Import and Export of Spectra Files Vignette for the R package hyperSpec

Claudia Beleites (cbeleites@units.it)
CENMAT, DMRN, University of Trieste

February 16, 2010

Reproducing the Examples in this Vignette

The source code of this vignette including the spectra files are available as .zip file at *hyperSpec*'s home page: http://r-forge.r-project.org/projects/hyperspec/FileIO.zip

Contents

1. Introduction

This document describes how spectra can be imported into *hyperSpec* objects. Some possibilities to export *hyperSpec* objects as files are mentioned, too.

The most basic funtion to create *hyperSpec* objects is `new ("hyperSpec")` (section 2). It makes a *hyperSpec* object from data already in R's workspace. Thus, once the spectra are imported into R, conversion to *hyperSpec* objects is straightforward.

However, *hyperSpec* comes with predifined import functions for different data formats. This document divides the discussion into dealing with ASCII files (section) and binary file formats. If data export for the respective format is possible, it is discussed in the same sections. As sometimes the actual data written by the spectrometer software exhibits peculiarities, *hyperSpec* offers specialized import functions. These are in general named after the data format followed by the manufacturer (e. g. read.ENVI.Nicolet).

Finally, we give overview lists of the directly supported file formats: sorted by manufacturer (appendix A) and by file format (appendix B)

2. Creating a hyperSpec object with `new`

If the data is in R's workspace, a *hyperSpec* object is created by:

```
spc <- new ("hyperSpec", spc, wavelength, data, label)}
```

With the arguments:

`spc`	the spectra matrix
`wavelength`	the wavelength axis vector
`data`	the extra data (possibly already including the spectra matrix in column `spc`)
`label`	a list with the proper labels. Do not forget the wavelength axis label in `$.wavelength` and the spectral intensity axis label in `$spc`.

Thus, once your data is in R's workspace, creating a *hyperSpec* object is easy. I suggest wrapping the code to import your data and the line joining it into a *hyperSpec* object by your own import function. You are more than welcome to contribute such import code to *hyperSpec*. Secion 4 gives an example of how to wrap up the code.

3. ASCII files

3.1. ASCII Files

Currently, *hyperSpec* provides four functions for general ASCII data import and export:

`read.txt.long` import long format ASCII files, i. e. one intensity value per row

`read.txt.wide` import wide format ASCII files, i. e. one spectrum per row

`write.txt.long` export long format ASCII files

`write.txt.wide` export wide format ASCII files

The import functions immediately return a *hyperSpec* object.

Internally, they use `read.table`, a very powerful ASCII import function.

R supplies another ASCII import function, `scan`. `scan` imports numeric data matrices and is faster than `read.table`, but cannot import column names. If your data does not contain a header or it is not important and can safely be skipped, you may want to import your data using `scan`.

3.2. ASCII files with samples in columns

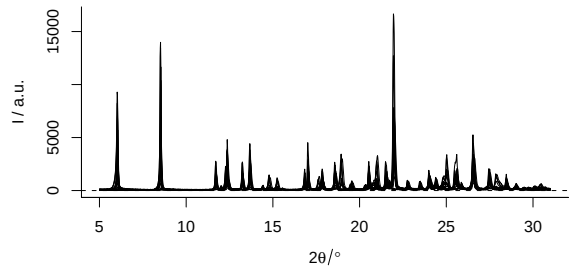
Richard Pena from Pierre Fabre asked about importing another ASCII file type:

> *Triazine5_31.txt file corresponds to X ray powder diffraction data (Bruker AXS). The native files data ".raw" are read with EVA software then they are converted into .uxd file with the File Exchange software (Bruker AXS). The .uxd file are opened with Excel software and saved as .txt file, csv file (ChemoSpec) or xls.*
> *The first and following columns corresponds to the angle diffraction and the intensity values of samples respectively.*

This file thus differs from the ASCII formats discussed above in that the samples are actually in columns whereas *hyperSpec* expects them to be in rows. The header line gives the name of the sample. Import is straightforward, just the spectra matrix needs to be transposed to make a *hyperSpec* object:

```
> file <- read.table ("txt.t/Triazine 5_31.txt", header = TRUE, dec = ",", sep = "\t")
> triazine <- new ("hyperSpec", wavelength = file [,1], spc = t (file [, -1]),
+                  data = data.frame (sample = colnames (file [, -1])),
+                  label = list (.wavelength = expression (2 * theta / degree),
+                                spc = "I / a.u."))
> triazine
hyperSpec object
   25 spectra
   2 data columns
   1759 data points / spectrum
wavelength: 2 * theta/degree [numeric] 5.0025 5.0173 ... 31.0042
data:  (25 rows x 2 columns)
   1. sample:  [factor] rng  DIV1208200 DIV1208300 ... VCA0106703
   2. spc: I / a.u. [AsIs matrix x 1759] rng  16 18 ... 16664

> plot (triazine)
```



4. Writing your own Import Function

This section gives examples of how to write import functions.

