Numerical compression schemes for proteomics mass spectrometry data.

Teleman, Johan; Dowsey, Andrew W; Gonzalez-Galarza, Faviel F; Perkins, Simon; Pratt, Brian; Rost, Hannes; Malmstrom, Lars; Malmström, Johan; Jones, Andrew R; Deutsch, Eric W; Levander, Fredrik

Published in:
Molecular & Cellular Proteomics

DOI:
10.1074/mcp.O114.037879

2014

Numerical compression schemes for proteomics mass spectrometry data

Johan Teleman[1], Andrew W. Dowsey[2,3], Faviel F. Gonzalez-Galarza[4], Simon Perkins[4], Brian Pratt[5], Hannes L. Röst[6], Lars Malmström[6] , Johan Malmström[7], Andrew R. Jones[4], Eric W. Deutsch[8] and Fredrik Levander[1,9*]*

1 Department of Immunotechnology, Lund University, Medicon Village building 406, 223 81 Lund Sweden

2 Institute of Human Development, Faculty of Medical and Human Sciences, University of Manchester, United Kingdom

3 Centre for Advanced Discovery and Experimental Therapeutics (CADET), University of Manchester and Central Manchester University Hospitals NHS Foundation Trust, Manchester Academic Health Sciences Centre, Oxford Road, Manchester M13 9WL, United Kingdom

4 Institute of Integrative Biology, University of Liverpool, Liverpool, L69 7ZB, United Kingdom

5 Department of Genome Sciences, University of Washington School of Medicine, Seattle, Washington, 98195, USA

6 Department of Biology, Institute of Molecular Systems Biology, Eidgenössische Technische Hochschule Zürich, Wolfgang-Pauli Strasse 16, 8093 Zurich, Switzerland

7 Department of Clinical Sciences, Faculty of Medicine, Lund University, SE-221 84 Lund, Sweden

8 Institute for Systems Biology, 401 Terry Avenue North, Seattle, Washington 98109, USA

9 Bioinformatics Infrastructure for Life Sciences, Lund University, Sweden

* Equal contribution.

RUNNING TITLE: Numerical compression of MS data

ABBREVIATIONS

XML - Extensible Markup Language

DDA - Data-dependent Acquisition

SRM - Selected Reaction Monitoring

DIA - Data-Independent Acquisition

numLin - Numpress Linear Prediction

numSlof - Numpress Short Logged

numPic - Numpress Positive Integer Count

numSafe - Numpress Linear Prediction Transformation (lossless)

numAll - combination of numLin and numSlof compression

mz5zlib - mz5 with zlib compression

SSD - Solid-State Drive

MGF - Mascot Generic Format

SUMMARY

The open XML format mzML, used for representation of mass spectrometry (MS) data, is pivotal for the development of platform-independent MS analysis software. Although conversion from vendor formats to mzML must take place on a platform on which the vendor libraries are available (i.e. Windows), once mzML files have been generated, they can be used on any platform. However, the mzML format has turned out to be less efficient than vendor formats. In many cases, the naïve mzML representation is 4-fold or even up to 18-fold larger compared to the original vendor file. In disk I/O limited setups, a larger data file also leads to longer processing times, which is a problem given the data production rates of modern mass spectrometers. In an attempt to reduce this problem, we here present a family of numerical compression algorithms called MS-Numpress, intended for efficient compression of MS data. To facilitate ease of adoption, the algorithms target the binary data in the mzML standard, and support in main proteomics tools is already available. Using a test set of 10 representative MS data files we demonstrate typical file size decreases of 90% when combined with traditional compression, as well as read time decreases of up to 50%. It is envisaged that these improvements will be beneficial for data handling within the MS community.

INTRODUCTION

Open XML formats for representation of MS data have been developed by the proteomics community to facilitate exchange and vendor neutral analysis of mass spectrometry data. Initially two formats, mzXML (1) and mzData (http://psidev.info/), existed in parallel, until these formats were merged into the single standard format mzML (2). The mzML format has been adopted widely by the proteomics community and is supported by many data processing tools. However, although successfully used in many pipelines, the mzML format has not reached its full usage potential, mainly due to large file sizes in comparison to the raw vendor formats. The file size problem has become more marked with the introduction of recent high-resolution high-frequency mass spectrometers. As an example, a raw data file from a data-independent acquisition experiment using an AB SCIEX TripleTOF resulted in a vendor format data file of 2.5 GB. Conversion of this file to standard mzML resulted in a 46.7 GB file, with a conversion time of about 12 min on a desktop computer (later called high-end I, Fig. 1A) dedicated to the conversion process. If the file is compressed using gzip to lower the storage footprint, the size drops to 21.6 GB, but the conversion now takes 2 hours instead. This (extreme) example pinpoints the need for increased efficiency in the standardized representation of MS data.

Across the community, there is little previous work done on compression of MS data. In two technical papers (3, 4), Miguel *et. al.* describe lossless and near-lossless compression methods for QTOF data, achieving compression factors above 10. No measurements are provided on the compression time however, and the algorithms are benchmarked on a very small set of data files. Blanckenburg *et. al.* describe a lossy compression technique for Fourier transform ion cyclotron resonance data (5), where known non-metabolite data points are discarded. Outside the MS community, potential benefits could come from recent work in the numerical computation field (6, 7), were many data types are similar to MS data in terms of precision and smoothness. Nevertheless, perhaps the most relevant recent advance is the emergence of the mz5 format (8), which yields performance increases via a binary representation and optimized libraries, as well as some regular

data compression. While this format, based on the open HDF5 standard (The HDF Group, Champaign, IL), is an efficient representation of mzML files, it suffers from the fact that the files are not readable without native libraries or specialized software, which, to some extent, has hampered its uptake. Also, while mz5 can be "lossless", default compression implies removal of zero intensity scans, which means the original data cannot be reconstructed, and some algorithms require zero intensity scans for correct functioning.

The standard XML representation used in mzML can be easily viewed as text on any operating system, and it is relatively easy to write a parser in any programming language. We thus sought to overcome the mzML efficiency shortcomings by introducing better compression of the binary data found in mzML files while still leaving the metadata in XML format, and propose such an extension to the format here. Furthermore, we envisage that decompression of this binary data should be easy to incorporate into software tools via permissively licensed stand-alone source code files for C++ and Java, which do not require any external dependencies. We here also exemplify the facility of usage by implementing support in several popular tools for proteomics data analysis.

EXPERIMENTAL PROCEDURES AND RESULTS

To efficiently compress the three main types of binary data present in mzML files: i) mass to charge ratios, ii) ion counts and iii) retentions times, we have developed three new near-lossless compression algorithms, while ensuring for each data type that precision losses are well below the precision of the most advanced mass spectrometers of today. The Numpress Linear Prediction Compression algorithm (hereafter called numLin, relative error < 2e-9) takes advantage of the linearly increasing values in m/z and retention time data, and is optimized for high-resolution m/z data. Ion count data does not linearly increase but requires less stored precision because of the lower instrument precision, and Numpress Short Logged Float (numSlof, relative error < 2e-4) is optimized for this data type. We also developed a second ion count compression (Numpress Positive

Integer Count, numPic) and a lossless transformation (Numpress Linear Prediction Transformation, numSafe), which are not used further here, but are presented in the supplementary materials. Although the least significant of the 16 double-precision decimals are lost in the first conversion to the compressed format for all the algorithms, compression and decompression after this does not incur further losses. To maximize speed, the algorithms are highly local in memory and only need a single traversal of the data. For a complete description of the algorithms we refer to Supplemental Methods, and to the reference implementations in Java and C++, found at https://github.com/ms-numpress/ms-numpress under the Apache 2.0 license.

To compare MS-Numpress to current alternatives for storing mzML data, we extensively evaluated size, write time and read time of available compression schemes on a varied set of data files using different computers (Fig. 1). For this we constructed a test set of 10 MS data files from different vendors, instruments, and experiment types (Fig. 1D and Suppl. Table 1). The test set files included data-dependent acquisition (DDA), selected reaction monitoring (SRM) and data-independent (DIA/SWATH) acquisition modes, and both simple and complex samples, giving a heterogeneous set of distributions of MS1 and MS2 spectrum data and chromatogram binary data arrays of different lengths (Suppl. Fig. 1). These files were converted to mzML, imzML (9) and mz5 (8), both without compression, using zlib compression, and using gzip of the entire file. Files were also compressed in multiple different setups using MS-Numpress compressions, resulting in a total of 18 tested compression schemes (Fig 1C and Suppl. Table 2). To avoid clutter, minor results are left out here, and readers interested in imzML-data or individual Numpress results are referred to the supplementary material. The different compression schemes were compared based on file size, read time and write time (Fig. 1B). Benchmarking was performed on four dedicated desktop computers of varying capacity (Fig. 1A), using a custom msconvert (10) build, and timed using a script written in Python. Write time was measured as the total time for an msconvert conversion from the vendor raw format. Since this includes the vendor read time it gives a constant offset, but this constant is in general small compared to the write time. For read benchmarking a custom program was made, that

reads files using the ProteoWizard (10) API. To ensure that all data is read, this program explicitly reads all binary values in the spectra and chromatograms found in the file. Test files, results, and program binaries are available at the Swestore repository (http://webdav.swegrid.se/snic/bils/lu_proteomics/pub).

