msAI

Release 1.3.1.dev0

Calvin Peters

Mar 12, 2020

CONTENTS:

CHAPTER

ONE

INTRODUCTION

msAI - Software Tools to Create AI Models for Mass Spectrometry Data

Note: msAI is in development. Currently, all releases are development previews. Features are incomplete and may change.

1.1 Overview

msAI provides a library of python modules for...

- Extracting data from mass spectrometry data files
  - Supported MS file types: mzML, TBD...
- Importing sample metadata
  - Unify MS data and sample metadata in a single view
- Partitioning data for training, testing, and validation
- Quality analysis of MS runs / samples
  - Metadata verification
  - Anomaly detection
- Plotting / visualization
- AI / machine learning frameworks
  - TensorFlow
  - TBD...
- Creating AI models
  - Neural network architectures - neuron layers, connections, activation functions
  - Model training - loss and optimization functions, hyperparameter tuning
  - Evaluation of models - measure accuracy, prevent overfitting
- Model lifecycle management
  - Deploy, update, backup, restore

1.2 Installation

Install from the Python Package Index (PyPI):

```
pip install msAI
```

1.3 Documentation

https://msai.readthedocs.io

1.4 Source Code

https://github.com/IsotopeC/msAI

CHAPTER

TWO

EXAMPLES

2.1 Basic Usage

This example demonstrates how to import and access data from MS and metadata files. Note that while individual *MSfile* instances can manually be created for every sample run, typically it is best to define a *MSfileSet* and *SampleMetadata* as shown below and use these instances to create a *SampleSet* as demonstrated in the next *example*.

2.1.1 Working with mass spectrometry data files

Accessing data from a MS sample run

Define the path to a MS data file, in this case a mzML file type.

```
>>> sample1_mzml_path = "./examples/data/mzML/EP2421.mzML"
```

MS data is accessed using a *MSfile* implementation matching the file type. *MZMLfile* is used with mzML data. On creation, the mzML file is imported into memory.

```
>>> sample1_ms = msData.MZMLfile(sample1_mzml_path)
```

The *MSfile* interface provides several properties for accessing MS metadata.

```
>>> sample1_ms.run_id
'EP2421'
```

```
>>> sample1_ms.run_date
'2017-06-28T04:10:21Z'
```

```
>>> sample1_ms.ms_file_version
'1.1.0'
```

```
>>> sample1_ms.spectrum_count
651
```

```
>>> sample1_ms.peak_count
1430013
```

```
>>> sample1_ms.tic_sum
103151911964.0
```

MS data is structured in dataframes and accessed by the *spectra* and *peaks* properties.

Spectra dataframe structure

Index: spec_id
Columns: rt, peak_count, tic, ms_lvl, filters

```
>>> sample1_ms.spectra
             rt  peak_count          tic  ms_lvl                                  ␣
↪filters
299    3.018841        1745   46977344.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
301    3.039366        1836   48066048.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
303    3.060012        2060   47754260.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
305    3.080646        1828   46855808.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
307    3.101156        1847   48759696.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
         ...         ...          ...     ...                                      .
↪..
1591  15.918533        3416  118047380.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
1593  15.938479        3328  128021860.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
1595  15.958450        3348  128402500.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
1597  15.978360        3156  152132620.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
1599  15.998312        3285  174533700.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
[651 rows x 5 columns]
```

Peaks dataframe structure

First Index Level: spec_id
Second Index Level: peak_number
Columns: rt, mz, i

```
>>> sample1_ms.peaks
                            rt          mz             i
spec_id peak_number
299     0              3.018841  115.03919  36447.125000
        1              3.018841  115.05045   2975.487549
        2              3.018841  115.07568   2015.634644
        3              3.018841  115.51699   1233.632690
        4              3.018841  115.96244   4875.453613
                          ...         ...          ...
1599    3280          15.998312  987.60944  12299.823242
        3281          15.998312  989.54504  39011.988281
        3282          15.998312  991.56219  57488.519531
        3283          15.998312  992.56891  21931.212891
        3284          15.998312  993.56921   7275.180176
[1430013 rows x 3 columns]
```

Get an individual spectrum with spec_id value.

```
>>> sample1_ms.spectra.loc[303]
rt                                                3.06001
peak_count                                           2060
tic                                           4.77543e+07
ms_lvl                                                  1
filters       FTMS + p ESI Full ms [115.0000-1000.0000]
Name: 303, dtype: object
```

Get a summary / distribution of peak values.

```
>>> sample1_ms.peaks.describe().round(2)
                rt           mz             i
count  1430013.00  1430013.00  1.430013e+06
mean          9.83      283.99  7.123009e+04
std           3.71      160.93  1.726981e+06
min           3.02      115.00  8.504700e+02
25%           6.79      167.07  5.720040e+03
50%           9.93      229.14  1.181848e+04
75%          12.90      349.25  3.166049e+04
max          16.00      999.95  9.182814e+08
```

Get all peaks in a spectrum with spec_id value.

```
>>> sample1_ms.peaks.loc[303]
                    rt         mz              i
peak_number
0             3.060012  115.03925  41569.882812
1             3.060012  115.05054   2562.014648
2             3.060012  115.07562   1966.861328
3             3.060012  115.08680   2180.555420
4             3.060012  115.52079   1273.498047
                 ...        ...           ...
2055          3.060012  717.65051   2805.519287
2056          3.060012  787.67346   2972.889648
2057          3.060012  896.67566   2859.390381
2058          3.060012  909.33502   3785.186035
2059          3.060012  926.53265   2564.230713
[2060 rows x 3 columns]
```

Get a single peak with spec_id and peak_number.

```
>>> sample1_ms.peaks.loc[303, 100]
rt        3.060012
mz      125.060060
i     10957.689453
Name: (303, 100), dtype: float64
```

Creating a set of MS files from a data directory

Define the data directory path. By default, contents of sub directories will be recursively included.

```
>>> mzml_dir = "./examples/data/mzML"
```

Create a set of the MS files in the data directory. This set is structured as a dataframe. Creating a *MSfileSet* does not import the MS data into memory. Rather, it provides a quick view of the MS data files available for use. The next *Sample Sets* example demonstrates how this MS file set is used to create a *SampleSet* and access the underlying MS data.

```
>>> ms_files = msData.MSfileSet(mzml_dir)
>>> ms_files
         file_type  file_size                              path
filename
EP0482        mzML  12.862821  examples/data/mzML/EP0482.mzML
EP2421        mzML  15.133800  examples/data/mzML/EP2421.mzML
EP2536        mzML  12.745723  examples/data/mzML/EP2536.mzML
```

2.1.2 Sample metadata

Additional sample metadata can be imported and associated with MS data.

