Compute the Number of Peptides of a Given Total Mass

Group Members:

1. Furkan Ayberk Binbay
2. Funmilayo
3. Linus
4. Ram Kumar R S

b-it Bonn-Aachen International Center for Information Technology

"Okay—who put my lunch through the mass spectrometer..?"

OUTLINE

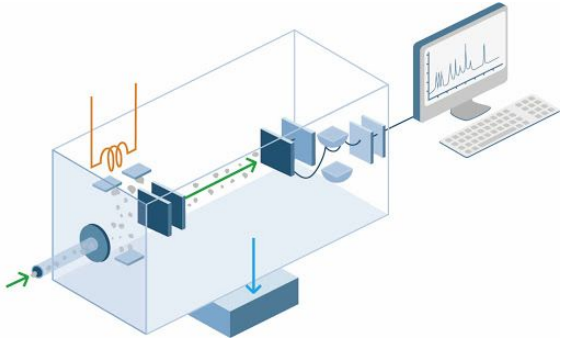
1. Introduction [Furkan Ayberk]
2. Methodology [Linus]
3. Software Overview [Ram Kumar]
4. Discussions [Funmilayo]

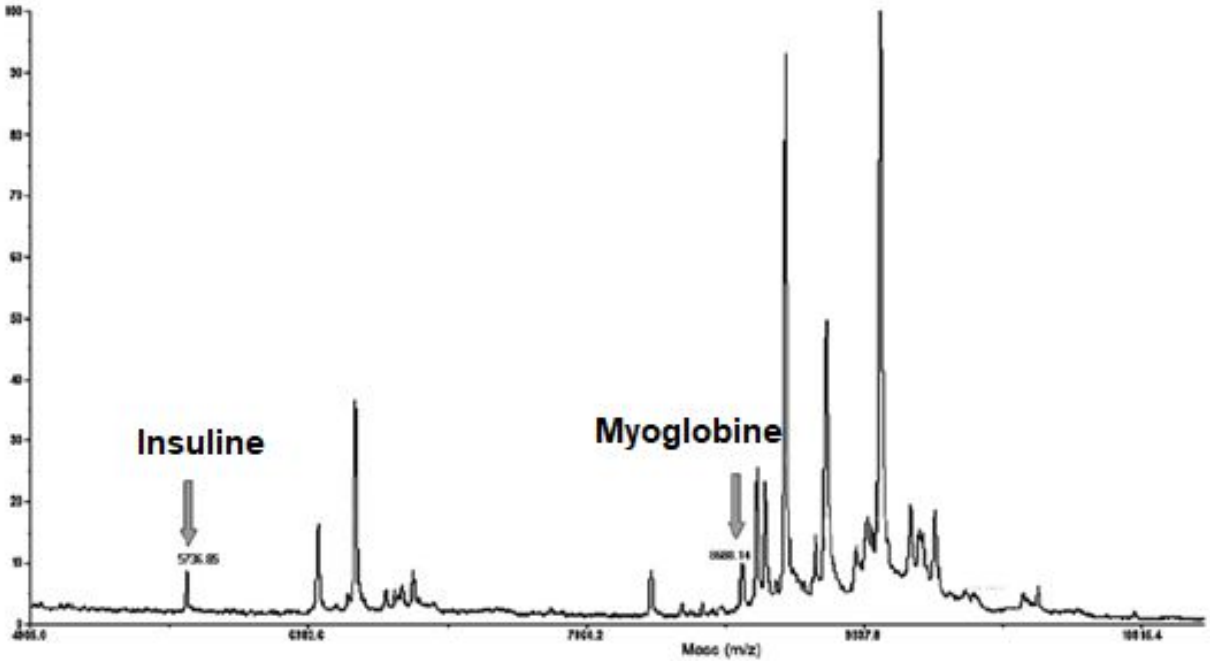
Mass Spectrometry

- Qualitative & quantitative analysis of proteins and peptides

- Large amounts of high-quality data that allow
  - Protein identification
  - Annotation of secondary modifications
  - Determination of the abundance of individual proteins

- A spectrum consists of
  - Intensity
  - Mass-to-charge ratio (m/z)



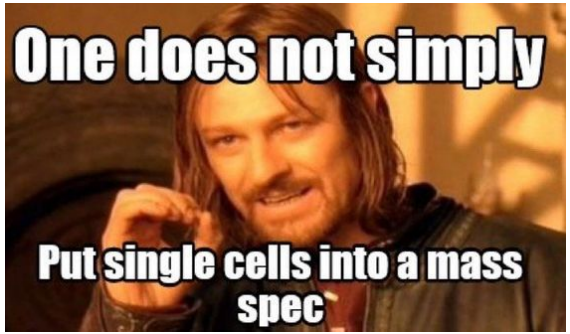


Veltri, P. (2008). Algorithms and tools for analysis and management of mass spectrometry data. Briefings in bioinformatics, 9(2), 144-155.

