Homework 1

Dieu My Nguyen | MCDB 5520 | February 9, 2018

Question #1. Calculate the following:
(a) (3pts) P(s = "MENDEL")
(b) (3pts) P(s = "ROSALIND")
(c) (4pts) P(s = "charged" or "aromatic")

Dictionary of codon usage table as percentages; this is our alphabet:

```
In [152]: codons_usage = {"A": 7.81, "Q": 3.94, "L": 9.62, "S": 6.88,
          "R": 5.32, "E": 6.60, "K": 5.93, "T": 5.45,
          "N": 4.20, "G": 6.93, "M": 2.37, "W": 1.15,
          "D": 5.30, "H" : 2.28, "F" : 4.01, "Y" : 3.07,
          "C": 1.56, "I" : 5.91, "P" : 4.84, "V" : 6.71}
```

New dictionary of codon usage table as probabilities:

```
In [162]: codons_prob = {}
          for i in codons_usage.keys():
              codons_prob[i] = codons_usage[i] / 100
```

```
In [163]: codons_prob
```

```
Out[163]: {'A': 0.0781,
           'C': 0.015600000000000001,
           'D': 0.053,
           'E': 0.066,
           'F': 0.0401,
           'G': 0.0693,
           'H': 0.022799999999999997,
           'I': 0.0591,
           'K': 0.0593,
           'L': 0.0962,
           'M': 0.023700000000000002,
           'N': 0.042,
           'P': 0.0484,
           'Q': 0.0394,
           'R': 0.053200000000000004,
           'S': 0.0688,
           'T': 0.0545,
           'V': 0.06709999999999999,
           'W': 0.0115,
           'Y': 0.030699999999999998}
```

Assuming the codons are independent, then the problem is a logical AND and we multiple individual amino acid probabilities to obtain the probability of a particular sequence. We also assume that we can have repeats (sampling with replacement).

The function below does this. Note that since there is no "O", the probability of "ROSALIND" is 0 because P("O") = 0.

```
In [203]: def prob_protein(protein, codon_dict):
              prob = 1
              for aa in protein:
                  if aa not in codon_dict.keys():
                      aa = 0
                      prob *= aa

                  else:
                      prob *= codon_dict[aa]

              return prob

          print("A) Probability of 'MENDEL': " + str(prob_protein("MENDEL", codons
          _prob)))
          print("B) Probability of 'ROSALIND': " + str(prob_protein("ROSALIND", co
          dons_prob)))
```

```
          A) Probability of 'MENDEL': 2.2107337892640003e-08
          B) Probability of 'ROSALIND': 0.0
```

Now, the probability of "charged" or "aromatic" with the same indepedent assumption:
According to this site: https://www.mcb.ucdavis.edu/courses/bis102/Aromatic.html (https://www.mcb.ucdavis.edu/courses/bis102/Aromatic.html)

Aromatic amino acids: F, Y, W
Charged amino acids: D, E, H, K, R Logical OR: P(charged or aromatic) = P(F)+P(Y)+P(W)+P(D)+P(E)+P(H)+P(K)+P(R)

```
In [204]: aromatic = ["F", "Y", "W"]
          charged = ["D", "E", "H", "K", "R"]
          merged = aromatic + charged
          merged
```

```
Out[204]: ['F', 'Y', 'W', 'D', 'E', 'H', 'K', 'R']
```

```
In [208]: def prob_arom_charged(codon_dict, arom, charged):
              prob = 0
              for aa in codon_dict:
                  if aa in arom or aa in charged:
                      prob += codon_dict[aa]
              return prob

          print("C) Probability of charged or aromatic: " + str(prob_arom_charged(
          codons_prob, aromatic, charged)))

          C) Probability of charged or aromatic: 0.33659999999999995
```

