



# FAST: FAST Analysis of Sequences Toolbox

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

<sup>1</sup>Quantitative and Systems Biology Program, University of California, Merced, CA, USA

<sup>2</sup>Molecular Cell Biology Unit, School of Natural Sciences, University of California, Merced, CA, USA

<sup>3</sup>Dept. of Viticulture and Enology, University of California, Davis, CA, USA

<sup>4</sup>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) 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 simplify the documentation and reproducibility of bioinformatic protocols, supporting better transparency in biological data science. Interface self-consistency and conformity with conventions of GNU, Matlab, Perl, BioPerl, R and GenBank help make FAST easy and rewarding to learn. FAST automates numerical, taxonomic, and text-based sorting, selection and transformation of sequence records and alignment sites based on content, index ranges, descriptive tags, annotated features, and in-line calculated analytics, including 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 thanks to the relative maturity of its Perl and BioPerl foundations, with stable releases posted to CPAN. Development as well as a publicly accessible Cookbook and Wiki are available on the FAST GitHub repository at <https://github.com/tlawrence3/FAST>. 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. 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; Stothard, 2000; Rammpp

et al., 2006; **Librado and Rozas**, 2009; **Waterhouse et al.**, 2009; **Gouy et al.**, 2010). However, the monolithic application paradigm is easily outscaled by today's big biological data, particularly Next Generation Sequencing (NGS) data at gigabyte- and terabyte-scales. Better empowerment of non-programmers for genome-scale analytics of big biological data has been achieved through web-based genome browser interfaces (**Cunningham et al.**, 2015; **Rosenbloom et al.**, 2015; **Markowitz et al.**, 2014). 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 (**Knuth**, 1984) such as emacs org-mode (<http://orgmode.org>, as demonstrated in, for example **Delescluse et al.**, 2012) 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-throughput and high-dimensional measurements of complex biological systems (**Baggerly and Coombes**, 2009; **Hutson**, 2010; **Baggerly and Coombes**, 2011; **Huang and Gottardo**, 2013). 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 (**Ioannidis et al.**, 2008). Furthermore, even though analytical errors are a major source of retractions in the scientific literature (**Casadevall et al.**, 2014), 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 (**Peng**, 2009) facilitates crowd-sourced verification, correction and extension of code-based analyses (**Barnes**, 2010; **Morin et al.**, 2012), and reuse of software and data to enable more scientific discovery returns from public data (**Peng**, 2011). Peer review and publication of the data science protocols associated to scientific studies stems temptation to overinterpret results and encourages more objectivity in data science (**Boulesteix**, 2010). The ultimate remedy for these problems is to expand literacy in modern computational and statistical data science for science students in general (**Morin et al.**, 2012; **Joppa et al.**, 2013).

Web-based open-source workflow suites such as Galaxy (**Blankenberg and Hillman-Jackson**, 2014), Taverna (**Oinn et al.**, 2006) and BioExtract (**Lushbough et al.**, 2011) 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 bioinformatics specialists and programmers through Application Programming Interfaces (APIs) such as BioPerl and Ensembl (**Yates et al.**, 2015).

Yet workflow design suites and programming APIs require dedication and time to learn. There is a need for more bioinformatics software in between GUIs and APIs, that empowers non-programmer scientists and researchers to interactively and reproducibly control, process and analyze their data without manual interventions. Closer inspection of data and interactive construction and control of data workflows 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 here present FAST: FAST Analysis Sequences Toolbox, modeled after the standard Unix toolkit (**Peek**, 2001), now called Coreutils. The FAST tools follow the Unix philosophy to "do one thing and do it well" and "write programs to work together." (**Stutz**, 2000). FAST workflows are completely automated; no manual interventions to data are required. FAST falls between a GUI and an API, because it is used through a Command-Line Interface (CLI). Although the FAST tools are written in Perl using BioPerl packages (**Stajich et al.**, 2002), FAST users do not need to be able to program Perl or know BioPerl. FAST users only need basic competence in Unix and the modest skill to compose command pipelines in the Unix shell. FAST therefore supports an emerging movement to empower non-programmer biologists to learn Unix for scientific computing. Books and courses in this emerging market include the

76 recent “UNIX and Perl to the Rescue!” (Bradnam and Korf, 2012) and the Software Carpentry and Data  
77 Carpentry Foundations workshops (Wilson, 2014).

78 Unix command pipe-lines are the paradigmatic example of the “pipes and filters” design pattern that  
79 embodies serial processing of data through sequences of modular and reusable computations. The “pipes  
80 and filters” design pattern is a special case of component-based software engineering (Mcilroy, 1969) and  
81 a core paradigm in software architecture (Garlan and Shaw, 1994). The component-wise organization of  
82 FAST affords access to an infinite variety of customizable queries and workflows on biological sequence  
83 data using a small command vocabulary and combinatorial logic. Component-based software is easier to  
84 learn, maintain and extend. It also makes it easy for users to interactively develop new protocols through  
85 the modular extension and recombination of existing protocols. As shown from the examples below,  
86 non-trivial computations may be expressed on a single line of the printed page. Thus, FAST can help em-  
87 power non-biologist programmers to develop and communicate powerful and reproducible bioinformatic  
88 workflows for scientific investigations and publishing.

89 Open-source command-line utilities for bioinformatics such as the EMBOSS package (Rice et al.,  
90 2000), the FASTX tools (Gordon, 2009) or the scripts that come with BioPerl (Stajich et al., 2002) typi-  
91 cally offer suites of tools with simple, well-defined functions that lend themselves to scripting, but are not  
92 necessarily designed according to the Unix toolbox philosophy specifically to interoperate through serial  
93 composition over pipes. Similarly, FaBox (Villesen, 2007) is a free and open online server with functions  
94 that overlap with FAST tools, but is not designed for serial composition. On the other hand, the Unix  
95 toolbox model has been used before in more or less more specialized bioinformatics applications such as  
96 the popular SAMTools suite (Li et al., 2009) and in the processing of NMR data (Delaglio et al., 1995). A  
97 toolsuite called bp-utils, with a similar design philosophy and some overlapping functionality with FAST,  
98 has recently been released at <http://diverge.hunter.cuny.edu/labwiki/Bioutils>.

