



FAST: FAST Analysis of Sequences Toolbox

Travis J. Lawrence¹, Kyle T. Kauffman², Katherine C.H. Amrine^{1,3}, Katherine C.H. Amrine^{1,3}, Dana L. Carper¹, Raymond S. Lee⁴, Peter J. Becich², Claudia J. Canales⁴ and David H. Ardell^{1,2*}

¹Quantitative and Systems Biology Program, University of California, Merced, CA, USA

²Molecular Cell Biology Unit, School of Natural Sciences, University of California, Merced, CA, USA

³Dept. of Viticulture and Enology, University of California, Davis, CA, USA

⁴School 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` are designed for rapid prototyping of expressive bioinformatic workflows in a compact and generic command vocabulary. FAST is intended to bring the analytic power of Perl and BioPerl to non-programmers and extend the UNIX tools paradigm to popular bioinformatics data formats. Compact combinatorial encoding of data workflows with FAST 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, 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; Ramp et al., 2006;

26 **Librado and Rozas**, 2009; **Gouy et al.**, 2010). However, today, the monolithic application paradigm
27 can be easily outscaled by big biological data, particularly Next Generation Sequencing (NGS) data at gi-
28 gabyte and terabyte-scale. Better empowerment of non-programmers for genome-scale analytics has been
29 achieved through web-based genome browser interfaces (**Cunningham et al.**, 2015; **Rosenbloom et al.**,
30 2015; **Markowitz et al.**, 2014). On the other hand, for smaller datasets, sequence and alignment editor
31 applications encourage manual manipulation of data, which is error-prone and essentially irreproducible.
32 To reduce error and increase reproducibility in the publishing of bioinformatic and biostatistical protocols
33 it is important to facilitate the documentation and automation of data science workflows through scripts
34 and literate programming facilities (**Knuth**, 1984) that both completely document and encode scientific
35 workflows for machine processing of biological data.

36 Reproducibility in bioinformatics and biostatistics protocols is crucial to maintaining public trust in
37 the value of its investments in high-dimensional analysis of complex biological systems (**Baggerly and**
38 **Coombes**, 2009; **Hutson**, 2010; **Baggerly and Coombes**, 2011; **Huang and Gottardo**, 2013). In one
39 analysis, only two of 18 published microarray gene-expression analyses were completely reproducible,
40 in part because key analysis steps were made with proprietary closed-source software (**Ioannidis et al.**,
41 2008). Furthermore, even though analytical errors are a major source of retractions in the scientific lit-
42 erature (**Casadevall et al.**, 2014), peer-review and publication of scientific data processing protocols is
43 generally not yet required to publish scientific studies. Adequate documentation of bioinformatic and
44 biostatistical workflows and open source sharing of code upon publication (**Peng**, 2009) facilitates crowd-
45 sourced verification, correction and extension of code-based analyses (**Barnes**, 2010; **Morin et al.**, 2012),
46 and reuse of software and data to enable more scientific discovery returns from public data (**Peng**, 2011).
47 Review and publication of open data science protocols may also help reduce temptations to overinterpret
48 data and encourage more objectivity in data science (**Boulesteix**, 2010), although perhaps it is expanded
49 computational and statistical literacy and training for all scientists that is the ultimate remedy for these
50 problems (**Morin et al.**, 2012; **Joppa et al.**, 2013).

51 Web-based open-source workflow suites such as Galaxy (**Blankenberg and Hillman-Jackson**, 2014),
52 Taverna (**Oinn et al.**, 2006) and BioExtract (**Lushbough et al.**, 2011) are a recent innovation in the direc-
53 tion of greater reproducibility in bioinformatics protocols for genome-scale analytics. However, the most
54 powerful, transparent and customizable medium for reproducible bioinformatics work is only available
55 to specialists through programming Application Programming Interfaces (APIs) such as BioPerl and En-
56 sembl (**Yates et al.**, 2015). Both workflow design suites and programming APIs require dedication and
57 time to learn.

