Variant calling and filtering for INDELs

Erik Garrison
SeqShop @ University of Michigan

Overview

1. **Genesis of insertion/deletion (indel) polymorphism**
2. Standard approaches to detecting indels
3. Assembly-based indel detection
4. Haplotype-based indel detection
5. Primary filtering: Bayesian variant calling
6. Post-call filtering: SVM
7. Graph-based resequencing approaches

An INDEL

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

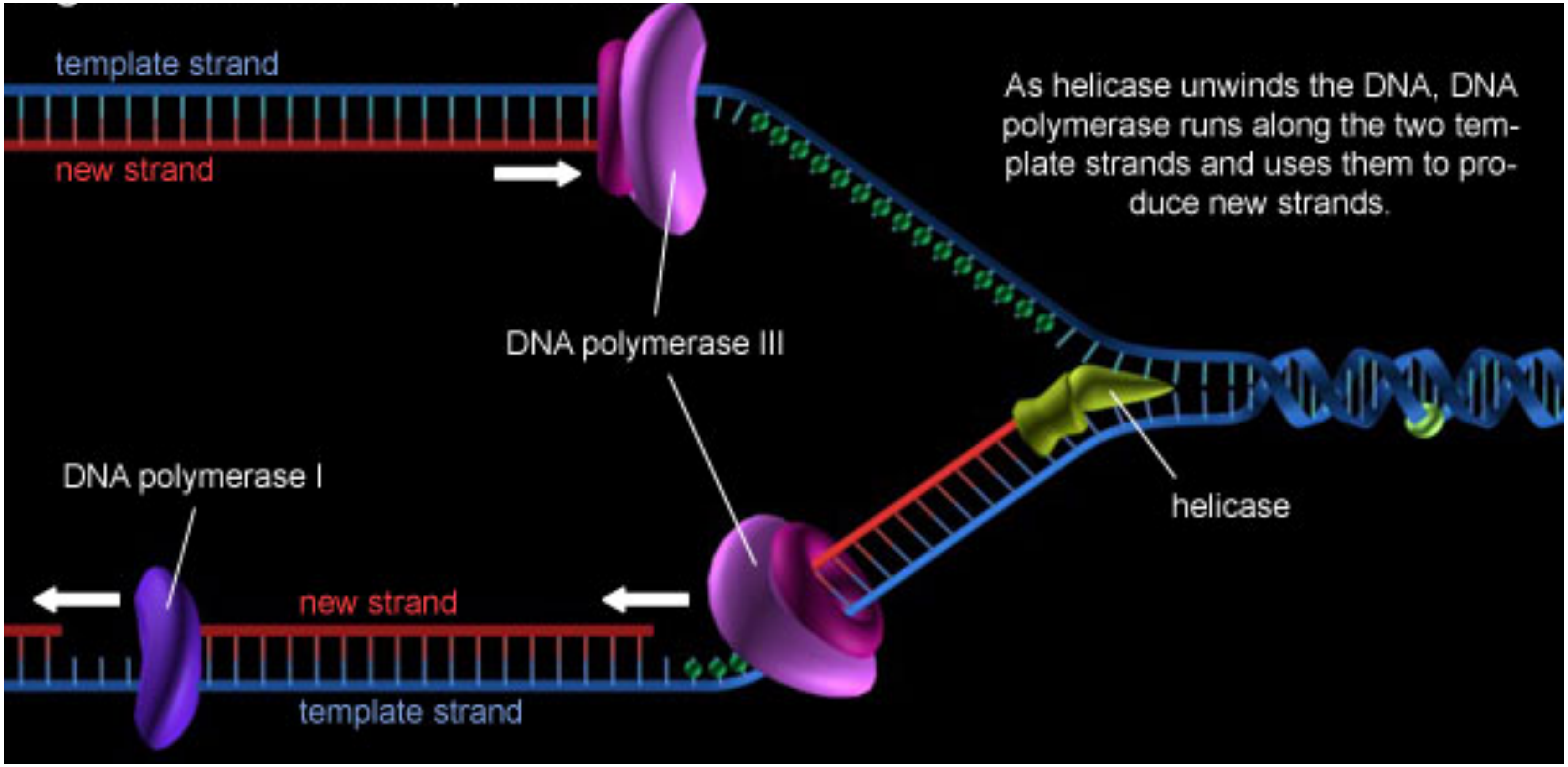
```
AATTAGCCATTA
AATTA--CATTA
```

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)

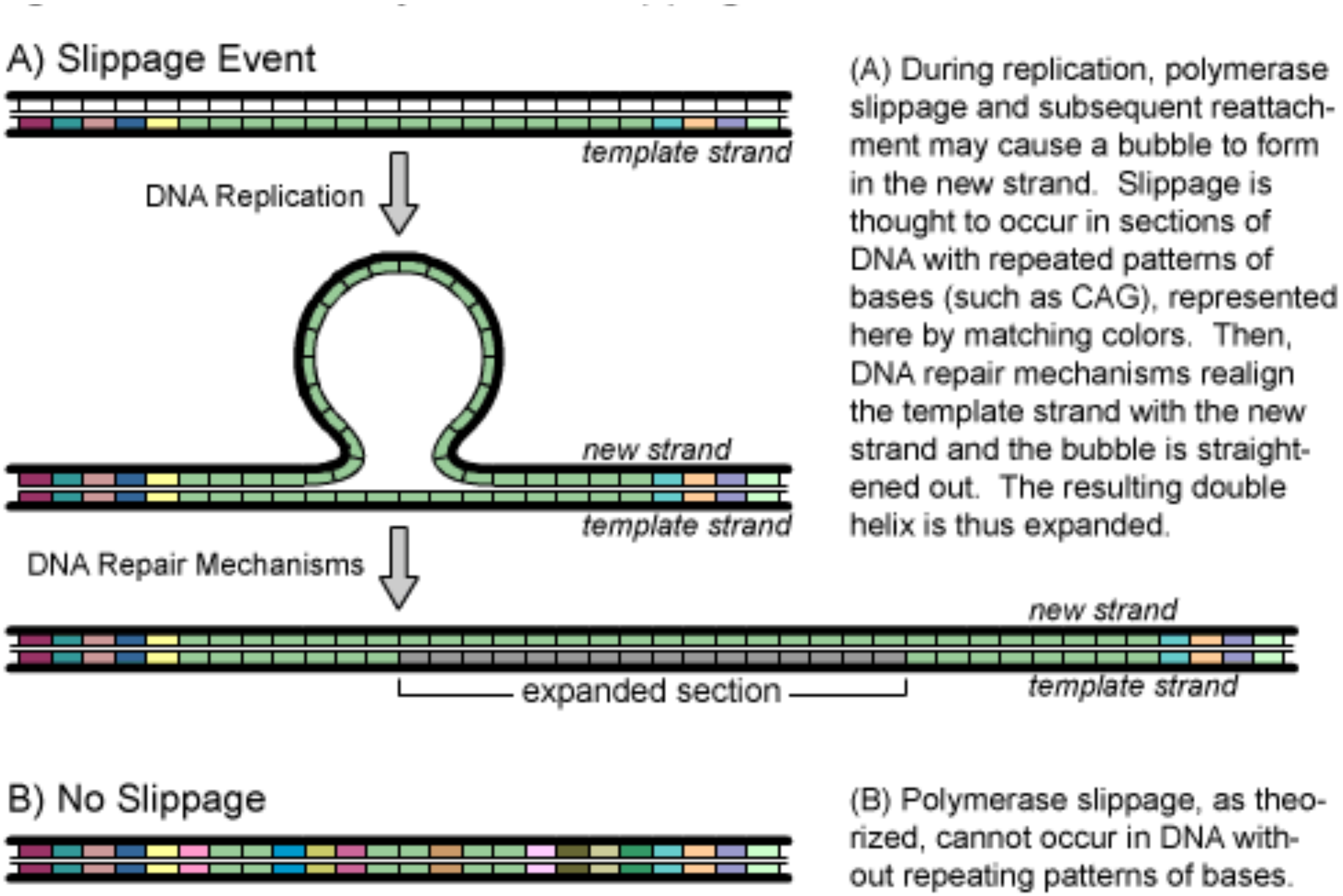
DNA replication



As helicase unwinds the DNA, DNA polymerase runs along the two template strands and uses them to produce new strands.

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

Polymerase *slippage*

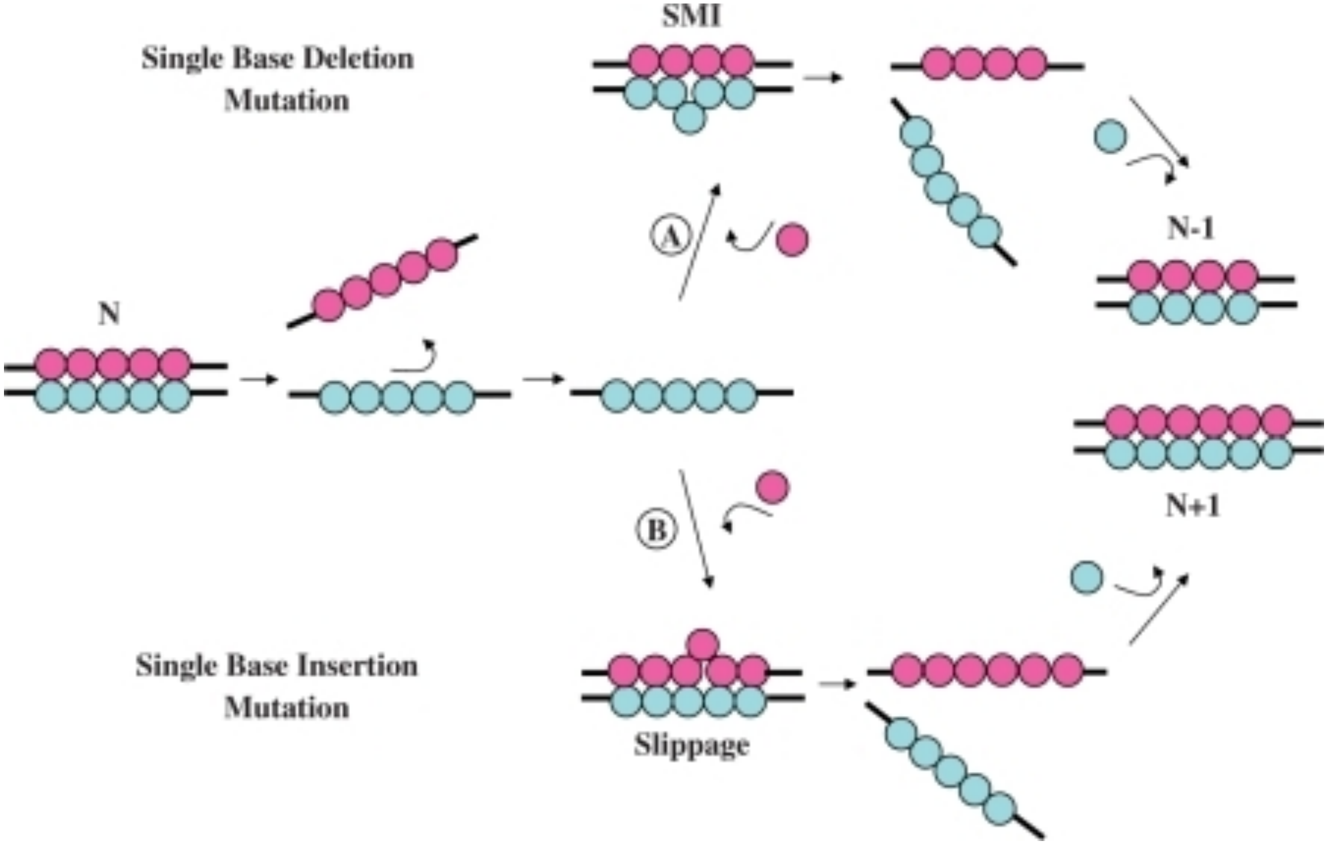


(A) During replication, polymerase slippage and subsequent reattachment may cause a bubble to form in the new strand. Slippage is thought to occur in sections of DNA with repeated patterns of bases (such as CAG), represented here by matching colors. Then, DNA repair mechanisms realign the template strand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

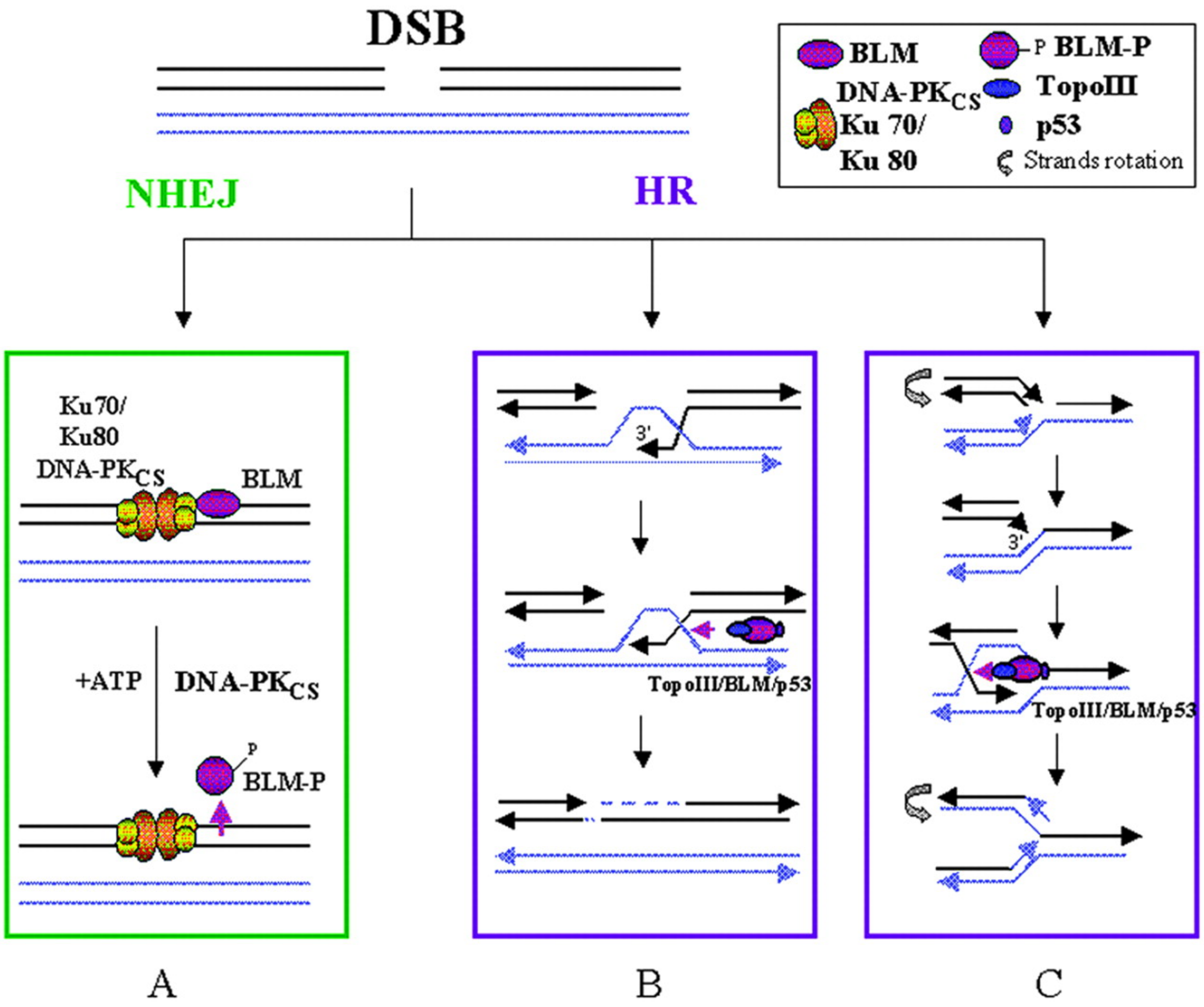
http://www.stanford.edu/group/hopes/cgi-bin/wordpress/2011/02/all-about-mutations/

Insertions and deletions via slippage



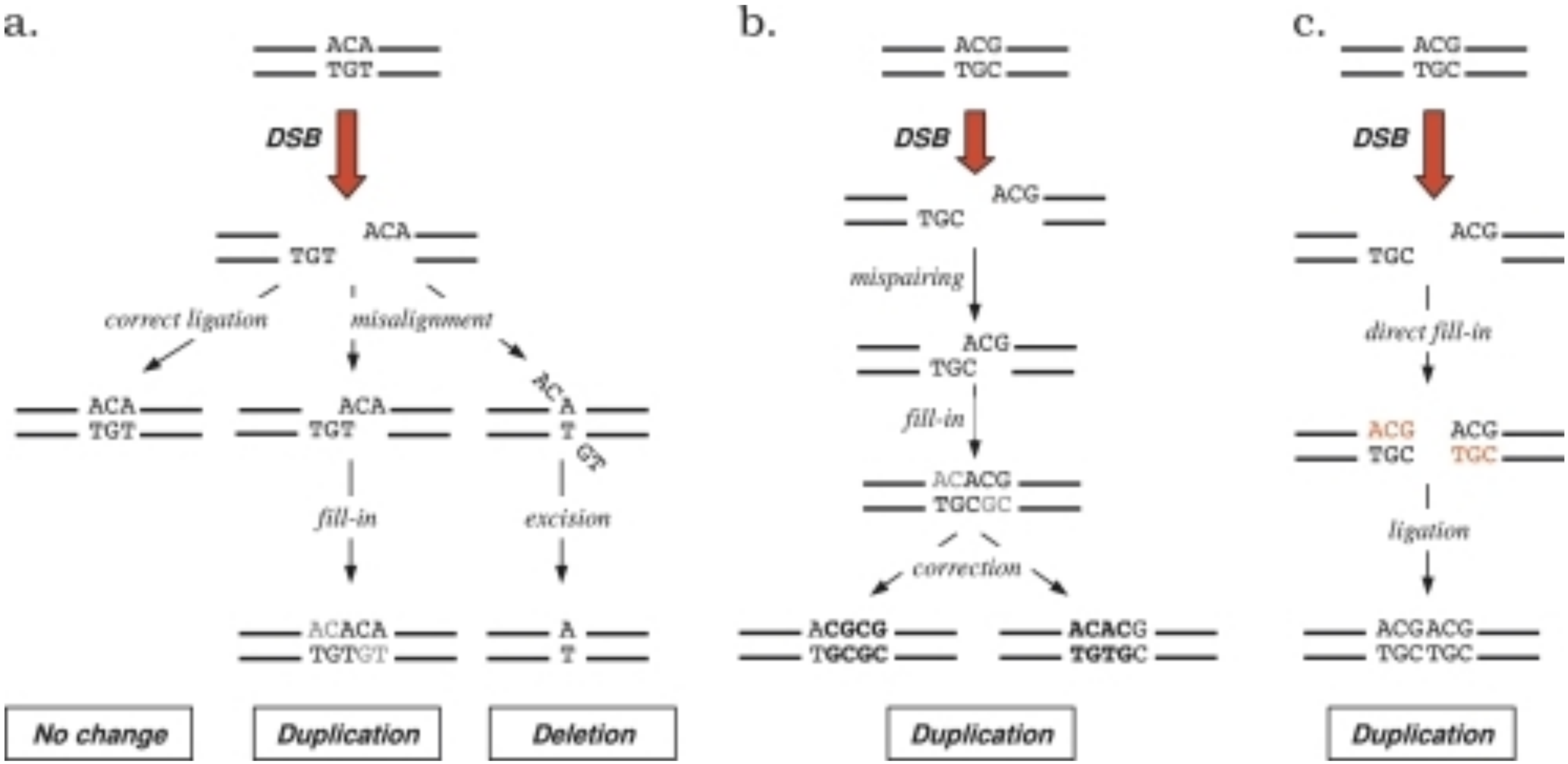
Energetic signatures of single base bulges: thermodynamic consequences and biological implications. Minetti CA, Remeta DP, Dickstein R, Breslauer KJ - Nucleic Acids Res. (2009)

Double-stranded break repair



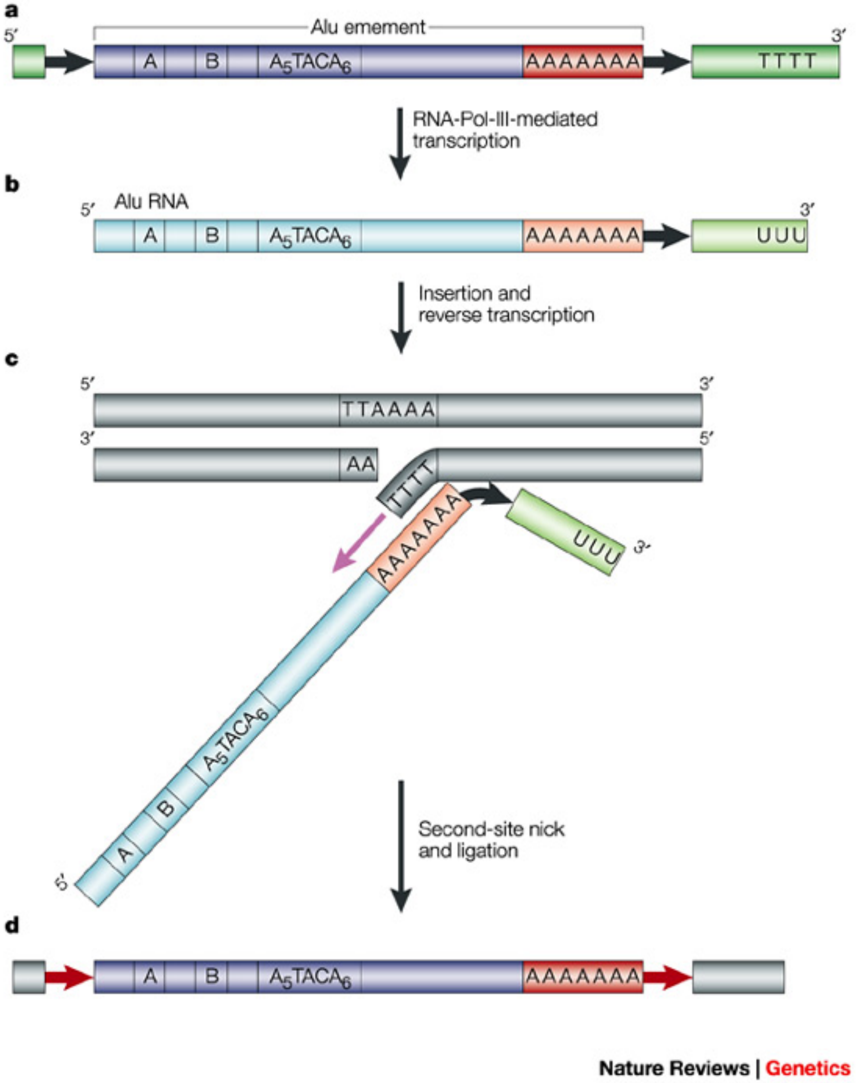
Possible anti-recombinogenic role of Bloom's syndrome helicase in double-strand break processing. doi: 10.1093/nar/gkg834

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)



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 insertion/deletion (indel) polymorphism
2. **Standard approaches to detecting indels**
3. Assembly-based indel detection
4. Haplotype-based indel detection
5. Primary filtering: Bayesian variant calling
6. Post-call filtering: SVM
7. Graph-based resequencing approaches

Calling INDEL variation

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

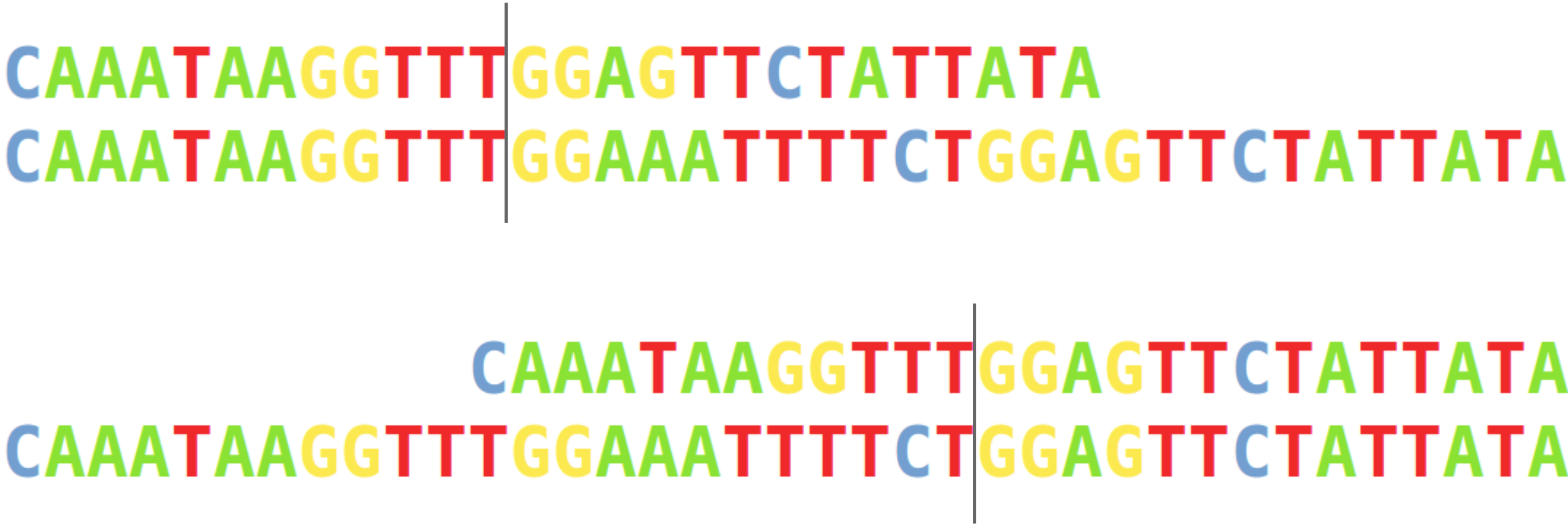
What are the steps you'd do to find the indel between these two sequences?

