The Stata Journal

The *Stata Journal* publishes reviewed papers together with shorter notes or comments, regular columns, book reviews, and other material of interest to Stata users. Examples of the types of papers include 1) expository papers that link the use of Stata commands or programs to associated principles, such as those that will serve as tutorials for users first encountering a new field of statistics or a major new technique; 2) papers that go "beyond the Stata manual" in explaining key features or uses of Stata that are of interest to intermediate or advanced users of Stata; 3) papers that discuss new commands or Stata programs of interest either to a wide spectrum of users (e.g., in data management or graphics) or to some large segment of Stata users (e.g., in survey statistics, survival analysis, panel analysis, or limited dependent variable modeling); 4) papers analyzing the statistical properties of new or existing estimators and tests in Stata; 5) papers that could be of interest or usefulness to researchers, especially in fields that are of practical importance but are not often included in texts or other journals, such as the use of Stata in managing datasets, especially large datasets, with advice from hard-won experience; and 6) papers of interest to those who teach, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

The *Stata Journal* is indexed and abstracted by *CompuMath Citation Index*, *Current Contents/Social and Behavioral Sciences*, *RePEc: Research Papers in Economics*, *Science Citation Index Expanded* (also known as *SciSearch*), *Scopus*, and *Social Sciences Citation Index*.

For more information on the *Stata Journal*, including information for authors, see the webpage

http://www.stata-journal.com

Subscriptions are available from StataCorp, 4905 Lakeway Drive, College Station, Texas 77845, telephone 979-696-4600 or 800-STATA-PC, fax 979-696-4601, or online at

http://www.stata.com/bookstore/sj.html

Subscription rates listed below include both a printed and an electronic copy unless otherwise mentioned.

U.S. and Canada		Elsewhere	
Printed & electronic		**Printed & electronic**	
1-year subscription	$ 98	1-year subscription	$138
2-year subscription	$165	2-year subscription	$245
3-year subscription	$225	3-year subscription	$345
1-year student subscription	$ 75	1-year student subscription	$ 99
1-year institutional subscription	$245	1-year institutional subscription	$285
2-year institutional subscription	$445	2-year institutional subscription	$525
3-year institutional subscription	$645	3-year institutional subscription	$765
Electronic only		**Electronic only**	
1-year subscription	$ 75	1-year subscription	$ 75
2-year subscription	$125	2-year subscription	$125
3-year subscription	$165	3-year subscription	$165
1-year student subscription	$ 45	1-year student subscription	$ 45

Back issues of the *Stata Journal* may be ordered online at

http://www.stata.com/bookstore/sjj.html

Individual articles three or more years old may be accessed online without charge. More recent articles may be ordered online.

http://www.stata-journal.com/archives.html

The *Stata Journal* is published quarterly by the Stata Press, College Station, Texas, USA.

Address changes should be sent to the *Stata Journal*, StataCorp, 4905 Lakeway Drive, College Station, TX 77845, USA, or emailed to sj@stata.com.

The Stata Journal (2014)
14, Number 2, pp. 342–362

From Stata to aML

Sara Ayllón
Department of Economics and EQUALITAS
Universitat de Girona
Girona, Spain
sara.ayllon@udg.edu

Abstract. In this article, I explain how to exploit Stata to run multilevel multiprocess regressions with applied maximum likelihood (aML). I show how a single do-file can prepare the dataset, write the control files, input the starting values, and run the regressions without the need to manually open the aML's Command Prompt window. If desired, results can be brought back to Stata for postestimation. I also provide an example that illustrates how well Stata and aML work together.

Keywords: st0340, aML, file open, file write, multilevel multiprocess models

1 Introduction

In this article, I show how to use Stata to run multilevel multiprocess models with applied maximum likelihood (aML).[1] The aML statistical software has three main advantages. First, it easily handles three or more data levels that can be mixed and matched as desired. Second, it is able to jointly fit models with two or more different outcomes or data structures (for example, a probit together with a hazard regression). Third, it makes it easy to program complex error structures.[2]

However, aML is not a user-friendly program. It requires the preparation of the dataset using a third-party software package. aML needs two different control files: one to upload the data and another to run the regressions. It also requires the researcher to typeset the starting values, which is tedious and time consuming, especially when working with a long list of explanatory variables or in comparative analysis. Finally, aML needs the starting values to be assigned in the same order as the regressor set, which introduces the potential for user error when assessing different model specifications.

1. aML can be downloaded free of charge from http://www.applied-ml.com/. Note that aML is not supported for Mac computers.
2. The command `cmp` by Roodman (2011) shares some of the features of aML. However, and as admitted by Roodman, aML is substantially broader than `cmp`. aML allows full simultaneity in systems of equations. It can also account for random coefficients and random effects at different cluster levels, and it offers a wider range of model types (for example, logit, multinomial logit, Poisson, exponential hazard models, etc.). The `gllamm` command by Rabe-Hesketh et al. (2004) includes some of the mentioned features but only for single-equation models.

