seq-to-first-iso

Release 1.1.0

Lilian Yang-crosson, Pierre Poulain

Dec 20, 2019

CONTENTS

Version 1.1.0

Seq-to-first-iso computes the first two isotopologues intensity from peptide sequences and charges.

It differentiates labelled and unlabelled amino acids with a 99.99 % 12C enrichment.

CHAPTER

ONE

INSTALLATION

To install `seq-to-first-iso`, use:

```
pip install seq-to-first-iso
```

CHAPTER

TWO

USER MANUAL

2.1 Theoretical background

2.1.1 Isotopologues in mass spectrometry

Depending on their isotopic composition, peptides have different isotopologues.

Element	Isotope	Relative abundance (%)
Hydrogen	H[1]	99.9885
Hydrogen	H[2]	0.0115
Carbon	C[12]	98.93
Carbon	C[13]	1.07
Nitrogen	N[14]	99.632
Nitrogen	N[15]	0.368
Oxygen	O[16]	99.757
Oxygen	O[17]	0.038
Oxygen	O[18]	0.205
Sulfur	S[32]	94.93
Sulfur	S[33]	0.76
Sulfur	S[34]	4.29

Stable isotopes of most common organic elements in peptides, values are taken from MIDAs[1]

The first isotopologue (noted ***M0***) contains peptides which elements are composed of only the **lightest stable isotopes**. In contrast the second isotopologue (***M1***) has one of its element with a **supplementary neutron**. Isotopologues follow notation *Mn* where *n* is the number of supplementary neutrons in the chemical formula compared to *M0*.

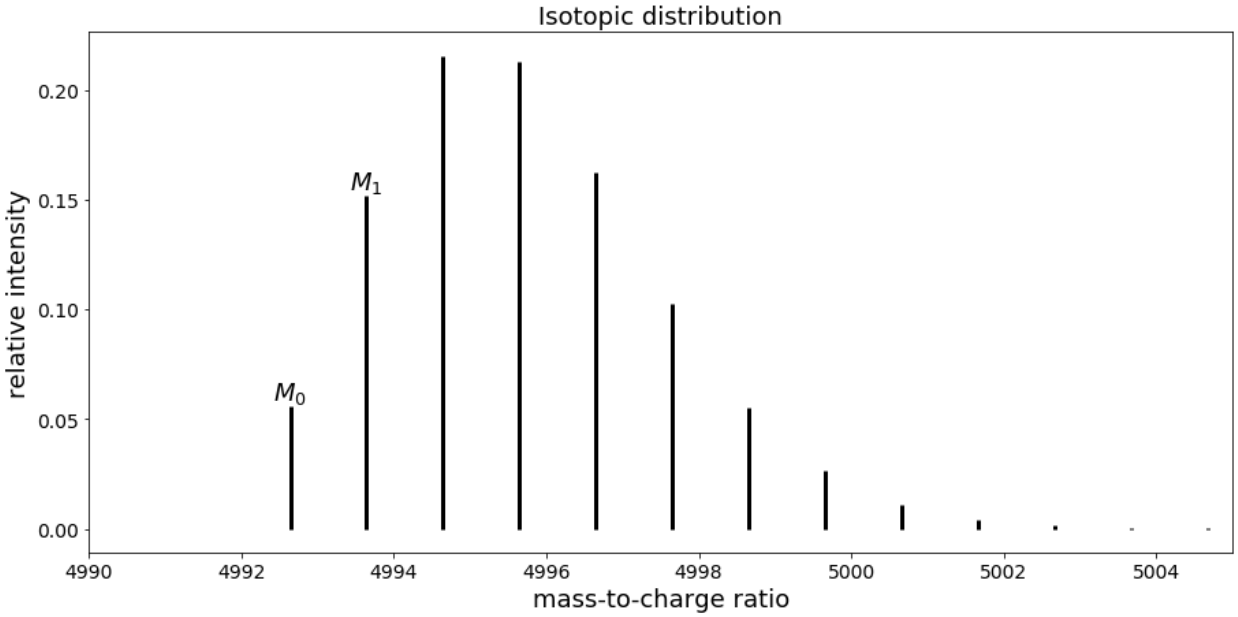
e.g: the chemical formula of Glycine is "C2H5O2N1", the different isotopic compositions for isotopologues *M0* and *M1* are described in the table below

Isotopologue	Carbon	Hydrogen	Oxygen	Nitrogen
M0	C[12]*2	H[1]*5	O[16]*2	N[14]*1
M1	C[12]*1, C[13]*1	H[1]*5	O[16]*2	N[14]*1
M1	C[12]*2	H[1]*4, H[2]*1	O[16]*2	N[14]*1
M1	C[12]*2	H[1]*5	O[16]*1, O[17]*1	N[14]*1
M1	C[12]*2	H[1]*5	O[16]*2	N[15]*1

Composition of M0 and M1 of Glycine

M0* can only have one composition**, meanwhile there are **multiple combinations for *M1, each with one of its elements swapped with a heavier isotope. The complexity of formulas increases even more with further isotopologues.

In high-resolution mass spectrometry, the mass spectrometer is able to differentiate peaks of isotopologues.



Mass spectrum of peptide with sequence "VGEVFINYIQRQNELFQGKLAYLIIDTCLSIVRPNDSKPLDNR"

In that case, the first peak corresponds to *M0* while the second one is *M1*, the third one *M2 etc*.

2.1.2 Computation of isotopologue intensity

Isotopologue intensity can be computed analytically. Formulas are adapted from an article by Wang, Benlian et al.[2].

We consider a peptide of formula

$$\mathrm{C}_{n(\mathrm{C})}\mathrm{H}_{n(\mathrm{H})}\mathrm{O}_{n(\mathrm{O})}\mathrm{N}_{n(\mathrm{N})}\mathrm{S}_{n(\mathrm{S})},$$

where *n(C)*, *n(H)*, *n(O)*, *n(N)* and *n(S)* denote the number of atoms of carbon, hydrogen, oxygen, nitrogen and sulfur respectively.

Compute M0 intensity

For such peptide, the normalized intensity of the monoisotopic ion is given by:

$$M_0 = a(\mathrm{C}[12])^{n(\mathrm{C})} \times a(\mathrm{H}[1])^{n(\mathrm{H})} \times a(\mathrm{O}[16])^{n(\mathrm{O})} \times a(\mathrm{N}[14])^{n(\mathrm{N})} \times a(\mathrm{S}[32])^{n(\mathrm{S})},$$

where *a(isotope)* is the relative abundance of the isotope.

Compute M1 intensity

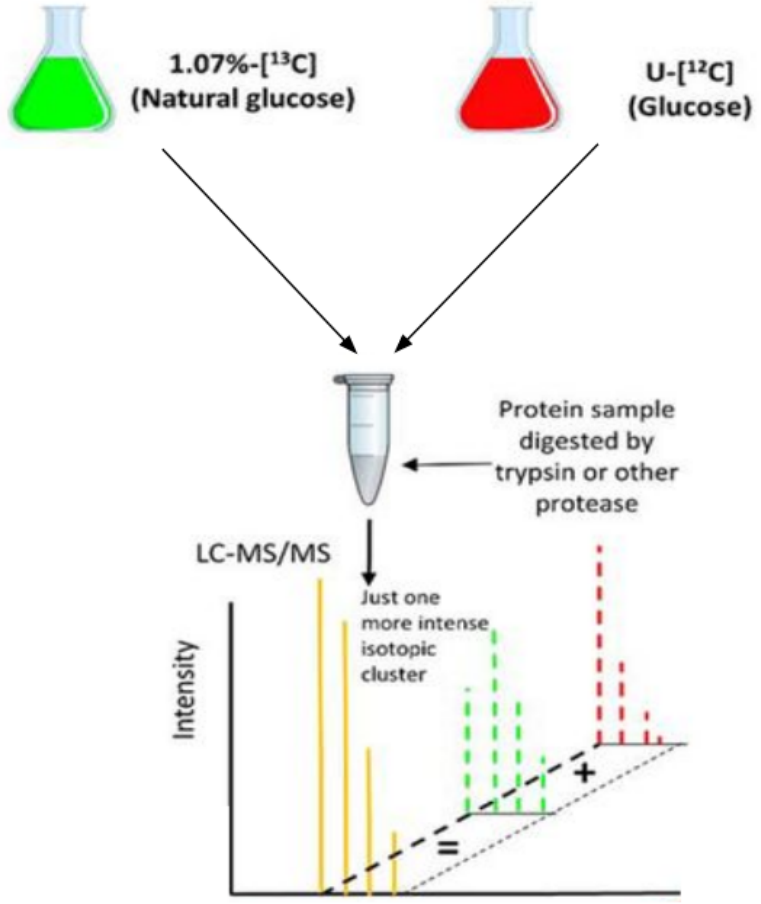
Following a polynomial expansion, intensity of the second isotopologue *M1* is:

$$\begin{aligned}
M_1 = {} & n(\mathrm{C}) \times a(\mathrm{C}[12])^{n(\mathrm{C})-1} \times a(\mathrm{C}[13]) \times a(\mathrm{H}[1])^{n(\mathrm{H})} \times a(\mathrm{O}[16])^{n(\mathrm{O})} \times a(\mathrm{N}[14])^{n(\mathrm{N})} \times a(\mathrm{S}[32])^{n(\mathrm{S})} \\
& + n(\mathrm{H}) \times a(\mathrm{C}[12])^{n(\mathrm{C})} \times a(\mathrm{H}[1])^{n(\mathrm{H})-1} \times a(\mathrm{H}[2]) \times a(\mathrm{O}[16])^{n(\mathrm{O})} \times a(\mathrm{N}[14])^{n(\mathrm{N})} \times a(\mathrm{S}[32])^{n(\mathrm{S})} \\
& + n(\mathrm{O}) \times a(\mathrm{C}[12])^{n(\mathrm{C})} \times a(\mathrm{H}[1])^{n(\mathrm{H})} \times a(\mathrm{O}[16])^{n(\mathrm{O})-1} \times a(\mathrm{O}[17]) \times a(\mathrm{N}[14])^{n(\mathrm{N})} \times a(\mathrm{S}[32])^{n(\mathrm{S})} \\
& + n(\mathrm{N}) \times a(\mathrm{C}[12])^{n(\mathrm{C})} \times a(\mathrm{H}[1])^{n(\mathrm{H})} \times a(\mathrm{O}[16])^{n(\mathrm{O})} \times a(\mathrm{N}[14])^{n(\mathrm{N})-1} \times a(\mathrm{N}[15]) \times a(\mathrm{S}[32])^{n(\mathrm{S})} \\
& + n(\mathrm{S}) \times a(\mathrm{C}[12])^{n(\mathrm{C})} \times a(\mathrm{H}[1])^{n(\mathrm{H})} \times a(\mathrm{O}[16])^{n(\mathrm{O})} \times a(\mathrm{N}[14])^{n(\mathrm{N})} \times a(\mathrm{S}[32])^{n(\mathrm{S})-1} \times a(\mathrm{S}[33])
\end{aligned}$$

We can observe that formulas follow a combinatorial explosion.

2.1.3 Quantify proteins with C[12] enrichment

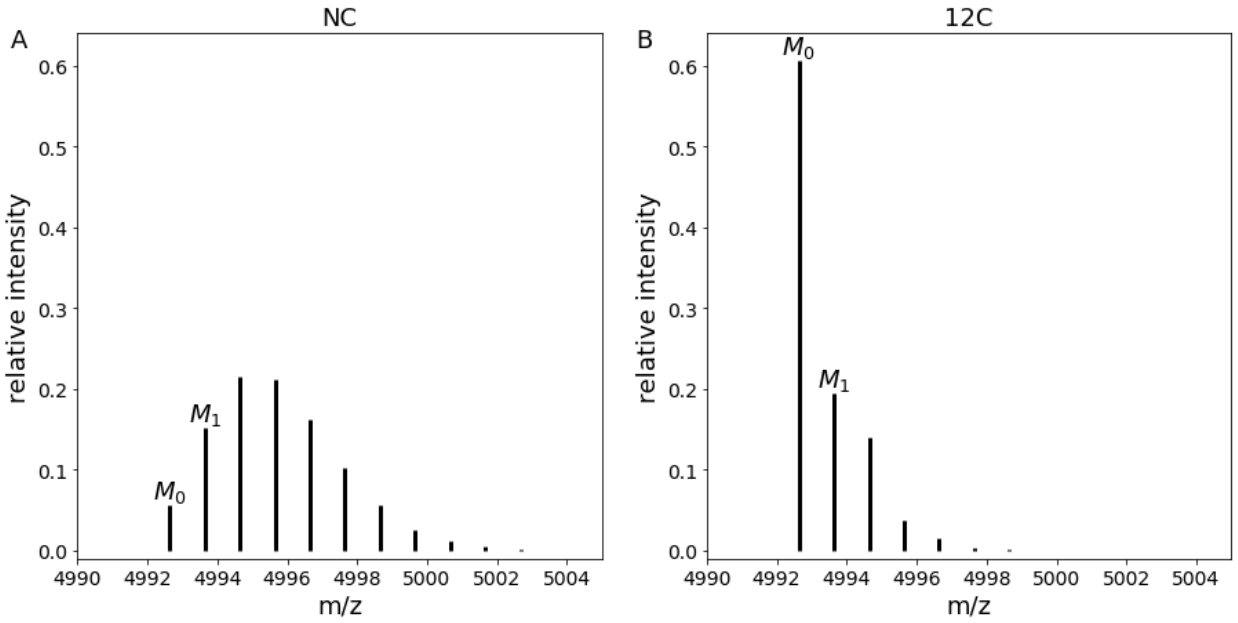
Simple Light Isotopic Metabolic Labeling (SLIM-labeling)[3] is a method developed by Léger et al. that allows **quantification of proteins via C[12] enrichment**. **Cells are provided** with a growth media containing **glucose enriched with C[12] at 99.99 % as the only carbon source**, the glucose is then assimilated by the cell to synthetize proteins which carbons have a C[12] abundance of 99.99 % instead of 98.93 %.



Usage of SLIM-labeling by combining two experimental conditions

SLIM-labeling allows the combination of proteins obtained in two different experimental conditions (Natural Carbon/Normal Condition **NC** and 99.99 % C[12] enrichment **12C**) in a single mass spectrometry run. By **comparing experimental and theoretical intensities** of adjacent isotopologues (in our case *M0* and *M1*), we are able to **get the ratio of proteins** between both conditions.

Another advantage of SLIM-labeling is that it increases the intensity of the first isotopologue, making it easier to detect.



Comparison of mass spectra in NC and 12C conditions for peptide with sequence "VGEVFINYIQRQNELFQGK-LAYLIIDTCLSIVRPNDSKPLDNR"

2.1.4 Strategy to take into account auxotrophies

A limitation of SLIM-labeling is that some organisms can have **auxotrophies to amino acids**, hence they cannot synthetize those for protein production. In this case, they **need to be provided with essential amino acids**. The problem is that 99.99 % C[12] enriched amino acids are not available nor are produced (as of 2019) thus **those essential amino acids will keep natural carbon abundance** (with 98.93 % C[12] and 1.07 % C[13]). Therefore, the formulas shown above become incorrect for SLIM-labeling (as abundances vary depending on wether the amino acid is labelled or not).

Seq-to-first-iso implements an algorithm that takes into account labelled and unlabelled amino acids for *M0* and *M1* computation.

To do so, we defined ***X*** as a virtual chemical element with **2 isotopes with abundance of natural carbon**. *X* can then be **substituted to the carbon of unlabelled amino acids** to compute correct isotopologue intensities.

New formulas were developed to take this new element into account:

$$M_0 = a(\mathrm{C}[12])^{n(\mathrm{C})} \times a(\mathrm{H}[1])^{n(\mathrm{H})} \times a(\mathrm{O}[16])^{n(\mathrm{O})} \times a(\mathrm{N}[14])^{n(\mathrm{N})} \times a(\mathrm{S}[32])^{n(\mathrm{S})} \times a(\mathrm{X}[12])^{n(\mathrm{X})}$$

and

$$\begin{aligned}
M_1 = {} & n(\mathrm{C}) \times a(\mathrm{C}[12])^{n(\mathrm{C})-1} \times a(\mathrm{C}[13]) \times a(\mathrm{H}[1])^{n(\mathrm{H})} \times a(\mathrm{O}[16])^{n(\mathrm{O})} \times a(\mathrm{N}[14])^{n(\mathrm{N})} \times a(\mathrm{S}[32])^{n(\mathrm{S})} \times a(\mathrm{X}[12])^{n(\mathrm{X})} \\
& + n(\mathrm{H}) \times a(\mathrm{C}[12])^{n(\mathrm{C})} \times a(\mathrm{H}[1])^{n(\mathrm{H})-1} \times a(\mathrm{H}[2]) \times a(\mathrm{O}[16])^{n(\mathrm{O})} \times a(\mathrm{N}[14])^{n(\mathrm{N})} \times a(\mathrm{S}[32])^{n(\mathrm{S})} \times a(\mathrm{X}[12])^{n(\mathrm{X})} \\
& + n(\mathrm{O}) \times a(\mathrm{C}[12])^{n(\mathrm{C})} \times a(\mathrm{H}[1])^{n(\mathrm{H})} \times a(\mathrm{O}[16])^{n(\mathrm{O})-1} \times a(\mathrm{O}[17]) \times a(\mathrm{N}[14])^{n(\mathrm{N})} \times a(\mathrm{S}[32])^{n(\mathrm{S})} \times a(\mathrm{X}[12])^{n(\mathrm{X})} \\
& + n(\mathrm{N}) \times a(\mathrm{C}[12])^{n(\mathrm{C})} \times a(\mathrm{H}[1])^{n(\mathrm{H})} \times a(\mathrm{O}[16])^{n(\mathrm{O})} \times a(\mathrm{N}[14])^{n(\mathrm{N})-1} \times a(\mathrm{N}[15]) \times a(\mathrm{S}[32])^{n(\mathrm{S})} \times a(\mathrm{X}[12])^{n(\mathrm{X})} \\
& + n(\mathrm{S}) \times a(\mathrm{C}[12])^{n(\mathrm{C})} \times a(\mathrm{H}[1])^{n(\mathrm{H})} \times a(\mathrm{O}[16])^{n(\mathrm{O})} \times a(\mathrm{N}[14])^{n(\mathrm{N})} \times a(\mathrm{S}[32])^{n(\mathrm{S})-1} \times a(\mathrm{S}[33]) \times a(\mathrm{X}[12])^{n(\mathrm{X})} \\
& + n(\mathrm{X}) \times a(\mathrm{C}[12])^{n(\mathrm{C})} \times a(\mathrm{H}[1])^{n(\mathrm{H})} \times a(\mathrm{O}[16])^{n(\mathrm{O})} \times a(\mathrm{N}[14])^{n(\mathrm{N})} \times a(\mathrm{S}[32])^{n(\mathrm{S})} \times a(\mathrm{X}[12])^{n(\mathrm{X}-1)} \times a(\mathrm{X}[13])
\end{aligned}$$

