Paper SAS3337-2019

SCORE! Techniques for Scoring Predictive Regression Models Using SAS/STAT® Software

Phil Gibbs and Randy Tobias, SAS Institute Inc.

ABSTRACT

Scoring new data to compute predictions for an existing model is a fundamental stage in the analytics life cycle. This task has never been easier, given recent additions to SAS/STAT® syntax. This paper shows how to use these new features in SAS/STAT software to make it simple to predict the response for new data. The paper discusses practical applications for scoring with the SCORE statement within a modeling procedure, with the PLM procedure using a stored model fit, and with the CODE statement in the DATA step. The paper also discusses tried-and-true methods of scoring using the SCORE procedure and using the missing-value-for-the-response trick.

INTRODUCTION

One of the most sought-after skills for a statistician is knowing how to create predictive regression models. This skill is also the core piece of the analytics life cycle. The key reason for developing a predictive regression model is to predict the response for data that is not observed in the model's training data set. Also known as model scoring, this prediction for unobserved data is a necessary element of evaluating the model.

This prediction also could be done for evaluating the model fit or for validating the model with a new set of data for which the response is known. You can score a simple in-sample prediction of combinations of the independent variables that are not observed. Finally, scoring can be used for simple out-of-sample forecasting to predict the future.

This paper covers the most common ways to score with a predictive regression model in SAS/STAT software. You will learn how to use methods, such as the SCORE and CODE statements, that are part of individual SAS procedures. Although the SCORE statement scores observations when the model is fit, the CODE statement works with a subsequent SAS DATA step to perform the scoring. The new STORE statement and the PLM procedure can score observations after a procedure creates a model without going through the expense of refitting. You will learn about older techniques, such as using macro variables to score a regression model yourself, as well as the tried-and-true technique of computing predictions by inserting data with missing values for the response. Finally, this paper briefly describes scoring models in SAS® Viya® software.

WHAT IS SCORING?

What exactly is scoring? Scoring is the task of producing predictions from a predictive model. Scoring requires three things:

1. a predictive model, that is, a mathematical method $f(x,\theta)$ for combining values of predictor variables x with values of certain quantities θ (also called the parameters of the model) in order to produce a predicted value for a target or response variable
2. specific values of the predictor variables x
3. specific values of the parameters θ

Predictive models can range from the simple to the complex. A model can be as simple as a linear regression or can become quite complex with the involvement of multivariate adaptive regression splines.

A simple linear model is just a linear combination of model variable and parameter values:

$f(x,\theta) = \theta_0 + x_1*\theta_1 + \ldots + x_p*\theta_p$

To score this model, all you need to know are the predictors and the parameters. PROC SCORE is a nice tool for working with this type of model.

With a more complicated model, more work needs to be done. A general linear model can be viewed as a linear combination of functions $f_i(x)$ of the predictors:

$f(x,\theta) = f_1(x)*\theta_1 + \ldots + f_p(x)*\theta_p$

SAS provides several methods for packaging up these functions into a form that allows for the creation of predicted values.

Something that you need to consider in deciding how to score is when the data will be scored. Some SAS/STAT techniques for scoring data work at the time the model is fit. Other techniques can be used to score new data after the model is fit, even when the original data is no longer available.

Table 1 shows the methods for scoring that are discussed in this paper (the methods are listed from the most general to the most specific).

Method	What	When	How
Missing-response trick	All model types	Fit time	Requires preprocessing of model fit data
PROC SCORE	Linear regression	Any time	PROC code
SCORE statement	Many model types	Fit time	PROC code
PROC PLM	Many model types	Any time	PROC code
CODE statement	Many model types	Any time	DATA step code

Table 1. Available Scoring Techniques in SAS/STAT Software

If you are ready to score new data at the point when you are fitting the model, then the SCORE statement or the missing-response trick should work just fine. If you need to score new data sometime after the model fit, possibly when the fitting data is no longer available, then you should use one of the techniques that packages up the model fit. If you want to do the scoring with a procedure call, then PROC PLM fills that need. If you want to do the scoring in a DATA step and possibly include calculations that could modify the predicted values, then the CODE statement is more appropriate.

