Rapid Development of Improved Data-dependent Acquisition Strategies - Supplementary Materials.

Vinny Davies[†,§], Joe Wandy[‡,§], Stefan Weidt[‡], Justin J. J. van der Hooft[¶], Alice Miller[†], Ronan Daly[‡] and Simon Rogers[†,*]

[†] School of Computing Science, University of Glasgow, Glasgow, UK

[‡] Glasgow Polyomics, University of Glasgow, Glasgow, UK

[¶] Bioinformatics Group, Department of Plant Sciences, Wageningen University, 6780 PBWageningen, The Netherlands

[§] Contributed equally

[*] E-mail: Simon.Rogers@Glasgow.ac.uk

S1. Peak picking parameters

Parameters	Settings
Mass Detector	Centroid
Mass Detector Noise Level	1000.0
Chromatogram Builder	ADAP
Min Group Size	3
Group Intensity Threshold	1000.0
Min Highest Intensity	1000.0
m/z Tolerance (ppm)	10.0
Deconvolution Method	Wavelets (ADAP)
S/N Threshold	5.0
S/N Estimator	Intensity Window SN
Min feature height	1000.0
Coefficient / Area Threshold	1.0
Peak Duration Range	(0.0, 5.0)
RT Wavelet Range	(0.1, 1.0)

Table SI-1: MZMine2 settings used for the peak picking used for evaluation.

Parameters	Value
algorithm	CentWave
ppm	15
peakwidth	(15, 80)
snthresh	5
noise	1000
prefilter	(3, 500)

Table SI-2: XCMS parameters used in peak picking. All other parameters were XCMS defaults.

S2. Scan timing analysis and comparison with vendor method.

Timings were compared for different methods to understand the difference in the number of scans that each method achieves for a given sample on the real MS hardware. Timings for each scan were calculated by looking at the resulting mzML file and computing the difference in scan start times for successive scans. We adopted this approach as we were interested in the full duty cycle, including any processing in ViMMS. Figure SIAA shows these times plotted against the retention time (RT) of the start of the scan for 6 different methods. These methods included 3 methods that worked through the Thermo Fusion IAPI: a full scan controller in C# (csharp_full) bypassing VIMMS and directly using IAPI, a full scan controller in Python using ViMMS (python_full), and a Top-N controller in Python using ViMMS (python_topn). These were compared against 3 vendor methods created using the Xcalibur software suite: a full scan method (fusion_full), a Top-N method (fusion_topn) and a Top-N method with background exclusion (fusion_topn_be). All methods used the same scan parameters, as detailed in the main paper.

The full scan methods in Figure SIAA allow us to compare the comparative speed of the vendor method with those written in C# and Python as part of ViMMS and hence assess any timing overhead introduced by our use of the IAPI. The results show that there is a small difference (around 0.05) seconds between the time taken for a vendor scheduled scan andscans produced by ViMMS. The difference appears to be as a result of the delay in using the IAPI in order to schedule scans. Our implementation requires a bridge between the IAPI (C#) and python (ViMMS). The very small difference between csharp_full and python_full suggest that this additional translation is not costly.

Comparison of the TopN methods is more difficult. The vendor TopN method makes use of some level of parallelisation (Senko *et al.*, 2013). The MS2 scans scheduled from a particular MS1 are started **after** the next MS1 scan has been injected. This means that the time between successive scan start times doesn't always reflect that actual time a scan takes but, for most MS1 scans, just gives the injection time. This can be seen in the vendor scenario 1 in Figure SI-1.

