`pavo`: **P**erceptual **A**nalysis, **V**isualization and **O**rganization of Spectral Color Data in R

Rafael Maia[1], Paul-Pierre Bitton[2], and Chad Eliason[1]

[1]Integrated Bioscience PhD Program, University of Akron, Akron OH
[2]Department of Biological Sciences, University of Windsor, Windsor ON

November 29, 2012

Contents

Introduction

`pavo` is an R package developed with the goal of establishing a flexible and integrated workflow for working with spectral color data. It includes functions that take advantage of new data classes in order to work seamlessly from importing raw data to visualization and analysis.

Although `pavo` deals largely with spectral reflectance data from bird feathers, it is meant to be applicable for a range of taxa and applications. It provides flexible ways to input spectral data from a variety of equipment manufacturers, process these data, extract variables and produce publication-quality graphics.

`pavo` was written with the following workflow in mind:

1. **Organize** spectral data by inputting files, processing spectra (e.g., to remove noise, negative values, smooth curves, etc.)

2. **Analyze** the resulting files, either using typical tristimulus color variables (hue, saturation, brightness) or using visual models based on perceptual data from the taxon of interest.

3. **Visualize** the output, with multiple options provided for exploratory analyses.

Below we will show the main functions in the package in an example workflow. The development version of `pavo` can be found on github.

1 Dataset Description

The data used in this example is available from github by clicking here[1]. You can download and extract it to follow the vignette.

The data consists of reflectance spectra obtained using Avantes equipment and software from seven bird species: Northern Cardinal (*Cardinalis cardinalis*), Wattled Jacana (*Jaca na jacana*), Baltimore Oriole (*Icterus galbula*), Peach-fronted Parakeet (*Aratinga aurea*), American Robin (*Turdus migratorius*), and Sayaca Tanager (*Thraupis sayaca*). Several individuals were measured (sample size varies by species), and 3 spectra were collected from each individual.

The samples do not have the same sample sizes and have additional peculiarities that should emphasize the flexibility `pavo` offers, as we'll see below.

2 Organizing and Processing Spectral Data

2.1 Importing Data

The first thing we need to do is import the spectral data into R using the funciton `getspec()`. Since the spectra were obtained using Avantes software, we will need to specify that the files have the "`.ttt`" extension. Further, the data is organized in subdirectories for each species. `getspec` does recursive sampling, and may include the names of the subdirectories in the spectra name if desired. A final issue with the data is that it was collected using a computer with international numbering input, which means it uses commas instead of periods as a decimal separator. We can specify that in the function call.

I have downloaded the file and placed it in a directory called "`~/github/pavo/vignette_data`". By default, `getspec` will search for files in the current folder, but a different one can be specified:

```
> specs <- getspec("~/github/pavo/vignette_data/", ext="ttt", decimal=",",
+                  subdir=T, subdir.names=F)
> specs[1:10,1:4]
```

[1] in case you can't click the printed version you have:
https://github.com/rmaia/pavo/blob/master/vignette_data/vignette_data.zip

```
    wl cardinal.0001 cardinal.0002 cardinal.0003
1  300        5.7453        8.0612        8.0723
2  301        6.0181        8.3926        8.8669
3  302        5.9820        8.8280        9.0680
4  303        6.2916        8.7621        8.7877
5  304        6.6277        8.6819        9.3450
6  305        6.3347        9.6016        9.4834
7  306        6.3189        9.5712        9.3533
8  307        6.7951        9.4650        9.9492
9  308        7.0758        9.4677        9.8587
10 309        7.2126       10.6172       10.5396
```

```
> dim(specs) #data has 214 spectra, from 300 to 700 nm
```

```
[1] 401 214
```

When `pavo` imports spectra, it creates an object of class `rspec`, which inherits attributes from the `data.frame` class:

```
> is.rspec(specs)
```

```
[1] TRUE
```

If you already have multiple spectra in a single data frame that you'd like to use with `pavo` functions, you can use the command `as.rspec` to convert it to an rspec object. The function will attempt to identify the wavelength variable or, if it doesn't have one, it can be specified in the function call.

