b-it Bonn-Aachen International Center for Information Technology

Compute the Number of Peptides of a Given Total Mass

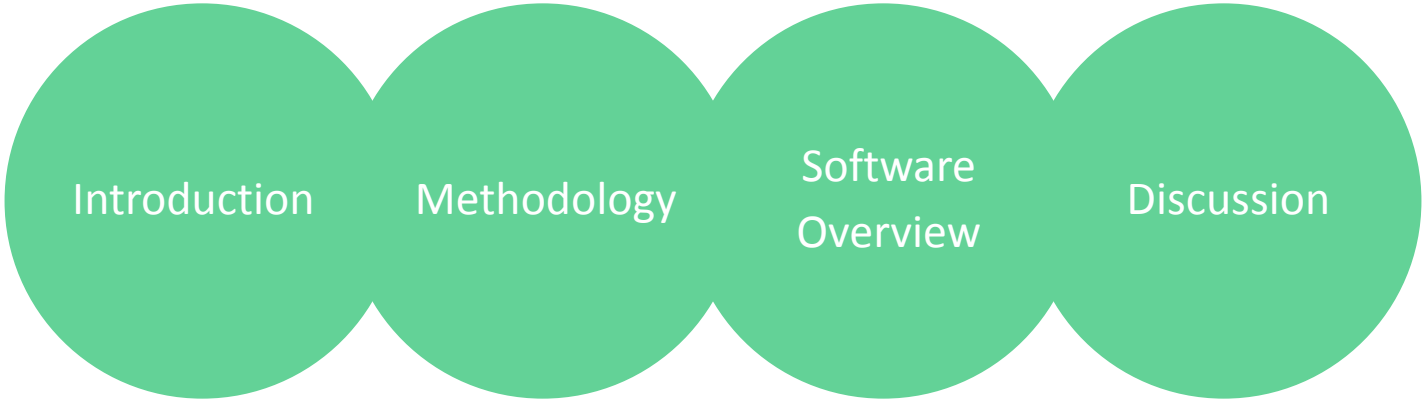
Mass Spectrum 1.0

Programming Lab 2 (Group - 3)

Lara Schmalenstroer, Priya Kempanna, Rohitha Ravinder, Shubhi Ambast

February 15, 2022

Overview



Introduction

Presenter : Lara Schmalenstroer

Mass Spectrometry

- Analysis of proteins and peptides
- Detect, identify and quantify molecules
  - First: trace heavy isotopes
  - sequence oligonucleotides and peptides and analyze nucleotide structure

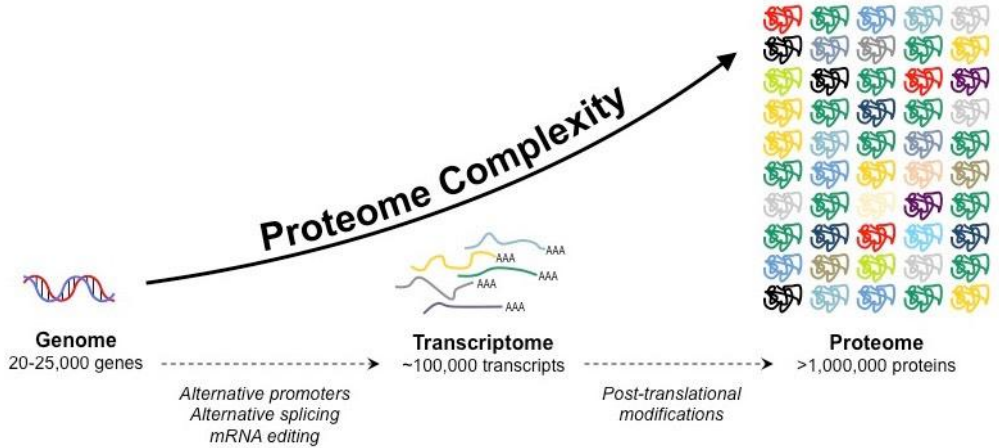


Image taken from:
https://ib.bioninja.com.au/standard-level/topic-2-molecular-biology/24-proteins/proteome.html

