BIO392 genomic variation file formats and tools, 2022

izaskun.mallona@mls.uzh.ch

- Learning objectives
- Schedule
- Strategy
- Exam questions

1. Understand the standard NGS -> QC -> mapping -> QC -> variant calling flow
2. Recognize a handful of the standard file formats (FASTQ, SAM, GTF, BED, VCF)
3. Be aware of the caveats of such formats (e.g. coordinate systems, 0/1 counting)
4. Manipulate standard file formats using UNIX tools
5. Run and interpret results: genomic analysis of human genetic variation due to mobile elements insertions (separate slide deck)

Schedule

1. **Thursday 22nd Sept**
   - Lecture on Unix, data generation, data formats, and data handling
     - Introduction to the commandline
     - Sequencing, alignment, coordinate files, variant files
   - Hands-on session on UNIX and file handling
     - Working with the terminal
     - Data streams
     - Basics of awk, sed, cut
   - Hands-on session on data formats
     - FASTA, FASTQ, SAM, GTF, BED, VCF
2. **Friday 23rd Sept**
   - Hands-on session on data formats (cont)
   - Small project regarding human genetic variation and interpretation
     - Retrotransposition events in 1000 genomes data

- Included within the lectures slides

- Slides
- Exercises https://github.com/compbiozurich/UZH-BIO392/blob/imallona/course-material/2022/imallona/exercises.md
- Papers/resources, most of which are 'extra' reads (linked within the slides and/or exercises)
- Literature/background for running the project on mobile elements insertions
    - https://www.nature.com/articles/nmeth.1906 chromHMM paper
    - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2832824/, bedtools paper
    - https://www.nature.com/articles/nature15393, 1000 genomes project

- Background
  - Retrotransposition events introduce genetic variation in humans
- Biological question
  - How often do retrotransposons jump inside a genome?
  - Do we detect any insertion patterns? Do transposons jump inside genes? in promoters? in heterochromatin?
  - (and any other questions you decide)
- Procedure
  - Integration of 1000 Genomes Project and functional genomics data (e.g. chromatin segmentation data)
- Running the analysis
  - Either locally (Mac), renku, or in your computer (easier with Linux)

- All analysis are run on open data, with free software running on a terminal = console
- To be run on a Linux, a Mac, or (a docker image in) renku http://renkulab.io/

- Commands/options are in `typewriter font`
- URLs are highlighted in blue

Exercise: Web browsing the genome

- Launch the UCSC Genome Browser
- Specify Human Assembly hg19
- Click go
- Notice the many data layers, including variation data

By default, the Genome Browser will render a genomic window with many data layers on it. How are these data encoded?

- Click UCSC Genes from the Genes and Gene Predictions section under the main genomic window.
- Click View table schema opens knownGene table schema

Schema for UCSC Genes - UCSC Genes (RefSeq, GenBank, CCDS, Rfam, tRNAs & Comparative Genomics)

Database: hg19   **Primary Table:** knownGene   **Row Count:** 82,960   **Data last updated:** 2013-06-14
Format description: Transcript from default gene set in UCSC browser

field	example	SQL type	info	description
name	uc001aaa.3	varchar(255)	values	Name of gene
chrom	chr1	varchar(255)	values	Reference sequence chromosome or scaffold
strand	+	char(1)	values	+ or - for strand
txStart	11873	int(10) unsigned	range	Transcription start position (or end position for minus strand item)
txEnd	14409	int(10) unsigned	range	Transcription end position (or start position for minus strand item)
cdsStart	11873	int(10) unsigned	range	Coding region start (or end position if for minus strand item)
cdsEnd	11873	int(10) unsigned	range	Coding region end (or start position if for minus strand item)
exonCount	3	int(10) unsigned	range	Number of exons
exonStarts	11873,12612,13220,	longblob		Exon start positions (or end positions for minus strand item)
exonEnds	12227,12721,14409,	longblob		Exon end positions (or start positions for minus strand item)
proteinID		varchar(40)	values	UniProt display ID, UniProt accession, or RefSeq protein ID
alignID	uc001aaa.3	varchar(255)	values	Unique identifier (GENCODE transcript ID for GENCODE Basic)

So they are database entries with **chrom**, **start** and **end** features. This is the most standard data representation in genomics: data refering to genomic coordinates. Why?

- How big is the human genome, in both nucleotides and file sizes?

- How big is the human genome, in both nucleotides and file sizes?
  - 3 billion nt
  - 0.8 GB (the primary assembly), e.g. ensembl data repository
- How would you store information, including sequence, related to human genomes?
  - Genes
  - Variants
  - Promotes
  - Physical properties, e.g. local G+C content
  - (etc)

- How big is the human genome, in both nucleotides and file sizes?
  - 3 billion nt
  - 0.8 GB (the primary assembly), e.g. ensembl data repository
- How would you store information, including sequence, related to human genomes?
  - Genes
  - Variants
  - Promotes
  - Physical properties, e.g. local G+C content
  - (etc)
- Using file formats based in coordinates (chromosome, start, end), and not sequences

- Reference genomes
- Fasta and FastQ (Unaligned sequences)
- SAM/BAM (Alignments)
- BED (Genomic ranges)
- GFF/GTF (Gene annotation)
- BEDgraphs (Genomic ranges)
- Wiggle files, BEDgraphs and BigWigs (Genomic scores).
- Indexed BEDgraphs/Wiggles
- VCFs (variants)