2.1.5 References

[1]: Alves, Gelio et al. "Molecular Isotopic Distribution Analysis (MIDAs) with adjustable mass accuracy." *Journal of the American Society for Mass Spectrometry* vol. 25,1 (2014): 57-70. doi:10.1007/s13361-013-0733-7

[2]: Wang, Benlian et al. "Isotopologue distributions of peptide product ions by tandem mass spectrometry: quantitation of low levels of deuterium incorporation." *Analytical biochemistry* vol. 367,1 (2007): 40-8. doi:10.1016/j.ab.2007.03.036

[3]: Léger, Thibaut et al. "A Simple Light Isotope Metabolic Labeling (SLIM-labeling) Strategy: A Powerful Tool to Address the Dynamics of Proteome Variations In Vivo." *Molecular & cellular proteomics : MCP* vol. 16,11 (2017): 2017-2031. doi:10.1074/mcp.M117.066936

2.2 KNIME configuration

You can install and use seq-to-first-iso with the KNIME Analytics Platform to process data from SLIM-labeling.

Requirements:

- KNIME
- conda
- a little bit of Python knowledge

This guide is for KNIME 3.7.2 only.

2.2.1 Install (Ana)conda

Install the latest version of Anaconda with Python 3.x.

2.2.2 Install KNIME

Install KNIME 3.7.2. You can download this version here.

2.2.3 Set up Python for KNIME

You need to install and configure a Python extension in KNIME.

This guide is adapted from the 3.7 Python installation guide from KNIME.

Set up the conda environment

On Windows, if you want to use conda with the default command-line interface *CMD*, you need to do some configuration, else use *Anaconda prompt* (or any other interface that recognizes conda) bundled with conda.

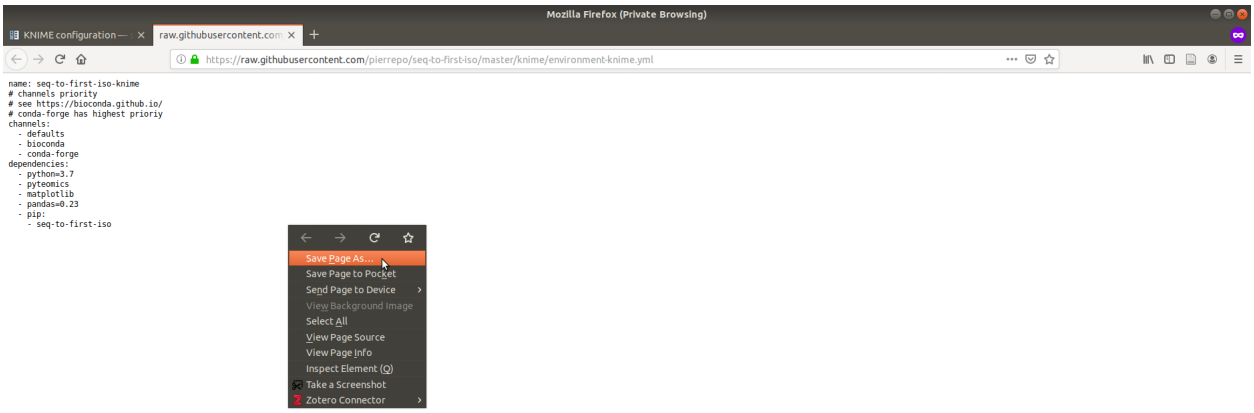
If conda was not added to the PATH environment variable during the installation, you have to configure your shell to use the `conda` commands:

```
setx PATH=%PATH%;C:<PATH_WHERE_YOU_INSTALLED_CONDA>\Scripts
```

By default `<PATH_WHERE_YOU_INSTALLED_CONDA>` is `C:\Users\<Username>\<CONDA_INSTALLATION>` where `<Username>` is the Windows username and `<CONDA_INSTALLATION>` is the name of the conda installer (e.g: "Anaconda3").

Create a conda environment

Download the environment file [*knime/environment-knime.yml*](https://raw.githubusercontent.com/pierrepo/seq-to-first-iso/master/knime/environment-knime.yml (*Right click → Save as . . .*).



In the directory where *environment-knime.yml* has been downloaded, open a shell/*Anaconda prompt* and use:

```
conda env create -f environment-knime.yml
```

The command

```
conda env list
```

should now list `seq-to-first-iso-knime` in available environments.

Create a start script

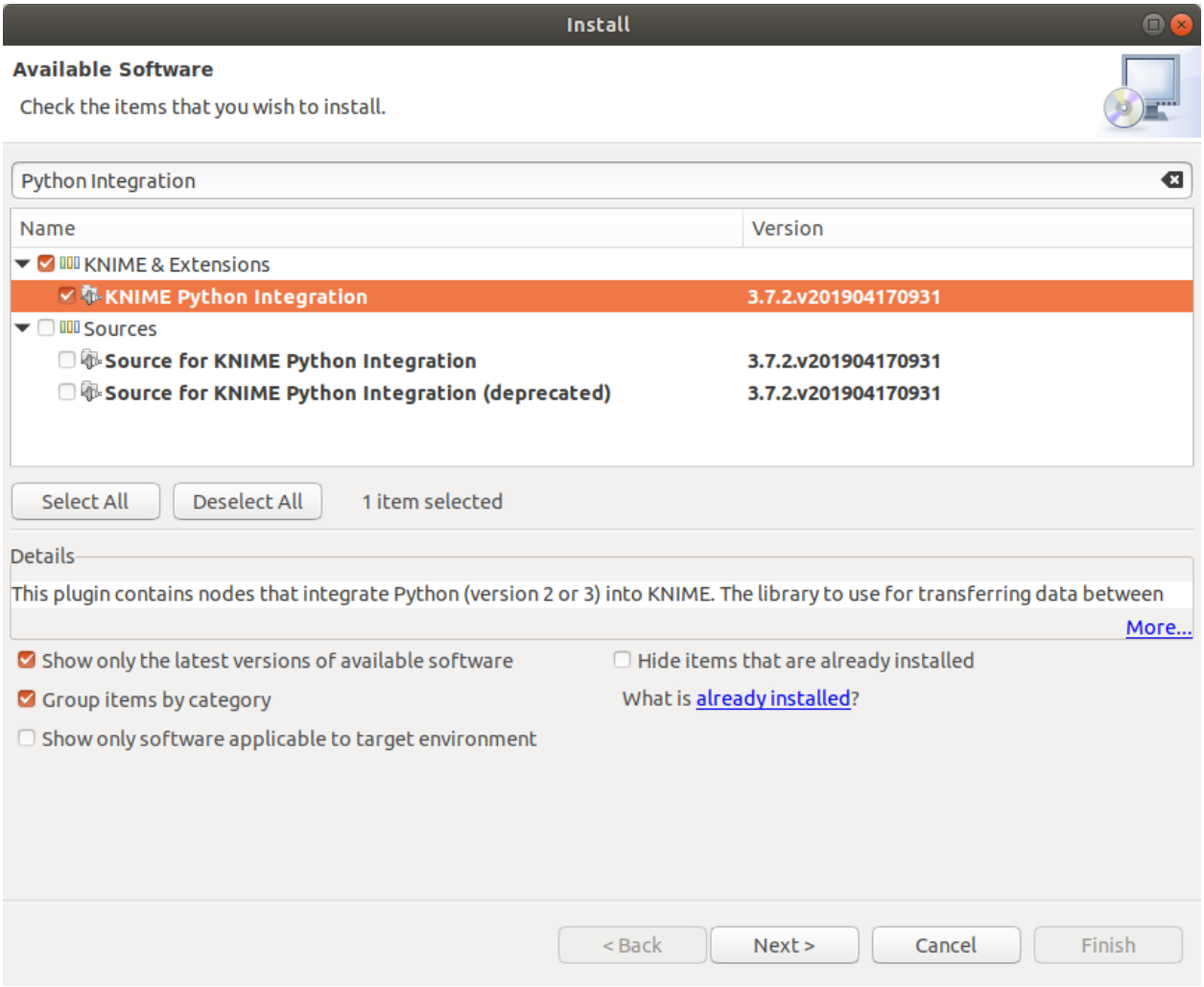
Create a small script to start the conda environment by using the templates defined by KNIME. In our case `<ENVIRONMENT_NAME>` is `seq-to-first-iso-knime` while `<PATH_WHERE_YOU_INSTALLED_ANACONDA>` depend on the user's conda configuration and operating sytem.

