Introduction to RNAseq

Version 1

Contents

Introduction to RNAseq **2**

1 Genomics **4**
1.1 Sequence . . . 4
1.2 Read sequence . . . 5
1.3 Contig sequence . . . 6
1.4 Summary . . . 6

2 Bioinformatics file formats **8**
2.1 Introduction . . . 8
2.2 What is a file format? . . . 9
2.3 Annotation files . . . 9
2.4 Glossary . . . 10

3 Reference sequence **11**
3.1 FASTA . . . 11
3.2 Exercise . . . 12
3.3 Rule of thumb . . . 13

4 Raw data **14**
4.1 FASTQ . . . 14
4.2 Unix tooling . . . 17

5 RNAseq bioinformatics workflow **18**
5.1 Alignment . . . 18
5.2 Read counting . . . 19
5.3 Summary . . . 22

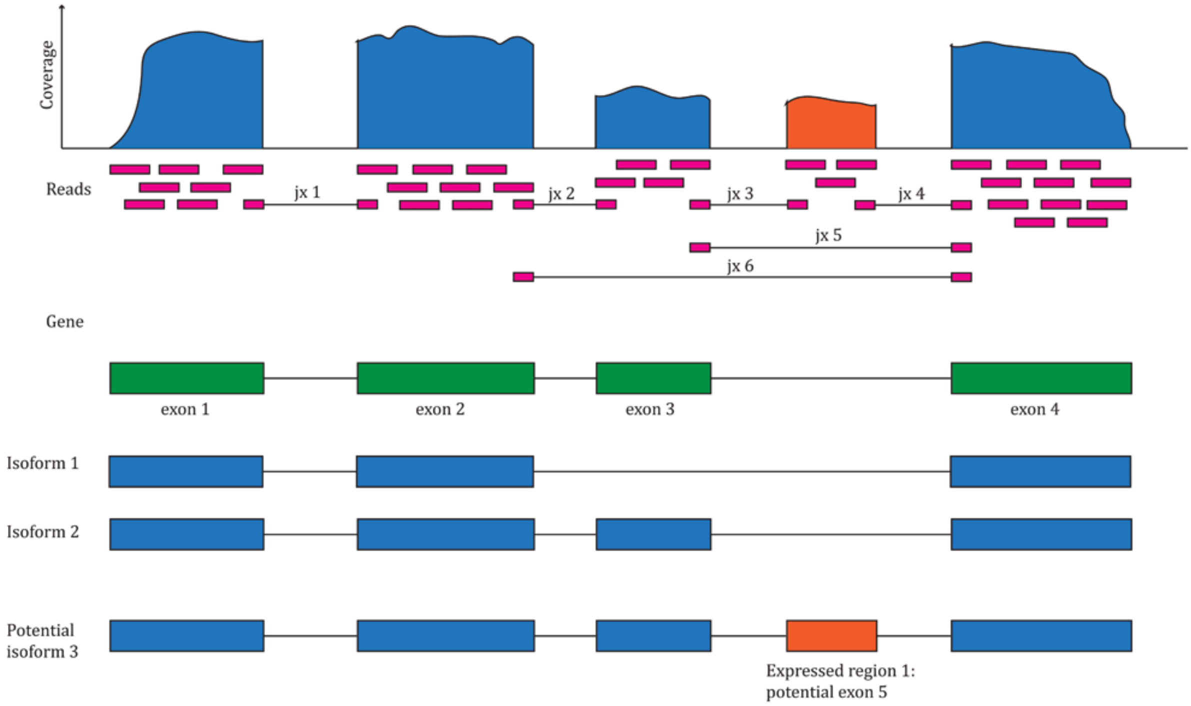
6 QC **24**
6.1 Number of mapped reads . . . 24
6.2 Number of assigned reads . . . 24
6.3 Intragenic vs Intergenic region . . . 24
6.4 Reads mapping bias . . . 24
6.5 Insert size distribution . . . 25

