Class 20: Working with gene features and genomes

Apr 4, 2019

Features in genbank files

Many important pieces of information in genbank files are stored in features. These features can be queried through Biopython by working with the `features` list of a genbank record object (`record.features`). More information about sequence features is available here. (http://biopython.org/DIST/docs/tutorial/Tutorial.html#htoc38) We will usually iterate over all features in the list with a `for` loop:

In [1]:
```
from Bio import Entrez, SeqIO
Entrez.email = "wilke@austin.utexas.edu" # put your email here

# Download sequence record for genbank id KT220438
# This is HA gene of Influenza A virus, strain A/NewJersey/NHRC_93219/2015(H3N2
)
handle = Entrez.efetch(db="nucleotide", id="KT220438", rettype="gb", retmode="t
ext")
record = SeqIO.read(handle, "genbank")
handle.close()

# Loop over all features in the record to see
# what features there are and what information
# they contain
for feature in record.features:
    print(feature)
```

```
type: source
location: [0:1701](+)
qualifiers:
    Key: collection_date, Value: ['17-Jan-2015']
    Key: country, Value: ['USA: New Jersey']
    Key: db_xref, Value: ['taxon:1682360']
    Key: host, Value: ['Homo sapiens']
    Key: isolation_source, Value: ['nasopharyngeal swab']
    Key: lab_host, Value: ['MDCK']
    Key: mol_type, Value: ['viral cRNA']
    Key: organism, Value: ['Influenza A virus (A/New Jersey/NHRC_93219/2015(H3N
2))']
    Key: segment, Value: ['4']
    Key: serotype, Value: ['H3N2']
    Key: strain, Value: ['A/NewJersey/NHRC_93219/2015']

type: gene
location: [0:1701](+)
qualifiers:
    Key: gene, Value: ['HA']

type: CDS
location: [0:1701](+)
qualifiers:
    Key: codon_start, Value: ['1']
    Key: function, Value: ['receptor binding and fusion protein']
    Key: gene, Value: ['HA']
    Key: product, Value: ['hemagglutinin']
    Key: protein_id, Value: ['AKQ43545.1']
    Key: translation, Value: ['MKTIIALSYILCLVFAQKIPGNDNSTATLCLGHHAVPNGTIVKTITND
RIEVTNATELVQNSSIGEICDSPHQILDGENCTLIDALLGDPQCDGFQNKKWDLFVERSKAYSNCYPYDVPDYASLRSL
VASSGTLEFNNESFNWTGVTQNGTSSACIRRSSSSFFSRLNWLTHLNYTYPALNVTMPNNEQFDKLYIWGVHHPGTDKD
QIFLYAQSSGRITVSTKRSQQAVIPNIGSRPRIRDIPSRISIYWTIVKPGDILLINSTGNLIAPRGYFKIRSGKSSIMR
SDAPIGKCKSECITPNGSIPNDKPFQNVNRITYGACPRYVKHSTLKLATGMRNVPEKQTRGIFGAIAGFIENGWEGMVD
GWYGFRHQNSEGRGQAADLKSTQAAIDQINGKLNRLIGKTNEKFHQIEKEFSEVEGRIQDLEKYVEDTKIDLWSYNAEL
LVALENQHTXDLTDSEMNKLFEKTKKQLRENAEDMGNGCFKIYHKCDNACIGSIRNGTYDHNVYRDEALNNRFQIKGVE
LKSGYKDWILWISXAISCFLLCVALLGFIMWACQKGNIRCNICI']

type: mat_peptide
location: [48:1035](+)
qualifiers:
    Key: gene, Value: ['HA']
    Key: product, Value: ['HA1']

type: mat_peptide
location: [1035:1698](+)
qualifiers:
    Key: gene, Value: ['HA']
    Key: product, Value: ['HA2']
```

The type of each feature is stored in `feature.type`, and its location on the DNA sequence is stored in `feature.location`. The additional feature information is stored in a dictionary called `qualifiers`. We can query this dictionary to retrieve individual elements of a feature. Note that these elements are all stored as lists, and so we generally have to add `[0]` at the end of our query to retrieve the first element in the list.

