Electronic Supplementary Material (ESI) for Analytical Methods.

Supporting Information

Chemometric analysis of MALDI mass spectrometric images of three-dimensional cell culture systems

The following is an instructional manual for the processing MALDI IMS datasets. As expected, this is specific for our work, but the commands can be adapted for other uses. Some general guidance is given on how to execute specific commands, but users should read comments in the .m files provided for a thorough review. The authors assume the reader has read through the supporting information theory secion and has a general knowledge of MATLAB. If MATLAB is new to the user, several general texts are available. While it describes an older version, the authors recommend “Getting Started with MATLAB: A Quick Introduction for Scientists and Engineers” by Rudra Pratap (ISBN: 0199731241).

We the authors provide these instructional materials in good faith and we believe them to be free of errors to the best of our knowledge. This supporting information was written for clarity and ease of execution. As such, in some cases the command lines may not include the most direct way to accomplish a desired task, but (as the authors feel) they are easier for the novice user to follow and execute. The authors are in no way liable for any damages (e.g. software inoperability, data loss, etc.) associated with the use of this material.

While some workarounds have been attempted in some cases, several toolboxes are required including the Signal Processing, Bioinformatics, Image Processing, and Statistic Toolboxes. It is also assumed that all of the following .m files (also Supporting Information) are located within the MATLAB directory:

open_read_interp.m
mzPCA_raw.m
selectdata.m [2]
alignspectra.m [3]
mzPCA_processed.m
gapstatkmeans.m [4]
kmeansclustering.m
PCALDA.m
gaussfilterdata.m [1]
plotandselect.m
subdatapreprocessing.m
align_and_normalize.m
image_gen.m
gapstatHCA.m [4]
HCAclustering.m
PCALDA_higher_order.m

1 K. C. Waldron and N. J. Dovichi, *Anal. Chem.*, 1992, **64**, 1396-1399.

2 This m-file was taken directly from the MATLAB file exchanged and used as is. Copyright (c) 2007, John D'Errico, http://www.mathworks.com/matlabcentral/fileexchange/13857-graphical-data-selection-tool/content/selectdata.m
Redistribution and use in source and binary forms, with or without modification, are permitted provided that redistributions in binary form must reproduce the copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. This software is provided by the copyright holders and contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the copyright owner or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

3 X. F. Li, H. J. Ren, X. C. Le, M. Qi, I. D. Ireland and N. J. Dovichi, *J Chromatogr A*, 2000, **869**, 375-384.

4. The supplied command lines were adapted from those given in Martinez, W. L., Martinez, A. R. and Solka, J. L. Exploratory data analysis with MATLAB. 2nd Ed., CRC Press, 2011.

PART ONE: PCA of Relevant Pixels

Step One: Load Data (Optional)

Summary: Use the open_read_interp command to load the mzXML files into the MATLAB programming environment. Because each pixel contains slight variations in the absolute *m/z* values recorded (eg. 1000.97 vs 1100.05), all spectra are interpolated such that they all share the same mass axis.

CompassXport 5.0 (freely available on Bruker's website) was used here to generate mzXML files for each pixel from the raw data. As such, the ability to open and read in data using our MATLAB commands is selective for this method. However, users who have an alternate approach to importing their data in MATLAB can skip this step or adapt the open_read_interp.m file solely for the purposes of interpolating their data. If taken from somewhere else, going forward users must have the following: a matrix containing XY coordinates for all mass spectra, an *m/z* axis, and mass spectral data that are interpolated such that all spectra share the same mass axis. See the open_read_interp command for more specific details on formatting for each of these matrices (rows vs columns).

If using our approach, all mzXML files must be located within a specific directory that has no other files within it. Any files that were under 100 kB were deleted prior to executing this command as these files contained incomplete spectral information associated with irrelevant pixels (e.g. an incomplete spectrum recorded by the mass spectrometer with only 20 points).

Command:

```
[mzaxis,interpdata,coords]=open_read_interp(directory,dataname,lowmzlimit,hig
hmzlimit,spacing);
```

The inputs include the name of the directory where the mzXML files are located, a data name that the workspace will be saved as when the file loading is finished, and a way for the user to set the lower limit, upper limit, and spectral *m/z* resolution of the loaded spectra. The outputs contain the *m/z* axis, the interpolated spectra for all pixels, and the XY coordinates for all pixels.

Example:

```
[mzaxis,interpdata,coords]=open_read_interp('C:\Users\Hummon
Lab\Desktop\ctrl_sp1_2','Ctrl1_2_raw.mat',7000,14000,1);
```

This command would look for files within a folder named ctrl_sp1_2 on the Desktop, interpolate all mass spectra such that all data spans from 7000 to 14000 *m/z* at unit resolution, and would save the spectra as a file called Ctrl1_2_raw.mat within the same directory. The three outputs would be named mzaxis, interpdata, and coords for the *m/z* axis, the interpolated data, and the XY coordinates (respectively). The user should be aware that this command can take a significant time to execute, depending on the amount of data to be imported.

Step Two: Trim Data (Optional)

Summary: You may trim the ends of the mass spectra to isolate only the relevant *m/z* range. To do this without accidentally altering the original data (in case you need it), we make a copy and trim the copy. You must trim both the mzaxis and interpdata. The trimming is done on a point index basis, not on an absolute *m/z* basis.

Commands/Example:

```
[trimmed]=interpdata;
[mzaxis_t]=mzaxis;
trimmed(:,1:1000)=[];
mzaxis_t(1:1000)=[];
```

The above commands duplicate both interpdata and mzaxis as well as remove the first 1000 points from each. You should look through the mzaxis vector prior to executing this command to see the range of *m/z* values that are removed. Using the data above that has unit mass resolution, these commands would remove masses 7000 to 7999. The remaining data would span from *m/z* 8,000 to 14,000 (inclusive) giving 6001 data points per spectrum.

This procedure can also be adapted to remove irrelevant *m/z* values on the upper end of the dataset (high mass range). While our example data below in subsequent steps does not include the following step, as an example a user could type

```
trimmed(:,5502:end)=[];
mzaxis_t(5502:end)=[];
```

The above commands would remove the last 499 data points from the mass spectra. Using the data above that has unit mass resolution, these commands would remove masses 13,501 to 14000, giving 5501 data points per spectrum. Again, this was not done for the subsequent data analyses, but the commands provide an example of how to accomplish the trimming task on the upper end of data.

Step Three: Gaussian Smooth Data (Optional)

Summary: The following commands will utilize a Gaussian smooth to remove noise from mass spectra. If used, this step **must** be performed prior to downsampling to prevent errors associated with peak shifts. Specifically, each of the mass spectra within the image are convolved with a Gaussian function to remove noise. After this command, a graph will pop up which will allow the user to assess the quality of the smoothing step. The command can be repeated multiple times with different parameters (overwriting the previous iteration) to allow adjustment if necessary.