CAAATAAGGTTTGGAGTTCTATTATA

CAAATAAGGTTTGGAAATTTTCTGGAGTTCTATTATA

Indel finder

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



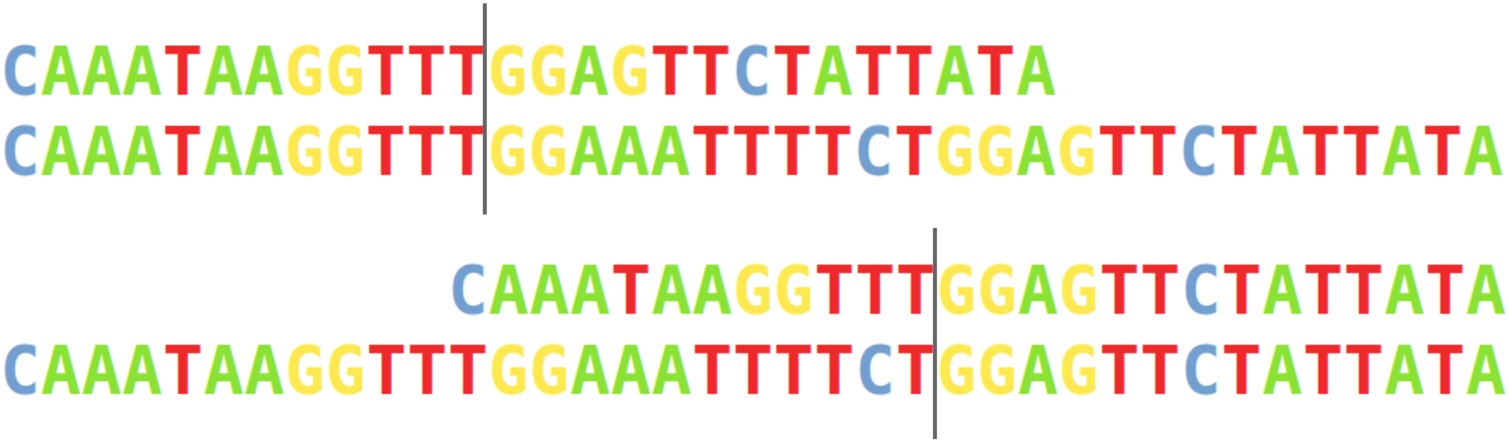
```
CAAATAAGGTTTGGAGTTCTATTATA
CAAATAAGGTTTGGAAATTTTCTGGAGTTCTATTATA

           CAAATAAGGTTTGGAGTTCTATTATA
CAAATAAGGTTTGGAAATTTTCTGGAGTTCTATTATA
```

Indel finder

We can see this more easily like this:

CAAATAAGGTTTGGAGTTCTATTATA

CAAATAAGGTTT*GGAAATTTTCT*GGAGTTCTATTATA



```
CAAATAAGGTTT|GGAGTTCTATTATA
CAAATAAGGTTT|GGAAATTTTCTGGAGTTCTATTATA

           CAAATAAGGTTT|GGAGTTCTATTATA
CAAATAAGGTTTGGAAATTTTCT|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

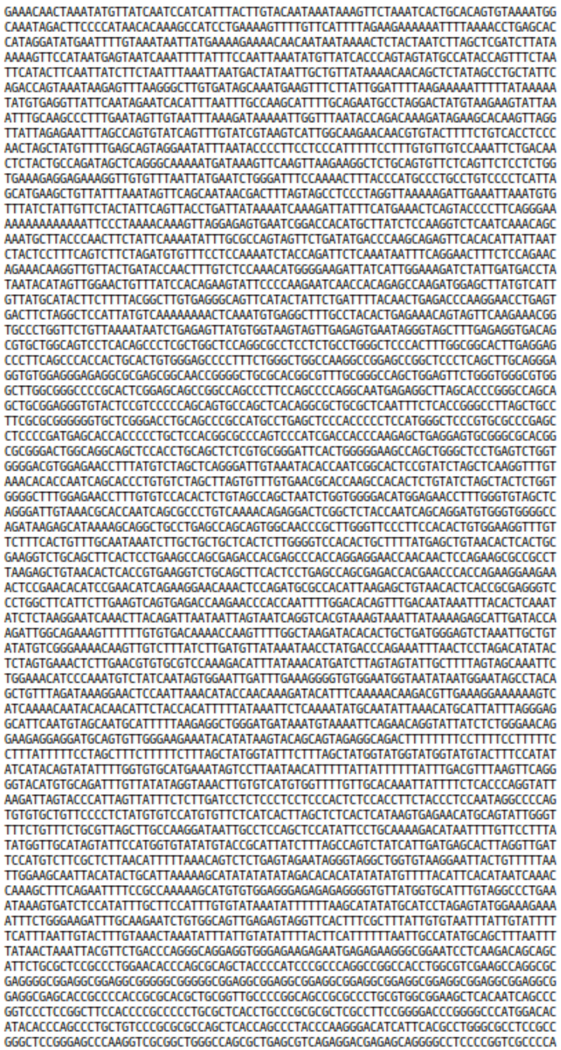
Different parameterizations can yield different results.

Different results suggest “different” variation.

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

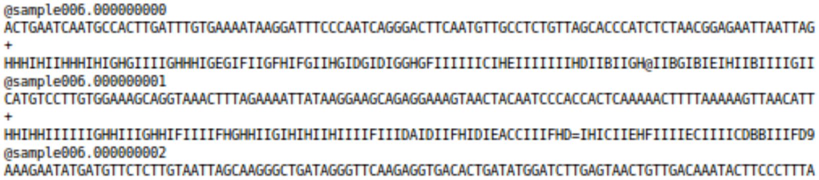
First, let’s review standard calling approaches.

Standard variant calling approach



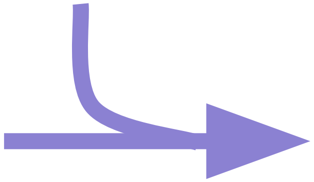
```
GAAACAACTAAATATGTTATCAATCCATCATTTACTTGTACAATAAATAAAGTTCTAAATCACTGCACAGTGTAAAATGG
CAAATAGACTTCCCCATAACACAAAGCCATCCTGAAAAGTTTTGTTCATTTTAGAAGAAAAAATTTTAAAACCTGAGCAC
CATAGGATATGAATTTTGTAAATAATTATGAAAAGAAAACAACAATAATAAAACTCTACTAATCTTAGCTCGATCTTATA
AAAAGTTCCATAATGAGTAATCAAATTTTATTTCCAATTAAATATGTTATCACCCAGTAGTATGCCATACCAGTTTCTAA
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
AGATTGGCAGAAAGTTTTTTGTGTGACAAAACCAAGTTTTGGCTAAGATACACACTGCTGATGGGAGTCTAAATTGCTGT
ATATGTCGGGAAAACAAGTTGTCTTTATCTTGATGTTATAAATAACCTATGACCCAGAAATTTAACTCCTAGACATATAC
TCTAGTGAAACTCTTGAACGTGTGCGTCCAAAGACATTTATAAACATGATCTTAGTAGTATTGCTTTTAGTAGCAAATTC
TGGAAACATCCCAAATGTCTATCAATAGTGGAATTGATTTGAAAGGGGTGTGGAATGGTAATATAATGGAATAGCCTACA
GCTGTTTAGATAAAGGAACTCCAATTAAACATACCAACAAAGATACATTTCAAAAACAAGACGTTGAAAGGAAAAAAGTC
ATCAAAACAATACACAACATTCTACCACATTTTTATAAATTCTCAAAATATGCAATATTAAACATGCATTATTTAGGGAG
GCATTCAATGTAGCAATGCATTTTTAAGAGGCTGGGATGATAAATGTAAAATTCAGAACAGGTATTATCTCTGGGAACAG
GAAGAGGAGGATGCAGTGTTGGGAAGAAATACATATAAGTACAGCAGTAGAGGCAGACTTTTTTTTCCTTTTCCTTTTTC
CTTTATTTTTCCTAGCTTTCTTTTTCTTTAGCTATGGTATTTCTTTAGCTATGGTATGGTATGGTATGTACTTTCCATAT
ATCATACAGTATATTTTGGTGTGCATGAAATAGTCCTTAATAACATTTTTATTATTTTTTATTTGACGTTTAAGTTCAGG
GGTACATGTGCAGATTTGTTATATAGGTAAACTTGTGTCATGTGGTTTTGTTGCACAAATTATTTTCTCACCCAGGTATT
AAGATTAGTACCCATTAGTTATTTCTCTTGATCCTCTCCCTCCTCCCACTCTCCACCTTCTACCCTCCAATAGGCCCCAG
TGTGTGCTGTTCCCCTCTATGTGTCCATGTGTTCTCATCACTTAGCTCTCACTCATAAGTGAGAACATGCAGTATTGGGT
TTTCTGTTTCTGCGTTAGCTTGCCAAGGATAATTGCCTCCAGCTCCATATTCCTGCAAAAGACATAATTTTGTTCCTTTA
TATGGTTGCATAGTATTCCATGGTGTATATGTACCGCATTATCTTTAGCCAGTCTATCATTGATGAGCACTTAGGTTGAT
TCCATGTCTTCGCTCTTAACATTTTTAAACAGTCTCTGAGTAGAATAGGGTAGGCTGGTGTAAGGAATTACTGTTTTTAA
TTGGAAGCAATTACATACTGCATTAAAAAGCATATATATATAGACACACATATATATGTTTTACATTCACATAATCAAAC
CAAAGCTTTCAGAATTTTCCGCCAAAAAGCATGTGTGGAGGGAGAGAGAGGGGTGTTATGGTGCATTTGTAGGCCCTGAA
ATAAAGTGATCTCCATATTTGCTTCCATTTGTGTATAAATATTTTTTAAGCATATATGCATCCTAGAGTATGGAAAGAAA
ATTTCTGGGAAGATTTGCAAGAATCTGTGGCAGTTGAGAGTAGGTTCACTTTCGCTTTATTGTGTAATTTATTGTATTTT
TCATTTAATTGTACTTTGTAAACTAAATATTTATTGTATATTTTACTTCATTTTTTAATTGCCATATGCAGCTTTAATTT
TATAACTAAATTACGTTCTGACCCAGGGCAGGAGGTGGGAGAAGAGAATGAGAGAAGGGCGGAATCCTCAAGACAGCAGC
ATTCTGCGCTCCGCCCTGGAACACCCAGCGCAGCTACCCCATCCCGCCCAGGCCGGCCACCTGGCGTCGAAGCCAGGCGC
GAGGGGCGGAGGCGGAGGCGGGGGCGGGGGCGGAGGCGGAGGCGGAGGCGGAGGCGGAGGCGGAGGCGGAGGCGGAGGCG
GAGGCGAGCACCGCCCCACCGCGCACGCTGCGGTTGCCCCGGCAGCCGCGCCCTGCGTGGCGGAAGCTCACAATCAGCCC
GGTCCCTCCGGCTTCCACCCCGCCCCCTGCGCTCACCTGCCCGCGCGCTCGCCTTCCGGGGACCCGGGGCCCATGGACAC
ATACACCCAGCCCTGCTGTCCCGCGCGCCAGCTCACCAGCCCTACCCAAGGACATCATTCACGCCTGGGCGCCTCCGCC
GGGCTCCGGGAGCCCAAGGTCGCGGCTGGGCCAGCGCTGAGCGTCAGAGGACGAGAGCAGGGGCCTCCCCGGTCGCCCCA
```

Genome (FASTA)

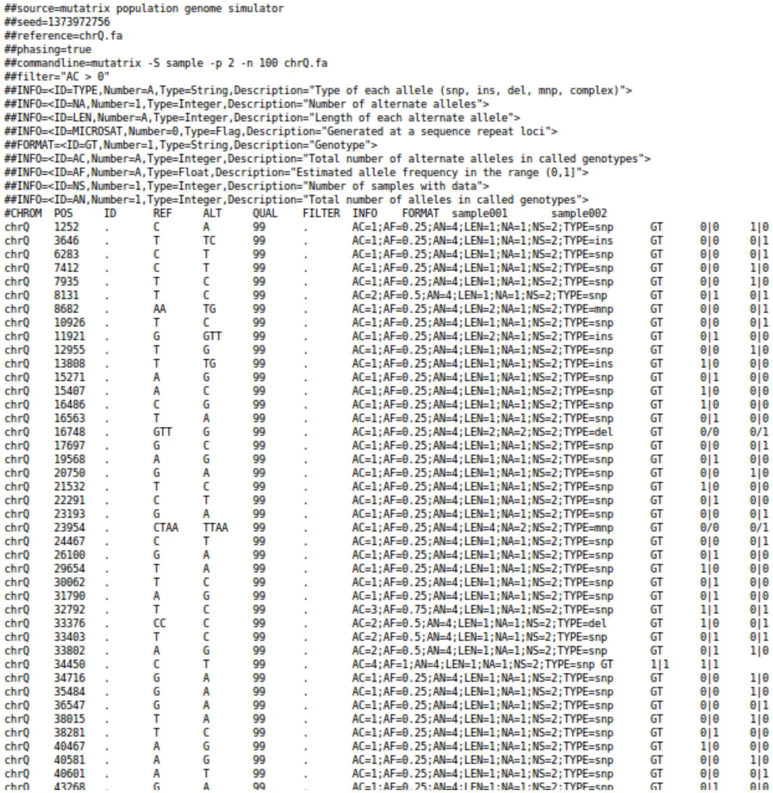


```
@sample006.000000000
ACTGAATCAATGCCACTTGATTTGTGAAAATAAGGATTTCCCAATCAGGGACTTCAATGTTGCCTCTGTTAGCACCCATCTCTAACGGAGAATTAATTAG
+
HHHIHIIHHHIHIGHGIIIIGHHHIGEGIFIIGFHIFGIIHGIDGIDIGGHGFIIIIIICIHEIIIIIIIHDIIBIIGH@IIBGIBIEIHIIBIIIIGII
@sample006.000000001
CATGTCCTTGTGGAAAGCAGGTAAACTTTAGAAAATTATAAGGAAGCAGAGGAAAGTAACTACAATCCCACCACTCAAAAACTTTTAAAAAGTTAACATT
+
HHIHHIIIIIIGHHIIIGHHIFIIIIFHGHHIIGIHIHIIHIIIIFIIIDAIDIIFHIDIEACCIIIFHD=IHICIIEHFIIIIECIIIICDBBIIIFD9
@sample006.000000002
AAAGAATATGATGTTCTCTTGTAATTAGCAAGGGCTGATAGGGTTCAAGAGGTGACACTGATATGGATCTTGAGTAACTGTTGACAAATACTTCCCTTTA
```

Reads
(FASTQ)



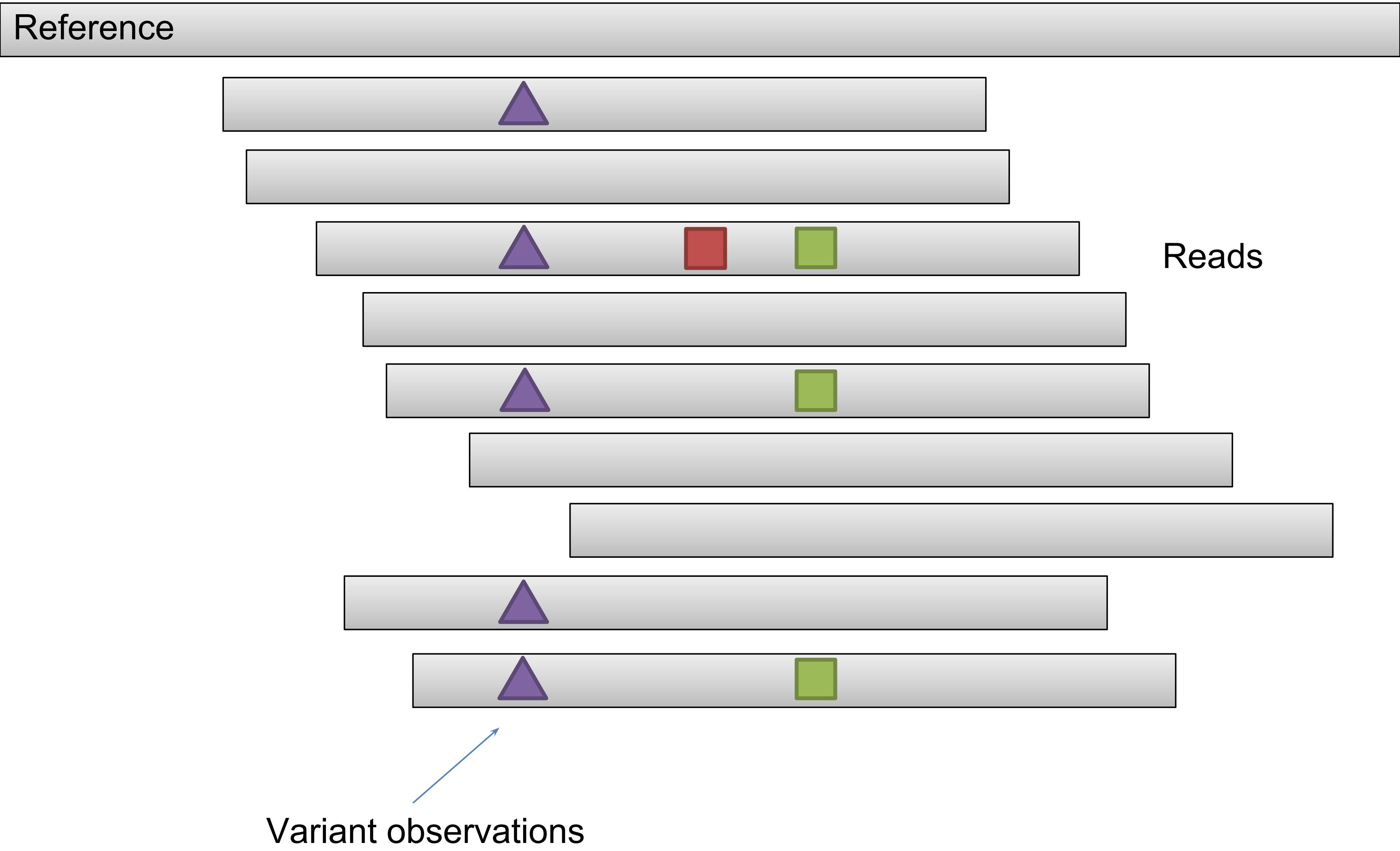
alignment and variant calling



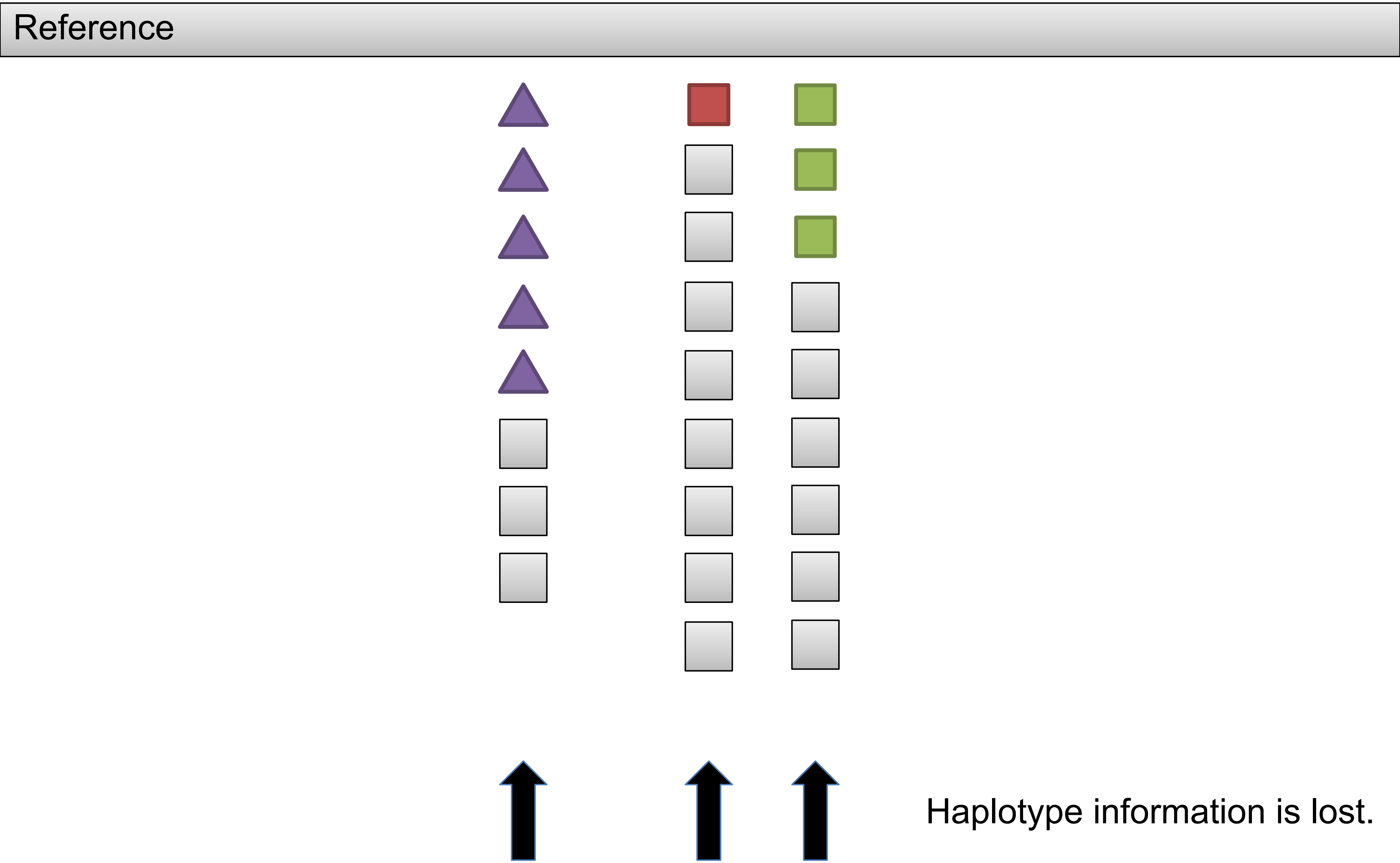
```
##source=mutatrix population genome simulator
##seed=1373972756
##reference=chrQ.fa
##phasing=true
##commandline=mutatrix -S sample -p 2 -n 100 chrQ.fa
##filter="AC > 0"
##INFO=<ID=TYPE,Number=A,Type=String,Description="Type of each allele (snp, ins, del, mnp, complex)">
##INFO=<ID=NA,Number=1,Type=Integer,Description="Number 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)

Alignments to candidates



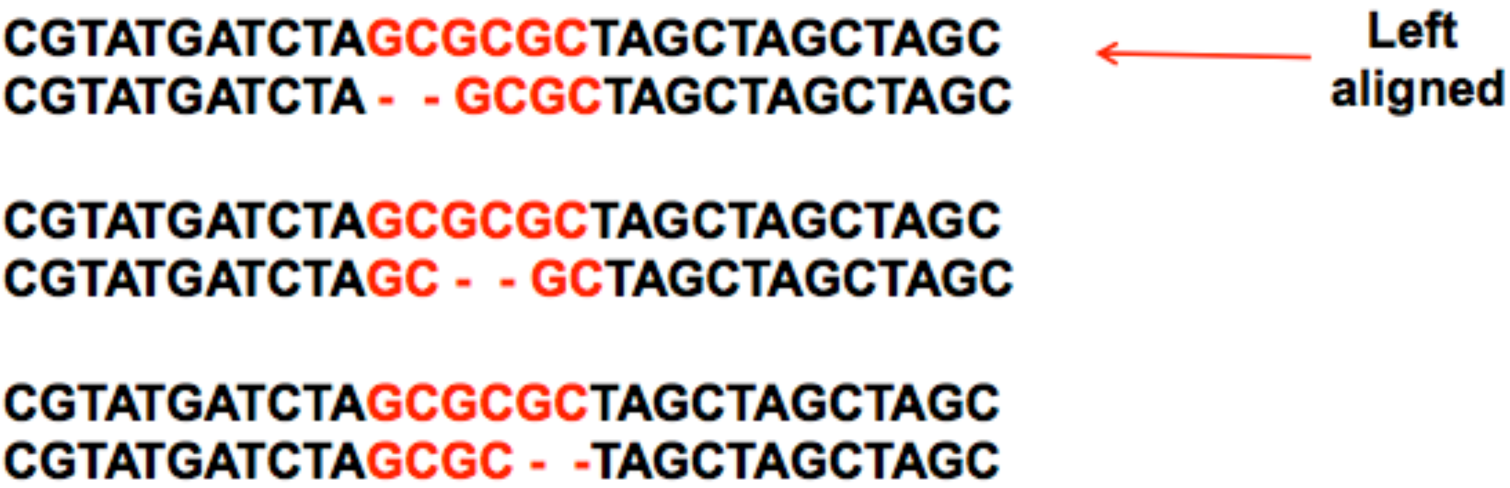
The data exposed to the caller



Haplotype information is lost.

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.



```
CGTATGATCTAGCGCGCTAGCTAGCTAGC
CGTATGATCTA -  - GCGCTAGCTAGCTAGC

CGTATGATCTAGCGCGCTAGCTAGCTAGC
CGTATGATCTAGC -  - GCTAGCTAGCTAGC

CGTATGATCTAGCGCGCTAGCTAGCTAGC
CGTATGATCTAGCGC -  -TAGCTAGCTAGC
```

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

example: 1000G PhaseI low coverage
chr15:81551110, ref:CTCTC alt:ATATA

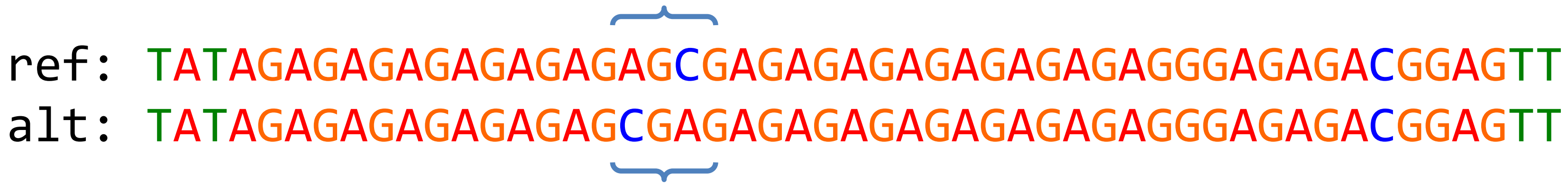
```
ref: TGTCACTCGCTCTCTCTCTCTCTCTCTATATATATATATTTGTGCAT
alt: TGTCACTCGCTCTCTCTCTCTATATATATATATATATATATTTGTGCAT
```

Interpreted as 3 SNPs

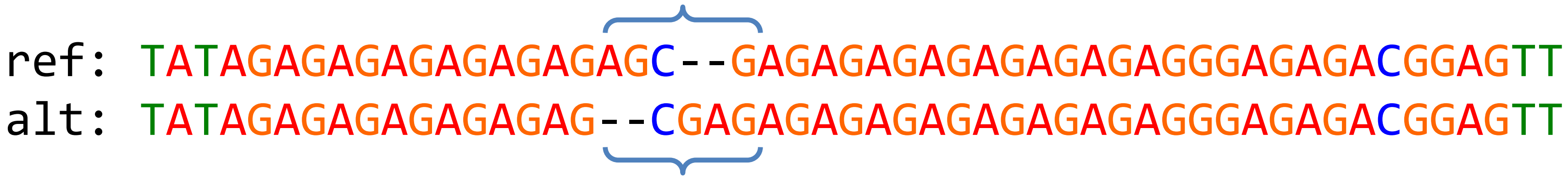
```
ref: TGTCACTCGCTCTCTCTCTCTCTCTCT------ATATATATATATTTGTGCAT
alt: TGTCACTCGCTCTCTCTCTCT------ATATATATATATATATATATTTGTGCAT
```

Interpreted as microsatellite expansion/contraction

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



```
ref: TATAGAGAGAGAGAGAGAGCGAGAGAGAGAGAGAGAGGGAGAGACGGAGTT
alt: TATAGAGAGAGAGAGAGCGAGAGAGAGAGAGAGAGAGGGAGAGACGGAGTT
```



```
ref: TATAGAGAGAGAGAGAGAGC--GAGAGAGAGAGAGAGAGGGAGAGACGGAGTT
alt: TATAGAGAGAGAGAGAG--CGAGAGAGAGAGAGAGAGAGGGAGAGACGGAGTT
```

Overview

1. Genesis of insertion/deletion (indel) polymorphism
2. Standard approaches to detecting indels
3. **Assembly-based indel detection**
4. Haplotype-based indel detection
5. Primary filtering: Bayesian variant calling
6. Post-call filtering: SVM
7. Graph-based resequencing approaches

Problem: inconsistent indel 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.

