Biological data format

Sequence: fasta and fastq

Interval: BED, GFF, GTF

Software installation

Entrez direct for data download:
```
conda install -c bioconda entrez-direct
```

Install seqkit sequence manipulation suit
```
conda install -c bioconda seqkit
```

Install seqtk
```
conda install -c bioconda seqtk
```

Install bedtools
```
conda install -c bioconda bedtools
```

Bio data formats

- **Genebank**: store sequence, functional annotations, intervals
- **Fasta**: sequence
- **Fastq**: sequence and qualities
- **bed, gff, gtf**: intervals, scores, annotations
- **sam, bam, cram**: sequence alignments (reviewed later)
- **vcf**: variant calls (reviewed later)

Gene bank

Take a look at the Hepatitis C genome with the accession number
https://www.ncbi.nlm.nih.gov/nuccore/NC_004102.1

This NCBI entry shows sequences in **Genbank** format

Let’s download and view this example:

```
$ efetch -db nuccore -id NC_004102.1 -format gb > NC_004102.1.gb
$ head -n 20 NC_004102.1.gb
```

```
LOCUS       NC_004102               9646 bp ss-RNA     linear   VRL 11-JUL-2019
DEFINITION  Hepatitis C virus genotype 1, complete genome.
ACCESSION   NC_004102
VERSION     NC_004102.1
DBLINK      BioProject: PRJNA485481
KEYWORDS    RefSeq.
SOURCE      Hepatitis C virus genotype 1
  ORGANISM  Hepatitis C virus genotype 1
            Viruses; Riboviria; Orthornavirae; Kitrinoviricota; Flasuviricetes;
            Amarillovirales; Flaviviridae; Hepacivirus.
REFERENCE   1  (bases 342 to 369; 371 to 827)
  AUTHORS   Choi,J., Xu,Z. and Ou,J.H.
  TITLE     Triple decoding of hepatitis C virus RNA by programmed
            translational frameshifting
  JOURNAL   Mol. Cell. Biol. 23 (5), 1489-1497 (2003)
   PUBMED   12588970
```

Gene bank

- Genebank is compex format that contains various types of information
- Different elements of sequence description including taxonomy
- Genomic intervals corresponding to various genomic features (3'UTR, CDS, genes)
- Links to peptide sequences
- The starting LOCUS field is not optional, without it this file will not be recognized as genebank
- The file must end with //, this is a signal for the software to stop reading the file
- Genebank file can hold nucleotides or amino-acids
- Typical extensions: **gb** or **gbk**
- Link to full specs of Genebank file:
  https://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html

Fasta format

The Genebank file always contains a sequence (nucleotide or peptide)

We can download this sequence separately in **fasta** format with efetch

```
$ efetch -db nuccore -id NC_004102.1 -format fasta > NC_004102.1.fasta
```

Typical file extension: **fasta, fa, fna,** rarely **seq**

```
$ head NC_004102.1.fasta
```

```
>NC_004102.1 Hepatitis C virus genotype 1, complete genome
GCCAGCCCCCTGATGGGGGCGACACTCCACCATGAATCACTCCCCTGTGAGGAACTACTGTCTTCACGCA
GAAAGCGTCTAGCCATGGCGTTAGTATGAGTGTCGTGCAGCCTCCAGGACCCCCCCTCCCGGGAGAGCCA
TAGTGGTCTGCGGAACCGGTGAGTACACCGGAATTGCCAGGACGACCGGGTCCTTTCTTGGATAAACCCG
CTCAATGCCTGGAGATTTGGGCGTGCCCCCGCAAGACTGCTAGCCGAGTAGTGTTGGGTCGCGAAAGGCC
TTGTGGTACTGCCTGATAGGGTGCTTGCGAGTGCCCCGGGAGGTCTCGTAGACCGTGCACCATGAGCACG
AATCCTAAACCTCAAAGAAAAACCAAACGTAACACCAACCGTCGCCCACAGGACGTCAAGTTCCCGGGTG
```

