chemotools: A Python Package that Integrates Chemometrics and scikit-learn

Pau Cabaneros Lopez [1]

1 Novo Nordisk A/S, Bagsvaerd, Denmark

Summary

`chemotools` stands as a production-oriented versatile Python library, developed to provide a unified platform for advancing *chemometric* model development. Integrating spectral preprocessing methodologies with the `scikit-learn` API and the expansive Python machine learning ecosystem, this library seeks to standardize and simplify the complex process of creating and implementing robust *chemometric* and machine learning models of spectral data.

Statement of need

Spectroscopy comprises a group of several analytical techniques used to understand the composition of materials using light. Traditionally, spectroscopic data is analyzed by a discipline called *chemometrics*, a branch of machine learning specialized on extracting chemical information from multivariate spectra. Over the last decades, *chemometricians*, have excelled by developing advanced preprocessing methods designed to attenuate instrument and measuring noise from the spectra, and to enhance the pure chemical information of the samples (Rinnan et al., 2009), (Mishra et al., 2020).

Spectroscopic methods are very suited for a wide range of applications because they allow analyzing the chemical properties of various samples in a fast and simple manner. For this reason, their adoption as integral components of Process Analytical Technology (PAT) has witnessed significant growth across industries, including chemical, biotech, food, and pharmaceuticals. Despite this surge, a notable obstacle has been the absence of open-source standardized, accessible toolkit for *chemometric* model development and deployment. `chemotools`, positioned as a comprehensive solution, addresses this void by integrating *chemometric* methods into the Python machine learning ecosystem. By implementing a variety of preprocessing and feature selection tools with the `scikit-learn` API (Pedregosa et al., 2018), `chemotools` opens up the entire `scikit-learn` toolbox to users, encompassing features such as:

- a rich collection of estimators for regression, classification, and clustering
- cross-validation and hyper-parameter optimization algorithms
- pipelining for efficient workflows
- and model persistence using standardized libraries such as `joblib` or `pickle`

This integration empowers users with a versatile array of tools for robust model development and evaluation (Figure 1).

In addition to its foundational capabilities, `chemotools` not only enables users to preprocess data and train models using `scikit-learn` but also streamlines the transition of these models into a production setting. By enabling users with a well defined interface, `chemotools` facilitates the reception of input data and delivery of predictions from the trained model. This can then be containerized using Docker, providing an efficient means for the distribution and implementation of the model in any Docker-compatible environment, facilitating the deployment of models to

cloud environments. This adaptive capability not only enables organizations to scale model usage but also allows them to monitor performance and promptly update or rollback the model as necessary.

chemotools also introduces a practical innovation by providing a streamlined framework for data augmentation of spectroscopic datasets through the scikit-learn API. This feature offers users a straightforward and consistent method to enhance spectral datasets, by introducing stochastic artifacts that represent real-world variations. By integrating data augmentation into the *chemometric* workflow, chemotools provides users with an efficient tool for improving their datasets to generalize the models and optimize their performance.
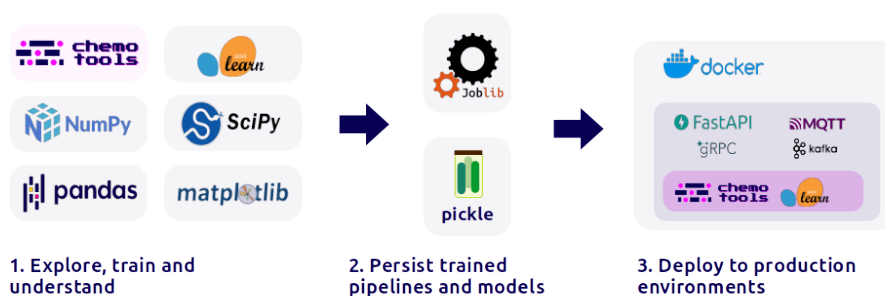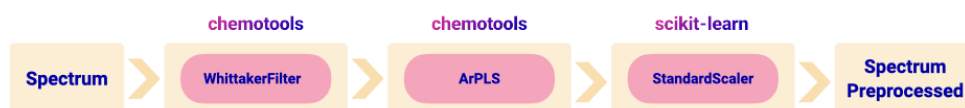

Figure 1: chemotools in the Python machine learning environment .

