BNFO 301: Introduction to Bioinformatics
Problem Set: Evolution and Genome Identification

1. (Feel free to contact me if you have trouble with this) Certain cyanobacteria are clearly related to each other. For example, the three species of *Prochlorococcus* known to BioLingua share obvious morphological and biochemical characteristics. The three cyanobactera (*Anabaena* PCC 7120, *Anabaena variabilis*, and *Nostoc punctiforme*) able to make N_2-fixing heterocysts share a wide range of similarities. You are interested in a cyanobacterium, *Trichodesmium erythreum*, that seems to fall between the cracks. Like the *Prochlorococcus* strains, *Synechococcus* WH8102, and *Crococcocus watsonii*, *Trichodesmium* lives in the ocean. Like *Anabaena* and *Nostoc*, it is filamentous (though *Trichodesmium* does not make heterocysts). Like *Crocosphaera*, it fixes nitrogen without heterocysts. Which amongst the cyanobacteria are *Trichodesmium's* closest relatives?

   You decide to explore this question using the techniques that Shinazawa's group used to explore the origins of nuclear genes. Recall that they defined the evolutionary distance between two sets of genes in terms of the number of proteins of one set that are similar to proteins of another, where "similar" is defined by an E-value threshold. If you make a set all proteins in a given cyanobacterium that have orthologs in all other cyanobacteria, you should be able to apply their method directly to your problem. Accordingly, you will:

   a. Identify all proteins in *Trichodesmium* that have orthologs in all other available cyanobacteria. The following BioLingua functions may be useful:
      ```
      (COMMON-ORTHOLOGS-OF *loaded-organisms* PRIMARY Tery)
      ```
      ; *Returns all orthologs common to all organisms known to BioLingua*
      ; *in terms of Trichodesmium (nickname Tery)*

      `(ORTHOLOG-OF` *protein* `IN` *organism*`)`
      ; *Returns protein in* organism *that is the ortholog of the given protein*
      ; *Useful for checking the results of* `COMMON-ORTHOLOGS-OF`
      ; *If you take any of the proteins returned by* `ORTHOLOG-OF` *and run it*
      ; *through* `ORTHOLOG-OF` *(using any cyanobacterium), you should get*
      ; *a non-NIL answer, or else something is wrong.*

   b. Use `DEFINE` *variable* (or `ASSIGN` *variable*) to give a name to the set of common orthologs you found in **Step a**.

   c. Shuffle the set of common orthologs,[*] i.e. rearrange the list so that the order of elements is random. To do that, use the SHUFFLE function. Here's an example of how it works:
      ```
      (DEFINE number-list AS (12 24 35 46 57))
      (DEFINE shuffled-list AS (SHUFFLE number-list))
      ```

   d. Extract the first 100 proteins in the shuffled list (using FIRST-N) and give it a name (using `ASSIGN` or `DEFINE`). Here's `FIRST-N` in action:
      ```
      (FIRST-N 3 shuffled-list)
      ```

[*] This is done so that everyone works with a different sample of the entire set of orthologs. It would be too time-consuming for everyone to work with the entire set. Also, we get to play with the statistics of sampling!

e. Just for practice, find the E-value for the ***first*** protein in your shuffled list blasted against its ortholog in A7120. To do this:

   – Assign a variable to be equal to the `FIRST` protein in the shuffled list. Here's `FIRST` in action:
   ```
   (FIRST number-list)
   ```

   – Assign a variable to be equal to the `PROTEIN-SIMILAR-TO` that protein `IN` A7120, asking it to return only the most similar protein. The keyword `RETURN` allows you specify that only one protein is returned, the most similar. Note that what is returned is a ***list***: (*protein e-value*).

   – Assign a variable to be equal to the `SECOND` element of that list, i.e. the E-value. `SECOND` works the same way as `FIRST`.

f. Make a loop that goes through each of the first 100 proteins in the shuffled list. In that loop transfer each of the operations you performed in **Step e**. You will need to make sure that those operations work within the context of a loop. You could do this in either of two ways:

   1. Change `(ASSIGN …)` to `AS` ….
   2. Put all of the `(ASSIGN …)` after the `DO` keyword.

   Note that this loop doesn't do anything useful yet. But it should at least not give an error message.

