ABSTRACT

Large scale proteomics datasets available in public repositories such as, PRIDE, jPOST etc., provides a great opportunity to reuse, reanalysis of these datasets to gain novel insights in biological systems. Proteomics studies or datasets contains metadata, which is used for downstream data analysis to find differential protein abundance or to find novel proteins using various proteomics tools. Currently, we have metadata available which lack clear and detailed annotation of experiments. Towards it, Kutum R et.al is developing a database of metadata for proteomics datasets, which will contain all metadata available in scrater in various databases. In this project, I have contributed and developed a web scraping application in python for acquiring metadata from jPOST database/repository.

INTRODUCTION

Introduction to proteomics:

Proteomics is the study of structure and function of the complete set of proteins present in a cell, organ or organism at a given time [1]. It can be used for study of protein-protein interaction, identification of posttranslational modification and analysis of two or more proteins.We are in the golden era of proteomics, where mass spectrometry (MS) based proteomics are publicly available in many repositories. In 2011, ProteomeXchange (PX) consortium [2] has adopted a unified framework to enable rapid dissemination of proteomics datasets existing in different repositories (PRIDE [3], MassIVE [4], PeptideAtlas [4], jPOST [5] and iProX [6]). PX follows the Minimum Information about Proteomics Experiment (MIAPE) guidelines [7] for the metadata; it provides only minimal information such as the title, description, species, modification, instrument, keywords and links to the repository etc.

Introduction to proteomics metadata:

Proteomics metadata is information about experimental context and analysis tools used for determination of proteins. Datasets from more than 900 different taxonomic identifiers are available in the PX repositories. Since proteomics dataset is partially understood, there is a great opportunity for orthogonal reuse of public data [8].

Introduction to ProMetaDB:

Proteomics Metadata Database contains curated meta information(phenotypes, conditions/concentrations and their associated raw/processed files) for all existing proteomics studies[8,9].Retrieval of metadata has been done from existing resources like PeptideAtlas [4], jPOST [5], PRIDE [3], MassIVE [4]. ProMetaDB will enable researchers to perform easy and rapid acquisition of existing datasets for discovering novel peptides/proteins.

Introduction to jPOST:

Proteomics database jPOST (Japan ProteOme STandard Repository/Database) was developed in 2015 to integrate proteome datasets generated from multiple projects and institutions [5]. JPOSTrepo has unique features such as high-speed file upload system and user-friendly interface with open-source libraries; all of the submission operations can be completed within a web browser.

Introduction to web scraping:

Web scraping is the technique for extracting web elements ranging from simple extraction robots to online meta-servers. The focus is to set a data scraping pipeline, with minimal programming effort and answer number of practical needs [10].

OBJECTIVES

1. To develop a web scraping application to retrieval proteomics metadata from jPOST database.

MATERIAL AND METHODS

Programming Language

- **Python 3.7**
  Python 3.7.3 is the third maintenance release of python 3.7 with several features such as data classes, improved deprecationWarning handling, built-in() breakpoint,context variables and python documentation translation [11].

Python modules used

- **Selenium**
  Selenium Python bindings provide a simple API to write functional/acceptance tests using selenium WebDriver.Through Selenium Python API we can access all functionalities of Selenium WebDriver such as Firefox, Chrome etc. in an intuitive way [12].
- **lxml**

- lxml is a Pythonic, mature binding for libxml2 and libxslt libraries. It provides safe and convenient access to these libraries using ElementTree API. It extends ElementTree API significantly to offer support for XPath, XML schema, XSLT and much more [13].

Installation or setup environment in Windows

i. Install Anaconda.
   - Select the path and go to advanced setting of PC and open environmental variables.select the path and click on edit button to set environmental variable.For e.g. "C:\Users\Aagam\Anaconda3;C:\Users\Aagam\Anaconda3\Scripts; C:\Users\Aagam\Anaconda3\Library\bin"

ii. Install selenium
   - Open Anaconda prompt and type the command "python3 -m pip install --upgrade pip" to install pip of latest version. After that type the command "pip install selenium or conda install selenium" to install selenium webdriver.

iii. Install Geckodriver
   - For installation of Geckodriver, first download the latest version for windows 64 bit and then unzip the file by clicking on extract here.After that copy the file in some new folder let's say Geckodriver. Copy the path and paste it in environmental variable.

After that restarts the computer, the error for geckodriver for executable path will be removed.

iv. We have used in spyder IDE (Interactive Developer Environment) to write our python code.

