Bioinformatics and Bioengineering Summer Institute (2005)

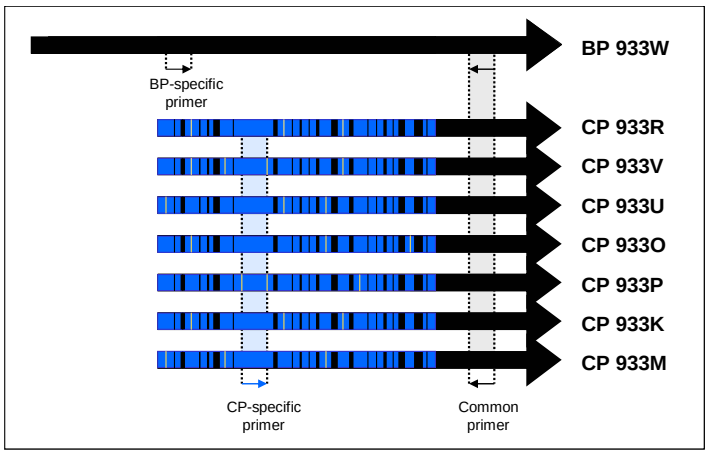
Phage sequences in bacterial genomes: Gathering the information

I. The Scenario: Construct a set of primers to detect recombinant Shiga toxin phages carrying variant tail fiber genes

We left our story having arrived at the brink of the Critical Experiment. *E. coli* O157:H7 EDL933 is able to cause death rather than mild discomfort because it contains beyond the normal pathogenic armament a gene encoding the powerful Shiga toxin. It was given this gene by bacteriophage 933W, which resides in its genome as a prophage. The phage is capable of infecting other strains of *E. coli*, and it is an open question whether the deadly effects of ED933 infection are due to expression of the toxin by this strain or instead due to infection by the phage of the army of *E. coli* that normally inhabit our intestines. It seems of great interest to learn precisely what bacteria 933W can infect.

The situation is made more complicated by the presence of seven cryptic prophages in the genome of EDL933 whose tail fiber genes could conceivably extend the host range of bacteriophage 933W. How can we determine whether genetic recombination between 933W and the cryptic phage is more than a theoretical possibility and actually takes place?

One strategy is to design PCR primers that will amplify DNA of the tail fiber gene (which determines host specificity) but only if recombination takes place. Our task is to design those primers.



One interpretation of the requirements for PCR primers to identify recombinant phage, those that have both sequences from the wild-type prophage (933W) and sequences from one of the cryptic phages.

The ideal primers should satisfy some rather strict criteria. The 5' primer needs to match all seven tail fiber genes from the cryptic prophage but not match the wild type gene, L0121. The 3' primer should match L0121 and lie beyond the point at which the kanamycin gene cassette was inserted into the gene. If cryptic phage are capable of coming back to life, then we'll need also to find design the 3' primer so that it matches L0121 but no other tail fiber gene.

II. Strategy: General considerations

How can we find sequences that fit these criteria? At first glance, the problem doesn't seem too difficult. Although it would be humane to automate the process, you could even find the primers by eye. All you'd have to do is… well, easier to show you.

A. Suppose you want to find primers common to all copies of *psbA* (encoding the central protein of the photosynthesis apparatus) of *Synechocystis* PCC 6803. You know that one

copy has the name slr1311 (that's ess-ell-are-one-three-one-one). Go to BioLingua and Blast that gene against all genes of the organism to find those that are similar:

```
(BLAST (GENE-NAMED slr1311) (GENES-OF S6803))*
```

You should find that there are three genes that show similarity to slr1311 (including slr1311 itself).

B. Align the three genes:

```
(ALIGNMENT-OF (slr1311 [two other gene names go here]))
```

C. Now scan by eye until you find a region of at least 20 nucleotides in common amongst all three gene sequences. To help you do this, note that asterisks show regions of identify at a given position.

There, that's it! Nothing to it.

Well, if you want to use this method (or an automated version of it) to find primer sequences in *E. coli* EDL933, the problem arises that EDL933 is not known by BioLingua (try Blasting some sequence against EDL933 and you'll learn how BioLingua expresses bewilderment).

The information ***is*** available, however. If we can extract the information we want from the huge (13 Mbyte) file supplied from GenBank† and get that information into BioLingua, then the world is at our feet.

With that in mind, here is the general strategy:

A. Wednesday
   1. Find the sequence of EDL933 in GenBank.
   2. Examine the sequence to figure out how to identify the information we need to tell BioLingua about
   3. Learn how to extract the information from the file
   4. (and in parallel) Learn about pattern matching as a means of extracting information.

B. Thursday
   1. Find the sequence of the L0121 (tail fiber) gene in GenBank
   2. Use that sequence within Blast to find all tail-fiber genes in EDL933
   3. Align the sequences of the tail-fiber genes to search by eye for sequences with the desired characteristics
   4. Devise a loop that will automate the process of finding candidate primer sequences and look exhaustively for them.

---

* Sorry about the syntax, which should be `(BLAST slr1311 AGAINST (GENES-OF S6803))` to be consistent with BioLingua's general feel.

