**IRootLab - A MATLAB toolbox for vibrational spectroscopy**

**Manual**

Updated on 21/Jul/2017



This document is licensed under a
[Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](http://creativecommons.org/licenses/by-nc-sa/3.0/).

**Table of contents**

[1. Introduction 2](#_Toc488428317)

[1.1. Overview 2](#_Toc488428318)

[1.2. This manual 2](#_Toc488428319)

[1.3. Conventions 3](#_Toc488428320)

[1.3.1. Formatting styles 3](#_Toc488428321)

[1.3.2. Terminology and abbreviations 3](#_Toc488428322)

[1.3.3. Citing IRootLab 3](#_Toc488428323)

[2. Setup 4](#_Toc488428324)

[2.1. Installation 4](#_Toc488428325)

[2.2. Starting IRootLab 4](#_Toc488428326)

[2.2.1. Windows only 4](#_Toc488428327)

[2.2.2. All platforms 4](#_Toc488428328)

[2.2.3. System requirements 4](#_Toc488428329)

[2.2.4. Platform-specific binaries 4](#_Toc488428330)

[2.3. Learning resources 4](#_Toc488428331)

[3. Quick start tutorial 5](#_Toc488428332)

[3.1. Open and visualize data (Figure 2) 5](#_Toc488428333)

[3.2. Simple analysis – differences between mean spectra (Figure 3) 5](#_Toc488428334)

[3.3. Principal components analysis (PCA) (Figure 4) 6](#_Toc488428335)

[4. Graphical user interfaces (GUIs) 7](#_Toc488428336)

[4.1.1. Objtool 7](#_Toc488428337)

[4.1.2. MATLAB Code generation with objtool 9](#_Toc488428338)

[4.1.3. Supported dataset file types within objtool 10](#_Toc488428339)

[4.2. Mergetool 11](#_Toc488428340)

[5. Class library 13](#_Toc488428341)

[6. Various topics 15](#_Toc488428342)

[6.1. IRootLab setup file 15](#_Toc488428343)

[6.2. MATLAB command sheet 15](#_Toc488428344)

[6.3. Dataset from existing MATLAB variables 15](#_Toc488428345)

[7. References 16](#_Toc488428346)

# Introduction

## Overview

IRootLab[1] is a framework for vibrational spectroscopy data analysis in MATLAB. It provides pattern recognition, biomarker extraction, imaging, pre-processing, feature extraction, and other methods, directed to vibrational spectroscopy (Fourier-Transform InfraRed (FTIR) and Raman).

The framework includes a class function library, and Graphical User Interfaces (GUIs) to import and analyse data (objtool, mergetool, sheload). The objtool GUI can be also used as a MATLAB code generator. A demonstration page can be opened by typing browse\_demos.

IRootLab is Free/Libre and Open-Source software, released under the LGPL licence.

Official website: <http://irootlab.googlecode.com>

IRootLab contains over 200 hierarchically organized object classes representing concepts, methods and algorithms. The two most important class branches are *datasets* and *blocks*. Data analyses are built by training/using *blocks* with *datasets*. *Block* is a very general concept encompassing pre-processing methods, classifiers, visualizations, reports, complex analysis sessions etc. Figure 1 illustrates different types of blocks with code examples using the IRootLab library.

**Datasets and blocks**

Dataset

Block

Modified
dataset

Dataset

Visualization
Block

Figure

Training dataset

Block

**1- Training stage**

Trained
block

Dataset

(same or other)

Trained
block

**2- Use stage**

Modified
dataset

**Blocks that don’t require training**

**Visualization blocks**

**Blocks that require training**

**Code example**

% Rubberband baseline correction

block = pre\_bc\_rubber();

ds02 = block.use(ds01);

% Class means visualization

block = vis\_means();

figure;

block.use(ds01);

% PCA-LDA

block = cascade\_pcalda();

block.blocks{1}.no\_factors = 10; % PCA factors

block = block.boot();

block = block.train(ds01);

ds02 = block.use(ds01);

**Code example**

**Code example**

Figure – Illustration of different types of blocks with code examples using the IRootLab library.