Command:

```
[filtered]= gaussfilterdata(mzaxis,dataraw,sigma,pixelnum);
```

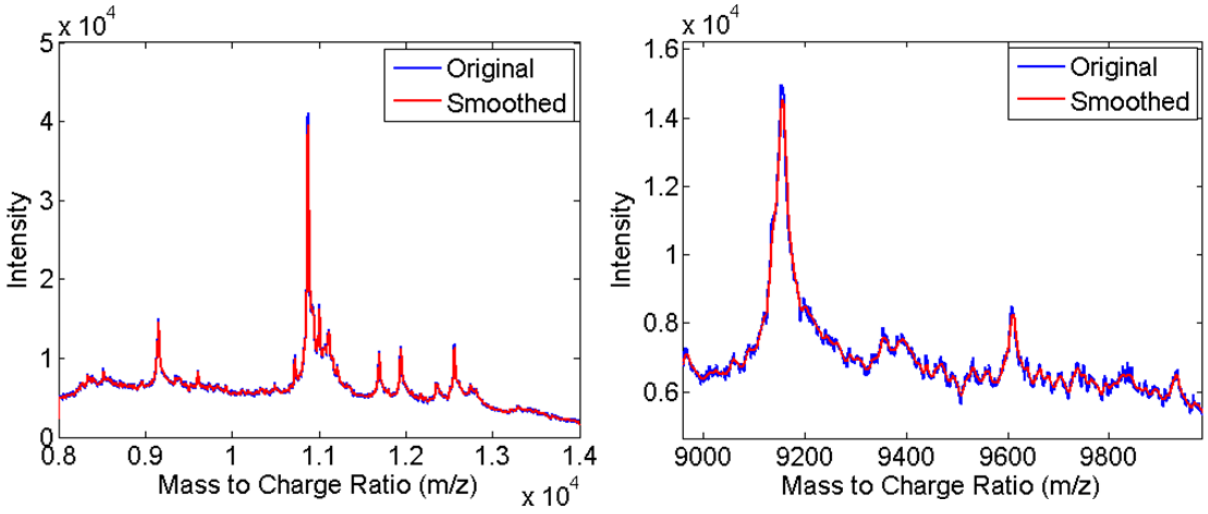
The inputs include the mass axis (may be trimmed), the raw data (again may be trimmed), a scalar number that helps defines the width of the Gaussian smoothing function, and a scalar number used to identify a specific pixel. This pixel number will be the spectrum that is viewed to assess the quality of the data processing.

Example:

```
[trimmed_f]=gaussfilterdata(mzaxis_t,trimmed,3,88);
```

Here, this command would smooth the data with a Gaussian that has a standard deviation of 3 *m/z* (because we have unit resolution), and the spectrum corresponding to the 88th pixel will be viewed to assess the quality of the Gaussian smoothing.

An example graph produced by this command is shown on the left below; an enlargement of one area is also presented on the right. Note that the two traces shown are the original data (blue) and the smoothed data (red). As expected with a Gaussian smooth, there is a balance between noise removal and peak amplitude suppression. The user should use what is best suited for their particular data set.



Step Four: Downsample Data (Optional)

Summary: You may choose to downsample the mass spectra to enable faster processing. If the computer lacks sufficient processing power, the subsequent PCA command sometimes will not execute on large datasets. In this case, MATLAB would return an out of memory error when trying to execute. There is no hard-and-fast rule as to when this can occur and depends only on the computer being used. As an alternative, this step could be completed after the first iteration of PCA (provided your computer has sufficient processing power to perform PCA on large datasets). The user would then just have to adapt the names of the matrices used in the commands.

This step takes advantage of MATLAB's downsample command.

Commands/Example:

```
[mzaxis_d]=downsample(mzaxis_t,10);
for x=1:size(trimmed_f,1)
   trimmed_d(x,:)=downsample(trimmed_f(x,:),10);
end
clear x
```

Alternatively, if one lacks the Signal Processing toolbox, one could simply use indexing to extract every nth point as shown below:

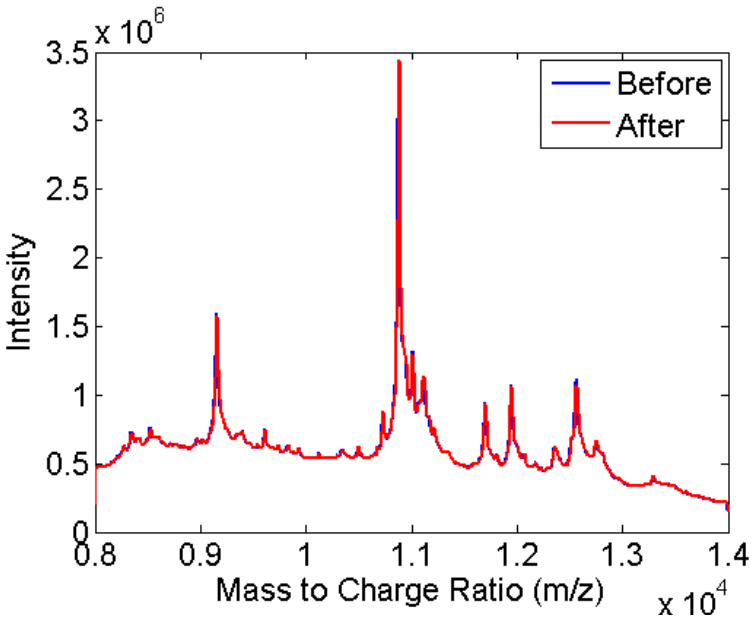
Alternate Commands/Example:

```
[mzaxis_d]=mzaxis_t(1:10:end);
for x=1:size(trimmed_f,1)
trimmed_d(x,:)=trimmed_f(x,1:10:end);
end
clear x
```

In both cases, the inputs include the mass axis (which you could have trimmed or not), the raw spectra (could be raw, trimmed and/or smoothed), and a scalar that represents the rate at which you wish to downsample which is typed in the 1st and 3rd lines of the above commands. Here, because the number 10 was used, these commands would downsample the trimmed smoothed data at every 10th point. Therefore, the downsampled range would span 601 points instead of 6001 (it keeps the extra point on the end as is). For the commands used here, a downsampled mass axis (mzaxis_d) and a downsampled, trimmed smoothed data set (trimmed_d) would be generated. As another example, if you wanted to down-sample by 5 to get every 5th point, replace the number 10 in the 1st and 3rd lines of the above commands with the number 5.

