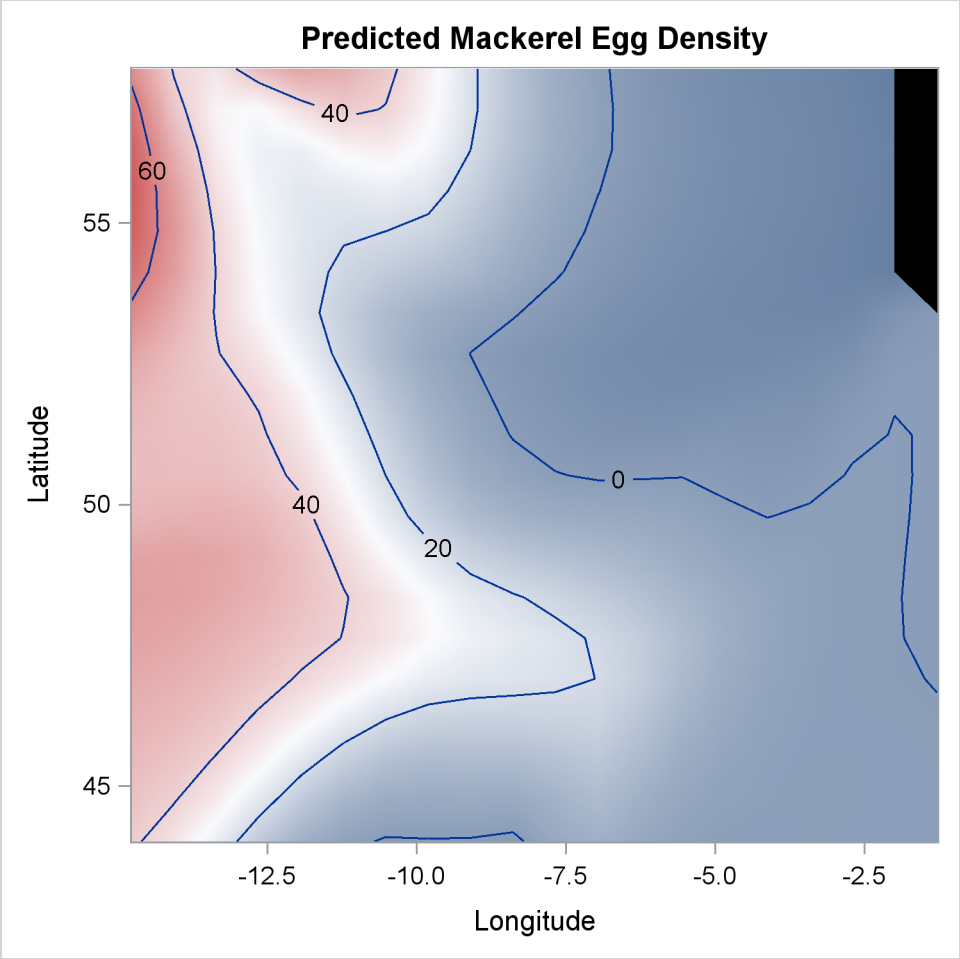
Output 24.4.5 Variable Importance

Variable Importance

Variable	Number of Bases	Importance
Longitude	7	100.00
Depth	8	30.26
Latitude	5	18.93
Distance	3	8.56

Output 24.4.6 displays the predicted mackerel egg density over the spawning area.

Output 24.4.6 Predicted Mackerel Egg Density



References

Asuncion, A. and Newman, D. J. (2007), "UCI Machine Learning Repository," http://archive.ics.uci.edu/ml/.

Bellman, R. E. (1961), *Adaptive Control Processes*, Princeton, NJ: Princeton University Press.

Bowman, A. W. and Azzalini, A. (1997), *Applied Smoothing Techniques for Data Analysis*, New York: Oxford University Press.

Breiman, L., Friedman, J., Olshen, R. A., and Stone, C. J. (1984), *Classification and Regression Trees*, Wadsworth.

Buja, A., Duffy, D., Hastie, T., and Tibshirani, R. (1991), "Discussion: Multivariate Adaptive Regression Splines," *The Annals of Statistics*, 19, 93–99.

Craven, P. and Wahba, G. (1979), "Smoothing Noisy Data with Spline Functions," *Numerical Mathematics*, 31, 377–403.

Friedman, J. (1991a), *Estimating Functions of Mixed Ordinal and Categorical Variables Using Adaptive Splines*, Technical report, Stanford University.

Friedman, J. (1991b), "Multivariate Adaptive Regression Splines," *The Annals of Statistics*, 19, 1–67.

Friedman, J. (1993), *Fast MARS*, Technical report, Stanford University.

Gu, C., Bates, D. M., Chen, Z., and Wahba, G. (1990), "The Computation of GCV Function through Householder Tridiagonalization with Application to the Fitting of Interaction Splines Models," *SIAM Journal on Matrix Analysis and Applications*, 10, 457–480.

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

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

Owen, A. (1991), "Discussion of "Multivariate Adaptive Regression Splines" by J. H. Friedman," *Annals of Statistics*, 19, 102–112.

Smith, P. L. (1982), *Curve Fitting and Modeling with Splines Using Statistical Variable Selection Techniques*, Technical report, NASA Langley Research Center.

Subject Index

Syntax Index

Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **yourturn@sas.com**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **suggest@sas.com**.