The use of near-lossless compression introduces the question of whether one can be sure that no analytically relevant data is lost. We measured the relative errors for the compressed versions of all the files in the test set (Suppl. Table 3-6), and found relative errors to be smaller than 2e-9 (0.002 ppm) for numLin compressed m/z data and smaller than 2e-4 (0.02 %) for numSlof compressed ion counts. To validate that the small errors introduced by Numpress compression do not have adverse effect on common proteomics analyses, we converted two Orbitrap DDA LC-MS/MS mzML files to compressed (combination of numLin and numSlof) versions and back to uncompressed mzML again, and compared analysis results obtained from the doubly converted files to those obtained using the original files. MS/MS identification using Mascot after extraction of MGF (Mascot Generic Format) peak lists yielded identical lists of identified peptides at a 1% peptide to spectrum match (PSM) false discovery rate (FDR, Supplementary data). Extraction of features from the MS1 data using msInspect (11) yielded the same lists of features, with differences only in the least significant decimals of some reported m/z values and intensities (Supplementary Data). When comparing lists of integrated precursor intensities for the peptides that were identified using MS/MS, the lists contained the same entries with a 0.004% maximum observed relative intensity difference introduced by the double conversion, confirming that the Numpress compression schemes could be safely used for proteomics analyses**.**

We found that to achieve minimal file size, a combination of numLin for m/z or retention time data and numSlof for ion count data (numAll), with subsequent gzipping of the entire file, was the most optimal in terms of file size. This yielded an average file size reduction of 87% compared to standard mzML across all 10 test set files (Fig. 2A), with 138% longer write times (Fig. 2B) but 21% shorter read

times (Fig. 2C) on average across all tested computers and files (Suppl. Table 7). This format is also half the size of the binary mz5 with zlib compression (mz5zlib), and also smaller than all vendor formats except for AB SCIEX's .wiff files (Suppl. Fig. 2, Suppl. Table 7). In our read speed tests, the text based mzML formats cannot quite compete with the binary mz5, although the difference is small (20%) in the largest files (Fig. 2C). The effects of minimizing disk I/O through compression are the most visible in the largest files on the lower performance computers (Suppl. Fig. 3 and 4), where the numAll alternatives catch up to mz5zlib. Overall, read times ranged over 3 orders of magnitude (Suppl. Fig. 5), and write times over 4 orders of magnitude (Suppl. Fig. 6).

Two of the test computers were equipped with SSD hard drives, which are fast enough to open up the disk I/O bottleneck, and reveal the next bottleneck: processor speed. On these machines the expensiveness of gzipping becomes apparent, with cost increasing with file size (Fig. 2B), and the largest gains from the fast disk I/O are seen for the processor-light schemes mzML, mz5zlib and numAll (Suppl. Fig. 7 and 8). Zlib compression of individual data arrays also shows minor slow-down of the write operation, whereas the numAll scheme does not affect write times at all (Fig. 2C).

Even though numAll decreased read times by on average 36% compared to standard mzML (Suppl. Table 7), mz5zlib further decreased read times by 25% (total 61% from mzML), and this might have a number of reasons. Some hypotheses for explaining this are 1) the aggressive file caching of mz5 provides block read benefits, 2) large amount of string-string comparisons while building the mzML object model is costly, or 3) Base64 decoding is costly. While 3) is not solvable while keeping a one-file, text-based format, both 1) and 2) could be improved in optimized reader implementations.

As the MS-Numpress compression techniques showed high degrees of compression, which was our primary goal, we set out to implement the technique as part of several proteomics pipelines in order to ensure easy adoption by the proteomics community. The initial implementation in ProteoWizard (10) enables conversion to the format from all major mass spectrometer raw data formats, and

provides read access to tools that use the ProteoWizard API for reading files, for example MyriMatch (12) and Skyline (13).

Compression should be especially effective for workflows that use high-resolution profile data for quantitation, because of the large data files this implies, and we therefore implemented support for reading of mzML files with numpress compressed binaries in OpenMS (14), msInspect (11) and the Proteios Software Environment (15, 16). The implementation in OpenMS also implies that the complete MS-Numpress compression and decompression algorithms are directly available in the Python scripting language due to the recent Python-wrapping of the complete OpenMS library (17).

The jmzML (18) library has also been extended with MS-Numpress read and write support for numLin, numSlof and numPic. The jmzML library is embedded in numerous Java-based mass spectrometry software solutions, including PRIDE Converter (19) and Proteosuite, an open source framework for the analysis of quantitative proteomics data (http://www.proteosuite.org/). Such solutions will now implicitly support MS-Numpress compressed data when utilising the latest jmzML library.

Support for mzML with MS-Numpress compression was implemented in the tools of the Trans-Proteomics Pipeline (20, 21). Reading was also implemented in the X!Tandem search engine (22). Finally, we implemented support for MS-Numpress in the Anubis (23) tool for SRM.

The main advantages of using the new MS-Numpress-compressed mzML format is probably seen where small file sizes are of highest importance, for example in data sharing over the Internet. Minimal file size is also of utmost importance in distributed or cloud computing, for example shown by Dowsey *et al.* for two-dimensional gel electrophoresis alignment (24). Because of the file size importance, simple peak lists formats are currently still used extensively for MS/MS database searches, and the more complete file representations provided by mzML have mainly been used for data sharing and quantitative workflows. However, with the increased number of MS/MS spectra acquired with modern mass spectrometers it is envisaged that compressed data formats could

become more widely used also for MS/MS database searches. As an example, we downloaded a raw data file from a single-shot LC-MS/MS analysis of a yeast lysate, performed on an Orbitrap Fusion acquired for a recent publication from the Coon lab (25). Conversion of the file to Mascot Generic Format (MGF, http://www.matrixscience.com) using ProteoWizard and no filtering yielded a file size of 1508 MB (715 MB gzipped), with only the centroid MS/MS data retained. ProteoWizard conversion of the original file using identical parameters to mzML with numAll compression resulted in a 944 MB (409 MB gzipped) file, while still retaining the MS1. The file size shrinks to 821 MB (343 MB gzipped) if only the MS/MS data is retained in the mzML file. Interestingly, an MS/MS search in X!Tandem resulted in slightly more peptide identifications at a 1% PSM FDR using the mzML file than using the MGF with a precursor mass tolerance of 7 ppm (Supplementary Data), probably due to a higher precision in the precursor mass in the mzML file, showing that there is no apparent drawback in using the compressed format for MS/MS database searches.

Our results demonstrate the power of some very simple techniques to improve the mzML format with respect to disk space and handling time. There are undoubtedly other algorithms that could further improve on the degree of compression or handling times, but for standard formats we believe it is crucial to provide simple and robust solutions, to minimize both the cost of implementation in tools, and the risk of mistakes in the algorithm. We further provide implemented support for the new algorithms in several tools, and thus give immediate access to the proteomics community. MS-Numpress will also be evaluated through the Proteomics Standards Initiative (PSI) process for formal inclusion in the next mzML release. We hope that this work may also stimulate additional data compression algorithm ideas, which, in the end, will lead to an amendment to the mzML standard to improve data handling for all mzML users.

ACKNOWLEDGEMENTS

We are very grateful to Jari Häkkinen for reviewing the C++ MS-Numpress library and to Giorgio Arrigoni for providing the Agilent QTOF data file. J.M. and J.T. were supported by the Swedish Research Council (projects 2008:3356 and 621-2012-3559), the Swedish Foundation for Strategic Research (grant FFL4), the Crafoord Foundation (grant 20100892), the Wallenberg Academy Fellow KAW (2012.0178) and European research council starting grant (ERC-2012-StG-309831). A.W.D.'s contribution was supported by UK Biotechnology and Biological Sciences Research Council (BBSRC) grant BB/K016733/1, and facilitated by the Manchester Biomedical Research Centre. F.G.G.G, S.P. and A.R.J. gratefully acknowledge support for this work from BBSRC (BB/l00095X/1, BB/K01997X/1). E.W.D. was supported by the National Institute of General Medical Sciences grant No. R01 GM087221, the National Science Foundation MRI grant No. 0923536, the National Human Genome Research Institute grant No. RC2 HG005805, and EU FP7 grant 'ProteomeXchange' (grant number 260558). F.L. was supported by the Swedish Foundation for Strategic Research (RBb08-0006), the Swedish Research Council (BILS, project 829-2009-6257) and the Mistra Biotech program.

