donner1.sas: Explanation of code

Goals of code:

- Fitting a logistic regression
- Generate the "automatic plot" of predicted values
- Store output and draw your own plot

Two SAS procs will fit logistic regressions: proc logistic and proc genmod. Both fit the logistic regressions we need; they provide slightly different sets of advanced methods. My code demonstrates proc logistic.

The example is the Donner party survival data used as the class example.
Note: You will probably need to change the path to the data file to where you stored the file on your computer. If you stored it in the folder that is your SAS default folder, you only need to give the file name.

Syntax for a no-options logistic regression:

`proc logistic` #1: This fits a logistic regression and gives you various numerical output.

`model survival(event='1') = age;`
Very similar syntax to a proc glm (or proc reg) model statement. The response variable goes on the left of the `=`; the predictor variables go on the right. So this statement models the probability of survival as a function of age.
Note: If you reverse the variables, e.g. `model age = survival`, you are trying to fit a t-test (comparing the age mean for survival = 0 to that for survival = 1). Proc logistic doesn't do that.

`(event = '1')`
Are you modeling P[death] or P[survived], in other words, are you modeling P[survival = 0] or P[survival = 1]? Proc logistic has to know which you want. By default, proc logistic models with probability of the smaller value of the response, i.e. survival = 0 when survival is 0 or 1. Whether that's what you want depends on how you coded survival; is 0 a death or a survivor? For the Donner data, 0 is a death. Specifying `event = '1'` in () after the response variable name tells SAS what you want to model the probability of a 1. Note that the value goes in quotes, even if it is a numeric value in the data set.

Output from a basic logistic regression: The first few blocks are descriptive information about the data. Things I check:
Number of observations read and number used:
Make sure they are what you expect
Frequency of each level of the response:

Logistic regression will not work well if the event is extremely rare or extremely common

Probability modeled: statement:

Tells you what is being modeled (see event= text above).

This is extemely important to check.

Interpretation of P[death] is opposite from that of P[survived].

The important results follow:

Model fit statistics: -2 Log L is the deviance (-2 log likelihood). We'll talk about AIC and SC (called BIC by most) later.

Testing Global Null Hypothesis: This tests whether **All** regression parameters are 0. This is a model comparison between the model you specified and the "intercept only" model. SAS computes 3. I talked about the likelihood ratio test and the one-parameter version of the Wald test. The Wald test here is a generalization to multiple parameters when there is more than 1.

Analysis of maximum likelihood estimates: Here's where you find the estimated coefficients, their se's and tests of parameter = 0. My lecture discussed the Z test of each parameter. SAS reports a Chi-square statistic that is the square of the Z statistic. Same p-value.

Odds ratio Estimates: Exponentiates the regression slopes (i.e., omitting the intercept) to give you odds ratios and their confidence intervals.

SAS also reports a block of measures that quantify classification accuracy. Details of their definition and interpretation are in the SAS documentation.

More statements for proc logistic:

`effectplot fit`:
This draws a plot of the observations, the predicted probability curve, and 95% confidence bands.

`estimate / ilink cl`:
Provides predicted probabilities and their uncertainty for specified X values. This is analogous to the use of the estimate statement in proc glm or proc reg. By default, you get the estimates on the logit probability (= intercept + slope * X) scale. The `ilink` option requests estimates of the probability. the `cl` option request confidence intervals (which SAS calls confidence limits, hence cl).

The output has one line for each estimate statement. The first set of numbers (estimate through upper) are for the logit probability. If you want to know about the probability, focus on results from Mean to Upper Mean. Mean is the probability, and lower Mean, Upper Mean are the 95% CI for the probability.

Printing predicted probabilities and more control over what is plotted

If you want to compute lots of probabilities, or specify exactly what you want shown in a plot, you'll need to follow the data + new data approach used for simple linear regressions. That is create a new data set with the X values for which you want predictions, concatenate that with the original data, and ask SAS to create an output data set with the information you request. As with the linear regression application, the response variable in the new data set is set to missing values so the new data doesn't change the fitted regression.

`output` followed by keyword = name pairs:
Requests an output data set from SAS. The `out=` is required. The rest are optional. You can use as many or as few as you need. You can request many different things; the SAS documentation has a complete list. Here are the ones I use most frequently:

`out = preds` names the output data set (here, it will be called preds). Required.
`xbeta = xb` stores the predicted logit in the specified variable name
`predicted = prob` stores the predicted probability
`lower = lowerci` stores the lower bound of the 95% ci
`upper = upperci` stores the upper bound of the 95% ci

`proc print; where age = 55`:
The where statement specifies a subset of observations to be used in that proc. Where can be used in any proc. Here, this prints only the observations where age = 55. These results match those from the estimate statement used earlier.

`proc sgplot; where survival = .`:
We only want to plot information from the new data set. These are the observations that are missing (value of "dot") a value for survival. Proc sgplot provides a huge number of possible plots. We've seen scatter to plot observations.

`series`: plots a line (connect the dots) for the specified X and Y values

When you provide multiple plot requests, these are overlaid on the same plot. So the first proc sgplot draws 3 lines: the predicted values and the 95% confidence interval.

`proc sort`
The lines in the previous plot look good because the age values are sorted in increasing order (at least in the survival = . subset). When the age values bounce back and forth, the lines do not look good. If this happens, you can fix this by sorting the data using proc sort; The by statement specifies what variable (or variables) to use to sort the data.

The second proc sgplot overlays the data and the predicted probabilities.