pyrawr

Release 0.1.0

RalfG

Apr 22, 2021

CONTENTS

Python wrapper around the ThermoRawFileParser command line interface.

This Python module uses the ThermoRawFileParser CLI to retrieve general run metadata, specific spectra, or specific XICs, directly as Python lists and dictionaries from mass spectrometry raw files. Parsing raw files to other file formats is also supported.

CHAPTER

ONE

INSTALLATION

- Install pyrawr with pip

```
$ pip install pyrawr
```

- Install ThermoRawFileParser or Docker.

For Docker, the current user must be added to the Docker group, that is, be callable as `docker run`, instead of `sudo docker run`.

CHAPTER

TWO

USAGE

See full API documentation for all pyrawr functionality.

Parse raw file to any supported output format:

```
>>> from pyrawr import ThermoRawFileParser
>>> trfp = ThermoRawFileParser(
...     executable="thermorawfileparser",
...     docker_image="quay.io/biocontainers/thermorawfileparser:1.3.3--ha8f3691_1"
... )
>>> trfp.parse("OR4_110719_OB_PAR14_sSCX_fr10.raw", output_format="mzml")
```

Get raw file metadata:

```
>>> trfp.metadata("OR4_110719_OB_PAR14_sSCX_fr10.raw")
{'FileProperties': [{'accession': 'NCIT:C47922', 'cvLabel': 'NCIT' ... }]}
```

Query a specific spectrum:

```
>>> trfp.query("OR4_110719_OB_PAR14_sSCX_fr10.raw", "508,680")
[{'mzs': [204.8467254638672,
   262.4529113769531,
   309.53961181640625,
   ...
```

Retrieve one or more chromatograms based on a query:

```
>>> trfp.xic(
...     "OR4_110719_OB_PAR14_sSCX_fr10.raw",
...     [{"mz":488.5384, "tolerance":10, "tolerance_unit":"ppm"}],
... )
{'OutputMeta': {'base64': False, 'timeunit': 'minutes'},
 'Content': [{'Meta': {'MzStart': 488.53351461600005,
    'MzEnd': 488.543285384,
    'RtStart': 0.007536666666666666,
    'RtEnd': 179.99577166666666},
   'RetentionTimes': [0.007536666666666666,
    0.022771666666666666,
    0.03693666666666666,
    ...
```

CHAPTER

THREE

CONTRIBUTING

Bugs, questions or suggestions? Feel free to post an issue in the issue tracker or to make a pull request! See Contributing.md for more info.

This module currently uses Python's `subprocess.run()` to access ThermoRawFileParser. There are probably much better methods that would directly access the ThermoRawFileParser library, circumventing the CLI. Suggestions and PRs are always welcome.

CHAPTER

FOUR

CHANGELOG

See Changelog.

4.1 API

Python wrapper around the ThermoRawFileParser command line interface.

class pyrawr.**ThermoRawFileParser**(*executable='thermorawfileparser'*, *docker_image=None*)

Wrapper around ThermoRawFileParser CLI.

Parameters

- **executable** (*str*) – ThermoRawFileParser shell command. For example, `thermorawfileparser` or `mono ThermoRawFileParser.exe`.
- **docker_image** (*str, optional*) – Docker image with ThermoRawFileParser. Requires the `docker run` CLI command.

version_requirement

ThermoRawFile semver version requirement.

Type str

installed_version

Installed ThermoRawFileParser version.

Type str

Examples

```
>>> from pyrawr import ThermoRawFileParser
>>> trfp = ThermoRawFileParser(
...     executable="thermorawfileparser",
...     docker_image="quay.io/biocontainers/thermorawfileparser:1.3.3--ha8f3691_1"
... )
>>> trfp.parse("OR4_110719_OB_PAR14_sSCX_fr10.raw")
```

```
>>> trfp.metadata("OR4_110719_OB_PAR14_sSCX_fr10.raw")
{'FileProperties': [{'accession': 'NCIT:C47922', 'cvLabel': 'NCIT' ... }]}
```

```
>>> trfp.query("OR4_110719_OB_PAR14_sSCX_fr10.raw", "508,680")
[{'mzs': [204.8467254638672,
262.4529113769531,
```