REFERENCES

1. Pedrioli, P. G. A., Eng, J. K., Hubley, R., Vogelzang, M., Deutsch, E. W., Raught, B., Pratt, B., Nilsson, E., Angeletti, R. H., Apweiler, R., Cheung, K., Costello, C. E., Hermjakob, H., Huang, S., Julian, R. K., Kapp, E., McComb, M. E., Oliver, S. G., Omenn, G., Paton, N. W., Simpson, R., Smith, R., Taylor, C. F., Zhu, W., and Aebersold, R. (2004) A common open representation of mass spectrometry data and its application to proteomics research. *Nat. Biotechnol.* 22, 1459–1466

2. Martens, L., Chambers, M., Sturm, M., Kessner, D., Levander, F., Shofstahl, J., Tang, W. H., Römpp, A., Neumann, S., Pizarro, A. D., Montecchi-Palazzi, L., Tasman, N., Coleman, M., Reisinger, F., Souda, P., Hermjakob, H., Binz, P.-A., and Deutsch, E. W. (2011) mzML--a community standard for mass spectrometry data. *Mol. Cell. Proteomics MCP* 10, R110.000133

3. Miguel, A. C., Keane, J. F., Whiteaker, J., Zhang, H., and Paulovich, A. G. (2006) Compression of LC/MS Proteomic Data. *Proc. 19th IEEE Symp. Comput.-Based Med. Syst. CBMS06*,

4. Miguel, A. C., Kearney-Fischer, M., Keane, J. F., Whiteaker, J., Feng, L.-C., and Paulovich, A. G. (2007) Near-lossless compression of mass spectra for proteomics. *Acoust. Speech Signal Process. 2007 ICASSP 2007 IEEE Int. Conf.* 1, I–369 – I–372

5. Blanckenburg, B., Burgt, Y. E. M., Deelder, A. M., and Palmblad, M. (2010) "Lossless" compression of high resolution mass spectra of small molecules. *Metabolomics* 6, 335–340

6. Engelson, V., Fritzson, D., and Fritzson, P. (2000) Lossless Compression of High-volume Numerical Data from Simulations. *Data Compression Conf.*, 574–586

7. Ratanaworabhan, P., Ke, J., and Burtscher, M. (2006) Fast lossless compression of scientific floating-point data. *Data Compression Conf. 2006 DCC 2006 Proc.* 1, 133 – 142

8. Wilhelm, M., Kirchner, M., Steen, J. A. J., and Steen, H. (2012) mz5: space- and time-efficient storage of mass spectrometry data sets. *Mol. Cell. Proteomics MCP* 11, O111.011379

9. Römpp, A., Schramm, T., Hester, A., Klinkert, I., Both, J.-P., Heeren, R. M. A., Stöckli, M., and Spengler, B. (2011) imzML: Imaging Mass Spectrometry Markup Language: A common data format for mass spectrometry imaging. *Methods Mol. Biol. Clifton NJ* 696, 205–224

10. Chambers, M. C., Maclean, B., Burke, R., Amodei, D., Ruderman, D. L., Neumann, S., Gatto, L., Fischer, B., Pratt, B., Egertson, J., Hoff, K., Kessner, D., Tasman, N., Shulman, N., Frewen, B., Baker, T. A., Brusniak, M.-Y., Paulse, C., Creasy, D., Flashner, L., Kani, K., Moulding, C., Seymour, S. L., Nuwaysir, L. M., Lefebvre, B., Kuhlmann, F., Roark, J., Rainer, P., Detlev, S., Hemenway, T., Huhmer, A., Langridge, J., Connolly, B., Chadick, T., Holly, K., Eckels, J., Deutsch, E. W., Moritz, R. L., Katz, J. E., Agus, D. B., MacCoss, M., Tabb, D. L., and Mallick, P. (2012) A cross-platform toolkit for mass spectrometry and proteomics. *Nat. Biotechnol.* 30, 918–920

11. Bellew, M., Coram, M., Fitzgibbon, M., Igra, M., Randolph, T., Wang, P., May, D., Eng, J., Fang, R., Lin, C., Chen, J., Goodlett, D., Whiteaker, J., Paulovich, A., and McIntosh, M. (2006) A suite of algorithms for the comprehensive analysis of complex protein mixtures using high-resolution LC-MS. *Bioinforma. Oxf. Engl.* 22, 1902–1909

12. Tabb, D. L., Fernando, C. G., and Chambers, M. C. (2007) MyriMatch: highly accurate tandem mass spectral peptide identification by multivariate hypergeometric analysis. *J. Proteome Res.* 6, 654–661

13. MacLean, B., Tomazela, D. M., Shulman, N., Chambers, M., Finney, G. L., Frewen, B., Kern, R., Tabb, D. L., Liebler, D. C., and MacCoss, M. J. (2010) Skyline: an open source document editor for creating and analyzing targeted proteomics experiments. *Bioinforma. Oxf. Engl.* 26, 966–968

14. Kohlbacher, O., Reinert, K., Gröpl, C., Lange, E., Pfeifer, N., Schulz-Trieglaff, O., and Sturm, M. (2007) TOPP--the OpenMS proteomics pipeline. *Bioinforma. Oxf. Engl.* 23, e191–197

15. Häkkinen, J., Vincic, G., Månsson, O., Wårell, K., and Levander, F. (2009) The proteios software environment: an extensible multiuser platform for management and analysis of proteomics data. *J. Proteome Res.* 8, 3037–3043

16. Sandin, M., Ali, A., Hansson, K., Månsson, O., Andreasson, E., Resjö, S., and Levander, F. (2013) An adaptive alignment algorithm for quality-controlled label-free LC-MS. *Mol. Cell. Proteomics MCP* 12, 1407–1420

17. Röst, H. L., Schmitt, U., Aebersold, R., and Malmström, L. (2014) pyOpenMS: A Python-based interface to the OpenMS mass-spectrometry algorithm library. *Proteomics*, 14, 74-77

18. Côté, R. G., Reisinger, F., and Martens, L. (2010) jmzML, an open-source Java API for mzML, the PSI standard for MS data. *Proteomics* 10, 1332–1335

19. Côté, R. G., Griss, J., Dianes, J. A., Wang, R., Wright, J. C., van den Toorn, H. W. P., van Breukelen, B., Heck, A. J. R., Hulstaert, N., Martens, L., Reisinger, F., Csordas, A., Ovelleiro, D.,

Perez-Rivevol, Y., Barsnes, H., Hermjakob, H., and Vizcaíno, J. A. (2012) The PRoteomics IDEntification (PRIDE) Converter 2 framework: an improved suite of tools to facilitate data submission to the PRIDE database and the ProteomeXchange consortium. *Mol. Cell. Proteomics MCP* 11, 1682–1689

20. Keller, A., Eng, J., Zhang, N., Li, X., and Aebersold, R. (2005) A uniform proteomics MS/MS analysis platform utilizing open XML file formats. *Mol. Syst. Biol.* 1, 2005.0017

21. Deutsch, E. W., Mendoza, L., Shteynberg, D., Farrah, T., Lam, H., Tasman, N., Sun, Z., Nilsson, E., Pratt, B., Prazen, B., Eng, J. K., Martin, D. B., Nesvizhskii, A. I., and Aebersold, R. (2010) A guided tour of the Trans-Proteomic Pipeline. *Proteomics* 10, 1150–1159

22. Craig, R., and Beavis, R. C. (2004) TANDEM: matching proteins with tandem mass spectra. *Bioinforma. Oxf. Engl.* 20, 1466–1467

23. Teleman, J., Karlsson, C., Waldemarson, S., Hansson, K., James, P., Malmström, J., and Levander, F. (2012) Automated selected reaction monitoring software for accurate label-free protein quantification. *J. Proteome Res.* 11, 3766–3773

24. Dowsey, A. W., Dunn, M. J., and Yang, G.-Z. (2004) ProteomeGRID: towards a high-throughput proteomics pipeline through opportunistic cluster image computing for two-dimensional gel electrophoresis. *Proteomics* 4, 3800–3812

25. Hebert, A. S., Richards, A. L., Bailey, D. J., Ulbrich, A., Coughlin, E. E., Westphall, M. S., and Coon, J. J. (2013) The One Hour Yeast Proteome. *Mol. Cell. Proteomics MCP*,

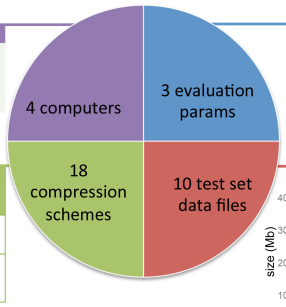
FIGURE LEGENDS

Figure 1: Overview of the compression scheme evaluation experiment. A) Evaluation was performed on four desktop computers of varying capacity, the most notable parameter being the use of solid state drive (SSD) hard drives compared to traditional mechanical hard drives. B) File size, write time and read time were used to evaluate compression. C) Main results include two new numerical compression methods optimized for the three different MS data types m/z, ion count and retention time (RT), with relative errors far below instrument precision. D) File sizes of vendor and mzML versions of the 10 used MS data files. The inset shows a magnification of the 7 smallest files.

Figure 2: File size, read and write time compared to standard mzML. A) Standard box plot of file size relative to mzML for 7 data formats. B-C) log_2 write time (B) and read time (C) subtracted by log_2 mzML write and read time, respectively, for the 10 test files and 7 data formats. Test files are sorted in descending mzML size, and dots represent measurements on individual data files on one of two SSD-hard drive equipped computers. Writing of vendor formats could not be tested with this setup and is thus missing in B. Missing read times are due to errors in execution (mzML.gz for file 5 and mz5zlib for file 6) as discussed in the supplementary material, where also global statistics for all compression schemes (Suppl. Table 7) and absolute timing figures (Suppl. Figure 5 and 6) are provided.

