Building a Simple Polynomial Regression Model

In this demonstration, we perform an initial data exploration and then build a simple polynomial regression model.

To start, we use PROC SGPLOT to explore the relationship between the amount of additive and the strength of the paper. The input data set is **paper**. The SCATTER statement specifies the variables to be plotted: **Amount** on the X axis and **Strength** on the Y axis.

We submit the code.

```
title1 "Paper Data Set";

proc sgplot data=mydata.paper;
   scatter x=amount y=strength;
title2 "Scatter Plot";
run;
```

From the scatter plot, it appears that the strength increases as the amount of additive increases up to a certain value. Then the strength of the paper begins to decrease. This clearly indicates that a simple linear regression model might not be an appropriate choice and a polynomial regression might be necessary.

Next, we use a set of PROC SGPLOT steps to compare the linear relationship between the variables with polynomial relationships. Instead of the SCATTER statement, we use the REG statement to fit a line to each scatter plot. The first PROC SGPLOT step fits a regression line to display the linear model. The remaining three PROC SGPLOT steps fit smoothed second-, third-, and fourth-degree curves to the scatter plot, respectively. We could overlay the plots by putting multiple REG statements in one PROC SGPLOT step. However, the plots are easier to interpret if they are separate, so we use individual PROC SGPLOT steps instead. Let's look at the code.

In all of these PROC SGPLOT steps, the REG statement specifies **Amount** as the X variable and **Strength** as the Y variable. The following options also appear in the REG statement in all of the PROC SGPLOT steps:

- LEGENDLABEL= specifies a label that identifies the regression curve in the legend. In the first step, this label is *Linear*.
- The LINEATTRS= option uses suboptions to specify the appearance of the fit line. For example:
  - COLOR= specifies the color of the line, which is brown for the linear model plot.
  - PATTERN= specifies the pattern for the line, which is solid for the linear model plot.

In the remaining three PROC SGPLOT steps, the DEGREE= option is added to the REG statement to specify the degree of the polynomial to fit – 2, 3, and 4, respectively. Each PROC SGPLOT step also has a TITLE2 statement to help us distinguish the plots.

We submit the code.

```
proc sgplot data=mydata.paper;
   reg  x=amount y=strength / lineattrs =(color=brown
         pattern=solid) legendlabel="Linear";
title2 "Linear Model";
run;

proc sgplot data=mydata.paper;
   reg  x=amount y=strength / degree=2 lineattrs =(color=green
        pattern=mediumdash) legendlabel="2nd Degree";
title2 "Second Degree Polynomial";
run;

proc sgplot data=mydata.paper;
   reg  x=amount y=strength / degree=3 lineattrs =(color=red
        pattern=shortdash) legendlabel="3rd Degree";
title2 "Third Degree Polynomial";
run;

proc sgplot data=mydata.paper;
```

```
   reg  x=amount y=strength / degree=4 lineattrs =(color=blue
        pattern=longdash) legendlabel="4th Degree";
title2 "Fourth Degree Polynomial";
run;
title;

title2;
```

Let's look at the plots.

From the first plot, it is clear that a linear relationship does not capture the curvature present in the data. However, from the remaining graphs, it is unclear whether the relationship is best captured by a second-, third-, or fourth-degree polynomial curve. Instead of running a separate model for each polynomial, it is a good idea to start with a fourth-degree polynomial model and use sequential tests to refine the model.

This PROC GLMSELECT step fits a polynomial model to the data in paper. In the PROC GLMSELECT statement, the OUTDESIGN= option creates a data set called **d_paper**, which we will analyze later. The EFFECT statement creates a polynomial term named **P_Amount**. The DEGREE=4 option fits a fourth-degree polynomial for the variable **Amount**. The polynomial term encompasses the original variable and all of its degrees—in this case, **Amount** through **Amount**^{4}. The MODEL statement specifies the response (**Strength**) and the predictor variable (**P_Amount**). The SELECTION=NONE option specifies that the model will be fit without selection. We submit the code.

```
proc glmselect data=mydata.paper outdesign=d_paper;
          effect p_amount=polynomial(amount / degree=4);
          model strength = p_amount / selection=none;
          title "Paper Data Set: 4th Degree Polynomial";
run;
```

Now we look at the results.

In the Analysis of Variance table, the *p*-value for the overall model is very small. What does this tell you? It indicates that the model fits the data better than the baseline model. The R square value is 0.7429, which means that about 74% of the variation in **Strength** is explained by the model. The Parameter Estimates table provides the estimates for the intercept and slopes for **Amount**, **Amount**^{2}, **Amount**^{3}, and **Amount**^{4}, as well as the associated standard errors and *p*-values. However, based on these values, it seems that none of the slopes (that is, for **Amount**, **Amount**^{2}, **Amount**^{3}, and **Amount**^{4}) is significantly different from zero at an alpha level of 0.05. You should be cautious when you interpret the tests of hypothesis for the parameter estimates. They test the significance of each variable given that all of the other independent variables are already in the model. Therefore, if the independent variables in the model are correlated with one another, the significance of both variables can be hidden in these tests. In polynomial regression, higher-order terms are often correlated with the corresponding independent variable, a situation referred to as multicollinearity. In this example, multicollinearity among **Amount**, **Amount**^{2}, **Amount**^{3}, and **Amount**^{4} is likely the cause of nonsignificant *p*-values for all the slopes.

In this situation, it is clear that we must use a model selection method to reduce the number of terms in the model. When we reduce the model by removing nonsignificant terms one at a time, it is usually desirable to construct hierarchically well-formulated models.

