MATLAB and Big Data: Illustrative Example

Rick Mansfield
Cornell University

August 19, 2014

Goals

- Use a concrete example from my research to:
- Demonstrate the value of vectorization
- Introduce key commands/functions facilitating vectorization in MATLAB
- Highlight the value (and necessity) of using sparse matrices
- Demonstrate the uses for subscript, logical and linear indexing
- Demonstrate the syntax for calling optimization routines in MATLAB

Background

- Research Question: How much does the school system/neighborhood you attend/grow up in affect long run outcomes (wages, college completion, etc.)?
- Challenge: Schools/neighborhoods with more desirable amenities attract students and parents with potentially superior (observable and unobservable) characteristics
- Simple model of outcomes:
- $Y_s = \overline{X}_s \beta + \overline{X}_s^U \beta^U + SQ_s$
- Project Goal: Provide a lower bound estimate of the contribution of schools/neighborhoods to the cross-sectional variance in later outcomes.

Background

- If students had fully idiosyncratic preferences for schools/neighborhoods . . .
- Then school-average values of each student characteristic would converge to the population average at each school as school sizes became large.
- $\Rightarrow$ Variation in school-average student characteristics reflects the fact that these characteristics predict relative taste for particular school/neighborhood amenities.

Background

- Key hypothesis: If unobservable student characteristics only affect tastes for the same set of school/neighborhood amenities as observable characteristics . . .
- Then average values of both sets of characteristics will only differ across schools because these amenities differ across schools.
- $\Rightarrow$ School-average values of observable student characteristics can serve as a control function for school-average values of unobservable student characteristics.
- $Y_s = \overline{X}_s \beta^C + \nu_s, \quad cov(\nu_s, \overline{X}_s^U \beta^U) = 0$

Background

- Control function will also soak up all variation in the desired amenities . . .
- . . . But can isolate the contribution to earnings of the component of school/neighborhood quality unknown or unvalued by parents at the time school/neighborhood is chosen.

Goal of Sample Program

- Test this hypothesis ...
- by performing a Monte Carlo simulation of a model of school choice.
- Since the hypothesis requires observing ...
  - A large number of student characteristics for each student
  - A large number of students at each school
  - A large number of schools
- ...Computational efficiency is at a premium here!
- Good context for illustrating MATLAB big data techniques.
- Currently using North Carolina administrative data that satisfies all these requirements!

Model of School Choice

- Hedonic model of school choice
- Set of L observable student characteristics, $\{X_{1i}, \ldots, X_{Li}\}$
- Set of M unobserved student characteristics, $\{X^U_{1i}, \ldots, X^U_{Mi}\}$
- Set of $K <= L$ underlying desired/undesired school/neighborhood-level amenities $\{A_{1s}, \ldots, A_{Ks}\}$
  - Assumes that amenity space has an underlying factor structure
  - All variation in school/neighborhood desirability captured by K factors.

Model of School Choice

$$WTP_i(s) = \gamma_{1i} A_{1s} + \gamma_{2i} A_{2s} + \cdots + \gamma_{Ki} A_{Ks} + \epsilon^*_{s,i}. \qquad (1)$$

- $\epsilon^*_{s,i}$ reflects idiosyncratic tastes of student i for school/neighborhood s that are unrelated to the school's amenities.

Model of School Choice

- Let the taste parameters γ_{ki} depend on all L elements of $\mathbf{X_i}$ and all M elements of $\mathbf{X_i^U}$

$$\gamma_{ki} = \sum_{l=1}^{L} \delta_{kl} X_{li} + \sum_{m=1}^{M} \delta_{km}^{U} X_{mi}^{U} + \kappa_{ki} \tag{2}$$

- Formulation allows each observable and unobservable student characteristic to differentially affect the relative taste for each of the K amenities
- κ_{ki} is the component of i's taste for amenity k that is unpredictable given $\mathbf{X_i}$ and $\mathbf{X_i^U}$
- κ_{ki} influences school choice but has no direct effect on student outcomes.

General Case: Model of School Choice

- Can collect the terms involving the κ_{ki} and combine them with ϵ^*_{si} to rewrite as

$$WTP_i(s) = \sum_{k=1}^{K} (\sum_{l=1}^{L} \delta_{kl} X_{li} + \sum_{m=1}^{M} \delta^U_{km} X^U_{mi}) A_{ks} + \epsilon_{si}$$

where

$$\epsilon_{si} = \sum_{k=1}^{K} \kappa_{ki} A_{ks} + \epsilon^*_{si}$$

- Given an inelastic housing supply, the equilibrium price function $P(A_{1s}, \ldots, A_{Ks})$ adjusts to clear the housing market.

