Report on MSIWarp

Code

MSIWarp and Cardinal

Author: Jonathan Ruepong

Summary

MSIWarp provides a method for aligning spectra. MSIWarp runs on Python. Processing data are similar between Cardinal and MSIWarp. The difference starts before the actual alignment begins, where, after peak picking to provide reference spectra, the peaks picked in MSIWarp need to be sorted by intensity. This list of sorted peaks is given as a reference for MSIWarp.

MSIWarp features node placements. From my understanding, nodes are used to space out spectra by segments, so that the same amount of information is equal between the reference spectra and the current spectra being warped. This means that in one segment if a peak is detected that needs warping within a segment then that information will need to be the same within all spectra being aligned. The placement of these nodes can dictate the quality of warping. MSIWarp offers two different methods of node placement. Either use the same warping nodes for all spectra or uniquely place them for each spectra.

Note: This code was from the MSIWarp git repo

Hide

```
#Setting up nodes to be used on all spectra
node_mzs = [150, 450, 750, 1050]
node_deltas = [0.015, 0.045, 0.075, 0.105] # slacks = node_deltas * n_steps
n_steps = 25
nodes = mx.initialize_nodes(node_mzs, node_deltas, n_steps)

#Setting up nodes uniquely for each spectra
params = mx.params_uniform(mx.Instrument.Orbitrap, # each instrument type has its own relationsh
ip between peak width and m/z
                           n_steps, # same as above
                           n_peaks, # the number of peaks per segment
                           max_n_nodes, # maximum number of nodes
                           mz_begin, #
                           mz_end, #
                           slack # the slack relative to peak width
                           )
```

Code used

The following are code snippets used to see what MSIWarp can do compared to Cardinal. As of now, I was not able to get MSIWarp to work as intended. I only got it to work on one dataset that was on the MSIWarp paper.

Loading cardinal and creating test simulated data

Hide

```
library(Cardinal)
set.seed(2022)
ori <- simulateImage(preset=1, npeaks=10, from=500, to=600,sdmz=750, units="ppm")
```

Hide

```
ori
```

Exporting simulated data to use in MSIWarp

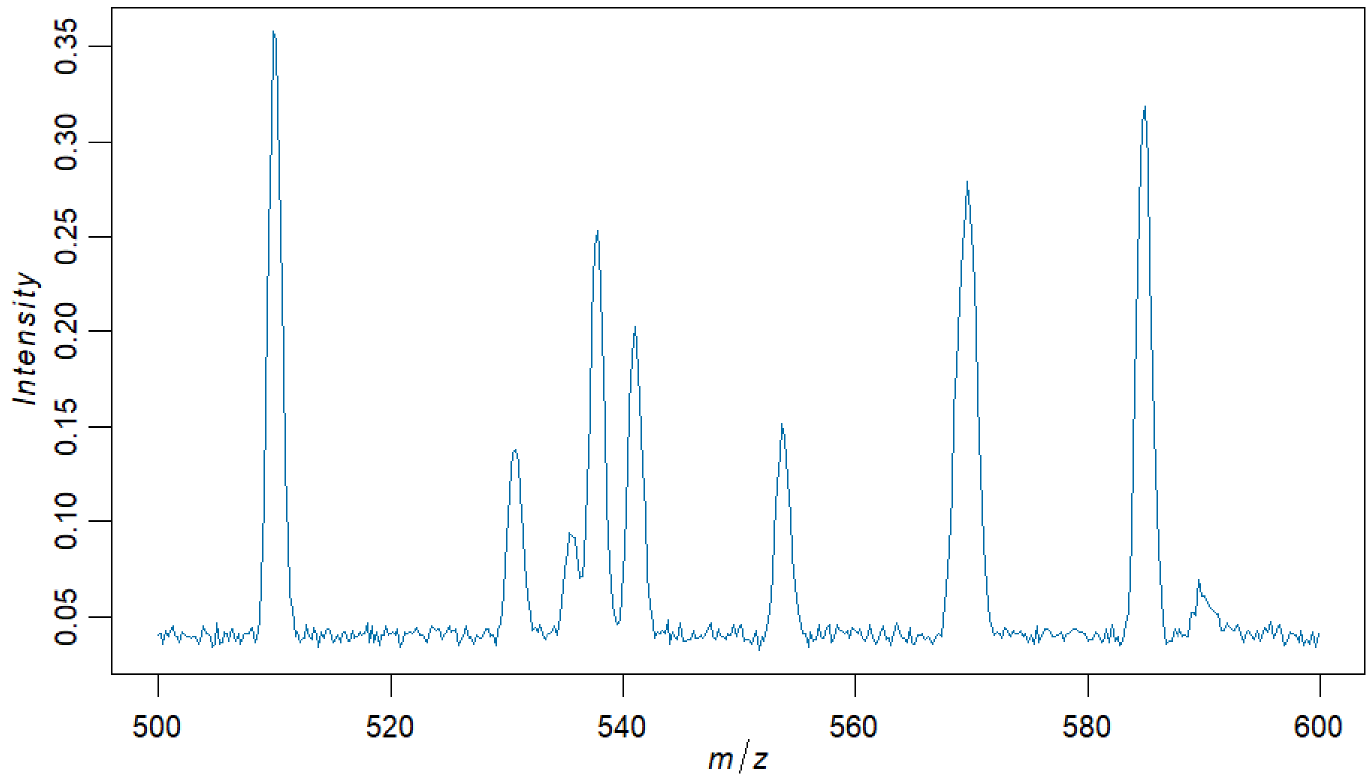
Hide

```
path <- tempfile()
writeMSIData(ori, file=path, outformat="imzML")
```

Plotting the simulated data

Hide

```
plot(ori)
```



Make reference by using Cardinal

Hide

```
ori_mean <- summarizeFeatures(ori)
```

```
summarizing [mean] by feature ...
processing chunk 1/20
processing chunk 2/20
processing chunk 3/20
processing chunk 4/20
processing chunk 5/20
processing chunk 6/20
processing chunk 7/20
processing chunk 8/20
processing chunk 9/20
processing chunk 10/20
processing chunk 11/20
processing chunk 12/20
processing chunk 13/20
processing chunk 14/20
processing chunk 15/20
processing chunk 16/20
processing chunk 17/20
processing chunk 18/20
processing chunk 19/20
processing chunk 20/20
```

Hide

```
ori_peaks <- peakPick(ori_mean, SNR=2)
ori_peaks <- peakAlign(ori_peaks, tolerance=1000, units="ppm")
ori_peaks <- process(ori_peaks)
```

```
processing [peakPick] ...
processing chunk 1/1
postprocessing [peakPick] ...
detected ~7 peaks per spectrum (min = 7, max = 7)
preprocessing [peakAlign] ...
processing chunk 1/1
postprocessing [peakAlign] ...
dropping feature metadata cols: mean
nrows changed from 456 to 7
aligned to 7 reference peaks (tol = 1000 ppm)
done.
```

Export the reference as a imzML file Note: This is used for MSIWarp, as at the time, to check mainly if MSIWarp was actually warping the originally spectra.