Fasta contains Id and the sequence (nucleotide or protein) itself

Id must start with **>**, this is the only requirement, the file will not be recognized as fasta without **>**

Fasta format

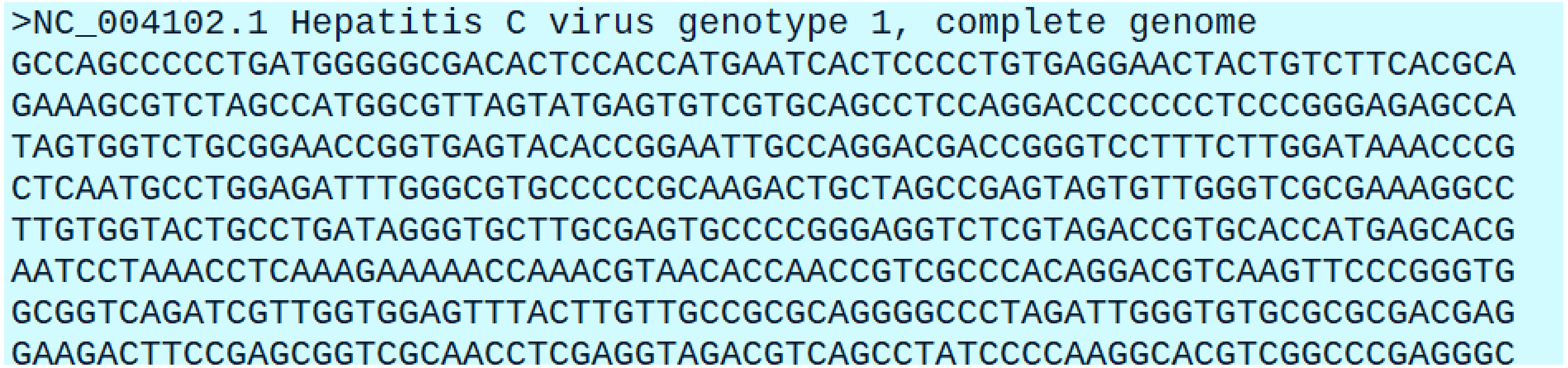
The Genebank file always contains a sequence (nucleotide or peptide)

We can download this sequence with efetch

```
$ efetch -db nuccore -id NC_004102.1 -format fasta > NC_004102.1.fasta
```

Typical file extension: **fasta, fa, fna,** rarely **seq**

```
$ head NC_004102.1.fasta
```



```
>NC_004102.1 Hepatitis C virus genotype 1, complete genome
GCCAGCCCCCTGATGGGGGCGACACTCCACCATGAATCACTCCCCTGTGAGGAACTACTGTCTTCACGCA
GAAAGCGTCTAGCCATGGCGTTAGTATGAGTGTCGTGCAGCCTCCAGGACCCCCCCTCCCGGGAGAGCCA
TAGTGGTCTGCGGAACCGGTGAGTACACCGGAATTGCCAGGACGACCGGGTCCTTTCTTGGATAAACCCG
CTCAATGCCTGGAGATTTGGGCGTGCCCCCGCAAGACTGCTAGCCGAGTAGTGTTGGGTCGCGAAAGGCC
TTGTGGTACTGCCTGATAGGGTGCTTGCGAGTGCCCCGGGAGGTCTCGTAGACCGTGCACCATGAGCACG
AATCCTAAACCTCAAAGAAAAACCAAACGTAACACCAACCGTCGCCCACAGGACGTCAAGTTCCCGGGTG
GCGGTCAGATCGTTGGTGGAGTTTACTTGTTGCCGCGCAGGGGCCCTAGATTGGGTGTGCGCGCGACGAG
GAAGACTTCCGAGCGGTCGCAACCTCGAGGTAGACGTCAGCCTATCCCCAAGGCACGTCGGCCCGAGGGC
```

Fasta format

Software to manipulate **fasta** files

fastx-toolkit: http://hannonlab.cshl.edu/fastx_toolkit/

Fasta utilities: https://github.com/jimhester/fasta_utilities

Pyfaidx: https://pypi.org/project/pyfaidx/

Seqmagick: https://github.com/fhcrc/seqmagick

Seqkit: https://bioinf.shenwei.me/seqkit/

Go ahead and install **seqkit** with conda

Scan through seqkit page to check its capabilities
https://bioinf.shenwei.me/seqkit/

Fasta format

Practice manipulating fasta files with seqkit

Download practice file:
https://raw.githubusercontent.com/slavailn/bioinf_training/main/mirna_mm10.fasta

```
$ head mirna_mm10.fasta
```

Print summary stats for a fasta file

```
$ seqkit stats mirna_mm10.fasta
```

Print name only

```
$ seqkit seq -n mirna_mm10.fasta | head
```

Print ids only

```
$ seqkit seq -n -i mirna_mm10.fasta
```

Fasta format

Practice manipulating fasta files with seqkit (continued)

Print only sequences

```
$ seqkit seq -s mirna_mm10.fasta | head
```

Remove wrap

```
$ seqkit seq -w 0 NC_004102.1.fasta | head -n 3
```

Why the output from this command looks strange?

No wrap is useful if you intend to use Unix commands that would “break” over carriage returns (**/n**), like grep, tr, etc.

Convert DNA to RNA

```
$ seqkit seq --dna2rna NC_004102.1.fasta | head
```

Convert RNA to DNA

```
$ seq --dna2rna NC_004102.1.fasta | seqkit seq --rna2dna | head
```

Fasta format

Practice manipulating fasta files with seqkit (continued)

Filter by sequence length

```
$ seq -m 20 -M 22 mirna_mm10.fasta | seqkit stats
```

Extract subsequences, for example extract first 3 bases

```
$ subseq -r 1:3 mirna_mm10.fasta | head
```

Extract last 3 bases

```
$ seqkit subseq -r -3:-1 mirna_mm10.fasta | head
```

Extract all except first 3 and 3 last bases

```
$ seqkit subseq -r 3:-3 mirna_mm10.fasta | head
```

Use sliding window to calculate GC content and show it as a table

```
$ seqkit sliding -s 5 -W 30 NC_004102.1.fasta | seqkit fx2tab -n -g | head
```

Fasta format

Practice manipulating fasta files with seqkit (continued)

Create fasta index

```
$ seqkit faidx NC_004102.1.fasta
$ cat NC_004102.1.fasta.fai
```

Format of faidx index file

- **NAME** Name of this reference sequence
- **LENGTH** Total length of this reference sequence, in bases
- **OFFSET** Offset within the FASTA file of this sequence's first base **LINEBASES** The number of bases on each line
- **LINEWIDTH** The number of bytes in each line, including the newline

Plot and collect sequence related data distributions

```
$ seqkit watch --fields ReadLen mirna_mm10.fasta -O len.png # -O len.png will save the graph as png
```

