MAX-PLANCK-INSTITUT FÜR DEMOGRAFISCHE FORSCHUNG
MAX PLANCK INSTITUTE FOR DEMOGRAPHIC RESEARCH

Object-oriented programming in Mata

Daniel C. Schneider

UK Stata Conference, London, September 7-8, 2023

Preliminary Remarks

- This presentation, due to time constraints,
  - is not a proper introduction, not rigorous.
  - You just get a **glimpse of OOP**.
- Uses the -dtms- package as an **illustrative example**.
- Assumes some knowledge of Mata.
- Good resources on the topic
  - Official Stata doc: `help [M-2] class`
  - Bill Gould's (2018) Mata book

The dtms Package

- "dtms": **D**iscrete-**t**ime **m**ulti**s**tate (models / estimation)
- Announcement on Stata Forum:
  - https://www.statalist.org/forums/forum/general-stata-discussion/general/1690703-dtms-new-stata-command-for-discrete-time-multistate-model-estimation

    or google "dtms Stata"
  - Contains location from which to -net install-.
  - Will be moved to SSC.
- Analytical contributions in two soon-to-be-released working papers
  - Schneider (2023)
  - Schneider and Myrskylä (2023)
  - Package doc "Methods and formulas" has sizable chunk of it.

The dtms Package

- `dtms` estimation proceeds in sequential steps:

  (1) model **setup**

  (2) regression **estimation** (`mlogit`)

  (3) predict **transition probabilities** from (2)

  (4) calculate various **results** from (3)

- Calculated/defined objects of levels (0)-(3) can contain multiple elements of the next level => **tree like structure**

The dtms Package: The dtms Tree

```
                                              dtms
                                                |
                            ------------------------ [...]
                            |                   |             LEVEL 1
           --------------- S1        [...] --- S2               MAIN : S-ETUP
          /               / |               /   |
         /         -------  |           [...]   |
        /         /         |                   |
       /         /     ---------- [...]    --------- [...]    - - - - - - - -
   P1--      W1--      |     |             |    |             LEVEL 2
   P2/|      W1/|     E1    E2            E1   E2               SIDE : P-ROPORTION
[...]/    [...]/       |     |             |    |                      RE-W-ARDS
                    [...] [...]            |   [...]            MAIN : E-STIMATE
                                           |
                                     ------------ [...]       - - - - - - - -
                                     |     |                  LEVEL 3
                              ----- T1    T2                    MAIN : T-RANSPROB
                             /       |     |
                            /        |   [...]
                           /         |
                          /    ---------- [...]               - - - - - - - -
                      I1--     |     |                        LEVEL 4
                      I2/|    R1    R2                          SIDE : S-I-MDATA
                   [...]/                                       MAIN : R-ESULT
```

The dtms Package: The dtms Tree

```
. dtms exampletree 7
(loaded/refreshed setup names: ex_tiny)

. dtms dir
  (S) ex_tiny : very small model setup | tra IDs: 1 2 | abs IDs: 4 | 6 ages: 50-100
        [...]

    (E) tiny : (no label) | cmdline: mlogit cog2 iL.cog2 c.age i.(sex educ) numdrinks..
      (T) all : (no label) | dtms trans atmeans: L.cog2=(1 2) age=(60 70 80 90) 1.edu..
        (R) lexp : (no label) | prop: p5060 | timing: mid | calc: analytic | LEXP
        (R) mafn : (no label) | prop: p5060 | timing: mid | calc: analytic | ini: 1 |..
        (R) epis : (no label) | prop: p5060 | timing:  | calc: analytic | EPIS
      (T) men_edlow : (no label) | dtms trans atmeans: L.cog2=(1 2) age=(60 70 80 90)..
        (R) lexp : (no label) | prop: p5060_men_edlow | timing: mid | calc: analytic ..
        (R) mafn : (no label) | prop: p5060_men_edlow | timing: mid | calc: analytic ..
        (R) epis : (no label) | prop: p5060_men_edlow | timing:  | calc: analytic | E..
      (T) men_edmed : (no label) | dtms trans atmeans: L.cog2=(1 2) age=(60 70 80 90)..
        (R) lexp : (no label) | prop: p5060_men_edmed | timing: mid | calc: analytic ..
        (R) mafn : (no label) | prop: p5060_men_edmed | timing: mid | calc: analytic ..
        (R) epis : (no label) | prop: p5060_men_edmed | timing:  | calc: analytic | E..
          [...]
```

