Running GAUSS programs Under Ox3

Philip A. Viton

October 9, 2003

Contents

1 Introduction ........................................ 2
2 Updates ............................................. 2
3 Principles of Conversion .......................... 3
4 OxEdit .............................................. 4
  4.1 Obtain and install OxEdit ...................... 4
  4.2 Configure OxEdit for Ox ........................ 5
  4.3 Configure OxEdit for OxGauss ............... 5
  4.4 Set up the Run buttons ....................... 5
  4.5 One-line comments ............................ 6
5 An extended example ................................. 6
  5.1 Get and install the Gauss code ............ 6
  5.2 Preliminary steps .............................. 7
  5.3 Round 1 — relocating the code ............. 7
  5.4 Round 2 — initializing globals ............. 8
  5.5 Round 3 — disambiguating the code ........ 9
  5.6 Round 4 — maxlik ............................ 9
  5.7 Round 5 — load ................................ 10
  5.8 Round 6 — indirection ....................... 11
  5.9 The final code ................................ 11
  5.10 Usability notes ............................... 12
    5.10.1 Optimization failures .................. 12
    5.10.2 Robust standard errors ............... 12
    5.10.3 Initial points .......................... 13
    5.10.4 Data conversion ......................... 13
6 Caveat and Conclusion ............................. 15
1 Introduction

This note describes how to get Gauss code to run under Jurgen Doornik’s Ox system, version 3.x. It is meant as a supplement to Jurgen’s own discussion in OxAppendix.pdf, distributed with Ox. From my perspective as a complete beginner, the problem with Jurgen’s write-up was that the examples turned out to be “too easy” to convert, and didn’t provide guidance on what to do when things didn’t go quite that smoothly.

For the academic user, the advantage of Ox + Gauss (OxGauss) is that it is in effect a free substitute for Gauss itself. (Note that this applies only for academic research and teaching: Ox is otherwise not free.) As a long-time user of Gauss remarked to me recently, Aptech seems to be making it much more difficult and expensive to use (for example, more difficult to have a copy of the software for home use when you have a licensed office version): indeed he went so far as to guess that if this keeps up, “Gauss’ days are numbered”. I don’t know whether this is an over-reaction, but if it is true, OxGauss provides a solution.

This note is aimed at the Gauss user who just wants to get programs running under Ox as quickly as possible. For this reason, I include details on setting up OxEdit, the free Ox text editor, for use with OxGauss. If you have another editor which allows you to run programs and capture their output, you should be able to get it working with OxGauss quite easily, though syntax highlighting may take a bit more effort to set up.

If you came to Ox after reading Francisco Cribari-Neto’s review in the Journal of Applied Econometrics (Vol 12, 1997, pp. 77–89), there’s one very important point to be made. Francisco mentions the availability of a Gauss-to-Ox converter (g2ox.exe). In my view, while using this converter might have been necessary in previous versions of Ox, you should now avoid it completely. As of version 3, Ox can run Gauss programs on its own, with very little fix-up on your part (as we’ll see below), and without converting them to the Ox language. If you do decide to give g2ox a try, be prepared for lots of (fairly routine) work; you could easily spend five to ten times as long on the conversion (experto crede) than if you use Ox’s “native” abilities, as described below.

2 Updates

This section lists updates to the discussion, for easy reference.

October 7, 2003:

• It appears that Ox3.3 has improved the OxGauss subsystem to the point that very little of the discussion of how to convert Gauss code for use in Ox appears necessary. In particular, Train’s mixed-logit code runs out-of-the box — no reordering necessary — if you just change the data-loading statement as described in Section 5.7. Also,
version 3.3 is significantly faster at computing the robust standard errors due to a change in the way Gauss’ \texttt{gradp} procedure is implemented in Ox. Obviously, everyone should upgrade.

- In my original write-up I noted that one couldn’t run Gauss routines which invoke its extra-cost \texttt{maxlik} module, and I suggested that it might be possible to map calls to \texttt{maxlik} into calls to Ox’s own optimizers. This has now been done by Sébastien Laurent and Jean-Pierre Urbain. Using their work it is possible to run the mixed-logit code even with the \texttt{maxlik} option specified (a couple of other add-on optimizers are also supported, but I’ve experimented with them). One reason this is important is that the included \texttt{domax} optimizer is highly sensitive to a number or parameters, for example the initial point and the number of Halton draws. The implementation of \texttt{maxlik} for OxGauss is available here.