For windows, here is an example of such a script (`conda_env.bat`):

```
@REM Adapt the folder in the PATH to your system
@SET PATH=%USERPROFILE%\Miniconda3\Scripts;%PATH%
@CALL activate seq-to-first-iso-knime || ECHO Activating python environment failed
@python %*
```

Configure the Python extension

In the KNIME interface, go to *File → Install KNIME Extensions*, then search for *Python Integration* to find the KNIME Python Integration.



Select and install this extension.

Then, go ot the configuration menu *File → Preferences → KNIME → Python*. In the Python 3 subsection, paste the absolute path to your start script.

e.g: Windows path `C:\Documents\<script_name>`

Select, Python 3 as default, then Apply and close.

If everything went alright, you should now be able to use Python script nodes with KNIME.

2.2.4 Use seq-to-first-iso with KNIME

Warning: this tutorial assumes you use the latest version of seq-to-first-iso.

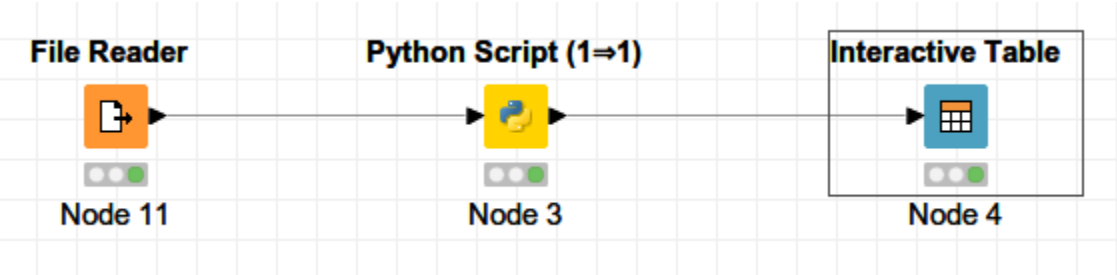
The following steps are made of examples, adapt them to your needs.

We assumes that:

- `input_table_1` contains (if any) unlabeled amino acids,
- `input_table_2` contains peptide sequences (column `pep_seq`) and charges (column `pep_charge`).

Create a Python scripting node by going in the Node Repository then *Scripting → Python → Python Script (11)*. The node will receive a table as an input.

```
import pandas as pd
import seq_to_first_iso as stfi

print(stfi.__version__)

# List of unlabelled amino acids from multiple selection node.
# It is the first node used in this case.
unlabelled_aa = list(input_table_1.iloc[:, 0])
print("Amino acids unlabelled:", unlabelled_aa)

# Take the content of the parsed file.
# The name output_table will inform KNIME that
# the variable is an output table.
output_table = input_table_2.copy()

# Extract relevant columns
df = pd.DataFrame()
df["sequence"] = output_table["pep_seq"]
df["charge"] = output_table["pep_charge"]

# Get M0/M1 intensities
df_peptides = stfi.compute_intensities(df, unlabelled_aa)

# Export final results
output_table = stfi.export_to_knime(outpu_table, df_peptides)
```



2.2.5 Update seq-to-first-iso in KNIME

Update conda environnement

Open *Anaconda prompt*, then enter the following commands:

```
conda activate seq-to-first-iso-knime
conda install seq-to-first-iso
```

Update Python script

Open the Python Script node and check the content of the script is similar to the above code.

CHAPTER

THREE

TUTORIAL

3.1 API of seq-to-first-iso

seq-to-first-iso computes the first two isotopologue intentities (M0 and M1) from peptide sequences with natural carbon and with 99.99% 12C enriched carbon.

The program can take into account unlabelled amino acids to simulate auxotrophies to amino acids.

seq-to-first-iso is available as a Python module.

[1]:
```
from pathlib import Path
from pprint import pprint

from pkg_resources import get_distribution  # Comes with setuptools.
import pandas as pd
from pyteomics import mass

import seq_to_first_iso as stfi
```

[2]:
```
try:
    print(f"pyteomics version: {get_distribution('pyteomics').version}")
except:
    print("pyteomics version not found")

print(f"pandas version: {pd.__version__}\n"
      f"seq-to-first-iso version: {stfi.__version__}"
     )
```

```
pyteomics version: 4.1.2
pandas version: 0.25.1
seq-to-first-iso version: 0.5.1
```

3.1.1 Abundances defined in seq-to-first-iso

[3]:
```
pprint(stfi.NATURAL_ABUNDANCE)
```

```
{'C[12]': 0.9893,
 'C[13]': 0.0107,
 'H[1]': 0.999885,
 'H[2]': 0.000115,
 'N[14]': 0.99632,
 'N[15]': 0.00368,
 'O[16]': 0.99757,
```

(continues on next page)

(continued from previous page)

```
 'O[17]': 0.00038,
 'O[18]': 0.00205,
 'S[32]': 0.9493,
 'S[33]': 0.0076,
 'S[34]': 0.0429,
 'X[12]': 0.9893,
 'X[13]': 0.0107}
```

[4]:
```
pprint(stfi.C12_ABUNDANCE)
```

```
{'C[12]': 0.9999,
 'C[13]': 9.999999999998899e-05,
 'H[1]': 0.999885,
 'H[2]': 0.000115,
 'N[14]': 0.99632,
 'N[15]': 0.00368,
 'O[16]': 0.99757,
 'O[17]': 0.00038,
 'O[18]': 0.00205,
 'S[32]': 0.9493,
 'S[33]': 0.0076,
 'S[34]': 0.0429,
 'X[12]': 0.9893,
 'X[13]': 0.0107}
```

`NATURAL_ABUNDANCE` and `C12_ABUNDANCE` are dictionaries with abundances of common isotopes of organic elements.

`C12_ABUNDANCE` has a 12C abundance of 99.99 %, hence 13C abundance is 0.01 %.

Element X is a **virtual element** created to replace the carbon of unlabelled amino acids, it has the **same isotopic abundances as natural carbon**.

3.1.2 Separate sequences according to unlabelled amino acids

[5]:
```
help(stfi.separate_labelled)
```

```
Help on function separate_labelled in module seq_to_first_iso.seq_to_first_iso:

separate_labelled(sequence, unlabelled_aa)
    Get the sequence of unlabelled amino acids from a sequence.

    Parameters
    ----------
    sequence : str
        String of amino acids.
    unlabelled_aa : container object
        Container (list, string...) of unlabelled amino acids.

    Returns
    -------
    tuple(str, str)
        | The sequences as a tuple of string with:
        |    - the sequence without the unlabelled amino acids
```

(continues on next page)

(continued from previous page)

```
        |    - the unlabelled amino acids in the sequence
```

[6]:
```
# Separate sequence "YAQEISRAR" with amino acids A and R unlabelled.
peptide_seq = "YAQEISRAR"
unlabelled_amino_acids = ["A", "R"]

labelled_sequence, unlabelled_sequence = stfi.separate_labelled(peptide_seq,␣
↪unlabelled_aa=unlabelled_amino_acids)

print(
    f"Original sequence: {peptide_seq}\n"
    f"Unlabelled amino acids: {unlabelled_amino_acids}\n"
    f"Sequence with labelled carbon: {labelled_sequence}\n"
    f"Sequence with unlabelled carbon: {unlabelled_sequence}")
```

```
Original sequence: YAQEISRAR
Unlabelled amino acids: ['A', 'R']
Sequence with labelled carbon: YQEIS
Sequence with unlabelled carbon: ARAR
```

3.1.3 Obtain a composition with element X

[7]:
```
# Get the chemical formula with unlabelled carbon as element X.
labelled_formula = mass.Composition(labelled_sequence)
unlabelled_formula = stfi.convert_atom_C_to_X(mass.Composition(parsed_
↪sequence=unlabelled_sequence))
peptide_formula = unlabelled_formula + labelled_formula
print(f"Composition of labelled amino acids: {labelled_formula}")
print(f"Composition of unlabelled amino acids (X is C): {unlabelled_formula}")
print(f"Composition of {peptide_seq} with {unlabelled_amino_acids} unlabelled:\n
↪{peptide_formula}")
```

```
Composition of labelled amino acids: Composition({'H': 42, 'C': 28, 'O': 11, 'N': 6})
Composition of unlabelled amino acids (X is C): Composition({'H': 34, 'O': 4, 'N': 10,
↪ 'X': 18})
Composition of YAQEISRAR with ['A', 'R'] unlabelled:
Composition({'H': 76, 'O': 15, 'N': 16, 'X': 18, 'C': 28})
```

3.1.4 Compute isotopologue intensity

[8]:
```
help(stfi.compute_M0_nl)
print("-" * 79)
help(stfi.compute_M1_nl)
```

```
Help on function compute_M0_nl in module seq_to_first_iso.seq_to_first_iso:

compute_M0_nl(formula, abundance)
    Compute intensity of the first isotopologue M0.

    Handle element X with specific abundance.

    Parameters
```

(continues on next page)

