TOFFEE – A HIGHLY EFFICIENT, LOSSLESS FILE FORMAT FOR DIA-MS

A PREPRINT

Brett Tully*
The ACRF International Centre for the Proteome of Human Cancer (ProCan®)
Children's Medical Research Institute,
Faculty of Medicine and Health, The University of Sydney
btully@cmri.org.au; brett.tully@magd.oxon.org

May 11, 2019

ABSTRACT

The closed nature of vendor file formats in mass spectrometry is a significant barrier to progress in developing robust bioinformatics software. In response, the community has developed the open mzML format, implemented in XML and based on controlled vocabularies [1]. Widely adopted, mzML is an important step forward; however, it suffers from two challenges that are particularly apparent as the field moves to high-throughput proteomics: a) large increase in file size – and corresponding increase in CPU time devoted to I/O, and b) a largely sequential I/O access pattern. Described here is 'toffee', an open, random I/O format backed by HDF5, with lossless compression that gives file sizes similar to the original vendor format and can be reconverted back to mzML without penalty. In addition to the file format, there are C++ and python libraries for creating and accessing the file format, along with a wrapper around OpenSWATH [2] that enables SWATH-MS data to be analyzed with standard algorithms. Using this library, the files can be accessed in the same manner as the Vendor file (or mzML) in a scan-by-scan manner; however, by accepting a degree of mass approximation (<5 parts per million) toffee enables data to be extracted as a two-dimensional slice analogous to an image, and thus amenable to deep-learning based peptide identification strategies. Documentation and examples are available at `https://toffee.readthedocs.io`, and all code is MIT licensed at `https://bitbucket.org/cmriprocan/toffee`.

Keywords Mass Spectrometry · File Format · Compression

There are many previous attempts at new and open formats for mass spectrometry. Some, such as mzML [1], and mz5 [3], aim to be archival formats that faithfully adopt the HUPO PSI guidelines.

*https://cmri.org.au/procan; https://brett-tully.id.au

In doing so, these formats adhere to recording data in vectors of mass over charge (m/z) and intensity pairs, with one vector for each scan of the mass spectrometer (at a recorded retention time). Typically, software will access the data in these files one scan at a time, loading all of the data for a scan before moving to the next scan. Performance gains are made by indexing the scans so that the file need not be read from the beginning each time. Regardless, algorithms are limited to taking slices of the data along the retention time (or spectrum) axis only. In contrast, other attempts such as mzDB [4], mzRTree [5], and mzTree [6], focus on random I/O access through the use of an RTree [7] data structure. This allows the data to be accessed along both the m/z (or chromatogram) axis as well as the retention time axis, at the cost of file size or mass accuracy. With toffee, I aim to combine the benefits of a variety of these approaches specifically geared towards minimizing the anticipated challenges of biobank-scale data independent acquisition mass spectrometry (DIA-MS). Generally, these benefits can be grouped into two categories: file size, and data access.

File size is crucial to manage long-term data storage and retrieval costs, in addition to minimizing the hardware that is required during any computational analysis of the raw data. For example, Figure 1 shows that converting from a vendor format to *profile* mzML – all raw data included – with the default options of 'msconvert' [8] increases the size on disk by an order of magnitude. Furthermore, tools such as OpenSWATH then cache this data to disk during analysis (requiring at least another 2x of hard disk space) and further steps in the pipeline may require raw chromatograms to also be saved to disk for reanalysis (e.g. DIAlignR [9]). If we assume that storage and analysis is being performed on the Amazon public cloud, that each raw vendor file is 3 GB (making a 30 GB mzML), and that we have 100,000 files in S3 storage to analyze, then we have 3 petabytes of data. At current prices, this equates to $63,000 US per month of on-going storage costs. With toffee, this would be $6,300 US per month. Further savings are found simply by reducing the CPU time spent waiting on disk I/O operations – in a standard OpenSWATH run with profile mzML input and sample specific spectral library, and depending on hardware setup, I/O can be as much as 25% of run-time (results not shown).

Enabling random data access is beneficial on multiple fronts. In peptide-centric approaches such as OpenSWATH, it is a significant algorithmic advantage to enable accessing data by slicing through the m/z axis. As shown in the 'openms-toffee' repository [10], operating this way is equally useful for analysis as well as testing. By providing an efficient peptide-centric interface into the data,

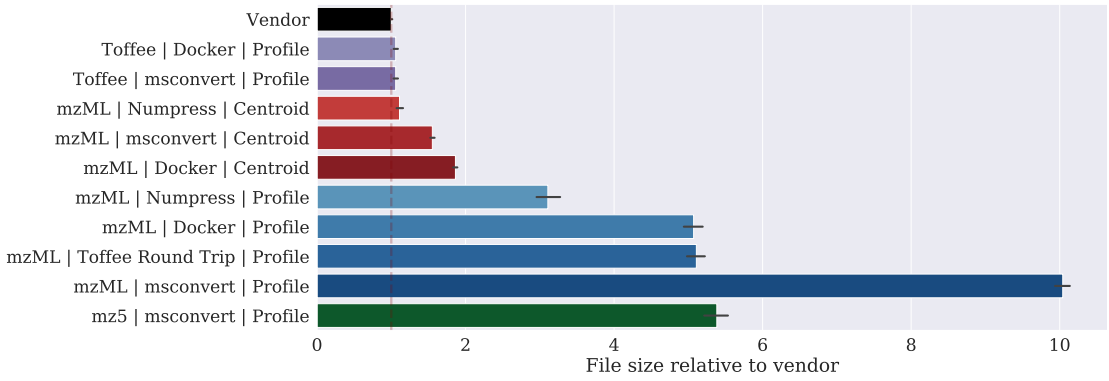

Figure 1: Comparison of several different file conversion methods, including: profile and centroided mzML created from 'msconvert' and the 'Sciex Docker image', mz5 files created using 'msconvert' and toffee files created using the method described in this work. Clearly, profile mzML files require the most storage although this can be improved based on the method of generation and if 'numpress' compression is used. Centroiding the data results in much smaller files, and toffee files are of equivalent size to the original vendor format. See the Online Methods 1.2 for a full description of all conversion methods.

