Sydney Soil School

Soil Spectral Inference Workshop

March 30 - April 1, 2016

Pretreatment and preprocessing of soil spectra

Introduction

Before making full use of our spectra for prediction of soil properties, we need to do spectra preprocessing. In the process of doing spectroscopic analysis, the sample volume, sample preparation, measurement method, and the measuring parameters, such as the choice of the scanning times and the scanning resolution will more or less bring inevitable noise to the spectral data. Further, when analyzing phenomena with non-uniform particle size of solids such as soils, the reflectance spectrum is often accompanied by scattering noise. At our disposal however are a number of spectral smoothing, filtering and signal enhancing algorithms to ensure that we can best use the spectra for soil property inference. While there may be many potential preprocessing algorithms, base lining procedures and other such spectral pretreatments, the few that are described below are often used for soil spectroscopy. A useful reference that describes in more detail each of these pretreatments is Buddenbaum and Steffens (2012).

The pretreatments of particular focus in this exercise are:

- The Savitzkty-Golay smoothing filter
- Standard-normal variate transformation (SNV)
- Multiplicative scatter correction (MSC)
- Continuum removal with convex hull

We then have a look at a few data reduction or compression methods which essentially reduce our spectra to a smaller dimension:

- Resample and averaging
- Wavelets

Spectra Pretreatments

Getting things prepared

There are a few R packages we need to use for the following examples

```
library(spectroscopy)
library(signal)
library(plyr)
library(wavethresh)
```

We will be using the vis-NIR spectra data set **spectra.txt** files as used in previous examples. We call this object spectra . Using the dim function returns us the dimensions of the spectral dataset.

```r
dim(spectra)
```

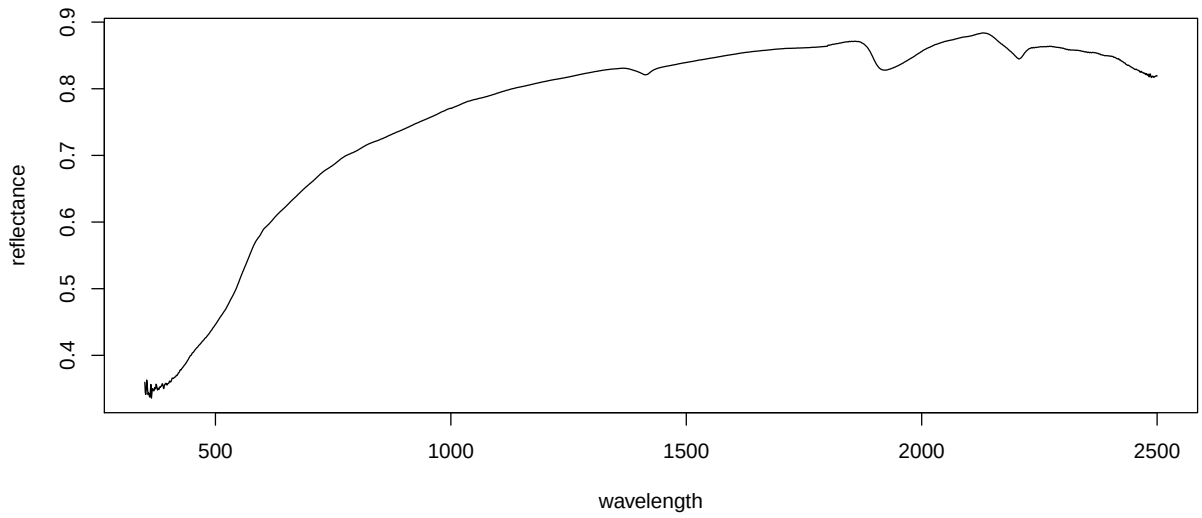
```
## [1]  389 2152
```

Next, we just want to curate the dataset a little. The first column of the dataset has spectral labels which we do not need. Secondly we can add column names to indicate the wavelength that each column of data represents.

```r
nc <- ncol(spectra)
spectra <- spectra[, 2:nc] #remove first column
wavelength <- seq(350, 2500, by = 1) #wavelength sequence
colnames(spectra) <- wavelength #append column names
```

We get some sense of the data we are working with by plotting the first spectrum.

```r
plot(wavelength, spectra[1, ], type = "l" ,ylab= "reflectance")
```

