BF528 - Biological Data Formats

Introduction

- Many different types of data
- Standard formats exist for many of them
- Always use/extend standard formats
- Don’t save data in non standard formats when standards are available - **No .xslx!**
- For complete list of formats see: https://genome.ucsc.edu/FAQ/FAQformat.html

Nucleotide/amino acids sequences

Any sequence of nucleotides/amino acids is saved in FASTA format.

```
>sequence_header [comments]
sequence_line1
sequence_line2
sequence_line3
...
```

Lower case letters means they have been masked.

FASTA format

```
>AT1G27360.1
AAGGTATCTATTTGCCTAGCCAGAGTTATATATAGGATTGATTGTCTAGTCTTTTCTTAT
ATGATTTTTGTTCTCATTTACTAATCAAAGTTCTGCAAACTTGTAGTTGTTGTAGGATTT
GTTGCTCTGGCTCTGGTGGTAGGTCTATGAAATCAACCCATATCGTGAATGGACTGCAAC
ATGGTATCTTCGTCCCAGTGGGA
>AT1G27360.4 squamosa promoter-binding protein-like 11
CTGGGTGAAACATAGAAAAGTTTCTCTTGCTCAAGTTAATGATAAAAGGGTGAGAGCAAT
AAACGCTGATAAGCCTTGTCTGGTCCTTGGAATTTTGAATTTTCTTTTTCTATCTTACTT
ATAGTATTGGTAGTTGAGGGTGTCGTCGATAAGTTGTTGTAGGATTTGTTGCTCTGGCTC
TGGTGGT
>AT1G32140.1
ATGACGATGATGTCCGACCTTTCACTTGATTTAGTCGAAGAGATATTGTGTAGGGTTCCG
ATAACTTCTCTTAAAGCAGTGAGATCTAGTTGCAAACTATGGAACGTTCTTTCCAAGAAC
CG
>ath-miR398b
UGUGUUCUCAGGUCACCCCUG
>ath-miR398c
UGUGUUCUCAGGUCACCCCUG
```

Sequencing reads: FASTQ

Each read/read pair should have an unique name.

A read has the sequence and the quality of each nucleotide sequences.

```
@read_name [comments]
sequences_nucleotides
+read_name [comments]
quality_of_each_nucleotide_in_ASCII
```

FASTQ format: single end

fastq_SRR1997469.fastq

```
@SRR1997469.1 1 length=36
CAGTCTTCTTAGAAATATCCACTTCGGAATAAAAGA
+SRR1997469.1 1 length=36
BBBBBFFFFF<FFFFFFFFFFFFFBFFFFFFFFFF
@SRR1997469.2 2 length=36
ACAGTTGAACGATCCTTTACAGANAGNAGNCTNGTA
+SRR1997469.2 2 length=36
<BBBBFFFBBF<BFF/<FFFFFF############
...
```

FASTQ format: paired end/mates

fastq_SRR1997469_1.fastq

```
@SRR1997469.1 1 length=36
CAGTCTTCTTAGAAATATCCACTTCGGAATAAAAGA
+SRR1997469.1 1 length=36
BBBBBFFFFF<FFFFFFFFFFFFFBFFFFFFFFFF
@SRR1997469.2 2 length=36
ACAGTTGAACGATCCTTTACAGANAGNAGNCTNGTA
+SRR1997469.2 2 length=36
<BBBBFFFBBF<BFF/<FFFFFF############
...
```

fastq_SRR1997469_2.fastq

```
@SRR1997469.1 1 length=36
AGATAAGATGGTAATCTTTGATGGAGAACATTAAGA
+SRR1997469.1 1 length=36
BBBBBFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SRR1997469.2 2 length=36
ACTGGGAANCCTTCTGTCTAGCCTTATATGAAAAAA
+SRR1997469.2 2 length=36
BBB/BFFB#BB/FFFF/FFFFFFBBFFFFFFFFFBF
...
```

Read alignments

- Alignments are kept in SAM format
- SAM is sorted and compressed into BAM or CRAM
- One line per alignment
- Use samtools to view, sort, merge, concatenate, index, get statistics, … on alignment files.

SAM/BAM/CRAM - header

The **header** lines start with @

- @HD → header definition
- @SQ → a sequence in the reference file you used, followed by how long it was and it's comment (from the reference file)
- @RG → read groups you assigned while mapping the reads
- @PG → programs used to obtain this bam, in order

SAM/BAM/CRAM - body

Each line keeps an alignment.

Each alignment has 11 mandatory fields:

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+33

SAM/BAM/CRAM - flag

The flag is the summation of the following binary attributes:

Binary (Decimal)	Hex	Description
00000000001 (1)	0x1	Is the read paired?
00000000010 (2)	0x2	Are both reads in a pair mapped "properly" (i.e., in the correct orientation with respect to one another)?
00000000100 (4)	0x4	Is the read itself unmapped?
00000001000 (8)	0x8	Is the mate read unmapped?
00000010000 (16)	0x10	Has the read been mapped to the reverse strand?
00000100000 (32)	0x20	Has the mate read been mapped to the reverse strand?
00001000000 (64)	0x40	Is the read the first read in a pair?
00010000000 (128)	0x80	Is the read the second read in a pair?
00100000000 (256)	0x100	Is the alignment not primary? (A read with split matches may have multiple primary alignment records.)
01000000000 (512)	0x200	Does the read fail platform/vendor quality checks?
10000000000 (1024)	0x400	Is the read a PCR or optical duplicate?