algorithm developers can write tests of their implementation down to individual units of work while still retaining the complexity of real world data. This approach is fundamental to writing robust software [11]. For example, toffee can make use of its random I/O functionality to construct *in-silico* dilution experiments that mimic real-world data but in a more controlled manner against which algorithms can be tested (more details below). Finally, by accepting a degree of mass approximation (<5 parts per million, see Supplementary Material *calculate_ims.ipynb* [12]) toffee enables data to be extracted as a two-dimensional slice analogous to an image, and thus amenable to deep-learning based peptide identification strategies. Figures 2 and 3 show the raw two-dimensional data for two peptide query parameters extracted directly from a HEK293 file. Here, the red, green, and blue channels are filled with data extracted around the m/z of the precursor and product ions with offsets of 0, 1, and 2 times the isotopic carbon-12 / carbon-13 mass difference. Proof-of-concept experiments have shown that this data can be successfully used to train a convolutional neural network object detection architecture capable of detecting peptides (results not shown), or two-dimensional wavelet decomposition can be used for signal denoising (results not shown).

Importantly, toffee does not aim to be a long-term *archival* format and takes the view that this is the role of the original vendor file. While vendor lock-in is a disadvantage, the highly regulated clinical environment in which vendors operate suggests that there *should* always be a way to access

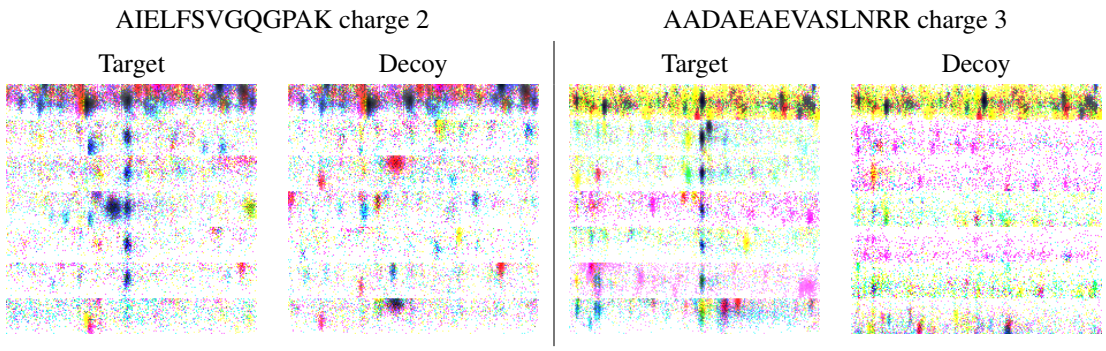

Figure 2: By accepting a mass accuracy loss of less than 5 parts per million, toffee data can be efficiently accessed in a way that is amenable for modern computer vision algorithms. For example, the images here are produced by stacking data from the mono-isotope, and isotopes 1 and 2, into the red, green and blue channels of an image respectively. The left panel shows target and corresponding data for AIELFSVGQGPAK charge 2, and the right panel is for AADAEAEVASLNRR charge 3. The top two-dimensional horizontal slice of each of these images is from the precursor ion (or MS1) data, while the remaining slices are from the six product ion (or MS2) data as specified in the spectral library. These images are zoomed to the retention time location of the peptide, however as shown in Figure 3, it is possible to keep the full retention time range if one desires.

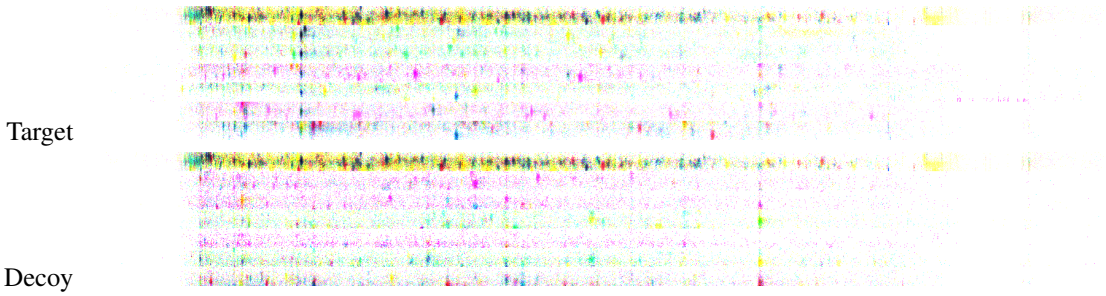

Figure 3: Zoom out of Figure 2 to show the data for AADAEAEVASLNRR charge 3 across all retention time. The peptide can be seen around 25 percent in from the left edge of the target image.

the data within these files. As such, toffee does not currently implement the PSI ontology, although attributes and metadata can be added trivially via the HDF5 back-end.

Born out of The ACRF International Centre for the Proteome of Human Cancer (ProCan®), toffee has been primarily developed for data generated on Sciex instruments, collecting data in DIA-MS mode. ProCan® aims to collect data from up to 150,000 mass spectrometer injections by 2025, thus the challenges of current file formats outlined above are acute. The following explanation of how toffee compression is achieved reflects its heritage, however, the fundamental approach of converting m/z to an integer index space should be applicable to data collected on both time-of-flight, and Orbitrap mass analyzers. In making toffee open-source, contributions on its extension by others more familiar with different instruments are highly encouraged.