A

id	processor	RAM (GB)	HDD	OS
high-end I	Intel i7 3.4 Ghz	16	SSD	Win 7
high-end II	Intel Xeon 3.4Ghz	24	SSD	Win 7
low-end I	Intel Quad 2.8 Ghz	8	mech.	Win XP
low-end II	Intel Quad 2.8 Ghz	8	mech.	Win XP



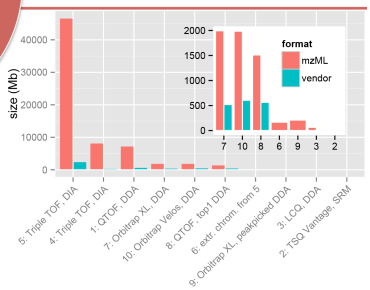
B

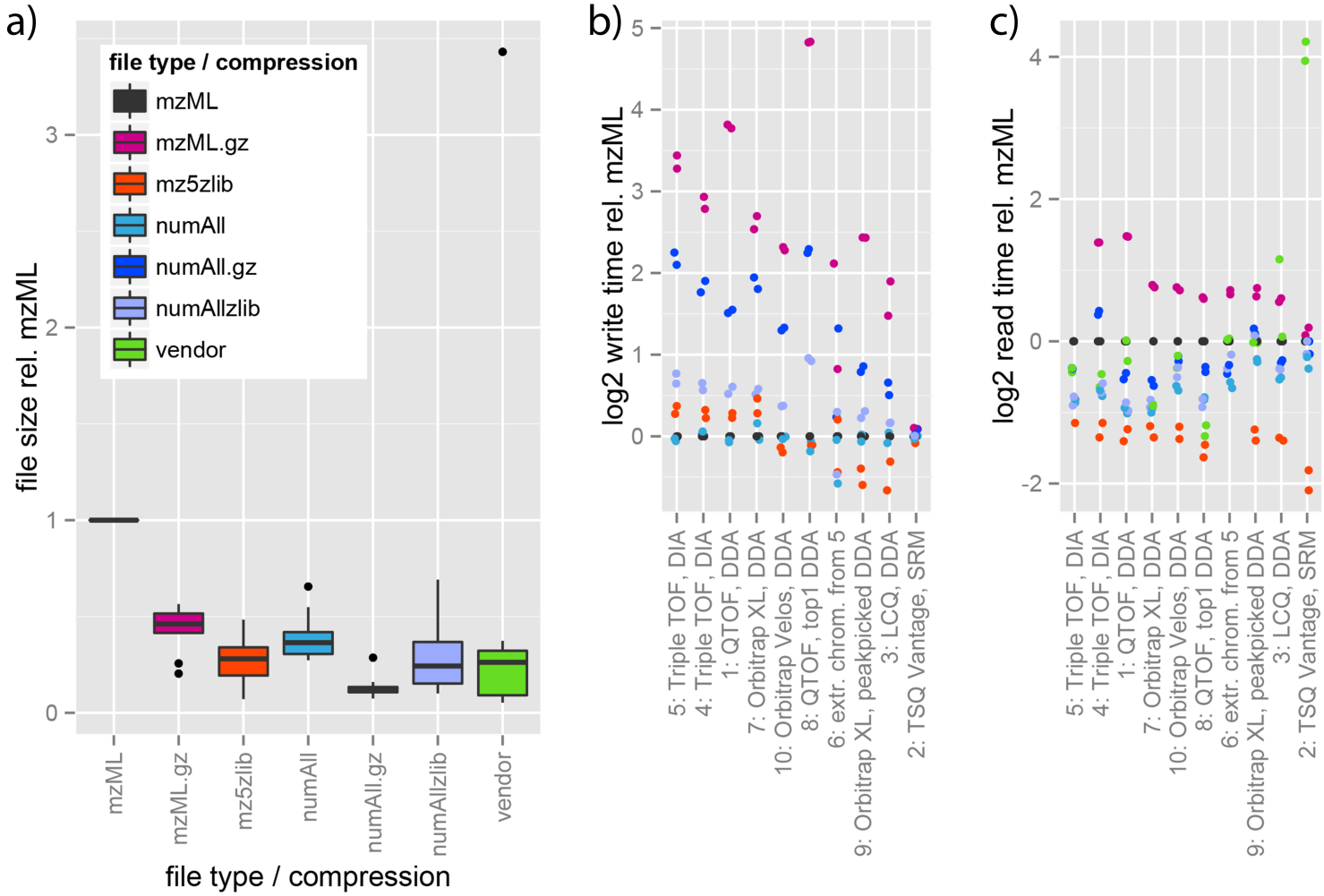
File size vs. mzML
Read time vs. mzML
Write time vs. mzML

C

MS-Numpress algorithm		Data type	Test set max rel. error
numAll	numLin	m/z, RT	2e-9
	numSlof	Ion count	2e-4

D




a)
file type / compression
mzML
mzML.gz
mz5zlib
numAll
numAll.gz
numAllzlib
vendor
file size rel. mzML
0
1
2
3
mzML
mzML.gz
mz5zlib
numAll
numAll.gz
numAllzlib
vendor
file type / compression
b)
log2 write time rel. mzML
0
1
2
3
4
5
5: Triple TOF, DIA
4: Triple TOF, DIA
1: QTOF, DDA
7: Orbitrap XL, DDA
10: Orbitrap Velos, DDA
8: QTOF, top1 DDA
6: extr. chrom. from 5
9: Orbitrap XL, peakpicked DDA
3: LCQ, DDA
2: TSQ Vantage, SRM
c)
log2 read time rel. mzML
-2
0
2
4
5: Triple TOF, DIA
4: Triple TOF, DIA
1: QTOF, DDA
7: Orbitrap XL, DDA
10: Orbitrap Velos, DDA
8: QTOF, top1 DDA
6: extr. chrom. from 5
9: Orbitrap XL, peakpicked DDA
3: LCQ, DDA
2: TSQ Vantage, SRM

Supplementary Material

Table of Contents

Algorithm descriptions

The library provides implementations of 4 different algorithms, 1 designed to compress first order smooth data like retention time or M/Z arrays, 1 designed to transform first order smooth data for more efficient zlib compression, and 2 for compressing non-smooth data with lower requirements on precision like ion count arrays.

Implementations and unit test are provided in C++ and java.

MS Numpress positive integer compression (numPic)

Intended for ion count data, this compression simply rounds values to the nearest integer, and stores

these integers in a truncated form which is effective for values close to zero.

MS Numpress short logged float compression (numSlof)

Also targeting ion count data, this compression takes the natural logarithm of values, multiplies by a scaling factor and rounds to the nearest integer. For typical ion count dynamic range these values fits into two byte integers, so only the two least significant bytes of the integer are stored.

The scaling factor can be chosen manually, but the library also contains a function for retrieving the optimal numSlof scaling factor for a given data array. In this case optimal refers to the greatest precision storable for this data in 2 bytes. Since the scaling factor is variable, it is stored as a regular double precision float first in the encoding, and automatically parsed during decoding.

MS Numpress linear prediction compression (numLin)

This compression uses a fixed point representation, achieved by multiplication by a scaling factor and rounding to the nearest integer. To exploit the frequent linearity of the data, linear prediction is then used in the following way.

The first two values are stored without compression as 4 byte integers. For each following value a linear prediction is made from the two previous values:

Xpred = (X(n) - X(n-1)) + X(n)
Xres = Xpred - X(n+1)

The residual Xres is then stored, using the same truncated integer representation as in Numpress Pic.

The scaling factor can be chosen manually, but the library also contains a function for retrieving the optimal numLin scaling factor for a given data array. Again, optimal here refers to the greatest precision for the fixed point byte size. Since the scaling factor is variable, it is stored as a regular double precision float first in the encoding, and automatically parsed during decoding.

MS Numpress safe linear prediction (numSafe)

This transformation uses the same linear prediction as numLin, but without the fixed point representation or integer truncation. This means that no compression is achieved and the resulting binary array will be exactly the same size as the input array. Note that even so some minimal degradation will occur due to the double operation rounding errors, but as sequential compression and decompression is hardly performed the transformation should still be practically lossless.

Truncated integer representation

This encoding works on a 4 byte integer, by truncating initial zeros or ones. If the initial (most significant) half byte is 0x0 or 0xf, the number of such halfbytes starting from the most significant is stored in a count halfbyte. This initial count is then followed by the rest of the int's halfbytes, in little-endian order. A count halfbyte c of

0 <= c <= 8 is interpreted as an initial c 0x0 halfbytes
9 <= c <= 15 is interpreted as an initial (c-8) 0xf halfbytes

Examples:

```
int     c       rest
0   =>  0x8
-1  =>  0xf     0xf
23  =>  0x6     0x7 0x1
```

Implementation notes

We recommend to simply embed the numpress library source files in your source when implementing numpress support in new tools. At the point of writing, implementations so far are all open source, which means there are many reference implementations, especially for C++. For the time being, we also recommend numpress writer implementations to produce mzML 1.1 compliant files, meaning amongst other things that only one compression per binary should be allowed (uncompressed, zlib, numPic, numSlof or numLin), and the 32/64-bit tag written out even though it's unnecessary with numpress compressions.

During this project we also implemented support for reading and writing imzML with ProteoWizard, in an attempt to investigate eventual gains with not having to encode all data in base64 binary data. While there is the obvious gain in file size from the reduced redundancy, we did not see any conclusive improvements in handling speed using our implementation. Nevertheless, we are in contact with the ProteoWizard team to eventually include this imzML-support in ProteoWizard. It should also be noted that imzML was simply used as a means for storing binary data in an external binary file, and we have not added any handling of imaging relevant information.

Tools and scripts used for testing

For testing we have extended ProteoWizards msconvert. Added abilities include supporting numpress compressions, allowing both numpress and zlib compression on the same binary data array, writing and reading imzML, as well as explicitly setting the write buffer size. At the time of writing, numpress support and double compression is already included in the official msconvert, and inclusion of other amendments is discussed. Access to exact binaries and scripts used for testing will be provided upon request.

Missing values

We were unable to achieve a few measurements. Because of their proprietary nature we cannot write custom vendor files. Reading the largest SWATH DIA file in the mzML.gz format crashed on all computers in a failure to allocate memory, even on the 24 GB machine. Reading of the extracted chromatograms (file 6) crashed for unknown reasons when trying to read the mz5 and mz5zlib formats. Unpromising and likely incorrect preliminary results for the imzML-based formats stopped completion of read and write timing for these formats.

