What’s New for MATLAB with R2012b through R2014b

Gerardo Hernández Correa
Senior Application Engineer
Agenda

- New Graphics System
- Working with Data
  - Importing
  - New Datatypes
  - Big Data
- Managing and Testing Code
- Sharing Apps and Custom Toolboxes
- Hardware Support
New Graphics System

Overview

- New look
  - New default colormap and line colors
  - Anti-aliased fonts and lines
  - Subtler grid lines

- Easier to customize
  - Graphics objects now behave like other MATLAB objects
  - Support dot-notation to access and change properties

Data easier to interpret

```
>> p = plot(x,y);
>> p.Color = 'red';
```
New Graphics System

Features

- Rotatable tick labels
- Automatic updating of datetime tick labels
- New visualization functions
  - histogram
  - animatedline
- Multiple colormaps per figure
- Multilingual text and symbols
- User interfaces with tab panels
Agenda

- New Graphics System
- Working with Data
  - Importing
  - New Datatypes
  - Big Data
- Managing and Testing Code
- Sharing Apps and Custom Toolboxes
- Hardware Support
Import Tool

- Interactive import of delimited and fixed-width text files

- Improved handling of:
  - Numbers
  - Text
  - Dates

- Define rules for handling nonnumeric values

- Automatically generate MATLAB code (scripts and functions) to automate the process
Additional Support for Importing Data

- Access online data *(webread)*
  - JSON, CSV, and image data
- Faster data import from text files
- Import data directly as *categorical* or *datetime*
- Read and write data from network-connected devices *(tcpclient)*
**table**

- New fundamental data type
- For mixed-type tabular data
  - Holds both data and metadata
- Supports flexible indexing
- Built-in functionality (merge, sort, etc.)
Categorical Arrays

- New fundamental data type
- For discrete non-numeric data
  - Values drawn from a finite set of possible values ("categories")
- More memory efficient than a cell array of strings
- Can be compared using logical operators
  - Similar to numeric arrays
Date and Time Arrays

- `datetime` for representing a point in time
- `duration`, `calendarDuration` for representing elapsed time
- Same data type for computation and display
  - Add, subtract, sort, compare, and plot
  - Customize display formats
  - Nanosecond precision
- Support for time zones
  - Accounts for daylight saving time
Big Data Capabilities in MATLAB

Memory and Data Access
- 64-bit processors
- Memory Mapped Variables
- Disk Variables
- Databases
- Datastores

Programming Constructs
- Streaming
- Block Processing
- Parallel-for loops
- GPU Arrays
- SPMD and Distributed Arrays
- MapReduce

Platforms
- Desktop (Multicore, GPU)
- Clusters
- Cloud Computing (MDCS on EC2)
- Hadoop
Access Big Data

- Easily specify data set
  - Single text file (or collection of text files)
- Preview data structure and format
- Select data to import using column names
- Incrementally read subsets of the data

```matlab
airdata = datastore('*.csv');
airdata.SelectedVariables = {'Distance', 'ArrDelay'};
data = read(airdata);
```
Analyze Big Data

**mapreduce**

- Use the powerful MapReduce programming technique to analyze big data
  - **mapreduce** uses a **datastore** to process data in small chunks that individually fit into memory
  - Useful for problems with complex grouping, or when intermediate results do not fit in memory

- **mapreduce** on the desktop
  - Increase compute capacity (Parallel Computing Toolbox)
  - Analyze big database tables (Database Toolbox)
  - Access data on HDFS to develop algorithms for use on Hadoop

- **mapreduce** with Hadoop
  - Run on Hadoop using MATLAB Distributed Computing Server
  - Deploy applications and libraries for Hadoop using MATLAB Compiler

---

```
****************************
*                  MAPREDUCE. PROGRESS                  *
********************************
Map 0%   Reduce 0%
Map 20%  Reduce 0%
Map 40%  Reduce 0%
Map 60%  Reduce 0%
Map 80%  Reduce 0%
Map 100% Reduce 25%
Map 100% Reduce 50%
Map 100% Reduce 75%
Map 100% Reduce 100%
```

---
Agenda

- New Graphics System
- Working with Data
  - Importing
  - New Datatypes
  - Big Data
- Managing and Testing Code
- Sharing Apps and Custom Toolboxes
- Hardware Support
Source Control Integration

- Manage your code from within the MATLAB Desktop

- Leverage modern source control capabilities
  - GIT and Subversion integration in Current Folder browser

- Use Comparison Tool to view and merge changes between revisions
Unit Testing Framework

- Write, run, and analyze tests for your MATLAB programs
  - Define how each test checks values and responds to failures
  - Setup and restore system before and after tests
  - Run tests individually or grouped into a test suite

- Supports either script-based, function-based or object-based unit tests
MATLAB Unit Test Framework

