Case study 1: mouse kidney peptide dataset

Yinyue Zhu

Table of contents

1 Overview

In this case study, we analyze a MALDI-TIMS-TOF tryptic peptide dataset of mouse kidney tissue. First we processed the data in TIMSImaging, then performed segmentation in Cardinal, in addition we compared processing workflow with/without ion mobility. The dataset is from Melanie Föll Lab and available at MassIVE (MSV000096373).
Li, M., Meyer, L., Meier, N., Witte, J., Maldacker, M., Seredynska, A., Schueler, J., Schilling, O. and Föll, M. (2025), Spatial Proteomics by Parallel Accumulation-Serial Fragmentation Supported MALDI MS/MS Imaging: A First Glance Into Multiplexed and Spatial Peptide Identification. Rapid Commun Mass Spectrom, 39: e10006. https://doi.org/10.1002/rcm.10006

2 Data processing in TIMSImaging

2.1 Introcution

In this part, we start from raw .d data, explore the dataset, perform peak processing, show separation by ion mobility and export the results in the open imzML format.

```
import timsimaging

# enable visualization in the Jupyter notebook
from bokeh.io import show, output_notebook
output_notebook()
# disable FutureWarning
import warnings
warnings.filterwarnings('ignore')
```

```
Unable to display output for mime type(s): text/html
```

```
Unable to display output for mime type(s): application/javascript, application/vnd.bokehjs_l
```

2.1.1 Load MALDI-TIMS-TOF raw data

Suppose we know nothing about this dataset. First we load the dataset and print basic information like pixel number and data range.

```
bruker_d_folder_name = r"D:\dataset\Kidney_MS1_IT06.d"
dataset = timsimaging.spectrum.MSIDataset(bruker_d_folder_name)
dataset
```

```
MSIDataset with 38267 pixels
        mz range: 799.998-2500.007
        mobility range: 1.200-2.600
```

2.1.2 View summary image and tables

Then we can view the TIC image without any processing, as the TIC intensities are stored in the `Frames` table in the raw tdf file. Hover on individual pixels to see the details like coordinates and intensity:

```
dataset.image()
```

```
Unable to display output for mime type(s): text/html
```

```
Unable to display output for mime type(s): application/javascript, application/vnd.bokehjs_ex
```

We can view the underlying data as tables, too

```
# pixel positions
dataset.pos
```

Frame	XIndexPos	YIndexPos
1	336	175
2	337	175
3	338	175
4	339	175
5	340	175
...	...	...
38263	737	116
38264	738	116
38265	739	116
38266	740	116
38267	741	116

```
# metadata of pixels
dataset.data.frames
```

	Id	Time	ScanMode	MsMsType	TimsId	MaxIntensity	SummedIntensities
0	0	0.000000	20	0	64	0	0
1	1	2.671587	20	0	64	688	5346970
2	2	2.941060	20	0	210641	637	4998845

	Id	Time	ScanMode	MsMsType	TimsId	MaxIntensity	SummedIntensities
3	3	3.210677	20	0	407604	605	4793642
4	4	3.480285	20	0	595229	626	4705085
...	...	...	...	...	...	...	...
38263	38263	10602.755832	20	0	11938087511	925	4587513
38264	38264	10603.025466	20	0	11938264725	720	4511870
38265	38265	10603.295060	20	0	11938440038	775	4700548
38266	38266	10603.564667	20	0	11938622040	626	4648396
38267	38267	10603.813145	20	0	11938802286	755	4574022

2.1.3 Set region of interest(ROI)

The left region is the kidney tissue, we can use the `set_ROI` method to set the ROI by `xmin`, `xmax`, `ymin`, `ymax` parameters.

```
dataset.set_ROI(name='kidney', xmax=500)
```

2.1.4 Process selected ROI

Then we run the processing pipline introduced in the basic use notebook.

```
results = dataset.process(sampling_ratio=0.1, frequency_threshold=0.02, tolerance=3, adaptive
```

```
Computing mean spectrum...
Traversing graph...
Finding local maxima...
Summarizing...
```

2.1.5 View peak list

We can view the interactive peak list:

```
peaklist = results["peak_list"]
table, _ = timsimaging.plotting.feature_list(results["peak_list"])
show(table)
```

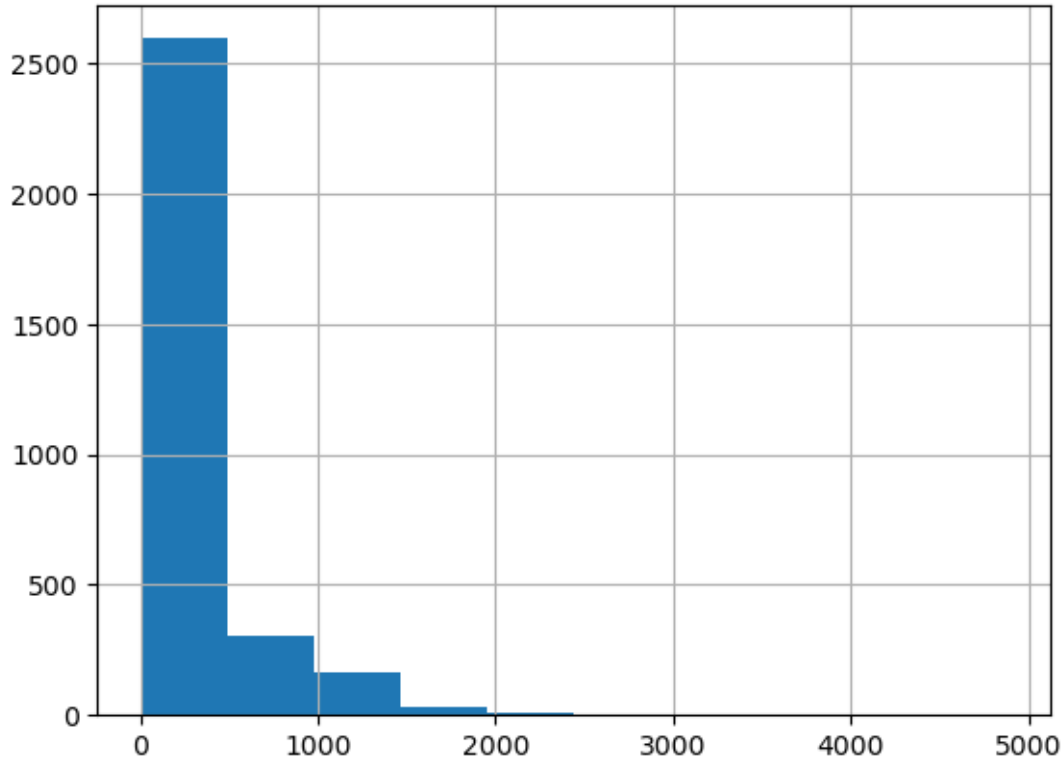
```
Unable to display output for mime type(s): text/html
```

```
Unable to display output for mime type(s): application/javascript, application/vnd.bokehjs_ex
```