You can type something similar to the commands on the next page to visualize the down-sampling result (total ion spectra are shown below before and after downsampling).

```
figure
plot(mzaxis_t,sum(trimmed_f),'LineWidth',2)
hold
plot(mzaxis_d,sum(trimmed_d),'r','LineWidth',2)
legend('Before','After')
set(gca,'FontSize',16)
set(gcf, 'color', 'white');
xlabel('Mass to Charge Ratio (m/z)','FontSize',16)
ylabel('Intensity','FontSize',16)
```



While downsampling is sometimes necessary, it can reduce your resolution and cause slight errors in the location of peaks (depending on how sharp peaks are, down-sampling can catch the edge of a peak, rather than its maximum). Therefore, if comparing the results of the discriminant spectra to those of a commercial mass spectrometer programming package, you should allow an error of $\pm$ 2 of the resolution of newly-down-sampled spectra.

Step Five: PCA on All Pixels

Summary: This command will generate the scores that are used in the next step to select relevant pixels.

Command:

```
[Aproj_raw]=mzPCA_raw(rawdata);
```

The input is the raw data (may be raw data or trimmed and/or Gaussian-smoothed and/or downsampled data). The output Aproj_raw contains the scores of the pixels along the first 20 (default) principal components.

Example:

```
[Aproj_raw]=mzPCA_raw(trimmed_d);
```

Here, this command would perform PCA on the trimmed, Gaussian smoothed, and downsampled raw spectra.

Step Six: Isolation of Relevant Pixels

Summary: This command will allow users to select and isolate a subset of specific pixels from a PCA score plot of the raw data. When the first plot comes up, immediately left-click and hold down the mouse to "lasso" the points. However, if the plot is too small, you can drag the edges or maximize the window to make it larger as long as you do not click inside of the graph. Make a complete enclosure before letting go of the mouse. Highlighted data points will change color from blue to red and be encompassed in a yellow color as you lasso. When you are finished, a second graph will pop up which shows the XY coordinates of the pixels that you have selected. You may repeat the process as many times as you need to such that the proper data is isolated. Irrelevant pixels associated solely with matrix, noise, etc. should aggregate outside of the range of the other pixels. If you choose, you may downsample the data after this command executes with subdataraw as an input.

Command:

```
[subcoords,subdataraw]=plotandselect(Aproj_raw,coords,rawdata);
```

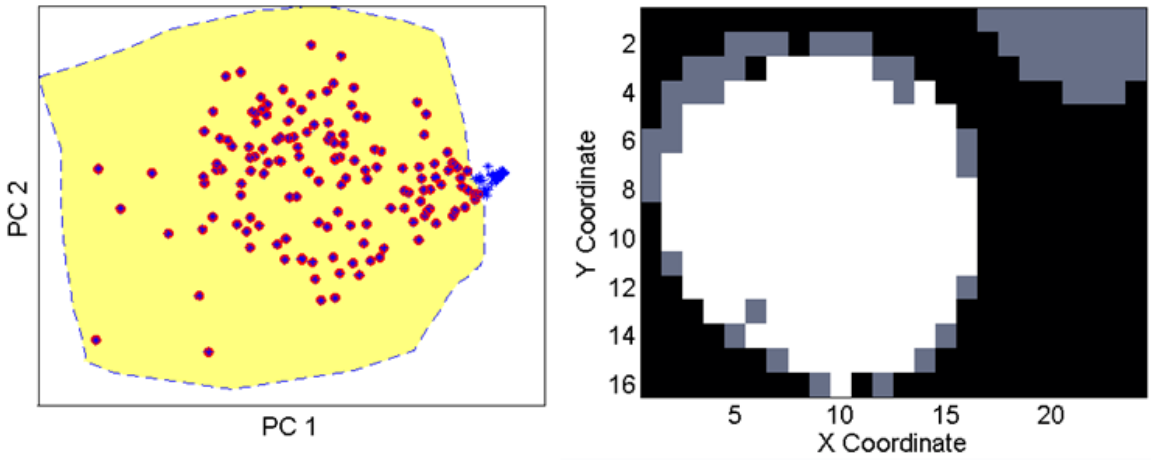
The inputs include the PC scores of the raw data, the XY coordinates of the data, and the raw data (may be raw data or trimmed and/or Gaussian smoothed and/or downsampled data). The outputs include the subset of XY coordinates and mass spectral data corresponding of the pixels the user selected.

Example:

```
[subcoords,subdataraw]=plotandselect(Aproj_raw,coords,trimmed_d);
```

Here, the user would isolate a selected pixel subset, extracting the corresponding XY coordinates and the trimmed, smoothed, downsampled data. The data subsets generated would be named subcoords and subdataraw for the XY coordinates and the selected data, respectively.

When isolating relevant pixels the results during the execution of this command, the first graph should resemble something shown below on the bottom left. Note: Other data will have a different distribution of points unique to that particular dataset. After selection of the relevant pixels, another graph will appear that shows the XY coordinates of the isolated subset of pixels as shown below in the bottom right. White pixels are the ones that the user selected and grey pixels are those that were discarded. The black pixels were never imaged by the MALDI. The numbers on the axes of this second graph correspond to X and Y coordinates of the pixels. If the user makes a mistake or doesn't agree with the assignments, the same command line can be repeated, overwriting the data generated by previous iterations.



Step Seven: Pre-Processing of Selected Subset of Data

Summary: This command will log-transform and top hat filter the subset of mass spectra selected from the previous step. During this command, a graph will pop up which will allow the user to assess the quality of the processing steps. The command can be repeated multiple times with different parameters (overwriting the previous iteration) to allow adjustment if necessary.

Command:

```
[processed]=subdatapreprocessing(mzaxis,subdataraw,tophat,pixelnum);
```

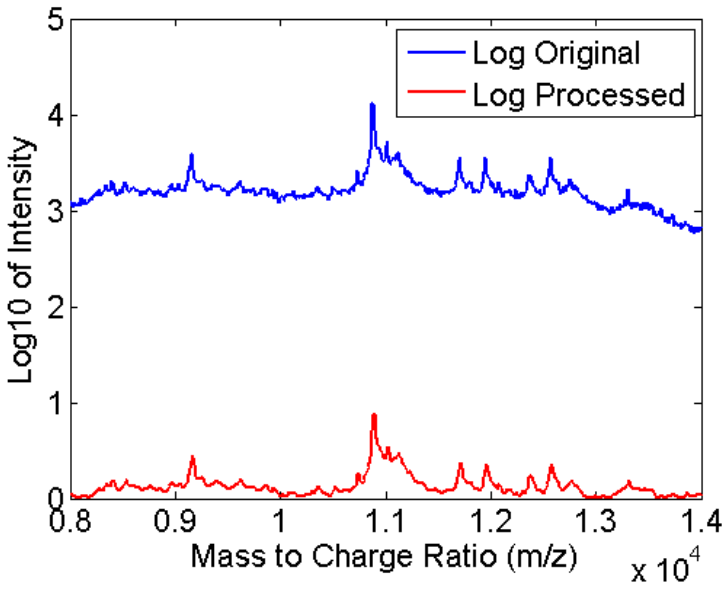
The inputs include the mass axis (may be trimmed and/or downsampled), the subset of data that includes only the relevant pixels (again may be trimmed and/or Gaussian-smoothed and/or downsampled), a scalar number that defines the top hat filer, and a scalar number used to identify a specific pixel. This pixel number will be the spectrum that is viewed to assess the quality of the data processing.