99 We have written extensive documentation for each FAST utility along with useful error messages fol-  
100 lowing recommended practice (Seemann, 2013). FAST is free and open source; its code is freely available  
101 to anyone to re-use, verify and extend through its GitHub repository.

## 2 DESIGN AND IMPLEMENTATION OF FAST TOOLS

### 2.1 THE FAST DATA MODEL

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

108 Another design feature of Unix tools that also characterizes the FAST tools is their ability to accept  
109 input not only from one or more files but also from what is called *standard input*, a data-stream supported  
110 by the Unix shell, and to output analogously to *standard output*. It is this facility that allows FAST  
111 tools to be serially composed in Unix *pipelines* that compactly represent an infinite variety of expressive  
112 bioinformatic workflows.

113 The default data exchange format for FAST tools is the universally recognized FastA format (Lipman  
114 and Pearson, 1985). While no universal standard exists for this format, for FAST, “FastA format”  
115 means what is conventionally called “multi-fasta” format of sequence or alignment data, largely as  
116 implemented in BioPerl in the module `Bio::SeqIO::fasta` (Stajich et al., 2002).

117 In the FAST implementation of FastA format, multiple sequence records may appear in a single file or  
118 input stream. Sequence data may contain gap characters. The logical elements (or fields) of a sequence  
119 record are its *identifier*, its *description* and its *sequence*. The identifier (indicated with `id` in the illustration

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 alter how description fields are defined in sequence records by using Perl-style *regular expressions* to define delimiters (indicated by `<delim>`). FAST uses one-based indexing of description fields.

The FAST data model is illustrated as follows:

```
>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 zero<sup>th</sup> 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 (see e.g. <https://www.sanger.ac.uk/resources/software/gff/>) or an alternative “name:value” format that certain FAST tools themselves can append to sequence records. Support for tagged values in FAST makes it possible to operate on sequence records with unordered or heterogeneous description fields.

## 2.2 OVERVIEW OF THE FAST TOOLS

FAST utilities may be assigned to categories according to their default behavior and intended use. There are FAST tools for **selection** of data from sequence records, **transformation** of data, **annotation** of sequence record descriptions with computed characteristics of the data, and **analysis**. A complete description of all utilities included in the first major release of FAST is shown in Table 1.

The **analysis** class is distinguished from the other classes because by default, these utilities output tables of plain-text data rather than sequence record data in FastA format. Two other tools, `fasconvert` and `gbfcut`, are designed to either input or output FastA format sequence records by default. Standardization of the FAST data model allows users to serially compose FAST tools into pipelines at the Unix command-line, which is indicated as the “main workflow” in the overview of the project shown in Figure 1.

## 2.3 GENERAL IMPLEMENTATION AND BENCHMARKING

The BioPerl backend of FAST 1.x 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.

**Table 1.** Utilities in first major release of FAST

Tool/Category	Function	Coreutil analog	Operates by default upon
<b>Selection</b>			
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	records
fastail	order-based selection of records	tail	records
fascut	index-based selection and reordering of data	cut	sequences
gbfcut	extract sequences by regex matching on features		features
alnucut	selection of sites by content		sites
gbfalncut	selection of sites by features		sites
<b>Transformation</b>			
fassort	numerical or text sorting of records	sort	identifiers
fastaxsort	taxonomic sorting of records		identifiers
fasuniq	remove or count redundant records	uniq	records
faspaste	merging of records	paste	sequences
fastr	character transformations on records	tr	identifiers
fassub	regex substitutions on records		identifiers
fasconvert	convert sequence formats		records
<b>Annotation</b>			
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
<b>Analysis</b>			
alnpi	molecular population genetic statistics		sites
faswc	tally sequences and characters	wc	sequences

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

163 Nearly all FAST utilities process sequence records inline and therefore have linear runtime complexity  
 164 in the number of sequences. Exceptions are *fassort* and *fastail* which both require some paging  
 165 of data into temporary files. We performed benchmarking of FAST tools using randomly generated se-  
 166 quences of even composition sourced generated in Python and the Benchmark v1.15 Perl module on a  
 167 MacBook Pro 2.5 Ghz Intel i7, with 8 Gb of RAM. We examined average CPU runtime over 100 repli-  
 168 cates, comparing input sizes of 25K, 250K, or 1M sequence records of length 100, 10K, 100K, or 1M  
 169 bp. Our benchmarking results show that despite data paging, *fassort* runtimes scale linearly with input  
 170 size (fig 2).

171 FAST is not designed to be fastest at computing its solutions. Rather the fastness of FAST lies in how  
 172 quickly an adept user can interactively prototype, develop, and express bioinformatic workflows with it.

## 2.4 INSTALLATION AND DEPENDENCIES

173 FAST requires a working Perl installation, with official releases distributed through the Comprehensive  
 174 Perl Archive Network (CPAN). A small footprint of BioPerl dependencies has been packaged together  
 175 in the FAST namespace. Other CPAN dependencies may be detected and installed by the *cpan* pack-  
 176 age manager. A fully automated install from CPAN may on many systems be initiated by executing  
 177 `perl -MCPAN -e 'install FAST'`. A manual install follows standard Perl install procedure.  
 178 After downloading and unpacking the source directory, change into that directory and execute: `perl`  
 179 `Makefile.PL; make; make test; (sudo) make install`.



180 We recommend that first-time users first complete the automated install from CPAN which will handle  
 181 prerequisites, and then download and open the source code directory in order to practice the example  
 182 usage commands (such as those in the sequel) on sample data provided within.

## 2.5 IMPLEMENTATION AND USAGE OF INDIVIDUAL TOOLS

183 Further implementation and usage details of individual FAST tools follows. Usage examples for indi-  
 184 vidual tools refer to example data that ships with the FAST source-code installer, available from CPAN.  
 185 The most recent version at the time of publication is 1.06, available from <http://search.cpan.org/~dhard/FAST-1.06/>. These usage examples should be able to run from within the installation  
 187 directory after installation has completed.

188 supports *regular expression*-based selection of sequence records. FAST uses Perl-style regular expres-  
 189 sions, which are documented freely online and within Perl, and are closely related to Unix extended  
 190 regular expressions. For reference on Perl regular expressions, try executing `man perlre` or `perldoc`  
 191 `perlre`. For example, to print only protein sequences that do *not* start with M for methionine, execute:

```
192     fasgrep -s -v "^M" t/data/P450.fas
```

193 In the above command the `-s` option directs `fasgrep` to search the sequence data of each record. The `-v`  
 194 option directs `fasgrep` to print records that *do not* match the pattern given by its argument, which is the  
 195 regular expression `^M`, in which the *anchor* `^` specifies the beginning of the sequence data. `fasgrep` uses  
 196 the BioPerl `Bio::Tools::SeqPattern` library to support ambiguity expansion of IUPAC codes in  
 197 its regular expression arguments. Thus, to show that a segment of *Saccharomyces cerevisiae* chromosome  
 198 1 contains at least one instance of an “Autonomous Consensus Sequences” characteristic of yeast origins  
 199 of replication (Leonard and Mchali, 2013), look whether the following command outputs a sequence or  
 200 not:

```
201     fasgrep -se 'WTTTAYRTTWT' t/data/chr01.fas
```

202 which is equivalent to:

```
203     fasgrep -se '[AT]TTTA[CT][AG]TTT[AT]' t/data/chr01.fas
```

204 These examples demonstrate queries on sequence data, but `fasgrep` may be directed to search against  
 205 other parts of sequence records including identifiers, descriptions, fields and more.

207 supports precise numerical-based selections of sequence records from numerical data in identifiers, de-  
 208 scriptions, fields or tagged-values in descriptions. `fasfilter` supports *open ranges* such as `100-`,  
 209 meaning “greater than or equal to 100”, closed ranges like `1e6-5e8` (meaning  $1 \times 10^6$  to  $5 \times 10^8$ ) and  
 210 compound ranges such as `200-400, 500-`. Ranges may be specified in Perl-style (or GenBank coordi-  
 211 nate style) like `from..to`, in R/Octave-style like `from:to` or UNIX `cut`-style as in `from-to`. For  
 212 example, to print records with `gi` numbers between 200 million and 500 million, try executing:

```
213     fasfilter -x "gi\\|(\d+)" 2e8..5e8 t/data/P450.fas
```

214 This example uses the `-x` option which directs `fasfilter` to filter on the value within the *capture*  
 215 *buffer* which occurs within the left-most pair of parentheses of the argument, here `(\d+)`, and `\d+` is a  
 216 regular expression matching a string of one or more digits from 0 to 9. The backslash after `gi` in the first  
 217 argument quotes the vertical bar character to make it literal, since the vertical bar character is a special  
 218 character in regular expressions.

219  
220

221 supports index-based selections of characters and fields in sequence records allowing repetition, reorder-  
222 ing, variable steps, and reversals. Ranges are specified otherwise similarly to *fasfilter*. Negative  
223 indices count backwards from last characters and fields. *fascut* outputs the concatenation of data selec-  
224 tions for each sequence record. Variable step-sizes in index ranges conveniently specify first, second or  
225 third codon positions in codon sequence records, for example. Examples using this syntax appear in the  
226 sequel. To print the last ten residues of each sequence, execute:

227 `fascut -10..-1 t/data/P450.fas`

228 implements content-based selection of sites in alignments including gap-free sites, non-allgap sites, vari-  
229 able or invariant sites and parsimoniously informative sites, or their set-complements, all with the option  
230 of state-frequency-thresholds applied per site. By default, *alnucut* prints only invariant sites. To print the  
231 set-complement or only variable sites, use the *-v* option:

232 `alnucut -v t/data/popset_32329588.fas`

233 To print sites in which no more than two sequences contain gaps, execute:

234 `alnucut -gf 2 t/data/popset_32329588.fas`

235 allows annotation-based sequence-extraction from GenBank format sequence files, useful for extracting  
236 all sequences that correspond to sets of the same type of annotated features in genome data. For example,  
237 to output 5' and 3' Untranslated Region (UTR) sequences from a GenBank formatted sequence of a gene,  
238 we use the *-k* option to restrict matching to features whose “keys” match the regular expression “UTR”:

239 `gbfcut -k UTR t/data/AF194338.1.gb`

240 *gbfcut* can handle split features such as a coding region (CDS) that is split over several exons:

241 `gbfcut -k CDS t/data/AF194338.1.gb`

242 More fine-grained queries of features are possible using qualifiers defined with the *-q* option. Multiple  
243 qualifiers may be provided at once, specifying the selection of records for which all qualifiers apply  
244 (conjunction). For example, compare the output of the following two commands:

245 `gbfcut -k tRNA t/data/mito-ascaris.gb`

246 `gbfcut -k tRNA -q product=Ser -q note^AGN t/data/mito-ascaris.gb`

247 The second command queries for features with key “tRNA” containing at least one qualifier “/product”  
248 whose value matches the string literal “Ser” and no qualifiers of type “/note” whose values match the  
249 string literal “AGN.”

250

251 automates the selection of sites from alignments that correspond to one or more features annotated on one  
252 of the sequences in a separate GenBank record. This workflow eliminates the need for manual entry of  
253 coordinates and implements a useful bioinformatic query in terms of known and reproducible quantities  
254 from public data and sequence records, allowing users to query sites based on biological vocabularies  
255 of sequence features. For an example of its use see the section “Composing Workflows in FAST” in the  
256 sequel.

257

concatenates data from records input in parallel from multiple data-streams or files, record-by-record. The user may paste data from the standard input stream and from multiple input files, in an order defined by the arguments. Records from standard input may be used multiple times in concatenating data. Like in some implementations of the Unix tool `paste`, a hyphen input argument `-` to `faspaste` refers to the standard input stream and may be used more than once as an input argument. For maximum configurability, `faspaste` concatenates only one data field type (*i.e.* sequences or descriptions) at a time. Users may select which data stream will provide templates to receive concatenated data in output records. For example, to paste sequences of corresponding records from two data-files together and output them with the identifiers and descriptions of the data in the first file, execute:

```
faspaste data1.fas data2.fas
```

See the sequel for more advanced usage examples with `faspaste`.

are designed to be often used together in Unix pipelines. The `fassort` utility implements numerical and textual sorting of sequence records by specific fields. The `fasuniq` utility removes (and optionally counts) records that are redundant with respect to a specific field, such as sequences or identifiers. In the implementation of `fassort`, 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`. Like its Unix Coreutil analog `uniq`, `fasuniq` compares only immediately successive input records. Therefore, users will usually want to first sort data with `fassort` before passing it to `fasuniq`. To illustrate, the following example combines and sorts input records from two instances of the same file, and then counts and removes each redundant record:

```
fassort -s t/data/P450.fas t/data/P450.fas | fasuniq -c
```

This example illustrates that the same file may be specified as an input stream more than once to any FAST command.

implement taxonomic searching and sorting of sequence records, whose records are already annotated with NCBI taxonomic identifiers using taxonomic data from NCBI taxonomy (Benson et al., 2009; Sayers et al., 2009). For example, a query of “Metazoa” would match records labeled “*Homo sapiens*,” “*Drosophila melanogaster*,” and “*Lepidoptera*” but not “*Candida albicans*” or “*Alphaproteobacteria*.” Taxonomic selections may be logically negated and/or restricted to only those records containing valid NCBI taxonomic identifiers. Purely for historical reasons, the internal implementation of NCBI taxonomic data is custom to FAST rather than the `Bio::Taxonomy` libraries in BioPerl. A sample of data from tRNADB-CE (Abe et al., 2014), in which data records are annotated with valid NCBI taxonomic identifiers in specific description fields, is included with the FAST installation package. After downloading datafiles “nodes.dmp” and “names.dmp” from NCBI Taxonomy, the following command filters sequences from Rhizobiales, assuming that records are labeled with their species (and strain) of origin in the third field of the description of the sample data file:

```
fastax -f 3 -S " \|" nodes.dmp names.dmp \
Rhizobiales t/data/tRNADB-CE.sample2000.fas
```

handle, respectively, character- and string-based transformations of sequence records. The utility `fastr` handles character-based transliterations, deletions and “squashing” (deletion of consecutive repeats), sequence degapping, and restriction or remapping of sequence data to strict or IUPAC ambiguity alphabets. For example, to lower-case all sequence characters, execute:

```
fastr -s 'A-Z' 'a-z' t/data/P450.fas
```



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		(Nei and Li, 1979)
Nucleotide Diversity	$\pi$	(Nei and Li, 1979)
Watterson estimator	$\theta_W$	(Watterson, 1975)
Expected number of alleles	$E(K)$	(Ewens, 1972)
Tajima's D	$D$	(Tajima, 1989)
Fu and Li's D*	$D^*$	(Fu and Li, 1993)
Fu and Li's F*	$F^*$	(Fu and Li, 1993; Simonsen et al., 1995)
Fu and Li's Eta S	$\eta_S$	(Fu and Li, 1993)
Fu and Li's Eta	$\eta$	(Fu and Li, 1993)

302 Degapping requires only the simple command:

```
303     fastr --degap t/data/P450.clustalw2.fas
```

304 The utility `fassub` allows more arbitrary substitutions on sets of strings matched to Perl regexes, imple-  
305 mented through direction of the Perl `s///` substitution operator on specific fields. Capture buffers may  
306 be used to refer to matched data in substitutions, for example, to reverse the order of genus and species in  
307 a file in which scientific names occur in descriptions enclosed with square brackets:

```
308     fassub -d '\[(\w+) (\w+)\]' '$2 $1' t/data/P450.fas
```

309 provide for annotation and analytics of compositions, translations, and codon usage frequencies of se-  
310 quence records (with start and stop codons counted distinctly, in the last case). All genetic codes included  
311 in BioPerl, ultimately from NCBI Entrez, are supported.  
312

313 outputs molecular population genetic statistics cited in Table 2 for each alignment on input. It can output  
314 a set of statistics for each alignment on input in plain text or  $\text{\LaTeX}$  format. `alnpi` also supports sliding  
315 window and pairwise analysis of input data. Data and command examples are provided to reproduce the  
316 tables and sliding window analyses of statistics published in (Ardell et al., 2003). Purely for historical  
317 reasons, `alnpi` does not use the perlymorphisms routines in the BioPerl library `Bio::PopGen` (Stajich  
318 and Hahn, 2005). However, all of the code for these calculations has been reviewed and compared against  
319 calculations produced from DNASP (Librado and Rozas, 2009) as described previously (Ardell, 2004).  
320

3 COMPOSING WORKFLOWS IN FAST

321 Here we show how to interactively prototype a pipeline that computes the sliding window profile of  
322 Tajima's  $D$  of Figure 4A in (Ardell et al., 2003) from a publicly available datafile. The datafile associated  
323 to this figure is an NCBI PopSet with accession ID 32329588 containing an alignment of a fully anno-  
324 tated ciliate gene (accession AF194338.1) against several partially sequenced allelic variants. One of the  
325 variants with accession ID AY243496.1 appears to be partly non-functionalized.

326 First to see this data, we view it in the pager `less` (press “q” to quit and “space” to page):

```
327 less t/data/popset_32329588.fas
```

328 A key feature of the Unix shell allows users to recall previous commands in their so-called *history*, usually  
329 by typing the “up-arrow” for possible re-use and editing. To check the number of sequences and characters  
330 in the alignment, execute:

```
331 faswc t/data/popset_32329588.fas
```

332 To compute our population genetic statistics we wish to remove the annotated reference sequence, the  
333 deactivated allele, and one potentially spurious additional haplotype from analysis, which we can do using  
334 `fasgrep`, and verify that it reduced data by the correct number of records (six) by piping to `faswc` (the  
335 command is broken over two lines here but may be entered as one line on the Unix prompt):

