Introduction to spectral processing with 'spectrolab'

Anna Schweiger

Install spectrolab

```
install.packages("spectrolab")
```

```
library("spectrolab")
```

The spectra class

`spectrolab` defines a new S3 class called `spectra` that holds all of the different components of a spectral data.

Without diving too much into its implementation, a `spectra` object holds the important information needed for most spectral data sets: reflectance, wavelengths, filen ames, metadata etc. The class has a bunch of requirements in terms of both format and values.

Read and inspect data

Our spectral data were measured with an instrument called ASD.

```
spec <- read_spectra("./example_data/", format="asd")
```

`spectrolab` can also read other file formats, but let's not worry about that for now. You can always look at `spectrolab's` help for more information.

```
help(read_spectra)
```

You can see what a `spectra` object contains by typing

```
spec
```

```
## spectra object
## number of samples: 9
## bands: 350 to 2500 (2151 bands)
## metadata: none
##
##                 350                 351                 352
## 103_SALHU00000 0.0553291944884881 0.044910506095899  0.0523716595312952
## 103_SALHU00001 0.0765274048215119 0.0717495824057209 0.0696882043408338
## 103_SALHU00002 0.0685952189080724 0.0611274666766303 0.0582451955497961
```

```
## 103_SPITO00000 0.0771597809774897 0.0708588590589189 0.0699624301626821
## 103_SPITO00001 0.059278354402922  0.0561048473944045 0.0479089283439103
##                353                354                355
## 103_SALHU00000 0.0492809575754814 0.0450551981658801 0.0472716432998146
## 103_SALHU00001 0.065932651726352  0.0641965595587686 0.0658921140029544
## 103_SALHU00002 0.0635829576472747 0.062906334767177  0.0529386103808933
## 103_SPITO00000 0.0683834918526438 0.0603142651099755 0.0496831080863538
## 103_SPITO00001 0.0404531509557369 0.0419312955805392 0.051260307662551
##                356                ...
## 103_SALHU00000 0.048303546854188
## 103_SALHU00001 0.0658540579024284
## 103_SALHU00002 0.0553967379235214
## 103_SPITO00000 0.0592112258665227
## 103_SPITO00001 0.0502698726818367
```

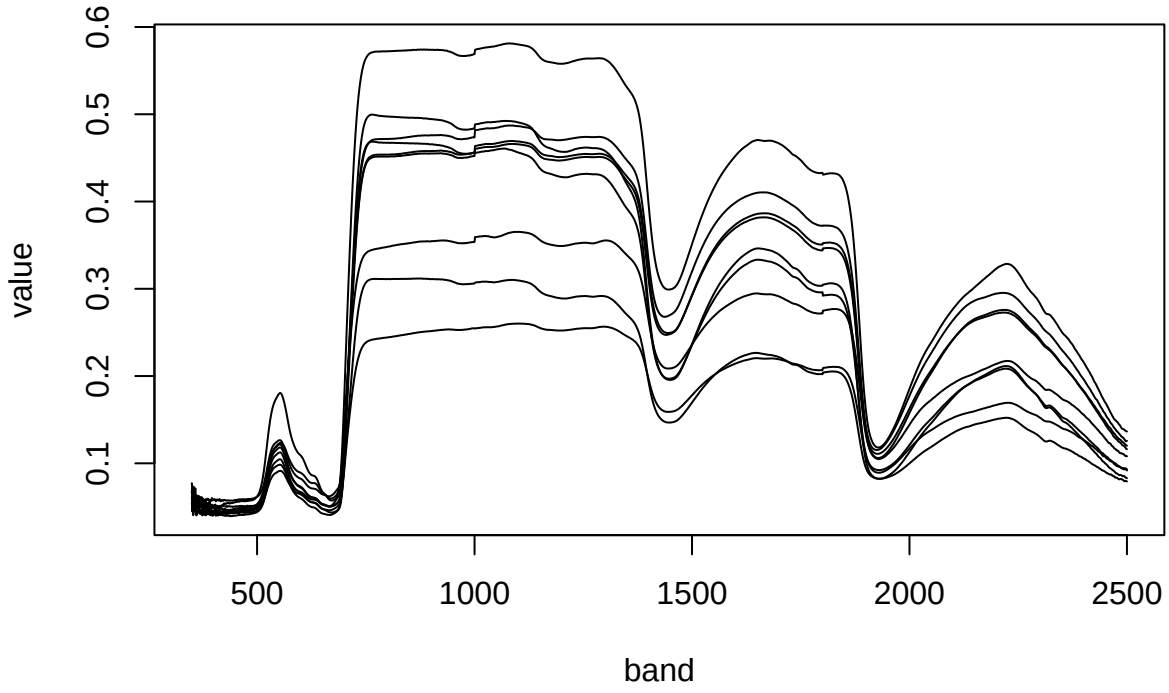
Our `spectra` object contains 9 samples, > 2000 bands, and no metadata. To access the individual components try

```
bands(spec)
tail(bands(spec))

names(spec)
meta(spec)
```

