How to use the Anderson-Moore Algorithm with EViews and R

Aneesh Raghunandan and Andrew Giffin

November 30, 2010

1 Using EViews

1.1 Preliminaries

You will need the "R and Friends" software, found here.[1] This is the software that uses Microsoft's COM technology to connect EViews to R.[2] This R package has not been tested on R versions lower than 2.11.0. Consequently, it may be necessary to use an R that is version 2.11.0 or later.

Note: you need to have the 'RJava' package installed. This comes automatically with almost all R distributions, however, and so you almost certainly already have it.

The EViews commands you will need to use from the EViews command line are `xopen, xclose, xrun, xput,` and `xget`.The first opens the connection to R; the second closes this connection; the third runs a command in that connection; the fourth places an object into R; and the fifth brings an existing object in the R workspace into EViews.

1.2 Installation

To install or update the **AMA** package, start R and click on Packages→Load package.... to get a list of packages to choose from. Then, click on AMA to install the package.[3] If you do not see it on the list or if you prefer using the R command line to install the package enter:

To install or update the **AMA** package, start R and enter:

```
options(repos=c(CRAN = "ftp://cran.r-project.org/pub/R"))
install.packages("AMA")
```

[1]For Board computers, you will need to ask ARC to install this software.

[2]This technology is also used by the package R.matlab

[3]The package may work for older versions of R, but it will not likely be available as a choice in the Load packages... menu. Using the command line installation alternative will allow you to try to load the package.

One need not repeat these commands in subsequent R session unless a package update is desired.

To use the installed **AMA** package enter:
`library(AMA)`

1.3 Model Processing

Currently, there are two ways to get a model into a form usable by AMA:

1. Structural matrices (the H matrices) already in EViews
2. Model in MODELEZ syntax in separate file (for details of MODELEZ syntax, see below)

In the first case, the procedure is straightforward and will be explained below. In the second case, you will be able to import the structural matrices to EViews (if you like); regardless, you can execute all commands from within EViews.

1.3.1 The Structural Coefficients Matrix H Needed by the Algorithm

The Anderson-Moore Algorithm defines and uses a matrix of structural coefficients for each lag and lead (and present state) in the model. If the model has τ lags and θ leads, then we have the matrices $H_{-\tau}, ..., H_0, ..., H_\theta$. The "structural coefficients" matrix needed by the algorithm is simply the block matrix $[H_{-\tau} \cdots H_0 \cdots H_\theta]$. These can be computed by providing the model equations and parameters in MODELEZ syntax.

1.3.2 MODELEZ Syntax

If you want to use this format, save the model in a file using the methods and syntax described below. The example in the last section includes a sample MODELEZ file. The first line of the file should be the model name, written as follows

- MODEL > NAMEOFMODEL

Next, list the variables (note: in the AMA formulation, all variables are considered endogenous) as follows:

- ENDOG >
  variable1
  variable2
  ⋮

After this, you'll need to list the equations. Write each equation as follows:

- EQUATION > NAMEOFEQUATION
  EQ > (model equation–see below)

The model equations can be written in two general forms. The first is *left hand side expression = right hand side expression*; the second can just list an expression, with no equals sign, which will be assumed equal to zero. Where leads and lags are appropriate, use LEAD(variablename, numberOfLead) or LAG(variablename, numberOfLag). For example, if your model includes the equation $Y_t = \delta Y_{t-2} + \beta IS_{t+3} + K$ you would write the equation in MODELEZ in one of the following two ways (assuming DELTA corresponds to δ and BETA corresponds to β):

Y = DELTA*LAG(Y, 2) + BETA*LEAD(IS, 3) + K OR
Y - DELTA*LAG(Y, 2) - BETA*LEAD(IS, 3) - K

After inputting all variables and equations into your .mod file, the last line of the file should just be "END" (without the quotation marks, and in all CAPS).

The example MODELEZ file will hopefully make this clearer.

1.4 Computing using R

For details about the arguments to each function, please see the "AMA-Manual". It is written in the R documentation standard. For each function you will find a list of inputs as well as a description of each input, the function's output, and in some cases a short example of how to use the function.
Note: for an easy-to-use example, please see section 2 - **Examples** (page 5.)

Before carrying out any computations, you need to open the connection to R and load the AMA library. Run the following:

- `xopen(R)` // (if you haven't already)
- `xrun "library(AMA)"` // this loads the AMA package

1.4.1 Getting the H matrix into R

If you have the structural coefficients matrix H already in the EViews workfile, then to place this matrix into **R** you would do the following:

- Run the command `xput(hmat)` (where `hmat` is the name I've chosen to give the matrix H–you can replace this with whatever name you'd like)

If you do not have the structural coefficients matrix H already in the EViews workfile, but you have it in a MODELEZ file (say, myModel.mod) then to get the H matrix into R you would do the following, utilizing the AMA package's function genHmat:

- If you have the model parameters stored in a text file (say, myParams.txt), then run the command `xrun 'hmat <- genHmat("myModel.mod", "myParams.txt")'`.
- If you instead have the model parameters stored in an EViews vector object, say "myParamVec", then you should first run `xput myParamVec` and then run the command `xrun 'hmat <- genHmat("myModel.mod", myParamVec)'`.
- If you have the model parameters stored in a text file (say, myParams.txt) AND would like to obtain not only the matrix H but the parameters in a vector as well (in order to easily update them in EViews or R), the procedure is slightly more complicated. Instead of the previous two options, you would run the following commands (after opening the connection to R):
  - `xrun 'hlist <- genHmat("myModel.mod", "myParams.txt", wantParamVec = TRUE)'`
  - Extract the H matrix using the command `xrun 'hmat<- hlist[1][[1]])'`, and the parameter vector using the command `xrun 'myParamVec <- hlist[2][[1]]'`

NOTE: Wherever you see `"myModel.mod"` or `"myParams.txt"` above, replace these with the full path to the respective files (for example, `"C:/Users/Aneesh/Documents/AMA/myModel.mod"`). Remember to use the quotation marks! Otherwise, R won't recognize this as a string object.

