Sequencing data

Read-mapping and assembly

ILLINOIS INSTITUTE OF TECHNOLOGY

Jean-François Pombert, Ph.D.
Office PS 296 (Lab PS 340)

Content

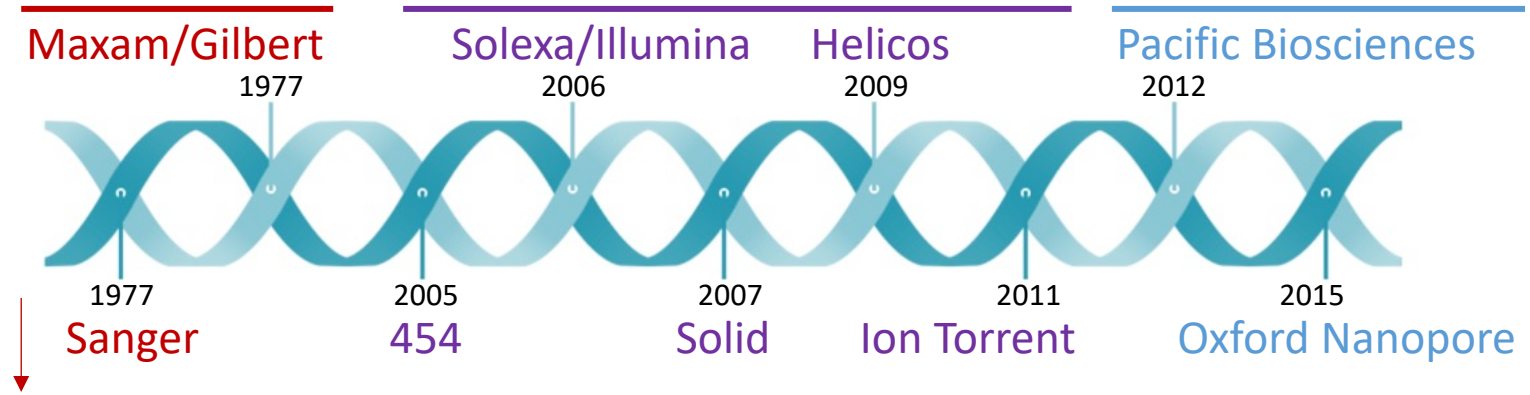
I. Sequencing technologies ## A quick primer about the various platforms
II. Sequencing data ## Files types and possible issues to expect
III. Read-mapping approaches ## Useful to assess genetic diversity
IV. Genome assembly ## Required for genome sequencing

I – Sequencing technologies

Short but accurate or long and noisy sequencing reads?

The sequence of sequencers: The history of sequencing DNA
Heather JM, et al.
Genomics. 2016. PMID: 26554401

Sequencing – By Age



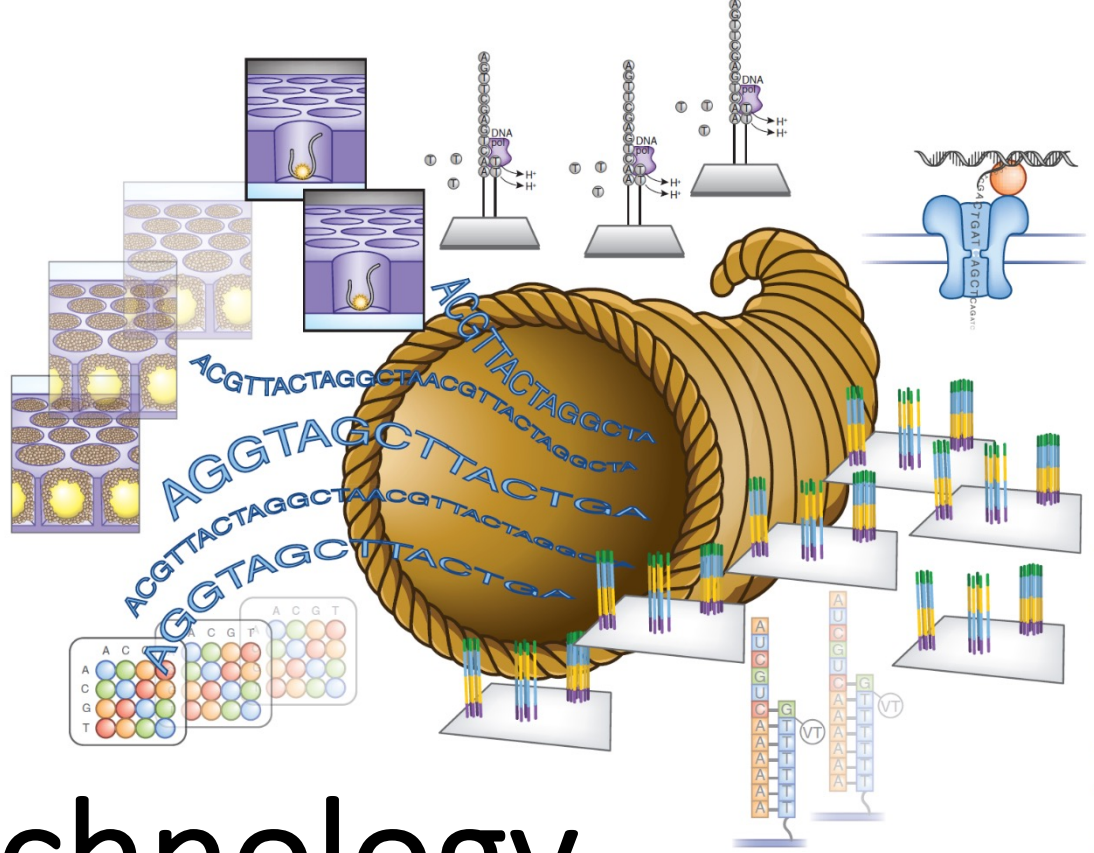
1975 plus/minus (Coulson & Sanger)

Generation

- 1st Sanger, MG
- 2nd HT short reads
- 3rd HT long reads

HT => high throughput

A defining decade in DNA sequencing
McPherson JD
Nat Methods. 2014. PMID: 25264775
DOI: 10.1038/nmeth.3106



Marina Corral Spence/Nature Publishing Group

Sequencing – By Technology

SBS	(Reversible) dyed-terminators	Sanger, Illumina, Helicos
	Fixed polymerase + dyed-nucleotides	Pacific Biosciences
	Intensity-based	454 pyrosequencing (light), IonTorrent (pH)
	Ligation-based	SOLiD
	Electrical variations	Oxford Nanopore
	Degradation	Maxam-Gilbert

Platforms in blue are commonly encountered

First generation – Sanger sequencing

Sequencing by synthesis

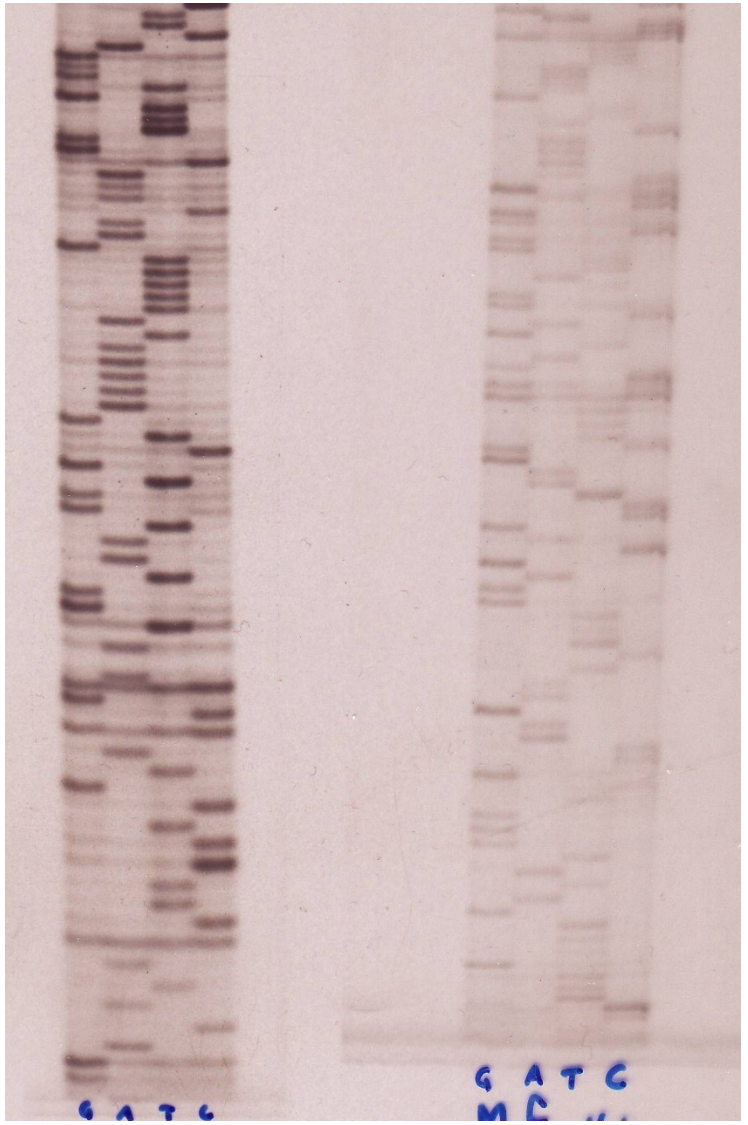
Uses non-reversible terminators

Fragments generated are size-separated by gel electrophoresis

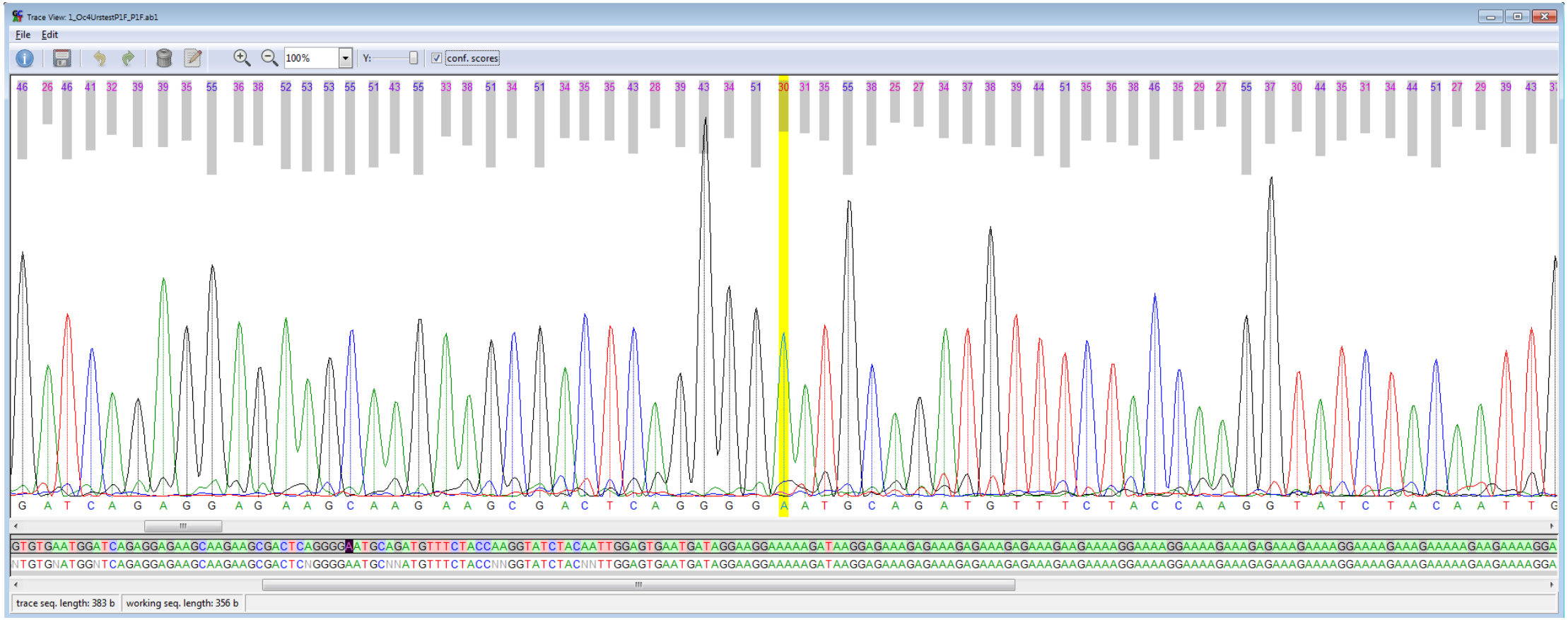
Great with PCRs

Old - Four reactions + P^{32} labelling

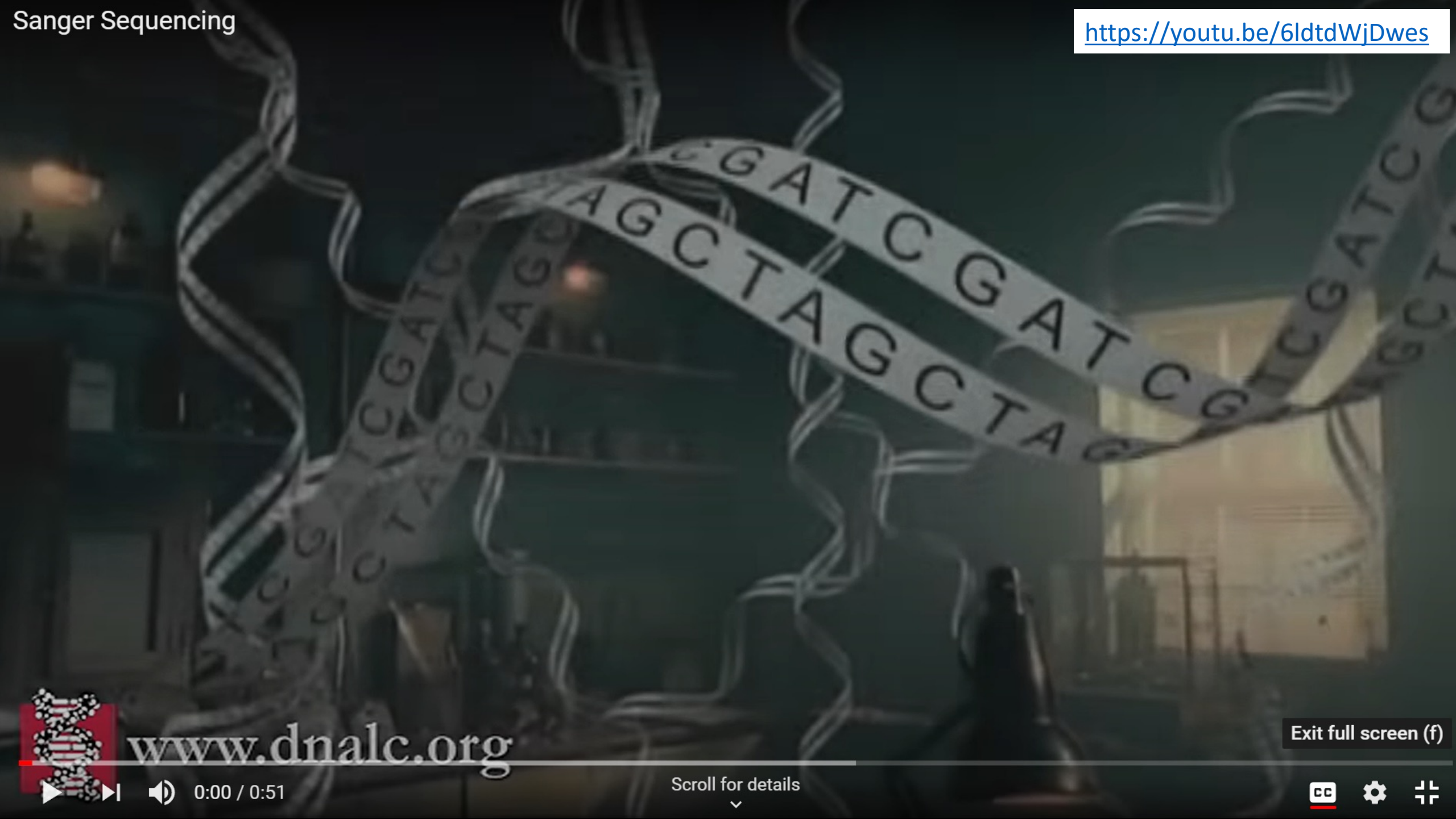
Now - One reaction + fluorochromes



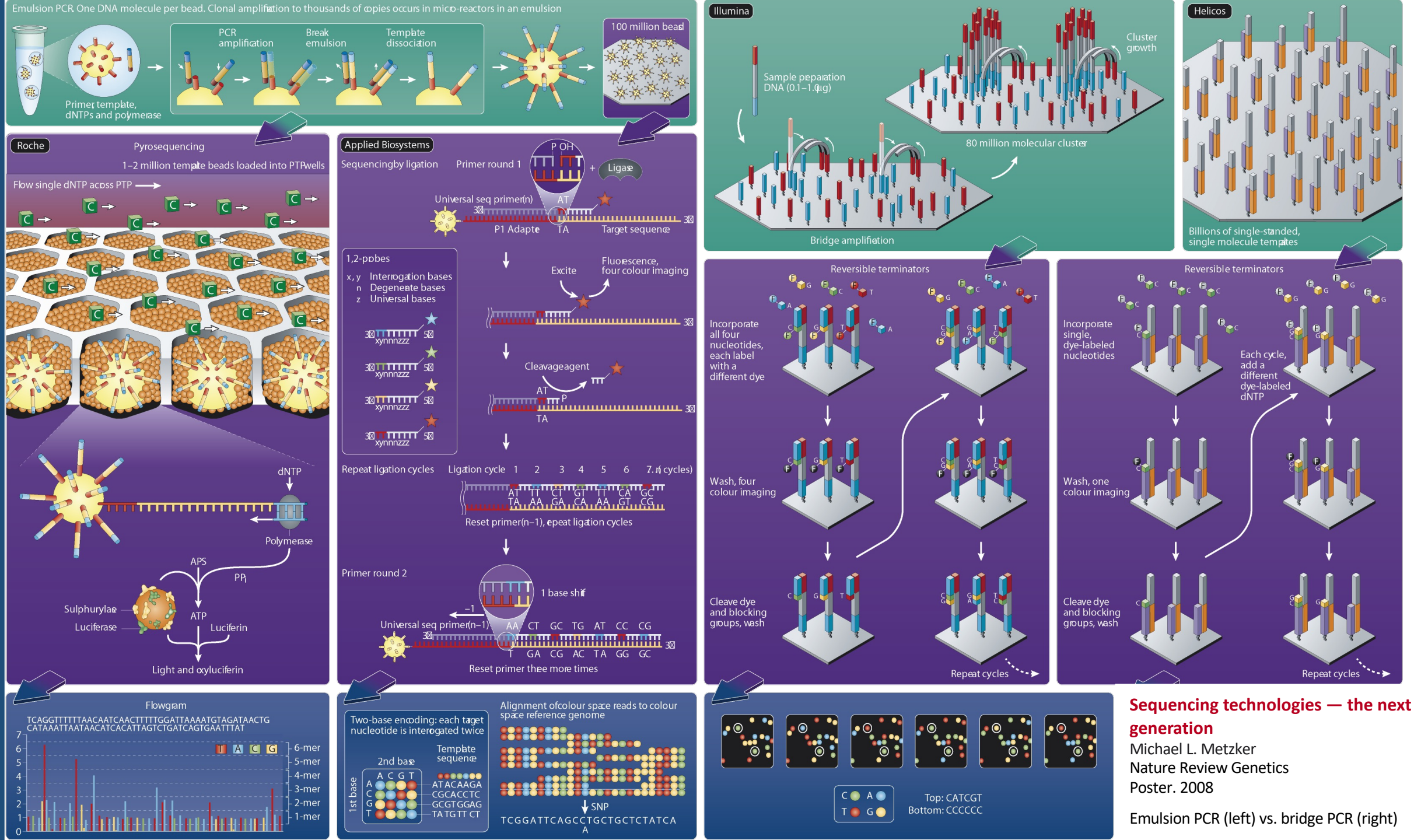
A Sanger chromatogram



code.google.com/p/seqtrace


Sanger Sequencing
https://youtu.be/6ldtdWjDwes
GATCGATCG
AGCTAGCTA
www.dnalc.org
0:00 / 0:51
Scroll for details
Exit full screen (f)

