The Journal of Open Source Software

matchms - processing and similarity evaluation of mass spectrometry data.

Florian Huber[1], Stefan Verhoeven[1], Christiaan Meijer[1], Hanno Spreeuw[1], Cunliang Geng[1], Simon Rogers[2], Justin J. J. van der Hooft[3], Adam Belloum[1], Faruk Diblen[1], and Jurriaan H. Spaaks[1]

1 Netherlands eScience Center, Science Park 140, 1098XG Amsterdam, The Netherlands **2** School of Computing Science, University of Glasgow, Glasgow, United Kingdom **3** Bioinformatics Group, Plant Sciences Group, University of Wageningen, Wageningen, the Netherlands

DOI: 10.21105/joss.02411

Software

- Review
- Repository
- Archive

Editor: Arfon Smith

Reviewers:

- @bittremieux
- @kaibioinfo

Submitted: 26 June 2020
Published: 01 July 2020

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

Mass spectrometry data is at the heart of numerable applications in the biomedical and life sciences. With growing use of high throughput techniques researchers need to analyse larger and more complex datasets. In particular through joint effort in the research community, fragmentation mass spectrometry datasets are growing in size and number. Platforms such as MassBank (Horai et al., 2010), GNPS (Wang et al., 2016) or MetaboLights (Haug et al., 2020) serve as an open-access hub for sharing of raw, processed, or annotated fragmentation mass spectrometry data (MS/MS). Without suitable tools, however, full quantitative analysis and exploitation of such datasets remains overly challenging. In particular, large collected datasets contain data aquired using different instruments and measurement conditions, and can further contain a significant fraction of inconsistent, wrongly labeled, or incorrect metadata (annotations).

`Matchms` is an open-access Python package to import, process, clean, and compare mass spectrometry data (MS/MS) (see Figure 1). It allows to implement and run an easy-to-follow, easy-to-reproduce workflow from raw mass spectra to pre- and post-processed spectral data. Raw data can be imported from commonly used MGF files (via pyteomics (Levitsky, Klein, Ivanov, & Gorshkov, 2019)(Goloborodko, Levitsky, Ivanov, & Gorshkov, 2013)) or more convenient-to-handle json files. `Matchms` contains a large number of metadata cleaning and harmonizing filter functions that can easily be stacked to construct a desired pipeline (Figure 2), which can also easily be extended by custom functions wherever needed. Available filters include extensive cleaning, correcting, checking of key metadata fields such as compound name, structure annotations (InChI, Smiles, InchiKey), ionmode, adduct, or charge.

