MSc Applied Bioinformatics
alexey.larionov@cranfield.ac.uk

Practical 5: Sequence files manipulation

In Lecture 6, we have learnt how to use Python to read, write, and manipulate data from files. However, as a bioinformatician you will often deal with the specialised data formats. The aim of this practical will be to practice Python file operations for handling several widely used sequence data formats:

1. FASTA: Simple and very popular format for nucleic acid and protein sequences, used by many programs and databases,

2. Genbank: Used by Genbank (the largest collection of publicly available genetic data),

3. PDB: Used by Protein DataBank, a *de-facto* standard for storing information about three-dimensional structure of large biological molecules, including proteins and nucleic acids.

There are, of course, many other bioinformatics data formats, you may see some here:

https://www.ncbi.nlm.nih.gov/sra/docs/submitformats

https://genome.ucsc.edu/FAQ/FAQformat.html

Learning the data formats and specialised tools to handle them is a significant part of practical bioinformatics.

Task 1: The FASTA format

FASTA is very widely used, due to its simplicity, so let's start with it.

Rather than having an entire biological sequence in one long line, the FASTA format breaks this sequence into more manageable chunks (by using the newline character). The length isn't specified, but usually it is about 80 characters (to fit on screen :)

Also, the FASTA format includes a single header line prefixed with **>** (greater than) character (some FASTA processing software will accept more than one header line.

```
> sample dna | (This is a typical fasta header)
agatggcggcgctgaggggtcttgggggctctaggccggccacctactgg
tttgcagcggagacgacgcatggggcctgcgcaataggagtacgctgcct
gggaggcgtgactagaagcggaagtagttgtgggcgcctttgcaaccgcc
tgggacgccgccgagtggtctgtgcaggttcgcgggtcgctggcgggggt
cgtgagggagtgcgccgggagcggagatatggagggagatggttcagacc
```

The header information normally contains the name of the sequence (e.g. gene name) plus some other information separated by the "|" (pipe) character.

One FASTA file may contain multiple sequences with headers.

MSc Applied Bioinformatics
alexey.larionov@cranfield.ac.uk

You already know how to read data from a file in Python. It only takes 3 simple steps discussed in the Lecture 5 today:

1. Open a file
2. Read the data
3. Close the file

Now you will apply these steps to read a FASTA file provided for this practical session (look for the **Data.zip** file on Canvas).

First, using any text editor, just have a look at the content of the ***"gene_sequence.fasta"*** file.

Q. How many sequences can you see in the file? (hint: look at the number of headers)

Q. Is this DNA, RNA or protein? (this is a tricky question !)

Q. Can you read this file to Python in one go, or do you have to read it line by line? (hint: this depends on the size of the file)

Using this FASTA file, we will do the following tasks:

1. Read the data
2. Discard the header
3. Join all the lines together (so the sequence is no longer split into separate lines)

Make a new project for this practical session, add a new Python file in it, give it a meaningful name (e.g. ***fasta_read.py / ipynb***). Start the file following the good programming practices (i.e. add information about the author, date and a short description of the file).

Now, let's open the FASTA file. The file may be located in your project folder or anywhere else in your computer, provided you give an absolute path when open it:

```
fasta = open(r"Your File Location\gene_sequence.fasta", "r")
```

Because the file is small, we will read all the file at once into a list of lines, including the header:

```
lines = fasta.readlines()
print(lines)
```

Make an empty list for the sequence lines w/o header:

```
sequence_list = []
```

Now, loop over the `lines` list, and copy lines to the `sequence_list` except for the line starting with **>** (the header):

```
for line in lines:
     if not line.startswith(">"):
          sequence_list.append(line)
```

Finally, convert `sequence_list` to one sequence, and print the concatenated sequence:

```
sequence = "".join(sequence_list)
print(sequence)
```

Don’t forget to close the FASTA file !

```
fasta.close()
```

Save and execute the program.

Q. Something doesn’t look right when you look at the results of the program. Can you see what is wrong?

Hint: Is the sequence still split into separate lines?

