Examples of implementation of pre-processing method described in paper with R code snippets - Electronic Supplementary Information (ESI)

Krzysztof Banas, Agnieszka Banas, Mariusz Gajda, Bohdan Pawlicki, Wojciech M. Kwiatek, and Mark B H Breese

February 9, 2015

Introduction

This document was build in R Studio [1] Graphic User Interface (GUI) with the help of following R [2] packages:

- hyperSpec [3]
- RColorBrewer [4]
- plotrix [5]
- gridExtra [6]
- shape [7]
- knitr [8]

and includes R code for typical workflow in hyperspectral dataset normalization.

Data loading

Loading data to R Environment is extremely user-dependent topic and there is no single recipe how to do this properly and most efficiently. In the case presented here Focal Plane Array (FPA) (64 by 64 pixels) of IR detector recordings were saved in a matrix format (ASCII text file) where wavenumber vector is stored in the first column and the subsequent spectra are stored

in the columns from 2 to 4097. Generic R commands read.csv or read.table for importing data into R Environment may be used in this case.

```
library(hyperSpec)
library(RColorBrewer)
library(plotrix)
library(gridExtra)


file <- read.table ("6m_600_256scans_03.dpt", header = FALSE, dec = ".",
                    sep = ",")
# Single shot FTIR map has 64 by 64 spectra, pixel size is equal 2.7
# by 2.7 microns
# number of rows
x1 = 64
# number of colums
y1 = 64
x = rep (seq(from=0, to=2.7*(y1-1), by=2.7), times=x1)
y = rep (seq(from=0, to=2.7*(x1-1), by=2.7), each=y1)
d = data.frame(x,y)
# defining hyperSpec object
map01 <- new ("hyperSpec", wavelength = file [,1], spc = t (file [, -1]),
            data = d,label = list(.wavelength = "Wavenumber /cm-1",
            spc = "I / a.u."))
```

Raw spectra and spatial distributions

Below it is shown how to obtain an intensity map of raw spectra, where integration was done over full spectral range, mean value of the absorbance is mapped here to the false-colour scale.

```
jet.colors <- colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan",
 "#7FFF7F", "yellow", "#FF7F00", "red", "#7F0000"))
plot1<-plotmap(map01,func=mean, col.regions = jet.colors(256),
        scales=list(x=list(cex=1),y=list(cex=1)),
        colorkey= list(labels=list(cex=1)),
        main=list(label="Whole spectral range", cex=1))
```

Analogically one can reduce the spectral range by cutting some undesired frequencies (CO_2 and air humidity influence).

```
map02 <- map01 [ , , c(900~1710, 2700~3700)]
plot2<-plotmap(map02,func=mean, col.regions = jet.colors(256),
        scales=list(x=list(cex=1), y=list(cex=1)),
        colorkey= list(labels=list(cex=1)),
        main=list(label="Reduced spectral range", cex=1))
grid.arrange(plot1,plot2, ncol=2)
```

Next example illustrate how to obtain spatial distributions for the particular biochemical groups by integrating over specific spectral ranges: 3000-2900 cm^{-1} (for lipids), 1600-1500 cm^{-1} (for proteins), 1300-1200 cm^{-1} (for carbohydrates). The maps showing the spatial distributions are presented in Fig. 2.

```
colors1 <- brewer.pal(9, "Oranges")
pal1 <- colorRampPalette(colors1)
colors2 <- brewer.pal(9, "Greens")
pal2 <- colorRampPalette(colors2)
colors3 <- brewer.pal(9, "Blues")
pal3 <- colorRampPalette(colors3)

plotmap(map01 [,,3000~2900], col.regions = pal1(256),scales=list(x=list(cex=2),
            y=list(cex=2)), colorkey= list(labels=list(cex=2)),
            main=list(label="Lipids", cex=2))
plotmap(map01 [,,1600~1500], col.regions = pal2(256),scales=list(x=list(cex=2),
            y=list(cex=2)), colorkey= list(labels=list(cex=2)),
            main=list(label="Proteins", cex=2))
plotmap(map01 [,,1300~1200], col.regions = pal3(256),scales=list(x=list(cex=2),
            y=list(cex=2)), colorkey= list(labels=list(cex=2)),
            main=list(label="Carbohydrates", cex=2))
```

Standard normalization

If one perform standard normalization (of any kind: area, vector or to the intensity of the particular band) on this type of dataset big problem will arise from the areas of low signal intensity (areas where there is no biological tissue, but only support material - mylar foil in this case). Normalization