Cont'd

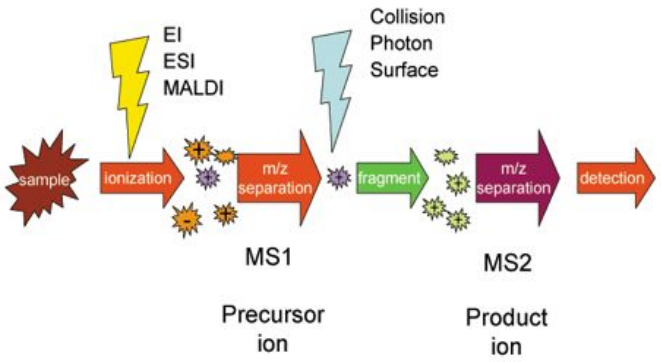
- Protein analysis by mass spectrometry is performed in a bottom-up fashion
  - Trypsin cleavage of protein
  - Some database search programs allow specification of enzyme specificity

- Problems
  - Not all peptides occur in the mass spectrum
  - Long proteins tend to be preferred



Identification of Peptides

- Two ways to identify of peptides using mass spectrometry
  - Peptide mass fingerprinting (PMF) in MS spectra
  - Peptide identification using MS/MS spectra
    - Collision induced dissociation



- The experimental data are compared with calculated peptide mass or fragment ion mass values
  - Mass values are counted
  - Aim: the best matches

Cont'd

- How do we score the data against the theoretical spectrum and how significant is the score?
  - MOWSE (Molecular Weight Search)
    - Each calculated value which falls within a given mass tolerance of an experimental value counts as a match

- Data preprocessing
  - mzML & mzXML
  - The final result from the processing algorithms :
    - a "peak list" of peptide masses (MS)
    - peptide fragment masses (MS/MS)

Cont'd

- Database searching
  – Peak list vs. Protein sequence databases

- Database-dependent search algorithms

- Programs to interpret MS and MS/MS spectra

Name	Input Data	Interfaced from VEMS	Public
VEMS v3.0	MS, MS/MS	No	Yes
Mascot	MS, MS/MS	Yes	Semi
X!Tandem	MS/MS	Yes	Yes
P3	MS/MS	No	Yes
Inspect	MS/MS	No	Yes
Phenyx	MS/MS	No	Semi
PepNovo	MS/MS	No	Yes
Lutefisk	MS/MS	Yes	Yes
OpenSea	MS/MS	No	Yes
De Novo peaks	MS/MS	No	Yes
PepHMM	MS/MS	No	Yes
ProteinProphet	MS/MS	No	Yes

Matthiesen, R., & Jensen, O. N. (2008). Analysis of mass spectrometry data in proteomics. In *Bioinformatics* (pp. 105-122). Humana Press.

2. Methodology

Task 1 - *Parse the Data* (3 pts)

- Given a mzML or mzXML file of a peptide mass spectrum, parse it for its relevant values
  - e.g. intensity, *m/z*, etc

Task 2 - *From Data to Peptides* (2 pts)

- Based on the values extracted in Task 1, compile a list of the most likely peptides that the spectrum may represent

Task 3 - *Protein Prediction* (2 pts)

- Assemble a list of possible proteins that the amino acid sequences (peptides) generated in Task 2 may represent
  - This may require using tools available from UniProt, EBI, NCBI, BLAST, or elsewhere

Task 4 - *GUI* (3 pts)

- Construct a web interface that allows one to upload a mzML or mzXML file and get a list of possible proteins it may be derived from. Your interface should include the following features:
  - An upload button that allows one to upload a mzML or mzXML file
  - A table of relevant values derived from the uploaded file
  - A collapsable table of the possible peptides that the spectrum values may represent (and their amino acid sequences)
  - A collapsable table of the possible proteins that each peptide could be derived from. In the case of multiple proteins, show the most likely based on the similarity search values

MassSpectronomy (peak data)
->
Task 2 – Peptide Sequence

MS Database Search Engines

X! TANDEM Spectrum Modeler (https://www.thegpm.org/tandem/)

SEQUEST (*Eng, McCormack and Yates, 1994)*

MS-BLAST: **M**ass **S**pectrometry driven **BLAST**: Shevchenko et al. (2001).
(http://genetics.bwh.harvard.edu/msblast/)

…

MASCOT ~1993/1999 (most widely used)

APIs ?

MATRIX SCIENCE

Search this site

Home | Mascot database search | Products | Technical support | Training | News | Blog | Newsletter | Contact | Useful links

Access Mascot Server | Database search help

Mascot database search > Access Mascot Server

Access Mascot Server

You are welcome to submit searches to this free Mascot Server. Searches of MS/MS data are limited in size and some functions, such as no enzyme searches, are unavailable. Automated searching of batches of files is not permitted. If you want to automate search submission, perform large searches, search additional sequence databases, or customise the modifications, quantitation methods, etc., you'll need to license your own, in-house copy of Mascot Server.

Peptide Mass Fingerprint

The experimental data are a list of peptide mass values from the digestion of a protein by a specific enzyme such as trypsin.

