The Journal of Open Source Software

PeakPerformance - A tool for Bayesian inference-based fitting of LC-MS/MS peaks

Jochen Nießer [1,2], **Michael Osthege** [1], **Eric von Lieres** [1,3], **Wolfgang Wiechert** [1,3], **and Stephan Noack** [1]

1 Institute for Bio- and Geosciences (IBG-1), Forschungszentrum Jülich GmbH, Jülich, Germany **2** RWTH Aachen University, Aachen, Germany **3** Computational Systems Biotechnology, RWTH Aachen University, Aachen, Germany

DOI: 10.21105/joss.07313

Software

- Review
- Repository
- Archive

Editor: Charlotte Soneson

Reviewers:

- @Adafede
- @lazear

Submitted: 21 August 2024
Published: 05 December 2024

Summary

A major bottleneck of chromatography-based analytics has been the elusive fully automated identification and integration of peak data without the need of extensive human supervision. The presented Python package `PeakPerformance` applies Bayesian inference to chromatographic peak fitting, and provides an automated approach featuring model selection and uncertainty quantification. Regarding peak acceptance, it improves on vendor software solutions with more sophisticated, multi-layered metrics for decision making based on convergence of the parameter estimation as well as the uncertainties of peak parameters. Currently, its application is focused on data from targeted liquid chromatography tandem mass spectrometry (LC-MS/MS), but its design allows for an expansion to other chromatographic techniques and accommodates users with little programming experience by supplying convenience functions and relying on Microsoft Excel for data input and reporting. `PeakPerformance` is implemented in Python, its source code is available on GitHub, and a thorough documentation is available under https://peak-performance.rtfd.io. It is unit-tested on Linux and Windows and accompanied by example notebooks.

Statement of need

In biotechnological research and industrial applications, chromatographic techniques are ubiquitously used to analyze samples from fermentations, e.g. to determine the concentration of substrates and products. Over the course of a regular lab-scale bioreactor fermentation, hundreds of samples and subsequently thousands of chromatographic peaks may accrue. This is exacerbated by the spread of microbioreactors causing a further increase in the amount of samples per time (Hemmerich et al., 2018; Kostov et al., 2001). While the recognition and integration of peaks by vendor software is – in theory – automated, it typically requires visual inspection and occasional manual re-integration by the user due to a large number of false positives, false negatives or incorrectly determined baselines, ultimately downgrading it to a semi-automated process. Since this is a time-consuming, not to mention tedious, procedure and introduces the problem of comparability between purely manual and algorithm-based integration as well as user-specific differences, we instead propose a peak fitting solution based on Bayesian inference. The advantage of this approach is the complete integration of all relevant parameters – i.e. baseline, peak area and height, mean, signal-to-noise ratio etc. – into one single model through which all parameters are estimated simultaneously. Furthermore, Bayesian inference comes with uncertainty quantification for all peak model parameters, and thus does not merely yield a point estimate as would commonly be the case. It also grants access to novel metrics for avoiding false positives and negatives by rejecting

signals where a) a convergence criterion of the peak fitting procedure was not fulfilled or b) the uncertainty of the estimated parameters exceeded a user-defined threshold. By employing peak fitting to uncover peak parameters – most importantly the area – this approach thus differs from recent applications of Bayesian statistics to chromatographic peak data which e.g. focussed on peak detection (Vivó-Truyols, 2012; Woldegebriel & Vivó-Truyols, 2015), method optimization (Wiczling & Kaliszan, 2016) and simulations of chromatography (Briskot et al., 2019; Yamamoto et al., 2021). The first studies to be published about this topic contain perhaps the technique most similar in spirit to the present one since functions made of an idealized peak shape and a noise term are fitted but beyond this common starting point the methodology is quite distinct (Kelly & Harris, 1971b, 1971a).

Materials and Methods

Implementation

`PeakPerformance` is an open source Python package compatible with Windows and Linux/Unix platforms. At the time of manuscript submission, it features three modules: `pipeline`, `models`, and `plotting`. Due to its modular design, `PeakPerformance` can easily be expanded by adding e.g. additional models for deviating peak shapes or different plots. Currently, the featured peak models describe peaks in the shape of normal or skew normal distributions, as well as double peaks of normal or skewed normal shape. The normal distribution is regarded as the ideal peak shape and common phenomena like tailing and fronting can be expressed by the skew normal distribution (Azzalini, 1985). Bayesian inference is conducted utilizing the PyMC package (Abril-Pla et al., 2023) with the external sampler `nutpie` for improved performance (Seyboldt & PyMC Developers, 2022). Both model selection and analysis of inference data objects are realized with the ArviZ package (Kumar et al., 2019). Since the inference data is stored alongside graphs and report sheets, users may employ the ArviZ package or others for further analysis of the results if necessary.

