mocca Documentation

Release 0.1.2.post1.dev9+g6db1627

HaasCP

Mar 03, 2023

CONTENTS

1 Contents **3**
- 1.1 Installation . . . 4
- 1.2 Getting started . . . 4
- 1.3 How to cite . . . 4
- 1.4 Note . . . 5
- 1.5 Contributing . . . 5
- 1.6 License . . . 9
- 1.7 Contributors . . . 9
- 1.8 Changelog . . . 9
- 1.9 mocca . . . 9

2 Indices and tables **39**

Python Module Index **41**

Index **43**

This is the documentation of **MOCCA** (Multivariate Online Contextual Chromatographic Analysis).

CHAPTER

ONE

CONTENTS

MOCCA (Multivariate Online Contextual Chromatographic Analysis) is an open-source Python project to analyze HPLC–DAD raw data.

Automation and digitalization solutions in the field of small molecule synthesis face new challenges for chemical reaction analysis, especially in the field of high-performance liquid chromatography (HPLC). Chromatographic data remains locked in vendors' hardware and software components limiting their potential in automated workflows and contradicting to FAIR data principles (findability, accessibility, interoperability, reuse), which enable chemometrics and data science applications. In this work, we present an open-source Python project called MOCCA (Multivariate Online Contextual Chromatographic Analysis) for the analysis of open-format HPLC–DAD (photodiode array detector) raw data. MOCCA provides a comprehensive set of data analysis features including a peak deconvolution routine which allows for automated deconvolution of known signals even if overlapped with signals of unexpected impurities or side products. By publishing MOCCA as a Python package, we envision an open-source community project for chromatographic data analysis with the potential of further advancing its scope and capabilities.

Open-source project: https://github.com/HaasCP/mocca

Documentation: https://mocca.readthedocs.io/en/latest/

Corresponding scientific publication (open access): *ACS Central Science*, **2023**, https://doi.org/10.1021/acscentsci.2c01042.

1.1 Installation

1. We recommend creating an isolated conda environment to avoid any problems with your installed Python packages:

```
conda create -n mocca python=3.9
conda activate mocca
```

2. Install `mocca` and its dependencies:

```
pip install mocca
```

3. If you want to use `mocca`'s reporting functionality:

```
pip3 install -U datapane==0.14
```

4. If you want to use Allotrope (adf) file format:

```
pip install h5py
pip install git+https://github.com/HDFGroup/h5ld@master
```

5. If you want to use `mocca` using JupyterLab notebooks:

```
pip install jupyterlab
ipython kernel install --user --name=mocca
```

1.2 Getting started

MOCCA is currently best used via JupyterLab notebooks. The notebooks folder of the GitHub repository contains a tutorial notebook with corresponding HPLC–DAD test data for the first steps.

Additionally, a full test data set from the scientific publication is added (cyanation of aryl halides via well plate screening). The corresponding notebook contains full data analysis details from the raw data level until the presented visualizations in the manuscript (Fig. 7e) and SI (Fig. S17).

1.3 How to cite

Peer-reviewed, open access:

Haas, C. P., Lübbesmeyer, M., Jin, E. H., McDonald, M. A., Koscher, B. A., Guimond, N., Di Rocco, L., Kayser, H., Leweke, S., Niedenführ, S., Nicholls, R., Greeves, E., Barber, D. M., Hillenbrand, J., Volpin, G., and Jensen, K. F. Open-Source Chromatographic Data Analysis for Reaction Optimization and Screening. *ACS Cent.Sci.* **2023**. https://doi.org/10.1021/acscentsci.2c01042.

Preprint:

Haas, C. P., Lübbesmeyer, M., Jin, E. H., McDonald, M. A., Koscher, B. A., Guimond, N., Di Rocco, L., Kayser, H., Leweke, S., Niedenführ, S., Nicholls, R., Greeves, E., Barber, D. M., Hillenbrand, J., Volpin, G., and Jensen, K. F. Open-Source Chromatographic Data Analysis for Reaction Optimization and Screening. *ChemRxiv* **2022**. https://doi.org/10.26434/chemrxiv-2022-0pv2d.

1.4 Note

This project has been set up using PyScaffold 4.1.1. For details and usage information on PyScaffold see https://pyscaffold.org/.

1.5 Contributing

Welcome to `mocca` contributor's guide.

This document focuses on getting any potential contributor familiarized with the development processes, but other kinds of contributions are also appreciated.

If you are new to using git or have never collaborated in a project previously, please have a look at contribution-guide.org. Other resources are also listed in the excellent guide created by FreeCodeCamp[1].

Please notice, all users and contributors are expected to be **open, considerate, reasonable, and respectful**. When in doubt, Python Software Foundation's Code of Conduct is a good reference in terms of behavior guidelines.

1.5.1 Issue Reports

If you experience bugs or general issues with `mocca`, please have a look on the issue tracker. If you don't see anything useful there, please feel free to fire an issue report.

Tip: Please don't forget to include the closed issues in your search. Sometimes a solution was already reported, and the problem is considered **solved**.

New issue reports should include information about your programming environment (e.g., operating system, Python version) and steps to reproduce the problem. Please try also to simplify the reproduction steps to a very minimal example that still illustrates the problem you are facing. By removing other factors, you help us to identify the root cause of the issue.

1.5.2 Documentation Improvements

You can help improve `mocca` docs by making them more readable and coherent, or by adding missing information and correcting mistakes.

`mocca` documentation uses Sphinx as its main documentation compiler. This means that the docs are kept in the same repository as the project code, and that any documentation update is done in the same way was a code contribution.

Tip: Please notice that the GitHub web interface provides a quick way of propose changes in `mocca`'s files. While this mechanism can be tricky for normal code contributions, it works perfectly fine for contributing to the docs, and can be quite handy.

If you are interested in trying this method out, please navigate to the `docs` folder in the source repository, find which file you would like to propose changes and click in the little pencil icon at the top, to open GitHub's code editor. Once you finish editing the file, please write a message in the form at the bottom of the page describing which changes have you made and what are the motivations behind them and submit your proposal.

---

[1] Even though, these resources focus on open source projects and communities, the general ideas behind collaborating with other developers to collectively create software are general and can be applied to all sorts of environments, including private companies and proprietary code bases.

When working on documentation changes in your local machine, you can compile them using `tox`:

```
tox -e docs
```

and use Python's built-in web server for a preview in your web browser (`http://localhost:8000`):

```
python3 -m http.server --directory 'docs/_build/html'
```

1.5.3 Code Contributions

Submit an issue

Before you work on any non-trivial code contribution it's best to first create a report in the issue tracker to start a discussion on the subject. This often provides additional considerations and avoids unnecessary work.

Create an environment

Before you start coding, we recommend creating an isolated virtual environment to avoid any problems with your installed Python packages. A suggested procedure is described in the installation guide of `mocca` or this can easily be done via either `virtualenv`:

```
virtualenv <PATH TO VENV>
source <PATH TO VENV>/bin/activate
```

or Miniconda:

```
conda create -n mocca python=3 six virtualenv pytest pytest-cov
conda activate mocca
```

Clone the repository

1. Create an user account on GitHub if you do not already have one.
2. Fork the project repository: click on the *Fork* button near the top of the page. This creates a copy of the code under your account on GitHub.
3. Clone this copy to your local disk:

   ```
   git clone git@github.com:YourLogin/mocca.git
   cd mocca
   ```

4. You should run:

   ```
   pip install -U pip setuptools -e .
   ```

   to be able run `putup --help`.

Implement your changes

1. Create a branch to hold your changes:

   ```
   git checkout -b my-feature
   ```

   and start making changes. Never work on the master branch!

2. Start your work on this branch. Don't forget to add docstrings to new functions, modules and classes, especially if they are part of public APIs.

3. Add yourself to the list of contributors in `AUTHORS.rst`.

4. When you're done editing, do:

   ```
   git add <MODIFIED FILES>
   git commit
   ```

   to record your changes in git.

   Please make sure to see the validation messages from `pre-commit` and fix any eventual issues. This should automatically use flake8/black to check/fix the code style in a way that is compatible with the project.

   Important: Don't forget to add unit tests and documentation in case your contribution adds an additional feature and is not just a bugfix.

   Moreover, writing a descriptive commit message is highly recommended. In case of doubt, you can check the commit history with:

   ```
   git log --graph --decorate --pretty=oneline --abbrev-commit --all
   ```

   to look for recurring communication patterns.

   ---

5. Please check that your changes don't break any unit tests with:

   ```
   tox
   ```

   (after having installed `tox` with `pip install tox` or `pipx`).

   You can also use `tox` to run several other pre-configured tasks in the repository. Try `tox -av` to see a list of the available checks.

Submit your contribution

1. If everything works fine, push your local branch to GitHub with:

   ```
   git push -u origin my-feature
   ```

2. Go to the web page of your fork and click "Create pull request" to send your changes for review.

Troubleshooting

The following tips can be used when facing problems to build or test the package:

1. Make sure to fetch all the tags from the upstream repository. The command `git describe --abbrev=0 --tags` should return the version you are expecting. If you are trying to run CI scripts in a fork repository, make sure to push all the tags. You can also try to remove all the egg files or the complete egg folder, i.e., `.eggs`, as well as the `*.egg-info` folders in the `src` folder or potentially in the root of your project.
2. Sometimes `tox` misses out when new dependencies are added, especially to `setup.cfg` and `docs/requirements.txt`. If you find any problems with missing dependencies when running a command with `tox`, try to recreate the `tox` environment using the `-r` flag. For example, instead of:

   ```
   tox -e docs
   ```

   Try running:

   ```
   tox -r -e docs
   ```

3. Make sure to have a reliable `tox` installation that uses the correct Python version (e.g., 3.7+). When in doubt you can run:

   ```
   tox --version
   # OR
   which tox
   ```

   If you have trouble and are seeing weird errors upon running `tox`, you can also try to create a dedicated virtual environment with a `tox` binary freshly installed. For example:

   ```
   virtualenv .venv
   source .venv/bin/activate
   .venv/bin/pip install tox
   .venv/bin/tox -e all
   ```

4. Pytest can drop you in an interactive session in the case an error occurs. In order to do that you need to pass a `--pdb` option (for example by running `tox -- -k <NAME OF THE FALLING TEST> --pdb`). You can also setup breakpoints manually instead of using the `--pdb` option.

1.5.4 Maintainer tasks

Releases

If you are part of the group of maintainers and have correct user permissions on PyPI, the following steps can be used to release a new version for `mocca`:

1. Make sure all unit tests are successful.
2. Tag the current commit on the main branch with a release tag, e.g., `v1.2.3`.
3. Push the new tag to the upstream repository, e.g., `git push upstream v1.2.3`
4. Clean up the `dist` and `build` folders with `tox -e clean` (or `rm -rf dist build`) to avoid confusion with old builds and Sphinx docs.
5. Run `tox -e build` and check that the files in `dist` have the correct version (no `.dirty` or git hash) according to the git tag. Also check the sizes of the distributions, if they are too big (e.g., > 500KB), unwanted clutter may have been accidentally included.