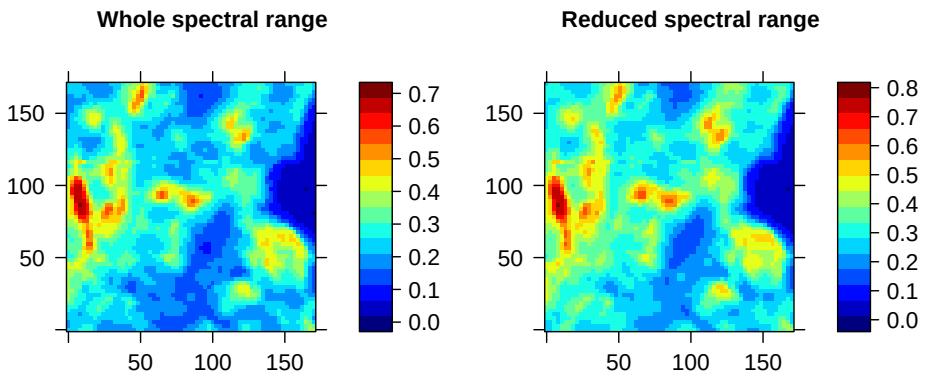

Figure 1: Raw Intensity maps in full (left) and reduced (right) spectral ranges - Mean value for every pixel is mapped to false-colour scale

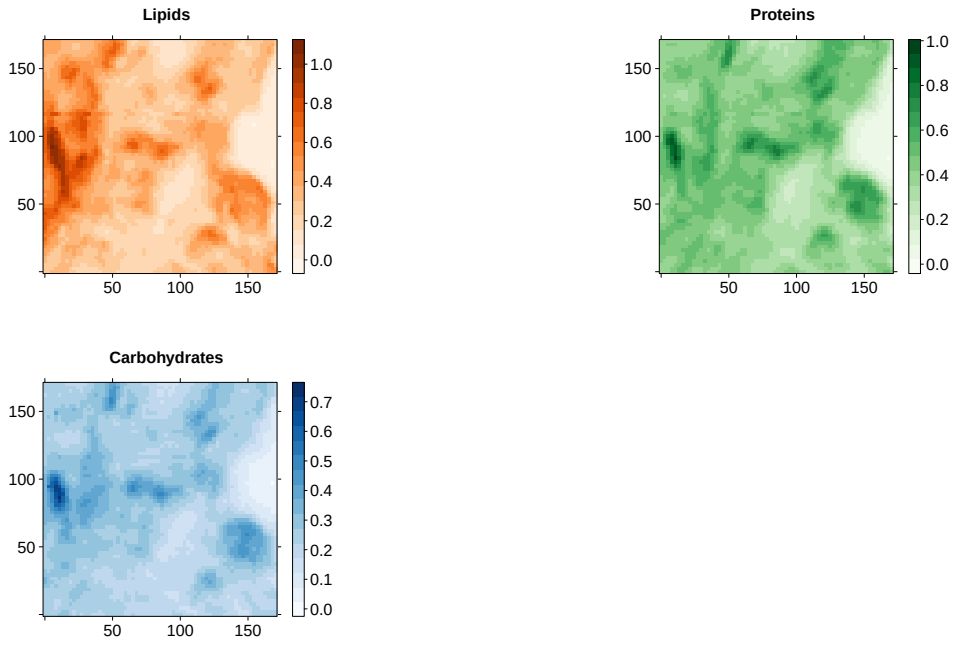

Figure 2: Raw spatial distributions of lipids, proteins and carbohydrates obtained by integrating over particular spectral ranges without prior preprocessing

factors for these regions will by very high. Taking into account that the normalization factor is multiplied by non-normalized values in normalization formula we end up with high normalized values in the empty areas. Fig. 3 illustrates the source of the problem.

```
#Area Normalization
norm01 <- sweep (map01, 1, mean, "/")
plot3<-plotmap(norm01 [,,3698~902], func=mean, col.regions = jet.colors(256),
        scales=list(x=list(cex=1),y=list(cex=1)),
        colorkey= list(labels=list(cex=1)),
        main=list(label="Normalized data - whole spectral range", cex=1))
```

You can see that reason for that: high values of the normalization factor in mylar foil areas.

```
#Area Normalization
factor01<- 1/ apply (map01, 1, mean)
plot4<-plotmap(factor01,func=mean, col.regions = jet.colors(256),
      scales=list(x=list(cex=1),
      y=list(cex=1)), colorkey= list(labels=list(cex=1)),
      main=list(label="Normalization Factor", cex=1))
grid.arrange(plot3,plot4, ncol=2)
```

