libsequence: a C++ class library for evolutionary genetic analysis

Kevin Thornton

Committee on Genetics, University of Chicago, 1101 East 57th Street, Chicago, IL 60637, USA

Received on March 31, 2003; revised on May 12, 2003; accepted on May 31, 2003

ABSTRACT
Summary: A C++ class library is available to facilitate the implementation of software for genomics and sequence polymorphism analysis. The library implements methods for data manipulation and the calculation of several statistics commonly used to analyze SNP data. The object-oriented design of the library is intended to be extensible, allowing users to design custom classes for their own needs. In addition, routines are provided to process samples generated by a widely used coalescent simulation.

Availability: The source code (in C++) is available from http://www.molpopgen.org
Contact: k-thornton@uchicago.edu

1 INTRODUCTION
The rapid increase in sequence data both from multiple species and from multiple individuals within species poses a demand for flexible software tools to deal with such data. Currently, much progress has been made on the front of generating routines for obtaining and manipulating large sequence databases, and for parsing the output of commonly used bioinformatics tools, in large part due to the efforts of the bioperl project (http://www.bioperl.org). However, widely available tools for the analysis of single nucleotide polymorphism (SNP) data are limited by an inability to perform 'batch' analyses (i.e. quickly summarize large volumes of data), and/or are simply not portable across different operating systems. This paper describes libsequence, a C++ class library providing an object-oriented framework for the analysis of molecular population genetic data. The library facilitates the development of custom software in three ways. First, it provides methods for data storage/manipulation. Second, a wide range of commonly used statistics to summarize and analyze SNP data are implemented. Third, the software is integrated with tools to perform analyses based on coalescent simulation. While the library itself is targeted primarily at developers, several software packages making use of the library have been written which will be useful to end-users and are able to perform a wide range of common analyses of SNP data and coalescent simulation data. Once libsequence is installed on a system, all of these other packages can be installed.

2 LIBRARY DESIGN
2.1 General comments
The C++ language (Stroustrup, 1997) has several advantages for software development. First, its support for object-oriented programming allows the addition of new functionality to software while requiring little extra code to be written, resulting in a flexible and maintainable code base. Second, support for polymorphic data types guarantees that custom extensions to existing class hierarchies remain compatible with existing library features. When the source code to object-oriented libraries is available, programmers can extend the library functionality (through the C++ inheritance mechanism) for their own purposes with relatively little effort, and these extensions remain compatible with the existing code base.

2.2 Limitations
libsequence was written to impose no arbitrary limitations on a programmer. There are no built-in limits on the length of a sequence or the size of a polymorphism table. Rather, the only limitation is the amount of RAM on a system and the length constraints imposed by the C++ system itself. On the author’s system (Linux on a Pentium II workstation), the maximum length of a std::string is approximately 4.2 billion, which imposes the upper limit on the length of an object of a sequence, and the built-in maximum size of a polymorphism table is the square of that value (the maximum length of a string × the maximum number of elements in a vector).

2.3 Development notes
libsequence is written in ANSI/ISO C++ and was developed on an x86 GNU/Linux system using the GNU C Compiler (GCC) 2.95, 3.1 and 3.2 compiler systems. It should compile with other compilers that have good support for the ANSI standard and for template instantiation (and template member functions). The build system was generated using autoproject (http://www.gnu.org/directory/devel/specific/autoproject.html) and GNU libtool

Published by Oxford University Press.
libsequence (http://www.gnu.org/software/libtool/libtool.html) is used to portably generate dynamic and static archives. A developer’s reference manual (containing several code examples) for the library was generated directly from comments in the library source code using the doxygen tool (http://www.doxygen.org) and is available in both html and pdf formats.

### 2.4 Operating systems

libsequence, and programs linking to it, have been compiled and tested on x86 GNU/Linux Systems with GCC 2.95, 3.1 and 3.2, Sun systems with GCC 2.95, and Apple OS X systems with GCC 3.1. Support for OS X is limited to machines with a recent version of the Developer’s CD installed (December 2002 at the time of this writing). In general, a recent version of GCC (http://gcc.gnu.org) is preferable, but any standards-compliant compiler system should suffice.

### 2.5 License and availability

libsequence is distributed as source code under the terms of the GNU Public License (GPL) (http://www.gnu.org/licenses.html). The source code, documentation, and examples are available for download from http://www.molpopgen.org. Documentation in html format can be found at http://www.molpopgen.org/software/libsequence/doc/html. There are also several software packages based on libsequence available for download, also under the terms of the GPL, from http://www.molpopgen.org. Online documentation (*man*) pages for these programs are available.

### 3 METHODS IMPLEMENTED

#### 3.1 Sequence manipulation

libsequence provides a flexible system for sequence manipulation, which is of use to the broader genomics community. Currently, the common FASTA format is supported, but developers can add custom formats via the C++ inheritance mechanism, requiring the writing of only two functions (one for input, one for output). These custom will be compatible with all library functions. This design principle extends to dealing with formatted alignment data and polymorphism tables. In general, however, bioperl may be a preferable platform if format conversion utilities are desired, as the intent here is to facilitate reading in data for more computationally intensive tasks.

In addition to input/output, sequences can be reversed, complemented, translated, and substrings can be taken from them. There are also methods for comparing sequences for equality, which take into account the possible presence of missing data in the sequence. libsequence also provides several function objects which facilitate writing code for tasks such as counting codon tables from a coding sequence (CDS) and calculating base composition.

#### 3.2 Divergence

libsequence implements two methods for calculating divergence between sequences. The first is the 2-parameter method of Kimura (Kimura, 1980) to calculate a distance per site between two sequences. The second is the \( K_a/K_s \) method of Comeron (Comeron, 1995), which is an extension of Li’s method (Li, 1993). Comeron’s method calculates amino acid \( (K_a) \) and silent divergence \( (K_s) \) per site between coding (CDS) sequence. The implementation of the method in libsequence uses Grantham’s distance (Grantham, 1974) to weight alternate evolutionary pathways in the case of codons that are divergent at more than one site. Also, there is the option to restrict the amount of divergence at a codon to a maximum of 1, 2 or all 3 positions. A set of classes also exist to simplify the writing of alternate weighting schemes, if they are desired. The point estimates of \( K_a \) and \( K_s \) from this method are highly correlated with maximum-likelihood estimates (i.e. Yang, 2000), but have the advantage of being much more rapid to compute. There is the disadvantage, however, that model-based hypothesis testing cannot be performed with these ‘approximate’ methods, and one should prefer the likelihood framework (Yang and Bielawski, 2000).

#### 3.3 SNP and coalescent simulation analysis

The main objective in the development of libsequence was to simplify writing custom programs for SNP analysis. SNP analysis often has two components—the summary of the data and hypothesis testing using coalescent simulation (Hudson, 1983). A modified version of the C code by Hudson (Hudson, 2002) to generate genealogies under a coalescent process is included in the library with a C++ class interface.

libsequence provides routines for a large number of summary statistics of SNP data in both sequence and simulated formats (Table 1). The details of all calculations are found in the library reference manual. libsequence was

### Table 1. List of methods for SNP analysis in libsequence

<table>
<thead>
<tr>
<th>Summary or test statistic</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Watterson’s theta</td>
<td>Watterson (1975)</td>
</tr>
<tr>
<td>Nucleotide diversity ( (\pi) )</td>
<td>Tajima (1983)</td>
</tr>
<tr>
<td>Theta ( H )</td>
<td>Fay and Wu (2000)</td>
</tr>
<tr>
<td>Number of polymorphic sites</td>
<td>N/A</td>
</tr>
<tr>
<td>Number of mutations</td>
<td>N/A</td>
</tr>
<tr>
<td>Number of external mutations</td>
<td>Fu and Li (1993)</td>
</tr>
<tr>
<td>Tajima’s ( D )</td>
<td>Tajima (1989)</td>
</tr>
<tr>
<td>Fu and Li’s ( D, F, D^<em>, F^</em> )</td>
<td>Fu and Li (1993); Simonsen et al. (1995)</td>
</tr>
<tr>
<td>Walls ( B, B' ) and ( Q )</td>
<td>Wall (1999)</td>
</tr>
<tr>
<td>Hudson’s partition test</td>
<td>Hudson et al. (1994)</td>
</tr>
<tr>
<td>Hudson’s ( C )</td>
<td>Hudson (1987)</td>
</tr>
<tr>
<td>4-Gamete test ( R_{\text{min}} )</td>
<td>Hudson and Kaplan (1985)</td>
</tr>
<tr>
<td>Number of haplotypes</td>
<td>Depaulis and Veuille (1998)</td>
</tr>
<tr>
<td>Haplotype diversity</td>
<td>Depaulis and Veuille (1998)</td>
</tr>
<tr>
<td>( F_{ST} )</td>
<td>Hudson et al. (1992a,b); Slatkin (1993)</td>
</tr>
</tbody>
</table>
written specifically to interact with Richard Hudson’s coalescent simulation program ms (Hudson, 2002), and separate versions of many routines are provided for simulated data where there would be a gain in efficiency to do so. There is also a routine for calculating the linkage disequilibrium coefficients $r^2$, $D$, and $D_1'$ [see Hudson (2001) for a recent review] with an optional frequency filter.

REFERENCES