Fig 01: Genome - Transcriptome - Proteome

Workflow

- Digestion of protein with enzymes
  - Most common: trypsin
- Mass spectrometry
- Analysis of spectrum

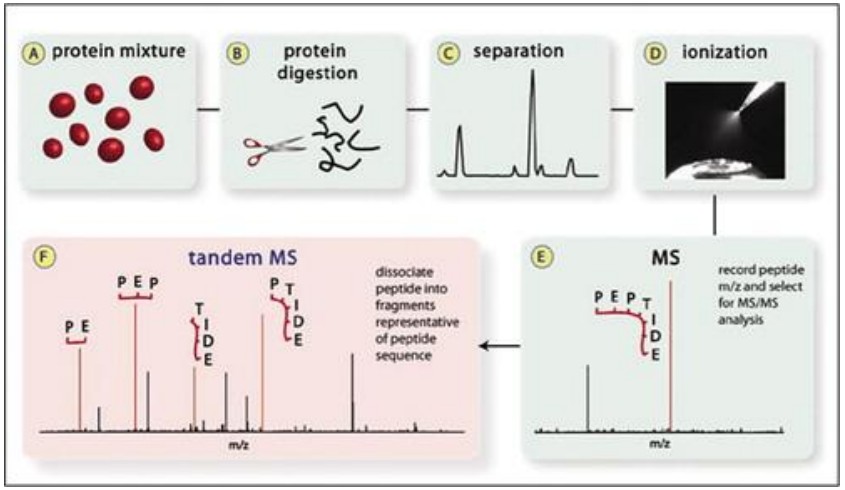


Fig 02: Mass spectrometry workflow

Image taken from: Coon, Joshua J et al., BioTechniques, 2018

Mass spectrum - ATP synthase

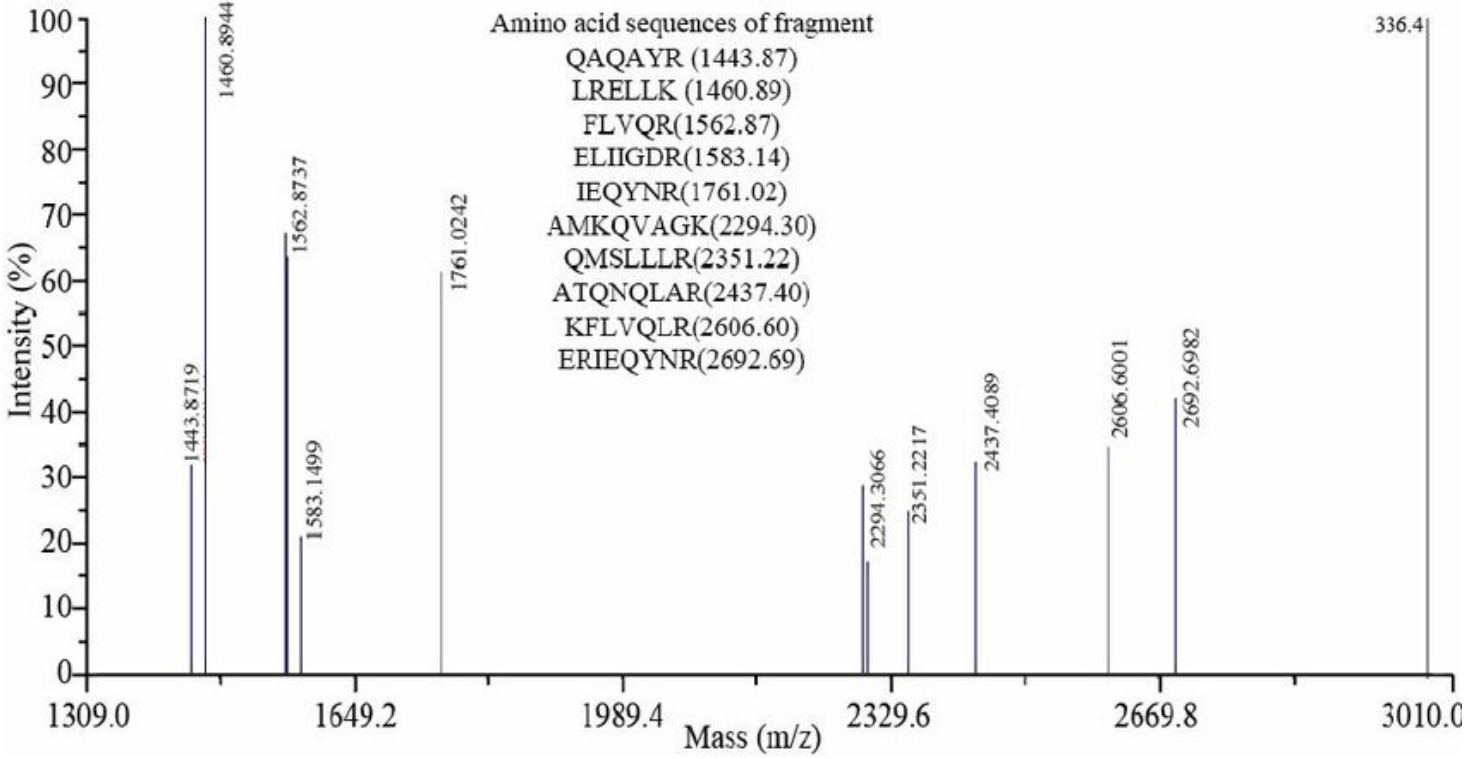


Image taken from: Ishtiaq Ch et al, Pakistan Journal of Botany, 2014

Fig 03: Mass spectrum of the α-chain of the ATP synthase

Identification of proteins

- Peptide mass fingerprinting
- Database search to identify peptide hits
  - E.g. MASCOT, Sequest, X! Tandem

MASCOT X!