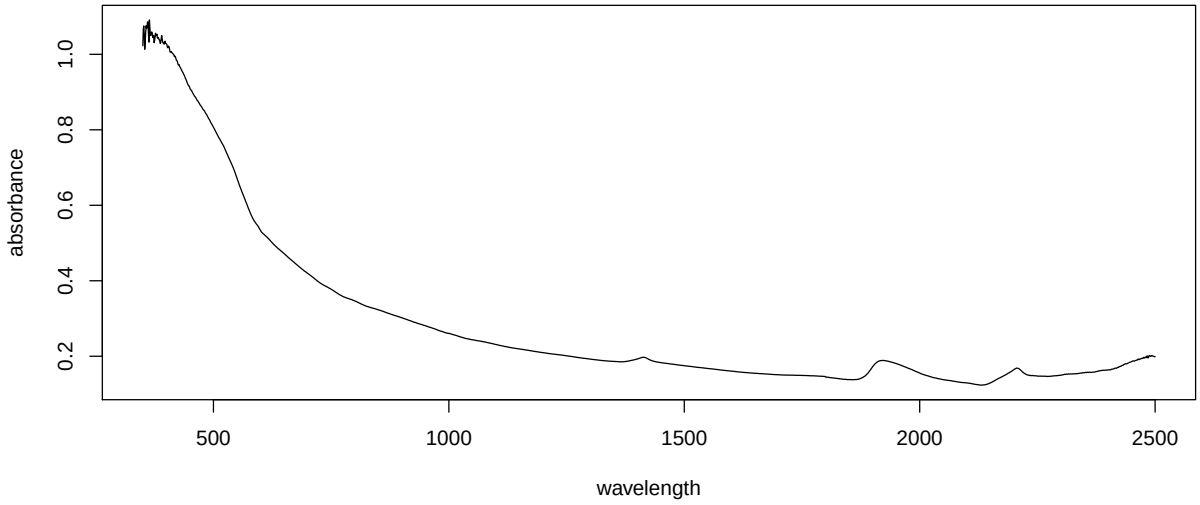


Converting data from reflectance values to absorbance values.

Soil spectral inference modeling seeks to derive the empirical relationship between measured soil properties and the associated spectral absorbance features of soil. If spectra is recorded as reflectance units, than we need to perform a conversion. This is done by calculating $\log(1/R)$ where R is the measured reflectance for a given wavelength. The script below does the conversion very efficiently for every spectrum in the data set. We call the new object (signifying the absorbance values) as spectraA .

```r
spectraA <- log(1/spectra)

#Plot first spectrum
plot(wavelength, spectraA[1, ], type = "l", ylab = "absorbance")
```



Spectral trimming

Trimming spectra is a generic procedure where we may wish to focus an analysis upon a specific spectral range, to remove those wavelengths that have a low signal-to-noise ratio. For example, it is a common convention with vis-NIR data to remove the wavelengths from 350-499nm and 2451-2500nm. Effectively we just want to retain the wavelengths in the spectral range of 500-2450. Without a common procedure for performing such an operation, we need to customize a function or routine ourselves so that this procedure can be easily automated. The function also needs to be generalized i.e. the specification of wavelength regions of interest needs to be flexible, and complimentary to the input spectral data.

A function is just a set of general instructions for given specific inputs that when executed will return a given output or outputs. There is no need to try to interpret the function below. However, note that it requires 2 inputs: spectra (the data set upon which we want to apply the spectral trim) and wavlimits (the region of interest we want to extract). The trimming function is called trimSpec and comes pre-packaged in the spectroscopy package.

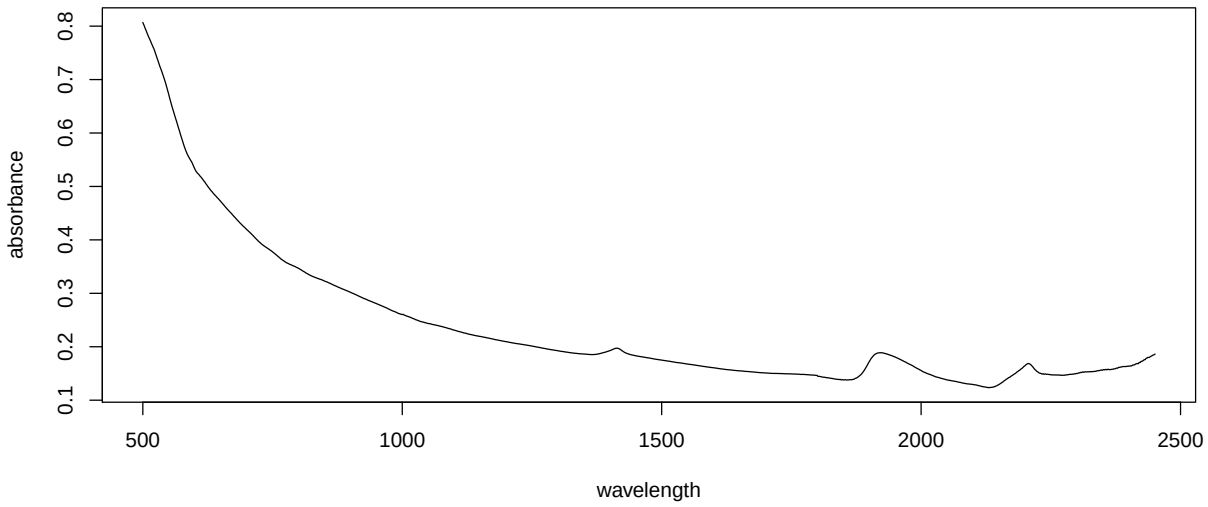
```
#Function for trimming spectra or targeting a specific spectral region of interest
trimSpec <- function(spectra, wavlimits) {
datawavs <- as.numeric(names(spectra))
limits <- which(datawavs %in% wavlimits)
kept_index <- seq(limits[1], limits[2], 1)
trimmed_spectra <- spectra[, kept_index]
kept_names <- datawavs[kept_index]
colnames(trimmed_spectra) <- kept_names
return(trimmed_spectra)}
```