A model that contains a variable to a power is hierarchically well-formulated if it also includes all lower powers of the variable. For example, consider the model $Y = \beta_0 + \beta_1X_1 + \beta_2X_1^2 + \beta_3Q + \varepsilon$. Is this model hierarchically well-formulated? Yes, this model is hierarchically well-formulated because it contains the term X_1^2 and the lower power of the variable, X_1.

By the same logic, if an interaction term is included in the model, then a hierarchically well-formulated model also includes the individual components of the interaction. Consider the model $Y = \beta_0 + \beta_1X_1 + \beta_2X_2 + \beta_3X_1X_2 + \varepsilon$. The term $\beta_3X_1X_2$ is a cross-product term. As is, this model is hierarchically well-formulated. However, if we remove either of the individual terms X_1 or X_2 , the model will no longer be hierarchically well-formulated.

Of course, if you can justifiably conclude that the true population model does not include a lower-order term, then the best model is not hierarchically well-formulated.

This PROC GLMSELECT step creates the reduced model. The PROC GLMSELECT and EFFECT statements are exactly the same as the previous PROC GLMSELECT step. However, now the MODEL statement has the following options:

- SELECTION=BACKWARD specifies the backward elimination method for automatic model selection.
- The SELECT= option specifies the criterion that PROC GLMSELECT uses to determine the order in which effects enter or leave (or do both) at each step of the specified selection method. SL specifies that effects enter and leave the model based on the significance level, which is the traditional approach.
- The SLSTAY= option specifies the significance level that must be achieved for a variable to remain in the model. In this example, the specified value is 0.05.
- The HIERARCHY= option specifies whether and how the model hierarchy requirement is applied. When HIERARCHY=SINGLE is specified, as shown here, only single terms can enter or leave the model at one time, following the principle of model hierarchy. The default is HIERARCHY=NONE, which specifies that the model hierarchy not be maintained. This means that any single effect can enter or leave the model at any given step of the selection process.
- The SHOWPVALUES option requests that p-values be displayed in the ANOVA and Parameter Estimates tables.

We submit the code.

```
proc glmselect data=mydata.paper outdesign=d_paper;
          effect p_amount=polynomial(amount / degree=4);
          model strength = p_amount / selection=backward select=sl slstay=0.05
         hierarchy=single showpvalues;
          title "Paper Data Set: Model Selection";
run;
```

The results show that the backward elimination method drops the **Amount^4** variable because its *p*-value (0.3682) is greater than 0.05. The Stop Details table indicates that no additional effect is removed because the *p*-value of the candidate for removal (that is, **Amount^3**) is less than the specified significance level of 0.05. The selected model is the cubic model, which is highly significant, as we can see from the *p*-value in the ANOVA table. Also the Parameter Estimates table shows that **Amount^3** is now significant. Although **Amount** and **Amount^2** are statistically insignificant, they are not removed, so that the model is hierarchically well-formulated.

We can write the following regression equation based on the parameter estimates: **Strength** = 2.73280 - 0.36900 * **Amount** + 0.22339 * **Amount^2** - 0.02862 * **Amount^3**.

Now that the cubic model is in place, we will use PROC REG to request various plots and tables, so that we can check the model assumptions.

The PROC REG statement specifies the data set **d_paper**, which was created by the OUTDESIGN= option in PROC GLMSELECT.

The PLOTS= option enables us to control the plots produced through ODS Graphics instead of simply getting the default plots. Global plot options apply to all plots generated by PROC REG. When used as a global plot option, UNPACK suppresses paneling of all plots.

The DIAGNOSTICS option is a plot request that produces a summary panel of fit diagnostics. The suboption STATS=NONE suppresses statistics that are included on a diagnostic panel.

The MODEL statement specifies the model with **Strength** as the response variable. The predictor variables are referenced by &_GLSMOD, a macro variable that PROC GLMSELECT creates automatically. This macro variable contains all of the predictor effects (including those created by the EFFECT statement) that were generated in the most recent run of the procedure. When automatic selection is used, then this macro variable contains only the effects in the final model. As we saw in the PROC GLMSELECT results, our final model has three predictors: **Amount**, **Amount^2**, and **Amount^3**. The LACKFIT option in the MODEL statement performs a lack-of-fit test, which compares the variation around the model with “pure” variation within replicated observations. The lack-of-fit test measures the adequacy of the specified model and can be specified only if some observations in your design are replicated. For additional details about the lack-of-fit test, click the Information button.

Notice that the label Cubic_Model is specified before the MODEL keyword. Let's run this code.

```
proc reg data=d_paper plots (unpack) =(diagnostics (stats=none));
   Cubic_Model: model strength=&_GLSMOD / lackfit;
   title "Paper Data Set: 3rd Degree Polynomial Model";
run;
quit;
```

In the results, the Analysis of Variance table now contains additional rows for the lack-of-fit test. This test is not significant: the *p*-value is 0.3682 at an alpha level of 0.05. So, there is not enough evidence to conclude that the specified model is inadequate.

Now let's look at the plots produced by the PLOTS=DIAGNOSTIC option in the PROC REG statement. The plot of the residual versus the predicted values appears to be a random scatter about the zero reference line. This is what we expected. The histogram might indicate a problem with the normality assumption, but the normal quantile plot appears less problematic.

The plot of the observed values by the predicted values is also created as part of the Diagnostics panel of plots and does not indicate any serious lack of fit. Although the data set consists of 22 points, several points have the same values for both **Amount** and **Strength**. Therefore, predicted values for only the 15 unique combinations are shown. The remaining plots in the output will be discussed in a later lesson.

Close