TECHSIGHTING
SOFTWARE

Premier Primer Designs

The success of experiments using the polymerase chain reaction (PCR) and related techniques is often dependent upon subtle design features of the primers employed. Primers designed with software have a significant advantage over "eyeballed" primers, because the computer is able to check thoroughly and rapidly for features that would be undesirable. Such problematic characteristics include potential mispriming sites in the target sequence, likely primer-primer pairings, and secondary structures within primers—features that are not always obvious, even to the experienced eye. Thus, primer design software eventually pays for itself, on the basis of savings for oligonucleotide syntheses, if a laboratory routinely buys its primers.

Primer Premier
PREMIER Biosoft International
Palo Alto, CA.
Demo, free;
Program, $850.
650-856-2703
www.PremierBiosoft.com

Primer Premier (PREMIER Biosoft International) is one software product for designing and optimizing primers on either Macintosh or IBM-compatible computers. Sequences can be entered into Primer Premier by pasting an existing sequence into the GeneTank window or by opening a sequence file previously created in Primer Premier. The program allows users to select one of three levels of control over the selection process. First, within the Primer window, one can manually choose primers. A second mechanism called Automatic lets the program do most of the work of identifying primers. Users simply define a few criteria, such as desired primer length, salt concentration, product size, and magnesium concentration, and the program does the rest. This includes searches at various levels of stringency; the program notes the specific stringency found next to each primer pair. The third and most powerful option is a sophisticated automated search (confusingly called "manual"). This function operates similarly to the Automatic method, but the user gains access to many more controls and can restrict searches to a single stringency criterion. Adjustable settings include a range of melting temperatures (T_m), G-C base pair content, 3′-end stability, G-C clamping, dimer elimination, and rejection of false priming structures.

When two primers are identified in a window, Primer Premier automatically checks them for undesirable qualities, such as a tendency to form stable intra- or intermolecular base pairs. If the program finds such a potential problem, the information is provided to the user via the Found button. Primer Premier also provides an interesting algorithm that is popular with some users which rates the potential success of a primer. Unexpected features in the program reveal some larger programming aspirations of the designers, including open reading frame analyses, restriction enzyme analyses, and sequence motif identification.

In the automated methods of identifying primers, however, an unfortunate design feature is apparent. The results of the top 100 or so primers are presented in one window, but users cannot view the primer pairs (and associated potential problems) unless they access a second window after selecting a pair in the first window. The problem, which is a concession to users with small screens, necessitates a good deal of shuffling back and forth between windows for what should be a simple operation in a single window. This rather inelegant design should be rethought by the programmers.

While there is little to complain about concerning the speed, accuracy, or completeness of primer searches in Primer Premier, the interface leaves something to be desired. The Macintosh version appears to be a direct port from Windows, with many standard features of the Mac interface (such as scroll bars, keyboard commands, and unobtrusive background operations) not properly implemented. The Windows product supports all standards of the Windows interface, according to the manufacturer. None of the shortcomings of Primer Premier are significant enough to recommend against purchase of the program, but tweaks of the Macintosh version would be welcome in the next version. —KEVIN AHERN

TECHSIGHTING
NET TIPS

Automating Windows

A good friend of mine is a well-known scientist who runs a lab that is using advanced functional magnetic resonance imaging (fMRI) techniques to map the workings of the brain. One of his biggest problems is handling all the data generated from a typical experiment. His data is dumped from the MRI machine into one computer, but his analysis is done on another one located in his office across campus. The problem is that he must use a proprietary piece of software to log on to the raw data computer and then he must move each file individually. Because the files are large, the process is time consuming. He asked me if he could automate this program so it could move the data without his intervention.

The answer is a definite maybe. To be more specific, we need to go into some detail about the technologies of the Windows operating system that allow automation. With automation, you can write simple scripts that will send commands to programs. Whether it is Windows 95, 98, NT, or 2000, you have at least four options when it comes to automation: COM, command line, proprietary scripting language, or keyboard macro. Let's examine these one at a time.

1) COM, which stands for Component Object Model, is nothing more than a set of specifications for how different programs should talk to each other. For example, if program A wanted to tell program B to save data to a file, program A would have to know if program B supported such a method and what the syntax was for communicating with it. COM essentially allows one program to control the behavior of another. The degree to which you can control a program is entirely dependent on how it was built. Some software, like Microsoft Word, has a rich collection of COM interfaces that can run just about all aspects of the program from code. You can automate formatting, spell-checking, HTML conversions, and so forth. Other software programs do not have any COM interfaces.

Remember, COM is not a programming language. Once you understand the COM framework of a program, you will need to manipulate COM programs (called COM Objects) with another programming language. For Windows, this is easily done with Visual Basic or Perl. A few lines of scripting code can launch a program, run some commands, exit the program, and copy output data across a network. The sky's the limit today, given the depth of functionality in languages like Visual Basic and Perl. In Windows, Visual Basic code can be embedded directly into programs like Microsoft Word to create macros. Although useful, embedded macros are one of the most common sources of computer viruses.

2) Command line. Windows programs can be started in one of two ways. You can double click on the program's icon or you can type in the path to the file in the com-

mand line window (Start Menu::Run). By running a program from the command line, you can take advantage of special flags that can be added after the path. For example, you could add a command line string that will tell the program to process an entire file of data when it starts. The command line interface is a way that programmers can expose some functionality to automation. Like in the COM description, you can use scripting code, like Visual Basic Script to run.

3) Proprietary scripting language. Some sophisticated programs have taken the concept of automation to the next level. These programs contain their own language designed specifically for automating their own actions. Take Adobe Photoshop for example. Using Photoshop's easy-to-learn scripting language, you can automate just about any command that is run from a button or pull-down menu. You can create scripts that will perform complex filtering and processing that will run a whole batch of target image files at once.

