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 order to predict conserved regulatory regions that affect alternative splicing and translation inhibition by micro-RNAs.

Firebird SQL

This open-source SQL database server is contained within a 1.5 MB footprint and comes with an embeddable shared object library containing C API functions.

Database files for NCBI's RefSeq and Gene (human, mouse, rat and chicken records) are extremely compact.

•The RefSeq DNA and protein sequence index file is 85 MB

•All Gene Database tables fit inside a single 187 MB file

GeneHuggers has wrapped the FireBird API into two Objective-C objects that effectively mask all SQL and C API details from the programmer.

These objects are used to store downloaded databases as well as results from one application that are required by one or more downstream applications.

http://firebird.sourceforge.net/

Uses for SQL

FASTA Indexing:

GeneHuggers methods rapidly locate sequence records by querying an SQL index table containing these fields:

•*UID*: Accession Number or unique identifier

•*Path*: the absolute path to the file containing the record

•*Offset*: the number of bytes before the record start

•*Length*: the number of letters in the sequence

Taxonomy Database:

Contains *taxid*, *genus* and *species* for all organisms that should be stored locally. Used by the Gene parsers to insert records selected by organism. Also used to dissect 2 and 3 letter organism abbreviations commonly found in file names.

Gene and HomoloGene Databases:

These databases are parsed into a series of tables that are used to perform ID transformations and acquire annotations

Locales and Shared Program Data

All subsequence data is stored as Locales: *UID, offset, length, complement, featureType, featureNumber*, and a 1024 byte |NAME1=VALUE1|NAME2=VALUE2| *buffer*

All gff files and mapping coordinates for assembled genomes are stored as locale tables (large datasets are stored as indexes into the locale file using the FASTA index)

Custom data obtained in one analysis and used by one or more downstream analyses can be stored in SQL tables so that other programs can be coded to access these results.

Main GeneHuggers Objects

File Objects for reading and parsing various file types
GHLine for files with non-uniform structure (spidey)
GHField for character-delimited files (Gene/Map/BLAST)
GHFast for sequence files
GHLocale for the GeneHuggers Locale I/O data files

FireBird API Objects
GHBase handles users, aliases and database connections
GHSock handles transactions and statements

GeneHuggers SQL Objects
GHTaxBase a taxonomy database
GHGeneBase the Entrez Gene tables
GHIndexBase index sequence or Locale files
GHLocaleBase create a table from Locale data

Application-wrapper Objects
GHBlast wraps both WU-BLAST and NBCI bl2seq
GHSpidey wraps Spidey RNA/genomic alignments
GHMuscle wraps multi-sequence alignments

Local Database Setup Process

- Download, using lftp, all database files of interest
- Run a series of GeneHuggers shell scripts that call all the appropriate parsers and generate data files for SQL, uniform, indexed FASTA and Locale files
- Run another series of scripts that create the database tables and then insert all of this data into Firebird

GHcreateAlias: creates empty database
GHcreateTable: defines the structure of a table
GHinsertTable: inserts prepared tab-delimited data files

Firebird Objective-C API

To select a table row from a database with the Firebird C API, you must perform 11 steps:

1. Create a database parameter buffer with login/password information
2. Connect to the database
3. Create a transaction parameter buffer
4. Open a transaction
5. Allocate a statement
6. Allocate and configure query and result XSQLDAs
7. Prepare the statement with an SQL command
8. Commit or rollback the transaction
9. Deallocate the statement
10. Deallocate the query and result data types
11. Disconnect from the database

We execute two main types of workflows:

2) Use annotations to obtain sequences for an analysis
3) Select subsequences through an analysis and then use the databases to annotate these zones with respect to transcripts, repeats, CpG islands, MARs, etc...

The GeneHuggers Firebird API, which consists of all the above C API function calls and their data types reduces the 11 steps down to the four steps shown below:

```
#import "ghFast.h"
#import "ghIndexBase.h"
#import "ghLocale.h"

main(){
      GHFast seq = [ [ GHFast alloc ] init ];
      GHLocale loc = [ [ GHLocale alloc ] init ];
      GHIndexBase idx = [ [ GHIndexBase alloc ] init ];

      [ loc open: NULL withMode: "r" ]; /* stdin */
      [ idx connect: "ncbi" withTable: "ref" ];
      while( [ loc readLocale ] != NO ) {
            [ idx select: [ loc acc ] ];
            [ seq loadWithIndex: [ idx index ]
                  withLocale: [ loc locale ] ];
            [ seq showSeqWithAcc: stdout ];
      }
CLEAN_UP:;
      [ idx cleanup ];
      [ loc cleanup ];
      [ seq cleanup ];
}
```

Example GeneHuggers Program: ***GHloc2fast****. While the code is abbreviated for the poster, it would work. All GHObject methods return Boolean "NO" on failure so file opening and database connections should be tested when called.*

Example Workflow : miRNA

miRNAs are short 20-22 nt. non-coding RNAs that regulate mRNA expression through hybridizations in the 3'UTR. miRNA expression patterns are able to classify tumor origins and specific miRNAs are know to play a role in cancer. Some are regulated transcriptionally by oncogenes and, in turn, regulate critical growth regulatory genes.

We aimed to classify all miRNAs and their targets by selecting hybridizing sites that are conserved in human, mouse, and rat 3'UTRs. In order to do this we obtained non-annotated UTR sequences for a set of 10, 380 genes.

List of human standard gene names (symbols)

Use GHGeneBase with the HomoloGene table to select PIDs from human, mouse, and rat

Use GHGeneBase and GHIndexBase to select: PID TID GID triplets corresponding to the desired RefSeq collection (complete Assembly)

Use GHIndexBase with GHFast to use the PID and TID to get the sequence into a temporary file and run bl2seq within GHBlast to obtain CDS coordinates. Store these position in SQL for later use and use the path from the INDEX for the GID, to place all RNAs into chromosome specific files for the next step.

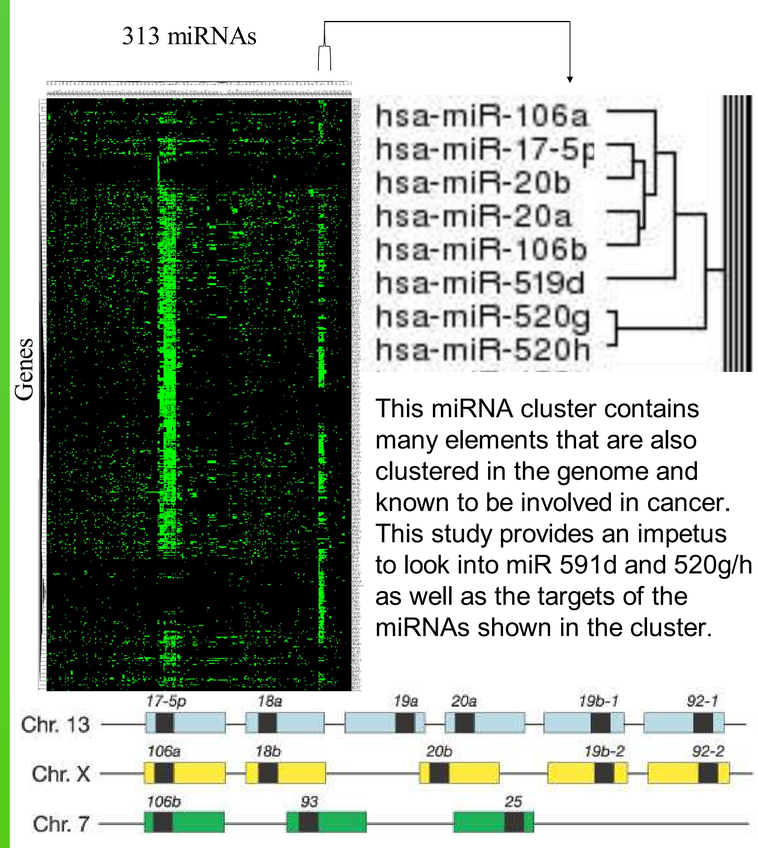
Run spidey with each RNA file against assembled chromosome

Parse spidey and combine with SQL CDS coordinates to write Locales for all 5'UTRs, exons and 3'UTRs

Use 3'UTR Locales to create FASTA file using the length of the longest UTR

Run muscle (align) on each set of 3'UTRs and then run miRNA hybridization algorithm for all miRNAs and select conserved sites

Convert data for use with cluster create images with slcview



This miRNA cluster contains many elements that are also clustered in the genome and known to be involved in cancer. This study provides an impetus to look into miR 591d and 520g/h as well as the targets of the miRNAs shown in the cluster.

Acknowledgements

We gratefully acknowledge all software developers and database curators mentioned on this poster.

Financial support provided by the INSERM.

http://sourceforge.net/projects/genehuggers

iacovoni@toulouse.inserm.fr