Features and functionality

chemotools implements a collection of scikit-learn transformers and selectors. Transformers are divided in preprocessing and augmentation methods. Preprocessing functions range from well-established *chemometric* methods such as the multiplicative scatter correction or the standard normal variate (Rinnan et al., 2009), to more recent methods such as the asymmetrically reweighted penalized least squares method to remove complex baselines (Baek et al., 2015). Several preprocessing methods can be conveniently concatenated using scikit-learn pipelines (Figure 2). An example of code used to create a preprocessing pipeline mixing scikit-learn and chemotools methods is shown in below:

Figure 2: Overview of the pipelines. A: Preprocessing pipeline. B: Augmentation pipeline.

Lopez. (2024). chemotools: A Python Package that Integrates Chemometrics and scikit-learn. *Journal of Open Source Software*, 9(100), 6802. 2
https://doi.org/10.21105/joss.06802.

```python
from chemotools.baseline import ArPls
from chemotools.smooth import WhittakerSmooth

from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler


pipeline = make_pipeline(
    WhittakerSmooth(),
    ArPls(),
    StandardScaler(with_std=False),
)
```

The data augmentation module contains transformers that introduce stochastic artifacts to the spectral data to reflect real-world variability (e.g. instrument-to-instrument variations). These include a variety of transformers ranging from adding noise to the spectra following a given distribution, to shifts on the spectral peaks or changes on the intensity of the peaks. Since the data augmentation functions are implemented as transformers, the user can leverage the pipelining functions of scikit-learn to concatenate different augmentation methods in pipelines to transform their data. An example of an augmentation pipeline is shown in Figure 2. An example of code to create an augmentation pipeline is shown below:

```python
from chemotools.augmentation import BaselineShift, IndexShift, NormalNoise
from sklearn.pipeline import make_pipeline

augmentation_pipeline = make_pipeline(
    NormalNoise(scale=0.001),
    BaselineShift(0.001),
    IndexShift(3),
)

spectra_augmented = np.array(
    [augmentation_pipeline.fit_transform(spectrum) for _ in range(5)]
)
```

Figure 3 shows five spectra augmented with the pipeline depicted in Figure 2 and the original spectrum.
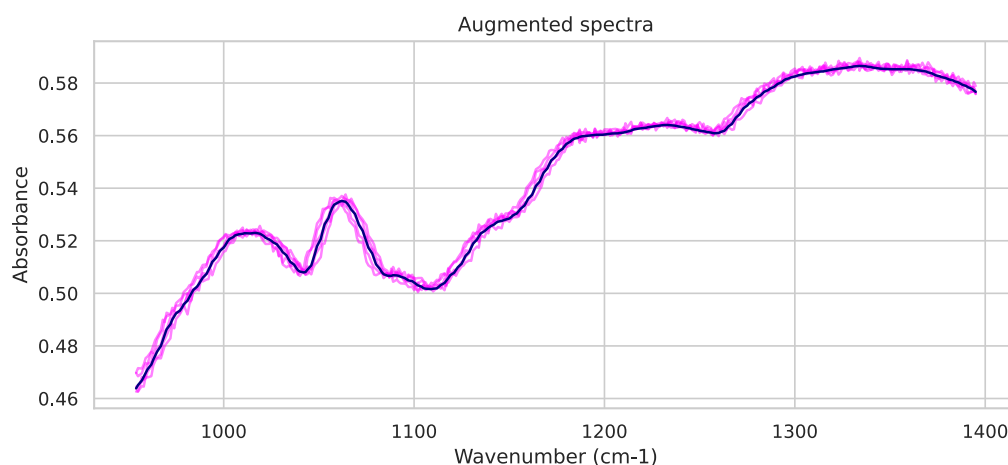

Figure 3: Spectral augmentation. Five augmented spectra (in magenta) are generated from an original spectrum (in blue) using an augmentation pipeline.