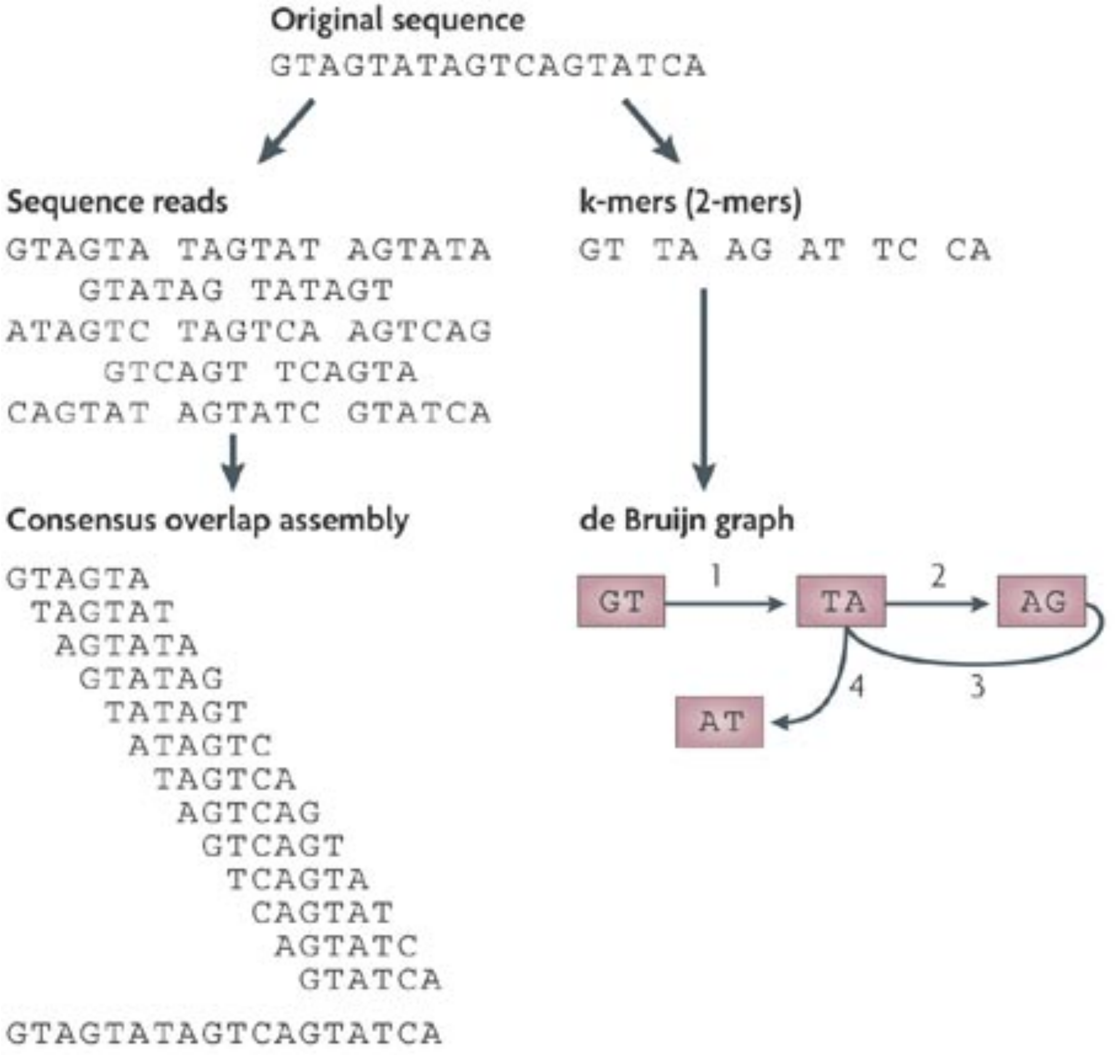
Basically, we use the alignments to localize reads, then process them again with assembly approaches to determine candidate alleles.

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)

Assembly



Original sequence

```
GTAGTATAGTCAGTATCA
```

Sequence reads

```
GTAGTA TAGTAT AGTATA
   GTATAG TATAGT
ATAGTC TAGTCA AGTCAG
    GTCAGT TCAGTA
CAGTAT AGTATC GTATCA
```

Consensus overlap assembly

```
GTAGTA
 TAGTAT
  AGTATA
   GTATAG
    TATAGT
     ATAGTC
      TAGTCA
       AGTCAG
        GTCAGT
         TCAGTA
          CAGTAT
           AGTATC
            GTATCA

GTAGTATAGTCAGTATCA
```

k-mers (2-mers)

```
GT TA AG AT TC CA
```

de Bruijn graph

Nature Reviews | Microbiology

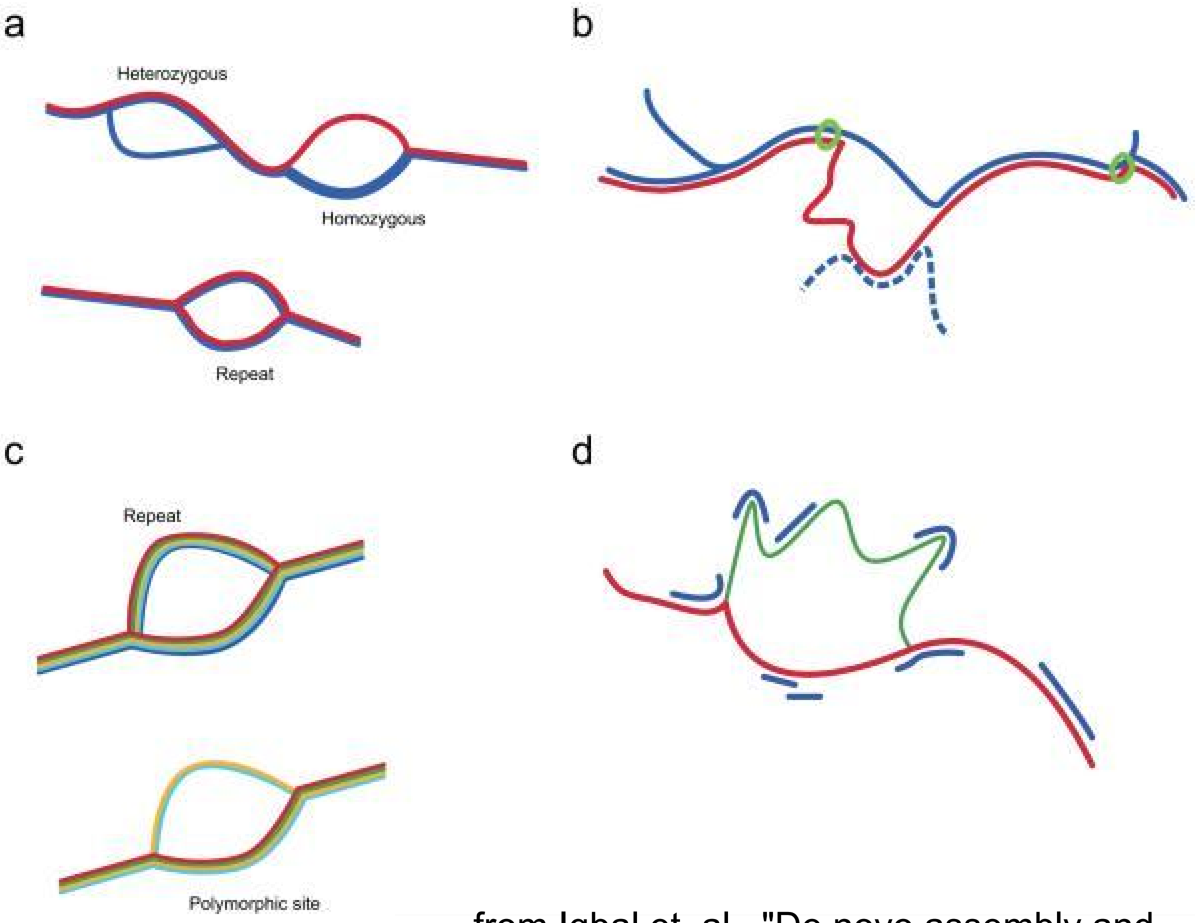
http://www.nature.com/nrmicro/journal/v7/n4/full/nrmicro2088.html

Using colored graphs (Cortex)

Variants can be called using bubbles in deBruijn graphs.

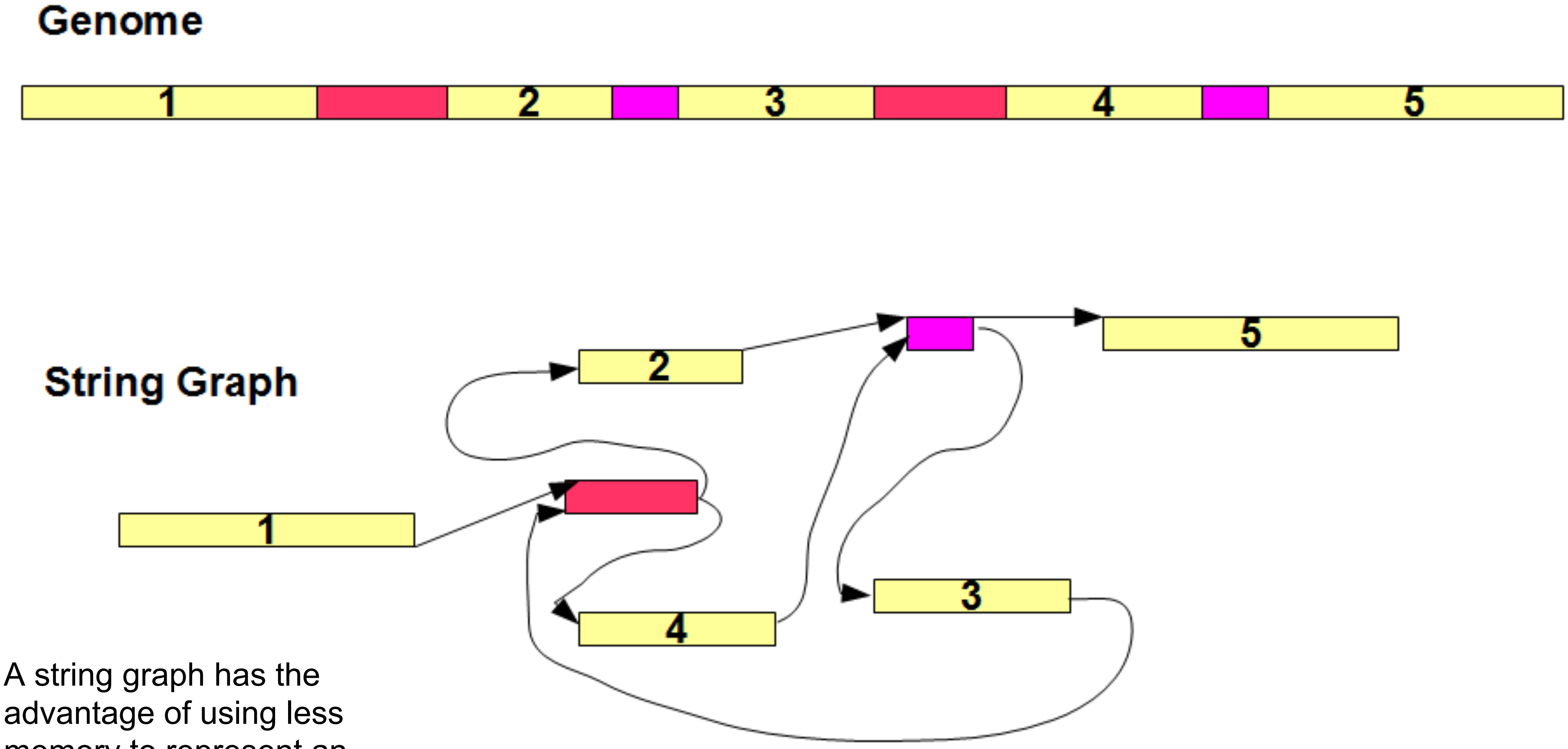
Method is completely reference-free, except for reporting of variants. The reference is threaded through the colored graph.

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)

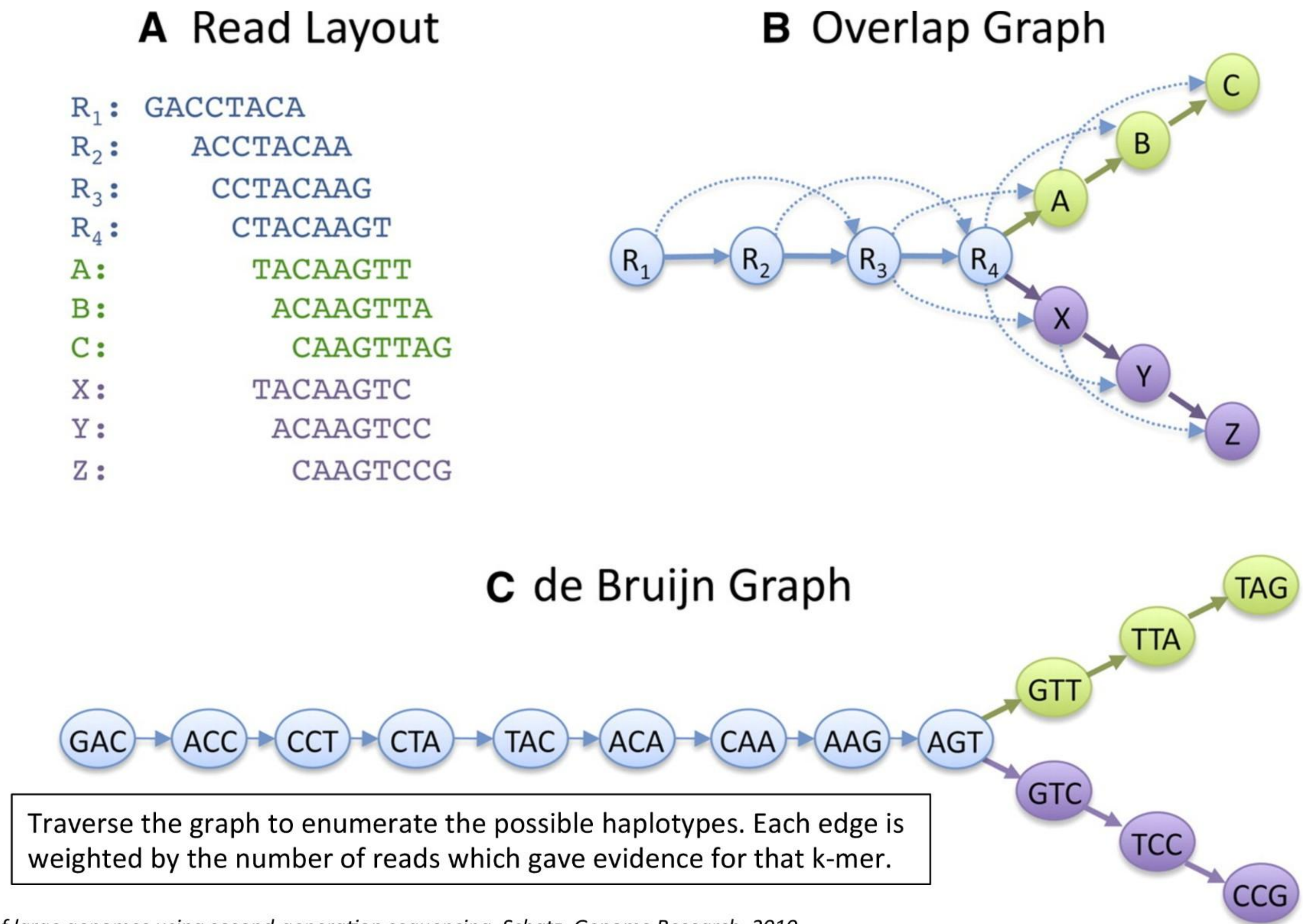
String graphs (SGA)



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)



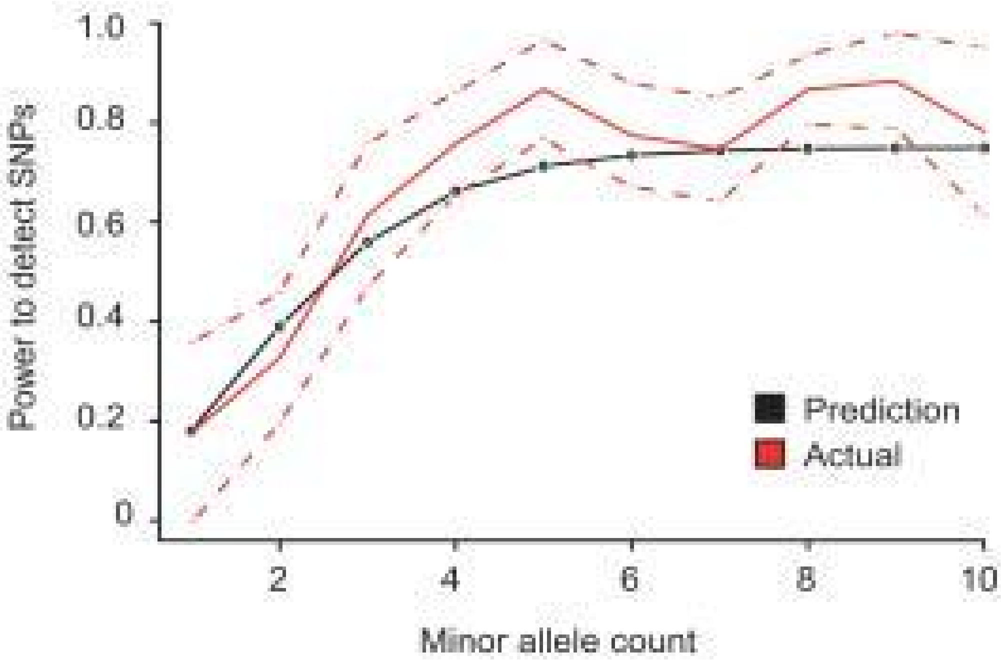
```
R1: GACCTACA
R2:   ACCTACAA
R3:    CCTACAAG
R4:     CTACAAGT
A:       TACAAGTT
B:        ACAAGTTA
C:         CAAGTTAG
X:       TACAAGTC
Y:        ACAAGTCC
Z:         CAAGTCCG
```

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.

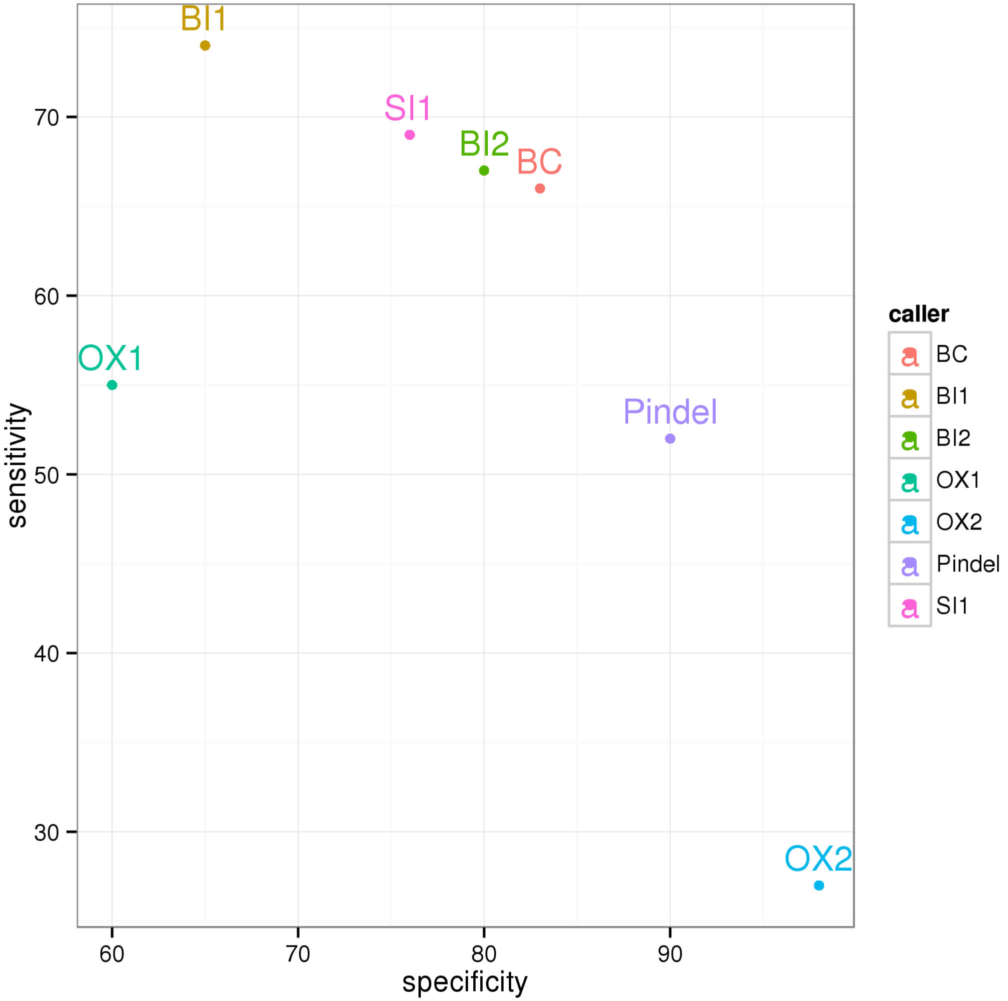
from Iqbal et. al., "De novo assembly and genotyping of variants using colored de Bruijn graphs." (2012)

Indel validation, 191 AFR samples

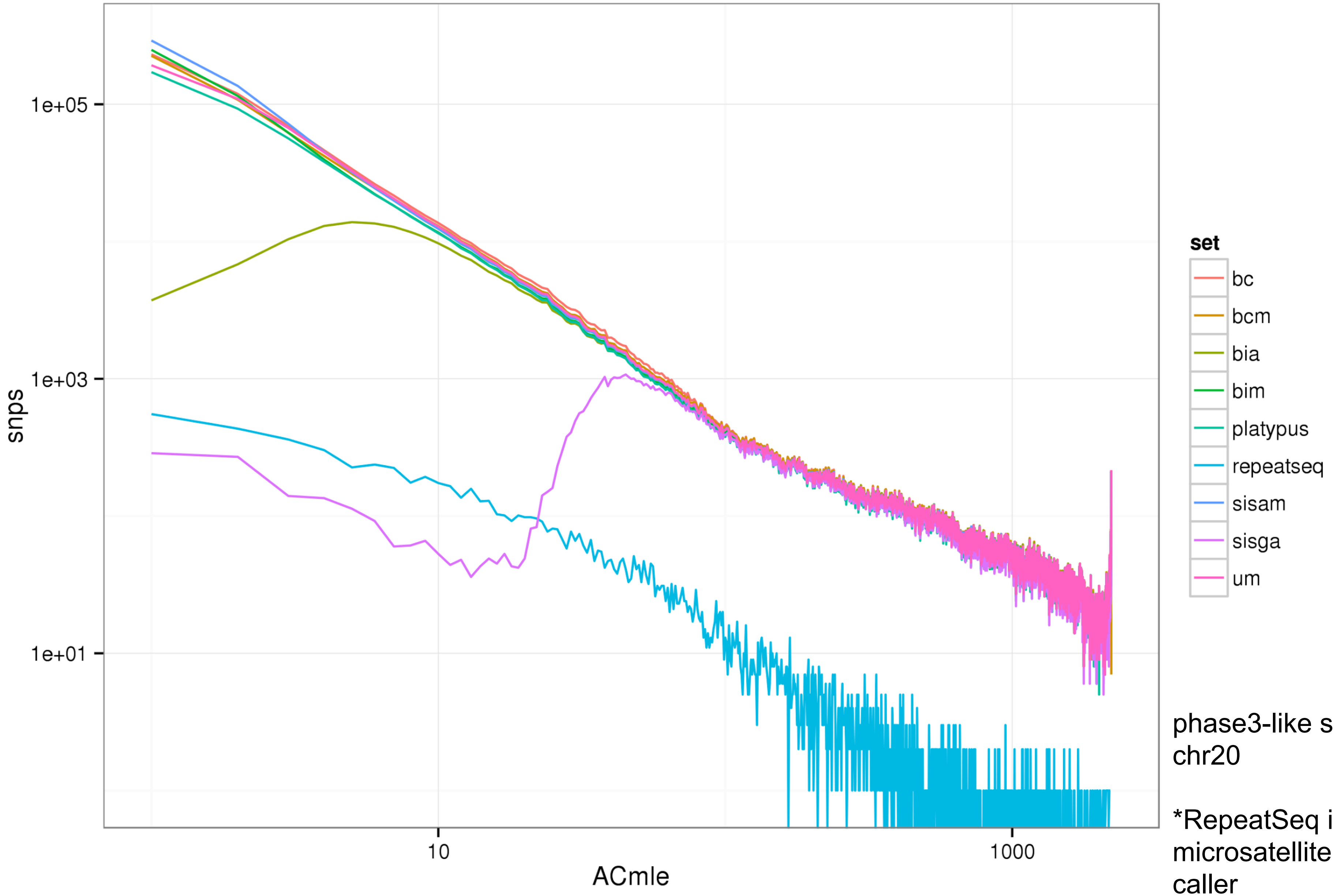
High-depth miSeq sequencing-based validation on 4 samples.

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.



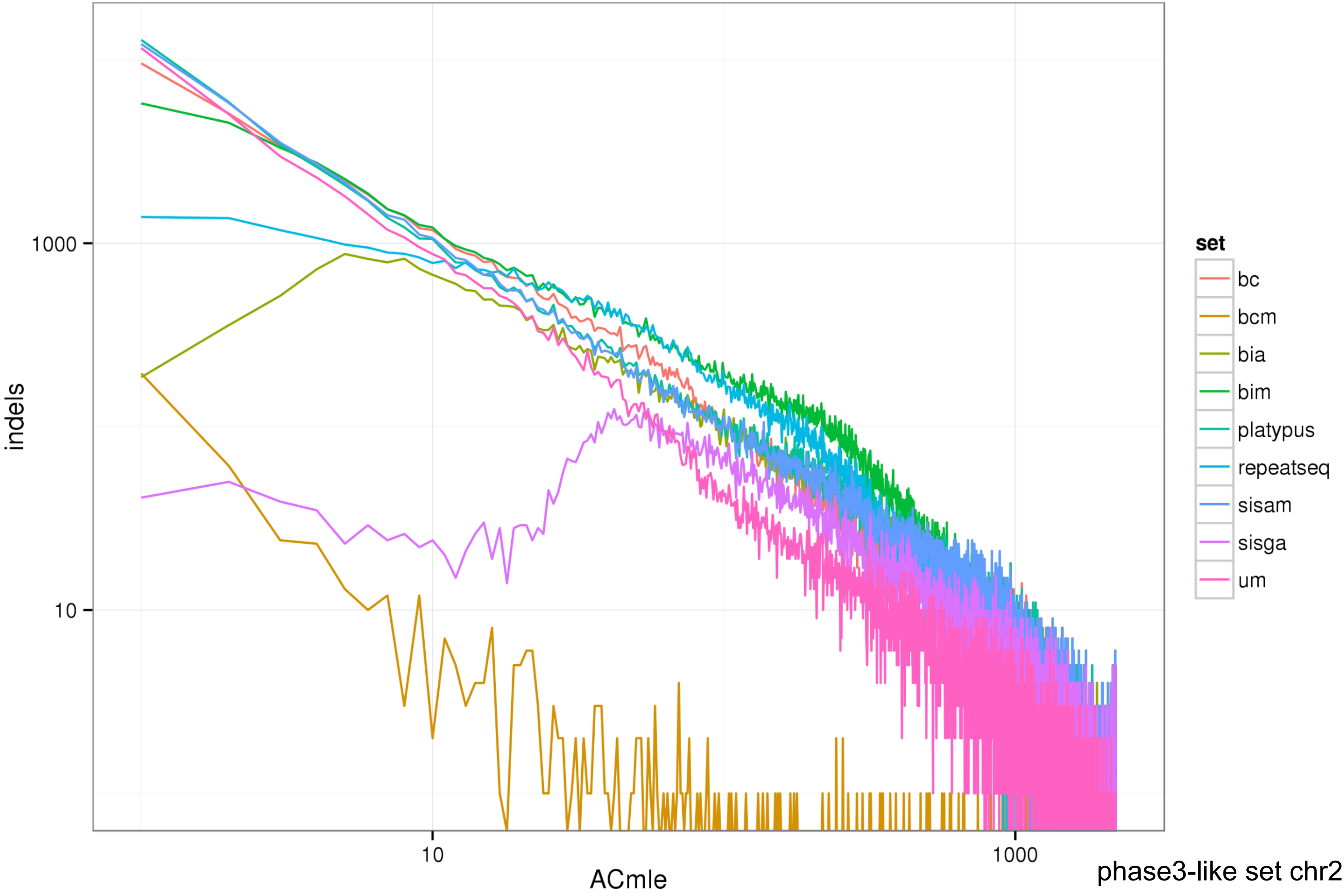
Site-frequency spectrum, SNPs



phase3-like set
chr20

*RepeatSeq is a microsatellite caller

Site-frequency spectrum, indels



phase3-like set chr20

Overview

1. Genesis of insertion/deletion (indel) polymorphism
2. Standard approaches to detecting indels
3. Assembly-based indel detection
4. **Haplotype-based indel detection**
5. Primary filtering: Bayesian variant calling
6. Post-call filtering: SVM
7. Graph-based resequencing approaches