March 5, 2003:

- \textbf{VERY IMPORTANT}: Train’s original code appears to have contained an error, whose symptom was that you got identical results when estimating a mixed logit model with uniform draws, no matter how many repetitions you used. The error has been silently corrected in the latest versions from his website. You should check that your answers change with the number of repetitions before you believe the results. It is not clear what \textit{was} being estimated in this case. The error does not seem to affect estimation with Halton numbers.

- As of now, Train appears to have withdrawn the two zip files originally referred to below. Instead there is a single zip file, \texttt{train0401.zip}, containing a variety of files associated with a course he taught on simulation methods in discrete choice. The mixed logit files are in the subfolder \texttt{ps05.mx1}, and the main file is \texttt{mxlp.g}. This is a panel-data formulation of the model; you can obviously adapt it to a non-panel-data context by specifying \texttt{TIMES = 1}. Because his two original archives are no longer available, I have withdrawn references to my converted code in the text below: you’ll have to go through the conversion exercise yourself; fortunately it shouldn’t take long. The upshot is that some of the line numbers referred to in the text will be a bit off; but the basic principles remain the same.

- Ox 3.2 has been released; and if you estimate the mixed logit model you really should upgrade. The computation of the robust standard errors (via \texttt{proc gradp}) is now \textit{much} faster than previously.

\textbf{September 13, 2001}: initial release.

\section{3 Principles of Conversion}

The main trick in converting Gauss code is to ensure that it meets two conditions:
• Any global variables must be declared before they are first used.

• Any subroutines (GAUSS procs) must be, too.

This means, in particular, that if the Gauss program is written in a top-down style (the main code at the top of the file, the procs below), you will need to do some shuffling of the code. What you’re aiming for is the following organization:

<table>
<thead>
<tr>
<th>Global variables (if any)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subroutines (procs)</td>
</tr>
<tr>
<td>Main startup code</td>
</tr>
</tbody>
</table>

If you keep that in mind, and develop some familiarity with Ox’s error messages, the conversion should go reasonably smoothly, as we’ll show via an example in section 5.

4 OxEdit

If you’re using the registered (non-free) version of Ox, then you’ve already set up the Windows interface, and you can skip this section. Otherwise I strongly urge you to get OxEdit, Jurgen Doornik’s general-purpose text editor. However, you don’t need OxEdit to be able to run Gauss code in Ox, and the next section tells you how to do so from a DOS command line. As soon as you try that, you’ll see just how inconvenient it is, so you might as well get a jump on the eventual outcome and set up OxEdit now.

4.1 Obtain and install OxEdit

OxEdit may be obtained from Jurgen Doornik’s site:

http://www.nuff.ox.ac.uk/Users/Doornik/index.html

Click on OxEdit in the left-hand frame, and follow the instructions.

When you’ve retrieved the distribution, double-click on it to start the installation. The program offers to install itself to the Program Files directory; personally I prefer to install it to a directory whose name doesn’t contain a space — it can make working with batch files a bit easier — but that’s up to you.
4.2 Configure OxEdit for Ox

We first configure OxEdit to handle both Ox and OxGauss code. In OxEdit’s terminology, this involves installing “modules”. We begin with Ox support. Start OxEdit.

- Click Preferences -> Add Predefined Modules
- In the next dialog, check the box marked Ox. Click OK.
- You’ll now be asked to point to the location of Ox’s bin directory. Find it, click OK.
- You get a message informing you that the Ox console has been successfully installed. Click OK.

4.3 Configure OxEdit for OxGauss

We’ll now set up OxEdit to handle OxGauss files. The first thing to do is settle on an extension. OxEdit allows your OxGauss code to have several extensions (.prg, .src, .e, .g, .gauss, .oxgauss), but since OxEdit modules apply to a single extension, it will make your life easier if you settle on one and stick with it. In the rest of this note I’ll assume you settle on .src (this is consistent with the Ox documentation). With OxEdit still running:

- Click Preferences -> Add/Remove Modules
- Select the Ox module. Click Clone Entry. This creates a new listing with a name something like Tool2. All we have to do now is edit a few of the entries for this Tool.
- In the Menu Text field, give the tool a name, like OxGauss
- In the Command field, add -g (with a preceding space) to the command already shown there.
- In the field for This module is for files with extension enter your chosen extension, here src (without a period).
- Click Close.

