py_diAID User Guide

Developed by: Patricia Skowronek, Eugenia Voytik, Sander Willems, and Matthias Mann.
This step-by-step guide presents instructions for the installation and usage of py_diAID.

Table of Contents

Description

py_diAID (Automated Isolation Design for Data-Independent Acquisition) is a Python package that streamlines the generation of data-independent acquisition (DIA) methods for mass spectrometry-based proteomics.

Supported scan modes

dia-PASEF

- Comprehensive proteome coverage
- High degree of reproducibility and quantitative accuracy
- Uses a much larger ion beam proportion than conventional DIA methods [1,2]
- Advantageous for studying deep proteomes from cell lines, clinical samples with regular and very low sample input, as well as for exploring post-translational modifications such as phosphorylation [2, 4, 5]

synchro-PASEF

- Successor of dia-PASEF
- More efficient precursor sampling leads to shorter cycle times than dia-PASEF improving quantitative accuracy
- Linking of fragment signals with precursor masses increases specificity [3]
- Unites the benefits of DDA and DIA

Orbitrap Astral DIA

- Comprehensive proteome coverage
- High degree of reproducibility and quantitative accuracy
- Window schemes can be generated with fixed or dynamic window widths
- Optimal window border placment within the forbidden zone

Key features of py_diAID

- Automatically generates optimal isolation windows
- Supports variable/dynamic isolation widths. Widths are aligned to the precursor density, enabeling short acquisition cycles while covering virtually the entire m/z (-ion mobility) range
- Facilitates quality control of measured samples by visualizing the precursor distribution
- Allows the evaluation of existing dia-PASEF and synchro-PASEF methods
- Available as Python module, as well as a Graphical User Interface (GUI), under an Apache 2.0 license

1. Meier, F. *et al.* diaPASEF: parallel accumulation–serial fragmentation combined with data independent acquisition. *Nat. Methods* **17**, 1229–1236 (2020).
2. Skowronek, P. *et al.* Rapid and In-Depth Coverage of the (Phospho-)Proteome With Deep Libraries and Optimal Window Design for dia-PASEF. *Mol. Cell. Proteomics* **21**, 100279 (2022).
3. Skowronek, P. *et al.* Synchro-PASEF Allows Precursor-Specific Fragment Ion Extraction and Interference Removal in Data-Independent Acquisition. *Mol. Cell. Proteomics* **22**, 100489 (2023).
4. Rosenberger, F. A. *et al.* Spatial single-cell mass spectrometry defines zonation of the hepatocyte proteome. *Nat. Methods* **20**, 1530–1536 (2023).
5. Thielert, M. *et al.* Robust dimethyl-based multiplex-DIA doubles single-cell proteome depth via a reference channel. *Mol. Syst. Biol.* **19**, 2022.12.02.518917 (2023).

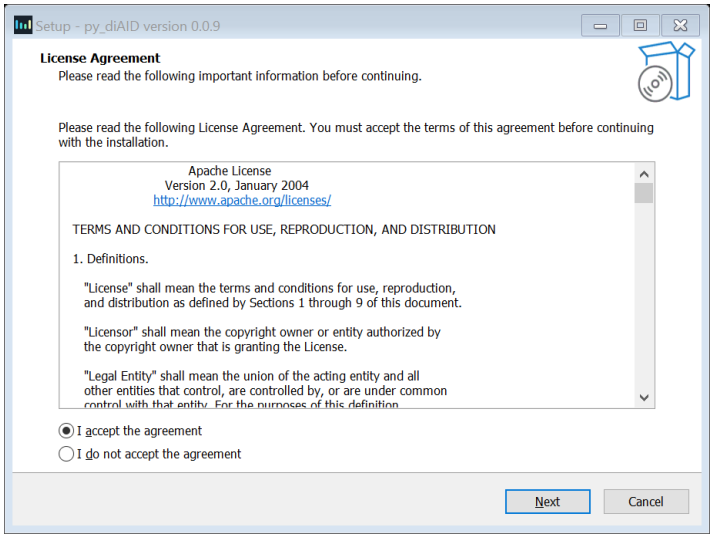
Installation

We offer py_diAID as a browser-based GUI and a Python package. All modes are also described on the GitHub repository (https://github.com/MannLabs/pydiaid).

Windows

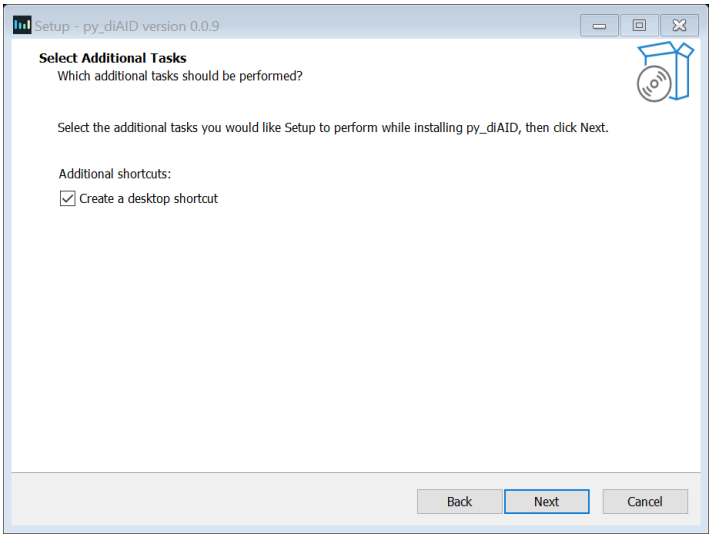
Windows 10 is required to install py_diAID, and you need admin privileges if you want to run py_diAID for all users (right-click on the py_diAID logo on your desktop and select "Run as administrator"). We recommend uninstalling any previous py_diAID versions before updating py_diAID to prevent installation errors.

1. Download the latest release of the package for Windows (pydiaid_gui_installer_windows.exe) from the GitHub repository and execute the .exe file.
2. If the "User Account Control" dialog asks you for permission for the app to make changes to your device, please press the "Yes" button.
3. We recommend selecting "Install for me only (recommended)" if a "Select Setup Install Mode" dialog window shows up.
4. Please accept the License Agreement and click "Next" in the "Setup – py_diAID version X.X.X" dialog window.

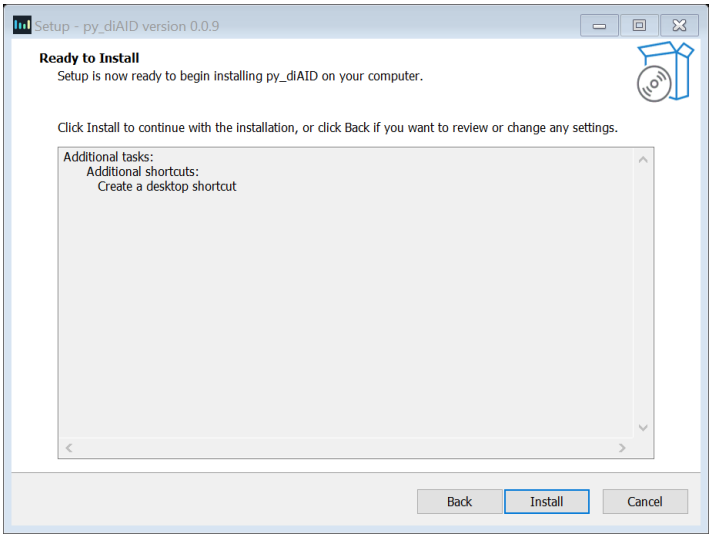


5. In the "Select Additional Tasks" dialog window, activate the "Create a desktop shortcut" check box

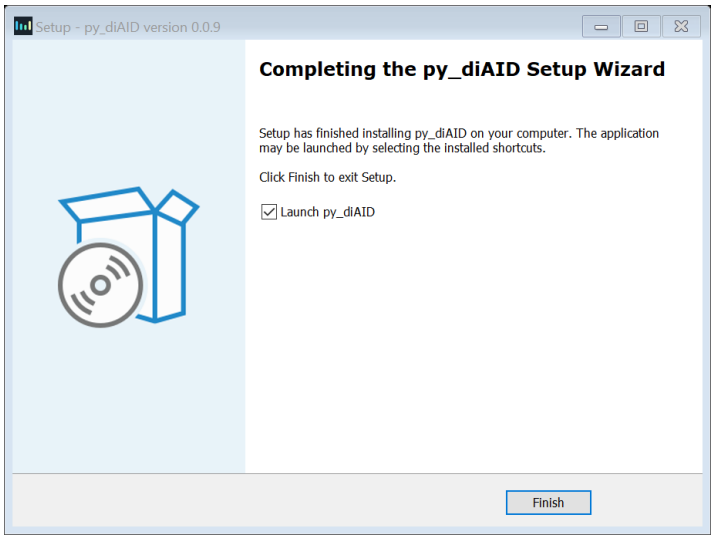
and click the “Next” button.



6. Check the settings and press “Install”. You can change the setting with the “Back” or “Cancel” button prior to the installation.



7. Please wait until the installation is finished, activate the “Launch py_diAID” box, and click “Finish”.



8. If a “Window Security Alert” window appears, press “Allow access” to prevent the Windows Defender Firewall from blocking py_diAID on your PC.

MacOS

macOS Big Sur (11) or higher is required to install py_diAID.

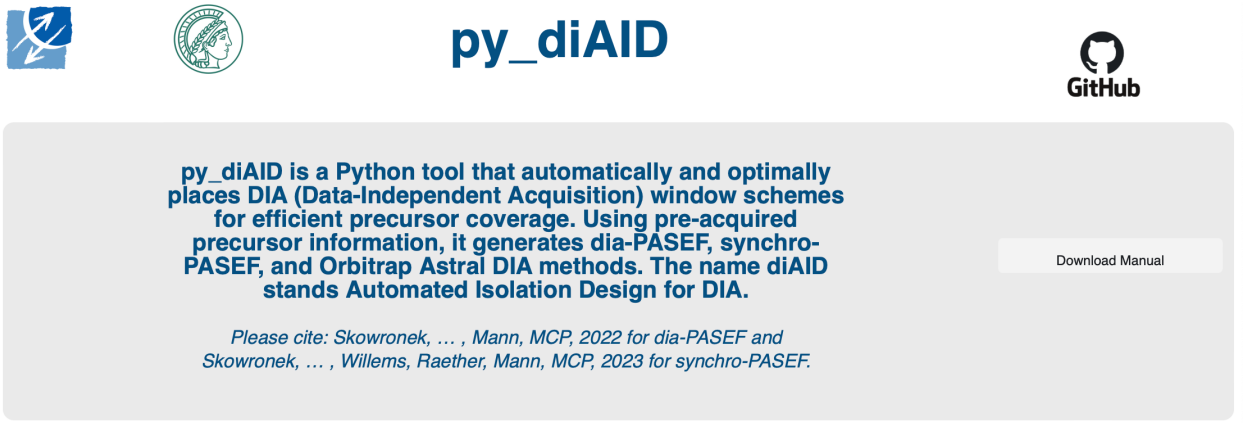
1. Download the latest release of the package for macOS (pydiaid_gui_installer_macos.pkg) from the GitHub repository and open the .pkg file. In rare cases, an error message shows up, which prevents the file from opening due to insufficient priveleges In this case, you can close the message by clicking “OK”, go to the “Security & Privacy” section of the “System Preferences” menu and press “Open Anyway” at the “General” tab.
2. Click “Continue” on the “Install py_diAID X.X.X” dialog window.
3. The License sections will show the Software License Agreement (Apache License 2.0). To continue the installation, press "Continue" button, and in the pop-up window you need to agree with the regulations of the license.
4. Please press “Install” to start the installation. This might take several minutes.
5. After the installation is finished, click “Close”. py_diAID is now available in the applications folder on your MacOS.

