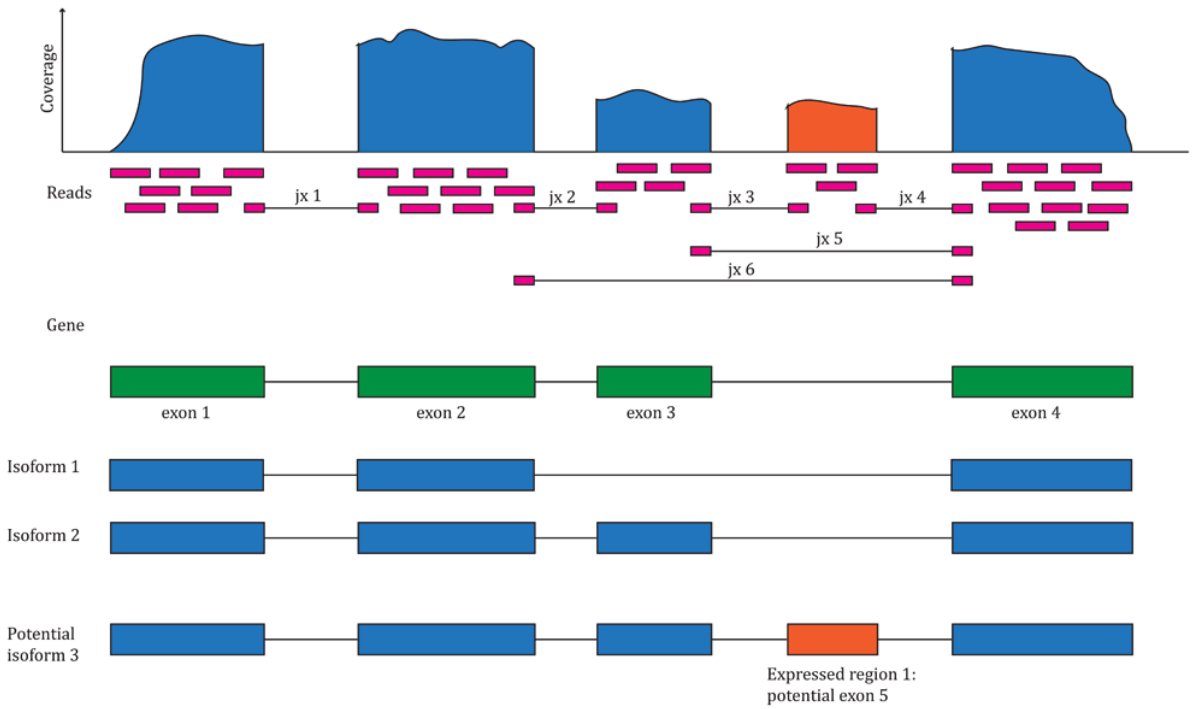
Introduction to RNAseq

Version 1

Contents

Introduction to RNAseq



These are course notes for the "Introduction to RNAseq" course given by the Monash Bioinformatics Platform[1] for the Monash Data Fluency[2] initiative. Our teaching style is based on the style of The Carpentries[3].

- PDF version for printing[4]

Source code

- GitHub page[5]

Authors and copyright

This course is developed for the Monash Bioinformatics Platform by Stuart Archer, Nick Wong and Kirill Tsyganov.

[1] https://www.monash.edu/researchinfrastructure/bioinformatics
[2] https://monashdatafluency.github.io/
[3] https://carpentries.org/
[4] https://monashdatafluency.github.io/rnaseq-intro/rnaseq-intro.pdf
[5] https://github.com/MonashDataFluency/rnaseq-intro

[6] http://creativecommons.org/licenses/by/4.0/

Chapter 1

Genomics

Genome is a collection of all genetic material in the organism. To be more accurate each cell in our body contains a copy of entire genome. The study of genomics is all things to do with genome including gene expression, methylation profiles, transcriptional factor binding, chromatic arrangements and three dimensional structure of chromosomes and many more things.

In this course we are going to focus on one particular technique of genomics - RNA sequencing or RNAseq for short. This is a study of primarily but not limited to gene expression. Other common application of RNAseq are study of splicing and isoforms.

In genomics everything rotates around sequence information, let's look into biological sequence next.

1.1 Sequence

A sequence is a generic name describing the order of biological letters in molecules like DNA, RNA or piptide and protein. In the case of DNA and RNA letters mean nucleotide bases, **A**denine, **T**hymine, **C**ytosine **G**uanine or ATCG for short. Also know that mRNA contains **U**racil instead of Thymine. This would be one way to represent all five possible bases `A[T|U]CG` In the case of piptide/protein molecules each letter means amino acid type, I won't list discuss proteins any further in this book since it isn't a focus on this book.

