Frontiers in Genetics

Methods
March 2, 2015

FAST: FAST Analysis of Sequences Toolbox

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

[1] *Quantitative and Systems Biology Program, University of California, Merced, CA, USA*
[2] *Molecular Cell Biology Unit, School of Natural Sciences, 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 Cell Biology Unit, 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), built on BioPerl, provides simple, powerful open source command-line tools to filter, transform, annotate and analyze biological sequence data. Modeled after the GNU (GNU's Not Unix) Textutils such as `grep`, `cut`, and `tr`, FAST tools such as `fasgrep`, `fascut`, and `fastr` make it easy to rapidly prototype expressive bioinformatic workflows in a compact and generic command vocabulary. Compact combinatorial encoding of data workflows with FAST commands can facilitate better documentation and reproducibility of bioinformatic protocols, supporting better transparency in big biological data science. Interface self-consistency and conformity with conventions of GNU, Matlab, Perl, BioPerl, R and GenBank, help make FAST easy to learn. FAST automates numerical, text-based, sequence-based and taxonomic searching, sorting, selection and transformation of sequence records and alignment sites based on indices, ranges, tags and feature annotations, and analytics for composition and codon usage. Automated content- and feature-based extraction of sites and support for molecular population genetic statistics makes FAST useful for molecular evolutionary analysis. FAST is portable, easy to install, and secure, with stable releases posted to CPAN and development on Github. The default data exchange format in FAST is Multi-FastA (specifically, a restriction of BioPerl FastA format). Sanger and Illumina 1.8+ FASTQ formatted files are also supported. The command-line basis of FAST makes it easier for non-programmer biologists to interactively investigate and control biological data at the speed of thought.

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

1 INTRODUCTION

Bioinformatic software for non-programmers is traditionally implemented for user convenience in monolithic applications with Graphical User Interfaces (GUIs) (**Smith et al.**, 1994; **Rampp et al.**, 2006; **?**; **?**; **?**). However, today, the monolithic application paradigm can be easily outscaled by big biological data,

particularly Next Generation Sequencing (NGS) data at gigabyte and terabyte-scale. Better empowerment of non-programmers for genome-scale analytics has been achieved through web-based genome browser interfaces (**???**). On the other hand, for smaller datasets, sequence and alignment editor applications encourage manual manipulation of data, which is error-prone and essentially irreproducible. To reduce error and increase reproducibility in the publishing of bioinformatic and biostatistical protocols it is important to facilitate the documentation and automation of data science workflows through scripts and literate programming facilities (**?**) that both completely document and encode scientific workflows for machine processing of biological data.

Reproducibility in bioinformatics and biostatistics protocols is crucial to maintaining public trust in the value of its investments in high-dimensional analysis of complex biological systems (**????**). In one analysis, only two of 18 published microarray gene-expression analyses were completely reproducible, in part because key analysis steps were made with proprietary closed-source software (**?**). Furthermore, even though analytical errors are a major source of retractions in the scientific literature (**?**), peer-review and publication of scientific data processing protocols is generally not yet required to publish scientific studies. Adequate documentation of bioinformatic and biostatistical workflows and open source sharing of code upon publication (**?**) facilitates crowd-sourced verification, correction and extension of code-based analyses (**??**), and reuse of software and data to enable more scientific discovery returns from public data (**?**). Review and publication of open data science protocols may also help reduce temptations to overinterpret data and encourage more objectivity in data science (**?**), although perhaps it is expanded computational and statistical literacy and training for all scientists that is the ultimate remedy for these problems (**??**).

Web-based open-source workflow suites such as Galaxy (**?**), Taverna (**?**) and BioExtract (**?**) are a recent innovation in the direction of greater reproducibility in bioinformatics protocols for genome-scale analytics. However, the most powerful, transparent and customizable medium for reproducible bioinformatics work is only available to specialists through programming Application Programming Interfaces (APIs) such as BioPerl and Ensembl (**?**). Both workflow design suites and programming APIs require dedication and time to learn.