Sets an iteration variable equal to the ortholog of the protein in Assigns to some variable

g. `COUNT` all E-values generated by the loop. If you're counting all of them, the loop should return the total number of genes considered, i.e. 100.

h. `COUNT` only those E-values that are less than 10^{-90} (which in Lisp is, unfortunately, rendered as `1d-90`).[†] To do that, use the `WHEN` keyword, which governs the action of the following keyword. Here's `WHEN` in action:
   ```
   (FOR-EACH number IN shuffled-list
        WHEN (< number 30)
       COUNT number)
   ```

   Wrap around the loop you just made an outer loop that considers each known organism. COLLECT a list consisting of the name of the organism and the COUNT of the E-values less than 1e-40. For example:
   ```
   (FOR-EACH organism IN *loaded-organisms*
     COLLECT (LIST organism
                   (FOR-EACH number IN shuffled-list
                        WHEN (< number 30)
                       COUNT number)))
   ```

[†] Lisp renders scientific notation in two ways, using **e** (stands for exponent I suppose) for exponents of 10 less than around 44 and **d** (stands for double, as in double precision, I think) for any exponent. So `3.04e23` = $3.04 \cdot 10^{23}$. And `3.14159d-100` = $3.14159 \cdot 10^{-100}$. Unfortunately `3.14159e-100` is legal but as the exponent is too small to be captured by the **e** syntax, it is stored as `0`.

If all has gone well thus far, you should have gotten a result something like this:

```
((#$thermosynechococcus elongatus bp1 57)
 (#$crocosphaera watsonii wh8501 64)
 (#$gloeobacter violaceus pcc7421 47)
 . . .)
```

i. Save this file, using `WRITE-TAB-DELIMITED-FILE`. Here's the function in action:

   ```
   (WRITE-TAB-DELIMITED-FILE "filename.txt" *)
   ```

   This writes the previous result (whatever it is) to a file named `filename.txt`.

j. Download the file to your computer and e-mail it to me. To download the file, click on **Browse BioFiles**, find the file you just created, click on it, and download it, using whatever method your web browser supports.

k. So far, what organisms do you think are most similar to *Trichodesmium*? Least similar?

2. In a moment, you're going to take the data generated in Problem 1 and calculate the average number of E-values more significant than the threshold value, comparing orthologous proteins from *Trichodesmium* and other cyanobacteria. Before we do this, we need to build some tools. Here we will build a tool that calculates averages. You could do it this way very simply:

   ```
   (DEFINE-FUNCTION Average (list)
      (LET ( (length (LENGTH list))
             (sum (SUM-OF list)) )
        (/ sum length)))
   ```

   Translation:
   - *Define a function called* `Average` *that is given an argument we'll call* `list`
   - *Define the local variable* `length` *and give it the value of the length of the given list (Note: if Average is given anything but a list, the function will crash here)*
   - *Define the local variable* `sum` *and give it the value of the sum of the values in* `list`
   - *While the local variables are still in force (the parenthesis before* `LET` *has not been closed), calculate* `sum` *divided by* `length`*, which indeed is the average.*
   - *Return as the result of the function the result of the last form, i.e. the average*

   But that would be too simple. As an exercise, define a function called Average that calculates the average of numbers in a list using a loop. It will look like the function above, except for:

   ```
   (LET ...
         (sum (FOR-EACH x IN ...
                                 )) )
   ```

   Be sure to test your function, in something like the following way:

   ```
   (DEFINE number-list AS (1 3 5 4 2))
   (AVERAGE number-list)
   ```

3. In another moment, you're going to do a t-test on all the averages, just as Shinozawa's group did… which means ***more*** tools to build. To do a t-test, we're going to need to know how to

find the standard deviation of a list of numbers (see formula in Problem 4). A standard deviation of a list of numbers from x_i to x_N can be calculated as:

$$\text{Variance} = \Sigma\ (x_i - \overline{x})^2\ / (N-1)$$

$$\text{StdDev} = \sqrt{}\ \text{variance}$$

3a. Define a function called Variance that returns the variance of a list of numbers

3b. Define a function called StdDev that returns the standard deviation of a list of numbers

4. Last tool… t-test. The t-score can be calculated:

$$\text{Average variance} = [(N_1 - 1)\cdot \text{var}_1 + (N_2 - 1)\cdot \text{var}_2] / [(N_1 - 1) + (N_2 - 1)]$$

$$\text{t-score} = \frac{\overline{x}_1 - \overline{x}_2}{\lfloor \sqrt{}(\text{average variance})]\ \ [(N_1 - 1) + (N_2 - 1)] / (N_1\ N_2)}$$

   where N_1 and N_2 are the population sizes and var_1 and var_2 are the variances of each population

   Define a function t-score that returns the t-score for a t-test performed on two lists of numbers.

5. Now the main event (part 1). You have available (see calendar) a data set generated from responses to Problem 1, put in the following form:

   ((*name-of-organism score1 score2 score3 ...*)
    (*name-of-organism score1 score2 score3 ...*)
    …
    (*name-of-organism score1 score2 score3 ...*))

   where each score is a hit-score as defined in Shinozawa's paper (and in DGPB). Write a loop that goes through each element of the main list and calculates a mean hit-score for each organism. The output of the loop should look something like this:

```
name-of-organism1 average-of-first-set-of-hit-scores
name-of-organism1 average-of-first-set-of-hit-scores
...
```

   The following may be of interest to you:

```
(DEFINE animal-list AS ("mammals" "cat" "dog" "mouse" "lemur"))
(DEFINE animal-label AS (FIRST animal-list))
(DEFINE animals AS (REST animal-list))
```

   Which organisms appear to be closest to *Trichodesmium*? Which organisms appear to be the most distant?