4.1. Example 1: read.txt.PerkinElmer

The raw spectra of the `flu` data set (see also the respective vignette) are in Perkin Elmer's ASCII file format, one spectrum per file. The files are completely ASCII text, with the actual spectra starting at line 55.

We need a function that automatically reads all files specified by a pattern, such as `*.txt`. In order to gain some speed, the spectra matrix should be preallocated after the first file is read.

A short examination of the files (`flu*.txt` in directory `txt.PerkinElmer`) reveals that the actual spectrum starts at line 55, after a line containing `#DATA`. As no other information of the files is to be extracted, it is easier to skip the first 54 lines instead of searching for the line after `#DATA`.

The resulting import function is saved as an .R file, that can be easily `source`d into an R session:

read.txt.PerkinElmer.R

```
read.txt.PerkinElmer <- function (files = "*.txt", skip = 54, ...) {
  ## find all files
  files <- Sys.glob (files)

  ## read the first file
  buffer <- matrix (scan (files [1], skip = skip, ...), ncol = 2, byrow = TRUE)

  ## first file gives the wavelength vector
  wavelength <- buffer [, 1]

  ## preallocate the spectra matrix: one row per file x as many columns as the
  ## first file has
  spc <- matrix (ncol = nrow (buffer), nrow = length (files))

  ## the first file's data goes into the first row
  spc [1, ] <-  buffer [, 2]

  ## now read the remaining files
  for (f in seq (along = files)[-1]) {
    buffer <- matrix (scan (files [f], skip = skip, ...),
                      ncol = 2, byrow = TRUE)

    ## check whether they have the same wavelength axis
    if (! all.equal (buffer [, 1], wavelength))
      stop (paste(files [f], "has different wavelength axis."))

    spc [f, ] <- buffer[, 2]
  }

  ## finally: make the hyperSpec object
  new ("hyperSpec", wavelength = wavelength, spc = spc,
       label = list (.wavelength = expression (lambda[fl] / nm),
         spc = "I / a.u."))
}
```

Note how labels giving the correct units (e. g. for axis labels) are set. The label with the special name `.wavelength` corresponds to the wavelength axis, all data columns should have a label with the same name. The spectra are always in a data column called `spc`.

Thus,

```
> source ("read.txt.PerkinElmer.R")
> read.txt.PerkinElmer ("txt.PerkinElmer/*.txt")

hyperSpec object
   6 spectra
   1 data columns
   181 data points / spectrum
wavelength: lambda[fl]/nm [numeric] 405.0 405.5 ... 495
data:  (6 rows x 1 columns)
   1. spc: I / a.u. [AsIs matrix x 181] rng  27.15000 32.34467 ... 677.4947
```

imports the spectra.

4.2. Example 2: read.uxd.Bruker

Bruker X-ray diffraction data may be converted to .uxd files, a long ASCII format. Again, multiple files need to be imported into one *hyperSpec* object.

5. Binary file formats

5.1. Matlab Files

Matlab files can be read and written using the package *R.matlab*[**?**], which is available at CRAN and can be installed by `install.packages ("R.matlab")`.

```
spc.mat <- readMat ("spectra.mat")
```

will import

If the .mat file was saved with compression, the additional package *Rcompression* is needed. It can be installed from omegahat:

```
install.packages("Rcompression", repos = "http://www.omegahat.org/R")
```

See the documentation of *R.matlab* for more details and possibly needed further packages.

`readMat` imports the .mat file's contents as a list. The variables in the .mat file are properly named elements of the list. The *hyperSpec* object can be created using `new`, see 2 (p. 2).

Again, you probably want to wrap the import of your matlab files into a function.

5.1.1. Writing Matlab Files

R.matlab's function `writeMat` can be used to write R objects into .mat files. To save an *hyperSpec* object `x` for use in Matlab, you most likely want to save:

- the wavelength axis as obtained by `wl (x)`,
- the spectra matrix as obtained by `x [[]]`, and
- possibly also the extra data as obtained by `x$.`.
- as well as the axis labels `labels (x)`.
- Alternatively, `x$.` yields the extra data together with the spectra matrix.

However, it may be convenient to transform the saved data according to how it is needed in Matlab. The functions `as.long.df` and `as.wide.df` may prove useful for reshaping the data.

5.2. ENVI Files

ENVI files are binary data accompanied by an ASCII header file. *hyperSpec*'s function `read.ENVI` can be used to import them.

As we experienced missing header files (Bruker's Opus software frequently produces header files without any content), the data that would usually be read from the header file can also be handed to `read.ENVI` as a list. The help page gives details on what elements the list should contain.

