Heatmaps in Spectroscopy: hmapSpectra

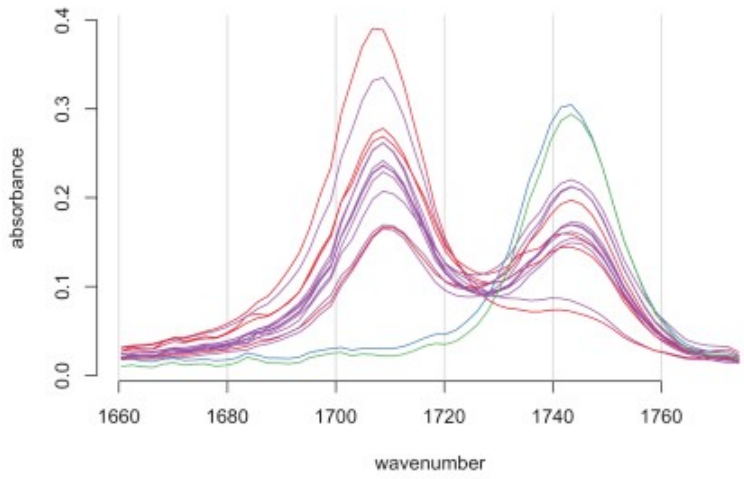
The `hmapSpectra` function in `ChemoSpec` displays a heatmap to help you focus on which frequencies drive the separation of your samples.[1] We'll use the example from `?hmapSpectra` which uses the built-in `SrE.IR` data set. This data set is a series of IR spectra of commercial *Serenoa repens* oils which are composed of mixtures of triglycerides and free fatty acids (see `?SrE.IR` for more information). Thus the carbonyl region is of particular interest. The example narrows the frequency range to the carbonyl region for easy interpretation. Let's look first at the spectra.

Note: rather than link every mention of a help page in this post, remember you can see all the documentation at this site.

```
library("ChemoSpec")
```

```
## Loading required package: ChemoSpecUtils
```

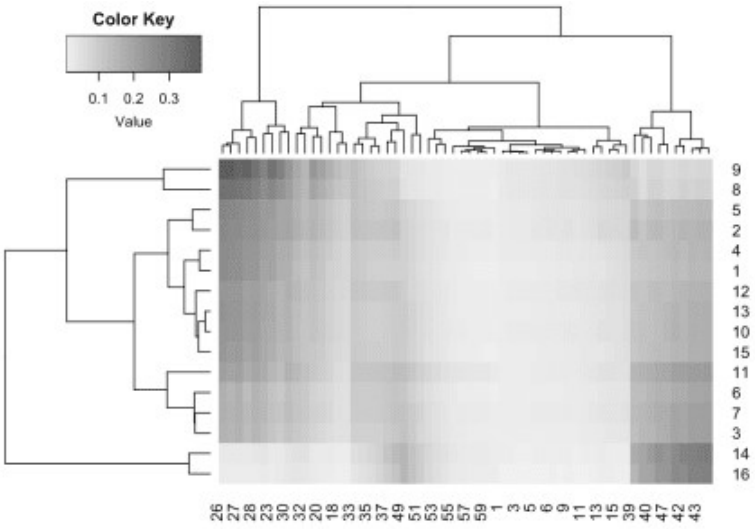
```
data(SrE.IR) # load the data set
# limit to the carbonyl region
IR <- removeFreq(SrE.IR, rem.freq = SrE.IR$freq > 1775 | SrE.IR$freq < 1660)
plotSpectra(IR, which = 1:16, lab.pos = 1800)
```



The blue and green spectra are samples composed only of triglycerides, and hence the ester carbonyl is the primary feature. All other samples are clearly mixtures of ester and carboxylic acid stretching peaks. And now for the heatmap, using defaults:

```
res <- hmapSpectra(IR)
```

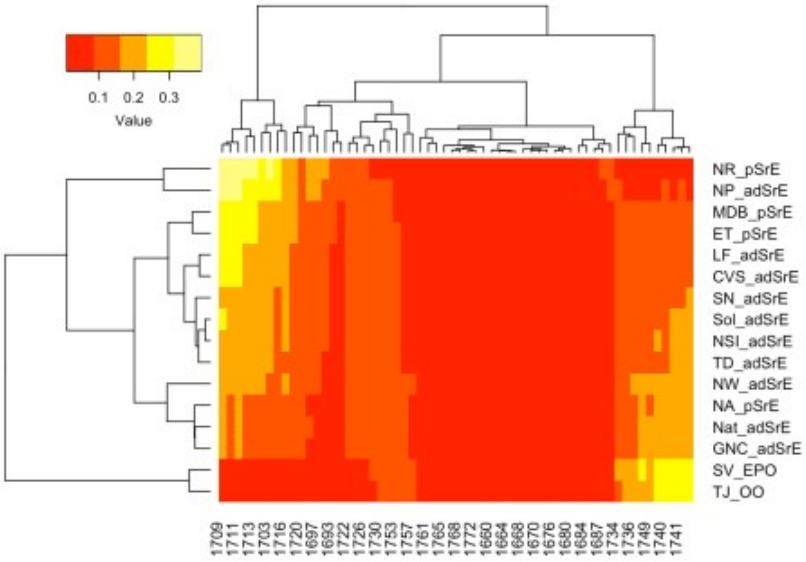
```
## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus
```



In this default display, you'll notice that the rows and column labels are indices to the underlying sample names and frequency list. This is not so helpful. The color scheme is not so exciting either. `hmapSpectra` uses the package `seriation` which in turn uses the `heatmap.2` function in package `gplots`. Fortunately we can use the `...` argument to pass additional arguments to `heatmap.2` to get a much more useful plot.

