Problem set #4 is online

Alexander Fleming discovered Antibiotics in 1928

Small peptide sequences that target and kill bacteria

Generated naturally by fungi and other bacteria

Today’s questions; Where do they come from?

How can one infer their sequence?
Discovery of Penicillin (1928)

- Upon returning from holiday in 1928 a British biologist and pharmacologist, Alexander Fleming, discovered a strange pattern of cell death in a stack of *staphylococci* cultures he had on a bench in a corner of his laboratory.
- Fleming noticed that one culture was contaminated with a fungus, *Penicillium*, and that the colonies of staphylococci immediately surrounding the fungus had been destroyed, whereas other staphylococci colonies farther away were normal, and famously remarked "That's funny".
- He attempted to isolate the agent that killed the bacteria, and even hypothesized that such an agent might be suitable for treating infections known to be caused by *staphylococci*. But, he had little luck.
- It proved hard to collect enough Penicillin to make it viable drug throughout the 1930s.
- Eventually, a mouldy cantaloupe found in Illinois in 1943, would provide a mold capable of generating high quantities of Penicillin and greatly reduce the World War II battlefield infection called *Sepsis*.
- The chemical structure of Penicillin was not determined until 1945 by Dorothy Hodgkin, another British X-ray crystallographer like Rosalind Franklin.
Other Antibiotics

- Meanwhile, a second important antibiotic, Gramicidin S or Gramicidin Soviet, was discovered by Russian microbiologist Georgyi Gause and his wife Maria Brazhnikova in 1942.
- This antibiotic was discovered in another strain of bacteria, Bacillus brevis, which had the tendency to kill staphylococci nearby it.
- It was actually easier to mass-produce than Penicillin, and it saved the lives of many Russians in the second world war, as well as the lives of its inventors.
- Gramicidin S is largely composed of a cyclodecapeptide, a protein-like peptide chain composed of 10 amino-acids joined head-to-tail to form a ring, called Tyrocidine B1.
- The species was reclassified into the genus Brevibacillus in 1996.
- The **Central Dogma of Molecular Biology** suggests that templates for such peptide chains are encoded in DNA.
- Our first task today is to see if we can find a genomic template where Gramicidin S is encoded.
Recall the Central Dogma from Lecture 1

Francis Crick, one of the discoverers of DNA structure, originally coined the term "Central Dogma" in 1958. That choice of words was particularly provocative.
DNA Transcription

- Triplets of nucleotide bases determine one of 20 amino acids in a protein's peptide sequence
- This mapping is called the **Genetic Code**
- Special STOP codes halt the translation process
- However, the DNA sequence that we see may not be a perfect indicator of the DNA sequence that produced it for two reasons.
  - Post-transcriptional modifications change the mRNA pattern. For example, in eucaryotes sections of code (Introns) are removed from the transcribed sequence.
  - Post-translational modifications modify the produced amino acid chain. For example when a protein is circularized
- So our goal is to find a partial pattern in the DNA which might have created Gramicidin S
- Recall the codon codings

Before a DNA sequence is translated into a protein, a copy is first made. This copy is made from RNA. In RNA, the nucleotide “Uracil” replaces “Thymine”. Uracil and Thymine are both chemically and structurally very similar.
Tyrocidine B1's Peptide Sequence

Tyrocidine B1 is a small peptide composed of only 10 amino acids.

Amino Acid: Valine Leucine Proline Phenylalanine Glutamine
3-letter Abbr: Val-Lys-Leu-Phe-Pro-Trp-Phe-Asn-Gln-Tyr
1-letter Abbr: V K L F P W F N Q Y
Amino Acid: Lysine Phenylalanine Tryptophan Asparagine Tyrosine

Also note that the Lysine is modified during the circularization into another non-proteinogenic amino acid, Ornithine.
A toolkit of Python Dictionaries