Example:

```
[processed_d]=subdatapreprocessing(mzaxis_d,subdataraw,250,75);
```

Here, this command would log-transform the data, use a top hat filter of 250 (removes baseline changes greater than 250 data points wide), and the spectrum corresponding to the 75th pixel will be viewed to assess the quality of the top hat filter.

An example graph produced by this command is shown below that is used to assess the top hat filter only. If the legend is in the way, use the white mouse button to move it elsewhere. Note that the two traces that are viewed are the log-transformed original data prior to top hat filtering (blue) and the log-transformed data after processing the top hat filter.



Step Eight: Align and Area-Normalize

Summary: This command will align all data such that all peaks should occur at the same *m/z* and normalize all spectra to unit area to correct for pixel-to-pixel differences in absolute intensity of the recorded mass spectra. To accomplish the aligning, two peaks are identified by the user that exist within all pixels of the data. For each of the two peaks, the algorithm will first determine the *m/z* value of the peak as it occurs within each spectrum. Next, the median *m/z* values of the two peaks are identified. Third, all spectra are aligned such that the two peaks occur at the same *m/z* ratios in all spectra (the median *m/z* values previously identified). The user can also determine how to best normalize the data: area, height, or no normalization. A graph will also be displayed to show the final resulting data.

Command:

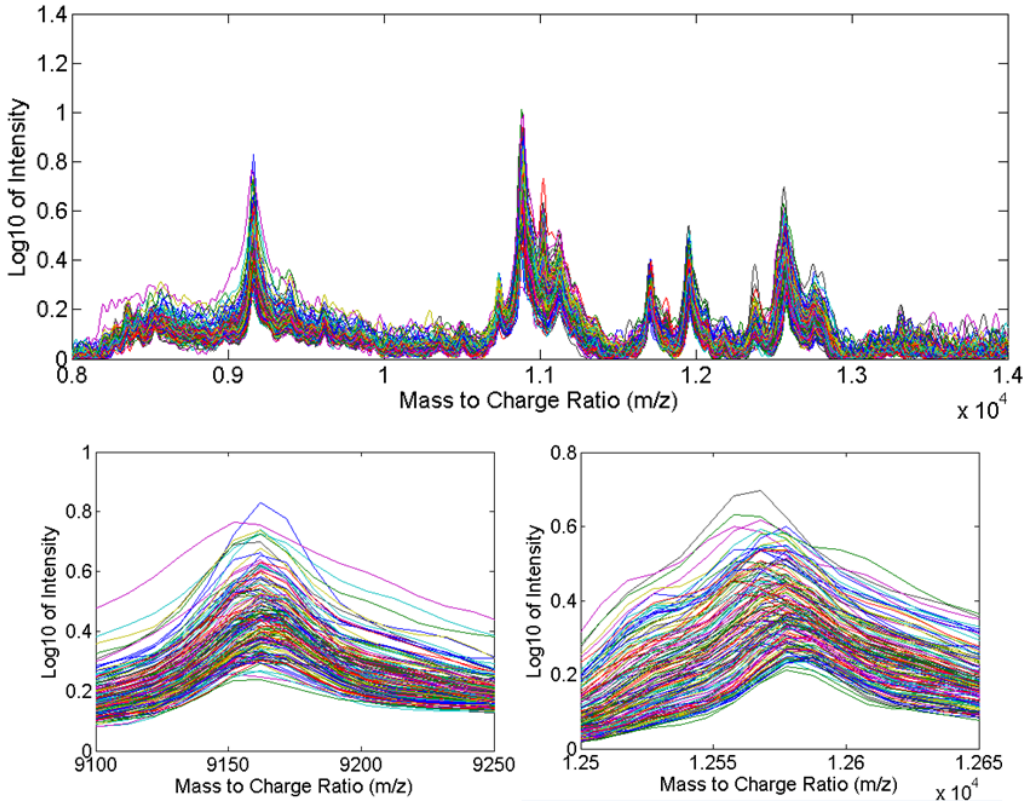
```
[finaldata]=align_and_normalize(mzaxis,processed,P1low,P1high,P2low,P2high,type);
```

The inputs include the mass axis (may be trimmed and/or downsampled), processed spectra of the relevant pixel subset, a range of *m/z* values that encompass one peak of lower *m/z* in all spectra, a range of *m/z* values that encompass one peak of higher *m/z* in all spectra, and the specified normalization type. The output will consist of aligned, area-normalized, processed data of the relevant pixel subset.

Example:

The user must first plot the data to identify the mass spectral ranges of the two peaks used for aligning. The command below can be used (although the plot will not be labeled as it is below).

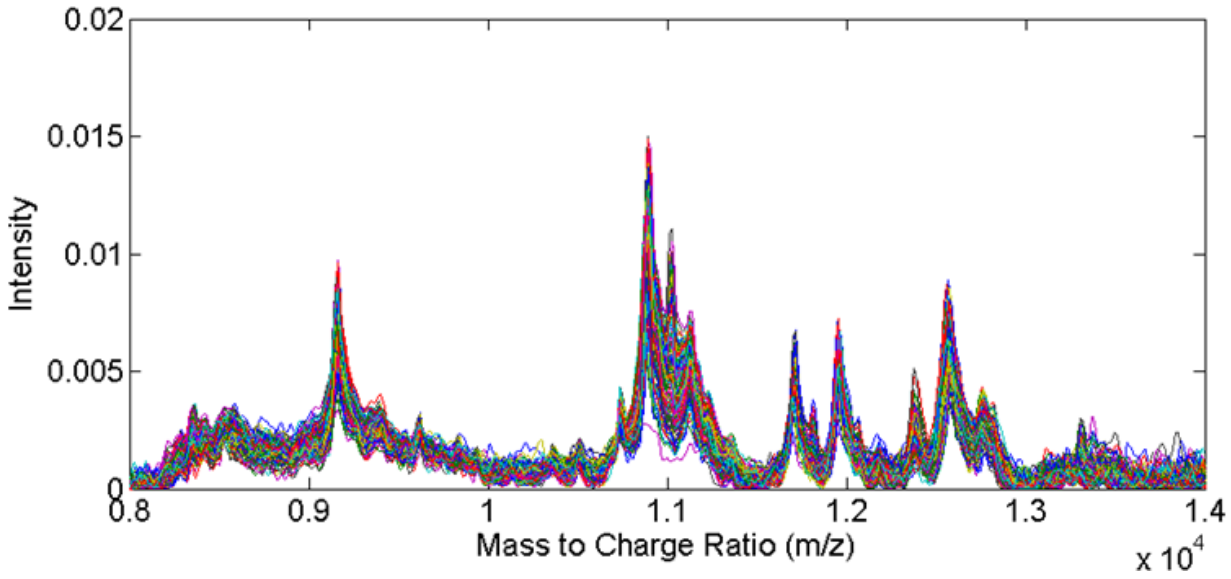
```
figure;plot(mzaxis_d,processed_d);
```



