DBInspector

Maren P., Rebeca F., Simon M., Lauren D.

Problem

Maren

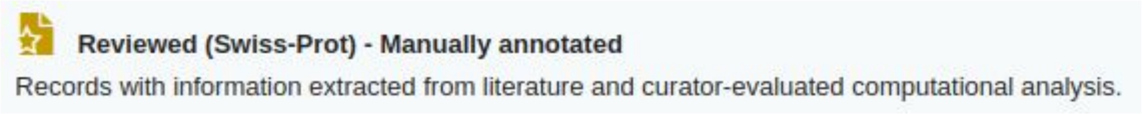
Databases

What can be expected?

How can they be utilized?

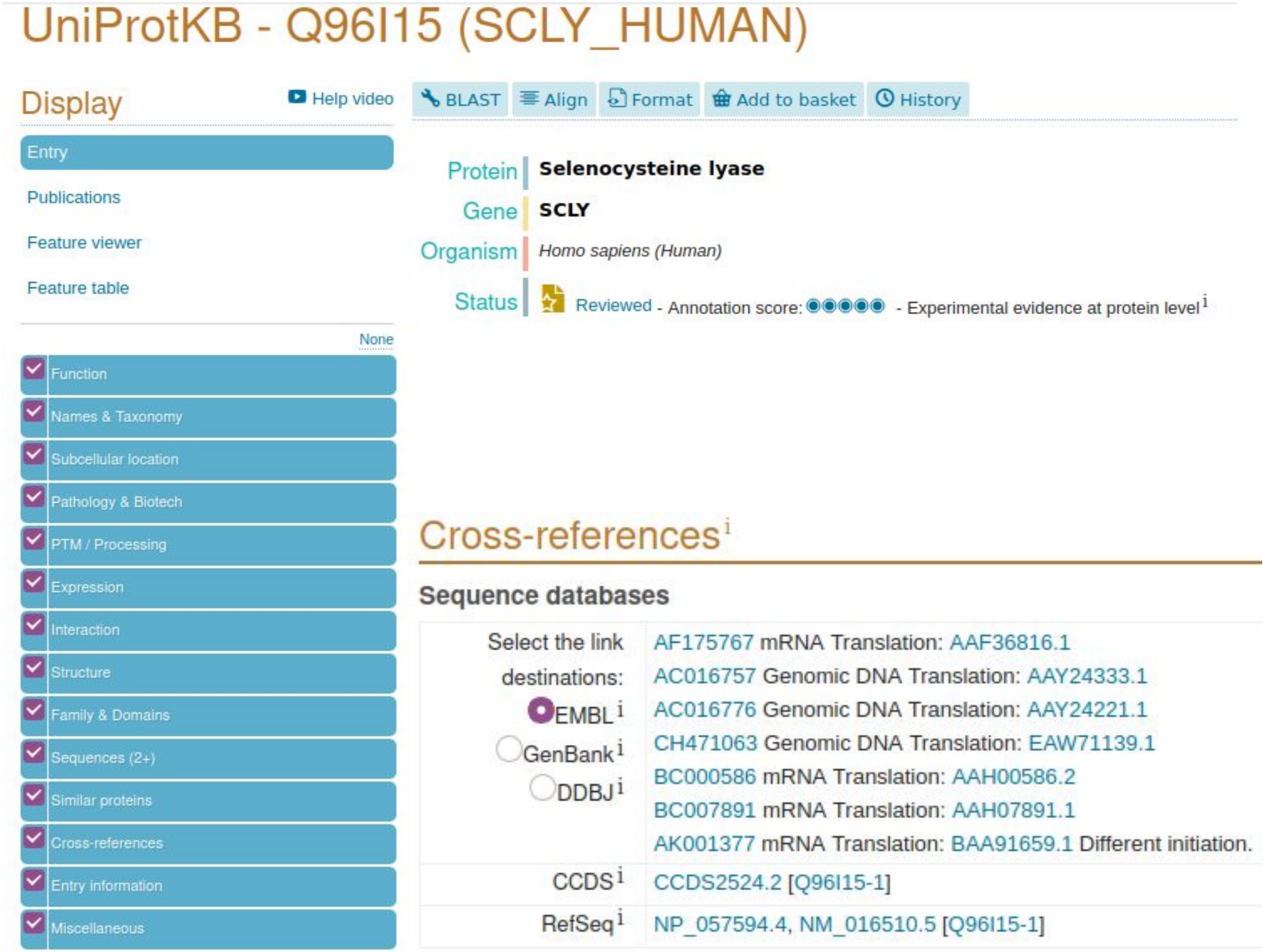
Different levels of curation

UniProt



Reviewed (Swiss-Prot) - Manually annotated

Records with information extracted from literature and curator-evaluated computational analysis.



UniProtKB - Q96I15 (SCLY_HUMAN)

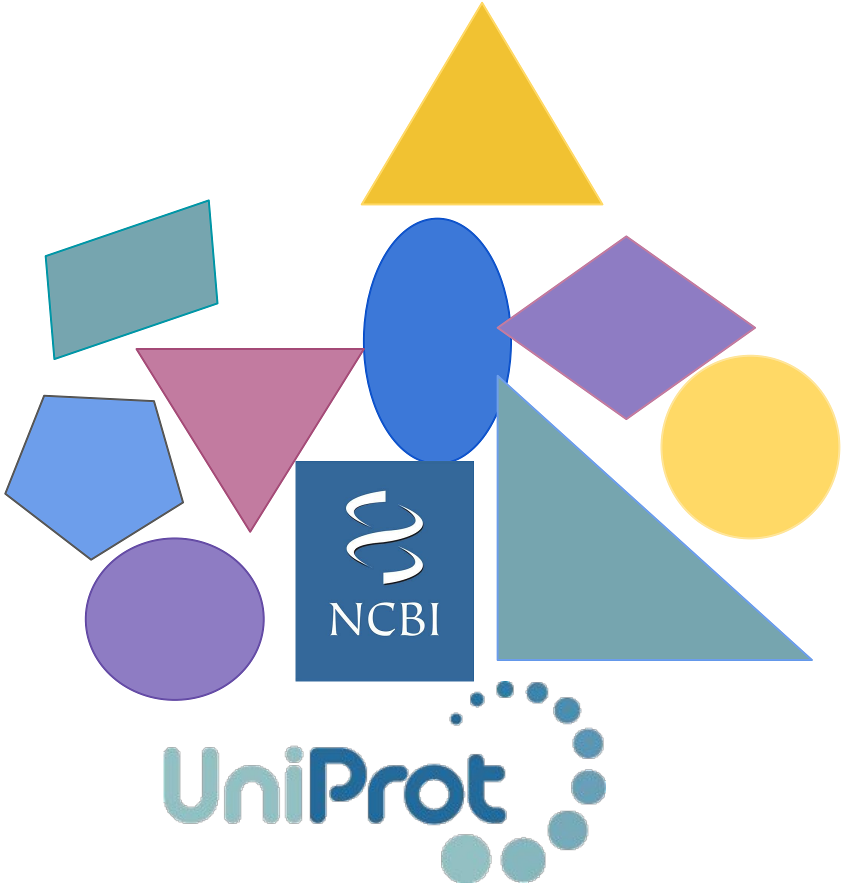
Protein	**Selenocysteine lyase**
Gene	**SCLY**
Organism	*Homo sapiens (Human)*
Status	Reviewed - Annotation score: ●●●●● - Experimental evidence at protein level i

Cross-references i

Sequence databases

Select the link destinations: ● EMBL i ○ GenBank i ○ DDBJ i	AF175767 mRNA Translation: AAF36816.1 AC016757 Genomic DNA Translation: AAY24333.1 AC016776 Genomic DNA Translation: AAY24221.1 CH471063 Genomic DNA Translation: EAW71139.1 BC000586 mRNA Translation: AAH00586.2 BC007891 mRNA Translation: AAH07891.1 AK001377 mRNA Translation: BAA91659.1 Different initiation.
CCDS i	CCDS2524.2 [Q96I15-1]
RefSeq i	NP_057594.4, NM_016510.5 [Q96I15-1]

scenario



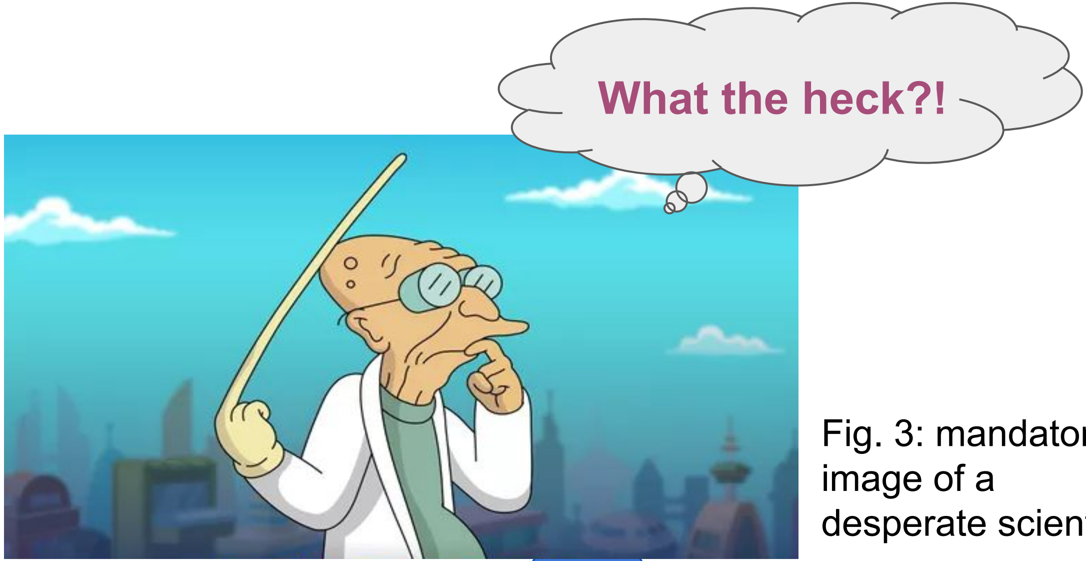
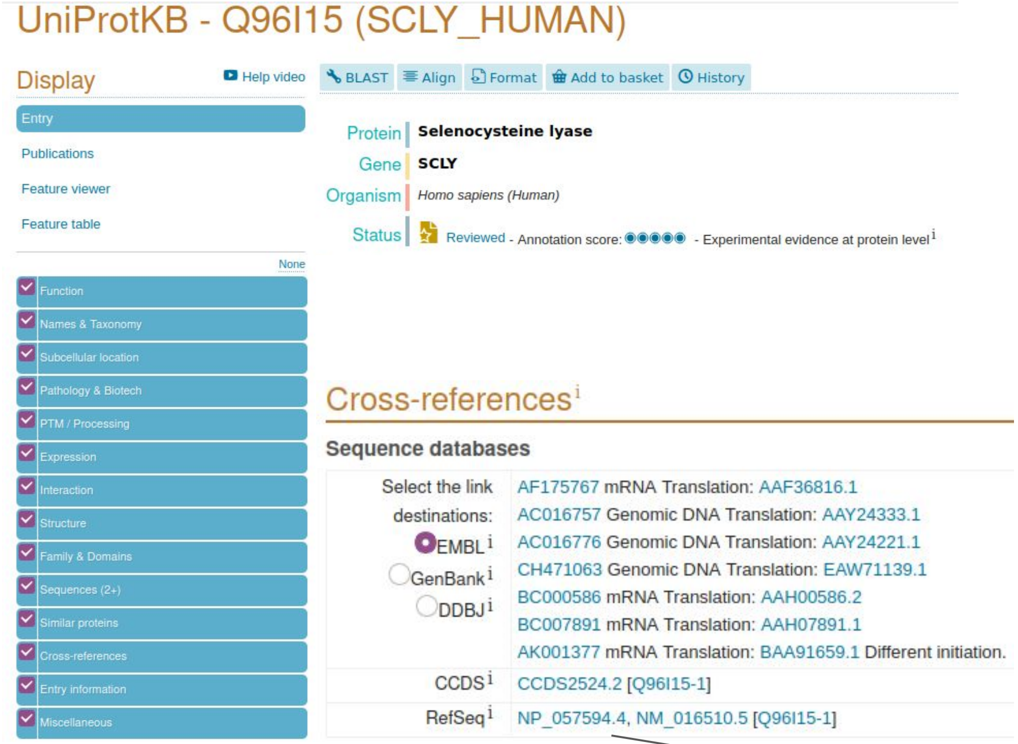


Fig. 3: mandatory image of a desperate scientist

NCBI

UniProt



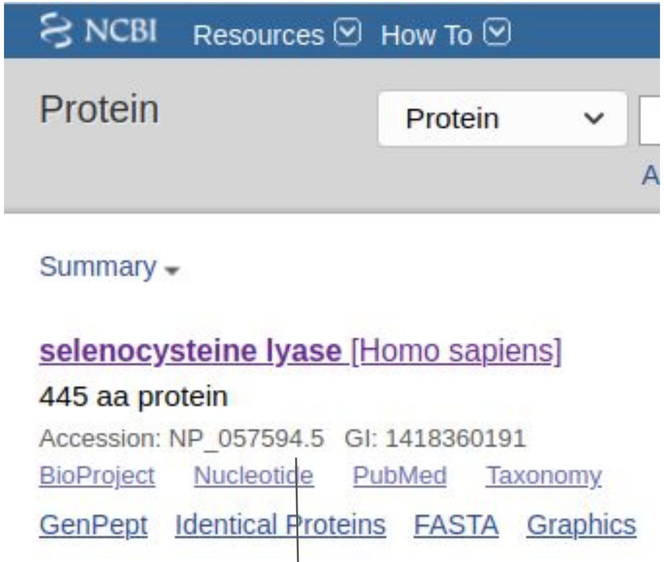
UniProtKB - Q96I15 (SCLY_HUMAN)

Protein | **Selenocysteine lyase**
Gene | **SCLY**
Organism | *Homo sapiens (Human)*
Status | Reviewed - Annotation score: ●●●●● - Experimental evidence at protein level[i]

Cross-references[i]

Sequence databases

Select the link destinations: ● EMBL[i] ○ GenBank[i] ○ DDBJ[i]	AF175767 mRNA Translation: AAF36816.1 AC016757 Genomic DNA Translation: AAY24333.1 AC016776 Genomic DNA Translation: AAY24221.1 CH471063 Genomic DNA Translation: EAW71139.1 BC000586 mRNA Translation: AAH00586.2 BC007891 mRNA Translation: AAH07891.1 AK001377 mRNA Translation: BAA91659.1 Different initiation.
CCDS[i]	CCDS2524.2 [Q96I15-1]
RefSeq[i]	NP_057594.4, NM_016510.5 [Q96I15-1]



selenocysteine lyase [Homo sapiens]

445 aa protein

Accession: NP_057594.5 GI: 1418360191

BioProject Nucleotide PubMed Taxonomy

GenPept Identical Proteins FASTA Graphics

NP_057594.5 NCBI

NP_057594.4 UniProt

Problem

UniProt ↔ NCBI RefSeq

- ➔ compare entries between the two databases to see whether they are up to date **?**
- ➔ how much can we as bioinformaticians rely on database cross-references **?**
- ➔ which “gold standard” protein database does the better job at referencing the other **?**

Overview Solution

Lauren

Tasks and Responsibilities

Lauren:

- Programmatically download all data
- Parse raw data files into well-organized metadata files

Maren:

- Map the metadata between UniProt and RefSeq, addressing various complicated relationships and discrepancies appropriately

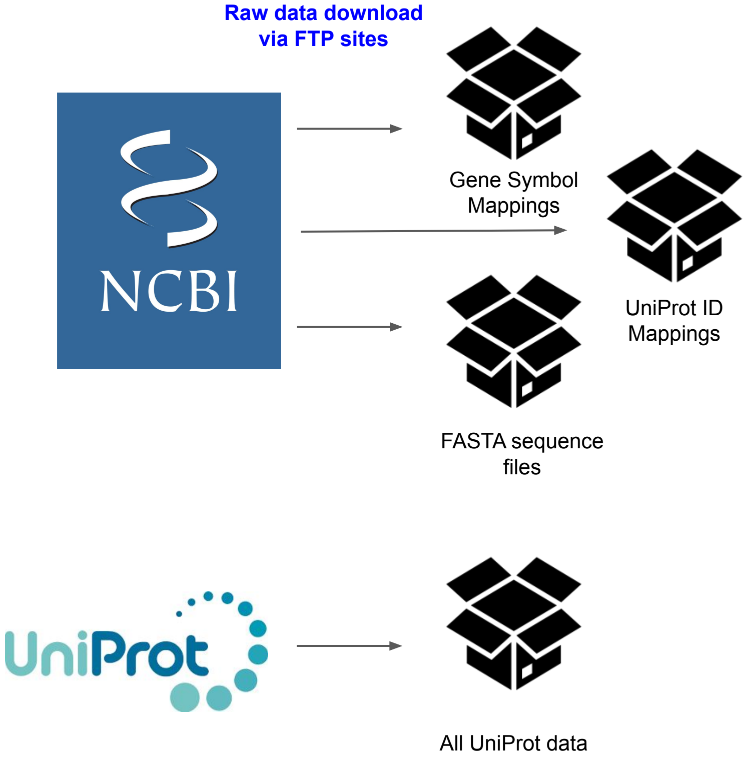
Rebeca:

- Create functions to allow queries into the metadata which utilize mappings
- Integrate into a command line interface

Simon:

- Integrate all functionality into a graphical user interface
- Ensure that the GUI is user-friendly and fully functional

Getting the database information via FTP download pages:



Raw data download via FTP sites

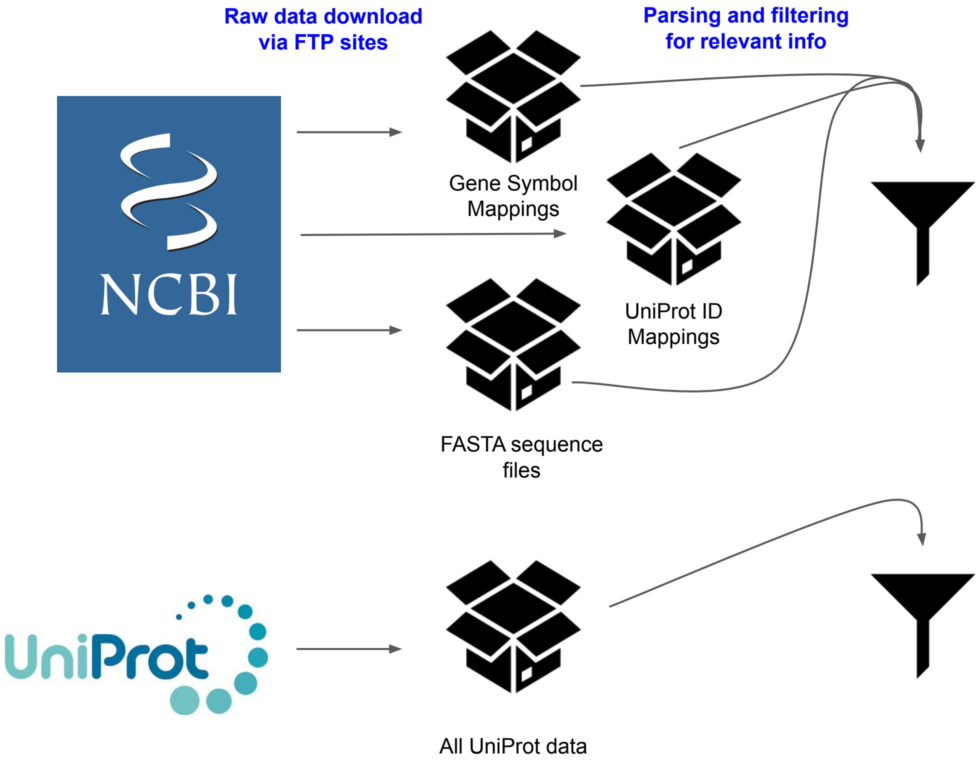
Gene Symbol Mappings

UniProt ID Mappings

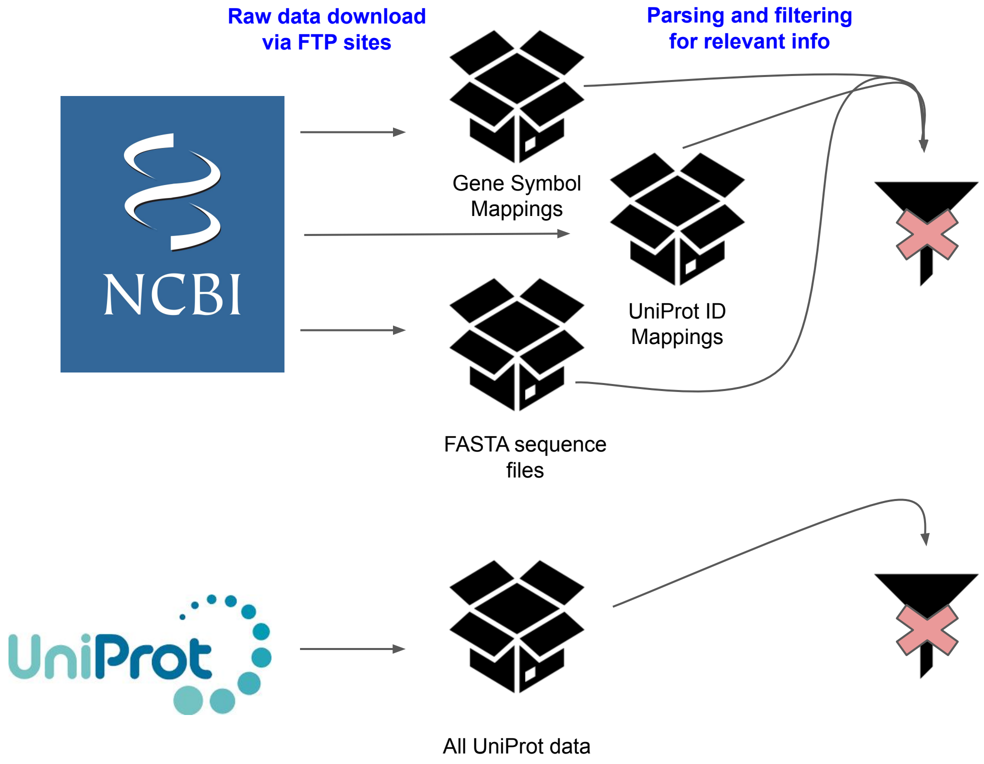
FASTA sequence files

All UniProt data

Filtering out and storing metadata that we need:



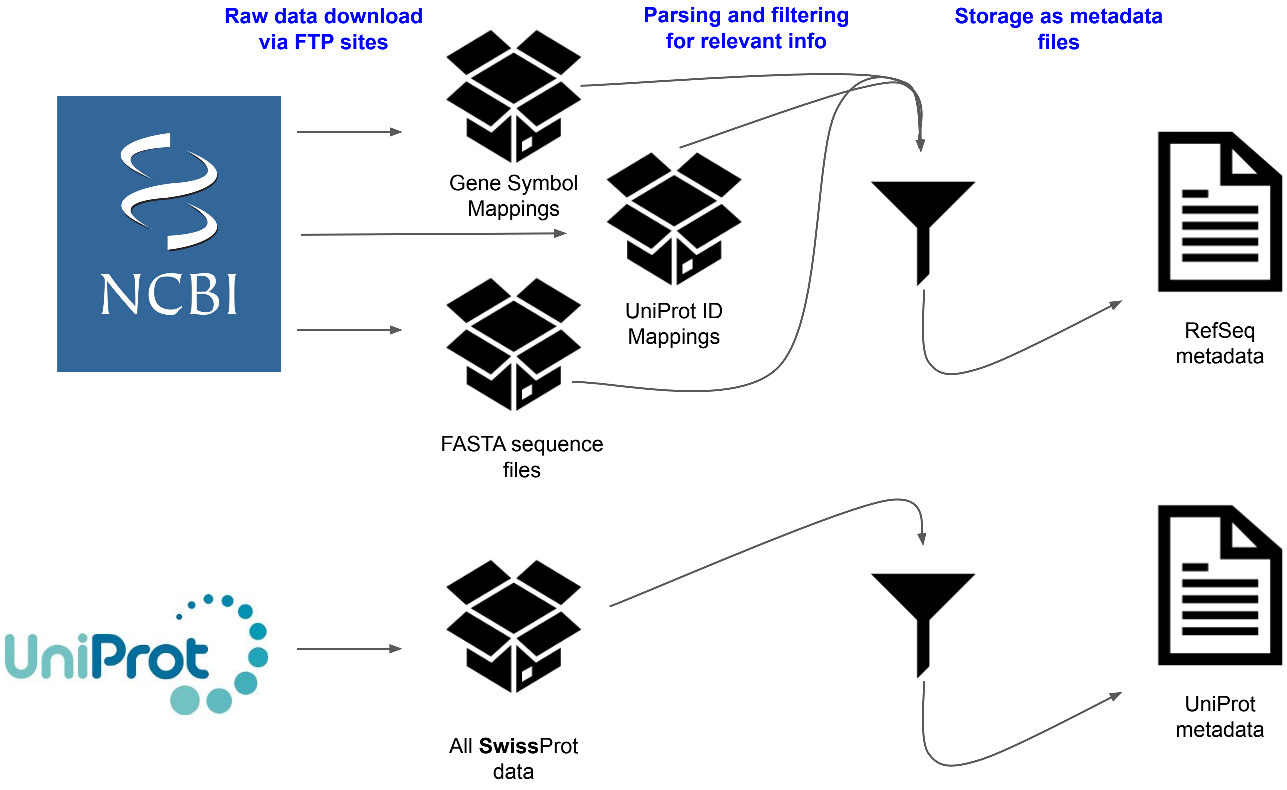
Problem: Too Much Data!



Solution:

- Use only curated / approved proteins
- Remove information from memory which is no longer needed
- Optimize code and data structures to take up the least memory possible

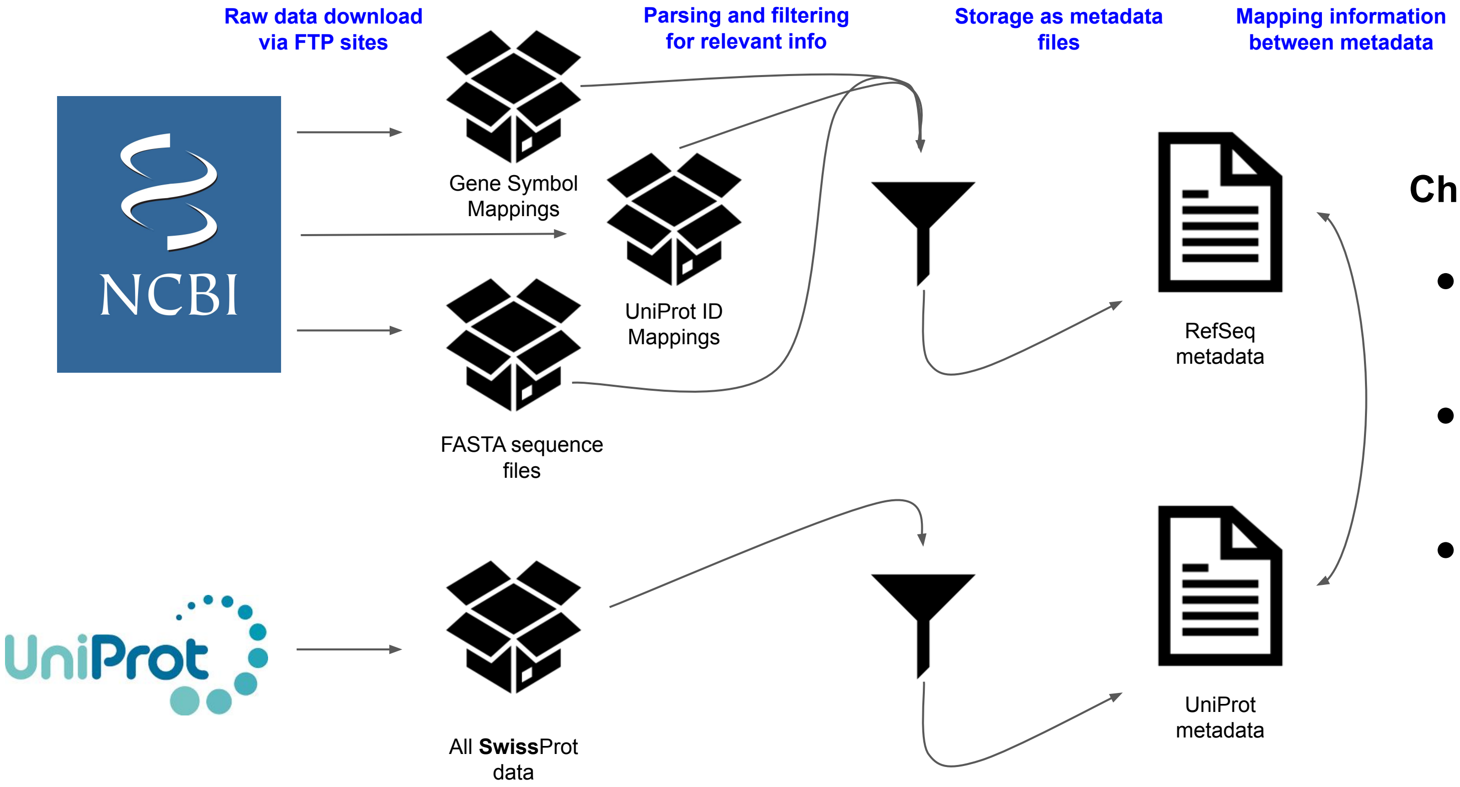
Next Challenge: How to store the metadata?



Options:

- “FASTA”-style
- **JSON format**
- Relational Database
- JSON-based database

Mapping RefSeq and UniProt to each other:

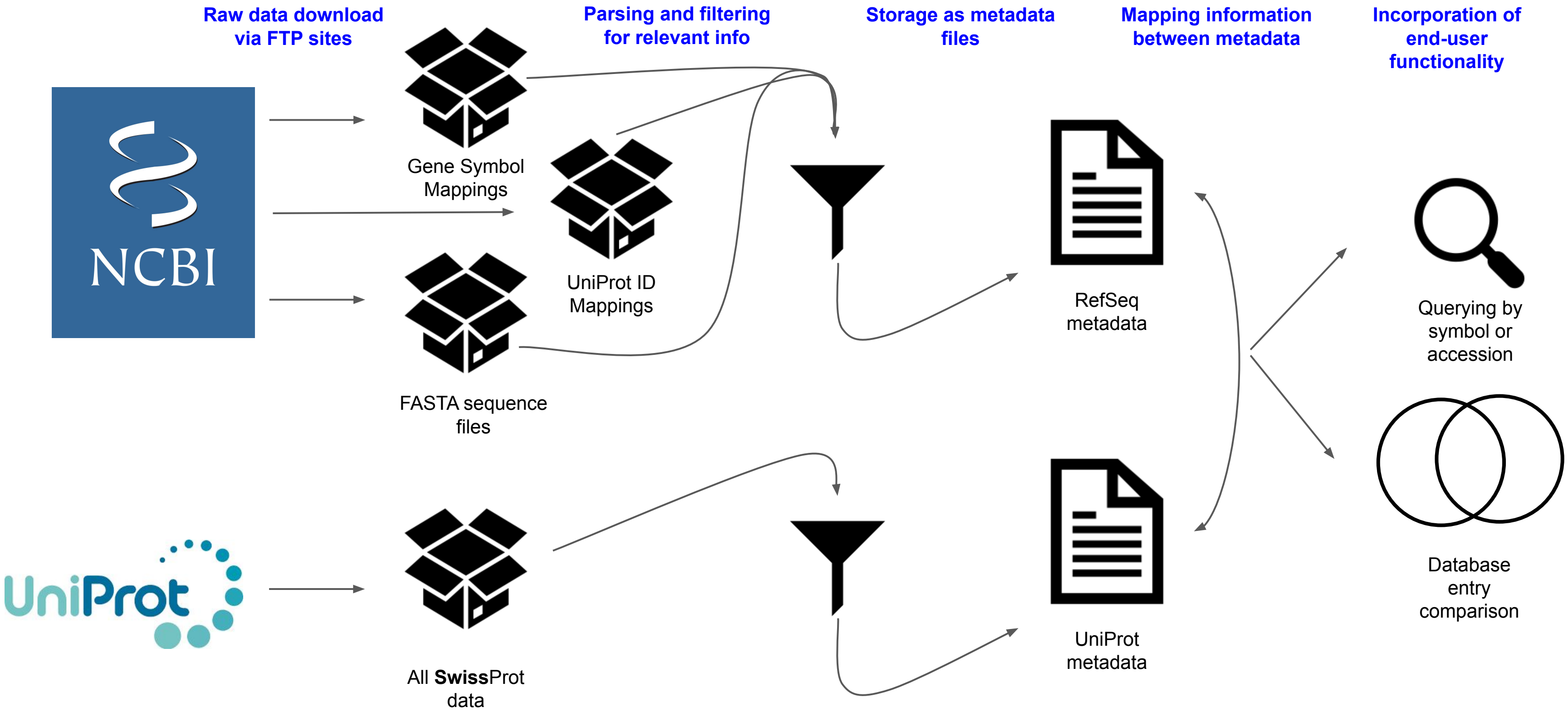


Raw data download via FTP sites
Parsing and filtering for relevant info
Storage as metadata files
Mapping information between metadata
NCBI
Gene Symbol Mappings
UniProt ID Mappings
FASTA sequence files
RefSeq metadata
UniProt
All **Swiss**Prot data
UniProt metadata

Challenges:

- 1-many and many-many relationships
- No unique ID to make perfect matches
- Inconsistencies mapping in either direction

Adding end-user functionality: querying & comparison



Raw data download via FTP sites

Parsing and filtering for relevant info

Storage as metadata files

Mapping information between metadata

Incorporation of end-user functionality

Gene Symbol Mappings

UniProt ID Mappings

FASTA sequence files

All **Swiss**Prot data

RefSeq metadata

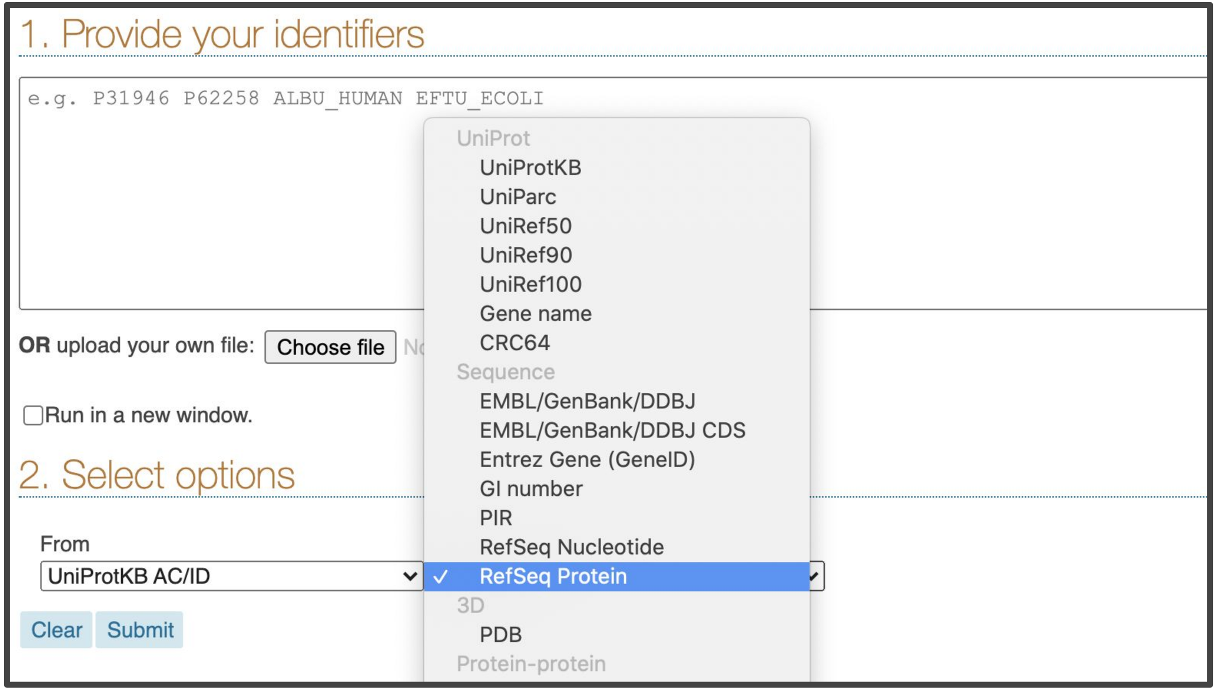
UniProt metadata

Querying by symbol or accession

Database entry comparison

Other Services: UniProt Retrieve/ID Mapping Tool

https://www.uniprot.org/uploadlists/



1. Provide your identifiers

e.g. P31946 P62258 ALBU_HUMAN EFTU_ECOLI

OR upload your own file: Choose file

□ Run in a new window.

2. Select options

From: UniProtKB AC/ID

To: ✓ RefSeq Protein

Pros:

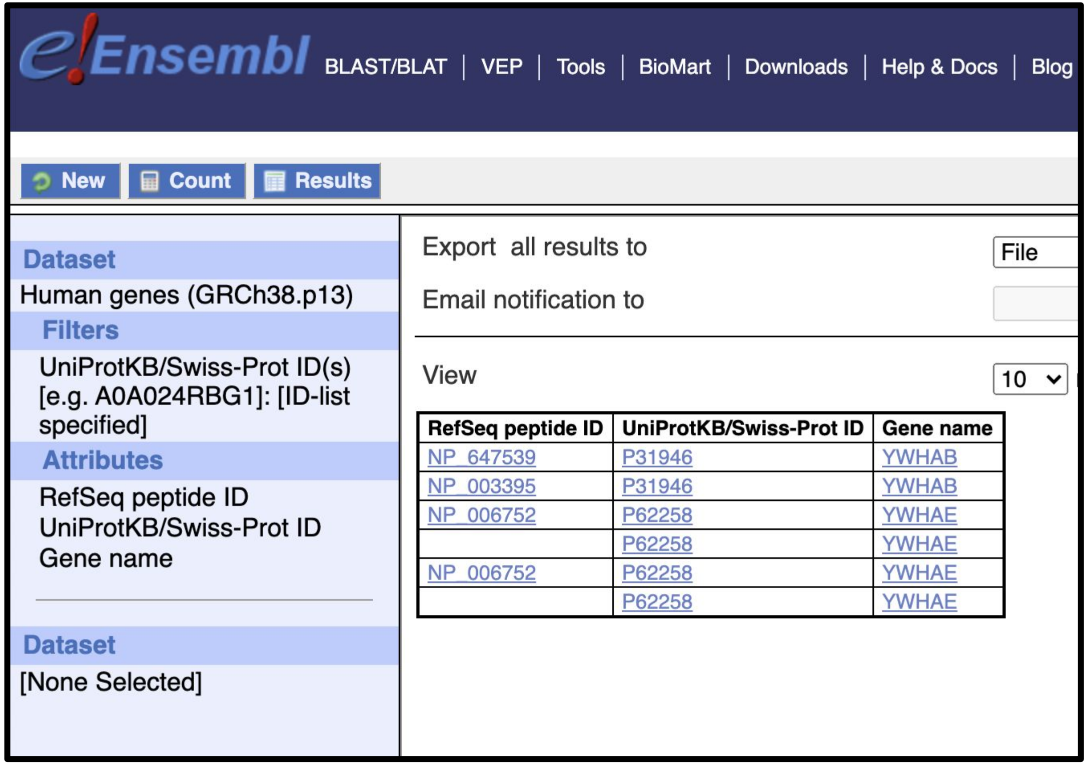
- Includes all (not just verified) proteins from RefSeq
- Includes far more mapping options besides RefSeq
- Can search multiple proteins at once
- Includes more organisms than human

Cons:

- Can only map to one other ID at a time
- Works based on UniProt info (not both UniProt and RefSeq info)

Other Services: Ensembl BioMart

https://m.ensembl.org/biomart/martview


e!Ensembl
BLAST/BLAT | VEP | Tools | BioMart | Downloads | Help & Docs | Blog
New
Count
Results

Dataset
Human genes (GRCh38.p13)

Filters
UniProtKB/Swiss-Prot ID(s) [e.g. A0A024RBG1]: [ID-list specified]

Attributes
RefSeq peptide ID
UniProtKB/Swiss-Prot ID
Gene name

Dataset
[None Selected]

Export all results to: File

Email notification to

View: 10

RefSeq peptide ID	UniProtKB/Swiss-Prot ID	Gene name
NP_647539	P31946	YWHAB
NP_003395	P31946	YWHAB
NP_006752	P62258	YWHAE
	P62258	YWHAE
NP_006752	P62258	YWHAE
	P62258	YWHAE

Pros:

- Includes far more mapping options besides RefSeq, UniProt, and gene symbol
- Can look up many IDs at once
- Fast
- Has API

Cons:

- Not very intuitive to use
- Can only look at sequence information from Ensembl
- Unclear where cross-references come from
- Does not include RefSeq versions

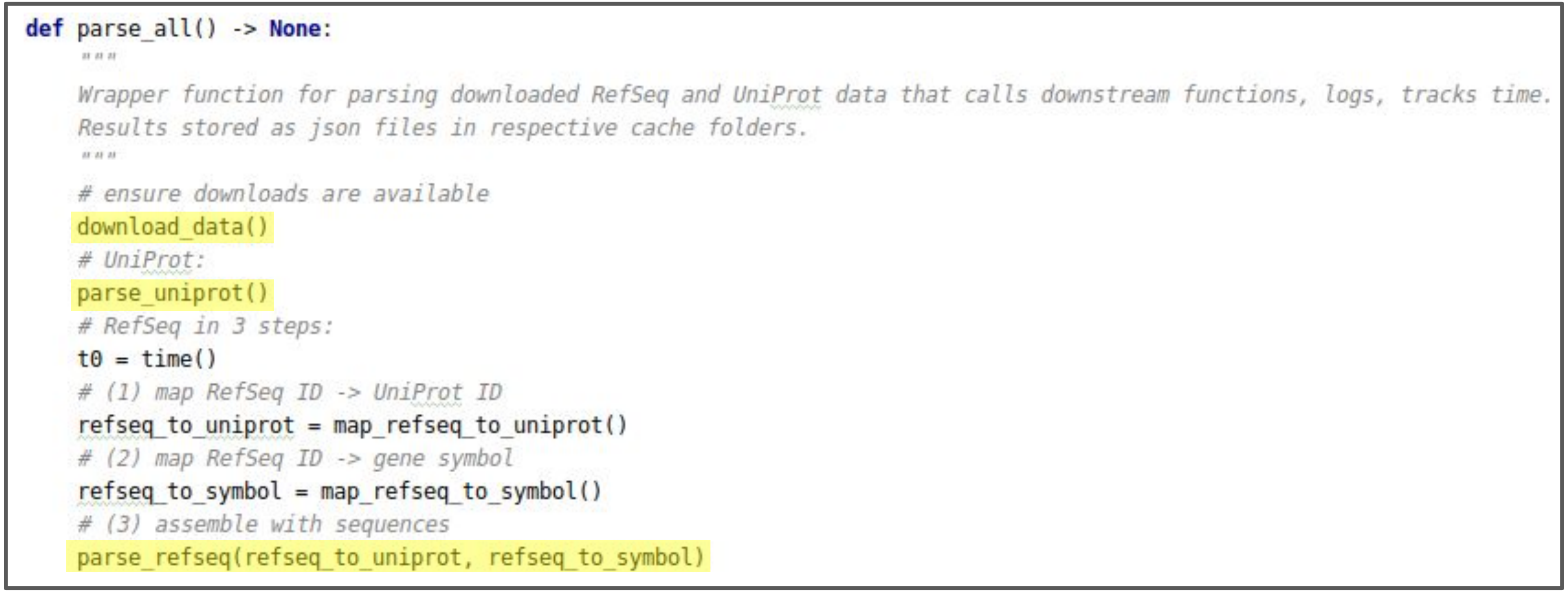
Software overview
CLI Demo

Rebeca

A brief look into our codebase

Download / prepare the data in three steps

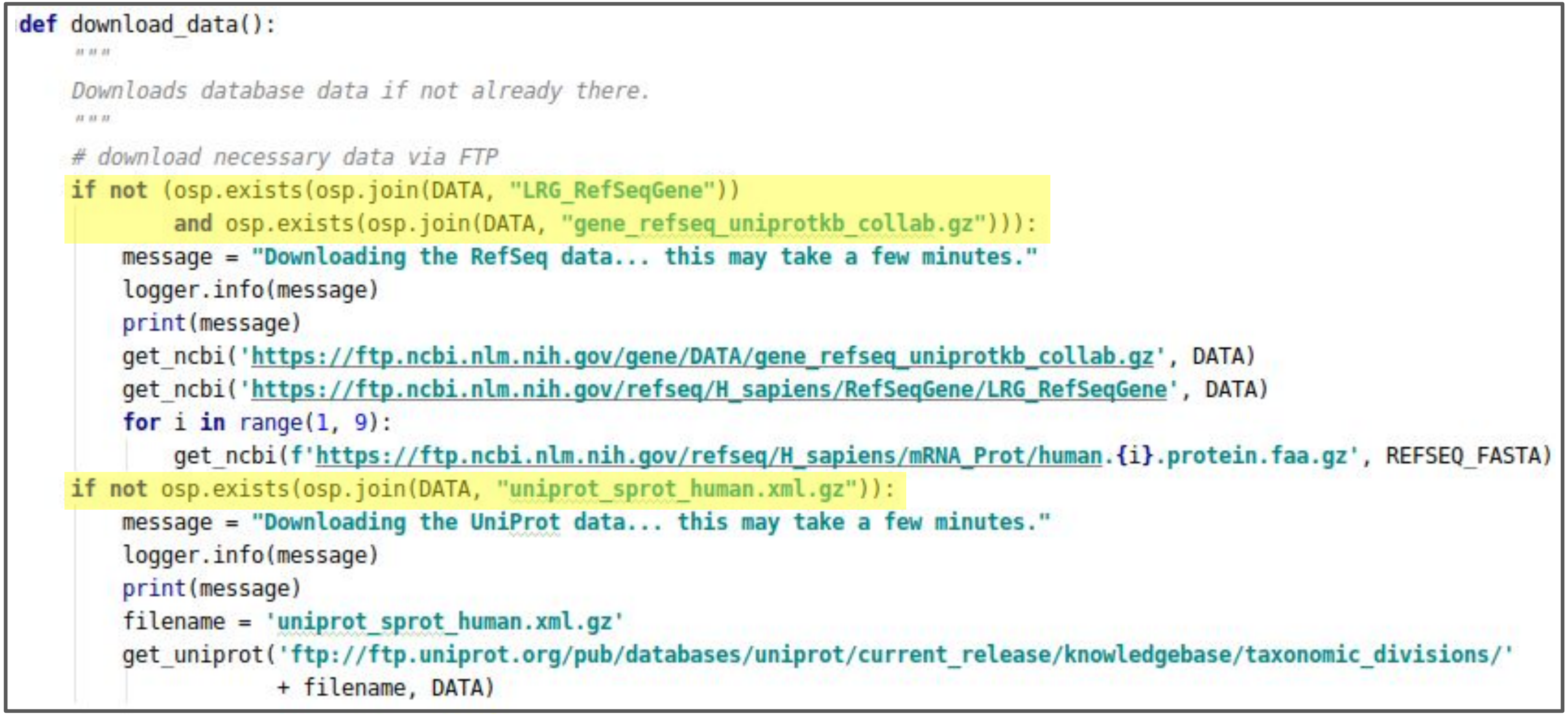
1. Download the data **2.** Parse UniProt data **3.** Parse RefSeq data



```python
def parse_all() -> None:
    """
    Wrapper function for parsing downloaded RefSeq and UniProt data that calls downstream functions, logs, tracks time.
    Results stored as json files in respective cache folders.
    """
    # ensure downloads are available
    download_data()
    # UniProt:
    parse_uniprot()
    # RefSeq in 3 steps:
    t0 = time()
    # (1) map RefSeq ID -> UniProt ID
    refseq_to_uniprot = map_refseq_to_uniprot()
    # (2) map RefSeq ID -> gene symbol
    refseq_to_symbol = map_refseq_to_symbol()
    # (3) assemble with sequences
    parse_refseq(refseq_to_uniprot, refseq_to_symbol)
```

Data download

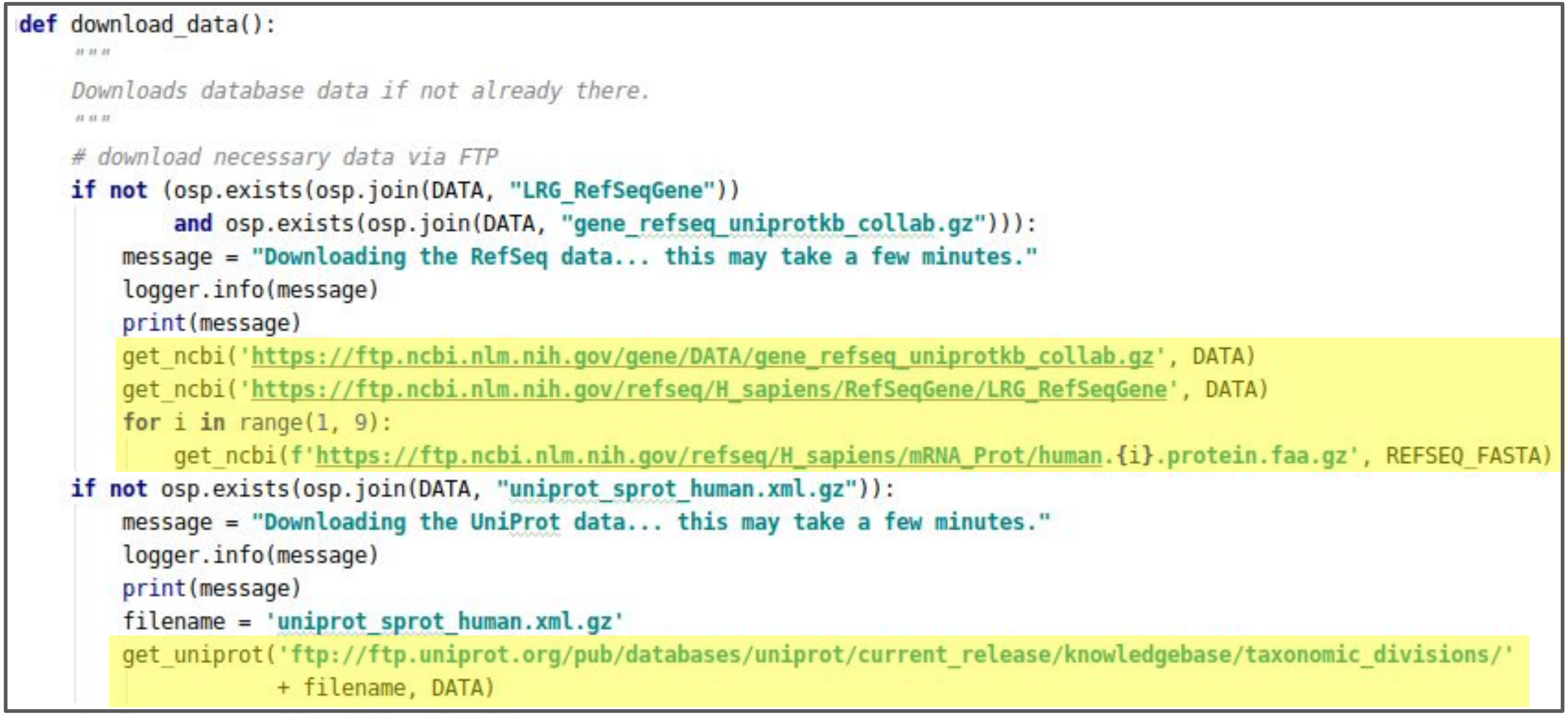
- Check if re-download necessary
- FTP download



```python
def download_data():
    """
    Downloads database data if not already there.
    """
    # download necessary data via FTP
    if not (osp.exists(osp.join(DATA, "LRG_RefSeqGene"))
            and osp.exists(osp.join(DATA, "gene_refseq_uniprotkb_collab.gz"))):
        message = "Downloading the RefSeq data... this may take a few minutes."
        logger.info(message)
        print(message)
        get_ncbi('https://ftp.ncbi.nlm.nih.gov/gene/DATA/gene_refseq_uniprotkb_collab.gz', DATA)
        get_ncbi('https://ftp.ncbi.nlm.nih.gov/refseq/H_sapiens/RefSeqGene/LRG_RefSeqGene', DATA)
        for i in range(1, 9):
            get_ncbi(f'https://ftp.ncbi.nlm.nih.gov/refseq/H_sapiens/mRNA_Prot/human.{i}.protein.faa.gz', REFSEQ_FASTA)
    if not osp.exists(osp.join(DATA, "uniprot_sprot_human.xml.gz")):
        message = "Downloading the UniProt data... this may take a few minutes."
        logger.info(message)
        print(message)
        filename = 'uniprot_sprot_human.xml.gz'
        get_uniprot('ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/taxonomic_divisions/'
                    + filename, DATA)
```

Data download

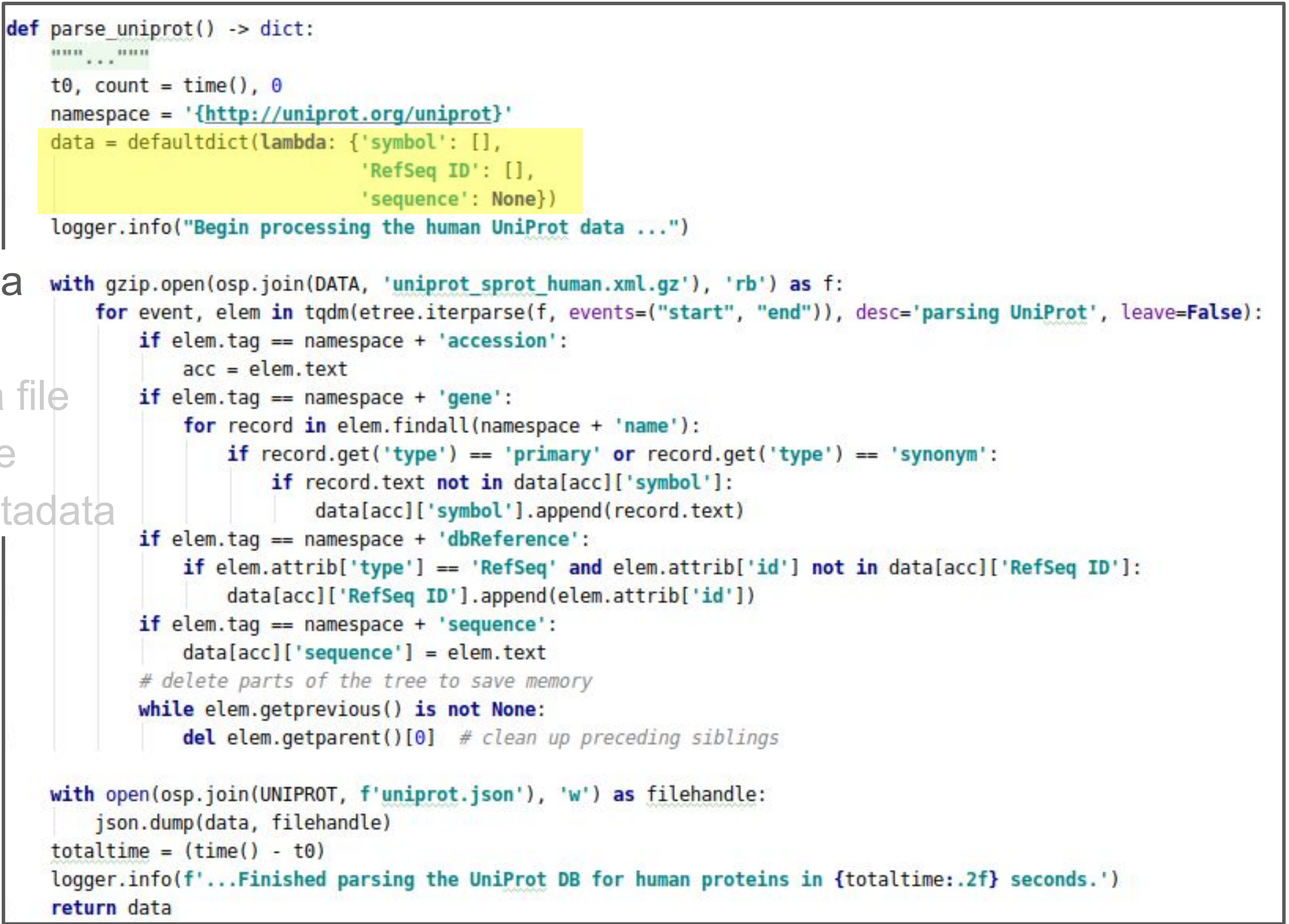
- Check if re-download necessary
- FTP download



```python
def download_data():
    """
    Downloads database data if not already there.
    """
    # download necessary data via FTP
    if not (osp.exists(osp.join(DATA, "LRG_RefSeqGene"))
            and osp.exists(osp.join(DATA, "gene_refseq_uniprotkb_collab.gz"))):
        message = "Downloading the RefSeq data... this may take a few minutes."
        logger.info(message)
        print(message)
        get_ncbi('https://ftp.ncbi.nlm.nih.gov/gene/DATA/gene_refseq_uniprotkb_collab.gz', DATA)
        get_ncbi('https://ftp.ncbi.nlm.nih.gov/refseq/H_sapiens/RefSeqGene/LRG_RefSeqGene', DATA)
        for i in range(1, 9):
            get_ncbi(f'https://ftp.ncbi.nlm.nih.gov/refseq/H_sapiens/mRNA_Prot/human.{i}.protein.faa.gz', REFSEQ_FASTA)
    if not osp.exists(osp.join(DATA, "uniprot_sprot_human.xml.gz")):
        message = "Downloading the UniProt data... this may take a few minutes."
        logger.info(message)
        print(message)
        filename = 'uniprot_sprot_human.xml.gz'
        get_uniprot('ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/taxonomic_divisions/'
                    + filename, DATA)
```

Parsing UniProt

- Set up the data structure
- Open the data file and extract the necessary metadata
- Save to json



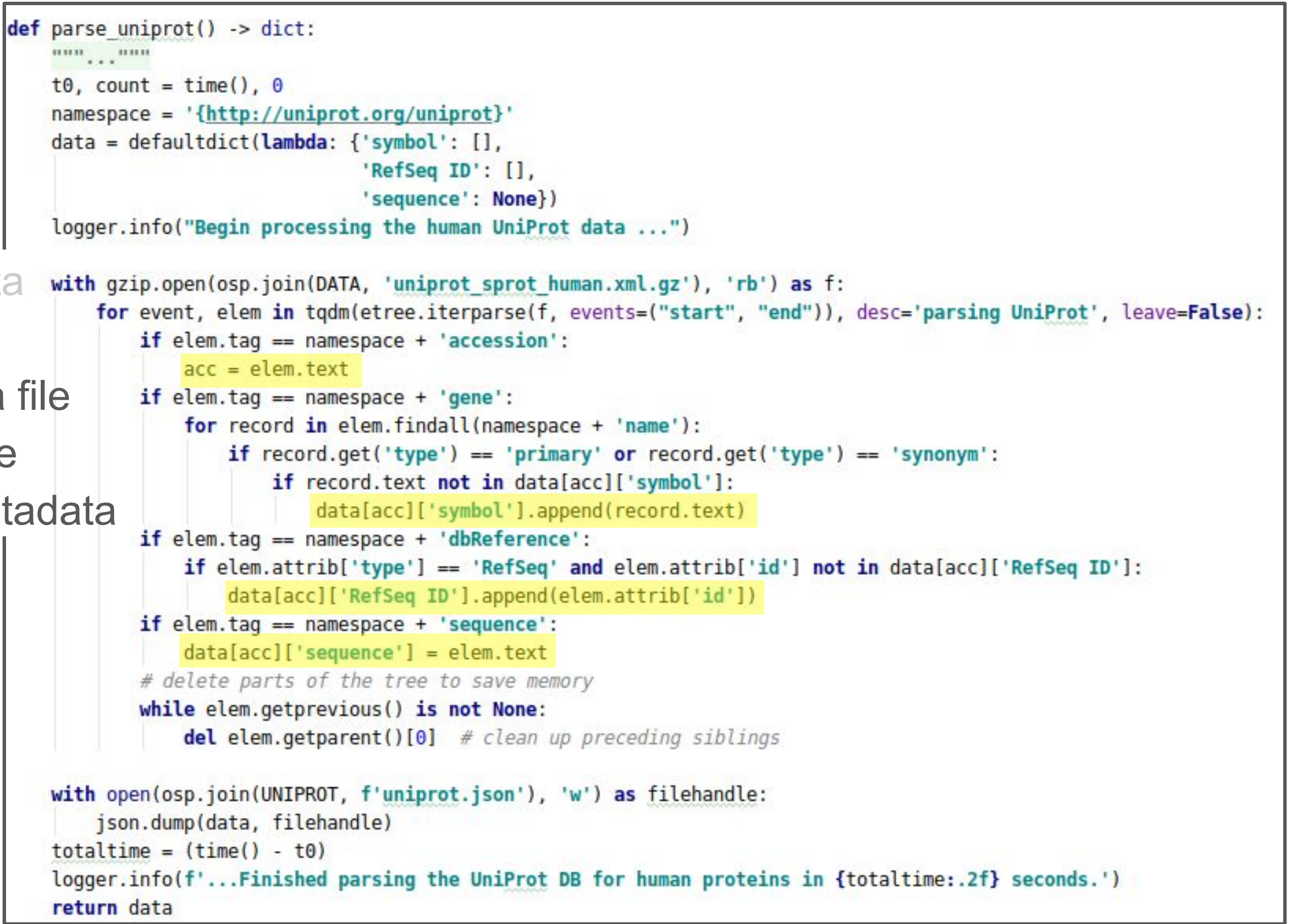
```
def parse_uniprot() -> dict:
    """..."""
    t0, count = time(), 0
    namespace = '{http://uniprot.org/uniprot}'
    data = defaultdict(lambda: {'symbol': [],
                                'RefSeq ID': [],
                                'sequence': None})
    logger.info("Begin processing the human UniProt data ...")

    with gzip.open(osp.join(DATA, 'uniprot_sprot_human.xml.gz'), 'rb') as f:
        for event, elem in tqdm(etree.iterparse(f, events=("start", "end")), desc='parsing UniProt', leave=False):
            if elem.tag == namespace + 'accession':
                acc = elem.text
            if elem.tag == namespace + 'gene':
                for record in elem.findall(namespace + 'name'):
                    if record.get('type') == 'primary' or record.get('type') == 'synonym':
                        if record.text not in data[acc]['symbol']:
                            data[acc]['symbol'].append(record.text)
            if elem.tag == namespace + 'dbReference':
                if elem.attrib['type'] == 'RefSeq' and elem.attrib['id'] not in data[acc]['RefSeq ID']:
                    data[acc]['RefSeq ID'].append(elem.attrib['id'])
            if elem.tag == namespace + 'sequence':
                data[acc]['sequence'] = elem.text
            # delete parts of the tree to save memory
            while elem.getprevious() is not None:
                del elem.getparent()[0]  # clean up preceding siblings

    with open(osp.join(UNIPROT, f'uniprot.json'), 'w') as filehandle:
        json.dump(data, filehandle)
    totaltime = (time() - t0)
    logger.info(f'...Finished parsing the UniProt DB for human proteins in {totaltime:.2f} seconds.')
    return data
```

Parsing UniProt

- Set up the data structure
- Open the data file and extract the necessary metadata
- Save to json



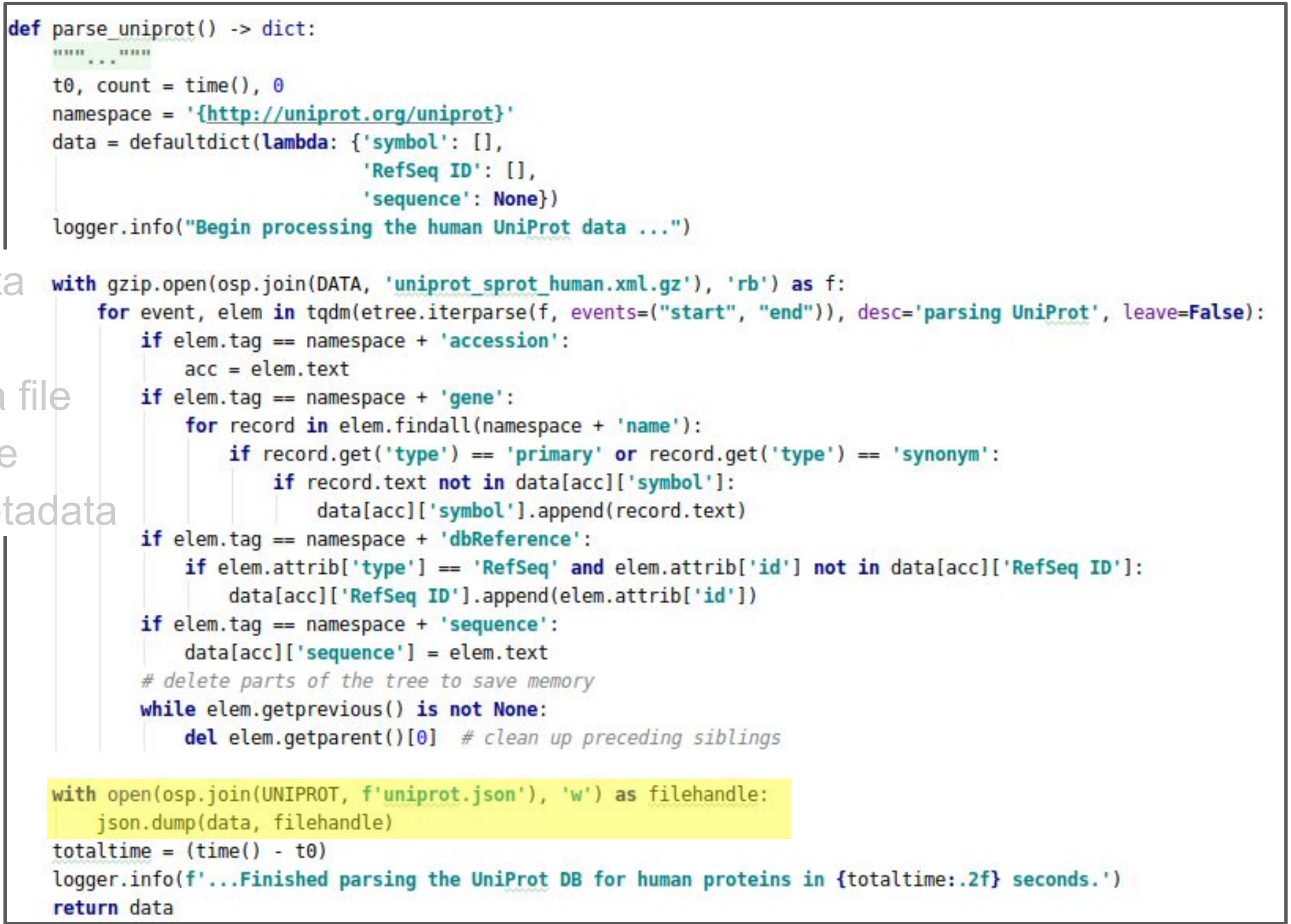
```
def parse_uniprot() -> dict:
    """..."""
    t0, count = time(), 0
    namespace = '{http://uniprot.org/uniprot}'
    data = defaultdict(lambda: {'symbol': [],
                                'RefSeq ID': [],
                                'sequence': None})
    logger.info("Begin processing the human UniProt data ...")

    with gzip.open(osp.join(DATA, 'uniprot_sprot_human.xml.gz'), 'rb') as f:
        for event, elem in tqdm(etree.iterparse(f, events=("start", "end")), desc='parsing UniProt', leave=False):
            if elem.tag == namespace + 'accession':
                acc = elem.text
            if elem.tag == namespace + 'gene':
                for record in elem.findall(namespace + 'name'):
                    if record.get('type') == 'primary' or record.get('type') == 'synonym':
                        if record.text not in data[acc]['symbol']:
                            data[acc]['symbol'].append(record.text)
            if elem.tag == namespace + 'dbReference':
                if elem.attrib['type'] == 'RefSeq' and elem.attrib['id'] not in data[acc]['RefSeq ID']:
                    data[acc]['RefSeq ID'].append(elem.attrib['id'])
            if elem.tag == namespace + 'sequence':
                data[acc]['sequence'] = elem.text
            # delete parts of the tree to save memory
            while elem.getprevious() is not None:
                del elem.getparent()[0]  # clean up preceding siblings

    with open(osp.join(UNIPROT, f'uniprot.json'), 'w') as filehandle:
        json.dump(data, filehandle)
    totaltime = (time() - t0)
    logger.info(f'...Finished parsing the UniProt DB for human proteins in {totaltime:.2f} seconds.')
    return data
```

Parsing UniProt

- Set up the data structure
- Open the data file and extract the necessary metadata
- Save to json

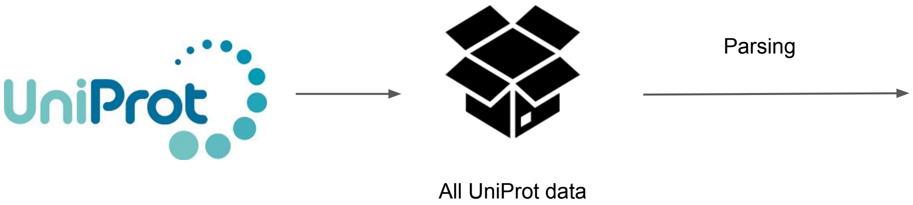


```
def parse_uniprot() -> dict:
    """..."""
    t0, count = time(), 0
    namespace = '{http://uniprot.org/uniprot}'
    data = defaultdict(lambda: {'symbol': [],
                                'RefSeq ID': [],
                                'sequence': None})
    logger.info("Begin processing the human UniProt data ...")

    with gzip.open(osp.join(DATA, 'uniprot_sprot_human.xml.gz'), 'rb') as f:
        for event, elem in tqdm(etree.iterparse(f, events=("start", "end")), desc='parsing UniProt', leave=False):
            if elem.tag == namespace + 'accession':
                acc = elem.text
            if elem.tag == namespace + 'gene':
                for record in elem.findall(namespace + 'name'):
                    if record.get('type') == 'primary' or record.get('type') == 'synonym':
                        if record.text not in data[acc]['symbol']:
                            data[acc]['symbol'].append(record.text)
            if elem.tag == namespace + 'dbReference':
                if elem.attrib['type'] == 'RefSeq' and elem.attrib['id'] not in data[acc]['RefSeq ID']:
                    data[acc]['RefSeq ID'].append(elem.attrib['id'])
            if elem.tag == namespace + 'sequence':
                data[acc]['sequence'] = elem.text
            # delete parts of the tree to save memory
            while elem.getprevious() is not None:
                del elem.getparent()[0]  # clean up preceding siblings

    with open(osp.join(UNIPROT, f'uniprot.json'), 'w') as filehandle:
        json.dump(data, filehandle)
    totaltime = (time() - t0)
    logger.info(f'...Finished parsing the UniProt DB for human proteins in {totaltime:.2f} seconds.')
    return data
```

Parsing UniProt



All UniProt data

Parsing

Uniprot Metadata

Uniprot ID: Q9NY95

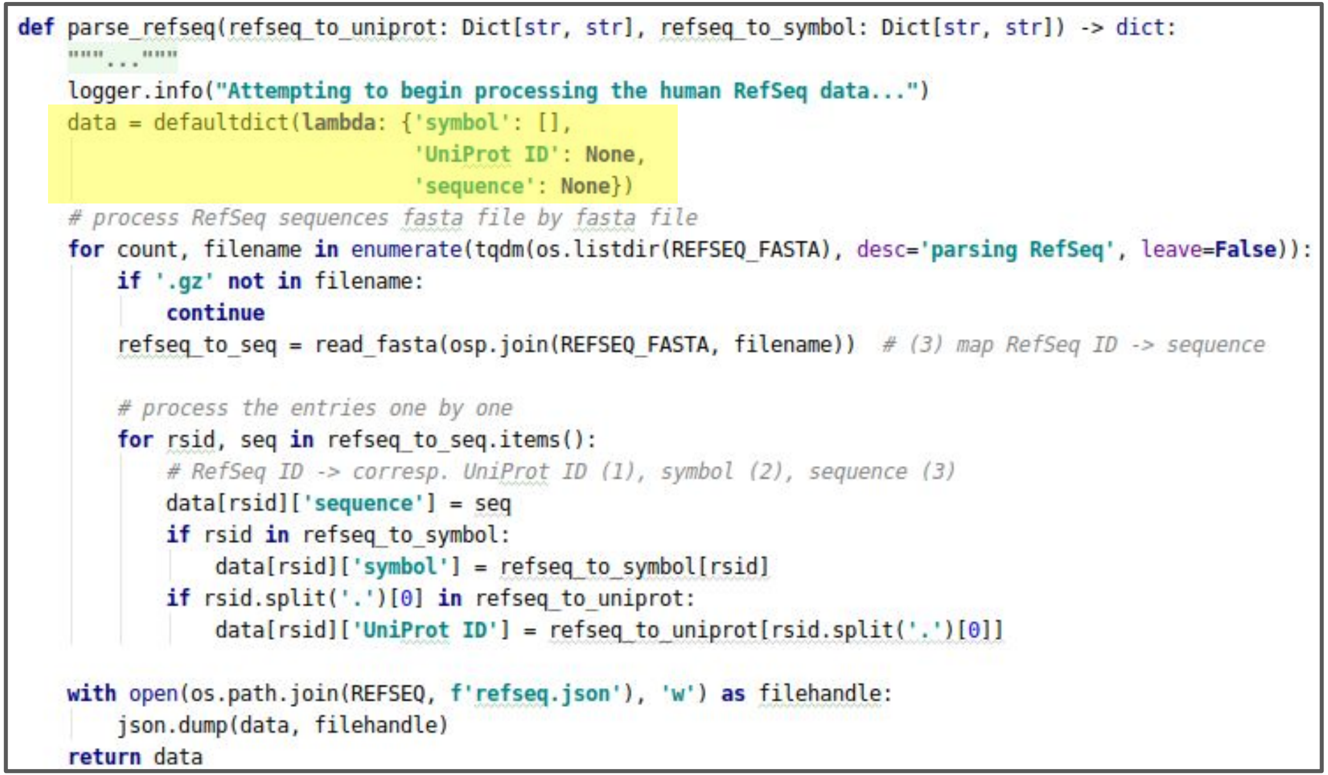
HGNC gene symbols:
ECEL1, XCE

Equiv. RefSeq IDs:
NP_001277716.1, NP_004817.2

Sequence:
MEPPYSLTAHYDEFQEVKYVSRC
GAGGARGASLPPGFPLGAARSA
TGARSGLPRWNRREVCLLSGLV
FAAGLCAILAAMLALKYLGPVAA
GGGACPEGCPERK…..

Parsing RefSeq

- Set up the data structure
- Open the data files and extract the necessary metadata
- Save to json



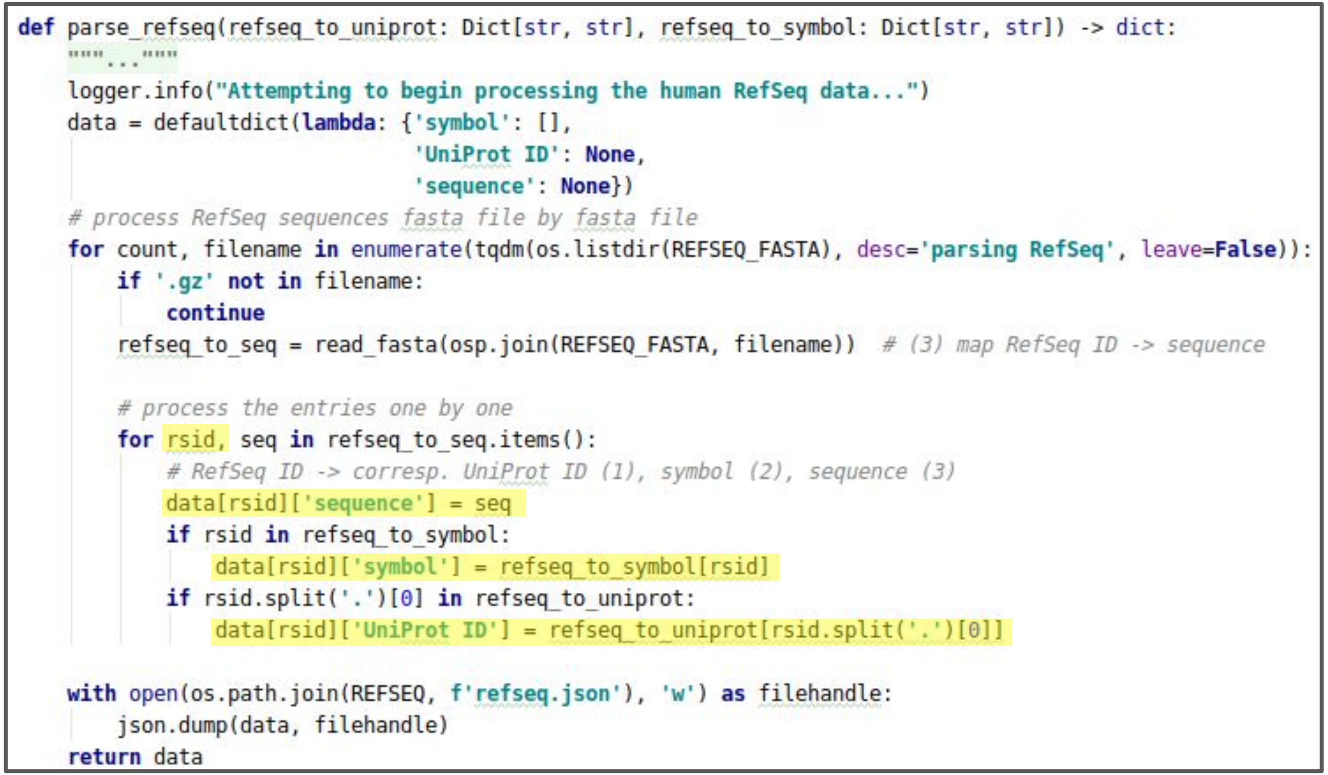
```python
def parse_refseq(refseq_to_uniprot: Dict[str, str], refseq_to_symbol: Dict[str, str]) -> dict:
    """..."""
    logger.info("Attempting to begin processing the human RefSeq data...")
    data = defaultdict(lambda: {'symbol': [],
                                'UniProt ID': None,
                                'sequence': None})
    # process RefSeq sequences fasta file by fasta file
    for count, filename in enumerate(tqdm(os.listdir(REFSEQ_FASTA), desc='parsing RefSeq', leave=False)):
        if '.gz' not in filename:
            continue
        refseq_to_seq = read_fasta(osp.join(REFSEQ_FASTA, filename))  # (3) map RefSeq ID -> sequence

        # process the entries one by one
        for rsid, seq in refseq_to_seq.items():
            # RefSeq ID -> corresp. UniProt ID (1), symbol (2), sequence (3)
            data[rsid]['sequence'] = seq
            if rsid in refseq_to_symbol:
                data[rsid]['symbol'] = refseq_to_symbol[rsid]
            if rsid.split('.')[0] in refseq_to_uniprot:
                data[rsid]['UniProt ID'] = refseq_to_uniprot[rsid.split('.')[0]]

    with open(os.path.join(REFSEQ, f'refseq.json'), 'w') as filehandle:
        json.dump(data, filehandle)
    return data
```

Parsing RefSeq

- Set up the data structure
- Open the data files and extract the necessary metadata
- Save to json



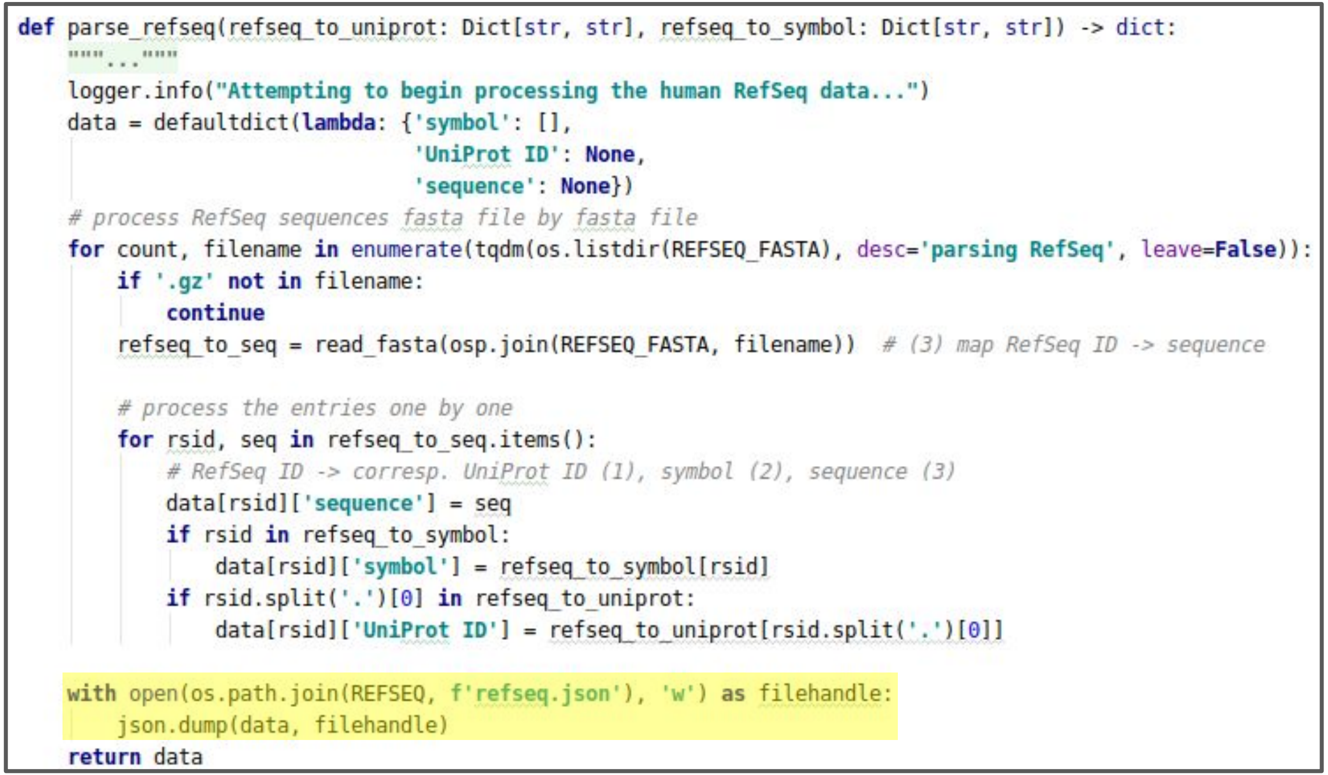
```
def parse_refseq(refseq_to_uniprot: Dict[str, str], refseq_to_symbol: Dict[str, str]) -> dict:
    """..."""
    logger.info("Attempting to begin processing the human RefSeq data...")
    data = defaultdict(lambda: {'symbol': [],
                                'UniProt ID': None,
                                'sequence': None})
    # process RefSeq sequences fasta file by fasta file
    for count, filename in enumerate(tqdm(os.listdir(REFSEQ_FASTA), desc='parsing RefSeq', leave=False)):
        if '.gz' not in filename:
            continue
        refseq_to_seq = read_fasta(osp.join(REFSEQ_FASTA, filename))  # (3) map RefSeq ID -> sequence

        # process the entries one by one
        for rsid, seq in refseq_to_seq.items():
            # RefSeq ID -> corresp. UniProt ID (1), symbol (2), sequence (3)
            data[rsid]['sequence'] = seq
            if rsid in refseq_to_symbol:
                data[rsid]['symbol'] = refseq_to_symbol[rsid]
            if rsid.split('.')[0] in refseq_to_uniprot:
                data[rsid]['UniProt ID'] = refseq_to_uniprot[rsid.split('.')[0]]

    with open(os.path.join(REFSEQ, f'refseq.json'), 'w') as filehandle:
        json.dump(data, filehandle)
    return data
```

Parsing RefSeq

- Set up the data structure
- Open the data files and extract the necessary metadata
- Save to json

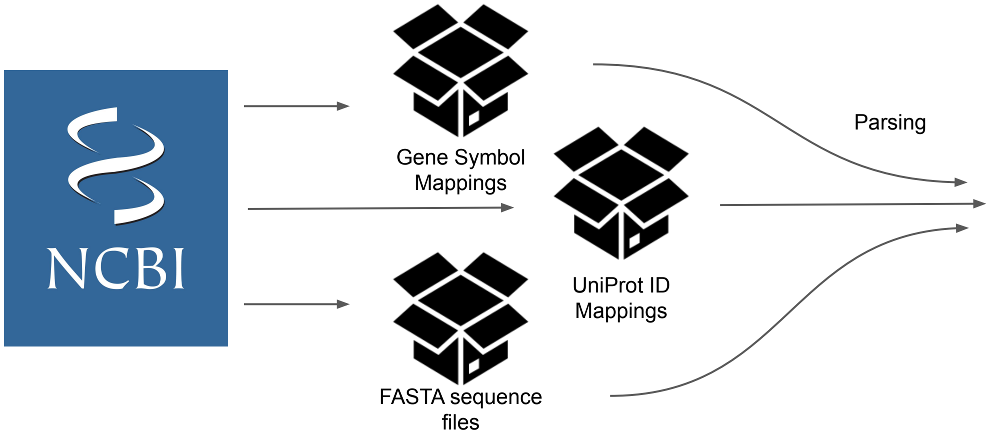


```
def parse_refseq(refseq_to_uniprot: Dict[str, str], refseq_to_symbol: Dict[str, str]) -> dict:
    """..."""
    logger.info("Attempting to begin processing the human RefSeq data...")
    data = defaultdict(lambda: {'symbol': [],
                                'UniProt ID': None,
                                'sequence': None})
    # process RefSeq sequences fasta file by fasta file
    for count, filename in enumerate(tqdm(os.listdir(REFSEQ_FASTA), desc='parsing RefSeq', leave=False)):
        if '.gz' not in filename:
            continue
        refseq_to_seq = read_fasta(osp.join(REFSEQ_FASTA, filename))  # (3) map RefSeq ID -> sequence

        # process the entries one by one
        for rsid, seq in refseq_to_seq.items():
            # RefSeq ID -> corresp. UniProt ID (1), symbol (2), sequence (3)
            data[rsid]['sequence'] = seq
            if rsid in refseq_to_symbol:
                data[rsid]['symbol'] = refseq_to_symbol[rsid]
            if rsid.split('.')[0] in refseq_to_uniprot:
                data[rsid]['UniProt ID'] = refseq_to_uniprot[rsid.split('.')[0]]

    with open(os.path.join(REFSEQ, f'refseq.json'), 'w') as filehandle:
        json.dump(data, filehandle)
    return data
```

Parsing RefSeq



RefSeq Metadata

RefSeq ID: NP_000585.2

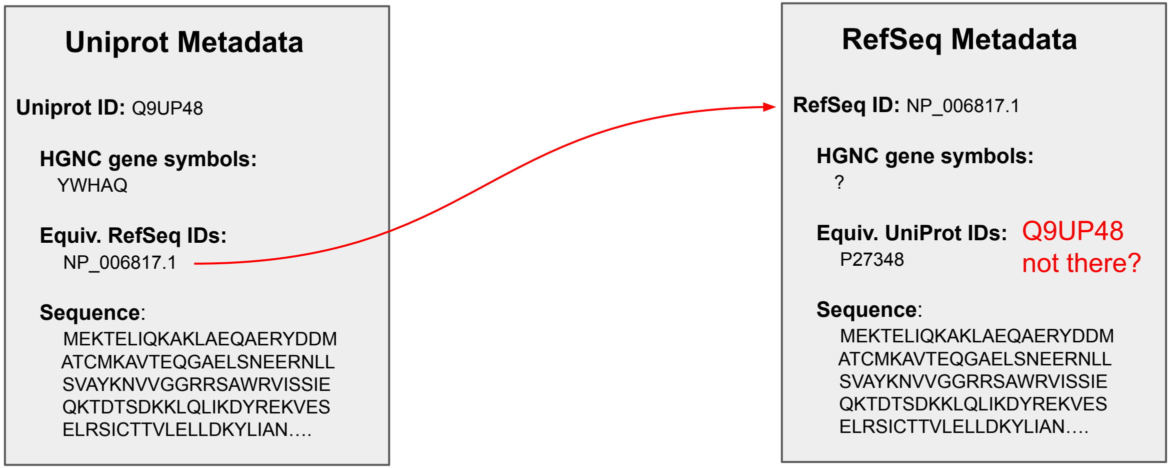
HGNC gene symbols:
TNF

Equiv. UniProt IDs:
Q5STB3

Sequence:
MSTESMIRDVELAEEALPKKTGG
PQGSRRCLFLSLFSFLIVAGATTL
FCLLHFGVIGPQREEFPRDLSLIS
PLAQAVRSSSRTPSDKPVAHVV
ANPQAEGQL...

Mapping

Inconsistent mapping!



Uniprot Metadata

Uniprot ID: Q9UP48

HGNC gene symbols:
YWHAQ

Equiv. RefSeq IDs:
NP_006817.1

Sequence:
MEKTELIQKAKLAEQAERYDDM
ATCMKAVTEQGAELSNEERNLL
SVAYKNVVGGRRSAWRVISSIE
QKTDTSDKKLQLIKDYREKVES
ELRSICTTVLELLDKYLIAN....

RefSeq Metadata

RefSeq ID: NP_006817.1

HGNC gene symbols:
?

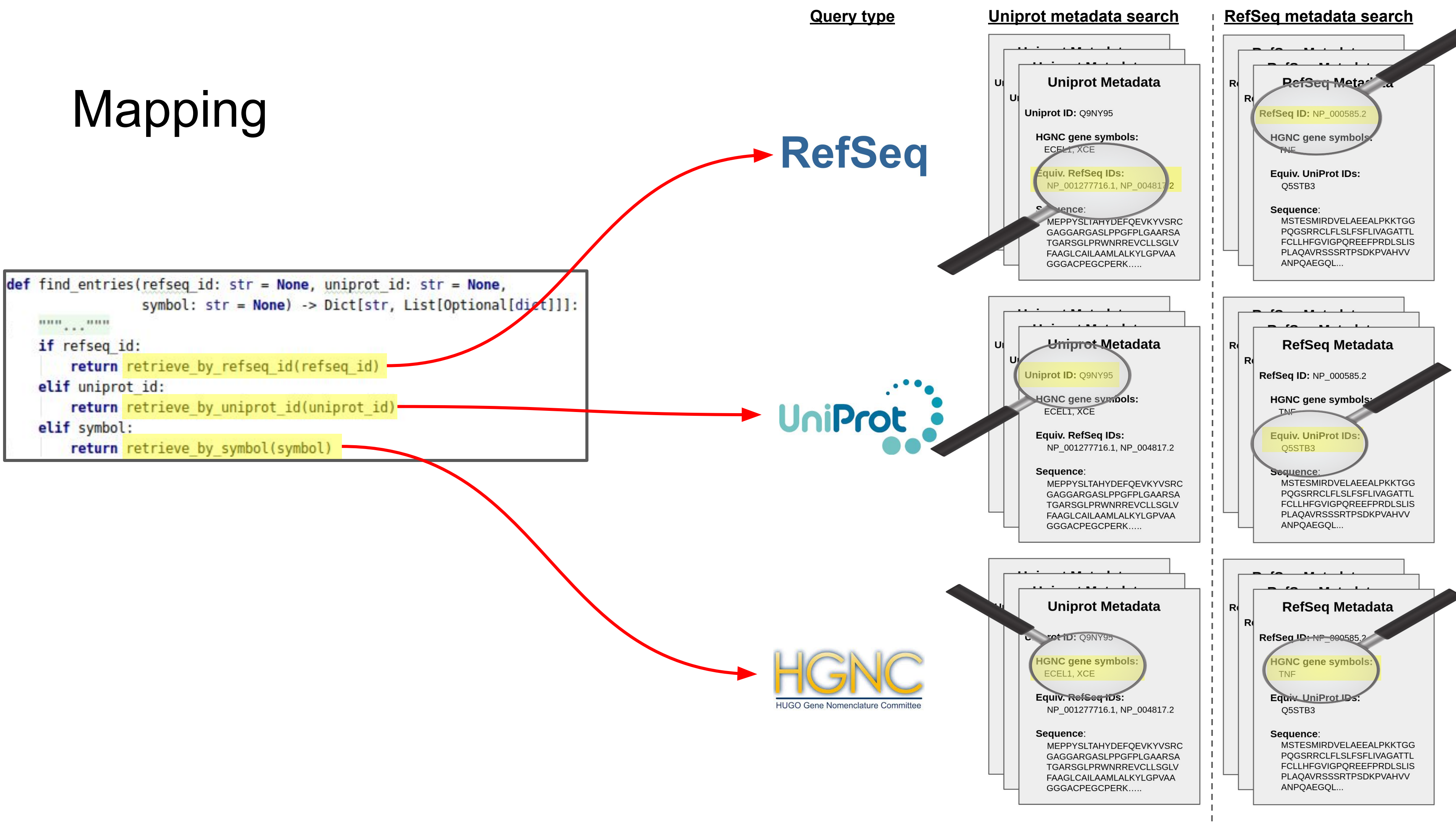
Equiv. UniProt IDs:
P27348

Q9UP48 not there?

Sequence:
MEKTELIQKAKLAEQAERYDDM
ATCMKAVTEQGAELSNEERNLL
SVAYKNVVGGRRSAWRVISSIE
QKTDTSDKKLQLIKDYREKVES
ELRSICTTVLELLDKYLIAN....

→Important to check mappings in both directions

Mapping



```python
def find_entries(refseq_id: str = None, uniprot_id: str = None,
                 symbol: str = None) -> Dict[str, List[Optional[dict]]]:
    """..."""
    if refseq_id:
        return retrieve_by_refseq_id(refseq_id)
    elif uniprot_id:
        return retrieve_by_uniprot_id(uniprot_id)
    elif symbol:
        return retrieve_by_symbol(symbol)
```

Uniprot Metadata

Uniprot ID: Q9NY95

HGNC gene symbols: ECEL1, XCE

Equiv. RefSeq IDs: NP_001277716.1, NP_004817.2

Sequence: MEPPYSLTAHYDEFQEVKYVSRC GAGGARGASLPPGFPLGAARSA TGARSGLPRWNRREVCLLSGLV FAAGLCAILAAMLALKYLGPVAA GGGACPEGCPERK.....

RefSeq Metadata

RefSeq ID: NP_000585.2

HGNC gene symbols: TNF

Equiv. UniProt IDs: Q5STB3

Sequence: MSTESMIRDVELAEEALPKKTGG PQGSRRCLFLSLFSFLIVAGATTL FCLLHFGVIGPQREEFPRDLSLIS PLAQAVRSSSRTPSDKPVAHVV ANPQAEGQL...

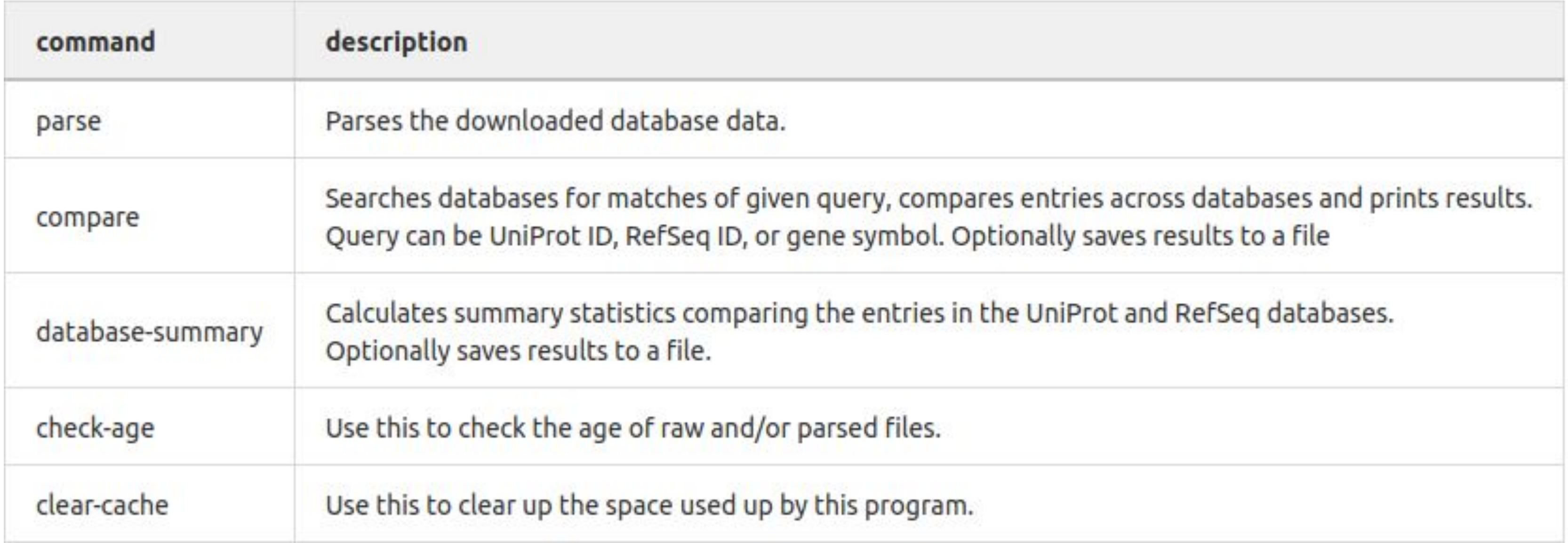
Comparison and Database similarity summary

Using this mapping, We can use these to make

- full-comparisons between database entries,
- get an overview comparison of how well the databases reference each other

Let’s look at some examples on the CLI !

CLI overview



command	description
parse	Parses the downloaded database data.
compare	Searches databases for matches of given query, compares entries across databases and prints results. Query can be UniProt ID, RefSeq ID, or gene symbol. Optionally saves results to a file
database-summary	Calculates summary statistics comparing the entries in the UniProt and RefSeq databases. Optionally saves results to a file.
check-age	Use this to check the age of raw and/or parsed files.
clear-cache	Use this to clear up the space used up by this program.

GUI Demo

Simon

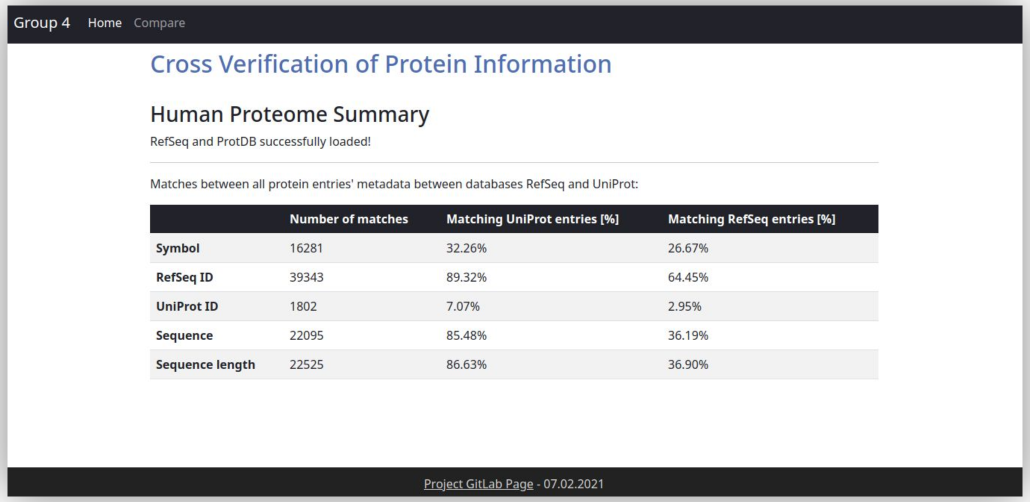
GUI Overview

- Creating a visual appealing alternative for the command line
  - Easier to use
  - Can be published to other devices
- Technology
  - Flask
  - HTML5, CSS 3
  - Jquery.js
  - Bootstrap 5
  - toastr.js
- Backend organization
  - Inspired by REST APIs

GUI - Backend + frontend architecture

- Only use HTTP methods to load data
  - GET
  - POST
  - PUT
  - DELETE
- Create easily accessible resources
- Manipulate data as little as possible -> gateway
- Two types of flask functions
  - Render and load the two main pages
  - Loading data from the python library

- Reform HTML content with **jquery.js** -> remove python artifacts
- Use **Bootstrap** to create appealing tables, forms and buttons
- **Toastr.js** transforms backend errors into beautiful popup messages



Cross Verification of Protein Information

Human Proteome Summary

RefSeq and ProtDB successfully loaded!

Matches between all protein entries' metadata between databases RefSeq and UniProt:

	Number of matches	Matching UniProt entries [%]	Matching RefSeq entries [%]
Symbol	16281	32.26%	26.67%
RefSeq ID	39343	89.32%	64.45%
UniProt ID	1802	7.07%	2.95%
Sequence	22095	85.48%	36.19%
Sequence length	22525	86.63%	36.90%

Project GitLab Page - 07.02.2021

Live Demo - Keep your distance from the wild proteins!

DBInspector +

Current Features (Pros):

- **Bi-directional** information **mapping**
  -> get the best information from both DBs
- Provides visual **side-by-side sequence** comparison
- Explicitly tells the user which version of RefSeq matches the UniProt entry
- Speedup powered by **caching**
- **Three interfaces** for maximal usability
- Takes RefSeq versions into account

Future Additions:

- Batch ID lookup
- Link cross-references to the respective information pages
- Extend to other organisms
- Extend to non-verified/predicted proteins

Thanks for your attention!

Are there any questions?