For our purposes we want to trim the spectra or essentially extract all the spectral data between and including 500nm and 2451nm. Note that we specify this region of interest (wavlimits) as range(500:2451) which equates simply as the integers 500, 2451.

```
# TRIMMING THE SPECTRA
specTrim <- trimSpec(spectra = spectraA, wavlimits = range(500:2451))
```

Plotting the spectrum as before is the same, except now we need to specify the new wavelength sequence of the trimmed spectra.

```
wavelength <- seq(500, 2451, by = 1) # new wavelength sequence
plot(wavelength, specTrim[1, ], type = "l", ylab = "absorbance")
```



Savitzky-Golay smoothing filter

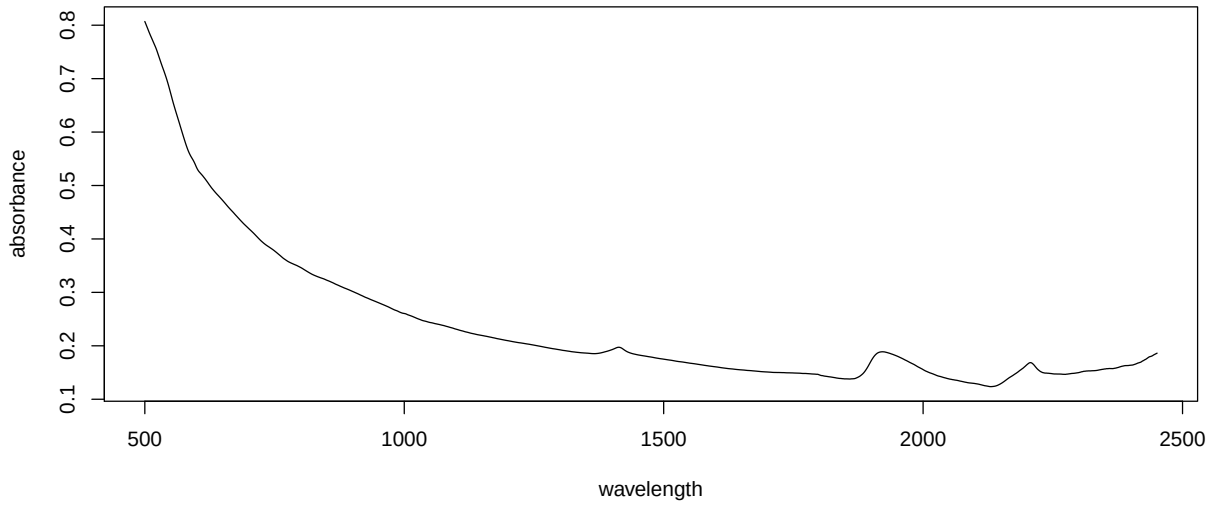
First pretreatment is the Savitzky-Golay smoothing filter. This filter performs a local polynomial regression (of degree k) on a series of values n to determine the smoothed value for each point (this is the filter length). Here we just want to perform the filter with second-order polynomials and use a filter length of 11. Note that the type of polynomial and window size can be changed to suit one's own specifications. The general R scripting form of the Savitzky-Golay filter is shown below. It is implemented in the spectroscopy package via the filter_spectra function.

```
#Function for applying Savitsky-Golay smoothing filter
filter_sg <- function(spectra, n, p, m) {
spectra <- as.matrix(spectra)
## run filter
sg <- aaply(spectra, 1, sgolayfilt, n = n, p = p, m = m)
## arrange appropriately if a single sample
if (nrow(spectra) == 1) {
sg <- matrix(sg, dim(spectra))}
## return data frame
sg<- as.data.frame(sg)
colnames(sg) <- colnames(spectra)
return(sg)
}
```

Essentially this function uses the sgolayfilt function from the signal package. It uses the aaply function from the plyr package to run the filter over the entire spectra collection. The parameters that it needs to operate with include: The spectra; a value for n i.e the window size; a value for p i.e the polynomial order; and a value

for m which is whether or not the derivative of the polynomial is returned. To return the first derivative we would set the value of m to 1. However, we just want to filter our spectra without returning the derivatives, in which case we set m to 0. So let's run the function and make a plot.