2.1.6 View intensity distribution

We can also visualize the intensity distribution:

```
results["peak_list"]["total_intensity"].hist()
```



2.1.7 View Interactive Results

User can explore the results in the GUI. By clicking pixels in the image(top left), users can view its mass spectrum(bottom middle), mobilogram(top right) and 2D spectogram combining these two(top middle). Also, users can click entries in the peak list(bottom left) to investigate ion images and spectral details of that peak. The menu(bottom right) provide summarizing functions like TIC image and mean spectrum, as well as file export options. This requires a living Python session to render, here shows a screenshot:

```
show(results['viz'])
```

```
Unable to display output for mime type(s): application/vnd.bokehjs_exec.v0+json, text/html
```

2.1.8 Separate ions by ion mobility

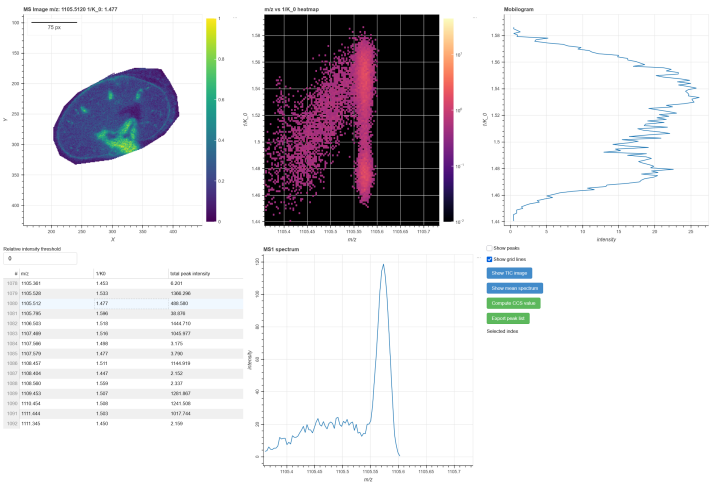
Isobaric ions, or ions with non-distinguishable m/z(for example, isomers) could be separated by ion mobility. As examples shown below, TIMSImaging detected both peaks.

```
from bokeh.layouts import row
```

```
isomer1, _  = timsimaging.plotting.image(dataset, i=1079, results=results)
isomer2, _  = timsimaging.plotting.image(dataset, i=1080, results=results)
show(row(isomer1, isomer2))
```

```
Unable to display output for mime type(s): text/html
```

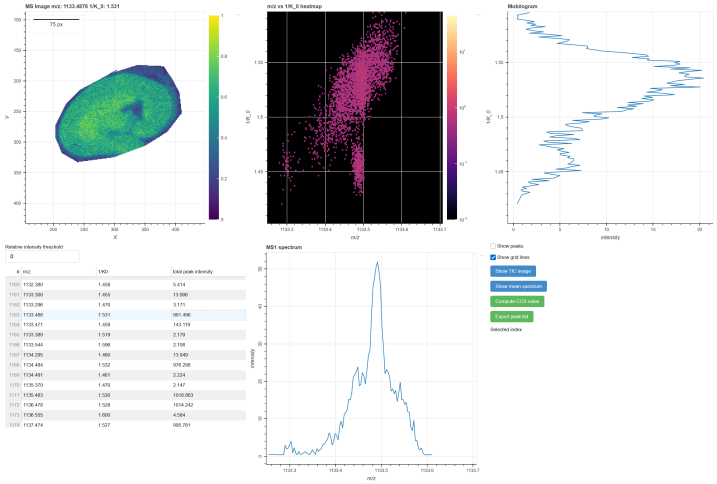
```
Unable to display output for mime type(s): application/javascript, application/vnd.bokehjs_ex
```



```
isomer1, _  = timsimaging.plotting.image(dataset, i=1163, results=results)
isomer2, _  = timsimaging.plotting.image(dataset, i=1164, results=results)
show(row(isomer1, isomer2))
```

```
Unable to display output for mime type(s): text/html
```

```
Unable to display output for mime type(s): application/javascript, application/vnd.bokehjs_ex
```



2.1.9 Export the processed data in imzML format

Finally, we export the processed data as continuous, centroid imzML file for downstream analysis in Cardinal.

```
timsimaging.spectrum.export_imzML(dataset, path=r"D:/dataset/Kidney_MS1_ITO6", peaks=results)
```

3 Downstream segmentation in Cardinal

3.1 Introduction

In this part, we start from processed imzML data in Section 2, first show how ion mobility is stored, then perform spatial shrunken centorids to show that it is compatible with Cardinal, and further illustrate benefits of ion moblity by single ion image segmentation.

3.1.1 Cardinal setup

First make sure Cardinal is installed. If not, run the code below in a R session:

```
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("Cardinal")
```

Then load Cardinal:

```
library("Cardinal")
```

3.1.2 Load processed dataset

The raw .d dataset was processed by TIMSImaging and exported as continuous, centroided imzML format, that each pixel corresponds to a spectrum with 3 dimensions: mz, mobility and intensity; all spectra share the same mz and mobility domain.
Here we use `extraArrays` to load ion mobility information, and `MS:1003006` is the entry number for ion mobility.

```
msa <- readMSIData("D:/dataset/Kidney_MS1_ITO6.imzML", as="MSImagingArrays", extraArrays=c(m
msa
```

```
MSImagingArrays with 22978 spectra
spectraData(3): intensity, mz, mobility
pixelData(3): x, y, run
coord(2): x = 204...409, y = 175...332
runNames(1): Kidney_MS1_ITO6
experimentData(5): spectrumType, spectrumRepresentation, lineScanSequence, scanType, lineScan
centroided: TRUE
continuous: TRUE
```