Identification of proteins

- De novo sequencing
  - Mass differences
  - b and y ions

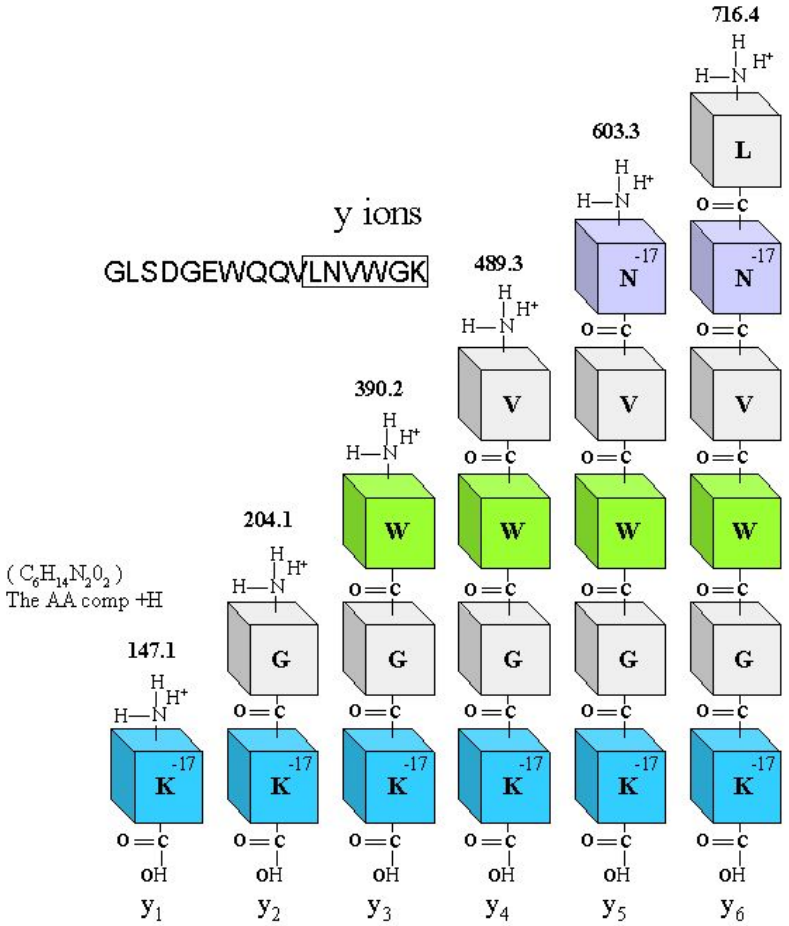


Fig 04: first six y ions of a peptide

Image taken from: https://www.ionsource.com/tutorial/DeNovo/b_and_y.htm

MS output file (.mzML)

```
<spectrum id="spectrum=1011" index="0" defaultArrayLength="467" dataProcessingRef="dp_sp_0">
        <cvParam cvRef="MS" accession="MS:1000127" name="centroid spectrum" />
        <cvParam cvRef="MS" accession="MS:1000511" name="ms level" value="1" />
        <cvParam cvRef="MS" accession="MS:1000294" name="mass spectrum" />
        <cvParam cvRef="MS" accession="MS:1000130" name="positive scan" />
                <userParam name="base peak m/z" type="xsd:double" value="391.284088134766"/>
                <userParam name="base peak intensity" type="xsd:double" value="928844.25"/>
                <userParam name="total ion current" type="xsd:double" value="6937649"/>
                <userParam name="lowest observed m/z" type="xsd:double" value="300.000828877017"/>
                <userParam name="highest observed m/z" type="xsd:double" value="2008.45845882999"/>
                <userParam name="filter string" type="xsd:string" value="FTMS + p NSI Full ms [300.00-2000.00]"/>
                <userParam name="preset scan configuration" type="xsd:string" value="1"/>
```

Methodology

Presenter : Shubhi Ambast

Tasks

