# Handbook for R package ‘processing Add On’

## Table of Contents


1 Introduction ..... 2
2 Needed packages ..... 2
2.1 ‘Cardinal‘ ..... 2
2.2 ‘ggplot2‘, ‘plotly‘ and ‘shiny‘ ..... 2
3 Functions explained ..... 3
3.1 .chunks ..... 3
3.2 combine_msa ..... 3
3.3 find_mz_matches ..... 4
3.4 .interpret ..... 4
3.5 path_convert ..... 5
3.6 pic ..... 5
3.7 pixel_filter ..... 6
3.8 pixel_filter_nb ..... 6
3.9 Pixel2Plot ..... 7
3.10 PixelViewer ..... 8
3.11 remove_noise ..... 9
3.12 scale ..... 10
3.12.1 scale_image ..... 11
3.12.2 scale_all ..... 11
3.13 select_matrix ..... 12
3.14 spectraViewer ..... 12
3.15 subtract_matrix ..... 13
4 Specific terms ..... 14

# 1 Introduction

This package is intended to be used for mass spectrometry imaging (MSI) data that has been converted to an ImzML file format. It has been developed to be used as an Add On for the Cardinal package. It adds a noise reduction method, that lets the user pick a noise threshold which is to be used similarly to 'peakPicking' methods in Cardinal. There are filters that can be used to remove non-significant features determined by their occurrence and a matrix reduction function that removes matrix molecules. Additionally, there are further tools that make handling the data easier like a mass finder that searches the experiment for specific masses.

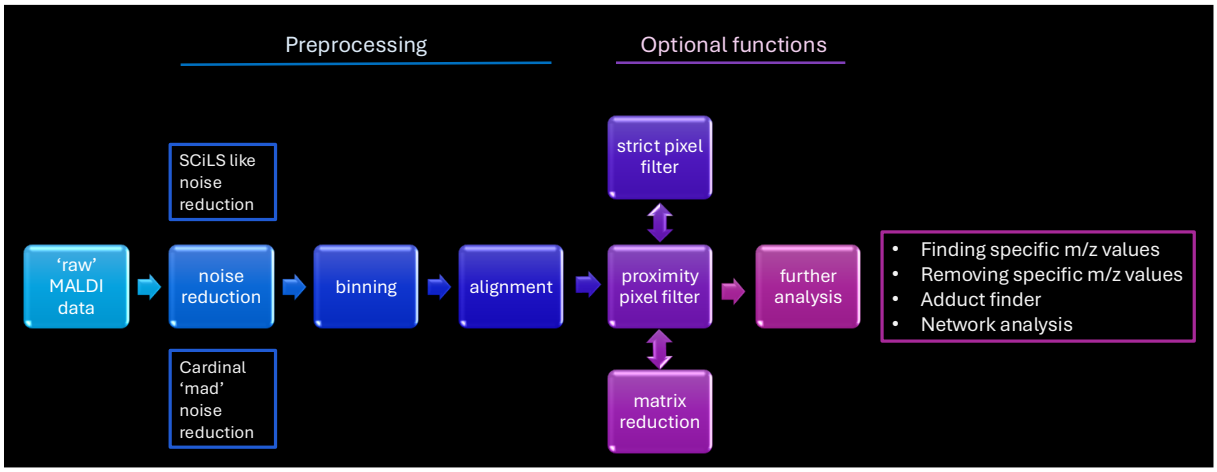


**Figure 1: Overview of workflows that can be achieved with this Add On.**

# 2 Needed packages

Before using the functions of the package, certain other packages need to be installed:

## 2.1 'Cardinal'

Since this package is intended to be an Add On for the Cardinal package, every function relies on the structure of MSI data that is created by Cardinal processing.

Use the following commands to install Cardinal and load its library.

```
install.packages('Cardinal')
library(Cardinal)
```

## 2.2 'ggplot2', 'plotly' and 'shiny'

These packages are only needed when using the functions '`SpectraViewer()`', '`PixelViewer()`' and '`Pixel2Plot()`'. The functions are used to create interactive images and plots of MSI data.

