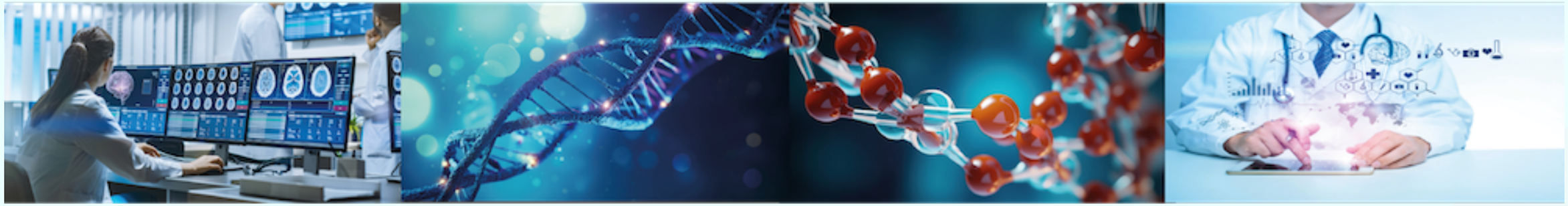


Programming for Biomedical Informatics

Lecture 7 - Biomedical Evidence

https://github.com/tisimpson/pbi

Ian Simpson
ian.simpson@ed.ac.uk

THE UNIVERSITY of EDINBURGH
informatics

eUtils Mapping Review

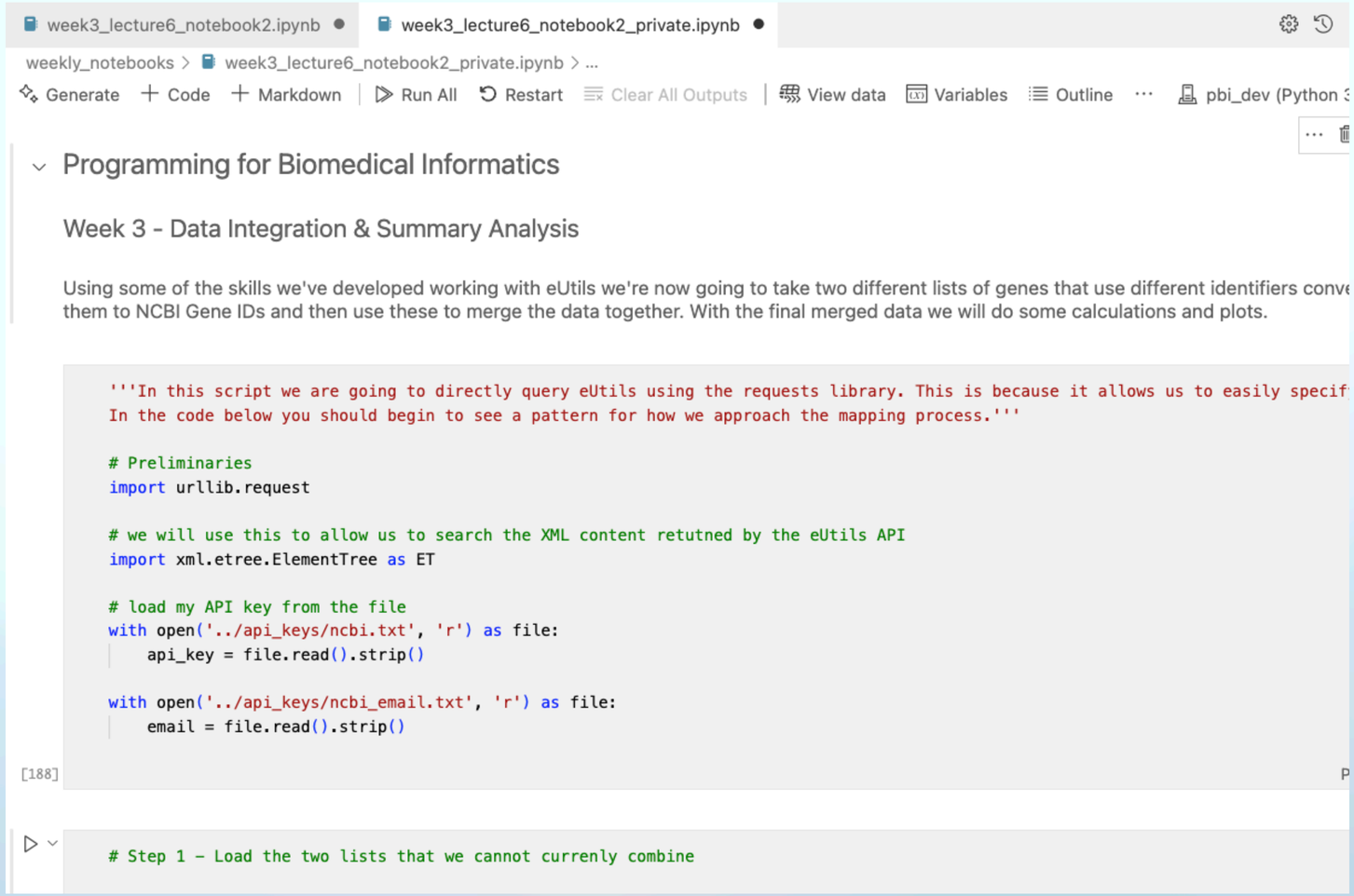
- We used a selection of combinations of eUtils tools to recover information that allows for mapping between sources
- eSearch - can take mixed queries, typically terms or phrases e.g. gene symbols “Pax6”
- eSummary - can take identifiers returned by eSearch to recover basic summary information. This is often what you need as it includes internal numerical NCBI identifiers and basic meta-data (descriptions, names...)
- eLink - can be used to map between NCBI databases e.g. from “nucleotide” to “gene”. It returns a LinkSet in which individual Link items contain the cross-mapped information (NM_12345 -> GeneID:678)
- eFetch - can be used to pull full entries. These are typically extremely long and complex records with deeply nested elements and/or long lists of features - there are options to filter these

