Unsupervised analysis of MS images using Cardinal

Kylie A. Bemis and April Harry

April 26, 2017

Contents

1 Introduction

The goal of unsupervised analysis of mass spectrometry imaging experiments is to discover the regions of the data with distinct chemical profiles, and the masses that distinguish the chemical profiles of these regions.

Algorithmically, this means clustering the data. In imaging experiments, the resulting cluster configurations are called spatial segmentations, and the clusters are called segments.

In this vignette, an example clustering workflow in *Cardinal* is shown, along with plots of results.

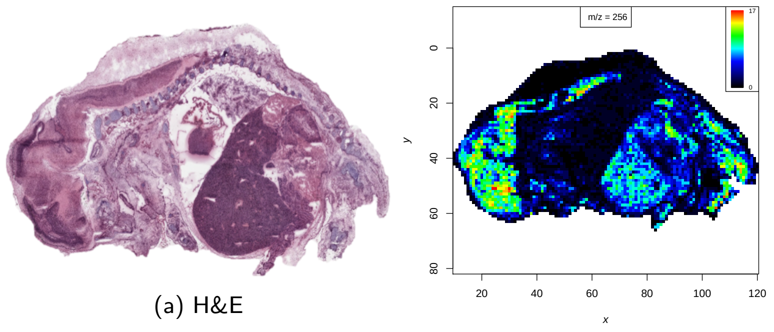
2 Analysis of a pig fetus wholy body cross section

This example uses the PIGII_206 dataset: a cross section of a pig fetus captured using a Thermo LTQ instrument using desorption electrospray ionization (DESI).

```
> library(CardinalWorkflows)
> data(pig206, pig206_analyses)
```

The PIGII_206 dataset is of interest for segmentation because we expect the underlying morphology of the tissue to be reflected in its chemical profile. It would be interesting, therefore, if we were able to identify abundant analytes certain physical systems such as the heart, spine, liver, or brain.

```
> image(pig206, mz = 256)
```



(a) H&E

(b) m/z 256

Figure 1: Optical image and a cropped ion image for the pig fetus dataset.

```
> summary(pig206)
```

```
Class: MSImageSet
Features: m/z = 150.08 ... m/z = 1000 (10200 total)
Pixels: x = 72, y = 1 ... x = 83, y = 66 (4959 total)
x: 10 ... 120
y: 1 ... 66
Size in memory: 195.4 Mb
```

In the optical image shown in Figure 1a, the brain (left), heart (center), and liver (large dark region below heart) can be seen. The MS dataset has been cropped to remove the background slide pixels, leaving only the tissue section itself for analysis as seen in Figure 1b. The image contains 4959 pixels with 10200 spectral features measured at each location (m/z range from 150 to 1000).

2.1 Pre-processing

For statistical analysis, it is necessary to perform some kind of dimension reduction on the dataset as part of the pre-processing, so that computation times are reasonable. We will use peak-picking, although dimension reduction via resampling or binning is also a possible.

Note that we can perform peak-picking here for our unsupervised analysis because we do not have to worry about cross-validation as we would when performing classification (see classification workflow).

2.1.1 Normalization

In order to ensure that the spectra are comparable pixel-to-pixel, normalization is often done as a pre-processing step. A popular choice for normalization in mass spectrometry image analysis is total ion current (TIC) standardization.

```
> pig206.norm <- normalize(pig206, method = "tic")
```

2.1.2 Peak picking and alignment

For computational efficency, it is necessary to do peak picking prior to analysing the data. As in [1], we peak pick on every 10^{th} mass spectrum, retaining only those peaks that occur in at least 1% of the considered spectra. The selection of peaks in *Cardinal* is done using a comparison of local maxima against noise.

First, we perform peak-picking on ever 10^{th} mass spectrum using the `peakPick` method, looking for peaks with a signal-to-noise ratio (SNR) of at least 6.

```
> pig206.peaklist <- peakPick(pig206.norm, pixel = seq(1, ncol(pig206), by = 10),
+     method = "simple", SNR = 6)
```

The peaks must be aligned using `peakAlign`. Below, the mean spectrum of the raw data is used as the reference, so the peaks will be aligned to the local maxima in the mean spectrum.

```
> pig206.peaklist <- peakAlign(pig206.peaklist, ref = pig206.norm, method = "diff",
+     units = "ppm", diff.max = 200)
```

Now we use the `peakFilter` method to drop peaks that occur less frequently than once every 100 spectra.

```
> pig206.peaklist <- peakFilter(pig206.peaklist, method = "freq", freq.min = ncol(pig206.peaklist)/100)
```

Finally, `reduceDimension` method is used to sweep back through the full normalized dataset retrieve the identified peaks from all of the pixels.

```
> pig206.peaks <- reduceDimension(pig206.norm, ref = pig206.peaklist, type = "height")
```

```
> summary(pig206.peaks)
```

```
Class: MSImageSet
Features: m/z = 151.33 ... m/z = 889.67 (143 total)
Pixels: x = 72, y = 1 ... x = 83, y = 66 (4959 total)
x: 10 ... 120
y: 1 ... 66
Size in memory: 6.6 Mb
```

An alternative pre-processing workflow would be to perform peak-picking on all mass spectra and use these peaks directly (after alignment) rather than use `reduceDimension`. However, this would result in 0 intensities for mass spectra where certain peaks were not found, so it places a greater burden on the accuracy of the peak detection algorithm.

2.2 Visualizing the dataset

In this section, we will walk through several methods for visualizing the dataset before statistical analysis.

2.2.1 Visualization of molecular ion images

Plotting ion images is the natural first step in exploring a mass spectrometry imaging dataset.

In Figure 2 we show ion images for three selected peaks. These images have been contrast enhanced using histogram equalization [2]. Further, Gaussian smoothing is used to improve visualization of the images.

```
> image(pig206, mz = 187, contrast.enhance = "histogram", smooth.image = "gaussian")
```

```
> image(pig206, mz = 284, contrast.enhance = "histogram", smooth.image = "gaussian")
```

```
> image(pig206, mz = 537, contrast.enhance = "histogram", smooth.image = "gaussian")
```

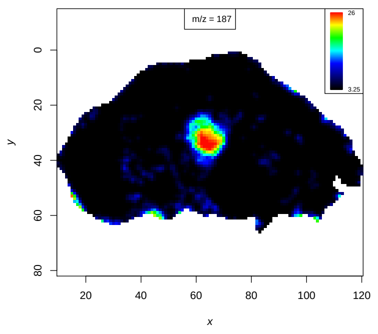
These ion images show morphological features such as the heart (Figure 2a), brain (Figure 2b), and liver (Figure 2c) that an unsupervised analysis should identify in a spatial segmentation.

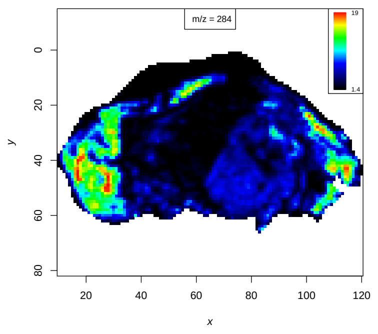
2.2.2 Exploratory analysis using PCA

Principal component analysis (PCA) is another popular method for exploring a dataset. PCA is also available in *Cardinal*. We now explore the peak-picked dataset using PCA with 5 components, fit below using the `PCA` method.

```
> pig206.pca <- PCA(pig206.peaks, ncomp = 5)
```

```
> summary(pig206.pca)
```


(a) m/z 187 (heart)


(b) m/z 284 (brain)

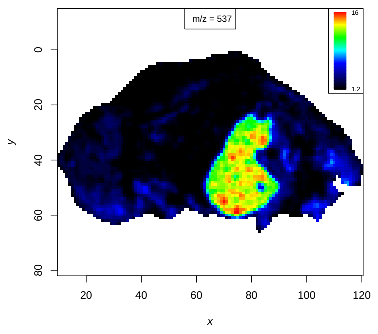

(c) m/z 537 (liver)

Figure 2: Ion images showing histogram equalization and Gaussian smoothing.