Results and Discussion

PeakPerformance workflow

`PeakPerformance` accommodates the use of a pre-manufactured data pipeline for standard applications (Fig. 1) as well as the creation of custom data pipelines using only its core functions. The provided data analysis pipeline was designed in a user-friendly way and is covered by multiple example notebooks.

Before using `PeakPerformance`, the user has to supply raw data files containing a NumPy array with time in the first and intensity in the second dimension for each peak as described in detail in the documentation. Using the `prepare_model_selection()` method, an Excel template file ("Template.xlsx") for inputting user information is prepared and stored in the raw data directory.

Since targeted LC-MS/MS analyses essentially cycle through a list of mass traces for every sample, a model type has to be assigned to each mass trace. If this is not done by the user, an optional automated model selection step will be performed, where one exemplary peak per mass trace is analyzed with all models to identify the most appropriate one. Its results for each model are ranked based on Pareto-smoothed importance sampling leave-one-out cross-validation (LOO-PIT) (Vehtari et al., 2016; Watanabe, 2010).

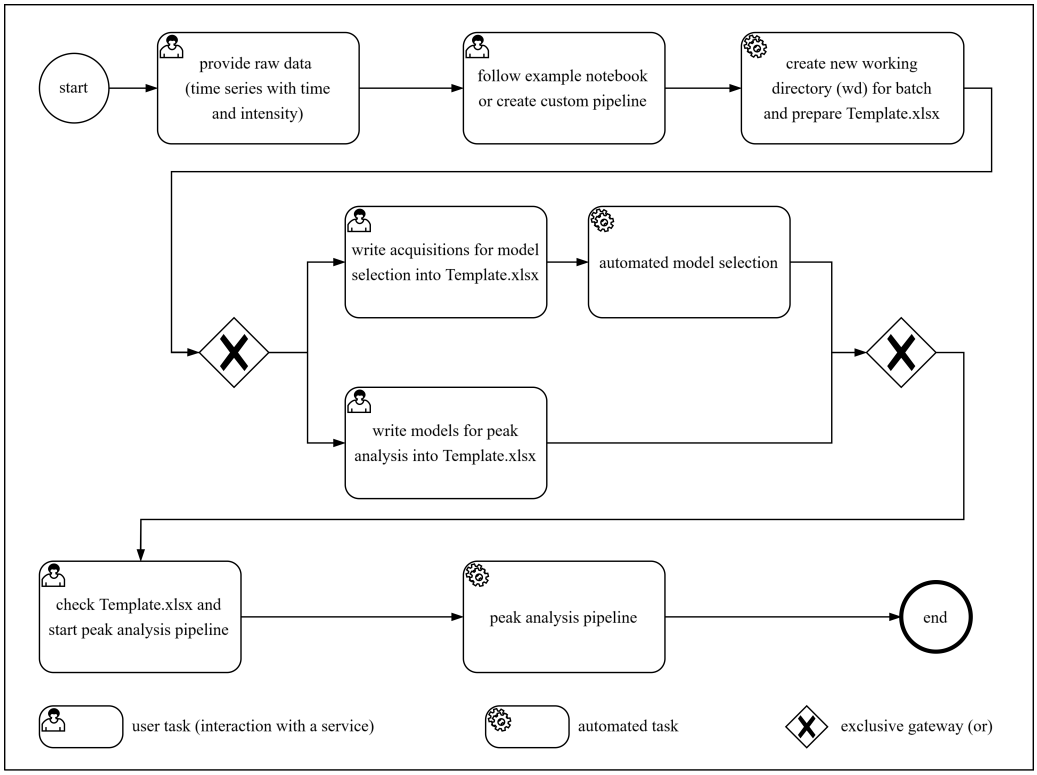

Figure 1: Overview of the pre-manufactured data analysis pipeline featured in `PeakPerformance`.

