FASTQ file preparation

Definition

Sequence

A **sequence** is an ordered list of symbols. It might describe the composition of a biopolymer. From this sequence we can infer some of the properties of the molecule, like it's isoelectic point or possibly it's location in the cell, or maybe it's function though each of these require a body of previous experimental knowledge to which we can make sequence comparisons. We might also be able to infer properties of subsequences, like a binding motif in a **DNA** sequence or a region of a **protein** sequence that may form a transmembrane helix.

Typical sequences are of nucleic acids or of proteins, using the alphabets of [ACGTNU] or [ACDEFGHIKLMNPQRSTVWY], respectively. We can also create sequences using characters of any alphabet, possibly representing the sequential shapes of a folded protein, or an encoding of the ordered phonemes in a bird song. The important properties of a sequence include the encoded characters of its alphabet and the ordered list of instances of those characters.

FASTA

A FASTA file is a collection of sequences with identifying labels.

A line that has a '>' character in the first column is an id line, with the id starting in the second column and running until white space (space, tab, or end-of-line). IDs end just before whitespace (including end of line). Comma, or any other character is legal in an ID. Note: no space is allowed between the '>' and the id. If you have a space after the '>', you have an empty id string, which is illegal in many fasta-reading programs.

The rest of the id line after the id is a comment and is frequently used to provide more information about the sequence. An id line identifies the sequence that follows, which consists of all lines until the next id line (or end-of-file). There is no standard way to provide comments other than on the id line (some, but not all, FASTA-reading programs ignore lines beginning with '#'). There is also no standard way to associate multiple ids with a sequence.

Lines that do not have a '>' in the first column are sequence lines. White space is ignored on sequence lines, but all other characters are interpreted as being part of the sequence, including characters that seem nonsensical for the alphabet of the sequence. A single sequence can span many lines, and it is quite common for the characters of the sequence to be grouped into sets of 10 characters. Putting 5 sets of 10 characters each on each line makes hand-counting easier---handy when you are debugging a program. To avoid breaking poorly written programs, it is a good idea not to make sequences lines extremely long.

Blank lines and lines containing just whitespace should be treated as legal anywhere in a FASTA file, even before the first ID line. Note: NCBI documentation states that blank lines are not allowed within the body of a FASTA record (ID with associated sequence), and we will take a more permissive stance.

FASTQ

FASTQ (see: The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants, Cock PJ et. al, 2009, NAR) records include an ID line and sequence information, similar to a FASTA file; and they also include a measure of quality associated with each symbol. FASTQ files are organized in 2 parts, the first has an ID and sequence, the second may repeat the ID and includes the quality scores. The ID/sequence section starts with @, and the quality section starts with +. The ID portion of the quality section is usually a copy of the ID section for the sequence section.

@identifier comment
ACGTACGTACGTN

+
HHHHHHHHHHB

The quality alphabet encodes a score that is mapped to the ASCII table, such that a single character can be used to encode a quality score for each character of the sequence. There are two basic mappings to the ASCII table - wikipediahas a nice description. The original Sanger mapping made use of ASCII characters range(33:73) to describe quality scores in range (0:40) and this encoding is called PHRED+33. Solexa (now Illumina) decided to use an offset of 64, creating PHRED+64 using ASCII range (64:104) to represent quality scores in range 0:40. Technically the upper bound on these ranges can go larger but this is not typically done.

The character value 33 corresponds to an ASCII "!", and the number 64 corresponds to ASCII "@", and each of these in PHRED+33 or PHRED+64 mappings (respectively), corresponds to a quality score of 0, which is the lowest quality score possible (exception: pre v 1.3). These names: "PHRED+33" or "PHRED+64", are used to name the specific representation of the quality scoring, and they remind us how the mapping can be computed. If you have a Qscore of 0, and you want it's character mapping for PHRED+33, you can add 33 to the quality score and convert that to a character. You may find the built-in python functions chr() and ord() useful for this task--example: chr(Q+33), or chr(Q + ord('!')).

In at least one version, Illumina decided that the lowest quality score should be "2" even when assigned to a Base read of N--this case means the machine could provide no information for that base, yet claim a P(error) < 1 !!. Some downstream programs get confused when we give them a base read of "N" with a quality score of 2.

Finally, Illumina versions before 1.3 specified the quality as -10*log10(P/(1-P)) (clarification: pre 1.3 Illumina is Solexa). This makes little difference where P(error) < 0.3, and where it does matter, the PHRED+64 mapping starts at ASCII 59 mapping Qscores of -1 through -5 (Ascii characters ;<=>?) which correspond to: 0.3 < P(error) < 1 for those 5 codes. These codes need to be fixed to appropriate PHRED+33 or PHRED+64 mappings, where the code reflectsP(error).