2.2 Processing Data

2.2.1 Averaging Spectra

As previously described, our data constitutes of multiple individuals, and each was measured three times, as is common in order to avoid measurement bias. A good way to visualize the repeatability of our measurements is to plot the spectra of each individual separately. The function `explorespec` provides an easy way of doing so. You may specify the number of spectra to be plotted in the same panel using the argument `specreps`, and the function will adjust the number of panels per page accordingly. We will exemplify this function using only the 12 cardinal individuals measured:

```
> explorespec(specs[,1:37], specreps=3)
> # 36 spectra plus the first (wl) column
```

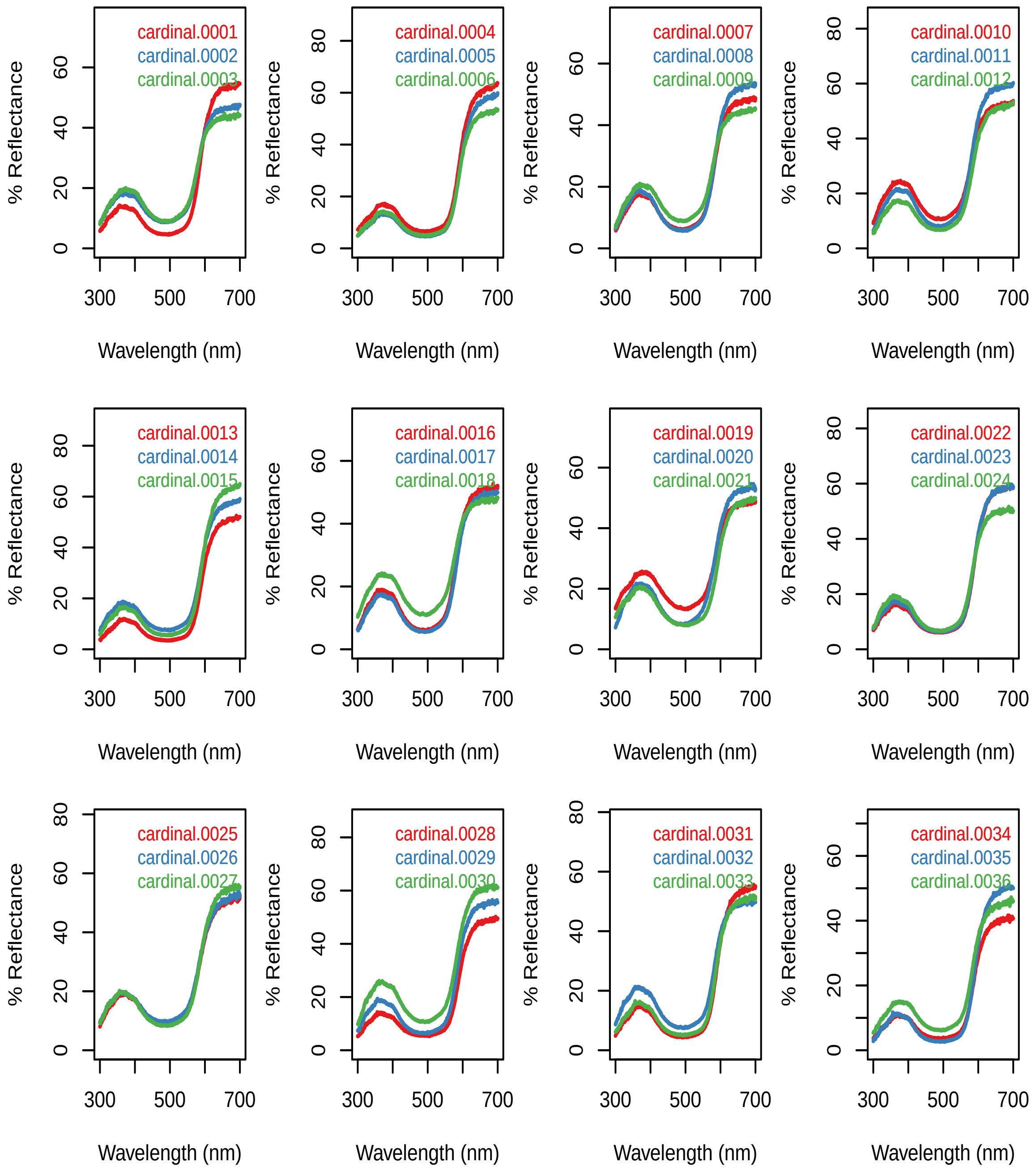

Figure 1: Result from `explorespec`, showing the three measurements for each individual in separate panels