## This manual

This manual explains the structure of IRootLab and discusses some of the most important details about the toolbox. Basic MATLAB knowledge is expected from the reader (MATLAB current directory, workspace, how to specify a MATLAB vector). Aspects of object-oriented programming (OOP) were concentrated in Section 5 and avoided elsewhere to spare readers who are mostly interested in using the GUIs.

The “official” IRootLab documentation remains the source code itself, and the Doxygen-generated documentation from there (<http://bioph.lancs.ac.uk/irootlabdoc>). However, the official documentation is for reference, rather than for learning. This manual provides selected, concentrated and relevant information.

## Conventions

### Formatting styles

The following styles are used throughout this manual:

name of a file

MATLAB code

MATLAB code

Some option available in a graphical interface

Note

### Terminology and abbreviations

|  |  |
| --- | --- |
| **class** | The word “class” may be used meaning either* *data class* (pattern classification), such as “control”, “cancer” etc, or
* *object class* (object-oriented programming).

The meaning should be clear from the context. |
| **group** | = sample/patient/colony/specimen etcA “group” represents all spectra that must be kept together at all times if the dataset is split, because they all belong to the same sample/patient/colony/specimen etc. |
| **observation** | = rowRefers to a row in a dataset matrix. May be a spectrum, but can be also an average spectrum, a set of PCA scores etc |
| **feature** | = data variableRefers to a column in a dataset matrix. May be a wavenumber, a PCA factor etc |
| **no** | “number of observations” = number of rows in a dataset matrix |
| **nf** | “number of features” = number of columns in a dataset matrix |
| **AS** | Analysis Session |
| **SGS** | Sub-dataset Generation Specification |
| **FSG** | Feature Subset Grader |

### Citing IRootLab

If IRootLab works well for you and you use it for your respective study and publication, please cite the following:

* <http://irootlab.googlecode.com> (please mention the website in your manuscript)

[1] Trevisan, J., Angelov, P.P., Scott, A.D., Carmichael, P.L. & Martin, F.L. (2013) "IRootLab: a free and open-source MATLAB toolbox for vibrational biospectroscopy data analysis". Bioinformatics 29(8), 1095-1097. doi: 10.1093/bioinformatics/btt084.

[2] Martin, F.L., Kelly, J.G., Llabjani, V., Martin-Hirsch, P.L., Patel, I.I., Trevisan, J., Fullwood, N.J. & Walsh, M.J. (2010) "Distinguishing cell types or populations based on the computational analysis of their infrared spectra". Nature Protocols 5(11), 1748-1760. doi:10.1038/nprot.2010.133

# Setup

This chapter gives information to get IRootLab running.

## Installation

Download the most recent ZIP file from <http://irootlab.googlecode.com> and extract the file to a directory of your choice.

## Starting IRootLab

### Windows only

1. Open the Windows Explorer
2. If you are starting a new project, create a directory for your project *e.g.*, […]/My Documents/brain\_project
3. Locate the directory created when the ZIP file was extracted
4. Double-click on the file named startup\_windows
5. Change MATLAB Current folder to your own project directory.

### All platforms

1. If you are starting a new project, open Windows Explorer and create a directory for your project *e.g.*, […]/My Documents/brain\_project
2. Start MATLAB
3. Change MATLAB Current folder to the directory containing the IRootLab library extracted from the ZIP file (this directory should contain a file called startup.m)
4. In MATLAB command line, enter startup
You should see a list of directories being added to the path, then a welcome message.
5. Change MATLAB Current folder to your own project directory.

**Note**

Alternatively, you can use MATLAB "Set path..." (add with subdirectories). This will keep you from having to run startup every time you start MATLAB. However, make sure you remove the old directories from the MATLAB path when you download a new version of IRootLab.

### System requirements

IRootLab will run under MATLAB, which is ported for Windows, Linux, and MacOS. The oldest MATLAB version tested was **r2007b**.

