Calling and filtering complex small variants (indels, MNPs, and haplotypes)

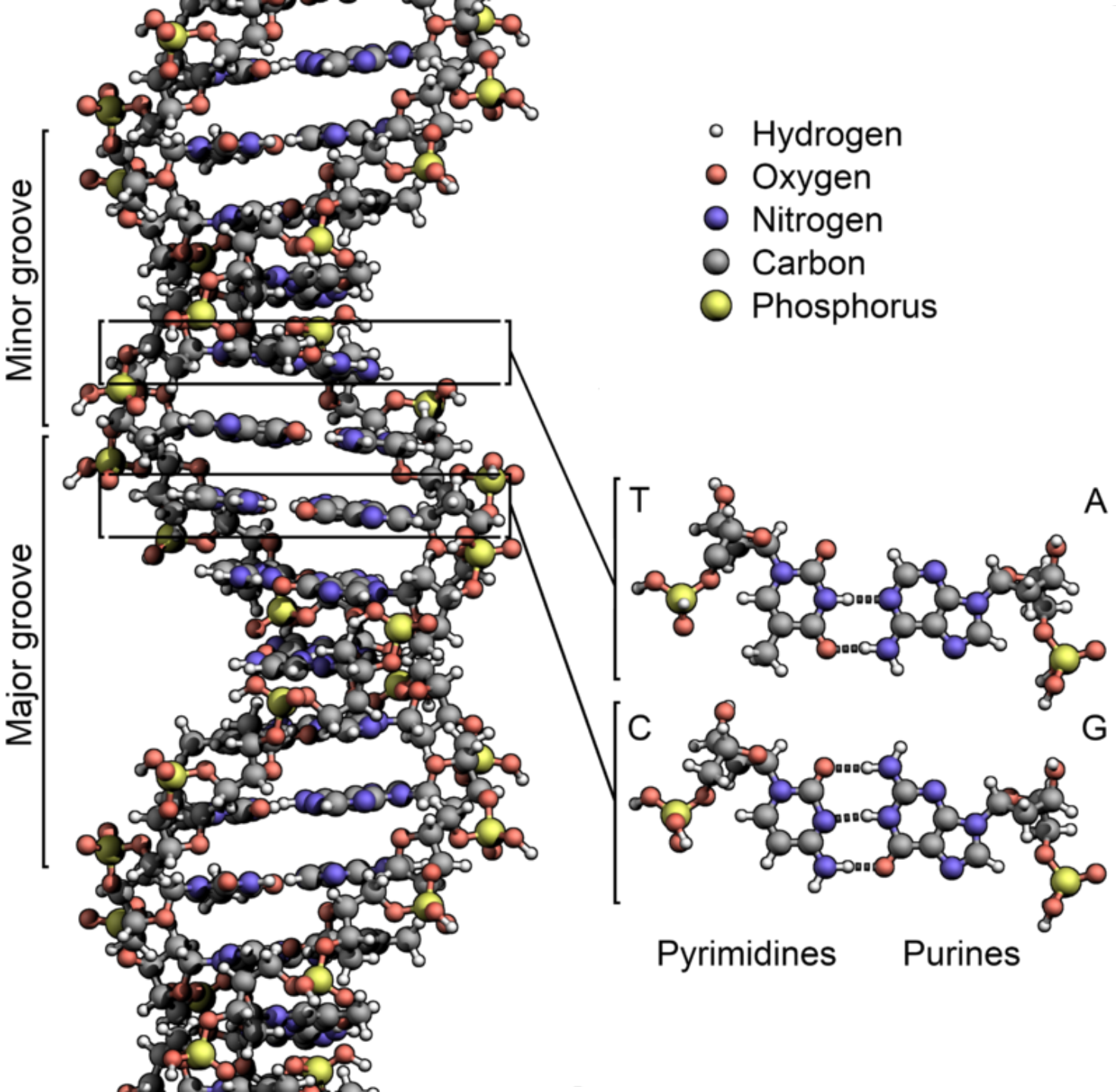
Erik Garrison
Wellcome Trust Sanger Institute
@ SeqShop, University of Michigan
May 20, 2015

UNIVERSITY OF CAMBRIDGE

Overview

1. **Genesis of variation (SNPs and indels)**
2. Causes of sequencing error
3. Sequence alignment
4. Alignment-based variant detection
5. Assembly-based variant detection
6. Haplotype-based variant detection
7. Graph-based methods

DNA


Hydrogen
Oxygen
Nitrogen
Carbon
Phosphorus
Minor groove
Major groove
T
A
C
G
Pyrimidines
Purines

A SNP

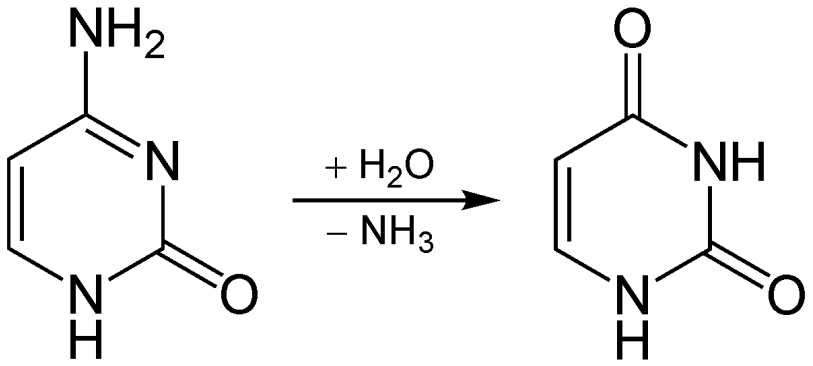
A point mutation in which one base is swapped for another.

AATTAG*C*CATTA

AATTAG*T*CATTA

(some) causes of SNPs

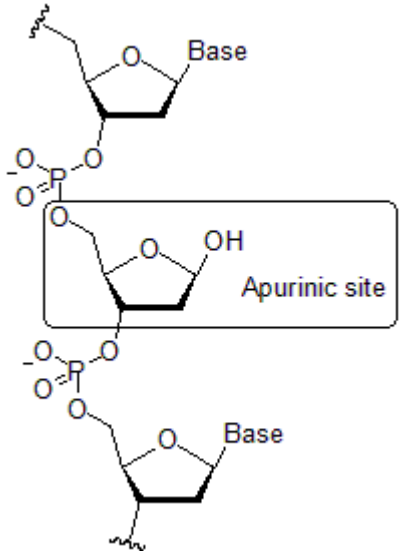
- Deamination
  - cytosine → uracil
  - 5-methylcytosine → thymine
  - guanine → xanthine (mispairs to A-T bp)
  - adenine → hypoxanthine (mispairs to G-C bp)

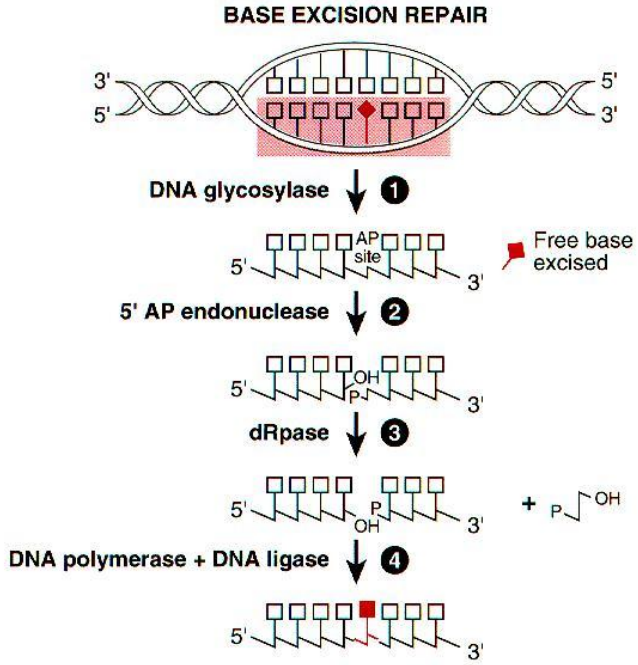


Deamination of Cytosine to Uracil
http://en.wikipedia.org/wiki/Deamination

(some) causes of SNPs

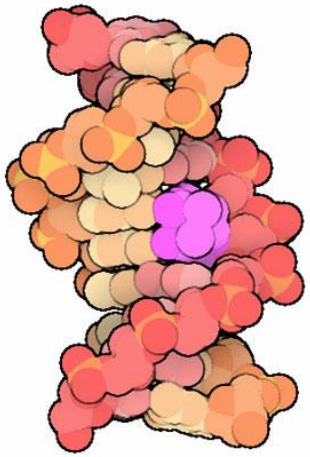
- Depurination
  - purines are cleaved from DNA sugar backbone (5000/cell/day, pyrimidines at much lower rate)
  - Base excision repair (BEP) can fail → mutation

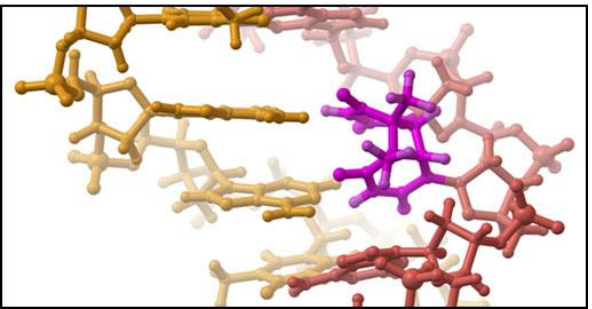


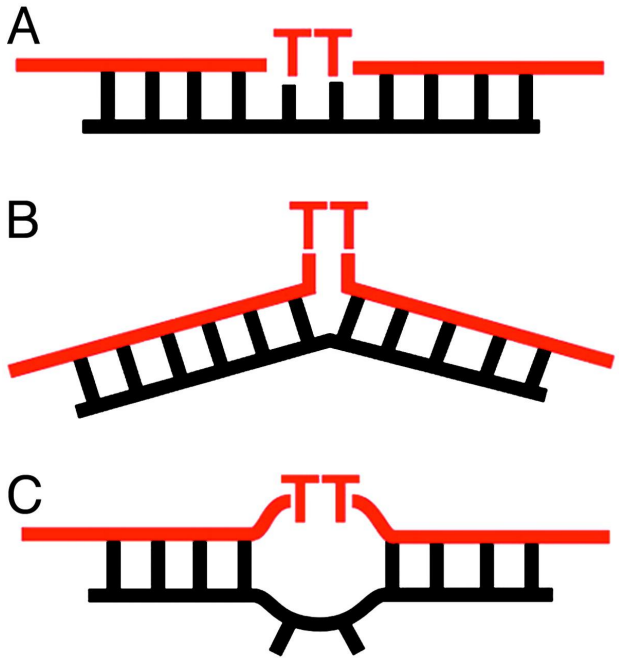


Multi-base events (MNPs)

- MNPs
  - thymine dimerization (UV induced)
  - other (e.g. oxidative stress induced)



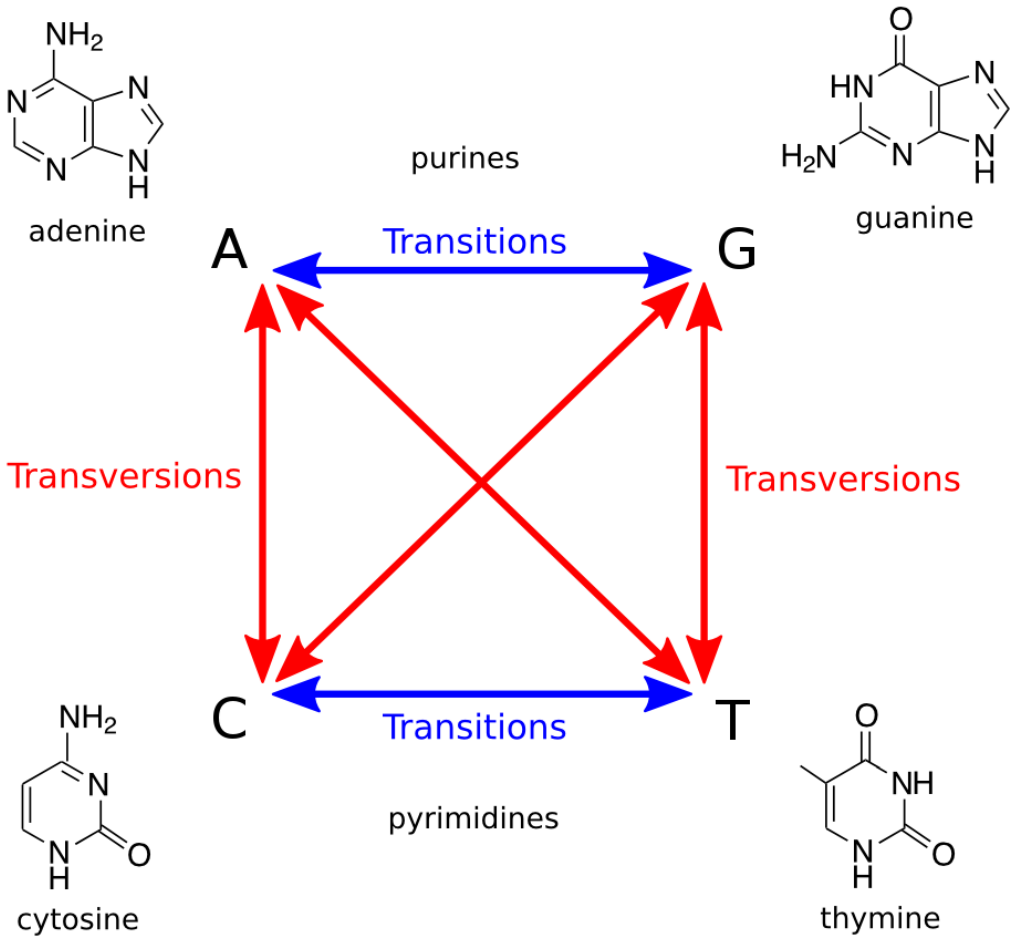




http://www.rcsb.org/pdb/101/motm.do?momID=91

Transitions and transversions

In general **transitions are 2-3 times more common than transversions**. (But this depends on biological context.)



https://upload.wikimedia.org/wikipedia/commons/3/35/Transitions-transversions-v3.png

An INDEL

A mutation that results from the gain or loss of sequence.

AATTAGCCATTA

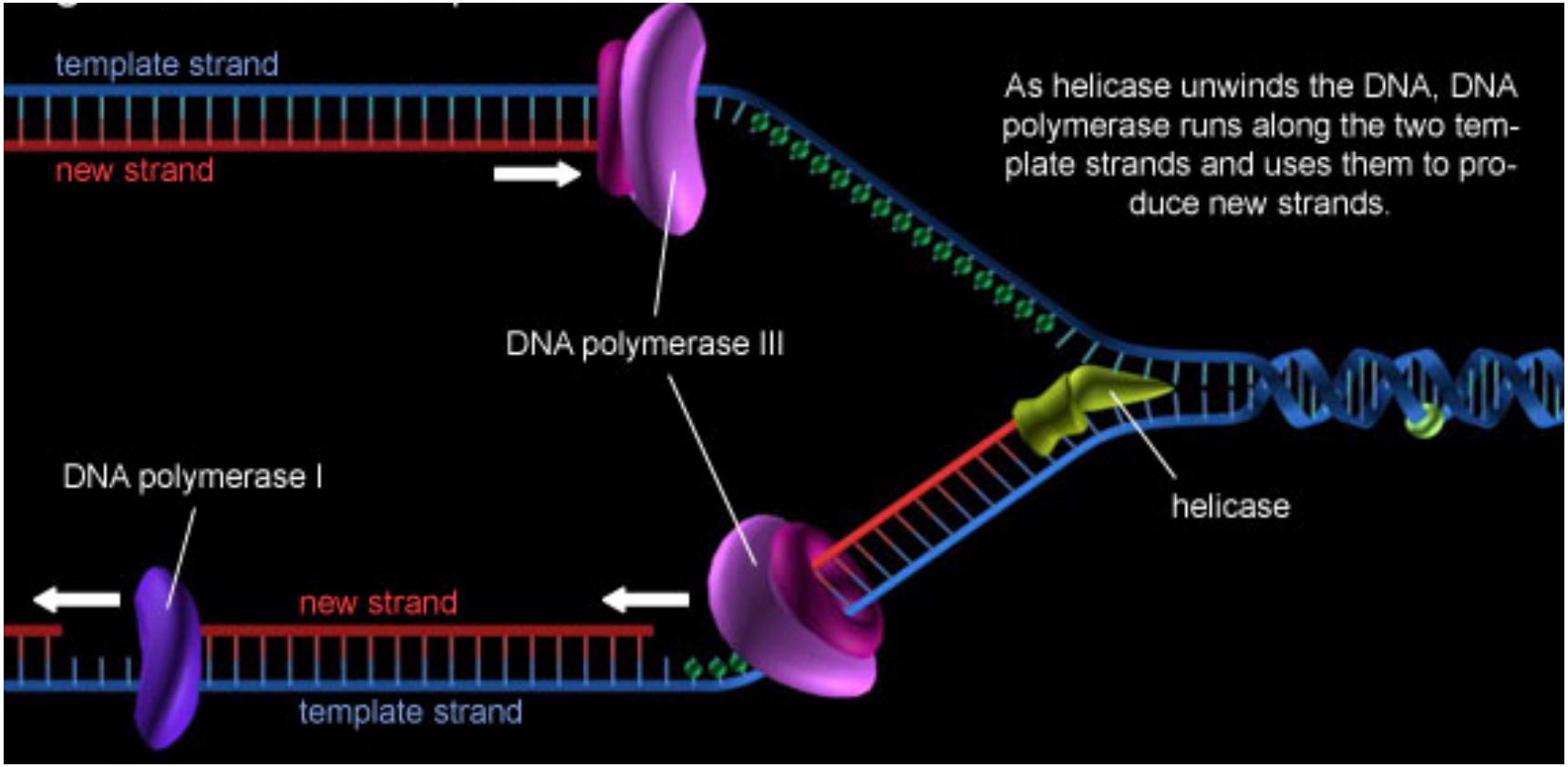
AATTA--CATTA

INDEL genesis

A number of processes are known to generate insertions and deletions in the process of DNA replication:

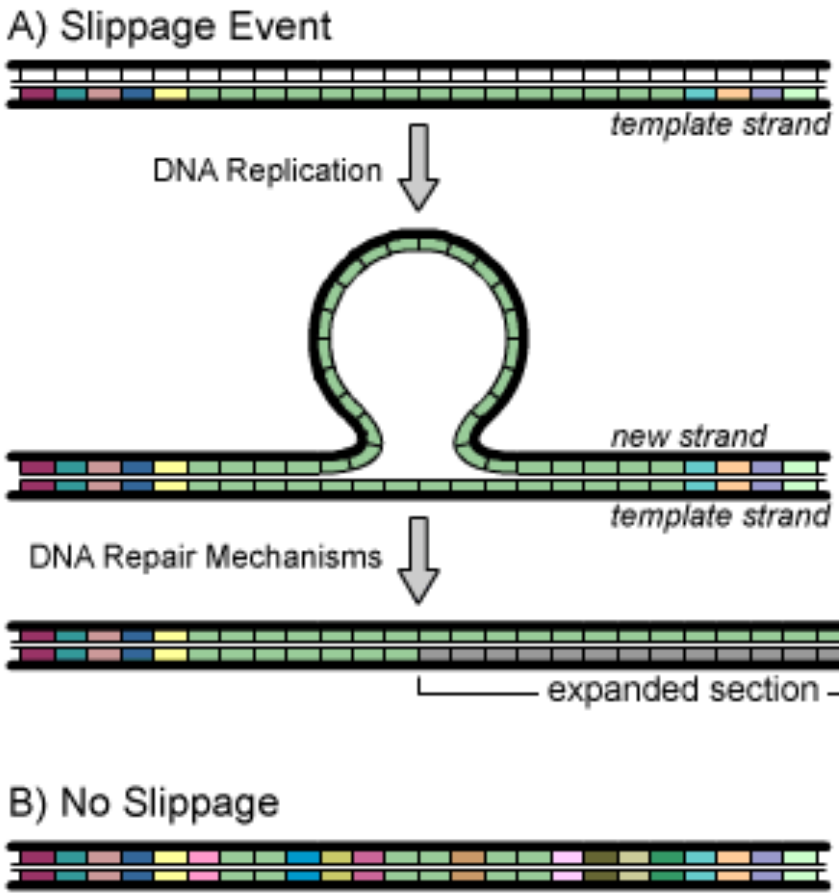
- Replication slippage
- Double-stranded break repair
- Structural variation (e.g. mobile element insertions, CNVs)

DNA replication



http://www.stanford.edu/group/hopes/cgi-bin/wordpress/2011/02/all-about-mutations/

Polymerase *slippage*

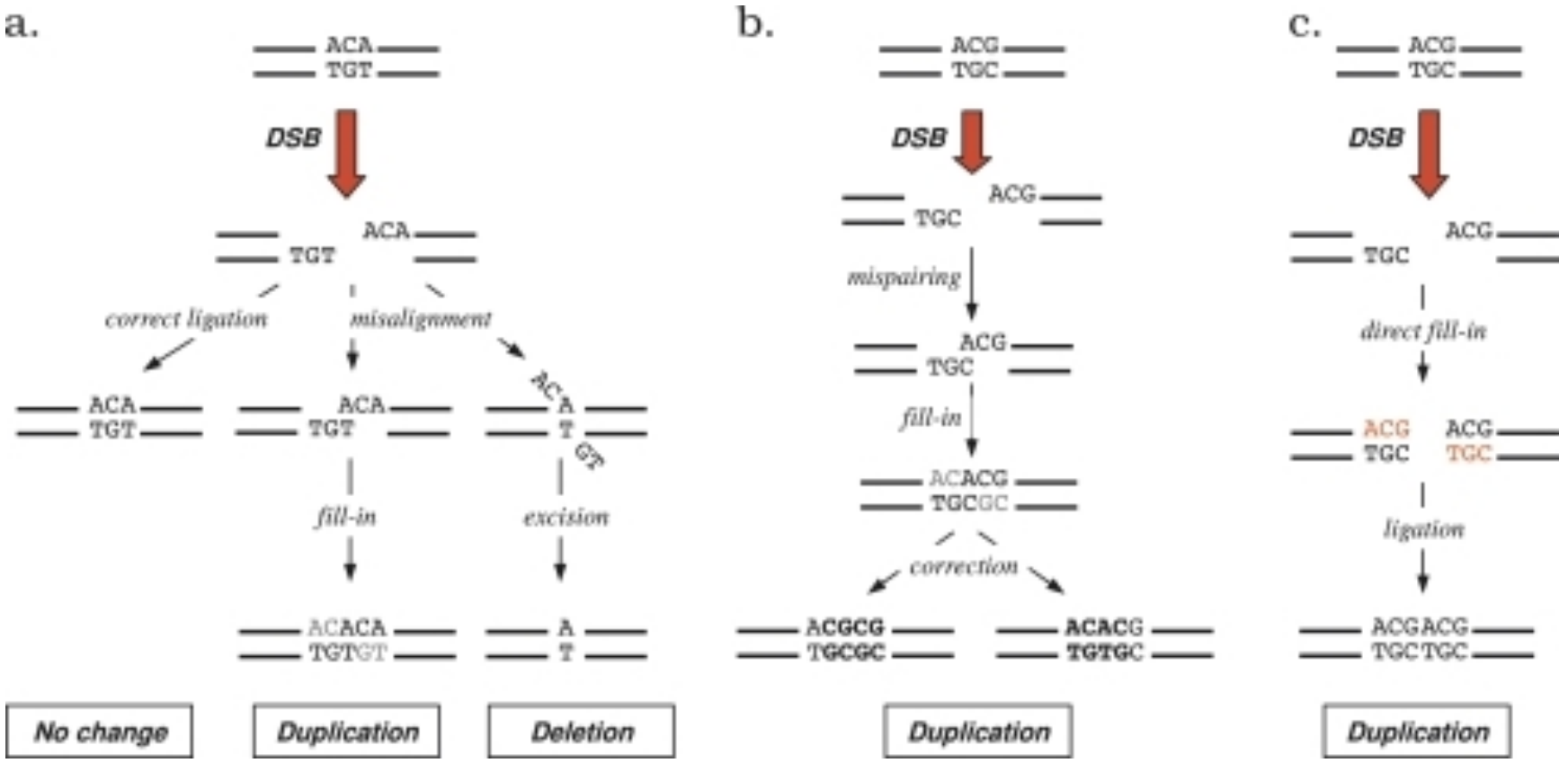


(A) During replication, polymerase slippage and subsequent reattachment may cause a bubble to form in the new strand. Slippage is thought to occur in sections of DNA with repeated patterns of bases (such as CAG), represented here by matching colors. Then, DNA repair mechanisms realign the template strand with the new strand and the bubble is straightened out. The resulting double helix is thus expanded.

(B) Polymerase slippage, as theorized, cannot occur in DNA without repeating patterns of bases.