Hide

```
path <- tempfile()
writeMSIData(ori_mean, file=path, outformat="imzML")
```

Python code using MSIWarp with the simulated data

Hide

```
import msiwarp as mx
from pyimzml.ImzMLParser import ImzMLParser

from msiwarp.util.warp import to_mx_peaks, to_mz, to_height
from msiwarp.util.warp import generate_mean_spectrum, plot_range

import matplotlib.pyplot as plt
import numpy as np

#Setting up nodes, this setup is taken from the git repo for MSIWarp
node_mzs = [500, 530, 580, 600]
node_deltas = [0.05, 0.053, 0.058, 0.06] # slacks = node_deltas * n_steps
n_steps = 25
nodes = mx.initialize_nodes(node_mzs, node_deltas, n_steps)
epsilon = 1.0

#code to load in unaligned spectrum
p = ImzMLParser("simTest2.imzML")
spectra = []

for idx, coords in enumerate(p.coordinates):
    mzs, hs = p.getspectrum(idx)
    spectra.append(to_mx_peaks(mzs, hs,
                               1.0, id = idx))

# choose a reference spectrum
ref = ImzMLParser("simTest2_Ref.imzML")
ref_spec = []

for idx, coords in enumerate(ref.coordinates):
    mzs, hs = ref.getspectrum(idx)
    ref_spec.append(to_mx_peaks(mzs, hs,
                               1.0, id = idx))


#NOTE: Currently this does not work
#The goal was to take the reference given from cardinal and use
#peaks_top_n which sorts the peaks picked by intensity
'''
ref_100 = mx.peaks_top_n(ref_spec, 100)
mz_ref = np.sort(to_mz(ref_100))
'''

#This section of code is taken from the example given from the git repo
#for MSIWarp. This sets up a reference spectra that will be used for the
#warping function below
i_r = 0
s_r = ref_spec[i_r]


print("warping spectra...")
```

```
import time
t0 = time.time()
optimal_moves = mx.find_optimal_spectra_warpings(spectra, s_r, nodes, epsilon) #uses ms_ref inst
ead of s_r
t1 = time.time()
print("found optimal warpings in {:0.2f}s".format(t1 - t0))

t2 = time.time()
warped_spectra = [mx.warp_peaks(s_i, nodes, o_i) for (s_i, o_i) in zip(spectra, optimal_moves)]
t3 = time.time()
print("warped spectra in {:0.2f}s".format(t3 - t2))

#This code is to export the aligned spectra to be used in cardinal for
#comparison. It is currently commented off.
'''
from pyimzml.ImzMLWriter import ImzMLWriter

output_imzml = 'msiwarp_Test8.imzML'
with ImzMLWriter(output_imzml) as w:
    for s_i, coords in zip(warped_spectra, p.coordinates):
        # writes data to the .ibd file
        w.addSpectrum(to_mz(s_i), to_height(s_i), coords)
'''
```

Run Cardinal Alignment on simulated data comparsion

Hide

```
ori_align1 <- mzAlign(ori, ref=mz(ori_peaks), tolerance=2000, units="ppm")
ori_align1 <- process(ori_align1)
```

```
processing [mzAlign] ...
processing chunk 1/20
processing chunk 2/20
processing chunk 3/20
processing chunk 4/20
processing chunk 5/20
processing chunk 6/20
processing chunk 7/20
processing chunk 8/20
processing chunk 9/20
processing chunk 10/20
processing chunk 11/20
processing chunk 12/20
processing chunk 13/20
processing chunk 14/20
processing chunk 15/20
processing chunk 16/20
processing chunk 17/20
processing chunk 18/20
processing chunk 19/20
processing chunk 20/20
done.
```

Loading MSIWarp alignment results of the simulated data

Hide

```
path_in <- paste0("msiwarp_Test7", ".imzML")
MSIWarp <- readMSIData(path_in, attach.only=TRUE)
```

```
reading imzML file: 'C:\Users\jonat\Desktop\Coding eviorments\R\Markdown\Cardinal-Testing\msiwar
p_Test7.imzML'
detected ibd binary type: ‘processed’
detected representation: ‘centroid spectrum’
reading ibd file: 'C:\Users\jonat\Desktop\Coding eviorments\R\Markdown\Cardinal-Testing\msiwarp_
Test7.ibd'
auto-determining mass range and resolution...
processing chunk 1/20
processing chunk 2/20
processing chunk 3/20
processing chunk 4/20
processing chunk 5/20
processing chunk 6/20
processing chunk 7/20
processing chunk 8/20
processing chunk 9/20
processing chunk 10/20
processing chunk 11/20
processing chunk 12/20
processing chunk 13/20
processing chunk 14/20
processing chunk 15/20
processing chunk 16/20
processing chunk 17/20
processing chunk 18/20
processing chunk 19/20
processing chunk 20/20
detected mass range: 499.95 to 599.8671
estimated mass resolution: 400 ppm
using mass range: 499.95 to 599.8671
using mass resolution: 400 ppm
done.
```

Hide

```
MSIWarp
```

```
An object of class 'MSProcessedImagingExperiment'
  <461 feature, 400 pixel> imaging dataset
    imageData(1): intensity
    featureData(0):
    pixelData(0):
    metadata(12): spectrum representation ibd binary type ... files name
    run(1): msiwarp_Test7
    raster dimensions: 20 x 20 x 1
    coord(3): x = 1..20, y = 1..20, z = 1..1
    mass range: 499.0000 to 599.8059
    centroided: TRUE
```

