AgEcon SEARCH
RESEARCH IN AGRICULTURAL & APPLIED ECONOMICS

The World's Largest Open Access Agricultural & Applied Economics Digital Library

This document is discoverable and free to researchers across the globe due to the work of AgEcon Search.

Help ensure our sustainability.

Give to AgEcon Search

AgEcon Search
<http://ageconsearch.umn.edu>
aesearch@umn.edu

*Papers downloaded from **AgEcon Search** may be used for non-commercial purposes and personal study only. No other use, including posting to another Internet site, is permitted without permission from the copyright owner (not AgEcon Search), or as allowed under the provisions of Fair Use, U.S. Copyright Act, Title 17 U.S.C.*

THE STATA JOURNAL

Editor

H. Joseph Newton
Department of Statistics
Texas A & M University
College Station, Texas 77843
979-845-3142; FAX 979-845-3144
jnewton@stata-journal.com

Associate Editors

Christopher Baum
Boston College
Rino Bellocco
Karolinska Institutet, Sweden and
Univ. degli Studi di Milano-Bicocca, Italy
A. Colin Cameron
University of California–Davis
David Clayton
Cambridge Inst. for Medical Research
Mario A. Cleves
Univ. of Arkansas for Medical Sciences
William D. Dupont
Vanderbilt University
Charles Franklin
University of Wisconsin–Madison
Joanne M. Garrett
University of North Carolina
Allan Gregory
Queen's University
James Hardin
University of South Carolina
Ben Jann
ETH Zurich, Switzerland
Stephen Jenkins
University of Essex
Ulrich Kohler
WZB, Berlin

Stata Press Production Manager

Stata Press Copy Editor

Editor

Nicholas J. Cox
Geography Department
Durham University
South Road
Durham City DH1 3LE UK
n.j.cox@stata-journal.com

Jens Lauritsen
Odense University Hospital
Stanley Lemeshow
Ohio State University
J. Scott Long
Indiana University
Thomas Lumley
University of Washington–Seattle
Roger Newson
Imperial College, London
Marcello Pagano
Harvard School of Public Health
Sophia Rabe-Hesketh
University of California–Berkeley
J. Patrick Royston
MRC Clinical Trials Unit, London
Philip Ryan
University of Adelaide
Mark E. Schaffer
Heriot-Watt University, Edinburgh
Jeroen Weesie
Utrecht University
Nicholas J. G. Winter
University of Virginia
Jeffrey Wooldridge
Michigan State University
Lisa Gilmore
Gabe Waggoner

Copyright Statement: The Stata Journal and the contents of the supporting files (programs, datasets, and help files) are copyright © by StataCorp LP. The contents of the supporting files (programs, datasets, and help files) may be copied or reproduced by any means whatsoever, in whole or in part, as long as any copy or reproduction includes attribution to both (1) the author and (2) the Stata Journal.

The articles appearing in the Stata Journal may be copied or reproduced as printed copies, in whole or in part, as long as any copy or reproduction includes attribution to both (1) the author and (2) the Stata Journal.

Written permission must be obtained from StataCorp if you wish to make electronic copies of the insertions. This precludes placing electronic copies of the Stata Journal, in whole or in part, on publicly accessible web sites, file servers, or other locations where the copy may be accessed by anyone other than the subscriber.

Users of any of the software, ideas, data, or other materials published in the Stata Journal or the supporting files understand that such use is made without warranty of any kind, by either the Stata Journal, the author, or StataCorp. In particular, there is no warranty of fitness of purpose or merchantability, nor for special, incidental, or consequential damages such as loss of profits. The purpose of the Stata Journal is to promote free communication among Stata users.

The *Stata Journal*, electronic version (ISSN 1536-8734) is a publication of Stata Press, and Stata is a registered trademark of StataCorp LP.

Mata Matters: Interactive use

William Gould
StataCorp
College Station, TX
wgould@stata.com

Abstract. Mata is Stata’s matrix language. In the Mata Matters column, we show how Mata can be used interactively to solve problems and as a programming language to add new features to Stata. In this quarter’s column, we look at interactive use of Mata.

Keywords: pr0024, Mata, interactive use

Introduction

Mata is a programming language, and because of that it is easy to overlook how easy it is to use Mata interactively. To enter Mata, all you have to do is type

```
. mata  
:  
_____ mata (type end to exit) _____
```

To exit back to Stata, type

```
: end  
_____  
.
```

In between, the syntax is different from Stata, but Mata behaves the same way. You might make a mistake,

```
: display exp(1i*pi())  
invalid expression  
r(3000);
```

but Mata doesn’t care, so you can correct it immediately:

```
: exp(1i*pi())  
-1
```

Mata is so much like Stata that you can even use the *PgUp* key to retrieve and then edit your previous command. The error above was typing `display`. Mata automatically displays results.

