(2 Mar 2012)

Introduction to Bioinformatics
Problem Set 5: Blast and Dotplots

Blast

1. Why did the search for FMRP in *Drosophila* work well when human FMRP protein was used as the query but failed so abysmally when the corresponding human gene was used as the query? Now's the time to find out.

1a. Hearken back to the tour *Search for FMRP in Drosophila* and consider item 15, the comparison of the human and *Drosophila* proteins. The comparison begins with a stretch of very high amino acid similarity, from the initial M (methionine) up to YK (tyrosine-lysine). How many amino acids are in that stretch?

1b. How many nucleotides are required at the beginning of the gene to encode those amino acids (M through YK)?

Let's get those nucleotides, from the human gene and from the *Drosophila* gene. If the amino acid sequences are extremely similar, then so should be the nucleotide sequences, no? The protein sequence found by Blast comes with a link to the corresponding gene, with the GenBank ID of LD09557. You determined during the tour that the GenBank sequence does not begin with the FMR gene but includes upstream sequence. You'll recall that the gene begins at nucleotide 423.

1c. In BioBIKE (any instance will do), DEFINE a variable (maybe `fly-seq`) as the beginning of the fly gene, using SEQUENCE-OF and specifying the FROM-GENBANK option. The argument should be the GenBank ID (in quotes). Also specify values for FROM and TO, so that you bring in only those nucleotides from the beginning of the gene to the end of the portion of the gene encoding Y and K. To check if you were successful, get the TRANSLATION-OF `fly-seq` and compare it to the amino sequence you considered in **1a**.

1d. Retrace your steps in the tour and find the GenBank ID for the human FMR gene and what nucleotide coordinate the gene starts with. DEFINE a variable (maybe `human-seq`) as in 1c, but using the appropriate GenBank ID.

1e. Align these two DNA sequences, using the ALIGNMENT-OF function. The argument should be a LIST consisting of two items: `fly-seq` and `human-seq`.

1f. Does the alignment look as good as the amino acid alignment? Why not? Do something to test your theory.

2. The gene D29p32 from mycobacteriophage D29 presents an anomaly. Using PhAnToMe/BioBIKE (Ph/BB), find the description of this gene, perhaps using DESCRIPTION-OF or by going to the gene within the Sequence Viewer. You'll see that the gene is called an integrase, i.e. a protein responsible for the integration of the phage genome into the host genome to initiate lysogeny. But then find the DESCRIPTION-OF D29. There's way too much information, but fortunately, the identifiers are in alphabetical order. Look for the lifestyle of the phage… it's described as lytic! What is a lytic phage doing with an integrase? Maybe the description of the gene is in error? Or maybe the description of the phage?

2a. Is there similarity between D29p32 and perhaps known integrases? Check that by blasting the sequence against all known protein. You could do this at the NCBI site as you have in the past, but why not enjoy the one-stop-shopping afforded by BioBIKE? In Ph/BB, use SEQUENCE-SIMILAR-TO to access Blast, giving the protein of D29p32 (remember the p- convention) as the query and *GENBANK* (obtainable from the DATA menu) as the target. This tells Ph/BB to ask NCBI to perform the Blast search using all proteins contained in the GenBank database. Having absorbed the lesson of the previous problem, you'll undoubtedly use the sequence of the protein, not the gene. Specify PROTEIN-VS-PROTEIN as the type of sequence comparison, and execute the function.

In a few seconds you should get the results. Note the identity of the best hit (note also that NCBI's name of the gene differs from that in BioBIKE… I hope you're getting used to this!). The other hits are mostly annotated as integrases. One is of particular interest. You'll soon read an article regarding the genes of mycobacteriophage Che12, so keep an eye on its genes. Note the E-value for the match between D29p32 and the gene from Che12. Write it down.

2b. You would think that most of the search performed by NCBI in **2a** was wasted effort. How likely is it that most of the GenBank database (mostly eukaryotic sequences) would have any chance of containing a phage integrase? Let's try it again with a more reasonable target, just phage sequences. Repeat the execution of SEQUENCE-SIMILAR-TO, but this time using as the target `all-phage` (a set that includes all phages known to Ph/BB).[*]

The format of the results may change, but the message is about the same, no? Why does the first Blast but not the second show hits to FRAT1 and Eagle? Again, note the best hit and also jot down the E-value for the match to the protein from Che12.