Q. Can you solve the problem?

Q. What happens if you do not close file after running the script?

Exercise 5.1: *Concatenate all FASTA files in a folder into a single file (use the FASTA files from the fasta_direcotry from **Data.zip** file) [Hint: you may use **os** module and function **listdir()** to get the list of files in the folder]*

Task 2: The GenBank Format

GenBank (Genetic Sequence Data Bank) is one of the largest bioinformatics repositories for genetic data.

On the next page, you can see a file retrieved from the GenBank. It’s describing a gene for *PCCX1*: like one of the FASTA files that you already processed. However, the GenBank file provides much more information about the *PCCX1* gene than the FASTA file!

This GenBank file is provided to you for this practical, it’s called ***“pccx1.gb”***. It might seem long, but this has already been truncated! Importantly, the information is organised into clearly identifiable sections with specific headers. This allows us to access the different sections of a GeneBank file programmatically.

```
LOCUS       AB031069                2487 bp    mRNA    linear   PRI 27-MAY-2000

DEFINITION  Homo sapiens PCCX1 mRNA for protein containing CXXC domain 1,
            complete cds.
ACCESSION   AB031069
VERSION     AB031069.1  GI:8100074
KEYWORDS    .
SOURCE      Homo sapiens (human)
  ORGANISM  Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini;
            Catarrhini; Hominidae; Homo.
REFERENCE   1
  AUTHORS   Fujino,T., Hasegawa,M., Shibata,S., Kishimoto,T., Imai,S. and
            Takano,T.
  TITLE     PCCX1, a novel DNA-binding protein with PHD finger and CXXC domain,
            is regulated by proteolysis
  JOURNAL   Biochem. Biophys. Res. Commun. 271 (2), 305-310 (2000)
   PUBMED   10799292
REFERENCE   2  (bases 1 to 2487)
  AUTHORS   Fujino,T., Hasegawa,M., Shibata,S., Kishimoto,T., Imai,S. and
            Takano,T.
  TITLE     Direct Submission
  JOURNAL   Submitted (15-AUG-1999) Tadahiro Fujino, Keio University School of
            Medicine, Department of Microbiology; Shinanomachi 35, Shinjuku-ku,
            Tokyo 160-8582, Japan (E-mail:fujino@microb.med.keio.ac.jp,
            Tel:+81-3-3353-1211(ex.62692), Fax:+81-3-5360-1508)
FEATURES             Location/Qualifiers
     source          1..2487
                     /organism="Homo sapiens"
                     /mol_type="mRNA"
                     /db_xref="taxon:9606"
                     /sex="male"
                     /cell_line="HuS-L12"
                     /cell_type="lung fibroblast"
                     /dev_stage="embryo"
     gene            1..2487
                     /gene="PCCX1"
     CDS             229..2199
                     /gene="PCCX1"
                     /note="a nuclear protein carrying a PHD finger and a CXXC
                     domain"
                     /codon_start=1
                     /product="protein containing CXXC domain 1"
                     /protein_id="BAA96307.1"
                     /db_xref="GI:8100075"
                     /translation="MEGDGSDPEPPDAGEDSKSENGENAPIYCICRKPDINCFMIGCD
                     NCNEWFHGDCIRITEKMAKAIREWYCRECREKDPKLEIRYRHKKSRERDGNERDSSEP
                     RDEGGGRKRPVPDPDLQRRAGSGTGVGAMLARGSASPHKSSPQPLVATPSQHHQQQQQ
                     QIKRSARMCGECEACRRTEDCGHCDFCRDMKKFGGPNKIRQKCRLRQCQLRARESYKY
                     FPSSLSPVTPSESLPRPRRPLPTQQQPQPSQKLGRIREDEGAVASSTVKEPPEATATP
                     EPLSDEDLPLDPDLYQDFCAGAFDDHGLPWMSDTEESPFLDPALRKRAVKVKHVKRRE
                     KKSEKKKEERYKRHRQKQKHKDKWKHPERADAKDPASLPQCLGPGCVRPAQPSSKYCS
                     DDCGMKLAANRIYEILPQRIQQWQQSPCIAEEHGKKLLERIRREQQSARTRLQEMERR
                     FHELEAIILRAKQQAVREDEESNEGDSDDTDLQIFCVSCGHPINPRVALRHMERCYAK
                     YESQTSFGSMYPTRIEGATRLFCDVYNPQSKTYCKRLQVLCPEHSRDPKVPADEVCGC
                     PLVRDVFELTGDFCRLPKRQCNRHYCWEKLRRAEVDLERVRVWYKLDELFEQERNVRT
                     AMTNRAGLLALMLHQTIQHDPLTTDLRSSADR"
ORIGIN
        1 agatggcggc gctgaggggt cttgggggct ctaggccggc cacctactgg tttgcagcgg
       61 agacgacgca tggggcctgc gcaataggag tacgctgcct gggaggcgtg actagaagcg
      121 gaagtagttg tgggcgcctt tgcaaccgcc tgggacgccg ccgagtggtc tgtgcaggtt
     … truncated …
     2341 agggactgtc cccgtcgaca tgttcagtgc ctggtggggc tgcggagtcc actcatcctt
     2401 gcctcctctc cctgggtttt gttaataaaa ttttgaagaa accaaaaaaa aaaaaaaaaa
     2461 aaaaaaaaaa aaaaaaaaaa aaaaaaa
//
```