Solution - First Attempt

First solution for this problem is to eliminate temporarily pixels with mylar only from the dataset, perform the normalization and introduce the mylar pixels again before visualization. The problem is how to determine the pixels that are mylar only. One of the possible and standard answer is to use the intensity limit. One can set the threshold to a certain value in order to remove pixels with values lower than this threshold. One drawback of this approach is the fact that one has to arbitraly decide what value of threshold would be correct. Let's try with 0.12 (pixels with spectra with the mean intensity lower than 0.12 we assign as pure mylar).

```
#setting the logic condition
low.int <- apply(map01,1,mean) < 0.12
#dividing the dataset to mylar and sample
mylar02 <- map01[low.int]
```

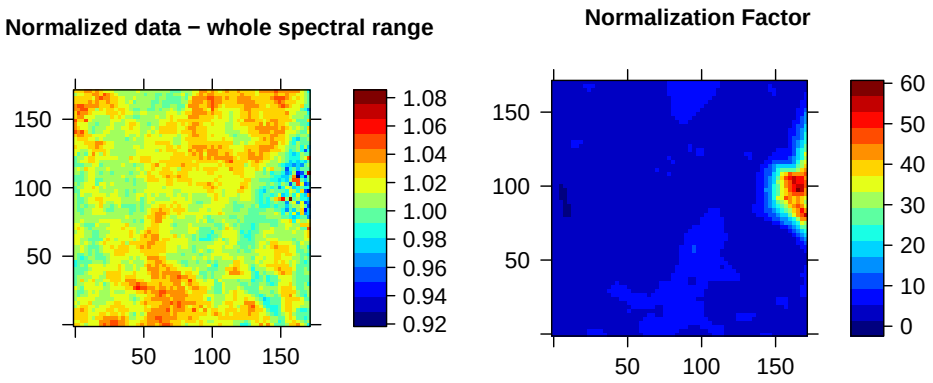

Figure 3: Area Normalized data - whole spectral range(left) and calculated area normalization factor map (right)

```
test02 <- map01[!low.int]
```

Here you could see how the parts of the map (test02 and mylar02) looks.

```
plotmap(test02,func=mean, col.regions = jet.colors(256),
  scales=list(x=list(cex=2),y=list(cex=2)), colorkey= list(labels=list(cex=2)),
  main=list(label="Tissue", cex=2))
plotmap(mylar02,func=mean, col.regions = jet.colors(256),
  scales=list(x=list(cex=2),y=list(cex=2)), colorkey= list(labels=list(cex=2)),
  main=list(label="Mylar", cex=2))
norm02 <- sweep (test02, 1, mean, "/")
```

Now we normalize only non-empty part. We can check how the spatial distributions for selected bands look like for the dataset nomalized this way.

```
plotmap(norm02 [,,3000~2900], col.regions = pal1(256),
  scales=list(x=list(cex=2),y=list(cex=2)), colorkey= list(labels=list(cex=2)),
  main=list(label="Lipids", cex=2))
plotmap(norm02 [,,1600~1500], col.regions = pal2(256),
  scales=list(x=list(cex=2),y=list(cex=2)), colorkey= list(labels=list(cex=2)),
  main=list(label="Proteins", cex=2))
plotmap(norm02[,,1300~1200], col.regions = pal3(256),
  scales=list(x=list(cex=2),y=list(cex=2)), colorkey= list(labels=list(cex=2)),
  main=list(label="Carbohydrates", cex=2))
```

Various threshold values

```
#setting the logic condition
low.int09 <- apply(map01,1,mean) < 0.09
#dividing the dataset to mylar and sample
mylar09 <- map01[low.int09]
test09 <- map01[!low.int09]

#setting the logic condition
low.int14 <- apply(map01,1,mean) < 0.14
#dividing the dataset to mylar and sample
mylar14 <- map01[low.int14]
```