Finding haplotype polymorphisms

Two reads

AGAACCCAGTGCTCTTTCTGCT

AGAACCCAGTGGTCTTTCTGCT

a SNP

```
           C
AGAACCCAGTG TCTTTCTGCT
           G
```

Their alignment

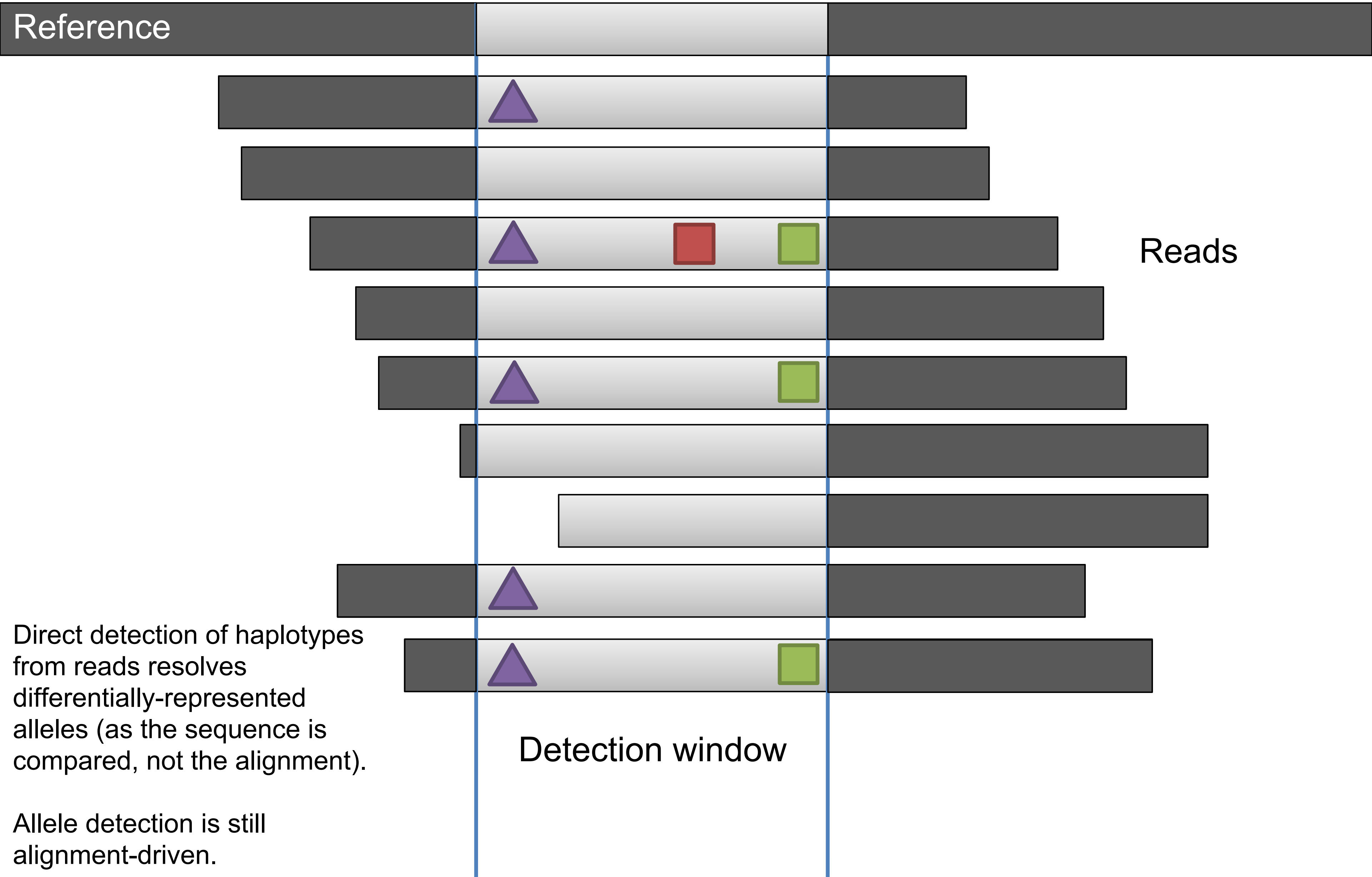
Another read showing a SNP on the same haplotype as the first

AGAACCCAGTGCTCTATCTGCT

```
           CTCTA
AGAACCCAGTG     TCTGCT
           GTCTT
```

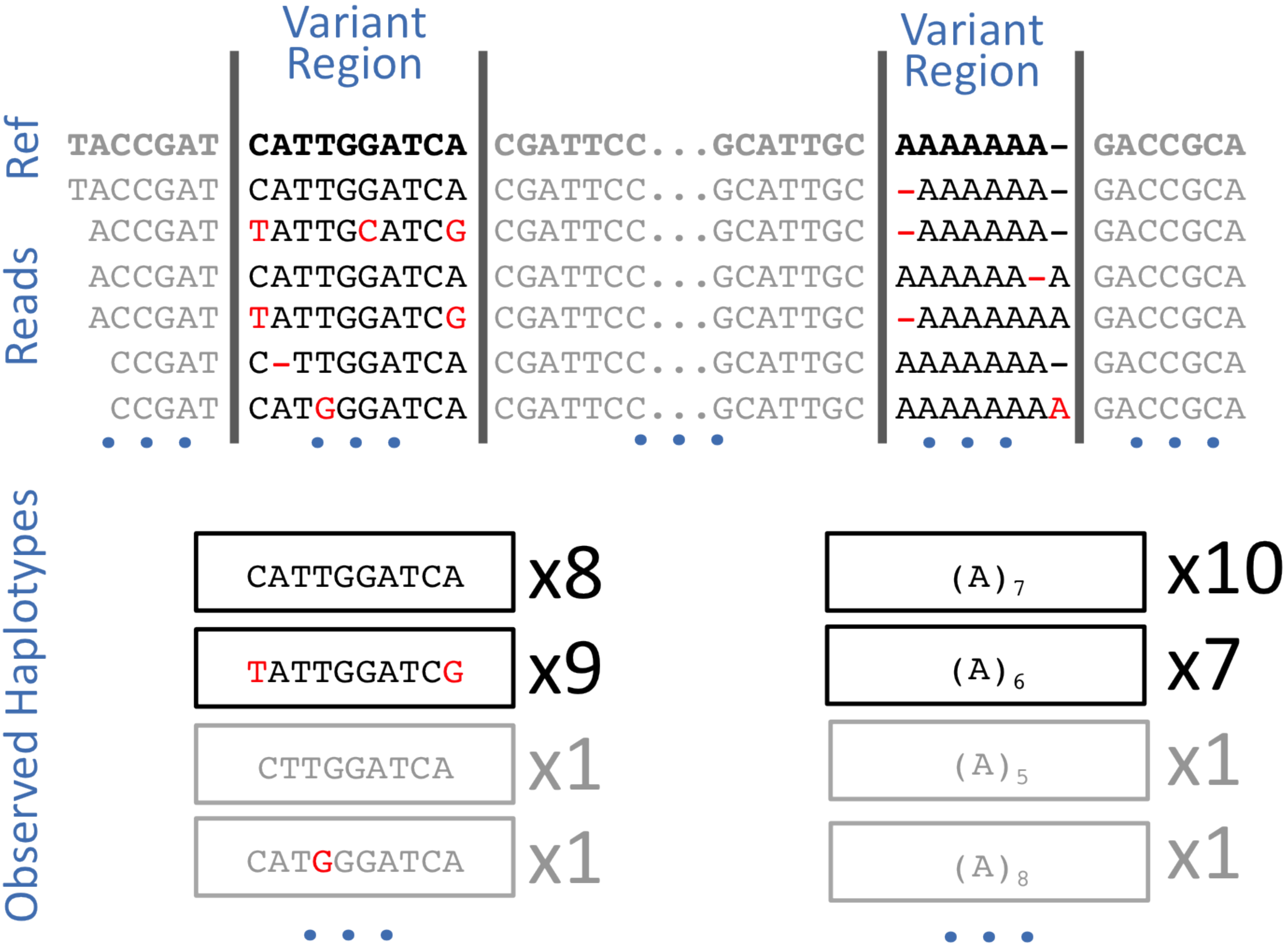
A variant locus implied by alignments

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

TACCGAT	CATTGGATCA	CGATTCC...GCATTGC	AAAAAAA-	GACCGCA

Reads

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
• • •	• • •	• • •	• • •	• • •

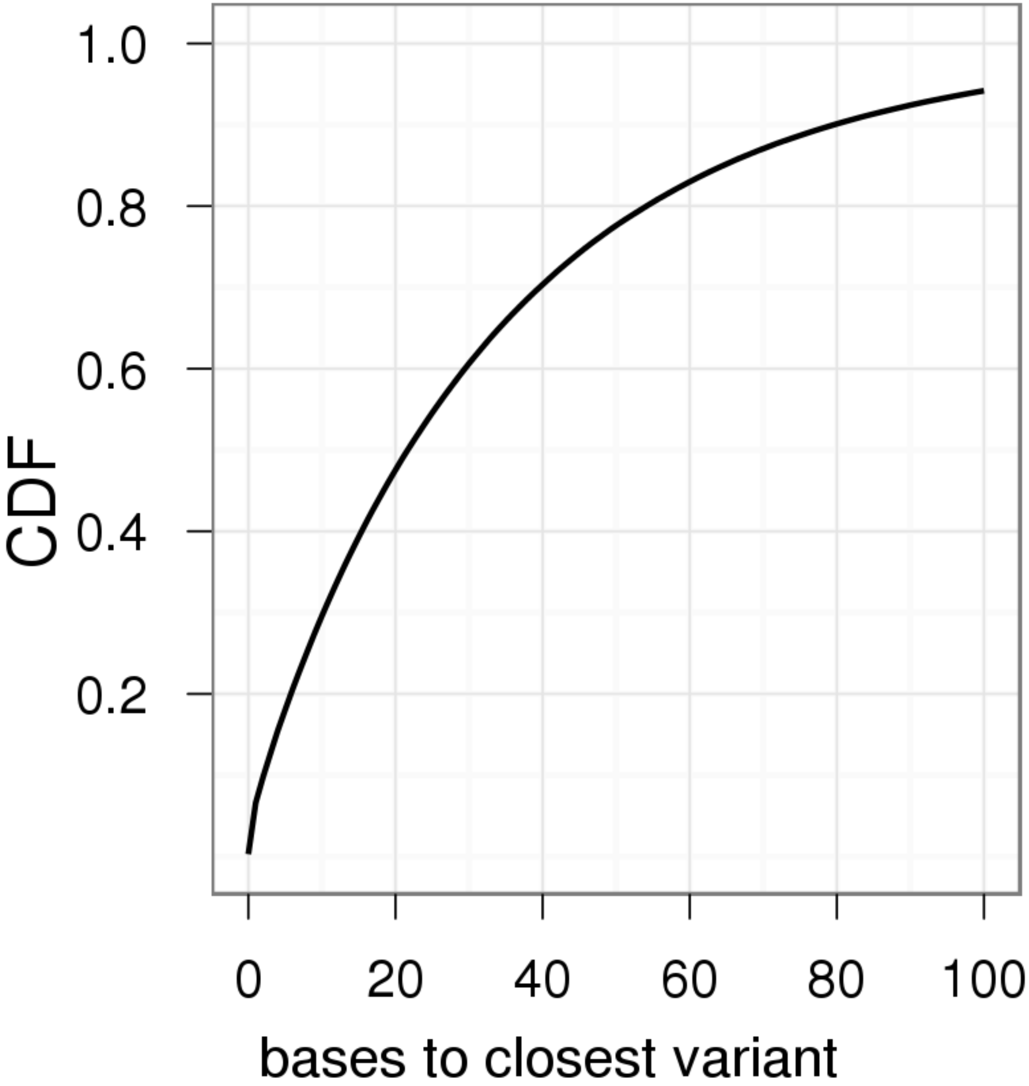
Observed Haplotypes

CATTGGATCA	x8	$(A)_7$	x10
TATTGGATCG	x9	$(A)_6$	x7
CTTGGATCA	x1	$(A)_5$	x1
CATGGGATCA	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
```

OTOF gene – mutations cause profound recessive deafness

```
apparent:    AGG TAG CTG
             Arg Ter ---
```

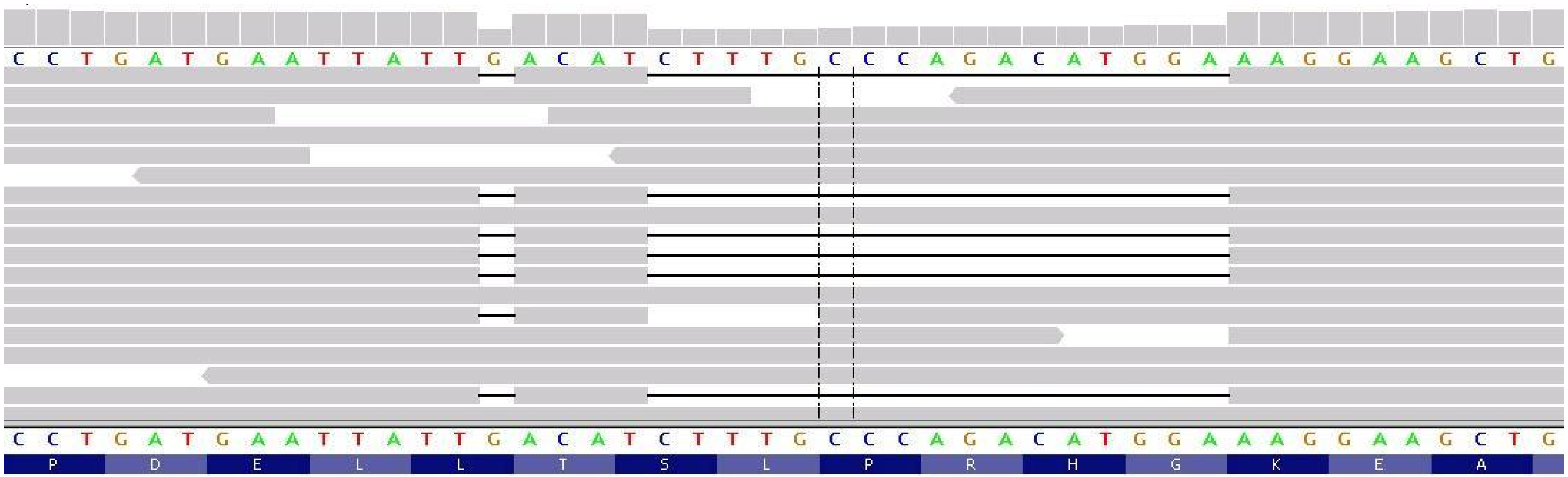
Apparent nonsense variant, one YRI homozygote

```
actual:      AGG TTG CTG
             Arg Leu Leu
```

Actually a block substitution that results in a missense substitution

(Daniel MacArthur)

Importance of haplotype effects: frame-restoring indels


C C T G A T G A A T T A T T G A C A T C T T T G C C C A G A C A T G G A A A G G A A G C T G
C C T G A T G A A T T A T T G A C A T C T T T G C C C A G A C A T G G A A A G G A A G C T G
P D E L L T S L P R H G K E A

- 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
```

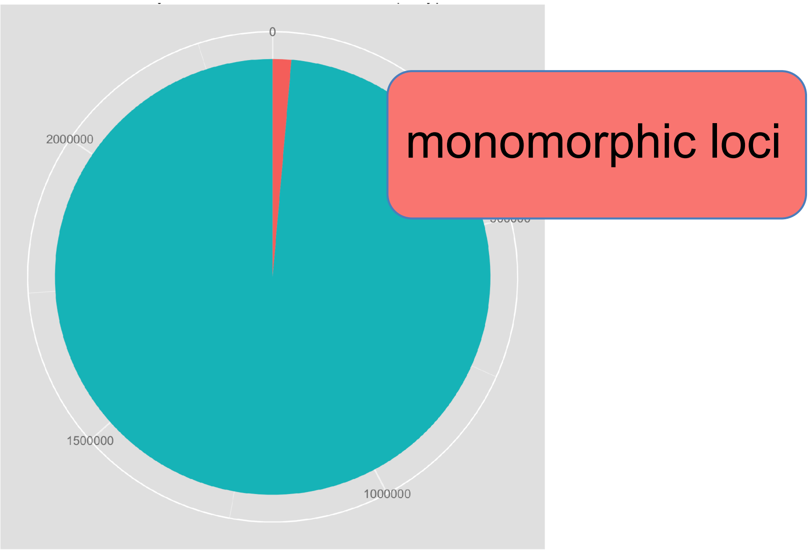
chr6:32551935, HLA-DRB1 **(~11% AF in 1000G)**

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

Impact on genotyping chip design

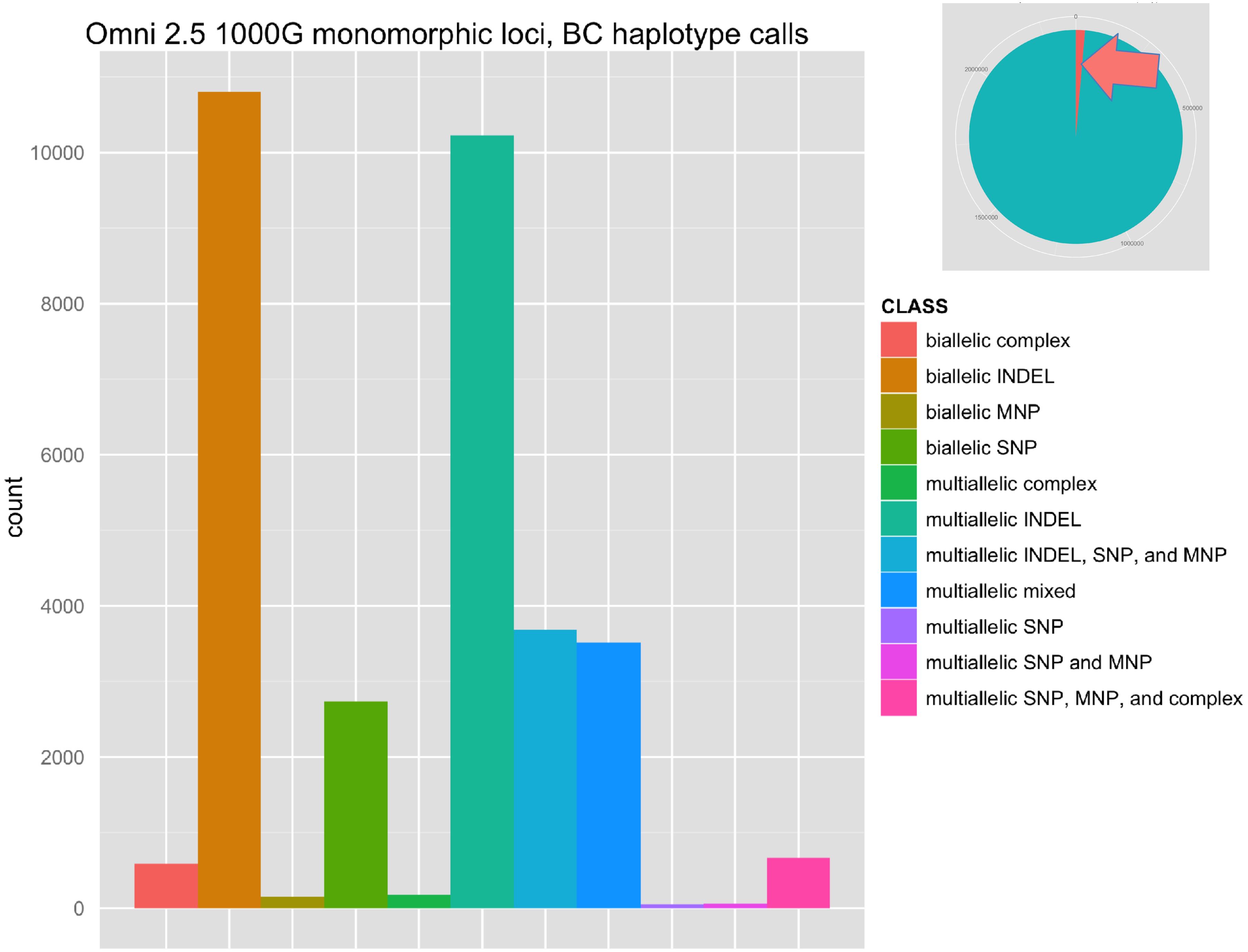
- Biallelic SNPs detected during the 1000 Genomes Pilot project were used to design a genotyping microarray (Omni 2.5).

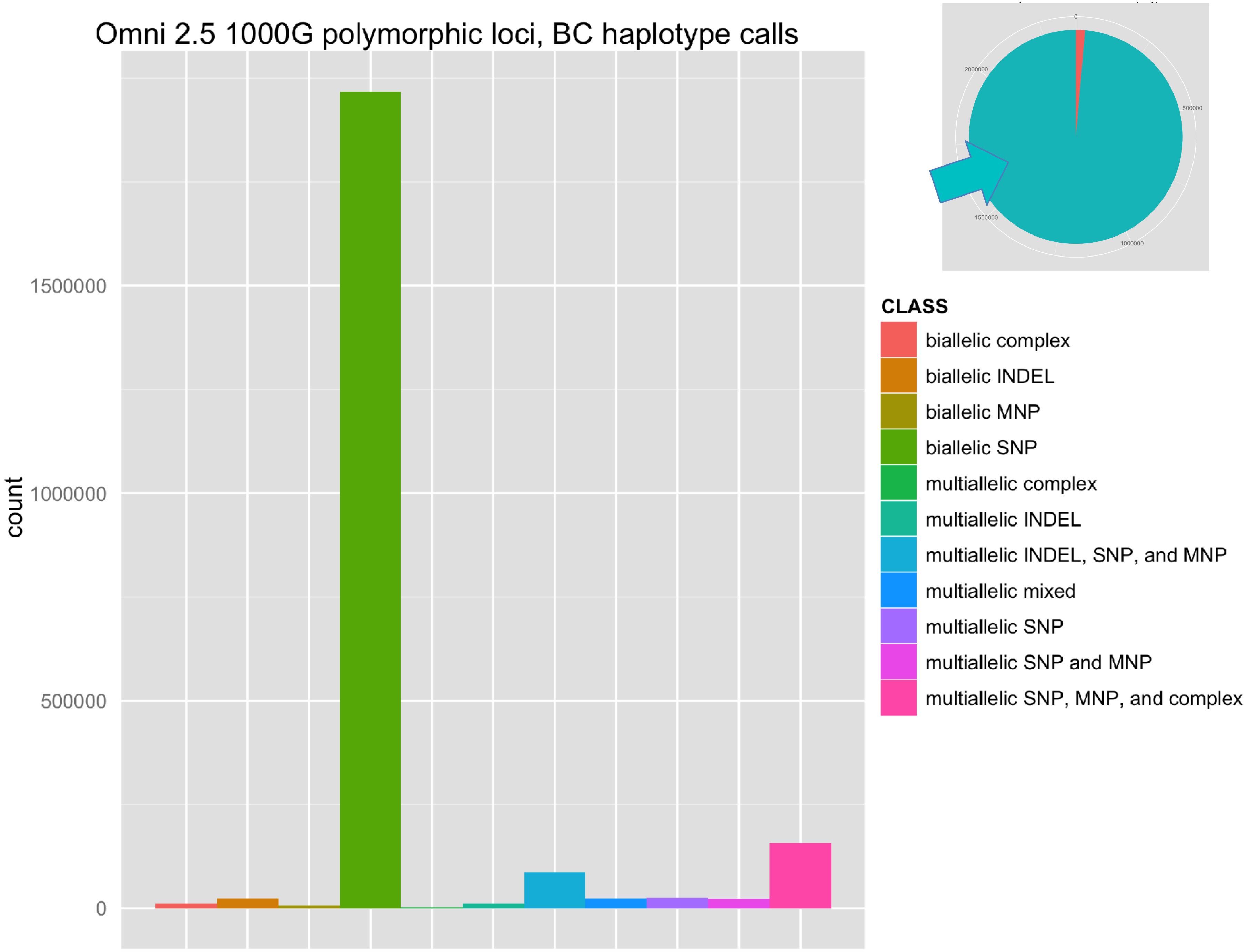
- When the 1000 Genomes samples were genotyped using the chip, 100k of the 2.5 million loci showed no polymorphism (monomorphs).



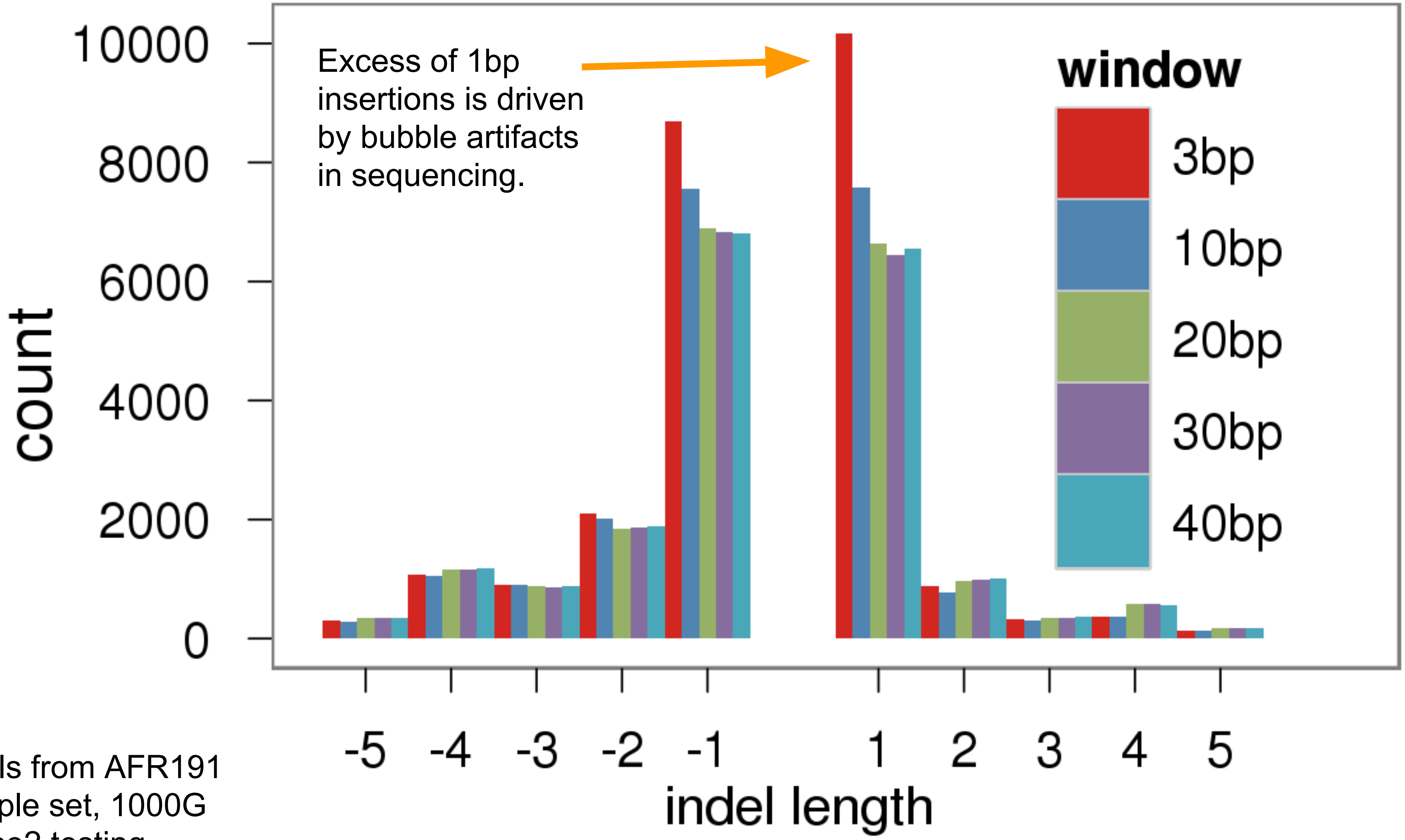
monomorphic loci

Omni 2.5 1000G monomorphic loci, BC haplotype calls




Omni 2.5 1000G polymorphic loci, BC haplotype calls
count
1500000
1000000
500000
0
0
500000
1000000
1500000
2000000
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

