7.13 Troubleshooting

If all the functions which you were trying to maximize were globally concave and were computable over the entire parameter space, there would be little need for this section. Unfortunately, that's not the real world, and if you do any significant amount of work with non-linear optimization, you will run into occasional problems. Make sure that you understand the effects of numerical precision and initial guess values described in 7.1. Of course, almost all problems are due to "bad" initial guess values, in the sense that if you could start out at the optimum, everything would be simple. And many problems can be solved by coming up with better settings for these.

If you haven't tried using preliminary simplex iterations (section 7.3) before using BFGS or BHHH, we would suggest that you start with that. If you have, and you still are having problems, read further.

"Missing Values And/Or SMPL Options Leave No Usable Data Points"

RATS will produce this message if it can't find any data points at which it can compute an objective function. In some cases, there is a simple fix. For instance, if you are maximizing a function which includes a term of `log(sigsq)` and you haven't given `sigsq` an initial guess value, the standard initialization value of zero will leave the function uncomputable. If you did this, you should see a warning like

```
## NL6. NONLIN Parameter SIGSQ Has Not Been Initialized. Trying 0
```

These warnings usually are of little consequence, since zero is, for many parameters, a quite reasonable initial guess value. If, however, you're having missing value problems, be careful about this.

The other common source for this problem is a failure to set up a recursively defined function properly. If you have an ARCH/GARCH model or one with moving average errors, you need to give a hard value for the *start* parameter since the recursion has to have a known initial point. See the "Recursive FRML's" part of Section 7.6 if you need more help.

"Subiterations Limit Exceeded"

This message is issued when the subiteration process described in Section 7.2 fails to produce a step size which meets the criteria described there. This indicates a very serious problem—the predicted rate of change of the function in the chosen direction is quite wrong. If this occurs on a very early iteration, it's *possible* that it can be fixed by increasing the subiterations limit from its standard value of 30. However, you're more likely to have success in a situation like this by doing some preliminary simplex iterations (`PMETHOD=SIMPLEX,PITERS=5` for instance) to improve the initial guess values.

If it occurs later in the iteration process, it is usually the result of one of two problems:

1. Estimates at or near the boundary of the computable zone
2. Collapse to near-singularity of the Hessian

If you have a function value which can't be computed over some region (a variance goes negative, a covariance matrix becomes singular, a probability exceeds 1) and the estimates have moved close to that region, it may be hard for general hill-climbing methods to work because the function value may be going from computable to uncomputable over a short distance. If this is what has happened (check the coefficient values on the TRACE output), you might be able to push the estimates away by restarting the estimation with simplex iterations. If it still doesn't move, you've probably found at least a local maximum at the boundary. You'll need to try the entire process from a different set of initial guess values to see if there is a higher global optimum away from the boundary.

The collapse to near-singularity of the Hessian is the more likely culprit if you're using METHOD=BFGS. The instruction DISPLAY %CVTOCORR(%XX) (which displays the last inverse Hessian as a correlation matrix) will likely show you some off-diagonal values very near 1.0 (such as .9999). In some cases, you may be able to fix this by just re-executing the instruction, which will reinitialize the BFGS process. If, however, you have some parameters which are very highly correlated, this will probably quite quickly reproduce the problem. If that's the case, you may need to find a different (equivalent) parameterization. Concentrating out a parameter or parameters, can, where possible, help out quite a bit. And substitutions like $bx^p \Rightarrow b^*\left(x/x_0\right)^p$ and $\alpha+\varphi x_{t-1} \Rightarrow \alpha^*(1-\varphi)+\varphi x_{t-1}$, generally produce better behavior in the parameter pairs involved.

Slow Convergence (Many Iterations)

The more parameters that you try to estimate, the more iterations you're likely to need. If you have a function which seems to be making very slow progress, check the TRACE output, paying particular attention to the distance scale on the subiterations:

```
Subiterations 2. Distance scale  0.500000000
```

If you're consistently seeing distance scales of 1.0 or 0.5, you may just need to be patient. Numbers like those mean that the predictions for the behavior of the function value seem to be accurate. If, on the other hand, you're seeing short steps like

```
Subiterations 5. Distance scale  0.062500000
```

you may need to rethink the parameterization of the model as described above.

You can sometimes improve the behavior of the estimates by doing small sets of iterations on subsets of the parameters. Something like

```
maximize(iters=20,parmset=pset1,noprint) maxfrml
maximize(iters=20,parmset=pset2,noprint) maxfrml
maximize(parmset=pset1+pset2) maxfrml
```

The first **MAXIMIZE** will do 20 iterations of the parameters in PSET1 given the initial guess values in PSET2. The second will do 20 iterations on PSET2 given the refined

values of PSET1. Then the final **MAXIMIZE** will estimate the full set. Because the subiterations apply the same step size multiple to the entire parameter set, the overall process may be slowed down by a few poorly estimated parameters.

Slow Convergence (Time)

If each iteration takes a long time, there are two possible reasons: the function evaluations themselves are slow, or there are a large number of parameters. You could, of course, be dealing with both. If you need to speed up your function evaluations, look for situations where you do redundant calculations.

- If your FRML or FUNCTION includes a calculation which doesn't depend upon the parameters, do it once and save the results.
- If you have a calculation which doesn't depend upon time, but does depend upon the parameters, use the START option on the estimation instruction and compute it in that.
- Within a FRML, avoid making multiple references to the same sub-FRML. For instance, if RESIDA is a FRML, the following are equivalent, but the second computes RESIDA just once:

```
frml frontier = log(theta) - .5*resida**2 + $
                log(%cdf(-lambda*resida))
frml frontier = alpha=resida, $
     log(theta) - .5*alpha**2 + log(%cdf(-lambda*alpha))
```

If the size of the parameter set is what is slowing things down, you might want to think about the suggestion above for using subsets of the parameters. This will give you the biggest improvement in speed if you can optimize over a less complicated formula for different subsets.

Zero Standard Errors

If the model seems to converge reasonably, but your output shows that some coefficients have (true) zero standard errors, this will almost always be because that coefficient doesn't actually affect the function value. Most commonly, this is just a simple error of including a coefficient doesn't appear in the formula(s) being estimated. More subtly, it can be a parameter which multiplies a series or subcalculation which takes a zero value.