Use the following commands to install Cardinal and load its library.

```
install.packages(c('ggplot2', 'plotly', 'shiny', 'Cardinal'))
library(shiny)
library(ggplot2)
library(plotly)
```

# 3 Functions explained

## 3.1 .chunks

`.chunks(experiment, num)`

This is a helper function that divides a large MSI Experiment into a defined number of chunks. The return is a list with the length of '`num`' and each position holds a subset of the experiment. The downsizing works by dividing the experiment on mass range. For an experiment with a mass range from 400-600 and using two chunks for example , the first chunk could hold masses from 400-500 and the second chunk masses from 500.01 to 600. This works by firstly creating an integer list with the integer indicating which chunk this position in the experiment feature list belongs to. Second is defining start and end indices for each chunk. This is done by finding the lowest and the highest position for each integer indicating the chunk. Afterwards the Cardinal function '`subsetFeatures`' is used to subset the feature number efficiently.

The function takes the following arguments: `experiment` and `num`:

- `experiment`: MSI Experiment data. This type of file is created by Cardinal functions after Alignment.
- `num`: This argument takes an integer. It defines how many chunks are created.

## 3.2 combine_msa

`combine_msa(experiment, method= c('mean','median'), round_by = 3L)`

This function creates a data frame that contains a m/z list and an intensity list from the MSI Array. These lists are created by first combining all features and intensities from all spectra into one list each. These lists are then sorted by the order of the m/z values leading to inclining m/z values and their respective intensities sorted in the same order of m/z values. All m/z values are rounded to the decimal set by the argument `round_by`. For the third decimal the tolerance for a mass range of 400 to 1000 is from around 2.5 to 0.5 parts per million tolerances. After rounding to the third decimal, repeats of the same mass values are combined to only one occurrence and their respective intensity values are combined by mean or median depending on the selection of the method used. The shortened m/z list and intensity list are saved into a data frame.

The function takes the following arguments: `experiment`, `method` and `round_by`.

- `experiment`: MSI Array data. ImzML files can be converted to a MSI Array by using the Cardinal function `readMSIData(file_path_to_msa)`.
- `method`: choose between the methods mean or median to combine intensities of matching m/z values.
- `round_by`: this argument is set to the default 3L. The m/z values of the experiment will be rounded to this decimal.

### 3.3 find_mz_matches

`find_mz_matches(mz_library, mz_find, tolerance)`

This function can be used to find m/z values within a certain tolerance in another list of m/z values. This function works by calculating a list of upper and a list of lower limits of each m/z value in mz_library within the given tolerance. Both lists are the same length as mz_library. The given tolerance will be added and removed from each m/z value so a tolerance of 5 ppm will create an interval of 10 ppm with the m/z value in the middle. Afterwards the m/z values in mz_find will be assigned a start and an end index. The start index is the index of the last value that is smaller than the m/z value to be found in the lower limit list plus one. The end index is the index of the last value that is smaller than or equal to the m/z value to be found in the upper limit list (Figure 2). The difference between the end and the start index of each m/z value that has a match within the given tolerance will be one. For no matches it will be zero and for more than one match it will be the number of matches.

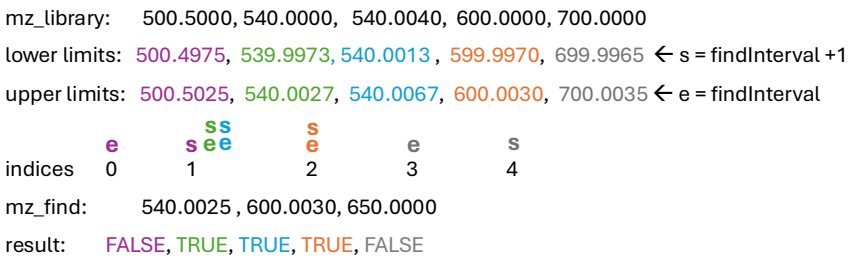


**Figure 2: Shown is an example of how the findInterval function works.** Start index is shortened to s and end index to e. Each value pair within the lower and upper limits lists get matched against the mz_find list. The Interval for the start index is the highest value that is smaller than the mz_find value. The end index is the highest value that is smaller than or equal to the mz_find value. When the end index is higher than or equal to the start index a match is found.

The return will be a logical vector holding a TRUE for every mz_library value with a match and a FALSE for every mz_library value that doesn't match with mz_find. This logical vector can be used on the m/z list creating a list that only holds the matching m/z values. When combined with which(find_mz_matches(…)) the result will be a list of the indices of the matched m/z values.