MATLAB toolboxes:

* (optional) MATLAB Parallel Computing Toolbox (PCT)
* MATLAB Wavelet Toolbox for Wavelet de-noising

### Platform-specific binaries

This information is only relevant if you intend to use SVM or the MySQL database.

* SVM classifier (LibSVM): LibSVM was successfully compiled for Windows 32-bit/Windows 64-bit; Linux 32-bit/64-bit.
* MySQL connector (mYm): mYm was currently compiled Windows 32-bit; Linux 32-bit/64-bit. Linux 64-bit: libmysqlclient.so.16 and libmysqlclient.so.18.

## Learning resources

* Tutorials and this manual: [http://irootlab.googlecode.com](http://irootlab.googlecode.com/) . There are currently about 8 step-by-step analysis tutorials on-line covering various topics.
* Demo files: type browse\_demos
* On-line documentation (reference): <http://bioph.lancs.ac.uk/irootlabdoc>
* Access to reference from MATLAB command line: help2 <filename>
* Context-sensitive help from IRootLab GUIs: press F1

# Quick start tutorial

This simple tutorial aims to get you started with using objtool, which is the main GUI. It will use Ketan’s Brain ATR data[3], which is included in the toolbox.

## Open and visualize data (Figure 2)

1. Run the steps on Section to start IRootLab
2. At MATLAB command line, enter browse\_demos
3. Scroll down the page
4. Click on the “LOAD\_DATA\_KETAN\_ATR” hyperlink
5. Click on the “objtool” hyperlink to launch **objtool**
6. Click on Apply new blocks/more actions
7. Click on **vis**
8. Click on **Class means**
9. Click on **Create, train & use**

**4**

**5**

**6**

**7**

**8**

**9**

Figure – Tutorial – Open and visualize data in objtool

## Simple analysis – differences between mean spectra (Figure 3)

At this part, we will compare the mean spectra of the “Glioblastoma” and “Astrocytoma” classes against the “Normal” class. The latter is used as a reference class to investigate changes in infrared absorption in the two former.

1. Click on **pre**
2. Click on **Subtract mean of a reference class**
3. Click on **Create, train & use**
4. A parameters window appears; click on **OK** (“1” is already the first class of the dataset; “Normal”)
5. Click on **ds01\_refmean01**
6. Click on **vis**
7. Click on **Class means**
8. Click on **Create, train & use**

**10**

**12**

**11**

**15**

**16**

**17**

**14**

Figure – Tutorial – Differences between mean spectra

## Principal components analysis (PCA) (Figure 4)

This part of the analysis investigates whether PCA can segregate the data classes.

1. Click on **ds01**
2. Click on **fcon**
3. Click on **Principal component analysis**
4. Click on **Create, train & use**
5. A parameters window appears; click on **OK** to accept the default options
6. Click on **ds01\_pca01**
7. Click on **vis**
8. Click on **2D Scatterplot**
9. Click on **Create, train & use**
10. On the window that appears, enter [1, 2, 3] as the Indexes of variables to plot. Click on **OK**

**19**

**21**

**20**

**18**

**24**

**25**

**26**

**23**

**27**

Figure – Tutorial – PCA

# Graphical user interfaces (GUIs)

IRootLab has four GUIs callable from MATLAB command line (Table 1).

Table 1 – IRootLab GUIs.

|  |  |
| --- | --- |
| **GUI name** | **Purpose** |
| objtool | This is the most important GUI. Used for data analysis, general manipulation of objects, and code generation |
| mergetool | Tool to assemble dataset from a multiple spectral files containing one spectrum each |
| irootlab | Launch other GUIs, get help, and view auto-generated MATLAB code |
| sheload | Loads dataset from online database[4] |

The GUIs read and write variables directly into the MATLAB workspace.

### Objtool

The main **objtool** “missions” are:

* Create objects or load datasets
* Boot, train and use blocks
* Execute objects “more actions”
* Automatically generate MATLAB code from any of the above

The look of **objtool** and its visible options will vary depend on selected items and toggled buttons. The most notorious change is the colour of the window background, which reflects which class is selected in the left panel[[1]](#footnote-1).

