Genome assembly

Genome sequencing, quality control, and assembly

Overview

- High level view of the concepts underlying the first steps in your work

- Starting point for further investigation (minimum knowledge)

- Intended to be practical
  - Considerations for your own work
  - Sense of best-practices (where relevant)
  - Some suggestions for software

Outline

- Historical context for sequencing and NGS
  - Sanger sequencing
  - Roche’s 454
  - Illumina Sequencing
  - PacBio + Oxford Nanopore
- FASTQ and quality scores
- Quality control
  - FASTQC
- Genome assembly
  - Reference versus *de novo* assembly
  - De novo assembly algorithmic paradigms
    - Overlap layout consensus
    - de Bruijn graphs
- Genome assembly metrics

Slides courtesy of:

ABiL
Applied BioInformatics Laboratory

A unique bioinformatics resource for translation of molecular data into actionable public health intelligence

http://www.abil.ihrc.com/

Outline

- Historical context for sequencing and NGS
  - Sanger sequencing
  - Roche’s 454
  - Illumina Sequencing
  - PacBio + Oxford Nanopore
- FASTQ and quality scores
- Quality control
  - FASTQC
- Genome assembly
  - Reference versus *de novo* assembly
  - De novo assembly algorithmic paradigms
    - Overlap layout consensus
    - de Bruijn graphs
- Genome assembly metrics

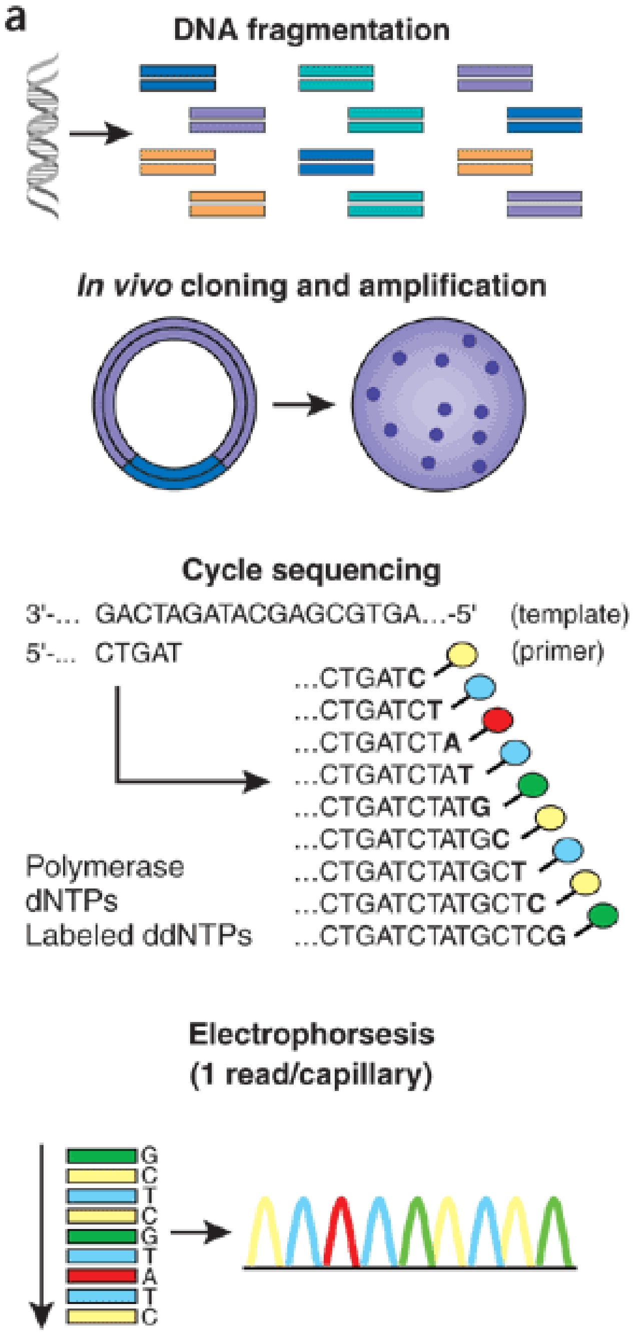
Slides courtesy of:

ABiL

Applied BioInformatics Laboratory

A unique bioinformatics resource for translation of molecular data into actionable public health intelligence

http://www.abil.ihrc.com/



Shendure and Ji. 2008 Nat Biotech. 26:1135–1145

Sanger Sequencing

- Developed in 1977
- Based on chain termination chemistry using fluorescently labeled di-deoxy NTPs (ddNTPs)
- A lot of sequencing happened with Sanger:
  - First gene
  - First virus
  - *H. influenza (1995)* – first free-living organism
  - *S. cerevisiae (1996)* – first eukaryote
  - *E. coli (1997)*
  - Human genome (2000)

Sequencing the human genome in a factory-style setting

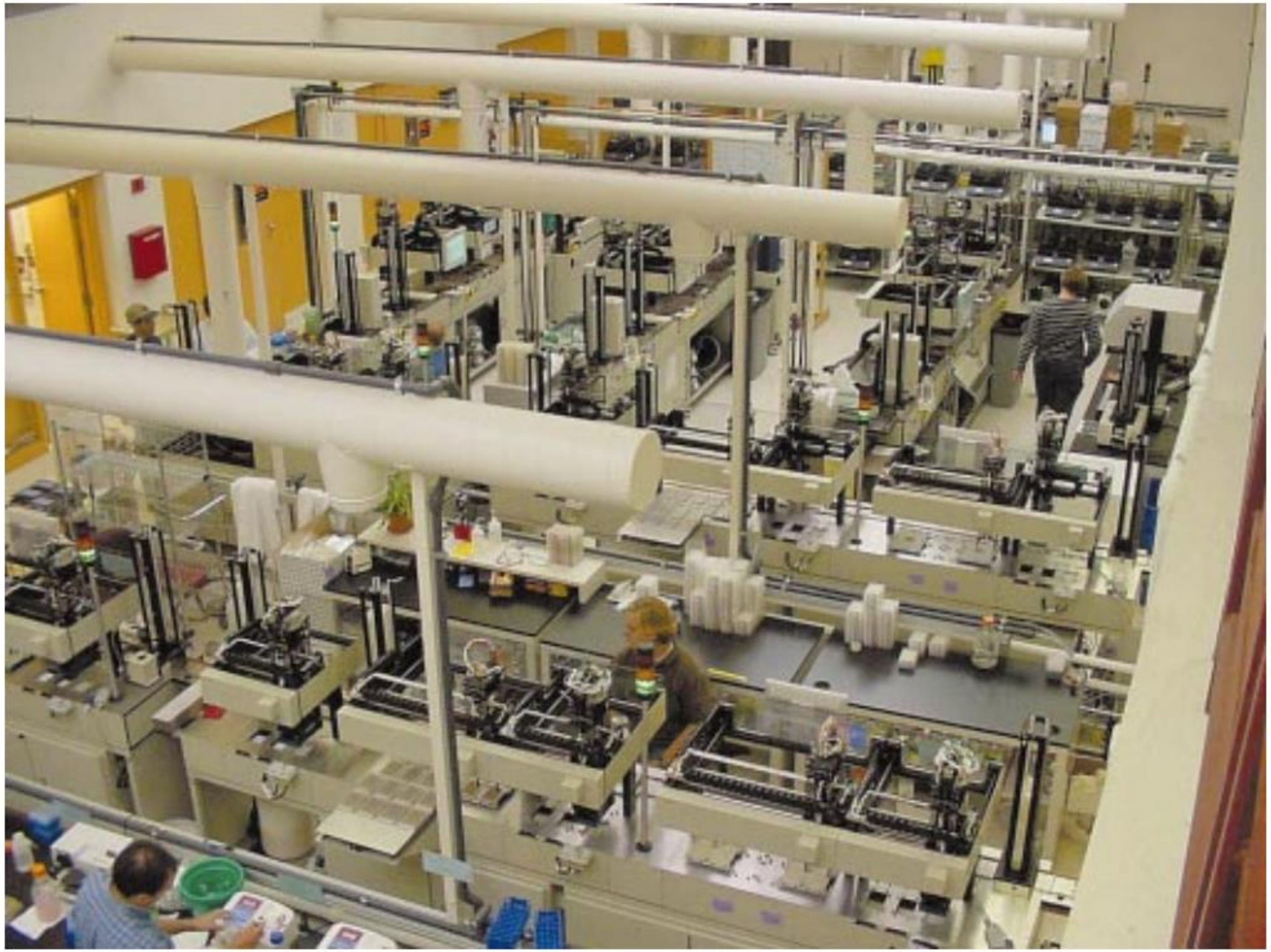
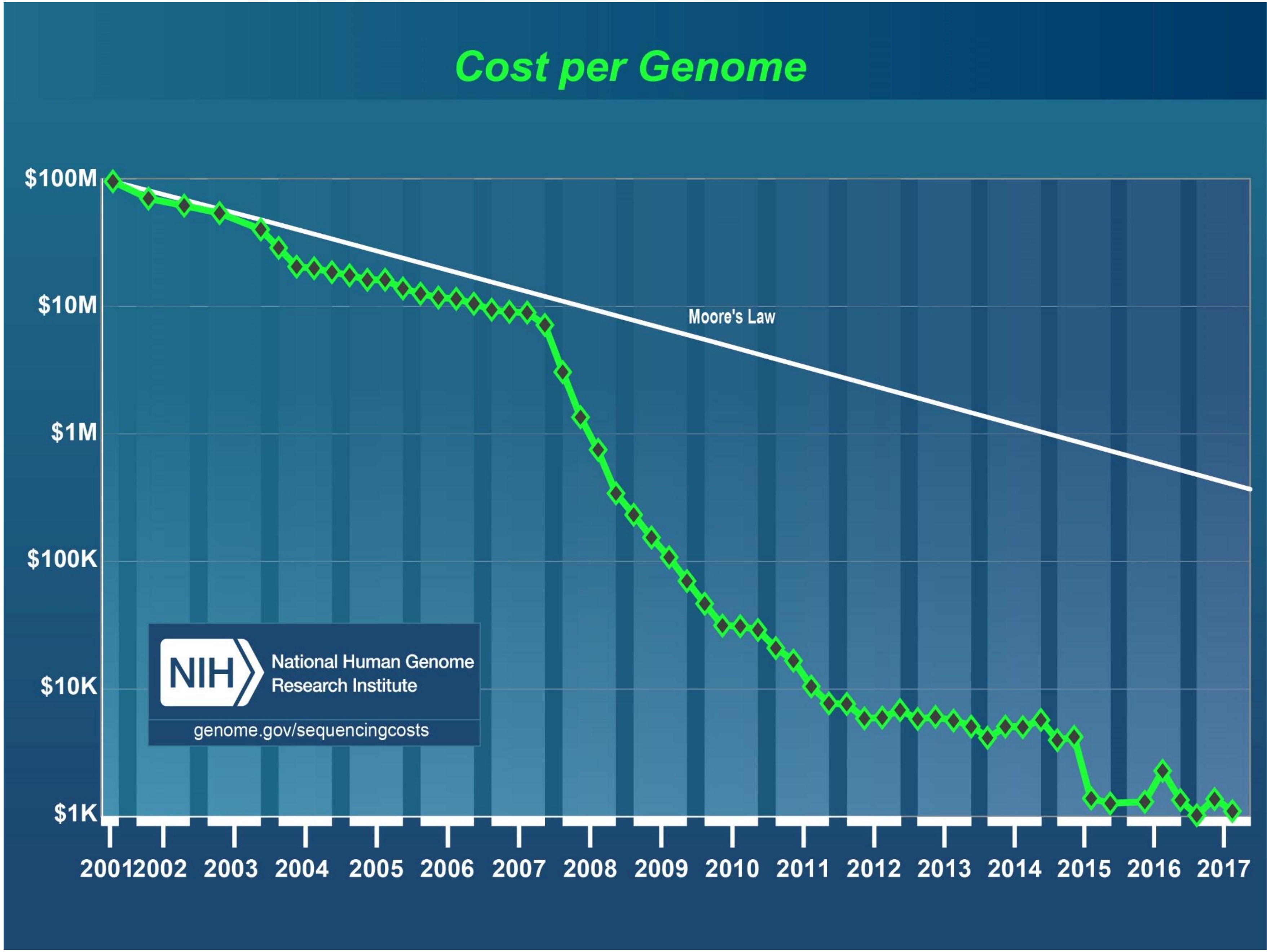


Figure 3 The automated production line for sample preparation at the Whitehead Institute, Center for Genome Research. The system consists of custom-designed factory-style conveyor belt robots that perform all functions from purifying DNA from bacterial cultures through setting up and purifying sequencing reactions.

A big change with (Roche) 454

Advent of next-generation sequencing (NGS)

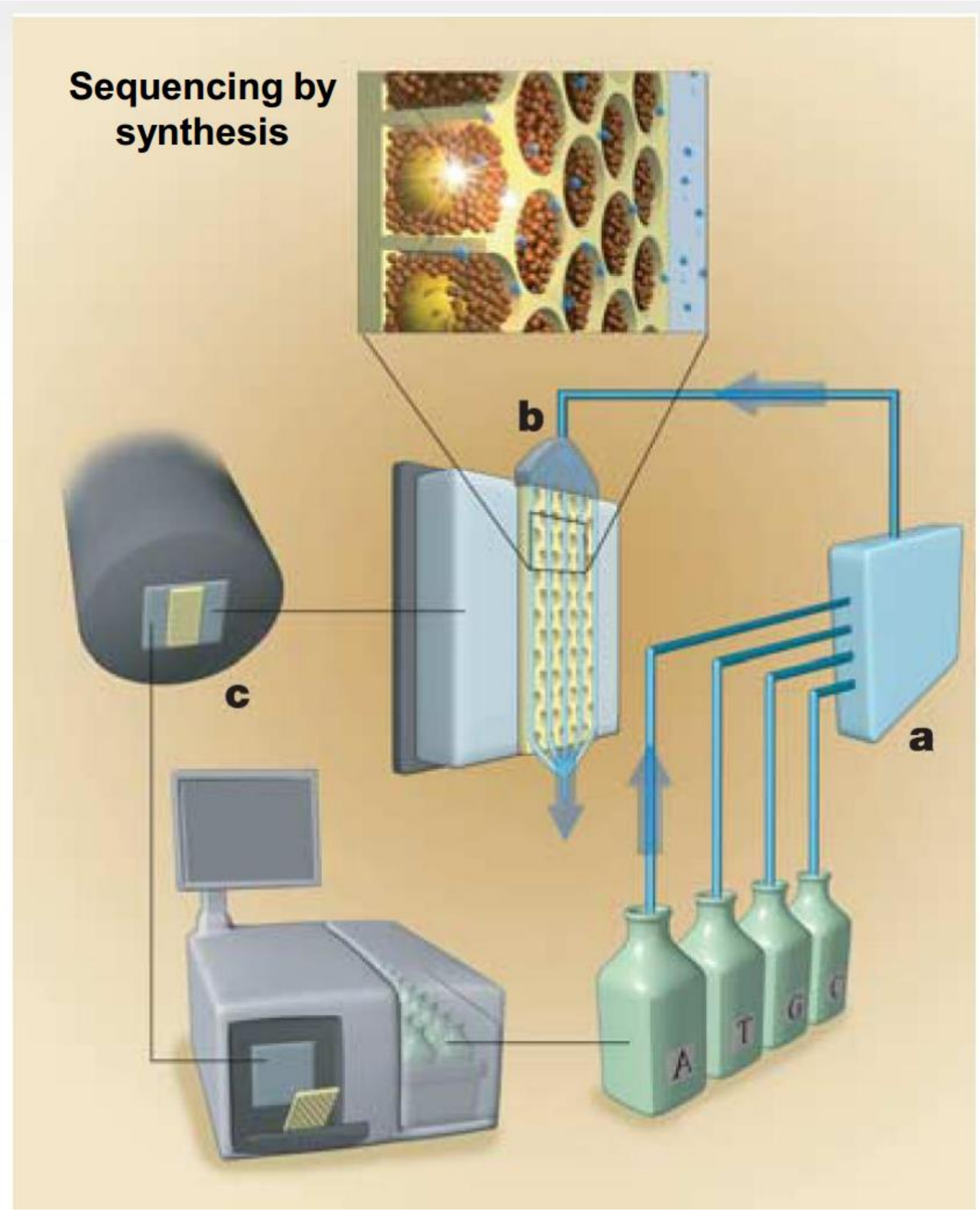


“Sequencing-by-synthesis” paradigm

- 454 was the first SBS (sequencing-by-synthesis) machine to reach the market
  - Bases of a DNA molecule are read as a complimentary molecule is synthesized
  - As opposed to the whole complimentary molecule being synthesized and then read out

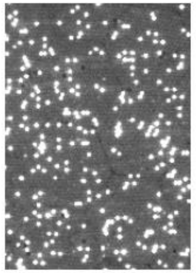
- Much smaller volumes of reagents
  - Many, many reads at the same time
  - 454 was originally a few 100K vs 96 in capillary

- Lost some bases in read length = 700 bp vs 1kb in Sanger



Margulies et al. 2005. Nature 437: 376–380

Photons generated are captured by CCD camera



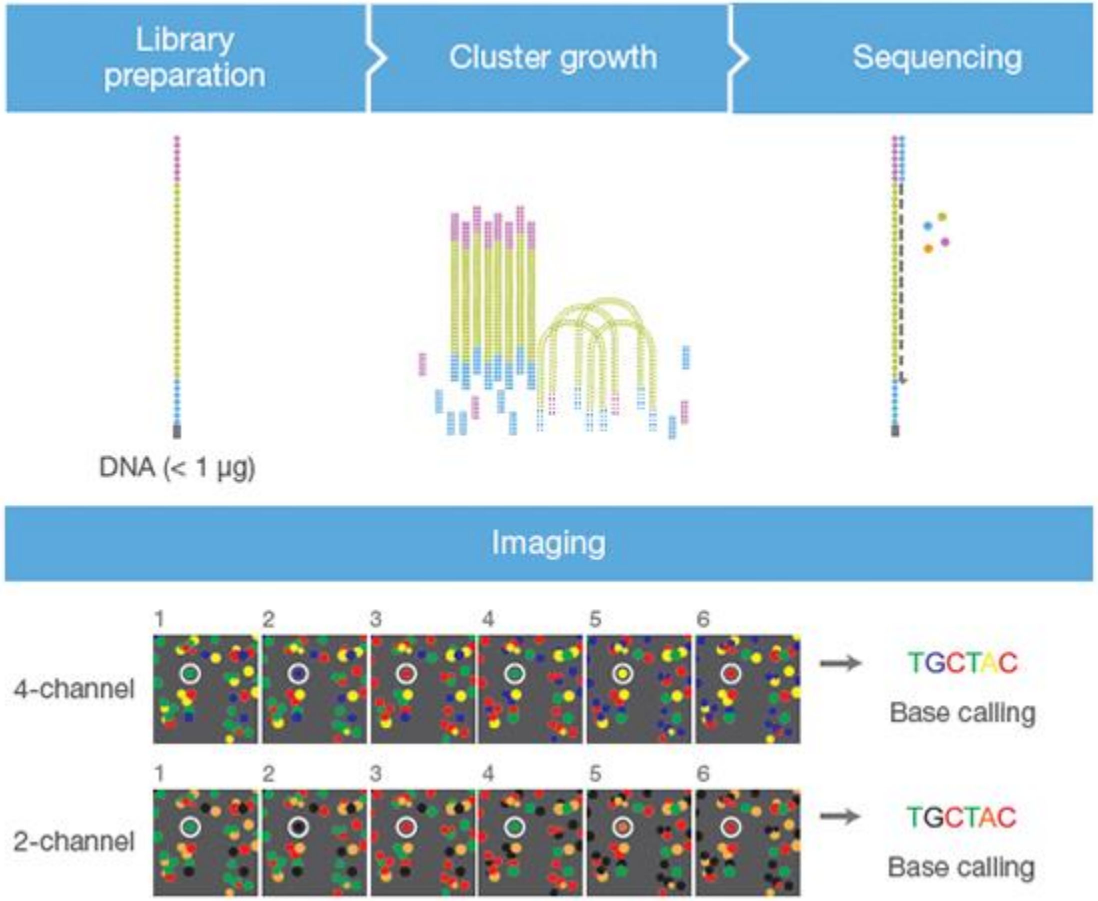
Illumina Sequencing

- Illumina brought three big changes with it:
  1. Small scale, very tiny reactions, even smaller than 454, allowing for much greater density
  2. Reversible terminator chemistry - allows the controlled addition of one base at a time
  3. Optics. The unsung hero of the sequencing revolution. This allows for the insane density and number of reads that you can fit on an Illumina flow cell