Question #2. We observe the following empirical frequencies for dinucleotides, where the first nucleotide s(i) is the row and the second nucleotide s(i+1) is given in the columns, hence the P(GC) = 0.0522. Convert the above frequency matrix into a transition matrix for the Markov model of di-nucleotide sequences discussed in class. Note that each entry of the transition matrix is the conditional probability: P(s(i+1)| s(i)).

```
 In [48]: import numpy as np
```

```
In [140]: freq_mat = array([
              [0.1202, 0.0505, 0.0483, 0.0912],
              [0.0665, 0.0372, 0.0396, 0.0484],
              [0.0514, 0.0522, 0.0363, 0.0499],
              [0.0721, 0.0518, 0.0656, 0.1189]
          ])
```

Get sum of each row, add to frequency matrix as last column:

```
In [141]: row_sums = list(freq_mat.sum(axis=1))
```

```
In [142]: B = np.column_stack([freq_mat, row_sums])
          B

Out[142]: array([[0.1202, 0.0505, 0.0483, 0.0912, 0.3102],
                 [0.0665, 0.0372, 0.0396, 0.0484, 0.1917],
                 [0.0514, 0.0522, 0.0363, 0.0499, 0.1898],
                 [0.0721, 0.0518, 0.0656, 0.1189, 0.3084]])
```

Summation of each row in the transition matrix = 1, so in the frequency matrix each element in the row was divided by the sum of the row. The transition matrix is:

```
In [143]: trans_mat = []
          for row in B:
              print(row[0:-1]/row[-1])
              trans_mat.append(row[0:-1]/row[-1])

          [0.38749194 0.16279819 0.155706   0.29400387]
          [0.34689619 0.19405321 0.20657277 0.25247783]
          [0.27081138 0.27502634 0.19125395 0.26290832]
          [0.23378729 0.16796368 0.21271077 0.38553826]
```

Question #3. The sum of all amino acids should be 1. The sum of all codons is 1. However, STOP is not a valid amino acid. Yet we should still be able to calculate the probability of any amino acid from the nt frequencies, by proper normalization. In this case, we normalize by the total probability that leads to codons. So: P(amino acid) = P(all codons for the amino acid) / P(all codons that are valid amino acids). Assume P(A) = 0.3, P(T) = 0.3, P(C) = 0.2, and P(G) = 0.2

(4pts) What is the P(all codons that are valid amino acids)?

P(all possible codons) = 64/64 = 1

STOP codons: TAA, TAG, TGA; P(a STOP codon) = 3/64

P(all codons that are valid aa) = 61/64

P(a STOP codon) + P(all codons that are valid aa) = 1

P(all codons that are valid aa) = 1 - P(a STOP codon)

P(all codons that are valid aa) = 1 - [P(TAA)+P(TAG)+(TGA)]

Assuming independent nucleotides:

P(TAA) = P(T)*P(A)*P(A) = 0.3 * 0.3 * 0.3 = 0.027

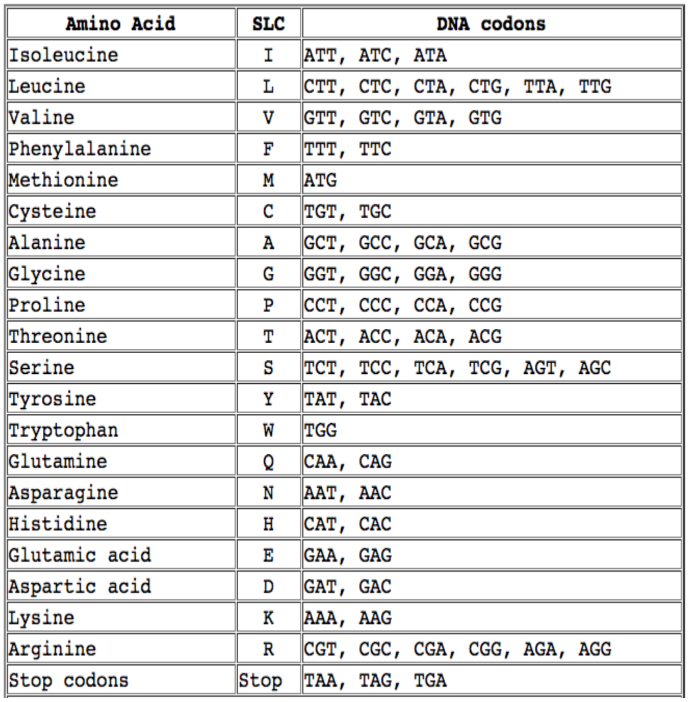
P(TAA) = P(T)*P(A)*P(G) = 0.3 * 0.3 * 0.2 = 0.018

