(20 April 2015)

MIC653: Advanced Molecular Genetics Bioinformatics & Computational Genomics
Projects: Computation to Solve Problems

I am offering six projects that guide you through a project requiring computer programming. Choose whichever one (or more) that strikes your fancy:

I. **What determines the beginning of a gene?**
This is a reasonable choice for those with no experience with BioBIKE or programming, unless one of the other topics particularly attracts you.

II. **Where in a bacterial genome are viruses integrated?**
An example of data mining that combines a search of text with a search of protein sequences.

III. **Determination of short tandem repeats (STRs)**
STRs are commonly used in forensic application.

IV. **Analysis of gene expression data**
Extracting useful insights from microarray data. How to tweak the data so that you can compare the results of different replicates and different experiments?

V. **CRISPRs in enteric bacteria**
How to find them and how to compare their locations in different genomes.

VI. **Finding targets for DNA-binding proteins**
Using position-specific scoring matrices to extend experimental knowledge of the genomic targets of a certain DNA-binding protein to find new, previously unknown sites.

To make it possible to make substantial progress within the time available, all the projects make use of the on-line programming environment, BioBIKE. There are several instances of BioBIKE, each resting on a specific collection of genomes. You'll make use of CyanoBIKE (devoted to cyanobacterial genomes) and/or Phantome/BioBIKE (devoted to phage and bacterial genomes). You can get to both instances through the BioBIKE portal.

I. What determines the beginning of a gene? (uses CyanoBIKE)

Well, this sounds like it's going to be real short… **The ATG start codon determines the beginning of a gene!** End of story, right? Maybe not so right. Rather than place your faith in what the textbooks say, try investigating the matter yourself, through computational experiments with your own fingers.

To do this, you'll need to bring up CyanoBIKE and a tour called *What is a Gene?* You'll probably want to precede this with a companion tour that leads you gently through the conventions of BioBIKE.

II. Where in a bacterial genome are viruses integrated?
(uses either CyanoBIKE or Phantome/BioBIKE)

Rationale

Suppose you are interested in the interaction of bacteria and their phages. You realize that many phage are quite cozy with bacteria, integrating themselves into a bacterial genome for long periods of time. In this way they're like HIV and many other viruses. When a phage is integrated they're called prophages, and it can be difficult to distinguish their DNA from that of the native host, but this is the task you've set for yourself.

To learn how to do identify prophages, it would help to have on hand a set of known prophages on which you can try out your methods. Several prophages in E. coli are known, so your thought is to go there.

Throughout this exercise on pattern-matching you may find two sources of help useful, both accessible by putting the word pattern into the **Help** box and pressing Enter. From the list that results, you'll be able to access a page called BioBIKE Pattern Matching and also the help page for MATCHES-OF-PATTERN. The in-class presentation may also be of some help. If you haven't used BioBIKE before, it might be a good idea to invest some time learning about how it works. You can do this through a short on-line tutorial available here or through the portal.[*]

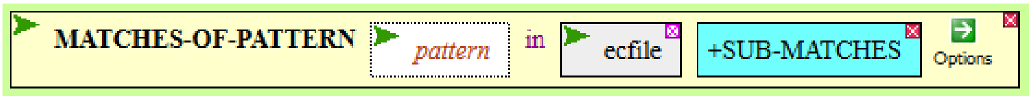
II.A. Search a Genbank file containing the annotation of *E. coli* for a specific text pattern

1. You could do this by going to NCBI nucleotide search, finding the entry for *Escherichia coli str. K-12 substr. MG1655*, downloading the file, and then uploading it into BioBIKE, but I've saved you the trouble. The annotation is available to you in a variable called `ecfile`.[†] You can also scroll through the same file by mousing over the black **File** button and clicking **Shared files**. Then find the file called "Escherichia coli-K-12-MG1655…" and click its name. Take a look at it. You'll need it for the rest of this question.

2. Search through the file until you find an annotation of a prophage (Ctrl-F is your friend). Don't pay much attention to gene-specific entries (unfortunately the great majority) but rather find features that describe the coordinates of the prophage as a whole. As you do this, try to reduce the tedium by thinking of a search strategy that will avoid the gene-specific entries and go straight to the lines you want, something that gives you the coordinates of the entire prophage region.