Workflow or approach:

1. We first create a link to open URL through selenium webdriver. After that we inspect the clickable buttons from 1 to 14 using for loop and xpath.
2. For extracting table from each page, **import lxml.html** and define the function let's say "extract_tb". Again inspect the table element and extract particular element from page_content using ***getchildren()*** function.
3. Apply for loop and a condition to extract "href ID" present in the even position and append it to an empty list. Now **import pickle**, which basically has two functions, first to dump object to a file object and second one is load, which loads an object to a file object. We have used it for storing and retrieving our data.
4. Create a new file and extract information present inside the xml file.For that, we will run for loop and apply condition if element gets **"XML file"**, extract that href element. Again **import pickle** to dump file object.
5. Create a new file and **import urllib** and **os** module. Run for loop for reading and writing data from file and extract metadata for each project.

ALGORITHM or PSEUDOCODE:

1. Extract hyperlink of each jPOST project from the table of contents as shown in **Figure-01.**
2. For each jPOST project page, using the above hyperlink download the project page content as shown in **Figure-02**.
3. Extracted xml hyperlink from each jPOST project page and downloaded the xml file. From the xml file we have extracted the metadata of the given jPOST project (**Figure-03**).

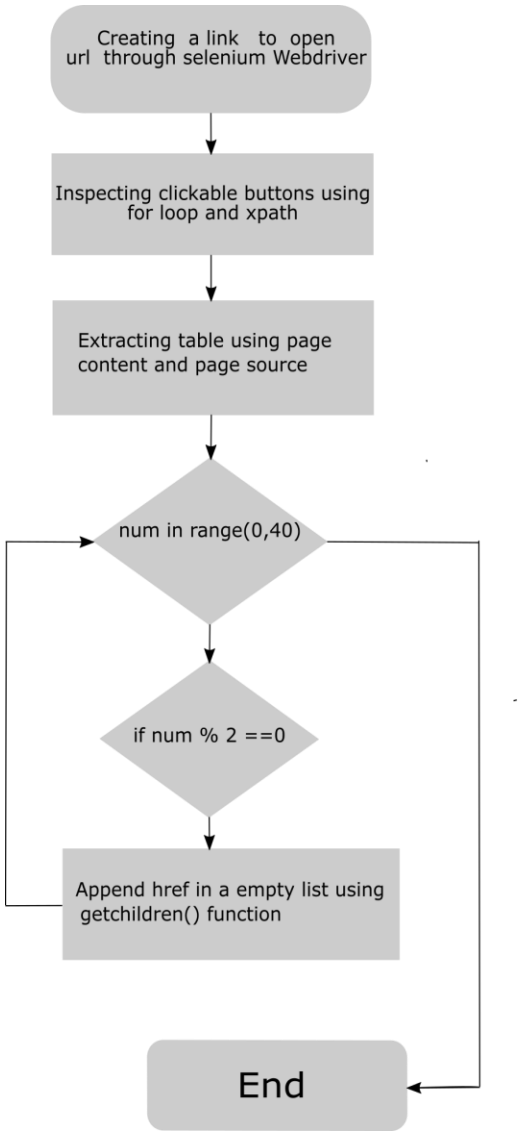

Figure 01: Flowcharts representing extraction of table_contents and href_ids

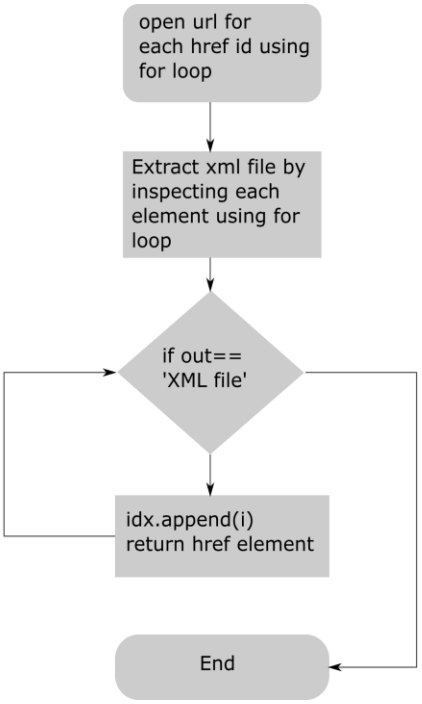

Figure-02: Flowchart representing extraction of xml files