Check that we can access ion mobility now. The mobiity values are in $1/K_0$:

```
spectraData(msa)$mobility
```

```
<22978 length> matter_list :: out-of-core list
                  [1]      [2]      [3]      [4]      [5]      [6] ...
$spectrum=1 1.262920 1.287529 1.265770 1.290552 1.311340 1.210486 ...
                  [1]      [2]      [3]      [4]      [5]      [6] ...
$spectrum=2 1.262920 1.287529 1.265770 1.290552 1.311340 1.210486 ...
                  [1]      [2]      [3]      [4]      [5]      [6] ...
$spectrum=3 1.262920 1.287529 1.265770 1.290552 1.311340 1.210486 ...
                  [1]      [2]      [3]      [4]      [5]      [6] ...
$spectrum=4 1.262920 1.287529 1.265770 1.290552 1.311340 1.210486 ...
                  [1]      [2]      [3]      [4]      [5]      [6] ...
$spectrum=5 1.262920 1.287529 1.265770 1.290552 1.311340 1.210486 ...
                  [1]      [2]      [3]      [4]      [5]      [6] ...
$spectrum=6 1.262920 1.287529 1.265770 1.290552 1.311340 1.210486 ...
...
(1.95 MB real | 0 bytes shared | 571.7 MB virtual)
```

3.1.3 Coerce to MSImagingExperiment

In order to do downstream analysis, we need to convert the MSImagingArray object to MSImagingExperiment. Unfortunately, currently MSImagingExperiment does not support extra arrays and the mobility array would be discarded. However, we can do *spatial analysis* like segmentation anyway as the features are from ion mobility-awere peak picking(after conversion, isobaric ions are still distinct.)

```
mse <- convertMSImagingArrays2Experiment(msa)
```

3.1.4 Spatial Shruken Centroids Segmentation

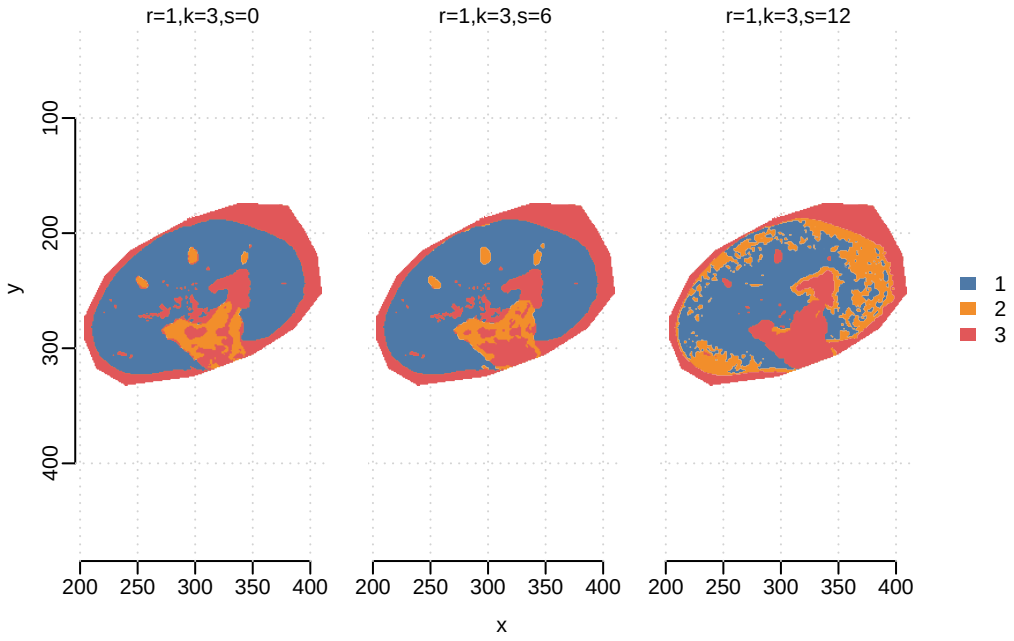
Here we use multivariate spatial shrunken centroids(SSC) to cluster pixels into segments. We use a fixed segment number(k) with different sparsity(s). The larger s is, the more unimportant features would be ignored and not contribute to segmentation.

```
set.seed(42, kind="L'Ecuyer-CMRG")
ssc <- spatialShrunkenCentroids(mse, r=1, k=3, s=c(0, 6, 12))
ssc
```

```
ResultsList of length 3
names(3): r=1,k=3,s=0 r=1,k=3,s=6 r=1,k=3,s=12
model: SpatialShrunkenCentroids
              r k  s  weights clusters sparsity      AIC       BIC
r=1,k=3,s=0   1 3  0 gaussian        3     0.00 24880.18 124926.30
r=1,k=3,s=6   1 3  6 gaussian        3     0.40 17379.63  87259.11
r=1,k=3,s=12  1 3 12 gaussian        3     0.63 13264.84  66006.19
```

We can visualize the segmentation results and compare with the results from vendor's SCiLS Lab software, which uses bisecting K-means. When s=6, SSC results in clear segments for Cortex(1), Medulla(2) and background(3).

```
image(ssc, i=1:3, type="probability", layout=c(1,3))
```