SCORING DATA WITH THE PLM PROCEDURE

One of the most valuable scoring methods in SAS is to use the PLM procedure. PROC PLM was released with SAS® 9.2 in 2008, so it has been around for a while. This method can be used with most SAS modeling procedures, including the following:

- GEE
- GENMOD
- GLIMMIX
- GLM
- GLMSELECT
- LIFEREG
- LOGISTIC
- MIXED
- ORTHOREG
- PHREG
- PROBIT
- RELIABILITY (SAS/QC)
- SURVEYLOGISTIC
- SURVEYPHREG
- SURVEYREG

To score data with PROC PLM, use the STORE statement in one of the above procedures to store the model results. Then run PROC PLM to score a new set of data using that model. There are two noticeable advantages to working with PROC PLM. First, if you get into the habit of including a STORE statement in your modeling work, then you will never have to refit a model just to produce predictions for a new set of data. PROC PLM can do that without the expense of refitting the model. Second, you no longer need access to the original data that was used to create the model to create predictions for new data. All you need is the results of the model.

You should be aware of two concerns when you work with PROC PLM and the STORE statement:

1. The STORE statement stores the model results in an internal SAS file that cannot be modified in any way. Those model results are useful only with PROC PLM because they were created when the model was built.
2. The model results can be used only with PROC PLM on the same operating system on which the model results were created. If the model was built in a 64-bit Microsoft Windows environment, then those results can be used only with PROC PLM in a 64-bit Microsoft Windows environment. Cross-environment use is not allowed.

Here is a closer look at how PROC PLM works scoring a model created with PROC GLMSELECT. The following DATA step generates data for a model with a CLASS effect TRT (with three levels) and three covariates (X1, X2, and X3):

```
data YourModelData;
   call streaminit(614325);
   do trt=1 to 3;
      do rep=1 to ceil(rand('uniform')*35);
         x1=rand('uniform')*10;
         x2=rand('normal')*2;
         x3=rand('normal')*3;
         e=rand('normal');
         y=2 + trt + x1 + 0*x2 + 1.4*x3 + e;
         output;
      end;
   end;
run;
```

If you are not familiar with simulating data in SAS, the RAND function simulates data from a wide range of distributions. In the code above, data is simulated for the covariate X1 from a uniform distribution on the interval (0,10). Data is simulated for the covariates X2 and X3

from a standard normal distribution. The CALL STREAMINIT routine provides a seed value for the simulation so that the results shown here can be duplicated.

Here is the algebraic representation of the model that is used in the simulation:

$Y_{ij} = \beta_0 + TRT_i + \beta_1*X1_{ij} + \beta_2*X2_{ij} + \beta_3*X3_{ij} + \varepsilon_{ij}$

The simulated values that are used are as follows: $\beta_0=2$, $\beta_1=1$, $\beta_2=0$, and $\beta_3=1.4$. The TRT effect takes on the levels 1, 2, and 3. ε represents the residual that is added to each observation.

The DATA step above creates the data used to build the regression model. A second DATA step, creating the data set YourScoreData, is needed to create data that you want to score with this model:

```
data YourScoreData;
   call streaminit(6142352);
   do rep=1 to 20;
      trt=ceil(rand('uniform')*3);
      x1=rand('uniform')*10;
      x2=rand('normal')*2;
      x3=rand('normal')*3;
      y=.;
      output;
    end;
run;
```

Notice that the response, Y, for each observation in this data is set to missing. The response variable is not needed in this second data set. Adding the response variable, though, does no harm.

The syntax for scoring new data with PROC PLM is straightforward. You fit the model using PROC GLMSELECT and save the fitted model with a STORE statement to compute the scores later:

```
proc glmselect data=YourModelData;
   class trt;
   model y=trt x1 x2 x3 / selection=stepwise;
   store out=YourModel;
run;
```

PROC GLMSELECT creates a SAS item store that is called YourModel. (Although, in this example, the item store is saved to your Work library, you can use a LIBNAME statement to save these item stores to permanent locations.)

You use this SAS item store to score new data with PROC PLM. You can perform this scoring today, tomorrow, or even next year without having to refit the model and without needing access to the original data. The following PLM procedure creates the predicted values and shows additional flexibility in the PLM syntax:

```
proc plm restore=YourModel;
   score data=YourScoreData out=YourDataScored
         pred=Predicted lcl=Lower ucl=Upper;
run;
```

PROC PLM not only produces predicted values, but also provides upper and lower confidence bounds on that prediction. The SCORE statement in PROC PLM can even produce predictions from the output of a Bayesian analysis.

PROC PLM does not stop at producing just predictions and associated statistics. The procedure has options for producing graphs and performing post hoc tests on your model results. For more information, see the SAS/STAT documentation and Tobias and Cai (2010).

