

Contents

	Acknowledgment	xvii
1	Introduction	1
1.1	Is this book for me?	1
1.2	What is Mata?	2
1.3	What is covered in this book	3
1.4	How to download the files for this book	6
2	The mechanics of using Mata	9
2.1	Introduction	9
2.2	Mata code appearing in do-files	10
2.3	Mata code appearing in ado-files	12
2.4	Mata code to be exposed publicly	14
3	A programmer's tour of Mata	17
3.1	Preliminaries	17
3.1.1	Results of expressions are displayed when not stored	18
3.1.2	Assignment	20
3.1.3	Multiple assignment	20
3.2	Real, complex, and string values	22
3.2.1	Real values	22
3.2.2	Complex values	22
3.2.3	String values (ASCII, Unicode, and binary)	22
3.3	Scalars, vectors, and matrices	24
3.3.1	Functions rows(), cols(), and length()	25
3.3.2	Function I()	25
3.3.3	Function J()	26

3.3.4	Row-join and column-join operators	26
3.3.5	Null vectors and null matrices	29
3.4	Mata's advanced features	32
3.4.1	Variable types	32
3.4.2	Structures	34
3.4.3	Classes	36
3.4.4	Pointers	38
3.5	Notes for programmers	40
3.5.1	How programmers use Mata's interactive mode	40
3.5.2	What happens when code has errors	42
3.5.3	The <code>_error()</code> abort function	43
4	Mata's programming statements	45
4.1	The structure of Mata programs	45
4.2	The program body	47
4.2.1	Expressions	47
4.2.2	Conditional execution statement	49
4.2.3	Looping statements	50
4.2.3.1	while	51
4.2.3.2	for	53
4.2.3.3	do while	56
4.2.3.4	continue and break	57
4.2.4	goto	58
4.2.5	return	59
4.2.5.1	Functions returning values	59
4.2.5.2	Functions returning void	60
5	Mata's expressions	61
5.1	More surprises	62
5.2	Numeric and string literals	64
5.2.1	Numeric literals	64
5.2.1.1	Base-10 notation	64

5.2.1.2	Base-2 notation	66
5.2.2	Complex literals	71
5.2.3	String literals	72
5.3	Assignment operator	73
5.4	Operator precedence	74
5.5	Arithmetic operators	75
5.6	Increment and decrement operators	76
5.7	Logical operators	77
5.8	(Understand this ? skip : read) Ternary conditional operator	79
5.9	Matrix row and column join and range operators	80
5.9.1	Row and column join	80
5.9.2	Comma operator is overloaded	81
5.9.3	Row and column count vectors	82
5.10	Colon operators for vectors and matrices	82
5.11	Vector and matrix subscripting	83
5.11.1	Element subscripting	84
5.11.2	List subscripting	86
5.11.3	Permutation vectors	88
5.11.3.1	Use to sort data	88
5.11.3.2	Use in advanced mathematical programming	91
5.11.4	Submatrix subscripting	92
5.12	Pointer and address operators	94
5.13	Cast-to-void operator	97
6	Mata's variable types	99
6.1	Overview	99
6.2	The forty variable types	103
6.2.1	Default initialization	105
6.2.2	Default eltype, orgtype, and therefore, variable type	106
6.2.3	Partial types	106
6.2.4	A forty-first type for returned values from functions	107

6.3	Appropriate use of transmorphic	109
6.3.1	Use transmorphic for arguments of overloaded functions . .	109
6.3.2	Use transmorphic for output arguments	110
6.3.2.1	Use transmorphic for passthru variables	111
6.3.3	You must declare structures and classes if not passthru . . .	112
6.3.4	How to declare pointers	112
7	Mata's strict option and Mata's pragmas	115
7.1	Overview	115
7.2	Turning matastrict on and off	117
7.3	The messages that matastrict produces, and suppressing them . . .	117
8	Mata's function arguments	121
8.1	Introduction	121
8.2	Functions can change the contents of the caller's arguments	121
8.2.1	How to document arguments that are changed	123
8.2.2	How to write functions that do not unnecessarily change arguments	125
8.3	How to write functions that allow a varying number of arguments . .	125
8.4	How to write functions that have multiple syntaxes	127
9	Programming example: n_choose_k() three ways	129
9.1	Overview	129
9.2	Developing n_choose_k()	130
9.3	n_choose_k() packaged as a do-file	134
9.3.1	How I packaged the code: n_choose_k.do	134
9.3.2	How I could have packaged the code	137
9.3.2.1	n_choose_k.mata	139
9.3.2.2	test_n_choose_k.do	141
9.3.3	Certification files	144
9.4	n_choose_k() packaged as an ado-file	145
9.4.1	Writing Stata code to call Mata functions	145
9.4.2	nchooseki.ado	147