- Reference genomes describe the 'consensus' DNA sequence
- (The human genome from whom? What does consensus mean?)
- Human variation aside, multiple assemblies have been released (i.e. hg18, hg19, hg38...)
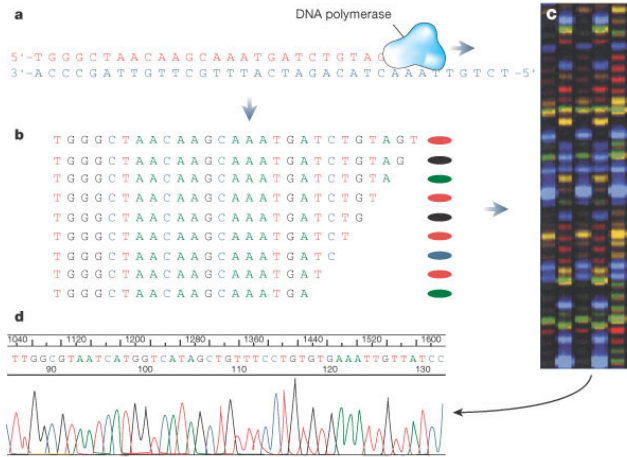
Hierarchical shotgun sequencing
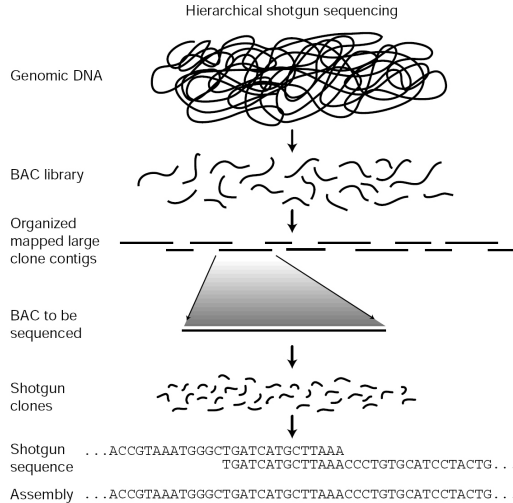
GRCh stands for 'Genome Reference Consortium'

- Human GRCh37 (hg19)
- Human GRCh38
- Mouse mm10
- Mouse GRCm38
- Zebrafish, chicken and others: https://www.ncbi.nlm.nih.gov/grcThe Genome Reference consortium

- Retrieve the sequence of the human mitochondrial genome
- Where (databases)? how? which format?
- Does the human mitochondrial genome exist?
- If succeeded to download one, is it mouse clicking automatable/reproducible?

Automation

- Using a Web browser to retrieve genomic sequences is not efficient nor reproducible: programmatic alternatives exist
- Need of standardizing data analysis using reproducible workflows
  - Scripts for data retrieval (in bioinformatics often R or python)
  - Data storage: standards (fasta, fastq, sam, vcf...)
  - Keeping track of data analysis steps and avoiding manual editing

Reproducibility

- What do we mean by data science reproducibility?
- In data science: avoid manual steps of data analysis using scripts plus version control systems
- Spreadsheet editors used with sequences of mouse clicks are not reproducible

- Paper: https://www.nature.com/news/1-500-scientists-lift-the-lid-on-reproducibility-1.19970

- In data analysis: keeping track of all steps (using scripts)
- Using data standards
- What if we don't know how to program?
- Still, switching to command-line tools and keeping track of the commands used
- Using control version systems

- Simple command line interface
- Present in MacOS and GNU/Linux computers
- Interprets the Unix shell language (commonly bash)

- Efficient
- Scalable
- Portable
- Open

Unix philosophy

According to Peter H. Salus in A Quarter-Century of Unix (1994):

- Write programs that do one thing and do it well
- Write programs to work together
- Write programs to handle text streams, because that is a universal interface

- We interpret DNA, proteins as text; Unix is for text streams
- Data are big (millions of lines of text, easily a couple of GB); spreadsheet software (Excel) cannot handle them
- We need to keep track of our analysis for the sake of reproducibility: bash scripts
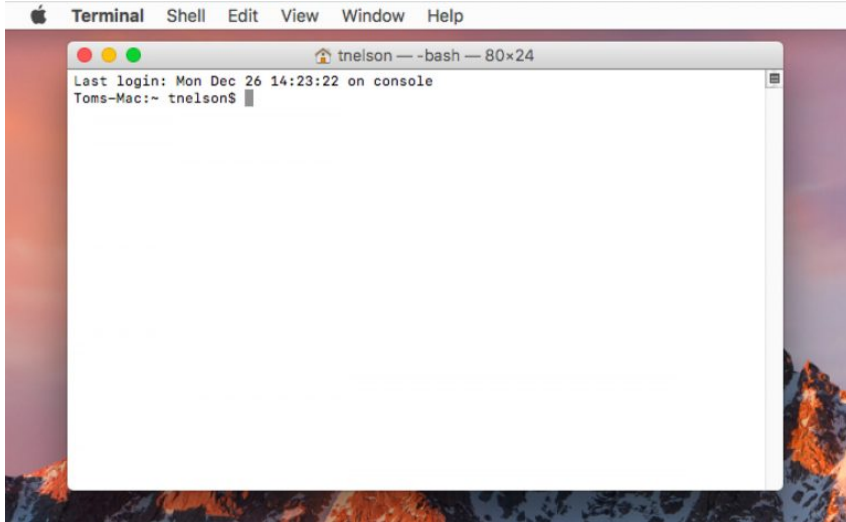
- The Unix shell allows to save the sequential commands in a reproducible manner
- Activity: run the tutorial by the Swiss Institute of Bioinformatics
  https://edu.sib.swiss/pluginfile.php/2878/mod_resource/content/4/couselab-html/content.html
- (Normally requires a full day)

A quick reminder on computer files

- Files are data representations stored in computers as arrays of bytes.
- File type is defined by its bytes and not by the filename extension.
- Files contain metadata.
- Importantly, plain text files are composed by bytes mapped directly to ASCII characters.
- Text editors (notepad, gedit, vim...) allow editing plain text files.
- (text files can be read without proprietary software)