01	Parse the data	• Parse mzML and mzXML files • Extract values e.g. intensity, m/z ratio, etc.
02	From data to peptides	• Compile a list of most likely peptides that spectrum represents
03	Protein prediction	• Assemble a list of proteins that peptides may represent • Use API
04	GUI	• Construct a web interface
05	Containerize the application	• Complies application into a Docker image • Bundles backend and frontend

Task 1

- Parse the raw mzML/mzXML files to extract spectrum values
- Used xml.dom.minidom to parse files
- Final output : Dataframe containing spectrum values
  - spectra_id
  - base_peak_m/z
  - base_peak_intensity
  - total_ion_current
  - lowest_observed_m/z
  - highest_observed_m/z

Task 2

https://pyopenms.readthedocs.io/en/latest/

- Used SimpleSearchEngineAlgorithm from PyOpenMS
- Compares experimental spectrum data from mzml file against theoretical data obtained from fasta file of protein sequences
- Final output: Dataframe containing obtained peptide hits and relevant information
  - Peptide hit sequence
  - Peptide hit score

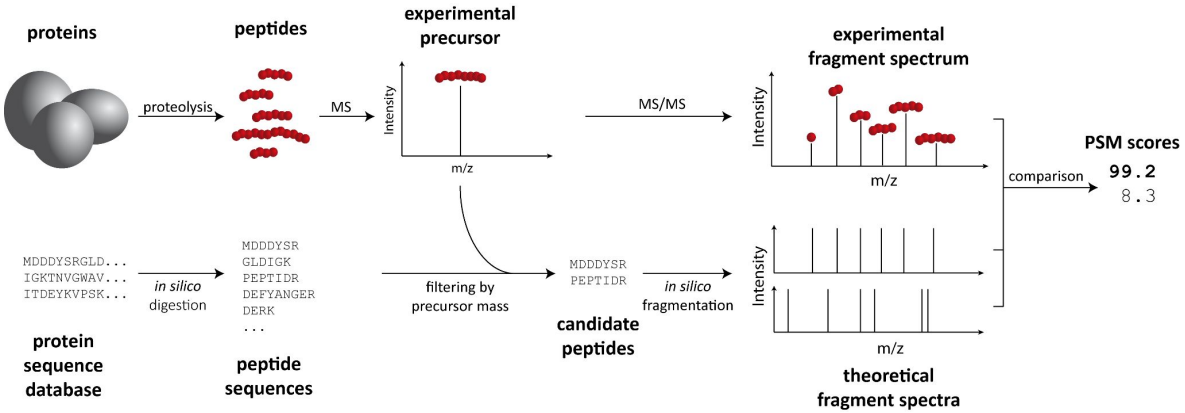


Image taken from : https://pyopenms.readthedocs.io/en/latest/peptide_search.html

Fig 05: Peptide Search

Task 3

- Used the Proteins API from EBI to map the given list of peptides
- Generates json file for each protein
- Final output: Dataframe of protein identification values
  - Peptide sequence
  - Protein_Accession
  - Taxonomy_ID
  - Data Source
  - Peptide Position