7 All about RNAseq **26**
7.1 General . . . 26
7.2 Technology . . . 26
7.3 Differential expression (DE) . . . 26

8 Lesson plan **28**

9 Meeting notes **29**

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/

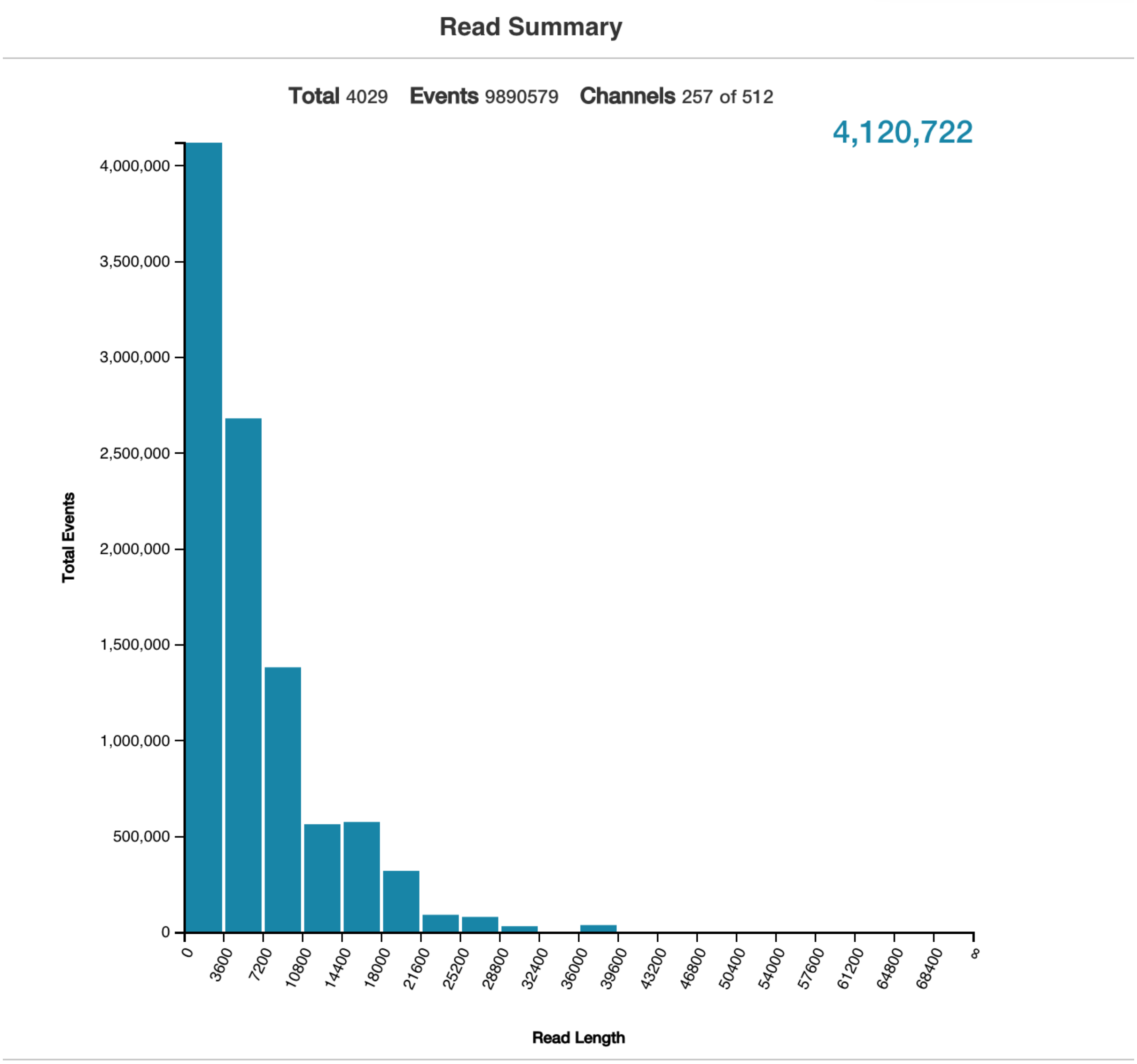


Figure 1.1: https://bioinformatics.stackexchange.com/questions/3623/how-many-reads-has-my-sequencing-run-produced-on-minion

Chapter 2

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.

2.1 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

2.2 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.

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
```

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
```

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.

2.3 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`

2.4 Glossary

- interface: extract or insert relevant information

Chapter 3

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.

3.1 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.

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]

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.

3.2 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

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

- 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
```

```
>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.

3.3 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 4

Raw data

4.1 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 preparation and sequencing.

Your reads come in the form of FASTQ files. By far the most dominant way of organising your FASTQ files one per sample. For example if your experiment is a straightforward `TP53` knock-out in mice. You'll need bear minimum 4 mice, best to get 6, this is 3 control and 3 tp53 knock-outs. This is way you have three replicates to account for mouse to mouse variability and have a better chance of estimating actual difference due to knock-out.

You should receive these files from sequencing facility:

```
cntr_rep1_R1.fastq.gz
cntr_rep2_R1.fastq.gz
cntr_rep3_R1.fastq.gz

tp53_ko_rep1_R1.fastq.gz
tp53_ko_rep2_R1.fastq.gz
tp53_ko_rep3_R1.fastq.gz
```

The naming of the files is completely up to you. Sequencing facility doesn't care about the names, as long as you give them 6 different microtubes with some unique naming on the lid, but remember that giving clear names simplifies the matter. Bioinformatician might be able to infer from FASTQ file names what the experiment was.