- Big issue with Illumina = smaller read length

- Started with 35bp, today we are at 2x300bp

https://www.illumina.com/science/technology/next-generation-sequencing/sequencing-technology/2-channel-sbs.html



Illumina Sequencing

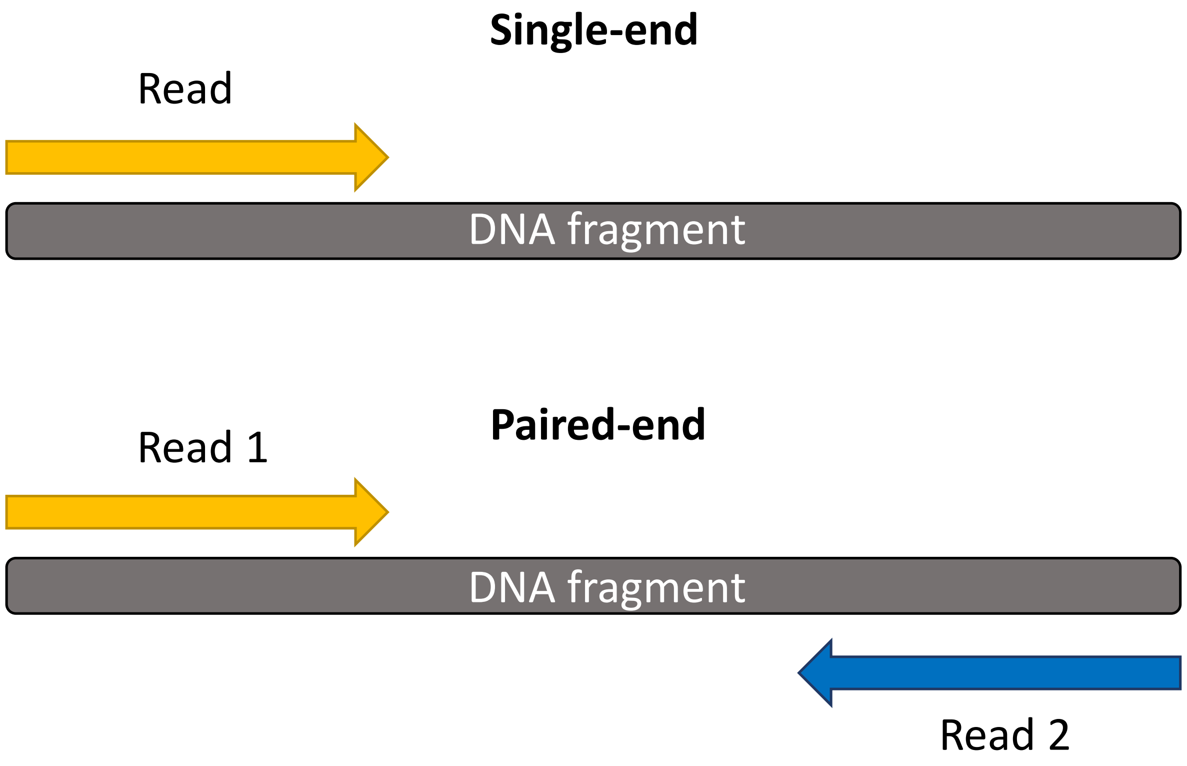
https://www.youtube.com/watch?v=fCd6B5HRaZ8

Paired-end sequencing

- In paired-end sequencing, a DNA fragment is read twice – once from each end (recall the Illumina video!)

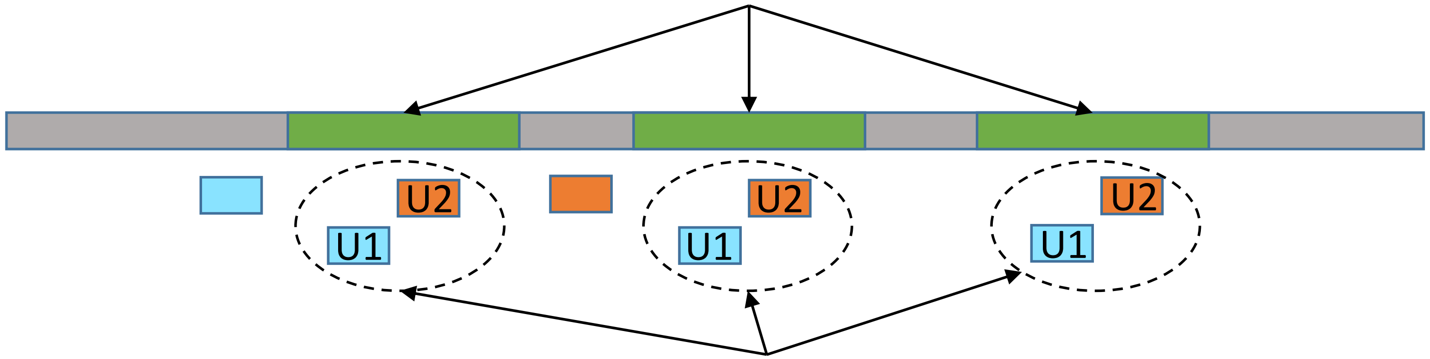
Advantages

- More efficient use of the fragment library
- Improves alignment
- Resolving chromosomal rearrangements like insertions, deletions and inversions
- Scaffolding becomes possible



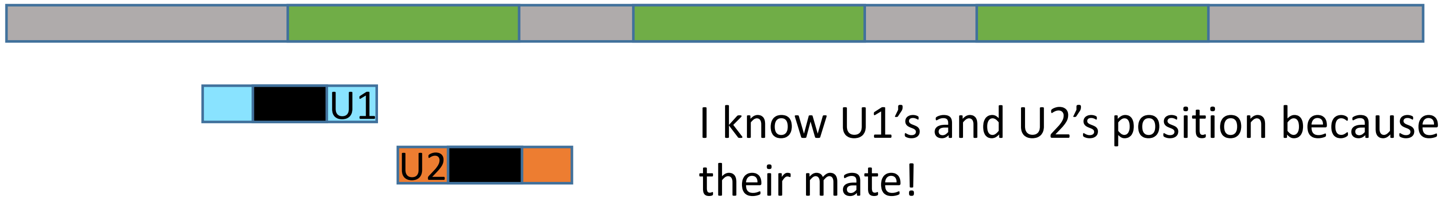
Resolving repeats with paired-end

Repeating region



Case 1: single-end

Where are these reads coming from???



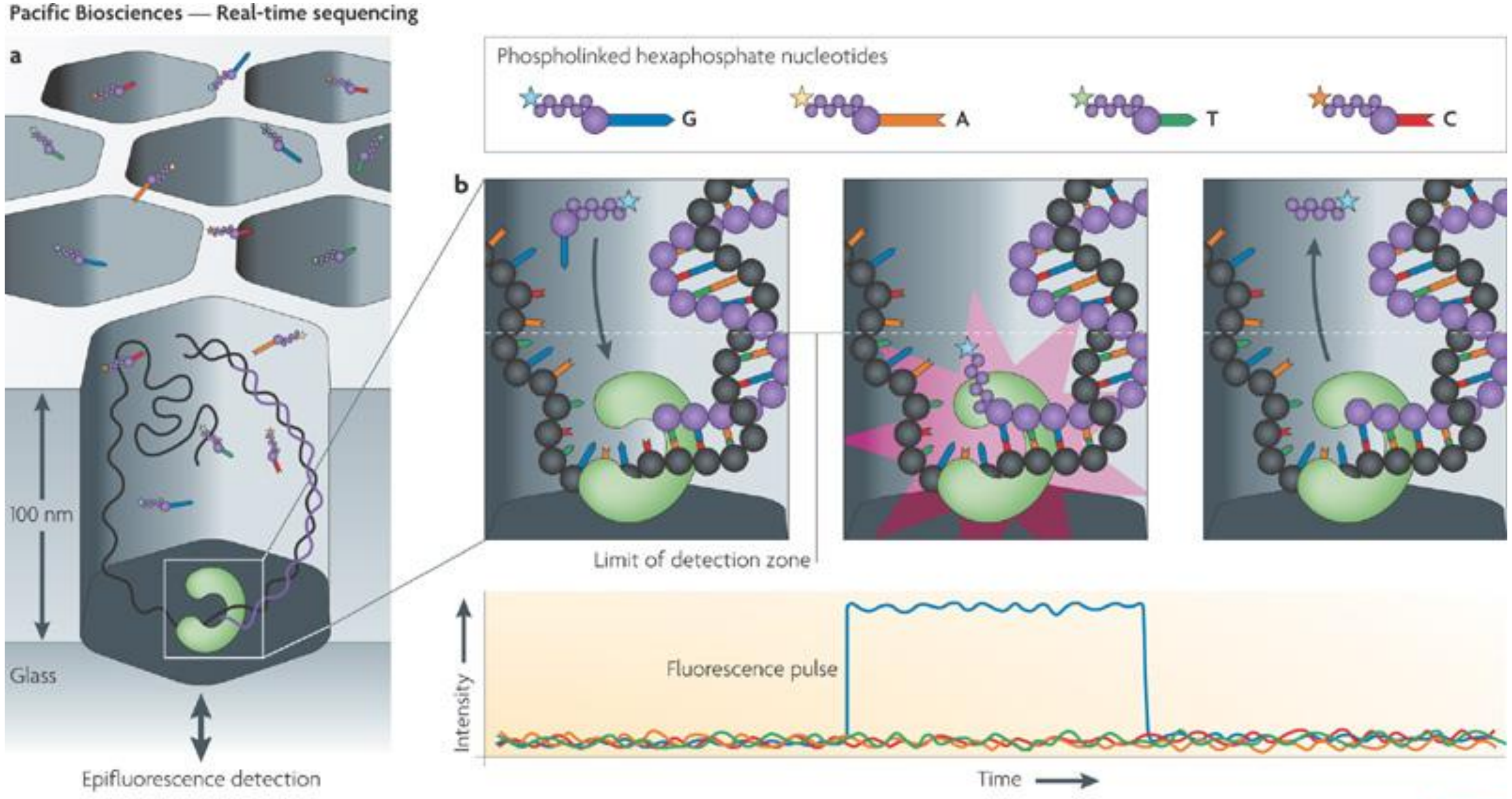
I know U1’s and U2’s position because of their mate!

Case 2: paired-end

Third Generation Sequencing

- Shift towards single molecule, long read sequencing
- Two big names here: Pacific Biosciences (PacBio) and Oxford Nanopore

Pacific Biosciences

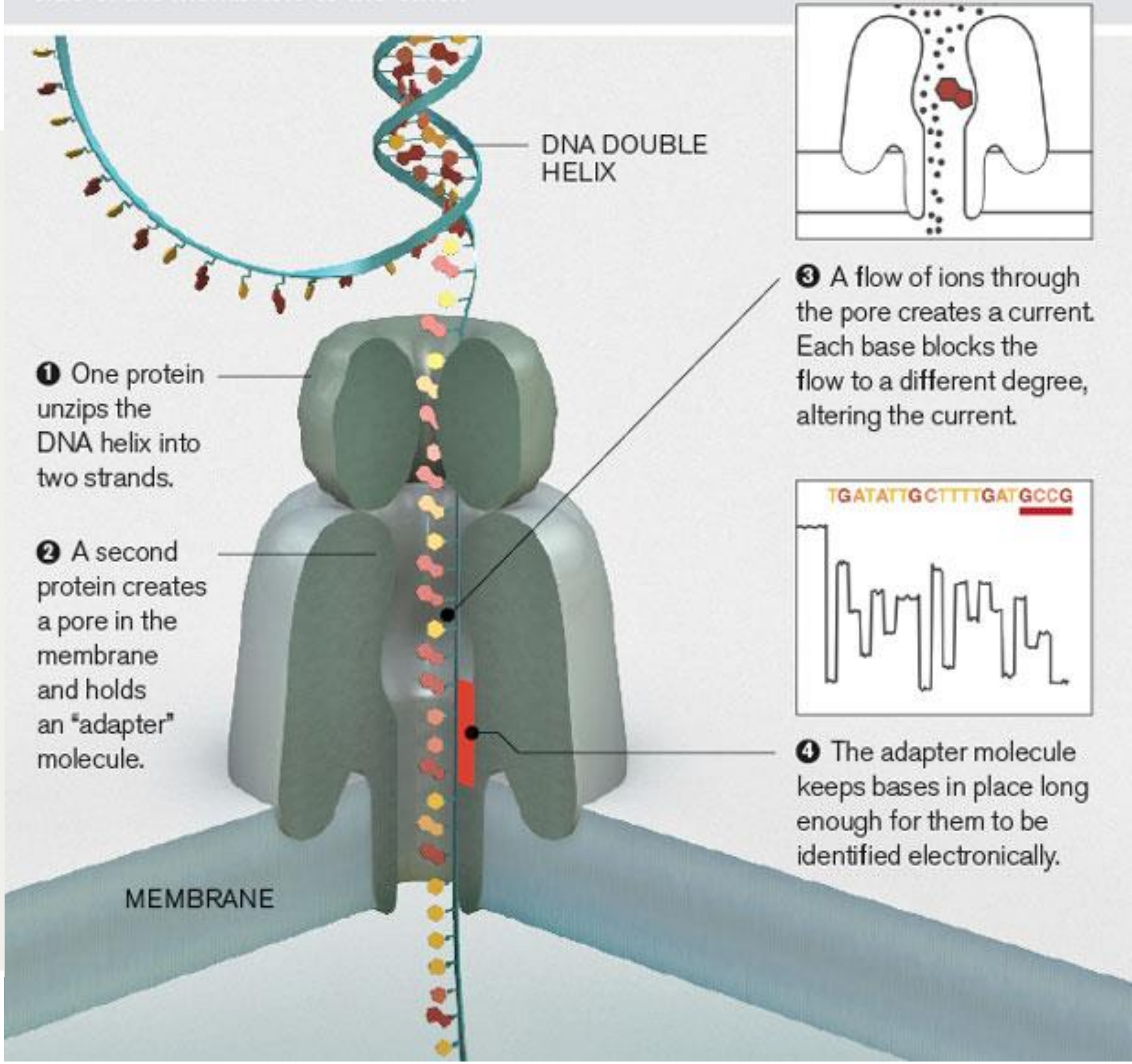

Pacific Biosciences — Real-time sequencing
a
100 nm
Glass
Epifluorescence detection
Phospholinked hexaphosphate nucleotides
G
A
T
C
b
Limit of detection zone
Fluorescence pulse
Intensity
Time

Nature Reviews | Genetics

Oxford Nanopore Technology (ONT)

https://nanoporetech.com/about-us/news/oxford-nanopore-announces-ps100-million-140m-fundraising-global-investors

DNA can be sequenced by threading it through a microscopic pore in a membrane. Bases are identified by the way they affect ions flowing through the pore from one side of the membrane to the other.



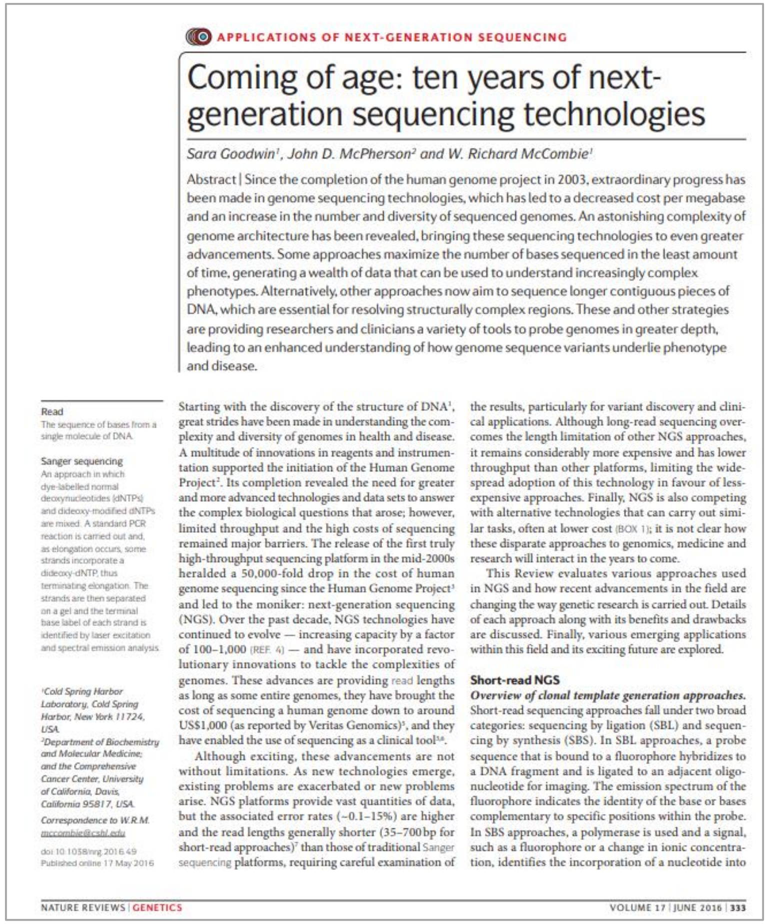
❶ One protein unzips the DNA helix into two strands.

❷ A second protein creates a pore in the membrane and holds an "adapter" molecule.

❸ A flow of ions through the pore creates a current. Each base blocks the flow to a different degree, altering the current.

