Research Report

Genome-Tools: A Flexible Package for Genome Sequence Analysis

BioTechniques 33:1334-1341 (December 2002)

**William Lee and Swaine
L. Chen**

Stanford University School of
Medicine, Stanford, CA, USA

ABSTRACT

Genome-tools is a Perl module, a set of programs, and a user interface that facilitates access to genome sequence information. The package is flexible, extensible, and designed to be accessible and useful to both nonprogrammers and programmers. Any relatively well-annotated genome available with standard GenBank® genome files may be used with *genome-tools*. A simple Web-based front end permits searching any available genome with an intuitive interface. Flexible design choices also make it simple to handle revised versions of genome annotation files as they change. In addition, programmers can develop cross-genomic tools and analyses with minimal additional overhead by combining *genome-tools* modules with newly written modules. *Genome-tools* runs on any computer platform for which Perl is available, including Unix®, Microsoft® Windows®, and Mac® OS. By simplifying the access to large amounts of genomic data, *genome-tools* may be especially useful for molecular biologists looking at newly sequenced genomes, for which few informatics tools are available. The *genome-tools* Web interface is accessible at <http://genome-tools.sourceforge.net>, and the source code is available at <http://sourceforge.net/projects/genome-tools>.

INTRODUCTION

We have created a software package called genome-tools that provides simple and powerful capabilities for viewing, retrieving, and searching genomic sequence data. It is designed to generalize to newly sequenced genomes. The tools are available for processing any fully sequenced and annotated genome currently available from GenBank® (National Center for Biotechnology Information, Bethesda, MD, USA) (2), and support for new genomes can be easily added as they become available. A Web interface to the programs provides a familiar, immediately usable front end to genomic sequence data. The back end programs can be scripted and easily interfaced with other programs in standard Unix® fashion to allow whole-genome and multi-genome sequence analysis. Finally, an application programming interface (API) is provided so that standard Perl programs can take advantage of simplified management of data files from multiple genomes provided by genome-tools. The Web front end can be accessed at <http://genome-tools.sourceforge.net>, and the source code can be downloaded at <http://sourceforge.net/projects/genome-tools/>.

MATERIALS AND METHODS

Development of the API (Orgmap.
pm and the org-map file), genome-
dependent and genome-independent

utilities, preprocessing utilities, and preprocessing of genomic data for several organisms was done under Perl version 5.004_04 for irix-n32 running on IRIX 6.5.10. The preprocessing of genome data for other organisms and the development of the Web interface application as done under Perl version 5.6.1 for i386-linux running on Debian GNU/Linux 3.0 (<http://www.debian.org>; <http://www.gnu.org>). Many of the tools use the GetOpt::Long Perl module (<http://search.cpan.org/search?dist=Getopt-Long>), which comes with the standard distribution of Perl 5.6.1 and is also available at the Comprehensive Perl Archive Network (<http://www.cpan.org>). Some preprocessing scripts use the GenBank::FF and GenBank::ParseFeatures modules by John A. Crow (available at <http://ccgb.umn.edu/groups/software/software.html>). The Concurrent Versions System (CVS) (<http://www.cvshome.org>) server for the current code and Web space is provided by SourceForge (<http://www.sourceforge.net>). The BLAST software is from the National Center for Biotechnology Information (NCBI) (<http://www.ncbi.nlm.nih.gov/BLAST>) (1). The tRNAscan software used is version 1.21 (<ftp://ftp.genetics.wustl.edu/pub/eddy/software/tRNAscan-SE.tar.Z>) (4,5).