2c. Even the confined search of **2b** was largely wasted effort. The closest matches to a mycobacteriophage protein would figure to be other mycobacteriophage proteins. So re-execute SEQUENCE-SIMILAR-TO, this time with a target of only a subset of all phages, the mycobacteriophages. Recall that you created such a subset in the second Hatfull tour.

How do the results compare with those of **2b**? Now what is the E-value for the match against the Che12 protein?

2d. If we're so interested in Che12, then why not do a directed search just of that genome? Do a last execution of SEQUENCE-SIMILAR-TO, using Che12 as the target. What is the E-value for the match against the Che12 protein?

2e. Consider all the E-values you've noted for matches of p-D29p32 against the Che12 protein. Why aren't they all the same? Develop a hypothesis that ***quantitatively*** accounts for the differences in E-values you observed and test that hypothesis.

[*] Ph/BB attempts to speed up the execution of SEQUENCE-SIMILAR-TO by pre-running every possible Blast of proteins it knows about against all other proteins it knows about. Then when called upon to do the blast, it simply looks up the results in a massive table. Fast, yes, but not always reliable. If you get an error message complaining that a file doesn't exist (probably an element of the lookup table), then bypass this feature, using the BYPASS-LOOKUP option. Then Ph/BB will perform a conventional Blast.

3. Execute SEQUENCE-SIMILAR-TO D29p32 IN Che12 using all five flavors of Blast (DNA-VS-DNA, etc), one at a time. Also use the BYPASS-LOOKUP option (see footnote on previous page), as doing so will ensure a more fair comparison.

3a. One of the five searches gives an E-value much worse than the others. Which one and why?

3b. One of the five searches gives far more hits than the others. Why?

3c. The search you just identified in **3b** has 6 hits that are far better than the rest. Why? Why is the best hit in that search so much better than hits 2 through 6?

4. Synechococcus WH7805 is a strain of marine cyanobacterium which with its relatives contributes significantly to the total CO_2 fixed in the world's oceans. Predict what its ribosome-binding site looks like. Hint: Never lose sight of the fact that it is alive!

Dotplots

4. Make your own DOTPLOT function

The strategy will be to slide a window across one sequence and compare that window against the second sequence. The matches for each window will provide information for one line of the dotplot, using the PLOT function to display the dotplot.

PLOT requires a set of pairs of numbers, e.g. ((1 1) (2 4) (3 9))

4a. Play with the PLOT function, providing it with a set of different number pairs, until you understand what it does with the pairs.

As a test of the function (and to give us raw material as we build it), we'll try it on the sequences of two small phage: Chlamydia phage 3 and Chlamydia phage phicpg1 (find the sequences in either Phantome/BioBIKE or ViroBIKE).

Your implementation of DOTPLOT will compare two sequences through a window of 11 nucleotides, grabbing 11 nucleotides at a time from the first sequence and searching the second sequence for matches to it. As you go through each step below, notice the result in the result pane and compare it with what you expect from the actual sequence.

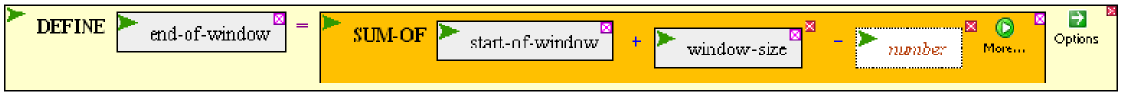
4b. DEFINE `seq1` as the SEQUENCE-OF Chlamydia phage 3 and `seq2` as the SEQUENCE-OF Chlamydia phage phicpg1.[†]

4c. DEFINE `window-size` as 11 (each comparison will be 11 nucleotides)

4d. DEFINE `start-of-window` as 1 (we'll start the first window at coordinate 1

4e. DEFINE `end-of-window` as… as what? If `start-of-window` is the first coordinate of the window, what is the last? Knowing the start-of-window and the window-size, how would you calculate the end coordinate? Keep in mind that the values of these variables may change. Use SUM-OF to do the calculation. The function will look something like this:

[†] Of course the names of the first phage will not be literally "Chlamydia phage 3". That's just the human name for the phage. Use the usual tricks to find the name of the gene as known to BioBIKE. By the way, NAME-OF [organism] SHORT will give you a short nickname,.



```
DEFINE end-of-window = SUM-OF start-of-window + window-size - number
```

What value should go in the *number* box? Check this by counting!

4f. Now use the values you've defined to define the portion of the first sequence you'll try to match with the second sequence. DEFINE `window` as SUBSTRING `seq1` FROM `start-of-window` TO `end-of-window`

4g. DEFINE `matches` as MATCHES-OF-ITEM `window` IN `seq2`. It will be more convenient if the function returns only the position of the match, so specify the POSITION-ONLY option. Try it with and without the option so you can see what I mean.