4.4 Set up the Run buttons

This step isn’t strictly necessary, but it will make working with OxEdit easier. We’ll set up the ability to run OxGauss programs with a single mouse click on an icon on the main toolbar.
• Place the cursor on one of the icons on the main toolbar (not on the blank space to the right) and right-click. A Customize Toolbar dialog opens. The Separator at the end of the right-hand window “Current Toolbar Buttons” should be selected; if not, select it.

• In the left-hand window (“Available Toolbar Buttons”) find and select the one called Run Default. Click the Add button.

• Find and select the one called Run 2nd Default. Click the Add button.

• Finally, add a bit of space between these two icons and the previous one: in the Available Toolbar Buttons window, select the Separator button (first one in the list), and select Run Default in the Current Toolbar Buttons. Click Add.

• Click Close to exit the dialog.

4.5 One-line comments

This step is also optional. I find OxEdit’s ability to comment-in and comment-out blocks of code with a single mouse click extremely useful in testing my work; but if you want to be able to do this, you need to set it up yourself. Open the file gauss.def (in your OxEdit directory). Find the [options] section, near the end, and in this section add

```
comment_eol=//
```

This makes lines starting with // act as comments. Note that this is not supported in Gauss itself, so you will need to remove any of these comments if you want your code to run under Gauss.

That’s it. OxEdit is now ready to be used to convert Gauss files.

5 An extended example

This section provides an extended example of converting a Gauss file to an OxGauss one. We shall convert Kenneth Train’s Gauss code for estimation of the panel-data mixed-logit model to Ox.

5.1 Get and install the Gauss code

Train’s code is available from his Home Page at UC Berkeley:
Click on Software then Mixed logit estimation for panel data. The file you want is train0299.zip. You may also want to retrieve the other papers listed on that page. If you’re interested in mixed logit, note that Train has other Gauss code available on other pages of his site.

When you’ve retrieved the zip file, create a working directory for it, and unzip the files to that area.

5.2 Preliminary steps

I’m assuming that your chosen OxGauss extension is .src, so the first thing to do is to make a copy of the original Gauss code (here, mxlp.g) as mxlp.src.

If you’re not using OxEdit, start a DOS session, switch to the directory containing mxlp.src. To try to compile the code under Ox, you say

\[ \text{c:\ox\bin\oxl.exe -g mxlp.src} \]

(where you will have to alter the path to the binary file if you installed it to a different place). When do this, you’ll see a series of messages from the Ox compiler, and you’ll probably want to save them somewhere. You could do this by redirecting the messages to a file, using something like \[ \text{c:\ox\bin\oxl.exe -g mxlp.src > mxlp.err} \], but it is far easier to use OxEdit, where the messages will be conveniently displayed in their own window, and where there is a one-click interaction between an error message and the corresponding line in the source code itself.

So from now on I’ll assume you’ll use OxEdit. Start OxEdit, and read in mxlp.src. You’ll see that the display is syntax-colored. You may want to enable line numbering: click View -> Line numbers.

5.3 Round 1 — relocating the code

OK, time to see what we’re in for. Click the Run Default icon. Ox tries to compile and run the program; the result is that a window opens up, containing about 20 lines of error reports. The first thing to note, of course, is how very few messages there are: after all, the source is about 1400 lines of Gauss code. So the prospects look good.

Many of the error messages are of the form

\[ \text{C:\OxWork\train2\mxlp.src (414): 'gauss::ll' undeclared identifier} \]
and we recognize the ll in gauss:ll as referring to a Gauss proc defined in the code (in this case, the one for computing the log-likelihood). What this is telling us is that the proc was used in line 414 before being declared. Train adopts a top-down style: first set up the main routine, then define the subroutines. To fix this, we shall move all the top-level code to the end of the file. But we need to be careful: the top-level code also declares several global variables, and these need to remain at the beginning, because they are referred to by many of the other routines.

Fortunately, it is fairly easy to see where the global declarations end, and the executable code begins: it’s about line 205 of the program:

```
200 @=====================================================================@
201 @@@ You should not need to change anything below this line @@
202 @=====================================================================@
203 @ Create global for the number of estimated variables @
204 NEVAR = NFC+(NNC+NUC+NTC+NLC)*2;
205
206 @ Check inputs if VERBOSE=1 @
207 if ((VERBOSE /= 1) and (VERBOSE /= 0));
```

and the main code extends down to about line 454, which is thoughtfully marked for us (but note that even if it wasn’t, we could locate it by searching for the first proc statement):

```
454 @ THIS IS THE END OF THE MAIN PROGRAM. @
455 @ PROCS ll, gr, gpnr, expand, domax, pcheck follow.@
456
457 /* LOG-LIKELIHOOD FUNCTION */
458 proc ll(b,x);
```

So we cut out everything between lines 205 and 454 (inclusive), and paste the relocated code at the end of the file.

