Supporting Information

mzMLb: a future-proof raw mass spectrometry data format based on standards-compliant mzML and optimized for speed and storage requirements

Ranjeet S. Bhamber[1], Andris Jankevics,[2], Eric W Deutsch[3], Andrew R Jones[4], Andrew W Dowsey[1]*

1. *Department of Population Health Sciences and Bristol Veterinary School, University of Bristol, Bristol BS8 2BN, United Kingdom*
2. *School of Biosciences and Phenome Centre Birmingham, University of Birmingham, Birmingham B15 2TT, United Kingdom*
3. *Institute for Systems Biology, Seattle, Washington 98109, United States*
4. *Institute of Integrative Biology, University of Liverpool, Liverpool L69 7ZB, United Kingdom*

** Corresponding Author; andrew.dowsey@bristol.ac.uk; +44 (0) 117 3319193*

Contents

MS Data File Name	Resolution	Type	File Size (MB)	Vendor	Peak Picked	Instrument
101125_JT_pl3_05	low	SRM	13.2	Thermo	no	TSQ Vantage
110620_fract_scxB05	low	DDA	16.2	Thermo	no	LCQ
peakPicked_121213_PhosphoMRM_TiO2_discovery	high	DDA	201.2	Thermo	yes	Orbitrap XL
121213_PhosphoMRM_TiO2_discovery	high	DDA	496.7	Thermo	no	Orbitrap XL
AgilentQToF	high	DDA	4197.51	Agilent	no	QToF
ABSciexTripleToF	high	SWATH DIA	2729.429	ABI Sciex	no	Triple ToF
SynaptG2	high	DDA	4348.531	Waters	no	Synapt G2
QExactive	high	DDA	1639.345	Thermo	no	Q-Exactive
16_PBQC-10_522-16470	high	MS full	700.9	Agilent	no	QTOF
Orbitrap_Exactive_MS1	high	MS full	503.2	Thermo	no	Orbitrap Exactive
Orbitrap_LTQ_MS1	high	MS full	40	Thermo	no	Orbitrap XL
20150813_11_pos_MM	high	Ion Mobility	49.9	Agilent	no	Agilent 6560

Table S1. The raw MS data files used in the validation, together with MS instrument and method information.

	Error				Truncation mzMLb		File Size (MB)		
MS Data File	Numpress mz	mzMLb - mz	Numpress - Intensity	mzMLb - Intensity	mz	Intensity	Vendor	Numpress	mzMLb
101125_JT_pl3_05	-	-	0.000041	0.000021	-	14	13.2	2.7	0.43
110620_fract_scxB05	5.500E-10	4.657E-10	0.000200	0.000122	21	10	16.2	22.8	7.9
peakPicked_121213_PhosphoMRM_TiO2_discovery	7.566E-10	0.000E+00	0.000139	0.000122	29	10	201.2	109.6	45.5
121213_PhosphoMRM_TiO2_discovery	7.566E-10	4.656E-10	0.000200	0.000122	21	10	496.7	323.5	368.8
AgilentQToF	2.635E-10	2.328E-10	0.000199	0.000118	20	10	4197.51	5874.6	6235.4
ABSciexTripleToF	1.768E-09	9.313E-10	0.000150	0.000120	22	10	2729.429	6766.2	6917.6
SynaptG2	2.000E-09	0.000E+00	0.000147	0.000000	29	10	4348.531	16855.2	1465.6
QExactive	2.000E-09	1.863E-09	0.000200	0.000122	23	10	1639.345	1608.4	1288
16_PBQC-10_522-16470	2.328E-10	2.328E-10	0.000200	0.000121	20	10	700.9	715.5	936.3
Orbitrap_Exactive_MS1	1.995E-09	1.862E-09	0.000200	0.000122	23	10	503.2	183.5	147.3
Orbitrap_LTQ_MS1	5.771E-10	4.653E-10	0.000194	0.000122	21	10	40	30.1	30.8
20150813_11_pos_MM	1.990E-09	1.857E-09	0.000164	0.000091	23	10	49.9	134.2	93.5

Table S2. mzMLb optimized values of the mantissa for both m/z and intensities for the data files listed in Table S1. The associated errors and files sizes for mzML with Numpress and mzMLb are also shown. Both formats used zlib compression with a compression strength of 4. For all results we show the maximum error across the whole dataset.