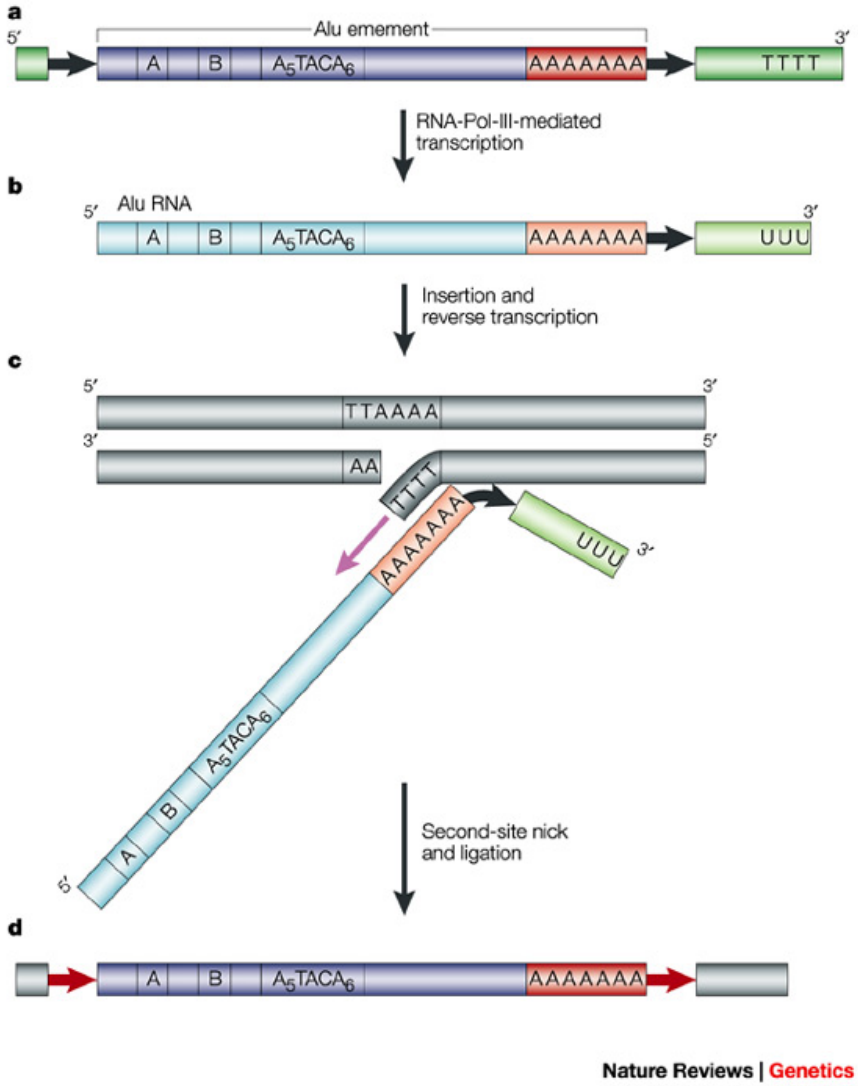
http://www.stanford.edu/group/hopes/cgi-bin/wordpress/2011/02/all-about-mutations/

NHEJ-derived indels



DNA Slippage Occurs at Microsatellite Loci without Minimal Threshold Length in Humans: A Comparative Genomic Approach. Leclercq S, Rivals E, Jarne P - Genome Biol Evol (2010)

Structural variation (SV)



Nature Reviews | Genetics

Transposable elements (in this case, an Alu) are sequences that can copy and paste themselves into genomic DNA, causing insertions.

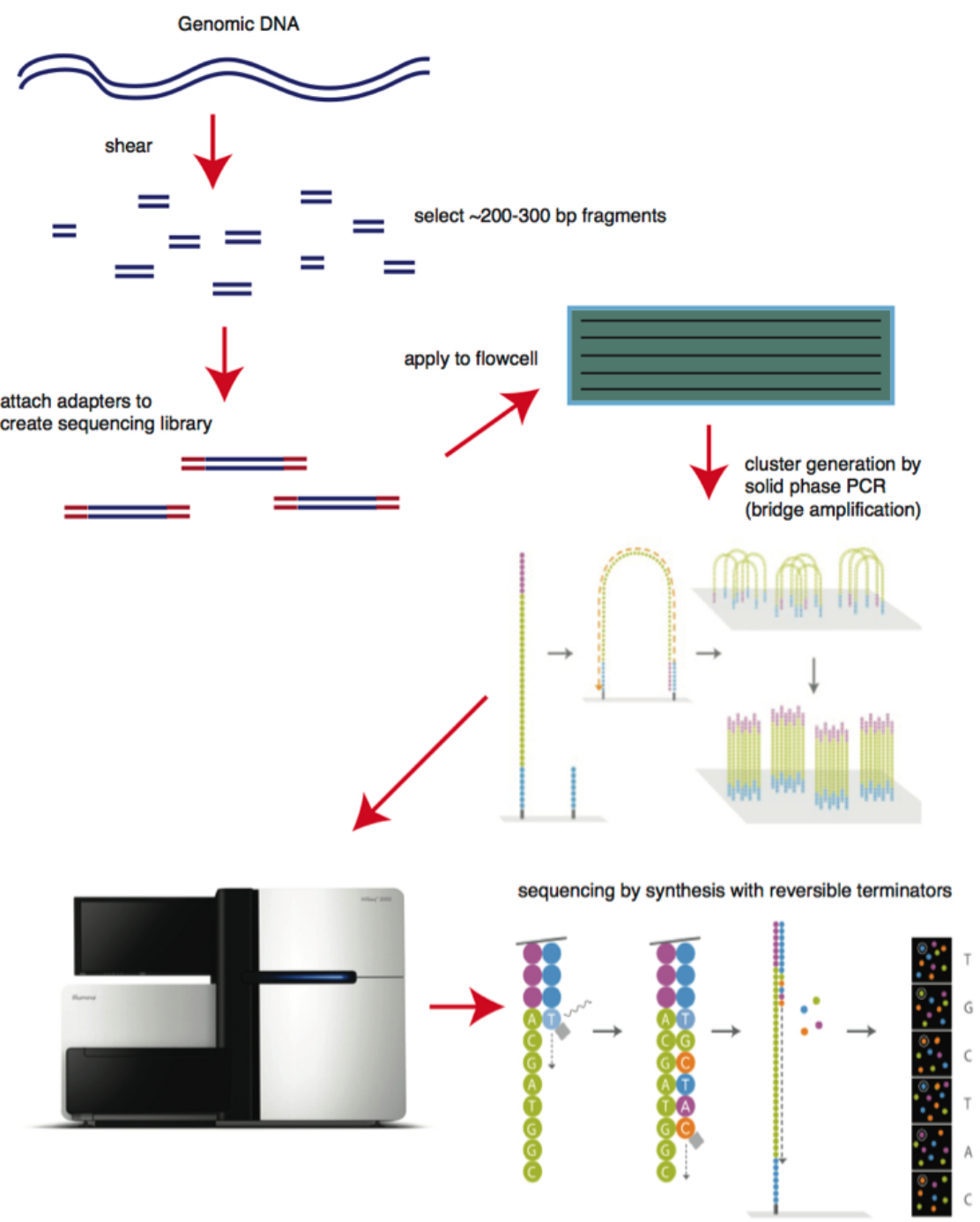
Deletions can also be mediated by these sequences via other processes.

http://www.nature.com/nrg/journal/v3/n5/full/nrg798.html

Overview

1. Genesis of variation (SNPs and indels)
2. **Causes of sequencing error**
3. Sequence alignment
4. Alignment-based variant detection
5. Assembly-based variant detection
6. Haplotype-based variant detection
7. Graph-based methods

Sequencing by synthesis (Illumina)

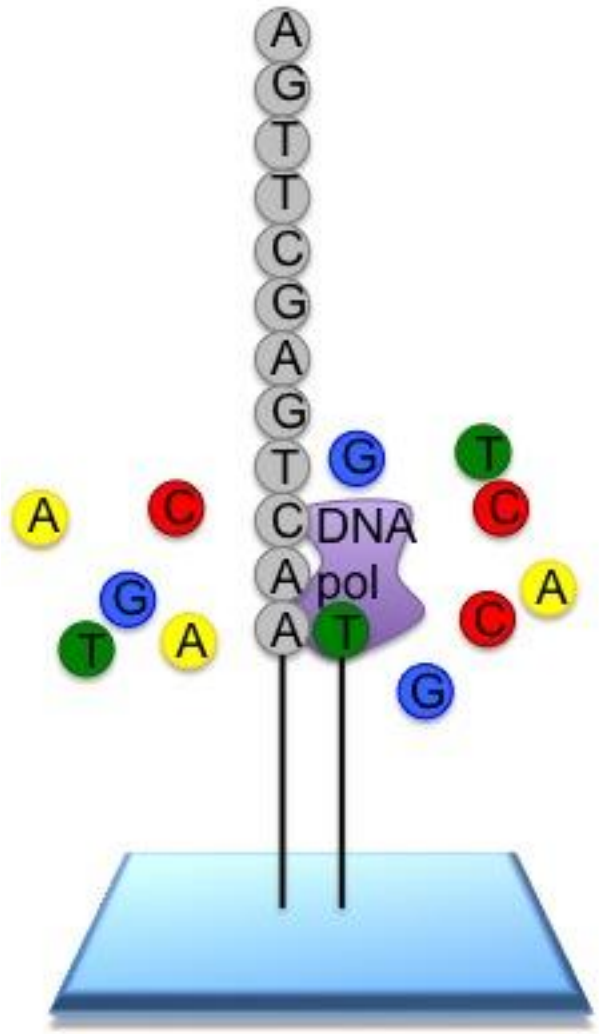


http://bitesizebio.com/13546/sequencing-by-synthesis-explaining-the-illumina-sequencing-technology/

Illumina sequencing process

(1)

Fluorescently labeled dNTPs with reversible terminators are incorporated by polymerase into growing double-stranded DNA attached to the flowcell surface.



http://www.ebi.ac.uk/training/online/course/ebi-next-generation-sequencing-practical-course/what-next-generation-dna-sequencing/illumina-

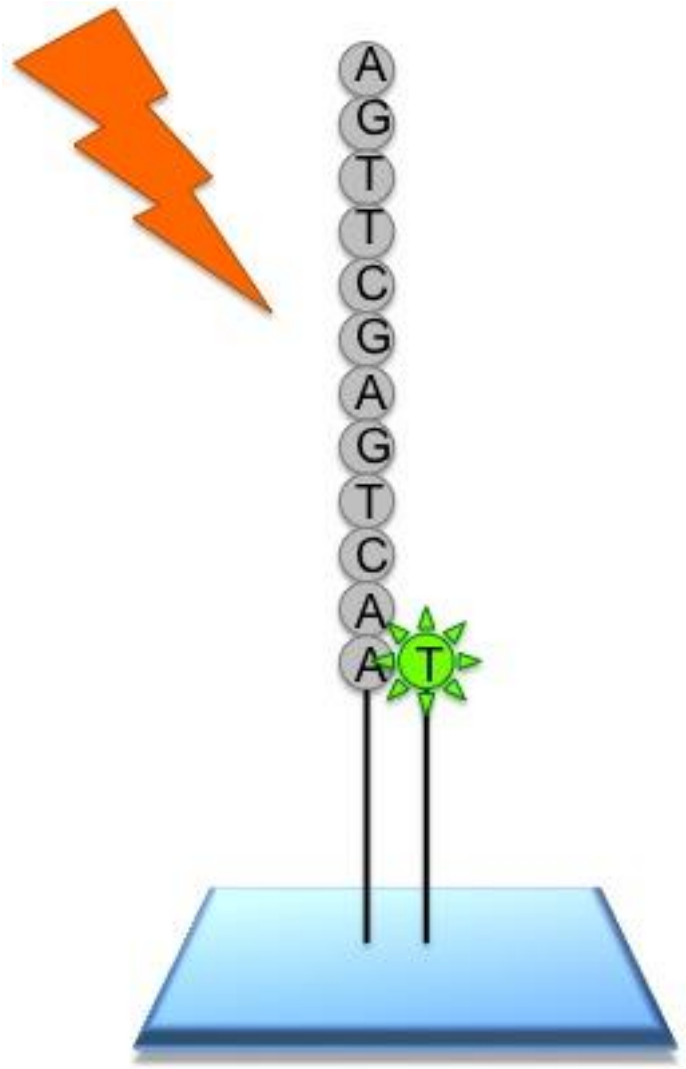
Illumina sequencing process

(2) A round of imaging determines the predominant base in each cluster at the given position.

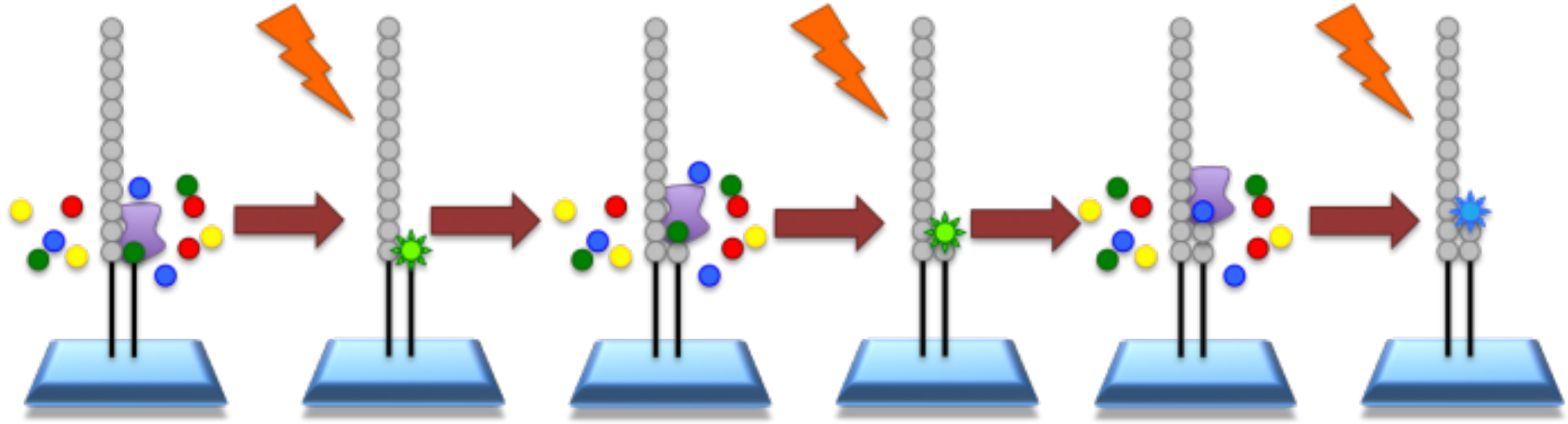
(3) “reverse” terminator by washing, goto **(1)**

(...)

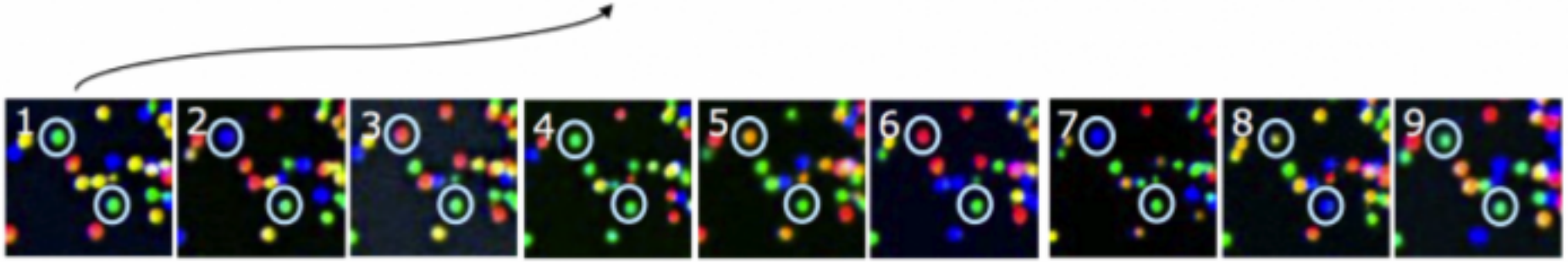
Build up images, process to determine sequence



http://www.ebi.ac.uk/training/online/course/ebi-next-generation-sequencing-practical-course/what-next-generation-dna-sequencing/illumina-



TGCTACGAT...



TTTTTTTGT...

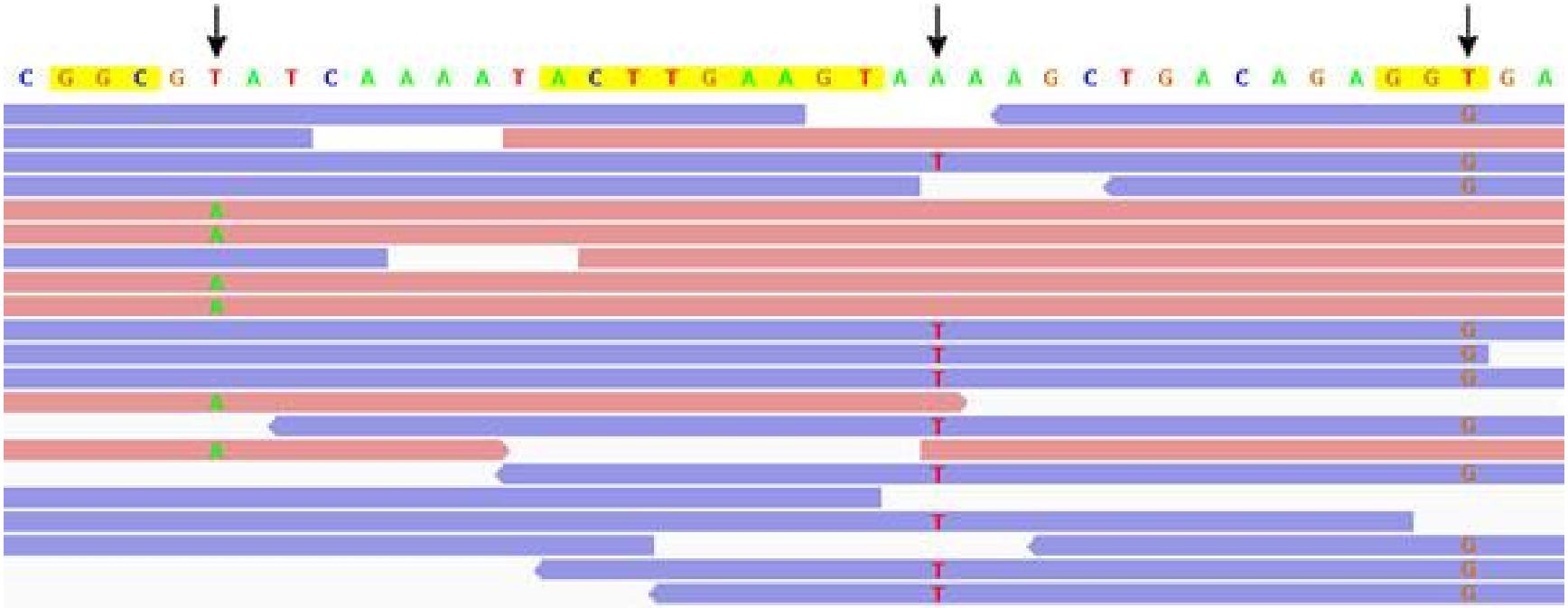
http://www.ebi.ac.uk/training/online/course/ebi-next-generation-sequencing-practical-course/what-next-generation-dna-sequencing/illumina-

What can go wrong?

1. input artifacts, problems with library prep
   a. replication in PCR has no error-correction (→ SNPs)
   b. no quaternary structures (e.g. clamp) to prevent slippage (→ indels)
   c. chimeras…
   d. duplicates (worse if they are errors)
2. Sequencing-by-synthesis
   a. phasing of step
      i. synthesis reaction efficiency is not 100%
      ii. particularly bad in A/T homopolymers
   b. certain *context specific errors*
      i. vary by sequencing protocol, device
      ii. often strand-specific

Context specific errors

Show up as strand-specific errors:



http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3622629/

Context specific errors (motifs)

Rank	Context	FER [%]	RER [%]	ERD [%]
1	ACGGCGGT	26.1	0.5	25.6
2	GTGGCGGT	25.1	0.7	24.4
3	GCGGCGGT	22.9	0.7	22.2
4	GTGGCTGT	22.4	0.6	21.8
5	ATGGCGGT	21.2	1.0	20.3
6	NCGGCGGT	20.0	0.7	19.3
7	GTGGCTTG	20.2	1.2	19.0
8	GNGGCGGT	19.2	0.7	18.5
9	GCGGCTGT	18.8	0.7	18.1
10	ACGGCTGT	18.6	0.8	17.7

← forward and reverse error rates for the ten most-common CSEs on a variety of illumina systems

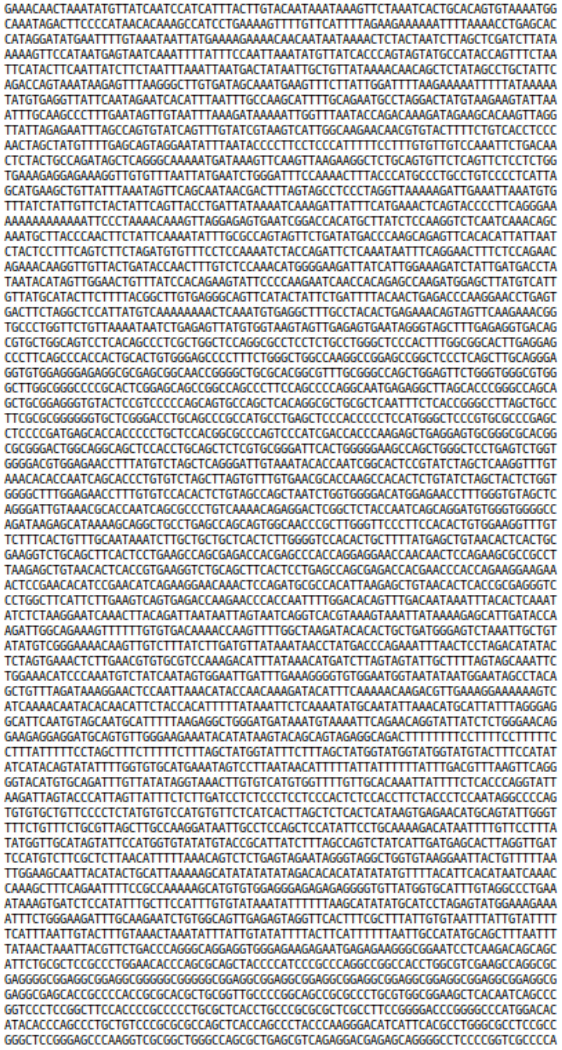
(often GC-rich)

http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3622629/

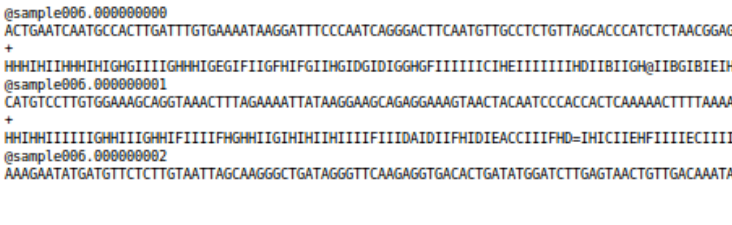
Overview

1. Genesis of variation (SNPs and indels)
2. Causes of sequencing error
3. **Sequence alignment**
4. Alignment-based variant detection
5. Assembly-based variant detection
6. Haplotype-based variant detection
7. Graph-based methods

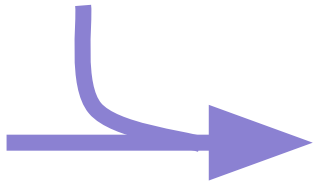
From reads to variants



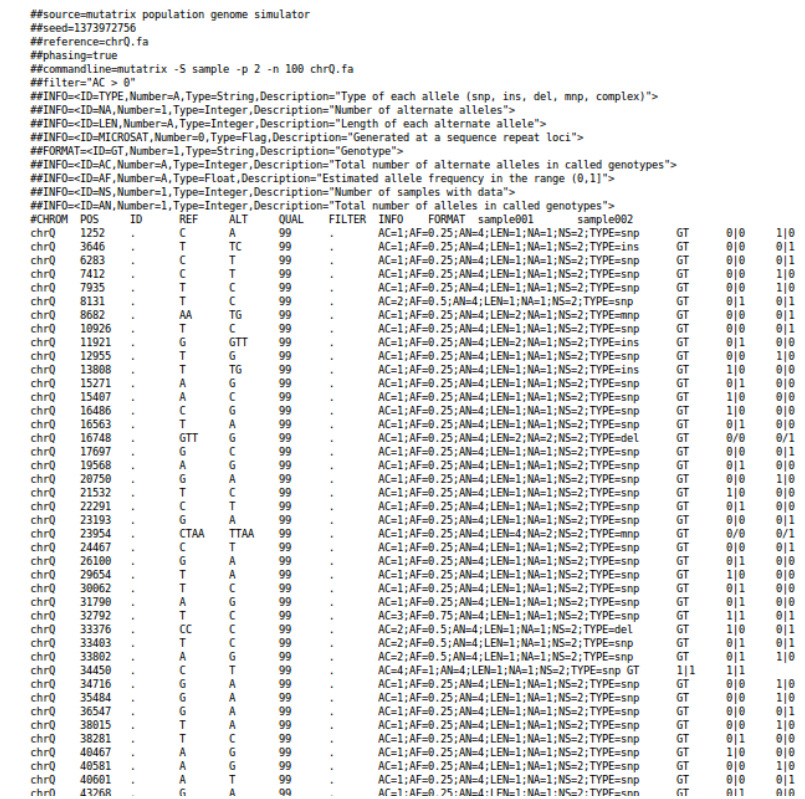
Genome (FASTA)



Reads (FASTQ)



alignment and variant calling



Variation (VCF)

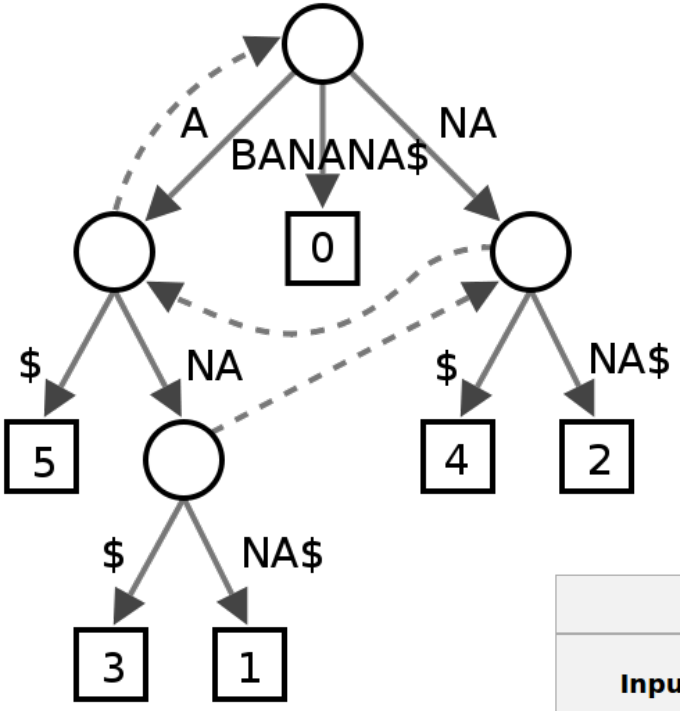
Global alignment

Idea: localize reads against a large reference system. Requires some cleverness.