Example flags:

Single end read mapped to + strand: 0 (no flags apply)

Single end read mapped to - strand: 16

Unmapped, paired end first mate read with unmapped mate: 69 = 1 + 4 + 8 + 64

SAM/BAM/CRAM format

```
@HD     VN:1.0  SO:coordinate
@SQ     SN:chr20        LN:64444167
@PG     ID:TopHat       VN:2.0.14       CL:/srv/dna_tools/tophat/tophat -N 3 --read-edit-dist 5 --read-rea
lign-edit-dist 2 -i 50 -I 5000 --max-coverage-intron 5000 -M -o out /data/user446/mapping_tophat/index/chr
20 /data/user446/mapping_tophat/L6_18_GTGAAA_L007_R1_001.fastq
HWI-ST1145:74:C101DACXX:7:1102:4284:73714        16      chr20   190930  3       100M    *       0       0
       CCGTGTTTAAAGGTGGATGCGGTCACCTTCCCAGCTAGGCTTAGGGATTCTTAGTTGGCCTAGGAAATCCAGCTAGTCCTGTCTCTCAGTCCCCCCTCT
C    BBDCCDDCCDDDDCDDDDDDCDCCCDBC?DDDDDDDDDDDDDDCCDCDDDDDDDDDDCCCCEDDDC?DDDDDDDDDDDDDDDDDDDDDBDHFFFFDC@@
    AS:i:-15        XM:i:3  XO:i:0  XG:i:0  MD:Z:55C20C13A9 NM:i:3  NH:i:2  CC:Z:=  CP:i:55352714   HI:i:0
HWI-ST1145:74:C101DACXX:7:1114:2759:41961        16      chr20   193953  50      100M    *       0       0
       TGCTGGATCATCTGGTTAGTGGCTTCTGACTCAGAGGACCTTCGTCCCCTGGGGCAGTGGACCTTCCAGTGATTCCCCTGACATAAGGGGCATGGACGA
G    DCDDDDEDDDDDDDCDDDDDDDCCCDDDCDDDDDEEC>DFFFEJJJJJIGJJJJIHGBHHGJIJJJJJJGJJJIJJJJJIHJJJJJJHHHHHFFFFFCCC
    AS:i:-16        XM:i:3  XO:i:0  XG:i:0  MD:Z:60G16T18T3 NM:i:3  NH:i:1
HWI-ST1145:74:C101DACXX:7:1204:14760:4030        16      chr20   270877  50      100M    *       0       0
       GGCTTTATTGGTAAAAAAGGAATAGCAGATTTAATCAGAAATTCCCACCTGGCCCAGCAGCACCAACCAGAAAGAAGGGAAGAAGACAGGAAAAAACCA
C    DDDDDDDDDCCDDDDDDDDDDEEEEEEEFFFEFFEGHHHHFGDJJIHJJIJIJJJIIIIGGFJJIHIIIIJJJJJJIGHHFAHGFHJHFGGHFFFDD@BB
    AS:i:-11        XM:i:2  XO:i:0  XG:i:0  MD:Z:0A85G13    NM:i:2  NH:i:1
HWI-ST1145:74:C101DACXX:7:1210:11167:8699        0       chr20   271218  50      50M4700N50M     *       0
       0       GTGGCTCTTCCACAGGAATGTTGAGGATGACATCCATGTCTGGGGTGCACTTGGGTCTCCGAAGCAGAACATCCTCAAATATGACCTCTCG
accepted_hits.sam
```

SAM/BAM/CRAM format

- Use samtools view to see the content of a SAM/BAM/CRAM file.
- Always sort and index them
- Use samtools view -f (include) and -F (exclude) to filter by flags
- Use samtools view -q to filter by quality.

Convert BAM to FASTQ

- Some public datasets are provided in BAM format
- Need to extract the reads to process the raw data
- BAM (usually) contains all information from FASTQ

samtools:

```
samtools bam2fq [-nO] [-s <outSE.fq>] <in.bam>
```

Standard format for keeping tables

field1	field2	field3	...
...	...	...	...

