Paper 1829-2014

Nonnegative Least Squares Regression in SAS®

Matthew Duchnowski, Educational Testing Service

ABSTRACT

It is often the case that parameters in a predictive model need to be restricted to an interval that is either reasonable or necessary given the model's application. A simple and classic example of such a restriction is the regression model which requires that all parameters be positive. In the case of multiple linear least squares regression (MLR), the resulting model is therefore strictly additive and, in certain applications, not only appropriate but also intuitive. This special case is commonly referred to as a "nonnegative least squares" (NNLS) regression. While Base SAS® contains a multitude of ways to perform a least squares regression (PROC REG and PROC GLM, to name two), there exists no native SAS procedure to conduct a nonnegative least squares regression. The author offers a concise way to conduct the nonnegative least squares analysis by using PROC NLIN (proc "nonlinear"). PROC NLIN offers parameter restriction to the user. By fashioning a linear model in the framework of a nonlinear procedure, the end result can be achieved. As an additional corollary, the author will show how to calculate the R^2 statistic for the resulting model, which has been left out of the PROC NLIN output for the reason that it is not interpretable in most nonlinear cases.

INTRODUCTION

The NNLS method seeks to produce an optimal model while constraining model weights to either the set of positive real numbers or to the trivial weight of zero.

It should be noted that some applications of NNLS could require an additional constraint on the model intercept. The NNLS method we will explore here implements no such restriction.

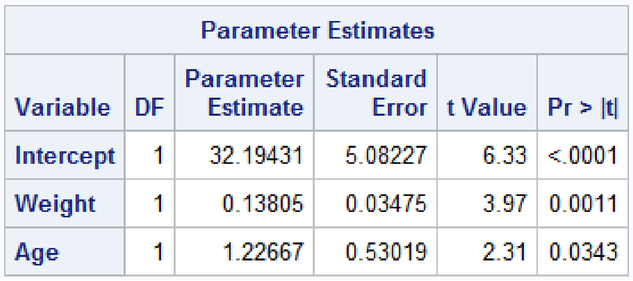
SIMPLE EXAMPLE

For the purposes of illustration, we will begin with an example using the SASHELP dataset "Class". This small dataset contains nineteen observations of student NAME, SEX, AGE, HEIGHT, and WEIGHT.

Suppose we wish to model student height using weight and age as predictors. PROC REG will suffice for this task and can be carried out as:

```
proc reg data=sashelp.class;
    model height = weight age ;
quit;
```

The default output for PROC REG displays a variety of information, but here we will focus on the resulting parameter estimates and standard errors.



Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	32.19431	5.08227	6.33	<.0001
Weight	1	0.13805	0.03475	3.97	0.0011
Age	1	1.22667	0.53019	2.31	0.0343

Table 1. Parameter estimates from PROC REG

The same linear model can be produced using, PROC NLIN, as shown below:

```
proc nlin data=sashelp.class method=Gauss;
    parms  parm_est0=1 parm_est1=1 parm_est2=1;
    bounds parm_est1   parm_est2    >= 0;
    model  height = parm_est0 + parm_est1*weight + parm_est2*age;
run;
```

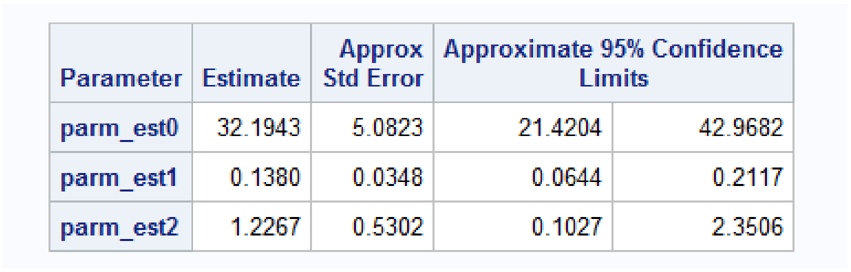
Here we have placed bounds on our parameters and in doing so, are required to name the parameters explicitly. The chosen naming convention here is `parm_est0` for the model intercept and `parm_estN` for the *N*th predictor variable in the model.

