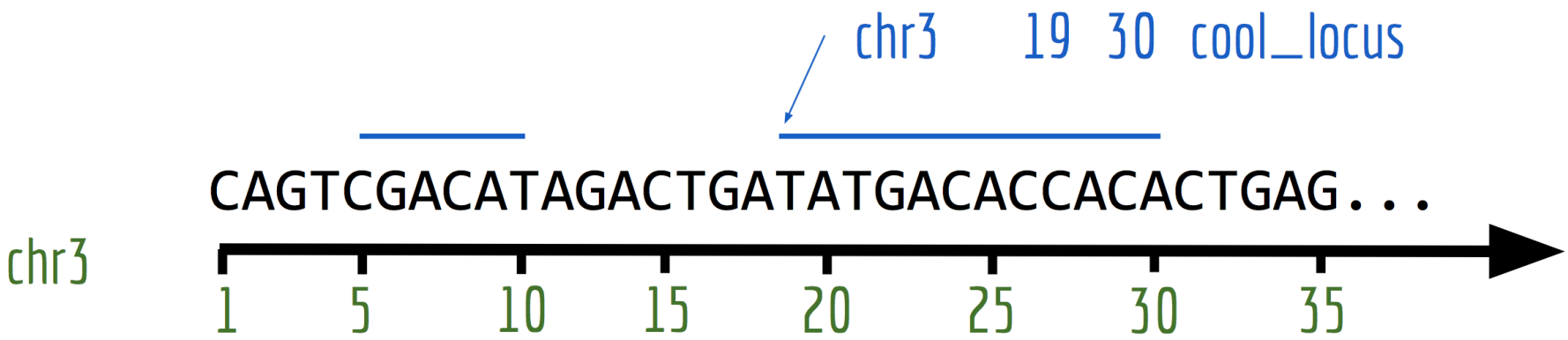
Genomic data formats

Mikhail Dozmorov

Spring 2018

Sequencing data ecosystem

The reference genome as a coordinate system



Binary and "plain text" files

"Plain text" ("flat") file formats

- Information often structured into lines and columns
- Human-readable, processable with simple command-line tools
- Space is often saved through the means of archiving (e.g. zip) and human-readable information coding (e.g. flags)

Binary file formats

- Not human-readable (not intended for reading)
- Require special software for processing (programs intended for plain text processing will not work properly on them, e.g. `wc`, `grep`)
- Efficient storage, (significant) reduction to file size when compared to a plain text counterpart (e.g. 75 % space saved)

Some common file formats

- **FASTA** - Simple collections of named DNA/protein sequences (text)
- **FASTQ** - Extension of FASTA format, contains additional quality information. Widely used for storing unaligned sequencing reads (text)
- **SAM/BAM** - Alignments of sequencing reads to a reference genome (text/binary)
- **BED** - Region-based genome annotation information (e.g. a list of genes and their genomic locations). Used for visualization or simple enumeration (text)
- **GFF/GTF** - gene-centric annotations (text)
- **(big)WIG** - continuous signal representation (text, binary)
- **VCF** - variant call format, to store information about genomic variants (text)

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* with sequence metadata, starting with the "greater-than" symbol ("$>$")
- *any number of lines* representing the actual sequence

```
>SRR493818.1.1 HWUSI-EAS519_0001_FC61TCNAAXX:2:1:960:20843 length=54
NGTAAATCCCGTTACCTCTCTGAGCCTCGGTNNCCNNNNNTGTAAAAAGGNNNN
>SRR493818.2.1 HWUSI-EAS519_0001_FC61TCNAAXX:2:1:961:7550 length=54
NACACTACAATGTAAAAGCTTGGCCTAACTTNNTTNNNNNGGCTGTTATTNNNN
```

FASTA format

The nucleic acid codes that can be found in FASTA file:

```
A --> adenosine              M --> A C (amino)
C --> cytidine               S --> G C (strong)
G --> guanine                W --> A T (weak)
T --> thymidine              B --> G T C
U --> uridine                D --> G A T
R --> G A (purine)           H --> A C T
Y --> T C (pyrimidine)       V --> G C A
K --> G T (keto)             N --> A G C T (any)
                             -  gap of indeterminate length
```

http://zhanglab.ccmb.med.umich.edu/FASTA/

FASTA format

- Human genome is there!
  http://hgdownload.cse.ucsc.edu/goldenPath/hg38/chromosomes/
- Get chromosome Y sequence:

```
$ URL=http://hgdownload.cse.ucsc.edu/goldenPath/hg38/chromosomes/ch
$ wget $URL; mv chrY.fa.gz hg38.chrY.fa.gz
# or
$ curl $URL > hg38.chrY.fa.gz
# Ungzip
$ gzip -d hg38.chrY.fa.gz
# Peek into the file
$ head hg38.chrY.fa
$ tail hg38.chrY.fa
# How many lines
$ wc -l hg38.chrY.fa
# Look somewhere in the middle
$ head -n 10000 hg38.chrY.fa | tail
```

Orienting in FASTA file

- Do not run:

```
$ # grep > hg38.chrY.fa
```

- What this command will do?

```
$ grep ">" hg38.chrY.fa
```

- What this command will do?

```
$ grep -v ">" hg38.chrY.fa
```

chrY summary statistics

- How long is chrY?

```
$ grep -v ">" hg38.chrY.fa | grep -o "[ATCGatcg]" | wc -l
26415043
```

- How many adenosines are there?

```
$ grep -v ">" hg38.chrY.fa | grep -o -i "A" | wc -l
7886192
```