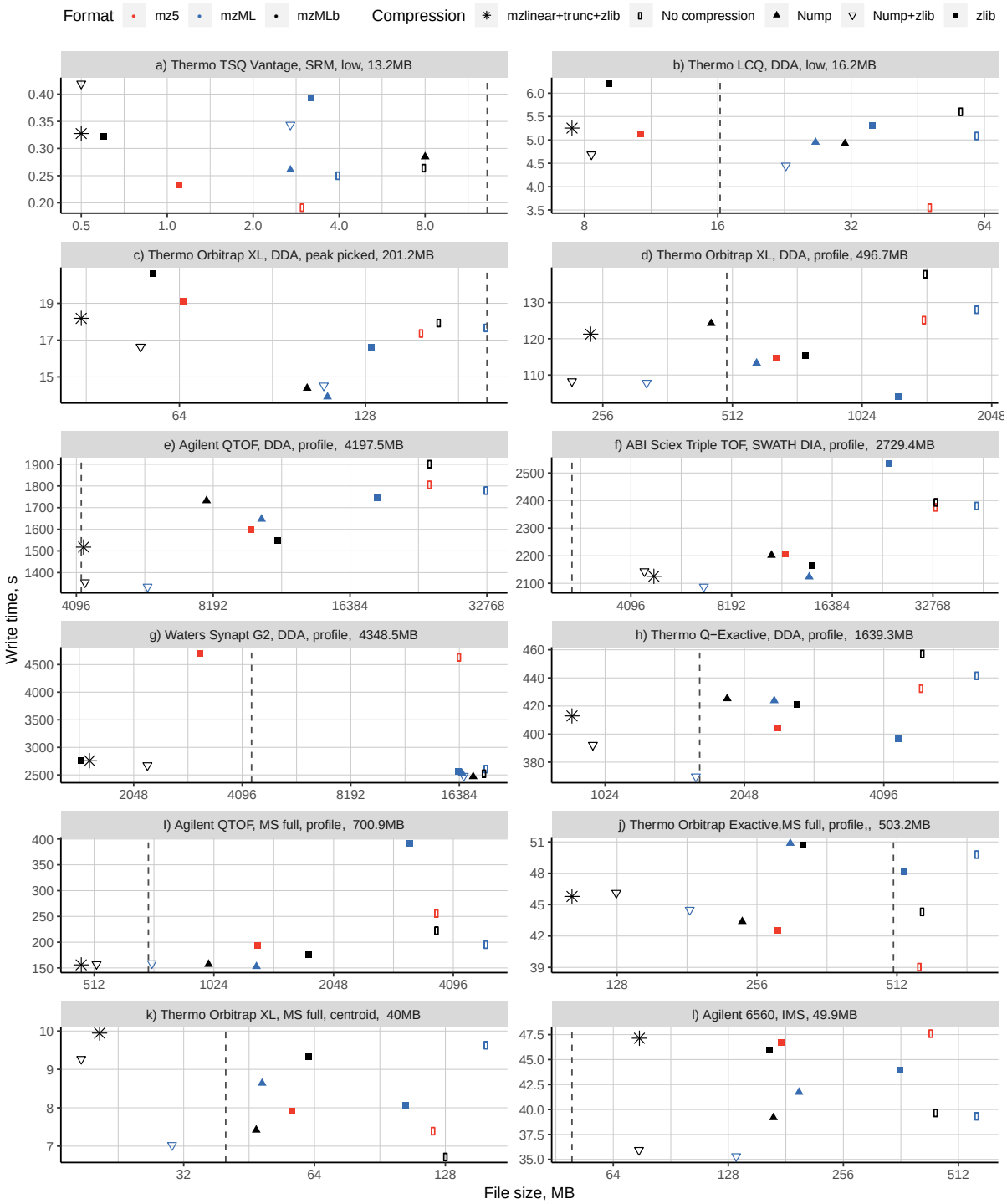

Figure S1. Summary data showing write times and file sizes for all datasets using the 3 formats; mzML, mz5 and mzMLb with 5 different compression combinations spanning both lossless and lossy configurations. The original vendor file sizes are represented by the vertical dashed line.