3. Figure out a pattern that will capture the beginning and end coordinates of each prophage along with the description of it (in quotes). Write out the pattern, including the quotation marks.

4. Use MATCHES-OF-PATTERN to extract all prophage information from the *E. coli* annotation. If you write the pattern artfully, capturing just the part of the text that you want, then you can use the +SUB-MATCHES option to create a much more satisfying output. The function might look something like this:

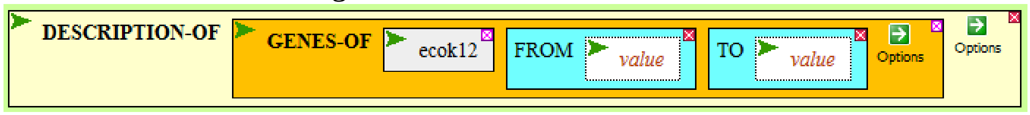


(of course fill in the pattern). What output did you get?

5. Use the coordinates you obtained in II.A.4 to display a list of genes contained in a few prophages. The GENES-OF function will give you the names of the genes within a given range of coordinates, and the DESCRIPTION-OF will give you their annotations. Your

[*] Once at the portal (http://biobike.csbc.vcu.edu), click the link to the *guided tours*, and choose the tour called *BioBIKE syntax and conventions.*
[†] If you execute a function calling for `ecfile` and get the error message "`PROBLEM: I don't understand what you mean by 'ECFILE'`", then see the addendum at the end of this problem set.

function will look something like this:


DESCRIPTION-OF GENES-OF ecok12 FROM *value* TO *value* Options Options

What output did you get, and what generalities can you glean from it?

II.B. Search a integrated phages by the presence of a gene encoding an integrase

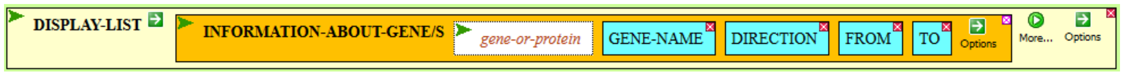
Another way to look for prophages is to look for regions possessing genes typical phages. One such gene is that which encodes an integrase, an enzyme responsible for the process of integration. Unfortunately, the amino acid sequences of integrases are highly variable, and so searching for them via Blast is often not a winning strategy. While the overall sequence is not well conserved, there are specific amino acid residues that are found almost universally in integrases [Nunes-Düby et al (1998). Nucl Acids Res 26:391-406]. Almost all integrases have the following characteristics:

- A histidine (H) residue 300-400 amino acids into the protein
- An arginine (R) three amino acids later
- Usually a histidine (H) 21 to 26 amino acids away from the arginine. Occasionally the histidine is replaced by an arginine (R) or tryptophan (W).
- A tyrosine (Y) 8 or 9 amino acids away from the previous histidine.

1. Use this observation to identify proteins in the same strain of E. coli (nicknamed ecok12) that are likely to be integrases. Do this by creating a pattern that represents the four characteristics, capturing the entire H..R..H..Y region (but not what comes before) and use it within MATCHES-OF-PATTERN (using the +SUB-MATCHES option). For the *target*, use PROTEINS-OF Ecok12 (so that the function searches protein sequences and not the *E. coli* genome). What proteins did you find? Does the sequence found match the pattern you provided?

2. Are these proteins annotated as integrases? To answer this question, repeat MATCHES-OF-PATTERN, this time with the following options: -MATCHES and -COORDINATES. This suppresses the display of sequences and coordinates so that the only thing returned are the names of the genes. Then use DESCRIPTION-OF on the list of proteins to display the annotations. What proteins did you find? How selective was the pattern?

3. Does identification of prophage regions by annotation (II.A) give similar results as their identification by the presence of integrases? Compare the prophage coordinates you obtained in II.A.4 with coordinates of the integrases you can get by the following function:


DISPLAY-LIST INFORMATION-ABOUT-GENE/S *gene-or-protein* GENE-NAME DIRECTION FROM TO Options More... Options

This function displays a list of the name of each gene, the left coordinate (FROM), right coordinate (TO), and the direction of the gene (F: left-to-right or B: right-to-left). How do the coordinates of the integrases correspond to the coordinates of the prophages?