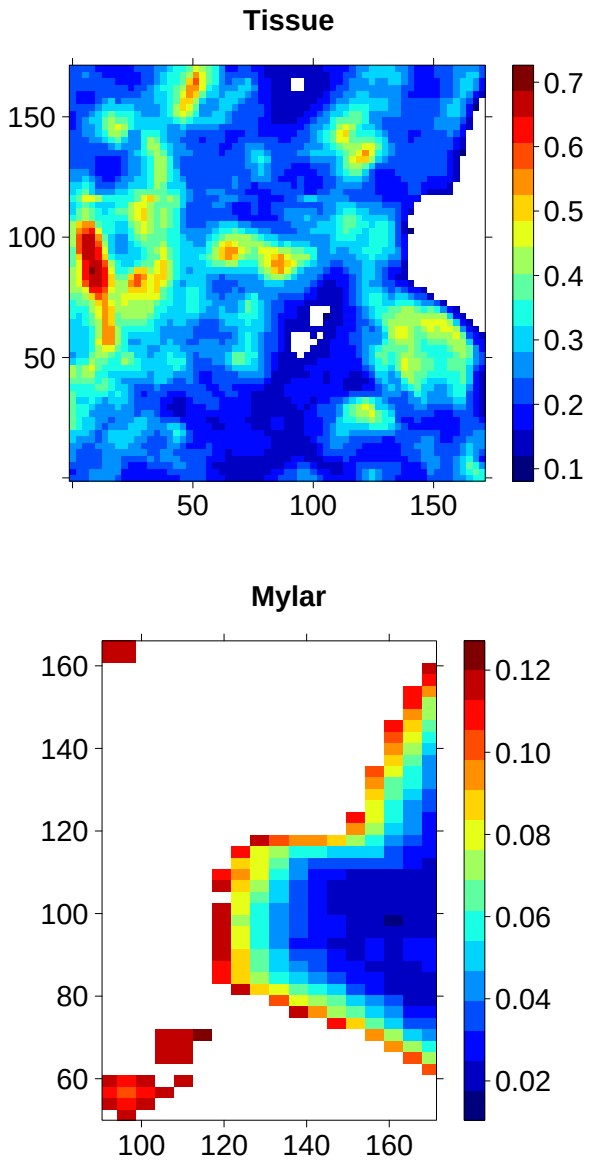

Figure 4: Parts of the map assign as tissue and mylar based on intensity threshold method

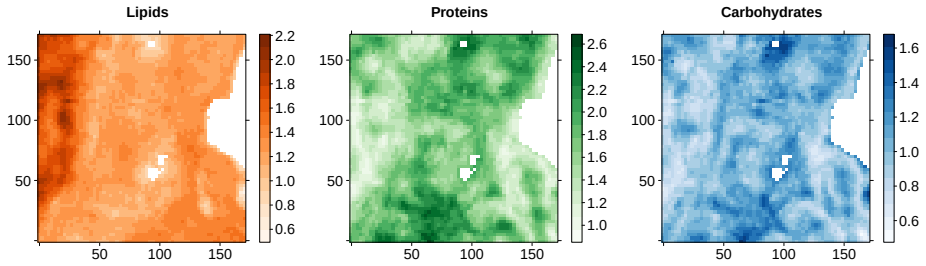

Figure 5: Spatial distributions of lipids, proteins and carbohydrates obtained by integrating over particular spectral ranges for area normalized spectra

```
test14 <- map01[!low.int14]

plott09<-plotmap(test09,func=mean, col.regions = jet.colors(256),
  scales=list(x=list(cex=1),y=list(cex=1)), colorkey= list(labels=list(cex=1)),
  main=list(label="Threshold 0.09", cex=1))

plott12<-plotmap(test02,func=mean, col.regions = jet.colors(256),
  scales=list(x=list(cex=1),y=list(cex=1)), colorkey= list(labels=list(cex=1)),
  main=list(label="Threshold 0.12", cex=1))

plott14<-plotmap(test14,func=mean, col.regions = jet.colors(256),
  scales=list(x=list(cex=1),y=list(cex=1)), colorkey= list(labels=list(cex=1)),
  main=list(label="Threshold 0.14", cex=1))
grid.arrange(plott09,plott12,plott14, ncol=2)
```

Solution - based on hierarchical cluster analysis

First we have to perform hierarchical cluster analysis.

```
#calculate distance between the spectra
dist <- dist (map01 [[]])
#construct dendrogram based on the distance and the linkage method
dendrogram <- hclust (dist, method ="ward")
#cut dendrogram for the expected number of clusters
#add cluster membership to hyperSpec object
```