A small video by yourgenome.org showing the basics of Sanger sequencing https://youtu.be/ONGdehkB8jU



Sequencing technologies — the next generation

Michael L. Metzker
Nature Review Genetics
Poster. 2008

Emulsion PCR (left) vs. bridge PCR (right)

Second generation

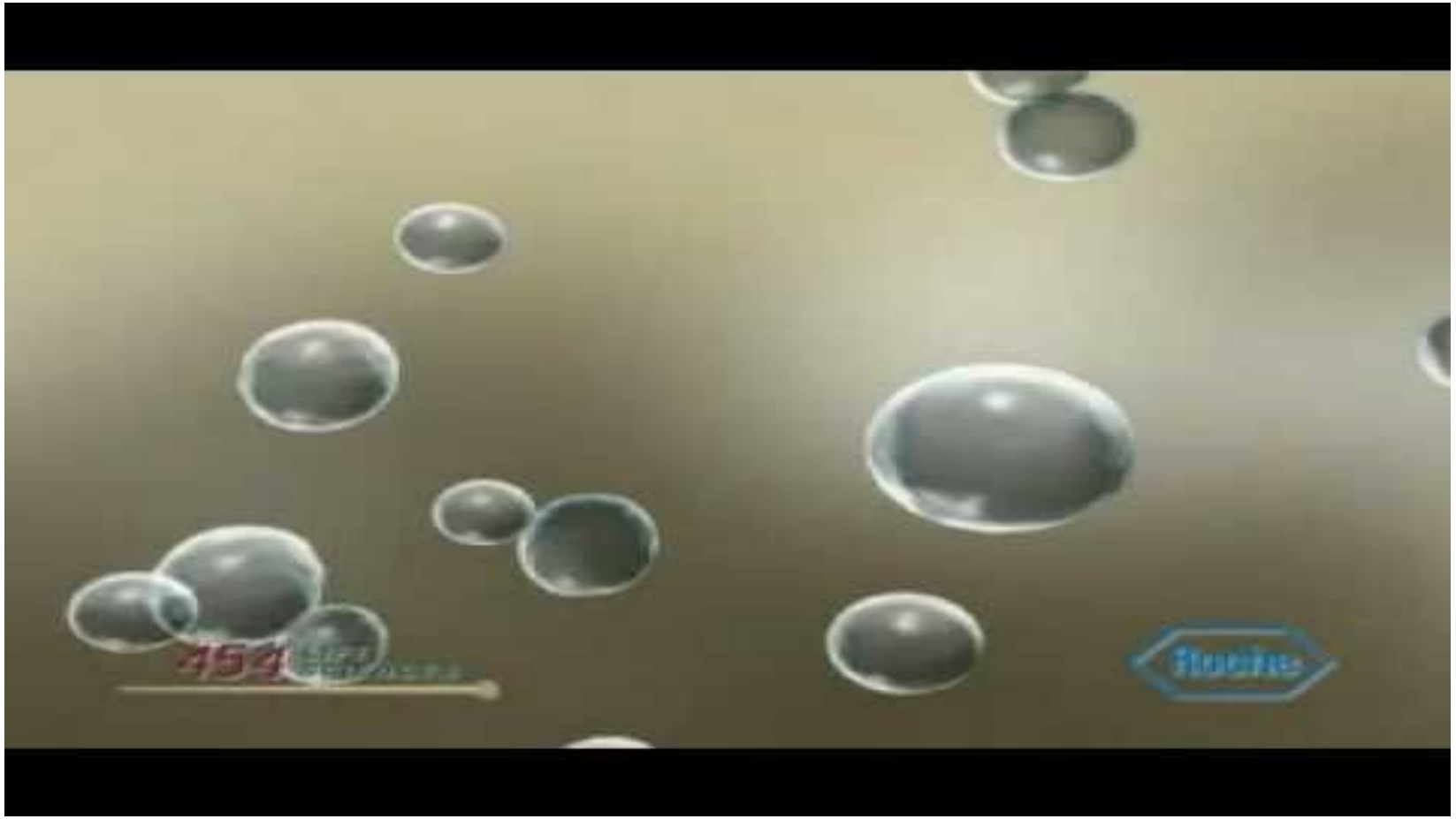
454 SEQUENCING

Roche

Pyrosequencing

Light is emitted by the luciferase when a base is added to the DNA chain

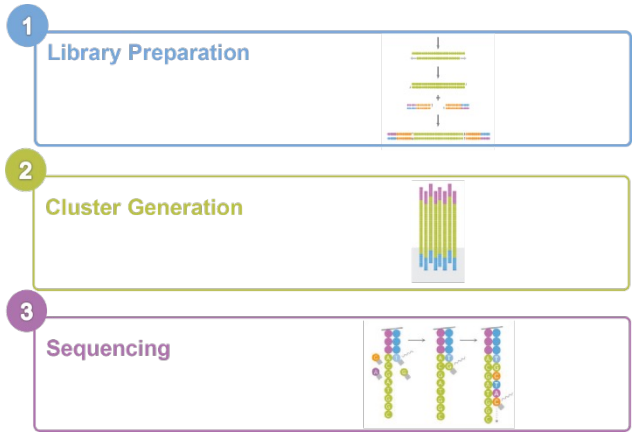
- Up to 1,000 bp
- ~ 700 Mb per GS FLX+ run
- Run Time 23 hours
- Homopolymer issues
- Launched in 2005 (GS20)
- Shut down mid-2016

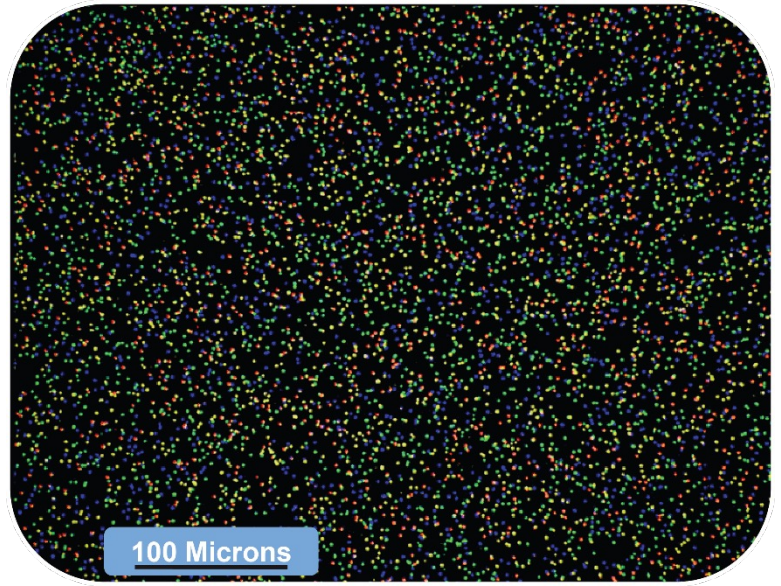


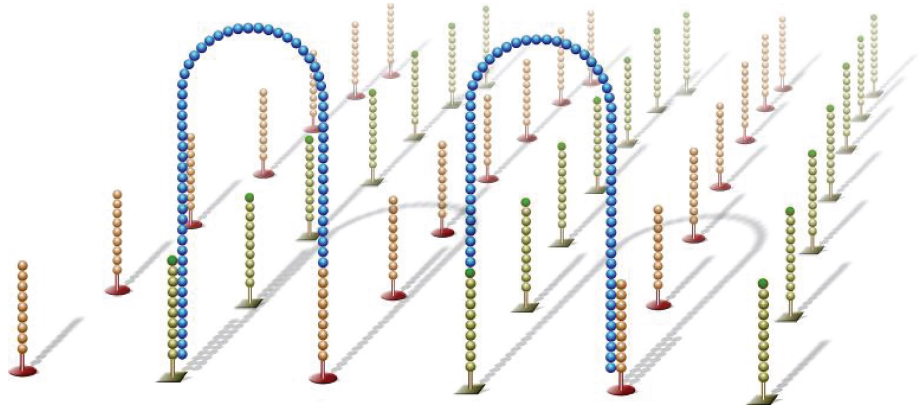
https://youtu.be/bFNjxKHP8Jc

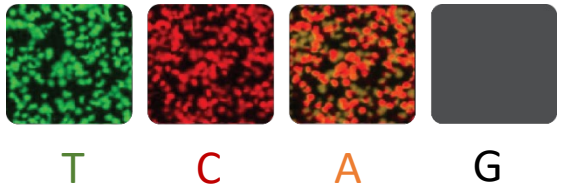
The development and impact of 454 sequencing
Rothberg JM, et al.
Nat Biotechnol. 2008. PMID: 18846085

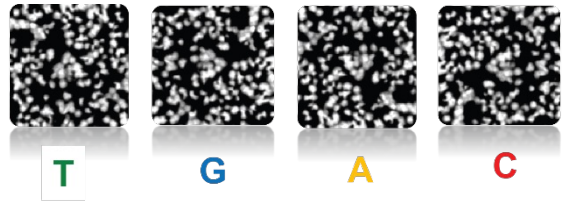
Second generation – illumina®

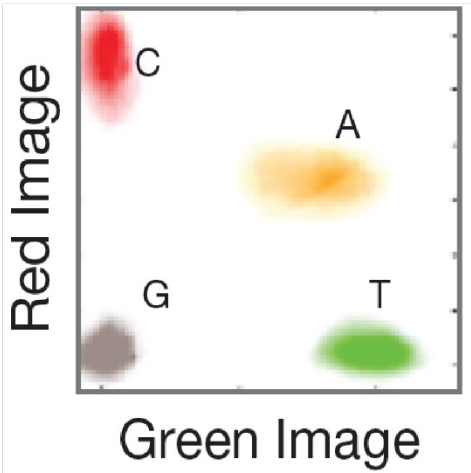












4, 2 or 1 channel(s)

Source: Illumina.com

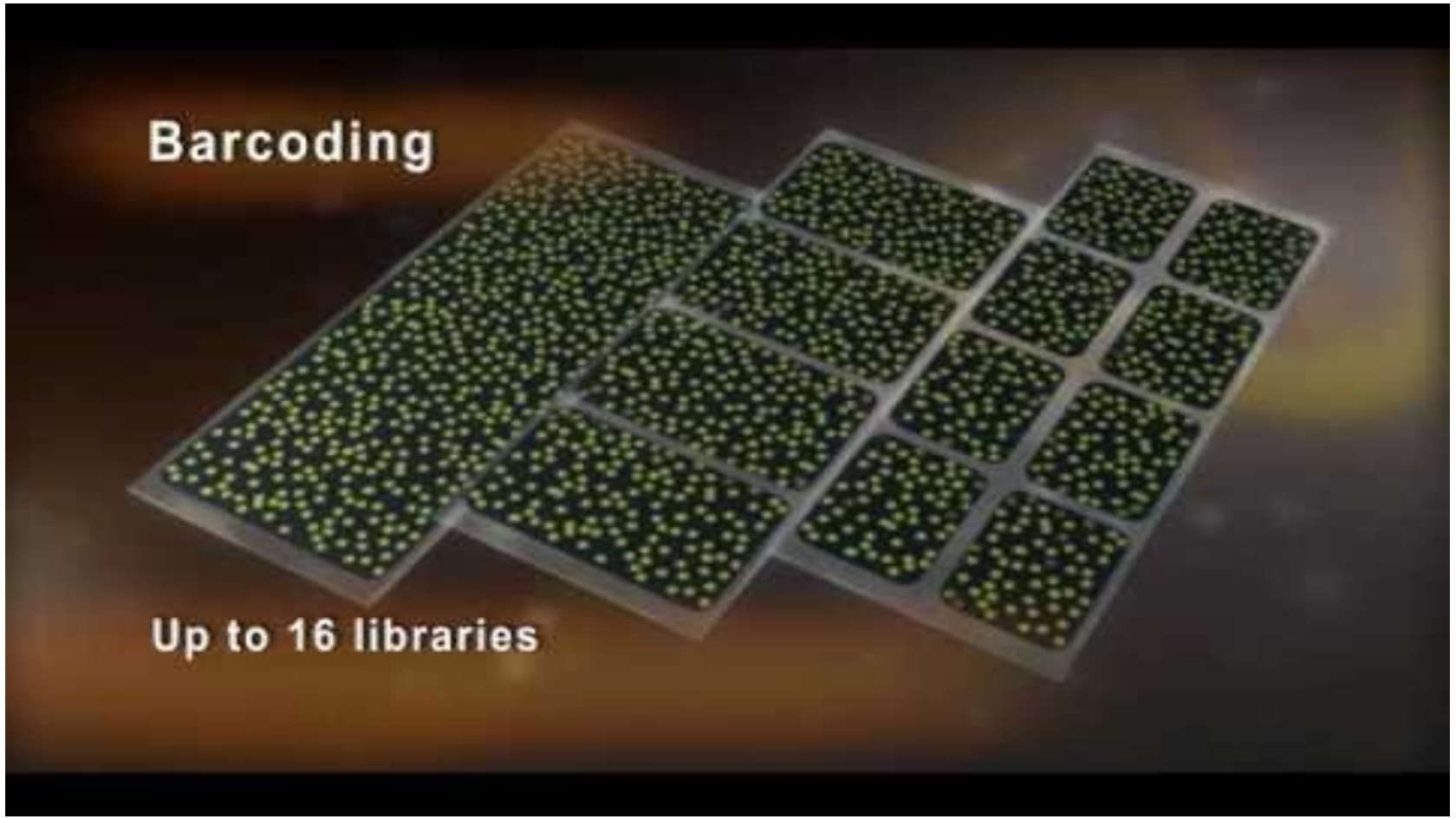

Illumina Sequencing by Synthesis
https://youtu.be/fCd6B5HRaZ8
The Illumina sequencing workflow is composed
of 4 basic steps– sample prep, cluster generation,
0:09 / 5:12
Scroll for details

Second generation – SOLiD™

Ligation-based

Dimers are added, then the 3’ base is cleaved off

- Emulsion PCR
- Up to 75 nt
- Bases queried twice
- Phasing issues
- Launched in 2006
- Obsolete



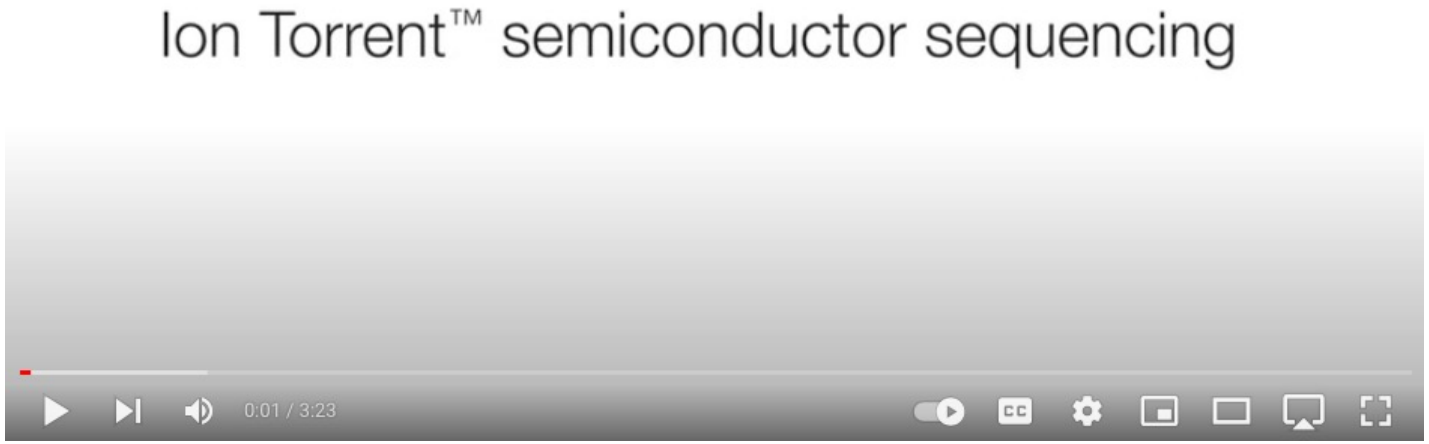
https://youtu.be/nlvyF8bFDwM

Second generation – ion torrent

pH-based

H+ is released upon base incorporation into DNA

- Emulsion PCR
- Up to 400 nt
- Cheap, Fast
- Launched in 2010
- Homopolymer issues



https://youtu.be/zBPKj0mMcDg

Third generation – PACIFIC BIOSCIENCES®

Lower accuracy, but stochastic errors
- 35,000 bp+ reads
- Requires 10 ug of input DNA for longest reads

SMRT cells; 500 to 1.5 Gb (x7 on the Sequel)

Kits; P6 – polymerase, C4 – chemistry

Can detect base modifications

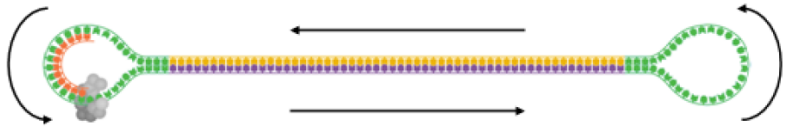
Launched in 2011 - Sequel (~ 350K$) launched in 2015

CCS - Closed Circular Consensus

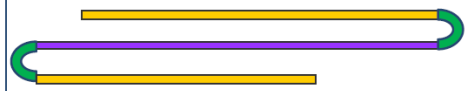
- Polymerase can roll around the read.
- If insert is short, it will be read multiple times.
- The CCS is a consensus of each subreads

***De novo* assembly - PacBio**
User Group Meeting, Menlo Park
2014
Click for URL

Read Metrics Definitions



SMRTbell™ Template



Polymerase Read

Definition:
- Sequence of nucleotides incorporated by polymerase while reading a template
- Includes adapters
- Often called "read"
- Includes adapters
- 1 molecule, 1 pol. read

Purpose:
- QC of instrument run
- Benchmarking

Subread

Definition:
- Single pass of template
- Adapters removed
- 1 molecule, ≥1 subreads

Unique data:
- Kinetic measurements
- Rich QVs

Purpose:
- For subsequent analysis

Read of Insert

Definition:
- Represents highest-quality single-sequence for an insert, regardless of number of passes
- Generalizes CCS for <2 passes and RQ <0.9
- 1 or more passes
- 1 molecule, 1 read

Purpose:
- For Library QC
- For subsequent analysis

PacBio Sequencing – How it Works

https://youtu.be/_lD8JyAbwEo

PACBIO

Single Molecule, Real-Time (SMRT®) Sequencing

0:10 / 1:29

Scroll for details

Third generation – Oxford NANOPORE Technologies®

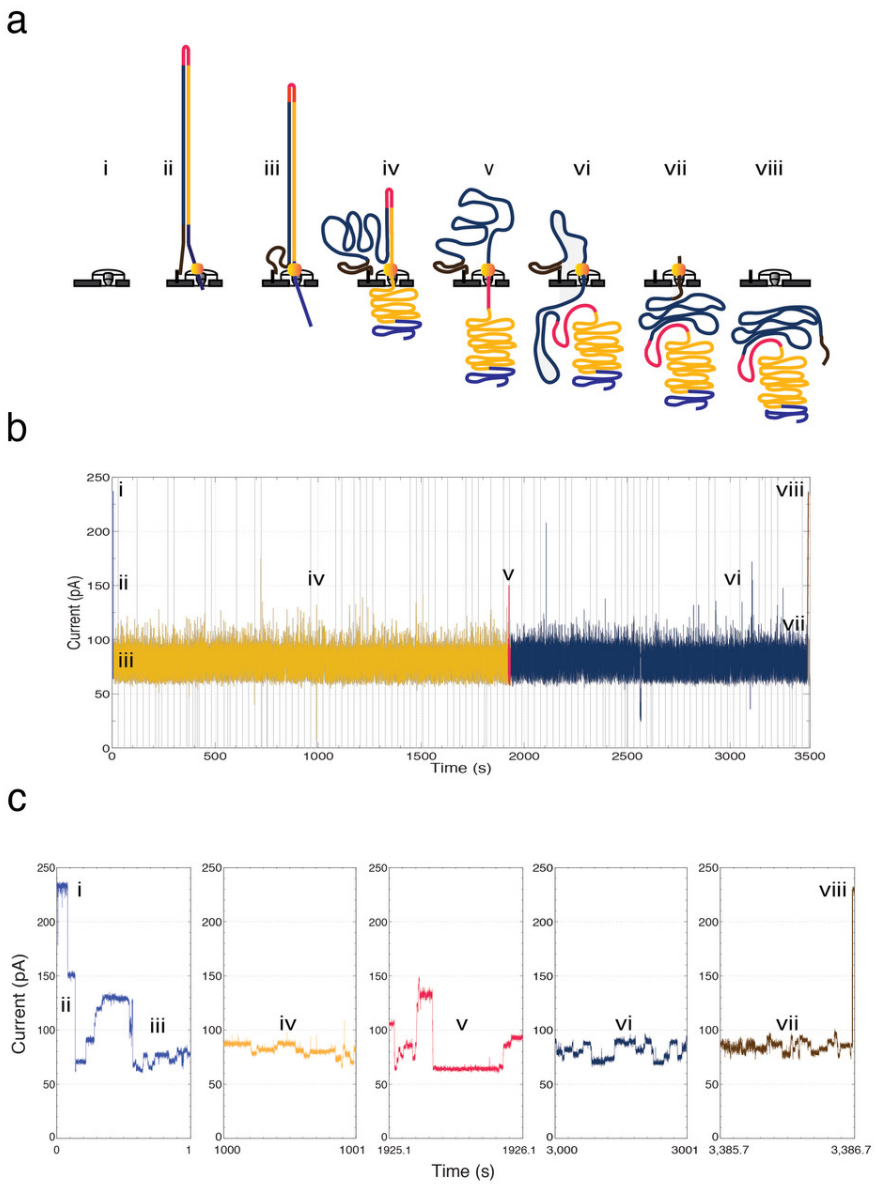
1D, 2D, or 1D^2?

