Supplement of Atmos. Meas. Tech., 10, 1323–1334, 2017
http://www.atmos-meas-tech.net/10/1323/2017/
doi:10.5194/amt-10-1323-2017-supplement

Atmospheric Measurement Techniques
Open Access
EGU

Supplement of

FATES: a flexible analysis toolkit for the exploration of single-particle mass spectrometer data

Camille M. Sultana et al.

Correspondence to: Kimberly A. Prather (kprather@ucsd.edu)

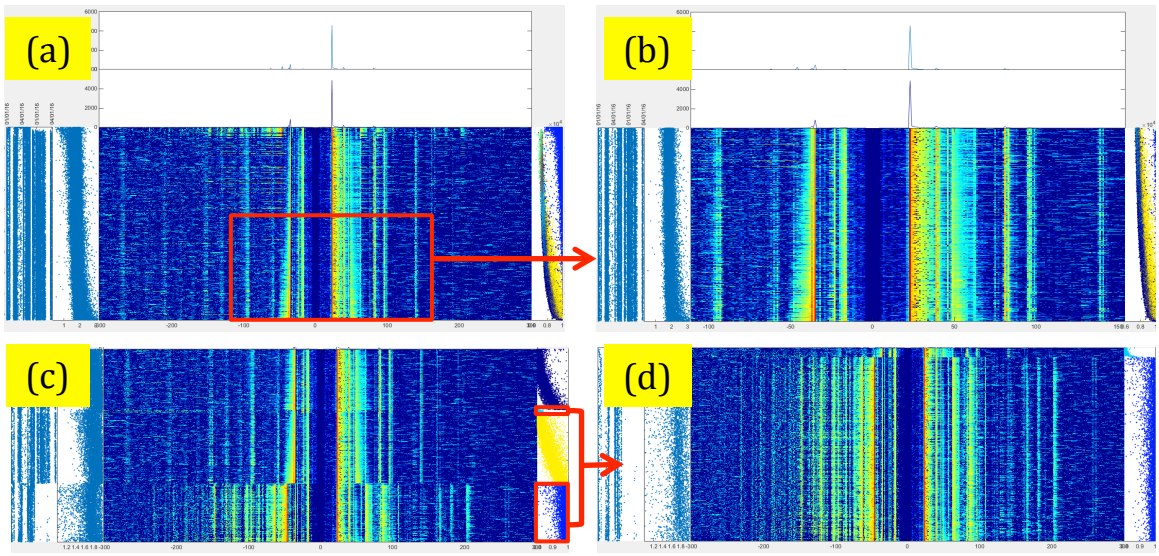


Figure S1. Screen captures of a guiFATES window with the same data as in Figure 1 filtered and sorted in various ways. a) Entire population sorted by m/z -35 peak intensity. b) Zoom in of S1a selecting the bottom half of the particles displayed and a decreased m/z range. c) The data, sorted by cluster, and filtered by size (1-2 um), m/z -35 peak area (0-3000), and clustering statistic (0.8-1). d) The same filtering parameters as S1c with the display limited to clusters 2 and 5.

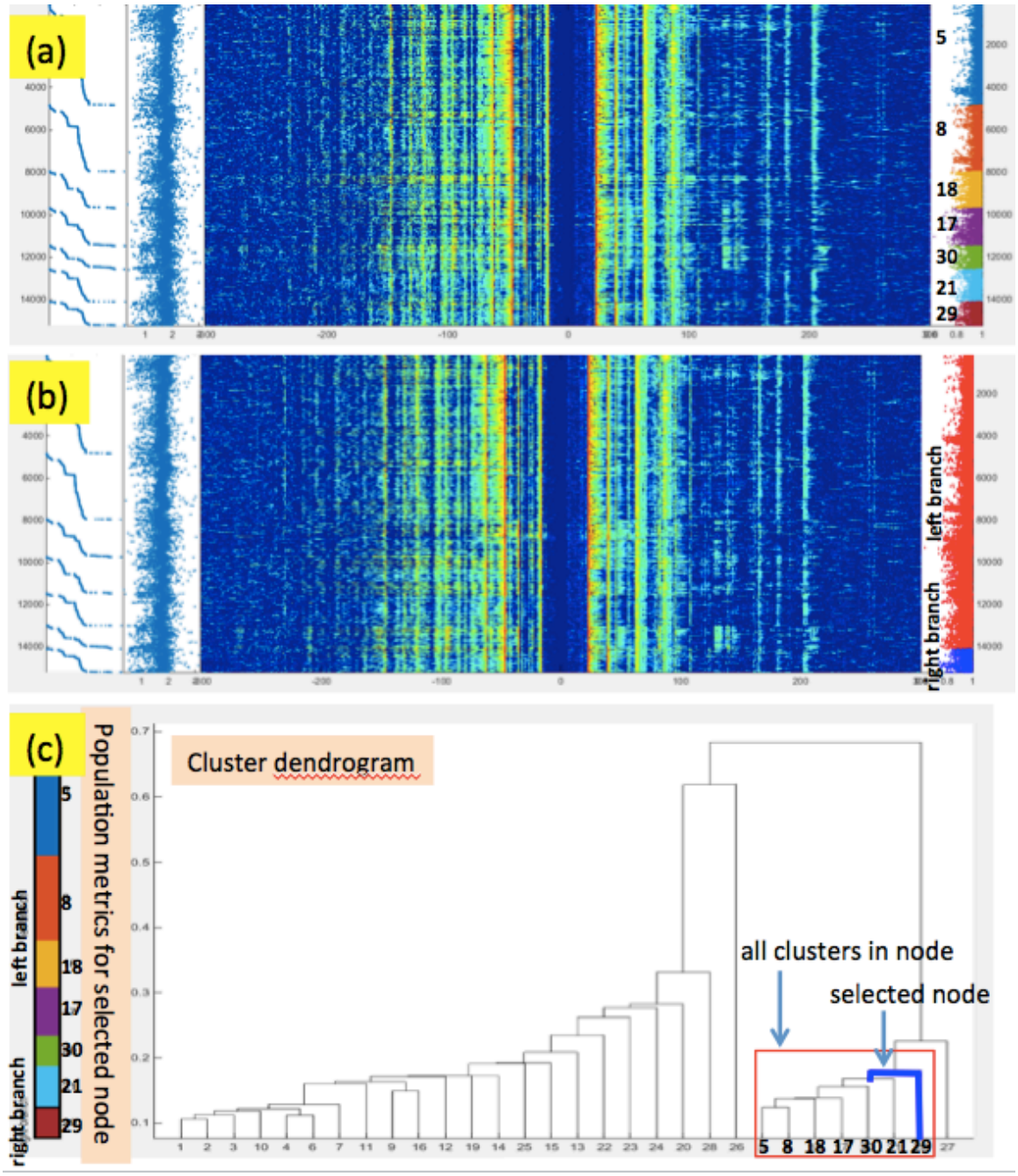


Figure S2. guiFATES windows generated automatically from the selection of a node in Figure 2 (reproduced in Figure S2c). a) The guiFATES window when the user chooses to group particles by their cluster label. The clusters are automatically displayed in the same order as displayed in the dendogram. b) The guiFATES window when the user chooses to group the same data by left or right branch. The clusters are displayed in the same order as displayed in the dendogram.

FATES

A Flexible Analysis Toolkit for the Exploration of Single Particle Mass Spectrometer Data

Reference Manual

Version 1.0

Last Updated February 13, 2017

Copyright Notice

Typographic Conventions

Bold sanserif font is used for **FUNCTION** names. Slanted sanserif font is used for *VARIABLE and FILE NAMES*. Text to be entered by the user are shown in serif font as User input**.**

1 Introduction to FATES

FATES (Flexible Analysis Toolkit for the Exploration of SPMS data), is a MATLAB toolkit easily extensible to an array of single particle mass spectrometer (SPMS) designs and data formats. FATES is provided as free software by the University of California, San Diego. FATES was developed to minimize the computational demands of working with large datasets while still allowing easy maintenance, modification, and utilization by novice programmers. The FATES framework allows users to creatively explore their data without previous assumptions or constraints with simple scripts and by leveraging built-in MATLAB functions. In addition FATES contains an array of data visualization GUIs for calibration of raw data, exploration of the dependence of mass spectra characteristics on size, time, and peak intensity, as well investigations of clustered data sets. Users are requested to cite Sultana et al., 2016 in works for which FATES was used.

This manual gives descriptions of the database architecture, data visualization GUIs, and examples of possible data analyses. Some familiarity with MATLAB is assumed. MATLAB and extensive documentation is available from the Mathworks (www.mathworks.com). **Note that the scripts provided within FATES are extensively documented and commented and users are highly encouraged to read the documentation within each script before utilizing them.**

In order to run FATES, you need MATLAB (version 2014b or later). The most recent version of FATES is available at github.com/CMSultana/FATESmatlabToolKit. This version will be updated with new features and to fix bugs. It is strongly suggested that FATES users utilize a git client and GitHub so that their copy of FATES can be easily synced to the master copy. In addition if a user privately develops FATES features or fixes bugs they are encouraged to submit a pull request to the FATES master copy on GitHub. All pull requests will be evaluated and likely integrated into the master copy for all FATES users to enjoy!! FATES is intended to enhance productivity and collaboration within the SPMS community, so sharing is encouraged!

2 Getting Started

This section demonstrates how to import a small set of provided data into FATES.

In order to run FATES, you need MATLAB (version 2014b or later) installed on your computer. No other programs, software, or drivers are necessary. A basic familiarity with MATLAB is assumed. Excellent and extensive MATLAB documentation is available online from Mathworks (www.mathworks.com).

2.1 Startup FATES

Start MATLAB and make sure that the directory where FATES is stored is within the MATLAB search path. Once the FATES directory has been installed in the search path run **startup_fates**. If this is your first time running FATES **startup_fates** will request the top directories where FATES scripts, user scripts, and user data is stored.

```
>> startup_fates
Main FATES directory?
User program directory?
Enter the high Level directory where your data is:
Number of PEAK matrix rows to read in at a time [1000000]
```

This information will be automatically stored as the variable *startup_fates_info.mat* in the FATES directory, and loaded into the *FATES* structure in the workspace when running **startup_fates** in the future. The main FATES directory and user program directory will be automatically added to the MATLAB search path. To use the default value of 1,000,000 PEAK matrix rows to read in at a time simply press enter. This value is set to limit the memory demands of loading in the spectra data from the external binary file containing all peak information. You may enter a higher or lower value dependent on the memory capabilities of your system.

2.2 Import Data

Approximately 1000 particles of demonstration data provided by the Prather group are distributed with FATES. The demonstration data is within the demoData/raw folder in the FATES directory. Note that these data are not representative of actual ambient aerosol measurements and are provided for demonstration only. All demo data is provided as text files, however with extensions specific to the data type and format within them.

