CIUSuite 3 Manual

By Chae Kyung Jeon, Carolina Rojas Ramirez, and Devin Makey

Table of Contents I

Each link might indicate one slide only or indicate the first slide in a group.

Table of Contents II

Table of Contents III

Citation Information

CIUSuite2 publication:

Polasky, D. A., Dixit, S. M., Fantin, S. M., & Ruotolo, B. T. (2019). CIUSuite 2: Next-Generation Software for the Analysis of Gas-Phase Protein Unfolding Data. Analytical Chemistry, 91(4), 3147–3155. https://doi.org/10.1021/acs.analchem.8b05762

Publications that use the multi-state classification method should also/alternatively cite:

Polasky, D. A., Dixit, S. M., Vallejo, D. D., Kulju, K. D., & Ruotolo, B. T. (2019). An Algorithm for Building Multi-State Classifiers Based on Collision-Induced Unfolding Data. Analytical Chemistry, 91(16), 10407–10412. https://doi.org/10.1021/acs.analchem.9b02650

A Note on Terminology

- The terms “drift time” for the y-axis and “collision voltage” for the x-axis of a CIU fingerprint are used throughout this manual. However, analyses are generally independent of the nature of the axes, with a few caveats:
  - The ion mobility data (dependent variable) should always be on the y-axis (but it can be arrival time, drift time, CCS, elution voltage, etc.)
  - The x-axis should be the independent variable. However, it can be just about anything that causes a change in observed IM (collision voltage, source/cone voltage, source temperature, solution temperature, laser energy, etc.)

Installation and Requirements

- Installation:
  - Download **CIUSuite3_Setup.exe** from https://github.com/RuotoloLab/CIUSuite3/releases
- Requirements (installer/executable version)
  - Windows
    - Tested primarily on Windows 10 and 11.
  - No other requirements
    - Setup will install a local Python interpreter and all required packages and resources.

Source Code

- Source Code
  - Source can be found at https://github.com/ruotololab/CIUSuite3
  - To run:
    - Clone repository above to download source and resource files
      - All source and resource files must be kept in the original folder configuration (or modified within the code)
    - Run CIU2_Main.py with Python (see below for package requirements)
- Requirements (Building from source)
  - Python 3.5+
    - Original CIUSuite 2 built with Python 3.5.3. (Should work on any 3.5+)
    - Version 2.3 update built with Python 3.7.9
    - CIUSuite3 has been tested on Python 3.7 and above
  - Packages:
    - Numpy, scipy, matplotlib, scikit-learn, LMfit, pygubu, PyQt5, peakutils, pandas

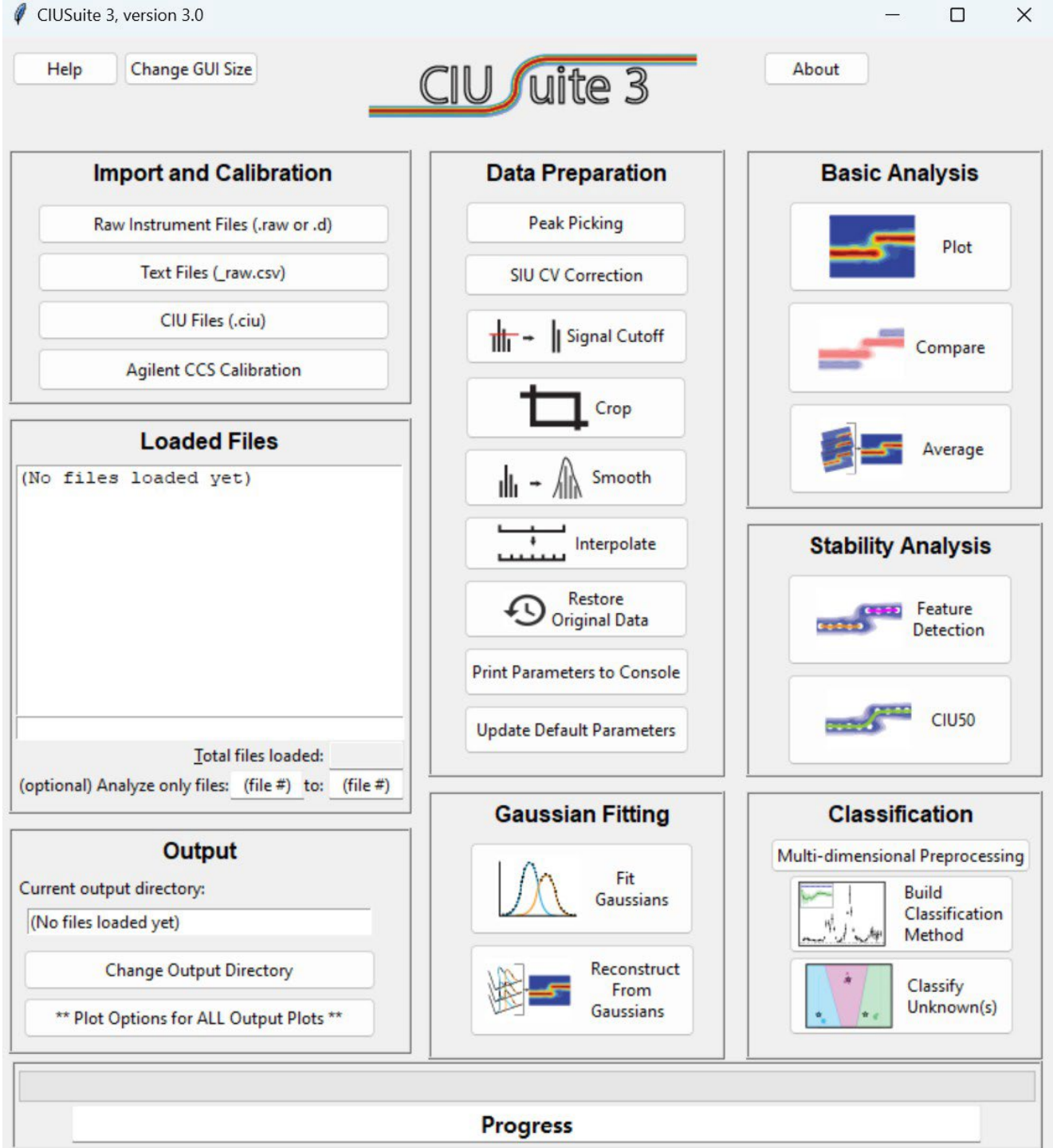
CIUSuite 3 Main GUI

Main GUI is an expansion of CIUSuite 2.
All the features that were originally in CIUSuite 2 remains in the main GUI.
Some of the new features have their own independent buttons:

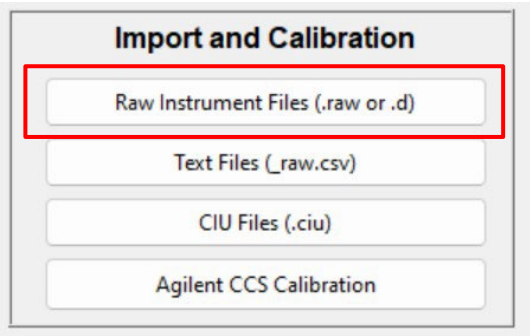
- Peak picking
- DTIM CCS calibration for Agilent
- SIU CV correction
- Signal cutoff
- Preprocessing for multi-dimensional classification

Other new features are built into the pre-existing buttons with a new pop-up window that allows you to choose whether you want to execute those features or not:

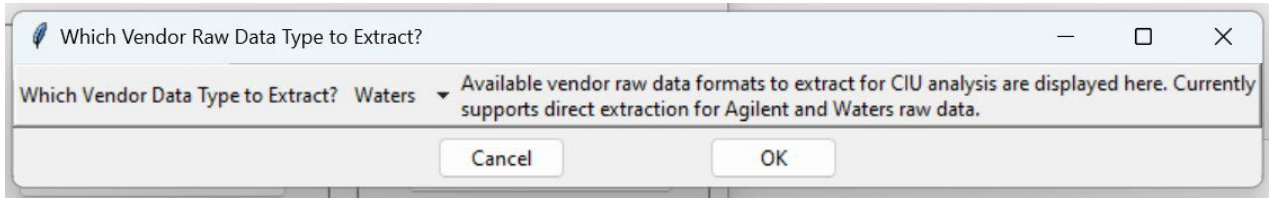
- Feature skipping
- TWIM CCS calibration for Waters

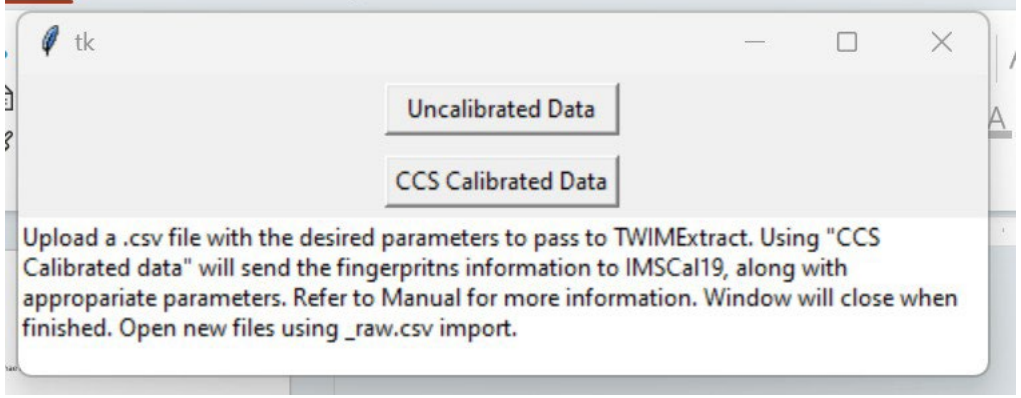


CIU Data Extraction with Raw Instrument Files

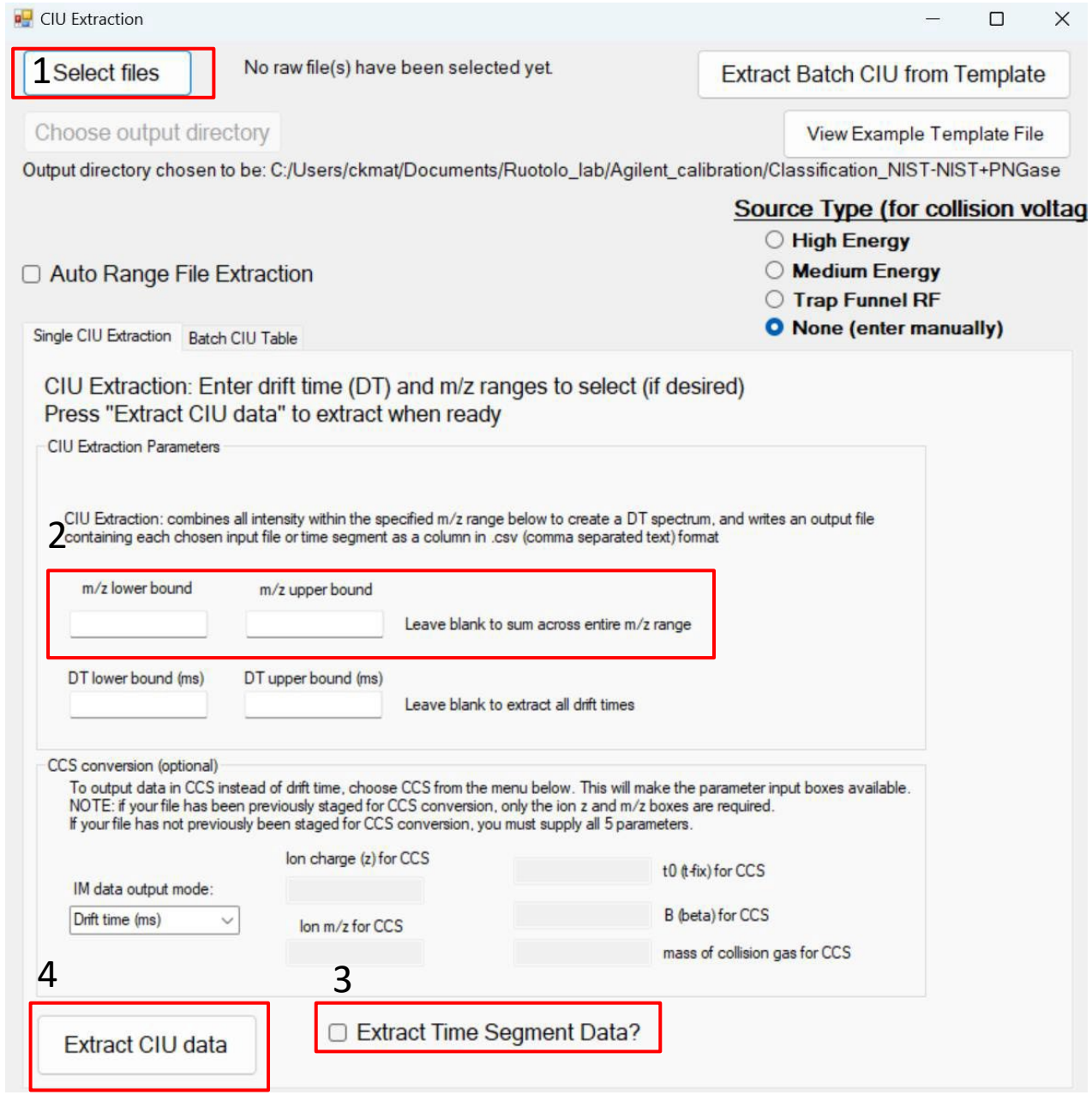


You can extract CIU Data with raw instrument files.
Once you click the red box, it will prompt which instrument vendor the data comes from (see below; Bruker users will need to use separate TIMS extractor)





Waters (top): you will select whether you want to calibrate or not. There will be a separate .csv file where you input the instrument settings which will automatically extract all data and perform calibration using IMSCal19.
Agilent (right): you will need to fill out your m/z bounds for CIU extraction. Check the box time segmented data if that is the case



Importing Raw Data: Waters Raw Files

Parameter .csv File for Waters Extraction

	A	B	C	D	E	F	G	H	I	J	K
1	#Waters .raw File	Range File Name	Range Rile	Range Rile	Range Rile	Range Rile	Range Rile	Range Rile	Molecular	charge	_raw.csv output location
2	*	protein_chargestate	5625	5725	0	100	1	200	147000	26	C:\path_to_where-to-sav
3	C:\path_to_.raw file										
4	C:\path_to_.raw file #2 (if fingerprint was made into several .raw files)										
5	*	ADH_z27	5900	6000	0	100	1	200	147000	27	C:\path_to_where-to-sav
6	C:\path_to_.raw file										
7	C:\path_to_.raw file #2 (if fingerprint was made into several .raw files)										
8											

Column A:

- * - indicates the start of a fingerprint (on this same row put the other parameters in the other columns for this fingerprint)
- path – the paths under the asterisk belong to the same fingerprint

Column B: range file name (a .txt file that will contain the information for extraction to be used by TWIMExtract)
Column C and D: lower m/z and high m/z for extraction
Column E and F: lower retention time and upper retention time for extraction
Column G and H: lower drift bin and upper drift bin for extraction
Column H: molecular weight of the ion of interest (rounding is fine)
Column J: charge state of the ion of interest
Column K: path to save uncalibrated versions (.txt_raw.csv)

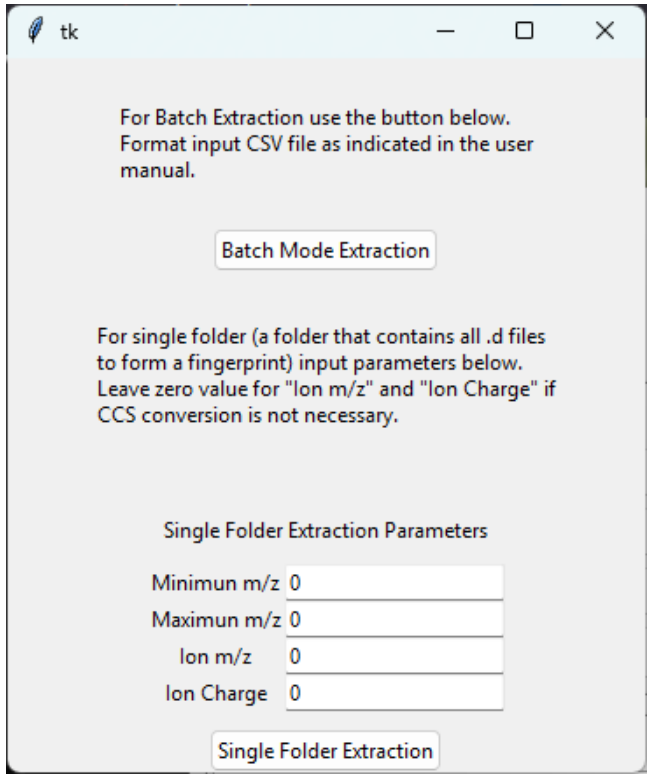
Importing Raw Data: Waters Raw Files CCS Cal

