meat0.sas: Explanation of code

Goals of code:

- Reading data included with the SAS code
- Fitting a regression line
- Predicting mean Y at a specified X

This week's material is an introduction to linear regression. We'll see a lot more next week.

Reading data saved with the SAS code: `datalines;`
The data can be stored in the data step. This can be very handy when you want to read a small data set. I discourage it when the data set is large (It is no fun to help debug a file of SAS commands that includes 200+ lines of data).

To save the data with the SAS program, write a data step to read that data. Omit the infile statement. **After** the input statement and any data manipulation commands, includes a `datalines;` statement followed by lines of data. The `cards;` command is equivalent to `datalines;`. (who remembers punched cards any more?) Notice that `datalines;` ends with a semi-colon. Subsequent lines are considered to be data lines **until the next ;**. My practice is to put a semicolon on a line by itself, so the end of the data is clearly marked. The run; tells SAS to execute the code and create the data set.

In this problem, the desired regression equation uses X = log(time), so the variable logtime is created with the log transformed time. Remember that the definition of the logtime variable is placed after the input statement but before the datalines; statement.

Fitting a regression line: `proc glm; model ph = logtime;`
Many SAS procs will fit a regression line. We will consider two, `glm` and `reg`. Each provides slightly different functionality. proc glm makes it easy to print out estimates of the predicted mean for any X value.

To fit a regression with either proc glm or proc reg, the desired regression model goes on the model statement. As with fitting an ANOVA model, the response variable (Y) goes on the left of the = and the predictor variable (X) goes on the right. You do not put X in a class statement. If you include `class ph`, each unique ph value is used to define a group and proc glm will fit a model with a separate mean for each ph. Here, we want to fit a regression line. No class statement.

The output from proc reg includes a block of results named: Parameter Estimates The proc glm provides the same information, but the block of output isn't labeled. In both procs, you see columns labeled Parameter (or Variable), Estimate (or Parameter Estimate) and other columns. The Estimate or Parameter Estimate values are the fitted regression coefficients. The intercept is

β_0. The logtime row is the regression slope, β_1. Slope coefficients are labelled by the name of the X variable, which simplifies understanding the output when the model has more than one X variable.

The values in the table are the estimate, its standard error, the T statistic testing H0: parameter = 0, and the p-value for that test. The proc reg output also includes a column labeled DF, but that will have values of 1 unless you're fitting some complicated models. The test of the intercept = 0 is usually not very interesting (ph 0 is seriously bad), but the slope test is almost always very interesting.

Estimating mean Y at a specified X: `estimate 'label' intercept 1 logtime 1.6094;`
The predicted pH at 5 hours is the predicted value from the regression line when logtime = log(5) = 1.6094. That can be calculated using an estimate statement. That prediction is:

$$\begin{aligned}\widehat{\text{pH}} &= \hat{\beta}_0 + 1.6094\hat{\beta}_1 \\ &= (1)\hat{\beta}_0 + (1.6094)\hat{\beta}_1\end{aligned}$$

The estimate statement estimates this. We need the intercept multiplied by 1 plus the logtime slope multiplied by 1.6094. That is written as: `intercept 1 logtime 1.6094`. As with earlier uses of estimate, the estimate statement has a label in quotes followed by the description of the quantity to estimate.

This is an example of a linear combination of parameters that is not a linear contrast. The coefficients don't sum to 0. Linear combinations are acceptable when working with linear regression parameters. In general, you need linear contrasts when working with ANOVA models.

Setup to calculate regression predictions at many X values: `data meat2;`
The estimate statement prints predicted values for specified X values. Each new X requires a separate estimate statement. If you want predicted values for more than a few observations, it is easier to add the desired X's to the data set.

The key idea is that if you provide an observation with an X value and a missing Y value, that observation won't be used to estimate parameters (because Y is missing) but you can calculate predicted values and related information (because X is specified). You just can't calculate a residual (again because Y is missing).

The `data meat2;` step shows one way to specify X values for predictions. This data step (with the datalines statement) reads a set of new X values at which you want to make predictions. The code to indicate a missing numeric value in SAS is . (not in quotes). The string '.' in quotes is the character value .. The lonely . without quotes is the numeric missing value. This can be used in any data set. In particular, if a value for one variable is missing, use a lonely . to indicate that.

The second piece of setup is to create a data set (to be called all) with both the real data (meat) and the prediction points (meat2). We have used the set command before to read observations from a pre-existing SAS data set. When there are multiple data set names, set will read all observations from the first data set then all from the second. If you have more than 2 data set names on the set statement, SAS will read the first, then the second, and then the third, and fourth, ... for all data sets. So, set meat meat2; will concatenate the meat and meat2 data sets. We use that data set to fit the regression.

Saving predicted values in a new data set: `output out=preds predicted = yhat;`
This is a reminder because we used output statements to save predicted values and residuals. This week, we only want predicted values.

Printing a subset of the data: `where`
If we want to print out the predicted values for the prediction points (i.e., those with missing values of Y), we don't have to print the entire data set. We just tell SAS to use only those observations where Y is the missing value code. This is done with a `where` statement. where is followed by a logical expression, e.g., `ph = .`. The analysis will use only the rows of data where this is true. `where ph = .;` will print only those observations that are missing the Y value. These are the prediction points that were specified in the meat2 data set. SAS provides lots of logical expressions. `where time <= 6;` will print only the data for times 1, 2, 4, and 6. Not 8. Where statements can be used in any SAS proc.

Overlaying observations and predicted values: `proc sgplot;`
We have previously used proc sgplot; scatter ; to plot data. You can provide multiple plot commands which will overlay all the requested pieces. The series command draws lines (by connecting the observations). Since we want the lines to connect adjacent time values, I sort the data set by time first (proc sort; by time).