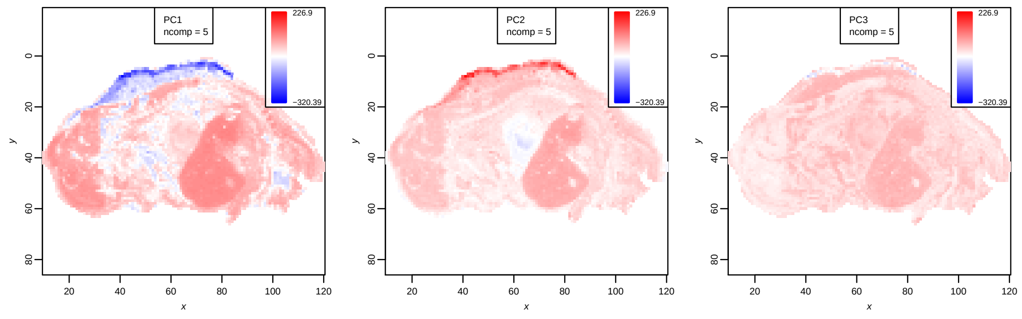
```
                        PC1      PC2      PC3      PC4      PC5
Standard deviation 55.50312 30.40846 18.97628 18.10846 14.54119
```

The summary statistics show that the first 5 principal components cumulatively explain approximately 85% of the variation in the data. However, plotting the PC scores as images (as shown below in Figure 3a) shows that PCA is only somewhat helpful in visualizing the data, as the variation explained by PCA is dominated by the tissue above the spine.

```
> image(pig206.pca, column = c("PC1", "PC2", "PC3"), superpose = FALSE, col.regions = risk.colors(100),
+     layout = c(3, 1))
```

In Figure 3b, we plot the loadings for the principal components as well.

```
> plot(pig206.pca, column = c("PC1", "PC2", "PC3"), superpose = FALSE, layout = c(3,
+     1))
```


(a) Images of the PCA scores

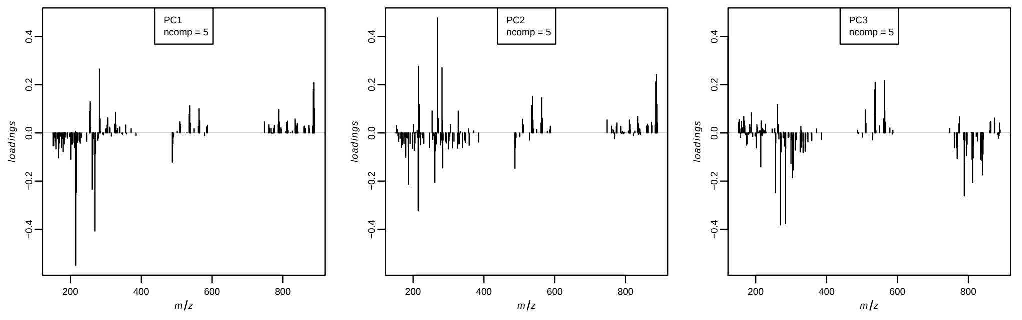

(b) Plot of the PCA loadings

Figure 3: Principal components analysis of PIGII_206.

Although PCA is widely used and can be visually useful when first exploring a dataset, it is rarely informative enough to constitute a complete analysis on its own, and additional statistical analyses are usually required to obtain more meaningful results.

2.3 Spatial segmentation using spatially-aware k-means clustering

This section demonstrates the spatially-aware clustering presented by Alexandrov and Kobarg [3], implemented in *Cardinal* in the spatialKMeans method.

This method uses a spatial distance function to perform dimension reduction on the mass spectra prior to clustering using ordinary k-means. The algorithm differentiates between "spatially-aware" (SA) and "spatially-aware structurally-adaptive" (SASA) clustering, which use different weights in the spatial distance function.

The "SA" variant uses simple Gaussian weights, and the "SASA" variant uses adaptive weights that often better preserves edges between segments. In *Cardinal*, these methods are differentiated by a method argument which can be set to either "gaussian" or "adaptive".

The parameters to be explicitly provided in the spatialKMeans method are:

- r: The neighborhood smoothing radius
- k: The number of segments (clusters)

The method is vectorized over these parameters, so multiple models can be fit simultaneously by giving them multiple values.

Below, we perform SA clustering with the method="gaussian" weights.

```
> set.seed(1)
> pig206.skmg <- spatialKMeans(pig206.peaks, r = c(1, 2), k = c(5, 10), method = "gaussian")
```

```
> summary(pig206.skmg)
```

```
  r  k   method  time Within-Cluster SS Between-Cluster SS Total SS
1 1  5 gaussian 1.425          10250814           18077178 28327992
2 1 10 gaussian 3.354           7460608           20867384 28327992
3 2  5 gaussian 1.422          11379227           16948765 28327992
4 2 10 gaussian 3.023           8831925           19496067 28327992
```

Additionally, we perform SASA clustering with the method="adaptive" weights.

```
> set.seed(1)
> pig206.skma <- spatialKMeans(pig206.peaks, r = c(1, 2), k = c(5, 10), method = "adaptive")
```

```
> summary(pig206.skma)
```

```
  r  k   method  time Within-Cluster SS Between-Cluster SS Total SS
1 1  5 adaptive 2.675          12066467           16261525 28327992
2 1 10 adaptive 2.950           9713476           18614516 28327992
3 2  5 adaptive 1.825          11492427           16835565 28327992
4 2 10 adaptive 2.739           9110589           19217403 28327992
```

Both of the resulting objects have four sets of model parameters, in the parameter space of $r = 1, 2$ and $k = 5, 10$.

2.3.1 Plotting the spatial segmentation

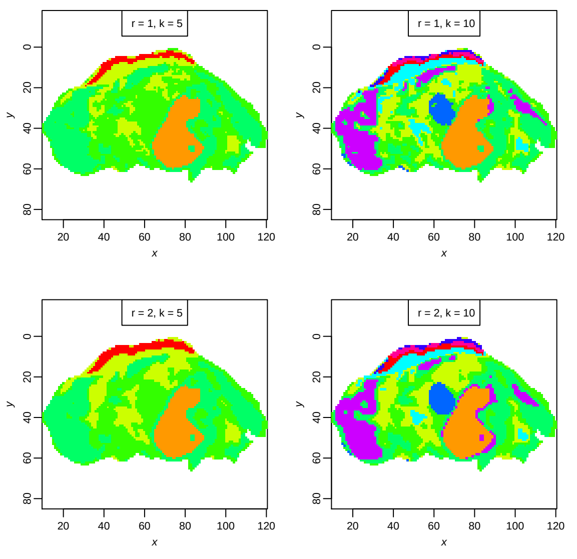
Cardinal has the capacity to simultaneously plot segmentations for the whole set of parameters. Figure 4a shows the SA results (Gaussian weights), and Figure 4b shows the SASA results (adaptive weights).

```
> image(pig206.skmg, key = FALSE, layout = c(2, 2))
```