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

64 In this tradition we present FAST: FAST Analysis Sequences Toolbox, modeled after the standard Unix
65 toolkit (**Peek**, 2001), now called Coreutils. The FAST tools follow the Unix philosophy to “do one thing
66 and do it well” and “write programs to work together.” (**Stutz**, 2000). Command-line utilities for bioinfor-
67 matics such as the EMBOSS package (**Rice et al.**, 2000), the FASTX tools (**Gordon**, 2009) or the scripts
68 that come with BioPerl (**Stajich et al.**, 2002) typically offer suites of tools with simple, well-defined func-
69 tions that lend themselves to scripting, but are not necessarily designed according to the UNIX toolbox
70 philosophy specifically to interoperate through serial composition. Similarly, FaBox is a free and open
71 online server with functions that overlap with FAST tools, but is not designed for serial composition. On
72 the other hand, the UNIX toolbox model has been used before in more or less more specialized bioinfor-
73 matics applications such as the popular SAMTools suite (**Li et al.**, 2009) and in the processing of NMR
74 data (**Delaglio et al.**, 1995).

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
fastpaste	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

75 The FAST tools are written in Perl using BioPerl packages (Stajich et al., 2002). This makes FAST
76 utilities easy to adopt if you are familiar with the Unix toolbox and allows fast sequence analysis even on
77 large datasets. Extensive documentation has been developed for each FAST utility along with useful error
78 messages following recommended practice (Seemann, 2013). FAST is free and open source, available to
79 anyone for re-use, verification and extension. This is in line with the call to make science generally more
80 accessible, open, and reproducible by other scientists and the public (Groves and Godlee, 2012).

2 DESIGN

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

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

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

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

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

```
111 >seq1-id<space>seq1-desc-field1<delim>seq1-desc-field2<delim>...
112 seq1-sequence
113 seq1-sequence
114 ...
115 seq1-sequence
116 >seq2-id<space>seq2-desc-field1<delim>seq2-desc-field2<delim>...
117 seq2-sequence
118 seq2-sequence
119 ...
120 seq2-sequence
```

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

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

3 IMPLEMENTATION DETAILS AND BENCHMARKING

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

137 The BioPerl backend of FAST 1.0 is version 1.6.901 downloaded in January, 2012. Bio::SeqIO com-
 138 ponents were updated to version 1.6.923 on June 4, 2014 and some Bio::Root components were updated
 139 on July 10, 2014 (github commit 50f87e9a4d). We introduced a small number of customizations to the

140 BioPerl code-base, primarily to enable the translation of sequences containing gaps. All of the BioPerl
141 dependencies of FAST are isolated under its own FAST name-space.

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

144 Further implementation details of individual FAST tools follows.

3.1 FASGREP

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

3.2 FASFILTER

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

3.3 FASCUT

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

3.4 ALNCUT

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

3.5 GBFCUT

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

3.6 FASSORT

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

3.7 FASUNIQ

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

3.8 FASTAX AND FASTAXSORT

171 Taxonomic searching and sorting of sequence records, when those records are already annotated with
172 NCBI taxonomic identifiers, is enabled by the FAST tools `fastax` and `fastaxsort` using taxonomic
173 data from NCBI taxonomy (Benson et al., 2009; Sayers et al., 2009). Taxonomic selections may be
174 logically negated and/or restricted to valid NCBI taxonomic identifiers. A sample of data from tRNADB-
175 CE (Abe et al., 2014), which includes taxonomic names from NCBI Taxonomy, is included with the
176 package and referred to in the documentation, showing how these utilities may be used.

3.9 FASTR AND FASSUB

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

3.10 FASCOMP, FASXL AND FASCODON

182 Annotation and analytics of compositions, translations, and codon usage frequencies of sequence records
183 (with start and stop codons counted distinctly, in the last case). All genetic codes included in BioPerl,
184 ultimately from NCBI Entrez, are supported.

3.11 ALNPI

185 `alnpi` outputs molecular population genetic statistics cited in Table 2 for each alignment on input. It can
186 output a set of statistics for each alignment on input in plain text or L^AT_EX format. `alnpi` also supports
187 sliding window and pairwise analysis of input data. This code computed the sliding window analyses
188 published in (Ardell et al., 2003). All of the code for these calculations has been reviewed and compared
189 against calculations produced from DNASP (Librado and Rozas, 2009) as described previously (Ardell,
190 2004).