The following Python Dictionaries will be used to aid our search

```python
In [4]:

codon = {
    # Maps an RNA triplet of nucleotides to a 1-letter Amino Acid Abbreviation
    "AAA": 'K', "AAG": 'K', "AAC": 'N', "AAU": 'N',
    "AGA": 'R', "AGG": 'R', "AGC": 'S', "AGU": 'S',
    "ACA": 'T', "ACC": 'T', "ACG": 'T', "ACU": 'T',
    "AUA": 'I', "AUG": 'M', "AUC": 'I', "AUA": 'I',
    "GAA": 'E', "GAG": 'E', "GAC": 'D', "GAU": 'D',
    "GGA": 'G', "GGG": 'G', "GGC": 'G', "GGU": 'G',
    "GCA": 'A', "GCG": 'A', "GCC": 'A', "GCU": 'A',
    "GUA": 'V', "GUG": 'V', "GUC": 'V', "GUU": 'V',
    "CAA": 'Q', "CAG": 'Q', "CAC": 'H', "CAU": 'H',
    "CGA": 'R', "CGG": 'R', "CGC": 'R', "CGU": 'R',
    "CCA": 'P', "CCG": 'P', "CCC": 'P', "CCU": 'P',
    "CUA": 'L', "CUG": 'L', "CUC": 'L', "CUL": 'L',
    "UUA": 'I', "UGA": 'I', "UAC": 'Y', "UAU": 'Y',
    "UGA": 'W', "UGC": 'C', "UGU": 'C',
    "UCA": 'S', "UCG": 'S', "UCC": 'S', "UCU": 'S',
    "UUA": 'L', "UUG": 'L', "UUC": 'F', "UUU": 'F'
}
AminoAcid = {
    # Maps 1-letter Amino Acid Abbreviations to their full name
    'A': 'Alanine', 'C': 'Cysteine', 'D': 'Aspartic acid', 'E': 'Glutamic acid', 'F': 'Phenylalanine',
    'G': 'Glycine', 'H': 'Histidine', 'I': 'Isoleucine', 'K': 'Lysine', 'L': 'Leucine', 'M': 'Methionine',
    'N': 'Asparagine', 'P': 'Proline', 'Q': 'Glutamine', 'R': 'Arginine', 'S': 'Serine',
    'T': 'Threonine', 'V': 'Valine', 'W': 'Tryptophan', 'Y': 'Tyrosine', 'STOP': 'STOP'
}
AminoAbbrv = {
    # Maps 1-letter Amino Acid Abbreviations to 3-letter Abbreviations
    'A': 'Ala', 'C': 'Cys', 'D': 'Asp', 'E': 'Glu', 'F': 'Phe', 'G': 'Gly', 'H': 'His', 'I': 'Ile',
    'K': 'Lys', 'L': 'Leu', 'M': 'Met', 'N': 'Asn', 'P': 'Pro', 'Q': 'Gln', 'R': 'Arg', 'S': 'Ser',
    'T': 'Thr', 'V': 'Val', 'W': 'Trp', 'Y': 'Tyr', 'STOP': 'STOP'
}
```
Now our Peptide sequence

We'll use the 1-letter Amino Acid abbreviations to represent our sequence, and create a dictionary that provides for every peptide in our chain the set of codons that could encode it.

```python
In [7]:
    TrimerCodes = {}
    for key, code in codon.items():
        TrimerCodes[code] = TrimerCodes.get(code, []) + [key]
    for key in sorted(TrimerCodes.keys()):
        print("%s (%s): %s\% (key, AminoAcid[key], TrimerCodes[key])")
```

Y (STOP): ['UAA', 'UAG', 'UGA']
A (Alanine): ['GCA', 'GCC', 'GCU', 'GCG']
C (Cysteine): ['UGC', 'UGU']
D (Aspartic acid): ['GAC', 'GAU']
E (Glutamic acid): ['GAA', 'GAG']
F (Phenylalanine): ['UUC', 'UUU']
G (Glycine): ['GGA', 'GGG', 'GGC', 'GGU']
H (Histidine): ['CAC', 'CAU']
I (Isoleucine): ['AUA', 'AUC', 'AUU']
K (Lysine): ['AAA', 'AAG']
L (Leucine): ['CUA', 'CUG', 'CUC', 'CUU', 'UUA', 'UUG']
M (Methionine): ['AUG']
N (Asparagine): ['AAC', 'AAU']
P (Proline): ['CCA', 'CCC', 'CCU', 'CCG']
Q (Glutamine): ['CAA', 'CAU']
R (Arginine): ['AGA', 'AGG', 'CGA', 'CGG', 'CGC', 'CGU']
S (Serine): ['AGC', 'AGU', 'UCA', 'UCG', 'UCC', 'UCU']
T (Threonine): ['ACA', 'ACG', 'ACC', 'ACU']
V (Valine): ['GUA', 'GUG', 'GUC', 'GUU']
W (Tryptophan): ['UGG']
Y (Tyrosine): ['UAC', 'UAA']
How many ways to code Tyrocidine B1?

- Since most Amino Acids have multiple codon encodings, most peptide sequences can be codes in multiple ways.
- Let's figure out exactly how many ways there are to encode our little peptide.

```python
In [4]:TyrocidineB1 = "VKLPWFNQY"
codes = 1
for residue in TyrocidineB1:
    print(residue, AminoAbbrev[residue], TrimerCodes[residue])
codes *= len(TrimerCodes[residue])
print("%d possible sequences" % codes)
```

V Val ['GUU', 'GUG', 'GUC', 'GUU']
K Lys ['AAA', 'AAG']
L Leu ['CUA', 'CUG', 'CUU', 'CUA', 'UUA', 'UUG']
F Phe ['UUU', 'UUG']
P Pro ['CCA', 'CCG', 'CCC', 'CCU']
W Trp ['UGG']
Y Tyr ['UAC', 'UAU']
6144 possible sequences
Where do we start?

