FAST:Fast Analysis of Sequences Toolbox Cookbook

Katherine C.H. Amrine & David H. Ardell

16 June 2014

Contents

These examples are executable from the installation directory.

1 Recipes

2 Tutorials

2.1 Prelude

2.1.1 The FAST definition of "FastA format"

FastA format began with the FastA search utilities of William Pearson. 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`.

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*, *description* and *sequence*. The *identifier* (indicated with `id` in the example here) and *description* (`desc`) together make the *identifier line* of a sequence record, that 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.

In FAST, the description may be broken into multiple *fields* defined by a *delimiter* (indicated with `<delim>`). FAST uses a "one-based" indexing of fields as indicated here:

```
>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
```

2.1.2 Use `man` pages for full documentation

All FAST utilities follow UNIX conventions in having default and optional behaviors. For more information about how to use and modify the behavior

of any FAST utility such as `faswc`, consult its manual page with *e.g.*:

```
man faswc
```

or alternatively:

```
perldoc faswc
```

2.2 Example 1: Prototyping a pipeline to cut, reverse complement, and translate a gene by coordinate from a genome

2.2.1 Calculating sequence length

Chromosome 1 from the *Saccharomyces cerevisiae* genome is available in `t/data/chr01.fsa`. By default, `faswc` calculates the lengths of sequence records on its input, and outputs its input, augmenting sequence descriptions with its calculations using the tag (or name) `length` and a (name,value) separator `:`, as in `length:872`. We can therefore easily obtain the length of this chromosome sequence as follows:

```
faswc t/data/chr01.fsa | egrep ">"
```

Alternatively, `faswc -c` will output the length of the chromosome directly to `STDOUT`:

```
faswc -c t/data/chr01.fsa
```

2.2.2 Cut out a subsequence by coordinate with `fascut`

`fascut` will cut a subsequence by coordinate. For example, suppose we know that the location of gene `YAR030C` in yeast chromosome 1 begins 186512 and ends 186853 on the minus strand. Let's cut this from our chromosome. The following code will extract this subsequence in fasta format to `STDOUT`:

```
fascut 186512..186853 t/data/chr01.fsa
```

2.2.3 Computing reverse complement of a sequence with `fasrc`

Knowing that this is on the minus strand, we need to obtain the reverse complement of this sequence. `fasrc` will compute this. The following code will take the output of `fascut` as its input and return the reverse complemeht in fasta file to `STDOUT`:

```
fascut 186512..186853 t/data/chr01.fsa | fasrc
```

2.2.4 Translating a sequence with `fasxl`

To translate this sequence, we extend the pipeline with the `fasxl` utility:

```
fascut 186512..186853 t/data/chr01.fsa | fasrc | fasxl
```

Examine the output, we will see that the peptide starts with a methionine, and ends with a stop codon, indicated by the `*` character by default.

2.2.5 Computing codon usage with `fascodon`

If we are interested in the codon usage of our gene, we can edit the last command-line (by typing `up-arrow` on most UNIX shells) and replace `fasxl` with `fascodon` at the end of our pipeline. `fascodon` outputs a space-delimited table indicating the normalized counts of each codon with information on starts and stops. With the following code, we can see that the most frequently used codon in this example is `AAT` (encoding an Asparagine)

```
fascut 186512..186853 t/data/chr01.fsa | fasrc | fascodon
```

2.2.6 Computing base composition with `fascomp`

`fascomp` will return the base/protein composition of a sequence. If we are interested in the normalized base composition of the first chromosome, we can run the following:

```
fascomp -n t/data/chr01.fsa
```

2.3 Example 2: Reformatting, selecting and transforming alignments in FAST

2.3.1 Reformatting alignment data with `fasconvert`

A file with protein sequences that match a search for "P450" is available in `t/data/P450.fas` under the FAST installation directory. Another file contains this data aligned using `clustalw` with the name `P450.clustalw2.aln`. The `fasconvert` tool can convert from fasta to many formats, or from many formats to fasta, including clustalw to fasta as showin in the following example

```
fasconvert -i clustalw -f t/data/P450.clustalw2.aln
```

The previous command automatically saves its output to an output file saves output to the same basename and an extension of `.fas`. The `faswc` utility will append sequence lengths to the sequence descriptions. To look at the length of all sequences, use the following code.

```
faswc t/data/P450.clustalw2.fas | head -1
```

which outputs `length:557` to `STDOUT`.

2.3.2 Selecting sequences with `fasgrep`

We can subset the output in many ways to get information we are interested in, for example, if we want to get the original sequence with the gi number "86475799", we can use `fasgrep`, which will pull out sequences that match a Perl regular expression. By default, `fasgrep` attempts to match sequence identifiers:

```
fasgrep "86475799" P450.fas
```

We can retrieve the aligned version of this sequence as it has the same identifier

```
fasgrep "86475799" P450.clustalw2.fas
```

2.3.3 Reformatting gap characters with `fastr`

`fastr` may be useful when we must change specific characters based on the requirements of a bioinformatic program. For example, to reformat gap characters in a fasta-format alignment from "-" to ".".

```
fastr -s "-" "." P450.clustalw2.fas
```

2.3.4 Degapping sites with `alndegap`

`alndegap` allows for editing of alignments based on their gap profile. To remove sites with at least one gap in all sequences, we can do the following:

```
alndegap -a P450clustalw2.clustalw.fas
```

We can then determine the length of the alignment by looking at the first identifier for your output after running the following:

```
alndegap -a P450clustalw2.clustalw.fas | faswc | head -1 | cut -f2 -d" "
```

And if we are interested in retaining only unique sequences, *fasuniq* appended to the output will collapse duplicate sequences to one, appending all of the identifiers to one large identifier.

```
alndegap -a P450clustalw2.clustalw.fas | faslen | fasuniq
```