In what follows, I show how it is possible to write a single Stata do-file that can prepare the dataset, write the control files, automatically input the starting values, and call aML using the `shell` command to upload the data and run the regressions.[3] At the end of the process, the aML user obtains the output file with the results without the need to even open the Command Prompt window used by aML. Also because Stata automatically generates all the control files, writes the list of explanatory variables, and typesets the starting values, there is no margin for typo errors. Moreover, working with a single Stata do-file permits changes in the model specification without the need to open and change each control file. I also show how to bring aML results back to Stata. In the last section of the article, I illustrate the process with an example that uses a dataset contained in Stata.

2 From Stata to aML, step by step

To exemplify the different steps we need to go through, let's imagine that we want to run a simultaneous multiprocess model in aML composed of two random-effects probits with a dataset that contains two levels of data.[4] For example, let's imagine a panel that has information on individuals (identified by `pid`) during several waves (`year`). The model controls for observed and unobserved heterogeneity while also permitting individual-specific effects to be freely correlated across equations. (aML easily deals with more complicated models, but I shall keep it simple for illustrative purposes.)

2.1 Data preparation

First, we need to write the list of explanatory variables that will be entered in the estimations. Because aML needs to know which variables are at which level, we have to organize them. In our case, we distinguish between time-invariant (level 1) and time-varying (level 2) variables.[5] We create a local macro in Stata that contains the name of the explanatory variables in each level. Assuming that the different regressions contain the same time-invariant variables but different time-varying variables, we can create level 1 for both regressions, level 2a for the first equation, and level 2b for the second equation:

3. Other commands connect Stata with software packages for running multilevel models. See, for example, the modules `runmlwin` by Leckie and Charlton (2013) for MLwiN, `hlm` by Reardon (2006) for HLM, `runmplus` by Jones (2010) for MPlus, or `bugsutils` by Greenwood (2006) for WINBUGS.
4. In the following section, I will show a variation of the code with a model that jointly estimates a random-effects linear regression together with a probit.
5. Note that in aML, level 1 is the highest level, which is the level that contains the most aggregated data. In a dataset with information on municipalities in regions of different countries, level 1 would be the countries, level 2 the regions, and level 3 the municipalities.

```
local level1 ""
local level1 "`level1´ ivar1"
local level1 "`level1´ ivar2"

local level2a ""
local level2a "`level2a´ ivar3"
local level2a "`level2a´ ivar4"

local level2b ""
local level2b "`level2b´ ivar3"
local level2b "`level2b´ ivar5"
```

For example, `ivar1` could be the time-invariant explanatory variable `female`. Note that one is able to easily omit the independent variables from the estimation by simply typing `*` in the appropriate line. For example, to see how the results would change when leaving the independent variable 4 out of the final regression, `*` should be added at the beginning of the sixth line above: `*local level2a "‘level2a’ ivar4"`. The do-file we are building automatically takes the variable out of the regressor set and the list of starting values.

Variable names in aML are limited to eight characters, so it is important to keep them short in Stata as well. Names are also case sensitive and should not start with an underscore (_), because variables that begin with this symbol are automatically created by aML. Also aML does not support character-string variables.

Second, we need to consider that aML does not accept missing values or variable labels. A few lines and a loop can make sure of that. Moreover, aML works faster if we keep only those variables that are strictly necessary. In what follows, `dvar1` and `dvar2` refer to the dependent variables.

```
keep pid year dvar1 dvar2 `level1´ `level2a´ `level2b´
label drop _all
foreach var of varlist `level1´ `level2a´ `level2b´ {
    drop if `var´==.
}
drop if dvar1==.
drop if dvar2==.
save dataset.dta, replace
```

The dataset can be saved at this point for later use (see section 2.4 below).

Once there are no missing values left in our dataset, we run separate regressions in Stata that will allow us to save the starting values that aML requires to initialize the estimations. The following code saves the starting values in a `.txt` file (which will be deleted at the end of the process), the standard deviation of the random effect in another `.txt` file, and the variables list in a `.dta` file.[6] It is very important in aML for the starting values to be assigned in the same order as the regressor set, because the names of the parameters are unimportant but the order in which they are specified is important. That is, when a parameter is deleted or included, the control files need to be modified accordingly. In our case, the do-file automatically takes this into account, thus avoiding any potential error.

6. Note that `estout` must be installed. `estout` was created by Jann (2005) for producing regression tables from stored estimates. See also Jann (2007).

The code for the two separate random-effects models reads as follows:

```
xtprobit dvar1 `level1´ `level2a´, i(pid)
estimates store m1
estout m1 using sv1.txt, replace style(tab) order(_cons) mlabels("") collabels("")
generate eps1=`e(sigma_u)´
outfile eps1 using eps1.txt if _N==_n, replace

preserve
clear
insheet using sv1.txt
drop if v2==.
drop v2
drop if v1=="_cons"
format v1 %15s
save sv1.dta, replace
restore

xtprobit dvar2 `level1´ `level2b´, i(pid)
estimates store m2
estout m2 using sv2.txt, replace style(tab) order(_cons) mlabels("") collabels("")
generate eps2=`e(sigma_u)´
outfile eps2 using eps2.txt if _N==_n, replace

preserve
clear
insheet using sv2.txt
drop if v2==.
drop v2
drop if v1=="_cons"
format v1 %15s
save sv2.dta, replace
restore
```

Note that `order(_cons)` obliges the constant to be the first variable in the list of covariates, as required by aML. The text between `preserve` and `restore` saves the list of parameters that will have the same name as the variables (thus avoiding confusion).