Define the path to he metadata file.

```
>>> csv_path = "./examples/data/metadata/coneflower_metadata.csv"
```

Import metadata by creating a *SampleMetadata* instance. At creation, metadata contents are initially imported into a dataframe with a numerical index. Metadata labels and values are analyzed and a new index is automatically assigned, if possible. This index will be used by *SampleSet* to match this metadata with corresponding MS data in *MSfileSet*.

Requirements to auto index metadata:

- Has 1 or more entries/rows
- Has 2 or more labels/columns
- For one and only one label/column:
  - All label/column values are unique
  - All entries/rows have a value for this label/column

```
>>> cone_flower_metadata = SampleMetadata(csv_path)
```

Access the metadata dataframe with the `df` attribute.

```
>>> cone_flower_metadata.df
                class sampleType    site block treatment plantID tissue  siteblock␣
↪sitetreatment polarity
sampleMetadata
EP0045         sample     sample  Becker    B1      HIGH    P031   leaf  Becker_B1  ␣
↪Becker_HIGH  unknown
EP0046         sample     sample  Becker    B1      HIGH    P032   leaf  Becker_B1  ␣
↪Becker_HIGH  unknown
EP0047         sample     sample  Becker    B1      HIGH    P033   leaf  Becker_B1  ␣
↪Becker_HIGH  unknown
```

(continues on next page)

(continued from previous page)

```
EP0048          sample     sample  Becker    B1      HIGH    P034   leaf  Becker_B1 ␣
↪Becker_HIGH  unknown
EP0049          sample     sample  Becker    B1      HIGH    P035   leaf  Becker_B1 ␣
↪Becker_HIGH  unknown
                ...        ...     ...   ...       ...     ...    ...       ...    ␣
↪     ...      ...
EP2848          sample     sample  Becker    B3        R1    P074   root  Becker_B3 ␣
↪  Becker_R1  unknown
EP2849          sample     sample  Becker    B3        R1    P075   root  Becker_B3 ␣
↪  Becker_R1  unknown
EP2850          sample     sample  Becker    B3        R1    P076   root  Becker_B3 ␣
↪  Becker_R1  unknown
EP2851          sample     sample  Becker    B3        R1    P077   root  Becker_B3 ␣
↪  Becker_R1  unknown
EP2852          sample     sample  Becker    B3        R1    P078   root  Becker_B3 ␣
↪  Becker_R1  unknown
[984 rows x 10 columns]
```

Get a summary of metadata contents.

```
>>> cone_flower_metadata.describe()
         class sampleType    site block treatment plantID  tissue  siteblock␣
↪sitetreatment polarity
count      984        984     984   984       984     984     984        984          ␣
↪ 984      984
unique       1          1       2     3         6     365       5          6          ␣
↪  12        1
top     sample     sample  Becker    B2       LOW    P102  flower  Becker_B1      ␣
↪Becker_R6  unknown
freq       984        984     510   330       167       4     216        172          ␣
↪  87      984
```

2.2 Sample Sets

This example demonstrates how to define an entire sample set which will automatically create a *MSfile* interface for each data file. This demo uses the *MSfileSet* and *SampleMetadata* from the previous *example* to create a *SampleSet*. The last section shows how to save all the *SampleRun* instances and *SampleMetadata* from the *SampleSet*, as new `msAIr` files (one for each *SampleRun*) and a single `msAIm` file for the *SampleMetadata*. This example concludes by creating the *SampleSet* again by loading from the msAI data files. The advantages of this new format is explained in that *section*.

2.2.1 Creating a sample set

Create the *MSfileSet* instance.

```
>>> mzml_dir = "./examples/data/mzML"
>>> ms_files = msData.MSfileSet(mzml_dir)
```

Create the *SampleMetadata* instance.

```
>>> csv_path = "./examples/data/metadata/coneflower_metadata.csv"
>>> cone_flower_metadata = SampleMetadata(csv_path)
```

Create the *SampleSet*. A set can be constructed from any *MSfileSet* along with 0 or more *SampleMetadata*. Upon creation, *SampleRun* instances are created for each MS file, but the MS data will not initialized until called. This allows a cheep view of the entire set to exist without importing all the data into memory.

```
>>> sample_set = SampleSet(ms_files, cone_flower_metadata)
>>> sample_set
         file_type  file_size                              path   class sampleType
↪ site block treatment plantID  tissue     siteblock   sitetreatment polarity
↪                                        run
filename
EP0482        mzML  12.862821  examples/data/mzML/EP0482.mzML  sample     sample
↪Rosemount    B1      HIGH    P360    seed  Rosemount_B1  Rosemount_HIGH  unknown
↪<msAI.samples.SampleRun object at 0x7f063ff54f50>
EP2421        mzML  15.133800  examples/data/mzML/EP2421.mzML  sample     sample
↪Rosemount    B1        R1    P109  flower  Rosemount_B1    Rosemount_R1  unknown
↪<msAI.samples.SampleRun object at 0x7f063fed80d0>
EP2536        mzML  12.745723  examples/data/mzML/EP2536.mzML  sample     sample
↪Rosemount    B1       LOW    P134    root  Rosemount_B1   Rosemount_LOW  unknown
↪<msAI.samples.SampleRun object at 0x7f063ff35550>
```

2.2.2 Accessing sample MS data and metadata

Get a single sample with filename.

```
>>> sample_set.df.loc["EP2421"]
file_type                                                    mzML
file_size                                                 15.1338
path                                 examples/data/mzML/EP2421.mzML
class                                                      sample
sampleType                                                 sample
site                                                     Rosemount
block                                                           B1
treatment                                                       R1
plantID                                                       P109
tissue                                                      flower
siteblock                                               Rosemount_B1
sitetreatment                                           Rosemount_R1
polarity                                                     unknown
run              <msAI.samples.SampleRun object at 0x7f063fed80d0>
Name: EP2421, dtype: object
```

