Frontiers in Genetics

Methods
13 November 2014

FAST: Fast Analysis of Sequences Toolbox

Travis J. Lawrence [1], Dana L. Carper [1], Kyle T. Kauffman [2], Katherine C.H. Amrine [1,3], Raymond S. Lee [4], Claudia J. Canales [4] and David H. Ardell [1,2*]

[1] *Quantitative and Systems Biology, University of California, Merced, CA, USA*
[2] *Molecular and Cell Biology Unit, University of California, Merced, CA, USA*
[3] *Dept. of Viticulture and Enology, University of California, Davis, CA, USA*
[4] *School of Engineering, University of California, Merced, CA, USA*

Correspondence*:
David H. Ardell
Molecular and Cell Biology, School of Natural Sciences, University of California,
Merced, 5200 North Lake Road, Merced, CA , 95343, USA,
dardell@ucmerced.edu

ABSTRACT

FAST (Fast Analysis of Sequences Toolbox) provides simple command-line tools for rapid prototyping of powerful and reproducible bioinformatic workflows on flat-file biological sequence databases. Modeled after the GNU (GNU's Not UNIX) Textutils such as `grep`, `cut`, and `tr`, FAST tools such as `fasgrep`, `fascut`, and `fastr` are designed to be serially composed in UNIX-style pipelines. Unlike UNIX tools, FAST processes data per-sequence-record rather than per-line. This enables efficient inline processing of biological sequence data in a concise, expressive and time-tested idiom with no programming experience required. Bringing all the advantages of open-source software consistent with scientific ideals, FAST has a shallow learning curve through consistency of interfaces across utilities and a high return on investment of mastery through conformity with conventions of GNU utilities, Matlab, Perl, BioPerl and R. Portability, ease of installation, and security of FAST are inherently derived from its BioPerl and Perl foundations. The default data exchange format in FAST is MultiFastA (specifically, a restriction of BioPerl FastA format), while Sanger and Illumina 1.8+ FASTQ formatted files are also compatible. Functionality highlights include fully automated numerical and taxonomic selection and sorting of sequence records, annotation-based selection of sites from alignments, selection and transformation with regular expressions, molecular biological statistics including composition and codon usage, and molecular population genetic statistics. FAST promotes reproducibility in bioinformatic workflows by reducing manual interventions and increasing documentability of data processing and facilitates the interactive investigation and control of data by its users. FAST brings the power of Perl and BioPerl to bioinformatic users at the command-line without requiring previous programming skills and experience.

Keywords: Unix philosophy, MultiFASTA, pipeline, bioinformatic workflow, open source, BioPerl, regular expression, NCBI Taxonomy

1 INTRODUCTION

The field of molecular biology has changed significantly with the advent of Next Generation Sequencing (NGS) technology. It is now commonplace to analyze gigabases of data per experiment. Commonly, bioinformatics programs developed for visualization and basic sequence manipulation use a monolithic application with a Graphical User Interface (GUI) (**Smith et al.**, 1994; **Rampp et al.**, 2006; **?**), usually without any facility to record and document user actions for later reproduction of a bioinformatic workflow.

Early calls for reproducible research and code publication in bioinformatics and biostatistics were motivated in part to combat fraudulent research in biomedicine (**?**). Open-source software in science facilitates crowd-sourced verification, correction and extension of code-based analysis (**?**), and reuse of software and data to make discoveries using public data (**?**). The absence of adequate documentation for analysis of high-dimensional data can obscure errors in processing with profound potential ethical and financial consequences (**????**). In one analysis, only 2 of eighteen published microarray gene-expression analyses were completely reproducible in part because of proprietary closed-source software (**?**). Analytical errors are a major source of retractions in the scientific literature (**?**). Peer-review of scientific software is generally not yet required for publication but seen as necessary for integrity of the scientific enterprise across fields and requiring more computational training for all scientists (**??**). Publication and documentation of bioinformatics workflows simplifies post-publication validation of research findings and may help reduce the temptation to fish for significance and encourage more objectivity in data analysis and interpretation (**?**)

To reduce human error and increase reproducibility in the management and documentable processing of biological data, it is desirable to create self-documenting automated bioinformatic workflows — scripts and other literate programming that encode scientific workflows for machine processing of biological data. Web-based open-source workflow suites such as Galaxy (**?**), Taverna (**?**) and BioExtract (**?**) facilitate this, but an even more rapid and time-tested development platform for rapid prototyping of reproducible workflows is the UNIX command-line, specifically UNIX pipelines.

The FAST utilities are modeled after the standard Unix toolkit(**Peek**, 2001) and follow the Unix philosophy to “do one thing and do it well” (**Stutz**, 2000).

Command-line utilities for bioinformatics such as the EMBOSS package (**Rice et al.**, 2000) or the scripts that come with BioPerl (**Stajich et al.**, 2002) typically offer suites of tools with simple, well-defined functions that lend themselves to scripting, but are not necessarily designed according to the UNIX toolbox philosophy specifically to interoperate through composition.

They are written in Perl using BioPerl packages (**Stajich et al.**, 2002). This makes FAST utilities easy to adopt if you are familiar with the Unix toolbox and allows fast sequence analysis even on large datasets. Extensive documentation has been developed for each utility along with useful error messages following the recommendations of (**Seemann**, 2013) to increase usability. Lastly, FAST is open source, which makes it available to anyone free of cost. This is in line with the call to make science more assessable, open, and reproducible by other scientists and the public (**Groves and Godlee**, 2012).

2 DESIGN

An overview of Version 1.0 of the FAST project is shown in Figure 1. Descriptions of the function of utilities along with GNU Textutil analogs are given in Table 1.

FAST is split into three categories selection, transformation, and annotation and analysis. The selection category contains utilities designed to select sequences and sites from alignments based on several different criteria. For example fasgrep selects sequences by matching a regular expression to the ID, description, or sequence. The transformation utilities are used to modify the ID, description, sequence, or order of sequences using several criteria. For example, fastaxsort sorts sequences within a multifasta file based on NCBI taxonomy (**Benson et al.**, 2009; **Sayers et al.**, 2009). The annotation and analysis category contains utilities to calculate sequence composition, codon usage, sequence length, and basic population genetic statistics. Additionally these utilities can also append the results of the analysis to

Table 1. FAST 1.0 utilities

Tool	Function	Textutil analog	Default field processed
`fasgrep`	regex selection of records	`grep`	identifiers
`fasfilter`	numerical selection of records		identifiers
`fastax`	taxonomic selection of records		descriptions
`fashead`	order-based selection of records	`head`	
`fastail`	order-based selection of records	`tail`	
`fascut`	index-based selection and reordering of data	`cut`	sequences
`fasuniq`	record reduction by content and order	`uniq`	sequences
`alncut`	selection of sites by content		sequences
`gbfalncut`	selection of sites by features		sequences
`fassort`	numerical or text sorting of records	`sort`	identifiers
`fastaxsort`	taxonomic sorting of records		identifiers
`faspaste`	merging of records	`paste`	sequences
`fastr`	character transformations on records	`tr`	identifiers
`fassub`	regex substitutions on records		identifiers
`faslen`	annotate sequence lengths		descriptions
`fascomp`	annotate monomeric compositions		descriptions
`fascodon`	annotate codon usage		descriptions
`fasxl`	annotate biological translations		descriptions
`fasrc`	annotate reverse complements		descriptions
`fasconvert`	convert format of records		
`gbfgrep`	select feature neighborhoods by context	`grep`	features
`gbf2fas`	emit sequences by regex matching on features	`grep`	features
`alnpi`	molecular population genetic statistics		
`faswc`	tally sequences and characters	`wc`	sequences

the sequence description, which then can be used as selection and sorting criteria by the utilities in the selection category.

Learnability of the FAST tools is helped by making interface components such as specific options, consistent with the standard UNIX tools and across the FAST suite. Learning one FAST tool generally helps the user anticipate how to use others. In addition, specification of numerical ranges, regular expressions and other useful parameters follows standard Perl and UNIX conventions, all with the intent of making the tools fast and easy to learn.

3 IMPLEMENTATION DETAILS AND BENCHMARKING

All FAST utilities can process files or input on what is called the "standard input" stream. They all by default out to the "standard output" stream which may be connected to the "standard input" of another utility by a UNIX "pipe." It is this latter facility that eases serial processing of data. Since most FAST utilities process sequences inline, they should mostly linear runtime complexity in number of sequences. Two exceptions to inline processing in FAST utilities concern `fassort` and `fastail` which both require some paging of data into temporrary files. However, some preliminary benchmarking suggests that `fassort` runtimes also scale linearly in sequence number (fig 2).

FAST is compatible with the zero-based indexing if the sequence identifier is thought as the zeroth field of the identifier line. This field must exist in Data selection in FAST is one-based as is conventional BioPerl coordinates and bioinformatics generally.

FAST supports automated logging to ease the reproducibility of workflows. The BioPerl backend of FAST was version 1.6.901 downloaded in January, 2012. BioPerl dependencies of FAST scripts were analyzed with the Cava packager (`http://www.cavapackager.com`). To fix some problems with

I/O, the SeqIO components were updated to version 1.6.923 on June 4, 2014 and the Root components were updated on July 10, 2014 (github commit 50f87e9a4d). Some customizations of the BioPerl codebase were introduced, primarily to enable the translation of sequences containing gaps.

4 INSTALLATION AND USAGE EXAMPLES

4.1 INSTALLATION AND DEPENDENCIES

FAST requires a working Perl installation and is distributed through the Comprehensive Perl Archive Network (CPAN). In a manual install, after download, installation follows standard Perl install procedure: `perl Makefile.PL; make; make test; (sudo) make install`. A small footprint of BioPerl dependencies has been packaged together in the FAST namespace. Other CPAN dependencies can be detected and installed by the `cpan` package manager. A fully automated install may on many systems be initiated by executing `perl -MCPAN -e 'install FAST'`.

4.2 SELECTING SEQUENCES BY ENCODED MOTIFS

An advantage of the annotation approach in FAST is the ability to select and sort sequences by attributes computed and annotated into data by utilities upstream in the pipeline. For example, to select protein-coding genes from a file `cds.fas` whose translations contain the *N*-glycosylation amino acid motif (**?**), one could execute:

```
fasxl -a cds.fas | fasgrep -t xl0 "N[^P][ST][^P]" | fascut -f 1..-2
```

The first command in the pipeline translates each sequence and appends the translation to the description with the tag “xl0” (indicating translation in the zeroth reading frame). The second command in the pipeline does a regex match using the specified motif pattern on the value of a “name:value” pair in the description with tag “xl0”, hence processing the annotations produced by `fasxl`. The regex argument to `fasgrep` is quoted to protect the argument from interpretation by the shell. The last command in the pipeline removes the last field in the description, restoring records as they were before they were annotated by `fasxl`.

4.3 SORTING SEQUENCES BY THIRD CODON POSITION COMPOSITION

Another example illustrates the powerful expression of ranges in `fascut`. An optional “by” parameter in ranges allows increments or decrements in steps larger than one. To extract the third codon position bases from a gene, annotate their compositions, and sort sequences by their third-position adenosine contents, do:

```
fascut 1:-1:3 cds.fas | fascomp | fassort -nt comp_A
```

5 CONCLUDING REMARKS

Planned additions in future versions of FAST include `fasrand` and `alnrand` for sampling, permutations and bootstrapping of sequences and sites, respectively.

AVAILABILITY

FAST is available through the Comprehensive Perl Archive Network (CPAN) at `http://search.cpan.org/~dhard/FAST`

DISCLOSURE/CONFLICT-OF-INTEREST STATEMENT

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

AUTHOR CONTRIBUTIONS

D.H.A. conceived, designed, and wrote much of FAST. T.J.L. contributed major code factorizations and reorganization and `fastail`. K.T.K. contributed code including `faspaste`, and `fashead`. R.S.L. contributed an analysis of code dependencies for the FAST installer. All authors, especially D.L.C. and C.J.C., contributed documentation, testing, and code fixes. K.C.H.A. and D.H.A. wrote the FAST Cookbook. D.H.A. wrote the paper with major contributions from D.L.C. and T.J.L. All authors made minor contributions to the manuscript, reviewed the final version of the manuscript and agree to be accountable for its contents.

ACKNOWLEDGEMENT

We acknowledge Christopher Clark for help in establishing a git repository for FAST. D.H.A. gratefully acknowledges Professors Laura Landweber, Siv Andersson and Leif Kirsebom in whose laboratories the FAST tools were first developed as well as the Linnaeus Centre for Bioinformatics at Uppsala University.

Funding: D.H.A. gratefully acknowledges an NSF-DBI Postdoctoral Fellowship in Biological Informatics and awards to D.H.A. from UC Merced's Graduate Research Council and a Chancellor's Award from UC Merced's second Chancellor Sung-Mo Kang.

REFERENCES

Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., and Sayers, E. W. (2009), GenBank., *Nucleic acids research*, 37, Database issue, D26–31, doi:10.1093/nar/gkn723

Groves, T. and Godlee, F. (2012), Open science and reproducible research., *BMJ (Clinical research ed.)*, 344, jun26_1, e4383, doi:10.1136/bmj.e4383

Peek, J. (2001), Why Use a Command Line Instead of Windows?, `http://www.linuxdevcenter.com/pub/a/linux/2001/11/15/learnunixos.html`

Rampp, M., Soddemann, T., and Lederer, H. (2006), The MIGenAS integrated bioinformatics toolkit for web-based sequence analysis., *Nucleic acids research*, 34, Web Server issue, W15–9, doi:10.1093/nar/gkl254

Rice, P., Longden, I., and Bleasby, A. (2000), EMBOSS: The European Molecular Biology Open Software Suite, *Trends in Genetics*, 16, 6, 276–277, doi:10.1016/S0168-9525(00)02024-2

Sayers, E. W., Barrett, T., Benson, D. A., Bryant, S. H., Canese, K., Chetvernin, V., et al. (2009), Database resources of the National Center for Biotechnology Information., *Nucleic acids research*, 37, Database issue, D5–15, doi:10.1093/nar/gkn741

Seemann, T. (2013), Ten recommendations for creating usable bioinformatics command line software., *GigaScience*, 2, 1, 15, doi:10.1186/2047-217X-2-15

Smith, S. W., Overbeek, R., Woese, C. R., Gilbert, W., and Gillevet, P. M. (1994), The genetic data environment an expandable GUI for multiple sequence analysis., *Computer applications in the biosciences : CABIOS*, 10, 6, 671–5

Stajich, J. E., Block, D., Boulez, K., Brenner, S. E., Chervitz, S. A., Dagdigian, C., et al. (2002), The Bioperl toolkit: Perl modules for the life sciences., *Genome research*, 12, 10, 1611–8, doi:10.1101/gr.361602

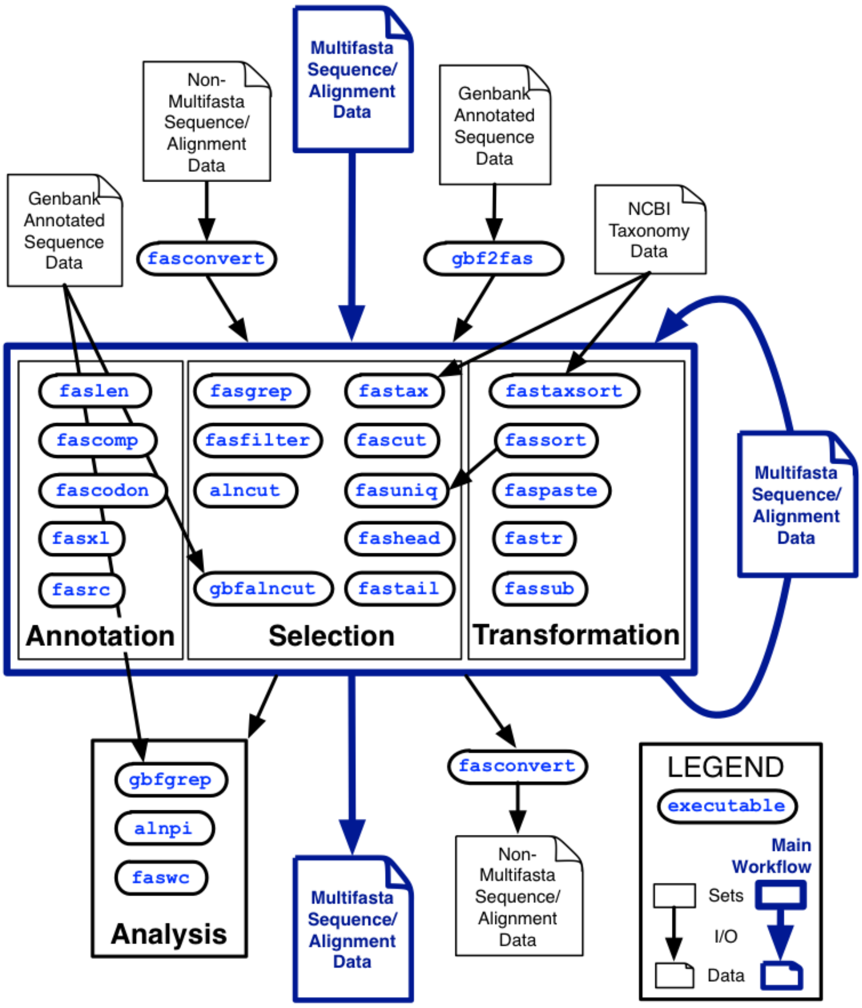

Figure 1. FAST version 1.0 with data and workflow dependencies indicated.

158 Stutz, M. (2000), Linux and the Tools Philosophy, `http://www.linuxdevcenter.com/pub/a/linux/2000/07/25/LivingLinux.html` 159

FIGURES

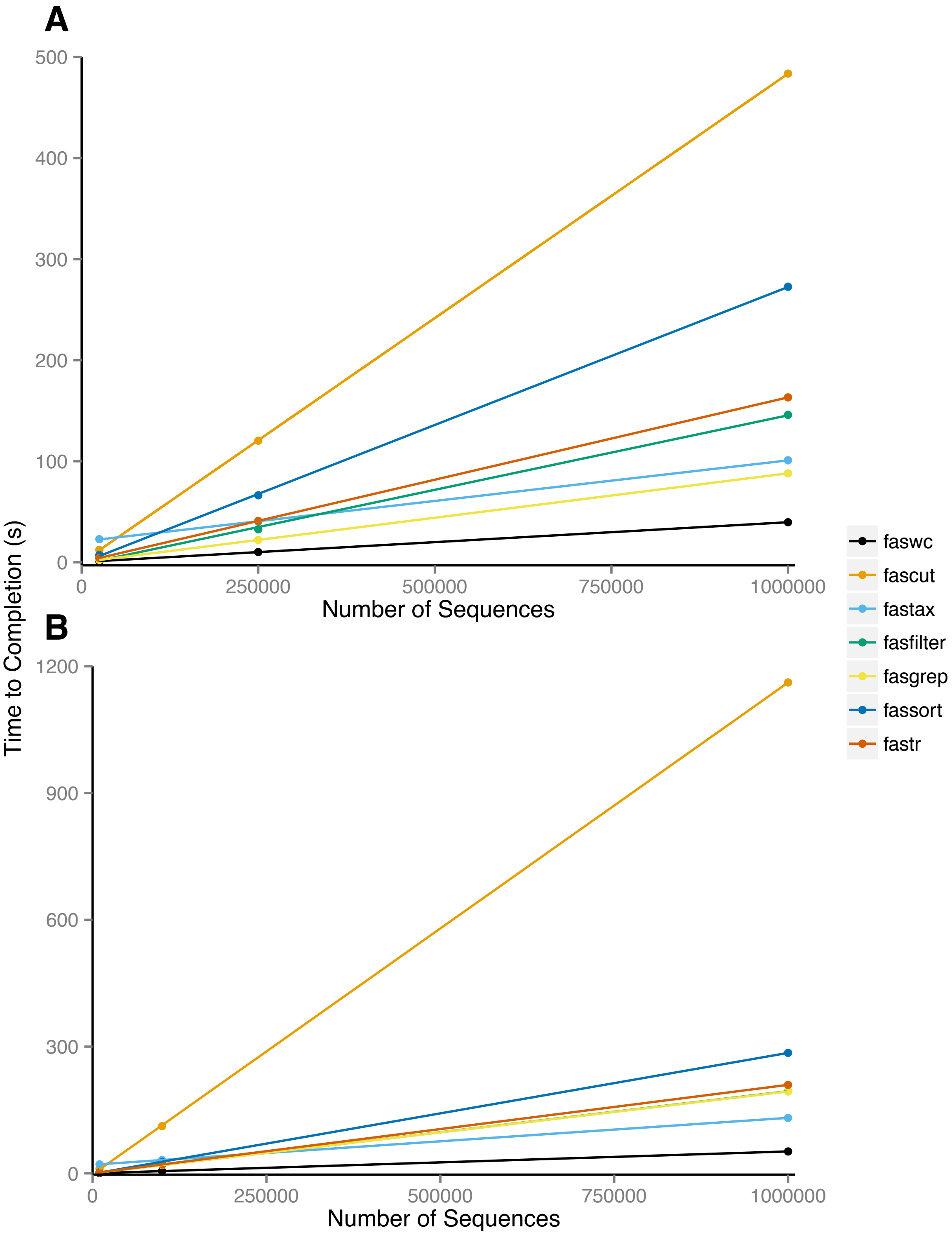

Figure 2. Average processor time of 100 repetitions required to complete analysis using indicated utility. Utilities were run on six datasets consisting of (a) 25000, 250000, and 1000000 100bp sequences and (b) 10000, 100000, and 1000000 1000bp sequences.