Let's plot our spectra. It's as simple as

```
plot(spec)
```



You can add spectral regions and quantiles to the plot. How? Try to figure it out yourself. Hint: you can search the help page, or type ***comma*** after ***spectrolab*** followed by the ***TAB*** key

```
help(package="spectrolab")
```

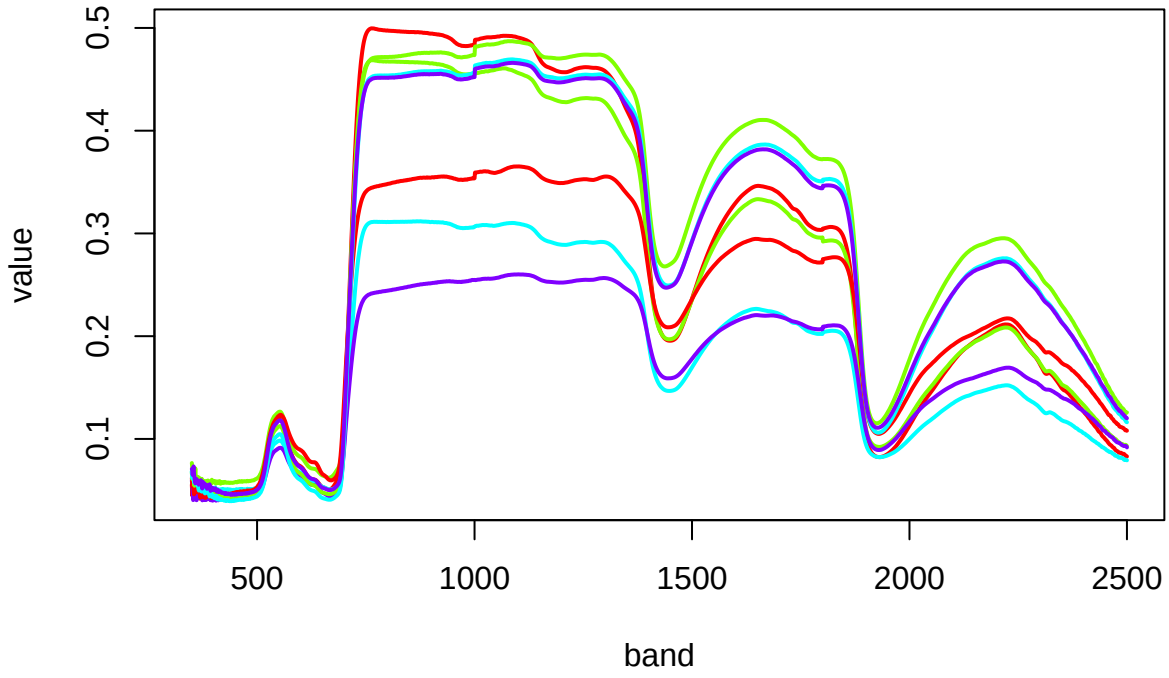
Try also `spectrolab's` interactive plotting option

```
plot_interactive(spec)
```

If you'd like to remove a spectrum for some reason, click on it and note the number. Let's create a new `spectra` object and remove the spectrum with the highest reflectance

```
spec2 <- spec[-4]

# standard plotting commands can be added
plot(spec2, col=rainbow(n=length(spec2)), lwd=2)
```



Adding metadata

Easy! How about adding some metadata? One option is to edit the `spectra` object.

```
meta(spec2, label = "some_cool_data") <- c(3,7,4,2,6,9,8,7)
```

Adding a dummy N content

```
n_content <- rnorm(n = nrow(spec2), mean = 2, sd = 0.5)
meta(spec2, label = "N_percent") = n_content
# and check
meta(spec2)
```

```
##   some_cool_data N_percent
## 1              3  1.769109
## 2              7  1.604207
## 3              4  2.343902
## 4              2  1.776716
## 5              6  1.833704
## 6              9  2.611718
## 7              8  2.558789
## 8              7  2.049988
```

Looks good! You can also import a metadata table and merge it to your spectral data.

```
meta_csv <- read.csv("./example_data/metadata.csv")
meta(spec2) <- meta_csv[,c(2,3)]

meta(spec2)
```

```
##   some_cool_data N_percent some_chemistry other_thing
## 1              3  1.769109            5.0        57.0
## 2              7  1.604207            3.0        22.0
## 3              4  2.343902            6.0        59.0
## 4              2  1.776716            9.0       100.0
## 5              6  1.833704           10.0       123.4
## 6              9  2.611718            3.3        99.9
## 7              8  2.558789            7.0        22.0
## 8              7  2.049988            4.0        34.0
```