eUtils Mapping Review

- We can use the BioPython Entrez module for more accessible use of eUtils and it can be extended to more complex implementations
- Once familiar with the basic function of the eUtils it is more flexible to begin using the requests library to make API calls using URLs that you have built
- This makes it easier to take advantage of eUtils features like “WebEnv & QueryKey” and to serialise large requests that are much more difficult to retrieve efficiently through other means
- Key to the requests approach is an understanding of how you parse XML/JSON response texts in Python to access the features you are looking for.
- Typically this requires that you first print out an XML response for a small query so that you can correctly format your XML element search.
- Once this has been perfected you can readily scale up your analysis efficiently.

eUtils Mapping Review

- We will now go briefly through a completed script from last week's example using this strategy.



Biomedical Evidence Gathering

- Careful use of source databases can help you to build evidence to support and interpret experimental results
- We have focussed on NCBI and (to a lesser extent) Ensembl as the pre-eminent multi-modal data sources especially for molecular data.
- We have looked at a few more bespoke sources such as GenCC and ontologies (via the Bioportal) where we have had to use bulk download and APIs to gather the data and then integrate it with other sources.
- These approaches will be used as the backbone for the applied cases we follow in the second half of the course
- The final core resource of information in biomedical research is the published literature. Whilst increasing numbers of studies are being released on pre-print servers such as bioRxiv, arXiv, and medRxiv the central source of primarily peer-reviewed studies is NCBI-PubMed