The objects listed in **objtool** are variables present in MATLAB workspace, created by **objtool** itself, by **mergetool**, by **sheload** or by other means (*e.g.*, user scripts). Every processing step leaves existing objects untouched and creates new ones, adding to the new variable name a suffix that reminds of the operation that has just been carried out. For example, in Figure 5 **ds01\_std01** is the result of the standardization of dataset **ds01**; then, **ds01\_std01\_lda01** is the result of applying LDA to **ds01\_std01**. **objtool** is divided in three areas, as seen in Figure 5.

**Left panel/**

**Classes panel**

**Middle panel/**

**Objects panel**

**Right panel**

Figure – Screenshot of objtool

#### Left panel

The left panel is a selector for what to show in the middle panel. It also shows how many object of each class exists in the MATLAB workspace. A trick helps identify objects that were just created by the last operation: when a number is surrounded by asterisks (“\*\*”), it means that new objects of that class have just appeared as a result of the last user operation.

#### Middle panel

This panel shows the existing objects of the class selected on the left panel. The buttons at the top vary slightly depending on the selected class: for irdata, the Load… and Save as… buttons appear, whereas for the other classes, the New… button appears instead.

#### Right panels

There are three different panels on the right. The panel to appear is selected by one of three toggle buttons on the top right.

The first of the right panels is the Apply new blocks/more actions panel (Figure 6). The Applicable blocks list shows only the classes that accept the object(s) that are selected in the middle panel as input[[2]](#footnote-2). There are three options when creating a block in this way: Create only; Create & train; and Create, train & use. The More actions list is dependent on the selected object in the middle panel. “More actions” are simple actions available to some objects only (mainly to extract some information from an object).

The Existing blocks panel (Figure 7) shows all blocks already in the workspace that are applicable to the object(s) that is (are) selected in the middle panel. The Train and Use operations will pass whichever is selected in the middle panel as input to the block.

Finally, the Object properties panel shows MATLAB-generated description for the selected object. Figure 8 shows this panel when a dataset is selected in the middle panel. Some objects have associated additional information. For example, a cascade block (class block\_cascade\_base) will show the descriptions of its component blocks, apart from its own description.

**A**

**B**

Figure – **(A)** Apply new blocks/more actions panel when the Dataset class is selected in the left panel; **(B)** Apply new blocks/more actions panel when a class other than a dataset is selected in the left panel. The main difference is set of buttons that appear only for the Dataset class.



Figure – Existing blocks panel in objtool

**Number of observations/(data rows)**

**Number of features/(data columns)**

**Number of groups (*e.g.*, patients)**

**Number of data classes**

**Data matrix, dimension no×nf**

**Classes, column vector of size no**

**Class labels**

**Group codes, (*e.g.*, patient names)**

**Dataset properties**

Figure – Object properties panel showing the properties of a dataset. Some properties are described. **irdata** is circled, which is the object class name for a dataset. All properties are explained in the reference, which is accessible by typing **help2 irdata** at MATLAB command window (or directly at the source file by typing **edit irdata.m**). The same applies to other object classes.

### MATLAB Code generation with objtool

A major feature of objtool is its ability to generate MATLAB code. Every analysis step done in **objtool** is recorded as MATLAB code into a file named **irr\_macro\_[nnnn].m**, which can be opened by entering **ircode\_edit** at MATLAB command window.

This resource was created to help with script/function/class writing, and it is highly efficient in doing so. For example, the demo files were almost completely created in this way, and a substantial part of IRootLab itself was automatically coded by **objtool**.

Code generation can easily help users to repeat the same analysis sequence on a different dataset, or batch-process several datasets.

### Supported dataset file types within objtool

Table 2 shows the file types currently supported in **objtool**. Directory demo/sampledata has some examples of data files of all types.