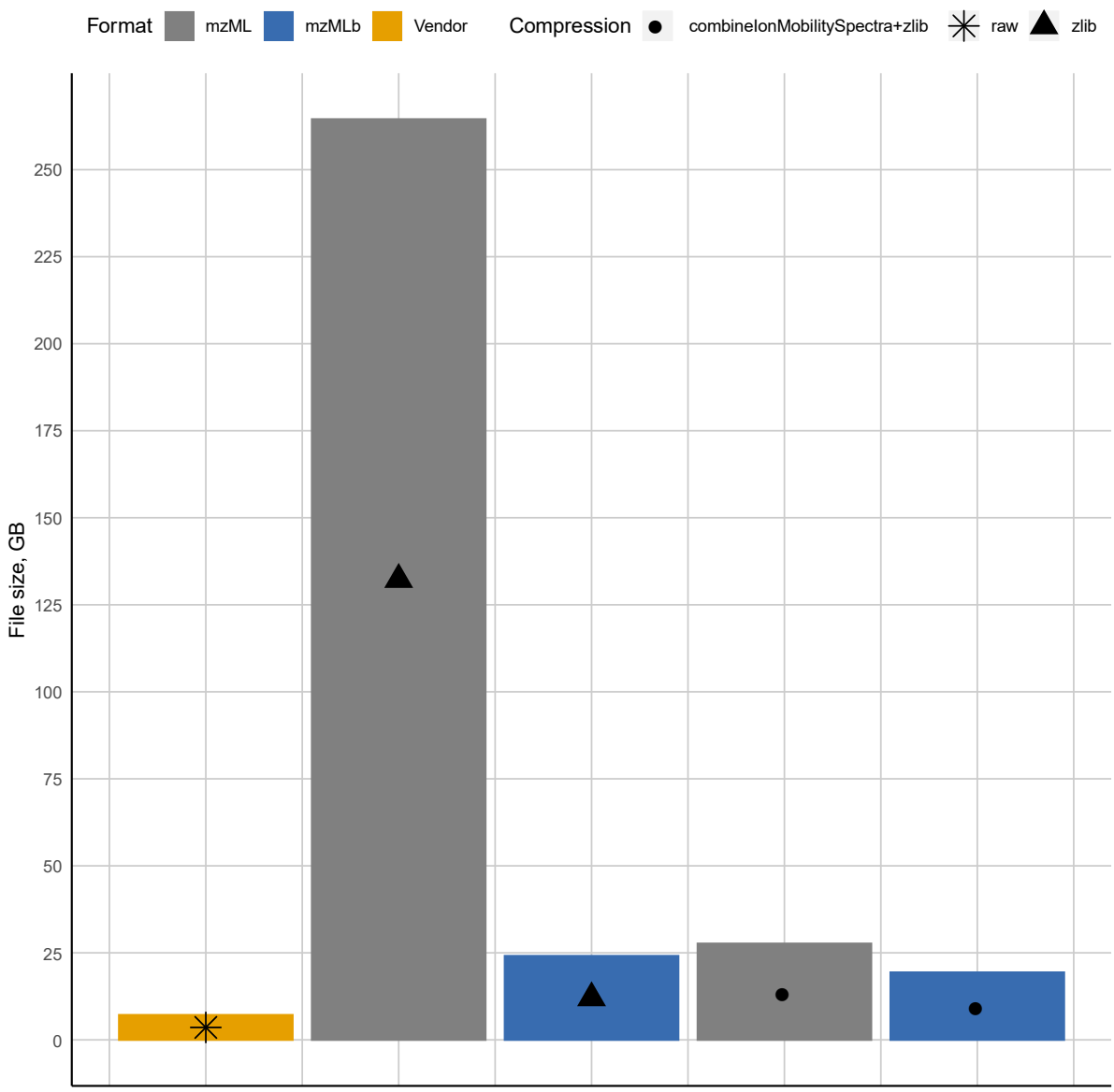

Figure S2. Bruker diaPASEF ion mobility dataset converted to both mzML and mzMLb with zlib compression enabled for both cases. Results are shown with and without the ProteoWizard switch “combineIonMobilitySpectra”, which is designed to allow more efficient storage of ion mobility data in mzML as per a forthcoming PSI recommendation. Note that for this data type, the vendor format is considerably more efficient than either open format.