Zooming in shows that a lower *m/z* peak ranges between 9120-9200 *m/z* among all pixels and a higher *m/z* peak ranges between 12520-12650 *m/z* among all pixels. Using this information, we may write the following command

```
[finaldata_d]=align_and_normalize(mzaxis_d,processed_d,9120,9200,12520,12650,
'area');
```

Here, this command would align all spectra such that the peaks that occur between 9120-9200 *m/z* and 12520-12650 *m/z* are aligned to the median *m/z* value for those peaks across all pixels. The output data are also area-normalized. When this command has finished executing, a graph will display to show the final result of the aligning and normalization algorithm as shown below. In this case, the intensity axis represents normalized intensity because area-normalization was used.



Step Nine: Perform PCA on the Processed Data of Relevant Pixels

Summary: This command will perform PCA on the processed data of the relevant pixels. A Scree plot will also be generated for the user to estimate the proper number of principal components to retain.

Command:

```
[Aproj_final,Vc_final]=mzPCA_processed(finaldata);
```

The input includes the final processed data. The outputs include the scores of each principal component along each pixel (Aproj_final) and the principal components (Vc_final).

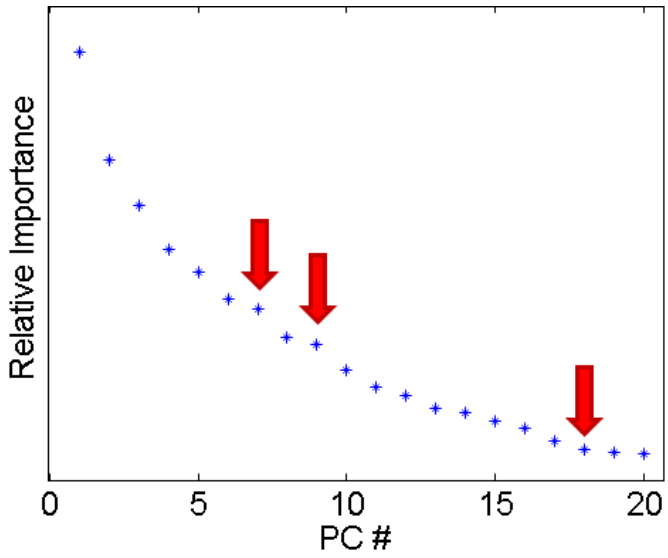
Example:

```
[Aproj_final_d,Vc_final_d]=mzPCA_processed(finaldata_d);
```

This command generates the scores and principal components of the final processed data. Upon execution, the Scree plot is generated.

This command automatically mean-centers the data. Any data set that is mean-centered prior PCA will cause a slight defect in the appearance of the Scree graph. Statistically, mean-centering removes one degree of freedom from the data being analyzed because all spectra were used to calculate a spectral average. Normally, the total number of principal components (relevant and noise principal components) equals the number of data elements (e.g. pixels) within a data set. However, by subtracting the spectral average prior to PCA, one principal component has already been removed by the user. Since principal components are calculated in order of decreasing importance, mean-centering typically causes the last principal component calculated to contain zero variance. The logarithmic-transformation of the variance captured by each principal component then gives an error/outlying value for the last principal component that can be ignored.

Zoom into the first part of the graph to estimate the proper number of principal components to retain. An example is shown below:



From the above graph, three “elbows” can be seen (8, 10, and 19). These points are taken to be the maximum number of relevant PCs, but many chemometric researchers (including these authors) remove one PC from the elbow; the elbow maximum is considered to contain the first

noise PC so many users deduct one PC from the elbow to deduce the proper number of PCs to retain. In this case, there are three options: 7, 9, and 18 (red arrows above). Unfortunately, this phenomenon of multiple breaks is sometimes seen with the Scree graph where the proper number of principal components to retain is unclear. It is possible that 18 principal components describe all relevant sources of information. However, it is also possible that only 7 or 9 principal components describe relevant information while all higher numbered principal components represent heteroscedastic (non-random) noise. One way to determine which is which is to plot the principal components (Step Ten) and visualize the spectral trends. Alternatively, the user may opt to choose fewer numbers of principal components moving forward in the process as the quality of discriminant spectra is highly dependent on noise. While it is possible that some information would be left out of the discriminant spectrum using fewer principal components, the peaks that are present in the discriminant spectrum will be of better quality (easier to visualize and have a higher signal-to-noise ratio).

Step Ten: Plot the Principal Components and Score Images (Optional)

Summary: This command will allow users to visualize the principal components produced by the previous commands and to generate a spatially-localized spectral trend "heat map". Two graphs will pop up for the users, one displaying the principal component and the other presenting the heat map. Users have several options on how to best visualize the data as described in the comments section within the image_gen.m command file.

Command:

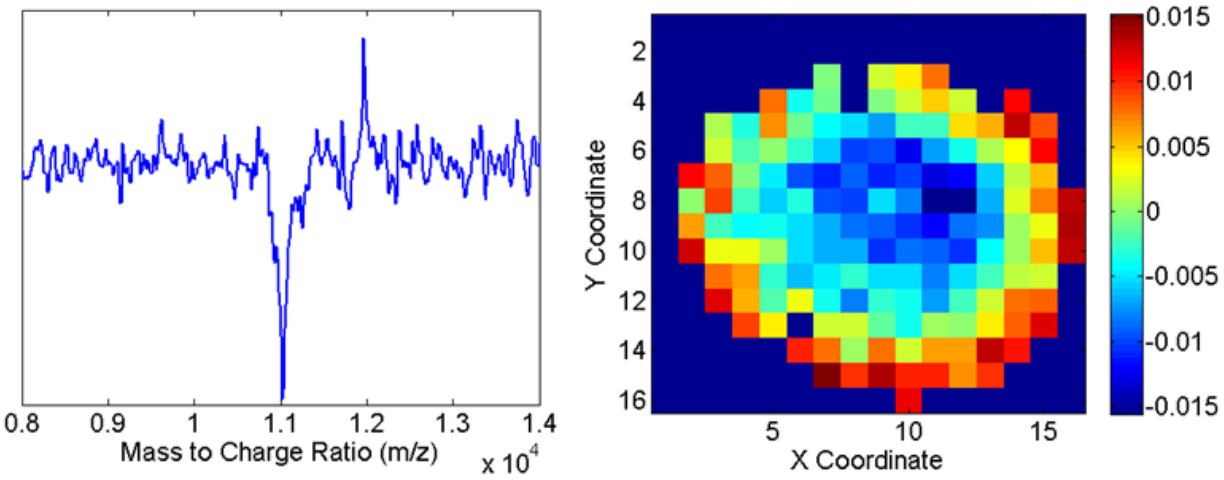
```
image_gen(Aproj_final,Vc_final,subcoords,mzaxis,PCnum,negate,cleangraph);
```

The inputs include the scores and principal components of the processed data (from the previous mzPCA_processed command), the subset of XY coordinates selected with the plotandselect command, the mass axis of your data, the principal component number you wish to visualize, and a set of values as to whether or not the user wishes to negate the principal component for easier viewing and to remove ancillary features from the heat map graph for better viewing (T for yes, F for no for the latter two inputs). There are no outputs.