4 INSTALLATION AND USAGE EXAMPLES

4.1 INSTALLATION AND DEPENDENCIES

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

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	Π	(Nei and Li, 1979)
Nucleotide Diversity	π	(Nei and Li, 1979)
Watterson estimator	θ_W	(Watterson, 1975)
Expected number of alleles	$E(K)$	(Ewens, 1972)
Tajima's D	D	(Tajima, 1989)
Fu and Li's F	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	η_S	(Fu and Li, 1993)
Fu and Li's Eta	η	(Fu and Li, 1993)

4.2 TEST SUITE AND DOCUMENTATION

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

4.3 SELECTING SEQUENCES BY ENCODED MOTIFS

199 An advantage of the annotation approach in FAST is the ability to select and sort sequences by at-
200 tributes computed and annotated into data by utilities upstream in the pipeline. For example, to select
201 protein-coding genes from a file `cds.fas` whose translations contain the *N*-glycosylation amino acid
202 motif (Kornfeld and Kornfeld, 1985), one could execute:

```
203 fasx1 -a cds.fas | fasgrep -t x10 "N[^P][ST][^P]" | fascut -f 1..-2
```

204 The first command in the pipeline translates each sequence and appends the translation to the description
205 with the tag “x10” (indicating translation in the zeroth reading frame). The second command in the pipeline
206 does a regex match using the specified motif pattern on the value of a “name:value” pair in the description
207 with tag “x10”, hence processing the annotations produced by `fasx1`. The regex argument to `fasgrep` is
208 quoted to protect the argument from interpretation by the shell. The last command in the pipeline removes
209 the last field in the description, restoring records as they were before they were annotated by `fasx1`.

4.4 SORTING RECORDS BY THIRD CODON POSITION COMPOSITION

210 Another example illustrates the powerful expression of ranges in `fascut`. An optional “by” parameter
211 in ranges allows increments or decrements in steps larger than one. To extract third-position bases from
212 codon sequence records, compute and annotate their compositions into record descriptions, ultimately
213 sorting records by their third-position adenosine contents, do:

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

5 CONCLUDING REMARKS AND FUTURE DIRECTIONS

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

AVAILABILITY

218 Stable versions of FAST are released through the Comprehensive Perl Archive Network (CPAN) at
219 <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
220 Lab homepage at <http://compbio.ucmerced.edu/ardell/software/FAST/>.
221

DISCLOSURE/CONFLICT-OF-INTEREST STATEMENT

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

AUTHOR CONTRIBUTIONS

224 D.H.A. conceived, designed, and wrote much of FAST. T.J.L. contributed major code factorizations and
225 reorganization and `fastail`. K.T.K. contributed code including `faspaste`, and `fashead`. R.S.L.
226 contributed an analysis of code dependencies for the FAST installer. P.J.B. tested installation and running
227 on Windows using Strawberry Perl. All authors, especially D.L.C. and C.J.C., contributed documenta-
228 tion, testing, and code fixes. K.C.H.A. and D.H.A. wrote the FAST Cookbook. D.H.A. wrote the paper
229 with major contributions from D.L.C. and T.J.L. All authors made minor contributions to the manuscript,
230 reviewed the final version of the manuscript and agree to be accountable for its contents.

ACKNOWLEDGEMENT

231 We acknowledge Christopher Clark for help in establishing a git repository for FAST and Peter Becich for
232 contributing some testing of the installer. D.H.A. gratefully acknowledges Professors Laura Landweber,
233 Siv Andersson and Leif Kirsebom in whose laboratories the FAST tools were first developed as well as
234 the Linnaeus Centre for Bioinformatics at Uppsala University.