The source data for genome-tools are the published genome files on NCBI's GenBank database site (<http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=Genome>). GenBank provides many files of information for each genome; the

Table 1. Variables Available in the Orgmap.pm Module and Their Descriptions

Variable Name	Description
\$stopology	Chromosome organization: 0 = circular and 1 = linear
\$speciesname	Full scientific name (e.g., <i>E. coli</i>)
\$subname	Additional qualifier string for \$speciesname (e.g., plasmid)
\$genbanklink	GenBank ID number
\$faafile	Full path to .faa. oneline file (processed version of .faa file, amino acid sequences of ORFs)
\$ffnfile	Full path to .ffn.oneline file (DNA sequences of ORFs)
\$fnafile	Full path to .fna.oneline file (DNA sequence of full genome)
\$pttfile	Full path to .ptt file (protein table, processed to ensure consistency and to include RNA genes)
\$gidfield	Field of .ptt file containing unique GenBank ID for each gene (0-based)
\$geneformat	Format of systematic gene name (e.g., b#### for <i>E. coli</i> K12)
\$genefield	Field of .ptt file containing systematic gene name
\$descfields	Field(s) of .ptt file containing common name and gene description
\$sequence	Variable that holds the entire genome sequence after the read_sequence subroutine is called
\$strainidir	Full path to the directory containing all data files for the organism
\$fileprefix	Base file name for all data files for the organism (includes path and common file prefix, and commonly the GenBank Genome accession number)
\$BINPATH	Full path to directory containing genome-tools utilities
\$LIBPATH	Full path to directory containing Orgmap.pm and org-map files
\$GENOMESPATH	Full path to directory containing all genomic data

files primarily used by genome-tools are the Protein Table (.ptt), FASTA Nucleic Acid file (.fna), FASTA Amino Acid file (.faa), and FASTA nucleotide coding regions file (.ffn) (11). Any genome for which these files are available is usable in genome-tools. A series of preprocessing steps for adjusting file formatting and generating some data expected by genome-tools is required before a genome downloaded from GenBank may be used with genome-tools. A script is included in the genome-tools package to perform these functions.

DESIGN

The genome-tools package consists

of three main parts: the API, utilities for preprocessing genome files, and genome analysis programs that use the API.

API

To make genome-tools useful and general, all genome-specific tasks are enclosed in a module. The general structure of any program using the genome-tools API, such as the utilities included with the package, has two main parts. The first part deals with the specifics of all the different genomes available, such as data file names, chromosome topology, and systematic gene naming convention. The rest of the program processes the genome sequence data. This structure has several advantages.

First, it provides generality; updating the first genome-specific module, for example, by adding a newly sequenced genome, immediately allows all processing programs to work on that new genome without further modification. Second, it facilitates maintenance, allowing the genome-specific module to be updated without knowing the details of any processing programs. Finally, it streamlines development; since the entire infrastructure for accessing genomic data is in one place in a consistent format, developers can concentrate on their specific data processing needs without worrying about details such as where certain files are located or which fields contain systematic or common gene names.

The key components of the genome-tools package are the Orgmap.pm Perl module file and the org-map genome description file. The org-map file stores the genome-specific information typically needed. Orgmap.pm is a Perl module that parses the org-map file and presents all the information through an interface familiar to Perl programmers. Using Perl provides the additional advantage of cross-platform compatibility so that genome-tools should run essentially unmodified on many Unix platforms and Microsoft Windows and Apple® Macintosh® computers. Including just two lines, “use Orgmap;” and “&read_orgmap;” allows any program to access all genome-specific information for any genome entered into org-map. Table 1 lists the available variables and their descriptions, and Table 2 lists subroutines. These variables and subroutines can be accessed in two ways: by using the syntax “\$Orgmap::topology;” or by requesting the export of the variable name with “use Orgmap qw(:DEFAULT \$stopology);” and then accessing the variable in the local symbol table as “\$stopology.”

Orgmap.pm handles the initial command line parsing, which usually consists of reading in the specified orgcode. The orgcode (organism code) is a unique four-letter identifier assigned during preprocessing to each genome that allows genome-tools to keep track of all installed genomes. However, other variables in genomic data files often change, particularly the gene annotations. Handling different

versions of annotation files as new data incorporated into genome annotations is truly a challenging task. If new definitive annotations become available, then the simplest solution would be to replace the old files. However, analyses very often need to be run on tentative annotations, and different applications may even need different annotations altogether if certain parts of the annotation are known to be less definitive than others. Genome-tools provides a simple mechanism for changing any of the files used for processing by adding a command line parameter. The following switches allow changing of the default data files for a particular orgcode: -faa FAA_FILENAME; -ffn FFN_FILENAME; -fna FNA_FILENAME; and -ptt PTT_FILENAME.