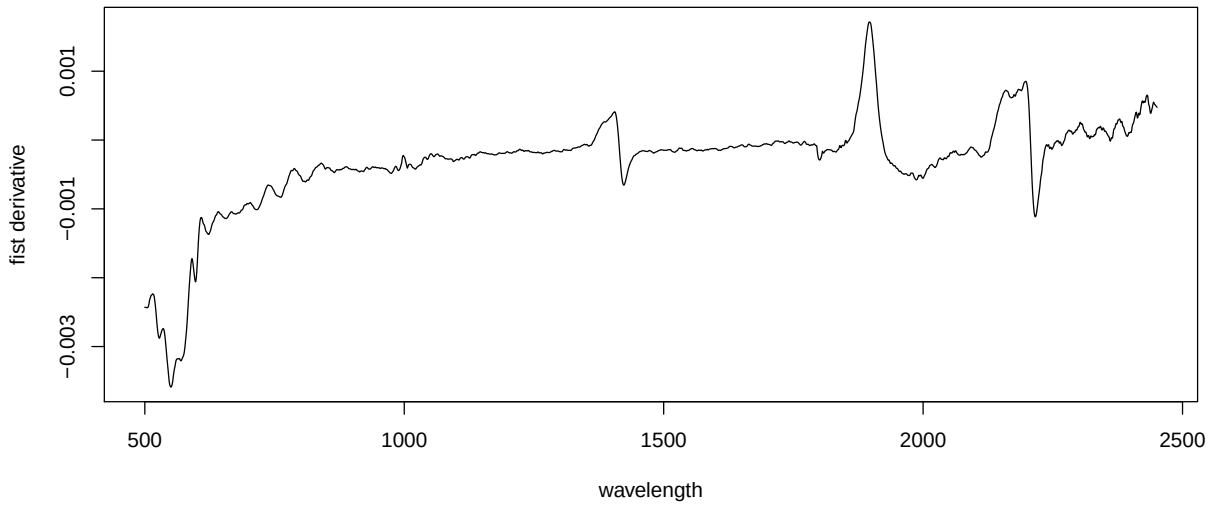
```
#load R libraries necessary to use filter_sg
library(signal);library(plyr)
#filter spectra dataset
spec_filtered <- filter_spectra(specTrim,type="S-Golay" ,n = 11, p = 2, m = 0)
#plot first spectrum
plot(wavelength, spec_filtered[1, ], type = "l",ylab = "absorbance")
```



First and second order derivatives

Continuing on from just smoothing the spectra with the Savitsky-Golay filter, taking the derivatives of the smoothing functions provides means of accentuating the regions of absorbance. Using the filter_spectra function again, we just need to specify 1 or 2 to indicate whether we want to take the first or second derivative of the smoothed spectra respectively.

```
#Take first derivative of spectra
spec_deriv1 <- filter_spectra(specTrim,type="S-Golay" ,n = 11, p = 2, m = 1)
#plot first spectrum
plot(wavelength, spec_deriv1[1, ], type = "l", ylab="fist derivative")
```



Standard normal variate transformation

Standard normal variate transform (SNV), also known as z-transformation or as centering and scaling, normalizes each spectrum (which we will define as ρ) to zero mean and unit variance by subtracting the mean of the spectrum ($\overline{\rho}$) and dividing the difference by its standard deviation (σ_ρ):

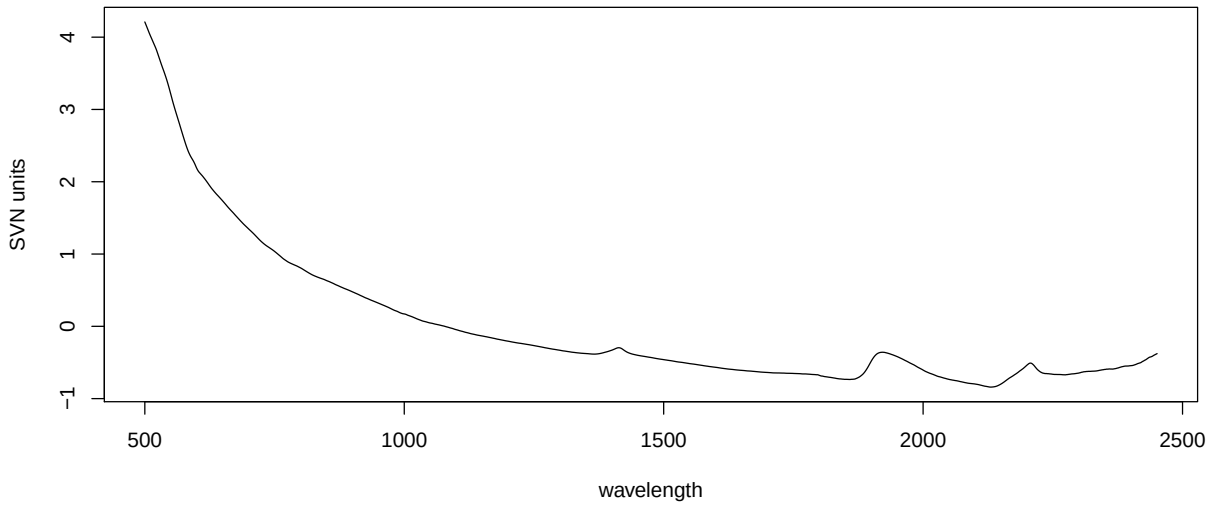
$$SVN = \frac{\rho - \overline{\rho}}{\sigma_\rho}$$