6. Run `tox -e publish -- --repository pypi` and check that everything was uploaded to PyPI correctly.

1.6 License

1.7 Contributors

- Christian P. Haas <57134208+HaasCP@users.noreply.github.com>
- Edward H. Jin

1.8 Changelog

1.8.1 Version 0.0.0

- Initial publication

1.9 mocca

1.9.1 mocca package

Subpackages

mocca.campaign package

Submodules

mocca.campaign.experiment_funcs module

Created on Tue Jan 25 09:11:54 2022

@author: haascp

mocca.campaign.experiment_funcs.**get_sorted_compound_experiments**(*experiments*)

Filters experiments for experiments with given compound. Sorts these experiments in the order: 1. solvent runs, 2. istd runs, 3. compound runs (sorted reversely by the compound concentration). In these categories, experiments are sorted in the order the user has given.

mocca.campaign.experiment_funcs.**get_unprocessed_experiments**(*experiments*, *quali_comp_db=None*)

Returns all experiments which have not been processed yet. Checks for internal standard condition, ie, that any given istd given in unprocessed peaks is already in the qualitative component db so that a corresponding peak can be found in the chromatogram.

mocca.campaign.process_funcs module

Created on Wed Dec 22 10:43:13 2021

@author: haascp

mocca.campaign.process_funcs.**preprocess_experiment**(*exp*, *quali_comp_db*, *settings*)

Returns a chromatogram object created out of the given experiment. The peaks in the chromatogram already have assigned possible matches but they are not yet assigned or quantified.

mocca.campaign.process_funcs.**process_compound_exp**(*exp*, *quali_comp_db*, *settings*)

Processes one compound experiment.

mocca.campaign.process_funcs.**process_compound_experiments**(*experiments*, *peak_db*, *quali_comp_db*, *quant_comp_db*, *settings*)

Sorts and processes all compound experiments (experiments from which the program learns). Updates both qualitative and quantitative component databases.

mocca.campaign.process_funcs.**process_experiments**(*experiments*, *peak_db*, *quali_comp_db*, *quant_comp_db*, *settings*)

Processes all unprocessed experiments (not compound experiments) which should be analyzed by the program.

mocca.campaign.process_funcs.**process_gradients**(*experiments*, *settings*)

Reads and processes gradient data for each experiment. Avoids double processing.

mocca.campaign.utils module

Created on Mon Dec 13 15:47:14 2021

@author: haascp

mocca.campaign.utils.**check_istd**(*exp*, *chrom*)

Checks internal standard condition, ie, if the user gives an istd information in the experiment, a corresponding peak has to be found in the chromatogram.

Module contents

mocca.chromatogram package

Submodules

mocca.chromatogram.assign module

Created on Fri Dec 17 18:55:41 2021

@author: haascp

mocca.chromatogram.assign.**assign_best_match_peak**(*peaks*)

Assigns the peak with the best correlation coefficient with the compound id and updates all remaining peaks accordingly.

mocca.chromatogram.assign.**assign_matched_peaks**(*peaks*, *assigned_peaks=[]*)

Assigns peaks containing matches with compound ids. In the rare case, that some peaks will not contain matches anymore, these are given back as unassigned and unmatched peaks.

mocca.chromatogram.assign.**assign_peaks_compound**(*chromatogram*, *compound*)

Assigns all matched peaks with compound_ids.

mocca.chromatogram.assign.**assign_peaks_react**(*chromatogram*, *peak_db*)

Assigns peaks of reaction runs with compound ids using unknown compound ids for unmatched peaks.

mocca.chromatogram.assign.**assign_unmatched_peaks_compound**(*peaks*, *compound_id*, *impurity_counter=0*)

Assigns peaks which do not contain matches with unknown compound ids.

mocca.chromatogram.assign.**assign_unmatched_peaks_react**(*peaks*, *peak_db*)

Assigns peaks which do not contain matches with unknown compound ids.

mocca.chromatogram.assign.**get_best_match_compound_id**(*peak*)

Returns the compound id of the best match of the given peak.

mocca.chromatogram.assign.**get_matched_peaks**(*peaks*)

Returns all peaks which have possible matches.

mocca.chromatogram.assign.**get_max_integral_peak**(*peaks*)

Returns the peak with the maximum integral value in the given list of peaks.

mocca.chromatogram.assign.**get_next_unknown_id**(*peak_db*)

Returns the next unknown compound_id.

mocca.chromatogram.assign.**get_unknown_impurity_peaks**(*assigned_peaks*)

Returns all peaks which are a compound impurity or are unknown.

mocca.chromatogram.assign.**get_unmatched_peaks**(*peaks*)

Returns all peaks which do not have possible matches.

mocca.chromatogram.assign.**reassign_impurities**(*chromatogram*, *peak_db*, *quali_comp_db*, *spectrum_correl_coef_thresh*, *relative_distance_thresh*, *print_similarity_dicts=False*)

This function is only allowed to be run in the process_all_experiments function which has to be run everytime a new compound should be added to quali_comp_db.

mocca.chromatogram.assign.**sort_peaks_by_best_match**(*peaks*)

Sorts peaks by descending spectrum correlation coefficient in their matches.

mocca.chromatogram.assign.**update_peaks_and_matches**(*sorted_peaks*)

Triggered after peak assignment. Deletes the peak which was assigned and removes the consumed compound id from the matches of all remaining peaks.

mocca.chromatogram.correct module

Created on Fri Dec 10 10:54:24 2021

@author: haascp

mocca.chromatogram.correct.**correct_istd_offset**(*chromatogram*, *quali_component_db*, *absorbance_threshold*, *spectrum_correl_coef_thresh*, *relative_distance_thresh*)

Corrects the peaks of the chromatogram by the average of the internal standard offsets. Adds the offset to the peak objects.

mocca.chromatogram.correct.**get_impure_istd_peak**(*chromatogram*, *istd_key*, *quali_comp_db*, *absorbance_threshold*, *spectrum_correl_coef_thresh*, *relative_distance_thresh*)

not doubled relative distance threshold

mocca.chromatogram.correct.**get_istd_offset**(*istd_peak*, *istd_key*, *quali_component_db*)

Finds possible istd peak in the chromatogram and calculates the retention time offset of the peak compared to its qualitative component in the database.

mocca.chromatogram.correct.**get_istd_peak**(*chromatogram*, *istd_key*, *quali_component_db*, *absorbance_threshold*, *spectrum_correl_coef_thresh*, *relative_distance_thresh*)

Tries to find an istd peak in the chromatogram from both pure or impure peaks.

mocca.chromatogram.correct.**get_pure_istd_peak**(*chromatogram*, *istd_key*, *quali_component_db*, *spectrum_correl_coef_thresh*, *relative_distance_thresh*)

not doubled relative distance threshold

mocca.chromatogram.model module

Created on Tue Dec 7 13:22:39 2021

@author: haascp

class mocca.chromatogram.model.**Chromatogram**(*experiment*, *dataset*)

Bases: `object`

Class containing data regarding chromatograms.

insert_peak(*peak*)

Adds a peak to the chromatogram.

mocca.chromatogram.preprocessor module

Created on Tue Dec 14 16:08:40 2021

@author: haascp

mocca.chromatogram.preprocessor.**preprocess_chromatogram**(*chromatogram*, *quali_comp_db*, *absorbance_threshold*, *detector_limit*, *spectrum_correl_thresh*, *relative_distance_thresh*, *print_purity_check=False*, *print_compound_prediction=False*, *print_parafac_analytics=False*)

Preprocesses the chromatogram of picked peaks. It includes expanding, checking, integrating, correcting, resolving impures, and matching of the peaks in the chromatogram.

mocca.chromatogram.quantify module

Created on Tue Jan 4 16:41:38 2022

@author: haascp

mocca.chromatogram.quantify.**quantify_peaks**(*chrom*, *quant_comp_db*, *quali_comp_db*)

Quantifies all peaks in the chromatogram.

mocca.chromatogram.utils module

Created on Tue Dec 7 13:26:09 2021

@author: haascp

mocca.chromatogram.utils.**check_overlap**(*peak*, *other*)

Returns True if peak overlaps with the peak 'other', and False otherwise.

mocca.chromatogram.utils.**check_same_dataset**(*peak*, *other*)

Raises Exception if the two peaks are not from the same dataset.

mocca.chromatogram.utils.**get_distance_between**(*peak*, *other*)

Returns the distance from the maxima of peak and the other peak.

Module contents

mocca.components package

Submodules

mocca.components.databases module

Created on Thu Dec 2 16:32:36 2021

@author: haascp

class mocca.components.databases.**BaseDatabase**

Bases: object

Base class for component databases with the unique constrained primary key compound_id.

delete_all_items()

Clears all components out of database.

insert_item(*item*, *unique=True*)

Inserts a new item to the database. If the unique argument is True, it checks if the item already exists in the databse.

class mocca.components.databases.**QualiComponentDatabase**

Bases: *BaseDatabase*

Database storing and processing qualitative components used for peak assignment.

insert_by_compound_id(*peak_database*, *compound_id*, *peak_filter_function=None*)

Inserts component in existing component list. If component with given compound_id already exists, it will be overwritten.

update(*peak_database*, *peak_filter_function=None*)

Creates components from the given peak database. Optionally, a condition can be given to filter peaks.

class mocca.components.databases.**QuantComponentDatabase**

Bases: *BaseDatabase*

Database storing and processing quantitative components used for peak quantification.

update(*peak_database*, *quali_comp_db*, *peak_filter_function=None*)

Creates components from the given peak database. Optionally, a condition can be given to filter peaks.

mocca.components.models module

class mocca.components.models.**QualiComponent**(*compound_id: str*, *left: int*, *right: int*, *maximum: int*, *offset: int*, *spectrum: list*, *spectrum_max: list*, *created_from: List[*ProcessedPeak*]*)

Bases: object

Class of a qualitative component created from a number of peaks of the same compound_id.

compound_id: *str*

created_from: *List[ProcessedPeak]*

left: *int*

maximum: *int*

offset: *int*

right: *int*

spectrum: *list*

spectrum_max: *list*

class mocca.components.models.**QuantComponent**(*compound_id: str*, *integrate_wl_idx: int*, *calib_factors: dict*, *calib_data: dict*, *calib_scores: dict*, *created_from: List[*ProcessedPeak*]*)

Bases: object

Class of a quantitative component created from a number of peaks of the same compound_id.

calib_data: dict

calib_factors: dict

calib_scores: dict

compound_id: str

created_from: List[*ProcessedPeak*]

integrate_wl_idx: int

mocca.components.quali_funcs module

Created on Fri Dec 3 09:54:04 2021

@author: haascp

mocca.components.quali_funcs.**create_quali_component**(*peaks*)

Creates a qualitative component object based on the given peaks.

mocca.components.quali_funcs.**get_absorbance_maxima**(*spectrum*)

Returns absorbance maxima of given spectrum. Maximum must be at least 5% intensity of the overall maximum intensity.

mocca.components.quant_funcs module

Created on Mon Dec 20 17:20:35 2021

@author: haascp

mocca.components.quant_funcs.**create_calibration_dict**(*peaks*, *integrate_wl_idx*, *quali_comp_db*)

Creates a dictionary with all data needed to create calibration curves.

mocca.components.quant_funcs.**create_linear_models**(*calib_data*)

Creates linear models out of the given data and returns a list of linear calibration factors (calibration curve is forced through origin) and a list of corresponing R-squared values of the regression.

mocca.components.quant_funcs.**create_quant_component**(*peaks*, *quali_comp_db*)

Creates a quantitative component object based on the given peaks

mocca.components.quant_funcs.**get_integrate_wl_index**(*compound_id*, *quali_comp_db*)

Returns index of the wavelength vector where spectrum of component has the highest maximum.

mocca.components.quant_funcs.**integrate_on_wl**(*peak*, *integrate_wl_idx*, *bandwidth=2*)

Integrates signal of given peak on a given wavelength with a bandwidth of 2 (default) by summing all absorbances.

mocca.components.utils module

Created on Fri Dec 3 13:23:45 2021

@author: haascp

mocca.components.utils.**average_ret_times_over_peaks**(*peaks*)

Calculates mean retention indices of a list of peaks.

mocca.components.utils.**average_spectra_over_peaks**(*peaks*)

Calculates mean spectrum of a list of peaks with averaged spectrum.

mocca.components.utils.**check_peaks_compound_id**(*peaks*)

Checks if all given peaks have the same compound_id and, if so, returns this compound_id.

mocca.components.utils.**filter_peaks**(*peaks*, *filter_function*)

Filters given peaks with regard to the given filter function (which takes a list of peaks and returns a filtered list of peaks).

mocca.components.utils.**get_filtered_peaks**(*peak_database*, *filter_function*)

Returns a filtered (by the given filter function) list of peaks from the database which are pure and unsaturated and have a compound_id.

mocca.components.utils.**get_filtered_peaks_by_compound**(*peak_database*, *filter_function*)

Creates a filtered (by the given filter function) list of peaks from the database which are pure and unsaturated and have a compound_id. From this list, it returns a dict with unique compound_id as keys and a list of corresponding peaks as values.

mocca.components.utils.**get_quant_peaks_by_compound**(*peak_database*, *filter_function*)

Returns a dict with compound_ids as keys and lists of peaks as values, where only peaks are includud which have is_compound True and which have a given concentration.

mocca.components.utils.**get_valid_peaks**(*peaks*)

Returns a list of peaks from the database which are pure and unsaturated and have a compound_id.

mocca.components.utils.**sort_peaks_by_compound**(*peaks*)

Returns dict with unique compound_id as keys and a list of corresponding peaks as values.