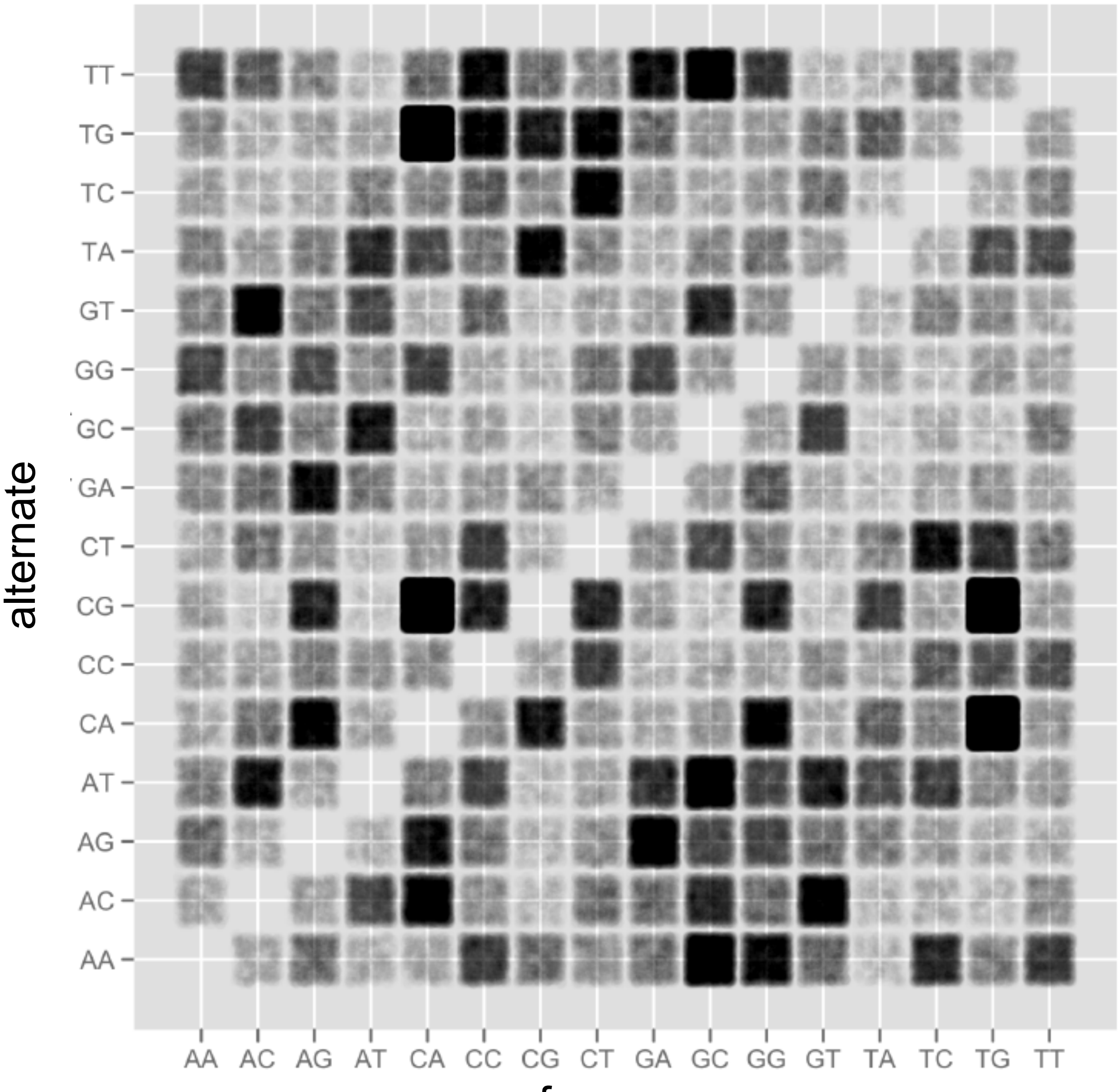
Measuring haplotypes improves specificity



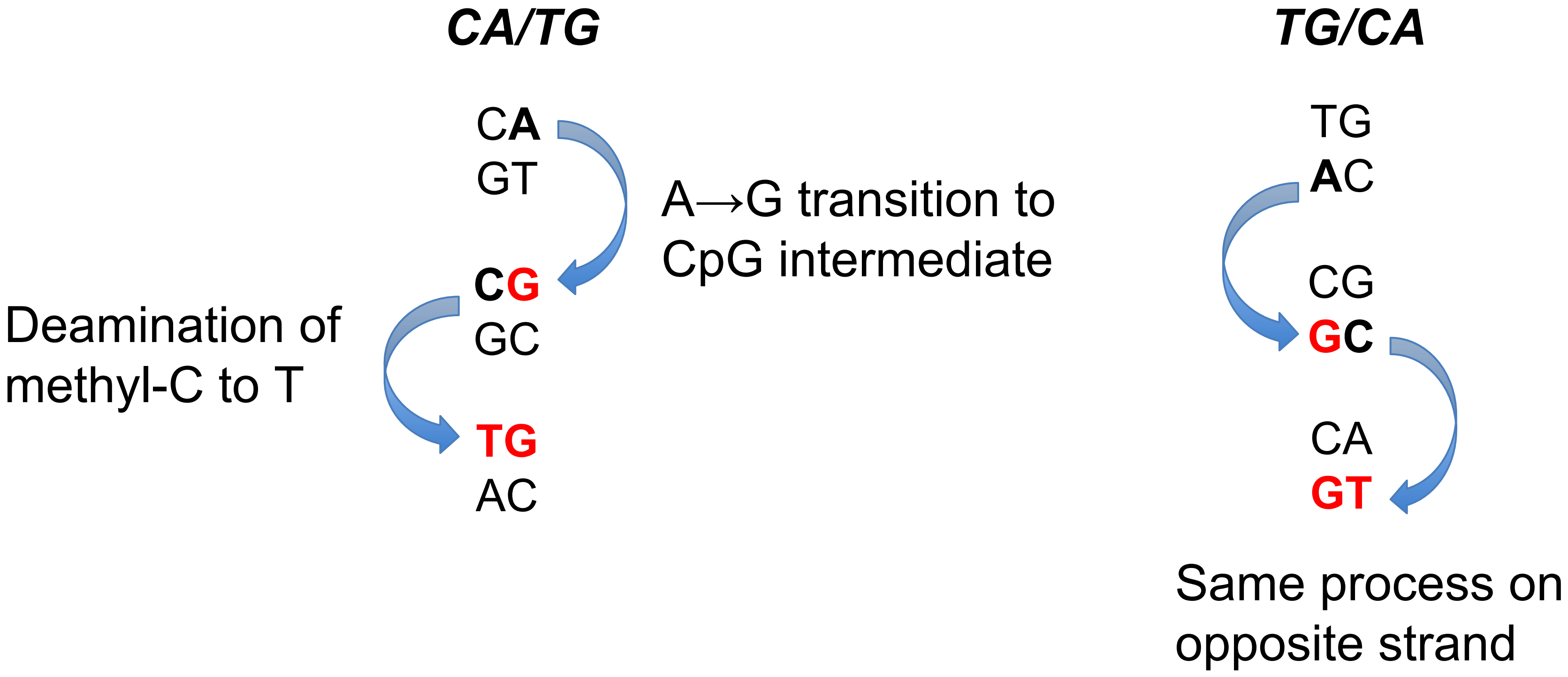
Excess of 1bp insertions is driven by bubble artifacts in sequencing.

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



CA/TG

C**A**
GT

A→G transition to CpG intermediate

CG
GC

Deamination of methyl-C to T

TG
AC

TG/CA

TG
AC

CG
GC

CA
GT

Same process on opposite strand

Overview

1. Genesis of insertion/deletion (indel) polymorphism
2. Standard approaches to detecting indels
3. Assembly-based indel detection
4. Haplotype-based indel detection
5. **Primary filtering: Bayesian variant calling**
6. Post-call filtering: SVM
7. Graph-based resequencing approaches

Filtering INDELs

As with SNPs, sequencing error rates are high.

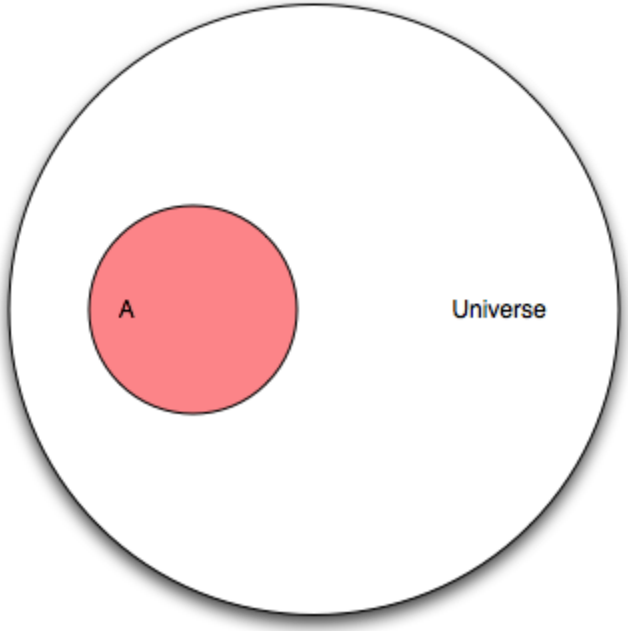
So, we need to filter.

The standard filter of NGS is the Bayesian variant caller.

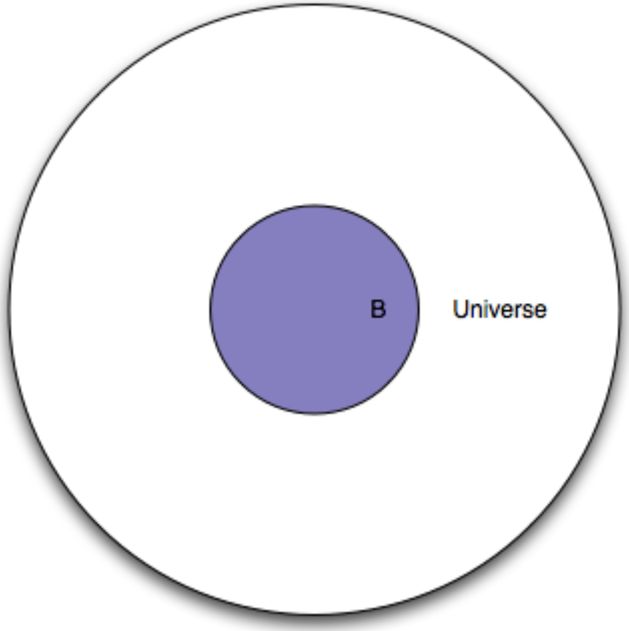
Combines population-based priors and data from many samples to make high-quality calls.

Bayesian (visual) intuition

We have a universe of individuals.



A = samples with a variant at some locus



B = putative observations of variant at some locus

Figures from http://oscarbonilla.com/2009/05/visualizing-bayes-theorem/

probability(A|B)

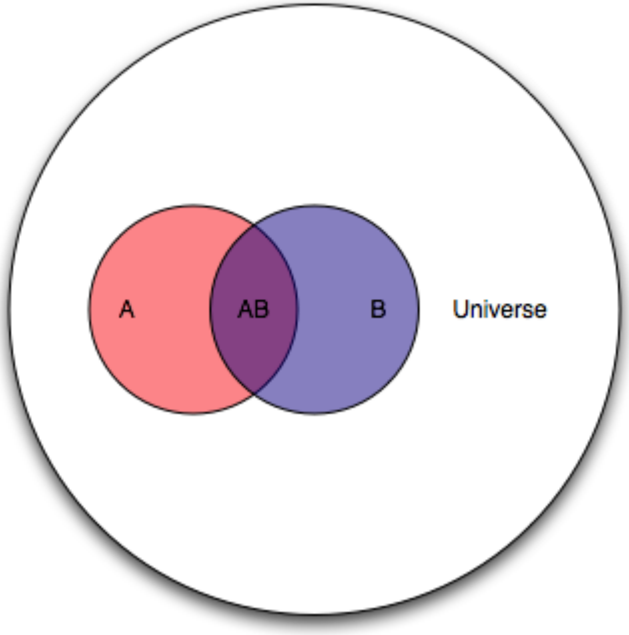
We want to estimate the probability that we have a real polymorphism "A" given "|" that we observed variants in our alignments "B".

$$P(A|B) = \frac{|AB|}{|B|}$$

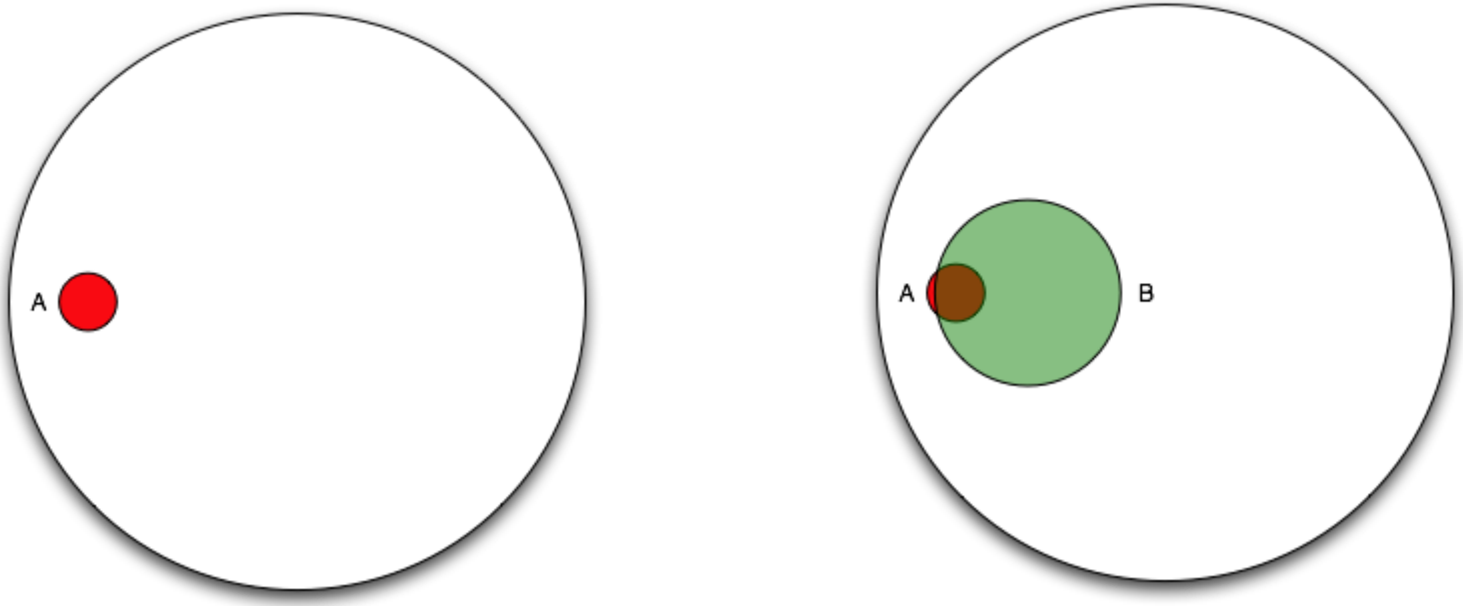
$$P(A|B) = \frac{P(AB)}{P(B)}$$

$$P(B|A) = \frac{P(AB)}{P(A)}$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

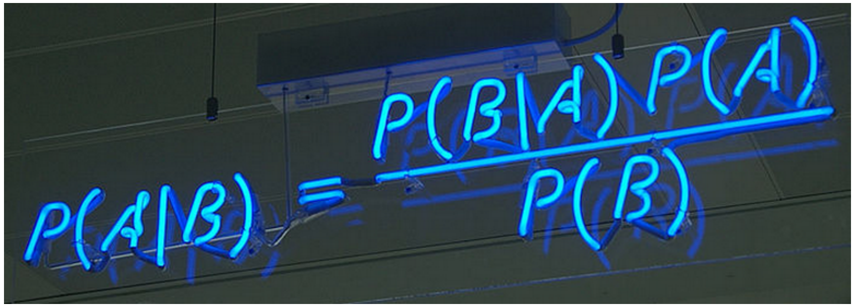


In our case it's a bit more like this...



Observations (B) provide pretty good sensitivity, but poor specificity.

The model



- Bayesian model estimates the probability of polymorphism at a locus given input data and the population mutation rate (~pairwise heterozygosity) and assumption of “neutrality” (random mating).
- Following Bayes theorem, the probability of a specific set of genotypes over some number of samples is:
    - **P(G|R) = (P(R|G) P(G)) / P(R)**
- Which in FreeBayes we extend to:
    - **P(G,S|R) = (P(R|G,S) P(G)P(S)) / P(R)**
    - **G** = genotypes, **R** = reads, **S** = locus is well-characterized/mapped
    - **P(R|G,S)** is our data likelihood, **P(G)** is our prior estimate of the genotypes, **P(S)** is our prior estimate of the mappability of the locus, **P(R)** is a normalizer.

Handling non-biallelic/diploid cases

We compose our data likelihoods, **P(Reads|Genotype)** using a discrete multinomial sampling probability:

$$P(reads|genoytpe) = \binom{|reads|}{|reads = A|, |reads = B| \ldots}$$

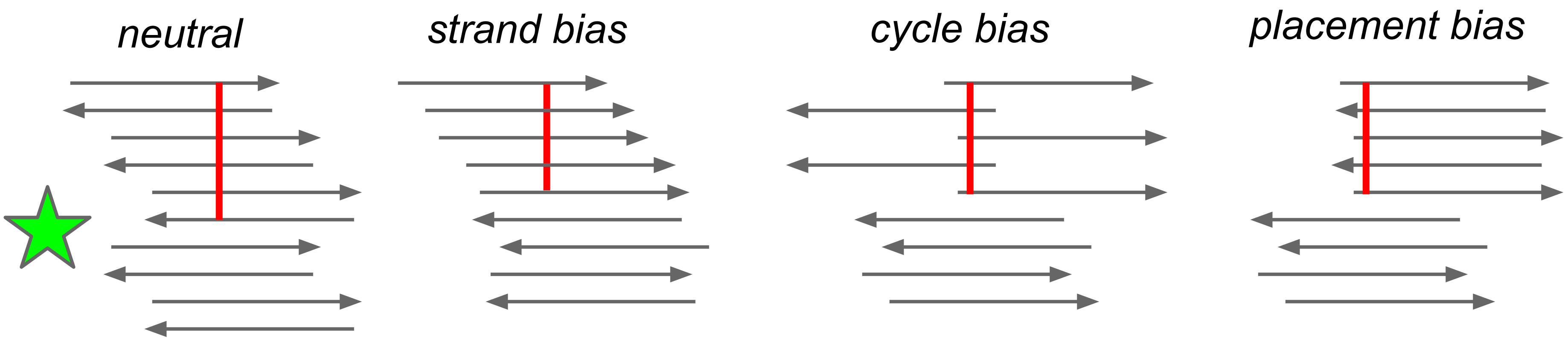
$$\text{x} \prod_{\forall alleles \in genotype} freq(allele \in genotype)$$

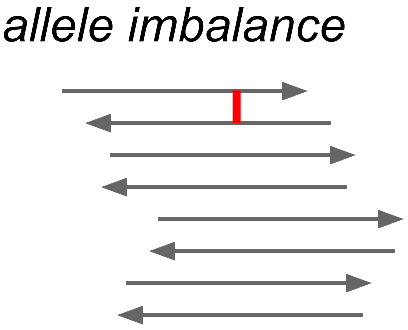
$$\text{x} \prod_{\forall reads} P(correct(read))$$

Our priors, **P(Genoypes)**, follow the Ewens Sampling Formula and the discrete sampling probability for genotypes.

Are our locus and alleles sequenceable?

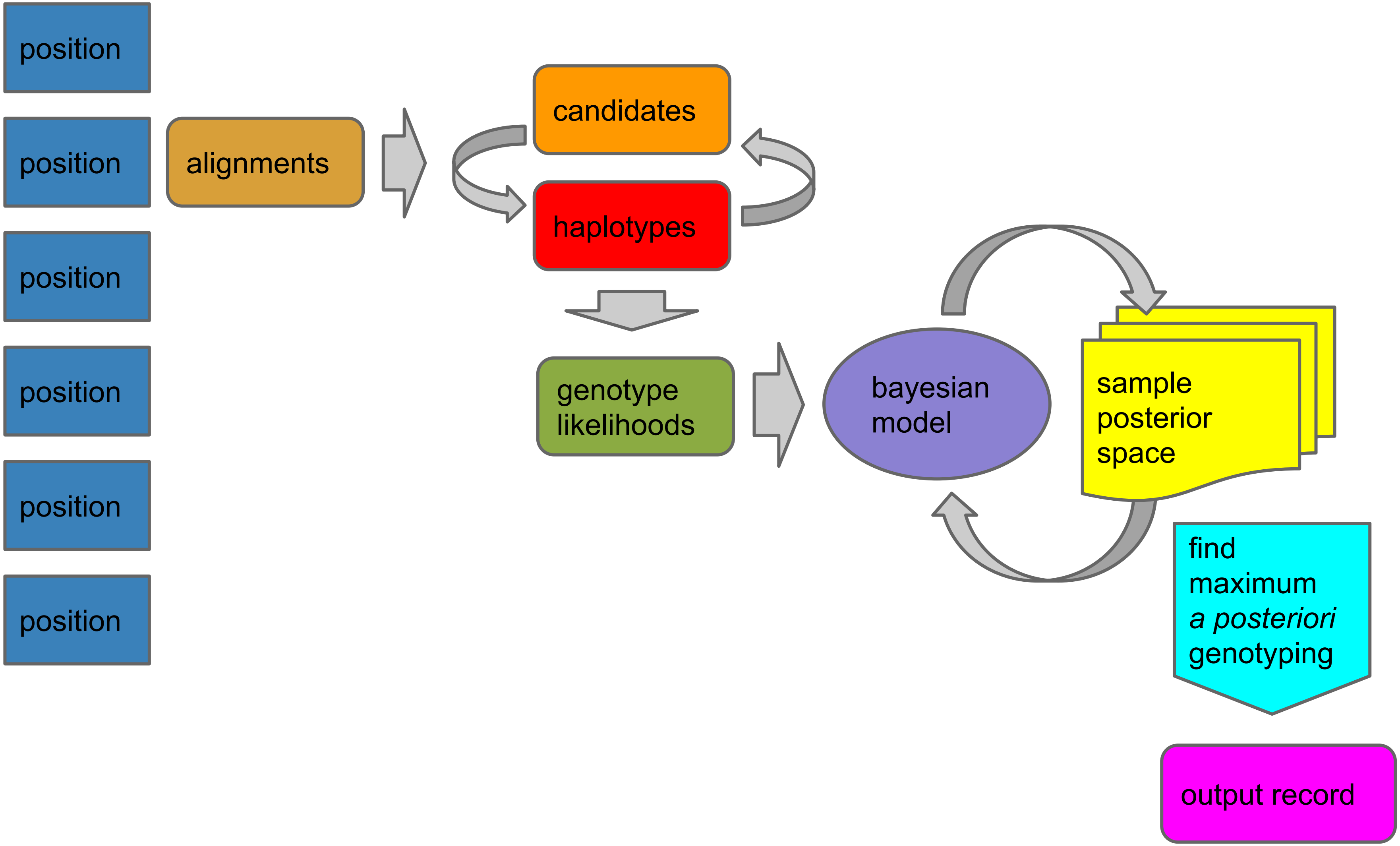
In WGS, biases in the way we observe an allele (placement, position, strand, cycle, or balance in heterozygotes) are often correlated with error. We include this in our posterior **P(G,S|R)**, and to do so we need an estimator of **P(S)**.





$$
\begin{aligned}
P(S) \propto \quad & multinom([|R \equiv b| \forall b_1, \ldots, b_K]; |\{R\}|, f_i, \ldots, f_K) \\
\times \prod_{\forall b \in \{B\}} \quad & binom(|forwardStrand(\{R \equiv b\})|; |\{R \equiv b\}|, 1/2) \\
& \times binom(|placedLeft(\{R \equiv b\})|; |\{R \equiv b\}|, 1/2) \\
& \times binom(|placedRight(\{R \equiv b\})|; |\{R \equiv b\}|, 1/2)
\end{aligned}
$$

The detection process



position

position

position

position

position

position

alignments

candidates

haplotypes

genotype likelihoods

bayesian model

sample posterior space

find maximum *a posteriori* genotyping

output record

Overview

1. Genesis of insertion/deletion (indel) polymorphism
2. Standard approaches to detecting indels
3. Assembly-based indel detection
4. Haplotype-based indel detection
5. Primary filtering: Bayesian variant calling
6. **Post-call filtering: SVM**
7. Graph-based resequencing approaches

SVM filtering

INDEL detection is hard.

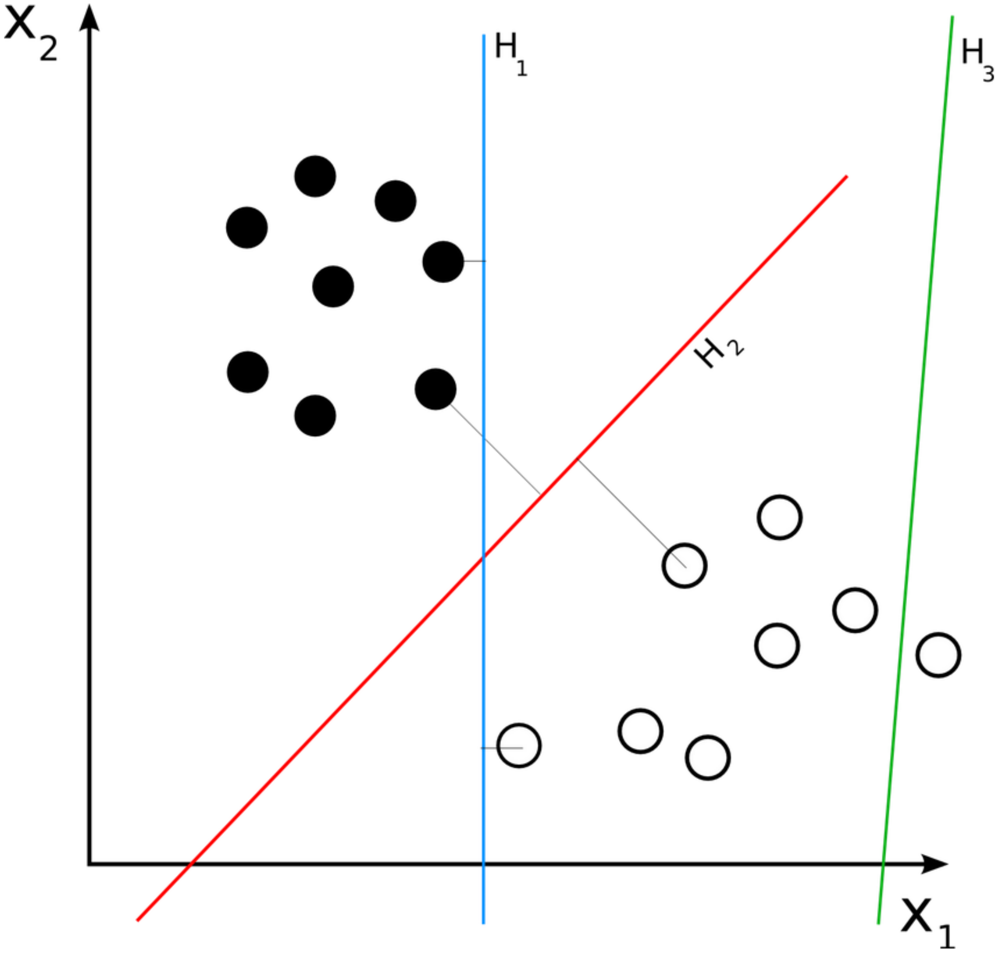
A priori models can’t capture all types of error.

It’s especially difficult when we try to make a consensus set from lots of input variant callers.

We can use classifiers like Support Vector Machines (SVM) to further improve results.

