Python based data analysis of ToF-SIMS

Sean Stuart
Sean.c.stuart@aero.org

April 3rd, 2019
Introduction

- Introduction
- Getting the data
- Visualizing the Data
- Principle component analysis & Multivariate Curve Resolution
- Binning data for PCA + MCR
- Multi-Sample comparison
- Conclusion
My background
How and why I started using Python

• I started using Python to make professional looking graphs
  – To fit with publication standards (font size, figure dpi… etc.)
  – Excel wasn’t cutting it, and other options were expensive….

• After getting a feel for python, it was easy to expand from making graphs
  – Modeling/simulation was relatively straightforward, just import a few packages.

• I started working at The Aerospace Corporation in 2015, focusing on ToF-SIMS
  – Working with John Chaney and Robert Moision, who had developed lots of python based techniques for ToF-SIMS data analysis.
  – Using their work I was able to expand on existing techniques, increasing the speed + resolution of the statistical analysis we performed.

• I will show code snippets to demonstrate how easy and straightforward Python can be
Toolbox for Python data analysis

Why Python?

- NumPy
  - N-dimensional arrays
  - Linear algebra and I/O tools
- Pandas
  - “database” with tools for visualizing, searching, sorting data
- HDF
  - Hierarchical data format, developed for synchrotron data provides high performance searching/binning tools
- Scikit Learn
  - Machine learning tools (and “classic” statistical analysis methods)
- Many more
  - MCR, output to PowerPoint, read your email, make a user interface….

All this, and more for free!
ToF-SIMS
Time of Flight Secondary Ion Mass Spectrometry

• ToF-SIMS is an imaging mass spectrometry technique, with a long history in the semiconductor industry.

• Pros:
  – Extreme surface sensitivity, sampling the top atomic / molecular layer
  – Generally non destructive
  – High sensitivity, generally ppm sometimes ppb
  – Capable of detecting and identifying elements and chemicals from hydrogen to large polymers simultaneously.

• Cons
  – Generates large complex datasets
  – UHV technique, generally expensive

Large complex datasets created in ToF-SIMS require statistical tools, and lots of visualization methods to understand and communicate the data.
ToF-SIMS
Time-of-flight secondary ion mass spectrometry

- PRIMARY IONS: A pulsed beam of high energy (30 keV) Bi$_{1-7}^+$ particles aimed at surface.

- Near-surface molecules and atoms are sputtered (~95% of detected species come from top two monolayers).

- Sputtering results in formation of species with neutral, positive, and negative charge (SECONDARY IONS).

- Polarity of extraction field selects either positive or negative ions.

- Ions are mass analyzed with a time-of-flight detector.

- Damage to surface is kept under 1% of total analysis area.

- The primary ion beam is scanned over the surface providing simultaneous collection of spatial and chemical information.
Time-of-flight (ToF) Mass Spectrometry

- Requires a pulsed source
- High mass resolution
- All secondary ions are detected

\[ \text{time} \propto \sqrt{\frac{\text{mass}}{\text{charge}}} \]
Surface Sensitivity
Thin layers of contamination will dominate the signal

Extreme surface sensitivity makes detecting monolayer contaminants easy. (in this case acetone residue is likely much more than a monolayer).
ToF-SIMS Depth Profiling

- During depth profiling, analysis (Bi\(^+\)) and sputtering (Cs\(^+\), Ar, O) beams are alternated.
- This provides chemical information as a function of depth.
- The resulting data looks like a series of slices, which each slice containing a spectral signature of that layer.
Getting the Data

• Introduction
• Getting the data
• Visualizing the Data
• Principle component analysis & Multivariate Curve Resolution
• Binning data for PCA + MCR
• Multi-Sample comparison
• Conclusion
Getting the data
Generally the hardest part of data analysis..

• Easy:
  – Text files, CSV

• Moderate:
  – XML, HDF, non proprietary data formats

• Difficult:
  – Proprietary binary files

Translating data from one format to another can be challenging... but usually worth it
Visualizing the Data

• Introduction
• Getting the data
• **Visualizing the Data**
• Principle component analysis & Multivariate Curve Resolution
• Binning data for PCA + MCR
• Multi-Sample comparison
• Conclusion
Visualizing data
Mass, location, depth

• ToF Sims records a mass spectra at every pixel in the data.
• We can create an Image of each peak in mass spectra
  – Including as a function of depth
Data selection with HDF5
Opened using PyTables