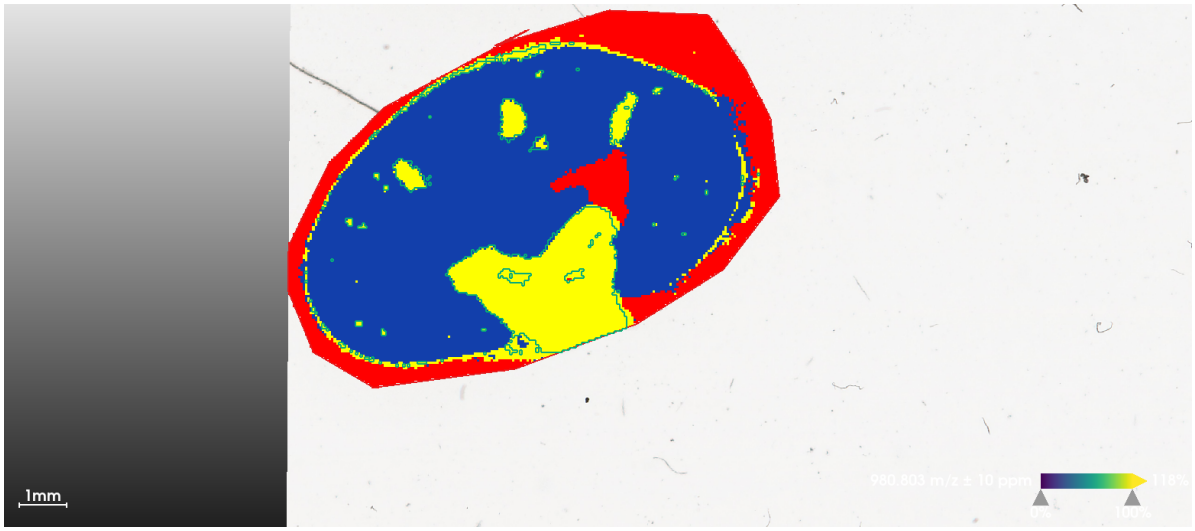
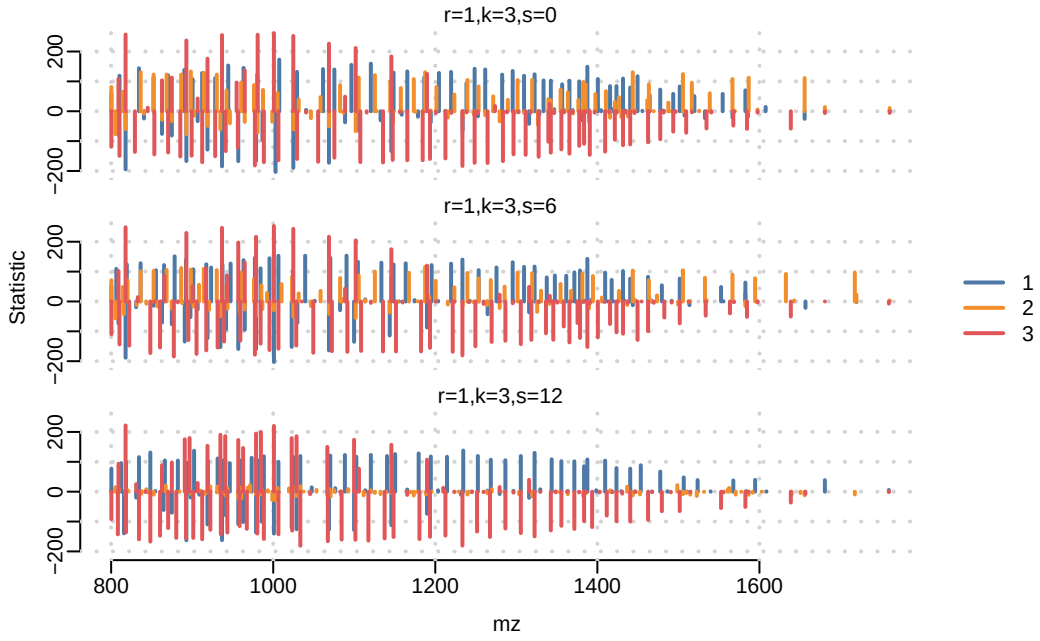


Figure 3.1: SCiLS result

Visualize statistic profile for each segment. Use the `plot` function to show which mass features are associated with which segments, higher statistics means the feature has more contribution to corresponding segment. Here we visualize top-100 features for each segment.

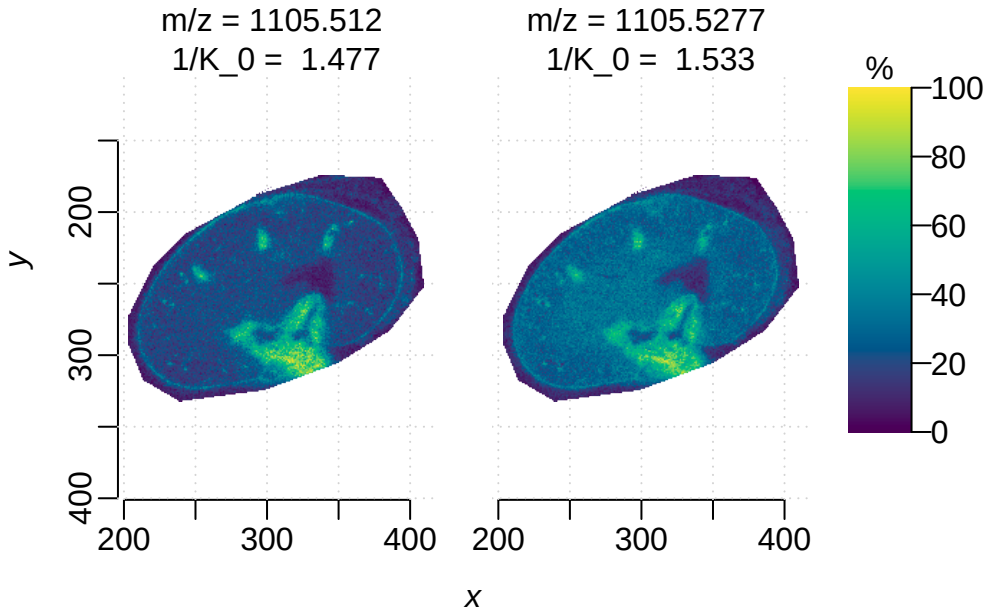
```
plot(ssc, i=1:3, n=100, linewidth=2, layout=c(3,1))
```



3.1.5 Univariate segmentation on isobaric features

Here we select two isobaric features, apply univariate Spatial Dirichlet Gaussian mixture models(DGMM) on them separately, and compare with segmentation that treats them as one feature.
First plot ion images separately. Note that two features have apexes at the same mass location, but here we use the weighted average m/z to distinguish them in the MSImagingExperiment object(see the heatmap in Section 1).