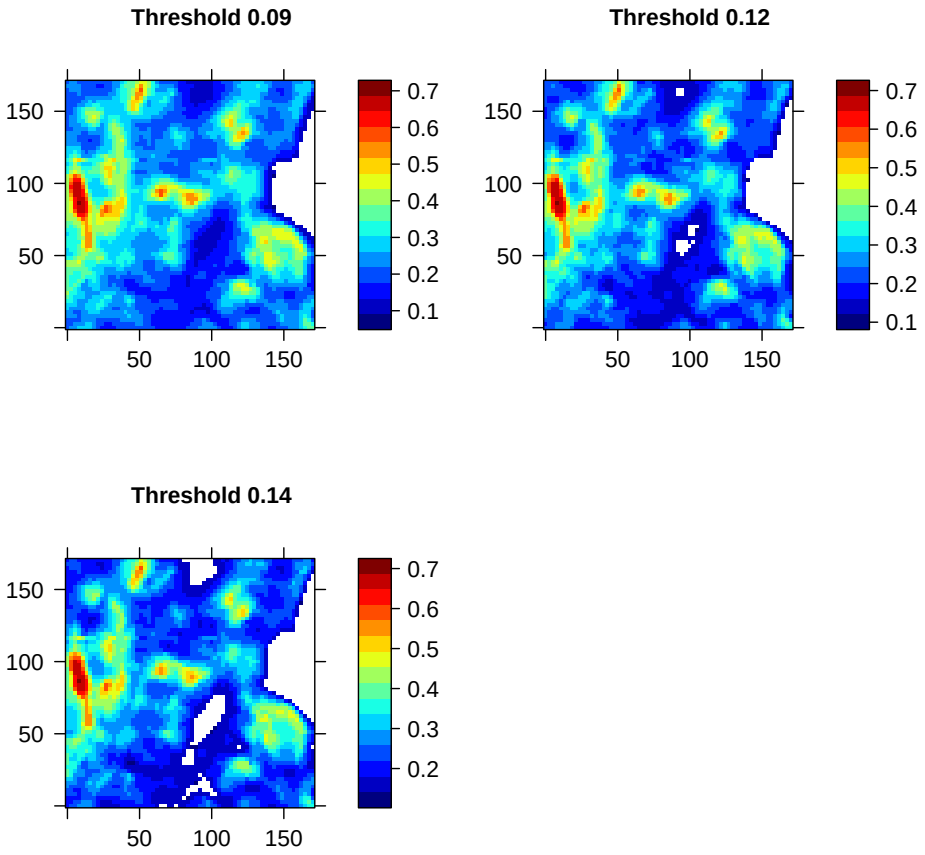

Figure 6: Effect of setting various threshold values

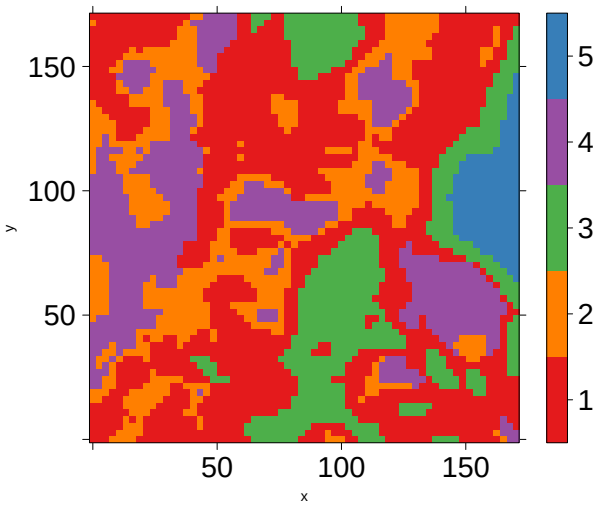

Figure 7: Map of the location of five clusters obtained by hierarchical cluster analysis using Euclidean distance measure and Ward linkage algorithm

```
map01$clusters <- as.factor (cutree (dendrogram, k = 5))
#set colours and names for clusters
col5 <- brewer.pal(5, "Set1")
pal5 <- colorRampPalette(col5)
cols5 =pal5(5)
cols6 = c("#E41A1C" ,"#FF7F00", "#4DAF4A" ,"#984EA3" ,"#377EB8")
levels (map01$clusters) <- c ("Cl_01", "Cl_02", "Cl_03", "Cl_04", "Cl_05")
#map of the cluster location
plotmap(map01, clusters ~ x * y, col.regions = cols6, scales=list(x=list(cex=2),
        y=list(cex=2)), colorkey= list(labels=list(cex=2)))
```

Dendrogram with marked cluster membership.

```
par (xpd = TRUE, cex.lab=1.5, cex.axis=1.5)
plot (dendrogram, labels = FALSE, hang = -1)
mark.dendrogram (dendrogram, map01$clusters, height= 200, col = cols6, cex=1.5)
```

Mean spectra for clusters.