Perform search | Example of results report | Tutorial

More info

- Mascot overview
- Search parameter reference
- Data file format
- Results report overview

www.matrixscience.com

Mascot Example Query

MASCOT Sequence Query

Your name | Email

Search title

Database(s): Plants_EST, Prokaryotes_EST, Rodents_EST, Vertebrates_EST, contaminants

Enzyme: Trypsin

Allow up to 1 missed cleavages

Quantitation: None

Taxonomy: All entries

Fixed modifications: --- none selected ---

Display all modifications

Variable modifications: --- none selected ---

Carboxymethyl (C)
Cation:Na (C-term)
Cation:Na (DE)
Deamidated (NQ)
Dehydrated (N-term C)
Dehydro (C)
Dioxidation (M)
Formyl (N-term)
Formyl (Protein N-term)
Gln->pyro-Glu (N-term Q)
Glu->pyro-Glu (N-term E)

Peptide tol. ± 1.2 Da # ^{13}C 0 MS/MS tol. ± 0.6 Da

Peptide charge Mr Monoisotopic Average

Query

Mascot Example Results

1. CASB_BOVIN **Mass:** 25091 **Score:** 78 **Matches:** 1(1) **Sequences:** 1(1)
Beta-casein OS=Bos taurus OX=9913 GN=CSN2 PE=1 SV=2
☐ Check to include this hit in error tolerant search

Query	Observed	Mr(expt)	Mr(calc)	Delta	Miss	Score	Expect	Rank	Unique	Peptide
☑ 1	1031.4000	2060.7854	1980.8548	79.9306	0	78	1.8e-06	1	U	K.FQSEEQQQTEDELQDK.I

Proteins matching the same set of peptides:
CASB_CAPHI **Mass:** 24849 **Score:** 78 **Matches:** 1(1) **Sequences:** 1(1)
Beta-casein OS=Capra hircus OX=9925 GN=CSN2 PE=2 SV=1
CASB_SHEEP **Mass:** 24859 **Score:** 78 **Matches:** 1(1) **Sequences:** 1(1)
Beta-casein OS=Ovis aries OX=9940 GN=CSN2 PE=1 SV=3

---

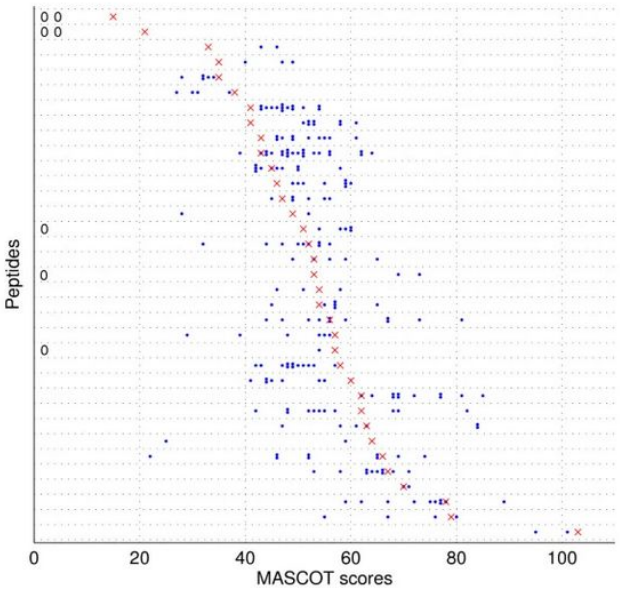
2. CLU_DICDI **Mass:** 148595 **Score:** 78 **Matches:** 1(1) **Sequences:** 1(1)
Clustered mitochondria protein homolog OS=Dictyostelium discoideum OX=44689 GN=clua PE=1 SV=2
☐ Check to include this hit in error tolerant search

Query	Observed	Mr(expt)	Mr(calc)	Delta	Miss	Score	Expect	Rank	Unique	Peptide
1	1031.4000	2060.7854	1798.8949	261.8906	1	78	1.8e-06	1	U	K.LGGTPEEQQKDIEDLK.A

3. PESC_NEUCR **Mass:** 75247 **Score:** 78 **Matches:** 1(1) **Sequences:** 1(1)
Pescadillo homolog OS=Neurospora crassa (strain ATCC 24698 / 74-OR23-1A / CBS 708.71 / DSM 1257 / FGSC 987) OX=
☐ Check to include this hit in error tolerant search

Query	Observed	Mr(expt)	Mr(calc)	Delta	Miss	Score	Expect	Rank	Unique	Peptide
1	1031.4000	2060.7854	1596.7380	464.0475	0	78	1.8e-06	1	U	K.AVTNGEEQQQGPDPK.V

Example Scoring (MASCOT)



- Liu, J., Bell, A.W., Bergeron, J.J. *et al.* Methods for peptide identification by spectral comparison. *Proteome Sci* **5,** 3 (2007). https://doi.org/10.1186/1477-5956-5-3

- https://proteomesci.biomedcentral.com/articles/10.1186/1477-5956-5-3

OpenMS

OpenMS is an open-source project for data analysis and processing in protein mass spectrometry and is released under the 3-clause BSD licence. It supports most common operating systems including Microsoft Windows, OS X and Linux.[2]

OpenMS has tools for many common data analysis pipelines used in proteomics, providing algorithms for signal processing, feature finding (including de-isotoping), visualization in 1D (spectra or chromatogram level), 2D