Fields (columns) separated by a character on each line:

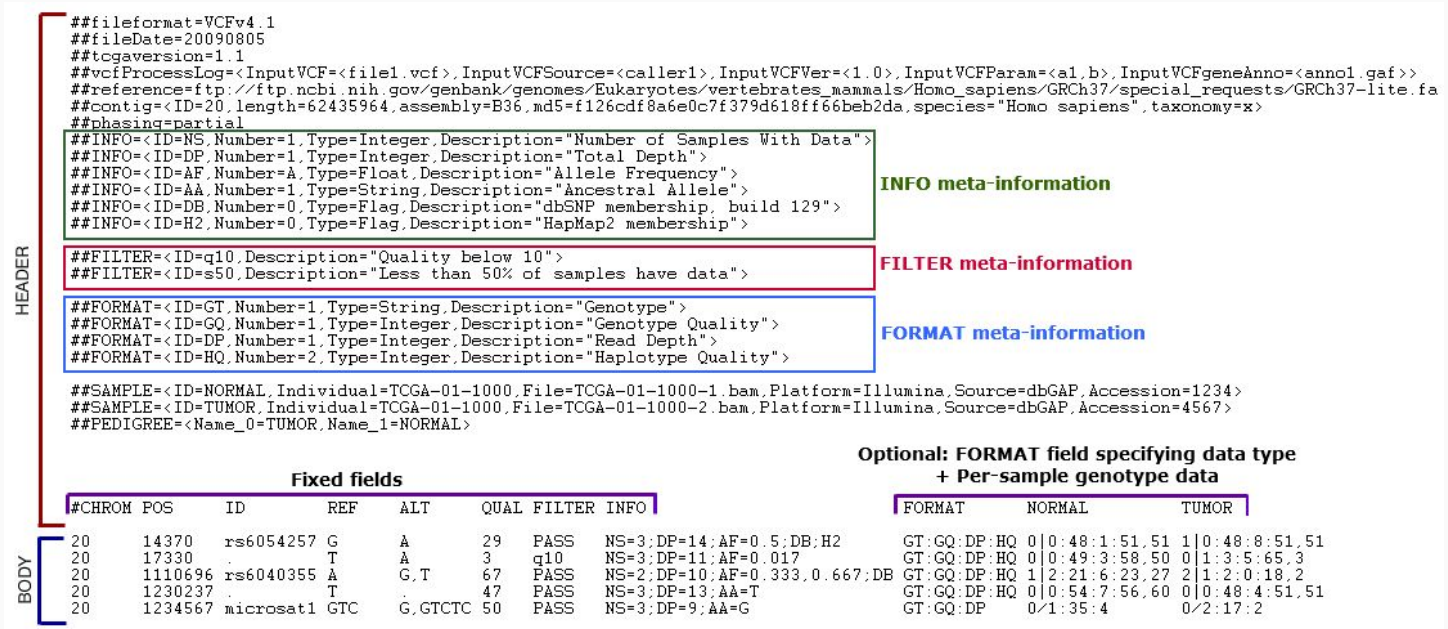
- Comma (or Character) Separated Vector (CSV)
- Tab Separated Vector (TSV)
- Some interpreters take any space (space or tab) as a separator (such as `awk`, `cut`).
- Some have column name as first row (header), some don't

Genomic variants

- Save them in VCF format
- VCF: Variant Call Format
- Current version: 4.0
- http://www.internationalgenome.org/wiki/Analysis/vcf4.0/
- Tab separated

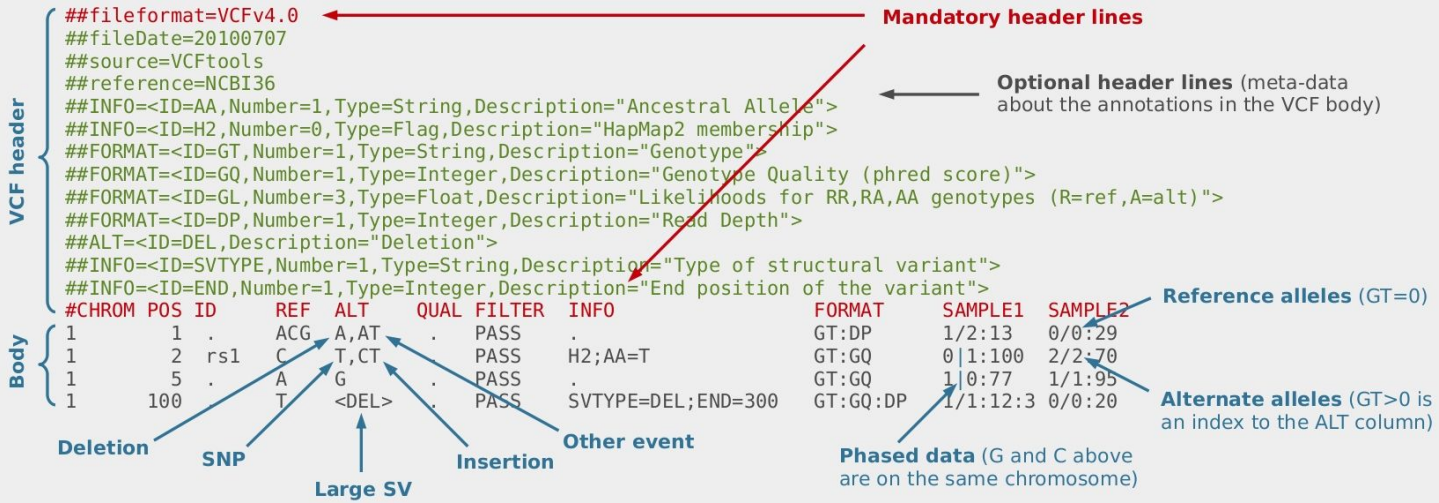
VCF format - header

- Mandatory header lines: information about the fields (columns) starting with ##INFO
- Extra: filtering, metadata, tools, ...



VCF format - body

- REF and ALT fields contain nucleotides in case of SNP and indels
- In case of large structural variants: <DEL> <INS> <DUP> <INV>



VCF format - example

```
##fileformat=VCFv4.1
##fileDate=20140930
##source=23andme2vcf.pl https://github.com/arrogantrobot/23
##reference=file://23andme_v3_hg19_ref.txt.gz
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype"
#CHROM  POS ID  REF ALT QUAL    FILTER  INFO    FORMAT  GEN
chr1    82154   rs4477212   a   .   .   .   .   GT  0/0
chr1    752566  rs3094315   g   A   .   .   .   GT  1/1
chr1    752721  rs3131972   A   G   .   .   .   GT  1/1
chr1    798959  rs11240777  g   .   .   .   .   GT  0/0
chr1    800007  rs6681049   T   C   .   .   .   GT  1/1
chr1    838555  rs4970383   c   .   .   .   .   GT  0/0
chr1    846808  rs4475691   C   .   .   .   .   GT  0/0
chr1    854250  rs7537756   A   .   .   .   .   GT  0/0
chr1    861808  rs13302982  A   G   .   .   .   GT  1/1
chr1    873558  rs1110052   G   T   .   .   .   GT  1/1
chr1    882033  rs2272756   G   A   .   .   .   GT  0/1
chr1    888659  rs3748597   T   C   .   .   .   GT  1/1
chr1    891945  rs13303106  A   G   .   .   .   GT  0/1
```

Genomic regions

- A region is defined by three required fields
  - **sequence name (e.g. chromosome)**
  - **start coordinate**
  - **end coordinate**
- Define regions of interest: introns, exons, genes, etc.
- Additional information saved as fields after the first three.
- Three standard tab-separated formats: BED, GFF, GTF
- No headers

BED format

Mandatory fields:

1. chrom - Name of the chromosome/scaffold/reference sequence
2. chromStart - 0-based starting position of the feature on chrom
3. chromEnd - Ending position of the feature in the chromosome or scaffold.

➔ The chromEnd base is not included in the display of the feature.
➔ For example, the first 100 bases of a chromosome are defined as:
- chromStart=0
- chromEnd=100
- span the bases numbered 0-99

BED format

Optional fields:

4. Name

5. Score

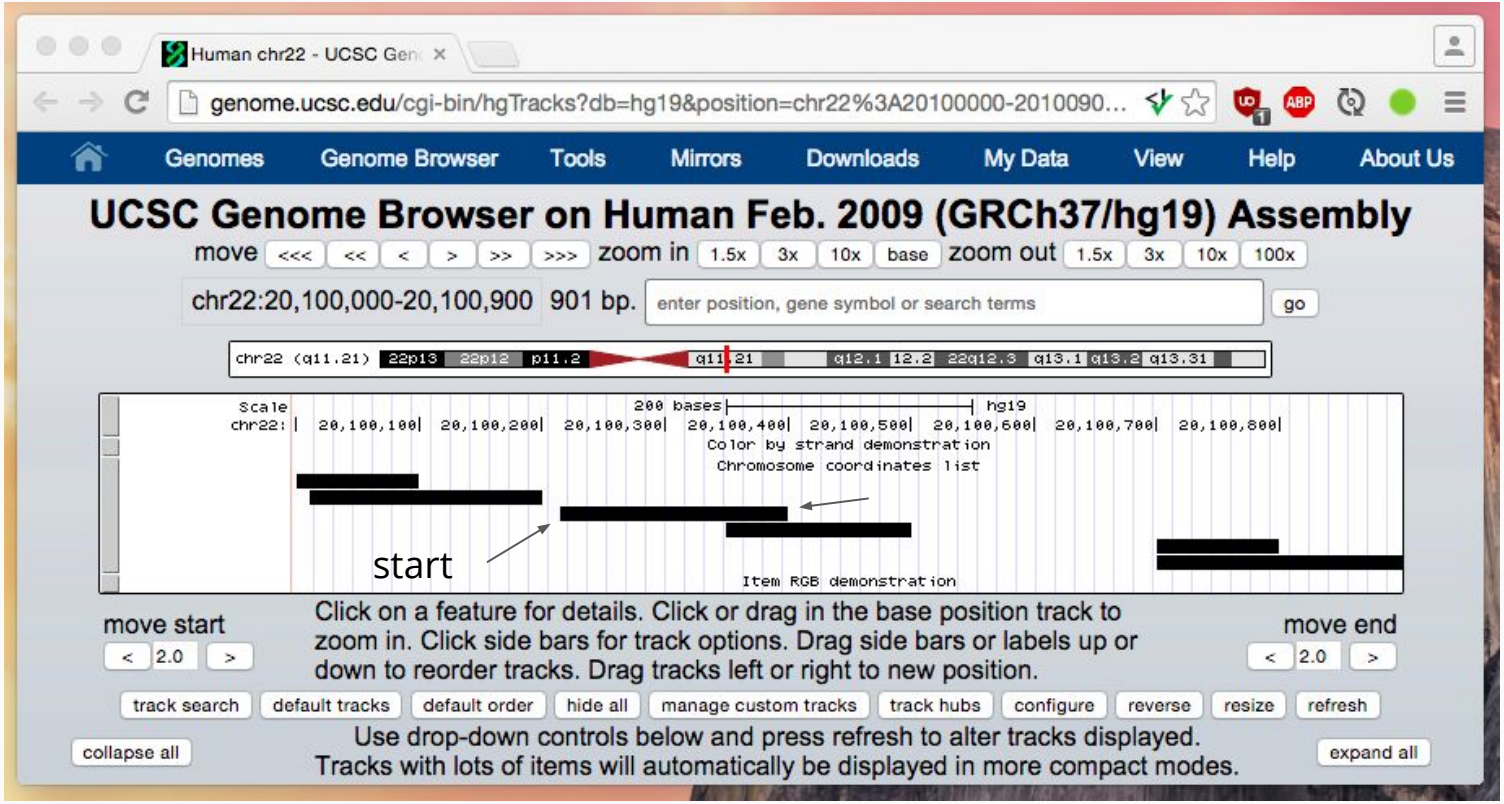
6. Strand

7-12. Display options (thick starts and end, color, blocks...) to control the view on the genome browser

BED format

chr1	11873	14409	uc001aaa.3	0	+	11873	11873
chr1	11873	14409	uc010nxr.1	0	+	11873	11873
chr1	11873	14409	uc010nxq.1	0	+	12189	13639
chr1	14361	16765	uc009vis.3	0	-	14361	14361
chr1	14361	19759	uc009vit.3	0	-	14361	14361
chr1	14361	19759	uc009viu.3	0	-	14361	14361
chr1	14361	19759	uc001aae.4	0	-	14361	14361
chr1	14361	29370	uc001aah.4	0	-	14361	14361
chr1	14361	29370	uc009vir.3	0	-	14361	14361
chr1	14361	29370	uc009viq.3	0	-	14361	14361
chr1	14361	29370	uc001aac.4	0	-	14361	14361
chr1	14406	29370	uc009viv.2	0	-	14406	14406
chr1	14406	29370	uc009viw.2	0	-	14406	14406
chr1	15602	29370	uc009vix.2	0	-	15602	15602
chr1	15795	18061	uc009vjd.2	0	-	15795	15795
chr1	16606	29370	uc009viy.2	0	-	16606	16606
chr1	16606	29370	uc009viz.2	0	-	16606	16606
chr1	16857	17751	uc009vjc.1	0	-	16857	16857
chr1	16857	19759	uc001aai.1	0	-	16857	16857

BED file records are intervals

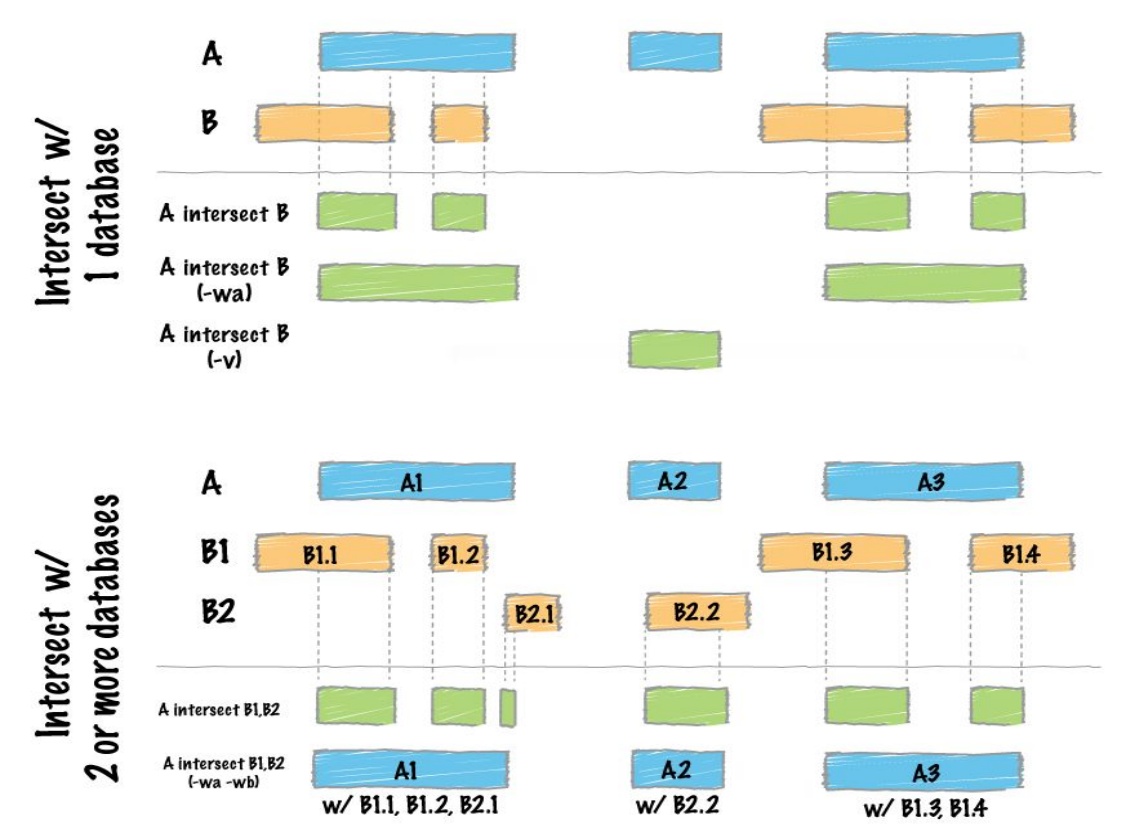


bedtools

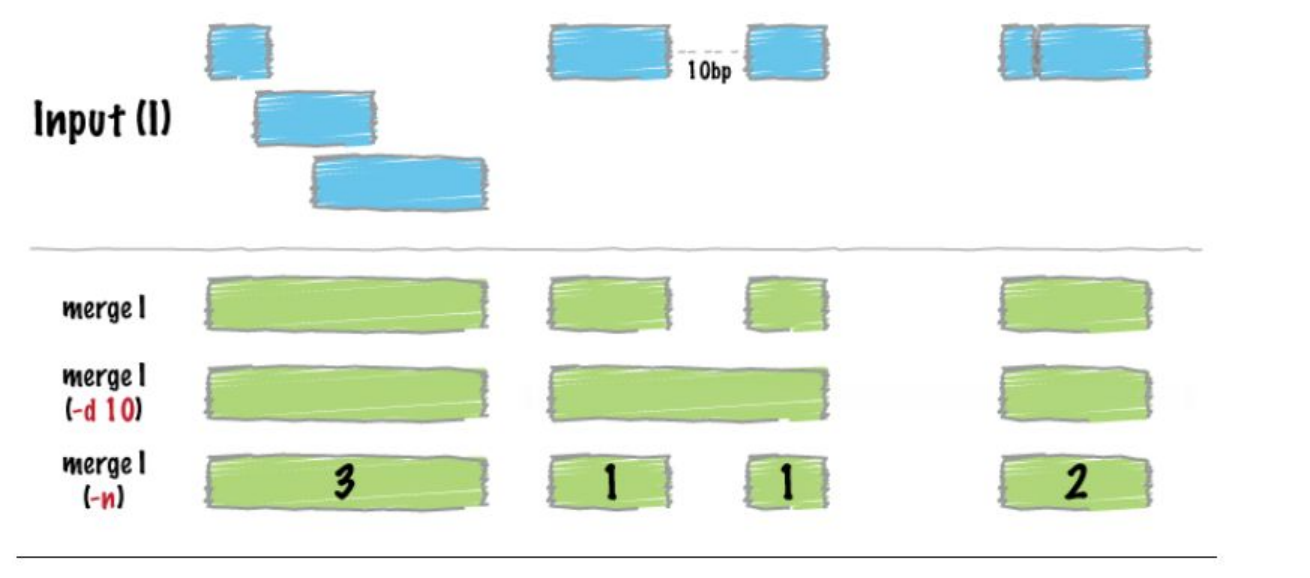
http://bedtools.readthedocs.io/

- sort (sort bed files)
- Intersect (get intersections of bed files)
- merge
- coverage
- overlap
- substract
- ...

bedtools - intersect



bedtools - merge



General features

9 mandatory fields, tab separated

1. **seqname** - The name of the sequence. Must be a chromosome or scaffold.
2. **source** - The program that generated this feature.
3. **feature** - The name of this type of feature (e.g. gene, exon, etc).
4. **start** - The starting position of the feature in the sequence (1-based)
5. **end** - The ending position of the feature (inclusive).
6. **score** - A score between 0 and 1000.
7. **strand** - Valid entries include "+", "-", or "." (for don't know/don't care).
8. **frame** - If the feature is a coding exon, *frame* should be a number between 0-2 that represents the reading frame of the first base. If the feature is not a coding exon, the value should be ".".
9. **group** - All lines with the same group are linked together into a single item.

General Feature Format (GFF) format

GFF2:

```
IV     curated  mRNA   5506800 5508917 . + .   Transcript B0273.1; Note "Zn-Finger"
IV     curated  5'UTR  5506800 5508999 . + .   Transcript B0273.1
IV     curated  exon   5506900 5506996 . + .   Transcript B0273.1
IV     curated  exon   5506026 5506382 . + .   Transcript B0273.1
IV     curated  exon   5506558 5506660 . + .   Transcript B0273.1
IV     curated  exon   5506738 5506852 . + .   Transcript B0273.1
IV     curated  3'UTR  5506852 5508917 . + .   Transcript B0273.1
```

GFF3:

```
##gff-version 3
ctg123  .  exon  1300  1500  .  +  .  ID=exon00001
ctg123  .  exon  1050  1500  .  +  .  ID=exon00002
ctg123  .  exon  3000  3902  .  +  .  ID=exon00003
ctg123  .  exon  5000  5500  .  +  .  ID=exon00004
ctg123  .  exon  7000  9000  .  +  .  ID=exon00005
```

Gene information

GTF (Gene Transfer Format, GTF2.2)

- Extension to GFF2, backwards compatible
- First eight GTF fields are the same as GFF
- *feature* field is the same as GFF, has controlled vocabulary:
  - *gene, transcript, exon, CDS, 5UTR, 3UTR, inter, inter_CNS,* and *intron_CNS*, etc
- *group* field expanded into a list of *attributes* (i.e. key/value pairs)

The attribute list must begin with the one mandatory attribute:

gene_id *value* - A globally unique identifier for the genomic source of the sequence

GTF

```
##description: evidence-based annotation of the human genome (GRCh38), version 27 (Ensembl 90)
##provider: GENCODE
##contact: gencode-help@sanger.ac.uk
##format: gtf
##date: 2017-08-01
chr1    HAVANA  gene    923928  944581  .       +       .       gene_id "ENSG00000187634.11"; gene_type
"protein_coding"; gene_name "SAMD11"; level 2; havana_gene "OTTHUMG00000040719.10";
```

- seqname: chr1
- source: HAVANA
- feature: gene
- start: 923928
- end: 944581
- score: . (no score)
- strand: +
- frame: . (not coding feature)
- attributes:
  - gene_id: ENSG00000187634.11
  - gene_type: protein_coding
  - gene_name: SAMD11
  - level: 2
  - havana_gene: OTTHUMG00000040719.10

GFF/GTF encodes relationships

- Features are hierarchical, e.g.:
  - A gene has 1 or more transcripts
  - A transcript has 1 or more exons
  - An exon is a coding sequence (CDS)
- Relationships encoded in attributes

```
##description: evidence-based annotation of the human genome (GRCh38), version 27 (Ensembl 90)
##provider: GENCODE
##contact: gencode-help@sanger.ac.uk
##format: gtf
##date: 2017-08-01
chr1    HAVANA  gene        923928  944581  .       +       .       gene_id "ENSG00000187634.11"; gene_type
"protein_coding"; gene_name "SAMD11"; level 2; havana_gene "OTTHUMG00000040719.10";

chr1    HAVANA  transcript  923928  939291  .       +       .       gene_id "ENSG00000187634.11";
transcript_id "ENST00000420190.6"; gene_type "protein_coding"; gene_name "SAMD11"; transcript_type
"protein_coding"; transcript_name "SAMD11-203";

chr1    HAVANA  exon        923928  924948  .       +       .       gene_id "ENSG00000187634.11";
transcript_id "ENST00000420190.6"; gene_type "protein_coding"; gene_name "SAMD11"; transcript_type
"protein_coding"; transcript_name "SAMD11-203"; exon_number 1; exon_id "ENSE00001637883.2";

chr1    HAVANA  CDS         924432  924948  .       +       0       gene_id "ENSG00000187634.11";
transcript_id "ENST00000420190.6"; gene_type "protein_coding"; gene_name "SAMD11"; transcript_type
"protein_coding"; transcript_name "SAMD11-203"; exon_number 1; exon_id "ENSE00001637883.2";
```

Example

Align reads to the reference, sort and index, call SNPs, extract the SNPs on chromosome Y overlapping with all the Alu elements.

Example

Align reads to the reference, sort and index, [illegible] ct the SNPs on chromosome Y overlapping with all the Alu elements.

FASTQ format

Example

Align reads to the reference, sort and index [illegible]ract the SNPs on chromosome Y overlapping with all the Alu elements.

FASTA format

Example

Align reads to the reference, sort and index, call SNPs, extract the SNPs on c overlapping with all the Alu elements.

save in BAM format

Exaple

Align reads to the reference, sort and index, call SNPs, extract the SNPs on c

overlapping with all the Alu elements.

Example

Align reads to the reference, sort and index, call SNPs, extract the SNPs on chromosome Y overlapping with all the Alu elements.

Example

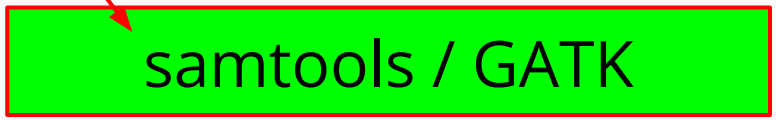
Align reads to the reference, sort and index, call SNPs, extract the SNPs on chromosome Y overlapping with all the Alu elements.