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

REFERENCES

- 238 Abe, T., Inokuchi, H., Yamada, Y., Muto, A., Iwasaki, Y., and Ikemura, T. (2014), trnadb-ce: trna gene
239 database well-timed in the era of big sequence data, *Frontiers in Genetics*, 5, 114, doi:10.3389/fgene.
240 2014.00114
241 Ardell, D. H. (2004), Scanms: adjusting for multiple comparisons in sliding window neutrality tests,
242 *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
- 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
- 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
- Gordon, A. (2009), FASTX Toolkit, 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
- Groves, T. and Godlee, F. (2012), Open science and reproducible research., *BMJ (Clinical research ed.)*, 344, jun26_1, e4383, doi:10.1136/bmj.e4383
- 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
- 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

- 293 Lushbough, C. M., Jennewein, D. M., and Brendel, V. P. (2011), The bioextract server: a web-based
294 bioinformatic workflow platform, *Nucleic Acids Research*, 39, suppl 2, W528–W532, doi:10.1093/nar/
295 gkr286
- 296 Markowitz, V. M., Chen, I.-M. A., Palaniappan, K., Chu, K., Szeto, E., Pillay, M., et al. (2014), Img 4
297 version of the integrated microbial genomes comparative analysis system, *Nucleic Acids Research*, 42,
298 D1, D560–D567, doi:10.1093/nar/gkt963
- 299 Morin, A., Urban, J., Adams, P. D., Foster, I., Sali, A., Baker, D., et al. (2012), Shining light into black
300 boxes, *Science*, 336, 6078, 159–160, doi:10.1126/science.1218263
- 301 Nei, M. and Li, W. H. (1979), Mathematical model for studying genetic variation in terms of restriction
302 endonucleases., *Proc Natl Acad Sci U S A*, 76, 10, 5269–5273
- 303 Oinn, T., Greenwood, M., Addis, M., Alpdemir, M. N., Ferris, J., Glover, K., et al. (2006), Tave-
304 rna: lessons in creating a workflow environment for the life sciences, *Concurrency and Computation:
305 Practice and Experience*, 18, 10, 1067–1100, doi:10.1002/cpe.993
- 306 Peek, J. (2001), Why Use a Command Line Instead of Windows?, <http://www.linuxdevcenter.com/pub/a/linux/2001/11/15/learnunixos.html>
307
- 308 Peng, R. D. (2009), Reproducible research and biostatistics, *Biostatistics*, 10, 3, 405–408, doi:10.1093/
309 biostatistics/kxp014
- 310 Peng, R. D. (2011), Reproducible research in computational science, *Science*, 334, 6060, 1226–1227,
311 doi:10.1126/science.1213847
- 312 Rampp, M., Soddemann, T., and Lederer, H. (2006), The MIGenAS integrated bioinformatics toolkit for
313 web-based sequence analysis., *Nucleic acids research*, 34, Web Server issue, W15–9, doi:10.1093/nar/
314 gkl254
- 315 Rice, P., Longden, I., and Bleasby, A. (2000), EMBOSS: The European Molecular Biology Open Software
316 Suite, *Trends in Genetics*, 16, 6, 276–277, doi:10.1016/S0168-9525(00)02024-2
- 317 Rosenbloom, K. R., Armstrong, J., Barber, G. P., Casper, J., Clawson, H., Diekhans, M., et al. (2015),
318 The ucsc genome browser database: 2015 update, *Nucleic Acids Research*, 43, D1, D670–D681, doi:10.
319 1093/nar/gku1177
- 320 Sayers, E. W., Barrett, T., Benson, D. A., Bryant, S. H., Canese, K., Chetvernin, V., et al. (2009), Database
321 resources of the National Center for Biotechnology Information., *Nucleic acids research*, 37, Database
322 issue, D5–15, doi:10.1093/nar/gkn741
- 323 Seemann, T. (2013), Ten recommendations for creating usable bioinformatics command line software.,
324 *GigaScience*, 2, 1, 15, doi:10.1186/2047-217X-2-15
- 325 Simonsen, K. L., Churchill, G. A., and Aquadro, C. F. (1995), Properties of statistical tests of neutrality
326 for DNA polymorphism data., *Genetics*, 141, 1, 413–429
- 327 Smith, S. W., Overbeek, R., Woese, C. R., Gilbert, W., and Gillevet, P. M. (1994), The genetic data envi-
328 ronment an expandable GUI for multiple sequence analysis., *Computer applications in the biosciences*
329 : *CABIOS*, 10, 6, 671–5
- 330 Stajich, J. E., Block, D., Boulez, K., Brenner, S. E., Chervitz, S. A., Dagdigian, C., et al. (2002), The
331 Bioperl toolkit: Perl modules for the life sciences., *Genome research*, 12, 10, 1611–8, doi:10.1101/gr.
332 361602
- 333 Stutz, M. (2000), Linux and the Tools Philosophy, [http://www.linuxdevcenter.com/pub/a/
334 linux/2000/07/25/LivingLinux.html](http://www.linuxdevcenter.com/pub/a/linux/2000/07/25/LivingLinux.html)
- 335 Tajima, F. (1989), Statistical method for testing the neutral mutation hypothesis by DNA polymorphism.,
336 *Genetics*, 123, 3, 585–595
- 337 The Gene Ontology Consortium (2015), Gene ontology consortium: going forward, *Nucleic Acids*
338 *Research*, 43, D1, D1049–D1056, doi:10.1093/nar/gku1179
- 339 Watterson, G. (1975), On the number of segregating sites in genetical models without recombination,
340 *Theoretical population biology*, 7, 2, 256–276
- 341 Yates, A., Beal, K., Keenan, S., McLaren, W., Pignatelli, M., Ritchie, G. R. S., et al. (2015), The ensembl
342 rest api: Ensembl data for any language, *Bioinformatics*, 31, 1, 143–145, doi:10.1093/bioinformatics/
343 btu613