When working interactively, pay attention to the prompt. When the prompt on the left is a colon, you are talking to Mata. Type something that Mata understands, or type `end`. When the prompt on the left is a period, you are talking to Stata. Type something that Stata understands, or type `mata`.

Obviously, you could use Mata interactively to teach. That's a wonderful use. What you may not realize, however, is that you can use Mata interactively in practical applications, and in fact, you can use Mata interactively to *avoid* writing programs and ado-files. That is what we are going to do here: we are going to use Mata to *not* write programs.

When you want to enter Mata interactively, type “mata” and press *Enter*. That is different from when you use Mata in programs and ado-files, when you should type “mata:” (with the colon). When you enter by typing “mata:” and then make a mistake, Mata stops and exits back to Stata. When you enter by typing “mata” without the colon and then make a mistake, Mata just issues an error message and continues.

Interactive use

I have the following data:

```
. tabulate cat
```

cat	Freq.	Percent	Cum.
1	60	62.50	62.50
2	23	23.96	86.46
3	6	6.25	92.71
4	7	7.29	100.00
Total	96	100.00	

Along with these data, I have theory that says that half the data should be in category 1, half the remainder in category 2, half again in 3, and the rest in 4. That is, the expected counts in the cells are 48, 24, 12, and 12.

Can I reject at the 5% level that these data are from the distribution suggested by theory?

The chi-squared test is easy enough to perform, but look around and you will find nothing in official Stata that will answer that question. That's absurd, but true. Look around more and you'll find user-written routines. Nick Cox has written one.

The chi-squared statistic is simple enough; it is

$$\chi^2(3) = \sum_{i=1}^4 \frac{(o_i - e_i)^2}{e_i}$$

which here is $(60 - 48)^2/48 + (23 - 24)^2/24 + \dots$

In Mata, we could calculate this by typing

```
. mata
----- mata (type end to exit) -----
: o = (60\ 23\ 6\ 7)
: e = (48\ 24\ 12\ 12)
```

```

: sum( (o-e)^2 :/ e )
      8.125
: chi2tail(3, 8.125)
      .0434977514
: end

```

That's a pretty easy solution. The worst part of it was entering the data, but we can solve that:

```

. tabulate cat, matcell(obs)
  (output omitted)
. mata
----- mata (type end to exit) -----
: o = st_matrix("obs")
: e = (48\ 24\ 12\ 12)
: sum( (o-e)^2 :/ e )
      8.125
: chi2tail(3, 8.125)
      .0434977514
: end

```

The `matcell()` option of Stata's `tabulate` command saves the frequencies in a Stata matrix. Mata function `st_matrix()` transfers Stata matrices to Mata matrices. Think of Mata as being separate from Stata but with functions that will allow Mata to access, manipulate, and change Stata objects. Do not expect to be able to use Stata objects directly; there's always an intervening import function.

In the solution above, we still had to enter the expected counts. Rather than counting on our fingers and typing

```

: e = (48\ 24\ 12\ 12)

```

we could do the following,

```

: N = sum(o)
: e = (N/2 \ N/4 \ N/8 \ N-N/2-N/4-N/8)

```

or

```

: N = sum(o)
: e = (N/2 \ N/4 \ N/8)
: e = e \ N-sum(e)

```

and then our entire solution would be

```

. tabulate cat, matcell(obs)
  (output omitted)
. mata
----- mata (type end to exit) -----
: o = st_matrix("obs")
: N = sum(o)
: e = (N/2 \ N/4 \ N/8)

```

```

: e = e \ N-sum(e)
: sum( (o-e)^2 :/ e )
      8.125
: chi2tail(3, 8.125)
      .0434977514
: end

```

Obtaining help interactively

To see the online help for a Mata function, type “**help mata**” followed by the function name, followed by “**()**”:

```

. help mata sum()
  (help appears)
. help mata st_matrix()
  (help appears)

```

Don’t forget the “**mata**”. Type “**help st_matrix()**” and you’ll get a help-not-found error.

You can type **help** at the Stata dot prompt, as above, or at the Mata colon prompt:

```

: help mata sum()
  (help appears)
: help mata st_matrix()
  (help appears)

```

Either way, you still have to type the “**mata**”.

Mata is fundamentally simple. The difficulty is that there are hundreds of Mata functions. Although the functions are logically named (if I tell you a name, you can often guess what the function does), the name may not occur to you given the description. You need to learn how to sniff out functions.

