Cardinal plotting of ion images and mass spectra

Kyle D. Bemis

July 23, 2014

Contents

1 Introduction

One of the most important parts of working with mass spectrometry imaging datasets is visualization of the data by examining the ion images and mass spectra. *Cardinal* provides powerful functionality for plotting both ion images, mass spectra, as well as other representations of imaging data.

2 Formula interface

The plotting facilities of *Cardinal* are based on the powerful formula interface used by the `lattice` graphics package.

3 Plotting using Cardinal

For the following examples, we will use a simulated dataset. The image is a cardinal with red and black feathers, where the colors represent different regions of the image. The mass spectra will have two peaks to indicate the two regions. We use `generateImage` to generate the dataset from an integer matrix where 0 represents black regions of the image and 1 represents the red regions of the image.

```
> data <- matrix(c(NA, NA, 1, 1, NA, NA, NA, NA, NA, NA, 1, 1, NA, NA,
+  NA, NA, NA, NA, NA, 0, 1, 1, NA, NA, NA, NA, NA, 1, 0, 0, 1,
+  1, NA, NA, NA, NA, NA, 0, 1, 1, 1, 1, NA, NA, NA, NA, 0, 1, 1,
+  1, 1, 1, NA, NA, NA, NA, 1, 1, 1, 1, 1, 1, 1, NA, NA, NA, 1,
+  1, NA, NA, NA, NA, NA, NA, 1, 1, NA, NA, NA, NA, NA), nrow=9, ncol=9)
```

We can plot the ground truth image directly.

```
> image(data[,ncol(data):1], col=c("black", "red"))
```

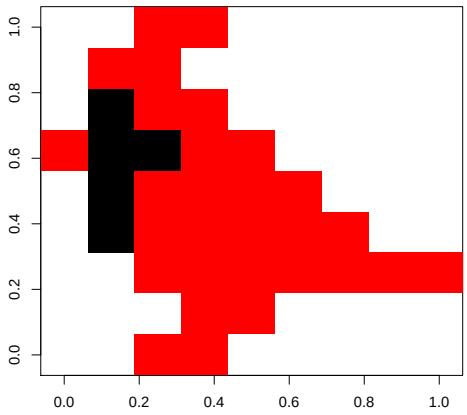

Figure 1: Ground truth image used to generate the simulated dataset.

Now we generate the data as if from a mass spectrometry imaging experiment with peaks at m/z 3000 (higher intensity in black pixels) and m/z 4000 (higher intensity in red pixels).

```
> set.seed(1)
> msset <- generateImage(data, range=c(1000,5000), centers=c(3000,4000), resolution=100,
+         step=3.3, as="MSImageSet")
```

We need to mark which pixels are black and which are red.

```
> pData(msset)$pg <- factor(data[is.finite(data)], labels=c("black", "red"))
```

Then we need to mark which features (which regions of the mass spectrum) belong to the peaks associated with "black" or "red" pixels; the rest of the spectrum is marked as background noise (bg).

```
> fData(msset)$fg <- factor(rep("bg", nrow(fData(msset))), levels=c("bg", "black", "red"))
> fData(msset)$fg[2950 < fData(msset)$mz & fData(msset)$mz < 3050] <- "black"
> fData(msset)$fg[3950 < fData(msset)$mz & fData(msset)$mz < 4050] <- "red"
```

Now we can experiment with different ways of plotting an imaging dataset.

3.1 Plotting mass spectra

The `plot` method is used to plot mass spectra. The `pixel` argument is used to specify the pixel to use to plot the mass spectrum. If no conditioning is desired, the formula does not need to be specified explicitly.

```
> plot(msset, pixel=1)
```

```
> plot(msset, ~ mz, pixel=1)
```

Specifying multiple pixels will apply a function, specified by `fun`, over those pixels. This can be used to create a plot of the mean spectrum (the default behavior). Below we obtain the mean spectrum of the red pixels, and the max spectrum of the black pixels.

```
> plot(msset, pixel=pData(msset)$pg=="red", fun=median, main="Median of red pixels")
```

```
> plot(msset, pixel=pData(msset)$pg=="black", fun=max, main="Max of black pixels")
```

Using the `lattice` graphics option allows for more complex plots to be made. Conditioning on variables in the `formula` argument allows direct comparison between regions of the image or mass spectrum. For example, by conditioning on the