Table - file types supported in **objtool.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **File type** | **Read?** | **Write?** | **Com****ment** | **Related OOP-class file** |
| *TXT (basic)* | Yes | Yes |  | [dataio\_txt\_basic.m](http://delasmina/irootdoc/html/dataio__txt__basic_8m.html) |
| *TXT ("Pirouette-like")* | Yes | Yes |  | [dataio\_txt\_pir.m](http://delasmina/irootdoc/html/dataio__txt__pir_8m.html) |
| *TXT (IRootLab)* | Yes | Yes | ; Only TXT format that saves all dataset properties | [dataio\_txt\_iroot.m](http://delasmina/irootdoc/html/dataio__txt__iroot_8m.html) |
| *TXT (IrootLab with labelled classes)* | Yes | Yes | ; Fills the “classes” column with the class labels, not numbers | [dataio\_txt\_iroot2.m](http://delasmina/irootdoc/html/dataio__txt__iroot_8m.html) |
| *MAT* | Yes | Yes | MATLAB compressed binary: short file size; opens fast | dataio\_txt\_mat.m |
| *OPUS image maps* | Yes | No | FPA images only, not point maps | [dataio\_opus\_nasse.m](http://delasmina/irootdoc/html/dataio__opus__nasse_8m.html) |
| *TXT (LibSVM format)* | No | Yes | LIB | [dataio\_opus\_libsvm.m](http://delasmina/irootdoc/html/dataio__opus__nasse_8m.html) |

The TXT (IRootLab) format is the only text format that saves all the properties of a dataset. There are two variants of the TXT (IRootLab) format. The first variant (Figure 10) saves the classes as integer numeric values starting at zero, and has a “classlabels” row in the file containing the labels respective to each number; the second variant (Figure 11; named IRootLab with labelled classes when you “Save as…” in objtool) does not have the “classlabels” row, and fills in the “classes” column directly with the labels instead. We recommend the second variant, as it is easier to edit and avoid confusion. You can choose between these variants when you “Save as…” in objtool.

The TXT (basic) can only store spectra and their associated classes.

The TXT (Pirouette-like) stores the wavenumbers, sample codes, spectra, and classes.



Figure - Basic and Pirouette-like text formats



Figure - IRootLab text format (first variant)



Figure – IRootLab with labelled classes text format (second variant)

## Mergetool



Figure – Screenshot of mergetool.

mergetool is a tool to merge several spectral files into a new dataset. The new dataset is created in the MATLAB workspace and can be immediately viewed in objtool. Table 3 shows the spectral file types currently supported in **mergetool**. A spectral file type auto-detection button is available.

Table - Spectral file types supported in **mergetool.**

|  |
| --- |
| **Spectral File type** |
| *Pirouette .DAT* |
| *OPUT binary* |
| *Wire TXT (Renishaw)* |
| *Wire (2016) TXT(Renishaw)* |
| *Diane's "Another FTIR" (2017) TXT* |
| *BWSpec CSV* |

Usage is nearly straightforward. Only the meaning of the Group code trimming dot (right-to-left) parameter may be slightly obscure. This parameter is a way to tell what part of the file name will be considered to be the code for a data group (*i.e.*, sample/patient etc). It is common for many spectra be taken from the same sample/group. The spectral files are often sequentially named by the spectrum acquisition software (*e.g.*, OPUS (Bruker Optik GmbH) creates files such as **sample.0**, **sample.1**, sample.2 etc). Within **mergetool**, the dots (“.”) in the file names are considered to be “dividers”. So, the Group code trimming dot (right-to-left) specifies how many dots to skip (from right to left) to compose the group code; the group code will start at the beginning of the file name and stop just before the specified dot (Figure 12). The **Perform checks** button gives a report that, among other things, gives a preview of what a group code will be before the files are imported.

For image maps, the **Image height** needs to be specified. Naturally, this number must be able to divide the total number of spectral files (such quotient will be the image width). The image map is mounted as follows. The first spectrum in the directory (sorted by name) will be the bottom left pixel of the image (Figure 13).

