

Raman Spectra of Chondrocytes in Cartilage: hyperSpec's chondro data set

Claudia Beleites <chemometrie@beleites.de>

CENMAT and DI3, University of Trieste

Spectroscopy · Imaging, IPHT Jena e.V.

June 7, 2013

Reproducing this vignette

The data set and source file of this vignette are available as a zipped archive at *hyperSpec*'s homepage, <http://hyperspec.r-forge.r-project.org/chondrocytes.zip> (ca. 9 MB). The original data file (ca. 31 MB) is far too large to be included in the package.

In order to reproduce the examples by typing in the commands, have a look at the definitions of color palettes used in this document are defined in `vignettes.defs`. Also, the package *hyperSpec* needs to be loaded first via `library(hyperSpec)`.

Contents

1	Introduction	2
2	The Data Set	2
3	Data Import	3
4	Preprocessing	3
4.1	Aligning the Spatial Grid	3
4.2	Spectral Smoothing	5
4.3	Baseline Correction	5
4.4	Normalization	6
4.5	Subtracting the Overall Composition	6
4.6	Outlier Removal by Principal Component Analysis (PCA)	6
5	Hierarchical Cluster Analysis (HCA)	7
6	Plotting a False-Colour Map of Certain Spectral Regions	9
7	Saving the data set	9

1 Introduction

This vignette describes the **chondro** data set. It shows a complete data analysis work flow on a Raman map demonstrating frequently needed preprocessing methods

- baseline correction
- normalization
- smoothing / interpolating spectra
- preprocessing the spatial grid

and other basic work techniques

- plotting spectra
- plotting false color maps
- cutting the spectral range,
- selecting (extracting) or deleting spectra, and
- *aggregating* spectra (e.g. calculating cluster mean spectra).

The chemometric methods used are

- Principal Component Analysis (PCA) and
- hierarchical cluster analysis,

showing how to use data analysis procedures provided by R and other packages.

2 The Data Set



Figure 1: Microphotograph of the cartilage section. The frame indicates the measurement area ($35 \times 21 \mu\text{m}$).

Raman spectra of a cartilage section were measured on each point of a grid, resulting in a so-called *Raman map*. Figure 1 shows a microscope picture of the measured area and its surroundings.

The measurement parameters were:

Excitation wavelength: 633 nm

Exposure time: 10 s per spectrum

Objective: 100×, NA 0.85

Measurement grid: 35 × 21 μm, 1 μm step size

Spectrometer: Renishaw InVia

3 Data Import

Renishaw provides a converter to export their proprietary data in a so-called long format ASCII file. Raman maps are exported having four columns, *y*, *x*, *raman shift*, and *intensity*. *hyperSpec* comes with a function to import such files, `scan.txt.Renishaw`. The function assumes a map as default, but can also handle single spectra (`data = "spc"`), time series (`data = "ts"`), and depth profiles (`data = "depth"`). In addition, large files may be processed in chunks. In order to speed up the reading `scan.txt.Renishaw` does not allow missing values, but it does work with NA.

```
> chondro <- scan.txt.Renishaw ("rawdata/chondro.txt", data = "xyspc")
> chondro

hyperSpec object
  875 spectra
   3 data columns
 1272 data points / spectrum
wavelength: Delta * tilde(nu)/cm^-1 [numeric] 601.62 602.66 ... 1802.2
data: (875 rows x 3 columns)
  1. y: y/(mu * m) [numeric] -4.77 -4.77 ... 19.23
  2. x: x/(mu * m) [numeric] -11.55 -10.55 ... 22.45
  3. spc: I / a.u. [matrix1272] 501.72 518.53 ... 151.92 + NA
```

To get an overview of the spectra (figure 2a):

```
> plot (chondro, "spcprct15")
```

A mean intensity map (figure 2b) is produced by:

```
> plotmap (chondro, func.args = list (na.rm = TRUE), col.regions = seq.palette (20))
```

`plotmap` applies a function to squeeze all spectral intensities into a summary characteristic for the whole spectrum. This function defaults to the `mean`. Further arguments that should be handed to this function can be given in list `func.args`. As the raw data contains NAs due to deleting cosmic ray spikes, this argument is needed here.

