The Journal of Open Source Software

SpectriPy: Enhancing Cross-Language Mass Spectrometry Data Analysis with R and Python

Marilyn De Graeve [1], **Wout Bittremieux** [2], **Thomas Naake** [3], **Carolin Huber** [4], **Matthias Anagho-Mattanovich** [5], **Nils Hoffmann** [6], **Pierre Marchal** [7], **Victor Chrone** [8], **Philippine Louail** [1], **Helge Hecht** [9], **Michael Witting** [10,11], **and Johannes Rainer** [1¶]

1 Institute for Biomedicine, Eurac Research, Bolzano, Italy **2** Department of Computer Science, University of Antwerp, Antwerpen, Belgium **3** Genome Biology Unit, European Molecular Biology Laboratory (EMBL), Heidelberg, Germany **4** Department of Exposure Science, Helmholtz Centre for Environmental Research - UFZ, Leipzig, Germany **5** Novo Nordisk Foundation Center for Basic Metabolic Research, University of Copenhagen, Copenhagen, Denmark **6** Institute for Bio- and Geosciences (IBG-5), Forschungszentrum Jülich GmbH, Jülich, Germany **7** Department of Medical Oncology, University of Bern, Bern, Switzerland **8** Alphalyse, Odense, Denmark **9** RECETOX, Faculty of Science, Masaryk University, Brno, Czech Republic **10** Metabolomics and Proteomics Core, Helmholtz Zentrum München, Munich, Germany **11** Chair of Analytical Food Chemistry, TUM School of Life Sciences, Technical University of Munich, Freising-Weihenstephan, Germany ¶ Corresponding author

DOI: 10.21105/joss.08070

Software

- Review
- Repository
- Archive

Editor: Daniel S. Katz

Reviewers:

- @Adafede
- @AnthonyOfSeattle

Submitted: 07 April 2025
Published: 19 May 2025

Figure 1: SpectriPy package logo

Summary

Mass spectrometry (MS) is a key technology used across multiple fields, including biomedical research and life sciences. The data is often times large and complex, and analyses must be tailored to the experimental and instrumental setups. Excellent software libraries for such data analysis are available in both R and Python, including R packages from the RforMassSpectrometry initiative such as *Spectra*, *MsCoreUtils*, *MetaboAnnotation*, and *CompoundDb* (Rainer et al., 2022), as well as Python libraries like *matchms* (Huber et al., 2020), *spectrum_utils* (Bittremieux, 2020), *Pyteomics* (Goloborodko et al., 2013), and *pyOpenMS* (Röst et al., 2014). The *reticulate* R package (Ushey et al., 2025) provides an R interface to Python enabling interoperability between the two programming languages. The open-source *SpectriPy* R package builds upon *reticulate* and provides functionality to efficiently translate between R and Python MS data structures. It can convert between R's `Spectra::Spectra` and Python's `matchms.Spectrum` and `spectrum_utils.spectrum.MsmsSpectrum` objects and

includes functionality to directly apply spectral similarity, filtering, normalization, etc. routines from the Python *matchms* library on MS data in R. *SpectriPy* hence enables and simplifies the integration of R and Python for MS data analysis, empowering data analysts to benefit from the full power of algorithms in both programming languages. Furthermore, software developers can reuse algorithms across languages rather than re-implementing them, enhancing efficiency and collaboration.

Statement of need

Over the past decade, tremendous efforts have been made to develop powerful algorithms and excellent data analysis software for MS data analysis. Each of these software packages covers different and in part complementary aspects in the analysis of MS data, but their integration into a single workflow remains a challenge, in particular across programming languages. To avoid the need for repeated implementation of algorithms in different programming languages we developed the *SpectriPy* package. By leveraging R's *reticulate* package, and translating between R and Python MS data structures, this package enables seamless cross-language integration of MS data analysis algorithms within unified analysis workflows.

Description

Reproducible examples and use case analyses on how to share and translate MS data structures between R and Python, and combined Python and R-based analysis workflows for LC-MS/MS data annotation enabled by *SpectriPy* can be found in the package's vignette and in one of the example workflows of the Metabonaut resource (Louail & Rainer, 2025). In this paper, we primarily focus on the technical details and features of the package.

Installation

During installation, *SpectriPy* automatically configures a Python environment and installs all required libraries. The installation can be configured through several environment variables that also allow users to disable the automatic setup and instead use an available Python environment of the host system. Detailed installation instructions can be found in the package's GitHub repository and in the package's vignette.