When I work interactively with Mata, the first thing I do is type

```

. help mata

```

I then go to the Viewer window and click on **[M-4] Index and guide to functions**. As I work interactively, I use the Viewer window to help me locate the function I need. The functions are categorized in the **[M-4] Index and guide to functions** help file:

<i>Mathematical</i>	
matrix	Matrix functions
solvers	Matrix solvers and inverters
scalar	Scalar functions
statistical	Statistical functions
mathematical	Other important functions
<i>Utility & manipulation</i>	
standard	Standard matrices
utility	Matrix utility functions
manipulation	Matrix manipulation functions
<i>Stata interface</i>	
stata	Stata interface functions
<i>String, I/O, & programming</i>	
string	String functions
io	I/O functions
programming	Programming functions

The categories are clickable. Getting familiar with the categorization is helpful, so spend some time clicking around.

Accessing data and saved results

If you are going to work with Mata interactively, you will need to access Stata results. Think of Mata as being separate from Stata. You import results from Stata into Mata and sometimes export them back again. Mata has functions to do that:

Purpose	Function
Import data	st_data() and st_view()
Export data	st_store()
Import/export scalars	st_numscalar()
Import/export matrices	st_matrix()
Import/export macros	st_global() and st_local()

Accessing data

If I need to import data, I usually use **st_data()**. I admit that **st_view()** is better in that it conserves memory, but when working interactively, I do not bother unless forced. Conserving memory is something I worry about when programming, not when working interactively.

I usually work in Stata to eliminate the irrelevant observations, including those with missing values. Then I can type

```
: y = st_data(., "varname")
: X = st_data(., ("varname1", "varname2", ...))
```

I don't like having to type each variable name separately, bound in quotes, so I type

```
. X = st_data(., tokens("varname1 varname2 ..."))
```

Even better, I have written a Stata program to make accessing the data easier. You can use it, too. To install it, type

```
. ssc install tomata
```

After installation, you can type `help tomata` to learn about it. Basically, if you type

```
. tomata
```

the program will instantly create column vectors in Mata with the same name and data as the variables in the dataset. And although I do not worry about using views rather than copies when I work interactively, the programs I write do. Each column vector will be a view and so consume almost no memory.

Let me show you how this process works:

```
. sysuse auto, clear
(1978 Automobile Data)
. tomata mpg weight foreign rep78, nomissing
. mata
----- mata (type end to exit) -----
: mata describe
      # bytes   type                name and extent
-----
      280    real colvector        foreign[69]
      280    real colvector        mpg[69]
      280    real colvector        rep78[69]
      280    real colvector        weight[69]
-----
: end
```

I typed `tomata mpg weight foreign rep78, nomissing`. That command specified that I wanted four column vectors created in Mata: one corresponding to `mpg`, another to `weight`, a third to `foreign`, and a fourth to `rep78`. I also specified `tomata`'s `nomissing` option, saying I wanted casewise exclusion of missing values. The automobile dataset contains 74 observations, but because I specified `nomissing`, and because `rep78` happens to have 5 observations that contain missing, all the vectors created excluded those 5 observations. The data were not changed; the observations were merely excluded from the vectors. Remember that I said I work in Stata first to exclude the missing values? `tomata` saves me from having to do that.

Say that I were teaching. I could now demonstrate how linear-regression coefficients are calculated on that sample:


```

. mata
----- mata (type end to exit) -----
: y = mpg
: X = (weight, foreign, J(rows(weight), 1, 1))
: b = invsym(X'X)*X'y
: b
      1
1   -.0065057732
2   -1.444831783
3   41.45527944
: end

```

If I were teaching, I would make a big deal about the `J(rows(weight), 1, 1)`, explaining that it created a column vector of ones, and explaining the purpose. I would next demonstrate that I would obtain the same results if I deviated all values from their respective means and excluded the constant vector:

```

. mata
----- mata (type end to exit) -----
: y = mpg :- mean(mpg)
: X = (weight, foreign)
: X = X :- mean(X)
: b = invsym(X'X)*X'y
: b
      1
1   -.0065057732
2   -1.444831783
: end

```

Accessing saved results

There are three functions useful for fetching results from `r()` and `e()`:

```

st_numscalar(name)
st_matrix(name)
st_global(name)

```

Remember to type the name in quotes:

```

: N = st_numscalar("e(N)")
: b = st_matrix("e(b)")

```

The best way to find out what a Stata command stores in `r()` or `e()` is to type `return list` or `ereturn list` after running the command.

When I work interactively with Mata, I seldom need to consume string results, so I seldom use `st_global()`. I will come upon an occasional ado-file, however, that saves a number in a macro. Then I need to type

```
: k = strtoreal(st_global("k"))
```

Invariably I forget the `strtoreal()` part,

```
: k = st_global("k")
```

and then I have to type

```
: k = strtoreal(k)
```

later.

Simulation

Stata has a reputation of being good for simulations, but Mata is even better because looping is so natural for Mata and Mata is so fast. We are going to use Mata interactively not to obtain results from Stata and process them, but to create results (a dataset of simulation results) and use Stata to process them.