Customizing the hmapSpectra Display

```
# Label samples and frequencies by passing arguments to heatmap.2
# Also make a few other nice plot adjustments
res <- hmapSpectra(IR,
  col = heat.colors(5),
  labRow = IR$names, labCol = as.character(round(IR$freq)),
  margins = c(4, 6), key.title = ""
)
```



This is a lot nicer plot, since the rows are labeled with the sample names, and the columns with frequencies. Note that not every column is labeled, only every few frequencies. If you need the actual frequencies, which you probably will, they can be obtained from the returned object (`res` in this case; see the end of this post for an example).

Interpreting the Plot

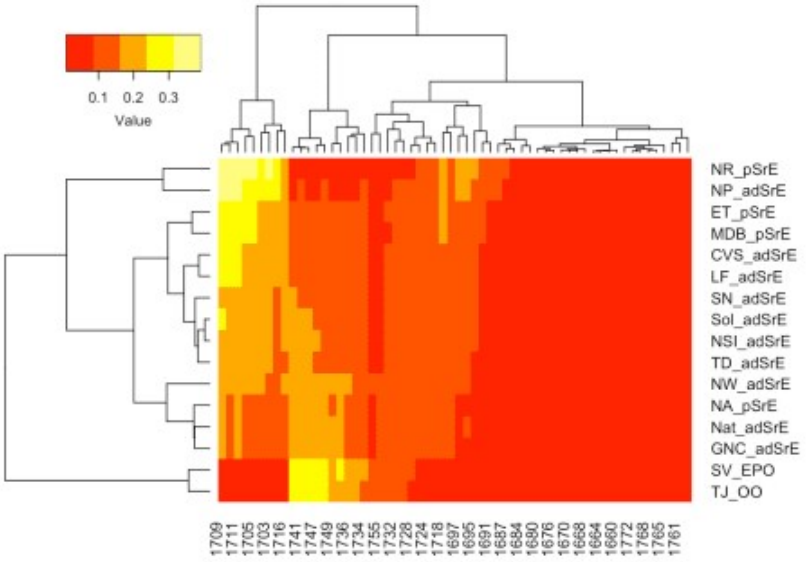
How do we interpret this plot? This is a *seriated* heatmap, which means the rows and columns have been re-

ordered according to some algorithm (more on this in a moment). The ordering puts the frequencies most important in distinguishing the samples in the upper left and lower right (the yellow regions). In the lower right corner, we see the two outlier samples TJ_OO and SV_EPO grouped together. On the frequency axis, we see that ester stretching peaks around 1740 \(\mathrm{cm}^1\) are characteristic for these samples. In the upper left corner, we see several samples grouped together, and associated with the fatty acid carboxylic acid peak around 1710 \(\mathrm{cm}^1\). From these two observations, we can conclude that these two peak ranges are most important in separating the samples. Of course, in this simple example using a small part of the spectrum, this answer was already clear by simple inspection. Using a simple/limited range of data helps us to be sure we understand what's happening when we try a new technique.

Using a Different Distance Measure & Seriation Method

The default data treatments for `hmapSpectra` are inherited from `hmap` in package `seriation`. The default distance between the samples is the Euclidean distance. The default seriation method is "OLO" or "optimal leaf ordering". The full list of seriation methods is described in `?seriate`. There are more than 20 options. As with the display details, we can change these defaults via the `...` arguments. Let's use the cosine distance (the same as the Pearson distance), and seriate using the Gruvaeus-Wainer algorithm (there's a brief explanation of this algorithm at `?seriate`).

```
cosine_dist <- function(x) as.dist(1 - cor(t(x)))
res <- hmapSpectra(IR,
  col = heat.colors(5),
  labRow = IR$names, labCol = as.character(round(IR$freq)),
  margins = c(4, 6), key.title = "",
    dist_fun = cosine_dist,
    method = "GW"
)
```



You can see that using different distance measures and seriation algorithms gives a rather different result: the ester "hot spots" which were in the lower right corner are now almost in the lower left corner. Which settings are best will depend on your data set, the goal of your analysis, and there are a lot of options from which to choose. The settings used here are simply for demonstration purposes, I make no claim these settings are appropriate!

Finally, if you want to capture the re-ordered frequencies, you can access them in the returned object:

```
round(IR$freq[res$colInd])
```

```
##  [1] 1709 1707 1711 1713 1705 1714 1703 1701 1716 1743 1741 1745 1747 1740
1749
```

```
## [16] 1738 1736 1751 1734 1757 1755 1753 1732 1730 1728 1726 1724 1722 1718
1720
## [31] 1697 1699 1695 1693 1691 1689 1687 1686 1684 1682 1680 1678 1676 1674
1670
## [46] 1672 1668 1666 1664 1662 1660 1774 1772 1770 1768 1767 1765 1763 1761
1759
```