Module contents

mocca.dad_data package

Subpackages

mocca.dad_data.apis package

Submodules

mocca.dad_data.apis.allotrope module

Created on Thu May 12 09:10:57 2022

@author: CPH

mocca.dad_data.apis.allotrope.**get_function_paramenters**(*path*)

Reads the parameters of the linear function which describes the wavelength vector out of the data description layer.

mocca.dad_data.apis.allotrope.**get_uvvis_dataset_name**(*path*)

Queries the data description layer of the adf file to find the name of the dataset which is of the type 'three-dimensional ultraviolet spectrum' as defined by the AFO.

mocca.dad_data.apis.allotrope.**preprocess_df**(*df*)

Preprocesses the df time column to be in line with the Chemstation API.

mocca.dad_data.apis.allotrope.**read_adf**(*path*, *wl_high_pass=None*, *wl_low_pass=None*)

Reads adf files as exported by the Agilent ADF Adapter.

mocca.dad_data.apis.allotrope.**read_adf_datacube**(*path*)

Reads the raw data stored in the data cube layer, which are the HPLC-DAD absorbance values and the time scale.

mocca.dad_data.apis.allotrope.**read_adf_description**(*path*, *wl_len*)

Queries the adf data description layer to extract the wavlength vector. For this query, the h5ld package is required which can be installed by editable pip install from https://github.com/laura-dirocco/h5ld. In case there are problems with installation, the user can give start and stop values as set on the DAD manually.

mocca.dad_data.apis.angi module

Created on 06/13/22

@author: haascp

mocca.dad_data.apis.angi.**read_angi**(*path*, *wl_high_pass=None*, *wl_low_pass=None*)

Chemstation read and processing function.

mocca.dad_data.apis.angi.**read_csv_angi**(*path*)

Reads the UTF-16 encoded 3D data exported by the ChemStation macro. :param path: The directory, in which the experimental data are stored. :type path: str

Returns

df – First column is time, the following columns obtain the absorbance values at the given detection wavelength in the column name.

Return type

pandas.DataFrame

mocca.dad_data.apis.angi.**tidy_df_agilent**(*dataframe*, *wl_high_pass=None*, *wl_low_pass=None*)

Tidies the raw data obtained from reading the CSV

Parameters

dataframe (*pandas.DataFrame*) – First column is time, the following columns obtain the absorbance values at the given detection wavelength in the column name.

Raises

ValueError – If acquisition rate of the DAD was not constant, this error is raised.

Returns

df –

Columns:

time: Chromatogram time wavelength: Detection wavelength absorbance: Absorbance value

Return type
pandas.DataFrame

mocca.dad_data.apis.chemstation module

Created on Wed Aug 4 15:28:24 2021

@author: haascp

mocca.dad_data.apis.chemstation.**read_chemstation**(*path*, *wl_high_pass=None*, *wl_low_pass=None*)

Chemstation read and processing function.

mocca.dad_data.apis.chemstation.**read_csv_agilent**(*path*)

Reads the UTF-16 encoded 3D data exported by the ChemStation macro. :param path: The directory, in which the experimental data are stored. :type path: str

Returns
df – First column is time, the following columns obtain the absorbance values at the given detection wavelength in the column name.

Return type
pandas.DataFrame

mocca.dad_data.apis.chemstation.**tidy_df_agilent**(*dataframe*, *wl_high_pass=None*, *wl_low_pass=None*)

Tidies the raw data obtained from reading the CSV

Parameters
dataframe (*pandas.DataFrame*) – First column is time, the following columns obtain the absorbance values at the given detection wavelength in the column name.

Raises
ValueError – If acquisition rate of the DAD was not constant, this error is raised.

Returns

df –

Columns:
time: Chromatogram time wavelength: Detection wavelength absorbance: Absorbance value

Return type
pandas.DataFrame

mocca.dad_data.apis.custom module

Created on Mon Aug 30 15:17:53 2021

@author: haascp

mocca.dad_data.apis.custom.**read_custom_data**(*experiment*)

Returns the given custom data without any preprocessing

mocca.dad_data.apis.empower module

Created on Fri Mar 25 14:53:54 2022

@author: haascp

For more information on how to export raw data out of Empower see https://support.waters.com/KB_Inf/Empower_Breeze/WKB77571_How_to_export_3D_raw _data_from_Empower_to_a_Microsoft_Excel_spreadsheet

mocca.dad_data.apis.empower.**read_arw_empower**(*path*, *wl_high_pass=None*, *wl_low_pass=None*)

mocca.dad_data.apis.empower.**read_empower**(*path*, *wl_high_pass=None*, *wl_low_pass=None*)

Labsolutions read and processing function.

mocca.dad_data.apis.labsolutions module

Created on Mon Aug 30 15:17:53 2021

@author: haascp

mocca.dad_data.apis.labsolutions.**read_labsolutions**(*path*, *wl_high_pass=None*, *wl_low_pass=None*)

Labsolutions read and processing function.

mocca.dad_data.apis.labsolutions.**read_txt_shimadzu**(*path*)

Reads the 3D data exported by the LabSolutions software. :param path: The directory, in which the experimental data are stored. :type path: str

Returns

df – First column is time, the following columns obtain the absorbance values at the given detection wavelength in the column name.

Return type

pandas.DataFrame

Module contents

Submodules

mocca.dad_data.models module

Created on Tue Aug 3 13:16:51 2021

@author: haascp

class mocca.dad_data.models.**CompoundData**(*hplc_system_tag: str*, *experiment: dataclasses.InitVar['mocca.user_interaction.user_objects.HplcInput']*, *wl_high_pass: dataclasses.InitVar[float] = None*, *wl_low_pass: dataclasses.InitVar[float] = None*)

Bases: *DadData*

Data container for HPLC-DAD data with peaks originating from compounds.

data: ndarray

experiment: dataclasses.InitVar['mocca.user_interaction.user_objects.HplcInput']

hplc_system_tag: str

path: str

time: ndarray

warnings: List[str]

wavelength: ndarray

class mocca.dad_data.models.**DadData**(*hplc_system_tag: str*, *experiment: dataclasses.InitVar['mocca.user_interaction.user_objects.HplcInput']*, *wl_high_pass: dataclasses.InitVar[float] = None*, *wl_low_pass: dataclasses.InitVar[float] = None*)

Bases: object

Base class for HPLC-DAD data.

data: ndarray

experiment: dataclasses.InitVar['mocca.user_interaction.user_objects.HplcInput']

hplc_system_tag: str

path: str

time: ndarray

warnings: List[str]

wavelength: ndarray

wl_high_pass: dataclasses.InitVar[float] = None

wl_low_pass: dataclasses.InitVar[float] = None

class mocca.dad_data.models.**GradientData**(*hplc_system_tag: str*, *experiment: dataclasses.InitVar['mocca.user_interaction.user_objects.HplcInput']*, *wl_high_pass: dataclasses.InitVar[float] = None*, *wl_low_pass: dataclasses.InitVar[float] = None*)

Bases: *DadData*

Data container for gradient HPLC-DAD data.

original_data: ndarray

class mocca.dad_data.models.**ParafacData**(*impure_peak: dataclasses.InitVar['mocca.peak.models.CorrectedPeak']*, *parafac_comp_tensor: dataclasses.InitVar[tuple]*, *boundaries: dataclasses.InitVar[tuple]*, *shift: dataclasses.InitVar[int]*, *y_offset: dataclasses.InitVar[float]*)

Bases: object

Data container for synthetic data generated from PARAFAC models.

boundaries: dataclasses.InitVar[tuple]

impure_peak: dataclasses.InitVar['mocca.peak.models.CorrectedPeak']

parafac_comp_tensor: dataclasses.InitVar[tuple]

shift: dataclasses.InitVar[int]

y_offset: dataclasses.InitVar[float]

mocca.dad_data.process_funcs module

mocca.dad_data.process_funcs.**get_peak_locs**(*summed_data*)

Finds all peaks of data.

Parameters
summed_data (*numpy.ndarray*) – A 1D array representing the absorbances over time. Best used on data that already had data below threshold zeroed (see function filter_absorbance_by_threshold).

Returns
peaks – List of all peaks, as a list of BasePeak classes

Return type
list

mocca.dad_data.process_funcs.**merge_peaks**(*summed_data*, *peaks*)

Merges overlapping peaks in the data.

Parameters

- **summed_data** (*numpy.ndarray*) – A 1D array representing the absorbances over time. Best used on data that already had data below threshold zeroed (see function filter_absorbance_by_threshold).
- **peaks** (*list*) – List of all peaks as BasePeak objects

Returns
new_peaks – List of all peaks in dictionary format with keys maximum, left, and right. Peaks that overlap are merged together into one BasePeak.

Return type
list

mocca.dad_data.process_funcs.**pick_peaks**(*compound_data*, *experiment*, *absorbance_threshold*, *peaks_high_pass*, *peaks_low_pass*)

Finds all peaks of data and returns them as a chromatogram

Parameters

- **data** (*numpy.ndarray*) – Actual experimental data with shape [# of wavelengths] x [timepoints]. Generated from dataframe with absorbance_to_array function
- **absorbance_threshold** (*float*) – The threshold below which peaks will. In other words, at at least one (wavelength, timepoint) will have absorbance greater than absorbance_threshold in order to be counted as a peak.
- **peaks_high_pass** (*float*) – Time high pass filter only using peaks with a retention time greater than the here given value for data analysis
- **peaks_low_pass** (*float*) – Time low pass filter only using peaks with a retention time lower than the here given value for data analysis
- **expand_peaks** (*boolean*) – If True, then peaks will be expanded to their peak boundaries. If this is set to False, then only timepoints with cumulative absorbance greater than absorbance_threshold will be counted as part of the peak.

Returns
peaks – List of all peaks, as a list of tuples (maximum, left, right)

Return type
list

mocca.dad_data.process_gradientdata module

Created on Wed Aug 4 15:44:47 2021

@author: haascp

mocca.dad_data.process_gradientdata.**bsl_als**(*absorbance_array*)

Applies the baseline als algorithm row-wise (for every wavelength) on an absorbance array

Parameters
absorbance_arry (*numpy 2D-array*) – Absorbance values obtained by an HPLC run (time, wavelength dimension).

Returns
baseline_array – Baseline absorbance values

Return type
numpy 2D-array

mocca.dad_data.process_gradientdata.**bsl_als_alg**(*y*, *lam=100000.0*, *p=0.01*, *niter=3*)

Baseline correction algorithm: Optimized Python implementation of "Asymmetric Least Squares Smoothing" by P. Eilers and H. Boelens in 2005: https://stackoverflow.com/questions/29156532/python-baseline-correction-library, answer by Rustam Guliev.