Subsequently, the peak analysis pipeline can be started with the function `pipeline()` from the `pipeline` module. Depending on whether the "pre-filtering" option was selected, an optional filtering step will be executed to reject signals where clearly no peak is present before sampling, thus saving computation time. Upon passing the first filter, a Markov chain Monte Carlo (MCMC) simulation is conducted using a No-U-Turn Sampler (NUTS) (Hoffmann & Gelman, 2014), preferably - if installed in the Python environment - the `nutpie` sampler (Seyboldt & PyMC Developers, 2022) due to its highly increased performance compared to the default sampler of PyMC. When a posterior distribution has been obtained, the main filtering step is next in line checking the convergence of the Markov chains via the potential scale reduction factor (Gelman & Rubin, 1992) or $\hat{R}$ statistic and based on the uncertainty of the determined peak parameters. If a signal was accepted as a peak, a posterior predictive check is conducted and added to the inference data object resulting from the model simulation. Regarding the performance of the simulation, in our tests an analysis of a single peak took 20 s to 30 s and of a double peak 25 s to 90 s. This is of course dependent on the type of sampler, the power of the computer as well as whether an additional simulation with an increased number of samples needs to be conducted.

Peak fitting results and diagnostic plots

The most complete report created after completing a cycle of the data pipeline is found in an Excel file called "peak_data_summary.xlsx". Here, each analyzed time series has multiple rows (one per peak parameter) with the columns containing estimation results in the form of mean and standard deviation (sd) of the marginal posterior distribution, highest density interval (HDI), and the $\hat{R}$ statistic among other metrics. The second Excel file created is denominated as "area_summary.xlsx" and is a more handy version of "peak_data_summary.xlsx" with a reduced degree of detail since subsequent data analyses will most likely rely on the peak area. The most valuable result, however, are the inference data objects saved to disk for each signal for which a peak function was successfully fitted. Conveniently, the inference data

objects saved as `*.nc` files contain all data and metadata related to the Bayesian parameter estimation, enabling the user to perform diagnostics or create custom visualizations not already provided by `PeakPerformance`. Regarding data visualization with the matplotlib package (Hunter, 2007; The Matplotlib Development Team, 2024), `PeakPerformance`'s `plots` module offers the generation of two diagram types for each successfully fitted peak. The posterior plot presents the fit of the intensity function alongside the raw data points. The first row of Figure 2 presents two such examples where the single peak diagram shows the histidine (His) fragment with a m/z ratio of 110 Da and the double peak diagram the leucine (Leu) and isoleucine (Ile) fragments with a m/z ratio of 86 Da.