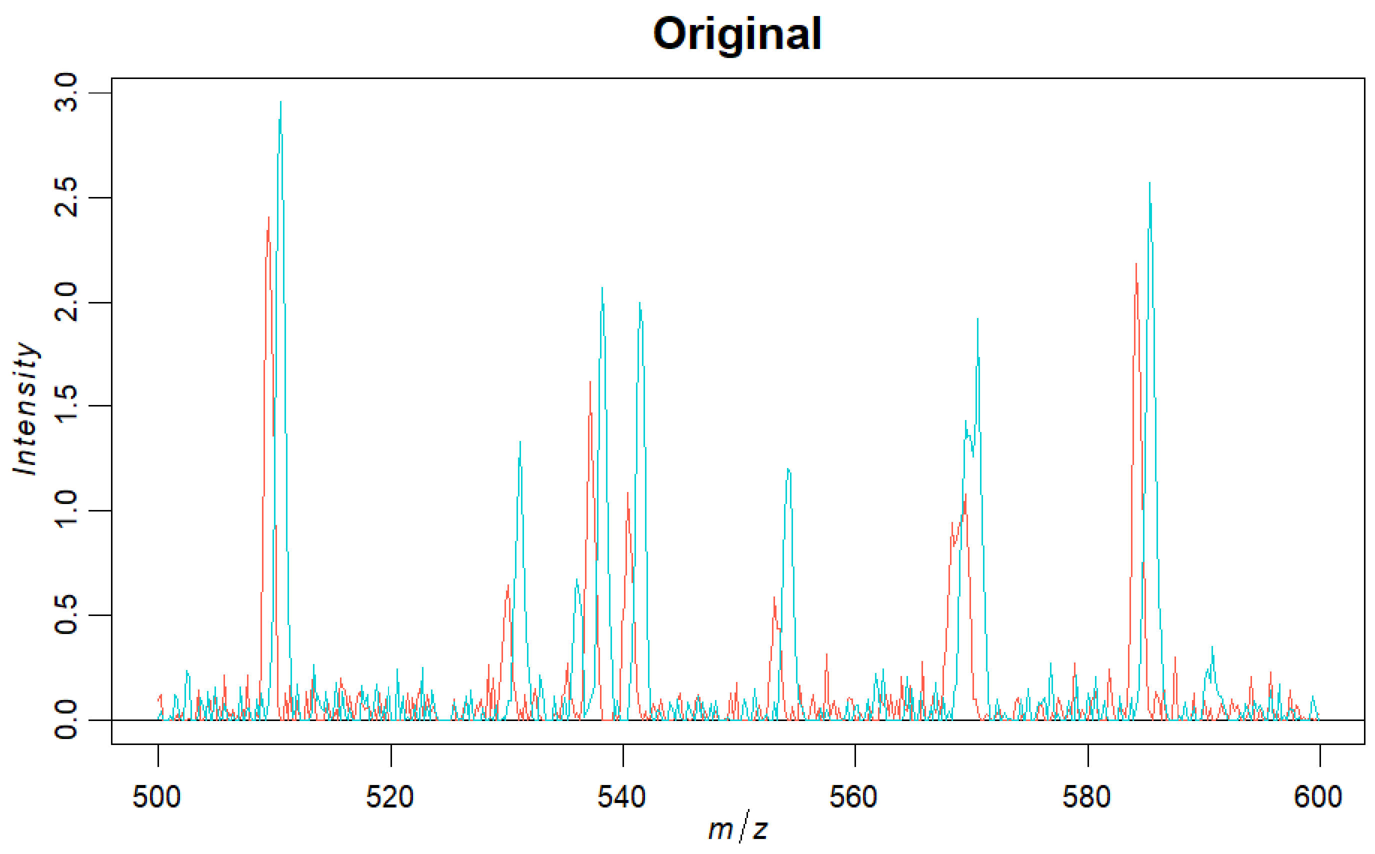
Hide

```
centroided(MSIWarp) = FALSE
```

Plotting original simulated data, reference used, xxx pixel for Cardinal and MSIWarp aligned spectra.

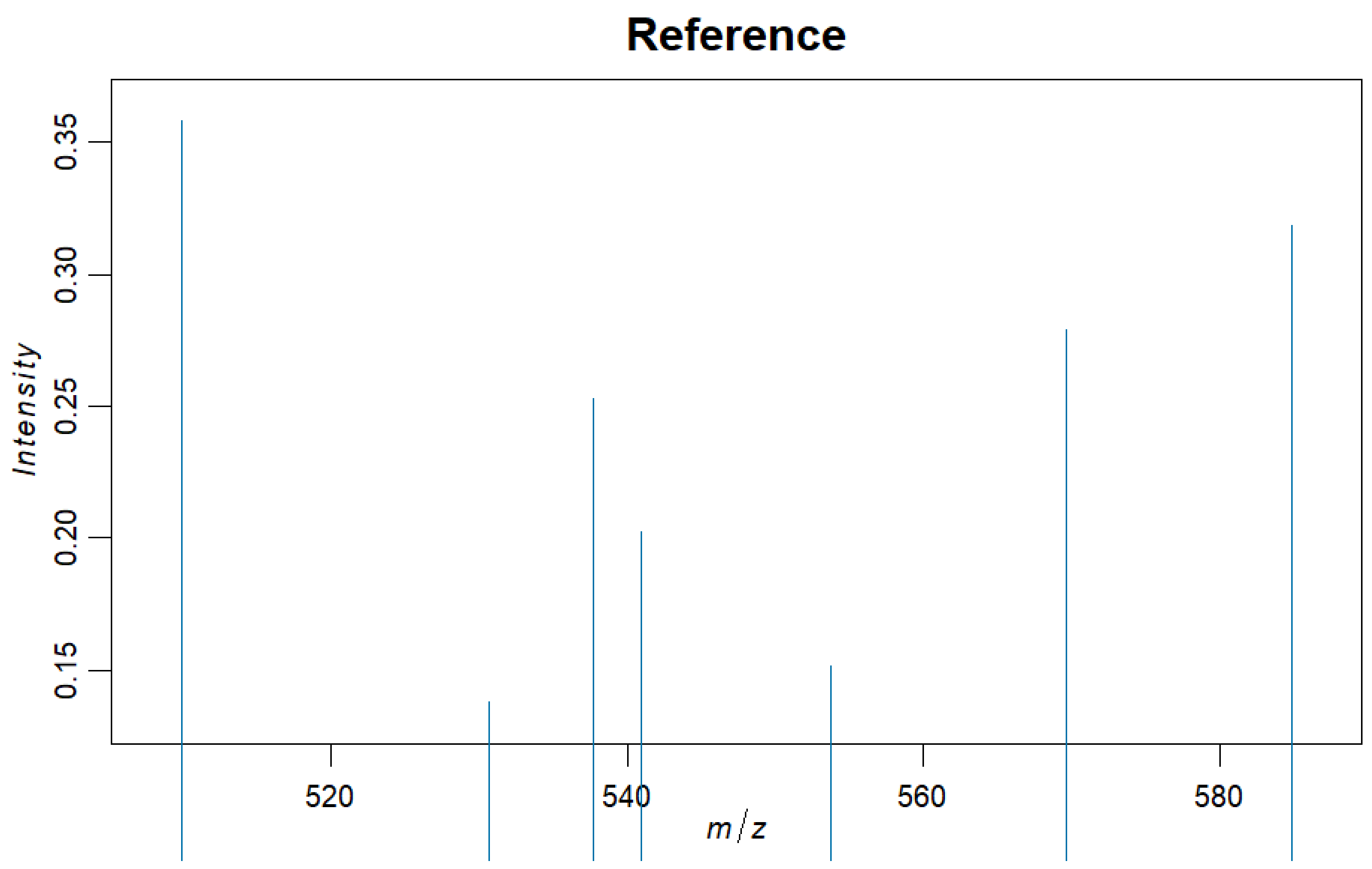
Hide

```
#Set the pixel(s) to look at
curr_pixel = c(193,195)

#plot original
plot(ori, pixel= curr_pixel, key=FALSE, superpose=TRUE, main="Original")
```