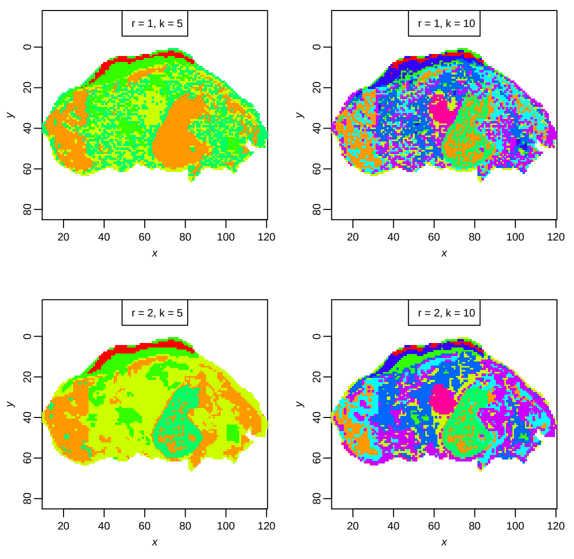
```
> image(pig206.skma, key = FALSE, layout = c(2, 2))
```

Figure 4a shows the segmentations for SA clustering and Figure 4b shows the segmentations for SASA clustering.

The segments are indicated by colors, and the additional colors used in the plots with $k = 10$ reflect the additional number of segments in those segmentations.



(a) Segmentation for SA k-means clustering



(b) Segmentation for SASA k-means clustering

Figure 4: The spatial segmentations for spatially-aware (SA) and spatially-aware structurally-adaptive (SASA) k-means clustering with varying smoothing radii ($r = 1, 2$) and number of segments ($k = 5, 10$). Pixels with the same color indicate membership in the same segment.

The noisiness in the SASA clusterings for $r = 1$ seem to suggest that it is too small a radius for use with SASA clustering for this particular dataset. In both sets of clusterings – particularly the SA clusterings with $k = 10$ – we can often see distinct segments that correspond to organs such as the brain, heart, and liver.

2.3.2 Plotting the mean spectra of the segments

Likewise, we can also plot the mean spectra for the spatial segmentations. For each parameter set, the mean spectra for all segments are overlaid in Figure 5a for SA clustering (Gaussian weights) and Figure 5b for SASA clustering (adaptive weights).

```
> plot(pig206.skmg, key = FALSE, layout = c(2, 2), type = c("p", "h"), pch = 20)
```

```
> plot(pig206.skma, key = FALSE, layout = c(2, 2), type = c("p", "h"), pch = 20)
```

Figure 5a shows the mean spectra for SA clustering and Figure 5b shows mean spectra for SASA clustering.

Like PCA, spatially-aware k-means clustering is useful for quickly visualizing an MS imaging dataset, but additional analysis is required for statistically-meaningful results.

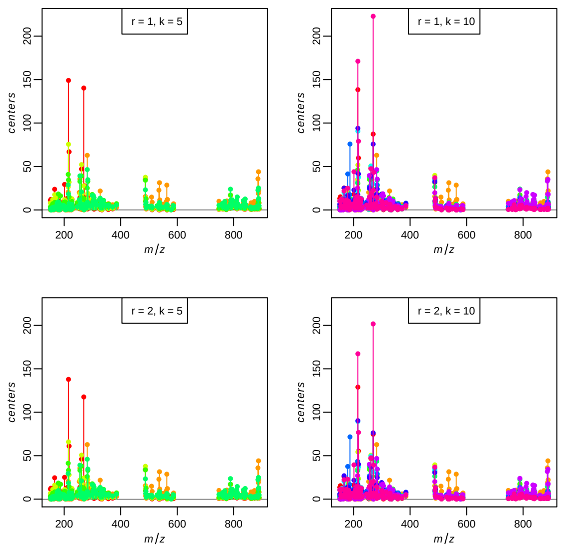
2.4 Spatial segmentation using spatial shrunken centroids clustering

This section demonstrates the spatial shrunken centroids clustering method for statistical analysis we introduce in *Cardinal* in the `spatialShrunkenCentroids` method.

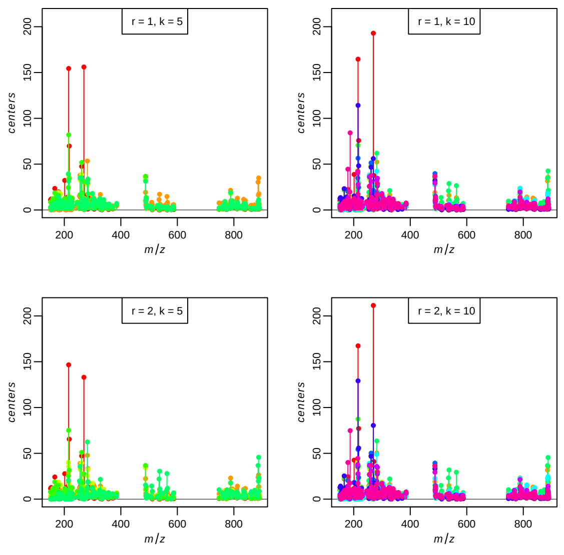
The Gaussian and adaptive weights are retained from the spatially-aware k-means clustering [3]. In addition, we introduce statistical regularization as in nearest shrunken centroids to enforce feature sparsity [4]. This means doing feature selection, so that the analysis automatically identifies informative m/z values.

The parameters to be explicitly provided in the `spatialShrunkenCentroids` method are:

- r: The neighborhood smoothing radius
- k: The initial number of segments (clusters)
- s: The shrinkage parameter



(a) Mean spectra for SA k-means clustering



(b) Mean spectra for SASA k-means clustering

Figure 5: The mean spectra for spatially-aware (SA) and spatially-aware structurally-adaptive (SASA) k-means clustering with varying smoothing radii ($r = 1, 2$) and number of segments ($k = 5, 10$). Colors indicate the segment from which each mean spectrum is calculated (corresponding segments can be found in Figure 4a and Figure 4b).

The r parameter is the same as in `spatialKMeans`. The k parameter now specifies only the *initial* number of segments. The `spatialShrunkenCentroids` method allows the number of segments to decrease according to the data. This allows automatic selection of the number of clusters.

The s parameter is the shrinkage parameter that enforces sparsity. As s increases, fewer mass features (m/z values) will be used in the spatial segmentation, and only the informative mass features will be retained.

Typically, the number of segments will also decrease as s increases. This makes sense, because as fewer mass features are used, the fewer segments they will be able to explain.

For a detailed explanation of the shrinkage parameter s, see [4] and [5].