When Orgmap.pm finds the orgcode parameter or one of the above command line switches, it removes the parameter from the command line after processing so that application-specific parameters may still be handled by the application itself.

GENOME PREPROCESSING UTILITIES

Some preprocessing of new genome files is required to present a consistent interface to programs that use Orgmap.pm. First, the new genome is assigned a unique orgcode (which consists of any combination of four letters or numbers and is case-sensitive), and the following information is entered on a new line in the org-map file (Figure 1): orgcode; full path to files (including base of filenames); systematic gene name format; field of .ptt file containing systematic gene name (0-based); field(s) of .ptt file containing gene description, often including the common gene name (comma-separated list); full organism name; qualifying string (e.g., for additional chromosomes or plasmids); GenBank organism ID for linking to the GenBank Web site; and chromosome topology (0 = circular and 1 = linear). Next, a prepro-

Table 2. Subroutines Available in the Orgmap.pm Module and Their Descriptions

Subroutine Name	Description
read_sequence	Read genome sequence into the \$sequence variable Example: &read_sequence; print \$sequence, "\n";
get_desc	Parse a line from the .ptt file and return the gene description Example: \$description = get_desc(\$ptt_line); print \$description, "\n";
get_genename	Parse a line from the .ptt file and return the systematic gene name Example: \$genename = get_genename(\$ptt_line); print \$genename, "\n";

cessing script, named preproc-new-genome.pl, is run on the GenBank data files and performs the following tasks: strips new line characters from sequence files; cleans up the .ptt file if necessary and merges any available RNA genes into the .ptt file; calculates an intergenic sequence file, nucleotide content files, and codon usage files; obtains tRNA summary statistics using tRNAscan (to identify anticodons for predicted tRNAs); and prepares all sequence files for BLAST using the BLAST database formatting program, formatdb. The entire preprocessing is typically completed within several minutes. To avoid any irreversible information loss, the preprocessing script ensures that, for every file preprocessed, an original copy is kept unaltered. All of the original file names as they appear on the GenBank FTP server (ftp://ftp.ncbi.nih.gov) correspond to the original GenBank files with the exception of the .ptt file; the original .ptt file is renamed to .ptt.genbank-original.

Several of the preprocessing steps can be omitted if they are not needed for a particular installation. For example, the last few preprocessing steps are unnecessary if detailed tRNA information and local BLAST capabilities are not required, which is the case for the Web interface that does not provide BLAST

functionality through genome-tools. In addition, not all information is available for all genomes. Therefore, some ad hoc rules fill in data where needed. First, for simplicity and speed, we wanted a file that could be easily parsed like the .ptt file that contained all the annotated genes, including structural RNA genes. The .ptt file was already very close to complete since it was missing only non-protein genes. Therefore, we created a slightly modified .ptt file by adding additional entries for RNA genes. However, RNA genes in the GenBank data are often not annotated with full gene descriptions, gene names, or GenBank ID numbers. Because most genomes are annotated with consecutive systematic gene names starting at 1, we created systematic gene names for additional RNA genes by counting backwards from 9999 or 999, depending on the systematic naming convention. In addition, several of the programs require a value in the GenBank ID field, so we also created dummy values here for the RNA genes, starting at -1 and counting down to more negative numbers. Finally, a common inconsistency in many .ptt files is the presence and location of the systematic and common gene names. Sometimes every gene is given a systematic name, sometimes no genes have

ecol	\$GENOMESPATH/Escherichia_coli/K12/U00096	b####	5	4,8
Escherichia coli	K12	115	0	

Figure 1. Sample line from org-map file. All fields are separated by tabs. Note this line has been broken to fit in the figure, but it is all on one line in the org-map file.

them, and sometimes two fields in the same .ptt file both contain systematic and common gene names—even within the same field. For many genomes, however, there is a naming clear system so that a gene may be annotated sensibly. For example, the third gene in the *Bacillus halodurans* .ptt file is annotated as BH0003, the fourth (recF) has no systematic name, and the fifth is annotated as BH0005. In this case, we inserted BH0004 as the systematic name for the fourth gene. These changes are all handled by a custom program written for each genome, named with a .ptt-sanemaker.pl extension, which is extensively commented to indicate all changes made. Again, the original GenBank .ptt file is simply renamed and not overwritten as a precaution.