Note that there is a greater nomenclature exists to cover all possible variances

```
##    Letter_code      Base_name
## 1            A        Adenine
## 2            C       Cytosine
## 3            G        Guanine
## 4          T|U Thymine|Uracil
## 5            R            A|G
## 6            Y            C|T
## 7            S            G|C
## 8            W            A|T
## 9            K            G|T
## 10           M            A|C
## 11           B          C|G|T
## 12           D          A|G|T
## 13           H          A|C|T
## 14           V          A|C|G
## 15           N       any base
```

The reason we have sequence information of any kind is because someone somewhere made an afford to sequence some part of the molecule. In the case of whole genome sequencing an afford was made to sequence the entire genome, for human the entire genome is 3 billion bases long. However this is a sum of

all bases across all chromosomes, but chromosomes are not all equal, below is a sequence lenght of each chromosome.

```
##    Chrom    Length
## 1      1 248956422
## 2      2 242193529
## 3      3 198295559
## 4      4 190214555
## 5      5 181538259
## 6      6 170805979
## 7      7 159345973
## 8      8 145138636
## 9      9 138394717
## 10    10 133797422
## 11    11 135086622
## 12    12 133275309
## 13    13 114364328
## 14    14 107043718
## 15    15 101991189
## 16    16  90338345
## 17    17  83257441
## 18    18  80373285
## 19    19  58617616
## 20    20  64444167
## 21    21  46709983
## 22    22  50818468
## 23     X 156040895
## 24     Y  57227415
## 25    MT     16569
```

While sequence is a generic name and often times it needs some context around it, for example DNA sequence, RNA sequence or chromosome 3 sequence. There are other names like contig and a read, which are both sequences too, they mean more specific thing. In general both mean shorter sequence, where contig almost always should be longer then a read.

Let's talk about a read sequence next.

1.2 Read sequence

A read is product of a machine read out that takes some biological material i.e DNA or RNA sample and produces number of reads that resembles what you've put into the machine. There have been a few iteration of sequencing technologies, each iteration brings about higher yield (more reads) and longer reads. Current players in the genomics space are:

Short read technologies

- Illumina[1]
  - MiSeq
  - HiSeq
  - NextSeq
  - NovaSeq
- BGI platforms[2]
  - BGISEQ-50
  - BGISEQ-500

[1] https://www.illumina.com/
[2] https://www.bgi.com/global/resources/sequencing-platforms/

- Ion Torrent[3]

Medium to long read technologies

- PacBio[4]
  - Sequel System
- Oxford Nanopore[5]
  - MinION
  - GridION
  - PromethION

All of the above technologies will one way or another result in digital reads. As name implies short reads are typically 50-300 bases, whereas long reads can range from 1k to 10k or more. Usually the quantity and the length of the read are inversely proportional, however the throughput of a short reads technology at the moment is vastly greater then the long read and it doesn't suffer from the coverage issue. In terms of number of reads we are talking about millions to billions for short reads technology and in the range of 1 to 4 millions for long reads. This is a good plot showing nanopore output, which illustrates quantity vs length nicely.

Getting the reads is just a first step. You have to do something with your reads. The two path one can take are really assemble your reads into contigs and further into full genome or map your reads to an existing reference genome. In this course we'll be taking the later path - mapping our RNAseq reads to an existing reference genome. But let me just talk a little more about what a contig sequence is next.

1.3 Contig sequence

Contig has specific meaning in genome assembly. It is a larger fragment assembled from short (100-300 bp) reads that hasn't yet been merged into chromosome. However often times this term can be loosely used to mean any long stretch of sequence. I often (incorrectly) refer to chromosomes as different contigs in the FASTA file. With long read technologies advancing pretty rapidly it breaks conventional definitions of a read and contig, where a single read can be longer then a contig. However assembling two long reads would still result in a contig sequence. So once again it is important to give more context when using this term.

1.4 Summary

- **Genomics** is all things genome that rotates around genomic sequence
- **Sequence** is an ordered collection of nucleotide bases
- **Read** is an output from a sequencing machine that covered biological sample in digital sequence
- **RNAseq** focuses on gene expression and splicing isoforms

[3] https://www.thermofisher.com/au/en/home/brands/ion-torrent.html
[4] https://www.pacb.com/
[5] https://nanoporetech.com/

