ChiToolbox Cookbook

Alex Henderson

Work in progress…

Basics

What is it?

The ChiToolbox is a collection of MATLAB files to read, and assist in the interpretation of, hyperspectral data, from a range of chemical analysis techniques.

Where to get it?

https://github.com/AlexHenderson/ChiToolbox/

How do I to install it?

https://alexhenderson.github.io/ChiToolbox/

What are the requirements to run it?

Most of the code requires a version of MATLAB later than R2014a. There are some aspects that require certain toolboxes, in addition to the default installation.

What are the licence restrictions?

The ChiToolbox is released under the GNU GPL 3.0 licence, a copy of which is included in the file collection, but which may also be found at https://www.gnu.org/licenses/gpl-3.0.en.html

Other licensing options are available. Please contact Alex Henderson (alex.henderson@manchester.ac.uk) for more information.

Reporting problems, special requests, etc.

If you find a bug in the Toolbox, please report it using the Issues section of the ChiToolbox website https://github.com/AlexHenderson/ChiToolbox/issues. Note that you will need a (free) account on GitHub to add or comment on issues. This is to prevent spam.

If you would like to suggest an enhancement to the Toolbox, please use this interface and flag the entry accordingly.

Structure

Structural elements

The Toolbox is written in an object oriented programming (OOP) style. This means that there are fundamental aspects that are built upon to generate specialist code, depending on the requirement.

The main structural elements are **ChiSpectrum**, **ChiSpectralCollection** and **ChiImage**. These are extended, as required, to create more specialised classes such as ChiRamanSpectrum that has capabilities that are specific to Raman spectroscopy data and would not be suitable, or meaningful, for other spectroscopies or spectrometries.

Usually there is no need to create one of these structural elements, but they are available if there is a desire to extend the Toolbox to accommodate new or different techniques.

Copying/cloning

One of the 'features' of MATLAB is the way it manages its memory. Usually if a user creates a variable, say a vector, and wants to create a copy of that, they use code such as

```
a = [1,2,3,4,5];
b = a;
```

Here `b` is a new variable that contains a duplicate of `a`. Modifying `a` will not modify `b`, and *vice versa*.

However, MATLAB's OOP design means that any class inheriting from a Handle behaves differently. Now, if one tries an equivalent copying procedure we get a different result.

```
c = <some Handle class>;
d = c;
```

Since `d` and `c` are now both Handle classes, they point to the same memory. Any change to `c` is automatically reflected in `d`, and *vice versa*. Essentially they both point to the same object.

Therefore, to circumvent this, all ChiToolbox classes (which are Handle classes under the hood) have a `clone()` function buried inside.

```
e = ChiSpectrum(___);
f = e.clone();
```

Now `e` and `f` are independent duplicates. Any change to one will not be reflected in the other. They are separate entities.

Note that member functions in MATLAB classes can also be called using a different syntax. These two lines are identical:

```
f = clone(e);
f = e.clone();
```

Assignment

Many function in the ChiToolbox produce a modified version of the original. Where it is reasonable to expect that a data set could be modified, and still be of an equivalent data type, it is possible to modify the variable *in situ*. However, if an output variable is requested, the input variable is first cloned, then the function is performed on the clone. In this way the original input variable is not modified.

For example, if we were to perform sum normalisation on a ChiSpectrum we have two options.

```
x = ChiSpectrum(values, yvalues);
x.sumnorm();           % Option 1
y = x.sumnorm();       % Option 2
```

Using Option 1, the values of `x` are changed internally. Using Option 2, `x` is not modified. `y` is created as a clone of `x` and then `y` is normalised. Note that if this code snippet is executed (all three lines) then line 2 (Option 1) will result in `x` being changed internally. Then when line 3 is executed it is working on the normalised version of `x`. This means that in this scenario, both `x` and `y` will be normalised versions of the original `x` variable.

Where we have functions that will generate an output that is not compatible with the input variable, this clone-then-execute functionality is not available and one or more output variables are required. For example, performing principal components analysis (PCA) on a ChiSpectralCollection results in a ChiSpectralPCAOutcome object. Since this is a type incompatible with a ChiSpectralCollection, we must provide a variable to hold the results of the calculation.

```
x = ChiSpectralCollection(xvalues,yvalues);
x.pca();          % incorrect usage resulting in an error
y = x.pca();      % correct usage
```

Opening a dataset from a file

The ChiToolbox supports a range of spectroscopies and spectrometries stored in various file formats.

