SESUG Paper 298-2018

Computing Predicted Values and Residuals from Data Sets Containing Classification Variables with Large Numbers of Levels

Jamie McClave Baldwin, Ph.D., Ramon Littell, Ph.D. and James McClave, Ph.D
Info Tech® Inc. | Consulting

ABSTRACT

Linear models frequently include terms to reduce bias in parameter estimation and extraneous variation. Large data sets typically contain observations from multiple sources, such as different locations, time periods or product types. If one or more of the model variables is a classification (*CLASS)* variable, then the required computations might overwhelm memory capacity and disable computation of model coefficients. This problem can be partially solved by using the *ABSORB* statement in the *GLM* procedure, which enables estimation of coefficients of non-classification variables. But it does not permit computation of linear combinations involving coefficients of *CLASS* variables, such as predicted values or comparisons of levels of among *CLASS* variables. This paper demonstrates a computational method that can be carried out in data steps that accomplishes the same objectives as the *ABSORB* statement, but enables computation of *predicted values* and *residuals*. An illustrative data set contains prices of machine products that were sold to multitudes of customers. The data were collected to estimate the effects of product cost, demand and a possible change in economic environment on price, adjusted for product effects. (Data for the example were simulated to represent a real data.) The *HPmixed* procedure has capabilities that overcome the shortcomings of the GLM ABSORB statement, but uses more computational resources, as is briefly shown.

INTRODUCTION

Econometric models are commonly used in antitrust litigation to provide evidence of collusive and illegal price fixing. In one such case data were collected from hundreds of thousands of financial transactions in sales of machine parts. Data from each transaction included a price of the item sold, identification of the type of product, date of the transaction, economic indices of cost to the seller and of demand in the product marketplace. Typically, litigation is the result of a "class action" lawsuit brought by a group of customers against sellers of the products. Usually, the illegal collusive price fixing is alleged to have occurred during a prescribed time segment, called the *DAMAGES period*. Evidence of collusion is obtained by comparing the actual prices during the damages period with prices that would have occurred had there been no collusion. The latter are called *But-for* prices; that is, prices that would have been charged in a free market with no collusive activity. Two statistical models of price are constructed, corresponding to the actual prices and the But-for prices. Ordinarily, linear models are used to represent both the actual and But-for prices. The two models can be simultaneously represented by one linear equation, wherein a term or set of terms, distinguishes the representation of the actual prices from the but-for prices. Here are the two models in standard statistical terms. First is the so-called *But-for* model representing the economic relationship between the prices and effects due to cost and demand factors in a fair market.

But-for model:

$$y_{ij} = \alpha_i + \beta_{cost} * x_{cindx,ij} + \beta_{demand} * x_{dindx,ij,} + e_{ij} ,$$

where y_{ij} = logarithm of price of product *i* in transaction *j*, and

α_i = intercept of the regression for Product *i*,
β_{cost} = expected effect of cost index on log price
$x_{cindx,ij}$ = value of cost index at the time of transaction *j*,
β_{demand} = expected effect of demand index on log price
$x_{dindx,ij,}$ = value of demand index at the time of transaction *j*,
e_{ij} = random error.

Second is the so-called *Damages model* that includes, in addition to all the terms in the But-for model, a set of terms that represent price changes (damages) due to presumed collusive activity during the class period.

Damages model:

$$y_{ij} = \alpha_i + \beta_{cost} * x_{cindx,ij} + \beta_{demand} * x_{dindx,ij,} + \tau * d + \pi_{cost} * d * x_{cindx,ij} + \pi_{demand} * d * x_{dindx,ij} + e_{ij},$$

where (in addition to the terms in the But-for model),

d = 1 if the transaction is in the damages period and$d$ = 0 if the transaction is *not* in the damages period, and
τ = the basal effect of collusion,
π_{cost} = increase in effect of cost index due to collusion, and
π_{demand} = increase in effect of demand index due to collusion.

A statistical model for the data in matrix notation is

1. $\boldsymbol{Y} = \boldsymbol{D\alpha} + \boldsymbol{d\delta} + \boldsymbol{X\beta} + \boldsymbol{e}$,

where
Y is the vector of LogPrice values,
D is the matrix of indicator variables corresponding the Products,
d is the vector of indicator variable for presence (*d = 1*) or absence (*d = 0*) in the time periods of econometric change,
X is the matrix of values of CostIndex and DemandIndex,
and
e is the vector of random errors.

The model (1) can be written more compactly using partitioned matrices as

2. $\boldsymbol{Y} = \boldsymbol{U\vartheta} + \boldsymbol{e}$,

where $\boldsymbol{U} = [\boldsymbol{D}: \boldsymbol{d}: \boldsymbol{X}]$ and $\boldsymbol{\vartheta' = [\alpha': \delta': \beta']'}$.