SCORING DATA WITH THE CODE STATEMENT

Want to create predictions with the DATA step? The CODE statement can help. The CODE statement creates DATA step code that contains the calculations necessary for creating predicted values from the observations in a SAS data set. This code is exportable. You can create the model on one operating system and score new data on another operating system.

The CODE statement is available in the following SAS/STAT procedures:

- GENMOD
- GLIMMIX
- GLM
- GLMSELECT
- LOGISTIC
- MIXED
- PLM
- REG

The syntax for the CODE statement has a single option, specifying the file in which you want to store the necessary code to create predictions in a DATA step:

```
code file='MyCode.sas';
```

Then you can include that code file in your SAS program today, tomorrow, or next year for when you want to score a new set of observations:

```
data ScoreObs;
   set DataToScore;
   %include 'MyCode.sas';
run;
```

This DATA step runs the code from that external SAS file and then scores the observations in the data set DataToScore. The data set must contain the same variables, using the same variable names, as the data set on which the model was developed.

The CODE statement creates a file that contains SAS programming statements with comments. Unlike the STORE statement, the results of the CODE statement are something that you can open and view and understand (and perhaps even modify).

A linear regression model shows the basic structure of the code that the CODE statement creates. The following DATA step produces observations with which you can fit a linear regression model. The PROC REG step fits the model and writes the code for creating predictions to the MyRegModel.sas file:

```
data RegData;
   call streaminit(8741235);
   do i=1 to 24;
      x1=rand('uniform')*10;
      x2=rand('uniform')*5;
      e=rand('normal');
      y=3 + 2*x1 + 1.5*x2 + e;
      output;
   end;
run;
```

(code continued)

```
proc reg data=RegData;
   model y=x1 x2;
   code file='MyRegModel.sas';
run;
quit;
```

The parameter estimates for this model are shown in Output 1.

Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	3.90121	0.44403	8.79	<.0001
x1	1	1.87751	0.07019	26.75	<.0001
x2	1	1.33105	0.12739	10.45	<.0001

Output 1. Parameter Estimates for Regression Model

These parameter estimates are used in the code that PROC REG produces. The following code, which scores new observations for this model, is exactly what PROC REG generates. The code is written to the FILE= location:

```
*****************************************;
** SAS Scoring Code for PROC REG;
*****************************************;

label P_y = 'Predicted: y' ;
drop _LMR_BAD;
_LMR_BAD=0;

*** Check interval variables for missing values;
if nmiss(x1,x2) then do;
   LMR_BAD=1;
   goto _SKIP_000;
end;

*** Compute Linear Predictors;
drop _LP0;
_LP0 = 0;

*** Effect: x1;
_LP0 = _LP0 + (1.87750737751729) * x1;
*** Effect: x2;
_LP0 = _LP0 + (1.33104533272017) * x2;

*** Predicted values;
_LP0 = _LP0 +     3.90121078413585;
_SKIP_000:
if _LMR_BAD=1 then do;
   P_y = .;
end;
else do;
   P_y = _LP0;
end;
```

The code checks to see whether any of the variables that are needed for the regression are missing. If variables are missing, no prediction is calculated. The predicted value is built, one regressor at a time, starting with the model effects X1 and X2. The intercept value is added in last. Notice that the code uses the full precision that is available for the parameter estimates. The code does not restrict the precision to that shown in the printed output from PROC REG (see Output 2).

The code for this model is available below for you to modify. You can remove effects if you need to create partial predictions. If you need a prediction just for the intercept and X1 effects, then you can remove the line of code that adjusts the prediction for X2.

A "linear predictor" is used in the code (abbreviated as LP) to store the calculation of the predicted value. In a linear regression, the LP is the predicted value itself. In generalized linear models, a link function is necessary to produce the predicted value. The following example demonstrates this.

The DATA step code below produces observations for a logistic regression model. There is a treatment effect, TRT, that has three levels and a covariate effect, X1. The LOGISTIC procedure step fits the model and provides the code that is necessary to produce predictions:

```
data LogisticData;
   call streaminit(25345278);
   do trt=1 to 3;
      do rep=1 to 20;
         x1=-rand('uniform')*2;
         logit=-1 + trt + 1.4*x1;
         p=exp(-logit)/(1+exp(-logit));
         if rand('uniform')>p then y=0; else y=1;
         output;
      end;
    end;
run;

proc logistic data=LogisticData;
   class trt / param=glm;
   model y=trt x1;
   code file='MyLogisticModel.sas';
run;
```

The code that is produced to score new observations is much more complicated than what is needed to score a linear regression model. The reason that it is more complicated is not only the logistic model but also the classification variable TRT. A full discussion of the concepts in this code is beyond the scope of this paper, but these two features are worth some special attention.

