Introduction to high-throughput sequencing file formats

Daniel Vodák
(Bioinformatics Core Facility,
The Norwegian Radium Hospital)
(danielvo@ifi.uio.no)

File formats

- ~~Do we need them?~~ Why do we need them?
  - A *standardized* file has a clearly defined structure – the nature and the organization of its content are known
    - Important for automatic processing (especially in case of large files)
    - Re-usability saves work and time
- Why the variability then?
  - Effective storage of specific information (differences between data-generating instruments, experiment types, stages of data processing and software tools)
  - Parallel development, competition
- Need for (sometimes imperfect) conversions

Binary and “flat” file formats

- “Flat” (“plain text”, “human readable”) file formats
  - Possible to process with simple command-line tools (field/column structure design)
  - Large in size
  - Space is often saved through the means of archiving (e.g. tar, zip) and human-readable information coding (e.g. flags)
    - File format specifications (“manuals”) are very important (often indispensable) for correct understanding of given file’s content
- Binary file formats
  - Not human-readable
  - Require special software for processing (programs intended for plain text processing will not work properly on them, e.g. wc, grep)
  - (significant) reduction to file size
- High-throughput sequencing files – typically GBs in size

Comparison of file sizes

- Plain text file
  - example.sam – 2.5 GB (100 %)
- Binary file
  - example.bam – 611 MB (23.36 %)
  - Possibility of indexing – quick random access
  - Archiving and processing
- Archived plain text file
  - example.sam.tar.gz – 570 MB (22.36 %)
  - Only archiving

Our file format menu

- FASTA
  - Simple collections of named DNA/protein sequences
- FASTQ
  - Same as FASTA, but with additional quality information
  - Widely used for storing sequencing reads
- SAM
  - Alignments of sequencing reads to a reference genome
- BED
  - Region-based genome annotation information, often used for visualization

FASTA format

- Origin in 1980's FASTP/FASTA software packages for sequence similarity searching
- Convenient format for storing/transferring simple sequence collections
  - Intended for both, nucleotide and protein sequences (FAST-A, "Fast-All")
  - Common output format of sequence databases
  - Input format for various tools

FASTA format

- Each sequence entry consists of:
  - a single line describing the sequence, starting with the "greater-than" symbol (">")
  - any number of lines representing the sequence

```
>sp|P08246|ELNE_HUMAN Neutrophil elastase OS=Homo sapiens GN=ELANE PE=1 SV=1
MTLGRRLACLFLACVLPALLLGGTALASEIVGGRRARPHAWPFMVSLQLRGGHFCGATLI
APNFVMSAAHCVANVNVRAVRVVLGAHNLSRREPTRQVFAVQRIFENGYDPVNLLNDIVI
LQLNGSATINANVQVAQLPAQGRRLGNGVQCLAMGWGLLGRNRGIASVLQELNVTVVTSL
CRRSNVCTLVRGRQAGVCFGDSGSPLVCNGLIHGIASFVRGGCASGLYPDAFAPVAQFVN
WIDSIIQRSEDNPCPHPRDPDPASRTH
```

FASTA format

- Line lengths are usually limited to a certain number of characters
- Details (e.g. acceptance of empty lines and allowed symbols for nucleic/amino acid codes) can be application-specific
- NCBI's requested FASTA format description: http://www.ncbi.nlm.nih.gov/BLAST/blastcgihelp.shtml

FASTQ format

- Extension of the FASTA file format, intended for storing sequences together with related quality information
- Typical NGS data file format at the beginning of a bioinformatics analysis
  - Final output of many sequencing instruments
  - Common way of storing raw sequence data