III. Determination of short tandem repeats (uses either CyanoBIKE or Phantome/BioBIKE)

Short tandem repeats (STRs) are regions of DNA consisting of a unit 3 to 6 nucleotides in length repeated multiple times, for example AGGTAGGTAGGTAGGT…. Since the number of repeating unit mutates frequently, relative to other DNA changes, they are commonly used as a source of variation to identify individuals. They are also important as the causative agent of some genetically determined diseases. We'll look at STRs through both lenses.

First, however, if you haven't used BioBIKE before, it might be a good idea to invest some time learning about how it works. You can do this through a short on-line tutorial available here or through the portal.*

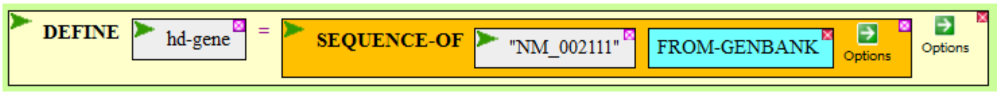
III.A. Recognizing a specific STR

1. The function MATCHES-OF-PATTERN can be used to recognize STRs. To try it out, use this function to find the full extent of the repeated AGGT unit in:

   TCCTTCAGGTAGGTAGGTAGGTCCGCCA

   Bring down MATCHES-OF-PATTERN from the **All** alphabetical menu. Copy/paste the above sequence into the *target* box, putting it between " " to identify it as a literal string rather than a variable name. In the *pattern* box, enter a string (between " "). That string should consist of the repeated unit as a group contained within ( ) repeated an indefinite number of times. Thus, the group would be (AGGT). See the list of BioBIKE Pattern Matching symbols to find the symbol for indefinite repetition. What pattern did you use? What result did you get?

2. Huntington's disease is caused by an excess of repeats of CAG (encoding glutamine) in the *HUNTINGTIN* gene. Individuals with fewer than 28 repeated units have a normal phenotype, while those with greater than 36 express some symptoms of Huntington's disease. Obtain the sequence of the gene from some individual using the SEQUENCE-OF function with the FROM-GENBANK option, as shown below:

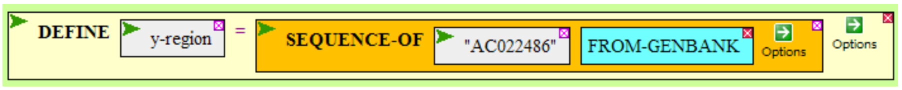
   
   ```
   DEFINE hd-gene = SEQUENCE-OF "NM_002111" FROM-GENBANK Options Options
   ```

   Use MATCHES-OF-PATTERN to identify the full extent of the CAG repeat in `hd-gene`. Finally, use the coordinates reported by this function to find the CAG repeat in the sequence of the gene, displayed with SEQUENCE-OF `hd-gene`. What do you predict is the phenotype of the person carrying this gene, what evidence do you have for that prediction, and what code did you use to find it?

III.B. Identifying unknown STRs

You have sequenced a portion of the Y-chromosome from an individual and want to determine if there are any STRs in the segment that might be useful in distinguishing DNA from different males.

1. Obtain the sequence from GenBank by executing the function shown below:

   
   ```
   DEFINE y-region = SEQUENCE-OF "AC022486" FROM-GENBANK Options Options
   ```

* Once at the portal (http://biobike.csbc.vcu.edu), click the link to the *guided tours*, and choose the tour called *BioBIKE syntax and conventions.*

2. Use MATCHES-OF-PATTERN to find all instances of 3 to 6 undetermined characters repeated at least 6 times. To specify a repeat of a previously captured group, you need to resort to a new symbol: `` `n ``, where n is a number. `1 refers to the first group captured, `2 refers to the second, and so forth. So the pattern "(Wa*...)-`1" would match "Walla-Walla", since "(Wa*...)" would capture "Walla" and be named group 1. What is the longest STR and what code did you use to find it?

