Brain.sas: Explanation of code

Goals of code:

- Producing a scatterplot matrix
- Fitting a multiple regression using proc glm
- Fitting a multiple regression using proc reg
- Regression diagnostics using proc reg

The preparatory code to read the data file and compute log transformed variables should be clear. Ask if you have any questions.

Scatterplot matrix: `proc sgscatter; matrix ;`
We have used proc sgplot before to draw bivariate plots (Y vs X). sgplot can draw many different types of plot and other sg (statistical graphics) procedures draw more. `proc sgscatter; matrix ;` draws a scatterplot matrix of the variables specified after the keyword matrix and before the terminating ;

You can add any of the usual modifying commands (by, where) or annotation commands (title) to further control what gets plotted.

Correlation coefficients: `proc corr;`
Proc corr calculates correlation coefficients between all pairs of the variables listed on the var statement. You can list variables individually: `var gest brain body litter;` or you can specify a group of variables.

There are two ways to specify a group of variables:
When variable names have the same stem, e.g. wt1, wt2, wt3, wt4, you can write `var wt1-wt4;`
When you want "adjacent" variables, you can write `gest -- litter;`. That is two hyphens side by side. SAS stores variables in the order it encounters them. This is the order of columns in the proc print output. You can specify the first and last variable separated by two hyphens and SAS will treat that as the first, the last and all variables "in between". Either specification can be used anyplace you want a list of variables (e.g., in the sgscatter matrix statement, or on the right hand side of a glm model statement.

multiple regression using proc glm `proc glm; model ;`
proc glm allows you to specify multiple X variables on the right hand side of the model statement. This will fit a multiple regression.

Lecture will discuss sequential (Type I) and partial (Type III) tests.

The numeric output includes:

- The ANOVA table comparing the full model (all X variables) to the intercept only model.
- Four numbers describing the full model: The most useful (for me) is Root MSE, which is the estimate of teh residual standard deviation, i.e., the square root of the MSE from the ANOVA table.
- Type I (sequential) sums-of-squares and associated tests, for adding each term in the model
- Type III (partial) sums-of-squares and associated tests, for adding each term last to the model.
- Parameter estimates, their standard errors, and tests of each parameter = 0.

This may be followed by a plot of the model predictions. What sort of plot depends on the model being fit. This is a surface plot if the model has 2 X variables. No plot is produced when there are 3 or more X variables.

Confidence intervals for parameters: `model /clparm;`
To include confidence intervals on the regression coefficients (intercept and slopes), add `/clparm` to the end of the model statement. That adds 95% confidence intervals to the information about each estimate. If you want something other than 95%, add `alpha=0.10` (for 90% intervals) to the list of options, e.g. `model ... /clparm alpha = 0.10;`. You can specify any desired coverage by alpha = 1-coverage.

multiple regression using proc reg `proc reg; model ;`
proc glm can fit models with continuous X variables or variables that indicate groups (i.e. in a class statement). If you don't have any grouping variables (no class statement needed), you can fit the model using proc reg . The results are identical, but the output is organized differently.

When we look at additional regression output, e.g. diagnostics, proc reg and proc glm provide overlapping but not identical options.

The proc reg syntax is identical to the proc glm syntax. Variable names for the model to be fit go on the right hand side of the equals sign in the model statement.

The numeric output includes:

- The overall ANOVA table, identical to that from proc glm
- A five number summary of the full model. Four numbers are the same as those in the proc glm output. Again, Root MSE is the number I find most useful.
- Parameter estimates, their standard errors, and tests of parameter = 0.

This is followed by a page of graphical diagnostics. The first one is the residual vs predicted values plot. We will talk about the most of the other diagnostics in the next week or so.

Regression diagnostics: `\r vif` and `output`
If you want to see the actual numbers for regression diagnostics, adding the `/r` option to the proc reg prints out diagnostics for each observation.

The VIF (variance inflation factor) diagnostic is characteristic of each variable (not each observation). To print that out, add `/vif` to the proc reg statement.

Note: If you have two options, e.g. `/r` and `/vif`, you only need one `/` to separate the model piece from the options. This is illustrated in the example code in the last proc reg.

If you want to store the diagnostics in a data set, which can then be printed or plotted, use `output` with the appropriate keyword. The example code gives keywords for the diagnostics we'll use in class and a few more that I'll briefly mention.

Many diagnostics, e.g., Cook's Distance, are best plotted against the observation number. This gives you a quick indication of the largest value (i.e., the answer to the question, do I need to care about influential observations?) and help you identify which point is the one (or more) with large D values.

Storing the observation number in a data set: `i = _n_;`
The easiest way to produce a full page plot using the observation number is to create a new variable (I use i) containing the observation number. This is done in the data step that creates brain2. Inside a data step, SAS creates various "internal" variables. These have values that are defined inside the data step but are not saved, so they are not accessible to procs. When SAS creates a variable, it usually has `_` before and after, so the created variable is not confused with a variable name you want. One of those "internal" variables is `_N_`. This contains the observation number. To save that information, copy it to a variable you name (e.g. `i`), which will then be stored and can used in subsequent procs.