Problem set 7:
Information in DNA sequence data

BIO-369

Prof. Anne-Florence Bitbol
EPFL

This problem was given as part of a graded numerical project for this class.

1 DNA sequence logo

In this problem, we will consider the data in the file `crp_sites.fa`, which contains DNA sequences of various binding sites of the same transcriptional regulator CRP in the *Escherichia coli* genome. If you open this file, which is in a format called `fasta`, you will see that, for each entry, there is first a header line starting with ">", and then the next line is the sequence itself as a string of letters A, T, C and G. These sequences are already aligned, meaning that they all have the same length and that the first nucleotide in the first sequence has the same role as the first nucleotide in the second sequence, and so on. We will call these positions in the alignment the columns of the alignment.

We would like to understand the conserved features of CRP binding sites from these examples of different binding site sequences. For this, we will first compute the frequency of occurrence of each nucleotide (A, T, C, G) in each column. The frequency of A in the first column is the number of sequences where there is an A in the first column divided by the total number of sequences in the alignment.

a) In Python, load the file `crp_sites.fa` and extract all the sequences in it as an array of strings. We will not need the headers for this analysis, so you can discard them.

   There are different ways to do this, and for instance you may use the function `SeqIO.parse` from `Bio`, and use `list` to transform its output into a list. Once you have obtained this list, sequences without headers can be obtained from it using the method `seq`, for instance if your list is called `mylist`, you can obtain the first sequence as a string from it by using `str(mylist[0].seq)`. Again, this is just one possible method, there are other possibilities.

b) Next, in order to be able to count the nucleotides in each column, transform the array of strings you obtained into a Numpy array of integer numbers, by using the mapping that A is 0, C is 1, G is 2 and T is 3.

   Again, there are different ways to do this, but one of them is to define the mapping as a Python dictionary using `dict`, and then to use it for each character in each row of the array of strings to replace it by the appropriate integer number.

c) Produce a Numpy array called `frequencies` that contains the frequency of each nucleotide in each column of the alignment. Each column should correspond to a nucleotide, and each row should correspond to a column of the alignment.

d) Produce a Numpy array called `entropies` that contains the entropy of each column of the alignment, computed from the frequencies determined before. Each row should correspond to a column of the alignment.

e) Plot entropy versus column index, using a bar plot. What is the maximum possible value that entropy could theoretically take for such a column? Which distribution would it correspond to? Compare columns 8 and 11 (where the first column is numbered 0 as per Python convention): which one has the largest entropy?

f) In light of the previous problem set, how much are you concerned about finite size effects in your entropy estimates for question 2e? What would change for these finite size effects if protein sequences were considered instead of DNA sequences?

g) Plot the difference between the maximal possible value of entropy and the actual value of entropy versus column index, using a bar plot. We will call this quantity the "height" of each column of the alignment, as it will become the height in a new logo we will construct later. How is this quantity related to conservation (a column is said to be highly conserved if it features the same amino acid in most sequences)? What are the minimum and maximum possible values of the height? Compare columns 8 and 11 again.

h) Produce a Numpy array called `products` that contains, for each nucleotide and each column of the alignment, the product of the frequency of each nucleotide in each column of the alignment and of the quantity "height" defined just before for each column of the alignment. Each column should correspond to a nucleotide, and each row should correspond to a column of the alignment.

i) Now we will make a graphical representation of the data, by using `logomaker` [1]. For this, first install `logomaker` e.g. from PyPI by executing `pip install logomaker` at the command line. Next, execute the following Python commands in your Jupyter notebook:
```
products_pd = pd.DataFrame(data=products,columns=["A","C","G","T"])
lm.Logo(products_pd)
```
You should now see a sequence logo, where the total height of each column (the total height of the stack of four letters) is the quantity that we called "height". In addition, at each site (in each column) the height of each letter is proportional to the frequency of the corresponding nucleotide, and the most frequent nucleotide is on top, the next frequent one just below, and so on.

j) The consensus sequence is the sequence that has the most frequent nucleotide at each site. How can you see the consensus sequence in the logo you just produced?

k) In what sense is this new logo a good representation of the data in the alignment?

l) Where on the DNA sequences studied here do you think the strongest binding of CRP to DNA occurs?

References

[1] A. Tareen and J. B. Kinney. Logomaker: beautiful sequence logos in Python. *Bioinformatics*, 36(7):2272–2274, 04 2020.