Example:

```
image_gen(Aproj_final_d,Vc_final_d,subcoords,mzaxis_d,1,'F','F');
```

The above command will plot the spectral trend identified within the first principal component without negation and will generate a labeled heat map as shown below:

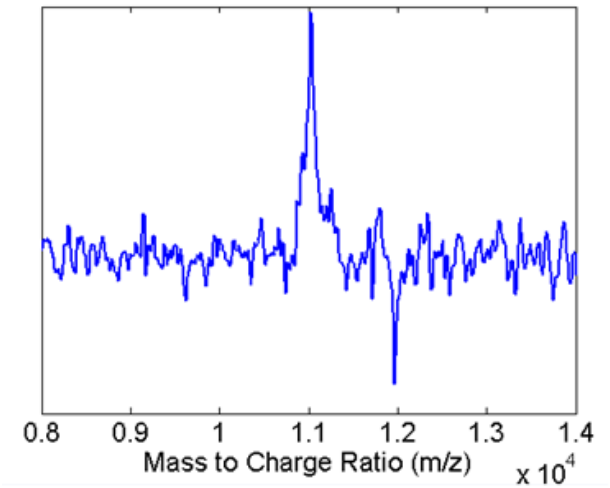


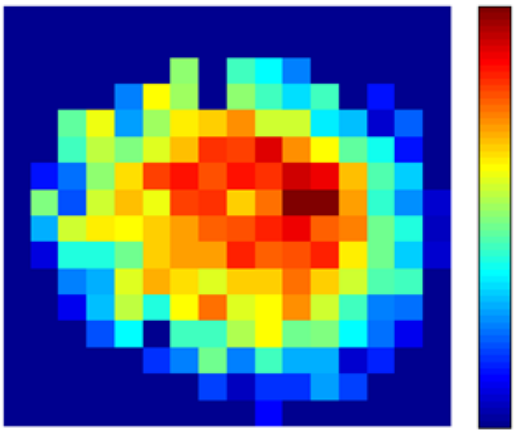
The spectral trend is shown on the left and the heat map of the spectral trend is on the right. The color scale represents the magnitude of the scores of this particular principal component for each pixel; large positive values indicate a positive correlation with a particular pixel and large negative values indicate a negative correlation. Note: In the heat map, the deepest darkest blue areas indicate pixels that were not isolated by the plotandselect command.

Because principal components are calculated without direction, it is perfectly reasonable to invert (multiply by -1) the principal component. To do this and to generate a heat map that is "cleaner" for easier exporting, the following command can be used:

```
image_gen(Aproj_final,Vc_final,subcoords,mzaxis_d,1,'T','T');
```

Note that negating the principal component also inverts the image as shown on the next page.


0.8
0.9
1
1.1
1.2
1.3
1.4
Mass to Charge Ratio (m/z)
x 10^4



PART TWO: Clustering

Step Eleven: Estimation of Cluster Number (Optional)

Summary: The following commands estimate the number of clusters that exist within a dataset using the gap statistic. If the user knows the proper number of distinct groups beforehand, this step may be omitted. Because both *k*-means and hierarchical clustering analysis (HCA) can be used in the estimation of the gap statistic, both sets of commands are used here. Choose the one that is the most appropriate. In either case, the gap statistic can generate two different types of reference distributions needed for its calculation. When finished, a graph will be generated that mimics Figure 5.10 in Martinez and Martinez (Martinez, W. L., Martinez, A. R. and Solka, J. L. Exploratory data analysis with MATLAB. 2nd edn, CRC Press, 2011.)

Command:

For *k*-means use

```
[khat]=gapstatkmeans(Aproj_final,PCs,type);
```

The inputs to this command include the scores generated by the mzPCA_processed command on the processed data, the proper number of relevant principal components, and a choice of type for the two types of reference distributions that are used for the gap statistic (See Martinez and Martinez). The inputs for type are 'uniform' and 'pc'. The authors used the pc method as it will preserve the underlying distribution of the data set. The output khat indicates an estimate of the number of distinct groups of pixels within the data set.

For HCA use

```
[khat]=gapstatHCA(Aproj_final,PCs,type);
```

The inputs have the same meaning as above.

Example:

A *k*-means example is shown below

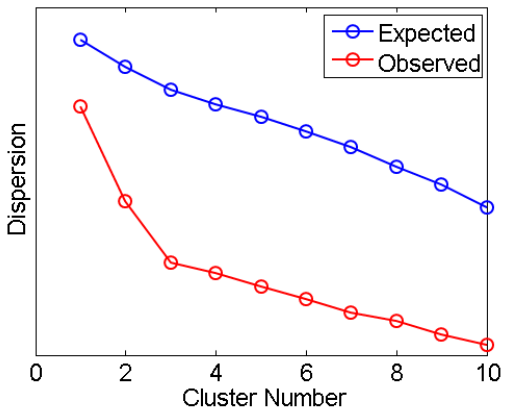
```
[khat]=gapstatkmeans(Aproj_final_d,7,'pc');
```

This command performs the gap statistic calculation using *k*-means to generate clusters. Only 7 of the 20 principal components generated by the mzPCA_processed command are used for cluster estimation. Also, this command uses the pc method for a reference data set.

An HCA example is shown below

```
[khat]=gapstatHCA(Aproj_final_d,7,'pc');
```

This command performs the gap statistic calculation using HCA. Only 7 of the 20 principal components generated by the mzPCA_processed command are used for cluster estimation. Again, this command uses the pc method for a reference data set.



When either approach is used, a graph will be generated that displays dispersion as a function of cluster number for the reference and test data sets as shown on the left. See Martinez and Martinez for more information.

Step Twelve: Cluster Analysis

Summary: The following commands perform cluster analysis on the scores of the relevant PCs of the processed data by either *k*-means or HCA. Choose the desired approach that is the most appropriate. Both methods can generate different clustering results as these methods cluster data points in a different manner. Regardless of the choice of clustering algorithm, two graphs are displayed to the user: average traces of all processed pixel spectra for a particular cluster type and a map highlighting the spatial localization of the pixels.