- Based on x-Unit (SUnit, JUnit, CppUnit …)

- Parts
  - Test case
  - Test suite
  - Test runner
function tests = TestTableIndexing
    tests = functiontests(localfunctions);
end

function testDotIndex (testCase)
    T = readtable('patients.csv');
    byDot = T.Age;
    byIndex = T(:,3);
    verifyEqual(testCase,byDot,byIndex);
end

function testNameIndex (testCase)
    T = readtable('patients.csv');
    byName = T(:,'Age');
    byIndex = T(:,3);
    verifyEqual(testCase,byName,byIndex);
end
function tests = TestTableIndexing
    tests = functiontests(localfunctions);
end

function testDotIndex (testCase)
    T = readtable('patients.csv');
    byDot = T.Age;
    byIndex = T[:,3];
    verifyEqual(testCase,byDot,byIndex);
end

function testNameIndex (testCase)
    T = readtable('patients.csv');
    byName = T(:, 'Age');
    byIndex = T(:, 3);
    verifyEqual(testCase, byName, byIndex);
end
function tests = TestTableIndexing
    tests = functiontests(localfunctions);
end

function testDotIndex (testCase)
    T = readtable('patients.csv');
    byDot = T.Age;
    byIndex = T(:,3);
    verifyEqual(testCase,byDot,byIndex);
end

function testNameIndex (testCase)
    T = readtable('patients.csv');
    byName = T(:, 'Age');
    byIndex = T(:, 3);
    verifyEqual(testCase,byName,byIndex);
end
function tests = TestTableIndexing
    tests = functiontests(localfunctions);
end

function testDotIndex (testCase)
    T = readtable('patients.csv');
    byDot = T.Age;
    byIndex = T(:,3);
    verifyEqual(testCase,byDot,byIndex);
end

function testNameIndex (testCase)
    T = readtable('patients.csv');
    byName = T(:, 'Age');
    byIndex = T(:,3);
    verifyEqual(testCase,byName,byIndex);
end
function tests = TestTableIndexing
    tests = functiontests(localfunctions);
end

function testDotIndex (testCase)
    T = readtable('patients.csv');
    byDot = T.Age;
    byIndex = T(:,3);
    verifyEqual(testCase,byDot,byIndex);
end

function testNameIndex (testCase)
    T = readtable('patients.csv');
    byName = T(:, 'Age');
    byIndex = T(:, 3);
    verifyEqual(testCase, byName, byIndex);
end
>> runtests TestTableIndexing.m
>> runtests TestTableIndexing.m

Running TestTableIndexing

..
>> runtests TestTableIndexing.m

Running TestTableIndexing
..
Done TestTableIndexing

ans =

1x2 TestResult array with properties:

Name
Passed
Failed
Incomplete
Duration

Totals:
2 Passed, 0 Failed, 0 Incomplete.
0.089878 seconds testing time.
Why use Unit Testing?

- Testing saves development time
- Testing makes development more enjoyable
  - Your time is spent making things, not fixing things.
  - Fewer nasty surprises and opportunities to make mistakes
- Framework is not trivial, but easily learnable
  - Well worth the effort if you maintain software.
Agenda

- New Graphics System
- Working with Data
  - Importing
  - New Datatypes
  - Big Data
- Managing and Testing Code
- Sharing Apps and Custom Toolboxes
- Hardware Support
MATLAB Apps

- **Apps are self-contained tools, typically with a UI**
  - Accessed in MATLAB Apps gallery
  - Included in many MATLAB Products
  - Can be authored by MATLAB users

- **Apps from the MATLAB Community**
  - Found on MATLAB File Exchange
  - Download and install into the MATLAB Apps gallery

- **Making your own apps**
  - Create single file for easier install and distribution
Packaging and Sharing MATLAB Apps

- Automatically includes all necessary files
- Documents required products
- Creates single installation file for easy distribution and installation into the MATLAB apps gallery
Toolbox Packaging

- Package your toolbox as a single installer file
  - Contains all of the code, data, apps, documentation, and examples
  - Checks for dependent files and automatically includes them
  - Documents required products

- Included folders and files automatically appear on path when installed

- View details and uninstall toolboxes with Manage Custom Toolboxes dialog box
Agenda

- New Graphics System
- Working with Data
  - Importing
  - New Datatypes
  - Big Data
- Managing and Testing Code
- Sharing Apps and Custom Toolboxes
- Hardware Support
Connecting to Low Cost Hardware

Engineer’s computer
MATLAB algorithm or Simulink model

Data I/O
Ethernet / USB / Bluetooth
MATLAB Hardware Support Packages

Low Cost Hardware
R2014b
Android Sensors
Lego EV3
Arduino
R2014a
Raspberry Pi
Webcam

Target
Simulink Hardware Support Packages

Get Support Package Now