Low-complexity regions

- Approximately 45% of the human genome can currently be recognized as being derived from transposable elements
- The majority are non-long terminal repeat (LTR) retrotransposons, such as LINE-1 (L1), Alu and SVA elements
- Other low-complexity biases: CG- and AT-rich DNA
- Represented by lower-case letters

```
GATCAAAGTGTCATACAGTAACAGCCCAGACAGACGATAGGTATGGCAGa
aaagaaaaaaactaaaaaaaaaaaaaaaaaaaaaaaaTCGCATGGGAAGTT
TCCCCGCCTCCTCTTTGGCCATTCTGTGCCCGGAGATCAAAGTTCTCATT
```

The conversion of sequencing signal to a nucleotide sequence

Nearly all sequencing technologies produce sequencing reads in **FASTQ** format

Sequence ID	`@SEQ_ID`
Sequence	`GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAA`
Separator	`+`
Quality scores	`!''*(((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>`

https://en.wikipedia.org/wiki/FASTQ_format

Sequence ID

```
@EAS139:136:FC706VJ:2:2104:15343:197393 1:Y:18:ATCACG
```

EAS139	the unique instrument name
136	the run id
FC706VJ	the flowcell id
2	flowcell lane
2104	tile number within the flowcell lane
15343	'x'-coordinate of the cluster within the tile
197393	'y'-coordinate of the cluster within the tile
1	the member of a pair, 1 or 2 *(paired-end or mate-pair reads only)*
Y	Y if the read is filtered, N otherwise
18	0 when none of the control bits are on, otherwise it is an even number
ATCACG	index sequence

FASTQ quality scores (Phred scores)

PHRED Base quality (Q) – integer value derived from the estimated probability (P) of the corresponding base being determined wrong

$Q = -10 * log_{10}(P_{err})$ (rounded to nearest integer)

- The *Ph* in Phred comes from Phil Green, the inventor of the encoding
- P might be computed in different ways, depending on the sequencing platform, but the meaning of Q remains the same

Peter J. A. Cock, Christopher J. Fields, Naohisa Goto, Michael L. Heuer, and Peter M. Rice, Nucleic Acids Res. 2010
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2847217/

http://www.gs.washington.edu/faculty/green.htm

FASTQ quality scores (Phred scores)

PHRED Base quality (Q) – integer value derived from the estimated probability (P) of the corresponding base being determined wrong

- A higher quality score is better (>=20 is considered "good")

- $Q = 0 \sim P = 10^{0} = 1$
- $Q = 10 \sim P = 10^{-1} = 0.1$
- $Q = 20 \sim P = 10^{-2} = 0.01$
- $Q = 40 \sim P = 10^{-4} = 0.0001$

ASCII table

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	Start of Header	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	Start of Text	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	End of Text	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	End of Transmission	36	24	044	&#036;	$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enquiry	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Acknowledgment	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Backspace	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	Horizontal Tab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	Line feed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	Vertical Tab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	Form feed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	Carriage return	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	Shift Out	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	Shift In	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	Data Link Escape	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	Device Control 1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	Device Control 2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	Device Control 3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	Device Control 4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	Negative Ack.	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Synchronous idle	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	End of Trans. Block	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Cancel	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	End of Medium	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Substitute	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Escape	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	File Separator	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	|
29	1D	035	Group Separator	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	Record Separator	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	Unit Separator	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Del

asciicharstable.com

$$ASCII(char) = Q + 33$$

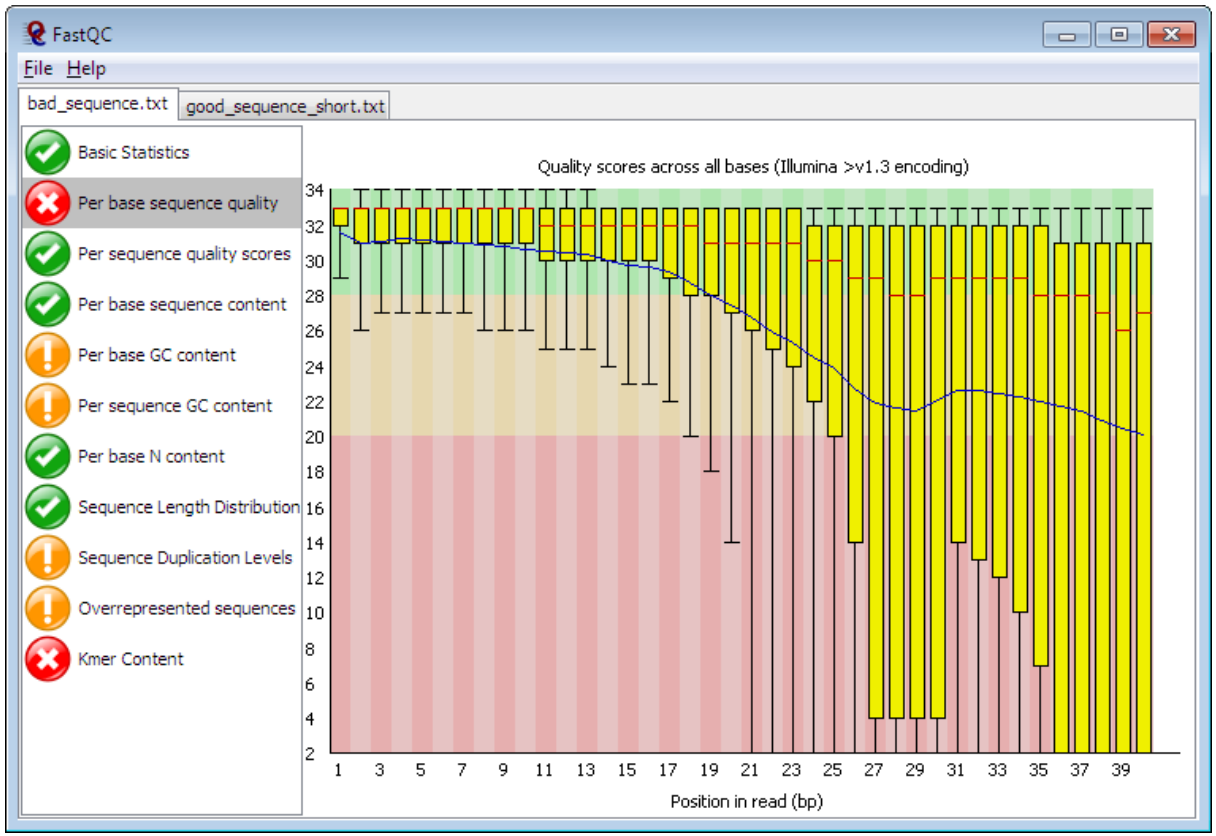
Older FASTQ has had different encoding schemes for encoding PHRED quality scores.

```
S - Sanger        Phred+33,  raw reads typically (0, 40)
X - Solexa        Solexa+64, raw reads typically (-5, 40)
I - Illumina 1.3+ Phred+64,  raw reads typically (0, 40)
J - Illumina 1.5+ Phred+64,  raw reads typically (3, 40)
    with 0=unused, 1=unused, 2=Read Segment Quality Control Indicator (bold)
    (Note: See discussion above).
L - Illumina 1.8+ Phred+33,  raw reads typically (0, 41)
```