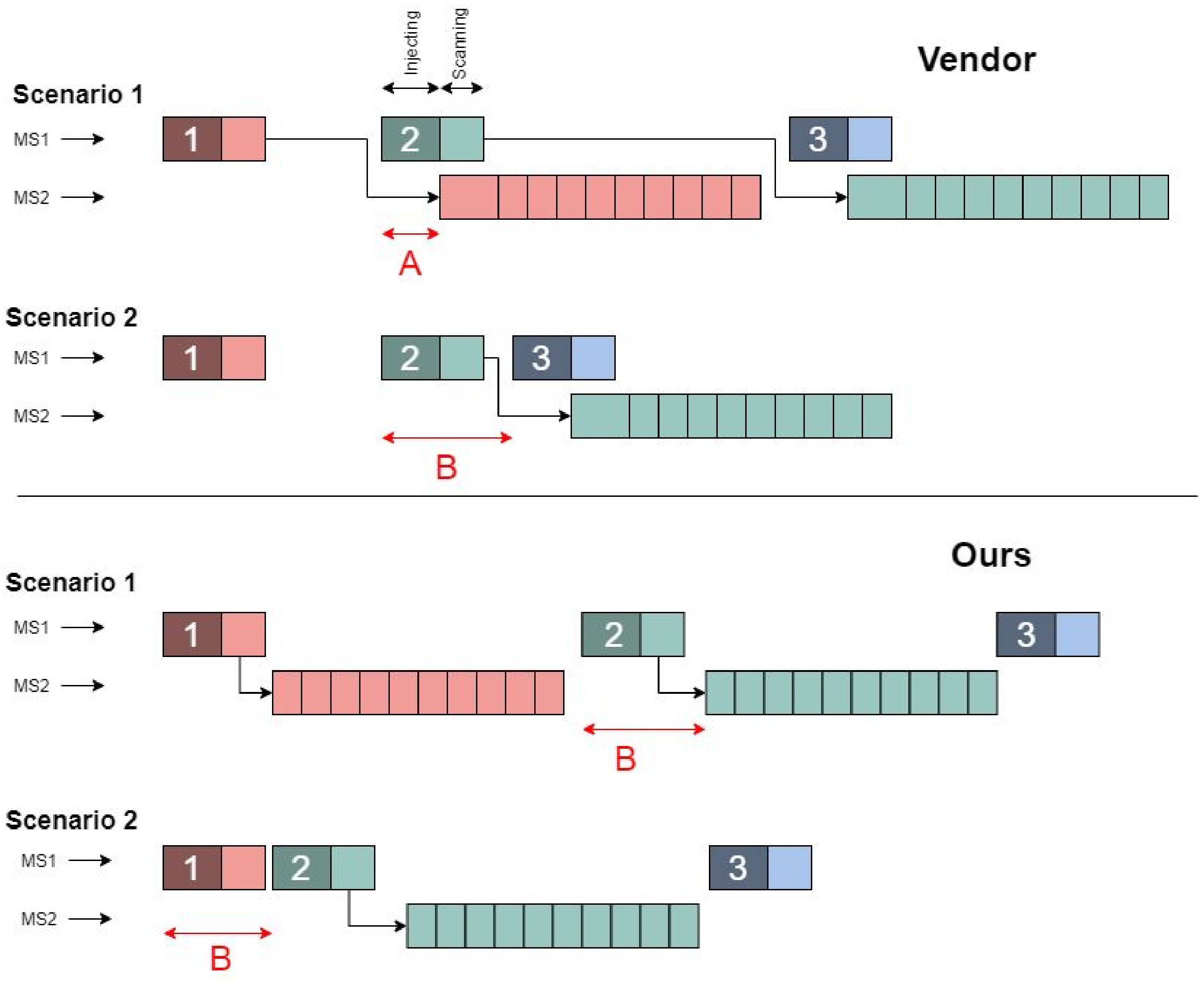

Figure SI-1: Schematic of scan ordering for vendor (top) and our methods (bottom). In both cases, two scenarios are shown, differing in whether any MS2 scans are scheduled from MS1 scan 1. In Scenario 1 scans are scheduled. In Scenario 2, they are not. Note that this scheme is slightly different from that shown in (Senko et al., 2013) where they have full parallelisation due to the MS2 scans being performed in the linear ion trap whereas we perform MS2 scans in the orbitrap. This is also why the first MS2 scans in the vendor setup take longer than subsequent ones: they can start injection after the MS1 injection, but scanning cannot start until the MS1 scanning is finished.

Scenario 1 in the vendor block of Figure SI-1 shows how the first MS2 scan scheduled from MS1 scan 1 starts after the injection of MS1 scan 2 (the time labeled A). Direct comparison of scan times is therefore not possible. However, we can compare the MS1 scan cycle time in situations where no

MS2 are scheduled from the previous MS1 scan. This is shown in vendor scenario 2. Here, no scans are scheduled from MS1 scan 1 meaning that the time from the start of MS1 scan 2 to the start of MS1 scan 3 does give us an indication of the MS1 cycle time. As we do not make use of parallelisation in our methods, the times are consistent, regardless of whether or not MS2 scans are scheduled.

Figure SI-2 shows a comparison of MS1 times for TopN methods where vendor times are limited to only the MS1 scan times for scans where no MS2 were scheduled from the previous MS1 scan (time B in vendor scenario 2 in Figure SI-1). We see roughly the same 0.05 second overheard for our methods (compare python_topn with fusion_topn_be). As we only plot vendor instances when no MS2 are scheduled, these are concentrated to the edges of the plot where the chromatogram is less busy. Fusion_topn_be is a machine TopN method with background exclusion turned on. Fusion_topn has no background exclusion and as such we see very few instances when it doesn't fragment and therefore very few points on this plot.