Third, we will `outfile` the data as a `.txt` file that is easily read by aML given that it requires the data to be in ASCII (raw text) format. aML will convert the data into aML's binary format with the `raw2aml` command. The data need to be sorted by the variable that reports level 1 (in our case, the individual `pid`) and level 2 (`year`). When outfiling the data, the level 1 variable must always be the first in the list.

Data can be easily organized into different data structures for each process to be estimated. Each data structure may have different levels, different explanatory variables, different samples, etc. For example, we may wish to jointly fit one model for males and another for females. If one needs to restrict the sample, this should be done before outfiling it.[7] In the example, there was no need to create two data structures; I have only done so for illustrative purposes.

7. Remember, if we restrict the sample, we also need to do so when running the separate regressions and saving the starting values.

The variable `structure` always has to be the second variable in the list when outfiling the data. (If there are more than two levels of data, add after `structure` the name of the variables that tell aML how many subbranches there are.)

Though optional, I also create an ID file that links the different ASCII files as one and tells aML about all possible IDs if the different data structures contain different samples. Thus I outfile the person identifiers once sorted by the level 2 variable (and other subbranch variables if the data contain more than two levels). If using very long identifier numbers, we need to make sure that they are in a format that can be fully read by a `.txt` file. One can easily change the format of the identifier if needed (for example, `format pid %13.0g` for person identifiers that contain fewer than 13 numbers).

```
generate structure = 1
sort pid year
outfile pid structure `level1´ `level2a´ dvar1 using data1.txt, wide comma replace
replace structure = 2
sort pid year
outfile pid structure `level1´ `level2b´ dvar2 using data2.txt, wide comma replace
keep pid year
sort pid year
outfile pid using id.txt, wide comma replace
```

Finally, we can start writing the control files that aML requires to upload the data. To do this, we will use the commands `file open` and `file write` (see Gini and Pasquini [2006]). Let's start with the file `data.r2a` that uploads the data in aML. It must contain the names of the ASCII input files that we previously outfiled and the level hierarchy of the explanatory variables that have been used. Outcome variables may be specified at any level.

```
clear
file open data using data.r2a, write replace
file write data "ascii data files = data1.txt" _n
file write data "                   data2.txt" _n
file write data ";" _n
file write data _n
file write data "id file = id.txt;" _n
file write data _n
file write data "level 1 var = " _n
file write data "`level1´" _n
file write data ";" _n
file write data _n
file write data "data structure = 1;" _n
file write data _n
file write data "level 2 var = " _n
file write data "`level2a´" _n
file write data "dvar1" _n
file write data ";" _n
file write data _n
file write data "data structure = 2;" _n
file write data _n
file write data "level 2 var = " _n
file write data "`level2b´" _n
file write data "dvar2" _n
file write data ";" _n
file close data
```

Note the semicolon (`;`) at the end of each line, as required by aML, and how `_n` will help the file to be written in different lines. This is important because aML does not read lines if they have more than 80 characters, including spaces. If planning to use a long list of explanatory variables, we must take this into account. We can easily resolve the issue by organizing our variables into more groups (local macros) and then adding `file write data "‘levelx’"` as many times as the number of groups created. It is important to be organized: the groups of variables must be entered in the same order that they were previously outfiled. Note that in the variables list, there are no aML control variables such as, for example, `structure`.

2.2 Model specification

Once the dataset is prepared, we need to write the `.aml` file where the regressions are specified. This is a little more complicated, so we will write it in parts that will be saved as `.dta` files in Stata and later appended to create one single file.

At the beginning of the `.aml` file, aML needs to be told which dataset is to be used and which regressor set for each of the equations is to be estimated. After that, any desired option can be added (see Lillard and Panis [2003]). I include the line with `option variance-covariance matrix` so that it is printed in the aML output. To generate an intercept, write `"var=1"` because this is not assumed by aML. When not willing to specify a constant, you may instead type `"var= "`.

```
clear
input str50 v1
"dsn = data.dat;"
"option variance-covariance matrix;"
"define regressor set Beta1;"
"var = 1"
end
save rsbeta1.dta, replace

clear
input str50 v1
"define regressor set Beta2;"
"var = 1"
end
save rsbeta2.dta, replace

clear
input str50 v1
";"
""
end
save semicolon.dta, replace
```

Next we need to specify the type of model we want to regress, the number of integration points, and the different error terms. Refer to the aML user’s guide (Lillard and Panis 2003) for full details of the different possible models, how to specify the error terms, and all the different options. For example, the following code indicates that we have a simultaneous model with two probits with random effects that need to be integrated out and are correlated across equations:

```
clear
input str50 v1
"define normal distribution;"
"dim=2;"
"number of integration points = 6;"
"name = eps1;"
"name = eps2;"
" "
"probit model;"
"data structure = 1;"
"outcome = dvar1;"
"model = regset Beta1 + intres(draw=1, ref=eps1);"
" "
"probit model;"
"data structure = 2;"
"outcome = dvar2;"
"model = regset Beta2 + intres(draw=1, ref=eps2);"
" "
"starting values;"
end
save estimation.dta, replace
```