(continued from previous page)

```
    ----------
    formula : pyteomics.mass.Composition
        Chemical formula, as a dict of the number of atoms for each element:
        {element_name: number_of_atoms, ...}.
    abundance : dict
        Dictionary of abundances of isotopes:
        {"element_name[isotope_number]": relative abundance, ..}.

    Returns
    -------
    float
        Value of M0.

    Notes
    -----
    X represents C with default isotopic abundance.

------------------------------------------------------------------------------
Help on function compute_M1_nl in module seq_to_first_iso.seq_to_first_iso:

compute_M1_nl(formula, abundance)
    Compute intensity of the second isotopologue M1.

    Handle element X with specific abundance.

    Parameters
    ----------
    formula : pyteomics.mass.Composition
        Chemical formula, as a dict of the number of atoms for each element:
        {element_name: number_of_atoms, ...}.
    abundance : dict
        Dictionary of abundances of isotopes:
        {"element_name[isotope_number]": relative abundance, ..}.

    Returns
    -------
    float
        Value of M1.

    Notes
    -----
    X represents C with default isotopic abundance.
```

[9]:
```
# Compute M0 with natural carbon.
first_isotopologue = stfi.compute_M0_nl(peptide_formula, stfi.NATURAL_ABUNDANCE)
print(f"M0 in normal (98.93% 12C) condition: {first_isotopologue}")

first_isotopologue = stfi.compute_M0_nl(peptide_formula, stfi.C12_ABUNDANCE)
print(f"M0 in    12C (99.99% 12C) condition: {first_isotopologue}")
```

```
M0 in normal (98.93% 12C) condition: 0.5493191520383802
M0 in    12C (99.99% 12C) condition: 0.7403283857401063
```

[10]:
```
# Compute M1 with natural carbon.
second_isotopologue = stfi.compute_M1_nl(peptide_formula, stfi.NATURAL_ABUNDANCE)
```

(continues on next page)

(continued from previous page)

```
print(f"M1 in normal (98.93% 12C) condition: {second_isotopologue}")

second_isotopologue = stfi.compute_M1_nl(peptide_formula, stfi.C12_ABUNDANCE)
print(f"M1 in    12C (99.99% 12C) condition: {second_isotopologue}")
```

```
M1 in normal (98.93% 12C) condition: 0.313702912736476
M1 in    12C (99.99% 12C) condition: 0.200655465179031
```

3.1.5 Get the composition of a list of Post-translational modifications (PTMs)

[11]:
```
help(stfi.get_mods_composition)
```

```
Help on function get_mods_composition in module seq_to_first_iso.seq_to_first_iso:

get_mods_composition(modifications)
    Return the composition of a list of modifications.

    Parameters
    ----------
    modifications : list of str
        List of modifications string (corresponding to Unimod titles).

    Returns
    -------
    pyteomics.mass.Composition
        The total composition change.
```

[12]:
```
# Modifications must be strict Unimod entries title.
modification_list = ["Acetyl", "Phospho", "phospho"]  # phospho does not correspond␣
↪to a real PTM name, it will be ignored
total_composition = stfi.get_mods_composition(modification_list)
print(f"Total composition for {modification_list} is {total_composition}")
```

```
[2019-12-05, 13:55:32] WARNING : Unimod entry not found for : phospho
```

```
Total composition for ['Acetyl', 'Phospho', 'phospho'] is Composition({'H': 3, 'C': 2,
↪ 'O': 4, 'P': 1})
```

3.1.6 Get human-readable chemical formula

[13]:
```
help(stfi.formula_to_str)
```

```
Help on function formula_to_str in module seq_to_first_iso.seq_to_first_iso:

formula_to_str(composition)
    Return formula from Composition as a string.

    Parameters
    ----------
    composition : pyteomics.mass.Composition
        Chemical formula.

    Returns
```

(continues on next page)

(continued from previous page)

```
    -------
    str
        Human-readable string of the formula.

    Warnings
    --------
    If the composition has elements not in USED_ELEMS, they will not
    be added to the output.
```

[14]:
```
# This is the function used to get the formulas in the output.
formula_str = stfi.formula_to_str(total_composition)
print(f"{total_composition} becomes {formula_str}")
```

```
Composition({'H': 3, 'C': 2, 'O': 4, 'P': 1}) becomes C2H3O4P1
```

[15]:
```
# !!! Warning: if the Composition has elements not in "CHONPSX", they will not be in␣
↪the final string.
bad_composition = mass.Composition("U")
formula_str = stfi.formula_to_str(bad_composition)
print(f"Compostion with unsupported element {bad_composition} becomes {formula_str}")
```

```
Compostion with unsupported element Composition({'H': 7, 'C': 3, 'O': 2, 'N': 1, 'Se':
↪ 1}) becomes C3H7O2N1
```

Here, "non-CHONPSX" element **Se (Selenium) is ignored**!

3.1.7 Parse a file with peptide sequences and charges

`seq-to-first-iso` reads tsv files with at least a sequence and a charge columns.

The parser will ignore lines where sequences have incorrect characters (not in `ACDEFGHIKLMNPQRSTVWY`) unless it corresponds to XTandem's PTMs notation.

[16]:
```
df_raw = stfi.parse_input_file("peptides.tsv")
df_filtered = stfi.filter_input_dataframe(df_raw, "pep_sequence", "pep_charge")
print(df_filtered)
```

```
[2019-12-05, 13:55:32] INFO    : Read peptides.tsv
[2019-12-05, 13:55:32] INFO    : Found 11 lines and 3 columns
                                              sequence  charge
0                                               YAQEISR       2
1          VLLIDLRIPQR(Phospho)SAINHIVAPNLVNVDPNLLWDK       3
2                             QRTTFFVLGINTVNYPDIYEHILER       2
3                                AELFL(Glutathione)LNR       1
4   .(Acetyl)VGEVFINYIQRQNELFQGKLAYLII(Oxidation)D...       4
5                          YKTMNTFDPD(Heme)EKFEWFQVWQAVK       2
6          HKSASSPAV(Pro-&gt;Val)NADTDIQDSSTPSTSPSGRR       2
7                                                FHNK       1
8                                 .(Glutathione)MDLEIK       3
9                                       LANEKPEDVFER       2
10  .(Acetyl)SDTPLR(Oxidation)D(Acetyl)EDG(Acetyl)...       3
```

[17]:
```
df_final = stfi.compute_intensities(df_filtered, unlabelled_aa=["A", "R"])
df_final
```

```
[2019-12-05, 13:55:33] INFO    : Reading sequences.
[2019-12-05, 13:55:33] INFO    : Computing composition and formula.
[2019-12-05, 13:55:33] WARNING : Fe in (Heme) is not supported in the computation of␣
↪M0 and M1
[2019-12-05, 13:55:33] INFO    : Computing neutral mass
[2019-12-05, 13:55:33] INFO    : Computing M0 and M1
```

[17]:

	stfi_sequence	stfi_charge	\
0	YAQEISR	2	
1	VLLIDLRIPQR(Phospho)SAINHIVAPNLVNVDPNLLWDK	3	
2	QRTTFFVLGINTVNYPDIYEHILER	2	
3	AELFL(Glutathione)LNR	1	
4	.(Acetyl)VGEVFINYIQRQNELFQGKLAYLII(Oxidation)D...	4	
5	YKTMNTFDPD(Heme)EKFEWFQVWQAVK	2	
6	HKSASSPAV(Pro->Val)NADTDIQDSSTPSTSPSGRR	2	
7	FHNK	1	
8	.(Glutathione)MDLEIK	3	
9	LANEKPEDVFER	2	
10	.(Acetyl)SDTPLR(Oxidation)D(Acetyl)EDG(Acetyl)...	3	

	stfi_sequence_clean	\
0	YAQEISR	
1	VLLIDLRIPQR(Phospho)SAINHIVAPNLVNVDPNLLWDK	
2	QRTTFFVLGINTVNYPDIYEHILER	
3	AELFL(Glutathione)LNR	
4	.(Acetyl)VGEVFINYIQRQNELFQGKLAYLII(Oxidation)D...	
5	YKTMNTFDPD(Heme)EKFEWFQVWQAVK	
6	HKSASSPAV(Pro->Val)NADTDIQDSSTPSTSPSGRR	
7	FHNK	
8	.(Glutathione)MDLEIK	
9	LANEKPEDVFER	
10	.(Acetyl)SDTPLR(Oxidation)D(Acetyl)EDG(Acetyl)...	

	stfi_modification	\
0	[]	
1	[Phospho]	
2	[]	
3	[Glutathione]	
4	[Acetyl, Oxidation]	
5	[Heme]	
6	[Pro->Val]	
7	[]	
8	[Glutathione]	
9	[]	
10	[Acetyl, Oxidation, Acetyl, Acetyl]	

	stfi_sequence_without_mod	\
0	YAQEISR	
1	VLLIDLRIPQRSAINHIVAPNLVNVDPNLLWDK	
2	QRTTFFVLGINTVNYPDIYEHILER	
3	AELFLLNR	
4	VGEVFINYIQRQNELFQGKLAYLIIDTCLSIVRPNDSKPLDNR	
5	YKTMNTFDPDEKFEWFQVWQAVK	
6	HKSASSPAVNADTDIQDSSTPSTSPSGRR	
7	FHNK	
8	MDLEIK	
9	LANEKPEDVFER	
10	SDTPLRDEDGLDFWETLRSLATTNPNPPVEK	