This is algorithm can be applied to each soil spectrum one at a time. Put into a function, it can be scripted as follows:

```
#Function for applying Standard Normal Variate Transformation
snvBLC <- function(spectra) {
spectra <- as.matrix(spectra)
snvMat <- matrix(NA, ncol = ncol(spectra), nrow = nrow(spectra))
for (i in 1:nrow(spectra)) {
snvMat[i, ] <- (spectra[i, ] - mean(spectra[i, ]))/sd(spectra[i,])}
snvMat<- as.data.frame(snvMat)
colnames(snvMat) <- colnames(spectra)
return(snvMat)}
```

This function performs SNV for all the spectra in the collection and outputs a new SNV transformed spectra data set. All it requires is the spectra table for input. We can thus run it as follows via the filter_spectra function.

```
spec_snvC <- filter_spectra(spec_filtered, type = "SNV")
#plot first spectrum
plot(wavelength, spec_snvC[1, ], type = "l",ylab = "SVN units")
```



Multiplicative scatter correction

Multiplicative scatter correction (MSC) works by correcting each spectra to an *ideal* spectrum so that baseline and amplification effects are at the same average level in every spectrum. As this ideal spectrum is unknown, the mean spectrum of a given spectral library or data set (which we define as $\overline{x}$) is used. This spectrum represents the mean scattering and offset. Each spectrum is then fitted to the mean spectrum using the least squares method:

$$x_i = a_i + b_i\overline{x} + e_i$$

Ideally e_i contains the important information, because scattering and offset are represented by the coefficients a_i and b_i. The MSC spectrum is calculated by determining the coefficients for each spectrum and then performing the transformation as follows:

$$MSC = \frac{x_i - a_i}{b_i}$$

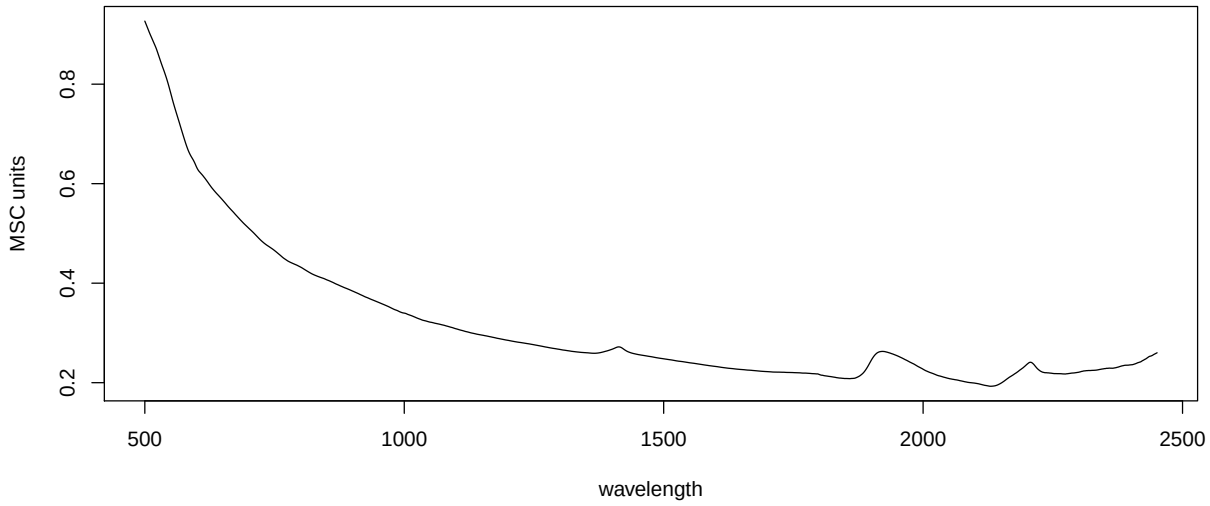
The MSC function can be scripted as follows:

```
#Function for applying Multiplicative Scatter Correction
mscBLC <- function(spectra) {
#first calculate a mean spectrum.
meanSpec <- as.matrix(colMeans(spectra))
mscMat <- matrix(NA, ncol = ncol(spectra), nrow = nrow(spectra))
spectra <- as.matrix(spectra)
for (i in 1:nrow(spectra)) {
# determine the slope and intercept co-efficents
specLM <- lm(spectra[i, ] ~ meanSpec)
specCE <- t(as.matrix(specLM$coefficients))
# Adjust the spectra
mscMat[i, ] <- t(as.matrix((spectra[i, ] - specCE[1, 1])/specCE[1,2]))}
mscMat<- as.data.frame(mscMat)
```

```
colnames(mscMat) <- colnames(spectra)
return(mscMat)}
```