Option: When changing the return from matches to `ocs` (short for occurrences) the result will be a list that shows how many matches were found for each m/z value.

The function takes the following arguments: `mz_library`, `mz_find` and `tolerance`.

- `mz_library`: This argument takes a m/z list. This list is the one, m/z from the other list will be matched to. The result of this function will be a logical vector with the length of this list.
- `mz_find`: This argument takes a m/z list or a single m/z value. This list is the one, m/z from the other list will be matched to.
- `tolerance`: The tolerance is in unit ppm. It will be removed and added to a m/z value.

### 3.4 .interpret

`.interpret (experiment, mz=NULL, i=NULL, tolerance = 5L)`

This function is a helper function used for handling the input of m/z values, index or “TIC” as an argument. For imaging functions, it is useful to be able to handle either input. Since different rows or columns of the MSI data need to be accessed by the different inputs this function will concentrate the input into one type of output. When the input is an index (`i= integer()`) this function will check whether the index of the pixel exists in the selected data or is out of range. After checking the index will be returned.
If the input is a m/z value this function will match the m/z value with a m/z value in the experiment within a default tolerance of 5 ppm. This tolerance can also be changed in the parent functions that use this helper function. After matching the index of the matched m/z value will be returned.
If the input is “TIC” this function returns `i=FALSE` since there is no index for the spectra. The parent function will check for `is.FALSE(i)`. If this sentiment is `TRUE`, the TIC will be selected for images.
The function takes the following arguments: `experiment`, `mz`, `i` and `tolerance`.

- `experiment`: MSI Experiment data. This type of file is created by Cardinal functions after Alignment.
- `mz`: This argument expects a float number. Select which m/z value is imaged by putting ‘`mz=`‘ in front of the number. Ignore this argument when selecting an integer for `i`. It can also take ‘TIC’ as input.
- `i`: This argument expects an integer. Select which position of the m/z list is imaged by putting ‘`i=`‘ in front of the number. Ignore this argument when selecting an integer for `mz`. It can also take ‘TIC’ as input.
- `tolerance`: If no other value is given, the default value is 5. Select the tolerance for matching the selected m/z value to the m/z values from the experiment. The unit for tolerance is ppm.

### 3.5 path_convert

`path_convert(file_path)`

Path_convert is a function that changes all back slashes to slashes inside a path. When copying a path in Windows the directories will be separated by ‘\’ instead of '/'. R cannot work with ‘\’ since it is part of regular expressions and will be understood as such by the program.
To use path_convert, the following form of string for the directory path needs to be used as an argument:

- `r'(C:\Users\MALDI_Data\perfect-experiment.IBD)'`
  ‘r’ stands for ‘raw’ and signals to R that this string is not interpreted as a regular expression. Inside the brackets the path to a file or a directory can be pasted.

### 3.6 pic

`pic(experiment)`

Pic is a function one can use when the TIC is supposed to be imaged. Pic will open a new graphic device an image the TIC of a given experiment in the style ‘dark’.
The function takes the following arguments: `experiment`.

- `experiment`: MSI Experiment data. This type of file is created by Cardinal functions after Alignment.

### 3.7 pixel_filter

```
pixel_filter(experiment, min_count_pix)
```

This function will remove all features of an experiment that occur in less or equal pixels given by the min_count_pix argument. The first part of this function is to reduce the data from the experiment to make the function more efficient. By aligning the data with Cardinal, the pData will show a column named count. This column holds the number of pixels the corresponding feature occurs above a certain intensity threshold. This threshold is not documented in Cardinal documentation. Since some features may occur in more pixels than the count suggests due to the intensity threshold it is not sufficient to remove pixels by, but tested data amount can be reduced since features that have a higher count will not appear in less features than the count suggests. This is why the first step is to reduce the data frame by only holding features that have a count lower than or equal to the `min_count_pix` number. To not change the data in the experiment a new MSI Experiment is defined as `mse_count`. To further reduce the number of spectra, spectra in `mse_count` where all intensity values are at zero get removed from `mse_count`. For this smaller data frame, the function `rowSums(intensity>0)` gets called. This will translate intensity over zero values to TRUE and the others to FALSE. The function `rowSums` will count all TRUE values. If the row sum of a feature is equal to or lower than `min_count_pix` the feature will be removed from `mse_count`. Afterwards features that occur in `mse_count` and the original experiment are deleted from the original experiment since these features didn’t reach the pixel threshold.
The function takes the following arguments: `experiment` and `min_count_pix`.