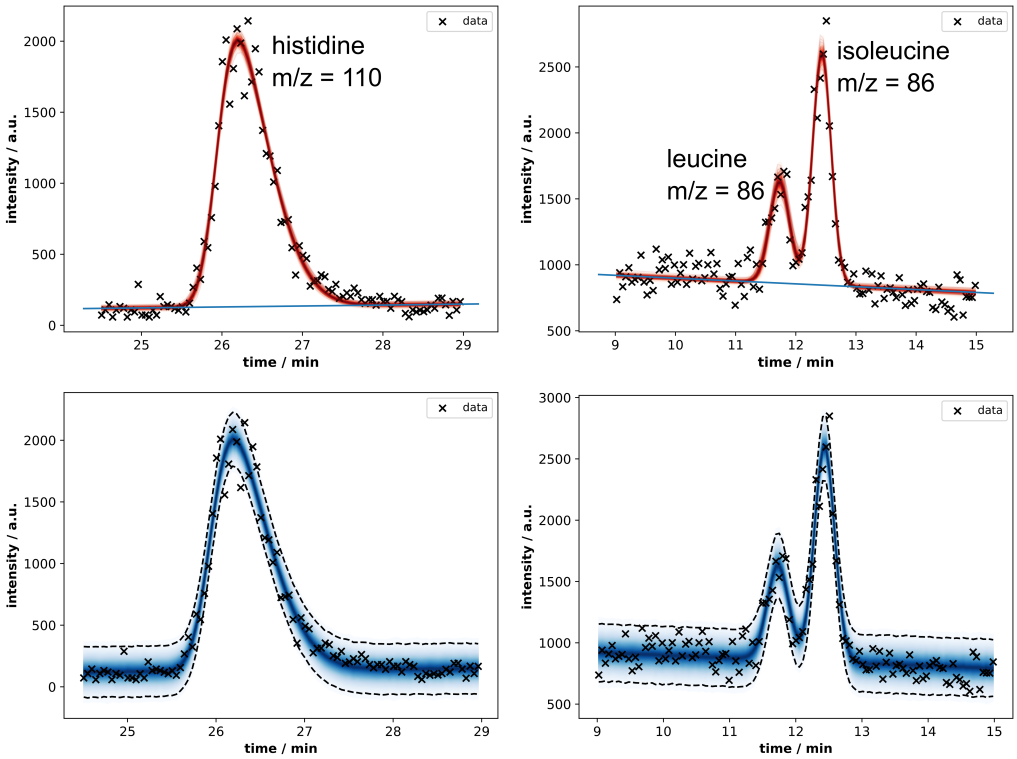

Figure 2: Results plots for a single His peak and a double Leu and Ile peak depicting the peak fit (first row) and the posterior predictive checks (second row) alongside the raw data. The numerical results are listed in Table 2.

The posterior predictive plots in the second row of Figure 4 are provided for visual posterior predictive checks, namely the comparison of observed and predicted data distribution. Since a posterior predictive check is based on drawing samples from the likelihood function, the result represents the theoretical range of values encompassed by the model. Accordingly, this plot enables users to judge whether the selected model can accurately explain the data. To complete the example, Table 2 shows the results of the fit in the form of mean, standard deviation, and HDI of each parameter's marginal posterior.

Table 2: Depiction of the results for the most important peak parameters of a single peak fit with the skew normal model and a double peak fit with the double normal model. Mean, area, and height have been highlighted in bold print as they constitute the most relevant parameters for further data evaluation purposes. The results correspond to the fits exhibited in Figure 2.

parameter	single skew normal model				double normal model			
	mean	sd	hdi_3%	hdi_97%	mean	sd	hdi_3%	hdi_97%
intercept	-43.94	7.41	-57.88	-30.02	1115.40	38.69	1040.14	1185.07
slope	6.66	0.51	5.71	7.63	-21.65	3.09	-27.50	-15.94
noise	103.63	7.51	89.50	117.26	118.63	8.01	103.52	133.29
mean	**25.95**	**0.01**	25.93	25.97	**11.73** **12.43**	**0.02** **0.01**	11.70 12.42	11.76 12.45
area	**1512.32**	**37.31**	1441.25	1581.37	**317.16** **674.34**	**28.84** **26.34**	263.23 623.47	370.56 722.88
height	**1879.72**	**37.71**	1809.30	1950.64	**774.99** **1762.66**	**65.28** **64.04**	653.50 1639.62	897.88 1881.66
std	0.53	0.02	0.48	0.56	0.16 0.15	0.02 0.01	0.13 0.14	0.20 0.17
sn	18.24	1.37	15.69	20.76	6.56 14.93	0.72 1.14	5.22 12.82	7.88 17.14
alpha	2.96	0.39	2.27	3.71	-	-	-	-

In this case, the fits were successful and convergence was reached for all parameters. Most notably and for the first time, the measurement noise was taken into account when determining the peak area as represented by its standard deviation and as can be observed in the posterior predictive plots where the noisy data points fall within the boundary of the 95 % HDI. In the documentation, there is a study featuring simulated and experimental data to validate PeakPerformance's results against a commercially available vendor software for peak integration showing that comparable results are indeed obtained.

Author contributions

PeakPerformance was conceptualized by JN and MO. Software implementation was conducted by JN with code review by MO. The original draft was written by JN with review and editing by MO, SN, EvL, and WW. The work was supervised by SN and funding was acquired by SN, EvL, and WW.

Acknowledgements

The authors thank Tobias Latour for providing experimental LC-MS/MS data for the comparison with the vendor software MultiQuant. Funding was received from the German Federal Ministry of Education and Research (BMBF) (grant number 031B1134A) as part of the innovation lab "AutoBioTech" within the project "Modellregion, BioRevierPLUS: BioökonomieREVIER Innovationscluster Biotechnologie & Kunststofftechnik".

Competing interests

No competing interest is declared.

Data availability

The datasets generated during and/or analysed during the current study are available in version 0.7.1 of the Zenodo record.