- Create new file
- Go to `Format` and select `Make Plain Text`
- For saving, go to `File`, `Save As` and `Plain Text Encoding` setting: `Unicode (UTF-8)`.
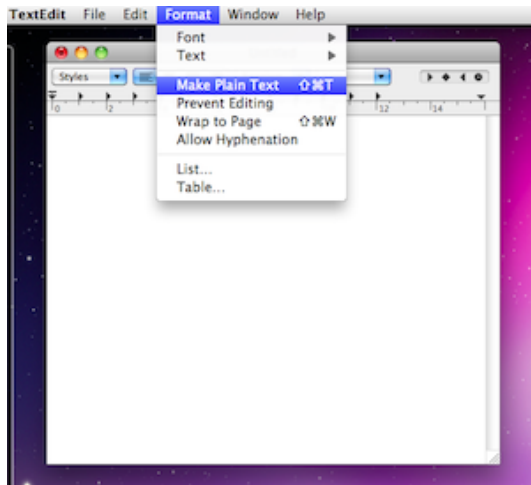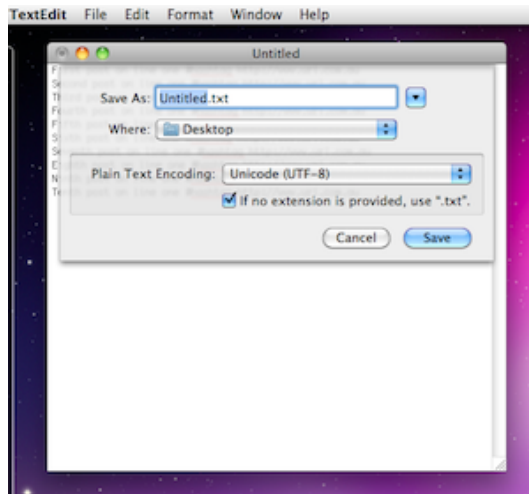
Turning off autospell check

- UNIX commands are case sensitive
- By default, TextEdit capitalizes/spell checks contents
- Exercise: disable this feature
- http:
  //osxdaily.com/2014/05/06/turn-off-autocorrect-pages-textedit-mac/

Organizing a shell script

- Write on top the shebang
- Write the date and what's the script about, your name and date
- Tip: comment lines start with #
- Tip: multiline chunks are split by a \
- Introduce the commands (one line each)

Scripting

- Save the file (script) somewhere with a back-up system
- You could run the script to run the commands in batch typing bash `name_of_the_script.sh`

- UNIX solves the reproducibility, scalability and openness for data (text) streams, but extra software might be needed
- The importance of software versioning for reproducibility: keeping track of the software installed
- Using open source software (no blackboxes!)
- Installs can be run command-line, so specific versions can be stored and included into the analysis script

- Use the arrow left/right keys to move back/forth within the same line
- Use the arrow up/down keys to retrieve previous commands
- Use the 'tab' key to autocomplete
- Lines starting with # are comments - not evaluated
- Text can be written after the prompt (dollar sign **$**)

```
man man # prints the manual for the manual command
man sed # commonly used editing program

# exit pressing 'q'
```

```
$ pwd
# prints my path, e.g. /home/guest
```

```
$ cd .. # goes to parent
$ cd /tmp # goes to a folder named '/tmp'
```

```
$ ls # list files in a given path
$ ls -l # lists files and their attributes
```

```
$ mkdir newdir # creates the directory newdir
```

```
# removes the directory newdir (created before)
$ rmdir newdir
```

```
$ cp source-file destination-file

# let's create a file using echo
#   and redirecting the echo to a file
$ echo 'lorem ipsum' > test_file

# cp test_file test_file_another
$ ls
```

```
$ less test_file_another

# exit pressing q
```

```
$ head test_file_another
```

```
cat - printing files onto the screen
mv - moving and renaming files
rm - removing files and directories
chmod - changing access permissions
grep - searching for strings in files
wc - counting words
sed - stream editing files
awk - a full language to process texts
```

- Commands not only send results to the screen: can be redirected
- Commands to redirect streams
  - $a > b$ sends the std output of 'a' to 'b'
  - $a|b$ pipe, sends the std output of 'a' to 'b' (with buffering)
  - $a < b$ sends the std output of 'b' to 'a'

Standard output, input, pipes, redirects

```
# editing the standard ouput using pipes
echo "hello there"
echo "hello there" | sed "s/hello/hi/"
echo "hello there" | sed "s/hello/hi/" | sed "s/there/world/"


# redirecting the standard output to a file
echo hello > newfile.txt
ls -l newfile.txt
cat newfile.txt


# piping commands and redirecting to a file
echo hello | sed 's/hello/hElLo/' > newfile2.txt
ls -l newfile2.txt
cat newfile2.txt
```

- Some tools we use are not standard Unix programs, but domain-specific software
- BEDtools is one of the most used tools for handling coordinate-based file formats
- If interested, read paper
  https://academic.oup.com/bioinformatics/article/26/6/841/244688

- We will download the open source code of bedtools and then (try to) generate the executables
- That is **compiling** the source, and generates binaries fit to our hardware and OS
- Also, makes the same code runnable in many platforms
- That's transparent for us: the developer wrote a Makefile that automates this
- Extra reading:
  https://www.wired.com/2010/02/Compile_Software_From_Source_Code/

(The code is available at the exercises file)

```
cd # to your home directory or wherever you decide
cd soft # the folder was created before (for kent utils)

curl -L https://github.com/arq5x/bedtools2/releases/download/v2.25.0/bedtools-2.2!
  > bedtools-2.25.0.tar.gz

tar zxvf bedtools-2.25.0.tar.gz
cd bedtools2
make
alias bedtools='./bin/bedtools'
```