In this practical we will extract the *ORIGIN sequence* from the GenBank similar to what we did with FASTA file earlier.

As usual, add a new Python file to your Project folder, don't forget about the good style of coding (meaningful file name, the author and a brief file description at the top), then start the code.

Use **"with open()"** instead of **"open()"** and **"close()"** to open the file:

```
with open("Your File Location\pccx1.gb", "r") as gbank:
```

Initialise an empty list to keep our sequence lines before joining (note the indentation!):

```
    sequence_list = []
```

Read line by line, do not use lines content until you reach at line containing the keyword 'ORIGIN' :

```
    for line in gbank:
```

When the file ***pointer*** has moved to the line containing the keyword 'ORIGIN', we will start a new loop, which will continue moving the ***pointer*** within the ***file handler*** after the ORIGIN line:

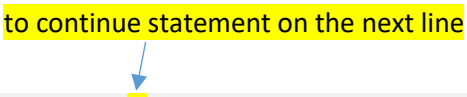
```
        if 'ORIGIN' in line:
            for seqline in gbank:
```

Note that behaviour of `for` loop when it deals with ***file handlers*** is different from what you would expect, if `gbank` was a list or other familiar to you collection. If `gbank` was a *list*, then the second `for` loop would iterate over the whole list again. However, because `gbank` is a file handler, the new `for` loop just continues to move the pointer from the position reached by the previous loop.

Now, let's split the lines containing sequence by white space

```
                seqline_split = seqline.split()
```

Then, we only join the elements, which are not numbers:




```
                sequence_line = ''.join(i \
                    for i in seqline_split if not i.isdigit())
                sequence_list.append(sequence_line)
```

At the end, convert the list of sequences to one string:

```
sequence = ''.join(sequence_list)
print(sequence)
```

Save and run the program.

Q. Do you see anything wrong at the end of the sequence? If yes, can you get rid of it?

Exercise 5.2. *Extract protein sequence from the GenBank file* ***"pccx1.gb"****. and convert them back to a DNA sequence that could encode this protein. You may use the first match from the amino acid codes dictionary given to you.*

Compare the result with the DNA sequence from the GeneBank file. Why the sequences are so different?

Possible answers:

- *The amino acid genetic code is redundant*
- *The GeneBank sequence includes Untranslated Regions (UTRs): this can be seen from the CDS (Coding Sequence) coordinates*

Task 3: The Protein Data Bank (PDB) format (optional)

The Protein Data Bank (PDB) is a database for the three-dimensional structures of large biological molecules, such as proteins and nucleic acids. A standard PDB file format includes information about all atoms in the molecule with their 3D coordinates. Also, it includes detailed metadata ("data that describe other data"), such as the authors, protein superfamily, quaternary structures, chemical moieties that may be present, technical details etc. (see more details in http://www.biostat.jhsph.edu/~iruczins/teaching/260.655/links/pdbformat.pdf,which is also included in the **Data.zip** file provided on Canvas for today's practical sessions).