The Journal of Open Source Software

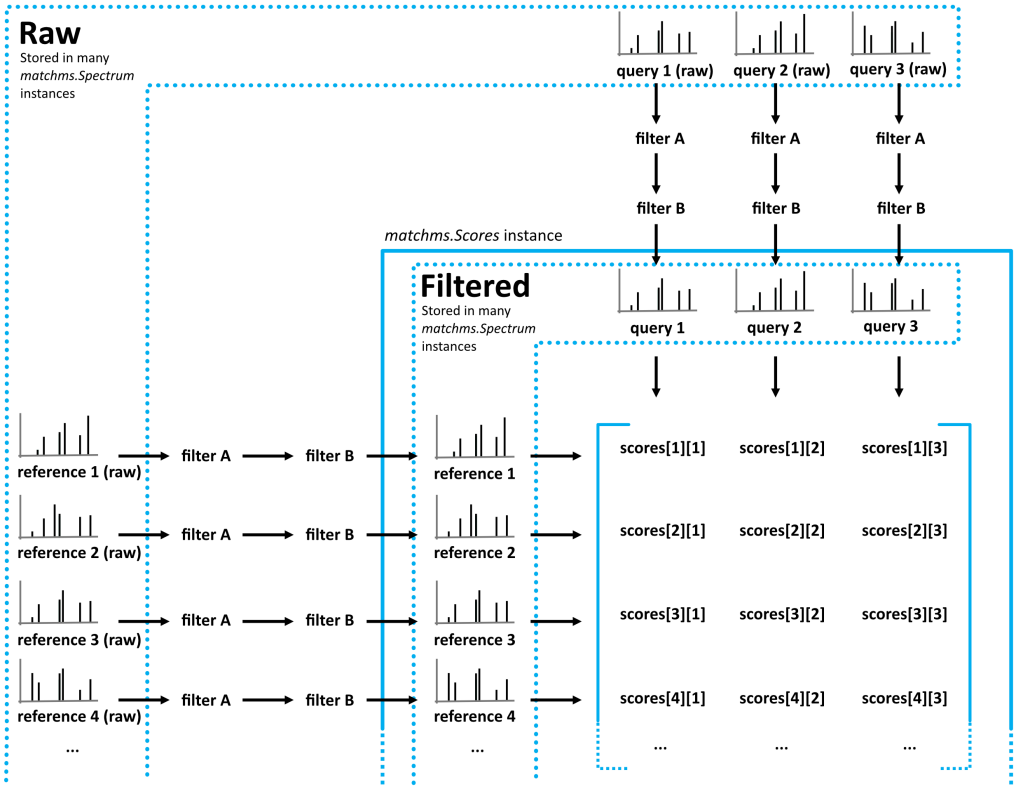

Figure 1: Flowchart of `matchms` workflow. Reference and query spectrums are filtered using the same set of set filters (here: filter A and filter B). Once filtered, every reference spectrum is compared to every query spectrum using the `matchms.Scores` object.

`Matchms` further provides functions to derive different similarity scores between spectra. Those include the established spectra-based measures of the cosine score or modified cosine score (Watrous et al., 2012). The package also offers fast implementations of common similarity measures (Dice, Jaccard, Cosine) that can be used to compute similarity scores between molecular fingerprints (rdkit, morgan1, morgan2, morgan3, all implemented using rdkit (Landrum, n.d.)). `Matchms` easily facilitates deriving similarity measures between large number of spectra at comparably fast speed due to score implementations based on Numpy (Walt, Colbert, & Varoquaux, 2011), Scipy (Virtanen et al., 2020), and Numba (Lattner & Adve, 2004). Additional similarity measures can easily be added using the `matchms` API. The provided API also allows to quickly compare, sort, and inspect query versus reference spectra using either the included similarity scores or added custom measures. The API was designed to be easily extensible so that users can add their own filters for spectra preocessing, or their own similarity functions for spectral comparisons. The present set of filters and similarity functions was mostly geared towards smaller molecules and natural compounds, but it could easily be extended by functions specific to larger peptides or proteins.

`Matchms` is freely accessible either as conda package (https://anaconda.org/nlesc/matchms), or in form of source-code on GitHub (https://github.com/matchms/matchms). For further code examples and documentation see https://matchms.readthedocs.io/en/latest/. All main functions are covered by tests and continuous integration to offer reliable functionality. We explicitly value future contributions from a mass spectrometry interested community and hope that "matchms" can serve as a reliable and accessible entry point for handling complex mass spectrometry datasets using Python.

Example workflow

A typical workflow with `matchms` looks as indicated in Figure 1, or as described in the following code example.

```
from matchms.importing import load_from_mgf
from matchms.filtering import default_filters
from matchms.filtering import normalize_intensities
from matchms import calculate_scores
from matchms.similarity import CosineGreedy

# Read spectrums from a MGF formatted file
file = load_from_mgf("all_your_spectrums.mgf")

# Apply filters to clean and enhance each spectrum
spectrums = []
for spectrum in file:
    spectrum = default_filters(spectrum)
    spectrum = normalize_intensities(spectrum)
    spectrums.append(spectrum)

# Calculate Cosine similarity scores between all spectrums
scores = calculate_scores(references=spectrums,
                          queries=spectrums,
                          similarity_function=CosineGreedy())

# Print the calculated scores for each spectrum pair
for score in scores:
    (reference, query, score, n_matching) = score
    # Ignore scores between same spectrum and
    # pairs which have less than 20 peaks in common
    if reference != query and n_matching >= 20:
        print(f"Reference scan id: {reference.metadata['scans']}")
        print(f"Query scan id: {query.metadata['scans']}")
        print(f"Score: {score:.4f}")
        print(f"Number of matching peaks: {n_matching}")
        print("-----------------------------")
```