Main approaches:

1. compressed suffix arrays
2. k-mer based seed+extend

Compressed suffix arrays



← suffix tree ~ suffix array

suffix array ~ Burrows-Wheeler transform

BWTs are very compressible

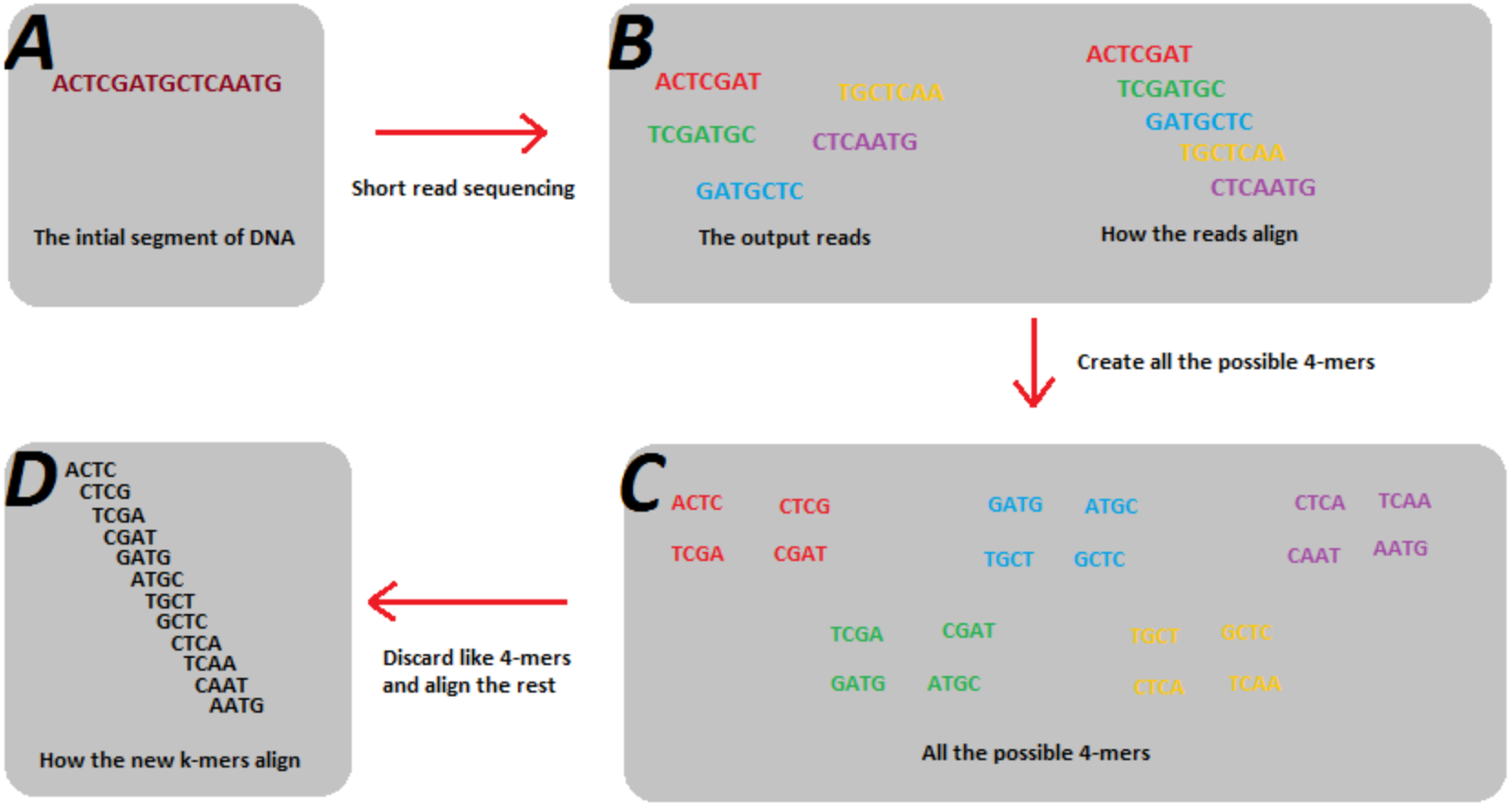
: bwa, bowtie, use FM-index (sampled BWT)

whole-genome index in 3GB of memory!

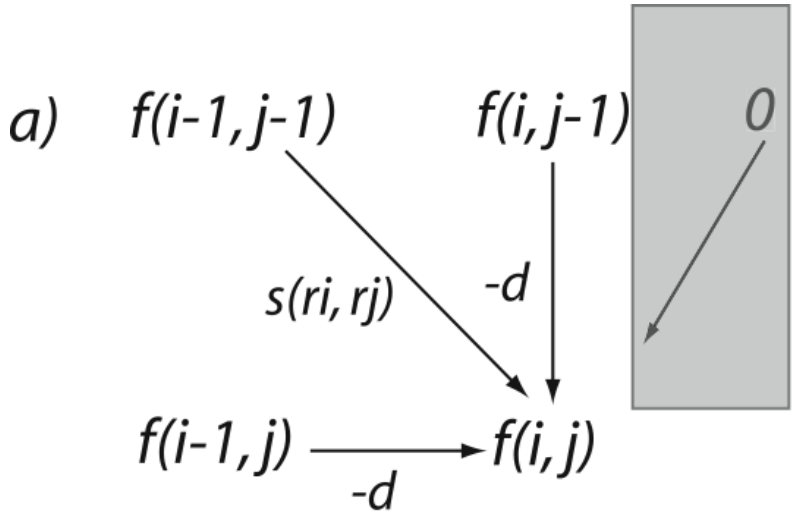
Transformation				
Input	**All Rotations**	**Sorting All Rows into Lex Order**	**Taking Last Column**	**Output Last Column**
^BANANA|	^BANANA| |^BANANA A|^BANAN NA|^BANA ANA|^BAN NANA|^BA ANANA|^B BANANA|^	ANANA|^B ANA|^BAN A|^BANAN BANANA|^ NANA|^BA NA|^BANA ^BANANA| |^BANANA	ANANA|^B ANA|^BAN A|^BANAN BANANA|^ NANA|^BA NA|^BANA ^BANANA| |^BANANA	BNN^AA|A

k-mer based seed+extend

Used in novoalign, MOSAIK.



Local alignment



b)

		A	G	G	T	T	G	C
	A	1	0	-1	-2	-3	-4	-5
	G	0	2	1	0	-1	-2	-3
	G	-1	1	3	2	1	0	-1
	T	-2	0	2	4	3	2	1
	C	-3	-1	1	3	4	3	2

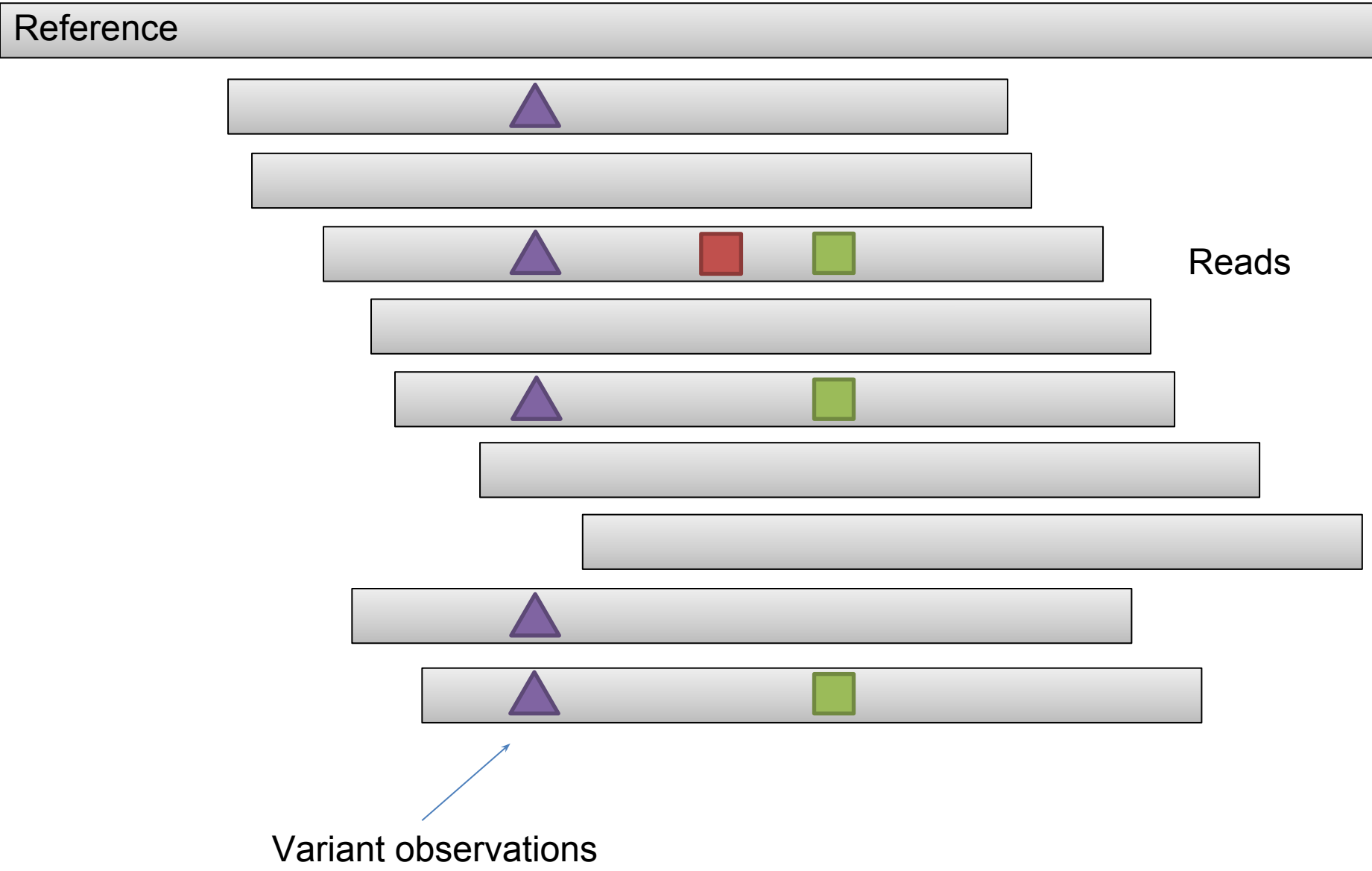
Schematic of Needleman and Wunsch algorithm. Smith and Waterman added the 0. http://www.hiv.lanl.gov/content/sequence/HIV/REVIEWS/2006_7/ABECASIS/abecasis.html

A	G	G	T	T	G	C
A	G	G	T	-	-	C

Overview

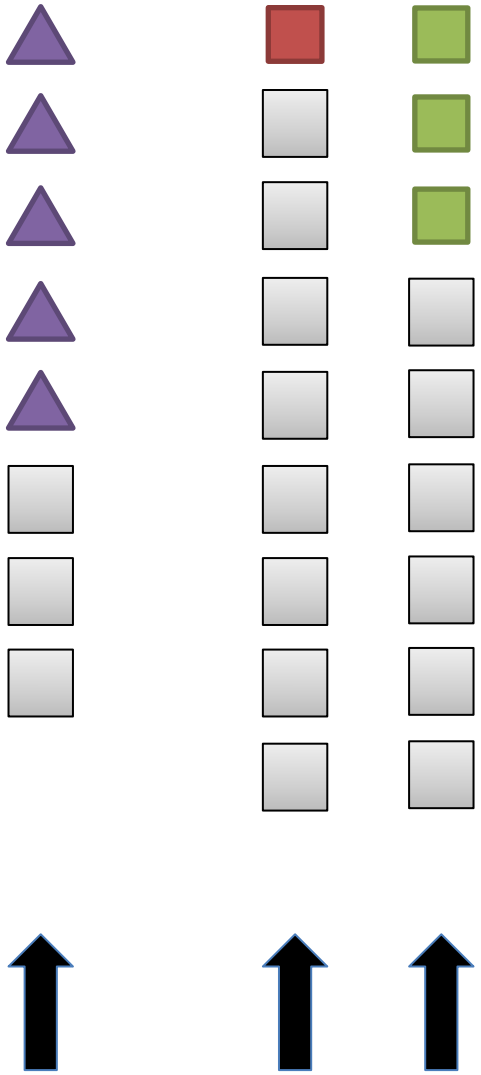
1. Genesis of variation (SNPs and indels)
2. Causes of sequencing error
3. Sequence alignment
4. **Alignment-based variant detection**
5. Assembly-based variant detection
6. Haplotype-based variant detection
7. Graph-based methods

Alignments to candidates



The data exposed to the caller

Reference



Alignment-based variant calling

For each candidate locus, collect information about support for each allele. Use heuristics (e.g. varscan) or a bayesian model (e.g. samtools, GATK, freebayes) to assign maximum-likelihood alleles and genotypes:

(more on this tomorrow...)

$$Pr\left(D|G=\{A_1,A_2\}\right)=\prod_{i=1}^{M}Pr\left(b_i|G=\{A_1,A_2\}\right)$$

$$=\prod_{i=1}^{M}\left(\frac{1}{2}p\left(b_i|A_1\right)+\frac{1}{2}p\left(b_i|A_2\right)\right),$$

$$p\left(b|A\right)=\begin{cases}\frac{e}{3} & b\neq A\\ 1-e & b=A.\end{cases}$$

But wait!

If we work directly from our alignments, we may assign too much significance to the particular way our reads were aligned.

Example: calling INDEL variation

Can we quickly design a process to detect indels from alignment data?

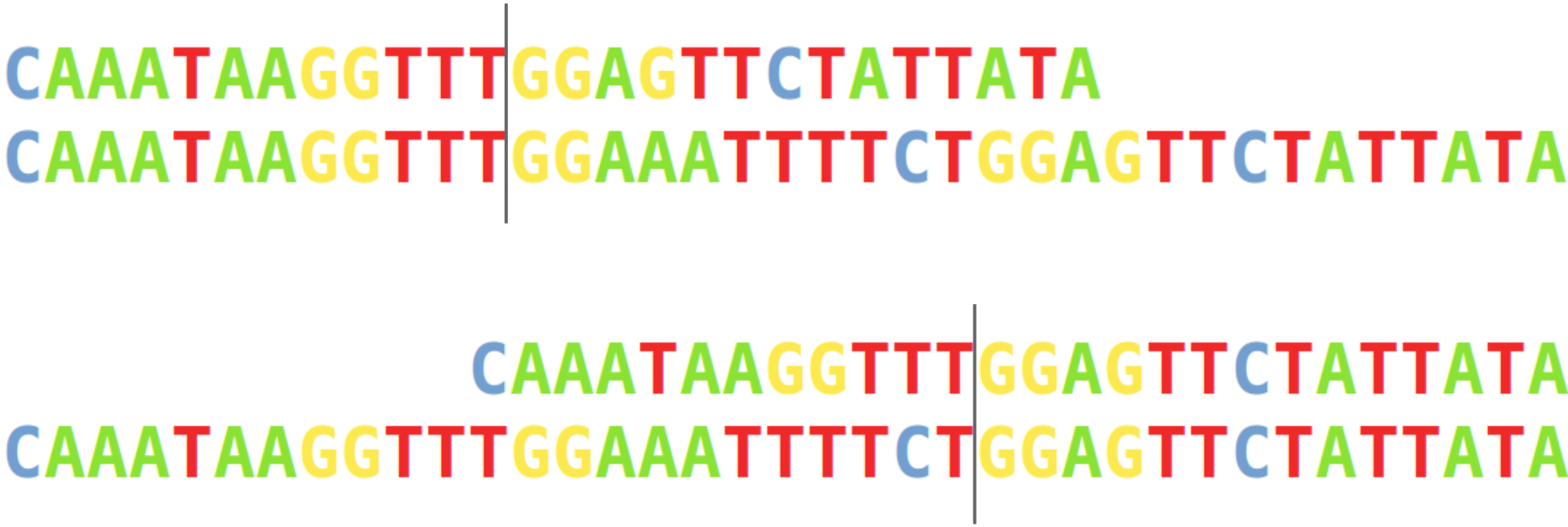
What are the steps you'd do to find the indel between these two sequences?

CAAATAAGGTTTGGAGTTCTATTATA

CAAATAAGGTTTGGAAATTTTCTGGAGTTCTATTATA

Indel finder

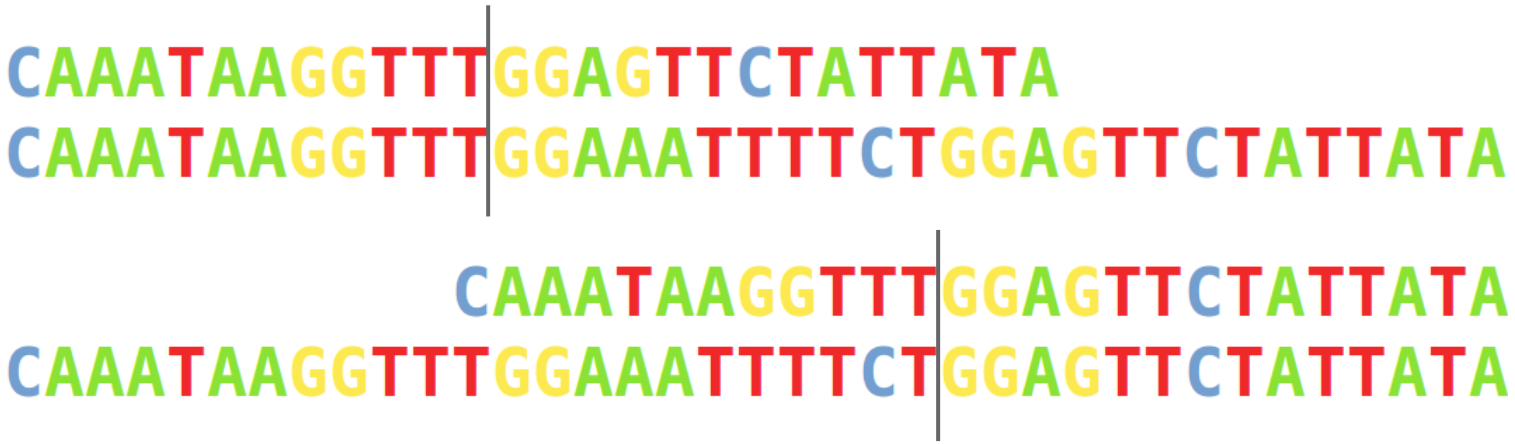
We could start by finding the long matches in both sequences at the start and end:



Indel finder

We can see this more easily like this:

CAAATAAGGTTTGGAGTTCTATTATA
CAAATAAGGTTT*GGAAATTTTCT*GGAGTTCTATTATA



Indel finder

The match structure implies that the sequence that doesn't match was inserted in one sequence, or lost from the other.

```
CAAATAAGGTT----------TGGAGTTCTATTATA
CAAATAAGGTTTGGAAATTTTCTGGAGTTCTATTATA
```

So that's easy enough….

Something more complicated

These sequences are similar to the previous ones, but with different mutations between them.

CAAATAAGGAAATTTTCTGGAGTTCTATTATA
CAAATAAGGTTTGCTATCTAGGTTATTATA

They are still (kinda) homologous but it’s not easy to see.

Pairwise alignment

One solution, assuming a particular set of alignment parameters, has 3 indels and a SNP:

```
CAAATAAGGAAATTT----TCTGGAGTTCTATTATA
CAAATAAGG---TTTGCTATCT--AGGT-TATTATA
```

But if we use a higher gap-open penalty, things look different:

```
CAAATAAGGAAATTT--TCTGGAGTTCTATTATA
CAAATAAGG---TTTGCTATCTAGGT-TATTATA
```

Alignment as interpretation

Different parameterizations can yield different results.

Different results suggest “different” variation.

What kind of problems can this cause? (And how can we mitigate these issues?)

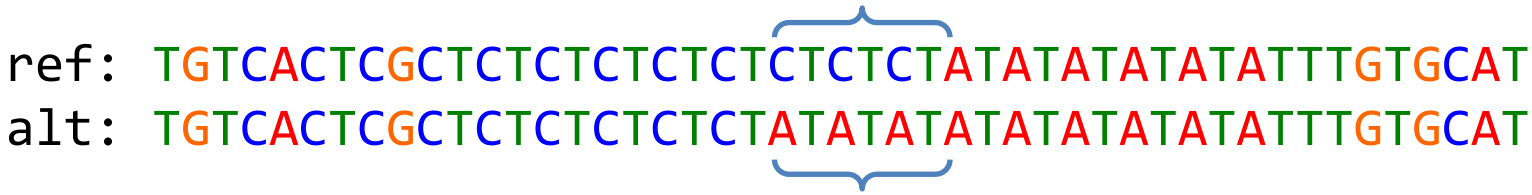
INDELs have multiple representations and require normalization for standard calling

Left alignment allows us to ensure that our representation is consistent across alignments and also variant calls.

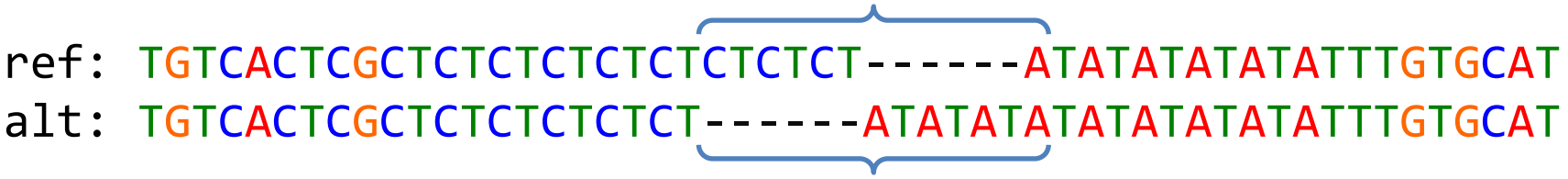
https://www.biostars.org/p/66843/ user sa9

example: 1000G PhaseI low coverage

chr15:81551110, ref:CTCTC alt:ATATA



Interpreted as 3 SNPs



Interpreted as microsatellite expansion/contraction

example: 1000G Phasel low coverage
chr20:708257, ref:AGC alt:CGA

Overview

1. Genesis of variation (SNPs and indels)
2. Causes of sequencing error
3. Sequence alignment
4. Alignment-based variant detection
5. **Assembly-based variant detection**
6. Haplotype-based variant detection
7. Graph-based methods

Problem: inconsistent variant representation makes alignment-based variant calling difficult

If alleles are represented in multiple ways, then to detect them correctly with a single-position based approach we need:

1. An awesome normalization method
2. Perfectly consistent filtering (so we represent our entire context correctly in the calls)
3. Highly-accurate reads

Solution: assembly and haplotype-driven detection

We can shift our focus from the specific interpretation in the alignments:

- this is a SNP
- whereas this is a series of indels

… and instead focus on the underlying sequences.

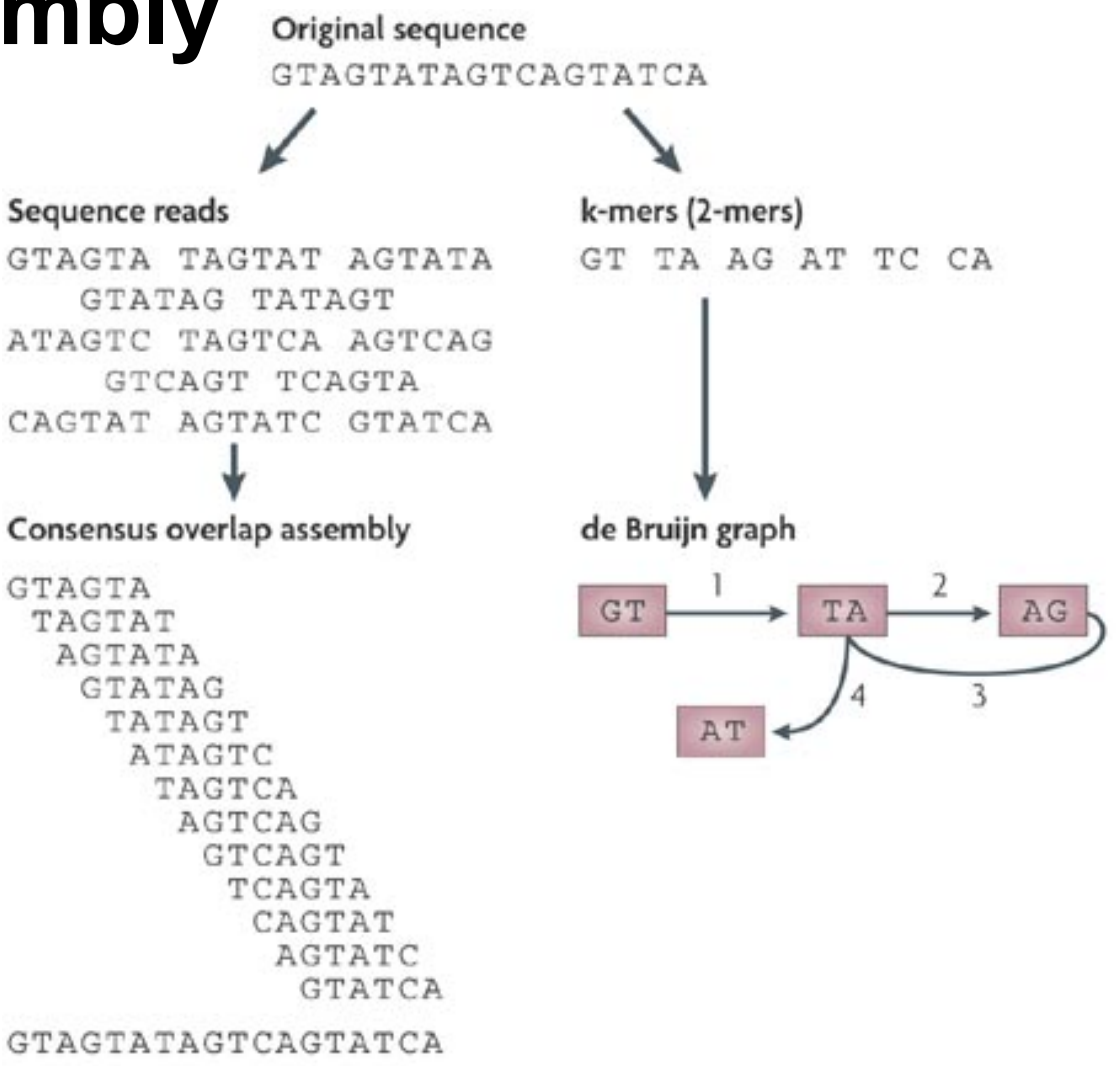
Basically, we use the alignments to localize reads, then process them again with assembly approaches to determine candidate alleles.