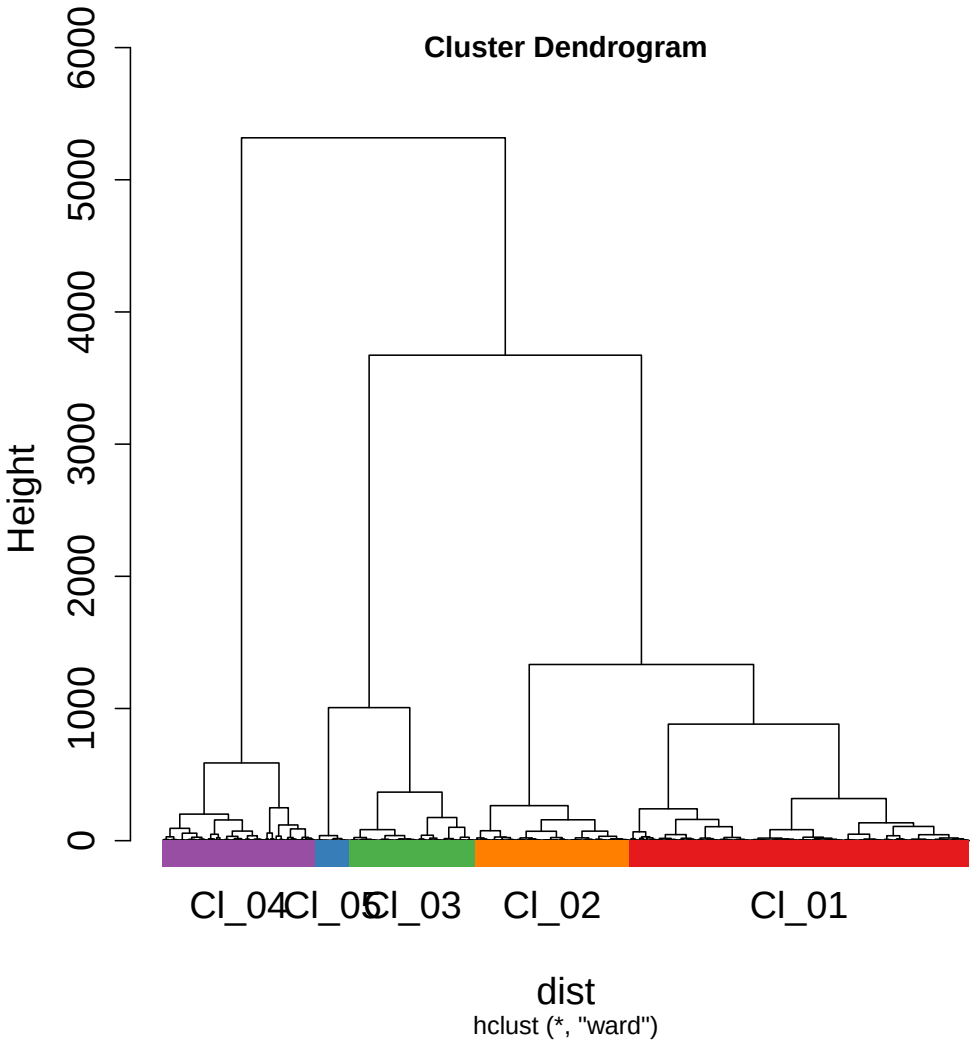

Figure 8: Dendrogram with marked cluster membership

```
cluster.means <- aggregate (map01, map01$clusters, mean_pm_sd)
plot(cluster.means, stacked = ".aggregate", fill = ".aggregate", col = cols6,
    wl.reverse =TRUE, wl.range = c(901~1710, 2700~3700), xoffset = 900,
    axis.args = list(cex.axis=1.6), title.args = list(cex.lab=1.6))
```

Cluster number 5 is identified as mylar only surface. Let's remove it from the sample.

```
clusters <- split (map01, map01$clusters)
mylarhca <- clusters$Cl_05
map03 <- rbind(clusters$Cl_01,clusters$Cl_02,clusters$Cl_03,clusters$Cl_04)
plotmap(map03 [,,3000~1200], col.regions = jet.colors(256),
  scales=list(x=list(cex=2),y=list(cex=2)), colorkey= list(labels=list(cex=2)),
  main=list(label="Tissue", cex=2))
plotmap(mylarhca [,,3000~1200], col.regions = jet.colors(256),
  scales=list(x=list(cex=2),y=list(cex=2)), colorkey= list(labels=list(cex=2)),
  main=list(label="Mylar", cex=2))
```

Now is the time for normalization.

```
#Area Normalization
norm03 <- sweep (map03, 1, mean, "/")
```