Command:

For *k*-means use

```
[allmeanspectra,idx]=kmeansclustering(Aproj_final,PCs,khat,subcoords,mzaxis,finaldata);
```

The inputs to this command include the scores generated by the mzPCA_processed command on the processed data, the proper number of relevant principal components, an estimate of the number of clusters that exist within the data set (taken from the gap statistic or *a priori* knowledge), the XY coordinates of the relevant pixels, the *m/z* mass axis, and the final normalized, aligned, processed data (from the align_and_normalize command). The outputs consist of averages of all the log-transformed processed spectra for each cluster identified and a set of indices identifying which pixel belongs with which cluster.

For HCA use

```
[allmeanspectra,idx]=HCAclustering(Aproj_final,PCs,khat,subcoords,mzaxis,finaldata);
```

The inputs and outputs have the same meaning as above.

Example:

A *k*-means example is shown below

```
[allmeanspectrakm,idxkm]=kmeansclustering(Aproj_final_d,7,3,subcoords,mzaxis_d,finaldata_d);
```

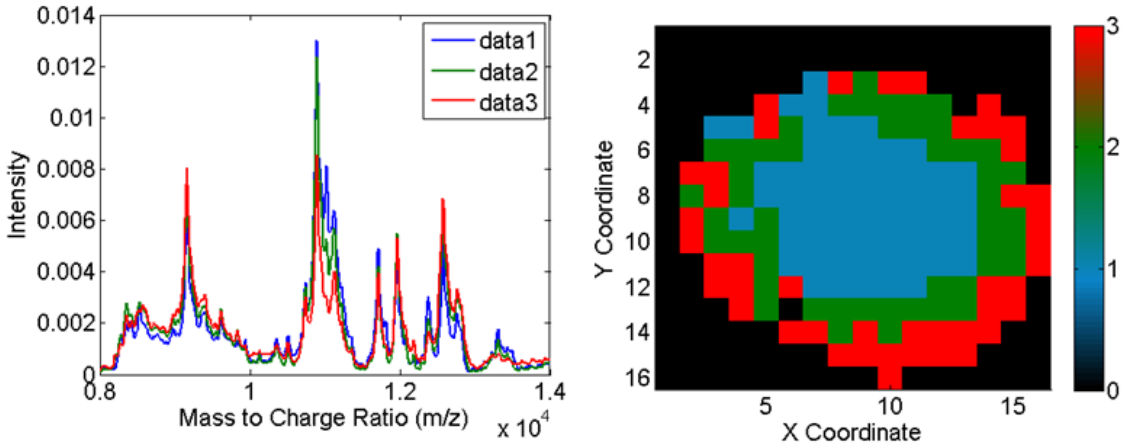
This command performs *k*-means clustering using the scores of the first seven principal components of the aligned and processed data to identify three groups of pixels. All other inputs have their same meaning as previously denoted.

An HCA example is shown below

```
[allmeanspectraHCA,idxHCA]=HCAclustering(Aproj_final_d,7,3,subcoords,mzaxis_d,finaldata_d);
```

This command performs HCA clustering using the scores of the first seven principal components of the aligned and processed data to identify three groups of pixels. All other inputs have their same meaning as previously denoted.

When the clustering command is finished, two graphs will be displayed as shown on the next page.



The graph on the left displays the average processed spectra for pixels of a specific group. The legend is numbered data1, data2, data3… to identify the cluster number associated with each average trace. The graph on the right displays an XY coordinate map of the spatial location of each type of pixel. Note: The colorbar identifies which region corresponds to which cluster number. In this example, blue identifies the pixels associated with cluster number 1, green identifies the pixels associated with cluster number 2, red identifies the pixels associated with cluster number 3, and black represents irrelevant pixels not lassoed from the plotandselect command and those not originally imaged (0th cluster).

Do not worry about cluster order (1 vs 2 vs 3) as that is arbitrarily determined by the clustering algorithm. Note that the cluster order may change as the command is repeated, but the groupings of points should remain consistent. Only in the case where three clusters are found by the algorithms will the color of the average trace correspond with the color on the spatial map (e.g. the average spectra for all green pixels within the right graph is shown as the green trace in the left graph). In all other cases, users should use the graph legends to interpret the data. The color scheme for the spatial map will also change if the number of clusters does not equal three.

As stated in the main manuscript, for HCA and *k*-means clustering algorithms provided here, Euclidean inter-point distances were used. In addition, the method of complete linkage was used to calculate the distance between clusters of points in HCA. There are a multitude of other options available to the user, each that can generate slightly different clusters. For more information (and to adjust the supporting command lines provided here) see MATLAB's help files on pdist (inter-point HCA distances), linkage (cluster-to-cluster distances), and kmeans (user can set a different inter-point distance metric using the Name & Value arguments).

Before moving on to the PCALDA step, the user must remember/record/acknowledge which cluster number corresponds to which spatial region. These numeric indices will be used as inputs for the PCALDA command. There is no way to record this in MATLAB and re-running the command line could possibly associate the different regions with a new cluster number. Here, the outer region of the spheroid is represented by cluster number 3, the middle region of the spheroid is represented by cluster number 2 and the core region of the spheroid is represented by cluster number 1. Again, the assignments are arbitrary.

PART THREE: PCA-LDA

Step Thirteen: Performing PCA-LDA and Obtaining a Discriminant Spectrum

Summary: The following command utilizes the relevant principal component scores of the normalized, aligned, processed data and the indices obtained by the clustering algorithm to obtain a discriminant spectrum that segregates between two groups of data. This command generates two graphs: a graph of the discriminant spectrum and a score plot of the first two principal components with the linear discriminant. As a note to the user, the borders/lines drawn on the discriminant spectra assume that the data has been mean-centered (which is done by default by the mzPCAprocessed command).

Command:

```
[discrimspectra]=PCALDA(mzaxis,Aproj_final,Vc_final,PCs,idx,clustera,clusterb
,graphtype);
```

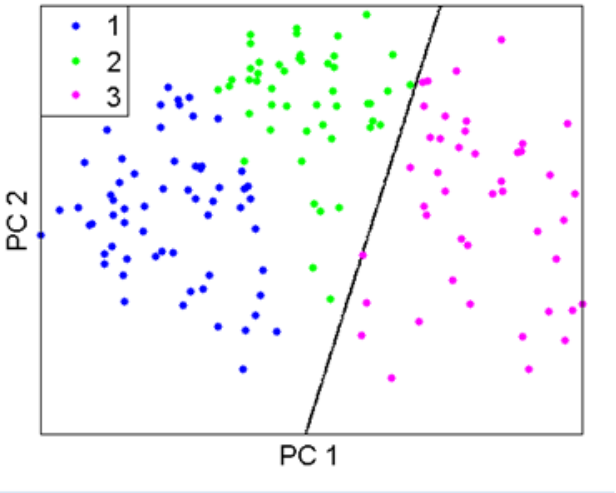
The inputs to this command include the mass axis, the scores and principal components generated by the mzPCA_processed command, the proper number of relevant principal components, the indices of cluster number for all pixels from the clustering command, the two cluster numbers that the user wishes to segregate against, and a user input as to the format of the graph for the discriminant spectrum (singletrace or colored). The output is an array containing the discriminant spectrum. Note: Positive values in the discriminant spectrum are associated more strongly with cluster a and negative values are associated more strongly with cluster b. Again, knowing which cluster number is which is essential to properly attributing spectral trends to the appropriate group of pixels.

