DNASeq: Analysis pipeline and file formats
Sumir Panji, Gerrit Boha and Amel Ghouila

Bioinformatics analysis and annotation of variants in NGS data workshop

Cape Town, 4th to 6th April 2016
DNA Sequencing: definitions

• Sequencing DNA is carried out to determine the order of 4 bases (C,G,A,T) within a DNA molecule.

• DNA sequencing is used to detect alterations to a patient’s DNA which may cause a genetic condition.

• Alterations are detected by comparing the sequence from the patient sample to a normal sequence, determined either by sequencing a control sample which is known not to have any alterations or by consulting databases such as the human genome database.

• Genome sequencing – allows detection of SNPs (single nucleotide polymorphisms) and structural variations among individuals: – within a population, – from different populations.
Why sequence DNA?

• The data allow comprehensive comparisons of organisms on a genomic level to find regions of similarity, difference, and functional significance.

• The data allow us to understand Human variation on a molecular level, for example, the genetic differences between tumor and normal tissue.

• This could lead to more specific medical treatments (personalized medicine).
Pre-NGS era

- Sanger sequencing (Frederick Sanger, 1970)
- Shotgun sequencing first used to produce a draft of the human genome (2001)
How does Next Generation Sequencing work?

Source: GATK website
A common pipeline

Library preparation

Massive parallel sequencing

Bioinformatics analysis

DNA samples
Sequencing terminology

Library preparation, each reaction produces a library of DNA fragments

Each NGS machine has a flowcell with 8 independent lanes during a single sequencing run

Source: GATK website
NGS

- **Next-generation sequencing (NGS)** techniques are based on a "sequencing by synthesis" principle, where nucleotides incorporated into a strand of DNA provide a signal.

- **Massively Parallel**: millions of sequencing reactions (1,000sMb/day)

- **Can be multiplexed**: simultaneous sequencing and analysis of samples
DNASeq and NGS

• The NGS technologies have changed the area of genomics:
  – personalized medicine: identify the genetic alterations that can predict how a patient might respond to a specific drug and decide what dose might be most effective;
  – Allows the screening of a great number of genes to identify the most relevant genetic alterations.
Structural Variants

• Structural variants are any rearrangements of the genome relative to a reference. They include:
  ✓ insertions/deletions
  ✓ Inversions
  ✓ Translocations
  ✓ Tandem repeat variations

Many can be detected with paired-end or mate-pair reads.
Other Experimental Methods That Use Sequencing

- **RNA Seq** – Quantitative measurement of gene expression in a tissue by counting the number of RNA fragments sequenced from each gene. Also used for alternative splicing detection.

- **ChIP Seq** – (Chromatin Immunoprecipitation): Identification of protein binding sites on DNA by determining where DNA fragments bound to a specific protein map onto the genome.
Sequencing costs as estimated by NHGRI

Cost to sequence a human genome as estimated by NHGRI
(September 2001 to July 2015)
Sequencing costs

• Sequencing technology has vastly improved in recent years.
• Sequencing the first human genome cost about $1 billion and took 13 years to complete; today it costs about $1,000 and takes just one to two days.
Platforms using NGS technologies

- Next generation sequencing (NGS) is used to describe a number of different modern sequencing technologies used by different platforms:
  - **Popular Platforms**
    - Illumina (HiSeq, MiSeq)
    - Roche 454 sequencing
    - SOLiD sequencing
  - **Newer Platforms (third generation)**
    - Ion torrent
    - PacBio
    - Oxford Nanopore
  - These recent technologies allow us to sequence DNA and RNA much more quickly and cheaply than the previously used Sanger sequencing.
Platform features: read length, cost, etc.