Parameters

- **y** (*list*) – List of absorbance values for which the baseline should be determined.
- **lam** (*numeric, optional*) – Smoothness parameter. 10^2 10^9, but exceptions may occur. In any case one should vary on a grid that is approximately linear for log . Often visual inspection is sufficient for good values. The default is 1e6.
- **p** (*numeric, optional*) – Asymmetry parameter. 0.001 p 0.1 (for a signal with positive peaks), but exceptions may occur. Often visual inspection is sufficient to get good parameter values. The default is 0.01.
- **niter** (*integer, optional*) – To emphasize the basic simplicity of the algorithm, the number of iterations has been fixed to 10 (original documentation). In practical applications one should check whether the weights show any change; if not, convergence has been attained. The default is 3.

Returns
z – Simulated baseline of the given absorbance data.

Return type
list

mocca.dad_data.utils module

Created on Fri Dec 10 13:31:37 2021

@author: haascp

mocca.dad_data.utils.**absorbance_to_array**(*df*)

Generates a 2D absorbance array of the absorbance values.

mocca.dad_data.utils.**apply_filter**(*dataframe*, *wl_high_pass*, *wl_low_pass*, *bandwidth=2*, *reference_wl=True*)

Filters absorbance data of tidy 3D DAD dataframes to remove noise and background systematic error.

mocca.dad_data.utils.**df_to_array**(*df*)

Takes a tidy dataframe of HPLC-DAD data and returns a numpy array of " absorbance values as well as a vector for the time domain and a vector for " the wavelength domain.

mocca.dad_data.utils.**get_reference_signal**(*dataframe*, *bandwidth=5*)

Returns the averaged signal over the last number of wavelengths as given by the bandwidth.

mocca.dad_data.utils.**sum_absorbance_by_time**(*data*)

Sums the absorbances for each time point over all wavelengths

Parameters

data (*numpy.ndarray*) – Actual experimental data with shape [# of wavelengths] x [time-points]. Generated from dataframe with absorbance_to_array function

Returns

A 1D array containing the sum of wavelengths at each time point

Return type

numpy.ndarray

mocca.dad_data.utils.**trim_data**(*data*, *time*, *length*)

Trims the 2D DADData in the time dimension to the length provided.

Module contents

mocca.decomposition namespace

Submodules

mocca.decomposition.alternative_objective_funcs module

Created on Wed Feb 23 09:17:34 2022

@author: haascp

mocca.decomposition.alternative_objective_funcs.**get_all_comp_sum**(*parafac_factors*, *comp_tensor_shape*, *show_parafac_analytics*)

Iterative PARAFAC approach is a maximization problem. Based on the component tensor data shape, the maximal sum of integrals is found for every component part in the data tensor. The integrals of these component integrals are summed up. This sum is the maximization objective for the iterative PARAFAC approach.

mocca.decomposition.alternative_objective_funcs.**get_all_non_comp_sum**(*parafac_factors*, *comp_tensor_shape*, *show_parafac_analytics*)

Iterative PARAFAC approach is a minimization problem. Based on the component tensor data shape, the maximal sum of integrals is found for every component part in the data tensor. The integrals of all other components in this component part are summed up and should be ideally zero (only component spectrum contributes to component part of tensor). This final sum is the minimization objective for the iterative PARAFAC approach.

mocca.decomposition.alternative_objective_funcs.**get_comp_sum**(*parafac_factors*, *start_slice*, *end_slice*)

Get summed integrals from the highest inegral component in one slice of PARAFAC factors.

mocca.decomposition.alternative_objective_funcs.**get_impure_integral_sum**(*parafac_factors*, *show_parafac_analytics*)

This objective function assumes that the PARAFAC model is best, when the summed integral of all components in the impure peak slice is maximized.

mocca.decomposition.alternative_objective_funcs.**get_non_comp_sum**(*parafac_factors*, *start_slice*, *end_slice*)

Get summed integrals from all components except the highest inegral component in one slice of the PARAFAC factors.

mocca.decomposition.alternative_objective_funcs.**get_total_integral_sum**(*parafac_factors*, *show_parafac_analytics*)

Iterative PARAFAC approach is a maximization problem. This objective function assumes that the PARAFAC model is best, when the overall integral over all slices is maximized.

mocca.decomposition.data_tensor module

Created on Fri Jan 14 09:04:46 2022

@author: haascp

mocca.decomposition.data_tensor.**create_data_tensor**(*parafac_data*)

Takes a list of peak data and returns a data tensor in the format used for PARAFAC decomposition.

mocca.decomposition.data_tensor.**get_comp_peak_data_list**(*relevant_comp*, *boundaries*, *iter_offset*)

Returns a list of maximum-aligned peak data of the comp peaks of the relevant components. Moreover, returns the shape of the component list (how many slices per component in the list).

mocca.decomposition.data_tensor.**get_comp_peaks**(*relevant_comp*)

Returns exactly five peaks from which the given component was created. If less than 5 peaks were used to create the component, the peaks are multiplied as long as five peaks are reached.

mocca.decomposition.data_tensor.**get_offset_peak_to_comp**(*created_from_peak*, *comp*)

Returns the time offset of the maxima of oeak and quali component.

mocca.decomposition.data_tensor.**get_parafac_tensor**(*impure_peak*, *quali_comp_db*, *iter_offset*, *show_parafac_analytics*)

Processor function of the data tensor creation. Takes in the impure peak, the qualitative component db and an iteration offset and returns a DataTensor object which is used for PARAFAC decomposition.

mocca.decomposition.data_tensor.**get_relevant_comp**(*impure_peak*, *quali_comp_db*)

Returns component which is not unknown or impurity (-> only components which were given by the user via compound) and which overlap with the impure peak. If multiple components overlapping the component with the best UV-Vis correlation coefficient is returned.

mocca.decomposition.data_tensor.**get_tensor_boundaries**(*impure_peak*, *relevant_comp*, *iter_offset*)

Returns a tuple containing the leftest and rightest peak/component boundaries. The boundaries are corrected by the iteration offset if the iterative PARAFAC algorithm is used.

mocca.decomposition.data_tensor.**get_zero_ext_impure_peak_data**(*impure_peak*, *boundaries*, *iter_offset*)

Returns zero-extended (to the boundaries) peak data of the impure peak. The location of the peak is corrected by the iteration offset in case the iterative PARAFAC algorithm is applied.

mocca.decomposition.data_tensor.**get_zero_extended_peak_data**(*peak_data*, *left*, *boundaries*)

Returns zero-extended (to the boundaries) peak data.

mocca.decomposition.data_tensor.**get_zeros_array**(*boundaries*, *n_wavelengths*)

Retruns array of zeros in the shape of the wavelengths and boundaries.

mocca.decomposition.data_tensor.**normalize_peak_data**(*parafac_data*)

Normalizes all slices to the maximum absorbance of the slice.

mocca.decomposition.iterative_parafac module

Created on Sun Jan 16 11:15:19 2022

@author: haascp

mocca.decomposition.iterative_parafac.**iterative_parafac**(*impure_peak*, *quali_comp_db*, *absorbance_threshold*, *relative_distance_thresh*, *spectrum_correl_coef_thresh*, *show_parafac_analytics*)

The trilinearity-breaking mode retention time requires iterative PARAFAC algorithm.

mocca.decomposition.model module

Created on Tue Feb 22 11:15:42 2022

@author: haascp

class mocca.decomposition.model.**DataTensor**(*tensor: ndarray*, *boundaries: tuple*, *relevant_comp:* QualiComponent, *comp_tensor_shape: tuple*, *y_offset: float*)

Bases: `object`

Model of data tensors used as input for the PARAFAC decomposition algorithm.

boundaries: `tuple`

comp_tensor_shape: `tuple`

relevant_comp: *QualiComponent*

tensor: `ndarray`

y_offset: `float`

class mocca.decomposition.model.**ParafacModel**(*impure_peak: CorrectedPeak | IntegratedPeak*, *n_comps: int*, *pca_explained_variance: list*, *weights: list*, *factors: list*, *data_tensor: DataTensor*, *iter_offset: int*, *iter_objective_func: list | None = None*, *peaks: List[CorrectedPeak | IntegratedPeak] | None = None*)

Bases: object

Stores all relevant information of a PARAFAC model.

create_parafac_peaks(*absorbance_threshold*, *spectrum_correl_coef_thresh*)

If two UV-Vis traces in the PARAFAC components are too similar, no PARAFAC peaks are created. PARAFAC peaks' dataseta are created synthetically generated by using the PARAFAC factors of the model and filling the rest of the array up with zeros. PARAFAC peaks get an index of -impure_peak.idx

data_tensor: *DataTensor*

factors: list

impure_mse: float

impure_peak: *CorrectedPeak* | *IntegratedPeak*

iter_objective_func: list = None

iter_offset: int

n_comps: int

pca_explained_variance: list

peaks: List[*CorrectedPeak* | *IntegratedPeak*] | None = None

weights: list

mocca.decomposition.parafac_funcs module

Created on Fri Jan 14 08:57:14 2022

@author: haascp

mocca.decomposition.parafac_funcs.**estimate_pca_n_comps**(*data_tensor*, *impure_peak*, *show_parafac_analytics*)

Returns an estimate of the number of components in the impure peak.

mocca.decomposition.parafac_funcs.**parafac**(*impure_peak*, *quali_comp_db*, *iter_offset*, *show_parafac_analytics*)

PARAFAC decomposition processing routine of impure peaks. An iteration offset is introduced to allow for iterative PARAFAC approach.

mocca.decomposition.parafac_funcs.**print_parafac_analytics**(*parafac_model*)

Prints out PARAFAC decomposition model results. Used for debugging and dev.

mocca.decomposition.utils module

Created on Fri Jan 14 09:01:51 2022

@author: haascp

mocca.decomposition.utils.**check_absorbance_thresh**(*parafac_peak*, *absorbance_threshold*)

Checks if maximum absorbance in synthetically created PARAFAC peak dataset exceeds absorbance threshold.

mocca.decomposition.utils.**check_any_compound_overlap**(*peak*, *quali_comp_db*)

Checks if a given peak overlaps with any component in the quali_comp_db.

mocca.decomposition.utils.**check_comp_in_impure**(*parafac_model*, *absorbance_threshold*)

Checks if the maximum absorbance of the known PARAFAC component in the impure peak exceeds the absorbance threshold.

mocca.decomposition.utils.**check_comp_overlap**(*peak*, *comp*)

Checks if a given peak overlaps with a given component.

mocca.decomposition.utils.**check_same_uvvis**(*parafac_model*, *spectrum_correl_coef_thresh*)

Checks if any two parafac components share the same UV-Vis trace.

mocca.decomposition.utils.**check_summed_factor_uvvis**(*parafac_model*, *spectrum_correl_thresh*)

Checks if summed UV-Vis spectra of the PARAFAC factors add up to the pure UV-Vis spectrum.

mocca.peak package

Submodules

mocca.peak.check module

Created on Thu Dec 2 09:16:47 2021

@author: haascp

mocca.peak.check.**check_peak**(*expanded_peak*, *detector_limit*, *show_analytics*, *param=2.5*)

Peak checking routine. Returns a checked peak with pure and saturation attributes.

mocca.peak.check.**check_peak_purity**(*peak*, *show_analytics*, *param=2.5*)

Returns peak purity prediction by performing the described test sequence. Plots and prints infromation about the peak purity prediction.

mocca.peak.check.**check_peak_saturation**(*picked_peak*, *detector_limit*)

Sets peak attribute saturation to either True or False based on if the peak absorbance exceeds detector_limit.

mocca.peak.correct module

Created on Tue Dec 14 15:30:29 2021

@author: haascp

mocca.peak.correct.**correct_offset**(*integrated_peak*, *istd_peaks*, *offset*)

Creates a corrected peak using internal standard peaks to obtain a retention time offset.