In the HDF data file, we can store information about each ion detected in the ToF-SIMS measurement:

```python
class data_dt(tb.IsDescription):
    px = tb.UInt32Col()
    ch = tb.UInt32Col()
    time = tb.UInt32Col()
    px_x = tb.UInt16Col()
    px_y = tb.UInt16Col()
    m_z = tb.Float32Col()

dt = np.dtype([('px', np.uint32), ('ch', np.uint32), ('time', np.uint32),
               ('px_x', np.uint16), ('px_y', np.uint16), ('m_z', np.float32)])
```

Making an image of a specific mass involves a search, basically asking: Give me all of the ions with mass between <lowmass> and <highmass>. Then count up how many are in each pixel, oh and arrange the pixels into an image:

```python
def makeMapofMassRange(raw_dat, lowmass, highmass, image_size = (256,256), scale = 1):
    search_str = '%(m_z)>({0})&%(m_z)<={1})'.format(lowmass, highmass)
    px_map = [x['px'] for x in raw_dat.where(search_str)]
    px_map = np.bincount(px_map, minlength=image_size[0]*image_size[1])

    if scale != 1:
        image = np.reshape(px_map,(image_size[0],image_size[1]),order='F')
        image = resizebox(image, scale)
        px_map = np.reshape(image, -1)
    return px_map
```

HDF provides efficient, parallel data searches.
By automating data visualization (with Python) you can generate enough images for a video.
Why use Python for data visualization?
Control over every detail.

• The software provided with the instrument might have most of what you need but…
  – Making changes to fonts, aspect ratios, graph styles is difficult or impossible (sometimes publications have strict requirements on figures)
  – Making lots of identical graphs can be tedious.
  – Extracting 100’s of images (for a video) is not worth the time.

• Once you have access to the data in Python, making a new visualization is about as time consuming as figuring out how to use a new feature of a program… and you are not limited by what is already implemented.
• If you have the code to make a graph, I find it much faster to use python.
PCA and MCR

- Introduction
- Getting the data
- Visualizing the Data
- Principle component analysis & Multivariate Curve Resolution
- Binning data for PCA + MCR
- Multi-Sample comparison
- Conclusion
Deconvolution of two similar signatures using PCA

*Using positive and negative correlations to separate components*

- Two overlapping thumb prints (from two people) are distinguished chemically.
- Positive and negative correlations can be used to distinguish regions.
- Both prints overlap but are clearly resolvable with different spectral signatures.
- Without PCA analysis these chemical differences would be extremely difficult to detect.
ToF SIMS analysis background

Data processing with multivariate analysis (MVA)

- A given data set contains mass, pixel position and time
  - Easily exceeding 9,000,000 data points
  - Single files may be 4 GB

- Multivariate curve resolution (MCR) and principle component analysis (PCA) are well suited to ToF-SIMS.
  - Methods describe data using scores and loadings
  - Scores resemble ion images or depth profiles
  - Loadings resemble mass spectra
Principle Component Analysis
Dimensionality reduction

- PCA is an orthogonal linear transformation, which transforms data into vectors that model the variance in the data.

- The PCA components look like mass spectra but with negative values, the negative values are peaks anticorrelated with the component.

- The scores of each component represent the spatial correlation of each component, it is an image of where a particular correlation spectra is located on the sample.

*PCA is a useful statistical technique for ToF-SIMS analysis, but produces somewhat non-intuitive and non-physical results.*
**Principle Component Analysis with Python**

*Dimensionality reduction*

- Data = principle components X score
- The data is provided in a matrix with each row containing a mass spectrum of a particular pixel.
- When the results are visualized, the pixels are just “wrapped” back up into a 2-D array

```python
In [3]:
1  pca = PCA(n_components=13)
2
In [4]:
1  fit = pca.fit(data)
2  pca_score = pca.explained_variance_ratio_
3  print('data fitted')
```

`data fitted`
**PCA**

*Dimensionality reduction*

- We also get an explained variance ratio, the amount of data contained in each principle component.
- The blue line plots this value as a function of principle component number (orange line is the derivative).
- We can see that there are diminishing returns past 8 components.
- Sometimes a interesting chemical feature is found which is not a statistically significant part of the data but shows a new contamination source. (ie a speckle of sodium/potassium from breathing on a sample)

*Explained Variance ratio decreases with each component, this can suggest how many components are necessary to model the dataset accurately.*
Multivariate Curve Resolution with pyMCR
PCA with constraints