<table>
<thead>
<tr>
<th>Feature</th>
<th>HiSeq2500 - High output</th>
<th>HiSeq2500 – Rapid mode</th>
<th>MiSeq</th>
<th>PacBio RSII</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of reads</td>
<td>150-180M/lane</td>
<td>100-150M/lane</td>
<td>12-15M (v2)</td>
<td>50-80K/SMRT cell</td>
</tr>
<tr>
<td>Read length</td>
<td>2 x 100 bp</td>
<td>2 x 150 bp</td>
<td>2 x 300 bp (v3)</td>
<td>~ 10-20 kb</td>
</tr>
<tr>
<td>Yield per lane (PF data)</td>
<td>up to 35 Gb</td>
<td>up to 45Gb</td>
<td>up to 15 Gb</td>
<td>up to 0.4 Gb</td>
</tr>
<tr>
<td>Instrument Time</td>
<td>~12-14 days</td>
<td>~2 days</td>
<td>~2 days</td>
<td>~2 hours</td>
</tr>
<tr>
<td>Pricing per Gb</td>
<td>$59 (PE100)</td>
<td>$53 (PE150)</td>
<td>$108 (PE300)</td>
<td>$697</td>
</tr>
</tbody>
</table>

Source: UC Davis Genome Center
Challenges of the NGS

- **Large amount of sequence data** generated in a single run: it can reach 600 Gb
- Management and storing
- Analyzing the data and transforming it into meaningful knowledge that could be used by clinicians
Pipeline and formats

1. FastQ reads
2. Cleaning/trimming
3. Assembly
4. Mapping
5. SAM files
6. BAM files
7. Read Quality analysis (FastQC)
8. CNV, read depth (BWA, Bowtie2, etc.)
9. Structural variation (Bedtools, Picard)
10. Variant detection
11. SNP call (GATK)
12. Visualization
13. Annotation
14. VCF files
File formats

There are multiple file formats used at various stages of NGS data processing. We can divide them into two basic types:

- **Text based** (FASTA, FASTQ, SAM, GTF/GFF, BED, VCF, WIG)
- **Binary** (BAM, BCF, SFF(454 sequencer data))

We can view and manipulate text based formats without special tools, but we will need these to access and view binary formats. The text-based format are often compressed to save space making them de facto binary, but still easy to read by eye using standard Unix tools.
FASTQ format

What is a FastQ file?

FASTQ = FASTA + Quality

FastQ format is a text-based format for storing both a biological sequence and its corresponding quality scores.
FastQ format

- Each FastQ file contains hundreds of millions of rows
- Each block of 4 lines, starting with " @" represents a read

**Line 1** begins with a '@' character and is followed by a sequence identifier and an optional description (like a FASTA title line)

**Line 2** is the raw sequence letters

**Line 3** begins with a '+' character and is optionally followed by the same sequence identifier (and any description)

**Line 4** encodes the quality values for the sequence in Line 2, and must contain the same number of symbols as letters in the sequence
FastQ example

A FastQ file containing a single sequence might look like this:

@read_ID
GATTTGGGGTTCAAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAAGTTT
+
!"*(((((***+)))%%++)(%%%).1***-+*"))**55CCF>>>>>>>>CCCCCCCC65

The character '!' represents the lowest quality while '~' is the highest
FastQ format

• Casava FastQ format is the same as regular FastQ except that the data is usually split across multiple files for a single sample.
Quality control of output file (FastQC)

• Before the alignment or mapping of sequence reads to a reference sequence, 3 characteristics of NGS data complicate this task:

• 1- If Read lengths are relatively short (36–250 bp) compared to traditional capillary-based sequencing \(\rightarrow\) less information to use for mapping + decreases likelihood that a read can be mapped to a single, unique location.

• 2- If Reads are of imperfect quality \(=\) reads contain higher rates of sequencing error.

• 3- Sheer volume of data. A single run produces millions of sequencing reads, whose alignment to a large reference sequence requires significant computing power.
Quality Control

FastQC: Quality Control for FastQ files

Command line
Available on Galaxy
Graphic reports

http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
What is FastQC?

• Before analyzing generated sequence to draw biological conclusions, a quality control check should be performed to make sure there is no biases in the data.