Get metadata values with label names.

```
>>> sample_set.df.loc["EP2421"].plantID
'P109'
```

```
>>> sample_set.df.loc["EP2421"].tissue
'flower'
```

```
>>> sample_set.df.loc["EP2421"].site
'Rosemount'
```

```
>>> sample_set.df.loc["EP2421"].treatment
'R1'
```

Note that a *SampleRun* is created,

```
>>> sample_set.df.loc["EP2421"].run
<msAI.samples.SampleRun object at 0x7f063fed80d0>
```

But MS data is not available until initialized.

```
>>> sample_set.df.loc["EP2421"].run.ms.spectra
Traceback (most recent call last):
  File "<input>", line 1, in <module>
AttributeError: 'NoneType' object has no attribute 'spectra'
```

Initialize all MS data.

```
>>> sample_set.init_all_ms()
```

Access MS data and metadata.

```
>>> sample_set.df.loc["EP2421"].run.ms.run_date
'2017-06-28T04:10:21Z'
```

```
>>> sample_set.df.loc["EP2421"].run.ms.spectra
             rt  peak_count          tic  ms_lvl                                                 ␣
↪filters
299    3.018841        1745   46977344.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
301    3.039366        1836   48066048.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
303    3.060012        2060   47754260.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
305    3.080646        1828   46855808.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
307    3.101156        1847   48759696.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
         ...         ...          ...     ...                                              .
↪..
1591  15.918533        3416  118047380.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
1593  15.938479        3328  128021860.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
1595  15.958450        3348  128402500.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
1597  15.978360        3156  152132620.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
1599  15.998312        3285  174533700.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
[651 rows x 5 columns]
```

```
>>> sample_set.df.loc["EP2421"].run.ms.peaks
                            rt          mz             i
spec_id peak_number
299     0              3.018841  115.03919  36447.125000
        1              3.018841  115.05045   2975.487549
        2              3.018841  115.07568   2015.634644
        3              3.018841  115.51699   1233.632690
        4              3.018841  115.96244   4875.453613
                          ...         ...          ...
1599    3280          15.998312  987.60944  12299.823242
```

(continues on next page)

(continued from previous page)

```
        3281         15.998312  989.54504  39011.988281
        3282         15.998312  991.56219  57488.519531
        3283         15.998312  992.56891  21931.212891
        3284         15.998312  993.56921   7275.180176
[1430013 rows x 3 columns]
```

2.2.3 Saving and loading sample sets

In this example workflow so far, the step requiring the most computational resources / time to complete was the step initializing the MS data - where data stored in mzML files is loaded into memory and structured as dataframes. When working with large data sets, this step becomes expensive to repeat.

If *SampleRun* data will be needed again, it can be saved in an alternative format (msAIr file) that enables faster access and smaller storage size. This msAIr file type is created by serializing and compressing a *SampleRun* instance, saving the state of all its in-memory data attributes. While there is an upfront cost to creating a msAIr save, future *SampleRun* instantiations from a msAIr file will be much faster as it is not necessary to parse the mzML file again. Additionally, since the entire *SampleRun* instance is saved, the results of calculations performed or new data attributes created will also be persist.

Saving

Define the paths to the directories where data will be saved.

```
>>> msAIr_dir = "./examples/data/msAIr"
>>> msAIm_dir = "./examples/data/msAIm"
```

Save all the samples in the *SampleSet* as msAIr files to a directory. The same filenames are used with the `.msAIr` extension.

```
>>> sample_set.save_all_ms(msAIr_dir)
```

A sha256 hash value is calculated for each sample and added to the *SampleSet* metadata.

```
>>> sample_set.df['msAIr_hash']
filename
EP0482    67a004385a71045b787c5cdc318d78fee3d890bf287473...
EP2421    fcf4c386c7051b6c5228faa120575a492eddfebf2b9914...
EP2536    b82ef4ddeaab36d5c9d68e2e0e192b1731fc5674430e10...
Name: msAIr_hash, dtype: object
```

Save the *SampleSet* metadata as a msAIm file to a directory, a sha256 hash is returned.

```
>>> sample_set.save_metadata(msAIm_dir, "sample_set1")
'dc0714b6fe0d05e10ef902bbb45f40d79ff50a87528c305c1f8161e0a15aeb6a'
```

Loading

Use the same path to the directory where the msAIr files were saved previously.

```
>>> msAIr_dir = "./examples/data/msAIr"
```

Create a *MSfileSet* from the msAIr files. New mzML files can also be added and used in the same way.

```
>>> msAIr_set = msData.MSfileSet(msAIr_dir)
>>> msAIr_set
         file_type  file_size                                path
filename
EP0482       msAIr   7.870908  examples/data/msAIr/EP0482.msAIr
EP2421       msAIr   9.659162  examples/data/msAIr/EP2421.msAIr
EP2536       msAIr   7.881509  examples/data/msAIr/EP2536.msAIr
```

Compare this set to the original mzML version created above - note the smaller sizes of the `msAIr` files.

```
>>> ms_files
         file_type  file_size                            path
filename
EP0482        mzML  12.862821  examples/data/mzML/EP0482.mzML
EP2421        mzML  15.133800  examples/data/mzML/EP2421.mzML
EP2536        mzML  12.745723  examples/data/mzML/EP2536.mzML
```

Define the path to the msAIm file created above.

```
>>> sample_set1_msAIm_path = "./examples/data/msAIm/sample_set1.msAIm"
```

Load the *SampleMetadata* from the msAIm file - notice the msAIr_hash column has been added.

```
>>> msAIm = SampleMetadata(sample_set1_msAIm_path)
>>> msAIm
           class sampleType       site block treatment plantID  tissue     siteblock ␣
↪ sitetreatment polarity                                        msAIr_hash
filename
EP0482    sample     sample  Rosemount    B1      HIGH    P360    seed  Rosemount_B1 ␣
↪Rosemount_HIGH  unknown  67a004385a71045b787c5cdc318d78fee3d890bf287473...
EP2421    sample     sample  Rosemount    B1        R1    P109  flower  Rosemount_B1 ␣
↪  Rosemount_R1  unknown  fcf4c386c7051b6c5228faa120575a492eddfebf2b9914...
EP2536    sample     sample  Rosemount    B1       LOW    P134    root  Rosemount_B1 ␣
↪ Rosemount_LOW  unknown  b82ef4ddeaab36d5c9d68e2e0e192b1731fc5674430e10...
```