1. inst: Contains information regarding the instrument and experiment conditions for the data contained within the folder
2. set: Contains non-spectra particle information for hit particles (i.e. detected particles that generated spectra)
3. sem: Contains non-spectra particle information for missed particles (i.e. detected particles that did not generate spectra)
4. pkl: Contains spectra (peak) information for all hit particles with the same file name within the folder. The pkl file should only contain relevant data you wish to eventually import into MATLAB. Thus it should not contain the raw spectra output from the SPMS which likely contains an excess of data points and is not calibrated in m/z space, but a "peak-picked" spectra that has also had any baseline, noise, or smoothing algorithms applied and has been converted from time to m/z.

More detail on the creation, format, organization, and data within these files is given in Section 3.

Once all files to be imported into a database have been created and are organized correctly a new FATES study can be created. Creating a FATES study requires two steps.

1. The study must be initialized with **init_study**. **init_study** will request the information from the user concerning study information about where files are located with the prompts or where new files are to be stored. All of the information input by the user is stored within the *STUDY* structure in the workspace and saved to a .mat file using *STUDY.Name*. The *STUDY* variable contains the paths to all the variables necessary to open the study in the future.

```
>> init_study
Study Name?
Path to save study to?
Spectra/peak file format? (cmsPKL2016...)
Raw data directory?
Processed data directory?
What is the unique file identifier for the files containing missed particle data?
(sem, ...)
What is the unique file identifier for the files containing particle data? (set, ...)
What is the unique file identifier for the files containing spectra data? (pkl, ...)
What is the unique file identifier for the files containing instrument/experiment
data? (inst, ...)
INFO, saving study info to test
```

The 'Study Name' should be a short description of the data without spaces (e.g. PacificCoast2015). The 'path to save the study to' is where the STUDY structure containing all other path information and study specific info will be saved. The 'Spectra/peak file format' is the format of the pkl file or other file that contains the processed spectra data. The 'raw data directory' is the top folder where all the raw data files (e.g.: .sem, .set, .pkl, .inst) for the study are held. Once the study is processed the 'raw data directory' will not be referenced again. The 'processed data directory' is where the processed data that are used to create the FATES study are held. The unique file identifiers should be unique part of each data type file name. These are used by MATLAB to identify the appropriate files within the 'raw data directory' to create the FATES study from.

2. Once the study is initialized the user then must run **make_study**. **make_study** can only be run when study information is already loaded into the *STUDY* structure in the workspace. This is done by initializing a new study with **init_study**, or loading in the information (i.e. **load(*STUDY.Name*)**) from a previously initialized study. **make_study** will process all raw data (.inst, .set, .sem, .pkl) within the path *STUDY.RawDir* and store it as the variables necessary for a FATES study in the folder *STUDY.ProcDir*. The following files will be created.

a. *datadef.mat*: A structure containing descriptions of all the variables stored in the INST, PEAK, and PART tables as defined by the user for the study.
b. *DataList.mat*: A cell array containing the list of all the raw data files (.sem, .set, .pkl) used to create the study.
c. *Data_STUDYname.mat*: File containing the processed study data variables (PARTidMAT, PARTdataMAT, etc). This is the file loaded by open_study.
d. *PARTidMissed_STUDYname.bin*: The particle ids and time for all missed particles (i.e. particles that did not generate spectra). This is held in a binary file to minimize size.
e. *PARTdataMissed_STUDYname.bin*: Particle data for all missed particles (i.e. particles that did not generate spectra). This is held in a binary file to minimize size.
f. *PEAKMat_STUDYname.bin*: Data for all spectra generated by all hit particles in the entire study. This is held in a binary file to minimize size.

After **make_study** is complete the workspace is automatically populated with all the data files for the new FATES study. To open a study that has already been initialized (init_study) and processed (make_study) the STUDY structure must be populated with the study file paths (i.e. **load(*STUDY.Name*)**) and then run **open_study**.

3 Raw Data Files

The following section describes the raw data file types and formats that are required to initiate a FATES study using the code as distributed. However, specific functions within the FATES toolkit can easily be edited to render FATES compatible with a wide range of file types and data values and formats. **For more detail on how to alter FATES for compatibility with the data output from your SPMS system go to Section 5.**

3.1 Raw Data File Types

FATES, as distributed, expects all data files to be written as delimited text files with extensions specific to the data type and format with them. Upon initializing a FATES study (**init_study**) the user is requested to define the unique file identifier for the files where raw data is held. MATLAB searches for this file identifier in the filenames. Usually these unique file identifiers would be file extensions. Note each extension should be unique to the raw data type! For example if a file identifier (or extension) of '.txt' is used for missed particle data, then the only .txt files within the top directory should be those for missed particle data. Otherwise an error will be thrown when trying to import the data. It is not advisable to use such generic file identifiers.

> What is the unique file identifier for the files containing missed particle data? (sem, ...)

What is the unique file identifier for the files containing hit particle data? (set, ...)
What is the unique file identifier for the files containing spectra data? (pkl, ...)
What is the unique file identifier for the files instrument/experiment data? (inst, ...)

For the descriptions in the following sections it is assumed that the following file extensions have been specified for each raw data type, though these exact extensions are not required. All information regarding the experiment and instrument conditions for the data contained within the folder are held in an .inst file. Non-spectra particle information for hit and missed particles are held in .set and .sem files respectively. Spectra information for all particles within the folder should be contained in the .pkl file. Note that it is possible that a single file could hold both the particle and spectra information for hit particles. In this case the user would give the same file identifier/extension for the 'hit particle data' and 'spectra data' queries. The details of the file formats required with FATES as distributed is given below.

3.1.1 Instrument and experiment data: .inst file

Below is a screen capture of an example .inst file

```
%*.inst - instrument data file for Elwood with 405nm Scatt LASER, Circa April 26th, 2016

InstCode = ELD
InstDesc = Elwood_405nm_LASER
ExpName = ELD_lp2015_newScatLasers
ExpDesc = laser power experiments with atomized and bubbled control solutions and seawater now using shorter wavelength
scat lasers (405 and 433)
DaCalibFunction = da_noz
% Elwood calibration from Hiroshi with nitrogen, see Access NOZ
%FORM, use for the velocity range 288 - 476 m/s, corresponds to
%size range 200 - 2500 nanometers. Format of Calibration Function:
%"Size = C1 + C2*V + C3*V^2 + C4*V^3 + C5*V^4 + C6*V^5 + C7*EXP(C8*V)"
% [C1 C2 C3 C4 C5 C6 C7 C8 V(min) V(max)]in um
DaCalibParam = [4.115985601842781e+02  -5.462120130945634    0.029264916194310  -7.845628373241157e-05
1.048190224035654e-07   -5.572744448446241e-11        0         0  2.239102001040300e+02  4.892198743321805e+02]
```

An .inst file must be added to every folder where there is particle and peak data to be imported into a FATES study. To add an .inst file to all data folders use **addInst**. Data within the .inst file is read into FATES by **parse_inst**. Within an .inst file the following variables must be defined

InstCode	A code associated with the instrument used to acquire the data (ex: ELD, LVN, TSI, etc.) This should be a very short user defined descriptor that is consistently used within a laboratory working group.
ExpName	The name of the experiment in which data was collected. This should be a short descriptor this is consistently used within a laboratory working group.
DaCalibFunction	Name of the function evaluated which calibrates a particle's aerodynamic diameter as a function of particle velocity. This must be a function within MATLAB's search path. The calibration function distributed with FATES are described in Appendix C.
DaCalibParam	The parameters used in the calibration function. These parameters are derived from performing a size calibration of the instrument using particles with well-defined sizes, usually

polystyrene latex spheres. Raw velocity data can be used to generate calibration parameters for FATES calibration functions using **pslCal**.

In addition a more detailed instrument description (InstDesc) and experiment description (ExpDesc) can also be provided. Any lines preceded by a % are to be read as comments and are ignored by FATES when reading in the .inst file.

3.1.2 Hit particle files: .set

Below is a screen capture of a section of a .set file from a Prather ATOFMS

```
1, 3539, 339.08, 0.0000, 14-JUL-2015 10:46:32, 90, 295, 321, 1433
2, 3597, 333.61, 0.0000, 14-JUL-2015 10:46:34, 42, 222, -1, 0
3, 3283, 365.52, 0.0000, 14-JUL-2015 10:46:35, 43, 123, 312, 1188
4, 3441, 348.74, 0.0000, 14-JUL-2015 10:46:37, 92, 371, 216, 758
5, 3618, 331.67, 0.0000, 14-JUL-2015 10:46:43, 98, 506, 238, 911
6, 3236, 370.83, 0.0000, 14-JUL-2015 10:46:49, 46, 90, 300, 875
7, 3400, 352.94, 0.0000, 14-JUL-2015 10:46:58, 95, 367, 342, 1361
8, 3499, 342.96, 0.0000, 14-JUL-2015 10:46:59, 87, 382, 365, 1119
9, 3366, 356.51, 0.0000, 14-JUL-2015 10:47:00, 1, 0, -1, 0
10, 3375, 355.56, 0.0000, 14-JUL-2015 10:47:01, 46, 173, 166, 530
11, 3512, 341.69, 0.0000, 14-JUL-2015 10:47:02, 90, 454, 358, 1370
12, 3513, 341.59, 0.0000, 14-JUL-2015 10:47:04, 47, 145, 220, 1091
13, 3359, 357.25, 0.0000, 14-JUL-2015 10:47:13, 56, 137, 175, 595
14, 3597, 333.61, 0.0000, 14-JUL-2015 10:47:13, 93, 329, 400, 1633
15, 3577, 335.48, 0.0000, 14-JUL-2015 10:47:14, 43, 178, 327, 1215
16, 3456, 347.22, 0.0000, 14-JUL-2015 10:47:15, 0, 0, 0, 0
17, 3420, 350.88, 0.0000, 14-JUL-2015 10:47:15, 86, 306, 185, 629
18, 3397, 353.25, 0.0000, 14-JUL-2015 10:47:20, 42, 177, 0, 0
19, 3597, 333.61, 0.0000, 14-JUL-2015 10:47:22, 0, 0, 1, 0
20, 3399, 353.05, 0.0000, 14-JUL-2015 10:47:23, 44, 112, 2, 0
21, 3389, 354.09, 0.0000, 14-JUL-2015 10:47:23, 47, 174, 321, 1207
22, 3520, 340.91, 0.0000, 14-JUL-2015 10:47:25, 97, 396, 332, 1039
```

Each row of a .set file contains information for a single hit particle delimited by commas. The .set file is read in by **read_part** and saved to a temporary variable *hitData*. Select columns from *hitData*, defined by **dataFileFields**, are saved into the columns of the PART matrices defined by **studyFields**. Currently only data within the third, fourth, and fifth columns within the .set file are imported into FATES while the rest are ignored. The section below indicates the data that must currently be held in each column of the .set file if utilizing FATES code as distributed. Required data is highlighted in bold. Note that the code to read in alternate file formats can be easily modified and has already been provided for ALABAMA and commercial TSI ATOFMS (Section 5.2).