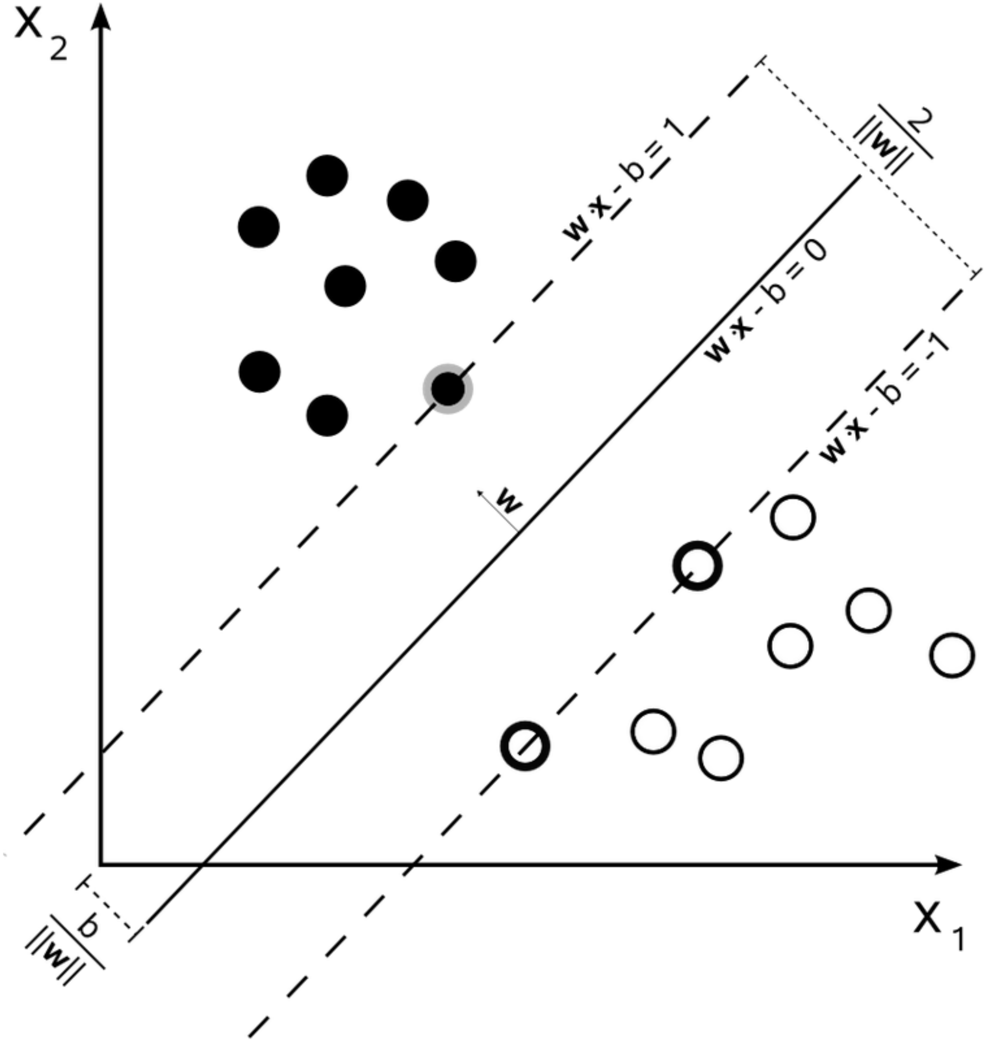
SVM classifier

Find a hyperplane (here a line in 2D) which separates observations.

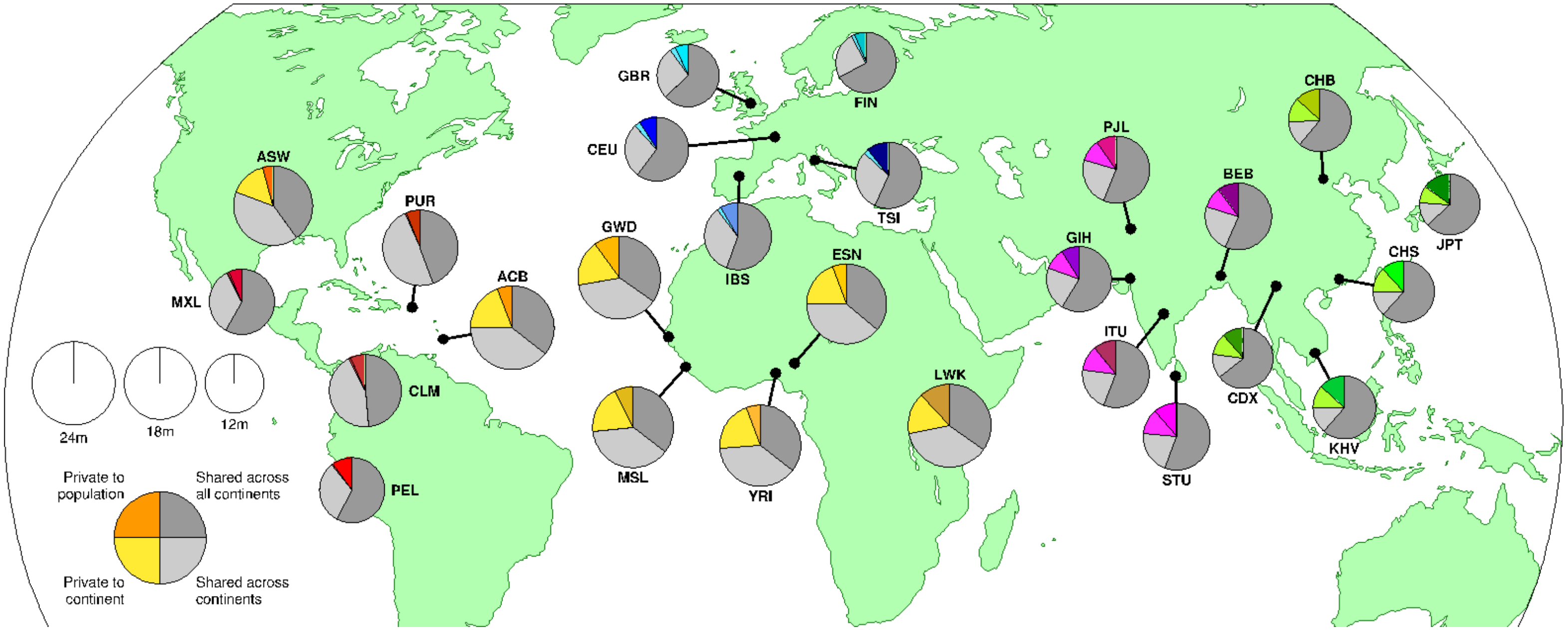


SVM classifier

The best separating hyperplane is determined by maximum margin between groups we want to classify.

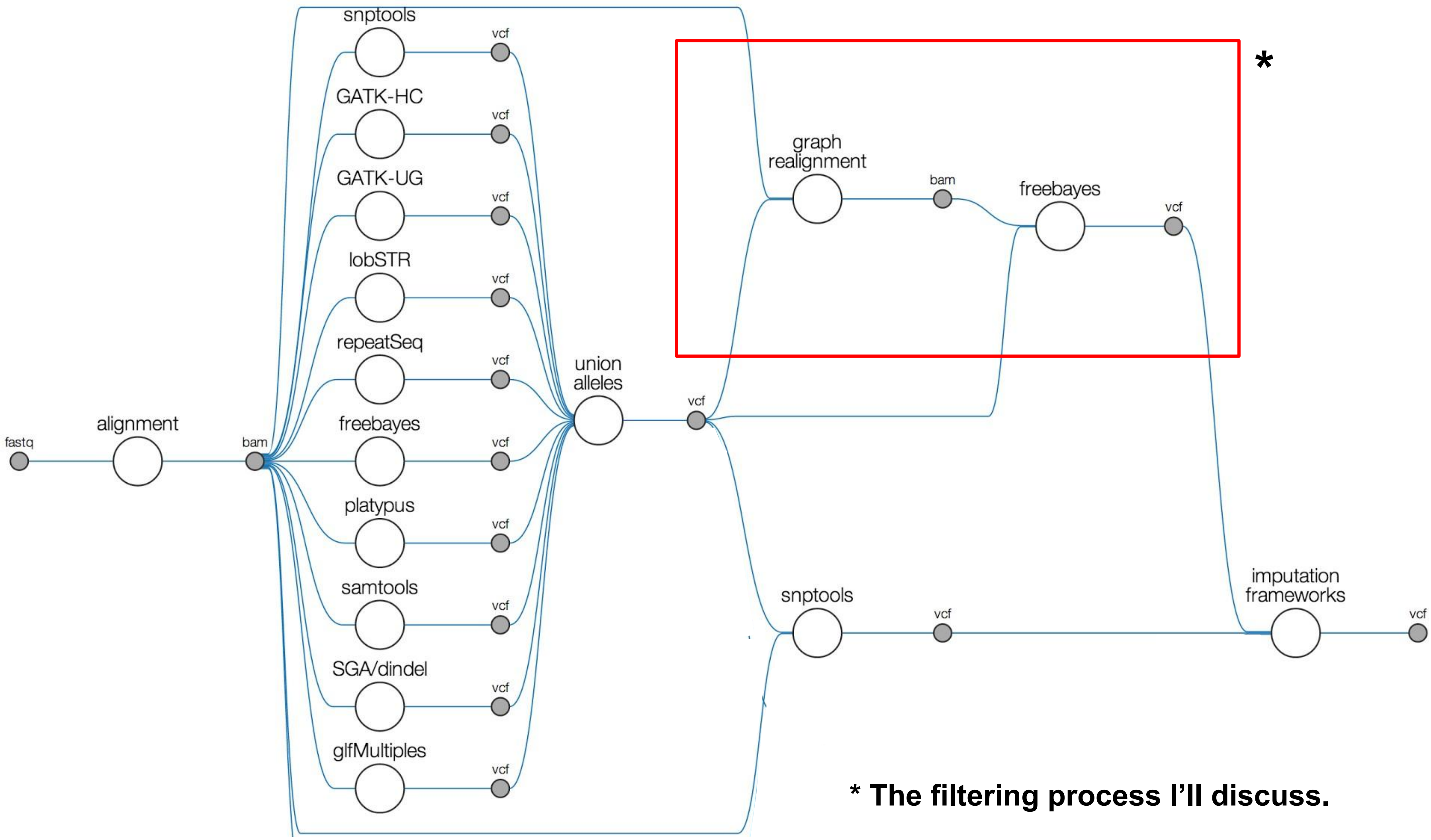


SVM filtering in the 1000 Genomes



25 human populations X ~100 samples each.

1000G variant integration process



* The filtering process I'll discuss.

SVM approach for INDEL filtering

Extract features that tend to vary with respect to call quality:

- call QUALity
- read depth
- sum of base qualities
- inbreeding coefficient (heterozygosity)
- entropy of sequence at locus
- mapping quality
- allele frequency in population
- read pairing rate
- etc.

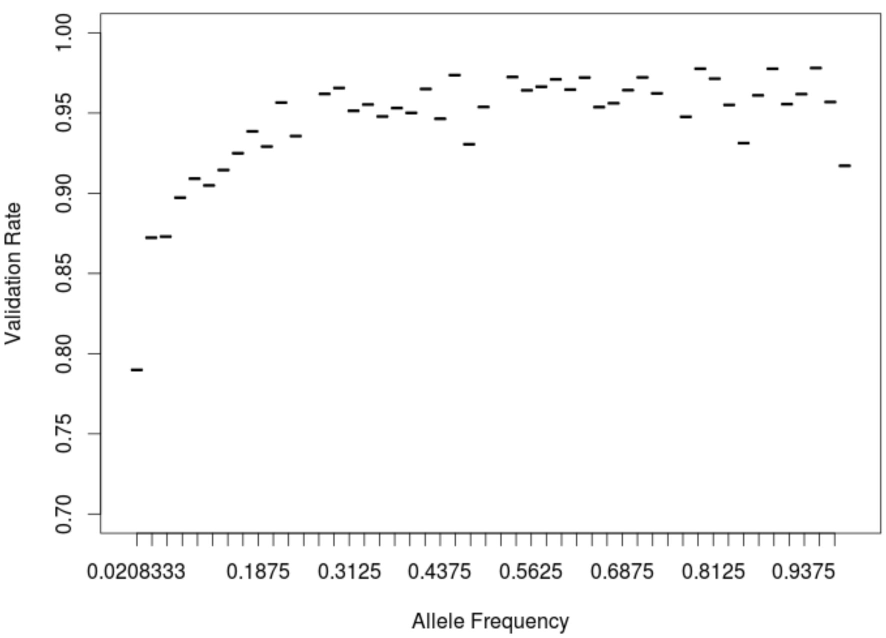
SVM approach for INDEL filtering

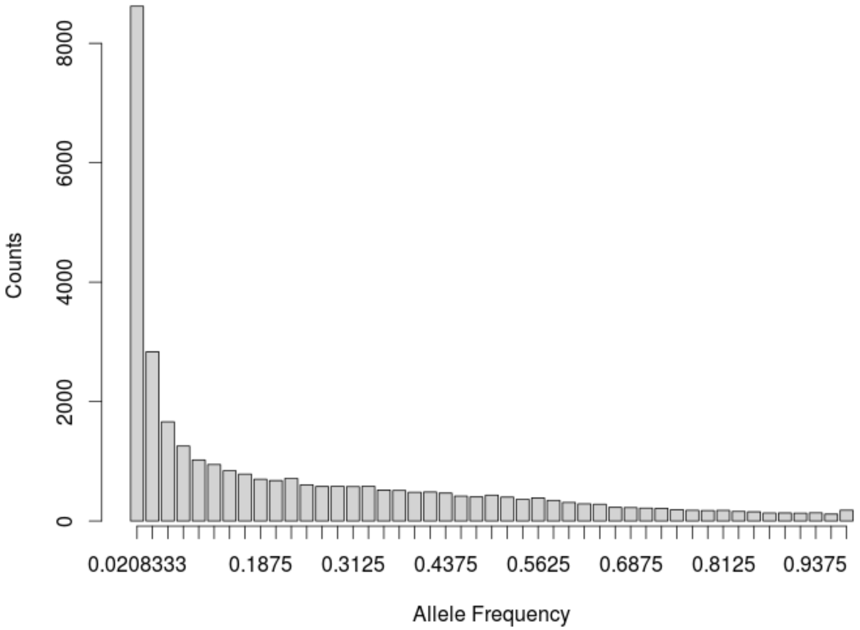
Now, use overlaps in validation samples or sites to determine likely errors and true calls.

Use this list + annotations of the calls to train an SVM model.

Apply the model to all the calls, filter, and measure validation rate of the whole set.

Application of SVM to 1000G INDELs

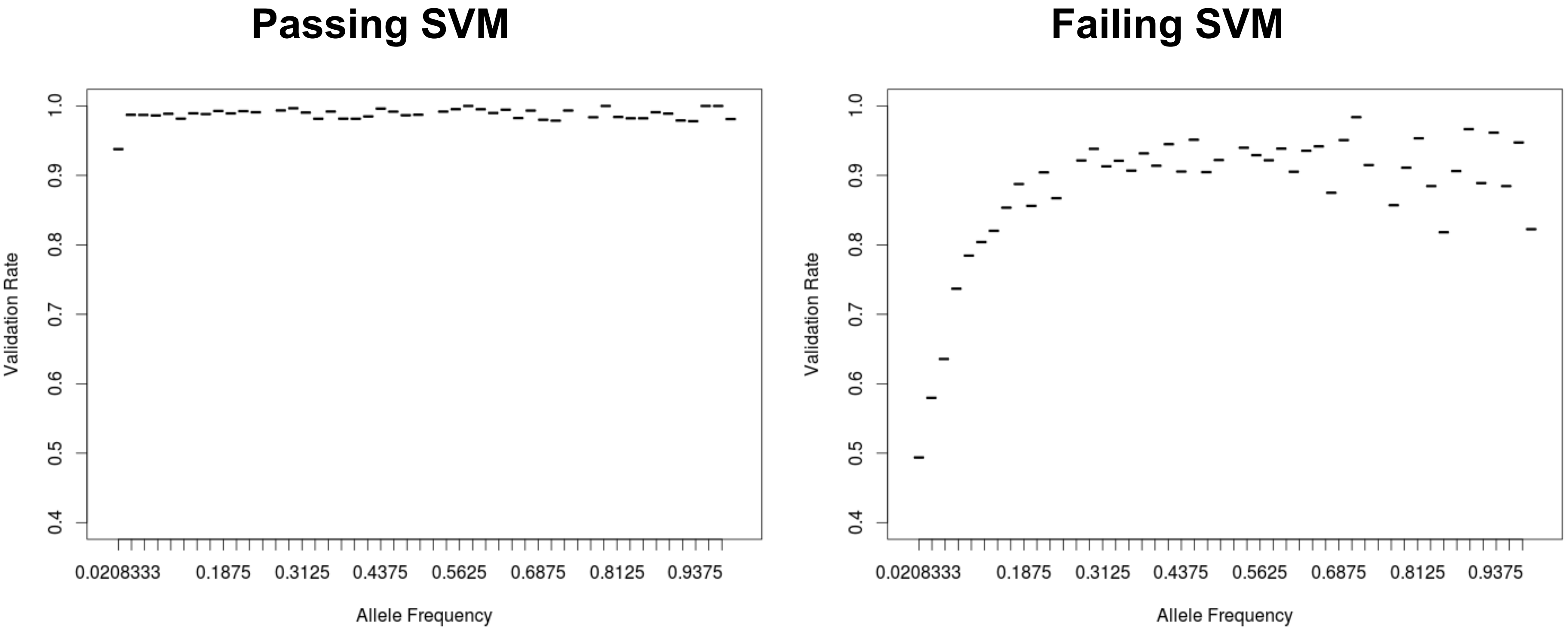




Raw validation rates of indels in 1000G phase 3, “MVNCall” set.

Tony Marcketta and Adam Auton

Application of SVM to 1000G INDELs



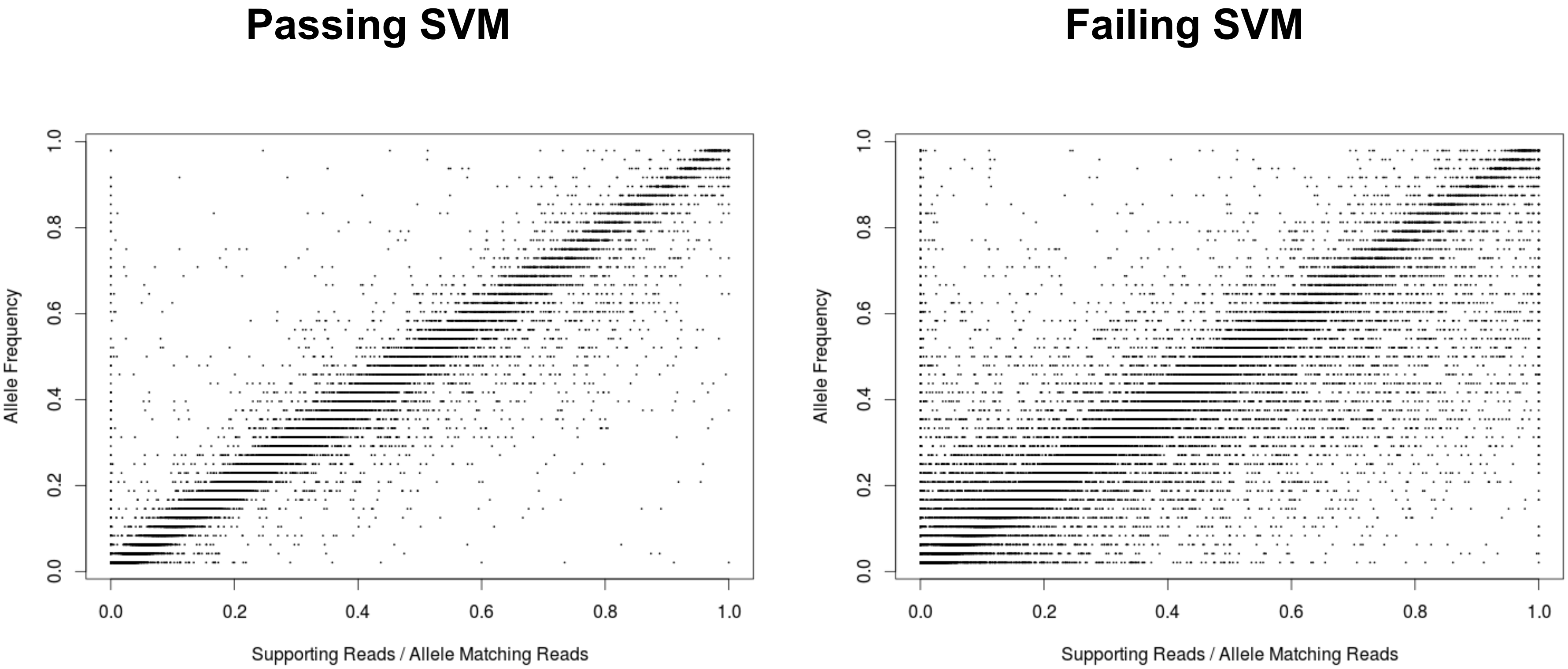
Passing SVM

Failing SVM

Filtering results, using SVM-based method.

Anthony Marcketta and Adam Auton

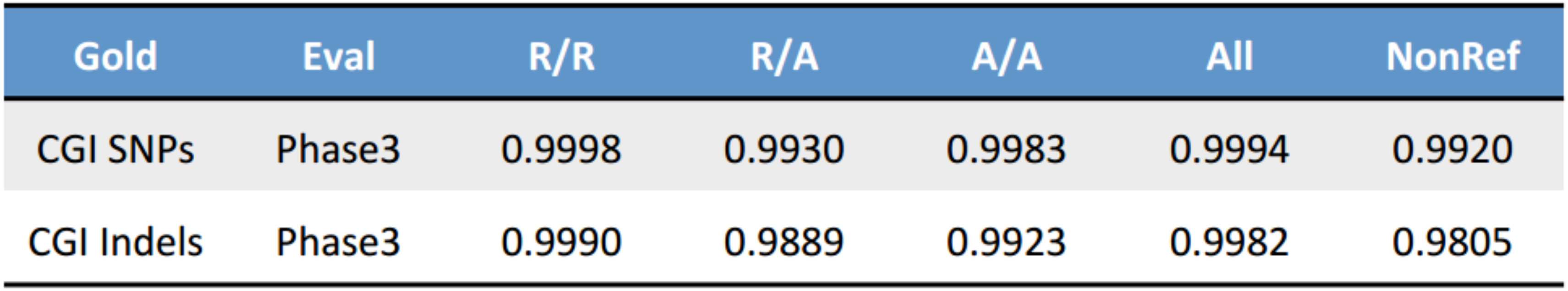
Application of SVM to 1000G INDELs



Correlation between allele frequency and observation counts.

Anthony Marcketta and Adam Auton

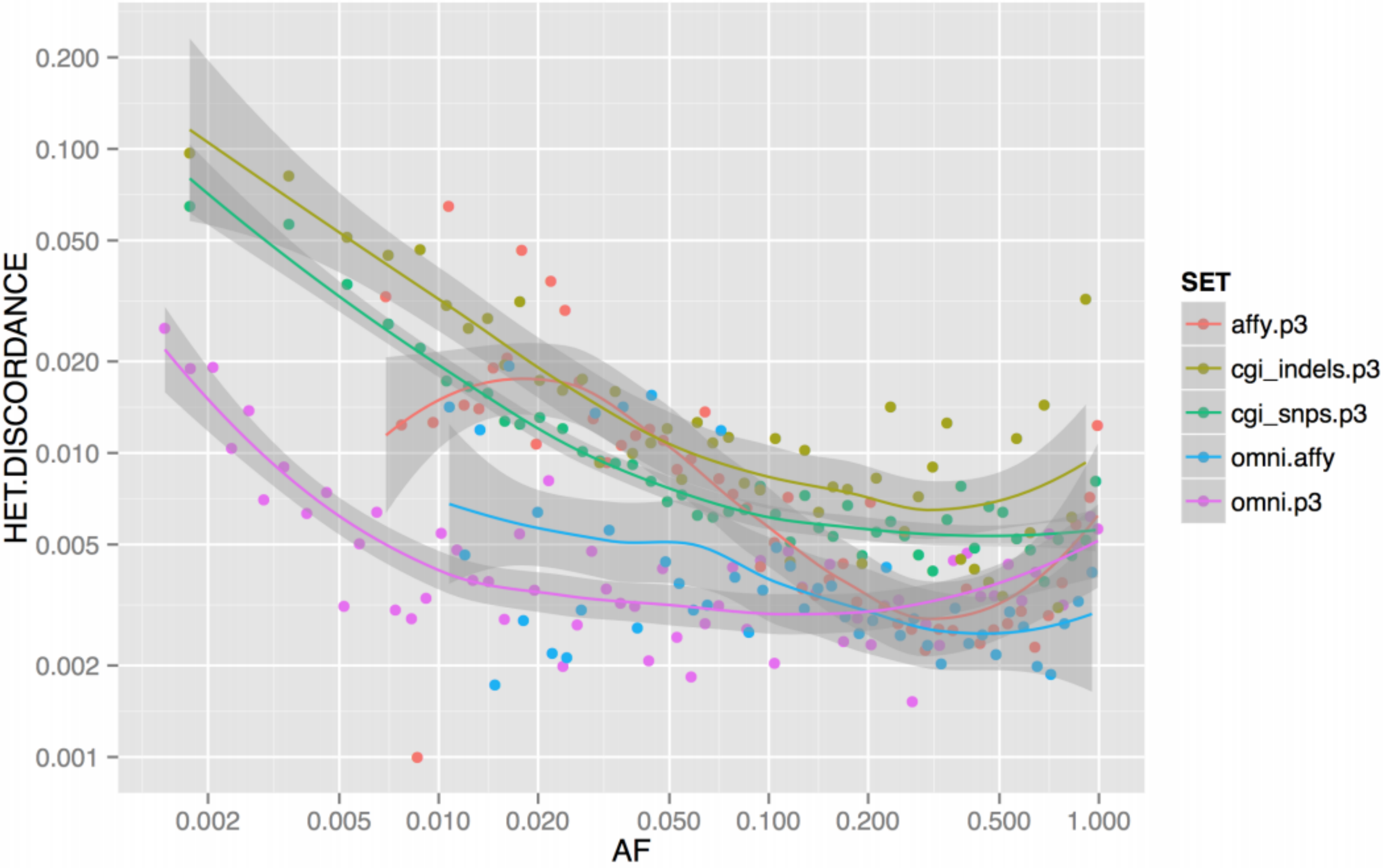
Indel results from 1000G



Gold	Eval	R/R	R/A	A/A	All	NonRef
CGI SNPs	Phase3	0.9998	0.9930	0.9983	0.9994	0.9920
CGI Indels	Phase3	0.9990	0.9889	0.9923	0.9982	0.9805

Comparing the phase3 results to the genotypes for indels in the subset of samples for which we also had high-quality, high-coverage genomes from Complete Genomics.

Genotype Accuracy by Allele Frequency



Overview

1. Genesis of insertion/deletion (indel) polymorphism
2. Standard approaches to detecting indels
3. Assembly-based indel detection
4. Haplotype-based indel detection
5. Primary filtering: Bayesian variant calling
6. Post-call filtering: SVM
7. **Graph-based resequencing approaches**

We know the variants,
so why not use them in our analysis?

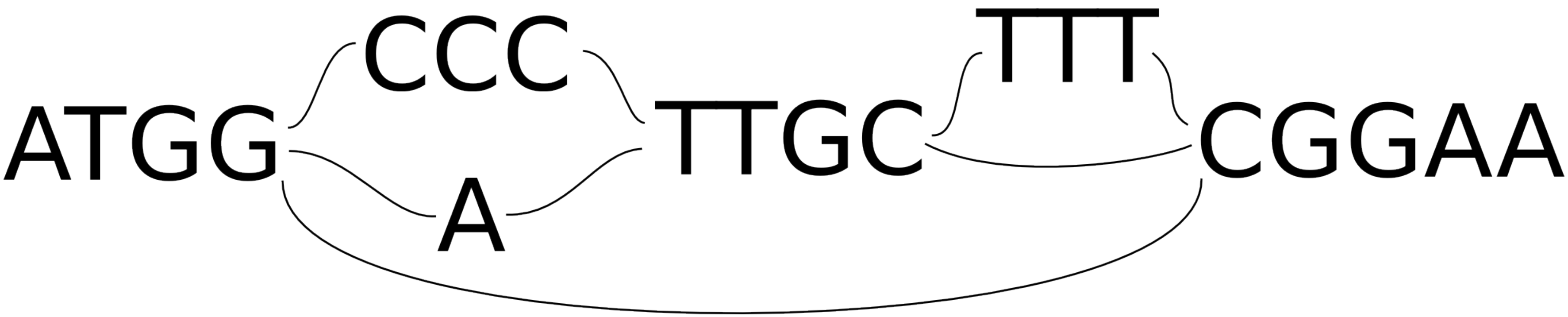
We resequence new genomes and compare them to a single reference haplotype.

