Debugging:

1) SAS will tell you where it found an error. The real problem is often above or to the left of the place where SAS found the error. It is especially likely to claim to have found an error in the wrong place if you have a missing semi-colon, quotation mark, comment close-mark or if you have misspelled an important word like "data" or "proc". It cannot be below the place where SAS reports the error.

2) One of SAS's left over habits from the main frame era is that it handles errors in a funny way. It does not determine whether your whole program can be run before it starts execution – so it can work for hours before it finds an error. (One solution to this is to tell it to execute the whole program with no observations.) Once it finds an error, it tries to continue to run but often keeps running without any data and generates spurious error messages.

Words to look for in log files that may indicate problems – from most serious and common to less serious:

"Error" indicates that the SAS system encountered something it could not handle and stopped running. SAS prints the word error a lot, along with a description of its problem, when it encounters an error.

"uninitialized" indicates that you requested a variable that is not in the data set. This often indicates an error in typing the variable name, that you have sent the wrong data set to the procedure, or that a data set used X first and defined it later. While the SAS program will which it will treat the unitialized variable as all missing and keep running, this is almost always a problem worth fixing.

"Warning" indicates something that is typically a problem, but that SAS can deal with.

"Missing values were generated" – indicates that a SAS function in a data step bumped into values it did not know how to handle. For example, you could have tried to generate a date with a missing month value. Figure out what is causing the missing data to be generated. Often you can correct the problem. Otherwise – even if the missing values are expected, I typically write write code that spots the problem values and explicitly sets the results to missing. It looks like:

```
if (var not bad) then X = function(var); else X = .a;
```

I would then also write a comment explaining the data problem this addresses.

"has 0 observations" or "no observations" – indicates an empty data set or a logical condition that selected no data. This is sometimes normal.

"Converted" indicates that you used a string when SAS expected a numeric value or vice-versa; this will often generate a great deal of missing data.

There are a couple error messages associated with inputting data: "invalid data", "lost card", and "reached past the end of the line".

Expect Problems with your own code and your data. Look for them.

Finding and solving problems early prevents having to run and interpret analysis repeatedly. It also prevents you from either getting bizarre results and spending hours puzzling out why or presenting incorrect results.

Thus the first step in your project should be to clean the data by looking for bad entries and figuring out whether to keep, fix, or delete them. Look for bad entries again or other signs of problems after you construct new variables or merge data sets.

-Think about what reasonable records should look like. Print out some data and look at it.

```
proc print data = weather (obs= 10);
```

Obs	WeatherStationWbanNumber	Date	weather_time	weather_hours	skyConditions	ClearSky	DryBulbTemperatureString	DryBulbTemperature	DewPointTemperatureString	DewPoint
1	13739	05/01/98	1:51:00	1	OVC110	0	61	61.0	55.9	55.9
2	13739	05/01/98	2:51:00	2	OVC120	0	60.1	60.1	55.9	55.9
3	13739	05/01/98	3:51:00	3	SCT035 OVC055	0	60.1	60.1	55	55.0
4	13739	05/01/98	4:51:00	4	FEW055 OVC110	0	59	59.0	55	55.0
5	13739	05/01/98	5:51:00	5	SCT060 BKN110	0	59	59.0	55	55.0
6	13739	05/01/98	6:51:00	6	BKN060 OVC070	0	59	59.0	55.9	55.9
7	13739	05/01/98	7:51:00	7	SCT060 BKN075 OVC120	0	60.1	60.1	57	57.0
8	13739	05/01/98	8:51:00	8	SCT080 BKN140 OVC180	0	62.1	62.1	57	57.0
9	13739	05/01/98	9:51:00	9	FEW080 BKN110 OVC180	0	63	63.0	57.9	57.9
10	13739	05/01/98	10:51:00	10	FEW080 SCT100 BKN120 OVC180	0	64	64.0	59	59.0

-Think about what reasonable values and distributions should look like (Consider some data about the population of Alameda County. The average age of the population should be 30-50; if the average is 85, you have a problem. Negative ages are a problem as are ages over 120.). Run

frequencies or univariates to check whether the distributions are reasonable. You probably want to use the /missing option on the frequencies to see missing values. If you start seeing strange entries, print them out. (is the 100 year old in the data set working? Getting children's health insurance? Riddled with other entries coded as 100 that suggest that 100 actually means missing?)

```
proc freq data=national_check;
/*checks Social Security Numbers -- are they long enough?  did we get the
ones we asked for when we told the state to send us a data set randomly
sampled by SSN?*/
        tables ssn_3_digit ssn_length /missing;
```

-When you merge together two data sets that should match perfectly – say observations of temperature and electricity demand for each hour of each day or matching each medical claim to the person who made it – make sure that every entry did in fact match.

```
data merged_load_and_price
        trouble;  /*I've named two output data sets on the data statement;
this is Kosher and a very powerful technique*/
        merge prices. raw_realtime_prices (in = in_price)
load.pjm_hourly_load (in=in_load);
        by date;
      /*the IN= varname statement creates a variable, varname, that is 1
(true) if this merged observation contains data from the specified dataset
and is 0 (false) otherwise*/



        if in_price and in_load then output merged_load_and_price;
              else output trouble;
/*now use the output statement to write observations to the datasets
specified above*/


Proc print data = trouble;  /*now see what we've found*/
```

Here, the following lines of the LOG file give us very good news – the merge worked the way that we expected and there will be no output in the LST file:

```
NOTE: The data set WORK.MERGED_LOAD_AND_PRICE has 43824 observations and 35
variables.
NOTE: The data set WORK.TROUBLE has 0 observations and 35 variables.
```