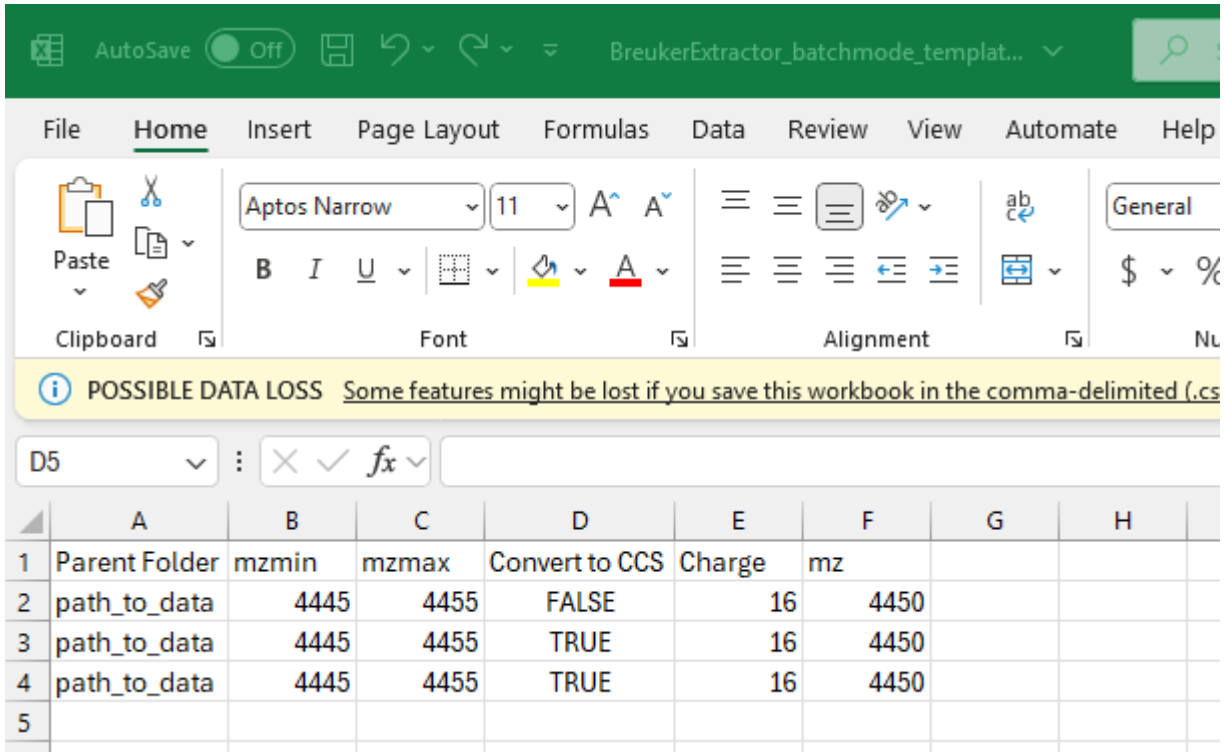
Parameter .csv File for Waters Extraction (L – M columns are REQUIRED for CCS CALIBRATION)

	L	M	N	O	P	Q	R
1	calibration .dat file location (path)	output location for calibrated file	TWIM cell length (m)	wave velocity (m/s)	wave height (V)	drift region pressure (mbar)	temperature (K)
2	C:\path_to_where is \IMSCal19\input\adh_ref_WV300-WH30.dat	C:\path_to_where_to_put_calibratedfiles	0.245	300	30	4.2	298
3							
4							

Column L: Where is the .dat file with all the calibrants information for CCS calibration
Column M: where to save CCS calibrated files (ccs_raw.csv)
Column N – R: CCS calibration parameters

CIU Data Extraction with Raw Instrument Files (Breuker)





	A	B	C	D	E	F	G	H
1	Parent Folder	mzmin	mzmax	Convert to CCS	Charge	mz		
2	path_to_data	4445	4455	FALSE	16	4450		
3	path_to_data	4445	4455	TRUE	16	4450		
4	path_to_data	4445	4455	TRUE	16	4450		
5								

Nov 2024 Update:
Once Breuker is selected as the vendor, the above GUI will appear.
Follow the prompts.
m/z values do not need decimals.

The above is the format of the CSV file use for Breuker Extraction (batch mode). The template file can also be found in the CIUSuite3 installation folder.

Importing Raw Data: Text format (_raw.csv)

- CIUSuite 3 imports data from '_raw.csv' format
  - Comma-separated values file (text)
  - First row is activation energy values (green)
  - First column is ion mobility values (blue)
  - Intensity data fills the matrix (white). Each column represents the summed arrival time distribution at a given activation energy.
- Activation and IM values can be anything (as long as they are numbers)
  - E.g. activation axis can be collision voltage, laser power, solution temperature, etc.

Activation Energy Axis (e.g. 'collision voltage (V)')

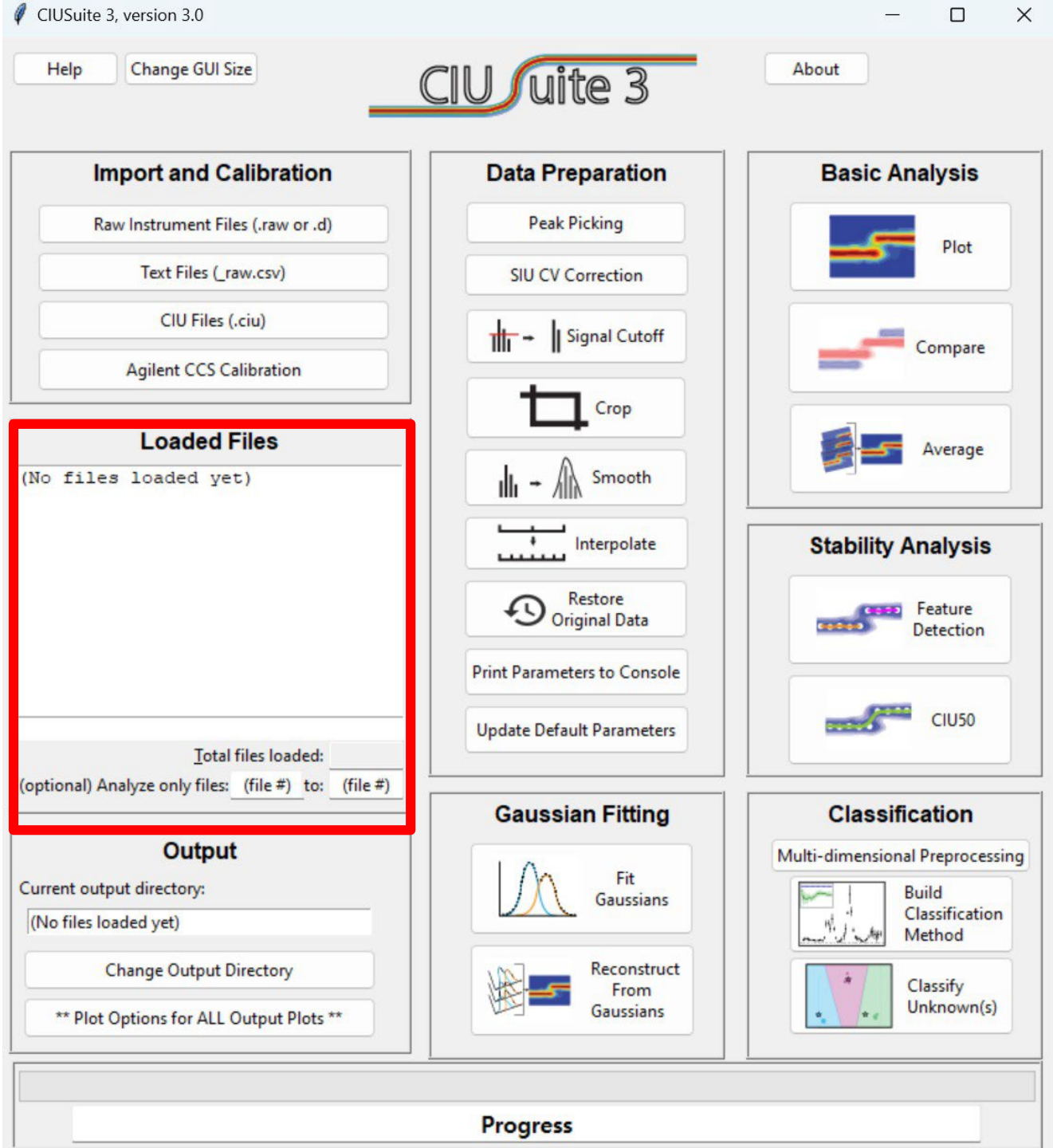
Ion Mobility Axis (e.g. 'drift time (ms)')	**10**	**15**	**20**	**…**
0	0	2	0	…
0.182	5	7	2	…
0.364	0	0	1	…
…	…	…	…	…

Loading Data: the .ciu file

- After raw data has been processed with CIUSuite 3, a “.ciu” file is created to save the data and any processing results for future reference.
  - A .ciu file contains:
    - Raw data
    - Processed data (after normalization and any smoothing, cropping, interpolation, etc.)
    - Analysis results (e.g. detected features, fitted Gaussians, CIU50 values, etc.)
    - Parameters used to generate the processed data and analysis results
  - .ciu files are saved automatically after all processing steps and can be loaded using the load ‘.ciu’ button in CIUSuite 3. This enables viewing of any previously generated results/analyses (so long as they are not overwritten by running an analysis again on the same data)

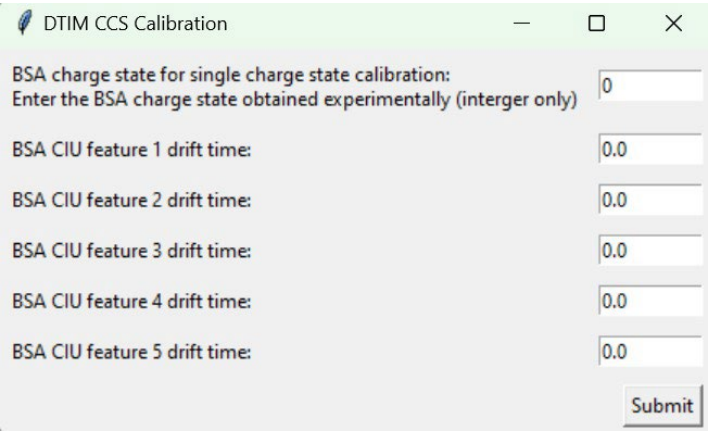
Loading Data Window

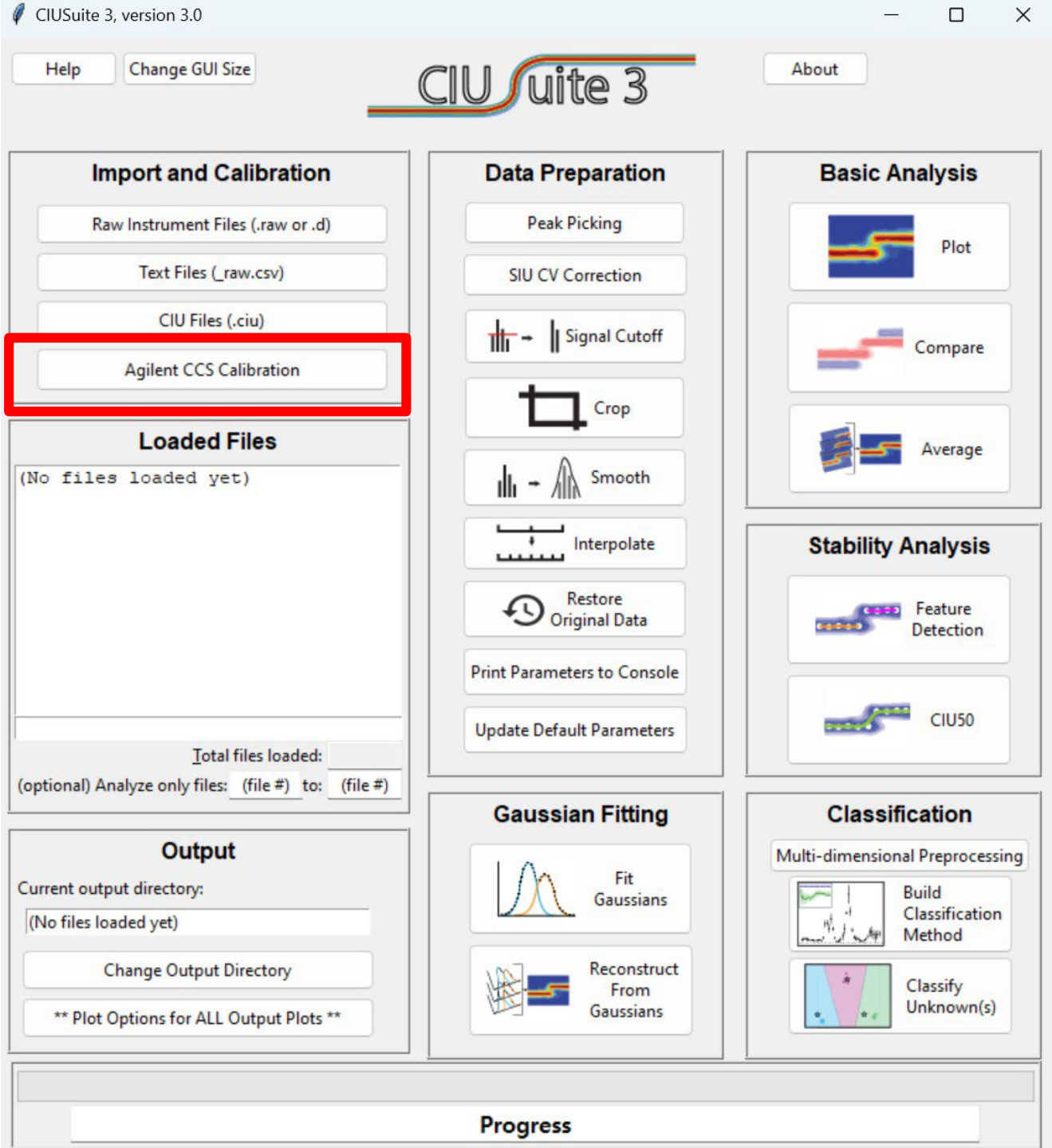
- After data is loaded (from any source), it will be displayed in the table of “Loaded Files”
  - Any manipulations/analysis performed will be done on ALL files loaded in the table
  - The table will only display ~15 files. Many more files can be processed at once, they will just extend beyond the table and will not be visible.
  - To analyze only a subset of the loaded files, use the “Analyze only files ___to __” boxes below the table to specify the files (by number) to process.



Agilent CCS Calibration

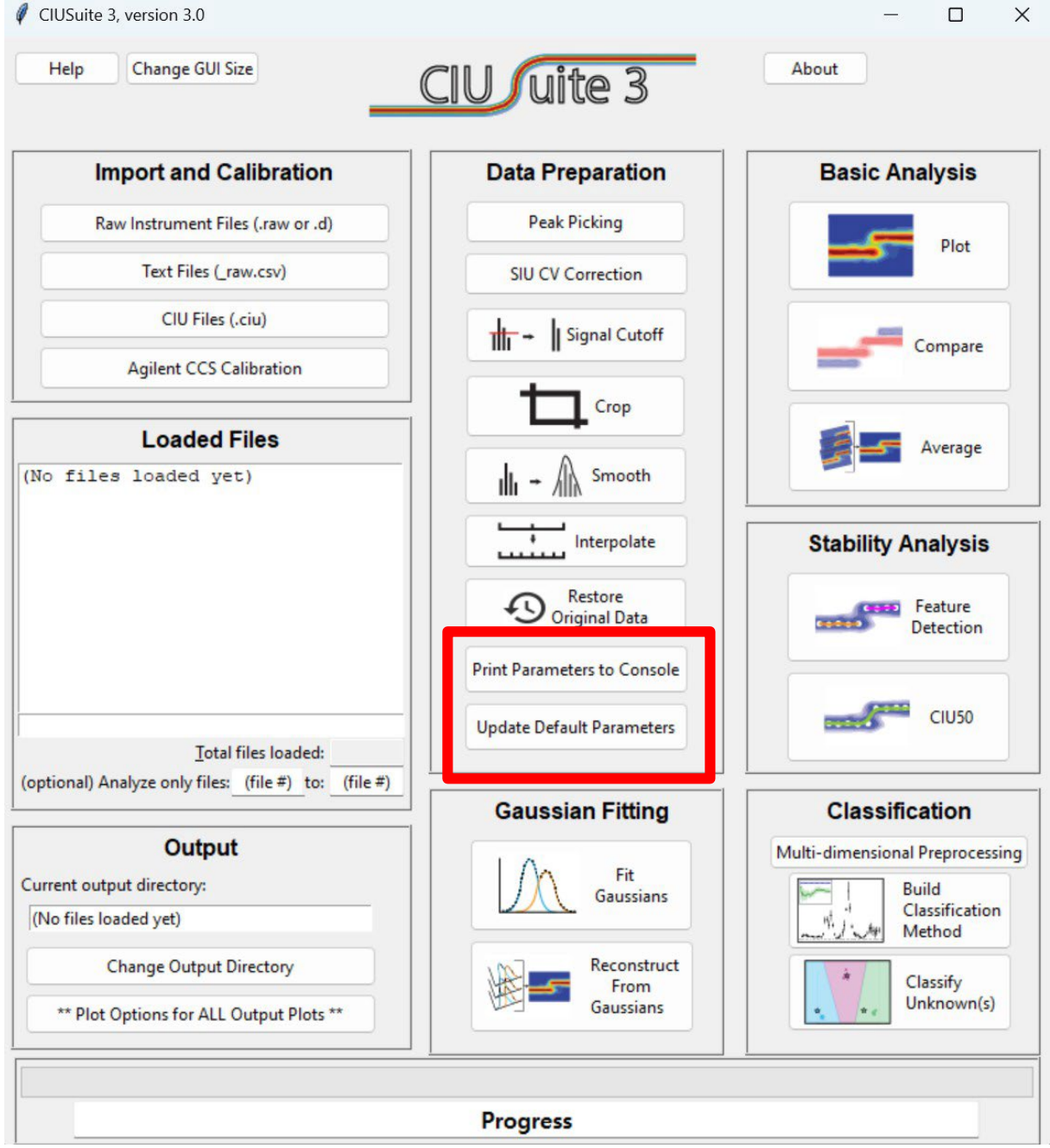
- To calibrate drift tube IM data from Agilent, you will need beta and t_{fix} values.
- Agilent CCS Calibration uses BSA CIU data where drift time of each feature will be used to determine beta and t_{fix}.
  - A new window will appear (bottom left). Enter the charge state of BSA CIU data, and the drift times of features detected. Click “Submit”.
  - A new window will print beta and t_{fix} values. These can be used to convert dt into CCS on Agilent IM-MS browser.





