Electronic Supplementary Material (ESI) for Polymer Chemistry.

Supporting Information

Folding Mass Spectra: How to Deal with the Signal to Noise $\sqrt{N}$ dilemma

Tanja Junkers, and Iyomali Abeysekera*

Polymer Reaction Design Group, School of Chemistry, Monash University, 17 Rainforest Walk, Clayton VIC 3800, Australia

tanja.junkers@monash.edu

All python code is available from github:

https://github.com/PRDMonash/mass_spec_folding

Installation of 'rdkit' environment to run the Jupyter Notebook

- **Install Anaconda3**

https://www.anaconda.com

- **Open Anaconda3 prompt and install rdkit in separate environment**

How to install RDKit with Conda

Creating a new conda environment with the RDKit installed requires one single command similar to the following::

```
$ conda create -c conda-forge -n my-rdkit-env rdkit
```

Finally, the new environment must be activated so that the corresponding python interpreter becomes available in the same shell:

```
$ conda activate my-rdkit-env
```

If for some reason this does not work, try:

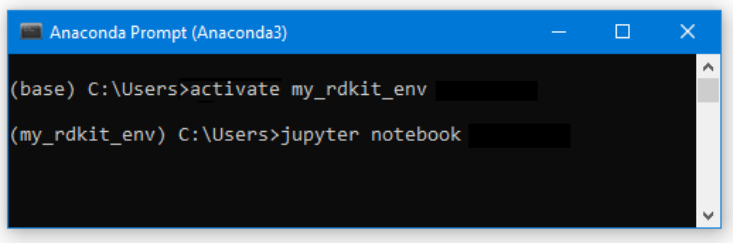
```
$ cd [anaconda folder]/bin
$ source activate my-rdkit-env
```

Windows users will use a slightly different command:

```
C:\> activate my-rdkit-env
```

(taken from https://www.rdkit.org/docs/Install.html)

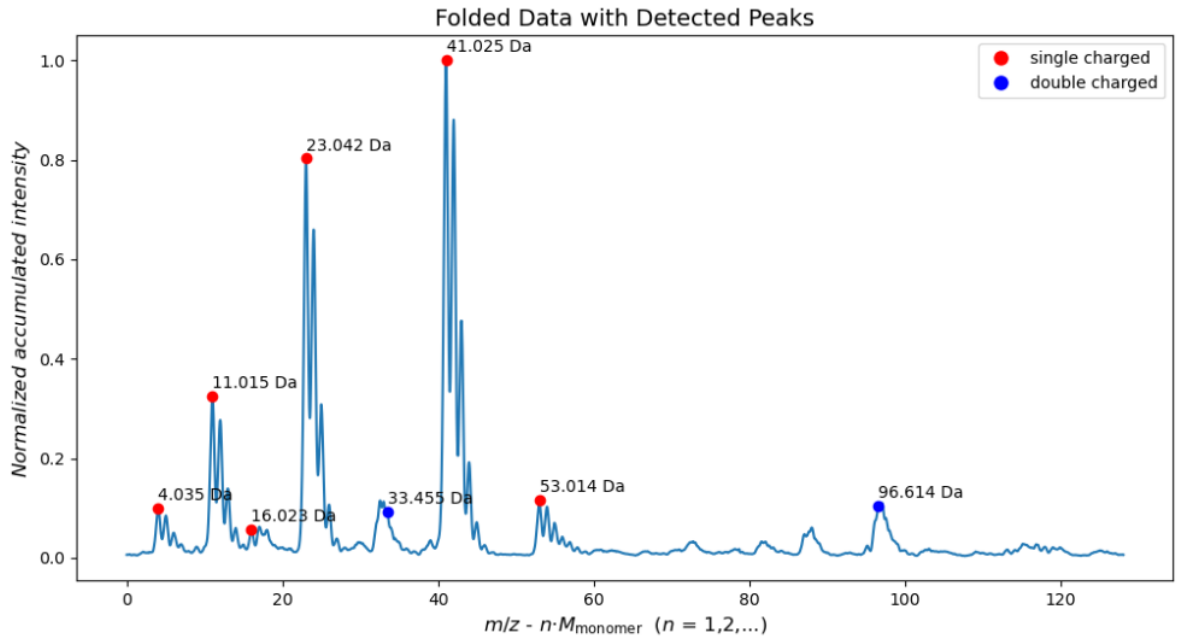
- **Start Jupyter Notebook from the command prompt**



This will open your browser. Select location of program code and run. Please note that the code will open a windows shell to select the csv file for processing in the background as Jupyter Notebook is forced to stay in the foreground. Hence, one needs to minimize the browser window after running the first sequence of the code.