Let's have a look to some bands.

```
plotmap(norm03 [,,3000~2900], col.regions = pal1(256),
  scales=list(x=list(cex=2),y=list(cex=2)), colorkey= list(labels=list(cex=2)),
  main=list(label="Lipids", cex=2))
plotmap(norm03 [,,1600~1500], col.regions = pal2(256),
  scales=list(x=list(cex=2),y=list(cex=2)), colorkey= list(labels=list(cex=2)),
  main=list(label="Proteins", cex=2))
plotmap(norm03[,,1300~1200], col.regions = pal3(256),
  scales=list(x=list(cex=2),y=list(cex=2)), colorkey= list(labels=list(cex=2)),
  main=list(label="Carbohydrates", cex=2))
```

Session information

In order improve the reproducibility of the data evaluation in R one should provide information about the software and packages versions and operating

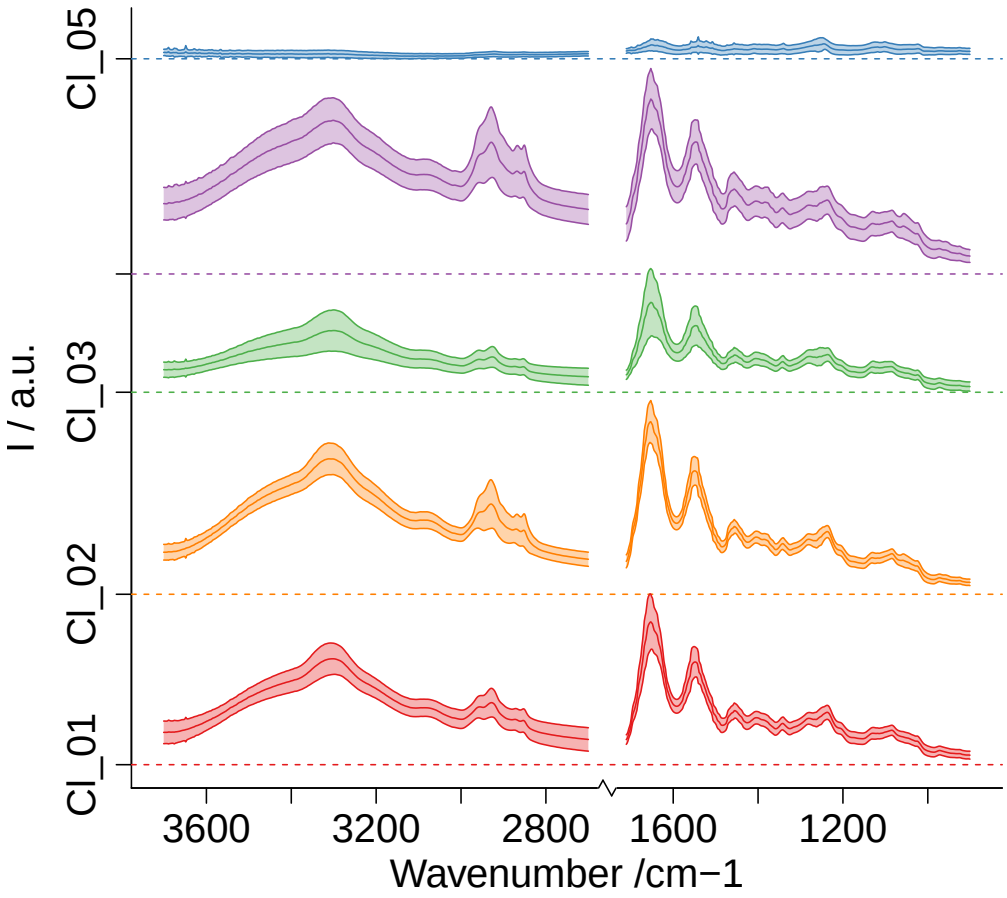

Figure 9: Mean spectra for clusters

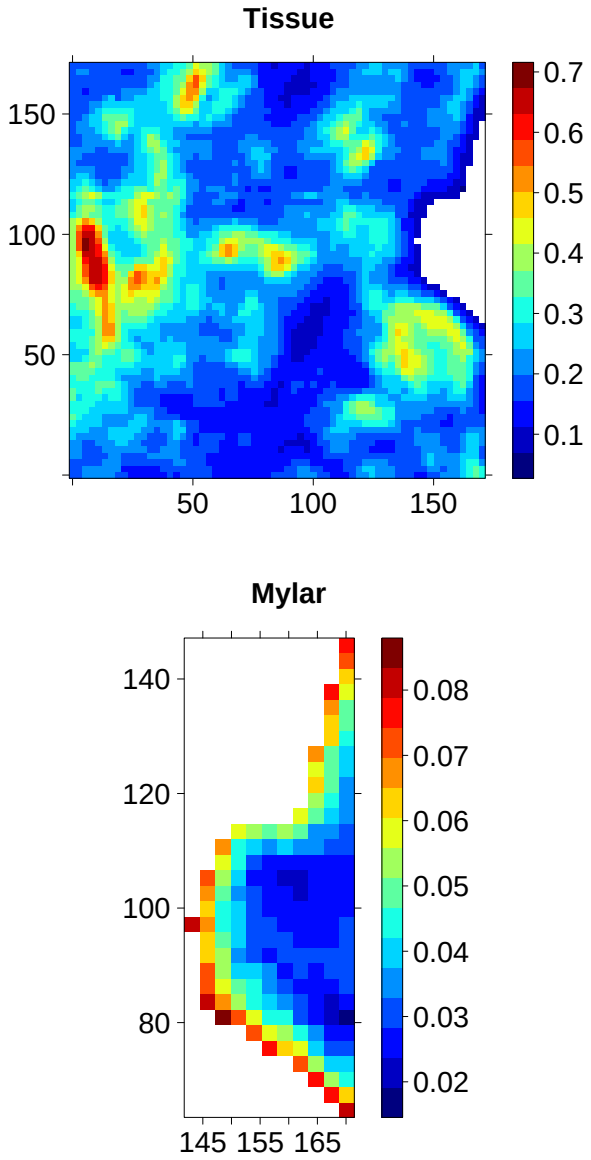

Figure 10: Parts of the map assign as tissue and mylar based on hierarchical cluster analysis

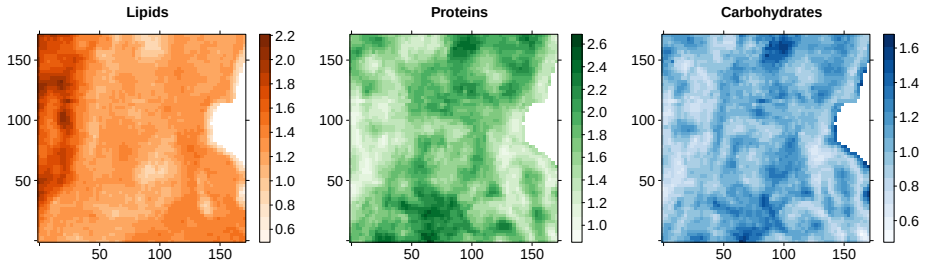

Figure 11: Spatial distributions of lipids, proteins and carbohydrates obtained by integrating over particular spectral ranges for area normalized spectra

system. This could be conviniently done with the one line of code - function `sessionInfo()`.

```
sessionInfo()

## R version 3.0.2 (2013-09-25)
## Platform: i386-w64-mingw32/i386 (32-bit)
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.1252
## [2] LC_CTYPE=English_United Kingdom.1252
## [3] LC_MONETARY=English_United Kingdom.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.1252
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] gridExtra_0.9.1         plotrix_3.5-7           RColorBrewer_1.0-5