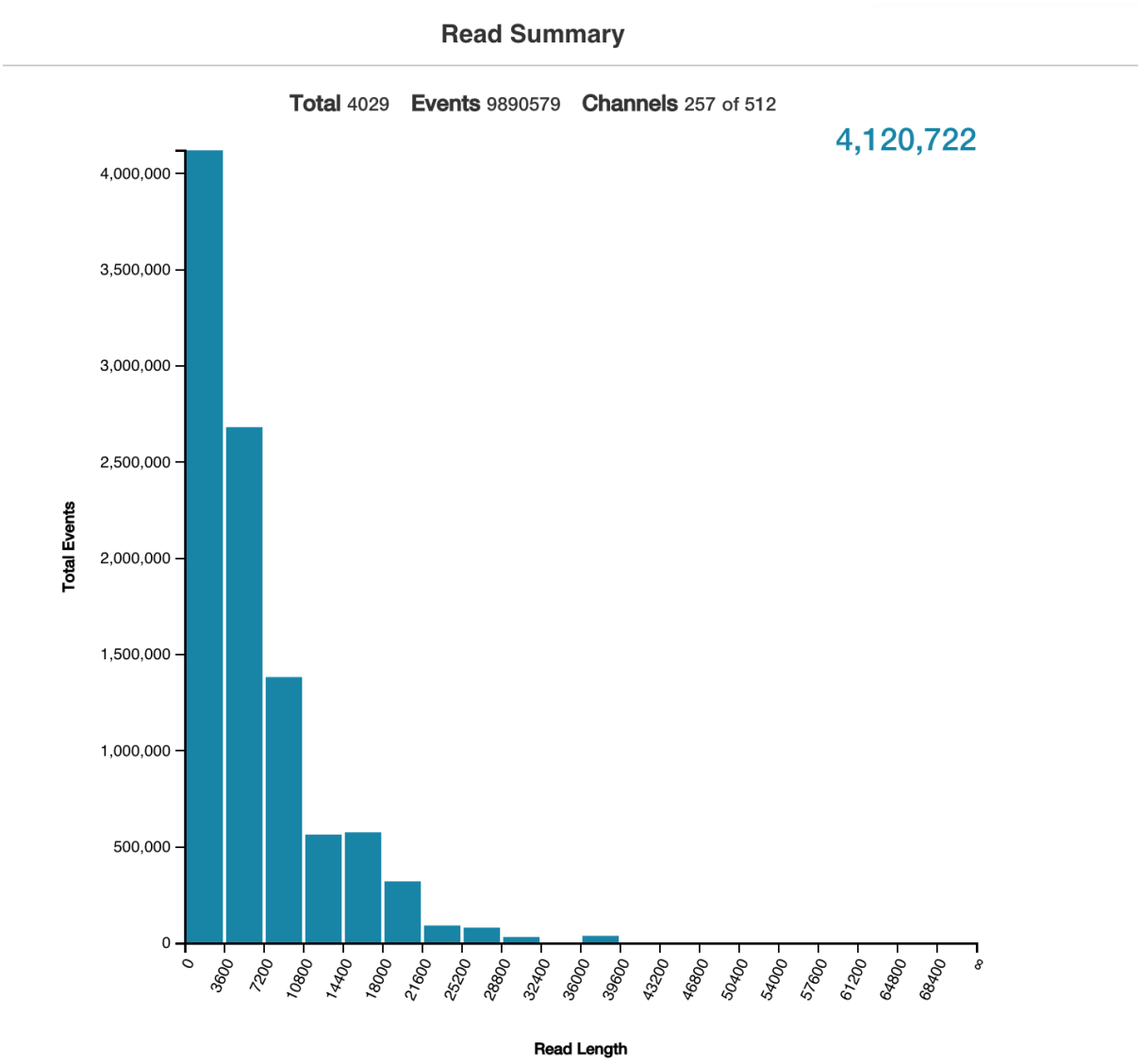


Figure 1.1: https://bioinformatics.stackexchange.com/questions/3623/how-many-reads-has-my-sequencing-run-produced-on-minion

Chapter 2

Reference sequence

In this course will be focusing on DNA/RNA molecules and hence our sequences will be nucleotides sequences. In the case of *Mus_musculus* (mouse) genome, we have 19 autosomal chromosomes and two sex chromosomes X and Y plus a mitochondrial chromosome (genome). This makes up 22 separate molecules and therefore 22 separate sequences. Typically that sequence is stored in **FASTA** file format

- FASTQ for raw short reads
- FAST5 for raw long reads
- FASTA for any other kind of sequence

The other possible bioinformatics files that we weren't focus on are, genbank and gff. Both of those files can hold annotation and sequence information in one file. It often works fine with small genomes, but large eukaryotic genomes are better split into two files reference sequence and annotation information.

2.0.1 FASTA

FASTA file typically can hold any type of sequence data. We are not going to talk about amino acid sequence in this course and so for your purpose we can think of FASTA as just holding nucleotides sequence. That sequence can be anything, primer or adapter sequence, individual transcripts (genes) sequence OR entire chromosome sequence. In the case when FASTA holds entire genome, that is all chromosomes for a given species, we tend to refer to that FASTA file as a reference genome file. If I'm asking biologist of a **reference genome**, I typically imply that I need a FASTA file with genomic sequence.

Be aware that there isn't such thing as "the genome" for a particular species. This is because any genome is a

There are a few different fasta files, but all of them will hold some sort of sequence. Sometime you'll see `.fna` to inidicate nuclear sequence or `.faa` to indicate amino acid sequence, but more generally it'll be `.fa` or a longer version `.fasta`. Like any biological plain text file. This most certainly will be zipped so expect to see `.fa.gz` or `.fasta.gz`.