- Download some data/install bedtools (https://github.com/compbiozurich/UZH-BIO392/blob/imallona/course-material/2022/imallona/exercises.md, section **Set up**)
- Run exercises 1-4 (same url above)
- Run the introductory course on Unix at https://edu.sib.swiss/pluginfile.php/2878/mod_resource/content/4/couselab-html/content.html

- Reference genomes
- Fasta and FastQ (Unaligned sequences)
- SAM/BAM (Alignments)
- BED (Genomic ranges)
- GFF/GTF (Gene annotation)
- Wiggle files, BEDgraphs (Genomic scores).
- VCFs (variants)

- Reference genomes describe the 'consensus' DNA sequence
- (The genome of who? who was chosen to be the 'reference?)
- Human variation aside, multiple assemblies have been released due to technical challenges/errors
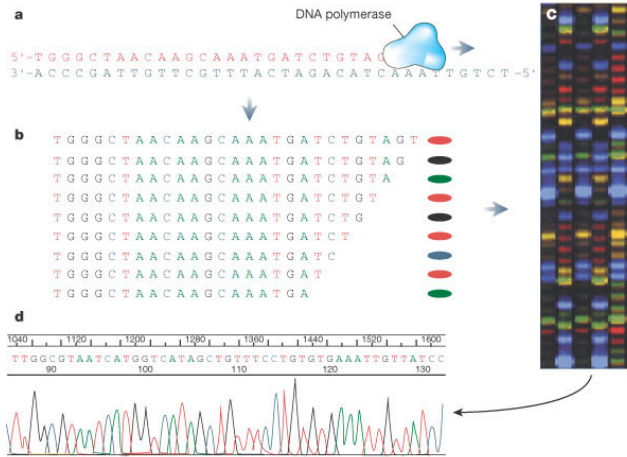
Hierarchical shotgun sequencing
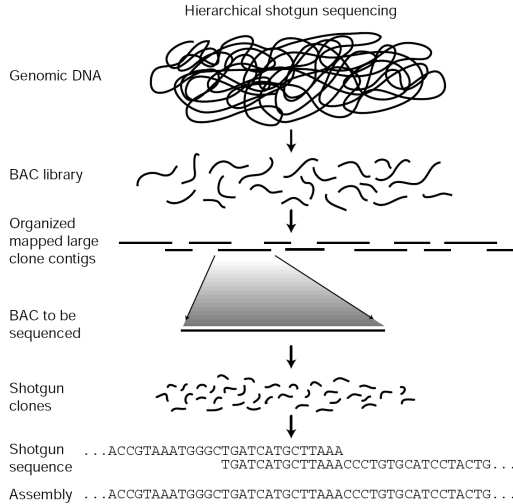
GRCh stands for 'Genome Reference Consortium'

- Human GRCh37 (hg19)
- Human GRCh38 (hg38)
- Mouse mm10
- Mouse GRCm38 (mm11)
- Zebrafish, chicken and others: The Genome Reference consortium

- A reference genome is a collection of contigs/scaffolds
- A contig is a stretch of DNA sequence encoded as A,G,C,T,N.
- Typically comes in FASTA format.
- ">" line contains the scaffold name
- Following lines contain the sequence (single line, 80 nt-column sized...)

```
>NC_009902.1 Babesia bovis T2Bo mitochondrion, complete genome
TTTAAAAAAGTGTTAAAAACTTTATACATTAAAAAATTTAAACAAGTGATCATGTATAAAGTACACTTGT
TACTGTGTAAATATCAAAAACAATTTAATTTCAAAATTTTTGAAATATGTTTTTTGTGTTGTGTTATAAA
GTTTTTTTTCAAAATTATATATGTTTGCATTTGCTGGATATAGTTCGGTCTCTGCAAACCATAAAGTCAT
CGGTATATCCTACATATGGCTTTCATATTGGTTTGGAGTTATTGGATTTTATATGAGTATTTTGATAAGA
ACAGAATTGAGTATGAGTGGTTTAAAGATTATGACAATGGATACTCTTGAGATATACAATATGATGTTTT
(...)
```

- Primary assembly: the best known assembly of a haploid genome
  - Chromosome assembly
  - Unlocalized sequence (associated to a chromosome but whose order/orientation is unknown)
  - Unplaced sequence (not linked to any chromosome)
- Alternate loci: An alternate representation of a locus (usually highly polimorphic regions, such as the MHC region)
- Patches: A contig sequence that is released outside of the full assembly release
  - Fix: error correction
  - Novel: new sequences that will be included into the next full assemblty release

- Activity: notice the assembly patches, i.e.
  https://www.ncbi.nlm.nih.gov/grc/human
- Activity: browse an individual patch
  http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.27_GRCh38.p12/README_patch_release.txt

- This we can do because the genome consortia and the science community release free data, software/toolsets
- We benefit from the Unix-like operating systems
- But the access to the same data and tools is not enough: the need for data standards

- **Fasta and FastQ**
- SAM/BAM (Alignments)
- BED (Genomic ranges)
- GFF/GTF (Gene annotation)
- BEDgraphs (Genomic ranges)
- Wiggle files, BEDgraphs and BigWigs (Genomic scores).
- Indexed BEDgraphs/Wiggles
- VCFs (variants)

- Sequencing very short reads (50 to 150 nucleotides) is common practice
- We get hundreds of millions of short reads for each experiment
- Instead of assembling them, we map them into a reference genome
- Sequencers provide sequence and error rates assessment: `fasta` format is not suitable, but `fastq` is