Once the models have been specified, we need to write the list of parameters, whether the parameter is to be estimated, and their starting values. To do so, we write a single string (called `all`) that contains the required information and places blank spaces in between. Because aML uses full information maximum likelihood with an iterative search algorithm, we need to specify which parameters are to be estimated (`T`, true) and fixed (`F`, false). In this example, I let the constants settle and then free up the parameters. Because both models in this example have the same specification, we can write a loop. If not, a specification for each model needs to be written.

```
forvalues i=1(1)2 {
    clear
    insheet using sv`i´.txt
    drop if v2==.
    generate str4 ft="FTTT"
    replace ft="TTTT" if v1=="_cons"
    order v1 ft v2
    tostring v2, generate(v2a) format(%10.6f) force
    drop v2
    rename v2a v2
    generate str4 blanc="          "
    generate str50 all=substr(v1,1,.)+substr(blanc,1,.)+substr(ft,1,.)+ ///
       substr(blanc,1,.)+substr(v2,1,.)
    drop v1 ft v2
    rename all v2
    * Next two lines make the _cons of lnsigmau to drop
    generate x=_n
    drop if x==_N
    drop x blanc
    save values`i´.dta, replace
}
```

At the bottom of the file, we need to add the starting value of the standard deviation of the random effects that are integrated out.

```
forvalues i=1(1)2 {
    clear
    insheet using eps`i´.txt
    generate x=_n
    keep if x==1
    generate str10 blanc="          "
    generate str4 v2="eps`i´"
    generate str4 ft="FFTT"
    tostring v1, generate(v1a) format(%10.6f) force
    generate str50 all=substr(v2,1,.)+substr(blanc,1,.)+substr(ft,1,.)+ ///
       substr(blanc,1,.)+substr(v1a,1,.)
    keep all
    save eps`i´.dta, replace
}
```

Finally, I add `rho`, which is the correlation between random effects of the different equations. This needs to be modified accordingly: if it is believed to be positive, set a positive number; if negative, do the opposite. One can always initialize it at 0.

```
clear
input str50 v1
"rho12 FFFT 0"
end
save rho.dta, replace
```

Note that four rounds of estimation are proposed in total: first, only the constants; second, the parameters; third, the standard deviations of the random effects; and finally, the correlation.

All that remains is to build the `.aml` file by appending its different parts: the regressor sets (`rsbeta`) with the list of covariates (`sv`), the estimation specification and errors structure (`estimation`), the starting values (`values`), and finally, the error terms and the rho (`eps`, `rho`). I outfile this as an `.aml` file:

```
use rsbeta1.dta, clear
append using sv1.dta
append using semicolon.dta
append using rsbeta2.dta
append using sv2.dta
append using semicolon.dta
append using estimation.dta
append using values1.dta
append using values2.dta
append using eps1.dta
append using eps2.dta
append using rho.dta
append using semicolon.dta
outfile using estimate.aml, replace noquote wide
```

To remove unnecessary files from the computer, we delete all the parts that have helped to compose the final `.aml` file:

```
forvalues i=1(1)2 {
    erase sv`i´.txt
    erase eps`i´.txt
    erase rsbeta`i´.dta
    erase values`i´.dta
}
erase rho.dta
erase semicolon.dta
erase estimation.dta
```

2.3 Model estimation

Now all we have to do is call aML within Stata by using the `shell` command, which will open the Command Prompt window (DOS window) and close it when the regressions are done.[8] First, we will upload the data with the `raw2aml` command. Second, we will estimate the regression with `aml`. And finally, we update the obtained results. At this point, it is very important to make sure that the file `data.r2a` does not contain more than 80 characters per line; otherwise, the data will not upload. If this is the case, you can manually break up the lines or work with more groups of dependent variables (as explained). The code reads as follows:

```
shell raw2aml -r data.r2a
shell aml -r estimate.aml
shell update -r estimate.out
```

After the model has been fit, you obtain the file `data.sum`, which gives a summary of the data file, where we can check whether the data were read as expected. We also obtain the file `estimation.out`, which contains the results, and a dataset in aML format called (in our case) `data.dat`.

8. One can manually open aML and type the same lines (one by one) without `sh` at the beginning of the statement. This way, the screen output can also be used if desired.

2.4 And, back to Stata

For convenience, you may wish to bring aML results back to Stata to perform postestimation tests or simply produce formatted tables.[9] To do that, we need to create an estimates table with the `mktab` utility in aML that will be imported into Stata with the `insheet` command. Given that in aML, we cannot produce a table that contains coefficients and standard errors without significance asterisks, we need to keep the numerical results with the `substring` command and then transform the string into numbers with the `real` function in `generate`.[10] After a few lines, we obtain a variable that contains coefficients (`coeff`) and another with standard errors (`sterror`) (and, optionally, the log likelihood if desired).

```
shell mktab -c estimate.out > estimates.txt

insheet using estimates.txt, comma clear
generate coef=substr(v2,1,7)
generate error=substr(v2,-6,5)
generate coeff=real(coef)
generate sterror=real(error)
replace sterror=sterror[_n+1]
drop if coeff==.
generate loglikelihood=coeff[_N] if _n==1
drop if _N==_n
keep coeff sterror loglikelihood
scalar ncoeff=_N
global nco=_N
save estimates.dta, replace
```

The exportation of the variance–covariance matrix is more cumbersome because, unfortunately, aML does not produce an exportable table with the covariances.[11] aML prints the matrix in the output file but does so in blocks of five columns each. In what follows, I exemplify how to deal with this problem for models that have between 10 and 15 parameters; however, the code needs to be adapted if the models have a smaller or greater number of parameters.