The method of Ordinary Least Squares (OLS) is most commonly used to obtain estimates of the parameters in linear models. Pertaining to model (2) the estimates of the parameters
in equation (1), would be given by evaluating

3. $\boldsymbol{T = (U'U)^{-1} U' Y}$.

The central topic in this paper concerns making the computation in equation 3 when the matrix ***U'U*** is "large," generally meaning ***D*** has 3500 or more rows and columns. Inversion of such a large matrix is a huge computational task which GLM cannot accomplish. Instead, GLM reverts to the "absorption," which does not actually invert ***U'U*** but rather "absorbs" the variation associated with the ***D*** portion of ***U***, which contains the columns associated with the transactions. In this example, there are 81,090 columns in ***D*** one in ***d***, and four in ***X***. Once the columns in ***D*** are absorbed the remaining portion of ***U'U*** only has five rows and five columns.

METHODOLOGY

The GLM procedure makes computations for *analysis of variance* and *regressions analysis*. These are two of the most important methods for statistical data analysis, and are under the broader grouping of *general linear models*, hence the term General Linear Model. GLM is one of the oldest procedures in SAS, dating back to the mid-seventies. HPMIXED is much newer and capable of making computations still in the realm of general linear models,

but utilizing computational methods unknown when GLM was developed. Three methods are illustrated; two using GLM and one using HPMIXED.

GLM WITH THE ABSORB STATEMENT

A statistical model for the data in terms of SAS proc GLM code is:

```
(1)   proc glm data = absorb;
            class product;
            model LogPrice = Product Period CostIndex Period*CostIndex
            DemandIndex Period*DemandIndex;

      run;
```

The variables names in the model statement are:

LogPrice = Logarithm of price in a transaction
Product is an identification of a particular part
Period = 0 or 1 depending on whether the transaction date was during the period of economic change
CostIndex is a combined measure of product costs
DemandIndex is a combined measure of product demand

Output 1 is a printout of the first five observations of the SAS data ABSORB:

Output 1: Observations in Data Set ABSORB

Obs	Product	LogPrice	Period	CostIndex	CostIndex_DM	DemandIndex	DemandIndex_DM
1	P00001	9.41222	1	77.0496	11.0348	-1.40432	-1.71539
2	P00001	8.71628	0	-1.9908	-2.3362	1.57576	-0.41131
3	P00001	8.99793	0	-1.9534	-2.2987	1.65133	-0.33573
4	P00001	9.43539	1	90.3732	24.3584	-0.11193	-0.42300
5	P00002	9.30849	1	43.6825	-22.3323	1.81546	1.50439

There being 81090 Products presents a computational difficulty. The CLASS statements would generate 81090 indicator variables and result in the matrix ***D*** having 81095 columns which would cause the GLM procedure to fail. To deal with this issue, GLM has the ABSORB statement that partially solves the problem by "absorbing" effects attributable to the variables in the ABSORB statement.

These statements would be used to invoke the absorption process:

```
(2)   proc glm data = absorb;
            absorb Product;
            model LogPrice = Period CostIndex Period*CostIndex
            DemandIndex Period*DemandIndex / solution;
      run;
```

The Absorb statement enables GLM to compute estimates of $\boldsymbol{\alpha}$ and the $\boldsymbol{\delta}$ vectors. But GLM will not compute parameter estimates for the indicator variables created by the ABSORB statement. Consequently, predicted values and residuals cannot be computed because the Product coefficients are not available.

Output 2 is a printout from submitting the statements in (2) showing parameter estimates for the estimated modes:

Output 2: Using the ABSORB Statement

The GLM Procedure

Dependent Variable: LogPrice

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	81095	587369.2142	7.2430	177.11	<.0001
Error	492552	20142.8397	0.0409		
Corrected Total	573647	607512.0539			

R-Square	Coeff Var	Root MSE	LogPrice Mean
0.966844	2.411284	0.202225	8.386600

Source	DF	Type I SS	Mean Square	F Value	Pr > F
Product	81090	577109.3803	7.1169	174.03	<.0001
Period	1	7226.4284	7226.4284	176708	<.0001
CostIndex	1	2949.5777	2949.5777	72125.9	<.0001
Period*CostIndex_DM	1	65.8201	65.8201	1609.50	<.0001
DemandIndex	1	4.1005	4.1005	100.27	<.0001
Period*DemandIndex_DM	1	13.9072	13.9072	340.07	<.0001