.set file organzation

Column	*Data Type*	*Details*
3	**Particle Velocity**	Particle velocity to be converted to size.
4	Laser Power	Measured power of desorption/ionization laser pulse (power was not measured in the example .set file above)

5	**Date and Time**	Date and time of particle detection. Format must be able to be parsed by MATLAB.

3.1.3 Missed particle files: .sem

To use FATES as distributed the current formatting requirements for .sem files are the same at the .set files. Each row of a .sem file contains information for a single missed particle delimited by commas. The .sem file is read in by **read_part** and saved to a temporary variable *missedData*. Select columns from *missedData*, defined by **dataFileFields**, are saved in **parse_part** into the external missed particle binary files (*PARTdataMissed_STUDYname.bin* and *PARTidMissed_STUDYname.bin*) with data ordered by **studyFields**. Currently only data within the third, fourth, and fifth columns within the .set file are imported into the binary files while the rest are ignored. Data required in each column of the .sem file are the same as those in the .set file, described above. Note that the code to read in alternate file formats can be easily modified and has already been provided for ALABAMA and commercial TSI ATOFMS (Section 5.2).

3.1.4 Spectra files: .pkl

Below are screen captures of two portions of a .pkl file

```
1, 0, 1, -16.073876, 13.069052, 314, 0, 0.002909
1, 0, 2, -16.259019, 1.726967, 72, 0, 0.000384
1, 0, 3, -17.076725, 5.320297, 125, 0, 0.001184
1, 0, 4, -23.073934, 2.687267, 79, 0, 0.000598
1, 0, 5, -24.102615, 46.752767, 1036, 0, 0.010405
1, 0, 6, -25.119100, 51.473439, 1097, 0, 0.011456
1, 0, 7, -26.109500, 128.915272, 3024, 0, 0.028692
1, 0, 8, -26.250863, 5.627564, 162, 0, 0.001252
1, 0, 9, -26.570327, 5.769587, 114, 0, 0.001284
1, 0, 10, -26.796293, 1.978062, 55, 0, 0.000440
1, 0, 11, -27.035186, 3.352244, 83, 0, 0.000746
1, 0, 12, -27.455799, 8.255209, 112, 0, 0.001837
```

2, 1, 162, 372.001608, 0.423041, 4, 0, 0.000550
2, 1, 163, 378.001169, 0.586345, 4, 0, 0.000762
2, 1, 164, 379.174806, 2.135585, 8, 0, 0.002777
2, 1, 165, 380.189866, 1.229732, 6, 0, 0.001599
2, 1, 166, 382.652999, 2.574627, 7, 0, 0.003347
2, 1, 167, 386.308721, 1.185613, 5, 0, 0.001541
2, 1, 168, 396.287812, 0.436632, 4, 0, 0.000568
2, 1, 169, 418.252261, 0.841120, 6, 0, 0.001094
2, 1, 170, 418.644858, 0.841518, 5, 0, 0.001094
3, 0, 1, -17.067212, 0.323412, 20, 0, 0.001019
3, 0, 2, -18.286708, 0.492278, 18, 0, 0.001551
3, 0, 3, -23.540766, 0.703856, 12, 0, 0.002218
3, 0, 4, -26.074218, 5.021122, 125, 0, 0.015820
3, 0, 5, -35.075697, 6.409853, 123, 0, 0.020196
3, 0, 6, -35.348939, 1.191038, 40, 0, 0.003753
3, 0, 7, -35.857232, 1.102978, 35, 0, 0.003475
3, 0, 8, -36.105821, 0.968408, 37, 0, 0.003051

Each row in a pkl file contains data for a single peak within a particle spectrum delimited by commas. The .pkl file is read in by **read_peak** and saved to a temporary variable *PeakDataTMP*. Select columns from *PeakDataTMP*, defined by **dataFileFields**, are saved into the external binary peak file (*PEAKMat_STUDYname.bin*) with the order defined by **studyFields**. This can be easily modified and the appropriate code to read in ALABAMA and commercial TSI ATOFMS has already been provided (Section 5.3). The section below indicates the data that FATES, as distributed, expects to be held in each column of the .pkl file. Required data is highlighted in bold.

.pkl file organzation

Column	*Data Type*	*Details*
1	**Particle ID**	The particle ID indicates the row in which the matching particle is defined in the .set file. This value should be an integer.
2	**Spectra Type**	Spectra type can either be 0 (negative spectra) or 1 (positive spectra).
3	Peak ID	The peak ID is a unique identifier for each peak within a particle spectra. This value should be an integer.
4	**Peak m/z**	This value can be an integer or floating point number.
5	Peak Area	This value can be an integer or floating point number.
6	Peak Height	This value can be an integer or floating point number.
7	Blowscale Indicator	Indicates whether the raw signal saturated the ion detector and thus the top of the peak was clipped. 1 if peak blows scale, 0 if false.
8	Peak Relative Area	Peak area relative to total area of spectrum. This value can be an integer or floating point number.

The first three columns within the .pkl file create a unique identifier for each peak within a .pkl file. Take the pkl row pasted below

2, 1, 161, 369.520348, 0.421654, 4, 0, 0.000548

This indicates that for the positive spectrum of the second particle listed in the .set file the 161st peak had a m/z of 369.520348. This peak had an absolute area of 0.421654, peak height of 4, did not blow scale, and a relative peak area of 0.000548.

3.2 Raw Data File Organization

Raw data files within FATES must obey the following organization and naming conventions. FATES iteratively searches all folders contained within the raw directory provided during **init_study** for raw data to import into the study**.** FATES only searches for files with extensions specified during **init_study.** Thus there are no restrictions on files that are not these raw files as FATES simply ignores them. Within each folder where particle and spectra data is held there must be exactly one .inst file (see note on file extensions at the beginning of 3.1). FATES will not search for the other raw data files unless an .inst file is present. The number and names of .set and .pkl files within a folder must match. This is because the spectra within .pkl files is generated by the particles listed in the .set files. However there can be unlimited sets of .set and .pkl files within a folder as long as each set is uniquely named. Finally each .sem file within a folder should be uniquely named. It is expected that in each folder with a .set and .pkl pair there will also be a .sem file of the same name, though this is not strictly required.

Examples are given below of acceptable and unacceptable sets of files contained within raw data folders

Acceptable
Example 1: a.pkl, a.sem, a.set, EXP1.inst
Example 2: a.pkl, a.sem, a.set, b.pkl, b.sem, b.set, c.sem EXP1.inst
Example 3: a.pkl, a.sem, a.set, b.pkl, b.sem, b.set, c.set, c.pkl, EXP1.inst

Unacceptable
Example 1: a.pkl, a.sem, a.set, EXP1.inst, EXP2.inst (*more than 1 .inst file*)
Example 2: a.pkl, b.set, a.sem EXP1.inst (*names of .pkl and .set files do not match*)
Example 3: a.pkl, a.set, a.sem, b.set, b.sem, EXP1.inst (*number of .pkl and .set files do not match*)
Example 3: a.pkl, a.set, a.sem, b.set, b.sem, b.pkl, a.set, a.sem, a.pkl, EXP1.inst (*file names are not unique*)

4 Database Structure

Variables to be stored in a FATES database during initialization (init_study and make_study) are defined in **studyFields**. All study information regarding database and variable organization defined in **studyFields** is stored for reference in *DATADEF.mat*. For more details on how the user can easily alter the variables to be saved within a FATES study see Section 5. However, the following section will detail the database structure for FATES as distributed.

4.1 Study information: STUDY structure

The *STUDY* structure is created and modified during initialization (**init_study**) and creation (**make_study**) of a FATES study. The *STUDY* structure mainly contains file paths, names, and extensions concerning the raw and processed data. This structure is utilized by **open_study** to open a FATES study loaded into the workspace and also by several data querying FATES scripts to be able to find external peak and missed data binary files. Descriptions of the *STUDY* fieldnames are given below.

STUDY fields

Field	*Source*	*Description*
Name	**	The name of the FATES study.
NameFull	**	The full path where the STUDY structure is saved.
RawDir	**	The top directory in which raw data for the study is held.
ProcDir	**	The folder where all processed FATES study files will be held.
RawFormat	**	The format for the spectra/peak file to be read in by FATES.
LastInstID	*	The largest InstID used in the study.
NumPkRows	*	The total number of spectral peaks in the study.
NumPkCols	*	The number of peak data fields
missedEXT	**	The extension for the raw missed particle data files
hitEXT	**	The extension for the raw hit particle data files
spectraEXT	**	The extension for the spectra data files
instEXT	**	The extension for the instrument and experiment data files
DataFile	*	The full path for the file that stores all the MATLAB variables for a FATES study

PARTidMissed_filename	*	The full path for the external binary file that stores all the particle id data for the missed particles in the study
PARTdataMissed_filename	*	The full path for the external binary file that stores all the particle data for the missed particles in the study
PeakMat_filename	*	The full path for the external binary file that stores all the spectra data for the study

**Defined by user during init_study
*Generated during study creation

4.2 Instrument and experiment data: INST structure

Descriptive instrument and experiment data contained within the raw .inst files are held within the FATES study workspace in the *INST* structure. Unique instrument operating conditions are assigned unique identifiers, InstID. Each row within the structure contains the following information pertaining to a unique experimental setup. Required fields are shown in bold.

INST fields

Field	*Source*	*Description*
InstID	*	A unique integer associated with each unique InstCODE and ExpNAME pair. Each particle has an InstID as the first part of its identifying information.
InstCODE	.inst	A code associated with the instrument used to acquire the data (e.g. ELD, LVN, TSI, etc.).
InstDESC	.inst	A more detailed description of the instrument used to acquire the data.
ExpNAME	.inst	The name of the experiment in which data was collected.
ExpDESC	.inst	A more detailed description of the experiment in which data was collected.
DAcalibFUNCTION	.inst	Name of the function evaluated which calibrates the particle sizes from velocity. A single InstID may specify multiple DAcalibFUNCTION as an instrument size calibration can change over the course of a single experiment.
DAcalibPARAM	.inst	The parameters used in the calibration function. A single InstID may specify multiple DAcalibPARAM as an instrument size calibration can change over the course of a single experiment.
DAPartID	*	Provides the last PartID associated with each DAcalibFUNCTION and DAcalibPARAM. This is a

		way to track what size calibrations are applied to the particles in the study.
LastPartID	*	The number of particles associated with the InstID.

