CHAPTER 20

Retrieving Data from Web Resources

LEARNING GOAL: You can search and fetch database records from NCBI via Biopython.

20.1 IN THIS CHAPTER YOU WILL LEARN

- How to read sequence files from the web
- How to submit PubMed queries
- How to submit queries to the NCBI nucleotide database
- How to retrieve Uniprot records and write them to a file

20.2 STORY: SEARCHING PUBLICATIONS BY KEYWORDS IN PUBMED AND DOWNLOADING AND PARSING THE CORRESPONDING RECORDS

20.2.1 Problem Description

In the previous chapter, you used Biopython to manipulate local sequence files (e.g., FASTA and GenBank files). In this chapter, you will use Biopython to access online NCBI databases, such as PubMed and GenBank, and Expasy resources, such as Uniprot, and retrieve and parse their contents. The following Python session shows how to find publications about PyCogent, a Python library complementary to Biopython. First, PubMed entries containing the keyword "PyCogent" need to be found and retrieved, and the

BOX 20.1 DOCUMENTATION AND SAMPLE QUERIES

Documentation

www.ncbi.nlm.nih.gov/books/NBK25500/

Searching for Papers in PubMed

http://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=pubmed&term=thermophilic,packing&rettype=uilist

Retrieving Publication Records in Medline Format

http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=pubmed&id=11748933,11700088&retmode=text&rettype=medline

Searching for Protein Database Entries by Keywords

http://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=protein&term=cancer+AND+human

Retrieving Protein Database Entries in FASTA Format

http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=protein&id=1234567&rettype=fasta

Retrieving Protein Database Entries in GenBank Format

http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=protein&id=1234567&rettype=gb

Retrieving Nucleotide Database Entries

http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=nucleotide&id=9790228&rettype=gb

resulting records need to be parsed. Since PubMed is one of the NCBI databases (www.ncbi.nlm.nih.gov/), it is connected to the Entrez data retrieval system (www.ncbi.nlm.nih.gov/Entrez). See Box 20.1 for sample queries to the NCBI server. The Biopython module to access NCBI web services is called `Entrez` as well. The `Entrez` module is needed to access and download NCBI database records. To further parse publication records, you need a specialized parser from the `Bio.Medline` module.

20.2.2 Python Session

```
from Bio import Entrez
from Bio import Medline
keyword = "PyCogent"
```

```
# search publications in PubMed
Entrez.email = "my_email@address.com"
handle = Entrez.esearch(db="pubmed", term=keyword)
record = Entrez.read(handle)
pmids = record['IdList']
print pmids
# retrieve Medline entries from PubMed
handle = Entrez.efetch(db="pubmed", id=pmids,\
    rettype="medline", retmode="text")
medline_records = Medline.parse(handle)
records = list(medline_records)
n = 1
for record in records:
   if keyword in record["TI"]:
      print n, ')', record["TI"]
      n += 1
```

Source: Adapted from code published by A.Via/K.Rother under the Python License.

The code produces the output:

```
['22479120', '18230758', '17708774']
1 ) Abstractions, algorithms and data structures for
    structural bioinformatics in PyCogent.
2 ) PyCogent: a toolkit for making sense from sequence.
```

20.3 WHAT DO THE COMMANDS MEAN?

20.3.1 The `Entrez` Module

`Entrez` provides a connection to the `esearch` and `efetch` tools on the NCBI servers. You can list methods and attributes available in the `Entrez` module by typing

```
>>> from Bio import Entrez
>>> dir(Entrez)
```

In the output, you will notice the `mail` attribute and the `esearch()` and `efetch()` functions used in Section 20.2.2. The `mail` attribute tells your email address to NCBI. This is not mandatory, but NCBI wants to be able to contact users in case of problems, and supplying your email address is fair. You can also provide your email address with each single access to NCBI by including `email = "my_email@address.com"` in the list of arguments of `Entrez.esearch()`.

`Entrez.esearch()`

The `Entrez.esearch()` conducts searches in NCBI databases using a query text. The function takes two mandatory arguments: `db`, the database to search (default is `pubmed`), and `term`, the query text.

In Section 20.2.2, the term to search is "PyCogent." If you want to search more than one keyword you can use "`AND`" or "`OR`", as you would do in an online search. You can also use keyword specifications such as `[Year]`, `[Organism]`, `[Gene]`, etc. (see Example 20.2). `Entrez.esearch()` returns a list of database identifiers in the form of a "handle," which can be read using the `Entrez.read()` function. The latter returns a dictionary with the keys that include, among others, "`IdList`" (its value is a list of IDs matching the text query) and "`Count`" (its value is the total number of IDs). In the case of the PubMed query shown in Section 20.2.2, the PMIDs are contained in the `record['IdList']` value of the `record` dictionary.

You can use the optional parameter `retmax` (maximum retrieved) to set how many entries matching the query text are to be retrieved (see Example 20.2). Other useful optional arguments are `datetype`, `reldate`, `mindate`, and `maxdate` (both in the form "YYYY/MM/DD"). `datetype` can be used to choose a type of date ("`mdat`": modification date, "`pdat`": publication date, "`edat`": Entrez date) to limit your search. `reldate` must be an integer `n` and tells the `esearch()` method to return only the IDs of the records matching `datetype` within the last `n` days. `mindate` and `maxdate` specify a date range that can be used to limit the search by the date type specified by `datetype`. For instance, the following query returns only papers in the period during which this book was written:

```
>>> handle = Entrez.esearch(db = "pubmed", \
... term = "Python", datetype = "pdat", \
... mindate = "2011/08/01", maxdate = "2013/10/28")
>>> record = Entrez.read(handle)
>>> record['Count']
'3'
```

To count matches, you can also use the `Entrez.egquery()` method, which returns the number of matches of the search term in each of the Entrez databases. It takes a single mandatory argument, which is the term to be searched:

```
>>> handle = Entrez.egquery(term = "PyCogent")
>>> record = Entrez.read(handle)
```

```
>>> for r in record["eGQueryResult"]:
...  print r["DbName"], r["Count"]
pubmed 3
pmc 49
mesh 0
...
```

`Entrez.efetch()`

So far, you have seen how to identify the IDs of records for one or more search terms. If you want to download these records from the NCBI server, you can use the `Entrez.efetch()` tool. In Biopython, the `Entrez.efetch()` function takes as arguments the database from which the records are to be retrieved and one ID or the list of IDs of the records you want to download. In the Python session in Section 20.2.2, the list of PMIDs to be downloaded is saved in the `pmids` variable. It is subsequently used as an ID list for `efetch()`.

The `Entrez.efetch()` function has many optional arguments: `retmode` specifies the format of the record(s) retrieved (text, HTML, XML), and `rettype` specifies what types of records are shown. It depends on the database you are accessing. For PubMed the `rettype` value can be, for example, `abstract`, `citation`, or `medline`. For Uniprot you can set `rettype` to `fasta` to retrieve the sequence of a protein record (see Example 20.4); `retmax` is the total number of records to be retrieved (up to a maximum of 10,000).

`Entrez.fetch()` returns a handle that "contains" your records. You can read the raw data from the handle like you would read an open Python file (with a `for` loop) or parse them using specialized functions.

20.3.2 The `Medline` Module

To parse the PubMed records you downloaded with `Entrez.efetch()`, you have to import the Biopython `Medline` module, which provides the `Medline.parse()` function. The result of this function can be conveniently converted into a list. This list contains `Bio.Medline.Record` objects that work like dictionaries. The most common keys are `TI` (Title), `PMID`, `PG` (pages), `AB` (Abstract), and `AU` (Authors). Not all keys are present in each dictionary. For example, if there is no abstract available for a PubMed record, the `AB` key will be

missing in the dictionary. The keys available for a given record can be visualized by typing the following:

```
>>> handle = Entrez.efetch(db = "pubmed", \
... retype = "medline", id = ['22479120', \
... '18230758', '17708774'], \
... retmode = 'text')
>>> records = Medline.parse(handle)
>>> list(records)[0].keys()
['STAT', 'IP', 'DEP', 'DA', 'AID', 'CRDT', 'DP', 'OWN',
    'PT', 'LA', 'FAU', 'JT', 'PG', 'PMC', 'TA', 'JID',
    'AB', 'VI', 'IS', 'TI', 'AU', 'MHDA', 'PHST', 'EDAT',
    'SO', 'PMID', 'PST']
```

Or if you also want to display the corresponding values:

```
>>> for record in records:
...    for k, v in record.items():
...       print k, v
```

Notice that if you have to parse a single record, you can use the `Medline.read()` function instead of `Medline.parse()`.

20.4 EXAMPLES

Example 20.1 What Are the Available Entrez Databases?

If you want to get information about the Entrez databases, you can use the function `Entrez.einfo()`. If you use it without arguments, you will get a dictionary with a single `key:value` pair where the value is a list of available databases in Entrez. If you pass a given database name as argument to `Entrez.einfo()`, you will get information about that database:

```
from Bio import Entrez
handle = Entrez.einfo()
info = Entrez.read(handle)
print info
raw_input('... press enter for a list of fields in PubMed')
handle = Entrez.einfo(db="pubmed")
record = Entrez.read(handle)
print record.keys()
print record['DbInfo']['Description']
print record['DbInfo']
```

Source: Adapted from code published by A.Via/K.Rother under the Python License.

The program generates the output:

```
{u'DbList': ['pubmed', 'protein', 'nuccore', 'nucleotide',
   'nucgss', 'nucest', 'structure', 'genome', 'assembly',
   'genomeprj', 'bioproject', 'biosample', 'blastdbinfo',
   'books', 'cdd', 'clinvar', 'clone', 'gap', 'gapplus',
   'dbvar', 'epigenomics', 'gene', 'gds', 'geoprofiles',
   'homologene', 'medgen', 'journals', 'mesh', 'ncbisearch',
   'nlmcatalog', 'omia', 'omim', 'pmc', 'popset', 'probe',
   'proteinclusters', 'pcassay', 'biosystems', 'pccompound',
   'pcsubstance', 'pubmedhealth', 'seqannot', 'snp', 'sra',
   'taxonomy', 'toolkit', 'toolkitall', 'toolkitbook',
   'unigene', 'unists', 'gencoll']}
```

... and a long list of fields after pressing enter.

Example 20.2 Searching PubMed with More Than One Term, Combining Keywords with AND/OR

```
from Bio import Entrez
handle = Entrez.esearch(db="pubmed", term="PyCogent AND RNA")
record = Entrez.read(handle)
print record['IdList']
handle = Entrez.esearch(db="pubmed", term="PyCogent OR RNA")
record = Entrez.read(handle)
print record['Count']
handle = Entrez.esearch(db="pubmed", \
    term="PyCogent AND 2008[Year]")
record = Entrez.read(handle)
print record['IdList']
handle = Entrez.esearch(db="pubmed", term= \
    "C. elegans[Organism] AND 2008[Year] AND Mapk[Gene]")
record = Entrez.read(handle)
print record['Count']
```

Source: Adapted from code published by A.Via/K.Rother under the Python License.

The program writes the lists of PMIDs and respective paper counts for the four queries. The optional parameter `retmax` (maximum retrieved items) makes it possible to set the maximum number of retrieved matches of the query text:

```
handle = Entrez.esearch(db = "pubmed", \
    term = "PyCogent OR RNA", retmax = "3")
record = Entrez.read(handle)
print record['IdList']
```

which results in:

```
['23285493', '23285311', '23285230']
```

Example 20.3 Retrieving and Parsing Nucleotide Database Entries in GenBank Format

This procedure is nearly identical to the procedure to retrieve and parse PubMed records. The main difference is that the IDs you need to fetch are the GI numbers of the sequences. Multiple IDs must be passed in the form of a string of comma-separated GI numbers instead of a list, and the file format (`retmode`) must be set to `xml`.

```
from Bio import Entrez
# search sequences by a combination of keywords
handle = Entrez.esearch(db="nucleotide", \
    term="Homo sapiens AND mRNA AND MapK")
records = Entrez.read(handle)
print records['Count']
top3_records = records['IdList'][0:3]
print top3_records
# retrieve the sequences by their GI numbers
gi_list = ','.join(top3_records)
print gi_list
handle = Entrez.efetch(db="nucleotide", \
    id=gi_list, rettype="gb", retmode="xml")
records = Entrez.read(handle)
print len(records)
print records[0].keys()
print records[0]['GBSeq_organism']
```

Source: Adapted from code published by A.Via/K.Rother under the Python License.

At time of writing, this code generates the output:

```
1053
['472824973', '433282995', '433282994']
472824973,433282995,433282994
3
[u'GBSeq_moltype', u'GBSeq_comment', u'GBSeq_feature-table',
   u'GBSeq_primary', u'GBSeq_references', u'GBSeq_locus',
   u'GBSeq_keywords', u'GBSeq_secondary-accessions', u'GBSeq_
   definition', u'GBSeq_organism', u'GBSeq_strandedness',
   u'GBSeq_source', u'GBSeq_sequence',
```

```
   u'GBSeq_primary-accession', u'GBSeq_accession-version',
   u'GBSeq_length', u'GBSeq_create-date', u'GBSeq_division',
   u'GBSeq_update-date', u'GBSeq_topology', u'GBSeq_other-
   seqids', u'GBSeq_taxonomy']
Homo sapiens
```

For a single `gi`, you can also use the "text" (`retmode = "text"`) format:

```
handle = Entrez.efetch(db = "nucleotide", \
    id = "186972394", rettype = "gb", retmode = "text")
record = handle.read()
```

Example 20.4 Searching for NCBI Protein Database Entries by Keywords

This procedure is very similar to that shown for PubMed and nucleotide records (Section 20.2.2 and Example 20.3, respectively):

```
from Bio import Entrez
# search IDs of protein sequences by keywords
handle = Entrez.esearch(db="protein", \
    term="Human AND cancer AND p21")
records = Entrez.read(handle)
print records['Count']
id_list = records['IdList'][0:3]
# retrieve sequences
id_list = ",".join(id_list)
print id_list
handle = Entrez.efetch(db="protein", \
    id=id_list, rettype="fasta", retmode="xml")
records = Entrez.read(handle)
rec = list(records)
print rec[0].keys()
print rec[0]['TSeq_defline']
```

Source: Adapted from code published by A.Via/K.Rother under the Python License.

This code creates the output:

```
920
229577056,131890016,113677036
[u'TSeq_accver', u'TSeq_sequence', u'TSeq_length',
   u'TSeq_taxid', u'TSeq_orgname', u'TSeq_gi',
   u'TSeq_seqtype', u'TSeq_defline']
CDC42 small effector protein 2 [Danio rerio]
```

Example 20.5 Retrieving SwissProt Database Entries and Writing Them to a File in FASTA Format

Biopython provides a module (called `ExPASy`) to access the SwissProt database and other Expasy resources (http://www.expasy.org/). The `get_sprot_raw()` method of the `ExPASy` module returns a handle, which can be read using the `SeqIO.read()` method (see Chapter 19). It is therefore necessary to import the `SeqIO` module first. As you learned in Chapter 19, the object returned by `SeqIO.read()` is a `SeqRecord` object and, as such, has `id`, `Seq`, and `description` attributes and can be written to a FASTA formatted file using the `SeqIO.write()` method.

```
from Bio import ExPASy
from Bio import SeqIO
handle = ExPASy.get_sprot_raw("P04637")
seq_record = SeqIO.read(handle, "swiss")
out = open('myfile.fasta','w')
fasta = SeqIO.write(seq_record, out, "fasta")
out.close()
```

Source: Adapted from code published by A.Via/K.Rother under the Python License.

Notice that if you want to do this for several SwissProt ACs, you have to retrieve and parse them one by one:

```
ac_list = ['P04637', 'P0CQ42', 'Q13671']
records = []
for ac in ac_list:
    handle = ExPASy.get_sprot_raw(ac)
    record = SeqIO.read(handle, "swiss")
    records.append(record)
out = open('myfile.fasta','w')
for rec in records:
    fasta = Bio.SeqIO.write(rec, out, "fasta")
out.close()
```

This code creates a local multiple FASTA file.

20.5 TESTING YOURSELF

Exercise 20.1 Search PubMed by Keywords

Use `Entrez.esearch()` to retrieve a list of PMIDs of papers about tRNA aminoacylation from 2008, using as search terms *trna*, *aminoacylation*, *"2008"[Publication Date]*. How many papers do you find?

Exercise 20.2 Get Paper Information and Save It to a File

Use `Entrez.efetch()` to get information from papers retrieved in Exercise 20.1 in the Medline format and save it to a file. How many lines does the file have?

Exercise 20.3 Fetch a Nucleotide Sequence

Use `Entrez.efetch()` to download the nucleotide sequence with the GI 433282994 and write it to a file in FASTA format.

Exercise 20.4 Search for Protein Sequences by Keyword

Use `Entrez.esearch()` to find protein sequences for the bacteriorhodopsin protein. Retrieve the first 20 sequences and save them to a file in GenBank format.

Exercise 20.5

Write a small program that performs a function similar to that of EndNote or Mendeley. The program should read a text with PMIDs in square brackets (e.g., `[23285311]`) and replace them by an increasing number in square brackets (e.g., `[1]`), plus a formatted reference at the end of the document, e.g.,

```
[1] Cieslik M, Derewenda ZS, Mura C. Abstractions, algorithms
    and data structures for structural bioinformatics in
    PyCogent. J Appl Crystallogr. 2011 Apr 1;44:424–428.
```