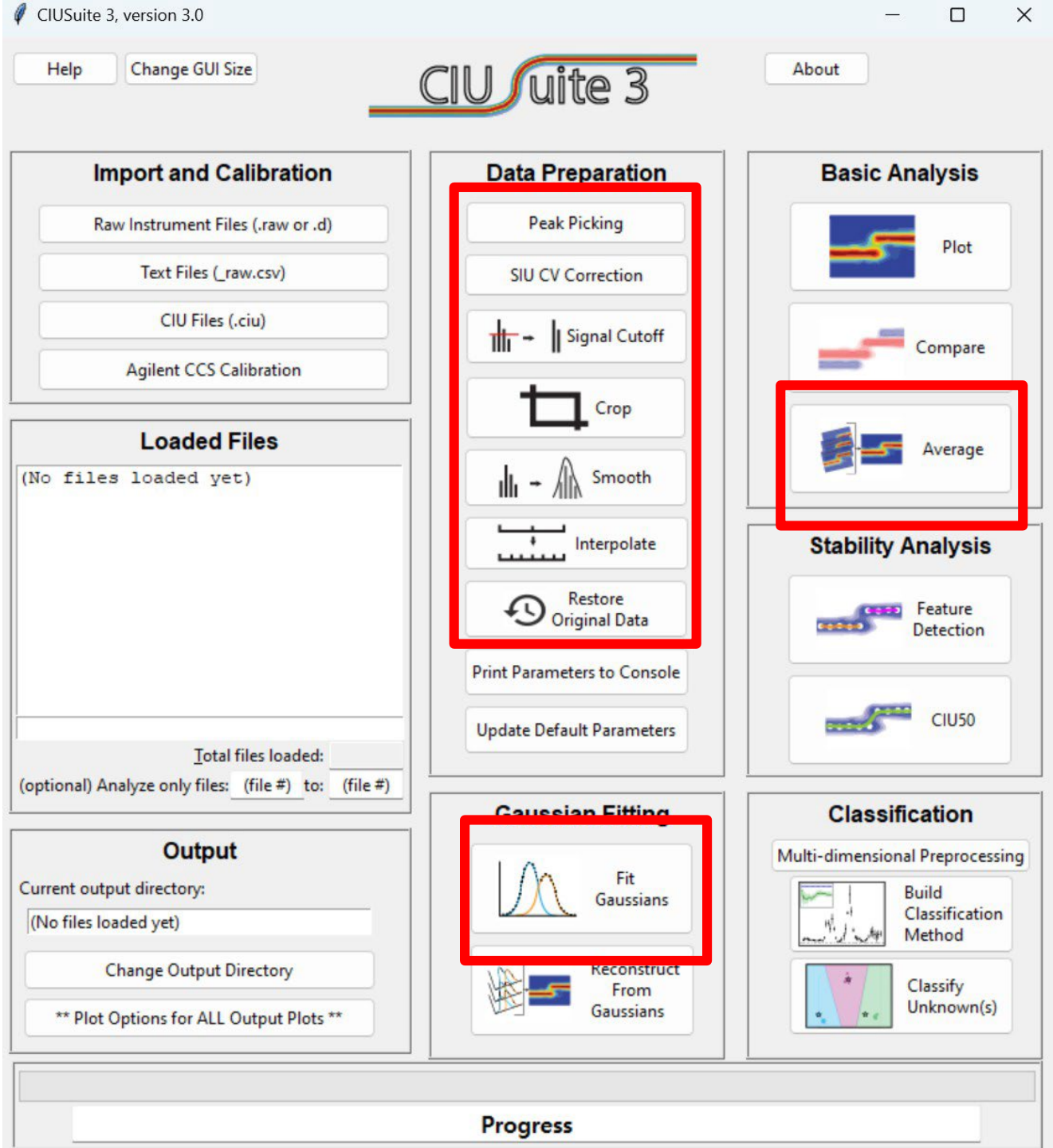
Parameters

- The parameters used in analysis are saved into the .ciu file – so to see how any particular data was analyzed, use the 'print parameters from file' button.
  - NOTE: these are updated every time the .ciu file is saved (which is after most analyses), so it is possible in some cases to overwrite them after the fact
- If you want to save parameters for future analyses, use the 'update default parameters' button, which will save all current parameter values to be the defaults.



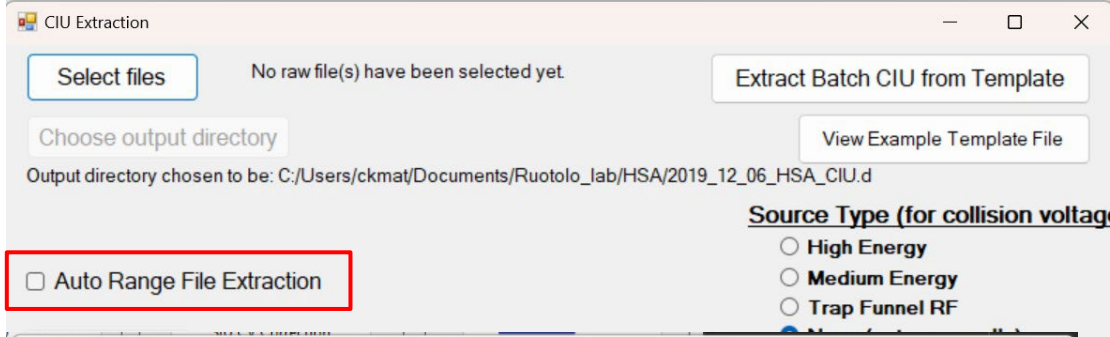
Data Processing

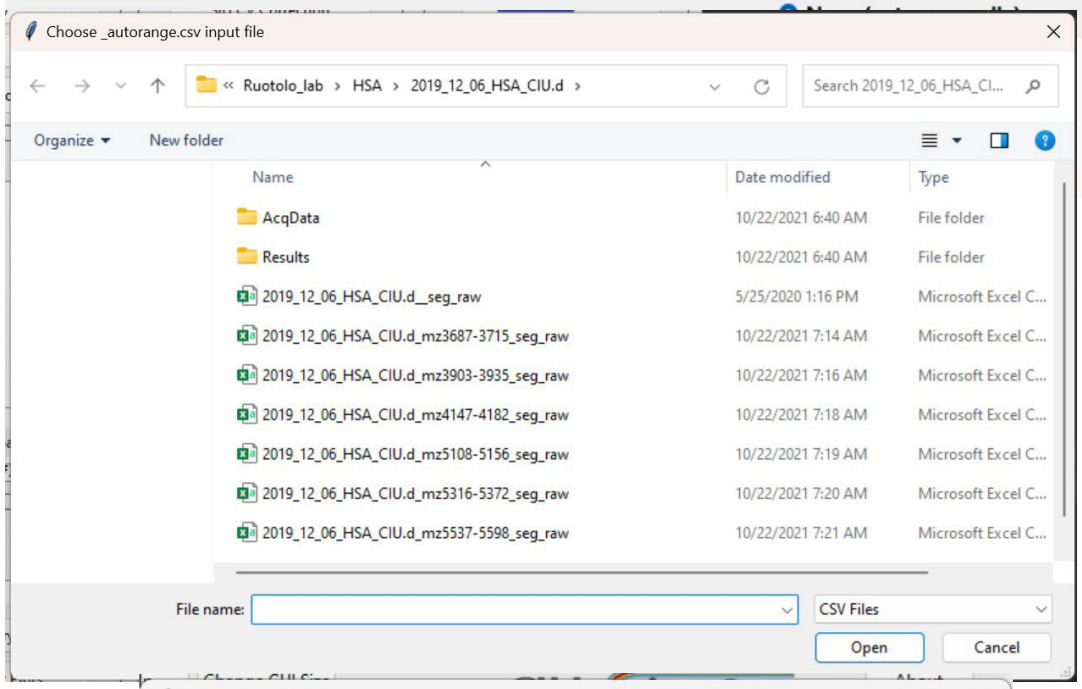
- CIU data can be prepared for analysis in a number of ways. See the following pages for details on the methods highlighted at right

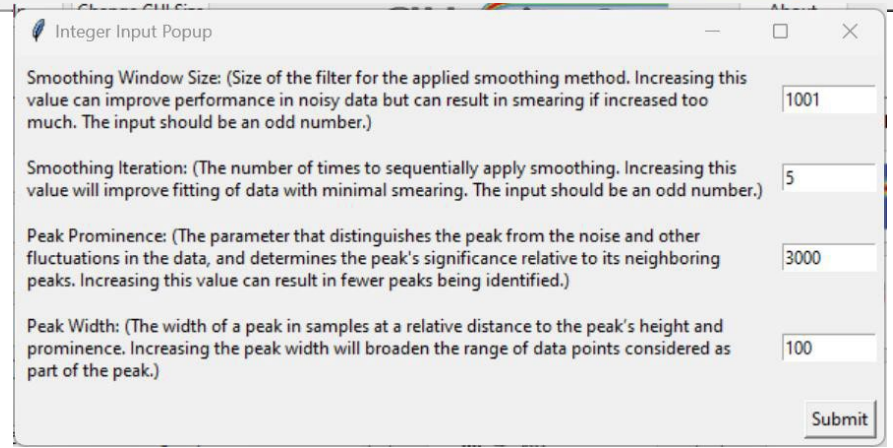


Data Processing: Peak Picking

- Peak picking automatically detects peaks of various charge states for m/z charges to extract for CIU data extraction and analysis.
- When you extract your data from raw data file,
  - Waters: set your m/z ranges from min and max m/z collection range to include all peaks in your _raw.csv
  - Agilent: check “Auto Range File Extraction” for extraction. This should create a new “_autorange.csv” file
- Click “Peak Picking” ➔ load _autorange.csv or _raw.csv
  - This will prompt a peak picking parameters. The values are default values that has worked successfully with BSA CIU data. You can modify these values to optimize your peak picking
  - You will get an output of spectrum with peaks annotated along with .csv summary of detected peaks’ centroids and m/z min and max range for each peak.





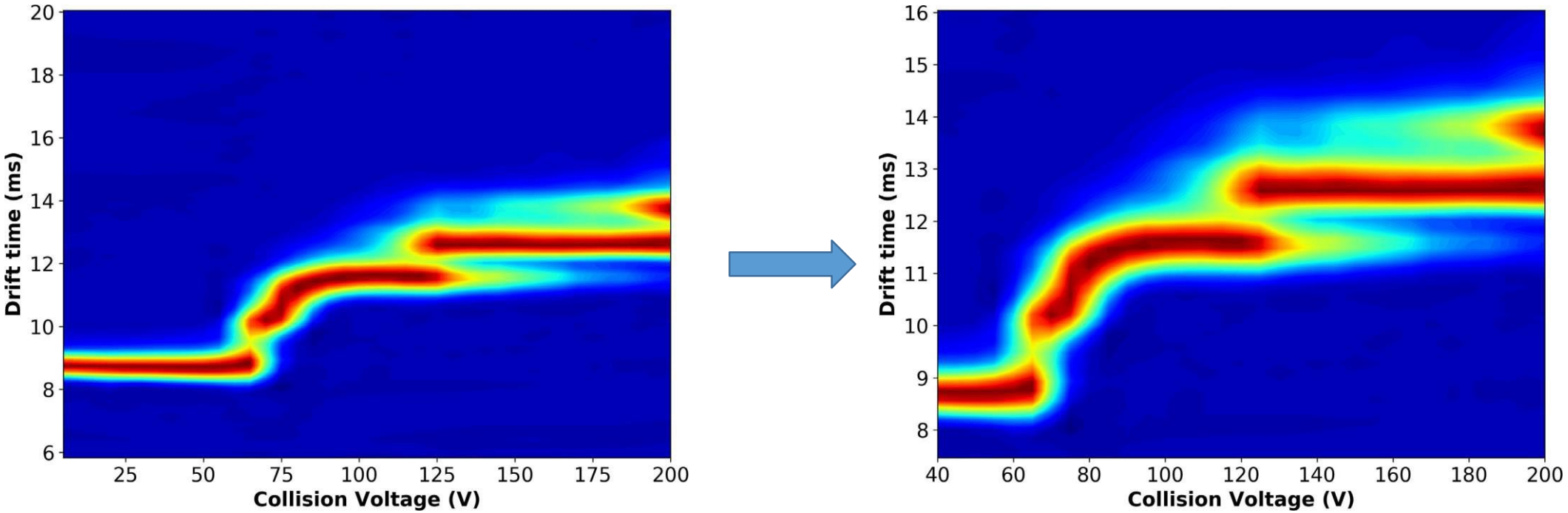


Data Processing: SIU CV Correction & Signal Cutoff

- SIU CV correction feature is to correct the CV offset that occurs during SIU correction.
  - For SIU performed on Waters cyclic IM-MS, hardware records the trap CV at 10V higher than the actual voltage.
  - To account for this offset, SIU CV Correction feature will ask to select a _raw.csv file you would like to correct. Once the file is selected, the software will subtract 10V from all trap CV in the csv file.
- Signal Cutoff remove noises below certain threshold intensity value.
  - Input a value (between 0 and 1) that you want as a threshold. Any signals that are below this threshold value will be removed.
    - i.e. if 0.2 is entered, any signal below normalized intensity of 0.2 will be removed.

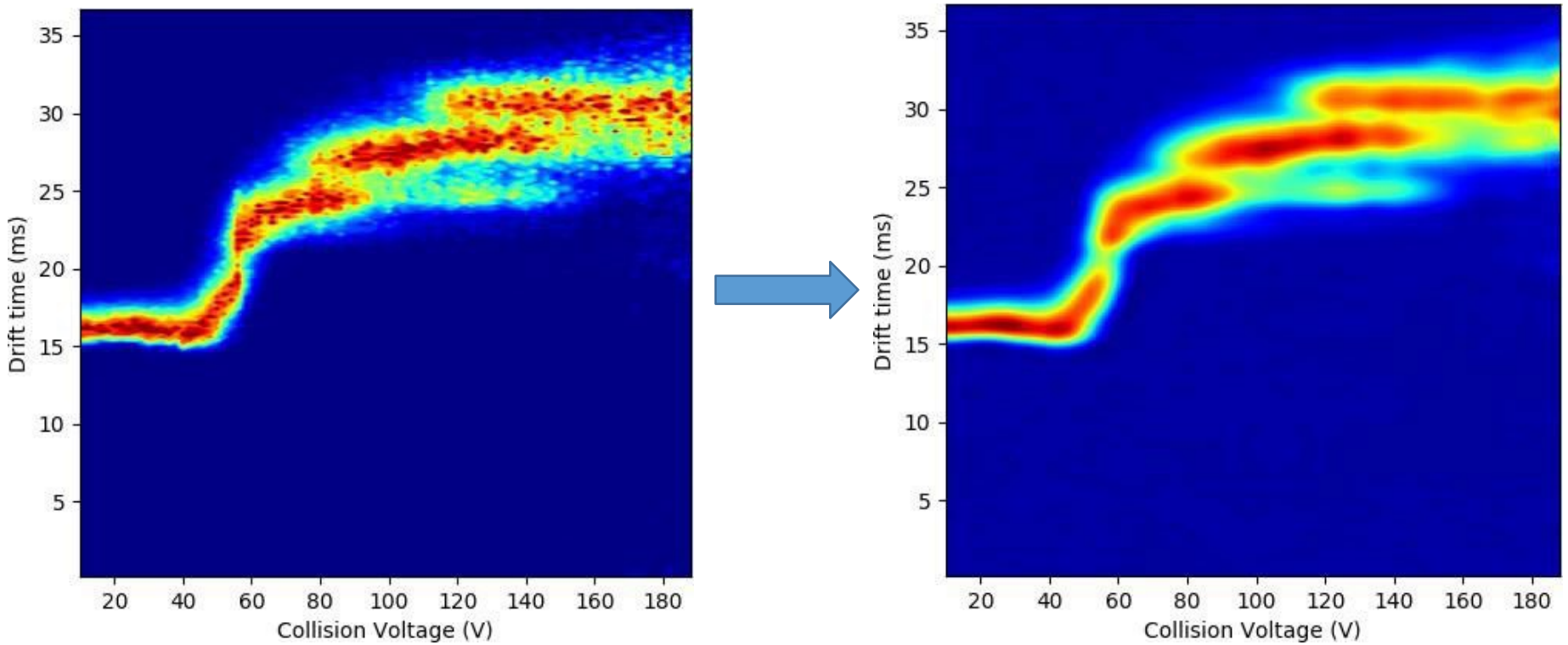
Data Processing: Cropping

- Cropping cuts the fingerprint down to the specified dimensions in one or both axes.
  - This can be useful to focus on particular areas of interest, or simply to generate higher quality visuals.
  - The cropping menu will automatically detect the starting boundaries of the first fingerprint loaded in the processing table
- NOTE: cropping will REMOVE any feature detection/Gaussian fitting/etc. from a .ciu file because the boundaries used for those analyses are no longer valid