1.4.2 Next Steps

You should now have the H matrix in R and the R package loaded.

You can now compute any of the matrices associated with AMA. You will need to tell these functions how many equations, lags, and leads are in the model. To do so:

- `xrun 'neq = <number of equations>'`
- `xrun 'leads = <number of leads in the model>'`
- `xrun 'lags = <number of lags in the model>'`

To compute the matrices you desire, please see AMA-Manual.pdf for a list of functions, their arguments, descriptions of their arguments, and descriptions of their outputs. Just xrun the function you desire. For example, if you wanted to compute the reduced-form coefficients matrix B, you would type (in EViews)

- `xrun 'bmat <- genBmat(hmat, neq, leads, lags)'`

This creates an object in the R workspace called `bmat`. To bring it back into EViews, you would run `xget bmat`; this places it in the current EViews workfile as a matrix called `bmat`. (Note: this means that you should assign distinct names to objects!

EViews will often complain if you try to replace an object with something else of the same name).

Note that you can generate the matrices B and Q directly via a call to the function `callAMA`. However, you'll get them back as vectors (which correspond to matrices stored columnwise) which may be an unnecessary hassle if you're trying to bring them back into EViews. To get around this, use the `genBmat, genQmat,` etc. functions directly. You don't need to use `callAMA` first–these functions will call AMA.
Obviously for functions such as `genScof` you'll have to run `genBmat` first (since `genScof` takes the matrix B as an input). Be aware of which matrices depend on which.

1.5 Using R as a standalone on Windows or Linux

The idea here is extremely similar to the EViews examples above. Just remove the `xput, xget`, and `xrun` statements. Although EViews does not run on Linux, you can still install and use the AMA R package on Linux.

2 Examples

We walk through a simple example from EViews, using a MODELEZ file. In the "Examples" folder, the model in MODELEZ syntax is "example7.mod", and the parameter file is "example7params.prm". From the EViews side, we run the following commands with an explanation of what they do:
Note: The contents of both these files are listed in the **Appendix.**

open R `xopen(r)`

loads the AMA (and rJava) library `xrun "library(AMA)"`

display function documentation `xrun "print(help(callAMA))"`

get local filename for package example model

```
xrun "modName<-system.file('extdata/example7.mod',package='AMA')"
```

get local filename for package example parameters

```
xrun "paramName<-system.file('extdata/example7params.prm',package='AMA')"
```

create H matrix `xrun 'hmat <- genHmat("modName", "paramName")'`

bring H matrix into EViews `xget 'hmat'`

create reduced-form coeff. matrix `xrun 'bmat <- genBmat(hmat, 4, 1, 1)'`

import reduced-form coeff. matrix into EViews `xget 'bmat'`

create observable structure matrix
`xrun 'scof <- genScof(hmat, bmat, 4, 1, 1)'`

import observable structure matrix into EViews `xget 'scof'`

make ϕ, F `xrun 'factMats <- getFactorMatrices(hmat, bmat, 4, 1, 1)'`

get the matrix ϕ explicitly `xrun 'phiMat <- factMats[1][[1]]'`

get the matrix F `xrun 'fMat <- factMats[2][[1]]'`

imports matrix ϕ into EViews `xget 'phiMat'`

imports matrix F back into EViews `xget 'fMat'`

create stochastic transition matrices
`xrun 'stochMats <-getStochTrans(hmat, scof)'`

get the matrix $\mathcal{A}$ `xrun 'scriptA <- stochMats[1][[1]]'`

get the matrix $\mathcal{B}$ `xrun 'scriptB <- stochMats[2][[1]]'`

imports matrix $\mathcal{A}$ into EViews `xget 'scriptA'`

imports matrix $\mathcal{B}$ into EViews `xget 'scriptB'`

closes the connection to R `xclose(r)`

Note that if you reopen the connection to R, all variables must be redefined. Make sure to use `xget` on all the variables you want before closing R, and that you've carried out all the computations you want R to do.

If the matrix H is already in EViews, then instead of using the `genHmat` function in the example above, just use `xput(hmat)` instead. The rest is unchanged.

3 Appendix

3.1 example7.mod

```
MODEL > EXAMPLE7

ENDOG>

          y    _NOTD
        inf    _NOTD
         dr    _NOTD
```

```
         rs   _NOTD

EQUATION >  is
EQTYPE >    IMPOSED
EQ >        y = LEAD(y,1) - (1/sigma)*(rs-LEAD(inf,1))


EQUATION >  phillips
EQTYPE >    IMPOSED
EQ >        inf = delta*LEAD(inf,1) + lambda*y

EQUATION >  dr
EQTYPE >    IMPOSED
EQ >        dr = rs - LAG(rs,1)

EQUATION >  policy
EQTYPE >    IMPOSED
EQ >        rs  =  rho*LAG(rs,1) + gampi*inf

END
```

3.2 example7params.prm

```
beta=1
delta=.99
lambda=.3
rho=0.66
gampi = 1.1
sigma=1
```