• Using MCR is not significantly more difficult to implement than PCA but it often gives better results.
• First it does not produce correlations and anticorrelations, just a spectra of individual components
• It has been implemented for Python: https://pypi.org/project/pyMCR/ with useful constraints (normalization, non negativity ect)
• Solves:
  – $Data = Concentration \times Spectra^T$
• The method requires an “initial guess” of the spectra or Concentration. This can be random, or the results of another algorithm (ie PCA)

MCR is a good choice for ToF-SIMS data.
Overfitting can make multiple components without significant (or meaningful) differences…
MCR results
6 component fit

6 components show most of the features of interest, perhaps this is enough?
**MCR results**

4 component fit

*Underfitting may miss details in the data that are important (ie small features buried in the dataset with unique chemistry)*
MCR results

11 component fit

Underfitting may miss details in the data that are important (ie small features buried in the dataset with unique chemistry)
Binning data

- Introduction
- Getting the data
- Visualizing the Data
- Principle component analysis & Multivariate Curve Resolution
- **Binning data for PCA + MCR**
- Multi-Sample comparison
- Conclusion
Computational constraints

- The size of the data matrix increases the computational time dramatically.
- Since the size mass x pixels, we can use less pixels or less masses.
- Then it is a classic database search problem, find all particles from location (x,y) with mass within the desired range.

```python
def makeMapofMassRange(raw_dat, lowmass, highmass, image_size = (256, 256), scale = 1):
    search_str = '((m_z>={})&(m_z<={}))'.format(lowmass, highmass)
    px_map = [x['px'] for x in raw_dat.where(search_str)]
    px_map = np.bincount(px_map, minlength=image_size[0]*image_size[1])

    if scale != 1:
        image = np.reshape(px_map, (image_size[0], image_size[1]), order='F')
        image = resizebox(image, scale)
        px_map = np.reshape(image, -1)

    return px_map
```

Function to get a map of a specific range
Binning mass spectra
How many bins are enough?

• Binning ToF-SIMS data is necessary for PCA and MCR analysis.
• The original data has a resolution of approximately 1/1000\textsuperscript{th} of an atomic mass unit (amu, about the mass of a proton). Keeping that resolution would require 2 million bins.
• By binning the mass spectra to 1/10 amu, most peaks can be segregated (ie Mg and C\textsubscript{2})
• Binning further speeds up the analysis, but will combine different peaks into one value.
• We can also manually define peaks, but this is time consuming and may neglect peaks that might be hiding in the noise.

Classic problem of processing time vs resolution.
**Reduction spatial resolution**

- For statistical analysis, the data is a matrix of size # mass spectra bins x # of pixels.
- Normally we collect 256x256 pixel images…. But for larger areas we can easily collect 2560x2560 pixel images by stitching multiple images together into a mosaic.
- The size of the data matrix can greatly increase the processing time for MCR, for example a 1000x1000 matrix takes <10 minutes.
- Decreasing resolution is relatively simple, just add the mass spectra of neighboring pixels to define a new, combined pixel.

```python
def resizebox(array, scale):
    scale = int(scale)
    shape = array.shape
    x_pixels = int(shape[0]/scale)
    y_pixels = int(shape[1]/scale)
    reducedimage = np.zeros((x_pixels, y_pixels))
    for newpx_x, x in enumerate(range(0, int(shape[0])/scale)):
        for newpx_y, y in enumerate(range(0, int(shape[1])/scale)):
            reducedimage[newpx_x, newpx_y] = np.sum(array[x:(x+scale), y:(y+scale)])
    return reducedimage
```
Multi-Sample Comparison

- Introduction
- Getting the data
- Visualizing the Data
- Principle component analysis & Multivariate Curve Resolution
- Binning data for PCA + MCR
- **Multi-Sample comparison**
- Conclusion
Multi-sample comparison

• Comparing multiple samples with MCR (or PCA) can identify variations in a process
• Comparable spectra from multiple samples can be selected (such as at an interface as shown)

You can also include data from other complementary measurements to correlate chemical data to electrical, optical… etc.
Conclusion

- Introduction
- Getting the data
- Visualizing the Data
- Principle component analysis & Multivariate Curve Resolution
- Binning data for PCA + MCR
- Multi-Sample comparison
- Conclusion
Python for ToF-SIMS data analysis
Access to state of the art methods

• Cons of using Python for data analysis
  – Learning curve can be significant
  – No graphical user interface (unless you make one yourself)
  – Proprietary data formats can be challenging to import

• Pros:
  – State of the art tools and methods are free, with huge community improving and expanding what’s available.
  – You can automate your workflow, eliminating tedious tasks.
  – Can run on your laptop, server, supercomputer
  – Your options for data analysis is limited only by time and imagination.