- `experiment`: MSI Experiment data. This type of file is created by Cardinal functions after Alignment.
- `min_count_pix`: This argument takes an integer. It is a threshold. All features that are in the experiment after using the function are in more spectra than this number.

### 3.8 pixel_filter_nb

```
pixel_filter_nb(experiment, min_pixel, proxim_range, chunked = FALSE,
chunk_amount = 4L)
```

This function is used to remove features that occur in less or an equal amount of a set number. An additional condition is that the pixel that this feature occurs in are not within a defined proximity range. The helper function inside the function is used. To be able to use the algorithm that calculates the proximity of pixel under the ’`min_pixel`‘ limit, it is stored in an inner function

'FUN' that can be called for whole experiments or for parts of experiments created by chunks. 'FUN' works by first decreasing the data frame by removing all features that occur in more pixel than the defined threshold. Additionally, pixels that this subset of features don't appear in, are removed. Next a matrix is generated that holds all integers of pixel with an intensity value above 0 for each feature. For each row of the matrix, representing a feature, the distances between these pixels are calculated. To get the coordinates of the pixel, a data frame is created that holds each pair of coordinates for every pixel, the row of the data frame being the same as index of the pixel in the spectra data. If any two pixels are within the proximity range of one another, calculations are stopped and this feature won't be removed from the experiment. The return of the function is a vector with TRUE values for features that occur in more pixel than the defined threshold or features that occur in pixels in proximity to each other and FALSE for the others. To determine whether to call this inner function for the whole experiment or for subsets of the experiment, there are some variables. The first option to create chunks is within the head of the function. When the object 'chunked' is TRUE, the experiment will be divided into the number of chunks defined by the object 'chunk_amount'. If 'chunked' is FALSE, the number of features will be multiplied by the number of spectra. To make the calculation work both numbers are divided by $1 \cdot 10^4$. If the product is smaller than 45, the experiment will not be parted into chunks. The upper limit 45 was decided empirically. In case of errors for this function, set 'chunked' to TRUE manually or set 'chunk_amount' to a higher integer.
The function takes the following arguments: `experiment`, `min_pixel`, `proxim_range`, `chunked` and `chunk_amount`.

- `experiment`: MSI Experiment data. This type of file is created by Cardinal functions after Alignment.
- `min_pixel`: An integer specifying the minimum pixel count for a feature. Features occupying fewer pixels than this threshold are removed from the experiment in case of no neighboring pixels.
- `proxim_range`: Takes a numeric, with a minimum of 2. Defines the distance classifying pixels as neighbors.
- `chunked`: Decides whether experiment will be separated into chunks. The default is FALSE.
- `chunk_amount`: Takes integer. Defines the number of chunks the experiment should be separated into. The default is 4.

### 3.9 Pixel2Plot

```
Pixel2Plot(experiment, mz=NULL, i=NULL, tolerance=5L)
```

This function images an experiment and by selecting a pixel with a click of the mouse and pressing 'p' on the keyboard, plots the selected pixel underneath the image. First the function calls the .interpret function to determine whether a m/z value (then converted to the associated

index), an index or the TIC was given. Then a data frame is created that holds all x-coordinates and y-coordinates of the experiment and the intensities of the interpreted index or TIC. The m/z value and index or the TIC are used as a plot label. To make the plot interactive the shiny package is used. The function `shinyApp()` that creates the interface takes two arguments: `ui` and `server`. In `ui` the information on the layout for outputs is defined and how events are called. The `server` defines those events and outputs. Values that are to be used for events and outputs need to be saved in shiny as a reactive value which are defined first in the server function and then can be used and changed. The heatmap of the intensities is created with the package ggplot2. The color gradient is set to recreate the Cardinal images. To have the pixel with coordinates (0/0) at the top left of the image, the y-axis was reversed. That is also why y-values are multiplied by (-1) to achieve positive y-values. Using 'plotly' the hovering and the clicking functions of the mouse are initiated. When hovering over the image the coordinates of the pixel that the mouse hovers over will be shown in a text box below the image. When the mouse is clicked, the values are saved to the shiny environment and shown below. The next event happens when pressing 'p'. When pressing p, a new data frame will be created in the shiny environment. This data frame holds a list of all m/z values from the experiment and the intensity list of the selected pixel. The intensities are then plotted against the m/z values and shown underneath the text box. The label of the plot shows the integer of the pixel that was selected. To exit the function, press 'Enter'.