Single strands travel through pores

Induce amperes (pA) variations

Complementary strands increase base calling

"U-turn" adapters added to DNA (2D)
are old tech, no longer available

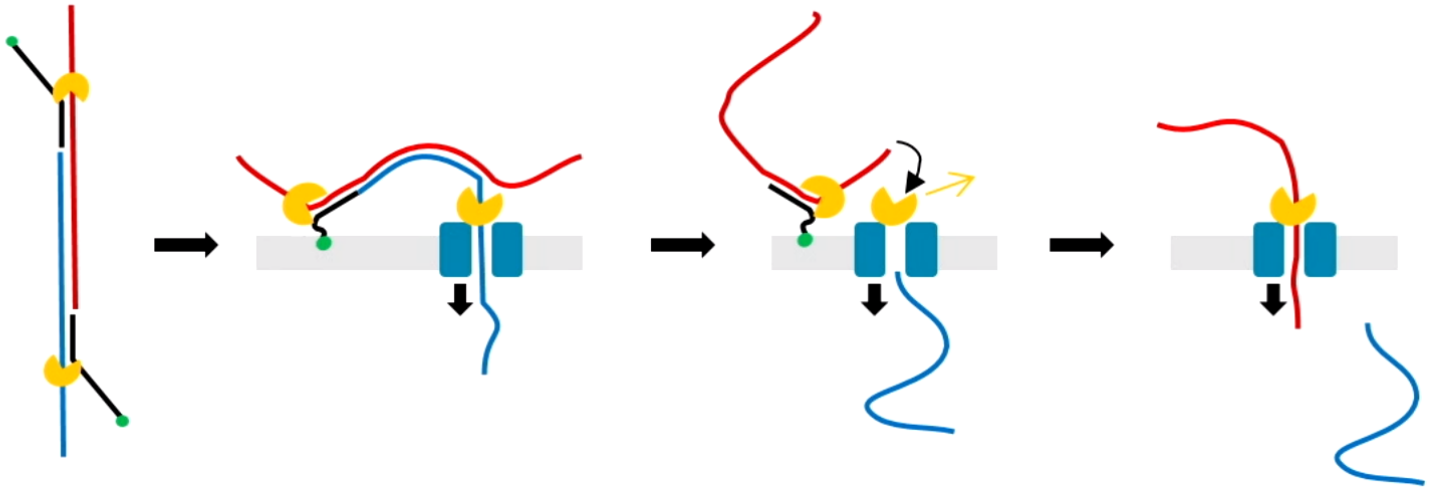


Improved data analysis for the MinION nanopore sequencer
Jain M, Fiddes IT, Miga KH, Olsen HE, Paten B, Akeson M
Nature Methods. 2015. 12:351–356
DOI: 10.1038/nmeth.3290

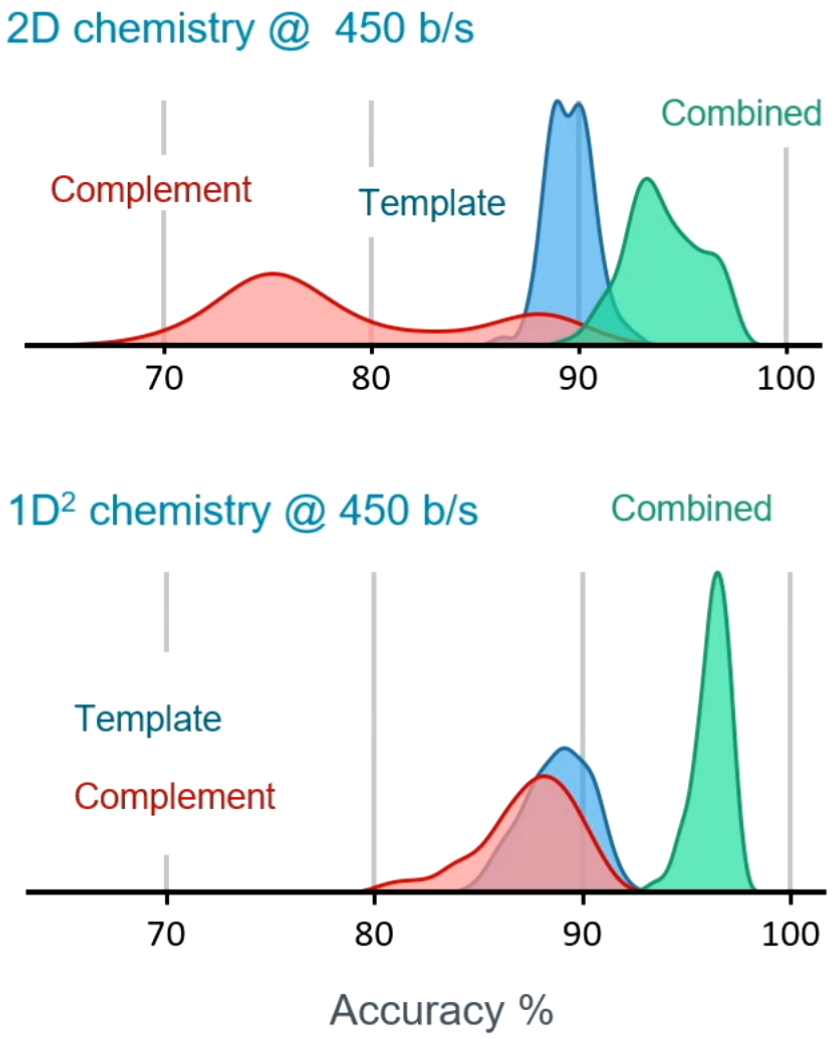
IMPROVING ON AND REPLACING 2D

$1D^2$

- New sequencing chemistry scheme where strands are not joined
- Simple library preparation, very similar to current 1D chemistries
  - Compatible with E8 and 450 b/s
- Each individual strand has high 1D accuracy (unlike current 2D)
- Data from both strands combined into a single basecall

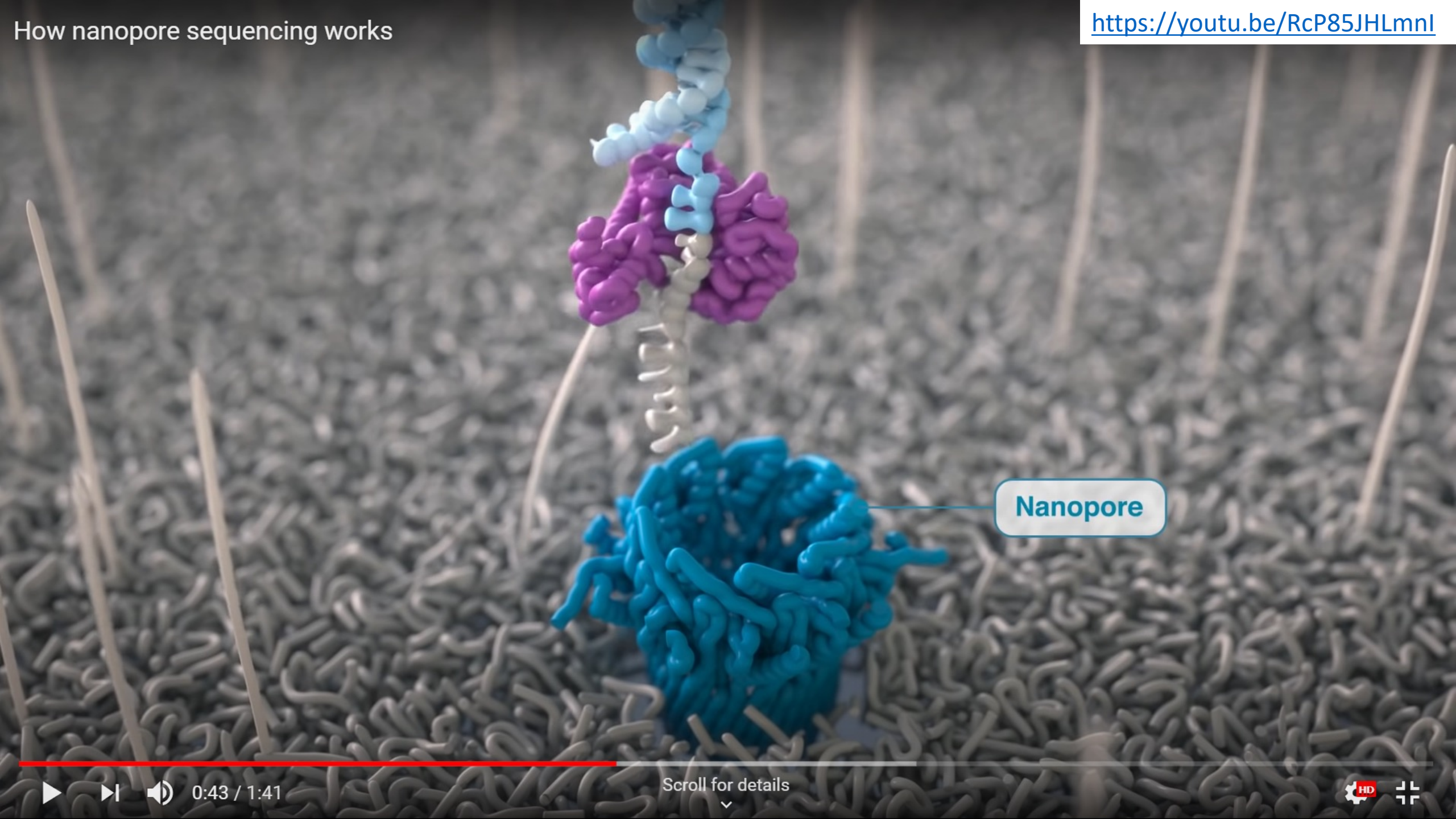


Expected accuracy



Dec 2016, Clive G Brown - https://nanoporetech.com/resource-centre/talk/clive-brown-plenary-talk-0

Oxford NANOPORE Technologies


How nanopore sequencing works
https://youtu.be/RcP85JHLmnI
Nanopore
0:43 / 1:41
Scroll for details

New tech? – New challenges!

Accuracy?

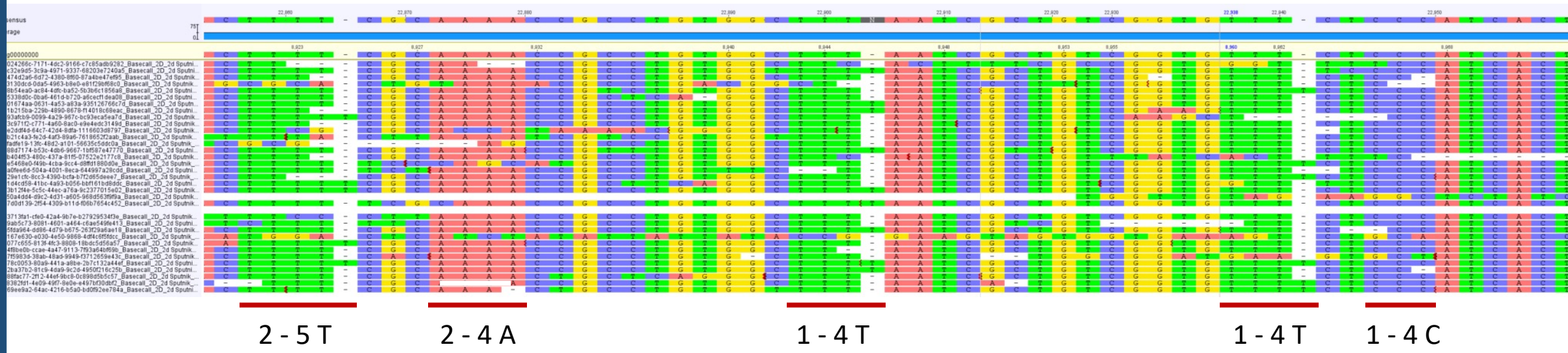
Homopolymers?

Phasing?

New file format?

Need help?

SEQanswers	http://www.seqanswers.com/
Biostars	https://www.biostars.org/



WARNING! – Homopolymers

Homopolymers are stretches of identical nucleotides (*e.g.* AAAAAA, TTTTTT, CCCC, GGGG)

Systematic error; Intensity-based sequencing technologies could not ascertain the # repeated bases

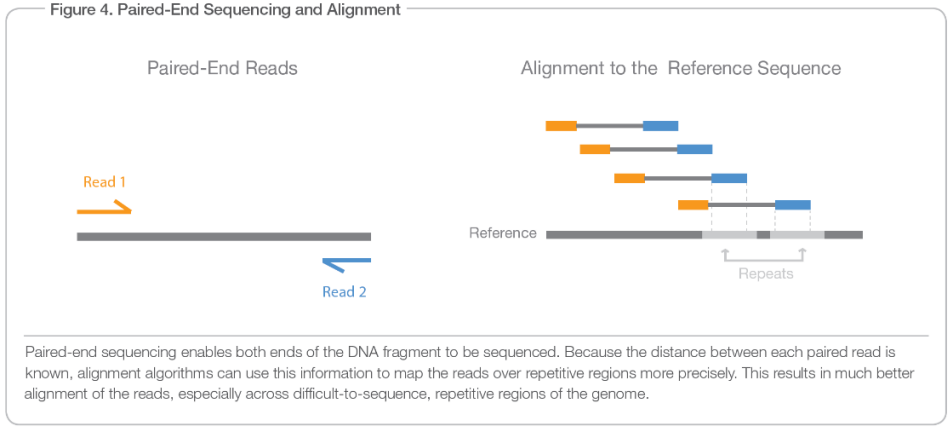
Also problematic with Nanopore! (same base = no variation in electrical current)

Cannot be corrected by increased sequencing depth

Not an issue with sequencing approaches based on reversible-terminators

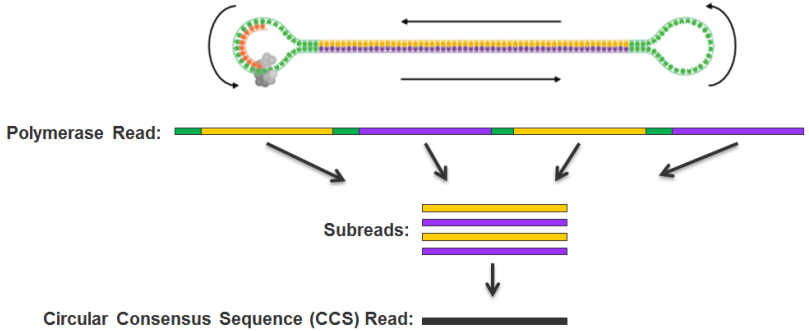
II – Sequencing data

Illumina



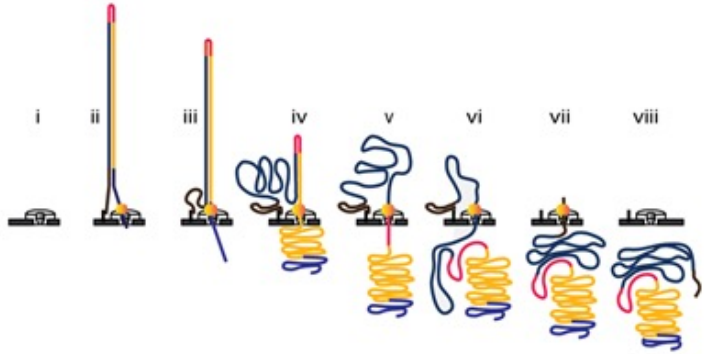
An introduction to next-generation sequencing
Illumina
2013. Datasheet

PacBio

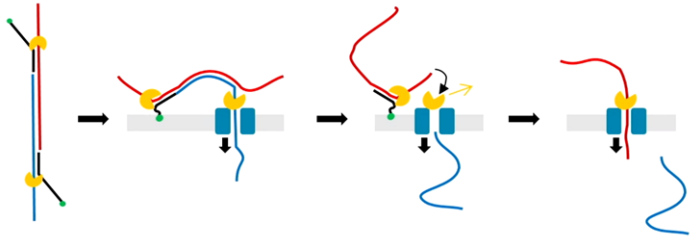


Source: PacBio

Nanopore



Improved data analysis for the MinION nanopore sequencer
Jain M, Fiddes IT, Miga KH, Olsen HE, Paten B, Akeson M
Nature Methods. 2015. 12:351–356



Source: Nanopore

Type of reads

Single ends (SE)

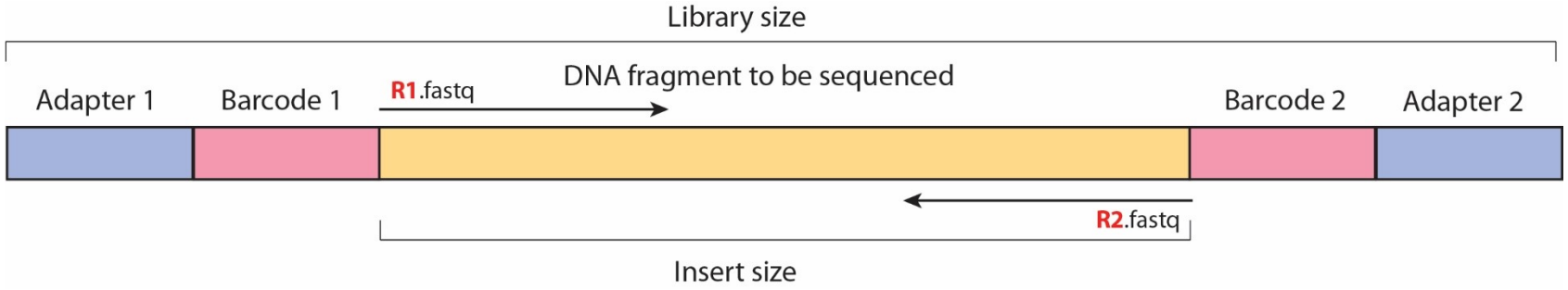
Paired-Ends (PE) ## Both ends of the same fragment; short distance

Mate-Pairs (MP) ## Two sides of a synthetic construct; long distance; replaced by long reads

Subreads/CCS ## PacBio subreads/circular consensus sequences

1D, ~~2D~~, 1D^2 ## Nanopore reads (1 or both strands)

About paired-end library insert sizes



A typical Illumina library

An 80 bases insert

	25 bases	30 bases	25 bases
In theory	TATTTGTAGCTGACGCTAGCTAGCT	NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN	GTACGTGCATCGCAGTCGATGCGTC

	25 bases	20 bases	25 bases
In practice	TATTTGTAGCTGACGCTAGCTAGCT	NNNNNNNNNNNNNNNNNNNN	GTACGTGCATCGCAGTCGATGCGTC

	25 bases	40 bases	25 bases
or	TATTTGTAGCTGACGCTAGCTAGCT	NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN	GTACGTGCATCGCAGTCGATGCGTC

Deviations are expected

https://projects.nfstc.org/workshops/resources/articles/ABIF_File_Format.pdf
https://support.illumina.com/bulletins/2016/04/fastq-files-explained.html
https://samtools.github.io/hts-specs/SAMv1.pdf
https://github.com/nanoporetech/ont_fast5_api

Sequencing output formats

ABI/AB1	Sanger	## Binary format; Applied Biosystem 1
FASTQ	Illumina	## Text file; **the standard in sequencing**
BAM	PacBio (Sequel+)	## Based on Samtools standard format
FAST5	Oxford Nanopore	## Single or multi

* BAM and FAST5 files are often converted to FASTQ format for downstream analyses

Other formats

BAX.H5	PacBio (Old format)	## Based on HDF5 data model
FASTQ	Solexa (ASCII 64)	## Old quality score encoding
FASTQ	SOLiD (colorspace)	## Defunct sequencing platform
SFF	454	## Defunct sequencing platform

Garbage in? => Garbage out…

Bad sequencing run? -> trim QVs

Sequencing library adapters? -> Must remove those!