• Most sequencers will generate a QC report as part of their analysis pipeline focusing on identifying problems generated by the sequencer.

• FastQC aims to provide a QC report which can spot problems which originate either in the sequencer or in the library material.
Basic statistics

<table>
<thead>
<tr>
<th>Measure</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename</td>
<td>good_sequence_short.txt</td>
</tr>
<tr>
<td>File type</td>
<td>Conventional base calls</td>
</tr>
<tr>
<td>Encoding</td>
<td>Illumina 1.5</td>
</tr>
<tr>
<td>Total Sequences</td>
<td>250000</td>
</tr>
<tr>
<td>Sequences flagged as poor quality</td>
<td>0</td>
</tr>
<tr>
<td>Sequence length</td>
<td>40</td>
</tr>
<tr>
<td>%GC</td>
<td>45</td>
</tr>
</tbody>
</table>
QC reports

Quality scores across all bases (Sanger encoding)

Position in read (bp)
FastQC reports

A summary of the modules which were run, and a quick evaluation of whether the results of the module seem entirely normal (green tick), slightly abnormal (orange triangle) or very unusual (red cross).
Per Base Sequence Quality
Per Base Sequence Quality

• The background of the graph divides the y axis:
  ✓ very good quality calls (green)
  ✓ calls of reasonable quality (orange)
  ✓ calls of poor quality (red)

The quality of calls on most platforms will degrade as the run progresses, so it is common to see base calls falling into the orange area towards the end of a read.
FastQC reporting

• In addition to providing an interactive report, FastQC also has the option to create an HTML version of this report for a more permanent record.
Examples of Quality reports

• Good quality FastQC report:
  http://www.bioinformatics.babraham.ac.uk/projects/fastqc/good_sequence_short_fastqc.html

• Bad quality FastQC report
  http://www.bioinformatics.babraham.ac.uk/projects/fastqc/bad_sequence_fastqc.html
Trimming

• Poor quality in the beginnings and the ends can be cut using trimming tools (remove tags/adapter sequences)
Quality filtering

• Flexible tools for Illumina NGS data:
  ✓ Cutadapt
  ✓ Trimmomatic
  ✓ Seq TK, etc.

**Interest:**
✓ Can align more reads: may increase mapping rates
✓ Improve probably de novo assembly

**Warning:**
✓ Loss of information if stringent trimming
SAM/BAM

- After mapping the FASTQ file to the reference genome you will end up with a **SAM** or **BAM** alignment file

- SAM stands **for** Sequence Alignment/Map **format**

- SAM is rarely helpful and really takes up too much space which is why we use only the BAM in principle

- A BAM file (.bam) is the **binary version** of a SAM file
Alignment

- **Alignment** is the process by which we discover how and where the read sequences are similar to the reference sequence.
- An alignment is a result from this process, specifically: an alignment is a way of "lining up" some or all of the characters in the read with some characters from the reference in a way that reveals how they're similar.

Read:  GACTGGGCGATCTCGACTTTCG

Reference: GACTG--CGATCTCGACATCG

dash symbols represent gaps and vertical bars show where aligned characters match
Mapping Reads – the Problem

• Given 100+ million reads from an experiment, for each:
  1. find the genomic coordinates, chromosome and first base, where it has the best match in a reference genome, either with the forward or reverse strand.
  2. best match means zero or a small number of differences with the reference.
  3. differences include mismatches and indels.
  4. determine if it has multiple matches or none at all.
Where can we get known reference sequence?

UCSC
–http://genome.ucsc.edu

NCBI
Fragment Alignment Example

3 spots align equally well ➔ multi-alignement: the read can match multiple positions in the genome
Identify the exact positions: read length, paired end sequencing
Alignement algorithms

2 different algorithmic approaches for aligning short (20–200-bp) sequencing reads:
✓ Spaced seeds
✓ Burrows-Wheeler transform

Spaced seeds (Maq):
- each position in the reference is cut into equal-sized pieces, called ‘seeds’
- seeds are paired and stored in a lookup table.
- Each read is also cut up according to this scheme
- pairs of seeds are used as keys to look up matching positions in the reference.
Advantages/Disadvantages of Seed Matching

• More types and number of differences allowed between reads and genome
• Very fast
• Many seeds per read tested
• Uses lots of memory to store index (>10GB)
• New index required if different seed structure is used or read length changes.
Burrows-Wheeler transform

• Burrows-Wheeler transform (Bowtie2, BWA)
  - Store a memory-efficient representation of the reference genome.
  - Align Reads character by character from right to left against the transformed string.
  - When all characters in the read have been processed, alignments are represented by any positions within the interval.
  - Faster algorithms mainly due to the memory efficiency of Burrows-Wheeler search.

Details about the algorithm and illustration: https://www.youtube.com/watch?v=4n7NPk5lwbl
Aligning pairs

- A "paired-end" or "mate-pair" read consists of pair of mates, called mate 1 and mate 2
- Depending on the protocol, these might actually be referred to as "paired-end" or "mate-paired." Also, we always refer to the individual sequences making up the pair as "mates."

**Paired inputs**

- Pairs are often stored in a pair of files, one file containing the mate 1 and the other containing the mates 2.
- When aligning pairs with Bowtie 2/BWA MEM, specify the file with the mate 1 and the file with the mate 2
- This causes the tool to take the paired nature of the reads into account when aligning them.
Aligners

• Aligners will try to select the “best” place that a read can match
• Minimize the number of mismatches and InDels
• May consider quality values For paired reads:
  • Consider the alignments of both reads
  • Consider proper orientation
  • Consider insert size
• Consider the chromosomes aligned
SAM/BAM

- A SAM file (.sam) is a tab-delimited text file that contains sequence alignment data
- SAM files can be opened using a text editor or viewed using the UNIX "more" command

- Most alignment programs will supply:
  - a header: describing the format version, sorting order of the reads, genomic sequences to which the reads were mapped
  - an alignment section: contains the information for each sequence about where/how it aligns to the reference genome
The following table gives an overview of the mandatory fields in the SAM format:

<table>
<thead>
<tr>
<th>Col</th>
<th>Field</th>
<th>Type</th>
<th>Regexp/Range</th>
<th>Brief description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>QNAME</td>
<td>String</td>
<td>![?-A-] {1,255}</td>
<td>Query template NAME</td>
</tr>
<tr>
<td>2</td>
<td>FLAG</td>
<td>Int</td>
<td>[0,2^{16}-1]</td>
<td>bitwise FLAG</td>
</tr>
<tr>
<td>3</td>
<td>RNAME</td>
<td>String</td>
<td>/*([-()+-&lt;-&gt;-] [-]*</td>
<td>Reference sequence NAME</td>
</tr>
<tr>
<td>4</td>
<td>POS</td>
<td>Int</td>
<td>[0,2^{31}-1]</td>
<td>1-based leftmost mapping POSition</td>
</tr>
<tr>
<td>5</td>
<td>MAPQ</td>
<td>Int</td>
<td>[0,2^{8}-1]</td>
<td>MAPping Quality</td>
</tr>
<tr>
<td>6</td>
<td>CIGAR</td>
<td>String</td>
<td>/*([-0-9]+[MDNSHPX=]+)</td>
<td>CIGAR string</td>
</tr>
<tr>
<td>7</td>
<td>RNEXT</td>
<td>String</td>
<td>/*([-()+-&lt;-&gt;-] [-]*</td>
<td>Ref. name of the mate/next read</td>
</tr>
<tr>
<td>8</td>
<td>PNEXT</td>
<td>Int</td>
<td>[0,2^{31}-1]</td>
<td>Position of the mate/next read</td>
</tr>
<tr>
<td>9</td>
<td>TLEN</td>
<td>Int</td>
<td>[-2^{31}+1,2^{31}-1]</td>
<td>observed Template LENgth</td>
</tr>
<tr>
<td>10</td>
<td>SEQ</td>
<td>String</td>
<td>/*([-A-Za-z.=.-]+)</td>
<td>segment SEQuence</td>
</tr>
<tr>
<td>11</td>
<td>QUAL</td>
<td>String</td>
<td>[!-~]+</td>
<td>ASCII of Phred-scaled base QUALity+33</td>
</tr>
</tbody>
</table>