The function takes the following arguments: `experiment`, `mz`, `i` and `tolerance`.

- `experiment`: MSI Experiment data. This type of file is created by Cardinal functions after Alignment.
- `mz`: This argument expects a float number. Select which m/z value is imaged by putting '`mz=`' in front of the number. Ignore this argument when selecting an integer for `i`. It can also take 'TIC' as input.
- `i`: This argument expects an integer. Select which position of the m/z list is imaged by putting '`i=`' in front of the number. Ignore this argument when selecting an integer for `mz`. It can also take 'TIC' as input.
- `tolerance`: If no other value is given, the default value is 5. Select the tolerance for matching the selected m/z value to the m/z values from the experiment. The unit for tolerance is ppm.

### 3.10 PixelViewer

```
PixelViewer(experiment, mz=NULL, i=NULL, tolerance= 5L)
```

Similar to Pixel2Plot this function creates an image of an experiment and by selecting a pixel with a click of the mouse, it will create a data frame of the coordinates of selected pixels. By pressing 'z' the pixel selected last can be removed. First the function calls the .interpret function to determine whether a m/z value (then converted to the associated index), an index or the

TIC was given. Then a data frame is created that holds all x-coordinates and y-coordinates of the experiment and the intensities of the interpreted index or TIC. The m/z value and index or the TIC are used as a plot label. To make the plot interactive the shiny package is used. The function `shinyApp()` that creates the interface takes two arguments: `ui` and `server`. In `ui` the information on the layout for outputs is defined and how events are called. The `server` defines those events and outputs. Values that are to be used for events and outputs need to be saved in shiny as a reactive value which are defined first in the server function and then can be used and changed. The heatmap of the intensities is created with the package ggplot2. The color gradient is set to recreate the Cardinal images. To have the pixel with coordinates (0/0) at the top left of the image, the y-axis was reversed. That is also why y-values are multiplied by (-1) to achieve positive y-values. Using 'plotly' the hovering and clicking functions of the mouse are initiated. When hovering over the image the coordinates of the pixel that the mouse hovers over will be shown in a text box below the image. When the mouse is clicked, the values are saved to the data frame and shown below. To save the data frame with saved pixels, press 'Enter'. The data frame is saved in the R environment as 'roi_pixel_df'. The information of this data frame like the column with indices of the pixel can be used to create a subset of features into another experiment to further analyze by principle component analysis (PCA) for example. The function takes the following arguments: `experiment`, `mz`, `i` and `tolerance`.

- `experiment`: MSI Experiment data. This type of file is created by Cardinal functions after Alignment.
- `mz`: This argument expects a numeric. Select which m/z value is imaged by putting '`mz=`' in front of the number. Ignore this argument when selecting an integer for `i`.
- `i`: This argument expects an integer. Select which position of the m/z list is imaged by putting '`i=`' in front of the number. Ignore this argument when selecting an integer for `mz`.
- `tolerance`: If no other value is given, the default value is 5. Select the tolerance for matching the selected m/z value to the m/z values from the experiment. The unit for tolerance is ppm.

### 3.11 remove_noise

```
remove_noise(experiment, noise)
```

This function is used to remove intensities and their corresponding m/z value from a MSI Array when intensities don't reach the given noise threshold. First it filters m/z values on whether the corresponding intensities meet the threshold and then it filters intensities on the same condition. The new filtered lists are saved where previous intensity and m/z lists were saved. The function takes the following arguments: `experiment` and `noise`.

- `experiment`: MSI Array data. ImzML files can be converted to a MSI Array by using the Cardinal function `readMSIData(file_path_to_msa)`.