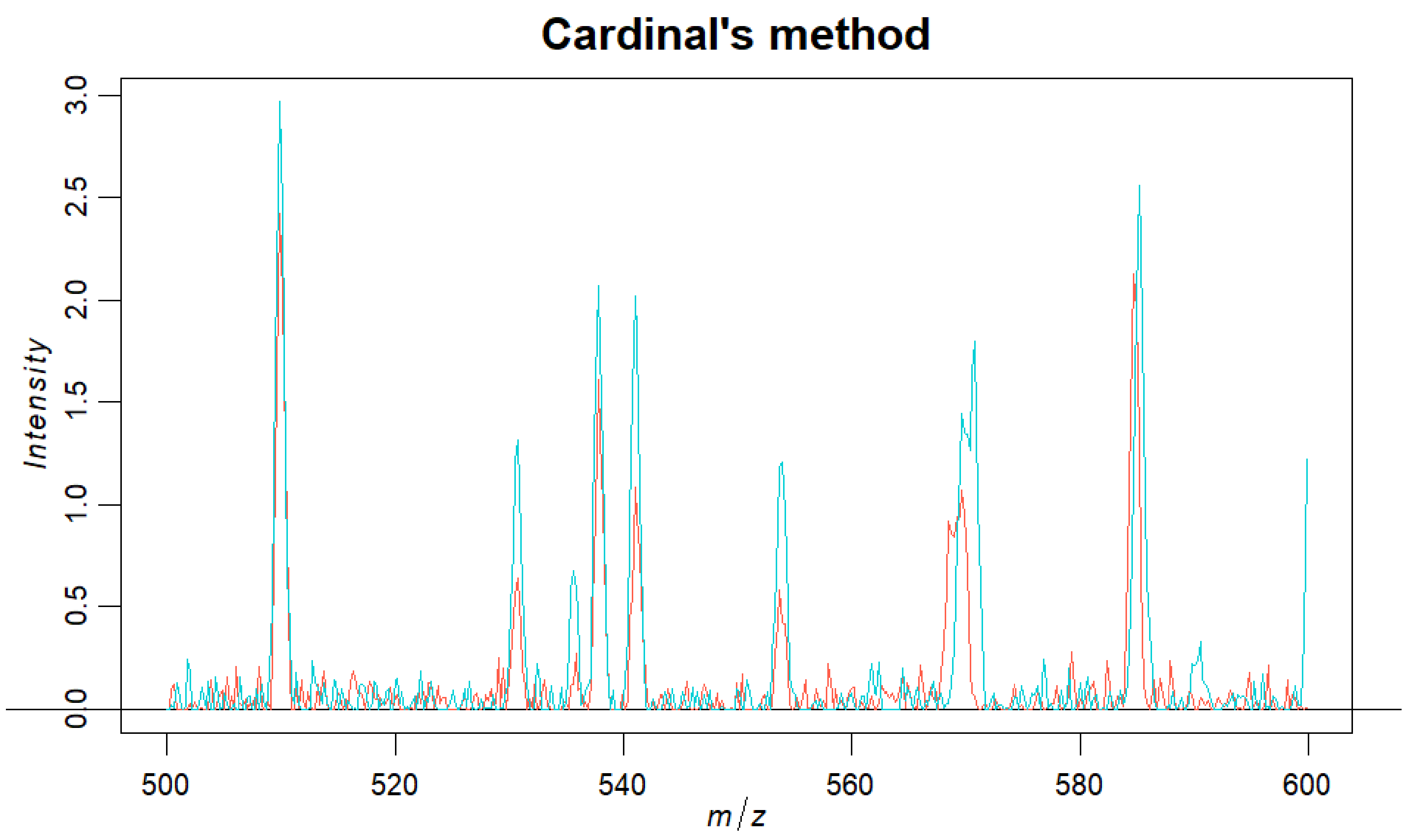
Hide

```
#plot ref used in cardinal and MSIWarp
plot(ori_peaks, main="Reference")
```