Variant detection by assembly

Multiple methods have been developed by members of the 1000G analysis group:

- Global joint assembly
  - cortex
  - SGA (localized to 5 megabase chunks)
- Local assembly
  - Platypus (+cortex)
  - GATK HaplotypeCaller
- k-mer based detection
  - FreeBayes (anchored reference-free windows)

de Bruijn Assembly



Nature Reviews | Microbiology

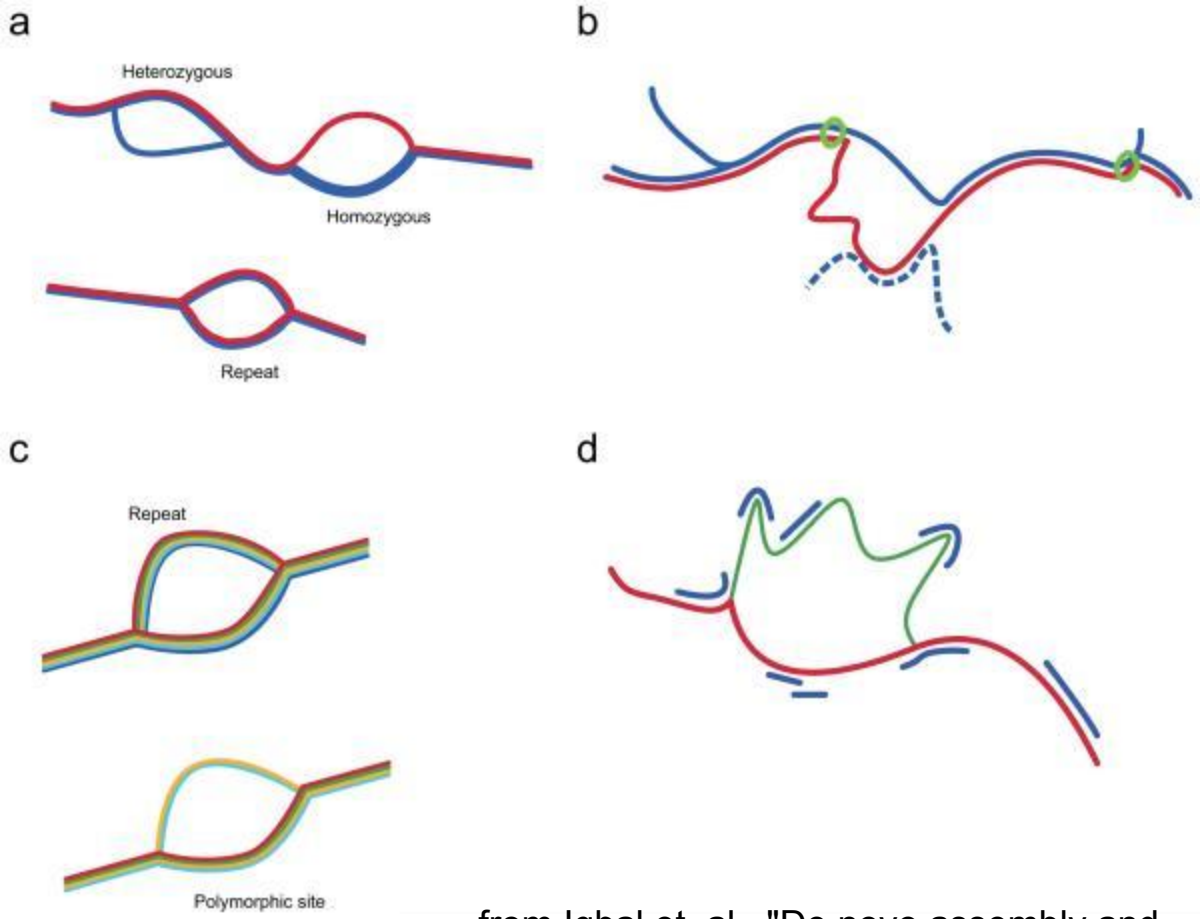
http://www.nature.com/nrmicro/journal/v7/n4/full/nrmicro2088.html

Using colored graphs (Cortex)

Variants can be called using bubbles in deBruijn graphs.

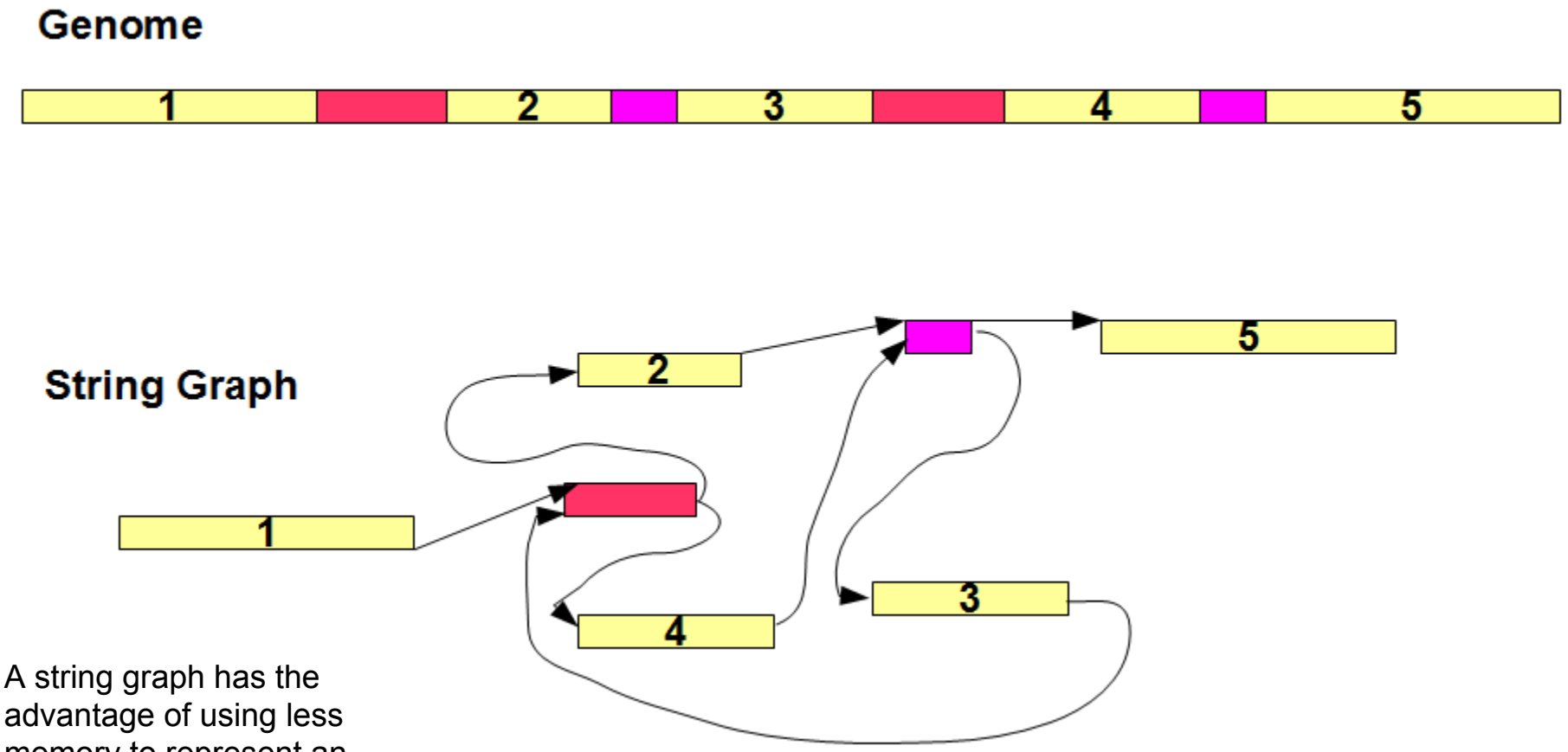
Method is completely reference-free, except for reporting of variants. The reference is threaded through the colored graph.

Many samples can be called at the same time.



from Iqbal et. al., "De novo assembly and genotyping of variants using colored de Bruijn graphs." (2012)

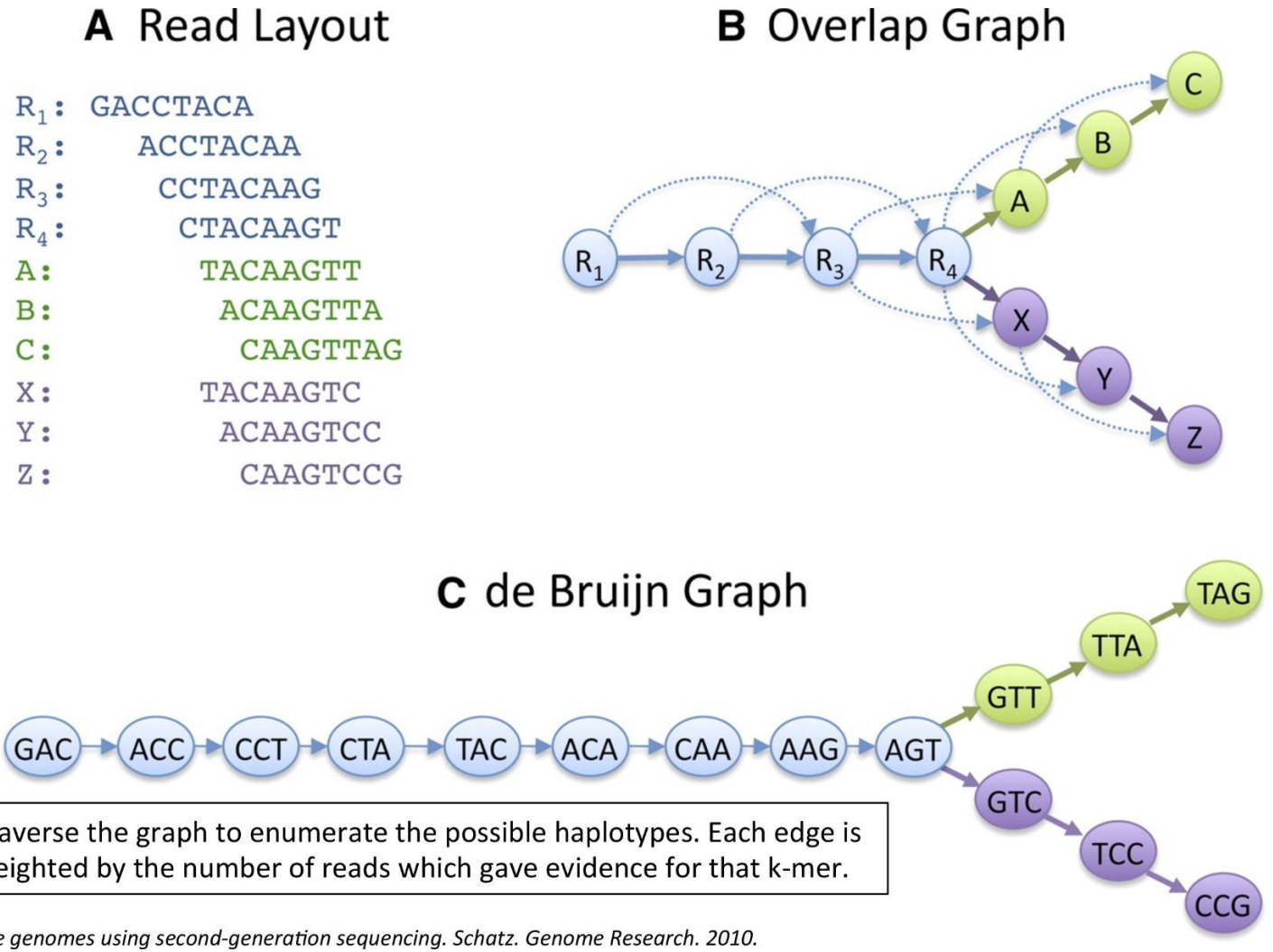
String graphs (SGA)



A string graph has the advantage of using less memory to represent an assembly than a de Bruijn graph. In the 1000G, SGA is run on alignments localized to ~5mb chunks.

http://www.homolog.us/blogs/2012/02/13/string-graph-of-a-genome/

Discovering alleles using graphs (GATK HaplotypeCaller)

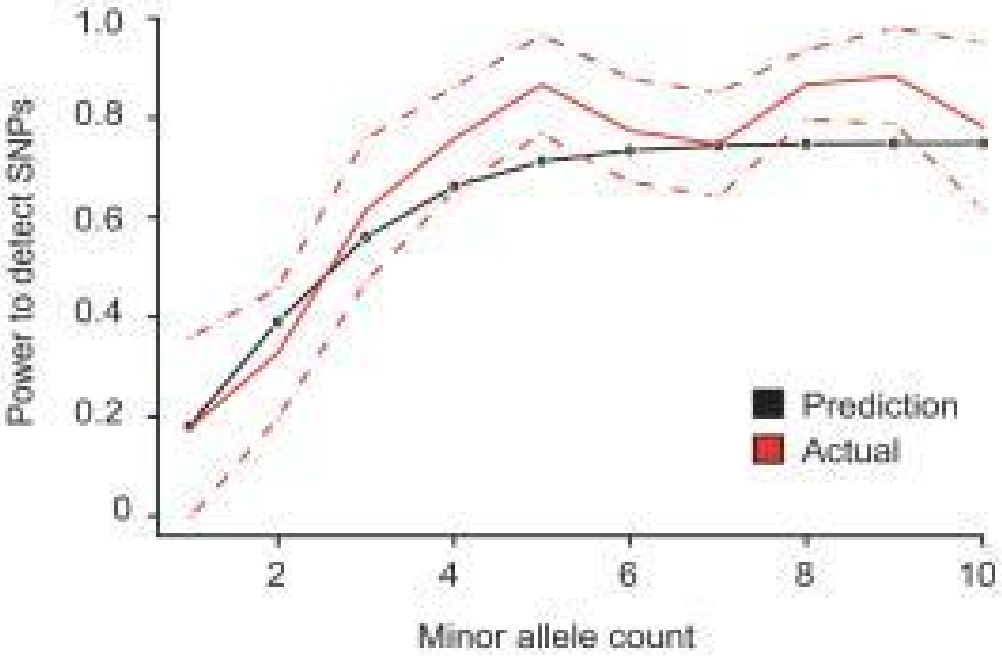


Traverse the graph to enumerate the possible haplotypes. Each edge is weighted by the number of reads which gave evidence for that k-mer.

Assembly of large genomes using second-generation sequencing. Schatz. Genome Research. 2010.

Why don't we just assemble?

Assembly-based calls tend to have high specificity, but sensitivity suffers.



The requirement of exact kmer matches means that errors disrupt coverage of alleles.

Existing assembly methods don't just detect point mutations--- they detect haplotypes.

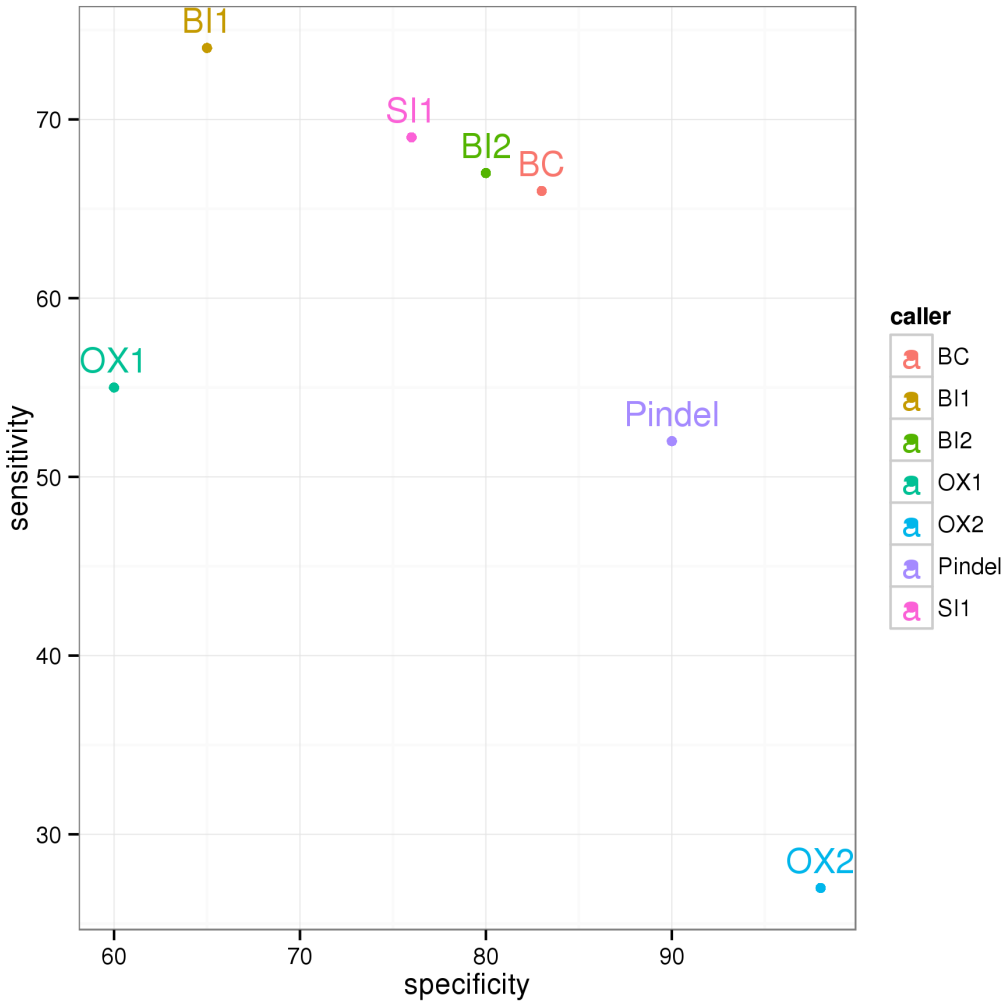
from Iqbal et. al., "De novo assembly and genotyping of variants using colored de Bruijn graphs." (2012)

Indel validation, 191 AFR samples

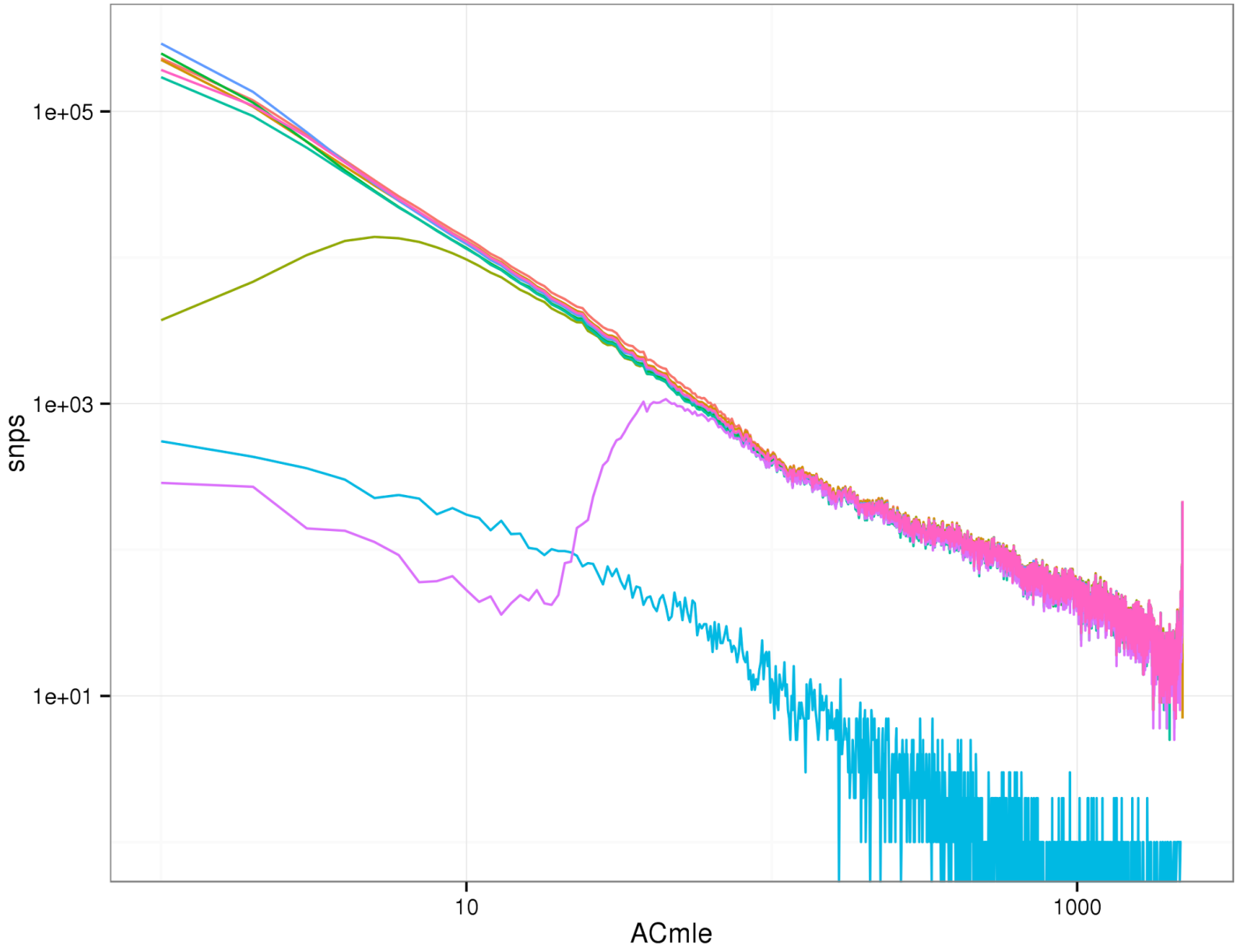
High-depth miSeq sequencing-based validation on 4 samples.

Local assembly methods (BI2, BC, SI1)* have higher specificity than baseline mapping-based calls (BI1), but lower sensitivity. Global assembly (OX2) yielded very low error, but also low sensitivity.

*The local assembly-based method Platypus (OX1) had a genotyping bug which caused poor performance.