- Phred + 33 is currently used

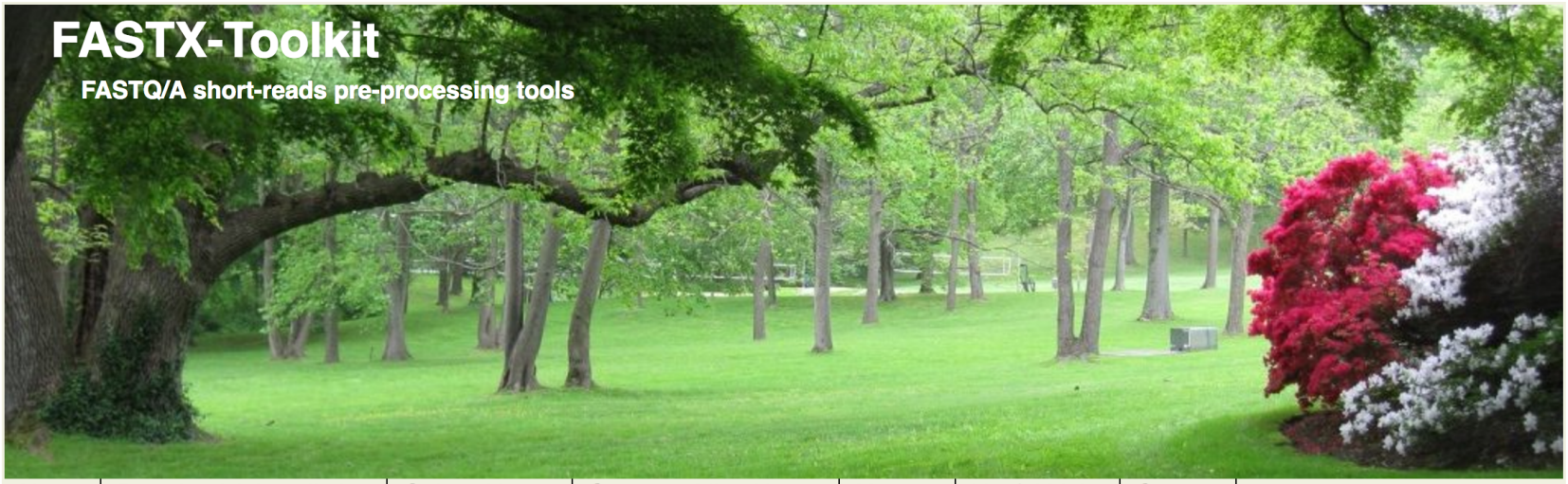
FASTQC - quality control

- Java program, HTML output



- http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
- FASTQC manual: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/
- Video tutorial how to interpret, https://www.youtube.com/watch?v=bz93ReOv87Y

fastx toolkit: manipulating FASTQ and FASTA files



Home | Download & Installation | Galaxy Usage | Command-line Usage | License | Useful Links | Contact

Introduction

The FASTX-Toolkit is a collection of command line tools for Short-Reads FASTA/FASTQ files preprocessing.

Next-Generation sequencing machines usually produce FASTA or FASTQ files, containing multiple short-reads sequences (possibly with quality information).

The main processing of such FASTA/FASTQ files is mapping (aka aligning) the sequences to reference genomes or other databases using specialized programs. Example of such mapping programs are: Blat, SHRiMP, LastZ, MAQ and many many others.

However,
It is sometimes more productive to preprocess the FASTA/FASTQ files before mapping the sequences to the genome - manipulating the sequences to produce better mapping results.

http://hannonlab.cshl.edu/fastx_toolkit/

fastx toolkit: manipulating FASTQ and FASTA files

```
fastx_quality_stats -h
fastx_quality_stats -Q 0 -i file.fastq -o fastx_report.txt
head fastx_report.txt
fastx_trimmer -h
fastx_trimmer -f 1 -l 100 -i file.fastq -o file_trimmed.fastq
```

The Sequence Alignment/Map format (SAM)

- Intended for storing read alignments against reference sequences
- Has a binary version with good software support (BAM format)

The SAM format consists of two sections:

Header section

- Used to describe source of data, reference sequence, method of alignment, etc.

