mwtab Documentation

Release 1.2.5.post1

Christian D. Powell, Andrey Smelter, Hunter N.B. Moseley

May 12, 2022

Contents:

CHAPTER 1

mwtab

The `mwtab` package is a Python library that facilitates reading and writing files in `mwTab` format used by the Metabolomics Workbench for archival of Mass Spectrometry (MS) and Nuclear Magnetic Resonance (NMR) experimental data.

The `mwtab` package provides facilities to convert `mwTab` formatted files into their equivalent `JSON` ized representation and vice versa. `JSON` stands for JavaScript Object Notation, an open-standard format that uses human-readable text to transmit data objects consisting of attribute-value pairs.

The mwtab package can be used in several ways:

- As a library for accessing and manipulating data stored in mwTab format files.
- As a command-line tool to convert between mwTab format and its equivalent JSON representation.

1.1 Citation

When using mwtab package in published work, please cite the following papers:

- Powell, Christian D., and Hunter NB Moseley. "The mwtab Python Library for RESTful Access and Enhanced Quality Control, Deposition, and Curation of the Metabolomics Workbench Data Repository." *Metabolites* 11.3 (2021): 163. doi: 10.3390/metabo11030163.
- Smelter, Andrey and Hunter NB Moseley. "A Python library for FAIRer access and deposition to the Metabolomics Workbench Data Repository." *Metabolomics* 2018, 14(5): 64. doi: 10.1007/s11306-018-1356-6.

1.2 Links

- mwtab @ GitHub
- mwtab @ PyPI
- Documentation @ ReadTheDocs

1.3 Installation

The mwtab package runs under Python 3.5+. Use pip to install. Starting with Python 3.4, pip is included by default.

1.3.1 Install on Linux, Mac OS X

```
python3 -m pip install mwtab
```

1.3.2 Install on Windows

```
py -3 -m pip install mwtab
```

1.3.3 Upgrade on Linux, Mac OS X

```
python3 -m pip install mwtab --upgrade
```

1.3.4 Upgrade on Windows

```
py -3 -m pip install mwtab --upgrade
```

1.4 Quickstart

```
>>> import mwtab
>>>
>>> # Here we use ANALYSIS_ID of file to fetch data from URL
>>> for mwfile in mwtab.read_files("1", "2"):
...       print("STUDY_ID:", mwfile.study_id)
...       print("ANALYSIS_ID:", mwfile.analysis_id)
...       print("SOURCE:", mwfile.source)
...       print("Blocks:", list(mwfile.keys()))
>>>
```

Note: Read the User Guide and the `mwtab` Tutorial on ReadTheDocs to learn more and to see code examples on using the `mwtab` as a library and as a command-line tool.

1.5 License

This package is distributed under the BSD *license*.

CHAPTER 2

Documentation index:

2.1 User Guide

2.1.1 Description

The `mwtab` package is a Python library that facilitates reading and writing files in `mwTab` format used by the Metabolomics Workbench for archival of Mass Spectrometry (MS) and Nuclear Magnetic Resonance (NMR) experimental data.

The `mwtab` package provides facilities to convert `mwTab` formatted files into their equivalent JSONized (JavaScript Object Notation, an open-standard format that uses human-readable text to transmit data objects consisting of attribute-value pairs) representation and vice versa.

The `mwtab` package can be used in several ways:

- As a library for accessing and manipulating data stored in `mwTab` format files.
- As a command-line tool to convert between `mwTab` format and its equivalent `JSON` representation.

2.1.2 Installation

The *`mwtab`* package runs under Python 2.7 and Python 3.4+. Starting with Python 3.4, pip is included by default. To install system-wide with pip run the following:

Install on Linux, Mac OS X

```
python3 -m pip install mwtab
```

Install on Windows

```
py -3 -m pip install mwtab
```

Install inside virtualenv

For an isolated install, you can run the same inside a virtualenv.

```
$ virtualenv -p /usr/bin/python3 venv  # create virtual environment, use python3␣
↪interpreter

$ source venv/bin/activate             # activate virtual environment

$ python3 -m pip install mwtab         # install mwtab as usual

$ deactivate                           # if you are done working in the virtual␣
↪environment
```

2.1.3 Get the source code

Code is available on GitHub: https://github.com/MoseleyBioinformaticsLab/mwtab

You can either clone the public repository:

```
$ https://github.com/MoseleyBioinformaticsLab/mwtab.git
```

Or, download the tarball and/or zipball:

```
$ curl -OL https://github.com/MoseleyBioinformaticsLab/mwtab/tarball/master

$ curl -OL https://github.com/MoseleyBioinformaticsLab/mwtab/zipball/master
```

Once you have a copy of the source, you can embed it in your own Python package, or install it into your system site-packages easily:

```
$ python3 setup.py install
```

2.1.4 Dependencies

The *mwtab* package depends on several Python libraries. The `pip` command will install all dependencies automatically, but if you wish to install them manually, run the following commands:

- **docopt for creating *mwtab* command-line interface.**
  - To install docopt run the following:

```
python3 -m pip install docopt  # On Linux, Mac OS X
py -3 -m pip install docopt    # On Windows
```

- **schema for validating functionality of `mwTab` files based on `JSON` schema.**
  - To install the schema Python library run the following:

```
python3 -m pip install schema  # On Linux, Mac OS X
py -3 -m pip install schema    # On Windows
```

2.1.5 Basic usage

The *mwtab* package can be used in several ways:

- As a library for accessing and manipulating data stored in mwTab formatted files.
  - Create the *MWTabFile* generator function that will generate (yield) a single *MWTabFile* instance at a time.
  - Process each *MWTabFile* instance:
    - Process mwTab files in a for-loop, one file at a time.
    - Process as an iterator calling the `next()` built-in function.
    - Convert the generator into a `list` of *MWTabFile* objects.
- As a command-line tool:
  - Convert from mwTab file format into its equivalent JSON file format and vice versa.
  - Validate data stored in mwTab file based on schema definition.

Note: Read *The mwtab Tutorial* to learn more and see code examples on using the *mwtab* as a library and as a command-line tool.

2.2 The mwtab Tutorial

The *mwtab* package provides classes and other facilities for downloading, parsing, accessing, and manipulating data stored in either the mwTab or JSON representation of mwTab files.

Also, the *mwtab* package provides simple command-line interface to convert between mwTab and JSON representations, download entries from Metabolomics Workbench, access the MW REST interface, validate the consistency of the mwTab files, or extract metadata and metabolites from these files.

2.2.1 Brief mwTab Format Overview

Note: For full official specification see the following link (`mwTab file specification`): http://www.metabolomicsworkbench.org/data/tutorials.php

The mwTab formatted files consist of multiple blocks. Each new block starts with #.

- Some of the blocks contain only "key-value"-like pairs.

```
#METABOLOMICS WORKBENCH STUDY_ID:ST000001 ANALYSIS_ID:AN000001
VERSION             1
CREATED_ON          2016-09-17
#PROJECT
PR:PROJECT_TITLE                    FatB Gene Project
```

(continues on next page)

(continued from previous page)

```
PR:PROJECT_TYPE                     Genotype treatment
PR:PROJECT_SUMMARY                  Experiment to test the consequence of a mutation␣
↪at the FatB gene (At1g08510)
PR:PROJECT_SUMMARY                  the wound-response of Arabidopsis
```

Note: *_SUMMARY "key-value"-like pairs are typically span through multiple lines.

- #SUBJECT_SAMPLE_FACTORS block is specially formatted, i.e. it contains header specification and tab-separated values.

```
#SUBJECT_SAMPLE_FACTORS:             SUBJECT(optional)[tab]SAMPLE[tab]FACTORS(NAME:
↪VALUE pairs separated by |)[tab]Additional sample data
SUBJECT_SAMPLE_FACTORS               -       LabF_115873     Arabidopsis Genotype:
↪Wassilewskija (Ws) | Plant Wounding Treatment:Control - Non-Wounded
SUBJECT_SAMPLE_FACTORS               -       LabF_115878     Arabidopsis Genotype:
↪Wassilewskija (Ws) | Plant Wounding Treatment:Control - Non-Wounded
SUBJECT_SAMPLE_FACTORS               -       LabF_115883     Arabidopsis Genotype:
↪Wassilewskija (Ws) | Plant Wounding Treatment:Control - Non-Wounded
SUBJECT_SAMPLE_FACTORS               -       LabF_115888     Arabidopsis Genotype:
↪Wassilewskija (Ws) | Plant Wounding Treatment:Control - Non-Wounded
SUBJECT_SAMPLE_FACTORS               -       LabF_115893     Arabidopsis Genotype:
↪Wassilewskija (Ws) | Plant Wounding Treatment:Control - Non-Wounded
SUBJECT_SAMPLE_FACTORS               -       LabF_115898     Arabidopsis Genotype:
↪Wassilewskija (Ws) | Plant Wounding Treatment:Control - Non-Wounded
```

