Cameron Shedlock ([cameron.shedlock@scranton.edu](mailto:cameron.shedlock@scranton.edu))

Dr. Kate Stumpo ([katherine.stumpo@bruker.com](mailto:katherine.stumpo@bruker.com))

Table of Contents

[Tutorial Objective 1](#_Toc82702058)

[Why use R for MSI 1](#_Toc82702059)

[Start using R 1](#_Toc82702060)

[Prepare packages and data 2](#_Toc82702061)

[MSI Data Processing 2](#_Toc82702062)

[Data Preprocessing 2](#_Toc82702063)

[Target Confirmation 3](#_Toc82702064)

[Making m/z images 5](#_Toc82702065)

[Selecting a desired ROI 7](#_Toc82702066)

[Unsupervised data exploration 9](#_Toc82702067)

[Analysis of SSC 9](#_Toc82702068)

[Principal Component Analysis 10](#_Toc82702069)

[Analysis of PCA 11](#_Toc82702070)

## Tutorial Objective

Mass spectrometry imaging (MSI) has emerged as a rapidly expanding field in the MS community. The analysis of large spectral data is further complicated by spatial information in MSI. A plethora of resources exist for expert users to begin parsing MSI data in R, but there is a critical lack of guidance for absolute beginners. This tutorial is designed to serve as a one stop guide to start using R with MSI data.

## Why use R for MSI

A host of powerful R packages have been created to enable efficient and relevant analysis of MSI data in R. This tutorial will guide users through a simple step by step workflow to allow MSI data to be analyzed in R without requiring a background in R or MSI. R packages have been optimized to load complex MSI data with a few simple commands. R has a host of packages for MSI which include powerful analytical methods accessible to beginners.

## Start using R

Depending on your familiarity with R, the following resources are designed to guide beginners through the user interface to start using this notebook.

<https://education.rstudio.com/learn/beginner/> <https://rstudio-education.github.io/hopr/starting.html>

With R and R Studio installed, the user interface can be quickly understood through freely available videos and guides such as the resources provided by RStudio itself.

## Prepare packages and data

Use your own or download a MSI dataset from github data section or metaspace (<https://metaspace2020.eu>) Briefly, metaspace is a free online collection of MSI datasets available to download. At the following this raw dataset can be downloaded from metaspace. #insert link here

# install packages  
if (!requireNamespace("BiocManager", quietly = TRUE))  
 install.packages("BiocManager")  
BiocManager::install("Cardinal")

# load MSI package Cardinal  
library(Cardinal)

With the Cardinal package installed and an imaging dataset downloaded, we are ready to begin.

## MSI Data Processing

The defining feature of MSI is the spatial component of the data. In order to make m/z images, data first needs to be read into R and processed.

# use your terminal or OS to find the file path of the .imzml file of the MSI data from the downloaded github files   
data\_path\_example <- "the/path/to/your/.imzml/imaging/data"

# make sure the imzml and ibd file are in the same folder. Use the file path for the .imzml file. It ends with .imzml in the path. Download the dataset from metaspace https://www.metaspace2020.eu/datasets?subm=e89830bd-ce18-11ea-96db-8faddc24c30c&organism=Danio%20rerio%20%28zebrafish%29 or use the preprocessed .rds file provided in the github repository and skip past creating it.  
data\_path <- "insert the path to the imzml downloaded from metaspace here"

Important note: depending on the OS of the workstation, different symbols are needed for file paths. In Windows the  symbol is the default for outputting file paths, but R will read this as its own command. This is why a second  is added manually. In UNIX based OS like Mac or Linux the native / file paths work

# use Cardinal function to easily load the complicated MSI data structure into R with only 2 lines of code  
data <- readMSIData(data\_path)

Now the object **data** is your MSI dataset loaded into R for further analysis. Many packages and data parsing methods can be used through R to make discoveries from the data.