OpenMS

Developer(s)	Over 65 individuals
Initial release	1 July 2007; 13 years ago
Stable release	2.6.0 / 30 September 2020; 4 months ago
Repository	github.com/OpenMS/OpenMS/releases
Written in	C++ (with bindings to Python)
Operating system	Linux, Windows, OS X
Size	215 MB [1]
Available in	English
Type	Bioinformatics / Mass spectrometry software
License	BSD licenses 3-clause
Website	openms.de

https://en.wikipedia.org/wiki/OpenMS

OpenMS

To achieve a wide variety of tasks in proteomics, OpenMS provides The OpenMS Proteomics Pipeline (TOPP) which is a set of computational tools that can be chained together to tailor problem-specific analysis pipelines for HPLC-MS data. It transforms most of the OpenMS functionality into small command line tools that are the building blocks for more complex analysis pipelines.[2]

- over 100 different executable tools

https://en.wikipedia.org/wiki/OpenMS

pyOpenMS

- file handling (mzXML, mzML, TraML, mzTab, fasta, pepxml, protxml, mzIdentML among others)
- chemistry (mass calculation, peptide fragmentation, isotopic abundances)
- signal processing (smoothing, filtering, de-isotoping, retention time correction and peak-picking)
- identification analysis (including peptide search, PTM analysis, Cross-linked analytes, FDR control, RNA oligonucleotide search and small molecule search tools)
- quantitative analysis (including label-free, metabolomics, SILAC, iTRAQ and SWATH/DIA analysis tools)
- chromatogram analysis (chromatographic peak picking, smoothing, elution profiles and peak scoring for SRM/MRM/PRM/SWATH/DIA data)
- interaction with common tools in proteomics and metabolomics
  - search engines such as Comet, Crux, Mascot, MSGFPlus, MSFragger, Myrimatch, OMSSA, Sequest, SpectraST, XTandem
  - post-processing tools such as percolator, MSStats, Fido
  - metabolomics tools such as SIRIUS, CSI:FingerId

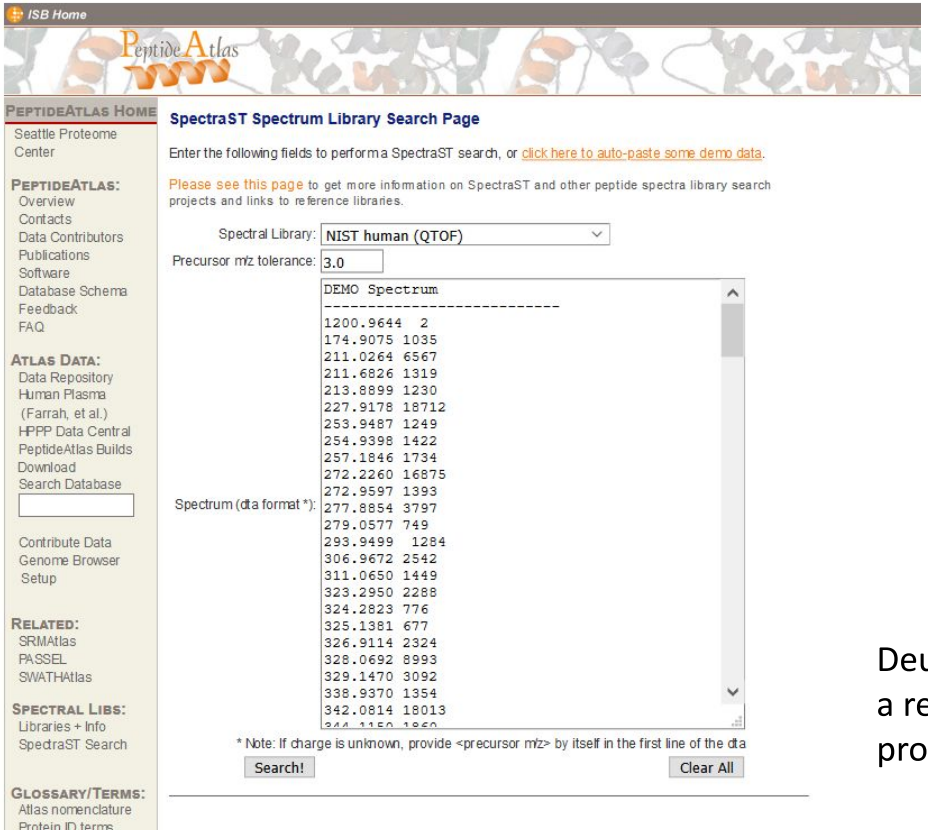
'pyopenms.readthedocs.io/en/latest/

=> PeptideSequence

Peptide hit sequence	Peptide hit monoisotopic m/z	Peptide ppm error	Peptide hit score
DFASSGGYVLHLHR	520.26	5.42	16.84
IALSRPNVEVVALNDPFITNDYAAYM(Oxidation)FK	1063.21	0.15	42.22
RPGADSDIGGFGGLFDLAQAGFR	775.39	3.60	34.94