This analysis allows us to conclude that there is very little overhead associated with our use of the IAPI but there may be benefits in the future of implementing similar parallelisation to that provided in the vendor method in our controllers as the parallelisation allows the vendor method to produce many more scans in total. For example, where our TopN controller did 6786 scans in total for Beer (652 MS1 and 6134 MS2), the vendor controller (fusion_topn) with the same scan parameters managed 9901 scans (1299 MS1 and 8602 MS2).

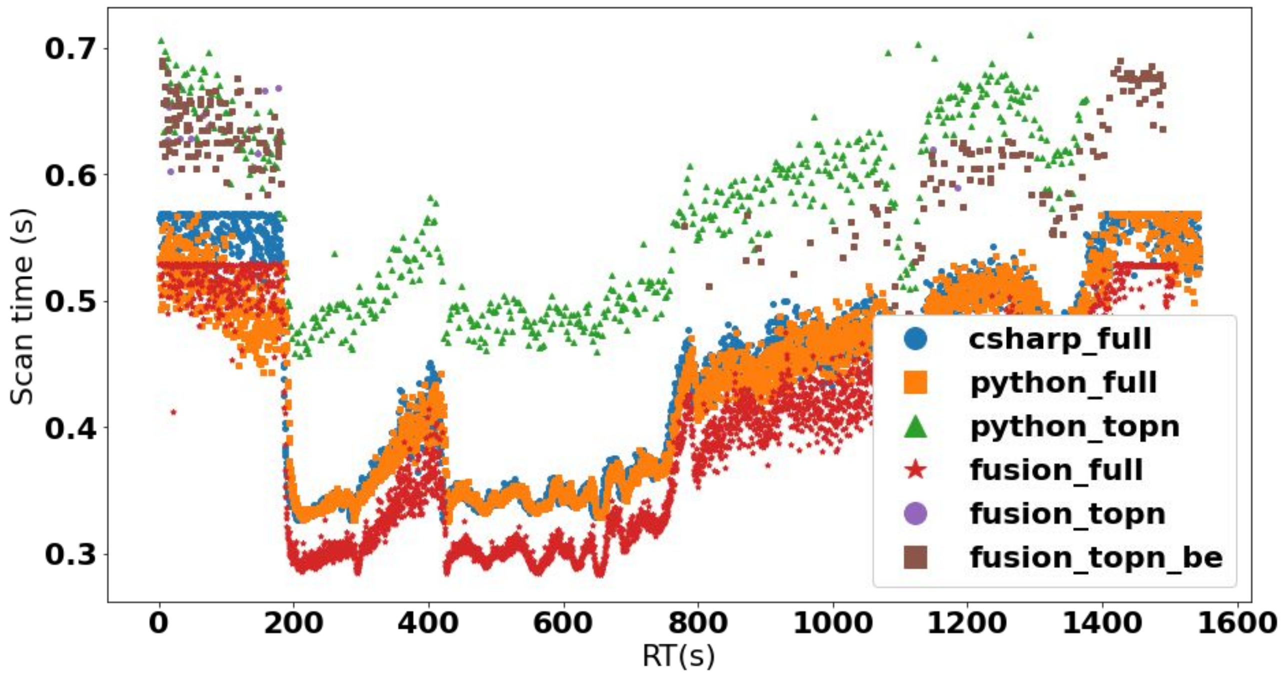

Figure SI-2*: Plot showing the timings for 6 different methods against the retention time (RT) of the scan. For Top-N methods only points that are above the threshold of 0.4 are plotted in order to enhance the visualisation of the results.*

It is interesting to note that this dramatic increase in number of scans does not translate to a large improvement in performance. Table SI-3 shows the coverage of our new controllers as well as the vendor TopN method. We can see that it doesn't outperform any of the proposed methods. It improves over our TopN for the Beer but not the Serum sample. This suggests that further development (including some level of parallelisation) could further improve our controllers. Indeed, the optimal results (shown in Table SI-3) using average timings from the vendor method offer an improvement over the optimal results using our TopN timings, particularly for Beer.

