Introduction to Bioinformatics
Making and Using Position-Specific Scoring Matrices (Part III)

By the end of the last tour (*Making and Using Position-Specific Scoring Matrices, Part II*), you had used a PSSM built from an ungapped region around proven NtcA-binding sites to scan the intergenic regions of *Synechocystis* PCC 6803. You no doubt found lots of genes near intergenic regions supposedly containing a sequence like the desired binding site… let's get there again:

```
(APPLY-PSSM-TO (INTERGENIC-SEQUENCES-OF S6803)
     WITH-PSSM-FROM S6803-NtcA-sites-short)
```

where `S6803-NtcA-sites-short` is defined as described in the last notes. It finds a lot of genes all right. How can we tell if the function is finding what it's supposed to? Certainly the function ought to find the genes of the training set. Recall the motifs used in the training set and their associated genes by listing the value of `S6803-NtcA-sites-short`. Compare these genes with the genes obtained by `APPLY-PSSM-TO`. How?

SQ1. Obtain the descriptions of the genes associated with the putative NtcA-binding sites by suitably modifying the following command:

`(DESCRIPTIONS-OF (COLUMN 1 OF (RESULT `*`n`*`)) DISPLAY)`

where *n* is the line number for the `APPLY-PSSM-TO` command.

The `COLUMN` command is often useful, and you should take the time to understand what it means. Lists within lists, such as:

```
(("S6803:amtl    " "TGAAAAGTAGTAAATCATACAGAAAACAATCATGTAAAA")
 ("S6803:glnA    " "AAAATGGTAGCGAAAAATACATTTTCTAACTACTTGACT")
 ("S6803:glnB    " "CAAACGGTACTGATTTTTACAAAAAAACTTTTGGAGAAC")
 ...
```

can be thought of as a table with multiple columns. The first element of each sublist (in this case `"S6803:amt1"`, `"S6803:glnA"`, etc) constitutes the first column.

SQ2. Suppose you want to find the two genes adjacent to a given gene. You could do this by adapting the following example:

`(CONTEXT-OF alr2339 GENE-WIDTH 1)`

Note that the function *displays* alr2339 and its two flanking genes, and the function *returns* a list containing the three genes. Do the same thing for all the orthologs of alr2339, by substituting `(ORTHOLOGS-OF alr2339)` for `alr2339`. Note that the output of the function is now a list of lists:

```
((#$Npun.NpF1721 #$Npun.NpR1722 #$Npun.NpR1723)
 (#$A7120.alr2338 #$A7120.alr2339 #$A7120.all2340)
 (#$Tery.Te?4066 #$Tery.Te?4067 #$Tery.Te?4199)
 (#$A29413.Av?1813 #$A29413.Av?1902 #$A29413.Av?1901))
```

where each sublist consists of the left-hand gene, the ortholog, and the right-hand gene in each organism where an ortholog is found. Use `COLUMN-OF` to extract from this list of lists the four right-hand genes.

…back to PSSMs. By scanning the descriptions obtained in SQ1, you would expect to find all five genes used in the training set. Hard to say, since the training set gives only gene abbreviations that are probably obscure to you. We need the descriptions of the genes of the training set.

SQ3. Obtain genes of the training set with the following command:

```
(GENES-DESCRIBED-BY {"amt1" "glnA" ... "rpoD2"} IN S6803)
```

where … is replaced by the remaining gene abbreviations in the training set. Note that the output is a list of list. Now get the descriptions of these genes.

SQ4. Compare the descriptions found in SQ1 with those found in SQ3. Which are in both sets?

APPLY-PSSM-TO appears to have missed some of the training set. That's hardly likely, so let's search more deeply for them.

SQ5. Apply CONTEXT-OF to the first several genes of the list obtained in SQ1. Can you find the missing descriptions?

Why do some of the genes of the training set appear as ***adjacent*** to the genes found by `APPLY-PSSM-TO`? The critical clue lies in the arrows supplied by `CONTEXT-OF`. Leftward arrows mean that the gene reads right-to-left. Rightward arrows mean that the gene reads left-to-right.