## Data Preprocessing

Raw data can be analyzed but preprocessing is the standard for MSI experiments. This step reduces the computational resources and time required for analysis and can be tuned for specific needs. The parameters such as total ion count (tic) and root mean square (rms) are listed in Cardinal documentation and user changable. The most important parameter is the signal to noise ratio (SNR). This number dictates how much more intense a peak must be than the noise region to be considered a real peak. By increasing this number a smaller number of more intense peaks will be used for analysis. If a particular peak is desired, it is useful to make an m/z image and display its spectrum before and after preprocessing to verify this step retains the target.

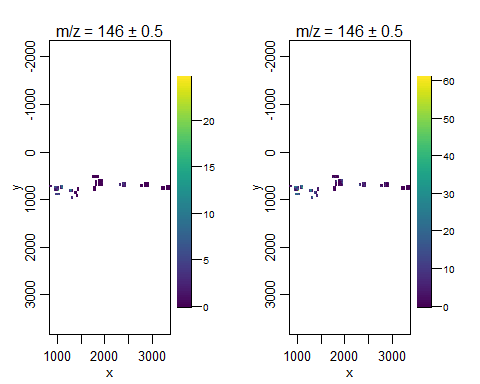
# preprocess datasets  
data\_proc <- data %>% # %>% this is a pipe operator. It performs the following steps on "data" and saves the resulting object as "data\_proc"

# note if you are skipping preprocessing and using the premade R object from github, you can run this code to load it into R. data\_proc <- readRDS("path\\to\\downloaded\\file\\from\\github\\.rds”)  
   
 #normalize the spectra  
 normalize(method="tic") %>%  
 #select main peaks, those below SNR will be zero  
 peakPick(method="mad", SNR=5) %>%   
 #align spectra to given peaks or mean spectra if left empty  
 peakAlign(tolerance=0.5, units="mz") %>%   
 #remove low frequency peaks  
 peakFilter(freq.min=0.01, rm.zero=TRUE) %>%  
 #process  
 process()

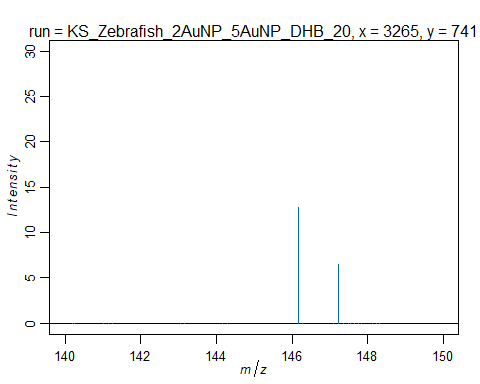
## nrows changed from 26700 to 1630

### Target Confirmation

target <- 146  
# verify target remains after preprocessing  
# create image of target before preprocessing  
before <- image(data, mz=target, plusminus=0.5)  
# create image of target after preprocessing  
after <- image(data\_proc, mz=target, plusminus=0.5)  
# display before  
print(before, layout=c(1,2))  
# display after  
print(after, layout=FALSE)



# plotting the spectrum from a random (here number 5) pixel of the processed data. Adjust x and y to contain an appropriate intensity for y and the m/z value in the x range  
plot(data\_proc, pixel=11000, xlim=c(140,150), ylim=c(0,30))