Question1: How many FASTQ files do you expect in a single-end experiment with 6 samples?

Remember that if you decided to sequence from both ends of the reads i.e paired-end sequencing, you'll give FASTQ file for each sample for each end. You should expect double the number of samples. It is important to have exactly double the number of files, because downstream analysis will rely on those files and will break if not supplied.

```
cntr_rep1_R1.fastq.gz
cntr_rep1_R2.fastq.gz

cntr_rep2_R1.fastq.gz
cntr_rep2_R2.fastq.gz

cntr_rep3_R1.fastq.gz
cntr_rep3_R2.fastq.gz
```

```
tp53_ko_rep1_R1.fastq.gz
tp53_ko_rep1_R2.fastq.gz

tp53_ko_rep2_R1.fastq.gz
tp53_ko_rep2_R2.fastq.gz

tp53_ko_rep3_R1.fastq.gz
tp53_ko_rep3_R2.fastq.gz
```

Question2: How many FASTQ files do you expect in a paired-end experiment with 6 samples?

One other thing that you should be aware is that sometimes to get extra deep sequencing your individual samples can be split across different lanes on the sequencing instrument (mainly talking about Illumina instruments here). This might seem confusing but you will need to multiple your samples by number of lanes. If you have single-end data of 6 sample that were split across 2 lanes then you are going to end up with 12 FASTQ files.

```
cntr_rep1_R1_L001.fastq.gz
cntr_rep1_R1_L002.fastq.gz

cntr_rep2_R1_L001.fastq.gz
cntr_rep2_R1_L002.fastq.gz

cntr_rep3_R1_L001.fastq.gz
cntr_rep3_R1_L002.fastq.gz

tp53_ko_rep1_R1_L001.fastq.gz
tp53_ko_rep1_R1_L002.fastq.gz

tp53_ko_rep2_R1_L001.fastq.gz
tp53_ko_rep2_R1_L002.fastq.gz

tp53_ko_rep3_R1_L001.fastq.gz
tp53_ko_rep3_R1_L002.fastq.gz
```

If your samples have been split across 3 lanes you are going to have 18 FASTQ files.

```
cntr_rep1_R1_L001.fastq.gz       -
cntr_rep1_R1_L002.fastq.gz       | cntr_rep1
cntr_rep1_R1_L003.fastq.gz       -

cntr_rep2_R1_L001.fastq.gz       -
cntr_rep2_R1_L002.fastq.gz       | cntr_rep2
cntr_rep2_R1_L003.fastq.gz       -

cntr_rep3_R1_L001.fastq.gz       -
cntr_rep3_R1_L002.fastq.gz       | cntr_rep2
cntr_rep2_R1_L003.fastq.gz       -

tp53_ko_rep1_R1_L001.fastq.gz    -
tp53_ko_rep1_R1_L002.fastq.gz    | tp53_ko_rep1
tp53_ko_rep1_R1_L003.fastq.gz    -

tp53_ko_rep2_R1_L001.fastq.gz    -
tp53_ko_rep2_R1_L002.fastq.gz    | tp53_ko_rep2
tp53_ko_rep2_R1_L003.fastq.gz    -

tp53_ko_rep3_R1_L001.fastq.gz    -
tp53_ko_rep3_R1_L002.fastq.gz    | tp53_ko_rep3
tp53_ko_rep3_R1_L003.fastq.gz    -
```

Note that when samples are split across lanes they become technical replicates and they are typically merged into single file. Also note that if you instead had paired end data you will have to further multiple by 2 to get total number of FASTQ file.

```
cntr_rep1_R1_L001.fastq.gz      -
cntr_rep1_R2_L001.fastq.gz      |
                                |
cntr_rep1_R1_L002.fastq.gz      | cntr_rep1
cntr_rep1_R2_L002.fastq.gz      |
                                |
cntr_rep1_R1_L003.fastq.gz      |
cntr_rep1_R2_L003.fastq.gz      -

cntr_rep2_R1_L001.fastq.gz      -
cntr_rep2_R2_L001.fastq.gz      |
                                |
cntr_rep2_R1_L002.fastq.gz      | cntr_rep2
cntr_rep2_R2_L002.fastq.gz      |
                                |
cntr_rep2_R1_L003.fastq.gz      |
cntr_rep2_R2_L003.fastq.gz      -

cntr_rep3_R1_L001.fastq.gz      -
cntr_rep3_R2_L001.fastq.gz      |
                                |
cntr_rep3_R1_L002.fastq.gz      | cntr_rep2
cntr_rep3_R2_L002.fastq.gz      |
                                |
cntr_rep2_R1_L003.fastq.gz      |
cntr_rep2_R2_L003.fastq.gz      -

tp53_ko_rep1_R1_L001.fastq.gz   -
tp53_ko_rep1_R2_L001.fastq.gz   |
                                |
tp53_ko_rep1_R1_L002.fastq.gz   | tp53_ko_rep1
tp53_ko_rep1_R2_L002.fastq.gz   |
                                |
tp53_ko_rep1_R1_L003.fastq.gz   |
tp53_ko_rep1_R2_L003.fastq.gz   -

tp53_ko_rep2_R1_L001.fastq.gz   -
tp53_ko_rep2_R2_L001.fastq.gz   |
                                |
tp53_ko_rep2_R1_L002.fastq.gz   | tp53_ko_rep2
tp53_ko_rep2_R2_L002.fastq.gz   |
                                |
tp53_ko_rep2_R1_L003.fastq.gz   |
tp53_ko_rep2_R2_L003.fastq.gz   -

tp53_ko_rep3_R1_L001.fastq.gz   -
tp53_ko_rep3_R2_L001.fastq.gz   |
                                |
tp53_ko_rep3_R1_L002.fastq.gz   | tp53_ko_rep3
tp53_ko_rep3_R2_L002.fastq.gz   |
                                |
tp53_ko_rep3_R1_L003.fastq.gz   |
tp53_ko_rep3_R2_L003.fastq.gz   -
```