- #MS_METABOLITE_DATA (results) block contains Samples identifiers, Factors identifiers as well as tab-separated data between *_START and *_END.

```
#MS_METABOLITE_DATA
MS_METABOLITE_DATA:UNITS     Peak height
MS_METABOLITE_DATA_START
Samples      LabF_115904     LabF_115909     LabF_115914     LabF_115919     LabF_
↪115924      LabF_115929     LabF_115842     LabF_115847     LabF_115852     LabF_
↪115857      LabF_115862     LabF_115867     LabF_115873     LabF_115878     LabF_
↪115883      LabF_115888     LabF_115893     LabF_115898     LabF_115811     LabF_
↪115816      LabF_115821     LabF_115826     LabF_115831     LabF_115836
Factors      Arabidopsis Genotype:fatb-ko KD; At1g08510 | Plant Wounding Treatment:
↪Control - Non-Wounded     Arabidopsis Genotype:fatb-ko KD; At1g08510 | Plant␣
↪Wounding Treatment:Control - Non-Wounded     Arabidopsis Genotype:fatb-ko KD;␣
↪At1g08510 | Plant Wounding Treatment:Control - Non-Wounded     Arabidopsis Genotype:
↪fatb-ko KD; At1g08510 | Plant Wounding Treatment:Control - Non-Wounded     ␣
↪Arabidopsis Genotype:fatb-ko KD; At1g08510 | Plant Wounding Treatment:Control - Non-
↪Wounded     Arabidopsis Genotype:fatb-ko KD; At1g08510 | Plant Wounding Treatment:
↪Control - Non-Wounded     Arabidopsis Genotype:fatb-ko KD; At1g08510 | Plant␣
↪Wounding Treatment:Wounded   Arabidopsis Genotype:fatb-ko KD; At1g08510 | Plant␣
↪Wounding Treatment:Wounded   Arabidopsis Genotype:fatb-ko KD; At1g08510 | Plant␣
↪Wounding Treatment:Wounded   Arabidopsis Genotype:fatb-ko KD; At1g08510 | Plant␣
↪Wounding Treatment:Wounded   Arabidopsis Genotype:fatb-ko KD; At1g08510 | Plant␣
↪Wounding Treatment:Wounded   Arabidopsis Genotype:fatb-ko KD; At1g08510 | Plant␣
↪Wounding Treatment:Wounded   Arabidopsis Genotype:Wassilewskija (Ws) | Plant␣
↪Wounding Treatment:Control - Non-Wounded        Arabidopsis Genotype:Wassilewskija␣
↪ (Ws) | Plant Wounding Treatment:Control - Non-Wounded        Arabidopsis Genotype:
↪Wassilewskija (Ws) | Plant Wounding Treatment:Control - Non-Wounded        ␣
↪Arabidopsis Genotype:Wassilewskija (Ws) | Plant Wounding Treatment:Control - Non-
↪Wounded        Arabidopsis Genotype:Wassilewskija (Ws) | Plant Wounding Treatment:
↪Control - Non-Wounded        Arabidopsis Genotype:Wassilewskija (Ws) | Plant␣
↪Wounding Treatment:Control - Non-Wounded        Arabidopsis Genotype:Wassilewskija␣
↪ (Ws) | Plant Wounding Treatment:Wounded      Arabidopsis Genotype:Wassilewskija␣
↪ (Ws) | Plant Wounding Treatment:Wounded      Arabidopsis Genotype:Wassilewskija␣
↪ (Ws) | Plant Wounding Treatment:Wounded      Arabidopsis Genotype:Wassilewskija␣
↪ (Ws) | Plant Wounding Treatment:Wounded      Arabidopsis Genotype:Wassilewskija␣
↪ (Ws) | Plant Wounding Treatment:Wounded      Arabidopsis Genotype:Wassilewskija␣
```

(continues on next page)

(continued from previous page)

```
1_2_4-benzenetriol   1874.0000       3566.0000       1945.0000       1456.0000
↪2004.0000       1995.0000       4040.0000       2432.0000       2189.0000
↪1931.0000       1307.0000       2880.0000       2218.0000       1754.0000
↪1369.0000       1201.0000       3324.0000       1355.0000       2257.0000
↪1718.0000       1740.0000       3472.0000       2054.0000       1367.0000
1-monostearin        987.0000        450.0000       1910.0000        549.0000
↪1032.0000       902.0000        393.0000        705.0000        100.0000        481.
↪0000        265.0000        120.0000        1185.0000       867.0000        676.
↪0000        569.0000        579.0000        387.0000        1035.0000       789.
↪0000        875.0000        224.0000        641.0000        693.0000
...
MS_METABOLITE_DATA_END
```

- `#METABOLITES` metadata block contains a header specifying fields and tab-separated data between `*_START` and `*_END`.

```
#METABOLITES
METABOLITES_START
metabolite_name      moverz_quant    ri      ri_type pubchem_id      inchi_key
↪kegg_id other_id        other_id_type
1,2,4-benzenetriol   239     522741  Fiehn   10787           C02814  205673  BinBase
1-monostearin        399     959625  Fiehn   107036          D01947  202835  BinBase
2-hydroxyvaleric acid        131     310750  Fiehn   98009                   218773
↪BinBase
3-phosphoglycerate   299     611619  Fiehn   724             C00597  217821  BinBase
...
METABOLITES_END
```

- `#NMR_BINNED_DATA` metadata block contains a header specifying fields and tab-separated data between `*_START` and `*_END`.

```
#NMR_BINNED_DATA
NMR_BINNED_DATA_START
Bin range(ppm)       CDC029  CDC030  CDC032  CPL101  CPL102  CPL103  CPL201  CPL202
↪CPL203  CDS039  CDS052  CDS054
0.50...0.56  0.00058149      1.6592  0.039301        0       0       0       0.034018
↪       0.0028746       0.0021478       0.013387        0       0
0.56...0.58  0       0.74267 0       0.007206        0       0       0       0
↪0       0       0       0.0069721
0.58...0.60  0.051165        0.8258  0.089149        0.060972        0.026307
↪0.045697        0.069541        0       0       0.14516 0.057489        0.042255
...
NMR_BINNED_DATA_END
```

- Order of metadata and data blocks (MS)

```
#METABOLOMICS WORKBENCH
VERSION             1
CREATED_ON          2016-09-17
...
#PROJECT
...
#STUDY
...
#SUBJECT
...
```

(continues on next page)

(continued from previous page)

```
#SUBJECT_SAMPLE_FACTORS:             SUBJECT(optional)[tab]SAMPLE[tab]FACTORS(NAME:
↪VALUE pairs separated by |)[tab]Additional sample data
...
#COLLECTION
...
#TREATMENT
...
#SAMPLEPREP
...
#CHROMATOGRAPHY
...
#ANALYSIS
...
#MS
...
#MS_METABOLITE_DATA
MS_METABOLITE_DATA:UNITS     peak area
MS_METABOLITE_DATA_START
...
MS_METABOLITE_DATA_END
#METABOLITES
METABOLITES_START
...
METABOLITES_END
#END
```

2.2.2 Using mwtab as a Library

Importing mwtab Package

If the *mwtab* package is installed on the system, it can be imported:

[1]:
```
import mwtab
```

Constructing MWTabFile Generator

The *fileio* module provides the *read_files()* generator function that yields *MWTabFile* instances. Constructing a *MWTabFile* generator is easy - specify the path to a local mwTab file, directory of files, archive of files:

[2]:
```
import mwtab

mwfile_gen = mwtab.read_files("ST000017_AN000035.txt")  # single mwTab file
mwfiles_gen = mwtab.read_files("ST000017_AN000035.txt", "ST000040_AN000060.json")  #␣
↪several mwTab files
mwdir_gen = mwtab.read_files("mwfiles_dir_mwtab")  # directory of mwTab files
mwzip_gen = mwtab.read_files("mwfiles_mwtab.zip")  # archive of mwTab files
mwanalysis_gen = mwtab.read_files("35", "60")       # ANALYSIS_ID of mwTab files
# REST callable url of mwTab file
mwurl_gen = mwtab.read_files("https://www.metabolomicsworkbench.org/rest/study/
↪analysis_id/AN000035/mwtab/txt")
```

Processing MWTabFile Generator

The *MWTabFile* generator can be processed in several ways:

- Feed it to a for-loop and process one file at a time:

[3]:
```
for mwfile in mwtab.read_files("35", "60"):
    print("STUDY_ID:", mwfile.study_id)       # print STUDY_ID
    print("ANALYSIS_ID", mwfile.analysis_id)  # print ANALYSIS_ID
    print("SOURCE", mwfile.source)            # print source
    for block_name in mwfile:                 # print names of blocks
        print("\t", block_name)
```

```
STUDY_ID: ST000017
ANALYSIS_ID AN000035
SOURCE https://www.metabolomicsworkbench.org/rest/study/analysis_id/AN000035/mwtab/txt
         METABOLOMICS WORKBENCH
         PROJECT
         STUDY
         SUBJECT
         SUBJECT_SAMPLE_FACTORS
         COLLECTION
         TREATMENT
         SAMPLEPREP
         CHROMATOGRAPHY
         ANALYSIS
         MS
         MS_METABOLITE_DATA
STUDY_ID: ST000040
ANALYSIS_ID AN000060
SOURCE https://www.metabolomicsworkbench.org/rest/study/analysis_id/AN000060/mwtab/txt
         METABOLOMICS WORKBENCH
         PROJECT
         STUDY
         SUBJECT
         SUBJECT_SAMPLE_FACTORS
         COLLECTION
         TREATMENT
         SAMPLEPREP
         CHROMATOGRAPHY
         ANALYSIS
         MS
         MS_METABOLITE_DATA
```

Note: Once the generator is consumed, it becomes empty and needs to be created again.

- Since the *MWTabFile* generator behaves like an iterator, we can call the `next()` built-in function:

[4]:
```
mwfiles_generator = mwtab.read_files("35", "60")

mwfile1 = next(mwfiles_generator)
mwfile2 = next(mwfiles_generator)
```

Note: Once the generator is consumed, `StopIteration` will be raised.

- Convert the *MWTabFile* generator into a `list` of *MWTabFile* objects:

[5]:
```
mwfiles_generator = mwtab.read_files("35", "60")
mwfiles_list = list(mwfiles_generator)
```

> **Warning:** Do not convert the *MWTabFile* generator into a `list` if the generator can yield a large number of files, e.g. several thousand, otherwise it can consume all available memory.