- `noise`: This argument takes a numeric

### 3.12 scale

```
scale <- function(experiment, mz=NULL, i=NULL, method = c('log', '75%', 'quantile'), tolerance = 5)
```

This function is used to change the values of the intensities of one feature to help visualize them. It uses the function .interpret to interpret a given m/z value, index or TIC to access the list of intensities There are three methods that can be used for rescaling: ‘log’, ‘75%’ or ‘quantile’. After scaling the intensity list of the chosen m/z value, index or ‘TIC’, the intensities will be changed in the spectra of the experiment. All other intensities stay the same. This way this rescaling function can be used for functions like ‘Pixel2Plot’.

1. Method ‘log’:

This method logarithmizes the intensity values. Using the `log1p` function adds one to each intensity value. This ensures zero values remain zero after logarithmization and avoids numerical errors. For the magnitude of intensity values, this should result in only minor difference compared to standard logarithmization. This will lower differences between the values so that values will get closer in color on the scale.

2. Method ‘%‘:

This method calculates a percentile of the maximal intensity and reduces every intensity value that is above that percentile of the maximal intensity to the calculated value. The percentile can be chosen individually by giving the function the argument ‘`percent= X`’. This lowers values of high intensities and could be used when specific pixel with high intensity values distorts the image.

3. Method ‘quantile‘:

This method identifies the top 25 % values of all non-zero intensities and sets them to the lowest value within that percentile. In comparison to the ‘%’ method, ‘quantile’ is more robust against outliers with extremely high intensity values.

The function takes the following arguments: `experiment`, `mz`, `i`, `method`, `percentile` and `tolerance`.

- `experiment`: MSI Experiment data. This type of file is created by Cardinal functions after Alignment.
- `mz`: This argument expects a numeric. Select which m/z value is imaged by putting ‘`mz=`‘ in front of the number. Ignore this argument when selecting an integer for `i`.
- `i`: This argument expects an integer. Select which position of the m/z list is imaged by putting ‘`i=`‘ in front of the number. Ignore this argument when selecting an integer for `mz`.
- `method`: Choose between the three methods ‘log’, ‘%” or ‘quantile’.
- `percent`: Takes a numeric. When selecting ‘%’, this argument takes the percentage.

- `tolerance`: If no other value is given, the default value is 5. Select the tolerance for matching the selected m/z value to the m/z values from the experiment. The unit for tolerance is ppm.

The limiting operations in this function cannot be executed in chunks and therefore is unsuitable for large experiments.

### *3.12.1 scale_image*

This function works similarly to `scale`, but unlike scale, the resulting experiment will only contain the specified feature and the corresponding scaled intensity list. This approach is faster than scale but makes it incompatible with functions like `Pixel2Plot`.