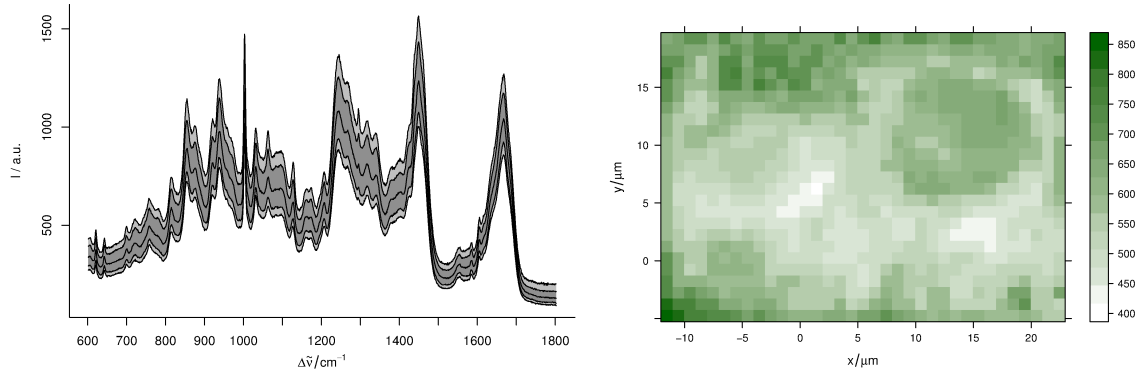
4 Preprocessing

As usual in Raman spectroscopy of biological tissues, the spectra need some preprocessing.

4.1 Aligning the Spatial Grid

The spectra were acquired on a regular square grid of measurement positions over the sample. *hyperSpec* however does not store the data on this grid but rather stores the position where the spectra were taken. This allows more handling of irregular point patterns, and of sparse measurements where not a complete (square) is retained but only few positions on this grid.

Occasionally, numeric or rounding errors will lead to seemingly irregular spacing of the data points. This is most obvious in false-colour maps of the sample (fig. fig:raw:b), as `plotmap` by default uses `panel.levelplot.raster` which assumes a regular grid is underlying the data. The symptoms are warnings “x’ values are not equispaced; output may be wrong” and white stripes in the false colour map (fig 3a) which occur even though the points are almost at their correct place (fig 3b)



(a) The raw spectra: median, 16th and 84th, and 5th and 95th percentile spectra.

(b) [The sum intensity of the raw spectra.

Figure 2: The raw data.

```
> ## disturb a few points
> chondro$x [500] <- chondro$x [500] + rnorm (1, sd = 0.01)
> chondro$y [660] <- chondro$y [660] + rnorm (1, sd = 0.01)

> plotmap (chondro, col.regions = seq.palette (20))

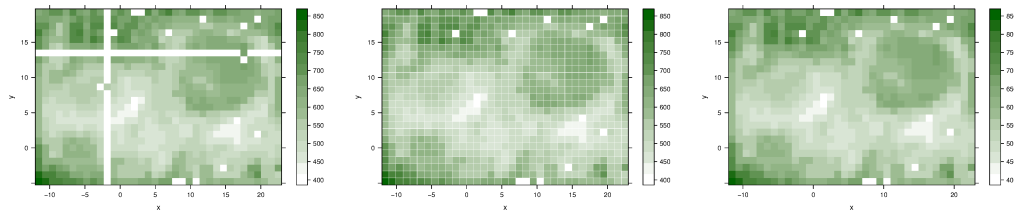
> require ("latticeExtra")
> plotmap (chondro, col.regions = seq.palette (20), panel = panel.levelplot.points,
+         col = NA, pch = 22, cex = 1.9)
```

Such slight rounding errors can be corrected by `fitraster` or `makeraster`. `fitraster` needs the step size of the raster and a starting coordinate, whereas `makeraster` tries to guess these parameters for `fitraster`. As long as just a few points are affected by rounding errors, `makeraster` works fine:

```
> chondro$x <- fitraster (chondro$x)$x
> chondro$y <- fitraster (chondro$y)$x
```

the result is shown in figure 3c.

```
> plotmap (chondro, col.regions = seq.palette (20))
```



(a) Raster plot with two off-grid points. (b) [Coordinates are slightly wrong only. (c) [Raster plot of corrected grid.

Figure 3: Rounding errors in the point coordinates of lateral measurement grid.