Toffee's lossless compression is achieved by understanding the physics of a time-of-flight (TOF) mass analyzer. Here, a charged ion is accelerated through an electric potential; by applying Maxwell's equations on the work done on the ion and Newton's laws of motion, we can say that the m/z of the ion is related to its time of arrival at the sensor through the following relationship: $\frac{m}{z}\Big|_i = (st_{\text{arrival},i} + t_{\text{offset}})^2$, where the arrival time is a multiple of the sensor sampling rate (Δt) plus an offset (t_0) such that $\frac{m}{z}\Big|_i = [s\,(\Delta ti + t_0) + t_{\text{offset}}]^2$, or finally $\frac{m}{z}\Big|_i = [\alpha(i+\gamma)+\beta]^2$, where $\alpha = s\Delta t$, $\beta = t_{\text{offset}} + st_0 - \alpha\gamma$, and γ are the Intrinsic Mass Properties (IMS) of the injection [13]. For a full derivation see Online Methods 1.4.1. Pragmatically, given the data of an mzML file, toffee works iteratively to calculate the IMS properties; ideally this data would be exposed directly from the vendor format but this cannot always be relied upon. Interestingly, in order to ensure lossless compression, the IMS parameters must be calculated and stored *for each scan* rather than for a given injection, or even scan type (i.e. MS1 or MS2). The implication here is that the mass spectrometer performs in-line calibration that is additional to the manual calibration completed as part of routine lab operation (see Supplementary Material *calculate_ims.ipynb* [12] for more information). From this transfer function, it is now apparent that the m/z values can be converted to a vector of integer indices (i) and retention time is calculated from the scan index and the instrument cycle time. Thus, all raw data in the file can be represented as a vector of integer triplets: m/z index, retention time index, and intensity. These triplets fall onto a Cartesian grid, and thus can be stored as a compressed row storage sparse matrix [14]. This sparse matrix can be efficiently saved as three data sets in the toffee file with zlib compression provided natively by HDF5. Further details can be seen in Online Methods 1.4.2 and Supplementary Material *analysis-profile.ipynb* [12].

As mentioned previously, the 'openms-toffee' repository [10] is a wrapper around OpenSWATH to enable toffee to be used with a current best-practice and open-source SWATH-MS analysis pipeline. The MIT-licensed openms-toffee serves two purposes: to demonstrate that toffee does not introduce any artifacts to the OpenSWATH pipeline converting raw data to a quantified list of peptides, and as an exemplar of how peptide-centric data extraction can be achieved using the C++ toffee library. However, due to the architecture of OpenSWATH (in particular, challenges around thread safety), this is by no means an optimum deployment of the technology and there is significant room for CPU performance improvements once const-correctness is addressed in OpenSWATH. For that reason, no attempted has been made to push toffee or openms-toffee into the upstream 'openms' repository.

Using openms-toffee I have conducted a thorough investigation into OpenSWATH with a variety of mzML conversions, and toffee itself. Using three public data sets covering both TripleTOF5600 (Swath Gold Standard [2] and TRIC manual validation set, only the y- and b-ions are included in the analysis, [15]) and TripleTOF6600 (ProCan90, including only the first injection from each mass spectrometer, [16]) the raw vendor files were converted to mzML using 'msconvert' in both profile and vendor peak-picking centroid mode, each with and without 'msnumpress' [17], as well as the 'sciex/wiffconverter' Docker image [18] in both profile and centroid modes. Toffee files were then produced from the msconvert and Sciex Docker profile mzML files, and the toffee file back to mzML. Finally, for reference, a small subset of files were converted to mz5 for file size comparisons. Figure 1 shows that the largest mzML files are produced by 'msconvert' with no 'msnumpress' compression, and the smallest are either the lossy centroided and compressed mzML files, or the lossless profile toffee files, both of which compare in size to the vendor format.

In order to assess the quality of the data in each file conversion method, they are input into the computational pipeline as described in Online Methods 1.3 (all analysis code is included in the 'openms-toffee-paper' repository [12] and is executable on MacOS or Linux). One of the reasons for selecting the SGS and TRIC data sets for this analysis, is their inclusion of a collection of manually validated peptide query matches (PQMs). Using this information, the results of the file format pipelines are assessed by categorizing those results with a peptide retention time within 15 seconds of the manually validated peak as a *true positive*, those not within this threshold as a *false positive*, and those peaks in the manual validation, but not found using this file format as *false negatives*. One can see from Figure 4 that each of the profile files (mzML and toffee) performs equivalently, while (particularly with the TRIC data, Figure 4C) centroided mzML files have a larger number of false positives. This is further demonstrated by the increase in missing values that are seen with centroided mzML files when analyzing the ProCan90 data set, see Figure 4D. From these results, we conclude that there is no meaningful difference to the final quantified peptide results from OpenSWATH regardless of using profile data from 'msconvert' (with or without 'msnumpress'), 'Sciex Docker', or toffee; however, there is a drop in performance when using centroided data. One may notice error bars on these figures; as the spectral library for both SGS and TRIC data sets is relatively small, run-to-run variation is introduced by stochasticity in the False Discovery Rate (FDR) scoring of PyProphet. To account for this, we apply PyProphet 10 times for each experiment and the plots summarize these results.