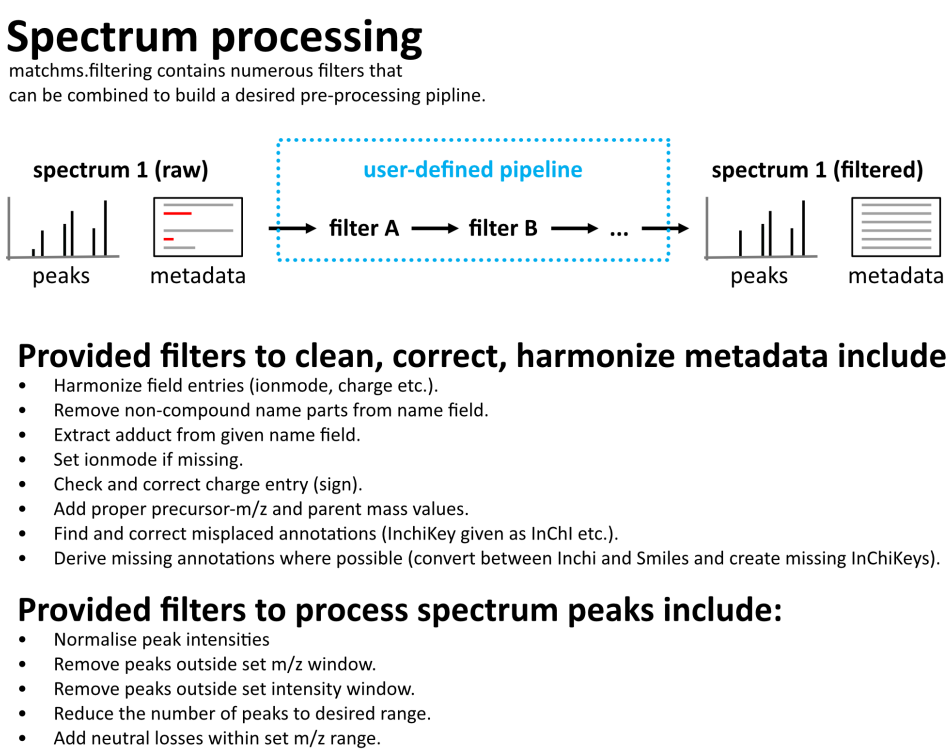

Figure 2: `Matchms` provided a range of filter functions to process spectrum peaks and metadata. Filters can easily be stacked and combined to build a desired pipeline. The API also makes it easy to extend customer pipelines by adding own filter functions.

Processing spectrum peaks and plotting

`Matchms` provides numerous filters to process mass spectra peaks. Below a simple example to remove low intensity peaks from a spectrum (Figure 3).

```
from matchms.filtering import require_minimum_number_of_peaks
from matchms.filtering import select_by_mz
from matchms.filtering import select_by_relative_intensity

def process_peaks(s):
    s = select_by_mz(s, mz_from=0, mz_to=1000)
    s = select_by_relative_intensity(s, intensity_from=0.001)
    s = require_minimum_number_of_peaks(s, n_required=10)
    return s

# Apply processing steps to spectra (here to a single "spectrum_raw")
spectrum_processed = process_peaks(spectrum_raw)

# Plot raw spectrum (all and zoomed in)
spectrum_raw.plot()
spectrum_raw.plot(intensity_to=0.02)

# Plot processed spectrum (all and zoomed in)
spectrum_processed.plot()
spectrum_processed.plot(intensity_to=0.02)
```