9. I would like to thank an anonymous referee for suggesting the exportation of results back to Stata.
10. By default, aML results with `mktab` are written with four digits after the decimal point. This can be changed using the `-n` option to increase or reduce precision. The creation of the substring with the numerical values needs to be changed accordingly.
11. Optionally, you may decide not to bring the variance–covariance matrix back to Stata and simply use the coefficients and the standard errors (as shown) given that some postestimation commands do not require covariances. When posting the variance–covariance matrix (see below), you can provisionally set covariances to 0.

First, we should `insheet` the aML output file into Stata (`estimate.out`) and drop the lines that do not contain the variance–covariance matrix:

```
insheet line using estimate.out, clear
drop if line==""
generate byte tmp=sum(line=="BHHH-based variance-covariance matrix:")
drop if tmp==0
replace tmp=sum(line=="------------------------------------------------------- ///
   ---------------")
keep if tmp==0
drop tmp
list
drop in 1/2
drop if _n==ncoeff+1
local j 10
forvalues i=18(2)26 {
    local j=`j´+1
    drop if _n==`i´ & ncoeff==`j´
}
drop if _n==_N
drop if _n==_N
```

Second, we need to create as many variables as the covariance matrix has columns. For the first five columns, it is very simple, but for the remainder, we need to relocate the covariances. For example, aML would print the following matrix for a model with six parameters:

varname	`var1`	`var2`	`var3`	`var4`	`var5`
`varname1`	var1[1]				
`varname2`	var1[2]	var2[2]			
`varname3`	var1[3]	var2[3]	var3[3]		
`varname4`	var1[4]	var2[4]	var3[4]	var4[4]	
`varname5`	var1[5]	var2[5]	var3[5]	var4[5]	var5[5]
`varname6`	var1[6]	var2[6]	var3[6]	var4[6]	var5[6]
`varname6`	var1[7]				

Thus we need to create a new variable (`var6`) that contains the variance of parameter 6, which is actually located in var1[7]. A couple of loops can help deal with this for all parameters:

```
generate x=_n
generate max=_N
generate str8 varname=word(line,1)
generate str12 var1=word(line,2)
generate str12 var2=word(line,3)
generate str12 var3=word(line,4)
generate str12 var4=word(line,5)
generate str12 var5=word(line,6)

forvalues i=6/$nco {
    generate var`i´=""
}

generate diff=max-ncoeff+10-ncoeff
generate diff2=max-ncoeff
```

```
local j 5
local s 0
while `j´<=9 {
    local j=`j´+1
    local s=`s´+1
    forvalues i=6/$nco {
        local p=`i´+diff
        replace var`j´=var`s´[`p´] in `i´
    }
}
local j 10
local s 0
while `j´<=ncoeff-1 {
    local j=`j´+1
    local s=`s´+1
    forvalues i=11/$nco {
        local p=`i´+diff2
        replace var`j´=var`s´[`p´] in `i´
    }
}
```

When all the covariances have been relocated to their correct spots, we drop the unnecessary lines and transform the matrix into numbers:

```
drop if _n>ncoeff
drop line x max
forvalues i=1(1)$nco {
    generate vce`i´=real(var`i´)
}
keep varname vce*
save vce.dta, replace
```

Next the easiest method is to merge the new variables that contain aML results into the dataset, and create a `bhat` matrix with the coefficients and a `vce` one with the variances and covariances. Note how the previously created macros are useful for naming the matrices columns and rows, distinguishing whether they belong to the first or second equation.

```
use dataset.dta, clear
merge using estimates.dta
drop _m
merge using vce.dta
drop _m

foreach var of varlist `level1´ `level2a´ {
    local equation1 "`equation1´ `var´_1"
}
foreach var of varlist `level1´ `level2b´ {
    local equation2 "`equation2´ `var´_2"
}
```

Now we simply need to create the coefficients matrix and the variance–covariance matrix from the variables with the `mkmat` command in Stata. In the case of `bhat`, the matrix needs to be transposed. Also the upper triangle of the variance–covariance matrix has to be completed.

```
mkmat coeff if coeff~=., matrix(bha)
mat bhat=bha´
mat colnames bhat = _cons_1 `equation1´ _cons_2 `equation2´ eps_1 eps_2 rho
mat list bhat

local covar ""
forvalues i=1(1)$nco {
   local covar="`covar´ vce`i´"
   }

mkmat `covar´ if varname~="", matrix(vce)
forvalues i=1/$nco {
    forvalues j=2/$nco {
        matrix vce[`i´,`j´]=vce[`j´,`i´]
    }
}
mat rownames vce =  _cons_1 `equation1´ _cons_2 `equation2´ eps_1 eps_2 rho
mat colnames vce =  _cons_1 `equation1´ _cons_2 `equation2´ eps_1 eps_2 rho
mat list vce, nohalf
```