P(TAA) = P(T)*P(G)*P(A) = 0.3 * 0.2 * 0.3 = 0.018

P(all codons that are valid aa) = 1 - [0.027+0.018+0.018] = 0.937

Using proper normalization, what is the probability of the following amino acids?

According to this DNa codon table:



Amino Acid	SLC	DNA codons
Isoleucine	I	ATT, ATC, ATA
Leucine	L	CTT, CTC, CTA, CTG, TTA, TTG
Valine	V	GTT, GTC, GTA, GTG
Phenylalanine	F	TTT, TTC
Methionine	M	ATG
Cysteine	C	TGT, TGC
Alanine	A	GCT, GCC, GCA, GCG
Glycine	G	GGT, GGC, GGA, GGG
Proline	P	CCT, CCC, CCA, CCG
Threonine	T	ACT, ACC, ACA, ACG
Serine	S	TCT, TCC, TCA, TCG, AGT, AGC
Tyrosine	Y	TAT, TAC
Tryptophan	W	TGG
Glutamine	Q	CAA, CAG
Asparagine	N	AAT, AAC
Histidine	H	CAT, CAC
Glutamic acid	E	GAA, GAG
Aspartic acid	D	GAT, GAC
Lysine	K	AAA, AAG
Arginine	R	CGT, CGC, CGA, CGG, AGA, AGG
Stop codons	Stop	TAA, TAG, TGA

(3pts) Ile (I)

DNA codons for I: ATT, ATC, ATA
P(all codons for I) = P(ATT) + P(ATC) + P(ATA)
= [P(A)*P(T)*P(T)] + [P(A)*P(T)*P(C)] + [P(A)*P(T)*P(A)]
= [0.3*0.3*0.3] + [0.3*0.3*0.2] + [0.3*0.3*0.3]
= 0.072

P(I) = P(all codons for I) / P(all codons that are valid aa) = 0.072 / 0.937 = 0.077

(3pts) Trp (W)

DNA codons for W: TGG
P(all codons for W) = P(TGG)
= P(T)*P(G)*P(G)
= 0.3*0.2*0.2
= 0.012

P(W) = P(all codons for W) / P(all codons that are valid aa) = 0.012 / 0.937 = 0.013

Question #4. Consider the following read returned from the sequencing facility:

@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345
GCATGTGGTGAGGTGGTAGTGATGGTGATATAGAGTGGTAGTATAAGTGT
+
IIIIIIIIIIIIIIIIIIIGIIIIIIIIIIIIIIIIIIIIIIAIIGIICI
Recommendation: Refer to the Wikipedia page on FASTQ format for the encoding schemes discussed below. </b>

(a) (2pts) Assume the quality scores are encoded using the Sanger offset (Phred+33). Is this sequence of generally good quality?

Phred+33 has the range 0 to 40. Resource:
https://support.illumina.com/help/BaseSpace_OLH_009008/Content/Source/Informatics/BS/QualityScoreE
(https://support.illumina.com/help/BaseSpace_OLH_009008/Content/Source/Informatics/BS/QualityScoreE

In [226]:
```
from Bio import SeqIO
# "fastq" is Sanger style with ascii offset of 33
for record in SeqIO.parse("fastq_Q4.fq", "fastq"):
    print("Sequence:")
    print("%s %s" % (record.id, record.seq))
    print("\nPhred quality score:")
    scores = record.letter_annotations["phred_quality"]
    print(scores)
```

```
Sequence:
SRR001666.1 GCATGTGGTGAGGTGGTAGTGATGGTGATATAGAGTGGTAGTATAAGTGT

Phred quality score:
[40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 4
0, 40, 38, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40,
40, 40, 40, 40, 40, 40, 40, 32, 40, 40, 38, 40, 40, 34, 40]
```

Each score represents the probability that the corresponding nucleotide call is incorrect. The greater the score, the lower the probability and higher the base call accuracy percentage. Looking at the above scores, which has mostly 40s and the lowest is 32, this sequence is generally good quality.