As you may note above, the first part of this function generates the mean spectrum meanSpec. Then for each spectrum we derive the coefficients using the lm function. The coefficients are saved in the specCE object, which are then used for the correction calculation. The MSC algorithm is also implemented in the filter_spectra by specifying 'MSC' in the type option.

```
spec_mscC <- filter_spectra(spec_filtered, type = "MSC")
#Plot first spectrum
plot(wavelength, spec_mscC[1, ], type = "l", ylab="MSC units")
```



Continuum Removal

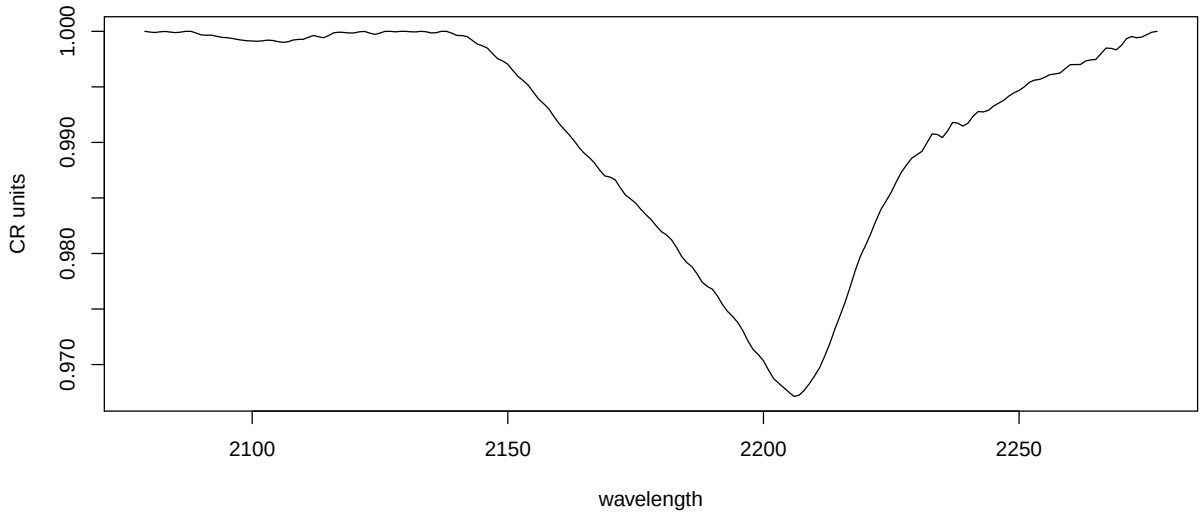
Convex hull or continuum removal (CR) is one type of baseline method that works by fitting a convex hull to each spectrum and computing the deviations from the hull (Clark and Roush 1984). For absorbance spectra, CR gives value of 0 to all parts of the spectrum that lie on the convex hull and values between 0 and 1 to regions inside absorption bands. The opposite is true if we are dealing with reflectance data. Essentially CR accentuates the absorption bands in the spectra while minimizing brightness differences (Buddenbaum and Steffens 2012). The function for applying CR involves a number of internal processes that are not important for understanding the main points of this algorithm, so we just apply the function and plot the first spectrum for illustrative purpose. Computing the convex hull is also implemented in the filter_spectra function via the 'C-hull' option for the type parameter. An addition parameter specType also needs to be specified, which we indicate as either 0 if the data is in absorbance units or 1 if the data is in reflectance units.

Note that the convex hull does not always work as intended if applied to the full spectrum range of the data because of issue with defining the hull points, then approximating the associated continuum line. The convex hull is usually applied when we are dealing with discrete regions of a spectrum. In the example below we will use filter_spectra function upon a specific NIR wavelength region where the secondary clay mineral Kaolinite can be detected. This region is between 2079-2277nm (Clark et al. 2007). We will apply this function to the reflectance data. First we trim the spectral library to the specified region, then apply the chBLC function.

```r
# First trim reflectance spectra to region
specTrim.r <- trimSpec(spectra = spectra, wavlimits = range(2079:2277))

# Apply CR function
spec_chC <- filter_spectra(specTrim.r, type = "C-hull", specType = 1)

#Plot first spectrum
plot(seq(from = 2079,to = 2277,by = 1), spec_chC[1, ], type = "l", ylab="CR units", xlab="wavelength")
```



If you are interested in further baseline and continuum removal algorithms, it is work checking out such R packages as baseline.