4h. Now we need to turn this information into number pairs. The first number in each pair will be the coordinate of the window (for simplicity, we'll use start-of-window). The second number of each pair will be the coordinate of a match. A pair will be formed by:

LIST start-of-window n

where n is one of the coordinates returned by MATCHES-OF. We don't know how many matches will be found, but we can MAP the LIST function over the list of matches:

```
APPLY-FUNCTION
    LIST start-of-window  n
    REPLACING n WITH matches
```

DEFINE number-pairs as this function (i.e., APPLY-FUNCTION …)

If all has gone well, you should have the number pairs necessary for PLOT to display one line of a dotplot of seq1 vs seq2. To generalize, i.e. repeat these operations over all windows within seq1, we need a loop.

4i. Bring down a FOR-EACH loop. We'll treat seq1, seq2, and window-size as constants for the moment. The loop will presume they already exist. The loop will repeat lines 4E through 4H and loop through all values of start-of-window, FROM . Open up the body of the FOR-EACH loop and create holes for each function made in lines 4E. through 4H. Then drag each of those four functions (in order) into one of the four holes.

4j. Open up a primary control *number from n1 to n2* box. Replace *var*iable with start-of-window. The first value should be 1 of course, so that the window begins at the beginning of seq2. What about the last value? Surely it's related to the length of seq2. Is it equal to the length? Devise a calculation for the last value and put it in the last value box. If you don't know how to calculate the number, make up a simpler case, say a sequence composed of just 11 letters.

4k. Open up a Results Section, using WHEN… APPEND. Why WHEN? You don't always want to add to the results, just when you found a match. Why APPEND? The difference between APPEND and COLLECT is subtle. We'll explore it later. For now, trust me.

4l. WHEN what? WHEN matches has a value. That's expressed simply as WHEN matches. Put the variable matches in the condition box.

4m. APPEND what? APPEND all the number-pairs you found. Put the variable number-pairs in the value box.

4n. Execute the loop. Do you get the expected collection of number pairs?

4o. Put the loop within a PLOT function. The easiest way to do this is to mouse over the action icon of FOR-EACH, click Surround, and then click PLOT in either the INPUT-OUTPUT menu or the ALL menu. The PLOT function by default plots lines, but you're doing a DOT plot, so choose the POINTS option from the option menu of PLOT. Execute the resulting function.

4p. How do you interpret the dotplot?

5. Now that you have a loop that works, we want to package it into a function, something that you can use like any other BioBIKE function

5a. Collapse the now huge PLOT function, by mousing over its action icon and clicking Collapse. Then go the same menu, click **Name me**, and give it a descriptive name (e.g. Plot Dotplot).

5b. Bring down the DEFINE-FUNCTION function from the DEFINITION menu.

5c. Drag Plot Dotplot into the body of the function.

5d. This function should be given two arguments, seq1 and seq2. Create a second *arg* box using the *More* icon. Then type seq1 in the first box and seq2 in the second

5e. Allow specification of the value of window-size. Do this by creating a WINDOW-SIZE option for this function. To do this, open the SPECIFICATION section, click Reveal keyword arguments, mouse over the keywords icon, and click keys. Then enter window-size into the *var*iable box and provide a default value, perhaps 11.

5f. Finally, give the function a name. MY-DOTPLOT might work for you. Then execute the DEFINE-FUNCTION. This will generate a lot of warnings, but don't worry about them.[‡] You should see a FUNCTION box appear in the palette, and your new function will be represented on its menu.

5g. Test your new function, bringing it into the workspace from the FUNCTION menu and providing it with various input values and… well, first it's probably best to first try it out with your original sequences, the two from Chlamydia.

6. What proteins are large and why?

6a. Display a list of the largest proteins in *Anabaena variabilis* ATCC 29413. Include in the list the length, name, and description of the proteins. You might be interested to know that the SORT function can sort in either ascending or descending order.

6b. Notice that three is a kind of protein that appears multiple times in the top ten. Why is it so large? Try running a dotplot of a the amino acid sequence of large protein against itself. What do you see? What does it mean?

[‡] For the computer cognoscenti amongst you, what BioBIKE is complaining about is the bad programming practice of assigning global variables within a loop. If you had made those assignments to local variables (in the Variable Update Section, as described in the notes on looping), BioBIKE would have been satisfied.