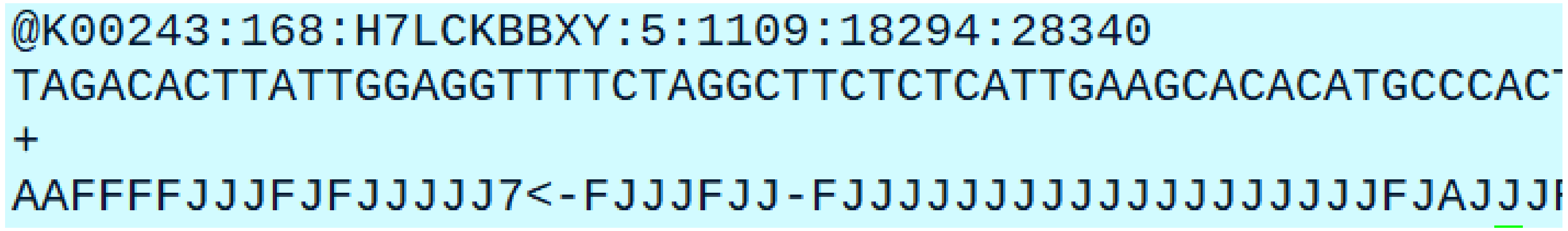
Fastq format

Fastq format contain the same data as fasta with added base quality values

Let’s download practice **fastq** file

```
$ wget https://raw.githubusercontent.com/slavailn/bioinf_training/main/sample.fastq
```

```
$ head -n 4 sample.fastq
```



```
@K00243:168:H7LCKBBXY:5:1109:18294:28340
TAGACACTTATTGGAGGTTTTCTAGGCTTCTCTCATTGAAGCACACATGCCCAC
+
AAFFFFJJJFJFJJJJ7<-FJJJFJJ-FJJJJJJJJJJJJJJJJFJAJJJ
```

- **Line 1** : must start with @
- **Line 2** : sequence
- **Line 3** : comment line
- **Line 4** : base qualities

Fastq format

The first line (ID) in fastq must follow @ and it has a specific format when generated by Illumina sequencers, see below

Example: @K00243:168:H7LCKBBXY:5:1109:18294:28340

```
K00243 : 168 : H7LCKBBXY  :  5  :  1109  :  18294  :  28340
```

- **K00243:** Instrument ID
- **168:** Run number
- **H7LCKBBXY:** Flowcell ID
- **5:** Flowcell lane
- **1109:** tile number
- **18294:** tile X coordinate
- **28340:** tile Y coordinate

Fastq format

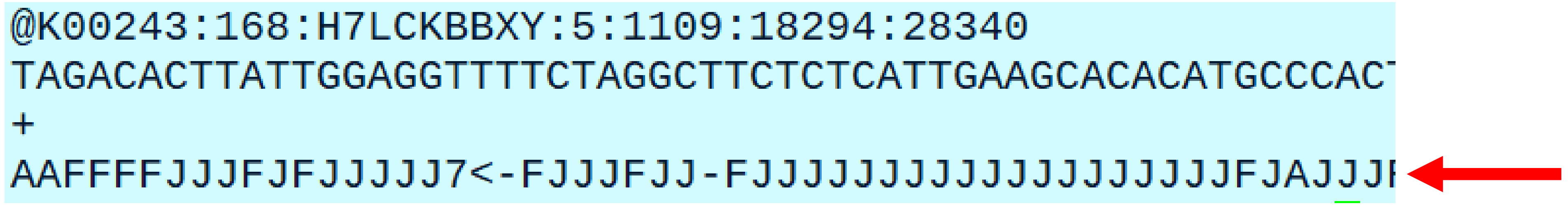
Other possible fields in Illumina ID that may follow standard fields shown on the previous slide depending on sequencing configuration

@InstrID:RunNum:FlowCell:Lane:Tile:X:I:**UMI**:**READ1/2:FILT:CONTROL:**

- **<UMI>:** Unique molecular identifier, useful for PCR duplicates filtering
- **READ:** Read 1 or 2 in paired-end sequencing
- **FILT:** passed quality filtering (Y/N)
- **CONTROL:** numeric 0, if none of the control bits are on, otherwise an even number
- **INDEX:** index sequence

Fastq format

Line 4 – base qualities



```
@K00243:168:H7LCKBBXY:5:1109:18294:28340
TAGACACTTATTGGAGGTTTTCTAGGCTTCTCTCATTGAAGCACACATGCCCAC
+
AAFFFFJJJFJFJJJJJ7<-FJJJFJJ-FJJJJJJJJJJJJJJJJJFJAJJJ
```

The qualities are integer mappings of probability that the base call is wrong

The base qualities are a way to assess the reliability of base calls

First, they were developed for Sanger sequencing

These are called Phred quality scores

$$Q_{sanger} = -10\mathrm{Log}_{10}P$$

Fastq format

Line 4 – base qualities

Interpreting the base qualities on Phred scale

- 10 → 1/10 [0.1] probability of base being wrong (Bad quality)
- 20 → 1/100 [0.01] probability of base being wrong (OK quality)
- 30 → 1/1000 [0.001] probability of base being wrong (Good quality)