Got contaminants?

- Know which ones? -> map & filter
- Don’t know? -> assemble, BLAST, map & filter, reassemble

QVs – Quality values

aka quality scores

Score given to each base

Probability that the base is erroneous

The algorithms to assign QVs differ depending on the sequencing technology

Quality scores – Phred scoring

Q is the score

P is the error probability

$Q = -10 \log_{10} P$

Q-Score	*P* error	Accuracy
10	1/10	90%
20	1/100	99%
30	1/1000	99.9%
40	1/10000	99.99%
50	1/100000	99.999%

Values below Q = 30 are considered bad, 28 is often the bare minimum

Base-calling of automated sequencer traces using phred. I. Accuracy assessment
Ewing *et al*.
Genome Res. 1998. 8(3):175-185
DOI: 10.1101/gr.8.3.175

Base-calling of automated sequencer traces using phred. II. Error probabilities
Ewing & Green
Genome Res. 1998. 8(3):186-194
DOI: 10.1101/gr.8.3.186

FASTA (+ .qual) files

Fasta contains only sequences

Quality values are found in separate .qual files

Sanger chromatograms are usually

binary .ab1 files

File.fasta

```
>sequence_1
ATGCCCCAAA
>sequence_2
GCCTTTAATT
```

File.qual

```
>sequence_1
33 34 39 39 38 37 32 29 28 26
>sequence_2
31 31 34 35 37 35 39 39 40 39
```

FASTQ files

Single, interleaved, or separated R1 and R2 files

QVs coded in ASCII: Phred+33 (Sanger + Solexa 1.4+), Phred+64 (Solexa ≤ 1.3)

CASAVA pipeline 1.8+
Solexa = illumina

A FASTQ file – (@ sequence, + qualities)

Identifies which file: 1 left or single, 2 right side of paired-ends

```
@M01893:20:000000000-A7TT6:1:1101:18726:1501 2:N:0:3
CATACTTGAAGCAAATGTACATCCTGCACCGGTATTCTTTTTATCCAGCACAGCTTCTTCAAACAATTTGTAGTCTGTTCCATCATAAACGACATCTATCGCTTTATTACCATCCACGCGATTTCCACCTTTGACACCGACATTCTTAGCCCCTA
+
--8,8<<,,,,;,,,<,<,C,6<C@,6,;<++8+:C,<6<CE,<CC,,;,;,,;@@@ECF,,,;,,;@C,<,,<CC,CC@C,<C,<,,,,,+8+8A@,<A,8A@@,B@,:?,:??,,,7++7?=E?9A?=BE,,9,,9++6++99A=,,4,444+
@M01893:20:000000000-A7TT6:1:1101:15534:1504 2:N:0:3
CTCAGCATGATTTTCGTCGACATCGACAGCTTTTTCCCGGTTTTTCACCATCCAGTCACCTGCGCAGTTACCACATTAGCAGCCAGTACCTTACCCTGTTACACGGTTGTCAGGTTCAACAGTGTCACGTCGACCACCAACCTCACTCCCCACTG
+
-88,-6,;,,<;;CC+;,++8,6;,,+,+;;;C@CC,,,++68C,,,<;,;;C,,<,,,;<,,++++89,<@,,,6,,,,,,9:,,:,:CE,,,:B=,??,:,,,+98+8,,,,:5A,,:,,:,:@,9+,+++88+9@,,8=4A,?=,,84,@>,
@M01893:20:000000000-A7TT6:1:1101:10793:1512 2:N:0:3
CGAGTCTTACCTGTTGAATCTTTCAGTACGACGCTCAATTCGTTAGCTTTCCGCTCACGACCTTCCAAACGTTTAGTGAATTTTTATTCCACTTCCCCTGGCACGTTCTTAACCCGTTTTTCCGTTGCTTTACTTTGTTTTTCACCTTTCTTAGT
+
--,,8,;C,;6;,C;,,,<<<C@,,,<,;,,++88,+,;<C,;;,6;@C@,,,88@+6,,8@@CEE,,,:,9CC,,<,,,,<9,,,<6C<,:@<A,,:B,,,,:,:B,:9,,9,9+74:?=E,+:5++:9A,,59,,99494A,,9A:?AA7,,8
@M01893:20:000000000-A7TT6:1:1101:14651:1524 2:N:0:3
CACCTTCGTATTCAGGCAGAGCATGATGCAGATCATCTGCGTGTGACTGCTGAACGCGAGAGTAAGGCTCAACGTAAAGAGTTCTTGTTAGAAGCAAAAGCGGAAGCTAGAAAATATCGAGAAGATATTGAAAAAGAATTCAAATCTGACCGTCA
+
<-A<@CF,C,,;C,,,;,,,,,;,;,,<,C,,,;;,<CC,<+8,8,6CC,;C,,,;,B+++++:,,,,:6C,,C,C,,,,,,<9EC@,:C,,,,,:,,,,,,+++++:?,,,,,,:,9@+++,+,,:,:A,,,,,,++,9A?,,,9AA,,,8+?=+
@M01893:20:000000000-A7TT6:1:1101:18376:1528 2:N:0:3
CGATCCGACGATCGTCGTTTAGTGCACGCGTGTATCTCTCGTTTTTCCCCGTCTCATTCACACACCCTCCCTCTCCCCCCCCCCCCTCCCCTTTCCCCCCCCTCCCTCTCCCCTCTCCCCCCCCCTCTCTCTCCTCTTCCCTCCCTTTCCCCTCCTC
+
6-,8A,,,+++,;+;C+8,:,,,,,,,,++8+8,,,;<@C,,9,,:C,9C+8,99,69,,,,,,,,+,6,,,,,,,9,64+4+++++4+++,,,5,,,+++++++,,,,,,3,+,3,3*3*5**1,,,,,,6,,,,,,,,2*1+,,,,,++**+2
@M01893:20:000000000-A7TT6:1:1101:18286:1537 2:N:0:3
CACTTACTGCAGACCTTGTTACAACACCTTAAGTAAAACTTTGTGCATATACCTCTCTTTAACTTTCTTTTGAAGCTTCTACCGAATTTACTTATGACTACTTTGTTGCTACGCTTTAACTCTACCGCAACAACTTTCCTGACTTGGTCGTTCAC
+
8-8A@,CE,C,,,,,;;,;;<,,,,86C,,,,<,,,,;CC,,<,;,;,6,,,,,;;C@,,,,66@C,,<C,,,,6;;C<,,,+,+,:6,<A,,<,,,,,,,99,,:,:A,,,48?,,,,,:,,,:+4+++,,9A?,,9=,,,9A,,99+8=,,,
@M01893:20:000000000-A7TT6:1:1101:15673:1545 2:N:0:3
CAAGATGACCTTGATTGATAATCTCCGTACGACAGCCCATGACCTGTCAGAGTAACCGGAGAAATTATGACTTTGCAAGACACTATTATTGCTCAACTAGTCGTTAAACCTAGCATTGATCCAAAGGAAGACATTCGAAAATCCATTGCTTTTTT
+
8-,,-;,,;CCE,,CC,,<,,;,<CF+=,,+,+++;CB,C,,;CC8CE,,,,<,,,,+++++,,9<,C,,:ECC,@,,,,,9CC,BC,CE,@6C,,BF,,,:+:B7,,CA=,,9,C?,,9??,,,,,,,,,,,9CF,,,++9A:,A;,,9=;?@>
@M01893:20:000000000-A7TT6:1:1101:18090:1546 2:N:0:3
CTTGCTTGATACGATCCGAACCGAAACGACGCATCACTTCGTCTTCAAGTGAAAGGTAGAATTGAGATTCCCCTGCGTCCCCTTGACGTCCCGCCCGACCACGCAACTGGTTATCAATACGACGGCTTTCGTGACGTTCTGTACCACTCACCCAA
+
888,8CC,,<,<,,6C,,,,8@++++86+6+6+87,,;<C,;@CC@,,,<,,,,,,:,,,9CC,,,,<CE@FCC,,+8@>=ECC,,:,8:?=+++8++8>+:+8++8A,,::,8C,,:,:++8++336>E:>+,8+88@C,9,8=,,@,,8,8++
```

A typical illumina FASTQ file; Qualities are coded so that double-digits values are represented by a single character

Two types

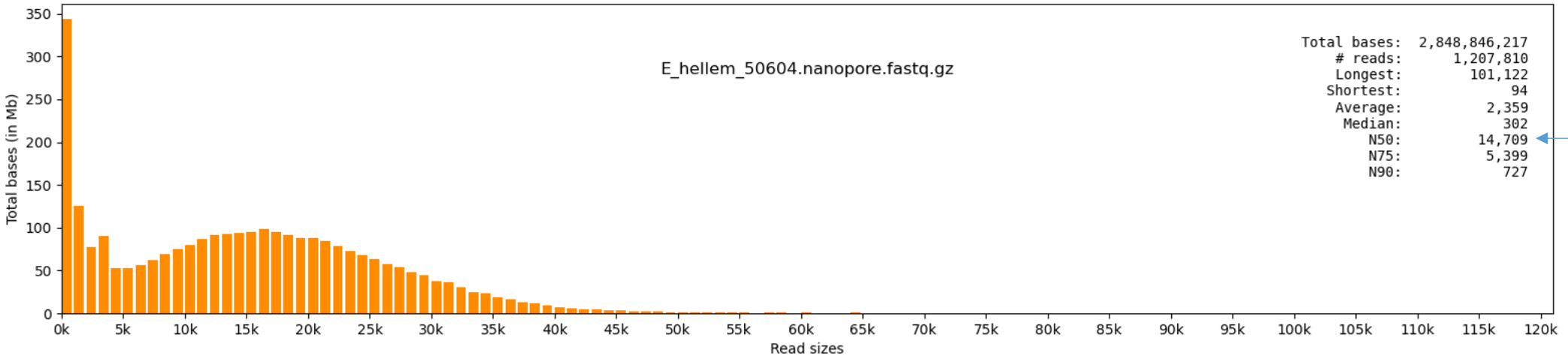
I) Left/right reads are separated in two files (*e.g.* R1.fastq R2.fastq)

II) Left/right reads are **interleaved** in one file

```
@M01893:20:000000000-A7TT6:1:1101:18726:1501 1:N:0:3
CATACTTGAAGCAAATGTACATCCTGCACCGGTATTCTTTTTATCCAGCACAGCTTCTTCAAACAAT
+
--8,8<<,,,,;,,,<,<,C,6<C@,6,;<++8+:C,<6<CE,<CC,,;;;,,,;@@@ECF,,,;,,;
@M01893:20:000000000-A7TT6:1:1101:18726:1501 2:N:0:3
CTCAGCATGATTTTCGTCGACATCGACAGCTTTTTCCCGGTTTTTTCACCATCCAGTCACCTGCGCAG
+
-88,-6,;,,<;;CC+;,++8,6;,,+,+;;;C@CC,,,++68C,,,<;,;;C,,<,,,;<,,++++
@M01893:20:000000000-A7TT6:1:1101:10793:1512 1:N:0:3
CGAGTCTTACCTGTTGAATCTTTCAGTACGACGCTCAATTCGTTAGCTTTCCGCTCACGACCTTCCA
+
--,,8,;C,;6;,C;,,,<<<C@,,,<,;,,++88,+,;<C,;;,6;@C@,,,88@+6,,8@@CEE,
@M01893:20:000000000-A7TT6:1:1101:10793:1512 2:N:0:3
CACCTTCGTATTCAGGCAGAGCATGATGCAGATCATCTGCGTGTGACTGCTGAACGCGAGAGTAAGG
+
<-A<@CF,C,,;C,,,;,,,,,;,,,<,C,,,;;,<CC,<+8,8,6CC,;C,,,;,B+++++:,,,,
@M01893:20:000000000-A7TT6:1:1101:18376:1528 1:N:0:3
CGATCCGACGATCGTCGTTTAGTGCACGCGTGTATCTCTCGTTTTTCCCCGTCTCATTCACACACCC
+
6-,8A,,,+++,;+;C+8,:,,,,,,,,++8+8,,,;<@C,,9,,:C,9C+8,99,69,,,,,,,,+
@M01893:20:000000000-A7TT6:1:1101:18376:1528 2:N:0:3
CACTTACTGCAGACCTTGTTACAACACCTTAAGTAAAACTTTGTGCATATACCTCTCTTTAACTTTC
+
8-8A@,CE,C,,,,,;;,;;,<,,,,86C,,,,<,,,,;CC,,<,;,;,6,,,,,;;C@,,,,66@C
```

Interleaved FASTQ files alternate between R1 and R2 reads

Interleaved FASTQ files are uncommon: illumina sequencers generate distinct R1 and R1 FASTQ files



n50

https://github.com/PombertLab/Misc/blob/main/read_len_plot.py

Assessing read lengths

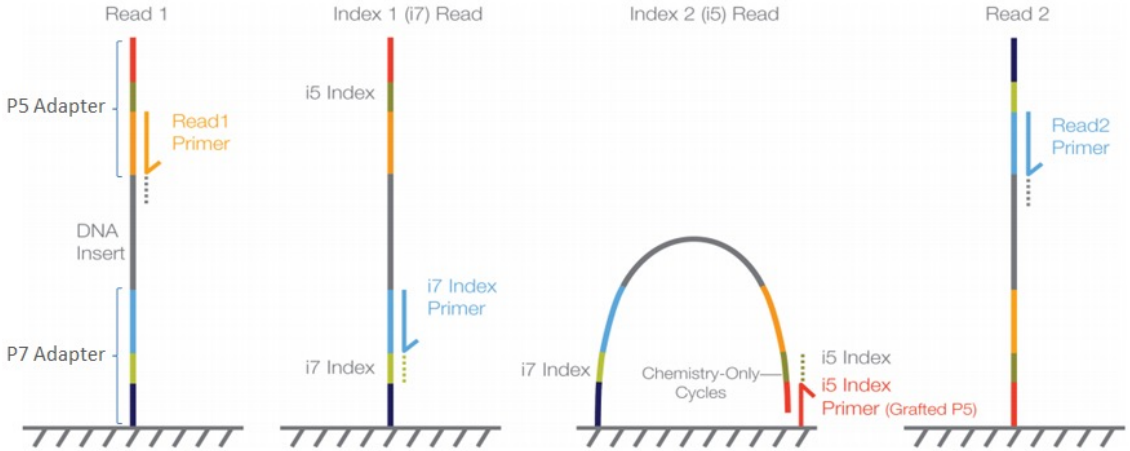
Important for long read platforms ## longer reads facilitate genome assembly

The key metric is called N50 ## i.*e.*, 50% of all bases are in reads of at least that size

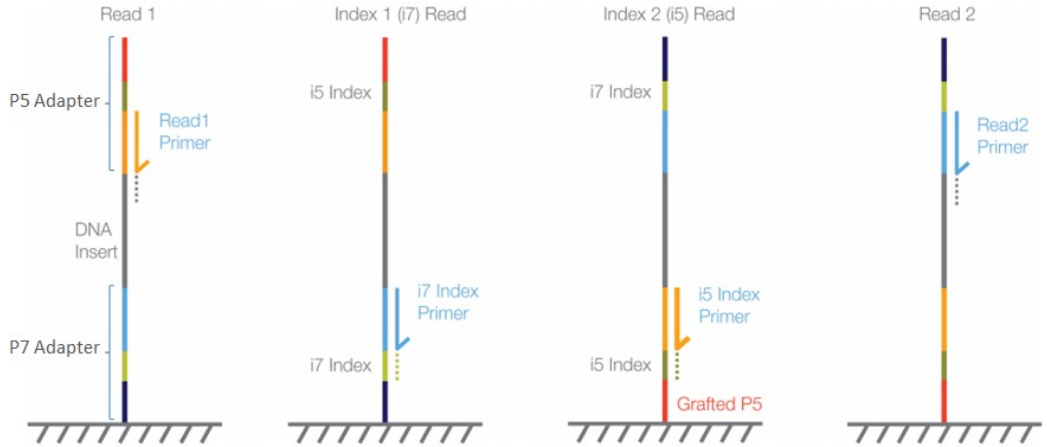
The standard assessment method is to plot the read length distribution

This can be plotted calculated with several tools

MiSeq, HiSeq 2000/2500 and NovaSeq paired-end flow cell



MiniSeq, NextSeq 500/550 and HiSeq 3000/4000 paired-end flow cell



https://support.illumina.com/bulletins/2016/04/adapter-trimming-why-are-adapter-sequences-trimmed-from-only-the--ends-of-reads.html

Read filtering

Sequencing reads can contain:

- Adapter sequences
- Low quality bases (locally and/or randomly distributed)
- Known and/or unknown contaminants