- **The code expects mass spec data as *csv file. m/z in first column, intensity in the second column. No header, data needs to start from row 1 on.**

The notebook will process data, show the original spectrum and ask for the SMILES code of the monomer, the minimum m/z and maximum m/z of the original spectrum that should be considered. The notebook will then create a key output:



Peaks are automatically picked where possible. In the last step of the code, the output data and the picked peaks are saved to a csv file which appends *'_output'* to the original file name.

Python code for the fold_massspec function

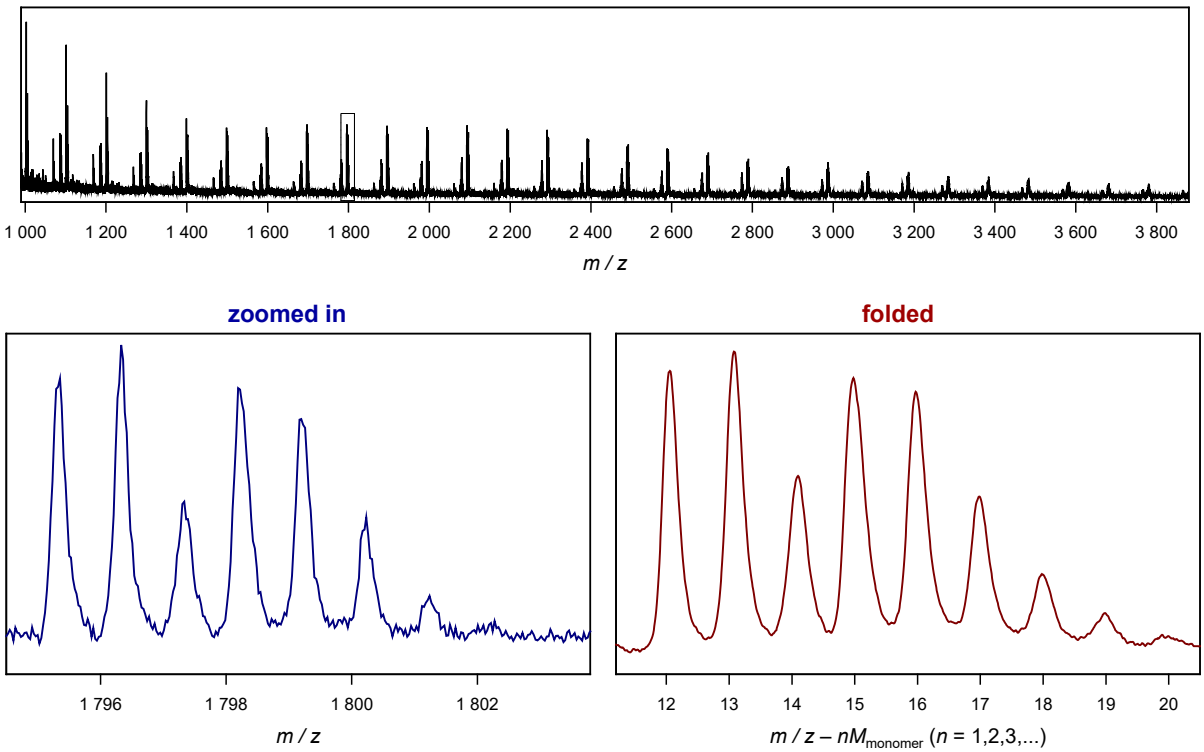
```python
import pandas as pd
import numpy as np
import math
from scipy.interpolate import interp1d
import os


def fold_massspec(mz_min, mz_max, monoisotopicweight, data):

    # Filter data within the mz range
    data = data[(data['m/z'] >= mz_min) & (data['m/z'] <= mz_max)]

    # Normalize m/z values
    data['m/z'] = (data['m/z'] - mz_min) / monoisotopicweight

    # Interpolation
    x = data['m/z'].values
    y = data['intensity'].values
    f = interp1d(x, y, bounds_error=False, fill_value=0)

    # Create a range of x values
    max_val = x[-1]
    new_x_0_1 = np.linspace(0, 1, 10001)
    step_size = new_x_0_1[1] - new_x_0_1[0]
    new_x_rest = np.arange(1 + step_size, max_val, step_size)
    new_x = np.concatenate((new_x_0_1, new_x_rest))

    # Interpolate y values
    new_y = f(new_x)

    # Fold data in chunks of 10,000 values
    new_y_folded = new_y[:10000].copy()
    for i in range(10000, len(new_y), 10000):
        end_idx = i + 10000 if i + 10000 < len(new_y) else len(new_y)
        chunk = new_y[i:end_idx]
        new_y_folded[:len(chunk)] += chunk

    # Create a DataFrame for the folded data
    folded_data = pd.DataFrame({
        'm/z': new_x[:10000] * monoisotopicweight,
        'intensity': new_y_folded
    })

    # Normalize intensity values
    folded_data['intensity'] = folded_data['intensity'] / folded_data['intensity'].max()

    return folded_data
```

