**MIBI low-level analysis tutorial**

**1. How does data comes off the MIBI?**

For each FOV (field of view), the MIBI software will generate a point. In it you will find the following hierarchy: RowNumber0 -> Depth\_Profile0. In it there will be Depth folders corresponding to the number of depths in the run. Each one will have 2 files used in the analysis:

1. Image.msdf – the TOF data for each pixel
2. ImageAccum.msdf – the TOF data accumulated over the entire image

**2. Files needed for analysis**

In addition to the data files you will need:

1. The run xml file generated by the MIBI software.
2. A .csv file delineating you panel. The csv file should look like this:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Isotope | Label | Start | Stop | BaselineStart | BaselineEnd |
| 12 | C | 11.8 | 12.1 | 11.5 | 11.6 |
| 23 | Na | 22.8 | 23.1 | 22.5 | 22.6 |
| 28 | Si | 27.8 | 28.1 | 27.5 | 27.6 |
| 31 | P | 30.8 | 31.1 | 30.5 | 30.6 |
| 40 | Ca | 39.8 | 40.1 | 39.5 | 39.6 |
| 56 | Fe | 55.8 | 56.1 | 55.5 | 55.6 |
| 89 | Antibody 1 | 88.7 | 89 | 88.5 | 88.6 |
| 128 | Background | 128 | 132 | 127.5 | 127.6 |
| 169 | Antibody 2 | 168.7 | 169.2 | 168.5 | 168.6 |
| 176 | Antibody N | 175.7 | 176.2 | 175.5 | 175.6 |
| 181 | Ta | 180.7 | 181.2 | 180.5 | 180.6 |
| 197 | Au | 196.7 | 197.2 | 196.5 | 196.6 |

This file is used to determine which regions of the continuous spectrum will be converted into images and how many images you will get. For the example above, twelve images will be generated. Of these three are of antibodies. The rest are:

* C, Na,P,Ca,Fe are elements naturally present in the sample.
* Si, Ta and Au are elements highly abundant in the slide.
* Background is a large region in which no signal is expected and will be used for background subtraction.

It is recommended to always extract these elements in addition to the antibody channels, because they can report on the general structure of the tissue and are useful for background subtraction.

The fields in the .csv file are:

1. Isotope – the mass of the element.
2. Label – the name of the element/antibody. These names will be used throughout the pipeline, so it is recommended to make them short and legible (e.g. better use ‘CD4’ instead of ‘cluster of differentiation 4’). Since these will be used for file/variable names, it is best to stick to the following characters: A-Za-z0-9, and avoid spaces, punctuation marks and brackets. Good example: *PD-1*. Bad example: *SMAD5 (phospho S463+S465)*.
3. Start and Stop – These are the edges of the mass window used for extraction. In the example above, anytime the program sees a mass in the range 11.8-12.1 it will be counted as carbon. These should be set the first time you run a panel and then kept constant for the rest of the runs to ensure consistency. If unsure, a good place to start is by integrating from -0.2 to + 0.2 around the mass of the isotope. For example, for an antibody conjugated to mass 148 it is recommended to set ‘Start’ to 147.8 and ‘Stop’ to 148.2.
4. BaselineStart and BaselineEnd – These are the edges for the region used as the baseline for each one of the elements. Should be set from -0.5 to -0.4 from the mass of the isotope. For example, for an antibody conjugated to mass 148 it is recommended to set ‘Start’ to 147.5 and ‘Stop’ to 147.6.

Important: The script expects the values to appear in increasing value with no overlap. Otherwise it will crash. Make sure that channels are inserted in increasing order and that there is no overlap between extracted regions.

**3. Setting up an analysis folder**

The .msdf files that come off the machine are very heavy. So, it is useful to save them in a remote location and only copy them to your hard drive for extraction. This can be done by working in Box and then unsyncing the folder, or by working on an external drive.

For example, the folder for ‘Experiment 1’, in which 3 points were acquired should be:

* Experiment 1
  + Point 1 directory
  + Point 2 directory
  + Point 3 directory

**4. Steps in the analysis**

|  |  |  |
| --- | --- | --- |
| **Step** | **Script** | **Description** |
| 1 | MIBIextractRawImages.m | Script is used to calibrate the spectra and extract tif files |
| 2 | MIBIgetBgSubtractionParams | Interactive script to get parameters for background subtraction |
| 3 | MIBIremoveBg.m | Remove Bg from data |
| 4 | MIBIgetNNthreshold.m | Interactive script to get the NN threshold for denoising |
| 5 | MIBIdenoise.m | Denoising |
| 6 | MIBIgetAggregateRemovalParams.m | Interactive script to get the parameters for aggregate removal |
| 7 | MIBIfilterAggregatesAllChannels.m | Aggregate removal |