❹ The adapter molecule keeps bases in place long enough for them to be identified electronically.

References



APPLICATIONS OF NEXT-GENERATION SEQUENCING

Coming of age: ten years of next-generation sequencing technologies

Sara Goodwin[1], John D. McPherson[2] and W. Richard McCombie[1]

Abstract | Since the completion of the human genome project in 2003, extraordinary progress has been made in genome sequencing technologies, which has led to a decreased cost per megabase and an increase in the number and diversity of sequenced genomes. An astonishing complexity of genome architecture has been revealed, bringing these sequencing technologies to even greater advancements. Some approaches maximize the number of bases sequenced in the least amount of time, generating a wealth of data that can be used to understand increasingly complex phenotypes. Alternatively, other approaches now aim to sequence longer contiguous pieces of DNA, which are essential for resolving structurally complex regions. These and other strategies are providing researchers and clinicians a variety of tools to probe genomes in greater depth, leading to an enhanced understanding of how genome sequence variants underlie phenotype and disease.

Starting with the discovery of the structure of DNA[1], great strides have been made in understanding the complexity and diversity of genomes in health and disease. A multitude of innovations in reagents and instrumentation supported the initiation of the Human Genome Project[2]. Its completion revealed the need for greater and more advanced technologies and data sets to answer the complex biological questions that arose; however, limited throughput and the high costs of sequencing remained major barriers. The release of the first truly high-throughput sequencing platform in the mid-2000s heralded a 50,000-fold drop in the cost of human genome sequencing since the Human Genome Project[3] and led to the moniker: next-generation sequencing (NGS). Over the past decade, NGS technologies have continued to evolve — increasing capacity by a factor of 100–1,000 (REF. 4) — and have incorporated revolutionary innovations to tackle the complexities of genomes. These advances are providing read lengths as long as some entire genomes, they have brought the cost of sequencing a human genome down to around US$1,000 (as reported by Veritas Genomics)[5], and they have enabled the use of sequencing as a clinical tool[5,6].

Although exciting, these advancements are not without limitations. As new technologies emerge, existing problems are exacerbated or new problems arise. NGS platforms provide vast quantities of data, but the associated error rates (~0.1–15%) are higher and the read lengths generally shorter (35–700 bp for short-read approaches)[7] than those of traditional Sanger sequencing platforms, requiring careful examination of the results, particularly for variant discovery and clinical applications. Although long-read sequencing overcomes the length limitation of other NGS approaches, it remains considerably more expensive and has lower throughput than other platforms, limiting the widespread adoption of this technology in favour of less-expensive approaches. Finally, NGS is also competing with alternative technologies that can carry out similar tasks, often at lower cost (BOX 1); it is not clear how these disparate approaches to genomics, medicine and research will interact in the years to come.

This Review evaluates various approaches used in NGS and how recent advancements in the field are changing the way genetic research is carried out. Details of each approach along with its benefits and drawbacks are discussed. Finally, various emerging applications within this field and its exciting future are explored.

Short-read NGS

Overview of clonal template generation approaches. Short-read sequencing approaches fall under two broad categories: sequencing by ligation (SBL) and sequencing by synthesis (SBS). In SBL approaches, a probe sequence that is bound to a fluorophore hybridizes to a DNA fragment and is ligated to an adjacent oligonucleotide for imaging. The emission spectrum of the fluorophore indicates the identity of the base or bases complementary to specific positions within the probe. In SBS approaches, a polymerase is used and a signal, such as a fluorophore or a change in ionic concentration, identifies the incorporation of a nucleotide into

Read
The sequence of bases from a single molecule of DNA.

Sanger sequencing
An approach in which dye-labelled normal deoxynucleotides (dNTPs) and dideoxy-modified dNTPs are mixed. A standard PCR reaction is carried out and, as elongation occurs, some strands incorporate a dideoxy-dNTP, thus terminating elongation. The strands are then separated on a gel and the terminal base label of each strand is identified by laser excitation and spectral emission analysis.

[1]*Cold Spring Harbor Laboratory, Cold Spring Harbor, New York 11724, USA.*
[2]*Department of Biochemistry and Molecular Medicine; and the Comprehensive Cancer Center, University of California, Davis, California 95817, USA.*
Correspondence to W.R.M.
mccombie@cshl.edu

doi:10.1038/nrg.2016.49
Published online 17 May 2016

Outline

- Historical context for sequencing and NGS
  - Sanger sequencing
  - Roche’s 454
  - Illumina Sequencing
  - PacBio + Oxford Nanopore
- **FASTQ and quality scores**
- Quality control
  - FASTQC
- Genome assembly
  - Reference versus *de novo* assembly
  - De novo assembly algorithmic paradigms
    - Overlap layout consensus
    - de Bruijn graphs
- Genome assembly metrics

Slides courtesy of:

ABiL

Applied BioInformatics Laboratory

A unique bioinformatics resource for translation of molecular data into actionable public health intelligence

http://www.abil.ihrc.com/

FASTQ format

1. The base calls for the sequence read (i.e. the sequence)
   - What wavelength did the machine pick up when the spots were high with a light source?

2. A set of quality scores; one per base call generated
   - Always have the exact same number of quality scores as you have base calls

```
@HWE1FGTJ-GH13-454470/1
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAA
+
!''*((((***+))%%%++)(%%%%).1***-+*'')
```

Read Identifier Line

Sequence

Description Line

Encoded Quality Scores

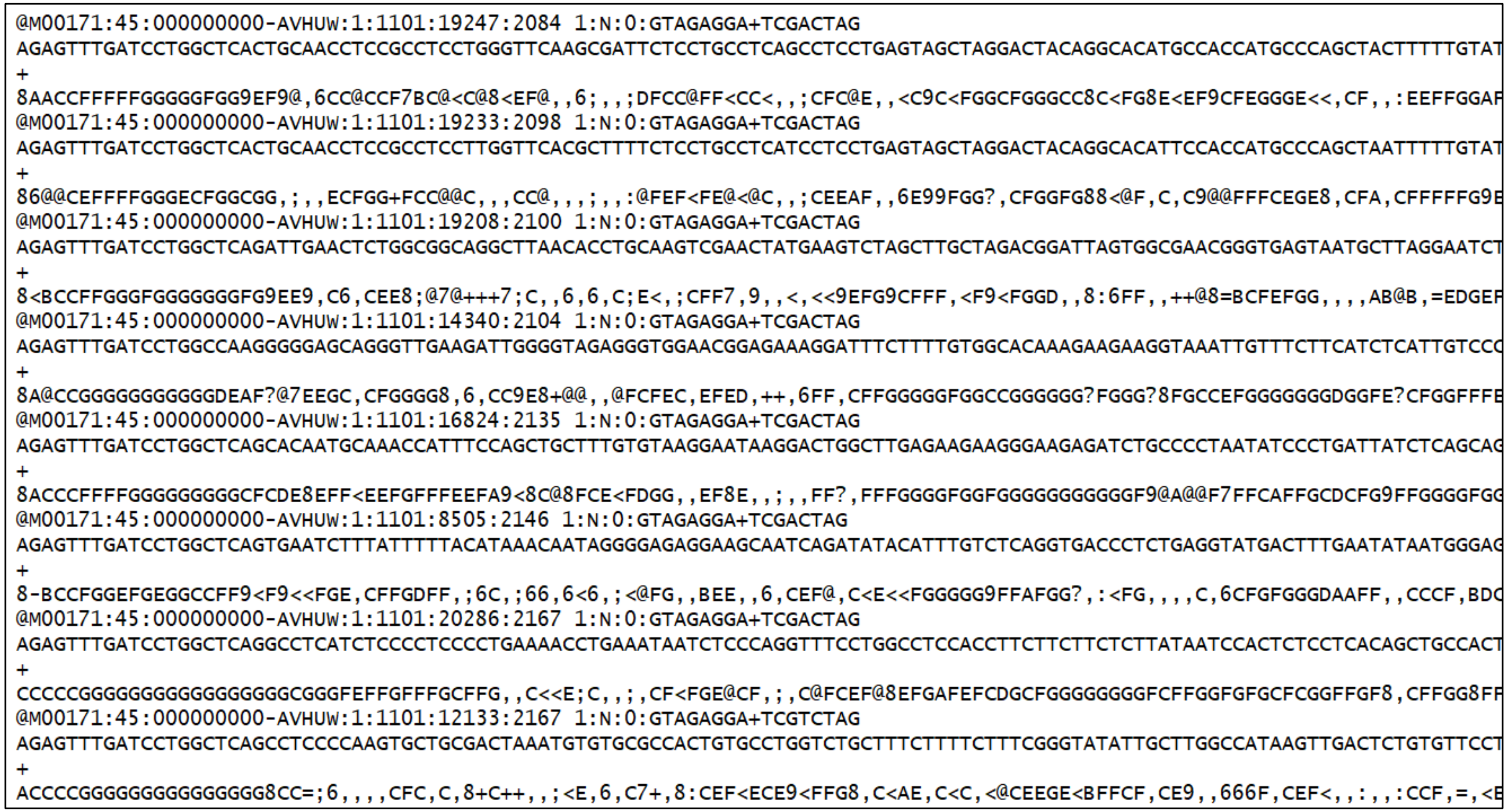
What does a quality score mean?

- A probabilistic measure of how confident the machine is that it called the correct base for that cycle for that spot
  - Illumina has a series of internal metrics for calculating quality: intensity of the called base, proximity to nearby spots, off-color intensity

- High quality is good, low quality is bad
  - **High quality** → **High probability of a correct base call** for that spot and cycle
  - Low quality → Low probability of a correct base call for that spot and cycle

- Illumina has gotten really good at producing good quality read files but every now and then you produce bad data
  - Garbage in, garbage out

An example FASTQ file



```
@M00171:45:000000000-AVHUW:1:1101:19247:2084 1:N:0:GTAGAGGA+TCGACTAG
AGAGTTTGATCCTGGCTCACTGCAACCTCCGCCTCCTGGGTTCAAGCGATTCTCCTGCCTCAGCCTCCTGAGTAGCTAGGACTACAGGCACATGCCACCATGCCCAGCTACTTTTTGTAT
+
8AACCFFFFFGGGGGFGG9EF9@,6CC@CCF7BC@<C@8<EF@,,6;,,;DFCC@FF<CC<,,;CFC@E,,<C9C<FGGCFGGGCC8C<FG8E<EF9CFEGGGE<<,CF,,:EEFFGGAF
@M00171:45:000000000-AVHUW:1:1101:19233:2098 1:N:0:GTAGAGGA+TCGACTAG
AGAGTTTGATCCTGGCTCACTGCAACCTCCGCCTCCTTGGTTCACGCTTTTCTCCTGCCTCATCCTCCTGAGTAGCTAGGACTACAGGCACATTCCACCATGCCCAGCTAATTTTTGTAT
+
86@@CEFFFFGGGECFGGCGG,;,,ECFGG+FCC@@C,,,CC@,,,;,,:@FEF<FE@<@C,,;CEEAF,,6E99FGG?,CFGGFG88<@F,C,C9@@FFFCEGE8,CFA,CFFFFFG9E
@M00171:45:000000000-AVHUW:1:1101:19208:2100 1:N:0:GTAGAGGA+TCGACTAG
AGAGTTTGATCCTGGCTCAGATTGAACTCTGGCGGCAGGCTTAACACCTGCAAGTCGAACTATGAAGTCTAGCTTGCTAGACGGATTAGTGGCGAACGGGTGAGTAATGCTTAGGAATCT
+
8<BCCFFGGGFGGGGGGGFG9EE9,C6,CEE8;@7@+++7;C,,6,6,C;E<,;CFF7,9,,<,<<9EFG9CFFF,<F9<FGGD,,8:6FF,,++@8=BCFEFGG,,,,AB@B,=EDGEF
@M00171:45:000000000-AVHUW:1:1101:14340:2104 1:N:0:GTAGAGGA+TCGACTAG
AGAGTTTGATCCTGGCCAAGGGGGAGCAGGGTTGAAGATTGGGGTAGAGGGTGGAACGGAGAAAGGATTTCTTTTGTGGCACAAAGAAGAAGGTAAATTGTTTCTTCATCTCATTGTCCC
+
8A@CCGGGGGGGGGGGGDEAF?@7EEGC,CFGGGG8,6,CC9E8+@@,,@FCFEC,EFED,++,6FF,CFFGGGGGFGGCCGGGGGG?FGGG?8FGCCEFGGGGGGGDGGFE?CFGGFFFE
@M00171:45:000000000-AVHUW:1:1101:16824:2135 1:N:0:GTAGAGGA+TCGACTAG
AGAGTTTGATCCTGGCTCAGCACAATGCAAACCATTTCCAGCTGCTTTGTGTAAGGAATAAGGACTGGCTTGAGAAGAAGGGAAGAGATCTGCCCCTAATATCCCTGATTATCTCAGCAG
+
8ACCCFFFFGGGGGGGGGCFCDE8EFF<EEFGFFFEEFA9<8C@8FCE<FDGG,,EF8E,,;,,FF?,FFFGGGGFGGFGGGGGGGGGGGF9@A@@F7FFCAFFGCDCFG9FFGGGGFGG
@M00171:45:000000000-AVHUW:1:1101:8505:2146 1:N:0:GTAGAGGA+TCGACTAG
AGAGTTTGATCCTGGCTCAGTGAATCTTTATTTTTACATAAACAATAGGGGAGAGGAAGCAATCAGATATACATTTGTCTCAGGTGACCCTCTGAGGTATGACTTTGAATATAATGGGAG
+
8-BCCFGGEFGEGGCCFF9<F9<<FGE,CFFGDFF,;6C,;66,6<6,;<@FG,,BEE,,6,CEF@,C<E<<FGGGGG9FFAFGG?,:<FG,,,,C,6CFGFGGGDAAFF,,CCCF,BDC
@M00171:45:000000000-AVHUW:1:1101:20286:2167 1:N:0:GTAGAGGA+TCGACTAG
AGAGTTTGATCCTGGCTCAGGCCTCATCTCCCCTCCCCTGAAAACCTGAAATAATCTCCCAGGTTTCCTGGCCTCCACCTTCTTCTTCTCTTATAATCCACTCTCCTCACAGCTGCCACT
+
CCCCCGGGGGGGGGGGGGGGGCGGGFEFFGFFFGCFFG,,C<<E;C,,;,CF<FGE@CF,;,C@FCEF@8EFGAFEFCDGCFGGGGGGGGFCFFGGFGFGCFCGGFFGF8,CFFGG8FF
@M00171:45:000000000-AVHUW:1:1101:12133:2167 1:N:0:GTAGAGGA+TCGTCTAG
AGAGTTTGATCCTGGCTCAGCCTCCCCAAGTGCTGCGACTAAATGTGTGCGCCACTGTGCCTGGTCTGCTTTCTTTTCTTTCGGGTATATTGCTTGGCCATAAGTTGACTCTGTGTTCCT
+
ACCCCGGGGGGGGGGGGGGG8CC=;6,,,,CFC,C,8+C++,,;<E,6,C7+,8:CEF<ECE9<FFG8,C<AE,C<C,<@CEEGE<BFFCF,CE9,,666F,CEF<,,:,,:CCF,=,<E
```

Phred quality score, Q

- The quality is related to the probability (p) that the base called is incorrect – basecalling error

$$Q = -10log_{10}p$$

- Quality scores are encoded as their representative ASCII (American Standard Code for Information Interchange) values with some offset

$$ASCII(quality\ +\ offset\ ) =\ FASTQ\ encoded\ quality$$

- If you have a quality of 30, the quality with offset will be 30 + 33 = 63, corresponding to “?”

Phred quality score (Q)	Probability of incorrect base call	Base call accuracy
10	1 in 10	90%
20	1 in 100	99%
30	1 in 1,000	99.9%
40	1 in 10,000	99.99%
50	1 in 100,000	99.999%
60	1 in 1,000,000	99.9999%

ASCII Table

Decimal	Character	Decimal	Character	Decimal	Character	Decimal	Character
0	[NULL]	32	[SPACE]	64	@	96	`
1	[START OF HEADING]	33	!	65	A	97	a
2	[START OF TEXT]	34	"	66	B	98	b
3	[END OF TEXT]	35	#	67	C	99	c
4	[END OF TRANSMISSION]	36	$	68	D	100	d
5	[ENQUIRY]	37	%	69	E	101	e
6	[ACKNOWLEDGE]	38	&	70	F	102	f
7	[BELL]	39	‘	71	G	103	g
8	[BACKSPACE]	40	(	72	H	104	h
9	[HORIZONTAL TAB]	41	)	73	I	105	i
10	[LINE FEED]	42	*	74	J	106	j
11	[VERTICAL TABL]	43	+	75	K	107	k
12	[FORM FEED]	44	,	76	L	108	l
13	[CARRIAGE RETURN]	45	-	77	M	109	m
14	[SHIFT OUT]	46	.	78	N	110	n
15	[SHIFT IN]	47	/	79	O	111	o
16	[DATA LINK ESCAPE]	48	0	80	P	112	p
17	[DEVICE CONTROL 1]	49	1	81	Q	113	q
18	[DEVICE CONTROL 2]	50	2	82	R	114	r
19	[DEVICE CONTROL 3]	51	3	83	S	115	s
20	[DEVICE CONTROL 4]	52	4	84	T	116	t
21	[NEGATIVE ACKNOWLEDGE]	53	5	85	U	117	u
22	[SYNCRHONOUS IDLE]	54	6	86	V	118	v
23	[END OF TRANS. BLOCK]	55	7	87	W	119	w
24	[CANCEL]	56	8	88	X	120	x
25	[END OF MEDIUM]	57	9	89	Y	121	y
26	[SUBSTITUTE]	58	:	90	Z	122	z
27	[ESCAPE]	59	;	91	[	123	{
28	[FILE SEPARATOR]	60	<	92	\	124	|
29	[GROUP SEPARATOR]	61	=	93	]	125	}
30	[RECORD SEPARATOR]	62	>	94	^	126	~
31	[UNIT SEPARATOR]	63	?	95	_	127	[DEL]

Outline

- Historical context for sequencing and NGS
  - Sanger sequencing
  - Roche’s 454
  - Illumina Sequencing
  - PacBio + Oxford Nanopore
- FASTQ and quality scores
- Quality control
  - FASTQC
- Genome assembly
  - Reference versus *de novo* assembly
  - De novo assembly algorithmic paradigms
    - Overlap layout consensus
    - de Bruijn graphs
- Genome assembly metrics

Slides courtesy of:

ABiL

Applied BioInformatics Laboratory

A unique bioinformatics resource for translation of molecular data into actionable public health intelligence

http://www.abil.ihrc.com/

Reads quality assessment

- Quality assessment is an absolute necessity when analyzing any dataset
- Many problems can be stopped at this point before they start – you won’t waste time analyzing bad data, for example
- What to look for when performing quality assessment:
  - Low read depth → bad for any analysis
  - Lingering adapters/primers → poor genome assembly and mapping
  - Low quality bases or PCR duplicates → poor variant calling
  - Small length reads → adds additional time but no value
- Simply, you have to make sure that the experiment worked