- Consider that translation could begin at any point in the genome.
- Genes can be encoded on either strand

```
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
```

- There are 3 possible open-read frames (mod-3 starting positions) on each strand.
- Meanwhile, our peptide has 10 possible starting positions, since it is circular we can't be sure which codon appears first.
- Let’s start with something simple
  - Converting a DNA sequence into a protein sequence.
  - In other words, the computational version of translation
Some Helper Functions

- Reverse-complement a sequence
- Translate a DNA sequence to a Protein sequence

```python
In [14]:
# one way to code TyrocidineB1
DNA = 'GTGAAACTTTTTTCTTTGGTTTAAATCAATAT'
DNAR = revComp(DNA)

print("A short DNA sequence")
print("5'-3'" % DNA)
print("3'-5'" % ' '.join(["A":'T','C'::'G','G'::'C','T'::'A']string[base] for base in DNA))

print("Read frames in primary sequence")
for frame in range(3):
    RNA, Peptides = proteinTranslation(DNA[frame:],DNA[frame:1])
    print("%d %s" % (frame, DNA))
    print("%s %s" % (frame, RNA))
    print("%s %s" % (frame, Peptides))
    print()

print("Read frames in the reverse-complement sequence")
for frame in range(3):
    RNA, Peptides = proteinTranslation(DNAR[frame:],DNAR[frame:1])
    print("%d %s" % (frame, DNAR))
    print("%s %s" % (frame, RNA))
    print("%s %s" % (frame, Peptides))
    print()
```

A short DNA sequence
5'-GTGAAACTTTTTTCTTTGGTTTAAATCAATAT-3'
3'-CATTCTGAAAAGGAACACAAATATTAGTATAA-T5'

Read frames in primary sequence
1 GTGAAACTTTTTTCTTTGGTTTAAATCAATAT
     GUGAACUUUCUCCUUGGAAUAAUCAAUAG
     ValLysLeuPheProTrpPheAsnGlnTyr
2 GTGAAACTTTTTTCTTTGGTTTAAATCAATAT
     UGAAACUCUUUCUCCUUGGAAUAAUCAAUAG
     STPAsnPhePheLeuGlyLeuIleAsnMet
3 GTGAAACTTTTTTCTTTGGTTTAAATCAATAT
     GAAACUCUUUCUCCUUGGAAUAAUCAAUAGU
     GluThrPhesErLeuValSTPSerIleCys

Read frames in the reverse-complement sequence
1 ATATTGATTTAAACCAAGAAAAAGTTCAC
     AUAUUGAUUAAACCAAGAAAAAGUUGUCAC
     IleLeuIleLysProArgLysPheHis
2 ATATTGATTTAAACCAAGAAAAAGTTCAC
     UAUUGAUUAAACCAAGAAAAAGUUGUCACA
     TyrSTPLeuAsnGlnGlyLysSerPheThr
3 ATATTGATTTAAACCAAGAAAAAGTTCAC
     AUUGAUUAAACCAAGAAAAAGUUGUCACA
     IleAspSTPThrLysGluLysValSerHis
```
Now let's try Something Real

```python
# Load FASTA file
def loadFasta(filename):
    """Parses a classically formatted and possibly compressed FASTA file into a list of headers and fragment sequences for each sequence contained""
    if filename.endswith(".gz"):
        fp = gzip.open(filename, 'r')
    else:
        fp = open(filename, 'r')
    data = fp.read().split('>')
    fp.close()
    # Ignore whatever appears before the 1st header
    data.pop(0)
    headers = []
    sequences = []
    for sequence in data:
        lines = sequence.split('
')
        headers.append(lines.pop(0))
        # Add an extra "+" to make string "1-referenced"
        sequences.append('+'.join(lines))
    return (headers, sequences)

header, seq = loadFasta("data/BacillusBrevis.fa")
for i in range(len(header)):
    print(header[i])
    print(len(seq[i])-1, "bases", seq[i][:30], "...", seq[i][-30:])
    print()
```

Chromosome dna:chromosome chromosome:GCA_000010165.1:Chromosome:1:6296436:1
6296436 bases +GGGTCTGTG6ATATATTATCCACAAA ... AAGGCAATATCCCATAAAAACTATTCCC
Some simple "Computational" experiments

Just like a biologist, it is ill-advised to jump into a problem without first getting some sense of what works and what doesn't.

- Let's first take a look at our data
- One thing you might have noticed about the possible encodings of Tyrocidine B1 is that there is a single *Tryptophan* and only one encoding for this amino acid
- Let's try to anchor our search around that seed and grow the full sequence from there.
- Where and how many *Tryptophan* encodings