To determine anything more than short variants, we must do everything *de novo.*

If we could merge sequence and variation, we could detect known alleles of arbitrary scale and divergence with minimal cost.

Pan-genomes as graphs

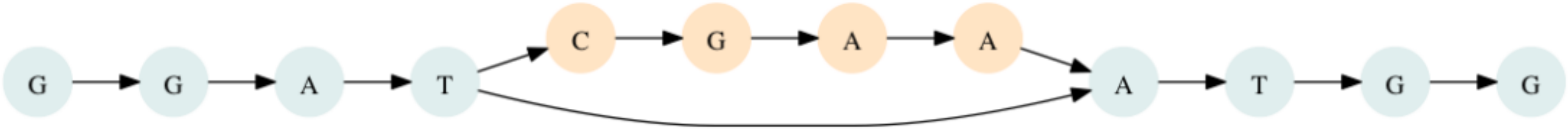
We can combine sequence and variation using a *variant graphs*, or *graph reference*.



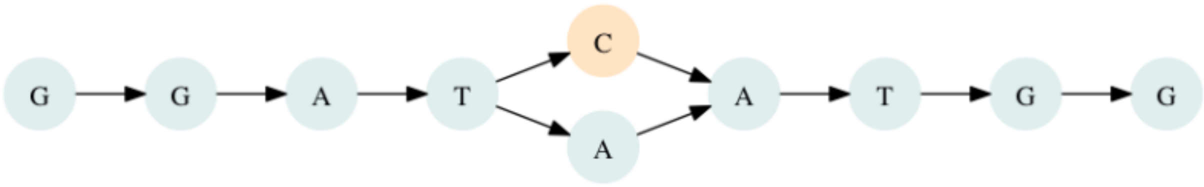
*This representation is directed (5’ to 3’), and acyclic.

Building the variant graph

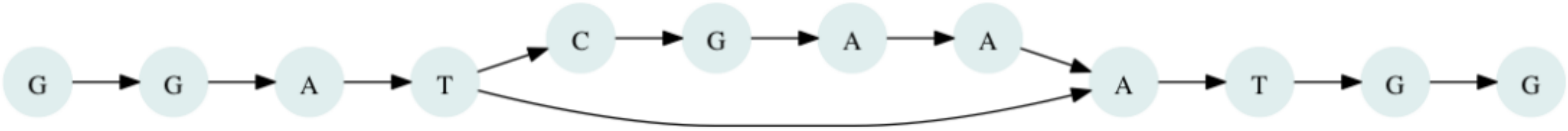
```
#CHROM POS     ID    REF   ALT
20     14370   .     T     TCGAA
```



```
#CHROM POS     ID    REF   ALT
20     14371   .     A     C
```

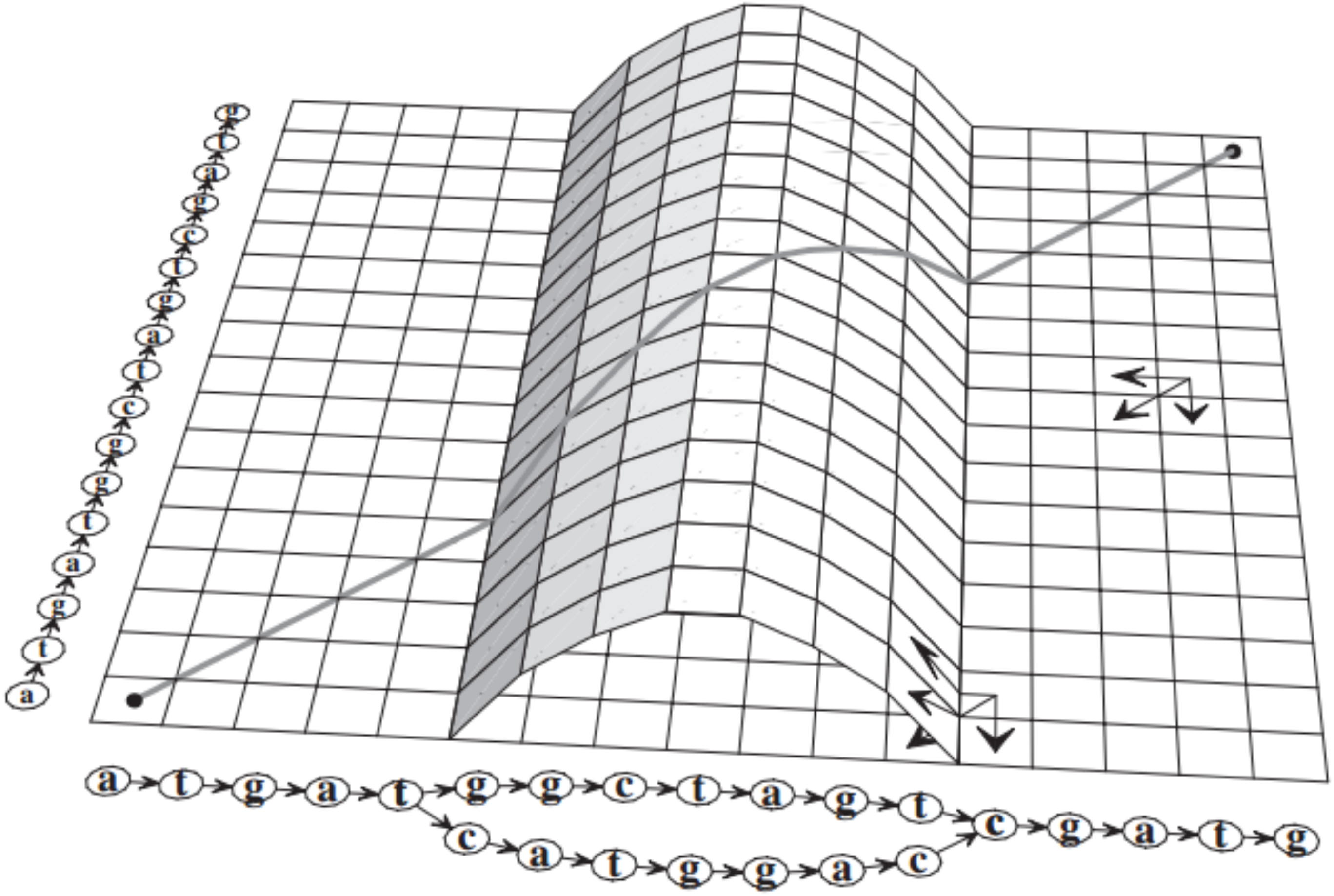


```
#CHROM POS     ID    REF   ALT
20     14370   .     TCGAA T
```



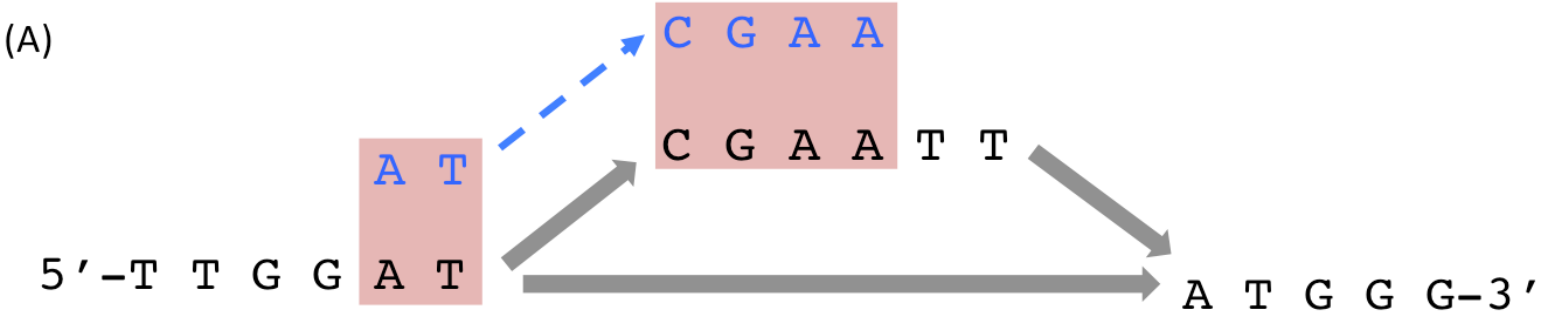
Deniz Kural, Boston College

Local alignment against the graph


a t g a t g g c t a g t c g a t g
a t g a t g g c t a g t c g a t g
c a t g g a c

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

Local alignment against the graph



(B)

		T	T	G	G	A	T
	0	0	0	0	0	0	0
A	0	0	0	0	0	**10**	0
T	0	10	10	0	0	0	**20**
C	0	0	0	0	0	0	10
G	0	0	0	10	10	0	0
A	0	0	0	0	0	20	10
A	0	0	0	0	0	10	10

	C	G	A	A	T	T
0	0	0	0	0	0	0

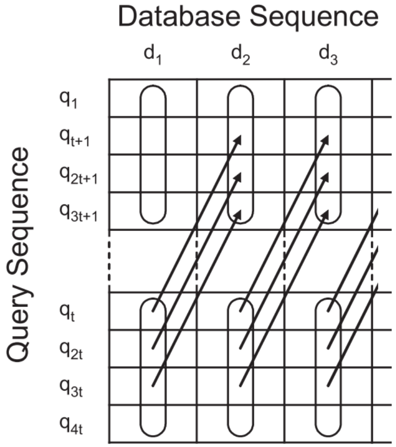
A	0	0	0	10	10	0	0
T	**20**	10	0	0	0	20	10
C	10	**30**	20	10	0	10	10
G	0	20	**40**	30	20	10	0
A	10	10	30	**50**	40	30	20
A	10	0	20	40	**60**	50	40

		A	T	G	G	G
	0	0	0	0	0	0
A	0	10	0	0	0	0
T	20	10	20	10	0	0
C	10	10	10	10	0	0
G	0	0	0	20	20	10
A	20	10	0	10	10	10
A	40	30	20	10	0	0

Deniz Kural, Boston College

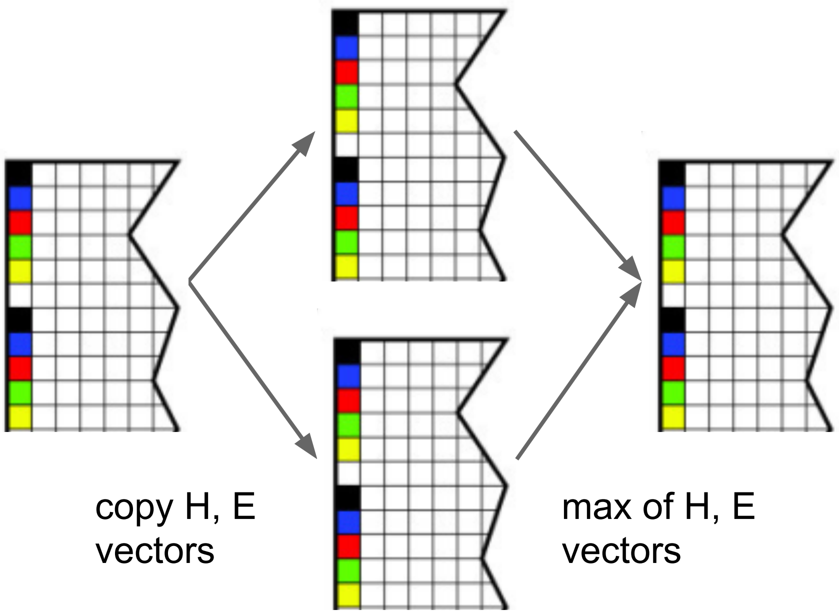
“Striped” string/DAG alignment

We improved performance of our aligner >10-fold by generalizing Farrar’s striped Smith-Waterman algorithm to DAGs. ***GSSW***



Data dependencies across DAG are limited to H and E vectors.

*Implemented using SSE2 instruction set.



copy H, E vectors

max of H, E vectors

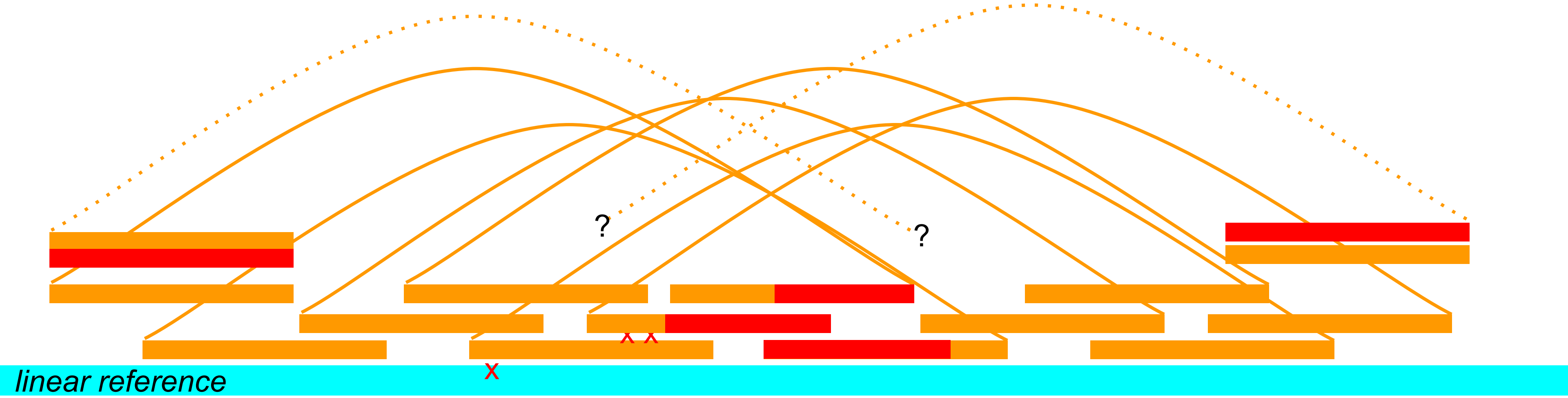
$$E_{i,j} = \max\left\{\begin{matrix} E_{i,j-1} - G_{\text{ext}} \\ H_{i,j-1} - G_{\text{init}} \end{matrix}\right\}$$

$$F_{i,j} = \max\left\{\begin{matrix} F_{i-1,j} - G_{\text{ext}} \\ H_{i-1,j} - G_{\text{init}} \end{matrix}\right\}$$

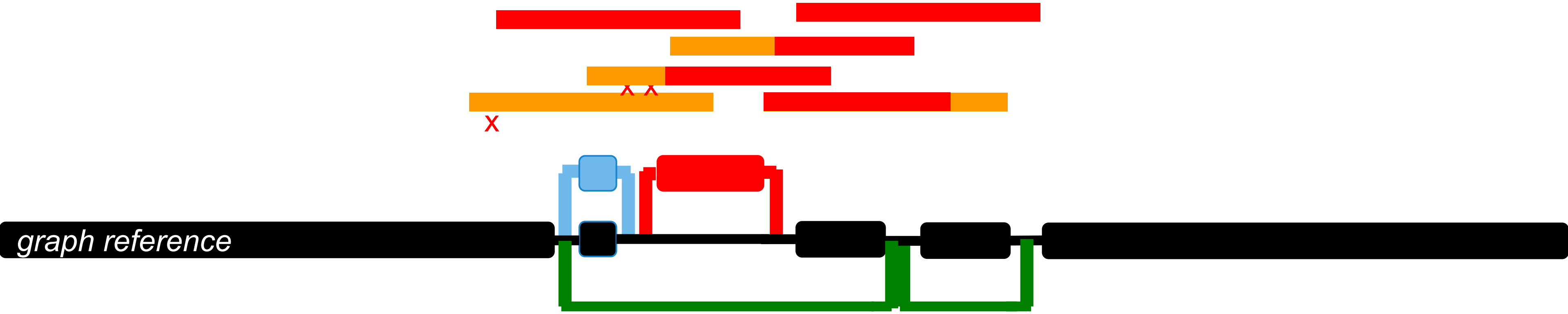
$$H_{i,j} = \max\left\{\begin{matrix} 0 \\ E_{i,j} \\ F_{i,j} \\ H_{i-1,j-1} - W(q_i, d_j) \end{matrix}\right\}$$

Farrar, Bioinformatics (2006); Rognes, BMC Bioinformatics (2011); Zhao, PLoS One (2014)

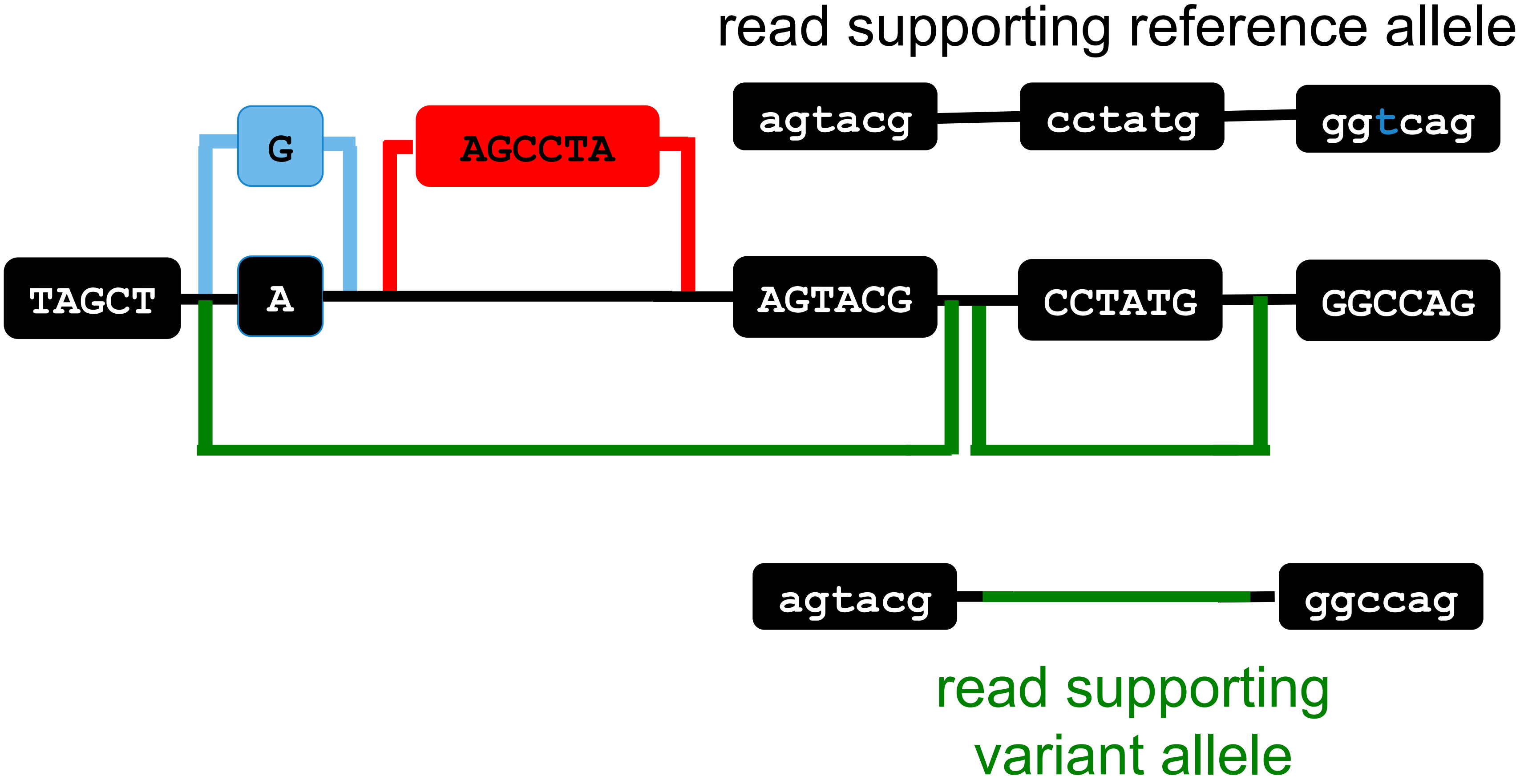
Seeding graph-based alignments



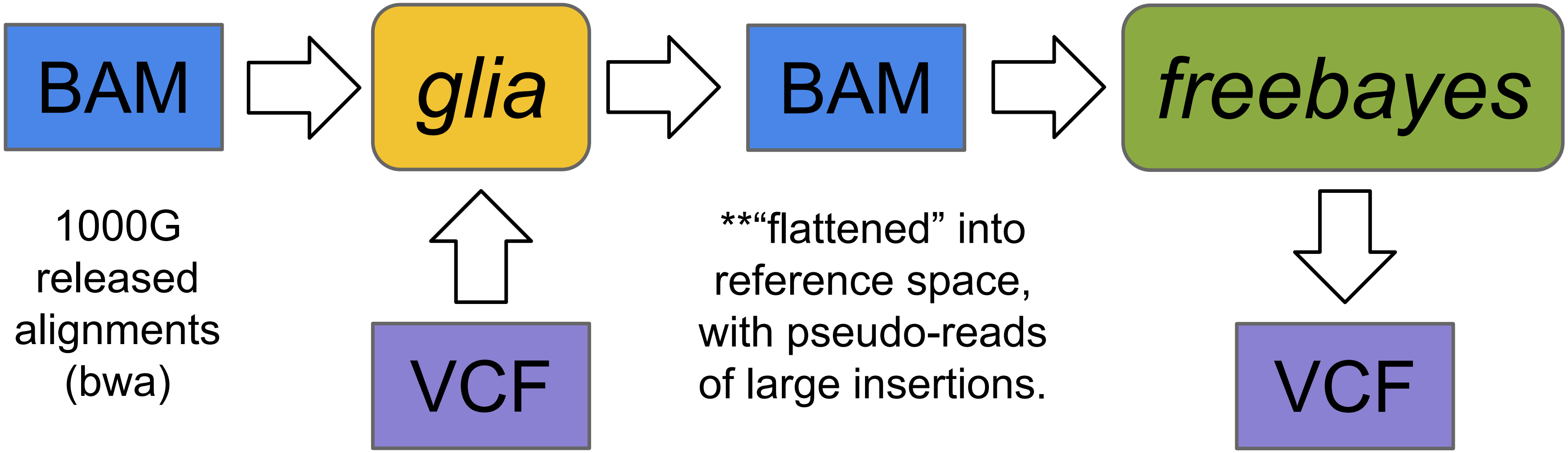
Test imperfectly-mapped reads against graph.



Detecting variation on the graph



Graph-based alignments with *glia*



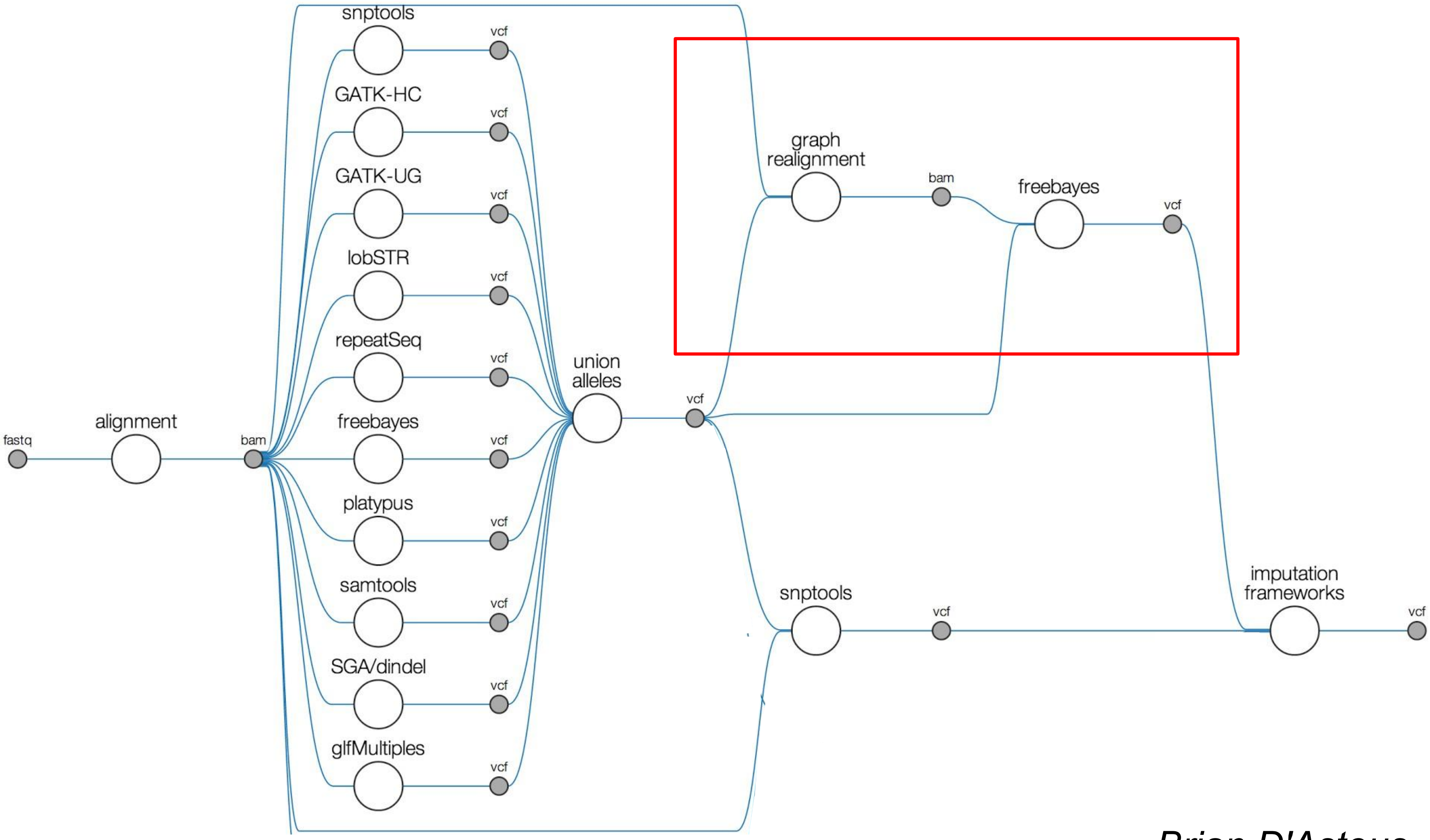
1000G released alignments (bwa)