```
336 fasgrep -v "(AF194|349[06])" t/data/popset_32329588.fas \  
337 | faswc
```

338 We can check the identifier lines by modifying the end of this pipeline:

```
339 fasgrep -v "(AF194|349[06])" t/data/popset_32329588.fas \  
340 | grep \>
```

341 Sequencing ambiguities and gap-characters can introduce noise and uncertainty in the execution and  
342 documentation of bioinformatic workflows. For some computations, for example in molecular population  
343 genetics, one may want to be conservative and remove ambiguity- and gap- containing sites from an  
344 alignment. We can check for ambiguities in our data by outputting a composition table:

```
345 fasgrep -v "(AF194|349[06])" t/data/popset_32329588.fas \  
346 | fascomp --table
```

347 To remap ambiguities to gap characters, with the intent of removing all sites containing either ambiguities  
348 or gaps, we may use `fastr` to remap all non-strict DNA characters to gap (-) and verify the result using  
349 `fascomp` again:

```
350 fasgrep -v "(AF194|349[06])" t/data/popset_32329588.fas \  
351 | fastr --strict -N - | fascomp --table
```

352 Now, with confidence in our filtered data, we can confidently extract only gap-free sites from the alignment  
353 using `alncut`, and verify that we reduced the size of the alignment with `faswc`:

```
354 fasgrep -v "(AF194|349[06])" t/data/popset_32329588.fas \  
355 | fastr --strict -N - | alncut -g | faswc
```

356 Finally, with confidence in the integrity of our pipeline developed so far, we pass the latest output to  
357 `alnpi` for sliding-window analysis of Tajima’s D:

```
358 fasgrep -v "(AF194|349[06])" t/data/popset_32329588.fas \  
359 | fastr --strict -N - | alncut -g | alnpi --window 100:25:d
```

## 4 FURTHER FAST WORKFLOW EXAMPLES

### 4.1 SELECTING SITES FROM ALIGNMENTS BY ANNOTATED FEATURES

Another example, that reproduces a published result from (Ardell et al., 2003), demonstrates the utility of combining `gbfalncut` with `alnpi`, allowing users to select sites from alignments corresponding to features annotated on one of the sequences in a separate GenBank file. For example, to calculate a Tajima's *D* statistic for 5' UTRs, corresponding to the the last line in Table 1 of that work, execute:

```
364 gbfaIncut -k t/data/ArdellEtAl03_ncbi_popset_32329588.fas \
365 t/data/AF194338.1.gb 5.UTR | fastgrep -v "(AF194|349[06])" \
366 | fastr --strict -N - | alncut -g | alnpi
```

### 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 (Kornfeld and Kornfeld, 1985), one could execute:

```
371 fasxl -a cds.fas | fastgrep -t x10 "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 “x10” (indicating translation in the zeroth reading frame). The second command in the pipeline uses a regular expression to represent the *N*-glycosylation amino acid motif pattern as the value of a “name:value” pair in the description with tag “x10”, hence processing the annotations produced by `fasxl`. The regex argument to `fastgrep` 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 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:

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

### 4.4 MORE ADVANCED MERGING OF DATA RECORDS

More advanced usage of `fastpaste` requires Unix pipelines. For example to join **both** descriptions and sequences from two data-files, execute:

```
386 fastpaste data1.fas data2.fas | fastpaste -d - data2.fas
```

The hyphen second argument (-) to the second instance of `fastpaste` refers to the input received from standard input through the pipe. This example works because by default, `fastpaste` uses (“mutates”) records from the data stream named in its first argument to receive the data concatenated from all records.

To prepend the first sequence of one file repeatedly to every sequence in another file, execute:

```
391 fashead -n 1 t/data/fasxl_test4.fas \
```

```
392 | faspaste -r - t/data/fasxl_test4.fas
```

393 To prepend the first sequence of one file repeatedly to every other sequence in another file, using identifiers  
394 and descriptions from the second file in the output, execute:

```
395 fashead -n 1 t/data/fasxl_test3.fas \  
396 | faspaste -r -R 2 - t/data/fasxl_test4.fas
```

## 5 FURTHER DOCUMENTATION AND USAGE EXAMPLES

397 Upon installation, FAST generates and installs a complete man page for each FAST utility, which should  
398 be accessible by one or both of the following commands, *e.g.*:

```
399 man fasgrep  
400 perldoc fasgrep
```

401 In addition, a FAST Cookbook has been contributed by the authors and is available with the source code  
402 distribution or from the project GitHub repository at <https://github.com/tlawrence3/FAST>.

## 6 CONCLUDING REMARKS AND FUTURE DIRECTIONS

403 Planned additions in future versions of FAST include `fasrand` and `alnrand` for automated sampling,  
404 permutations and bootstrapping of sequences and sites, respectively, and `fasgo` and `fasgosort` for  
405 selection and sorting of records by Gene Ontology categories (The Gene Ontology Consortium, 2015).

## AVAILABILITY

406 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 repository at <https://github.com/tlawrence3/FAST>. For latest news on the FAST project please check the Ardell  
407 Lab homepage at <http://compbio.ucmerced.edu/ardell/software/FAST/>.

## DISCLOSURE/CONFLICT-OF-INTEREST STATEMENT

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

## AUTHOR CONTRIBUTIONS

412 D.H.A. conceived, designed, and wrote much of FAST. T.J.L. contributed major code factorizations and  
413 reorganization and `fastail`. K.T.K. contributed code including `faspaste`, and `fashead`. R.S.L.  
414 contributed an analysis of code dependencies for the FAST installer. P.J.B. tested installation and running  
415 on Windows using Strawberry Perl. All authors, especially D.L.C. and C.J.C., contributed documenta-  
416 tion, testing, and code fixes. K.C.H.A. and D.H.A. wrote the FAST Cookbook. D.H.A. wrote the paper  
417 with major contributions from D.L.C. and T.J.L. All authors made minor contributions to the manuscript,  
418 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, as well as Julie Phillips and other Ardell lab members or students who used these tools and gave feedback. 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, an AY2009-2010 award from UC Merced's Graduate Research Council, a Chancellor's Award from UC Merced's second Chancellor Sung-Mo Kang, the NSF-funded URM program Undergraduate Research in Computational Biology at UC Merced (DBI-1040962) for support of K.T.K. and P.J.B., and support of the UC Merced Library Open Access Fund Work Group.

