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

Rafael Maia, Paul-Pierre Bitton, Chad Eliason

November 28, 2012

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. **O**rganize spectral data by inputting files, processing spectra (e.g., to remove noise, negative values, smooth curves, etc.)
2. **A**nalyze 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. **V**isualize the output

Below we will show the main functions in the package in an example workflow.

1 Data Description

The data used in this example is available from github by clicking here. 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*), Violet-green Swallow (*Tachycineta thalassina*), 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 default 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]
```

```
    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 235 spectra, from 300 to 700 nm
```

```
[1] 401 235
```

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

2.3 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 10 of the cardinal individuals measured:

```
> explorespec(specs[,1:40], specreps=3)
> # 39 spectra plus the first (wl) column
```

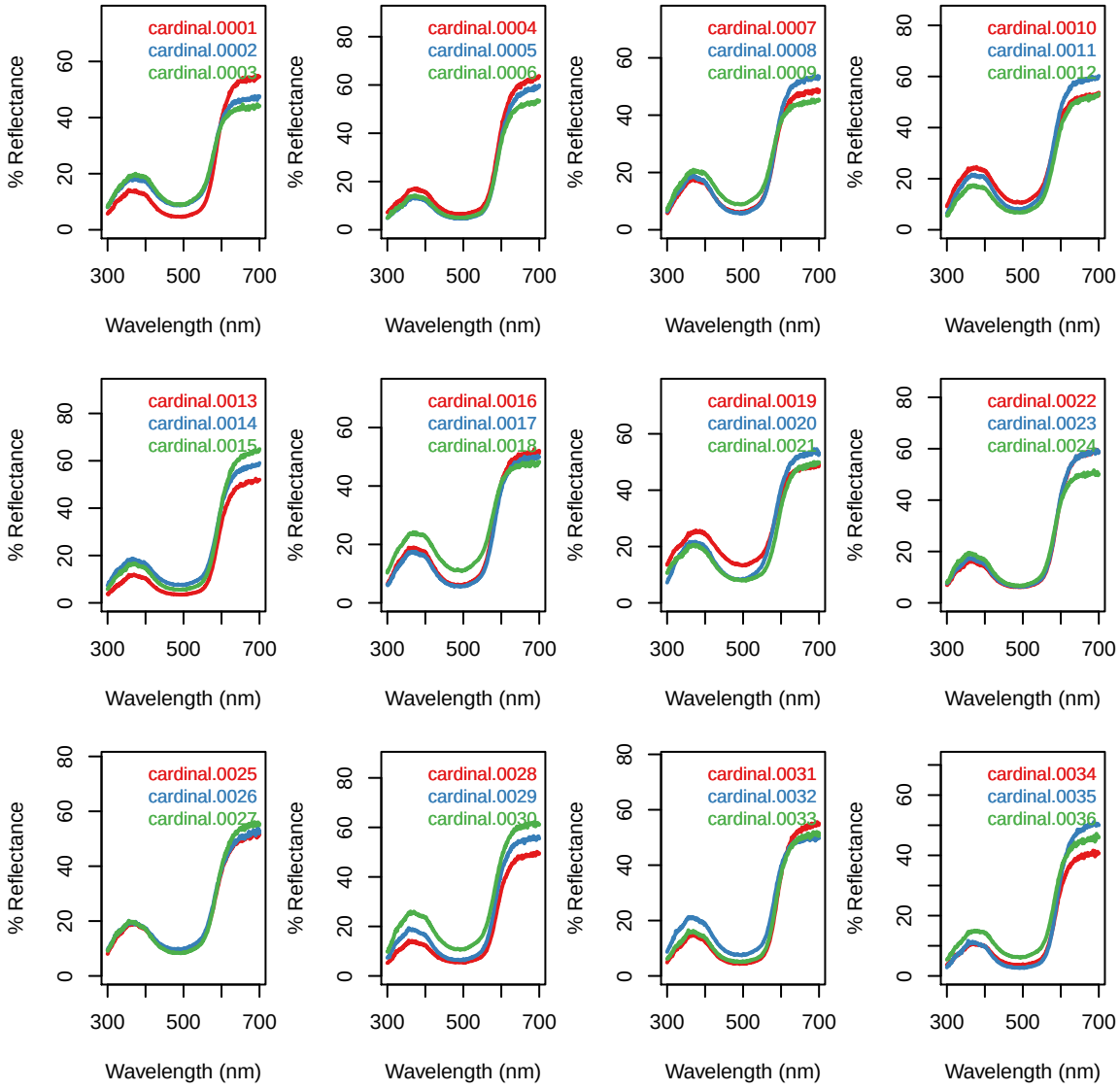

Figure 1: Result from `explorespec`, showing the three measurements for each individual in separate panels

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 79 spectra, one for each individual

[1] 401  79
```