Supplementary Table 1: MS data files

id	resolution	type	original size (MB)	vendor	peak picked	instrument
1	high	DDA	724	Agilent	no	QTOF
2	low	SRM	13.8	Thermo	no	TSQ Vantage
3	low	DDA	16.2	Thermo	no	LCQ
4	high	SWATH DIA	430[1]	ABI Sciex	no	Triple TOF
5	high	SWATH DIA	2500[1]	ABI Sciex	no	Triple TOF
6	high	Chrom from 5[2]	219.7	ABI Sciex	no	Triple TOF
7	high	DDA	520.8	Thermo	no	Orbitrap XL
8	high	MS+MS/MS	538	Waters	no	QTOF Ultima
9	high	DDA	202[3]	Thermo	yes	Orbitrap XL
10	high	DDA	577	Thermo	no	Orbitrap Velos

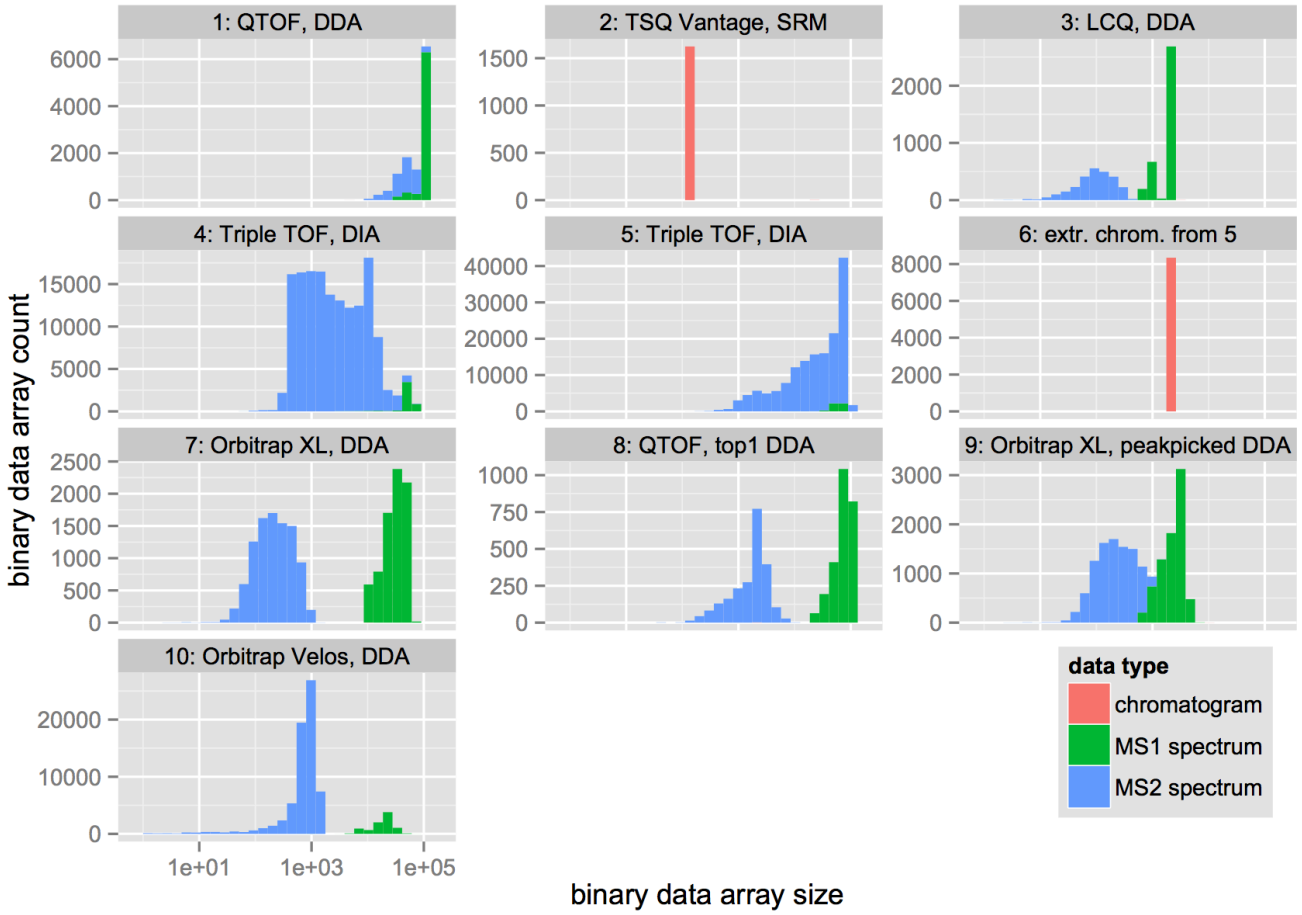
[1)] The large size difference comes from the very different samples, were 4) is an information sparse dilution of stable isotope peptides in water, and 5) is a information dense yeast extract (Suppl. Fig. 1).

[2)] Chromatograms were extracted using a custom extraction tool, by deconvolution using a top-hat filter of 10 ppm total width. This means peaks within 10 ppm of a known fragment m/z, in MS2 spectra of the swath corresponding to the known precursor m/z, were summed with weights decreasing linearly with distance from the exact fragment mass.

[3)] This is in mzML format since we cannot write custom Thermo raw files.

Supplementary Figure 1: Binary data array length distributions

Distributions of binary data array lengths for the 10 MS data files used for benchmarking. Depending on instrument speed, a varying amount of MS2 spectra can be seem in relation to the number of MS1 spectra. Also, the more complex MS1 spectra tend to give long binary data arrays compared to the simpler MS2 spectra, especially for high-resolution and/or high sampling frequency instruments. File 4 and 5 are acquired with the same method on the same instrument, but on hundreds of synthetic peptides in file 4, and a yeast lysate in file 5.



Supplementary Table 2: All tested compression schemes

Table showing the tested compression schemes

scheme	binary data type			
	m/z	ion count	retention time	whole file
mzML	-	-	-	-
mzML.gz	-	-	-	gzip
zlib	zlib	zlib	zlib	-
mz5	-	-	-	-
mz5zlib	zlib	zlib	zlib	-
numAll	numLin	numSlof	numLin	-
numAll.gz	numLin	numSlof	numLin	gzip
numAllzlib	numLin & zlib	numSlof & zlib	numLin & zlib	gzip
numLin	numLin	-	numLin	-
numPic	-	numPic	-	-
numSlof	-	numSlof	-	-
numLinZlib	numLin & zlib	-	-	-
numSafeZlib	numSafe & zlib	-	-	-
imzML	-	-	-	-
imzMLnumAll	numLin	numSlof	numLin	-
imzMLnumAllzlib	numLin & zlib	numSlof & zlib	numLin & zlib	-
imzMLzlib	zlib	zlib	zlib	-
vendor	-	-	-	-

Supplementary Table 3: Max relative error for numSlof on 10 test set files

file	negative	positive
06May2010_TestSuperHirn_BSA30fmol_01.d.ms1.clean.diff	-7.43E-005	7.59E-005
110620_fract_scxB05.raw.ms1.clean.diff	-0.000151466	0.000176515
110620_fract_scxB05.raw.ms2.clean.diff	-0.000127385	0.000134896
120224_006_SW.wiff.ms1.clean.diff	-6.63E-005	6.58E-005
120224_006_SW.wiff.ms2.clean.diff	-5.44E-005	5.41E-005
120302_006_SW.wiff.ms1.clean.diff	-6.73E-005	6.79E-005
120302_006_SW.wiff.ms2.clean.diff	-5.71E-005	5.71E-005
121213_PhosphoMRM_TiO2_discovery.RAW.ms1.clean.diff	-0.000160543	0.000149204
121213_PhosphoMRM_TiO2_discovery.RAW.ms2.clean.diff	-1.03E-005	1.81E-005
ADH_100126_mix.raw.ms1.clean.diff	-4.45E-005	4.32E-005
Velos_120905_09.raw.ms1.clean.diff	-0.000141388	9.60E-005
Velos_120905_09.raw.ms2.clean.diff	-6.37E-005	5.94E-005
peakPicked.121213_PhosphoMRM_TiO2_discovery.ms1.clean.diff	-9.86E-005	9.84E-005
peakPicked.121213_PhosphoMRM_TiO2_discovery.ms2.clean.diff	-1.03E-005	1.81E-005

Supplementary Table 4: Max relative error for numLin on 10 test set files

file	negative	positive
06May2010_TestSuperHirn_BSA30fmol_01.d.ms1.clean.diff	-2.33E-010	2.33E-010
110620_fract_scxB05.raw.ms1.clean.diff	-2.34E-010	2.35E-010
110620_fract_scxB05.raw.ms2.clean.diff	-2.67E-010	3.14E-010
120224_006_SW.wiff.ms1.clean.diff	-2.33E-010	2.32E-010
120224_006_SW.wiff.ms2.clean.diff	-1.61E-009	1.59E-009
120302_006_SW.wiff.ms1.clean.diff	-2.33E-010	2.30E-010
120302_006_SW.wiff.ms2.clean.diff	-1.57E-009	1.63E-009
121213_PhosphoMRM_TiO2_discovery.RAW.ms1.clean.diff	-2.33E-010	2.33E-010
121213_PhosphoMRM_TiO2_discovery.RAW.ms2.clean.diff	-1.22E-010	2.79E-010
ADH_100126_mix.raw.ms1.clean.diff	-2.33E-010	2.33E-010
Velos_120905_09.raw.ms1.clean.diff	-2.32E-010	2.32E-010
Velos_120905_09.raw.ms2.clean.diff	-2.39E-010	2.69E-010
peakPicked.121213_PhosphoMRM_TiO2_discovery.ms1.clean.diff	-2.33E-010	2.31E-010
peakPicked.121213_PhosphoMRM_TiO2_discovery.ms2.clean.diff	-1.22E-010	2.79E-010