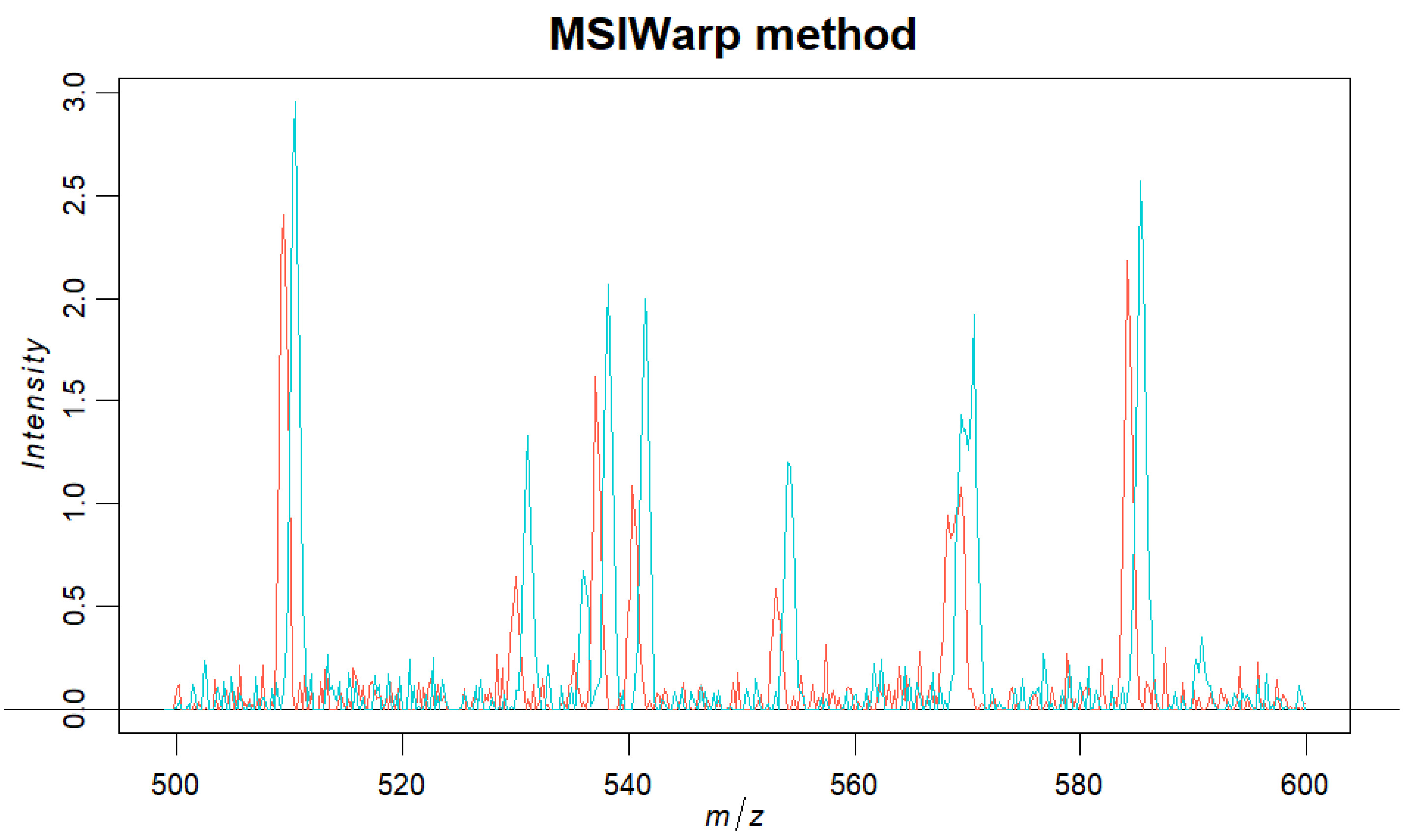
Hide

```
#Plot cardinals alignment
plot(ori_align1, pixel=curr_pixel, key=FALSE, superpose=TRUE, main="Cardinal's method")
```



Hide

```
#plotting for the MSIWarp alignment results for the current pixel
plot(MSIWarp, pixel= curr_pixel, key=FALSE, superpose=TRUE, main="MSIWarp method")
```



Hide

```
NA
NA
```

MSIWarp data featured on paper. Data named “A67 CT S4-centroid” (named MSIWarp_orbitrap_desi)

Hide

```
path_in <- paste0("A67 CT S4-centroid", ".imzML")
MSIWarp_orbitrap_desi <- readMSIData(path_in, attach.only=TRUE)
```

```
reading imzML file: 'C:\Users\jonat\Desktop\Coding eviorments\R\Markdown\Cardinal-Testing\A67 CT
S4-centroid.imzML'
detected ibd binary type: ‘processed’
detected representation: ‘centroid spectrum’
reading ibd file: 'C:\Users\jonat\Desktop\Coding eviorments\R\Markdown\Cardinal-Testing\A67 CT S
4-centroid.ibd'
auto-determining mass range and resolution...
processing chunk 1/20
processing chunk 2/20
processing chunk 3/20
processing chunk 4/20
processing chunk 5/20
processing chunk 6/20
processing chunk 7/20
processing chunk 8/20
processing chunk 9/20
processing chunk 10/20
processing chunk 11/20
processing chunk 12/20
processing chunk 13/20
processing chunk 14/20
processing chunk 15/20
processing chunk 16/20
processing chunk 17/20
processing chunk 18/20
processing chunk 19/20
processing chunk 20/20
detected mass range: 200.037 to 1000.1647
estimated mass resolution: 1172 ppm
using mass range: 200.037 to 1000.1647
using mass resolution: 1172 ppm
done.
```

Hide

```
MSIWarp_orbitrap_desi
```

```
An object of class 'MSProcessedImagingExperiment'
  <1375 feature, 20286 pixel> imaging dataset
    imageData(1): intensity
    featureData(0):
    pixelData(0):
    metadata(14): spectrum representation ibd binary type ... files name
    run(1): A67 CT S4-centroid
    raster dimensions: 147 x 138
    coord(2): x = 1..147, y = 1..138
    mass range:  200.000 to 1000.891
    centroided: TRUE
```

Hide

```
centroided(MSIWarp_orbitrap_desi) = FALSE

path_in <- paste0("msiwarp_Test_orbi_desi_1", ".imzML")
MSIWarp_orbitrap_desi_align <- readMSIData(path_in, attach.only=TRUE)
```

```
reading imzML file: 'C:\Users\jonat\Desktop\Coding eviorments\R\Markdown\Cardinal-Testing\msiwar
p_Test_orbi_desi_1.imzML'
detected ibd binary type: ‘processed’
detected representation: ‘centroid spectrum’
reading ibd file: 'C:\Users\jonat\Desktop\Coding eviorments\R\Markdown\Cardinal-Testing\msiwarp_
Test_orbi_desi_1.ibd'
auto-determining mass range and resolution...
processing chunk 1/20
processing chunk 2/20
processing chunk 3/20
processing chunk 4/20
processing chunk 5/20
processing chunk 6/20
processing chunk 7/20
processing chunk 8/20
processing chunk 9/20
processing chunk 10/20
processing chunk 11/20
processing chunk 12/20
processing chunk 13/20
processing chunk 14/20
processing chunk 15/20
processing chunk 16/20
processing chunk 17/20
processing chunk 18/20
processing chunk 19/20
processing chunk 20/20
detected mass range: 200.037 to 999.9991
estimated mass resolution: 1172 ppm
using mass range: 200.037 to 999.9991
using mass resolution: 1172 ppm
done.
```