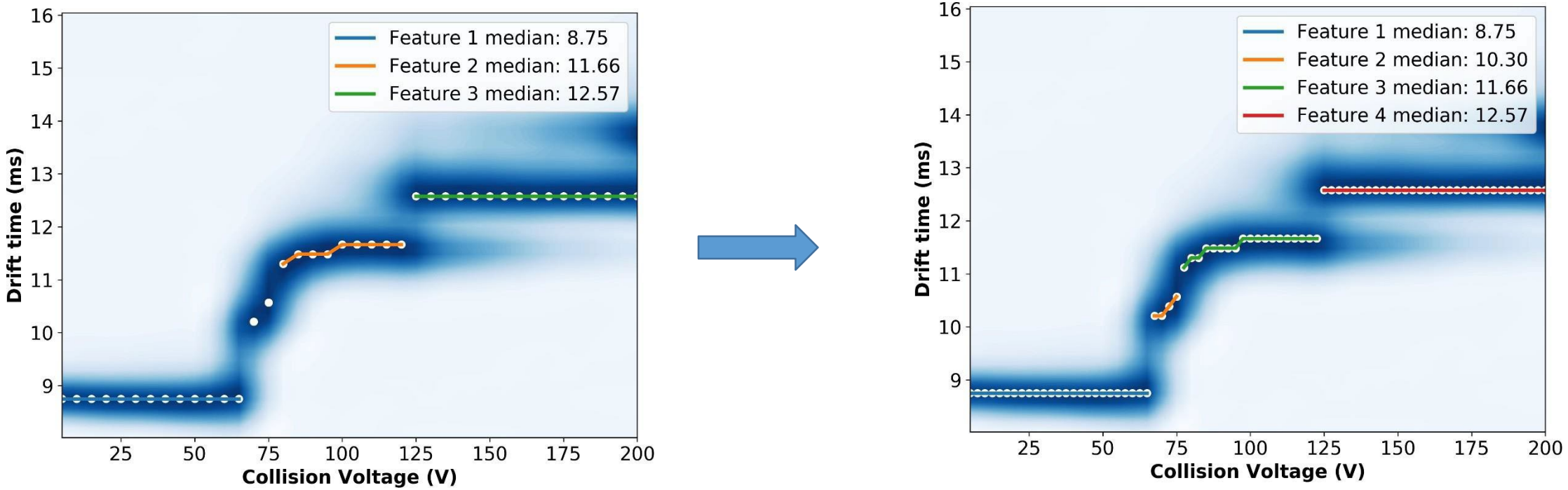
Data Processing: Smoothing

- A 2D Savitzky-Golay filter is used to smooth by default. 1D S-G filter (in drift time only) or no smoothing can be used instead if desired
- Window Size: The size (in steps/bins) of the filter to use (in both dimensions) for smoothing. Larger windows will include more data in the smooth, increasing smoothing but potentially blurring features
- Iterations: The number of times to smooth in succession



Data Processing: Interpolation

- Interpolation increases the number of bins/steps along an axis (or both axes).
  - Interpolating the activation axis can be very helpful for feature detection (shown below) to find features that are only present for a few steps in the original data
  - Interpolating the drift time/mobility axis can be helpful for Gaussian fitting
- Interpolation is done by a "factor," or fold change in the number of bins. For example, interpolating by a factor of 2 means the number of bins along the axis will double (factor of 4 will quadruple, etc.).



Data Processing: Restore Original Data

- Any smoothing, interpolation, or cropping can be undone by using the ‘restore original data’ button.
  - This loads the original raw data (saved from the original _raw.csv file used to generate a given .ciu file)
  - You will be prompted to do smoothing on the restored data. If you do not want to smooth, select “none” as the smoothing method.
- NOTE: Restoring original data will clear ALL processing and analysis from the file

Data Processing: Averaging

- Multiple replicate datasets can be merged into a single average fingerprint using the average button.
  - Averaging generates a new .ciu file containing the averaged data that can be used for further analysis/comparisons/etc
- NOTE: the averaging button will combine all files loaded in the table by default, so make sure only replicates of the same thing are loaded. Or use the ‘analyze files from/to’ entries to select only the files to be averaged.

Data Processing: Gaussian Fitting

- Gaussian fitting models the arrival time distribution as a sum of Gaussian functions (peaks) corresponding to one or more conformations of the analyte ions
- Fitting can be done in two modes:
  - 1) Protein only mode
    - All peaks are assumed to be from the analyte of interest, and are expected to fall within a relatively narrow distribution of peak widths
  - 2) Denoising mode (signal and noise present)
    - Both analyte and other (noise) components are fit. Noise components are assumed to have larger width than analyte components, and can thus be separated and filtered out.
- Terminology:
  - In many places, the term “protein” is used to refer to the signal/analyte peak and “non-protein” to noise peak(s). This does not mean that fitting only works for proteins – so long as the width of signal and noise peaks are different, this method should be able to differentiate any signal peaks from any noise peaks.

Data Processing: Gaussian Fitting Parameters

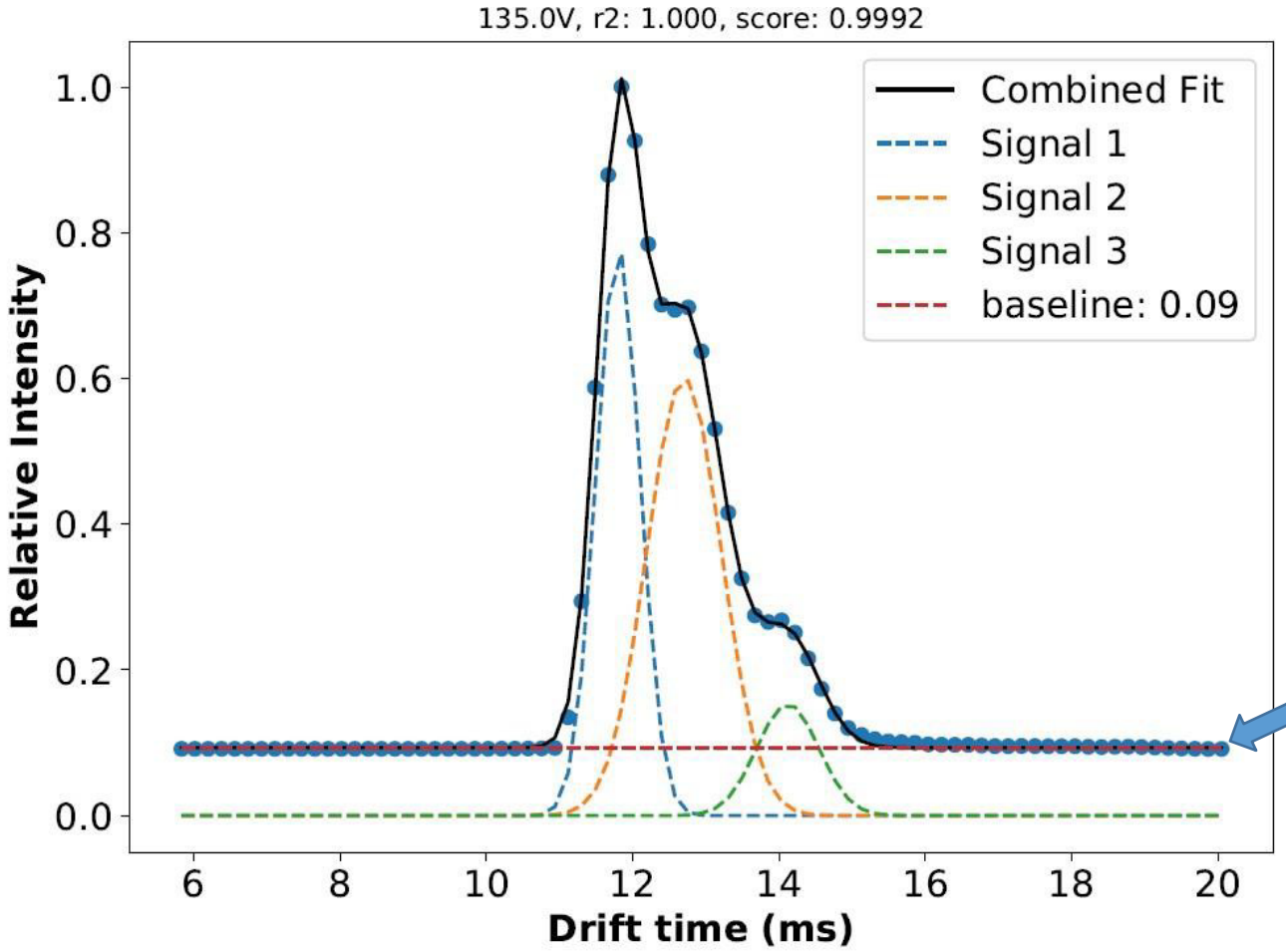
- Protein/Signal Only Mode:
  - True/False – True for protein only, False for protein and noise
- Minimum Peak Amplitude:
  - Minimum normalized (0 to 1) intensity for a peak to be fit
- Save Diagnostics:
  - If True, generates plots of all fits considered during the optimization, not just the highest scoring ones chosen. Useful for finding appropriate fitting parameters.
- Number of Cores for Multithreading:
  - Uses Python's multiprocessing module to increase the speed of calculations
  - NOTE: this can cause problems on some systems because of how Python interacts with the operating system. If you experience problems, set this value to 1, which will perform serial processing instead and not use multiprocessing at all.

Data Processing: Gaussian Fitting Parameters

- Max Protein Components:
  - The maximum number of protein (signal) peaks to consider fitting at a single collision voltage. Higher numbers will generally improve results, but can increase processing time significantly.
- Expected Protein Peak Width:
  - ****This value is very important****
  - This is the expected width (full width at half max, FWHM, in the units of your ion mobility axis) of the protein peak(s). It depends on instrument resolution and the protein/analyte in question, which is why it needs to be entered by the user.
  - If you aren't sure what value to use, try fitting with the defaults and see what the resulting peak widths are; use that as a guide for refining the input width.
- Protein Peak Width Tolerance:
  - Protein peak widths are constrained to be within (Expected width +/- Tolerance), meaning that any peaks with widths outside this range will NOT be considered by the fitting algorithm.
  - Same units as expected peak width
- Peak Overlap Mode: (see manuscript SI for details)
  - "strict" penalizes peaks that overlap > 50% and is good for preventing overlapped fits
  - "relaxed" is the recommended setting – it penalizes overlaps of > ~85% but not so harshly as to prevent overlapping peaks from being considered when comparing to other solutions.
  - "none" means no peak overlap penalties will be applied. Use this if you think you have closely-overlapped peaks, but be aware that overfitting is very possible.

Data Processing: Gaussian Fitting Parameters

- Baseline mode: If True, includes a “baseline” function in the fit that provides a flat, vertical offset for all Gaussian functions. Useful in cases where data has a significant non-zero baseline (e.g. below)

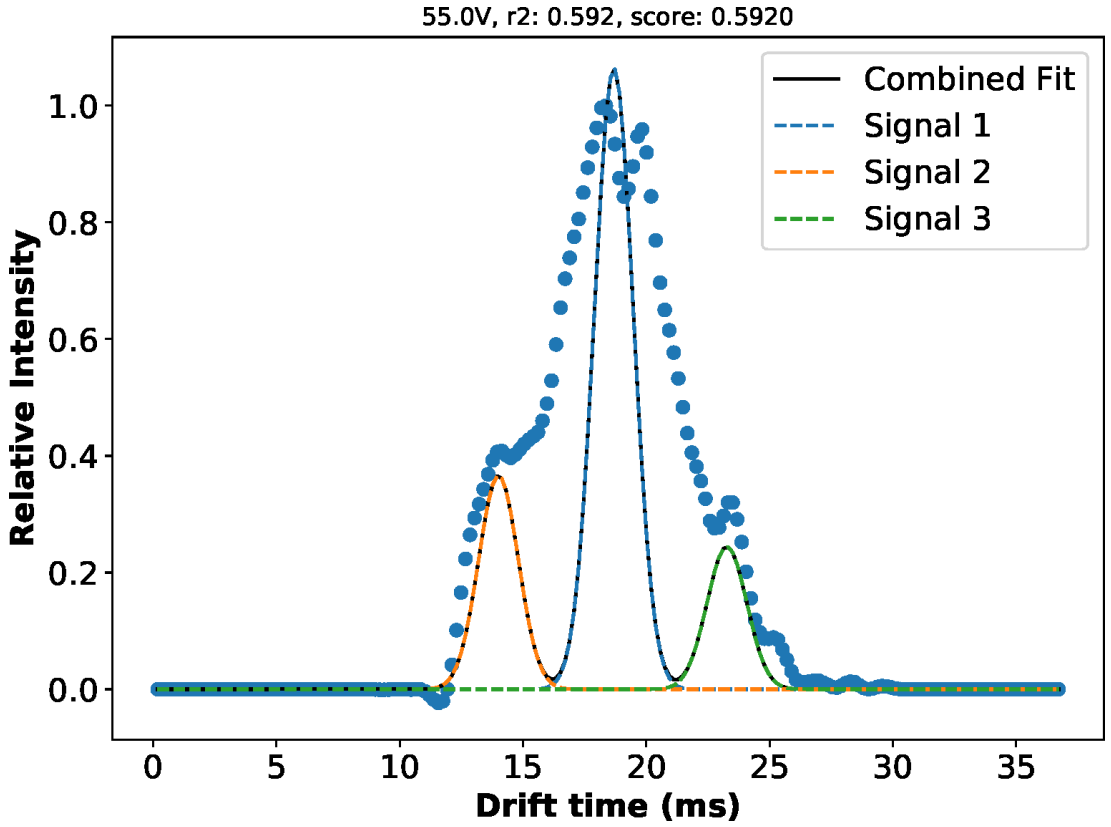


Input data does not drop to 0 away from peaks – it has a baseline. In this case, the baseline is about 9% relative intensity

Gaussian Fitting - Troubleshooting

It can be challenging to get accurate and consistent Gaussian fit results because the parameters used are critical and can vary significantly for different proteins and/or instruments. Some guidelines for improving fit quality are on the following slides

- **The more components allowed, the better the fits will be**
  - Allowing more components gives the fitting algorithm more chances to get the ‘right’ answer with different initial conditions. Overfitting will be filtered out by the scoring system (adjust between ‘strict’ and ‘relaxed’ as needed if overfitting is a problem), so the only downside to allowing extra components is that the fit will take longer.