(b) (5pts) Under this encoding, what base is the lowest quality? (You may circle it in the above) What is the probability of this position being correct?

In [223]:
```
# Find index of lowest Phred score (32) from above list:
import numpy as np
index_min = np.argmin(scores)
index_min
```

Out[223]:
```
42
```

```
In [225]: # Get base that has the lowest score:
          record.seq[42]
```

```
Out[225]: 'A'
```

The probability of this position (base "A") being incorrect is about 0.00063. Therefore, the probability of it being correct is: 1 - 0.00063 = 0.99937. Resource: http://www.genomicidlab.com/quality-1 (http://www.genomicidlab.com/quality-1)

(c) (3pts) You realize that you were mistaken in the encoding and it is given in the Illumina 1.3+ (Phred+64) format. Under this encoding scheme, is this sequence of generally good quality? Is the worst position still the one you circled in question b?

```
In [227]: # "fastq-illumina" is newer Illumina 1.3-1.7 files with ascii offset 64
          for record in SeqIO.parse("fastq_Q4.fq", "fastq-illumina"):
              print("Sequence:")
              print("%s %s" % (record.id, record.seq))
              print("\nPhred quality score:")
              scores2 = record.letter_annotations["phred_quality"]
              print(scores2)
```

```
Sequence:
SRR001666.1 GCATGTGGTGAGGTGGTAGTGATGGTGATATAGAGTGGTAGTATAAGTGT

Phred quality score:
[9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 7, 9, 9, 9,
9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 1, 9, 9, 7, 9,
9, 3, 9]
```

```
In [228]: # Find index of lowest Phred score (32) from above list:
          index_min2 = np.argmin(scores2)
          index_min2
```

```
Out[228]: 42
```

Under this encoding scheme, the quality scores are on the low end of the spectrum (higher probability of error), indicating a generally poor quality sequence. The worst position is still at index 42 (starting from 0 in Python), base "A."

Question #5. You begin to sequence the genome of Tamatoa, gathering 5,000 reads that were each 600 base pairs long. You have hypothesized that the Tamatoa genome is about 2 million bases long.

(a) (5pts) At what coverage have you sequenced the genome thus far?

$C = \frac{nL}{G}$;

where G = length of genomic seq; n = number of reads; L = length of each read

$C = \frac{5000*600}{2000000} = 1.5$

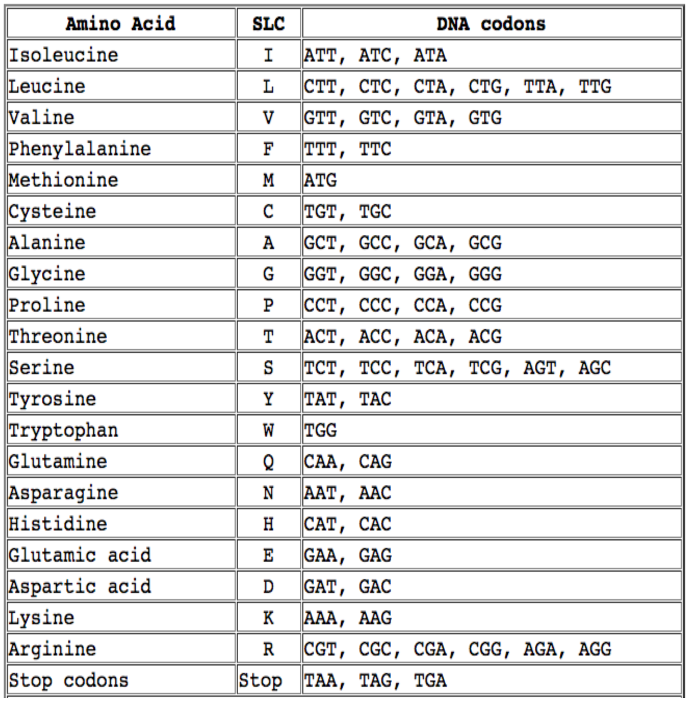
(b) (5pts) If the coverage of the Tamatoa genome were 6X, what is the probability that a base will be unsequenced?