4.2 Unix tooling

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 extension is, rather what they hold.

Essentially all bioinformatics files are either plain text (a.k.a ASCII) or binary. 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 popular 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 binary, this will requires designated tool for interfacing with the file for example SAM/BAM files have `samtools` that aid interaction with them.

Chapter 5

RNAseq bioinformatics workflow

Hands on bioinformatics starts from FASTQ files. You typically will get them from sequencing provider. However, as mentioned previously, everything you did in the lab before sequencing i.e RNA extraction, storage and handling as well as experimental design in general, feeds into bioinformatics analysis, so do try to record as much of that as possible and be ready to communicate this to a bioinformaticina.

5.1 Alignment

Once you've got your raw data - FASTQ files, the first thing you will do in RNAseq analysis is to map to the reference genome. How to find and select reference genome has been discussed previously in reference sequence section. Here we are starting from an alignment step. Let's do another thought experiment; what information might you need from an aligner? A simpler question is; Once you've found which part of the genome a read belongs to, what information should you record?

- which chromosome each read mapped to
- which coordinate on that chromosome read map to
- some quality metric on how well it mapped, all bases mapped or just some?

This is a bear minimum information you want your aligner to output, but there is actually a lot more information aligners typically output including was the read a multi-mapper, the orientation of read in terms of strand and position first or second, this is in the case of paired-end data. Some aligners like STAR[1] also include information about splice junctions. Also with sample and UMI barcoding there are moves to ask aligners to record that information as well.

Essentially all aligners these days will produce a sequence alignment map or sam file for short. Sam file is once again is a simple plain text file with rows and columns tab separated, with a header section that is unique to a sam file. It is best practice to keep sam files zipped, but instead of conventional `gzip` utility use special for this purpose `samtools` utility and the convention is to call compressed files BAM for binary sam. An example below illustrates relatively extreme case, never the less show a massive advantage in disk space in using bam files instead of sam.

```
-rw-r--r--  1 kirill kirill  41G Aug 30 14:01 cntr_rep1_sorted_mdups.bam
-rw-rw-r--  1 kirill kirill 399G Aug 31 11:06 cntr_rep1_sorted_mdups.sam
```

We always never store sam files on your disk. Always converted to bam files. Bam files are an industry standard, a lot of bioinformatics analysis, certainly true for RNAseq, rotates around bam files. There even some thoughts to keep raw data in bam files instead of FASTQs.

One important thing you need to know about aligners is that broadly speaking there are two types, splice away aligners and splice unaware aligners. This is different from gapped alignment. Essentially any good aligners will do gapped alignment to account for insertions and deletions of few bases. However in the case of RNAseq some gaps can actually be quite large and can be few kilobases, those are the splice

[1] https://github.com/alexdobin/STAR

junctions. Below is a list of a few alingers that are commonly used in our group, but of course there are many more aligners out there.

Don't use these ones for RNAseq analysis, these ones are for DNA mapping, including data from ChipSeq and ATACseq.

Common splice unaware aligners:

- `bowtie2`
- `bwa`

These aligners commonly used for RNAseq analysis. You can use for ChipSeq and ATACseq as well, but best not to, unless you know what you are doing. This is mainly because these aligners will model gaps differently, since large gaps (splice junctions) are expected you might get wrong scoring of insertion and deletions in your Chip and ATAC seqs.

Common splice aware aligners:

- `tophat2`
- `hisat2`
- `STAR`

Let's talk a little more about bam files, before moving on to read counting as a step two in typical RNAseq analysis.