Now we perform spatial shrunken centroids clustering with the `method="gaussian"` weights.

```
> set.seed(1)
> pig206.sscg <- spatialShrunkenCentroids(pig206.peaks, r = c(1, 2), k = c(15,
+     20), s = c(0, 3, 6, 9), method = "gaussian")
```

```
> summary(pig206.sscg)
```

```
   r  k s   method   time Predicted # of Classes Mean # of Features per Class
1  1 15 0 gaussian 14.557                     15                          143
2  1 15 3 gaussian 25.602                     10                           91
3  1 15 6 gaussian 32.892                      7                           77
4  1 15 9 gaussian 19.852                      6                           63
5  1 20 0 gaussian 20.906                     19                          143
6  1 20 3 gaussian 28.911                     11                           92
7  1 20 6 gaussian 27.199                      8                           74
8  1 20 9 gaussian 36.020                      6                           62
9  2 15 0 gaussian 55.460                     13                          143
10 2 15 3 gaussian 42.302                     10                           90
11 2 15 6 gaussian 40.441                      6                           82
12 2 15 9 gaussian 42.164                      6                           62
13 2 20 0 gaussian 48.375                     18                          143
14 2 20 3 gaussian 77.894                      9                           90
15 2 20 6 gaussian 50.536                      6                           82
16 2 20 9 gaussian 43.045                      6                           60
```

And we perform spatial shrunken centroids clustering with the `method="adaptive"` weights.

```
> set.seed(1)
> pig206.ssca <- spatialShrunkenCentroids(pig206.peaks, r = c(1, 2), k = c(15,
+     20), s = c(0, 3, 6, 9), method = "adaptive")
```

```
> summary(pig206.ssca)
```

```
   r  k s   method   time Predicted # of Classes Mean # of Features per Class
1  1 15 0 adaptive 16.397                     15                          143
2  1 15 3 adaptive 32.927                      9                          100
3  1 15 6 adaptive 39.063                      6                           85
4  1 15 9 adaptive 23.927                      6                           62
5  1 20 0 adaptive 16.943                     20                          143
6  1 20 3 adaptive 28.908                     11                           83
7  1 20 6 adaptive 25.515                      8                           69
8  1 20 9 adaptive 28.771                      6                           62
9  2 15 0 adaptive 23.259                     15                          143
10 2 15 3 adaptive 41.330                      9                          100
11 2 15 6 adaptive 34.436                      8                           69
12 2 15 9 adaptive 30.122                      7                           55
13 2 20 0 adaptive 62.661                     18                          143
14 2 20 3 adaptive 46.154                     10                           91
15 2 20 6 adaptive 51.921                      7                           78
16 2 20 9 adaptive 51.214                      6                           66
```

Both of the resulting objects have sixteen sets of model parameters, in the parameter space of $r = 1, 2$, $k = 15, 20$, and $s = 0, 3, 6, 9$.

As seen in the summaries above, many of the segmentations result in fewer numbers of segments than at initialization, and the number of segments is generally lower for higher sparsity. We will show how this can be used to determine the number of segments in Section 2.4.4.

2.4.1 Plotting the spatial segmentations

We will plot four of the spatial segmentations for each of the Gaussian weights and adaptive weights. This is specified by the `model` argument, where we can list the parameters for the models we would like to plot.

```
> image(pig206.sscg, model = list(r = 2, s = c(0, 6)), key = FALSE, layout = c(2,
+     2))
```

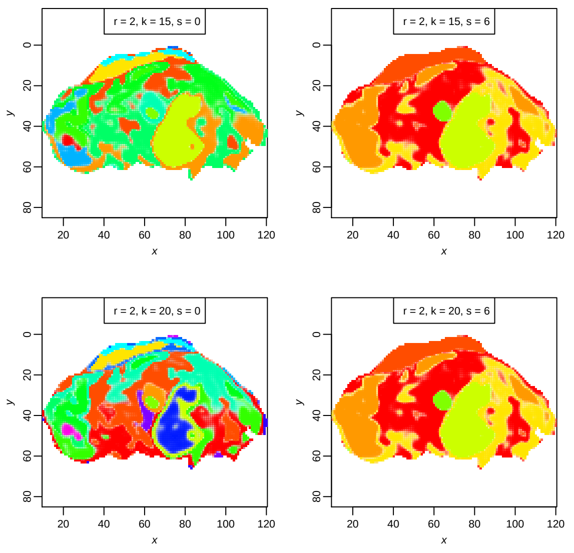
```
> image(pig206.ssca, model = list(r = 2, s = c(0, 6)), key = FALSE, layout = c(2,
+     2))
```

Figure 6a shows the segmentations for spatial shrunken centroids with Gaussian weights and Figure 6b shows the segmentations for adaptive weights.

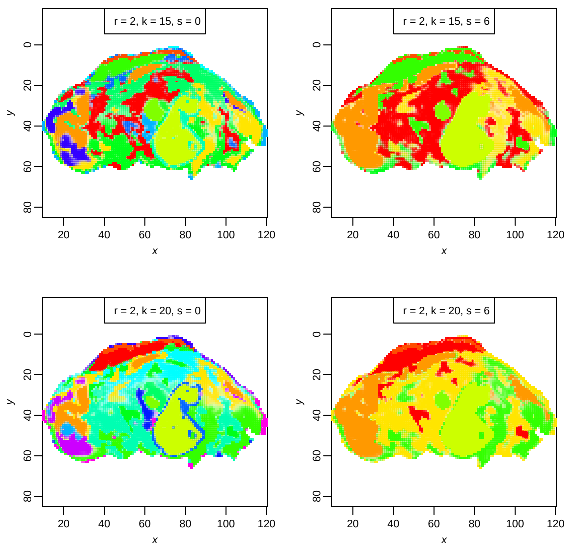
Colors indicate segment membership, but rather than hard assignment to segments, `spatialShrunkenCentroids` produces probabilities of segment membership. We visualize these probabilistic segment assignments using opacity, where higher opacity of a color indicates higher probability of that pixel belonging to that segment.

Visualizing probabilistic segment membership is useful in quickly assessing the quality of a spatial segmentation. Low quality segmentations will yield high uncertainty in the segment assignments, resulting in low probabilities and low opacities. This means that the segmentation will appear "fuzzier" and segments will have blurry or indistinct edges. Better segmentations will have higher opacity and sharper edges.

Figure 6a and Figure 6b show that an increase the shrinkage parameter s results in fewer segments. We started the clustering interations with both $k = 15$ and $k = 20$ initial segments, but at $s = 6$, the segmentations look very similar. For the Gaussian weights at $s = 6$, the segmentation has 6 segments for both $k = 15$ and $k = 20$. For adaptive weights at $s = 6$, both segmentations have 7 segments.



(a) Segmentation for SSC with Gaussian weights



(b) Segmentation for SSC with adaptive weights

Figure 6: The spatial segmentations for spatial shrunken centroids clustering with Gaussian and adaptive weights, varying smoothing radii ($r = 1, 2$), initial number of segments ($k = 15, 20$), and shrinkage parameter ($s = 0, 3, 6, 9$). Colors indicate segment membership. Opacity of the colors indicate probability of the belonging to that segment.

2.4.2 Plotting the (shrunken) mean spectra of the segments

As part of the feature selection process, the mean spectrum for each segment is shrunken toward the global mean spectrum. The reason for this will be discussed in Section 2.4.3. We can plot these shrunken mean spectra and interpret them by analogy to ordinary mean spectra.

```
> plot(pig206.sscg, model = list(r = 2, s = c(0, 6)), key = FALSE, layout = c(2,
+     2), type = c("p", "h"), pch = 20)
```

```
> plot(pig206.ssca, model = list(r = 2, s = c(0, 6)), key = FALSE, layout = c(2,
+     2), type = c("p", "h"), pch = 20)
```

Figure 7a shows the shrunken mean spectra for spatial shrunken centroids with Gaussian weights and Figure 7b shows the shrunken mean spectra for adaptive weights.