Accessing Data From a Single MWTabFile

Since a *MWTabFile* is a Python `collections.OrderedDict`, data can be accessed and manipulated as with any regular Python `dict` object using bracket accessors.

- Accessing top-level "keys" in *MWTabFile*:

[7]:
```
mwfile = next(mwtab.read_files("ST000017_AN000035.txt"))

# list MWTabFile-level keys, i.e. saveframe names
list(mwfile.keys())
```

[7]:
```
['METABOLOMICS WORKBENCH',
 'PROJECT',
 'STUDY',
 'SUBJECT',
 'SUBJECT_SAMPLE_FACTORS',
 'COLLECTION',
 'TREATMENT',
 'SAMPLEPREP',
 'CHROMATOGRAPHY',
 'ANALYSIS',
 'MS',
 'MS_METABOLITE_DATA']
```

- Accessing individual blocks in *MWTabFile*:

[8]:
```
# access "PROJECT" block
mwfile["PROJECT"]
```

[8]:
```
OrderedDict([('PROJECT_TITLE', 'Rat Stamina Studies'),
             ('PROJECT_TYPE', 'Feeding'),
             ('PROJECT_SUMMARY', 'Stamina in rats'),
             ('INSTITUTE', 'University of Michigan'),
             ('DEPARTMENT', 'Internal Medicine'),
             ('LABORATORY', 'Burant Lab'),
             ('LAST_NAME', 'Beecher'),
             ('FIRST_NAME', 'Chris'),
             ('ADDRESS', '-'),
             ('EMAIL', 'chrisbee@med.umich.edu'),
             ('PHONE', '734-232-0815'),
             ('FUNDING_SOURCE', 'NIH: R01 DK077200')])
```

- Accessing individual "key-value" pairs within blocks:

[9]:
```
# access "INSTITUTE" field within "PROJECT" block
mwfile["PROJECT"]["INSTITUTE"]
```

[9]:
```
'University of Michigan'
```

- Accessing data in `#SUBJECT_SAMPLE_FACTORS` block:

[10]:
```
# access "SUBJECT_SAMPLE_FACTORS" block and print first three
mwfile["SUBJECT_SAMPLE_FACTORS"][:3]
```

[10]:
```
[OrderedDict([('Subject ID', '-'),
              ('Sample ID', 'S00009477'),
              ('Factors',
               {'Feeeding': 'Ad lib', 'Running Capacity': 'High'})]),
 OrderedDict([('Subject ID', '-'),
              ('Sample ID', 'S00009478'),
              ('Factors',
               {'Feeeding': 'Ad lib', 'Running Capacity': 'High'})]),
 OrderedDict([('Subject ID', '-'),
              ('Sample ID', 'S00009479'),
              ('Factors',
               {'Feeeding': 'Ad lib', 'Running Capacity': 'High'})])]
```

[11]:
```
# access individual factors (by index)
mwfile["SUBJECT_SAMPLE_FACTORS"][0]
```

[11]:
```
OrderedDict([('Subject ID', '-'),
             ('Sample ID', 'S00009477'),
             ('Factors', {'Feeeding': 'Ad lib', 'Running Capacity': 'High'})])
```

[12]:
```
# access individual fields within factors
mwfile["SUBJECT_SAMPLE_FACTORS"][0]["Sample ID"]
```

[12]:
```
'S00009477'
```

- Accessing data in `#MS_METABOLITE_DATA` block:

[13]:
```
# access data block keys
list(mwfile["MS_METABOLITE_DATA"].keys())
```

[13]:
```
['Units', 'Data', 'Metabolites']
```

[14]:
```
# access units field
mwfile["MS_METABOLITE_DATA"]["Units"]
```

[14]:
```
'peak area'
```

[15]:
```
# access samples field (by index)
mwfile["MS_METABOLITE_DATA"]["Data"][0].keys()
```

[15]:
```
odict_keys(['Metabolite', 'S00009477', 'S00009478', 'S00009479', 'S00009480',
↪'S00009481', 'S00009500', 'S00009501', 'S00009502', 'S00009503', 'S00009470',
↪'S00009471', 'S00009472', 'S00009473', 'S00009474', 'S00009475', 'S00009494',
↪'S00009495', 'S00009496', 'S00009497', 'S00009498', 'S00009499', 'S00009488',
↪'S00009489', 'S00009490', 'S00009491', 'S00009492', 'S00009493', 'S00009509',
↪'S00009510', 'S00009511', 'S00009512', 'S00009513', 'S00009514', 'S00009482',
↪'S00009483', 'S00009484', 'S00009486', 'S00009504', 'S00009505', 'S00009506',
↪'S00009507', 'S00009508'])
```

[16]:
```
# access metabolite data and print first three
mwfile["MS_METABOLITE_DATA"]["Metabolites"][:3]
```

[16]:
```
[OrderedDict([('Metabolite', '11BETA,21-DIHYDROXY-5BETA-PREGNANE-3,20-DIONE'),
              ('moverz_quant', ''),
```

(continues on next page)

(continued from previous page)

```
              ('ri', ''),
              ('ri_type', ''),
              ('pubchem_id', '44263339'),
              ('inchi_key', ''),
              ('kegg_id', 'C05475'),
              ('other_id', '775216_UNIQUE'),
              ('other_id_type', 'UM_Target_ID')]),
 OrderedDict([('Metabolite', '11-BETA-HYDROXYANDROST-4-ENE-3,17-DIONE'),
              ('moverz_quant', ''),
              ('ri', ''),
              ('ri_type', ''),
              ('pubchem_id', '94141'),
              ('inchi_key', ''),
              ('kegg_id', 'C05284'),
              ('other_id', '771312_PRIMARY'),
              ('other_id_type', 'UM_Target_ID')]),
 OrderedDict([('Metabolite', '13(S)-HPODE'),
              ('moverz_quant', ''),
              ('ri', ''),
              ('ri_type', ''),
              ('pubchem_id', '1426'),
              ('inchi_key', ''),
              ('kegg_id', 'C04717'),
              ('other_id', '775541_UNIQUE'),
              ('other_id_type', 'UM_Target_ID')])]
```

Manipulating Data From a Single MWTabFile

In order to change values within *MWTabFile*, descend into the appropriate level using square bracket accessors and set a new value.

- Change regular "key-value" pairs:

[17]:
```
# access phone number information
mwfile["PROJECT"]["PHONE"]
```

[17]:
```
'734-232-0815'
```

[18]:
```
# change phone number information
mwfile["PROJECT"]["PHONE"] = "1-530-754-8258"
```

[19]:
```
# check that it has been modified
mwfile["PROJECT"]["PHONE"]
```

[19]:
```
'1-530-754-8258'
```

- Change `#SUBJECT_SAMPLE_FACTORS` values:

[20]:
```
# access the first subject sample factor by index
mwfile["SUBJECT_SAMPLE_FACTORS"][0]
```

[20]:
```
OrderedDict([('Subject ID', '-'),
             ('Sample ID', 'S00009477'),
             ('Factors', {'Feeeding': 'Ad lib', 'Running Capacity': 'High'})])
```

[21]:
```
# provide additional details to the first subject sample factor
mwfile["SUBJECT_SAMPLE_FACTORS"][0]["Additional sample data"] = {"Additional detail␣
↪key": "Additional detail value"}
```

[22]:
```
# check that it has been modified
mwfile["SUBJECT_SAMPLE_FACTORS"][0]
```

[22]:
```
OrderedDict([('Subject ID', '-'),
             ('Sample ID', 'S00009477'),
             ('Factors', {'Feeeding': 'Ad lib', 'Running Capacity': 'High'}),
             ('Additional sample data',
              {'Additional detail key': 'Additional detail value'})])
```

Printing a MWTabFile and its Components

`MWTabFile` objects provide the `print_file()` method which can be used to output the file in either *mwTab* or JSON format. The method takes a `file_format` keyword argument which specifices the output format to be displayed.

The MWTabFile can be printed to output in *mwTab* format in its entirety using:

- mwfile.print_file(file_format="mwtab")
- Print the first 20 lines in `mwTab` format.

[23]:
```
from io import StringIO
mwtab_file_str = StringIO()
mwfile.print_file(file_format="mwtab", f=mwtab_file_str)

# print out first 20 lines
print("\n".join(mwtab_file_str.getvalue().split("\n")[:20]))
```

```
#METABOLOMICS WORKBENCH STUDY_ID:ST000017 ANALYSIS_ID:AN000035 PROJECT_ID:PR000016
VERSION                  1
CREATED_ON              2016-09-17
#PROJECT
PR:PROJECT_TITLE                        Rat Stamina Studies
PR:PROJECT_TYPE                         Feeding
PR:PROJECT_SUMMARY                      Stamina in rats
PR:INSTITUTE                             University of Michigan
PR:DEPARTMENT                            Internal Medicine
PR:LABORATORY                            Burant Lab
PR:LAST_NAME                            Beecher
PR:FIRST_NAME                           Chris
PR:ADDRESS                               -
PR:EMAIL                                 chrisbee@med.umich.edu
PR:PHONE                                 1-530-754-8258
PR:FUNDING_SOURCE                       NIH: R01 DK077200
#STUDY
ST:STUDY_TITLE                          Rat HCR/LCR Stamina Study
ST:STUDY_TYPE                           LC-MS analysis
ST:STUDY_SUMMARY                        To determine the basis of running capacity and␣
↪health differences in outbread
```

The MWTabFile can be printed to output in JSON format in its entirety using:

- mwfile.print_file(file_format="json")
- Print the first 20 lines in `JSON` format.

