How to map millions of short DNA reads produced by Next-Gen Sequencing instruments onto a reference genome

Stratos Efstathiadis
stratos@nyu.edu

Slides are from Cole Trapneli, Steven Salzberg, Ben Langmead, Thomas Keane, Dennis Wall
Previous Generation: Sanger Sequencing

*Chain-Termination Sequencing*: Introduced by Fred Sanger (1975).

- Large-size DNA reads (several hundred bases per read)
- Low throughput, high cost, labor intensive.
Advancement of Sequencing Technologies

**454 Life Sciences/Roche**: “The Game Changer” (Pyrosequencing, 2005)
   From 450,000bp/day “Sanger-based” sequencing => 100,000,000bp/day

**Solexa/Illumina**: “Ups the Game” (Sequencing by Synthesis, 2006)
   60 million short reads (36bp) => 4 billion bases of DNA per run (3 days) using

Next Generation Sequencing

Massively Parallel
High Throughput
Low cost per base

**February 2001**:  
Ten-year international effort produced $22.5 \times 10^9$ bases of DNA sequence, costing $3B

**February 2011**:  
One NGS instrument produces $60 \times 10^9$ bases per day, costing several thousand dollars
Sequencing By Synthesis

DNA (<1 mg) → Sample preparation → Cluster growth → Sequencing

Courtesy of Illumina Inc.

High Throughput Sequencing → High Data Throughput → Dramatic increase in data storage and computing requirements
Dramatic Increase in Data Storage and Computing Requirements

Image Acquisition
Very Precise, High Resolution imaging: millions of images (.tiff) ➞ Terabytes of data

Gigabytes of Text Output
Millions of DNA reads (Billions of Nucleotides and associated quality scores)
FASTQ Files

- The de-facto file format for sharing DNA sequence read data
- 4 Lines per read
- Sequence line and a per-base Phred quality score line per read
- FASTQ Files are Text files
- There is No file Header
An Introduction to Phred Quality Score

\[ \varepsilon = 10^{-\frac{Q_{\text{Phred}}}{10}} \]

\( Q_{\text{Phred}} = -10 \cdot \log_{10}(\varepsilon) \)

\( \varepsilon \) is the **Error Probability**: The probability that a base call is wrong.

<table>
<thead>
<tr>
<th>Q</th>
<th>( \varepsilon )</th>
<th>Probability the base call in wrong (confidence)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0.0001</td>
<td>0.01% (99.99%)</td>
</tr>
<tr>
<td>30</td>
<td>0.001</td>
<td>0.1% (99.9%)</td>
</tr>
<tr>
<td>20</td>
<td>0.01</td>
<td>1% (99%)</td>
</tr>
<tr>
<td>10</td>
<td>0.1</td>
<td>10% (90%)</td>
</tr>
</tbody>
</table>

- Phred Quality Score encoding in FASTQ/SAM files: **ASCII Character** \( = Q + 33 \)
  - FASTQ Files: \( Q \) represents **Base Call Quality**: Probability the base call is wrong.
  - SAM Files: \( Q \) represents **Mapping Quality**: Probability the mapping position of the read is incorrect.

\[ \text{http://en.wikipedia.org/wiki/FASTQ_format} \]

\[ \text{$perl -e 'print chr(33);'} \]
Exercise: Examining a FASTQ File

-bash-3.2$ cd $SCRATCH
-bash-3.2$ mkdir project1
-bash-3.2$ cd project1
-bash-3.2$ cp /share/apps/examples/bio-workflow/ChIPseq_chr19.fastq .

-bash-3.2$ head ChIPseq_chr19.fastq
-bash-3.2$ tail ChIPseq_chr19.fastq
-bash-3.2$ more ChIPseq_chr19.fastq
-bash-3.2$ wc -l ChIPseq_chr19.fastq

1082524 ChIPseq_chr19.fastq
Exercise: Examining a FASTQ File with **fastqc**

```
project1]$ qsub -l

project1]$ module load fastqc/0.10.1

project1]$ fastqc ChIPseq_chr19.fastq
Started analysis of ChIPseq_chr19.fastq
Approx 5% complete for ChIPseq_chr19.fastq
Approx 10% complete for ChIPseq_chr19.fastq
Approx 15% complete for ChIPseq_chr19.fastq

...........

project1]$ ls -l
```

Open the file **ChIPseq_chr19_fastqc/fastqc_report.html** with the web browser.
Short Read Alignment

• Given a reference and a set of reads, report at least one “good” local alignment for each read if one exists
  – Approximate answer to: where in genome did the read originate?