FASTQ: Short read sequencing

- Next step to FASTA: including quality data
- Standard de facto for short read, high-throughput sequencing instruments such (i.e. Illumina)

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=36
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACC
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9IC
```

- Sequence quality is represented using Phred scores
- The sequencing quality score of a given base Q is defined by as
- $Q = -10 \ \log_{10} P$

Phred quality scores are logarithmically linked to error probabilities

Phred Quality Score	Probability of incorrect base call	Base call accuracy
10	1 in 10	90%
20	1 in 100	99%
30	1 in 1000	99.9%
40	1 in 10,000	99.99%
50	1 in 100,000	99.999%
60	1 in 1,000,000	99.9999%

phred scores (old school Sanger electrophoretogram)

```
SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS...............................................
..............................XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX...............
.................................IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII.....
.................................JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ....
LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL...............................
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
|                                |    |       |                                 |                   |
33                               59   64      73                                104                 126
0.........................26...31.......40
                          -5....0........9...............................40
                                0........9...............................40
                                 3.....9.............................41
0.2......................26...31........41
```
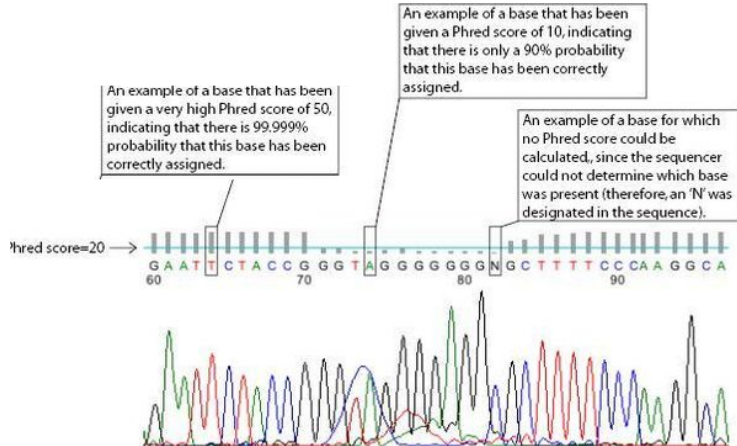
S - Sanger   Phred+33, raw reads typically (0, 40)
X - Solexa   Solexa+64, raw reads typically (-5, 40)
I - Illumina 1.3+ Phred+64, raw reads typically (0, 40)
J - Illumina 1.5+ Phred+64, raw reads typically (3, 41)
    with 0=unused, 1=unused, 2=Read Segment Quality Control Indicator (bold)
    (Note: See discussion above).
L - Illumina 1.8+ Phred+33, raw reads typically (0, 41)

- FASTQs stands for FASTA with Qualities
- Plain text files with chunks of four lines:
  - @ identifier line
  - Sequence
  - "+" (sometimes the sequence name, again)
  - Quality scores (different encodings exist)

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=36
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACC
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9IC
```

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=36
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACC
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9IC
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:346 length=36
GGGAGAAGGCCGCAGCCGAAGGCGACAAAACCCACC
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9I9999
```

- It's always on the second line of each stanza
- How could we describe this mathematically?

- It's always on the second line of each stanza
- How could we describe this mathematically?
- Use the modulo operator on the line number: extract lines, 2 and 6
  - $2 \, mod \, 4 = 2$
  - $6 \, mod \, 4 = 2$
- Tip: in awk, line numbers can be printed with the NR variable: `awk 'print NR,$0' filename`

- Activity: FASTQ/A exercises (exercises 5 to 14)

```
## retrieving an example fastq file
curl  https://molb7621.github.io/workshop/_downloads/SP1.fq \
 > file.fastq

## transforming into fasta
awk 'NR % 4 == 1 {print ">"$1};
     NR % 4 == 2 {print}' file.fasta
```

So in fastq each data chunk is stored in four different lines. We'll need to be able to extract the first, second, third of fourth line for each block of four lines. Using awk,

```
awk 'END{print NR/4}' file.fastq
```

- NR gives the number of records (line numbers)
- FASTQ are chunks of 4 lines for each sequence
- NR/4 at the END of the file indicates the number of sequences

- These tasks can be done in many programming languages
- In bioinformatics, we normally use AWK
- https://en.wikipedia.org/wiki/AWK
- http://www.grymoire.com/Unix/Awk.html

```
## retrieving an example fasta file
curl  https://molb7621.github.io/workshop/_downloads/SP1.fq \
 > file.fastq

## counting number of reads
awk 'END{print NR/4}' file.fastq

## transforming into fasta
awk 'NR%4==1{a=substr($0,2);} NR%4==2 {print ">"a"\n"$0}' \
  file.fastq
```

```
awk 'NR%4==1{a=substr($0,2);} NR%4==2 {print ">"a"\n"$0}' \
  file.fastq
```

- % is a modulo operator
- NR%4==1 will retrieve the first line of a fastq chunk (header)
- NR%4==2 will retrieve the second line (the sequence)
- the id line will be preprended with the > and reduced to a substring (chopped)
- This will be applied to all lines!

- Discussion: how to get rid of the sequences and to have a smaller data representation?
- Trying to transform sequence to reference genome coordinates (= aligning to the genome/mapping)
- i.e. transforming `ACGCACGCACGCACGCCCC` to human genome hg19 'chr10:10010-10030'

- SAM - Sequence Alignment Map.
- The standard stores where the reads (i.e. the ones we had as FASTQs) map in the reference genome

- Sequence alignment: arrange a set of sequences to identify regions of similarity/identity
- Mapping short reads against a reference genome: aligning large amounts short reads to a reference genome

Local alignments vs global alignment

- Alignment/mapping consist on finding homologies of query sequences to the reference
  - Local: aligns a substring of the query to a substring of the reference
  - Global: contains all letters from both query and reference

(reference: top, query: bottom)