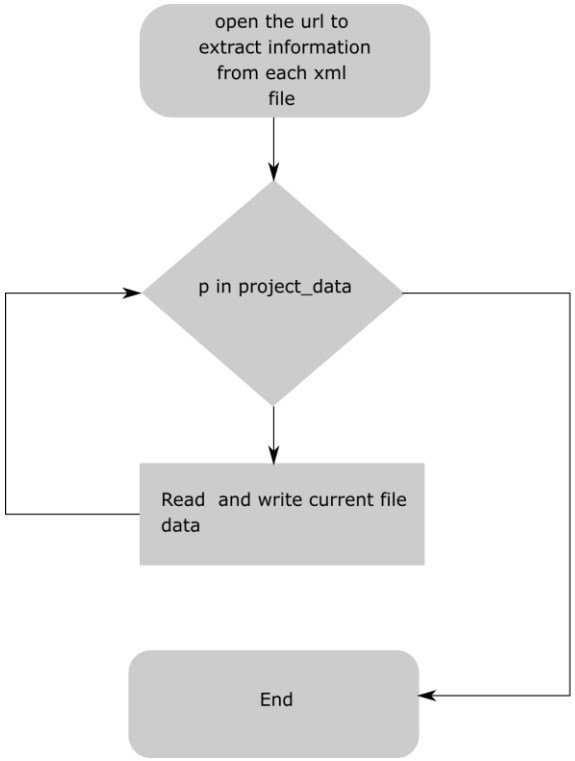

Figure-03: Flowchart representing the reading and writing from xml ids.

RESULTS AND DISCUSSION

Through python programming language [11], we first inspected buttons present on the database to extract each page using ***Selenium*** module. After that we extracted table elements from each page to extract href_ids of each jPOST project page and extract xml file that contains the metadata of the project. We created various functions for extraction of information such as species, instruments, modification, sample-ids etc. With the help of the R statistical programming language [14], we generated the bar-graph (**Figure-01**) representing the proteomics data for top 10 species in jPOST database using **ggplot2** package [14]. The graph shows that *Homo sapiens* have the highest frequency of data i.e. 92 in the database accompanied by *Mus musculus* (n=20). The detail code for the web scraping application is available in GitHub repository

(https://github.com/rintukutum/trainee-jpost).

(For further details refer to **Table-01** in APPENDIX)

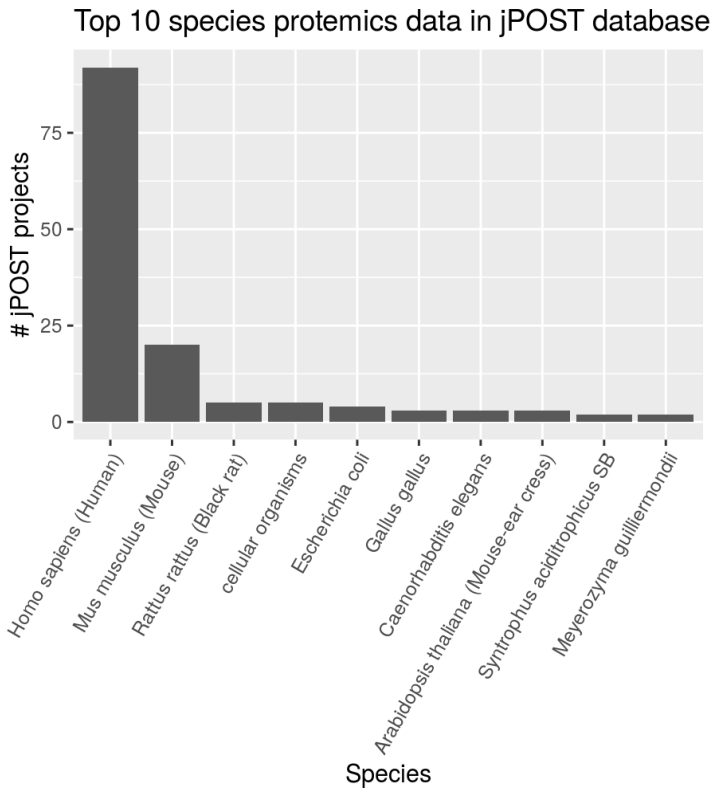

Figure-04: Bar-graph representing the number of proteomics dataset of top 10 species present in jPOST database.

CONCLUSION

There is a lot of meta information which is generated in a proteomics experiment and stored in various public databases or repositories. Acquisition of these metadata is very difficult to handle manually. Here, I have developed a web scraping python application to retrieve proteomics metadata from jPOST database/repository. In future, I will write an additional application in python which will update metadata according to jPOST database is updated.