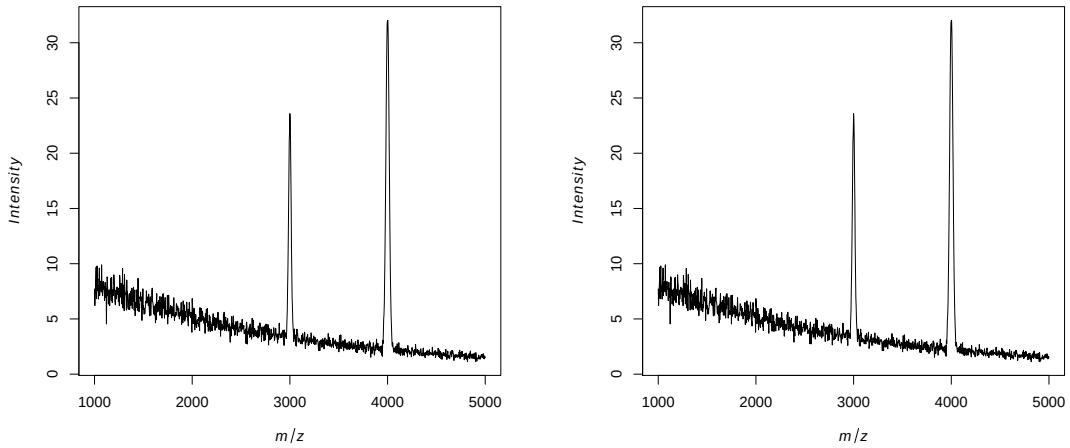

Figure 2: A simple mass spectrum plot. Both forms produce the same plot.

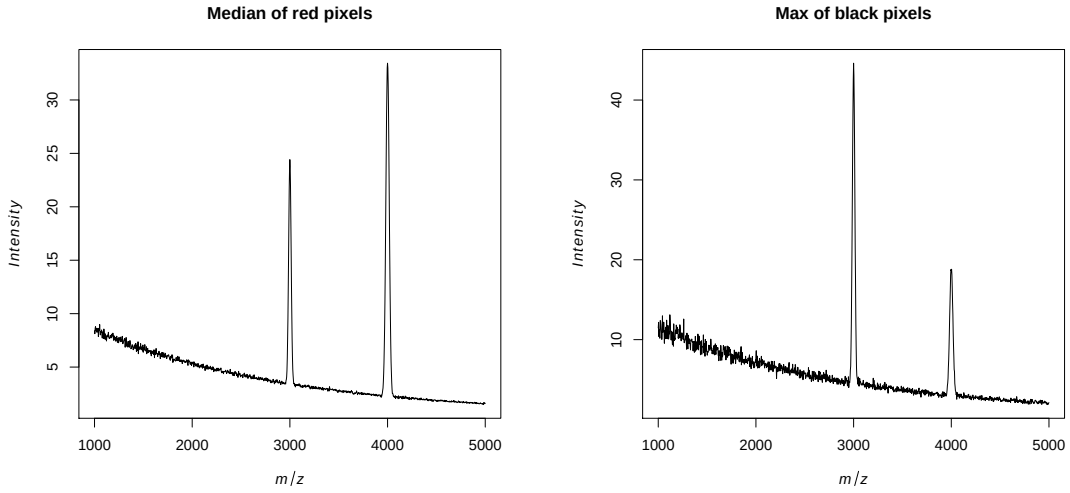

Figure 3: Applying a function over pixels to plot a median and max spectrum.

variable `pData(msset)$pg` which specifies the color of the pixels, we can obtain mean spectra for each type of pixel in a single step; notice that the `plot` method knows where to find the `pg` variable, because it is contained in `msset`. Likewise, we use the `fg` variable (which we used to mark notable m/z-values) with the argument `groups` to distinguish different regions of the mass spectrum with different colors.

```
> print(plot(msset, ~ mz | pg, pixel=1:ncol(msset), groups=fg, lattice=TRUE, col=c("blue", "black", "red"),
```

3.2 Plotting ion images

The `image` method is used to plot images. The `feature` argument is used to specify the feature to use to create the image. For a mass spectrometry imaging dataset, the features are the m/z-values corresponding to single-ion images. As before, if no conditioning is desired, the formula does not need to be specified explicitly.

```
> image(msset, feature=1, col.regions=gradient.colors(100, "red", "black"))
```