GENOME ANALYSIS PROGRAMS

The core of the functionality of genome-tools lies in the tools provided for genomic analyses. Each of these tools will display a help file if it is run

with the command line option “-h” (e.g., position.pl -h). There are two main categories of tools: genome dependent and genome independent.

Genome-Dependent Tools

position.pl. This program takes a sequence (e.g., a nucleotide motif consensus sequence) and finds all instances of this sequence in a particular genome on both strands. The program returns the strand and start coordinate for each occurrence of the sequence, as shown in Figure 2A. All International Union of Pure and Applied Chemistry (IUPAC; www.iupac.org) abbreviations (listed at <http://www.chem.qmul.ac.uk/iupac/misc/naseq.html>) (3,6–10) for multiple nucleotides are accepted. In addition, “X” is also accepted as a substitute for any nucleotide.

nearestorf.pl. This program takes an integer coordinate and finds the nearest ORFs to that coordinate on the genome. It does a binary search on the table of predicted coding regions and returns a specified number of neighbor-

ing coding regions, as shown in Figure 2B. If a given coordinate is inside a coding region, then, by default, nearestorf.pl will only return the name for the enclosing coding region. However, nearestorf.pl can be run with a command line parameter in a -# format to specify how many closest coding regions to show. Combining position.pl with nearestorf.pl provides a powerful and simple way to find all instances of a motif and the coding regions the motif appears next to.

intergenic.pl. Prints all sequences between coding regions of a genome.

Sequence Retrieval

There are three means by which specific sequences of the genome may be retrieved. Each of these programs handles circular chromosomes in that sequences will wrap around the origin.

subseq.pl. Given a position and a length, this will simply retrieve that region of the genome, as in Figure 3A. If the position is negative then it looks at the negative strand of the chromosome,

A

```
% position.pl ecol
BDHKMNRSVWXYGATCTT
>Matches_to_[CGT][AGT][ACT][GT][AC][ACGT][AG][CG][ACG][AT][ACGT][CT]GATCTT_in_ecol:
+1575119
-800075
-1914241
-2232062
```

B

```
% nearestorf.pl bsub
204900
Position 204900 in bsub is
+24 away from +BSu0182 adaB O6-methylguanine-DNA methyltransferase
-495 away from +BSu0183 ndhF NADH dehydrogenase (subunit 5)
% nearestorf.pl bsub -1
204900
Position 204900 in bsub is
+24 away from +BSu0182 adaB O6-methylguanine-DNA methyltransferase
-495 away from +BSu0183 ndhF NADH dehydrogenase (subunit 5)
+550 away from +BSu0181 adaA methylphosphotriester-DNA alkyltransferase and
transcriptional regulator (AraC/XylS family)
-2027 away from +BSu0184 ybcC ybcC
```

Figure 2. Examples of running the position.pl and nearestorf.pl programs. In all figures, the system command prompt is represented by a percent sign. Text input by the user is shown in bold, and text that the operating system or program prints is not bolded. (A) Example session for the position.pl program. “ecol” is the organism code for *E. coli* strain K12. (B) Example session for the nearestorf.pl program. “bsub” is the organism code for *B. subtilis*.

and if the length is negative then it looks in the opposite direction. Therefore, if either the position or length is negative then the sequence will be on the negative strand, and if both or neither are negative then it will be on the positive strand.

surround.pl. This program will return a sequence of specified length that is centered on the given position. If the specified position is negative then it returns the sequence on the negative strand (Figure 3B).

orfregion.pl. This provides a simple method to find the sequence of a given coding region and a specified number of bases upstream or downstream from that coding region. Coding regions are denoted by their systematic names (e.g., b numbers in *E. coli* K12 or CC numbers in *Caulobacter crescentus*), which typically contain a genome-specific alphabetic prefix, followed by a three- or four-digit number unique to that particular coding region. Figure 4 shows that orfregion.pl allows the retrieval of upstream and/or downstream

sequences on the coding strand of the specified coding region. Prefixing the systematic name with a “-” will allow the retrieval of upstream/downstream sequences without the sequence of the actual coding region.

