File Format in Bioinformatics

1 Fasta

The FASTA format is a text-based format for representing either nucleotide sequences or amino acid sequences, in which nucleotides or amino acids are represented using single-letter codes. A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The definition line (defline) is distinguished from the sequence data by a ">" symbol at the beginning. The word following the ">" symbol is the identifier of the sequence, and the rest of the line is the description (optional). There should be no space between the ">" and the first letter of the identifier.

2 Fastq

FASTQ format is a text-based format for storing both a biological sequence (usually nucleotide sequence) and its corresponding quality scores. Both the sequence letter and quality score are each encoded with a single American Standard Code for Information Interchange (ASCII) character for brevity. A FASTQ file normally uses four lines per sequence: line 1 begins with a '@' character and is followed by a sequence identifier and an optional description; line 2 is the raw sequence represented by single-letter codes; line 3 begins with a '+' character and is optionally followed by the same sequence identifier and any description again; line 4 encodes the quality values for the sequence in line 2, and must contain the same number of symbols as letters in the sequence.

3 Phred quality score

A Phred quality score is a measure of the quality of the identification of the nucleobases generated by automated DNA sequencing. It is defined as

$$Q = -10 \log_{10} p \tag{1}$$

where p is base-calling error probability.

In FASTQ format, quality values equals sum of Phred quality score and a constant. The constant is 33 in most cases (Phred33). Sometimes it is 64 (Phred64), *e.g.* Solexa, before Illumina 1.8.

4 SAM

Sequence Alignment Map (SAM) is a text-based format originally for storing biological sequences aligned to a reference sequence. It consists of a header and an alignment section. The header section must be prior to the alignment section if it is present. Lines in header begin with the '@' symbol, which distinguishes them from the alignment section.

Alignment sections have 11 mandatory fields:

1st: QNAME. Query template name. Reads/segments having identical QNAME are regarded to come from the same template. A QNAME * indicates the information is unavailable. In a SAM file, a read may occupy multiple alignment lines, when its alignment is chimeric or when multiple mappings are given.

2ed: FLAG. Combination of bitwise FLAGs (seen below).

3rd: RNAME. Reference sequence name of the alignment. If @SQ header lines are present, RNAME (if not *) must be present in one of the SQ-SN tag. An unmapped segment without coordinate has a * at this field. However, an unmapped segment may also have an ordinary coordinate such that it can be placed at a desired position after sorting. If RNAME is *, no assumptions can be made about POS and CIGAR.

4th: POS. 1-based leftmost mapping position of the first matching base. The first base in a reference sequence has coordinate 1. POS is set as 0 for an unmapped read without coordinate. If POS is 0, no assumptions can be made about RNAME and CIGAR.

5th: MAPQ. Mapping quality. It equals $10 \log_{10} Prmappingpositioniswrong$, rounded to the nearest integer. A value 255 indicates that the mapping quality is not available.

6th: CIGAR. Concise idiosyncratic gapped alignment report (CIGAR) string.

7th: RNEXT. Reference sequence name of the primary alignment of the next read in the template. For the last read, the next read is the first read in the template. If @SQ header lines are present, RNEXT (if not * or =) must be present in one of the SQ-SN tag. This field is set as * when the information is unavailable, and set as = if RNEXT is identical RNAME. If not = and the next read in the template has one primary mapping (see also bit 0x100 in FLAG), this field is identical to RNAME at the primary line of the next read. If RNEXT is *, no assumptions can be made on PNEXT and bit 0x20.

8th: PNEXT. Position of the primary alignment of the next read in the template. Set as 0 when the information is unavailable. This field equals POS at the primary line of the next read. If PNEXT is 0, no assumptions can be made on RNEXT and bit 0x20.

9th: TLEN. Signed observed template Length. If all segments are mapped to the same reference, the unsigned observed template length equals the number of bases from the leftmost mapped base to the rightmost mapped base. The leftmost segment has a plus sign and the rightmost has a minus sign. The sign of segments in the middle is undefined. It is set as 0 for single-segment template or when the information is unavailable.

10th: SEQ. Segment Sequence. This field can be a * when the sequence is not stored. If not a *, the length of the sequence must equal the sum of lengths of M/I/S/=/X operations in CIGAR. An = denotes the base is identical to the reference base. No assumptions can be made on the letter cases.

11th: QUAL. ASCII of base quality (Phred33). This field can be a * when quality is not stored. If not a *, SEQ must not be a * and the length of the quality string ought to equal the length of SEQ.