5.2.1. ENVI Export

Use package *caTools* or *rgdal* with GDAL for writing ENVI files.

5.3. spc Files

Thermo Galactic's .spc file format can be read by `read.spc`.

A vast variety of sub-formats is available. *hyperSpec*'s import function does not support the *old file format* that was used before 1996. In addition, no test data with *w planes* was available – thus the import of such files could not be tested. If you come across such files, please contact the package maintainer.

These functionality is tested using Thermo Galactic's example files:

```
> ## old format files stop with an error:
> old <- paste ("spc", c ('CONTOUR.SPC', 'DEMO 3D.SPC', 'LC DIODE ARRAY.SPC'), sep = "/")
> for (f in old)
+   try (read.spc (f))
> ## all other files should be good for import
> other <- setdiff (Sys.glob ("spc/*.[sS][pP][cC]"), old)
> for (f in other){
+   spc <- read.spc (f)
+
+   if (is (spc, "hyperSpec"))
+     cat (f, ": ", nrow (spc), " spectrum(a), ", nwl (spc), " data pts / spc.\n", sep = "")
+   else
+     cat (f, ": list of ", length (spc), " spectra, ",
+          paste (range (sapply (spc, nwl)), collapse = " - "),
+          " data pts / spc\n", sep = "")
+ }
```

```
spc/BARBITUATES.SPC: list of 286 spectra, 4 - 101 data pts / spc
spc/barbsvd.spc: list of 286 spectra, 4 - 101 data pts / spc
spc/BENZENE.SPC: 1 spectrum(a), 1842 data pts / spc.
spc/DRUG SAMPLE_PEAKS.SPC: list of 6 spectra, 80 - 253 data pts / spc
spc/DRUG SAMPLE.SPC: list of 400 spectra, 2 - 254 data pts / spc
spc/FID.SPC: 1 spectrum(a), 8192 data pts / spc.
spc/HCL.SPC: 1 spectrum(a), 8361 data pts / spc.
spc/HOLMIUM.SPC: 1 spectrum(a), 901 data pts / spc.
spc/IG_BKGND.SPC: 1 spectrum(a), 4096 data pts / spc.
spc/IG_MULTI.SPC: 10 spectrum(a), 4096 data pts / spc.
spc/IG_SAMP.SPC: 1 spectrum(a), 4645 data pts / spc.
spc/KKSAM.SPC: 1 spectrum(a), 751 data pts / spc.
spc/POLYR.SPC: 1 spectrum(a), 1844 data pts / spc.
spc/POLYS.SPC: 1 spectrum(a), 1844 data pts / spc.
```

```
spc/SINGLE POLYMER FILM.SPC: 1 spectrum(a), 1844 data pts / spc.
spc/SPECTRUM WITH BAD BASELINE.SPC: 1 spectrum(a), 1400 data pts / spc.
spc/TOLUENE.SPC: 1 spectrum(a), 801 data pts / spc.
spc/TUMIX.SPC: 1 spectrum(a), 1775 data pts / spc.
spc/TWO POLYMER FILMS.SPC: 1 spectrum(a), 1844 data pts / spc.
spc/XYTRACE.SPC: 1 spectrum(a), 3469 data pts / spc.
```

Like `read.ENVI`, `read.spc` allows to use pre-defined header values. See the help for

6. Manufacturer-Specific Discussion of File Import

6.1. Renishaw Raman Spectrometers

Renishaw's Wire software comes with an file format converter. This program can produce a long ASCII format, .spc or .jdx files.

An optimized import function for the ASCII files is available: `scan.txt.Renishaw`.

```
> paracetamol <- scan.txt.Renishaw ("txt.Renishaw/paracetamol.txt", "spc")
> paracetamol

hyperSpec object
   1 spectra
   1 data columns
   4064 data points / spectrum
wavelength: Delta * tilde(nu)/cm^-1 [numeric] 96.7865 98.1432 ... 3200.07
data:  (1 rows x 1 columns)
   1. spc: I / a.u. [AsIs matrix x 4064] rng  299.229 317.041 ... 49052.2
```

The converter also offers to write .spc files. We experienced that this conversion is not fully reliable: maps were saved as depth profile, loosing all spatial information. Also, an evenly spaced wavelength axis was produced, although this was de-selected in the converter. We therefore recommend using the ASCII format.

A. I/O Functions by Manufacturer

B. I/O Functions by File Format

B.1. Manufacturer Specific Import Functions

Many spectrometer manufacturers provide a function to export their spectra into ASCII files. The functions discussed above are written in a very general way, and are highly customizable. I recommend wrapping these calls with the appropriate settings for your spectra format in an import function. You may also consider contributing such import filters to *hyperSpec*: send me (cbeleites@units.it) the documented code (either .R + .Rd file or Roxygen commented .R).

