Chemometric Analysis of Bio-Spectroscopic Data in hyperSpec

Claudia Beleites¹,² (Claudia.Beleites@ipht-jena.de), Christoph Krafft², Jürgen Popp²,³, and Valter Sergo¹

¹ CENMAT and Dept. of Industrial and Information Engineering, University of Trieste, Trieste/Italy
² Institute of Photonic Technology, Jena/Germany
³ Institute of Physical Chemistry and Abbe Center of Photonics, University Jena/Germany

Saratov Fall Meeting 2011
Outline

Introduction
  Chemometric Software for Spectroscopic Data Analysis
  Excursion: Literate Programming
  Hyperspectral Data and hyperSpec

Example Work Flows
  Example: Raman Map of Chondrocytes in Cartilage
  Example: Linear Calibration

Graphical User Interaction

Acknowledgements & Contact

References
Software Wish List: Key Problems ...

...for the design of biospectroscopic data analysis software:

- Biological samples may not be rectangular in shape, so are their spatially resolved data sets.
- Non-rectangular measurement grids, e.g.
  - Raman maps with circular laser spot: hexagonal instead of square grid
  - Changing spatial resolution: add high resolution measurements of region of interest to overview measurement
- Hierarchical (clustered) data structures are typical in biospectroscopy: multiple samples per patient, many spectra per sample.
  Validation must use patient-wise splits of the data
Software Wish List: Key Problems

Availability of specialized statistical methods

- Often, the “specialized” method requires just a very small change from the standard method:
  - E.g.: Spectroscopic data sets often do not have normal intensity distributions. Therefore, I like to plot median, 16th and 84th percentiles of the spectra instead of mean ± 1 standard deviation.

- Or could easily be implemented if access was granted to the model components (coefficients, etc.)

⇝ Data analysis environment should be extensible.

- Programming statistical routines is error-prone: programming errors often lead to wrong results rather than the program stopping.
  - Avoid reimplementing existing chemometric methods,
  - and rather use existing statistical method libraries.
Software Wish List: Key Requirements

- Programming environment more powerful than GUI,
- but particular problems are best solved using graphical interaction

⇝ tailored GUIs for specific tasks

- Data analysis is essentially an interactive workflow:

⇝ interpreted programming language best suited.

⇝ We use \texttt{R} [1], a well-tested statistical programming environment [2]

For a short introduction to \texttt{R}, check out the first pages of \texttt{R}'s introduction. Matlab users may be interested in David Hiebeler's comparison tables. \texttt{R} code is marked by blueish background (with prompt “>” and continued prompt “+”), and its output by greenish background.
Excursion: Literate Programming

- Source code and description/interpretation intermingled in one file
- Insert results (incl. graphics) automatically into text
- Automatic report generation
- Optionally show calculation as if entered at command line
  I use this throughout this presentation to show code and results on the same slide
- \LaTeX: Sweave produces pdf (via \LaTeX), html, Openoffice documents

\begin{verbatim}
\begin{figure}
<<fig = TRUE, echo = FALSE, width = 6, height = 4>>=
plot(chondro, "spcmeansd")
@
\caption{The first 50 spectra.}
\label{fig:spectra}
\end{figure}
\end{verbatim}

The mean ± one standard deviation of the 869 spectra are shown in fig. 1.
Hyperspectral Data

= Spectra + Spectra-related Information

Bio-spectroscopic data usually consists of spectra plus extra data accompanying the spectra, e.g.:

- sample number
- patient
- reference diagnoses
- spatial information (location on sample):
  ⇝ spectral maps, images, depth profiles
- temporal information: kinetic studies

⇝ store and use arbitrary information together with spectra
hyperSpec’s Data Structure

Spectra are stored row-wise in a so-called S4 class that ensures a consistent data set. The central part is a data.frame, a table containing columns with the extra data (here spatial coordinates $x$ and $y$) and one special column holding the spectra matrix. Each column of a data.frame can have its appropriate data type: numeric, boolean (logical), categorical (factor), etc. As the spectra are stored in a numeric matrix rather than data.frame columns for each wavelength, fast matrix calculations are used.
hyperSpec’s Data Structure

> print (chondro)

hyperSpec object
  875 spectra
  4 data columns
  300 data points / spectrum
wavelength: Delta * tilde(nu)/cm^-1 [numeric] 602 606 ... 1798
data:  (875 rows x 4 columns)
  1. y: y/(mu * m) [numeric] -4.77 -4.77 ... 19.2
  2. x: x/(mu * m) [numeric] -11.6 -10.6 ... 22.4
  3. clusters: clusters [factor] matrix matrix ... lacuna + NA
  4. spc: I / a.u. [matrix300] 502 500 ... 169

hyperSpec also stores proper labels for plotting and by default logs modifications to the object.
hyperSpec allows convenient handling of spectroscopic data in R. It provides functions to

- read and write spectra: various ASCII formats, .spc, ENVI, .mat
- plot spectra, incl. interactive plots
- do spectroscopic processing of the data
- It does not provide any chemometric data analysis itself. Instead, hyperSpec provides methods to hand over the spectroscopic data to normal R functions, and to take back the results.
Example: Chondrocytes in Cartilage

Raman Map of Cartilage containing two lacunae and chondrocytes.

