4-Translation

April 21, 2023

1 How to transcribe and translate a DNA sequence

First, we need to import the required modules:

[1]:
```
from Bio import SeqIO
from Bio.SeqIO.FastaIO import as_fasta
```

Then, we can open the file using `with open(<filename>)` and parse its content with the `SeqIO.read()` function:

[2]:
```
with open("inputs/tutorial4/ftsZ.fna", "r") as handle: #Open the file (read␣
↪only mode, r)
    record = SeqIO.read(handle, "fasta") #Read the file, which is in fasta␣
↪format, with the read function from SeqIO
```

OBS! `SeqIO.read()` only works on single records. If we want to read a file with **several sequences**, we use the function `SeqIO.parse()` instead.

When we read a `fasta` file in Biopython, we get an object of class `SeqRecord`, that we stored in the `record` variable, and what you can see after printing are the various attributes that this class has:

[3]:
```
print(record)
```

```
ID: NC_000913.3:105305-106456
Name: NC_000913.3:105305-106456
Description: NC_000913.3:105305-106456 ftsZ [organism=Escherichia coli str. K-12
substr. MG1655] [GeneID=944786] [chromosome=]
Number of features: 0
Seq('ATGTTTGAACCAATGGAACTTACCAATGACGCGGTGATTAAAGTCATCGGCGTC…TAA')
```

We can look more into attributes and methods by using `dir()`:

[4]:
```
dir(record)
```

[4]:
```
['__add__',
 '__bool__',
 '__bytes__',
 '__class__',
 '__contains__',
```

```
'__delattr__',
'__dict__',
'__dir__',
'__doc__',
'__eq__',
'__format__',
'__ge__',
'__getattribute__',
'__getitem__',
'__getstate__',
'__gt__',
'__hash__',
'__init__',
'__init_subclass__',
'__iter__',
'__le__',
'__len__',
'__lt__',
'__module__',
'__ne__',
'__new__',
'__radd__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__setattr__',
'__sizeof__',
'__str__',
'__subclasshook__',
'__weakref__',
'_per_letter_annotations',
'_seq',
'_set_per_letter_annotations',
'_set_seq',
'annotations',
'count',
'dbxrefs',
'description',
'features',
'format',
'id',
'islower',
'isupper',
'letter_annotations',
'lower',
'name',
'reverse_complement',
```

```
 'seq',
 'translate',
 'upper']
```

To check if the parsing worked, let's take a look at one attribute in particular: the `seq` attribute, where the gene sequence is stored:

[5]:
```
print(record.seq)
```

```
ATGTTTGAACCAATGGAACTTACCAATGACGCGGTGATTAAAGTCATCGGCGTCGGCGGCGGCGGCGGTAATGCTGTTGA
ACACATGGTGCGCGAGCGCATTGAAGGTGTTGAATTCTTCGCGGTAAATACCGATGCACAAGCGCTGCGTAAAACAGCGG
TTGGACAGACGATTCAAATCGGTAGCGGTATCACCAAAGGACTGGGCGCTGGCGCTAATCCAGAAGTTGGCCGCAATGCG
GCTGATGAGGATCGCGATGCATTGCGTGCGGCGCTGGAAGGTGCAGACATGGTCTTTATTGCTGCGGGTATGGGTGGTGG
TACCGGTACAGGTGCAGCACCAGTCGTCGCTGAAGTGGCAAAAGATTTGGGTATCCTGACCGTTGCTGTCGTCACTAAGC
CTTTCAACTTTGAAGGCAAGAAGCGTATGGCATTCGCGGAGCAGGGGATCACTGAACTGTCCAAGCATGTGGACTCTCTG
ATCACTATCCCGAACGACAAACTGCTGAAAGTTCTGGGCCGCGGTATCTCCCTGCTGGATGCGTTTGGCGCAGCGAACGA
TGTACTGAAAGGCGCTGTGCAAGGTATCGCTGAACTGATTACTCGTCCGGGTTTGATGAACGTGGACTTTGCAGACGTAC
GCACCGTAATGTCTGAGATGGGCTACGCAATGATGGGTTCTGGCGTGGCGAGCGGTGAAGACCGTGCGGAAGAAGCTGCT
GAAATGGCTATCTCTTCTCCGCTGCTGGAAGATATCGACCTGTCTGGCGCGCGCGGCGTGCTGGTTAACATCACGGCGGG
CTTCGACCTGCGTCTGGATGAGTTCGAAACGGTAGGTAACACCATCCGTGCATTTGCTTCCGACAACGCGACTGTGGTTA
TCGGTACTTCTCTTGACCCGGATATGAATGACGAGCTGCGCGTAACCGTTGTTGCGACAGGTATCGGCATGGACAAACGT
CCTGAAATCACTCTGGTGACCAATAAGCAGGTTCAGCAGCCAGTGATGGATCGCTACCAGCAGCATGGGATGGCTCCGCT
GACCCAGGAGCAGAAGCCGGTTGCTAAAGTCGTGAATGACAATGCGCCGCAAACTGCGAAAGAGCCGGATTATCTGGATA
TCCCAGCATTCCTGCGTAAGCAAGCTGATTAA
```