PeptideSequence
->
Task 3 – Protein Prediction

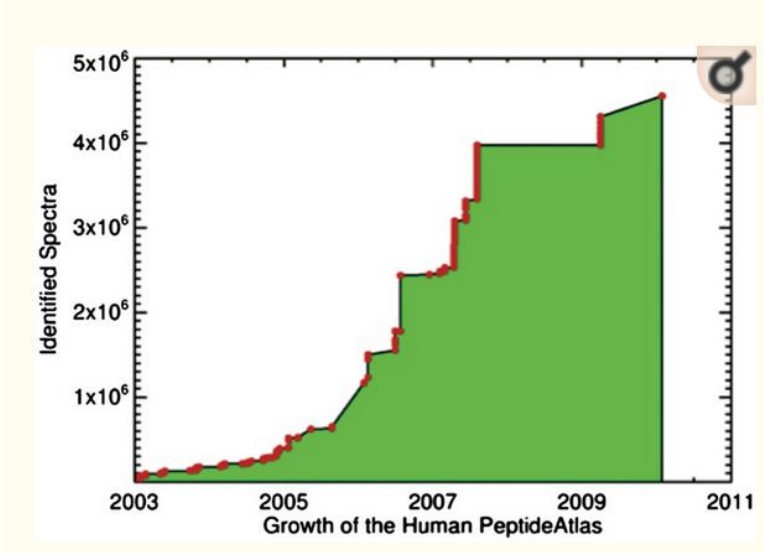
PeptideAtlas



PeptideAtlas is a proteomics data resource that gathers **tandem mass spectrometry datasets** from around the world, **reprocesses** them with the **Trans-Proteomic Pipeline**, and makes the combined result **freely available to the community**.

Deutsch, E. W., Lam, H., & Aebersold, R. (2008). PeptideAtlas: a resource for target selection for emerging targeted proteomics workflows. *EMBO reports*, *9*(5), 429-434.

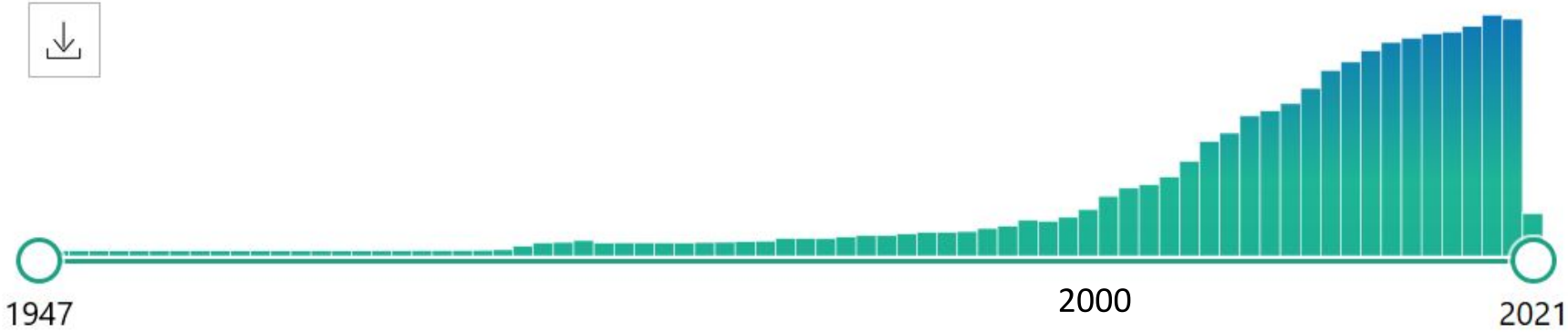
PeptideAtlas



Killcoyne, S., Handcock, J., Robinson, T., Deutsch, E. W., & Boyle, J. (2012). Interfaces to PeptideAtlas: a case study of standard data access systems. *Briefings in bioinformatics*, *13*(5), 615–626. https://doi.org/10.1093/bib/bbr067

PubMed: Mass Spectronomy

369,273 results



PeptideAtlas Interfaces

Various APIs:

- SOAP

- REST

Killcoyne, S., Handcock, J., Robinson, T., Deutsch, E. W., & Boyle, J. (2012). Interfaces to PeptideAtlas: a case study of standard data access systems. *Briefings in bioinformatics*, *13*(5), 615–626. https://doi.org/10.1093/bib/bbr067

https://dzone.com/articles/comprehensive-guide-rest-vs-soap

PeptideAtlas Interfaces

- Google Data Source service: http://informatics.systemsbiology.net/google-dsapi-svc/addama/datasources/spectra_service/spectra_peptide, please see http://informatics.systemsbiology.net/informatics/project/spectraservice for example queries with HTML table results.
- BioMart service: http://informatics.systemsbiology.net/biomart/martview.

-> SQL Databases

Killcoyne, S., Handcock, J., Robinson, T., Deutsch, E. W., & Boyle, J. (2012). Interfaces to PeptideAtlas: a case study of standard data access systems. *Briefings in bioinformatics*, *13*(5), 615–626. https://doi.org/10.1093/bib/bbr067

PeptideAtlas RESTAPI

ISB Home

PeptideAtlas

PeptideAtlas Home

Seattle Proteome Center

PeptideAtlas:

Overview
Contacts
Data Contributors
Publications
Software
Database Schema
Feedback
FAQ

Atlas Data:

Data Repository
Human Plasma (Farrah, et al.)
HPPP Data Central
PeptideAtlas Builds
Download
Search Database

Contribute Data
Genome Browser Setup

Documentation for ProMaST map function

Endpoint:

http://www.peptideatlas.org/api/promast/v1/map

Parameters:

```
peptide     peptide sequence to search
proteome    name of the reference proteome to search
fuzzy       number of wildcards to consider (0-3)
tolerance   mass tolerance for matching wildcards
```