The dtms Package: Results Example

```
. dtms erestore (ex_tiny tiny men_edhgh mafn) , replay

Mean age (at) first entry / lifetime risk:
```

cname (composite name): uniquely identifies tree elements

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
rawprob						
60	0.044	0.009	4.746	0.000	0.026	0.063
70	0.116	0.016	7.125	0.000	0.084	0.148
80	0.179	0.018	10.191	0.000	0.144	0.213
90	0.090	0.014	6.343	0.000	0.062	0.118
100	0.000	(omitted)				
lrsk						
lrsk	0.429	0.042	10.170	0.000	0.346	0.511
nrmprob						
60	0.104	0.017	5.972	0.000	0.070	0.138
70	0.270	0.022	12.352	0.000	0.227	0.313
80	0.417	0.015	27.300	0.000	0.387	0.447
90	0.210	0.030	7.031	0.000	0.151	0.268
100	0.000	(omitted)				
mafn						
mafn	72.321	0.831	87.047	0.000	70.693	73.949

All help files and/or subcommands of the **dtms** package:

package and conceptual overview (help dtms)

managing the dtms tree or its elements (help dtms tree)

dtms dir	list a dtms tree
dtms describe	describe a dtms tree element
dtms label	label a dtms tree element
dtms rename	rename a dtms tree element
dtms drop	drop a dtms tree element
dtms usedby	list dtms tree elements that use a particular side tree element
dtms file	save and load setups and all of their downstream elements
dtms settings	query and modify global dtms settings
dtms clear	delete the entire dtms tree and all global dtms settings

adding elements to the dtms tree (help dtms add)

dtms setup	add basic model setup
dtms proportion	add initial proportion
dtms rewards	add non-standard transition timing specification
dtms estimate	add model regression estimate
dtms transprob	add transition probabillities
dtms simdata	add simulated trajectories

calculate results (help dtms result)

dtms result	calculate and add one of the 14+ different outcomes to the dtms tree

extract information from the dtms tree (help dtms extract)

dtms erestore	restore e()-results that are held in some dtms tree elements
dtms combine	post combined estimates of two result tree elements to e()
dtms getmatrix	query various kinds of matrices related to dtms calculations
dtms matbrowse	browse Stata matrix in Stata's data browser

extensive examples and helper commands for executing them (help dtms examples)

dtms exampledata	load example data sets
dtms exampletree	load example tree elements into the dtms tree

Object-Oriented Programming I

- A class: an entity that
  - has members containing data
  - can do stuff: has functions (called methods)
  - defined by a class definition
  - code uses instances of a class (objects)
- OOP section I: Simple but interrelated classes
  Goal: a class that holds Stata e()-results
- OOP section II: Explain how the dtms tree works

OOP I: Simple but Interrelated Classes: exStataMatrix

```
// class definition
class exStataMatrix {

    // data members
    string  scalar  name
    real    matrix  data
    string  matrix  colstripe,
                    rowstripe

    // methods (functions)
    void  fromStata()
    void  toStata()
    void  display()
    [...]
}
```

```
// example usage
sysuse auto
regress mpg weight trunk
mata:
    stm = exStataMatrix(1)
    stm.fromStata("e(V)")
    stm.display()
    stm.toStata("V")
end
```

```
// member function definitions
void exStataMatrix::fromStata(string scalar name) {

    this.name = name
    data      = st_matrix(name)
    colstripe = st_matrixcolstripe(name)
    rowstripe = st_matrixrowstripe(name)
}

void exStataMatrix::toStata(| string scalar newname) {

    if (args()==0) newname = this.name
    st_matrix(newname, data)
    st_matrixcolstripe(newname, colstripe)
    st_matrixrowstripe(newname, rowstripe)
}

void exStataMatrix::display() {

    string scalar tmp
    tmp = st_tempname()
    this.toStata(tmp)
    st_matrix_list(tmp)
}
```

