Loading data v1.0

30th March 2015

Chris Rodgers – christopher.rodgers@cardiov.ox.ac.uk
Lucian Purvis – lucian.purvis@cardiov.ox.ac.uk

An example of loading and using data is shown in `example_loadViennaData`.

1. Script only

Data is loaded into the Matlab workspace using the dicomTree class. It checks for dicom files in the chosen directory and sorts them into a structure. It then saves a cache in a .mat file in the local /Temp/ directory to allow faster subsequent loading.

The first step is to make a Spectro.dicomTree object with the following code:

```
dt = Spectro.dicomTree('dir',datadir);
```

where the variable `datadir` is a string of the directory containing the data. A useful option is to make the search recursive, so that dicom files can be found if the series are separated by folder. Recursion is included by adding the name/value pair 'recursive'/true to the call i.e.

```
dt = Spectro.dicomTree('dir',datadir, 'recursive',true);
```

The details of the dicomTree structure can be found in section 4. The dicomTree object can then be used to find the correct study, series and instance.

This is done using various search functions. The two that are most commonly used are `searchForSeriesInstanceNumber` and `searchForSeriesNumber`. These are called using

```
matched = dt.searchForSeriesInstanceNumber(seriesNumber,
instanceNumber);
```

and

```
matched = searchForSeriesNumber(seriesNumber);
```

The first is used to load a particular instance from a series, and the second is used to load a whole series. More detail on the search function is given in section 5.

The spec object is then loaded from a Spectro.dicomTree series or instance structure.

```
ss = Spectro.Spec(matched);
```

This object contains all the fields needed to be inputted in `amares.AMARES`.

2. GUI Method for loading Spectro.Spec

The function `guiLoad` is a simplified version of `guiPromptSpectroPlotCsi` (see the next section). It loads data into the Matlab workspace using a GUI, and requires the base directory as input.

When the script is run, a GUI will open. The data is organised into study, series and instance. Dropping through the folders will allow the either a series or an instance to be selected using enter or right click (see Fig. 1). After the data is chosen, it is then loaded using the same dicomTree code as above.

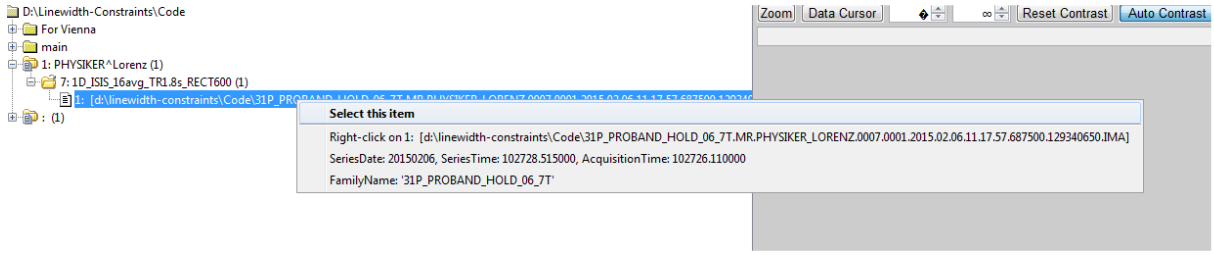

Figure 1: Selecting the data

3. GUI method for loading a Spectro.PlotCsi obj

The main function used is `guiPromptSpectroPlotCsi`. It can be called by:

```
guiPromptSpectroPlotCsi datadir
```

The data to load is chosen in the same way as `guiLoad`. Then the plotCsi obj is made, and a figure is opened (see Fig. 2). If they are in the same directory, the three localizers that were on screen when the protocol was added to the queue on the scanner are shown. If the protocol was CSI, the grid of voxels is overlaid on the images.

Several functions can be run from this figure. Any voxel can be chosen, and by clicking the “Quick plot spectrum” button, the real and imaginary spectra are loaded in a new figure. By pressing the ϕ button and clicking the table, the spectra can be phased by dragging the mouse across the figure. “Quick plot FID” plots the real and imaginary FIDs. The “ALL spectra” button loads the spectra and updates when a new voxel is chosen. By default this is the absolute spectra, but the real or imaginary parts can be chosen in the “QuickPlot” menu. “Quick fit” runs AMARES for the chosen voxel and outputs the results.