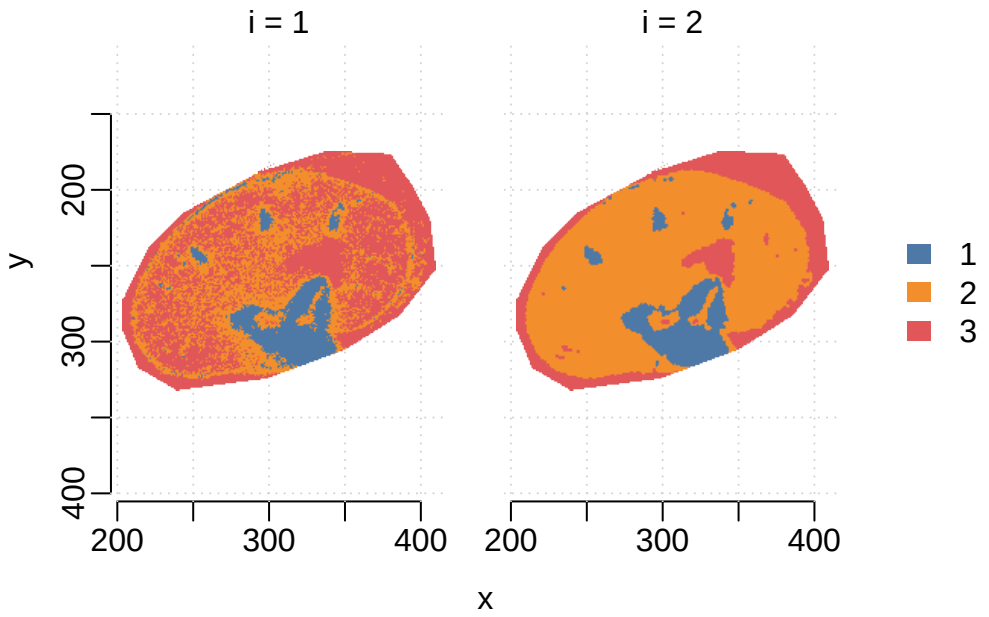
```
i <- findInterval(1105.5, mz(mse))

mobs <- spectraData(msa)$mobility[[1]][c(i+1, i+2)]
img <- image(mse, i=c(i+1, i+2), scale=TRUE)
# add ion mobility in plot titles
img$labels <- paste(img$labels, "\n1/K_0 = ", sprintf("%.3f", mobs))
plot(img)
```



DGMM on individual features. Though two features are at the same m/z, i=2($1/K_0$=1.533) has higher correlation with the sub-structures while i=1 is more uniformly distributed.

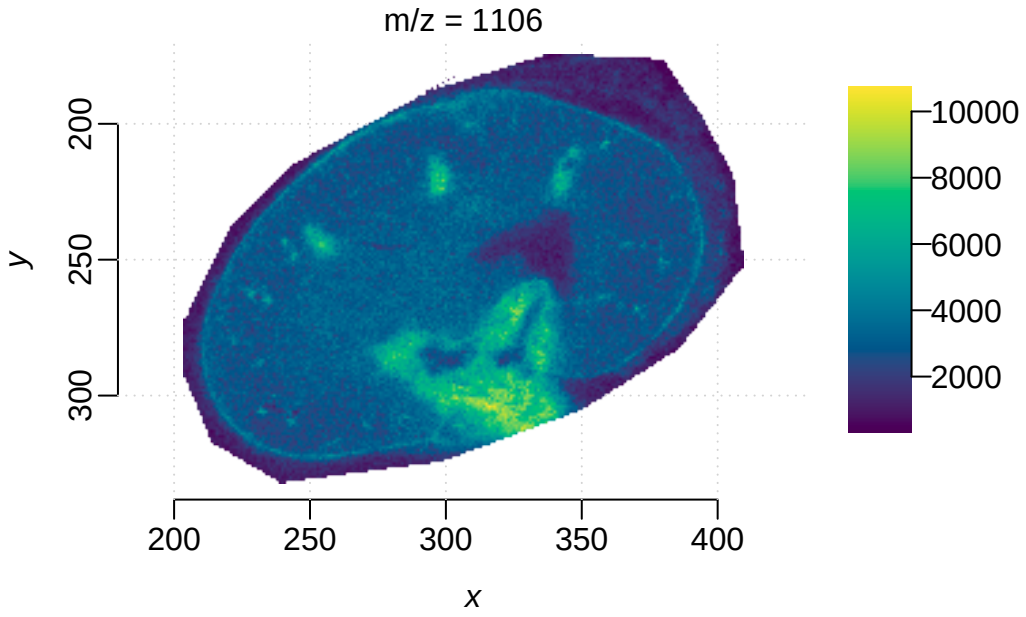
```
ions <- subsetFeatures(mse, i=c(i+1, i+2))

set.seed(42, kind="L'Ecuyer-CMRG")
```

```
dgmm <- spatialDGMM(ions, r=1, k=3, weights="gaussian")
image(dgmm, i=c(1,2), layout=c(1,2))
```



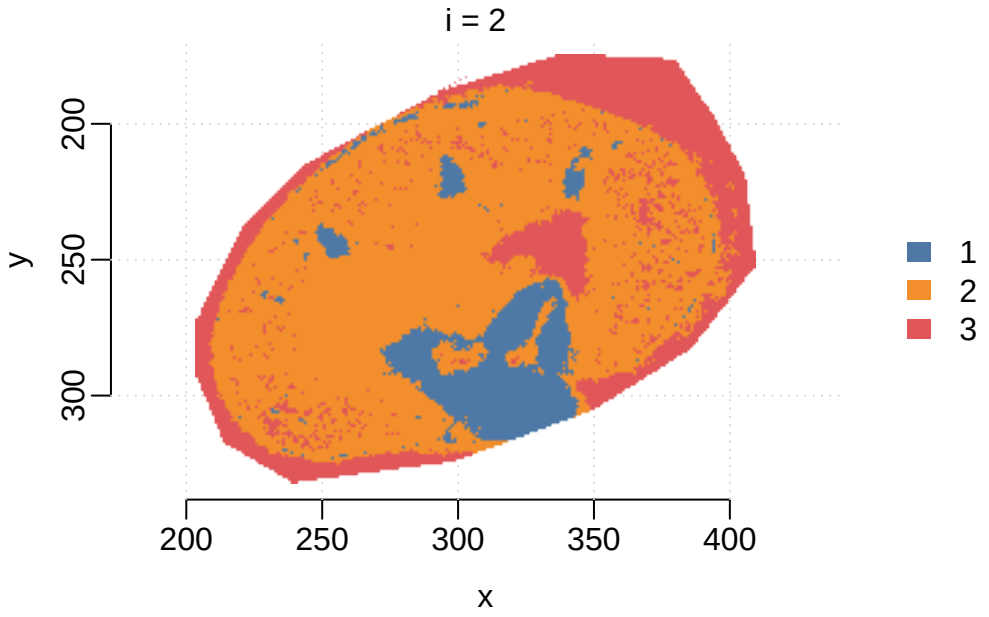
Plot the ion image collapsing two features(ignore separation by ion mobility). Here we use the `bin()` function in Cardinal which sums intensities within 1 Da bin, as there are no other feature in the mass interval.

```
summed <- bin(ions, resolution=1, units="mz")
image(summed, i=2)
```