**4.1 MIBIextractRawImages.m**

This script allows you to move from a tof spectra file (.msdf), generated by the MIBI to an N-dimensional image. It performs the following steps:

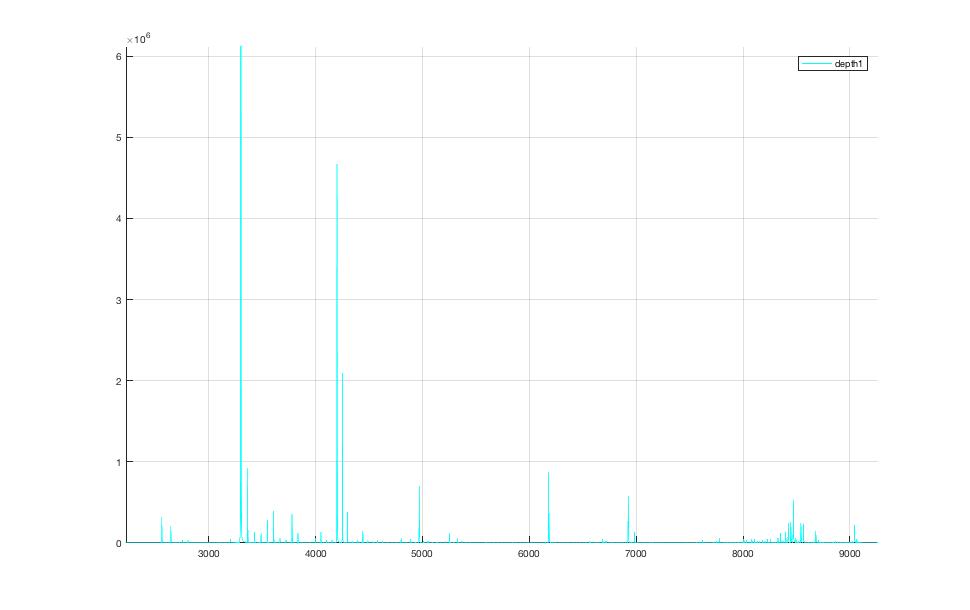
1. Spectra calibration
2. Cross correlation and summation of depths
3. Generation of QC plots

To run the script you need to adjust the parameters to match your data. The default parameters in the repository are set for the analysis of the sample data provided in the directory ‘SampleData-Moly’. You should first try to analyze this data to make sure that you can do this correctly, before moving on to your own data.

* fileNameXML – the file name of the run xml.
* fileNameMass – the file name for the csv file of your panel.
* dataDir – the directory of the raw data.
* processedDataDir – the directory to which the processed data will be written
* pointNumber – the number of the point that is being analyzed.
* depthStart and depthEnd – the depths that you want to extract. For a run with only a single depth, both variables should equal 1.
* calibrateSpectra – a boolean (1 or zero), indicating whether you want to calibrate the spectra for this run. If you want to use the parameters from the instrument set this to zero. Otherwise, if you suspect that the spectra needs to be calibrated, set this to 1.
* First – a boolean indicating whether this is the first run of the script. This is only important if you want to calibrate the spectra. If you want to calibrate the spectra, the scrip will behave differently if this is the first time that you run it or not (see below).
* SpectraVec – values used for spectra calibration (see below).

Spectra calibration

Data comes off the instrument as time of flight. We want to convert it to mass. There is a quadratic equation relating the two, whose parameters, a and b, can change according to the tuning settings. To calibrate the spectra, we need to indicate two points for which we know the time and the mass, and then the script can solve and find the parameters. We typically use Na and Au, as these elements are at the two edges of the acquired spectrum and are usually highly abundant and easily identifiable in our sample. We know the mass of Na (22.93) and Au (196.96). To see what was the time of flight of these elements in the current run, we need to see the run spectrum. To this end, we run the script MIBIextractRawImages.m with the parameter First=1. This should plot (Figure 1) the spectrum, summed over the entire image as such:



Counts

TOF

Au

Na

Zooming-in on this image, we can see that the time of flight for Na in this run was 3300 and for Au was 9042.

To use these values for calibration we go back to the script and insert them into the SpectraVec variable as such: SpectraVec = [3300, 22.93, 9042, 196.96]. This tells the script to find parameters such that a TOF of 3300 will be converted to a mass of 22.93, and a TOF of 9042 will be converted to 196.96.