### 5.4 Round 2 — initializing globals

Click the Run Default icon. We note that we’ve gotten rid of most of the errors relating to procs; but we’ve picked up a set of new “undeclared identifier” errors, for example

```
C:\OxWork\train2\mxlp.src (256): 'gauss::hm' undeclared identifier
```

This time we recognize the names of variables. This is a variant of the previous problem: the variables are being used before being declared, and the solution is to initialize them.
The simplest way to do this is to \texttt{clear} them, and we insert the following lines at the end of the global variables (just before the first \texttt{proc} declaration), with a comment to remind ourselves that we’ve added something to Train’s original code:

\begin{verbatim}
@ initialize additional globals for OxGauss @
clear necol, hm, ida, yperm, hmname, maxset, _max_finalhess;
\end{verbatim}

\section*{5.5 Round 3 — disambiguating the code}

When we run the program this time, we’re down to about 13 error messages, most of which are of the form

\begin{verbatim}
C:\OxWork\train2\mxlp.src (254): warning: dot part of number, not dot operator
\end{verbatim}

What’s happening is that Gauss permits you to write (for example) term-by-term multiplication by 2 as \texttt{2.*} but OxGauss doesn’t know whether the dot is attached to the \texttt{2} or to the \texttt{*} symbol. We can use OxEdit’s interaction between the output window and the main code: select one of the errors, and double-click. We’re immediately placed at the appropriate line in the Gauss code, and all that remains to do is disambiguate it. In this case the dot should be attached to the \texttt{*} for term-by-term multiplication: the simplest way to ensure this is to change \texttt{2} to \texttt{2.0}. So for example, when we see

\begin{verbatim}
err[.,(NNC+NUC+1):(NNC+NUC+NTC)] = (sqrt(2.*mm)-1).*(mm .<= 0.5)
\end{verbatim}

we just change it to

\begin{verbatim}
err[.,(NNC+NUC+1):(NNC+NUC+NTC)] = (sqrt(2.0.*mm)-1).*(mm .<= 0.5)
\end{verbatim}

and similarly for the rest of these errors.

\section*{5.6 Round 4 — \texttt{maxlik}}

This time when we run the program we’re down to about 8 errors. But these require a bit more understanding of the original code.

The last two errors

\begin{verbatim}
C:\OxWork\train2\mxlp.src (1372): 'maxlik' file not found
C:\OxWork\train2\mxlp.src (1373): 'maxlik.ext' include file not found
\end{verbatim}
are from part of the code which calls Gauss's `maxlik` routine. This is an extra-cost add-on, and not even all GAUSS installations have it; fortunately Train's code provides an alternative, a maximization routine called `domax` written by Paul Ruud. The offending call to `maxlik` occurs at about line 1370 of the modified code (the numbers may not match up exactly, depending on any comments or blank lines you inserted when making the previous changes) as part of the if `OPTIM == 2` conditional. What we'll do is insert a warning message, and then comment out the rest of the conditional. We begin with the if statement, add our warning, and close the conditional:

```gauss
if OPTIM==2
   print "ERROR : OPTIM == 2 not supported by OxGauss. Re-run with OPTIM = 1";
   stop;
endif;
```

after which we comment-out the rest of the original conditional down to the concluding `endif` (which in this case happens to be the end of the code itself). If you set up the one-line-comment feature of OxEdit (see section 4.5) you can do this by highlighting all the code to be commented-out and then clicking the Comment-out icon on the main toolbar. Note that we now have no need of the variables `maxset` and `_max_finalhess` which we declared earlier, so we might as well remove them from the `clear` statement (though not doing so is harmless).

### 5.7 Round 5 — `load`

The next errors relate to the beginning of the code, at around lines 32–34, having to do with the data loading:

```gauss
32 load XMAT[NOBS,28] = c:\temp2\xmat.asc;
33 load YVEC[NOBS,1] = c:\temp2\yvec.asc;
34 load TIMES[NP,1] = c:\temp2\times.asc;
```

The problem here is that OxGauss' implementation of the `load` statement expects numeric constants — not variables — as arguments. This is not obvious from the documentation; and this is a good time to remind you that Jurgen Doornik maintains a mailing list for Ox-related questions, and this might be a good time to consider joining.