*Generated during study creation

4.3 Hit Particle Information: PART matrices

Data contained in the .set files is held in a FATES study in the *PARTidMat* and *PARTdataMat* matrices. The .set files are read in with **read_part.** Each row of the matrix holds data for a single particle. The rows in *PARTidMat* and *PARTdataMat* correspond to the same particle. Data within *PARTidMat* is stored in the double-precision format (64 bits). Data within *PARTdataMat* is stored in the single-precision format (32 bits). Splitting up the data this way allows the applicable data to be stored with less precision, therefore saving space in memory. The *PARTidFlds* and *PARTdataFlds* structures contain the variable name for each column in the corresponding matrices. As distributed FATES organizes hit particle data into columns according to the descriptions below. Required columns are shown in bold.

PARTidMat Columns

Data	*Column*	*Source*	*Description*
InstID	1	* & .inst	A unique integer associated with each experiment within a study. Each particle has an InstID as the first part of its identifying information.
PartID	2	*&.pkl	A unique integer used to identify particles which have the same Inst ID.
Time	3	.set	Time of particle detection (contained in .set).

PARTdataMat Columns

Data	*Column*	*Source*	*Description*
Velocity	1	.set	The speed of the particle detected by the light scattering region.
Laser Power	2	.set	Recorded laser pulse energy for each particle
DA	3	*	Aerodynamic size of particle.
Hit	*4*	* or .pol	Integer indicating spectra generated.
Ring	5	* or .pol	Integer indicating if “ringing” detected in spectra
Posit	6	.set	Integer indicating the row of the particle in the raw .set file.

*Generated during study creation

Each InstID and PartID pair generates a unique identifier for each particle within a study.

4.4 Missed Particle Information: PARTmissed external files

Data contained in the .sem files is held in files external to the FATES study workspace. This prevents the memory from being overloaded by data that is rarely utilized. During data importation .sem files are read with **read_part** and then written to two external binary files *PARTdataMissed_STUDYname.bin* and *PARTidMissed_STUDYname.bin.* Data within these files are organized and written to the same precision as specified for the hit particle data as described above. To read in these files into MATLAB use **load_missed**.

4.5 Spectra/Peak Information: PEAK variables and files

Data contained in the .pkl files is held in a binary file external to the FATES study workspace. This prevents the memory from being overloaded as holding the spectra data for an entire study can often be several gigabytes of data. For a study .pkl files are read with **read_peak** and then written to *PEAKMat_STUDYname.bin.* All peak data is written in single-precision format (32 bits). To read in the *PEAKMat_STUDYname.bin* into the *PEAKMat* matrix in the FATES study use **peak_commhelper_script**. Once spectral data is loaded into the *PEAKMat* matrix each row contains data for a single peak. The *PEAKFlds* structure contains the variable name for each column in the *PEAKMat* matrix. As distributed FATES organizes peak data into columns according to the descriptions below. Required columns are shown in bold.

PEAKMat Columns

Data	*Column*	*Source*	*Description*
InstID	1	* & .inst	A unique integer associated with each experiment within a study. Each particle has an InstID as the first part of its identifying information.
PartID	2	* & .pkl	A unique integer used to identify particles which have the same Inst ID.
SpecID	3	.pkl	Either 0 (negative spectra) or 1 (positive spectra) used to indicate polarity of spectra
PeakID	*4*	* & .pkl	A unique integer used to identify peaks within a spectra (same InstID, PartID, and SpecID).
MZ	5	.pkl	Mass to charge value of the peak
Area	6	.pkl	Absolute area of the peak
Height	7	.pkl	Height of the peak
BlowScale	8	.pkl	Indicates whether the raw signal saturated the ion detector and thus the top of the peak was clipped. 1 if peak blows scale, 0 if false.

RelArea 9 .pkl Relative area of the peak
*Generated during study creation

Each InstID, PartID, SpecID, PeakID set is a unique identifier for each peak within a study.

A FATES study also contains a *PEAK* structure. This structure is not intended to be directly used by users, but is a tool utilized by the **peak_commhelper_script** to more quickly index into the external *PEAKMat_STUDYname.bin* file when it is not necessary to read-in the entire file. For this reason this structure should never be altered by the users.

While users can directly import peak data into the workspace using **peak_commhelper_script** as described above, more often users will desire to obtain the data organized in spectra form with peak data at regular intervals. To obtain spectra for a list of particles use **get_int_spectrum_SUM** or **get_NONint_spectrum_SUM**. For more detail on querying peak and spectra data see section 6.

5 Changing the Database Structure

Sections 3 and 4 describe the required raw data file format and FATES database structure for FATES version 01 as distributed. However, FATES was purposely designed to be easily altered to accept a wide variety of data formats and allow a multitude of variables to be stored in the database structure. This section provides details on how to make these alterations within FATES scripts.

5.1 Instrument and experiment data

Different raw file format

Instrument and experiment data contained in .inst file (or other file extension specified during **inst_study**, see Section 3.1) is read into MATLAB using **read_inst**. If the .inst file utilized does not follow the format described in Section 3.1.1 alter **read_inst** to be able to parse the .inst file.

Change database structure

The field names for the *INST* structure (described in Section 4.2) are defined in **studyFields**. Only variables in the .inst file that are defined in **studyFields** will be saved to the *INST* structure in a FATES study. While INST field names can be generally added or removed the following are required and cannot be removed: InstID, InstCODE, ExpNAME, DAcalibFUNCTION, DAcalibPARAM, DAPartID, LastPartID.

5.2 Particle data

Different raw file format

Particle data contained in .sem and .set files (or other file extension specified during **inst_study**, see Section 3.1) is read into MATLAB using **read_part**. If the .sem/.set files utilized do not follow the format described in Section 3.1.2 alter **read_part** to be able to parse in the files. Note that the appropriate code to read in the ALABAMA 'hit' and 'detected' file formats has already been provided and simply needs to be uncommented to utilize. In addition note that the appropriate code to read in the *commercial* TSI ATOFMS .sem and .set file formats has already been provided and simply needs to be uncommented to utilize.

Change database structure

Changing the database structure of the particle information usually consists of altering the variables defined to be saved to the database (**studyFields**) and indicating where data is located in the raw data files (**dataFileFields**). Note that the code appropriate to define ALABAMA and commercial TSI ATOFMS data has already been written into these functions and simply needs to be uncommented to utilize. The variable name to be saved to each column of *PARTdataMat* or *PARTidMat* is defined in **studyFields**. If a variable is not defined in **studyFields** it will not be incorporated into the FATES database. The current variables saved to the PART matrices are described in Section 4.3. Currently *PARTidMat* has three columns containing the following data: INSTID, PARTID, and Time. These cannot be removed from *PARTidMat*, though studyFields can be altered so that more data columns are defined for *PARTidMat*. There are no variables required to be saved to *PARTdataMat*, and variables to be stored in the columns can be removed or added within **studyFields** at the user's discretion.

Once the raw particle data is read into MATLAB via **read_part** it is held in temporary matrices *missedData* and *hitData*. The user defines in **dataFileFields** the particle data held in the columns of *missedData* and *hitData*. For data to be imported from a raw particle file into the FATES database the variable name must be defined in both **studyFields** and **dataFileFields**. However some particle data may not be held in the raw particle files but calculated from data when the study is being created. If a variable is defined in **studyFields** but not contained in the raw particle data file (.sem or .set) then the data must be added explicitly in code in **parse_part**. For example this is currently how particle size (DA) is added to *PARTdataMat*, because it is calculated from particle speed using raw particle velocity and the calibration functions and parameters defined in the .inst file.

5.3 Spectra data

Different raw file format

Spectra data contained in .pkl file (or other file extension specified during **inst_study**, see Section 3.1) is read into MATLAB using **read_peak**. If the .pkl file utilized does not follow the format described in Section 3.1.4 alter **read_peak** to be able to parse in the file. Note that the appropriate code to read in the spectra data

from the ALABAMA 'hit' file format has already been provided and simply needs to be uncommented to utilize. In addition note that the appropriate code to read in the *commercial* TSI ATOFMS .pkl file formats has already been provided and simply needs to be uncommented to utilize.

Change database structure

Changing the database structure of the spectra/peak information usually consists of altering the variables saved to the database (**studyFields**) and indicating where data is located in the raw data files (**dataFileFields**). Note that the code appropriate to define ALABAMA and commercial TSI ATOFMS data has already been written into these functions and simply needs to be uncommented to utilize. The variable name to be saved to each column of *PEAKMat* is defined in **studyFields**. If a variable is not defined in **studyFields** it will not be incorporated into the FATES database. The current variables saved to *PEAKMat* are described in Section 4.5. While PEAKMat variables can be generally added or removed the following are required and cannot be removed: InstID, PartID, SpecID, MZ.

Once the raw spectra data is read into MATLAB via **read_peak** it is held in a temporary matrix *PeakDataTMP*. The user defines in **dataFileFields** the peak data held in the columns of *PeakDataTMP*. For data to be imported from a raw spectra file into the FATES database the variable name must be defined in both **studyFields** and **dataFileFields**.

6 Script Based Analysis

In this section we provide a brief overview of common analyses that can be performed on SMPS data once a study has been initialized. However it should be noted that it is impossible to describe or predict all data analyses and plotting options easily available to FATES users due to the extensive library of built-in and user-developed MATLAB functions. The code utilized to complete the example analyses below are only a possible option for the described analysis. Often the same analysis can be performed using a myriad of script variations. Example code for SPMS data analysis is found within the demoData folder in *exampleCode.m* some parts of which are described in more detail below. Utilizing a combination of logical indexing and functions native to MATLAB, such as **intersect**, **histcounts**, and **find** particles and spectra can be filtered and grouped using any single or combination of particle and mass spectra characteristics (e.g. particle size, peak area at a certain m/z, etc.).

6.1 Querying particle data

The user should search upon the two particles matrices: *PARTidMat* and *PARTdataMat*. These two arrays are the same length (each row corresponds to the same hit particle for each table). *PARTidMat* has three columns: (1) the InstID (2) the PartID (3) and particle time (in MATLAB datenum format).The first two columns act as a unique two-number identifier, often referred to as a PID. The first number, the InstID, corresponds to the instrument and/or experiment on which the particle was

collected. The second number, the PartID, corresponds to the order in which the particle was imported into the study, *for each unique experiment and instrument*, meaning that for each new InstID the PartID will start over from 1.