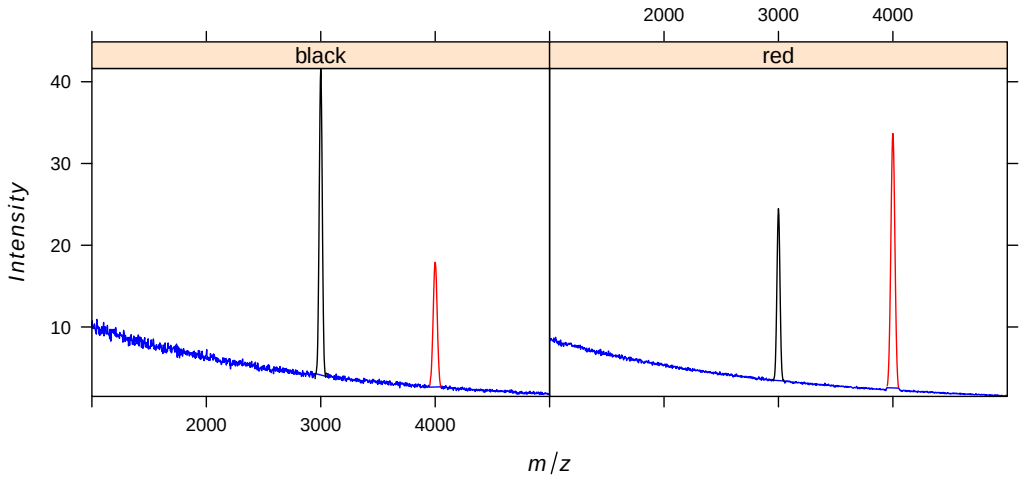

Figure 4: A plot conditioning on variables using `lattice` graphics.

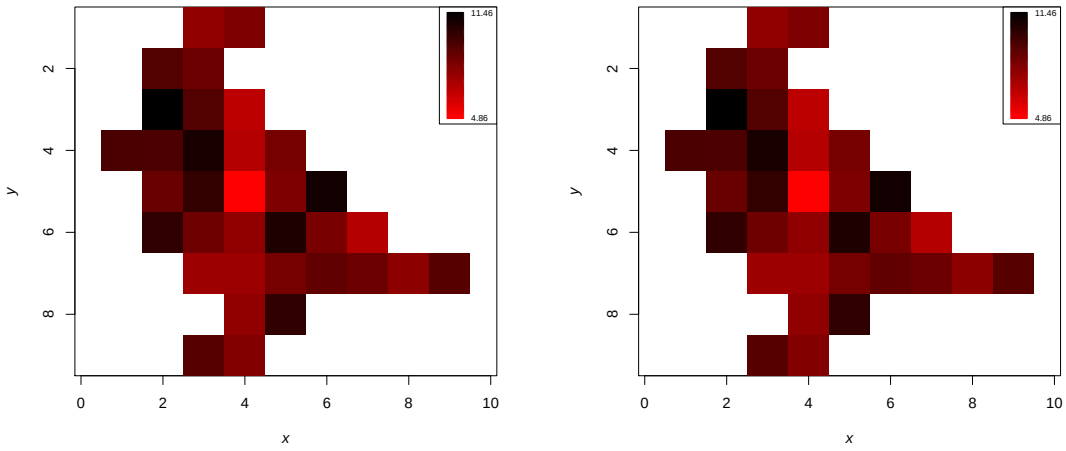

Figure 5: A simple single-ion image. Both forms produce the same plot.

```
> image(msset, ~ x * y, feature=1, col.regions=gradient.colors(100, "red", "black"))
```

Like with the `plot` method, the `image` method can apply functions over features (m/z-values) when multiple features are specified. By default, `mean` is used to average the images over the features. In the following example, we specify two plots, first using the features from the peak that has a higher intensity associated with black pixels, and then using the features from the peak that has a higher intensity associated with red pixels.

```
> image(msset, feature=fData(msset)$fg=="black", col.regions=alpha.colors(100, "black"))
```

```
> image(msset, feature=fData(msset)$fg=="red", col.regions=alpha.colors(100, "red"))
```