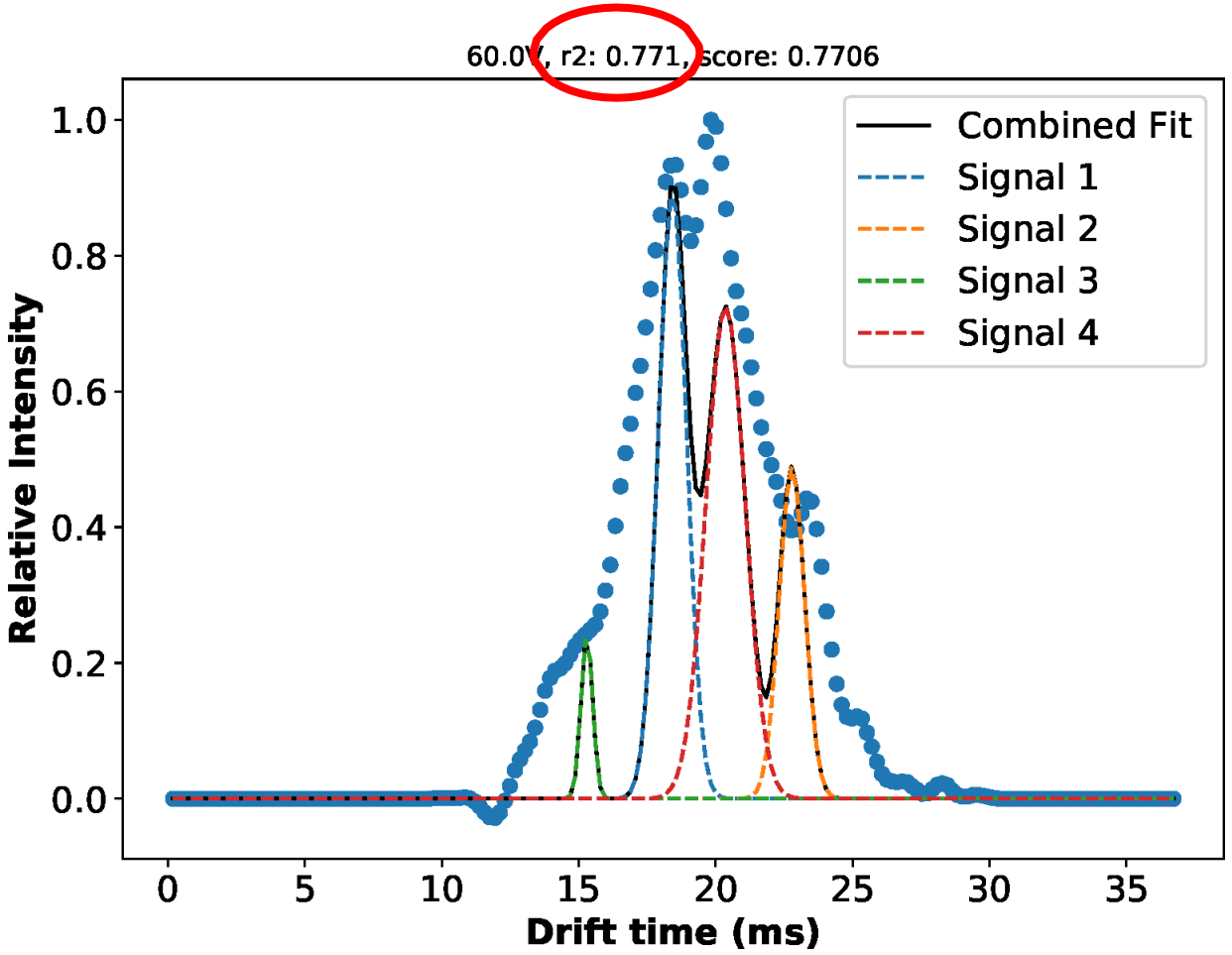


Too few components are allowed in this example – there are at least 4 peaks present, but only 3 are fit due to limitations on number of components. Try allowing more components than needed to get a better chance of good results (e.g. for this data, 6-8 components would be good)

Gaussian Fitting – Troubleshooting (ctd)

- **If your combined fits aren't matching the data (r^2 < 0.98 or so), the parameters need to be adjusted**
  - The peak widths and number of components are constrained by the parameters set by the user, and it is entirely possible that the parameters provided do not allow a good fit.
    - E.g. if the actual width of the peaks is not in the range of allowed widths, the correct fits will never be found because they are disallowed by the parameters used!
  - A good strategy when you don't know what your width values are is to allow a wide range of widths (by setting the 'peak width tolerance' to be a large value), letting the fitting algorithm find good widths, and then using those output values as a guide in future rounds of fitting. The widths can be found in the output _gaussians.csv file.

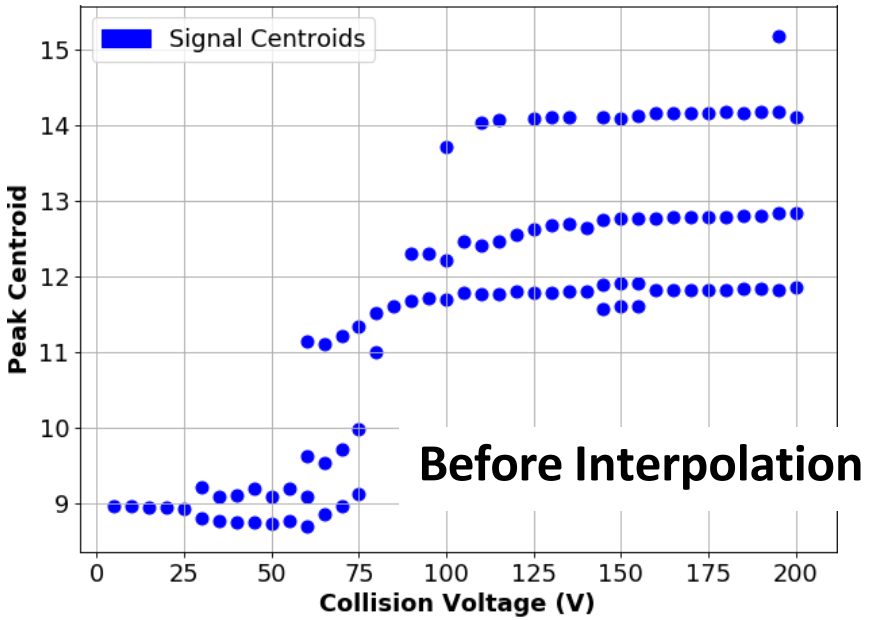
Low r^2 – poor overall fit

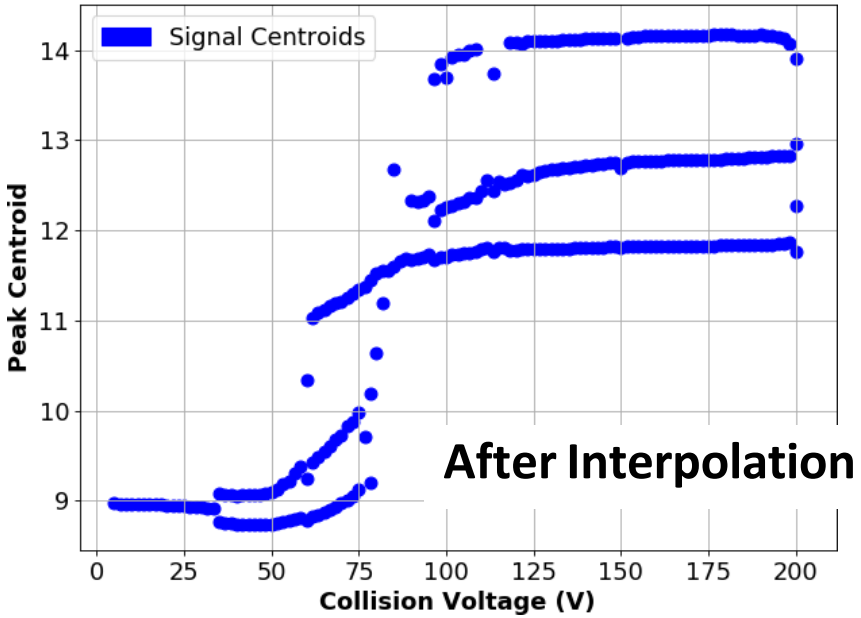


The peaks in this example are constrained to be too narrow – they don't match the actual width of the observed peaks, resulting in a poor fit. (There are also too few components allowed)

Gaussian Fitting – Troubleshooting (ctd)

- **If adjusting width/number of components isn't helping, try interpolating the data**
  - Interpolating across the drift time axis helps Gaussian fitting by increasing the number of points being fit for a given peak
  - Interpolating across the collision voltage axis is also helpful, because it gives the fitting more opportunities to find the right answer
    - For example, if the fitting is 90% accurate, with 20 collision voltages, 2 would be incorrect. With 60 interpolated voltages, 6 would incorrect, but because they are generally not next to each other, they are easier to ignore in feature detection/etc.





Interpolating across the drift time axis improved the quality of individual fits (slightly)

Interpolating across the collision voltage axis made it easier to distinguish good fits from bad (and improved feature detection accuracy)

In this example, changing peak overlap method to 'strict' could also be helpful to reduce the congestion in the first transition region

Noise Removal Mode: Gaussian Fitting

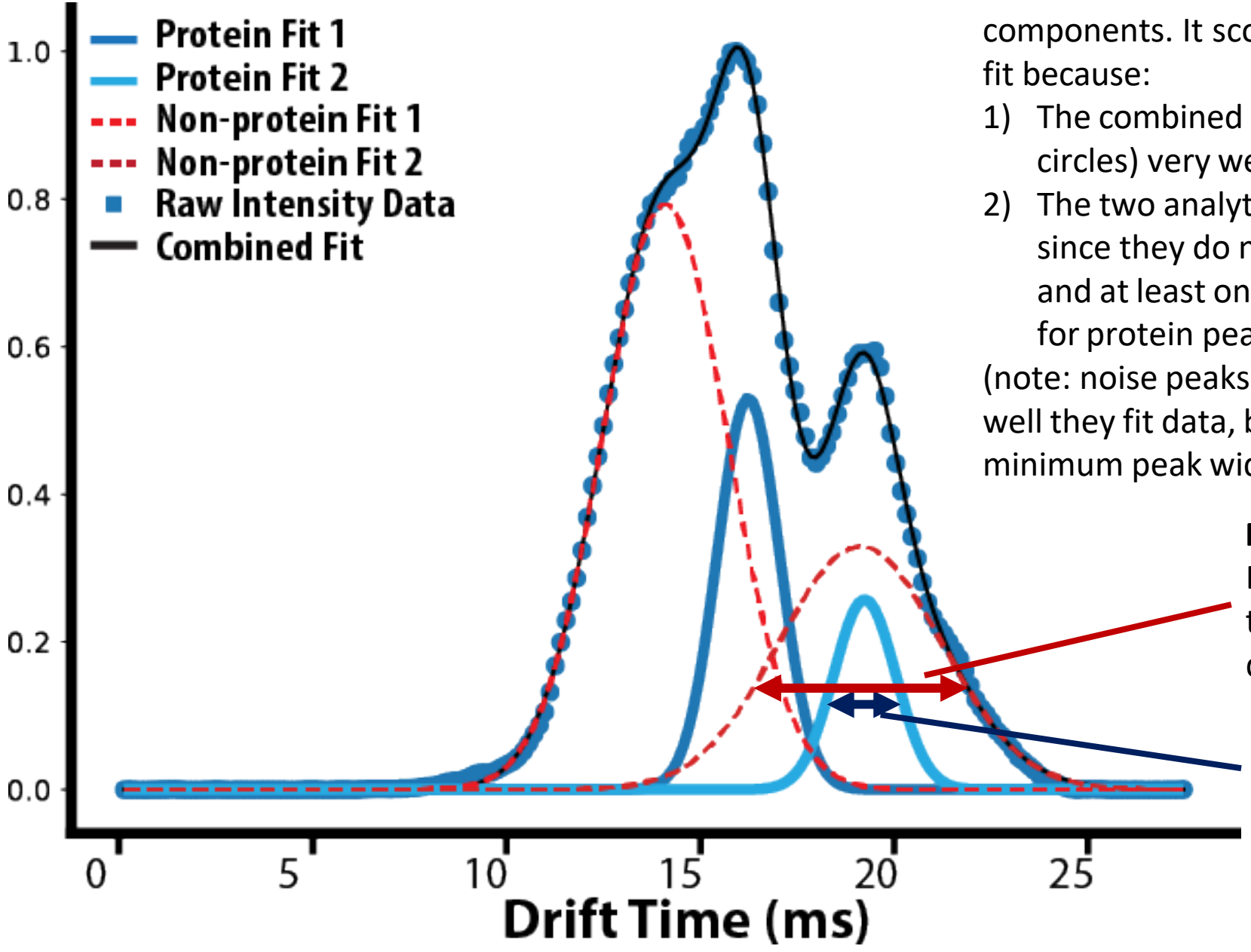
****THESE PARAMETERS ARE FOR NON-PROTEIN MODE ONLY – THEY ARE NOT USED IN PROTEIN ONLY MODE****

- Min/Max NonProtein Components:
  - The minimum and maximum number of non-protein peaks to consider in fits. Increasing the Max number generally improves fit quality but will significantly increase fitting time. The Min number is useful if you know your data always has a certain number of nonprotein peaks, but generally shouldn't be adjusted.
- Minimum Width NonProtein:
  - ****This value is very important****
  - The minimum width for non-protein components. This **must** be greater than the maximum protein width (which is equal to [Expected protein width + protein width tolerance]) for accurate results, and generally should be at least (1-2 * tolerance) greater than max protein width for best results.
  - Because the algorithm uses only peak width to differentiate protein from non-protein components, the more different their widths are, the more accurate the results will be.
- Minimum amplitude for highest protein peak:
  - If a fit has no protein peaks above this amplitude, it is heavily penalized. This prevents fitting only non-protein components, and should be adjusted to reflect the relative intensity of protein/noise in your data.

Data Processing: Gaussian Noise Removal

- Gaussian fitting in Non-Protein mode generates a list of Protein and Non-Protein peaks. The non-protein peaks can be removed to allow for cleanup of noisy data.
- There are two ways to do this:
  - 1) Run Gaussian fitting, then do feature detection in Gaussian mode. The feature detection will only consider protein peaks, effectively removing the noise from Gaussian feature data but leaving the original raw CIU data intact.
    - Gaussian feature data is used for classification, so this is all that is necessary to run classification in Gaussian mode.
    - NOTE: CIU50 relies on the original raw data to some extent, even in Gaussian mode, so this will NOT completely remove the effects of noise on CIU50 analyses. For that, mode #2 is recommended.
  - 2) After feature detection, use the “reconstruct from Gaussians” button to generate a new .ciu file using the data from only the fitted protein Gaussians.
    - It’s recommended to smooth this data afterwards, as amplitude values will vary after noise is removed
    - This reconstructed data can then undergo feature detection/CIU50 analysis in standard mode because the noise has been removed already
    - NOTE: if all the noise isn’t removed, Gaussian fitting can be performed again on the reconstructed data, followed by a second reconstruction (and this process can be repeated as many times as necessary)

Data Processing: Gaussian Noise Removal Example



This fit contains 2 analyte and 2 noise peaks, for a total of 4 components. It scored highly and was chosen as the correct fit because:

1) The combined fit (black) matches the observed data (blue circles) very well ($r^2 > 0.99$)
2) The two analyte components do not incur any penalties since they do not overlap significantly with each other and at least one of them is above the minimum amplitude for protein peaks (both of them are in this case).

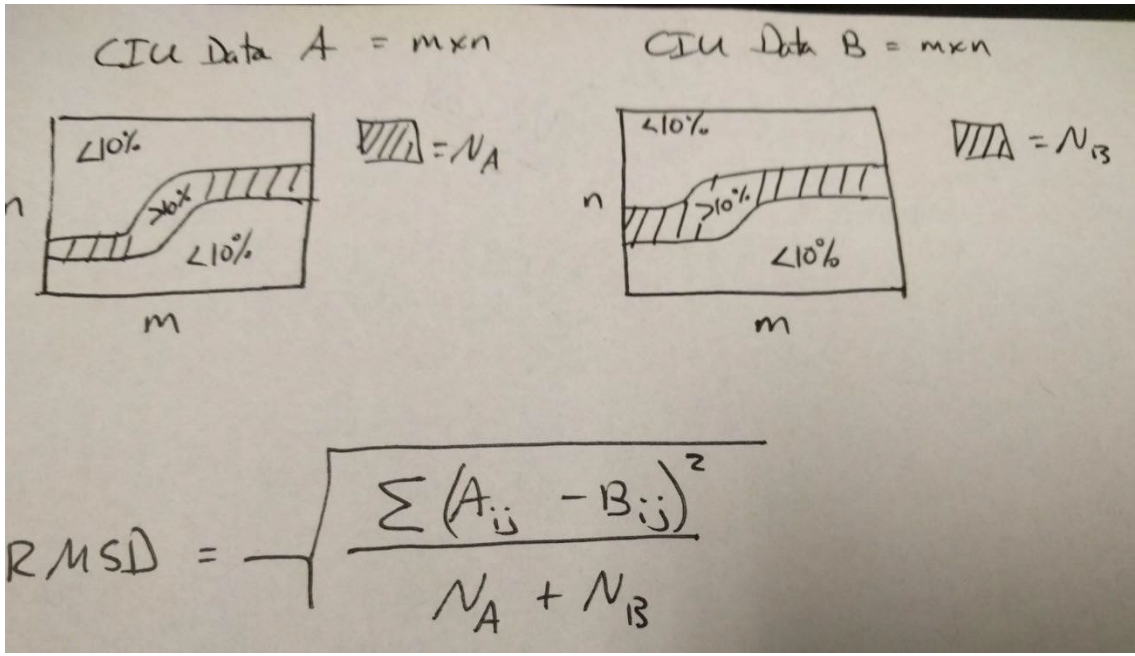
(note: noise peaks are not scored by anything other than how well they fit data, but are constrained to be above the minimum peak width specified).

Noise ("non-protein") peak width (FWHM). Because the noise peaks are much wider than the protein peaks, they can be distinguished by the fitting algorithm.

Analyte ("protein") peak width (FWHM).

Data Analysis: RMSD Comparison

- CIUSuite 3 includes the RMSD (root-mean-square deviation) analysis found in the original CIUSuite software.
  - Details on RMSD analysis can be found in the original CIUSuite manuscript (DOI: 10.1021/acs.analchem.5b03292)
  - **NOTE: RMSD values computed in CIUSuite 3 will differ from those calculated in the original CIUSuite**. (see below for details)
  - Because RMSD is a relative comparison, it does not matter which computation is used so long as RMSD values calculated with CIUSuite 3 are not compared to those calculated with the original CIUSuite or vice versa.