## REFERENCES

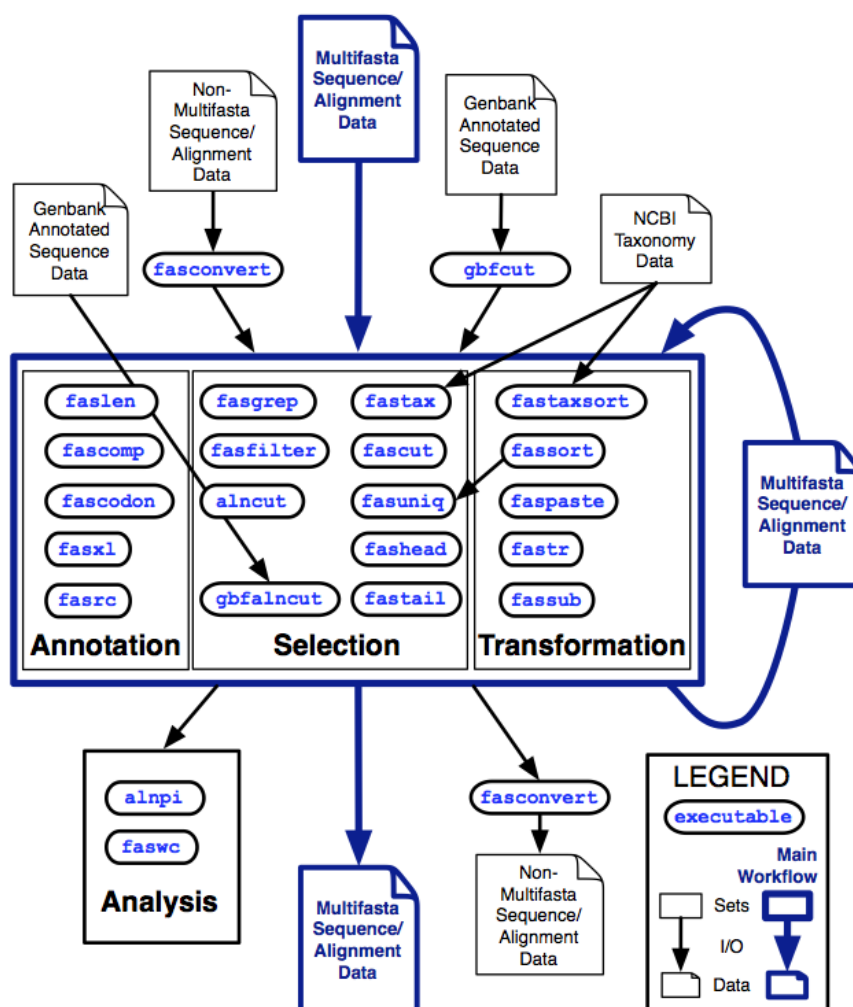
- Abe, T., Inokuchi, H., Yamada, Y., Muto, A., Iwasaki, Y., and Ikemura, T. (2014), tRNADB-CE: tRNA gene database well-timed in the era of big sequence data, *Frontiers in Genetics*, 5, 114, doi:10.3389/fgene.2014.00114
- Ardell, D. H. (2004), SCANMS: adjusting for multiple comparisons in sliding window neutrality tests, *Bioinformatics*, 20, 12, 1986–1988, doi:10.1093/bioinformatics/bth187
- Ardell, D. H., Lozupone, C. A., and Landweber, L. F. (2003), Polymorphism, recombination and alternative unscrambling in the DNA polymerase alpha gene of the ciliate *stylonychia lemnae* (*alveolata*; class *spirotrichea*), *Genetics*, 165, 4, 1761–1777
- Baggerly, K. A. and Coombes, K. R. (2009), Deriving chemosensitivity from cell lines: Forensic bioinformatics and reproducible research in high-throughput biology, *The Annals of Applied Statistics*, 3, 4, pp. 1309–1334
- Baggerly, K. A. and Coombes, K. R. (2011), What information should be required to support clinical omics publications?, *Clinical Chemistry*, 57, 5, 688–690, doi:10.1373/clinchem.2010.158618
- Barnes, N. (2010), Publish your computer code: it is good enough, *Nature*, 467, 7317, 753–753
- 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
- Blankenberg, D. and Hillman-Jackson, J. (2014), Analysis of next-generation sequencing data using Galaxy, in B. L. Kidder, ed., Stem Cell Transcriptional Networks, volume 1150 of *Methods in Molecular Biology* (Springer New York), 21–43, doi:10.1007/978-1-4939-0512-6\_2
- Boulesteix, A.-L. (2010), Over-optimism in bioinformatics research, *Bioinformatics*, 26, 3, 437–439, doi:10.1093/bioinformatics/btp648
- Bradnam, K. and Korf, I. (2012), UNIX and Perl to the Rescue!: A Field Guide for the Life Sciences (and Other Data-rich Pursuits) (Cambridge University Press)
- Casadevall, A., Steen, R. G., and Fang, F. C. (2014), Sources of error in the retracted scientific literature, *The FASEB Journal*, 28, 9, 3847–3855, doi:10.1096/fj.14-256735
- Cunningham, F., Amode, M. R., Barrell, D., Beal, K., Billis, K., Brent, S., et al. (2015), Ensembl 2015, *Nucleic Acids Research*, 43, D1, D662–D669, doi:10.1093/nar/gku1010
- Delaglio, F., Grzesiek, S., Vuister, G. W., Zhu, G., Pfeifer, J., and Bax, A. (1995), NMRPipe: a multidimensional spectral processing system based on unix pipes, *Journal of Biomolecular NMR*, 6, 3, 277–293
- Delescluse, M., Franconville, R., Joucla, S., Lieury, T., and Pouzat, C. (2012), Making neurophysiological data analysis reproducible: why and how?, *Journal of Physiology, Paris*, 106, 3-4, 159–170, doi:10.1016/j.jphysparis.2011.09.011
- Ewens, W. J. (1972), The sampling theory of selectively neutral alleles, *Theoretical population biology*, 3, 1, 87–112
- Fu, Y. X. and Li, W. H. (1993), Statistical tests of neutrality of mutations., *Genetics*, 133, 3, 693–709