Bibliography

Abril-Pla, O., Andreani, V., Carroll, C., Dong, L., Fonnesbeck, C. J., Kochurov, M., Kumar, R., Lao, J., Luhmann, C. C., Martin, O. A., Osthege, M., Vieira, R., Wiecki, T., & Zinkov,

R. (2023). PyMC: A modern, and comprehensive probabilistic programming framework in Python. *PeerJ Computer Science*, *9*, e1516. https://doi.org/10.7717/peerj-cs.1516

Azzalini, A. (1985). A class of distributions which includes the normal ones. *Scandinavian Journal of Statistics*, *12*, 171–178. http://www.jstor.org/stable/4615982

Briskot, T., Stückler, F., Wittkopp, F., Williams, C., Yang, J., Konrad, S., Doninger, K., Griesbach, J., Bennecke, M., Hepbildikler, S., & others. (2019). Prediction uncertainty assessment of chromatography models using Bayesian inference. *Journal of Chromatography A*, *1587*, 101–110. https://doi.org/10.1016/j.chroma.2018.11.076

Gelman, A., & Rubin, D. B. (1992). Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, *7*(4). https://doi.org/10.1214/ss/1177011136

Hemmerich, J., Noack, S., Wiechert, W., & Oldiges, M. (2018). Microbioreactor Systems for Accelerated Bioprocess Development. *Biotechnology Journal*, *13*(4), e1700141. https://doi.org/10.1002/biot.201700141

Hoffmann, M. D., & Gelman, A. (2014). The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, *15*.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Kelly, P., & Harris, W. (1971a). Application of method of maximum posterior probability to estimation of gas-chromatographic peak parameters. *Analytical Chemistry*, *43*(10), 1184–1195. https://doi.org/10.1021/ac60304a005

Kelly, P., & Harris, W. (1971b). Estimation of chromatographic peaks with particular consideration of effects of base-line noise. *Analytical Chemistry*, *43*(10), 1170–1183. https://doi.org/10.1021/ac60304a011

Kostov, Y., Harms, P., Randers-Eichhorn, L., & Rao, G. (2001). Low-cost microbioreactor for high-throughput bioprocessing. *Biotechnology and Bioengineering*, *72*(3), 346–352. https://doi.org/10.1002/1097-0290(20010205)72:3%3C346::aid-bit12%3E3.0.co;2-x

Kumar, R., Carroll, C., Hartikainen, A., & Martin, O. (2019). ArviZ a unified library for exploratory analysis of Bayesian models in Python. *Journal of Open Source Software*, *4*(33). https://doi.org/10.21105/joss.01143

Seyboldt, A., & PyMC Developers. (2022). *nutpie*. https://github.com/pymc-devs/nutpie

The Matplotlib Development Team. (2024). *Matplotlib: Visualization with Python* (Version v3.9.0). Zenodo. https://doi.org/10.5281/zenodo.11201097

Vehtari, A., Gelman, A., & Gabry, J. (2016). Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and Computing*, *27*(5), 1413–1432. https://doi.org/10.1007/s11222-016-9696-4

Vivó-Truyols, G. (2012). Bayesian approach for peak detection in two-dimensional chromatography. *Analytical Chemistry*, *84*(6), 2622–2630. https://doi.org/10.1021/ac202124t

Watanabe, S. (2010). Asymptotic Equivalence of Bayes Cross Validation and Widely Applicable Information Criterion in Singular Learning Theory. *Journal of Machine Learning Research*, *11*, 3571–3594.

Wiczling, P., & Kaliszan, R. (2016). How much can we learn from a single chromatographic experiment? A Bayesian perspective. *Analytical Chemistry*, *88*(1), 997–1002. https://doi.org/10.1021/acs.analchem.5b03859

Woldegebriel, M., & Vivó-Truyols, G. (2015). Probabilistic model for untargeted peak detection in LC–MS using Bayesian statistics. *Analytical Chemistry*, *87*(14), 7345–7355. https://doi.org/10.1021/acs.analchem.5b01521

Yamamoto, Y., Yajima, T., & Kawajiri, Y. (2021). Uncertainty quantification for chromatography model parameters by Bayesian inference using sequential Monte Carlo method. *Chemical Engineering Research and Design*, *175*, 223–237. https://doi.org/10.1016/j.cherd.2021.09.003