Load the SampleSet and initialize.

```
>>> sample_set1 = SampleSet(msAIr_set, msAIm)
>>> sample_set1.init_all_ms()
>>> sample_set1
         file_type  file_size                                path   class sampleType ␣
↪    site block treatment plantID  tissue     siteblock   sitetreatment polarity     ␣
↪                                    msAIr_hash                                      ␣
↪         run
filename
EP0482       msAIr   7.870908  examples/data/msAIr/EP0482.msAIr  sample     sample ␣
↪Rosemount    B1      HIGH    P360    seed  Rosemount_B1  Rosemount_HIGH  unknown ␣
↪67a004385a71045b787c5cdc318d78fee3d890bf287473...  <msAI.samples.SampleRun object␣
↪at 0x7fda7adb02d0>
EP2421       msAIr   9.659162  examples/data/msAIr/EP2421.msAIr  sample     sample ␣
↪Rosemount    B1        R1    P109  flower  Rosemount_B1    Rosemount_R1  unknown
↪fcf4c386c7051b6c5228faa120575a492eddfebf2b9914...  <msAI.samples.SampleRun object␣
↪at 0x7fda6cf5b0d0>
```

(continues on next page)

(continued from previous page)

```
EP2536       msAIr   7.881509  examples/data/msAIr/EP2536.msAIr  sample     sample␣
↪Rosemount    B1       LOW    P134    root  Rosemount_B1   Rosemount_LOW  unknown␣
↪b82ef4ddeaab36d5c9d68e2e0e192b1731fc5674430e10...  <msAI.samples.SampleRun object␣
↪at 0x7fda7adb0750>
```

Access MS data and metadata in the same way as before.

```
>>> sample_set1.df.loc["EP2421"]
file_type                                                    msAIr
file_size                                                  9.65916
path                                 examples/data/msAIr/EP2421.msAIr
class                                                       sample
sampleType                                                  sample
site                                                     Rosemount
block                                                           B1
treatment                                                       R1
plantID                                                       P109
tissue                                                      flower
siteblock                                              Rosemount_B1
sitetreatment                                          Rosemount_R1
polarity                                                    unknown
msAIr_hash       fcf4c386c7051b6c5228faa120575a492eddfebf2b9914...
run              <msAI.samples.SampleRun object at 0x7fda6cf5b0d0>
Name: EP2421, dtype: object
```

```
>>> sample_set1.df.loc["EP2421"].plantID
'P109'
```

```
>>> sample_set1.df.loc["EP2421"].tissue
'flower'
```

```
>>> sample_set1.df.loc["EP2421"].site
'Rosemount'
```

```
>>> sample_set1.df.loc["EP2421"].treatment
'R1'
```

```
>>> sample_set1.df.loc["EP2421"].run.ms.run_date
'2017-06-28T04:10:21Z'
```

```
>>> sample_set1.df.loc["EP2421"].run.ms.spectra
             rt  peak_count          tic  ms_lvl                                 ␣
↪filters
299    3.018841        1745   46977344.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
301    3.039366        1836   48066048.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
303    3.060012        2060   47754260.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
305    3.080646        1828   46855808.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
307    3.101156        1847   48759696.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
         ...         ...          ...     ...                                      .
↪..
```

(continues on next page)

(continued from previous page)

```
1591  15.918533        3416  118047380.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
1593  15.938479        3328  128021860.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
1595  15.958450        3348  128402500.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
1597  15.978360        3156  152132620.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
1599  15.998312        3285  174533700.0       1  FTMS + p ESI Full ms [115.0000-1000.
↪0000]
[651 rows x 5 columns]
```

```
>>> sample_set1.df.loc["EP2421"].run.ms.peaks
                              rt          mz             i
spec_id peak_number
299     0             3.018841  115.03919  36447.125000
        1             3.018841  115.05045   2975.487549
        2             3.018841  115.07568   2015.634644
        3             3.018841  115.51699   1233.632690
        4             3.018841  115.96244   4875.453613
                         ...        ...          ...
1599    3280         15.998312  987.60944  12299.823242
        3281         15.998312  989.54504  39011.988281
        3282         15.998312  991.56219  57488.519531
        3283         15.998312  992.56891  21931.212891
        3284         15.998312  993.56921   7275.180176
[1430013 rows x 3 columns]
```

CHAPTER

THREE

MSAI MODULES

3.1 __init__

msAI package initialization.

See package README for introduction.

Todo

- Move configuration functions to separate modules

class msAI.__init__.**LogMode**
Bases: enum.Enum

Enumeration of arguments accepted by the *set_logging* function's mode parameter.

See *set_logging* for mode details.

DEV = 1
Specifies logging mode for development.

RELEASE = 2
Specifies logging mode for release.

LIB = 3
Specifies logging mode for use as a library.

NONE = 4
Specifies a silent logging mode.

msAI.__init__.**set_logging**(*mode: msAI.__init__.LogMode*) → logging.Logger
Configures msAI logging for development, release, library, or silent use.

Parameters **mode** – The logging configuration to set.

***LogMode.DEV*:**

- Logging exceptions will be raised
- Messages of severity INFO and higher are displayed on console
- Messages of severity DEBUG and higher are saved to the log file
- Log file is overwritten each run

***LogMode.RELEASE*:**

- Logging exceptions will NOT be raised
- Messages of severity WARNING and higher are displayed on console
- Messages of severity INFO and higher are saved to the log file

- All log files are saved for each run - named with date/time

***LogMode.LIB*:**

- Logging exceptions will NOT be raised
- Log handlers are left unconfigured Python default will write messages or severity WARNING or higher to console

***LogMode.NONE*:**

- Logging exceptions will NOT be raised
- Root logger will use NullHandler to prevent messages from being displayed

Returns The msAI root logger.