Source	DF	Type III SS	Mean Square	F Value	Pr > F
Period	1	26.54776049	26.54776049	649.17	<.0001
CostIndex	1	6.61124018	6.61124018	161.66	<.0001
Period*CostIndex_DM	1	32.25536070	32.25536070	788.74	<.0001
DemandIndex	1	17.39974480	17.39974480	425.48	<.0001
Period*DemandIndex_DM	1	13.90717669	13.90717669	340.07	<.0001

Parameter	Estimate	Standard Error	t Value	Pr > |t|
Period	0.1773744129	0.00696164	25.48	<.0001
CostIndex	0.0014294476	0.00011242	12.71	<.0001
Period*CostIndex_DM	0.0032004609	0.00011396	28.08	<.0001
DemandIndex	0.0159861182	0.00077501	20.63	<.0001
Period*DemandIndex_DM	-.0150221659	0.00081461	-18.44	<.0001

Notice that estimates of Product parameters in the vector $\boldsymbol{\alpha}$ are not printed due to the statement:

Absorb Product;

Those estimates usually are not of inherent interest. However, they would be needed to compute predicted values from the estimated regression model. In fact, predicted values and residuals are not available from proc GLM when using the ABSORB statement, as is clearly stated in SAS mentation of proc GLM. Closer inspection of the Type I SS output provides insight into the function of the statement. Look at the portion labelled "Type I SS" and notice the line labelled "Product." It shows DF = 81090, the number of columns in ***D***. This is the only line of the output pertaining to Products.

The value of the Type I SS = 577,109 is not computed in the same manner as the other SS shown in the output. SAS GLM documentation states it is computed in a manner similarly to those used by PROC NESTED; that is, standard analysis of variance computations. Specifically, let a_1 be the mean of the LogPrice values from Product 1 (see Output 1),

$$a_1 = (9.412 + 8.716 + 8.998 + 9.435)/4 = 9.140$$

and similarly for Product 2, 3, … , 80190. Then (uncorrected) Type I SS = $\Sigma_i (s_1^2 + \ldots + s_{811782}^2) = 577{,}109$, where $s_i^2 = n_i a_i^2$. You can think of $n_i a_i^2$ as being the SS associated with the "intercept" for Product i, which is technically correct, and thus the degrees of freedom for Product is the number of Products (10810).

GLM with Long-hand Absorption (LHA):

The purpose of this article is to present a method (referred to as "LHA" that basically duplicates what the ABSORB statement accomplishes, but retains information in a way that enables computation of predicted values and residuals. Predicted values require estimates of all the parameters in the model, namely those in the vectors ***α***, ***δ***, and ***β***. Estimates of parameters in ***α*** are not computed due to the ABSORB statement. But those estimates would be given by the Product means, $a_1, a_2, \ldots, a_{10810}$. The LHA process utilizes preliminary computations in data steps that essentially transform the indicators that are "absorbed," but does not require inversion of a matrix. Basically, in terms of values used in the previous section, it explicitly computes quantities of the form:

9.412 – 9.140 = **0.272**, 8.716 – 9.140 = **-0.424**, 8.998 – 9.140 = **-0.142**, 9.435 – 9.140 = **0.295**,

in a data step. These values are *residuals* that would result from regressing LogPrice on the Product indicator variables. The illustrated process pertains only to 0-1 indicator variables, such as those created for a variable in a CLASS statement.

Output 3 shows a data set with indicator variables for the first three products:

Output3: Data Set Absorb with Indicator Variables for Products P00001 P00002 and P00003

Obs	LogPrice	Product	D1	D2	D3	Period	CostIndex	DemandIndex
1	9.41222	P00001	1	0	0	1	77.0496	-1.40432
2	8.71628	P00001	1	0	0	0	-1.9908	1.57576
3	8.99793	P00001	1	0	0	0	-1.9534	1.65133
4	9.43539	P00001	1	0	0	1	90.3732	-0.11193
5	9.30849	P00002	0	1	0	1	43.6825	1.81546

The technique used to compute the parameter estimates is based on a well-known ***two-step*** process. Consider a statistical model with a dependent variable ***y*** and two sets of independent variables, denoted collectively in matrices *D* and *T,* with corresponding parameter vectors α and τ. The model equation is

$$\boldsymbol{y = D\alpha + T\tau + e}.$$

Relative to model 1. ($\boldsymbol{Y = D\alpha + d\delta + X\beta + e}$), $\boldsymbol{T\tau}$ in model 4. Is equal to $\boldsymbol{d\delta + X\beta}$ in model 1.