In [2]:
```
for feature in record.features:
    if feature.type == "CDS":
        print("Coding sequence found at position:", feature.location)
        print("Gene product:", feature.qualifiers["product"][0])
        print("Protein ID:", feature.qualifiers["protein_id"][0])
        print("Protein sequence:", feature.qualifiers["translation"][0])
```

```
Coding sequence found at position: [0:1701](+)
Gene product: hemagglutinin
Protein ID: AKQ43545.1
Protein sequence: MKTIIALSYILCLVFAQKIPGNDNSTATLCLGHHAVPNGTIVKTITNDRIEVTNATELVQN
SSIGEICDSPHQILDGENCTLIDALLGDPQCDGFQNKKWDLFVERSKAYSNCYPYDVPDYASLRSLVASSGTLEFNNES
FNWTGVTQNGTSSACIRRSSSSFFSRLNWLTHLNYTYPALNVTMPNNEQFDKLYIWGVHHPGTDKDQIFLYAQSSGRIT
VSTKRSQQAVIPNIGSRPRIRDIPSRISIYWTIVKPGDILLINSTGNLIAPRGYFKIRSGKSSIMRSDAPIGKCKSECI
TPNGSIPNDKPFQNVNRITYGACPRYVKHSTLKLATGMRNVPEKQTRGIFGAIAGFIENGWEGMVDGWYGFRHQNSEGR
GQAADLKSTQAAIDQINGKLNRLIGKTNEKFHQIEKEFSEVEGRIQDLEKYVEDTKIDLWSYNAELLVALENQHTXDLT
DSEMNKLFEKTKKQLRENAEDMGNGCFKIYHKCDNACIGSIRNGTYDHNVYRDEALNNRFQIKGVELKSGYKDWILWIS
XAISCFLLCVALLGFIMWACQKGNIRCNICI
```

Problems

Problem 1:

When we are working with larger genbank files (e.g. entire genomes), we generally first download them and store them as a file on our local drive. Then we work with those local files. To practice this workflow, write code that downloads the complete genome of bacteriophage T7 and stores it in a file called `T7.gb`. The accession number for that genome is NC_001604. Make sure you can find the file you have created on your harddrive. Also verify that the file contains the same genome you can see on the NCBI website.

In [3]:
```
# Problem 1

# Download T7 genome:
download_handle = Entrez.efetch(db="nucleotide", id="NC_001604", rettype="gb",
retmode="text")
data = download_handle.read()
download_handle.close()

# Store data into file "T7.gb":
out_handle = open("T7.gb", "w")
out_handle.write(data)
out_handle.close()
```

Problem 2:

Now read in the T7 genome, and for each coding sequence (CDS) in that genome print out the name (`locus_tag`) and the product name (`product`).

Hints:

- First you need to parse the genbank file into a record, using `SeqIO.read()`.
- All gene annotations are stored as features, and `record.features` gives you a list of all these features.
- Each feature has a type, accessible through `feature.type`. We are only interested in features of type "CDS".
- The actual annotations we are interested in, such as `locus_tag`, `product`, etc., are stored in the dictionary `feature.qualifiers`.

