BED Format

Browser Extensible Data *.bed*

- https://en.wikipedia.org/wiki/BED_(file_format)

- File containing Genomic Regions
  - 0-base coordinates (more on this later!)

- Tab *\t* delimited

- BEDtools command line tool
  - *https://bedtools.readthedocs.io/en/latest/*
  - *Fast AF*

```
1       0       87112
1       87113   267707
1       267719  752672
1       752747  756780
1       759039  802515
1       802572  807887
1       808259  834081
1       891301  895937
```

BED requires *chrom start end*

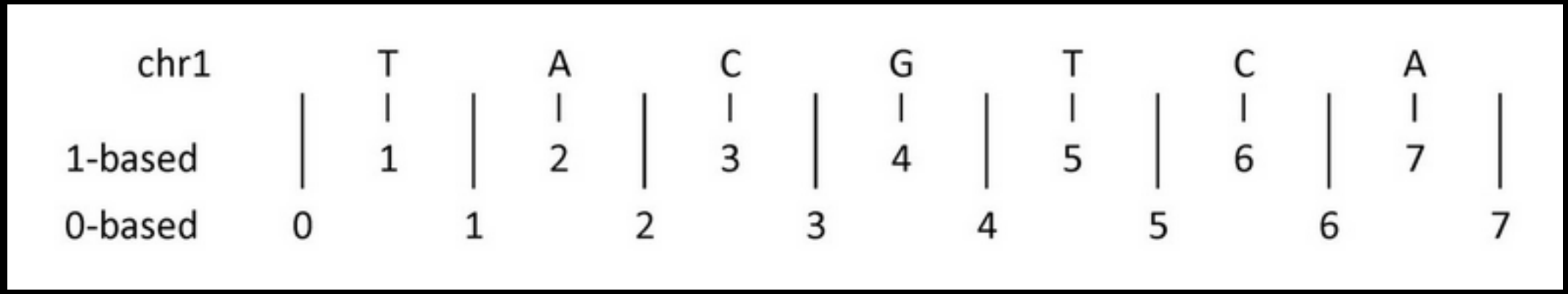
- 2 flavors of BED file
  - UCSC BED for Genome Browser
  - Everything else
- *Append information to a genomic position!*
  - *chr1 0 100 GeneX ScoreY …*

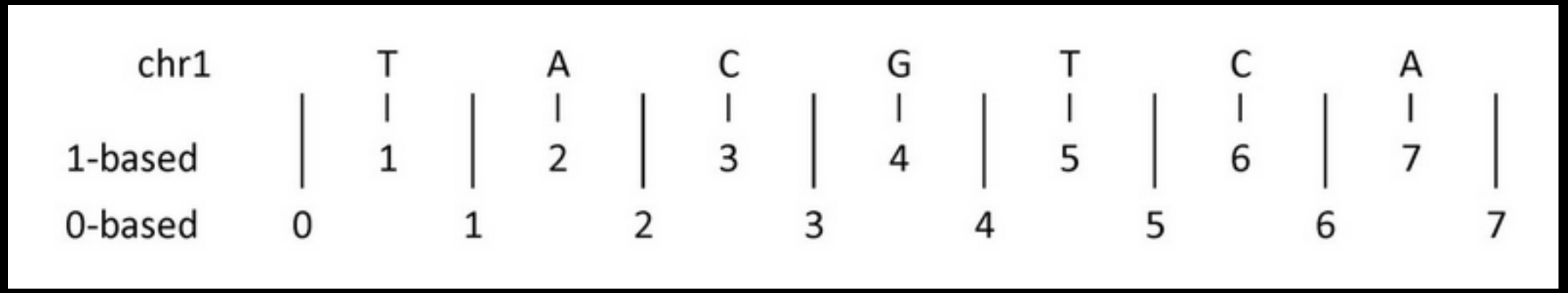
Columns of BED files (in red are the obligatory columns)