DGMM on collapsed feature without ion mobility separation, the segmentation result is noisier than using feature(m/z = 1105.5277, $1/K_0$=1.533) only. DGMM segmentation is usually used to find features whose segments colocalize with certain ROIs, on ion moblity-resolved data could help screening informative features more accurately.

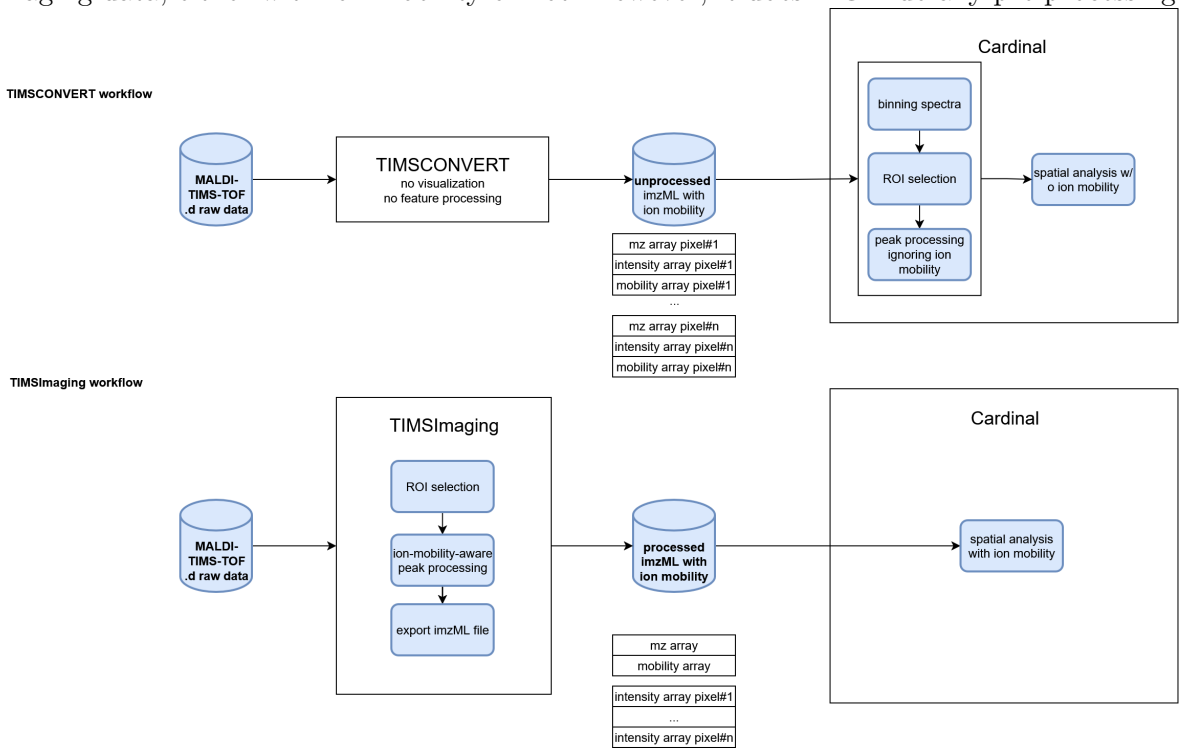
```
dgmm_summed <- spatialDGMM(summed, r=1, k=3, weights="gaussian")
image(dgmm_summed, i=2)
```


i = 2
y
200
250
300
x
200
250
300
350
400
1
2
3

4 TIMSCONVERT workflow comparison

4.1 Introduction

In this part, we compare TIMSImaging workflow with existing method that ignores ion mobility on the same mosue kidney peptide dataset. TIMSCONVERT is an open-source tool to convert general Bruker raw data into open formats, users could install from https://github.com/gtluu/timsconvert. Specifically, it outputs imzML for MALDI-TIMS-MS imaging data, either with ion mobility or not. However, it does **NOT** do any pre-processing.



4.1.1 Convert raw data using TIMSCONVERT

First we use TIMSCONVERT to get **unprocessed** imzML. Note that unprocessed imzML with ion mobility is usually huge, we recommmend to do conversion using HPC.

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --time=4:00:00
#SBATCH --job-name=Casestudy_timsconvert
#SBATCH --mem=80G
#SBATCH --partition=short
#SBATCH -o output_%j.txt                    # Standard output file
#SBATCH -e error_%j.txt                     # Standard error file
#SBATCH --mail-user=zhu.yiny@northeastern.edu  # Email
#SBATCH --mail-type=ALL                     # Type of email notifications
eval "$(conda shell.bash hook)"
conda activate timsconvert
cd /work/VitekLab/Data/MS/Melanie_manuscript/
timsconvert --input Kidney_MS1_ITO6.d --outdir mouse_kidney_timsconvert_output --compression
conda deactivate
```

4.1.2 Process data from TIMSCONVERT

Load the imzML data from TIMSCONVERT, which is obtained by running the shell script.