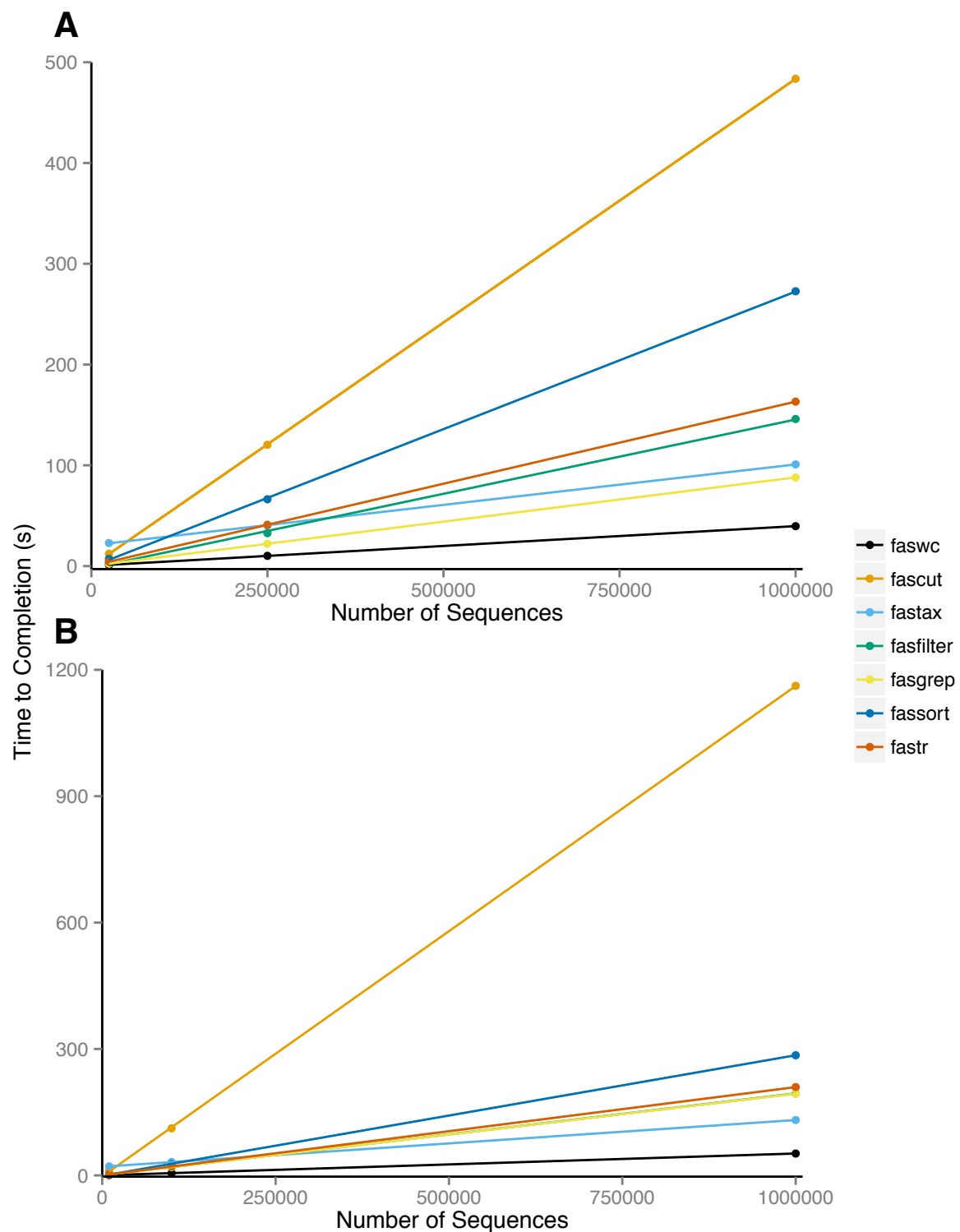
- Garlan, D. and Shaw, M. (1994), An Introduction to Software Architecture, *Computer Science Department*
- Gordon, A. (2009), FASTX Toolkit, [http://cancan.cshl.edu/labmembers/gordon/fastx\\_toolkit/index.html](http://cancan.cshl.edu/labmembers/gordon/fastx_toolkit/index.html), [Online; accessed 25-January-2015]
- Gouy, M., Guindon, S., and Gascuel, O. (2010), SeaView version 4: a multiplatform graphical user interface for sequence alignment and phylogenetic tree building, *Molecular biology and evolution*, 27, 2, 221–224
- Huang, Y. and Gottardo, R. (2013), Comparability and reproducibility of biomedical data, *Briefings in Bioinformatics*, 14, 4, 391–401, doi:10.1093/bib/bbs078
- Hutson, S. (2010), Data handling errors spur debate over clinical trial, *Nature medicine*, 16, 6, 618
- Ioannidis, J. P. A., Allison, D. B., Ball, C. A., Coulibaly, I., Cui, X., Culhane, A. C., et al. (2008), Repeatability of published microarray gene expression analyses, *Nat Genet*, 41, 2, 149–155
- Joppa, L. N., McInerny, G., Harper, R., Salido, L., Takeda, K., O'Hara, K., et al. (2013), Troubling trends in scientific software use, *Science*, 340, 6134, 814–815, doi:10.1126/science.1231535
- Knuth, D. E. (1984), Literate programming, *The Computer Journal*, 27, 2, 97–111
- Kornfeld, R. and Kornfeld, S. (1985), Assembly of asparagine-linked oligosaccharides, *Annual Review of Biochemistry*, 54, 1, 631–664, doi:10.1146/annurev.bi.54.070185.003215, PMID: 3896128
- Leonard, A. C. and Mchali, M. (2013), DNA Replication Origins, *Cold Spring Harbor Perspectives in Biology*, 5, 10, a010116, doi:10.1101/cshperspect.a010116
- Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., et al. (2009), The sequence alignment/map format and SAMtools, *Bioinformatics*, 25, 16, 2078–2079, doi:10.1093/bioinformatics/btp352
- Librado, P. and Rozas, J. (2009), DnaSP v5: a software for comprehensive analysis of DNA polymorphism data, *Bioinformatics*, 25, 11, 1451–1452, doi:10.1093/bioinformatics/btp187
- Lipman, D. J. and Pearson, W. R. (1985), Rapid and sensitive protein similarity searches, *Science*, 227, 4693, 1435–1441
- Lushbough, C. M., Jennewein, D. M., and Brendel, V. P. (2011), The bioextract server: a web-based bioinformatic workflow platform, *Nucleic Acids Research*, 39, suppl 2, W528–W532, doi:10.1093/nar/gkr286
- Markowitz, V. M., Chen, I.-M. A., Palaniappan, K., Chu, K., Szeto, E., Pillay, M., et al. (2014), IMG 4 version of the integrated microbial genomes comparative analysis system, *Nucleic Acids Research*, 42, D1, D560–D567, doi:10.1093/nar/gkt963
- Mcilroy, D. (1969), Mass-produced Software Components, in J. Buxton, P. Naur, and B. Randell, eds., *Proceedings of the 1st International Conference on Software Engineering*, Garmisch Pattenkirchen, Germany, 138–155
- Morin, A., Urban, J., Adams, P. D., Foster, I., Sali, A., Baker, D., et al. (2012), Shining light into black boxes, *Science*, 336, 6078, 159–160, doi:10.1126/science.1218263
- Nei, M. and Li, W. H. (1979), Mathematical model for studying genetic variation in terms of restriction endonucleases., *Proc Natl Acad Sci U S A*, 76, 10, 5269–5273
- Oinn, T., Greenwood, M., Addis, M., Alpdemir, M. N., Ferris, J., Glover, K., et al. (2006), Taverna: lessons in creating a workflow environment for the life sciences, *Concurrency and Computation: Practice and Experience*, 18, 10, 1067–1100, doi:10.1002/cpe.993