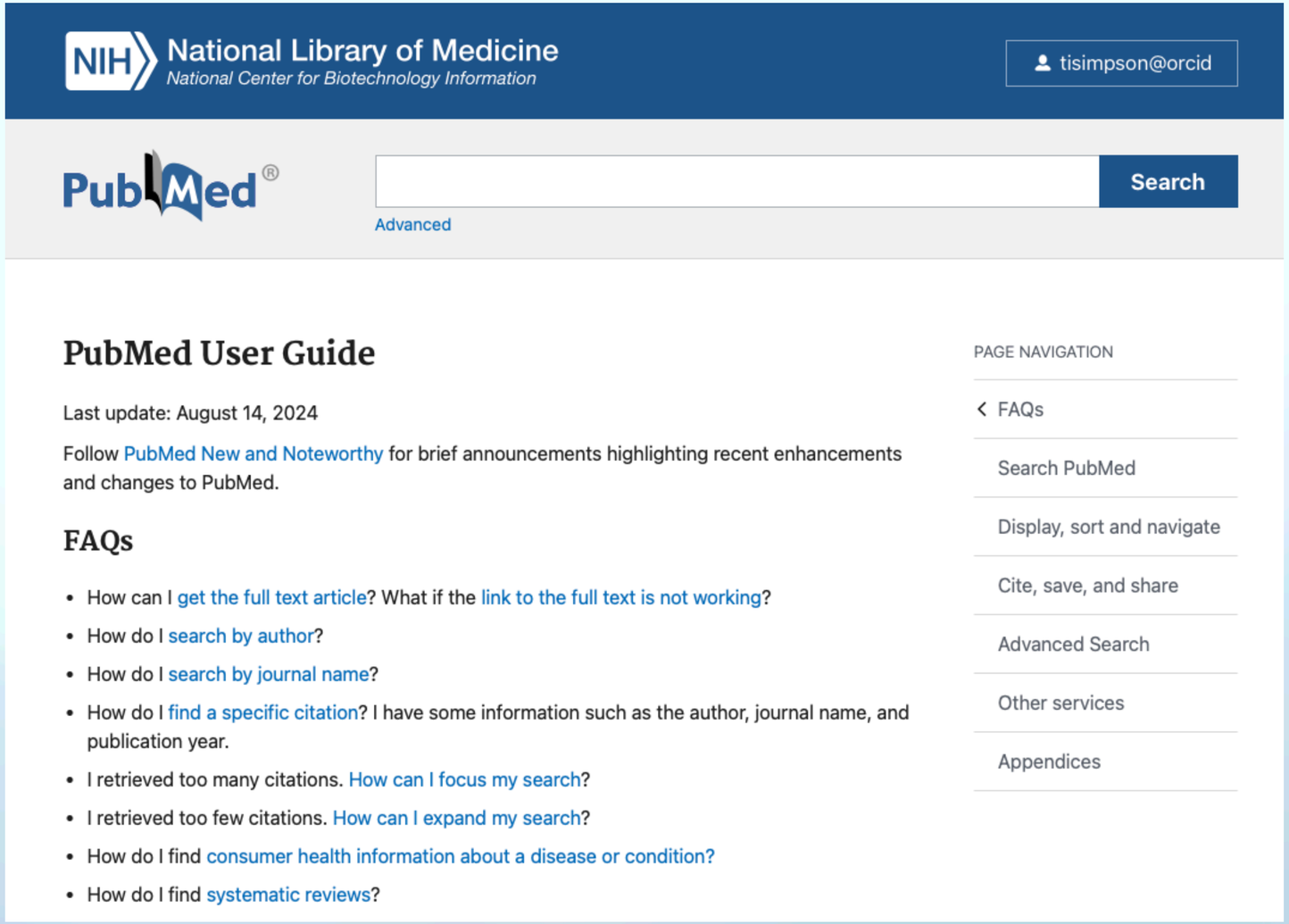
PubMed

https://pubmed.ncbi.nlm.nih.gov/

- PubMed contains >37 million articles
- It is fully integrated into the eUtils family so we can use the tools we've been learning about over the last few weeks to query PubMed
- We should first look at how best to search PubMed - a good strategy is to practice performing searches online and view the full search query that PubMed executes for you (this is more complicated than it sounds)
- Once you've perfected your search you can implement it in code and retrieve summary or full detail records from PubMed and use these for downstream analysis and/or further evidence building (for example building a paper corpus for a domain/topic)
- We will explore this in detail in the coding session this week