FastQC

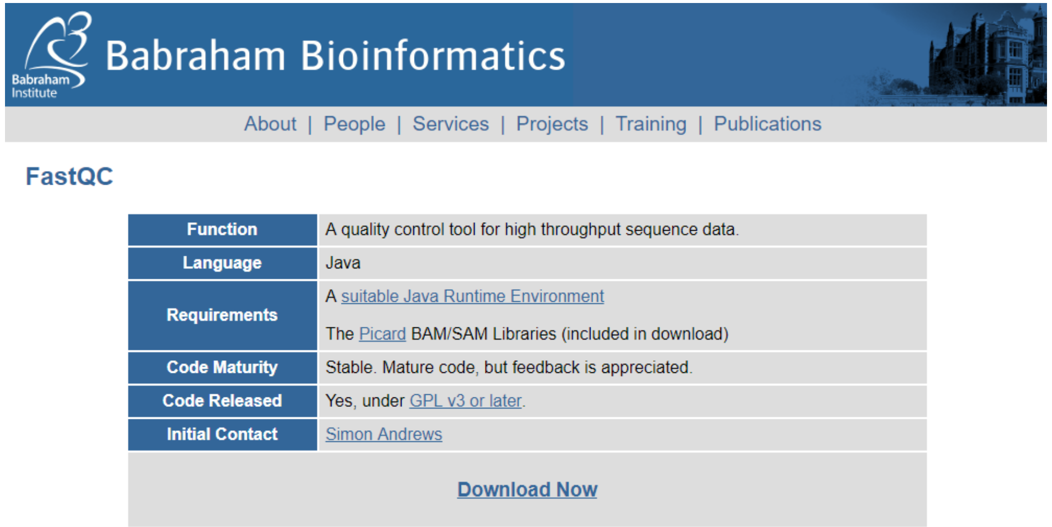
- Great tool for assessing the quality of FASTQ files

- Comes as both a graphical user interface (GUI) and command line interface (CLI)

- The GUI is great if you have one or two sets of FASTQ reads you want to quality assess

- The CLI is better if you have lots of FASTQ reads you want to quality assess

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



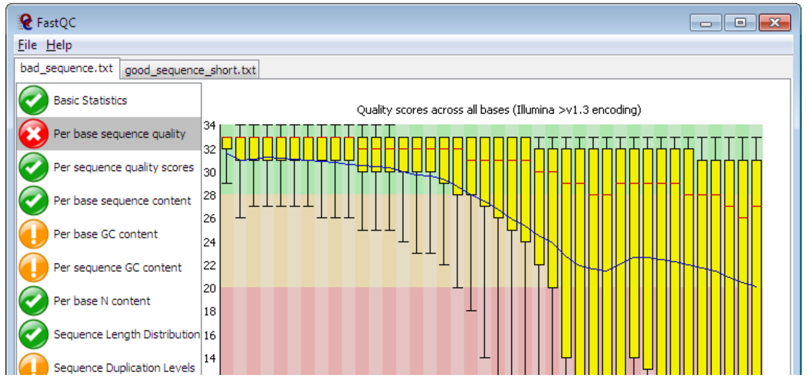
Babraham Bioinformatics

About | People | Services | Projects | Training | Publications

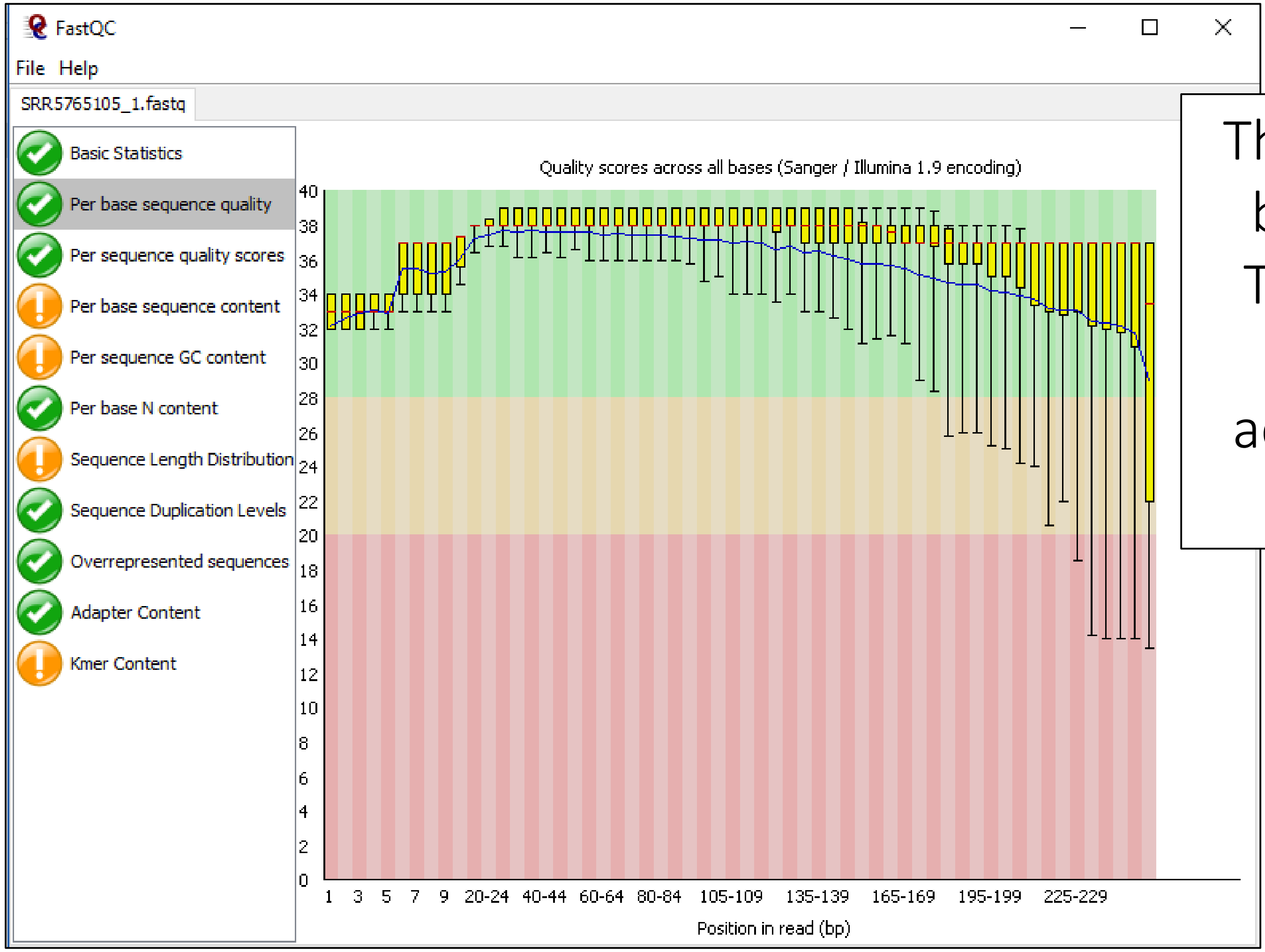
FastQC

Function	A quality control tool for high throughput sequence data.
Language	Java
Requirements	A suitable Java Runtime Environment The Picard BAM/SAM Libraries (included in download)
Code Maturity	Stable. Mature code, but feedback is appreciated.
Code Released	Yes, under GPL v3 or later.
Initial Contact	Simon Andrews

Download Now


FastQC
File Help
bad_sequence.txt
good_sequence_short.txt
Basic Statistics
Per base sequence quality
Per sequence quality scores
Per base sequence content
Per base GC content
Per sequence GC content
Per base N content
Sequence Length Distribution
Sequence Duplication Levels
Quality scores across all bases (Illumina >v1.3 encoding)
34
32
30
28
26
24
22
20
18
16
14

FastQC: Per base sequence quality



FastQC

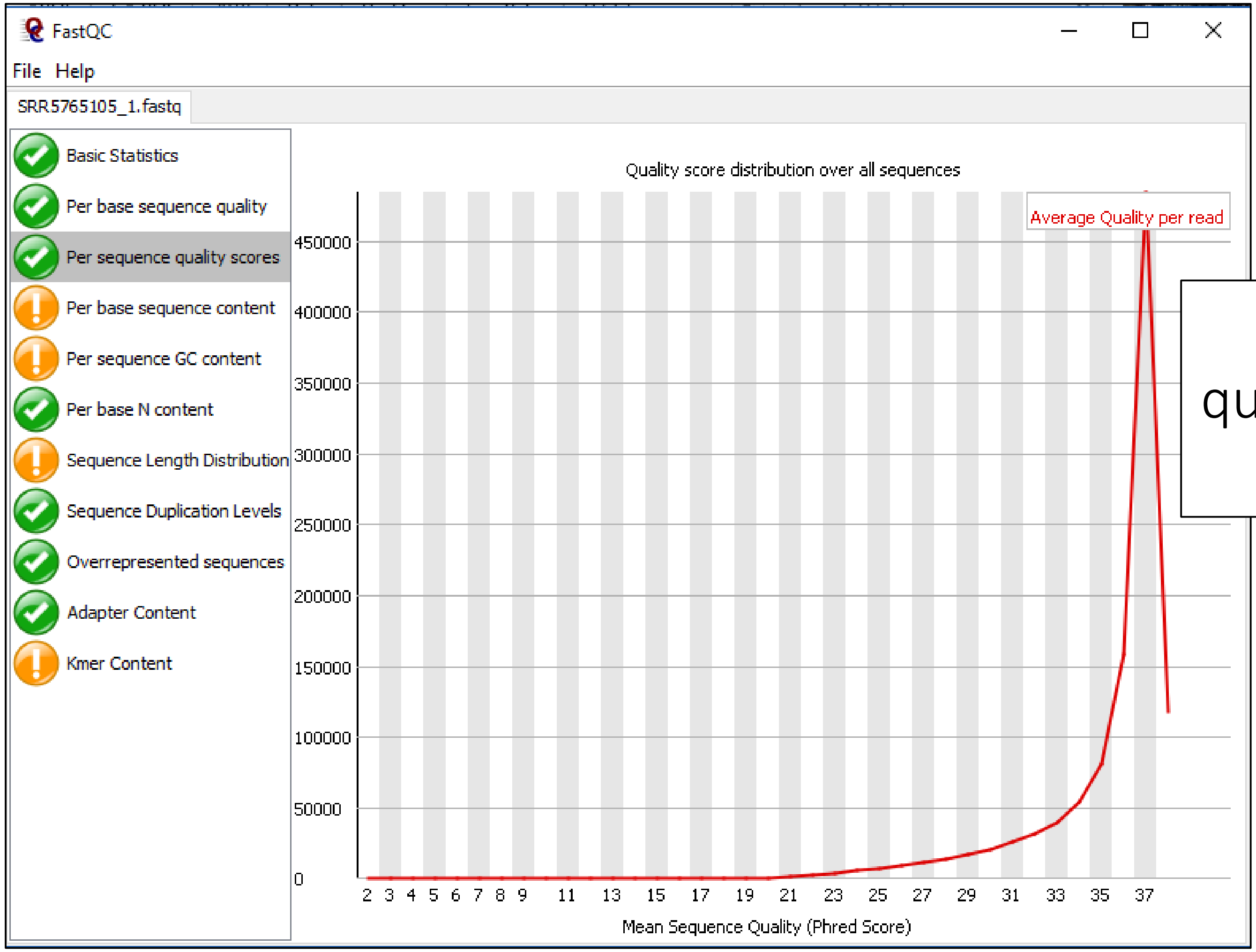
File Help

SRR5765105_1.fastq

- ✓ Basic Statistics
- ✓ Per base sequence quality
- ✓ Per sequence quality scores
- ! Per base sequence content
- ! Per sequence GC content
- ✓ Per base N content
- ! Sequence Length Distribution
- ✓ Sequence Duplication Levels
- ✓ Overrepresented sequences
- ✓ Adapter Content
- ! Kmer Content

This graph shows the per base sequence quality. The red region denotes bad quality, orange is acceptable, and green is good quality

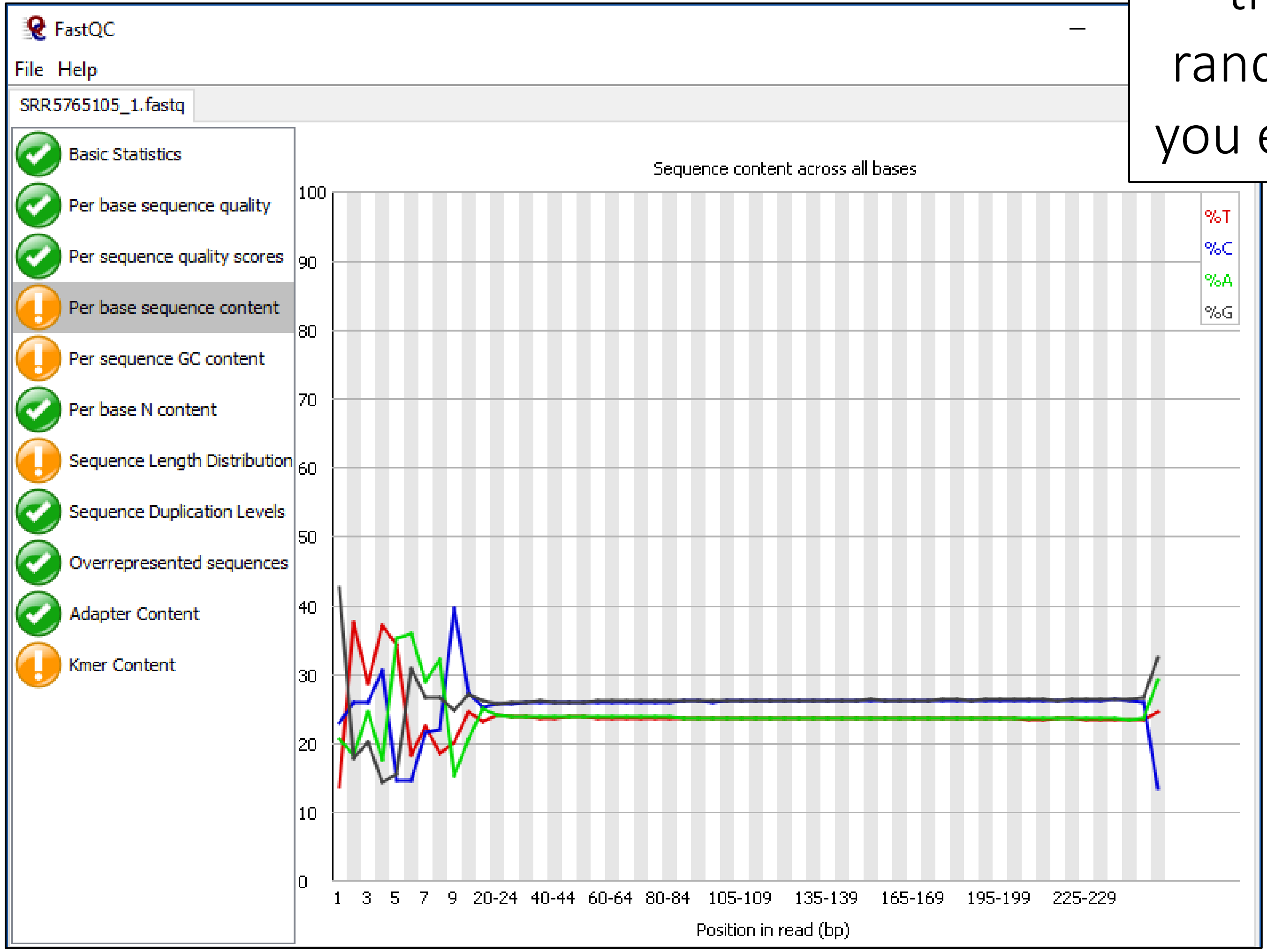
FastQC: Per sequence quality scores


FastQC
File
Help
SRR5765105_1.fastq
✓ Basic Statistics
✓ Per base sequence quality
✓ Per sequence quality scores
! Per base sequence content
! Per sequence GC content
✓ Per base N content
! Sequence Length Distribution
✓ Sequence Duplication Levels
✓ Overrepresented sequences
✓ Adapter Content
! Kmer Content
Quality score distribution over all sequences
Average Quality per read
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
2 3 4 5 6 7 8 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37
Mean Sequence Quality (Phred Score)

This graph shows the quality distribution of each sequence

FastQC: Per base sequence content

This graph shows the per base composition across the read length. For random library selection, you expect 4 parallel lines.



Sequence quality control (trimming)

- Given the initial sequence quality, you will want to perform trimming to ensure maximum quality
  - Primer/adapter removal
  - Read quality trimming
  - Read quality filtering
  - Read length filtering

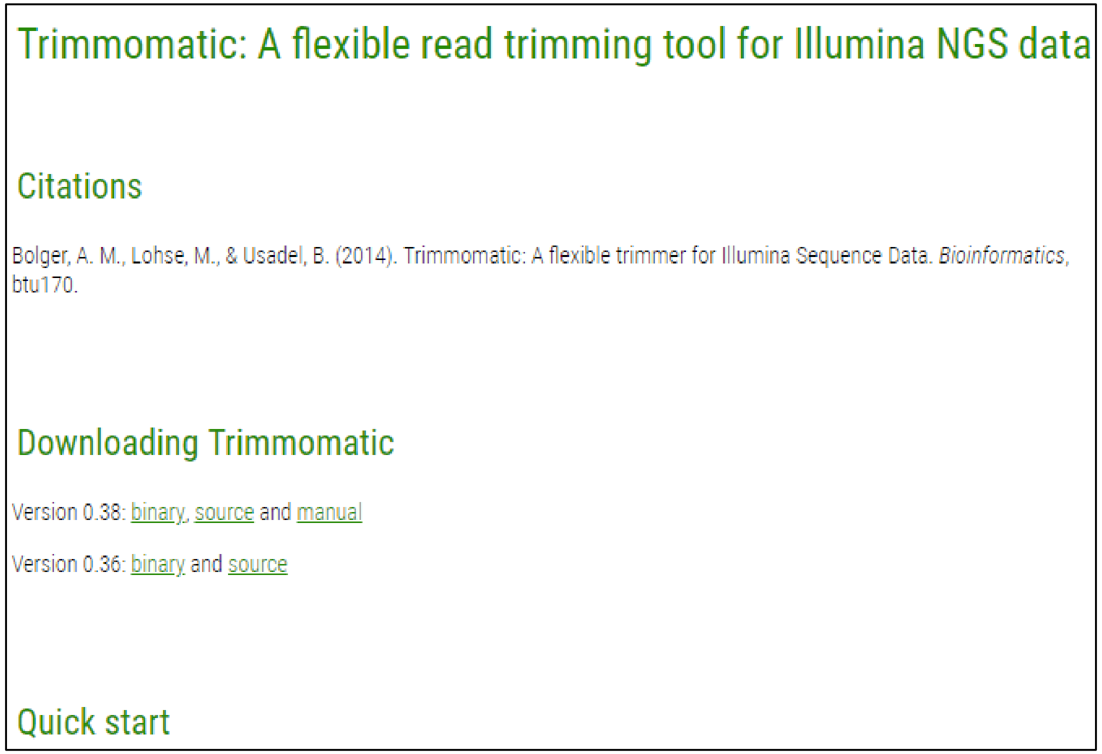
Quality control with Trimmomatic

- Trimmomatic is a command line tool used for performing quality control on reads

- Performs a variety of filtering operations for Illumina paired-end and single end data

- These options typically require users to provide some quality or quantity threshold

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



Trimmomatic: A flexible read trimming tool for Illumina NGS data

Citations

Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: A flexible trimmer for Illumina Sequence Data. *Bioinformatics*, btu170.