The first step is to regress the variable $\boldsymbol{y}$ and the variables in $\boldsymbol{T} = \{t_1, \ldots, t_l\}$ on the set of variables in $\boldsymbol{D} = \{d_1, \ldots, d_k\}$, and collect the residuals from those regressions. (This is the step that is not explicitly carried out by the ABSORB statement.) It turns out, since each of the columns in $\boldsymbol{D}$ consist of 0's and 1's, that ***these residuals can be computed directly*** without explicitly fitting a linear regression model. In fact, the values 0.272, -0.424, -0.142 and 0.295 in 5. are the residuals corresponding to Product P00001. In words, the coefficients for the regression on the dummy variables would be means of the variable being regressed (regressors), such as the number 0.272, for Product P00001. In turn, the residuals would be the values of the regressor minus the means of the values of the regressors corresponding to the Product group containing the value of the regressor, such as the numbers 9.412, 8.716, 8.998 and 9.453.

Assume the residuals from the regression have been obtained and entered into two sets of variables {*Res_y*} and {$Res\_t_1, \ldots, Res\_t_m$}. The second step is to regress *Res_y* on $Res\_t_1, \ldots, Res\_t_m$. The coefficients of this regression will be equal to the coefficient estimates for the variables in $\boldsymbol{T}$ as if the entire model (5) had been fitted, although no coefficients for the variables in $\boldsymbol{D}$ will be explicitly produced.

These SAS statements create variables needed to perform the computations in SAS data steps are:

```
data Absorb; set Absorb ;
      Per_CstInd=Period*CostIndex;
      Per_DemInd=Period*DemandIndex;
run;
```

The MEANS procedure is used to compute Product means of model variables and save the means in a data set named Prodmeans:

```
proc means data=Absorb noprint;
      by Product ;
      var LogPrice Period
      CostIndex Per_CstInd
      DemandIndex Per_DemInd;
      output out=Prodmeans mean=LogPr_Avg Period_Avg
      CstInd_Pavg Per_CstInd_Avg
      DemInd_Pavg Per_DemInd_Avg
      n=n;
run;
```

Output 4 shows a data set containing the means of all the quantitative variables in the model (note the suffix "Avg" on all these variable names):

Output 4: Product Means

Obs	Product	LogPr_Avg	Period_Avg	CstInd_Avg	Per_CstInd_dm_Avg	DemInd_Avg	Per_DemInd_dm_Avg
1	P00001	9.14045	0.50000	40.8697	8.84831	0.42771	-0.53460
2	P00002	9.25691	0.71429	52.9647	4.91834	0.90442	0.15567
3	P00003	9.18200	0.25000	14.2963	0.02548	0.87530	-0.68906
4	P00004	9.05492	0.57143	40.7255	3.62713	1.26628	0.26018
5	P00005	8.68597	0.66667	43.6838	0.71913	1.64128	0.43219

Next, use these statements to merge the data set Prodmeans with the original data set Absorb to obtain a new data set named Obsmeans:

```
data Obsmeans; merge Absorb Prodmeans; by Product;
run;
```

The first 5 observations of the data set Obsmeans are shown in Output 5:

Output 5: Observation with Product Means

Obs	Product	LogPr_Avg	Period_Avg	CstInd_Avg	Per_CstInd_dm_Avg	DemInd_Avg	Per_DemInd_dm_Avg
1	P00001	9.14045	0.50000	40.8697	8.84831	0.42771	-0.53460
2	P00001	9.14045	0.50000	40.8697	8.84831	0.42771	-0.53460
3	P00001	9.14045	0.50000	40.8697	8.84831	0.42771	-0.53460
4	P00001	9.14045	0.50000	40.8697	8.84831	0.42771	-0.53460
5	P00002	9.25691	0.71429	52.9647	4.91834	0.90442	0.15567

Now create a new data set named Resids whose observations contain values in Absorb minus the corresponding value in Obsmeans using this code:

```
data Resids; set Obsmeans;
      LogPr_Res=LogPrice-LogPr_Avg;
      PeriodRes=Period-Period_Avg;
      CstIndRes=CostIndex-CstInd_Avg;
      PeriodCstIndRes=Per_CstInd-Per_CstInd_Avg;
      DemIndRes=DemandIndex-DemInd_Avg;
      PeriodDmdIndRes=Per_DemInd-Per_DemInd_Avg;
run;
```

Output 6 contains the first five of the data set Resids:

Output 6: Data set Resids

Obs	Product	LogPr_Res	PeriodRes	CstIndRes	PeriodCstIndRes	DemIndRes	PeriodDmdIndRes
1	P00001	0.27176	0.50000	36.1799	2.1865	-1.83203	-1.18079
2	P00001	-0.42418	-0.50000	-42.8604	-8.8483	1.14805	0.53460
3	P00001	-0.14252	-0.50000	-42.8230	-8.8483	1.22362	0.53460
4	P00001	0.29494	0.50000	49.5035	15.5101	-0.53964	0.11159
5	P00002	0.05158	0.28571	-9.2822	-27.2506	0.91104	1.34872