In order to arrive at an optimal solution, PROC NLIN uses an iterative method (explicitly chosen above as Gauss) and so must be given an initial value. The PARMS statement declares the parameter names and assigns this value. Note that the parameter representing our model intercept also needs to be declared and initialized. Choosing an appropriate starting value will improve the rate at which the iterative method will converge to a solution.

Next, the BOUNDS statement restricts the parameter estimates to be greater than or equal to zero. As mentioned earlier, we won't be restricting the model's intercept, and so it is not listed here.

The METHOD option explicitly chooses an iteration method. The models produced by different iteration methods should not differ from one another. There are cases, however, when a particular iteration method will not converge to a final set of regression weights. The author has had success with METHOD=Gauss and chooses to use that option in these examples.

Because the original (unrestricted) linear model had no negative parameter estimates, the PROC NLIN code was able to produce the identical model using a single iteration. Shown in the output are the original estimates and a set of "approximate" standard errors which match our original.



Parameter	Estimate	Approx Std Error	Approximate 95% Confidence Limits	
parm_est0	32.1943	5.0823	21.4204	42.9682
parm_est1	0.1380	0.0348	0.0644	0.2117
parm_est2	1.2267	0.5302	0.1027	2.3506

Table 2. Parameter Estimates from PROC NLIN

This example was chosen to illustrate the syntax of PROC NLIN. Practically speaking, the code is trivial as application of the boundaries produces no real effect. Next we will consider an example where our results differ for PROC REG.

ENHANCING THE CODE

The next example comes from the SASHELP dataset "Cars", which lists sales data and model specifications for a collection of cars. We will try to model the manufacturer's suggested retail price (MSRP) using the size of the car's engine and the fuel efficiency ratings (MPG - city and highway) as predictor variables.

We should note that for this example and the previous one, the number of predictors was quite small for using the NNLS method. On the contrary, NNLS is generally well-suited for scenarios where the number of predictor variables is large. To develop code that is conducive to this scenario, we will create some macro variables that will simplify processing.

CREATE SOME HELPER VARIABLES

It might be convenient for the user to enter the name of the response variable along with a space-delimited list of predictor variable names.

```
%let PredictorVars = EngineSize MPG_City MPG_Highway;
%let ResponseVar   = MSRP;
```

This makes the PROC REG statement slightly easier.

```
proc reg data=sashelp.cars;
    model &ResponseVar. = &PredictorVars.;
quit;
```

The real benefit, however, in creating these macro variables is in simplifying the PROC NLIN code. This can be done by first counting the number of predictor variables and capturing this number in the macro variable `CountPred`.

```
data _null_;
   call symput("CountPred", STRIP(put(countw("&PredictorVars.", ' '), 3.0)));
run;
```

Next we will create the macro variable `ModelString`, which holds the entire equation needed for the MODEL statement.

```
data _null_;
   length myString $ 1000;
   array parm{&CountPred.} $ 20 parm1 - parm&CountPred.;
   myString = strip("&ResponseVar.") || '=parm_est0';

   do i=1 to &CountPred.;
      parm{i} = scan("&PredictorVars.", i, ' ');
      myString =
         strip(myString) || ' + parm_est' || strip(i) || '*' || strip(parm{i});
   end;

   call symput("ModelString", strip(myString));
run;
```

The PROC NLIN code segment is now shown with our new macro variables and two additional output datasets.

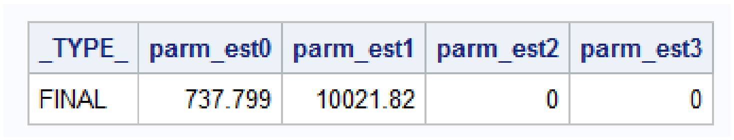
```
proc nlin data=sashelp.cars
   method = Gauss
   outest=NNLS_Weights(where=(_TYPE_='FINAL')
   keep=_TYPE_ parm_est0-parm_est&CountPred.) save ;

    parms  parm_est0-parm_est&CountPred.  = 1;
    bounds parm_est1-parm_est&CountPred. >= 0;
    model  &ModelString.;
    output out=Predictions predicted=predicted residual=residual;
run;
```

The output dataset specified by OUTEST will capture our model parameters and the SAVE option ensures that the final estimates are captured in that data set. By using the WHERE statement, we interest ourselves only in these final parameter estimates and filter out all other intermediate estimates.