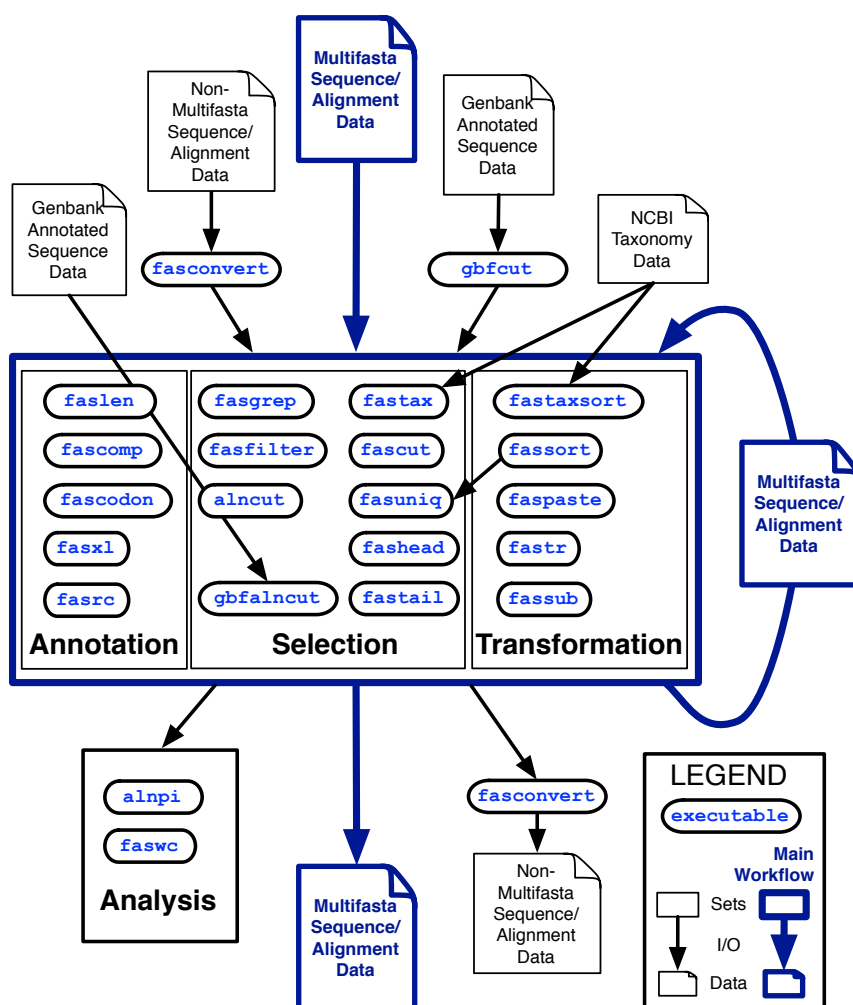


Figure 1. FAST version 1.0 with data and workflow dependencies indicated.