mocca.peak.database module

Created on Fri Nov 26 08:28:12 2021

@author: haascp

class mocca.peak.database.**PeakDatabase**(*peaks: List[ProcessedPeak] | None = None*)

Bases: object

Database class to store and organize peaks of HPLC-DAD data.

increment_unknown_counter()

Increments the unknown counter by one.

insert_peak(*new_peak*)

Inserts a peak in the database given it is of the processed peak type. If a peak with the same borders in the same dataset already exists, it will be overwritten.

update_unknown_counter()

Updates the unknown counter which gives the highest trailing number in the peak database of a peak assigned with a compound_id starting with the string literal 'unknown'.

mocca.peak.expand module

Created on Thu Dec 2 09:16:47 2021

@author: haascp

mocca.peak.expand.**expand_peak**(*picked_peak*, *absorbance_threshold*)

Expands peak boundaries to those actually in the data. It keeps expanding them until the absorbance falls below one twentieth of the given absorbance threshold. Returns a picked peak with modified peak boundaries (left, right).

mocca.peak.integrate module

Created on Tue Dec 7 10:57:23 2021

@author: haascp

mocca.peak.integrate.**integrate_peak**(*checked_peak*)

Integrates the peak. Returns an integrated peak with the integral attribute set.

mocca.peak.match module

Created on Wed Dec 1 12:06:57 2021

@author: haascp

mocca.peak.match.**get_filtered_similarity_dicts**(*peak*, *component_db*, *spectrum_correl_coef_thresh*, *relative_distance_thresh*, *print_out=False*)

Filters the list of similarity dictionaries with regard to the given thresholds. Return possible matches which have a spectral correlation coefficient higher than the given threshold and a relative distance between the peak maxima lower than the given threshold.

mocca.peak.match.**get_relative_distance**(*peak*, *component*)

Returns the distance of an offset-corrected peak maximum and a component maximum relative to the length of the time vector.

mocca.peak.match.**get_similarity_dicts**(*peak*, *component_db*, *relative_distance_thresh*)

Returns a sorted list of dictionaries. For each component in the given database, similarity values to the given peak are stored.

mocca.peak.match.**get_spectrum_correl_coef**(*peak*, *component*)

Returns the correlation coefficient of the average peak spectrum and the spectrum of the component.

mocca.peak.match.**match_peak**(*corrected_peak*, *component_db*, *spectrum_correl_coef_thresh*, *relative_distance_thresh*, *print_similarity_dicts=False*)

Routine to assign possible matches to a returned preprocessed peak.

mocca.peak.match.**update_matches**(*peak*, *new_matches*)

Updates the matches of a given peak.

mocca.peak.models module

class mocca.peak.models.**BasePeak**(*left: int*, *right: int*, *maximum: int*, *offset: int*)

Bases: object

Base peak class.

left: int

maximum: int

offset: int

right: int

class mocca.peak.models.**CheckedPeak**(*left: int*, *right: int*, *maximum: int*, *offset: int*, *dataset:* CompoundData, *idx: int*, *saturation: bool*, *pure: bool*)

Bases: *PickedPeak*

Class for peaks checked with regard to saturation and purity.

pure: bool

saturation: bool

class mocca.peak.models.**CorrectedPeak**(*left: int*, *right: int*, *maximum: int*, *offset: int*, *dataset:* CompoundData, *idx: int*, *saturation: bool*, *pure: bool*, *integral: float*, *istd: List[*IstdPeak*]*)

Bases: *IntegratedPeak*

Class for peaks with added retention time offset. From this class on, retention times in the peaks are already corrected. This means, that accessing data from the dataset attribute require prior un-offsetting.

istd: List[*IstdPeak*]

class mocca.peak.models.**IntegratedPeak**(*left: int*, *right: int*, *maximum: int*, *offset: int*, *dataset:* CompoundData, *idx: int*, *saturation: bool*, *pure: bool*, *integral: float*)

Bases: *CheckedPeak*

Class for integrated peaks.

integral: float

class mocca.peak.models.**IstdPeak**(*left: int*, *right: int*, *maximum: int*, *dataset:* CompoundData, *integral: float*, *offset: int*, *compound_id: str*, *concentration: float*)

Bases: object

Class for istd peaks to be added to the peak classes below.

compound_id: str

concentration: float

dataset: *CompoundData*

integral: float

left: int

maximum: int

offset: int

right: int

class mocca.peak.models.**PickedPeak**(*left: int*, *right: int*, *maximum: int*, *offset: int*, *dataset:* CompoundData, *idx: int*)

Bases: *BasePeak*

Class for picked peaks out of DAD data. Also valid for expanded peaks.

dataset: *CompoundData*

idx: int

class mocca.peak.models.**PreprocessedPeak**(*left: int*, *right: int*, *maximum: int*, *offset: int*, *dataset:* CompoundData, *idx: int*, *saturation: bool*, *pure: bool*, *integral: float*, *istd: List[*IstdPeak*]*, *matches: List[dict]*)

Bases: *CorrectedPeak*

Class for preprocessed peaks containing a list of possible component matches in the attribute compound_id.

matches: List[dict]

class mocca.peak.models.**ProcessedPeak**(*left: int*, *right: int*, *maximum: int*, *offset: int*, *dataset:* CompoundData, *idx: int*, *saturation: bool*, *pure: bool*, *integral: float*, *istd: List[*IstdPeak*] | None = None*, *compound_id: str | None = None*, *concentration: float | None = None*, *is_compound: bool = False*)

Bases: object

Class of fully processed peaks ready to be put in the peak database.

compound_id: str | None = None

concentration: float | None = None

dataset: *CompoundData*

idx: int

integral: float

is_compound: bool = False

istd: List[*IstdPeak*] = None

left: int

maximum: int

offset: int

pure: bool

right: int

saturation: bool

mocca.peak.process module

Created on Fri Dec 17 11:03:08 2021

@author: haascp

mocca.peak.process.**process_peak**(*peak*, *compound*, *is_compound=False*)

Creates a processed peak by addding compound information to it.

mocca.peak.purity_funcs module

Created on Tue Nov 23 15:55:25 2021

@author: haascp

mocca.peak.purity_funcs.**get_agilent_thresholds**(*peak_data*, *max_loc*, *noise_variance*, *param=2.5*)

Returns the thresholds calculated by the Agilent purity algorithm.

mocca.peak.purity_funcs.**get_correls**(*peak_data*, *max_loc*)

Get a list with correlation coefficients of UV-Vis spectra at every timepoint with reference to the UV-Vis spectrum at maximum absorbance.

mocca.peak.purity_funcs.**get_max_loc**(*peak_data*)

Returns the maximum location of the given peak data.

mocca.peak.purity_funcs.**get_noise_variance**(*peak*)

Filters dataset with only timepoints whose max absorbance at any wavelength is below 1% of max absorbance. Returns the average of the variance over all wavelengths.

mocca.peak.purity_funcs.**get_pca_explained_variance**(*peak_data*)

Calculates the ration of explained variance by the first principal component of the devonvoluted peak data.

mocca.peak.purity_funcs.**get_purity_value_agilent**(*peak_data*, *correls*, *agilent_thresholds*)

Uses Agilent's peak purity algorithm to predict purity of peak. Param gives strictness of test (original was 0.5, which is more strict)

mocca.peak.purity_funcs.**get_trimmed_peak_data**(*peak*)

Returns peak data trimmed with cut edges of the peak to 5% of max absorbance to avoid noise artifacts.

mocca.peak.purity_funcs.**predict_purity_unimodal**(*correls*)

Checks for unimodality of a peak by an averaging filter of length 3 on the correlation vector to the maximum https://stackoverflow.com/questions/14313510/how-to-calculate-rolling-moving-average-using-numpy-scipy

mocca.peak.quantify module

Created on Tue Jan 4 15:49:54 2022

@author: haascp

mocca.peak.quantify.**quantify_peak**(*peak*, *quant_comp_db*, *quali_comp_db*)

Takes a peak with an integral and quantifies it by calculating the corresponding concentration if the compound is present in the quantification database.

mocca.peak.resolve_impure module

Created on Fri Jan 14 08:59:36 2022

@author: haascp

mocca.peak.resolve_impure.**create_parafac_peak**(*comp_i*, *parafac_model*)

Return synthetic PARAFAC peaks created from the PARAFAC decomposition results.

mocca.peak.resolve_impure.**create_pure_peak**(*impure_peak*)

Takes an impure peak and returns its copy with the pure attribute True.

mocca.peak.resolve_impure.**get_parafac_data_shift**(*iter_offset*)

If the iteration offset is larger than zero, the impure peak was shifted and must therefore be shifted back in the resulting PARAFAC data. If the iteration shift was negative, the pure signals were shifted and nothing has to be done.

mocca.peak.utils module

Created on Thu Dec 2 10:00:18 2021

@author: haascp

mocca.peak.utils.**average_peak_spectrum**(*peak*)

Calculates mean spectrum over peak from left to right border.

mocca.peak.utils.**get_peak_data**(*peak*)

Returns absorbance data from the left to the right border of the peak for all wavelengths. If the peak was offset-corrected, the left and right border are un-offset in order to access the correct data.

mocca.peak.utils.**get_retention_times**(*peak*)

Returns left and right borders as well as maximum of a peak as retention times.

mocca.peak.utils.**is_unimodal**(*L*, *high_val_threshold=inf*)

Checks if a list is unimodal (for use in peak purity).

Parameters

- **L** (*list*) – A list to test unimodality for

- **high_val_threshold** (*numeric, optional*) – If set, then values above high_val_threshold will not be counted in unimodality testing. Default is np.inf (i.e. this threshold is not used).

Returns
True if the list is unimodal ignoring high values; False otherwise.

Return type
TYPE boolean

Module contents

mocca.report namespace

Submodules

mocca.report.bad_chromatograms module

mocca.report.calibration_library module

mocca.report.chromatograms module

mocca.report.compound_library module

mocca.report.compound_tracking module

mocca.report.deconvolution module

mocca.report.gradient module

mocca.report.hplc_input module

mocca.report.main module

mocca.report.peak_library module

mocca.report.utils module

Created on Fri Jan 7 16:28:04 2022

@author: haascp

mocca.report.utils.**settings_to_df**(*settings*)

Transfers relevant information of Settings objects in a pandas df.

mocca.user_interaction namespace

Submodules

mocca.user_interaction.campaign module

Created on Mon Dec 13 09:05:19 2021

@author: haascp

class mocca.user_interaction.campaign.**HplcDadCampaign**(*autosave_path=None*)

Bases: object

Main class for HPLC-DAD campaigns containing all user input as well as results from the data analysis.

add_hplc_input(*hplc_input*)

All reaction concentration are metadata and should be trated like reaction temperature etc. Only give compound_conc if standard. If compound_id given: Add new peak to component db and update it If conc given: Add peak to quanti_component with same compound_id and update Store user input concs as negative conc in peak

load_campaign(*path='hplc_dad_campaign.pkl'*)

Loads campaign object which was saved as pkl file.

process_all_hplc_input(*settings*)

This function sets all expeirments of the cmapaign to the unprocessed state and processes all given hplc input. NOTE: This function has to be run if a new compound is added to the component database via compound experiment so that all peaks are assigned consistently

process_new_hplc_input()

Only unprocessed runs are analyzed. No compound runs are allowed. Settings can only be changed via provess_all_hplc_input.

save_campaign(*path='hplc_dad_campaign.pkl'*, *remove_raw_data=False*)

Saves campaign object as pkl file. If remove_raw_data is True, all raw data are removed before saving to reduce file sizes.