Genome-Independent Tools

The genome-tools package contains several general purpose sequence analysis tools that do not require Orgmap.pm and therefore do not require an orgcode as a command line argument. However, to facilitate their integration with the genome-dependent tools, they also use FASTA formatted sequences for both input and output. These programs are listed below.

MW.pl. Calculates the molecular weight of a given protein sequence. Sequences must be entered as single-letter abbreviations.

translate.pl. Translates a nucleotide sequence into an amino acid sequence. Optionally, it outputs the sequence for all six reading frames or outputs single-

letter or three-letter abbreviations.

codonuse.pl. Calculates codon usage for a given nucleotide sequence.

gatk.pl. Calculates nucleotide content for a given nucleotide sequence.

revcomp.pl. Prints the reverse complement of a given nucleotide sequence. Accepts IUPAC single-letter abbreviations for multiple nucleotides.

DISCUSSION

The genome-tools package enables easy access to any genome archived in GenBank. Adding support for a new genome, depending on its annotation, typically takes only a few minutes. Additionally, the API, encapsulated in the Orgmap.pm module, transparently handles the specific details of each of the genomes, allowing any application written to the genome-tools API to be immediately generalized to nearly 70 different organisms. To further facilitate the development of new programs, we have adhered as much as possible to

Figure 3. Examples of sequence retrieval programs. (A) Example session for the subseq.pl program. “ypes” is the organism code for *Yersinia pestis*. (B) Example session for the surround.pl program. “paer” is the organism code for *Pseudomonas aeruginosa*.

BioComputing/BioInformatics>>>>>>>>

basic Unix input/output standards, which allows genome-tools to be used seamlessly in conjunction with standard GNU command line utilities.

Two examples will demonstrate the simplicity and the power of genome-tools. A very practical advantage of having genomic sequence is the ability to clone genes directly from genomic DNA. To do this by PCR, one often needs the DNA sequence flanking the gene to design PCR primers. The program orfregion.pl was written specifically for this task. Running the program on a particular organism, such as *Chlamydia trachomatis*, is done by typing in “orfregion.pl ctra” at a command prompt, as shown in Figure 4. The pro-

gram then waits for input. To extract 200 bp of sequence upstream and downstream of the β chain of DNA polymerase III (dnaN), whose systematic name is CT075, we would type "075 200 200." CT075 occurs on the complementary strand as that in GenBank's data files, and the program indicates this by placing a "-" sign in front of CT075. For convenience, all sequences are printed relative to their reading frame so that the sequences of CT075 and its flanking regions have been reversed and complemented, and the program prints the sequence on the coding strand. In addition, the output of the program is set so that the user can concatenate upstream, coding, and

downstream sequence together (this can be verified by concatenating the three sequences in that order and searching for an exact match using the position.pl program). This sequence can then be used in many primer design programs to design PCR primers for amplifying and cloning the gene.