1. This logistic model involves a CLASS effect, TRT. The code must construct design variables for the levels of the TRT effect. Those design variables are used to determine which parameter estimate for the TRT effect applies to the predicted value.
2. A logistic regression model includes a logit link function. This link function is applied to the linear predictor to produce a predicted probability for each level of the response. The response level with the highest predicted probability is made the predicted response for each observation.

Output 2 contains the last 10 observations of the LogisticData data set, after the scoring code is applied to that data. The variables P_y0 and P_y1 contain the predicted probabilities for the two levels of the response. If P_y0>.5, then the predicted response, I_y, is set to 0. If P_y1>.5, then the predicted response is set to 1. For these 10 observations, the model does a nice job of predicting the response. Only observation 51 has an inaccurate prediction.

Obs	trt	rep	x1	logit	p	y	I_y	U_y	P_y0	P_y1
51	3	11	-1.64040	-0.29656	0.57360	0	1	1	0.48056	0.51944
52	3	12	-0.78651	0.89889	0.28928	0	0	0	0.73697	0.26303
53	3	13	-1.00089	0.59875	0.35463	0	0	0	0.67963	0.32037
54	3	14	-1.97502	-0.76503	0.68244	1	1	1	0.37472	0.62528
55	3	15	-1.59781	-0.23694	0.55896	1	1	1	0.49437	0.50563
56	3	16	-0.01961	1.97255	0.12212	0	0	0	0.88344	0.11656
57	3	17	-0.91860	0.71395	0.32873	0	0	0	0.70242	0.29758
58	3	18	-1.70002	-0.38002	0.59388	1	1	1	0.46129	0.53871
59	3	19	-0.55457	1.22360	0.22730	0	0	0	0.79105	0.20895
60	3	20	-0.54739	1.23365	0.22554	0	0	0	0.79258	0.20742

Output 2. LogisticData Data Set with the Scoring Code Applied

Modifications to prediction code like the code used here are difficult because of the code complexity. The code can be modified to create partial predictions, but make sure that you fully understand what the prediction code is doing before you attempt any modifications.

The full scoring code is in Appendix 1.

SCORING DATA USING MACRO VARIABLES FOR MODEL PARAMETERS

The CODE statement provides a convenient method for scoring new observations with a statistical model. What if the procedure you need to use lacks this statement, or you just want to develop the scoring model yourself?

All SAS modeling procedures write the parameter estimates for the model to an output data set. These parameter estimates can be stored in SAS macro variables, and those macro variables can be used in SAS DATA step code to create predictions.

Recall the DATA step that created data for the section on scoring models with PROC PLM. The MIXED procedure can also be used to fit a model against this data:

```
proc mixed data=YourModelData;
   class trt;
   model y=trt x1 x2 x3 / solution;
   ods output solutionf=ParmEsts;
run;

proc print data=ParmEsts;
run;
```

The parameter estimates table, as stored in the data set ParmEsts, is shown as Output 3.

Obs	Effect	trt	Estimate	StdErr	DF	tValue	Probt
1	Intercept	_	6.1572	0.9495	27	6.48	<.0001
2	trt	1	-3.2252	0.8799	27	-3.67	0.0011
3	trt	2	-2.2725	0.9460	27	-2.40	0.0234
4	trt	3	0	.	.	.	.
5	x1	_	0.9911	0.05594	27	17.72	<.0001
6	x2	_	0.1341	0.08654	27	1.55	0.1329
7	x3	_	1.3908	0.04616	27	30.13	<.0001

Output 3. Parameter Estimates from PROC MIXED

The power of the SQL procedure can extract these parameter estimates from this data set and store them in SAS macro variables. Here is the SQL code:

```
proc sql noprint;
   select count(*)into :nobs from ParmEsts;
   select Estimate into :Beta_1 - :Beta_%sysfunc(strip(&nobs)) from
   ParmEsts;
quit;
```

The code counts the number of parameter estimates in the data set ParmEsts and then stores that value in the SAS macro variable &NOBS. The second SELECT statement reads each of the observation values for the ESTIMATE variable. Those values are then stored in the SAS macro variables &Beta_1, &Beta_2, and so on.