Method	Beer (4592 peaks)		Serum (3032 peaks)	
	Iter 1	Iter 2	Iter 1	Iter 2
TopN	1046		656	
WeightedDEW	**1859**	**1768**	**1105**	**1226**
SmartROI	1660	1546	991	1015
SmartROI (shift = 1)	1837	1740	1101	1193
SmartROI (shift = 2)	1838	1745	1040	1168
Vendor TopN method	*1359*		*511*	
Optimal (TopN scan timings)	*2955*		*1542*	
Optimal (vendor TopN scan timings)	*3229*		*1555*	

Table SI-3: method coverage including vendor methods.

Parellisation must be used with care however. Studying Figure S1, we can see that the parallelisation can lead to a very large time between the survey scan and the later MS2 scans scheduled from it. Consider, for example, the large time from MS1 scan 2 until its MS2 scans. Smaller peaks may have stopped eluting by the time they are needed for fragmentation. Indeed, in our analysis (and other analysis pipelines too), when looking for the precursor m/z for the MS2 scans scheduled from MS1 scan 2, we would actually look to the most recent MS1 (which in this case would be MS1 scan 3). We hypothesis that this may be what is causing the reduction in performance for Serum of the vendor TopN over our TopN, but testing that hypotheses is outside the scope of this work.

S3. Results for alternative peak picking methods

To ensure our results were not biased by a particular peak picking algorithm, we evaluated performance using two other methods. The Centwave algorithm from XCMS (Smith *et al.*, 2006) and peakonly: a recently proposed method based on a convolutional neural network (Melnikov, Tsentalovich and Yanshole, 2020). Peakonly was run with default parameters whilst XCMS parameters were tuned to give a roughly comparable number of peaks to those found with MzMine. -- the parameters are shown in Table SI-2. Coverage results are shown in Table SI-4. We can see that with both peak picking approaches, our new methods all outperform TopN. For peakonly we see that the SmartROI (with shifts) outperforms the WeightedDEW, but the differences are rather small.

	XCMS				**Peak Only**			
Method	**Beer (5393 peaks)**		**Serum (2739 peaks)**		**Beer (1556 peaks)**		**Serum (715 peaks)**	
TopN	1939		1204		857		470	
WeightedDEW	**2418**	**2317**	**1530**	**1543**	1061	995	465	476
SmartROI	2141	1996	1289	1323	1126	1072	520	555
SmartROI (shift = 1)	2289	2156	1361	1414	1219	1156	**573**	559
SmartROI (shift = 2)	2334	2186	1351	1378	**1254**	**1181**	566	**560**

Table SI-4: Coverage (number of picked peaks fragmented) for each controller for both beer and serum samples, where peaks have been picked using XCMS and PeakOnly. The first five rows show the performance of our (Python) controllers, with the shift set to be the same as Table 1 in the main paper. The final row gives the performance of the instrument method.

S4. Computing theoretical fragmentation bounds

We here provide more details on the optimal matching.

A graph `G=(V,E)` is a pair of sets `V` and `E`, where `V` is a set of *vertices* and `E` a set of *edges*. Each edge is an unordered pair of vertices. The vertices that are joined by an edge are known as the *endpoints* of the edge.

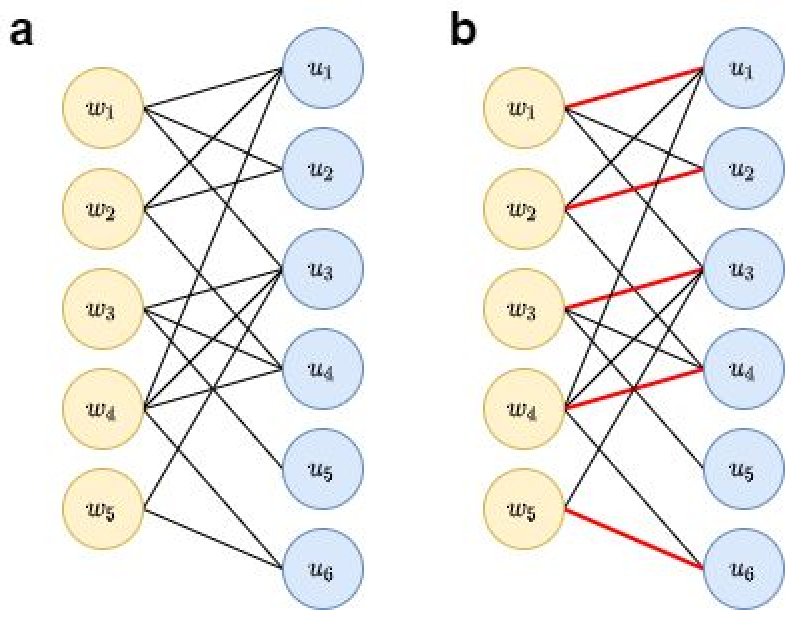

Figure SI-3: (a) Bipartite graph G and (b) G with a maximum matching shown in red. $W=\{w_1,w_2,w_3,w_4,w_5\}$ *corresponds to MS2 scans,* $U=\{u_1,u_2,u_3,u_4,u_5,u_6\}$ *corresponds to peak bounding boxes (from peak picking). Edges are drawn for MS2 scans that can be connected to bounding boxes if certain conditions are met to produce the fragmentation events.*