PARTdataMat has 6 columns (1) Velocity (2) Laser Power (3) Aerodynamic diameter (4) Hit (5) Ring (6) Posit, but these can be expanded or modified as described in section 5.2. Each row corresponds to one particle, and these rows correspond to the rows in PARTidMat. This allows the user to search on either table and then relate it to the other variables in the other table as shown below.

```
% find PIDs of particles with an InstID of 2
secondInstrumentPID = PARTidMat(PARTidMat(:,1) == 2, 1:2);
% find time that particles were collected
secondInstrumentTimes= PARTidMat(PARTidMat(:,1) == 2,
PARTdataMat.Time);
% find aerodynamic diameter of particles
secondInstrumentDa = PARTdataMat(PARTidMat(:,1) == 2, PARTdataMat.DA);
```

Another example to get the indexes, within the *PARTidMat* and *PARTdataMat* tables, for all submicron particles and the particle ids is shown below.

```
submicronIDX = PARTdataMat(:,PARTdataFlds.DA) < 1;
submicronPID = PARTidMat(submicronIDX,1:2);
```

Multiple filters can also be applied within a single line of code. To return the indexes and particle ids for all submicron particles with a laser power greater than 1.5 use

```
submicronHIGHlpIDX = PARTdataMat(:,PARTdataFlds.DA) < 1 &
    PARTdataMat(:,PARTdataFlds.LASERPOWER) > 1.5;
submicronHIGHlpPID = PARTidMat(submicronHIGHlpIDX,1:2);
```

Additionally lists of particles can be compared with the built-in function **intersect**. For example another way to return the particle ids of submicron particles with high laser power would be to

```
% create particle id list of particles with high laser power
HIGHlpIDX = PARTdataMat(:,PARTdataFlds.LASERPOWER) > 1.5;
HIGHlpPID = PARTidMat(HIGHlpPID,1:2);
% generate particle id list of submicron particles with high laser
power
submicronHIGHlpPID = intersect(HIGHlpPID, submicronPID, 'rows');
```

Note that since the tables are the same length and each row corresponds to the data from the single particle, the indexing for one particle or set of particles derived from one table will match the particles from other table. However, this works only if you're using the original tables. It is highly discouraged to alter the rows of the *PARTidMat* or *PARTdataMat* tables for this reason.

6.2 Querying peak data

To find particles with certain peak area characteristics such as peak areas over or below certain thresholds, the user must first call the relevant peak areas into the

workspace so that they can be searched against. To do so, the user should use one of the query scripts to generate a matrix of peak areas for the particles they want to search through. FATES supplies a script to generate both mass spectra at integer values (**get_int_spectrum_SUM**) and at regular non-integer values for higher resolution (**get_NONint_spectrum_SUM**) from the external binary file which holds the peak information for the entire study. Then, use logical indexing to find particles greater or less than your threshold. The user can then return either the indices or PIDs associated with the desired particles as shown below:

```
% return the mass spectra for a list of particles
MaxMZ = 350;
ResponseType = 'Area';
Polarity = 2;
partData = PARTidMat(1:1000,1:2);
[negData, posData] =
get_int_spectrum_SUM(partData,MaxMZ,ResponseType,Polarity);

% find the PIDs for particles in partData with -62 > 250
selectPID = partData(negData(62,:)>250,1:2);
% find the particle indexes for the PIDs in the entire study
[~,selectIDX] = intersect(PARTidMat(:,1:2),peakPID,'rows');
```

The user can expand the filtering by using '&' and '|' modifiers in the logical indexing statements as in below:

```
% get particles with peak areas of -62 OR -46 greater than 250
peakPID = partData(negData(62,:)>250 | negData(46,:) > 250,1:2);
% get particles with peak areas of -62 AND -46 greater than 250
peakPID = partData(negData(62,:)>250 & negData(46,:) > 250,1:2);
```

Note that in addition users can return mass spectra which contain all the peaks stored in the binary file for a list of particles (**get_spectrum**), however it is not straightforward to search peak information from the output of **get_spectrum** as the mass spectra stored in the binary file often do not have the same number of data points or regular spacing.

6.3 Grouping/Binning data

Binning of particles by particle and spectral characteristics, such as binning data based on time, can be accomplished in a single line with the built-in function **histcounts**. For example to bin all submicron particles by time use the set of actions below.

```
% find time of submicron particles
submicronTime = PARTidMat(submicronIDX, PARTidFlds.Time);
% create time bin edges
timeBin = submicronTime(1):1:submicronTime(end);
% Find number of particles in each time bin and the time bin into which
each particle belongs
[timeBinCounts,~,timeBinIDX] = histcounts(submicronTime, timeBin);
```

A similar technique can also be used to find the distribution of *m/z* signal for a set of particles.

```
% get the response at m/z -62 for all particles
neg62 = negData(62,:);
% create bins for m/z 62
neg62bin = min(neg62):100:max(neg62);
% Find number of particles in each response bin and the response bin
into which each particle belongs
[neg62BinCounts,~, neg62BinIDX] = histcounts(neg62, neg62bin);
```

Grouping data based on algorithmic clustering of the spectra is also easily performed. Clustering methods commonly used by the SPMS community such as k-means, hierarchical clustering, and k-medoids are built-in to MATLAB.

```
% cluster spectra with kmeans into ten clusters use
[clusterIDX, clusterSpc] = kmeans([PosSpc' NegSpc'], 10);
% cluster spectra using the ART-2a algorithm in the FATES toolkit
[clusterID, clusterSpc, clusterCounts] = run_art2a(submicronPID, 2, 350,
   0.05, 0.85,  20);
clustIDX = zeros(size(submicronPID,1),1);
for i = 1:length(clusterID)
      [~,tmp] = intersect(submicronPID, clusterID{i}, 'rows');
      clustIDX(tmp) = i;
end
```

The scripts **regroup_art2a** and **match_art2a** which are also commonly utilized in the process of clustering by art2a are also provided.

6.4 Editing Particle Matrices

Due to the flexible nature of MATLAB variables, FATES users have the option of editing the *PARTdataMat* or *PARTidMat* variables. Doing so allows the user to easily associate particle-specific data with each particle. These new columns can then be queried like any of the other variables shown above. Good examples of these are clustering related analytical variables (cluster number or dot product), environmental data (wind speed/direction, relative humidity, etc.), or location (latitude/longitude, altitude). The user can then save this information to the file permanently by overwriting the DataFile. An example script to edit the *PARTdataMat* in order to include cluster number is shown below. Note, it is wise to always save an original unaltered version of the *PARTdataMat* and *PARTidMat* variables as a backup in the case that there is a mistake with your code!

```
% Create new PARTdataMat column for cluster number
PARTdataFlds(7) = 'Cluster' % Edit PARTdataFlds for clarity
for i = 1:length(art2a_outID) % match cluster to particles
    [~,clusterIdx] = intersect(PARTidMat(:,1:2),art2a_outID{i},'rows');
    PARTdataMat(clusterIdx,7) = i;
    clear clusterIdx
end
save(STUDY.DataFile,'INST','PARTdataFlds','PARTdataMat','PARTidFlds','P
ARTidMat','PEAK','PEAKFlds'); % save over old DataFile
```

7 Visual Data Exploration

FATES provides a number of graphical user interfaces (GUIs) to allow users to explore their data using robust data visualizations.

7.1 guiFATES: spectra visualization, grouping, and exploration

Description of guiFATES

guiFATES is a powerful tool for visualizing and exploring single particle data. The myriad plotting options, ability to select parts of your data, and easily substitutable inputs also allows the user flexibility in their data analysis. **guiFATES** has 6 different plotting regions to visualize data. The three horizontal plots visualize mass spectral features, and the three vertical plots are for visualizing any other data associated with the particles such as time, size and cluster relationship data. The bottom of the **guiFATES** windows contains all the display, sorting, filtering, grouping, and output parameters that the user may select and change to interact with and explore the data.

The main panel of the **guiFATES** display is the heat map of the individual particle mass spectra. Each row is an individual mass spectrum with peak intensity indicated by color, with red being high intensity and blue is low. The default display is for the heat map to be linear. However, the radio buttons in the corner can be selected in order to switch from a linear scale to a log scale. Log spacing will make differences in spectra clearer and trends easier to spot.

The top line plot shows the average spectra for all particles displayed in the heat map. The bottom line plot shows the average spectra of a single cluster selected by the user in a listbox at the bottom of the window. The line color in the average cluster spectra plot matches the colors used to indicate the assigned cluster for each particle in the right most vertical scatter plot. Only one cluster can be selected at a time for this plot.

Implementing guiFATES

To call **guiFATES**, the user will need to input the variables in the table below.

Variable	Description
useMZdata	**M** x **N** matrix corresponding to peak area for particles where **M** = # of particles **N** = # of points in spectra. Note that it has been found that the data visualization is most effective when the mass spectra supplied have half integer resolution (as opposed to integer) and are normalized somehow so that all peak intensities are between 0 and 1.
MZvector	Vector of length **N** corresponding to points in spectra. MUST BE MONOTONICALLY INCREASING OR DECREASING IN ORDER FOR GUIfates TO WORK!
partData	**M** x 2 matrix corresponding to InstIDs and PartIDs for each particle.
timeData*	Vector of length **M** corresponding to a select data metric for each

	particle. This data is plotted in the far left vertical panel (A). It is suggested to supply particle time data (in numeric form) as Panel A can display the x axis using date strings.
sizeData*	Vector of length **M** corresponding to a select data metric for each particle. This data is plotted in the second vertical panel to the left (B). It is suggested to supply a cluster relation statistic computed by the user.
clustData*	Vector of length **M** corresponding to the cluster number for each particle.
clustRelation*	Vector of length **M** corresponding to a select data metric for each particle. This data is plotted in the vertical panel on the right (C). It is suggested to supply particle size data or total ion intensity as useful metrics.
inputColors*+	Optional **K** x 3 color matrix where **K** = # of unique clusters.
clustOrdered*+	Optional vector of length **K** corresponding to the order in which to display clusters. Default is in numeric order.

*Optional inputs. Use [ ] if no data supplied.

+ This input is utilized internally by **dendroFATES** and **scatterFATES** when calling **guiFATES**, but is not expected to be leveraged frequently by the user.

Users can choose to substitute the time, size, or cluster relation vectors for the particle parameters of their choice. Some examples are to use environmental variables, like wind speed, wind direction, relative humidity, temperature, solar radiation, or etc.