SQ6. What general rule can you discern regarding the directions of genes from the training set found by `APPLY-PSSM-TO` and the directions of genes from the training set *adjacent* to genes found by `APPLY-PSSM-TO`? How do you explain this generality in biological terms?

You may realize now that we made a slight mistake in running `APPLY-PSSM-TO`. We shouldn't have scanned intergenic sequences. Rather, we should have scanned sequences *upstream* of genes.

SQ7. Rerun `APPLY-PSSM-TO` but replacing `(INTERGENIC-SEQUENCES-OF …)` with `(UPSTREAM-SEQUENCES-OF …)`. Then get the descriptions of the genes found. Can you identify all the genes of the training set?

Notice that **`APPLY-PSSM-TO`** tells you whether the putative NtcA-binding site is on the forward strand or the backward strand.

SQ8. In the case of the genes of the training set, are the NtcA-binding sites on the forward or backward strand?

Since by convention, the strand that's displayed is written 5' to 3', left-to-right, and since known NtcA-binding sites lie upstream (to the left) of genes, we'd expect all the NtcA-binding sites to be on the forward strand.

SQ9. Do you see why the proven NtcA-binding sites should be found on the forward strand?

SQ10. By extension, you might think that *all* NtcA-binding sites should be on the forward strand, upstream of regulated genes. Yet `APPLY-PSSM-TO` returns some putative sites identified on the backward strand. Can you explain why this is?

Presuming that unexplained putative binding-sits on backward strands are just wrong, then `APPLY-PSSM-TO` is returning a significant number of false positives. This might be expected, considering that the protein probably binds only to a minority of the nucleotides found in the sequences of the training set. Perhaps if we were more selective in which positions we used in the PSSM, the results might be improved. But which positions to choose? At one extreme, we could choose only those positions where there is 100% agreement amongst the sequences of the training set. Alternatively, we could accept those positions, plus other positions that had good, if not perfect, agreement. How to capture the concept of "good" agreement?

What we want is a measure of ***information***. We have the most information when all sequences agree at a particular position. Then we can say with the most assurance that other sequences will probably also agree. We have the least information when all sequences are random at a particular position. Then we have no basis for predicting other sequences. The opposite concept to information is ***uncertainty***. We have the most uncertainty when the sequences at a particular position are random.

Uncertainty can be quantified by considering how many yes-no questions must be answered to determine the value at a position. In the case of nucleotides, you might think that you could need as many as four questions: Are you A? Are you C? Are you G? Are you T? Well, not four… three, since the answer is known by the time you've excluded A, C, and G. But even three questions is wasteful. You can get the answer in just two questions:

Are you A or C?
    If yes, then are you A?
    If no, then are you G?

We say that the uncertainty is 2.

SQ11. How many questions would you need to ask to determine which of 64 teams won the NCAA tournament?

You may have answered SQ11 by running through the process, cutting 64 in half, then half again, and so forth. You can get the same answer by asking:

For what n does $2^n = 64$

or $n = \log_2(64)$

Satisfy yourself that $\log_2$ of some power of 2 is the number of times you need to cut it in half before reaching 1.

The situation is often much better than I've indicated. Suppose that you've examined 100 DNA sequences and 99% of the time the first position is A. Looking at the 101st sequence, how many questions would you need to ask. You could do it the same way as before and guarantee an

answer in two questions, or you could save time by asking straightaway, "Are you A?". 99% of the time you'll get the answer in 1 question. You can go the slow route in the 1% of the remaining cases. So the average number of questions in this case will be close to 1.