So our first step would be to take the average of each of these three measurements in order to obtain average individual spectra to be used in further analyses. The function `aggspec` does this. The `by` argument can be either a number (specifying how many specs should be averaged for each new sample) or a vector specifying the identities of the spectra to be combined (see below):

```
> mspecs <- aggspec(specs, by=3, FXN=mean)
> mspecs[1:5, 1:4]
```

```
   wl cardinal.0001 cardinal.0004 cardinal.0007
1 300      7.292933      5.676700      6.387233
2 301      7.759200      5.806700      6.698200
3 302      7.959333      5.858467      6.910500
4 303      7.947133      6.130267      7.357567
5 304      8.218200      6.127933      7.195267
```

```
> dim(mspecs) #data now has 72 spectra, one for each individual
```

```
[1] 401  72
```

Now we'll use the `aggspec` function again, but this time to take the average spectrum for each species. However, each species has a different number of samples, so we can't use the `by` argument as before. Instead we will use regular expressions to create a species name vector by removing the numbers that identify individual spectra:

```
> # create a vector with species identity names
> spp <- gsub('\\.[0-9].*$','',names(mspecs))[-1]
> table(spp)
```

```
spp
cardinal   jacana   oriole parakeet    robin  tanager
      12        9        9       13       10       18
```

Instead, we are going to use the `spp` vector we created to tell the `aggspec` function how to average the spectra in `mspec`:

```
> sppspec <- aggspec(mspecs, by=spp, FXN=mean)
> sppspec[1:5, ]
```

```
   wl cardinal   jacana   oriole parakeet    robin
1 300 7.049397 7.334781 3.889693 7.629954 3.981747
2 301 7.254161 7.354033 3.905322 7.746882 3.914297
3 302 7.444275 7.452556 4.126619 7.886877 4.187073
4 303 7.820686 8.085541 4.390685 8.491367 4.507410
5 304 7.843394 7.714526 4.183637 8.658000 4.068800
   tanager
1  9.021043
```

```
2  9.525854
3  9.405980
4 10.199843
5  9.684522
```

```
> explorespec(sppspec, 6)
```

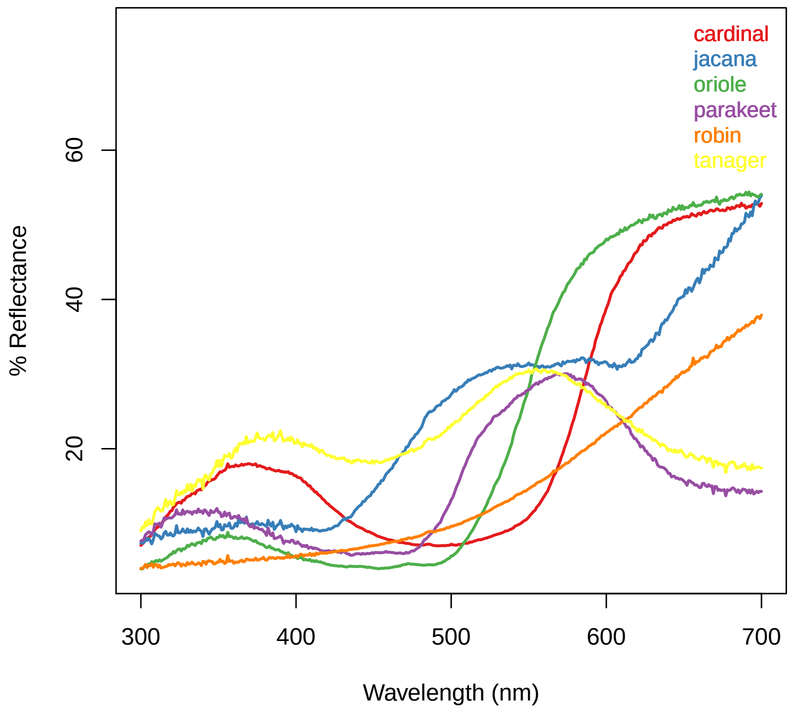

Figure 2: Result from `explorespec` for species means

3 Visualizing Spectral Data

```
> data(sicalis)
> par(mfrow=c(1,2))
> plot(sicalis, type='o', col=spec2rgb(sicalis))
> plot(procspec(sicalis, opt='smooth'), type='o', col=spec2rgb(sicalis))
```

```
processing options applied:
 smoothing spectra with a span of 0.25
```