Example quality line:

```
AAFFFJJFFJ-FJFJJJJJJJJJJJJJJJJJJJJAF-
```

Examples of quality encodings

Symbol	ASC code	Quality
A	65	32
F	70	37
J	74	41
-	45	12

Table of Illumina qualities from 0 - 40: https://tinyurl.com/mr2y4w7a

Fastq format

Line 4 – base qualities

Illumina had different quality encoding schemes

Since Illumina v.1.8 their sequencing base qualities are on Phred33 scale – the original format used in Sanger sequencing

Phred33 used ASCII character 33 as base, for example if the quality value is **I**:

`Q = AscII_code(‘I’) – 33 = 73 – 33 = 40`

AscII table link: https://www.ascii-code.com/

Earlier version of quality encoding since **Illumina 1.3 and before 1.8** used **Phred64** scheme

Before **Illumina 1.3** we had **ASCII 59 – 126** with quality values: **-5 - 62**

Fastq format

How to determine which version of quality encoding you are dealing with

A quality control software **FastQC** will print the encoding version as part of the analysis

We can also use **fastqFormatDetect.pl** perl script:
https://gist.github.com/tjanez/d23e20c1a777a222fd7d

Let's download this script:

```
$ wget
https://gist.githubusercontent.com/tjanez/d23e20c1a777a222fd7d/raw/afc2883838dd83981c6
442c29ec45b7be8750dac/fastqFormatDetect.pl
```

Give yourself a permission to run this script

```
$ perl fastqFormatDetect.pl # print help
$ perl fastqFormatDetect.pl sample.fastq -a
```

Fastq format

Manipulate fastqc files with ***seqtk***

Install ***seqtk*** with conda

seqtk is a fast, lightweight tool created by Heng Li designed for the processing of fasta and fastq sequences

Many functions in seqtk will overlap with those in seqkit

Take a look at documentation, there is not much
https://github.com/lh3/seqtk#readme

What tools are available?

```
$ seqtk
```

Help for individual tools

```
$ seqtk seq
```

Fastq format

Manipulate fastqc files with ***seqtk*** (continuation)

Convert fastq to fasta

```
$ seqtk seq -A sample.fastq | head
```

Mask bases with quality lower than integer

```
$ seqtk seq -q 20 sample.fastq | grep '[acgt]’
```

Drop sequences shorter than integer

```
$ seqtk seq -L 20 mirna_mm10.fasta | grep '^>' | wc –l
```

Reverse complement

```
$ seqtk seq -r mirna_mm10.fasta | head
$ head mirna_mm10.fasta
```

Take a fraction of the reads as a random subsample

```
$ seqtk seq -f 0.1 sample.fastq | seqkit stats # take 10% percent of reads as random sample
```

Fastq format

Manipulate fastqc files with ***seqtk*** (continuation)

Why do we need ‘seed’ with random sampling?

Setting ‘seed’ will ensure that the same random numbers will be generated between sampling procedures

```
$ seqtk seq -f 0.1 -s 10 sample.fastq | head
$ seqtk seq -f 0.1 -s 10 sample.fastq | head # these sampling procedures have the same seed and will retrieve the same sequences
```

```
$ seqtk seq -f 0.1 -s 100 sample.fastq | head # changing the seed will result in different sequences being retrieved
```

We must set the same seed for Read 1 and Read 2 when down-sampling paired-end reads

Identify high or low GC regions

```
$ seqtk gc # look at help
$ seqtk gc NC_004102.1.fasta
```

Fastq format

Manipulate fastqc files with ***seqtk*** (continuation)

Get nucleotide composition

```
$ seqtk comp
```

Help seems unclear, what is the meaning of columns?
https://github.com/lh3/seqtk/issues/47

```
$ seqtk comp NC_004102.1.fasta
```

Introduce point mutation

```
$ seqtk mutfa
```

```
$ head NC_004102.1.fasta # print head of Hepatitis C genome
```

Change C at position 3 to A

```
$ echo 'NC_004102.1 3 bla A' > in.snp
$ seqtk mutfa NC_004102.1.fasta in.snp | head
```