OOP I Side Note: Pointer Variables

- Pointers
  - variables that contain the memory address of another variable
  - can contain addresses of (loosely speaking) anything, where "anything" includes objects
  - see -help [M2] pointers-

OOP I Side Note: Pointer Variables

```
real matrix M
pointer(real matrix) scalar pM

M = J(3,3,3)
pM = &M              // "&": "the address of"
*pM                  // "*": "the thing pointed to by"
pM                   // hex address like 0x3b7e2dc0
*pM = J(3,3,7)       // assignments work too
```

```
// works with objects
class exStataMatrix scalar stm
pointer(class exStataMatrix scalar) scalar pstm

stm.fromStata("e(b)")
pstm = &stm
(*pstm).display()
pstm->display()
```

OOP I: Simple but Interrelated Classes: exSimpleCollection

```
class exSimpleCollection {

    string  vector  names
    pointer vector  elems

    void  add()
    void  drop()
    pointer scalar  getp()
}

void exSimpleCollection::add(pointer scalar p, string scalar name) {

    names = (names , name)
    elems = (elems , p    )
}

pointer scalar exSimpleCollection::getp(string scalar name) {

    real scalar  pos

    pos = selectindex(name:==names)
    return(elems[pos])
}
```

OOP I: Simple but Interrelated Classes: exEsave

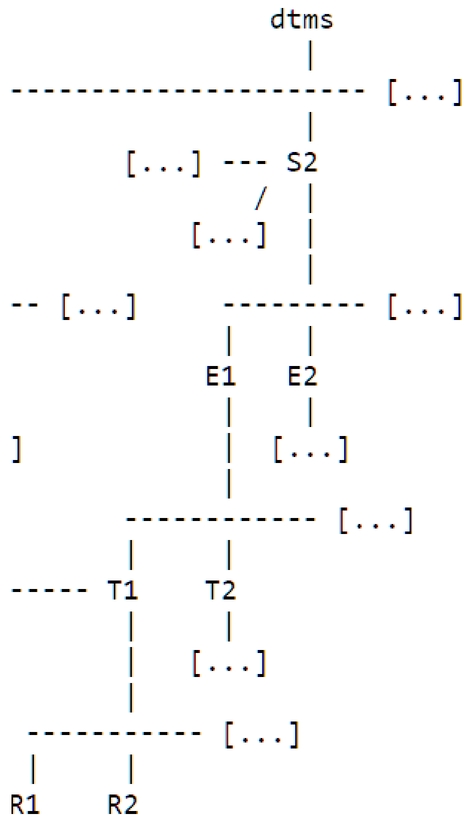
```
class exEsave {

    class exSimpleCollection  scalar  scalars
    class exSimpleCollection  scalar  macros
    class exSimpleCollection  scalar  matrices

    void from_e()
    void to_e()
}

void exEsave::from_e() {

    real scalar          i
    string vector        names
    class exStataMatrix scalar stm

    // [...] (store scalars in collection)
    // [...] (store macros  in collection)

    names = st_dir("e()", "matrix", "*")'
    for (i=1; i<=length(names); i++) {
        stm.fromStata("e(" + names[i] + ")")
        matrices.add(&(stm.copy()), names[i])
            // the copy() method was omitted
            //   from the class def above
    }
}
```

many <u>dtms tree elements</u> have a similar class as a member

OOP II: Objects behind the dtms Tree

Four things need to be solved:

1. getting a pointer to a tree element
2. tree elements must know about upstream tree elements
3. iteration through all elements of a tree
4. idiosyncrasies of tree elements (levels) must be accounted for