Finally, regress LogPr_Res on the other residual values using the GLM statements

```
(3)       Proc GLM data=Resids;
            model LogPr_Res = PeriodRes CstIndRes PeriodCstIndRes
          DemIndRes PeriodDmdIndRes;
            output out=Pred p=LogPr_ResHat r=LogPr_ResRes;
          Run;
```

Output 7 shows results from statements (3):

Output 7: Results of the regression of LogPr_Res on RHS residuals

The GLM Procedure

Dependent Variable: LogPr_Res

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	5	10259.83390	2051.96678	58437.4	<.0001
Error	573642	20142.83972	0.03511		
Corrected Total	573647	30402.67362			

R-Square	Coeff Var	Root MSE	LogPr_Res Mean
0.337465	3.3577E16	0.187387	5.5808E-16

Source	DF	Type I SS	Mean Square	F Value	Pr > F
PeriodRes	1	7226.428400	7226.428400	205799	<.0001
CstIndRes	1	2949.577744	2949.577744	84000.2	<.0001
PeriodCstIndRes	1	65.820116	65.820116	1874.47	<.0001
DemIndRes	1	4.100460	4.100460	116.78	<.0001
PeriodDmdIndRes	1	13.907177	13.907177	396.06	<.0001

Source	DF	Type III SS	Mean Square	F Value	Pr > F
PeriodRes	1	26.54776049	26.54776049	756.05	<.0001
CstIndRes	1	6.61124018	6.61124018	188.28	<.0001
PeriodCstIndRes	1	32.25536071	32.25536071	918.59	<.0001
DemIndRes	1	17.39974480	17.39974480	495.52	<.0001
PeriodDmdIndRes	1	13.90717669	13.90717669	396.06	<.0001

Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	0.0000000000	0.00024741	0.00	1.0000
PeriodRes	0.1773744129	0.00645085	27.50	<.0001
CstIndRes	0.0014294476	0.00010418	13.72	<.0001
PeriodCstIndRes	0.0032004609	0.00010560	30.31	<.0001
DemIndRes	0.0159861182	0.00071814	22.26	<.0001
PeriodDmdIndRes	-.0150221659	0.00075484	-19.90	<.0001

Compare the output from statements (4) with that from statements (6), and see that parameter estimates are identical, with a minor caveat. Output from (4) shows no intercepts due to the use of the Absorb statement. As noted earlier, that is because the 81090 indicator variables are implicitly in the model. Each of those indicator variables corresponds to one and only one Product. The indicator equals 0 for all observations except those corresponding to the one Product for which the indicator has values of 1. (See output 3). Thus, if all the 81090

indicator variables were added together, ***the result would have the value 1 in each observation***, as would an "indicator" for an intercept. All you see in Output 2 is the single row labeled "Product" that tells you there are 81090 Products each with 1 DF. If you added all these columns related to Products, the sum would be a column with a 1 in all 81090 rows. So that's a long explanation of why no other "intercept" is in the model. All information in Output 2 below the row just discussed is identical to that found in Output 7. But Output 7 contains one row not found in Output 2), the row labeled "Intercept" whose value is zero out to ten decimal places. The correct value is exactly 0 because all the variables in the model are residuals, each of has mean 0.

Moreover, predicted values and residuals are available in the data set Pred. The GLM code in (3) above contains the statement in the GLM code (5) that creates the predicted values and residuals, namely

```
output out=Pred p=LogPr_ResHat r=LogPr_ResRes;
```

which produces predicted values and residuals and outputs them to a new data set named Pred. The data set named **Resids** contains the variables named LogPr_ResHat and LogPr_ResRes. These respectively contain predicted values and residuals from the regression. These variables are not computed when using the ABSORB statement, which reveals the major benefit of the methods presented in this paper. A prime use is to construct graphs of the type commonly presented to show results of the regression analysis. The predicted values and residuals are found in the variables LogPr_ResHat and LogPr_ResRes. These variable names derive from the name of the name of the dependent variable in the regression, LorPr_Res. "Res" in the variable name created from the computations leading up to Table 6.