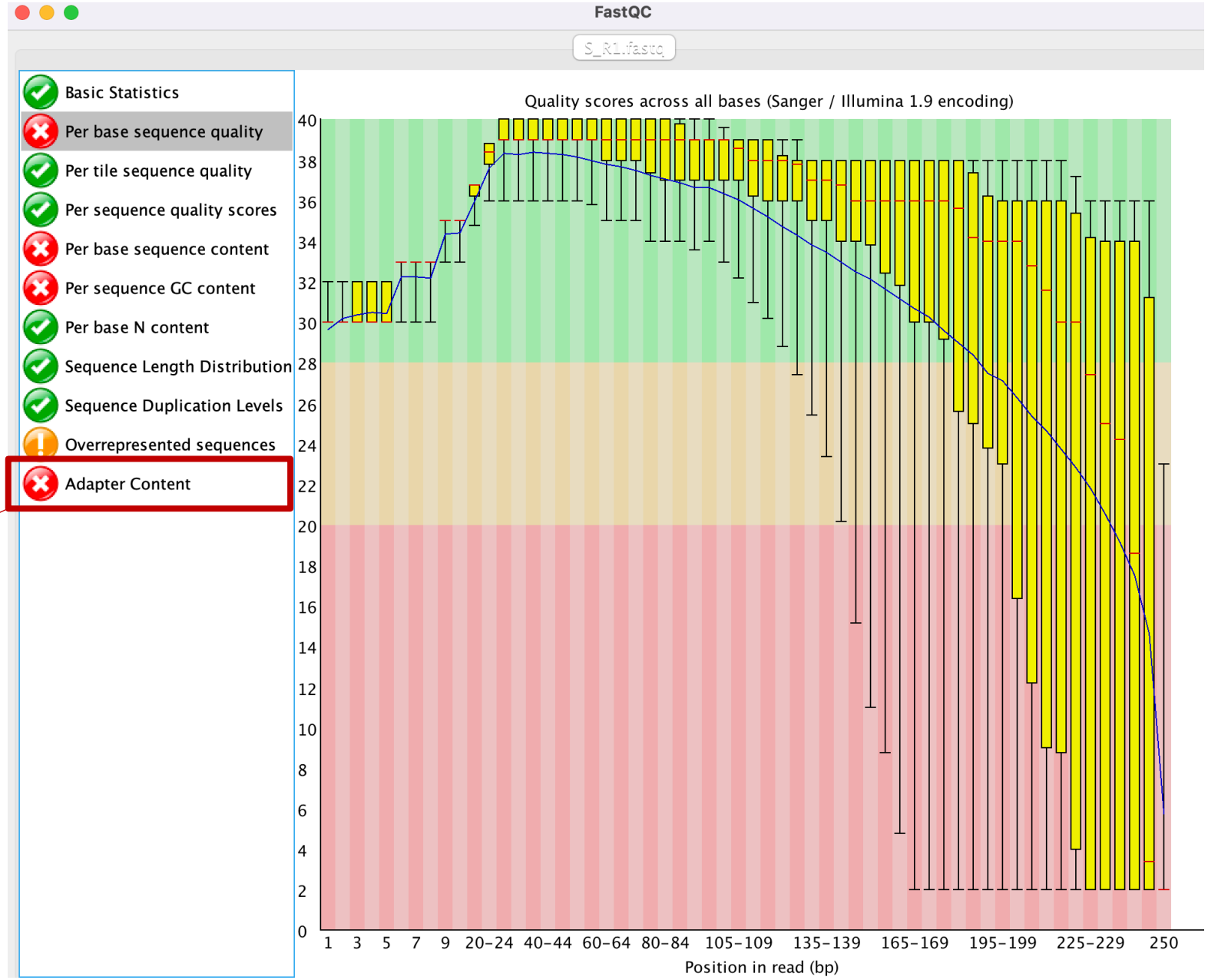
ALWAYS check your data **before** performing downstream analyses (garbage in => garbage out)

ALWAYS check your data **after** filtering

FastQC – Assessing quality scores in FASTQ files

- A simple, intuitive GUI program
- Also works from the command line
- Generates reports in HTML format

Important for Illumina data



http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

Exercise 01 – FASTQC; Is your data any good?

To use FastQC in GUI from Mozart, use the ssh -X switch. ## You can also use it in command line mode or install it on your laptop.

1) Test #1 – Illumina. S_R1.fastq (1st PE read sequenced in the machine).

2) Test #2 – Illumina. S_R2.fastq (2nd PE read). Notice the drop earlier on.

3) Test #3 – Illumina. Broad_R1.fastq and Broad_R2.fastq

4) Test #4 – PacBio. PacBio.fastq

5) Test #5 – Nanopore. Nanopore.fastq

As you can see, error profiles and quality scores can differ within and between technologies

Cutadapt Removes Adapter Sequences From High-Throughput Sequencing Reads
Martin M
EMBnet Journal. 2011. 17:10-12
DOI: 10.14806/ej.17.1.200

```
#!/usr/bin/bash
cutadapt \
  -j 10 \                                                    ## Number of threads to use
  -b CTGTCTCTTATACACATCT -B CTGTCTCTTATACACATCT \             ## Adapters to remove
  -m 100 \                                                   ## Minimum read length to keep
  -o Dlong_R1_100.nextera.cutadapt.fastq.gz \                ## R1 output file; can be GZIP compressed
  -p Dlong_R2_100.nextera.cutadapt.fastq.gz \                ## R2 output file; can be GZIP compressed
  Dlong_Nextera280bp_R1.fastq \                              ## R1 input file; can be  GZIP compressed
  Dlong_Nextera280bp_R2.fastq                                ## R2 input file; can be  GZIP compressed
```

Cutadapt – Remove adapters

https://cutadapt.readthedocs.io/en/stable/

```
-b CTGTCTCTTATACACATCT -B CTGTCTCTTATACACATCT      ## Nextera adapters

-b AGATCGGAAGAGCACACGTCTGAACTCCAGTCA \            ## Truseq adapters
-B AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT
```

https://support-docs.illumina.com/SHARE/AdapterSeq/illumina-adapter-sequences.pdf

Trimmomatic: a flexible trimmer for Illumina sequence data
Bolger AM *et al.*
Bioinformatics. 2014. 30(15):2114-20
DOI: 10.1093/bioinformatics/btu170

```
#!/usr/bin/bash
java -jar /opt/trimmomatic/trimmomatic-0.38.jar \                    ## Path to jar file
  PE \                                                               ## Pair-ends mode
  R1.input.fastq R2.input.fastq \                                    ## Input FASTQ files
  R1.output.fastq S1.output.fastq \                                  ## Output FASTQ R1 files
  R2.output.fastq S2.output.fastq \                                  ## Output FASTQ R2 files
  ILLUMINACLIP:/opt/trimmomatic/adapters/TruSeq3-PE.fa:2:30:10 \     ## Path to adapter sequences
  SLIDINGWINDOW:4:28 \                                               ## Min. quality score to keep
  MINLEN:100                                                         ## Minimum read length to keep
```

Trimmomatic – Trim PE files

http://www.usadellab.org/cms/?page=trimmomatic

https://github.com/timflutre/trimmomatic

Fastp: an ultra-fast all-in-one FASTQ preprocessor
Chen S, *et al*.
Bioinformatics. 2018. 34(17):i884-i890
DOI: 10.1093/bioinformatics/bty560

```
#!/usr/bin/bash
fastp \
  -w 10 \                    ## Number of threads (workers)
  -i R1.input.fastq \        ## R1 input file; can be GZIP compressed
  -I R2.input.fastq \        ## R2 input file; can be GZIP compressed
  -o R1.output.fastq.gz \    ## R1 output file; can be GZIP compressed
  -O R2.output.fastq.gz \    ## R2 output file; can be GZIP compressed
  -M 30 \                    ## Minimum quality score to keep
  -r \                       ## Sliding window from front to tail, drop if mean quality < threshold
  -l 100                     ## Minimum read length to keep
```

Fastp – Trim PE files

https://github.com/OpenGene/fastp

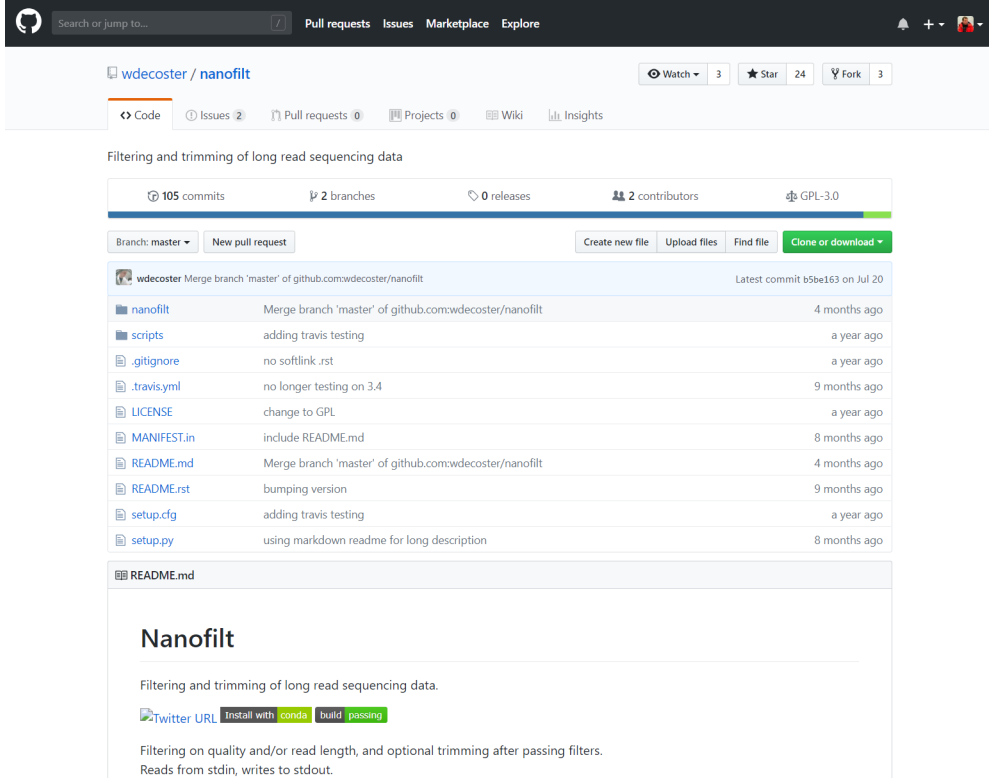
Built for speed

Tries to detect and remove adapter sequences automatically

Adapters can be specified with: --adapter_sequence=NNNN and --adapter_sequence_r2=NNNN

The PacBio suite filters reads and discards SMRTbell adapters automatically

```
## To install on your user account:
pip install --user nanofilt
```



NanoFilt – Trim nanopore reads

https://github.com/wdecoster/nanofilt

Example:

gunzip -c reads.fastq.gz | NanoFilt -q 10 -l 500 --headcrop 50 | gzip > trimmed-reads.fastq.gz

Exercise 02 – Removing garbage (Trimmomatic)

1) Trim the Broad_R1.fastq/Broad_R2.fastq files with trimmomatic (in PE mode). Save the outputs as R1.trimo.fastq/R2.trimo.fastq and S1.trimo.fastq/S2.trimo.fastq.

2) Look at the output with FASTQC. Notice how bad it is?

3) Try again by using SLIDINGWINDOW:4:15.

4) Look at it again with FASTQC

5) Notice the very short reads? The adapters left in the data? Let's keep paired-end reads that are at least 35 nt long and remove the TruSeq adapters. Assess the new dataset.

Exercise 03 – Removing garbage (Fastp)

1) Let's use Fastp to clean up the same dataset. Use the default parameters.

2) Look at the trimmed data with FASTQC. How is the quality? Were the adapters removed?

3) Use Fastp again, this time using a quality score of 30 (-M 30) with the sliding windows options set to -r. ## Cut afterwards if quality drops below threshold

4) Assess the output with FASTQC. Better?

```
#!/usr/bin/bash
split \
  -l 40000 \                          ## Split after X lines; 40000 lines = 10000 reads
  --additional-suffix=.fastq \        ## File extension to add
  --numeric-suffix=01 \               ## Autoincrement by number rather than xaa, xab…
  Sequence.R1.fastq \                 ## FASTQ input file to split
  Sequence.R1.                        ## Desired output file prefix
```

TIP – The split command

Divides text files in smaller ones

FASTQ files = one sequence per 4 lines

```
split -l 4000000 input.fastq     ## Will generate files with one million sequences each
```

III – Read mapping approaches

Assembly algorithms for next-generation sequencing data
Miller JR, Koren S, Sutton G
Genomics. 2010. 95(6):315-27
DOI: 10.1016/j.ygeno.2010.03.001

Sense from sequence reads: methods for alignment and assembly
Flicek P, Birney E
Nat Methods. 2009. 6(11 Suppl):S6-S12
DOI: 10.1038/nmeth.1376

Read-mapping *vs.* Assembly

Read-mapping => Aligns reads to reference sequence(s)

- Much lower memory footprint, faster
- Great for calling variants (SNVs or SNPs)

Assembly => Puzzles back pieces together

- Guided or *de novo* ## *i.e.* with or without a reference
- New genome? No close reference? The only way

Algorithms – Read-mapping

Smith-Waterman is a no go ## Aligning each query against the full reference(s)? Ouch...

Hashes (of arrays) ## BLAST, minimap2

Suffix trees & arrays ## Bowtie, BWA, MUMmer

-> Burrows-Wheeler Transform: most common form of suffix arrays

make_kmers.pl

```perl
#!/usr/bin/perl
use strict; use Getopt::Long qw(GetOptions);
my $name = 'make_kmers.pl';
my $version = 0.1;

my $usage = <<"OPTIONS";
NAME        $name
VERSION     $version
SYNOPSIS    Creates kmers from a string
COMMAND     $name -k 5 -s ATGATGACTGCACGTCAGCTGACTCGACTGATCGCTCGACGCTCGACTG

OPTIONS:
  -k    kmer length [Default: 5]
  -s    string to modify
OPTIONS
die "\n$usage\n" unless @ARGV;

my $k = 5; my $string;
GetOptions('k=i' => \$k, 's=s' => \$string);

print "Original string: $string\n";
my $len = length$string; my $knum = 0;
for (my $i = 0; $i < ($len - ($k - 1)); $i++){
    my $kmer = substr($string, $i, $k);
    $knum++;
    print "kmer # $knum = $kmer\n";
}
```

make_kmers.pl -k 5 -s ATGATGACTGCACG

```
Original string: ATGATGACTGCACG
kmer # 1  = ATGAT
kmer # 2  = TGATG
kmer # 3  = GATGA
kmer # 4  = ATGAC
kmer # 5  = TGACT
kmer # 6  = GACTG
kmer # 7  = ACTGC
kmer # 8  = CTGCA
kmer # 9  = TGCAC
kmer # 10 = GCACG
```

K-mers – What are they?

Substrings of sequences (DNA, RNA, proteins)

Sliding windows **of length K** ## aka tiny words, often used to build dictionaries

Used by many algorithms, implicitly or explicitly

Read mapping – Hash-based

A kmer based approach

Divide & conquer

Reads or genome can be hashed

Tolerate high polymorphism

1) Reference is %hashed into a table

Reference

```
ATGCTATGCTAAATAAAA
ATGCT
 TGCTA
  GCTAT
   CTATG
    TATGC
    ATGCT…
```

same as 1st one!

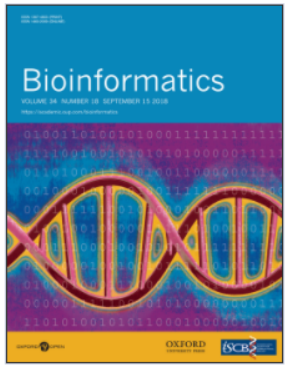
K-mers of length 5

Hash of arrays

Key	=>	Value(s)
ATGCT		0 , 5
TGCTA		1
GCTAT		2
CTATG		3
TATGC		4

Remember Perl: keys must be unique, values can be repeated

2) Query is %hashed too, pieces are searched against the table



Volume 34, Issue 18
15 September 2018

Minimap2: pairwise alignment for nucleotide sequences FREE

Heng Li

Bioinformatics, Volume 34, Issue 18, 15 September 2018, Pages 3094–3100,
https://doi.org/10.1093/bioinformatics/bty191
Published: 10 May 2018 **Article history**

PDF Split View Cite Permissions Share

Minimap2: pairwise alignment for nucleotide sequences
Li H
Bioinformatics. 2018. 34(18):3094-3100
DOI: 10.1093/bioinformatics/bty191

Minimap2 – Short/Long RM

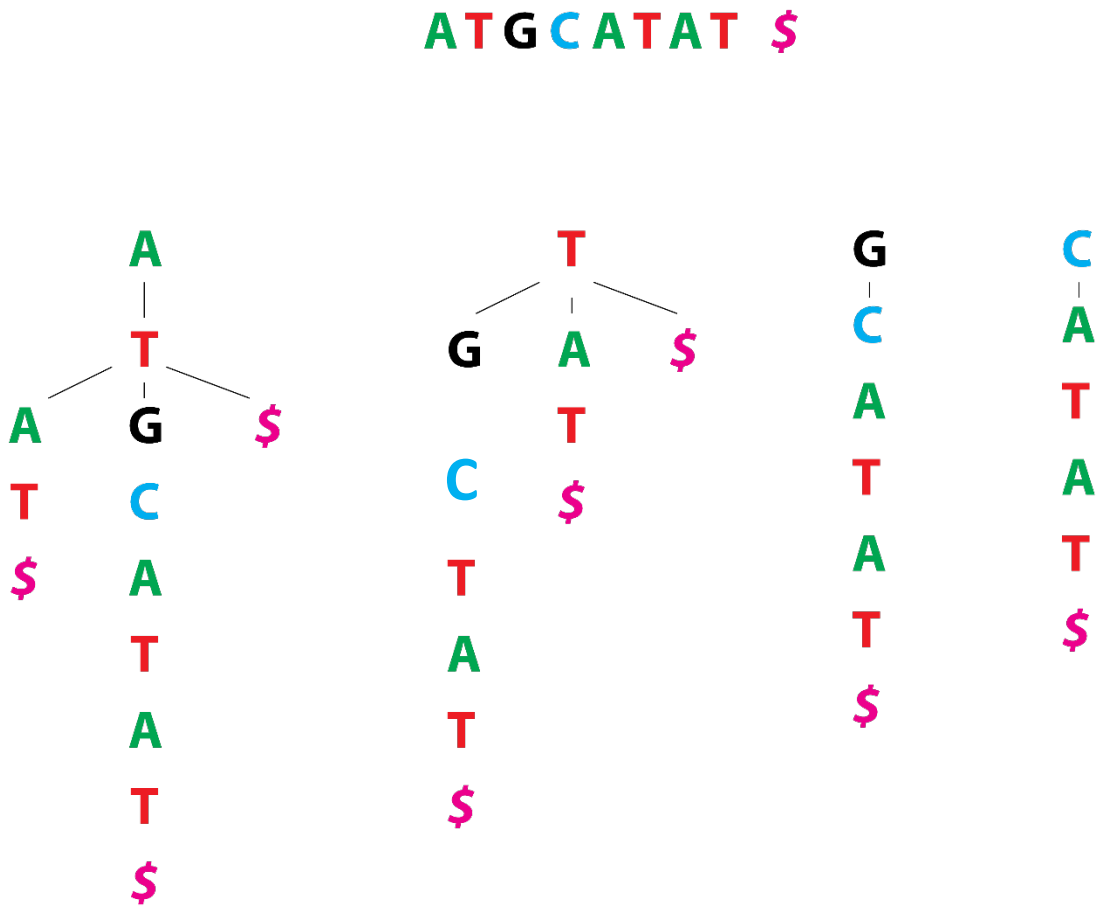
https://github.com/lh3/minimap2

Seed-chain-align ## Uses hash tables + minimizers to reduce memory usage

Allows split alignments ## Great for RNA or noisy data

Really fast ## Always good with large datasets

Suffixes are right-anchored substrings



A suffix tree (aka trie)

A suffix array

Watch Ben Langmead's excellent explanation: https://youtu.be/hLsrPsFHPcQ (1h10 min)