Linux

Py_diAID can be used with Linus by installing it as a Debian package, which requires the Apache License 2.0’s acceptance.

1. Download the latest release of the package for Linux (pydiaid_gui_installer_linux.deb) from the GitHub repository.
2. Run the installer either by double clicking it, or by executing the command <*sudo dpkg -i pydiaid_gui_installer_linux.deb*> (copy everything between <>).

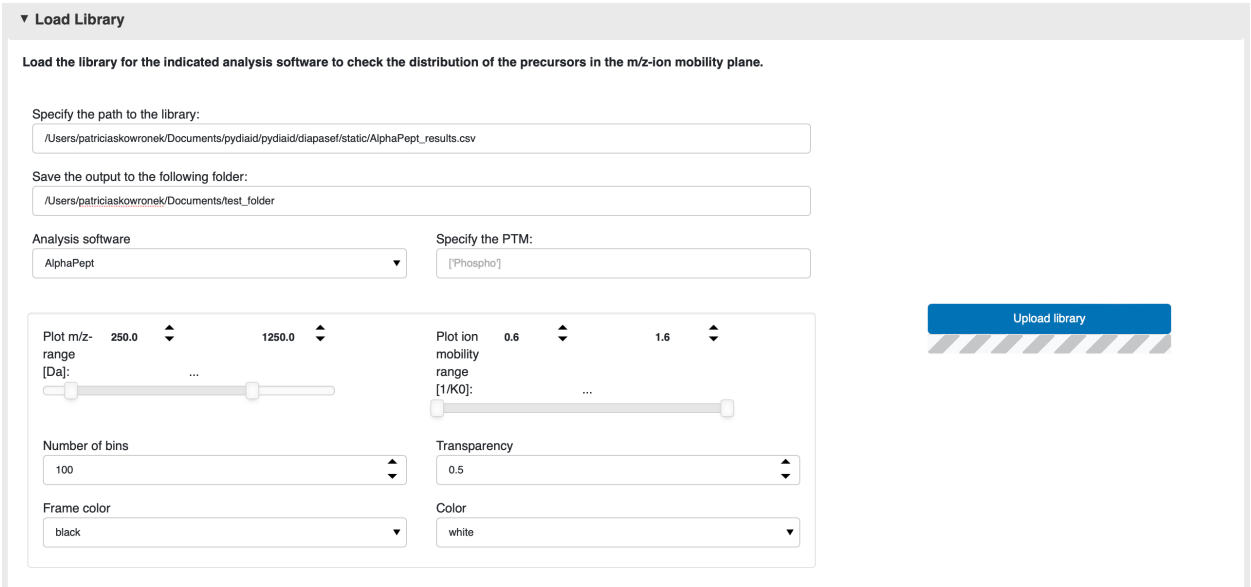
How to use py_diAID



When you launch py_diAID, a terminal window opens, displaying all the background information about py_diAID, and a new browser tab titled “py_diAID X.X.X” opens in your default browser. You can terminate py_diAID by pressing “Ctrl+C” in the active terminal window. We recommend using either Google Chrome or Mozilla Firefox for the best performance. However, py_diAID does not require an internet connection as it runs directly on your local system.
Py_diAID allows the generation of optimal dia-PASEF, synchro-PASEF and Orbitrap Astral DIA methods, indicated by separate tabs. First, we will explain the steps for generating an optimal dia-PASEF method.

Optimal dia-PASEF methods

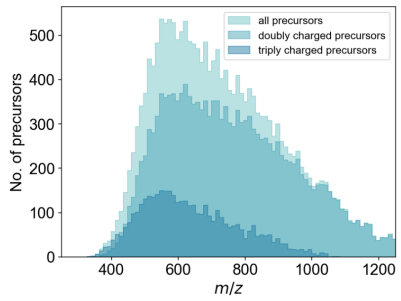
Load a library



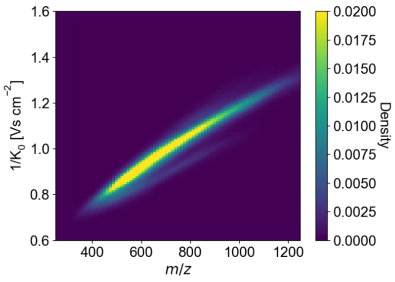
- Provide the filepath to the library (i.e., proteomics dataset) analyzed by AlphaPept, MaxQuant, MSFragger, Spectronaut (single-run or library), DIA-NN (single-run or library), AlphaPeptDeep library, AlphaDIA. Important: Indicate the software used in the drop-down menu labeled “Analysis software”.
- Specify the unique characters related to the post-translational modification of interest using a list. This function filters the column for modified peptide names based on the specified strings in the list. For Spectronaut, [“STY”] would be suitable for filtering phosphopeptide precursors and [“UniMod:28”] for FragPipe.
- The following settings apply to all plots generated by py_diAID: “Plot m/z-range” and “Plot ion mobility-range” establish the limits of the m/z and ion mobility axes in all plots. py_diAID calculates and plots kernel density estimates to represent the precursor distribution. The “Number of bins” influences the resolution of these plots. We recommend 100 bins for efficiency and up to 300 bins for sharper figures; however, this might take up to 30 minutes. “Transparency,” “Frame color,” and “Color” are properties of the isolation windows that are plotted to visualize the dia-PASEF methods.

A test library can be found at the following path: *<py_diAID installation directory>*\ *pydiaid**diapasef**static**AlphaPept_results.csv*. It represents a single-run from 200 ng tryptic HeLa digest acquired with 21-minute Evosep gradients and dda-PASEF.
Press the “Upload Library” button to visualize the precursor distribution of the library/dataset with a histogram over m/z, a kernel density plot, and the charge state ratios. You can use these plots to check the ion mobility distribution of your samples. This is the only required step to activate the “create” and “evaluate” buttons.

Histogram: Precursor distribution in m/z



Precursor cloud plotted across m/z and ion mobility



Percentage of multiple charged precursors

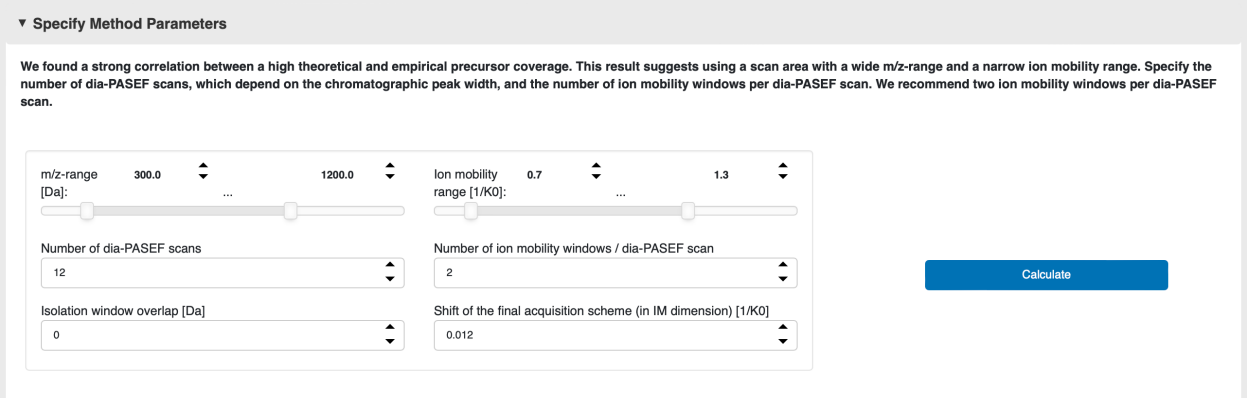
index	charge state of precursors	ratio [%]
0	1	0.91
1	2	77.69
2	3	20.19
3	4	1.19

Analysis software instructions

- *AlphaPept:* Please use the **results.csv** file. py_diAID uses the columns “q_value”, “decoy”, “mz”, “charge”, “protein”, and “precursor”. Optionally: “mobility”.
- *MaxQuant:* Please use the **evidence.txt** file. py_diAID uses the columns “Reverse”, “m/z”, “Charge”, “Proteins”, and “Modified sequence”. Optionally: “1/K0” and “1/K0 length”.
- *MSFragger:* The dataset should be analyzed while spectral library generation is active. Upload the **library.tsv** file. py_diAID will use the columns “PrecursorMz”, “PrecursorIonMobility”, “PrecursorCharge”, “ProteinId”, and “ModifiedPeptideSequence”. Optionally: “PrecursorIonMobility”.
- *Spectronaut single-run:* Please export the dataset in the **normal long** format already pre-filtered with “Quantification Data Filtering” and “No Decoy” as .tsv or .csv file. py_diAID requires the columns “'FG.PrecMzCalibrated”, “FG.Charge”, “PG.ProteinGroups”, and “EG.PrecursorId”. Optionally: “FG.ApexIonMobility” and “FG.IonMobilityPeakWidth”.

- *Spectronaut library:* Please export the library as an **.xls file**. py_diAID requires the columns “PrecursorMz”, “PrecursorCharge”, “UniProtIds”, and “ModifiedPeptide”. Optionally: “IonMobility”.
- *DIANN single-run:* py_diAID requires the columns “precursor_mz”, “charge”, “proteins”, “sequence”, and “fdr”. Optionally: “mobility”, “mod_sites”, and “mod”.
- *DIA-NN library:* py_diAID uses the columns 'PrecursorMz', 'PrecursorCharge', 'ProteinName', 'ModifiedPeptide', 'decoy', and 'QValue'. Optionally: “IonMobility”.
- *AlphaPeptDeep:* Please use the library as a **.tsv file**. py_diAID uses the columns 'PrecursorMz', 'IonMobility', 'PrecursorCharge', 'ProteinID' and 'ModifiedPeptide'. Optionally: “IonMobility”.
- *AlphaDIA:* Upload the **precursors.tsv** file. py_diAID will use the columns “mz_calibrated”, “charge”, “proteins”, “precursor_idx”, “decoy”, and “qval”. Optionally: “mobility_calibrated” and “base_width_mobility”.