5.1.1 Bam files

Bam files can be sorted in a couple of ways:

- by coordinate (4th column): arrange reads from first base till last base on each chromosome. Remember that chromosomes themselves may not be sorted by name or length
- by name (1st column): arrange reads alphabetically on each chromosome. Remember each read in a FASTQ file has a unique name (yes all 20 million or more reads).

The reason for name sorting is to place R1 and R2 reads next to each other (one after another), this mainly applicable for paired-end sequencing. I can't see any effect sorting by name single-end data, this will make it as random as not sorting at all. Sorting by coordinates is a dominant way of sorting bam files, big advantage of sorting by coordinates is ability to for quick lookup of any region with in the bam. You will need index bam files.

Remember that you should end up with one bam file per sample and typically each bam will be also sorted. You don't need both copies of sorted and unsorted bam files. Both contain the same information. Sorted bam files will result in slightly reduced bam file sizes, this is because sorted bam files will give slightly better compression.

```
cntr_rep1_R1.fastq.gz   cntr_rep1.bam   cntr_rep1_sorted.bam
cntr_rep2_R1.fastq.gz   cntr_rep2.bam   cntr_rep2_sorted.bam
cntr_rep3_R1.fastq.gz   cntr_rep3.bam   cntr_rep3_sorted.bam

tp53_ko_rep1_R1.fastq.gz    tp53_ko_rep1.bam    tp53_ko_rep1_sorted.bam
tp53_ko_rep2_R1.fastq.gz    tp53_ko_rep2.bam    tp53_ko_rep2_sorted.bam
tp53_ko_rep3_R1.fastq.gz    tp53_ko_rep3.bam    tp53_ko_rep3_sorted.bam
```

5.2 Read counting

5.2.1 Introduction

Before leaping into what read counting is let's remind ourselves quickly what an RNAseq data looks like if we visualise it. There are number of different applications that you can use to visualise, but the one we most commonly use is Integrative Genomics Viewer (IGV)[2]