The dataset "Predictions" specified in the OUTPUT statement will capture all records of our original dataset, but with additional columns containing the predicted values for each observation as well as the residual values.

We can inspect our weights by running a PROC PRINT on the dataset “NNLS_Weights”.



TYPE	parm_est0	parm_est1	parm_est2	parm_est3
FINAL	737.799	10021.82	0	0

Table 3. Parameter Estimates from NNLS

It is worth noting that the NNLS model weights are identical to the weights that are generated directly by running a PROC REG that uses only the first parameter in the model:

```
proc reg data=sashelp.cars;
   model MSRP = EngineSize;
quit;
```

Though this syntax is more straightforward, the real power of NNLS lies in arriving at the final set of predictors that will yield a set of nonnegative weights, a task that grows more difficult when a large number of predictors are attempted.

FIT STATISTICS

It is often not appropriate to determine the quality of fit for a nonlinear model using R^2 and so PROC NLIN does not produce this statistic by default. However, because we are using this PROC to produce a linear model, it may be of interest to produce _RSQUARE_ in the same way it is generated in PROC REG/GLM. The output dataset “Predictions” can be used for this purpose, as it contains predicted values and residuals. The residual sum of squares (SSE), the explained sum of squares (SSR), and the total sum of squares (SST) can be obtained directly and used to calculate _RSQUARE_ as shown below:

```
proc sql;

  select

     _SSE_,
     _SSR_,
     _SSE_+_SSR_ as _SST_ ,
     1-_SSE_/(_SSE_+_SSR_) as _RSQUARE_

   from (
      select
         SUM(A.residual**2)  as _SSE_ ,
         SUM((A.predicted-B.Ybar)**2) as _SSR_ ,
         SUM((A.&ResponseVar.-B.Ybar)**2) as _SST_
      from
         Predictions as A,
         (select MEAN(&ResponseVar.) as Ybar from predictions) as B
   )
   ;
quit;
```

CONCLUSION

Though there is no explicit procedure in SAS for generating a NNLS model, PROC NLIN offers plenty of tools for the user to leverage. Users should keep in mind that the iterative method does not always converge and it might be necessary to choose an alternate iterative methods in the METHOD option (i.e. GAUSS, MARQUARDT, NEWTON, GRADIENT and DUD). All of these methods, when successful, will converge to the same set of parameter estimates.

REFERENCES

Chen, D., & Plemmons, R. J. (n.d.). *Nonnegativity constraints in numerical analysis.* Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.157.9203&rep=rep1&type=pdf

SAS Institute. (2009). *SAS global forum 2009: Table of contents.* Retrieved from support.sas.com/resources/papers/proceedings09/TOC.html

SAS Institute. (2014). PROC NLIN statement. In *SAS/STAT® 9.2 user's guide* (2nd ed.). Retrieved from http://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_nlin_sect007.htm

UCLA, Institute for Digital Research and Education. (n.d.). *SAS library: Nonlinear regression in SAS.* Retrieved from http://www.ats.ucla.edu/stat/sas/library/SASNLin_os.htm

ACKNOWLEDGMENTS

I would like to thank all friends and colleagues at ETS who are endlessly willing to discuss topics such as those discussed here, particularly Florian Lorenz. I would like to especially thank my wife Robin, who provides me with unending support in all that I do.

CONTACT INFORMATION

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

Matthew Duchnowski
Educational Testing Service
Rosedale Rd, MS 20T
Princeton, NJ 08541
(609) 683-2939
mduchnowski@ets.org