Excitation wavelength: 633 nm
Exposure time: 10 s per spectrum
Objective: $100 \times$, NA 0.85
Measurement grid: $35 \times 21$ µm,
1 µm step size
Spectrometer: Renishaw InVia

After loading the package into R,

```r
> library (hyperSpec)
```

Renishaw ASCII files are directly supported by hyperSpec:

```r
> chondro <- scan.txt.Renishaw ("chondro.txt", data = "xyspc")
```
Pre-Processing

1. **Smoothing interpolation** to reduce noise and remove uneven wavenumber spacing due to spectrometer geometry:
   ```r
   > chondro <- spc.loess (chondro, seq (602, 1800, 4))
   ```

2. **Automatic linear baseline correction** (see also package baseline [3])
   ```r
   > baselines <- spc.fit.poly.below (chondro)
   Fitting with npts.min = 15
   > chondro <- chondro - baselines
   ```

3. **Intensity normalization**: divide each spectrum by its mean intensity
   ```r
   > chondro <- sweep (chondro, 1, mean, /)
   ```
The minimum spectrum reflects the biochemical composition common to all spectra and is thus suitable for **centering**. However, it is subject to much noise. A good alternative is the 5th percentile spectrum:

```r
> minspc <- apply(chondro, 2, quantile, 0.05)
> chondro <- sweep(chondro, 2, minspc, -)
```

Here's the result of the pre-processing:

```r
> plot(chondro, "spcprctile")
```
Principal Component Analysis

Score maps of the first 2 principal components.

> pca <- prcomp (~ spc, chondro$.)

hyperSpec's "." operator produces the data.frame expected by function prcomp. Join the results of the matrix decomposition with the additional information about the spectroscopic data set:

> scores <- decomposition (chondro, pca$x)
Hierarchical Cluster Analysis

The spectra matrix is extracted by the "[[ ]]" operator.

```
> dist <- dist (chondro [[]])
```

HCA yields nice separation into **matrix**, **lacunae** and **cells**, which is stored as a new column of the data set:

```
> chondro$clusters <- as.factor (cutree (dendrogram, k = 3))
```
I calculate the median, 5\textsuperscript{th} and 95\textsuperscript{th} percentile spectra for each cluster:

```r
> spc <- aggregate(chondro, chondro$clusters, +    quantile, probs = c(.05, .5, .95))
```

```r
> plot(spc, col = cols, +    stacked = ".aggregate", fill = ".aggregate")
```

Note that the matrix contains more lipids, and that the cells have much less collagen compared to the overall composition of the sample.
Plotting Irregular Spectral Maps

- If a map did not use a regular grid, or
- some spectra are missing, or
- you want to inspect a random subset of the map (left graph)

```r
> spc <- sample(chondro, 300)
```

...a Voronoi tessellation may be the plot of choice.

Indexing works directly with **wavenumbers** of typical DNA bands:

```r
> plotvoronoi(spc [, , c(728, 782, 1098, 1240, 1482, 1577)],
+ col.regions = cols, border = NA, points = FALSE)
```
Linear Calibration

Example data set “flu” contains fluorescence spectra of quinine solutions suitable for linear calibration:

```R
> qplotspc(flu) + aes(colour = c) + labs(colour = labels(flu)$c)
> qplotc(flu) + stat_smooth(method = "lm")
```
Graphical User Interfaces

Compared to source code based approaches, GUIs ...

... are much faster for tasks that require user interaction on plots.

... may be faster for occasional users as icons may be easier to remember than commands.

(Command completion/lookup does the same for source code editing.)

... are more restrictive. This speeds up doing exactly what the GUI was designed for, but inhibits other uses.

... cannot be run automatically.

- How to use the advantages without the disadvantage (restriction)?

  Design tailored graphical applets for specific tasks.

  Implementation as modal dialog allows GUI applets to be used like other functions: "interactive functions"
Example: Raman Spike Filter

It is easy to automatically detect suspicious points in Raman spectra. However, the automatically determined points may need some manual adjustment:

```r
> spikes <- spikes.interactive.cb (cartilage, suspicions)
```

Compared to comprehensive graphical data analysis environments, applets tailored to specific tasks are ...

- faster to develop
- faster to use as fewer controls are needed and be optimally placed
IDE integration

- Large data sets may need to be analyzed in a two-step procedure
  1. Set up the data analysis using a (random) subset of the data
  2. Run the analysis on the whole data set, possibly as batch process, on a computer cluster, etc.

- Interactive functions to determine parameters for the final analysis can produce the code for the final data analysis
  - wrapping by IDE: replace interactive function call by code to produce the result (and save the original interactive function call e.g. in a comment)
  - via clipboard: applet pushes code to produce the result into the clipboard
Acknowledgements

- Homepage: hyperSpec.r-forge.r-project.org
- Contact: Claudia.Beleites@ipht-jena.de
- Installation:

```
> install.packages("hyperSpec",
+ repos = "http://r-forge.r-project.org")
```

- Documentation:
  check out the pdf vignettes
References

R Development Core Team.

*R: A Language and Environment for Statistical Computing.*
ISBN 3-900051-07-0.

Kellie B. Keeling and Robert J. Pavur.

A comparative study of the reliability of nine statistical software packages.

Kristian Hovde Liland and Bjørn-Helge Mevik.

*baseline: Baseline Correction of Spectra*, 2011.
R package version 1.0-0.