[2] http://software.broadinstitute.org/software/igv/

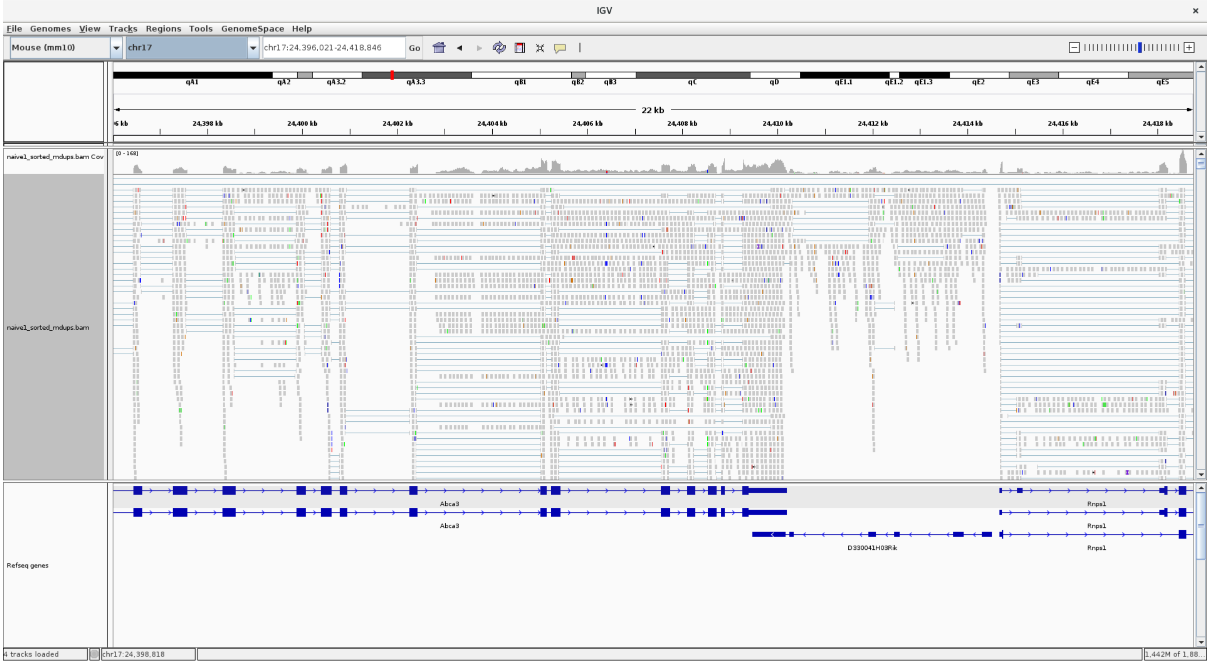
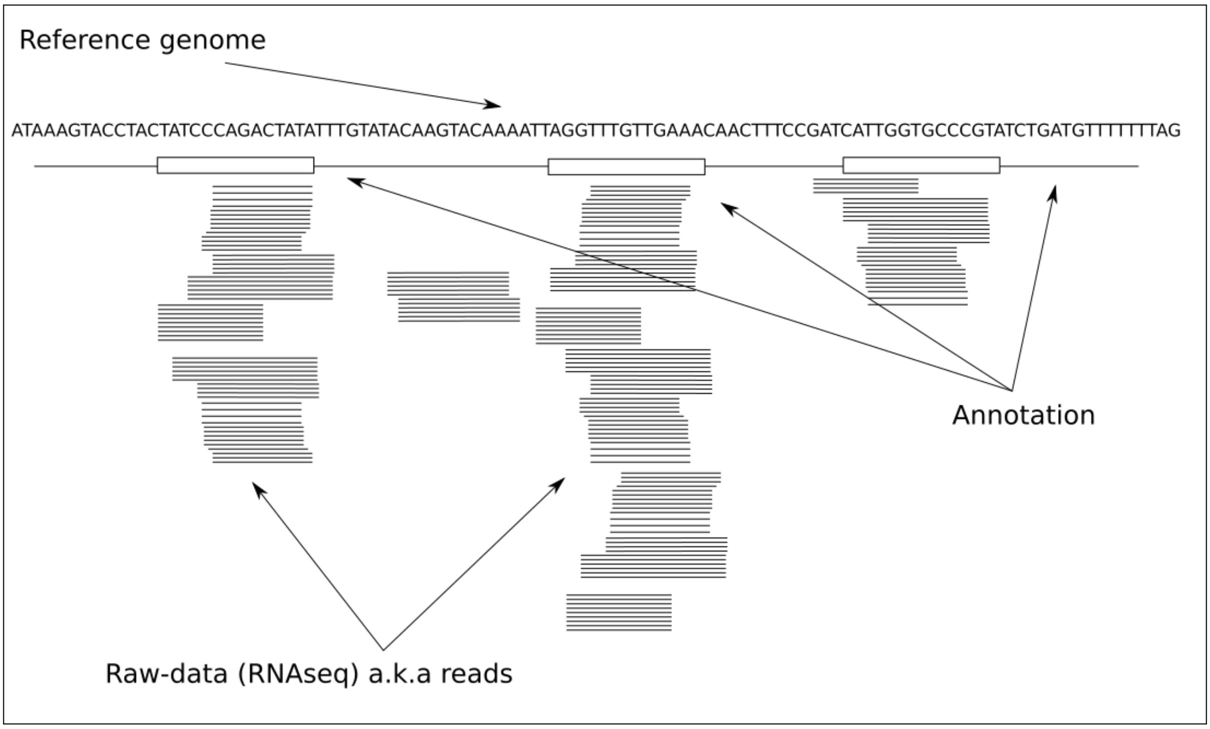


Figure 5.1: IGV browser view of one sample

Typically however there are multiple samples you'd be looking, which I left out from the image for clarity of view.

Here is schematic view of what you typical see in a genome viewer


Reference genome
ATAAAGTACCTACTATCCCAGACTATATTTGTATACAAGTACAAAATTAGGTTTGTTGAAACAACTTTCCGATCATTGGTGCCCGTATCTGATGTTTTTTTAG
Annotation
Raw-data (RNAseq) a.k.a reads

The three things that we always need for visualisation of that kind are:

- reference genome file (FASTA)
- annotation file (GTF/GFF)
- data:

  - bam: these are information reach, but slow to load and navigate
  - bigWig: one example of coverage plot file, just show the coverage, samll and quick
  - vcf: variant information, not part of this course

Note that something like IGV already comes with in built reference and annotation files, but just remember that it's likely comes with different version of the genome. Often time IGV reference files will be good enough for first look and if you start suspecting something you should load your specific reference.

Go through igv screen shoots and explain few different concenpts, like overlapping features

Now that we've recapped what RNAseq data typically looks like lets talk how we approach read counting bioinformatically

5.2.2 Table of counts

Given an alignment map file a.k.a bam file and an annotation file e.g GTF we need to systematically go through every read in the bma file and assign it to a feature in the annotation file. Ultimately we would like to get contiguous table or simply, table of counts, where each row is a gene and columns are our samples. Here is a top few lines from typical table of counts.

```
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
```

Note that this table has a few extra columns with some extra information, those aren't strictly necessary, but do allow to put things into biological picture.

