Corn.sas: Explanation of code

Goals of code:

- labeling data plots, using datalabels
- Fitting a polynomial regression using proc reg
- Fitting a polynomial regression using proc glm
- Creating groups from a continuous variable
- labeling data plots, using group=
- "Trellis" plot, using groups

labeling data points: `scatter / datalabels= ;`
We have used proc sgplot before to draw bivariate plots (Y vs X). If you add the / datalabels=variablename option, the points will be labeled with the values of the specified variable.

fitting a polynomial regression using proc reg:
Proc reg requires that all variables in the model be defined in the data set. The data corn creates the quadratic term, which I called year2. You can either multiply the year by itself, or use the power (**) operator to compute the square. The first computation creates year2; the second creates year2a. They have identical values.

To fit a quadratic regression in proc reg, you include year and year2 (or year2a) in the model statement.

fitting a polynomial regression using proc glm:
You could use a model statement with an already created variable in proc glm exactly as would in proc reg. Proc glm also allows you to create powers of variables "on the fly". That's what year*year specifies in the proc glm model statement. This will give an error if you try it in proc reg, but proc glm understands it and generates the quadratic term in the model without requiring you to calculate it first.

Creating groups: `data corn2;` including `if ; else;`
The corn data set only has year and yield. If we want to group points by their decade, we need to create a variable with the decade. This will be based on the value of year for each observation. The data corn2; data step creates the decade variable. It reads observations from the corn data set, then uses a sequence of if else statements to assign values to decade. If the year is less than 1900, then the first then clause (decade = '1900') is executed. If not, then the else clause is executed. In this case, that is another if statement. So on until the very end. If none of the if statements are true (i.e., the year is $\geq$ 1920, then the last else clause is executed.

Note that the else's are crucial. If you just had a chain of if's (if year ¡ 1900 ; if year ¡ 1910 ;, etc.), when the year value is 1895, the first if is true so decade is set to '1890'. The next statement is

executed and that if is true, so decade is set to '1900'. Not good. The else makes it so that when one if is true, none of the following ones are executed.

labeling data points: `scatter / group= ;`
Instead of labeling points by their year, you can tell SAS to indicate groups by the symbol color. This is done by an option to the `proc sgplot; scatter` command. The `/ group = variable name` option tells SAS to use groups defined by the specified variable to mark the points.

Trellis plot: `proc sgpanel;`
A "Trellis plot" plots Y vs X for subsets of the data. The example code plots the yield against rainfall separately for each decade. proc sgpanel requires three pieces of information to draw that plot: 1) the name of the variable that defines each subset. This goes in the `panelby` statement. The `/novarname` option suppresses the "decade=" information from the panel header. 2 & 3): the names of the X and Y variable, which go in the scatter statement.

You can ask for many other types of plots instead of a scatter plot in `proc sgpanel; panelby` . Just replace `scatter ;` by the plot you want (e.g., `series ;` to connect the dots).

Note: If you ask for `panelby year`, you will get one plot for each year; each plot will have one observation. Probably not a useful plot.