The best solution is to `load` the data as a `vector` and then re-shape it to the desired dimensions. This works because, unlike `load`, `reshape` allows variables as arguments. At the same time, we'll change the location of the files to reflect the fact that our code and the data files are in the same place. The upshot is that we replace the lines above by

```gauss
load XMAT[] = xmat.asc;
```
XMAT = reshape(XMAT,NOBS,28);
load YVEC[] = yvec.asc;
YVEC = reshape(YVEC,NOBS,1);
load TIMES[] = times.asc;
TIMES = reshape(TIMES,NP,1);

There is one important thing to remember. If you specify a file name without extension, OxGauss assumes that the extension is .mat; and this is probably not what you want. In the present case there’s no problem, since Train’s original data files have an extension; but in other cases you may need to rename data files (or add an extension if they don’t already have one) before loading them in this way.

5.8 Round 6 — indirection

The final error involves HMNAME. What’s happening here is that we are trying to save to a file whose name is specified indirectly via the data in this variable. Perhaps a better solution is possible; but what I’ll do is to use Ox’s automatic preprocessing stage as a text-replacement utility: I’ll replace the text HMNAME by a value, specified in a preprocessor instruction. To do this,

- Comment-out the line HMNAME="c:\temp2\hm125.fmt";
- Add a preprocessor instruction below it:

```
#define HMNAME hm125.fmt
```

and note that because this is a preprocessor instruction, it does not end with a semi-colon, nor is there an equal sign between definiens and definiendum.

- Remove HMNAME from the data-initialization (clear) statement.
- Finally, replace the two instances of "HMNAME by HMNAME. (Yes I know, only one instance was noted in the error report; but if you run the code after fixing just this one, the second will be flagged).

5.9 The final code

When these changes are made, the Gauss codeshould run to completion. If you are using Ox 3.0 there’s a long wait until it computes the robust standard errors, so don’t give up. You can get a very considerable speedup by upgrading to Ox 3.2 or later.
5.10 Usability notes

This section contains some notes on using the mixed-logit routines themselves (as opposed to the details of getting them to work under Ox).

5.10.1 Optimization failures

Train remarks somewhere that the lognormal specification is hard to estimate, and can easily result in a failure of the optimization routine with a singular Hessian. I think that the problem is more general than this: I have had relatively simple models fail under any stochastic (ie, not standard logit) specification. I have often been able to get the optimization to run to completion by such unexpected techniques as (a) adding one additional Halton repetition to the problem setup (eg going from 125 to 126 Halton repetitions), or (b) using \(3\) as the initial prime from which the Halton sequences are generated. I’ve no idea why these help, but they do. Alternatively, it is possible, with a little work, to adapt the DFP routines available at the Gaussian website at American University, http://gurukul.ucc.american.edu/econ/gaussres/gaussidx.htm to work with Train’s code. (Note, if you try this, that the routines in the DFP package minimize, while Ruud’s optimizer maximizes; and that the new optimizers are much slower than Ruud’s. On the other hand, they seem to succeed more consistently.)

5.10.2 Robust standard errors

The mixed-logit programs can compute robust standard errors via elements of \(H^{-1}g'gH^{-1}\), where \(H\) is the Hessian matrix and \(g\) the gradient vector (at convergence). In the Gauss code, \(g\) is referred to as grad, and is meant to be available in proc domax as the optimization algorithm reaches convergence. The Hessian is computed via a call to the built-in routine gradp. As best I can tell, calling gradp has the side-effect of altering grad, so that after the Hessian is computed, you are no longer looking at the correct grad. In many cases the error is small, but it can alter the results (and in particular make them dependent on the order of the variables in the problem).