In [4]:
```
# Problem 2

in_handle = open("T7.gb", "r")
record = SeqIO.read(in_handle, "genbank")
in_handle.close()

for feature in record.features:
    if feature.type == 'CDS':
        # all the qualifiers are returned as lists, so we have to add `[0]` at
the end to just get
        # the first element of each list
        locus_tag = feature.qualifiers['locus_tag'][0]
        product = feature.qualifiers['product'][0]
        print(locus_tag + ": " + product)
```

```
T7p01: hypothetical protein
T7p02: hypothetical protein
T7p04: hypothetical protein
T7p05: hypothetical protein
T7p06: hypothetical protein
T7p03: protein kinase
T7p07: T3/T7-like RNA polymerase
T7p09: hypothetical protein
T7p08: host dGTPase inhibitor
T7p10: ATP-dependent DNA ligase
T7p11: hypothetical protein
T7p12: hypothetical protein
T7p13: hypothetical protein
T7p14: hypothetical protein
T7p15: hypothetical protein
T7p16: inhibitor of host bacterial RNA polymerase
T7p17: single-stranded DNA-binding protein
T7p18: hypothetical protein
T7p19: endonuclease I
T7p20: lysozyme
T7p21: putative NHN endonuclease
T7p22: DNA primase/helicase
T7p23: hypothetical protein
T7p24: helicase
T7p25: hypothetical protein
T7p26: hypothetical protein
T7p27: hypothetical protein
T7p28: hypothetical protein
T7p29: DNA polymerase
T7p30: hypothetical protein
T7p31: hypothetical protein
T7p32: host protein H-NS-interacting protein
T7p33: hypothetical protein
T7p34: host recBCD nuclease inhibitor
T7p35: exonuclease
T7p36: hypothetical protein
T7p37: hypothetical protein
T7p38: hypothetical protein
T7p39: hypothetical protein
T7p40: tail assembly protein
T7p41: hypothetical protein
T7p42: head-tail connector protein
T7p43: capsid assembly protein
T7p44: major capsid protein
T7p45: major capsid protein
T7p46: tail tubular protein A
T7p47: tail tubular protein B
T7p48: internal virion protein A
T7p49: internal virion protein B
T7p50: internal virion protein C
T7p51: internal virion protein D
T7p52: tail fiber protein
T7p53: type II holin
T7p54: DNA packaging protein, small subunit
T7p55: phage lambda Rz-like lysis protein
T7p56: phage lambda Rz1-like protein
T7p57: DNA maturation protein
T7p58: hypothetical protein
T7p59: hypothetical protein
T7p60: hypothetical protein
```

Problem 3:

Calculate the mean gene length (in amino acids) for all CDSs in the T7 genome.

Hint: Use the function `round()` to round your result to 2 decimal digits.

In [5]:
```
in_handle = open("T7.gb", "r")
record = SeqIO.read(in_handle, "genbank")
in_handle.close()

count = 0 # counts the number of genes in the genome
length_sum = 0. # holds the total sum of all lengths of all genes
for feature in record.features:
    if feature.type == 'CDS':
        seq = feature.qualifiers['translation'][0]
        length_sum += len(seq)
        count += 1

print("mean length:", round(length_sum/count, 2), "amino acids")
```

```
mean length: 227.62 amino acids
```

Problem 4:

(a) Download the *E. coli* K12 genome, available here: http://www.ncbi.nlm.nih.gov/nuccore/CP009685 (http://www.ncbi.nlm.nih.gov/nuccore/CP009685), and store it in a file called `Ecoli_K12.gb`.

(b) Calculate the fraction of coding sequences that are marked as "hypothetical protein". (This information is provided in the "product" qualifier of a "CDS" feature.)

(c) Calculate the GC content of the *entire* genome sequence. GC content is the fraction of G and C nucleotides in the sequence.

In [6]:
```
# Problem 4a

# Download E. coli K12 genome:
download_handle = Entrez.efetch(db="nucleotide", id="CP009685", rettype="gb", r
etmode="text")
data = download_handle.read()
download_handle.close()

# Store data into file "Ecoli_K12.gb":
out_handle = open("Ecoli_K12.gb", "w")
out_handle.write(data)
out_handle.close()
```