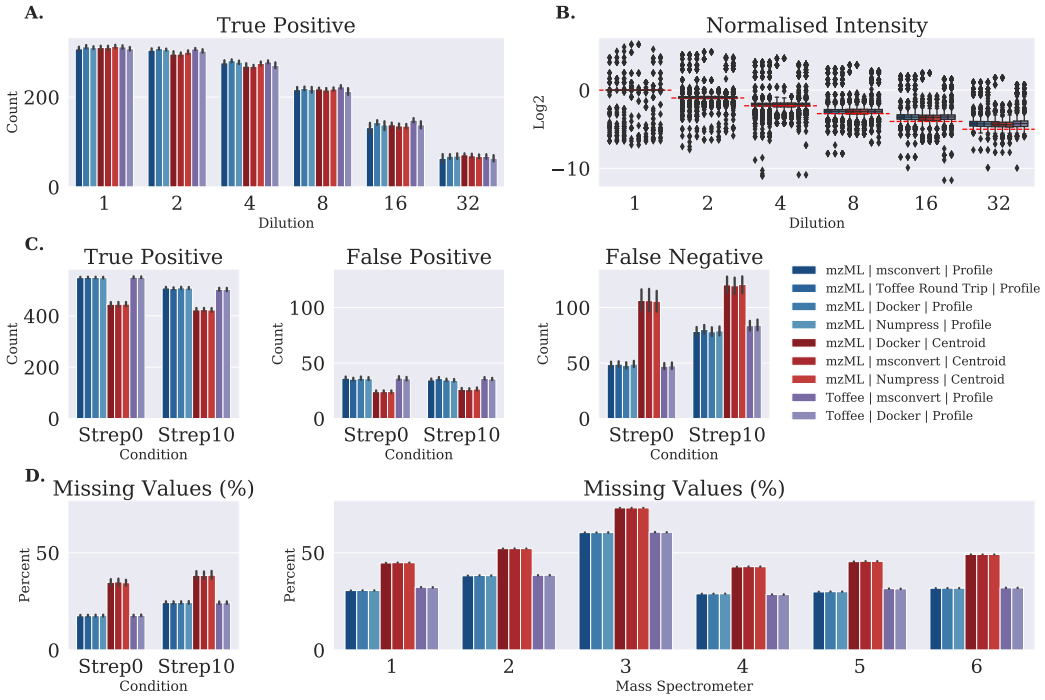

Figure 4: Analysis of four profile mzML file conversions, three centroid mzML file conversions, and two toffee file conversions. **A.** shows the number of true positive discoveries in the Swath Gold Standard data set and **B.** is their normalized intensity – from this data set there is essentially no difference between file conversion method. **C.** shows the confusion matrix for the TRIC manual validation data set clearly highlighting the degradation of centroiding during mzML creation. **D.** shows the percentage of missing values in the final peptide matrix, mimicking the false negative plot, for both the TRIC and ProCan90 data sets.

Having confirmed that both the toffee file format and openms-toffee pipeline are equivalent to the profile mzML and OpenSWATH pipeline, a novel use of toffee files can be explored. It is of critical importance to developing robust scientific software that one can isolate and test algorithms with controlled inputs and known expected outputs. For example, in high-energy physics, the US National Laboratories have developed the Tri-Lab test suite that pits algorithms against *toy* problems with analytic solutions [19]. In the context of mass spectrometry, this is much more difficult due to the stochastic nature of the instruments, and of the subject under study. Often, validation experiments are based around injecting samples that contain a controlled dilution of a known group of peptides and technical replicates are performed to normalize the stochastic impact of the instrument. An alternative approach, is to create data *in-silico* through analytic models [20, 21],

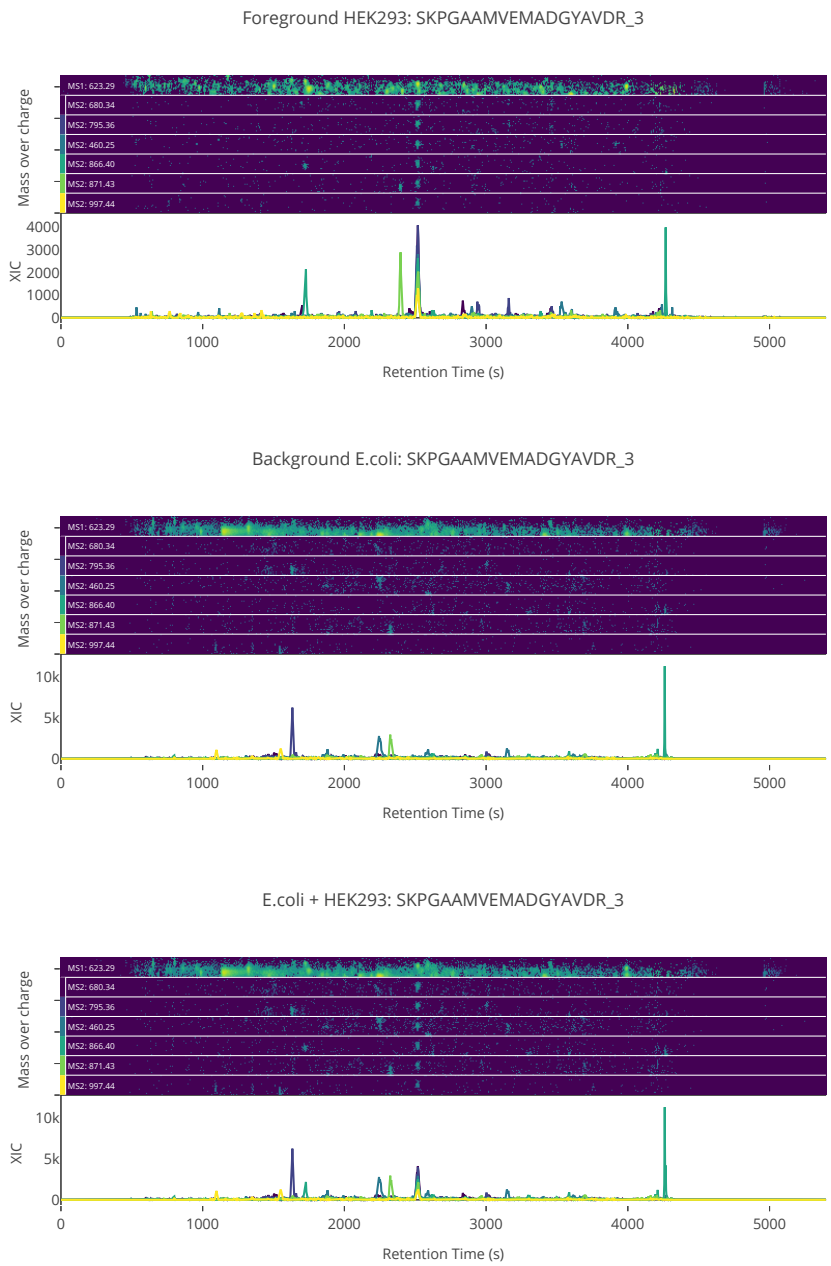

Figure 5: Combining data from a background file of *E.coli* and a foreground file of HEK293 for the peptide SKPGAAMVEMADGYAVDR charge 3. Shown in the top row are the HEK293 and *E.coli* data extracted for MS1 and the top six MS2 fragments. In the bottom row, the data (isolated at the peak retention time) from the HEK293 has been added to the background with a theoretical dilution of 1.

however, it is highly improbable that the noise artificially added to these models is an accurate reflection of data in reality. Toffee offers us a new approach. By treating the data as triplets on a Cartesian grid, it becomes trivial to extract data from one toffee file, the foreground data, and add

it to another toffee file, the background data, to create an entirely new toffee file. Referred to as an *in-silico dilution series*, data for specific peptides are extracted from the foreground file, scaled based on a *theoretical* dilution, and placed into the background file at a known retention time (see Figure 5).