Awesome! You can query your **spectra** object and metadata by name or index

```
meta(spec2, sample=c(1,3))
```

```
##   some_cool_data N_percent some_chemistry other_thing
## 1              3  1.769109              5          57
## 3              4  2.343902              6          59
```

```
names(spec2)
```

```
## [1] "103_SALHU00000" "103_SALHU00001" "103_SALHU00002" "103_SPITO00001"
## [5] "103_SPITO00002" "103_VITRI00001" "103_VITRI00002" "103_VITRI00003"
```

```
meta(spec2,sample="103_SALHU00000")
```

```
##   some_cool_data N_percent some_chemistry other_thing
## 1              3  1.769109              5          57
```

Bonus question: What's the difference in the output of these two functions

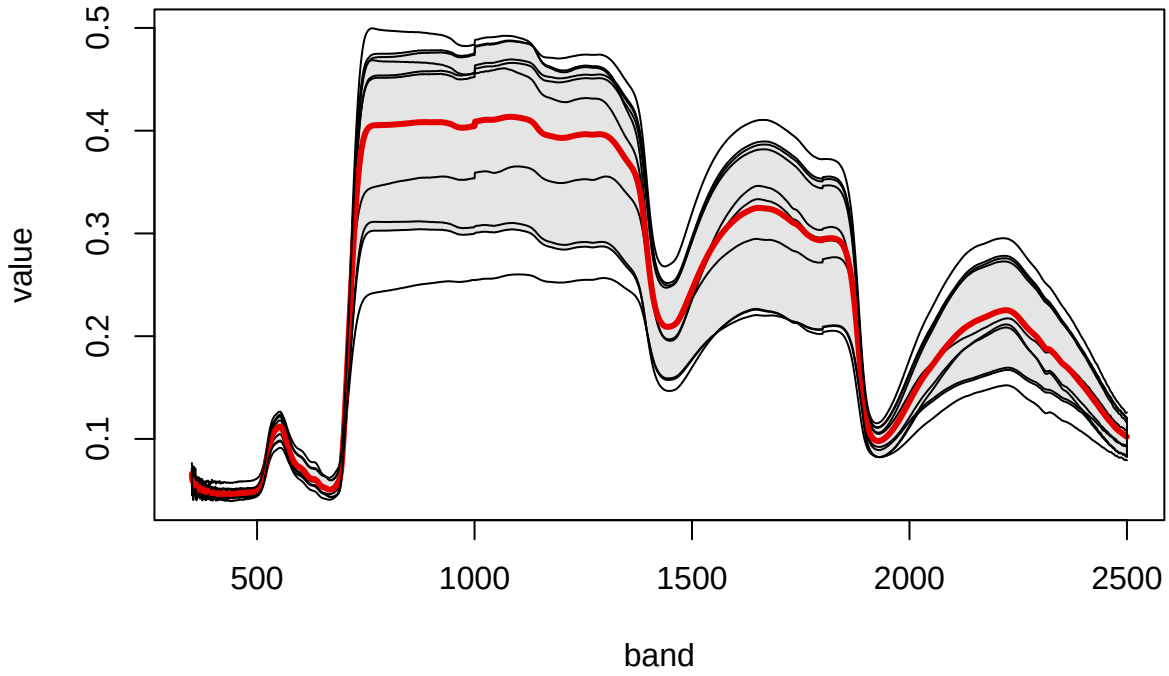
```
meta(spec2,label = 2)
meta(spec2,label = 2, simplify = TRUE)
```

Basic calculations

Sometimes you might want to calculate a mean spectrum, or other things

```
spec_mean <- mean(spec2)

plot(spec2)
plot(spec_mean, col="red", lwd=3, add=T)
plot_quantile(spec2,total_prob = 0.75,add=T)
```



Conveniently, this calculates also the mean of your metadata

```
meta(spec_mean)
```

```
##   some_cool_data N_percent some_chemistry other_thing
## 1           5.75  2.068517         5.9125     64.6625
```

What about minimum and maximum?

```
spec_min <- min(spec2)
meta(spec_min)
```

This does not work because `spectrolab` does not know if you want a minimum spectrum, a minimum reflectance value, a minimum values of a specific metadata element, etc. But there are better ways to calculate each of those... can you figure them out?