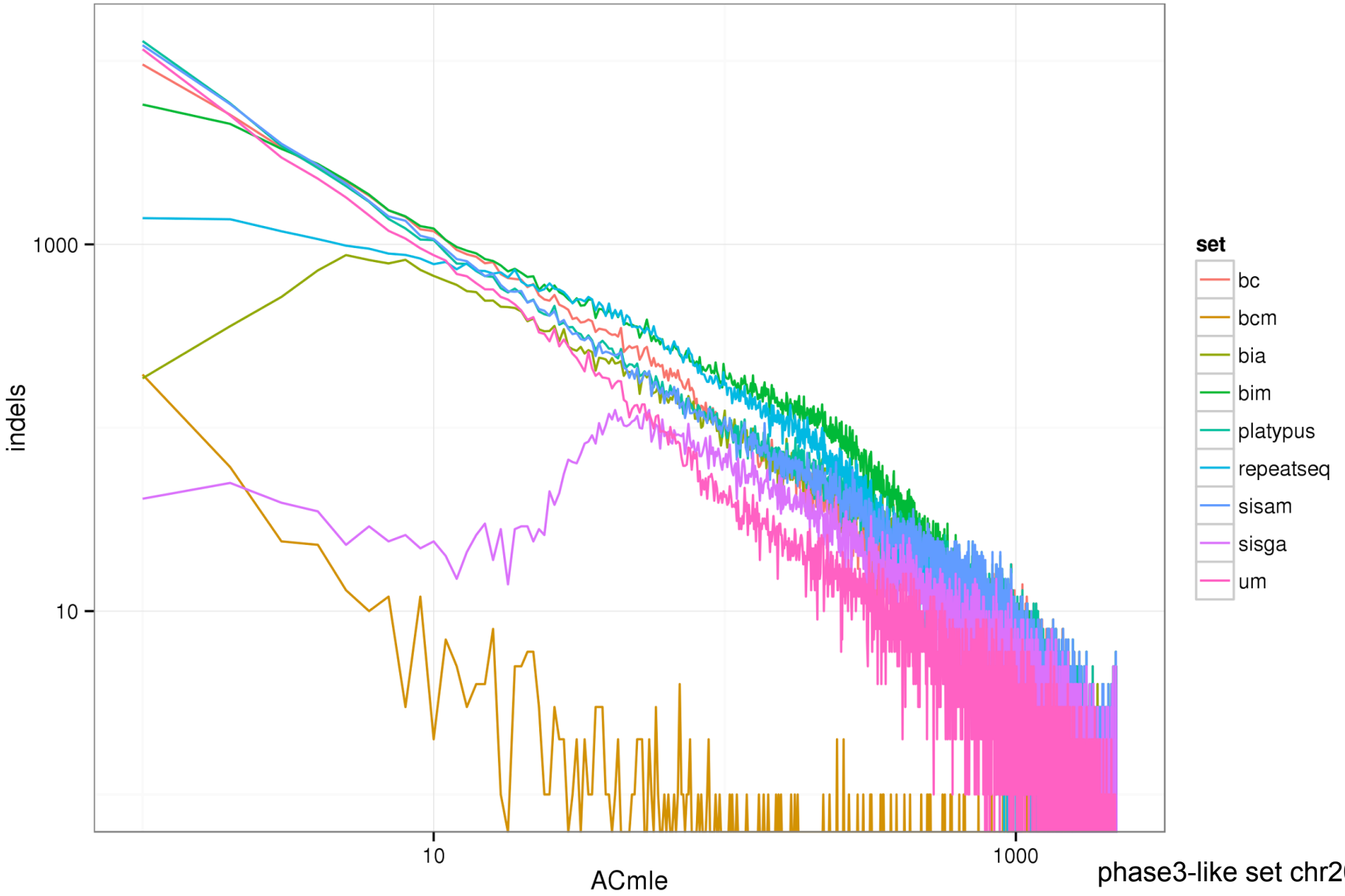
Site-frequency spectrum, SNPs



phase3-like set
chr20

*RepeatSeq is a microsatellite caller

Site-frequency spectrum, indels



phase3-like set chr20

Overview

1. Genesis of variation (SNPs and indels)
2. Causes of sequencing error
3. Sequence alignment
4. Alignment-based variant detection
5. Assembly-based variant detection
6. **Haplotype-based variant detection**
7. Graph-based methods

Finding haplotype polymorphisms

Two reads

AGAACCCAGTGCTCTTTCTGCT

AGAACCCAGTGGTCTTTCTGCT

a SNP

AGAACCCAGTG C/G TCTTTCTGCT

Their alignment

Another read showing a SNP on the same haplotype as the first

AGAACCCAGTGCTCTATCTGCT

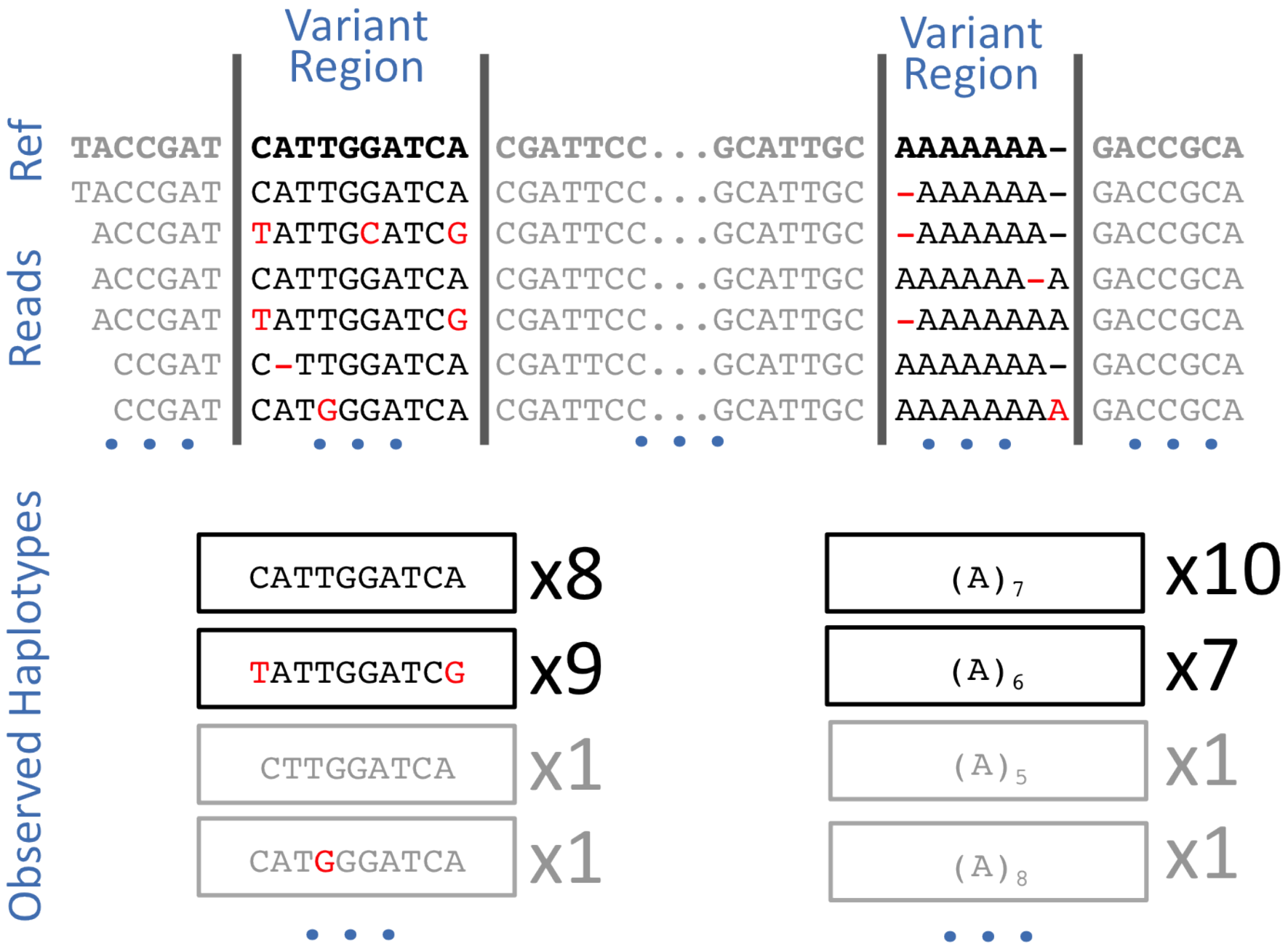
AGAACCCAGTG CTCTA/GTCTT TCTGCT

A variant locus implied by alignments

Direct detection of haplotypes

Direct detection of haplotypes from reads resolves differentially-represented alleles (as the sequence is compared, not the alignment).

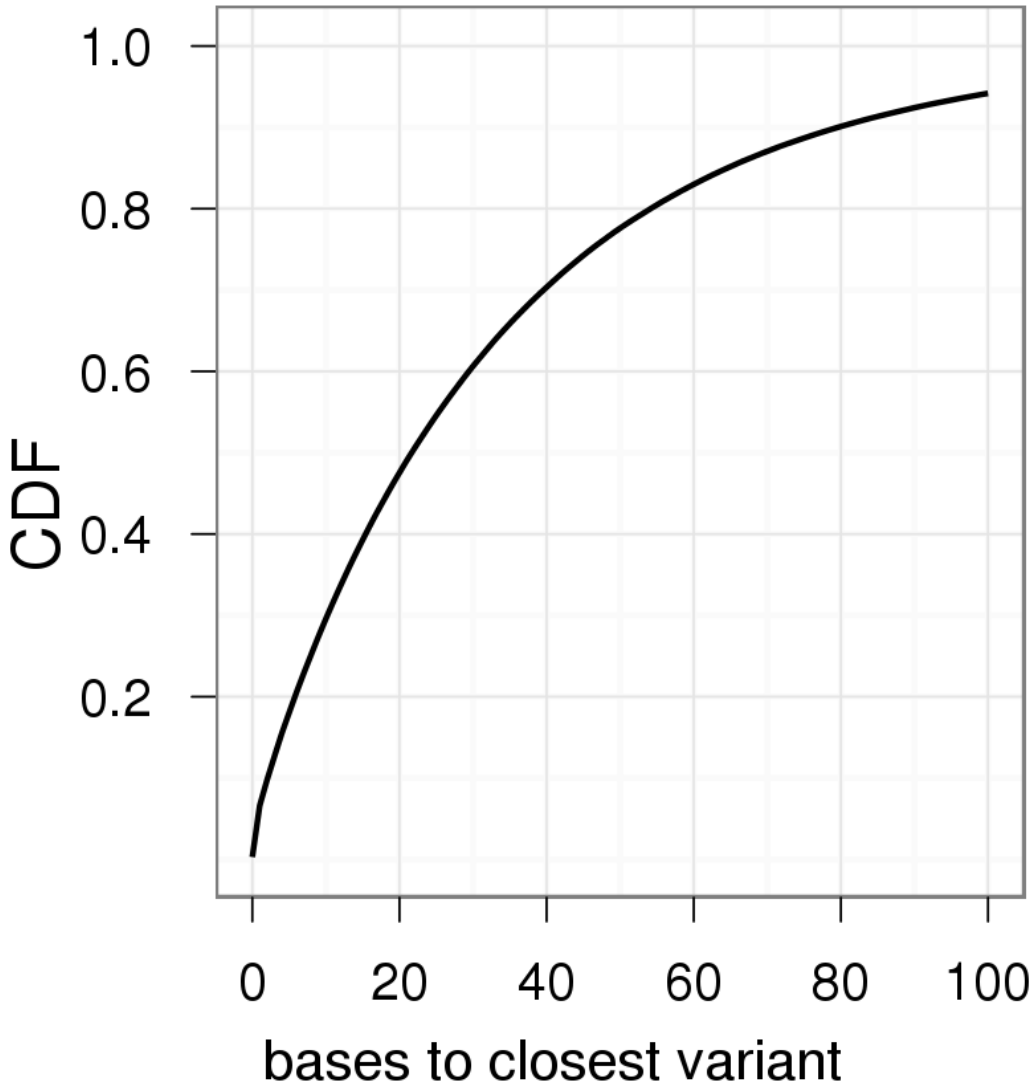
Allele detection is still alignment-driven.


Variant Region
Variant Region
Ref
Reads
TACCGAT CATTGGATCA CGATTCC...GCATTGC AAAAAAA- GACCGCA
TACCGAT CATTGGATCA CGATTCC...GCATTGC -AAAAAA- GACCGCA
ACCGAT TATTGCATCG CGATTCC...GCATTGC -AAAAAA- GACCGCA
ACCGAT CATTGGATCA CGATTCC...GCATTGC AAAAAA-A GACCGCA
ACCGAT TATTGGATCG CGATTCC...GCATTGC -AAAAAAA GACCGCA
CCGAT C-TTGGATCA CGATTCC...GCATTGC AAAAAAA- GACCGCA
CCGAT CATGGGATCA CGATTCC...GCATTGC AAAAAAAA GACCGCA
Observed Haplotypes
CATTGGATCA x8
TATTGGATCG x9
CTTGGATCA x1
CATGGGATCA x1
(A)7 x10
(A)6 x7
(A)5 x1
(A)8 x1

Why haplotypes?

- Variants cluster.
- This has functional significance.
- Observing haplotypes lets us be more certain of the local structure of the genome.
- We can improve the detection process itself by using haplotypes rather than point mutations.
- We get the sensitivity of alignment-based approaches with the specificity of assembly-based ones.

Sequence variants cluster



In ~1000 individuals, ½ of variants are within ~22bp of another variant.

Variance to mean ratio (VMR) = 1.4.

The functional effect of variants depends on other nearby variants on the same haplotype

```
reference:  AGG GAG CTG
            Arg Glu Leu

apparent:   AGG TAG CTG
            Arg Ter ---

actual:     AGG TTG CTG
            Arg Leu Leu
```

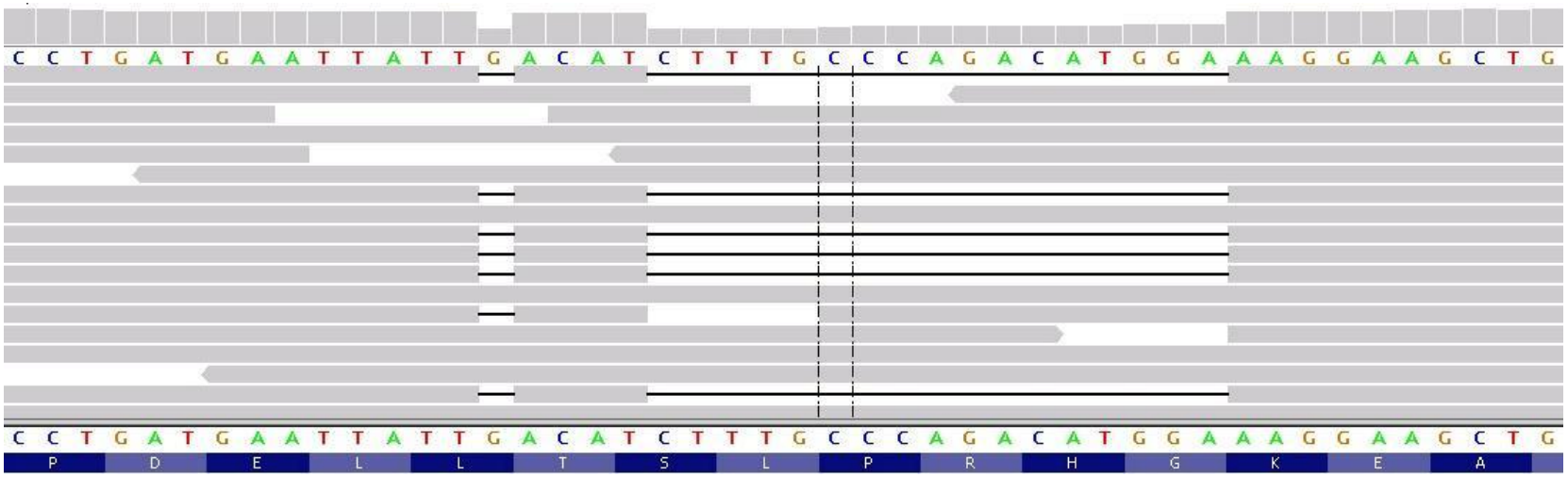
OTOF gene – mutations cause profound recessive deafness

Apparent nonsense variant, one YRI homozygote

Actually a block substitution that results in a missense substitution

(Daniel MacArthur)

Importance of haplotype effects: frame-restoring indels



(in NA12878)

- Two apparent frameshift deletions in the *CASP8AP2* gene (one 17 bp, one 1 bp) on the same haplotype
- Overall effect is in-frame deletion of six amino acids

(Daniel MacArthur)

Frame-restoring indels in 1000 Genomes Phase I exomes

chr6:117113761, GPRC6A **(~10% AF in 1000G)**

```
ref: ATTGTAATTCTCA--TA--TT--TGCCTTTGAAAGC
alt: ATTGTAATTCTCAGGTAATTTCCTGCCTTTGAAAGC
```

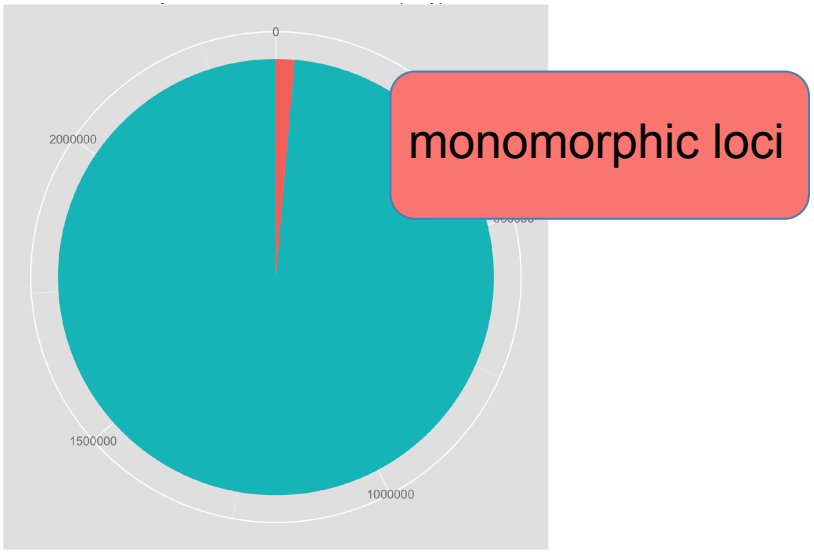
chr6:32551935, HLA-DRB1 **(~11% AF in 1000G)**

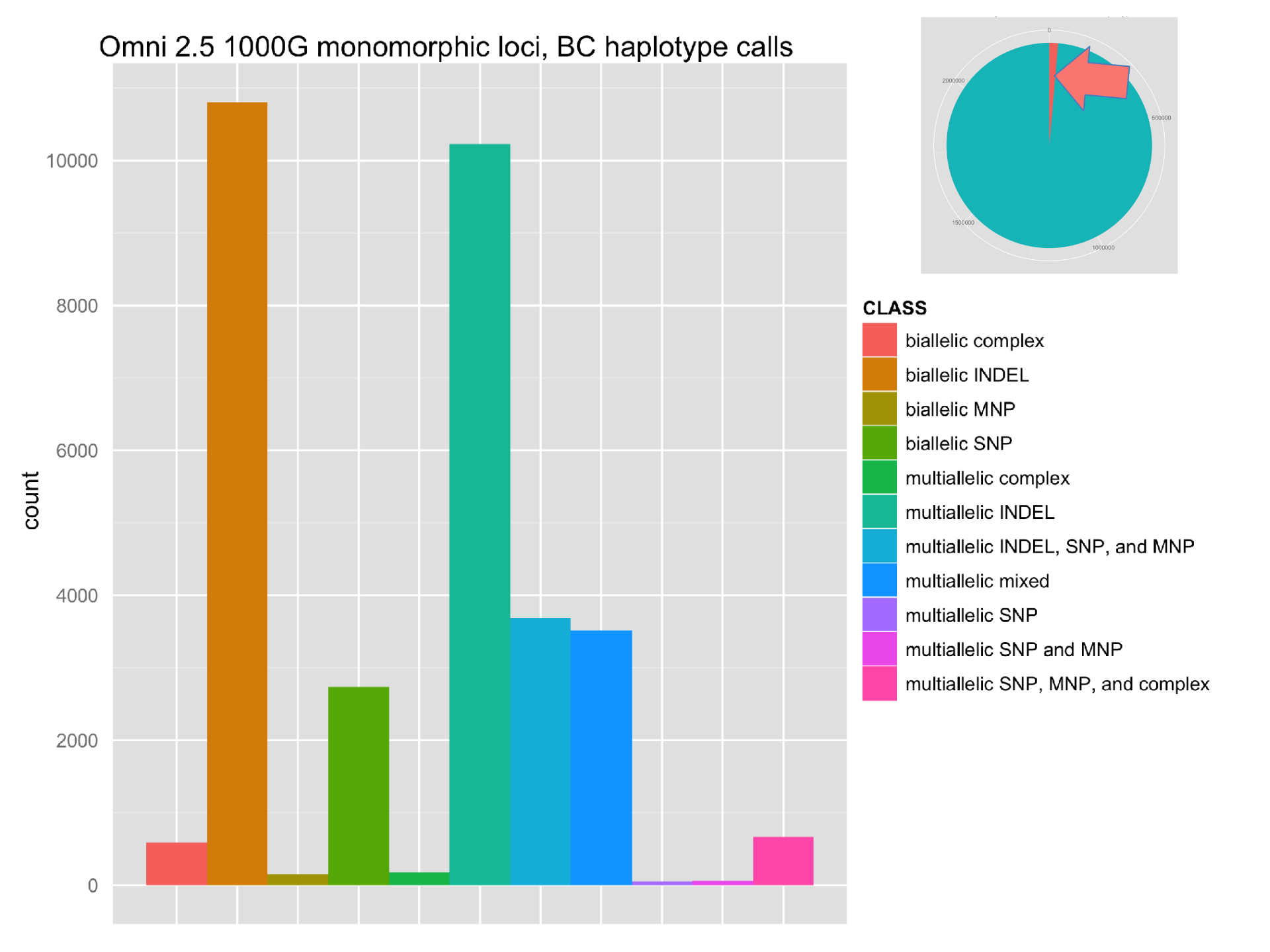
```
 ref: CCACCGCGGCCCGCGCCTG-C-TCCAGGATGTCC
 Alt: CCACCGCGG--CGCGCCTGTCTTCCAGGAGGTCC
```

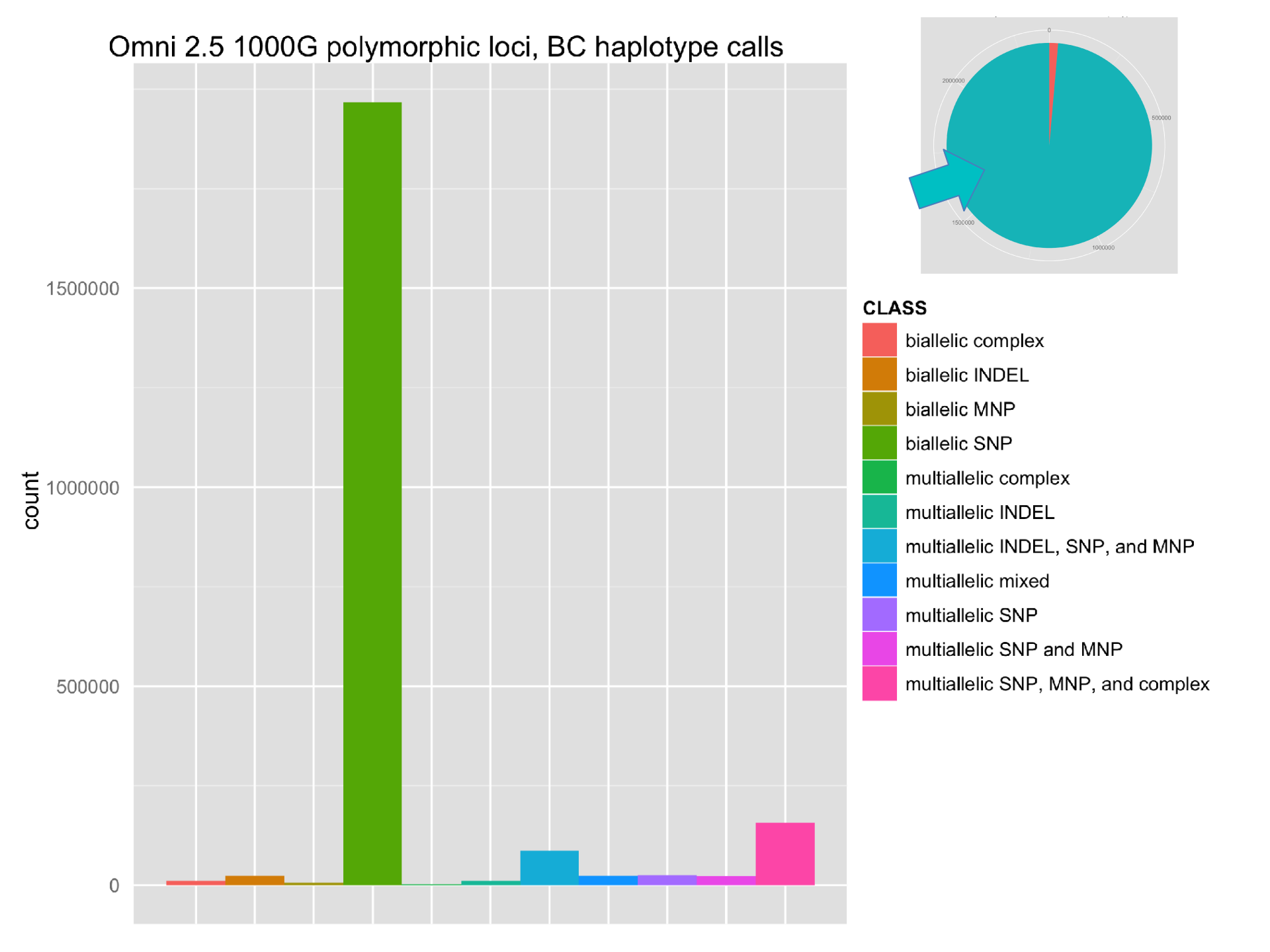
Impact on genotyping chip design

- Biallelic SNPs detected during the 1000 Genomes Pilot project were used to design a genotyping microarray (Omni 2.5).

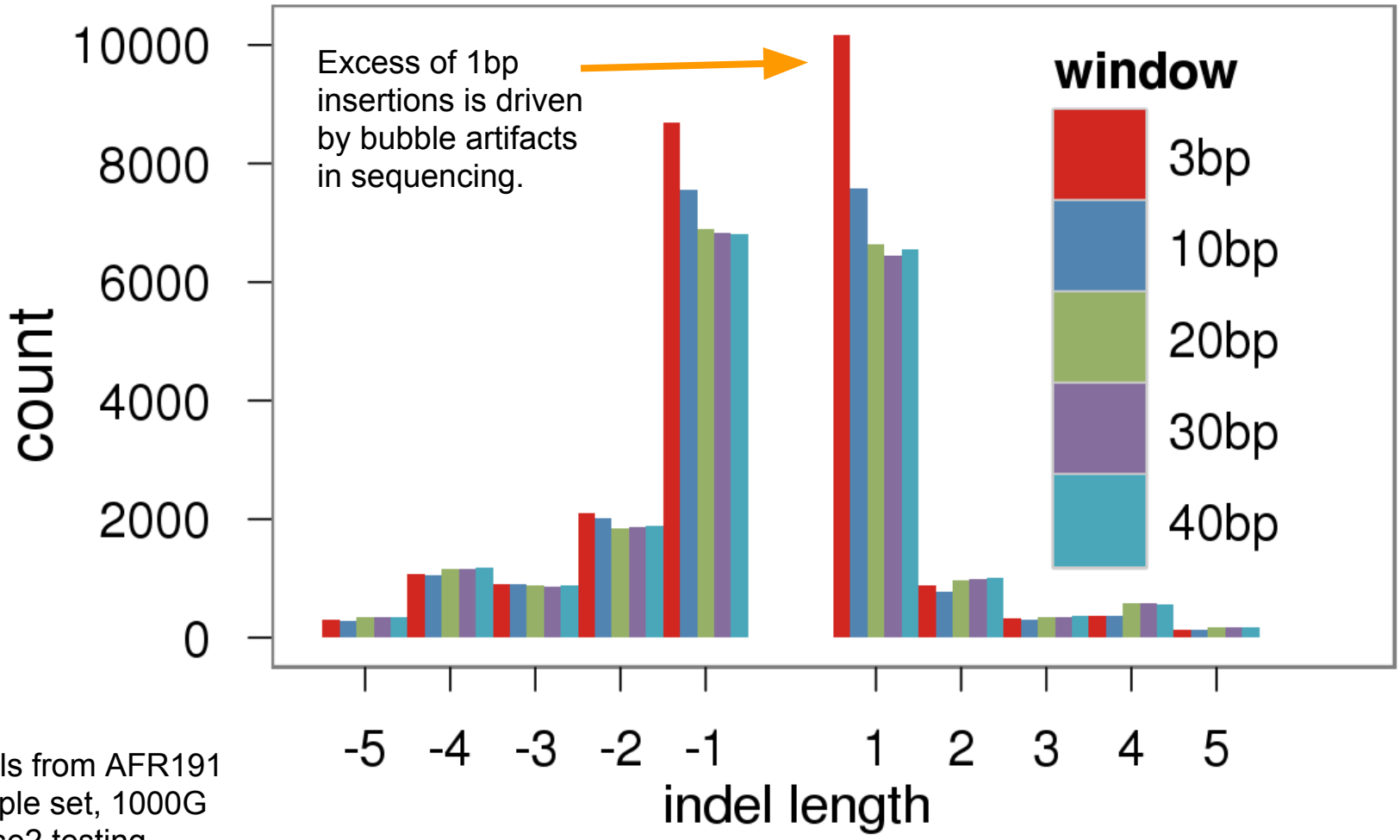
- When the 1000 Genomes samples were genotyped using the chip, 100k of the 2.5 million loci showed no polymorphism (monomorphs).