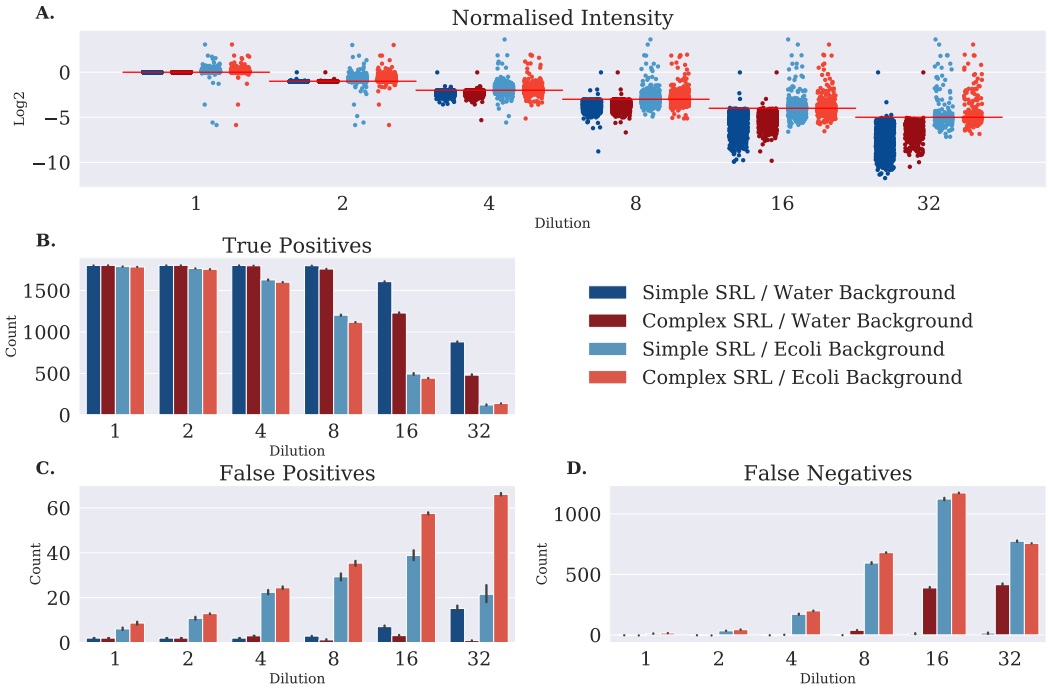

Figure 6: Analysis of the *in-silico* dilution series using simple and complex spectral libraries. **A.** shows the normalized intensity of each peptide query match against the expected value found in the most concentrated water background. **B.–D.** shows the confusion matrix found by comparing the retention time of the discovered peptide query match with the theoretical retention time at which the raw data was added. This clearly shows that the *E.coli* background leads to less true discoveries, and more false discoveries. Furthermore, FDR control with the complex spectral library leads to increased false negatives at high dilutions.

In this study, two in-silico dilutions are constructed: one with a water background and another with an *E.coli* background, such that the impact of background noise can be assessed. Furthermore, we analyze these files with two spectral libraries: a 'Simple' SRL containing just those peptides that were added in-silico, and a 'Complex' SRL that includes the in-silico peptides, plus an SRL derived from this *E.coli* sample. By comparing the results from these two SRLs, we can investigate the impact of the π_0 parameter discussed at length in [22]. Figure 6A shows the normalized intensity

quantified by the openms-toffee pipeline; as expected, the dilution curve of the water background is bounded on the upper limit by the theoretical dilution (red horizontal lines) and the data that that is below the theoretical limit is due to signal truncation at the lower limit of detection of the mass spectrometer (as set by the background file). In contrast, the *E.coli* dilution series often exceeds the theoretical limit and shows that the OpenSWATH peak integration algorithms are at risk of quantifying noise by treating the extracted ion chromatogram as a one-dimensional signal rather than its two-dimensional reality. Figures 6B-6D show the confusion matrix where PQMs detected more than 10 seconds from the theoretical retention time are labelled as false positives, and PQMs correctly identified in the Simple SRL / Water Background for a given dilution and not detected in the file of interest are labelled as false negatives. From Figures 6B and 6C the complexity of the SRL seems unimportant at low dilutions and false positives remain around the expected FDR. At dilutions 4 and above, the more complex *E.coli* background makes it harder to distinguish between target and decoy, leading to a more stringent FDR, which leads to a marked increase in the number of false negatives when compared to the water background (see Figure 6D).

In summary, the challenges of file size and data access with current open formats for data independent acquisition mass spectrometry are acute when a scientific program needs to operate at biobank-scale. These challenges significantly increase the cost and complexity of data management and analysis, and hold back the progress of writing efficient algorithms that are routinely tested on real-world data. Toffee aims to address these issues by taking a first-principles approach to understanding the raw data, and translating those findings into a best-practice software library. Code is released with an MIT license, python packages should be easily installed for MacOS and Linux using 'conda', both toffee and an exemplar implementation wrapping OpenSWATH are available via version-controlled Docker images, and all analyses performed in this paper are available as Jupyter notebooks.

1 Online Methods

1.1 Open Science

To the extent possible, this work aims to have full and automated reproducibility; this would not be possible without the extensive community efforts around technologies such as Docker [23], Project Jupyter [24], and pandas [25].

1.1.1 Data sets

Three publicly available data sets are used in the current work: Swath Gold Standard and TRIC data available from PeptideAtlas raw data repository with accession number PASS00289 [2] and PASS00788 [15], respectively, and the ProCan90 dataset can be obtained from the PRIDE archive under the identifier PXD011093 [16].