Specify method parameters

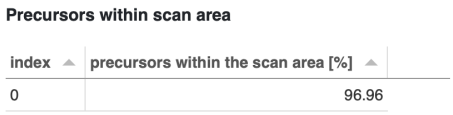


The following parameters define the optimal dia-PASEF method:

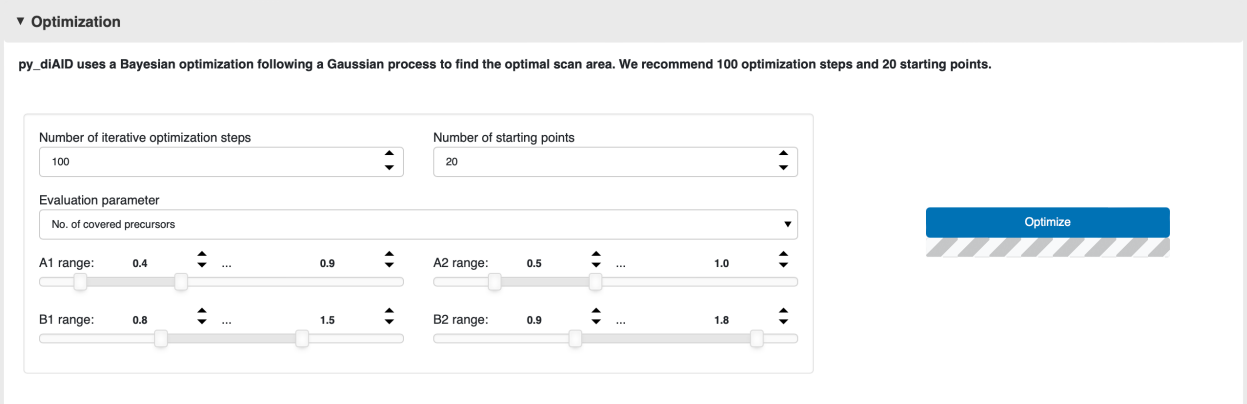
- ***m/z*-range:** Variable isolation windows allow for the coverage of a broad m/z-range while maintaining a short cycle time. We found that methods with a high theoretical precursor coverage also result in the acquisition of more precursors in single runs. Therefore, we recommend using wide m/z-ranges when generating dia-PASEF methods.
- **Ion mobility range:** We recommend a reduced ion mobility range. For proteomics experiments, we use 0.7-1.3 $1/K_0$.
- **Number of dia-PASEF scans:** The number of dia-PASEF scans influences the cycle time. Since the ramp and accumulation time is set to 100 ms by default, you can calculate the cycle time as follows: (MS1 scan + fragment scans) x (ramp time + transfer time) = (1+4) x (100 ms + 6.6 ms) = 533 ms. We recommend 8 dia-PASEF scans for 11 min Evosep gradients, 12 dia-PASEF scans for 21 min Evosep gradients, and 20 dia-PASEF scans for 44 min Evosep gradients (see “Rapid and in-depth coverage of the (phospho-)proteome with deep libraries and optimal window design for dia-PASEF” Experimental procedures).
- **Number of ion mobility windows / dia-PASEF scans:** One dia-PASEF scan can be separated into multiple ion mobility windows. The number of ion mobility windows defines how often the quadrupole changes its position. For two ion mobility windows per dia-PASEF scan, the quadrupole occupies two isolation positions during a single dia-PASEF scan. We recommend two ion mobility windows per dia-PASEF scan since this results in the highest precursor coverage based on simulations and in higher peptide identifications (see “Rapid and in-depth coverage of the (phospho-)proteome with deep libraries and optimal window design for dia-PASEF” Suppl. Figure S5).

- **Isolation window overlap:** This value defines the overlap in Da. We recommend no isolation window overlap for short cycle times.
- **Shift of the final acquisition scheme (in IM dimension):** The dia-PASEF acquisition scheme will be optimized for a maximized theoretical precursor coverage. However, in practice, the quadrupole has a blind spot and does not record signal while changing to the isolation position. This blind spot results in missed precursors in the top ion mobility window region. Therefore, we recommend to shift the final acquisition scheme upwards to not miss signal in the densest doubly charged precursor cloud.

Press "calculate" to calculate the precursors, which fall within the selected scan area defined by the *m/z*- and ion mobility ranges.



Optimization



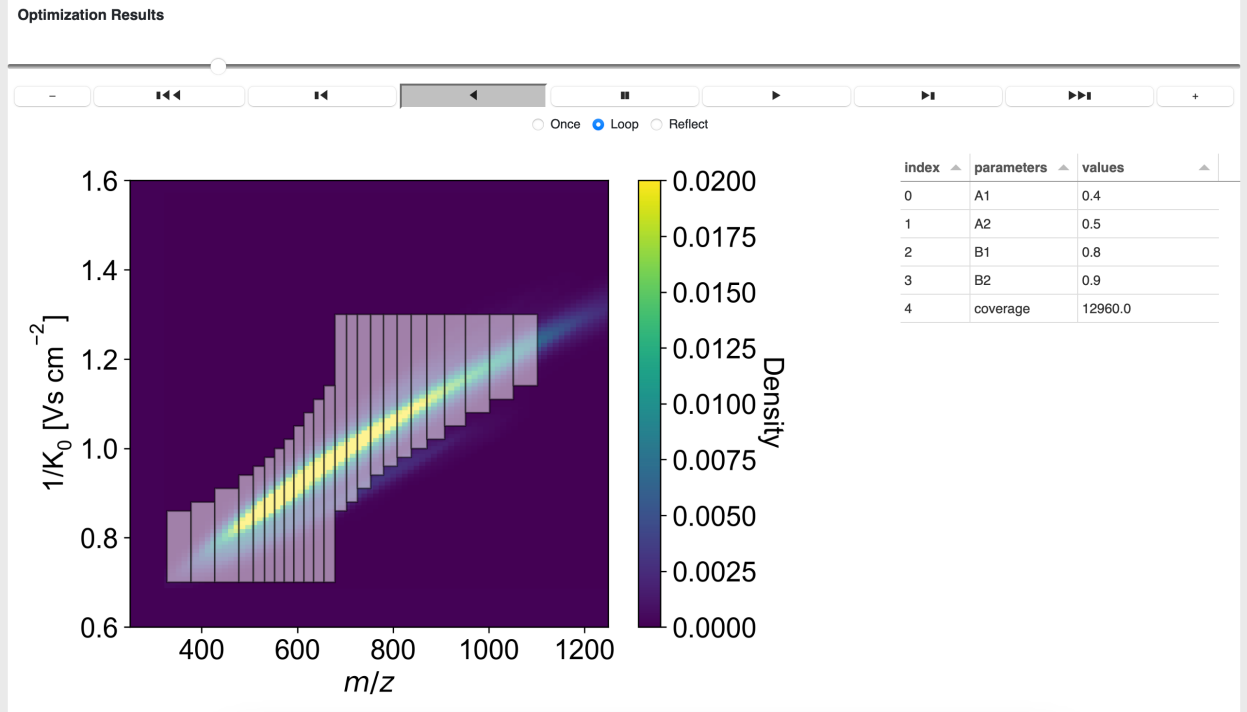
py_diAID calculates the isolation window width based on the precursor distribution in *m/z.* The position of the isolation windows in the *m/z* and ion mobility plane is dependent on a trapezoid that reflects the scan area. py_diAID selects a set of random A1, A2, B1, and B2 coordinates that define the corners of the trapezoid. A1, A2, B1, and B2 coordinates must lie within the respective ranges. Next, it generates a dia-PASEF method based on the placement of the trapezoid and the specified method parameters. This step is followed by the evaluation of the developed dia-PASEF method. Depending on the evaluation result, py_diAID decides which trapezoid coordinates should be attempted next, to find the optimal precursor coverage. The decision is based on a Bayesian optimization following a Gaussian process offered by the skopt package. A more detailed description of the py_diAID algorithm can be found in our publication: "Rapid and in-depth coverage of the (phospho-)proteome with deep libraries and optimal window design for dia-PASEF" Figure 2 and Suppl. Figure S6.

- **Number of iterative optimization steps:** This value defines the number of iterations to find the optimal position of the dia-PASEF acquisition scheme.
- **Number of starting points**: The algorithm initially tries out multiple random sets of coordinates until it considers the evaluation results to decide which parameters are reasonable to try next. **"**Number of starting points" defines the number of this initial random set. We recommend 100 iterative optimization steps and 20 starting points for a result that provides 99.2% reproducibility (see "Rapid

and in-depth coverage of the (phospho-)proteome with deep libraries and optimal window design for dia-PASEF" Suppl. Figure S7).

- **Evaluation parameter:** py_diAID can use different factors to evaluate the coverage of the dia-PASEF methods. "Number of covered precursors" includes all precursors and is recommended.
- **A1/A2/B1/B2 range:** The coordinates A1, A2, B1, B2 of the trapezoid are selected from within these ranges. (Please find a more detailed description in Figure 2 and supplementary Figure S6 in our publication.)

Press the "Optimize" button to start the optimization process. py_diAID needs approximately 10 minutes for 100 iterative optimization steps. You can supervise all optimization steps after the optimization process is finished using the built-in player.



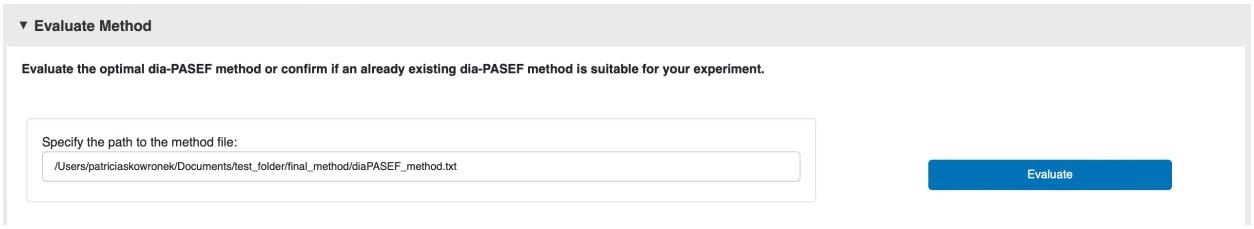
Create method

Py_dIAID automatically writes the scan area coordinates generated during the optimization step at "Scan area A1/A2/B1/B2". Alternatively, if you have coordinates from a previous optimization process, you can bypass optimization and use these existing coordinates.
Press the "Create" button to generate the final dia-PASEF method based on the specified trapezoid coordinates. py_diAID shifts this method upwards in the ion mobility dimension by the value specified in "Shift of the final acquisition scheme".