mocca.user_interaction.settings module

Created on Tue Dec 21 14:57:25 2021

@author: haascp

class mocca.user_interaction.settings.**Settings**(*hplc_system_tag: str*, *detector_limit: float | None = None*, *absorbance_threshold: float | None = 500*, *wl_high_pass: float | None = None*, *wl_low_pass: float | None = None*, *peaks_high_pass: float | None = None*, *peaks_low_pass: float | None = None*, *spectrum_correl_thresh: float | None = 0.95*, *relative_distance_thresh: float | None = 0.01*)

Bases: object

Data container to store all user given data analysis settings.

absorbance_threshold: float | None = 500

detector_limit: float | None = None

hplc_system_tag: str

peaks_high_pass: float | None = None

peaks_low_pass: float | None = None

relative_distance_thresh: float | None = 0.01

spectrum_correl_thresh: float | None = 0.95

wl_high_pass: float | None = None

wl_low_pass: float | None = None

mocca.user_interaction.suggest_calibration module

Created on Tue Jan 25 09:10:24 2022

@author: haascp

mocca.user_interaction.suggest_calibration.**suggest_initialization_runs**(*n_calib_dict*, *max_conc_dict=None*, *istd_key=None*, *n_solvents=0*)

Returns data frame of suggested runs for a standardized HPLC initialization (initialization = calibration + database of analytes).

mocca.user_interaction.user_objects module

Created on Tue Jan 25 10:46:55 2022

@author: haascp

class mocca.user_interaction.user_objects.**Compound**(*key: str*, *conc: float | None = None*, *is_solvent: bool = False*, *is_istd: bool = False*)

Bases: object

Data container to store user input regarding added compounds.

conc: float | None = None

is_istd: bool = False

is_solvent: bool = False

key: str

class mocca.user_interaction.user_objects.**CustomData**(*data: ndarray*, *time: list*, *wavelength: list*)

Bases: object

Data container to store custom data like, e.g., from HPLC chromatogram simulations.

data: ndarray

time: list

wavelength: list

class mocca.user_interaction.user_objects.**Gradient**(*path: str*)

Bases: object

Data container to store user input regarding gradients.

dataset: *GradientData*

path: str

class mocca.user_interaction.user_objects.**HplcInput**(*path: str*, *gradient:* Gradient | *None*, *compound:* Compound | *None* = *None*, *istd:* *List[*InternalStandard*]* | *None* = *None*, *processed: bool* = *False*, *custom_data:* CustomData | *None* = *None*)

Bases: object

Data container to store user input.

compound: *Compound* | None = None

custom_data: *CustomData* = None

gradient: *Gradient* | None

istd: List[*InternalStandard*] | None = None

path: str

processed: bool = False

class mocca.user_interaction.user_objects.**InternalStandard**(*key: str*, *conc: float* | *None* = *None*)

Bases: object

Data container to store user input regarding added internal standards.

conc: float | None = None

key: str

mocca.visualization package

Submodules

mocca.visualization.basic_plots module

Created on Wed Aug 4 18:57:16 2021

@author: haascp

mocca.visualization.basic_plots.**plot_1D_data**(*df*, *xlabel=''*, *ylabel=''*, *title=''*, *color=None*, *reduce_data=True*)

Plots a set of 1D data.

mocca.visualization.basic_plots.**plot_1D_layer**(*df*, *xlabel=''*, *ylabel=''*, *title=''*, *color=None*, *reduce_data=True*)

Plots a set of 1D data.

mocca.visualization.basic_plots.**plot_1D_scatter**(*df*, *xlabel=''*, *ylabel=''*, *title=''*, *color=None*, *reduce_data=True*)

Plots a set of 1D data.

mocca.visualization.basic_plots.**plot_1D_scatter_layer**(*df*, *xlabel=''*, *ylabel=''*, *title=''*, *color=None*, *reduce_data=True*)

Plots a set of 1D data.

mocca.visualization.calibration_plots module

Created on Fri Jan 7 14:15:20 2022

@author: haascp

mocca.visualization.calibration_plots.**plot_calibration_curves**(*comp*)

Function for the visualization of the calibration curves.

mocca.visualization.parafac_plots module

Created on Fri Jan 7 18:05:10 2022

@author: haascp

mocca.visualization.parafac_plots.**plot_aligned_tensor**(*parafac_model*)

Plots retention profiles (summed absorbance over all wavelengths) of all slices in the data tensor on the optimized iteration shift.

mocca.visualization.parafac_plots.**plot_impure_peak_spectra**(*impure_peak*)

Generates plot with UV-Vis spectra at every time point in the impure peak.

mocca.visualization.parafac_plots.**plot_normalized_integrals**(*normalized_integrals*)

Plots integrals of all PARAFAC components over all slices of the tensor.

mocca.visualization.parafac_plots.**plot_objective_func**(*parafac_model*)

Plots objective function outcome vs iteration offset when using the iterative PARAFAC approach.

mocca.visualization.parafac_plots.**plot_retention**(*parafac_model*)

Plots normalized retention profiles of the PARAFAC components as well as of the impure peak.

mocca.visualization.parafac_plots.**plot_uvvis_specs**(*parafac_model*)

Generates plot of the UV-Vis traces of the PARAFAC component together with the UV-Vis spectrum of the known compound used for the data tensor.

mocca.visualization.results_plot module

Created on Fri Jan 7 15:12:02 2022

@author: haascp

mocca.visualization.results_plot.**plot_chrom_with_peaks**(*chrom*)

Plots summed absorbance vs time with highlighted picked peak zones.

mocca.visualization.utils module

Created on Fri Jan 7 14:11:19 2022

@author: haascp

mocca.visualization.utils.**round_to_n**(*x*, *n*)

Returns number in a format suitable for data visualization.

Module contents

Module contents

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

mocca, 38
mocca.campaign, 11
mocca.campaign.experiment_funcs, 9
mocca.campaign.process_funcs, 10
mocca.campaign.utils, 10
mocca.chromatogram, 13
mocca.chromatogram.assign, 11
mocca.chromatogram.correct, 12
mocca.chromatogram.model, 12
mocca.chromatogram.preprocessor, 13
mocca.chromatogram.quantify, 13
mocca.chromatogram.utils, 13
mocca.components, 16
mocca.components.databases, 13
mocca.components.models, 14
mocca.components.quali_funcs, 15
mocca.components.quant_funcs, 15
mocca.components.utils, 16
mocca.dad_data, 23
mocca.dad_data.apis, 19
mocca.dad_data.apis.allotrope, 16
mocca.dad_data.apis.angi, 17
mocca.dad_data.apis.chemstation, 18
mocca.dad_data.apis.custom, 18
mocca.dad_data.apis.empower, 19
mocca.dad_data.apis.labsolutions, 19
mocca.dad_data.models, 19
mocca.dad_data.process_funcs, 21
mocca.dad_data.process_gradientdata, 22
mocca.dad_data.utils, 23
mocca.decomposition, 23
mocca.decomposition.alternative_objective_funcs, 23
mocca.decomposition.data_tensor, 24
mocca.decomposition.iterative_parafac, 25
mocca.decomposition.model, 25
mocca.decomposition.parafac_funcs, 26
mocca.decomposition.utils, 27
mocca.peak, 33
mocca.peak.check, 27
mocca.peak.correct, 27
mocca.peak.database, 28
mocca.peak.expand, 28
mocca.peak.integrate, 28
mocca.peak.match, 28
mocca.peak.models, 29
mocca.peak.process, 31
mocca.peak.purity_funcs, 31
mocca.peak.quantify, 32
mocca.peak.resolve_impure, 32
mocca.peak.utils, 32
mocca.report, 33
mocca.report.utils, 33
mocca.user_interaction, 34
mocca.user_interaction.campaign, 34
mocca.user_interaction.settings, 34
mocca.user_interaction.suggest_calibration, 35
mocca.user_interaction.user_objects, 35
mocca.visualization, 38
mocca.visualization.basic_plots, 36
mocca.visualization.calibration_plots, 37
mocca.visualization.parafac_plots, 37
mocca.visualization.results_plot, 37
mocca.visualization.utils, 38

INDEX

A

absorbance_threshold (*mocca.user_interaction.settings.Settings attribute*), 34
absorbance_to_array() (*in module mocca.dad_data.utils*), 23
add_hplc_input() (*mocca.user_interaction.campaign.HplcDadCampaign method*), 34
apply_filter() (*in module mocca.dad_data.utils*), 23
assign_best_match_peak() (*in module mocca.chromatogram.assign*), 11
assign_matched_peaks() (*in module mocca.chromatogram.assign*), 11
assign_peaks_compound() (*in module mocca.chromatogram.assign*), 11
assign_peaks_react() (*in module mocca.chromatogram.assign*), 11
assign_unmatched_peaks_compound() (*in module mocca.chromatogram.assign*), 11
assign_unmatched_peaks_react() (*in module mocca.chromatogram.assign*), 11
average_peak_spectrum() (*in module mocca.peak.utils*), 32
average_ret_times_over_peaks() (*in module mocca.components.utils*), 16
average_spectra_over_peaks() (*in module mocca.components.utils*), 16

B

BaseDatabase (*class in mocca.components.databases*), 13
BasePeak (*class in mocca.peak.models*), 29
boundaries (*mocca.dad_data.models.ParafacData attribute*), 20
boundaries (*mocca.decomposition.model.DataTensor attribute*), 25
bsl_als() (*in module mocca.dad_data.process_gradientdata*), 22
bsl_als_alg() (*in module mocca.dad_data.process_gradientdata*), 22

C

calib_data (*mocca.components.models.QuantComponent attribute*), 15
calib_factors (*mocca.components.models.QuantComponent attribute*), 15
calib_scores (*mocca.components.models.QuantComponent attribute*), 15
check_absorbance_thresh() (*in module mocca.decomposition.utils*), 27
check_any_compound_overlap() (*in module mocca.decomposition.utils*), 27
check_comp_in_impure() (*in module mocca.decomposition.utils*), 27
check_comp_overlap() (*in module mocca.decomposition.utils*), 27
check_istd() (*in module mocca.campaign.utils*), 10
check_overlap() (*in module mocca.chromatogram.utils*), 13
check_peak() (*in module mocca.peak.check*), 27
check_peak_purity() (*in module mocca.peak.check*), 27
check_peak_saturation() (*in module mocca.peak.check*), 27
check_peaks_compound_id() (*in module mocca.components.utils*), 16
check_same_dataset() (*in module mocca.chromatogram.utils*), 13
check_same_uvvis() (*in module mocca.decomposition.utils*), 27
check_summed_factor_uvvis() (*in module mocca.decomposition.utils*), 27
CheckedPeak (*class in mocca.peak.models*), 29
Chromatogram (*class in mocca.chromatogram.model*), 12
comp_tensor_shape (*mocca.decomposition.model.DataTensor attribute*), 25
Compound (*class in mocca.user_interaction.user_objects*), 35
compound (*mocca.user_interaction.user_objects.HplcInput attribute*), 36
compound_id (*mocca.components.models.QualiComponent attribute*), 14

compound_id (*mocca.components.models.QuantComponent attribute*), 15
compound_id (*mocca.peak.models.IstdPeak attribute*), 30
compound_id (*mocca.peak.models.ProcessedPeak attribute*), 30
CompoundData (*class in mocca.dad_data.models*), 19
conc (*mocca.user_interaction.user_objects.Compound attribute*), 35
conc (*mocca.user_interaction.user_objects.InternalStandard attribute*), 36
concentration (*mocca.peak.models.IstdPeak attribute*), 30
concentration (*mocca.peak.models.ProcessedPeak attribute*), 30
correct_istd_offset() (*in module mocca.chromatogram.correct*), 12
correct_offset() (*in module mocca.peak.correct*), 27
CorrectedPeak (*class in mocca.peak.models*), 29
create_calibration_dict() (*in module mocca.components.quant_funcs*), 15
create_data_tensor() (*in module mocca.decomposition.data_tensor*), 24
create_linear_models() (*in module mocca.components.quant_funcs*), 15
create_parafac_peak() (*in module mocca.peak.resolve_impure*), 32
create_parafac_peaks() (*mocca.decomposition.model.ParafacModel method*), 26
create_pure_peak() (*in module mocca.peak.resolve_impure*), 32
create_quali_component() (*in module mocca.components.quali_funcs*), 15
create_quant_component() (*in module mocca.components.quant_funcs*), 15
created_from (*mocca.components.models.QualiComponent attribute*), 14
created_from (*mocca.components.models.QuantComponent attribute*), 15
custom_data (*mocca.user_interaction.user_objects.HplcInput attribute*), 36
CustomData (*class in mocca.user_interaction.user_objects*), 35