- Original CIUSuite RMSD Calculation:
  - Divided the sum of root-mean-square (RMS) differences by the number of points above the intensity cutoff (10%) in File A PLUS the number of points above the cutoff in File B. (pictured above). This results in some points being counted twice, resulting in reduced RMSD values.
- CIUSuite 3 RMSD Calculation:
  - Divides the sum of RMS differences by the number of points above the intensity cutoff in the difference matrix (N_{dif}) formed by subtracted File B from File A. Formula is as pictured, but with N_{dif} instead of $N_A + N_B$ in the denominator.

Data Analysis: RMSD Comparison

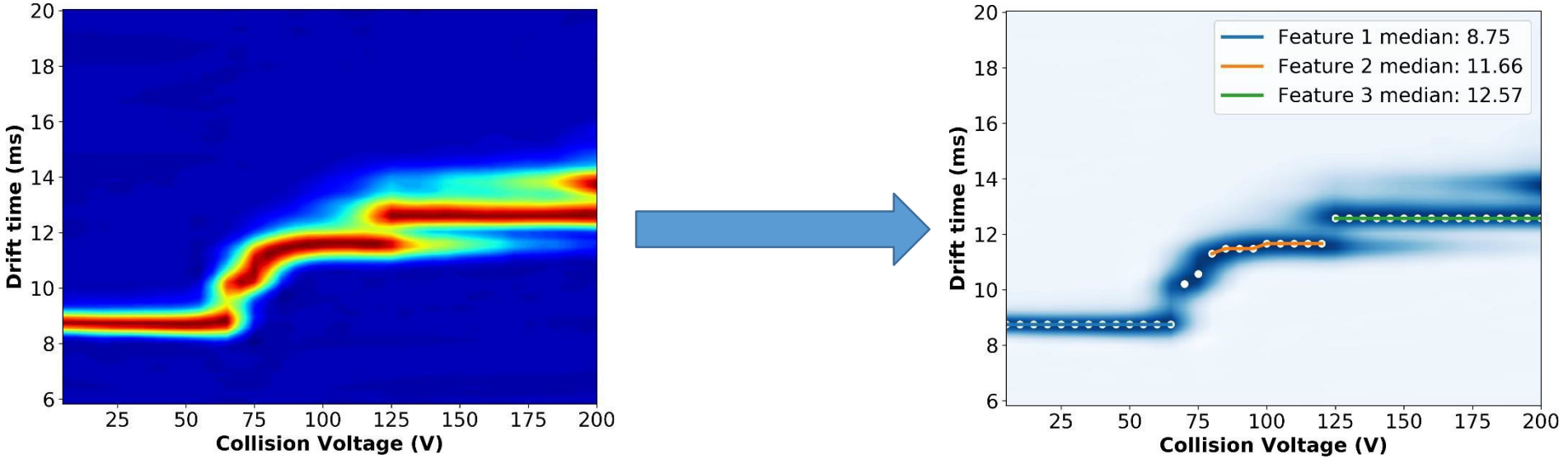
- CIUSuite 3 offers two ways to generate comparisons:
  - 1) Batch mode: load many data files, then press the ‘compare’ button
    - This will compare each loaded file against all other files
  - 2) Standard or single file mode: Load a **single** data file in the table, then press the ‘compare’ button.
    - This will open a filechooser to choose other .ciu files to compare against the single loaded file. All chosen files will only be compared against the “standard” file (the one loaded in the table) and not against each other.
    - This is equivalent to the “batch” mode in the original CIUSuite software

Data Analysis: RMSD Comparison

- Parameters:
  - Relative Intensity Cutoff: (relative ratio, max intensity = 1.0)
    - Data below this value will NOT be included in the RMSD calculation. This is intended to remove instrument/electronic noise that can cause minor fluctuations in intensity in drift bins with no actual signal, which would impact RMSD values if included.
    - NOTE: changing this value WILL change the calculated RMSD. Comparisons between calculated RMSD values can ONLY be done using the same intensity cutoff value.
  - High Contrast Mode:
    - Normalizes to the largest RMSD difference observed to make all colors brighter (higher contrast) on the comparison plot generated. No change to the underlying data.
  - Custom Label for File 1/2:
    - Optional labels that can be added to the colorbar legend.
    - Useful to keep track of which color represents which data file in single/standard analysis mode.
    - NOTE: in batch mode, order of the files (which one is file 1 and which is file 2) is not guaranteed. “Compare in both directions” (below) can be used to ensure the desired label order is generated.
  - Compare in both directions:
    - If True, will generate an output plot for File 1 – File 2 AND File 2 – File 1

Data Analysis: Feature Detection

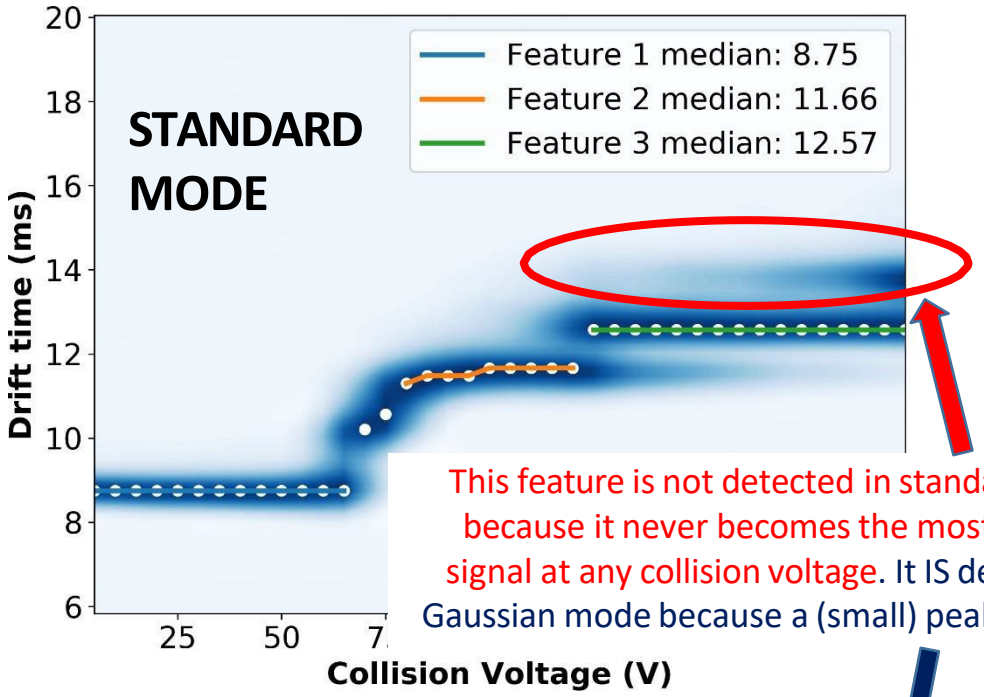
- Features are detected by looking for “flat” regions of a fingerprint
  - “Flat” meaning that the drift time observed is constant across a range of activation energies
- Only the apex of the arrival time distribution at each collision voltage in considered in “standard” mode
  - The apex values are shown as the white circles on the features plot



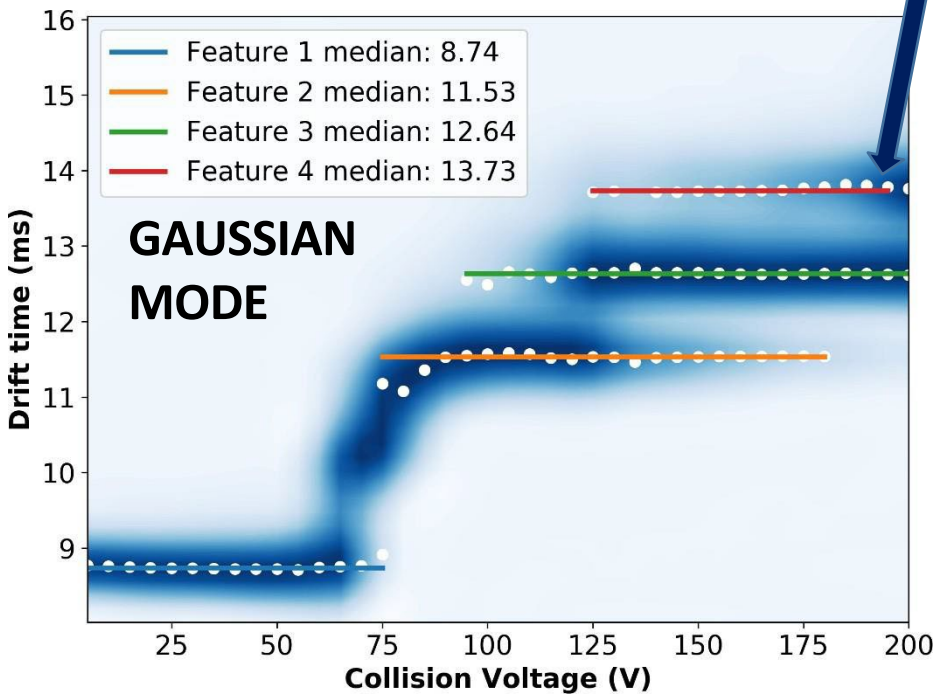
Data Analysis: Feature Detection

Gaussian vs Standard mode

- "Standard" mode uses apex drift time at each collision voltage used to determine features
- "Gaussian" mode uses fitted Gaussian centroids instead of raw apex drift times
  - This means that features can overlap (more than 1 feature can be present at a given collision voltage). This is a key difference between Gaussian and standard modes: **Gaussian mode detects all voltages where features are present, whereas standard mode only detects voltages where a feature is dominant** (more intense than all other features).
  - Gaussian mode will also show features that never become the most intense signal, like the 4th feature (red) in the example at right. It is not detected in standard mode because it never reaches 100% relative intensity.
- Gaussian mode requires that Gaussian fitting has previously been performed on the data



This feature is not detected in standard mode because it never becomes the most intense signal at any collision voltage. It IS detected in Gaussian mode because a (small) peak is present



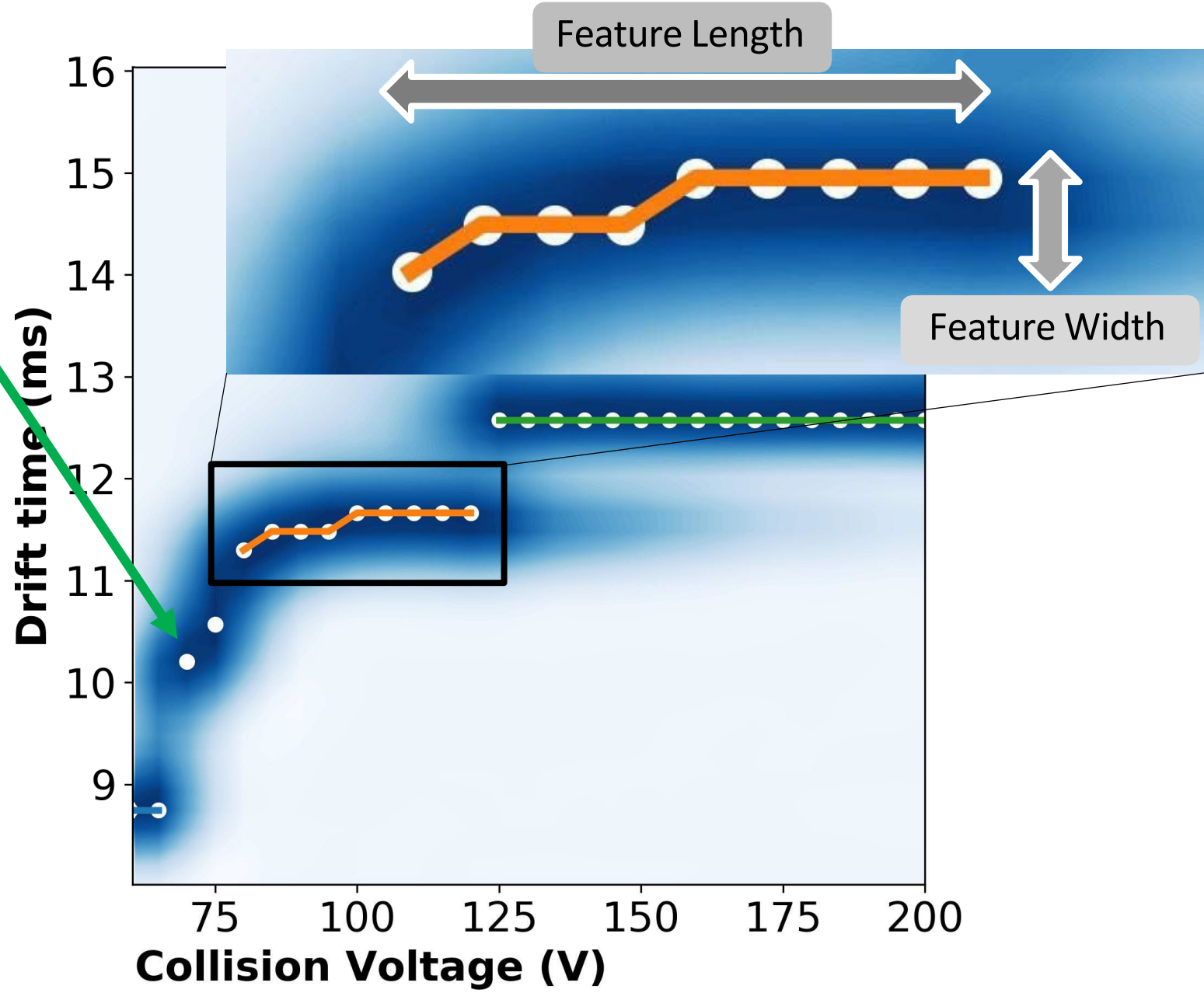
Data Analysis: Feature Detection - Parameters

Minimum Feature Length (steps):

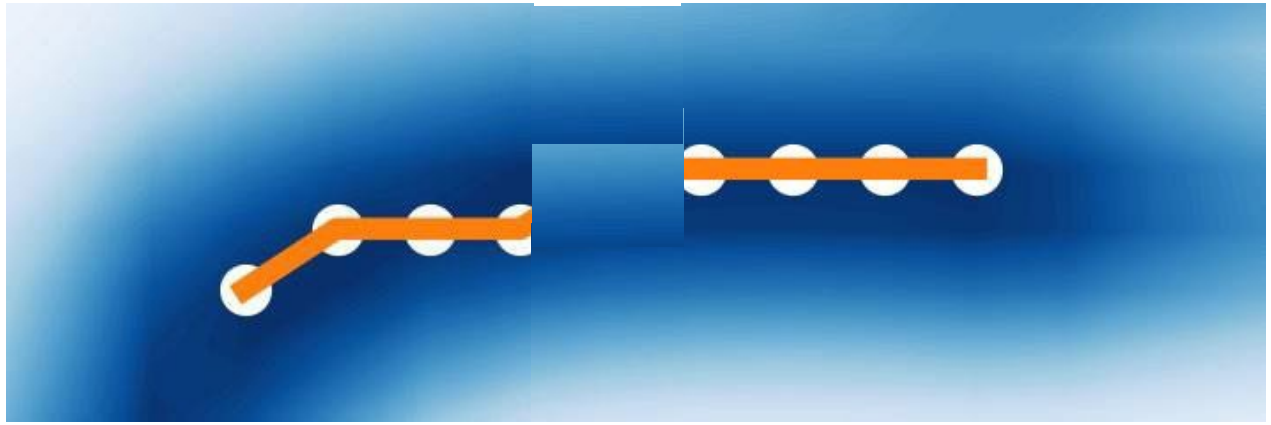
- The minimum number of collision voltages across which a feature is stable to be counted as a featur e
- The yellow feature has length 9, because it include 9 voltages (from 85 to 125 V in 5V steps)
- The two dots to the left of the example feature are NOT considered a feature by themselves in this example because the minimum length was set to 4
- *NOTE: if you want to detect a feature that has a length of only 1-2 steps, interpolate along the activation axis to increase the number of steps*

Feature Allowed Width:

- The distance from the feature median (in mobility axis units) a point can be and still be considered part of the feature.
- For example, if the width was reduced, the left-most point of the example feature (at lower drif time) could be excluded for being too different from the other points. Or if the width was increased, the two points not included in any features could be included in the yellow feature



Data Analysis: Feature Detection - Parameters



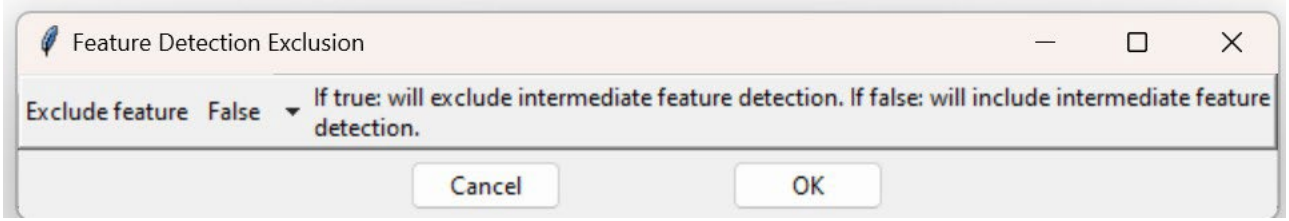
Maximum CV Gap Length (steps)