Solution:

1.-3. are based on only two class definitions: exTreeColl and exTreeElem

4. makes use of OOP features: inheritance and polymorphism

OOP II

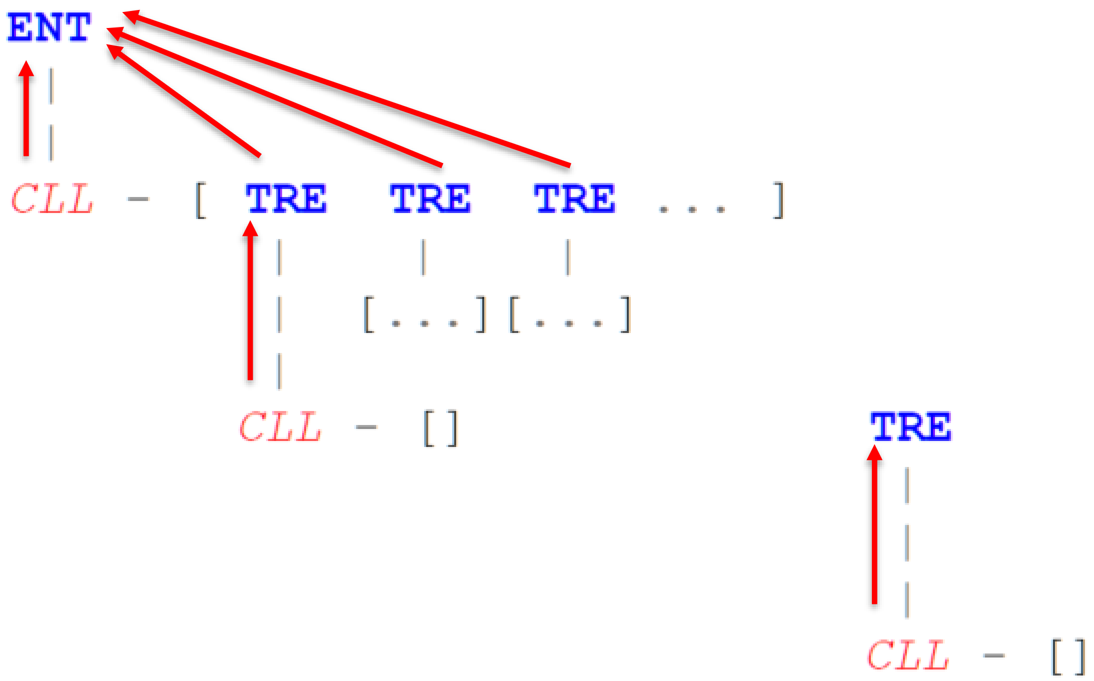
```
// ENT : exEntry     object
// CLL : exTreecoll  object
// TRE : exTreeElem  object

    ENT
     |
    CLL - [ TRE  TRE  TRE ... ]
             |    |    |
             |   [...][...]
             |
            CLL - [ TRE  TRE  TRE ... ]
                     |    |    |
                     |   [...][...]
                     |
                    CLL - [ TRE  TRE  TRE ... ]
                             |    |    |
                             |   [...][...]
                             |
                            CLL [ TRE  TRE  TRE ... ]

// line connections indicate member variables
// brackets indicate a vector of pointers
// ignores distinction b/w objects and pointers to them
```

```
class exEntry {
    class exTreeColl scalar bsecll
}

class exTreeColl {

    string vector  names
    pointer(class exTreeElem scalar) vector  elems

    void  add()
    void  drop()

    pointer(class exTreeElem scalar) scalar  getp()

    pointer(class exTreeElem scalar) scalar  pparelem

    void  dir()
    [...] // more functions that do stuff
}

class exTreeElem {

    string scalar name

    pointer(class exTreeElem scalar) scalar  pup
    pointer(class exTreeElem scalar) scalar  pup()

    real scalar  level

    class exTreeColl nxtcll

    string scalar  infostring()
    [...] // more functions that do stuff

    void setup()
}
```