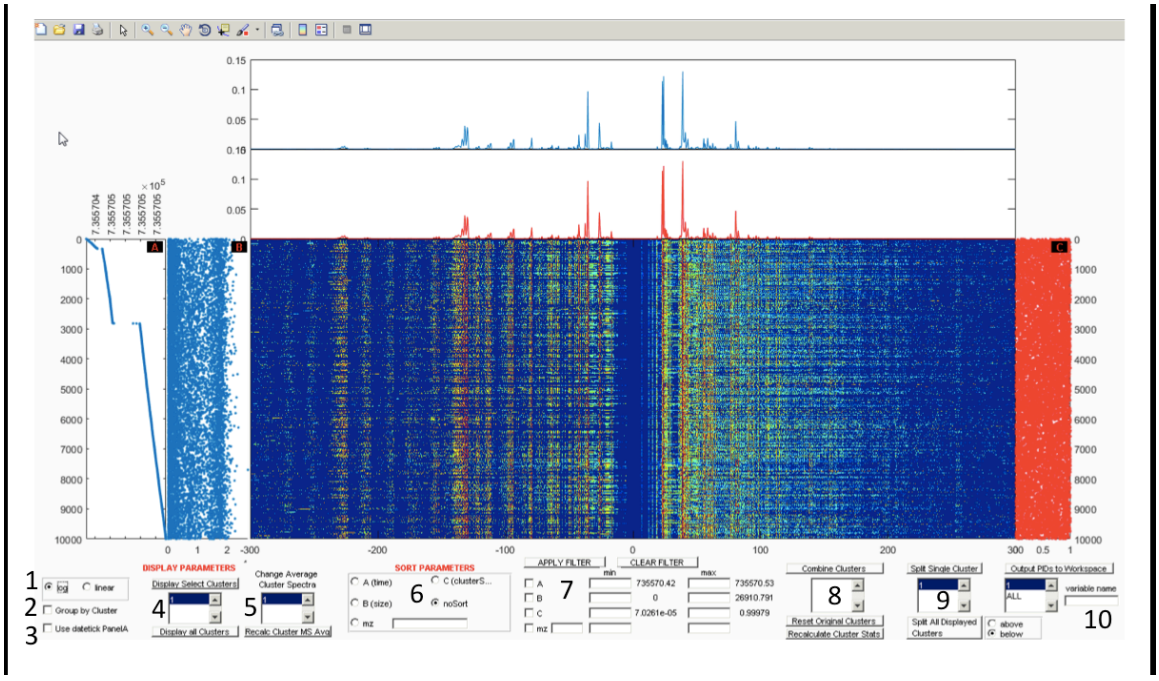
A sample script calling **guiFATES** has been provided below.

```
%% GUIfates Initialization Demo
% Set parameters to get peak areas
MZbins = 0.25:0.5:300.5;
ResponseType = 'RelArea';
Polarity = 2;
partData = PARTidMat(1:10:end,1:2);
% Get peak areas using get_NONint_spectrum_SUM
[negData, posData] =
get_NONint_spectrum_SUM(partData,MZbins,ResponseType,Polarity);
concatenate peak areas for GUIFates initialization--IMPORTANT to do it
this way
useMZdata = [fliplr(negData') zeros(size(negData,2),1) posData']; %
MZvector = -300:0.5:300; %MZ need to be monotonically increasing or
decreasing

% particle data inputs
timeData = PARTidMat(1:10:end,3); % Time data for each particle--CAN BE
SUBSTITUTED FOR A DIFFERENT PARAMETER--one example is to use altitude
data for aircraft studies

clustIDX  % Cluster identifier for each particle output by a clustering
algorithm. See section 6.3 above
```

```
sizeData = PARTdataMat(1:10:end,3); % Size data for each particle--CAN
BE SUBSTITUTED FOR A DIFFERENT PARAMETER--one example is to use total
ion yield
clustRelation = []; % cluster relation statistic

% Call GUIfates
GUIfates(useMZdata,MZvector,partData,timeData,sizeData,clustData,clustR
elation)
```

Interacting with the Data

This section describes the functions and options that **guiFATES** provides for displaying and interacting with the data. Descriptions of the capabilities are provided moving from left to right in the **guiFATES** window.



Zooming and Panning (0)

The user should use the native MATLAB capabilities to zoom in and out and pan the plots created. The x-axis of the horizontal mass spectra plots scales together. The y-axis of the vertical scatter plots and the mass spectra heatmap scales together.

Colormap scale (1)

The users can choose to display the mass spectra heat map (main panel) using either a log10 or linear color scale. The logarithmic scale makes it easier to visually detect relatively small peak intensities in the spectra, while the linear scale helps users visualize absolute differences between peak intensities. The two mass spectra line plots always are displayed utilizing a linear scale.

Group by Cluster (2)

If the 'Group by Cluster' box is checked particles are sorted so that all particles of the same cluster are displayed next to each other.

Displaying date strings (3)
If the input timeData does indeed hold date and time info the user can check the 'Use dateTick PanelA' box so that the tick labels for the x axis on Panel A are displayed using a date string format such as 12-11-16 01:02.

Choose clusters to display (4)
Individual clusters may be selected through the list box underneath the "Clusters to Display" button. Once a cluster has been selected by clicking on it, press the "Clusters to Display" button to replot. Additionally, multiple clusters may be selected and displayed by pressing either "shift" or "ctrl" during cluster selection, and then pressing the "Clusters to Display" button. This plot can be reset to viewing all clusters by pressing the "Display all Clusters" button. No data is lost by choosing to display only a subset of the clusters.

Select average cluster mass spectra displayed (5)
The average cluster mass spectra displayed in the bottom line plot is chosen by selecting from the list of all current clusters in the list box. The cluster color is consistent between the line plot and the colors of the rightmost scatter plot. Note that when cluster populations are altered due to combining (8) or splitting (9) by the user, the average cluster mass spectra are not updated automatically to improve the responsiveness of guiFATES. When the user wishes to recalculate the average mass spectra simply click the 'Recalc Cluster MS Avg' button.

Sort displayed data (6)
Initially, particles are sorted by the order in which they are entered into the GUI. The user can choose to arrange particles based upon any of the metrics displayed in the vertical scatter plots (Panels A, B, C) or a specific m/z. To do so, the user should click on the appropriate radio button in the "Sort Parameters" box. To return to the original order select the "noSort" radio button. If the user selects to sort by a specific m/z the must input the desired m/z in the box to the right of the radio button. The exact value must be in the MZVector supplied as an input to guiFATES.

Filter displayed data (7)
The user can also choose to filter particles based upon the particle size, cluster number, time, or peak area for a specific M/Z. In order to apply one or more of these filters, the user should specify the mix and max range for the specific parameter to be filtered in the "Min" and "Max" text boxes, and then check the box next to it. The available range for each metric is displayed to the right as well. Once all filters have been specified, then press the "Apply Filters" button. Additional filters can be applied by specifying another parameter and then clicking the "Apply Filters" button again. To undo the sorting, uncheck the boxes and press the "Clear Filter" button again.

Combine clusters (8)
To combine current clusters into a single cluster the user should list all desired cluster to combine in the box under the "Combine Clusters" button. Pressing the "Combine Clusters" button will merge all the listed clusters into a single cluster

labeled with the lowest listed identifier. For example, combining clusters 1, 2 and 5 will result in a single cluster labeled 1. The user can return to the original cluster list input into guiFATES by pressing the "Reset Original Clusters" button below the "Combine Clusters" button.

Create a new cluster (9)
GUIfates provides the option to split a current cluster (Ccluster) into two or more new clusters. To do so, the user should select the "Group by Cluster" option or alternatively select Ccluster using the "Clusters to Display" list box and button. Then the user should organize Ccluster as they desire using the appropriate sorting and filtering metrics. Then, select Ccluster in the list box below the "Split Single Cluster" button. When the "Split Single Cluster" button is pressed then a crosshairs will come up and the user can choose where in the y axis to split the cluster. By clicking the mouse button, the user will then split the selected cluster in to two new clusters. The top fraction will remain as the original cluster, while the newly formed cluster will be given an identifier one greater than the highest current cluster identifier in the GUIfates window.

Users can also multiple clusters at once using a similar procedure. This is most effective when the 'Group by Cluster' box is unchecked. Then the user should organize the data as they desire using the appropriate sorting and filtering metrics. When the "Split All Displayed Clusters" button is pressed then a crosshairs will come up and the user can choose where in the y axis to split the displayed data. By clicking the mouse button, the user will then create a new cluster at the location of the curser on the y axis. If the below radio button is selected all data below the cursor will be grouped into a single cluster when the user clicks the mouse button and all data above the cursor will have their cluster identifiers preserved. If the above radio button is selected all data above the cursor will be grouped into a single cluster when the user clicks the mouse button and all data below the cursor will have their cluster identifiers preserved. The user can return to the original cluster list input into guiFATES by pressing the "Reset Original Clusters" button below the "Combine Clusters" button.

Output clusters lists to workspace (10)
Users can choose output any of the current clusters to the workspace. To do so, choose the desired cluster/s in the list box in the bottom right corner, below the button that says "Output PIDs to the Workspace." To the right of this list box, there is a textbox titled "variable name." Put the desired name of the variable in here, and then press the "Output PIDs to Workspace" button. If only a single cluster is selected the new variable will be a two column double corresponding to the instID and partID for the particles in the selected cluster, will be created in the workspace. If multiple clusters are selected to ouput the new variable will be a cell array with the top row containing the PID list for each selected cluster and the bottom row is the cluster identifier.

7.2 dendroFATES: hierarchical cluster relations

Description of dendroFATES

dendroFATES sets up a dendrogram using relations calculated from input representative cluster mass spectra. The clusters are then automatically linked by a hierarchical analysis performed within MATLAB, which is displayed in a dendrogram in the main panel of the **dendroFATES** GUI window. The dendrogram links clusters in a binary fashion creating new groups which are then further linked. Low linkage heights indicate a high degree of similarity between the left and right branch and larger linkage heights between levels in the dendrogram are indicative of natural divisions in the dataset. The user can then interact with the GUI and select nodes/branches to examine the particles in each cluster in **guiFATES** or output the particle ids to the workspace. The bottom of the **dendroFATES** window contains all the selection and output parameters that the user may select and change to interact with and explore the data. When a linkage is selected the fractional cluster contribution to the selected node is displayed on the right in the vertical panel in the **dendroFATES** window and the fraction of the selected node to the total population is also displayed in text.

Implementing dendroFATES

To call **dendroFATES**, the user will need to input the variables in the table below.