Note that our purpose we can think of FASTA file holding our reference genome sequence of intereste e.g mouse genome.

This is previously assembled genome that was made puplically available. There are several different vendors that provide reference genomes. In general either one of these three will is a good place to start looking for you reference genome.

- Ensembl[1]
- RefSeq[2]
- UCSC genomes[3]

[1] https://asia.ensembl.org/index.html
[2] https://www.ncbi.nlm.nih.gov/refseq/
[3] http://hgdownload.soe.ucsc.edu/downloads.html

Those will have majority of species, however it is common to have a stand along community that host reference genomes from a different place. A good example is yeast genome database[4]. If you are working with yeast this is the place to look for reference genomes.

It is important to note that reference genome isn't a defined sequence. It is simply the best we have so far and it is a continuos effort to improve quality of the genomes. For widely used model organisms like mouse and yeast their genomes have been sequenced and assembled many times over the years and we have very good quality genomes. Species like axolotl or spiny mouse are less commonly used as model organism in comparison to mouse genomes and quality of those genomes is poorer. There are however other factors in place that influence genome quality e.g genome complexety.

2.1 Exercise

- go to Ensembl website[5]
- from the drop down on the left hand site select *Saccharomyces_cerevisiae*, which is a yeast species.
- click on download FASTA file. This will redirect you to ftp site
- find file **Saccharomyces_cerevisiae.R64-1-1.dna_sm.toplevel.fa.gz** and download it by clicking
- once downloaded double click on the file to open it and view the content.

depending on the operating system that you have you might have to unzip fasta file first before opening. It is best not to open with MS Office suite. Use text editor instead. On windows it'll be `notepad` on macOS it'll `textmate`.

At the very top you should see this

```
>I dna_sm:chromosome chromosome:R64-1-1:I:1:230218:1 REF
ccacaccacacccacacacccacacaccacaccacacaccacaccacacccacacacaca
caTCCTAACACTACCCTAACACAGCCCTAATCTAACCCTGGCCAACCTGTCTCTCAACTT
ACCCTCCATTACCCTGCCTCCACTCGTTACCCTGTCCCATTCAACCATACCACTCCGAAC
```

Note the very first line.

```
>I dna_sm:chromosome chromosome:R64-1-1:I:1:230218:1 REF
```

This is a FASTA header that indicates which sequence is follows. In this case it is `I` i.e sequence from chromosome one of yeast. If there are multiple chromosomes (usually the case for eukaryotic species) then you keep reading the sequence until you hit either the end of the file OR the next ">" sign

NOTE the actual sequnce have been truncated

```
>I dna_sm:chromosome chromosome:R64-1-1:I:1:230218:1 REF
ccacaccacacccacacacccacacaccacaccacacaccacaccacacccacacacaca
caTCCTAACACTACCCTAACACAGCCCTAATCTAACCCTGGCCAACCTGTCTCTCAACTT
ACCCTCCATTACCCTGCCTCCACTCGTTACCCTGTCCCATTCAACCATACCACTCCGAAC
>II dna_sm:chromosome chromosome:R64-1-1:II:1:813184:1 REF
CCCACACACCACACCCACACCACACCCACACACCACACACACCACACCCACACACCCACA
CCACACCACACCCACACCACACCCACACACCCACACCCACACACCACACCCACACACACC
ACACCCACACACACCCACACCCACACACCACACCCACACACACACCACACCCACACACAC
CACACCACACCCACACCACACCCACACCCACACACCACACCCACACCCACACCCCACACC
```

The format of the FASTA header is rather loosely defined, but these are key points:

- must start with ">" sign
- the contig/chromsome name follows straight after ">" sign. Can be any string
- anything after a space is a comment/description

Below are all valid FASTA headers

```
>I
```

[4] https://www.yeastgenome.org/
[5] https://asia.ensembl.org/index.html

```
>chrI
```

```
>NC_000067.6
```

```
>I_named_it
```

Each header in a single FASTA file has to be uqniue in order to identify each sequence.

Are this two FASTA header the same?

```
>I
```

```
>I named it
```

Are this two FASTA header unique?

```
>I
```

```
>I_named_it
```

While this apper to be very are small details, the reason it is important because you want to know which chromosome (molecule) your reads came from.

2.2 Rule of thumb

Unless you are working with some exotic species and/or have a strong prefernce for genome vendor you can generally leave this part up to your bioinformatician to sort out. It is important to know that there are different vendors that have different versions (builds) of genomes.

. Often times bacterial RNAseq data come with some custome reference and annotation files. If that happens bioinformatician will need to convert everythign to FASTA and GTF/GFF files.

Chapter 3

Gene annotation

As discussed in [reference genome] module. The reference genome is just a string of charaters.