† GenBank is the central repository for genetic information at the National Center for Biotechnological Information

III. Pattern matching

In examining the GenBank file containing the sequence and annotation of the EDL933 genome, you're going to encounter lines like the following:

```
  CDS         26803..29619
              /gene="ileS"
              /locus_tag="Z0030"
              /function="enzyme; Amino acyl tRNA syn; tRNA modific'n"
              /note="Residues 1 to 938 of 938 are 99.46 pct identical to
              residues 1 to 938 of 938 from Escherichia coli K-12 Strain
              MG1655: B0026"
              /codon_start=1
              /transl_table=11
              /product="isoleucine tRNA synthetase"
              /protein_id="AAG54328.1"
              /db_xref="GI:12512709"
```

Some of this is quite useful knowledge. The numbers 26803 and 29619 are the boundary coordinates of the gene. You certainly need that to extract the sequence from the large genomic sequence. The description of the gene "isoleucine tRNA synthetase"… that sounds potentially interesting. But you have no use for what's labeled `protein_id` and still less for `db_xref`.

That's all right. Just cut and paste the useful stuff and… well, this really isn't practical for all 5000+ *E. coli* genes. If you want to get this information into BioLingua, you're going to have to automate your thought processes.

How do you explain to someone (a computer for example) how you identify interesting things? Don't bother trying to list all the interesting possible words a product can contain. And it doesn't matter if you saw:

```
/product="hypocollywoggle defragulase"
```

you'd still keep it as the closest thing to a description for the gene there is. What's important, then, is the format, not the content:

`/product="`*[keep whatever is here]*`"`

In BioLingua, you can capture this sense as follows:

`(TEXT-LIKE-PATTERN "/product=\"(.*)\"" IN` *source-of-text*`)`

(note that the \ symbol is necessary to prevent the quotation mark from being interpreted as the end of the pattern). Let's look at pattern matching in action:

A. Load a sample miniaturized GenBank file:

```
(LOAD-SHARED-FILE "mini-genbank-example")
```

(this will give you a variable, mini-genbank-example, to play with).

B. Try out your pattern-matching form:

```
(TEXT-LIKE-PATTERN "/product=\"(.*)\"" IN mini-genbank-example)
```

It will return ***two*** things: the ***full match*** (corresponding to the entire pattern you specified) and the ***submatch*** corresponding to the part between the parentheses, which here is the text you really wanted to keep. Since all you want is the submatch, limit the capture to that:

```
(TEXT-LIKE-PATTERN "/product=\"(.*)\"' IN mini-genbank-example
        CAPTURE submatch)
```

We'll spend considerable time Wednesday learning how to use this powerful pattern-matching capability to extract the necessary information from the EDL933's GenBank file. What follows will be useful information as a reference, but I don't expect it to be very comprehensible right now.

IV. Description of pattern matching in BioLingua

Basic syntax:

(SEQUENCE-LIKE-PATTERN *pattern* IN *sequence*)
(SEQUENCES-LIKE-PATTERN *pattern* IN *sequence*)
(TEXT-LIKE-PATTERN *pattern* IN *text*)
(TEXTS-LIKE-PATTERN *pattern* IN *text*)

where *sequence* is anything that can be interpreted as a sequence (e.g. a string, a gene, a protein, a set of same), *text* is any string, and *pattern* is a string the nature of which is discussed below.

Simple patterns:

Any string of characters, excluding special characters (see below).

Example:

```
(SEQUENCES-LIKE-PATTERN "GGATCC" IN (SEQUENCE-OF A7120.chromosome))
```

Character sets and some special characters:

.	Any character
\\d	Any digit
\\D	Any non-digit
\\w	Any word character (letters and digits)
\\W	Any non-word character
\\s	Any space character (space, tab, and newline)
\\S	Any non-space character
[*abc*]	Set of characters
[^*abc*]	Set of excluded characters
[*a*-*z*]	Set of characters from first character to last

Examples:

```
(TEXT-SIMILAR-TO-PATTERN "\\d\\d-\\w\\w\\w-\\d\\d\\d\\d"
    IN "LOCUS    ANGLNA    2225 bp    DNA     linear   BCT 12-SEP-1993")
```

Extracts the date from a locus line of a GenBank file

```
(TEXT-LIKE-PATTERN "[^ACGT]" IN (SEQUENCE-OF Cw?0002))
```
Returns positions of the sequence with nonstandard nucleotides

Repetition symbols

`?`	Previous element may be present or absent
`+`	Previous element may be present 1 or any number of times (choose maximum number of times)
`+?`	Previous element may be present 1 or any number of times (choose minimum number of times)
`*`	Previous element may be absent or present any number of times (choose maximum number of times)
`*?`	Previous element may be absent or present any number of times (choose minimum number of times)
`{`*n*`}`	Previous element must be present the *n* number of times
`{`*m*`,n}`	Previous element may be present anywhere from *m* to *n* number of times

Example:

```
(SEQUENCE-LIKE-PATTERN "C..C..C...C" IN (SEQUENCE-OF p-Ssr3184))
```
Finds amino acid sequence with spaced cysteines.

```
(SEQUENCE-LIKE-PATTERN " [acgt]*" IN
    " 1021 accacgaagt tgctactggt ggtcagtgcg agctaggctt ccgctttggt")
```

Other special symbols

`\\t`	tab
`\\n`	newline
`\\.`	Period (because . itself is special)
`\\+`	Plus (because + itself is special)
`\\*`	Asterisk (because * itself is special)
`^`	Element that follows must occur at beginning of string (not to be confused with ^ within a set designation, e.g. `[^ACGT]`)
`$`	Element that follows must occur at end of string
`( )`	Group (to be considered a single element in pattern matching)
`( )`	Remember these elements
`|`	Or