MATLAB: Structures and Cell Arrays

Kipp Martin
University of Chicago
Booth School of Business

February 16, 2012
The M-files

The following files are used in this lecture.

- studentStruct.mat
- stockData.mat
- readStudents.m
- studentscores.txt
- createCellArray.m
- studentRecordCell.mat
Outline

Structures

Cell Arrays
Structures

In MATLAB everything is an array.

**Problem 1:** You need **mixed** data types as in an Excel range.
In MATLAB you cannot create an array that is a direct analog to an Excel range which can contain mixed data types. In MATLAB you cannot do the following:

\[
\text{>> } x = [\text{'}hello\text{' }55 \text{ ; } \text{'}world\text{' }77]
\]

\[
x =
\]

hello7
worldM

Poor MATLAB gets very confused when you try to mix data types in a single array.
Structures

There are excellent alternatives to this problem:

▶ Use a MATLAB **structure**. A structure is like a class. Each property of the structure is an array. The arrays may be of a different data type.

▶ Use a MATLAB **cell array**. In a cell array, each element of the array is another array. We are generating arrays recursively and they do not need to be of the same data type.

First we work with structures.
In order to create the equivalent of an Excel range in MATLAB you would create a MATLAB structure. Then you would create an array, where each element of the array is a structure.

A structure is similar to a class in VBA except it does not have methods. A MATLAB structure only has properties.

In our example of students and their grades, there will be structure for each student (row).

There will be property for each column. For example, there will be a LastName, FirstName, Midterm, etc. property.
Consider the first student. I define the structure (which I name \texttt{studentStruct}) for \textbf{Tom Jones} as follows:

\begin{verbatim}
studentStruct(1).FirstName = 'Tom'
studentStruct(1).LastName = 'Jones'
studentStruct(1).Program = 'Evening MBA'
studentStruct(1).Section = 81
studentStruct(1).Midterm = 63
.
.
.
studentStruct(1).Quiz5 = 56
\end{verbatim}
Likewise for student number 4, **Kathy Murigami**, I have:

\[
\begin{align*}
\text{studentStruct}(4).\text{FirstName} &= \text{'Kathy'} \\
\text{studentStruct}(4).\text{LastName} &= \text{'Murigami'} \\
\text{studentStruct}(4).\text{Program} &= \text{'Evening MBA'} \\
\text{studentStruct}(4).\text{Section} &= 81 \\
\text{studentStruct}(4).\text{Midterm} &= 96 \\
\text{studentStruct}(4).\text{Quiz5} &= 66
\end{align*}
\]
Here is another way I could have created the structure associate with Kathy Murigami using the **struct** function.

```matlab
studentStruct(4) = struct('FirstName', 'Kathy', ... 'LastName', 'Murigami' ... 'Program', 'Evening MBA' ... 'Section', 81 ... 'Midterm' 96 ...)
```

In this case we are using name-value pairs.
**Problem 2:** In a MATLAB array every row must have the same number of columns (or each column the same number of rows).

You cannot have:

\[ A = \begin{bmatrix} 1 & 2 & 3; & 4 & 5 \end{bmatrix} \]

This is very limiting. We want to store data where parts of the data are not only of a different type, but a different size.

Consider the stock data in the file **portOptData.xlsx**
How might we represent this as a structure?
See `stockData.mat`

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>yearNames</td>
<td>structure</td>
</tr>
<tr>
<td>stockNames</td>
<td>structure</td>
</tr>
<tr>
<td>stockReturn</td>
<td>matrix (double)</td>
</tr>
<tr>
<td>S&amp;PReturns</td>
<td>array (double)</td>
</tr>
<tr>
<td>Probabilities</td>
<td>array (double)</td>
</tr>
<tr>
<td>MinReturnLowerBound</td>
<td>double</td>
</tr>
<tr>
<td>MinReturnUpperBound</td>
<td>double</td>
</tr>
</tbody>
</table>
The structure `stockData` in the variable editor.

Why did I store `yearNames` and `stockNames` as `structures` rather than `arrays`. 
Structures

For the student scores data see:

- `studentStruct.mat` – a MATLAB structure with the student grades
- `studentscores.txt` – text file with comma separated values, this is used to generate `studentStruct`
- `readStudents.m` – the MATLAB m-file that reads the data in `studentscores.txt` and generates `studentStruct`

For the stock data data see:

- `stockData.mat` – a MATLAB structure with the magazine subscription data
- `hw_5.xlsm` – the structure `stockData.mat` is modeled after the spreadsheet `stockData` is this workbook.
Creating a structure: How do we create a structure? How did I create stockData.mat?