**Image map**

**First file in directory
(sorted by name)**

**Second file**

**. . .**

**Vertically**

**Horizontally**

**First file in directory
(sorted by name)**

**Second file**

**.**

**.**

 **.**

Figure – **mergetool** file reading sequence to compose image map, depending on how the Pixel mapping option is set.

# Class library

IRootLab contains more than 200 object classes hierarchically organized. The classes are data models and implementations of analysis methods, concepts and algorithms. Parts, concepts or techniques that are common to many methods have been implemented separately (*e.g.*, **sgs**, **fsg**). This provides flexibility and avoids code repetition. The most important class branches, with their corresponding descriptions, are shown in Table 1. A full listing can be obtained by running the demo file demo\_classes\_html.

Table 4 – Some important branches and leaves from the IRootLab class tree.

|  |  |
| --- | --- |
| **Class (.m file name)** | **Description** |
|        Dataset (irdata) | Datasets for analysis can be point spectra or image maps. The only difference between these is that image maps have the height property set to a value > 0.Classifiers output a special kind called an “estimation dataset” (class estimato). These datasets have their classes unset and need to be processed by a decider.A final special dataset is generated by hierarchical clustering. In this case, the variables represent cluster numbers (class irdata\_clus). |
|        Block (block) | This is the base class for all processing operations.A block is an object containing the boot(), train(), and use() methods, and a few more properties to define how the block class will be handled by objtool. |
|           Miscellanea (blmisc) | Miscellanea blocks lack a common identity; they have been grouped for convenience to display grouped in objtool. |
|           Visualization (vis) | Figure and report (HTML) generation |
|           Cascade block (**block\_cascade\_base**) | Cascade blocks encapsulate a sequence of blocks.Cascade blocks can mimic the behaviour of linear transformation blocks (fcon\_linear; such as PCA) if it contains one or more such blocks. It has all the properties that a fcon\_linear block has, however its valid functioning will depend on the component blocks. The loadings matrix is calculated by multiplying the loadings matrix of successive component blocks.When a cascade block is about created in the GUI, it will call the parameters GUIs of its component blocks in sequence.Customized/frequent sequences can be created by inheriting the block\_cascade\_base class.  |
|           Decider (decider) | This is an abstraction of class decisions based on the posterior probabilities calculated by a classifier. Classifiers generate an estimato dataset which is later processed by a decider. If the highest per-class posterior probability is below the decider decisionthreshold property, it will “refuse to decide”, assigning to the data row a class of -1 instead of a valid one. |
|           Classifier (clssr) | A classifier is a block whose use() method outputs an estimato dataset. |
|              Ensemble (aggr) | Various classifier ensemble architecture |
|              Incremental (clssr\_incr) | Classifiers capable of incremental learning. Such classifiers train0 method uses one data row at a time to modify the classifier internal structure. |
|                 Fuzzy Rule-Based Model (frbm) | Contains a set of parameters that allow various fuzzy classifier to be set up, including eClass0 and eClass1[5] |
|           Feature Extraction (fext) | IRootLab uses Guyon *et al*.’s division of feature extraction between feature selection and feature construction methods[6]. |
|              Feature Construction (fcon) | Methods that combine variables (linearly or non-linearly) into new ones |
|                 Hierarchical Clustering (clus\_hca) | Clustering was placed under feature construction because it creates a new dataset whose features represent cluster numbers. |
|                 Linear Transformation (fcon\_linear) | Blocks with loadings vectors to transform the input dataset |
|                 Measure (fcon\_mea) | Measure over the entire row (*e.g.*, norm, maximum, minimum etc), creates output that contains one variable only (*i.e.*, the measuse). |
|              Feature Selection (fsel) | Feature selection. This block is not trainable. It contains a fixed vector of feature indexes to be selected from the input to use(). Feature selection methods are found under “Analysis Session” (as class) |
|           Pre-processing (pre) | This branch contains methods that are classically called “pre-processing” methods in the literature, such as de-noising, baseline correction and normalization [7–9]. |
|           Analysis Session (as) | These blocks usually perform complex and potentially time-demanding analyses inside their use(), and the use() output is a irlog (not a dataset as is the output of most blocks) from where many results can be extracted and/or visualized. |
|        Sub-dataset Generation Specs (sgs) | This class is an abstraction of sub-dataset generation. Sub-dataset generation is needed nearly everywhere in data analysis. Situations include obtaining train and test datasets, cross-calculation of scores, and as part of algorithms (*e.g.* bagging and boosting). sgs centralizes the task of calculating the rows to be extracted from a dataset to generate sub-datasets. |
|        Feature Subset Grader (fsg) | fsg centralizes the task of calculating a “grade” (a measure of for a sub-set of features). This class was created to be used in feature selection algorithms.  |
|        Peak Detector (peakdetector) | This class centralizes the work of detecting peaks from a spectrum, loadings vector, histogram etc. |
|        Vector Comparer (frbm) | This class is used to compare vectors with the same number of elements (paired tests). The main application is for comparison between classifiers (see report\_sovalues\_comparison.m). |
|        Log (irlog) | This is a general information/results container. It stores information whose format is unsuitable to be stored by a irdata object. Typically generated by as::use(). |