4.2 Spectral Smoothing

As the overview print shows that the spectra contain NAs (from cosmic spike removal that was done previously), the first step is to remove these. Also, the wavelength axis of the raw spectra is not evenly spaced (the data points are between 0.85 and 1 cm^{-1} apart from each other). Furthermore, it would be good to trade some spectral resolution for higher signal to noise ratio. All three of these issues are tackled by interpolating and smoothing of the wavelength axis by `spc.loess`. The resolution is to be reduced to 8 cm^{-1} , or 4 cm^{-1} data point spacing.

```
> chondro <- spc.loess (chondro, seq (602, 1800, 4))
> chondro

hyperSpec object
  875 spectra
  3 data columns
  300 data points / spectrum
wavelength: Delta * tilde(nu)/cm^-1 [numeric] 602 606 ... 1798
data: (875 rows x 3 columns)
  1. y: y [numeric] -4.77 -4.77 ... 19.23
  2. x: x [numeric] -11.55 -10.55 ... 22.45
  3. spc: I / a.u. [matrix300] 517.03 499.77 ... 168.04
```

The spectra are now the same as in the data set `chondro`. However, the data set also contains the clustering results (see at the very end of this document). They are stored for saving as the distributed demo data:

```
> spectra.to.save <- chondro
```

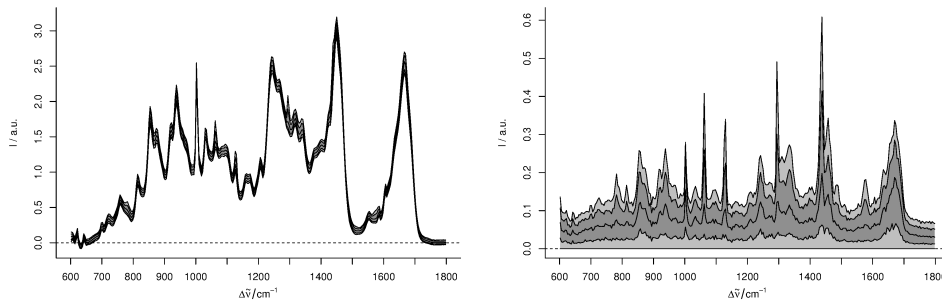
4.3 Baseline Correction

The next step is a linear baseline correction. `spc.fit.poly.below` tries to automatically find appropriate support points for polynomial baselines. The default is a linear baseline, which is appropriate in our case:

```
> baselines <- spc.fit.poly.below (chondro)

Fitting with npts.min = 15

> chondro <- chondro - baselines
```



(a) The spectra after smoothing, baseline correction, and normalization. (b) The spectra after subtracting the 5th percentile spectrum.

Figure 4: The preprocessed spectra.

4.4 Normalization

As the spectra are quite similar, area normalization should work well:

```
> chondro <- sweep (chondro, 1, apply (chondro, 1, mean), "/")
> plot (chondro, "spcprct15")
```

Note that normalization effectively cancels the information of one variate (wavelength), it introduces a collinearity. If needed, this collinearity can be broken by removing one of the variates involved in the normalization. Note that the `chondro` object shipped with *hyperSpec* set has multiple collinearities as only the first 10 principal components are shipped (see below).

For the results of these preprocessing steps, see figure 4a.

4.5 Subtracting the Overall Composition

The spectra are very homogeneous, but I'm interested in the differences between the different regions of the sample. Subtracting the minimum spectrum cancels out the matrix composition that is common to all spectra. But the minimum spectrum also picks up a lot of noise. So instead, the 5th percentile spectrum is subtracted:

```
> chondro <- sweep (chondro, 2, apply (chondro, 2, quantile, 0.05), "-")
> plot (chondro, "spcprct15")
```

The resulting data set is shown in figure 4b. Some interesting differences start to show up: there are distinct lipid bands in some but not all of the spectra.