In addition to basic genome search utilities, genome-tools provides a powerful yet simple API for the development of cross-genome analysis programs. As long as organism-dependent variables are handled by Orgmap.pm, generalizing an analysis program is as simple as changing one command line parameter. In addition, the genome-tools utilities are designed to use stan-

```
%orfregion.pl ctra
075 200 200
>-CT075_in_ctra
GTGAATTTCCGCCCTTCATGGGGCTTTTTCAAATCCGCTCTTTGTCTAAGATATTTTTTACTAAAGCTAGCCCCGAGAGACG
ACTTACTTGCTTCCATGTCAAAAACATGGTTATTTCATTCCAAACAGACTTACATGATTACAGGAAAATAATGAAATTCCG
TTATCTCCCGCAATGAATTAGGAATCTAATTAATAAAAGTTTCAGAATGTCGTTCCCAAAGACACGCCAATTCAGTATTG
ACTCATGTACTCTATTGAAAGCTGTAAACGACGAGTTGGTTTTACGGCTACAGACCTGACGGTCAGCACTCGTTGTGTTGT
TAAAGCAAAGGCTCATGAGTCTGGATCTGTTACCATCCCTTCTCGAAGATTCTTCCAACCTCATTCGTGAGTTAACAGAGG
CAAATATTGAAGTAGCAGCTCATTCTGGAGAAATGGCGACAATCACGTCGGGTTCTTCCTGTTTTCGCCTGTTAAGTATG
GGGAAAGAAGATTTCCCTATGCTTCCTGATATGCAAAATGCTCTGCGTTTCACTCTGGATTCTGAGCGATTAAAGACAT
GTTCCAAAGAACCTCTTTTGTGTATCCAGAGAAGAGAGTCGCTATGTACTCACTGGCGTTTTACTTTCTATTGCGAATG
GAACATATGACAGTCGTGGAACTGATGGGAAACGGTTAGCTAAGATCGATACAGAGATTTCTTTAGATCCTAGTTTTTCA
GGAGACTATATCATCCCTATCAAAGCCGTCGAAGAAATTTATTCGCATGTCTTCGGAAGATGTTCAATCCACTATTTTCTT
AGATCAAACATAAATCGCTGTCTGAATGTGGAAATACACTATTAGTAACCAAACCTCCTTTCTGGTGAATTCCCGGACTTTT
CCCCCTGTGATCTCCACTCATAGCAGTGTAACAATTAGATCTACATCGTGAAGAACTGATCTCCCTTCTCAAACAAGTGGCC
CTTTTCACTAATGAGTCTTCGCATTCTGTCAAAATTTAGCTTCTCACTTGAGAACTAATCTTTACAGCCAACTGCACAAA
AGTAGGAGAAAGGCAAGGTAAGTATGGCTGTCAATTATACCTGGTGAAACCTAGAAATAGCCTTTAATCCTTTCTTCTTCT
TAGACATCTGAAACACAGCCGTGATGAACTTGTTCAATTAGGTATCTCAGATTCTTACAATCCAGGAATCATTACAGAC
TCCACTCGCAGCTTGTTTGTATCATGCGCTATGAGATTACATGACGATTAA
>200_bases_upstream_of_CT075_in_ctra
GCATCCAAGGTCTCTAAAACCTCGTAATGATGAAAAGCTTTGCGATTTCGAAACAATTTCTTTAACGCCCATGTTTTTACC
TCTTAAAAACTAAGGAGAGTATAGCAAAAAGGGCTTCTTCTTGAAAGAACCGTATGCCCTCTCTCTTAAAGGATTCT
TCCTTTGTCTATTGATAATAATCCTAATCAGCAAGTACCAT
>200_bases_downstream_of_CT075_in_ctra
TGCGGGTCTTTCTTTTCTTTTCTTAAGGATTTTCAGAAATTATACAGATCTTCGTTTAGAATTAGGACCGGAGATGAATTCC
ATCTTCGGTCTTAATGCACAAGGAAGACGAATCTTCTTGAGGCTCTGTACATTCTATCACTAGGTAGGTCTTTTCGAAC
TAGTCGACTGACCGATGCCATACGTTTTGGAGCATCCAC
```

Figure 4. Use of orfregion.pl to extract flanking genomic sequences of CT075 in *C. trachomatis*.

```
% foreach orgcode (`cut -f1 LIB_PATH/org-map`)
foreach? echo ttatncaca | position.pl $orgcode | nearestorf.pl $orgcode | grep
'^ -' > $orgcode.dnaA.upstreams
foreach? end
```

Figure 5. Example loop command for processing all available genomes in the tcsh shell.

dard input and standard output. Therefore, in traditional Unix fashion, genome-tools can be combined in a pipeline with other tools and run in a loop to process all available genomes. For example, the “foreach loop” command in Figure 5 will search for the sequence “TTATNCACA,” the consensus *E. coli* DnaA box (12), in all genomes (via position.pl) (LIB_PATH should be replaced with the actual path to the org-map file.). Then, for each match, it will find the closest ORF in the genome (using nearestorf.pl). Finally, it will extract only those instances where the motif occurred upstream of a nearby ORF (through the grep utility) and put the results into a separate file for each genome with the extension “dnaA.upstreams.” These results might then be examined further for any functional relationship that may explain the presence of a common motif in the upstream regulatory regions of the genes.