A bipartite graph is a graph for which V is the union of two distinct sets, W and U, say, and for which every edge has one endpoint in W, and one endpoint in V. Figure SI-3(a) shows a bipartite graph G_1 where $W=\{w_1,w_2,w_3,w_4,w_5\}$ and $U=\{u_1,u_2,u_3,u_4,u_5,u_6\}$. Here W corresponds to MS2 scans, while U corresponds to the peak boxes. A matching (Erciyes, 2018) of graph G is a subset M of E such that no two edges in M share an endpoint. A maximum matching M is a matching such that no matching M' of G contains more edges. Figure SI-3(b) shows a maximum matching of G_1 (red edges). Polynomial time algorithms exist to find maximum matchings. The problem can be solved by adding source and sink nodes and then using an optimised version of the Ford-Fulkerson algorithm (Ford and Fulkerson, 1956) to solve the resulting maximum network flow problem. This algorithm runs in $O(|V|.|E|)$ time. An alternative, faster ($O(\sqrt{|V|}.|E|)$algorithm is the Hopcroft-Karp algorithm (Hopcroft and Karp, 1973), on which we base our implementation.

S5. Optimality results

Sample	***Average Scan time (s)***		***Coverage***	
	MS1	***MS2***	***Observed***	***Optimal***
Beer (4592 peaks)	*0.56*	*0.19*	*1046*	*2955*
Serum (3032 peaks)	*0.59*	*0.19*	*656*	*1542*

Table SI-5: Observed and optimal coverages for Beer and Serum for a TopN method.

S6. Scan timings for different controller methods

	Beer		Serum	
Method	**MS1**	**MS2**	**MS1**	**MS2**
Fullscan	0.54		0.56	
TopN	0.56	0.19	0.59	0.19
SmartROI	0.68	0.19	0.67	0.18
WeightedDEW	0.61	0.19	0.61	0.19

Table SI-6: Scan timings for the different controller methods implemented on top of ViMMS. All times are averaged over one injection (iter 1), in seconds. Time for scan A is computed from the .mzML file as the difference between the scan start time of scans A and A+1 and therefore include both the time required to scan and process. As expected, MS2 times are constant, but MS1 times vary depending on the computation required to process the MS1 scan.

S7. *In-silico* optimisation of controllers

In order to optimise the parameters for the SmartROI and WeightedDEW methods we completed a grid search of a some possible parameters in simulation within the ViMMS framework. We firstly converted an .mzml file from a recent real TopN run for each sample into ViMMS chemicals, allowing us to run in-silico simulations for each method and dataset. For each method we then chose a number of possible values for the parameters as given on the x and y axes of the plots. For each combination of parameters and dataset we then ran a simulation in ViMMS and calculated the coverage (colour scale on right hand side of the plots) based on peaks picked from the original .mzml file. The final choice of parameter to use on the real sample were chosen such that they had consistent high performance across both samples for each method.