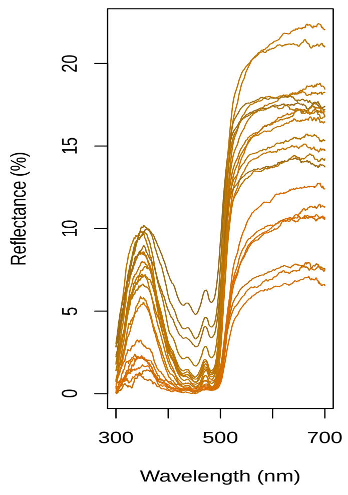

Figure 3: Overlay plot with colors calculated from human color matching functions

```
>
> #plot(sicalis, type='s', col=spec2rgb(sicalis))
```

Colors can be mapped to spectra using `spec2rgb` as shown in Figure 3.

4 Analyzing Spectral Data

4.1 Overview

`pavo` offers two main approaches for spectral data analysis. First, variables can be calculated based on the shape of the reflectance spectra. By using special R classes for spectra data frame objects, this can easily be done using the `summary` function in an rspec object (see below). The second function for spectral shape analysis is `peakshape`, which returns descriptors for peaks in spectral curves, as outlined below.

Second, reflectance spectra can be analyzed accounting for the visual system receiving the color signal, therefore representing reflectance spectra as preceived colors. We have implemented Endler's (REF) segment analysis model, which approximates visual models but does not directly use sensory information; the model of Osorio & Vorobyev (REF), which provides a flexible framework for visual modeling; and the tetrahedral color space (GOLDSMITH, ENDLER, STODDARD) which has been extensively developed to represent colors in the avian vision color space.

4.2 Spectral Shape Analysis

4.2.1 Trichromatic variables

Obtaining trichromatic color variables (related to hue, saturation and value) is pretty straightforward in `pavo`. Since reflectance spectra is stored in an object of class `rspec`, the `summary` function recognizes the object as such and extracts 23 variables, as outlined in MONTGOMERIE. Though outlined in a book chapter on bird coloration, these variables are broadly applicable to any reflectance data, particularly if the taxon of interest has color vision within the UV-human visible range.

The description and formulas for these variables can be found in Table 1.

```
> summary(sppspec)
```

```
               B1       B2       B3      S1.UV  S1.violet
cardinal 8983.870 22.40367 52.96101 0.16618070 0.19132718
jacana   9668.442 24.11083 53.48957 0.09623268 0.11048401
oriole   9107.568 22.71214 54.31982 0.07474281 0.08314578
parakeet 6022.866 15.01962 28.82353 0.16803871 0.18515359
robin    5741.033 14.31679 37.96515 0.08500350 0.10012432
tanager  8516.756 21.23879 29.76561 0.20342675 0.23931496
            S1.blue  S1.green S1.yellow    S1.red        S2
cardinal 0.11784952 0.1900225 0.2517886 0.5332009  7.329535
jacana   0.19475522 0.3082406 0.2479744 0.4079294  6.883918
oriole   0.05721376 0.3256520 0.3630756 0.5491879 12.992914
parakeet 0.14334383 0.4252528 0.3397279 0.2717663  4.768352
robin    0.14429801 0.2675057 0.2668015 0.5099805  9.172749
tanager  0.26109724 0.3199401 0.2430633 0.2238215  3.120773
                S3         S4       S5       S6          S7
cardinal 0.3682957 0.15813546 4004.193 45.73531 -0.38403228
jacana   0.2469875 0.01690805 3312.697 45.71935 -0.46942827
oriole   0.3444943 0.06767440 5232.176 50.13910 -0.70982859
parakeet 0.4467452 0.23727515 1900.192 22.77878 -0.20889035
robin    0.3031191 0.01271259 2531.901 33.82625 -0.57712488
tanager  0.3341862 0.15240014 1105.252 20.22771 -0.08198454
                S8         S9        S10  H1  H2  H3
cardinal 2.0414210 -0.8373693  12.909318 700 425 596
jacana   1.8962165 -0.7357544 112.148743 700 388 504
oriole   2.2075903 -0.9257720  32.620756 700 388 500
parakeet 1.5166018 -0.5816948   6.391743 568 613 502
robin    2.3626975 -0.8148189 185.854859 700 349 500
tanager  0.9523946  0.0359801   6.249303 557 600 428
                 H4  H5
cardinal 1.54995089 617
jacana   0.85480076 463
oriole   1.15606660 538
parakeet 0.59271149 500
robin    1.15020735 637
tanager  0.04509807 512
```