## [4] hyperSpec_0.98-20140523 mvtnorm_1.0-0           lattice_0.20-29
## [7] knitr_1.6
##
## loaded via a namespace (and not attached):
## [1] digest_0.6.4   evaluate_0.5.5 formatR_1.0    highr_0.3
```

```
## [5] stringr_0.6.2  tools_3.0.2
```

References

[1] RStudio. *RStudio*. R Foundation for Statistical Computing. Vienna, Austria, 2014. URL: http://www.rstudio.org/.

[2] R Development Core Team. *R: A Language and Environment for Statistical Computing*. ISBN 3-900051-07-0. R Foundation for Statistical Computing. Vienna, Austria, 2014. URL: http://www.R-project.org.

[3] Claudia Beleites and Valter Sergo. *hyperSpec: a package to handle hyperspectral data sets in R*. R package version 0.98-20140523. 2014. URL: http://hyperspec.r-forge.r-project.org.

[4] Erich Neuwirth. *RColorBrewer: ColorBrewer palettes*. R package version 1.0-5. 2011. URL: http://CRAN.R-project.org/package=RColorBrewer.

[5] Lemon J. "Plotrix: a package in the red light district of R". In: *R-News* 6.4 (2006), pp. 8–12.

[6] Baptiste Auguie. *gridExtra: functions in Grid graphics*. R package version 0.9.1. 2012. URL: http://CRAN.R-project.org/package=gridExtra.

[7] Karline Soetaert. *shape: Functions for plotting graphical shapes, colors*. R package version 1.4.1. 2014. URL: http://CRAN.R-project.org/package=shape.

[8] Yihui Xie. "knitr: A Comprehensive Tool for Reproducible Research in R". In: ed. by Victoria Stodden, Friedrich Leisch, and Roger D. Peng.