Hide

```
MSIWarp_orbitrap_desi_align
```

```
An object of class 'MSProcessedImagingExperiment'
  <1374 feature, 20286 pixel> imaging dataset
    imageData(1): intensity
    featureData(0):
    pixelData(0):
    metadata(12): spectrum representation ibd binary type ... files name
    run(1): msiwarp_Test_orbi_desi_1
    raster dimensions: 147 x 138 x 1
    coord(3): x = 1..147, y = 1..138, z = 1..1
    mass range: 200.0000 to 999.7183
    centroided: TRUE
```

Hide

```
centroided(MSIWarp_orbitrap_desi_align) = FALSE
```

Using Cardinals alignment method

Hide

```
MSIWarp_orbitrap_desi
```

```
An object of class 'MSProcessedImagingExperiment'
  <1375 feature, 20286 pixel> imaging dataset
    imageData(1): intensity
    featureData(0):
    pixelData(0):
    metadata(14): spectrum representation ibd binary type ... files name
    run(1): A67 CT S4-centroid
    raster dimensions: 147 x 138
    coord(2): x = 1..147, y = 1..138
    mass range:  200.000 to 1000.891
    centroided: FALSE
```

Hide

```
orbitrap_mean <- summarizeFeatures(MSIWarp_orbitrap_desi)
```

```
summarizing [mean] by feature ...
processing chunk 1/20
processing chunk 2/20
processing chunk 3/20
processing chunk 4/20
processing chunk 5/20
processing chunk 6/20
processing chunk 7/20
processing chunk 8/20
processing chunk 9/20
processing chunk 10/20
processing chunk 11/20
processing chunk 12/20
processing chunk 13/20
processing chunk 14/20
processing chunk 15/20
processing chunk 16/20
processing chunk 17/20
processing chunk 18/20
processing chunk 19/20
processing chunk 20/20
```

Hide

```
orbitrap_peaks <- peakPick(orbitrap_mean, SNR=2)
orbitrap_peaks <- peakAlign(orbitrap_peaks, tolerance=1000, units="ppm")
orbitrap_peaks <- process(orbitrap_peaks)
```

```
processing [peakPick] ...
processing chunk 1/1
postprocessing [peakPick] ...
detected ~79 peaks per spectrum (min = 79, max = 79)
preprocessing [peakAlign] ...
processing chunk 1/1
postprocessing [peakAlign] ...
dropping feature metadata cols: mean
nrows changed from 1375 to 79
aligned to 79 reference peaks (tol = 1000 ppm)
done.
```

Hide

```
#print/plot resulting alignment
orbitrap_align1 <- mzAlign(MSIWarp_orbitrap_desi, ref=mz(orbitrap_peaks), tolerance=2000, units=
"ppm")
orbitrap_align1 <- process(orbitrap_align1)
```

```
processing [mzAlign] ...
processing chunk 1/20
processing chunk 2/20
processing chunk 3/20
processing chunk 4/20
processing chunk 5/20
processing chunk 6/20
processing chunk 7/20
processing chunk 8/20
processing chunk 9/20
processing chunk 10/20
processing chunk 11/20
processing chunk 12/20
processing chunk 13/20
processing chunk 14/20
processing chunk 15/20
processing chunk 16/20
processing chunk 17/20
processing chunk 18/20
processing chunk 19/20
processing chunk 20/20
done.
```

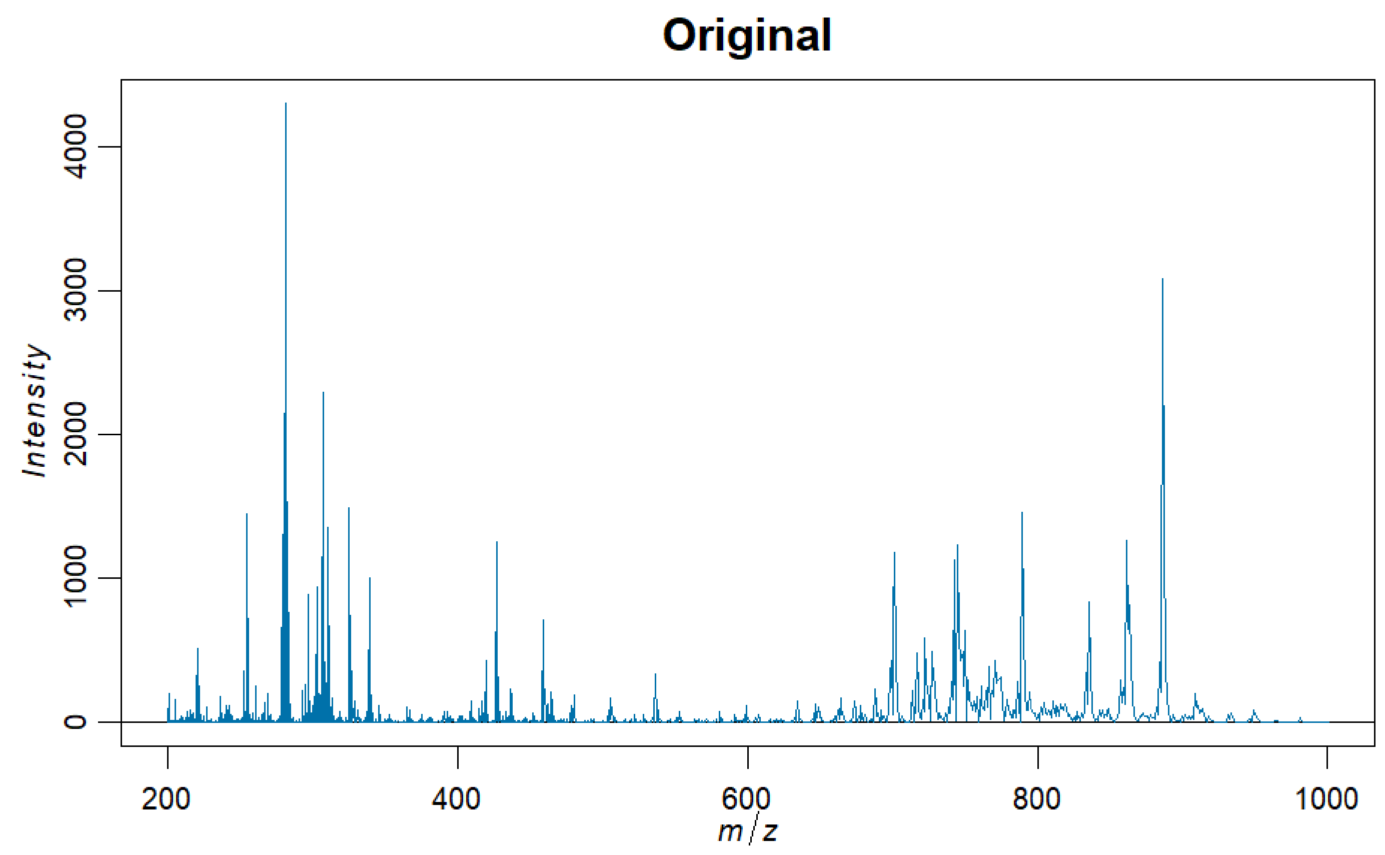
Python code that works with the data. This code is a snippet from an example used in the git repo for MSIWarp (file: warp_orbitrap_desi.py)