Omni 2.5 1000G monomorphic loci, BC haplotype calls
count
10000
8000
6000
4000
2000
0
CLASS
biallelic complex
biallelic INDEL
biallelic MNP
biallelic SNP
multiallelic complex
multiallelic INDEL
multiallelic INDEL, SNP, and MNP
multiallelic mixed
multiallelic SNP
multiallelic SNP and MNP
multiallelic SNP, MNP, and complex

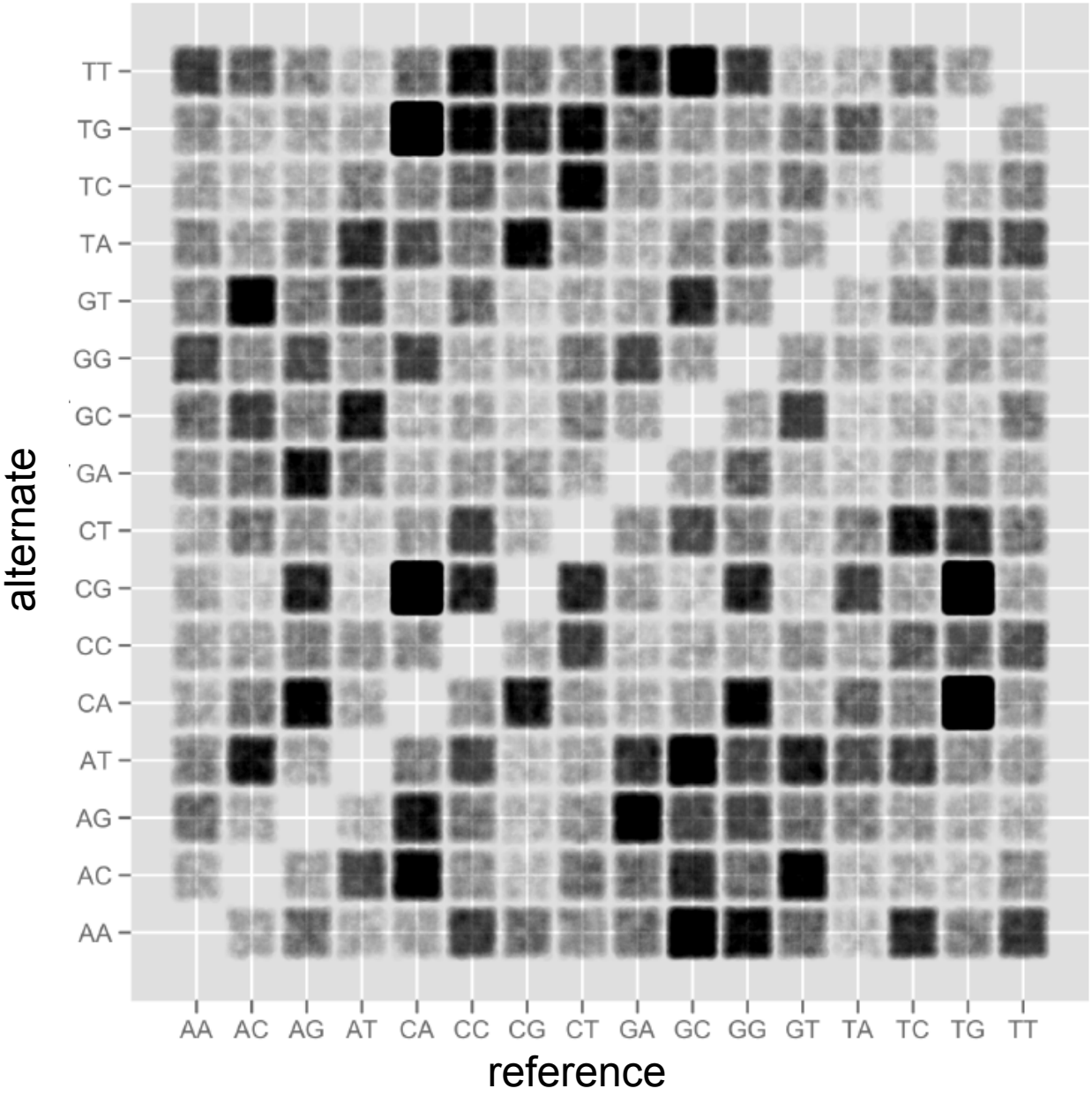

Omni 2.5 1000G polymorphic loci, BC haplotype calls
count
1500000
1000000
500000
0
CLASS
biallelic complex
biallelic INDEL
biallelic MNP
biallelic SNP
multiallelic complex
multiallelic INDEL
multiallelic INDEL, SNP, and MNP
multiallelic mixed
multiallelic SNP
multiallelic SNP and MNP
multiallelic SNP, MNP, and complex

Measuring haplotypes improves specificity

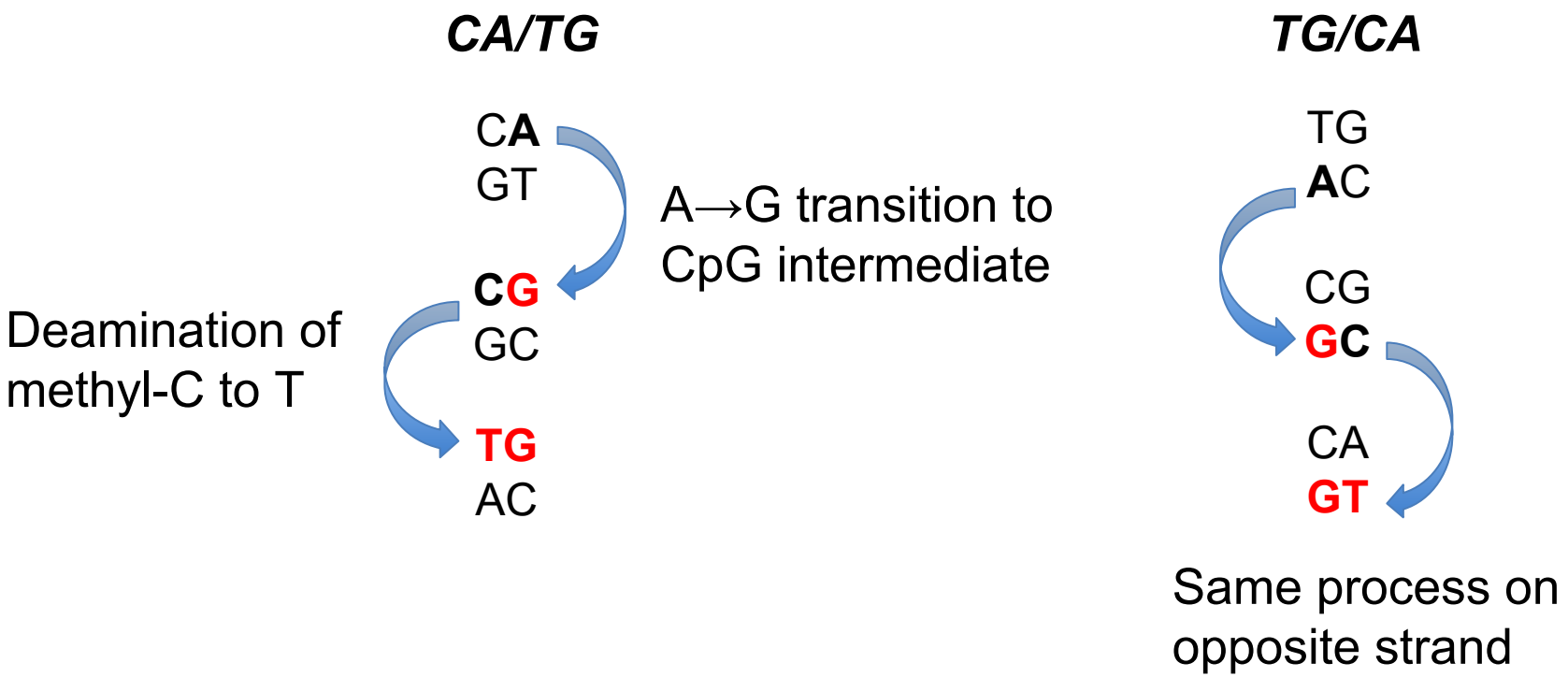


Indels from AFR191 sample set, 1000G phase2 testing.

2bp MNPs and dinucleotide intermediates



Direct detection of haplotypes can remove directional bias associated with alignment-based detection



Overview

1. Genesis of variation (SNPs and indels)
2. Causes of sequencing error
3. Sequence alignment
4. Alignment-based variant detection
5. Assembly-based variant detection
6. Haplotype-based variant detection
7. **Graph-based methods**

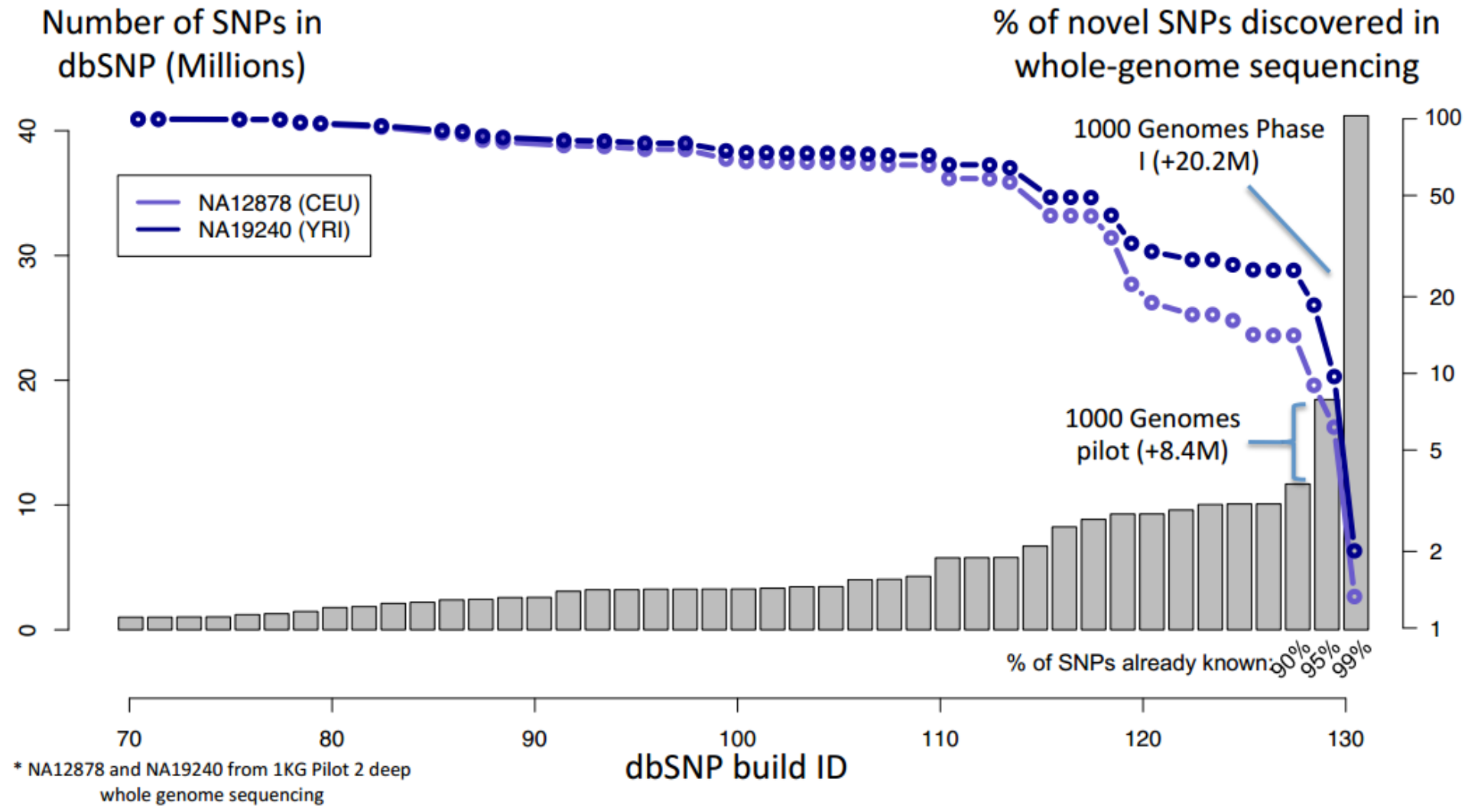
Graph-based methods

1. Variation + sequence = variation graphs
2. Local realignment to a variation graph
3. Constructing a whole genome variation graph
4. Aligning to a whole genome variation graph

(1) Variation + sequence

= variation graph

Most *SNPs* (>99%) are now known



* NA12878 and NA19240 from 1KG Pilot 2 deep whole genome sequencing

Current variant detection is *de novo*

```
GAAACAACTAAATATGTTATCAATCCATCATTTACTTGTACAATAAATAAAGTTCTAAATCACTGCACAGTGTAAAATGG
CAAATAGACTTCCCCATAACACAAAGCCATCCTGAAAAGTTTTGTTCATTTTAGAAGAAAAAATTTTAAAACCTGAGCAC
CATAGGATATGAATTTTGTAAATAATTATGAAAAGAAAACAACAATAATAAAACTCTACTAATCTTAGCTCGATCTTATA
AAAAGTTCCATAATGAGTAATCAAATTTTATTTCCAATTAAATATGTTATCACCCAGTAGTATGCCATACCAGTTTCTAA
TTCATACTTCAATTATCTTCTAATTTAAATTAATGACTATAATTGCTGTTATAAAACAACAGCTCTATAGCCTGCTATTC
AGACCAGTAAATAAGAGTTTAAGGGCTTGTGATAGCAAATGAAGTTTCTTATTGGATTTTAAGAAAAATTTTTATAAAAA
TATGTGAGGTTATTCAATAGAATCACATTTAATTTGCCAAGCATTTTGCAGAATGCCTAGGACTATGTAAGAAGTATTAA
ATTTGCAAGCCCTTTGAATAGTTGTAATTTAAAGATAAAAATTGGTTTAATACCAGACAAAGATAGAAGCACAAGTTAGG
TTATTAGAGAATTTAGCCAGTGTATCAGTTTGTATCGTAAGTCATTGGCAAGAACAACGTGTACTTTTCTGTCACCTCCC
AACTAGCTATGTTTTGAGCAGTAGGAATATTTAATACCCCTTCCTCCCATTTTTCCTTTGTGTTGTCCAAATTCTGACAA
CTCTACTGCCAGATAGCTCAGGGCAAAAATGATAAAGTTCAAGTTAAGAAGGCTCTGCAGTGTTCTCAGTTCTCCTCTGG
TGAAAGAGGAGAAAGGTTGTGTTTAATTATGAATCTGGGATTTCCAAAACTTTACCCATGCCCTGCCTGTCCCCTCATTA
GCATGAAGCTGTTATTTAAATAGTTCAGCAATAACGACTTTAGTAGCCTCCCTAGGTTAAAAAGATTGAAATTAAATGTG
TTTATCTATTGTTCTACTATTCAGTTACCTGATTATAAAATCAAAGATTATTTCATGAAACTCAGTACCCCTTCAGGGAA
AAAAAAAAAAAATTCCCTAAAACAAAGTTAGGAGAGTGAATCGGACCACATGCTTATCTCCAAGGTCTCAATCAAACAGC
AAATGCTTACCCAACTTCTATTCAAAATATTTGCGCCAGTAGTTCTGATATGACCCAAGCAGAGTTCACACATTATTAAT
CTACTCCTTTCAGTCTTCTAGATGTGTTTCCTCCAAAATCTACCAGATTCTCAAATAATTTCAGGAACTTTCTCCAGAAC
AGAAACAAGGTTGTTACTGATACCAACTTTGTCTCCAAACATGGGGAAGATTATCATTGGAAAGATCTATTGATGACCTA
TAATACATAGTTGGAACTGTTTATCCACAGAAGTATTCCCCAAGAATCAACCACAGAGCCAAGATGGAGCTTATGTCATT
GTTATGCATACTTCTTTTACGGCTTGTGAGGGCAGTTCATACTATTCTGATTTTACAACTGAGACCCAAGGAACCTGAGT
GACTTCTAGGCTCCATTATGTCAAAAAAAACTCAAATGTGAGGCTTTGCCTACACTGAGAAACAGTAGTTCAAGAAACGG
TGCCCTGGTTCTGTTAAAATAATCTGAGAGTTATGTGGTAAGTAGTTGAGAGTGAATAGGGTAGCTTTGAGAGGTGACAG
CGTGCTGGCAGTCCTCACAGCCCTCGCTGGCTCCAGGCGCCTCCTCTGCCTGGGCTCCCACTTTGGCGGCACTTGAGGAG
CCCTTCAGCCCACCACTGCACTGTGGGAGCCCCTTTCTGGGCTGGCCAAGGCCGGAGCCGGCTCCCTCAGCTTGCAGGGA
GGTGTGGAGGGAGAGGCGCGAGCGGCAACCGGGGCTGCGCACGGCGTTTGCGGGCCAGCTGGAGTTCTGGGTGGGCGTGG
GCTTGGCGGGCCCCGCACTCGGAGCAGCCGGCCAGCCCTTCCAGCCCCAGGCAATGAGAGGCTTAGCACCCGGGCCAGCA
GCTGCGGAGGGTGTACTCCGTCCCCCAGCAGTGCCAGCTCACAGGCGCTGCGCTCAATTTCTCACCGGGCCTTAGCTGCC
TTCGCGCGGGGGGTGCTCGGGACCTGCAGCCCGCCATGCCTGAGCTCCCACCCCCTCCATGGGCTCCCGTGCGCCCGAGC
CTCCCCGATGAGCACCACCCCCTGCTCCACGGCGCCCAGTCCCATCGACCACCCAAGAGCTGAGGAGTGCGGGCGCACGG
CGCGGGACTGGCAGGCAGCTCCACCTGCAGCTCTCGTGCGGGATTCACTGGGGGAAGCCAGCTGGGCTCCTGAGTCTGGT
GGGGACGTGGAGAACCTTTATGTCTAGCTCAGGGATTGTAAATACACCAATCGGCACTCCGTATCTAGCTCAAGGTTTGT
AAACACACCAATCAGCACCCTGTGTCTAGCTTAGTGTTTGTGAACGCACCAAGCCACACTCTGTATCTAGCTACTCTGGT
GGGGCTTTGGAGAACCTTTGTGTCCACACTCTGTAGCCAGCTAATCTGGTGGGGACATGGAGAACCTTTGGGTGTAGCTC
AGGGATTGTAAACGCACCAATCAGCGCCCTGTCAAAACAGAGGACTCGGCTCTACCAATCAGCAGGATGTGGGTGGGGCC
AGATAAGAGCATAAAAGCAGGCTGCCTGAGCCAGCAGTGGCAACCCGCTTGGGTTCCCTTCCACACTGTGGAAGGTTTGT
TCTTTCACTGTTTGCAATAAATCTTGCTGCTGCTCACTCTTGGGGTCCACACTGCTTTTATGAGCTGTAACACTCACTGC
GAAGGTCTGCAGCTTCACTCCTGAAGCCAGCGAGACCACGAGCCCACCAGGAGGAACCAACAACTCCAGAAGCGCCGCCT
TAAGAGCTGTAACACTCACCGTGAAGGTCTGCAGCTTCACTCCTGAGCCAGCGAGACCACGAACCCACCAGAAGGAAGAA
ACTCCGAACACATCCGAACATCAGAAGGAACAAACTCCAGATGCGCCACATTAAGAGCTGTAACACTCACCGCGAGGGTC
CCTGGCTTCATTCTTGAAGTCAGTGAGACCAAGAACCCACCAATTTTGGACACAGTTTGACAATAAATTTACACTCAAAT
ATCTCTAAGGAATCAAACTTACAGATTAATAATTAGTAATCAGGTCACGTAAAGTAAATTATAAAAGAGCATTGATACCA
AGATTGGCAGAAAGTTTTTGTGTGACAAAACCAAGTTTTGGCTAAGATACACACTGCTGATGGGAGTCTAAATTGCTGT
ATATGTCGGGAAAACAAGTTGTCTTTATCTTGATGTTATAAATAACCTATGACCCAGAAATTTAACTCCTAGACATATAC
TCTAGTGAAACTCTTGAACGTGTGCGTCCAAAGACATTTATAAACATGATCTTAGTAGTATTGCTTTTAGTAGCAAATTC
TGGAAACATCCCAAATGTCTATCAATAGTGGAATTGATTTGAAAGGGGTGTGGAATGGTAATATAATGGAATAGCCTACA
GCTGTTTAGATAAAGGAACTCCAATTAAACATACCAACAAAGATACATTTCAAAAACAAGACGTTGAAAGGAAAAAAGTC
ATCAAAACAATACACAACATTCTACCACATTTTTATAAATTCTCAAAATATGCAATATTAAACATGCATTATTTAGGGAG
GCATTCAATGTAGCAATGCATTTTTAAGAGGCTGGGATGATAAATGTAAAATTCAGAACAGGTATTATCTCTGGGAACAG
GAAGAGGAGGATGCAGTGTTGGGAAGAAATACATATAAGTACAGCAGTAGAGGCAGACTTTTTTTTCCTTTTCCTTTTTC
CTTTATTTTTCCTAGCTTTCTTTTTCTTTAGCTATGGTATTTCTTTAGCTATGGTATGGTATGGTATGTACTTTCCATAT
ATCATACAGTATATTTTGGTGTGCATGAAATAGTCCTTAATAACATTTTTATTATTTTTTATTTGACGTTTAAGTTCAGG
GGTACATGTGCAGATTTGTTATATAGGTAAACTTGTGTCATGTGGTTTTGTTGCACAAATTATTTTCTCACCCAGGTATT
AAGATTAGTACCCATTAGTTATTTCTCTTGATCCTCTCCCCTCTCCCACTCTCACCCTTCCAATAGGCCCCAG
TGTGTGCTGTTCCCCTCTATGTGTCCATGTGTTCTCATCACTTAGCTCTCACTCATAAGTGAGAACATGCAGTATTGGGT
TTTCTGTTTCTGCGTTAGCTTGCCAAGGATAATTGCCTCCAGCTCCATATTCCTGCAAAAGACATAATTTTGTTCCTTTA
TATGGTTGCATAGTATTCCATGGTGTATATGTACCGCATTATCTTTAGCCAGTCTATCATTGATGAGCACTTAGGTTGAT
TCCATGTCTTCGCTCTTAACATTTTTAAACAGTCTCTGAGTAGAATAGGGTAGGCTGGTGTAAGGAATTACTGTTTTTAA
TTGGAAGCAATTACATACTGCATTAAAAAGCATATATATATAGACACACATATATATGTTTTACATTCACATAATCAAAC
CAAAGCTTTCAGAATTTTCCGCCAAAAAGCATGTGTGGAGGGAGAGAGAGGGGTGTTATGGTGCATTTGTAGGCCCTGAA
ATAAAGTGATCTCCATATTTGCTTCCATTTGTGTATAAATATTTTTTAAGCATATATGCATCCTAGAGTATGGAAAGAAA
ATTTCTGGGAAGATTTGCAAGAATCTGTGGCAGTTGAGAGTAGGTTCACTTTCGCTTTATTGTGTAATTTATTGTATTTT
TCATTTAATTGTACTTTGTAAACTAAATATTTATTGTATATTTTACTTCATTTTTTAATTGCCATATGCAGCTTTAATTT
TATAACTAAATTACGTTCTGACCCAGGGCAGGAGGTGGGAGAAGAGAATGAGAGAAGGGCGGAATCCTCAAGACAGCAGC
ATTCTGCGCTCCGCCCTGGAACACCCAGCGCAGCTACCCCATCCCGCCCAGGCCGGCCACCTGGCGTCGAAGCCAGGCGC
GAGGGGCGGAGGCGGAGGCGGGGGCGGGGGCGGAGGCGGAGGCGGAGGCGGAGGCGGAGGCGGAGGCGGAGGCG
GAGGCGAGCACCGCCCCACCGCGCACGCTGCGGTTGCCCCGGCAGCCGCGCCCTGCGTGGCGGAAGCTCACAATCAGCCC
GGTCCCTCCGGCTTCCACCCCGCCCCCTGCGCTCACCTGCCCGCGCGCTCGCCTTCCGGGGACCCGGGGCCCATGGACAC
ATACACCCAGCCCTGCTGTCCCGCGCGCCAGCTCACCAGCCCTACCCAAGGGACATCATTCACGCCTGGGCGCCTCCGCC
GGGCTCCGGGAGCCCAAGGTCGCGGCTGGGCCAGCGCTGAGCGTCAGAGGACGAGAGCAGGGGCCTCCCCGGTCGCCCCA
```