File formats

The ChiToolbox can read the following file formats natively:

- Agilent (FTIR)
    - Single FTIR images (.seq)
    - Mosaicked FTIR images (.dmt)
- Biotof (ToFSIMS)
    - Spectral files (.dat)
    - Retrospective image files (.xyt)
- Bruker (FTIR)
    - Multiple spectra exported as a .mat file
- Ionoptika (ToFSIMS)
    - Retrospective image files exported in the .h5 format
- Mettler Toledo (FTIR)
    - Spectra exported to ASCII (.asc) files
- Renishaw (Raman)
    - WiRE Version 4 (.wdf). Single spectra, multiple spectra or images
- Thermo Fisher Scientific GRAMS SPC (Generic)
    - Data stored in .spc files

Each of these file formats has its own reader to import the data in the file(s) into a MATLAB variable. However, there is a shortcut to this. If one uses the **ChiFile** function, the file will be interrogated to determine the most appropriate reader and that will be enacted accordingly. Therefore, it is often not required to remember the reader name and simply to use ChiFile.

Opening a file

There are three approaches to open and read a file, depending on the user's circumstances:

- Pass a filename to the ChiFile function;
- Allow the user to search for the file interactively;
- Pass a specially formatted Microsoft Excel spreadsheet to ChiFile.

The easiest method is to search interactively, however users often wish to record data processing steps in a MATLAB script. Here it is useful to supply a list of filenames and have ChiToolbox open them sequentially, or alternatively, all at once.

The approach using an Excel spreadsheet will be covered in a later section.

Example

Say we wish to open an Agilent FTIR image data set. These data sets consist of a number of individual files. A single FPA image tile consists of three files: example.bsp, example.dat and example.seq.

In order to import this data set we can use the following syntax:

```
dataset = ChiFile('location\of\file\example.seq');
```

Where `'location\of\file\example.seq'` is the name and location of the file on disc. Here `dataset` will contain all the relevant information pertaining to the FTIR image tile.

If the user does not have the filename to hand, it is possible to use a similar syntax:

```
dataset = ChiFile();
```

MATLAB will then prompt the user to select a file of their choice.

Other file types can be opened in an analogous manner.

File types obtained

Opening a file from disc will result in one of three types of variable being created: ChiSpectrum, ChiSpectralCollection or ChiImage. The actual filetype will depend on the underlying spectroscopy or spectrometry data in the file. For example, if a Renishaw file (say renishaw1.wdf) contains a single Raman spectrum then:

```
dataset = ChiFile('location\of\file\example.seq');
```

will result in `dataset` being of type ChiRamanSpectrum. If renishaw2.wdf contains more than one spectrum, `dataset` will be of type ChiRamanSpectralCollection. Likewise, if renishaw3.wdf contains a hyperspectral image, `dataset` will be of type ChiRamanImage. Each of these data set types has some properties in common, while other properties and methods are specific to the underlying data structure. A full list of properties and methods is given in Appendix XXX.

Opening multiple files

In some cases it is possible to open a number of files at the same time. This can be advantageous when the user wishes to generate a collection of spectra that should be treated as a set. Some instrument vendors store different types of data (spectra, images etc.) in different file formats with different filename extensions. Other vendors use the same filename extension for all types of data. Therefore it is often difficult to know in advance what any given file will contain. Therefore the following approach has been take with the ChiToolbox.

If multiple filenames are passed to ChiFile (or the associated underlying file readers) the Toolbox will first determine if each of them is readable. If the files are of incompatible types, for example if we have a Renishaw .wdf file and a Biotof .dat file, an error will be reported.

If all files are of the same spectroscopy/spectrometry they will be imported in the order they appear in the list of filenames. Once the first file has been imported, the Toolbox moves on to the second file. If the spectral length is different from the previously imported file the data will be truncated to the maximum overlap and the data points interpolated, in order to generate a matrix. The Toolbox then moves on to the next filename, truncating and interpolating as required. Subsequent files are treated in the same manner. This results in a Chi*SpectralCollection (where the * depends on the spectroscopy/spectrometry of the files) where the data is in a consistent matrix and the x-axis values match the maximum common overlap of the data and interpolated (if required) to match the first file imported.

Therefore reading many files, each containing a single spectrum will result in a ChiSpectralCollection.

If some of the files contain a single spectrum, while others contain multiple spectra, all the spectra in all the files will be represented in the resultant ChiSpectralCollection. These will be in the order of the filenames presented and where a filename contains multiple spectra, in the order that those spectra are contained within the file.

For example, assume we have four files: file1, file2, file3 and file4. file1, file2 and file4 contain single spectra, while file3 contains three spectra in the order a, b, c. If the filenames are passed in the order file1, file2, file3, file4 we will produce a ChiSpectralCollection with spectra in the order:

file1
file2
file3a
file3b
file3c
file4

If some of the files contain hyperspectral images then the result will depend on the underlying spectroscopy/spectrometry. For example, Agilent Mosaic files can hold many thousands of spectra (one spectrum per pixel). If a number of these files were to be passed to ChiFile and each spectrum extracted and concatenated, we would soon run out of memory. Therefore, if an image of a data type considered to be 'large' is encountered in a list of filenames, only the first file will be imported and a warning issued.

Should the user wish to import multiple large images, each should be loaded individually.