RM – Burrows-Wheeler transform

Create a rotating index that will be queried against

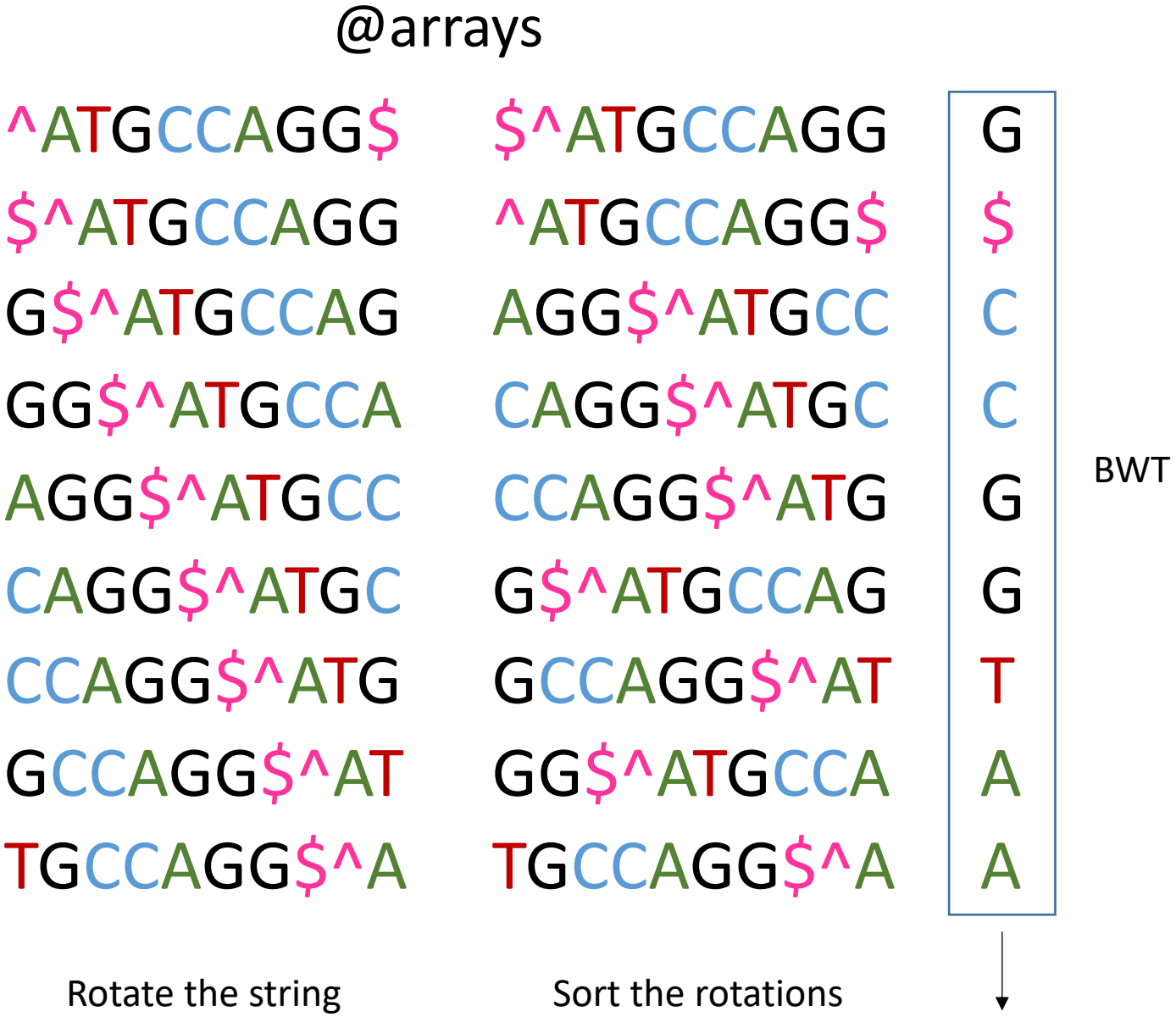
Usually fast

Require less RAM

Struggle with high polymorphism

Make sure to filter the reads by QVs before using a BWT

BWT are used for bzip2 data compression



Watch Ben Langmead's excellent explanation: https://youtu.be/4n7NPk5lwbI (37 min)

RAM ≅ genome size

Bowtie 2

Fast and sensitive read alignment

JOHNS HOPKINS UNIVERSITY

Bowtie 2 is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences. It is particularly good at aligning reads of about 50 up to 100s or 1,000s of characters, and particularly good at aligning to relatively long (e.g. mammalian) genomes. Bowtie 2 indexes the genome with an FM Index to keep its memory footprint small: for the human genome, its memory footprint is typically around 3.2 GB. Bowtie 2 supports gapped, local, and paired-end alignment modes.

OSI certified

Fast gapped-read alignment with Bowtie 2
Langmead B *et al.*
Nat Methods. 2012 Mar 4;9(4):357-9
DOI: 10.1038/nmeth.1923

Bowtie2 – Short read mapping

http://bowtie-bio.sourceforge.net/bowtie2/index.shtml

Burrows-Wheeler transform aligner ## One of the early good ones for illumina data

Involves two steps; index creation + read mapping:

1. bowtie2-build --threads $threads $INPUT $OUTPUT.bt2
2. bowtie2 -x $bt2-idx -p $threads -1 $R1.fastq -2 $R2.fastq -X $VALUE --al $MAPPING --un $UNMAP -S output.sam

#	Decimal	Description of first read
1	1	Read paired
2	2	Read mapped in proper pair
3	4	Read unmapped
4	8	Mate unmapped
5	16	Read reverse strand
6	32	Mate reverse strand
7	64	First in pair
8	128	Second in pair
9	256	Not primary alignment
10	512	Read fails platform/vendor quality checks
11	1024	Read is PCR or optical duplicate
12	2048	Supplementary alignment
Sum	77	

Decimal	Description of second read
1	Read paired
2	Read mapped in proper pair
4	Read unmapped
8	Mate unmapped
16	Read reverse strand
32	Mate reverse strand
64	First in pair
128	Second in pair
256	Not primary alignment
512	Read fails platform/vendor quality checks
1024	Read is PCR or optical duplicate
2048	Supplementary alignment
141	

Common flags*

One of the reads is unmapped:
73, 133, 89, 121, 165, 181, 101, 117, 153, 185, 69, 137

Both reads are unmapped:
77, 141

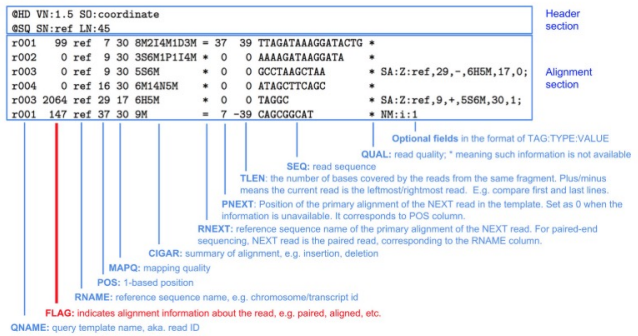
Mapped within the insert size and in correct orientation:
99, 147, 83, 163

Mapped within the insert size but in wrong orientation:
67, 131, 115, 179

Mapped uniquely, but with wrong insert size:
81, 161, 97, 145, 65, 129, 113, 177

* Collected from here

Instruction:



The Sequence alignment/map (SAM) format and SAMtools
Li H *et al.*
Bioinformatics. 2009. 25(16):2078-9
DOI: 10.1093/bioinformatics/btp352

https://www.samformat.info/sam-format-flag
https://broadinstitute.github.io/picard/explain-flags.html

SAM/BAM – Standard alignments

SAM (text); BAM (binary) alignment formats

http://www.htslib.org/

Uses flags to describes reads contained in the files (mapped and/or unmapped)

https://gist.github.com/darencard/72ddd9e6c08aaff5ff64ca512a04a6dd
https://github.com/PombertLab/SSRG/blob/master/bam2fastq.pl

Very useful when the mapping tool doesn't keep track of reads that do not map!

Bam2fq – Extracting reads from BAM files

SE

```
samtools bam2fq -F 4 input.bam > mapped.fastq          ## All mapped reads (illumina, PacBio, Nanopore)
samtools bam2fq -f 4 input.bam > unmapped.fastq        ## All unmapped reads (illumina, PacBio, Nanopore)
```

PE

```
samtools bam2fq -f 1 -F 12 -1 mapped_R1.fastq -2 mapped_R2.fastq input.bam    ## R1 + R2 mapped (illumina)
samtools bam2fq -f 12 -F 256 -1 unmap_R1.fastq -2 unmap_R2.fastq input.bam    ## R1 + R2 didn't map (illumina)
samtools bam2fq -f 8 -F 260 -1 XXX_R1.fastq -2 YYY_R2.fastq input.bam         ## R1 mapped, R2 didn't (illumina)
samtools bam2fq -f 4 -F 264 -1 ZZZ_R1.fastq -2 WWW_R2.fastq input.bam         ## R1 didn't, R2 mapped (illumina)
```

https://github.com/PombertLab/SSRG

Exercise 04a – Contaminant removal by read-mapping

A FASTQ dataset is contaminated by a known chloroplast genome. Let's use samtools's bam2fq and my read mapping pipeline to parse it out.

1) Let's run get_SNPs.pl (a Perl script) in read-mapping only mode (-rmo):

```
get_SNPs.pl \
  -fa MA_cpDNA.fasta \      ## Reference FASTA file(s)
  -pe1 *R1.fastq \          ## Paired end R1 FASTQ input file(s)
  -pe2 *R2.fastq \          ## Paired end R2 FASTQ input file(s)
  -mapper minimap2 \        ## Read mapper to use
  -preset sr \              ## Short reads preset for minimap2
  -rmo \                    ## Read-mapping only
  -bam                      ## Keep the BAM output file(s)
```

#	Decimal	Description of first read
1	1	Read paired
2	2	Read mapped in proper pair
3	4	Read unmapped
4	8	Mate unmapped
5	16	Read reverse strand
6	32	Mate reverse strand
7	64	First in pair
8	128	Second in pair
9	256	Not primary alignment
10	512	Read fails platform/vendor quality checks
11	1024	Read is PCR or optical duplicate
12	2048	Supplementary alignment
Sum	1	

#	Decimal	Description of read
1	1	Read paired
2	2	Read mapped in proper pair
3	4	Read unmapped
4	8	Mate unmapped
5	16	Read reverse strand
6	32	Mate reverse strand
7	64	First in pair
8	128	Second in pair
9	256	Not primary alignment
10	512	Read fails platform/vendor quality checks
11	1024	Read is PCR or optical duplicate
12	2048	Supplementary alignment
Sum	12	

2) Let's parse the BAM file with bam2fq:

```
samtools bam2fq \
  -f 1 \                    ## Keep PE reads that map
  -F 12 \                   ## Discard PE reads that do not map
  -1 map_R1.fastq \         ## FASTQ R1 output file of reads that do map
  -2 map_R2.fastq \         ## FASTQ R2 output file of reads that do map
  minimap2.BAM/MA_R1.fastq.MA_cpDNA.fasta.minimap2.bam   ## BAM file to query
```

https://github.com/PombertLab/SSRG

Exercise 04b – Contaminant removal by read-mapping

A FASTQ dataset is contaminated by a known chloroplast genome. Let's use samtools's bam2fq and my read mapping pipeline to parse it out.

#	Decimal	Description of read
1	1	Read paired
2	2	Read mapped in proper pair
3	4	Read unmapped
4	8	Mate unmapped
Sum	12	

#	Decimal	Description of read
9	256	Not primary alignment
10	512	Read fails platform/vendor quality checks
11	1024	Read is PCR or optical duplicate
12	2048	Supplementary alignment
Sum	256	

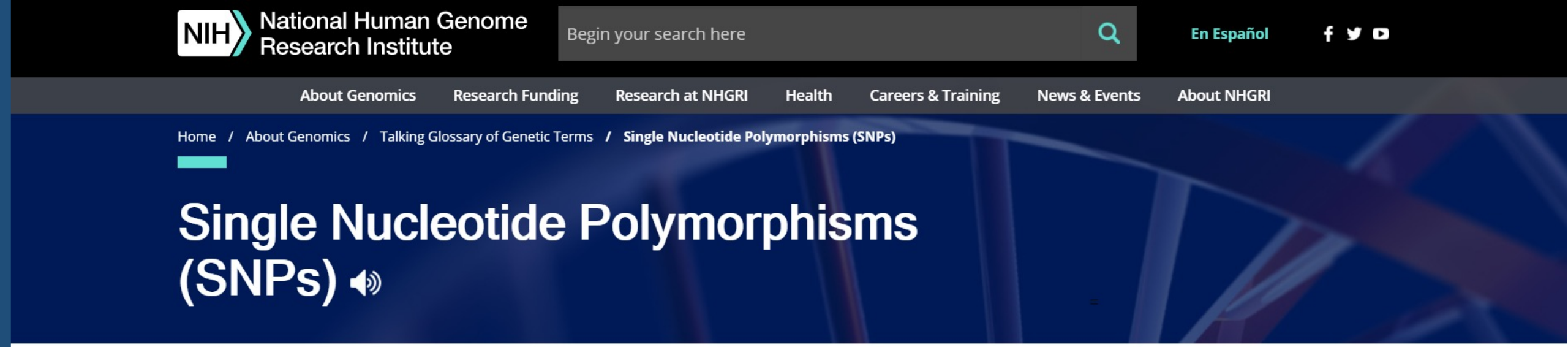
2) Let's parse the BAM file with bam2fq (cont'd):

```
samtools bam2fq \
  -f 12 \                                                    ## Keep PE reads that do not map
  -F 256 \                                                   ## Discard PE reads with secondary alignments
  -1 unmap_R1.fastq \                                        ## FASTQ R1 output file of reads that do not map
  -2 unmap_R2.fastq \                                        ## FASTQ R2 output file of reads that do not map
  minimap2.BAM/MA_R1.fastq.MA_cpDNA.fasta.minimap2.bam       ## BAM file to query
```

3) Let's check if it worked:

```
##  same get_SNPs.pl command as before
get_SNPs.pl -fa MA_cpDNA.fasta -pe1 *R1.fastq -pe2 *R2.fastq -mapper minimap2 -preset sr  -rmo -bam

cat minimap2.rmo.stats/*  ## Basic mapping statistics
```



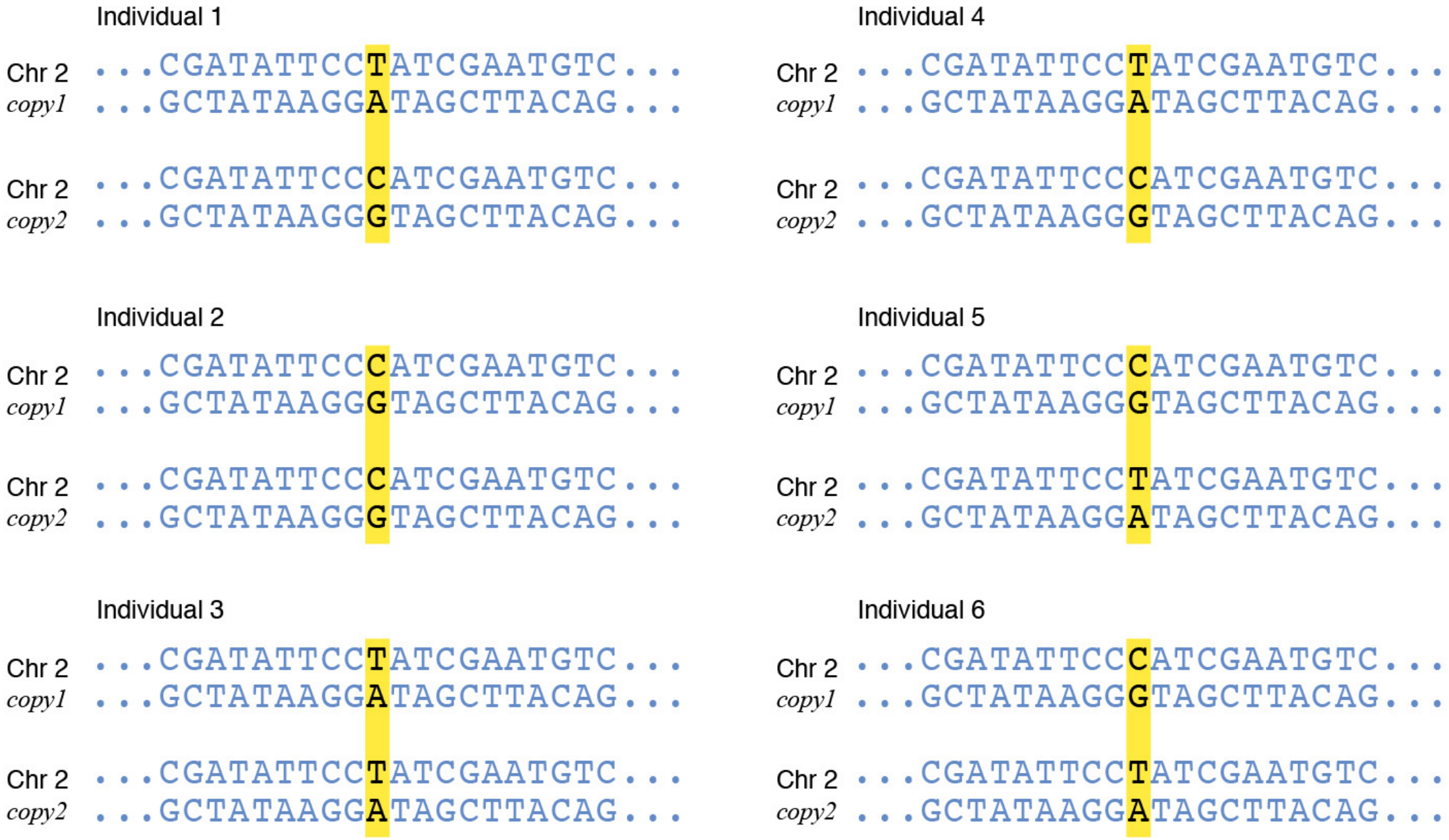
Variants – What are they?