Alignment section

- Used to describe the read, quality of the read, and nature alignment of the read to a region of the genome

SAM format specification https://samtools.github.io/hts-specs/SAMv1.pdf

BAM file format of aligned data

- BAM is the binary version of a SAM file. Smaller, but not easily readable.
- Compressed using lossless BGZF format
- Other BAM compression strategies are a subject of research. See 'CRAM' format for example

http://www.internationalgenome.org/faq/what-are-cram-files/

BAM file format of aligned data

- BAM files are usually indexed. An index is stored alongside the BAM file with a ".bai" extension
- Indexing aims to achieve fast retrieval of alignments overlapping a specified region without going through the whole alignments.
- BAM must be sorted before indexing. Depending on the downstream tools, sort by
  - Name
  - Coordinate

SAM example

Example SAM/BAM header section (abbreviated)

```
mgriffit@linus270 ~> samtools view -H /gscmnt/gc13001/info/model_data/2891632684/build136494552/alignments/136080019.bam  | grep -P "SN\:22|HD|RG|PG"
@HD     VN:1.4  SO:coordinate
@SQ     SN:22   LN:51304566     UR:ftp://ftp.ncbi.nih.gov/genbank/genomes/Eukaryotes/vertebrates_mammals/Homo_sapiens/GRCh37/special_requests/GRCh37-lite.fa.gz AS:GRCh37-lite  M5:a718acaa
4211dd  SP:Homo sapiens
@RG     ID:2888721359   PL:illumina     PU:D1BA4ACXX.3  LB:H_KA-452198-0817007-cDNA-3-lib1      PI:365  DS:paired end   DT:2012-10-03T19:00:00-0500     SM:H_KA-452198-0817007  CN:WUGSC
@PG     ID:2888721359   VN:2.0.8        CL:tophat --library-type fr-secondstrand --bowtie-version=2.1.0
@PG     ID:MarkDuplicates       PN:MarkDuplicates       PP:2888721359   VN:1.85(exported)       CL:net.sf.picard.sam.MarkDuplicates INPUT=[/gscmnt/gc13001/info/build_merged_alignments/merg
e10-2-5.gsc.wustl.edu-jwalker-15434-136080019/scratch-ILg6Y/H_KA-452198-0817007-cDNA-3-lib1-2888360300.bam] OUTPUT=/gscmnt/gc13001/info/build_merged_alignments/merged-alignment-blade10-2-5
alker-15434-136080019/scratch-ILg6Y/H_KA-452198-0817007-cDNA-3-lib1-2888360300-post_dup.bam METRICS_FILE=/gscmnt/gc13001/info/build_merged_alignments/merged-alignment-blade10-2-5.gsc.wustl
4-136080019/staging-1iuJS/H_KA-452198-0817007-cDNA-3-lib1-2888360300.metrics REMOVE_DUPLICATES=false ASSUME_SORTED=true MAX_FILE_HANDLES_FOR_READ_ENDS_MAP=9500 TMP_DIR=[/gscmnt/gc13001/in
ignments/merged-alignment-blade10-2-5.gsc.wustl.edu-jwalker-15434-136080019/scratch-ILg6Y] VALIDATION_STRINGENCY=SILENT MAX_RECORDS_IN_RAM=500000    PROGRAM_RECORD_ID=MarkDuplicates PROGRA
Duplicates MAX_SEQUENCES_FOR_DISK_READ_ENDS_MAP=50000 SORTING_COLLECTION_SIZE_RATIO=0.25 READ_NAME_REGEX=[a-zA-Z0-9]+:[0-9]:([0-9]+):([0-9]+):([0-9]+).* OPTICAL_DUPLICATE_PIXEL_DISTANCE=10
QUIET=false COMPRESSION_LEVEL=5 CREATE_INDEX=false CREATE_MD5_FILE=false
mgriffit@linus270 ~>
```