FASTQ format

- Each sequence entry usually consists of exactly 4 lines
  - The “at” sign (“@”), followed by a sequence identifier and optional description information
  - The sequence
  - The “plus” symbol (“+”), optionally followed by the sequence identifier and description information
    - Usually only the initial symbol
  - Sequence quality information – each sequence symbol is matched by exactly one corresponding quality symbol

```
@SRR494504.65 GA2MACHINE1_0000:3:1:6851:969 length=78
GTTCAACTCTGTGACTTGAATGCAAACATCACAAAGAAGTTACTGGGAATGCTGCTGTCTGCTTTTTATATGTAATAG
+SRR494504.65 GA2MACHINE1_0000:3:1:6851:969 length=78
,00023230@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@22@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@22@@@@@@II
```

FASTQ format

- Several incompatible versions of the format
  - Different means of computing and/or recording the quality information
- We will focus on the newest version (Illumina 1.8+)
- Detailed version overview and up to date information available on Wikipedia
  - (http://en.wikipedia.org/wiki/FASTQ_format)
- History of the “unfortunate developments” was well captured in an article by Peter J. A. Cock et al.
  - (The Sanger FASTQ file format … variants)

FASTQ format – Base qualities

- PHRED Base quality (Q) – value derived from the estimated probability (P) of given base being determined wrong
  - $Q = -10*\log_{10}(P)$ (rounded to nearest integer)
- P might be computed in different ways, depending on the sequencing platform, but the meaning of Q remains the same
  - $Q = 40 \sim P = 10^{-4} = 0.0001$
  - $Q = 20 \sim P = 10^{-2} = 0.01$
  - $Q = 10 \sim P = 10^{-1} = 0.1$
  - $Q = 0 \sim P = 10^{0} = 1$

FASTQ format – Base qualities

- A single-character notation of the values is more efficient than recording the actual numbers
  - Lengths of the base quality record and the sequence record match
  - Individual integer values are represented by corresponding characters from the ASCII code table, offset of 33 ensures that only visible characters are used
    - $Q = 40 \sim P = 10^{-4} = 0.0001$ (ASCII [33 + 40] = "I")
    - $Q = 20 \sim P = 10^{-2} = 0.01$ (ASCII [33 + 20] = "5")
    - $Q = 10 \sim P = 10^{-1} = 0.1$ (ASCII [33 + 10] = "+")
    - $Q = 0 \sim P = 10^{0} = 1$ (ASCII [33 + 0] = "!")
  - Some of the older FASTQ format versions used offset of 64 instead

The ASCII code table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

The Sequence Alignment/Map format (SAM)

- Intended for storing read alignments against reference sequences
  - Widely used for that purpose (e.g. in expression analysis and variant discovery)
- Has a binary version with good software support (BAM format)
- Full latest format specification available on-line:

  http://samtools.sourceforge.net/SAM1.pdf
- Samtools – a set of useful programs for manipulating files in SAM/BAM format:

  http://samtools.sourceforge.net

SAM – header section

- Header section – general information about the file's origin and about the sources of its content
  - Meta-information about the used reference genome, sequencing reads, alignment software, etc.
  - Optional, but has to be at the beginning of the file if present
    - However, some programs might require it
  - All header lines are started with a special symbol ("@") for easy identification
  - Example:

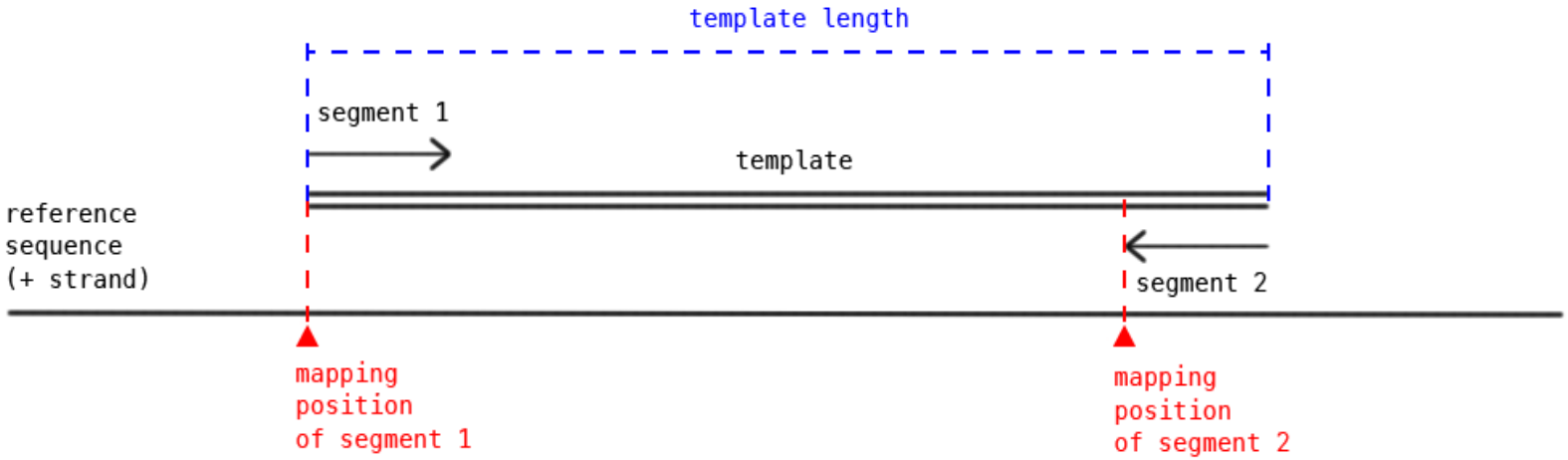
```
@HD     VN:1.0  SO:unsorted
@PG     ID:novoalign    VN:V2.07.13
@SQ     SN:chr10        AS:hg18.nix     LN:135374737
…
```

SAM – alignment section

- Alignment section – detailed information about the individual read-alignments
  - Tab-delimited value fields, 1 alignment per line
  - 11 mandatory fields
    - All must be present, and in given order, for each alignment, i.e. if some of the information is unavailable, this fact must be recorded (depending on the field, either as "*" or as "0")
    - The specification provides a range of valid values for each field
  - Variable number of optional fields (optional fields can be sequencing platform-specific or aligner-specific)

Templates and segments

- Template – (piece of a) DNA/RNA molecule which was a subject to sequencing
  - "Insert size" = template size
- Segment – part of the template which was "read" by a sequencing machine ("sequencing read")



SAM – alignment section Mandatory fields overview

- Example alignment:

```
SRR039769.917  0       chr8    128124816       150     44M
        *       0       0
ACACAGGNAGTGCTCAAGAATTATTTGCATCCAAGAACTATTTG
B6B;BB8!<>:BB@@B@B93<@AB><BABAA=AAB718@>@@@@
PG:Z:novoalign AS:i:6  UQ:i:6  NM:i:1  MD:Z:7A36
```

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	`[!-?A-~]{1,255}`	Query template NAME
2	FLAG	Int	$[0,2^{16}-1]$	bitwise FLAG
3	RNAME	String	`\*|[!-()+-<>-~][!-~]*`	Reference sequence NAME
4	POS	Int	$[0,2^{29}-1]$	1-based leftmost mapping POSition
5	MAPQ	Int	$[0,2^{8}-1]$	MAPping Quality
6	CIGAR	String	`\*|([0-9]+[MIDNSHPX=])+`	CIGAR string
7	RNEXT	String	`\*|=|[!-()+-<>-~][!-~]*`	Ref. name of the mate/next segment
8	PNEXT	Int	$[0,2^{29}-1]$	Position of the mate/next segment
9	TLEN	Int	$[-2^{29}+1,2^{29}-1]$	observed Template LENgth
10	SEQ	String	`\*|[A-Za-z=.]+`	segment SEQuence
11	QUAL	String	`[!-~]+`	ASCII of Phred-scaled base QUALity+33

SAM – bitwise FLAG

- FLAG – A sum of indicator values, enumerates conditions true for given alignment
  - Each value present in the sum corresponds to one fulfilled condition
  - "bitwise" – the possible decimal values are powers of two (each can be represented by a single positive bit in a binary number)
  - No FLAG can be the result of two different sums