OOP II: Objects behind the dtms Tree

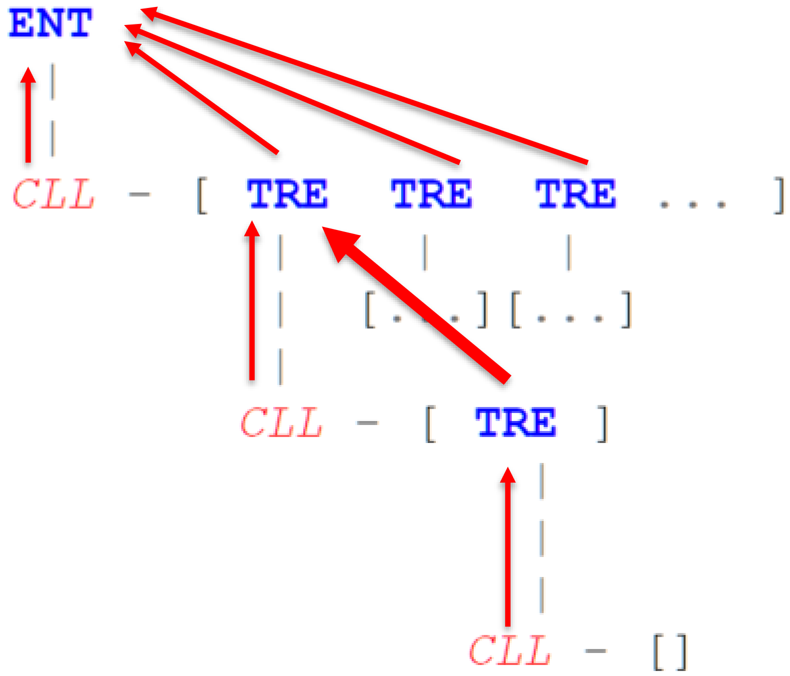
Step 1

An exTreeElem object gets instantiated in memory.
Its member function setup() must be called in order to pass information about its address to its member collection.



Step 2

That way, when the exTreeColl member uses its add() method to add an exTreeElem object, it can pass on that information into the exTreeElem object member.



OOP II: Objects behind the dtms Tree: Solutions to 1.-3.

```
// local def, saves typing and more; see Gould (2018)
local pTRE  pointer(class exTreeElem scalar) scalar



// solution to 1.: locating a downstream element
`pTRE' exTreeColl::getp(string vector cname) {

    real scalar      cname_len,
                     pos
    string vector    cname_rest
    `pTRE'           ptre

    cname_len = length(cname)

    pos = selectindex(cname[1]:==names)

    if(cname_len==1)
        return(elems[pos])
    else {
        cname_rest = cname[2..cname_len]
        ptre = elems[pos]
        return(ptre->nxtcll.getp(cname_rest))
    }
}
```

```
// solution to 2.: locating an upstream element
`pTRE' dtmsTreeElem::pup(real scalar numlevelsup) {

    if (numlevelsup==1)
        return(pup)
    else if (numlevelsup==2)
        return(pup->pup)
    else if (numlevelsup==3)
        return(pup->pup->pup)
}
```

```
// solution to 3.: iterating through all elements
void exTreeColl::dir() {

    `pTRE' ptre

    for (i=1;i<=length(elems);i++) {
        ptre = elems[i]
        ptre->infostring()  // returns TRE description;
                            // or do whatever
        ptre->nxtcll.dir()
    }
}
```