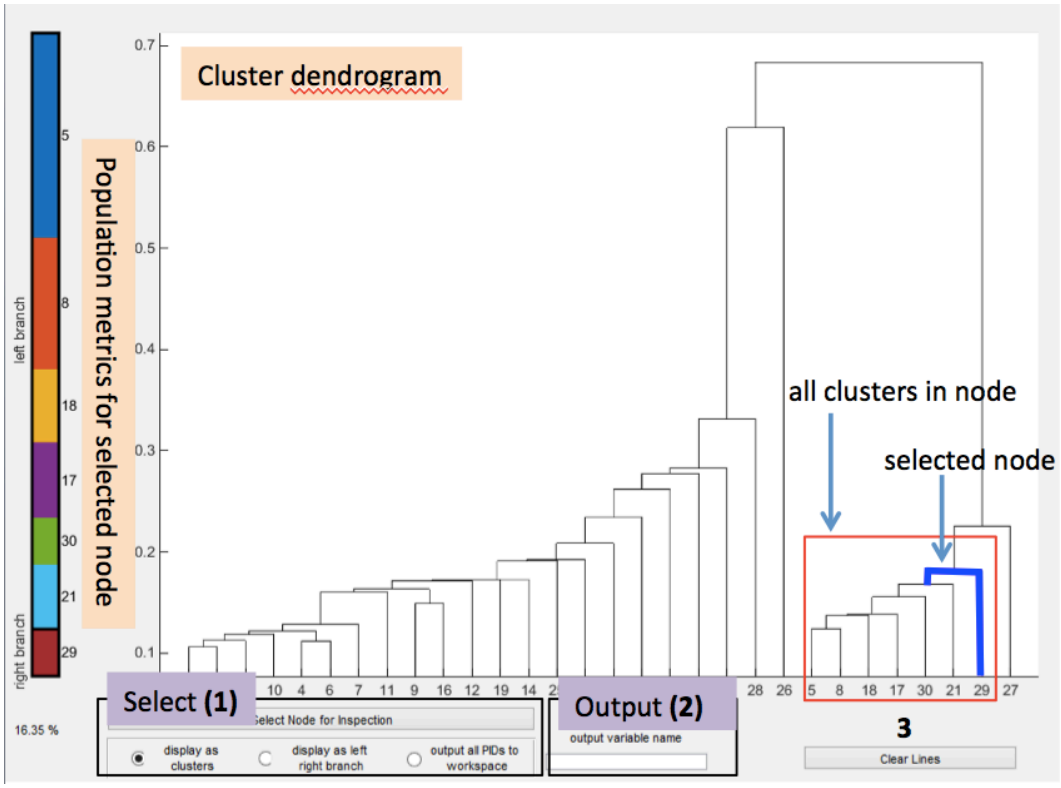
Variable	Description
outWM	A **J** x **K** matrix containing K average cluster mass spectra. The spectra are used to relate the K clusters by hierarchical clustering. outWM would usually be the output from art2a or some other clustering algorithm.
clustLabels	A vector of length **K** with each entry corresponding to the cluster label for each column of outWM. If clusterLabels is empty then the vector labels will just be 1:K.
clustData	Vector of length **M** corresponding to the cluster number for each particle. The cluster identifiers in clustData should correspond to the cluster numbers listed in clustLabels.
partData	**M** x 2 matrix corresponding to InstIDs and PartIDs for each particle.
useMZdata*	**M** x **N** matrix corresponding to peak area for particles where **M** = # of particles **N** = # of points in spectra. Note that it has been found that the data visualization is most effective when the mass spectra supplied have half integer resolution (as opposed to integer) and are normalized somehow so that all peak intensities are between 0 and 1.
MZvector*	Vector of length **N** corresponding to points in spectra. MUST BE MONOTONICALLY INCREASING OR DECREASING IN ORDER FOR GUIfates TO WORK!
timeData*	Vector of length **M** corresponding to a select data metric for each particle. This data is plotted in the far left vertical panel (A). It is

	suggested to supply particle time data (in numeric form) as Panel A can display the x axis using date strings.
sizeData*	Vector of length **M** corresponding to a select data metric for each particle. This data is plotted in the second vertical panel to the left (B). It is suggested to supply a cluster relation statistic computed by the user.
clustRelation*	Vector of length **M** corresponding to a select data metric for each particle. This data is plotted in the vertical panel on the right (C). It is suggested to supply particle size data or total ion intensity as useful metrics.

*Optional inputs if want to use in coordination with **guiFATES**. Use [ ] if no data supplied.

Interacting with the Data

This section describes the functions and options that **dendroFATES** provides for displaying and interacting with the data.



The user can select nodes (1) to either explore the data in **guiFATES** or output to the current workspace. To do so the user clicks the "Select Node for Inspection" button. A cursor then appears in the dendrogram and the user clicks the near the tee on the horizontal line for the node they wish to inspect. If the nodes are close together the user can zoom in and out of the dendrogram using MATLAB's built-in functionality. This way the user can supply **dendroFATES** with hundreds of clusters and still explore the cluster tree quickly without having to set new parameters. If the

"display as clusters" radio button is selected the node is highlighted in the dendrogram and the fraction of the selected node to the total population is also displayed in text. On the right in the vertical panel the fractional cluster contribution to the selected node is displayed with the left and right branches labeled and outlined by thick black boxes. In addition a **guiFATES** window is generated populated by all clusters contributing to the selected node and displayed in the same order and with the same color scheme as in the dendrogram. This assists intuitive visual comparisons and combinations of data. If the "display at left/right branch" radio button is selected a similar sequence executes except data is displayed only grouped by either left (red) or right (blue) branch. Finally if the "output all PIDS to workspace" radio button is selected the entire tree belonging to the selected node will be highlighted and a new variable is output to the current workspace To the right of this radio button, there is a textbox titled "output variable name" (2). Put the desired name of the variable in here before depressing the "Select Nodes for Inspection" button. The output variable will be 1x2 cell array. The first cell contains the PID list for all particles in the selected node. The second cell contains the cluster assignment for all particles, in the same order as the first cell, in the selected node. If the user wishes to clear all the highlights added to the dendrogram they press the "Clear Lines" button (3).

7.3 scatterFATES: user defined particle relations

Description of scatterFATES

The main panel of the **scatterFATES** window displays a scatter plot using particle metrics input by the user with each point representing a single particle and color coded by cluster identifier. A legend of cluster labels is also displayed. The user can then interact with the GUI and select regions of the scatter plot to examine the particles in **guiFATES** or output the particle ids to the workspace.

Implementing scatterFATES

To call **scatterFATES**, the user will need to input the variables in the table below.

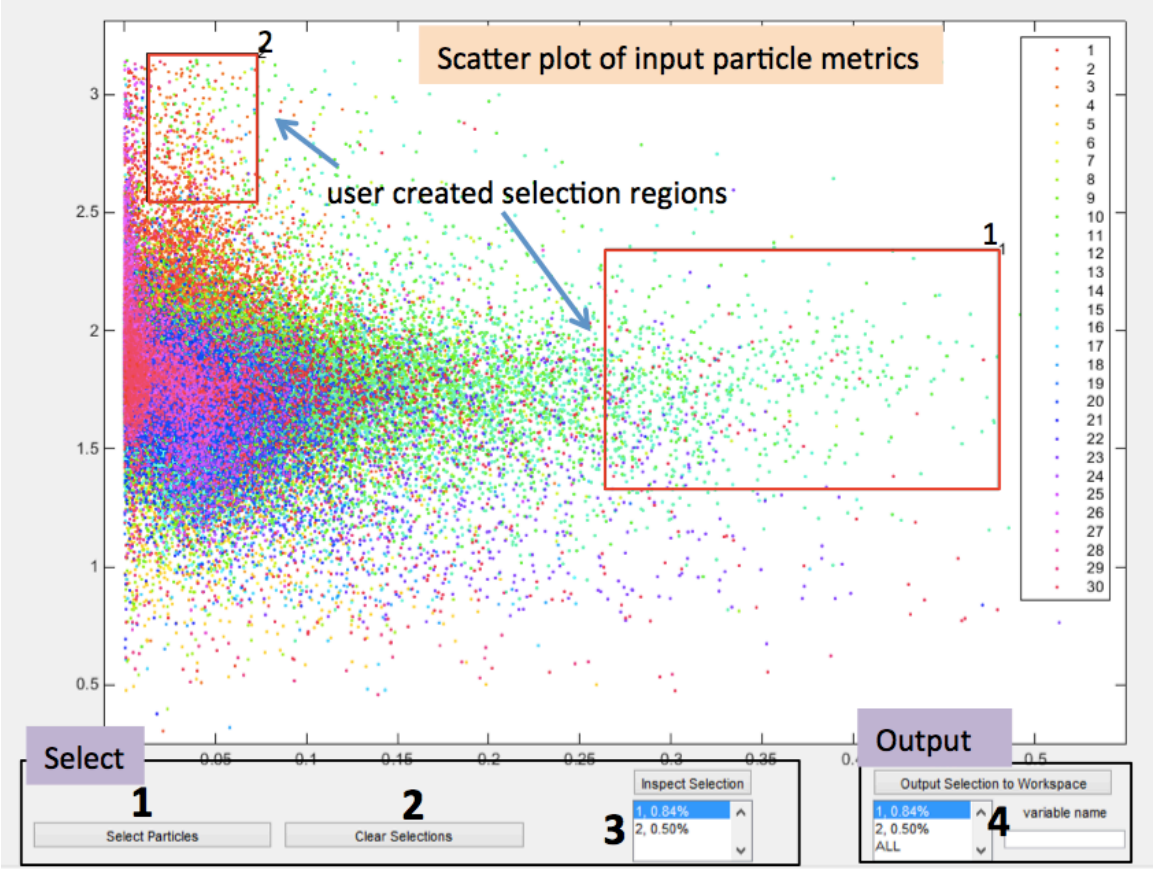
Variable	Description
XVector	Vector of length **M** corresponding to a select data metric for each particle.
YVector	Vector of length **M** corresponding to a select data metric for each particle.
CVector	Vector of length **M** corresponding to the cluster number for each particle. The cluster identifiers in
partData	**M** x 2 matrix corresponding to InstIDs and PartIDs for each particle.
useMZdata*	**M** x **N** matrix corresponding to peak area for particles where **M** = # of particles **N** = # of points in spectra. Note that it has been found that the data visualization is most effective when the mass spectra supplied have half integer resolution (as opposed to integer) and are normalized somehow so that all peak intensities are between 0

	and 1.
MZvector*	Vector of length **N** corresponding to points in spectra. MUST BE MONOTONICALLY INCREASING OR DECREASING IN ORDER FOR GUIfates TO WORK!
timeData*	Vector of length **M** corresponding to a select data metric for each particle. This data is plotted in the far left vertical panel (A). It is suggested to supply particle time data (in numeric form) as Panel A can display the x axis using date strings.
sizeData*	Vector of length **M** corresponding to a select data metric for each particle. This data is plotted in the second vertical panel to the left (B). It is suggested to supply a cluster relation statistic computed by the user.
clustRelation*	Vector of length **M** corresponding to a select data metric for each particle. This data is plotted in the vertical panel on the right (C). It is suggested to supply particle size data or total ion intensity as useful metrics.

*Optional inputs if want to use in coordination with **guiFATES**. Use [ ] if no data supplied.

Interacting with the Data

This section describes the functions and options that **scatterFATES** provides for displaying and interacting with the data.



Selecting Data (1)
The user can select regions of the scatter plot by pressing the "Select Particles" button. A cursor then appears on the scatter plot and the user then clicks two locations on the plot which describe the corners of a rectangle. A rectangle highlighting the selected region is then drawn labeled with a selection counter. In addition the selection label appears in the "Inspect Selection" (3) and "Output Selection" (4) listboxes. The fractional contribution of the selected region to the entire input population is also displayed in the listboxes. Selection regions can overlap.

Clearing Selections (2)
The user can clear all the selection rectangles in the scatter plot by clicking the "Clear Selections" button. This also clears all selections from the listboxes.

Inspecting Data (3)
The user can inspect the particle data within any selected region by selecting a region in the listbox below the "Inspect Selection" button and then clicking the button. A **guiFATES** window populated by the particle data in the selection region is then created using the same cluster color scheme and labels in the **scatterFATES** window.

