Internet On-Ramp

Biological Sequence Analysis Using Regular Expressions
Robert M. Horton, Attotron Biosensor Corporation, Carson City, NV, USA (rmhorton@attotron.com)

“Regular expressions”, also known as “RegExps” or sometimes as “grep patterns”, are a way of specifying patterns of characters within a text file or longer string of characters. Regular expressions can contain various “wild cards” or sets of alternatives, which can be used to include ambiguity in the pattern to be matched. One common use is to perform sophisticated find-and-replace operations.

The fourth-generation Web browsers (Version 4.0 or higher of Netscape Navigator and Internet Explorer) incorporate regular expressions into their JavaScript™ scripting language, bringing the power of these tools to client-side, browser-based scripting. This article illustrates some ways that this technology can be used for analysis of nucleic acid sequences.

Finding Sequence Patterns

We begin with the classic problem of finding a genetic sequence capable of encoding a given peptide. The Sequence Pattern Finder program illustrated in Figure 1 uses a two-step process to generate a regular expression representing a degenerate nucleotide sequence that matches any gene sequence capable of encoding the specified peptide. The regular expression is then used to search a nucleic acid sequence pasted in the target box.

After typing in the desired peptide sequence, clicking on reverse translate generates an ambiguous nucleic acid sequence in which each single-letter amino acid code character is replaced with the corresponding nucleotide sequence, using the standard genetic code and ambiguity characters (1). For amino acids whose codons can vary at more than one nucleotide position, the genetic representation contains alternative codons. For example, the six codons encoding leucine are represented by either “uur” (uracil, uracil, purine) or “cun” (cytosine, uracil, any base). These alternatives are written concisely as (uur|cun); they are separated by a vertical bar character, used to represent OR, and the limit of what is included in each alternative is set by the parentheses.

The disambiguate button replaces each ambiguous code letter with the set of bases it represents; for example, n in the ambiguous sequence is replaced with the character set [acgt]. The degenerate sequence encoding leucine then becomes (uu[ag]|cu[acgu]). The highlight matches button searches for the pattern and displays a copy of the target sequence in a new browser window. If a region of the sequence matches the pattern, it is wrapped in Hypertext Markup Language (HTML) <FONT> tags to change its color.

The two-step reverse translation process makes the tool more flexible; you can type ambiguous codes directly into the second box, without starting from amino acids, or enter your own completed regular expressions in the third box. In fact, regular expressions...
entered directly are not confined to matching nucleic acid sequences; you can enter peptide sequence patterns to match against a protein sequence pasted in the target box or, similarly, match any text pattern. Regular expression pattern matching is fundamentally different from sequence alignment and "best-fit" algorithms that show you where the best match is found between probe and target sequences. While the regular expression can contain ambiguities, it will only find exact matches to that ambiguous sequence. However, it has flexibility in other ways that make it an interesting alternative to sequence similarity searching.

For example, the following regular expression matches two EcoRI sites in the same reading frame: gaattc([acgt]3)*gaattc

This regular expression matches codons encoding serine or threonine, separated by any two other codons: ((uc[acgu]|ag[cu])|ac[acgu])((acgt[3])2)((uc[acgu]|ag[cu])|ac[acgu])

For an example of an on-line database supporting searches with regular expressions, see the Yeast Genome Grepmatch page (http://www-genome.stanford.edu/Sacch3D/grepmatch.html).

Massaging Sequences

The massage sequence button calls up another JavaScript program in a new window. This Sequence Massager uses regular expressions to manipulate a DNA sequence file. Various buttons cause predefined (and hopefully useful) operations to be carried out on whatever sequence (or other text) has been pasted into the target box. Most of these actions are programmed using regular expressions. The DNA ->RNA and RNA->DNA buttons, for instance, simply replace t (or T) with u (or U) and vice versa. Other buttons remove comments, line breaks, spaces etc. Using a slightly more elaborate pattern, a button labeled remove HTML tags finds any HTML tags, such as <H1> or <FONT COLOR=red> and removes them (by replacing them with nothing). The pattern for an HTML tag is defined by the regular expression <[^>]*>, which tells the replace method to look for an opening angle bracket (or less than sign, '<'), followed by any number of characters that are NOT a closing angle bracket (greater than sign, '>'), followed by a closing angle bracket.

Readers with an interest in programming are encouraged to view the source code for these programs (by choosing View:Page Source from the top menu in Netscape). The use of regular expressions makes these programs remarkably compact; many of the JavaScript statements using regular expressions for find-and-replace in the Sequence Massager are concise enough to be incorporated directly into the HTML button tags, rather than being defined as functions. The heart of the Pattern Finder program (the findit() function) contains only two lines.

The "dialect" of regular expressions used in JavaScript is very similar to that used in the Perl programming language, and JavaScript regular expressions can generally be used in Perl with little or no modification. Perl comes with many Unix/Linux distributions and is available for Macintosh® and Windows® at no cost. Unlike client-side JavaScript, Perl can directly access files on your hard drive, and is more appropriate for such tasks as file searching and manipulation. It is also very well suited for server-side programs. For more information on uses of Perl for biological sequence analysis and manipulation, see the BioPerl home page (bio.perl.org) and Reference 2. For in-depth background and tutorials on regular expressions and discussion of various dialects, see Reference 3.

The tools presented here are intentionally simple, and are geared for student use. Since they are written in client-side JavaScript, they can be used in classrooms with or without a network. A variety of related sequence analysis tools can be found on the following Web site (www.attotron.com/dna/analysis).

These tools illustrate some of the power, flexibility, and relative simplicity of using regular expressions for analysis and manipulation of sequence data. This approach to pattern matching in biological sequences can be useful in a wide variety of applications.

ACKNOWLEDGMENT

Development of the JavaScript programs described here was supported by Grant No. 1 R43 RR13645-01 from the National Center for Research Resources.

REFERENCES