5 Bitwise FLAGs of SAM

The bitwise FLAGs is displayed as a single integer, but is the sum of bitwise flags to denote multiple attributes of a read alignment. Each attribute denotes one bit in the binary representation of the integer.

Integer 1 (binary 000000000001): template having multiple templates in sequencing (read is paired).

Integer 2 (binary 000000000010): each segment properly aligned according to the aligner (read mapped in proper pair)

Integer 4 (binary 000000000100): segment unmapped (read1 unmapped).

Integer 8 (binary 000000001000): next segment in the template unmapped (read2 unmapped).

Integer 16 (binary 000000010000): SEQ being reverse complemented (read1 reverse complemented).

Integer: 32 (binary 000000100000): SEQ of the next segment in the template being reverse complemented (read2 reverse complemented).

Integer 64 (binary 000001000000): the first segment in the template (is read1).

Integer 128 (binary 000010000000): the last segment in the template (is read2).

Integer 256 (binary 000100000000): not primary alignment.

Integer 512 (binary 001000000000): alignment fails quality checks.

Integer 1024 (binary 010000000000): PCR or optical duplicate.

Integer 2048 (binary 100000000000): supplementary alignment (e.g. aligner specific, could be a portion of a split read or a tied region).

6 BED

The BED (Browser Extensible Data) format is a text file format used to store genomic regions as coordinates and associated annotations. The data are presented in the form of columns separated by spaces or tabs.

A BED file can optionally contain a header. However, there is no official description of the format of the header.

A BED file consists of a minimum of three columns (1st-3rd) to which nine optional columns (4th-12th) can be added for a total of twelve columns:

1st: chrom. Chromosome or scaffold name

2nd: chromStart. Start coordinate on the chromosome or scaffold for the sequence considered. This position is inclusive. The first base on the chromosome is numbered 0.

3rd: chromEnd. End coordinate on the chromosome or scaffold for the sequence considered. This position is non-inclusive.

4th: name. Name of the line in the BED file.

5th: score. Score between 0 and 1000.

6th: strand. DNA strand orientation (positive ["+"] or negative ["-"] or "." if no strand).

7th: thickStart. Starting coordinate from which the annotation is displayed in a thicker way on a graphical representation (e.g. the start codon of a gene).

8th: thickEnd. End coordinates from which the annotation is no longer displayed in a thicker way on a graphical representation (e.g. the stop codon of a gene).

9th: itemRgb. RGB value in the form R,G,B (e.g. 255,0,0) determining the display color of the annotation contained in the BED file.

10th: blockCount. Number of blocks (e.g. exons) on the line of the BED file.

11th: blockSizes. List of values separated by commas corresponding to the size of the blocks. The number of values must correspond to that of the "blockCount".

12th: blockStarts. List of values separated by commas corresponding to the starting coordinates of the blocks, coordinates calculated relative to those present in the chromStart column. The number of values must correspond to that of the "blockCount".

7 GFF

General feature format (gene-finding format, generic feature format, GFF) is a file format used for describing genes and other features of DNA, RNA and protein sequences. All GFF formats (GFF2, GFF3

and GTF) are tab delimited with 9 fields per line:

1st: sequence. The name of the sequence where the feature is located.

2ed: source. Keyword identifying the source of the feature, like a program (e.g. Augustus or RepeatMasker) or an organization (like TAIR).

3rd: feature. The feature type name, like "gene" or "exon". In a well structured GFF file, all the children features always follow their parents in a single block (so all exons of a transcript are put after their parent "transcript" feature line and before any other parent transcript line). In GFF3, all features and their relationships should be compatible with the standards released by the Sequence Ontology Project.

4th: start. Genomic start of the feature. This position is inclusive. The first base on the sequence is numbered 0.

5th: end. Genomic end of the feature. This position is inclusive.

6th: score. Numeric value that generally indicates the confidence of the source in the annotated feature. A value of "." (a dot) is used to define a null value.

7th: strand. Single character that indicates the strand of the feature; it can assume the values of "+" (positive, or 5'-¿3'), "-", (negative, or 3'-¿5'), "." (undetermined).

8th: phase. Phase of CDS features; it can be either one of 0, 1, 2 (for CDS features) or "." (for everything else). 0, 1, or 2, indicating the number of bases that should be removed from the beginning of this CDS feature to reach the first base of the next codon.

9th: attributes All the other information pertaining to this feature. The format, structure and content of this field is the one which varies the most between the three competing file formats.

References

Wikipedia