[24]:
```
from io import StringIO
mwtab_file_str = StringIO()
mwfile.print_file(file_format="json", f=mwtab_file_str)

# print out first 20 lines
print("\n".join(mwtab_file_str.getvalue().split("\n")[:20]))
```

```
{
    "METABOLOMICS WORKBENCH": {
        "STUDY_ID": "ST000017",
        "ANALYSIS_ID": "AN000035",
        "PROJECT_ID": "PR000016",
        "VERSION": "1",
        "CREATED_ON": "2016-09-17"
    },
    "PROJECT": {
        "PROJECT_TITLE": "Rat Stamina Studies",
        "PROJECT_TYPE": "Feeding",
        "PROJECT_SUMMARY": "Stamina in rats",
        "INSTITUTE": "University of Michigan",
        "DEPARTMENT": "Internal Medicine",
        "LABORATORY": "Burant Lab",
        "LAST_NAME": "Beecher",
        "FIRST_NAME": "Chris",
        "ADDRESS": "-",
        "EMAIL": "chrisbee@med.umich.edu",
        "PHONE": "1-530-754-8258",
```

- Print single block in `mwTab` format.

[25]:
```
mwfile.print_block("STUDY", file_format="mwtab")
```

```
ST:STUDY_TITLE                          Rat HCR/LCR Stamina Study
ST:STUDY_TYPE                           LC-MS analysis
ST:STUDY_SUMMARY                        To determine the basis of running capacity and␣
↪health differences in outbread
ST:STUDY_SUMMARY                        N/NIH rats selected for high capacity (HCR)␣
↪and low capacity (LCR) running (a for
ST:STUDY_SUMMARY                        VO2max) (see:Science. 2005 Jan 21;307(5708):
↪418-20). Plasma collected at 12 of
ST:STUDY_SUMMARY                        age in generation 28 rats after ad lib feeding␣
↪or 40% caloric restriction at week
ST:STUDY_SUMMARY                        8 of age. All animals fasted 4 hours prior to␣
↪collection between 5-8
ST:INSTITUTE                             University of Michigan
ST:DEPARTMENT                            Internal Medicine
ST:LABORATORY                            Burant Lab (MMOC)
ST:LAST_NAME                            Qi
ST:FIRST_NAME                           Nathan
ST:ADDRESS                               -
ST:EMAIL                                 nathanqi@med.umich.edu
ST:PHONE                                 734-232-0815
ST:NUM_GROUPS                           2
ST:TOTAL_SUBJECTS                       42
```

- Print single block in `JSON` format.

[26]:
```
mwfile.print_block("STUDY", file_format="json")
```

```
{
    "STUDY_TITLE": "Rat HCR/LCR Stamina Study",
    "STUDY_TYPE": "LC-MS analysis",
    "STUDY_SUMMARY": "To determine the basis of running capacity and health␣
↪differences in outbread N/NIH rats selected for high capacity (HCR) and low␣
↪capacity (LCR) running (a for VO2max) (see:Science. 2005 Jan 21;307(5708):418-20).␣
↪Plasma collected at 12 of age in generation 28 rats after ad lib feeding or 40%␣
↪caloric restriction at week 8 of age. All animals fasted 4 hours prior to␣
↪collection between 5-8",
    "INSTITUTE": "University of Michigan",
    "DEPARTMENT": "Internal Medicine",
    "LABORATORY": "Burant Lab (MMOC)",
    "LAST_NAME": "Qi",
    "FIRST_NAME": "Nathan",
    "ADDRESS": "-",
    "EMAIL": "nathanqi@med.umich.edu",
    "PHONE": "734-232-0815",
    "NUM_GROUPS": "2",
    "TOTAL_SUBJECTS": "42"
}
```


Writing data from a MWTabFile object into a file

Data from a *MWTabFile* can be written into file in original mwTab format or in equivalent JSON format using *write()*:

- Writing into a mwTab formatted file:

[27]:
```
with open("out/ST000017_AN000035_modified.txt", "w") as outfile:
    mwfile.write(outfile, file_format="mwtab")
```

- Writing into a JSON file:

[28]:
```
with open("out/ST000017_AN000035_modified.json", "w") as outfile:
    mwfile.write(outfile, file_format="json")
```

Extracting Metadata and Metabolites from mwTab Files

The *mwtab.mwextract* module can be used to extract metadata from mwTab files. The module contains two main methods: 1) extract_metadata() which can be used to parse metadata values from a mwTab file, and 2) extract_metabolites() which can be used to gather a list of metabolites and samples containing the found metabolites from multiple mwTab files which contain a given metadata key value pair.

Extracting Metadata Values

- Extracting metadata values from a given mwTab file:

[29]:
```
from mwtab.mwextract import extract_metadata

extract_metadata(mwfile, ["STUDY_TYPE", "SUBJECT_TYPE"])
```

[29]:
```
{'STUDY_TYPE': {'LC-MS analysis'}, 'SUBJECT_TYPE': {'Animal'}}
```

Extracting Metabolites Values

- Extracting metabolite information from multiple `mwTab` files and outputing the first three metabolites:

[30]:
```
from mwtab.mwextract import extract_metabolites, generate_matchers
from mwtab import read_files

mwtab_gen = read_files(
    "ST000017_AN000035.txt",
    "ST000040_AN000060.txt"
)

matchers = generate_matchers([
    ("ST:STUDY_TYPE",
    "LC-MS analysis")
])
list(extract_metabolites(mwtab_gen, matchers).keys())[:3]
```

[30]:
```
['11BETA_21-DIHYDROXY-5BETA-PREGNANE-3_20-DIONE',
 '11-BETA-HYDROXYANDROST-4-ENE-3_17-DIONE',
 '13(S)-HPODE']
```

- Extracting metabolite information from multiple `mwTab` files using regualar expressions and outputing the first three metabolites:

[31]:
```
from mwtab.mwextract import extract_metabolites, generate_matchers
from mwtab import read_files
from re import compile

mwtab_gen = read_files(
    "ST000017_AN000035.txt",
    "ST000040_AN000060.txt"
)

matchers = generate_matchers([
    ("ST:STUDY_TYPE",
    compile("(LC-MS)"))
])
list(extract_metabolites(mwtab_gen, matchers).keys())[:3]
```

[31]:
```
['11BETA_21-DIHYDROXY-5BETA-PREGNANE-3_20-DIONE',
 '11-BETA-HYDROXYANDROST-4-ENE-3_17-DIONE',
 '13(S)-HPODE']
```

Converting mwTab Files

`mwTab` files can be converted between the `mwTab` file format and their `JSON` representation using the *`mwtab.converter`* module.

One-to-one file conversions

- Converting from the `mwTab` file format into its equivalent `JSON` file format:

[32]:
```
from mwtab.converter import Converter
```

(continues on next page)

(continued from previous page)

```
# Using valid ANALYSIS_ID to access file from URL: from_path="1"
converter = Converter(from_path="35", to_path="out/ST000017_AN000035.json",
                      from_format="mwtab", to_format="json")
converter.convert()
```

- Converting from JSON file format back to `mwTab` file format:

[33]:
```
from mwtab.converter import Converter

converter = Converter(from_path="out/ST000017_AN000035.json", to_path="out/ST000017_
↪AN000035.txt",
                      from_format="json", to_format="mwtab")
converter.convert()
```

Many-to-many files conversions

- Converting from the directory of `mwTab` formatted files into their equivalent `JSON` formatted files:

[34]:
```
from mwtab.converter import Converter

converter = Converter(from_path="mwfiles_dir_mwtab",
                      to_path="out/mwfiles_dir_json",
                      from_format="mwtab",
                      to_format="json")
converter.convert()
```

- Converting from the directory of `JSON` formatted files into their equivalent `mwTab` formatted files:

[35]:
```
from mwtab.converter import Converter

converter = Converter(from_path="out/mwfiles_dir_json",
                      to_path="out/mwfiles_dir_mwtab",
                      from_format="json",
                      to_format="mwtab")
converter.convert()
```

Note: Many-to-many files and one-to-one file conversions are available. See *`mwtab.converter`* for full list of available conversions.

2.2.3 Command-Line Interface

The mwtab Command-Line Interface provides the following functionality:

- Convert from the `mwTab` file format into its equivalent `JSON` file format and vice versa.
- Download files through Metabolomics Workbench's REST API.
- Validate the `mwTab` formatted file.
- Extract metadata and metabolite information from downloaded files.