(continues on next page)

(continued from previous page)

```
                     stfi_sequence_to_process stfi_log  \
0                                       YAQEISR
1              VLLIDLRIPQRSAINHIVAPNLVNVDPNLLWDK
2                     QRTTFFVLGINTVNYPDIYEHILER
3                                     AELFLLNR
4   VGEVFINYIQRQNELFQGKLAYLIIDTCLSIVRPNDSKPLDNR
5                        YKTMNTFDPDEKFEWFQVWQAVK
6                  HKSASSPAVNADTDIQDSSTPSTSPSGRR
7                                       FHNK
8                                     MDLEIK
9                                   LANEKPEDVFER
10              SDTPLRDEDGLDFWETLRSLATTNPNPPVEK

                     stfi_sequence_labelled stfi_sequence_unlabelled  \
0                                       YQEIS                       AR
1              VLLIDLIPQSINHIVPNLVNVDPNLLWDK                     RRAA
2                    QTTFFVLGINTVNYPDIYEHILE                       RR
3                                     ELFLLN                       AR
4   VGEVFINYIQQNELFQGKLYLIIDTCLSIVPNDSKPLDN                     RARR
5                     YKTMNTFDPDEKFEWFQVWQVK                        A
6                   HKSSSPVNDTDIQDSSTPSTSPSG                    AAARR
7                                       FHNK
8                                     MDLEIK
9                                   LNEKPEDVFE                       AR
10              SDTPLDEDGLDFWETLSLTTNPNPPVEK                      RRA

                           stfi_composition_mod  ...  \
0                                            {}  ...
1                      {'H': 1, 'O': 3, 'P': 1}  ...
2                                            {}  ...
3    {'H': 15, 'C': 10, 'N': 3, 'O': 6, 'S': 1}  ...
4                      {'H': 2, 'C': 2, 'O': 2}  ...
5   {'H': 32, 'C': 34, 'N': 4, 'O': 4, 'Fe': 1}  ...
6                                     {'H': 2}  ...
7                                            {}  ...
8    {'H': 15, 'C': 10, 'N': 3, 'O': 6, 'S': 1}  ...
9                                            {}  ...
10                     {'H': 6, 'C': 6, 'O': 4}  ...

                     stfi_composition_peptide_neutral  \
0                 {'H': 59, 'C': 37, 'O': 13, 'N': 11}
1      {'H': 285, 'C': 172, 'O': 49, 'N': 48, 'P': 1}
2               {'H': 212, 'C': 140, 'O': 40, 'N': 36}
3        {'H': 89, 'C': 55, 'O': 18, 'N': 15, 'S': 1}
4      {'H': 361, 'C': 226, 'O': 68, 'N': 61, 'S': 1}
5   {'H': 225, 'C': 173, 'O': 42, 'N': 35, 'S': 1,...
6               {'H': 196, 'C': 118, 'N': 40, 'O': 49}
7                   {'H': 36, 'C': 25, 'O': 6, 'N': 8}
8        {'H': 72, 'C': 42, 'S': 2, 'O': 17, 'N': 10}
9                 {'H': 99, 'C': 63, 'O': 22, 'N': 17}
10              {'H': 243, 'C': 159, 'O': 58, 'N': 41}

                   stfi_composition_peptide_with_charge  \
0                  {'H': 61, 'C': 37, 'O': 13, 'N': 11}
1      {'H': 288, 'C': 172, 'O': 49, 'N': 48, 'P': 1}
2               {'H': 214, 'C': 140, 'O': 40, 'N': 36}
```

(continues on next page)

(continued from previous page)

```
3        {'H': 90, 'C': 55, 'O': 18, 'N': 15, 'S': 1}
4      {'H': 365, 'C': 226, 'O': 68, 'N': 61, 'S': 1}
5   {'H': 227, 'C': 173, 'O': 42, 'N': 35, 'S': 1,...
6              {'H': 198, 'C': 118, 'N': 40, 'O': 49}
7                  {'H': 37, 'C': 25, 'O': 6, 'N': 8}
8        {'H': 75, 'C': 42, 'S': 2, 'O': 17, 'N': 10}
9               {'H': 101, 'C': 63, 'O': 22, 'N': 17}
10             {'H': 246, 'C': 159, 'O': 58, 'N': 41}

               stfi_composition_peptide_with_charge_X      stfi_formula  \
0        {'H': 61, 'C': 28, 'O': 13, 'N': 11, 'X': 9}      C37H61O13N11
1   {'H': 288, 'C': 154, 'O': 49, 'N': 48, 'X': 18...  C172H288O49N48P1
2     {'H': 214, 'C': 128, 'O': 40, 'N': 36, 'X': 12}    C140H214O40N36
3   {'H': 90, 'C': 46, 'O': 18, 'N': 15, 'X': 9, '...    C55H90O18N15S1
4   {'H': 365, 'C': 205, 'O': 68, 'N': 61, 'S': 1,...  C226H365O68N61S1
5   {'H': 227, 'C': 170, 'O': 42, 'N': 35, 'S': 1,...  C173H227O42N35S1
6      {'H': 198, 'C': 97, 'N': 40, 'O': 49, 'X': 21}    C118H198O49N40
7                  {'H': 37, 'C': 25, 'O': 6, 'N': 8}        C25H37O6N8
8        {'H': 75, 'C': 42, 'S': 2, 'O': 17, 'N': 10}    C42H75O17N10S2
9       {'H': 101, 'C': 54, 'O': 22, 'N': 17, 'X': 9}     C63H101O22N17
10    {'H': 246, 'C': 144, 'O': 58, 'N': 41, 'X': 15}    C159H246O58N41

         stfi_formula_X stfi_neutral_mass stfi_M0_NC  stfi_M1_NC  stfi_M0_12C  \
0        C28H61O13N11X9        865.429381   0.620499    0.280949     0.836258
1   C154H288O49N48P1X18       3838.102264   0.113085    0.236277     0.583716
2     C128H214O40N36X12       3037.566156   0.171920    0.290033     0.672639
3      C46H90O18N15S1X9       1279.623072   0.470882    0.318073     0.768822
4   C205H365O68N61S1X21       5049.638616   0.054173    0.148735     0.481545
5    C170H227O42N35S1X3       3552.561645   0.114128    0.234021     0.698631
6      C97H198O49N40X21       2957.407483   0.210376    0.308292     0.591515
7            C25H37O6N8        544.275781   0.728121    0.223157     0.950424
8        C42H75O17N10S2       1052.451833   0.525852    0.274658     0.822740
9       C54H101O22N17X9       1445.715058   0.446843    0.341468     0.794506
10    C144H246O58N41X15       3654.732565   0.131200    0.252105     0.608763

    stfi_M1_12C
0      0.127729
1      0.256348
2      0.212157
3      0.140356
4      0.264287
5      0.159873
6      0.251993
7      0.036677
8      0.059443
9      0.147405
10     0.230393

[11 rows x 22 columns]
```

[18]:
```
# Most interesting columns are the following
df_final[["stfi_sequence", "stfi_charge", "stfi_M0_NC", "stfi_M1_NC", "stfi_M0_12C",
↪"stfi_M1_12C"]]
```

[18]:
```
                                    stfi_sequence  stfi_charge  \
0                                          YAQEISR            2
1          VLLIDLRIPQR(Phospho)SAINHIVAPNLVNVDPNLLWDK            3
```

(continues on next page)

(continued from previous page)

```
2                            QRTTFFVLGINTVNYPDIYEHILER            2
3                               AELFL(Glutathione)LNR             1
4   .(Acetyl)VGEVFINYIQRQNELFQGKLAYLII(Oxidation)D...             4
5                         YKTMNTFDPD(Heme)EKFEWFQVWQAVK            2
6          HKSASSPAV(Pro-&gt;Val)NADTDIQDSSTPSTSPSGRR              2
7                                                FHNK             1
8                                .(Glutathione)MDLEIK             3
9                                     LANEKPEDVFER                2
10  .(Acetyl)SDTPLR(Oxidation)D(Acetyl)EDG(Acetyl)...             3

    stfi_M0_NC  stfi_M1_NC  stfi_M0_12C  stfi_M1_12C
0     0.620499    0.280949     0.836258     0.127729
1     0.113085    0.236277     0.583716     0.256348
2     0.171920    0.290033     0.672639     0.212157
3     0.470882    0.318073     0.768822     0.140356
4     0.054173    0.148735     0.481545     0.264287
5     0.114128    0.234021     0.698631     0.159873
6     0.210376    0.308292     0.591515     0.251993
7     0.728121    0.223157     0.950424     0.036677
8     0.525852    0.274658     0.822740     0.059443
9     0.446843    0.341468     0.794506     0.147405
10    0.131200    0.252105     0.608763     0.230393
```

3.1.8 Concatenation of results with input data

[19]:
```
input_file_name = "peptides.tsv"
output_file_name = Path(input_file_name).stem + "_stfi.tsv"

column_of_interest = ["stfi_neutral_mass",
                      "stfi_formula", "stfi_formula_X",
                      "stfi_M0_NC", "stfi_M1_NC",
                      "stfi_M0_12C", "stfi_M1_12C"]

# Read original file and append STFI data.
df_old = pd.read_csv(input_file_name, sep="\t")
df_new = pd.concat([df_old, df_final[column_of_interest]], axis=1)
df_new.to_csv(output_file_name, sep="\t", index=False)
```