Color Variable	**Equation**	**Description**
`B1`	$\sum_{\lambda=300}^{700} R_\lambda$	Total brightness, total reflectance
`B2`	B_1/n_{wl}	Mean brightness.
`B3`	$R_{\max}$	Intensity.
`S1`		Chroma, spectral purity.
`S2`	$R_{\max}/R_{\min}$	Spectral saturation
`S3`		
`S4`		
`S5`		
`S6`		
`S7`		
`S8`		
`S9`		
`S10`		
`H1`	λ_{Rmax}	Hue: wavelength of peak reflectance
`H2`		
`H3`		
`H4`		
`H5`		

Table 1: The complete set of tristimulus variables calculated by `summary` in `pavo`

4.2.2 Peak shape descriptors

Particularly in cases of reflectance spectra that have a single, discrete peak, it might be useful to obtain variables that describe that peak's properties. The `peakshape` function identifies the peak location (`H1`), returns the reflectance at that point (`B3`), and identifies the wavelengths at which the reflectance is half that at the peak, calculating the wavelength bandwith of that interval (the **Full Width at Half Maximum**, or `FWHM`). The function also returns the half widths, which are useful when the peaks are located near the edge of the measurement limit and half maximum reflectance can only be reliably estimated from one of its sides.

If this all sounds too esoteric, fear not: `peakshape` has the option of returning plots indicating what it's calculating. The vertical continuous red line indicates the peak location, the horizontal continuous red line indicates the half-maximum reflectance, and the distance between the dashed lines is the FWHM:

```
> par(mfrow=c(2,3),mar = c(5, 4, 0.5, 0.5) + 0.1)
> peakshape(sppspec, plot=T)
```

```
               B3  H1 FWHM HWHM.l HWHM.r status
cardinal 52.89508 687  104    100      4     OK
jacana   53.90258 700   NA     93     NA     OK
oriole   54.40384 692  148    141      7     OK
parakeet 30.05703 575  125     66     59     OK
robin    37.91676 700   NA    107     NA     OK
tanager  30.71021 555  232    142     90     OK
```