[36]:
```
! mwtab --help
```

```
The mwtab command-line interface
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Usage:
    mwtab -h | --help
    mwtab --version
    mwtab convert (<from-path> <to-path>) [--from-format=<format>] [--to-format=
↪<format>] [--validate] [--mw-rest=<url>] [--verbose]
    mwtab validate <from-path> [--mw-rest=<url>] [--verbose]
    mwtab download url <url> [--to-path=<path>] [--verbose]
    mwtab download study all [--to-path=<path>] [--input-item=<item>] [--output-
↪format=<format>] [--mw-rest=<url>] [--validate] [--verbose]
    mwtab download study <input-value> [--to-path=<path>] [--input-item=<item>] [--
↪output-item=<item>] [--output-format=<format>] [--mw-rest=<url>] [--validate] [--
↪verbose]
    mwtab download (study | compound | refmet | gene | protein) <input-item> <input-
↪value> <output-item> [--output-format=<format>] [--to-path=<path>] [--mw-rest=<url>
↪] [--verbose]
    mwtab download moverz <input-item> <m/z-value> <ion-type-value> <m/z-tolerance-
↪value> [--to-path=<path>] [--mw-rest=<url>] [--verbose]
    mwtab download exactmass <LIPID-abbreviation> <ion-type-value> [--to-path=<path>]␣
↪[--mw-rest=<url>] [--verbose]
    mwtab extract metadata <from-path> <to-path> <key> ... [--to-format=<format>] [--
↪no-header]
    mwtab extract metabolites <from-path> <to-path> (<key> <value>) ... [--to-format=
↪<format>] [--no-header]

Options:
    -h, --help                      Show this screen.
    --version                       Show version.
    --verbose                       Print what files are processing.
    --validate                      Validate the mwTab file.
    --from-format=<format>          Input file format, available formats: mwtab, json␣
↪[default: mwtab].
    --to-format=<format>            Output file format [default: json].
                                    Available formats for convert:
                                        mwtab, json.
                                    Available formats for extract:
                                        json, csv.
    --mw-rest=<url>                 URL to MW REST interface
                                    [default: https://www.metabolomicsworkbench.org/
↪rest/].
    --context=<context>             Type of resource to access from MW REST interface,
↪ available contexts: study,
                                    compound, refmet, gene, protein, moverz,␣
↪exactmass [default: study].
    --input-item=<item>             Item to search Metabolomics Workbench with.
    --output-item=<item>            Item to be retrieved from Metabolomics Workbench.
    --output-format=<format>        Format for item to be retrieved in, available␣
↪formats: mwtab, json.
    --no-header                     Include header at the top of csv formatted files.

    For extraction <to-path> can take a "-" which will use stdout.
```

Converting `mwTab` files in bulk

CLI one-to-one file conversions

- Convert from a local file in `mwTab` format to a local file in `JSON` format:

```
[37]: ! mwtab convert ST000017_AN000035.txt out/ST000017_AN000035.json \
          --from-format=mwtab --to-format=json
```

- Convert from a local file in `JSON` format to a local file in `mwTab` format:

```
[38]: ! mwtab convert ST000017_AN000035.json out/ST000017_AN000035.txt \
          --from-format=json --to-format=mwtab
```

- Convert from a compressed local file in `mwTab` format to a compressed local file in `JSON` format:

```
[39]: ! mwtab convert ST000017_AN000035.txt.gz out/ST000017_AN000035.json.gz \
          --from-format=mwtab --to-format=json
```

- Convert from a compressed local file in `JSON` format to a compressed local file in `mwTab` format:

```
[40]: ! mwtab convert ST000017_AN000035.json.gz out/ST000017_AN000035.txt.gz \
          --from-format=json --to-format=mwtab
```

- Convert from an uncompressed URL file in `mwTab` format to a compressed local file in `JSON` format:

```
[41]: ! mwtab convert 35 out/ST000017_AN000035.json.bz2 \
          --from-format=mwtab --to-format=json
```

Note: See *mwtab.converter* for full list of available conversions.

CLI Many-to-many files conversions

- Convert from a directory of files in `mwTab` format to a directory of files in `JSON` format:

```
[42]: ! mwtab convert mwfiles_dir_mwtab out/mwfiles_dir_json \
          --from-format=mwtab --to-format=json
```

- Convert from a directory of files in `JSON` format to a directory of files in `mwTab` format:

```
[43]: ! mwtab convert mwfiles_dir_json out/mwfiles_dir_mwtab \
          --from-format=json --to-format=mwtab
```

- Convert from a directory of files in `mwTab` format to a zip archive of files in `JSON` format:

```
[44]: ! mwtab convert mwfiles_dir_mwtab out/mwfiles_json.zip \
          --from-format=mwtab --to-format=json
```

- Convert from a compressed tar archive of files in `JSON` format to a directory of files in `mwTab` format:

```
[45]: ! mwtab convert mwfiles_json.tar.gz out/mwfiles_dir_mwtab \
          --from-format=json --to-format=mwtab
```

- Convert from a zip archive of files in `mwTab` format to a compressed tar archive of files in `JSON` format:

```
[46]: ! mwtab convert mwfiles_mwtab.zip out/mwfiles_json.tar.bz2 \
          --from-format=mwtab --to-format=json
```

Note: See *mwtab.converter* for full list of available conversions.

Download files through Metabolomics Workbenchs REST API

The *mwtab* package provides the *mwtab.mwrest* module, which contains a number of functions and classes for working with Metabolomics Workbenchs REST API.

Note: For full official REST API specification see the following link (`MW REST API (v1.0, 5/7/2019)`): https://www.metabolomicsworkbench.org/tools/MWRestAPIv1.0.pdf

Download by URL

- To download a file based on a given url, simply call the `download url` command with the desired URL and provide an output path:

```
[47]: ! mwtab download url "https://www.metabolomicsworkbench.org/rest/study/analysis_id/
      ↪AN000035/mwtab/txt" --to-path=out/ST000017_AN000035.txt
```

- To download single analysis `mwTab` files, simply call `download study` and specifiy the analysis ID:

```
[48]: ! mwtab download study AN000035 --to-path=out/ST000017_AN000035.txt
```

- To download an entire study `mwTab` file, simply call `download study` and specifiy the study ID:

```
[49]: ! mwtab download study ST000017 --to-path=out/ST000017_AN000035.txt
```

Note: It is possible to validate downloaded files by adding the `--validate` option to the command line.

Download study, compound, refmet, gene, and protein files

- To download study, compound, refmet, gene, and protein context files, call the `download` command and specify the context, input iten, input value, and output item (optionally specifiy the output format).
- Download a study:

```
[50]: ! mwtab download study analysis_id AN000035 mwtab --output-format=txt --to-path=out/
      ↪ST000017_AN000035.txt
```

- Download compound:

```
[51]: ! mwtab download compound regno 11 name --to-path=out/tmp.txt
```

- Download refmet:

[52]:
```
! mwtab download refmet name Cholesterol all --to-path=out/tmp.txt
```

- Download gene:

[53]:
```
! mwtab download gene gene_symbol acaca all --to-path=out/tmp.txt
```

- Download protein:

[54]:
```
! mwtab download protein uniprot_id Q13085 all --to-path=out/tmp.txt
```

Download all `mwTab` formatted files

The `mwTab` package provides contains a number of command line functions for downloading Metabolomics `mwtab` formatted files through the Workbenchs REST API.

- To download all available analysis files, simply call the `download study all` command:

! mwtab download study all

- It is also possible to download all study files by calling the `download study all` command and providing an input item and output path:

! mwtab download study all –input-item=study_id

Download moverz and exactmass

- To download moverz files, call the `download moverz` command and specify the input value (LIPIDS, MB, or REFMET), m/z value, ion type value, and m/z tolerance value.

[55]:
```
! mwtab download moverz MB 635.52 M+H 0.5 --to-path=out/tmp.txt
```

- To download exactmass files, call the `download exactmass` command and specify the LIPID abbreviation and ion type value.

[56]:
```
! mwtab download exactmass "PC(34:1)" M+H --to-path=out/tmp.txt
```

Note: It is not necessary to specify an output format for exactmass files.

Extracting metabolite data and metadata from `mwTab` files

The *`mwtab`* package provides the *`extract_metabolites()`* and *`extract_metadata()`* functions that can parse `mwTab` formatted files. The *`extract_metabolites()`* takes a source (list of `mwTab` file) and list of metadata key-value pairs that are used to search for `mwTab` files which contain the given metadata pairs. The *`extract_metadata()`* takes a source (list of `mwTab` file) and list of metadata keys which are used to search the `mwTab` files for possible values to the given keys.

- To extract metabolite from `mwTab` files in a directory, call the `extract metabolites` command and provide a list of metadata key value pairs along with an output path and output format:

[57]:
```
! mwtab extract metabolites mwfiles_dir_mwtab out/output_file.csv SU:SUBJECT_TYPE␣
↪Plant --to-format=csv
```

Note: It is possible to use ReGeXs to match the metadata value (eg. ... SU:SUBJECT_TYPE "r'(Plant)'").

- To extract metadata from `mwTab` files in a directory call the `extract metadata` command and provide a list of metadata keys along with an output path and output format:

```
[58]: ! mwtab extract metadata mwfiles_dir_json out/output_file.json SUBJECT_TYPE --to-
      ↪format=json
```

Validating `mwTab` files

The *`mwtab`* package provides the *`validate_file()`* function that can validate files based on a `JSON` schema definition. The *`mwtab.mwschema`* contains schema definitions for every block of `mwTab` formatted file, i.e. it lists the types of attributes (e.g. `str` as well as specifies which keys are optional and which are required).

- To validate file(s), simply call the `validate` command and provide path to file(s):

```
[59]: ! mwtab validate 35
```

2.2.4 Using the mwtab Python Package to Find Analyses Involving a Specific Disease or Condition

The Metabolomics Workbench data repository stores mass spectroscopy and nuclear magnetic resonanse experimental data and metadata in `mwTab` formatted files. Metabolomics Workbench also provides a number of tools for searching or analyzing `mwTab` files. The mwtab Python package can also be used to perform similar functions through both a programmatic API and command-line interface, which has more search flexibility.

In order to search the repository of `mwTab` files for analyses associated with a specific disease, Metabolomics Workbench provid

- https://www.metabolomicsworkbench.org/data/metsearch_MS_form2.php

The mwtab Python package can be used in a number of ways to similar effect. The package provides the *`extract_metabolites()`* method to extract and organize metabolites from multiple `mwTab` files through both Python scripts and a command-line interface. This method has more search flexibility, since it can take either a search string or a regular expression.