Created Method Parameters

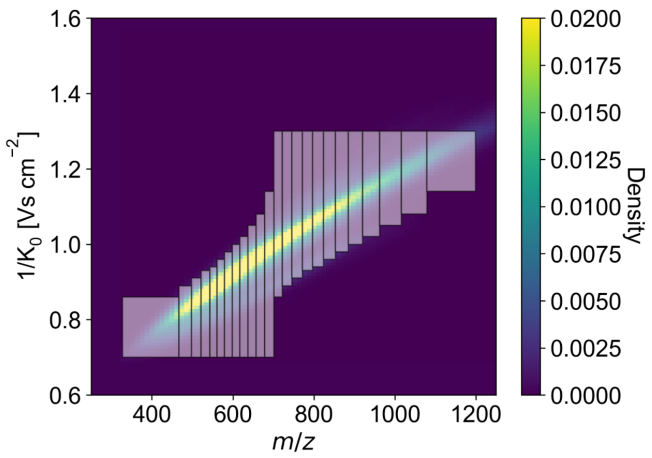
index	MS Type	Cycle Id	Start IM	End IM	Start Mass	End Mass	CE
0	MS1	0	-	-	-	-	-
1	PASEF	1	0.86	1.3	700.83	722.4	-
2	PASEF	1	0.7	0.86	327.52	466.77	-
3	PASEF	2	0.89	1.3	722.4	745.87	-
4	PASEF	2	0.7	0.89	466.77	498.27	-
5	PASEF	3	0.91	1.3	745.87	771.91	-
6	PASEF	3	0.7	0.91	498.27	521.82	-
7	PASEF	4	0.93	1.3	771.91	796.9	-
8	PASEF	4	0.7	0.93	521.82	543.28	-
9	PASEF	5	0.94	1.3	796.9	823.9	-
10	PASEF	5	0.7	0.94	543.28	561.3	-

Evaluate method



py_diAID automatically writes the file path of the final acquisition scheme in the field labeled “Specify the path to the method file.” Alternatively, you can designate a custom path to evaluate whether an existing dia-PASEF method is suitable for your experiment, based on the previously generated library.
Press the “Evaluate” button to display the result of the evaluation

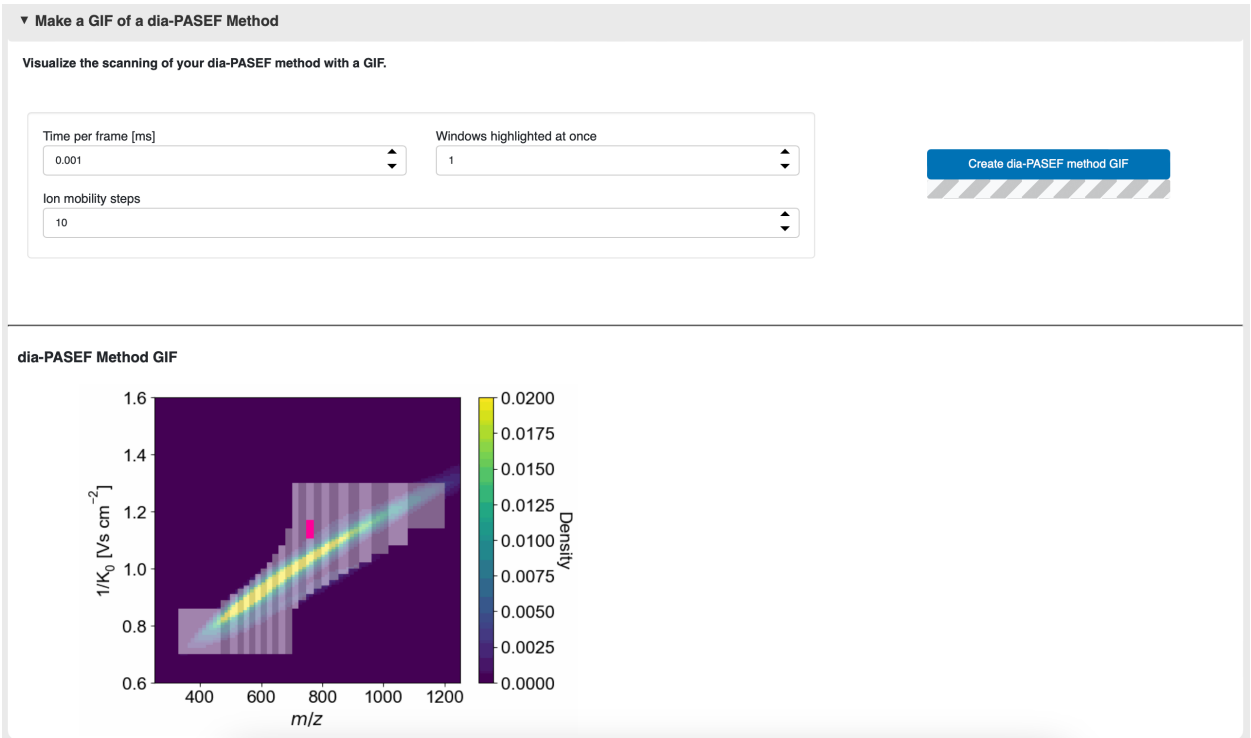
Final method plotted on top of precursor cloud



Evaluation Results

index	evaluation parameter	value
0	precursors within m/z-range [%]	96.96
1	unique proteins in the library	3420
2	unique precursors in the library	19998
3	smallest diaPASEF window	18.02
4	biggest diaPASEF window	139.25
5	average diaPASEF window size	36.34
6	No. of covered proteins	3397
7	No. of covered precursors	19165
8	all proteins covered	99.3%
9	all precursors covered	95.8%
10	No. of covered, doubly charged precursors	14911
11	all doubly charged precursors covered	96.0%
12	No. of covered, triply charged precursors	3845
13	all triply charged precursors covered	95.2%
14	No. of covered, quadruply charged precursors	233
15	all quadruply charged precursors covered	97.9%
16	No. of covered, singly charged precursors	171
17	all singly charged precursors covered	94.5%

Make a GIF of a dia-PASEF Method



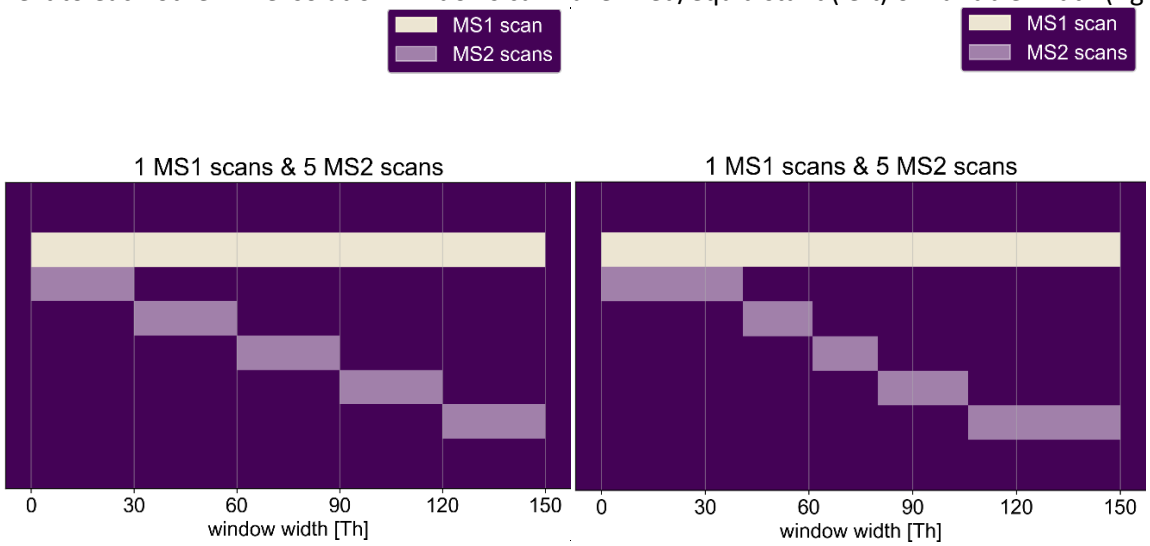
In this final card, you can create a GIF of your own method to visualize the TIMS elution aligned with the quadrupole movement through the m/z and ion mobility plane. The GIF will be saved in the previously specified output table under “gif”.

- **Time per frame [ms]:** This setting specifies the duration of each frame in the GIF. A detailed representation of the diagonal synchro scan movement requires a short frame time, such as 0.002 ms.
- **TOF trigger highlighted at once:** If you select many ion mobility steps, a single TOF trigger highlighted in pink might be visually difficult to follow. This setting allows you to highlight multiple TOF triggers at once in pink.
- **Ion mobility step:** This setting determines the resolution of the pink highlight illustrating a ion mobility step. Py_diAID will generate one snapshot per ion mobility step per scan. Therefore, GIFs with numerous ion mobility steps will take longer to generate and will result in longer, but more detailed GIFs. The maximum resolution is at 200 ion mobility steps.

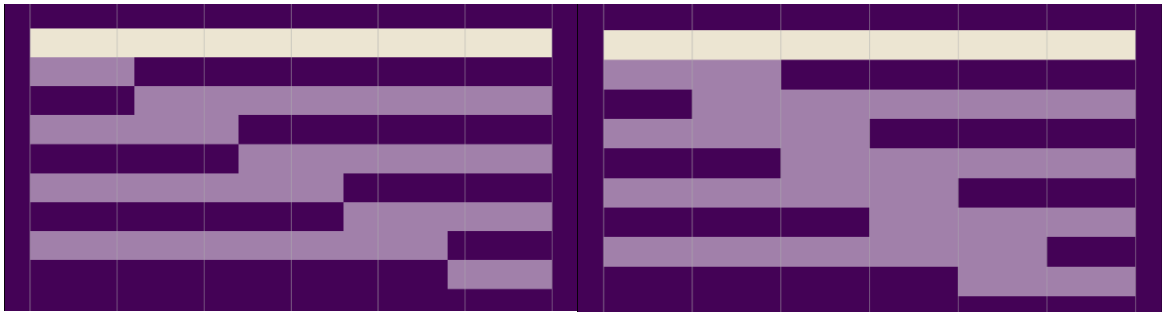
Synchro-PASEF methods

Py_diAID enables the generation of three synchro-PASEF acquisition schemes. Please note that only methods with equidistant isolation windows are currently supported by timsControl 6.0. Py_diAID allows the generation of a wider variety of method schemes, which we believe will be relevant for future developments in this scan mode - hence py_diAID already supports these. If your generated acquisition scheme is not compatible with timsControl 6.0, py_diAID will issue a warning.

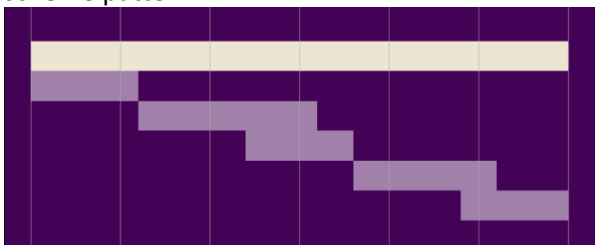
- **Classical synchro-PASEF** refers to an acquisition scheme where the isolation windows are positioned next to each other. The isolation windows can have fixed/equidistant (left) or variable width (right).