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  swallow
      12        9        9       13       10        7
 tanager
      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  swallow
1 300 7.049397 7.334781 3.889693 7.629954 3.981747 4.035262
```

```
2 301 7.254161 7.354033 3.905322 7.746882 3.914297 4.036576
3 302 7.444275 7.452556 4.126619 7.886877 4.187073 4.098371
4 303 7.820686 8.085541 4.390685 8.491367 4.507410 4.132943
5 304 7.843394 7.714526 4.183637 8.658000 4.068800 4.026381
    tanager
1  9.021043
2  9.525854
3  9.405980
4 10.199843
5  9.684522
```

```
> explorespec(sppspec, 7)
```

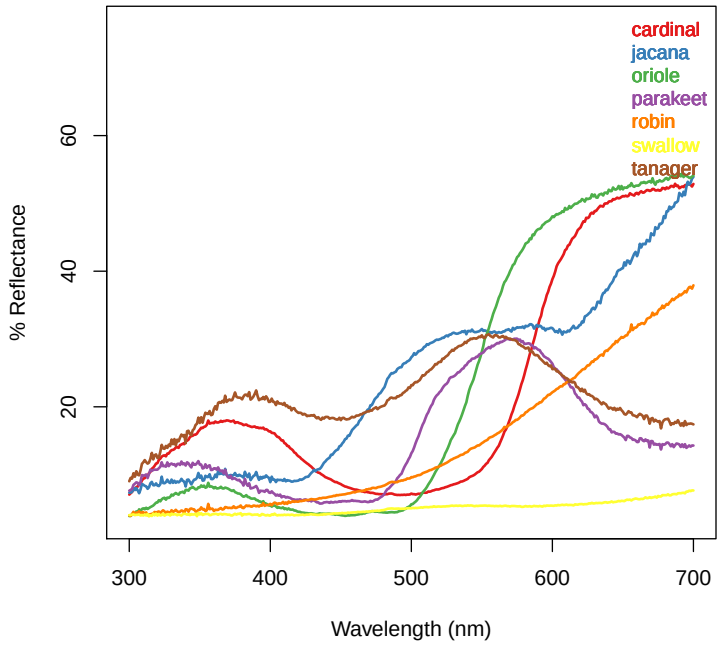

Figure 2: Result from `explorespec` for species means

3 Analyzing Spectral Data

3.1 Overview

add description here

3.2 Variables calculated

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	$R_{\mathrm{max}}/R_{\mathrm{min}}$	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`

Color variables described in Table 1.

blah blah blah[1] and also this.

4 Visualizing Spectral Data

```
> data(sicalis)
> par(mfrow=c(1,2))
```

[1] some footnote text here

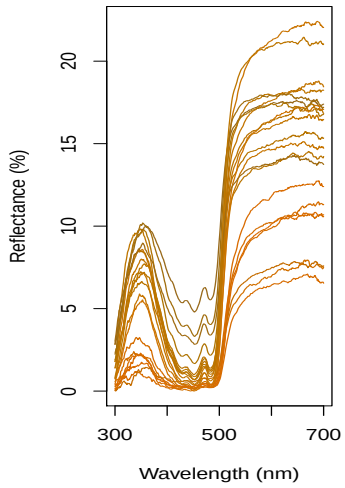

Figure 3: Overlay plot with colors calculated from human color matching functions

```
> 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

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

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

Examples

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

<< HEAD ======= >> 410ed83d33665cdb62e17e176f63f0d3ce6a4bae

More examples

Some more examples: