Discovery of conserved translational regulatory signals with the GeneHuggers bioinformatics application development framework

Jason S. Iacovoni and Herve Prats
Institut National de la Sante et de la Recherche Medicale INSERM U589, Hormones, Facteurs de Croissance et Physiopathologie Vasculaire, Institut Federatif de Recherche Louis Bugnard, C. H. U. Rangueil, 31403 Toulouse Cedex 04, France.

ABSTRACT

We have created an application development framework that facilitates the creation of customized bioinformatics applications, code-named GeneHuggers. A large number of functions required by diverse applications are provided through a collection of Objective-C objects. In addition, GeneHuggers utilizes an embedded FireBird SQL database engine for managing annotations and sequences as well as results obtained through programmed analyses. Any required application can be integrated into the framework by wrapping that tool into an object with its own specialized methods and data types. In addition to wrapping commonly used programs such as blast, spidey, hmmer and rnamotif, we have also completely wrapped the FireBird API into a set of object methods, permitting the use of SQL features without directly interacting with the FireBird API functions. We are providing GeneHuggers as an open source distribution, under the GNU General Public License, and have utilized Doxygen to create extensive source documentation so that programmers interested in developing customized applications can take advantage of our work and simplify the coding of their actual application by using the objects provided by GeneHuggers. To exemplify the application development process using the GeneHuggers framework, we will present our work from projects concerning the study of mechanisms of translational control of gene expression. Namely, our interests are in defining sets of homologous sequences from organisms related to humans in oder to predict conserved regulatory regions that affect alternative splicing and translation inhibition by micro-RNAs.

IMPLEMENTATION

The GeneHuggers object hierarchy

The development of programs with the previous ANSI-C version of GeneHuggers was becoming cumbersome due to the separation of data types and functions inherent within the C language. Through a thoughtful object design stage, we have developed a class hierarchy that extends the GNU Object super-class. Since FASTA formatted sequences and flat files of annotations provided by Entrez Gene or the NCBI mapping coordinate data sets are all contained in files, the GHFile object is the super class to each of the relevant objects which handle any basic file Input/Output functionality. Due to the simplicity of the FASTA format, a single sub-class of GHFile can be used to handle any sequence file. Instances of GHFast can be used to read a single sequence or the entire non-redundant blast database file. GHLine handles line-oriented text found in GenBank and UniGene formatted files and GHField handles character-delimited flat files such as Gene and Homologene as well as BLAST output reports. Together, these three objects have been able to handle parsing of any flat file and sequence analysis output report we have encountered.

Embedded FireBird SQL database engine

The desire to parse all available annotation files is worthless without a method for storing and subsequently accessing the information. We have chosen the FireBird SQL engine as the core database system and have embedded its API within the GeneHuggers framework. A single GHBase object has been created with performs all base (low level) database operations. The companion GHSock object handles SQL transactions and statements. Together these two objects represent an Objective-C API for FireBird. The object methods handle all arguments to the FireBird functions and perform

the required error checking/reporting steps. In addition to the core function wrappers, there are wide ranges of methods available to perform tasks ranging from database and table creation to insert, update and select statement processing.

Bioinformaitic application wrapping with GHObjects

With access to sequences and annotations in hand, one presumably then desires to perform some form of work. We have found great utility in wrapping each application inside a object so that parameters and allocations required by the algorithm can be sequestered away from the main application. Inside each algorithm's object, a method is used to place sequences into temporary files and execute a system() call to perform the operation. Another method parses the resulting report file into its component data and then various methods can be employed to filter these results or retrieve results back into the main application. Most of the application report parsing methods can make use of the GHLine or GHField objects in order to facilitate the process of extracting information from the textual report format.

Homologous genome annotation and translational regulation

The biggest hurdle for our own research projects has been obtaining reliable sets of homologous genomic regions for genes of interest. These problems stem from multiple factors including deficiencies in annotations to variations in annotated alternative spliced transcripts and missing untranslated regions. Thus, depending upon the gene, we have had to perform one or more of the following tasks in order to obtain human, mouse and rat 3'UTR sequences for miRNA analyses.

- protein/protein blast to obtain homologous member genes from each organisms
- protein/nucleotide blast to obtain longest cDNA sequences for each protein
- nucleotide/nucleotide blast to obtain genomic neighborhood of transcript
- spidey alignment of transcript and genomic region to discern exon/intron structure
- selection of longest 3'UTR sequence from the set and subsequent spidey or sim4 alignment of that region with the relevant genomic regions of the organisms lacking annotations in the 3'UTR

Each of these processes makes use of local GeneHuggers managed databases created and mined with GeneHuggers developed applications. All regions corresponding to newly annotated 3'UTRs are dropped into a flat file FASTA database file which is subsequently parsed and integrated into the GeneHuggers framework. Then specific algorithms that find polyadenylation signals or miRNA interaction sites are run on this database and these results are further filtered based on conservation between organisms.

REMARKS

We have a complex network of interactions in place within the bioinformatics section of our laboratory. While our primary goal is to provide leads for biological research, we have also a strong desire to develop novel software tools. Tantamount to this development process is the desire to never write the same code twice. This approach has led us to the use of object-oriented programing as a means to simplify the maintenance and development of our core function libraries. The choice to integrate SQL into the GeneHuggers framework stems from the availability of an unencumbered open source SQL engine such as that offered by the Firebird project. With the basic objects described in this extended abstract, we have vastly improved the efficiency by which programs are developed. Most every method contained with the objects have found uses in multiple different programs. Of course GeneHuggers, like the resources it depends upon, is a rapidly moving target. Thus it will be of paramount interest to us to see what changes are required as the software is utilized by other researchers with other specific aims.