• What is “good”? For now, we concentrate on:
  – Fewer mismatches is better
  – Failing to align a low-quality base is better than failing to align a high-quality base
  – Match **Uniqueness** of the alignment
Finding the alignments is typically the performance bottleneck
• New alignment algorithms must address the requirements and characteristics of NGS reads

  – Millions of reads per run (30x of genome coverage)
  – Short Reads (as short as 36bp)
  – Different types of reads (single-end, paired-end, mate-pair, etc.)
  – Base-calling quality factors
  – Sequencing errors ( ~ 1%)
  – Repetitive regions
  – Sequencing organism vs. reference genome
  – Must adjust to evolving sequencing technologies and data formats
Indexing

- Genomes and reads are too large for direct approaches like dynamic programming
- **Indexing** is required

![Suffix tree](image1)
![Suffix array](image2)
![Seed hash tables](image3)

- Choice of index is key to performance
Is the process reversible?
NGS Read Alignment
Burrows Wheeler Transformation (BWT)

• Invented by David Wheeler in 1983 (Bell Labs). Published in 1994.
  “A Block Sorting Lossless Data Compression Algorithm”

• Originally developed for compressing large files (bzip2, etc.)

• Lossless, Fully Reversible

• Alignment Tools based on BWT: bowtie, BWA, SOAP2, etc.

• Approach:
  – Align reads on the transformed reference genome, using an efficient index (FM index)
  – Solve the simple problem first (align one character) and then build on that solution to solve a slightly harder problem (two characters) etc.

• Results in great speed and efficiency gains (a few GigaByte of RAM for the entire H. Genome). Other approaches require tens of GigaBytes of memory and are much slower.
Burrows Wheeler Transformation

Text = ctgaaacctggt $ 

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>0</th>
</tr>
</thead>
</table>

- Introduce $ at the end and construct all cyclic permutations of Text

  - c t g a a a a c t g g t $
  - t g a a a a c t g g t $ c
g a a a c t g g t $ c t
a a a c t g g t $ c t g
a a c t g g t $ c t g a a
a c t g g t $ c t g a a
b t g g t $ c t g a a a a
t g g t $ c t g a a a c
g g t $ c t g a a a c t
g t $ c t g a a a c t g
t $ c t g a a a c t g
g t $ c t g a a a c t g
t $ c t g a a a c t g
$ c t g a a a a c t g g t
**Burrows Wheeler Transformation**

- Sort rows alphabetically, keeping of which row went where

<table>
<thead>
<tr>
<th></th>
<th>$</th>
<th>c</th>
<th>t</th>
<th>g</th>
<th>a</th>
<th>a</th>
<th>a</th>
<th>a</th>
<th>c</th>
<th>t</th>
<th>g</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>t</td>
<td>g</td>
<td>g</td>
<td>t</td>
<td>$</td>
<td>c</td>
<td>t</td>
<td>g</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>t</td>
<td>g</td>
<td>g</td>
<td>t</td>
<td>$</td>
<td>c</td>
<td>t</td>
<td>g</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>c</td>
<td>t</td>
<td>g</td>
<td>g</td>
<td>t</td>
<td>$</td>
<td>c</td>
<td>t</td>
<td>g</td>
<td>a</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>t</td>
<td>g</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>t</td>
<td>g</td>
<td>g</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>t</td>
<td>g</td>
<td>g</td>
<td>t</td>
<td>$</td>
<td>c</td>
<td>t</td>
<td>g</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>t</td>
<td>g</td>
<td>g</td>
<td>t</td>
<td>$</td>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>t</td>
<td>$</td>
<td>c</td>
<td>t</td>
<td>g</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>t</td>
<td>$</td>
<td>c</td>
<td>t</td>
<td>g</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>t</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>$</td>
<td>c</td>
<td>t</td>
<td>g</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>t</td>
<td>g</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>g</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>t</td>
<td>g</td>
<td>g</td>
<td>t</td>
<td>$</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>g</td>
<td>g</td>
<td>t</td>
<td>$</td>
<td>c</td>
<td>t</td>
<td>g</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td></td>
</tr>
</tbody>
</table>