Genome (FASTQ)

```
@sample006.000000000
ACTGAATCAATGCCACTTGATTTGTGAAAATAAGGATTTCCCAATCAGGGACTTCAATGTTGCCTCTGTTAGCACCCATCTCTAACGGAGAATTAATTAG
+
HHHIHIIHHHHIHIGHGIIIIGHHHIGEGIFIIGFHIFGIIHGIDGIDIGGHGFIIIIIIICIHEIIIIIIIHDIIBIIGH@IIBGIBIEIHIIBIIIIGII
@sample006.000000001
CATGTCCTTGTGGAAAGCAGGTAAACTTTAGAAAATTATAAGGAAGCAGAGGAAAGTAACTACAATCCCACCACTCAAAAACTTTTAAAAAGTTAACATT
+
HHIHHIIIIIIIGHHIIIGHHIFIIIIFHGHHIIGIHHIHIIHIIIIFIIIDAIDIIFHIDIEACCIIIFHD=IHICIIEHFIIIIECIIIICDBBIIIFD9
@sample006.000000002
AAAGAATATGATGTTCTCTTGTAATTAGCAAGGGCTGATAGGGTTCAAGAGGTGACACTGATATGGATCTTGAGTAACTGTTGACAAATACTTCCCTTTA
```

Reads (FASTQ)

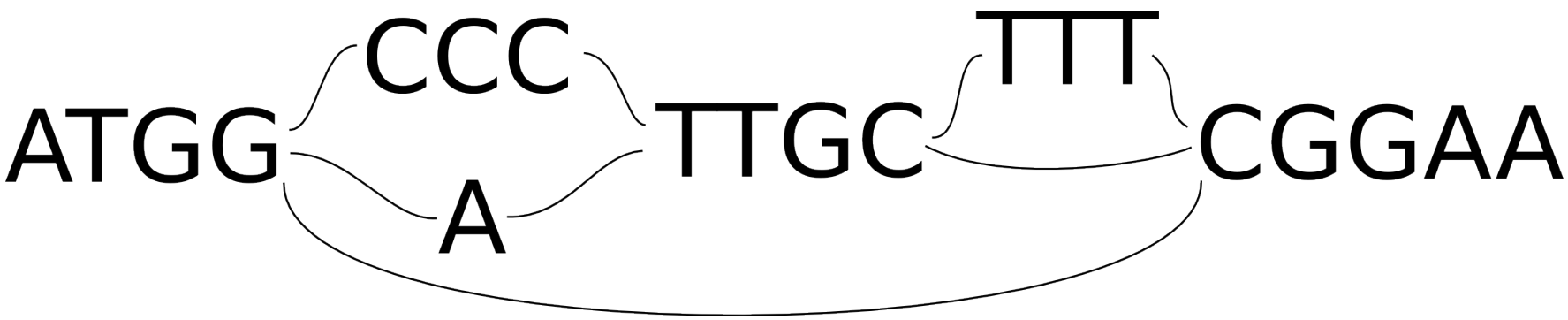
alignment and variant calling

```
##source=mutatrix population genome simulator
##seed=1373972756
##reference=chrQ.fa
##phasing=true
##commandline=mutatrix -S sample -p 2 -n 100 chrQ.fa
##filter="AC > 0"
##INFO=<ID=TYPE,Number=A,Type=String,Description="Type of each allele (snp, ins, del, mnp, complex)">
##INFO=<ID=NA,Number=1,Type=Integer,Description="Number of alternate alleles">
##INFO=<ID=LEN,Number=A,Type=Integer,Description="Length of each alternate allele">
##INFO=<ID=MICROSAT,Number=0,Type=Flag,Description="Generated at a sequence repeat loci">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##INFO=<ID=AC,Number=A,Type=Integer,Description="Total number of alternate alleles in called genotypes">
##INFO=<ID=AF,Number=A,Type=Float,Description="Estimated allele frequency in the range (0,1]">
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of samples with data">
##INFO=<ID=AN,Number=1,Type=Integer,Description="Total number of alleles in called genotypes">
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  sample001       sample002
chrQ    1252    .       C       A       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|0     1|0
chrQ    3646    .       T       TC      99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=ins      GT      0|0     0|1
chrQ    6283    .       C       T       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|0     0|1
chrQ    7412    .       C       T       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|0     1|0
chrQ    7935    .       T       C       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|0     1|0
chrQ    8131    .       T       C       99      .       AC=2;AF=0.5;AN=4;LEN=1;NA=1;NS=2;TYPE=snp       GT      0|1     0|1
chrQ    8682    .       AA      TG      99      .       AC=1;AF=0.25;AN=4;LEN=2;NA=1;NS=2;TYPE=mnp      GT      0|0     0|1
chrQ    10926   .       T       C       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|0     0|1
chrQ    11921   .       G       GTT     99      .       AC=1;AF=0.25;AN=4;LEN=2;NA=1;NS=2;TYPE=ins      GT      0|1     0|0
chrQ    12955   .       T       G       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|0     1|0
chrQ    13808   .       T       TG      99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=ins      GT      1|0     0|0
chrQ    15271   .       A       G       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|1     0|0
chrQ    15407   .       A       C       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      1|0     0|0
chrQ    16486   .       C       G       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      1|0     0|0
chrQ    16563   .       T       A       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|1     0|0
chrQ    16748   .       GTT     G       99      .       AC=1;AF=0.25;AN=4;LEN=2;NA=2;NS=2;TYPE=del      GT      0/0     0/1
chrQ    17697   .       G       C       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|0     0|1
chrQ    19568   .       A       G       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|1     0|0
chrQ    20750   .       G       A       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|0     1|0
chrQ    21532   .       T       C       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      1|0     0|0
chrQ    22291   .       C       T       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|1     0|0
chrQ    23193   .       G       A       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|0     0|1
chrQ    23954   .       CTAA    TTAA    99      .       AC=1;AF=0.25;AN=4;LEN=4;NA=2;NS=2;TYPE=mnp      GT      0/0     0/1
chrQ    24467   .       C       T       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|0     0|1
chrQ    26100   .       G       A       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|1     0|0
chrQ    29654   .       T       A       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      1|0     0|0
chrQ    30062   .       T       C       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|1     0|0
chrQ    31790   .       A       G       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|1     0|0
chrQ    32792   .       T       C       99      .       AC=3;AF=0.75;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      1|1     0|1
chrQ    33376   .       CC      C       99      .       AC=2;AF=0.5;AN=4;LEN=1;NA=1;NS=2;TYPE=del       GT      1|0     0|1
chrQ    33403   .       T       C       99      .       AC=2;AF=0.5;AN=4;LEN=1;NA=1;NS=2;TYPE=snp       GT      0|1     0|1
chrQ    33802   .       A       G       99      .       AC=2;AF=0.5;AN=4;LEN=1;NA=1;NS=2;TYPE=snp       GT      0|1     1|0
chrQ    34450   .       C       T       99      .       AC=4;AF=1;AN=4;LEN=1;NA=1;NS=2;TYPE=snp GT      1|1     1|1
chrQ    34716   .       G       A       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|0     1|0
chrQ    35484   .       G       A       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|0     1|0
chrQ    36547   .       G       A       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|0     0|1
chrQ    38015   .       T       A       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|0     1|0
chrQ    38281   .       T       C       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|1     0|0
chrQ    40467   .       A       G       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      1|0     0|0
chrQ    40581   .       A       G       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|0     1|0
chrQ    40601   .       A       T       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|0     0|1
chrQ    43268   .       G       A       99      .       AC=1;AF=0.25;AN=4;LEN=1;NA=1;NS=2;TYPE=snp      GT      0|1     0|0
```

Variation (VCF)

Variation graphs

A variation graph represents many genomes in the same context.



Nodes contain sequence and directed edges represent potential links between successive sequences.

A multiple sequence alignment is a variation graph

traditional MSA

(a)

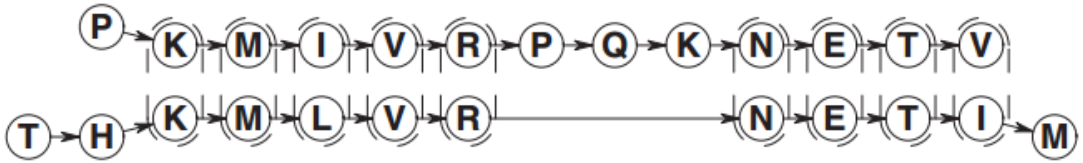
```
.  .  P  K  M  I  V  R  P  Q  K  N  E  T  V  .
T  H  .  K  M  L  V  R  .  .  .  N  E  T  I  M
```

consensus sequence

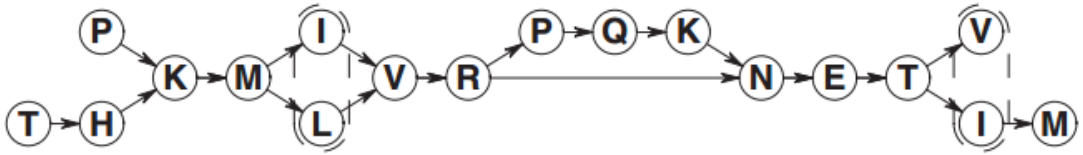
(b)

positionally-matching regions aligned

(c)

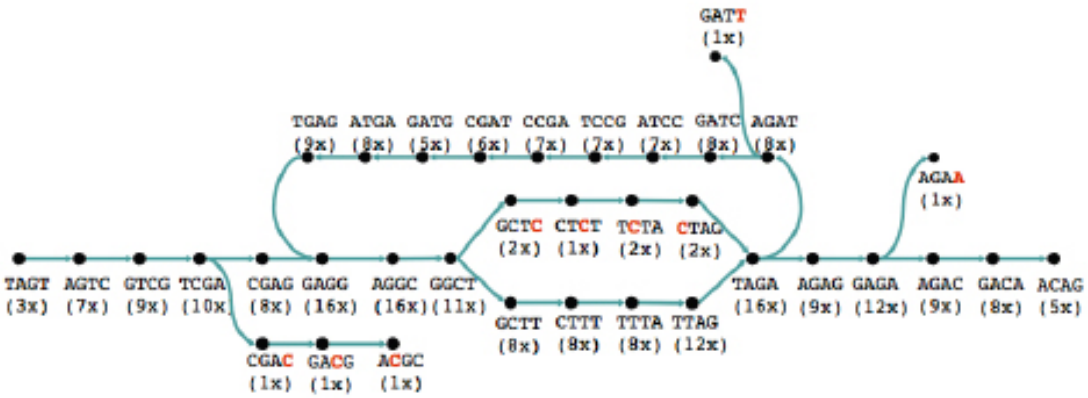


(d)



Christopher Lee, Catherine Grasso, Mark F. Sharlow. **Multiple sequence alignment using partial order graphs.** *Bioinformatics*, 2002.

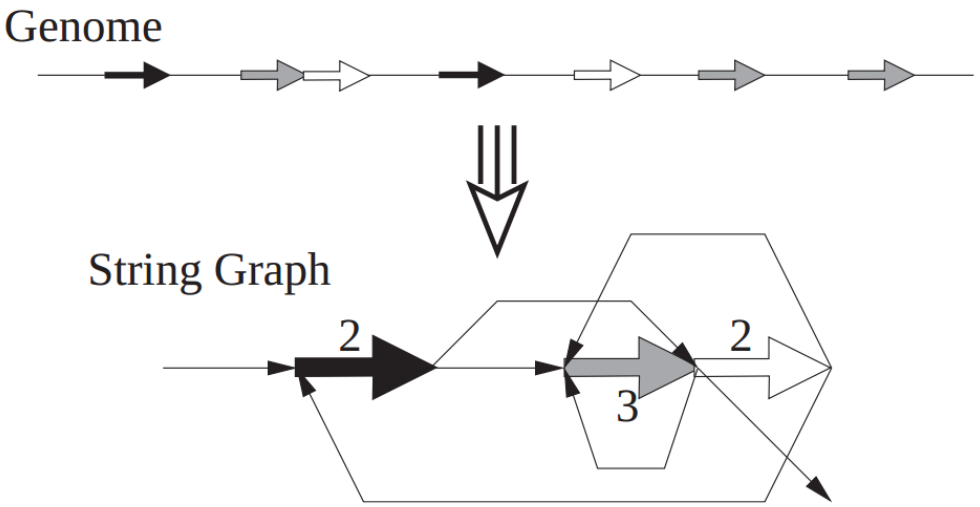
Assembly graphs are variation graphs



A de Bruijn graph can be converted into a variation graph (as previous) by setting node sequences to the first letter in the kmer and compressing non-branching runs into single nodes.

http://plus.maths.org/content/os/issue55/features/sequencing/index, credit Daniel Zerbino

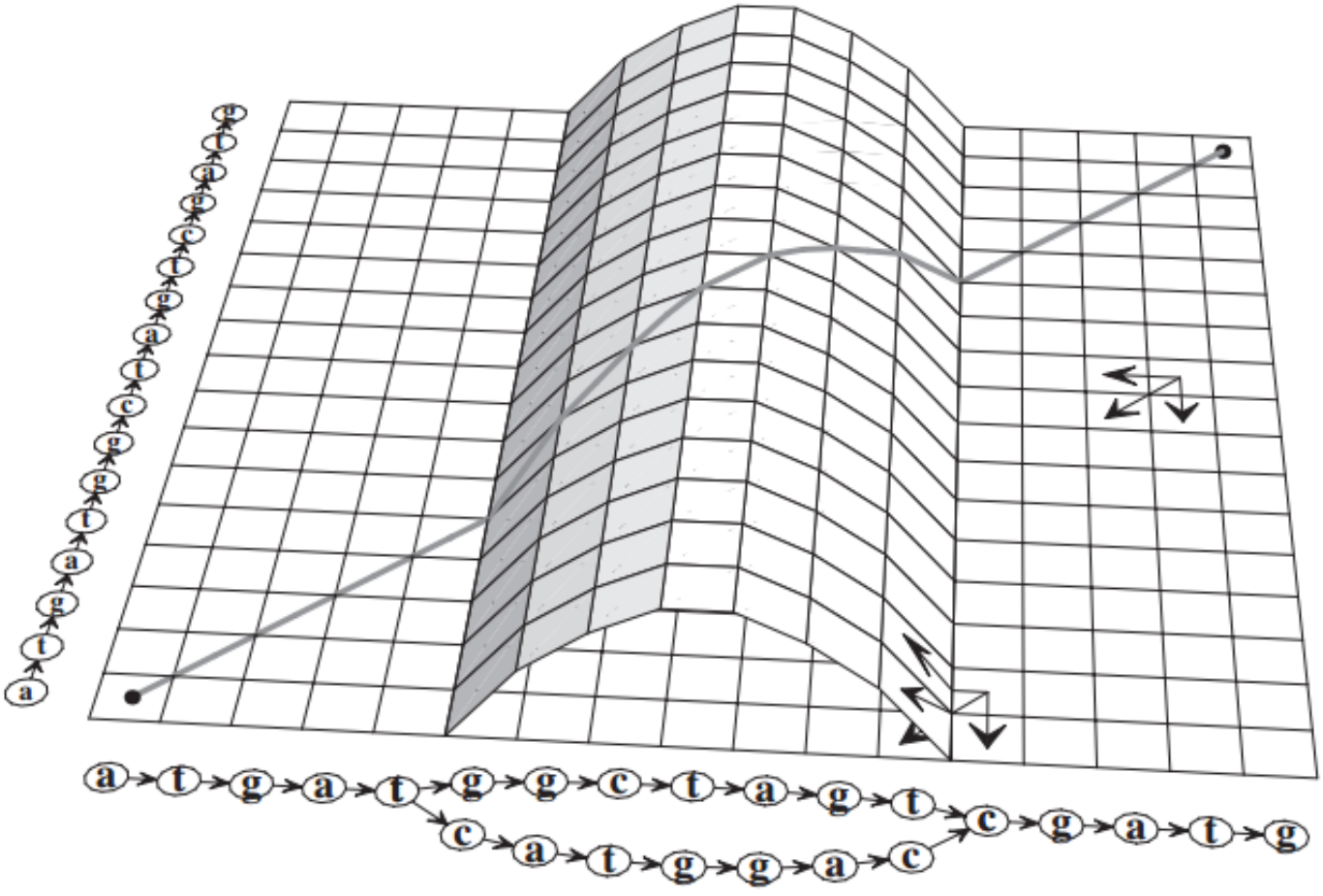
A string graph follows the same format as the variation graph.



Eugene Myers. **The fragment assembly string graph**. *Bioinformatics*, 2005.

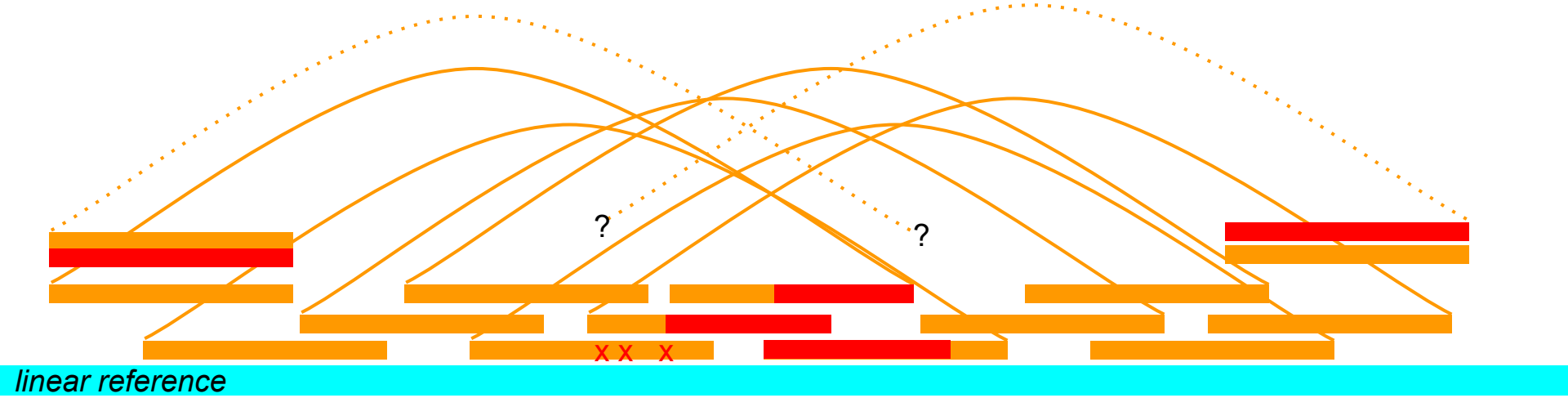
(2) Local realignment to a variation graph

Local alignment against a graph

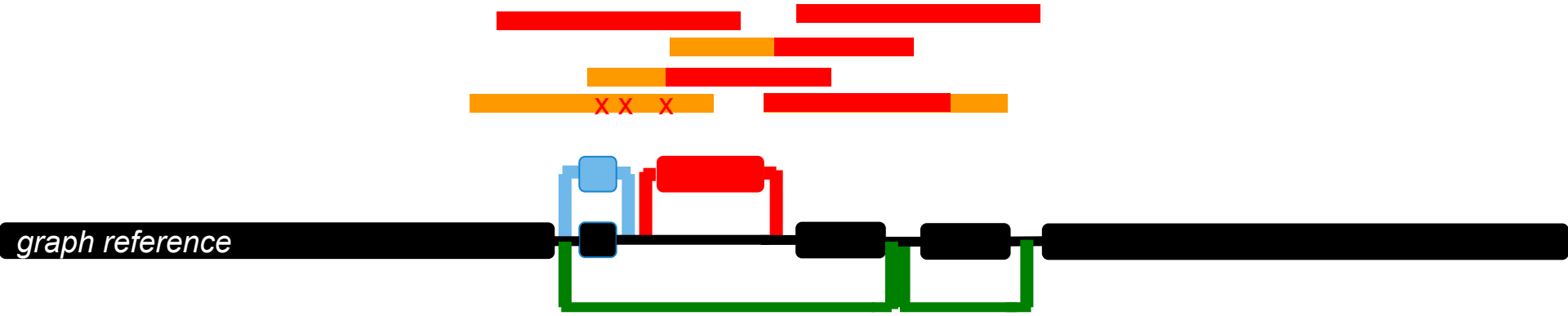


Christopher Lee, Catherine Grasso, Mark F. Sharlow. **Multiple sequence alignment using partial order graphs.** *Bioinformatics*, 2002.

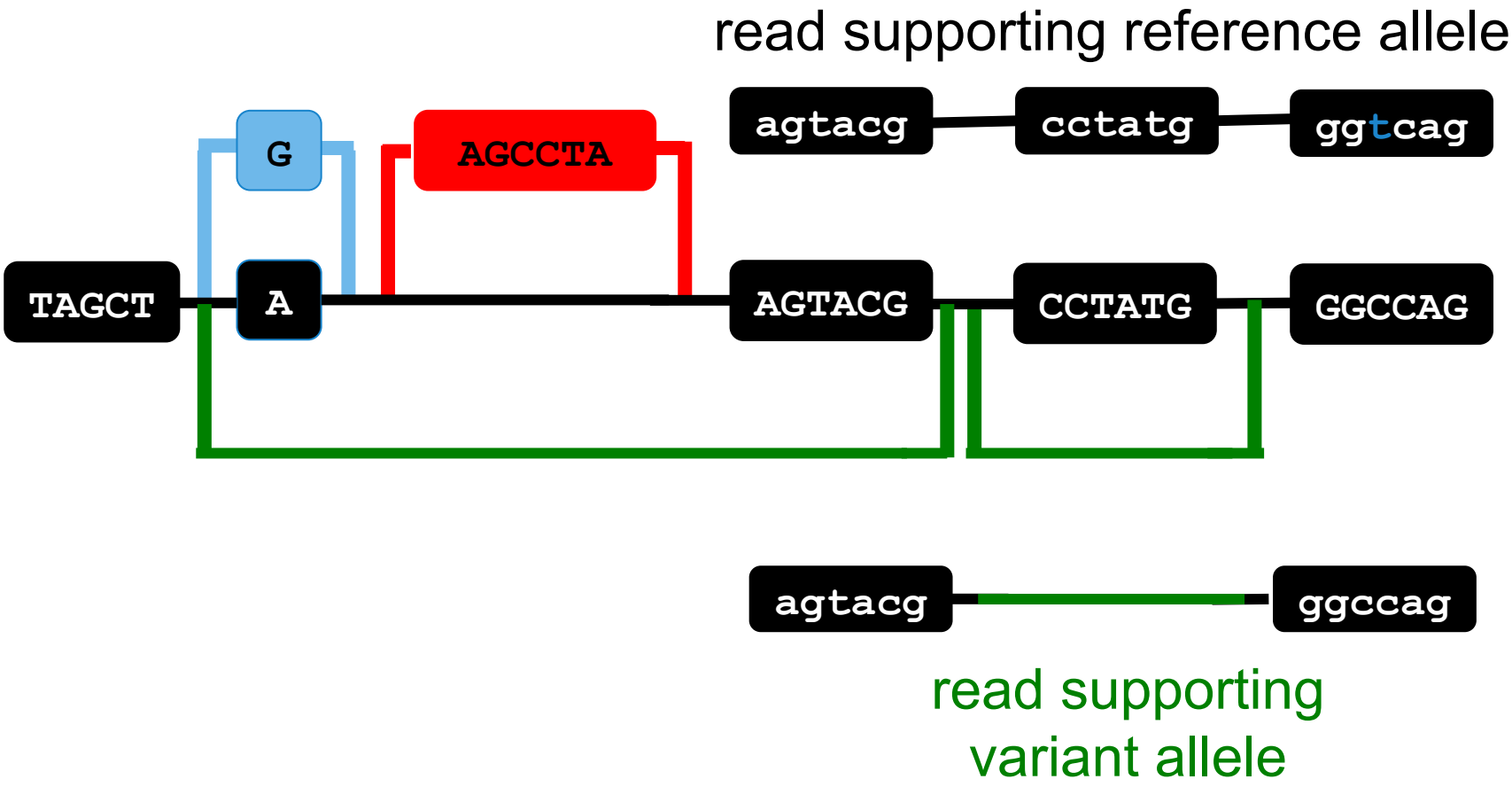
Seeding graph-based alignments

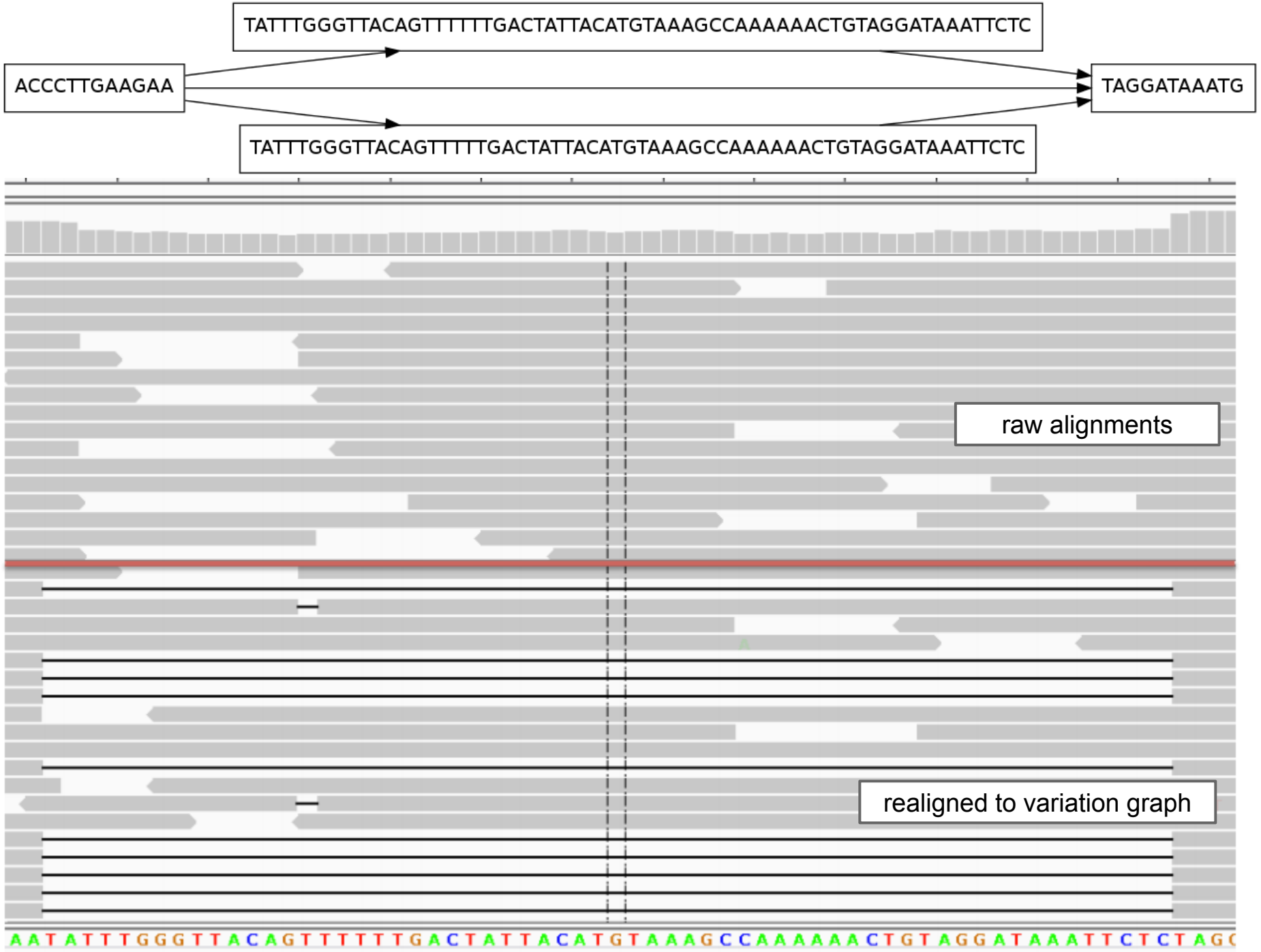


Test imperfectly-mapped reads against graph.