Using mwtab package API to extract study IDs, analysis IDs, and metabolites

The *`extract_metabolites()`* method takes two parameters: 1) a iterable of *`MWTabFile`* instances and 2) an iterable of *`ItemMatcher`* or *`ReGeXMatcher`* instances. The iterable of *`MWTabFile`* instances can be created using byt passing `mwTab` file sources (filenames, analysis IDs, etc.) to the *`read_files()`* method. The iterable of matcher instances can be created using the *`generate_matchers()`* method.

- An example of using the mwtab package API to extract data from analyses associated with diabetes and output the first three metabolites:

```
[60]: from mwtab.mwextract import extract_metabolites, generate_matchers
      from mwtab import read_files
      import re

      mwtab_gen = read_files("diabetes/")
```

(continues on next page)

(continued from previous page)

```
matchers = generate_matchers([
    ("ST:STUDY_SUMMARY",
    re.compile("(diabetes)"))
])
list(extract_metabolites(mwtab_gen, matchers).keys())[:3]
```

[60]:
```
['1_5-anhydroglucitol', '1-monopalmitin', '1-monostearin']
```

Using mwtab CLI to extract study IDs, analysis IDs, and metabolites

The mwtab command line interface includes a `mwtab extract metabolites` method which takes a directory of `mwTab` files, an output path to save the extracted data in, and a series of `mwTab` section item keys and values to be matched (either string values or regular expressions). Additionally an output format can be specified.

> mwtab extract metabolites <from-path> <to-path> (<key> <value>) ... [–to-format=<format>] [–no-header]

- An example of using the mwtab CLI to extract data from analyses associated with diabetes:

[61]:
```
! mwtab extract metabolites diabetes/ out/output_file.json ST:STUDY_SUMMARY "r'(?
↪i)(diabetes)'" --to-format=json
```

2.3 The mwtab API Reference

Routines for working with `mwTab` format files used by the Metabolomics Workbench.

This package includes the following modules:

mwtab This module provides the *MWTabFile* class which is a python dictionary representation of a Metabolomics Workbench *mwtab* file. Data can be accessed directly from the *MWTabFile* instance using bracket accessors.

cli This module provides command-line interface for the `mwtab` package.

tokenizer This module provides the *tokenizer()* generator that generates tuples of key-value pairs from *mwtab* files.

fileio This module provides the *read_files()* generator to open files from different sources (single file/multiple files on a local machine, directory/archive of files, URL address of a file).

converter This module provides the *Converter* class that is responsible for the conversion of `mwTab` formated files into their JSON representation and vice versa.

mwschema This module provides JSON schema definitions for the `mwTab` formatted files, i.e. specifies required and optional keys as well as data types.

validator This module provides routines to validate `mwTab` formatted files based on schema definitions as well as checks for file self-consistency.

mwrest This module provides the *GenericMWURL* class which is a python dictionary representation of a Metabolomics Workbench REST URL. The class is used to validate query parameters and to generate a URL path which can be used to request data from Metabolomics Workbench through their REST API.

2.3.1 mwtab.mwtab

This module provides the *MWTabFile* class that stores the data from a single mwTab formatted file in the form of an OrderedDict. Data can be accessed directly from the *MWTabFile* instance using bracket accessors.

The data is divided into a series of "sections" which each contain a number of "key-value"-like pairs. Also, the file contains a specially formatted SUBJECT_SAMPLE_FACTOR block and blocks of data between *_START and *_END.

class mwtab.mwtab.**MWTabFile**(*source*, **args*, ***kwds*)

MWTabFile class that stores data from a single mwTab formatted file in the form of collections.OrderedDict.

read(*filehandle*)

Read data into a *MWTabFile* instance.

Parameters **filehandle** (io.TextIOWrapper, gzip.GzipFile, bz2.BZ2File, zipfile.ZipFile) – file-like object.

Returns None

Return type None

write(*filehandle*, *file_format*)

Write *MWTabFile* data into file.

Parameters

- **filehandle** (io.TextIOWrapper) – file-like object.
- **file_format** (*str*) – Format to use to write data: *mwtab* or *json*.

Returns None

Return type None

writestr(*file_format*)

Write *MWTabFile* data into string.

Parameters **file_format** (*str*) – Format to use to write data: *mwtab* or *json*.

Returns String representing the *MWTabFile* instance.

Return type str

print_file(*f=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*, *file_format='mwtab'*)

Print *MWTabFile* into a file or stdout.

Parameters

- **f** (io.StringIO) – writable file-like stream.
- **file_format** (*str*) – Format to use: *mwtab* or *json*.

Returns None

Return type None

print_subject_sample_factors(*section_key*, *f=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*, *file_format='mwtab'*)

Print *mwtab SUBJECT_SAMPLE_FACTORS* section into a file or stdout.

Parameters

- **section_key** (*str*) – Section name.

- **f** (io.StringIO) – writable file-like stream.
- **file_format** (*str*) – Format to use: *mwtab* or *json*.

Returns None

Return type None

print_block(*section_key*, *f=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*, *file_format='mwtab'*)

Print *mwtab* section into a file or stdout.

Parameters

- **section_key** (*str*) – Section name.
- **f** (io.StringIO) – writable file-like stream.
- **file_format** (*str*) – Format to use: *mwtab* or *json*.

Returns None

Return type None

2.3.2 The mwtab command-line interface

Usage: mwtab -h | –help mwtab –version mwtab convert (<from-path> <to-path>) [–from-format=<format>] [–to-format=<format>] [–validate] [–mw-rest=<url>] [–verbose] mwtab validate <from-path> [–mw-rest=<url>] [–verbose] mwtab download url <url> [–to-path=<path>] [–verbose] mwtab download study all [–to-path=<path>] [–input-item=<item>] [–output-format=<format>] [–mw-rest=<url>] [–validate] [–verbose] mwtab download study <input-value> [–to-path=<path>] [–input-item=<item>] [–output-item=<item>] [–output-format=<format>] [–mw-rest=<url>] [–validate] [–verbose] mwtab download (study | compound | refmet | gene | protein) <input-item> <input-value> <output-item> [–output-format=<format>] [–to-path=<path>] [–mw-rest=<url>] [–verbose] mwtab download moverz <input-item> <m/z-value> <ion-type-value> <m/z-tolerance-value> [–to-path=<path>] [–mw-rest=<url>] [–verbose] mwtab download exactmass <LIPID-abbreviation> <ion-type-value> [–to-path=<path>] [–mw-rest=<url>] [–verbose] mwtab extract metadata <from-path> <to-path> <key> ... [–to-format=<format>] [–no-header] mwtab extract metabolites <from-path> <to-path> (<key> <value>) ... [–to-format=<format>] [–no-header]

Options:

-h, --help Show this screen.

--version Show version.

--verbose Print what files are processing.

--validate Validate the mwTab file.

--from-format=<format> Input file format, available formats: mwtab, json [default: mwtab].

--to-format=<format> Output file format [default: json]. Available formats for convert:

mwtab, json.

Available formats for extract: json, csv.

--mw-rest=<url> URL to MW REST interface [default: https://www.metabolomicsworkbench.org/rest/].

--context=<context> Type of resource to access from MW REST interface, available contexts: study, compound, refmet, gene, protein, moverz, exactmass [default: study].

--input-item=<item> Item to search Metabolomics Workbench with.

--output-item=<item> Item to be retrieved from Metabolomics Workbench.

--output-format=<format> Format for item to be retrieved in, available formats: mwtab, json.

--no-header Include header at the top of csv formatted files.

For extraction <to-path> can take a "-" which will use stdout.

mwtab.cli.**cli**(*cmdargs*)

Implements the command line interface.

param dict cmdargs: dictionary of command line arguments.

2.3.3 mwtab.tokenizer

This module provides the *tokenizer()* lexical analyzer for *mwTab* format syntax. It is implemented as Python generator-based state machine which generates (yields) tokens one at a time when `next()` is invoked on *tokenizer()* instance.

Each token is a tuple of "key-value"-like pairs, tuple of `SUBJECT_SAMPLE_FACTORS` or tuple of data deposited between `*_START` and `*_END` blocks.

mwtab.tokenizer.**tokenizer**(*text*)

A lexical analyzer for the *mwtab* formatted files.

Parameters **text** (py:class:*str*) – *mwTab* formatted text.

Returns Tuples of data.

Return type py:class:~*collections.namedtuple*

2.3.4 mwtab.fileio

This module provides routines for reading `mwTab` formatted files from difference kinds of sources:

- Single `mwTab` formatted file on a local machine.
- Directory containing multiple `mwTab` formatted files.
- Compressed zip/tar archive of `mwTab` formatted files.
- URL address of `mwTab` formatted file.
- `ANALYSIS_ID` of `mwTab` formatted file.

mwtab.fileio.**read_files**(**sources*, ***kwds*)

Construct a generator that yields file instances.

Parameters **sources** – One or more strings representing path to file(s).

2.3.5 mwtab.converter

This module provides functionality for converting between the Metabolomics Workbench `mwTab` formatted file and its equivalent JSONized representation.