# Various topics

## IRootLab setup file

IRootLab configuration for colour scheme, font, scatterplot markers etc is stored as MATLAB global variables. These variables are automatically saved by **objtool** in a file called irootlab\_setup.m, which can be changed at will. Each project directory has its own **irootlab\_setup.m** file. You can force the creation of this file by entering setup\_write at MATLAB command window.

To open the setup file, enter edit irootlab\_setup at MATLAB command window. To make changes to this file immediately available in MATLAB environment, simply execute the file (*e.g.*, by pressing the F5 key at MATLAB editor; or entering **irootlab\_setup** at MATLAB command window). Table 5 describes some setup options. For a complete reference to IRootLab global variables, please check **assert\_all.m**

Table 5 - some global/**irootlab\_setup.m** setup variables.

|  |  |
| --- | --- |
| **Variable name** | **Purpose** |
| SCALE | Controls the relative size of text, markers, and line widths.**Tip:** Increase the SCALE if you want a figure to be small in a publication. For example, if SCALE = 1 looks fine when you create a full-width figure panel, you should use SCALE = 2 for a half-width panel. |
| COLORS | Defines a colour sequence to represent different data classes. Each colour is coded as a 3-element vector representing its [Red, Green, Blue] (RGB) composition. Each values ranges from 0 to 255. For example, [255, 0, 0] is red, [0, 0, 0] is black, and [255, 255, 0] is yellow (red+green). **Tip:** There is a good website at <http://www.colourlovers.com> with lots of nice colour palettes! |
| MARKERS | Sequence of markers (triangles, circles, squares etc) to represent different data classes in scatterplots.**Tip:** For a list of available marker symbols, enter help plot at MATLAB command window. |
| FONT | Font name |
| FONTSIZE | Font size |

## MATLAB command sheet

|  |
| --- |
| **IRootLab-specific** |
| v\_x2ind([1650, 1450, 1080], ds01.fea\_x) | Converts wavenumbers to feature indexes for dataset ds01. |
| save\_as\_png([], "filename.png”, 300) | Saves current figure as a PNG file named filename.png at 300 dpi resolution. |
| edit\_ircode | Opens the current auto-generated macro MATLAB editor |
| help2 | Opens main page of documentation in browser |
| help2 irdata | Opens help in browser for file irdata.m |
| objtoolmergetoolsheload | Opens respective GUIs |
| ds01.classlabels’ | Shows class labels from dataset ds01 |
| **MATLAB basics** |
| 1:5 | Same as [1, 2, 3, 4, 5] |
| 1:2:21 | Same as [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21] |
| xlim([900, 1800]) | Sets current figure x-axis limits to 1800-900 cm-1 region. Note that the lower value comes first in the command. |
| ylim([0, 1]) | Sets current figure y-axis limits between 0 and 1. |
| var(ds01\_meanc01\_pca01.X, 1)’ | Displays Variances of principal components (assuming that the dataset name is ds01\_meanc01\_pca01) |

## Dataset from existing MATLAB variables

If you already use MATLAB for data analysis, you are likely to have at some stage the dataset variables, such as X and classes separately. The example below shows how to pack such existing variables into an IRootLab dataset.