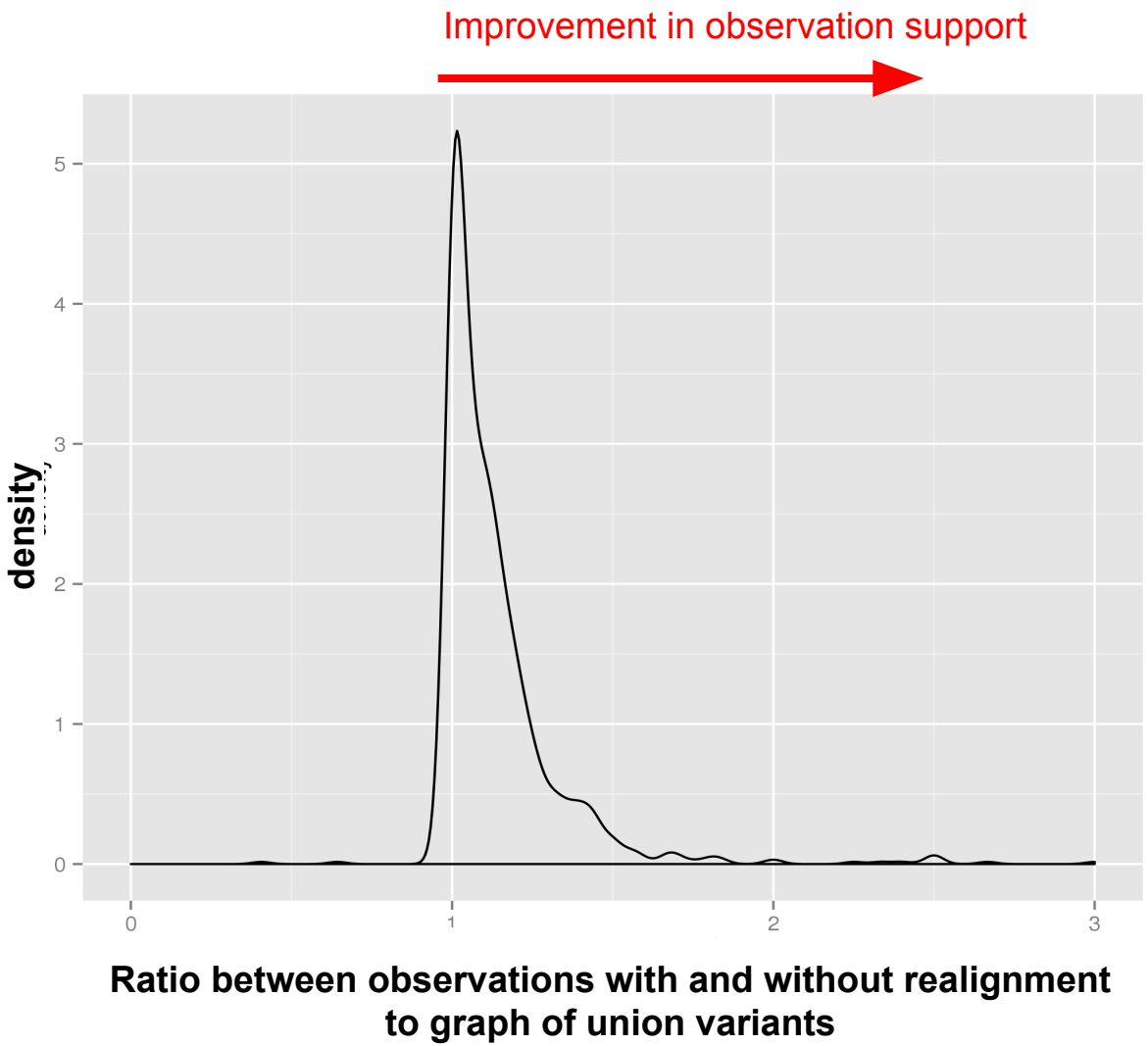


Detecting variation on the graph




TATTTGGGTTACAGTTTTTTGACTATTACATGTAAAGCCAAAAAAACTGTAGGATAAATTCTC
ACCCTTGAAGAA
TAGGATAAATG
TATTTGGGTTACAGTTTTTGACTATTACATGTAAAGCCAAAAAAACTGTAGGATAAATTCTC
raw alignments
realigned to variation graph
AATATTTGGGTTACAGTTTTTTTGACTATTACATGTAAAGCCAAAAAAACTGTAGGATAAATTCTCTAG

realignment to variation graph reduces reference bias



Standard alignment is frustrated even by small variants

(tested against 1mb segment on chr20 using 1000Gp3 union allele list)

(3) Constructing a whole genome variation graph

Objective

Construct a whole genome variation graph to serve as our reference for sequence analysis.

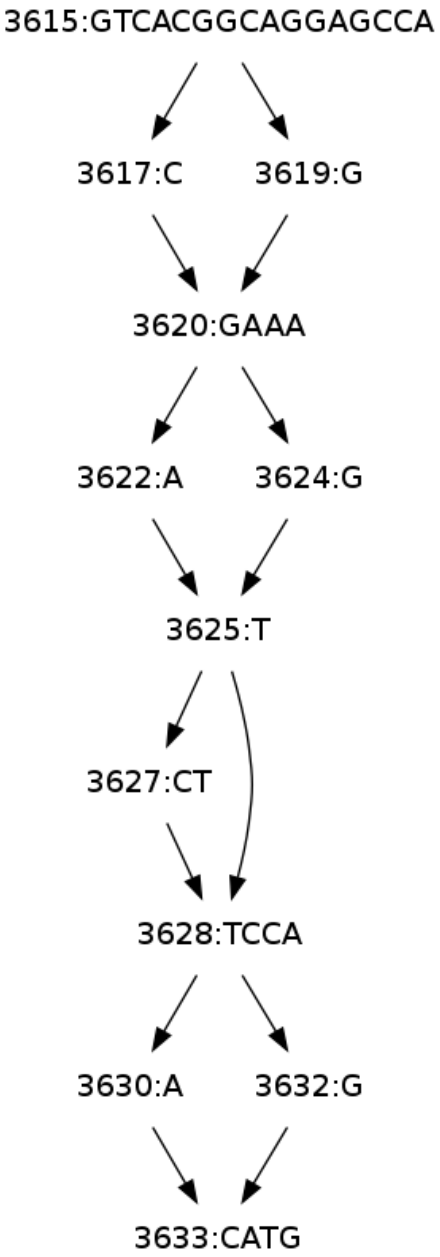
We'll need:

(1) a data model
(2) a serialization format
(3) an API (construct, align, view, …)

Conceptual model

We have sequences (nodes), and allowed linkages between them (edges).

Paths (walks through the graph) are also useful.



Data model

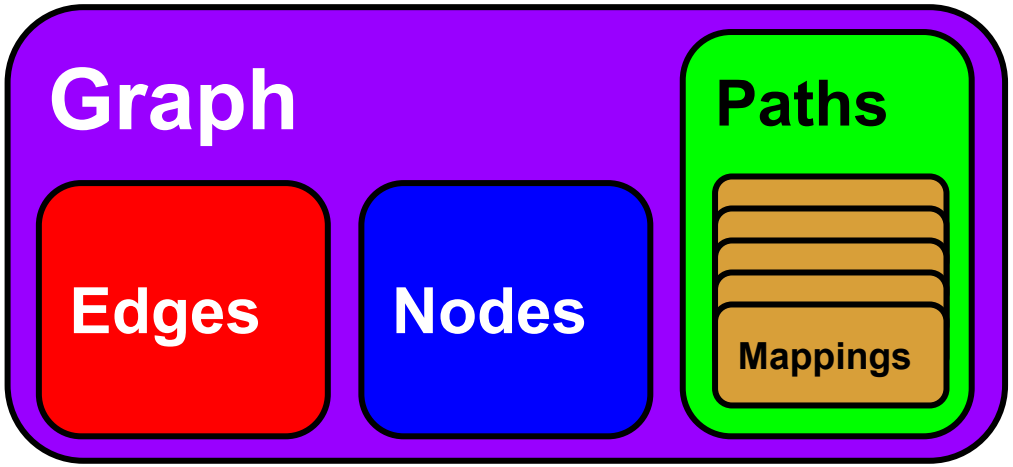
Use protocol buffers to define objects

Graph, Node, Edge, Path, Alignment, Edit

```
message Graph {
    repeated Node node = 1;
    repeated Edge edge = 2;
    repeated Path path = 3;
}

message Edge {
    required int64 from = 1;
    required int64 to = 2;
    optional bytes data = 3;
}

message Path {
    optional int32 target_position = 1;
    optional string name = 2;
    repeated Mapping mapping = 3;
}

message Node {
    optional string sequence = 1;
    optional string name = 2;
    required int64 id = 3;
    optional double quality = 4;
    optional int32 coverage = 5;
    optional bytes data = 6;
}

message Mapping {
    required int64 node_id =
1;
    repeated Edit edit = 2;
}
```

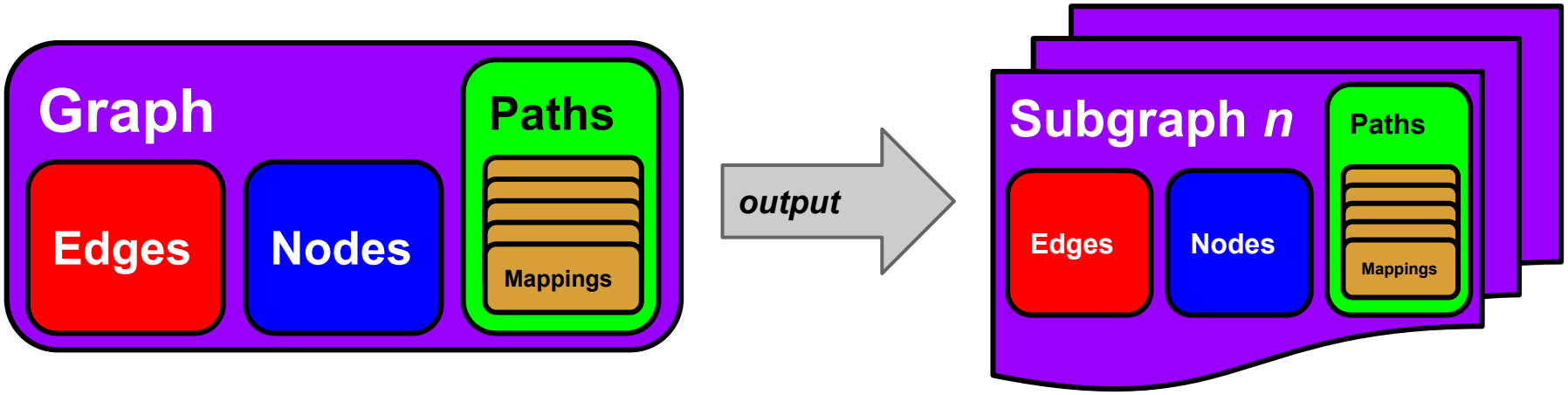
Format

Basic entity is a Graph, which is composed of nodes, edges, and paths.



Serialization

To serialize the graph, we generate a stream of sub-graphs that can be reassembled into the whole.



in memory **serialized stream**

Construction

```
POS     ID      REF     ALT
...
```

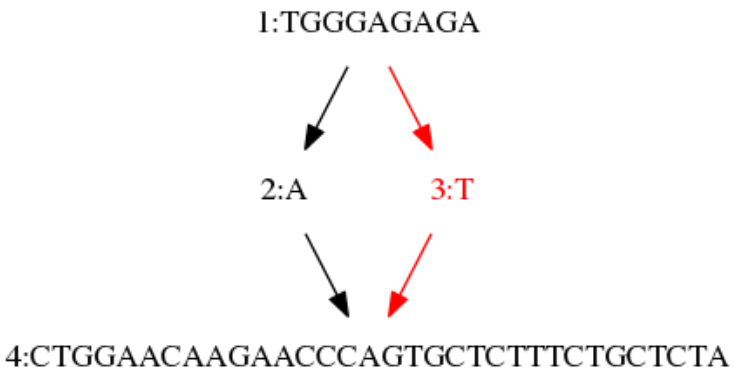
1:TGGGAGAGAACTGGAACAAGAACCCAGTGCTCTTTCTGCTCTA

For each variant

1. cut the reference path around the variant
2. add the novel (ALT) sequence to the graph

Construction

```
POS     ID      REF     ALT
10      .       A       T
...
```



For each variant

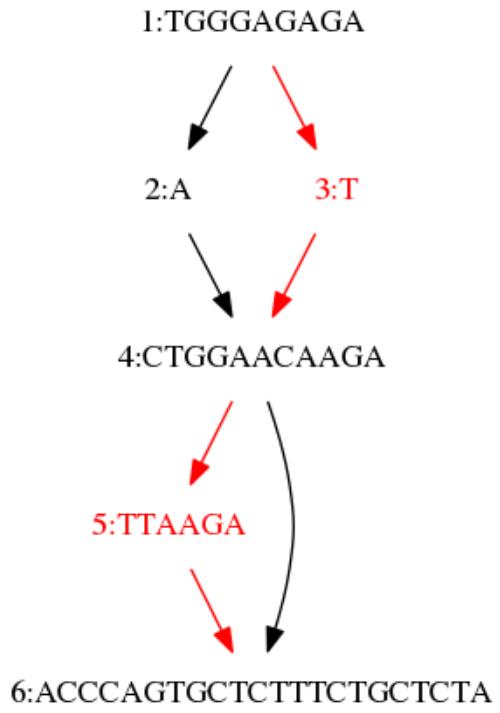
1. cut the reference path around the variant
2. add the novel (ALT) sequence to the graph

Construction

POS	ID	REF	ALT
10	.	A	T
21	.	A	ATTAAGA
...			

For each variant

1. cut the reference path around the variant
2. add the novel (ALT) sequence to the graph

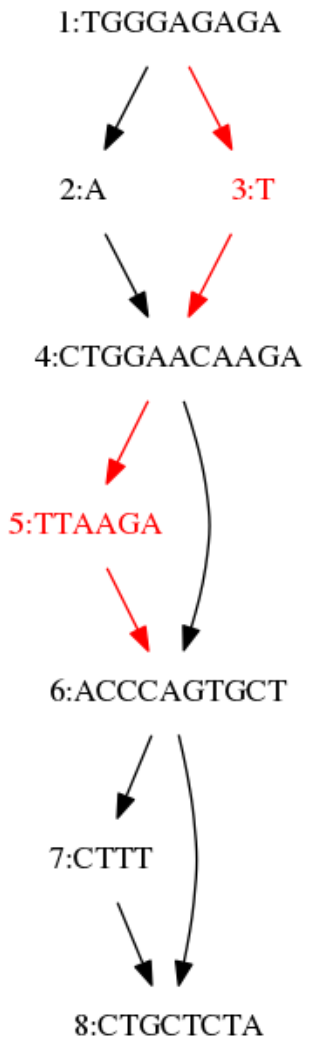


Construction

POS	ID	REF	ALT
10	.	A	T
21	.	A	ATTAAGA
31	.	TCTTT	T

For each variant

1. cut the reference path around the variant
2. add the novel (ALT) sequence to the graph



Indexing

Instead of indexing the serialized graph representation (e.g. BAM, VCF), write the graph into a disk-backed key/value store:

{"key":"+g+26+n", "value":{"id": 26, "sequence": "TATTTGAAGT"}}
{"key":"+g+26+p+1+103", "value":{"node_id": 26, "edit": []}}
{"key":"+g+26+t+24", "value":{"from": 24, "to": 26}}
{"key":"+g+26+t+25", "value":{"from": 25, "to": 26}}
{"key":"+g+27+f+29", "value":{"from": 27, "to": 29}}
…
{"key":"+k+TCATACTACTG+69", "value":-2}
{"key":"+k+TCATACTACTG+70", "value":-4}
{"key":"+k+TCATACTACTG+72", "value":-5}
{"key":"+k+TCATATGTCCA+167", "value":29}
{"key":"+k+TCATATGTCCA+169", "value":-1}
{"key":"+k+TCATATGTCCA+171", "value":-2}
...

This allows our graphs and kmer indexes to have > main memory size. (Uses rocksdb.)

Command-line API

```
usage: vg <command> [options]
commands:
  -- construct     graph construction
  -- view          format conversions for graphs and alignments
  -- index         index features of the graph in a disk-backed key/value store
  -- find          use an index to find nodes, edges, kmers, or positions
  -- paths         traverse paths in the graph
  -- align         local alignment
  -- map           global alignment
  -- stats         metrics describing graph properties
  -- join          combine graphs via a new head
  -- ids           manipulate node ids
  -- concat        concatenate graphs tail-to-head
  -- kmers         enumerate kmers of the graph
  -- sim           simulate reads from the graph
  -- mod           filter and transform the graph
  -- surject       map alignments onto specific paths
```

(4) Aligning to a whole genome variation graph

Constructing a whole human genome variation graph

Constructed 1000G phase3 + GRCh37 variation graph using `vg construct`.

5h20m on 32-core system @Sanger

3.07G on disk

3.181 Gbp of sequence in graph

286 million nodes (11 bp/node)

376 million edges (8.5 bp/edge)

Indexing 1000G+GRCh37

for each node:

for each *k*-path extending no more than *n* edges or *k* bp away from the node:

→ index the k-mers of the *k*-path

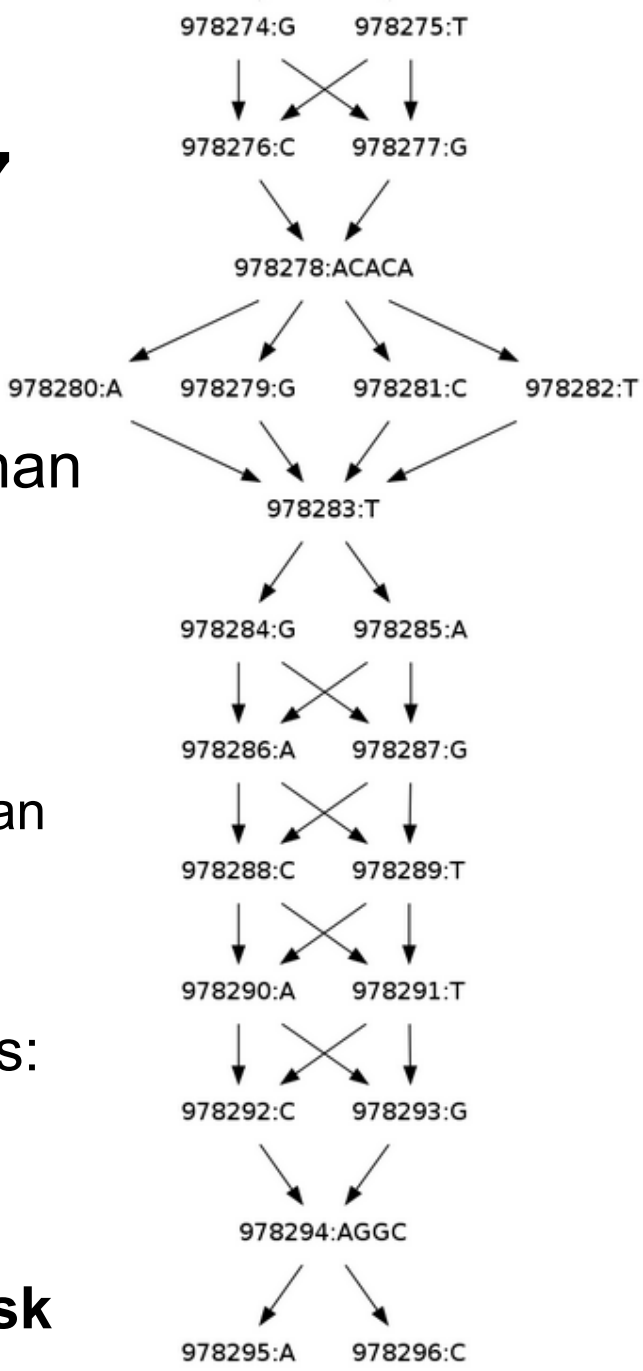
(Edge bounding limits the k-mer space, which can be very important in real data. See →)

for 27-mers crossing no more than 11 edges:

52h on a 32-core host @Sanger

Index is **175G** and on Lustre

Can align **70 read/s/CPU with index on disk**



Aligning to 1000G+GRCh37

For each k-mer in the read

    If the k-mer is informative

        → sort hits by node id

For each cluster of hits : max(id)-min(id) < *M*

    Get local region of graph from index

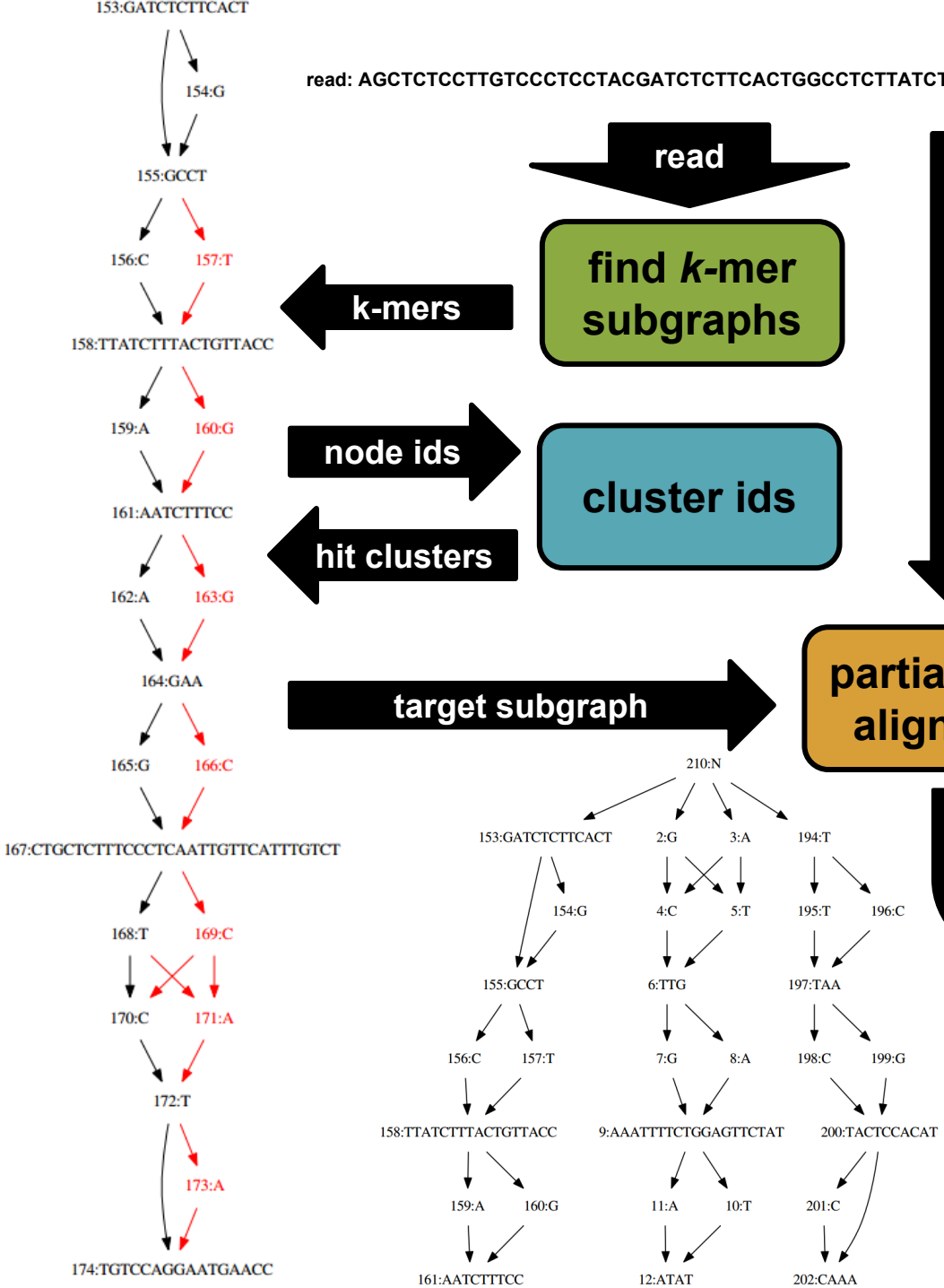
Join the subgraphs together

Align to the joined graph using POA → alignment

If alignment failed, decrease k-mer length or stride

8000 read/s on 32-core machine @Sanger (cached)

~ 40 hours for a deep 150x2 Illumina X10 run



k-mer based alignment of short reads to a variation graph

Alignment

Thanks!

Gabor Marth
Deniz Kural
Wan-Ping Lee
Alistair Ward
Richard Durbin
Josh Randall
Benedict Patton
Zamin Iqbal
Jerome Kelleher
Jared Simpson
David Haussler
Daniel Zerbino
Jouni Siren