There is a need for more software falling between GUIs and APIs, that provides non-programmers greater bioinformatic power and more direct, real-time access to their biological data for interactive and reproducible exploration, control, and analysis. Closer inspection of data and interactive construction and control of data flows makes it so much easier to rapidly prototype error-free workflows, nipping errors in the bud that can completely confound downstream analyses. In scientific computing, the time-tested paradigm for rapid prototyping of reproducible data workflows is the Unix command-line.

In this tradition we present FAST: FAST Analysis Sequences Toolbox, modeled after the standard Unix toolkit (**Peek**, 2001), now called Coreutils. FAST is written in Perl and BioPerl, but its users don't need to know Perl, only Unix. The FAST tools follow the Unix philosophy to "do one thing and do it well" and "write programs to work together." (**Stutz**, 2000). Command-line utilities for bioinformatics such as the EMBOSS package (**Rice et al.**, 2000), the FASTX tools (**?**) 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 serial composition. Similarly, FaBox is a free and open online server with functions that overlap with FAST tools, but is not designed for serial composition. On the other hand, the Unix toolbox model has been used before in more or less more specialized bioinformatics applications such as the popular SAMTools suite (**?**) and in the processing of NMR data (**?**).

The FAST tools 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 FAST utility along with useful error messages following recommended practice (**Seemann**, 2013). FAST is free and open source; its code is freely available to anyone to re-use, verify and extend. FAST is intended to help make scientific resources

Table 1. FAST 1.0 utilities

Tool	Function	Textutil analog	Default element 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		
`gbfcut`	emit sequences by regex matching on features	`grep`	features
`alnpi`	molecular population genetic statistics		
`faswc`	tally sequences and characters	`wc`	sequences

generally more accessible, open, and reproducible by other scientists and the public (**Groves and Godlee**, 2012).

2 DESIGN

The Unix Textutils paradigm allows users to treat plain-text files and data streams as databases in which *records* correspond to single lines containing *fields* separated by *delimiters* such as commas, tabs, or strings of white-space characters. FAST extends this paradigm to biological sequence data, allowing users to treat collections files and streams of sequence records as databases for complex queries, transformations and analytics. The Textutils model is generalized exactly by FAST because it models sequence record *descriptions* as an ordered collection of fields (see below).

Another design feature of Unix tools that also characterizes the FAST tools is their ability to accept input not only from one or more files but also from what is called *standard input*, a data-stream supported by the Unix shell, and to output analogously to *standard output*. It is this facility that allows FAST tools to be serially composed in Unix *pipelines* that compactly represent an infinite variety of expressive bioinformatic workflows. The serial composability of FAST tools is represented in the overview of the project shown in Figure 1. FAST utilities may be categorized as for selection, transformation, and annotation and analysis. Utilities in the selection category select sequences or alignment sites based on various criteria. For example, `fasgrep` selects sequence records by matching regular expressions against identifiers, descriptions, sequences or specified components of descriptions. A full description of all utilities included in FAST 1.0 is shown in Table 1.

The default data exchange format for FAST tools is the universally recognized FastA format (**?**). While no universal standard exists for this format, for FAST, "FastA format" means what is conventionally called

"multi-fasta" format of sequence or alignment data, largely as implementated in BioPerl in the module `Bio::SeqIO::fasta` (**Stajich et al.**, 2002).

In the FAST implementation of FastA format, multiple sequence records may appear in a single file or input stream. Sequence data may contain gap characters. The logical elements of a sequence record are its *identifier*, its *description* and its *sequence*. The identifier (indicated with `id` in the example below) and description (`desc`) together make the *identifier line* of a sequence record, which must begin with the sequence record start symbol > on a single line. The description begins after the first block of white-space on this line (indicated with `<space>`). The *sequence* of a record appears immediately after its identifier line and may continue over multiple lines until the next record starts.

In FAST, users may specify fields in sequence records using delimiters (indicated by `<delim>`) quite generally using perl-style *regular expressions*. FAST uses one-based indexing of fields as indicated in this example:

```
>seq1-id<space>seq1-desc-field1<delim>seq1-desc-field2<delim>...
seq1-sequence
seq1-sequence
...
seq1-sequence
>seq2-id<space>seq2-desc-field1<delim>seq2-desc-field2<delim>...
seq2-sequence
seq2-sequence
...
seq2-sequence
```

In FAST, the sequence identifier is thought as the zeroth field of the identifier line. One-based indexing of description fields in FAST is therefore consistent with zero-based indexing in Perl and one-based indexing of sequence coordinates, making all indexing consistent and uniform in FAST.

Most FAST tools extend the field-based paradigm further by supporting *tagged values* in sequence record descriptions. Tagged values are name-value pairs with a format "name=value" as common in General Feature Format (GFF) used in sequence annotation. Support for tagged values makes it possible to operate on sequence records using unordered or heterogeneous annotations in descriptions. Also, many FAST tools have an "annotation" option directing them to augment sequence records with their own output calculations, vastly expanding the types of operations and queries that FAST can represent.

3 IMPLEMENTATION DETAILS AND BENCHMARKING

Nearly all FAST utilities process sequence records inline and therefore have linear runtime complexity in the number of sequences. Exceptions are `fassort` and `fastail` which both require some paging of data into temporary files. We performed benchmarking of FAST tools using randomly generated sequences and the Benchmark v1.15 perl module on a MacBook Pro 2.5 Ghz Intel i7, with 8 Gb of RAM. We examined average CPU runtime over 100 replicates, comparing input sizes of 25K, 250K, or 1M sequence records of length 100, 10K, 100K, or 1M bp. Our benchmarking results show that despite data paging, `fassort` runtimes scale linearly with input size (fig 2).

The BioPerl backend of FAST 1.0 is version 1.6.901 downloaded in January, 2012. `Bio::SeqIO` components were updated to version 1.6.923 on June 4, 2014 and some Bio::Root components were updated on July 10, 2014 (github commit 50f87e9a4d). We introduced a small number of customizations to the BioPerl code-base, primarily to enable the translation of sequences containing gaps. All of the BioPerl dependencies of FAST are isolated under its own FAST name-space.

To help reduce the overall installation footprint of FAST, BioPerl dependencies of FAST scripts were analyzed with the Cava packager (`http://www.cavapackager.com`).

Further implementation details of individual FAST tools follows.

3.1 FASGREP

Mainly for fine-grained regular expression-based searching and selection of sequence records, `fasgrep` also supports motif-based sequence selections and general analysis of sequence patterns as represented by Perl regular expressions. The BioPerl `Bio::Tools::SeqPattern` library supports optional ambiguity expansion of IUPAC codes for nucleotides and proteins in regular expression arguments. All of the search and selection utilities in FAST support optional complementation and case-insensitive searching.

3.2 FASFILTER

`fasfilter` supports precise numerical-based selections of sequence records from numerical data in identifiers, descriptions, fields or tagged-values in descriptions. Both open, closed and compound ranges are supported in different syntax.

3.3 FASCUT

`fascut` supports index-based selections of characters and fields in sequence records allowing repetition, reordering, variable steps, and reversals. A sequence of selection indices and index-ranges are specified conventionally by comma-separated lists of integers and integer ranges in Perl-style or Genbank coordinate style ("from..to") in R/Octave-style ("from:to") or Unix `cut`-style ("from-to"). Negative indices count backwards from last characters and fields. `fascut` outputs the concatenation of data selections for each sequence record. Variable step-sizes in index ranges conveniently specify first, second or third codon positions in codon sequence records, for example.

3.4 ALNCUT

Content-based selection of sites in alignments including gap-free sites, non-allgap sites, variable or invariant sites and parsimoniously informative sites, or their set-complements, all with the option of state-frequency-thresholds applied per site.

3.5 GBFCUT

Allows annotation-based sequence-extraction from GenBank format sequence files, useful for extracting all sequences that correspond to sets of genes or other annotated features in genome data.

3.6 GBFALNCUT

This utility automates the selection of sites from alignments that correspond to one or more features annotated on one of the sequences in a separate GenBank record. This workflow eliminates the need for manual entry of coordinates and implements a useful bioinformatic query in terms of known and reproducible quantities from public data and sequence records, allowing users to query sites based on biological vocabularies of sequence features. Data and command examples are provided to reproduce the feature-based analysis of polymorphism data in (**?**).

3.7 FASSORT

Our implementation of `fassort` handles numerical and textual sorting of records by their components, including reversals. Pages of data are sorted with optimized routines in Perl `Sort::Key` that if necessary are written to temporary files and merged with `Sort::MergeSort`.

3.8 FASUNIQ

With `fasuniq` the user may remove records that are duplicates with respect to a specified component or field. Like its Unix Coreutil analog, `fasuniq` only compares subsequent records on input, usually requiring that its input is sorted first by `fassort`.

3.9 FASTAX AND FASTAXSORT

Taxonomic searching and sorting of sequence records, when those records are already annotated with NCBI taxonomic identifiers, is enabled by the FAST tools `fastax` and `fastaxsort` using taxonomic data from NCBI taxonomy (**Benson et al.**, 2009; **Sayers et al.**, 2009). Taxonomic selections may be logically negated and/or restricted to valid NCBI taxonomic identifiers. A sample of data from tRNAdb-CE (**?**), which includes taxonomic names from NCBI Taxonomy, is included with the package and referred to in the documentation, showing how these utilties may be used.

3.10 FASTR AND FASSUB

`fastr` handles all character-based transformations of sequence records including transliterations, deletions and "squashing" (deletion of consecutive repeats). Includes support to restrict or remap sequence data to strict or IUPAC ambiguity alphabets and removal of gap characters from sequencing. `fassub` allows more arbitrary substitutions on sets of strings matched to Perl regexes, analogous to the Perl `s///` substitution operator.

3.11 FASCOMP, FASXL AND FASCODON

These utilities provide for annotation and analytics of compositions, translations, and codon usage frequencies of sequence records (with start and stop codons counted distinctly, in the last case). All genetic codes included in BioPerl, ultimately from NCBI Entrez, are supported.

3.12 ALNPI

`alnpi` outputs molecular population genetic statistics cited in Table 2 for each alignment on input. It can output a set of statistics for each alignment on input in plain text or LaTeX format. `alnpi` also supports sliding window and pairwise analysis of input data. Data and command examples are provided to reproduce the tables and sliding window analyses of statistics published in (**?**). All of the code for these calculations has been reviewed and compared against calculations produced from DNASP (**?**) as described previously (**?**).

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

Table 2. Molecular Population Genetic Statistics in FAST

Statistic	Symbol	Citation
Number of sequences	n	
Number of alleles/distinct sequences	k	
Number of segregating sites	S	
Fraction of segregating sites	s	
Average number of pairwise differences		(**?**)
Nucleotide Diversity	π	(**?**)
Watterson estimator	θ_W	(**?**)
Expected number of alleles	$E(K)$	(**?**)
Tajima's D	D	(**?**)
Fu and Li's D*	$D*$	(**?**)
Fu and Li's F*	$F*$	(**??**)
Fu and Li's Eta S	η_S	(**?**)
Fu and Li's Eta	η	(**?**)

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 TEST SUITE AND DOCUMENTATION

FAST includes rudimentary test suites for each utility, and a FAST Cookbook has been contributed to the installation package.

4.3 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`.

4.4 SORTING RECORDS 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 third-position bases from codon sequence records, compute and annotate their compositions into record descriptions, ultimately sorting records by their third-position adenosine contents, do:

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

5 CONCLUDING REMARKS AND FUTURE DIRECTIONS

Planned additions in future versions of FAST include `fasrand` and `alnrand` for automated sampling, permutations and bootstrapping of sequences and sites, respectively, and `fasgo` and `fasgosort` for selection and sorting of records by Gene Ontology categories (**?**).