In addition to the transformers, `chemotools` also implements selectors. Selectors are mathematical functions used to select the relevant features from the spectral dataset based on a given criterion. Selectors are used to select the features that contain the chemical information of the sample, making the models more robust and generalizable.

Beyond its mathematical features, `chemotools` goes a step further by providing real-world spectral datasets (Cabaneros Lopez et al., 2021). Accompanied by guides demonstrating the integration of scikit-learn and `chemotools` for training regression and classification models, these datasets immerse learners in practical applications. This hands-on approach bridges theoretical concepts and real-world implementation, nurturing a deeper understanding of potential challenges in real-world scenarios.

The documentation page (https://paucablop.github.io/chemotools/) meticulously outlines all available mathematical functions within chemotools. This comprehensive resource serves as a guide for users exploring the extensive capabilities of the library.

Adoption and applications

The ultimate objective of developing *chemometric* and machine learning models is either to gain insights about complex datasets or to train models that can be used in production applications (Figure 4). From a research and development perspective, `chemotools` offers a wide range of transformers and selectors that, combined with the rest of the Python machine learning environment, enables researchers to investigate and understand their spectral datasets. From an industrial point of view, `chemotools` allows users to streamline the deployment of their trained models into production environments adhering to standard frameworks developed by the machine learning community in Python (Figure 1).

Beyond its practical applications, `chemotools` has been utilized as an educational tool at universities for both Master's (MSc) and Doctoral (PhD) levels. Integrating chemotools into academic curricula using Jupyter notebooks, offers students a valuable opportunity to gain hands-on experience with real-world datasets, providing practical insights into the application of sophisticated techniques for preprocessing and analyzing spectral data. The tool's user friendly interface, coupled with comprehensive documentation, has proven an enriching learning experience for students pursuing education in fields relate to analytical chemistry, process analytical technology, data science or *chemometrics*.
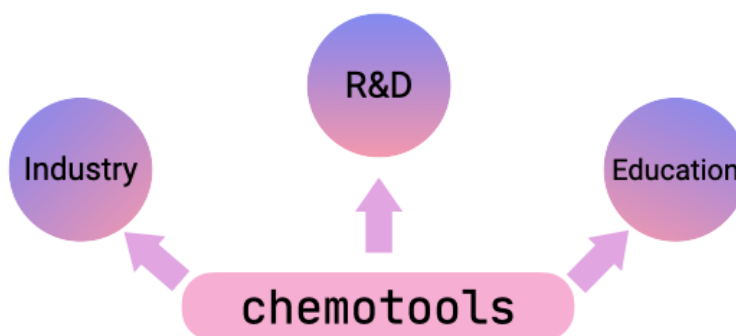

Figure 4: Applications of `chemotools`.

Author contribution statement

Conceptualization, coding, developing and paper writing by Pau Cabaneros Lopez.

References

Baek, S.-J., Park, A., Ahn, Y.-J., & Choo, J. (2015). Baseline correction using asymmetrically reweighted penalized least squares smoothing. *Analyst*, *140*(1), 250–257. https://doi.org/10.1039/c4an01061b

Cabaneros Lopez, P., Udugama, I. A., Thomsen, S. T., Roslander, C., Junicke, H., Iglesias, M. M., & Gernaey, K. V. (2021). Transforming data to information: A parallel hybrid model for real-time state estimation in lignocellulosic ethanol fermentation. *Biotechnology and Bioengineering*, *118*(2), 579–591. https://doi.org/10.1002/bit.27586

Mishra, P., Biancolillo, A., Roger, J. M., Marini, F., & Rutledge, D. N. (2020). New data preprocessing trends based on ensemble of multiple preprocessing techniques. *TrAC Trends in Analytical Chemistry*, *132*, 116045. https://doi.org/10.1016/j.trac.2020.116045

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Müller, A., Nothman, J., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2018). *Scikit-learn: Machine learning in python*. https://arxiv.org/abs/1201.0490

Rinnan, Å., Berg, F. van den, & Engelsen, S. B. (2009). Review of the most common pre-processing techniques for near-infrared spectra. *TrAC Trends in Analytical Chemistry*, *28*(10), 1201–1222. https://doi.org/10.1016/j.trac.2009.07.007