SWATHLib

Technical Manual and Common Usage

Toan K. Phung

November 19, 2018

Contents

Chapter 1

Introduction

SWATHLib is an attempt at creating an automated workflow for creation of manually curated spectral library for SWATH downstream analysis. The program itself is composed of two main component, the front-end created using TypeScript with Electron and the Angular framework and the back-end created using Python with Tornado as the server implementation.

1.1 General Automated Workflow Summary

1. Input of user sequences and select parameters within the GUI (e.g SWATH windows, retention time (RT), modifications, result output format...).
2. Submission of user's queries to SWATHLib back-end.
3. Check whether there are any modifications and change the workflow settings accordingly.
4. The program attempt to generate possible combination of all the modifications from user query.
5. For each combination, transition sequences of Y and/or b and/or y-series would be created and their m/z values calculated.
6. Transition with m/z within **50-1800 m/z** range would be recorded for each combination of RT and window.
7. If oxonium ions were included within the queries, for each unique combination of precursor, RT and window, entries for oxonium ions would be recorded for them that combination.
8. The end results would be communicated to the user for retrieval.

1.2 GUI System Requirement

Windows

- Windows 7 or later
- Intel Pentium 4 processor or later with SSE2
- 512 MB of RAM

Mac

- OS X 10.9 and later
- 64-bit Intel processor
- 512 MB of RAM

Linux

- Debian 8, Fedora 21, Ubuntu 12.04 and later
- Intel Pentium 4 processor or later with SSE2

1.3 Back-end System Requirement

- Any system that can support Python 3.5+ with Tornado v5+ and Pyteomics v3.5.1+ packages.

Chapter 2

Graphical User Interface

The GUI at its core is a chromium web instant. At the top right is a button that can be used to access the back-end connection management windows where the user can start an instant of the Python back-end and manually adding more server connections to split user's the workload.

2.1 User Input

The GUI accepts protein sequences in FASTA format either in raw text or a text file. The user can perform in-silico trypsin digestion of the input with optional ability to manually or automatically select sites for misdigestion.

Modification Customization

Within user settings tab, the user can use their own table of modifications as modifications source for the program. The table is made up of 10 columns where,

- **name** is the name of the modification to be displayed on the GUI.
- **label** is the internal label of the modification to be used by the back-end. Must be in lowercase. No modifications within the same query should have the same label beside transition of the same Ytype variable modification.
- **mass** is the mass of the modification.
- **regex** is the regular expression representation of the the modification.
- **type** is the modification type *Ytype*, *variable*, or *static*.
- **Ytype** is the label of the Ytype transition.
- **multiplepattern** is whether the query with this variable modification is default to have all modification pattern generated.
- **status** is whether the query with this variable modification is default to be filled in every instance on the sequence.

Table 2.1: Example of a user submitted modifications table

name	label	mass	regex	type	Ytype	multiplepattern	status	mlabel	offset
O-Mannose	g	0	[S|T]	Ytype	Y0	false	false		0
O-Mannose	g	162.05	[S|T]	Ytype	Y1	false	false		0
HexNAc	h	0	N[^P][S|T]	Ytype	Y0	false	false		2
Propionamide	c	71.037114	C	static		false	false	PPa	0
Carboxylation	e	43.98983	E	variable		false	false		0