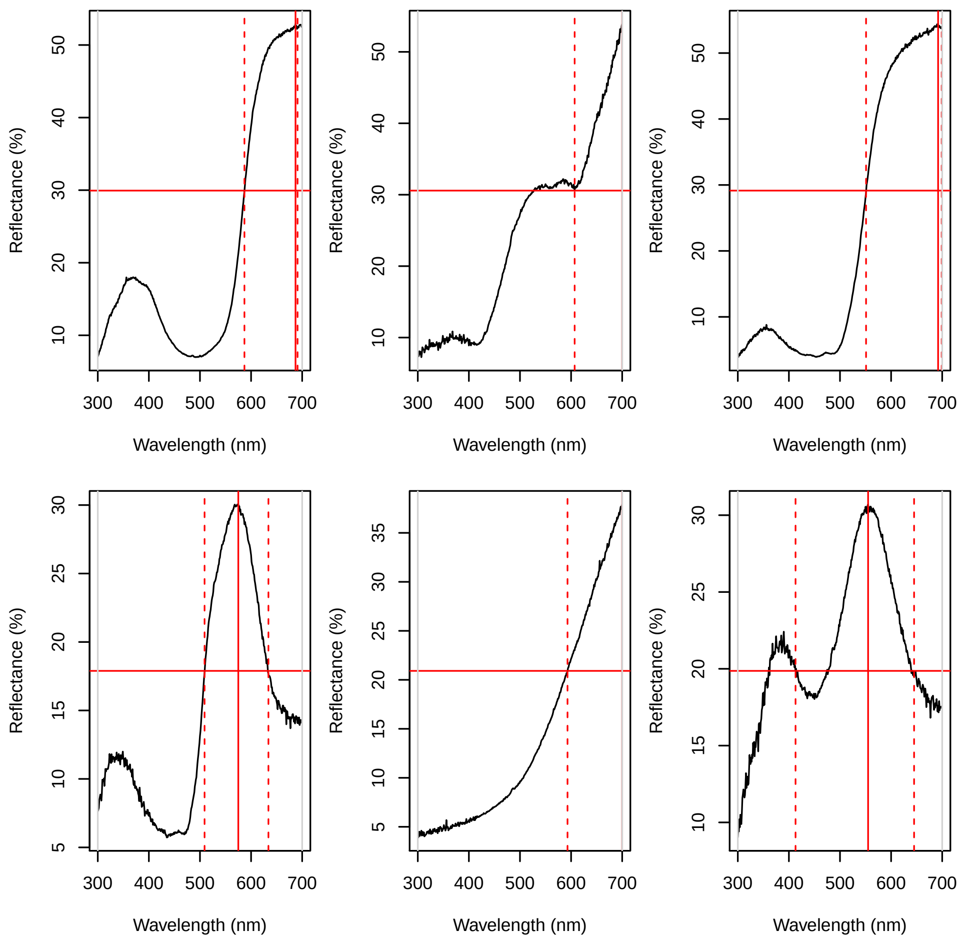

Figure 4: Plots from `peakshape`

As you can see, the variable is meaningless if the curve doesn't have a clear peak. Sometimes , such as in the case of the Cardinal (Figure 4, first panel), there might be a peak which is not the point of maximum reflectance of the entire spectral curve. The half-width can also be erroneously calculated when there are two peaks, as can be seen in the case of the Tanager (Figure 4, last panel). In this case, it's useful to set bounds when calculating the FWHM, using the `bounds` argument. `peakshape` also offers a `select` argument to facilitate subsetting the spectra data frame:

```
> peakshape(sppspec, select=2, bounds=c(300,500), plot=T)
```

```
               B3  H1 FWHM HWHM.l HWHM.r status
cardinal 17.99853 369  100     47     53     OK
```

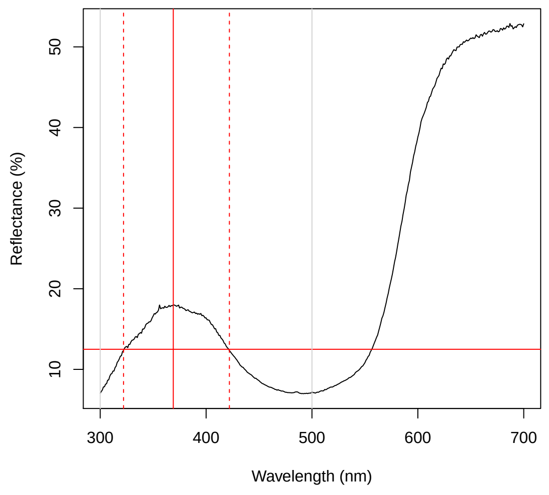

Figure 5: Plot from `peakshape`, setting the bounds to 300-500nm

Examples

```
> hist(rnorm(50))
```

4.3 Segment Classification Model

```
> fakedata1<-  sapply(seq(100,500,by=20), function(x) rowSums(cbind(dnorm(300:700,x,30), dnorm(30
> fakedata1<-data.frame(wl=300:700,fakedata1)
> fakedata1<-as.rspec(fakedata1)
> fakedata1<- procspec(fakedata1, "max")

processing options applied:
 Scaling spectra to a maximum value of 1

> segfd1 <- segclass(fakedata1)
> plot(fakedata1, type='stack', col=spec2rgb(fakedata1))
> plot(segfd1, pch=20, cex=2, col=spec2rgb(fakedata1))
```

More examples

Some more examples:

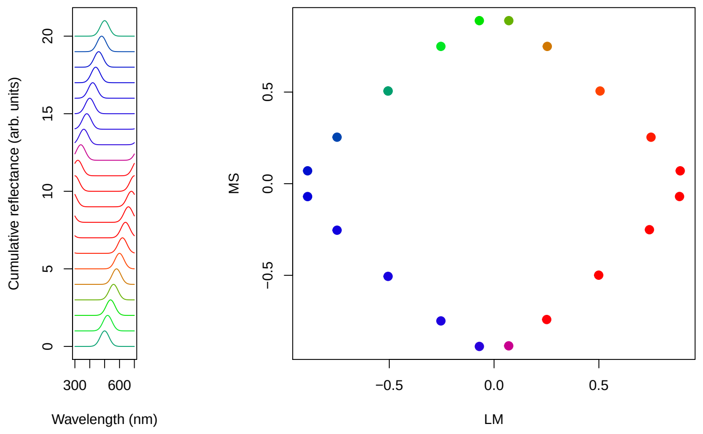

Figure 6: segment plot