sat.sas: Explanation of code

Goals of code:

- Reading an excel worksheet
- Model selection statistics
- Storing model selection results in a data set
- Manipulating model selection statistics
- Calculating PRESS and rMSEP

Reading an excel worksheet: `proc import`;

Proc Import can read .xslx or .xls files. Just provide the appropriate filename and give a name for the resulting SAS data set. Remember `replace` allows SAS to overwrite an old copy of the data set. If you only have one sheet in the workbook, that is what is read. Doesn't matter whether that sheet has the default name of Sheet1 or some other name.

When there are two or more sheets, you have to tell proc import which sheet you want to read. That is done by `sheet=' name ';`. This is a separate command, unlike `out=` or `replace`. `sheet=' name ';` goes after the `;` at the end of `proc import` and before the `run;`. Other commands, such as `guessingrows=` also go between the proc statement and run statements. The SAT file has one sheet, called sat, so `sheet='sat';` will read that sheet by name. It is commented out because it isn't necessary here.

All the rest of this week's information concerns `proc reg`. Although both proc glm and proc reg fit regression models, proc reg provides the better set of tools for multiple regression. The only time I revert to proc glm for multiple regression is when I want SAS to create indicator variables automatically. proc glmselect provides model selection for models usually fit with proc glm; glmselect doesn't provide all subsets selection (at least the last time I checked).

Controlling output using ODS

SAS provides a very powerful set of commands to manipulate output. This is ODS, the output delivery system. You can copy "printed" output to a word document, save a bit of printed output as an excel file or as a SAS data set, or other related tasks. All we want to do right now is not print out any of the results (there are lots) from model selection in proc reg. The results window is called the html destination; `ods html <stuff>` controls that destination.

Model selection produces a lot of printed output, which we don't need to see. To exclude any output, turn it off by `ods html exclude all;`. Any output from any proc is excluded until output is restored, by `ods html select all`.

Two other helpful uses of ODS:

- `ods html close; ods html;` The `html` destination describes what goes in the Results Viewer window. This pair of commands closes the output and restarts it. The result is that the previous contents of the Results Viewer window are erased.
- `ods rtf file='name.rtf';` This starts the `rtf` destination in addition to the `html` destination. The result is that all results go to both the Results Viewer window and to the file you specify. The `ods rtf close;` closes that destination and saves the file. The result is a file that Word can read.

Model selection statistics: `proc reg; model / selection = cp`

Proc reg does all subsets variable selection by adding the option `/ selection = cp` to the model statement. The variable list (right-hand side of the model equation) lists the variables to be considered. Variables not on this list are ignored. By default, you will get information about every possible model.

My preferred approach is to turn off printed output, fit all models, save the results into a data set, then extract what I want from that data set.

`proc reg;` fits regression models. Very similar to proc glm; except that class statements (to automatically generate indicator variables) aren't allowed in proc reg, and proc glm doesn't do any model selection. We want model selection, so we're using proc reg now.

`outest =` we want to save information about all the subset models in a data set. `outest` names that data set. The default information includes the parameter estimates, r^2 statistic and the root-mean-square-error (`_RMSE_`), i.e. the error sd.

The AIC and SBC options request that the AIC and BIC statistic are printed (or stored) for each model. There are a couple of versions of the BIC criterion. SBC is SAS's name what I (and most others) call BIC. When results are saved in a data set, `_AIC_` and `_SBC_` are the variable names containing the AIC and BIC statistics.

Once the numbers are saved in a data set, you can sort observations by the desired statistic (proc sort). If you then print the first few observations, using `proc print data=models(obs=5);`, you get the best models by the criterion used in the sort. You have to explicitly name the data set and can choose whatever number of observations you want to see. The code illustrates sorting and printing by AIC and sorting by BIC.

Calculating the change from the best

The SAS data step is a very powerful data manipulation language. proc reg saves the AIC and BIC statistics, but it doesn't compute delta, the change from the best value of that statistic. We use a data step to compute that. The concept is to start with a data set sorted by the criterion you want (`_AIC` or `_SBC_`). After sorting by either, the first row corresponds to the best model (smallest value). So we save the value for the first observation, then use that to calculate the change from the best.

If you want to compute delta AIC, the code does not need any modification. If you want to compute delta BIC, change the `%let` line to `%let criterion = _SBC_;`. The data step computes delta, then the following proc print print all observations that satisfy the "where" criterion. Here, that's delta < 2.

Detailed explanation of the code, using macro variables and data step retain statements:

The SAS macro language provides a way to define text that will be used multiple times and potentially in multiple data or proc steps. Macro variables start with `&` and macro commands start with `%`. My code allows you to define delta in terms of AIC (the `_AIC_` variable) or BIC (the `_SBC_` variable). You specify which you want by defining the macro variable `&criterion` as either `_AIC_` or `_SBC_`. Then any time `&criterion` occurs in the SAS code, it is replaced by the contents of the criterion macro variable.

A data step calculates delta. The first thing we need to do is save the criterion for the first row of data (`if _n_ = 1 then best = &criterion;`) `_n_` is a "special" SAS variable containing the observation number, so `_n_ = 1` is true for the first observation, and the if statement saves the value of the criterion in the variable called best. By default SAS resets the values of all variables to missing values when it reads a new observation. The second thing we need is to override that so the value of best is retained across the observations (`retain best;`). Then all we need to do is calculate the difference between the criterion value for any observation in all models (i.e., for a specific model) and the best value (`delta = &criterion - best;`). Then we can use the value of delta to print only the desired observations (`where delta <= 2;`).

Calculating PRESS and rMSEP: `proc reg data=sat outest=press0 press;`

PRESS is the Predicted REsidual Sum of Squares statistic. This quantifies how well the model predicts new observations, by removing each observation one at a time, predicting its value, and calculating the sum of squared differences. It is an out-of-sample version of the error SS.

In SAS, you request it by adding `press` and specifying an output data set with `outest= <data set name>`. The Press statistic is added to the data set as the variable called `_PRESS_` along with estimates and model fit information. Once `_PRESS_` is stored, you can use a data step to do further calculations.

SAS does not automatically calculate the root mean square prediction error, rMSEP, which I find a lot more useful because it doesn't depend on the number of observations and is on the same scale as the response (not a squared scale like SSE or PRESS). rMSEP is calculated from PRESS as $\sqrt{PRESS/n}$ where n is the number of observations. SAS doesn't save the number of observations but it does save the total number of coefficients, as `_P_`, and the error degrees of freedom, as `_EDF_`. The number of observations is the sum of these.

The data step that creates the press1 data set implements these calculations. This is followed by a proc print to see the values of PRESS and the rMSEP.