This table would be minimal information you need for differential analysis. Note that `Gene.ID` column has to be unique values.

```
Gene.ID KO3_S3  KO4_S4  WT1_S1  WT2_S2
ENSMUSG00000051951  0   0   0   0
ENSMUSG00000025900  0   0   0   0
ENSMUSG00000109048  0   0   0   0
ENSMUSG00000025902  0   0   0   0
ENSMUSG00000033845  934 1317    888 939
ENSMUSG00000025903  705 848 647 747
ENSMUSG00000104217  0   0   0   0
ENSMUSG00000033813  354 436 368 442
ENSMUSG00000002459  0   0   0   0
```

In RNAseq we typically should see reads only around feature, all reads should remain within the feature coordinates boundary. Reads should pile up on exons with clear boundaries between adjacent exons. Reads can and do map anywhere though and if reads fall outside of "normal" regions it could be an interesting case to follow up. As illustrated before visualisation is very important part of working with RNAseq data.

A few things to know when visualising RNAseq data:

- use the same reference sequence that you mapped against, otherwise you might see more variance between reference genome and the data
- be sure to visualise against the same annotation version and vendor type. This will ensure that your features (boxes) align best with your data. This is more important then picking the same reference files as features do change coordinates from version to version. Therefore if you are using different version of annotation to which you used for read assignment for visualisation you might see too

many reads outside your feature boundaries, leading to false sense that there is some biological effect going on.
- typically RNAseq viewers sub-sample your data for visualisation purposes. This is mainly for speed and performance of the viewer. This is especially for highly expressed genes with deep read coverage. Be aware of that when doing your exploratory analysis. Often times you can turn sub-sampling off. Once again not seeing all of the data may result in misleading observations.

5.2.3 Details

We normally use words like "counting features" or "read assignment to features", all means the same obtain table of counts. It is important to appreciate that "feature" could mean several things, it is some region of DNA that has some value/interest, a protein coding gene is a DNA region that produces a protein. For the purpose of RNAseq experiment we are most likely interested in protein coding features and therefore need to count those.

Similar to aligners there are number of tools out there that do read counting, but the two most popular are:

- htseq-count[3]
- featureCounts[4]

One thing to mention is that STAR aligner, mentioned previously, can actually output count files as well. In essence one can do most of RNAseq processing with STAR aligner.

I'm most familiar with how `featureCounts` works and it is a versatile tool, by default it counts reads that mapped to exons, which are referred as features, and summarises at meta-feature level which by default are gene. The number that ends up in the table of counts is a sum of all of reads from exons in a give gene. Remember that if a read overlaps more then one feature, by default, it is marked as an ambiguous read and isn't counted toward a feature. It is possible to summarise reads at exon level which opens a possibility for differential exon usage analysis from RNAseq, something that isn't covered in this course.

We are going to through more QC's in the next section but here is nice summary image from `htseq-count` docs, to cement all possible read assignment problems.

The final step in RNAseq analysis is to take table of counts and apply statistical modeling to study differential expressions of genes. This will be covered in later sections.