4) Keyboard macro. When all else fails, you can use a keyboard macro program (like Jmsoft's KeyText) to record keystrokes and type them into a programs form dialog. This method can work for simple forms that have standard text entry and submit buttons. Keyboard macro programs employ their own scripting language to record your actions.

The best way to find out if you can automate your program of choice is to get "under the hood": read the manual, pull down the help menu, or visit the company's Web site. Enterprise software, like Microsoft's SQL server or Exchange server, even ships with special programs whose only purpose is to automate tasks. These small programs, often called utilities, allow you to run backups, add new users from files, and perform routing maintenance.

As we move to Windows 2000, automation technology will play an even greater role in the way programs operate in a Windows environment. In fact, Windows 2000 can be viewed as nothing more than a collection of components that are designed to talk to each other. After taking some time to figure out how these components work, you should be able to automate just about any task that can be performed by a computer.

—ROBERT SIKORSKI AND RICHARD PETERS

Tech.Sight is published in the third issue of each month. Contributing editors: Robert Sikorski and Richard Peters, Mednav.com, Byfield, MA, and Kevin Ahern, Department of Biochemistry and Biophysics, Oregon State University, Corvallis, OR. Send your comments by e-mail to techsight@aaas.org.

TECHSIGHTING
IMMUNOTHERAPY

PEG Antibodies

Antibodies are part of the armamentarium that physicians can use for the treatment of a range of human conditions. A few examples include injection of antibody that recognizes tumor necrosis factor–α to treat rheumatoid arthritis; the experimental use of monoclonal antibodies to treat metastatic cancer; injection of antibody that recognizes the antigen OKT3 to revert organ rejection after transplantation; and use of sheep antibodies to bind the heart medication digoxin in case of an overdose. Physicians often prefer to use human antibodies, or at least animal antibodies in which parts of the antibody have been replaced with the human equivalent to decrease the likelihood of an allergic reaction. Such reactions occur when an epitope of an antibody is not common across species and is recognized as foreign, which prompts an immune reaction.

Generating large quantities of such "human" antibodies from mammalian cell expression systems, however, can be very expensive. An alternative is to use bacteria to express antibody fragments such as Fab′, Fv, and scFv (antibody fragments in which the VH and VL domains are linked with a peptide linker); these fragments retain the antigen-binding activity of the whole antibody and can be expressed in large quantities in bacteria. The drawback is that their plasma half-life is drastically shorter than the half-life for the whole antibody, making them unsuitable for clinical treatments that require the presence of the antibody in the circulation for sustained periods.

To circumvent this drawback, a team of scientists at Celltech Therapeutics recently published a simple method that allows the fusion of active antibody fragments to polyethylene glycol (PEG) molecules (*1*). The resulting conjugate has a much longer in vivo half-life and retains its antigen-binding activity.

The authors at first decided to randomly conjugate Fab′ from a humanized antibody to PEG molecules. The covalent reaction was catalyzed by reacting the protein with 2-iminothiolane and then with PEG-maleimide. With this approach, PEG molecules link randomly to the lysine residues of the Fab′ protein. To estimate the number of PEG molecules attached, they titrated the thiol groups both before and after the addition of PEG-maleimide. Next, they tested the pharmacokinetics of these conjugates after intravenous injection in male Wistar rats. Prior to injection, the protein-PEG conjugates had been radiolabeled with iodine-125 and the radioactivity of successive blood samples was measured with a gamma counter. They found that the half-life of the conjugates increased progressively as more PEG molecules were attached to the Fab′. Unfortunately, the antigen binding capacity of Fab′ decreased as the number of PEG molecules increased.

To get around this problem, they created conjugates with PEG molecules of varying sizes: 5, 25, and 40 kD. Increasing the size rather than the number of PEG molecules led to an increase in half-life and bioavailability, as well as an improvement in the retention of antigen binding. However, the binding of PEG still inhibited the antigen binding activity. The authors reasoned that this deleterious effect was probably due to their inability to target the binding of PEG away from the antigen-binding region of the antibody fragment.

To resolve this, they engineered Fab′ molecules that contain a hinge with a free cysteine residue. The hinge was placed far away from the antigen-binding domain and the free cysteine residue was intended as a prime target for PEG-maleimide binding. And indeed, competition enzyme-linked immunosorbent assay and surface plasmon resonance analyses revealed that the PEG conjugates now retained full antigen-binding activity; they also increased the half-life and AUC (an estimate of bioavailability) of the antibody in rats. The increase in half-life was due to an increase in the α phase, a phase that represents redistribution of a molecule in the extravascular environment. This effect on the α phase suggests that binding Fab′ to PEG slows the redistribution from the plasma to the interstitial compartment. Finally, the authors engineered a Fab′ with two hinge cysteine residues. A conjugate of two 25-kD PEG with this Fab′ retained full antigen-binding activity and reached an AUC that was 80% of the AUC of the whole IgG, as compared to an AUC of 3.7% for the free Fab′.

It will be interesting to test the applicability of this method to other antibody fragments such as Fv and scFv. Clearly, the attractiveness of this approach for clinical medicine is the ability to quickly and cheaply generate antibodies with long half-lives so they can be used to treat human ailments that would require their chronic administration. Of course, further studies would need to ensure that conjugation of Fab with PEG does not lead to new side effects.

—RICHARD PETERS AND ROBERT SIKORSKI

References

1. A. P. Chapman *et al.*, *Nature Biotechnol.* **17**, 780 (1999).