It worked! But we are looking only at a part of the information that we have. The `seq` attribute is not the only useful one. In this tutorial, we want to create a new `fasta` file. In this file type, we have a `description` as a header and then the sequence, this way:

[6]:
```
print(f">{record.description}")
print(record.seq)
```

```
>NC_000913.3:105305-106456 ftsZ [organism=Escherichia coli str. K-12 substr.
MG1655] [GeneID=944786] [chromosome=]
ATGTTTGAACCAATGGAACTTACCAATGACGCGGTGATTAAAGTCATCGGCGTCGGCGGCGGCGGCGGTAATGCTGTTGA
ACACATGGTGCGCGAGCGCATTGAAGGTGTTGAATTCTTCGCGGTAAATACCGATGCACAAGCGCTGCGTAAAACAGCGG
TTGGACAGACGATTCAAATCGGTAGCGGTATCACCAAAGGACTGGGCGCTGGCGCTAATCCAGAAGTTGGCCGCAATGCG
GCTGATGAGGATCGCGATGCATTGCGTGCGGCGCTGGAAGGTGCAGACATGGTCTTTATTGCTGCGGGTATGGGTGGTGG
TACCGGTACAGGTGCAGCACCAGTCGTCGCTGAAGTGGCAAAAGATTTGGGTATCCTGACCGTTGCTGTCGTCACTAAGC
CTTTCAACTTTGAAGGCAAGAAGCGTATGGCATTCGCGGAGCAGGGGATCACTGAACTGTCCAAGCATGTGGACTCTCTG
ATCACTATCCCGAACGACAAACTGCTGAAAGTTCTGGGCCGCGGTATCTCCCTGCTGGATGCGTTTGGCGCAGCGAACGA
TGTACTGAAAGGCGCTGTGCAAGGTATCGCTGAACTGATTACTCGTCCGGGTTTGATGAACGTGGACTTTGCAGACGTAC
GCACCGTAATGTCTGAGATGGGCTACGCAATGATGGGTTCTGGCGTGGCGAGCGGTGAAGACCGTGCGGAAGAAGCTGCT
GAAATGGCTATCTCTTCTCCGCTGCTGGAAGATATCGACCTGTCTGGCGCGCGCGGCGTGCTGGTTAACATCACGGCGGG
CTTCGACCTGCGTCTGGATGAGTTCGAAACGGTAGGTAACACCATCCGTGCATTTGCTTCCGACAACGCGACTGTGGTTA
TCGGTACTTCTCTTGACCCGGATATGAATGACGAGCTGCGCGTAACCGTTGTTGCGACAGGTATCGGCATGGACAAACGT
CCTGAAATCACTCTGGTGACCAATAAGCAGGTTCAGCAGCCAGTGATGGATCGCTACCAGCAGCATGGGATGGCTCCGCT
GACCCAGGAGCAGAAGCCGGTTGCTAAAGTCGTGAATGACAATGCGCCGCAAACTGCGAAAGAGCCGGATTATCTGGATA
TCCCAGCATTCCTGCGTAAGCAAGCTGATTAA
```

In the example above, we printed the `description` attribute of the `record` object as a header, and then the `seq` attribute. The `seq` attribute consists of an object of class `Seq` that includes a string representing the gene sequence. There are several interesting methods that you can apply on a `Seq` object; for example, you can search for specific sub-sequences inside your sequence. Let's search for the start codon (`ATG`):

[7]:
```
record.seq.find("ATG")
```

[7]:
```
0
```

The cell above returned positon 0, since, of course, the start codon should be at the start of the sequence. We can also count the number of times a sub-sequence is present in a sequence. For example, let's do that for the stop codon `TAA`:

[8]:
```
record.seq.count("TAA")
```

[8]:
```
14
```

It returned 14! How is that possible? Do we have 14 stop codons? Of course not! The sequence is searched in the sequence string without taking into account the codon frames. In this case, because we have a gene, codons start at every third possition, but if we were looking at a full genome, for example, we might have genes overlapping over different codon frames, so this can be useful. We can look into all the methods that are available for the `Seq` class by using `dir()`:

[9]:
```
dir(record.seq)
```

[9]:
```
['__abstractmethods__',
 '__add__',
 '__array_ufunc__',
 '__bytes__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
```

```
'__len__',
'__lt__',
'__module__',
'__mul__',
'__ne__',
'__new__',
'__radd__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__rmul__',
'__setattr__',
'__sizeof__',
'__slots__',
'__str__',
'__subclasshook__',
'__weakref__',
'_abc_impl',
'_data',
'back_transcribe',
'complement',
'complement_rna',
'count',
'count_overlap',
'defined',
'defined_ranges',
'endswith',
'find',
'index',
'islower',
'isupper',
'join',
'lower',
'lstrip',
'replace',
'reverse_complement',
'reverse_complement_rna',
'rfind',
'rindex',
'rsplit',
'rstrip',
'split',
'startswith',
'strip',
'transcribe',
'translate',
'ungap',
```

```
 'upper']
```

Here you can see many other useful attributes and, more commonly, methods. Some of these methods are shared with strings, such as **split**, **strip** or **startwith**. However, there are others that are unique to **Seq** objects. For this tutorial, we will try some of them to transform the DNA sequence into protein. First, the DNA sequence can be converted to RNA by using the **transcribe()** function:

[10]:
```
RNA_seq = record.seq.transcribe() # Transcribe (DNA > RNA) the sequence
print(RNA_seq) # Print the RNA sequence
```

```
AUGUUUGAACCAAUGGAACUUACCAAUGACGCGGUGAUUAAAGUCAUCGGCGUCGGCGGCGGCGGCGGUAAUGCUGUUGA
ACACAUGGUGCGCGAGCGCAUUGAAGGUGUUGAAUUCUUCGCGGUAAAUACCGAUGCACAAGCGCUGCGUAAAACAGCGG
UUGGACAGACGAUUCAAAUCGGUAGCGGUAUCACCAAAGGACUGGGCGCUGGCGCUAAUCCAGAAGUUGGCCGCAAUGCG
GCUGAUGAGGAUCGCGAUGCAUUGCGUGCGGCGCUGGAAGGUGCAGACAUGGUCUUUAUUGCUGCGGGUAUGGGUGGUGG
UACCGGUACAGGUGCAGCACCAGUCGUCGCUGAAGUGGCAAAAGAUUUGGGUAUCCUGACCGUUGCUGUCGUCACUAAGC
CUUUCAACUUUGAAGGCAAGAAGCGUAUGGCAUUCGCGGAGCAGGGGAUCACUGAACUGUCCAAGCAUGUGGACUCUCUG
AUCACUAUCCCGAACGACAAACUGCUGAAAGUUCUGGGCCGCGGUAUCUCCCUGCUGGAUGCGUUUGGCGCAGCGAACGA
UGUACUGAAAGGCGCUGUGCAAGGUAUCGCUGAACUGAUUACUCGUCCGGGUUUGAUGAACGUGGACUUUGCAGACGUAC
GCACCGUAAUGUCUGAGAUGGGCUACGCAAUGAUGGGUUCUGGCGUGGCGAGCGGUGAAGACCGUGCGGAAGAAGCUGCU
GAAAUGGCUAUCUCUUCUCCGCUGCUGGAAGAUAUCGACCUGUCUGGCGCGCGCGGCGUGCUGGUUAACAUCACGGCGGG
CUUCGACCUGCGUCUGGAUGAGUUCGAAACGGUAGGUAACACCAUCCGUGCAUUUGCUUCCGACAACGCGACUGUGGUUA
UCGGUACUUCUCUUGACCCGGAUAUGAAUGACGAGCUGCGCGUAACCGUUGUUGCGACAGGUAUCGGCAUGGACAAACGU
CCUGAAAUCACUCUGGUGACCAAUAAGCAGGUUCAGCAGCCAGUGAUGGAUCGCUACCAGCAGCAUGGGAUGGCUCCGCU
GACCCAGGAGCAGAAGCCGGUUGCUAAAGUCGUGAAUGACAAUGCGCCGCAAACUGCGAAAGAGCCGGAUUAUCUGGAUA
UCCCAGCAUUCCUGCGUAAGCAAGCUGAUUAA
```

Then, we can easily convert the DNA or RNA sequence into protein by using the **translate()** function:

[11]:
```
protein_seq = RNA_seq.translate() # Overwrite the DNA sequence with the␣
 ↪translation (DNA > RNA > protein)
print(protein_seq) # Print the protein sequence
```

```
MFEPMELTNDAVIKVIGVGGGGGNAVEHMVRERIEGVEFFAVNTDAQALRKTAVGQTIQIGSGITKGLGAGANPEVGRNA
ADEDRDALRAALEGADMVFIAAGMGGGTGTGAAPVVAEVAKDLGILTVAVVTKPFNFEGKKRMAFAEQGITELSKHVDSL
ITIPNDKLLKVLGRGISLLDAFGAANDVLKGAVQGIAELITRPGLMNVDFADVRTVMSEMGYAMMGSGVASGEDRAEEAA
EMAISSPLLEDIDLSGARGVLVNITAGFDLRLDEFETVGNTIRAFASDNATVVIGTSLDPDMNDELRVTVVATGIGMDKR
PEITLVTNKQVQQPVMDRYQQHGMAPLTQEQKPVAKVVNDNAPQTAKEPDYLDIPAFLRKQAD*
```

The asterisk at the end represents the **stop codon**. If we don't want the asterisk, we have to specify that the translation should stop at the stop codon:

[12]:
```
protein_seq = RNA_seq.translate(to_stop = True) # Translate until the stop␣
 ↪codon (not included)
print(protein_seq) # Print the protein sequence again
```

```
MFEPMELTNDAVIKVIGVGGGGGNAVEHMVRERIEGVEFFAVNTDAQALRKTAVGQTIQIGSGITKGLGAGANPEVGRNA
ADEDRDALRAALEGADMVFIAAGMGGGTGTGAAPVVAEVAKDLGILTVAVVTKPFNFEGKKRMAFAEQGITELSKHVDSL
ITIPNDKLLKVLGRGISLLDAFGAANDVLKGAVQGIAELITRPGLMNVDFADVRTVMSEMGYAMMGSGVASGEDRAEEAA
```

```
EMAISSPLLEDIDLSGARGVLVNITAGFDLRLDEFETVGNTIRAFASDNATVVIGTSLDPDMNDELRVTVVATGIGMDKR
PEITLVTNKQVQQPVMDRYQQHGMAPLTQEQKPVAKVVNDNAPQTAKEPDYLDIPAFLRKQAD
```

OBS! The translate function can be used directly on DNA sequences, but here we used the `transcribe()` function first to test more methods.

Now, we can save the output to a fasta file by using the **`as_fasta()`** function that we imported in the beginning:

[16]:
```
import os
out_file = "inputs/tutorial5/ftsZ.faa" # Name of output file
protein_record = record # Create a new record that is a copy of the original
protein_record.seq = protein_seq # Replace the DNA sequence with the protein␣
↪sequence
with open(out_file, "w") as faa: # Open new file in write mode
    faa.write(as_fasta(protein_record)) # Write amino acid sequence in fasta␣
↪format with as_fasta
```

There are alternatives to using this function; for example, you can use the **`SeqIO.write()`** function and specify the output file type:

[17]:
```
out_file = "inputs/tutorial5/ftsZ.faa"
protein_record = record
protein_record.seq = protein_seq
with open(out_file, "w") as faa:
    SeqIO.write(protein_record, faa, "fasta") # Write amino acid sequence to␣
↪fasta with SeqIO.write
```

Last, we can check the content of the newly-created file:

[18]:
```
with open(out_file, "r") as faa:
    for line in faa: #Here we loop normally through the file to check its␣
↪contents
        print(line, end = "") #Fasta files already have a line break at the end␣
↪of each line
```

```
>NC_000913.3:105305-106456 ftsZ [organism=Escherichia coli str. K-12 substr.
MG1655] [GeneID=944786] [chromosome=]
MFEPMELTNDAVIKVIGVGGGGGNAVEHMVRERIEGVEFFAVNTDAQALRKTAVGQTIQI
GSGITKGLGAGANPEVGRNAADEDRDALRAALEGADMVFIAAGMGGGTGTGAAPVVAEVA
KDLGILTVAVVTKPFNFEGKKRMAFAEQGITELSKHVDSLITIPNDKLLKVLGRGISLLD
AFGAANDVLKGAVQGIAELITRPGLMNVDFADVRTVMSEMGYAMMGSGVASGEDRAEEAA
EMAISSPLLEDIDLSGARGVLVNITAGFDLRLDEFETVGNTIRAFASDNATVVIGTSLDP
DMNDELRVTVVATGIGMDKRPEITLVTNKQVQQPVMDRYQQHGMAPLTQEQKPVAKVVND
NAPQTAKEPDYLDIPAFLRKQAD
```

We managed to translate the sequence and save it to a new file! Perhaps you want to try to do the same now.

1.1 So, what do I do now?

Download a file containing the DNA sequence of a gene from NCBI and try to translate the sequence yourself. You can also save the translated version to a new file if you want, and you can play around with other attributes of the `SeqRecord` class; for example, by modifying the description. You can use the cell below.

[ ]:
```
# Write your code here
```