Outputing Data (4)
The user can inspect the particle data within any selected region by selecting a region in the listbox below the "Inspect Selection" button and then clicking the button. A **guiFATES** window populated by the particle data in the selection region is then created using the same cluster color scheme and labels in the **scatterFATES** window. Finally if the "Output Selection to Workspace" button is selected the PIDs in the region selected in the listbox below is output to a new variable to the current workspace. The output variable is named according to the label input into the variable name textbox. If a single region is selected the output variable will be 1x4 cell array. The first cell contains the PID list for all particles in the selected node. The second cell contains the cluster assignment for all particles, in the same order as the first cell, in the selected node. The third cell is the xlimits for the rectangle that made up the selection, and the fourth row is the ylimits. If the user chooses to output 'ALL' regions from the listbox the output variable is a Mx4 cell array with each column containing the output data for each selection region.

APPENDIX

A. Merging Studies

One of the features of FATES is the ability to add data to an existing study without having to reinitialize or recreate the entire dataset. This is especially helpful with ongoing field studies or experiments where the existing dataset is very large and reinitializing frequently would take a considerable amount of time. Adding data to an existing study requires three steps described below.

1. FATES adds data to an existing dataset by merging existing studies. Therefore two studies must have already been created via the init_study/make_study sequence described in Section 2.2. Study1 should be the large preexisting dataset. Study2 should be created (separate from Study1) containing the new data that eventually will be added to Study1 in a new study (Study3).
2. After both studies to be merged are created the user should run **merge_study_load.** To merge, Study1 and Study2 must have the same *PEAKFlds*, and thus the same organization of the *PEAKMat* matrix. However *PARTidFlds*, *PARTdataFlds*, and the field names for the *INST* structures may vary between the two studies. When running **merge_study_load** the user will be asked study names and directories for creation of the new merged study

```
>> merge_study_load
Path to First Study file?
Path to Second Study file?
Name for new study?
Path to save new study to?
Processed data directory?
```

merge_study_load begins the process of creating a new study (Study3) containing both Study1 and Study2 data. To speed the creation of Study3 the larger dataset should be entered as the first study and the smaller dataset should be entered as the second study in **merge_study_load**.

3. To complete the creation of a new study (Study3) containing both Study1 and Study2 data the user then must run either **merge_study_inst** or **merge_study_part** after **merge_study_load**. The user must select which function to use based upon how the InstIDs and PartIDs for the two studies should be merged.
    a. **merge_study_inst:** Use to merge studies if only InstIDs in study2 and not PartIDs should be changed. Call as **merge_study_inst(0)** if all InstIDs and therefore experiments in both studies are unique and can simply append data without rewriting anything. Call as **merge_study_inst(1)** if there are overlapping instids (but not actually overlapping experiments) and new InstIDs need to be made for Study2.
    b. **merge_study_part:** Use to merge studies if Study1 and Study2 contain data from the same experiment and therefore the PartIDs of Study2 need to be altered and potentially the InstIDs. Call as **merge_study_part(inst1_set,inst2_set).** inst1_set and inst2_set are the Study1 InstIDs and Study2 InstIDs that match to the same experiment and therefore should have the same InstID. This program will go thru each pair and change InstIDs and PartIDs in Study2 to follow the last PartID for Study1.

B. CalibFATES: Mass Calibration of Raw TOF Data

calibFATES was developed to quickly view and calibrate spectra within the native MATLAB environment. To initiate the GUI run the command **calibFATES.** Once the GUI has initialized, the user will be asked to navigate to the folder with raw mass spectra files and select a spectrum. The current configuration of the GUI will look for .ams or .amz files. A function to read this file format from binary into MATLAB is included with **calibFATES**. Additional file formats will require additional functions to read them into MATLAB and can easily be incorporated into the **loadSpectra** function within **calibFATES** (see detailed comments in subfunction **loadSpectra**). Once a spectrum has been selected and read into MATLAB, it will be displayed in the spectrum plot area.

When first loaded, the spectrum will be displayed in its raw, uncalibrated form. The x-axis will be set from 1 to the default number of spectrum data points. The default number of points is 15,000 though this may be changed within the *calibFATES* source code by changing the NumPoints variable. By pressing the "Toggle Calibration" button, the user may change the X-axis from the time domain to the mass-to-charge (m/z) domain. Pressing the button again will reverse this change. Mass calibration is discussed more below.

The displayed spectrum can be interacted with in order to better discern its features. The view may be changed by utilizing MATLAB's native toolbar zoom functionalities in addition to using options in the View menu. The user can choose to reset the view to the full width and full height of the spectrum with the *Reset* command (Ctrl+R). The X- and Y-axis limits can be manually set by the user with the *Set-X* and *Set-Y* commands (Ctrl-X and Ctrl-Y respectively). Selecting either of these options opens a text-box prompting for the desired minimum value of the new range. Once the user has entered a value for the minimum value, there will be a text-box prompting the maximum value. Once both values have been input then the appropriate axis will be re-scaled to the new values.

To find the value of a point within the spectrum the user can utilize the native data cursor functionality as well as a specific the *Get Point* (Ctrl+W) function in the Calibration menu. Once activated, the mouse cursor will be replaced by a crosshairs. The crosshairs can then be placed over the plot to find a point to "get." Pressing either mouse button will select the current point. This will display the X-value, X-value index, and Y-value for that coordinate in the box below the plotting region.

Navigation

calibFATES makes it easy to view positive and/or negative spectra within a given folder. To view positive spectra, the checkbox labeled "Positive Spectra" in the bottom right corner must be checked. Likewise, the checkbox labeled "Negative Spectra" must be checked in order to view negative spectra. At least one of these boxes must be checked before the user can navigate between spectra. Navigation between spectra can be accomplished through the toolbar buttons ( and) or with the keyboard arrow buttons. When navigating between spectra the x-axis will

remain fixed in the time domain and the y-axis will rescale to the points currently displayed. The user can also choose to open spectra from another folder. This option, *Open Spectrum* (Ctrl+O) is found under the File menu. Loading it will open a file prompt for spectra. The user can navigate to the appropriate file location and select spectra based upon their chosen file format. The new spectrum will be loaded and displayed in the plot area.

Create Calibration

calibFATES is initially loaded with no mass calibration. Default values of 2.0E-6 and 0 are loaded for the calibration parameters. The user may choose to manually set the calibration values, to load a previous calibration file, or to complete a new calibration. If the calibration is used (i.e. the "Toggle Calibration" button has been toggled, then the spectra is plotted against *m/z* instead of time. *m/z* are calculated according to the following relationship: $m/z = A(t-B)^2$ where A and B are calibration parameters (slope and intercept) and t is the timepoint.

These calibration parameters can be manually saved and loaded by the use of .pcal and .ncal files. To save the currently displayed values, select *Save Calibration* under the Calibration menu (Ctrl+S). These will save the parameters in the appropriate polarity calibration file of the user's choice (i.e. if the current spectrum is positive, then it will save it to a .pcal file and if it is negative then it will save it to a .ncal file). These same parameters can then be loaded by *Load Calibration*, also under the Calibration menu (Ctrl+L). Selecting this option will load the calibration parameters from a user-selected file. It will only consider calibration files for the polarity of the current spectrum.

The calibration parameters can also be defined by the user manually. The *Set Calibration Values* function under the Calibration menu (Ctrl+B) will bring up text prompts for the calibration parameters A and B.

Mass Calibration

One of the primary features of **calibFATES** is the ability to generate mass calibrations of loaded spectra. The calibration process uses data entered into the table on the right side of the figure. The data needed are timepoints and the calibrated *m/z*. This data can be loaded into this column manually or through the *Add Point* function under the Calibration menu (Ctrl+D). The add point will bring up a crosshairs which will, upon button click, enter the corresponding timepoint and a user-determined m/z into the calibration data table. As with the *Get Point* function described previously, these coordinates will display below the plot area. Up to 20 points, from any number of spectra can be incorporated into the calibration. To clear the values in the table press the "Clear" button. Once the desired values have been entered into the table, the user should ensure that the appropriate polarity radio button is selected (i.e. positive if calibrating positive spectra and negative if calibrating negative spectra). At any time to apply the calibration calculated from the points in the table to the mass spectra press the "Calibrate" button. The user should press the "Final Calibrate" button to bring up a file prompt to save the calibration file in a

particular location and with a particular name. Calibration files saved this way will also include the timepoints, user-defined m/z and the particle filename and location for each calibration point. Despite this extra information, these calibration files can be loaded like any other calibration file.

C. Velocity to Aerodynamic Size Calibration

Within FATES the function **pslCal** is provided to generate size calibration parameters for the **da_noz** function, specified in the .inst file, utilized to calculate aerodynamic particle size from particle velocities. To use pslCal the user must have run a set of aerosols of controlled size, usually polystyrene latex spheres (PSLs) on the SPMS. In addition the PSL data needs to be already imported into MATLAB via a FATES study. To import the data a placeholder DaCalibFunction can be used in the .inst file.

To call **pslCal**, the user will need to input the variables in the table below.

Variable	Description
pslTime	**N** x 2 matrix specifying the start and stop times, in datenum format, for each PSL size run. Start times are in column 1 and stop times are in column 2.
pslVels	**N** x 2 matrix specifying the min (column 1) and max (column 2) velocities to be considered for each PSL size run. This variable is used to eliminate any spurious data from being considered in the speed distribution of each PSL size, which should be very tight.
pslSizes	A vector of length **N** that specifies the size of each PSL used in the calibration. Note the **N**th entry of pslSizes corresponds to the **N**th row of pslVels and pslTime.
N	Indicates the order of the polynomial to be fit (default is 3)
bins	Indicates the number of size bins (default is 100) to group the PSLs by.

pslCal outputs two variables described below.

Variable	Description
calParams	Size calibration parameters for the **da_noz** function.
speedModes	The center of the distribution for each PSL size in speed.

D. Writing PKL from AMS or AMZ raw data files

The FATEs toolkit also provides a set of scripts to write .pkl files in the format described in 3.1.4 from .ams (**redopkls**) and .amz (**redopkls_AMZ**) files. To run these scripts the user supplies the top directory containing raw data that needs to be written to .pkl files and the mass calibration parameters for the contained data. Proper functioning of **redopkls** depends upon **get_spectrumAMS**, which extracts .ams file data into MATLAB, and a script to calculate the baseline and remove noise from the raw data and then peak picks and to be written into the .pkl file. The Prather group has developed two different scripts which have been empirically optimized to perform this set of operations for the linear reflectron TOF-MS (**PeakList_gen_LVNFinal**) and Z-TOF-MS (**PeakList_gen_SLYFinal**) utilized in the Prather lab. These scripts are provided within the FATEs toolkit however users should complete thorough validation checks if they choose to use them to verify that these **PeakList_gen*** scripts perform as desired with their raw data. Users should test against a variety of their own raw mass spectra with a wide range of spectral features, anomalies, and intensities.