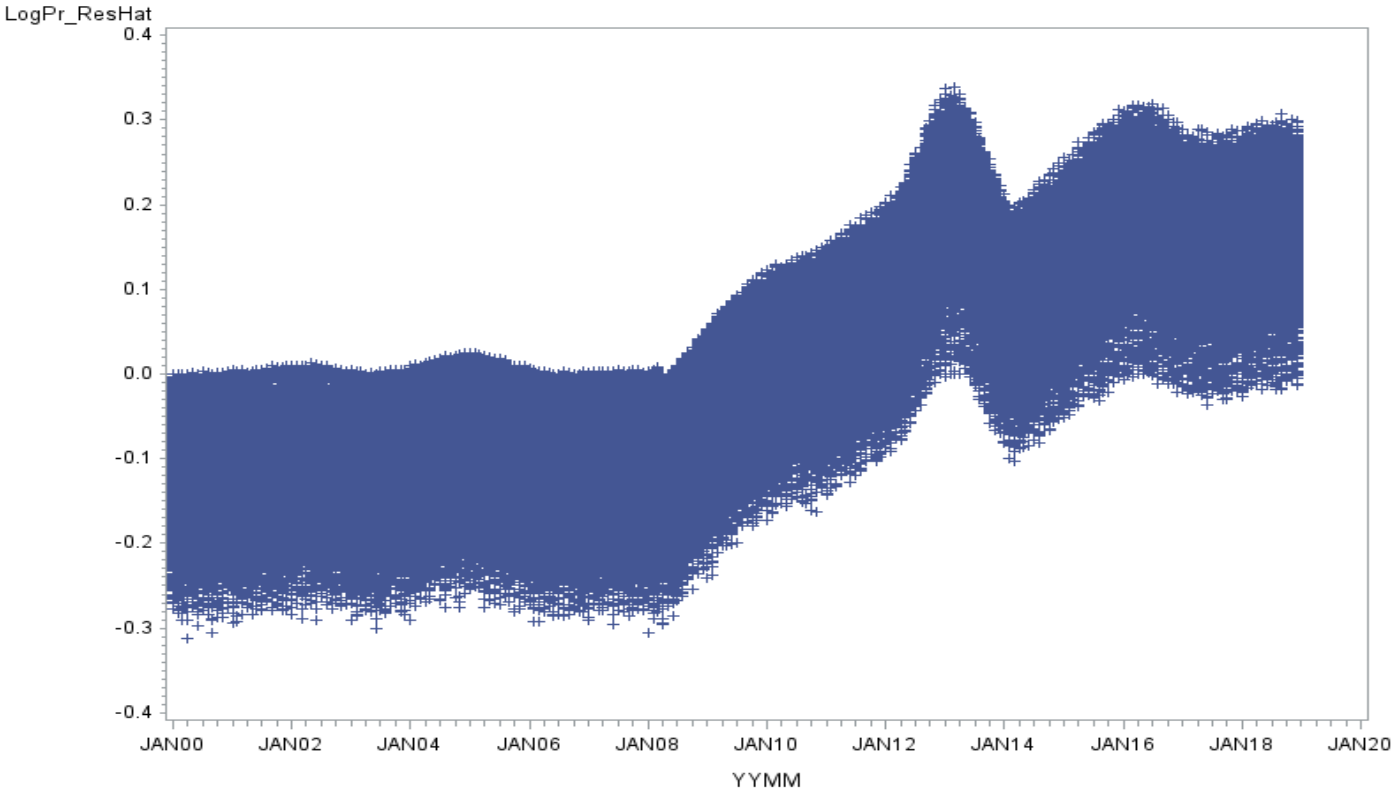
Output 8: Data set Pred

Obs	Product	CstIndRes	PeriodCstIndRes	DemIndRes	PeriodDmdIndRes	LogPr_ResHat	LogPr_ResRes
1	P00001	36.1799	2.1865	-1.83203	-1.18079	0.13585	0.13591
2	P00001	-42.8604	-8.8483	1.14805	0.53460	-0.16795	-0.25623
3	P00001	-42.8230	-8.8483	1.22362	0.53460	-0.16669	0.02417
4	P00001	49.5035	15.5101	-0.53964	0.11159	0.19879	0.09615
5	P00002	-9.2822	-27.2506	0.91104	1.34872	-0.05550	0.10708

Output 8 shows the list of variables in Output 6 that were used in the regression analysis displayed in Output7, and two variables that were produced by the regression analysis in Output 7, namely LogPr_ResHat and LogPr_ResRes. Please pardon the complicated notation for these variables. Just remember that LogPr_Res is the name of the dependent variable in the analysis from Output 7. Then LogPr_ResHat is the ***predicted*** value from that regression and LogPr_ResRes is the ***residual*** value from the regression.