Finally, all we have to do is save both matrices in the Stata system. We can do that with the `ereturn post` command. Now aML results become official estimates in Stata, thus enabling the use of postestimation features.

```
ereturn post bhat vce
ereturn display
```

3 Other applications

aML supports different multilevel multiprocess models. Thus the code I am presenting here can be adapted to different kinds of estimation. For example, let's imagine that instead of two random-effects probits, we need to jointly fit a random-effects linear model and a random-effects probit model. In such a case, we would only need to make a few changes. First, we would simply substitute `xtprobit` with `xtreg` in the regression that saves the starting values and parameters list. Moreover, there would be no need to make `_cons` of `lnsigmau` drop, so we would simply type an asterisk in front of the line `*drop if x==_N` (remember, this is the part of the `aml` file called `valuesx.dta`). Finally, when writing the file `estimation.dta`, we would substitute `"probit model;"` with `"continuous model;"` and adapt the error structure in accordance with our research question.[12]

4 Example

To illustrate how easy it is to move from Stata to aML (and back), let's run a simple (and naïve) example using a panel dataset in Stata (`nlswork.dta`) that contains information on a subsample of young women, aged 14 to 26 in 1968, drawn from the National Longitudinal Survey of Youth. Level 1 data are the individual (identified with `idcode`), and level 2 data are time (`year`). Imagine that we are interested in explaining the

12. The part of the code that brings aML results back to Stata needs to be modified accordingly as well.

probability of being married and the probability of belonging to a union. Moreover, we want to control for unobserved heterogeneity by integrating out a random effect in each equation and to allow these individual-specific effects to be freely correlated. Basically, we are presuming that there are certain unobserved personality traits that affect each outcome. Marriage (`msp`) and `union` are explained by the same covariates: year of birth (`birth_yr`) and `race` as time-invariant and `age` and the logarithm of wage (`ln_wage`) as time-varying. Furthermore, `union` is entered as an explanatory variable for marriage.