D

DadData (*class in mocca.dad_data.models*), 20
data (*mocca.dad_data.models.CompoundData attribute*), 19
data (*mocca.dad_data.models.DadData attribute*), 20
data (*mocca.user_interaction.user_objects.CustomData attribute*), 35
data_tensor (*mocca.decomposition.model.ParafacModel attribute*), 26
dataset (*mocca.peak.models.IstdPeak attribute*), 30
dataset (*mocca.peak.models.PickedPeak attribute*), 30
dataset (*mocca.peak.models.ProcessedPeak attribute*), 30
dataset (*mocca.user_interaction.user_objects.Gradient attribute*), 36
DataTensor (*class in mocca.decomposition.model*), 25
delete_all_items() (*mocca.components.databases.BaseDatabase method*), 14
detector_limit (*mocca.user_interaction.settings.Settings attribute*), 34
df_to_array() (*in module mocca.dad_data.utils*), 23

E

estimate_pca_n_comps() (*in module mocca.decomposition.parafac_funcs*), 26
expand_peak() (*in module mocca.peak.expand*), 28
experiment (*mocca.dad_data.models.CompoundData attribute*), 19
experiment (*mocca.dad_data.models.DadData attribute*), 20

F

factors (*mocca.decomposition.model.ParafacModel attribute*), 26
filter_peaks() (*in module mocca.components.utils*), 16

G

get_absorbance_maxima() (*in module mocca.components.quali_funcs*), 15
get_agilent_thresholds() (*in module mocca.peak.purity_funcs*), 31
get_all_comp_sum() (*in module mocca.decomposition.alternative_objective_funcs*), 23
get_all_non_comp_sum() (*in module mocca.decomposition.alternative_objective_funcs*), 23
get_best_match_compound_id() (*in module mocca.chromatogram.assign*), 11
get_comp_peak_data_list() (*in module mocca.decomposition.data_tensor*), 24
get_comp_peaks() (*in module mocca.decomposition.data_tensor*), 24
get_comp_sum() (*in module mocca.decomposition.alternative_objective_funcs*), 24
get_correls() (*in module mocca.peak.purity_funcs*), 31
get_distance_between() (*in module mocca.chromatogram.utils*), 13
get_filtered_peaks() (*in module mocca.components.utils*), 16

get_filtered_peaks_by_compound() (*in module mocca.components.utils*), 16
get_filtered_similarity_dicts() (*in module mocca.peak.match*), 28
get_function_paramenters() (*in module mocca.dad_data.apis.allotrope*), 16
get_impure_integral_sum() (*in module mocca.decomposition.alternative_objective_funcs*), 24
get_impure_istd_peak() (*in module mocca.chromatogram.correct*), 12
get_integrate_wl_index() (*in module mocca.components.quant_funcs*), 15
get_istd_offset() (*in module mocca.chromatogram.correct*), 12
get_istd_peak() (*in module mocca.chromatogram.correct*), 12
get_matched_peaks() (*in module mocca.chromatogram.assign*), 11
get_max_integral_peak() (*in module mocca.chromatogram.assign*), 11
get_max_loc() (*in module mocca.peak.purity_funcs*), 31
get_next_unknown_id() (*in module mocca.chromatogram.assign*), 11
get_noise_variance() (*in module mocca.peak.purity_funcs*), 31
get_non_comp_sum() (*in module mocca.decomposition.alternative_objective_funcs*), 24
get_offset_peak_to_comp() (*in module mocca.decomposition.data_tensor*), 24
get_parafac_data_shift() (*in module mocca.peak.resolve_impure*), 32
get_parafac_tensor() (*in module mocca.decomposition.data_tensor*), 24
get_pca_explained_variance() (*in module mocca.peak.purity_funcs*), 31
get_peak_data() (*in module mocca.peak.utils*), 32
get_peak_locs() (*in module mocca.dad_data.process_funcs*), 21
get_pure_istd_peak() (*in module mocca.chromatogram.correct*), 12
get_purity_value_agilent() (*in module mocca.peak.purity_funcs*), 31
get_quant_peaks_by_compound() (*in module mocca.components.utils*), 16
get_reference_signal() (*in module mocca.dad_data.utils*), 23
get_relative_distance() (*in module mocca.peak.match*), 28
get_relevant_comp() (*in module mocca.decomposition.data_tensor*), 24
get_retention_times() (*in module mocca.peak.utils*), 32
get_similarity_dicts() (*in module mocca.peak.match*), 29
get_sorted_compound_experiments() (*in module mocca.campaign.experiment_funcs*), 10
get_spectrum_correl_coef() (*in module mocca.peak.match*), 29
get_tensor_boundaries() (*in module mocca.decomposition.data_tensor*), 24
get_total_integral_sum() (*in module mocca.decomposition.alternative_objective_funcs*), 24
get_trimmed_peak_data() (*in module mocca.peak.purity_funcs*), 31
get_unknown_impurity_peaks() (*in module mocca.chromatogram.assign*), 11
get_unmatched_peaks() (*in module mocca.chromatogram.assign*), 11
get_unprocessed_experiments() (*in module mocca.campaign.experiment_funcs*), 10
get_uvvis_dataset_name() (*in module mocca.dad_data.apis.allotrope*), 17
get_valid_peaks() (*in module mocca.components.utils*), 16
get_zero_ext_impure_peak_data() (*in module mocca.decomposition.data_tensor*), 25
get_zero_extended_peak_data() (*in module mocca.decomposition.data_tensor*), 25
get_zeros_array() (*in module mocca.decomposition.data_tensor*), 25
Gradient (*class in mocca.user_interaction.user_objects*), 36
gradient (*mocca.user_interaction.user_objects.HplcInput attribute*), 36
GradientData (*class in mocca.dad_data.models*), 20

H

hplc_system_tag (*mocca.dad_data.models.CompoundData attribute*), 19
hplc_system_tag (*mocca.dad_data.models.DadData attribute*), 20
hplc_system_tag (*mocca.user_interaction.settings.Settings attribute*), 35
HplcDadCampaign (*class in mocca.user_interaction.campaign*), 34
HplcInput (*class in mocca.user_interaction.user_objects*), 36

I

idx (*mocca.peak.models.PickedPeak attribute*), 30
idx (*mocca.peak.models.ProcessedPeak attribute*), 31
impure_mse (*mocca.decomposition.model.ParafacModel attribute*), 26

impure_peak (*mocca.dad_data.models.ParafacData attribute*), 20
impure_peak (*mocca.decomposition.model.ParafacModel attribute*), 26
increment_unknown_counter() (*mocca.peak.database.PeakDatabase method*), 28
insert_by_compound_id() (*mocca.components.databases.QualiComponentDatabase method*), 14
insert_item() (*mocca.components.databases.BaseDatabase method*), 14
insert_peak() (*mocca.chromatogram.model.Chromatogram method*), 12
insert_peak() (*mocca.peak.database.PeakDatabase method*), 28
integral (*mocca.peak.models.IntegratedPeak attribute*), 30
integral (*mocca.peak.models.IstdPeak attribute*), 30
integral (*mocca.peak.models.ProcessedPeak attribute*), 31
integrate_on_wl() (*in module mocca.components.quant_funcs*), 15
integrate_peak() (*in module mocca.peak.integrate*), 28
integrate_wl_idx (*mocca.components.models.QuantComponent attribute*), 15
IntegratedPeak (*class in mocca.peak.models*), 29
InternalStandard (*class in mocca.user_interaction.user_objects*), 36
is_compound (*mocca.peak.models.ProcessedPeak attribute*), 31
is_istd (*mocca.user_interaction.user_objects.Compound attribute*), 35
is_solvent (*mocca.user_interaction.user_objects.Compound attribute*), 35
is_unimodal() (*in module mocca.peak.utils*), 32
istd (*mocca.peak.models.CorrectedPeak attribute*), 29
istd (*mocca.peak.models.ProcessedPeak attribute*), 31
istd (*mocca.user_interaction.user_objects.HplcInput attribute*), 36
IstdPeak (*class in mocca.peak.models*), 30
iter_objective_func (*mocca.decomposition.model.ParafacModel attribute*), 26
iter_offset (*mocca.decomposition.model.ParafacModel attribute*), 26
iterative_parafac() (*in module mocca.decomposition.iterative_parafac*), 25

K

key (*mocca.user_interaction.user_objects.Compound attribute*), 35
key (*mocca.user_interaction.user_objects.InternalStandard attribute*), 36

L

left (*mocca.components.models.QualiComponent attribute*), 14
left (*mocca.peak.models.BasePeak attribute*), 29
left (*mocca.peak.models.IstdPeak attribute*), 30
left (*mocca.peak.models.ProcessedPeak attribute*), 31
load_campaign() (*mocca.user_interaction.campaign.HplcDadCampaign method*), 34

M

match_peak() (*in module mocca.peak.match*), 29
matches (*mocca.peak.models.PreprocessedPeak attribute*), 30
maximum (*mocca.components.models.QualiComponent attribute*), 14
maximum (*mocca.peak.models.BasePeak attribute*), 29
maximum (*mocca.peak.models.IstdPeak attribute*), 30
maximum (*mocca.peak.models.ProcessedPeak attribute*), 31
merge_peaks() (*in module mocca.dad_data.process_funcs*), 21
mocca
    module, 38
mocca.campaign
    module, 11
mocca.campaign.experiment_funcs
    module, 9
mocca.campaign.process_funcs
    module, 10
mocca.campaign.utils
    module, 10
mocca.chromatogram
    module, 13
mocca.chromatogram.assign
    module, 11
mocca.chromatogram.correct
    module, 12
mocca.chromatogram.model
    module, 12
mocca.chromatogram.preprocessor
    module, 13
mocca.chromatogram.quantify
    module, 13
mocca.chromatogram.utils
    module, 13
mocca.components
    module, 16
mocca.components.databases
    module, 13
mocca.components.models
    module, 14