Column number	Title	UCSC BED assumes col4 will always contain a name entry
1	**chrom**	Chromosome (e.g. chr3, chrY, chr2_random) or *scaffold* (e.g. scaffold10671) name
2	**chromStart**	Start coordinate on the chromosome or scaffold for the sequence considered (the first base on the chromosome is numbered 0)
3	**chromEnd**	End coordinate on the chromosome or scaffold for the sequence considered. This position is non-inclusive, unlike chromStart.
4	**name**	Name of the line in the BED file
5	**score**	Score between 0 and 1000
6	**strand**	DNA strand orientation (positive ["+"] or negative ["-"])
7	**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)
8	**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)
9	**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
10	**blockCount**	Number of blocks (e.g. exons) on the line of the BED file
11	**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")
12	**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")

0-base: chrom. begins with 0 not 1

- Convert 0-base to 1-base
  - `start1 = start0+1;`
  - `end1 = end0;`





	1-based	0-based
Indicate a single nucleotide	chr1:4-4 G	chr1:3-4 G
Indicate a range of nucleotides	chr1:2-4 ACG	chr1:1-4 ACG
Indicate a single nucleotide variant	chr1:5-5 T/A	chr1:4-5 T/A

BED 0-Base.

VCF 1-Base.

Bad rule of thumb:
if you can read it, it's 1-base

File	Coordinate System
BED	0-based
GTF	1-based
GFF	1-based
SAM	1-based
BAM	0-based
VCF	1-based
BCF	0-based
Wiggle	1-based
GenomicRanges	1-based
BLAST	1-based
GenBank/EMBL Feature Table	1-based

Convert VCF -> BED with AWK

- Print SNPs into a BED file

```
grep -v "#" 1000G_omni2.5.b37.sites.vcf | \
awk '{ print $1"\t"$2-1"\t"$2"\t"$3 }'

>1kgp_omni2.5.hg19.bed
```

- Why?
  - Intersect to regions of interest (blacklisted regions, genes, etc.)
  - BEDtools can take VCF as input BUT
    - Only considers the POS not the END if SV

BEDtools 💚 sorted BED files

- `sort –k1,1 –k2,2n in.bed >in.sorted.bed`
- Many commands require sorted files
- `intersectBed –sorted …` much faster
- `bgzip in.bed; tabix –p bed in.bed.gz`
  - Compress and index a BED file
- Naming convention
  - dataset.reference.bed
  - *`gnomad_sv.hg38.bed`*

Split BED by chromosome with awk

```
zcat in.bed.gz | awk '{ print $0 >>"in."$1".bed" }'
```

- Note: >> is the append operator
  - Delete files if you need to run again!

Check the Docs!

- Search online ”bedtools <subcommand>”
  - https://bedtools.readthedocs.io/en/latest/content/tools/intersect.html
- Great visual examples

-wao Write *amounts* of overlap for all features.

The `-wao` option extends upon the `-wo` option in that, unlike `-wo`, it reports an overlap of 0 for features in A that do not have an intersection in B.

```
$ cat A.bed
chr1    10    20
chr1    30    40

$ cat B.bed
chr1    15  20
chr1    18  25

$ bedtools intersect -a A.bed -b B.bed -wao
chr1    10    20    chr1    15  20  5
chr1    10    20    chr1    18  25  2
chr1    30    40    .       -1  -1  0
```

Filter variants in Bad regions

```
intersectBed –a in.bed \
    –b mask.bed \
    –wa –v \
    >in.filtered.bed
```

- -wa : write original entry in A
- -v : report the opposite (variants that do NOT overlap B)

GTF -> BED: making a BED file of exons

```
zcat gencode.gtf.gz | \
grep “protein_coding” | \
awk ‘{ if($3 == “exon”) { print $1”\t”$4-1”\t”$5”\t”$0 }’ | \
cut –f 1-3,12 >gencode_exons.bed
```

****** WARNING: File gencode_exons.hg19.bed has inconsistent naming convention for record:*

If this happens likely one file has “chr1” and the other has “1”

Perl one-liners to fix “chr” problems

```
perl –pi –e ‘s/^chr//g’ in.bed
```

- *Remove “chr” from the beginning of each line*

```
perl –pi –e ‘s/^/chr/g’ in.bed
```

- *Add “chr” to the beginning of each line*

Report overlap to unique genes

```
intersectBed –a in.bed –b exons.bed –wa –wb >in.exons.bed
```

- Write both the A and B entry if they overlap
- Then write a script to parse out the gene name (if needed)
  - Dictionary of dictionaries example