Raises ***RootError*** – For an invalid logging mode.

msAI.__init__.**set_mp_support**(*mode: str = 'auto'*, *workers: Union[str, int] = 'auto'*) → Tuple[bool, int]

Configures msAI multiprocessing.

The package variable MP_SUPPORT is a boolean set to enable / disable multiprocessing in the msAI package. This is necessary as certain operations will fail if the multiprocessing module uses the 'spawn' start method. The start method default is determined by OS type.

Parameters

- **mode** – A string specifying the multiprocessing configuration.

  `auto`: Configures multiprocessing support automatically based on OS. Multiprocessing will be enabled if the multiprocessing start method in use is 'fork'.

  `enable`: Manually enables multiprocessing. Errors may occur if multiprocessing is not fully supported by OS.

  `disable`: Manually disables multiprocessing.

- **workers** – The number of worker processes to use for multiprocessing.

  `auto`: Sets number of workers to CPU count.

Returns A Tuple specifying MP_SUPPORT and working count.

Raises ***RootError*** – For an invalid multiprocessing mode.

3.2 metadata

msAI module for importing sample metadata into dataframes.

Features

- Extraction of metadata from various file types
- Importing metadata into a dataframe
- Verification of metadata usability
- Auto indexing of metadata

Todo

- Move .msAIm saving to this module
- Refactor auto indexing

- Add anomaly detection
- Add additional file types: TBD...

msAI.metadata.**logger: logging.Logger = <Logger msAI.metadata (DEBUG)>**
Module logger.

class msAI.metadata.**SampleMetadata**(*file_path: str*, *auto_index: bool = True*)
Bases: object

Imports sample metadata from a supported file type into a dataframe and assigns an index.

Supported file types: *.csv*, *.msAIm*, TBD... (A *.msAIm* file can be created from a previous *SampleSet*).

Content from the metadata file is initially imported into a dataframe with a default numerical index. By default, metadata labels and values are analyzed and if possible, a new index is assigned from an existing column. This index is used by *SampleSet* to match this metadata with corresponding MS data in *MSfileSet*.

Requirements to auto index metadata imported into a dataframe:

- Dataframe has 1 or more rows
- Dataframe has 2 or more columns
- For one and only one column:
  - All column values are unique
  - All entries/rows have a value for this column

__init__(*file_path: str*, *auto_index: bool = True*)
Initializes an instance of SampleMetadata class.

Parameters

- **file_path** – A string representation of the path to the metadata file. Path can be relative or absolute.
- **auto_index** – A boolean indicating if the metadata should be automatically indexed. Default is True.

Raises ***MetadataInitError*** – For an invalid file type/extension.

file_path: str = None
A string representation of the path to the metadata file.

_hf: DF = None
High fidelity copy of imported data.

Leave this original data untouched for future reference if needed.

df: MetaDF = None
The metadata dataframe.

__repr__()
Returns a string representation of the metadata dataframe.

_verify_import()
Verifies the imported metadata is usable.

Ensures at least one metadata entry/row and at least two metadata labels/columns exist.

Raises ***MetadataVerifyError*** – If No metadata entries or not enough metadata labels are found

_auto_index()
Attempts to identify and set the dataframe index from a metadata label/column.

This index is used to match metadata to *SampleRun*.

describe()
Prints a summary of metadata contents.

set_index(*new_index: str*)
Manually sets the metadata dataframe index to an existing label/column.

This index is used to match metadata to *SampleRun*.

Parameters **new_index** – The name of the metadata label/column to use as the index.

3.3 msData

msAI module for importing mass spectrometry data into dataframes.

Features

- Extraction of data from MS files (mzML, TBD...)
- Creation of in-memory data structures for spectra / peaks values
- Building a set of MS data files

Todo

- Change MSfile to dataclass
- Change properties to attributes
- Modify public / private
- Create types for peaks and spectra dataframes

msAI.msData.**logger = <Logger msAI.msData (DEBUG)>**
Module logger.

class msAI.msData.**MSfile**
Bases: object

Interface class for accessing data from a MS file stored in various file types.

Subclass implementations provide support for the various file types and override the init method to set values. The *peaks* and *spectra* properties hold data structured in dataframes.

__init__()
Initializes an instance of MSfile class.

No need to call this superclass initialization, as subclasses provide values for all attributes initialized here.

_run_id: str = None

_run_date: str = None

_ms_file_version: str = None

_spectrum_count: int = None

_peak_count: int = None

_tic_sum: float = None

_peaks: DF = None

_spectra: DF = None

property run_id
Get the sample's run ID as specified from its MS data file.

property run_date
Get the date the sample was run as specified from its MS data file.

property ms_file_version
Get the data format version in which the sample was originally saved as specified from its ms file.

Note: Currently, this is equivalent to mzML version number.

property spectrum_count
Get the number of MS spectra from a sample run.

This value is calculated from the number of spectra imported, rather than from MS file metadata.

property peak_count
Get the total number of MS peaks from all MS spectra in sample run.

property tic_sum
Get the total ion current sum of all spectra in sample run

property peaks
Get a dataframe of all peaks in a MS file.

Dataframe structure

First Index Level: spec_id
Second Index Level: peak_number
Columns: rt, mz, i

property spectra
Get a dataframe of all spectra in an MS file.

Dataframe structure

Index: spec_id
Columns: rt, peak_count, tic, ms_lvl, filters

class msAI.msData.**MZMLfile**(*mzml_file_path: str*)
Bases: *msAI.msData.MSfile*

Class to access MS data stored in an mzML file.

__init__(*mzml_file_path: str*)
Initializes an instance of MZMLfile class.

Parameters **mzml_file_path** – A string representation of the path to the mzML data file. Path can be relative or absolute.

_run_id = None

_run_date = None

_ms_file_version = None

_spectrum_count = None

_tic_sum = None

_peak_count = None

_create_spectrum_peaks_df(*spectrum*)
Creates a dataframe of all the peaks for a single spectrum in an mzML file.

_create_spectrum_df(*spectrum*)
Creates a dataframe of all the spectra in an mzML file.

