SEALS: A System for Easy Analysis of Lots of Sequences

D Roland Walker[1,2] and Eugene V Koonin[2]

[1]Department of Biology
Johns Hopkins University
Baltimore MD, 21218 USA
and
[2]National Center for Biotechnology Information
National Library of Medicine
National Institutes of Health
Bethesda, MD 20894 USA
Tel (301)496-2475
FAX (301)435-2433
Email should be directed to DRW at walker@ncbi.nlm.nih.gov

Abstract

We present a system of programs designed to facilitate sequence analysis projects involving large amounts of data. SEALS (System for Easy Analysis of Lots of Sequences) is a logically organized set of flexible, easily modifiable research tools, designed to run on open systems. Functionality is divided into approximately 50 commands which follow consistent syntax and semantics; wrappers are also provided for commonly used sequence analysis software to effect similar syntax for these programs. SEALS includes software for retrieving sequence information, scripting database search tools such as BLAST and MoST, viewing and analyzing search outputs, searching in and processing nucleotide and protein sequences using regular expressions, and constructing rational predictions of protein features. The system is designed to provide modular elements which can be combined, modified, and integrated with other methods in order to quickly design and execute computer experiments for sequence analysis projects at the scale of whole genomes.

Introduction

The exponential accumulation of genomic sequences, including those of complete genomes of prokaryotes and unicellular eukaryotes (Koonin et al., 1996a; Ouzounis et al., 1996), places extraordinary demands on researchers who seek to systematically analyze genetic information. Computer environments allowing researchers to handle this information efficiently are critical for the continuing success of the science of genome analysis. Several systems designed to automate different stages of genome analysis have been described (Scharf et al., 1994; Medigue et al., 1995; Ouzounis et al., 1996; Gaasterland and Sensen, 1996). However, after formulating the project of a comprehensive analysis of the complete set of protein sequences of the yeast *Saccharomyces cerevisiae*, including the comparison of the yeast proteome to those of bacteria and archaea, we found no set of unified sequence analysis tools which met our need to pursue this project with optimum accuracy and speed. A prototype of such a system, SEALS (System for Easy Analysis of Lots of Sequences), was developed and is described here.

Highly structured data representations such as Abstract Syntax Notation One (ASN.1) are appropriate for many computing tasks. For some research projects, however, especially those involving large amounts of data which must be processed by both human and machine, it is more convenient and practical to represent biological sequence information in such forms as FASTA libraries or lists of gi (NCBI global id) numbers — in other words, as simple text files.

A command-line shell such as tcsh, ksh, or bash provides the most appealing context for efficiently manipulating large numbers of text files. A modern shell provides many useful conventions such as a standard input and standard output, file globbing, and command completion, the power of which are underutilized by many research tools. SEALS consists of a set of programs which take full advantage of the command-line environment to implement straightforward text-file manipulations which effect many essential functions for large-scale sequence analysis. SEALS provides a large number of commands and routines each of which encapsulates the minimum logical function, creating a system in which it is straightforward for non-programmers to write macros and shell scripts consisting of a series of familiar commands. Because the data structures are standard and simple, the tools in SEALS can be quickly and easily combined among themselves and with standard or novel tools for exploratory investigations. We find this to be the most practical model for computer tools used in new and evolving research projects.

It has been estimated that annotating genomic sequence by hand requires as much as one year per megabase (Gaasterland and Sensen, 1996). While it is true that creating accurate sequence annotations is labor-intensive, this slow rate can be improved by orders of magnitude with the application of tools such as those included in SEALS, which aim to maximize the value of judgments made by the researcher and minimize the time spent on repetitive tasks. Automatic predictions are a useful springboard and indeed a necessary starting point, but manual adjudication and validation by a biologist yields a dataset which is more dependable and ultimately more useful.

SEALS is composed of 5 packages:

UniPred, a system for rational prediction of protein features

BLASTMORE, an interactive system for viewing and analyzing BLAST output files

GREF, a system for performing sophisticated pattern-matching and processing in fasta libraries

RWidgets, a set of basic command-line tools for manipulating fasta libraries, GenBank flatfiles, lists of gi numbers, and BLAST outputs

SPLAT, a set of tools for scripting BLAST and MoST searches and processing the results

Materials and Methods

SEALS was developed on a Silicon Graphics Challenge XL running IRIX 6.2, with 20 R10000 processors and 2560 megabytes of memory. The majority of SEALS programs were written in Perl. We used the following packages to develop the current implementation:

NCBI Software Development Toolkit 5.0
ncurses 1.9.9 e
Tcl 7.6 patchlevel 2
tcsh 6.04.00
Tk 4.2 patchlevel 2
Perl 5.004 beta 2
Perl modules not included with the standard Perl distribution:
- Storable 0.4
- Sys::AlarmCall 1.1
- Term::Curses 1.01
- Term::Readkey 2.07
- Tk 400.202

SEALS also interacts with or uses information from the following packages or programs:

BLAST (Altschul, et al, 1990)
Clustal W 1.6 (Thompson, et al., 1994)
Coils2 (Lupas, 1996)
Hmmer 1.8 (Eddy, et al., 1995)
MACAW 2.0.4 (Schuler, et al., 1991)
MoST (Tatusov, et al., 1994)
NCBI taxonomy database
Netscape 3.0
PHD (Rost, 1996)
PROSITE (Bairoch, et al. 1996)
seg (Wootton, et al., 1996)
SignalP 1.1 (Nielsen, et al., 1997)
Washington University BLAST 2.0a7 (Altschul, et al., 1996)

Please refer to the SEALS web page for information on how to obtain the packages listed above (http://www.ncbi.nlm.nih.gov/Walker/SEALS/index.html). Most of these packages have been or can be ported to any standard Unix platform. Portions of SEALS have also been implemented on an Intel platform under Windows 95 and Windows NT.

Description of SEALS packages

UniPred

The UniPred (Unified Predictor) system seeks to describe a protein or group of proteins according to the following characteristics:

predicted signal peptides
predicted transmembrane domains
predicted coiled-coil domains
predicted large non-globular domains
regions matching known motifs
regions matching other known sequences at a high degree of similarity.

UniPred consists of a series of scripts for running the programs SignalP, PHD, coils2, seg, GREF, and BLAST. Output from these programs is parsed and combined in a rational manner. For instance, signal peptides are disallowed from consideration as proper transmembrane domains, and globular domains of negligible length are merged into adjacent structural domains. Only large globular domains are submitted for BLAST searches, and only the best nonidentical statistically significant match is reported for any such search. Summaries of the analysis are generated in the following forms:

tables of each predicted feature type
unified tables for each analyzed sequence
text summary and visual representation for features of each sequence
HTML summary and visual representation for features of each sequence
overview of predicted features in complete input set.

UniPred is intended to be used for genome-scale examination of features as well as single-protein prediction. For large-scale predictions, the opportunity to hand-validate results is provided at each step through

editing of the tables of predicted features, which present each predicted feature in the context of adjacent sequence.

BLASTMORE

Blastmore is a fast, customizable, text-based viewer for BLAST output files and a launcher for other programs which can use data derived from BLAST outputs. It was designed specifically to aid in the semi-automatic processing of large numbers of BLAST outputs. Blastmore renders a BLAST output file as a navigable menu under any standard terminal screen, allowing the user to select which portions of the output to view and/or act upon. The HSPs (High-scoring Segment Pairs) associated with each hit can be viewed interactively by selecting the appropriate sequence, and sets of hits may be selected for further operations. A selection set may be created manually, or may be created based on criteria such as the p-value or score for each search hit, the text contained in the sequence deflines (definition lines), a pattern match against the subsequences corresponding to each HSP, or the taxonomic information associated with the subject sequences. Furthermore, arbitrarily complex logical operations may be performed on selection sets.

Once a sequence or set of sequences has been selected in blastmore, the user can, for instance, direct that the relevant SWISS-PROT or GenBank web page be displayed by Netscape, extract any associated abstracts from Medline, create a fasta library corresponding to the selected sequences, or create a fasta library corresponding to the portions of the selected sequences which are bounded by HSPs in the search output.

Many other built-in functions are provided, and a flexible interpolated shell-command function allows the user to issue any command which can operate on datasets generated by blastmore. For instance, blastmore can create a new BLAST output which contains only those hits which are currently selected. This pseudo-blast output can be read by any analysis program which accepts BLAST data files. If the following shell command is issued from within blastmore:

```
cap &pseudo_blast
```

blastmore will create a temporary file containing a blast output constructed from the currently selected sequences, open a new terminal window, and issue the shell command in that window, interpolating the name of the temporary file in place of the key "&pseudo_blast".

Another useful example command is

```
cat &fasta | tsplatn est - -more
```

which directs that each of the currently selected nucleic-acid sequences be searched against the database of expressed sequence tags.

Blastmore is keystroke- rather than menu-driven, for efficiency and speed. The user may bind any of the hundreds of available functions to any key using a simple configuration file, and may also associate any interpolated shell commands to keystrokes. It is also possible for simple macros to be associated with any key. For example, the following configuration file entry:

```
Z  select_by_taxnode("Saccharomyces cerevisiae"); \
   select_invert;   \
   select_by_p_value(.001);   \
   set_and;    \
   goto_number(1);   \
   item_next_selected;   \
   item_set_view
```

binds a macro to the key "Z" which causes blastmore to select all hits of reasonably significant p-value which are not derived from yeast, and to view the best hit among this set, if any.

Blastmore parses and manipulates gapped alignments correctly, and has been tested with the NCBI and Washington University BLAST programs.

RWidgets

A summary of selected RWidgets commands follows in tabular form :

and	find the logical intersection of the lines in two sets of files, maintaining input order
bert	replaces the Unix 'cat' command
blast2bounded	given a BLAST output file, return a fasta library containing only those portions of the sequences of search hits which are bounded by high-scoring segment pairs
blast2filename	report the names of BLAST output files which match the given criteria
blast2gi	extract the gi numbers associated with search hits in a BLAST output file, optionally filtering the results
blast2header	remove HSP data from BLAST outputs
clust2fasta	convert Clustal W alignment outputs into aligned fasta libraries
clustalify	wrapper for Clustal W alignment software
coilify	wrapper for coils2 coiled-coil prediction software
columnize	format fasta libraries
daffy	remove small sequences from fasta libraries
defkeys	show defline keys used by SEALS programs such as fasort and shatter
defline2gi	extract gi numbers from standard NCBI-format deflines

dupes	find duplicate lines in an unsorted file
entrezping	test whether Entrez dispatcher is functioning
fafilt	extract only DNA or amino-acid sequences from fasta libraries
famask	mask fasta files according to criteria such as predicted transmembrane domains, coiled-coil domains, or non-globular domains
fashuffle	shuffle sequences in fasta libraries
fasort	sort fasta libraries according to sequence data or keys extracted from the defline
fasta2gi	extract gi numbers from standard NCBI-format fasta libraries
fatweak	interactively truncate sequences in a fasta library
fauniq	remove duplicate entries from a sorted fasta library
feature2fasta	extract features from GenBank flatfiles as fasta entries
gap_cds	insert appropriate gaps into a nucleic acid sequence based on a gapped amino acid alignment
genbank2gi	extract gi numbers from GenBank flatfiles
gi2abstract	retrieve Medline abstracts for the given gi numbers
gi2asn1	retrieve ASN.1-format GenBank records for the given gi numbers
gi2defline	retrieve fasta deflines for the given gi numbers
gi2fasta	retrieve fasta records for the given gi numbers
gi2genbank	retrieve GenBank flatfile records for the given gi numbers
gi2genbank_html	retrieve html-ified GenBank flatfile records for the given gi numbers
gi2sibling	find the gi number for the amino acid sequence corresponding to the given nucleic acid sequence gi, for or the nucleic acid sequence corresponding to the given amino acid sequence gi
hmm2gi	extract gi numbers from a search results generated with the HMMER package
hmm_align	wrapper for the HMMER alignment programs
hmm_search	wrapper for the HMMER search programs
prettymask	convert lower-case-masked fasta entries into other representations
macaw2fasta	convert MACAW alignment outputs into aligned fasta libraries
mask_split	split masked fasta entries into multiple entries
most2gi	extract gi numbers from the results of a MoST search, optionally filtering the results
not	find the logical difference between the lines in two sets of files, maintaining input order
or	find the logical union between the lines in two sets of files, maintaining input order
pipescape	place the given file or the standard input into an open Netscape window
prosite2perl	convert a PROSITE-style regular expression into a perl-style regular expression
segify	wrapper for seg low-complexity detection program
shatter	save each fasta entry under a new filename based on words extracted from the defline
shatterblast	save each BLAST result under a new filename based on words extracted from the query information
sieve	remove large sequences from fasta libraries
signalpify	wrapper for SignalP
sortby	sort lines in a set of files according to the order of a reference set
spitscape	place the text from a Netscape window into the standard output
tax_break	show a breakdown of the taxonomic information known for organisms associated with the given gi numbers
tax_filt	filter gi numbers based on any taxonomic criterion
tax_guess	return the canonical name for a node in the NCBI taxonomy
tax_hunt	find gi numbers which match taxonomy criteria
uniquniq	remove duplicated lines from an unsorted list, maintaining input order
wubwub	find duplications within sequences using gapped BLAST
xor	find the logical symmetric difference between the lines in two sets of files, maintaining input order

GREF

Gref harnesses the powerful perl regular expression language to the task of searching in and manipulating sequence information. The pattern syntax is native perl (plus specific extensions for biological sequences), which allows sophisticated operations, such as matching regions of indeterminate length using greedy and non-greedy repetition operators.

The use of a score matrix permits weighting of characteristics of patterns. Gref supports traditional single-residue additive matrices, as well as an advanced syntax in which scores can be assigned to residues singly or in groups of any size. Scores may be simple values or arbitrarily complex expressions and can also include calls to external programs (such as seg), which allows the user to search for patterns such as "chromo domain followed by nuclear localization signal", or "zinc finger followed by large non-globular region."

Furthermore, gref can format sequences according to regular-expression rules. For example, it can force the alignment of determinate residues in a motif, mask portions of a sequence according to a pattern, or replace portions of sequences with their respective number of residues. Gref also has the ability to produce output which is compatible with motif-searching programs such as HMMER and MoST.

Gref uses a mnemonic shorthand to represent common groupings of amino acids. This contributes considerably to the readability of gref patterns. For instance, hydrophobic residues can be represented by the shorthand symbol "&oil". We also include a program, guess, which derives simple patterns from alignments using common amino-acid groupings.

Gref score matrices are particularly powerful. Each grouping which is to be subjected to scoring is defined arbitrarily by the use of parentheses in the search expression. The score matrix assigns scores based on a perl expression which may use the grouping or surrounding sequence as operands.

Consider the trivial pattern

```
C.{2,6}C
```

which indicates two cysteines separated by 2 to 6 amino acid residues. To add a scoring system to this pattern, a section of the pattern is enclosed in parentheses

```
C(.{2,6})C
```

whereupon it is "exposed" to scoring. The score matrix could have an entry such as

```
1 V 5
```

which would add 5 to the score for that sequence if a valine occurred between the two cysteines. One may also use an expression such as

```
1 + $length
```

which adds to the score the number equal to the length of the grouping.

A more practical example is found in the following pattern:

```
(.{5})[GA].{2}G.GK[ST]
```

and matrix entry:

```
1 + (&howmany("[&lard]")>=3)
```

which constitute a description of the p-loop motif, which is typical of a wide variety of NTP-utilizing enzymes (Walker et al., 1982; Gorbalenya & Koonin, 1989; Saraste et al., 1990). A score of 1 is provided for any sequence which matches the pattern and has 3 bulky hydrophobic residues (V, I, L, F, Y, M, or W) among the 5 residues in grouping one. The ability to specify a criterion such as "3 of these 5 amino acids should be large and hydrophobic" has not previously been available in publicly available pattern-matching software.

SPLAT

Splat is a BLAST scripter which includes the facility to perform iterative searches. Splat is integrated with blastmore and offers usability features such as crash recovery, progress indicators, and filtering of query sequences and resulting blast outputs. Pre-filtering the query through splat is more flexible than the built-in BLAST filtering, allowing the query sequence to be saved in a separate file before searching, permitting low-complexity masking through seg using non-default parameters, as well as allowing masking of transmembrane domains and coiled-coil domains through PHD and coils2.

The iterative search facility is implemented as a "filter" between search iterations. The user will generally specify a filter based on a p-value for search hits. Each new hit which qualifies to pass the filter is submitted in the next iteration of BLAST searches until the iterative searches cease to match novel sequences. Splat also offers the simple but powerful feature of limiting the iterative search query sequences to the subsequences bounded by HSPs in the previous iteration.

We also include the programs crumple and grouper, which are intended to work with splat. Crumple removes very closely related sequences from a set based on user-defined criteria for BLAST searches within the set. Grouper makes a simple calculation of the clusters of related sequences within the set.

The maxmost program, a derivative of splat, can be used to script motif searches with MoST.

Discussion

The motivating philosophy behind SEALS is to provide a critical mass of practical, robust command-line tools that make it possible for the sequence analysis researcher to contemplate reaping economies of scale, rather than be

cowed by the prospect of pasting a bit of text into a browser 6000 times in succession.

SEALS is an outgrowth of our previous efforts towards generating coherent schemes for genome analysis by combining multiple computer tools with expert evaluation of the results (Koonin et al., 1996b). Because it embraces only simple, basic conventions, SEALS has great potential for further evolution, since new tools can be incorporated (or can incorporate SEALS programs) as needed. Portability of the system is ensured by implementing the software (wherever possible) through portable elements, such as Perl and ncurses, which have already been implemented on dozens of platforms.

In spite of the simplicity of its design, SEALS has allowed us to more easily address "intelligent" queries. Consider, for instance, the following single command line:

```
gref nr "C.{2,4}C.{3}[LIVMFYWC].{8}H.{3,5}H" |
fasta2gi | tax_filt -node=Metazoa | gi2fasta | splatp nr
- -iter_filt="blast2gi -pcut=10e-5 -iter_limit=5 |
tax_filt -node=Metazoa"
```

which extracts all proteins from Metazoans matching the canonical C2H2 zinc-finger PROSITE pattern, and performs an iterative BLAST search, ultimately finding a large set of Metazoan proteins with significant similarity to the zinc-finger proteins. We are currently using this type of approach to delineate the evolution of gene families in various clades.

In the last few years, it has become evident that close analysis of genomic sequences can greatly enrich the original annotation with a number of new functional predictions as well as additional observations on families of paralogs, conservation of genome organization and other important features. The original and subsequent analyses of the first complete genome sequence of a cellular life form (that of *Haemophilus influenzae*) amply illustrate this point (Fleischmann et al., 1995; Caesari et al., 1995; Brenner et al., 1995; Robison et al., 1996; Brosius et al., 1996; Tatusov et al., 1996). In most cases, no new methods were required to reveal important additional information; it was sufficient to combine well-known methods and carefully interpret the results. Perhaps the greatest need in computational analysis of genomes at the moment is to intelligently and efficiently apply known methods rather than develop entirely new ones. For example, iterative database searching is not a standard part of genome analysis, though it has been shown that it frequently results in new functional predictions in cases where the original search output only contains hits to hypothetical proteins (e.g. Koonin and Tatusov, 1994; Mushegian and Koonin, 1996). It also has been shown that alternating cycles of BLAST searches and motif analysis result in significant enhancement of search sensitivity and allows the semi-automatic delineation of protein superfamilies (Koonin and Tatusov, 1994). These approaches, which have not previously been automated, are simplified by the splat tool, and should become routine procedure in genome analysis.

SEALS is still a rough prototype of the systems for genome analysis which will be required in coming years. However, it is already sufficient in its current form to greatly facilitate and accelerate the work of a computational biologist.

Acknowledgements:

DRW thanks Colombe Chappey for her generous assistance, and gratefully acknowledges the support of the Foundation for Advanced Education in the Sciences.

References

Altschul SF, Gish W (1996) Local alignment statistics. *Meth Enzymol* **266**: 460-481.

Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. *J Mol Biol* **215**: 403-410.

Bairoch A, Bucher P, Hofmann K (1996) The PROSITE database, its status in 1995. *Nucleic Acids Res* **24**: 189-196.

Brenner SE, Hubbard T, Murzin AG, Chothia C (1995) Gene duplications in H. influenzae. *Nature* **378**: 140 (1995)

Casari G, Andrade MA, Bork P, Boyle J, Daruvar A, Ouzounis CA, Schneider R, Tamames J, Valencia A, Sander C (1995) Challenging times for bioinformatics. *Nature* **376**: 647-648.

Eddy SR, Mitchison G, Durbin R (1995) Maximum discrimination models of sequence consensus. *J Comput Biol* **2**: 9-23.

Fleischmann RD, Adams MD, White O, Clayton RA, Kirkness EF, Kerlavage AR, Bult CJ, Tomb J-F, Dougherty BA, Merrick JM, et al. (1995) Whole-genome random sequencing and assembly of Haemophilus influenzae *Science* **269**: 496-512.

Gaasterland T, Sensen CW (1996) MAGPIE: automated genome interpretation. *Trends Genet* **12**: 76-78

Gorbalenya AE, Koonin EV (1989) Viral proteins containing the purine NTP-binding sequence pattern. *Nucleic Acids Res* **17**: 8413-8440.

Koonin EV, Tatusov, RL (1994) Computer analysis of bacterial dehalogenases defines a large superfamily of hydrolases with diverse specificity. Application of an iterative approach to database search. *J Mol Biol* **245**: 125-132.

Koonin, EV, Bork, P, and Sander, C (1994) Yeast Chromosome III: New gene functions. *EMBO J* **13**: 493-503.

Koonin, EV, Mushegian, AR, and Rudd, KE (1996a)

Sequencing and analysis of bacterial genomes. *Curr Biol* **6**: 404-416 .

Koonin, EV, Tatusov, RL, and Rudd, KE (1996b) Genome-scale comparison of protein sequences. *Meth Enzymol* **266**: 295-322

Lupas, A (1996) Prediction and analysis of coiled-coil structures. *Meth Enzymol* **266**: 513-525.

Medigue C, Moszer I, Viari A, Danchin A (1995) Analysis of a Bacillus subtilis genome fragment using a co-operative computer system prototype. Gene **165**: GC37-GC51.

Mushegian AR, Koonin EV (1996) Sequence analysis of eukayotuic developmental proteins: ancient and novel domains. *Genetics* **144**: 817-828.

Nielsen H, Engelbrecht J, Brunak S, von Heijne G (1997) Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *Protein Engineering* **10**: 1-6.

Ouzounis C, Casari G, Sander C, Tamames J, Valencia A (1996) Computational comparisons of model genomes. *Trends Biotechnol* **14**: 280-285.

Robison K, Gilbert W, Church GM (1996) More Haemophilus and Mycoplasma genes. *Science* **271**: 1302-1303.

Rost B (1996) PHD: predicting one-dimensional protein structure by profile-based neural networks. *Methods Enzymol* **266**: 525-539

Saraste M, Sibbald PR, Wittinghofer A (1990) The P-loop--a common motif in ATP- and GTP-binding proteins. *Trends Biochem Sci* **15**: 430-434.

Scharf M, Schneider R, Casari G, Bork P, Valencia A, Ouzounis C, Sander C (1994) GeneQuiz: a workbench for sequence analysis. *ISMB* **2**: 348-353.

Schuler GD, Altschul SF, Lipman DJ (1991) A workbench for multiple alignment construction and analysis. *Proteins Struct Funct Genet* **9**: 180-190.

Tatusov RL, Altschul SF, Koonin EV (1994) Detection of conserved segments in proteins: iterative scanning of sequence databases with alignment blocks. *Proc Natl Acad Sci* USA **91**: 12091-12095.

Tatusov RL, Mushegian AR, Bork P, Brown NP, Borodovsky M, Hayes WS, Rudd KE, Koonin EV (1996) Metabolism and evolution of Haemophilus influenzae deduced from a whole genome comparison to Escherichia coli. *Curr Biol* **6**: 279-291.

Thompson JD, Higgins DG, Gibson T (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res* **22**: 4673- 4680.

Walker JE, Saraste M, Runswick MJ, Gay NJ (1982) Distantly related sequences in the alpha- and beta-subunits of ATP synthase, myosin, kinases and other ATP-requiring enzymes and a common nucleotide binding fold. *EMBO J* **1**: 945-951.

Wootton JC, Federhen S (1996) Analysis of compositionally biased regions in sequence databases. *Meth Enzymol* **266**, 554-573.