2.4.3 Plotting and interpreting the t-statistics of the m/z values

An important goal of our approach to spatial segmentation is that we not only want a meaningful segmentation, but we also want to be able to identify and rank the important mass features that inform that segmentation. The `spatialShrunkenCentroids` method produces t-statistics for this purpose.

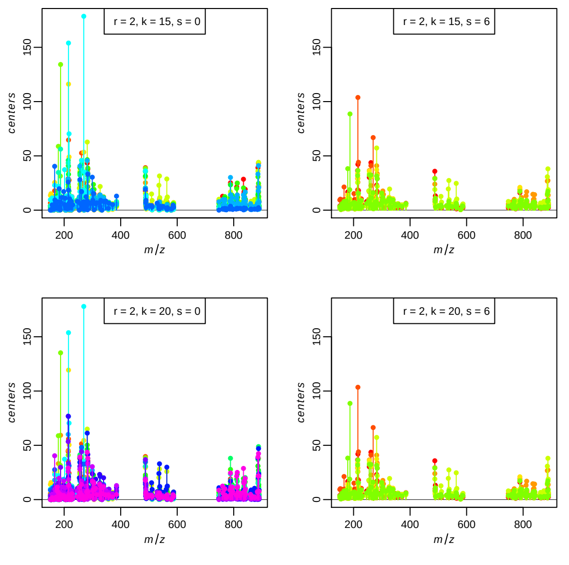
For each mass feature (m/z value), t-statistics are calculated for each segment as in [4] and [5], by comparison to the global mean spectrum.

Positive t-statistics correspond to systematic enrichment in that segment. Negative t-statistics correspond to systematic absence from that segment. The shrinkage parameter s is used to shrink t-statistics toward 0, and when a t-statistic is set to 0, that mass feature is no longer used to determe segment membership.

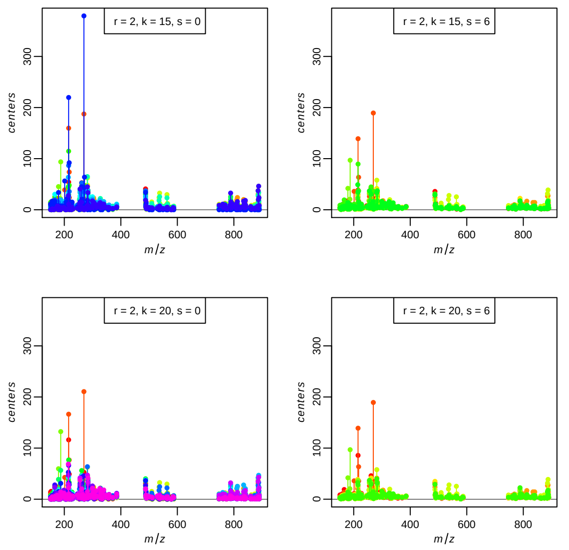
```
> plot(pig206.sscg, mode = "tstatistics", model = list(r = 2, s = c(0, 6)), key = FALSE,
+     layout = c(2, 2), type = c("p", "h"), pch = 20)
```

```
> plot(pig206.ssca, mode = "tstatistics", model = list(r = 2, s = c(0, 6)), key = FALSE,
+     layout = c(2, 2), type = c("p", "h"), pch = 20)
```

Note in Figure 8a and Figure 8b that the segments that disappear as s increases from $s = 0$ to $s = 6$ generally have low t-statistics. These low t-statistics are quickly driven to 0, and their segments are removed from the model and joined with other segments.

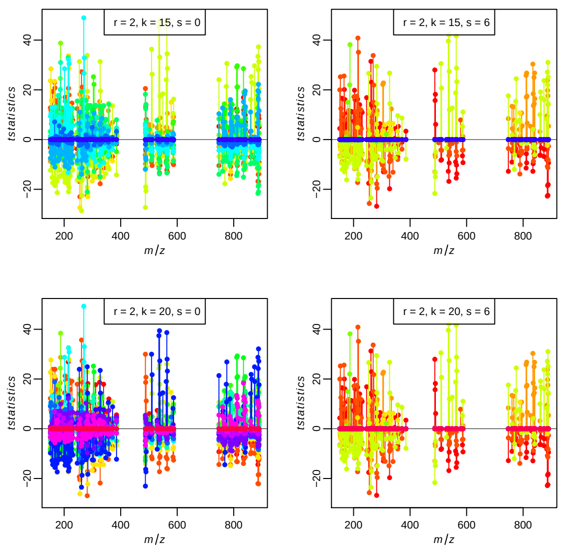


(a) Shrunken mean spectra for SSC with Gaussian weights

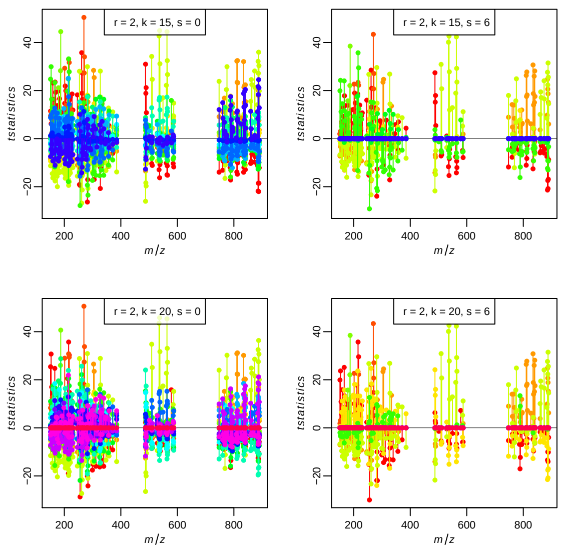


(b) Shrunken mean spectra for SSC with adaptive weights

Figure 7: The shrunken mean spectra for spatial shrunken centroids for Gaussian and adaptive weights with varying smoothing radii ($r = 1, 2$), initial number of segments ($k = 5, 10$), and shrinkage parameter ($s = 0, 3, 6, 9$). Colors indicate the segment from which each shrunken mean spectrum is calculated (corresponding segments can be found in Figure 6a and Figure 6b).



(a) Shrunken t-statistics for SSC with Gaussian weights



(b) Shrunken t-statistics for SSC with adaptive weights

Figure 8: The shrunken t-statistics for spatial shrunken centroids for Gaussian and adaptive weights with varying smoothing radii ($r = 1, 2$), initial number of segments ($k = 5, 10$), and shrinkage parameter ($s = 0, 3, 6, 9$). Colors indicate the segment from which the t-statistics are calculated (corresponding segments can be found in Figure 6a and Figure 6b).