Simulation

- First step: Draw a matrix of preferences for N students choosing between S schools.
- Could do this by writing a nested for-loop ...

Vectorization

- Or, could vectorize.
- Write the above equation in matrix form, and use matrix operations:
- $WTP_i(s) = \sum_{k=1}^{K}(\mathbf{X}_i\boldsymbol{\delta}_k + \mathbf{X}_i^U\boldsymbol{\delta}_k^U)A_{ks} + \epsilon_{si}$
- $\mathbf{X}_i$: $1xL$, $\boldsymbol{\delta}_k$: $Lx1$
- Did it speed up the computation?
- Try re-running the program with L = 500, M = 500
- How about now?
- Vectorization tends to pay off when loops require many iterations.

Vectorization

- Can vectorize further ...
- $WTP_i(s) = (\mathbf{X}_i\boldsymbol{\delta} + \mathbf{X}_i^U\boldsymbol{\delta}^U)\mathbf{A}_s + \epsilon_{si}$
- $\boldsymbol{\delta}$: LxK, $\mathbf{A}_s$: $Kx1$
- Small efficiency payoff this time (despite small K)

Vectorization

- Can vectorize further ...
- $\mathbf{WTP} = (\mathbf{X}\boldsymbol{\delta} + \mathbf{X}^U \boldsymbol{\delta}^U + \boldsymbol{\kappa})\mathbf{A} + \epsilon$
- $\mathbf{X}$: IxL, $\mathbf{A}$: KxS.
- Huge efficiency payoff from eliminating the long student loop.
- Note that testing with a small sample would be misleading about the time savings from vectorization!

Approach to Simulation

- One possibility:
- Have students/parents make school/neighborhood choices based on initial price vector ...
- Raise the price of oversubscribed schools/neighborhoods and lower the price of undersubscribed schools/neighborhoods ...
- Until equilibrium is reached in which house prices clear the housing market.
- Requires a while-loop, can be quite slow to converge.

Alternative approach to Simulation

- Use the 1st Welfare Theorem:
- A perfectly competitive market always reaches the efficient allocation.
- Can solve directly for the allocation that maximizes the total willingness of pay of all students . . .
- Subject to the constraints that
  - No school can be oversubscribed
  - Every child must attend exactly one school

Formal Statement of Problem

$$\max_{f:\mathcal{I}\to\mathcal{S}} \sum_{i\in\mathcal{I}} WTP_{is}$$
$$s.t. \sum_i 1(f(i)=s) = \overline{I}_s \ \forall\ s \qquad s.t. \sum_s 1(f(i)=s) = 1\ \forall\ i \tag{3}$$

- $1(f(i)=s)$ indicates that student i chooses school/neighborhood s.

Linear Programming Version

- This optimization problem can be recast as a linear programming problem:

$$\begin{aligned}
&\max_{\mathbf{x}} \mathbf{U}^T * \mathbf{x} \\
&s.t.\ P * \mathbf{x} = \bar{I}_s \ \forall\ s \\
&s.t.\ Q * \mathbf{x} = 1 \ \forall\ i \\
&s.t.\ \mathbf{x} \geq 0 \\
&s.t.\ \mathbf{x} \leq 1
\end{aligned} \tag{4}$$

- $\mathbf{U}$ is a $(N * S)$x1 column created by stacking the rows of the NxS matrix of student WTP for schools.
- $\mathbf{x}$ consists of a $(N * S)\mathrm{x}1$ vector indicating potential school assignments.

Linear Programming Problems in MATLAB

- Can call MATLABs linear programming solver: linprog.
- linprog requires the user to supply several arguments:
    - An objective function (the row vector $\mathbf{U}$, in this case)
    - A matrix of inequality constraints on feasible allocations (each row represents a different constraint)
    - The values associated with these inequality constraints
    - A matrix of equality constraints on feasible allocations
    - The values associated with these equality constraints
    - An initial allocation (starting point)
    - Upper and lower bounds for each parameter
    - Display and algorithm choice options

Linear Programming Problem

- Need $\mathbf{U}$ to look like:

$$\mathbf{U} = \begin{pmatrix} U_{11} \\ \vdots \\ U_{N1} \\ U_{12} \\ \vdots \\ U_{N2} \\ \vdots \\ U_{1S} \\ \vdots \\ U_{NS} \end{pmatrix}$$