To check that you have stored the parameter estimates correctly, the macro %PrintBetas writes out the values of the parameter estimate macro variables:

```
%macro PrintBetas;
   %do i=1 %to &nobs;
      %put &&Beta_&i;
   %end;
%mend;
%PrintBetas;
```

The final action is to build the prediction model in a SAS DATA step. If the model is a linear regression, then the SAS code that is needed to calculate the predicted value is easy to write. This model, however, involves a CLASS variable.

The following code determines the appropriate parameter estimate to use for the TRT effect and assigns that value to the prediction:

```
data YourDataScored;
   set YourModelData;
   if trt=1 then TrtEffect=&Beta_2;
   else if trt=2 then TrtEffect=&Beta_3;
   else TrtEffect=&Beta_4;
   PredY = &Beta_1 + TrtEffect + x1*&Beta_5 + x2*&Beta_6 + x3*&Beta_7;
run;
```

The PredY variable stores the predicted value, adding in the intercept (&Beta_1) and the covariate components for the prediction.

One advantage to this technique is that you can easily construct partial predictions when you want to look only at the TRT effect or at the prediction accounting for a set of covariates. A disadvantage is that you cannot easily construct standard errors for these predictions.

SCORING DATA WITH THE SCORE STATEMENT

The SCORE statement discussed in the section on scoring with PROC PLM is also available in some SAS/STAT procedures that do direct modeling. Here are some of the procedures that can use the SCORE statement:

- ADAPTIVEREG
- COUNTREG (SAS/ETS)
- GAM
- GLMSELECT
- KDE
- LOESS
- LOGISTIC
- PLM
- TPSPLINE

The syntax for the SCORE statement is simple and easy to use. Use the DATA= option to specify the data set that you want to score. Use the OUT= option to specify the data set to contain the predictions, along with all the variables from the DATA= data set. This method is convenient when you need to generate predictions quickly, in-line with estimating the model. Multiple SCORE statements can score different data sets, and predictions are created for observations that have missing responses.

The following model was developed in the previous section on PROC PLM. You can build the model and score a new data set in one step with code like the following:

```
proc glmselect data=YourModelData;
   class trt;
   model resp=trt x1 x2 x3;
   score data=YourScoreData out=YourDataScored;
run;
```

This code builds a model using the data in YourModelData. It then scores the data in YourScoreData and stores the new scored responses in YourDataScored. This technique comes in very handy when you need to quickly produce new responses for either a new set of data or for a holdout sample to test your model fit.

Output 4 shows a partial listing of the output from the scoring process.

Obs	rep	trt	x1	x2	x3	y	p_y
1	1	2	5.31117	1.16800	0.49239	.	9.8301
2	2	3	3.61597	-4.22169	0.54214	.	10.2308
3	3	3	0.25733	3.15060	2.80126	.	10.0062
4	4	3	6.81606	0.43551	-1.28190	.	10.8997
5	5	1	2.14987	2.88828	-0.69783	.	4.0555
6	6	3	5.32001	-0.26400	-0.75993	.	10.1287
7	7	2	4.67343	-0.40099	0.06066	.	8.5949
8	8	2	5.85452	-1.31424	-0.41285	.	9.1185
9	9	1	8.67574	1.69846	1.75475	.	13.9720
10	10	2	4.11072	-0.54709	-4.04628	.	2.3424

Output 4. Partial Output of Regression Model Predictions with the SCORE Statement

The SCORE statement stores the predictions in P_Y, taking the name of the response variable from the model and adding "P_" to the response variable name. You can specify your own variable name for the predictions with the PRED= option. You can also use the RESID= option to specify a variable for the residuals if you have nonmissing values for the response.

The SCORE statement in PROC LOGISTIC is somewhat unique. You can do much more in terms of scoring data with this version of the SCORE statement. In addition to predicted values, the SCORE statement in PROC LOGISTIC can also produce confidence limits, fit statistics (when the response variable is nonmissing in the new data set), and data for the receiver operator characteristic (ROC) curve when the response is binary.

PROC LOGISTIC offers another trick for producing predictions and associated statistics. The OUTMODEL= option in the PROC LOGISTIC statement can be used to save the model results in a compact (and nonmodifiable) form. You can then run PROC LOGISTIC a second time, using the INMODEL= option. This action scores the data set that you specify in the SCORE statement's DATA= option. There are quite a few statements that do not work in PROC LOGISTIC when you use this approach. So this method of scoring is most useful when you want to produce only predicted values.