$f(k; \lambda) = P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$; **where** $\lambda = C$ **and** $k = 0$ **for unsequenced base probability**

$f(0; 6) = P(X = 0) = \frac{6^0 e^{-6}}{0!} = 0.0025 = 0.25\%$ **chance that a base will be unsequenced**

Question #6. (Advanced) Due to redundancy in the genetic code, a sequence of amino acids could be encoded by several DNA sequences. For a ten amino acid long protein fragment, what is the lower and upper bound for the number of possible DNA sequences that can encode this protein sequence? (5 pts per bound)

According to this DNa codon table:



Amino Acid	SLC	DNA codons
Isoleucine	I	ATT, ATC, ATA
Leucine	L	CTT, CTC, CTA, CTG, TTA, TTG
Valine	V	GTT, GTC, GTA, GTG
Phenylalanine	F	TTT, TTC
Methionine	M	ATG
Cysteine	C	TGT, TGC
Alanine	A	GCT, GCC, GCA, GCG
Glycine	G	GGT, GGC, GGA, GGG
Proline	P	CCT, CCC, CCA, CCG
Threonine	T	ACT, ACC, ACA, ACG
Serine	S	TCT, TCC, TCA, TCG, AGT, AGC
Tyrosine	Y	TAT, TAC
Tryptophan	W	TGG
Glutamine	Q	CAA, CAG
Asparagine	N	AAT, AAC
Histidine	H	CAT, CAC
Glutamic acid	E	GAA, GAG
Aspartic acid	D	GAT, GAC
Lysine	K	AAA, AAG
Arginine	R	CGT, CGC, CGA, CGG, AGA, AGG
Stop codons	Stop	TAA, TAG, TGA

If the protein is consisted of only Tryptophans or Methionines, it could be encoded by only 1 DNA sequence as these amino acids are each encoded by a single codon. Thus, lower bound = 1 possible sequence.

If the protein is consisted of only Arginine, Leucine, or Serine, it could be encoded by 6^{10} DNA sequences as these amino acids are each encoded by 6 codons. Thus, upper bound = 6^{10} or 60466176 possible sequences.

Question #7. (Advanced) Consider Ravenhall et. al. Inferring Horizontal Gene Transfer. PLoS Comp Biol 11(5): e1004095 (2015). doi:10.1371/journal.pcbi.1004095 (This article is available on Canvas as RavenhallPLoS2015.pdf).

(5pts) (a) Briefly describe the difference between parametric and phylogenetic approaches to detecting horizontal gene transfer (HGT).

The parametric approach is based on sequence composition, searching for significant deviations from genomic averages, using metrics such as GC content or codon usage. The phylogenetic approach is based on evolutionary history, to identify genes that have significantly different evolutionary history from the history of a host species. This approach can be divided into explicit methods (reconstruct phylogenetic trees and explicitylu infer horizontal gene transfer events that resulted in that tree) and implicit methods (bypass gene reconstruction and instead looks at species and gene distances).

(5 pts) (b) What are the pros and cons of the parametric approaches?

Pros: Only need genome of interest and its genomic signatures to infer horizontal gene transfer events;

Cons: Overpredictions resulting from the method not accounting for intragenomic variability of the host and marking native segments as HTG events; low ability to detect ancient HGTs due to amelioration;