Single **N**ucleotide **V**ariants (**SNV**s) ## Variations from a reference sequence

Single **N**ucleotide **P**olymorphisms (**SNP**s) ## Semantics; SNVs restricted within species

Insertion/**del**etions (indels)

SNVs/SNPs and short indels can be detected from mapped reads by variant callers



https://www.genome.gov/genetics-glossary/Single-Nucleotide-Polymorphisms

Exercise 05 – SNP calling (bowtie2 + VarScan2)

1) Let's run get_SNPs.pl again, this time with bowtie2 (read mapper) and with varscan2 (variant caller)

```
get_SNPs.pl -fa reference.fsa -pe1 *R1.fastq -pe2 *R2.fastq -mapper bowtie2 \
-var /opt/varscan/VarScan.v2.4.3.jar -type both  ## -var => java jar file to use; -type both => SNPs + indels
```

2) The standard output in the shell will give you a short description, something like:

```
34177 variant positions reported (33969 SNP, 208 indel)
```

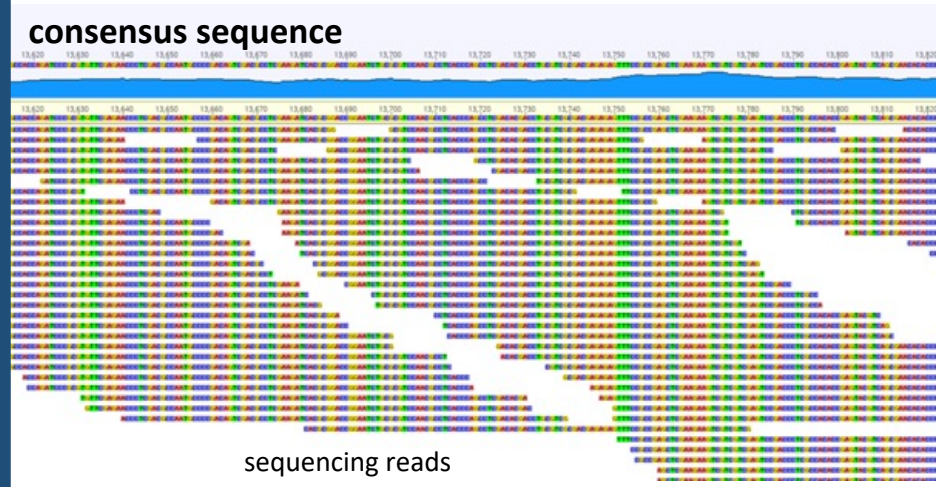
3) You can look at the output in **V**ariant **C**alling **F**ormat (**VCF**), yes that same file format we used for regular expressions:

```
less bowtie2.varscan2.VCFs/R1.fastq.reference.fsa.bowtie2.both.vcf
```

4) Now, imagine doing all that manually without a pipeline!

```
## You can look at the script: https://github.com/PombertLab/SSRG/blob/master/get_SNPs.pl
```

IV – Genome assembly



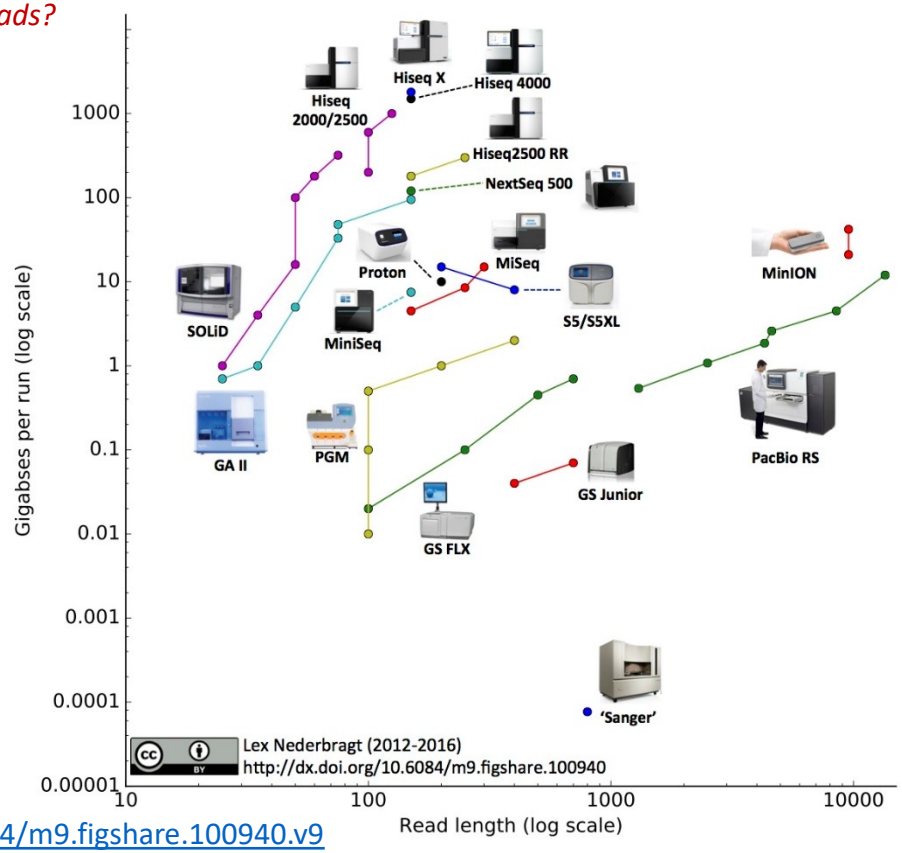
Short but accurate or long and noisy sequencing reads?

The shotgun puzzle

Genomes are usually too large to be sequenced in one pass:

1. We must sequence them piece by piece
2. These pieces must be put back together computationally
3. Requires **redundancy** and **random** overlaps

Different sequencing platforms? Different pros and cons...



DOI: 10.6084/m9.figshare.100940.v9

Published: 27 March 2012

***De novo* genome assembly: what every biologist should know**

Download PDF

Monya Baker

Nature Methods **9**, 333–337(2012) | Cite this article

12k Accesses | **95** Citations | **139** Altmetric | Metrics

As more genomes are assembled from scratch, scientists are struggling to assess and improve their quality.

DOI: 10.1038/nmeth.1935

F1000Research F1000Research 2018, 7(ELIXIR):148 Last updated: 09 APR 2020

Check for updates

OPINION ARTICLE

Ten steps to get started in Genome Assembly and Annotation [version 1; peer review: 2 approved]

Victoria Dominguez Del Angel [1], Erik Hjerde [2], Lieven Sterck [3,4], Salvadors Capella-Gutierrez [5,6], Cederic Notredame[7,8], Olga Vinnere Pettersson[9], Joelle Amselem [10], Laurent Bouri [1], Stephanie Bocs [11-13], Christophe Klopp [14], Jean-Francois Gibrat [1,15], Anna Vlasova [8], Brane L. Leskosek[16], Lucile Soler[17], Mahesh Binzer-Panchal [17], Henrik Lantz [17]

DOI: 10.12688/f1000research.13598.1

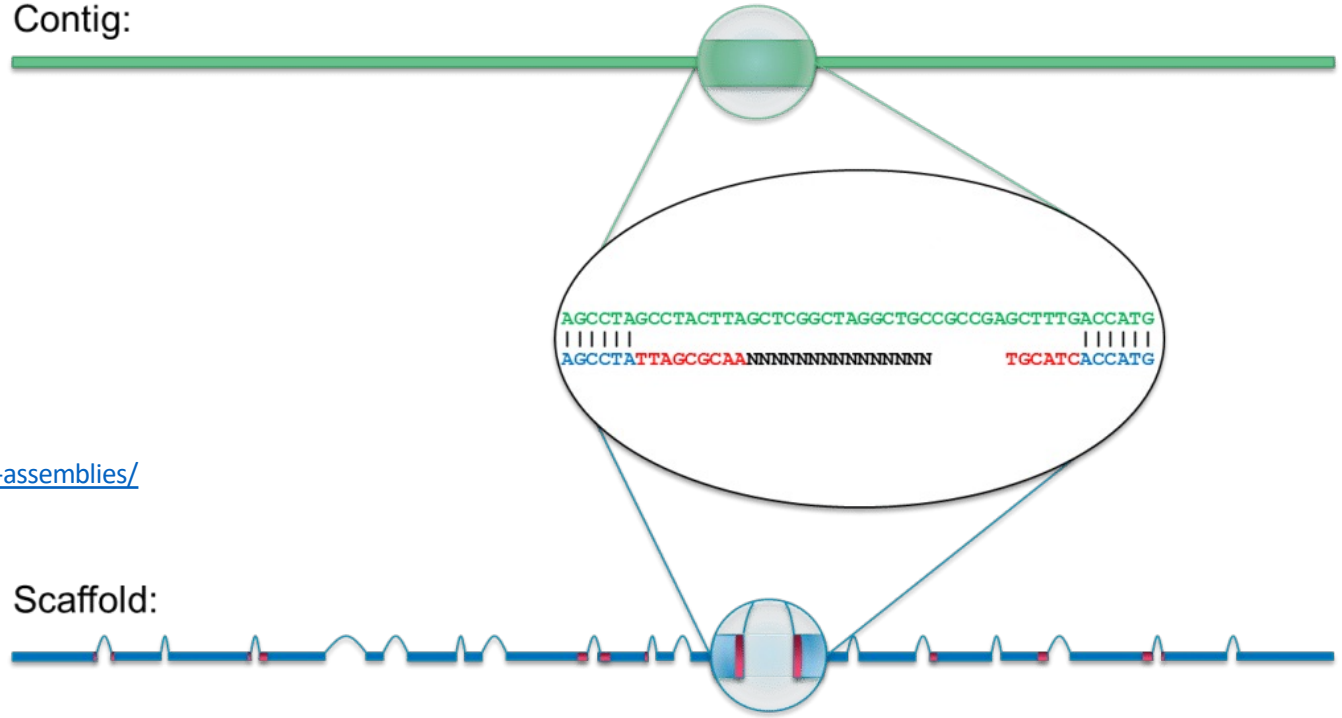
Assemblies – *De novo vs.* guided

De novo	**Anew**, *i.e.* without prior knowledge and/or from scratch
Guided	Using a reference or long reads as scaffold
Meta	## Metagenome/metatranscriptome assemblies:
	## Assemblies containing material from several sources
	## Common with environmental samples

Genomes vs. GenNNNes: The Difference between Contigs and Scaffolds in Genome Assemblies

PACBIO®

https://www.pacb.com/blog/genomes-vs-gennnnes-difference-contigs-scaffolds-genome-assemblies/



Contigs *vs.* scaffolds

Contigs -> **contiguous** sequences derived from assemblies

Scaffolds -> **non-contiguous** sequences linked by spatial information

Links in scaffolds are often represented by NNNs; ATGCCNNNNTCCC

GenBank doesn't accept scaffolds unless you have solid evidence

Genomic assembly software – non-exhaustive

OSS (open source; free)

Ray	https://github.com/sebhtml/ray
SPAdes	http://cab.spbu.ru/software/spades/
Canu	https://github.com/marbl/canu
Flye	https://github.com/fenderglass/Flye
Ra	https://github.com/lbcb-sci/ra
Shasta	https://github.com/chanzuckerberg/shasta
Unicycler	https://github.com/rrwick/Unicycler
MaSuRCA	https://github.com/alekseyzimin/masurca

Commercial (usually very expensive)

Geneious	http://www.geneious.com/
Sequencher	http://www.genecodes.com/
SeqMan NGen DNAStar	http://www.dnastar.com/
CLC Assembly Cell	http://www.clcbio.com/

^ Transcriptomics *vs.* Genomics ^

Transcriptomes – RNA ne DNA!

Sequencing depth is not uniform (RNAs are expressed at different levels)

Genes/exons can be short

Spliced-leaders, targeting signals, poly(A) tails, isoforms…

DNA assemblers often perform poorly with mRNAs

Volume 8, Issue 5
May 2019

De novo transcriptome assembly: A comprehensive cross-species comparison of short-read RNA-Seq assemblers

Martin Hölzer ✉, Manja Marz

GigaScience, Volume 8, Issue 5, May 2019, giz039,
https://doi.org/10.1093/gigascience/giz039
Published: 11 May 2019 **Article history** ▾

***De novo* transcriptome assembly: A comprehensive cross-species comparison of short-read RNA-Seq assemblers**
Hölzer M, Marz M
Gigascience. 2019 May 1;8(5):giz039. DOI: 10.1093/gigascience/giz039

RNA-Seq assemblers

Trinity	https://github.com/trinityrnaseq/trinityrnaseq/wiki
Oases	http://www.ebi.ac.uk/~zerbino/oases/
Trans-ABySS	http://www.bcgsc.ca/platform/bioinfo/software/trans-abyss
Cufflinks	http://cufflinks.cbcb.umd.edu/
CLASS	http://sourceforge.net/projects/splicebox/

Approximation algorithms for the shortest common superstring problem
Turner JS
Information and Computation. 1989. 83(1):1-20
DOI: 10.1016/0890-5401(89)90044-8

Approximation Algorithms for the Shortest Common Superstring Problem

JONATHAN S. TURNER*

Computer Science Department, Washington University, St. Louis, Missouri 63130

The object of the shortest common superstring problem (SCS) is to find the shortest possible string that contains every string in a given set as substrings. As the problem is NP-complete, approximation algorithms are of interest. The value of an aproximate solution to SCS is normally taken to be its length, and we seek algorithms that make the length as small as possible. A different measure is given by the sum of the overlaps between consecutive strings in a candidate solution. When considering this measure, the object is to find solutions that make it as large as possible. These two measures offer different ways of viewing the problem. While the two viewpoints are equivalent with respect to optimal solutions, they differ with respect to approximate solutions. We describe several approximation algorithms that produce solutions that are always within a factor of two of optimum with respect to the overlap measure. We also describe an efficient implementation of one of these, using McCreight's compact suffix tree construction algorithm. The worst-case running time is $O(m \log n)$ for small alphabets, where m is the sum of the lengths of all the strings in the set and n is the number of strings. For large alphabets, the algorithm can be implemented in $O(m \log m)$ time by using Sleator and Tarjan's lexicographic splay tree data structure.

The shortest common superstring

A general assumption

The assumption? The shortest possible string is the real one

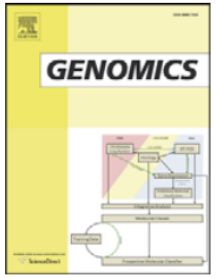
i.e. More overlaps? more positive matches? Higher score...

Doesn't hold in the presence of repeats

Contents lists available at ScienceDirect

Genomics

journal homepage: www.elsevier.com/locate/ygeno



Review

Assembly algorithms for next-generation sequencing data

Jason R. Miller *, Sergey Koren, Granger Sutton

J. Craig Venter Institute, 9704 Medical Center Drive, Rockville MD 20850-3343, USA

A good introduction on the topic

Assembly algorithms for next-generation sequencing data

Miller JR, Koren S, Sutton G

Genomics. 2010 Jun;95(6):315-27.

DOI: 10.1016/j.ygeno.2010.03.001

Algorithms – Assemblies

Overlap-**L**ayout **C**onsensus (OLC)

de **B**ruijn **g**raphs (DBG)

Seed extension

Overlap-layout consensus (OLC)

Use k-mers implicitly to discard impossible matches

Assign scores to overlaps, can use QVs

Possible outcomes of A and B

1.

Same direction

2.

Overlap inwardly

4.

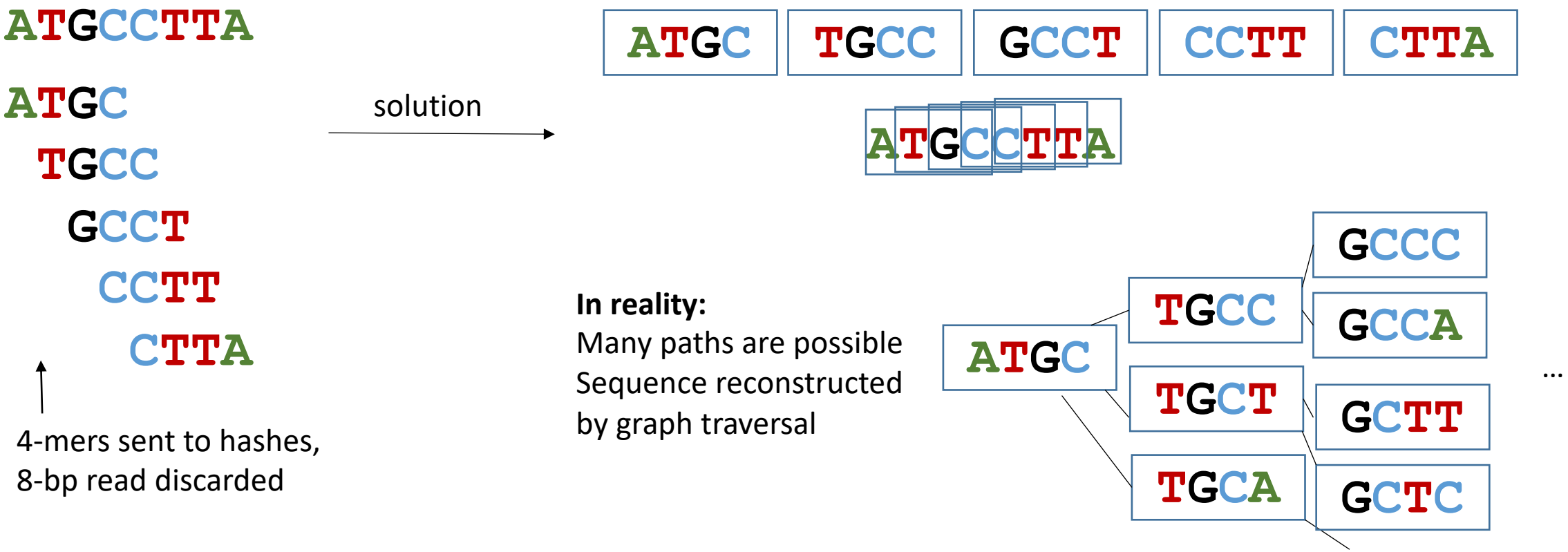
No overlap

3.

Overlap outwardly

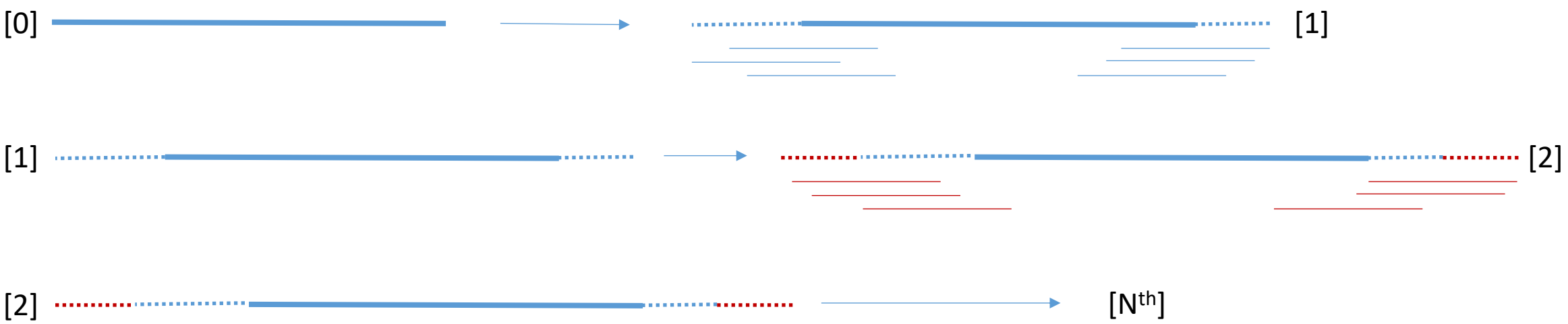
de Bruijn graphs (DBG)

Use k-mers explicitly for reconstruction, reads and QVs are discarded, struggle with repeats



Seed extension (*in silico* chromosome walking)

Start from one or more seeds [Gen 0], extend using read mapping [Gen 1], use previous generation as seed [Generation 2+] ## An iterative process



Exercise 06 – de Bruijn graph (SPAdes)

1) Start a remote screen called SPAdes ## Allows you to disconnect

```
screen -S SPAdes
```

2) Run SPAdes using 4 kmers as described below:

```
spades.py \
  -k 21,33,55,77 \          ## Kmers to evaluate
  --careful \               ## Avoid unsafe constructs; minimize chimeras
  --pe1-1 cp_R1.fastq \     ## Paired end R1 FASTQ input file
  --pe1-2 cp_R2.fastq \     ## Paired end R2 FASTQ input file
  -t 4 \                    ## Number of threads
  -o spades                 ## Output directory
```

3) Detach from the screen:

```
hold ctrl + A + D     ## To re-attach: screen -r SPAdes (or PID)
```

SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing
Bankevich A, Nurk S, Antipov D, Gurevich AA, Dvorkin M, Kulikov AS, et al.
J Comput Biol. 2012. 19(5):455-477; DOI: 10.1089/cmb.2012.0021

Exercise 07 – Long read assembly (Canu)

Canu is an OLC assembler derived from the Celera assembler and optimized for long reads.