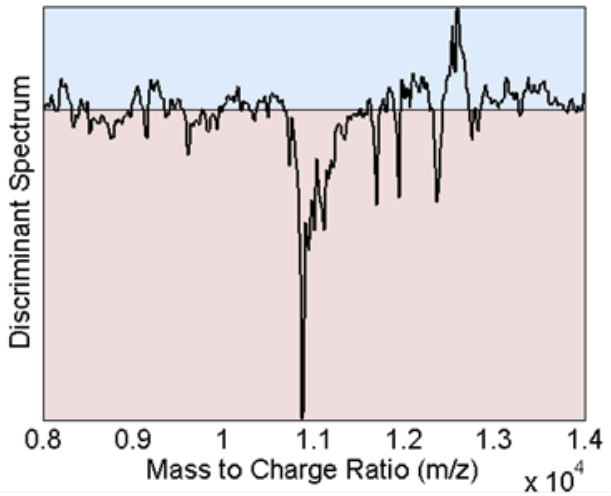
Example:

A *k*-means example is shown below

```
[discrimspectra_o_vs_m]=PCALDA(mzaxis_d,Aproj_final_d,Vc_final_d,7,idxkm,3,2,
'colored');
```

This command generates a discriminant spectrum using the first seven principal components of the processed relevant pixel subset, comparing cluster number 3 versus cluster number 2. The resulting discriminant spectrum will give a colored, shaded graph. Using the data and associations from step 11, positive values in the discriminant spectrum would yield spectral trends more strongly associated with the outer region (cluster 3) of the spheroid and negative values would yield spectral trends more strongly associated with the middle region of the spheroid (cluster 2). The graphical outputs of the above command are shown below:



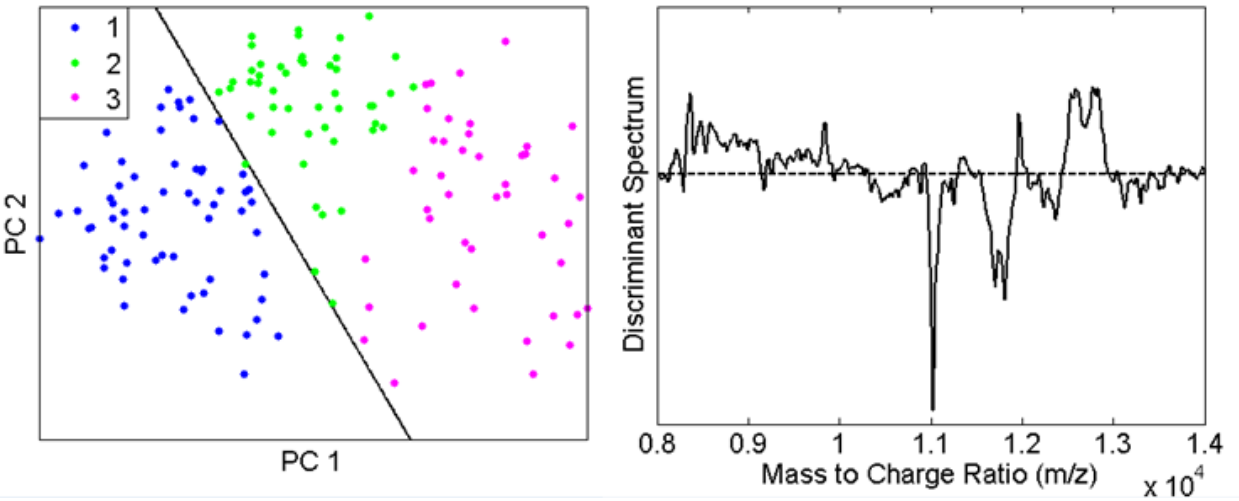


The graph on the left shows a cluster plot of the principal component scores for each group identified. The dark line is the linear discriminant/boundary calculated by PCALDA. Note that its position successfully segregates the two clusters of data. The graph on the right shows the discriminant spectrum. When 'colored' mode is used, positive values will be enclosed within an area of blue while negative values will be enclosed in red. The line through the center marks zero (this will only work for mean-centered data).

Continuing with the above example

```
[discrimspectra_m_vs_c]=PCALDA(mzaxis_d,Aproj_final_d,Vc_final_d,7,idxkm,2,1,
'singletrace');
```

will generate a discriminant spectrum that differentiates between the middle (cluster 2) and core (cluster 1) regions. Using the data and associations from step 11, positive values in this discriminant spectrum would yield spectral trends more strongly associated with the middle region (cluster 2) of the spheroid and negative values would yield spectral trends more strongly associated with the core region of the spheroid (cluster 1). The graphical outputs of the above command are shown below:



As expected, the linear discriminant shifts to separate groups 2 and 1. Note that the discriminant spectrum is not shaded because the 'singletrace' distinction was used in the command line. The dotted line is used to mark the zero position.

In either type of graph, the user may wish to use the zoom tool to identify peaks within the calculated discriminant spectra.

While all relevant principal components are used in the calculation of the discriminant spectrum, by default, the scatterplot generated by the PCALDA command only shows the first two principal components. These may or may not highlight the appropriate PCs that ideally segregate two data sets. An adapted PCALDA command can be used to plot other PCs against one another:

```
[discrimspectra]=PCALDA_higher_order(mzaxis,Aproj_final,Vc_final,PCs,idx,clus
tera,clusterb,graphtype,PCx,PCy);
```

Similar to the other PCALDA command, the inputs contain the mass axis, the pixel scores, the relevant principal components, the number of relevant principal components, the cluster indices of all pixels, the cluster numbers of the two groups you wish to discriminate against, and the

type of discriminant spectrum graph desired. Two new inputs are now included: the two PCs you wish to plot against one another. You may choose any values for PCx and PCy up to a certain maximum (the number of relevant principal components). Remember, different clusters of points may share similar spectral characteristics, so you will likely not have segregation of the different clusters in all dimensions.

Using the data and matrix names above, an example command is given by

```
[discrimspectra]=PCALDA_higher_order(mzaxis_d,Aproj_final_d,Vc_final_d,7,
idxkm,2,1,'colored',2,3);
```

In this case, a plot would be generated of PC3 (y-axis) vs PC2 (x-axis).