pepfrag

Jun 04, 2020

Contents:

CHAPTER 1

Installation

pepfrag can be installed via PyPI:

```
pip install pepfrag
```

pepfrag is compatible with python > 3.7 and most operating systems. The package has been tested on

- Windows 10
- MacOS 10.15
- Ubuntu 18.04 LTS

Because *pepfrag* includes C/C++ extensions, installation requires the presence of a C++ 11 compatible compiler on your machine.

CHAPTER 2

Usage

2.1 Peptide Construction

pepfrag provides one key public class: *Peptide*. This class includes public methods for calculating the mass of the peptide, including any configured modifications (*ModSite* s), and the peptide fragment ions, with configurable neutral losses.

A *Peptide* can be constructed from its amino acid sequence, charge state and modifications, for example:

```
from pepfrag import ModSite, Peptide

peptide = Peptide(
    "ABCMPK",
    2,
    (ModSite(15.994915, 4, "Oxidation"), ModSite(304.20536, "nterm", "iTRAQ8plex"))
)
```

Peptide modifications are defined using a sequence of *ModSite* instances.

Additional keyword arguments are available, allowing the use of average masses instead of monoisotopic masses and introducing radical peptide fragment generation.

2.2 Fragment Generation

Fragment ions can be generated using the *fragment()* method; for efficiency when the same *Peptide* instance is used repeatedly, the resulting fragments are cached in the *fragment_ions* attribute. This cache is invalidated if the instance attributes are changed.

The generated fragment ions can be customized using the `ion_types` argument to *fragment()*, which takes a dictionary mapping the desired *IonType* s to their planned neutral losses. The default is:

```
from pepfrag import IonType

DEFAULT_IONS = {
    IonType.precursor: ['H2O', 'NH3', 'CO2'],
    IonType.imm: [],
    IonType.b: ['H2O', 'NH3', 'CO'],
    IonType.y: ['NH3', 'H2O'],
    IonType.a: [],
    IonType.c: [],
    IonType.z: []
}
```

The generated ions can be changed by providing a custom `ion_types` dictionary when calling *`fragment()`*, for example:

```
from pepfrag import IonType, Peptide

peptide = Peptide('AMYK', 2, [])
peptide.fragment(ion_types={
    IonType.precursor: [],
    IonType.b: ['NH3'],
    IonType.y: ['H2O']
})
```

outputs the following fragment ions, including precursor ions, *b* ions with *NH3* losses and *y* ions with *H2O* losses:

```
[
    (72.044390252029, 'b1[+]', 1),
    (55.01784115090901, '[b1-NH3][+]', 1),
    (147.11280416609898, 'y1[+]', 1),
    (129.10223948206897, '[y1-H2O][+]', 1),
    (203.084875340499, 'b2[+]', 2),
    (186.058326239379, '[b2-NH3][+]', 2),
    (310.17613269973896, 'y2[+]', 2),
    (292.16556801570897, '[y2-H2O][+]', 2),
    (366.14820387413897, 'b3[+]', 3),
    (349.121654773019, '[b3-NH3][+]', 3),
    (183.57774017050897, 'b3[2+]', 3),
    (175.06446561994898, '[b3-NH3][2+]', 3),
    (441.21661778820896, 'y3[+]', 3),
    (423.206053104179, '[y3-H2O][+]', 3),
    (221.11194712754397, 'y3[2+]', 3),
    (212.10666478552898, '[y3-H2O][2+]', 3),
    (512.253731573359, '[M+H][+]', 4),
    (256.63050402011896, '[M+H][2+]', 4)
]
```

2.2.1 Customizing Neutral Losses

`pepfrag` includes a number of common neutral losses available using only their string names. These are: *NH3*, *H2O*, *CO2* and *CO*.

Additional neutral losses can be specified using a tuple of *(label, mass)*. For example:

```
from pepfrag import IonType
```

(continues on next page)

(continued from previous page)

```
ion_types = {
    IonType.b: [('testLoss1', 17.04), 'NH3']
}
```

This would generate *b* ions, along with *b-testLoss1* and *b-NH3* fragment ions.

CHAPTER 3

pepfrag package

3.1 Module contents

class pepfrag.**Mass**(*mono: float*, *avg: float*)
Bases: object

Represents a mass pair of monoisotopic and average masses.

Parameters

- **mono** – Monoisotopic mass.
- **avg** – Average mass.

class pepfrag.**MassType**
Bases: enum.Enum

An enumeration representing the possible mass types.

Note that the values of these enumerations correspond to their index in *Mass*, and similarly in the C++ code underneath methods such as calculate_mass.

mono = 0
Monoisotopic mass

avg = 1
Average mass

class pepfrag.**IonType**
Bases: enum.Enum

Enumeration of possible fragment ion types.

precursor = 1
Precursor ions

imm = 2
Immonium ions

b = 3
b-type ions

y = 4
y-type ions

a = 5
a-type ions

c = 6
c-type ions

z = 7
z-type ions

class pepfrag.**ModSite**(*mass: float, site: Union[int, str], mod: str*)
Bases: object

Class representing an instance of *mod_name* at position *site*.

Parameters

- **mass** – Mass of the modification.
- **site** – Position of the modification. Integer for sequence position, 'nterm' for N-terminus or 'cterm' for C-terminus.
- **mod** – Name of the modification.

class pepfrag.**Peptide**(*sequence: str, charge: int, modifications: Sequence[pepfrag.pepfrag.ModSite], mass_type: pepfrag.constants.MassType = <MassType.mono: 0>, radical: bool = False*)
Bases: object

A class to represent a peptide, including its charge state and any modifications, including PTMs and quantitative tags. The class should be used to fragment the peptides for mass spectrum annotation.

mass_type
Type of masses used in calculations (see *MassType*).

radical
Flag indicating whether the peptide is a radical peptide.

fragment_ions
Cache of generated fragment ions.

seq
Peptide amino acid sequence.

charge
Peptide charge state.

mods
Peptide modifications.

peptide_mass
The mass of the peptide along the sequence, with each position calculated separately.

Note: In the returned list, index 0 is the N-terminus mass, while index -1 is the C-terminus mass.

mass
Total mass of the peptide, including modifications.

clean_fragment_ions()
Cleans the cached *fragment_ions*.

calculate_mass() → List[float]
Calculates the theoretical mass of the peptide along the sequence, including any modifications.

Returns Masses along the peptide sequence. Index 0 is the N-terminus mass, while index -1 is the C-terminus mass.

fragment(*ion_types: Optional[Dict[pepfrag.pepfrag.IonType, List[Union[str, Tuple[str, float]]]]] = None*, *force: bool = False*) → List[Tuple[float, str, int]]
Fragments the peptide to generate the ion types specified.

Parameters

- **ion_types** – Dictionary of *IonType* s to list of configured neutral losses. Only fragments for *IonType* s specified here will be generated.
- **force** – Force re-fragmentation of the peptide.

Returns List of generated ions, as tuples of *(fragment mass, ion label, sequence position)*.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index