% Assumes that these variables exist:

% - X of dimension [no]x[nf]

% - classes of dimension [no]x[1]

ds = irdata();

ds.X = X;

ds.classes = classes;

ds = ds.assert\_fix(); % Automatically fills in the classlabels property with defaults

Also, check the file called demo\_import\_fisheriris.m; this example performs some analysis on Fisher’s Iris dataset that is shipped with MATLAB and performs a somewhat more advanced than the example above.

# References

[1] J. Trevisan, P. P. Angelov, A. D. Scott, P. L. Carmichael, and F. L. Martin, “IRootLab: a free and open-source MATLAB toolbox for vibrational biospectroscopy data analysis.,” *Bioinformatics (Oxford, England)*, pp. 1–2, Mar. 2013.

[2] F. L. Martin, J. G. Kelly, V. Llabjani, P. L. Martin-Hirsch, I. I. Patel, J. Trevisan, N. J. Fullwood, and M. J. Walsh, “Distinguishing cell types or populations based on the computational analysis of their infrared spectra,” *Nat. Prot.*, vol. 5, no. 11, pp. 1748–1760, Jan. 2010.

[3] K. Gajjar, L. D. Heppenstall, W. Pang, K. M. Ashton, J. Trevisan, I. I. Patel, V. Llabjani, H. F. Stringfellow, P. L. Martin-Hirsch, T. Dawson, and F. L. Martin, “Diagnostic segregation of human brain tumours using Fourier-transform infrared and/or Raman spectroscopy coupled with discriminant analysis,” *Analytical Methods*, vol. 5, pp. 89–102, 2013.

[4] J. Trevisan, P. P. Angelov, I. I. Patel, G. M. Najand, K. T. Cheung, V. Llabjani, H. M. Pollock, S. W. Bruce, K. Pant, P. L. Carmichael, A. D. Scott, and F. L. Martin, “Syrian hamster embryo (SHE) assay (pH 6.7) coupled with infrared spectroscopy and chemometrics towards toxicological assessment,” *Analyst*, vol. 135, no. 12, pp. 3266–3272, Dec. 2010.

[5] P. P. Angelov and X. Zhou, “Evolving Fuzzy-Rule-Based Classifiers From Data Streams,” *IEEE T. Fuzzy Syst.*, vol. 16, no. 6, pp. 1462–1475, Dec. 2008.

[6] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature Extraction - Foundations and Applications*. New York: Springer, 2006.

[7] R. L. Somorjai, M. Alexander, R. Baumgartner, S. Booth, C. Bowman, A. Demko, B. Dolenko, M. Mandelzweig, A. E. Nikulin, N. J. Pizzi, E. Pranckeviciene, and P. Summers, RZhilkin, “A data-driven, flexible machine learning strategy for the classification of biomedical data,” in *Artificial Intelligence Methods And Tools For Systems Biology*, vol. 5, W. Dubitzky and F. Azuaje, Eds. Dordrecht: Springer Netherlands, 2004, pp. 67–85.

[8] R. M. Jarvis and R. Goodacre, “Genetic algorithm optimization for pre-processing and variable selection of spectroscopic data,” *Bioinformatics*, vol. 21, no. 7, pp. 860–868, Apr. 2005.

[9] J. Trevisan, P. P. Angelov, P. L. Carmichael, A. D. Scott, and F. L. Martin, “Extracting biological information with computational analysis of Fourier-transform infrared (FTIR) biospectroscopy datasets: current practices to future perspectives.,” *The Analyst*, vol. 137, no. 14, pp. 3202–15, Jul. 2012.

1. See irobj.color property. [↑](#footnote-ref-1)
2. See block::inputclass property. [↑](#footnote-ref-2)