Example code to run *fold_massspec* function and plot residual spectrum

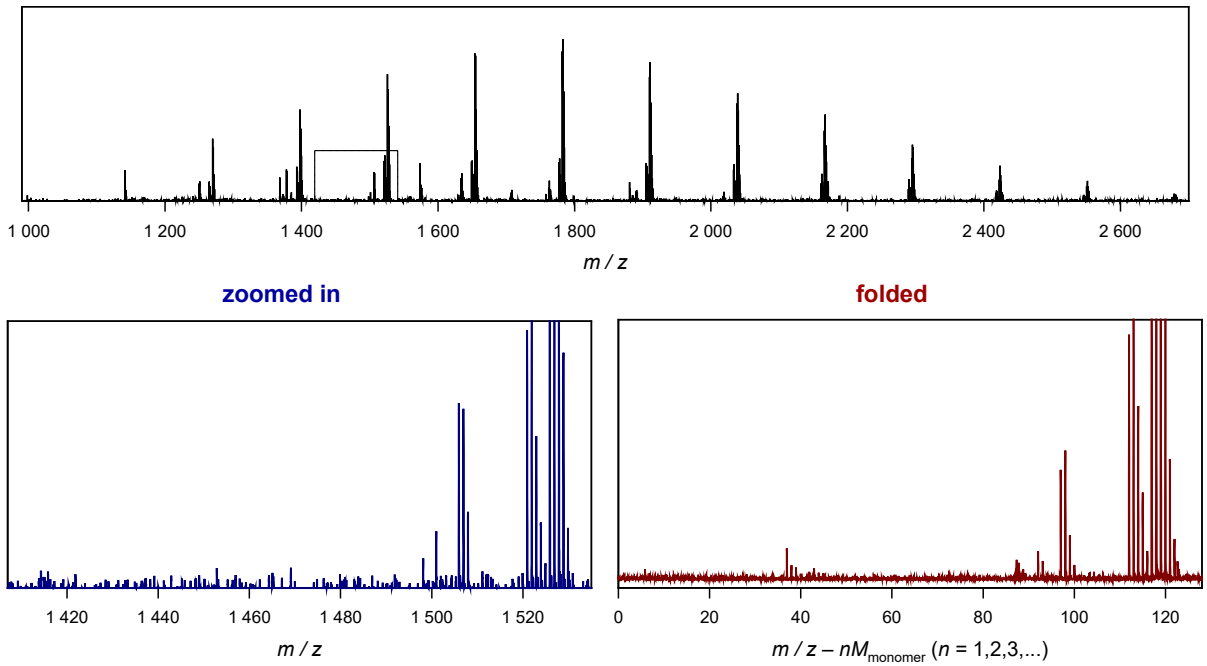
```
import pandas as pd
import numpy as np
import math
from scipy.interpolate import interp1d
import os
import matplotlib.pyplot as plt


def fold_massspec(mz_min, mz_max, monoisotopicweight, data):

[...]

# Example usage:
mz_min = 750
mz_max = 2000
monoisotopicweight = 128.083729624
csv_file_path =  r"YOUR PATH AND FILENAME HERE *.csv"

# Read CSV data
data = pd.read_csv(csv_file_path, header=None, names=['m/z', 'intensity'])

# Fold the spectrum and apply the function
folded_data = fold_massspec(mz_min, mz_max, monoisotopicweight, data)

# Plot 'intensity' against 'm/z'
plt.figure(figsize=(10, 6))  # Adjust the figure size as needed
plt.plot(folded_data['m/z'], folded_data['intensity'])
plt.xlabel('m/z')
plt.ylabel('Intensity')
plt.title('Mass Spectrometry Data')
plt.grid(True)

# Show the plot
plt.show()

input("Press Enter to exit...")
```

(full code available for download from github)

Further examples for folded spectra



Example of the output generated by the Jupyter Notebook for folding of a MALDI spectrum of RAFT-derived pDMA (*N,N* dimethyl acrylamide). The top depicts the full MALDI raw spectrum. The bottom left shows the zoom into one repeat unit with an enlargement of the main product, and the bottom right gives the corresponding folded spectrum. The authors are grateful of donation of this data by Richard Whitfield and Athina Anastasaki.



Example of the output generated by the Jupyter Notebook for folding of an ESI-MS spectrum of RAFT-derived pBA (butyl acrylate). The top depicts the full ESI raw spectrum. The bottom left shows the zoom into one repeat unit with an enlargement of the main product, and the bottom right gives the corresponding folded spectrum.