- Peek, J. (2001), Why Use a Command Line Instead of Windows?, <http://www.linuxdevcenter.com/pub/a/linux/2001/11/15/learnunixos.html>
- Peng, R. D. (2009), Reproducible research and biostatistics, *Biostatistics*, 10, 3, 405–408, doi:10.1093/biostatistics/kxp014
- Peng, R. D. (2011), Reproducible research in computational science, *Science*, 334, 6060, 1226–1227, doi:10.1126/science.1213847
- 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)00204-2
- Rosenbloom, K. R., Armstrong, J., Barber, G. P., Casper, J., Clawson, H., Diekhans, M., et al. (2015), The UCSC Genome Browser database: 2015 update, *Nucleic Acids Research*, 43, D1, D670–D681, doi:10.1093/nar/gku1177
- 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
- Simonsen, K. L., Churchill, G. A., and Aquadro, C. F. (1995), Properties of statistical tests of neutrality for DNA polymorphism data., *Genetics*, 141, 1, 413–429
- 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
- Stajich, J. E. and Hahn, M. W. (2005), Disentangling the effects of demography and selection in human history, *Molecular Biology and Evolution*, 22, 1, 63–73, doi:10.1093/molbev/msh252
- Stothard, P. (2000), The sequence manipulation suite: JavaScript programs for analyzing and formatting protein and DNA sequences, *BioTechniques*, 28, 6, 1102, 1104
- Stutz, M. (2000), Linux and the Tools Philosophy, <http://www.linuxdevcenter.com/pub/a/linux/2000/07/25/LivingLinux.html>
- Tajima, F. (1989), Statistical method for testing the neutral mutation hypothesis by DNA polymorphism., *Genetics*, 123, 3, 585–595
- The Gene Ontology Consortium (2015), Gene ontology consortium: going forward, *Nucleic Acids Research*, 43, D1, D1049–D1056, doi:10.1093/nar/gku1179
- Villesen, P. (2007), FaBox: an online toolbox for fasta sequences, *Molecular Ecology Notes*, 7, 6, 965–968, doi:10.1111/j.1471-8286.2007.01821.x
- Waterhouse, A. M., Procter, J. B., Martin, D. M. A., Clamp, M., and Barton, G. J. (2009), Jalview Version 2—a multiple sequence alignment editor and analysis workbench, *Bioinformatics (Oxford, England)*, 25, 9, 1189–1191, doi:10.1093/bioinformatics/btp033
- Watterson, G. (1975), On the number of segregating sites in genetical models without recombination, *Theoretical population biology*, 7, 2, 256–276
- Wilson, G. (2014), Software Carpentry: lessons learned, *F1000Research*, doi:10.12688/f1000research.3-62.v1
- Yates, A., Beal, K., Keenan, S., McLaren, W., Pignatelli, M., Ritchie, G. R. S., et al. (2015), The Ensembl REST API: ensembl data for any language, *Bioinformatics*, 31, 1, 143–145, doi:10.1093/bioinformatics/btu613

## FIGURES



**Figure 1.** Overview of the first major release of FAST with data and workflow dependencies indicated. Inputs to FAST tools are shown at the top of the figure with outputs at the bottom. Outlined in blue is the primary working model, in which Multifasta sequence or alignment data is successively annotated, selected upon and transformed into new Multifasta sequence alignment data, or fed into a utility in the **analysis** category for tabular output of data summaries. Many of the utilities in the **annotation** category are also optionally capable of tabular output.



**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.