Example

Align reads to the reference, sort and index, call SNPs, extract the SNPs o samtools Y overlapping with all the Alu elements.

Example

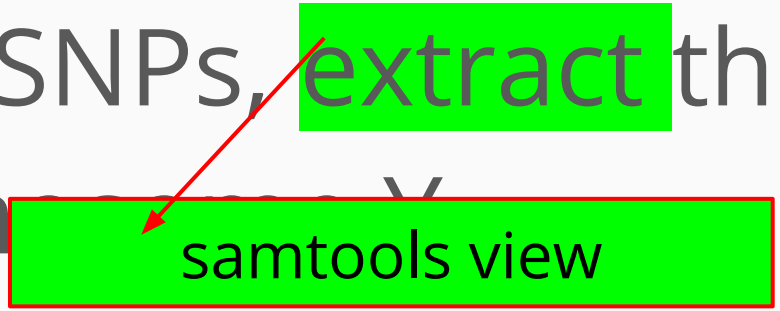
Align reads to the reference, sort and index, call SNPs, extract the SNPs on chromo



overlapping with all the Alu elements.

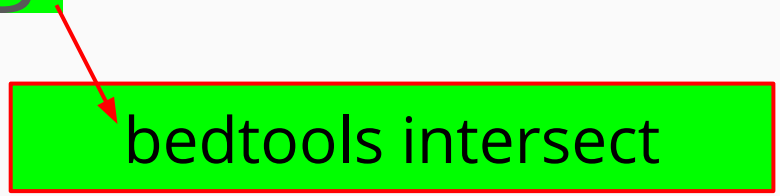
Example

Align reads to the reference, sort and index, call SNPs, extract the SNPs on chrom[illegible] X overlapping with all the Alu elements.



Example

Align reads to the reference, sort and index, call SNPs, extract the SNPs on chromosome Y overlapping with all the Alu elements.

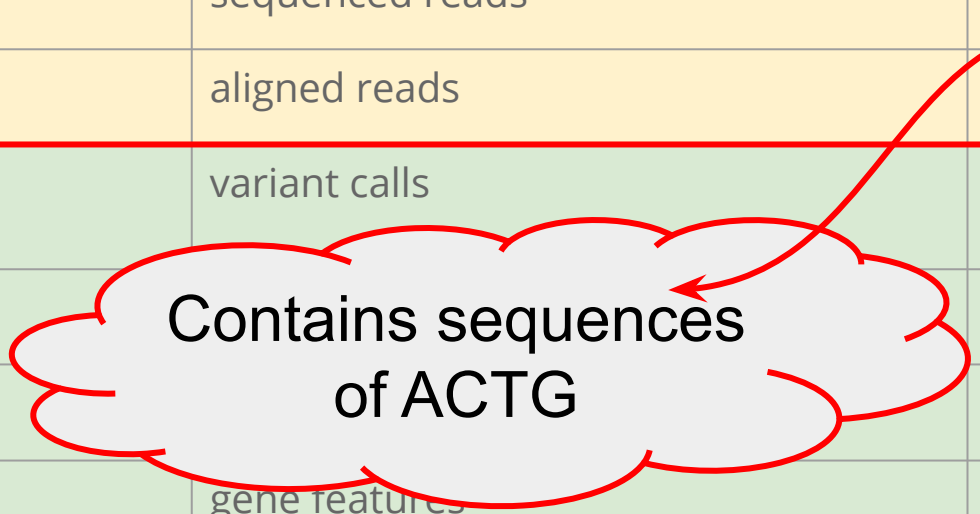


Summary

format	data	tool(s)
FASTA	sequence of nucleotides	samtools faidx
FASTQ	sequenced reads	-
SAM/BAM/CRAM	aligned reads	samtools
VCF	variant calls	vcftools bedtools
BED / BED-PE	genomic regions	bedtools
GFF	general features	-
GTF	gene features	-

Summary

format	data	tool(s)
FASTA	sequence of nucleotides	samtools faidx
FASTQ	sequenced reads	-
SAM/BAM/CRAM	aligned reads	samtools
VCF	variant calls	vcftools bedtools
BED / BED-PE		bedtools
GFF		-
GTF	gene features	-



Summary

format	data	d(s)
FASTA	s	faidx
FASTQ	sequen	
SAM/BAM/CRAM	aligned reads	samtools
VCF	variant calls	vcftools bedtools
BED / BED-PE	genomic regions	bedtools
GFF	general features	-
GTF	gene features	-

Contains positions: chromosome, start and end

Summary - data formats

- Always use standard data formats
- view/edit the data with standard tools.
- Cite the tools you used, along with the *version*, to make your work reusable.
- If you find a bug:
  - Make sure it's a bug!
  - If it's actually a bug, politely report it to the author