IV. Analysis of gene expression data (uses CyanoBIKE)

About 20 years ago microarrays were introduced to measure simultaneously the expression of thousands of genes. With the drastic decrease in cost of sequencing, RNAseq has largely supplanted microarrays as a tool to measure gene expression. Despite the shift in methodology, certain problems remain, for example the issue reproducibility and the problem of how to compare different experiments performed under different conditions. There are many programs available to help researchers work through these problems and analyze gene expression data. However, what can you do if you want to analyze the data in a way that was not anticipated by those who wrote those programs. For that, you need to be able to program the computer yourself.

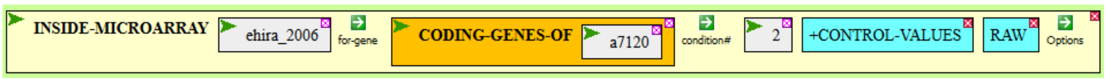
IV.A. Introduction to computational analysis of microarray data in BioBIKE

A tour entitled Combining pathways with microarrays[*] takes you through a specific problem of gene expression analysis, using BioBIKE functions. Go through this tour, replicating each step within CyanoBIKE.

IV.B. How to compare the results of different microarray experiments?

The main purpose of measuring gene expression is usually to determine a ratio of some sort, the change of expression at some time point or condition relative to another. It is also useful at times to know absolute levels of expression – perhaps expression went up by a factor of two, but did it rise from a high level to a higher level or a very low level to a still low level. Attempts to do this are bedeviled by the variability in expression measurements, which is only partially addressed by replicate measurements. While measurements of gene expression via RNAseq generally have less variability than measurements via microarrays, the issue still remains, and considering how it can be handled with microarrays may offer general insight into the problem.

1. Look at the raw expression data from the Ehira experiment, i.e. the fluorescence intensity measurements before statistical manipulation. To do this, bring down INSIDE-MICROARRAY, and specify ehira_2006 as the experiment, 2 as the condition, and CODING-GENES-OF A7120 for the gene (all the genes measured by the microarray). Choose the options RAW and +CONTROL-VALUES, getting the following:



INSIDE-MICROARRAY ehira_2006 for-gene CODING-GENES-OF a7120 condition# 2 +CONTROL-VALUES RAW Options

Each line gives you the six replicate measurements for one gene. Note the range of values from amongst replicates of one gene and amongst genes.

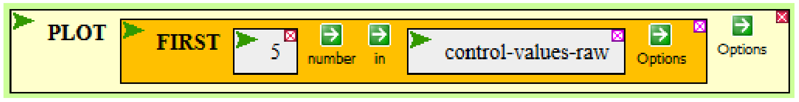
2. Of course there is variation from one replicate to the next, but Are the replicates systematically different from one another? To determine this DEFINE a variable (maybe call it *control-values-raw* and drag the above function into the *value* box. Before

[*] Available by clicking the link or by going to the BioBIKE portal (http://biobike.csbc.vcu.edu), clicking the link to the guided tours, and choosing the tour called *Combining Pathways with Microarray Data*.

executing, choose the –LABELS option in addition to the others. This will suppress listing of the names of the genes. You'll get just the numbers, which will simplify matters. Execute the function and verify that the result (in the Result Pane) has the same numbers as before.

3. Now to plot the distribution of values for each replicate. Right now, the numbers are grouped horizontally by gene. We want the same numbers grouped horizontally by replicate. You can swap vertical for horizontal by using the TRANSPOSE-LIST function. Bring that function down into the workspace and put *control-values-raw* into the *list* box. Execute the function, and examine the result to verify that the transposition has taken place.

4. The second step in plotting the distributions is to bring down the PLOT function into the workspace. Unfortunately, this function can display no more than five distributions at one time, so we'll use just the first five replicates. In the *list-or-table* box bring down the FIRST function, mouse over the *number* icon and click *number*, and enter 5 for *n*. Then drag into the *entity* box the TRANSPOSE-LIST function you made in Step 3. Finally execute the function, which should look something like what you see below:



5. This plot shows the raw expression value for each gene, from the first at the left to the 5336th at the right. That's not what we want. We want a distribution, where the X-axis is the expression value. To change the nature of the plot in this way, go back to the PLOT function and choose the BIN-INTERVAL. If you enter 1 as the BIN-INTERVAL *value*, then the function will count how many values lie between 0 and 1, 1 and 2, 2 and 3, etc. Execute the function…

6. Still not very helpful, because almost all of the values are scrunched near 1. To spread out the distributions, go back to the PLOT function and choose the options MAX, specifying 300 as the maximum value on the X axis. Now when you execute the function, you should see 5 distributions. Do you see them? Are they the same?