Downloading Trimmomatic

Version 0.38: binary, source and manual

Version 0.36: binary and source

Quick start

Outline

- Historical context for sequencing and NGS
  - Sanger sequencing
  - Roche’s 454
  - Illumina Sequencing
  - PacBio + Oxford Nanopore
- FASTQ and quality scores
- Quality control
  - FASTQC
- Genome assembly
  - Reference versus *de novo* assembly
  - De novo assembly algorithmic paradigms
    - Overlap layout consensus
    - de Bruijn graphs
- Genome assembly metrics

Slides courtesy of:

David Gifford MIT

Foundations of Computational and Systems Biology

https://www.youtube.com/watch?v=ZYW2AeDE6wU

Ben Langmead JHU

Algorithms for DNA sequencing

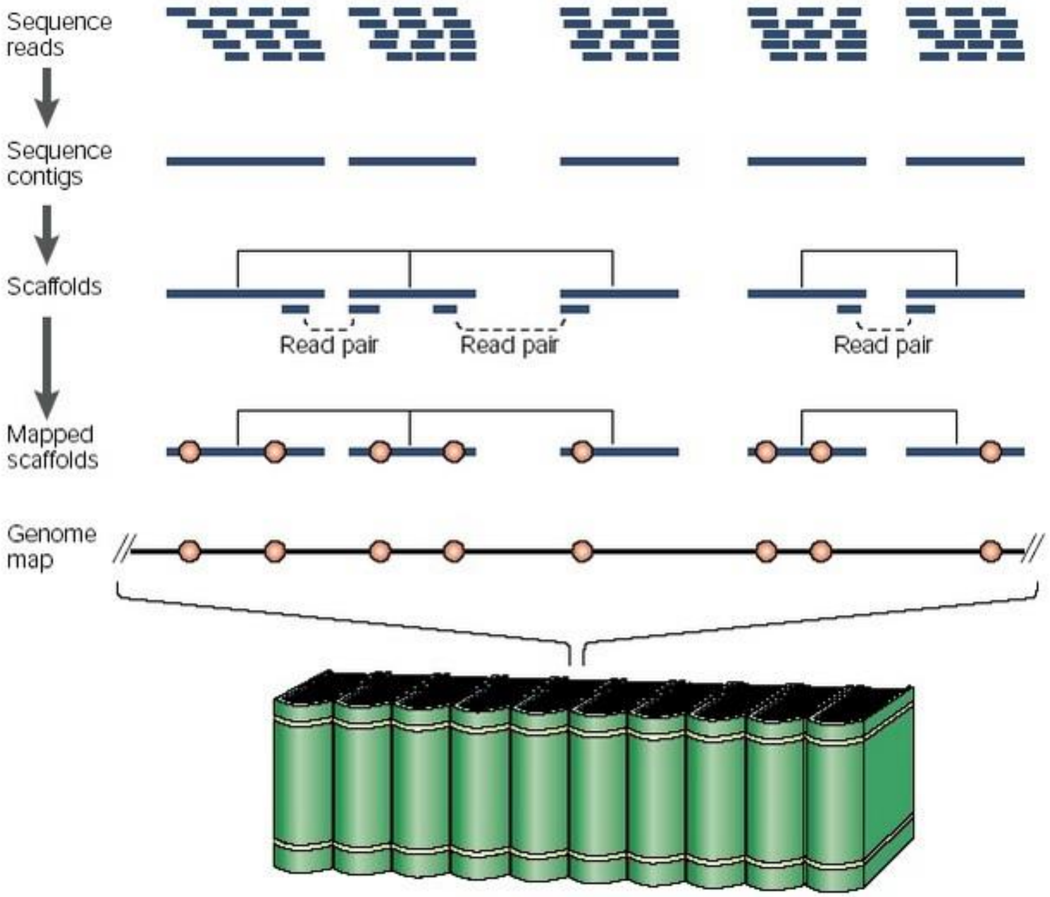
https://www.youtube.com/playlist?list=PL2mpR0RYFQsBiCWVJSvVAO3OJ2t7DzoHA

Pavel Pevzner UCSD

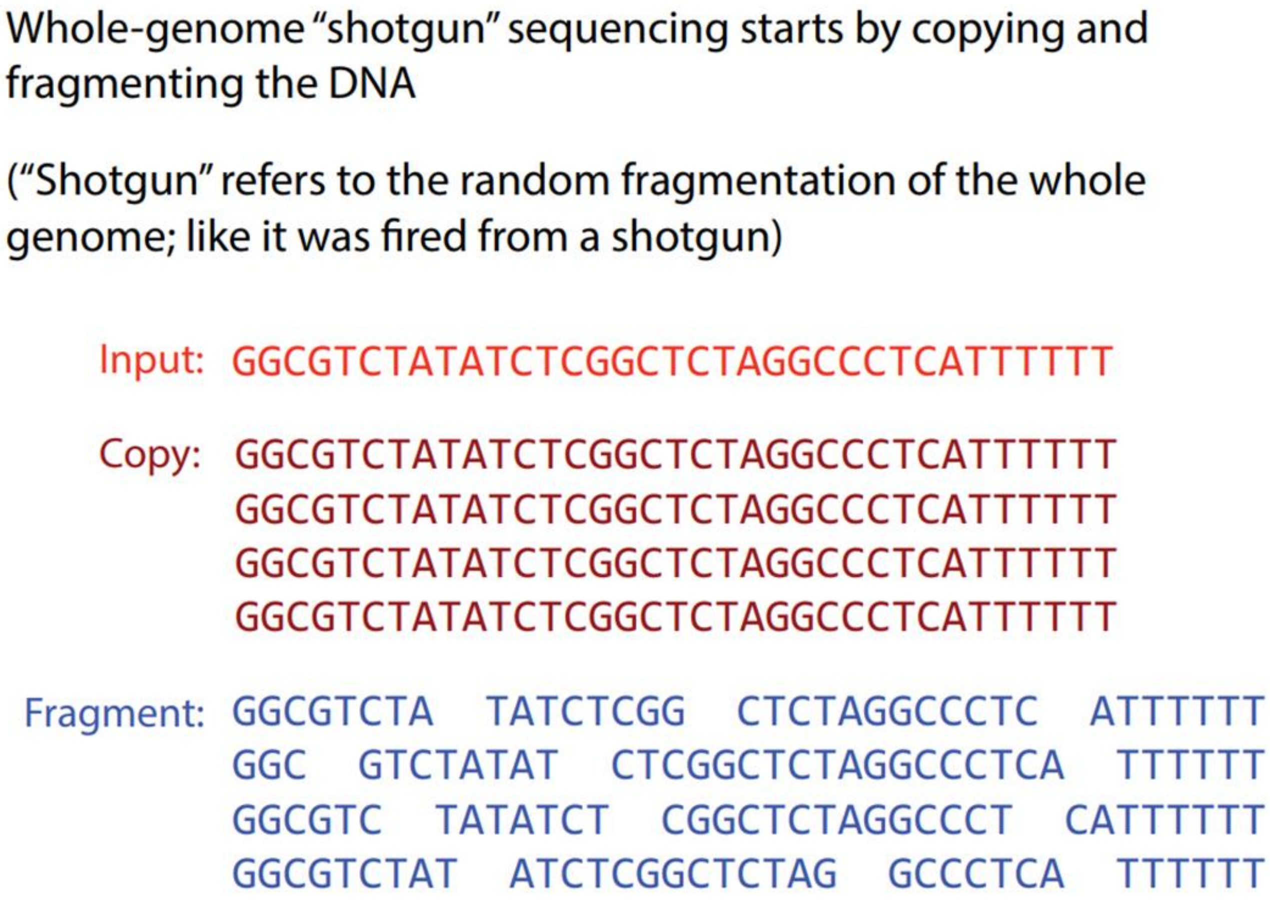
Bioinformatics Algorithms: An Active Learning Approach

https://www.youtube.com/watch?v=f-ecmECK7lw

Genome assembly



Genome assembly



Whole-genome "shotgun" sequencing starts by copying and fragmenting the DNA

("Shotgun" refers to the random fragmentation of the whole genome; like it was fired from a shotgun)

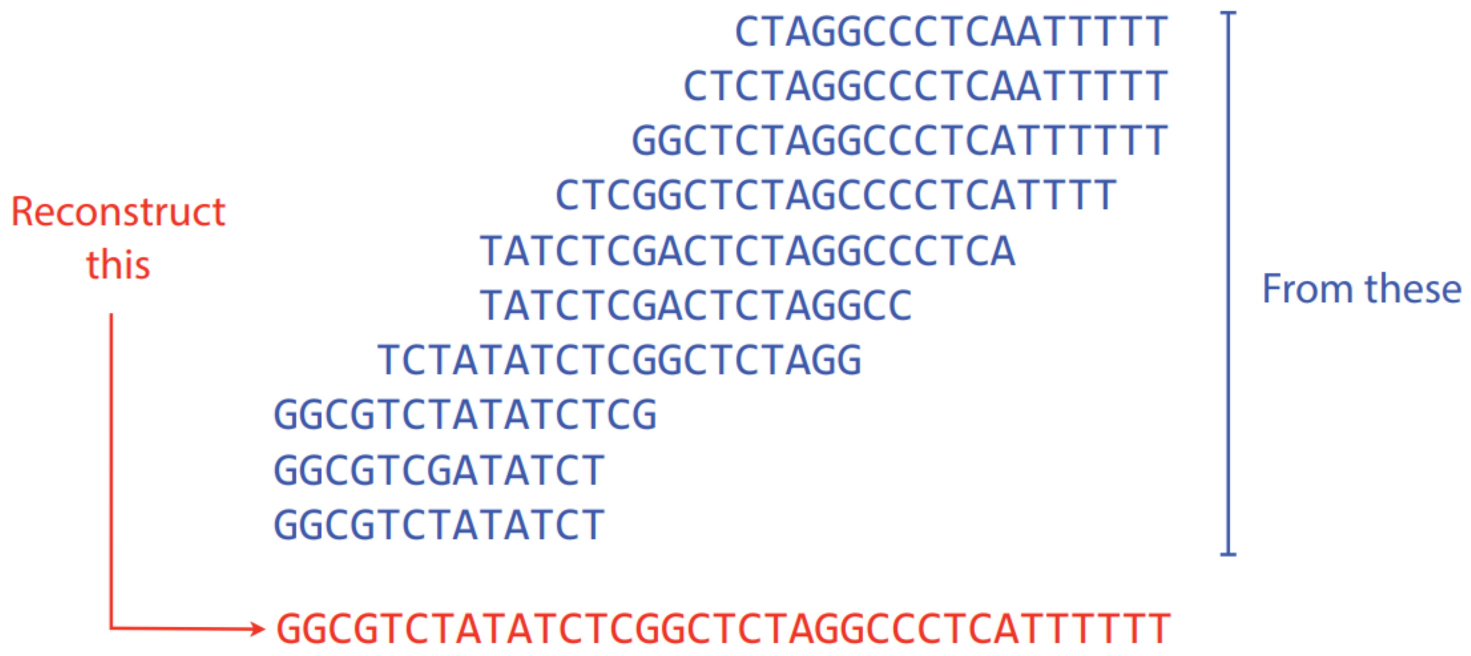
Input: GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT

Copy: GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT

Fragment: GGCGTCTA TATCTCGG CTCTAGGCCCTC ATTTTTT
GGC GTCTATAT CTCGGCTCTAGGCCCTCA TTTTTT
GGCGTC TATATCT CGGCTCTAGGCCCT CATTTTTT
GGCGTCTAT ATCTCGGCTCTAG GCCCTCA TTTTTT

Genome assembly

Assume sequencing produces such a large # fragments that almost all genome positions are *covered* by many fragments...



CTAGGCCCTCAATTTTT
CTCTAGGCCCTCAATTTTT
GGCTCTAGGCCCTCATTTTTT
CTCGGCTCTAGCCCCTCATTTT
TATCTCGACTCTAGGCCCTCA
TATCTCGACTCTAGGCC
TCTATATCTCGGCTCTAGG
GGCGTCTATATCTCG
GGCGTCGATATCT
GGCGTCTATATCT

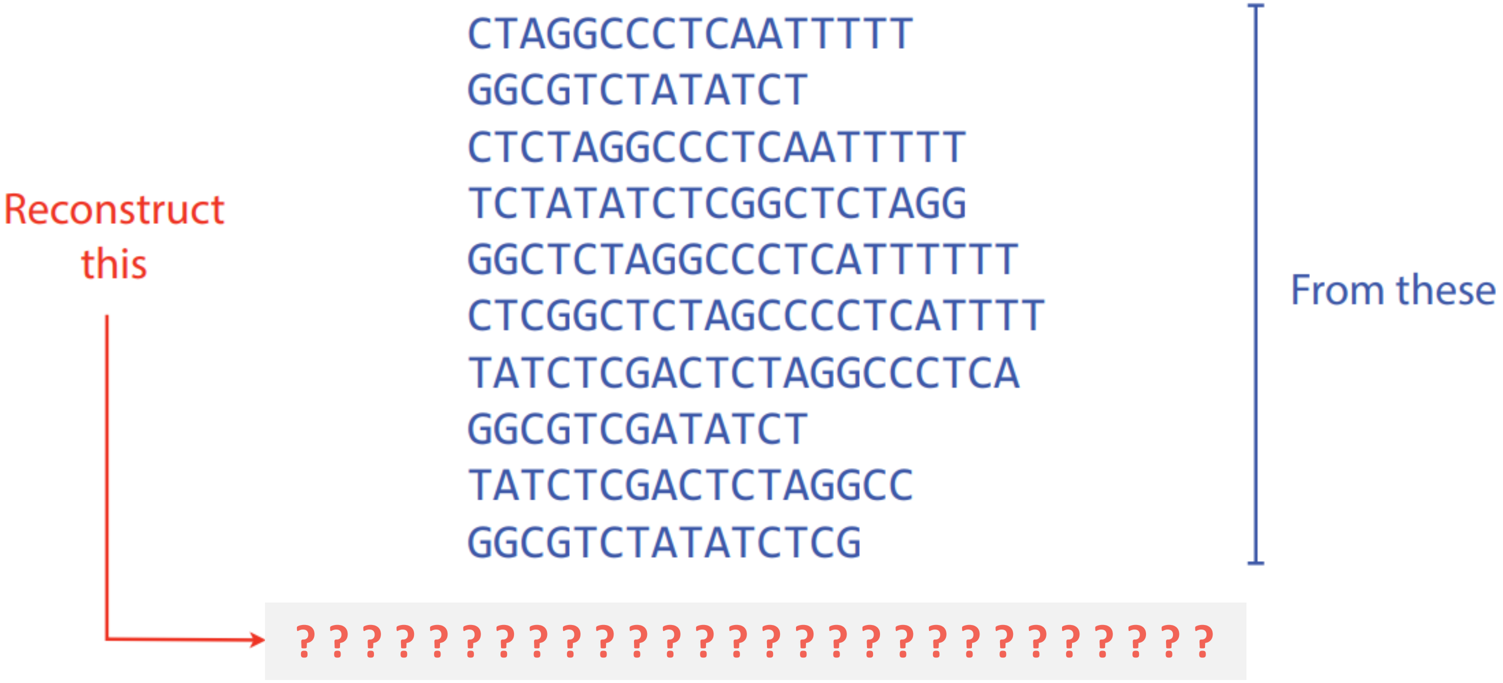
From these

Reconstruct this

GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT

Genome assembly

...but we don't know what came from where



Reconstruct this

CTAGGCCCTCAATTTTT
GGCGTCTATATCT
CTCTAGGCCCTCAATTTTT
TCTATATCTCGGCTCTAGG
GGCTCTAGGCCCTCATTTTTT
CTCGGCTCTAGCCCCTCATTTT
TATCTCGACTCTAGGCCCTCA
GGCGTCGATATCT
TATCTCGACTCTAGGCC
GGCGTCTATATCTCG

From these

? ?