The following conversions are possible:

Local files:

- **One-to-one file conversions:**

  - textfile - to - textfile
  - textfile - to - textfile.gz
  - textfile - to - textfile.bz2
  - textfile.gz - to - textfile
  - textfile.gz - to - textfile.gz
  - textfile.gz - to - textfile.bz2
  - textfile.bz2 - to - textfile
  - textfile.bz2 - to - textfile.gz
  - textfile.bz2 - to - textfile.bz2
  - textfile / textfile.gz / textfile.bz2 - to - textfile.zip / textfile.tar / textfile.tar.gz / textfile.tar.bz2 (TypeError: One-to-many conversion)

- **Many-to-many files conversions:**
  - **Directories:**
    - directory - to - directory
    - directory - to - directory.zip
    - directory - to - directory.tar
    - directory - to - directory.tar.bz2
    - directory - to - directory.tar.gz
    - directory - to - directory.gz / directory.bz2 (TypeError: Many-to-one conversion)
  - **Zipfiles:**
    - zipfile.zip - to - directory
    - zipfile.zip - to - zipfile.zip
    - zipfile.zip - to - tarfile.tar
    - zipfile.zip - to - tarfile.tar.gz
    - zipfile.zip - to - tarfile.tar.bz2
    - zipfile.zip - to - directory.gz / directory.bz2 (TypeError: Many-to-one conversion)
  - **Tarfiles:**
    - tarfile.tar - to - directory
    - tarfile.tar - to - zipfile.zip
    - tarfile.tar - to - tarfile.tar
    - tarfile.tar - to - tarfile.tar.gz
    - tarfile.tar - to - tarfile.tar.bz2
    - tarfile.tar - to - directory.gz / directory.bz2 (TypeError: Many-to-one conversion)
    - tarfile.tar.gz - to - directory
    - tarfile.tar.gz - to - zipfile.zip
    - tarfile.tar.gz - to - tarfile.tar

        * tarfile.tar.gz - to - tarfile.tar.gz
        * tarfile.tar.gz - to - tarfile.tar.bz2
        * tarfile.tar.gz - to - directory.gz / directory.bz2 (TypeError: Many-to-one conversion)
        * tarfile.tar.bz2 - to - directory
        * tarfile.tar.bz2 - to - zipfile.zip
        * tarfile.tar.bz2 - to - tarfile.tar
        * tarfile.tar.bz2 - to - tarfile.tar.gz
        * tarfile.tar.bz2 - to - tarfile.tar.bz2
        * tarfile.tar.bz2 - to - directory.gz / directory.bz2 (TypeError: Many-to-one conversion)

URL files:

- **One-to-one file conversions:**
  - analysis_id - to - textfile
  - analysis_id - to - textfile.gz
  - analysis_id - to - textfile.bz2
  - analysis_id - to - textfile.zip / textfile.tar / textfile.tar.gz / textfile.tar.bz2 (TypeError: One-to-many conversion)
  - textfileurl - to - textfile
  - textfileurl - to - textfile.gz
  - textfileurl - to - textfile.bz2
  - textfileurl.gz - to - textfile
  - textfileurl.gz - to - textfile.gz
  - textfileurl.gz - to - textfile.bz2
  - textfileurl.bz2 - to - textfile
  - textfileurl.bz2 - to - textfile.gz
  - textfileurl.bz2 - to - textfile.bz2
  - textfileurl / textfileurl.gz / textfileurl.bz2 - to - textfile.zip / textfile.tar / textfile.tar.gz / textfile.tar.bz2 (TypeError: One-to-many conversion)
- **Many-to-many files conversions:**
  - **Zipfiles:**
    * zipfileurl.zip - to - directory
    * zipfileurl.zip - to - zipfile.zip
    * zipfileurl.zip - to - tarfile.tar
    * zipfileurl.zip - to - tarfile.tar.gz
    * zipfileurl.zip - to - tarfile.tar.bz2
    * zipfileurl.zip - to - directory.gz / directory.bz2 (TypeError: Many-to-one conversion)
  - **Tarfiles:**
    * tarfileurl.tar - to - directory

* tarfileurl.tar - to - zipfile.zip
* tarfileurl.tar - to - tarfile.tar
* tarfileurl.tar - to - tarfile.tar.gz
* tarfileurl.tar - to - tarfile.tar.bz2
* tarfileurl.tar - to - directory.gz / directory.bz2 (TypeError: Many-to-one conversion)
* tarfileurl.tar.gz - to - directory
* tarfileurl.tar.gz - to - zipfile.zip
* tarfileurl.tar.gz - to - tarfile.tar
* tarfileurl.tar.gz - to - tarfile.tar.gz
* tarfileurl.tar.gz - to - tarfile.tar.bz2
* tarfileurl.tar.gz - to - directory.gz / directory.bz2 (TypeError: Many-to-one conversion)
* tarfileurl.tar.bz2 - to - directory
* tarfileurl.tar.bz2 - to - zipfile.zip
* tarfileurl.tar.bz2 - to - tarfile.tar
* tarfileurl.tar.bz2 - to - tarfile.tar.gz
* tarfileurl.tar.bz2 - to - tarfile.tar.bz2
* tarfileurl.tar.bz2 - to - directory.gz / directory.bz2 (TypeError: Many-to-one conversion)

class mwtab.converter.**Translator**(*from_path*, *to_path*, *from_format=None*, *to_format=None*, *validate=False*)

Translator abstract class.

class mwtab.converter.**MWTabFileToMWTabFile**(*from_path*, *to_path*, *from_format=None*, *to_format=None*, *validate=False*)

Translator concrete class that can convert between mwTab and JSON formats.

class mwtab.converter.**Converter**(*from_path*, *to_path*, *from_format='mwtab'*, *to_format='json'*, *validate=False*)

Converter class to convert mwTab files from mwTab to JSON or from JSON to mwTab format.

convert()

Convert file(s) from mwTab format to JSON format or from JSON format to mwTab format. :return: None :rtype: None

2.3.6 mwtab.validator

This module contains routines to validate consistency of the mwTab formatted files, e.g. make sure that Samples and Factors identifiers are consistent across the file, make sure that all required key-value pairs are present.

mwtab.validator.**validate_file**(*mwtabfile*, *section_schema_mapping={'ANALYSIS': Schema({'ANALYSIS_TYPE': <class 'str'>, Optional('LABORATORY_NAME'): <class 'str'>, Optional('OPERATOR_NAME'): <class 'str'>, Optional('DETECTOR_TYPE'): <class 'str'>, Optional('SOFTWARE_VERSION'): <class 'str'>, Optional('ACQUISITION_DATE'): <class 'str'>, Optional('ANALYSIS_PROTOCOL_FILE'): <class 'str'>, Optional('ACQUISITION_PARAMETERS_FILE'): <class 'str'>, Optional('PROCESSING_PARAMETERS_FILE'): <class 'str'>, Optional('DATA_FORMAT'): <class 'str'>, Optional('ACQUISITION_ID'): <class 'str'>, Optional('ACQUISITION_TIME'): <class 'str'>, Optional('ANALYSIS_COMMENTS'): <class 'str'>, Optional('ANALYSIS_DISPLAY'): <class 'str'>, Optional('INSTRUMENT_NAME'): <class 'str'>, Optional('INSTRUMENT_PARAMETERS_FILE'): <class 'str'>, Optional('NUM_FACTORS'): <class 'str'>, Optional('NUM_METABOLITES'): <class 'str'>, Optional('PROCESSED_FILE'): <class 'str'>, Optional('RANDOMIZATION_ORDER'): <class 'str'>, Optional('RAW_FILE'): <class 'str'>}), 'CHROMATOGRAPHY': Schema({Optional('CHROMATOGRAPHY_SUMMARY'): <class 'str'>, 'CHROMATOGRAPHY_TYPE': <class 'str'>, 'INSTRUMENT_NAME': <class 'str'>, 'COLUMN_NAME': <class 'str'>, Optional('FLOW_GRADIENT'): <class 'str'>, Optional('FLOW_RATE'): <class 'str'>, Optional('COLUMN_TEMPERATURE'): <class 'str'>, Optional('METHODS_FILENAME'): <class 'str'>, Optional('SOLVENT_A'): <class 'str'>, Optional('SOLVENT_B'): <class 'str'>, Optional('METHODS_ID'): <class 'str'>, Optional('COLUMN_PRESSURE'): <class 'str'>, Optional('INJECTION_TEMPERATURE'): <class 'str'>, Optional('INTERNAL_STANDARD'): <class 'str'>, Optional('INTERNAL_STANDARD_MT'): <class 'str'>, Optional('RETENTION_INDEX'): <class 'str'>, Optional('RETENTION_TIME'): <class 'str'>, Optional('SAMPLE_INJECTION'): <class 'str'>, Optional('SAMPLING_CONE'): <class 'str'>, Optional('ANALYTICAL_TIME'): <class 'str'>, Optional('CAPILLARY_VOLTAGE'): <class 'str'>, Optional('MIGRATION_TIME'): <class 'str'>, Optional('OVEN_TEMPERATURE'): <class 'str'>, Optional('PRECONDITIONING'): <class 'str'>, Optional('RUNNING_BUFFER'): <class 'str'>, Optional('RUNNING_VOLTAGE'): <class 'str'>, Optional('SHEATH_LIQUID'): <class 'str'>, Optional('TIME_PROGRAM'): <class 'str'>, Optional('TRANSFERLINE_TEMPERATURE'): <class 'str'>, Optional('WASHING_BUFFER'): <class 'str'>, Optional('WEAK_WASH_SOLVENT_NAME'): <class 'str'>, Optional('WEAK_WASH_VOLUME'): <class 'str'>, Optional('STRONG_WASH_SOLVENT_NAME'): <class 'str'>, Optional('STRONG_WASH_VOLUME'): <class 'str'>, Optional('TARGET_SAMPLE_TEMPERATURE'): <class 'str'>, Optional('SAMPLE_LOOP_SIZE'): <class 'str'>, Optional('SAMPLE_SYRINGE_SIZE'): <class 'str'>, Optional('RANDOMIZATION_ORDER'): <class 'str'>, Optional('CHROMATOGRAPHY_COMMENTS'): <class 'str'>}), 'COLLECTION':*

Validate mwTab formatted file.

