Centering Variables

Let's see how to handle multicollinearity by centering a variable.

In the previous demonstration, we used PROC REG to look for multicollinearity—and we found it. These results clearly indicate that all three independent variables (**Amount**, **Amount^2**, and **Amount^3**) are involved in the multicollinearity.

To deal with the multicollinearity, we will use PROC GLMSELECT to create a new centered cubic polynomial effect in our model. In the OUTDESIGN= option, notice that we've added a C to the name of the output data set. The C refers to "centered." In the EFFECT statement, the polynomial effect is now named **QC_Amount**. As before, this effect is based on **Amount**, with a degree of 3. The STANDARDIZE option standardizes **Amount**, and the additional polynomial effects (**Amount^2** and **Amount^3**) are generated from the standardized value of **Amount**. By default, S_ is prefixed to the names of the standardized variables. =CENTER specifies that the variables be centered but not scaled. And METHOD=MOMENTS specifies that the center be estimated by the variable mean. In other words, the mean is subtracted from the variable values. Only observations that are used in the analysis are used for the standardization. The MODEL statement specifies the name of the centered polynomial, which represents the independent variables. SELECTION=NONE turns off variable selection.

Let's run this code.

```
proc glmselect data=mydata.paper outdesign=dc_paper;
   effect qc_amount=polynomial(amount /
          degree=3 standardize(method=moments)=center);
   model strength = qc_amount / selection=none;
   title "Paper Data Set: Centered Cubic Model";
run;
```

In the results, the Least Squares Summary table lists the effects. Notice that the three standardized variables in the polynomial effect have names that begin with S_. The Analysis of Variance table is exactly the same as the table for the model with the original uncentered variables (which was shown in a previous demonstration). This is because the centering changed only the location of the data values—not their variation.

At this point, we need to verify that we have successfully dealt with the multicollinearity in the model. We can use PROC REG to do this. The input data set is the data set created by the OUTDESIGN= option in the PROC GLMSELECT step that we just ran. The MODEL statement specifies the dependent variables and, for the independent variables, references the &_GLSMOD macro variable that PROC GLMSELECT created automatically. Again, the VIF, COLLIN, and COLLINOINT options are specified.

Above the PROC REG step, the ODS SELECT statement specifies the three output objects that we want to appear in the results: the Parameter Estimates table and the two Collinearity Diagnostics tables.

We submit the code.

```
ods select ParameterEstimates CollinDiag CollinDiagNoInt;
proc reg data=dc_paper;
   model strength = &_GLSMOD / vif collin collinoint;
   title 'Diagnostics for Centered Cubic Model';
run;
title;
quit;
```

In the Parameter Estimates table, notice how low the VIF values are for the centered variables. For the original variables, these values were huge. Now they are all below 10! In addition, the standard errors of the parameter estimates are also much smaller than the original model. These are all indications that this model is more stable. Also, all of the variables in the model are now significant at the 0.05 alpha level.

Let's look at the first Collinearity Diagnostics table. Is the Condition Index value in the last row below 100? Yes – it is below 10! This also shows that the multicollinearity has been dealt with.

Here are a few things to keep in mind about centering the data:

- If you score with a centered model, you need to center the scoring data on the same value that was used to center the training data.
- In regression models, you can validate the model only within the range of the data. Predictions based on independent variable values that are outside the range of the data (in other words, extrapolation) should be done with care because the relationship might be different outside of this range.

Let's quickly look at a few alternative methods of centering variables. Here, PROC STDIZE is used to center the original variable, **Amount**. The input data set is **paper**. The PROC STDIZE statement includes the METHOD= option, which is set to MEAN here (instead of MOMENTS, as in PROC GLMSELECT). The OUT= option specifies the temporary output data set **paper1**, in which the **Amount** variable is renamed to **MCAmount** (mean-centered Amount). The VAR statement specifies the variable to be standardized, which is **Amount**.
PROC STDIZE centers only the original **Amount** variable, so a DATA step is needed to create the squared and cubed terms based on the centered variable, and add them to the output data set. The new variables are called **MCAmount^2** and **MCAmount^3**, respectively. Finally, a PROC PRINT step prints the **paper1** data set.

We run the code.

```
proc stdize data=mydata.paper method=mean out=paper1(rename=(amount=mcamount));
   var amount;
run;

data paper1;
   set paper1;
   mcamount2 = mcamount**2;
   mcamount3 = mcamount**3;
run;

proc print data=paper1;
run;
```

The output shows the centered data.

Yet another method starts with a PROC SQL step. The SELECT statement creates a variable by calculating the mean of **Amount** from the **paper** data set, and puts the value into a macro variable named **MAmount**. The DATA step then creates the temporary **paper2** data set. First, the DATA step creates the centered version of **Amount**, called **MCAmount**, by subtracting the mean that is stored in the macro variable **MAmount**. The DATA step then creates the higher-order terms **MCAmount^2** and **MCAmount^3**, based on **MCAmount**, exactly as it did in the previous method. Again, a PROC PRINT step prints the centered data.

We run this code.

```
proc sql;
   select mean(amount) into: mamount
   from mydata.paper;
run;

data paper2;
   set mydata.paper;
   mcamount=amount-&mamount;
   mcamount2 = mcamount**2;
   mcamount3 = mcamount**3;
run;

proc print data=paper2;
run;
```

At the top of the results, we see the output from the PROC SQL step: the mean of **Amount**. Next is the centered data in the output data set.

Close