7. That was the raw values. Now repeat Steps 1 through 6, replacing RAW in Step 1 with NORMALIZED-BY-MEDIAN. What do you make of the plotted distributions?

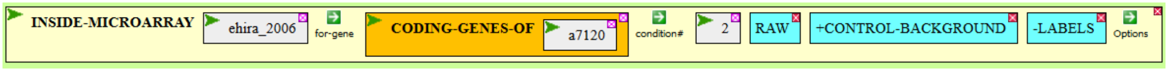
8. The replicates are now much more comparable to each other. How was this achieved? NORMALIZED-BY-MEDIAN achieves the compression in the following steps for each replicate:

   a. Subtract from each raw value the corresponding background value. Call this the *net expression*.

   b. Find the median value for the net expression

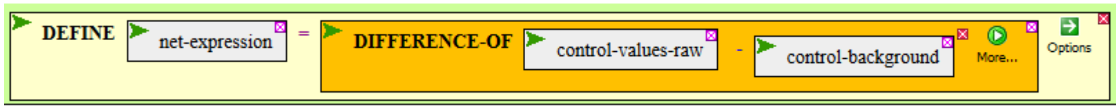
   c. Divide each net expression by the median value

   We can do this ourselves. First DEFINE *control-background* as the raw background values, obtained by INSIDE-MICROARRAY:



   Execute this function.

9. Now subtract the control background from the control values, using DIFFERENCE-OF, and call it *net-expression*:

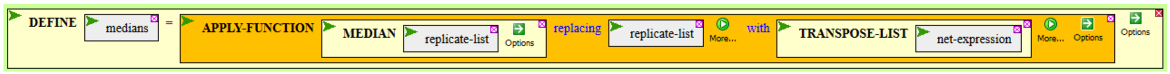


DEFINE net-expression = DIFFERENCE-OF control-values-raw - control-background More... Options

Check the first result in the Result Pane – Are the numbers indeed what is expected from the subtraction?

10. Calculate the median net expression using the MEDIAN function. Bring it into the work space and… problem! MEDIAN calculates the median of a simple list of numbers, but *net-expression* contains a list of lists, 5336 lists, each containing the values of the 6 replicates for the expression of a single gene. We need 6 lists, each containing the 5336 values for a single replicate. This part of the problem was already solved – see Step 3, and execute the function with *net-expression* as the *list*.

11. The second part of the problem is feeding MEDIAN the six lists one at a time. Learning how to do this sort of thing – iteration – is perhaps the biggest single hurdle in going from a non-programmer to a programmer. To do the iteration, construct the following function:



DEFINE medians = APPLY-FUNCTION MEDIAN replicate-list Options replacing replicate-list More... with TRANSPOSE-LIST net-expression More... Options Options

APPLY-FUNCTION gives MEDIAN the simple list it requires (called *replicate-list*), taken one at a time from the transposed *net-expression*. Executing this function should give you 6 medians, one for each replicate. Does it? Do the numbers make sense, from your observations of the numbers in *net-expression*?

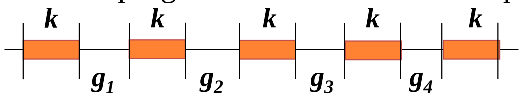
12. Almost there. Divide *net-expression* by medians, using QUOTIENT-OF. Execute the function and compare the first result with the first result of the normalized values (Step 7). How do they compare? Why?

If you've made it this far, then you have gone through the process of normalizing a microarray, a procedure required in gene expression analysis whether you use microarrays or RNAseq.

V. CRISPRs in enteric bacteria (uses Phantome/BioBIKE)

Clustered Regularly Interspaced Short Palindromic Repeats (CRISPRs) are widely used as tools to mediate targeted gene replacement. But for millions of years, long before they were bent to our technological needs, they have served as bacterial immune systems. Their natural purpose makes them well worth studying

There are many available programs to find CRISPR sequences, all making use in some way of their structure:



(where ***k*** is the constant repeat and g_n are the variable spacers) …but all they do is find CRISPR sequences. What if you want to do more? For example, what if you want to compare the characteristics of CRISPRs amongst related bacteria – their sequences, their associated CAS proteins, their genomic positions? What if you want to do an analysis for which there is no pre-made tool? Let's go down that road.