Chapter 4

Bioinformatics file formats

Before diving into specific file formats. I would like to discuss what could file actually mean and hold in general, after all bioinformatics files aren't that different form any other files. In fact fair number of bioinformatics files are just a variant of TSV file format, discussed shortly. Bioinformatics world largely rotates around Unix/Linux ecosystem, and remember that macOS is Unix derived operating system too. That means that a lot of the tooling and compute happens on linux computers often referred to as servers. In unix world file name is simply a string of letters (text) that points to a location of that file content that is stored elsewhere. Therefore the only thing that matters for file names is they are unique. Often people get caught out by file's extensions, they do not matter. File extentions are simply an indicator for a user what to expect when they are going to open that file. To more accurate other programs often look at file extensions too trying to guess if this is the file format they can handle. In general it is a very good idea to have correct file extension. I just want you to know that just because you've renamed a file from `ref.fasta` to `ref.fa` nothing had happened to it content. Bioinformatics is an evolving field and new technologies will bring about new file formats.

4.1 Introduction

Broadly speaking any computer file is either a plain text or binary format. Under the hood everything is binary of course, but a plain text file can be looked at and modified using standard unix/linux tool sets as well simply opened with any text editor for more custom modification. If the file isn't plain text you have to have specific tool to work with that file type. For example pdf files aren't plain text and this is why you need pdf viewer (e.g adobe reader) to work with pdfs. Similarly in bioinformatics BAM files aren't plain text and do require `samtools` to work with it. As I've hinted earlier large number of bioinformatics files are plain text, however a lot of the times plain text files are ziped with `gzip` tool adding another suffix to the file name and thereby changing the file into binary for that now will require `gzip` tool to work with. We zip file to reduce they size. Zipping works particular well on files that have repetitive patterns e.g repeats of ATCG sequence.

Note the difference in size is essentially three times !

```
-rw-r--r-- 1 kirill kirill 3.0G Apr 10 10:11 Homo_sapiens.GRCh38.dna_sm.removed_contigs.fa
-rw-r--r-- 1 kirill kirill 895M Apr 10 10:10 Homo_sapiens.GRCh38.dna_sm.removed_contigs.fa.gz
```

It is a natural behaviour wanting to unzip file before looking inside. It is okay to do so, but be aware that this isn't the best practice, and you don't want to be storing your files in plain text for a long time, this will eat up your disk space very quickly.

`example of large BAM file and plain text SAM`

Remember that best practice is always keep your files zipped

Let's talk next about what makes different file formats different

4.2 What is a file format?

The file format at it simplest defines structure with in the file. By far the most common structure of the plain text file is tabular. The spreadsheet that everyone is so familiar with is simply a plain text file with rows and columns, where each cell separated by a "special" character called `tabular` or `tab` for short, which makes tab separated variables file or tsv for short with a sister file of comma separated variables or csv for short where instead of `tab` character comma is used.

Comma Separated Variables (csv)

```
Gene.ID Chrom   Gene.Name   Biotype KO3_S3  KO4_S4  WT1_S1  WT2_S2
ENSMUSG00000051951  1   Xkr4    protein_coding  0   0   0   0
ENSMUSG00000025900  1   Rp1 protein_coding  0   0   0   0
ENSMUSG00000109048  1   Rp1 protein_coding  0   0   0   0
ENSMUSG00000025902  1   Sox17   protein_coding  0   0   0   0
ENSMUSG00000033845  1   Mrpl15  protein_coding  934 1317    888 939
ENSMUSG00000025903  1   Lypla1  protein_coding  705 848 647 747
ENSMUSG00000104217  1   Gm37988 protein_coding  0   0   0   0
ENSMUSG00000033813  1   Tcea1   protein_coding  354 436 368 442
ENSMUSG00000002459  1   Rgs20   protein_coding  0   0   0   0
```

Tab Separated Variables (tsv)

```
Gene.ID,Chrom,Gene.Name,Biotype,KO3_S3,KO4_S4,WT1_S1,WT2_S2
ENSMUSG00000051951,1,Xkr4,protein_coding,0,0,0,0
ENSMUSG00000025900,1,Rp1,protein_coding,0,0,0,0
ENSMUSG00000109048,1,Rp1,protein_coding,0,0,0,0
ENSMUSG00000025902,1,Sox17,protein_coding,0,0,0,0
ENSMUSG00000033845,1,Mrpl15,protein_coding,934,1317,888,939
ENSMUSG00000025903,1,Lypla1,protein_coding,705,848,647,747
ENSMUSG00000104217,1,Gm37988,protein_coding,0,0,0,0
ENSMUSG00000033813,1,Tcea1,protein_coding,354,436,368,442
ENSMUSG00000002459,1,Rgs20,protein_coding,0,0,0,0
```

Common bioinformatics files such gff/gtf, bed and vcf are all variant of tsv in some respect except with defined column names and hence value types that can fit in those columns. I haven't explained what each of those files used for. Let's focus on gff/gtf format only and leave the other two for later discussion.