A typical PDB file looks like this:

```
HEADER    EXTRACELLULAR MATRIX                      22-JAN-98   1A3I
TITLE     X-RAY CRYSTALLOGRAPHIC DETERMINATION OF A COLLAGEN-LIKE
TITLE    2 PEPTIDE WITH THE REPEATING SEQUENCE (PRO-PRO-GLY)
...
EXPDTA    X-RAY DIFFRACTION
AUTHOR    R.Z.KRAMER,L.VITAGLIANO,J.BELLA,R.BERISIO,L.MAZZARELLA,
AUTHOR   2 B.BRODSKY,A.ZAGARI,H.M.BERMAN
...
REMARK 350 BIOMOLECULE: 1
REMARK 350 APPLY THE FOLLOWING TO CHAINS: A, B, C
REMARK 350   BIOMT1   1  1.000000  0.000000  0.000000        0.00000
REMARK 350   BIOMT2   1  0.000000  1.000000  0.000000        0.00000
...
SEQRES   1 A    9  PRO PRO GLY PRO PRO GLY PRO PRO GLY
SEQRES   1 B    6  PRO PRO GLY PRO PRO GLY
SEQRES   1 C    6  PRO PRO GLY PRO PRO GLY
...
ATOM      1  N   PRO A   1       8.316  21.206  21.530  1.00 17.44           N
ATOM      2  CA  PRO A   1       7.608  20.729  20.336  1.00 17.44           C
ATOM      3  C   PRO A   1       8.487  20.707  19.092  1.00 17.44           C
ATOM      4  O   PRO A   1       9.466  21.457  19.005  1.00 17.44           O
ATOM      5  CB  PRO A   1       6.460  21.723  20.211  1.00 22.26           C
...
HETATM  130  C   ACY   401       3.682  22.541  11.236  1.00 21.19           C
HETATM  131  O   ACY   401       2.807  23.097  10.553  1.00 21.19           O
HETATM  132  OXT ACY   401       4.306  23.101  12.291  1.00 21.19           O
…
```

You have been given a PDB file (**"1FAT.pdb"**). This is a protein from Phaseolus vulgaris (in simple language green bean :)

Using this PDB file, we will calculate the centroid of a structure: the average position of all atoms. Strictly speaking, this should be biased by the weight of each atom, but we will ignore for simplicity (and in practice it does not make much of a difference).

As usual, start with adding a new Python file to your project, etc. Then, open the PDB file:

```
pdb = open("1fat.pdb", 'r') # add path to the file, if needed
```

Initialise 3D coordinate positions and a counter for the number of atoms:

```
xsum = ysum = zsum = 0
natoms = 0
```

Same way as with the FASTA and GeneBank data, we will iterate through lines to search for a keyword. However, in this case, the keyword is 'ATOM':

```
for line in pdb:
     if line.startswith("ATOM"):
```

We will count the total number of atoms along with the sum of their 3D coordinates (x, y, z). Note using the addition assignment "+=". The positions of the coordinates in each line are hardcoded here because they are fixed in a PDB file:

```
          natoms += 1

          xsum += float(line[30:38])
          ysum += float(line[38:46])
          zsum += float(line[46:54])

pdb.close()
```

Why function float() is needed?

Now we can calculate the average position of the atoms (which is the centroid):

```
xavg = xsum / natoms
yavg = ysum / natoms
zavg = zsum / natoms
```

Finally, let's print the centroid with the number of atoms

```
print("Number of atoms:", natoms, "\n", \
      "The centroid coordinates:", xavg, yavg, zavg)
```

Exercise 5.3. *Read the given protein databank file **1HST.pdb** and collect the required information into your preferred Python data structure.*

The information that you need to collect includes:

PDB ID
TITLE
ATOM: with element name, residue, chain identifier and 3D coordinates,
(of course, you need to collect information for each atom :)

Now from the populated data structure, count the atoms, chains, residues and elements.

Well done! You have finished Practical 5.