Example SAM/BAM alignment section (only 10 alignments shown)

```
mgriffit@linus270 ~> samtools view -f 3 -F 1804 /gscmnt/gc13001/info/model_data/2891632684/build136494552/alignments/136080019.bam | head
HWI-ST495_129147882:3:2114:15769:38646  99      1       11306   3       100M    =       11508   302     ACTGCGGGGCCCTCTTGCTTACTGTATAGTGGTGGCACGCCGCCTGCTGGCAGCTAGGGACATTGCAGGGTCCTCTTGCTCAA
CCFFFFFHHHGHJJJJJJIJJJHGIJJIJJHIIJJJJJJHFDDDDDDDDDDDDDCDDDDDDDDDDDDD<CDDDDDDDDCCCC>@>CDDDDDD?1<B     CC:Z:15 MD:Z:5A94       PG:Z:MarkDuplicates     RG:Z:2888721359 XG:i:0  NH:i:2  HI:
1       XN:i:0  XO:i:0  CP:i:102519765  AS:i:-5 XS:A:+  YT:Z:UU
HWI-ST495_129147882:3:2114:15769:38646  147     1       11508   3       100M    =       11306   -302    ACTCCTAAATATGGGATTCCTGGGTTTAAAAGTATAAAATAAATATGTTTAATTTGTGAACTGATTACCATCAGAATTGTACT
;5:CDCDCDECEFCD@9E=?7EEIIIIHCEGGIJJJJIIJJIHF@?00IHHFFGG?*JJJIJGHGEIJJIJJJJJJIHHCIEJJJHFHHGHFFEDFCCB     CC:Z:15 MD:Z:34A65      PG:Z:MarkDuplicates     RG:Z:2888721359 XG:i:0  NH:i:2  HI:
1       XN:i:0  XO:i:0  CP:i:102519563  AS:i:-6 XS:A:+  YT:Z:UU
HWI-ST495_129147882:3:1210:1257:16203   163     1       11810   3       100M    =       12055   345     CCTGCATGTAGTTTAAACGAGATTGCCAGCACCGGGTATCATTCACCATTTTTCTTTTCGTTAACTTGCCGTCAGCCTTTTCT
CCFFFFFHFHAFGGIIIJJJEEHGIGGGIJIJJGI?@EHIGIJDGHIHIGGIJJJJJJJIJGHHHGHFFFCDDDDDDCDCCCCCA;>@>@AA@:AA>AA     CC:Z:15 MD:Z:100        PG:Z:MarkDuplicates     RG:Z:2888721359 XG:i:0  NH:i:2  HI:
0       XN:i:0  XO:i:0  CP:i:102519261  AS:i:0  XS:A:-  YT:Z:UU
HWI-ST495_129147882:3:1210:1257:16203   83      1       12055   3       100M    =       11810   -345    GAGCACTGGAGTGGAGTTTTCCTGTGGAGAGGAGCCATGCCTAGAGTGGGATGGGCCATTGTTCATCTTCTGGCCCCTGTTGT
CC>4C>DCCCACACDCC?BDCEE@ECFFFFHHHHHIJJJIIJJIIIHHEHIGJIJIJJIGHIIIJJJJJIIJJJJJIIJJIJIJJHGHHHDFEFFCCC     CC:Z:15 MD:Z:100        PG:Z:MarkDuplicates     RG:Z:2888721359 XG:i:0  NH:i:2  HI:
0       XN:i:0  XO:i:0  CP:i:102519016  AS:i:0  XS:A:+  YT:Z:UU
HWI-ST495_129147882:3:2111:3117:78828   163     1       12634   3       100M    =       12746   212     GCCCTTCCCCAGCATCAGGTCTCCAGAGCTGCAGAAGACGACGGCCGACTTGGATCACACTCTTGTGAGTGTCCCCAGTGTTG
@@FFFFFDHHHH9FHGIIFGAFDHEGII>GHIIIIIIIIIIIIIIIFHDDFFEEECEECCCACCCCCC:AADCCBCC>CAC<CCCCCC:@CB@@BAB##     CC:Z:15 MD:Z:85G14      PG:Z:MarkDuplicates     RG:Z:2888721359 XG:i:0  NH:i:2  HI:
1       XN:i:0  XO:i:0  CP:i:102518437  AS:i:-5 XS:A:-  YT:Z:UU
HWI-ST495_129147882:3:2111:3117:78828   83      1       12746   3       100M    =       12634   -212    GGGAGTGGCGTCGCCCCTAGGGCTCTACGGGGCCGGCATCTCCTGTCTCCTGGAGAGGCTTCGATGCCCCTCCACACCCTCTT
DCABDBDDDDDDDDDDCDDDDBDB@BDDDB@;CCCCCDEFD@;.?<HIGGEIGEHIGJJJIIGIGIIHEGFEHFJIIIIIGJJJJHHHHHFFFFFC@@     CC:Z:15 MD:Z:37G62      PG:Z:MarkDuplicates     RG:Z:2888721359 XG:i:0  NH:i:2  HI:
1       XN:i:0  XO:i:0  CP:i:102518325  AS:i:-5 XS:A:-  YT:Z:UU
HWI-ST495_129147882:3:1102:4242:26638   99      1       13503   3       100M    =       13779   376     CGCTGTGCCCTTCCTTTGCTCTGCCCGCTGGAGACGGTGTTTGTCATGGGCCTGGTCTGCAGGGATCCTGCTACAAAGGTGAA
CCFFFFFHHHHHJJIJJJJJJJJJJJJJGIIIIJFHGGIJGIJJEGIJIJJHHIHHGHFFEFDEEEECCCAACDDACDCDDDDDB?8?<B>A@CDC     CC:Z:2  MD:Z:100        PG:Z:MarkDuplicates     RG:Z:2888721359 XG:i:0  NH:i:2  HI:
0       XN:i:0  XO:i:0  CP:i:114357414  AS:i:0  XS:A:+  YT:Z:UU
HWI-ST495_129147882:3:1309:15328:74082  99      1       13534   3       100M    =       13780   346     AGACGGTGTTTGTCATGGGCCTGGTCTGCAGGGATCCTGCTACAAAGGTGAAACCCAGGAGAGTGTGGAGTCCAGAGTGTTGC
CCFFFADHHHHFIJJJJJIJIJIHIJJJIHJJIJJIIJJIIJIJJJJJBFHIIJJJJJJIJHH=EEFFFFCEEECEDCDCDDDDDDDDDDDDDDDDBDCCD     CC:Z:2  MD:Z:100        PG:Z:MarkDuplicates     RG:Z:2888721359 XG:i:0  NH:i:2  HI:
0       XN:i:0  XO:i:0  CP:i:114357383  AS:i:0  XS:A:+  YT:Z:UU
HWI-ST495_129147882:3:1308:10126:19636  99      1       13779   3       100M    =       14027   348     CCTCTGCAGGAGGCTGCCATTTGTCCTGCCCACCTTCTTAGAAGCGAGACGGAGCAGACCCATCTGCTACTGCCCTTTCTATA
CCFFFFFHHGHHJJJJJJJJJJIJJJJJJIJJJJJJJJJIJIJJJJJJJJJHJJJHGHHFFFFFFEEEEEEDDDDCDCDCDFADCACCACDDCCCCCCD     CC:Z:2  MD:Z:100        PG:Z:MarkDuplicates     RG:Z:2888721359 XG:i:0  NH:i:2  HI:
0       XN:i:0  XO:i:0  CP:i:114357140  AS:i:0  XS:A:+  YT:Z:UU
```