```
        1   2   3   4   5   6   7   8   9  10  11
        C   G   T   C   C   G   A   A   G   T   G
                |   .   |   |   |   |
        *   *   T   A   C   G   A   A   *   *   *
```

(a) Global alignment

```
    3   4   5   6   7   8              1   2   3   4   5   6   7   8
    T   C   C   G   A   A              C   G   T   C   C   G   A   A
    |   .   |   |   |   |                      |   .   |   |   |   |
    T   A   C   G   A   A              *   *   T   A   C   G   A   A
```

(b) Local alignment   (c) Semi-global alignment

Alachiotis et al, 2013

- Chromosome
- Locus (coordinate)
- CIGAR string, i.e.
    - 30M1D2M - 30 bases continuously match, 1 deletion from reference, 2 base match
- Some flags
  (https://broadinstitute.github.io/picard/explain-flags.html)

Pavlopoulos et al 2013

Pavlopoulos et al 2013

Pavlopoulos et al 2013

- Extra activity: read the SAM format specification
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2723002/

- Exercise number 15

- SAM (Sequence Alignment Map) files are human-readable text files
- BAM (Binary Alignment Map) files are their binary (and compressed) equivalent

- Next step after getting the SAM files: extract features from reads, by coordinates and get rid of sequences
  - RNA-seq: expression levels, per gene
  - ChiP-seq: binding, per bin
  - Variant calling: variation detection
- Why? they are smaller and easier to handle
- What do we mean by feature quantification?
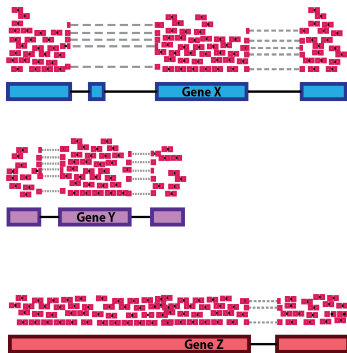
Sample A Reads   Sample B Reads
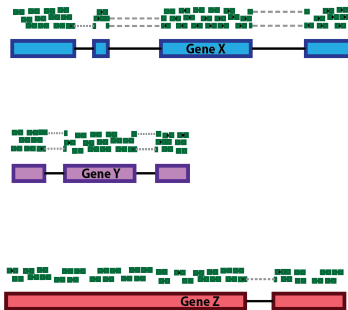
- Discussion: how to handle expression data, i.e. transcripts without introns etc? how do we count them?
- Food for thought: https://bedtools.readthedocs.io/en/latest/content/tools/genomecov.html

- BED (Browser Extensible Data) files come in different flavours
- BED3: 3 tab separated columns, chromosome (scaffold), start, end
- BED6: BED3 plus name, score, strand

Just chromosome, start, end

chr22 1000 5000
chr22 2000 6000

BED3 and name, score, strand

```
chr22 1000 5000 cloneA 960 +
chr22 2000 6000 cloneB 900 -
```

BED6 and thickStart, thickEnd, itemRgb, blockCount, blockSizes, blockStarts

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

```
chr22 1000 5000 cloneA 960 + 1000 5000 0 2 567,488, 0,3512
chr22 2000 6000 cloneB 900 - 2000 6000 0 2 433,399, 0,3601
```

- Even though BED files are standard how to count nucleotides is not
- 0-start vs. 1-start : Does counting start at 0 or 1?
- Fully-open, fully-closed, or hybrid-intervals (e.g., half-open)?
- Extra activity: browse http://genome.ucsc.edu/blog/the-ucsc-genome-browser-coordinate-counting-systems/

Given an interval ----a------b---
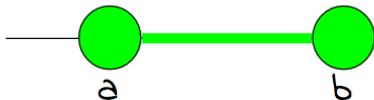
FULLY CLOSED (HINT: THINK EN-CLOSED!)
Closed-start, closed-end
Both endpoints "a" and "b" are included
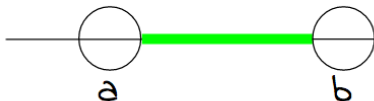
FULLY OPEN
open-start, open-end
Both endpoints "a" and "b" are excluded
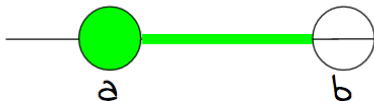
HALF-OPEN
Closed-start, open-end
endpoint "a" is included, "b" is excluded

0-Start, half-open
closed-start (included)
open-end (excluded)

Range= 0-5

Size = Stop - Start
5 = 5 - 0

Start (0) **in**cluded in set
Stop (5) **ex**cluded in set

COORDINATES STORED IN THE UCSC Genome Browser Tables
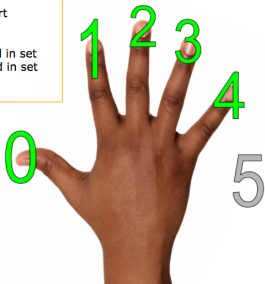
1-Start, Fully-Closed
closed-start (included)
closed-end (included)

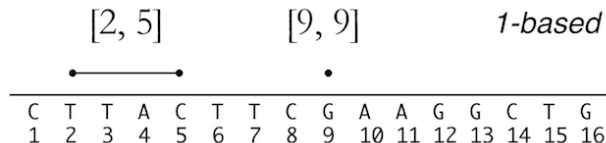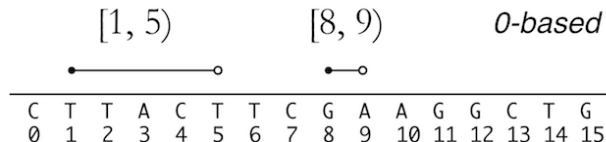Range= 1-5

Size = Stop - Start (+ 1)
5= 5 - 1 (+1)

Start (1) included in set
Stop (5) included in set

COORDINATES POSITIONED WITHIN THE UCSC Genome Browser Web Interface
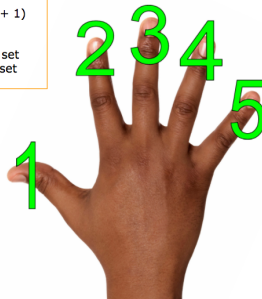
Unlike the coordinate system used by other standards (GFF), BED format is zero-based half-open

V Buffalo (2015), Bioinformatics data skills