(http://samtools.github.io/hts-specs/SAMv1.pdf)

**QNAME**: Query template NAME. Reads/segments having identical QNAME are regarded to come from the same template. A QNAME ‘*’ indicates the information is unavailable.
SAM format

The following table gives an overview of the mandatory fields in the SAM format:

<table>
<thead>
<tr>
<th>Col</th>
<th>Field</th>
<th>Type</th>
<th>Regexp/Range</th>
<th>Brief description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>QNAME</td>
<td>String</td>
<td>![?-?A-?] {1,255}</td>
<td>Query template NAME</td>
</tr>
<tr>
<td>2</td>
<td>FLAG</td>
<td>Int</td>
<td>[0, 2&lt;sup&gt;16&lt;/sup&gt;-1]</td>
<td>bitwise FLAG</td>
</tr>
<tr>
<td>3</td>
<td>RNAME</td>
<td>String</td>
<td>*</td>
<td>![()-+-&lt;&gt;-] ![+-]</td>
</tr>
<tr>
<td>4</td>
<td>POS</td>
<td>Int</td>
<td>[0, 2&lt;sup&gt;31&lt;/sup&gt;-1]</td>
<td>1-based leftmost mapping POSition</td>
</tr>
<tr>
<td>5</td>
<td>MAPQ</td>
<td>Int</td>
<td>[0, 2&lt;sup&gt;8&lt;/sup&gt;-1]</td>
<td>MAPping Quality</td>
</tr>
<tr>
<td>6</td>
<td>CIGAR</td>
<td>String</td>
<td>*</td>
<td>([0-9]+[MIDNSHPX=]+)</td>
</tr>
<tr>
<td>7</td>
<td>RNEXT</td>
<td>String</td>
<td>*</td>
<td>=</td>
</tr>
<tr>
<td>8</td>
<td>PNEXT</td>
<td>Int</td>
<td>[0, 2&lt;sup&gt;31&lt;/sup&gt;-1]</td>
<td>Position of the mate/next read</td>
</tr>
<tr>
<td>9</td>
<td>TLEN</td>
<td>Int</td>
<td>[-2&lt;sup&gt;31&lt;/sup&gt;+1, 2&lt;sup&gt;31&lt;/sup&gt;-1]</td>
<td>observed Template LENgth</td>
</tr>
<tr>
<td>10</td>
<td>SEQ</td>
<td>String</td>
<td>*</td>
<td>![A-Za-z=.]</td>
</tr>
<tr>
<td>11</td>
<td>QUAL</td>
<td>String</td>
<td>![+-]</td>
<td>ASCII of Phred-scaled base QUALity+33</td>
</tr>
</tbody>
</table>

(http://samtools.github.io/hts-specs/SAMv1.pdf)

**FLAG**: FLAG: bitwise FLAG. Each bit is explained in a table.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1</td>
<td>template having multiple segments in sequencing</td>
</tr>
<tr>
<td>0x2</td>
<td>each segment properly aligned according to the aligner</td>
</tr>
<tr>
<td>0x4</td>
<td>segment unmapped</td>
</tr>
</tbody>
</table>
The following table gives an overview of the mandatory fields in the SAM format:

<table>
<thead>
<tr>
<th>Col</th>
<th>Field</th>
<th>Type</th>
<th>Regexp/Range</th>
<th>Brief description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>QNAME</td>
<td>String</td>
<td>![?- A-] {1,255}</td>
<td>Query template NAME</td>
</tr>
<tr>
<td>2</td>
<td>FLAG</td>
<td>Int</td>
<td>[0, 2^{16}-1]</td>
<td>bitwise FLAG</td>
</tr>
<tr>
<td>3</td>
<td>RNAME</td>
<td>String</td>
<td>*</td>
<td>![()-+-&lt;&gt;-] ![^-]*</td>
</tr>
<tr>
<td>4</td>
<td>POS</td>
<td>Int</td>
<td>[0, 2^{31}-1]</td>
<td>1-based leftmost mapping POSition</td>
</tr>
<tr>
<td>5</td>
<td>MAPQ</td>
<td>Int</td>
<td>[0, 2^{8}-1]</td>
<td>MAPping Quality</td>
</tr>
<tr>
<td>6</td>
<td>CIGAR</td>
<td>String</td>
<td>*</td>
<td>([0-9]+ [MIDNSHPX=]+)</td>
</tr>
<tr>
<td>7</td>
<td>RNEXT</td>
<td>String</td>
<td>*</td>
<td>= ![()-+-&lt;&gt;-] ![^-]*</td>
</tr>
<tr>
<td>8</td>
<td>PNEXT</td>
<td>Int</td>
<td>[0, 2^{31}-1]</td>
<td>Position of the mate/next read</td>
</tr>
<tr>
<td>9</td>
<td>TLEN</td>
<td>Int</td>
<td>[-2^{31}+1, 2^{31}-1]</td>
<td>observed Template LENgth</td>
</tr>
<tr>
<td>10</td>
<td>SEQ</td>
<td>String</td>
<td>*</td>
<td>[A-Za-z=.]+</td>
</tr>
<tr>
<td>11</td>
<td>QUAL</td>
<td>String</td>
<td>![^-]+</td>
<td>ASCII of Phred-scaled base QUALity+33</td>
</tr>
</tbody>
</table>

(http://samtools.github.io/hts-specs/SAMv1.pdf)

The MAPQ value can be used to figure out how unique an alignment is in the genome (large number, >10 indicate it's likely the alignment is unique).
What other information does the SAM contain?

• The CIGAR string is a sequence of numbers and letters representing the associated information on bases alignment used to indicate things like which bases align (either a match/mismatch) with the reference, are deleted from the reference, and if there are insertions that are not in the reference.

More information about these formats available here:
http://samtools.sourceforge.net
What other information does the SAM contain?

The following table gives an overview of the mandatory fields in the SAM format:

<table>
<thead>
<tr>
<th>Col</th>
<th>Field</th>
<th>Type</th>
<th>Regexp/Range</th>
<th>Brief description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>QNAME</td>
<td>String</td>
<td>![?-A-]{1,255}</td>
<td>Query template NAME</td>
</tr>
<tr>
<td>2</td>
<td>FLAG</td>
<td>Int</td>
<td>[0,2^{16}-1]</td>
<td>bitwise FLAG</td>
</tr>
<tr>
<td>3</td>
<td>RNAME</td>
<td>String</td>
<td>![-()]+-&lt;- -+-] ![-]*</td>
<td>Reference sequence NAME</td>
</tr>
<tr>
<td>4</td>
<td>POS</td>
<td>Int</td>
<td>[0,2^{31}-1]</td>
<td>1-based leftmost mapping POSition</td>
</tr>
<tr>
<td>5</td>
<td>MAPQ</td>
<td>Int</td>
<td>[0,2^{8}-1]</td>
<td>MAPping Quality</td>
</tr>
<tr>
<td>6</td>
<td>CIGAR</td>
<td>String</td>
<td>![*</td>
<td>([0-9]+)[(MIDNSHPX=)]+</td>
</tr>
<tr>
<td>7</td>
<td>RNEXT</td>
<td>String</td>
<td>![*</td>
<td>([!-()]+&lt;--] ![-]*</td>
</tr>
<tr>
<td>8</td>
<td>PNEXT</td>
<td>Int</td>
<td>[0,2^{31}-1]</td>
<td>Position of the mate/next read</td>
</tr>
<tr>
<td>9</td>
<td>TLEN</td>
<td>Int</td>
<td>![2^{31}+1,2^{31}-1]</td>
<td>observed Template LENgth</td>
</tr>
<tr>
<td>10</td>
<td>SEQ</td>
<td>String</td>
<td>![A-Za-z=.] +</td>
<td>segment SEQUENCE</td>
</tr>
<tr>
<td>11</td>
<td>QUAL</td>
<td>String</td>
<td>![-]*</td>
<td>ASCII of Phred-scaled base QUALity+33</td>
</tr>
</tbody>
</table>