```
msa <- readMSIData("D:\\dataset\\Melanie_case_study\\Kidney_MS1_ITO6.imzML")
msa
```

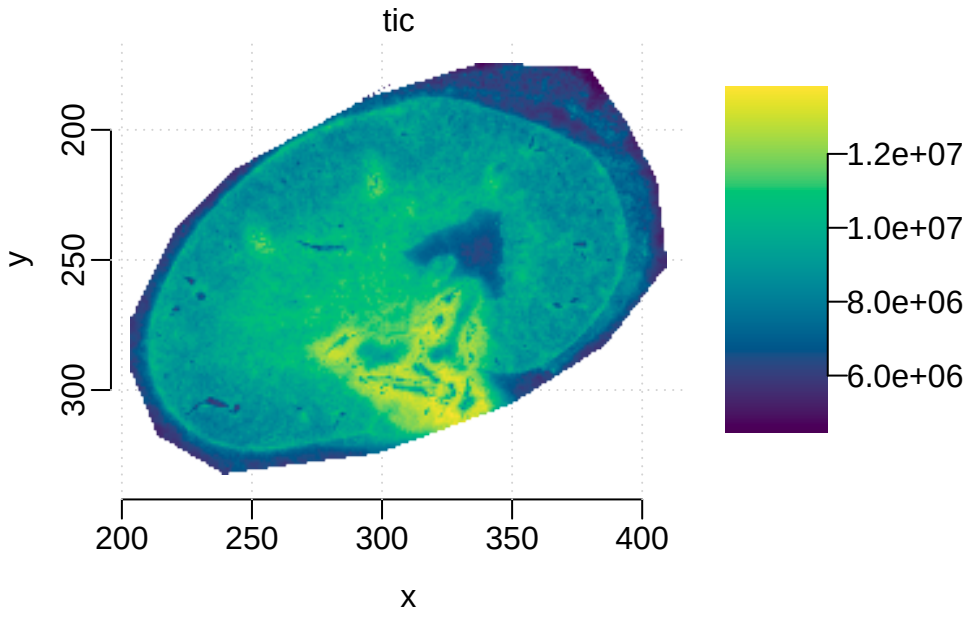
```
MSImagingArrays with 38267 spectra
spectraData(2): intensity, mz
pixelData(3): x, y, run
coord(2): x = 204...753, y = 99...442
runNames(1): Kidney_MS1_ITO6
experimentData(5): spectrumType, spectrumRepresentation, lineScanSequence, scanType, lineSca
centroided: TRUE
continuous: FALSE
```

TIMSCONVERT keeped ion mobility in the imzML data, however the peak processing functions in Cardinal cannot take advantage of it. Here we bin the spectra to project data points with the same m/z but different ion mobilities together, then do processing **without** ion mobility in Cardinal.

```
mse_binned <- bin(msa, resolution=20, units="ppm")
mse<-summarizePixels(mse_binned)
```

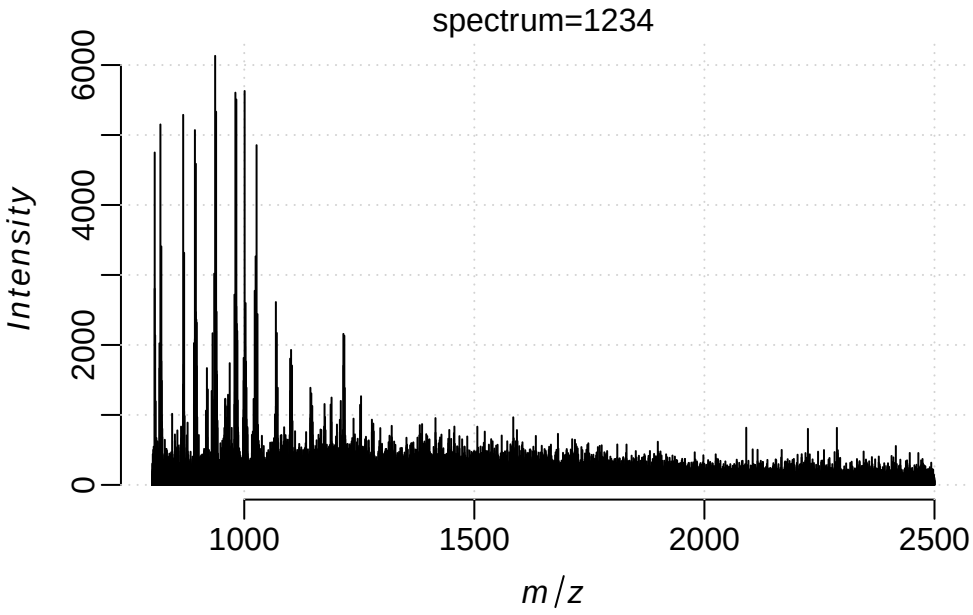
To reduce the computation, we only process the kidney tissue region in following steps.

```
kidney <- subsetPixels(mse, x<500)
image(kidney, 'tic')
```



Plot one spectrum, though a single spectrum is centroided on m/z dimension by the instrument, it looks more similar with a profile spectrum after projection.

```
plot(kidney, i=1234)
```



Treat the data as profile spectra and process it to reduce the number of features.

```
kidney <- subsetPixels(mse_binned, x<500)
centroided(kidney)<-FALSE
set.seed(1, kind="L'Ecuyer-CMRG")
#kidney <- normalize(kidney, method='rms')
peaks <- peakProcess(kidney, SNR=3, tolerance=200, units="ppm")
```

4.1.3 Comparing ion images with/without ion mobility

Then we load the processed data from TIMSImaging, which is already peak-picked.

```
peaks_timsimaging <- readMSIData("D:\\dataset\\Melanie_case_study\\mouse_kidney.imzML")
peaks_timsimaging
```