[20]:
```
!head peptides_stfi.tsv
```

```
pep_name       pep_sequence   pep_charge     stfi_neutral_mass     stfi_formula
↪stfi_formula_X        stfi_M0_NC    stfi_M1_NC    stfi_M0_12C   stfi_M1_12C
seq1    YAQEISR 2       865.42938099921 C37H61O13N11    C28H61O13N11X9  0.
↪6204986747402674      0.28094895790268576     0.8362584492452608      0.
↪1277294394585608
seq2    VLLIDLRIPQR(Phospho)SAINHIVAPNLVNVDPNLLWDK      3       3838.1022643587894
↪  C172H288O49N48P1        C154H288O49N48P1X18     0.11308454311128492     0.
↪23627735941497488     0.5837157078086469      0.256348239423703
seq3    QRTTFFVLGINTVNYPDIYEHILER       2       3037.56615575404
↪C140H214O40N36  C128H214O40N36X12       0.17192000472677066     0.29003268314604863
↪    0.6726389393255647      0.2121565119028707
seq4    AELFL(Glutathione)LNR   1       1279.6230720783099      C55H90O18N15S1
↪C46H90O18N15S1X9        0.47088227298965996     0.31807282610880205     0.
↪7688224723128251      0.1403559631032404
seq5    .(Acetyl)VGEVFINYIQRQNELFQGKLAYLII(Oxidation)DTCLSIVRPNDSKPLDNR 4       5049.
↪63861600015        C226H365O68N61S1        C205H365O68N61S1X21     0.
↪05417296058666768     0.14873470210020426     0.48154538801515706     0.
↪26428662893114313
```

(continues on next page)

(continued from previous page)

```
seq6    YKTMNTFDPD(Heme)EKFEWFQVWQAVK   2       3552.56164490527
↪C173H227O42N35S1        C170H227O42N35S1X3      0.11412815567709074     0.
↪23402086836029898      0.6986310451922292      0.15987291091234185
seq7    HKSASSPAV(Pro-&gt;Val)NADTDIQDSSTPSTSPSGRR      2       2957.40748283616
↪ C118H198O49N40  C97H198O49N40X21        0.21037550761092094     0.
↪30829218128938995      0.5915145465128161      0.2519928490706656
seq8    FHNK    1       544.27578091028 C25H37O6N8      C25H37O6N8      0.
↪7281205110566825       0.2231565512772339      0.950423678912205       0.
↪036676880813002036
seq9    .(Glutathione)MDLEIK    3       1052.4518328895601      C42H75O17N10S2
↪C42H75O17N10S2  0.5258517009900313      0.27465762228958784     0.8227403058336873
↪    0.05944288050042882
```

[ ]:

3.2 Command line interface of seq-to-first-iso

seq-to-first-iso computes the first two isotopologue intentities (M0 and M1) from peptide sequences with natural carbon and with 99.99% 12C enriched carbon.

The program can take into account unlabelled amino acids to simulate auxotrophies to amino acids.

seq-to-first-iso is available as a Python module.

[1]:
```
import pandas as pd  # For output visualisation.
```

Note: the exclamation mark ``!`` is a magic command to run a Linux command within a Jupyter notebook. In a real Linux terminal, you don't need it.

[2]:
```
!seq-to-first-iso -v
```

```
seq-to-first-iso 1.0.0
```

[3]:
```
!seq-to-first-iso -h
```

```
usage: seq-to-first-iso [-h] [-o OUTPUT] [-u amino_a] [-v]
                        input_file_name sequence_col_name charge_col_name

Read a tsv file with sequences and charges and compute intensity of first
isotopologues

positional arguments:
  input_file_name       file to parse in .tsv format
  sequence_col_name     column name with sequences
  charge_col_name       column name with charges

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT, --output OUTPUT
                        name of output file
  -u amino_a, --unlabelled-aa amino_a
                        amino acids with default abundance
  -v, --version         show program's version number and exit
```

[4]:
```
# File used.
!cat peptides.tsv
```

```
pep_name       pep_sequence   pep_charge
seq1    YAQEISR 2
seq2    VLLIDLRIPQR(Phospho)SAINHIVAPNLVNVDPNLLWDK      3
seq3    QRTTFFVLGINTVNYPDIYEHILER       2
seq4    AELFL(Glutathione)LNR   1
seq5    .(Acetyl)VGEVFINYIQRQNELFQGKLAYLII(Oxidation)DTCLSIVRPNDSKPLDNR 4
seq6    YKTMNTFDPD(Heme)EKFEWFQVWQAVK   2
seq7    HKSASSPAV(Pro-&gt;Val)NADTDIQDSSTPSTSPSGRR     2
seq8    FHNK    1
seq9    .(Glutathione)MDLEIK    3
seq10   LANEKPEDVFER    2
seq11   .(Acetyl)SDTPLR(Oxidation)D(Acetyl)EDG(Acetyl)LDFWETLRSLATTNPNPPVEK     3
```

3.2.1 Minimal command

[5]:
```
!seq-to-first-iso peptides.tsv pep_sequence pep_charge
```

```
Namespace(charge_col_name='pep_charge', input_file_name=PosixPath('peptides.tsv'),␣
↪output=None, sequence_col_name='pep_sequence', unlabelled_aa=[])
[2019-12-05, 17:22:32] INFO    : Parsing file
[2019-12-05, 17:22:32] INFO    : Read peptides.tsv
[2019-12-05, 17:22:32] INFO    : Found 11 lines and 3 columns
[2019-12-05, 17:22:32] INFO    : Reading sequences.
[2019-12-05, 17:22:32] INFO    : Computing composition and formula.
[2019-12-05, 17:22:32] WARNING : Fe in (Heme) is not supported in the computation of␣
↪M0 and M1
[2019-12-05, 17:22:32] INFO    : Computing neutral mass
[2019-12-05, 17:22:32] INFO    : Computing M0 and M1
```

Running the command above will write a tab-separated-values file (`peptides_stfi.tsv`).

[6]:
```
# Read basic output file.
df = pd.read_csv("peptides_stfi.tsv", sep="\t")
df.head()
```

[6]:

	pep_name	pep_sequence	pep_charge	\
0	seq1	YAQEISR	2	
1	seq2	VLLIDLRIPQR(Phospho)SAINHIVAPNLVNVDPNLLWDK	3	
2	seq3	QRTTFFVLGINTVNYPDIYEHILER	2	
3	seq4	AELFL(Glutathione)LNR	1	
4	seq5	.(Acetyl)VGEVFINYIQRQNELFQGKLAYLII(Oxidation)D...	4	

	stfi_neutral_mass	stfi_formula	stfi_formula_X	stfi_M0_NC	\
0	865.429381	C37H61O13N11	C37H61O13N11	0.620499	
1	3838.102264	C172H288O49N48P1	C172H288O49N48P1	0.113085	
2	3037.566156	C140H214O40N36	C140H214O40N36	0.171920	
3	1279.623072	C55H90O18N15S1	C55H90O18N15S1	0.470882	
4	5049.638616	C226H365O68N61S1	C226H365O68N61S1	0.054173	

	stfi_M1_NC	stfi_M0_12C	stfi_M1_12C
0	0.280949	0.920444	0.051819
1	0.236277	0.707156	0.174161
2	0.290033	0.764407	0.142807

(continues on next page)

(continued from previous page)

```
3    0.318073     0.846220     0.072875
4    0.148735     0.602333     0.195036
```

3.2.2 Changing output name

You can also change the name of the output file

[7]:
```
!seq-to-first-iso peptides.tsv pep_sequence pep_charge -o seq_stfi
```

```
Namespace(charge_col_name='pep_charge', input_file_name=PosixPath('peptides.tsv'),␣
↪output='seq_stfi', sequence_col_name='pep_sequence', unlabelled_aa=[])
[2019-12-05, 17:22:34] INFO    : Parsing file
[2019-12-05, 17:22:34] INFO    : Read peptides.tsv
[2019-12-05, 17:22:34] INFO    : Found 11 lines and 3 columns
[2019-12-05, 17:22:34] INFO    : Reading sequences.
[2019-12-05, 17:22:34] INFO    : Computing composition and formula.
[2019-12-05, 17:22:34] WARNING : Fe in (Heme) is not supported in the computation of␣
↪M0 and M1
[2019-12-05, 17:22:34] INFO    : Computing neutral mass
[2019-12-05, 17:22:34] INFO    : Computing M0 and M1
```

[8]:
```
# Read output file with different name.
df = pd.read_csv("seq_stfi.tsv", sep="\t")
df.head()
```