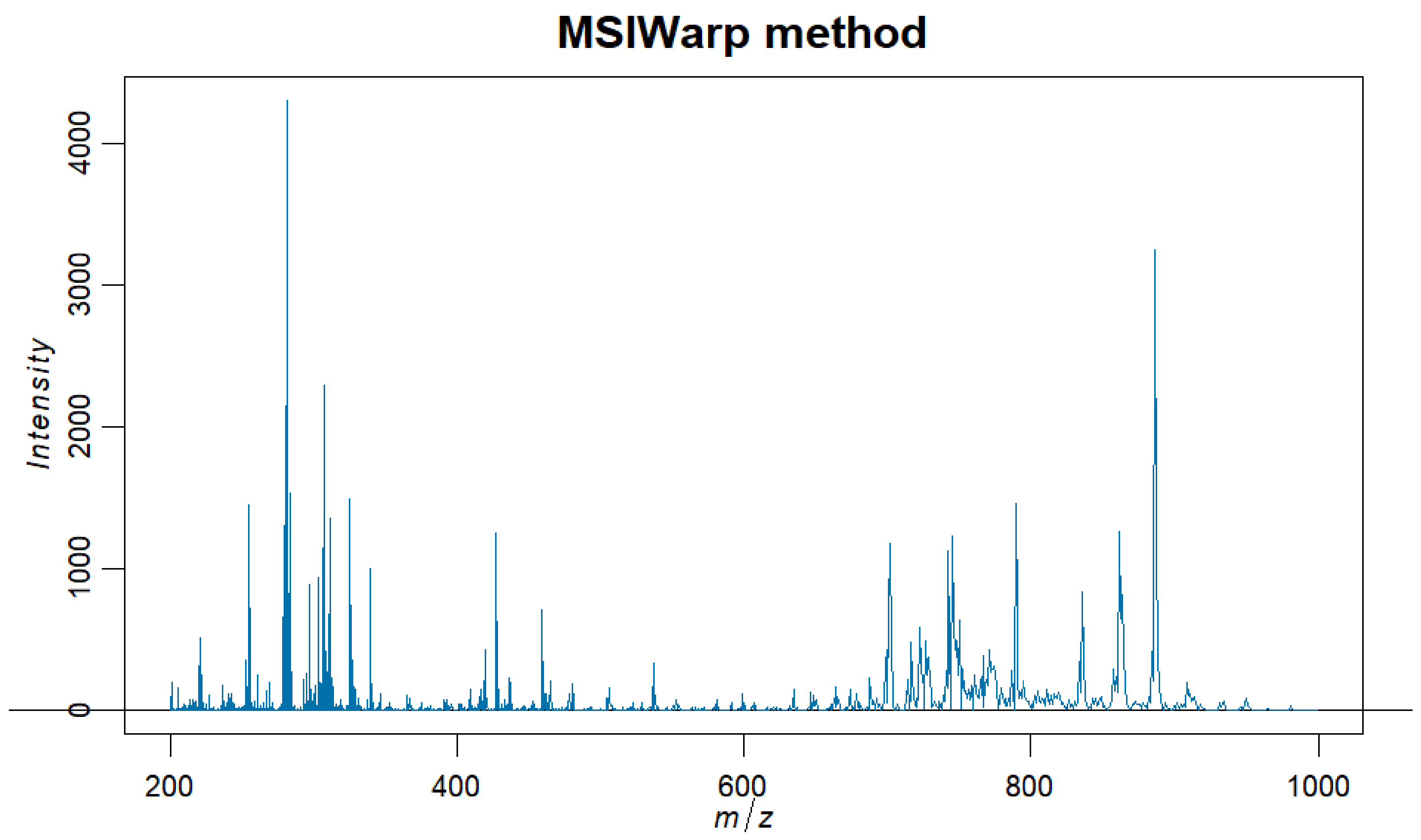
Hide

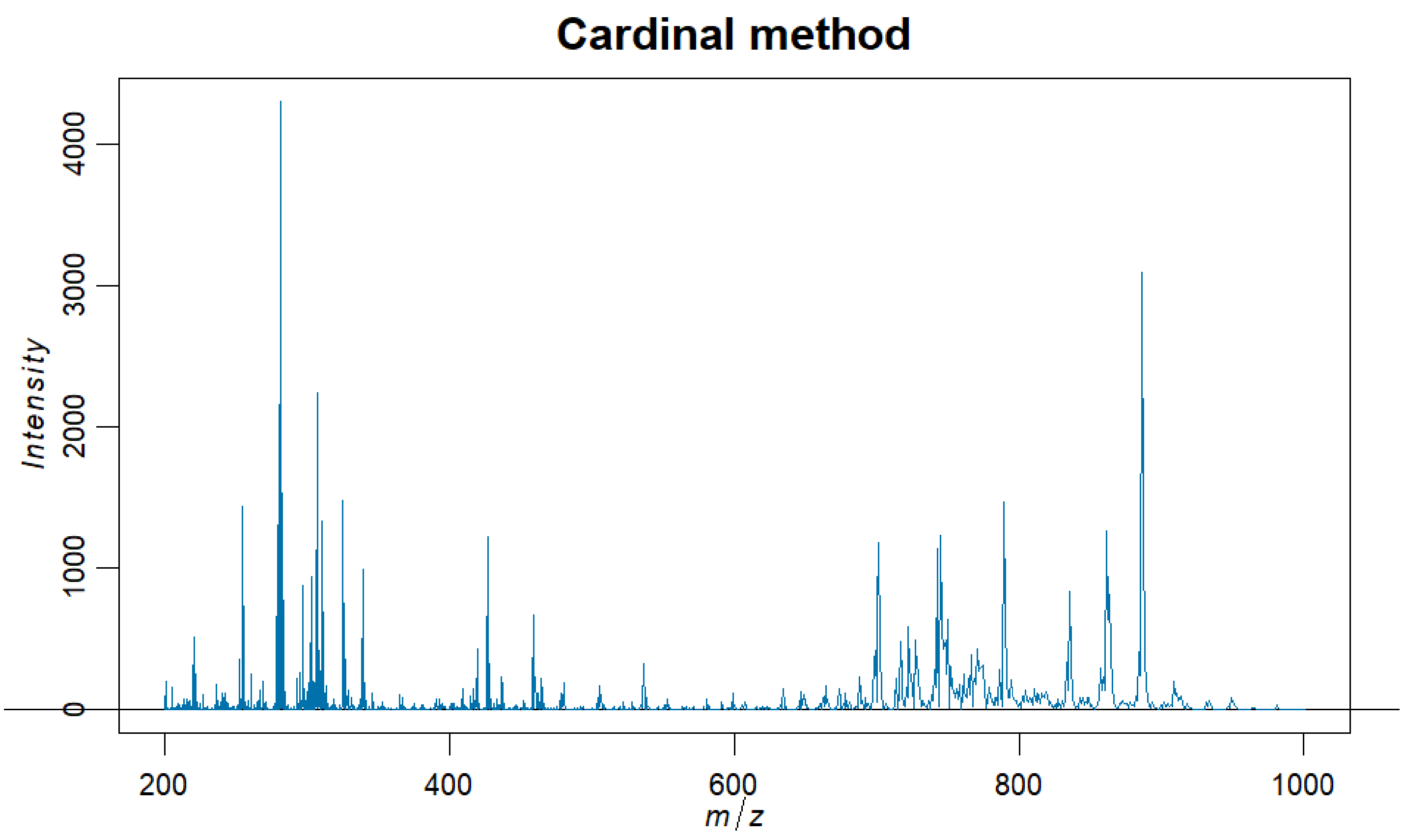
```
import msiwarp as mx

from msiwarp.util.warp import to_mx_peaks, to_mz, to_height
from msiwarp.util.warp import generate_mean_spectrum, plot_range

import matplotlib.pyplot as plt
import numpy as np

imzml_path = "A67 CT S4-centroid.imzML"

# experiment settings
sigma_1 = 3.0e-7
epsilon = 1.0
instrument_type = 'orbitrap'

# --------- load spectra and meta data ----------
from pyimzml.ImzMLParser import ImzMLParser
from msiwarp.util.warp import to_mx_peaks

p = ImzMLParser("A67 CT S4-centroid.imzML")
spectra = []

for idx, coords in enumerate(p.coordinates):
    mzs, hs = p.getspectrum(idx)
    spectra.append(to_mx_peaks(mzs, hs,
                               sigma_1, id = idx,
                               instrument_type = instrument_type))


tic = np.array([np.sum(to_height(s_i)) for s_i in spectra])
n_peaks = np.array([len(s_i) for s_i in spectra])


# ---------- use uniform node placement function ----------
n_steps = 33
n_peaks_min = 30
n_nodes = 8
mz_begin = 175
mz_end = 1000
slack = 2.0 * epsilon * sigma_1

params = mx.params_uniform(mx.Instrument.Orbitrap,
                           n_steps,
                           n_peaks_min,
                           n_nodes,
                           mz_begin,
                           mz_end,
                           slack)
```

```
# ---------- mean spectrum ----------
n_points = 2000000
s_m = generate_mean_spectrum(spectra, n_points, sigma_1,
                             mz_begin, mz_end, tic, instrument_type)

print("using mean spectrum as reference")