Supplementary Table 5: Max relative error for numPic on 10 test set files

file	negative	positive
06May2010_TestSuperHirn_BSA30fmol_01.d.ms1.clean.diff	0	0
110620_fract_scxB05.raw.ms1.clean.diff	0	0
110620_fract_scxB05.raw.ms2.clean.diff	0	0
120224_006_SW.wiff.ms1.clean.diff	0	0
120224_006_SW.wiff.ms2.clean.diff	0	0
120302_006_SW.wiff.ms1.clean.diff	0	0
120302_006_SW.wiff.ms2.clean.diff	0	0
121213_PhosphoMRM_TiO2_discovery.RAW.ms1.clean.diff	-0.66481	0.390022
121213_PhosphoMRM_TiO2_discovery.RAW.ms2.clean.diff	-0.14251	0.0750765
ADH_100126_mix.raw.ms1.clean.diff	0	0
Velos_120905_09.raw.ms1.clean.diff	-0.52203	0.198162
Velos_120905_09.raw.ms2.clean.diff	-0.13293	0.114401
peakPicked.121213_PhosphoMRM_TiO2_discovery.ms1.clean.diff	-0.00113	0.0011207
peakPicked.121213_PhosphoMRM_TiO2_discovery.ms2.clean.diff	-0.14251	0.0750765

Supplementary Table 6: Max error for numPic on 10 test set files

file	negative	positive
06May2010_TestSuperHirn_BSA30fmol_01.d.ms1.clean.diff	0	0
110620_fract_scxB05.raw.ms1.clean.diff	0	0
110620_fract_scxB05.raw.ms2.clean.diff	0	0
120224_006_SW.wiff.ms1.clean.diff	0	0
120224_006_SW.wiff.ms2.clean.diff	0	0
120302_006_SW.wiff.ms1.clean.diff	0	0
120302_006_SW.wiff.ms2.clean.diff	0	0
121213_PhosphoMRM_TiO2_discovery.RAW.ms1.clean.diff	-0.499985	0.5
121213_PhosphoMRM_TiO2_discovery.RAW.ms2.clean.diff	-0.42815	0.325621
ADH_100126_mix.raw.ms1.clean.diff	0	0
Velos_120905_09.raw.ms1.clean.diff	-0.499939	0.5
Velos_120905_09.raw.ms2.clean.diff	-0.49959	0.499481
peakPicked.121213_PhosphoMRM_TiO2_discovery.ms1.clean.diff	-0.499939	0.5
peakPicked.121213_PhosphoMRM_TiO2_discovery.ms2.clean.diff	-0.42815	0.325621

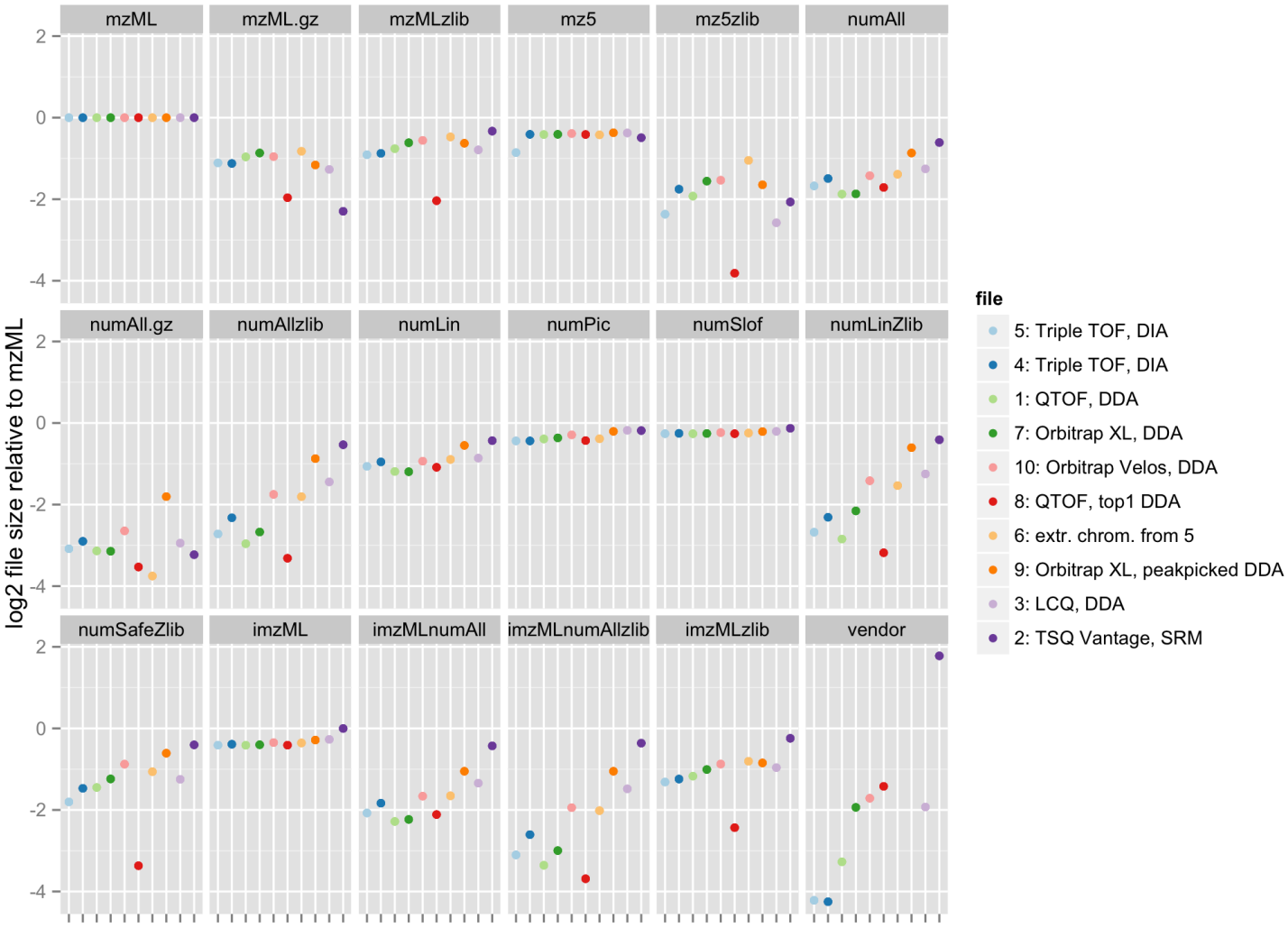
Supplementary Table 7: Size and timing statistics

color legend:	20%	50%	100%	200%	500%

			high performance computers				all computers			
	size		read time		write time		read time		write time	
	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd
mzML	79.9%	7.3%	30.7%	31.0%	79.8%	16.6%	32.1%	31.6%	198.5%	581.1%
mzMLnumAll	34.2%	16.1%	26.3%	31.4%	106.2%	37.3%	31.0%	34.1%	288.3%	784.3%
mzMLnumAllzlib	27.1%	21.4%	27.4%	33.1%	144.6%	64.5%	29.4%	33.2%	140.6%	71.9%
mzMLzlib	49.8%	16.1%	30.5%	31.0%	237.4%	107.6%	34.2%	33.4%	257.4%	176.6%
mz5	73.3%	6.4%	23.3%	14.1%	69.7%	15.6%	28.8%	21.2%	135.6%	220.1%
mz5zlib	27.2%	11.1%	39.4%	7.1%	101.1%	22.5%	39.3%	13.6%	118.2%	103.3%
mzML	100.0%	0.0%	100.0%	0.0%	100.0%	0.0%	100.0%	0.0%	100.0%	0.0%
mzML.gz	43.9%	11.7%	179.2%	53.0%	836.7%	776.5%	153.7%	56.5%	705.3%	713.1%
mzMLzlib	59.9%	14.5%	113.8%	15.0%	257.3%	111.1%	102.5%	35.1%	218.6%	110.8%
numAll	39.0%	12.1%	63.8%	11.2%	97.0%	8.5%	62.3%	22.2%	117.5%	129.3%
numAll.gz	13.2%	5.8%	87.0%	19.8%	281.0%	129.2%	78.9%	29.1%	237.7%	125.1%
numAllzlib	29.3%	18.9%	71.2%	17.8%	136.2%	30.3%	68.6%	28.5%	121.9%	39.9%
numLin	53.8%	9.8%	70.7%	9.8%	92.7%	6.2%	65.8%	20.6%	89.7%	18.5%
numLinZlib	33.8%	21.5%	76.8%	22.4%	157.2%	54.3%	71.8%	31.2%	136.9%	58.3%
numPic	79.7%	5.8%	82.1%	14.9%	111.5%	9.7%	75.4%	24.9%	107.7%	31.0%
numSafeZlib	43.9%	18.1%	102.8%	18.5%	288.5%	137.5%	93.6%	35.1%	240.2%	128.4%
numSlof	85.2%	2.5%	92.7%	1.4%	114.2%	49.6%	85.3%	22.9%	109.7%	41.9%
vendor	60.5%	110.9%	247.4%	499.6%	NA	NA	208.9%	414.3%	NA	NA