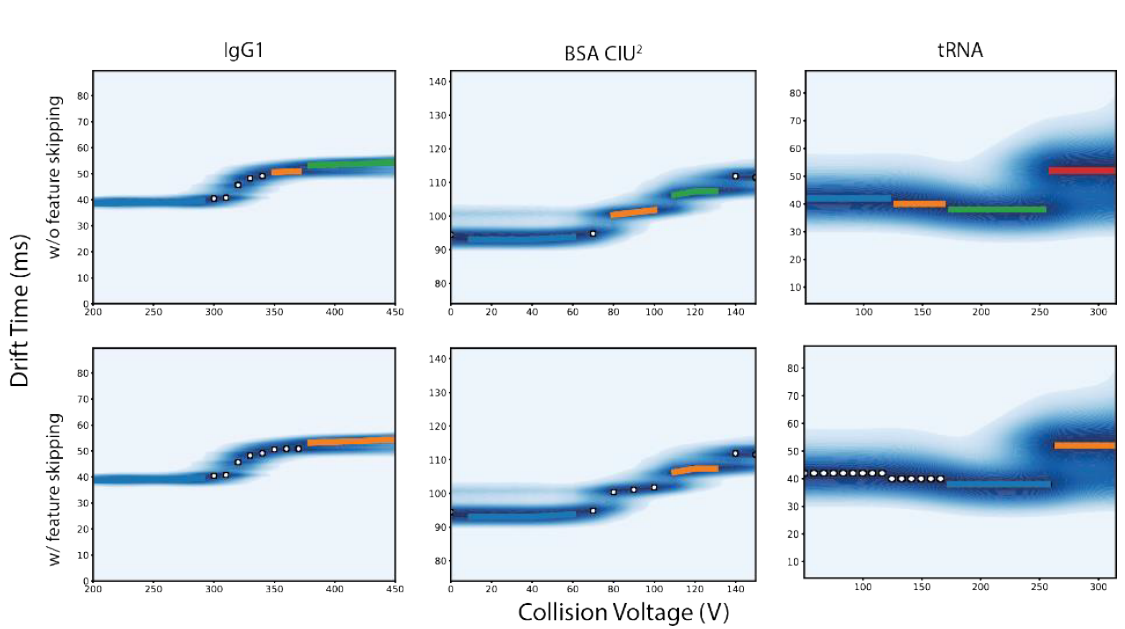
- The maximum number of steps (along the activation axis) between points of a feature before it is considered a break between features
- The points on either side of a gap must still fall within the allowed width to be considered part of a feature
- Gap length is generally more important in Gaussian mode than standard mode

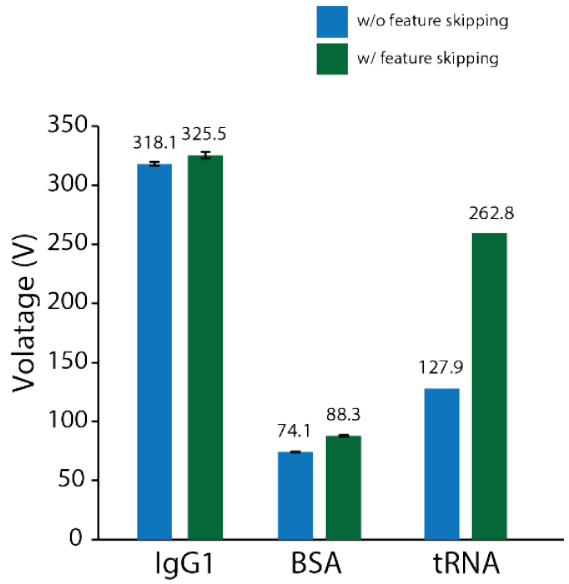
- A ‘gap’ of 1 step is present in between parts of this feature. If the Maximum Gap Length is 2 (or more), this will be detected as 1 feature
- If the Maximum Gap Length is 1, this will be detected as 2 different features because of the interruption from the gap

Data Analysis: Feature Detection – Skipping

After setting up the parameters, another popup prompt will show up (right). This will ask whether the user would like to skip features. If set to True, it will skip all middle features and only detect minimum and maximum DT features. If set to False, it will not perform feature skipping. See below for examples of feature skipping.
Feature skipping will affect the downstream CIU50 analysis.

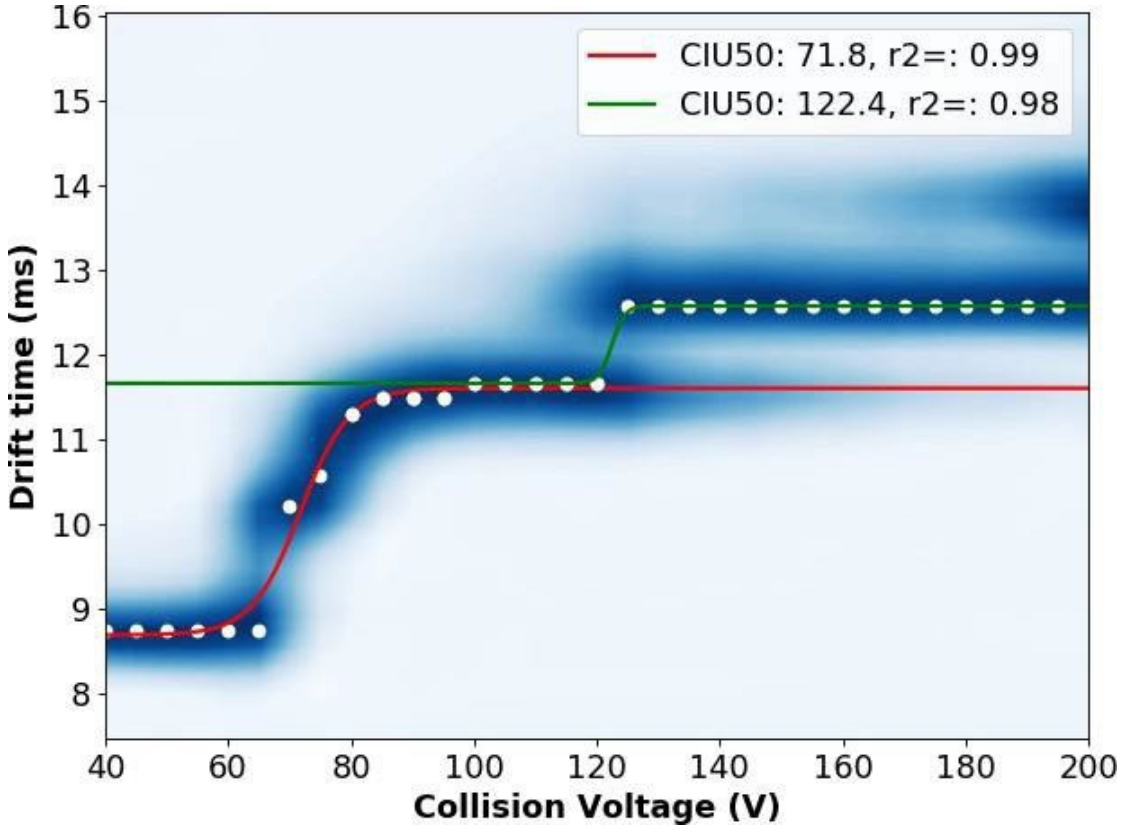






Data Analysis: CIU50

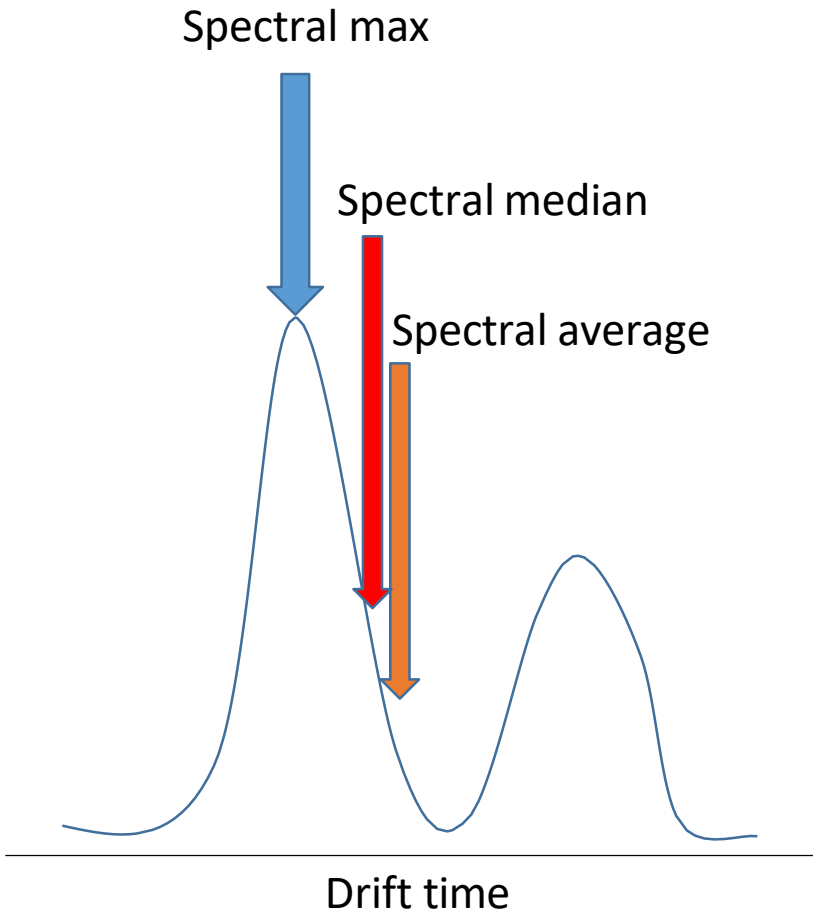
- CIU50 analysis fits a sigmoid (logistic) function to the features detected by Feature Detection
- “CIU50” refers to the midpoint of the transition between two features



Data Analysis: CIU50 Parameters

Spectral Centroiding Mode:

- Max: (Default) In max mode, each collision voltage is reduced to just the location (drift time) of the most intense peak. This is equivalent to centroiding the drift time peak by taking the max value.
  - Max mode is the most robust because it is generally insensitive to noise
- Average and Median: In some cases (e.g. if features overlap at similar intensity across many voltages), different centroiding can be helpful. In average/median mode, the arrival time distribution is centroided to the average or median, respectively, of the distribution at each collision voltage.
  - This generally produces much less steep transitions, and can result in poor fits (as transitions may no longer resemble sigmoids)
  - NOTE: average and median include ALL data in the arrival time distribution, and should NOT be used if more than two features overlap at any point (or if significant noise is present), as the presence of other features/noise will affect the transition data



Data Analysis: CIU50 Parameters

Transition Region Padding:

- CIU50 fitting determines transition regions using the boundaries of detected features. Additional data is included in the actual logistic function fitting to ensure fidelity to the raw data, which is called the ‘padding’.
  - Generally, this parameter should be equivalent to 2-4 collision voltage steps and does not need to be adjusted. In some cases, where fitting is poor (or if using average/median centroiding), adjusting the size of the padding region can improve fit quality.

Standard vs Gaussian Mode:

- Just like Feature Detection, CIU50 analysis can use Gaussian data.
  - Unlike feature detection, Gaussian mode CIU50 analysis is not significantly different from standard mode (peak max values of the raw data are used rather than Gaussian centroids), but uses the Gaussian feature data correctly to set up the fitting.
  - **NOTE: transition fitting ONLY considers features that reach 100% relative intensity, so some features detected by Gaussian fitting may not have a transition fit to them. This is because they have not yet reached the CIU50 point in the transition from the previous feature.
  - NOTE: performing standard mode CIU50 analysis on Gaussian feature detection results will likely cause a crash

Data Analysis: Classification

- Classification works in two steps:
  - 1) Train a classifier using known data (i.e. data that you know can be grouped into certain classes)
  - 2) Use the classifier to classify unknown data into the groups you set up

- Multi-state classification
  - The training data can be divided into multiple sets (or states), that correspond to the same class under different conditions
    - The most common example is charge states. Different charge state CIU fingerprints generally cannot be directly compared but can be included as states for a multi-state classifier.

- Multi-dimensional classification
  - The training data can utilize CIU data from various dimensions (CIU^1 and CIU^2). This feature is currently only available for Waters cyclic IM-MS platform.

Data Analysis: Classification

- Training a Classifier – data input
  - To build a classification scheme, click the ‘Build Classification’ button
  - There are 3 input methods: “prompt”, “table”, and “template” (see the following slides for details on each)

  - NOTE: The data files chosen for each class are crucial to the success of the classification. Generally, having more replicates for each class is better – because more of the possible variation within each class will be accounted for in the resulting classification scheme.
    - ****The differences between classes generally need to be larger than the differences within classes for a classification scheme to be successful!****
      - Thus, if the replicates (within a class) have substantial differences between them, it may be challenging to generate a robust classifying scheme

Data Analysis: Classification

- Training a Classifier – **data input**
  - **Prompt Mode**:
    - A pop will ask for the number of classes you want to classify
    - Then, a series of popups will ask for:
      - The name of Class 1
      - The data files corresponding to Class 1
      - The name of Class 2
      - The data files corresponding to Class 2
      - (etc. if more classes were specified)
    - Note: to input multiple states in prompt mode, data files must be labeled according to their state. If specifying multi-state mode, a dialog box will ask for the names of the states being used and will search for those names (exactly) in the names of each input data file.

Data Analysis: Classification

- Training a Classifier – **data input**
  - **Table Mode**:
    - Data files loaded in the table (the list of loaded files in the GUI) will be used to train a classifier. Popups will ask for the names of each class (and states if using multi-state) and search the names of each loaded data file to assign them to classes/states.
      - This means that data file names must exactly include the class name:
        - For example:
          - Classes: “IgG1” and “IgG2”
          - States: “22+”, “23+”, and “24+”
          - Data file “2019_Jan-01_IgG1_22+.ciu” will be assigned correctly
          - Data file “2019_Jan-01_igg1_22+.ciu” will NOT be assigned a class because the string “IgG1” (case sensitive) does not appear in the filename
          - Data file “2019_Jan-01_IgG1_22.ciu” will NOT be assigned a state because the string “22+” does not appear exactly in the filename.

Data Analysis: Classification

- Training a Classifier – **data input**
  - **Table Mode**:
    - Data files loaded in the table (the list of loaded files in the GUI) will be used to train a classifier. Popups will ask for the names of each class (and states if using multi-state) and search the names of each loaded data file to assign them to classes/states.
      - This means that data file names must exactly include the class name:
        - For example:
          - Classes: “IgG1” and “IgG2”
          - States: “22+”, “23+”, and “24+”
          - Data file “2019_Jan-01_IgG1_22+.ciu” will be assigned correctly
          - Data file “2019_Jan-01_igg1_22+.ciu” will NOT be assigned a class because the string “IgG1” (case sensitive) does not appear in the filename
          - Data file “2019_Jan-01_IgG1_22.ciu” will NOT be assigned a state because the string “22+” does not appear exactly in the filename.

Data Analysis: Classification

- Training a Classifier – **data input**
  - **Template Mode**:
    - A template file (csv) is used to tell the program where to find .ciu files and what classes and states (if using multi-state mode) to assign.
    - NOTE: like table mode, template mode searches for class/state names in the name of the data file, and will not load any files that do not contain an exact (case-sensitive) match for the class/state name
    - An example template file is included with CIUSuite 3 and can be found in the installation directory
    - Template file formatting: (comma-separated text file)
      - Line 1: class names. Must start with "classes", then each class name (case-sensitive) separated by commas
      - Line 2: (optional) state names. Must start with "states", then state names. Providing only 1 state name will run a single state classifier.
      - Line 3 (or line 2 if not including states): folder path. Full system path to the folder(s) in which to find ALL .ciu files to include in all classes/states. Can include multiple folders, separated by commas on this line. All folders will be searched for all classes – putting different class CIU files in different folders will NOT automatically assign them to the correct classes

Standard Mode (single state)

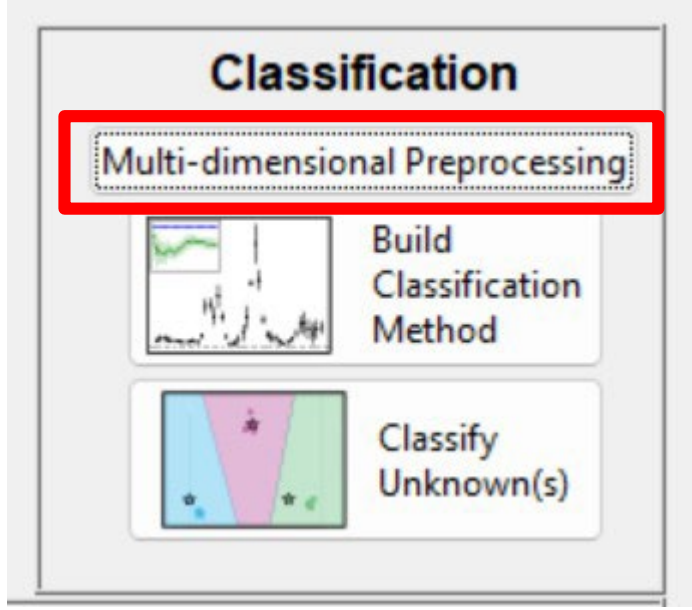
	A	B	C	D	E
1	classes	IgG1	IgG4		
2	folder	C:\Data\ClassificationCIUFiles\			
3					

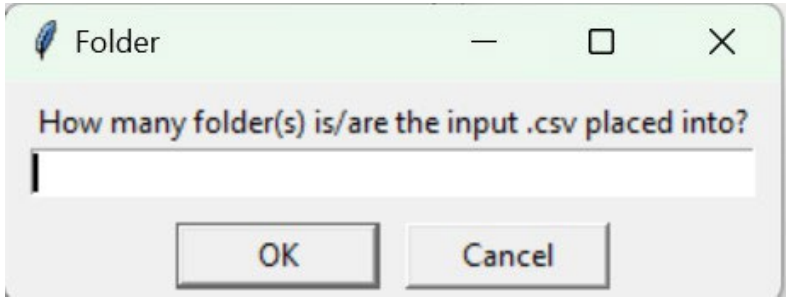
Multi-state

	A	B	C	D	E	F	G
1	classes	IgG1	IgG4				
2	states	22+	23+	24+	25+	26+	
3	folder	C:\Data\ClassificationCIUFiles\					

Data Analysis: Classification

- Multi-dimensional classification
  - To correct the dt offset in CIU^2 data, multi-dimensional classification needs some data processing for proper classification.
  - A new button (in red) is added to process all CIU data (both CIU^1 and CIU^2) prior to classification.
  - First pop up will ask how many folders the .csv file you plan to use for classification are available.
  - In the following pop ups, you'll be asked to select all the files for preprocessing.
    - i.e. If you entered "2" in the previous window, you'll have two pop ups for file selection.
  - Once completed, you'll find "[original file name]_multidimensional_raw.csv" in the folders you selected your original data. Use these .csv for downstream classification analysis.