B.1.1. Renishaw

B.1.2. Bruker FTIR Imaging

We use `read.ENVI` to import IR-Images collected with a Bruker Hyperion spectrometer with OPUS software. As mentioned above, the header files are frequently missing. We found the necessary information to be:

```
> header <- list (samples = 64 * no.images.in.row,
+                 lines = 64 * no.images.in.column,
+                 bands = no.data.points.per.spectrum,
+                 `data type` = 4,
+                 interleave = "bip")
```

No spatial information is given in the ENVI header (if correctly written). The lateral coordinates can be setup by specifying origin and pixel size for x and y directions. For details please see the help page.

B.1.3. Nicolet FTIR Imaging

Also Nicolet saves imaging data in ENVI files. These files use some non-standard keywords in the header file that should allow to reconstruct the lateral coordinates as well as the spectral axes' units.

There is indication that the position of the first spectrum is recorded in µm, while the pixel size is in mm. Thus a flag *nicolet.correction* is provided that divides the pixel size by 1000. Also here, giving the correct offset and pixel size values as function arguments is possible.

B.1.4. Kaiser Raman Maps

Spectra obtained using Kaiser's Hologram software can be saved either in their own .hol format and imported into Matlab (from where the data may be written to a .mat file readable by *R.matlab*'s `readMat`. Alternatively, Hologram can write ASCII files and .spc files. We found working with .spc files the best option.

hyperSpec provides the function `read.spc.KaiserMap` to easily import spatial collections of .spc files written by Kaiser's Hologram software. The filenames of all .spc files to be read into one *hyperSpec* object can be provided either in a character vector or as a wildcard expression (e. g. "path/to/files/*.spc").

B.2. Creating a hyperSpec Object from Spectra Matrix and Wavelength Vector

Once the data is in R's workspace, a *hyperSpec* object is created by:
`spc <- new ("hyperSpec", spc = spectra.matrix, wavelength = wavelength.vector)`
You will usually give the following arguments:

`spc`	the spectra matrix
`wavelength`	the wavelength axis vector
`data`	the extra data
`label`	a list with the proper labels. Do not forget the wavelength axis label in `$.wavelength` and the spectral intensity axis label in `$spc`.

C. File Import by Format

Format	Manufacturer	Function	see	Notes
ASCII file formats				
long txt		`read.txt.long`	??, p. ??	
long txt	Renishaw (Raman)	`scan.txt.Renishaw`	B.1.1, p. ??	
long txt	Perkin Elmer (Fluorescence)	`read.txt.PerkinElmer`	??, p. ??	Reads multiple files.
uxd	Bruker (X-Ray diffraction)	`scan.uxd.Bruker`	??, p. ??	Reads multiple files.
wide txt		`read.txt.wide`	??, p. ??	
binary file formats				
ENVI		`read.ENVI`	??, p. ??	
ENVI	Bruker (FTIR Imaging)	`read.ENVI`	B.1.2, p. 8	
ENVI	Nicolet (FTIR Imaging)	`read.ENVI.Nicolet`	B.1.3, p. 8	
Matlab	Matlab	`R.matlab::readMat`	5.1, p. 5	
spc		`read.spc`	??, p. ??	
spc	Kaiser (Raman Map)	`read.spc.KaiserMap`	B.1.4, p. 8	Reads multiple files
spc	Renishaw (Raman)	`read.spc`	B.1.1, p. 8	*Not* recommended, see discussion of ASCII files.

D. File Import by Manufacturer

Manufacturer	Format	Function	see	Notes
Manufacturers				
Bruker (FTIR Imaging)	ENVI	`read.ENVI`	B.1.2, p. 8	
Nicolet (FTIR Imaging)	ENVI	`read.ENVI.Nicolet`	B.1.3, p. 8	
Renishaw (Raman)	long txt	`scan.txt.Renishaw`	B.1.1, p. 8	
Perkin Elmer (Fluorescence)	long txt	`read.txt.PerkinElmer`	??, p. ??	Reads multiple files.
Matlab	Matlab	`R.matlab::readMat`	5.1, p. 5	
Kaiser (Raman Map)	spc	`read.spc.KaiserMap`	B.1.4, p. 8	Reads multiple files
Renishaw (Raman)	spc	`read.spc`	B.1.1, p. 8	*Not* recommended, see discussion of ASCII files.
Bruker (X-Ray diffraction)	uxd	`scan.uxd.Bruker`	??, p. ??	Reads multiple files.
General import functions				
	long txt	`read.txt.long`	??, p. ??	
	wide txt	`read.txt.wide`	??, p. ??	
	ENVI	`read.ENVI`	??, p. ??	
	spc	`read.spc`	??, p. ??	