```
# generate a bed3 record
echo -e 'chr1\t0\t1000' > test.bed

# check the original file
cat test.bed

# offset all starts +1
awk '{OFS=FS="\t"; print $1,$2+1,$3}' test.bed
```

- Exercises 16 to 24

- So to sum up until now generally we have a reference genome, reads that were retrieved as FASTQ files, mapped and transformed to SAM files
- (Which step was the last one with actual sequences on them?)
- So, at last, we can answer questions
- Which fraction of the human genome is covered by exons?
- Genomic locations of SNPs associated with prostate cancer?
- Are gene bodies more variable (in terms of SNPs) than intergenic regions?
- For this we need further data analysis tools (i.e. BEDtools) on BED files and other data formats (annotations and variation)

- Genomic annotations are layers to genomic coordinates specifying their nature

Henrik Lantz, BILS/SciLifeLab

Seqid	source	type	start	end	score	strand	phase	attributes
Chr1	Snap	gene	234	3657	.	+	.	ID=gene1; Name=Snap1;
Chr1	Snap	mRNA	234	3657	.	+	.	ID=gene1.m1; Parent=gene1;
Chr1	Snap	exon	234	1543	.	+	.	ID=gene1.m1.exon1; Parent=gene1.m1;
Chr1	Snap	CDS	577	1543	.	+	0	ID=gene1.m1.CDS; Parent=gene1.m1;
Chr1	Snap	exon	1822	2674	.	+	.	ID=gene1.m1.exon2; Parent=gene1.m1;
Chr1	Snap	CDS	1822	2674	.	+	2	ID=gene1.m1.CDS; Parent=gene1.m1;
		start_ codon						Alias, note, ontology_term ...
		stop_c odon						

Henrik Lantz, BILS/SciLifeLab

Seqid	source	type	start	end	score	strand	phase	attributes
Chr1	Snap	exon	234	1543	.	+	.	gene_id "gene1"; transcript_id "transcript1";
Chr1	Snap	CDS	577	1543	.	+	0	gene_id "gene1"; transcript_id "transcript1";
Chr1	Snap	exon	1822	2674	.	+	.	gene_id "gene1"; transcript_id "transcript1";
Chr1	Snap	CDS	1822	2674	.	+	2	gene_id "gene1"; transcript_id "transcript1";
		start_codon						
		stop_codon						

Henrik Lantz, BILS/SciLifeLab

Steven M. Carr

- How many transcript does a gene have?
- Genes have transcripts, exons, CDS, start codons
- Some exons belong to several transcripts
- In GTF and GFF, parent information is needed to link different records (lines)
  - i.e. linking a transcript to its gene

- Extra documentation:
  https://www.ensembl.org/info/website/upload/gff.html and
  https://github.com/The-Sequence-Ontology/Specifications/blob/
  master/gff3.md

- Run exercises 25 to 27

- Strand-less score information
- BEDgraph (another flavour of BED)
- Wiggle (bigWig, if compressed), (even more) compact data representations

- To display continuous-valued data in track format.
- Uueful for probability scores

```
chromA   chromStartA   chromEndA   dataValueA
chromB   chromStartB   chromEndB   dataValueB

chr19 49303800 49304100 0.50
chr19 49304100 49304400 0.75
chr19 49304400 49304700 1.00
```

- BED, BED6, BED12?
- Advantages: the coordinates are specified, so sparsity is allowed
- Next step in file formats: trying to cover all the genome (that is, no sparsity anymore)
- Example: does it make sense to generate a BED file with GC content? (GC content at Wikipedia)
- How can we store features with definite start and ends but for which the value is the primary purpose, but not their starts and ends?

- To display continuos-value data
- GC percent, probability scores, and transcriptome data.
- Data is not sparse! Wiggle data elements must be equally sized (step)

Let's construct a Wig file: a score per nucleotide

- Basic Wig file: specifies the chromosome (once), the nucleotide, and the score.
- Any improvements?

```
variableStep chrom=chr2
300701 12.5
300702 12.5
300703 12.5
300704 12.5
300705 12.5
```

- Improved Wig: add the span if the score applies to several nucleotides

```
variableStep chrom=chr2 span=5
300701 12.5
```

- Improved Wig: how to represent the scores of 300 nucleotides from chr3, starting from 400601, where the first 100 nt are scored 11, the next 100 nt 22, and the last 33?