4.3 Annotation files

The essence of annotation file, guessable from the name, is to provide information of coordinates of features in the genome. A feature could mean several things including a gene, transcript, exon, 3'UTR or any other region of the gnome that is "worth" documenting. Basically this is your google map without pictures. If you think about what information would you need in order to identify a gene for example well known tumor suppressor gene `TP53`.

- bear minimum we need to know which chromosome it is one
- where about it is on chromosome 17
- remember that humans are diploid - which strand of chromosome 17 it is on

In a simply tabular format this would look like this

```
tp53    chr17   7661779 7687550 +
```

Turns out that this is actually is an annotation file format called Simplified Annotation Format (SAF)[1]. It isn't that widely used, because it has some limitations, which is outside of the course to discuss and there are at least three other annotation formats that exist, which we not going to discuss any further. Side tracking to our earlier discussion about file extensions, `saf` is a file format given that it has defined structure, but what should the extension be? Anything you like `.txt` because it is a text file, `.tsv` well it

[1] http://bioinf.wehi.edu.au/featureCounts/

is tab separated. I usually do `.saf`, because that tells me straight away exactly what the content of the file is. This is what file extension supposed to do.

The file format that we are going to focus on is General Feature Format (gff)

- Plain text file format
- The format was proposed as a means to transfer feature information
- We do not intend GFF format to be used for complete data management of the analysis and annotation of genomic sequence
- Originally wanted file format that will be easily parsable by Unix tools such `grep`, `sort` and `awk`

For the purpose of understanding RNAseq experiment the bear minimum you'll need to understand are these three file types:

- FASTQ
- FASTA
- GFF/GTF

4.3.0.1 FASTQ

This is a very important file(s) you don't mess directly with FASTQ files. Best not to every unzip those either. If think that you need to do some thing fastq files in terms of editing or some other out of ordinary manipulations. Make a copy and work on the copy. If you modify, corrupt or loose your FASTQ files, days, month or year of wet-lab work go down a toilet, let a lone large sum of money spend on RNA extraction, library prepartion and sequencing.

4.3.0.2 FASTA

There are a few different fasta files, but all of them will hold some sort of sequence. Sometime you'll see `.fna` to inidicate nuclear sequence or `.faa` to indicate amino acid sequence, but more generally it'll be `.fa` or a longer version `.fasta`. Like any biological plain text file. This most certainly will be zipped so expect to see `.fa.gz` or `.fasta.gz`.

For the purpose of RNAseq analysis know that FASTA file holds your reference sequence.

4.3.0.3 GTF/GFF

These are annotation files

There are many different biology / bioinformatics related file formats. You will naturally come across different files formats. It doesn't matter so much what they are called, i.e what they file extention is, rather what they hold.

Essentially all bioinformaticie files are either plain text (a.k.a ASCII) or binnary. Plain text files are "easy" to interface with. Unix/Linux command line provide many great tools that can work with plain text files. To list a few populart and powerful command line tools:

- `grep` search for a pattern
- `sed` search and replace
- `awk` search and replace and more
- `cat` look inside
- `cut` what the knife would do to bread

It is out of the scope of this course to go through any of the tools or command line in general, but if you are going to do any kind of bioinformatics you'll eventually going to come across these tools.

The other file type is binnary, this will requires designated tool for interfacing with the file for example SAM/BAM files have `samtools` that aid interaction with them.

4.4 Plain text files

Given that we know the structure of the file TSV or CSV we can use standard command line tools to get relevant/interesting bits e.g

```
~$ cut -f1,3,5-6 t.tsv

Gene.ID Gene.Name   KO3_S3  KO4_S4
ENSMUSG00000051951  Xkr4    0   0
ENSMUSG00000025900  Rp1 0   0
ENSMUSG00000109048  Rp1 0   0
ENSMUSG00000025902  Sox17   0   0
ENSMUSG00000033845  Mrpl15  934 1317
ENSMUSG00000025903  Lypla1  705 848
ENSMUSG00000104217  Gm37988 0   0
ENSMUSG00000033813  Tcea1   354 436
ENSMUSG00000002459  Rgs20   0   0
```

4.5 Glossary

- interface: extract or insert relevant information

Chapter 5

GFF 1, 2 and 3

The essence of GFF file format is to avoid databases

5.1 General Feature Format

A 'Feature' could mean complete gene, RNA transcript or protein structure or pretty much anything

- Plain text file format
- The format was proposed as a means to transfer feature information
- We do not intend GFF format to be used for complete data management of the analysis and annotation of genomic sequence
- Originally wanted file format that will be easily parsable by Unix tools such `grep`, `sort` and `awk`

5.2 General Feature Format Version 1 (GFF1)

- I have never seen one
- I think it is obsolete and was superseded by GFF2

The main change from version 1 to 2 is the requirement for a tag-value type structure