The top m/z values for a segmentation can be queried using the topLabels method.

```
> topLabels(pig206.sscg, n = 20)

        mz r  k s classes   centers tstatistics p.values adj.p.values
1 269.3333 2 20 0      11 177.93947    49.26370        0            0
2 269.3333 2 15 0      11 178.60039    48.95050        0            0
3 269.3333 1 20 0      11 167.71484    48.59475        0            0
4 269.3333 1 15 0      11 168.12032    48.50941        0            0
```

```
5  537.2500 2 15 0       5  31.48324    47.73488        0             0
6  537.2500 1 15 0       5  31.72642    47.64653        0             0
7  269.3333 1 15 3       4 165.49849    47.43609        0             0
8  269.3333 1 20 3      11 165.49371    47.41058        0             0
9  563.2500 2 15 0       5  28.73309    47.40687        0             0
10 563.2500 1 15 0       5  28.93201    47.25001        0             0
11 537.2500 1 20 0      14  31.81774    47.21773        0             0
12 563.2500 1 20 0      14  29.01530    46.81710        0             0
13 537.2500 1 20 3       5  29.67071    45.47840        0             0
14 537.2500 1 15 3       5  29.67604    45.46918        0             0
15 269.3333 2 15 3       4 116.89795    45.39698        0             0
16 535.2500 2 15 0       5  22.89653    45.32567        0             0
17 537.2500 2 20 3       5  29.25750    45.29100        0             0
18 535.2500 1 15 0       5  23.06823    45.20485        0             0
19 537.2500 2 15 3       5  29.60737    45.15213        0             0
20 563.2500 1 20 3       5  26.98541    45.12665        0             0
```

This list can be filtered by the segment, model parameters, etc.

```
> topLabels(pig206.sscg, model = list(r = 2, s = 6, k = 20), filter = list(classes = 5))
```

```
        mz r  k s classes  centers tstatistics p.values adj.p.values
1 537.2500 2 20 6       5 27.35050    42.19177        0            0
2 563.2500 2 20 6       5 24.70218    41.70469        0            0
3 535.2500 2 20 6       5 19.56177    39.62103        0            0
4 887.6667 2 20 6       5 38.07015    31.04249        0            0
5 509.2500 2 20 6       5 12.68780    30.56082        0            0
6 281.6667 2 20 6       5 57.27400    29.44007        0            0
```

See `?topLabels` for details. This will be explored further in Section 2.4.5.

2.4.4 Identifying the number of segments

A unique property of our spatial shrunken centroids clustering is that it facilitates a natural way to identify an appropriate number of segments for a segmentation. We do this by plotting the number of predicted segments against the shrinkage parameter s as shown below in Figure 9a. For this dataset, adaptive weights appear to offer no advantage over Gaussian weights, so we focus on the segmentations with Gaussian weights.

```
> plot(summary(pig206.sscg), main = "Number of segments")
```

In Figure 9a, we look for the shrinkage parameter s for which the predicted number of segments match up between different initialized numbers of segments k. For $r = 2$, this occurs at $s = 6$, and we can also see from Figure 6a earlier that the final segmentations for both $k = 15$ and $k = 20$ at $s = 6$ are nearly identical. This suggests that either of these models are appropriate segmentations.

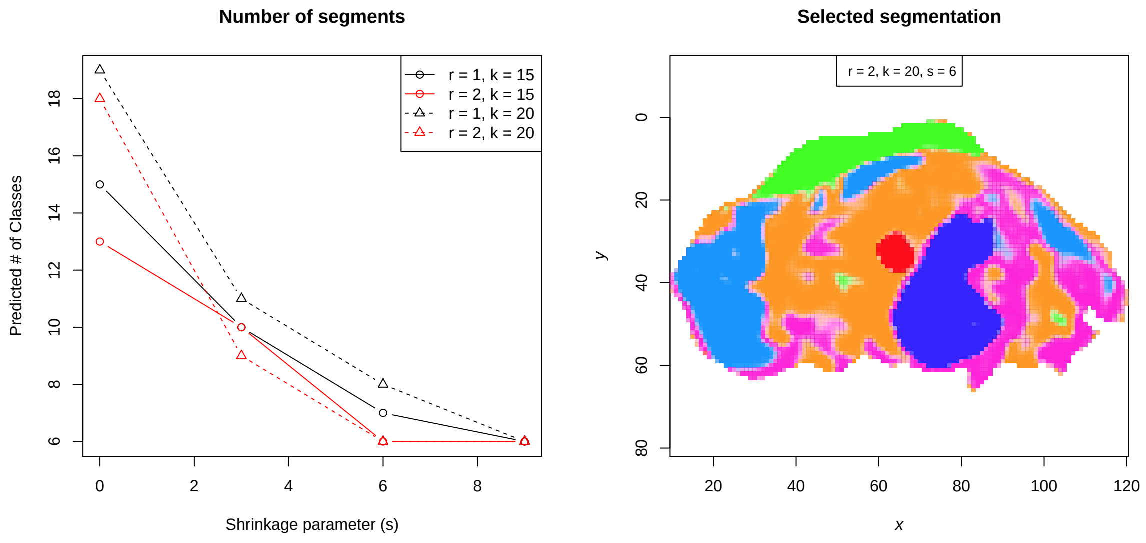
Therefore, we choose the segmentation with Gaussian weights with $r = 2, k = 20, r = 6$ for further exploration. This segmentation is plotted with custom colors in Figure 9b.

```
> mycol <- c(internal1 = "#FD9827", back = "#42FD24", internal2 = "#1995FC", brain = "#FC23D9",
+     liver = "#3524FB", heart = "#FC0D1B", bg = "#CDFD34")
> image(pig206.sscg, model = list(r = 2, k = 20, s = 6), key = FALSE, col = mycol,
+     main = "Selected segmentation")
```

Typically, we recommend choosing the segmentation with the most retained features (least sparsity) after which the predicted number of segmentations become approximately equal between different initializations of k.

2.4.5 Interpretting the spatial segmentation

To further explore the spatial segmentation we selected in Section 2.4.4, we want to create plots that show the mass features that are important in distinguishing certain segments.



(a) Selecting the number of segments for Gaussian weights

(b) Selected segmentation for Gaussian weights ($r = 2, k = 20, s = 6$)

Figure 9: Plotting the predicted number of segments shows that for Gassian weights and $r = 2$, the resulting number of segments coincide after $s = 6$. Therefore, we choose the segmentation for $r = 2, k = 20, s = 6$, which results in 6 segments.

For this purpose, we write a function summaryPlot that will plot the probabilities for a segment, plot its shrunken mean spectrum and t-statistics, and plot the top 3 ion images associated with the segment.

```
> summaryPlots <- function(dataset, results, model, segment, name, col) {
+     image(results, model = model, key = FALSE, column = segment, main = name,
+         layout = c(3, 2), col = col)
+     plot(results, model = model, key = FALSE, column = segment, mode = "centers",
+         main = "Shrunken mean spectrum", col = col)
+     plot(results, model = model, key = FALSE, column = segment, mode = "tstatistics",
+         main = "Shrunken t-statistics", col = col)
+     top <- topLabels(results, n = 3, model = model, filter = list(classes = segment))
+     image(dataset, mz = top$mz, normalize.image = "linear", contrast.enhance = "histogram",
+         smooth.image = "gaussian")
+ }
```

Now we create these summary plots for the segment associated with the liver.

```
> summaryPlots(pig206, pig206.sscg, model = list(r = 2, s = 6, k = 20), segment = 5,
+     name = "Liver segment", col = mycol)
```

From the shrunken mean spectrum and t-statistics plot in Figure 11, we can see that many of the mass features in the m/z 500–600 range and m/z 800–900 range are positively associated with the liver from their high positive t-statistics, showing that they are systematically enriched in the liver. Many of the low mass range features are negatively associated with the liver, as seen by their negative t-statistics, showing that they are systematically absent in the liver. The three top ions associated with the liver are also plotted.

To contrast with the liver segment, we do the same for the heart segment below.

```
> summaryPlots(pig206, pig206.sscg, model = list(r = 2, s = 6, k = 20), segment = 6,
+     name = "Heart segment", col = mycol)
```

Compared to the liver segment, we see that for the heart segment, only a few mass features have non-zero t-statistics. This shows that fewer ions are associated with the heart compared to the liver.