The mzMLb format

An mzMLb dataset is a HDF5 file which must include in its root a HDF5 dataset ***mzML*** with fixed length string attribute ***version***. The currently supported version string is: “mzMLb 1.0”. The mzML XML document is stored in the ***mzML*** dataset, which is a 1D character array, with two modifications:

(1) HDF5 binary indexes replace the <*indexedmzML*> wrapper schema. Here:

- HDF5 datasets ***mzML_spectrumIndex*** and ***mzML_chromatogramIndex*** replace the respective <*indexedmzML*> <*index*> blocks. Each is a 1D array of 64bit integers replicating the set of <*offset*> file pointer offsets - except note that there is an extra offset at the end of each array representing one past the end position of the last spectrum/chromatogram.
- HDF5 datasets ***mzML_spectrumIndex_idRef*** and ***mzML_chromatogramIndex_idRef***, 1D character arrays, then replicate the idRef attributes of <*offset*> as null-terminated strings concatenated together.
- Similarly and optionally, *spotID* attributes can be stored in HDF5 1D character array datasets ***mzML_spectrumIndex_spotID*** and ***mzML_chromatogramIndex_spotID***, while *scanTime* attributes can be stored in HDF5 1D floating point array dataset ***mzML_spectrumIndex_scanTime***.
- (2) The mzML base64 encoded binary data is removed from the <mzML> and moved into one or more native binary HDF5 datasets. Floating point binary data (*i.e.* all non-Numpress compressed <*BinaryDataArray*>) is stored as one or more HDF5 1D floating point arrays, while Numpress data can be stored as a non-base64 encoded bytestream with HDF5 data type ***OPAQUE***.

As in imzML, the mzML is modified slightly to specify this linkage to external data; the resulting XML is still valid mzML. Here, any <*binary*> blocks within the <*binaryDataArray*> blocks are removed, and the *encodedLength* attribute is set to "0". To link to the native HDF5 binary data, within the <*binaryDataArray*> three <*cvParam*> tags need to be given, specifying the external dataset name, offset to the start of the relevant data within the dataset, and the length of the relevant data. These three tags enable flexibility over the nature and number of HDF5 datasets used to store the binary data (*e.g.* separate datasets can be used to store different datatypes; multiple spectra can be stored in the same dataset for improved chunking and compression).

ProteoWizard mzMLb msconvert arguments

In order to convert input data into the mzMLb format using **msconvert**; the following new arguments have been introduced that allow you to alter the default parameters of converting files to mzMLb when using **“--mzMLb**” switch.

--mzTruncation=[0-] --intenTruncation=[0-]

Perform lossy compression by removing the last n bits of mantissa from floating point data before storage. The default is 0 (no removal). Set to -1 to truncate to integers.

--mzDelta --intenDelta --mzLinear --intenLinear

Store mz/rt or intensity values after delta or linear prediction. Predictive encoding of mz/rt values may lead to moderate improvements in gzip compression, or further improvements after floating point precision loss.

--mzMLbChunkSize=[4096-]

Defines the chunk size to use for the mzML and all binary HDF5 datasets, in bytes. A smaller amount improves random access speed at the detriment of compression efficiency.

--mzMLbCompressionLevel=[0-9]

Define to use either no compression (0) or GZIP compression strength 1 to 9. Compression is applied to the mzML and all binary HDF5 datasets. Specifying **--zlib** or **-z** instead will use the default compression strength of 4. If no compression is specified, the default chunk size is 1024 KB. If compression is specified, the defaults are chunk size 1024 KB, mzLinear on, mzTruncation 19 and intenTruncation 7 (as described in the main manuscript).

Data and implementation availability

The mass spectrometry datasets used during the analysis of mzMLb in this study have been deposited at Zenodo.org.

```
https://doi.org/10.5281/zenodo.3951164
```

All the results in this paper were created using v0.6 of our reference ProteoWizard mzMLb implementation available at https://github.com/biospi/pwiz/releases/tag/v0.6

Examples:

Below is an example of converting a vendor raw (or any file that ProteoWizard can read) into mzMLb with mzLinear, mz truncation = 19, intensity truncation = 13 and a compressed chunking size of 1MB.

```
msconvert <Input file> –mzMLb --mzMLbCompressionLevel=4 --mzLinear --mzTruncation=19 --intenTruncation=13 --mzMLbChunkSize=10485760 --outfile <Name of Converted file>
```

Converting a vendor file using the Numpress scheme would entail the following.

```
msconvert <Input file> --mzML --zlib --mz64 --inten32 -n --outfile <Name of Converted file>
```

Converting a vendor file using the Numpress scheme with mzMLb.

```
msconvert <Input file> –mzMLb -n --outfile <Name of Converted file>
```

Converting a vendor file to lossless mzMLb with zLib:

```
msconvert <Input file> –mzMLb -z --outfile <Name of Converted file>
```