We apply the following code:

```
use http://www.stata-press.com/data/r12/nlswork

* time-invariant
local level1 ""
local level1 "`level1´ birth_yr"
local level1 "`level1´ race"

* time-variant
local level2a ""
local level2a "`level2a´ age"
local level2a "`level2a´ ln_wage"
local level2a "`level2a´ union"

* time-variant
local level2b ""
local level2b "`level2b´ age"
local level2b "`level2b´ ln_wage"

keep idcode year `level1´ `level2a´ `level2b´ msp union
label drop _all
foreach var of varlist `level1´ `level2a´ `level2b´ {
    drop if `var´==.
}
drop if msp==.
drop if union==.

save dataset.dta, replace

* ****************************************************************************
* Runs regressions and saves starting values and list of variables.
* ****************************************************************************

xtprobit msp `level1´ `level2a´, i(idcode) intpoints(4)
estimates store m1
estout m1 using sv1.txt, replace style(tab) order(_cons) mlabels("") collabels("")
generate eps1=`e(sigma_u)´
outfile eps1 using eps1.txt if _N==_n, replace

preserve
clear
insheet using sv1.txt
drop if v2==.
drop v2
drop if v1=="_cons"
format v1 %15s
save sv1.dta, replace
restore
```

```
xtprobit union `level1´ `level2b´, i(idcode) intpoints(4)
estimates store m2
estout m2 using sv2.txt, replace style(tab) order(_cons) mlabels("") collabels("")
generate eps2=`e(sigma_u)´
outfile eps2 using eps2.txt if _N==_n, replace

preserve
clear
insheet using sv2.txt
drop if v2==.
drop v2
drop if v1=="_cons"
format v1 %15s
save sv2.dta, replace
restore

* ****************************************************************************
* Outfiles data for aML use.
* ****************************************************************************

generate structure = 1
sort idcode year
outfile idcode structure `level1´ `level2a´ msp using data1.txt, wide comma replace

replace structure = 2
sort idcode year
outfile idcode structure `level1´ `level2b´ union using data2.txt, wide ///
   comma replace

keep idcode year
sort idcode year

outfile idcode using id.txt, wide comma replace

* ******************************************************************************
* Writes .r2a file that uploads data into aML.
* ******************************************************************************
clear
file open data using data.r2a, write replace
file write data "ascii data files = data1.txt" _n
file write data "                   data2.txt" _n
file write data ";" _n
file write data _n
file write data "id file = id.txt;" _n
file write data _n
file write data "level 1 var = " _n
file write data "`level1´" _n
file write data ";" _n
file write data _n
file write data "data structure = 1;" _n
file write data _n
file write data "level 2 var = " _n
file write data "`level2a´" _n
file write data "msp" _n
file write data ";" _n
file write data _n
file write data "data structure = 2;" _n
file write data _n
file write data "level 2 var = " _n
file write data "`level2b´" _n
file write data "union" _n
file write data ";" _n
file close data

* ****************************************************************************
* Defines in aML language the first two lines that set the explanatory variables.
* String is set to 50 characters. If willing to use longer names, change format.
* ****************************************************************************

clear
input str50 v1
"dsn = data.dat;"
"option variance-covariance matrix;"
"define regressor set Beta1;"
"var = 1"
end
save rsbeta1.dta, replace

clear
input str50 v1
"define regressor set Beta2;"
"var = 1"
end
save rsbeta2.dta, replace

clear
input str50 v1
";"
""
end
save semicolon.dta, replace

* ****************************************************************************
* Defines in aML language the middle part of the .aml file that sets the
* integration points, random effects, etc.
* ****************************************************************************

clear
input str50 v1
"define normal distribution;"
"dim=2;"
"number of integration points = 6;"
"name = eps1;"
"name = eps2;"
" "
"probit model;"
"data structure = 1;"
"outcome = msp;"
"model = regset Beta1 + intres(draw=1, ref=eps1) ;"
" "
"probit model;"
"data structure = 2;"
"outcome = union;"
"model = regset Beta2 + intres(draw=1, ref=eps2);"
" "
"starting values;"
end
save estimation.dta, replace
```

```
* ****************************************************************************
* The following defines the starting values list.
* I needed to create a final total string so that there is only one variable at
* the end that can be appended with the rest of the file.
* *****************************************************************************
forvalues i=1(1)2 {
    clear
    insheet using sv`i´.txt
    drop if v2==.
    generate str4 ft="FTTT"
    replace ft="TTTT" if v1=="_cons"
    order v1 ft v2
    tostring v2, generate(v2a) format(%10.6f) force
    drop v2
    rename v2a v2
    generate str4 blanc="          "
    generate str50 all=substr(v1,1,.)+substr(blanc,1,.)+substr(ft,1,.)+ ///
        substr(blanc,1,.)+substr(v2,1,.)
    drop v1 ft v2
    rename all v2
    * Next two lines make the _cons of lnsigmau to drop
    generate x=_n
    drop if x==_N
    drop x blanc
    save values`i´.dta, replace
}

* ***************************************************
* And, finally, random effects and rho´s.
* ***************************************************
forvalues i=1(1)2 {
    clear
    insheet using eps`i´.txt
    generate x=_n
    keep if x==1
    generate str10 blanc="          "
    generate str4 v2="eps`i´"
    generate str4 ft="FFTT"
    tostring v1, generate(v1a) format(%10.6f) force
    generate str50 all=substr(v2,1,.)+substr(blanc,1,.)+substr(ft,1,.)+ ///
       substr(blanc,1,.)+substr(v1a,1,.)
    keep all
    save eps`i´.dta, replace
}

clear
input str50 v1
"rho12 FFFT 0"
end
save rho.dta, replace

* *************************************************
* The final .aml file.
* *************************************************

use rsbeta1.dta, clear
append using sv1.dta
append using semicolon.dta
append using rsbeta2.dta
append using sv2.dta
append using semicolon.dta
append using estimation.dta
append using values1.dta
append using values2.dta
append using eps1.dta
append using eps2.dta
append using rho.dta
append using semicolon.dta

outfile using estimate.aml, replace noquote wide
* noquote specifies that strings not be outfiled in double quotes.

forvalues i=1(1)2 {
    erase sv`i´.txt
    erase eps`i´.txt
    erase rsbeta`i´.dta
    erase values`i´.dta
}
erase rho.dta
erase semicolon.dta
erase estimation.dta

clear

* Careful when adding more variables; lines may exceed 80 characters in .r2a.

shell raw2aml -r data.r2a
shell aml -r estimate.aml
shell update -r estimate.out


* ***********************************************************************
* (Optional) If desired, the following code brings the results back to Stata.
* ***********************************************************************

shell mktab -c estimate.out > estimates.txt

insheet using estimates.txt, comma clear
generate coef=substr(v2,1,7)
generate error=substr(v2,-6,5)
generate coeff=real(coef)
generate sterror=real(error)
replace sterror=sterror[_n+1]
drop if coeff==.
generate loglikelihood=coeff[_N] if _n==1
drop if _N==_n
keep coeff sterror loglikelihood
scalar ncoeff=_N
global nco=_N
save estimates.dta, replace

insheet line using estimate.out, clear
drop if line==""
generate byte tmp=sum(line=="BHHH-based variance-covariance matrix:")
drop if tmp==0
replace tmp=sum(line=="---------------------------------------------------- ///
   ------------------")
keep if tmp==0
drop tmp
list
```

```
drop in 1/2
drop if _n==ncoeff+1
local j 10
forvalues i=18(2)26 {
    local j=`j´+1
    drop if _n==`i´ & ncoeff==`j´
}
drop if _n==_N
drop if _n==_N
generate x=_n
generate max=_N
generate str8 varname=word(line,1)
generate str12 var1=word(line,2)
generate str12 var2=word(line,3)
generate str12 var3=word(line,4)
generate str12 var4=word(line,5)
generate str12 var5=word(line,6)

forvalues i=6/$nco {
    generate var`i´=""
}

generate diff=max-ncoeff+10-ncoeff
generate diff2=max-ncoeff

local j 5
local s 0
while `j´<=9 {
    local j=`j´+1
    local s=`s´+1
    forvalues i=6/$nco {
        local p=`i´+diff
        replace var`j´=var`s´[`p´] in `i´
    }
}
local j 10
local s 0
while `j´<=ncoeff-1 {
    local j=`j´+1
    local s=`s´+1
    forvalues i=11/$nco {
        local p=`i´+diff2
        replace var`j´=var`s´[`p´] in `i´
    }
}

drop if _n>ncoeff
drop line x max
forvalues i=1(1)$nco{
    generate vce`i´=real(var`i´)
}
keep varname vce*
save vce.dta, replace

use dataset.dta,clear
merge using estimates.dta
drop _m
merge using vce.dta
drop _m
```

```
foreach var of varlist `level1´ `level2a´ {
    local equation1 "`equation1´ `var´_1"
}
foreach var of varlist `level1´ `level2b´ {
    local equation2 "`equation2´ `var´_2"
}

mkmat coeff if coeff~=., matrix(bha)
mat bhat=bha´
mat colnames bhat = _cons_1 `equation1´ _cons_2 `equation2´ eps_1 eps_2 rho
mat list bhat

local covar ""
forvalues i=1(1)$nco {
   local covar="`covar´ vce`i´"
}

mkmat `covar´ if varname~="", matrix(vce)
forvalues i=1/$nco {
    forvalues j=2/$nco {
        matrix vce[`i´,`j´]=vce[`j´,`i´]
    }
}
mat rownames vce =  _cons_1 `equation1´ _cons_2 `equation2´ eps_1 eps_2 rho
mat colnames vce =  _cons_1 `equation1´ _cons_2 `equation2´ eps_1 eps_2 rho
mat list vce, nohalf

ereturn post bhat vce
ereturn display
```