Some post-processing options

You might want to cut off noisy regions at the beginning and end of the spectrum. Simple sub-setting does the trick. Use `plot_interactive` to decide which regions are "good"

```
spec_trim <- spec2[ , bands(spec2, 500, 2400)]
```

You can also re-sample wavelengths, e.g. to reduce the amount of data

```
spec_sub <- resample(spec2, new_bands = seq(400, 2400, 10))
```

Or select specific samples

```
spec3 <- spec2[meta(spec2,"N_percent")>2,]
meta(spec3, "N_percent")
```

```
##   N_percent
## 1  2.343902
## 2  2.611718
## 3  2.558789
## 4  2.049988
```

```
spec4 <- spec2[grepl("SPITO",names(spec2))]
spec4
```

```
## spectra object
## number of samples: 2
## bands: 350 to 2500 (2151 bands)
## metadata (3 of 4): some_cool_data, N_percent, some_chemistry, ...
##
##                 350                 351                 352
## 103_SPITO00001 0.059278354402922  0.0561048473944045 0.0479089283439103
## 103_SPITO00002 0.0618781277965329 0.0540265308384619 0.0552005150509939
##                 353                 354                 355
## 103_SPITO00001 0.0404531509557369 0.0419312955805392 0.051260307662551
## 103_SPITO00002 0.0544231961151201 0.050287062871541  0.0460093041280549
##                 356                 ...
## 103_SPITO00001 0.0502698726818367
## 103_SPITO00002 0.050276376903866
```

Export data

You can of course export and re-import your data. Just be careful how `spectrolab` imports tabular data. You will need to specify names, bands and metadata

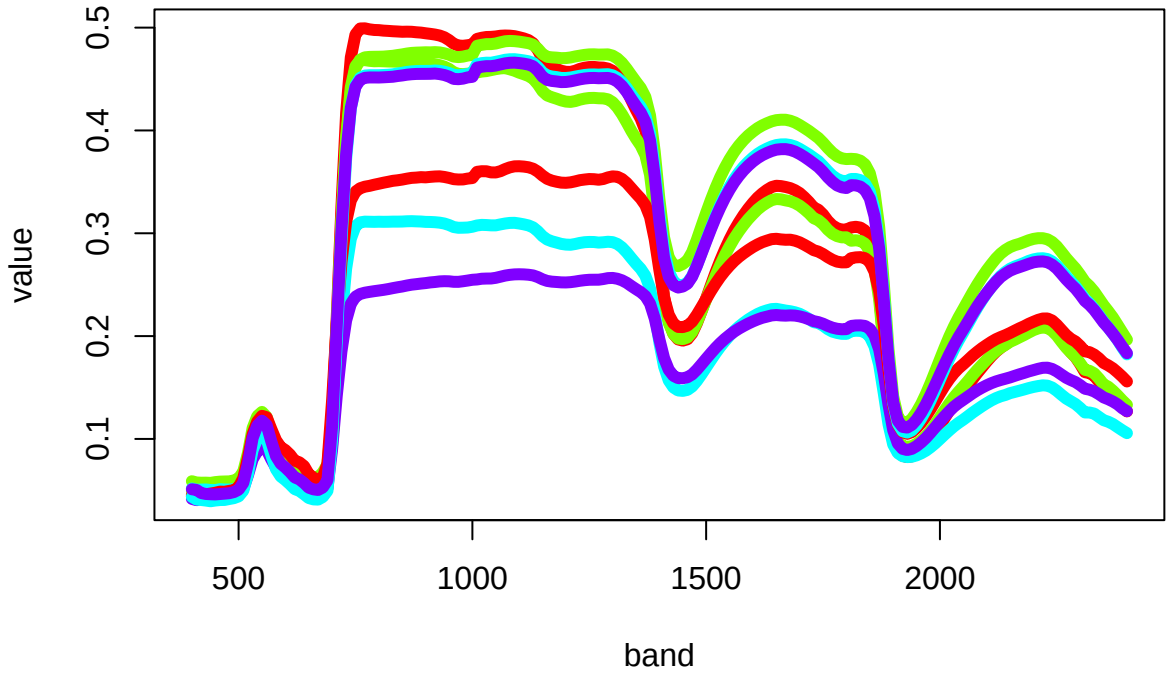
```
write.csv(spec_sub, "./example_data/export_spec_sub.csv", row.names = F)
```

```
### Data should be the same as
spec_csv <- read.csv("./example_data/export_spectra.csv")
```

`spectrolab's` bands names need to be numeric. So we need to modify our column names first

```
names(spec_csv) <- gsub("X","",names(spec_csv))
spec5 <- as_spectra(spec_csv,name_idx = 1, meta_idxs = 2:5)

plot(spec5, col=rainbow(n=length(spec5)), lwd=6)
```



Beautiful!!

Wishing you lots of fun with all your spectral endevours!