How to use the Anderson-Moore R Implementation

Aneesh Raghunandan and Andrew Giffin*

August 6, 2010

1 Preliminaries

The package was written using R 2.11.1 and is available from the Comprehensive R Archive Network (CRAN). 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.

1.1 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.[1] If you do not see it on the list or if you prefer using the R command line to install the package enter:

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

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)

*Updated by Gary S. Anderson February 21, 2011.

[1]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.

1.2 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.

1.3 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 dependent variable = ...; 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.pdf" file. 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.

Before carrying out any computations, you will need load the AMA library.

library(AMA) // this loads the AMA package

If you have the structural coefficients matrix H 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

hmat <- genHmat("myModel.mod", "myParams.txt")

.

If you instead have the model parameters stored in an EViews vector object, say "myParamVec", and then run the command

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):

hlist <- genHmat("myModel.mod", "myParams.txt", wantParamVec = TRUE)

Extract the H matrix using the command 'hmat<- hlist[1][[1]]', and the parameter vector using the command

'myParamVec <- hlist[2][[1]]'

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.

1.4.1 Next Steps

You should now have the H matrix into 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:

- neq = <number of equations>
- leads = <number of leads in the model>
- 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 description of their outputs. Just run the function you desire. For example, if you wanted to compute the reduced-form coefficients matrix B, you would type

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 **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 create 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 call AMA already. 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.

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", and the parameter file is "example7params.txt". From the EViews side, we run the following commands with an explanation of what they do:

loads the AMA (and rJava) library library(AMA)

display function documentation help(callAMA)

get local filename for package example model
modName<-system.file("extdata/example7.mod",package="AMA")

get local filename for package example parameters
paramName<-system.file("extdata/example7params.prm",package="AMA")

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

bring H matrix into EViews hmat

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

import reduced-form coeff. matrix into EViews bmat

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

import observable structure matrix into EViews scof

make ϕ, F factMats <- getFactorMatrices(hmat, bmat, neq, nlead, nlag)

get the matrix ϕ **explicitly** phiMat <- factMats[1][[1]]

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

imports matrix ϕ **into EViews** phiMat

imports matrix F **back into EViews** fMat

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

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

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

imports matrix $\mathcal{A}$ **into EViews** scriptA

imports matrix $\mathcal{B}$ **into EViews** scriptB

closes the connection to R xclose(r)