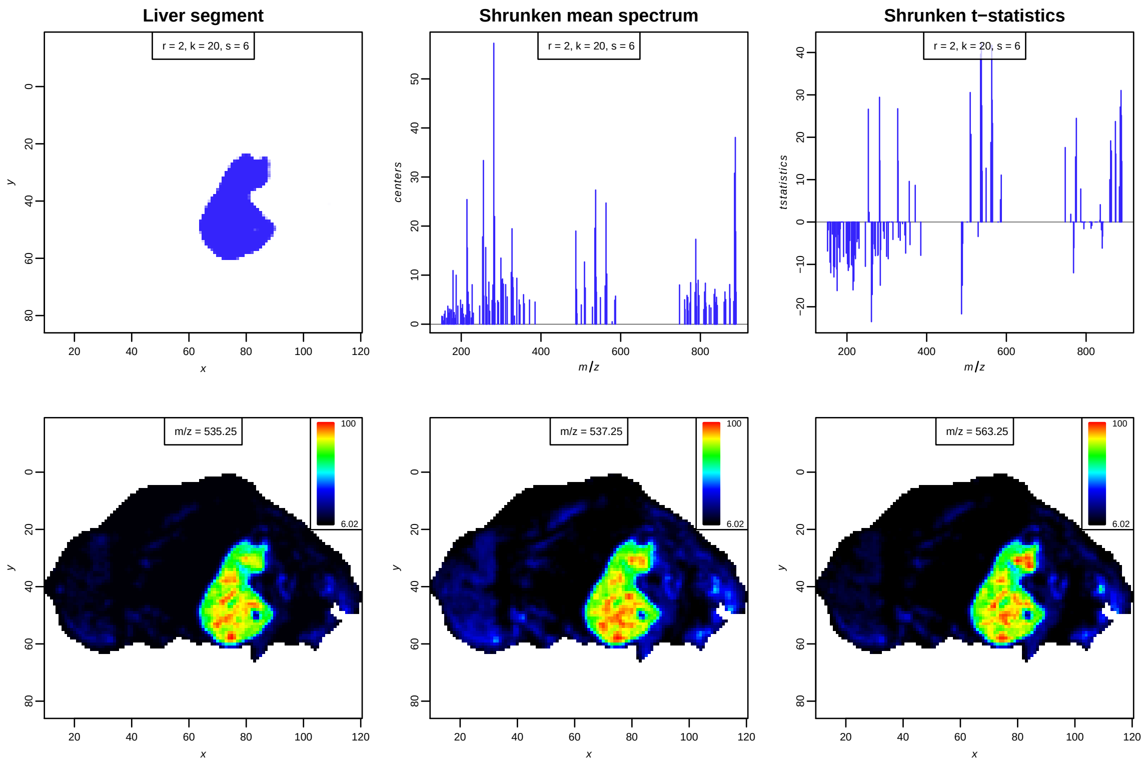


Figure 10: Summary plots of the liver segment for PIGII_206 segmentation with Gaussian weights and $r = 2, k = 20, s = 6$.

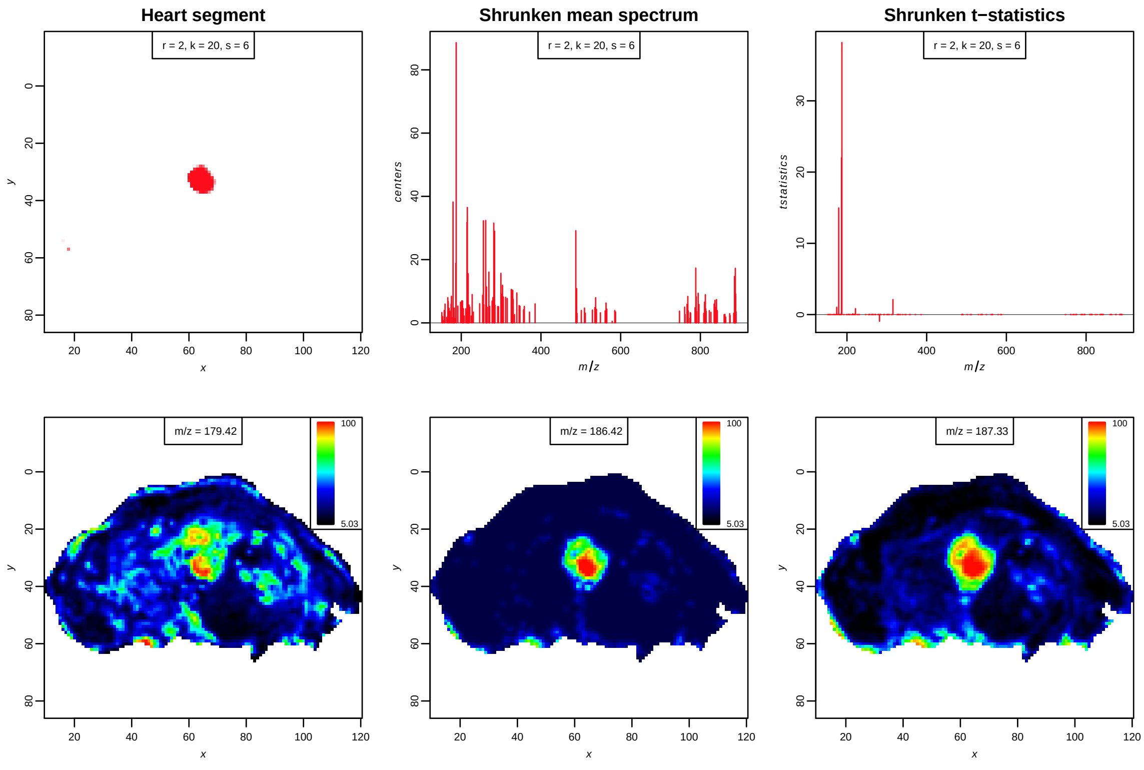


Figure 11: Summary plots of the heart segment for PIGII_206 segmentation with Gaussian weights and $r = 2, k = 20, s = 6$.

3 Additional example

A difficulty with evaluating spatial segmentation methods is that typically the ground truth is not known. Our experimental subjects are often complex biological systems, and there is usually no "gold standard" to compare with our results.

We now perform spatial segmentation on an MS image for which the ground truth is known.