Linear Programming Problem

- Need $\mathbf{x}$ to look like:

$$x = \begin{pmatrix} x_{11} \\ \vdots \\ x_{N1} \\ x_{12} \\ \vdots \\ x_{N2} \\ \vdots \\ x_{1S} \\ \vdots \\ x_{NS} \end{pmatrix}$$

Linear Programming Problem

- Matrix of school size constraints $P\mathbf{x} = \bar{I}$

$$P = \begin{pmatrix} 1 & \dots & 1 & 0 & \dots & 0 & \dots & \dots & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 & \dots & 1 & \dots & \dots & 0 & \dots & 0 \\ & & \vdots & & & \vdots & & \vdots & & \vdots & \\ 0 & \dots & 0 & 0 & \dots & 0 & \dots & \dots & 1 & \dots & 1 \end{pmatrix}$$

$$\bar{I} = \begin{pmatrix} \bar{I}_1 \\ \bar{I}_2 \\ \vdots \\ \bar{I}_S \end{pmatrix}$$

Linear Programming Problem

- Matrix of constraints requiring that each student select one school: $\mathbf{Qx} = \mathbf{1}$

$$Q = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & \dots & \dots & 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & 0 & 1 & \dots & 0 & \dots & \dots & 0 & 1 & \dots & 0 \\ & & \vdots & & & & \vdots & & \vdots & & & & \vdots & \\ 0 & 0 & \dots & 1 & 0 & 0 & \dots & 1 & \dots & \dots & 0 & 0 & \dots & 1 \end{pmatrix}$$

Kronecker Product

- Can write a loop to populate these constraint matrices ...
- ...or can vectorize using the Kronecker product.
- Kronecker product of two matrices formed by taking products of each pair of elements of the two matrices.
- Kronecker product of A and B denoted $A \otimes B$

Example

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

$$B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$$A \otimes B = \begin{pmatrix} A_{11} * B & A_{12} * B \\ A_{21} * B & A_{22} * B \end{pmatrix}$$

$$= \begin{pmatrix} A_{11} * B_{11} & A_{11} * B_{12} & A_{12} * B_{11} & A_{12} * B_{12} \\ A_{11} * B_{21} & A_{11} * B_{22} & A_{12} * B_{21} & A_{12} * B_{22} \\ A_{21} * B_{11} & A_{21} * B_{12} & A_{22} * B_{11} & A_{22} * B_{12} \\ A_{21} * B_{21} & A_{21} * B_{22} & A_{22} * B_{21} & A_{22} * B_{22} \end{pmatrix}$$

School Capacity Constraints

- Let A be an NxN identity matrix:

$$A = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ & & \vdots & \\ 0 & 0 & \dots & 1 \end{pmatrix}$$

- Let B be a row vector of S ones:

$$B = \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix}$$

$$A \otimes B = \begin{pmatrix} 1 & \dots & 1 & 0 & \dots & 0 & \dots\dots & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 & \dots & 1 & \dots\dots & 0 & \dots & 0 \\ & & \vdots & & & \vdots & \vdots & & \vdots & \\ 0 & \dots & 0 & 0 & \dots & 0 & \dots\dots & 1 & \dots & 1 \end{pmatrix}$$

Individual School Attendance Requirements

$$B \otimes A = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & \dots & \dots & 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & 0 & 1 & \dots & 0 & \dots & \dots & 0 & 1 & \dots & 0 \\ & & \vdots & & & & \vdots & & \vdots & & & & \vdots & \\ 0 & 0 & \dots & 1 & 0 & 0 & \dots & 1 & \dots & \dots & 0 & 0 & \dots & 1 \end{pmatrix}$$

Problem

- Just one problem . . .
- The school size constraint matrix (P) is $Sx(N * S)$ . . .
- and the individual attendance constraint matrix (Q) is $Nx(N * S)$
- If S is 100 and N is 100000 . . .
- then P is $100x10,000,000$ . . .
- And Q is $100,000x10,000,000$!
- Matrices this big require far more memory than even the largest servers can provide to MATLAB. . .
- unless elements are stored in binary format!

Solution: Sparse Matrices

- Notice that to re-create either constraint matrix we only need to know the row and column positions of each of the 1s . . .
- And there are only $N * S$ ones in each matrix (a single one in each column).
- Thus, we only really need to store $N * S * 2$ pieces of information (rows and columns for each one)
- More generally, if the ones were replaced by an arbitrary set of real numbers . . .
- Could recreate the matrix if we knew the positions of the real numbers and their values
- $N * S * 3$ pieces of information.