_create_dfs()
Creates spectra and peaks dataframes for an mzML file.

This method sets the following properties:

- self._peaks
- self._spectra

_peaks = None

_spectra = None

property ms_file_version
Get the data format version in which the sample was originally saved as specified from its ms file.

Note: Currently, this is equivalent to mzML version number.

property peak_count
Get the total number of MS peaks from all MS spectra in sample run.

property peaks
Get a dataframe of all peaks in a MS file.

Dataframe structure

First Index Level: spec_id
Second Index Level: peak_number
Columns: rt, mz, i

property run_date
Get the date the sample was run as specified from its MS data file.

property run_id
Get the sample's run ID as specified from its MS data file.

property spectra
Get a dataframe of all spectra in an MS file.

Dataframe structure

Index: spec_id
Columns: rt, peak_count, tic, ms_lvl, filters

property spectrum_count
Get the number of MS spectra from a sample run.

This value is calculated from the number of spectra imported, rather than from MS file metadata.

property tic_sum
Get the total ion current sum of all spectra in sample run

class msAI.msData.**MSfileSet**(*dir_path: str*, *data_type: str = 'all'*, *recursive: bool = True*)
Bases: object

Class to create a set of MS files from a data directory.

Creating a set enables a large number of datafiles to be viewed / manipulated as a dataframe, without loading their entire contents into memory.

By default, contents of sub directories will be recursively included. However, an error is raised if included filenames are duplicated. A Set can include any MSfile type (mzML, msAIr, or a mix). By default, any datafile matching these extensions will be included. An exclusive type may alternatively be specified.

mzML_exts: ClassVar[List[str]] = ['mzML', 'mzml', 'MZML']
File extensions considered to be mzML files.

msAIr_exts: ClassVar[List[str]] = ['msAIr', 'msair', 'MSAIR']
File extensions considered to be msAIr files.

__init__(*dir_path: str*, *data_type: str = 'all'*, *recursive: bool = True*)
Initializes an instance of MSfileSet class.

Parameters

- **dir_path** – A string representation of the path to the data directory. Path can be relative or absolute.
- **data_type** – (`all`, `mzML`, `msAIr`) The type of MS files to include in the set. By default, all types are included.
- **recursive** – A boolean indicating if files in subdirectories are included in the set. Defaults to `True`.

Raises ***MSfileSetInitError*** – For duplicated filenames.

property df
Get a dataframe of MS files.

Dataframe structure

Index: name (from filename)
Columns: type, size_MB, path

3.4 samples

msAI module to create a unified set of MS data samples paired with any additional metadata.

Features

- Creation of a sample set from a directory of MS data files
- Pairing of MS data and sample metadata
- Extraction of sample metadata from csv files
- Saving / loading data (serialization, compression, checksum)

Todo

- init_ms mp logging calls
- Create a msFile subclass form msAIr files - and move loading to that class

`msAI.samples.`**logger = <Logger msAI.samples (DEBUG)>**
Module logger.

class `msAI.samples.`**SampleSet**(*ms_file_set*, **sample_metadata*, *metadata_inner_merge=False*, *init_ms=False*)
Bases: `object`

Class to create a dataframe of a set of SampleRuns created from a MSfileSet and paired with 0 or more SampleMetadata.

SampleMetadata objects provide a dataframe with a matching index to MSfileSet.

A dataframe is created from a set of MS data files (MSfileSet) and joined with matching SampleMetadata. By default (metadata_inner_merge=False), all files in the passed MSfileSet will be included- even if no matching metadata is found. Passing metadata_inner_merge=True, will only include MS files that have matching metadata for every SampleMetadata included.

SampleRun objects are created for each MS file when the SampleSet is created, but MS data is not initialized until called.

__init__(*ms_file_set*, **sample_metadata*, *metadata_inner_merge=False*, *init_ms=False*)
Initialize self. See help(type(self)) for accurate signature.

static _set_run_metadata(*sample_name*, *run*, *metadata*)
Adds metadata to SampleRuns, if possible.

Samples with missing metadata are logged.

_create_sampleruns()
Creates of SampleRuns for all samples in the SampleSet.

Multi or single process according to MP_SUPPORT.

_create_sampleruns_sp()
Single-process creation of SampleRuns for all samples in the SampleSet.

static _create_samplerun_mpf(*row*)
Multiprocessing function to create a single SampleRun for a row/sample in the SampleSet.

_create_sampleruns_mp()
Multiprocess creation of SampleRuns for all samples in the SampleSet.

_init_all_ms_sp()
Single-process initialization of MS data for all samples in the SampleSet.

static _init_ms_mpf(*row*)
Multiprocessing function to initialize the MS data of a single SampleRun (a row of a SampleSet).

_init_all_ms_mp()
Multiprocess initialization of MS data for all samples in the SampleSet.

_save_all_ms_sp(*dir_path*)
Single-process save of MS data for all samples in the SampleSet.

static _save_ms_mpf(*dir_path*, *row*)
Multiprocessing function to save the MS data of a single SampleRun (a row of a SampleSet).

_save_all_ms_mp(*dir_path*)
Multiprocess save of MS data for all samples in the SampleSet.

property df
Get a dataframe of sample runs paired with sample metadata.

Index: name (from filename) Columns: type, size_MB, path, (metadata...), run (python object)

init_all_ms()
Initializes MS data for all samples in the SampleSet.

Multi or single process according to MP_SUPPORT.

save_all_ms(*dir_path*)
Saves MS data for all samples in the set as .msAIr files (in dir_path) and add hash value to metadata (msAIr_hash).

Multi or single process according to MP_SUPPORT.

save_metadata(*dir_path*, *filename*)
Saves all metadata for a SampleSet as a .msAIm file.

This enables faster loading when recreating a sample set, and verification of msAIr_hash values.

Contents will include all metadata passed at SampleSet creation + msAIr hash values (if created). MSfile data and SampleRuns are not included, as data paths may change.

Data is serialized with pickle and compressed via bzip2. A sha256 hash is returned.

class msAI.samples.**SampleRun**(*file_path*)
Bases: object

Holds data from a MS analysis run of a sample and any additional metadata.