Data Analysis: Classification Parameters

Data Mode: 3 input modes are available

- **All_data:** The entire CIU dataset is used for classification. Each feature is the complete mobility profile at each collision voltage.
- **Gaussian_Raw:** Gaussian fitting results are used for classification WITHOUT feature detection. The Gaussian centroids, widths, and amplitudes are used as features at each collision voltage (for however many Gaussians are fit).
- G**aussian_Feat:** Gaussian fitting results filtered by Gaussian Feature Detection are used as the input to classification. Feature detection is useful for reducing variation in poorly fit Gaussians, ensuring that only signals that persist across several collision voltages are considered for classification. This is particularly helpful when attempting to classify noisy datasets using the Gaussian de-noising workflow.

Data Analysis: Classification Parameters

Cross Validation Score Tolerance:

- This is a minor heuristic added to reduce overfitting. Generally, the highest score (most accurate classifier) is used. In some cases, there may be a classifier with similar accuracy but fewer features. If the score of the scheme with fewer features is within this tolerance of the score of the classifier with more features, the scheme with fewer features will be chosen.

Feature Selection Mode:

- Manual selection of the voltage(s) to use in the classifier is possible by switching this from automatic to manual.

Data Analysis: Classification Parameters

Cross Validation Score to Use

- Both accuracy of cross validation (test data) and AUC (area under the ROC curve) can be used to choose the optimal classifier.

Show AUC on CrossVal Plot:

- Whether to show the AUC along with accuracy on the cross validation output plot (cosmetic only – no change to data processing).

Use Feature Error in Auto Selection:

- If true, the variance of feature (UFS) scores will be considered when sorting features for classification. If true, sorting is done by (feature mean – variance), vs sorting just by feature mean if false. Visually, this sorts features by the bottom of the error bar on the UFS plot rather than the dot at the mean.

• (Optional) Maximum # Voltages to use in Scheme:

- Maximum number of features to consider in cross validation. Can be set to reduce computation time, especially in single-state classifiers.

Data Analysis: Classification Parameters

(Optional) Maximum Iterations for Cross Validation

- Number of randomly sampled data configurations to test for cross validation. This prevents large datasets from taking excessive time/memory to cross validate.

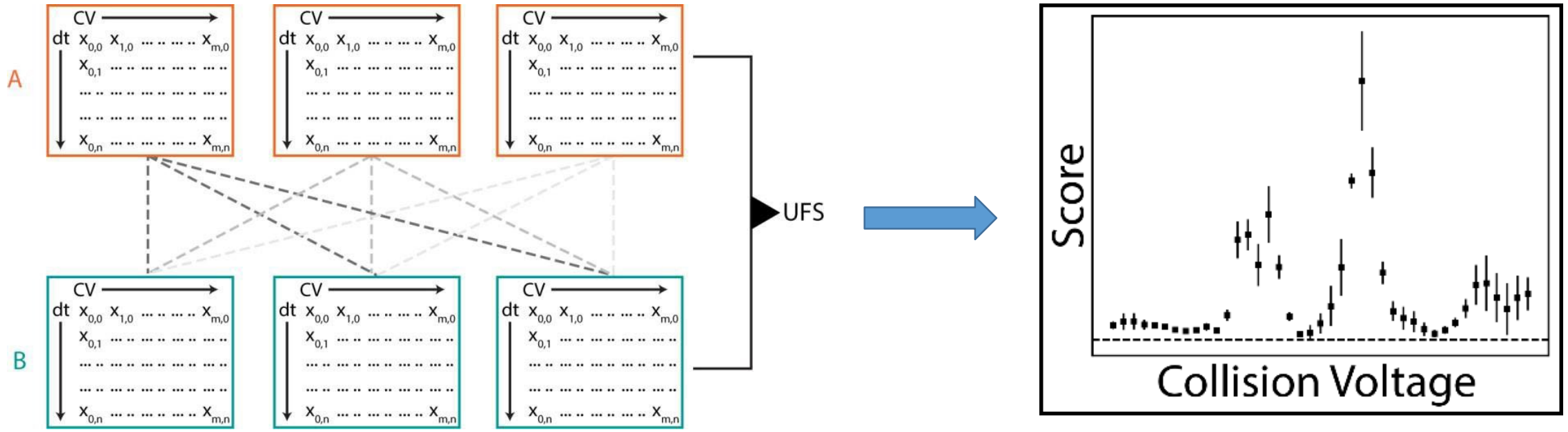
- Cross Validation Test Size
  - The number of CIU data files (replicates) to withhold as test data for cross validation. This should be adjusted to appropriate size (e.g. 10-30% of the input size for each class/state) to ensure reasonable estimation of classifier accuracy. Increasing the test size will make it more challenging for the classifier (resulting in lower accuracy), but can be a better estimation of its accuracy for unknown data (i.e. data not included in the training set).
- (Advanced) Standardize Data
  - By default, data is standardized to 0 mean and unit variance across the entire input dataset prior to classification. This is very helpful for training an appropriate classifier and should almost always be used.

Data Analysis: Classification Parameters

- (Advanced) Standardize Gaussians Together
  - Recommended. All Gaussian peak characteristics (centroid, width, and amplitude) are standardized across all Gaussian peaks present
    - That is, all centroids are standardized together, all widths together, and all amplitudes together
  - This typically results in good standardization. However, a more precise procedure (in some cases) would be to standardize each Gaussian separately (i.e. “Gaussian 1” centroid is considered separately from “Gaussian 2” centroid). The current implementation (if this option is set to False) does NOT align the peaks, however, so which peak is “Gaussian 1” will change across the course of a fingerprint…making this mode not useful unless Gaussian features are extremely consistent across all classes/replicates.

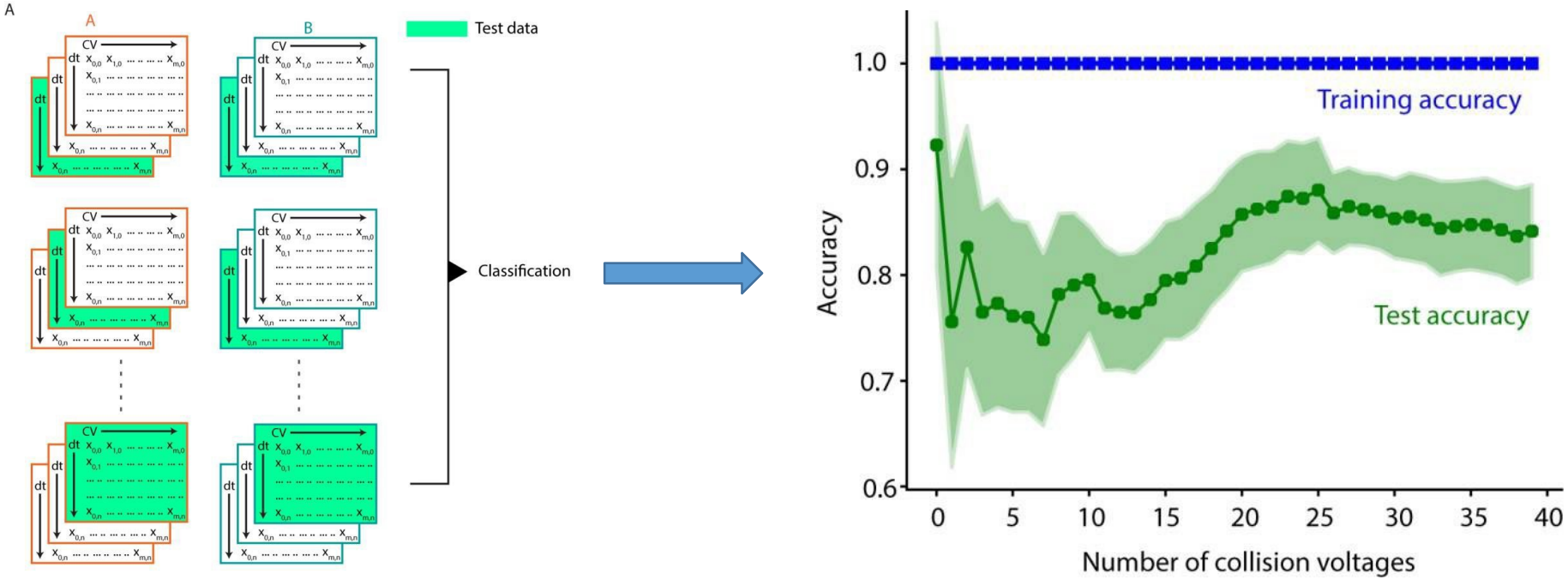
Data Analysis: Classification Diagnostics

- After a scheme is constructed, several diagnostic outputs are generated
  - UFS output: Voltage Selection plot
    - UFS (univariate feature selection) compares inter- and intra-class variation at each collision voltage using F-tests. (F = inter-class variation / intra-class variation)
    - An output plot of UFS score is generated as in the lower right. The score ($-\log_{10}$ (p-value)) for each collision voltage is displayed with error bars. The higher the score, the more difference between classes at that voltage relative to the differences within classes at that voltage.



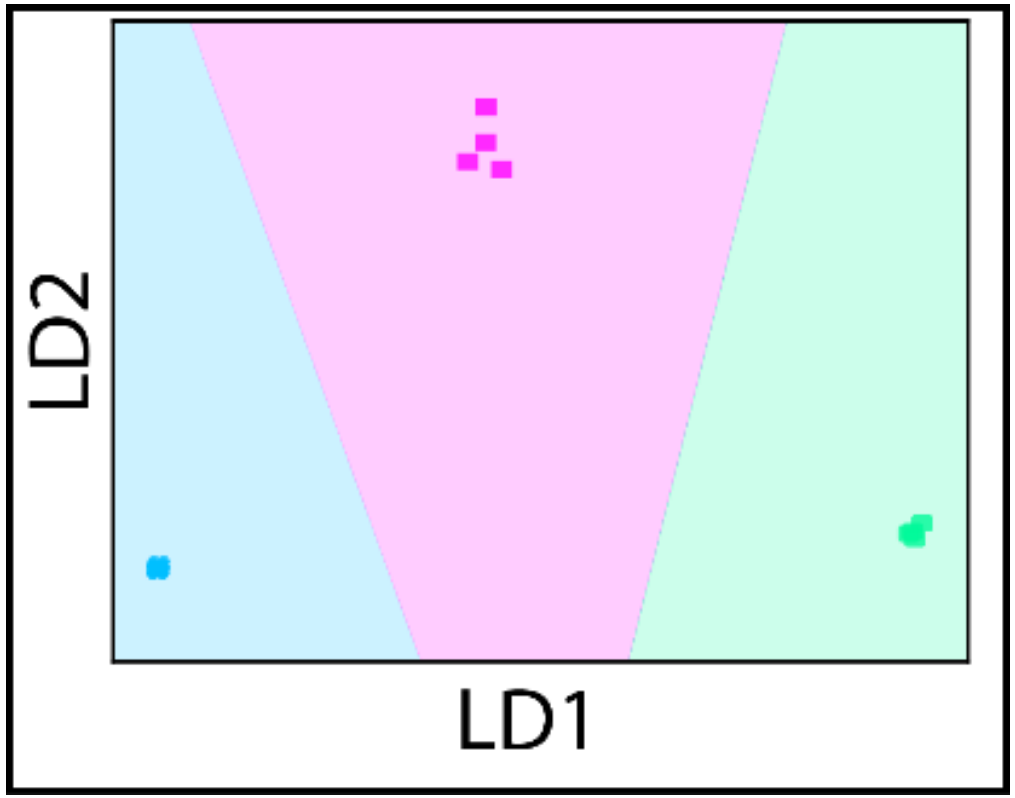
Data Analysis: Classification Diagnostics

- Cross Validation plot
  - “Leave one out” cross validation (illustrated below left, each input takes a turn being the test data) is used to evaluate the accuracy of the classification at each possible number of voltages included.
    - Voltages are added to the classification in decreasing order of score from the UFS output (see previous page)
  - A cross validation plot showing the training (blue) and test (green) accuracy with error bars (shaded) as a function of the number of voltages used is produced, as in the lower right. The highest test accuracy is chosen as the final number of voltages for the scheme.
    - Except if the highest accuracy is within tolerance (default 0.05) of the accuracy of a lower number of voltages, the scheme with the lower number of voltages will be used instead. Reduce tolerance to 0 to avoid this behavior.



Data Analysis: Classification Diagnostics

- Decision Region (classification) plot
  - The linear discriminant space used for classification is divided into regions corresponding to each class, and the locations of the training data within the space are shown (for up to 4 classes. Above 4 classes, the plot would be more than 3D and thus challenging to generate..).
  - In the example at right, 3 classes are used (generating a 2D linear discriminant space). Cross validation test data results are shown in the colored squares.

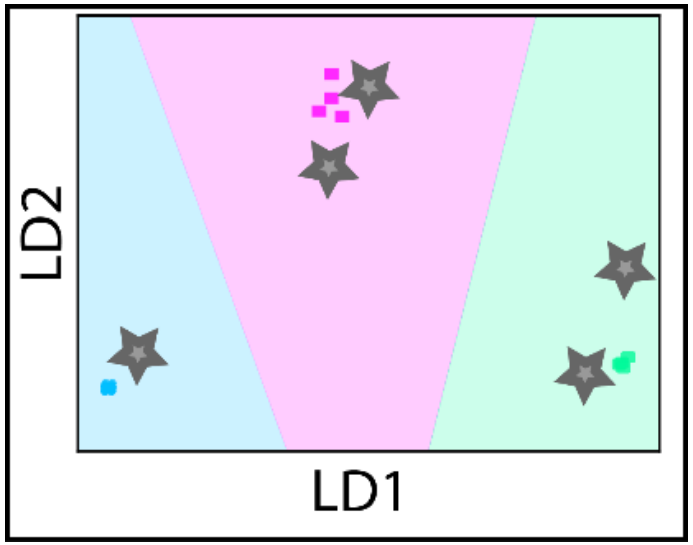


Data Analysis: Classification Diagnostics

- How to determine if you have a “good” classifier
  - After a classifier is trained, look at the outputs, particularly the cross validation plot, ROC curve(s), and feature selection (UFS) plot.
  - The most important indicators are the voltages chosen from the UFS plot and the cross validation accuracy.
    - High cross validation accuracy is an excellent indication of a good classifier IF the cross validation is accurately representing the variation within the data. Larger test data sizes and inclusion of both biological and technical replicates help make the accuracy reported more realistic relative to what to expect when classifying unknown data. If there are sources of variation in the unknowns that are not included in the training data, it’s possible to greatly overestimate the ultimate accuracy of the classifier.
    - If the voltages chosen by the feature selection make sense (by corresponding to region(s) of the CIU fingerprint that clearly differ between classes) that’s a good sign. If there aren’t really any clear regions of difference between the classes (no features with error bars that clearly don’t extend below 0), the classifier is likely to perform poorly.
  - The best way to evaluate any trained classifier is external validation: test the classifier with new/other data that was NOT included in the training and see if the output is classified correctly.

Data Analysis: Classifying Unknowns

- Once a classification scheme is constructed, a .clf file is saved (the names of each of the classes + .clf).
- To classify unknowns, load their .ciu file(s), then press the 'classify unknown' button.
  - A popup will ask you to select the classifying scheme (.clf) file to use to classify
- Each unknown will generate a probability to be in each class, and a decision regions plot will show the locations of unknowns relative to the classifying regions
- NOTE: if the classifying scheme used Gaussian data, the unknowns must have Gaussian feature detection performed prior to classification
- NOTE: The unknowns ONLY need to have data at the voltages selected in the classification scheme (but can be full fingerprints as well)



Stars correspond to unknowns, squares to training data