Spectral or dimension reduction

Resampling and averaging

Despite all the filtering and baseline corrections we may perform, some models we try to use for calibrating soil data with spectra are difficult to fit and validate when there are many wavelengths to consider. So we may want to reduce the dimensions of our spectra, but at the same time keep the important absorbance features. One such technique that could be used in tandem with the baseline corrections is simple aggregation through resampling and averaging. Essentially with a moving window of n size or wavelength we calculate the average. Thus the bigger the window, the more detail is lost from the spectrum.

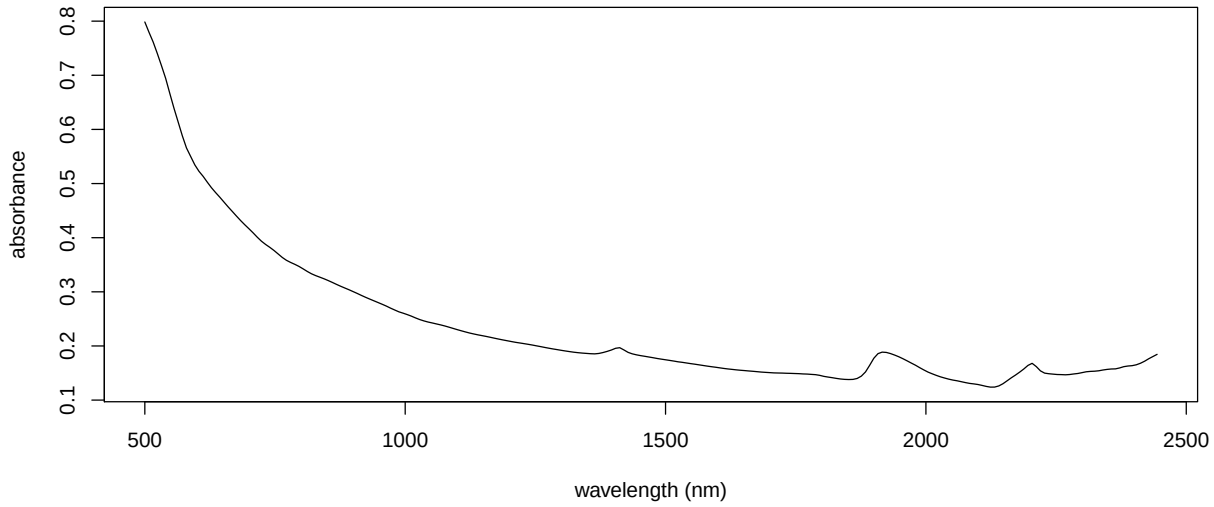
The resampling and averaging function is called compSpec from the spectroscopy package, and it requires two inputs: the spectra that we want to resample and average, and window size. The window size can be arbitrarily selected. However, if the spectra range (though more importantly the number of columns) is not divisible by the window size, then an error will be produced, to which you will need to find an appropriate window size, or implement spectral trimming that will allow you to use that requested window size. In the following example we use the specTrim spectra data set from before (number of columns = 1952) and use a window size of 8 to reduce the column-wise dimension of the spectra library to 244.

```r
#Run Function
spec_Comp <- compSpec(spectra = specTrim, window = 8)

#dimension of spectra library
dim(spec_Comp)
```

```
## [1] 389 244
```

```r
#Plot first spectrum
plot(as.numeric(names(spec_Comp)), spec_Comp[1, ], type = "l", xlab="wavelength (nm)", ylab= "absorbanc
```



Wavelets

Wavelets are a naturally useful tool for processing spectra in terms of filtering and performing dimension reduction. More importantly, with wavelets we can investigate distinct scales of variation within each spectrum by changing the support of the wavelet filtering function. Some further background on wavelets can be found in Viscarra Rossel and Lark (2009) for example. However for our purposes here, we just want to reduce our spectra down to a few important elements without losing the important absorbance features. We can run the wavelet function over the spectra for which it will compute the wavelet coefficients and a smoothed approximation of the data at each scale. Scale is represented as the support of the filter, where for example the increasing dilation of the support will result in smoother and more general representations of the original spectra. However, if we want to, we could reconstruct the original spectra from any scale by multiplying the wavelet coefficients and the smoother signal at each particular scale. What we are using the wavelet functions for here is to simply reduce the dimensions of our original spectra down to a few components i.e. just return the smoothed or more generalized spectra. Here we use the wavethresh package and its wd or wavelet. The help file for the wd function describes in more detail the parameters necessary for the wavelet filtering.