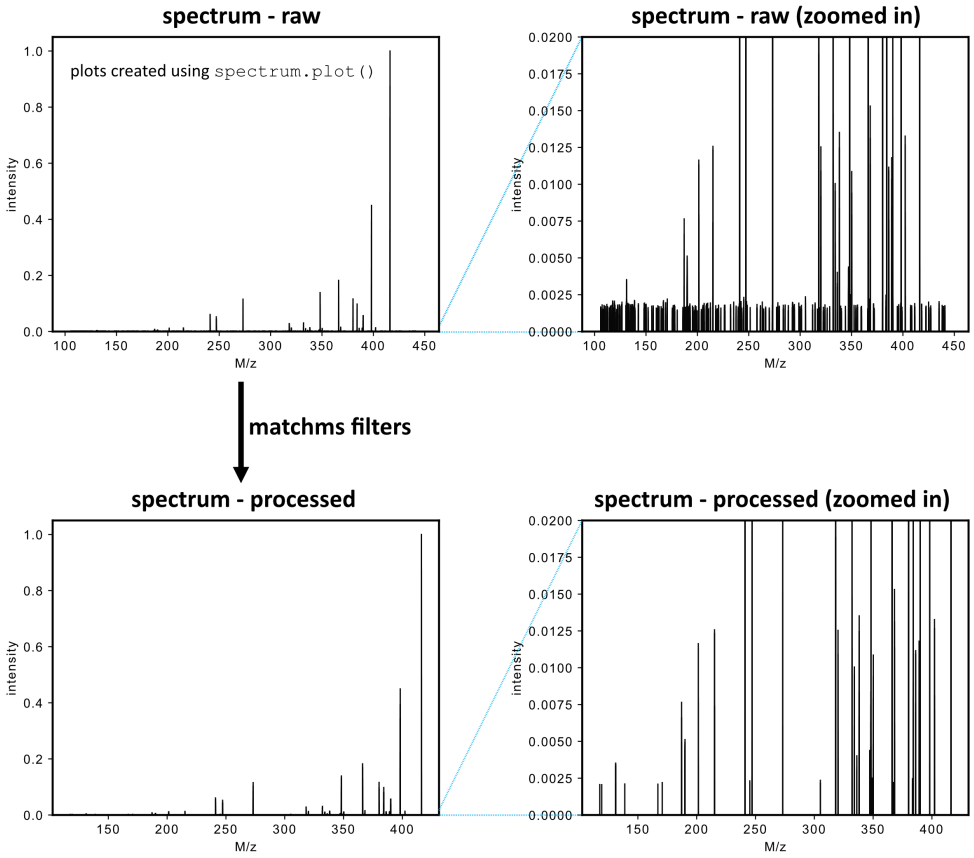

Figure 3: Example of `matchms` peak filtering applied to an actual spectrum using `select_by_relative_intensity` to remove peaks of low relative intensity. Spectra are plotted using the provided `spectrum.plot()` function.

References

Goloborodko, A. A., Levitsky, L. I., Ivanov, M. V., & Gorshkov, M. V. (2013). Pyteomics - a Python Framework for Exploratory Data Analysis and Rapid Software Prototyping in Proteomics. *Journal of the American Society for Mass Spectrometry*, *24*(2), 301–304. doi:10.1021/jasms.8b04453

Haug, K., Cochrane, K., Nainala, V. C., Williams, M., Chang, J., Jayaseelan, K. V., & O'Donovan, C. (2020). MetaboLights: A resource evolving in response to the needs of its scientific community. *Nucleic Acids Research*, *48*(D1), D440–D444. doi:10.1093/nar/gkz1019

Horai, H., Arita, M., Kanaya, S., Nihei, Y., Ikeda, T., Suwa, K., Ojima, Y., et al. (2010). MassBank: A public repository for sharing mass spectral data for life sciences. *Journal of Mass Spectrometry*, *45*(7), 703–714. doi:10.1002/jms.1777

Landrum, G. (n.d.). RDKit: Open-source cheminformatics. Retrieved from http://www.rdkit.org

Lattner, C., & Adve, V. (2004). LLVM: A compilation framework for lifelong program analysis and transformation. In *Cgo* (pp. 75–88). San Jose, CA, USA.

Levitsky, L. I., Klein, J. A., Ivanov, M. V., & Gorshkov, M. V. (2019). Pyteomics 4.0: Five Years of Development of a Python Proteomics Framework. *Journal of Proteome Research*,

18(2), 709–714. doi:10.1021/acs.jproteome.8b00717

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., et al. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. doi:https://doi.org/10.1038/s41592-019-0686-2

Walt, S. van der, Colbert, S. C., & Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science Engineering*, *13*(2), 22–30. doi:10.1109/MCSE.2011.37

Wang, M., Carver, J. J., Phelan, V. V., Sanchez, L. M., Garg, N., Peng, Y., Nguyen, D. D., et al. (2016). Sharing and community curation of mass spectrometry data with global natural products social molecular networking. *Nature Biotechnology*, *34*(8), 828–837. doi:10.1038/nbt.3597

Watrous, J., Roach, P., Alexandrov, T., Heath, B. S., Yang, J. Y., Kersten, R. D., Voort, M. van der, et al. (2012). Mass spectral molecular networking of living microbial colonies. *Proceedings of the National Academy of Sciences of the United States of America*, *109*(26), E1743–1752. doi:10.1073/pnas.1203689109