3.1 Evaluation using a cardinal painting

In this first example, we present an oil painting of a cardinal, created to illustrate the utility of DESI imaging. Mass spectra were collected from the pigments in the paint, and we expect that different colors will produce unique patterns of spectral peaks, allowing for spatial segmentation.

```
> data(cardinal, cardinal_analyses)
```

We normalize the data using TIC normalization.

```
> cardinal.norm <- normalize(cardinal, method = "tic")
```

Now we perform peak-picking and alignment as in Section 2.1.2.

```
> cardinal.peaklist <- peakPick(cardinal.norm, pixel = seq(1, ncol(cardinal),
+     by = 10), method = "simple", SNR = 6)
> cardinal.peaklist <- peakAlign(cardinal.peaklist, ref = cardinal.norm, method = "diff",
+     units = "ppm", diff.max = 200)
> cardinal.peaklist <- peakFilter(cardinal.peaklist, method = "freq", freq.min = ncol(cardinal.peaklist
> cardinal.peaks <- reduceDimension(cardinal.norm, ref = cardinal.peaklist, type = "height")
```

Next we perform spatial shrunken centroids clustering on the dataset, with both Gaussian and adaptive weights and a range of parameters for r, k, and s.

```
> set.seed(1)
> cardinal.sscg <- spatialShrunkenCentroids(cardinal.peaks, r = c(1, 2), k = c(10,
+     15), s = c(0, 3, 6, 9), method = "gaussian")
> set.seed(1)
> cardinal.ssca <- spatialShrunkenCentroids(cardinal.peaks, r = c(1, 2), k = c(10,
+     15), s = c(0, 3, 6, 9), method = "adaptive")
```

As in Section 2.4.4, we plot the predicted number of classes against the shrinkage parameter s to help determine the most appropriate model parameters for the data. Figure 12d shows this plot for the segmentations with Gaussian weights, and Figure 12e shows this for adaptive weights.

```
> plot(summary(cardinal.sscg))
```

```
> plot(summary(cardinal.ssca))
```

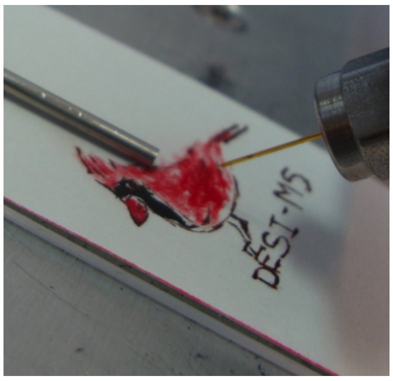
For adaptive weights, the predicted number of segments quickly converge at $s = 3$, while the same does not happen for Gaussian weights until $s = 9$. This is likely because adaptive weights are more appropriate for this dataset, in order to fit the thin "DESI-MS" writing below the cardinal.

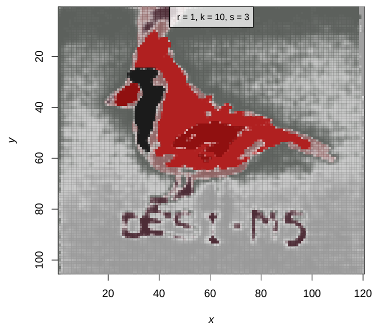
We plot the segmentation for $r = 1, k = 10, s = 3$, resulting in 8 segments, shown in Figure 12b.

```
> mycol <- c("#5C605C", "#4D2C36", "#A1A1A1", "#999999", "#B02020", "#1B1B1B",
+     "#901010", "#906565")
> image(cardinal.sscg, model = list(r = 1, k = 10, s = 3), key = FALSE, col = mycol)
```

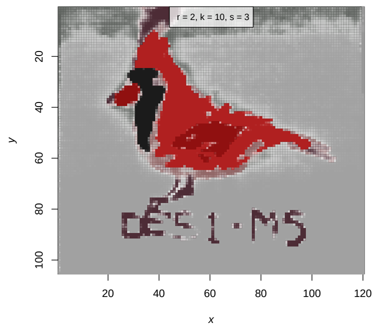
We also plot the segmentation for $r = 2, k = 10, s = 3$, resulting in 8 segments, shown in Figure 12c.

```
> mycol <- c("#1B1B1B", "#A1A1A1", "#906565", "#999999", "#B02020", "#901010",
+     "#5C605C", "#4D2C36")
> image(cardinal.ssca, model = list(r = 2, k = 10, s = 3), key = FALSE, col = mycol)
```

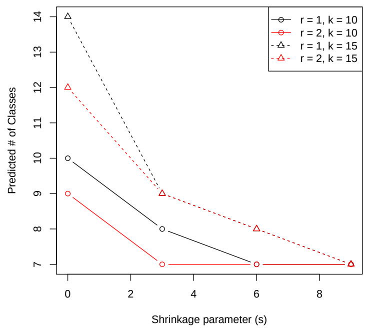

(a) Cardinal painting



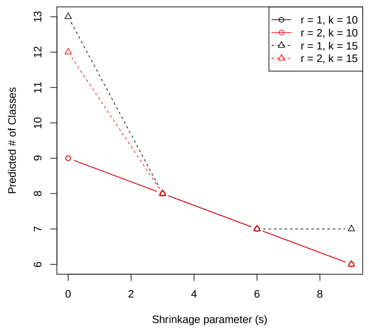
(b) Segmentation for $r = 1, k = 10, s = 3$ with Gaussian weights



(c) Segmentation for $r = 2, k = 10, s = 3$ with adaptive weights



(d) Selecting the number of segments for Gaussian weights



(e) Selecting the number of segments for adaptive weights

Figure 12: Spatial segmentation of the cardinal painting for both Gaussian and adaptive weights, and the plots for selecting the number of segments.

For Gaussian weights, we choose $r = 1$ so that the "DESI-MS" writing is not oversmoothed, and $s = 3$, where the the predicted number of segments are nearly convergent at 8 segments and 9 segments for $k = 10$ and $k = 15$, respectively.

For adaptive weights, we choose $r = 2, s = 3$, since with the adaptive weights we can use a larger smoothing radius without oversmoothing, and the number of segments first converge between both initializations for k at $s = 3$.

We note that the segmentation matches up well against the original painting, so the method is performing well.

4 Session info

- R version 3.4.0 (2017-04-21), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 16.04.2 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.5-bioc/R/lib/libRblas.so`
- LAPACK: `/home/biocbuild/bbs-3.5-bioc/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: Biobase 2.36.0, BiocGenerics 0.22.0, Cardinal 1.7.2, CardinalWorkflows 1.8.0, DBI 0.6-1, ProtGenerics 1.8.0, biglm 0.9-1, matter 1.2.0
- Loaded via a namespace (and not attached): BiocStyle 2.4.0, MASS 7.3-47, Matrix 1.2-9, Rcpp 0.12.10, backports 1.0.5, compiler 3.4.0, digest 0.6.12, evaluate 0.10, grid 3.4.0, htmltools 0.3.5, irlba 2.1.2,

knitr 1.15.1, lattice 0.20-35, magrittr 1.5, rmarkdown 1.4, rprojroot 1.2, signal 0.7-6, sp 1.2-4, stats4 3.4.0, stringi 1.1.5, stringr 1.2.0, tools 3.4.0, yaml 2.1.14

References

[1] Theodore Alexandrov, Michael Becker, Sören-oliver Deininger, Liane Wehder, Markus Grasmair, Ferdinand Von Eggeling, Herbert Thiele, and Peter Maass. Spatial Segmentation of Imaging Mass Spectrometry Data with Edge-Preserving Image Denoising and Clustering. *Journal of proteome research*, 9(12):6535–6546, 2010.

[2] Theodore Alexandrov. MALDI imaging mass spectrometry: statistical data analysis and current computational challenges. *BMC Bioinformatics*, 13(Suppl 16):S11, November 2012.

[3] Theodore Alexandrov and Jan Hendrik Kobarg. Efficient spatial segmentation of large imaging mass spectrometry datasets with spatially aware clustering. *Bioinformatics (Oxford, England)*, 27(13):i230–8, July 2011. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3117346&tool=pmcentrez&rendertype=abstract, doi:10.1093/bioinformatics/btr246.

[4] Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences of the United States of America*, 99(10):6567–6572, 2002.

[5] Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu. Class Prediction by Nearest Shrunken with Applications to DNA Microarrays. *Statistical Science*, 18(1):104–117, 2003.