Reference versus *de novo* assembly

- Reference based (guided) assembly
  - Assemble genome via comparison with reference genome assembly
  - More for re-sequencing … not really assembly per se
  - Uses short read mapping algorithms (treated elsewhere)
  - Most relevant for variant calling

- *De novo* (shotgun) assembly
  - Assemble genome based on sequence reads alone
  - Comparison between read sequences or k-mers
  - Graph traversal and genome reconstruction

Algorithm approaches for *de novo* assembly

- Overlap Layout Consensus (OLC)
  - Compares sequence reads to find overlaps
  - Construct directed read overlap graph
  - Trace (Hamiltonian) path through graph for assembly
  - Determine sequence of assembly via consensus of overlapped reads

- de Bruijn graph (DBG)
  - Parse reads into k-mers … sequence substrings of length k
  - Create directed k-mer graph by joining k-1 prefix-> suffix
  - Trace (Eulerian) path through graph for assembly
  - Determine sequence of assembly directly from k-mer graph

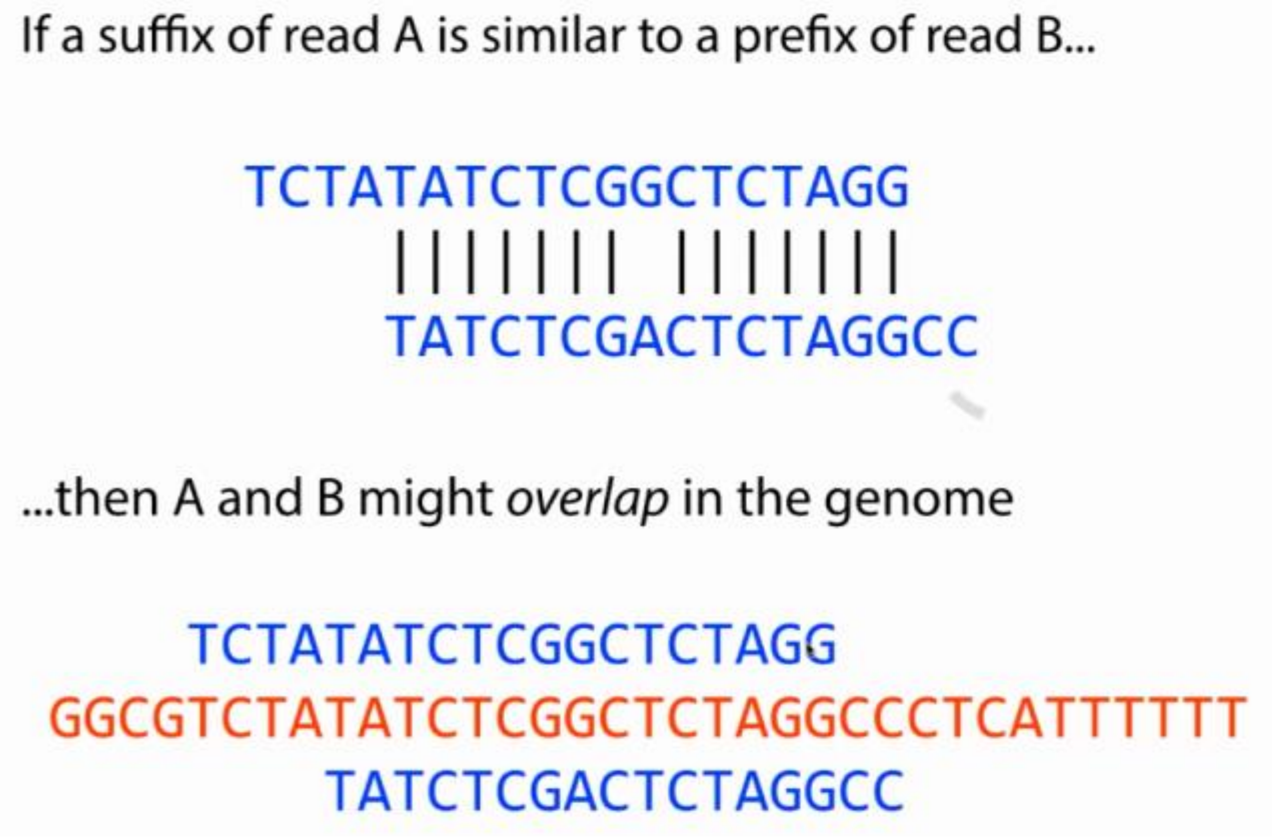
Overlap layout consensus (OLC)

- Overlap – build overlap graph from sequence reads
  - Pairwise comparison of all reads (computationally costly, particularly for repeats)
    - $O(N^2)$ where N is number of reads or $O(N \log N)$ at best with indexing
  - Join reads as nodes in directed graph if overlap exceeds threshold

- Layout – traverse overlap graph to join reads into contigs
  - Hamiltonian path problem – visit each node in graph exactly once – NP hard problem

- Consensus – determine contig sequences by consensus (most common) bases at each position of overlapped reads

Read Overlaps



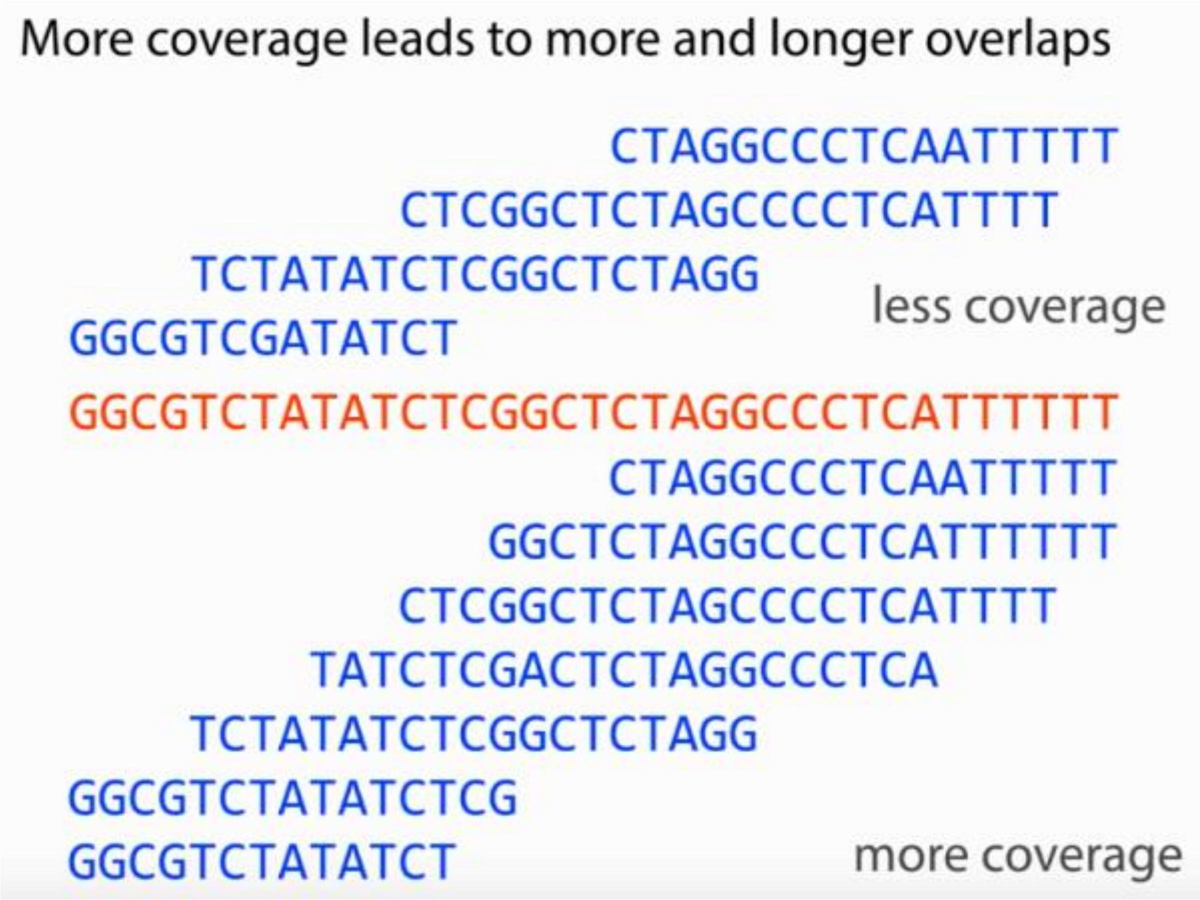
If a suffix of read A is similar to a prefix of read B...

```
TCTATATCTCGGCTCTAGG
    ||||||| |||||||
    TATCTCGACTCTAGGCC
```

...then A and B might *overlap* in the genome

```
    TCTATATCTCGGCTCTAGG
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT
        TATCTCGACTCTAGGCC
```

Read Overlaps



More coverage leads to more and longer overlaps

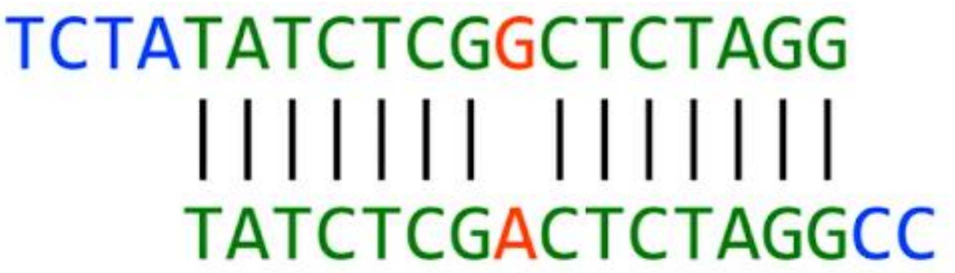
```
                       CTAGGCCCTCAATTTTT
                   CTCGGCTCTAGCCCCTCATTTT
       TCTATATCTCGGCTCTAGG
 GGCGTCGATATCT
```

less coverage

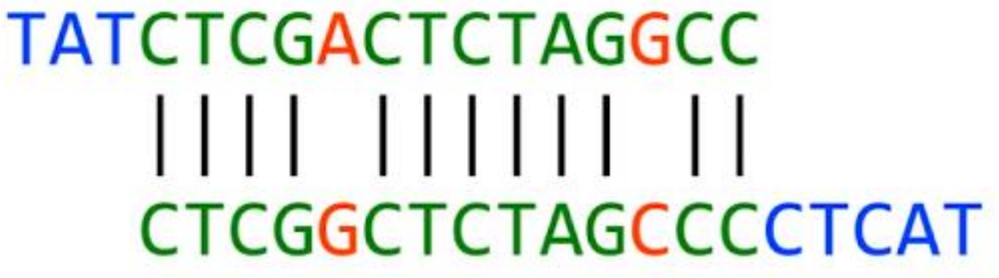
```
 GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT
                       CTAGGCCCTCAATTTTT
                     GGCTCTAGGCCCTCATTTTTT
                   CTCGGCTCTAGCCCCTCATTTT
               TATCTCGACTCTAGGCCCTCA
       TCTATATCTCGGCTCTAGG
 GGCGTCTATATCTCG
 GGCGTCTATATCT
```

more coverage

Overlap graph



```
TCTATATCTCGGCTCTAGG
    ||||||| |||||||
    TATCTCGACTCTAGGCC
```



```
TATCTCGACTCTAGGCC
   |||| |||||| ||
   CTCGGCTCTAGCCCCTCAT
```

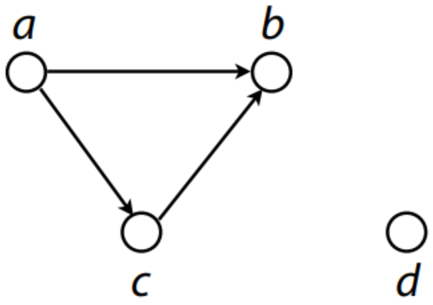
Directed graphs

Directed graph $G(V, E)$ consists of set of *vertices, V* and set of *directed edges, E*

Directed edge is an *ordered pair* of vertices. First is the *source*, second is the *sink*.