REFERENCES

1. [PMID-20948568] Kondethimmanahalli Chandramoulli and Pei-Yuan Quian (2009) Proteomics: Challenges, Techniques and Possibilities to Overcome Biological Sample Complexity.
2. [PMID-27924013] Deutsch EW, Csordas A, Sun Z, The ProteomeXchange consortium in 2017 supporting the cultural change in proteomics public data deposition.
3. [PMID-30395289] Perez-Reverol, Csordas A, Bai J, The Pride database and related tools and Resources in 2019: improving support for quantification data.
4. [PMID-25158685] Yasset Perez-Reverol, Emanuele Alpi, Rui Wang (2015) Making Proteomics Data accessible and reusable: Current state of proteomics databases and repositories.
5. [PMID-6324006] Yuki Moriya, Shin Kawano, Shjiro Okuda (2018); The JPOST environment: an integrated proteomics data repository and database.
6. [PMID-30252092] Ma J, Chen T, Wu S (2019) iProX: an integrated integrated proteome resource.
7. [PMID-17687369]Taylor CF1, Paton NW, Lilley KS, Binz PA (2007) The minimum information about a proteomics experiment (MIAPE).
8. [PMID-28118949] Lennart Martens and Juan Antonio Vizcaino (2017); A Golden Age For Working With Public Proteomics Data.
9. ProMETAdb: A unified database of proteomics metadata. Rintu Kutum, Saikat Bandyopadhyay, Debasis Dash http://prometadb.igib.res.in/
10. [PMID-23632294] Glez-Pena D, Lourenco A, Lopez-Fernandez H, Reboiro-Jato M,Fdez-Riverola F(2014); Web Scraping Technologies in an API World.
11. Python https://www.python.org/
12. Selenium with Python https://selenium-python.readthedocs.io/
13. lxml - XML and HTML with Python https://lxml.de/
14. The R Project for Statistical Computing https://www.r-project.org/

APPENDIX

Table-01: REPRESENTATION OF FREQUENCY OF SPECIES IN jPOST DATABASE

S.no	Species	Freq
1	Homo sapiens (Human)	92
2	Mus musculus (Mouse)	20
3	Rattus rattus (Black rat)	5
4	cellular organisms	5
5	Escherichia coli	4
6	Gallus gallus	3
7	Caenorhabditis elegans	3
8	Arabidopsis thaliana (Mouse-ear cress)	3
9	Syntrophus aciditrophicus SB	2
10	Meyerozyma guilliermondii	2
11	Bos taurus	2
12	Zea mays	1
13	Strongyloides venezuelensis	1
14	Staphylococcus aureus	1
15	Solanum lycopersicum	1
16	Serratia marcescens	1
17	Oryza sativa	1

18	none	1
19	Natrialba magadii ATCC 43099	1
20	Methanococcus maripaludis	1
21	Malassezia sympodialis	1
22	Honeybee (Apis mellifera L.)	1
23	Helicobacter pylori F16	1
24	Haloferax volcanii DS2	1
25	Glycine max	1
26	Cyanidioschyzon merolae strain 10D	1
27	Crocodylus siamensis	1
28	Clostridium cellulovorans 743B	1
29	Ciona intestinalis Type A	1
30	Andrographis paniculata	1

00-get-hrefs-jpost.py

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
#---This program involves the recursive working of clickable button and extraction of href_id from
table_contents---#

import time
from selenium import webdriver
import lxml.html as html
# from selenium.webdriver.common.Keys import Keys
#-----------
#Aagam Mishra
#-----------
def extract_tb(page_source):
    page_content=html.document_fromstring(page_source)
    table =
page_content.getchildren()[1].getchildren()[13].getchildren()[2].getchildren()[4].getchildren()[8].getchildren
()[1].getchildren()
    href = []
    for number in range(0,40):
        if number % 2 == 0:
            href.append(table[number].getchildren()[0].getchildren()[0].get('href'))
    return href


driver = webdriver.Firefox()
time.sleep(5)
jpost_href = []
for i in range(1,15):
   if i==1:
      driver.get('https://repository.jpostdb.org/')
      time.sleep(4)
      page_source=driver.page_source
      jpost_href.append(extract_tb(page_source))
   else:
      driver.find_element_by_xpath("//a[@data-page=%d]" % (i)).click() #using xpath to find location of
particular button
      page_source=driver.page_source
      jpost_href.append(extract_tb(page_source))

