Class 24: Using regular expressions to analyze data

April 18, 2019

In this class, we will continue discuss a few more real-world scenarios of how we can use regular expressions to analyze data. We will also encounter some other useful methods and techniques, such as reading a file directly from the internet.

Reading a file from the internet

To read a file from the internet (i.e., from a URL), we can use the `urlopen()` function that is available in the package `urllib.request`. One subtle difficulty with this function is that it returns encoded strings (also called "byte objects") rather than raw strings. For example, the following code downloads the poem "The Road Not Taken" from the class website and prints it out. You can see how the output is encapsulated in quotes with a 'b' in front (for "byte object"), and also that the newline character is explicitly written out as '\n'.

```
In [1]: from urllib.request import urlopen

        file_URL = "http://wilkelab.org/classes/SDS348/data_sets/road_not_taken.txt"

        with urlopen(file_URL) as infile:
            for line_encoded in infile:
                print (line_encoded)
```

```
b'        THE ROAD NOT TAKEN\n'
b'           Robert Frost\n'
b'\n'
b'Two roads diverged in a yellow wood,\n'
b'And sorry I could not travel both\n'
b'And be one traveler, long I stood\n'
b'And looked down one as far as I could\n'
b'To where it bent in the undergrowth;\n'
b'Then took the other, as just as fair,\n'
b'And having perhaps the better claim,\n'
b'Because it was grassy and wanted wear;\n'
b'Though as for that the passing there\n'
b'Had worn them really about the same,\n'
b'And both that morning equally lay\n'
b'In leaves no step had trodden black.\n'
b'Oh, I kept the first for another day!\n'
b'Yet knowing how way leads on to way,\n'
b'I doubted if I should ever come back.\n'
b'I shall be telling this with a sigh\n'
b'Somewhere ages and ages hence:\n'
b'Two roads diverged in a wood, and I-\n'
b'I took the one less traveled by,\n'
b'And that has made all the difference.\n'
```

To convert the encoded byte objects into raw strings that you can work with, you can call the function `decode()` on the byte object:

```
In [2]: with urlopen(file_URL) as infile:
            for line_encoded in infile:
                line = line_encoded.decode()
                print(line, end='') # need to set end='' because each line comes with a
        newline character already
```

```
        THE ROAD NOT TAKEN
           Robert Frost

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;
Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same,
And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.
I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I-
I took the one less traveled by,
And that has made all the difference.
```

Problems

Problem 1:

Write a function that can tell whether a string is DNA, RNA, or neither. Strings that contain some DNA/RNA and some other stuff, such as "This is a gene: AGTACCGTAG", should be flagged as neither.

In [3]:
```
import re

def what_is_it(string):
    if re.search(r"^[AGCTagct]+$", string):
        print("DNA:", string)
    elif re.search(r"^[AGCUagcu]+$", string):
        print("RNA:", string)
    else:
        print("neither:", string)

what_is_it("AGCTCGAGCTA")  # DNA
what_is_it("AGCUCGAGCUA")  # RNA
what_is_it("AGCTCGAGCUA")  # neither, uses Ts and Us at the same time
what_is_it("This is a gene: AGTACCGTAG")  # neither, contains other characters
what_is_it("ucgcuucgacacgu")  # RNA
what_is_it("atgtctacact")  # DNA
```

```
DNA: AGCTCGAGCTA
RNA: AGCUCGAGCUA
neither: AGCTCGAGCUA
neither: This is a gene: AGTACCGTAG
RNA: ucgcuucgacacgu
DNA: atgtctacact
```

Problem 2:

Write code that counts the number of lines in Robert Frost's "The Road Not Taken" that contain the article "the". Also keep track of the words appearing immediately after the "the", by recording them in a list.

Hint 1: Each line contains the article "the" either 0 or 1 times. You don't have to worry about multiple occurrences of "the" in the same line. However, make sure you don't count "them", "there", or similar words that contain "the" as a substring.

Hint 2: Make sure that you capture both lower-case, upper-case, and all-caps spellings of "the". See if you can do this with a single regular expression.

In [4]:
```
# We will use urlopen to read the file directly from the class website
from urllib.request import urlopen

file_URL = "http://wilkelab.org/classes/SDS348/data_sets/road_not_taken.txt"

count = 0
wordlist = []
with urlopen(file_URL) as infile:
    for line_encoded in infile:
        # urllib returns encoded strings (byte objects), and
        # we need to use the `decode()` function to turn them
        # into strings we can work with
        line = line_encoded.decode()

        match = re.search(r"\s+[tT][hH][eE]\s+(\w+)", line)
        if match:
            count += 1
            wordlist.append(match.group(1))

print('Number of lines containing the article "the":', count)
print('Words appearing after "the":', wordlist)
```

```
Number of lines containing the article "the": 9
Words appearing after "the": ['ROAD', 'undergrowth', 'other', 'better', 'passin
g', 'same', 'first', 'one', 'difference']
```

If this was easy

Problem 3:

For this problem, we will parse the strain names of influenza A virus. First, we will perform an Entrez search to retrieve influenza A viruses from 2014 for which we have a complete hemagglutinin coding sequence. (We limit the search to 50 results to keep things manageable.)