Task 4

- Web interface for entire application

- Raw mzML/mzXML files
- FASTA file

- Backend coding and HTML

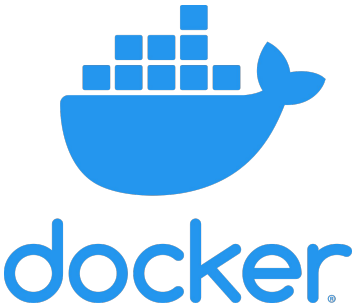
- Tables consisting of:
  - relevant spectrum values
  - peptide identification values
  - Protein identification values

- Live demonstration later

Task 5

- Created a docker image using Dockerfile
- Bundles up backend and frontend code
- Access at http://127.0.0.1:5000/

https://www.docker.com/



Software Overview

Presenter : Priya Kempanna

Important sections: Backend code

01	reader.py	• Input: mzML or mzXML file • Reading and parsing the files • Returns spectrum values
02	peptide_prediction.py	• Input: mzML and FASTA file • Prediction of possible peptides • Returns peptide information
03	protein_prediction.py	• Input: Peptide list • Mapping of peptides to proteins • Returns protein information

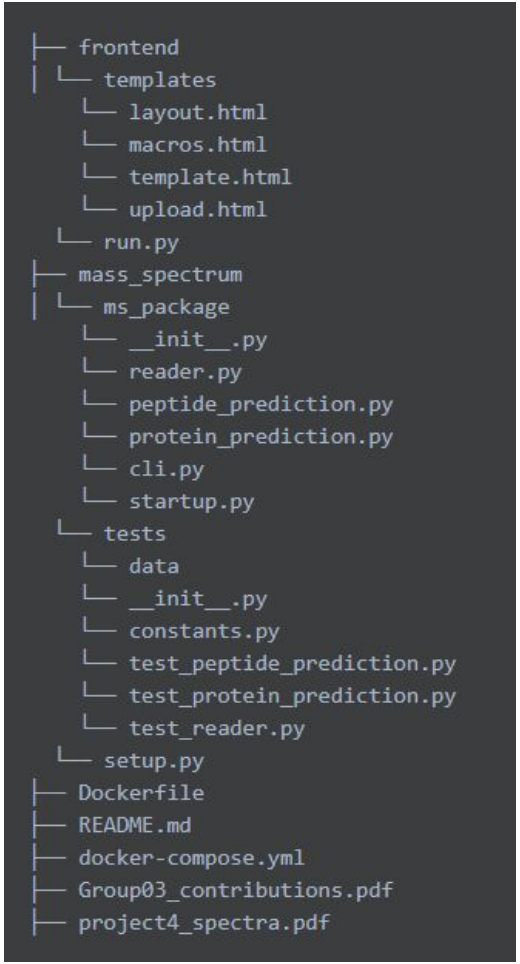


Fig 06: Skeleton of package

Important sections:

01 reader.py

- Parsing both mzML and mzXML files to get spectrum values

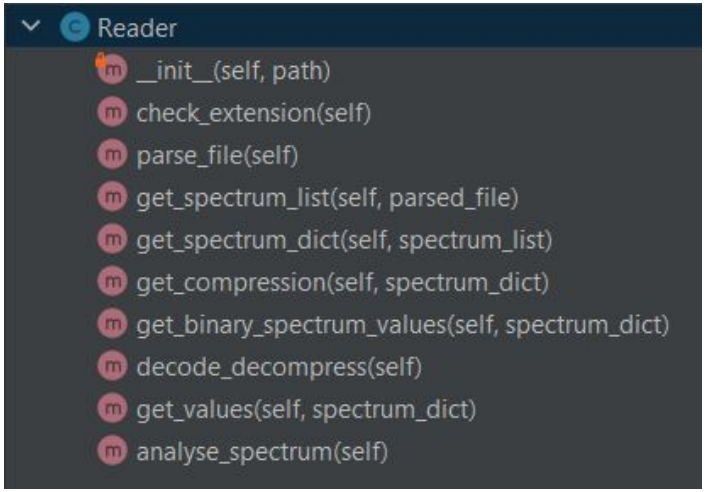


Fig 07: Structure of reader.py

```
def analyse_spectrum(self) -> pd.DataFrame:
    """..."""
    parsed_file = self.parse_file()
    s_list = self.get_spectrum_list(parsed_file)
    spectrum_dictionary = self.get_spectrum_dict(s_list)
    if self.format == 'mzml':
        self.get_compression(spectrum_dictionary)
        self.get_binary_spectrum_values(spectrum_dictionary)
        self.decode_decompress()
    values_spectrum = self.get_values(spectrum_dictionary)
    df_values = pd.DataFrame.from_dict(values_spectrum, orient='index',
```

Fig 08: Wrapper function of reader.py

→ Check extension

→ Minidom usage to get values

→ Extraction of binary values - attributes

Important sections:

02 peptide_prediction.py

- Need two input files:
  - mzML and FASTA

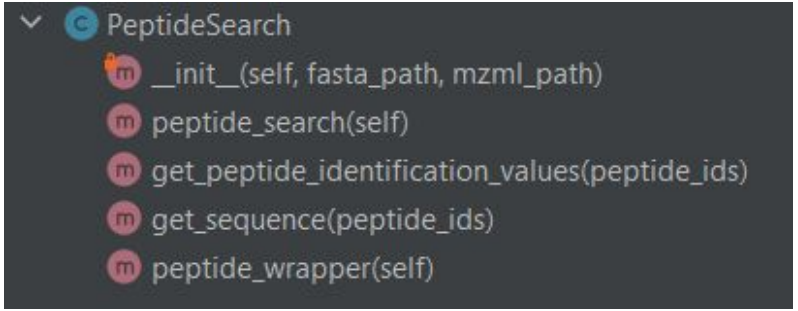


Fig 09: Structure of peptide_prediction.py

```
def peptide_search(self) -> tuple[list, list]:
    """..."""
    protein_ids = list()
    peptide_ids = list()

    assert self.mzml_path.endswith('.mzML')
    assert self.fasta_path.endswith('.fasta')

    if not self.mzml_path and self.fasta_path:
        logger.error('Input files are invalid')
    else:
        SimpleSearchEngineAlgorithm().search(self.mzml_path, self.fasta_path, protein_ids, peptide_ids)
        logger.info('mzml file and fasta file exists')
        return protein_ids, peptide_ids
```

Inputs

Outputs

Fig 10: Use of pyopenms in peptide_prediction.py

➡ Check extension

➡ Calling pyopenms to search

Important sections:

03 protein_prediction.py

- The Proteins API query:
  - Takes 20 input peptide sequence at once
  - Returns json format
- If previously queried, data returned from cache

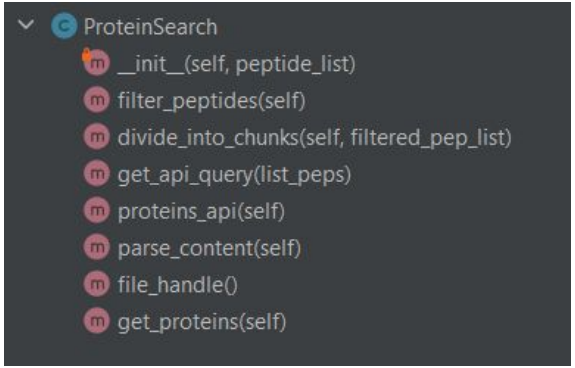


Fig 11: Structure of protein_prediction.py

```python
def proteins_api(self):
    """..."""

    logger.info("...")

    for pep_lil in self.sel_peptides:
        api_query = self.get_api_query(pep_lil)
        requestURL = f"https://www.ebi.ac.uk/proteins/api/proteomics?offset=0&size=100&peptide={api_query}"
        r = requests.get(requestURL, headers={"Accept": "application/json"})

        if not r.ok:...

        self.protein_response += r.text
```