mocca.components.quali_funcs
    module, 15
mocca.components.quant_funcs
    module, 15
mocca.components.utils
    module, 16
mocca.dad_data
    module, 23
mocca.dad_data.apis
    module, 19
mocca.dad_data.apis.allotrope
    module, 16
mocca.dad_data.apis.angi
    module, 17
mocca.dad_data.apis.chemstation
    module, 18
mocca.dad_data.apis.custom
    module, 18
mocca.dad_data.apis.empower
    module, 19
mocca.dad_data.apis.labsolutions
    module, 19
mocca.dad_data.models
    module, 19
mocca.dad_data.process_funcs
    module, 21
mocca.dad_data.process_gradientdata
    module, 22
mocca.dad_data.utils
    module, 23
mocca.decomposition
    module, 23
mocca.decomposition.alternative_objective_funcs
    module, 23
mocca.decomposition.data_tensor
    module, 24
mocca.decomposition.iterative_parafac
    module, 25
mocca.decomposition.model
    module, 25
mocca.decomposition.parafac_funcs
    module, 26
mocca.decomposition.utils
    module, 27
mocca.peak
    module, 33
mocca.peak.check
    module, 27
mocca.peak.correct
    module, 27
mocca.peak.database
    module, 28
mocca.peak.expand
    module, 28
mocca.peak.integrate
    module, 28
mocca.peak.match
    module, 28
mocca.peak.models
    module, 29
mocca.peak.process
    module, 31
mocca.peak.purity_funcs
    module, 31
mocca.peak.quantify
    module, 32
mocca.peak.resolve_impure
    module, 32
mocca.peak.utils
    module, 32
mocca.report
    module, 33
mocca.report.utils
    module, 33
mocca.user_interaction
    module, 34
mocca.user_interaction.campaign
    module, 34
mocca.user_interaction.settings
    module, 34
mocca.user_interaction.suggest_calibration
    module, 35
mocca.user_interaction.user_objects
    module, 35
mocca.visualization
    module, 38
mocca.visualization.basic_plots
    module, 36
mocca.visualization.calibration_plots
    module, 37
mocca.visualization.parafac_plots
    module, 37
mocca.visualization.results_plot
    module, 37
mocca.visualization.utils
    module, 38
module
    mocca, 38
    mocca.campaign, 11
    mocca.campaign.experiment_funcs, 9
    mocca.campaign.process_funcs, 10
    mocca.campaign.utils, 10
    mocca.chromatogram, 13
    mocca.chromatogram.assign, 11
    mocca.chromatogram.correct, 12
    mocca.chromatogram.model, 12
    mocca.chromatogram.preprocessor, 13
    mocca.chromatogram.quantify, 13

mocca.chromatogram.utils, 13
mocca.components, 16
mocca.components.databases, 13
mocca.components.models, 14
mocca.components.quali_funcs, 15
mocca.components.quant_funcs, 15
mocca.components.utils, 16
mocca.dad_data, 23
mocca.dad_data.apis, 19
mocca.dad_data.apis.allotrope, 16
mocca.dad_data.apis.angi, 17
mocca.dad_data.apis.chemstation, 18
mocca.dad_data.apis.custom, 18
mocca.dad_data.apis.empower, 19
mocca.dad_data.apis.labsolutions, 19
mocca.dad_data.models, 19
mocca.dad_data.process_funcs, 21
mocca.dad_data.process_gradientdata, 22
mocca.dad_data.utils, 23
mocca.decomposition, 23
mocca.decomposition.alternative_objective_funcs, 23
mocca.decomposition.data_tensor, 24
mocca.decomposition.iterative_parafac, 25
mocca.decomposition.model, 25
mocca.decomposition.parafac_funcs, 26
mocca.decomposition.utils, 27
mocca.peak, 33
mocca.peak.check, 27
mocca.peak.correct, 27
mocca.peak.database, 28
mocca.peak.expand, 28
mocca.peak.integrate, 28
mocca.peak.match, 28
mocca.peak.models, 29
mocca.peak.process, 31
mocca.peak.purity_funcs, 31
mocca.peak.quantify, 32
mocca.peak.resolve_impure, 32
mocca.peak.utils, 32
mocca.report, 33
mocca.report.utils, 33
mocca.user_interaction, 34
mocca.user_interaction.campaign, 34
mocca.user_interaction.settings, 34
mocca.user_interaction.suggest_calibration, 35
mocca.user_interaction.user_objects, 35
mocca.visualization, 38
mocca.visualization.basic_plots, 36
mocca.visualization.calibration_plots, 37
mocca.visualization.parafac_plots, 37
mocca.visualization.results_plot, 37
mocca.visualization.utils, 38

N

n_comps (*mocca.decomposition.model.ParafacModel attribute*), 26
normalize_peak_data() (*in module mocca.decomposition.data_tensor*), 25

O

offset (*mocca.components.models.QualiComponent attribute*), 14
offset (*mocca.peak.models.BasePeak attribute*), 29
offset (*mocca.peak.models.IstdPeak attribute*), 30
offset (*mocca.peak.models.ProcessedPeak attribute*), 31
original_data (*mocca.dad_data.models.GradientData attribute*), 20

P

parafac() (*in module mocca.decomposition.parafac_funcs*), 26
parafac_comp_tensor (*mocca.dad_data.models.ParafacData attribute*), 20
ParafacData (*class in mocca.dad_data.models*), 20
ParafacModel (*class in mocca.decomposition.model*), 25
path (*mocca.dad_data.models.CompoundData attribute*), 20
path (*mocca.dad_data.models.DadData attribute*), 20
path (*mocca.user_interaction.user_objects.Gradient attribute*), 36
path (*mocca.user_interaction.user_objects.HplcInput attribute*), 36
pca_explained_variance (*mocca.decomposition.model.ParafacModel attribute*), 26
PeakDatabase (*class in mocca.peak.database*), 28
peaks (*mocca.decomposition.model.ParafacModel attribute*), 26
peaks_high_pass (*mocca.user_interaction.settings.Settings attribute*), 35
peaks_low_pass (*mocca.user_interaction.settings.Settings attribute*), 35
pick_peaks() (*in module mocca.dad_data.process_funcs*), 21
PickedPeak (*class in mocca.peak.models*), 30
plot_1D_data() (*in module mocca.visualization.basic_plots*), 36
plot_1D_layer() (*in module mocca.visualization.basic_plots*), 36
plot_1D_scatter() (*in module mocca.visualization.basic_plots*), 36
plot_1D_scatter_layer() (*in module mocca.visualization.basic_plots*), 37

plot_aligned_tensor() (*in module mocca.visualization.parafac_plots*), 37
plot_calibration_curves() (*in module mocca.visualization.calibration_plots*), 37
plot_chrom_with_peaks() (*in module mocca.visualization.results_plot*), 37
plot_impure_peak_spectra() (*in module mocca.visualization.parafac_plots*), 37
plot_normalized_integrals() (*in module mocca.visualization.parafac_plots*), 37
plot_objective_func() (*in module mocca.visualization.parafac_plots*), 37
plot_retention() (*in module mocca.visualization.parafac_plots*), 37
plot_uvvis_specs() (*in module mocca.visualization.parafac_plots*), 37
predict_purity_unimodal() (*in module mocca.peak.purity_funcs*), 32
preprocess_chromatogram() (*in module mocca.chromatogram.preprocessor*), 13
preprocess_df() (*in module mocca.dad_data.apis.allotrope*), 17
preprocess_experiment() (*in module mocca.campaign.process_funcs*), 10
PreprocessedPeak (*class in mocca.peak.models*), 30
print_parafac_analytics() (*in module mocca.decomposition.parafac_funcs*), 26
process_all_hplc_input() (*mocca.user_interaction.campaign.HplcDadCampaign method*), 34
process_compound_exp() (*in module mocca.campaign.process_funcs*), 10
process_compound_experiments() (*in module mocca.campaign.process_funcs*), 10
process_experiments() (*in module mocca.campaign.process_funcs*), 10
process_gradients() (*in module mocca.campaign.process_funcs*), 10
process_new_hplc_input() (*mocca.user_interaction.campaign.HplcDadCampaign method*), 34
process_peak() (*in module mocca.peak.process*), 31
processed (*mocca.user_interaction.user_objects.HplcInput attribute*), 36
ProcessedPeak (*class in mocca.peak.models*), 30
pure (*mocca.peak.models.CheckedPeak attribute*), 29
pure (*mocca.peak.models.ProcessedPeak attribute*), 31

Q

QualiComponent (*class in mocca.components.models*), 14
QualiComponentDatabase (*class in mocca.components.databases*), 14
QuantComponent (*class in mocca.components.models*), 14
QuantComponentDatabase (*class in mocca.components.databases*), 14
quantify_peak() (*in module mocca.peak.quantify*), 32
quantify_peaks() (*in module mocca.chromatogram.quantify*), 13

R

read_adf() (*in module mocca.dad_data.apis.allotrope*), 17
read_adf_datacube() (*in module mocca.dad_data.apis.allotrope*), 17
read_adf_description() (*in module mocca.dad_data.apis.allotrope*), 17
read_angi() (*in module mocca.dad_data.apis.angi*), 17
read_arw_empower() (*in module mocca.dad_data.apis.empower*), 19
read_chemstation() (*in module mocca.dad_data.apis.chemstation*), 18
read_csv_agilent() (*in module mocca.dad_data.apis.chemstation*), 18
read_csv_angi() (*in module mocca.dad_data.apis.angi*), 17
read_custom_data() (*in module mocca.dad_data.apis.custom*), 18
read_empower() (*in module mocca.dad_data.apis.empower*), 19
read_labsolutions() (*in module mocca.dad_data.apis.labsolutions*), 19
read_txt_shimadzu() (*in module mocca.dad_data.apis.labsolutions*), 19
reassign_impurities() (*in module mocca.chromatogram.assign*), 11
relative_distance_thresh (*mocca.user_interaction.settings.Settings attribute*), 35
relevant_comp (*mocca.decomposition.model.DataTensor attribute*), 25
right (*mocca.components.models.QualiComponent attribute*), 14
right (*mocca.peak.models.BasePeak attribute*), 29
right (*mocca.peak.models.IstdPeak attribute*), 30
right (*mocca.peak.models.ProcessedPeak attribute*), 31
round_to_n() (*in module mocca.visualization.utils*), 38

S

saturation (*mocca.peak.models.CheckedPeak attribute*), 29
saturation (*mocca.peak.models.ProcessedPeak attribute*), 31
save_campaign() (*mocca.user_interaction.campaign.HplcDadCampaign method*), 34
Settings (*class in mocca.user_interaction.settings*), 34

settings_to_df() (*in module mocca.report.utils*), 33
shift (*mocca.dad_data.models.ParafacData attribute*), 20
sort_peaks_by_best_match() (*in module mocca.chromatogram.assign*), 11
sort_peaks_by_compound() (*in module mocca.components.utils*), 16
spectrum (*mocca.components.models.QualiComponent attribute*), 14
spectrum_correl_thresh (*mocca.user_interaction.settings.Settings attribute*), 35
spectrum_max (*mocca.components.models.QualiComponent attribute*), 14
suggest_initialization_runs() (*in module mocca.user_interaction.suggest_calibration*), 35
sum_absorbance_by_time() (*in module mocca.dad_data.utils*), 23

T

tensor (*mocca.decomposition.model.DataTensor attribute*), 25
tidy_df_agilent() (*in module mocca.dad_data.apis.angi*), 17
tidy_df_agilent() (*in module mocca.dad_data.apis.chemstation*), 18
time (*mocca.dad_data.models.CompoundData attribute*), 20
time (*mocca.dad_data.models.DadData attribute*), 20
time (*mocca.user_interaction.user_objects.CustomData attribute*), 35
trim_data() (*in module mocca.dad_data.utils*), 23

U

update() (*mocca.components.databases.QualiComponentDatabase method*), 14
update() (*mocca.components.databases.QuantComponentDatabase method*), 14
update_matches() (*in module mocca.peak.match*), 29
update_peaks_and_matches() (*in module mocca.chromatogram.assign*), 12
update_unknown_counter() (*mocca.peak.database.PeakDatabase method*), 28

W

warnings (*mocca.dad_data.models.CompoundData attribute*), 20
warnings (*mocca.dad_data.models.DadData attribute*), 20
wavelength (*mocca.dad_data.models.CompoundData attribute*), 20
wavelength (*mocca.dad_data.models.DadData attribute*), 20
wavelength (*mocca.user_interaction.user_objects.CustomData attribute*), 35
weights (*mocca.decomposition.model.ParafacModel attribute*), 26
wl_high_pass (*mocca.dad_data.models.DadData attribute*), 20
wl_high_pass (*mocca.user_interaction.settings.Settings attribute*), 35
wl_low_pass (*mocca.dad_data.models.DadData attribute*), 20
wl_low_pass (*mocca.user_interaction.settings.Settings attribute*), 35

Y

y_offset (*mocca.dad_data.models.ParafacData attribute*), 21
y_offset (*mocca.decomposition.model.DataTensor attribute*), 25