In [7]:
```
# Problem 4b

# Read in E. coli K12 genome:
in_handle = open("Ecoli_K12.gb", "r")
record = SeqIO.read(in_handle, "genbank")
in_handle.close()

# count all CDSs and the hypothetical ones
CDS_count = 0
hyp_CDS_count = 0
for feature in record.features:
    if feature.type == 'CDS':
        CDS_count += 1
        if "product" in feature.qualifiers:
            if feature.qualifiers["product"][0] == "hypothetical protein":
                hyp_CDS_count += 1

print("Total coding sequences in the E. coli K12 genome:", CDS_count)
print("Number of hypothetical proteins in the E. coli K12 genome:", hyp_CDS_cou
nt)
print("Fraction:", round(100*hyp_CDS_count/CDS_count, 2), "%")
```

```
Total coding sequences in the E. coli K12 genome: 4391
Number of hypothetical proteins in the E. coli K12 genome: 897
Fraction: 20.43 %
```

In [8]:
```
# Problem 4c

# Read in E. coli K12 genome:
in_handle = open("Ecoli_K12.gb", "r")
record = SeqIO.read(in_handle, "genbank")
in_handle.close()

# Count all the G and C nucleotides
GC_count = 0
for nucleotide in record.seq:
    if nucleotide=='G' or nucleotide=='C':
        GC_count += 1
# Calculate GC percentage
GC_perc = round(100*GC_count/len(record.seq), 2)
print("GC content: " + str(GC_perc) + "%")
```

```
GC content: 50.79%
```

If this was easy

Problem 5:

In the T7 genome, find the CDS feature corresponding to locus tag "T7p45" (major capsid protein), and then use the `feature.extract()` function to extract the DNA sequence corresponding to that gene. Then translate that sequence to verify (by inspection) that it corresponds to the protein sequence that is listed for gene T7p45.