Vertex is drawn as a circle

Edge is drawn as a line with an arrow connecting two circles



Vertex also called *node* or *point*

Edge also called *arc* or *line*

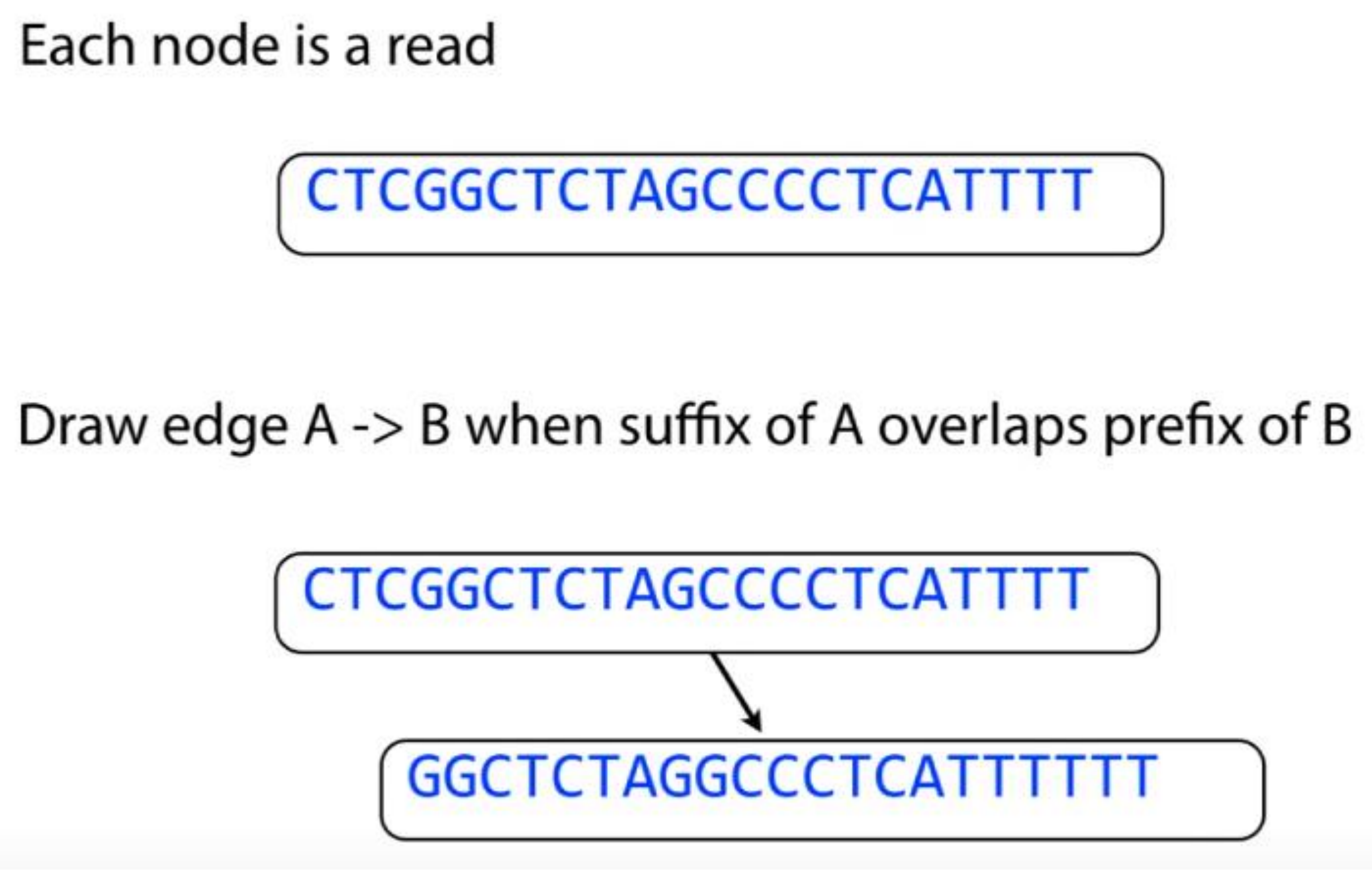
Directed graph also called *digraph*

$V = \{ a, b, c, d \}$

$E = \{ (a, b), (a, c), (c, b) \}$

Source Sink

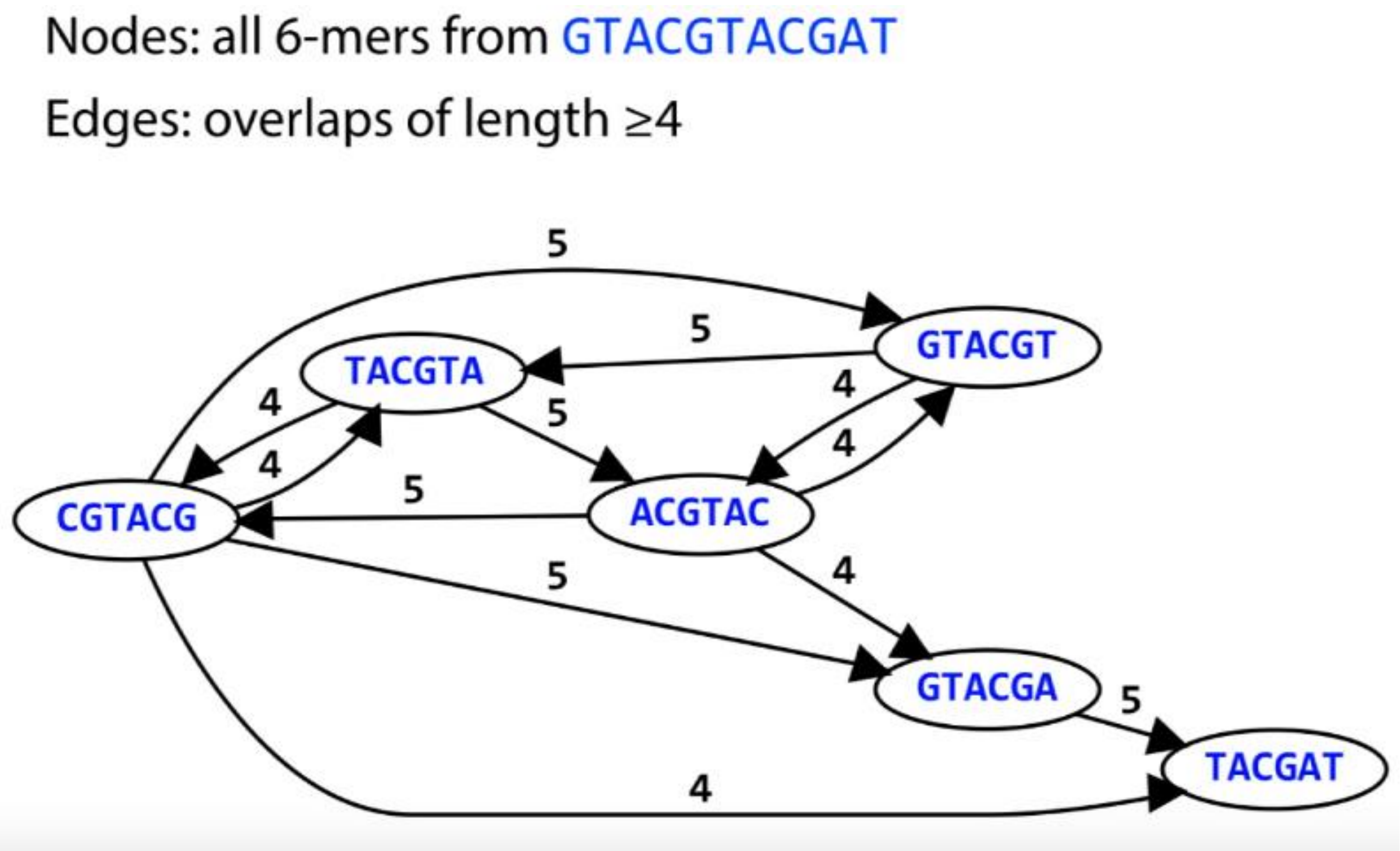
Overlap graph



Each node is a read

Draw edge A -> B when suffix of A overlaps prefix of B

Overlap graph



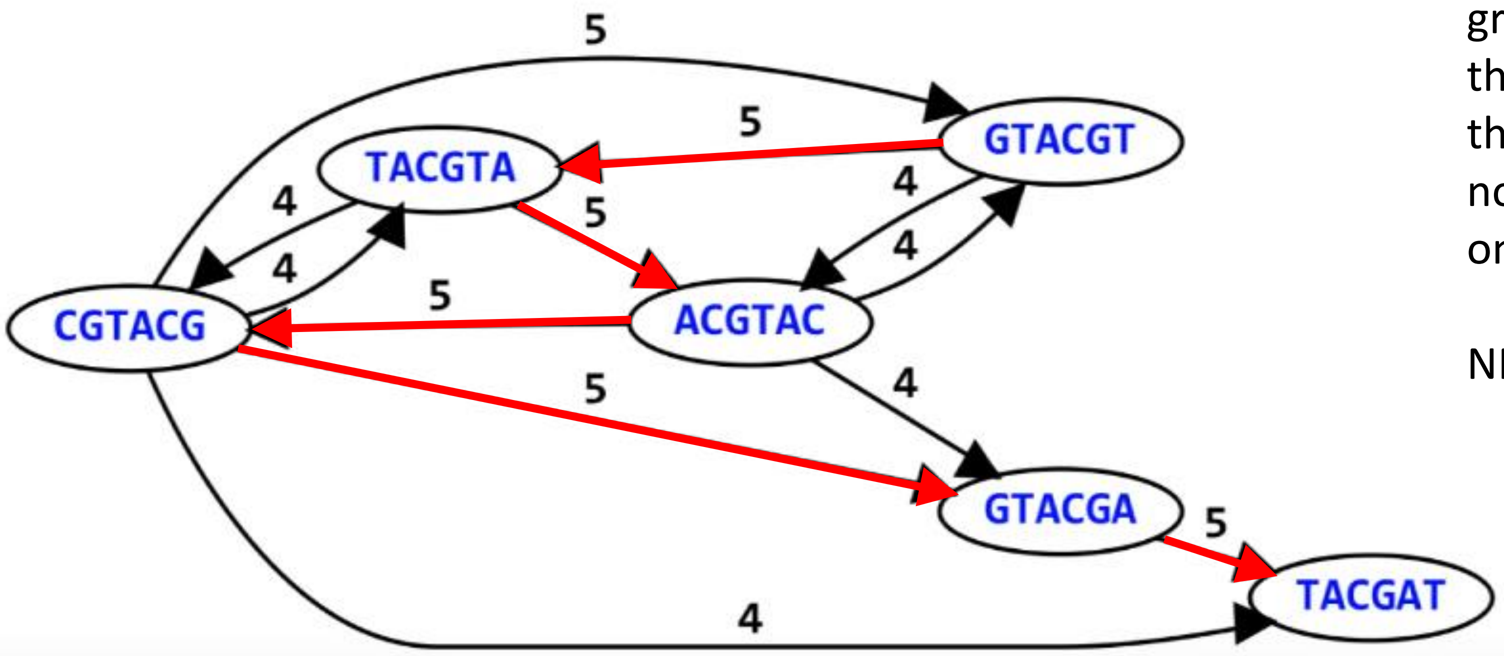
Nodes: all 6-mers from GTACGTACGAT

Edges: overlaps of length ≥4

Layout – graph traversal for assembly

Nodes: all 6-mers from GTACGTACGAT

Edges: overlaps of length ≥4

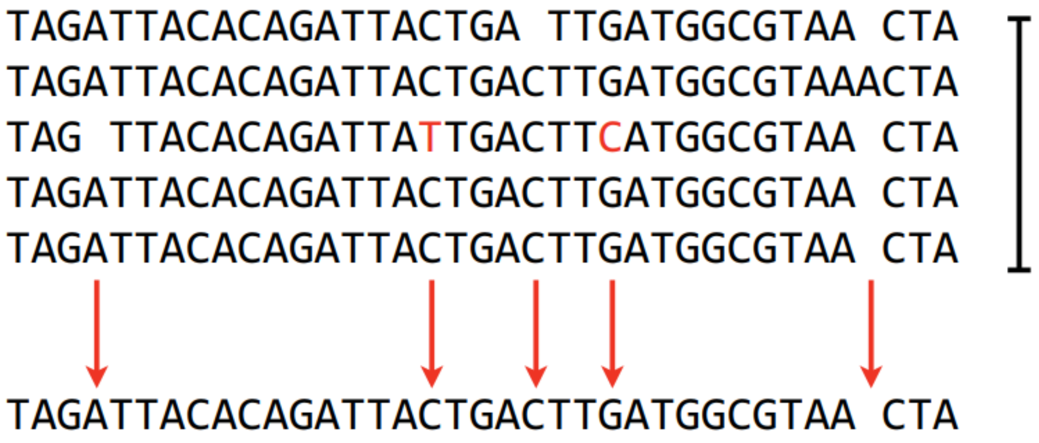


Hamiltonian path

graph traversal that passes through each node (read) only once

NP complete

Consensus



```
TAGATTACACAGATTACTGA TTGATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAAACTA
TAG TTACACAGATTATTGACTTCATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAA CTA

TAGATTACACAGATTACTGACTTGATGGCGTAA CTA
```

Take reads that make up a contig and line them up

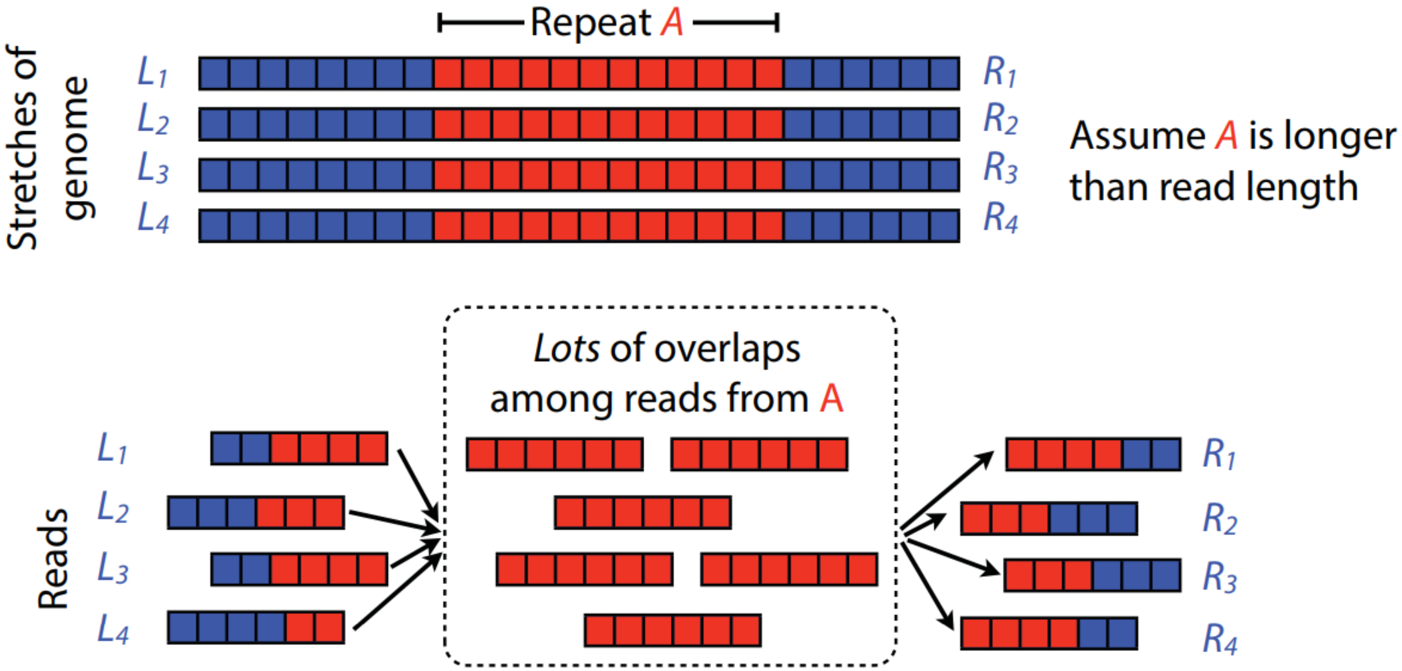
Take *consensus*, i.e. majority vote

At each position, ask: what nucleotide (and/or gap) is here?

Complications: (a) sequencing error, (b) ploidy

The challenge of repeats

Picture the portion of the overlap graph involving repeat *A*



Assume *A* is longer than read length

Lots of overlaps among reads from A

Even if we avoid collapsing copies of *A*, we can't know which paths *in* correspond to which paths *out*

Overlap Layout Consensus (OLC)

- Computationally costly and slow
  - Overlap – All-against-all read pair comparison – O(N^2)
  - Layout – Hamiltonian path problem is NP complete
  - Repeats break assembly
  - Better for long reads (comeback with third generation sequencing?)

de Bruijn graph (DBG)

- Parse reads into k-mers … sequence substrings of length k
    - Counter intuitive … trying to assemble long contigs by breaking sequences down even further
- Create directed k-mer graph by joining k-1 prefix -> suffix
- Trace (Eulerian) path through graph for assembly
- Determine sequence of assembly directly from k-mer graph

k-mers

"*k*-mer" is a substring of length *k*

S: GGCGATTCATCG

mer: from Greek meaning "part"

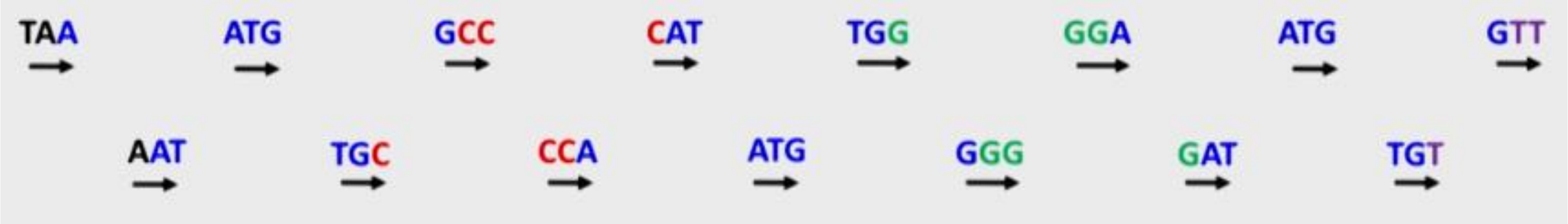
A 4-mer of *S*: ATTC

All 3-mers of *S*: GGC
GCG
CGA
GAT
ATT
TTC
TCA
CAT
ATC
TCG

I'll use "*k*-1-mer" to refer to a substring of length *k* - 1

de Bruijn graph construction

- Break sequences down into k-mers (3-mers), which will be edges in graph

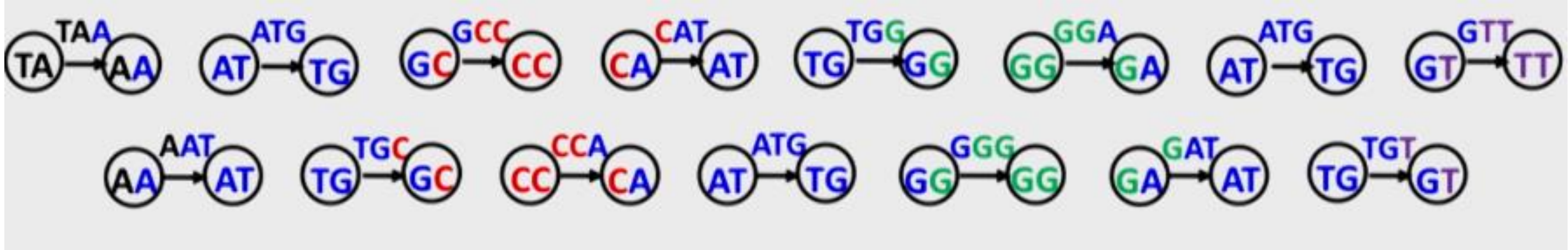


de Bruijn graph construction

- Represent every k-mer as an edge between its prefix and suffix k-1-mers
- Collapse all nodes with identical labels

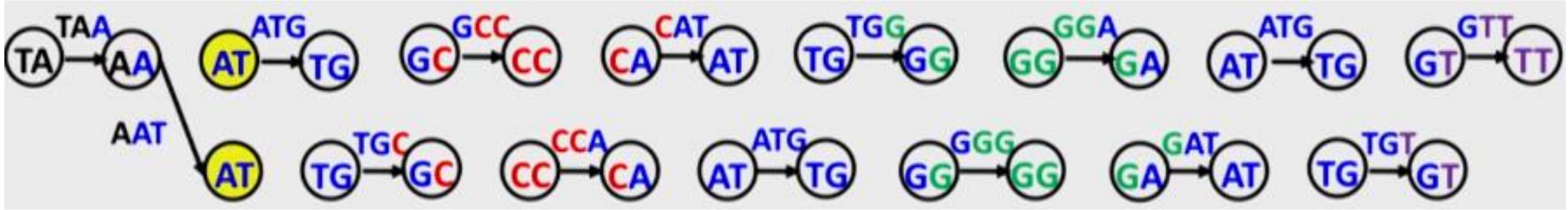
de Bruijn graph construction

- Construct nodes of graph by connecting k-1-mer prefixes and suffixes



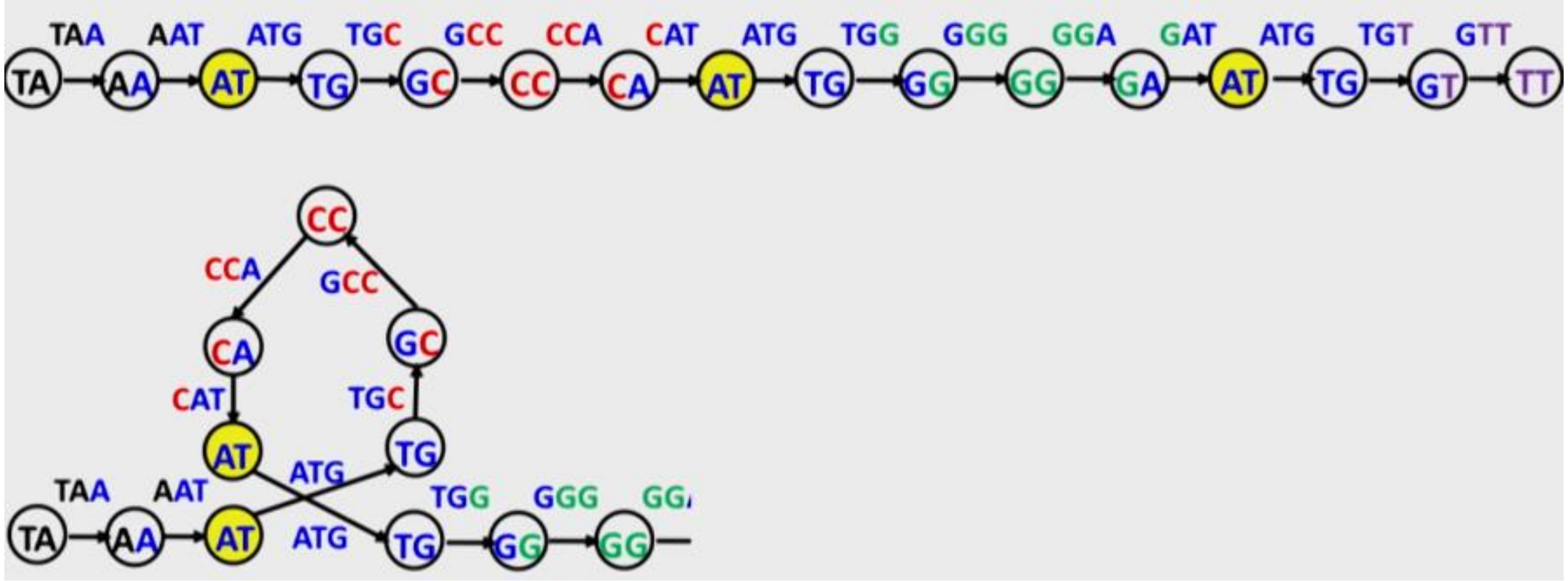
de Bruijn graph construction

- Collapse identically labeled nodes

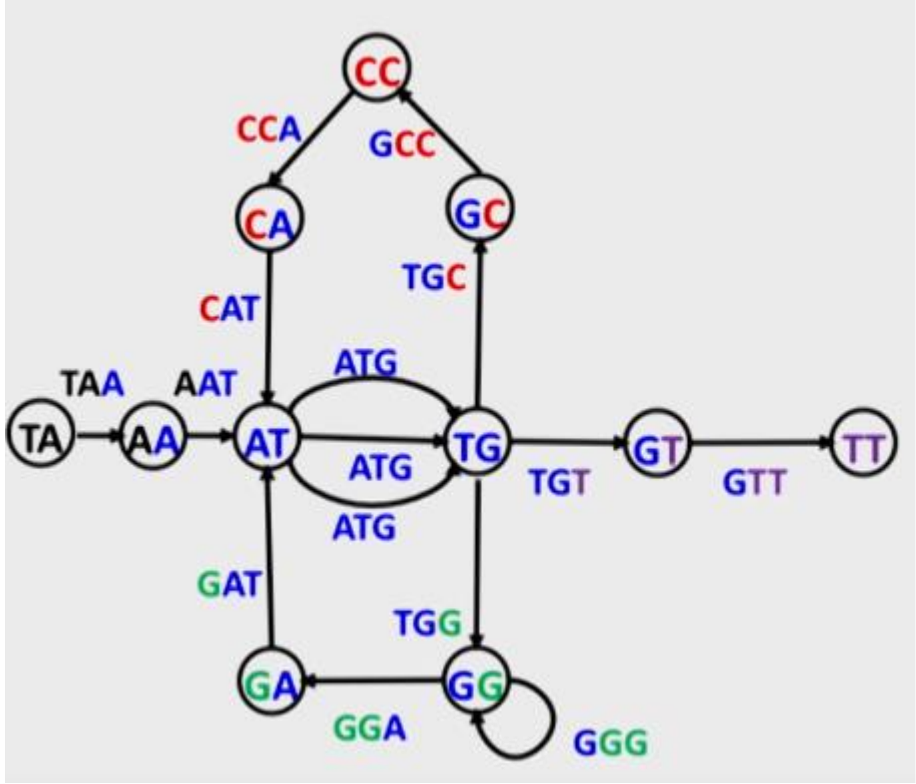


de Bruijn graph construction

- Collapse identically labeled nodes



Find Eulerian path through de Bruijn graph

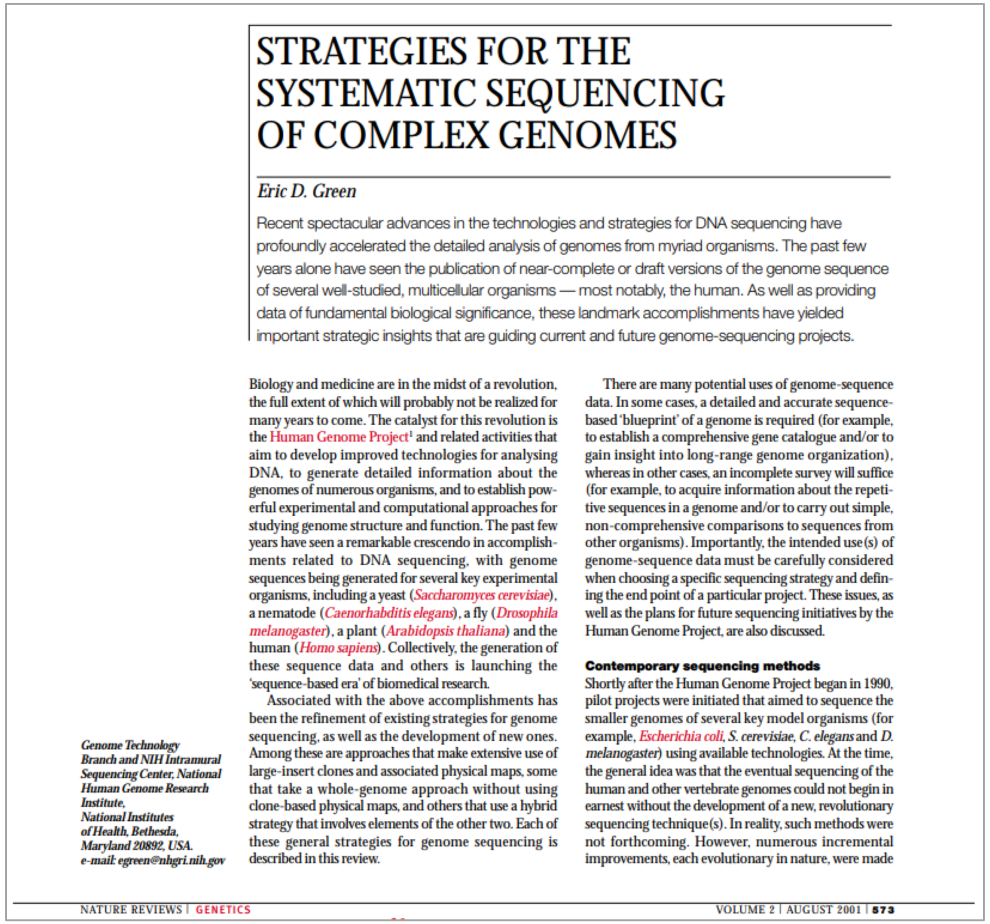


- Can be multiple Eulerian paths through graph
- Solution: disconnect graph into multiple (non-branching) components = contigs
- Contigs can then be connected using read-pair information
- Helps specify best path through branching paths in graph

de Bruijn graph (DBC)

- Computationally more efficient and faster
  - Scales linearly as O(N), where N is the number of k-mers
  - Eulerian path problem is more tractable than Hamiltonian path
  - Repeats break assembly
  - Very sensitive to sequencing errors (greatly inflates # of k-mers)
  - Loses sequence context of reads
  - Better for short reads (how will this be adopted for third generation sequencing?)
  - Choice of k-mer size very important
    - Empirical decision – try out different sizes
    - Odd k-mer size to avoid palindromes
    - Longer k-mers better resolution but use much more memory

References



STRATEGIES FOR THE SYSTEMATIC SEQUENCING OF COMPLEX GENOMES

Eric D. Green

Recent spectacular advances in the technologies and strategies for DNA sequencing have profoundly accelerated the detailed analysis of genomes from myriad organisms. The past few years alone have seen the publication of near-complete or draft versions of the genome sequence of several well-studied, multicellular organisms — most notably, the human. As well as providing data of fundamental biological significance, these landmark accomplishments have yielded important strategic insights that are guiding current and future genome-sequencing projects.