1.1.2 Software and Analysis

- Code repositories
  - `https://bitbucket.org/cmriprocan/toffee`
  - `https://bitbucket.org/cmriprocan/openms-toffee`
  - `https://bitbucket.org/cmriprocan/dia-test-data` – a resource of small DIA-MS files that are used in continuous integration regression tests
  - `https://bitbucket.org/cmriprocan/openms-toffee-paper` – all analyses produced for this paper
  - `https://bitbucket.org/cmriprocan/openms` – a versioned Docker file for creating the base OpenMS Docker image
- Documentation
  - `https://toffee.readthedocs.io`
  - `https://openms-toffee.readthedocs.io`
- Docker images
  - `cmriprocan/toffee`
  - `cmriprocan/openms-toffee`
  - `cmriprocan/openms`
- Python conda library: `conda install -c cmriprocan -c plotly toffee`

1.2 File conversion

Original Sciex wiff files were converted to various forms of mzML, mz5, and toffee files. While conversion methods are provided here for reference, a self-contained script is included in the Supplementary Material *convert_mzml.py* [12].

mzML | msconvert | Profile: Conversion of wiff to mzML using msconvert version 3.0.18304 in a windows environment

```
msconvert.exe --mzML -z \
    --outfile ${mzml_fname} \
    ${wiff_fname}
```

mzML | msconvert | Centroid: Conversion of wiff to mzML using msconvert version 3.0.18304 in a windows environment, and applying vendor peak picking to centroid the data

```
msconvert.exe --mzML -z \
    --filter "peakPicking vendor" \
    --outfile ${mzml_fname} \
    ${wiff_fname}
```

mzML | Numpress | Profile: Conversion of wiff to mzML using msconvert version 3.0.18304 in a windows environment, and applying the Numpress compression

```
msconvert.exe --mzML -z \
    --numpressLinear \
    --outfile ${mzml_fname} \
    ${wiff_fname}
```

mzML | Numpress | Centroid: Conversion of wiff to mzML using msconvert version 3.0.18304 in a windows environment, and applying the Numpress compression algorithm and vendor peak picking to centroid the data

```
msconvert.exe --mzML -z \
    --numpressLinear \
    --filter "peakPicking vendor" \
    --outfile ${mzml_fname} \
    ${wiff_fname}
```

mz5 | msconvert | Profile: Conversion of wiff to mz5 using the Dockerised version of msconvert `chambm/pwiz-skyline-i-agree-to-the-vendor-licenses:3.0.19073-85be84641` [26]

```
wine msconvert -z --mz5 \
```

```
    --outfile ${mz5_fname} \
    ${wiff_fname}
```

mzML | Docker | Profile: Conversion of wiff to mzML using the publicly available Sciex Docker image `sciex/wiffconverter:0.9` [18]

```
mono \
    /usr/local/bin/sciex/wiffconverter/OneOmics.WiffConverter.
   exe \
    WIFF ${wiff_fname} \
    -profile MZML ${mzml_fname} \
    --zlib --index
```

mzML | Docker | Centroid: Conversion of wiff to mzML using the publicly available Sciex Docker image `sciex/wiffconverter:0.9` [18] and applying the included peak picking algorithm

```
mono \
    /usr/local/bin/sciex/wiffconverter/OneOmics.WiffConverter.
   exe \
    WIFF ${wiff_fname} \
    -centroid MZML ${mzml_fname} \
    --zlib --index
```

Toffee | msconvert | Profile: Conversion of the msconvert profile mzML to toffee using the toffee Docker image `cmriprocan/toffee:0.12.16` [27]

```
mzml_to_toffee ${mzml_fname} ${tof_fname}
```

Toffee | Docker | Profile: Conversion of the Sciex Docker profile mzML to toffee using the toffee Docker image `cmriprocan/toffee:0.12.16` [27]

```
mzml_to_toffee ${mzml_fname} ${tof_fname}
```

mzML | Toffee Round Trip | Profile: Using the toffee file created using the Sciex Docker profile mzML, convert back to mzML using the toffee Docker image `cmriprocan/toffee:0.12.16` [27]

```
toffee_to_mzml ${tof_fname} ${mzml_fname}
```

1.3 DIA pipeline

All analysis for raw data was completed using the Docker image `cmriprocan/openms-toffee:0.13.12.dev` [28] and a complete python script for marshalling analysis is included in the Supplementary Material *pipeline.py* [12]. In short, the current best practice OpenSWATH workflow is used whereby the spectral library is provided in SQLite format, scores are saved to an `osw` result file, and PyProphet (version 2.0.4; git hash d35a53af86131e7c4eb57bbb09be8935a1f30c70) FDR control is applied at the peak-group, peptide, and protein levels. For the latter two, scores are calculated across the full cohort of the experiment (i.e. the 'global' context is used); in a departure from the defaults, the `-parametric` model is used as it was found to be less conservative on the small SRLs used in this study. mzML files are analyzed using a modified version of OpenMS v2.4.0 (`CMRI-ProCan/OpenMS` version `CMRI-ProCan-v1.1.2` [29]). This same version is used as the basis for the code in the openms-toffee bitbucket repository [10], a wrapper around OpenMS that enables toffee files to be used in conjunction with the algorithms of OpenSWATH. Furthermore, openms-toffee has a series of tests that ensure regressions are not introduced to OpenSWATH scoring algorithms when the version of OpenMS is updated. Finally, `OpenMSToffeeWorkflow` is used as a drop-in replacement for `OpenSwathWorkflow` when analyzing toffee files.