I don’t know whether this is a problem with the original Gauss code, or with Ox’s interpretation of it. I incline to the latter, since it looks to me as if grad is a local variable everywhere, and hence should not be changed by the Hessian computation. Be that as it may, there’s a simple fix, namely reset (re-compute) grad after the Hessian is computed. In proc domax, after the call to gradp, insert \(\text{grad} = \text{gr(b, XMAT)}\). (This is at about line 761 of the code).
5.10.3 Initial points

The order of the elements of the parameter vector depends on the model’s specification, so every time you change the specification, you need to adjust the initial points. A better approach is to do this automatically by setting up a matrix whose \(i\)-th row contains initial points for the mean and deviation of variable \(i\), and then have the program use them as the model specification demands. Here’s a sketch of some code which does that:

[@ construct the matrix ibp of initial points @]
ibp = zeros(NVAR,2);
[@ NB NO COMMAS separating the entries @]
[@ first entry is for mean, second for deviation @]
ibp[1,] = [-.90 1.00];
ibp[2,] = [-.21 .39];
[@ etc etc @]

B = {};
if nfc ge 1; b = b |ibp[idfc,1] ; endif;
if nnc ge 1; b = b |reshape(ibp[idnc,],(2*nnc),1); endif;
if nuc ge 1; b = b |reshape(ibp[iduc,],(2*nuc),1); endif;
if ntc ge 1; b = b |reshape(ibp[idtc,],(2*ntc),1); endif;
if nlc ge 1; b = b |reshape(ibp[idlc,],(2*nnc),1); endif;

5.10.4 Data conversion

Programs like Limdep expect their data to be arranged in rows by individual and then by alternative, as in

<table>
<thead>
<tr>
<th>Var 1</th>
<th>Var 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ind 1</td>
<td>Alt 1</td>
</tr>
<tr>
<td></td>
<td>Alt 1</td>
</tr>
<tr>
<td></td>
<td>Alt 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B = {}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ind 1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

while Train’s Gauss programs expect it to be arranged as

<table>
<thead>
<tr>
<th>Var 1</th>
<th>Var 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ind 1</td>
<td>Alt 1</td>
</tr>
<tr>
<td></td>
<td>Alt 2</td>
</tr>
<tr>
<td>Ind 1</td>
<td>Alt 1</td>
</tr>
<tr>
<td></td>
<td>Alt 2</td>
</tr>
<tr>
<td>Ind 1</td>
<td>Alt 1</td>
</tr>
<tr>
<td></td>
<td>Alt 2</td>
</tr>
</tbody>
</table>

13
The following Ox code — Ox, not Ox-Gauss — will convert a dataset in Limdep form to that required by Train's Gauss code, as long as each person faces all alternatives in the choice set. It should be straightforward to modify this to cope with cases of varying choice sets.

/*
 Convert a limdep-like dataset to the form required by Ken Train’s
 MMNL programs
 */

#include <oxstd.h>

main()
{
 decl np, nobs, nalt, nvar, i ;
decl data, newdata ;
decl f1, infile, f2, outfile;

infile="c:/limdep7/work/tdfp/775.dat";
outfile="c:/OxWork/tdfp/775.asc";

// number of persons
np = 777;
// no. of alts per person. NB we assume everyone faces all nalt choices
nalt= 4;
// nobs is therefore the number of rows in the data
nobs = np * nalt;
// nvar is the number of columns in the limdep version
nvar = 21;

f1 = fopen(infile);
if (!isfile(f1))
{
 println("Failed to open file : ",infile, " - quitting");
 exit(1);
}

fscan(f1,"%#m",nobs,nvar,&data);
fclose(f1);

newdata = <>;
for (i=0 ; i < nvar ; i++)
{
    newdata = newdata ~reshape(data[][i],np,nalt);
}

println("newdata has ", sizer(newdata)," rows and ", sizec(newdata), " columns");

// now write out the data using the default format
f2 = fopen(outfile, "w");
fprint(f2, "%#M", newdata);
fclose(f2);
println("Data has been written");

exit(0);
}

6 Caveat and Conclusion

This long-ish write-up shouldn’t blind you to the fact that amazingly little fixup was necessary to get Train’s code working under Ox. And of course, the next time you need to convert some Gauss code, it should go much faster.

However, not every piece of Gauss code will be convertable to OxGauss. As we’ve seen, one important case is when it uses Gauss’s maxlik procedure; or more generally, other extra-cost Gauss add-ons. Even in the maxlik case it may be possible to substitute Ox’s own optimization routine; if anyone knows how to do this, I’d like to know. In the mixed-logit case we were fortunate in being provided with an alternative optimization routine domax written in Gauss as a substitute to maxlik; and this could be converted. So another possibility is to map maxlik and its output generally to domax. Again, if anyone does this and would like to explain it, I’d be glad to include the information here.