4.6 Outlier Removal by Principal Component Analysis (PCA)

PCA is a technique that decomposes the data into scores and loadings (virtual spectra). It is known to be quite sensitive to outliers. Thus, I use it for outlier detection. The resulting scores and loadings are put again into *hyperSpec* objects by `decomposition`:

```
> pca <- prcomp (~ spc, data = chondro$, center = TRUE)
> scores <- decomposition (chondro, pca$x, label.wavelength = "PC", label.spc = "score / a.u.")
> loadings <- decomposition (chondro, t(pca$rotation), scores = FALSE, label.spc = "loading I / a.u.")
```

Plotting the scores of each PC against all other gives a good idea where to look for outliers.

```
> pairs (scores [[,1:20]], pch = 19, cex = 0.5)
```

Now the spectra can be found either by plotting two scores against each other (by `plot`) and identifying with `identify`, or they can be identified in the score map by `map.identify`. There is also a function to identify spectra in a spectra plot, `spc.identify`, but this is not helpful here.

```
> out <- map.identify (scores [,,5])
> out <- c (out, map.identify (scores [,,6]))
> out <- c (out, map.identify (scores [,,7]))

> out

[1] 105 140 216 289 75 69

> outcols <- c ("red", "blue", "#800080", "orange", "magenta", "brown")
> cols <- rep ("black", nrow(chondro))
> cols [out] <- outcols
```

We can check our findings by comparing the spectra to the bulk of spectra (figure):

```
> plot(chondro[1], plot.args = list (ylim = c (1, length (out) + .7)), lines.args = list( type = "n"))
> for (i in seq (along = out)){
+   plot(chondro, "spcprctl5", yoffset = i, add = TRUE, col = "gray")
+   plot (chondro [out[i]], yoffset = i, col = outcols[i] , add = TRUE, lines.args = list (lwd = 2))
+   text (600, i + .33, out [i]) }
```

and also by looking where these spectra appear in the scores pairs plot (figure):

```
> png ("chondro-fig--pca-pairs2.png", width = 500, height = 500)
> pch <- rep (46L, nrow (chondro))
> pch [out] <- 19L
> pairs (scores [[,1:7]], pch = pch, cex = 1, col = cols)
> dev.off ()
```

Finally, the outliers are removed:

```
> chondro <- chondro [- out]
```



Figure 5: Outlier removal by PCA

5 Hierarchical Cluster Analysis (HCA)

HCA fuses objects according to their (dis)similarity. The result is a dendrogram, a graph stating at which level two objects are similar and thus grouped together.

The first step in HCA is the choice of the distance. The R function `dist` offers a variety of distance measures to be computed. The so-called PEARSON distance $D_{Pearson}^2 = \frac{1-COR(X)}{2}$ is popular in data analysis of vibrational spectra and is provided by *hyperSpec*.

Also for computing the dendrogram, a number of choices are available. I choose WARD's method, and, as it uses EUCLIDEAN distance for calculating the dendrogram, EUCLIDEAN distance also for the distance matrix :

```

> dist <- dist (chondro [[]])
> dendrogram <- hclust (dist, method = "ward")

> plot (dendrogram)

```

In order to get clusters, the dendrogram is cut at a level specified either by height or by the number of clusters.

```

> chondro$clusters <- as.factor (cutree (dendrogram, k = 3))
> cols <- c ("dark blue", "orange", "#C02020")

```

The result for $k=3$ clusters is plotted as a map (figure 6b). If the color-coded variate (left hand side of the formula) is a factor, the legend bar does not show intermediate colors, and *hyperSpec*'s `levelplot` method uses the levels of the factor for the legend.

Thus meaningful names are assigned

```

> levels (chondro$clusters) <- c ("matrix", "lacuna", "cell")

```

and the cluster membership map is plotted:

```

> print (plotmap (chondro, clusters ~ x * y, col.regions = cols))

```

The cluster membership can also be marked in the dendrogram:

```

> par (xpd = TRUE) # allow plotting the markers into the margin
> plot (dendrogram, labels = FALSE, hang = -1)
> mark.dendrogram (dendrogram, chondro$clusters, col = cols)

```

Figure 6a shows the dendrogram and 6b the resulting cluster map. The three clusters correspond to the cartilage matrix, the lacuna and the cells. The left cell is destroyed and its contents are leaking into the matrix, while the right cells looks intact.