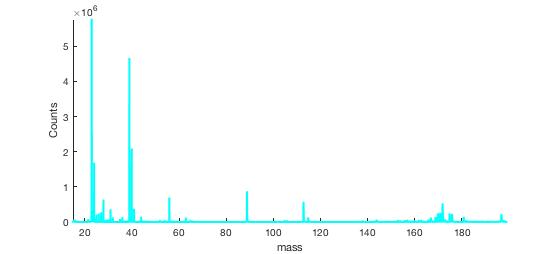
The script also plots the spectrum as a function of mass, after calibration. So, if we now run with

calibrateSpectra=1;

First=1;

spectraVec=[3300, 22.93, 9042, 196.96];

you should see the image for the TOF, as well as the following image of counts per mass:



Gold

Vimentin

dsDNA

Ca

Na

This image should make sense. You should see peaks for natural elements (such as Na and Ca) in their expected masses. You should see a relatively large peak for your nuclear marker (e.g. dsDNA on 89 in this case), and additional peaks on the elements that you included in your panel (e.g. 113,115,141-176 in this example.

Use this plot to double check the values that you set on your integration windows in the .csv file. Once you see how the data looks like, you may want to slightly change these.

Now that you are sure of your spectra calibration and integration windows, you can rerun the script with First = 0. This will extract the images. Beware, this script may take a few minutes to run, depending on the size of your data and strength of your computer.

The output for the script is:

1. Folder of TIF images
2. Results folder with figures of quality controls
3. data.mat – a matlab file with the data. It contains:
   1. massDS – a table with the information from the CSV
   2. countsAllSFiltCRSum – a matrix of size [x-dimension,y-dimension,number-of-channels] with all the data
   3. totalIonFiltSum – image of total ion intensity

**4.2 MIBIgetBgSubtractionParams.m**

This is an interactive script, which allows you to choose a proper threshold for background subtraction. The MIBI data often has background signal, usually coming from bare slide regions. This signal in common across different channels and can obscure the data unless it is removed.

The script works on a user-defined background channel. This can be a region with no antibody labeling (e.g. masses 128-132) or one of the channels that are strong on the slide (e.g. Si, Ta and Au). This background channel will be used to generate a binary mask according to a user-defined threshold. The binary mask will later be applied to all other channels, and the signal in positive regions in the mask will be attenuated by a specified number of counts. The purpose of this script in to identify adequate parameters for this process.

The script generates the following plots:

1. Image of the raw background channel.
2. Smoothed histogram of the counts on the raw channel. This is useful for setting the threshold for signal and noise. A recommended signal by Otsu’s method is shown in red.
3. The binary mask of the background channel identified according to the parameters in the script. Regions in yellow will be subtracted across all channels.
4. An evaluation channel, defined by the user, before subtraction.
5. An evaluation channel, defined by the user, after subtraction according to the parameters in the script.

Parameters to change in case of inadequate background removal:

* bgChannel - channel used for background signal. Can use Au/Ta/Si/Background.
* gausRad - radius of gaussian to use for signal smoothing (typically 1-3)
* t - threshold for binary thresholding (0-1)
* removeVal - value to remove from all channels in background-positive areas (increase for more aggressive removal).

You can test a set of parameters on several cores by adding several paths to the corePath variable. Once you have identified good parameters for background removal, proceed to applying those using the next script.

**4.3 MIBIremoveBg.m**

The purpose of this script is to use the parameters identified in the previous section to subtract background for a list of cores.

You will need to set the following variables:

* corePath – a path to all the cores you want to subtract background from. Several paths can be specified by separating them by commas
* bgChannel, gausRad, t, removeVal – parameters for subtraction. See explanation above. These should be set to the values identified as optimal in the script above.

Output:

* TIFsNoBg – directory with TIFs after bg removal
* dataNoBg.mat – matlab file with the following variables:
  + massDS – a table with the information from the CSV
  + countsNoBg – a matrix of size [x-dimension,y-dimension,number-of-channels] with all the data after background subtraction

**4.4 MIBIgetNNthreshold.m**

This is an interactive script that allows you to choose a good threshold for noise removal for each channel. For each channel, for each positive pixel a density score is calculated by a KNN approach. This script slows you to identify a density threshold, which separates signal from noise. A different threshold should be identified for each channel.

The script has three parts:

1. Loading the data. If you are working on a single point then this is fast. If you’re working on many points this may take some time. You want to run this part only once, when first loading your data.
2. Calculating nearest neighbor density for the marker of choice. The time it takes to calculate the density increases for high-abundance markers. You want to run this part once for every marker that you analyze.
3. Plotting density distributions. A nice marker should have a bimodal density distribution, like in the figure below. In this case a good threshold for distinguishing signal from noise is ~3.5 (dashed red line).