1.4 Toffee file format

1.4.1 Time of Flight (TOF) m/z transfer function

The mass analyzer in a TOF mass spectrometer is a digital sensor that samples at a constant frequency. It effectively measures the time taken for a charged ion to move a known distance (d) through an electric field of known strength (U). Thus, measurements are made at integer multiples (i) of a constant time interval (Δt), such that

$$t_{\text{arrival},i} = \Delta t i + t'_0 \tag{1}$$

where t'_0 is an offset applied such that $t_{\text{arrival},0}$ makes sense. This time of arrival is related to the mass over charge of the ion through the electric potential energy, $E_p = zU$, and kinetic energy, $E_k = \frac{1}{2}mv^2 = \frac{1}{2}md^2/t^2$, such that

$$t_{\text{arrival},i} = \frac{d}{\sqrt{2U}}\sqrt{\frac{m}{z}\Big|_i} \tag{2}$$

With this in mind, it becomes possible to convert mass over charge values to an integer index through the following,

$$i = \frac{d}{\Delta t\sqrt{2U}}\sqrt{\frac{m}{z}\Big|_i} - \frac{t'_0}{\Delta t} \tag{3}$$

or equivalently,

$$\sqrt{\frac{m}{z}\Big|_i} = \frac{\Delta t\sqrt{2U}}{d}\left(i + \frac{t'_0}{\Delta t}\right) \tag{4}$$

By introducing the intrinsic mass spacing parameters, α, β, and γ, we get

$$\frac{m}{z}\Big|_i = \left[\alpha\left(i + \gamma\right) + \beta\right]^2 \tag{5}$$

where

$$\alpha = \frac{\Delta t\sqrt{2U}}{d} \tag{6}$$

$$\gamma = \text{int}\left(\text{ceil}\left(\frac{t'_0}{\Delta t}\right)\right) \tag{7}$$

$$\beta = \alpha\left(\frac{t'_0}{\Delta t} - \gamma\right) \tag{8}$$

1.4.2 HDF5 structure

The toffee HDF5 file is structured as follows:

attr:CREATED_BY_LIBRARY_VERSION (str) the version of the toffee library used to create the file

attr:FILE_FORMAT_MAJOR_VERSION (int) the major version of the toffee file format of this file

attr:FILE_FORMAT_MINOR_VERSION (int) the major version of the toffee file format of this file

attr:IMSType (str) the IMS type of the file. Currently, toffee is tested on "TOF", however, infrastructure is in place for "Orbitrap"

attr:metadataXML (int) An mzML style XML header that mimics the header of a normal mzML file

group:ms1 A group representing the MS1 data

- **attr:IMSAlpha (double)** the median IMS α parameter for this window
- **attr:IMSBeta (double)** the median IMS β parameter for this window
- **attr:IMSGamma (int)** the IMS γ parameter for this window
- **attr:firstScanRetentionTimeOffset (double)** the first retention time for scans in this window
- **attr:scanCycleTime (double)** the cycle time between scans for this window – this is expected (but not enforced) to be constant for all windows in the toffee file
- **attr:precursorLower (double)** the lower precursor m/z bound of this window (-1 / ignored for MS1)
- **attr:precursorCenter (double)** the center of precursor m/z bounds of this window (-1 / ignored for MS1)
- **attr:precursorUpper (double)** the upper precursor m/z bound of this window (-1 / ignored for MS1)
- **dataset:IMSAlphaPerScan (double)** the IMS α parameter for each scan of this window
- **dataset:IMSBetaPerScan (double)** the IMS β parameter for each scan of this window
- **dataset:retentionTimeIdx (unsigned int32)** the equivalent of the 'index pointer' of a compressed sparse row matrix. For toffee, there is one entry for each scan in the window and the value represents the **last** index of the `imsCoord` and `intensity` that is valid for this scan. For example `start = retentionTimeIdx[i - 1]; end = retentionTimeIdx[i]; scan_i_intensity = intensity[start:end]`.
- **dataset:imsCoord (unsigned int32)** the IMS m/z coordinate vector
- **dataset:intensity (unsigned int32)** the intensity vector

group:ms2-(001, ..., n) Groups with the same format as 'ms1' that represent the MS2 data

Acknowledgements

ProCan® is supported by the Australian Cancer Research Foundation, Cancer Institute New South Wales (NSW) (2017/TPG001, REG171150), NSW Ministry of Health (CMP-01), University of Sydney, National Breast Cancer Foundation (IIRS-18-164), Cancer Council NSW (IG 18-01), Ian Potter Foundation, the Medical Research Futures Fund (MRFF-PD), and the National Health and Medical Research Council of Australia (GNT1047070; GNT1170739).

This project was supported by equipment grants from the Australian Cancer Research Foundation (ACRF), Australia, the Cancer Institute NSW (CINSW) as a Research Equipment Grant (TPG), The Ian Potter Foundation, and Perpetual IMPACT Philanthropy. It was supported by project and Fellowship grants from the Office for Health and Medical Research (OHMR) NSW, from the National Health & Medical Research Council (NHMRC) Australia (GNT1137064), and from the Cancer Council NSW (IG 18-01).

References

[1] Lennart Martens, Matthew Chambers, Marc Sturm, Darren Kessner, Fredrik Levander, Jim Shofstahl, Wilfred H. Tang, Andreas Römpp, Steffen Neumann, Angel D. Pizarro, Luisa Montecchi-Palazzi, Natalie Tasman, Mike Coleman, Florian Reisinger, Puneet Souda, Henning Hermjakob, Pierre-Alain Binz, and Eric W. Deutsch. mzml—a community standard for mass spectrometry data. *Molecular & Cellular Proteomics*, 10(1), 2011.

[2] Hannes L. Röst, George Rosenberger, Pedro Navarro, Ludovic Gillet, Saša M Miladinović, Olga T. Schubert, Witold Wolski, Ben C. Collins, Johan Malmström, Lars Malmström, and Ruedi Aebersold. OpenSWATH enables automated, targeted analysis of data-independent acquisition MS data. *Nature Biotechnology*, 32(3):219–223, mar 2014.

[3] Mathias Wilhelm, Marc Kirchner, Judith A. J. Steen, and Hanno Steen. mz5: Space- and time-efficient storage of mass spectrometry data sets. *Molecular & Cellular Proteomics*, 11(1), 2012.