SCORING DATA THE OLD-FASHIONED WAY

The final scoring technique to be discussed is in fact the oldest method in SAS, and it is available for any SAS modeling procedure. The idea is to combine your scoring data with your modeling data into a single SAS data set. If you are a longtime SAS user, then this technique might already be familiar. This method was the only scoring technique available in many early SAS procedures.

The following code generates data for a linear regression and fits a model using PROC REG:

```
data RegData;
   call streaminit(661346);
   do i=1 to 47;
      x1=rand('uniform')*5;
      x2=rand('uniform')*3;
       x3=rand('uniform')*4;
       e=rand('normal');
       y = 2 + 2*x1 + 1.5*x2 - 2*x3 + e;
       output;
    end;
run;

proc reg data=RegData;
   model y=x1 x2 x3;
run;
```

Output 5 shows the results of the PROC REG model fit. Notice that 47 observations were read and used from the input data set and that the corrected total sum of squares (SS) = 777.79156.

Number of Observations Read	47
Number of Observations Used	47

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	745.30281	248.43427	328.81	<.0001
Error	43	32.48875	0.75555		
Corrected Total	46	777.79156			

Output 5. Model Fit Results from PROC REG

Now suppose you want to obtain predictions for a new set of observations. You could use any of the techniques previously discussed, or you can simply merge these new observations with the existing data. If you set the response value to missing, then none of these new observations affect your model fit. This code creates eight new observations with a missing response, adds the new observations to the existing data, and refits the model:

```
data ScoreData;
   input x1 x2 x3 @@;
   y = .;
datalines;
2.50 1.13 1.84   3.78 2.43 2.49
1.75 2.27 2.79   3.19 0.65 0.41
1.28 1.58 0.24   0.03 2.10 1.89
3.54 0.75 0.03   3.70 0.14 1.74
;
run;
```

(code continued)

```
data AllData;
   set RegData ScoreData;
run;

proc reg data=AllData;
   model y=x1 x2 x3;
   output out=Scores p=pred r=resid stdp=stderr;
run;
```

The code uses the power of the SAS SET statement to merge the two data sets together. The OUTPUT statement in PROC REG scores all observations in the input data set and writes those predictions, along with the residuals and standard errors of the mean prediction, to the Scores data set.

PROC REG can read all 55 observations from the data set AllData. However, the procedure uses only the original 47 observations to fit the model, because the response variable Y for the new eight observations in ScoreData is set to missing. That is the key distinction here. Setting those responses to missing enables you to fit the model without the new observations affecting the fit.

Output 6 shows a partial printout (the last 16 observations) of the Scores data set. The predictions are there for the new observations, where the response is missing, and you also get standard errors for these predictions. Residual values are not available for these new observations because you have no actual response to compare with the prediction.

Obs	i	x1	x2	x3	e	y	pred	stderr	resid
40	40	1.68548	1.63200	2.46655	0.49406	3.3799	3.0415	0.16985	0.33841
41	41	4.64151	2.38339	1.16785	-0.20298	12.3194	12.5706	0.25891	-0.25123
42	42	2.91302	1.50175	0.63604	-0.06028	8.7463	8.8767	0.20402	-0.13044
43	43	0.23684	1.29598	3.36870	-0.98972	-3.3095	-2.1040	0.29425	-1.20547
44	44	4.60396	2.39599	0.19916	-0.27797	14.1256	14.4278	0.31217	-0.30213
45	45	1.60528	2.52620	1.30903	0.68397	7.0658	6.5238	0.25622	0.54201
46	46	1.53546	2.15619	0.10239	-1.94655	6.1539	8.2069	0.31333	-2.05301
47	47	1.82643	2.08163	2.48953	1.33225	5.1285	3.9556	0.19085	1.17288
48	.	2.50000	1.13000	1.84000	.	.	5.1227	0.13905	.
49	.	3.78000	2.43000	2.49000	.	.	8.3335	0.22279	.
50	.	1.75000	2.27000	2.79000	.	.	3.4975	0.22201	.
51	.	3.19000	0.65000	0.41000	.	.	8.5776	0.25925	.
52	.	1.28000	1.58000	0.24000	.	.	6.5582	0.29156	.
53	.	0.03000	2.10000	1.89000	.	.	1.6259	0.30504	.
54	.	3.54000	0.75000	0.03000	.	.	10.1691	0.29059	.
55	.	3.70000	0.14000	1.74000	.	.	6.1845	0.25880	.