9.4.3	test_nchooseki.do	150
9.4.4	Mata code inside of ado-files is private	153
9.5	n_choose_k() packaged as a Mata library routine	153
9.5.1	Your approved source directory	154
9.5.1.1	make_lmatobook.do	156
9.5.1.2	test.do	157
9.5.1.3	hello.mata	157
9.5.1.4	n_choose_k.mata	158
9.5.1.5	test_n_choose_k.do	158
9.5.2	Building and rebuilding libraries	158
9.5.3	Deleting libraries	159
10	Mata's structures	161
10.1	Overview	161
10.2	You must define structures before using them	164
10.3	Structure jargon	165
10.4	Adding variables to structures	166
10.5	Structures containing other structures	166
10.6	Surprising things you can do with structures	167
10.7	Do not omit the word scalar in structure declarations	167
10.8	Structure vectors and matrices and use of the constructor function	168
10.9	Use of transmorphic with structures	170
10.10	Structure pointers	172
11	Programming example: Linear regression	177
11.1	Introduction	177
11.2	Self-threading code	181
11.3	Linear-regression system lr*() version 1	187
11.3.1	lr*() in action	187
11.3.2	The calculations to be programmed	192
11.3.3	lr*() version-1 code listing	194
11.3.4	Discussion of the lr*() version-1 code	197

11.3.4.1	Getting started	198
11.3.4.2	Assume subroutines	199
11.3.4.3	Learn about Mata's built-in subroutines	200
11.3.4.4	Use of built-in subroutine cross()	202
11.3.4.5	Use more subroutines	204
11.4	Linear-regression system lr*() version 2	205
11.4.1	The deviation from mean formulas	206
11.4.2	The lr*() version-2 code	208
11.4.3	lr*() version-2 code listing	211
11.4.4	Other improvements you could make	211
11.5	Closeout of lr*() version 2	213
11.5.1	Certification	213
11.5.2	Adding lr*() to the lmatobook.mlib library	218
12	Mata's classes	221
12.1	Overview	221
12.1.1	Classes contain member variables	222
12.1.2	Classes contain member functions	222
12.1.3	Member functions occult external functions	224
12.1.4	Members—variables and functions—can be private	225
12.1.5	Classes can inherit from other classes	227
12.1.5.1	Privacy versus protection	229
12.1.5.2	Subclass functions occult superclass functions	229
12.1.5.3	Multiple inheritance	230
12.1.5.4	And more	231
12.2	Class creation and deletion	231
12.3	The this prefix	233
12.4	Should all member variables be private?	234
12.5	Classes with no member variables	236
12.6	Inheritance	238
12.6.1	Virtual functions	240

12.6.2	Final functions	243
12.6.3	Polymorphisms	245
12.6.4	When to use inheritance	246
12.7	Pointers to class instances	247
13	Programming example: Linear regression 2	249
13.1	Introduction	249
13.2	LinReg in use	250
13.3	LinReg version-1 code	252
13.4	Adding OPG and robust variance estimates to LinReg	253
13.4.1	Aside on numerical accuracy: Order of addition	257
13.4.2	Aside on numerical accuracy: Symmetric matrices	258
13.4.3	Finishing the code	259
13.5	LinReg version-2 code	260
13.6	Certifying LinReg version 2	260
13.7	Adding LinReg version 2 to the <code>lmatabook.mlib</code> library	261
14	Better variable types	263
14.1	Overview	263
14.2	Stata's macros	264
14.3	Using macros to create new types	264
14.4	Macroed types you might use	266
14.4.1	The boolean type	268
14.4.2	The Code type	269
14.4.3	Filehandle	271
14.4.4	Idiosyncratic types, such as Filenames	272
14.4.5	Macroed types for structures	272
14.4.6	Macroed types for classes	273
14.4.7	Macroed types to avoid name conflicts	273
15	Programming constants	277
15.1	Problem and solution	277
15.2	How to define constants	278