Binary	Decimal	Bit	Description
1	1	0x1	template having multiple segments in sequencing
10	2	0x2	each segment properly aligned according to the aligner
100	4	0x4	segment unmapped
1000	8	0x8	next segment in the template unmapped
10000	16	0x10	SEQ being reverse complemented
100000	32	0x20	SEQ of the next segment in the template being reversed
1000000	64	0x40	the first segment in the template
10000000	128	0x80	the last segment in the template
100000000	256	0x100	secondary alignment
1000000000	512	0x200	not passing quality controls
10000000000	1024	0x400	PCR or optical duplicate

SAM – bitwise FLAG (2)

- Example – FLAG 83:
  - In binary numbers:
    - FLAG 1010011 is a sum of 1000000, 10000, 10 and 1
    - Easy to break down, but binary numbers are long
  - In decimal numbers:
    - FLAG 83 is a sum of 64, 16, 2 and 1
      - 83 = 64 + 19 = 64 + (16 + 3) = 64 + 16 + (2 + 1)
    - What does FLAG 83 mean?

1024	512	256	128	64	32	16	8	4	2	1
0	0	0	0	1	0	1	0	0	1	1

SAM – Mapping qualities

- PHRED Mapping quality (Q) – value derived from the estimated probability (P) of the reported alignment being determined wrong
  - $Q = -10*\log_{10}(P)$ (rounded to nearest integer)
- P might be computed in different ways, depending on the aligner, but the meaning of Q always remains the same
  - $Q = 40 \sim P = 10^{-4} = 0.0001$
  - $Q = 20 \sim P = 10^{-2} = 0.01$
  - $Q = 10 \sim P = 10^{-1} = 0.1$
  - $Q = 0 \sim P = 10^{0} = 1$
- Very high mapping qualities can be commonly seen (e.g. 150 in our example) if given read/segment aligns only to a single location in a (large) genome
- Multiple alignments of the same read/segment lead to steep drop in mapping quality

SAM – CIGAR string

- CIGAR string summarizes how the given segment aligned to the reference sequence
  - Sequence mismatch (X) is considered to be an alignment match (M), CIGAR string doesn't always give a good overview of sequence mismatches
  - Good overview of insertions (I), deletions (D) and low quality read/segment ends, usually indicated by soft clipping (S)
  - (N) implies an intron in case of mRNA sequencing

Op	BAM	Description
M	0	alignment match (can be a sequence match or mismatch)
I	1	insertion to the reference
D	2	deletion from the reference
N	3	skipped region from the reference
S	4	soft clipping (clipped sequences present in SEQ)
H	5	hard clipping (clipped sequences NOT present in SEQ)
P	6	padding (silent deletion from padded reference)
=	7	sequence match
X	8	sequence mismatch