AVAILABILITY

Stable versions of FAST are released through the Comprehensive Perl Archive Network (CPAN) at `http://search.cpan.org/~dhard/`. Development of FAST is through its GitHub at `https://github.com/tlawrence3/FAST`. For latest news on the FAST project please check the Ardell Lab homepage at `http://compbio.ucmerced.edu/ardell/software/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. P.J.B. tested installation and running on Windows using Strawberry Perl. 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 and Peter Becich for contributing some testing of the installer. 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 from UC Merced's Graduate Research Council and a Chancellor's Award from UC Merced's second Chancellor Sung-Mo Kang, as well as the NSF-funded program in Undergraduate Research in Computational Biology at UC Merced (DBI-1040962).

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

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

FIGURES

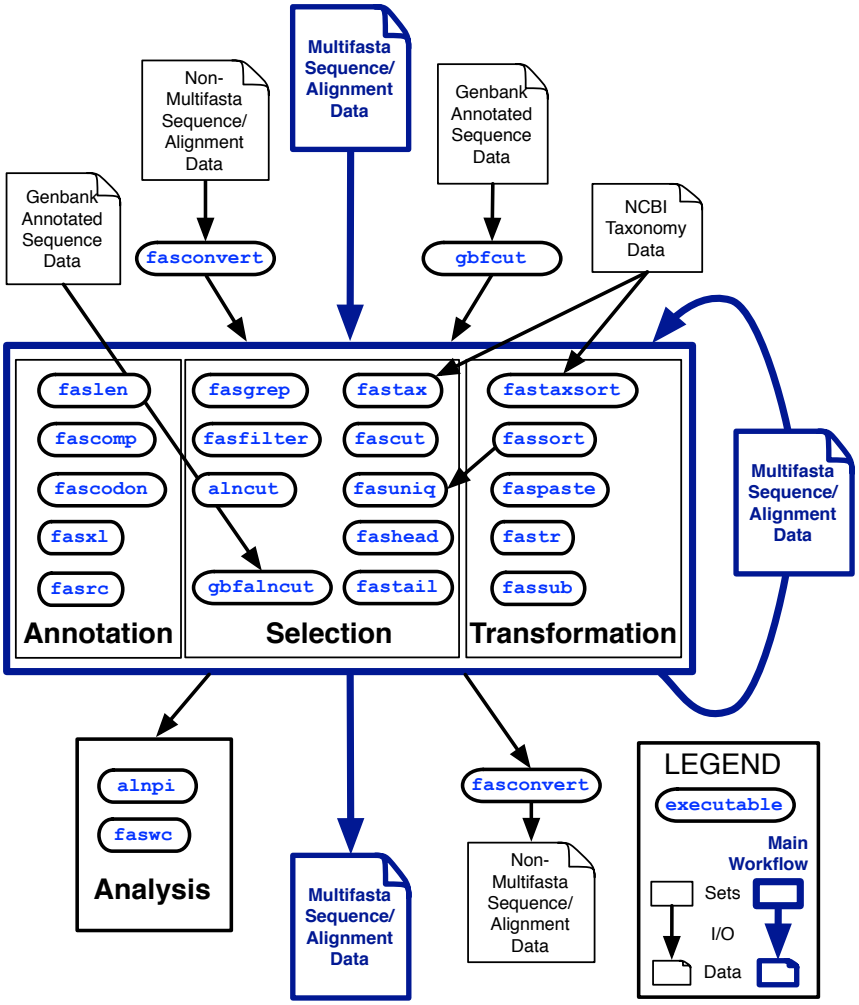

Figure 1. FAST version 1.0 with data and workflow dependencies indicated.