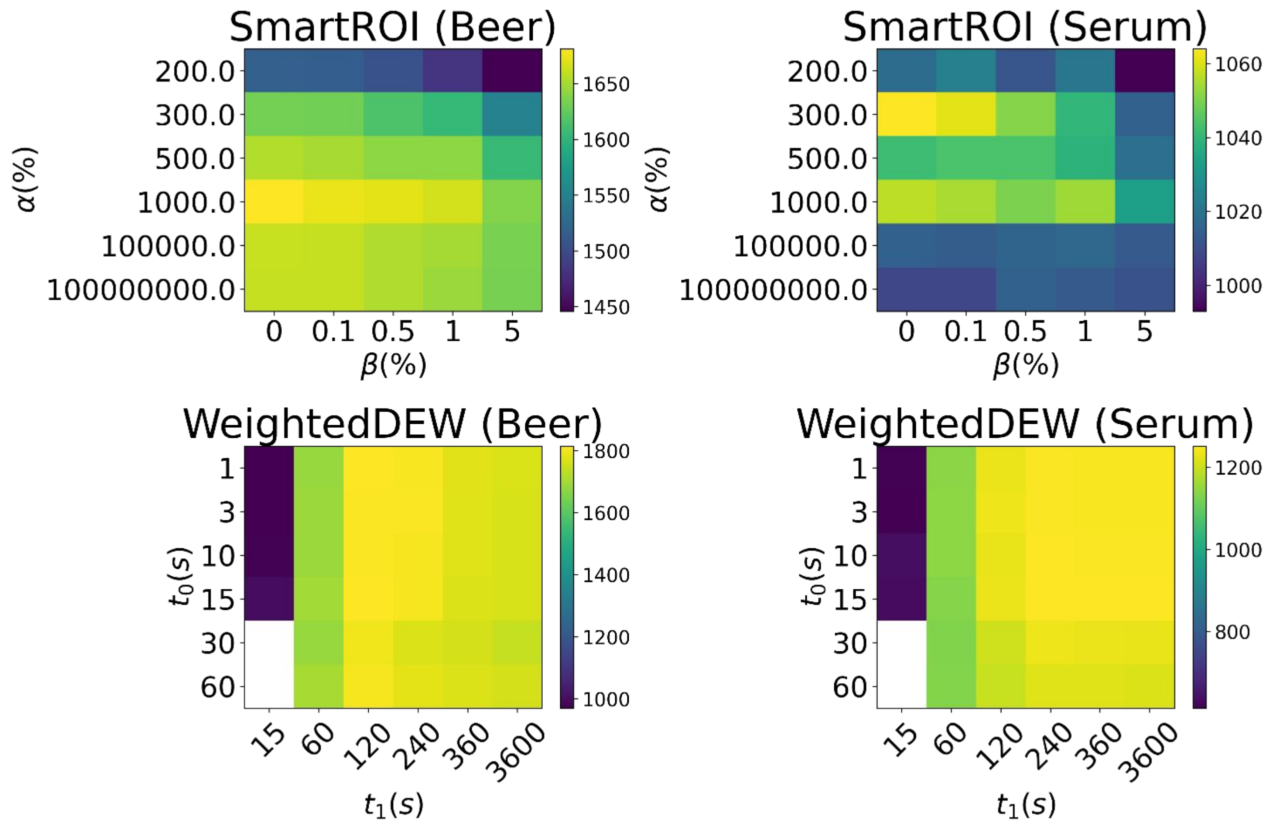

Figure SI-4: Heatmaps of simulated coverage for SmartROI (top) and WeightedDEW (bottom). The simulated results could be used to suggest appropriate parameters to use for validation on the actual instruments. White cells in the WeightedDEW results indicate invalid parameter combinations.

S8. Performance comparison for different dataset sizes

We have compared the performance of the WeightedDEW, SmartROI and Top-N methods as the complexity of the sample increased. Dataset were generated with 250, 500, 750, 1000, 1250, 1500, 1750, 2000, 2250, 2500, 2750 and 3000 Gaussian shaped chemicals using ViMMS, with 3 replicates in each case. Each method was ran with each samples with the resulting average coverage calculated. In Figure SI-5 we can see the comparative performance, with the left plot showing the average number of chemicals fragmented and the right figure showing the percentage of possible chemicals fragmented. The results of Figure SI-5 show that the methods are approximately equal in terms of performance for 1000 chemicals or less. As the number of chemicals increases both the WeightedDEW and SmartROI method outperform Top-N, with WeightedDEW then outperforming SmartROI when there are more than 2000 chemicals or more.