API query format

Fig 12: API query in protein_prediction.py

Important sections:

cli.py

- Three main commands to access from command line
- ‘Click’ Python package was used
- Options to print data on STDOUT and/or to download in specified directory

```
- ms_package get-spectrum-values /tests/data/BSA1.mzML -v

- ms_package peptide-info /tests/data/BSA.fasta /tests/data/BSA1.mzML -v

- ms_package protein-info -f /tests/data/BSA.fasta -m /tests/data/BSA1.mzML -v
```

Fig 13: Usage of command line

Important sections: Tests

- Tests for every python script to test every method
- Pytest framework was used
- Tests checked and passed
- Re-validation using ‘coverage’ - 91% of lines covered

83% files, 91% lines covered in 'ms_package'

Element	Statistics, %
__init__.py	100% lines covered
cli.py	not covered
peptide_prediction.py	96% lines covered
protein_prediction.py	93% lines covered
reader.py	87% lines covered
startup.py	100% lines covered

Fig 14: Unit test coverage over python files

Obstacles:

- Research on new field
  - Understanding the concept
  - Dealing with new data file types
  - Various methodologies available
- Going overboard - knowing when to stop
  - Time and scope of this project into consideration to limit ourselves
  - Keeping the tasks balanced
  - Made use of external package to deal with peptide prediction

Discussion

Presenter : Rohitha Ravinder

ENVIRONMENTAL ANALYSIS

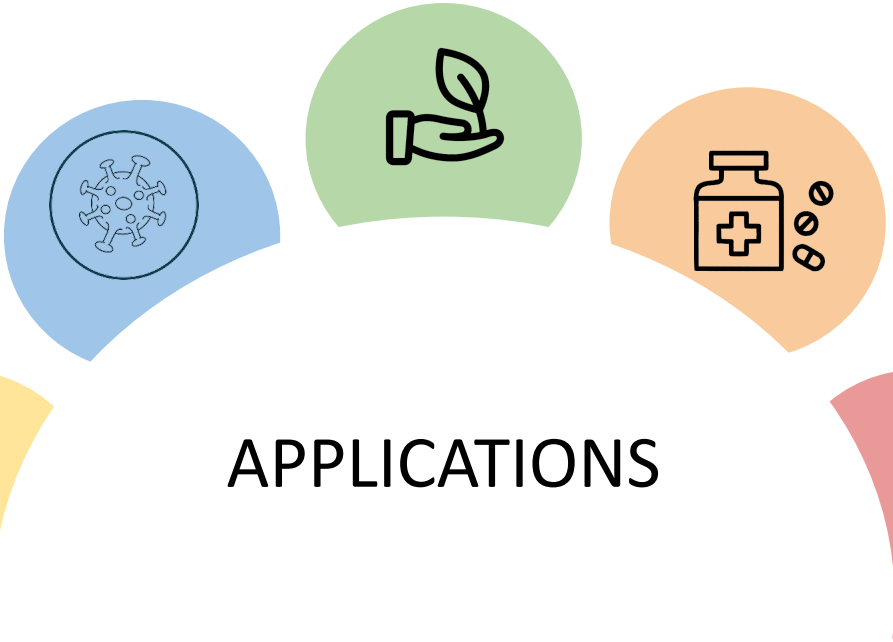
Drinking water testing, soil contamination assessment, carbon dioxide and pollution monitoring

METABOLOMICS

Cancer screening and diagnosis,metabolic disorder profiling

PHARMACEUTICAL ANALYSIS

Drug discovery and absorption



APPLICATIONS

PROTEOMICS

Characterization of proteins, identification of post translational modifications

FORENSICS

Analysis of trace evidence


DEMO

Possible extensions for current package...

Mass Spectrum 2.0

- Use of databases to store retrieved protein files
- Avoid dependency on an external package
- Expand usage to other MS input files
- Improvise GUI


THANK
YOU !