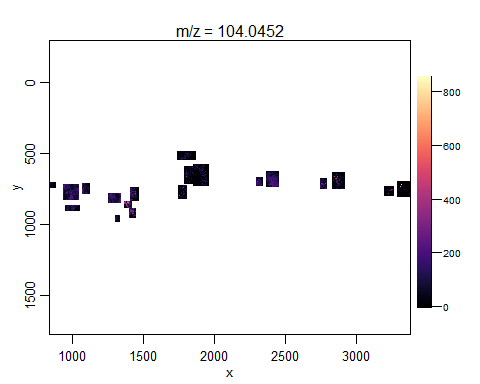


# note the pixel is chosen to have a corresponding (x,y) coordinate on the sample. This is derived from the image.

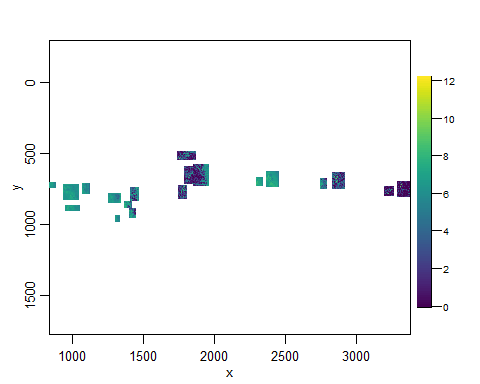
## Making m/z images

The defining feature of MSI is the m/z image. By simply changing the value of the **mz** in the code, a new image with a new target can be created.

image(data\_proc, mz=104, colorscale=magma)



# intensity for all pixels in run  
image(data\_proc)



This image is signal for the entire processed dataset. By changing the parameters in the code specific m/z values can be used along with changing the color theme. Cardinal offers dark and light themes along with cividis, magma, inferno, and plasma for colorscales. An m/z image of the whole dataset is useful for quickly visualizing signal hotspots and trends without further analysis. Furthermore, the image generated can be used to estimate coordinates for particular areas of the slide.

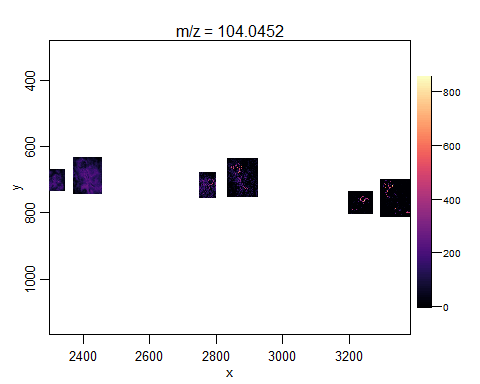
## Selecting a desired ROI

This code enables the user to chose coordinates from the previous m/z image of the entire imaging run. By using estimated coordinates a select region can be turned into an R object. This is useful for targeted analysis of specific sections and having a smaller object for later processing.

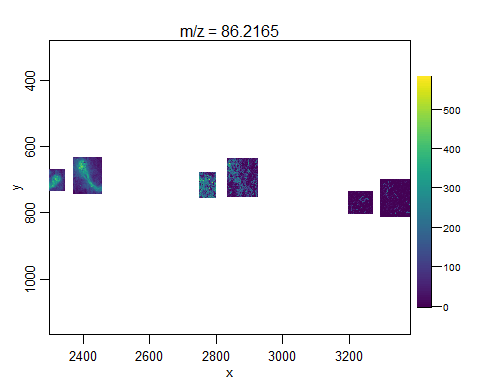
# cut an imaging run into a smaller object  
# save features of data  
features <- features(data\_proc)  
# select desired pixel range to form square arount region of interest  
pixels <- pixels(data\_proc,   
 x>= 2250 & x<=3500,  
 y>= 625 & y<=850)  
# make object with features of selected area  
data\_cut <- data\_proc[features,pixels]

Now when the smaller chunk is projected as an image it only includes the desired region.

# image of selected area  
image(data\_cut,  
 mz=104,  
 colorscale=magma)



# image any m/z value  
# simply change the mz= aparameter to target  
image(data\_cut, mz=86.21196)



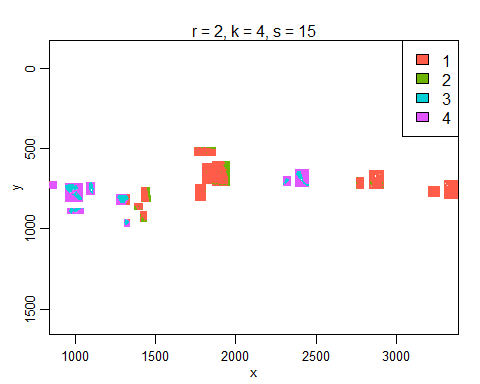
## Unsupervised data exploration

Data dimension reducing techniques such as principal component analysis and spatial shrunken centroids can be used to parse MSI datasets without the need for targeted user inputs.