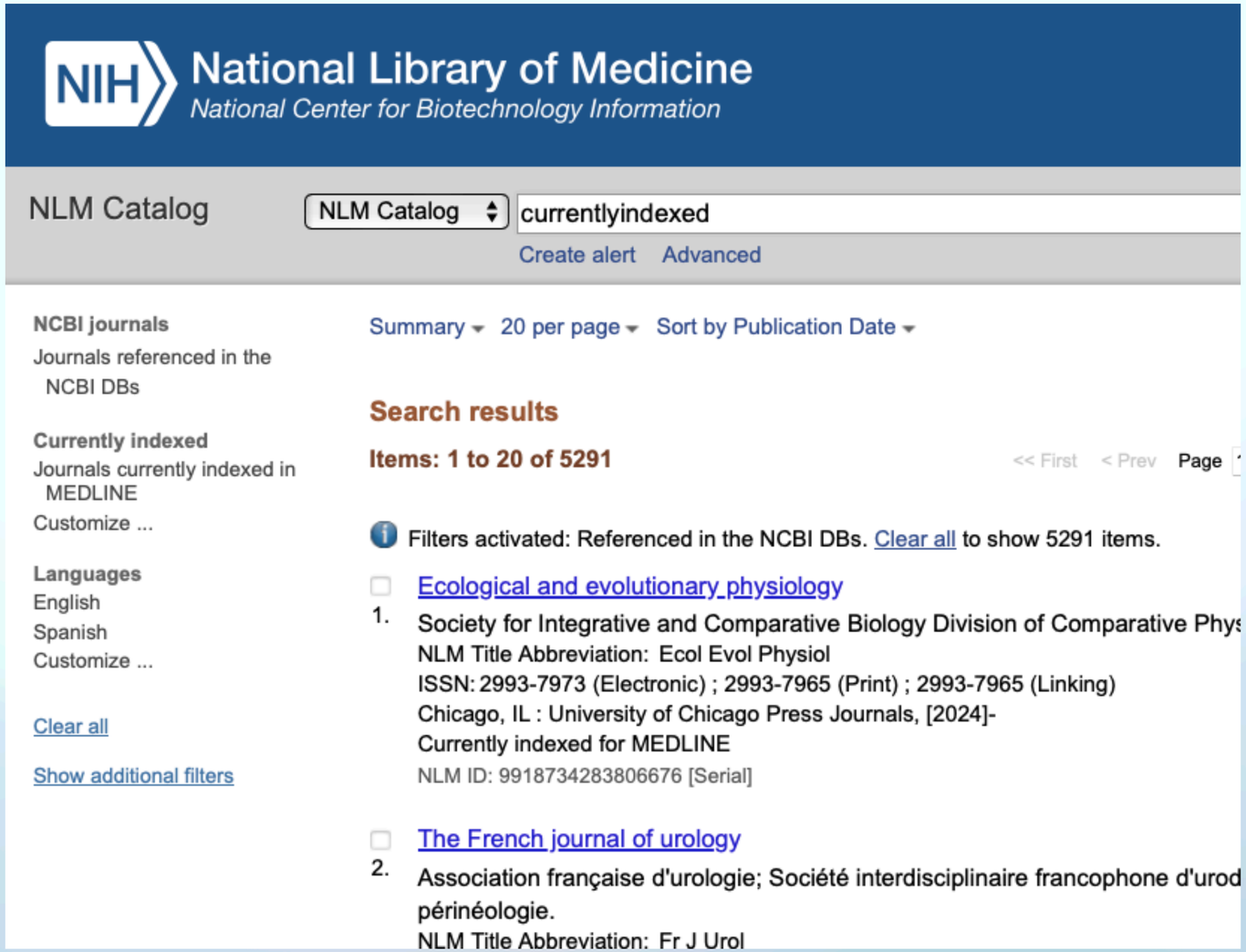
PubMed - FAQ & Help

https://pubmed.ncbi.nlm.nih.gov/help/



PubMed - Journal Lookup

https://www.ncbi.nlm.nih.gov/nlmcatalog?term=currentlyindexed



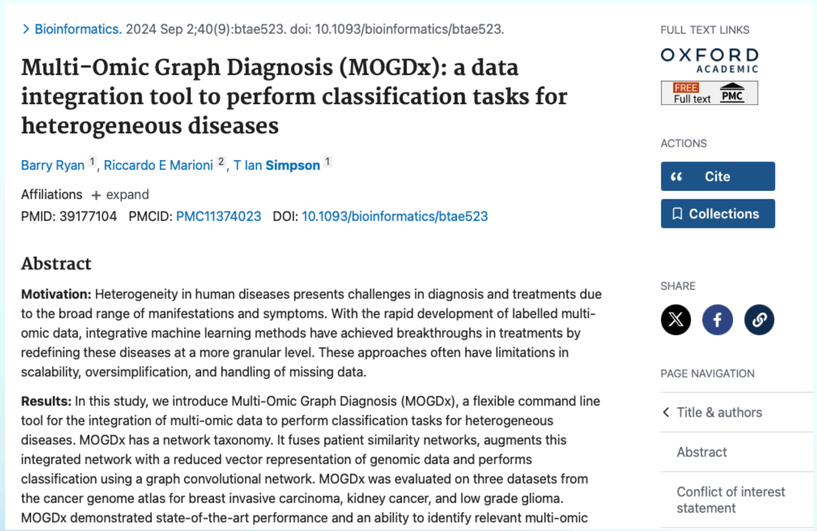
PubMed - Search Field Tags

https://pubmed.ncbi.nlm.nih.gov/help/#using-search-field-tags

Search field tags

Affiliation [ad]
All Fields [all]
Article Identifier [aid]
Author [au]
Author Identifier [auid]
Book [book]
Comment Correction Type
Completion Date [dcom]
Conflict of Interest Statement [cois]
Corporate Author [cn]
Create Date [crdt]
EC/RN Number [rn]
Editor [ed]
Entry Date [edat]
Filter [filter] [sb]
First Author Name [1au]
Full Author Name [fau]
Full Investigator Name [fir]
Grants and Funding [gr]
Investigator [ir]
ISBN [isbn]
Issue [ip]
Journal [ta]
Language [la]
Last Author Name [lastau]
Location ID [lid]
MeSH Date [mhda]
MeSH Major Topic [majr]
MeSH Subheadings [sh]
MeSH Terms [mh]
Modification Date [lr]
NLM Unique ID [jid]
Other Term [ot]
Owner
Pagination [pg]
Personal Name as Subject [ps]
Pharmacological Action [pa]
Place of Publication [pl]
PMCID and MID
PMID [pmid]
Publication Date [dp]
Publication Type [pt]
Publisher [pubn]
Secondary Source ID [si]
Subset [sb]
Supplementary Concept [nm]
Text Words [tw]
Title [ti]
Title/Abstract [tiab]
Transliterated Title [tt]
Volume [vi]

PubMed - Medline Format



> Bioinformatics. 2024 Sep 2;40(9):btae523. doi: 10.1093/bioinformatics/btae523.

Multi-Omic Graph Diagnosis (MOGDx): a data integration tool to perform classification tasks for heterogeneous diseases

Barry Ryan [1], Riccardo E Marioni [2], T Ian **Simpson** [1]

Affiliations + expand

PMID: 39177104 PMCID: PMC11374023 DOI: 10.1093/bioinformatics/btae523

FULL TEXT LINKS

OXFORD ACADEMIC

FREE Full text PMC

ACTIONS

Cite

Collections

SHARE

PAGE NAVIGATION

Title & authors

Abstract

Conflict of interest statement

Abstract

Motivation: Heterogeneity in human diseases presents challenges in diagnosis and treatments due to the broad range of manifestations and symptoms. With the rapid development of labelled multi-omic data, integrative machine learning methods have achieved breakthroughs in treatments by redefining these diseases at a more granular level. These approaches often have limitations in scalability, oversimplification, and handling of missing data.

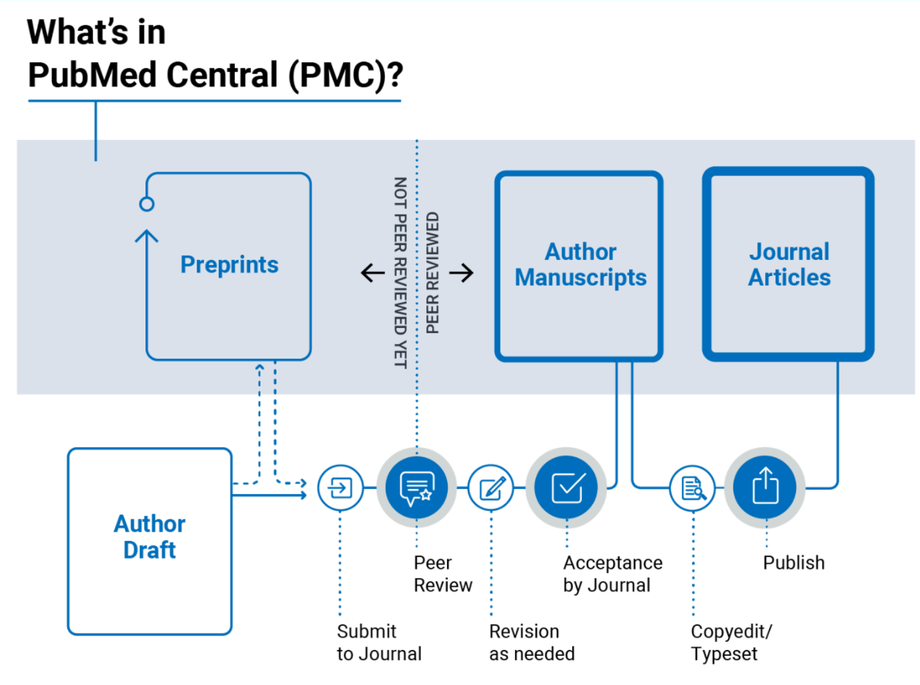
Results: In this study, we introduce Multi-Omic Graph Diagnosis (MOGDx), a flexible command line tool for the integration of multi-omic data to perform classification tasks for heterogeneous diseases. MOGDx has a network taxonomy. It fuses patient similarity networks, augments this integrated network with a reduced vector representation of genomic data and performs classification using a graph convolutional network. MOGDx was evaluated on three datasets from the cancer genome atlas for breast invasive carcinoma, kidney cancer, and low grade glioma. MOGDx demonstrated state-of-the-art performance and an ability to identify relevant multi-omic

```
PMID- 39177104
OWN - NLM
STAT- MEDLINE
DCOM- 20240904
LR  - 20240912
IS  - 1367-4811 (Electronic)
IS  - 1367-4803 (Print)
IS  - 1367-4803 (Linking)
VI  - 40
IP  - 9
DP  - 2024 Sep 2
TI  - Multi-Omic Graph Diagnosis (MOGDx): a data integration tool to perform
      classification tasks for heterogeneous diseases.
LID - 10.1093/bioinformatics/btae523 [doi]
LID - btae523
AB  - MOTIVATION: Heterogeneity in human diseases presents challenges in diagnosis and
      treatments due to the broad range of manifestations and symptoms. With the rapid
      development of labelled multi-omic data, integrative machine learning methods
      have achieved breakthroughs in treatments by redefining these diseases at a more
      granular level. These approaches often have limitations in scalability,
      oversimplification, and handling of missing data. RESULTS: In this study, we
      introduce Multi-Omic Graph Diagnosis (MOGDx), a flexible command line tool for
      the integration of multi-omic data to perform classification tasks for
      heterogeneous diseases. MOGDx has a network taxonomy. It fuses patient similarity
      networks, augments this integrated network with a reduced vector representation
      of genomic data and performs classification using a graph convolutional network.
      MOGDx was evaluated on three datasets from the cancer genome atlas for breast
      invasive carcinoma, kidney cancer, and low grade glioma. MOGDx demonstrated
      state-of-the-art performance and an ability to identify relevant multi-omic
      markers in each task. It integrated more genomic measures with greater patient
      coverage compared to other network integrative methods. Overall, MOGDx is a
      promising tool for integrating multi-omic data, classifying heterogeneous
      diseases, and aiding interpretation of genomic marker data. AVAILABILITY AND
      IMPLEMENTATION: MOGDx source code is available from
      https://github.com/biomedicalinformaticsgroup/MOGDx.
CI  - © The Author(s) 2024. Published by Oxford University Press.
FAU - Ryan, Barry
AU  - Ryan B
AUID- ORCID: 0009-0006-2779-5722
AD  - School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh,
      EH8 9AB, United Kingdom.
FAU - Marioni, Riccardo E
AU  - Marioni RE
AUID- ORCID: 0000-0003-4430-4260
AD  - Centre for Genomic and Experimental Medicine, Institute of Genetics and Cancer,
      University of Edinburgh, Edinburgh, EH4 2XU, United Kingdom.
FAU - Simpson, T Ian
AU  - Simpson TI
AUID- ORCID: 0000-0003-0495-7187
AD  - School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh,
      EH8 9AB, United Kingdom.
LA  - eng
GR  - EP/S02431X/1/United Kingdom Research and Innovation/
PT  - Journal Article
PT  - Research Support, Non-U.S. Gov't
PL  - England
TA  - Bioinformatics
JT  - Bioinformatics (Oxford, England)
```

```
CI  - © The Author(s) 2024. Published by Oxford University Press.
FAU - Ryan, Barry
AU  - Ryan B
AUID- ORCID: 0009-0006-2779-5722
AD  - School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh,
      EH8 9AB, United Kingdom.
FAU - Marioni, Riccardo E
AU  - Marioni RE
AUID- ORCID: 0000-0003-4430-4260
AD  - Centre for Genomic and Experimental Medicine, Institute of Genetics and Cancer,
      University of Edinburgh, Edinburgh, EH4 2XU, United Kingdom.
FAU - Simpson, T Ian
AU  - Simpson TI
AUID- ORCID: 0000-0003-0495-7187
AD  - School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh,
      EH8 9AB, United Kingdom.
LA  - eng
GR  - EP/S02431X/1/United Kingdom Research and Innovation/
PT  - Journal Article
PT  - Research Support, Non-U.S. Gov't
PL  - England
TA  - Bioinformatics
JT  - Bioinformatics (Oxford, England)
JID - 9808944
SB  - IM
MH  - Humans
MH  - *Genomics/methods
MH  - Software
MH  - Breast Neoplasms
MH  - Neoplasms
MH  - Kidney Neoplasms/genetics/classification
MH  - Machine Learning
MH  - Computational Biology/methods
MH  - Glioma/genetics/classification
MH  - Multiomics
PMC - PMC11374023
COIS- R.E.M. is a scientific advisor to Optima Partners and the Epigenetic Clock
      Development Foundation.
EDAT- 2024/08/23 12:46
MHDA- 2024/09/04 18:42
PMCR- 2024/08/23
CRDT- 2024/08/23 06:53
PHST- 2024/04/19 00:00 [received]
PHST- 2024/07/17 00:00 [revised]
PHST- 2024/08/23 00:00 [accepted]
PHST- 2024/09/04 18:42 [medline]
PHST- 2024/08/23 12:46 [pubmed]
PHST- 2024/08/23 06:53 [entrez]
PHST- 2024/08/23 00:00 [pmc-release]
AID - 7739700 [pii]
AID - btae523 [pii]
AID - 10.1093/bioinformatics/btae523 [doi]
PST - ppublish
SO  - Bioinformatics. 2024 Sep 2;40(9):btae523. doi: 10.1093/bioinformatics/btae523.
```

PubMedCentral

https://www.ncbi.nlm.nih.gov/pmc/



- PMC is a free archive for biomedical and life sciences literature.
- Launched online in 2000 and maintained by NCBI at NLM.
- Contains over 10 million full-text articles from the late 1700s to present.
- Includes formally published journal articles, peer-reviewed author manuscripts accepted for publication, and preprints prior to peer review.
- Content is freely accessible, sometimes after an embargo; the goal is to enhance accessibility and ensure the longevity of digital material.
- Although free to read, copyright restrictions still apply as noted in the PMC Copyright Notice.
- Access to content in human readable and XML format
- Supports integration with other resources, enhancing the utility for research through structured data.
- Allows bulk retrieval of data for text mining and other research purposes within certain collections.

PubMed eUtils Simple Search

Simple example showing the effect of field tags

```
from Bio import Entrez

#read in your email from a file
with open('../../api_keys/ncbi_email.txt', 'r') as file:
    email = file.read().replace('\n', '')

#set up Entrez
Entrez.email = email

#search 1 - for autism spectrum disorder
handle = Entrez.esearch(db='pubmed',term='Autism Spectrum Disorder')
record = Entrez.read(handle)
handle.close()

# print the number of records found
print(record['Count'])
```

✓ 0.5s

```
59932
```

```
#search 2 - for autism
handle = Entrez.esearch(db='pubmed',term='Autism')
record = Entrez.read(handle)
handle.close()

print(record['Count'])
```

✓ 0.4s

```
79913
```

```
#search 3 - for autism spectrum disorder using MeSH
handle = Entrez.esearch(db='pubmed',term='"Autism Spectrum Disorder"'+'[MH]')
record = Entrez.read(handle)
handle.close()

print(record['Count'])
```

✓ 0.4s

```
46301
```

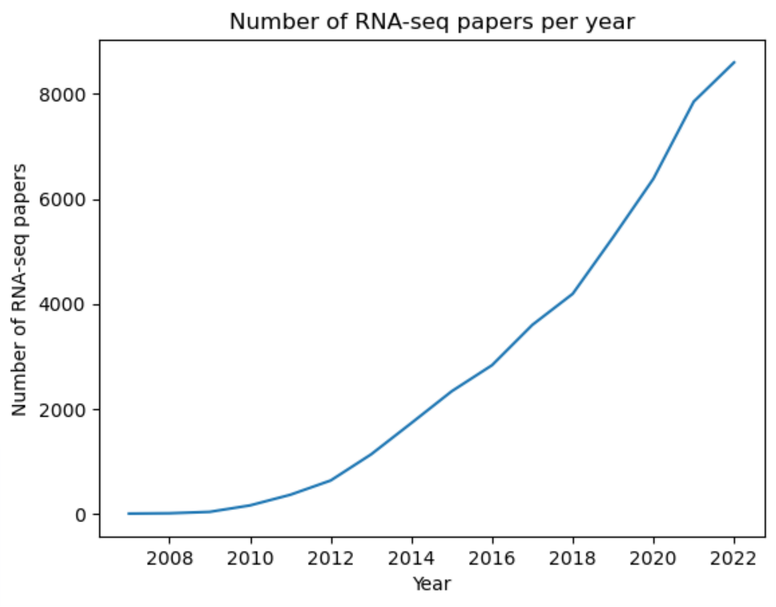
PubMed eUtils Temporal Search

Simple example of temporal data exploration

```python
from Bio import Entrez
import pandas as pd

#dicdtionary to store the counts of papers per year
RNAseq_counts = {}

#read in your email from a file
with open('../../api_keys/ncbi_email.txt', 'r') as file:
    email = file.read().replace('\n', '')

#set up Entrez
Entrez.email = email

#search for papers with RNA-seq in the title or abstract for each year
for i in range(2007,2023,1):
    handle = Entrez.esearch(db='pubmed',term=str(i)+'[dp] AND RNA-seq[TIAB]')
    record = Entrez.read(handle)
    handle.close()
    # we can iterate through the record and only return the 'nucleotide' result
    for row in record:
        if row == 'Count':
            RNAseq_counts[i] = int(record[row])
```

```python
#convert the dictionary to a pandas dataframe with columns year and paper count
rnaseq_papers_by_year = pd.DataFrame(list(RNAseq_counts.items()), columns=['year', 'paper_count'])

rnaseq_papers_by_year.head()
```

```python
#plot the data as a line plot
import matplotlib.pyplot as plt

plt.plot(rnaseq_papers_by_year['year'], rnaseq_papers_by_year['paper_count'])
plt.xlabel('Year')
plt.ylabel('Number of RNA-seq papers')
plt.title('Number of RNA-seq papers per year')
plt.show()
```



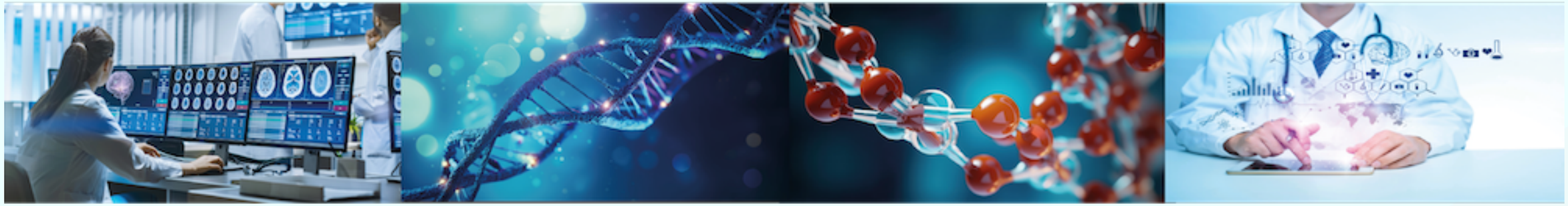
PubMed eUtils Gene Based Search

A complex query using eUtils on the GenCC set to find the earliest citation for each gene

```python
def get_first_pubmed_year(pubmed_ids):
    pubmed_ids = pubmed_ids.split(',')
    #convert the list of pubmed_ids to a string with [uid] following each id and then join them with 'AND'
    pubmed_ids_query = ' OR '.join([f'{pubmed_id}[pmid]' for pubmed_id in pubmed_ids])

    # Define the parameters for the eSearch request
    esearch_params = {
        'db': 'pubmed',
        'term': pubmed_ids_query,
        'api_key': api_key,
        'email': email,
        'usehistory': 'y'
    }

    # encode the parameters so they can be passed to the API
    encoded_data = urllib.parse.urlencode(esearch_params).encode('utf-8')

    # the base request url for eSearch
    url = f"https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi"

    # make the request
    request = urllib.request.Request(url, data=encoded_data)
    response = urllib.request.urlopen(request)

    # read into an XML object
    esaerch_data_XML = ET.fromstring(response.read())

    # Extract WebEnv and QueryKey
    webenv = esaerch_data_XML.find('WebEnv').text
    query_key = esaerch_data_XML.find('QueryKey').text

    esummary_params = {
    'db': 'pubmed',
    'query_key': query_key,
    'WebEnv': webenv,
    'api_key': api_key,
    'email': email
}

    # encode the parameters so they can be passed to the API
    encoded_data = urllib.parse.urlencode(esummary_params).encode('utf-8')

    # the base request url for eSummary
    url = f"https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esummary.fcgi"

    # make the request
    request = urllib.request.Request(url, data=encoded_data)
    response = urllib.request.urlopen(request)

    # read into an XML object
    esummary_data_XML = ET.fromstring(response.read())

    #print the XML
    # print(ET.tostring(esummary_data_XML).decode())

    # list to store the years of publication
    years = []

    # Extract the year of publication for each pubmed_id
    for doc in esummary_data_XML.findall('DocSum'):
        for item in doc.findall('Item'):
            if item.attrib['Name'] == 'PubDate':
                #take the first 4 characters of the date to get the year
                year = item.text[:4]
                #append the year to the list
                years.append(int(year))

    #return the lowest year
    return min(years)
```

```python
#pass the pubmed_ids from genccGene2PubMed to the function for the first 10 genes
for i in range(10):
    gene = genccGene2PubMed.loc[i, 'gene_symbol']
    pubmed_ids = genccGene2PubMed.loc[i, 'pubmed_id']
    try:
        year = get_first_pubmed_year(pubmed_ids)
        print(f'{gene} - {year}')
    except:
        print(f'{gene} - Error')
```

```
A2ML1 - 2015
BRAF - 2009
CBL - 2009
HRAS - 2005
KRAS - 2011
LZTR1 - 2015
MAP2K1 - 2011
MAP2K2 - 2011
MRAS - 2017
NF1 - 2003
```



Programming for Biomedical Informatics

Next Lecture this Thursday - “Mining & Analysing Biomedical Literature”

Please Bring your Laptop!

Ask Questions on the EdStem Discussion Board

https://github.com/tisimpson/pbi

THE UNIVERSITY of EDINBURGH
informatics