(continues on next page)

(continued from previous page)

```
309.53961181640625,
...
```

```
>>> trfp.xic(
...     "OR4_110719_OB_PAR14_sSCX_fr10.raw",
...     [{"mz":488.5384, "tolerance":10, "tolerance_unit":"ppm"}],
... )
{'OutputMeta': {'base64': False, 'timeunit': 'minutes'},
'Content': [{'Meta': {'MzStart': 488.53351461600005,
    'MzEnd': 488.543285384,
    'RtStart': 0.007536666666666666,
    'RtEnd': 179.99577166666666},
'RetentionTimes': [0.007536666666666666,
    0.022771666666666666,
    0.036936666666666666,
    ...
```

parse(*input_file*, *output_format=None*, *options=None*)
Parse raw file to one of the supported output formats.

Parameters

- **input_file** (*str*) – Path to input file.
- **output_format** (*str, optional*) – Output format. One of `mgf`, `mzml`, `indexed_mzml`, `parquet`, or `scan_info`.
- **options** (*List*) – List of other CLI options to pass to ThermoRawFileParser. See ThermoRawFileParser docs for more info.

metadata(*input_file*)
Get raw file metadata as Python dictionary.

Parameters **input_file** (*str*) – Path to input file.

Returns **metadata** – Dictionary containing raw file metadata.

Return type dict

query(*input_file*, *scans*, *options=None*)
Retrieve specific spectra by scan number in ProXI format.

Parameters

- **input_file** (*str*) – Path to input file.
- **scans** (*str*) – Scan numbers, e.g. `1-5, 20, 25-30`.
- **options** (*list*) – List of other CLI options to pass to ThermoRawFileParser. See ThermoRawFileParser docs for more info.

Returns **spectra** – List of spectra in ProXI format.

Return type list

xic(*input_file*, *query*, *base64=False*)
Retrieve one or more chromatograms based on a query.

Parameters

- **input_file** (*str*) – Path to input file.

- **query** (*list*) – List of query dictionaries. See ThermoRawFileParser docs for more info.
- **base64** (*boolean, optional*) – Encode the content of the xic vectors as base 64.

Returns **xic** – Dictionary containing XICs.

Return type dict

exception pyrawr.**ThermoRawFileParserError**
General ThermoRawFileParser error.

exception pyrawr.**ThermoRawFileParserInstallationError**
Could not find ThermoRawFileParser CLI.

exception pyrawr.**ThermoRawFileParserRunError**
Error running ThermoRawFileParser.

4.2 Changelog

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog, and this project adheres to Semantic Versioning.

4.2.1 [0.1.0] - 22/04/2021

- Initial release

4.3 Contributing

This document briefly describes how to contribute to pyrawr.

4.3.1 Before you begin

If you have an idea for a feature, use case to add or an approach for a bugfix, you are welcome to communicate it with the community by creating an issue in GitHub issues.

4.3.2 How to contribute

- Fork pyrawr on GitHub to make your changes.
- Commit and push your changes to your fork.
- Open a pull request with these changes. You pull request message ideally should include:
  - A description of why the changes should be made.
  - A description of the implementation of the changes.
  - A description of how to test the changes.
- The pull request should pass all the continuous integration tests which are automatically run by GitHub Actions.

4.3.3 Development setup

1. Setup Python 3 and Poetry
2. Clone the pyrawr repository and run `poetry install` to setup the development environment.

4.3.4 Development workflow

- When a new version is ready to be published:
  1. Change the `__version__` in `pyproject.toml` following semantic versioning.
  2. Update the documentation (`README.md`), if required.
  3. Update the changelog (if not already done) in `CHANGELOG.md` according to Keep a Changelog.
  4. Commit all final changes to the `master` branch.
  5. On `master`, set a new tag with the version number, e.g. `git tag v0.1.5`.
  6. Push to GitHub, with the tag: `git push; git push --tags`.
- When a new tag is pushed to (or made on) GitHub that matches `v*`, the following GitHub Actions are triggered:
  1. The Python package is build and published to PyPI.
  2. Using the Git Release action, a new GitHub release is made with the changes that are listed in `CHANGELOG.md`.

PYTHON MODULE INDEX

p

INDEX