s_m_1000 = mx.peaks_top_n(s_m, 1000) # returns peak list sorted by intensity, not m/z
s_ref = sorted(s_m_1000, key=lambda peak: peak.mz)

s_m_100 = mx.peaks_top_n(s_m, 100)
mz_ref = np.sort(to_mz(s_m_100))



# ---------- warp spectra ----------
print("warping spectra...")

import time

n_cores = 8

t0 = time.time()
warping_funcs = mx.find_optimal_warpings_uni(spectra, s_ref, params, epsilon, n_cores)
t1 = time.time()
print("found optimal warpings in {:0.2f} seconds".format(t1 - t0))

t2 = time.time()
warped_spectra = [mx.warp_peaks_unique(s_i, r_i) for (s_i, r_i) in zip(spectra, warping_funcs)]
t3 = time.time()
print("warped spectra in {:0.2f}s".format(t3 - t2))


#This code is to export the aligned spectra to be used in cardinal for
#comparesion. It is currently commented off.

'''
from pyimzml.ImzMLWriter import ImzMLWriter

output_imzml = 'msiwarp_Test_orbi_desi_1.imzML'
with ImzMLWriter(output_imzml) as w:
    for s_i, coords in zip(warped_spectra, p.coordinates):
        # writes data to the .ibd file
        w.addSpectrum(to_mz(s_i), to_height(s_i), coords)
'''
```

plotting original, MSIWarp, and Cardinal aligned spectra


Original
Intensity
4000
3000
2000
1000
0
200
400
600
800
1000
m/z


MSIWarp method
Intensity
4000
3000
2000
1000
0
200
400
600
800
1000
m/z



References

1. MSIWarp - https://pubs.acs.org/doi/full/10.1021/acs.analchem.0c03833 (https://pubs.acs.org/doi/full/10.1021/acs.analchem.0c03833)
2. MSIWarp Git Repo- https://github.com/horvatovichlab/MSIWarp (https://github.com/horvatovichlab/MSIWarp)
3. MSIWarp Data - https://www.ebi.ac.uk/metabolights/MTBLS289/files (https://www.ebi.ac.uk/metabolights/MTBLS289/files)