15.3	How to use constants	279
15.4	Where to place constant definitions	279
16	Mata's associative arrays	281
16.1	Introduction	281
16.2	Using class AssociativeArray	282
16.3	Finding out more about AssociativeArray	284
17	Programming example: Sparse matrices	285
17.1	Introduction	285
17.2	The idea	286
17.3	Design	287
17.3.1	Producing a design from an idea	287
17.3.2	The design goes bad	293
17.3.3	Fixing the design	295
17.3.3.1	Sketches of R_x^* () and S_x^* () subroutines	299
17.3.3.2	Sketches of class's multiplication functions	303
17.3.4	Design summary	308
17.3.5	Design shortcomings	311
17.4	Code	312
17.5	Certification script	321
18	Programming example: Sparse matrices, continued	323
18.1	Introduction	324
18.2	Making overall timings	325
18.2.1	Timing T1, Mata $R=RR$	327
18.2.2	Timing T2, SpMat $R=RR$	327
18.2.3	Timing T3, SpMat $R=SR$	328
18.2.4	Timing T4, SpMat $R=RS$	328
18.2.5	Timing T5, SpMat $R=SS$	329
18.2.6	Call a function once before timing	329
18.2.7	Summary	330
18.3	Making detailed timings	330

18.3.1	Mata's timer() function	331
18.3.2	Make a copy of the code to be timed	331
18.3.3	Make a do-file to run the example to be timed	331
18.3.4	Add calls to timer_on() and timer_off() to the code	332
18.3.5	Analyze timing results	336
18.4	Developing better algorithms	338
18.4.1	Developing a new idea	338
18.4.2	Aside	340
18.4.2.1	Features of associative arrays	341
18.4.2.2	Advanced use of pointers	344
18.5	Converting the new idea into code sketches	348
18.5.0.3	Converting the idea into a sketch of R_SxS()	349
18.5.0.4	Sketching subroutine cols_of_row()	352
18.5.1	Converting sketches into completed code	354
18.5.1.1	Double-bang comments and messages	357
18.5.1.2	// NotReached comments	358
18.5.1.3	Back to converting sketches	358
18.5.2	Measuring performance	360
18.6	Cleaning up	360
18.6.1	Finishing R_SxS() and cols_of_row()	361
18.6.2	Running certification	365
18.7	Continuing development	366
19	The Mata Reference Manual	369
A	Writing Mata code to add new commands to Stata	373
A.1	Overview	373
A.2	Ways to structure code	375
A.3	Accessing Stata's data from Mata	379
A.4	Handling errors	384
A.5	Making the calculation and displaying results	387
A.6	Returning results	389

A.7	The Stata interface functions	393
A.7.1	Accessing Stata's data	393
A.7.2	Modifying Stata's data	394
A.7.3	Accessing and modifying Stata's metadata	394
A.7.4	Changing Stata's dataset	396
A.7.5	Accessing and modifying Stata macros, scalars, matrices	396
A.7.6	Executing Stata commands from Mata	397
A.7.7	Other Stata interface functions	399
B	Mata's storage type for complex numbers	401
B.1	Complex values	401
B.2	Complex values and literals	403
B.3	Complex scalars, vectors, and matrices	404
B.4	Real, complex, and numeric eltypes	406
B.5	Functions Re(), Im(), and C()	408
B.6	Function eltype()	408
C	How Mata differs from C and C++	411
C.1	Introduction	411
C.2	Treatment of semicolons	411
C.3	Nested comments	412
C.4	Argument passing	412
C.5	Strings are not arrays of characters	412
C.6	Pointers	413
C.6.1	Pointers to existing objects	413
C.6.2	Pointers to new objects, allocation of memory	413
C.6.3	The size and even type of the object may change	415
C.6.4	Pointers to new objects, freeing of memory	415
C.6.5	Pointers to subscripted values	415
C.6.6	Pointer arithmetic is not allowed	416
C.7	Lack of switch/case statements	416
C.8	Mata code aborts with error when C would crash	416

D	Three-dimensional arrays (advanced use of pointers)	417
D.1	Introduction	417
D.2	Creating three-dimensional arrays	417
	References	419
	Author index	421
	Subject index	423