While searching for an *E. coli* consensus sequence in different genomes may not be useful in itself, the preceding example illustrates how genome-tools can facilitate the generalization of genome-processing tools and leverage off of existing utilities. Thus, developing a general application with the genome-tools API adds very little overhead but greatly increases the value of such a program, especially as more genomes become available.

Generality and flexibility are the fundamental strengths of the genome-tools package. Because all the programs were designed to be flexible as well as powerful, they are easily adapted to a conveniently accessible Web interface. We have made available preprocessed versions of all microbial (bacterial and archaeal) genomes published on GenBank’s Web site as of January 7, 2002 and the complete *Saccharomyces cerevisiae* genome at <http://genome-tools.sourceforge.net>. These genomes can thus be used with the basic search utilities of the genome-tools Web interface. In addition to the existing programs and Web interface, new tools may be developed using the genome-tools API or any other Perl module, such as those available with the bioperl project (<http://www.bioperl.org>). We hope the genome-tools package and its Web site will assist all biologists to explore the

wealth of data embedded in published genome sequences and to conceive new creative ways of using the genomic data.

ACKNOWLEDGMENTS

We thank members of the Lucy Shapiro laboratory and the Harley McAdams laboratory in the Department of Developmental Biology of the Stanford University School of Medicine for help with testing genome-tools. We especially thank Harley McAdams for help with computer equipment, interface issues, and critical comments on the manuscript. This work was supported by Defence Advanced Research Projects Agency (DARPA) grant no. MDA972-00-1-0032, DOE grant no. DE-FG03-01ER63219 (W.L.), and National Institutes of Health grant no. 2T32GM07365 to the Medical Scientist Training Program (S.L.C.).

REFERENCES

1. Altschul, S.F., W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. 1990. Basic local alignment search tool. *J. Mol. Biol.* 215:403-410.
2. Benson, D.A., I. Karsch-Mizrachi, D.J. Lipman, J. Ostell, B.A. Rapp, and D.L. Wheeler. 2000. GenBank. *Nucleic Acids Res.* 28:15-18.
3. Cornish-Bowden, A. 1985. Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984. *Nucleic Acids Res.* 13:3021-3030.
4. Eddy, S.R. and R. Durbin. 1994. RNA sequence analysis using covariance models. *Nucleic Acids Res.* 22:2079-2088.
5. Fichant, G.A. and C. Burks. 1991. Identifying potential tRNA genes in genomic DNA sequences. *J. Mol. Biol.* 220:659-671.
6. International Union of Biochemistry, Nomenclature Committee. 1985. Nomenclature for incompletely specified bases in nucleic acid sequences. Recommendations 1984. *Biochem. J.* 229:281-286.
7. International Union of Biochemistry, Nomenclature Committee. 1985. Nomenclature for incompletely specified bases in nucleic acid sequences. Recommendations 1984. *Eur. J. Biochem.* 150:1-5.
8. International Union of Biochemistry, Nomenclature Committee. 1986. Nomenclature for incompletely specified bases in nucleic acid sequences. Recommendations 1984. *J. Biol. Chem.* 261:13-17.
9. International Union of Biochemistry, Nomenclature Committee. 1986. Nomenclature for incompletely specified bases in nucleic acid sequences. Recommendations 1984. *Mol. Biol. Evol.* 3:99-108.
10. International Union of Biochemistry, Nomenclature Committee. 1986. Nomenclature for incompletely specified bases in nucleic acid sequences. Recommendations 1984. *Proc. Natl. Acad. Sci. USA* 83:4-8.
11. Pearson, W.R. and D.J. Lipman. 1988. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA* 85:2444-2448.
12. Schaper, S. and W. Messer. 1995. Interaction of the initiator protein DnaA of *Escherichia coli* with its DNA target. *J. Biol. Chem.* 270:17622-17626.

Received 17 April 2002; accepted 18 June 2002.

Address correspondence to:

Dr. Swaine L. Chen
Department of Developmental Biology
Stanford University School of Medicine
279 Campus Drive, Beckman Center B300
Stanford, CA 94305, USA
e-mail: slchen@stanford.edu