5.3 General Feature Format Version 2 (GFF2)

- 9 fields
- Tab separated

```
<SeqName>\t<Source>\t<Feature>\t<start>\t<end>\t<score>\t<strand>\t<frame>\t<[Attributes]>\t<[Commen
```

- The attribute field is some what free form:
  1. It must have Tag-Value Pairing, where each pair is separated by semicolon
  2. The Tag name can be anything within `[A-Za-z][A-Za-z0-9_]*`
  3. The value can be anything. Free text must be surrounded by double quotes
  4. All 'special' Unix character must be properly escaped e.g newline as `\n` and tab as `\t`
- General Feature Format Version 2 (GFF2) specifications[1]

All other flavours of GFF's and GTF's are divergent of GFF2

[1] http://www.sanger.ac.uk/resources/software/gff/spec.html

5.4 General Feature Format Version 3 (GFF3)

- Reason for 'new', GFF3 format is that GFF2 has become insuffcient for bioinformatcians
- Key aspects about GFF3:
  1. Adds a mechanism for representing more than one level of hierarchical grouping of features and subfeatures.
  2. Separates the ideas of group membership and feature name/id.
  3. Constrains the feature type field to be taken from a controlled vocabulary.
  4. Allows a single feature, such as an exon, to belong to more than one group at a time.
  5. Provides an explicit convention for pairwise alignments.
  6. Provides an explicit convention for features that occupy disjunct regions.
- Tag-Value pairing is now must be separated by an `=` sign
- Tag-Value pairs are still separated by `;`
- Predefined meaning for some tags
  1. **ID** Indicates the ID of the feature. IDs for each feature must be unique within the scope of the GFF file. In the case of discontinuous features (i.e. a single feature that exists over multiple genomic locations) the same ID may appear on multiple lines. All lines that share an ID collectively represent a single feature.
  2. **Name** Display name for the feature. This is the name to be displayed to the user. Unlike IDs, there is no requirement that the Name be unique within the file.
  3. **Alias** A secondary name for the feature. It is suggested that this tag be used whenever a secondary identifier for the feature is needed, such as locus names and accession numbers. Unlike ID, there is no requirement that Alias be unique within the file.
  4. **Parent** Indicates the parent of the feature. A parent ID can be used to group exons into transcripts, transcripts into genes, an so forth. A feature may have multiple parents. Parent can *only* be used to indicate a partof relationship.
  5. **Target** Indicates the target of a nucleotide-to-nucleotide or protein-to-nucleotide alignment. The format of the value is "target_id start end strand", where strand is optional and may be "+" or "-". If the target_id contains spaces, they must be escaped as hex escape
  6. **Gap** The alignment of the feature to the target if the two are not collinear (e.g. contain gaps). The alignment format is taken from the CIGAR format described in the Exonerate documentation. See "THE GAP ATTRIBUTE" for a description of this format.
  7. **Derives_from** Used to disambiguate the relationship between one feature and another when the relationship is a temporal one rather than a purely structural "part of" one. This is needed for polycistronic genes. See "PATHOLOGICAL CASES" for further discussion.
  8. **Note** A free text note.
  9. **Dbxref** A database cross reference. See the section "Ontology Associations and Db Cross References" for details on the format.
  10. **Ontology_term** A cross reference to an ontology term. See the section "Ontology Associations and Db Cross References" for details.
  11. **Is_circular** A flag to indicate whether a feature is circular. See extended discussion below.
- Able to add sequence to the GFF3 file. Use `## FASTA` as a separator line between annotation and sequence
- Generic Feature Format Version 3 (GFF3) specifications[2]
  1. **Tag-Value pair separated by =**
  2. **Tag-Value pairs separated by ;**
  3. **You can now associate features together through `ID/Parent` tag**

[2] http://www.sequenceontology.org/gff3.shtml

5.5 Gene Transfer Format (GTF)

- `GTF` is a refinement of `GFF2` and is sometimes referred to as `GFF2.5`

Be careful about this assumption ! Some tools might produce/convert your other `GFF` file to `GFF2.5`, but this new `GFF2.5` file might not be compatible with your specific tool that expects `GTF` file

- `GTF` file is somewhat a subclass of `GFF`
- Original GTF specification said that all features must have two mandatory attributes:
  1. `gene_id` *value*
  2. `transcript_id` *value*
- This is to handle different transcript from the same gene
- However GENCODE[3] has more mandatory fields
- Tag-Value pair separated by space !
- Tag-Value paies separated by `;`
- Gene Transfer Format (GTF) specification[4]

To me at least `GTF` is the most established and predefined gene annotation format

Three GFF versions comparison[5]

5.6 How to parse GFF/GTF file..?

1. Unix tools
   - `grep Grhl1 yourAnnotationFile.gff | less`
   - `grep -w gene yourAnnotationFile.gff | less`
   - `grep -w exon yourAnnotationFile.gff | less`
   - `cut -f1,3,4,5,7 yourAnnotationFile.gff | less`
2. Text Editor or spreadsheet tools
   - Vim
   - Gedit
   - sublime
   - LibreOffice
   - MS Office
3. Programmatically
   - Python
   - R
   - Perl