```python
In [19]:
    def CodonSeqCount(genome, codonSeq):
        N = 0
        start = 0
        while True:
            pos = genome.find(codonSeq, start)
            if (pos < 0):
                break
            N += 1
            start = pos + 1
        return N

genome = seq[8]
tryptophanCode = TrimerCodes['W'][0].replace('U','T')
print("Searching for", tryptophanCode, "Found it", CodonSeqCount(genome,tryptophanCode), "times")
revCompCode = revComp(tryptophanCode)
print("Searching for", revCompCode, "Found it", CodonSeqCount(genome,revCompCode), "times")

Searching for TGG Found it 106555 times
Searching for CCA Found it 107112 times
```

Is this about what you expected? $\frac{6.3 \times 10^6}{64} \approx 100,000$. We've narrowed our search some, but there's still allot.
Narrowing things down more

Rather than searching for every possible string, let's examine the codons around this initial anchor.

```python
def AminoAcidSeqCount(genome, peptideSeq):  # Note: Only checks primary strand
    readframe = []
    anchor = peptideSeq[0]
    for rnaSeq in TrimerCodes[anchor]:
        dnaSeq = rnaSeq.replace('U', 'T')
    start = 0
    while True:
        pos = genome.find(dnaSeq, start)
        if (pos < 0):
            break
            i = 1
        while (i < len(peptideSeq) and (pos+3*i+1) <= len(genome)):
            if (codon[genome[pos+3*i:pos+3*(i+1)].replace('T', 'U')] != peptideSeq[i]):
                break
            i += 1
        else:
            readframe.append(pos)
            start = pos + 1
    return readframe

peptide = "WFNQY"
for i in range(1, len(peptide)+1):
    ORFs = AminoAcidSeqCount(genome, peptide[:i])
    print("Searching for ", peptide[:i], " Found it ", len(ORFs), " times")
```

Searching for W Found it 106555 times
Searching for WF Found it 3922 times
Searching for WFN Found it 111 times
Searching for WFNQ Found it 5 times
Searching for WFNQY Found it 1 times
Could we have been Lucky this time?

Let's take a look.

In [23]:
   for pos in ORFs:
       start = pos
       end = pos + (3*len(peptide))
       print("%9d %s %s" % (start, genome[start:end], proteinTranslation(genome[start:end])[1]))

2789614 TGGTTCAACCAATAT TrpPheAsnGlnTyr

We didn't find what we were expecting.
What Went Wrong?

1. We only searched in one direction from our anchor
2. We didn't search the reverse-complement sequence
3. We didn't consider that the cycle could have been broken somewhere in between "W-F-N-Q-Y"
   - One of the earlier 5, 111, 3922, or 106555 candidates might be the solution
4. The best approach might be to call AminoAcidSeqCount() with all 10 circular permutations of "VKLFPWFNQY" (assuming that we also fix the reverse-complement sequence problem)

```python
In [24]: [TyrocidineB1[i:]+TyrocidineB1[:i] for i in range(len(TyrocidineB1))]
Out[24]: ['VKLFPWFNQY', 'KLFPWFNQYV', 'LFPWFNQYVK', 'FPWFNQYVKL', 'PWFNQYVKLF', 'WFNQYVKLFP', 'FNQYVKLFPW', 'NQYVKLFPWF', 'QYVKLFPWFN', 'YVKLFPWFNQ']
```

But even if you tried all these fixes, you would still not find the code for Tyrocidine B1 in *Brevibacillus brevis*.
Proteins that make Peptide Chains

The Central Dogma is not the only way to make proteins!

Bacteria and fungi use large multifunctional enzymes called NonRibosomal Peptide Synthetases (NRPSs) to produce peptide chains that are not encoded in DNA. Nonribosomal peptides are synthesized by nonribosomal peptide synthetases, which, unlike the ribosomes, are independent of messenger RNA. Each nonribosomal peptide synthetase can synthesize only one type of peptide. They are often toxins, siderophores, or pigments. Nonribosomal peptide antibiotics, cytostatics, and immunosuppressants are widely used as drugs.
So we are left with a new Question

- How do you figure out a peptide's amino acid sequence? We just saw it may not appear in the genome.
- Next time we will delve deeper into laboratory methods to start a discussion about peptide sequence inference.