Using a `lattice`-style formula, we can condition on other variables with `image` too. Here we use all of the features, but condition on which part of the mass spectrum those features come from using the variable `fData(msset)$fg`. Again, since `image` knows to look in `msset`, we only need to specify the variable as `fg`.

```
> print(image(msset, ~ x * y | fg, feature=1:nrow(msset), col.regions=intensity.colors(100), lattice=TRUE),
```

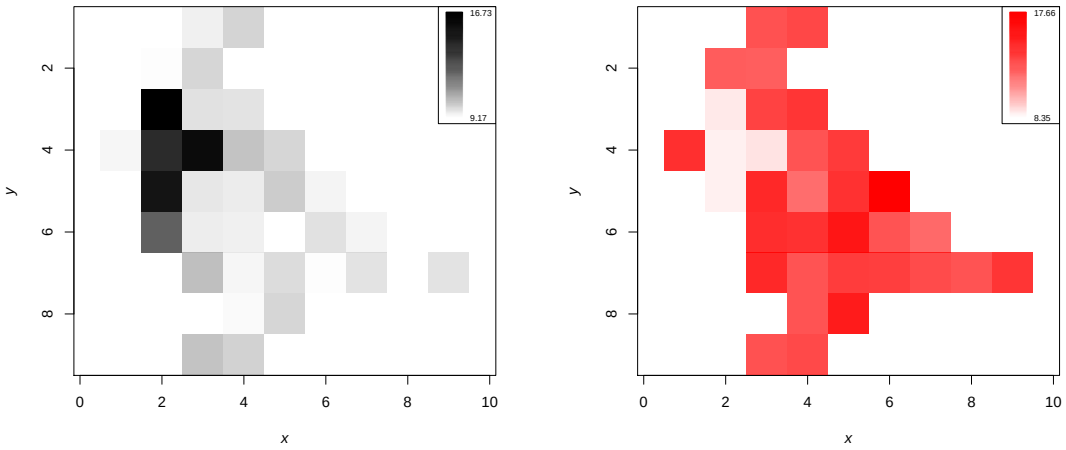

Figure 6: Averaging over different sets of mass features.

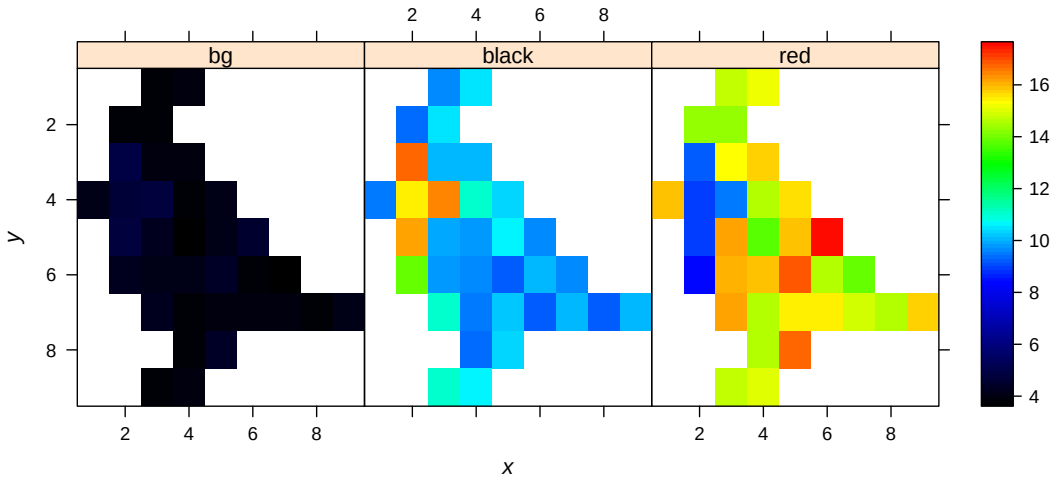

Figure 7: Images conditioning on variables using `lattice` graphics.

4 Session info

- R version 3.1.1 (2014-07-10), `x86_64-apple-darwin13.1.0`
- Locale: `en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, utils
- Other packages: Biobase 2.24.0, BiocGenerics 0.10.0, Cardinal 0.8.2
- Loaded via a namespace (and not attached): BiocStyle 1.2.0, fields 7.1, grid 3.1.1, irlba 1.0.3, lattice 0.20-29, maps 2.3-7, MASS 7.3-33, Matrix 1.1-4, signal 0.7-4, sp 1.0-15, spam 0.41-0, stats4 3.1.1, tools 3.1.1