- **Highly accurate synchro-PASEF**, on the other hand, describes a method where two scans are always paired together. The first scan covers the left half of the scan range, and the second scan covers the right half. The division point where the scans are separated shifts to the right. This scheme is akin to the second slice-PASEF scheme developed by the groups of Markus Ralser and Vadim Demichev.



- Finally, the "**individual synchro-PASEF**" option allows complete freedom in defining the acquisition scheme pattern.



Load a library

To load a peptide library, follow the instructions for optimal dia-PASEF methods on page 6. A test library can be found at <py_diAID installation directory>\pydiaid\synchropasef\static \evidence_MaxQuant_270223.txt.

Define a scan area

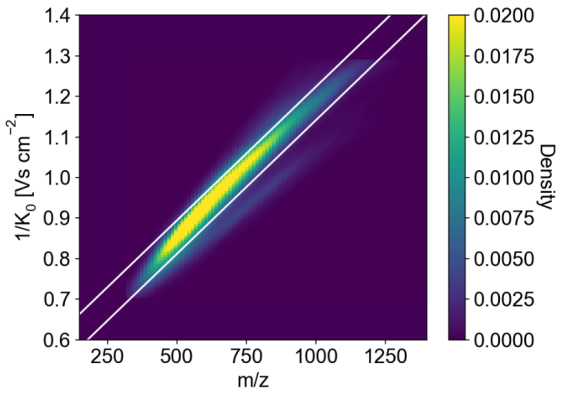
This card allows you to define the scan area that the synchro-PASEF method should cover.

- **Scan area width (horizontal) [m/z]:** The scan area is a parallelogram, the width of which can be specified in m/z terms. If the isolation windows have a fixed/equidistant width, this scan area width also determines the width of the isolation window. For instance, having a scan area width of 125 Th and 4 synchro scans results in 25 Th for each isolation window.
- **Charge state:** The peptide library is filtered based on the specified charge states. Subsequently, the scan area is optimally positioned by considering only the filtered peptide population.

To determine the scan area, py_diAID begins by filtering the peptide library for the specified charge states. It then calculates a linear regression based on the precursor population and subsequently moves the scan area in small steps both left and right. The purpose of this step-by-step movement is to find the ideal positioning so that the maximum number of precursors fall within the scan area.
By pressing “Calculate scan area”, you can initiate this process. Once the process is complete, the final scan area coordinates will be displayed in the “Scan area” field.

To visualize the scan area and evaluate its coverage, simply click on “Plot and evaluate scan area”. This will generate a graph with the scan area superimposed over the precursor distribution, along with a percentage representing the number of covered precursors.

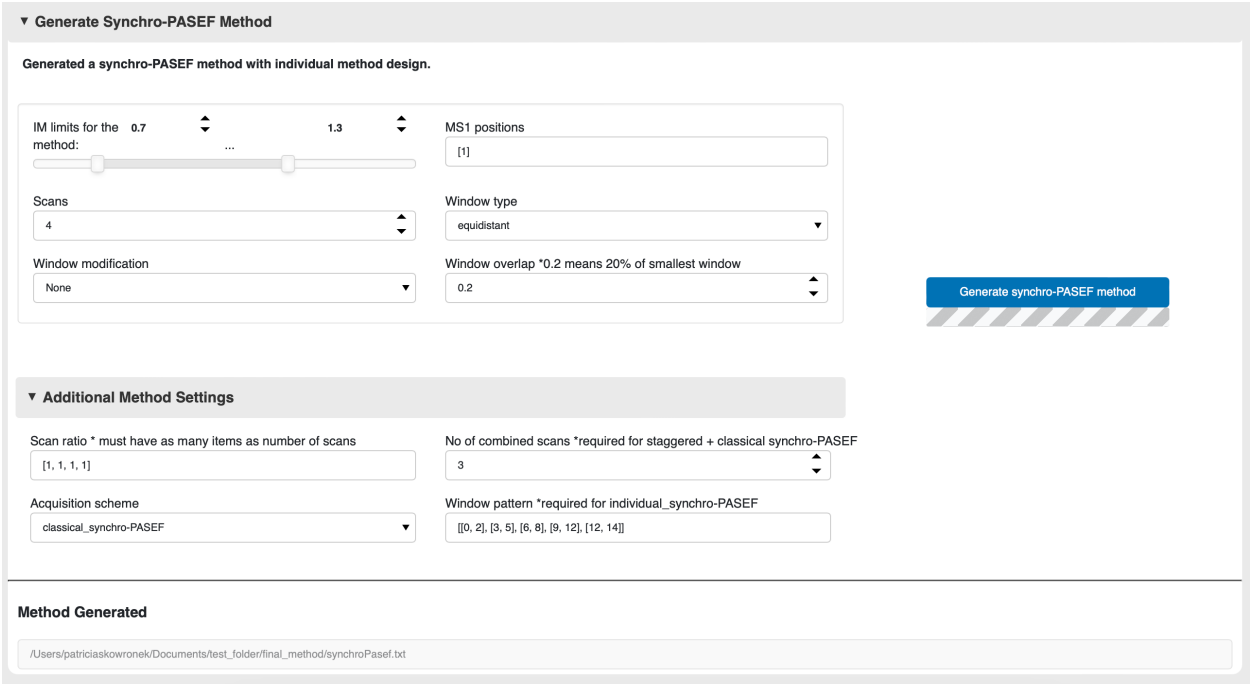
Scan area of method



Percentage of covered precursors

index	metrix for coverage	coverage
0	center of IM peak	0.741841

Generate a synchro-PASEF method



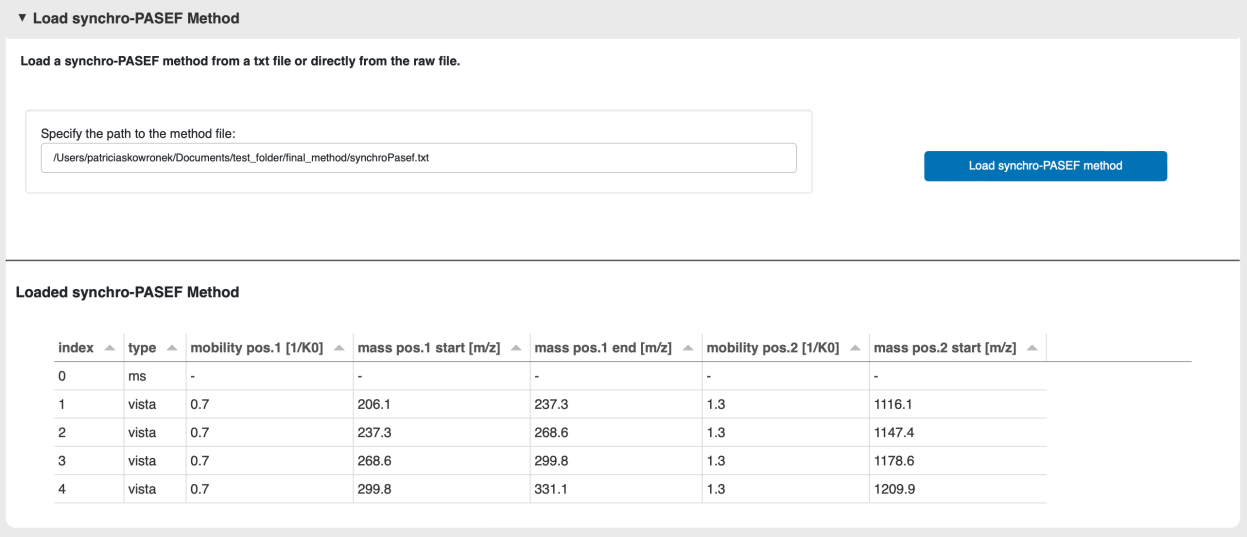
In this card, you can define the method parameters for the synchro-PASEF method. Only methods with equidistant isolation window widths can be calibrated using timsControl 6.0. If your generated acquisition scheme is not compatible with timsControl 6.0, py_diAID will issue a warning.

- **Ion Mobility (IM) limits for the method:** This range defines the ion mobility covered by the synchro-PASEF method. For proteomics experiments, we typically use 0.7-1.3 1/K0, as most precursors are present within this range.
- **MS1 positions:** Here, you can specify the positions at which py_diAID places MS1 scans. For instance, [1, 4] would result in a synchro-PASEF method with an MS1 scan as the first scan, followed by three MS2 scans before the next MS1 scan is scheduled.

- **Scans:** This setting defines the number of synchro scans (= isolation windows). We recommend a low scan count to create synchro-PASEF methods with shorter cycle times to maximize the efficiency of this scan mode.
- **Window type:** Py_diAID can generate isolation windows with fixed/equidistant width, or with variable width where each synchro scan covers approximately the same number of precursors, or a pre-defined width. If you select the pre-defined setting, specify the desired window ratios in 'additional method settings" and "scan ratio". If not, py_diAID will use the last used scan ratio as a default. Please note, the number of scans and items in the scan ratio list must match to use the pre-defined setting.
- **Window modification:** This setting allows you to choose between no window modification, window overlap, or a staggered isolation window scheme. For "overlap", define a "window overlap", or py_diAID will use a default of 0.2 (20% window overlap). If you choose "staggered," py_diAID will combine adjacent isolation windows into one. For the "classical synchro-PASEF" acquisition scheme, it combines three scans by default. If you want a different number of combined scans, specify it under "Additional method settings" and "No. of combined scans." For the "highly accurate synchro-PASEF" method, py_diAID will always have an overlap of one scan/bin length which equals a 50% window overlap. For an individual Synchro-PASEF method, this setting will be ignored.
- **Window overlap:** This defines the extent of overlap between isolation windows. "Overlap" must be defined under "Window modification" for this setting to be necessary.
- **Scan ratio:** This parameter sets the ratio of scan widths. For example, using [1, 5, 1] results in a middle isolation window that is five times wider. The "pre-defined" option must be selected under "Window type" for this setting to be applicable.
- **Acquisition scheme:** Py_diAID offers three types of acquisition schemes: classical synchro-PASEF, highly accurate synchro-PASEF, and the customizable individual synchro-PASEF. For highly accurate synchro-PASEF, the step width is defined by the "scans" and "window type" settings. For the "individual synchro-PASEF" option the acquisition scheme is designed according to the window distributions defined under "window pattern".
- **Window pattern:** This setting is only in use for the individual synchro-PASEF acquisition scheme. Here, the scan lengths are defined by ranges. For instance, [0, 2] indicates that the first three scans/bins are combined into a single isolation window, starting at the m/z position 0. Py_diAID automatically updates the 'scans' and 'scan_ratio' fields.