SAM/BAM header section

- Used to describe source of data, reference sequence, method of alignment, etc.
- Each section begins with '@' followed by a two-letter record type code. These are followed by two-letter tags and values

```
@HD The header line
VN: format version
SO: Sorting order of alignments
@SQ Reference sequence dictionary
SN: reference sequence name
LN: reference sequence length
SP: species
@RG Read group
ID: read group identifier
CN: name of sequencing center
SM: sample name
@PG Program
PN: program name
```

SAM/BAM alignment section

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	`[!-?A-~]{1,254}`	Query template NAME
2	FLAG	Int	$[0, 2^{16}-1]$	bitwise FLAG
3	RNAME	String	`\*|[!-()+-<>-~][!-~]*`	Reference sequence NAME
4	POS	Int	$[0, 2^{31}-1]$	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 read
8	PNEXT	Int	$[0, 2^{31}-1]$	Position of the mate/next read
9	TLEN	Int	$[-2^{31}+1, 2^{31}-1]$	observed Template LENgth
10	SEQ	String	`\*|[A-Za-z=.]+`	segment SEQuence
11	QUAL	String	`[!-~]+`	ASCII of Phred-scaled base QUALity+3

https://samtools.github.io/hts-specs/SAMv1.pdf

Using SAM flags to filter subsets of reads

- 12 bitwise flags describing the alignment
- "Bitwise" – values contributing to the sum are powers of two (each can be represented by a single positive bit in a binary number)
- These flags are stored as a binary string of length 11
- Value of "1" indicates the flag is set. e.g. 00100000000
- All combinations can be represented as a number from 1 to 2048 (i.e. $2^{11} - 1$). This number is used in the BAM/SAM file. You can specify "required" or "filter" flags in `samtools view` command using the `-f` and `-F` options, respectively

Using SAM flags to filter subsets of reads

- Unambiguous – there is exactly one possible decomposition of each FLAG sum
- Each value contributing to the sum represents one fulfilled condition

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

https://broadinstitute.github.io/picard/explain-flags.html

https://samtools.github.io/hts-specs/SAMv1.pdf

SAM – bitwise FLAG (example)

ST-E00223:32:H5J57CCXX:4:1220:14651:8868 **99** 1 10086

base2	base10	base16	Meaning	Applies to:
00000000001	1	0x0001	The read originated from a paired sequencing molecule	Both
00000000010	2	0x0002	The read is mapped in a **proper** pair	Pairs only
00000000100	4	0x0004	The query sequence itself is unmapped	Both
00000001000	8	0x0008	The query's mate is unmapped	Pairs only
00000010000	16	0x0010	Strand of the query (0 for forward; 1 for reverse strand)	Both
00000100000	32	0x0020	Strand of the query's mate	Pairs only
00001000000	64	0x0040	The query is the first read in the pair	Pairs only
00010000000	128	0x0080	The read is the second read in the pair	Pairs only
00100000000	256	0x0100	The alignment is not primary	Both
01000000000	512	0x0200	The read fails platform/vendor quality checks	Both
10000000000	1024	0x0400	The read is either a PCR duplicate or an optical duplicate	Both

00001100011

$2^6+2^5+2^1+2^0 = 64+32+2+1 = 99$

SAM – bitwise FLAG (example)

FLAG 83

- 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

CIGAR string

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

- The CIGAR string is a sequence of base lengths and associated "operations" that are used to indicate which bases align to the reference (either a match or mismatch), are deleted, are inserted, represent introns, etc.
  - e.g. 81M859N19M
  - Read as: A 100 bp read consists of: 81 bases of alignment to reference, 859 bases skipped (an intron), 19 bases of alignment

https://samtools.github.io/hts-specs/SAMv1.pdf

Tools to work with SAM/BAM files

- **samtools** - view, sort, index, QC, stats on SAM/BAM files, and more
- **sambamba** - view, sort, index, merge, stats, mark duplicates. fast laternative to `samtools`
- **picard** - QC, validation, duplicates removal and many more utility tools

https://github.com/samtools/samtools

https://lomereiter.github.io/sambamba/index.html

https://broadinstitute.github.io/picard/

SAMTOOLS Commands: Manipulating SAM/BAM

```
-- Indexing
   dict - create a sequence dictionary file
   faidx - index/extract FASTA
   index - index alignment
-- Editing
   calmd - recalculate MD/NM tags and '=' bases
   fixmate - fix mate information
   reheader - replace BAM header
   rmdup - remove PCR duplicates
   targetcut - cut fosmid regions (for fosmid pool only)
   addreplacerg - adds or replaces RG tags
-- Viewing
   flags - explain BAM flags
   tview - text alignment viewer
   view - SAM<->BAM<->CRAM conversion
   depad - convert padded BAM to unpadded BAM
```