hrefs=[]
for i in jpost_href:
    for j in i:
        hrefs.append(j)
```

```
import pickle
output = open('hrefs-jPOST.pickle','wb')
pickle.dump(hrefs,output)
output.close()
```

01-hrefs-to-project-details.py

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jun 24 20:21:03 2019
@author: Aaga
"""
#---This program involves the recursive working of href_id and extraction of xml file from each id----#

import time
from selenium import webdriver

import pickle
import lxml.html as html

hrefs = open('hrefs-jPOST.pickle', 'rb')
hrefs_data = pickle.load(hrefs)
hrefs.close()

driver = webdriver.Firefox()
page_contents = []
for href in hrefs_data:
    webpage = 'https://repository.jpostdb.org/' + href
    driver.get(webpage)
    time.sleep(4)
    page_source=driver.page_source
    page_content=html.document_fromstring(page_source)
    page_contents.append(page_content)
     #xmls.append(t.getchildren()[0].get('href'))


def extract_href(x):
    tb =
x.getchildren()[1].getchildren()[13].getchildren()[2].getchildren()[0].getchildren()[3].getchildren()[0].getchild
ren()
    #to find xml file
    idx = []
    i = 0
    for element in tb:
```

```
        out = element.getchildren()[0].text
        if out == 'XML file':
            idx.append(i)
        i = i + 1
    # href element
    output = tb[idx[0]].getchildren()[1].getchildren()[0].get('href')
    return output


i = 0
xmls = []
for page in page_contents:
    print(i)
    xmls.append(extract_href(page))
    i  = i + 1



import pickle
output2=open('hrefs-to-project-details.pickle','wb')
pickle.dump(xmls,output2)
output2.close()
```

02-Project-Details.py

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jun 24 22:08:03 2019
@author: Aaga
"""
#---This program involves recursive working of xml file as well as extraction of data ---#

import urllib
import pickle

project = open('hrefs-to-project-details.pickle', 'rb')
hrefs = pickle.load(project)
project.close()

import os
#os.mkdir('./xml')
i = 1
for p in project_data:
    file=urllib.request.urlopen('https://repository.jpostdb.org' + p)
    data=file.read()
```

```
    file.close()
    output =  '.' + p
    print(i,'=',output)
    i = i + 1
    filehandle = open(output,'wb')
    filehandle.write(data)
    filehandle.close()
--------------------------------------------------
```

03-convert-xml-to-metadata.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Wed Jun 26 12:28:01 2019

@author: rintu and aagam
"""

# for reading and locating xml files
import os

from xml.dom import minidom
# reference
# https://docs.python.org/3/library/xml.dom.minidom.html

xml_files = []
for (dirpath, dirname, filenames) in os.walk("./xml"):
    for filename in filenames:
        xml_file = ''.join([dirpath,'/',filename])
        print(xml_file)
        xml_files.append(xml_file)


doc = minidom.parse(xml_files[1])

def extract_id_value_with_Species(doc):
    tagName = 'Species'
    y = doc.getElementsByTagName(tagName)[0].childNodes[1]
    tagName_id=y.attributes.get('id').value
    tagName_val=y.attributes.get('value').value
    tagName_out={'id': tagName_id, 'value': tagName_val}
    return tagName_out


species = []
for xml in xml_files:
    doc = minidom.parse(xml)
```

```
    species.append(extract_id_value_with_Species(doc))


# os.mkdir('./data')
output = []
output.append("ID,Species Name")
for entry in species:
    output.append(entry['id'].encode('utf8') + ',' + entry['value'].encode('utf8'))

fileout = open('./data/jPOST-species-info.csv','w')
fileout.write('\n'.join(output))
fileout.close()
# Modification
```

04-viz-data.R

```
rm(list=ls())
data = read.csv('./data/jPOST-species-info.csv')

tb_jPOST = data.frame(rev(sort(table(data$Species.Name))))
colnames(tb_jPOST)[1] <- 'Species'

write.csv(
    tb_jPOST,
    './data/jPOST-data-statistics-24-06-19.csv',
    row.names = FALSE)

library(ggplot2)
p = ggplot(tb_jPOST[1:10,], aes(x = Species, y = Freq)) +
    geom_bar(stat='identity') +
    ylab('# jPOST projects') +
    ggtitle('Top 10 species proteomics data in jPOST database') +
    theme(axis.text.x = element_text(angle=60,vjust=1,hjust=1))

png('./figures/Figure-xx.png',width=900,height=1000,res=200)
p
dev.off()
```