1) Open a screen called Nanopore: screen -S Nanopore
2) Run canu on the Nanopore.fastq file:

```
canu \
  -p Canu \                    ## Assembly prefix
  -d Canu \                    ## Output directory
  -maxThreads=4 \              ## Number of threads
  -genomeSize=2.5m \           ## Here, we assume that the genome is small
  -nanopore-raw *.fastq        ## The nanopore reads are not corrected
```

https://github.com/marbl/canu

Exercise 08 – Long read assembly (Flye)

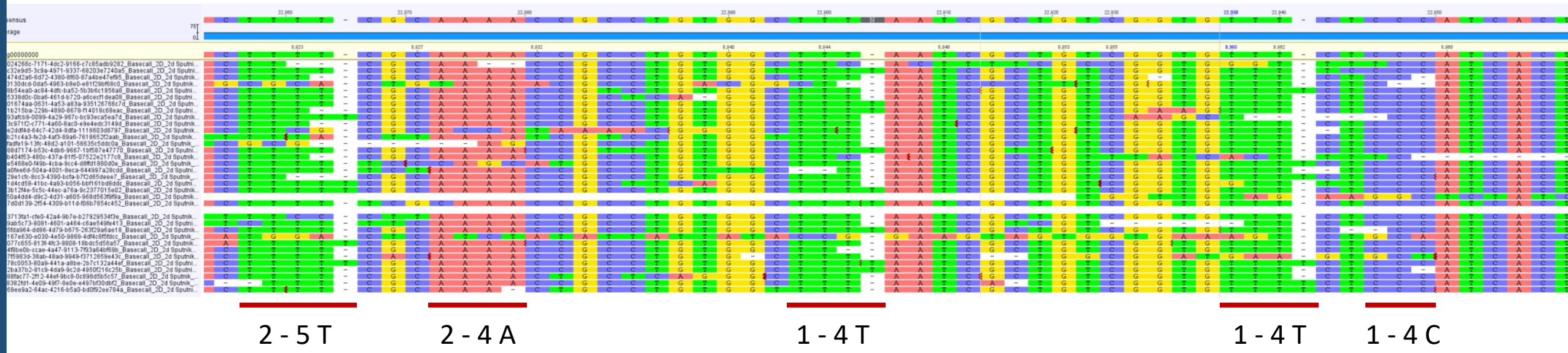
An assembler released in 2019. It appears very promising. Requires
the minimap2 read-mapping tool.

1) Open a screen called Flye: screen -S Flye
2) Run Flye on the Nanopore.fastq file using the command line switches:

```
flye \
  -t 8 \                              ## Number of threads
  --nano-raw Nanopore.fastq \         ## Type of FASTQ reads
  -g 2.5m \                           ## Expected genome size
  --asm-coverage 25 \                 ## Use longest reads until depth of 25X
  -o flye_2.5M_25x                    ## Desired output folder
```

To reduce computation time, the actual depth for the dataset in this exercise is low at ~25X

https://github.com/fenderglass/Flye

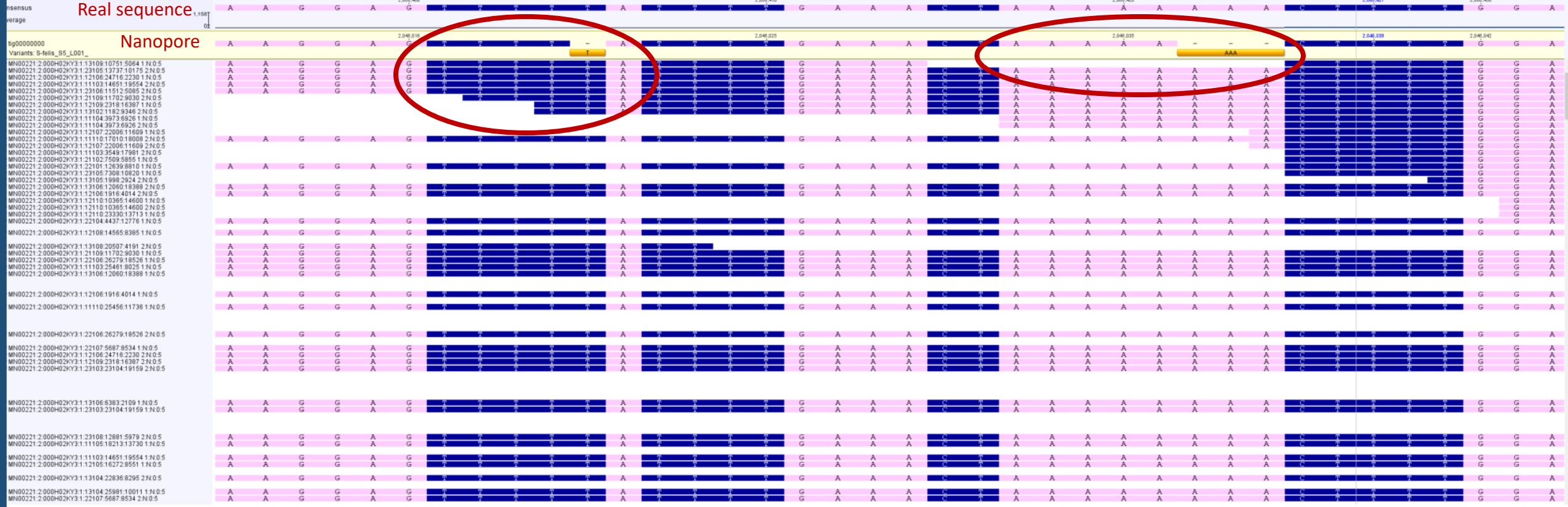


Homopolymers

Nanopore struggles with homopolymers

This is a systematic error

Other basecalling errors are stochastic



Vs. Illumina

Illumina uses reversible terminators

Virtually immune to homopolymer artefacts

Can be used to polish Nanopore assemblies

		Overall accuracy	Point mutations	Indels	Errors /10 kb
Nanopore *vs.* illumina	BC03	99.67%	81	8,667	34
	BC04	99.67%	35	8,311	33
	BC06	99.55%	279	11,321	45
	BC07	99.64%	22	10,325	40
After Pilon polishing	~ 750 to 1100 differences; ~ 3 to 5 errors / 10 kb				
	i.e. 99.95 to 99.7% accuracy				

Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement

Walker BJ, Abeel T, Shea T, Priest M, Abouelliel A, Sakthikumar S, et al.

PLoS One. 2014. 9(11):e112963. DOI: 10.1371/journal.pone.0112963

Exercise 9 – Hybrid assembler (Unicycler)

1) Open a screen called unicycler: screen -S unicycler
2) Run your unicycler assembly using 8 threads:

```
unicycler \
  -t 8 \                                          ## Number of threads
  -1 Illumina_R1.fastq.gz \                       ## Illumina paired end R1 FASTQ input file
  -2 Illumina_R2.fastq.gz \                       ## Illumina paired end R2 FASTQ input file
  --pilon_path /opt/pilon/pilon-1.22.jar \        ## Pilon java jar file to use
  -l nanopore.fastq.gz \                          ## Nanopore FASTQ input file
  -o hybrid_assembly                              ## Output directory
```

3) Detach from the screen:
   hold ctrl + A + D
4) To reattach:
   screen -r unicycler (or PID)

https://github.com/rrwick/Unicycler

Assemblies: the good, the bad, the ugly
Birney E
Nat Methods. 2011 Jan;8(1):59-60
DOI: 10.1038/nmeth0111-59

Published: 29 December 2010

Assemblies: the good, the bad, the ugly

Ewan Birney

Nature Methods **8**, 59–60(2011) | Cite this article

The low cost of short-read sequencing has motivated the development of *de novo* assemblies from only short-read data; impressively, assemblies for large mammalian genomes are now available. However, this is still a developing field, and these *de novo* assemblies have many artifacts, as do all *de novo* assemblies.

"If there is one message to remember [...], it is to not fully trust any assembly." - E. Birney. *Nature Methods*. 8, 59 (2011)

Verify your assemblies:

- Try different algorithms & implementations
- Got repeats? Watch out for chimeras!

Repeated elements break algorithms, imagine telomeres!

QUAST

Quality Assessment Tool for Genome Assemblies by CAB

Installation | QUAST | MetaQUAST | QUAST-LG | Icarus | Web interface | Manual | Publications

Quast

QUAST evaluates genome assemblies.

QUAST works both with and without a reference genome.
The tool accepts multiple assemblies, thus is suitable for comparison.

Versatile genome assembly evaluation with QUAST-LG
Mikheenko A, Prjibelski A, Saveliev V, Antipov D, Gurevich A
Bioinformatics. 2018 Jul 1;34(13):i142-i150.
DOI: 10.1093/bioinformatics/bty266

Comparing assemblies

Not all assemblies are equally good

There is no absolute way to tell which assembly is better; it depends on the context

A few metrics are in place to help assess them

A typical QUAST report

Report Example

E. coli single-cell assemblies

Aligned to "e.coli_reference" | 4 639 675 bp | 50.79 % G+C | 884 operons

All statistics are based on contigs of size >= 500 bp, unless otherwise noted (e.g., "# contigs (>= 0 bp)" and "Total length (>= 0 bp)" include all contigs).

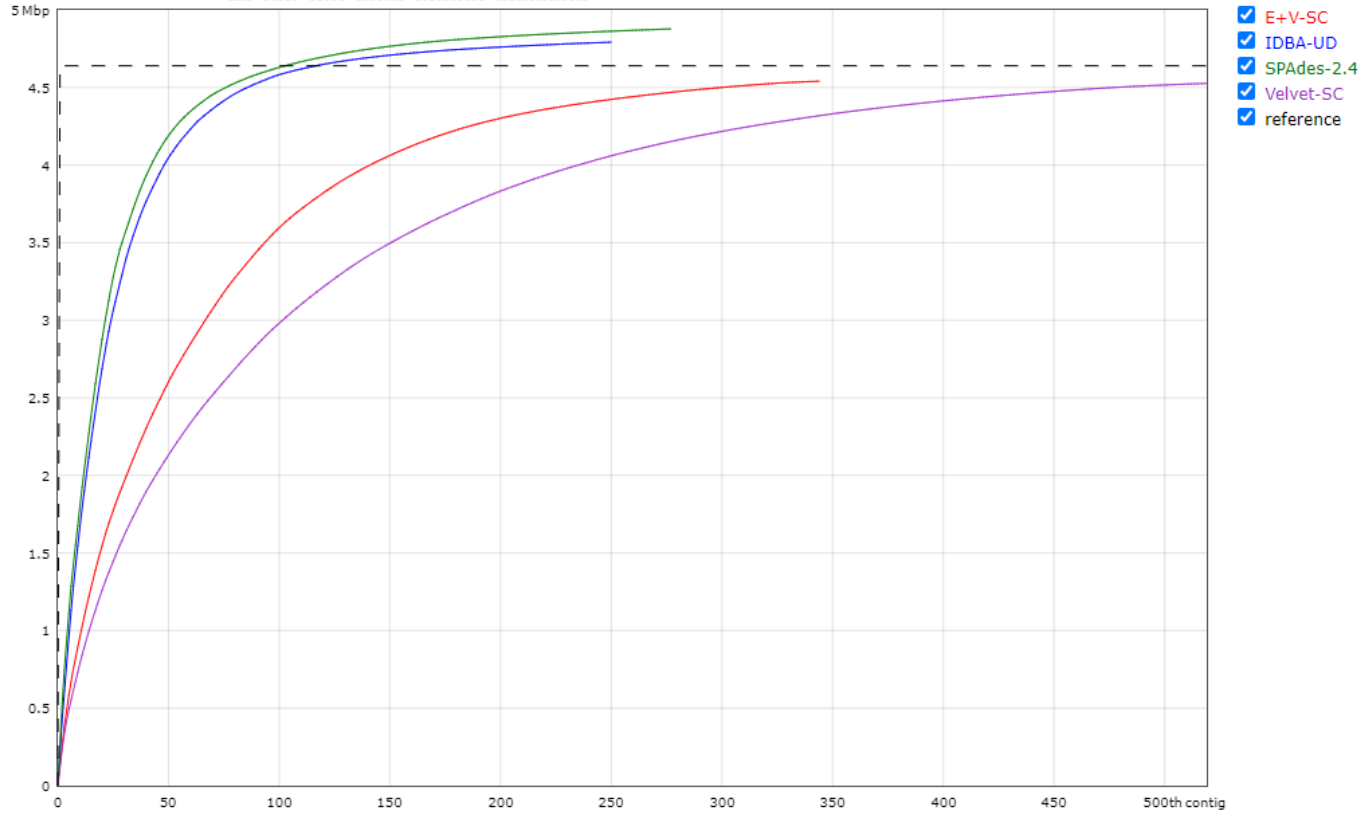
Worst Median Best ☑ Show heatmap

Statistics without reference	E+V-SC	IDBA-UD	SPAdes-2.4	Velvet-SC
# contigs	344	250	277	519
Largest contig	132 865	224 018	269 177	121 367
Total length	4 540 286	4 791 744	4 877 521	4 526 656
N50	33 616	96 947	106 927	20 445
Misassemblies				
# misassemblies	2	9	2	2
Misassembled contigs length	23 485	66 335	26 551	22 359
Mismatches				
# mismatches per 100 kbp	2.26	3.65	5.06	1.77
# indels per 100 kbp	0.7	0.2	0.7	0.92
# N's per 100 kbp	0	0	4.86	0
Genome statistics				
Genome fraction (%)	91.727	94.943	95.759	91.43
Duplication ratio	1.001	1.001	1.004	1.002
# genes	3767 + 160 part	4026 + 80 part	4046 + 102 part	3630 + 288 part
# operons	723 + 87 part	802 + 40 part	809 + 48 part	650 + 158 part
NGA50	32 051	96 947	110 539	19 791
Predicted genes				
# predicted genes (unique)	4258	4394	4417	4331
# predicted genes (>= 0 bp)	4258	4394	4490	4331
# predicted genes (>= 300 bp)	3643	3736	3784	3666
# predicted genes (>= 1500 bp)	524	559	559	515
# predicted genes (>= 3000 bp)	44	49	48	39

Extended report

http://cab.cc.spbu.ru/quast/

Plots: Cumulative length Nx NAx NGx NGAx Operons GC content



Contigs are ordered from largest (contig #1) to smallest.

What are N50s ?

N50 values are estimates of contiguity

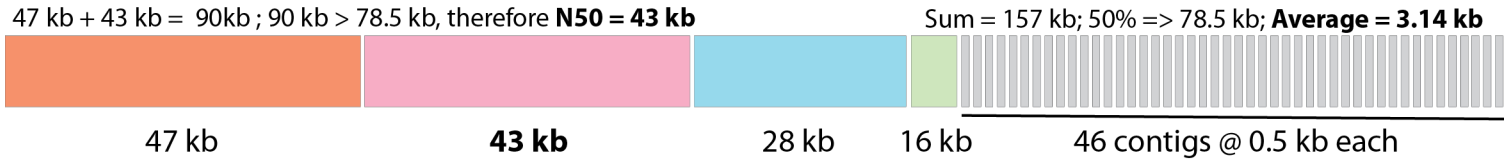
They give a quick overview of the size distribution of the contigs

As a rule of thumb, we try to optimize the N50s

N50s are **NOT** a measure of accuracy

Over-optimization can lead to wrong assemblies (e.g. chimeras)

Example



Contig lengths

len	**47 kb**	+	**43 kb**	+	**28 kb**	+	**16 kb**	+	**0.5 kb**	+	**0.5 kb**	+ ...
sum		**90 kb**		**118 kb**		**134 kb**		**134.5 kb**		**135 kb**		

N50 = 43 kb

Total = 157 kb
Total/2 = 78.5 kb

Usually, the higher the N50 the better

Exercise 10 – QUAST

1) Run quast on the Contigs folder

```
quast.py \                    ## Requires python3
  --min-contig 500 \          ## Minimum size of contigs to evaluate
  Contigs/*.fasta \           ## Assemblies to compare
  -o EX10_quast               ## Output directory
```

2) Open the html report with chrome requires ssh -X switch)

```
firefox EX10_quast/report.html
```

3) Look at the metrics

QUAST: quality assessment tool for genome assemblies
Gurevich A, Saveliev V, Vyahhi N, Tesler G
Bioinformatics. 2013. 29(8):1072-1075
DOI: 10.1093/bioinformatics/btt086

Filtering out contaminants

Contaminants can be filtered out post-assembly by:

- Homology searches (e.g. BLAST) ## We can search assembled contigs against databases
- Sequencing depth ## Contaminants may not be present in the same abundance as our target(s)
- GC content ## Genomes may differ significantly in % of GC and AT nucleotides

Can be determined by read mapping

Can be determined with a simple Perl script

Exercise 11a – Filter assembly contaminants by BLAST (with taxonomy)

The genome assembly Exercise_11.fasta contains contaminant(s). We are looking for *Streptococcus pneumoniae*.

1) Perform a taxonomized BLASTN search against the ref_prok_rep_genomes database to assess the origin of each contig. You can do it using the commands you learned or by using the provided runTaxonomizedBLAST.pl:

```
runTaxonomizedBLAST.pl \
 -t 8 \                          ## Number of threads to use
 -p blastn \                     ## Program to use: blastn, blastp, blastx…
 -a megablast \                  ## Algorithm to use for blastn
 -db ref_prok_rep_genomes \      ## Database to query: representative bacterial genomes
 -q Exercise_11.fasta \          ## FASTA file to query
 -e 1e-30                        ## Evalue cutoff
 -c 1                            ## Culling limit
```

Exercise 11a – Filter assembly contaminants by BLAST (with taxonomy)

The genome assembly Exercise_11.fasta contains contaminant(s). We are looking for *Streptococcus pneumoniae*.

2) Look at the content of the output file, *e.g.*: less Exercise_11.fasta.blastn.6

3) From this output, we can identify which contigs are from our genome(s) of interest. We could extract the information from the files manually, but let's use the provided parseTaxonomizedBLAST.pl script instead:

```
parseTaxonomizedBLAST.pl \
  -v \                                                                  ## Verbose switch
  -b Exercise_11.fasta.blastn.6 \                                       ## BLAST result
  -f Exercise_11.fasta \                                                ## FASTA used as query
  -n "Streptococcus pneumoniae" Streptococcus pneumoniae  \             ## Names (regex) to search
  -c sscinames \                                                        ## Column; scientific names
  -o S_pneumoniae.fasta                                                 ## Output of corresponding ctgs
```

Old stuff

```
#!/usr/bin/bash
sickle pe \                          ## pe => paired ends mode
   -q 32 \                           ## Minimum quality score
    -t sanger \                      ## Encoding type; Sanger = ASCII33
   -l 35 \                           ## Minimum read length
   -f Broad_R1.fastq \               ## Paired end R1 FASTQ input file
   -r Broad_R2.fastq \               ## Paired end R2 FASTQ input file
   -o Broad_R1.sickle.q32.fastq \    ## Filtered R1 FASTQ output file
   -p Broad_R2.sickle.q32.fastq \    ## Filtered R2 FASTQ output file
   -s Broad_singles.q32.fastq        ## Reads passing filter but without mates
```

Sickle – Trim paired-ends files

https://github.com/ucdavis-bioinformatics/sickle

sickle pe -l 150 -f S_R1.fastq -r S_R2.fastq -t sanger -o S_R1.sickle.fastq -p S_R2.sickle.fastq -s singles.sickles.fastq

Works by sliding windows ## pe, paired ends; se, single ends

When working in pairs, dumps singletons into a file

Scaffolding errors

Contigs are linked in scaffolds by NNNs

Numbers of NNNs are estimates, perfect distances rarely happen

Sometimes errors result from wrong distance estimates

Overestimations are annoying, but deletions require read-mapping + visual inspection to fix

Perfect assembly

```
AGCTGTCTGTTTTCTGTAGCTCGTATTACATATCGATGGA
AGCTGTCTGTTTTCTGTAGCTCGTA
AGCTGTCTGTTTTCTGTAGCTCG
          GTAGCTCGTATTACATATCGATGGA
            AGCTCGTATTACATATCGATGGA
```

Scaffold overestimation False repeats, relatively easy to fix

```
AGCTGTCTGTTTTCTGTAGCTCGTANNNNNNNTGTAGCTCGTATTACATATCGATGGA
AGCTGTCTGTTTTCTGTAGCTCGTA
AGCTGTCTGTTTTCTGTAGCTCG
                          TGTAGCTCGTATTACATATCGATGGA
                            AGCTCGTATTACATATCGATGGA
```

Scaffold underestimation

```
AGCTGTCTGTTTTCTGTTATTACATATCGATGGA
AGCTGTCTGTTTTCTGTAGCTCGTA
AGCTGTCTGTTTTCTGTAGCTCG
          TGTAGCTCGTATTACATATCGATGGA
            AGCTCGTATTACATATCGATGGA
```

Deleted sequences, a nightmare to fix