Solution: Sparse Matrices

- Matrices that consist primarily of zeros are referred to as *sparse* matrices.
- MATLAB can store matrices in sparse format ...
- Only the positions and values of non-zero entries are stored.
- Can perform operations on huge matrices ...
- As long as there aren't too many non-zero elements.

The Sparse Command in MATLAB

- The “sparse” command in MATLAB creates a sparse matrix given the following arguments
  - A vector of row indices for each non-zero element
  - A vector of column indices for each non-zero element
  - A vector of values for each non-zero element
  - The number of total rows in the matrix
  - The number of total columns in the matrix
  - The total number of non-zero elements for which MATLAB should clear space (useful if you expect to add non-zero elements to the matrix later on in the program)

School size constraints in sparse format

- What is the appropriate vector of row indices for the school size constraints?
- Need the first S elements to be 1, the next S elements to be 2, ... last S elements to be S
- Can create this vector using $R = D \otimes E$, where:

$$D = \begin{pmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ S \end{pmatrix} \quad E = \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

School size constraints in sparse format

- Column indices are straightforward:

$$C = \begin{pmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ N * S \end{pmatrix}$$

- Since all the non-zero entries are ones, the value vector simply consists of $N * S$ ones.
- Number of rows = S, number of columns = $N * S$, number of nonzeros = $N * S$

Matrix Indexing

- Three basic techniques for selecting elements of a matrix:
  - Subscript Indexing
  - Logical Indexing
  - Linear Indexing
- Each is appropriate for different forms of selection

Subscript Indexing

- 5th student's WTP for the 8th school: WTP(5,8)
- 5th student's WTP for all schools: WTP(5,:)
- First 5 student's WTP at all schools: WTP(1:5,:)
- 5th and 7th student's WTP at 4th and 6th schools: WTP([5,7],[4,6])
- 5th student's WTP at their chosen school: WTP(5,Assignment(5))
- 5th and 7th student's WTP at their own and each other's chosen schools: WTP([5,7],Assignment([5,7]))

Logical Indexing

$$WTP = \begin{pmatrix} -2 & 4 \\ 1 & -5 \\ -2 & -7 \\ 3 & -4 \end{pmatrix}$$

- Suppose you want to identify all student-school combinations in which the student WTP is less than 0 (willing to pay to avoid the school/live in the neighborhood);
- Can create a logical matrix: $WTP_{ltzero} = (WTP < 0)$:

$$WTP_{ltzero} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$$

- The logical matrix places ones at the positions of elements that satisfy the provided condition.

Logical Indexing

- Can replace negative values with zero via: $WTP(WTP_{ltzero}) = 0$:

$$WTP = \begin{pmatrix} 0 & 4 \\ 1 & 0 \\ 0 & 0 \\ 3 & 0 \end{pmatrix}$$

Logical Indexing

- In my program, I want to convert the $(N * S)x1$ vector of 0's and 1's indicating school choices into a vector that provides the index number of the school chosen for each student.
- First, can transform the output from linprog into an NxS matrix using reshape:
- $Assignment2 = reshape(Assignment, [N, S])$:

$$Assignment = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \Rightarrow Assignment2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Logical Indexing

- Can also transpose to observe, for each school, which students selected the school:
- $Assignment3 = (Assignment2)'$:

$$Assignment3 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

- Consider the following matrix, which replicates N times a column of school indices $(1, \ldots, S)$:
- $Temp = repmat((1:S)', 1, N)$;

$$Temp = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \end{pmatrix}$$

Logical Indexing

- Can generate the desired N-vector of school assignments via:
- Assignment3 = logical(Assignment3);
- Schoolassign = Temp(Assignment3):

$$\begin{pmatrix} 1 \\ 2 \\ 2 \\ 1 \end{pmatrix}$$

- Vector is created by moving down columns (left to right), selecting elements flagged by a “1”.
- Logical indexing is generally quite useful (and efficient!) when you want to select (or replace) elements that satisfy an easily programmed condition.