[4] David Bouyssié, Marc Dubois, Sara Nasso, Anne Gonzalez de Peredo, Odile Burlet-Schiltz, Ruedi Aebersold, and Bernard Monsarrat. mzDB: A File Format Using Multiple Index-

ing Strategies for the Efficient Analysis of Large LC-MS/MS and SWATH-MS Data Sets. *Molecular & Cellular Proteomics*, 14(3):771–781, 2015.

[5] Sara Nasso, Francesco Silvestri, Francesco Tisiot, Barbara Di Camillo, Andrea Pietracaprina, and Gianna Maria Toffolo. An optimized data structure for high-throughput 3D proteomics data: mzRTree. *Journal of Proteomics*, 73(6):1176–1182, 2010.

[6] Kyle Handy, Jebediah Rosen, André Gillan, and Rob Smith. Fast, axis-agnostic, dynamically summarized storage and retrieval for mass spectrometry data. *PLoS ONE*, 12(11):1–14, 2017.

[7] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA*, pages 47–57. ACM, 1984.

[8] Darren Kessner, David Agus, Matt Chambers, Parag Mallick, and Robert Burke. ProteoWizard: open source software for rapid proteomics tools development. *Bioinformatics*, 24(21):2534–2536, 07 2008.

[9] Shubham Gupta, Sara Ahadi, Wenyu Zhou, and Hannes Röst. Dialignr provides precise retention time alignment across distant runs in dia and targeted proteomics. *Molecular & Cellular Proteomics*, 18(4):806–817, 2019.

[10] CMRI ProCan Software Engineering. Bitbucket code repository for openms-toffee. `https://bitbucket/cmriprocan/openms-toffee`, 2019. [Online; accessed 18-April-2019].

[11] Greg Wilson, D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, and Paul Wilson. Best practices for scientific computing. *PLOS Biology*, 12(1):1–7, 01 2014.

[12] Brett Tully. Analysis code for openms-toffee paper. `https://bitbucket/cmriprocan/openms-toffee-paper`, 2019. [Online; accessed 18-April-2019].

[13] Luke Schneider. Mass spectral data processing. Technical report, Veritomyx, 11 2016.

[14] Wikipedia contributors. Sparse matrix — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Sparse_matrix&oldid=892846660`, 2019. [Online; accessed 18-April-2019].

[15] Hannes L Röst, Yansheng Liu, Giuseppe D'Agostino, Matteo Zanella, Pedro Navarro, George Rosenberger, Ben C Collins, Ludovic Gillet, Giuseppe Testa, Lars Malmström, and Ruedi

Aebersold. Tric: an automated alignment strategy for reproducible protein quantification in targeted proteomics. *Nature methods*, 13(9):777—783, September 2016.

[16] Sean Peters, Peter G. Hains, Natasha Lucas, Phillip J. Robinson, and Brett Tully. A case study and methodology for openswath parameter optimization using the procan90 data set and 45,810 computational analysis runs. *Journal of Proteome Research*, 18(3):1019–1031, 2019. PMID: 30652484.

[17] Johan Teleman, Andrew W. Dowsey, Faviel F. Gonzalez-Galarza, Simon Perkins, Brian Pratt, Hannes L. Röst, Lars Malmström, Johan Malmström, Andrew R. Jones, Eric W. Deutsch, and Fredrik Levander. Numerical compression schemes for proteomics mass spectrometry data. *Molecular & Cellular Proteomics*, 13(6):1537–1542, 2014.

[18] Sciex. Docker image: sciex/wiffconverter:0.9. `https://hub.docker.com/r/sciex/wiffconverter`, 2018. [Online; accessed 18-April-2019].

[19] W. J. Rider S. Brandon C. Woodward P. Knupp T. G. Trucano J. S. Brock, J. R. Kamm. Verification test suite for physics simulation codes. Technical report, Lawrence Livermore National Laboratory, 12 2006. [Online; accessed 18-April-2019].

[20] Chris Bielow, Stephan Aiche, Sandro Andreotti, and Knut Reinert. Mssimulator: Simulation of mass spectrometry data. *Journal of Proteome Research*, 10(7):2922–2929, 2011. PMID: 21526843.

[21] Muaaz Gul Awan and Fahad Saeed. Mass-simulator: A highly configurable simulator for generating ms/ms datasets for benchmarking of proteomics algorithms. *PROTEOMICS*, 18(20):1800206, 2018.

[22] George Rosenberger, Isabell Bludau, Uwe Schmitt, Moritz Heusel, Christie L Hunter, Yansheng Liu, Michael J MacCoss, Brendan X MacLean, Alexey I Nesvizhskii, Patrick G A Pedrioli, Lukas Reiter, Hannes L Röst, Stephen Tate, Ying S Ting, Ben C Collins, and Ruedi Aebersold. Statistical control of peptide and protein error rates in large-scale targeted data-independent acquisition analyses. *Nature Methods*, 14:921, aug 2017.

[23] Docker. Docker – build, ship, and run any app, anywhere. `https://www.docker.com/`, 2018. [Online; accessed 5-September-2018].

[24] Project Jupyter. Project jupyter | home. `http://jupyter.org/`, 2018. [Online; accessed 5-September-2018].

[25] Numfocus. Python data analysis library – pandas: Python data analysis library. https://pandas.pydata.org/, 2018. [Online; accessed 5-September-2018].

[26] Matt Chambers. chambm/pwiz-skyline-i-agree-to-the-vendor-licenses:3.0.19073-85be84641. https://hub.docker.com/r/chambm/pwiz-skyline-i-agree-to-the-vendor-licenses, 2019. [Online; accessed 18-April-2019].

[27] CMRI ProCan Software Engineering. Docker image: cmriprocan/toffee:0.12.16. https://hub.docker.com/r/cmriprocan/toffee, 2019. [Online; accessed 18-April-2019].

[28] CMRI ProCan Software Engineering. Docker image: cmriprocan/openms-toffee:0.13.12.dev. https://hub.docker.com/r/cmriprocan/openms-toffee, 2019. [Online; accessed 18-April-2019].

[29] CMRI ProCan Software Engineering. Openms fork. https://github.com/CMRI-procan/OpenMS, 2018. [Online; accessed 5-September-2018].