Biology and medicine are in the midst of a revolution, the full extent of which will probably not be realized for many years to come. The catalyst for this revolution is the Human Genome Project[1] and related activities that aim to develop improved technologies for analysing DNA, to generate detailed information about the genomes of numerous organisms, and to establish powerful experimental and computational approaches for studying genome structure and function. The past few years have seen a remarkable crescendo in accomplishments related to DNA sequencing, with genome sequences being generated for several key experimental organisms, including a yeast (*Saccharomyces cerevisiae*), a nematode (*Caenorhabditis elegans*), a fly (*Drosophila melanogaster*), a plant (*Arabidopsis thaliana*) and the human (*Homo sapiens*). Collectively, the generation of these sequence data and others is launching the 'sequence-based era' of biomedical research.

Associated with the above accomplishments has been the refinement of existing strategies for genome sequencing, as well as the development of new ones. Among these are approaches that make extensive use of large-insert clones and associated physical maps, some that take a whole-genome approach without using clone-based physical maps, and others that use a hybrid strategy that involves elements of the other two. Each of these general strategies for genome sequencing is described in this review.

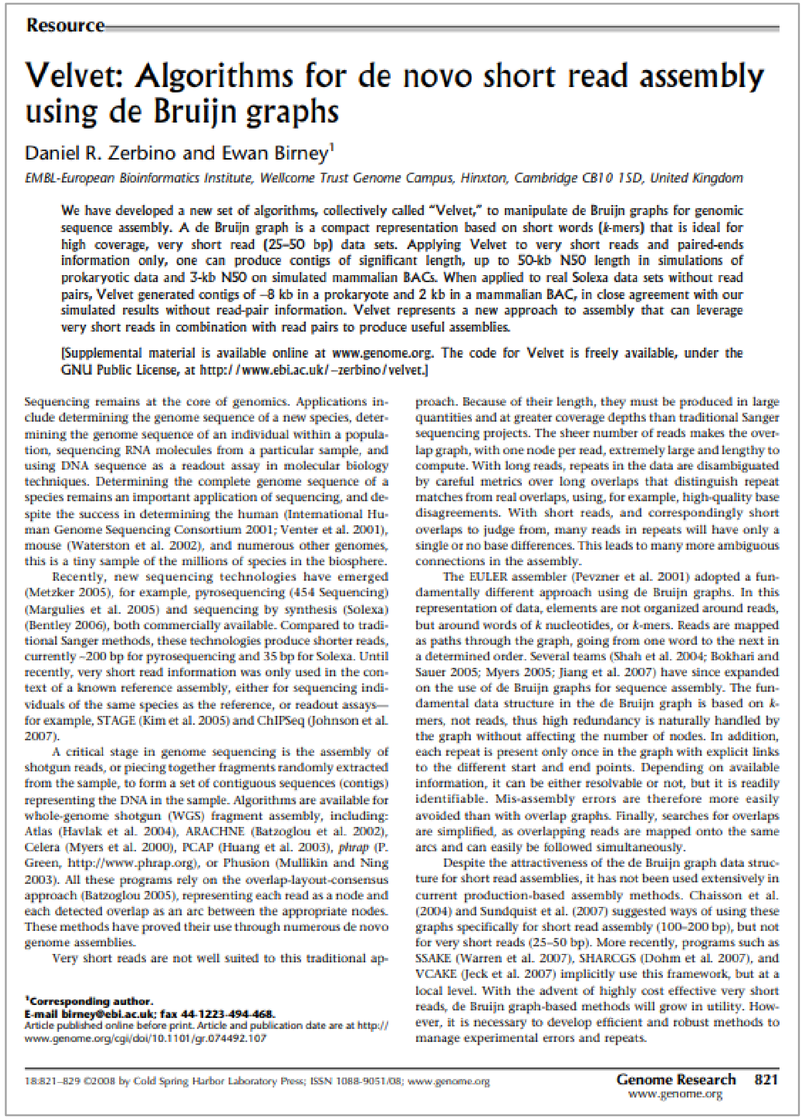
There are many potential uses of genome-sequence data. In some cases, a detailed and accurate sequence-based 'blueprint' of a genome is required (for example, to establish a comprehensive gene catalogue and/or to gain insight into long-range genome organization), whereas in other cases, an incomplete survey will suffice (for example, to acquire information about the repetitive sequences in a genome and/or to carry out simple, non-comprehensive comparisons to sequences from other organisms). Importantly, the intended use(s) of genome-sequence data must be carefully considered when choosing a specific sequencing strategy and defining the end point of a particular project. These issues, as well as the plans for future sequencing initiatives by the Human Genome Project, are also discussed.

Contemporary sequencing methods

Shortly after the Human Genome Project began in 1990, pilot projects were initiated that aimed to sequence the smaller genomes of several key model organisms (for example, *Escherichia coli*, *S. cerevisiae*, *C. elegans* and *D. melanogaster*) using available technologies. At the time, the general idea was that the eventual sequencing of the human and other vertebrate genomes could not begin in earnest without the development of a new, revolutionary sequencing technique(s). In reality, such methods were not forthcoming. However, numerous incremental improvements, each evolutionary in nature, were made

Genome Technology Branch and NIH Intramural Sequencing Center, National Human Genome Research Institute,
National Institutes of Health, Bethesda, Maryland 20892, USA.
e-mail: egreen@nhgri.nih.gov

NATURE REVIEWS | GENETICS VOLUME 2 | AUGUST 2001 | 573



Resource

Velvet: Algorithms for de novo short read assembly using de Bruijn graphs

Daniel R. Zerbino and Ewan Birney[1]

EMBL-European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, United Kingdom

We have developed a new set of algorithms, collectively called "Velvet," to manipulate de Bruijn graphs for genomic sequence assembly. A de Bruijn graph is a compact representation based on short words (*k*-mers) that is ideal for high coverage, very short read (25–50 bp) data sets. Applying Velvet to very short reads and paired-ends information only, one can produce contigs of significant length, up to 50-kb N50 length in simulations of prokaryotic data and 3-kb N50 on simulated mammalian BACs. When applied to real Solexa data sets without read pairs, Velvet generated contigs of ~8 kb in a prokaryote and 2 kb in a mammalian BAC, in close agreement with our simulated results without read-pair information. Velvet represents a new approach to assembly that can leverage very short reads in combination with read pairs to produce useful assemblies.

[Supplemental material is available online at www.genome.org. The code for Velvet is freely available, under the GNU Public License, at http://www.ebi.ac.uk/~zerbino/velvet.]

Sequencing remains at the core of genomics. Applications include determining the genome sequence of a new species, determining the genome sequence of an individual within a population, sequencing RNA molecules from a particular sample, and using DNA sequence as a readout assay in molecular biology techniques. Determining the complete genome sequence of a species remains an important application of sequencing, and despite the success in determining the human (International Human Genome Sequencing Consortium 2001; Venter et al. 2001), mouse (Waterston et al. 2002), and numerous other genomes, this is a tiny sample of the millions of species in the biosphere.

Recently, new sequencing technologies have emerged (Metzker 2005), for example, pyrosequencing (454 Sequencing) (Margulies et al. 2005) and sequencing by synthesis (Solexa) (Bentley 2006), both commercially available. Compared to traditional Sanger methods, these technologies produce shorter reads, currently ~200 bp for pyrosequencing and 35 bp for Solexa. Until recently, very short read information was only used in the context of a known reference assembly, either for sequencing individuals of the same species as the reference, or readout assays—for example, STAGE (Kim et al. 2005) and ChIPSeq (Johnson et al. 2007).

A critical stage in genome sequencing is the assembly of shotgun reads, or piecing together fragments randomly extracted from the sample, to form a set of contiguous sequences (contigs) representing the DNA in the sample. Algorithms are available for whole-genome shotgun (WGS) fragment assembly, including: Atlas (Havlak et al. 2004), ARACHNE (Batzoglou et al. 2002), Celera (Myers et al. 2000), PCAP (Huang et al. 2003), *phrap* (P. Green, http://www.phrap.org), or Phusion (Mullikin and Ning 2003). All these programs rely on the overlap-layout-consensus approach (Batzoglou 2005), representing each read as a node and each detected overlap as an arc between the appropriate nodes. These methods have proved their use through numerous de novo genome assemblies.

Very short reads are not well suited to this traditional approach. Because of their length, they must be produced in large quantities and at greater coverage depths than traditional Sanger sequencing projects. The sheer number of reads makes the overlap graph, with one node per read, extremely large and lengthy to compute. With long reads, repeats in the data are disambiguated by careful metrics over long overlaps that distinguish repeat matches from real overlaps, using, for example, high-quality base disagreements. With short reads, and correspondingly short overlaps to judge from, many reads in repeats will have only a single or no base differences. This leads to many more ambiguous connections in the assembly.

The EULER assembler (Pevzner et al. 2001) adopted a fundamentally different approach using de Bruijn graphs. In this representation of data, elements are not organized around reads, but around words of *k* nucleotides, or *k*-mers. Reads are mapped as paths through the graph, going from one word to the next in a determined order. Several teams (Shah et al. 2004; Bokhari and Sauer 2005; Myers 2005; Jiang et al. 2007) have since expanded on the use of de Bruijn graphs for sequence assembly. The fundamental data structure in the de Bruijn graph is based on *k*-mers, not reads, thus high redundancy is naturally handled by the graph without affecting the number of nodes. In addition, each repeat is present only once in the graph with explicit links to the different start and end points. Depending on available information, it can be either resolvable or not, but it is readily identifiable. Mis-assembly errors are therefore more easily avoided than with overlap graphs. Finally, searches for overlaps are simplified, as overlapping reads are mapped onto the same arcs and can easily be followed simultaneously.

Despite the attractiveness of the de Bruijn graph data structure for short read assemblies, it has not been used extensively in current production-based assembly methods. Chaisson et al. (2004) and Sundquist et al. (2007) suggested ways of using these graphs specifically for short read assembly (100–200 bp), but not for very short reads (25–50 bp). More recently, programs such as SSAKE (Warren et al. 2007), SHARCGS (Dohm et al. 2007), and VCAKE (Jeck et al. 2007) implicitly use this framework, but at a local level. With the advent of highly cost effective very short reads, de Bruijn graph-based methods will grow in utility. However, it is necessary to develop efficient and robust methods to manage experimental errors and repeats.

[1]**Corresponding author.**
E-mail birney@ebi.ac.uk; fax 44-1223-494-468.
Article published online before print. Article and publication date are at http://www.genome.org/cgi/doi/10.1101/gr.074492.107

18:821–829 ©2008 by Cold Spring Harbor Laboratory Press; ISSN 1088-9051/08; www.genome.org

Genome Research 821
www.genome.org

Web resources

- MIT: Foundations of Computational and Systems Biology

https://ocw.mit.edu/courses/biology/7-91j-foundations-of-computational-and-systems-biology-spring-2014/

- JHU: Ben Langmead Teaching Materials

http://www.langmead-lab.org/teaching-materials/

- UCSD: Bioinformatics Algorithms: An Active Learning Approach

https://www.youtube.com/channel/UCKSUVRs2N2FdDNvQoRWKhoQ

Outline

- Historical context for sequencing and NGS
  - Sanger sequencing
  - Roche’s 454
  - Illumina Sequencing
  - PacBio + Oxford Nanopore
- FASTQ and quality scores
- Quality control
  - FASTQC
- Genome assembly
  - Reference versus *de novo* assembly
  - De novo assembly algorithmic paradigms
    - Overlap layout consensus
    - de Bruijn graphs
- Genome assembly metrics

Slides courtesy of:

ABiL
Applied BioInformatics Laboratory

A unique bioinformatics resource
for translation of molecular data into
actionable public health intelligence

http://www.abil.ihrc.com/

Assessing the “goodness” of an assembly

- The goal of genome assembly is to yield a finished contig which is exactly the same as the input microbial genome

- An assembly is deemed good if it can get as close to this goal as possible

- Goodness of assembly is measured by three parameters:

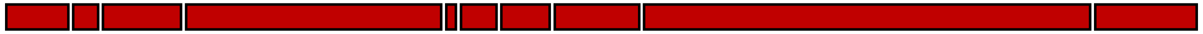
⬆ Number of bases assembled

⬇ Number of assembled contigs (length > 500bp)

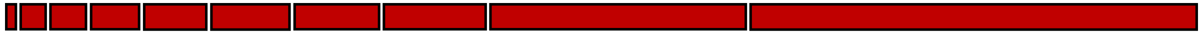
⬆ N50 value

N50

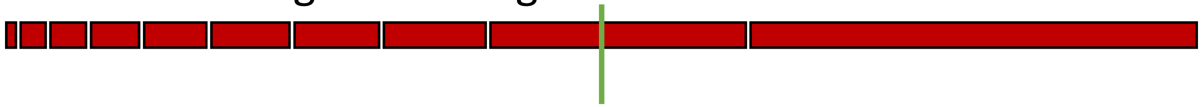
If these are the assembled contigs:



and I order them by length in an ascending manner,



N50 is the length of contig which is in the center



A good way to combine these three parameters

$$Assembly\ score = \log_{10}\left(\frac{Assembly\ size\ \times N50}{Number\ of\ contigs}\right)$$

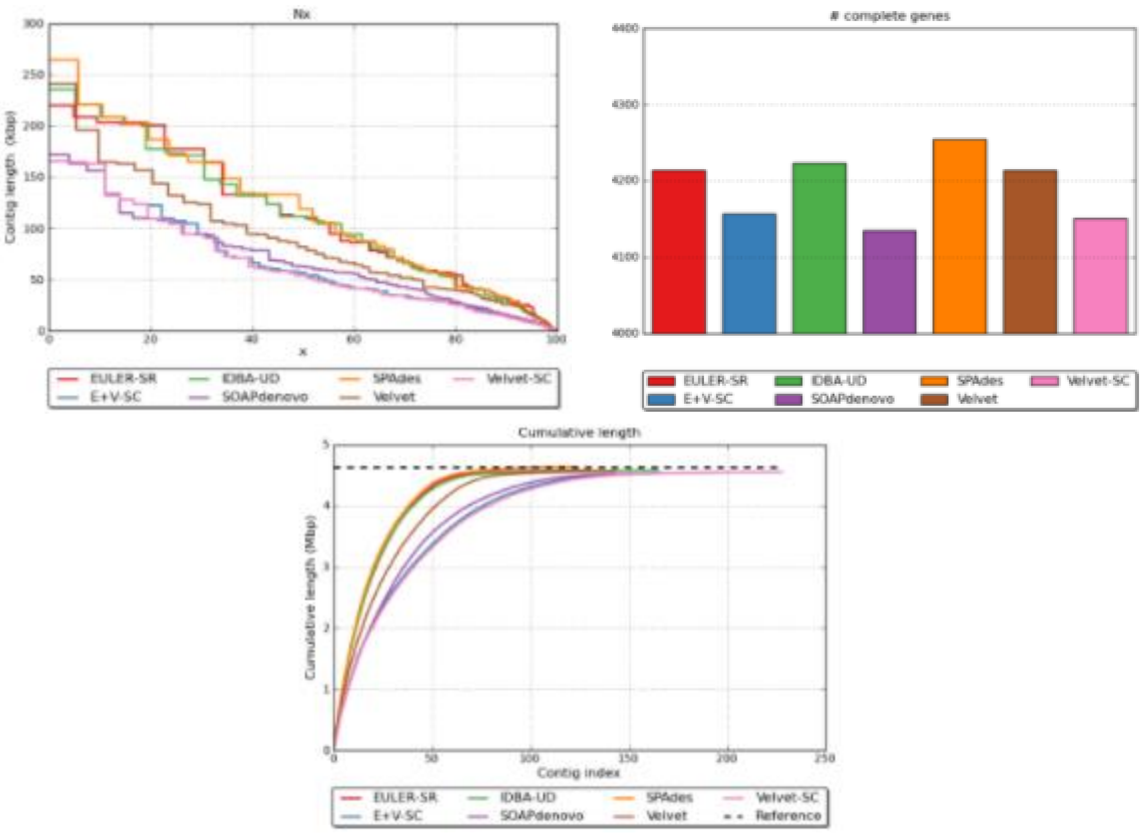
* Developed by Lee Katz and Lava

QUAST

- Excellent utility from the SPAdes group
- Produces nice graphical output of genome assembly sizes and assembly metrics
- Project contains 3 tools for assembly evaluation and comparison
  - QUAST: regular genome assemblies
  - MetaQUAST: metagenome assemblies
  - Icarus: contig alignment visualization

QUAST

Quality Assessment Tool for Genome Assemblies by CAB



Additional questions?