[8]:

	pep_name	pep_sequence	pep_charge	stfi_neutral_mass	stfi_formula	stfi_formula_X	stfi_M0_NC	stfi_M1_NC	stfi_M0_12C	stfi_M1_12C
0	seq1	YAQEISR	2	865.429381	C37H61O13N11	C37H61O13N11	0.620499	0.280949	0.920444	0.051819
1	seq2	VLLIDLRIPQR(Phospho)SAINHIVAPNLVNVDPNLLWDK	3	3838.102264	C172H288O49N48P1	C172H288O49N48P1	0.113085	0.236277	0.707156	0.174161
2	seq3	QRTTFFVLGINTVNYPDIYEHILER	2	3037.566156	C140H214O40N36	C140H214O40N36	0.171920	0.290033	0.764407	0.142807
3	seq4	AELFL(Glutathione)LNR	1	1279.623072	C55H90O18N15S1	C55H90O18N15S1	0.470882	0.318073	0.846220	0.072875
4	seq5	.(Acetyl)VGEVFINYIQRQNELFQGKLAYLII(Oxidation)D...	4	5049.638616	C226H365O68N61S1	C226H365O68N61S1	0.054173	0.148735	0.602333	0.195036

3.2.3 Specifying unlabelled amino acids

[9]:
```
!seq-to-first-iso peptides.tsv pep_sequence pep_charge -u V,W
```

```
Namespace(charge_col_name='pep_charge', input_file_name=PosixPath('peptides.tsv'),␣
↪output=None, sequence_col_name='pep_sequence', unlabelled_aa=['V', 'W'])
[2019-12-05, 17:22:36] INFO    : Amino acid with default abundance: ['V', 'W']
[2019-12-05, 17:22:36] INFO    : Parsing file
[2019-12-05, 17:22:36] INFO    : Read peptides.tsv
[2019-12-05, 17:22:36] INFO    : Found 11 lines and 3 columns
[2019-12-05, 17:22:36] INFO    : Reading sequences.
[2019-12-05, 17:22:36] INFO    : Computing composition and formula.
[2019-12-05, 17:22:36] WARNING : Fe in (Heme) is not supported in the computation of␣
↪M0 and M1
[2019-12-05, 17:22:36] INFO    : Computing neutral mass
[2019-12-05, 17:22:36] INFO    : Computing M0 and M1
```

[10]:
```
# Read output file with different name and unlabelled amino acids.
df = pd.read_csv("peptides_stfi.tsv", sep="\t")
df.head()
```

[10]:

	pep_name	pep_sequence	pep_charge	\
0	seq1	YAQEISR	2	
1	seq2	VLLIDLRIPQR(Phospho)SAINHIVAPNLVNVDPNLLWDK	3	
2	seq3	QRTTFFVLGINTVNYPDIYEHILER	2	
3	seq4	AELFL(Glutathione)LNR	1	
4	seq5	.(Acetyl)VGEVFINYIQRQNELFQGKLAYLII(Oxidation)D...	4	

	stfi_neutral_mass	stfi_formula	stfi_formula_X	stfi_M0_NC	\
0	865.429381	C37H61O13N11	C37H61O13N11	0.620499	
1	3838.102264	C172H288O49N48P1	C141H288O49N48P1X31	0.113085	
2	3037.566156	C140H214O40N36	C130H214O40N36X10	0.171920	
3	1279.623072	C55H90O18N15S1	C55H90O18N15S1	0.470882	
4	5049.638616	C226H365O68N61S1	C211H365O68N61S1X15	0.054173	

	stfi_M1_NC	stfi_M0_12C	stfi_M1_12C
0	0.280949	0.920444	0.051819
1	0.236277	0.508195	0.293976
2	0.290033	0.687130	0.202001
3	0.318073	0.846220	0.072875
4	0.148735	0.513344	0.248734

The carbon of unlabelled amino acids is shown as `X` in column `stfi_formula_X`.

For peptide `YAQEISR`, there is no unlabelled amino acids, `stfi_formula` and `stfi_formula_X` are identical. M0 and M1 intensities are not affected by the V and W auxotrophy.

[ ]:
```
```

CHAPTER

FOUR

REFERENCE MANUAL

4.1 seq-to-first-iso

Compute intensities of the first two isotopologue.

Use peptide sequences and charges.

The program computes M0 and M1 and differentiate labelled (with a 99.99 % C[12] enrichment) and unlabelled amino acids.

Read a .tsv file composed of amino acid sequences on each line and return: sequence, mass, formula, formula_X, M0_NC, M1_NC, M0_12C and M1_12C in a .tsv file.

Formula_X is the chemical formula with carbon of unlabelled amino acids marked as X.

NC means Normal Condition, 12C means C[12] enrichment condition.

Example

Running the script after installation

```
$ seq-to-first-iso sequences.tsv sequence_column_name charge_column_name
```

will provide file 'sequences_stfi.tsv'

Notes

Carbon of unlabelled amino acids keep default isotopic abundance, and are represented as X in formulas. Naming conventions for isotopes follow pyteomics's conventions.

seq_to_first_iso.**parse_input_file**(*filename*, *sep='\t'*)

Parse input file.

Parameters

- **filename** (*str*) – Filename, the file can either just have sequences for each line or can have have annotations and sequences with a separator in-between.
- **sep** (*str, optional*) – Separator for files with annotations (default is \t).

Returns

Return type pandas.DataFrame

Raises

- **FileNotFoundError** – If the input file is not found. Exception chaining is explicitly suppressed (from None).
- **Exception** – If the input file cannot be read with pandas. Exception chaining is explicitly suppressed (from None).

seq_to_first_iso.**filter_input_dataframe**(*dataframe*, *sequence_col_name*, *charge_col_name*)

Filter input file with peptide sequences and charges.

Parameters

- **dataframe** (*pandas.DataFrame*) – Raw dataframe with all input columns
- **sequence_col_name** (*str*) – Name of column with peptide sequences
- **charge_col_name** (*str*) – Name of column with peptide charges

Returns

With columns :
- "sequence": peptide sequences.
- "charge": peptide charges.

Return type pandas.DataFrame

Raises **KeyError** – If the sequence or charge column is not found.

seq_to_first_iso.**check_amino_acids**(*seq*)

Check elements of a sequence are known amino acids.

Parameters **seq** (*str*) – Peptide sequence.

Returns

(sequence, "") if the sequence is composed of allowed amino acids
("", "Unrecognized amino acids.") if the sequence is composed of unallowed amino acids.

Return type Tuple of two str

seq_to_first_iso.**separate_labelled**(*sequence*, *unlabelled_aa*)

Get the sequence of unlabelled amino acids from a sequence.

Parameters

- **sequence** (*str*) – String of amino acids.
- **unlabelled_aa** (*container object*) – Container (list, string...) of unlabelled amino acids.

Returns

The sequences as a tuple of string with:
- the sequence without the unlabelled amino acids

- the unlabelled amino acids in the sequence

Return type tuple(str, str)

seq_to_first_iso.**compute_M0_nl**(*formula*, *abundance*)
Compute intensity of the first isotopologue M0.

Handle element X with specific abundance.

Parameters

- **formula** (*pyteomics.mass.Composition*) – Chemical formula, as a dict of the number of atoms for each element: {element_name: number_of_atoms, ... }.
- **abundance** (*dict*) – Dictionary of abundances of isotopes: {"element_name[isotope_number]": relative abundance, ..}.

Returns Value of M0.

Return type float

Notes

X represents C with default isotopic abundance.

seq_to_first_iso.**compute_M1_nl**(*formula*, *abundance*)
Compute intensity of the second isotopologue M1.

Handle element X with specific abundance.

Parameters

- **formula** (*pyteomics.mass.Composition*) – Chemical formula, as a dict of the number of atoms for each element: {element_name: number_of_atoms, ... }.
- **abundance** (*dict*) – Dictionary of abundances of isotopes: {"element_name[isotope_number]": relative abundance, ..}.

Returns Value of M1.

Return type float

Notes

X represents C with default isotopic abundance.

seq_to_first_iso.**formula_to_str**(*composition*)
Return formula from Composition as a string.

Parameters **composition** (*pyteomics.mass.Composition*) – Chemical formula.

Returns Human-readable string of the formula.

Return type str

Warning: If the composition has elements not in USED_ELEMS, they will not be added to the output.

seq_to_first_iso.**convert_atom_C_to_X**(*sequence*)
Replace carbon atom by element X atom in a composition.

Parameters **sequence** (*str or pyteomics.mass.Composition*) – Sequence or composition.

Returns Composition with carbon atoms replaced by element X atoms.

Return type pyteomics.mass.Composition

seq_to_first_iso.**get_charge_composition**(*charge*)
Return the composition of a given charge (only H+).

Parameters **charge** (*int*) – Peptide charge.

Returns Composition of the change (H+).

Return type pyteomics.mass.Composition

seq_to_first_iso.**get_mods_composition**(*modifications*)
Return the composition of a list of modifications.

Parameters **modifications** (*list of str*) – List of modifications string (corresponding to Unimod titles).

Returns The total composition change.

Return type pyteomics.mass.Composition

seq_to_first_iso.**compute_intensities**(*df_peptides*, *unlabelled_aa=[]*)
Compute isotopologues intensities from peptide sequences.

Parameters

- **df_peptides** (*pandas.DataFrame*) – Dataframe with column 'sequence' and 'charge'
- **unlabelled_aa** (*container object*) – Container of unlabelled amino acids.

Returns

Dataframe with all computed values, compositions and formulas.

Return type pandas.DataFrame

Notes

Supports Xtandem's Post-Translational Modification notation (0.4.0).

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex

PYTHON MODULE INDEX

S

INDEX