**Burrows Wheeler Matrix**

\[
\begin{align*}
\text{BWT (Text)} & = t \ g \ a \ a \ a \ $ \ a \ t \ t \ g \ g \ c \ c \\
\end{align*}
\]

| 11 | 3 | 4 | 5 | 0 | 6 | 2 | 8 | 9 | 10 | 1 | 7 |
The Transformation is **reversible** through **LF Mapping**:  
*The i-th occurrence of a character in the Last column is the same Text occurrence as the i-th occurrence in the First column*
Apply the LM mapping repeatedly to re-create (un-permute) the original Text from BWT(T)
Exercise:

• Find the BWT of $T=tctggt$
• Find the Suffix Array, $S[i]$
• Reproduce the original Text using LF mapping

$$
\begin{array}{cccccccc}
t & c & t & t & g & g & t & $
\end{array}
$$

1 2 3 4 5 6 0

• $BWT(T) = tttgg\$c$
• $S[i] = 6 1 3 4 5 0 2$

• http://allisons.org/ll/AlgDS/Strings/BWT/
Exact Matching with FM Index

• To match Q in T using BWT(T), repeatedly apply rule:

  \[ \text{top} = LF(\text{top}, \text{qc}); \text{bot} = LF(\text{bot}, \text{qc}) \]

  Where \( \text{qc} \) is the next character in Q (right-to-left) and \( LF(i, \text{qc}) \) maps row \( i \) to the row whose first character corresponds to \( i \)'s last character as if it were \( \text{qc} \)
Exact Matching with FM Index

- In progressive rounds, **top** & **bot** delimit the range of rows beginning with progressively longer suffixes of Q
FM Index is Small

- Entire FM Index on DNA reference consists of:
  - BWT (same size as T)
  - Checkpoints (~15% size of T)
  - SA sample (~50% size of T)

- Total: ~1.65x the size of T

Assuming 2-bit-per-base encoding and no compression, as in Bowtie
Assuming a 16-byte checkpoint every 448 characters, as in Bowtie
Assuming Bowtie defaults for suffix-array sampling rate, etc
Exercise

For $\text{BWT}(T) = \begin{array}{cccccccccccc}
    & t & g & a & a & $ & a & t & t & g & g & c & c \\
   11 & 3 & 4 & 5 & 0 & 6 & 2 & 8 & 9 & 10 & 1 & 7
\end{array}$

Align $Q = \text{ctgg}$
Backtracking

• Backtracking attempt for $Q = \text{“agc”}$, $T = \text{“acaacg”}$:

```
$ a c a a c g$
$ a a c g $ a c$
$ a c a a c g$
$ a c g $ a c a$
$ c a a c g $ a$
$ c g $ a c a a$
$ g $ a c a a c$
```

```
$ a c a a c g$
$ a a c g $ a c$
$ a c a a c g$
$ a c g $ a c a$
$ c a a c g $ a$
$ c g $ a c a a$
$ g $ a c a a c$
```

Substitution

```
$ a c a a c g$
$ a a c g $ a c$
$ a c a a c g$
$ a c g $ a c a$
$ c a a c g $ a$
$ c g $ a c a a$
$ g $ a c a a c$
```

Found this alignment:

```
acaacg
    |
    agc
```
Burrows-Wheeler

Reference genome (> 3 gigabases)    Short read

Chr1  ACTCCCGTACTCTAAT
Chr2  
Chr3  
Chr4  

Concatenate into single string

Burrows-Wheeler transform and indexing

Bowtie index (~2 gigabytes)

Look up ‘suffixes’ of read

Hits identify positions in genome where read is found

Convert each hit back to genome location

Alignment to user
Ready-to-use References and Annotations: *iGenomes*

- A collection of reference genomes and annotation files for commonly analyzed organisms.
  - [http://support.illumina.com/sequencing/sequencing_software/igenome.ilmn](http://support.illumina.com/sequencing/sequencing_software/igenome.ilmn)

Exercise:

```bash
[efstae01@]$ module load iгеномes
[efstae01@]$ echo $IGENOMES_ROOT
[efstae01@]$ ls -l $IGENOMES_ROOT/Mus_musculus/UCSC/mm10/Sequence/
```

$IGENOMES/[organism]/[source]/[build]/Sequence/BWAIndex/genome.fa
- [organism] organism of interest (ex. Mus_musculus)
- [source] source of the sequence (ex. UCSC)
- [build] genome draft (ex. mm10)
The Reference and the Reference Index files