6. Now the main event (part 2). The differences between the average hit-scores are sometimes not that big. But are they big enough to support a reasonable conclusion? Use t-tests to address that question. Use the t-score function you wrote to test specific questions (e.g. "Is the mean hit-score for *Trichodesmium* vs *Nostoc* significantly different from the mean hit-score for *Trichodesmium* vs *Anabaena* PCC 7120?"). You will run up against the issue: How

high a t-score must you have to declare significance? What does "significance" mean? We can operationally define "significance" in this way:

> Suppose that the two sets of hit-scores come from the same population (which would occur, for example, if *Nostoc* and *Anabaena* are really the same organism). If under these circumstances, a t-score as good as the one observed or better would have arisen *k* percent of the time, then I say that I am (1 – *k*) percent confident that my result did ***not*** arise from the same population. If I'm willing to accept that level of confidence, then I declare the difference significant.

You will be pleased to learn that there is an on-line calculator that can tell you what fraction of the time a t-score will arise by chance from one population. The calculator needs to know the sample sizes that produced each mean. This is reasonable. You'd think that there would be more variation in the mean of small sample, and the t-test thinks the same thing. The on-line calculator gets what it needs concerning sample sizes by having you enter the degrees of freedom, which is calculated as df = $(N_1 - 1) + (N_2 - 1)$. You can find the calculator at:

> http://psych.rice.edu/online_stat/chapter8/difference_means.html
> (scroll down about 30% of the way down the page)

What differences amongst hit scores are significant at the 95% confidence level?

7. We're going to make phylogenetic trees but… you guessed it! We need some tools. We need to calculate the phylogenetic distance between pairs of cyanobacteria, estimated as the genetic difference between pairs of cyanobacterial rRNA molecules. So we'll end up with the 16S-RNA-difference of Npun vs Tery, S7942 vs ss120, and so forth. The results will look like a table (see table at right). How do we store this kind of information? Storing it as a list is possible, but doing this would make it difficult to pull out specific cells. What I'd like to be able to do is to say "Tell me the number for Cwat Vs Avar". Not easy with a list.

	A7120	Avar	Cwat	. . .	Tery
A7120	382				
Avar	374	382			
Cwat	352	352	381		
. . .	. . .	. . .	. . .	. . .	
Tery	350	348	347	. . .	382

This is the time to introduce a basic tool to BioLingua: Tables. Once I've defined a table to represent what I want, then I can refer to specific elements by their labels. For example:

```
(VALUE-OF distance-table (Cwat Avar))
:: 352
```

For practice, let's use tables to do something simple: make a multiplication table:

```
(FOR-EACH x FROM 1 TO 10
     DO (FOR-EACH y FROM 1 TO 10
          DO (ASSIGN mult-table(x y) = (* x y))))
```

Translation:
- Consider each `x` *from 1 to 10 (think of them as the rows)*
- Within each row, consider each `y` *from 1 to 10 (think of them as columns)*
- Enter into the table `mult-table` *at position* `(x y)` *the product of* `x` *and* `y`

Those with programming experience will recognize this table as a two-dimensional array.‡ But these are arrays with a difference. You can refer to the element by number (as in Excel) or by any label you like. For example:

```
(FOR-EACH word IN (LIST "cat" "dog" "mouse" "lemur")
     AS len = (LENGTH word)
     AS letter1 = (GET-ELEMENT 1 FROM word)
     AS vowels = (SUM-OF (COUNT-OF ("A" "E" "I" "O" "U") IN word))
     DO (ASSIGN properties (word length) = len)
        (ASSIGN properties (word 1st-letter) = letter1)
        (ASSIGN properties (word vowel-count) = vowels))
```

then (for example):

```
(VALUE-OF properties ("mouse" vowel-count))
```

To try it out, let's revisit an old problem. Consider the nucleotides preceding the genes of S7942. Is there any pattern there? You (may have) found one way to answer the question. Here's another. Define a set of strings as the nucleotides –15 to +3 with respect to the beginnings of all genes of S7942. Then loop through each string, splitting up the string into characters, and make a table of positions (columns) going from 1 to 18 and nucleotides (rows) consisting of "A", "C", "G", and "T"), so that the final table gives the number of each nucleotide at each position. Your loop will look something like this:

```
(FOR-EACH ... IN ...
     DO (FOR-EACH letter IN ...
              FOR pos FROM 1 TO 18    ; This is a new feature!
              DO (INCREMENT . . .)))
```

There are two new things here to understand before attempting the final loop: `FOR` and `INCREMENT`. Try them out with simple tests!

‡ If the term means nothing to you, no problem. Let it go.