Radiologists A and B each evaluated 100 X-rays. Radiologist A evaluated 76 of them positively, and Radiologist B, 72. They agreed in 72% of their evaluations. The results were as follows:

```
. kap A B
```

Agreement	Expected Agreement	Kappa	Std. Err.	Z	Prob>Z
72.00%	61.44%	0.2739	0.0995	2.75	0.0029

Wait a minute, you say to yourself, can these results be correct? Radiologist A evaluated 76% positively, B, 72%, and they agreed on only 72%? Given each evaluated 70+ percent of the X-rays positively, they just had to agree by some large amount. Given those margins, `kap`, in fact, reports 61.44% expected agreement due merely to randomness. You may believe that, but is the distribution really so tight that a mere 72% agreement is enough to reject, at the 0.29% level, that the agreement is due to chance?

If you said that to yourself, it is time for a simulation. Its results will be (1) you will convince yourself that `kap` is right, (2) you will convince yourself that `kap` is wrong, or (3) you will convince yourself that `kap` has a hidden assumption that you do not like. Whatever the result, you will have a deeper understanding of what it means to “agree”.

So let’s set up a loop where A rates exactly 76 of 100 positively but randomly, and B rates exactly 72 of 100 positively but randomly. We’ll do this 5,000 times and record the number of “agreements”. We’ll make a 5,000-observation Stata dataset out of the results, which we can then examine in detail. For instance, it would be interesting to look at a histogram or to look at `summarize`, `detail` output.

Here's the solution:

```
. clear
. mata
----- mata (type end to exit) -----
: A = J(76, 1, 1) \ J(100-76, 1, 0)
: B = J(72, 1, 1) \ J(100-72, 1, 0)
: result = J(5000, 1, .)
: uniformseed(39299)
: for (rep=1; rep<=5000; rep++) {
>     _jumble(A)
>     _jumble(B)
>     result[rep] = sum(A == B)
> }
: st_addobs(5000)
: st_addvar("float", "agree")
: st_store(., "agree", result)
: end
-----

. replace agree = agree/100
(5000 real changes made)

. summarize, detail
```

agree				
	Percentiles	Smallest		
1%	.52	.5		
5%	.56	.5		
10%	.56	.5	Obs	5000
25%	.58	.5	Sum of Wgt.	5000
50%	.62		Mean	.614232
		Largest	Std. Dev.	.0389317
75%	.64	.74		
90%	.66	.74	Variance	.0015157
95%	.68	.76	Skewness	.0908267
99%	.7	.76	Kurtosis	2.956061

We ran 5,000 replications and, on my computer, that ran in less than 1 second. My computer is not particularly fast.

Results are pretty much as **kap** reported: 61.44% expected agreement, and in our simulation, average agreement was 61.42%. We see that, given the 70+ percent positive evaluations, agreement could not have fallen below 50%. The 95th percentile is .68, so if there is 68% or more agreement, we can reject the null hypothesis at the 5% level (one-tailed, because we never suspected that A and B's judgments would be negatively correlated). We can reject the null at the 1% level if there is 70% or more agreement.

There are other interesting things you can explore for yourself about these results: type **histogram agree**.

In the simulation above, we forced A and B to rate exactly 76% and 72% of the X-rays positively. What would happen, you might wonder, if we relaxed that and instead merely *expected* that they would rate 76% and 72% positively?

Let's modify our simulation and rerun it:

```
. clear
. mata
----- mata (type end to exit) -----
: result = J(5000, 1, .)
: uniformseed(39240)
: for (rep=1; rep<=5000; rep++) {
>     a = uniform(100,1) :>= (1-.76)
>     b = uniform(100,1) :>= (1-.72)
>     result[rep] = sum(a==b)
> }
: st_addobs(5000)
: st_addvar("float", "agree")
  1
: st_store(., "agree", result)
: end
-----

. replace agree = agree/100
(5000 real changes made)

. summarize, detail
```

		agree			
Percentiles		Smallest			
1%	.495	.43			
5%	.53	.46			
10%	.55	.46	Obs	5000	
25%	.58	.46	Sum of Wgt.	5000	
50%	.61		Mean	.614434	
		Largest	Std. Dev.	.0495137	
75%	.65	.79			
90%	.68	.79	Variance	.0024516	
95%	.695	.79	Skewness	-.0278444	
99%	.73	.79	Kurtosis	3.106706	

The expected agreement is still 61.44%, but now we would need 69.5% or greater agreement to reject the null at the 5% level, rather than 68%. Not holding the proportions of positive evaluations constant, we obtain similar results, but with greater variation.

Summary

I hope I have convinced you that Mata is not just a programming language. Mata is, like Stata, a useful tool for analysis. Using Mata interactively, we have not done anything we could not have done in Stata if we thought about the problem hard enough, but we have done it with less code, less complicated code, in less time, and it even ran faster!

About the author

William Gould is President of StataCorp, head of development, and principal architect of Mata.