Where the image size is likely to be relatively small, the spectrum at each pixel is extracted and the data is returned in a ChiSpectralCollection in the same manner as above. The pixels are arranged in the order top left to bottom left, and then left to right across the image. Therefore the first pixel is top left and the last pixel is bottom right.

Pre-processing

Normalisation

Sum normalisation

First we add together the intensity values of each data point in a spectrum. Then we divide each data point in that spectrum by the total. This results in the sum of the spectrum being unity. Each spectrum is normalised independently.

```
mydata_normalised = mydata.sumnorm();
```

Advanced topics

This section will be completed later since it is envisioned most users will wish to become familiar with the basic functionality of the Toolbox, before embarking on more complicated journeys. Please get in touch if there is something you feel the Toolbox should be able to do, or there is a direction you would like to see it extended.

Building a data set from scratch

See the constructors for ChiSpectrum, ChiSpectralCollection and ChiImage.

Credits

Standing on the shoulders of giants, the following code is included in the ChiToolbox.
In no specific order:

Perceptually uniform colormaps

Version 1.3.1
Copyright (c) 2016, Ander Biguri
Available from https://uk.mathworks.com/matlabcentral/fileexchange/51986-perceptually-uniform-colormaps

error_ellipse

Version 1.0
Copyright (c) 2004, AJ Johnson
Available from https://uk.mathworks.com/matlabcentral/fileexchange/4705-error-ellipse

GSTools

Version 1.3
Copyright (c) 2006, Kris De Gussem
Available from https://uk.mathworks.com/matlabcentral/fileexchange/9938-gstools

shadedErrorBar

Version 1.65 (downloaded from Github 2018-02-07, commit 1ea286c)
Rob Campbell
Available from https://uk.mathworks.com/matlabcentral/fileexchange/26311-raacampbell-shadederrorbar
See also https://github.com/raacampbell/shadedErrorBar

mksqlite

Version 2.5
Martin Kortmann and Andreas Martin
Available from http://mksqlite.sourceforge.net/
See also https://github.com/AndreasMartin72/mksqlite

sql_object

Version 1.1
Andreas Martin
Available from https://uk.mathworks.com/matlabcentral/fileexchange/58433-using-sqlite-databases-via-objects

cividis

Jamie R. Nuñez, Christopher R. Anderton, Ryan S. Renslow
Available from https://github.com/pnnl/cmaputil
See also https://arxiv.org/abs/1712.01662

sgolayfilt

Version 1.0
Andrew D. Horchler
Available from https://github.com/horchler/sgolayfilt

RMieS

Version 5.0
Paul Bassan
Available from https://github.com/GardnerLabUoM/RMieS

cluster-toolbox
Version 2.0
Laboratory for Bioanalytical Spectroscopy (Biospec)
Available from https://github.com/Biospec/cluster-toolbox-v2.0

getSubclasses
Oleg Komarov
Downloaded from Github 2018-08-10, (commit 95dc300, Jun 30, 2016)
Available from https://github.com/okomarov/getSubclasses

GUI Layout Toolbox
Version 2.3.2.0
David Sampson and Ben Tordoff
Available from https://uk.mathworks.com/matlabcentral/fileexchange/47982-gui-layout-toolbox

DataHash
Version 1.5.0.0
Jan
Available from https://uk.mathworks.com/matlabcentral/fileexchange/31272-datahash

GetFullPath
Version 1.7.0.0
Jan
Available from https://uk.mathworks.com/matlabcentral/fileexchange/28249-getfullpath

m2html
Revision: 1.5 Date: 2005/05/01 16:15:30
Guillaume Flandin
Available from https://www.artefact.tk/software/matlab/m2html/

ImportOpus
Received: 24 July 2017
Jacob Filik
For general availability, please contact Jacob Filik at jacob.filik@diamond.ac.uk

ME-EMSC
Johanne Solheim, Evgeniy Gunko, Dennis Petersen and Achim Kohler
Downloaded from GitLab 2019-02-18, (commit cc752da1)
Available from https://gitlab.com/BioSpecNorway/me-emsc

Thresholding Tool
Version 1.6.0.1
Robert Bemis
Interactively select intensity level for image thresholding.
Available from https://uk.mathworks.com/matlabcentral/fileexchange/6770-thresholding-tool

ENVI file reader/writer
version 1.2.0.0
Felix Totir
Mini-toolbox for reading and writing ENVI data and header files (including complex). Available from https://uk.mathworks.com/matlabcentral/fileexchange/27172-envi-file-reader-writer

read_envihdr
version 1.0.0.0

Jaroslaw Tuszynski
Robust reader of ENVI header files.
Available from https://uk.mathworks.com/matlabcentral/fileexchange/38500-read_envihdr

Glossary

SIMS	Secondary ion mass spectrometry
Raman	Raman spectroscopy
FTIR	Fourier transform infrared spectroscopy
OOP	Object oriented programming

Appendix XXX

Sorry, nothing here yet ☺