Parameters

- **mwtabfile** (*MWTabFile* or collections.OrderedDict) – Instance of *MWTabFile*.
- **section_schema_mapping** (*dict*) – Dictionary that provides mapping between section name and schema definition.
- **verbose** (*bool*) – whether to be verbose or not.
- **metabolites** (*bool*) – whether to validate metabolites section.

Returns Validated file.

Return type collections.OrderedDict

2.3.7 mwtab.mwrest

This module provides routines for accessing the Metabolomics Workbench REST API.

See https://www.metabolomicsworkbench.org/tools/MWRestAPIv1.0.pdf for details.

mwtab.mwrest.**analysis_ids**(*base_url='https://www.metabolomicsworkbench.org/rest/'*)

Method for retrieving a list of analysis ids for every current analysis in Metabolomics Workbench.

Parameters **base_url** (*str*) – Base url to Metabolomics Workbench REST API.

Returns List of every available Metabolomics Workbench analysis identifier.

Return type list

mwtab.mwrest.**study_ids**(*base_url='https://www.metabolomicsworkbench.org/rest/'*)

Method for retrieving a list of study ids for every current study in Metabolomics Workbench.

Parameters **base_url** (*str*) – Base url to Metabolomics Workbench REST API.

Returns List of every available Metabolomics Workbench study identifier.

Return type list

mwtab.mwrest.**generate_mwtab_urls**(*input_items*, *base_url='https://www.metabolomicsworkbench.org/rest/'*, *output_format='txt'*)

Method for generating URLS to be used to retrieve *mwtab* files for analyses and studies through the REST API of the Metabolomics Workbench database.

Parameters

- **input_items** (*list*) – List of Metabolomics Workbench input values for mwTab files.
- **base_url** (*str*) – Base url to Metabolomics Workbench REST API.
- **output_format** (*str*) – Output format for the mwTab files to be retrieved in.

Returns Metabolomics Workbench REST URL string(s).

Return type str

mwtab.mwrest.**generate_urls**(*input_items*, *base_url='https://www.metabolomicsworkbench.org/rest/'*, ***kwds*)

Method for creating a generator which yields validated Metabolomics Workbench REST urls.

Parameters

- **input_items** (*list*) – List of Metabolomics Workbench input values for mwTab files.

- **base_url** (*str*) – Base url to Metabolomics Workbench REST API.
- **kwds** (*dict*) – Keyword arguments of Metabolomics Workbench URL Path items.

Returns Metabolomics Workbench REST URL string(s).

Return type str

class mwtab.mwrest.**GenericMWURL**(*rest_params*, *base_url='https://www.metabolomicsworkbench.org/rest/'*)

GenericMWURL class that stores and validates parameters specifying a Metabolomics Workbench REST URL.

Metabolomics REST API requests are performed using URL requests in the form of https://www.metabolomicsworkbench.org/rest/context/input_specification/output_specification

where:

if context = "study" | "compound" | "refmet" | "gene" | "protein" input_specification = input_item/input_value output_specification = output_item/[output_format]

elif context = "moverz"

input_specification = input_item/input_value1/input_value2/input_value3 input_item = "LIPIDS" | "MB" | "REFMET" input_value1 = m/z_value input_value2 = ion_type_value input_value3 = m/z_tolerance_value

output_specification = output_format output_format = "txt"

elif context = "exactmass"

input_specification = input_item/input_value1/input_value2 input_item = "LIPIDS" | "MB" | "REFMET" input_value1 = LIPID_abbreviation input_value2 = ion_type_value

output_specification = None

class mwtab.mwrest.**MWRESTFile**(*source*)

MWRESTFile class that stores data from a single file download through Metabolomics Workbench's REST API.

Mirrors *MWTabFile*.

read(*filehandle*)

Read data into a *MWRESTFile* instance.

Parameters **filehandle** (io.TextIOWrapper, gzip.GzipFile, bz2.BZ2File, zipfile.ZipFile) – file-like object.

Returns None

Return type None

write(*filehandle*)

Write *MWRESTFile* data into file.

Parameters **filehandle** (io.TextIOWrapper) – file-like object.

Returns None

Return type None

2.3.8 mwtab.mwextract

This module provides a number of functions and classes for extracting and saving data and metadata stored in mwTab formatted files in the form of *MWTabFile*.

class mwtab.mwextract.**ItemMatcher**(*full_key*, *value_comparison*)
ItemMatcher class that can be called to match items from mwTab formatted files in the form of *MWTabFile*.

class mwtab.mwextract.**ReGeXMatcher**(*full_key*, *value_comparison*)
ReGeXMatcher class that can be called to match items from mwTab formatted files in the form of *MWTabFile* using regular expressions.

mwtab.mwextract.**generate_matchers**(*items*)
Construct a generator that yields Matchers ItemMatcher or ReGeXMatcher.

Parameters **items** (*iterable*) – Iterable object containing key value pairs to match.

Returns Yields a Matcher object for each given item.

Return type ItemMatcher or ReGeXMatcher

mwtab.mwextract.**extract_metabolites**(*sources*, *matchers*)
Extract metabolite data from mwTab formatted files in the form of *MWTabFile*.

Parameters

- **sources** (*generator*) – Generator of mwtab file objects (*MWTabFile*).
- **matchers** (*generator*) – Generator of matcher objects (*ItemMatcher* or

ReGeXMatcher). :return: Extracted metabolites dictionary. :rtype: dict

mwtab.mwextract.**extract_metadata**(*mwtabfile*, *keys*)
Extract metadata data from mwTab formatted files in the form of *MWTabFile*.

Parameters

- **mwtabfile** (*MWTabFile*) – mwTab file object for metadata to be extracted from.
- **keys** (*list*) – List of metadata field keys for metadata values to be extracted.

Returns Extracted metadata dictionary.

Return type dict

mwtab.mwextract.**write_metadata_csv**(*to_path*, *extracted_values*, *no_header=False*)
Write extracted metadata dict into csv file.

Example: “metadata”,”value1”,”value2” “SUBJECT_TYPE”,”Human”,”Plant”

Parameters

- **to_path** (*str*) – Path to output file.
- **extracted_values** (*dict*) – Metadata dictionary to be saved.
- **no_header** (*bool*) – If true header is not included, otherwise header is included.

Returns None

Return type None

mwtab.mwextract.**write_metabolites_csv**(*to_path*, *extracted_values*, *no_header=False*)
Write extracted metabolites data dict into csv file.

Example: “metabolite_name”,”num-studies”,”num_analyses”,”num_samples” “1,2,4-benzenetriol”,”1”,”1”,”24” “1-monostearin”,”1”,”1”,”24” ...

Parameters

- **to_path** (*str*) – Path to output file.
- **extracted_values** (*dict*) – Metabolites data dictionary to be saved.

- **no_header** (*bool*) – If true header is not included, otherwise header is included.

Returns None

Return type None

class mwtab.mwextract.**SetEncoder**(*, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*)

SetEncoder class for encoding Python sets `set` into json serializable objects `list`.

default(*obj*)

Method for encoding Python objects. If object passed is a set, converts the set to JSON serializable lists or calls base implementation.

Parameters **obj** (*object*) – Python object to be json encoded.

Returns JSON serializable object.

Return type `dict`, `list`, `tuple`, `str`, `int`, `float`, `bool`, or `None`

mwtab.mwextract.**write_json**(*to_path*, *extracted_dict*)

Write extracted data or metadata `dict` into json file.

Metabolites example: {

"1,2,4-benzenetriol": {

"ST000001": {

"AN000001": ["LabF_115816", ...

]

}

}

}

Metadata example: {

"SUBJECT_TYPE": ["Plant", "Human"

]

}

Parameters

- **to_path** (*str*) – Path to output file.
- **extracted_dict** (*dict*) – Metabolites data or metadata dictionary to be saved.

Returns None

Return type None

2.3.9 mwtab.mwschema

This module provides schema definitions for different sections of the `mwTab` Metabolomics Workbench format.

mwtab.mwschema.**metabolomics_workbench_schema**

Entry point of the library, use this class to instantiate validation schema for the data that will be validated.

mwtab.mwschema.**project_schema**

Entry point of the library, use this class to instantiate validation schema for the data that will be validated.

mwtab.mwschema.**study_schema**
Entry point of the library, use this class to instantiate validation schema for the data that will be validated.

mwtab.mwschema.**analysis_schema**
Entry point of the library, use this class to instantiate validation schema for the data that will be validated.

mwtab.mwschema.**subject_schema**
Entry point of the library, use this class to instantiate validation schema for the data that will be validated.

mwtab.mwschema.**subject_sample_factors_schema**
Entry point of the library, use this class to instantiate validation schema for the data that will be validated.

mwtab.mwschema.**collection_schema**
Entry point of the library, use this class to instantiate validation schema for the data that will be validated.

mwtab.mwschema.**treatment_schema**
Entry point of the library, use this class to instantiate validation schema for the data that will be validated.

mwtab.mwschema.**sampleprep_schema**
Entry point of the library, use this class to instantiate validation schema for the data that will be validated.

mwtab.mwschema.**chromatography_schema**
Entry point of the library, use this class to instantiate validation schema for the data that will be validated.

mwtab.mwschema.**ms_schema**
Entry point of the library, use this class to instantiate validation schema for the data that will be validated.

mwtab.mwschema.**nmr_schema**
Entry point of the library, use this class to instantiate validation schema for the data that will be validated.

mwtab.mwschema.**ms_metabolite_data_schema**
Entry point of the library, use this class to instantiate validation schema for the data that will be validated.

mwtab.mwschema.**nmr_binned_data_schema**
Entry point of the library, use this class to instantiate validation schema for the data that will be validated.

2.4 License

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index

T

V

W