Examples:

```
curl -X GET 'http://www.peptideatlas.org/api/promast/v1/map?proteome=Human&peptide=ALFLETEQLK'

curl -X GET --header 'Accept: application/json' 'http://www.peptideatlas.org/api/promast/v1/map?proteome=Human&peptide=ALFLETEQLK'

curl -X GET 'http://www.peptideatlas.org/api/promast/v1/map?proteome=Human&peptide=ALFLETEQLK&fuzzy=1'

curl -X GET 'http://www.peptideatlas.org/api/promast/v1/map?proteome=Human&peptide=ALFLETEQLK&fuzzy=3&tolerance=0.0001'

curl -X GET --header 'Accept: application/json' 'http://www.peptideatlas.org/api/promast/v1/map?proteome=Human&peptide=ALFLETEQLK&fuzzy=3&tolerance=0.0
```

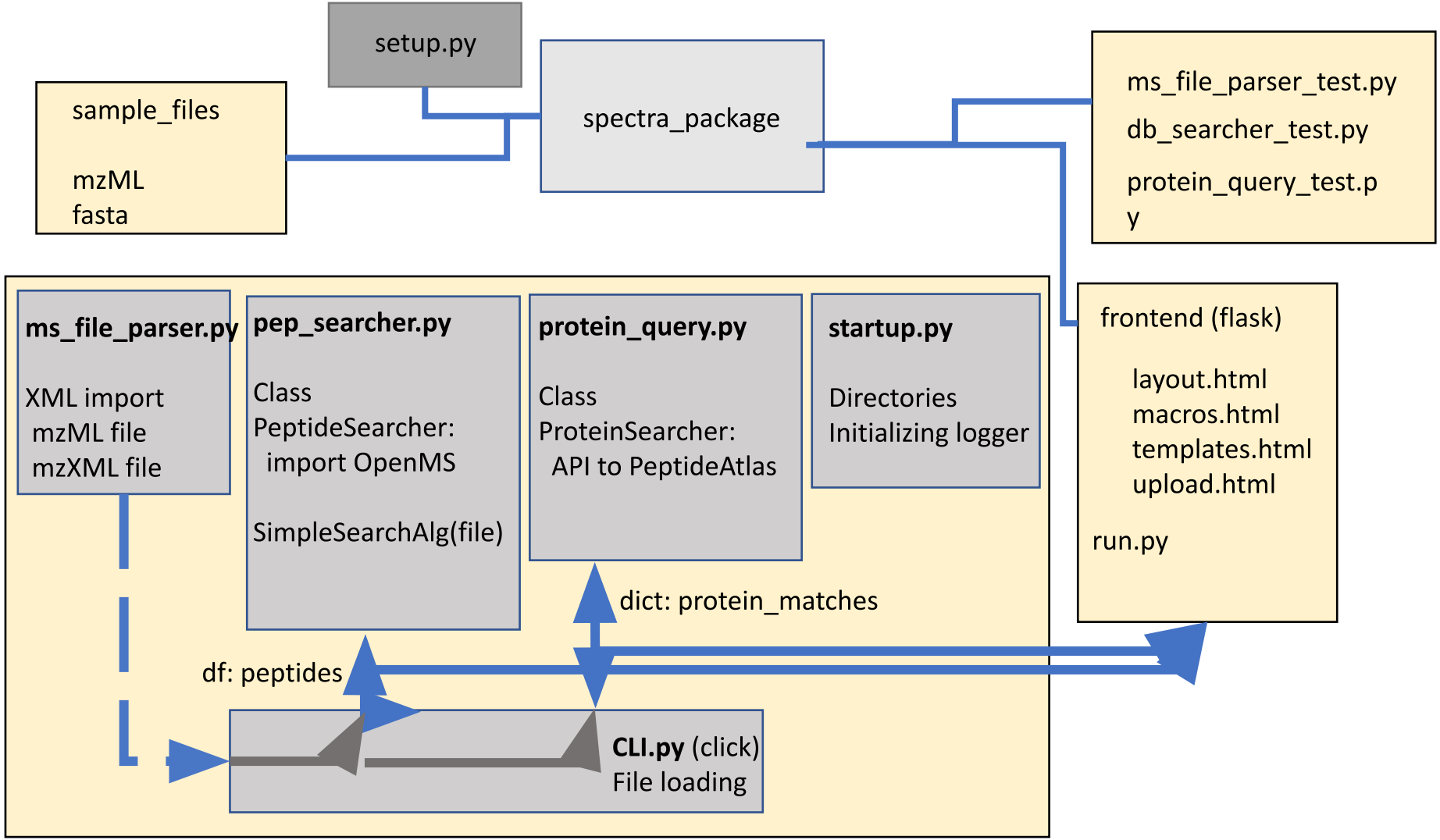
http://www.peptideatlas.org/api/promast/v1/map?proteome=Human&peptide=ALFLETEQLK&output=tsv