Linear Indexing

- Suppose you wanted a vector with each student's WTP at their chosen schools only.
- IOW, you wanted a vector that looks like:
- $$[WTP(1, Schoolassign(1)), \ldots, WTP(N, Schoolassign(N))]$$
- Can't use $WTP(:, Schoolassign(:))$ … Will get an NxN matrix!
- Solution: Linear indexing

Linear Indexing

- Can reshape any matrix into a vector by stacking all the columns (moving left to right)
- Each original element's position in the stacked vector is its linear index.
- e.g. if a matrix is MxN, the (i, j) element's linear index is $M * (j - 1) + i$
- If a matrix is called with only one argument (e.g. WTP(25)), MATLAB will find the element whose linear index value is 25.
- If WTP were 10x10, $WTP(25) = WTP(5, 3)$

Linear Indexing

- To obtain the elements of a matrix associated with a list of subscript pairs:
  1. Use the command *sub2ind* to convert the list to its linear index equivalent
  2. Evaluate the original matrix at the linear index vector.
- Example:
- $WTP\_linindex = sub2ind(size(WTP), [1 : N], Schoolassign);$
- $WTP\_choice = WTP(WTP\_linindex);$

Accumarray

- Given the allocation of students to schools provided by the solution to the linear programming problem . . .
- . . . now need to calculate means of each observable and unobservable student characteristic . . .
- at each school.
- Could parallelize this loop. . .
- Or could vectorize using the accumarray command.

Accumarray

- Accumarray offers the same functionality as the “by” option in Stata:
- Performs functions within (potentially many) subgroups.

Arguments Passed to Accumarray

- A (possibly multidimensional) index vector identifying the subgroup associated with each observation
- A vector of values on which the function is to be performed
- A size for the resulting vector or matrix (necessary when not all subgroups of interest are represented in the index vector)
- The function being applied within each subgroup
- A flag for whether you want the output to be sparse
- A value to be filled in when no observations exist in the subgroup

Accumarray Example 1

- 4 student, 2 school case:
- Consider a single characteristic, X_1:

$$X_1 = \begin{pmatrix} 2 \\ 1 \\ 3 \\ 8 \end{pmatrix}$$

- Suppose the 1st and 4th students attend school 1, 2nd and 3rd students attend school 2.
- Index vector:

$$IX = \begin{pmatrix} 1 \\ 2 \\ 2 \\ 1 \end{pmatrix}$$

Accumarray Example 1

- $Means = accumarray(IX, X_1, [2, 1], @mean)$

$$Means = \begin{pmatrix} 5 \\ 2 \end{pmatrix}$$

Accumarray Example 2

- Can also provide two (or more) dimensions of subscripts to create a matrix of means
- Note: Vector of values must be a vector not a matrix (so must first reshape matrix to a vector)

$$IX2 = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 2 & 1 \\ 1 & 1 \\ 1 & 2 \\ 2 & 2 \\ 2 & 2 \\ 1 & 2 \end{pmatrix} \qquad X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 3 \\ 8 \\ 6 \\ 4 \\ 2 \\ 5 \end{pmatrix}$$

Accumarray Example 2

- $Means2D = accumarray(IX2, X, [8, 2], @mean)$

$$Means2D = \begin{pmatrix} 5 & 5.5 \\ 2 & 3 \end{pmatrix}$$

- In my code:
- $X_s = accumarray([repmat(Schoolassign, L, 1), kron((1 : L)', ones(N, 1))], X_O(:), [], @mean)$

Results of Simulation

- Given school-average values of all observable and unobservable characteristics . . .
- Can regress the combined contribution of school-average unobservables on the vector of school-average observables . . .
- . . . And observe that the contribution to the average school outcome of sorting on unobservables is completely predictable using the average observables.
- IOW, school-average observables span the same space as school-average unobservables.

Results of Simulation

- Result requires:
  - Large samples of students choosing schools (but not more than is realistic)
  - Dimension of relevant school/neighborhood amenities < Dimension of observed student characteristics
  - No neighborhood attribute whose desirability varies with unobservable but not observable characteristics
- Residual between-school variation thus reflects the component of school quality not known or valued at the time neighborhoods are chosen.
- Can evaluate the quality of information about school/neighborhood quality that parents have at their disposal.

Useful Commands for Vectorizing

- MATLAB provides a list of commands commonly used in vectorizing:
- typing "Vectorization" into help search box and scroll to bottom
- . . . But they omit a number of commands I have found to be absolutely essential.

Useful Commands for Vectorizing Code

- accumarray: perform functions by subgroup (equivalent of "by" in Stata)
- unique: identify unique elements and (importantly) rows of elements (and their positions in the original matrix)
- ismember: Determine whether elements of matrix A are also elements of matrix B (and the positions of the A elements in the B matrix, where available)
- kron: Kronecker product! Very useful for creating analytical Jacobians!
- intersect: Finds the intersection of the elements of vectors A and B.

Final Notes

- Vectorization is a substitute in many cases for parallel processing ...
- But it is also complementary.
- Vectorize to the extent possible (given memory limits) ...
- And then process the remaining (much smaller) loop on a small number of processors