5.6.1 Parse GFF/GTF with Python

- Tricky to associate features together e.g exon to transcript and transcript to gene
- Need to 'look ahead and look behind', but how far ..?
- Different gene will have different number of features (lines) associated with it
- GFF/GTF is rather big file around 1.0 - 1.5 Gb (the size will really depend on the species)
- Can make one big hash (dictionary) and keep it all in memory..
- Best I think to write your personalised hash (dictionary) to a file

[3] http://www.gencodegenes.org/gencodeformat.html
[4] http://mblab.wustl.edu/GTF2.html
[5] http://www.broadinstitute.org/annotation/argo/help/gff.html

5.6.2 Python packages for dealing with GFF/GTF

1. Nice bunch of functions from Gist[6]
   - Found this too slow and didn't want to keep it all in memory
2. bcbio-gff a.k.a GFF parser[7]
   - Also found it to be slow and confusing
3. gffutils[8]
   - Really liked it using it and now use if all the time
   - Two step process:
     1. Make database file `.db` from your GFF/GTF file
     2. Parse anything you want from your `.db` file

```
db = gffutils.FeatureDB(dbFile, keep_order=True)
features = db.all_features()
```

- `features` is now your generator object, meaning you can loop over it

```
db = gffutils.FeatureDB(dbFile, keep_order=True)
features = db.all_features()

for feature in features:
    print feature
```

- And now you can get all this things for your single feature (GFF line)
  - `astuple`
  - `attributes`
  - `bin`
  - `calc_bin`
  - `chrom`
  - `dialect`
  - `end`
  - `extra`
  - `featuretype`
  - `file_order`
  - `frame`
  - `id`
  - `keep_order`
  - `score`
  - `seqid`
  - `sequence`
  - `sort_attribute_values`
  - `source`
  - `start`
  - `stop`
  - `strand`

  By trying very hard to move away from databases we ended having database as the best solution

[6] https://gist.github.com/slowkow/8101481
[7] https://pypi.python.org/pypi/bcbio-gff/0.6.2
[8] http://pythonhosted.org/gffutils/

Chapter 6

All about RNAseq

6.1 General

- Functional annotation of the mouse (FANTOM)
- Encyclopedia of DNA elements (ENCODE)
- Single loci can lead to to six different mRNA isoforms, on average, instead of single transcript

6.2 Technology

What is it for?

- what is expressed (exon locatios /strand) ?
- how much is expressed (hits per transcripts) ?

old way to do DE - microarray

- do need as much starting material for RNAseq

Noise:

- biological
- poisson noise from samplinig (have level of uncertanty at low level, low counts)
- technical
- Effect size (do the genes go up by alot or not - high signal)
- amount of variation

get better depth using poly-A enriched, because rRNA depletion will leave lots of other RNA around therefore diluting mRNA by some percent

mean fragment size ~ 300 bp

measure of uncertanty of the base call

6.3 Differential expression (DE)

- library size is total number of mapped reads. Library size might vary for individual samples.
- Total number of reads per gene proportional to:
  - gene expression level (to get cpm (count per million) divide each read count by library size multiplied by million). This is enable you to compare different library sized samples
  - transcript length

  - sequencing depth of the library # Ideas
- have a url with multiqc report ready for browsing
- explain reference file (FASTA) and annotation files (GTF)

Chapter 7

Lesson plan

Perhaps it'll be good idea to introduce several different plain text files

- `txt`
- `csv/tsv`
- `gff/gff3/gtf`
- `fastq/fq`
- `fasta/fa/faa/fna`

DO NOT unzip your fastq file and store them as plain text ! DO NOT mess with your raw fastq files!

Chapter 8

Meeting notes

This is overall them of the day

How to instruct them to be good RNAseq user?

have a url with multiqc page ready?

This is

explain reference file (FASTA) and annotation files (GTF)

This if you can set up ftp link so that users can pont and click through directory structure to interact with the sikRun output.

Mentione salmon

Good to mention in parsing - salmon if you are using salmon you not going to see DNA contamination. so something to be aware.

Also mentione that this is for model organism that are well annotation/assembled.

Use Chiara's data to study chromosome 21. From multiqc report chromosome 21 has a lot more reads. It looks like ribosomal contamination even though library was polyA pooled.

Take all reads from chr21 and do feature counts and look where reads go. Suspecting that reads go to ribosomal genes. If so this would be great case study for the workshop.

https://www.ncbi.nlm.nih.gov/pubmed/29788454

genotify

Reproducible research using the same reference files

TCGA for example is using a single reference for the whole cohort

If you gonna do second study e.g chipSeq you gotta have it on the same reference as your RNAseq

there is no such thing as "the reference genome" there are different builds, it is work in progress