Press "Generate synchro-PASEF method" to create the synchroPasef.txt file. You will find it in the "final_method" folder as defined under "Load library", "Save the output at the following folder". This file also serves as input for timsControl.

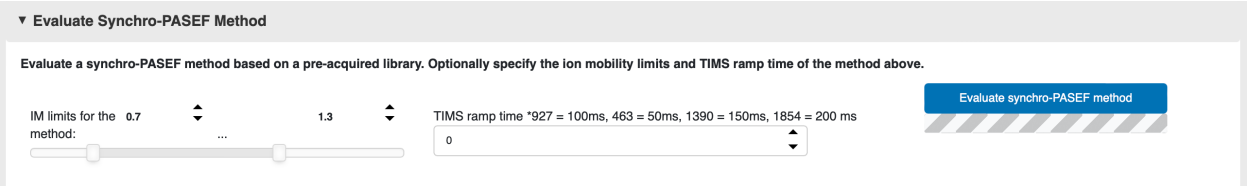
Load a synchro-PASEF method



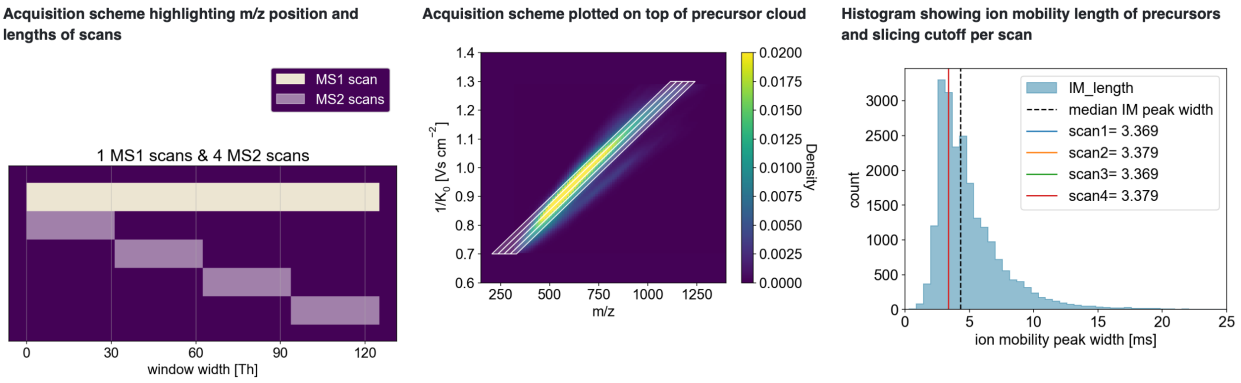
index	type	mobility pos.1 [1/K0]	mass pos.1 start [m/z]	mass pos.1 end [m/z]	mobility pos.2 [1/K0]	mass pos.2 start [m/z]
0	ms	-	-	-	-	-
1	vista	0.7	206.1	237.3	1.3	1116.1
2	vista	0.7	237.3	268.6	1.3	1147.4
3	vista	0.7	268.6	299.8	1.3	1178.6
4	vista	0.7	299.8	331.1	1.3	1209.9

In this card, you have the option to load the newly generated synchro-PASEF method and review its parameters. The current method path is automatically updated in the “Specify the path to the method file” field. Alternatively, you can also load a pre-existing method. Press “Load synchro-PASEF method” to display the method parameters in a table format.

Evaluate a synchro-PASEF method



In this card, you have the option to evaluate either the newly generated synchro-PASEF method or any pre-existing methods. This is only possible if you have already loaded a library in the “load library” card and a method in the “load synchro-PASEF method” card. If your peptide library includes information about the precursors’ ion mobility length, you can further define two settings: the ion mobility range of the method, which defaults to the one defined while generating the method, and the TIMS ramp time for raw data acquisition. Based on these values, py_diAID can calculate and display the synchro scan width in milliseconds along the ion mobility dimension or in other words the slicing cutoff per scan.
Press “Evaluate synchro-PASEF method" for py_diAID to generate up to three method evaluation plots and characteristics for benchmarking the method’s suitability.
Plots to benchmark synchro-PASEF methods:



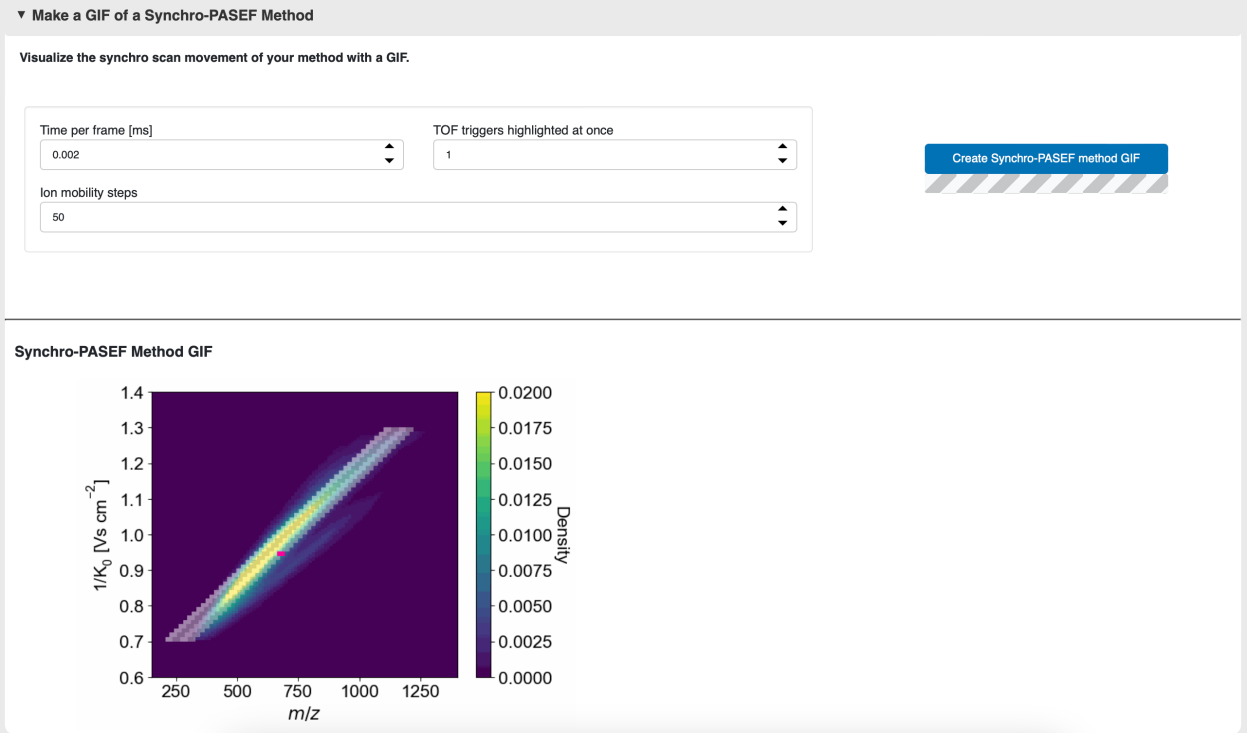
The left plot shows the widths of the isolation windows in the acquisition scheme and their order, regardless of their placement in the m/z and ion mobility plane. This plot aids in better understanding the placement of overlapping isolation windows.
The middle plot overlays the isolation window on top of the kernel density distribution of precursors, helping to estimate precursor coverage.
The right histogram displays the ion mobility peak width of peptides in the library, overlaid with the slicing cutoff of each synchro scan. Partitioning the fragment signal into adjacent synchro scans along the ion mobility peak enables linking fragment signal to their precursor signal, resulting in high specificity (refer to our publication for further details). Precursors with an ion mobility peak width wider than the slicing cutoff of a synchro scan will be partitioned, and those with a smaller width will only be sliced if they are not positioned in the center of the synchro scan. We suggest using methods where the median ion mobility peak width is wider than the slicing cutoff for the majority of isolation windows. The last plot is only generated if the library contains ion mobility peak width information and if the TIMS ramp time exceeds zero.

index	all synchro scans	#synchro scan 1	#synchro scan 2	#synchro scan 3	#synchro scan 4
unique proteins in the library	3874	3874	3874	3874	3874
unique precursors in the library	21878	21878	21878	21878	21878
No. of covered proteins	3615	1824	2368	2111	1422
No. of covered precursors	16231	3521	5790	4497	2421
No. of covered, singly charged precursors	1	0	1	0	0
No. of covered, doubly charged precursors	12355	3145	4969	3247	993
No. of covered, triply charged precursors	3323	317	693	1078	1235
No. of covered, quadruply charged precursors	552	59	127	172	193
proteins covered [%]	93.3	47.1	61.1	54.5	36.7
precursors covered [%] (counting IM peak at center)	74.2	16.1	26.5	20.6	11.1
singly charged precursors covered [%]	50	0	50	0	0
doubly charged precursors covered [%]	84.6	21.5	34	22.2	6.8
triply charged precursors covered [%]	51.2	4.9	10.7	16.6	19
quadruply charged precursors covered [%]	69.6	7.4	16	21.7	24.3
covered and sliced precursors [%] (counting complete IM peak width)	95.99	98.58	98.33	97.92	97.15
average fragment coverage [%] (counting complete IM peak width)	86.96000000000001	47.29	50.91	46.53	42.88
sliced precursors [%] (counting complete IM peak width)	80.03	39.9	53.88	47.64	29.45
covered precursors [%] (counting complete IM peak width)	83.37	40.48	54.79	48.65	30.31

Moreover, py_diAID calculates the total precursor coverage as well as coverage per charge state, for all synchro scans and individual synchro scans. If the peptide library contains ion mobility peak width

information, the displayed table will also show slicing characteristics such as the percentage of covered and sliced precursors, and the average percentage of covered fragments.
Together, these plots and the table should assist you in evaluating the suitability of the generated method. We recommend using methods with high precursor coverage and slicing percentage. All this information will be saved in the earlier specified output table at “final_method”.

Make a GIF for a synchro-PASEF method



In this final card, you can create a GIF of your own method to visualize the diagonal and ultra-fast movement of the quadrupole through the m/z and ion mobility plane. The GIF will be saved in the previously specified output table under “gif”.

- **Time per frame [ms]:** This setting specifies the duration of each frame in the GIF. A detailed representation of the diagonal synchro scan movement requires a short frame time, such as 0.002 ms.
- **TOF trigger highlighted at once:** If you select many ion mobility steps, a single TOF trigger highlighted in pink might be visually difficult to follow. This setting allows you to highlight multiple TOF triggers at once in pink.
- **Ion mobility step:** This setting determines the resolution of the synchro scan movement. Py_diAID will generate one snapshot per ion mobility step per scan. Therefore, GIFs with numerous ion mobility steps will take longer to generate and will result in longer, but more detailed GIFs. The maximum resolution is at 200 ion mobility steps.