In MATLAB there is no Dim command. There is no VBA equivalent of

Dim stockData As Structure

We have three options in MATLAB.
Structures

Creating a structure – Option 1: Just start typing away in the command window! Let’s create a simple structure names that will have two properties: firstName and lastName.

```plaintext
names(1).firstName = 'Joey'
names(1).lastName = 'Votto'

names(2).firstName = 'Aroldis'
names(2).lastName = 'Chapman'
```

- we use 1-based counting
- records don’t need to be complete – e.g. we can have a firstName but no lastName
Creating a structure – Option 2: Use the \texttt{struct()} method in MATLAB.

Create a 1 by 2 structure name with properties \texttt{firstName} and \texttt{lastName}

\begin{verbatim}
name(1, 2) = struct('firstName', [], 'lastName', [])
\end{verbatim}

Create a 2 by 1 structure name with properties \texttt{firstName} and \texttt{lastName}

\begin{verbatim}
name(2, 1) = struct('firstName', [], 'lastName', []
\end{verbatim}

You can add a third, fourth, etc., record at any time.
Creating a structure – Option 3: Read a file (or some other data source – text file, XML file, SQL database) and use MATLAB scripting.

We illustrate the above with createStudentsStruct.m in the next module.
Structures

After creating a structure in MATLAB use `save`, e.g.

```
save stockData
```

This will save the structure with a .MAT extension.

To get the structure back:

- `load stockData`
- `double click on stockData.mat in explorer window`

Caution: the above will save any other variables you have in your work space.
Structures

If you have other variables around and want to save only the structure, do the following:

```
save FILENAME STRUCTURENAME
save tmp stockData
```

will save the structure `stockData` in the `.mat` file `tmp.mat`. The practice I like is:

```
save stockData stockData
```

This way I have a `.mat` file with the same name as my structure.
Structures

The \texttt{size()} function is very useful when working with structures.

- What is \texttt{size(stockData)}?
- What is \texttt{size(stockData.stockNames)}?
- What is \texttt{size(stockData(1).stockNames)}?
- What is \texttt{size(stockData.stockNames(1))}?
- What is \texttt{size(stockData.stockNames(1).Name)}?
- What is \texttt{size(stockData.stockReturns)}?
- What is \texttt{size(stockData(1,1).stockNames)}?
Structures

**Addressing:** What you learned about addressing for matrices of numbers and strings pretty much applies to structures.

- What is `stockData.stockReturns(5,3)`?
- What is `stockData.stockReturns(5,:)`?
- What is `stockData.stockReturns(5,[1 4])`?
- What is `stockData.yearNames(1,[1 4])`? – be careful
- What is `stockData.yearNames(1).Name(1,[1 4])`?
Structures

Structures versus string data:

What is the size of the following A-matrix?

\[ A = \text{strvcat}('a', \ 'abcdefghijklmnopqrstuvwxyz') \]

What about the following?

\[ A(1).\text{string} = 'a' \]
\[ A(2).\text{string} = 'abcdefghijklmnopqrstuvwxyz' \]

Advantages/disadvantages of the two approaches?
Instead of

```matlab
>> x = ['hello' 55 ; 'world' 77]
```

we write

```matlab
>> x = {'hello' 55 ; 'world' 77}
```

and we get

```
>> x =
    'hello'      [55]
    'world'      [77]
```

We have created a 2 × 2 cell array.
It is even neater. Watch this:

```matlab
>> A = [1 3; 7.1 9]
A =
    1.0000    3.0000
    7.1000    9.0000

>> x = {'hello' 55; 'world' A}
x =
    'hello'    [55]
    'world'    [2x2 double]
```
Here is what we have:

- Cell (1,1) is a string array with five characters
- Cell (1,2) is a $1 \times 1$ double array
- Cell (2,1) is a string array with five characters
- Cell (2,2) is a $2 \times 2$ double array
Cell Array

Addressing cell arrays:

- \( x\{1,1\} \) refers to the string ‘hello’

- \( x\{2,2\} \) refers to the 2 double array \([1 \ 3; \ 7.1 \ 9]\)

- What does \( x\{2,2\}(2, 1) \) refer to?

- What does \( x\{2,1\}(3) \) refer to?

- What does \( x\{2,1\}(1, 2) \) refer to?
Cell Array

See the m-file `createCellArray.m`.

This takes the file `studentscores.txt` and creates a cell array `studentRecordCell`.