First, however, if you haven't used BioBIKE before, it might be a good idea to invest some time learning about how it works. You can do this through a short on-line tutorial available here or through the portal.[*]

V.A. Find a known CRISPR

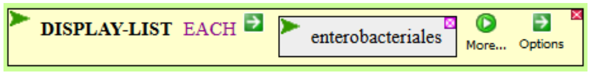
Escherichia coli strain K12 is known to possess a CRISPR with a 29-nt repeat. You could no doubt look up the sequence and use it to find the coordinates of the CRISPR, but let's take this as an opportunity to find the coordinates by a means that does not rely on prior knowledge of the sequence.

1. Go into Phantome/BioBIKE and bring down the MATCHES-OF-PATTERN function. From the structure of CRISPRs, devise a pattern that will match a CRISPR where ***k*** is 29 nucleotides, the spacer region is between 29 and 33 nucleotides, and there are at least three repeated units. What's the pattern?
2. Use the pattern to search E. coli (nicknamed *ecok12*). Verify that the sequence found is indeed a CRISPR. How many did you find? What are the repeated units? Are the units related to each other?

V.B. Find CRISPRs in related bacteria

E. coli is a gamma proteobacterium of the order Enterobacteriales. It might be interesting to compare CRISPRs in other bacteria of this order, determining whether they have the same repeated sequences and are located in the same genomic positions.

1. Examine what Enterobacteriales are available on Phantome/BioBIKE. To do this, mouse over the blue **ORGANISMS** button and click Bacteria. After a few seconds you'll be advised that the bacteria menus have been prepared. Mouse over the same button, then **Bacteria (phylo)**, then **Proteobacteria**, then **Gammaproteobacteria**, then **Enterobacteriales**, and finally click **All Enterobacteriales**. A set of bacteria will come into the workspace. DEFINE a variable (why not be creative and call it *enterobacteriales*?) by dragging the set of bacteria into the *value* box and executing the function. See what's in the set by:



DISPLAY-LIST EACH enterobacteriales More... Options

2. Look for CRISPRs in several (or perhaps all) of these bacteria by one or more of the following means:
   - Run the same MATCHES-OF-PATTERN over these other bacterial genomes (one at a time). Perhaps play with the pattern.
   - Use SEQUENCE-SIMILAR-TO, using the E.coli K-12 repeated unit as the query and the set of all enterobacteriales as the target (the object of the IN option).
   - Use SEQUENCE-SIMILAR-TO, using the E.coli K-12 repeated unit as the query, and choosing the MISMATCHES option (specifying some reasonable number of mismatches.

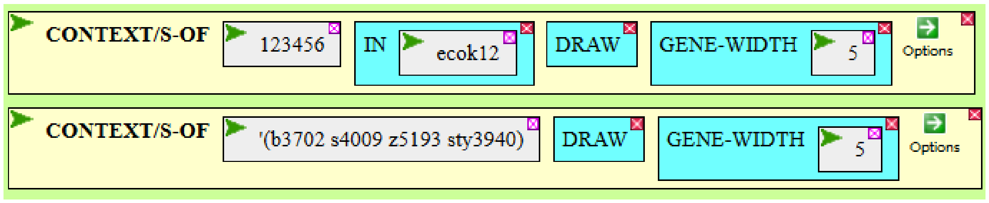
   What fraction of enterobacteria have CRISPRs like E.coli K-12's? What variation do you see in the sequence?

[*] Once at the portal (http://biobike.csbc.vcu.edu), click the link to the *guided tours*, and choose the tour called *BioBIKE syntax and conventions.*

V.C. Find the genetic context of enterobacterial CRISPRs

What genes are nearby the CRISPR repeats? You might expect that some number of CAS genes are present. How many? The same ones in each instance? It would also be interesting to see whether the genetic context of the CRISPR insertions are the same from organism to organism. You would expect it to vary if CRISPRs move around rapidly but not if they persist in the same spot over time since the enterobacteria diverged from one another.