Orbitrap Astral DIA methods

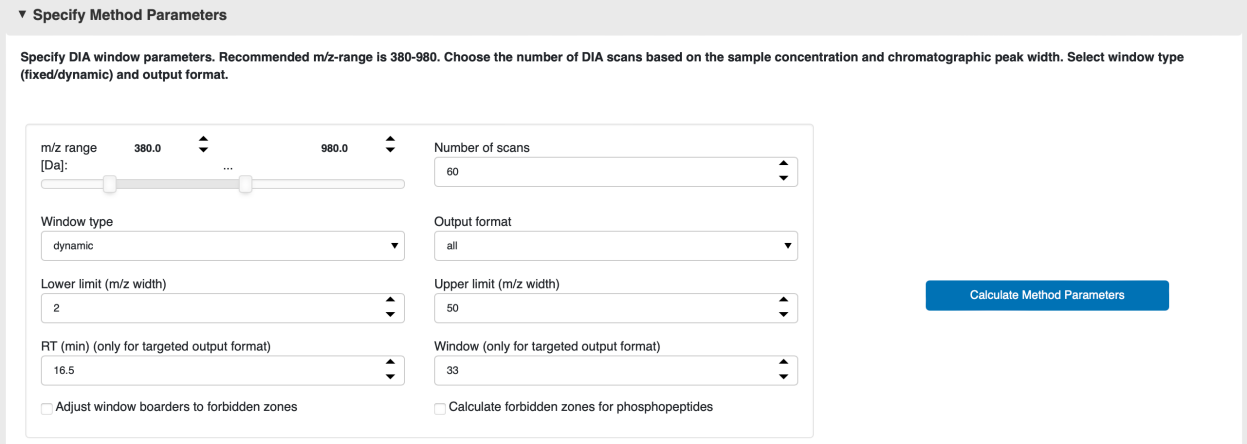
Optimal FAIMS CV prediction

We will soon publish a Jupyter notebook demonstrating how to generate a library of peptides with predicted properties, including their optimal FAIMS compensation voltage (CV), at https://github.com/MannLabs/pydiaid/tree/main/nbs.

Load Library

To load a peptide library, follow the instructions for optimal dia-PASEF methods on page 6. A test library can be found at <py_diAID installation directory>\pydiaid\oadia\static\AlphaPept_results.csv. To load a predicted library from the step above use 'AlphaPeptDeep library' as setting for 'Analysis software'.

Specify Method Parameters

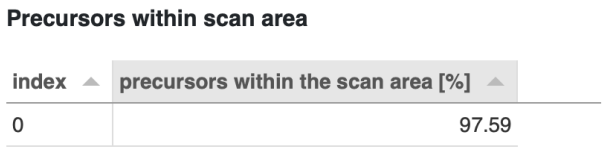


In this card, you can define the method parameters for an Orbitrap Astral DIA method.

- **m/z range:** This range defines the total m/z range for DIA precursor selection. For most proteomics experiments, we typically use 380-980 as most precursors are covered by this range.
- **Number of Scans:** This setting defines the number of DIA scans (=isolation windows). We recommend optimizing the number of scans to obtain good quantification (define desired number of data points per peak and calculate theoretical cycle time to achieve it).
- **Window type:** Choose either a fixed or dynamic window width.
- **Output format:** py_diAID supports three different window output tables, which differ in the columns that define the methods: 'targeted', 'center mass' and 'm/z ranges'. The first is to be used in the targeted MS mode, while the latter two can be used to set the windows in the DIA mode. Selection of the output format depends on instrumentation, Tune software version and personal preference. If Orbitrap instrumentation and software version supports custom windows in the DIA mode, the output formats 'center mass' or 'm/z ranges' is preferable. For loading the respective methods, please refer to the chapter 'Load an Orbitrap Astral DIA method at Thermo Method Editor'.
- **Lower limit (m/z width):** This setting defines the minimum width of DIA isolation windows. For Astral MS2 scans, we recommend setting this to 2 Th.

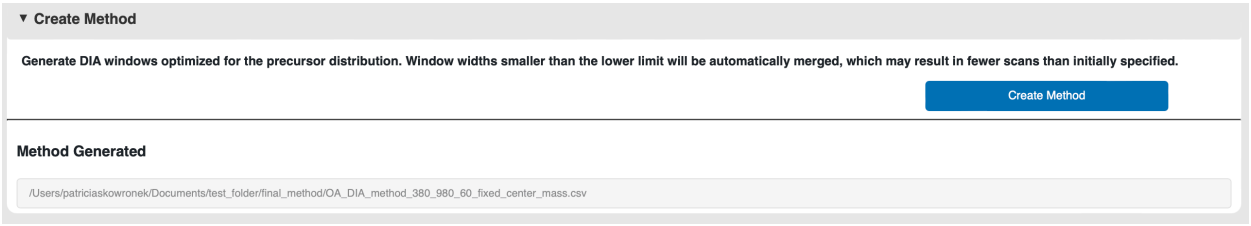
- **Upper limit (m/z width):** This setting defines the maximum width of DIA isolation windows. Optimal limit values might differ based on sample type and application. For standard proteomics applications, we recommend 50 Th.
- **RT (min) (only for targeted output format):** Defines the intended gradient length. This will generate the “RT (min)” column necessary for the targeted output format.
- **Window (only for targeted output format):** Defines the intended gradient length. This will generate the “Window” column necessary for the targeted output format.

Press “calculate method parapeters” to calculate the precursors, which fall within the selected scan area defined by the *m/z*-range.



Create Method

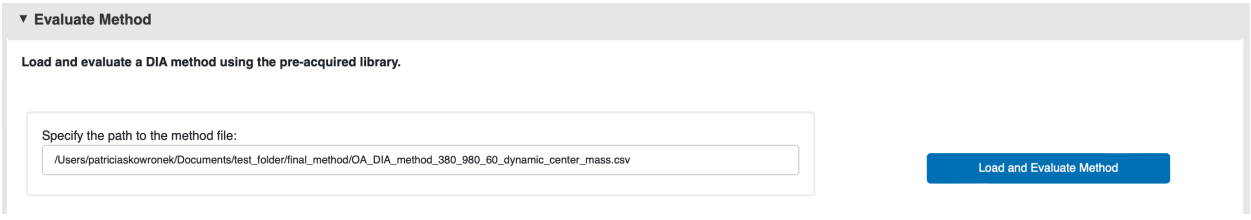
In this card, you can create the Orbitrap Astral DIA method.



Press “Create method” to generate the Method.csv file. You will find the file in the “final_method” folder as defined under “Load library”, “Specify the path to the library”. This file also serves as input for the Thermo Method Editor.

Evaluate Method

In this card, you have the option to evaluate either the newly generated synchro-PASEF method or any pre-existing methods. This is only possible if you have already loaded a library in the “load library” card. The path to the newly generated method will be automatically updated in the 'Specify the path to the method file' field whenever a new method is generated in the previous tab. Alternatively, you can indicate other methods.



Press 'Load and Evaluate Method' to generate three analysis panels:

- Left: Orbitrap Astral DIA Method table
- Center: Method statistics table displaying key metrics including number of scans, window dimensions, and items per scan.
- Right: Visualization of the precursor distribution across the m/z range, where color intensity indicates scan window width and the y-axis shows the number of precursors per window.

With 'Adjust window borders to forbidden zones' activated:

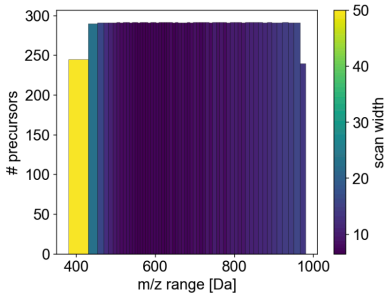
Method Table

index	Center Mass (m/z)	Window Width (m/z)
0	405.0	50.0
1	441.619605	23.23921
2	461.256968	16.035516
3	475.510916	12.472381
4	487.169365	10.844516
5	497.196856	9.210467
6	506.153093	8.702007
7	514.434318	7.860442
8	522.313505	7.897932
9	530.115308	7.705675
10	537.634853	7.333414
11	544.797917	6.992716
12	551.536022	6.483493

Method Statistics

index	Metric	Value
0	Number of scans	60
1	Min width (Da)	6.47
2	Max width (Da)	50.00
3	Average width (Da)	10.00
4	Min items per scan	240
5	Max items per scan	292
6	Average items per scan	289.95
7	Window type	dynamic

Precursor per Window



Without 'Adjust window borders to forbidden zones' activated:

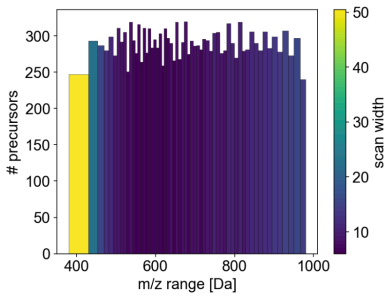
Method Table

index	Center Mass (m/z)	Window Width (m/z)
0	405.222771	50.445543
1	441.950772	23.010459
2	461.45964	16.007276
3	475.466006	12.005457
4	486.971236	11.005002
5	496.975783	9.004093
6	505.979876	9.004093
7	514.483742	8.003638
8	522.48738	8.003638
9	529.99079	7.003183
10	537.494201	8.003638
11	544.997611	7.003183
12	551.500567	6.002728

Method Statistics

index	Metric	Value
0	Number of scans	60
1	Min width (Da)	6.00
2	Max width (Da)	50.45
3	Average width (Da)	10.00
4	Min items per scan	240
5	Max items per scan	320
6	Average items per scan	289.37
7	Window type	dynamic