```
fixedStep chrom=chr3 start=400601 step=100 span=5
11
22
33
```

- Read more on Wig files at
  https://genome.ucsc.edu/goldenpath/help/wiggle.html

- Standard file format for storing variation data
- Unambiguous, scalable and flexible
- 8 columns:
    1. CHROM
    2. POS
    3. ID
    4. REF
    5. ALT
    6. QUAL
    7. FILTER
    8. INFO

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	NA19909
11	5248232	rs334	T	A	100	PASS	AA=T|||;AC=1;AF=0.0273562;AFR_AF=0.0998;AMR_AF=0.0072;AN=2;DP=22876;EAS_AF=0;EUR_AF=0;EX_TARGET;NS=2504;SAS_AF=0;VT=SNP	GT	0|1

EMBL/EBI training

- Phred-scaled quality score for the assertion made in ALT. i.e. $Q = -10 \, \log_{10} P$ being P(call in ALT is wrong)
- Read quality
- Mapping quality
- Variant calling quality

- Activity: read https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3137218/

- Remember the unusual 0 and 1 counting when manipulating GTF, GFF, BED, Wiggle and/or when using genomic positions at the UCSC genome browser
- Resource: http://genome.ucsc.edu/blog/ the-ucsc-genome-browser-coordinate-counting-systems/

- 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

V Buffalo (2015), Bioinformatics data skills

- Complete the remaining exercises
- Brainstorm; design an alternative for a common task: retrieve the human gene DIEXF promoter

- Try to retrieve the DIEXF gene promoter in a non-programmatic way
- (What is a promoter? do we know where they are?)
- Choose a reference assembly, i.e. hg38
- Query gene symbol (i.e. DIEXF)
- Click into the gene (gencode track)
- Click into the sequence and links item
- Specify your promoter definition

Manually downloading the DIEXF promoter

Manually downloading the DIEXF promoter

```
>hg38_knownGene_uc001hhr.3 range=chr1:209827007-2098
gtttctgctgttgttaaatggggaatgctggaacagatttgtttgcgggg
actcttccaatactttcagaaaatgcgagaataggggtgagggtgggaatc
tcagacttgtgggccccatgattgatataacacacacaggcggcagaccc
taatgggtaaaagcatgtggttgcatcagttaaggttttctctcttctc
ttgctagcgtgttatctttctttctttctttcttctttttttttttttcg
agatggagtctagctttgtcgcccaggctggagtaggctggagtgcagt
ggagtgatctcggctcattgcaacctccacctcccgggttccagcgattc
tcctgcctcacctcctgagtagctgggattacaggcgcccgctaccacgc
ccggctgattttttgtacttttagtagagacggggtttcaccatgtttggc
catgctggtctcgaactcctgacctcaggtgatccgcccatctcggcctc
ccaaagtgttgagattacaggcgtgagccaccgcgcccggccgctagcgt
gttatctttctaagcatcagtttccttatctgcaacaccaggcttatta
acaagacctatctgtacactgttgtggtgatgaagtgagatgttcaggca
cccttaaatgttggttgatattttattgcagtatactgtaaagtcactg
cattcgactatctccgctactacacatttacgcagactgatttccataac
caaaacacaagcacaaagctcatgcccccgactcacgcaacccgggaagc
ccagctgcccacgttctagggctctgagaacactagtgaacgaactcccg
tgctttcaaagagctgcggtaggggggcagaaccgggaaccggatgttcta
agcctgtcgtacgagcgcgacgtaaagcggatctgctttatggcaccttg
ctttcgccgtaaagcgcagtcagcgagccacgtgcttgtgttgactgga
```

Same concept that when we did manually:

1. Download the reference genome sequence (if we didn't have that already)
2. Download a file with all the genes (transcripts) locations (not sequences, but their coordinates)
3. Then select the gene we are looking for (DIEXF)
4. Decide what a promoter is (i.e. 2 kb upstream of the gene) and update the coordinates accordingly
5. Then use a specific tool to slice the full genome to only report the DIEXF promoter

Same concept that when we did manually + some standards

1. Download the reference genome sequence (fasta)
2. Download a file with all the genes (transcripts) locations (GTF)
3. Then select the gene we are looking for (DIEXF) (grep/awk)
4. Decide what a promoter is (i.e. 2 kb upstream of the gene) and update the coordinates accordingly (BEDfile, bedtools/awk to subtract the start coordinates in a strand-aware manner)
5. Then use a specific tool to slice the full genome to only report the DIEXF promoter (bedtools)

- The 1000 Genomes Project profiles genetic variation in humans, including SNVs and structural variation
- Practical use case: explore the events of insertion of mobile elements using the CEU population's VCF
  - How frequent are these events?
  - Where do they happen? e.g. in gene bodies, in promoters, in heterochromatic regions?
  - Is there any biological explanation for that?

```
chr22 1000 5000 cloneA 960 +
chr22 2000 6000 cloneB 900 -
```

- These records belong to a:
  1. Wiggle file
  2. VCF file
  3. GTF file
  4. BED6 file

```
fixedStep chrom=chr3 start=400601 step=100 span=5
11
22
33
```

- These records belong to a:
  1. Wiggle file
  2. VCF file
  3. GTF file
  4. BED6 file

```
#CHROM POS        ID          REF    ALT     QUAL  FILTER    INFO
20     14370      rs6054257   G      A       29    PASS      NS=3;DP=14;AF=0.5;DB;H2
20     17330      .           T      A       3     q10       NS=3;DP=11;AF=0.017
```

- These records belong to a:
  1. Wiggle file
  2. VCF file
  3. GTF file
  4. BED6 file

@MN00537:51:000H2K25G:1:11101:2213:1092 1:N:0:9

CTCCAGTCCTTACTCCCATATCTAACCTCTTACCCCTACNTCATAGGTANACATTTTAATGAAT

+

FFFFFFFFFFFFAFFFFFFFF=FFFFAFFFFFFF/AFFF#FFFFFFFFF#FFFFFFFFFFFFFFF

- This record belongs to a:
  1. Wiggle file
  2. VCF file
  3. FASTQ file
  4. BED6 file

Assuming a 0-based, half-open coordinate system and a chromosome named chr1 whose start is 'AACCGGTT...'. Which sequence is encoded as a BED3 record 'chr1 0 2'

1. AA
2. AAC
3. AC
4. A