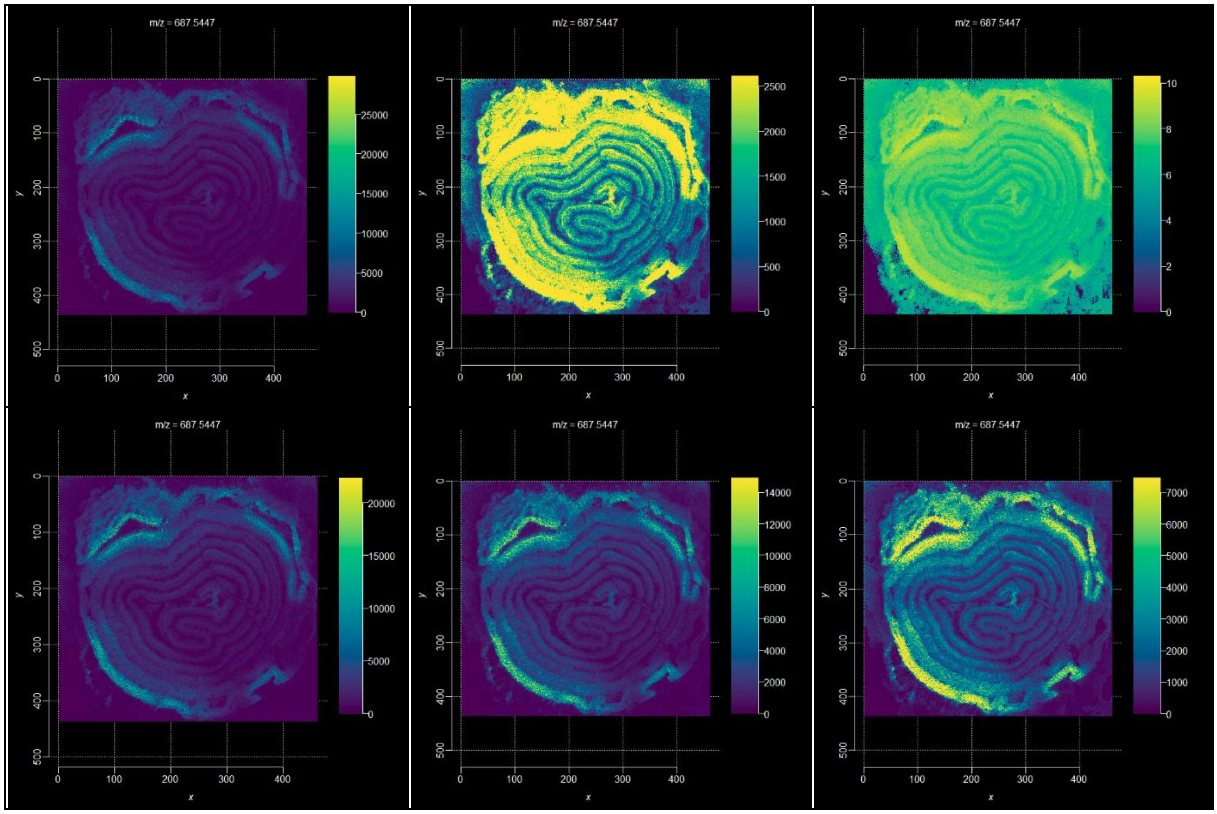


**Figure 3: Shown are six MALDI images of a mouse intestine depicting a lipid with a mass to charge ratio of 687.5447 colored according to the intensity in each pixel.** Intensity scaling was performed using the `scale_image` function. The top left panel shows the unmodified image. The center of the top row shows effects of the method 'quantile' and the top right panel shows the effects of the 'log' method. The bottom row depicts effects of the '%' method with decreasing thresholds from left to right: 75 %, 50 %, 25 %.

### *3.12.2 scale_all*

This function uses `.row.scale` as a helper function to use scale methods on the whole data set. Therefore, this function works like scale but produces an experiment where every feature was scaled row wise. This function will take longer to process and has the same limitations as `scale`.

The function takes the following arguments: `experiment`, `mz`, `i`, `method`, `percentile` and `tolerance`.

- `experiment`: MSI Experiment data. This type of file is created by Cardinal functions after Alignment.
- `method`: Choose between the three methods ‘log’, ‘%’ or ‘quantile’.
- `percent`: Takes a numeric. When selecting ‘%’, this argument takes the percentage.
- `tolerance`: If no other value is given, the default value is 5. Select the tolerance for matching the selected m/z value to the m/z values from the experiment. The unit for tolerance is ppm.

### 3.13 select_matrix

```
select_matrix(experiment, corner_size_tl, corner_size_tr, corner_size_bl, corner_size_br)
```

This function works best for round samples with excess matrix in the corners of the MALDI image. This function can be used to create a logic vector that contains TRUE values for corners of the image. When combined with the Cardinal function `subsetPixels` an object can be created that only contains matrix molecules and spectra. This object can further be used to analyze matrix molecules or subtract matrix molecules from the experiment using the function `subtract_matrix`. Depending on the position of the sample, the corner sizes are to be set individually. The corners calculated by the function are isosceles right-angled triangles for which the isosceles sides are set by the parameter `corner_size`. A starting value for corner_size can be a sixth of the width of the image. To verify whether there is sample inside the selected matrix region an image can be done of the selected region.

The function takes the following arguments: `experiment`, `corner_size_tl`, `corner_size_tr`, `corner_size_bl` and `corner_size_br`.

- `experiment`: MSI Experiment data. This type of file is created by Cardinal functions after Alignment.
- `corner_size_tl`: This argument takes a numeric. The number is the length of the legs of the top left isosceles right-angled triangle.
- `corner_size_tr`: This argument takes a numeric. The number is the length of the legs of the top right isosceles right-angled triangle.
- `corner_size_bl`: This argument takes a numeric. The number is the length of the legs of the bottom left isosceles right-angled triangle.
- `corner_size_br`: This argument takes a numeric. The number is the length of the legs of the bottom right isosceles right-angled triangle.