Precursor per Window

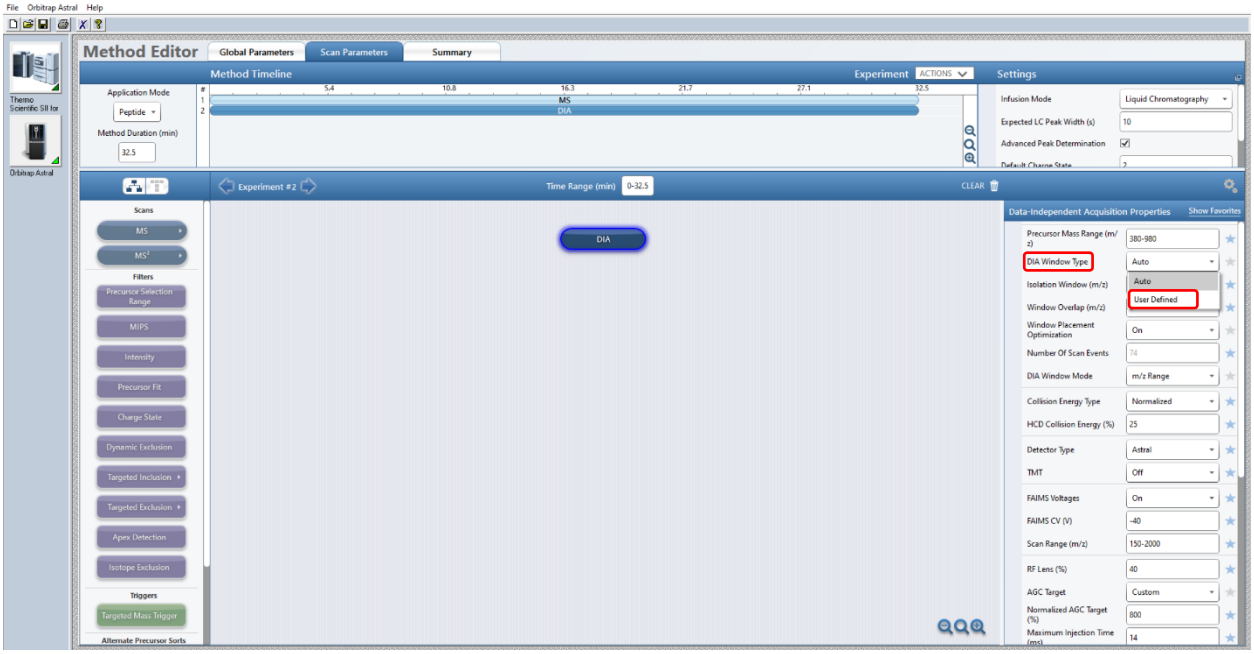


Load an Orbitrap Astral DIA method at Thermo Method Editor

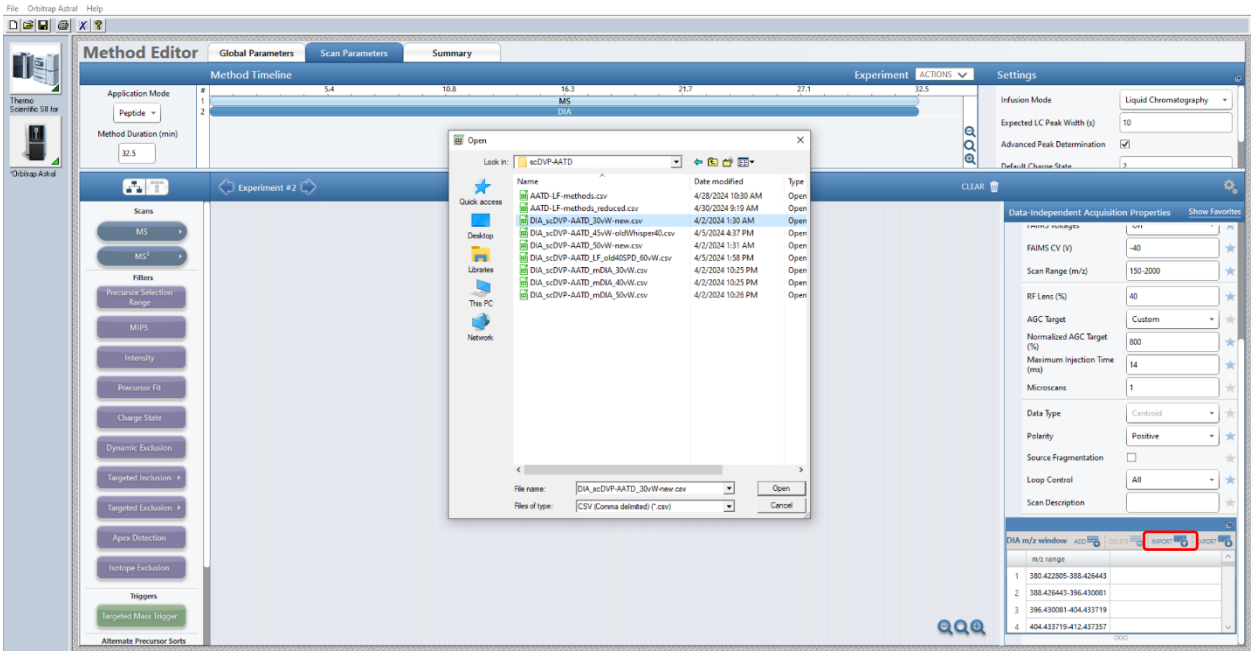
Please note that the Obitrap Astral DIA method generator only generates the optimal distribution of DIA windows, general MS1 and MS2 parameters need to be set separately.
Depending on the output format chosen in the previous step, instructions for loading the methods slightly differ.

DIA mode

Select the DIA mode, if you choose “Center Mass” or “m/z Range” output formats. Open a new method file, navigate to Scan parameter and set desired MS1 parameters. Add a DIA scan, change “DIA Window Type” from Auto to User Defined. Chose the “DIA Window Mode”, center mass or m/z range, based on which py_diAID method output you selected in the previous step.

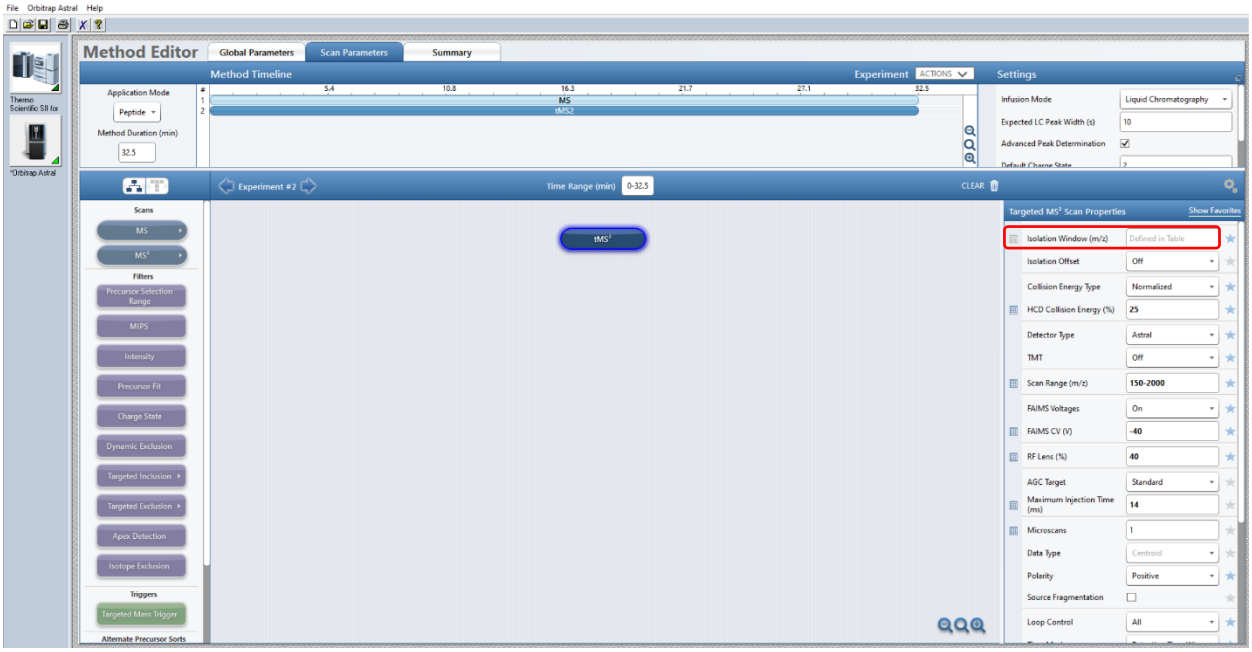


Set desired MS2 parameters, such as HCD collision energy, FAIMS Voltages, max injection time etc. Finally load the py_diAID generated window distribution. Navigate to the bottom of the DIA method parameters and click “Import” and load the py_diAID generated .csv file

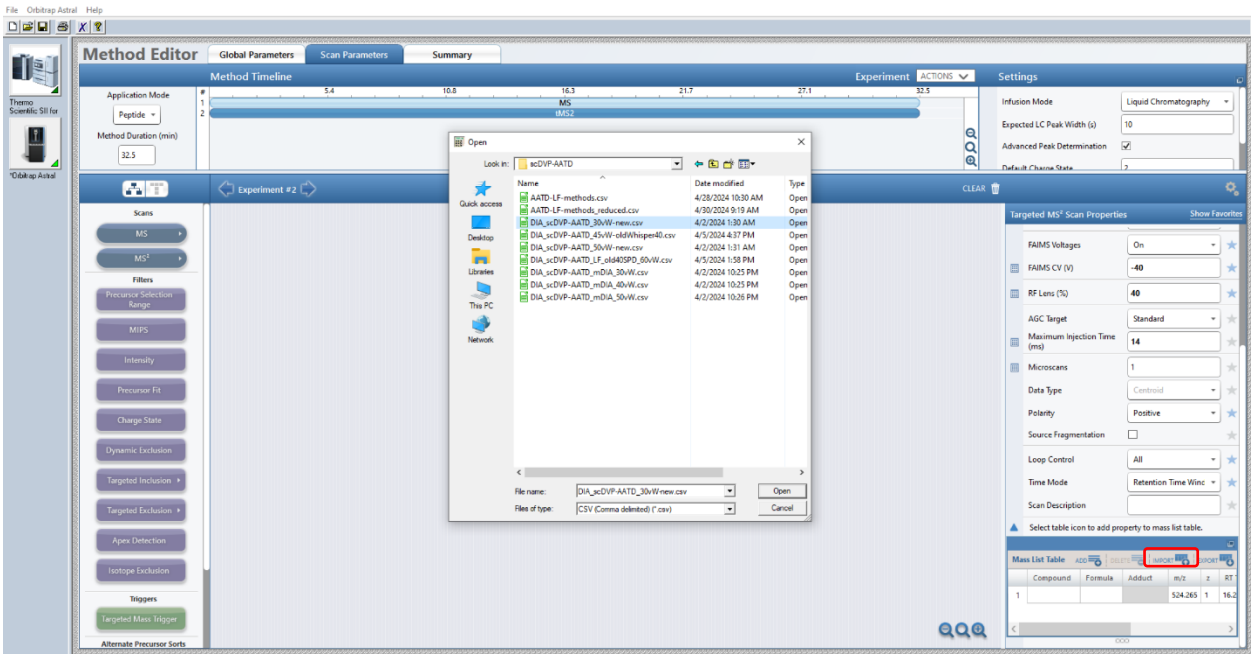


Targeted MS2 mode

Select the Targeted MS2 mode, if you choose the “Targeted” output format. Open a new method file, navigate to Scan parameter and set desired MS1 parameters. Add a MS2 scan and click the table symbol next to the MS2 parameter “Isolation Window (m/z)”. The field should gray out and read “Defined in Table” (see screenshot).



Set desired MS2 parameters, such as HCD collision energy, FAIMS Voltages, max injection time etc. Finally load the py_diAID generated window distribution. Navigate to the bottom of the DIA method parameters and click “Import” and load the py_diAID generated .csv file.



Imported method parameters from the “targeted” py_diAID output, are define as such: