BIMM 185 Lab Report 3

Lihui Lu A99108553 4/25/2017

Intro

Last week we have learned how to access data in biological databases such as NCBI and UniProt using `rsync` and `wget`. We have also learned how to use BioPython, which can faster the process of reading inputs with a specific format. We have learned parsing data use `SeqIO` with GenBank format, fasta format, as well as UniProt format.

Accessing and parsing data on NCBI

We have learned how to use `rsync` to download desired data from the NCBI database using the summary file.

We can first download the summary file to the current directory using the command:

```
rsync -avzL rsync://rsync.ncbi.nlm.nih.gov/genomes/refseq/assembly_summary_refseq.txt .
```

The usage of options used above are listed as follow:

```
-a archive mode, equivalent of using all of the following options:
    -r, --recursive recurse into directories

    -l, --links copy symlinks as symlinks

    -p, --perms preserve permissions

    -t, --times preserve modification times

    -g, --group preserve group

    -o, --owner preserve owner (super-user only)
```

```
    -D same as --devices --specials

    --devices preserve device files (super-user only)

    --specials preserve special files)
-v verbose mode, print out the files being copied
-z compress the data through transfer
-L transform symlink into referent file/dir
```

After downloading the summary file, we then can use bash commands to process it and get the ftp links to the data of desire.

To download all the genomes of E.coli K12 MG1655 type but from different stains, we can use bash scripts or python scripts to parse the summary file.

Use bash script to parse the summary file:

```
cut -f 6,8,20 assembly_summary_refseq.txt | grep 'coli' | grep 'K-12' | grep 'M
G1655' > E.coli.K12.txt
```

Use python script to download genome:

```
import os


samples = []
with open('E.coli.K12.txt','r') as file:
        for line in file:
                line = line.strip().split('\t')
                tax_id = line[0]
                name = line[1]
                name = name.replace(" ", "_")
                print(name)
                name += "_"+tax_id
                ftp = line[2]
                ftp = ftp.replace("ftp","rsync")
                print(ftp)
                samples.append((name,ftp))

for s in samples:
        #print('rsync -avzL {ftp} {name}'.format(ftp=s[1],name=s[0]))
        os.system('rsync -avzL {ftp} ./genome'.format(ftp=s[1]))
```

Exercise 1

We practiced using BioPython to parse the GenBank file. The main function used to parse the file was `SeqIO.parse(inputfile, filetype)`. To specify the file format to be GenBank file, either 'gb' or 'genbank' can be used as the filetype. Other than using `SeqIO`, since the data that we downloaded was compressed and we didn't want to uncompress it which might take much space on the disk, the package `gzip` can be used for file ended with '.gz'. And `gzip.open(file)` can be used to read compressed file.

Exercise Description

Write a script that uses biopython to parse the GenBank file for E. coli K12 MG1655
Extract for every CDS:

1. Tax ID (Check ‘source’, and /db_xref for taxon)
2. the accession (/protein_id)
3. coordinates (CDS line)
4. strand (see word ‘complement’ in CDS line)
5. gene name (/gene)
6. locus tag (/locus_tag)
7. synonyms (/gene_synonym)
8. protein name (/product)
9. EC-number(s) (/EC_number)
10. external references (/db_xref)

Separated by tab.

The GenBank file has the following structure:

Example Output

```
accession   coordinates strand  gene_name   locus_tag   synonyms    protein_nam
e    Tax_ID  EC-numbers  external_references
NP_414542.1 189-255 +   thrL    b0001   ECK0001,JW4367  thr operon leader pepti
de   511145  -   GI:16127995,ASAP:ABE-0000006,UniProtKB/Swiss-Prot:P0AD86,EcoGe
ne:EG11277,GeneID:944742
NP_414543.1 336-2799    +   thrA    b0002   ECK0002,Hs,JW0001,thrA1,thrA2,thrD
 Bifunctional aspartokinase/homoserine dehydrogenase 1   511145  1.1.1.3,2.7.2.
4 GI:16127996,ASAP:ABE-0000008,UniProtKB/Swiss-Prot:P00561,EcoGene:EG10998,Gene
ID:945803
```

```
NP_414544.1 2800-3733   +   thrB    b0003   ECK0003,JW0002  homoserine kinase
 511145  2.7.1.39    GI:16127997,ASAP:ABE-0000010,UniProtKB/Swiss-Prot:P00547,E
coGene:EG10999,GeneID:947498
NP_414545.1 3733-5020   +   thrC    b0004   ECK0004,JW0003  L-threonine synthas
e    511145  4.2.3.1 GI:16127998,ASAP:ABE-0000012,UniProtKB/Swiss-Prot:P00934,E
coGene:EG11000,GeneID:945198
NP_414546.1 5233-5530   +   yaaX    b0005   ECK0005,JW0004  DUF2502 family puta
tive periplasmic protein 511145  -GI:16127999,ASAP:ABE-0000015,UniProtKB/Swiss-
Prot:P75616,EcoGene:EG14384,GeneID:944747
NP_414547.1 5682-6459   -   yaaA    b0006   ECK0006,JW0005  peroxide resistance
 protein, lowers intracellular iron  511145  -   GI:16128000,ASAP:ABE-0000018,U
niProtKB/Swiss-Prot:P0A8I3,EcoGene:EG10011,GeneID:944749
NP_414548.1 6528-7959   -   yaaJ    b0007   ECK0007,JW0006  putative transporte
r    511145  -   GI:16128001,ASAP:ABE-0000020,UniProtKB/Swiss-Prot:P30143,EcoGe
ne:EG11555,GeneID:944745
NP_414549.1 8237-9191   +   talB    b0008   ECK0008,JW0007,yaaK transaldolase B
 511145  2.2.1.2 GI:16128002,ASAP:ABE-0000027,UniProtKB/Swiss-Prot:P0A870,EcoGe
ne:EG11556,GeneID:944748
NP_414550.1 9305-9893   +   mog b0009   bisD,chlG,ECK0009,JW0008,mogA,yaaG  mol
ybdochelatase incorporating molybdenum into molybdopterin    511145  -   GI:161
28003,ASAP:ABE-0000030,UniProtKB/Swiss-Prot:P0AF03,EcoGene:EG11511,GeneID:94476
0
```

Source Code

```
from Bio import SeqIO
import gzip


#print string ends with tab
def print_function(s):
    print(s,end='\t')

#read the gzipped input
file = gzip.open('GCF_000005845.2_ASM584v2/GCF_000005845.2_ASM584v2_genomic.gbf
f.gz','rt')

#print header
print('accession','coordinates','strand','gene_name','locus_tag','synonyms','pr
otein_name','Tax_ID','EC-numbers','external_references',sep='\t')

#iterate through all the records in the input
for seq_record in SeqIO.parse(file,'genbank'):
```

```
    for f in seq_record.features:
       #check if it is CDS
        if f.type == "CDS":
          #check if it has protein_id
            if 'protein_id' in f.qualifiers:
                print_function(','.join(f.qualifiers['protein_id']))
            elif 'pseudo' in f.qualifiers:
                #print(','.join(f.qualifiers['gene']),end='\t')
                print_function('pseudo')
            else:
                #print(','.join(f.qualifiers['gene']),end='\t')
                print_function('i don\'t know')
            #get location
            print(f.location.start,f.location.end,sep='-',end='\t')

            #forward strand = 1, reverse strand = -1
            if f.location.strand == 1:
                print_function('+')
            elif f.location.strand == -1:
                print_function('-')

            #print gene name
            if 'gene' in f.qualifiers:
                print_function(','.join(f.qualifiers['gene']))
            else:
                print_function('-')

            #print locus tag
            if 'locus_tag' in f.qualifiers:
                print_function(','.join(f.qualifiers['locus_tag']))
            else:
                print_function('-')

            #print synonym
            if 'gene_synonym' in f.qualifiers:
                print_function(','.join(f.qualifiers['gene_synonym']).replace('
; ',','))
                #GSs = f.qualifiers['gene_synonym']
                #print(','.join(GSs),end='\t')
            else:
                print_function('-')

            #print product name
            if 'product' in f.qualifiers:
                print_function(','.join(f.qualifiers['product']))
```

```
            else:
                print_function('-')


            print_function(taxid)

           #print EC_number
            if 'EC_number' in f.qualifiers:
                print_function(','.join(f.qualifiers['EC_number']))
            else:
                print_function('-')

           #print external references
            if 'db_xref' in f.qualifiers:
                print_function(','.join(f.qualifiers['db_xref']))
            else:
                print_function('-')


            #change new line
            print()
        #obtain taxid from source line
        elif f.type == 'source':
            taxid = ','.join(f.qualifiers['db_xref']).replace('taxon:','')



file.close()
```

Exercise 2

Fasta file(protein sequences) can also be parsed using `SeqIO.parse` function.

Exercise Description

Write a script that uses biopython to parse the fasta file with all the protein sequences of E. coli K12 MG1655 (GCF_000005845.2_ASM584v2_protein.faa.gz). Write the sequences to a tab-delimited file with two-columns:

1. Accession (e.g. NP_414542.1)
2. Protein sequence in one string.

Example Output

```
NP_414542.1 MKRISTTITTTTITITTGNGAG
NP_414543.1 MRVLKFGGTSVANAERFLRVADILESNARQGQVATVLSAPAKITNHLVAMIEKTISGQDALPNISDA
ERIFAELLTGLAAAQPGFPLAQLKTFVDQEFAQIKHVLHGISLLGQCPDSINAALICRGEKMSIAIMAGVLEARGHNVT
VIDPVEKLLAVGHYLESTVDIAESTRRIAASRIPADHMVLMAGFTAGNEKGELVVLGRNGSDYSAAVLAACLRADCCEI
WTDVDGVYTCDPRQVPDARLLKSMSYQEAMELSYFGAKVLHPRTITPIAQFQIPCLIKNTGNPQAPGTLIGASRDEDEL
PVKGISNLNNMAMFSVSGPGMKGMVGMAARVFAAMSRARISVVLITQSSSEYSISFCVPQSDCVRAERAMQEEFYLELK
EGLLEPLAVTERLAIISVVGDGMRTLRGISAKFFAALARANINIVAIAQGSSERSISVVVNNDDATTGVRVTHQMLFNT
DQVIEVFVIGVGGVGGALLEQLKRQQSWLKNKHIDLRVCGVANSKALLTNVHGLNLENWQEELAQAKEPFNLGRLIRLV
KEYHLLNPVIVDCTSSQAVADQYADFLREGFHVVTPNKKANTSSMDYYHQLRYAAEKSRRKFLYDTNVGAGLPVIENLQ
NLLNAGDELMKFSGILSGSLSYIFGKLDEGMSFSEATTLAREMGYTEPDPRDDLSGMDVARKLLILARETGRELELADI
EIEPVLPAEFNAEGDVAAFMANLSQLDDLFAARVAKARDEGKVLRYVGNIDEDGVCRVKIAEVDGNDPLFKVKNGENAL
AFYSHYYQPLPLVLRGYGAGNDVTAAGVFADLLRTLSWKLGV
NP_414544.1 MVKVYAPASSANMSVGFDVLGAAVTPVDGALLGDVVTVEAAETFSLNNLGRFADKLPSEPRENIVYQ
CWERFCQELGKQIPVAMTLEKNMPIGSGLGSSACSVVAALMAMNEHCGKPLNDTRLLALMGELEGRISGSIHYDNVAPC
FLGGMQLMIEENDIISQQVPGFDEWLWVLAYPGIKVSTAEARAILPAQYRRQDCIAHGRHLAGFIHACYSRQPELAAKL
MKDVIAEPYRERLLPGFRQARQAVAEIGAVASGISGSGPTLFALCDKPETAQRVADWLGKNYLQNQEGFVHICRLDTAG
ARVLEN
NP_414545.1 MKLYNLKDHNEQVSFAQAVTQGLGKNQGLFFPHDLPEFSLTEIDEMLKLDFVTRSAKILSAFIGDEI
PQEILEERVRAAFAFPAPVANVESDVGCLELFHGPTLAFKDFGGRFMAQMLTHIAGDKPVTILTATSGDTGAAVAHAFY
GLPNVKVVILYPRGKISPLQEKLFCTLGGNIETVAIDGDFDACQALVKQAFDDEELKVALGLNSANSINISRLLAQICY
YFEAVAQLPQETRNQLVVSVPSGNFGDLTAGLLAKSLGLPVKRFIAATNVNDTVPRFLHDGQWSPKATQATLSNAMDVS
QPNNWPRVEELFRRKIWQLKELGYAAVDDETTQQTMRELKELGYTSEPHAAVAYRALRDQLNPGEYGLFLGTAHPAKFK
ESVEAILGETLDLPKELAERADLPLLSHNLPADFAALRKLMMNHQ
NP_414546.1 MKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDHGWWKQHYEWRGNR
WHLHGPPPPPRHHKKAPHDHHGGHGPGKHHR
NP_414547.1 MLILISPAKTLDYQSPLTTTRYTLPELLDNSQQLIHEARKLTPPQISTLMRISDKLAGINAARFHDW
QPDFTPANARQAILAFKGDVYTGLQAETFSEDDFDFAQQHLRMLSGLYGVLRPLDLMQPYRLEMGIRLENARGKDLYQF
WGDIITNKLNEALAAQGDNVVINLASDEYFKSVKPKKLNAEIIKPVFLDEKNGKFKIISFYAKKARGLMSRFIIENRLT
KPEQLTGFNSEGYFFDEDSSSNGELVFKRYEQR
NP_414548.1 MPDFFSFINSVLWGSVMIYLLFGAGCWFTFRTGFVQFRYIRQFGKSLKNSIHPQPGGLTSFQSLCTS
LAARVGSGNLAGVALAITAGGPGAVFWMWVAAFIGMATSFAECSLAQLYKERDVNGQFRGGPAWYMARGLGMRWMGVLF
AVFLLIAYGIIFSGVQANAVARALSFSFDFPPLVTGIILAVFTLLAITRGLHGVARLMQGFVPLMAIIWVLTSLVICVM
NIGQLPHVIWSIFESAFGWQEAAGGAAGYTLSQAITNGFQRSMFSNEAGMGSTPNAAAAAASWPPHPAAQGIVQMIGIF
IDTLVICTASAMLILLAGNGTTYMPLEGIQLIQKAMRVLMGSWGAEFVTLVVILFAFSSIVANYIYAENNLFFLRLNNP
KAIWCLRICTFATVIGGTLLSLPLMWQLADIIMACMAITNLTAILLLSPVVHTIASDYLRQRKLGVRPVFDPLRYPDIG
RQLSPDAWDDVSQE
NP_414549.1 MTDKLTSLRQYTTVVADTGDIAAMKLYQPQDATTNPSLILNAAQIPEYRKLIDDAVAWAKQQSNDRA
QQIVDATDKLAVNIGLEILKLVPGRISTEVDARLSYDTEASIAKAKRLIKLYNDAGISNDRILIKLASTWQGIRAAEQL
EKEGINCNLTLLFSFAQARACAEAGVFLISPFVGRILDWYKANTDKKEYAPAEDPGVVSVSEIYQYYKEHGYETVVMGA
SFRNIGEILELAGCDRLTIAPALLKELAESEGAIERKLSYTGEVKARPARITESEFLWQHNQDPMAVDKLAEGIRKFAI
DQEKLEKMIGDLL
NP_414550.1 MNTLRIGLVSISDRASSGVYQDKGIPALEEWLTSALTTPFELETRLIPDEQAIIEQTLCELVDEMSC
```

```
HLVLTTGGTGPARRDVTPDATLAVADREMPGFGEQMRQISLHFVPTAILSRQVGVIRKQALILNLPGQPKSIKETLEGV
KDAEGNVVVHGIFASVPYCIQLLEGPYVETAPEVVAAFRPKSARRDVSE
NP_414551.1 MGNTKLANPAPLGLMGFGMTTILLNLHNVGYFALDGIILAMGIFYGGIAQIFAGLLEYKKGNTFGLT
AFTSYGSFWLTLVAILLMPKLGLTDAPNAQFLGVYLGLWGVFTLFMFFGTLKGARVLQFVFFSLTVLFALLAIGNIAGN
AAIIHFAGWIGLICGASAIYLAMGEVLNEQFGRTVLPIGESH
```

```
from Bio import SeqIO
import gzip
file = gzip.open('GCF_000005845.2_ASM584v2/GCF_000005845.2_ASM584v2_protein.faa
.gz','rt')

def print_function(s):
    print(s,end='\t')

for seq_record in SeqIO.parse(file,'fasta'):
    print_function(seq_record.id)
    print_function(seq_record.seq)
    print()
```

Accessing and parsing data on UniProt

UniProt is a database for protein sequences and annotations. Since UniProt is not `rsync` compatible, `ftp` and `wget` can be used to download data from the database. Using the option `-P` with wget, it can save the downloaded data into a directory of desire. The summary file can be downloaded using the following command:

```
wget ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/
reference_proteomes/README
```

To parse UniProt formatted data, the `parse` function from `SwissProt` package in BioPython can be used.

Exercise 3

Exercise Description

Write a script to download the README file from the reference proteomes UniProt ftp site:

ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/reference_proteomes/README
Read the Proteome ID for and download from the following ftp address the corresponding genomic files for 3 bacteria of your choosing:
ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/reference_proteomes/Bacteria/

I have downloaded the genomic files for proteome UP000034024, UP000050566, and UP000000948. However, I have had trouble downloading proteome UP000029777 using both command

```
wget -P UP000029777 ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/reference_proteomes/Bacteria/UP000029777*
```

and

```
wget -P UP000029777 ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/reference_proteomes/Bacteria/UP000029777_*
```

The error was `No matches on pattern 'UP000029777_*'` and `No matches on pattern 'UP000029777*'`. After scanning through the README file and the ftp address, I have figured out that this proteome is actually the proteome of a virus. The genomic files for viruses can be downloaded following address into the viruses directory. The following command can be used:

```
wget -P UP000029777 ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/reference_proteomes/Viruses/UP000029777\*
```

Exercise 4

Exercise Description
Identify the biopython functions that allow parsing of UniProt functional annotations. Then download the taxonomic information for all Archaea:
ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/taxonomic_divisions/uniprot_sprot_archaea.dat.gz
Write a python script using biopython to extract the taxonomic information for all records in that file. Generate an tab-delimited file with the following columns:

1. NCBI tax_id (OX)
2. Organism (OS)
3. Taxonomy (OC)

`SwissProt.parse(file)` function can be used to parse the uniprot functional annotation file. In order to remove duplicates, a set of all the tax_ids that have been encountered before was used. If the new taxonomy has the same tax_id that has appeared before, skip the line to prevent duplicates.

Example Output

```
272844  Pyrococcus abyssi (strain GE5 / Orsay). Archaea,Euryarchaeota,Thermococ
ci,Thermococcales,Thermococcaceae,Pyrococcus
186497  Pyrococcus furiosus (strain ATCC 43587 / DSM 3638 / JCM 8422 / Vc1).
 Archaea,Euryarchaeota,Thermococci,Thermococcales,Thermococcaceae,Pyrococcus
70601   Pyrococcus horikoshii (strain ATCC 700860 / DSM 12428 / JCM 9974 / NBRC
 100139 / OT-3). Archaea,Euryarchaeota,Thermococci,Thermococcales,Thermococcace
ae,Pyrococcus
272557  Aeropyrum pernix (strain ATCC 700893 / DSM 11879 / JCM 9820 / NBRC 1001
38 / K1).    Archaea,Crenarchaeota,Thermoprotei,Desulfurococcales,Desulfurococc
aceae,Aeropyrum
351160  Methanocella arvoryzae (strain DSM 22066 / NBRC 105507 / MRE50).    Arc
haea,Euryarchaeota,Methanomicrobia,Methanocellales,Methanocellaceae,Methanocell
a
368407  Methanoculleus marisnigri (strain ATCC 35101 / DSM 1498 / JR1). Archaea
,Euryarchaeota,Methanomicrobia,Methanomicrobiales,Methanomicrobiaceae,Methanocu
lleus
309800  Haloferax volcanii (strain ATCC 29605 / DSM 3757 / JCM 8879 / NBRC 1474
2 / NCIMB 2012 / VKM B-1768 / DS2) (Halobacterium volcanii). Archaea,Euryarchae
ota,Halobacteria,Haloferacales,Haloferacaceae,Haloferax
243232  Methanocaldococcus jannaschii (strain ATCC 43067 / DSM 2661 / JAL-1 / J
CM 10045 / NBRC 100440) (Methanococcus jannaschii)Archaea,Euryarchaeota,Methano
cocci,Methanococcales,Methanocaldococcaceae,Methanocaldococcus
330779  Sulfolobus acidocaldarius (strain ATCC 33909 / DSM 639 / JCM 8929 / NBR
C 15157 / NCIMB 11770).  Archaea,Crenarchaeota,Thermoprotei,Sulfolobales,Sulfol
obaceae,Sulfolobus
```

Source Code

```
from Bio import SwissProt
import gzip
```

```
file = gzip.open("uniprot_sprot_archaea.dat.gz",'rt')


ids = set() #set to store all the tax_ids that have been encountered
for seq_record in SwissProt.parse(file):

    #if the same tax_id has appeared before, used to remove duplicate
    if ','.join(seq_record.taxonomy_id) in ids:
        continue
    else:
        print(','.join(seq_record.taxonomy_id),seq_record.organism,','.join(seq
_record.organism_classification),sep='\t')
        ids.add(','.join(seq_record.taxonomy_id))
```

Conclusion

Last week we have learned to use tools to access as well as download data from online biological databases. We have also learned that using the summary file, downloading multiple data satisfying certain criteria can be done using python or bash scripts. In addition, tools like BioPython can be used to speed up the process of parsing data with special format. Writing parsing process by oneself can be trivial but would also take a lot of time to debug. Using tools can also avoid introducing errors since when parsing the data manually, some special cases are not included. These tools can be utilized so that time can be spent on the developmental process other than debugging the parsing block.

All of the source codes can also be found in the git repository: https://github.com/luna5124/BIMM185 in the Week3 folder