Signal

Noise

Average distance to 25

nearest neighbors

Counts

1. Denoising according to the threshold and plotting images of before and after. You want to run this part for several thresholds until settling on one that you like.

Parameters:

* corePath - Path to cores that you want to evaluate for noise reduction. Specify several paths by separating with commas
* massPath - Path to the CSV file with the panel data. The script expects the same panel for all cores.
* load\_data – Boolean (0/1) indicating whether you need to load the data. If you’re working on many cores it is recommended to change to 0 after the first time that you run the script to save the loading time.
* plotChannel - Channel that you want to denoise. Should be spelled as in your CSV file.
* new\_channel - Boolean (0/1) indicating whether you need to calculate nearest neighbor density for this channel. This should only be done once. After the first time that you run the script for a specific channel you can change to 0 to save the calculation time.
* t - Threshold used for separating signal and noise. Play with this number until you're happy with the denoising results.
* capImage - Capping value for plotting. Set to lower to see dynamic range of low-abundant antigens.
* K = 25 - Number of neighbors to use for density calculation. Usually can be kept as 25.

It is recommended to first run the script with load\_data=1 and new\_channel=1 and to start with an easy channel (e.g. CD8), to get the hang of it. After the first time that you run the script you can turn both load\_data and new\_channel to zero to save time while homing in on the exact threshold that you like for CD8. Once you found a good threshold, write it down in the NoiseT column of your panel csv file. For the example above, that number should be 3.5. You can now proceed to identifying the threshold for the next channel. Make sure to turn new\_channel back to one for the first run of the next channel!

Tips:

* It is recommended to test your threshold on more than one point.
* Different tissue types may need different thresholds for noise removal. If your cohort contains more than one type, test the parameters on all of them.
* After you’ve identified a threshold that you like, store it in your panel csv file in a column names ‘NoiseT’.

**4.5 MIBIdenoise.m**

This script denoises the data according to the thresholds identified in the previous step and stored in the ‘NoiseT’ column in the panel csv folder.

Parameters:

* corePath - Paths of points to denoise. Add several by separating with commas
* cleanDataPath - Path to store clean data. Data will be renumbered Point1..PointN.
* massPath - Path to panel csv. Make sure it has a column 'NoiseT' which has the noise threshold for each channel.
* K - Number of neighbors to use for density calculation. Usually can be kept as 25.

Output:

* TIFsNoBg – directory with TIFs after denoising.
* dataDeNoiseCohort.mat – matlab file with the denoised data

**4.6 MIBIgetAggregateRemovalParams.m**

This is an interactive script that allows you to choose parameters for aggregate removal for each channel. The script works by gaussian-smoothing the data and then removing connected components below a certain size. These should remove any experimental aggregates as well as small clumps left over after noise removal. The script has two parts:

1. Loading the data – you can save time by running this only once.
2. Aggregate filtering according to the threshold given by the user.

Parameters:

* corePath - Path to cores that you want to evaluate for aggregate removal. Specify several paths by separating with commas
* massPath - Path to the CSV file with the panel data. The script expects the same panel for all cores.
* load\_data – Boolean (0/1) indicating whether you need to load the data. If you’re working on many cores it is recommended to change to 0 after the first time that you run the script to save the loading time.
* plotChannel - Channel that you want to denoise. Should be spelled as in your CSV file.
* gausFlag - flag of whether to do gaussian smoothing or not.
* gausRad - gauss radius for smoothing (No need to play with this normaly).
* capImage - Capping value for plotting. Set to lower to see dynamic range of low-abundant antigens and higher for high-abundant antigens.
* t - Threshold used for filtering aggregates. Play with this number until you're happy with the filtering results.

After you’ve identified appropriate t thresholds for each channel:

* Store thresholds in your csv file in a column named ‘AggFilter’.
* Create a column called ‘GausFlag’. This should have 1 if this threshold was found using gaussian filtering (default), or 0 if you decided to remove the gaussian step.

**4.7 MIBIfilterAggregatesAllChannels.m**

This script remove aggregates according to the thresholds identified in the previous step and stored in the ‘AggFilter’ column in the panel csv folder.