PHRED Q scores

What do the quality values refer to? In all cases except the early Solexa data (before version 1.3), the score is an error probability (denoted P(error) or P(e)). The score is encoded as the log10 of that error probability, and.. in order to make use of the ASCII encoding, the log10-prob is then multiplied by -10. (-10*log10(P)). If you see a "!" character in a PHRED+33 encoding, corresponding to a Quality score of 0, then this means that there is a probability of 1 that the base called in the corresponding sequence position is an error. In PHRED+64 encoding, a "@" encodes a quality score of 0 which we interpret as P(error)=1 that this base is in error. In other words.. that base is not known.

Examples:

Here is an example of a FASTA file with two sequences

```
>1914
masmtggqqm gripgnsprM VLLESEQFLT ELTRLFQKCR SSGSVFITLK
KYDgrtkpip rkssvEGLEP AENKCLLRAT DGKRKISTVV SSKEVNKFQM
AYSNLLRANM DGLKKRdkkn kskkskpAQG GEQKLiseed dsagspmpqF
QTWEEFSRAA EKLYLADPMK VRVVLKYRHV DGNLCIKVTD DLVCLVYRTD
QAQDVKKIEK FHSQLMRLMV AKESRNVtme te
>1a2xB
gdEEKRNRAITARRQHLKSVMLQIAATELEKEEgrreaekqnylaeh
```

Here is an example of a FASTQ file with 2 sequences:

```
@HWI-ST611_0189:1:1101:1237:2140#0/1
ACTAGCTGTCCTTGGTGCCCGAGTGTATTGAAAGTTGATTCCCTTATAGATGTTCGTTTTCCACACAACTCTGTAGGCACCANCAATNACTAGTAGATCG
```

```
+HWI-ST611_0189:1:1101:1237:2140#0/1
___`ccccceeeeehh`dcfhe`dQb`deehehebY^aedfdhhhhe_ebaa]cebe_eehfhS__ede_V\HHV^^^acccBBBBBBBBBBBBBBBBBB
@HWI-ST611_0189:1:1101:1178:2158#0/1
ACTAGCTATTGATGGTGCCTACAGTCTCAGTTGAGGGGAAANNNNCGNGNNNCNNNNNNNNNNNNNANNNCNGNNNAGNNNNNNNNNNNNNNNGCCNANN
+HWI-ST611_0189:1:1101:1178:2158#0/1
bbbeeeeeggfegiidgggggghghhhhifhghhhiiiifghBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
```

Notice that the quality scores go from "B" to "b", meaning Q2 through Q34. This tells us immediately that this is PHRED+64 encoded, and it is using "B" as its lowest score even when the corresponding sequence letter is N. So, all of those "B"s should be considered as P(error) = 1. A string of "B" on the 3' end, further means that all of the read sequence should be disregarded even if it has apparently valid bases.

Suggested reading

The following selections from the on-line documentation will probably give you most of what you need and include more tutorial examples (thanks to Andrew Uzilov for recommending them):

Tutorial docs:

http://docs.python.org/tutorial/interpreter.html
http://docs.python.org/tutorial/introduction.html
http://docs.python.org/tutorial/controlflow.html
http://docs.python.org/tutorial/inputoutput.html

Understanding strings (and more generally, sequences):

http://docs.python.org/library/stdtypes.html#string-methods
http://docs.python.org/library/stdtypes.html#string-formatting-operations
http://docs.python.org/library/stdtypes.html#sequence-types-str-unicode-list-tuple-buffer-xrange

Useful standard library modules:

http://docs.python.org/library/string.html
http://docs.python.org/library/re.html
http://docs.python.org/library/sys.html

PYTHON program:

Read a FASTQ file that has may have mixed upper and lower case letters, make them all upper case. We might see any of "*", ".", "n" or "N" all meaning unknown base. Make all of these "N".

The program should be a "filter" program that reads from STDIN and outputs to stdout. We will not delete any character, so the final length of each line should remain unchanged.

Implement options that specify an input of PHRED+33 or PHRED+64 and a desired output of PHRED+33 or PHRED+64. Use:

- -P33in (--PHRED33input),
- -P64in (--PHRED64input)
- -P64Bin (--PHRED64 with B offset in quality values)
- -P64SOLin (--PHRED64 with SOLEXA interpretation of Q score)
- -P33out (--PHRED33output) [make this your default output - SANGER FASTQ]
- -P64out (--PHRED64output)

In the case of the PHRED64 Solexa mapping (--PHREDSOLin), remap these scores taking into consideration the pre 1.3 Solexa quality mapping and assuming that the output will be a mapping of P(error), rather than

P(error)/(1- P(error)).