We can calculate the cluster mean spectra using `aggregate`. However, we can do even better and plot the cluster mean spectra ± 1 standard deviation (see figure 7):

```

> cluster.means <- aggregate (chondro, chondro$clusters, mean_pm_sd)
> plot(cluster.means, stacked = ".aggregate", fill = ".aggregate", col = cols)

```

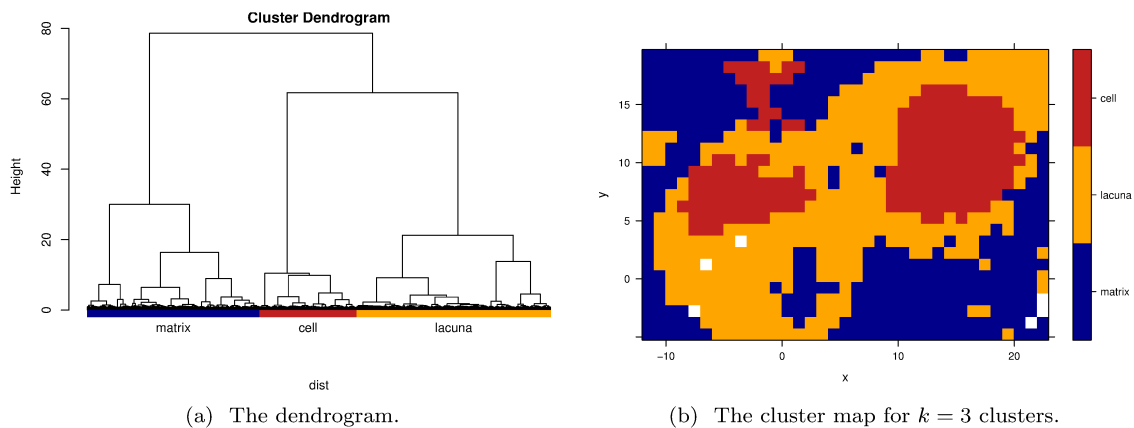


Figure 6: Hierarchical cluster analysis.



Figure 7: The cluster mean ± 1 standard deviation spectra. The blue cluster shows distinct lipid bands, the green cluster collagen, and the red cluster proteins and nucleic acids.

6 Plotting a False-Colour Map of Certain Spectral Regions

hyperSpec comes with a sophisticated interface for specifying spectral ranges. Expressing things like $1000\text{ cm}^{-1} \pm 1$ data points are easily possible. Thus, we can have a fast look at the nucleic acid distribution, using the DNA bands at 728, 782, 1098, 1240, 1482, and 1577 cm^{-1} :

```
> plotmap (chondro[, , c( 728, 782, 1098, 1240, 1482, 1577)],
+         col.regions = colorRampPalette (c("white", "gold", "dark green"), space = "Lab") (20))
```



Figure 8: False colour map of the DNA band intensities.

The result is shown in figure 8. While the nucleus of the right cell shows up nicely, nothing is detected in the remainders of the left cell.

7 Saving the data set

Finally, the example data set is put together and saved. In order to keep the package size small, only a PCA-compressed version with 10 PCs is shipped as example data set of the package.

```
> spectra.to.save$clusters <- factor (NA, levels = levels (chondro$clusters))
> spectra.to.save$clusters[- out] <- chondro$clusters
> pca <- prcomp (spectra.to.save [,,])
> .chondro.scores <- pca$x      [, seq_len (10)]
```

```

> .chondro.loadings <- pca$rot      [, seq_len (10)]
> .chondro.center   <- pca$center
> .chondro.wl       <- wl (chondro)
> .chondro.labels   <- lapply (labels (chondro), as.expression)
> .chondro.extra    <- spectra.to.save$.
> save (.chondro.scores, .chondro.loadings, .chondro.center,
+       .chondro.wl, .chondro.labels, .chondro.extra,
+       file = "chondro-internal.rda")

```

This is the file distributed with *hyperSpec* as example data set.

Session Info

```

R version 3.0.1 (2013-05-16)
Platform: x86_64-pc-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=de_DE.UTF-8      LC_NUMERIC=C              LC_TIME=de_DE.UTF-8
 [4] LC_COLLATE=de_DE.UTF-8   LC_MONETARY=de_DE.UTF-8  LC_MESSAGES=de_DE.UTF-8
 [7] LC_PAPER=C               LC_NAME=C                 LC_ADDRESS=C
[10] LC_TELEPHONE=C           LC_MEASUREMENT=de_DE.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] grid      stats      graphics  grDevices  utils      datasets  methods   base

other attached packages:
[1] latticeExtra_0.6-24      RColorBrewer_1.0-5       hyperSpec_0.98-20130607  mvtnorm_0.9-9995
[5] lattice_0.20-15

loaded via a namespace (and not attached):
[1] tools_3.0.1

```