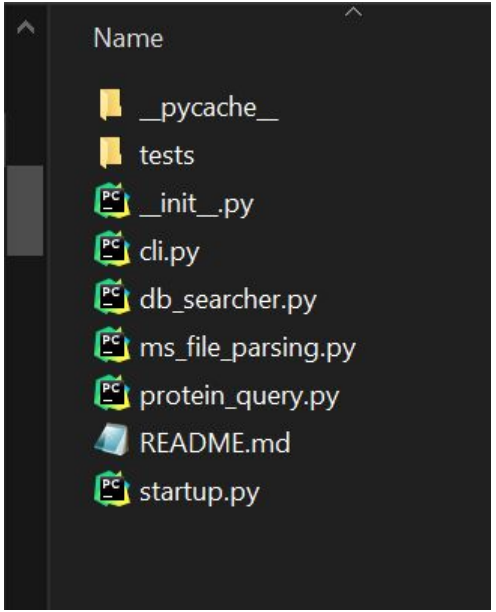
http://www.peptideatlas.org/api/promast/v1/map?proteome=Human&peptide=ALFLETEQLK&output=json

File Edit Format View Help

```
peptide protein location
ALFLETEQLK      ENSP00000305988.5 sp|Q13740|CD166_HUMAN NP_001618.2     143
ALFLETEQLK      ENSP00000418213.2 tr|F5GXJ9|F5GXJ9_HUMAN        92
ALFLETEQLK      ENSP00000419236.2 sp|Q13740-2|CD166_HUMAN NP_001230209.1        143
ALFLETEQLK      NP_001230210.1  143
ALFLETEQLK      nxp:NX_Q13740-1 143
ALFLETEQLK      nxp:NX_Q13740-2 143
ALFLETEQLK      tr|B3KNN9|B3KNN9_HUMAN  143
ALFLETEQLK      tr|B4DX43|B4DX43_HUMAN  92
```

Project Structure


setup.py
sample_files
mzML
fasta
spectra_package
ms_file_parser_test.py
db_searcher_test.py
protein_query_test.p
y
ms_file_parser.py
XML import
mzML file
mzXML file
pep_searcher.py
Class
PeptideSearcher:
import OpenMS
SimpleSearchAlg(file)
protein_query.py
Class
ProteinSearcher:
API to PeptideAtlas
startup.py
Directories
Initializing logger
frontend (flask)
layout.html
macros.html
templates.html
upload.html
run.py
dict: protein_matches
df: peptides
CLI.py (click)
File loading


Name
__pycache__
tests
__init__.py
cli.py
db_searcher.py
ms_file_parsing.py
protein_query.py
README.md
startup.py

3. Software Overview

Software File Structure:

- group01 C:\Users\Ram Kumar R S\Desktop\group01
  - .pytest_cache
  - frontend
    - templates
      - layout.html
      - macros.html
      - template.html
      - upload.html
    - __init__.py
    - run.py
  - presentation
  - sample_files
  - spectra_package
    - .idea
    - tests
    - __init__.py
    - cli.py
    - db_searcher.py
    - ms_file_parsing.py
    - protein_query.py
    - README.md
    - startup.py

Base Coding Scripts:

1. ms_file_parsing.py
2. db_searcher.py
3. protein_query.py

MS file Types:

```
<scan num="11"
      msLevel="2"
      peaksCount="5"
      retentionTime="PT14.860000S"
      collisionEnergy="35"
      startMz="110.0000"
      endMz="905.0000"
      lowMz="358.5060"
      highMz="430.1332"
      basePeakMz="428.8915"
      basePeakIntensity="129564.0000"
      totIonCurrent="171832.0000">
  <precursorMz precursorIntensity="126541.000000">445.293030</precursorMz>
  <peaks precision="32"
         byteOrder="network"
         pairOrder="m/z-int">Q7NAxkbsEgBDs+hwRcOwAEPWchxH/Q4AQ9bBakBAAABD1xEMRbTQAA==</peaks>
</scan>
```

Structure of mzML file

Structure of mzXML files


```
<spectrumList count="4" defaultDataProcessingRef="pwiz_processing">
  <spectrum index="0" id="scan=19" defaultArrayLength="15">
    <referenceableParamGroupRef ref="CommonMS1SpectrumParams"/>
    <cvParam cvRef="MS" accession="MS:1000511" name="ms level" value="1"/>
    <cvParam cvRef="MS" accession="MS:1000127" name="centroid spectrum" value=""/>
    <cvParam cvRef="MS" accession="MS:1000528" name="lowest observed m/z" value="400.38999999999999" unitCvRef="MS" unitAccession="MS:1000040" unitName="m/z"/>
    <cvParam cvRef="MS" accession="MS:1000527" name="highest observed m/z" value="1795.5599999999999" unitCvRef="MS" unitAccession="MS:1000040" unitName="m/z"/>
    <cvParam cvRef="MS" accession="MS:1000504" name="base peak m/z" value="445.34699999999998" unitCvRef="MS" unitAccession="MS:1000040" unitName="m/z"/>
    <cvParam cvRef="MS" accession="MS:1000505" name="base peak intensity" value="120053" unitCvRef="MS" unitAccession="MS:1000131" unitName="number of counts"/>
    <cvParam cvRef="MS" accession="MS:1000285" name="total ion current" value="16675500"/>
    <scanList count="1">
      <cvParam cvRef="MS" accession="MS:1000795" name="no combination" value=""/>
      <scan instrumentConfigurationRef="LCQ_x0020_Deca">
        <cvParam cvRef="MS" accession="MS:1000016" name="scan start time" value="5.8905000000000003" unitCvRef="UO" unitAccession="UO:0000031" unitName="minute"/>
        <cvParam cvRef="MS" accession="MS:1000512" name="filter string" value="+ c NSI Full ms [ 400.00-1800.00]"/>
        <cvParam cvRef="MS" accession="MS:1000616" name="preset scan configuration" value="3"/>
        <scanWindowList count="1">
          <scanWindow>
            <cvParam cvRef="MS" accession="MS:1000501" name="scan window lower limit" value="400" unitCvRef="MS" unitAccession="MS:1000040" unitName="m/z"/>
            <cvParam cvRef="MS" accession="MS:1000500" name="scan window upper limit" value="1800" unitCvRef="MS" unitAccession="MS:1000040" unitName="m/z"/>
          </scanWindow>
        </scanWindowList>
      </scan>
    </scanList>
```