SAMTOOLS Commands: Manipulating SAM/BAM

```
-- File operations
   collate - shuffle and group alignments by name
   cat - concatenate BAMs
   merge - merge sorted alignments
   mpileup - multi-way pileup
   sort - sort alignment file
   split - splits a file by read group
   quickcheck - quickly check if SAM/BAM/CRAM file appears intact
   fastq - converts a BAM to a FASTQ
   fasta - converts a BAM to a FASTA
-- Statistics
   bedcov - read depth per BED region
   depth - compute the depth
   flagstat - simple stats
   idxstats - BAM index stats
   phase - phase heterozygotes
   stats - generate stats (former bamcheck)
```

BED format

- Text-based, tab-separated 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 – E.g. strand, name, score, color
- Intended for visualizing genomic annotations in IGV, UCSC Genome Browser (context of expression, regulation, variation, conservation, . . .)

http://genome.ucsc.edu/FAQ/FAQformat.html#format1

BED format

3 mandatory columns (must be in correct order)

- "chrom" – chromosome
- "chromStart" – the first base of the region with respect to the chromosome (counting starts from 0)
- "chromEnd" – the first base after the region with respect to the chromosome
- [chromStart, chromEnd) allows easy region-length calculation
- Optional fields: "name", "score", "strand", other annotation columns

```
chr1  115263684 115263685 rs10489525  0 +
chr12 97434219  97434220  rs6538761   0 +
chr14 102360744 102360745 rs7142002   0 +
chr16 84213683  84213684  rs4150167   0 -
chr2  206086170 206086171 rs4675502   0 +
chr20 14747470  14747471  rs4141463   0 +
```

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 (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")
- BED12 format specification available

https://genome.ucsc.edu/FAQ/FAQformat.html#format1.7

Tools to work with BED files

- **bedtools** - universal tools for manipulating genomic regions
- **bedops** - complementary to `bedtools`, providing additional functionality and speedup

https://bedtools.readthedocs.io/en/latest/

https://bedtools.readthedocs.io/en/latest/

GFF/GTF file format

- Generic feature format for storing genomic annotation data
- **GFF** - Generic Feature Format - describes gene-centric elements
- **GTF** - Gene Transfer Format - similar to GFF, but more restrictive

- GFF specifications: https://github.com/The-Sequence-Ontology/Specifications/blob/master/gff3.md - GTF specifications: http://www.gencodegenes.org/gencodeformat.html - More about file formats, http://journals.plos.org/ploscompbiol/article/file?type=supplementary&id=info:doi/10.1371/journal.pcbi.1004393.s008

GFF/GTF file format

- Tab delimited text file (with optional header lines beginning ##):
  - contig (chromosome)
  - source
  - type
  - start
  - end
  - score
  - strand
  - phase
  - attributes

```
wget ftp://ftp.ensembl.org/pub/release-89/gtf/homo_sapiens/Homo_sapiens.GRCh38.89.gtf.gz
```

GTF gene annotations: http://www.ensembl.org/info/data/ftp/index.html

WIG format

- Text format to represent continuous signal - “wiggle” format
- variable step format - for data with irregular intervals

```
variableStep  chrom=chrN
[span=windowSize]
  chromStartA  dataValueA
  chromStartB  dataValueB
  ... etc ...  ... etc ...
```

- fixed step format - for data with regular intervals

```
fixedStep  chrom=chrN
start=position  step=stepInterval
[span=windowSize]
  dataValue1
  dataValue2
  ... etc ...
```

Tools to work with WIG format

- **Wiggler** - converts aligner reads to continuous WIG signal
- **WiggleTools** - Basic operations on the space of numerical functions defined on the genome. Works with multiple genomic file formats. Operates on single file or pairs of files

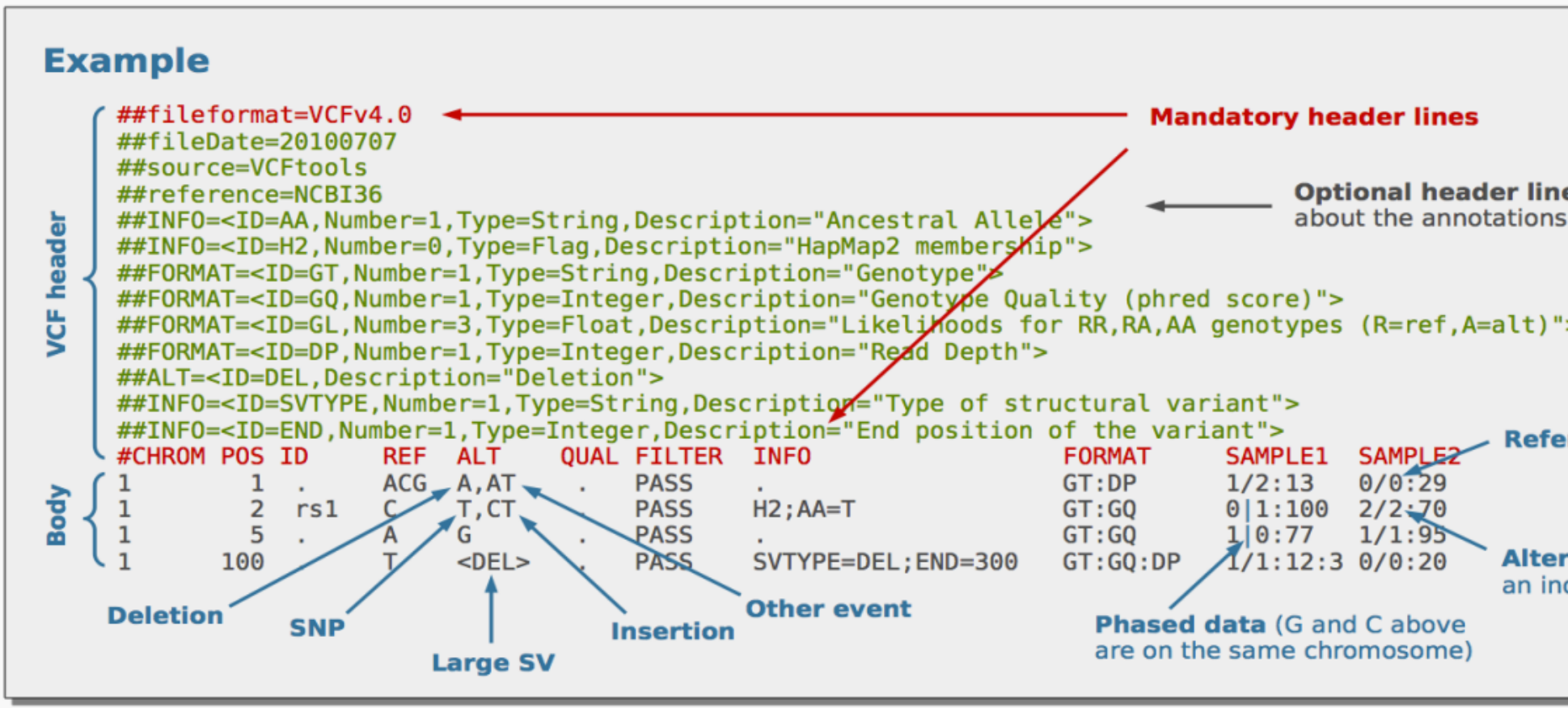
https://genome.ucsc.edu/goldenpath/help/wiggle.html