Translating MS data objects between R and Python

As its core functionality, *SpectriPy* allows translation between R and Python MS data structures. In particular, *SpectriPy* provides the functions `rspec_to_pyspec()` and `pyspec_to_rspec()` to convert between R's `Spectra::Spectra` and Python's `matchms.Spectrum` and `spectrum_utils.spectrum.MsmsSpectrum` objects. These functions also handle the conversion, and any required renaming and reformatting of spectra metadata, such as MS level, retention times, or any other arbitrary metadata available in the MS data object. An example combined R-Python data analysis workflow, which can be realized using the Quarto system is provided in the following code snippets. In this particular example we start the analysis in Python, loading and processing the MS data with functions from the *matchms* library.

```
#' Python session:
#'  Import data and perform initial processing
import matchms
import matchms.filtering as mms_filt
from matchms.importing import load_from_mgf
mgf_py = list(load_from_mgf(<MGF file>))
```

```
#'  Scale intensities
for i in range(len(mgf_p)):
  mgf_py[i] = mms_filt.normalize_intensities(mgf_py[i])
```

To continue the analysis in R, we could either translate to full MS data to R using the `pyspec_to_rspec()` function, or, as shown in the code block below, create a `Spectra::Spectra` object using *SpectriPy*'s `MsBackendPy` *backend* class. This class acts as an interface to the MS data in the associated Python session. All data from the referenced Python data object is accessible in R, with the entire or subsets of the data translated on-the-fly from Python to R only upon request. This strategy ensures memory efficiency and minimizes the number of data copies.

```
#' R session:
#'  Create an R data object for the MS data in the associated Python session
library(Spectra)
library(SpectriPy)
sps <- Spectra("mgf_py", source = MsBackendPy())

#'  Retrieve the MS peaks data for the 1st spectrum
peaksData(sps[1])
```

The use of the `MsBackendPy` enables thus seamless and, compared to the alternative `pyspec_to_rspec()`, more memory-efficient integration of Python MS data objects into R for powerful cross-language analysis workflows.

Integrated functionality from the *matchms* Python library

SpectriPy's `compareSpectriPy()` and `filterSpectriPy()` functions allow spectra comparison, and filtering and processing routines, respectively, from the *matchms* Python library to be called directly from R. These functions internally translate the MS data from a `Spectra::Spectra` object to the respective Python MS data structures, execute the Python functions, and collect and convert the results to R data types, enabling the integration of functionality from the *matchms* Python library directly into R-based analysis workflows.

As such, *SpectriPy* provides an easy way to compare spectra similarity functions from commonly-used R and Python libraries, e.g. during LC-MS/MS data annotation. As an example, the Cosine (i.e., Dot product) and Cosine Hungarian similarity scores are compared between two sets of spectra, calculated with *Spectra*'s built-in `compareSpectra()` and *SpectriPy*'s `compareSpectriPy()` calling the `CosineHungarian` function from *matchms*, respectively.

```
#' R session:
#'  Calculate similarity scores
res_cosine <- compareSpectra(sps1, sps2)
res_cosinehungarian <- compareSpectriPy(
    sps1, sps2, param = CosineHungarian(tolerance = 0.1))

#'  Plot the similarity scores
plot(res_cosine, res_cosinehungarian, pch = 21, col = "#000000ce",
     bg = "#00000060", xlab = "Dot product", ylab = "Cosine Hungarian")
grid()
```