(http://samtools.github.io/hts-specs/SAMv1.pdf)

Name of mate (mate pair information for paired-end sequencing, often "=")

Position of mate (mate pair information)

Obviously, the chromosome and position are important. The CIGAR string is also important to know where insertions (i.e. introns) might exist in your read.
Decoding SAM flags

• Explain flag tool: https://broadinstitute.github.io/picard/explain-flags.html
SAM tools

• **samtools** is a free software package for manipulating SAM/BAM files

• samtools is one of the most used tools since BAM files are often the input files needed for many different analysis programs

• **samtools view** test.sam > test.bam

• Different samtools options available: http://www.htslib.org/doc/samtools.html
SAM tools: get statistics from the alignment

- `samtools flagstat` on an aligned bam
- Does a full pass through the input file to calculate and print statistics such as:
  - % reads mapped
  - % unmapped reads
  - % reads properly paired
  - Other information
Creating a BAM index

Many tools require a BAM Index file to more efficiently access reads in a BAM file.

To create a BAM index:

- You must first sort the BAM file to create a sorted.bam
- Run samtools index with the sorted.bam as input
- This will create a file named sorted.bam.bai which contains the index
Indexes

Some programs require that for faster access we need a companion file (often called index) for the different formats

FASTA (.fa & .fai)
BAM and BAI formats (suffixes .bam & .bai)
VCF (.vcf & .vcf.idx)
Bed format

- The bed format is used to represent genomic features and annotations
- Supports up to 12 columns
- 3 required fields in a bed format (for the UCSC, Galaxy and bedtools)
  - Chrom (id)
  - ChromStart the starting position of the feature in the chromosome; the first base in a chromosome is numbered 0
  - Chromend position of the feature in the chromosome
Coverage histogram

- `coveragebed --hist` file1 file2
- Report a histogram of coverage for each feature in file1 as well as a summary histogram for _all_ features in file1.
- Contains:
  - 1) depth
  - 2) # bases at depth
  - 3) size of file1
  - Etc.
BedGraph format

- The BedGraph format allows display of continuous-valued data in track format
- This display type is useful for probability scores and transcriptome data
- Recognized Extension: .bedgraph

More information: http://bedtools.readthedocs.org/en/latest/content/tools/coverage.html
Mapping

Keep in mind that alignment is not the end of the story!

- Set of **aligned reads** and **non aligned reads**
- Several problems after mapping: new annotations, detection of variations, etc.
VCF files

• **Variant Call Format** files: a text file storing gene sequences variations
• Extension based on the GFF format. By using the variant call format only the variations need to be stored along with a reference genome.
• VCF contains:
  ✓ header lines starting with # describing the rest of the file
  ✓ Body: 8 separated tab columns containing the chromosome, position, the REF base, the ALTERNATIVE, the QUAL score and more info
Illumina Vs PacBio

• While Illumina specializes in data quantity for full-length genomes, the PacBio allows zooming into specific regions of a sequence with exceptional accuracy.

• PacBio is the only technology available today that provides transcript read lengths from a single molecule (without PCR amplification). This allows detection of gene fusions and rearrangement events, and read-through of genomic regions with low complexity such as long repeats and regions of high GC content.

• The two systems are often used together to provide a full range of capabilities

• PacBio is an excellent option for providing scaffolds for de novo assembly.