After the process, these are the aML results in Stata:

```
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     _cons_1 |     2.3125   .3274141     7.06   0.000      1.67078     2.95422
  birth_yr_1 |      -.031   .0062929    -4.93   0.000    -.0433338   -.0186662
      race_1 |     -.8534   .0383275   -22.27   0.000    -.9285206   -.7782794
       age_1 |      .0345   .0016174    21.33   0.000     .0313299    .0376701
   ln_wage_1 |     -.1564   .0330606    -4.73   0.000    -.2211975   -.0916025
     union_1 |     -.0891   .0344093    -2.59   0.010     -.156541    -.021659
     _cons_2 |    -2.3221   .3689173    -6.29   0.000    -3.045165   -1.599035
  birth_yr_2 |     -.0245   .0070922    -3.45   0.001    -.0384006   -.0105994
      race_2 |      .5751   .0416893    13.79   0.000     .4933904    .6568096
       age_2 |     -.0083   .0021321    -3.89   0.000    -.0124789   -.0041211
   ln_wage_2 |      .9753   .0356511    27.36   0.000     .9054252    1.045175
       eps_1 |     1.4076    .022059    63.81   0.000     1.364365    1.450835
       eps_2 |     1.3215   .0271219    48.72   0.000     1.268342    1.374658
         rho |      .0216     .01728     1.25   0.211    -.0122683    .0554683
------------------------------------------------------------------------------
```

5 Acknowledgments

I would like to thank Stan Panis for answering my questions about aML and helping me to bring aML results back to Stata. I am also grateful to Stephen Jenkins for his comments. Financial support from the Spanish project ECO2010-21668-C03-02 and XREPP (Direcció General de Recerca) is greatly acknowledged.

6 References

Gini, R., and J. Pasquini. 2006. Automatic generation of documents. *Stata Journal* 6: 22–39.

Greenwood, D. 2006. WinBugs & Stata. http://www.personal.leeds.ac.uk/~hssdg/Stata.

Jann, B. 2005. Making regression tables from stored estimates. *Stata Journal* 5: 288–308.

———. 2007. Making regression tables simplified. *Stata Journal* 7: 227–244.

Jones, R. 2010. runmplus: Stata module to run Mplus from Stata. Statistical Software Components S457154, Department of Economics, Boston College. http://ideas.repec.org/c/boc/bocode/s457154.html.

Leckie, G., and C. Charlton. 2013. runmlwin—a program to run the MLwiN multilevel modelling software from within Stata. *Journal of Statistical Software* 52: 1–40.

Lillard, L. A., and C. W. A. Panis. 2003. *aML Multilevel Multiprocess Statistical Software, Version 2.0.* Los Angeles, CA: EconWare.

Rabe-Hesketh, S., A. Skrondal, and A. Pickles. 2004. GLLAMM manual. Working Paper 160, Division of Biostatistics, University of California–Berkeley. http://www.bepress.com/ucbbiostat/paper160/.

Reardon, S. F. 2006. hlm: Stata module to invoke and run HLM v6 software from within Stata. Statistical Software Components S448001, Department of Economics, Boston College. http://econpapers.repec.org/software/bocbocode/s448001.htm.

Roodman, D. M. 2011. Fitting fully observed recursive mixed-process models with cmp. *Stata Journal* 11: 159–206.

About the author

Sara Ayllón has a PhD in applied economics and works on the analysis of poverty dynamics and their relationship with labor and demographic events. She is an assistant professor in the Department of Economics at the University of Girona in Spain.