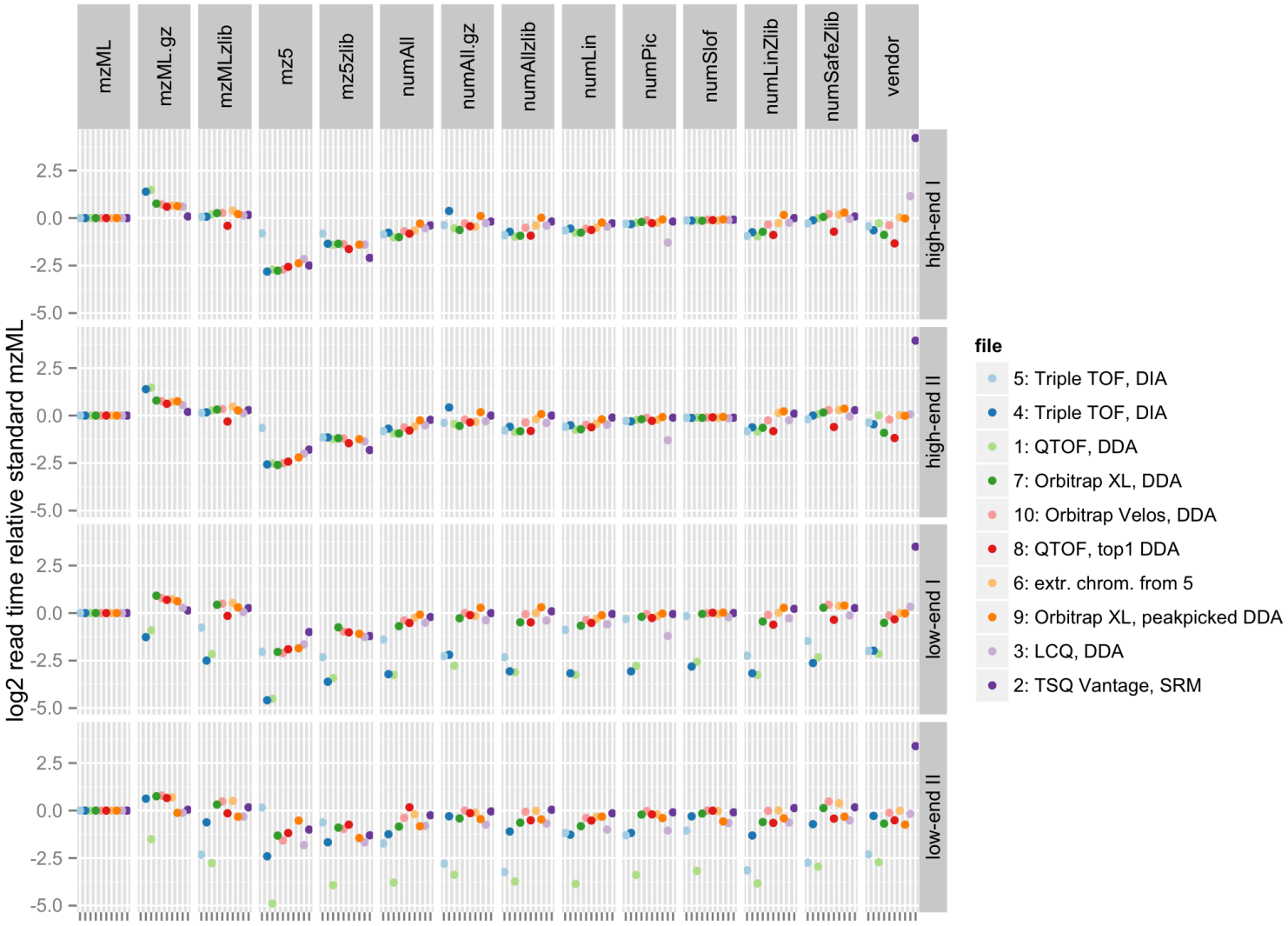
Supplementary Figure 2: Relative file size

Log_2 file size subtracted by log_2 mzML file size for all file formats and files. Files are sorted in descending mzML-size order (same as in Fig 1c). Note that file 6 and 9 do not exists in vendor format, since they are computer derivates on other files.



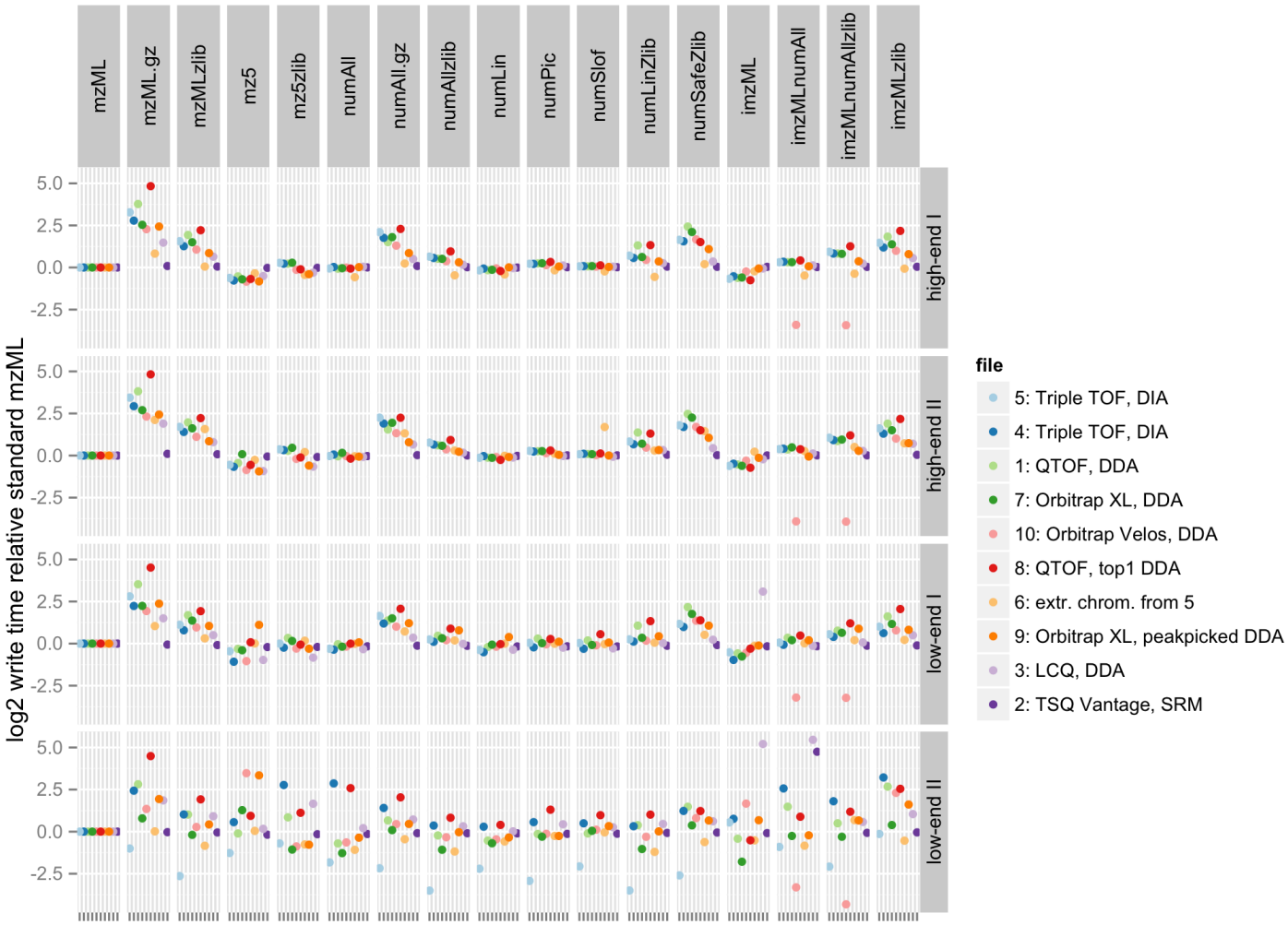
Supplementary Figure 3: Relative read time

Log$_2$ read times subtracted by mzML log$_2$ read time for all file formats, computers and files. Files are sorted in descending mzML-size order (same as in Fig 1c).



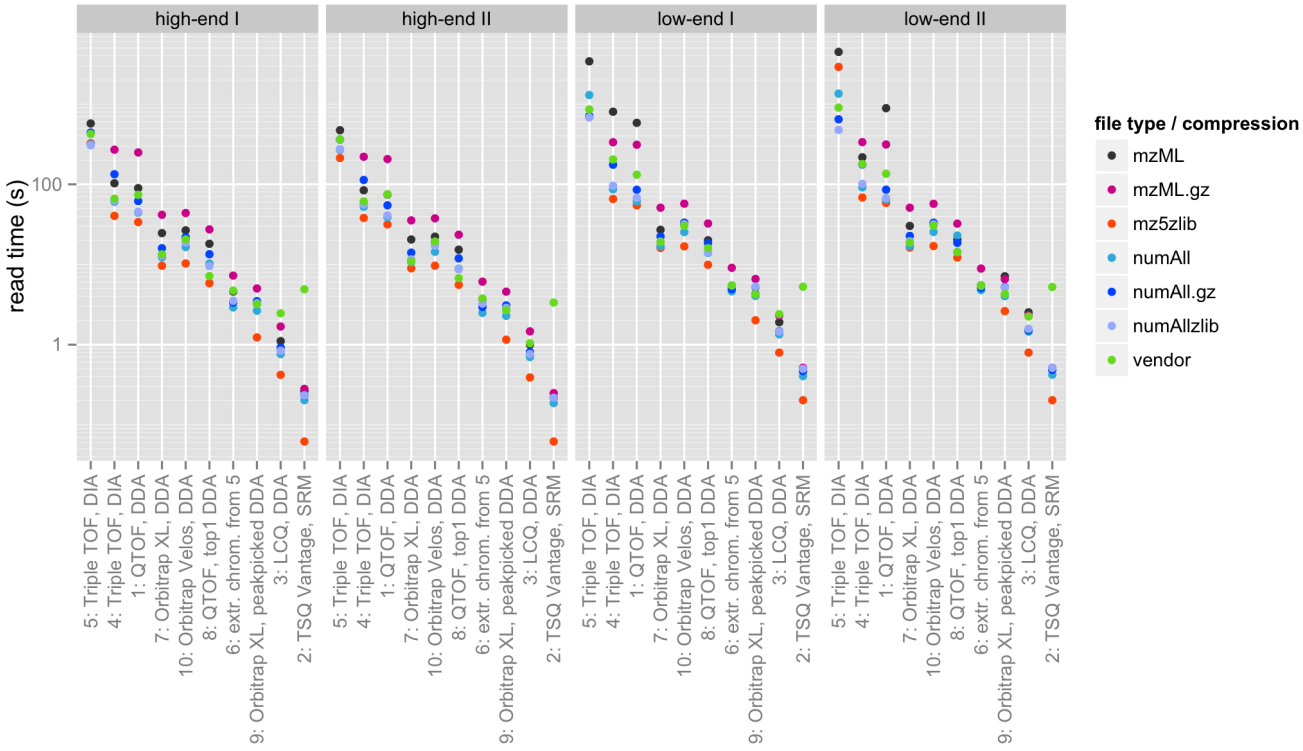
Supplementary Figure 4: Relative write time