# ssc  
data\_proc\_ssc <- spatialShrunkenCentroids(data\_proc, method="adaptive", r=2, s=15, k=4)  
# save created object  
saveRDS(data\_proc\_ssc, file="data\_proc\_ssc.rds")

In the spatial shrunken centroid function there are four user changable parameters, which are method, r, s, and k. K is the number of segments and what is most often changed by the user. R is the shrinkage parameter and s alters the way peaks are chosen.

# image results of ssc  
ssc <- image(data\_proc\_ssc,   
 model=list(s=15))  
ssc



### Analysis of SSC

In this dataset the 4 classes chosen show underlying trends in the data. Visual analysis reveals the far left DHB section in purple and light blue appears to spread into the first 5nm AuNP region. This was confirmed by slide analysis revealing accidental overspraying. Beyond visual interpretation, numerical analysis is possible. Using the topFeatures() function the top 10 m/z values used to determine each class are shown. The class can be changed in the class parameter.

# determine top m/z features  
topFeatures(data\_proc\_ssc, model=list(s=15), class == 2)

## Top-ranked features:   
## mz count freq r k s class centers statistic  
## 1 228.0971 35567 0.27916268 2 4 15 2 86.62037 437.8864  
## 2 118.2341 72695 0.57057752 2 4 15 2 171.45237 428.6921  
## 3 172.1460 41099 0.32258292 2 4 15 2 72.23712 425.4727  
## 4 439.0271 10917 0.08568670 2 4 15 2 25.23939 378.2302  
## 5 368.2398 11222 0.08808062 2 4 15 2 22.65367 358.7807  
## 6 250.0754 61146 0.47993030 2 4 15 2 142.40629 354.9686  
## 7 369.2370 28702 0.22527981 2 4 15 2 28.76257 336.7534  
## 8 212.1292 74945 0.58823760 2 4 15 2 347.34154 318.4535  
## 9 213.1380 62999 0.49447436 2 4 15 2 87.23428 312.8753  
## 10 134.2361 7555 0.05929862 2 4 15 2 18.38128 255.8710

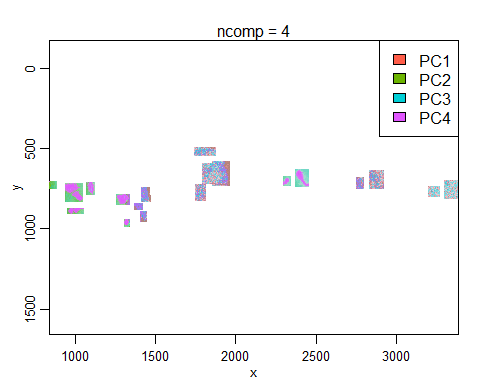
## Principal Component Analysis

Dimension reduction by orthogonal transformation is accomplished in PCA which has emerged as a common approach for unsupervised exploration of MSI data.

image(dhb52\_processed\_all\_pca\_n8, contrast.enhance=“histogram”, normalize.image=“linear”)

# PCA  
# ncomp parameter is number of principal components  
data\_proc\_pca <- PCA(data\_proc, ncomp=4)  
# save object due to longer run time  
saveRDS(data\_proc\_pca, file="data\_proc\_pca.rds")

# image results of PCA  
image(data\_proc\_pca, contrast.enhance="histogram", normalize.image="linear")



### Analysis of PCA

The unsupervised method picked up on accidental overspray of DHB matrix into 5nm AuNP zone. All methods can be performed on a selected area of imaging data.