Uncertainty is *related* to the number of questions asked, but it is not quite the same thing. It is defined as the average (- $\log_2$(probability of a state)) for all the states considered. In the case of nucleotides, you might think that the uncertainty is:

Average of (- $\log_2$(probability of A) = - $\log_2(1/4)$ = 2
- $\log_2$(probability of C) = - $\log_2(1/4)$ = 2
- $\log_2$(probability of G) = - $\log_2(1/4)$ = 2
- $\log_2$(probability of T) = - $\log_2(1/4)$ = 2)

and the average of four 2's is, of course, 2, similar to the answer we got before – two questions required. But what if the probabilities of the four nucleotides are NOT the same? Suppose that from your alignment, you find that at a certain position, you have 7 A's, 1 C, 1 G, and 1 T. Now the uncertainty is:

Average of (- $\log_2$(probability of A) = - $\log_2(7/10)$ = 2
- $\log_2$(probability of C) = - $\log_2(1/10)$ = 2
- $\log_2$(probability of G) = - $\log_2(1/10)$ = 2
- $\log_2$(probability of T) = - $\log_2(1/10)$ = 2)

SQ12. What is the uncertainty in this case? [Don't have a calculator that can handle $\log_2$? YES YOU DO! In BioBIKE, (`LOG2` …) gives you what you want.]

If you got 2.62, then something is wrong. How can your uncertainty be GREATER than the case where you have no bias (all frequencies are the same)? Surely you have more information by knowing that the position is grossly skewed towards A. The problem is that you didn't take the average properly. There were seven A's with a probability of 7/10, so the average should be weighted 7:1:1:1. Multiplying $\log_2(7/10)$ by 7 and taking the average you should get the correct answer of 1.36.

A more efficient way of taking this average is to divide through by the sum as you add:

Average = - [7 * $\log_2(7/10)$ + 1 * $\log_2(1/10)$ + 1 * $\log_2(1/10)$ + 1 * $\log_2(1/10)$] / 10

= -[7/10 * $\log_2(7/10)$ + 1/10 * $\log_2(1/10)$ + 1/10 * $\log_2(1/10)$ + 1/10 * $\log_2(1/10)$]

or in general:

Average = - [p_1 * $\log_2(p_1)$ + p_2 * $\log_2(p_2)$ + p_3 * $\log_2(p_3)$ + p_4 * $\log_2(p_4)$]

= - SUM[p_i * $\log_2(p_i)$]

This is the definition of uncertainty.

SQ13. Calculate the uncertainty where the counts of A, C, G, and T are {5 5 0 0}. Note that the 0*$\log_2$(0) is taken to be 0.

Information is derived from uncertainty:

Information = (maximal uncertainty) - uncertainty

The maximal uncertainty is the uncertainty in the absence of any bias or other information.

SQ14. What is the maximal uncertainty for a position in a nucleotide sequence?

SQ15. What is the information at a position where the counts of A, C, G, and T are equal? Or the counts are as in SQ13?

You can graphically represent information as a histogram, where the height of the bar at each position is the information. A common representation is a sequence Logo, which is just such a histogram, but bars are replaced by letters of different heights, the heights proportional to the frequency of the letter. The sum of the heights is the information for that position. Here is a web site that produces sequence Logos:

http://weblogo.berkeley.edu/

SQ16. Produce a sequence Logo based on the aligned NtcA binding sites for *Synechocystis* PCC 6803.

You can also get information in quantitative form through BioBIKE, using the command:

`(INFORMATION-OF` *set-of-aligned-sequences*`)`

SQ17. Produce a list of information values for the aligned NtcA binding sites for *Synechocystis* PCC 6803. Compare the list with the Logo. Do you see a reasonable correspondence?

Back to PSSMs (again). Recall that our problem was confining the PSSM to those positions that we judge to be sufficiently informational. Suppose you judge that only positions with information content greater than 1.0 is worth looking at and the other positions just add to the noise. You can limit the PSSM to such positions in the following way:

```
(APPLY-PSSM-TO ... WITH-PSSM-FROM ... INFO-THRESHOLD 1.0)
```

SQ18. Rerun the PSSM scan of *Synechocystis* considering only those columns with information content of 1.0. Are there fewer high-scoring putative binding sites on backward strands?