In [5]:
```
from Bio import Entrez, SeqIO

Entrez.email = 'wilke@austin.utexas.edu'

# let's do a search for influenza H1N1 viruses from 2014
handle = Entrez.esearch(db="nucleotide",  # database to search
                        term="influenza a virus 2014 h1n1 hemagglutinin complet
e cds",  # search term
                        retmax=50  # limit the search to the first 50 results
                        )
record = Entrez.read(handle)
handle.close()

# download the results and turn into a list of records
gi_list = record["IdList"] # list of genbank identifiers found
handle = Entrez.efetch(db="nucleotide", id=gi_list, rettype="gb", retmode="text
")
records = list(SeqIO.parse(handle, "genbank")) # the list(...) statement ensure
s we will still have
                                               # access to the results after ha
ndle.close()
handle.close()

for record in records:
    print(record.description)
```

```
Influenza A virus (A/swine/Valparaiso/VN1401-559/2014(H1N1)) segment 4 hemagglu
tinin (HA) gene, complete cds
Influenza A virus (A/mallard/Balkhash/6304_HA/2014(H1N1)) segment 4 hemagglutin
in (HA) gene, complete cds
Influenza A virus (A/Cameroon/15v-7696/2015(H1)) segment 4 hemagglutinin (HA) g
ene, complete cds
Influenza A virus (A/Saudi Arabia/147/2014(H1N1)) segment 4 hemagglutinin (HA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/144/2014(H1N1)) segment 4 hemagglutinin (HA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/143/2014(H1N1)) segment 4 hemagglutinin (HA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/138/2014(H1N1)) segment 4 hemagglutinin (HA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/137/2014(H1N1)) segment 4 hemagglutinin (HA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/136/2014(H1N1)) segment 4 hemagglutinin (HA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/134/2014(H1N1)) segment 4 hemagglutinin (HA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/131/2014(H1N1)) segment 4 hemagglutinin (HA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/130/2014(H1N1)) segment 4 hemagglutinin (HA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/129/2014(H1N1)) segment 4 hemagglutinin (HA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/147/2014(H1N1)) segment 6 neuraminidase (NA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/144/2014(H1N1)) segment 6 neuraminidase (NA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/143/2014(H1N1)) segment 6 neuraminidase (NA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/138/2014(H1N1)) segment 6 neuraminidase (NA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/137/2014(H1N1)) segment 6 neuraminidase (NA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/136/2014(H1N1)) segment 6 neuraminidase (NA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/134/2014(H1N1)) segment 6 neuraminidase (NA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/131/2014(H1N1)) segment 6 neuraminidase (NA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/130/2014(H1N1)) segment 6 neuraminidase (NA)
gene, complete cds
Influenza A virus (A/Saudi Arabia/129/2014(H1N1)) segment 6 neuraminidase (NA)
gene, complete cds
Influenza A virus (A/swine/O'Higgins/VN1401-380/2014(H1N1)) segment 4 hemagglut
inin (HA) gene, complete cds
Influenza A virus (A/swine/France/57-140136/2014(H1N1)) segment 4 hemagglutinin
(HA) gene, complete cds
Influenza A virus (A/swine/France/35-140384/2014(H1N1)) segment 4 hemagglutinin
(HA) gene, complete cds
Influenza A virus (A/swine/France/35-140382/2014(H1N1)) segment 4 hemagglutinin
(HA) gene, complete cds
Influenza A virus (A/swine/Brazil/103-2/2014(H1N1)) segment 4 hemagglutinin (HA
) gene, complete cds
Influenza A virus (A/blue-winged_teal/Louisiana/UGAI14-282/2014(H1N1)) segment
4 hemagglutinin (HA) gene, complete cds
Influenza A virus (A/swine/Rengo/VN1401-95/2014(H1N1)) segment 4 hemagglutinin
(HA) gene, complete cds
Influenza A virus (A/swine/Rengo/VN1401-94/2014(H1N1)) segment 4 hemagglutinin
(HA) gene, complete cds
Influenza A virus (A/swine/Rengo/VN1401-83/2014(H1N1)) segment 4 hemagglutinin
```

Now write code that goes over all records in the list and counts (1) how many human strains there are and (2) how many strains were collected in Texas. To do this, you need to know how influenza strains are named. The naming scheme is "A/*host (if not human)*/*region of origin*/*lineage number*/*year*/(*antigen type*)" (see also here (http://blog.h1n1.influenza.bvsalud.org/en/2009/09/20/how-do-we-name-influenza-a/)). The tricky part is that the host field is optional. For example, A/swine/Mexico/10466772/2014(H1N1) is a 2014 H1N1 swine strain from Mexico, but A/Texas/6180/2017(H1N1) is a 2017 H1N1 human strain from Texas. Note that even though we searched for 2014, we are retrieving some strains for 2017.

In [6]:
```
human_count = 0
TX_count = 0
for record in records:
    match = re.search(r"^Influenza A virus \(A/(\w*/)*(\w.*)/[-\w]+/20\d\d\s*\(
H1N1\)\)", record.description)
    if match: # should always be true
        if match.group(1) == None: # for human strains, match group 1 will be e
mpty
            human_count += 1
        if match.group(2) == 'Texas':
            TX_count += 1
    else:
        print("Cannot match:", record.description)

print("Total number of human strains:", human_count)
print("Total number of strains from Texas:", TX_count)
```

```
Cannot match: Influenza A virus (A/Cameroon/15v-7696/2015(H1)) segment 4 hemagg
lutinin (HA) gene, complete cds
Total number of human strains: 21
Total number of strains from Texas: 0
```