https://code.google.com/archive/p/align2rawsignal/

https://github.com/Ensembl/WiggleTools

VCF format

- Variant Call Format (VCF) is a flexible and extendable format for variation data



VCF format

- CHROMO: chromosome / contig
- POS: the reference position with the 1st base having position 1
  - The value in POS refers to the position of the first base in the string. For indels, the reference string must include the base before the event (and this must be reflected in POS)
- ID: an id; rs number if dbSNP variant
- REF: reference base
- ALT: comma sepearated list of alternate non-ref alleles called on at least one of the samples
  - If no alternate alleles then the missing value should be used "."
- QUAL: phred-scaled quality score of the assertion made in ALT (whether variant or non-variant)
- FILTER: PASS if the position has passed all filters (defined in meta-data)
- INFO: additional information

VCF Format

BIOINFORMATICS APPLICATIONS NOTE

Vol. 27 no. 15 2011, pages 2156–2158
doi:10.1093/bioinformatics/btr330

Sequence analysis

Advance Access publication June 7, 2011

The variant call format and VCFtools

Petr Danecek[1,†], Adam Auton[2,†], Goncalo Abecasis[3], Cornelis A. Albers[1], Eric Banks[4], Mark A. DePristo[4], Robert E. Handsaker[4], Gerton Lunter[2], Gabor T. Marth[5], Stephen T. Sherry[6], Gilean McVean[2,7], Richard Durbin[1,*] and 1000 Genomes Project Analysis Group[‡]

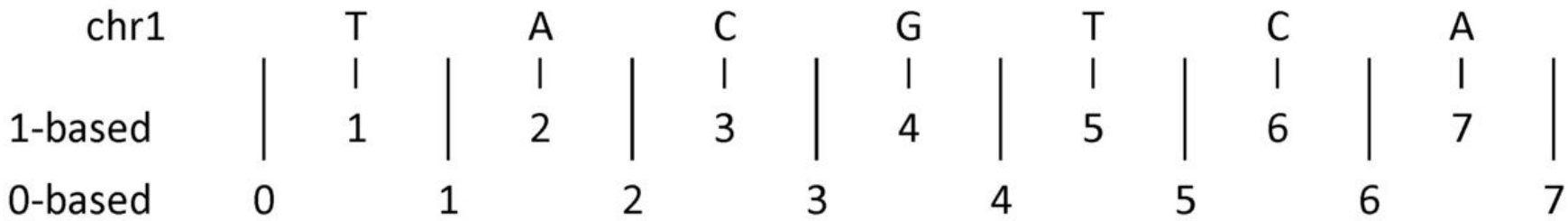
[1]Wellcome Trust Sanger Institute, Wellcome Trust Genome Campus, Cambridge CB10 1SA, [2]Wellcome Trust Centre for Human Genetics, University of Oxford, Oxford OX3 7BN, UK, [3]Center for Statistical Genetics, Department of Biostatistics, University of Michigan, Ann Arbor, MI 48109, [4]Program in Medical and Population Genetics, Broad Institute of MIT and Harvard, Cambridge, MA 02141, [5]Department of Biology, Boston College, MA 02467, [6]National Institutes of Health National Center for Biotechnology Information, MD 20894, USA and [7]Department of Statistics, University of Oxford, Oxford OX1 3TG, UK

Associate Editor: John Quackenbush

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3137218/pdf/btr330.pdf

Formats use different coordinate systems

- **BED**: 0-based, half-open
- **GFF**: 1-based, closed
- **SAM**: 1-based, closed
- **BAM**: 0-based, half-open
- **VCF**: 1-based, closed



https://www.biostars.org/p/84686/

Formats use different chromosome naming conventions

- The UCSC and Ensembl databases name their chromosomes differently.
- By convention, UCSC chromosomes start with chr while Ensembl chromosome names do not.
- UCSC will call the fifth chromosome chr5 and Ensembl will call it 5