- The Reference Genome file is a text file containing the genome sequence in FASTA format.

```
[efstae01@] $ ls -lh $IGENOMES_ROOT/Mus_musculus/UCSC/mm10/Sequence/WholeGenomeFasta/genome.fa
```

- The Reference Index (lookup table) file helps access any region (sub-sequence) of the reference genome quickly. Text file containing one line for each chromosome (contig).
  Format: Sequence Name, Sequence Length, Offset of first base of the sequence in the file, Length (number of bases) in each line in Reference FASTA file, Number of Bytes in each line.

```
[efstae01@$] more genome.fa.fai
chr10 129993255 7 50 51
chr11 121843856 132593135 50 51
chr12 121257530 256873876 50 51
chr13 120284312 380556564 50 51
chr14 125194864 503246570 50 51
chr15 103494974 630945339 50 51
chr16 98319150 736510220 50 51
..........
.....
```
Using **BWA**

```
[efstae01]$ module avail bwa
[efstae01]$ module load bwa/gnu/0.7.8
[efstae01]$ module display bwa
[efstae01]$ export REF=$IGENOMES_ROOT/Mus_musculus/UCSC/mm10/Sequence/BWAIndex/genome.fa
```

The BWA **aln** command generates the alignments in Suffix Array (SA) coordinates

```
[efstae01@phoenix1 ~]$ bwa aln $REF ChIPseq_chr19.fastq -f ChIPseq_chr19.sai
```

The BWA **samse** command converts to chromosomal coordinates

```
[efstae01@phoenix1 ~]$ bwa samse $REF ChIPseq_chr19.sai ChIPseq_chr19.fastq -f \ ChIPseq_chr19.sam
```
**Exercise: How many alignments are listed in the SAM file?**

```plaintext
[efste01@phoenix1 ~]$ more ChIPseq_chr19.sam
@SQ SN:chr1  LN:195471971
@SQ SN:chr2  LN:182113224
@SQ SN:chr3  LN:160039680
@SQ SN:chr4  LN:156508116
@SQ SN:chr5  LN:151834684
@SQ SN:chr6  LN:149736546
@SQ SN:chr7  LN:145441459
@SQ SN:chr8  LN:129401213
@SQ SN:chr9  LN:124595110
@SQ SN:chrM LN:16299
@SQ SN:chrX LN:171031299
@SQ SN:chrY LN:91744698
@SQ SN:chr10  LN:130694993
@SQ SN:chr11  LN:122082543
@SQ SN:chr12  LN:120129022
@SQ SN:chr13  LN:120421639
@SQ SN:chr14  LN:124902244
@SQ SN:chr15  LN:104043685
@SQ SN:chr16  LN:98207768
@SQ SN:chr17  LN:94987271
@SQ SN:chr18  LN:90702639
@SQ SN:chr19  LN:61431566
@SQ HWUSI-EAS610_0001:3:1:4:1405#0  16 chr19  60798324  37  36M  *  0  0
     CACTCAGCTCCTCTGATCTCTGGGAATTGAACTATC  XT:A:U  NM:i:1  X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0  MD:Z:9T26
@SQ HWUSI-EAS610_0001:3:1:5:1490#0  0 chr19  32885883  37  36M  *  0  0
     GGGCTGGTGGAGTGATCCCAAGGGGTGGGGATGGGG  XT:A:U  NM:i:0  X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0  MD:Z:36
@SQ HWUSI-EAS610_0001:3:1:6:388#0  16 chr19  18103063  37  36M  *  0  0
     GTCTAGGAAGACTAGAGGCCTATTTCATGAACTCTG  XT:A:U  NM:i:0  X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0  MD:Z:36
@SQ HWUSI-EAS610_0001:3:1:7:1045#0  16 chr19  60622265  37  36M  *  0  0
     ATGTGAGGCAATGTGCTCCATTTCCTTCCCTATCC  XT:A:U  NM:i:0  X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0  MD:Z:36
```
SAM Mandatory Fields

1. QNAME: Query name of the read or the read pair
2. FLAG: Bitwise flag (pairing, strand, mate strand, etc.)
3. RNAME: Reference sequence name
4. POS: 1-Based leftmost position of clipped alignment
5. MAPQ: Mapping quality (Phred-scaled)
6. CIGAR: Extended CIGAR string (operations: MIDNSHP)
7. MRNM: Mate reference name (‘=’ if same as RNAME)
8. MPOS: 1-based leftmost mate position
9. ISIZE: Inferred insert size
10. SEQQuery: Sequence on the same strand as the reference
11. QUAL: Query quality (ASCII-33=Phred base quality)

http://samtools.sourceforge.net/SAM1.pdf
## Mapping Quality (MAPQ) in BWA

Mapping Quality is a function of **Edit Distance** and the **Uniqueness** of the alignment.