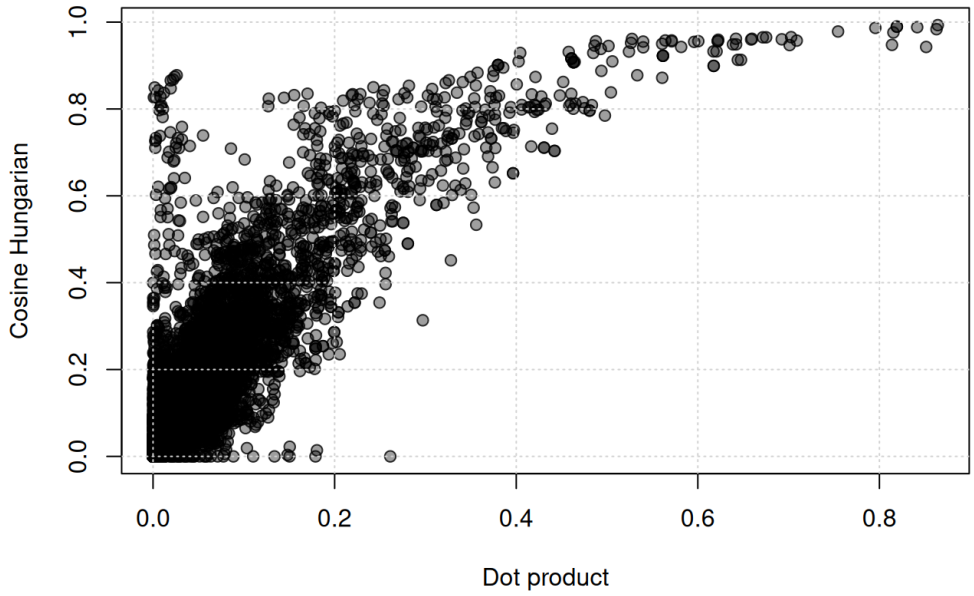

Figure 2: Comparison of different spectra similarity scores calculated with either the *Spectra* R package or the Python *matchms* library.

Perspective

SpectriPy started as a collaboration of R and Python developers, with the latest contributions added during the EuBIC-MS Developers Meeting in 2025. Collaborative development will be further encouraged to extend *SpectriPy* with additional functionality (e.g., advanced spectra similarity matching methods *spec2vec* and *ms2deepscore*), support for additional libraries (e.g., *Pyteomics*, *pyOpenMS*), and data structures. New use cases will be integrated into larger interactive tutorial frameworks such as the Metabonaut resource (Louail & Rainer, 2025), enabling users to seamlessly integrate R and Python into their MS data analysis pipelines.

Ultimately, the long-term goal is to promote cross-language compatibility and reproducibility in computational mass spectrometry. By leveraging the strengths of both R and Python, *SpectriPy* will help develop flexible and efficient MS data analysis workflows, reduce redundancy and promote innovation in the field.

Acknowledgements

The authors declare that they do not have any competing financial or personal interests that could have influenced the work reported in this paper. Part of this work was supported by the European Union HORIZON-MSCA-2021 project 101073062: "HUMAN - Harmonizing and unifying blood metabolic analysis networks" to Philippine Louail, Johannes Rainer and Micheal Witting. Helge Hecht thanks the RECETOX Research Infrastructure (No LM2023069) financed by the Ministry of Education, Youth and Sports for supportive background. Part of this project was supported from the European Union's Horizon 2020 research and innovation programme under grant agreement 857560 (CETOCOEN Excellence) and from the Horizon Europe programme under grant agreement 101079789 (EIRENE PPP). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of

the European Union or European Research Executive Agency (REA). Neither the European Union nor the granting authority can be held responsible for any use that may be made of the information it contains.

References

Bittremieux, W. (2020). Spectrum_utils: A Python package for mass spectrometry data processing and visualization. *Analytical Chemistry*, *92*(1), 659–661. https://doi.org/10.1021/acs.analchem.9b04884

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. https://doi.org/10.1007/s13361-012-0516-6

Huber, F., Verhoeven, S., Meijer, C., Spreeuw, H., Castilla, E., Geng, C., Van Der Hooft, J., Rogers, S., Belloum, A., Diblen, F., & Spaaks, J. (2020). Matchms - processing and similarity evaluation of mass spectrometry data. *Journal of Open Source Software*, *5*(52), 2411. https://doi.org/10.21105/joss.02411

Louail, P., & Rainer, J. (2025). *Rformassspectrometry/metabonaut: Metabonaut version 1.0.0* (Version v1.0.0). Zenodo. https://doi.org/10.5281/ZENODO.15062930

Rainer, J., Vicini, A., Salzer, L., Stanstrup, J., Badia, J. M., Neumann, S., Stravs, M. A., Verri Hernandes, V., Gatto, L., Gibb, S., & Witting, M. (2022). A modular and expandable ecosystem for metabolomics data annotation in R. *Metabolites*, *12*(2), 173. https://doi.org/10.3390/metabo12020173

Röst, H. L., Schmitt, U., Aebersold, R., & Malmström, L. (2014). pyOpenMS: A Python-based interface to the OpenMS mass-spectrometry algorithm library. *Proteomics*, *14*(1), 74–77. https://doi.org/10.1002/pmic.201300246

Ushey, K., Allaire, J., & Tang, Y. (2025). *Reticulate: Interface to 'Python'*. https://doi.org/10.32614/CRAN.package.reticulate