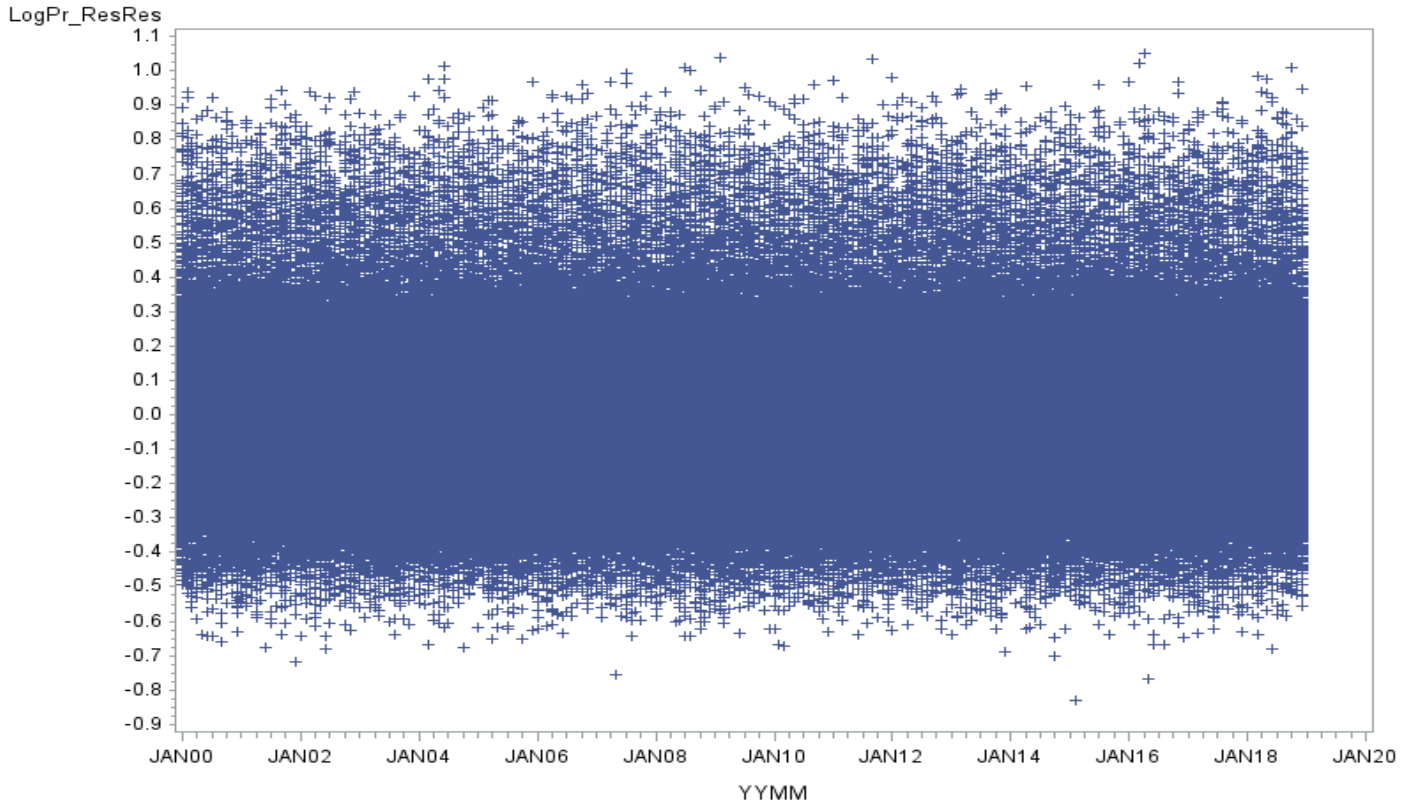
Predicted values and residuals are largely used to construct graphs and compute diagnostic measures of model fit and validity.

Output 9a: Plot of LogPrice Predicted Values vs Year and Month



Output 9a shows a plot of LogPr_ResHat versus the variable YYMM, whose values are of month and year. This plot reveals features of the predicted values over the years 2000 – 2018. Predicted values follow a trend that changes only modestly over years 2000 - 2007, increases steeply over 2008 - 2012 and then changes jaggedly over 2013 – 2018.

Output 9b: Plot of LogPrice Residuals vs Year and Month



Output 9b shows a plot of LogPr_ResRes versus YYMM. It reveals features of the variability in the residuals. A perfectly fitting model would produce residuals that (1) follow about the same range of vertical variation over the years and (2) also have a symmetric vertical pattern over the years. Output 9b shows the residuals meet criterion (1) but not quite as well criterion (2) because of being more densely packed along the lower edge of the plot than along the upper edge.