Log_2 write times subtracted by mzML log_2 write time for all file formats, computers and files. Files are sorted in descending mzML-size order (same as in Fig 1c). Note that for us it is not possible to write in the vendor formats.



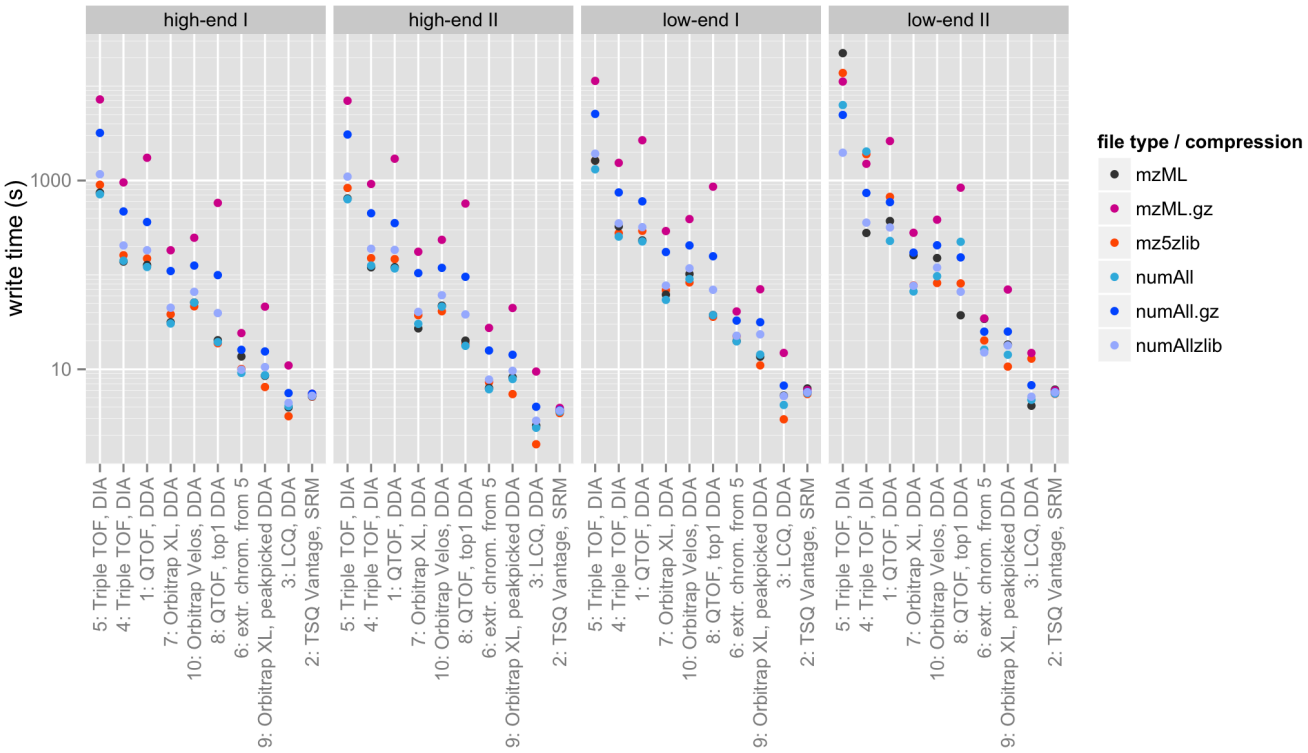
Supplementary Figure 5: Absolute read time

The read time in seconds for selected file formats and all files, spanning over 3 orders of magnitude. Files are sorted in descending mzML-size order (same as in Fig 1c).



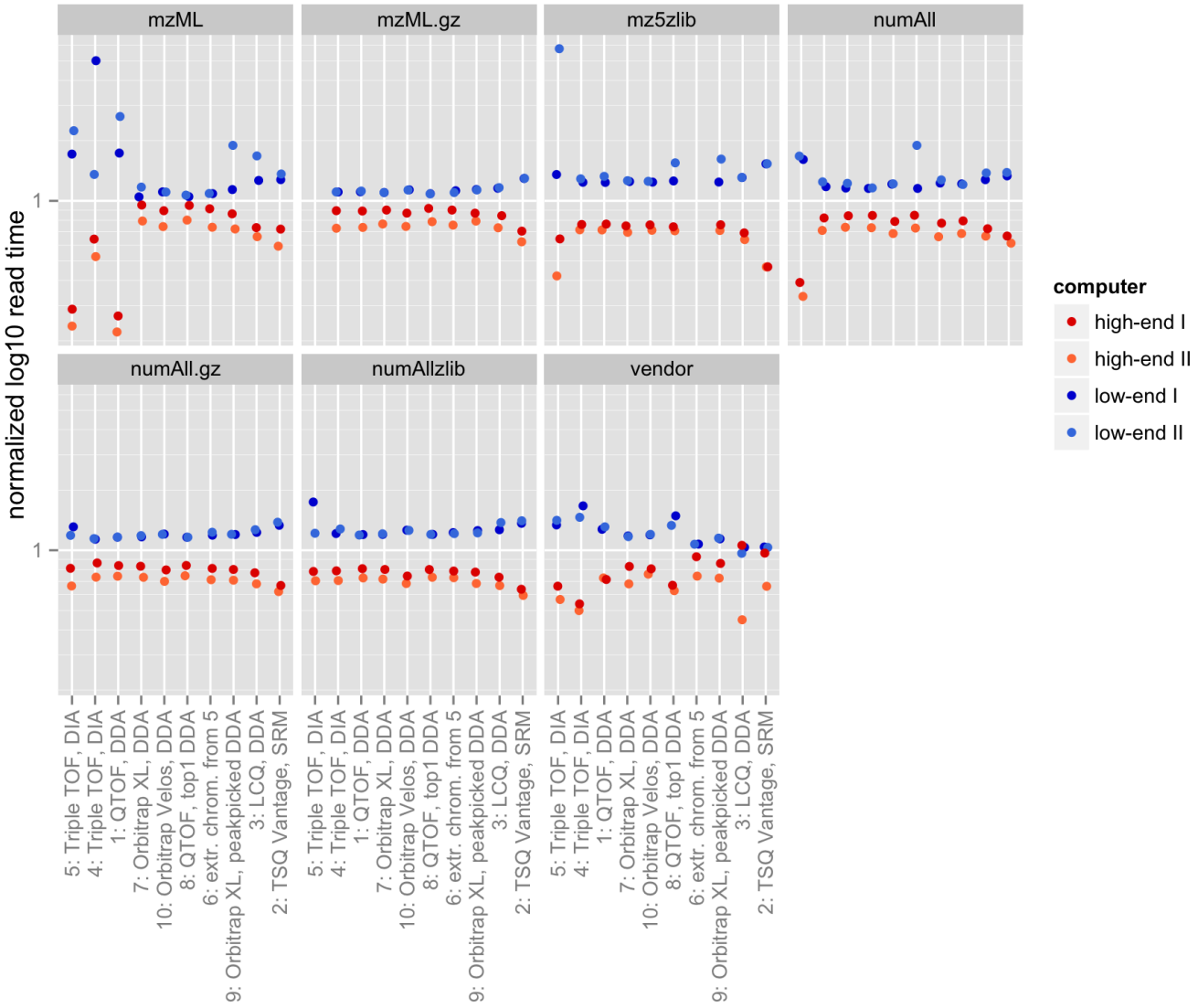
Supplementary Figure 6: Absolute write time

The write time in seconds for selected file formats and all files, spanning over 4 orders of magnitude. Files are sorted in descending mzML-size order (same as in Fig 1c).



Supplementary Figure 7: Read time for different computers

Comparison of read times on different computer hardware for different file formats. Files are sorted in descending mzML-size order (same as in Fig 1c). Read times were normalized by the median read time for each file and format.



Supplementary Figure 8: Write time for different computers

Comparison of write times on different computer hardware for different file formats. Files are sorted in descending mzML-size order (same as in Fig 1c). Write times were normalized by the median write time for each file and format.