In [9]:
```
in_handle = open("T7.gb", "r")
record = SeqIO.read(in_handle, "genbank")
in_handle.close()

for feature in record.features:
    if feature.type == 'CDS':
        locus_tag = feature.qualifiers['locus_tag'][0]
        if locus_tag == 'T7p45':
            target_feature = feature
            break

print(target_feature)
DNA_seq = target_feature.extract(record).seq
print("DNA:", str(DNA_seq))
print("protein:", DNA_seq.translate())
```

```
type: CDS
location: [22966:24004](+)
qualifiers:
    Key: codon_start, Value: ['1']
    Key: db_xref, Value: ['GOA:P19726', 'UniProtKB/Swiss-Prot:P19726', 'GeneID:
1261026']
    Key: locus_tag, Value: ['T7p45']
    Key: note, Value: ['major capsid protein. Involved in F-exclusion of wt T7
phage. A minor capsid protein (gp10B) is produced from gene 10 by a -1 frameshi
ft towards the end of 10A, resulting in a slightly larger protein. Other names:
gp10A.']
    Key: product, Value: ['major capsid protein']
    Key: protein_id, Value: ['NP_041998.1']
    Key: transl_table, Value: ['11']
    Key: translation, Value: ['MASMTGGQQMGTNQGKGVVAAGDKLALFLKVFGGEVLTAFARTSVTTS
RHMVRSISSGKSAQFPVLGRTQAAYLAPGENLDDKRKDIKHTEKVITIDGLLTADVLIYDIEDAMNHYDVRSEYTSQLG
ESLAMAADGAVLAEIAGLCNVESKYNENIEGLGTATVIETTQNKAALTDQVALGKEIIAALTKARAALTKNYVPAADRV
FYCDPDSYSAILAALMPNAANYAALIDPEKGSIRNVMGFEVVEVPHLTAGGAGTAREGTTGQKHVFPANKGEGNVKVAK
DNVIGLFMHRSAVGTVKLRDLALERARRANFQADQIIAKYAMGHGGLRPEAAGAVVFKVE']

DNA: ATGGCTAGCATGACTGGTGGACAGCAAATGGGTACTAACCAAGGTAAAGGTGTAGTTGCTGCTGGAGATAAACT
GGCGTTGTTCTTGAAGGTATTTGGCGGTGAAGTCCTGACTGCGTTCGCTCGTACCTCCGTGACCACTTCTCGCCACATG
GTACGTTCCATCTCCAGCGGTAAATCCGCTCAGTTCCCTGTTCTGGGTCGCACTCAGGCAGCGTATCTGGCTCCGGGCG
AGAACCTCGACGATAAACGTAAGGACATCAAACACACCGAGAAGGTAATCACCATTGACGGTCTCCTGACGGCTGACGT
TCTGATTTATGATATTGAGGACGCGATGAACCACTACGACGTTCGCTCTGAGTATACCTCTCAGTTGGGTGAATCTCTG
GCGATGGCTGCGGATGGTGCGGTTCTGGCTGAGATTGCCGGTCTGTGTAACGTGGAAAGCAAATATAATGAGAACATCG
AGGGCTTAGGTACTGCTACCGTAATTGAGACCACTCAGAACAAGGCCGCACTTACCGACCAAGTTGCGCTGGGTAAGGA
GATTATTGCGGCTCTGACTAAGGCTCGTGCGGCTCTGACCAAGAACTATGTTCCGGCTGCTGACCGTGTGTTCTACTGT
GACCCAGATAGCTACTCTGCGATTCTGGCAGCACTGATGCCGAACGCAGCAAACTACGCTGCTCTGATTGACCCTGAGA
AGGGTTCTATCCGCAACGTTATGGGCTTTGAGGTTGTAGAAGTTCCGCACCTCACCGCTGGTGGTGCTGGTACCGCTCG
TGAGGGCACTACTGGTCAGAAGCACGTCTTCCCTGCCAATAAAGGTGAGGGTAATGTCAAGGTTGCTAAGGACAACGTT
ATCGGCCTGTTCATGCACCGCTCTGCGGTAGGTACTGTTAAGCTGCGTGACTTGGCTCTGGAGCGCGCTCGCCGTGCTA
ACTTCCAAGCGGACCAGATTATCGCTAAGTACGCAATGGGCCACGGTGGTCTTCGCCCAGAAGCTGCTGGTGCAGTGGT
TTTCAAAGTGGAGTAA
protein: MASMTGGQQMGTNQGKGVVAAGDKLALFLKVFGGEVLTAFARTSVTTSRHMVRSISSGKSAQFPVLGRTQ
AAYLAPGENLDDKRKDIKHTEKVITIDGLLTADVLIYDIEDAMNHYDVRSEYTSQLGESLAMAADGAVLAEIAGLCNVE
SKYNENIEGLGTATVIETTQNKAALTDQVALGKEIIAALTKARAALTKNYVPAADRVFYCDPDSYSAILAALMPNAANY
AALIDPEKGSIRNVMGFEVVEVPHLTAGGAGTAREGTTGQKHVFPANKGEGNVKVAKDNVIGLFMHRSAVGTVKLRDLA
LERARRANFQADQIIAKYAMGHGGLRPEAAGAVVFKVE*
```

Problem 6:

For the influenza HA gene with ID "KT220438", write python code that checks whether the protein sequence listed in the CDS feature corresponds to the protein sequence you obtain when you translate the gene sequence. The code should print "The two translations match" or "The two translations do not match" depending on whether there is a match or not.

In [10]:
```
# download the record again
handle = Entrez.efetch(db="nucleotide", id="KT220438", rettype="gb", retmode="t
ext")
record = SeqIO.read(handle, "genbank")
handle.close()

# find the CDS feature
for feature in record.features:
    if feature.type == "CDS":
        CDS_feature = feature

# Extract the amino-acid sequence from the CDS feature. We must not forget the
[0]
# at the end to extract the first element of the list
CDS_feature_seq = CDS_feature.qualifiers["translation"][0]
# Extract the translated sequence from the DNA sequence in the record.
# We need to remove the trailing '*' indicating a stop codon,
# and we also need to convert the whole thing into a string.
translated_seq = str(record.seq.translate()[:-1])
if CDS_feature_seq == translated_seq:
    print("The two translations match.")
else:
    print("The two translations do not match.")
```

```
The two translations match.
```