Methodology: HPMixed

The HPMixed procedure is much newer than GLM. It can be used to obtain all these results illustrated in the previous two sections; GLM with ABSORB statement and GLM with longhand absorption. HPMixed also has mixed model methods and more, but at the cost of much more cpu time. For example, the relatively small analysis shown here executed in 12 minutes by HPMixed, compared with only a few seconds by either of the GLM-based methods. The difference increases quickly as more levels of the absorbed variables increase. About 50% more levels results in cpu of more than an hour of cpu by HPMixed, but only a few more seconds by the method presented here. The difference is due to the necessity of essentially inverting an enormous matrix by HPMixed, but the complex computations using the present method increase only minutely due to only using simple arithmetic for the data step manipulations. In the end, the choice comes down to the nature of the problem at hand.

Here are statements for using HPMixed in the present problem:

```
Proc HPmixed Data= Absorb ;
    Class Product;
    model LogPrice = Product Period CostIndex
Period*CostIndex DemandIndex Period*DemandIndex ;
      estimate 'Period' Period 1;
```

```
        estimate 'CostIndex' CostIndex 1;
        estimate 'Period*CostIndex' Period*CostIndex 1;
        estimate 'DemandIndex' DemandIndex 1;
        estimate 'Period*DemandIndex' Period*DemandIndex 1;
Run;
```

Notice the use of “estimate” statements to obtain the desired parameter estimates.

Output 10 shows results from the HPMixed procedure.

Output 10: HPMixed

The HPMIXED Procedure

Model Information

Data Set	WORK.ABSORB
Response Variable	LogPrice
Estimation Method	Restricted Maximum Likelihood (REML)
Degrees of Freedom Method	Residual

Class Level Information

Class	Levels	Values
Product	81091	P00001 P00002 P00003 P00004 P00005 P00006 P00007 P00008 P00009 P00010 P00011 P00012 P00013 P00014 P00015 P00016 P00017 P00018 P00019 P00020 ...

Number of Observations Read	573648
Number of Observations Used	573648

Dimensions

G-side Cov. Parameters	0
R-side Cov. Parameters	1
Columns in X	81097
Columns in Z	0
Subjects (Blocks in V)	1

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	81095	587369	7.242977	177.11	<.0001
Error	492552	20143	0.040895		
Corrected Total	573647	607512			

Covariance Parameter Estimates

Cov Parm	Estimate
Residual	0.04089

Fit Statistics

-2 Res Log Likelihood	-24569
AIC (Smaller is Better)	-24567
AICC (Smaller is Better)	-24567
BIC (Smaller is Better)	-24556
CAIC (Smaller is Better)	-24555
HQIC (Smaller is Better)	-24564
R-Square	0.96684
Root MSE	0.20222
Coeff Var	2.41128

Estimates

Label	Estimate	Standard Error	DF	t Value	Pr > |t|
Period	0.1774	0.006962	493E3	25.48	<.0001
CostIndex	0.001429	0.000112	493E3	12.71	<.0001
Period*CostIndex_dm	0.003200	0.000114	493E3	28.08	<.0001
DemandIndex	0.01599	0.000775	493E3	20.63	<.0001
Period*DemandIndex_dm	-0.01502	0.000815	493E3	-18.44	<.0001

Results from HPMixed are very close to those of GLM with the ABSORB statement and GLM with the LHA computations. Given the reason for not using GLM with the ABSORB statement is the need to gain access to predicted values and values, the choice is between GLM with Long-hand Absorption and HPMixed. If the user is only about convenience of running the code, then HPMixed clearly is best. But if execution time is a concern and the user has time to invest the overhead in adapting the then is viable, particularly if the user wished to run the code frequently. Relative to the example illustrated in this paper, GLM with LHA used 0.53 seconds and HPMixed used 10 minutes and 39 seconds.

One other aspect of the choice between the options concerns there are two or more class variables. This is no problem with HPMixed, but the execution time may be greatly increased. GLM with LHA can be used, but the class variables would have to be combined into one variable. For example, it the two variables are A B, then and new variable could be constructed, say named AB, with its levels being the combinations of all levels of A with all levels of B. This is relatively easy to do, but it creates a large number of levels. The combined DF for the variable AB would be ab, where a is the number of levels of a and b is the number of levels of b. That is an increase of ab - a – b = (a -1)(b - 1) -1, which is the degrees of freedom for A*B interaction plus 1. If interaction is “significant” in whatever manner the user considers important, then it should be accommodated. If interaction is not significant, then *ideally* it should not be in the model. Having interaction in the model when it is not significant is not harmful except for wasting degrees of freedom. And a shortage of degrees of freedom is not a concern if the number of levels Is large.

On a final note regarding absorption: This topic is not treated extensively in statistics courses. But, at least in earlier times, it was of keen interest in fields like animal breeding. The interested reader might consult “Nellie Landbloom’s Copybook for Beginners in Research Work.”

Acknowledgements: Thanks to Joe Bloom and Paul Manning for preparation of the data set and helpful discussions about the manuscript.