Properties extraction from MS files:

ms_file_parsing.py script

Input : MS file
Output : Dataframe containing all attributes

```
for item in root.iter():
    if 'cvParam' in item.tag:
        if item.attrib['name'] == "base peak m/z": base_peak_array.append(item.attrib['value'])
        if item.attrib['name'] == "base peak intensity": base_peak_intensity_array.append(
            item.attrib['value'])
        if item.attrib['name'] == "highest observed m/z": highest_observed_mz.append(
            item.attrib['value'])
        if item.attrib['name'] == "lowest observed m/z": lowest_observed_mz.append(item.attrib['value'])
        if item.attrib["name"] == "total ion current": ion_current.append(item.attrib["value"])

mzMl_df = pd.DataFrame({"Base Peak": base_peak_array,
                        "Base Peak Intensity": base_peak_intensity_array,
                        "Highest Observed M/Z": highest_observed_mz,
                        "Lowest Observed M/Z": lowest_observed_mz,
                        "Total Ion Current": ion_current})
```

Application of pyOpenms to extract Peptide Information:

db_searcher.py

Input : mzML file and its Fasta file
Output : Dataframe containing Peptide Properties

```
protein_ids = []
peptide_ids = []

#SimpleSearchEngineAlgorithm compares mzML file against fasta file, and it gives an output that cont
SimpleSearchEngineAlgorithm().search(self.mzml_file, self.fasta_file, protein_ids, peptide_ids)


# Results Preprocessing
mz_lst_1 = []
mz_lst_2 = []

MZ = int()
RT = int()
meta_val = int()
score_type = int()
hit_rank = int()
hit_charge = int()
hit_seq = str()
hit_monoisotopic = int()
ppm_error = int()
hit_score = int()
```

Code snippet taken from pyOpenMS Documentation

Cont'd:

```
if peptide_ids != []:
    #Exploring the individual hits, and gathering the peptide information
    for peptides in peptide_ids:
        MZ = round(peptides.getMZ(), 2)
        RT = round(peptides.getRT(), 2)
        meta_val = peptides.getMetaValue("scan_index")
        score_type = peptides.getScoreType()
        mz_lst_1.append([MZ, RT, meta_val, score_type])

        for hit in peptides.getHits():
            hit_rank = round(hit.getRank(), 2)
            hit_charge = round(hit.getCharge(), 2)
            hit_seq = hit.getSequence()
            hit_monoisotopic = round(
                hit.getSequence().getMonoWeight(Residue.ResidueType.Full, hit.getCharge()) / hit.getCharge(), 2)
            ppm_error = round(abs(hit_monoisotopic - peptides.getMZ()) / hit_monoisotopic * 10 ** 6, 2)
            hit_score = round(hit.getScore(), 2)

        mz_lst_2.append([hit_rank, hit_charge, str(hit_seq), hit_monoisotopic, ppm_error, hit_score])
```

Packages into a DataFrame

Protein Querying using Protein Atlas API:

Input: Peptide Properties DF
Output: Matched Proteins list

protein_query.py

```
api_query = f"http://www.peptideatlas.org/api/promast/v1/map?peptide={peptide}"

print("Api_query with peptide:", peptide)
r = requests.get(api_query, headers={"Accept": "application/json"})

if r.status_code != 200:
    logger.error(f"{api_query} returned bad status code: {r.status_code}")
    break

# Positive Server Result
if r.json()['status'] == 'OK':
    # Mapping Results
    if 'mappings' in r.json():
        mapping_result = r.json()['mappings']
    else:
        mapping_result = []
        print("0 mappings found")

    # Add to peptide Dictionary
    protein_matches[peptide] = mapping_result
```

4. Discussions

Funmilayo


LIVE DEMO

How our software can be used in the real world

- It can applicable across diverse fields, including forensic toxicology, metabolomics, proteomics, pharma/biopharma, and clinical research.

Environmental Analysis

1. Drinking water testing
2. Pesticide screening and quantitation
3. Soil contamination assessment
4. Carbon dioxide and pollution monitoring,
5. Trace elemental analysis of heavy metals leaching.

Thank You

For Your Attention...