SAM – Optional fields

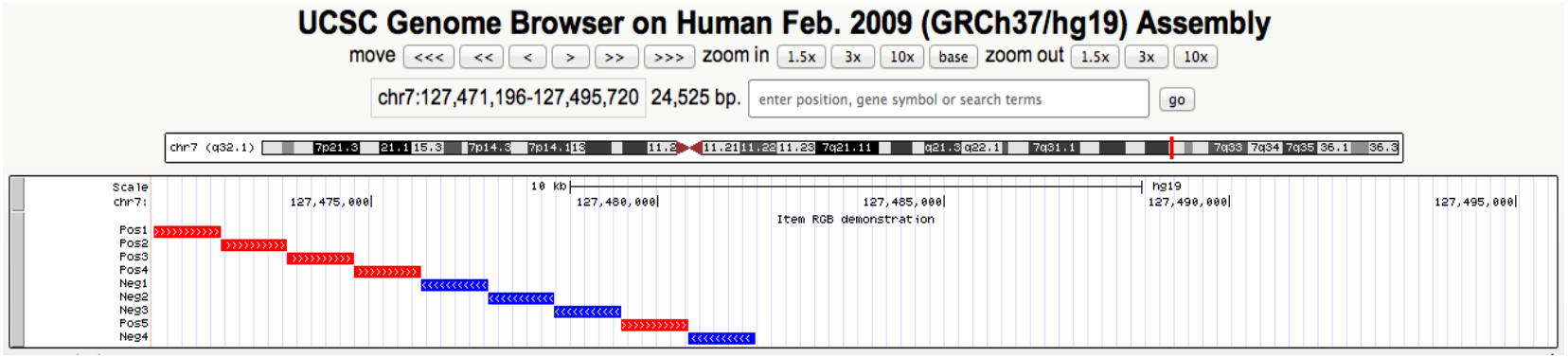
- Not all alignments in a given SAM file need to have the same optional fields
- Some defined in the SAM format specification
  - MD:Z – mismatching positions
  - NM:i – edit distance to the reference
  - OQ:Z, OP:i, OC:Z – original base quality, mapping position and CIGAR (before recalibration/realignment)
- Some defined by aligners
  - ZH:i – miRNA hairpin alignment score (Novoalign)
  - XO:i – number of gap openings in the alignment (BWA)

BED format

- List of genomic regions (each region is specified by a reference sequence and the start and end positions on it)
- Optionally, each region can have additional properties defined
  - Strand, name, score
  - Color
- Intended for visualizing genomic annotations in UCSC Genome Browser
- Format specification available online:
  http://genome.ucsc.edu/FAQ/FAQformat.html#format1
- BEDtools – a set of useful programs for manipulating files in BED format:
  http://code.google.com/p/bedtools/

BED file viewed in the Genome Browser

```
browser position chr7:127471196-127495720
browser hide all
track name="ItemRGBDemo" description="Item RGB demonstration" visibility=2
itemRgb="On"
chr7    127471196  127472363  Pos1  0  +  127471196  127472363  255,0,0
chr7    127472363  127473530  Pos2  0  +  127472363  127473530  255,0,0
chr7    127473530  127474697  Pos3  0  +  127473530  127474697  255,0,0
chr7    127474697  127475864  Pos4  0  +  127474697  127475864  255,0,0
chr7    127475864  127477031  Neg1  0  -  127475864  127477031  0,0,255
chr7    127477031  127478198  Neg2  0  -  127477031  127478198  0,0,255
chr7    127478198  127479365  Neg3  0  -  127478198  127479365  0,0,255
chr7    127479365  127480532  Pos5  0  +  127479365  127480532  255,0,0
chr7    127480532  127481699  Neg4  0  -  127480532  127481699  0,0,255
```



BED format – mandatory fields

- "Track lines" instead of header lines
  - Might do more harm than good outside of UCSC GB
- 3 mandatory fields (must be in correct order)
  - "chrom" – the reference sequence
  - "chromStart" – the first base of the region with respect to the reference sequence (counting starts from 0)
  - "chromEnd" – the first base *after* the region with respect to the reference sequence
    - [chromStart, chromEnd) allows easy region-length calculation
    - Example: chr1 7 8
    - Important to remember when working with very small regions and/or when the exact position matters (point mutations)

BED format – optional fields

- 9 additional optional fields, their order is binding (unlike with SAM format)
  - if *n* optional fields are included, they will be considered to be the first *n* optional fields from the format specification
- All regions must have the same optional fields described (unlike with SAM format)
- Most important optional fields:
  - "name" – name of the region
  - "score" – score value between 0 and 1000 (read-count, transformed p-value, "quality", …)
    - Can be interpreted as shades of grey during visualization
  - "strand" – either "+" or "-" (not "1"/"-1")

BED format – “pseudoBED” files

- Information captured in BED’s mandatory fields is essential for many applications/analysis types, however, additional custom information (not fitting into the BED format specification) is often required
- Other file formats might resemble the BED format while not being compatible
  - Be careful when creating and interpreting BED-like files
  - Be especially careful when it comes to regions’ genomic positions