A *SampleRun* instance is created with a path reference that is used to create a future MSfile, or load MS data from a previously saved *SampleRun*. This allows a cheep view of this data to exist without importing it all into memory. A very large number of *SampleRun* instances can be created and their MS data initialized when needed.

Typically, *SampleRun* instances are not manually created, but instead arise from a *SampleSet*.

Data is extracted from a supported MS file type or loaded from a previous msAIr save. File type is determined by file extension (.mzML .msAIr). A sha256 hash may be provided for a .msAIr file which will be verified during init_ms().

_ms: msAI.msData.MSfile = None
MS data from a`.MSfile` or a msAIr save.

_metadata: NewType.<locals>.new_type = None
The metadata as a *Series*.

__init__(*file_path*)
Initializes an instance of SampleRun class.

Parameters **file_path** – A string representation of the path to the MS file. Path can be relative or absolute.

file_path: str = None
A string representation of the path to the MS file.

property ms
Access to MS data of a sample run.

property metadata
Access to sample metadata.

property msAIr_hash
Hash value of the SampleRun.

This value was generated when the SampleRun was saved, and re-associated from SampleSet metadata.

save(*dir_path*, *filename*)
Save a SampleRun ms data as a msAIr file for fast loading later.

Data is serialized with pickle and compressed via bzip2. A sha256 hash is returned.

init_ms()
Initialize MS data at the SampleRun's set file_path from a .mzML or .msAIr file.

For a .msAIr file, it is first tested against a sha256 hash, if provided. Data is decompressed via bzip2 and deserialized with pickle.

3.5 miscUtils

Miscellaneous utilities used by msAI.

Todo

- Add type info for funcs passed as arguments

msAI.miscUtils.**logger = <Logger msAI.miscUtils (DEBUG)>**
Module logger.

class msAI.miscUtils.**FileGrabber**
Bases: object

Functions to grab files.

static multi_extensions(*directory: str*, **extensions: str*, *recursive: bool = True*) → Iterable[pathlib.Path]
Creates an iterator of path objects to all files in a directory matching the passed extensions.

Use `str(path_obj)` to get the platform independent path string. Subdirectories will be recursively searched by default.

Parameters

- **directory** – A string representation of the path to the directory. Path can be relative or absolute.
- **extensions** – One or more file extensions specified as strings without leading (.).
- **recursive** – A boolean indicating if files in subdirectories are included. Defaults to `True`.

Returns An iterator of path objects to all files found.

static path_type(*directory: str = '.'*) → str
Get the path type of a directory.

Path type is identified by the class of Path object created. This test is used for determining what glob patterns to apply based on path case sensitivity. Windows paths are case insensitive, while Posix paths are case sensitive.

Parameters **directory** – A string representation of the path to the directory. Path can be relative or absolute. Defaults to current directory.

Returns A string of either `'posix'` or `'windows'`, indicating the path type.

Raises ***MiscUtilsError*** – For unknown path type.

class msAI.miscUtils.**Sizer**
Bases: object

Functions to measure memory / storage sizes.

static obj_mb(*obj: object*) → float
Measures the memory size of a python object in MBs.

Parameters **obj** – The python object to measure.

Returns The Python object's size in memory in MBs.

static print_obj_mb(*obj: object*)
Prints the memory size of a python object in MBs to 4 decimals.

Parameters **obj** – The python object to measure.

static file_mb(*file: str*)
Measures the storage size of a file in MBs.

Parameters **file** – A string representation of the path to the file to measure. Path can be relative or absolute.

Returns The storage size of the file in MBs.

static print_file_mb(*file: str*)
Prints the storage size of a file in MBs to 4 decimals.

Parameters **file** – A string representation of the path to the file to measure. Path can be relative or absolute.

class msAI.miscUtils.**Saver**
Bases: object

Functions to save / load, serialize, and compress files and objects.

static save_obj(*obj: object*, *file: str*) → str
Saves a python object to the path / filename given.

Data is serialized with pickle and compressed via bzip2. A sha256 hash is also calculated.

Parameters

- **obj** – The python object to save.
- **file** – A string representation of the path to the file to save. Path can be relative or absolute.

Returns A sha256 hash as a string.

static get_hash(*file: str*) → str
Calculates the sha256 hash of a file.

Parameters **file** – A string representation of the path to the file to calculate a hash for. Path can be relative or absolute.

Returns A sha256 hash as a string.

static verify_hash(*file: str*, *test_hash: str*) → bool
Verifies the sha256 hash of a file.

Parameters

- **file** – A string representation of the path to the file to calculate and compare hash value for. Path can be relative or absolute.
- **test_hash** – A sha256 hash as a string to test against.

Returns A boolean indicating if the hash value is verified. True means the calculated hash matches the test hash.

static load_obj(*file: str*, *test_hash: Optional[str] = None*) → Tuple[object, Optional[bool]]
Loads a previously saved object.

The file will be tested against a sha256 hash, if provided. Data is decompressed via bzip2 and deserialized with pickle.

Parameters

- **file** – A string representation of the path to the file to load the object from. Path can be relative or absolute.
- **test_hash** – A sha256 hash as a string to test against.

Returns A tuple of the object and an optional boolean indicating if the hash of the saved file was verified.

class msAI.miscUtils.**MultiTaskDF**
Bases: object

Functions to parallelize work on dataframes through multiprocessing.

static _partition_by_rows(*df_in: pandas.core.frame.DataFrame*, *subset_func*) → pandas.core.frame.DataFrame
Partitions a dataframe into subsets across rows and assigns a worker to each to apply a function.

Creates a process pool with a number of workers equal to cpu count (by default), and splits the dataframe `df_in` into a number of subsets equal to number of workers. Each worker applies the `subset_func` to a dataframe subset in parallel.

Parameters

- **df_in** – The input dataframe.
- **subset_func** – A partial object containing the function to apply to each dataframe subset. This is received as a partial object, and its call input is completed with a dataframe subset after the dataframe is split.

Returns: A dataframe formed by concating all subset results.

static _run_on_subset_rows(*func*, *df_subset: pandas.core.frame.DataFrame*) → pandas.core.frame.DataFrame
Applies a function to each row in a dataframe subset.

Rows are passed to `func` as `Series` objects whose index is the dataframe's columns.