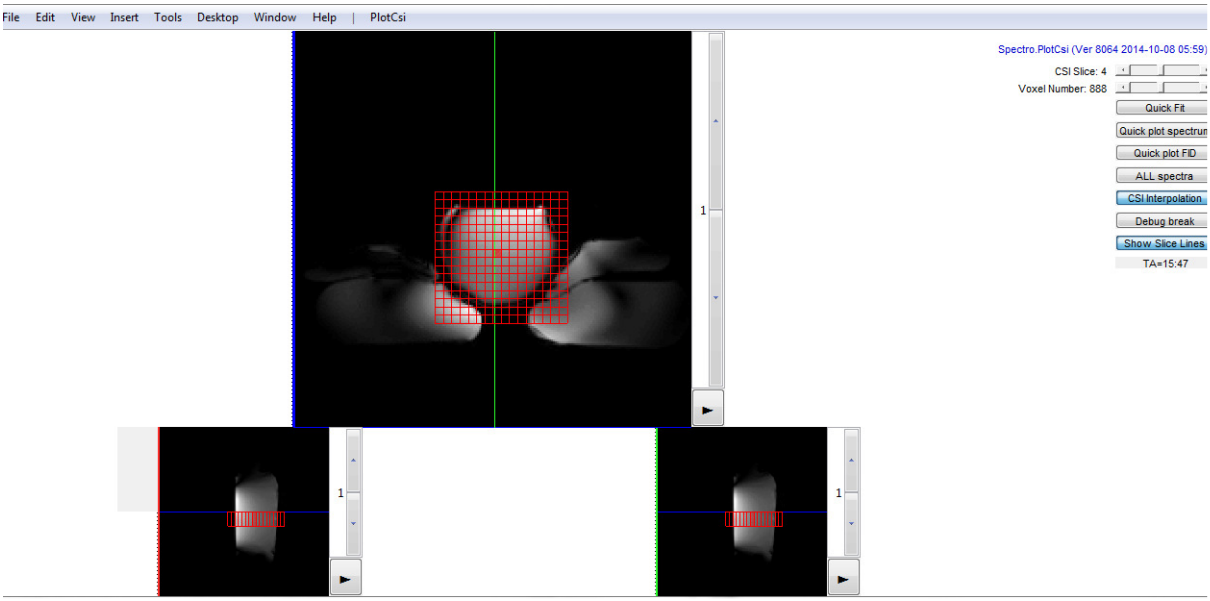

Figure 2: A Spectro.PlotCsi figure

4. Stucture of dicomTree

dicomTree scans the directory to provide information about studies, series and instances. It has the following structure.

1. dicomTree
    a. path
    b. study
        i. StudyInstanceUID
        ii. StudyID
        iii. StudyDescription
        iv. Series
            1. SeriesInstanceUID
            2. SeriesNumber
            3. SeriesDescription
            4. SeriesDate
            5. SeriesTime
            6. Instance
                a. SOPInstanceUID
                b. InstanceNumber
                c. Filename

5. dicomTree search functions

The search functions given in section 1 are simplified to make them easier to call. All dt.searchXXX functions call the main `dt.search` function e.g. for `searchForSeriesInstanceNumber`:

```
matched =
dt.search('target','instance','return','instance','query',@(inst,ser,stu)
ser.SeriesNumber == seriesNum && inst.InstanceNumber == instanceNum);
```

In this example the target is a single instance (rather than a series). The 'query' field is used to choose which instances are returned. It takes the form of an anonymous function with the instance, series and study inputted. The correct series and instance numbers can then be chosen.

The anonymous function is powerful, because it allows for any of the fields in the dicomTree structure to be queried.

If a whole series should be loaded, the anonymous query function should have variables @(ser, stu) only. Similarly, if a whole study is to be loaded, it should have variables @(stu) only.