Output 6. Predictions for the Last 16 Observations in the Scores Data Set

The technique of appending observations to score is a tried-and-true method. One disadvantage is that in order to score new data, you need to refit the model. If the model takes hours to estimate, there is no way to avoid that cost using this technique.

PROBLEMS IN SCORING NEW OBSERVATIONS

Situations arise where predictions will not be possible for new observations. One of the more common situations is when the data contains missing values for the predictors. If any of the variables involved with the right-hand side of the model are missing, then no prediction for that observation can be produced. There are techniques for overcoming this deficiency in the data. Missing values in the data can be imputed, using the statistical information that exists in the nonmissing observations. A common technique is multiple imputation, where more than one set of plausible values are imputed for the missing variables. The MI procedure can help with such multiple imputation. With the MIANALYZE procedure, you can combine the results from these multiple imputations to obtain valid statistical inferences. For more information, see "The MI Procedure" and "The MIANALYZE Procedure" chapters in the SAS/STAT documentation (SAS Institute Inc. 2018a; SAS Institute Inc. 2018b).

If the model contains CLASS variables, then the data to be scored cannot contain any new levels to those CLASS variables. If your model has a CLASS effect with three types of treatment and the scoring data has a fourth treatment type, then you cannot create predictions for those observations. There is no way to fix this issue.

Defining spline effects with the EFFECT statement makes it easy to fit nonparametric models to your data. But there are some points to watch out for when it comes to scoring. First, for nonparametric models, the variable values for the scoring observations should be within the range of variable values used for the model. Scoring spline models on observations that are outliers can lead to unreliable predictions. Another caveat about models with spline effects is that you cannot use the CODE statement with them to generate DATA step scoring code. Use one of the other scoring methods instead, such as providing the stored model fit to PROC PLM. For more information about using the EFFECT statement to define models, see Gibbs et al. (2013).

SCORING IN SAS® VIYA®

Here is a peek at the most recent developments in scoring in SAS® Software. This paper has discussed scoring with predictive statistical models strictly within traditional SAS products, such as SAS/STAT. Scoring is also a fundamental task for analytics built with SAS Viya. Accordingly, in SAS Viya you have various approaches to scoring that are similar to those discussed here. And at the top of the SAS Viya scoring heap are the **`aStore`** action and the corresponding ASTORE procedure.

Scoring with PROC ASTORE for SAS Viya is portable, much like PROC PLM. However, there are no limitations due to the type of operating system or the version of SAS Viya in which the model has been trained. You can manage your PROC ASTORE models with SAS® Model Manager and use them to score new data in parallel inside SAS or SAS Viya, from within a database, or even in a SAS® Event Stream Processing instance. You can also store your model on your machine in a universal format that is sufficient for scoring on any other machine. PROC ASTORE supports a broad range of SAS Viya analytical models, from random forest to deep neural networks in addition to statistical regression procedures. For more information about PROC ASTORE, see *SAS® Visual Data Mining and Machine Learning 8.3: Procedures* (SAS Institute Inc. 2018c) and *SAS® Visual Data Mining and Machine Learning 8.3: Programming Guide* (SAS Institute Inc. 2018d).

CONCLUSION

This paper presented many techniques for scoring new data with SAS/STAT software. Although the methods of appending scoring data to the model data and of creating macro variables from your model parameters can work well, the methods involving the SCORE statement or PROC PLM are more attractive for their simplicity and their reduced cost. They are also far more versatile. The SCORE statement and PROC PLM can avoid the cost of refitting the model when new predictions are needed.

As always, though, use the approach that makes the most sense for your model and data.

APPENDIX 1. SCORING CODE FOR LOGISTIC REGRESSION MODEL

```
*****************************************;
** SAS Scoring Code for PROC Logistic;
*****************************************;

length I_y $ 12;
label I_y = 'Into: y' ;
label U_y = 'Unnormalized Into: y' ;

label P_y0 = 'Predicted: y=0' ;
label P_y1 = 'Predicted: y=1' ;

drop _LMR_BAD;
_LMR_BAD=0;

*** Check interval variables for missing values;
if nmiss(x1) then do;
   _LMR_BAD=1;
   goto _SKIP_000;
end;

*** Generate design variables for trt;
drop _0_0 _0_1 _0_2 ;
_0_0= 0;
_0_1= 0;
_0_2= 0;
length _st12 $ 12; drop _st12;
_st12 = left(trim(put (trt, BEST12.)));
if _st12 = '1'  then do;
   _0_0 = 1;
end;
else if _st12 = '2'  then do;
   _0_1 = 1;
end;
else if _st12 = '3'  then do;
   _0_2 = 1;
end;
else do;
   _0_0 = .;
   _0_1 = .;
   _0_2 = .;
   _LMR_BAD=1;
```