Parameters

- **func** – The function to apply to each row in the `df_subset`. This function must be a static method and return the row, reflecting the results. Additional arguments can be passed with a partial object by the caller.
- **df_subset** – A dataframe subset, to which a single worker applies `func` to all rows.

Returns: A dataframe reflecting the changes from the applied `func`.

static parallelize_on_rows(*df: pandas.core.frame.DataFrame*, *func*) → pandas.core.frame.DataFrame
Applies a function to rows in a dataframe in parallel.

Parameters

- **df** – The input dataframe.
- **func** – The function to apply to each row in the `df`. This function must be a static method and return the row, reflecting the results. Additional arguments can be passed with a partial object by the caller.

Returns: A new dataframe reflecting the changes from the applied `func`.

class msAI.miscUtils.**EnvInfo**
Bases: object

Functions to get info about the environment running python.

static platform() → str
Get a string (multiline) describing the platform in use.

static os() → str
Get a string (multiline) describing the operating system in use.

static python() → str
Get a string (multiline) describing the python interpreter in use.

static all() → str
Get a string (multiline) describing the environment running python.

static mp_method() → str
Get a string describing the start method used by the multiprocessing module to create new processes.

Defaults are set according to OS type:

POSIX = 'fork'
Windows = 'spawn'

Use this function to test and switch to single processing if necessary. Certain functions will fail under the spawn start method.

3.6 miscDecos

Miscellaneous decorator functions used by msAI.

msAI.miscDecos.**logger = <Logger msAI.miscDecos (DEBUG)>**
Module logger.

msAI.miscDecos.**log_timer**(*func*)
Decorator to log the runtime and dataframe shape (if present) of the decorated function.

If the passed function's instance object has a dataframe attribute (`df`), its shape will be included. The wrapped function's own module logger will be used to create the log.

3.7 types

Module containing types used by msAI.

Todo

- Prevent pandas docs from importing and switch back to type alias

msAI.types.**logger: logging.Logger = <Logger msAI.types (DEBUG)>**
Module logger.

msAI.types.**Series**(*x*)
Type derived from Pandas Series.

msAI.types.**DF**
alias of `pandas.core.frame.DataFrame`

msAI.types.**MetaDF**(*x*)
Type derived from *DF* for use with metadata.

3.8 errors

msAI exception hierarchy.

msAI.errors.**logger = <Logger msAI.errors (DEBUG)>**
Module logger.

exception msAI.errors.**msAIerror**
Bases: Exception

Base class for exceptions in the msAI package.

message: str = None
Explanation of the cause of the error.

exception msAI.errors.**RootError**(*message: str*)
Bases: *msAI.errors.msAIerror*

Exceptions raised for errors in msAI package __init__.

__init__(*message: str*)
Initializes an instance of RootError.

Parameters **message** – Explanation of the cause of this error.

message = None

exception msAI.errors.**MiscUtilsError**(*message: str*)
Bases: *msAI.errors.msAIerror*

Exceptions raised for errors in miscUtils the module.

__init__(*message: str*)
Initializes an instance of MiscUtilsError.

Parameters **message** – Explanation of the cause of this error.

message = None

exception msAI.errors.**MetadataError**(*message: str*)
Bases: *msAI.errors.msAIerror*

Exceptions raised for errors in the metadata module.

__init__(*message: str*)
Initializes an instance of MetadataError.

Parameters **message** – Explanation of the cause of this error.

message = None

exception msAI.errors.**MetadataInitError**(*message: str*)
Bases: *msAI.errors.MetadataError*

Exceptions raised for errors in initializing metadata.

__init__(*message: str*)
Initializes an instance of MetadataInitError.

Parameters **message** – Explanation of the cause of this error.

message = None

exception msAI.errors.**MetadataVerifyError**(*message: str*)
Bases: *msAI.errors.MetadataError*

Exceptions raised for errors in verifying imported metadata.

__init__(*message: str*)
Initializes an instance of MetadataVerifyError.

Parameters **message** – Explanation of the cause of this error.

message = None

exception msAI.errors.**MetadataIndexError**(*message: str*)
Bases: *msAI.errors.MetadataError*

Exceptions raised for errors in setting metadata index.

__init__(*message: str*)
Initializes an instance of MetadataIndexError.

Parameters **message** – Explanation of the cause of this error.

message = None

exception msAI.errors.**SampleRunError**(*message: str*)
Bases: *msAI.errors.msAIerror*

Exceptions raised for errors in the SampleRun module.

__init__(*message: str*)
Initializes an instance of SampleRunError.

Parameters **message** – Explanation of the cause of this error.

message = None

exception msAI.errors.**SampleRunMSinitError**(*message: str*)
Bases: *msAI.errors.SampleRunError*

Exceptions raised for errors in initializing MS data in SampleRun.

__init__(*message: str*)
Initializes an instance of SampleRunMSinitError.

Parameters **message** – Explanation of the cause of this error.

message = None

exception msAI.errors.**MSdataError**(*message: str*)
Bases: *msAI.errors.msAIerror*

Exceptions raised for errors in the msData module.

__init__(*message: str*)
Initializes an instance of MSdataError.

Parameters **message** – Explanation of the cause of this error.

message = None

exception msAI.errors.**MSfileSetInitError**(*message: str*)
Bases: *msAI.errors.MSdataError*

Exceptions raised for errors in initializing MSfileSet.

__init__(*message: str*)
Initializes an instance of MSfileSetInitError.

Parameters **message** – Explanation of the cause of this error.

message = None

CHAPTER

FOUR

DEVELOPMENT

msAI development information.

4.1 Source Code

This project uses git for source code version control. The main msAI source code repository is hosted on GitHub at: https://github.com/IsotopeC/msAI.

4.2 Testing

msAI uses pytest as the main testing tool. Tests and test data are located in the project's `tests` directory. The `fixtures` module defines reusable fixtures for use by test functions.

4.3 Documentation

msAI uses Sphinx to generate documentation from reStructuredText. The project's reStructuredText documents are located in the `docs` directory. Sphinx extensions are used to automatically parse source code docstrings and generate documentation. Docstrings are written in Google style.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

INDEX

Symbols

P

R

S

T

V