1. Explore the utility of the CONTEXT-OF function in both of its forms:



```
CONTEXT/S-OF  123456  IN ecok12  DRAW  GENE-WIDTH 5  Options
CONTEXT/S-OF  '(b3702 s4009 z5193 sty3940)  DRAW  GENE-WIDTH 5  Options
```

   In the first form, you provide a coordinate or set of coordinates within parentheses. They could be, for example, the coordinates of CRISPRs found V.A and V.B. You also provide the organism (one at a time) as the object of the IN option. The DRAW option causes pretty graphical output to be produced, but it requires that you list the biobike URL in your whitelist of sites that can use Java. You can still get useful results without the DRAW option. The GENE-WIDTH option specifies how many genes on either side of the coordinate to list.

   In the second form, you provide one or more gene names. No need to provide the name of the organism, since BioBIKE gets this information from the gene.

2. Use CONTEXT-OF and perhaps other tools to determine what genes lie nearby CRISPR repeats. What generalities can you draw regarding the genes nearby the repeats?

VI. Finding targets for DNA-binding proteins (uses CyanoBIKE)

NtcA (for **nit**rogen **c**ontrol) is a DNA-binding protein that responds to the availability of nitrogen, e.g. in the form of ammonia, to differentially regulate the expression of many cyanobacterial proteins involved in nitrogen metabolism. In several cases, the DNA sequence to which NtcA binds has been determined in the laboratory (see table on the next page).

You happen to be interested in a cyanobacterium, Anabaena variabilis (nicknamed Avar) for which there is no laboratory evidence concerning the binding of NtcA. This is an instance where bioinformatics may come to the rescue!

VI.A. Predict binding sites for a regulatory protein in an organism with no experimental data

1. Use the sequences from this table to find possible NtcA-binding sites in Anabaena variabilis ATCC 29413 (nickname Avar). Investigate at least the first few matches to determine if the annotation is suggestive of a role in nitrogen metabolism. You'll be interested in the following functions:

   APPLY-PSSM-TO
   DESCRIPTION-OF

Strain	gene/operon	Promoter sequence
PCC 7942	*nir* operon	AAAGTT**GTA**GTTTCTGT**TAC**CAATTGCGAATCGAGAACTGCC..**TA**ATC**T**GCCGA**g**
	nirB-ntcB	TTTTTA**GTA**GCAATTGC**TAC**AAGCCTTGACTCTGAAGCCCGC..**T**TAGG**T**GGAGCCATT**a**
	ntcA	GAAAAA**GTA**GCAGTTGC**TAC**AAGCAGCAGCTAGGCTAGGCCG..**TA**CGG**T**AACG**a**
	glnB	TTGCT**GTA**GCAGTAAC**TAC**AACTGTGGTCTAGTCAGCGGTGT.**TA**CCAAAGAGT**c**
	glnA	TTTTAT**GTA**TCAGCTGT**TAC**AAAAGTGCCGTTTCGGGCTACC..**TA**GGA**T**GAAAG**c**
	amt1	CGAACT**GT**TACATCGAT**TAC**AAAACAACCTTGAGTCTCGCTG..A**A**TGC**T**TACAGAG**a**
PCC 7120	*glnA* (RNAI)	CGTTCT**GTA**ACAAAGAC**TAC**AAAACTGTCTAATGTTTAGAATC.**TA**CGA**T**ATTTC**a**
	nir operon	AATTTT**GTA**GCTACTTA**TAC**TATTTTACCTGAGATCCCGACA..**TA**ACC**T**TAGAAG**t**
	urt operon	AATTTA**GTA**TCAAAAATA**AC**AATTCAATGGTTAAATATCAAAC.**TA**ATA**T**CACAA**t**
	ntcB	AAAGCT**GTA**ACAAAATC**TAC**CAAATTGGGGAGCAAAATCAGC..**TA**ACT**T**AATTGA**a**
	devBCA	TCATTT**GTA**CAGTCTGT**TAC**CTTTACCTGAAACAGATGAATG..**TA**GAA**T**TTAT**a**
PCC 6803	*amt1*	TGAAAA**GTA**GTAAATCA**TAC**AGAAAACAATCATGTAAAAA....**T**TGAA**T**ACTCT**aa**
	glnA	AAAATG**GTA**GCGAAAAA**TAC**ATTTTCTAACTACTTGACTCTT..**TA**CGA**T**GGATAGT**cg**
	glnB	CAAACG**GTA**CTGATTTT**TAC**AAAAAAACTTTTGGAGAACATGT.**TA**AAAGTGTCT**gg**
	icd	AATTTC**GTA**ACAGCCAA**T**G**C**AATCAGAGCCTCCAGAAAGGAT..**TA**TGA**T**CTGCTCC**g**
	rpoD2-V	AAGTTT**GTA**TCACGAAT**TAC**ACTGCCGTGAAAATTTAACGA...**TA**TTT**T**GGACA**g**
PCC 7601	*glnA* (P1)	GAATCT**GTA**ACAAAGAC**TAC**AAAAATTCTTAATGTCATATCCT.**TA**GGA**T**ATTCCAG**gt**
PCC 6903	*glnN*	TTTTTT**GT**GCGCGTTTA**TAC**CAATCAAGTGCGATCTAATCGG..**TA**TCT**T**TTTTAT**c**
PCC 7002	*nrtP*	TAAAGA**GTA**TCAGCGGT**TAC**GAATTTAGCGAAGAAAGAATGTGA**T**TCTT**T**ATCAC**a**
WH 7803	*ntcA*	GGAACC**GT**GTGCGTTGC**TAC**AGGGTGGGAATCGATCGCTCCT..**TA**ATT**T**CCTTGA**a**