OOP II: Objects behind the dtms Tree: Solution to 4.

```
// accounting for the idiosyncracies of
//   tree elements: inheritance
// before:
//   ENT : exEntry    object
//   CLL : exTreeColl object
//   TRE : exTreeElem object

// now:
//   TRE gets extended into:
//     SET : exSetup  object
//     EST : exEstim  object
//     TRN : exTrans  object
//     RSL : exReslt  object

    ENT
     |
    CLL - [ SET  SET  SET ... ]
             |    |    |
             |   [...][...]
             |
            CLL - [ EST  EST  EST ... ]
                     |    |    |
                     |   [...][...]
                     |
                    CLL - [ TRN  TRN  TRN ... ]
                             |    |    |
                             |   [...][...]
                             |
                            CLL [ RSL  RSL ... ]
```

```
// OOP-feature : inheritance
class exSetup extends exTreeElem {

    [...]
}

class exEstim extends exTreeElem {

    [...]
}

class exTrans extends exTreeElem {

    [...]
}

class exReslt extends exTreeElem {

    [...]
}

// for example:

class exTrans extends exTreeElem {

    class exEsave scalar esv
    real matrix calc_transprobs()
}
```

OOP II: Objects behind the dtms Tree

```
// OOP-feature : polymorphism
class exTreeElem {
    [...]

    virtual string scalar infostring()
}

class exSetup extends exTreeElem {

    [...]
    virtual string scalar infostring()
}

class exEstim extends exTreeElem {

    [...]
    virtual string scalar infostring()
}

class exTrans extends exTreeElem {

    [...]
    virtual string scalar infostring()
}

class exReslt extends exTreeElem {

    [...]
    virtual string scalar infostring()
}
```

This technique was used in dtms dir and coded explicitly in solution to 3.

Thank you

schneider@demogr.mpg.de

References

Gould, William W. (2018). The Mata Book: A Book for Serious Programmers and Those Who Want to Be. Stata Press.

Schneider, Daniel C. (2023). "Inference for Discrete-Time Multistate Models: Asymptotic Covariance Matrices, Partial Age Ranges, and Group Comparisons." [working title] *MPDIR Working Paper*, forthcoming.

Schneider, Daniel C. and Mikko Myrskylä (2023). Extending Discrete-Time Multistate Models Using Markov Chains with Rewards: New Outcome Measures and Inference Results. [working title] *MPDIR Working Paper*, forthcoming.

The dtms Package: The dtms Tree

The dtms tree

```
The dtms tree is structured as:

                                              dtms
                                                |
                            ------------------------ [...]
                            |                   |             LEVEL 1
           --------------- S1        [...] --- S2               MAIN : S-ETUP
          /               / |                /  |
         /         -------  |            [...]  |
        /         /         |                   |
       /         /    ---------- [...]     --------- [...]    - - - - - - - - -
   P1--      W1--     |     |              |    |             LEVEL 2
   P2/|      W1/|    E1    E2             E1   E2               SIDE : P-ROPORTION
[...]/    [...]/      |     |              |    |                      RE-W-ARDS
                    [...] [...]            |  [...]             MAIN : E-STIMATE
                                           |
                                     ------------ [...]       - - - - - - - - -
                                     |     |                  LEVEL 3
                              ----- T1    T2                    MAIN : T-RANSPROB
                             /       |     |
                            /        |   [...]
                           /         |
                          /    ----------- [...]              - - - - - - - - -
                      I1--     |     |                        LEVEL 4
                      I2/|    R1    R2                          SIDE : S-I-MDATA
                   [...]/                                       MAIN : R-ESULT

where the four main tree elements are:

   S# = (Model) Setup number #
   E# = Estimate number # of the setup it belongs to
   T# = Transition probability ("transprob") number # of the estimate it belongs to
   R# = Result number # of the transprob it belongs to

plus three helper (side tree) elements:

   P# = (Initial) Proportion number # of the setup it belongs to
   W# = (Time) Rewards number # of the setup they belong to
   I# = Simulation data number # of the transprob it belongs to
```

The dtms Package: Project Size

- Lines of code, counting blank lines, roughly:
  - Stata: 5,000
  - test script: 6,000
  - auxiliary: 3,000
  - Mata: 11,000, makes heavy use of OOP
- Source code not (yet) online
  may be made available in the future