```
MSImagingExperiment with 3110 features and 22978 spectra
spectraData(1): intensity
featureData(1): mz
pixelData(3): x, y, run
coord(2): x = 204...409, y = 175...332
runNames(1): mouse_kidney
experimentData(5): spectrumType, spectrumRepresentation, lineScanSequence, scanType, lineScan
```

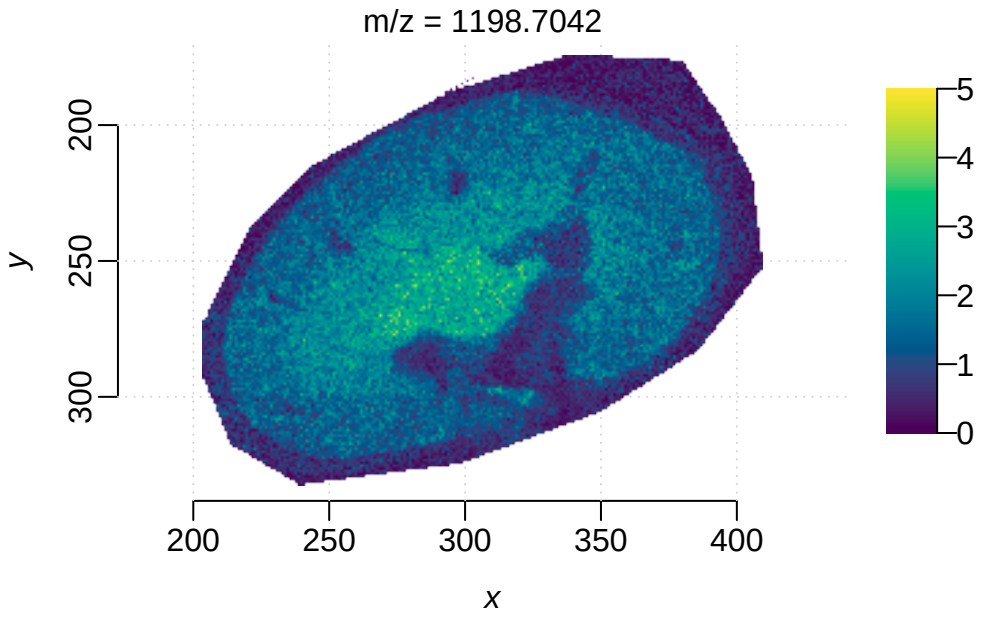
```
mass range:  800.2957 to 1760.7449
centroided: TRUE
```

Normalize on peak-picked data for fair comparison:

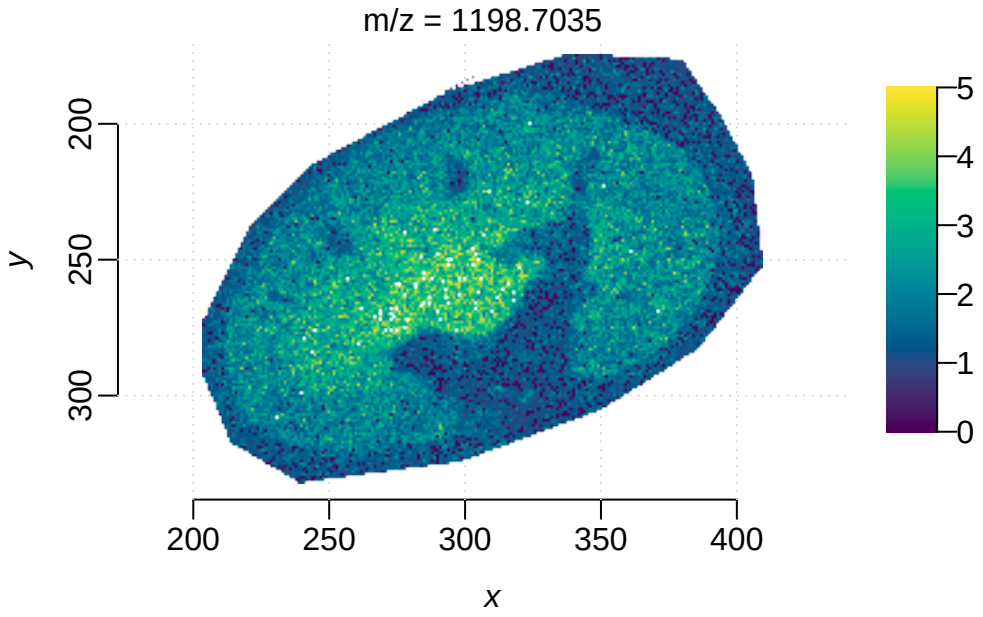
```
peaks_norm = process(normalize(peaks, method='tic'))
peaks_timsimaging <- process(normalize(peaks_timsimaging, method='tic'))
```

Plot ion images from processsing with/without ion mobility in the same color scale.

```
m <- 1198.7
image(peaks_timsimaging, i = findInterval(m, mz(peaks_timsimaging))+1, zlim=c(0, 5))
```

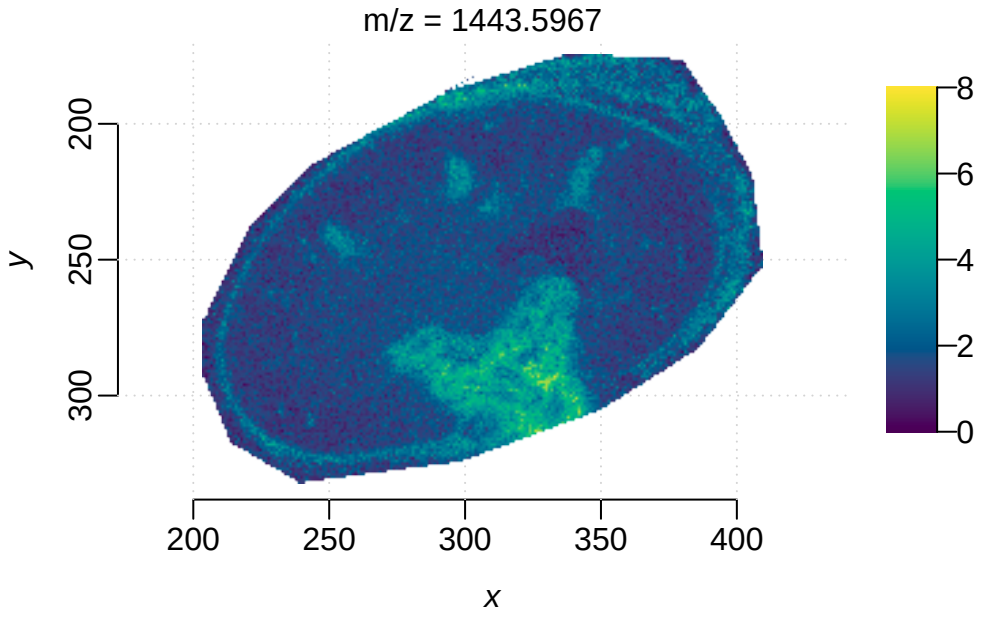



```
image(peaks_norm, mz = m, zlim=c(0, 5))
```



The ion images from TIMSCONVERT and processed in Cardinal looks noiser, as well as less contrast between the background and kidney edge.

```
m <- 1443.59
image(peaks_timsimaging, i = findInterval(m, mz(peaks_timsimaging))+1, zlim=c(0, 8))
```




```
image(peaks_norm, mz = m, zlim=c(0, 8))
```