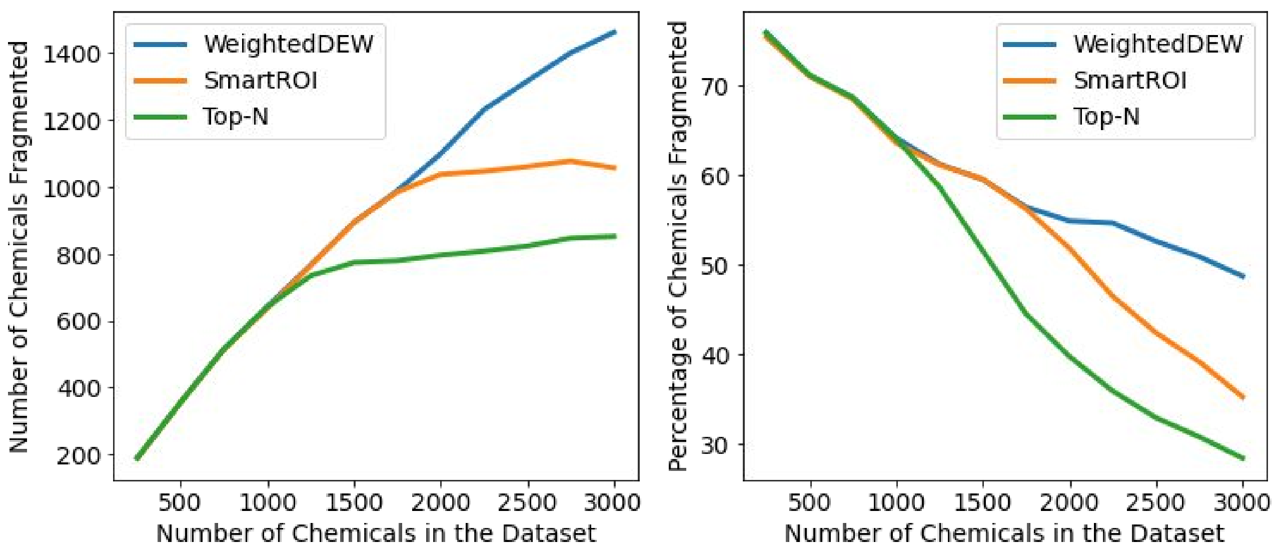

Figure SI-5: Plots showing the comparative average performance of the WeightedDEW, SmartROI and Top-N methods on dataset a varying number of true chemicals.

S9. Intensity analysis of fragmented precursor peaks

We evaluated how WeightedDEW, SmartROI and Top-N methods varied in terms of the precursor peak intensity at fragmentation. For each picked peak using MZMine2, we determined the intensity of the precursor ions when fragmentation events occurred (if multiple fragmentation events were to occur within a single picked peak, the one with the largest precursor intensity was selected). It is important to note that all methods were set to never fragment below a minimum MS1 intensity (5000).

Figure SI-6 shows the cumulative plots of fragmented precursor peaks intensities. The results in Figure SI-6A and C demonstrated that the SmartROI and WeightedDEW methods were able to fragment more precursor peaks of lower intensities that Top-N missed (right hand region of the plots) for Serum and Beer respectively. The results here agree with the increased coverage in Table 1 of the main text. Note that all fragmentation events are already above the minimum intensity for fragmentation defined by the user during the acquisition, so increased coverage means a greater number of potentially useful fragmentation spectra could be obtained.

To determine if there is a consistent difference in fragmented precursor intensities, we identified 429 and 779 common peaks that were fragmented by all three methods in the Serum and Beer samples respectively. Plotting the same cumulative plots for these common peaks (Figure SI-6 B and D), we see that the increased coverage from SmartROI and WeightedDEW comes at the cost of a slight reduction of precursor intensities at the time of fragmentation compared to Top-N (Top-N curve slightly higher than others in left-hand region). This is not surprising, as Top-N is a method that prioritises based on intensity, while the other two methods attempt to balance coverage and intensity.