FIGURES

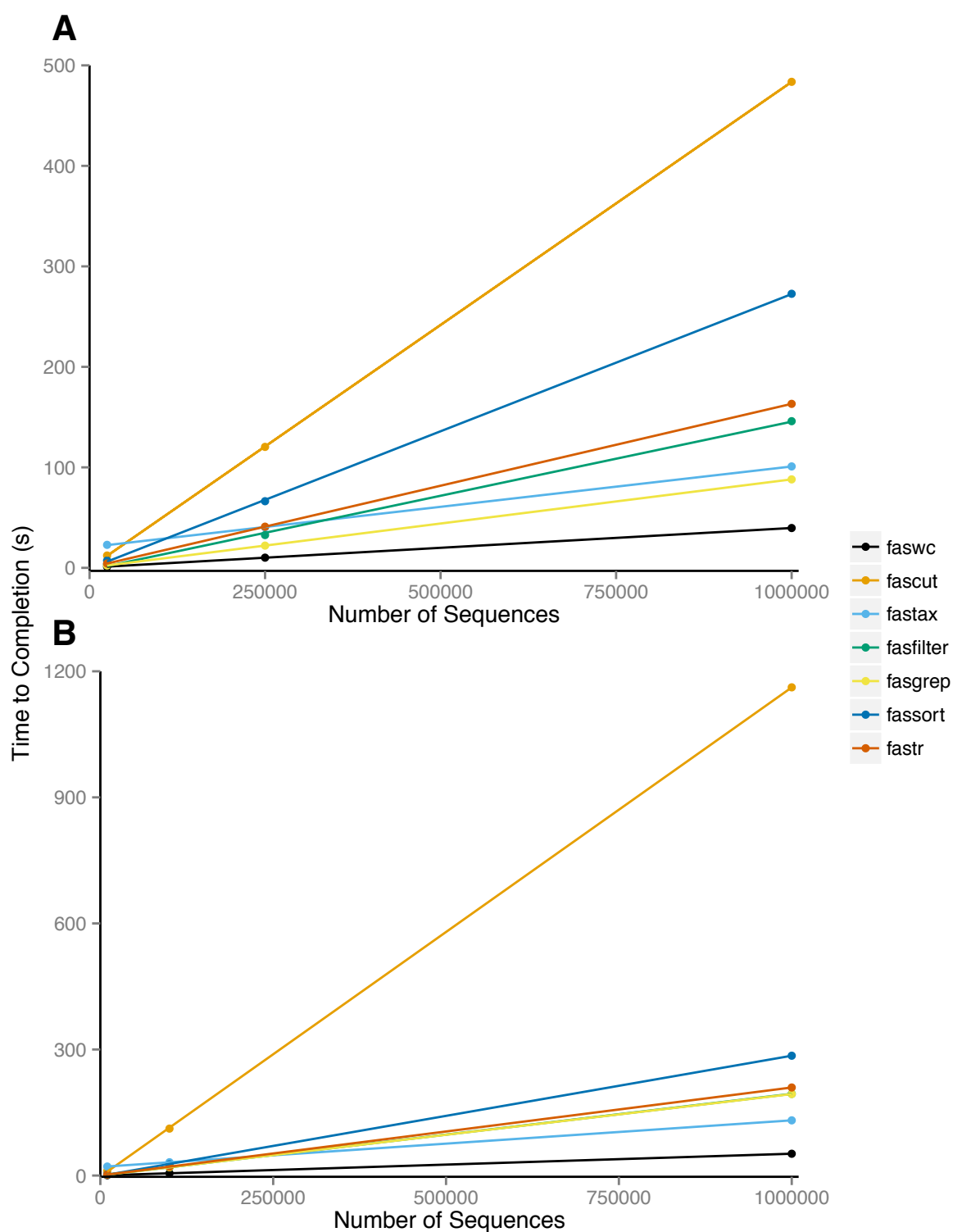


Figure 2. Average processor time of 100 repetitions required to complete analysis using indicated utility. Utilities were run on six datasets consisting of (a) 25000, 250000, and 1000000 100bp sequences and (b) 10000, 100000, and 1000000 1000bp sequences.