### 3.14 spectraViewer

```
spectraViewer(experiment, mz=NULL, i=NULL, xcol = "mz", ycol = "intensity")
```

This function can be used to plot data frames or MSI Experiments by intensity over m/z value. Therefore, `spectraViewer` can plot MSI Array data after the function `combine_msa` converted

it into a data frame. In the plot a noise threshold can be selected by clicking the mouse and saved into the R environment for further usage.
First the data frame for plotting is created within the function if the given experiment is a MSI Experiment and not a data frame. Otherwise just label names are created and the given data frame will be used. It is important that the data frame has column names that match the column names determined by 'xcol' and 'ycol'.
The function `shinyApp()` that creates the interface takes two arguments: `ui` and `server`. In `ui` the information on the layout for outputs is defined and how events are called. The `server` defines those events and outputs. The line plot of the intensities is created with the package ggplot2. Using the package 'plotly' the violet line is created that shows the noise threshold while hovering over the plot. Next the outputs are created. When hovering over the plot, a text box is shown that contains the current y-axis value of the mouse. When the mouse is not above the plot, it will say to move the cursor over the plot. When the mouse is clicked, a text will appear that shows the noise threshold that was last chosen. This value will be saved in the R environment as the object 'noise_threshhold' and overwritten if the mouse is clicked again at a different threshold. To continue in R the window needs to be closed by pressing 'Enter' or closing the window.
The function takes the following arguments: `experiment`, `mz`, `i` and `tolerance`.

- `experiment`: MSI Experiment data or data frame. This is the data that is plotted.
- `i`: This argument expects an integer or a column name of the fData like 'mean'. Select which pixel is plotted putting '`i=`' in front of the number.
- `tolerance`: If no other value is given, the default value is 5. Select the tolerance for matching the selected m/z value to the m/z values from the experiment. The unit for tolerance is ppm.

### 3.15 subtract_matrix

`subtract_matrix(experiment, mse_sample, mse_matrix)`

This function removes matrix molecules from the experiment. These molecules are determined by the ratio of means between the matrix and the sample. If the ratio of matrix means to sample means per feature is above one, the molecule is deemed a matrix molecule and is removed (Equation 1).

$$for\ each\ \frac{m}{z}\ value\colon\ \frac{mean\ intenstiy\ matrix}{mean\ intensity\ sample} \qquad \textbf{Equation 1}$$

To use the function, a sample region and a matrix region must be selected. For this purpose, functions like `selectROI(experiment, 'TIC')` can be used and specifically for the matrix, the function `select_matrix` can be used. Both functions give back a logic vector. This can be used with `subsetPixels(experiment, logic_vector)` as the logic vector. Since the mean is needed for this function the Cardinal function `summarizeFeatures(experiment,`

`stat='mean')` must be called beforehand for the matrix and sample experiments. The output of this function will be an MSI Experiment.

The function takes the following arguments: `experiment`, `mse_sample` and `mse_matrix`.

- `experiment`: MSI Experiment data. This type of file is created by Cardinal functions after Alignment. This is the main experiment with both matrix regions and sample regions from which the matrix molecules are subtracted.
- `mse_sample`: MSI Experiment data. This experiment should only contain sample regions of the experiment and needs to be summarized to have column 'mean' in the featureData
- `mse_matrix`: MSI Experiment data. This experiment should only contain matrix regions of the experiment and needs to be summarized to have column 'mean' in the featureData

# 4 Specific terms

| | |
|---|---|
| MSI Array | This is a Cardinal file format of unprocessed MSI data. The m/z values and intensities are saved in a list of lists with each list corresponding to a spectrum. |
| MSI Experiment | This is a Cardinal file format of aligned MSI data. There is one list of all m/z values occurring in the experiment data. For spectra where there was no intensity measured for certain features their intensity value is zero. |
| Feature | Feature describes a molecule in a spectrum but for these data frames describes specific m/z values. |
| m/z values | m/z values are the mass over charge ratio that is measured by the mass spectrometer. |
| Pixel | A pixel has a specific x- and y-coordinate where the laser was shot at on the sample resulting in a spectrum of said pixel. |
| Spectrum | A spectrum is a graph for one pixel that shows the m/z values on the x-axis and the corresponding intensities on the y-axis. |