Binding sites of NtcA protein upstream from the promoter of several cyanobacterial genes. The sequences in bold are merely to draw to your attention relatively conserved nucleotides. The left hand portion is the NtcA-binding region and the right hand portion is the RNA polymerase binding region (i.e. the promoter). The table is from Herrero et al (2001) J Bacteriol 183:411-425.

You will be relieved to know that the sequences are available to you (no need to type) by using the variable `ntcA-sites`.* Note that the *with-pssm-from* box of APPLY-PSSM-TO requires a list of sequences (e.g. `ntcA-sites`). You might reasonably think that it takes an actual PSSM, such as that produced by the MAKE-PSSM-FROM function, but it doesn't. What are the top matches and what did you conclude? What code did you use?

2. Determine the information content of the aligned sequences. Based on what you find, consider altering your approach to III.A1. These functions will be helpful:

   INFORMATION-OF
   PLOT

   Note that, like APPLY-PSSM-TO, INFORMATION-OF requires a list of sequences (e.g. `ntcA-sites`). Again, a PSSM won't work. What does a plot of the information content look like? What code did you use to get the plot? How did you make use of your findings?

3. The table shown above has variable gaps in the alignment to make both parts of the sequence align. PSSM's don't do well with sequences with variable gaps. Consider ways in which you could make use of the full information in the table. Ideas?

VI.B. Troubleshooting through patterns.

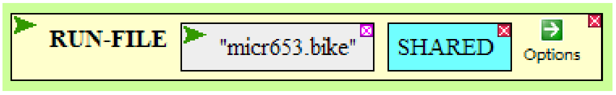
When I first put in the sequences shown in the table, I made some typographical errors. You can verify this by using my original effort (available to you as `ntcA-sites-old`)* in III.A. You

* If you execute a function calling for `ntca-sites` and get the error message "`PROBLEM: I don't understand what you mean by ...`", then see the addendum at the end of this problem set.

can stare at the sequences (as I did) by displaying them using DISPLAY-LIST with the EACH pre-option, but you'll probably have better luck if you use pattern matching to detect the problem. What pattern can you use to find the characters in the sequences that are ***not*** legitimate nucleotides? The pattern cheat sheet on the course web site might help. Ideas?

Addendum: What if pre-made variables don't work?

If you attempt to use ecfile, ntca-sites, or ntca-sites-old (properly spelled!) and get the error message "`PROBLEM: I don't understand what you mean by…`", *it could be the variables have disappeared. To get them back, go to the* ***All*** *menu, bring down RUN-FILE, and execute the function as shown to the right.*


RUN-FILE
"micr653.bike"
SHARED
Options