Interval formats

In genomics we frequently deal with **interval**-type data

- Intervals are also called **ranges**
- Interval describes a genomic position of a subsequence
- In the simplest case we only need the **name of the sequence (chromosome)**, the **start** and the **end** of the subsequence to describe the interval

Example:
>NC12345
ACT<u>**GGG**</u>TCAATG

If positions are 1-based, the subsequence **GGG** can be described as follows:

Chr	**start**	**end**
NC12345	**4**	**6**

This is an example of BED file in its simplest form

Interval formats

BED format

In the minimal case, bed format requires only 3 columns separated by TAB: **chr, start, end**

BED file below will describe 3 intervals in Hep C genome

```
NC_004102.1     3       10
NC_004102.1     25      56
NC_004102.1     50      65
```

There are different versions of BED files that contain additional attributes, such as, name, strand, score and others

The most informative BED format contains 12 columns

Interval formats

BED format

Column	Title	Description
1	Chrom	Chromosome, Scaffold, sequence
2	Start	Start position (0-based)
3	End	End position (1-based)
4	Name	Name of the interval
5	Score	Score associated with the interval (for example, p-value)
6	Strand	Forward or reverse strand
7	ThickStart	Start of the thick block in the browser
8	ThickEnd	End of the thick block in the browser
9	itemRGB	Color of the block as it appears in the browser
10	BlockCount	Number of blocks (useful for exons)
11	BlockSizes	Size of the blocks (exons)
12	BlockStarts	Start of the blocks (exons)

Interval formats

BED format

- BED files can have an optional header with one or more lines of text
- Header has no established format
- A header typically gives instructions to genomic browser regarding display of the intervals stored in the bed file or provide information about the file

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

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

Interval formats

BED format

Bed files are given extension *.bed,* or *bed3, bed9, bed12* based on the number of columns in the file

Software that manipulates bed and other interval files:
Bedtools https://bedtools.readthedocs.io/en/latest/

Bedops https://bedops.readthedocs.io/en/latest/

Bedtk https://github.com/lh3/bedtk

GFF (general feature format) and **GTF** files is another way address genomic intervals, like **BED**, they are tab delimited text files

GFF/GTF files have 9 columns, the coordinates start at 1

Interval formats

GFF/GTF format

Column	Title	Description
	seqid	Chromosome, contig, name of the sequence
	source	Algorithm, procedure that generated the feature
	type	The feature type like gene, transcript, exon, etc
	start	Feature start, 1-based
	end	Feature end, 1-based
	score	Numeric value associated with the feature (for example p-value)
	strand	Forward or reverse
	Phase	Coding sequence (CDS) phase, can be 0,1, or 2. Phase is relative of open reading frame, it indicates how many bases need to be removed (0, 1 or 2) from the start of CDS to reach the next codon
	Attributes	A list of “tag:value” pairs separated by semicolons

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

Interval formats

GFF/GTF format

GFF example

```
browser position chr22:10000000-10025000
browser hide all
track name=regulatory description="TeleGene(tm) Regulatory Regions" visibility=2
chr22   TeleGene        enhancer        10000000        10001000        500     +       .       touch1
chr22   TeleGene        promoter        10010000        10010100        900     +       .       touch1
chr22   TeleGene        promoter        10020000        10025000        800     -       .       touch2
```

GTF is an extension of GFF

8 fields first fields in GTF file are the same as in GFF

The 9th column is different and must contain ***gene_is*** and ***transcript_id*** attributes

Example of 9th field of the GFT file

```
gene_id "Em:U62317.C22.6.mRNA"; transcript_id "Em:U62317.C22.6.mRNA"; exon_number 1
```

Working with bedtools

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

Bedtools is comprehensive toolset for interval arithmetic

It has many functions, but the main ones include calculating intersect, complement, merge, count, and shuffle intervals

Bedtools has a comprehensive tutorial:
http://quinlanlab.org/tutorials/bedtools/bedtools.html

List the tools

```
$ bedtools -h
```