We have made a customized function that runs on top of the wd function so that it returns spectra at the requested scale. This wrapper function is implemented in the filter_spectra function of the spectroscopy package. To run the wavelet implementation we need to ensure our spectra in terms of numbers of wavelengths (the columns) is dyadic or in other words we just need a number for x in 2^x. Where vis-NIR spectra instruments output 1nm spectral resolution data within the 350-2500nm range (2151 wavelengths), it is most appropriate

to find the nearest dyadic number which is 2048 or $2^(11)$. A viable spectra range we may work with could be between and including 404nm and 2451nm in this specific case with this particular vis-NIR data. Essentially we need to do this procedure because the support of the wavelets are dilated by a factor of 2^x for each increasing scale.

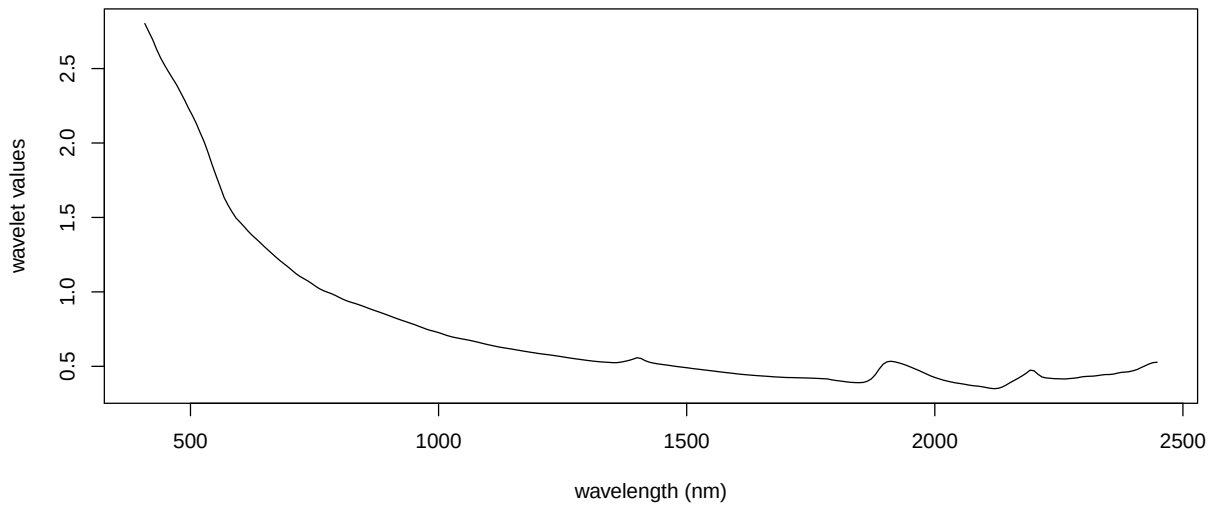
With the appropriately trimmed spectra, the filter_spectra function requires two inputs: the spectra and an integer (less than 11 in this case) for res which indicates the scale that we want to output the smoothed spectra. The example below uses the original unfiltered absorbance spectra spectraA from earlier. We will set the resolution res to 8. Keep in mind that setting res to 11 in this case will potentially return the original spectra table, yet the function does not allow this to occur because it is a redundant operation.

```
library(wavethresh)
#First Trim the spectraA object to 404-2451 spectral range
specTrim2 <- trimSpec(spectraA, wavlimits = range(404:2451))

#Run wavelet smooth function
res <- 8 # the resolution of the decomposed spectra # res =11 when you want to return the original data
spec_wavelet <- filter_spectra(specTrim2, type = "Wavelet", res = 8)
dim(spec_wavelet)
```

```
## [1] 389 256
```

```
#Plot first spectrum
plot(as.numeric(names(spec_wavelet)), spec_wavelet[1, ], type = "l", xlab="wavelength (nm)", ylab= "wave
```



Exercise

To evaluate the differences between the various filtering, smoothing and baseline corrections we have just investigated (and to slightly challenge yourself), plot the spectra which correspond to the maximum, minimum, and median soil carbon concentrations using each of the different pretreatments. Hint: you will first need to merge the relevant soil data to their corresponding vis-NIR spectra as a first step.

References

Buddenbaum, H., and M. Steffens. 2012. "The Effects of Spectral Pretreatments on Chemometric Analyses of Soil Profiles Using Laboratory Imaging Spectroscopy." *Applied and Environmental Soil Science*, Article ID: 274903.

Clark, R.N., and T.L Roush. 1984. "Refectance Spectroscopy - Quantitative Analysis Techniques for Remote Sensing Applications." *Journal of Geophysical Research*, no. 89(NB7): 6329–40.

Clark, R.N., G.A. Swayze, R. Wise, K.E. Livo, T.M. Hoefen, R.F. Kokaly, and S.J. Sutley. 2007. "USGS Digital Spectral Library Splib06a." *Geological Survey, Data Series* 231.

Viscarra Rossel, R., and R.M. Lark. 2009. "Improved Analysis and Modelling of Soil Diffuse Reflectance Spectra Using Wavelets." *European Journal of Soil Science* 60 (3): 453–64.