Parameters:

* corePath - Path of points to denoise. Script expects to find inside folders named Point1..PointN.
* massPath - Path to panel csv. Make sure it has the columns ‘GausFlag’ and ‘AggFilter’.
* coreNum – Number of points in the corePath folder
* gausRad – gauss radius for smoothing (No need to play with this normaly)

Output:

* TIFsNoAgg – directory with TIFs after aggregate removal
* dataNoAgg – mat file with data after aggregate removal

**5. Comparing titers:**

A common analysis task when building a panel is to compare stains for an antibody (either across titers, across tissues or both). Here, we’ll go over a few helpful scripts to aid in this task.

**5.1 MIBIcompare\_titers.m**

This script compares a single channel between different cores.

Parameters:

* corePath - Path of points to work on. Several can be specified by separating with commas
* Headers – Headers for each one of the points (e.g. ‘low titer’,’med titer’ etc.).
* Cap – Capping value for plotting. A good number for most antigens is 5. Set to lower to see dynamic range of low-abundant antigens and higher for high-abundant antigens.
* Channel – channel to compare.
* K - Number of neighbors to use for density calculation. Usually can be kept as 25.
* First – 1 if this is the first time running. If loading many points, you can change to 0 after first run to save loading time.

The script generates the following plot to help compare titers:

1. *Images of the target channel across all points, all capped the same way*. These are useful for looking at the signal and the noise and seeing how they compare visually between titers.
2. *Histograms of positive intensity counts.* For each point, a histogram is plotted showing the number of pixels that had a value of 1, 2 etc. Noise is generally random and generates pixels with counts of 1. Signal generally generates more pixels with higher counts. These histograms allow you to see whether a different titer improves the number of high-intensity pixels.
   1. Important: Be mindful of the tissue architecture when comparing. Differences in these histograms can also result from comparing fields with a different number of positive cells. Use this metric as a guide in your overall decision. Don’t take it as a solid truth.
3. *Histograms of density.* In MIBI data, positive signal of weak markers may manifest as higher density of positive pixels, rather than higher intensity. For each point, a histogram is plotted showing the KNN-density for all positive pixels. High density (low values on x-axis) are signal and low density (high values on x-axis) are noise. A good titer will provide a good separation between signal and noise.
   1. Important: Be mindful of the tissue architecture when comparing. Differences in these histograms can also result from comparing fields with a different number of positive cells. Use this metric as a guide in your overall decision. Don’t take it as a solid truth.

Tips and tricks when choosing titers:

1. Always work on data after it was background-subtracted. A common mistake is to interpret background signal as real signal.
2. Be conscious of bleed-through (when the signal of one channel carries over into another). It is always good to check that your signal is not a result of bleeding from the -1, -16 or -17 channels.
3. Use other channels in your decision. For example: if you’re not sure about the signal of Tbet (a transcription factor expressed in T helper cells), compare the staining with that of CD4, CD3 and dsDNA to look for colocalizations.
4. When choosing final titers, take into account the tissue that you will be working on. Many immune markers will require high titers in tonsil simply because there are a ton of immune cells there. Some of these will require lower titers in other tissues, because there are less real targets and the excess antibody just binds non-specifically.

**5.2 MIBItileOneMarkerAcrossPoints.m**

This script plots a tiled image comparing a single marker between points/titers/tissues. It is useful for summarizing data and getting back to it after a while. *It should not replace rigorous examination as detailed in section 5.1*. To facilitate comparisons, the script will cap all images of a certain marker to the same value. This value can be given on a per-channel basis by adding a ‘Cap’ column to the panel csv file. If a channel-specific cap is not provided, the script will use the default value defined in the script parameters.

Parameters:

* corePath - Path of points to work on. Several can be specified by separating with commas
* massFile – File name of panel csv file. Can include a ‘Cap’ column, with numerical capping values for the channels (5 should be adequate for most channels. Particularly strong channels like dsDNA may require higher caps).
* xTileNum – Number of rows in the tile
* yTileNum – Number of columns in the tile
* outDir – Output directory for the tiled images
* defaultCap – Default capping value for plotting. Used only if no channel-specific value is mentioned in the csv file. A good number for most antigens is 5. Set to lower to see dynamic range of low-abundant antigens and higher for high-abundant antigens.
* xSize – X-Size of the largest image to be tiled. If, for example you’re tiling images of 1024x1024 and 512x512, then this should be 1024. Increase this number (e.g. to 1030) To create space between the images.
* ySize – Y-Size of the largest image to be tiled. If, for example you’re tiling images of 1024x1024 and 512x512, then this should be 1024. Increase this number (e.g. to 1030) To create space between the images.

Output:

* Tiled tiff files generated in the output folder. Warning - these are heavy!

Important:

* Use these images for a birds-eye view of the data. Don’t decide titers based on tiled images. To decide titers plot images in full screen as described above.