<table>
<thead>
<tr>
<th>BWA Mapping Quality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A read aligns <em>equally well</em> to multiple positions (hits). BWA picks randomly one of the positions and assigns MAPQ=0</td>
</tr>
<tr>
<td>1 – 23</td>
<td>Only 1 Best hit (with no suboptimal hits) with more than 2 mismatches. Or Only 1 Best hit, with 1 suboptimal hit.</td>
</tr>
<tr>
<td>25</td>
<td>Only 1 Best hit (no suboptimal hits), with up to 2 mismatches (edit distance could be more than 2)</td>
</tr>
<tr>
<td>37</td>
<td>Only 1 Best hit (no suboptimal hits), with up to 2 mismatches (edit distance could be more than 2)</td>
</tr>
</tbody>
</table>
SAM/BAM format

Header section

Alignment section

CIGAR operators

M: match/mismatch
I: insertion
D: deletion
S: softclip
H: hardclip
P: padding
N: skip

Ref: GCATTCAGATGAGTGATACGC
Read: ccTCAG--GCAGTGttg

POS CIGAR
5 2S4M2D6M3S
What is the cigar line?

E.g. Read: ACGCA–TGCAGTtagacgt
Ref: ACTCAGTG—GT
Cigar: 5M1D2M2I2M7S

E.g. Read: tgtcgtcACGCATG---CAGTtagacgt
Ref: ACGCATGCGGCAGT
Cigar:
Post-processing: Tools and programming APIs for parsing and manipulating alignments:

**Samtools:** [http://samtools.sourceforge.net/](http://samtools.sourceforge.net/)
- Convert SAM to BAM and vice versa
- Sort and Index BAM files
- Merge multiple BAM files
- Show alignments in text viewer
- Remove Duplicates from PCR amplification step

**Picard Tools:** (Java-based)
Converting the **SAM** file to a **BAM** file

Binary, platform independent format, resulting in more efficient storage.

```
[efstae01@]$ module avail samtools

[efstae01@]$ module load samtools

[efstae01@]$ module display samtools

[efstae01@]$ samtools view -bt $REF -o ChIPseq_chr19.bam ChIPseq_chr19.sam

[samopen] SAM header is present: 22 sequences.

[efstae01@]$ samtools sort ChIPseq_chr19.bam ChIPseq_chr19.sort

[efstae01@]$ samtools index ChIPseq_chr19.sort.bam
```
Examining the **BAM** file

```bash
$ samtools view -c ChIPseq_chr19.sorted.bam

$ samtools view -c -q 30 ChIPseq_chr19.sorted.bam

$ samtools view -c -q 30 ChIPseq_chr19.sorted.bam \ chr19:10,000,000-11,000,000

$ samtools view -c -f 4 ChIPseq_chr19.sorted.bam

$ samtools view -c -F 4 ChIPseq_chr19.sorted.bam
```
FASTQ Format:
• The de-facto file format for sharing sequence read data
• Sequence and a per-base quality score

SAM (Sequence Alignment/Map) format:
• A unified format for storing read alignments to a reference genome.
• Generally large files (a byte per bp)
• Very compact in size but computationally efficient to access.

BAM (Binary Alignment/Map) format:
• A Binary equivalent to SAM.
• Developed for fast processing and indexing

http://bioinformatics.oxfordjournals.org/cgi/reprint/btp352v1
Putting it all together

In your `project1` directory:

-bash-3.2$ `pwd`
`/scrarch/NetId/project1`

-bash-3.2$ `cp /share/apps/examples/bio-workflow/align.pbs .`

== Examine the file using `nano`:
-bash-3.2$ `nano align.pbs`

== Submit the job using `qsub`:
-bash-3.2$ `qsub align.pbs`

-bash-3.2$ `qstat –u $USER`
NYU HPC Contacts

- NYU HPC Wiki: https://wikis.nyu.edu/display/NYUHPC/High+Performance+Computing+at+NYU

- Email Contact: hpc@nyu.edu