union alleles with exact breakpoints, used to build local graph

**“flattened” into reference space, with pseudo-reads of large insertions.

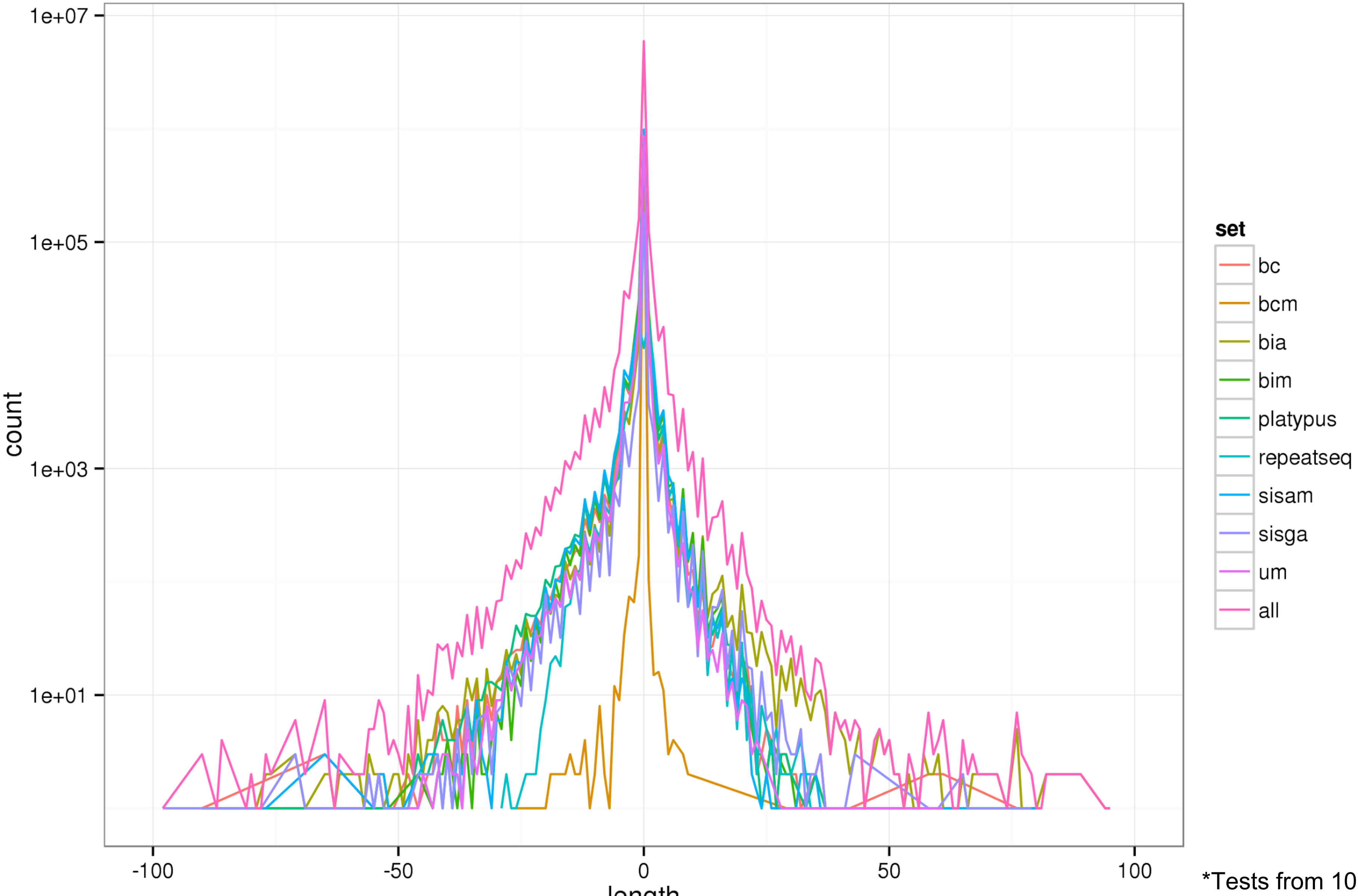
genotyped input alleles

Application to 1000G variant integration

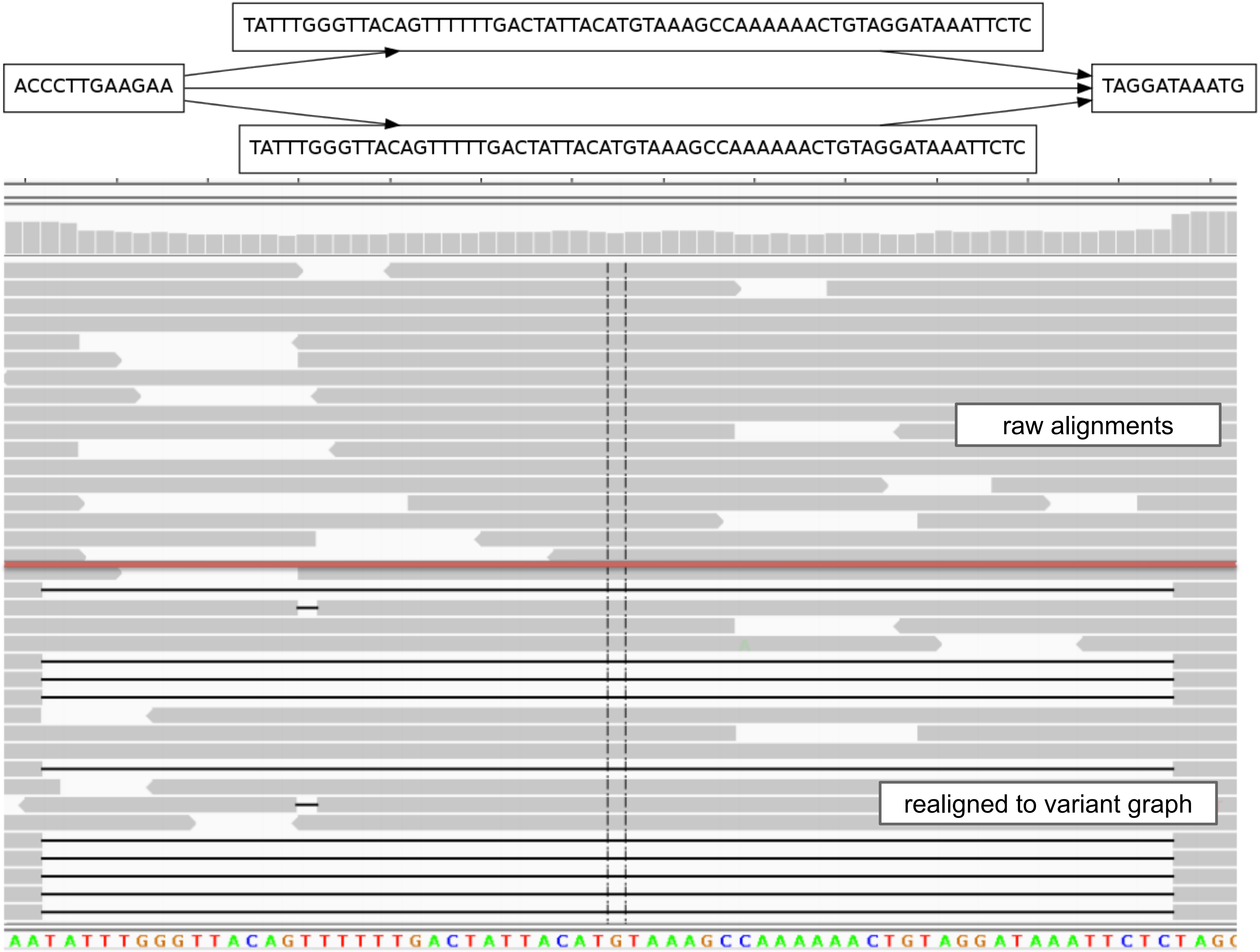


Brian D'Astous

Unifying calls from many methods



*Tests from 1000G
“phase3-like” chr20.



TATTTGGGTTACAGTTTTTTGACTATTACATGTAAAGCCAAAAAACTGTAGGATAAATTCTC

ACCCTTGAAGAA

TAGGATAAATG

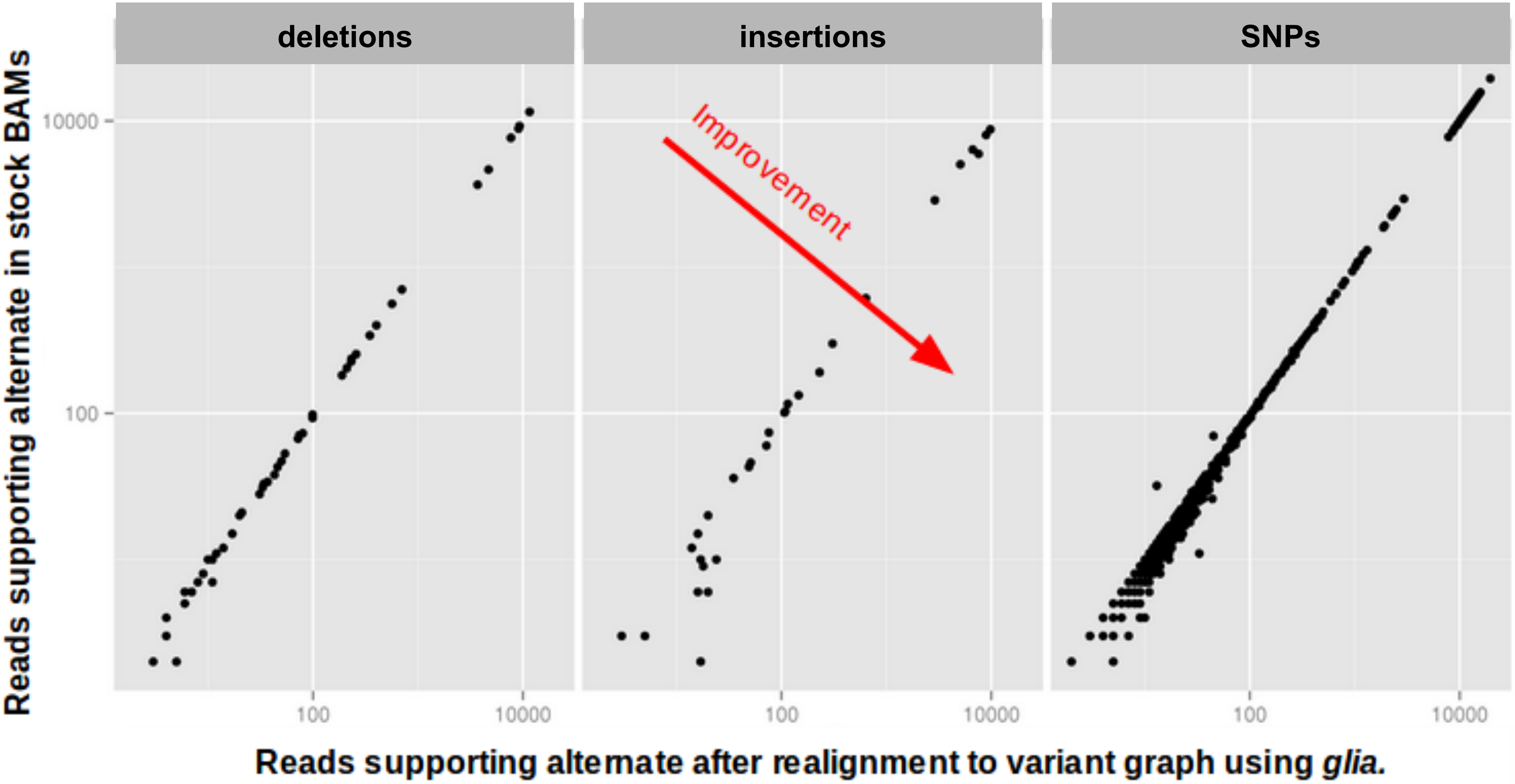
TATTTGGGTTACAGTTTTTGACTATTACATGTAAAGCCAAAAAACTGTAGGATAAATTCTC

raw alignments

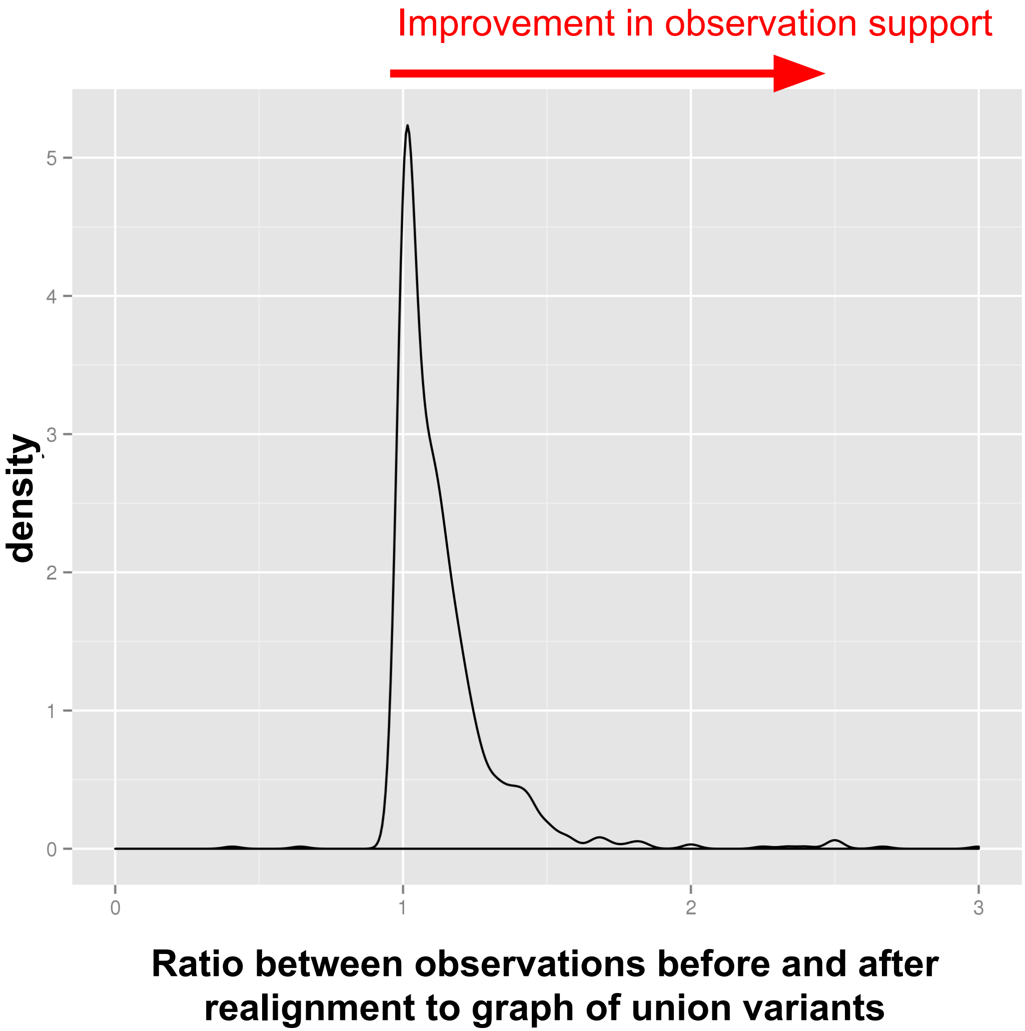
realigned to variant graph

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

glia reduces reference bias



glia reduces reference bias



Standard alignment is frustrated even by small variants!

Improving genotype likelihoods

Genotype Likelihood = P(data|genotype)

SET	GRP	N	RR	RA	AA	ALT	ALL
SVM indels	UM	6743	0.285	1.008	2.947	1.698	0.561
SVM indels	BC*	6743	**0.034**	**0.673**	**0.245**	**0.521**	**0.129**
SNPs	BCM	404270	0.029	1.373	0.445	1.093	0.111

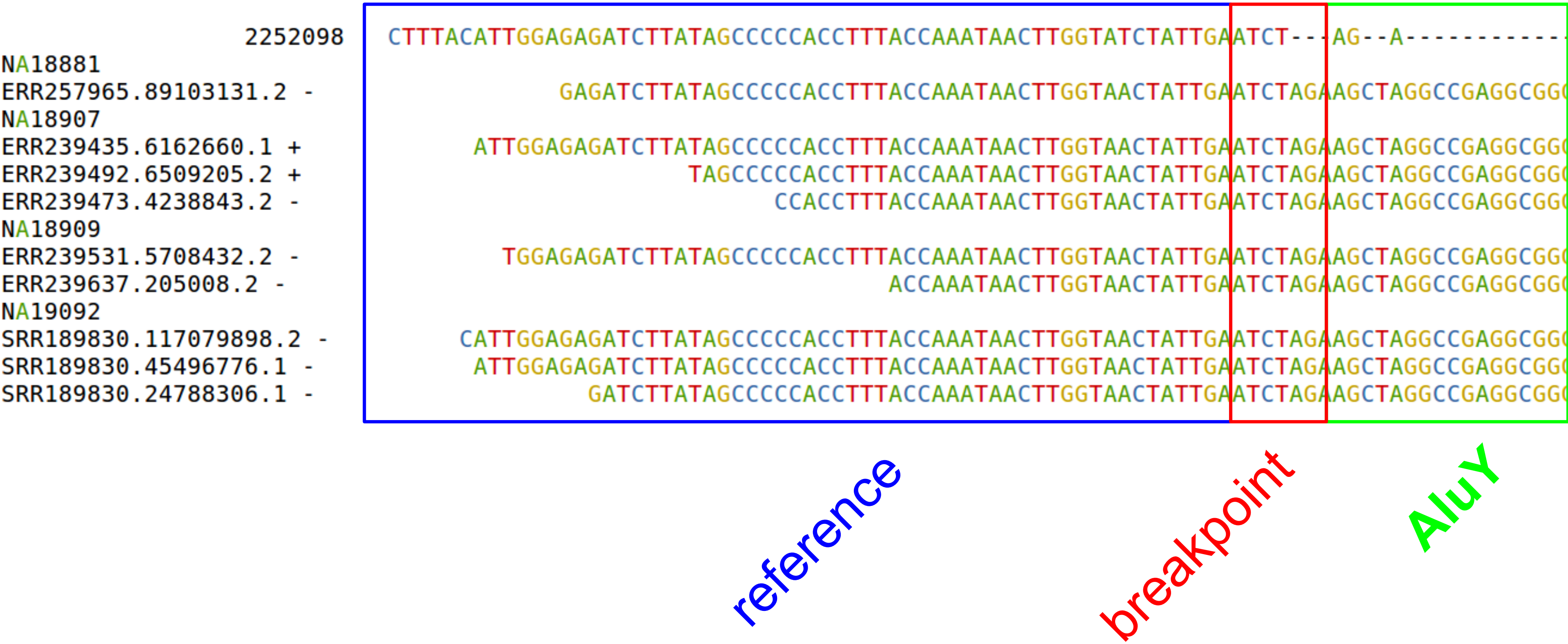
* includes glia realignment

Imputation of variant calls on chr20 via SHAPEIT 2. Imputed results are tested against Complete Genomics samples in 1000 Genomes.

We do as well for high-quality indels as SNPs!

Olivier Delaneau, Androniki Menelaou, Jonathan Marchini

Mobile element detection



```
                       2252098 CTTTACATTGGAGAGATCTTATAGCCCCCACCTTTACCAAATAACTTGGTATCTATTGAATCT---AG--A-----------
NA18881
ERR257965.89103131.2 -                     GAGATCTTATAGCCCCCACCTTTACCAAATAACTTGGTAACTATTGAATCTAGAAGCTAGGCCGAGGCGG
NA18907
ERR239435.6162660.1 +                ATTGGAGAGATCTTATAGCCCCCACCTTTACCAAATAACTTGGTAACTATTGAATCTAGAAGCTAGGCCGAGGCGG
ERR239492.6509205.2 +                               TAGCCCCCACCTTTACCAAATAACTTGGTAACTATTGAATCTAGAAGCTAGGCCGAGGCGG
ERR239473.4238843.2 -                                     CCACCTTTACCAAATAACTTGGTAACTATTGAATCTAGAAGCTAGGCCGAGGCGG
NA18909
ERR239531.5708432.2 -                  TGGAGAGATCTTATAGCCCCCACCTTTACCAAATAACTTGGTAACTATTGAATCTAGAAGCTAGGCCGAGGCGG
ERR239637.205008.2 -                                  ACCAAATAACTTGGTAACTATTGAATCTAGAAGCTAGGCCGAGGCGG
NA19092
SRR189830.117079898.2 -             CATTGGAGAGATCTTATAGCCCCCACCTTTACCAAATAACTTGGTAACTATTGAATCTAGAAGCTAGGCCGAGGCGG
SRR189830.45496776.1 -               ATTGGAGAGATCTTATAGCCCCCACCTTTACCAAATAACTTGGTAACTATTGAATCTAGAAGCTAGGCCGAGGCGG
SRR189830.24788306.1 -                      GATCTTATAGCCCCCACCTTTACCAAATAACTTGGTAACTATTGAATCTAGAAGCTAGGCCGAGGCGG
```

Using *glia+freebayes* to re-genotype an AluY insertion at 20:2252139 in the YRI population. Insertion structure is estimated from split-read mappings.

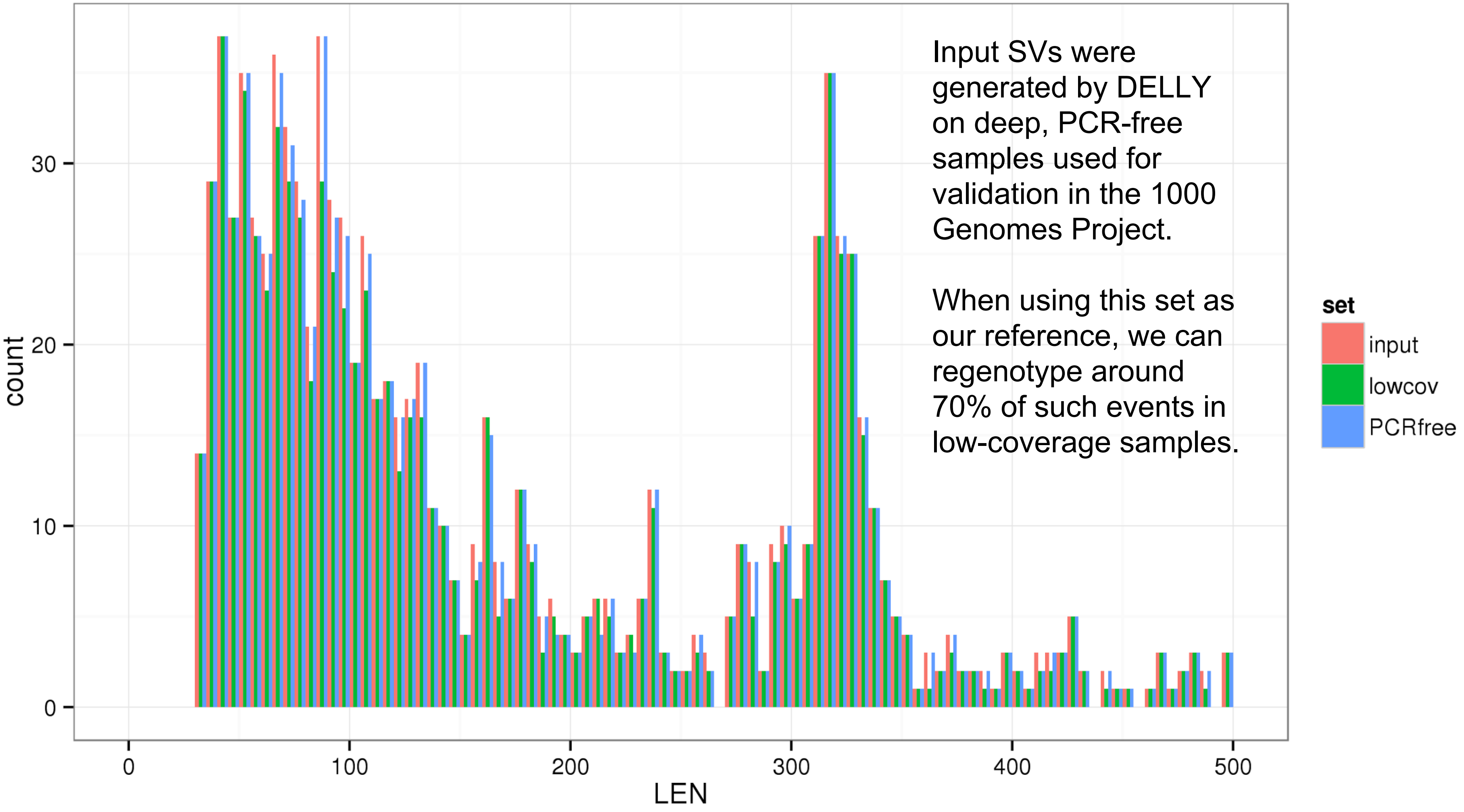
Alu genotyping efficiency

We re-call a substantial fraction of known (validated) Alus in 1000G low-coverage bwa alignments.

set	re-genotyped Alus	%
Pilot 2 data (source)	282	99.6
PCR-free NA12878	281	99.3
5x NA12878 (low-coverage)	173	61.1

Stewart et. al 2011. A Comprehensive Map of Mobile Element Insertion Polymorphisms in Humans. *PLoS Genetics.*

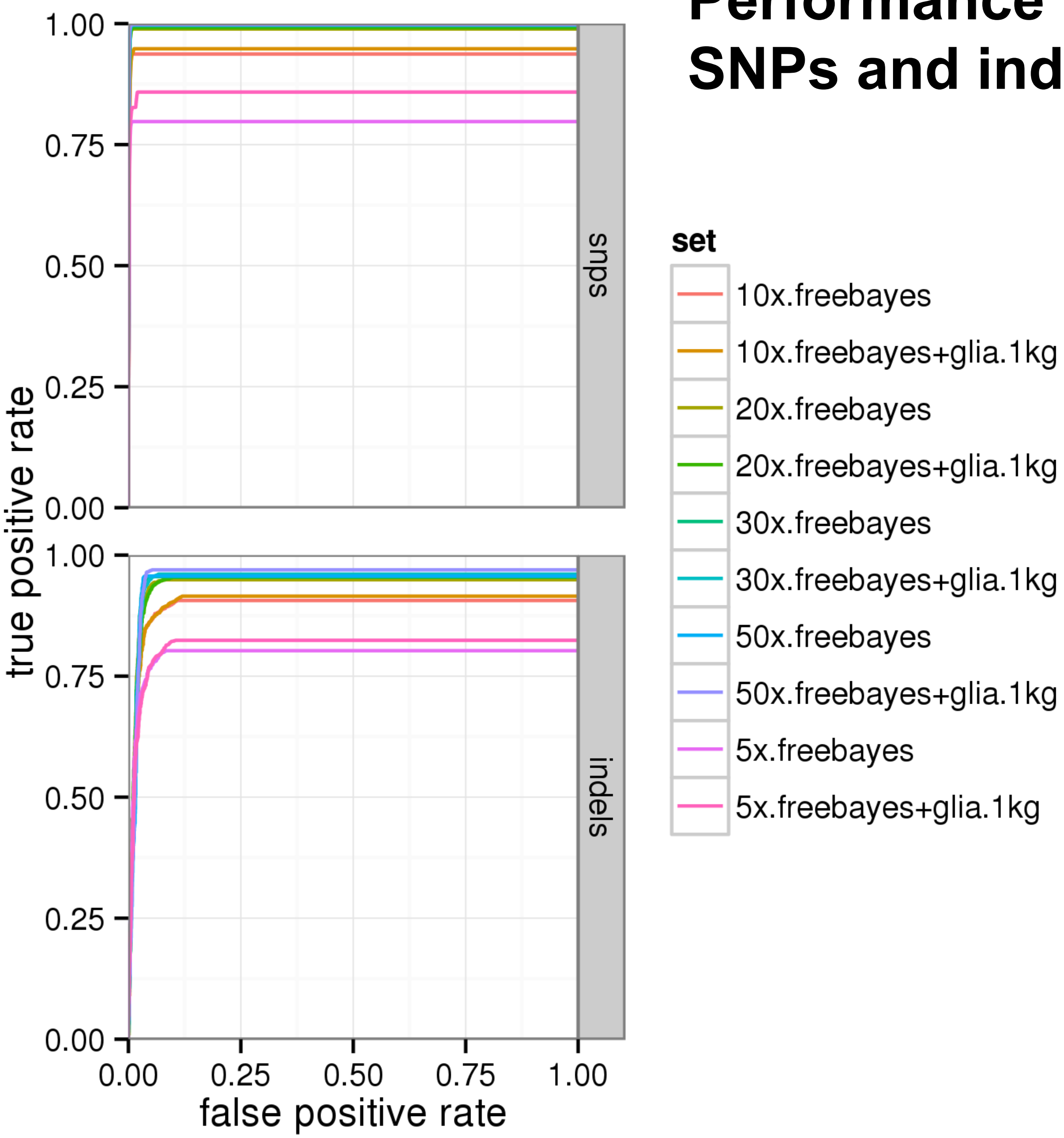
Genotyping large deletions



Input SVs were generated by DELLY on deep, PCR-free samples used for validation in the 1000 Genomes Project.

When using this set as our reference, we can regenotype around 70% of such events in low-coverage samples.

Performance using 1000G phase 3 SNPs and indels >1% frequency



Deep-coverage 100bp Illumina data on NA12878 was downsampled to 5, 10, 20, 30, and 50-fold. Calling by both freebayes and freebayes+glia (realigning to 1000G variants >1% MAF), and comparing the results to the **Genome In a Bottle truth set** demonstrates marked improvement in sensitivity, particularly at low-coverage.

depth	snp AUC diff	indel AUC diff
5	6.02%	1.87%
10	1.07%	0.78%
20	0.26%	0.37%
30	0.08%	0.40%
50	0.02%	1.2%

Questions?

…