5.3 Summary

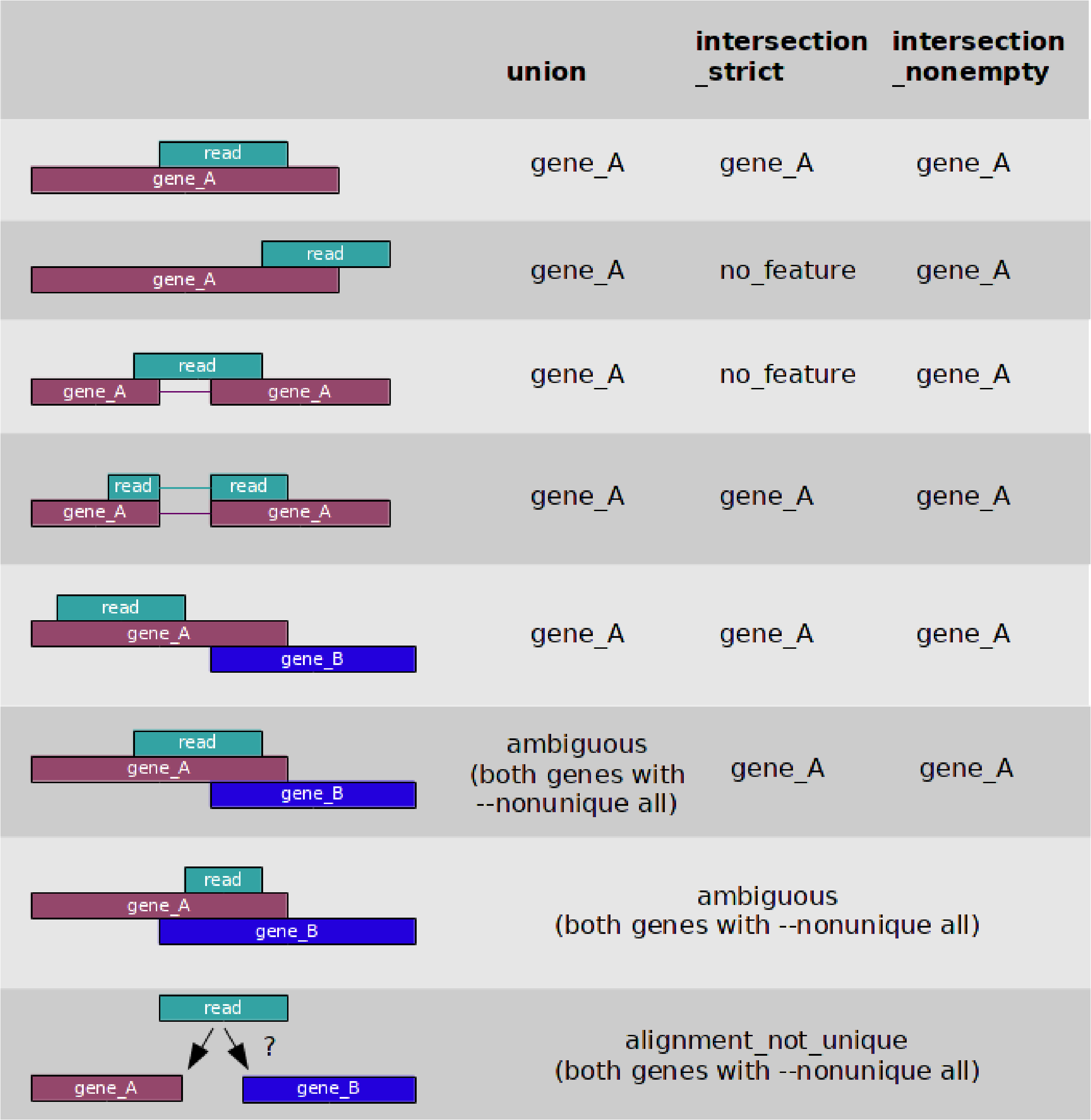
- RNAseq analysis is a relative measure of reads between samples.
- step one in RNAseq analysis is to get bam files
- step two in RNAseq analysis is to get table of counts.
- step three in RNAseq analysis do differential expression

Question: What is the correct order for an alignment based RNAseq workflow?

- differential expression, mapping, read counting
- read counting, mapping, differential expression
- mapping, read counting, differential expression
- differential expression, read counting, mapping

[3] https://htseq.readthedocs.io/en/release_0.10.0/index.html

[4] http://subread.sourceforge.net/



union	intersection_strict	intersection_nonempty
gene_A	gene_A	gene_A
gene_A	no_feature	gene_A
gene_A	no_feature	gene_A
gene_A	gene_A	gene_A
gene_A	gene_A	gene_A
ambiguous (both genes with --nonunique all)	gene_A	gene_A
ambiguous (both genes with --nonunique all)		
alignment_not_unique (both genes with --nonunique all)		

Figure 5.2: https://htseq.readthedocs.io/en/release_0.10.0/count.html

Chapter 6

QC

6.1 Number of mapped reads

Different mapping types:

- Uniquely mapped (255)
- Mapped to multiple location a.k.a multi-mapper
- Mapped to too many loci - discarded

6.2 Number of assigned reads

- Unassigned Unmapped: reads are reported as unmapped in SAM/BAM input. Note that if the '–primary' option of featureCounts program is specified, the read marked as
- primary alignment will be considered for assigning to features.
- Unassigned MappingQuality: mapping quality scores lower than the specified threshold.
- Unassigned Chimera: two reads from the same pair are mapped to different chromosomes or have incorrect orientation.
- Unassigned FragementLength: length of fragment does not satisfy the criteria.
- Unassigned Duplicate: reads marked as duplicate in the FLAG field in SAM/BAM input.
- Unassigned MultiMapping: reads marked as multi-mapping in SAM/BAM input (the 'NH' tag is checked by the program).
- Unassigned Secondary: reads marked as second alignment in the FLAG field in SAM/BAM input.
- Unassigned Nonjunction: reads that do not span exons will not be assigned if the '–countSplitAlignmentsOnly' option is specified.
- Unassigned NoFeatures: not overlapping with any features included in the annotation.
- Unassigned Overlapping Length: no features/meta-features were found to have the minimum required overlap length.
- Unassigned Ambiguity: overlapping with two or more features (feature-level summarization) or meta-features (meta-feature-level) summarization.

6.3 Intragenic vs Intergenic region

6.4 Reads mapping bias

If the RNA wasn't carefully treated and became fragmented then you are more likely to see 3 prime bias for reads mapping

6.5 Insert size distribution

Chapter 7

All about RNAseq

7.1 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

7.2 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

7.3 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 8

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 9

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