(code continued)

```
   goto _SKIP_000;
end;

*** Compute Linear Predictors;
drop _LP0;
_LP0 = 0;

*** Effect: trt;
_LP0 = _LP0 + (-2.23097285884297) * _0_0;
_LP0 = _LP0 + (-1.37571829625743) * _0_1;
*** Effect: x1;
_LP0 = _LP0 + (1.29765942262973) * x1;

*** Predicted values;
drop _MAXP _IY _P0 _P1;
_TEMP = 2.05089016000076  + _LP0;
if (_TEMP < 0) then do;
   _TEMP = exp(_TEMP);
   _P0 = _TEMP / (1 + _TEMP);
end;
else _P0 = 1 / (1 + exp(-_TEMP));
_P1 = 1.0 - _P0;
P_y0 = _P0;
_MAXP = _P0;
_IY = 1;
P_y1 = _P1;
if (_P1 >  _MAXP + 1E-8) then do;
   _MAXP = _P1;
   _IY = 2;
end;
select( _IY );
   when (1) do;
      I_y = '0' ;
      U_y = 0;
   end;
   when (2) do;
      I_y = '1' ;
      U_y = 1;
   end;
   otherwise do;
      I_y = '';
      U_y = .;
   end;
end;
_SKIP_000:
if _LMR_BAD = 1 then do;
I_y = '';
U_y = .;
P_y0 = .;
P_y1 = .;
end;
drop _TEMP;
```

REFERENCES

Gibbs, Phil, Randy Tobias, Kathleen Kiernan, and Jill Tao. 2013. "Having an EFFECT: More General Linear Modeling and Analysis with the new EFFECT Statement in SAS/STAT® Software." *Proceedings of the SAS Global Forum 2013 Conference*. Cary, NC: SAS Institute Inc. Available at **support.sas.com/resources/papers/proceedings13/437-2013.pdf**.

SAS Institute Inc. 2018a. "The MI Procedure." In *SAS/STAT® 15.1 User's Guide.* Cary, NC: SAS Institute Inc. Available at **go.documentation.sas.com/?docsetId=statug&docsetTarget=statug_mi_toc.htm&docsetVersion=15.1&locale=en**.

SAS Institute Inc. 2018b. "The MIANALYZE Procedure." In *SAS/STAT® 15.1 User's Guide.* Cary, NC: SAS Institute Inc. Available at **go.documentation.sas.com/?docsetId=statug&docsetTarget=statug_mianalyze_toc.htm&docsetVersion=15.1&locale=en**.

SAS Institute Inc. 2018c. *SAS® Visual Data Mining and Machine Learning 8.3: Procedures.* Cary, NC: SAS Institute Inc. Available at **go.documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.4&docsetId=casml&docsetTarget=titlepage.htm&locale=en**.

SAS Institute Inc. 2018d. *SAS® Visual Data Mining and Machine Learning 8.3: Programming Guide.* Cary, NC: SAS Institute Inc. Available at **go.documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.4&docsetId=casactml&docsetTarget=titlepage.htm&locale=en**.

Tobias, Randy, and Weijie Cai. 2010. "Introducing PROC PLM and Postfitting Analysis for Very General Linear Models in SAS/STAT® 9.22." *Proceedings of the SAS Global Forum 2010 Conference.* Cary, NC: SAS Institute Inc. Available at **support.sas.com/resources/papers/proceedings10/258-2010.pdf**.

ACKNOWLEDGMENTS

The authors acknowledge Arash Banadaki for the material on scoring in SAS® Viya® with PROC ASTORE.

RECOMMENDED READING

For more information about scoring and other statistical topics, see Rick Wicklin's excellent SAS blog "The DO Loop," **blogs.sas.com/content/iml/**.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Phil Gibbs
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Email: support@sas.com
Web: support.sas.com/en/support-home.html

Randy Tobias
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Email: Randy.Tobias@sas.com
Web: support.sas.com/en/support-home.html