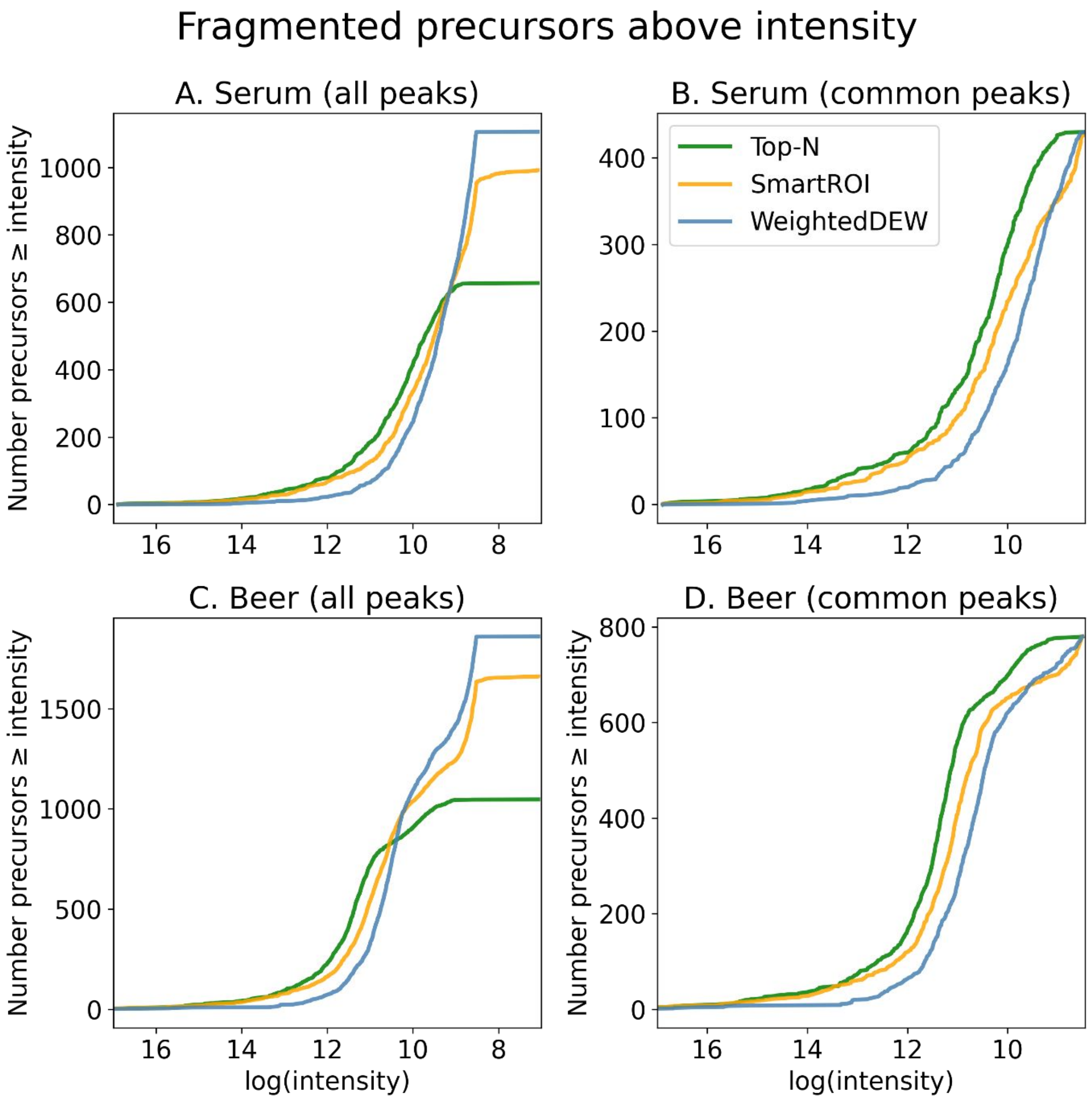

Figure SI-6: Plots showing the cumulative count of fragmented picked peaks with intensities above log(intensity). In A and C, all picked peaks fragmented by each method were counted, while in B and D, only shared common peaks fragmented across the three methods were evaluated.

References

Erciyes, K. (2018) ‘Matching’, in Erciyes, K. (ed.) *Guide to Graph Algorithms: Sequential, Parallel and Distributed*. Cham: Springer International Publishing, pp. 263–303. doi: 10.1007/978-3-319-73235-0_9.

Ford, L. R. and Fulkerson, D. R. (1956) 'Maximal Flow Through a Network', *Canadian Journal of Mathematics. Journal Canadien de Mathematiques*. Cambridge University Press, 8, pp. 399–404. doi: 10.4153/CJM-1956-045-5.

Hopcroft, J. E. and Karp, R. M. (1973) 'An n5/2 Algorithm for Maximum Matchings in Bipartite Graphs', *SIAM Journal on Computing*. Society for Industrial and Applied Mathematics, 2(4), pp. 225–231. doi: 10.1137/0202019.

Melnikov, A. D., Tsentalovich, Y. P. and Yanshole, V. V. (2020) 'Deep Learning for the Precise Peak Detection in High-Resolution LC-MS Data', *Analytical chemistry*, 92(1), pp. 588–592. doi: 10.1021/acs.analchem.9b04811.

Senko, M. W. *et al.* (2013) 'Novel parallelized quadrupole/linear ion trap/Orbitrap tribrid mass spectrometer improving proteome coverage and peptide identification rates', *Analytical chemistry*, 85(24), pp. 11710–11714. doi: 10.1021/ac403115c.

Smith, C. A. *et al.* (2006) 'XCMS: processing mass spectrometry data for metabolite profiling using nonlinear peak alignment, matching, and identification', *Analytical chemistry*, 78(3), pp. 779–787. doi: 10.1021/ac051437y.