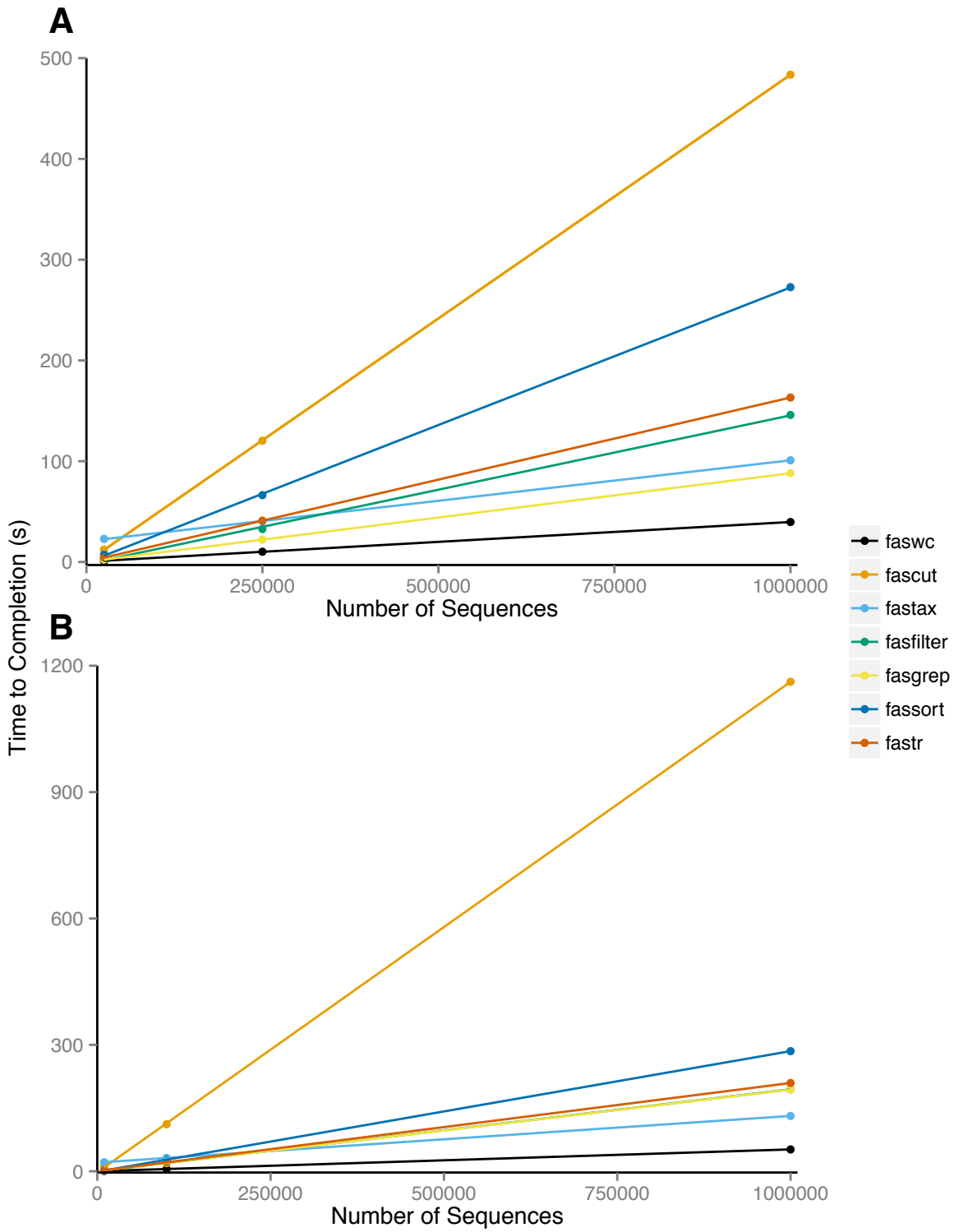

Figure 2. Average processor time of 100 repetitions required to complete analysis using indicated utility. Utilities were run on six datasets consisting of (a) 25000, 250000, and 1000000 100bp sequences and (b) 10000, 100000, and 1000000 1000bp sequences.