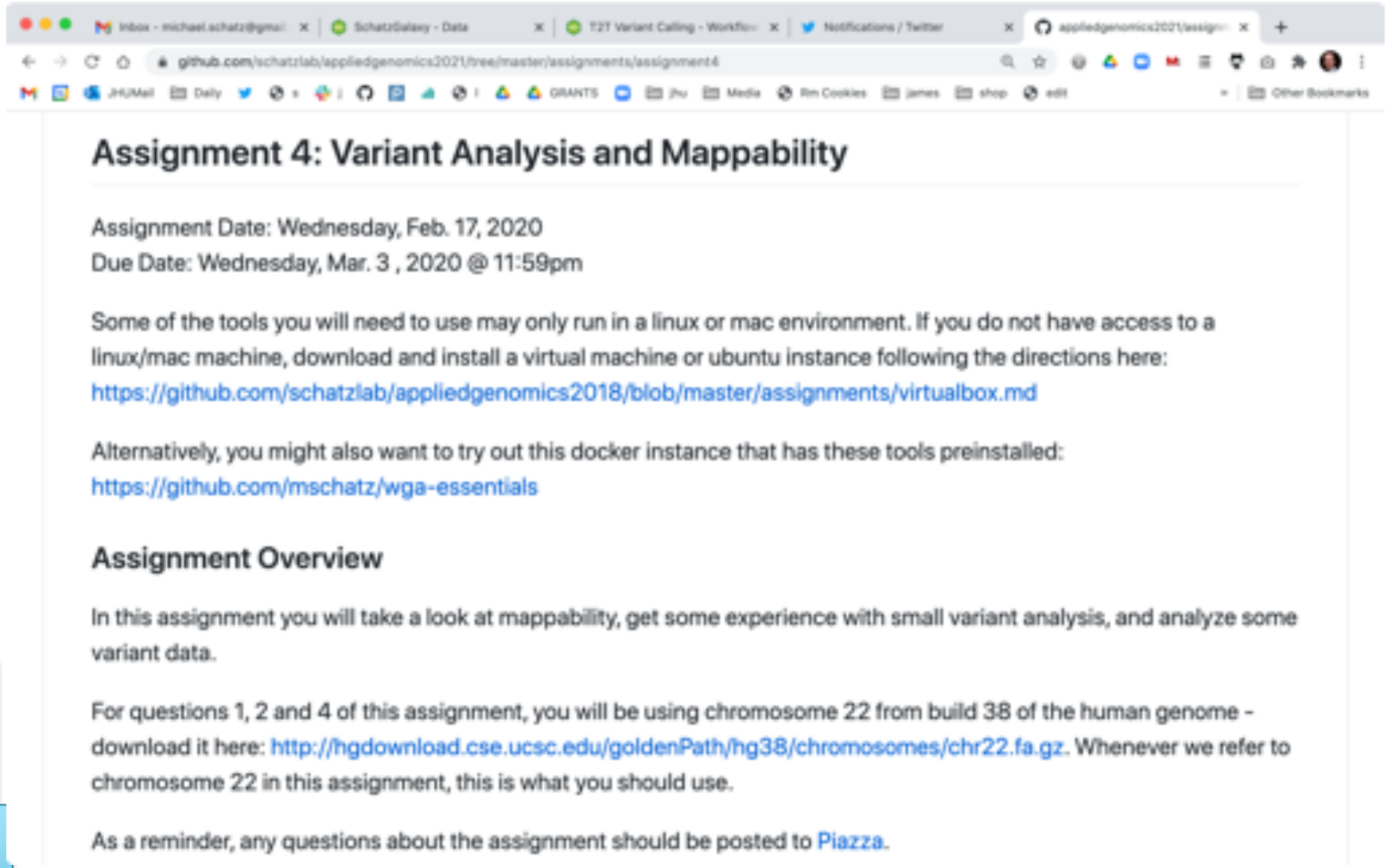
Variant Calling (part 2)

Michael Schatz

March 1, 2021

Lecture 11: Computational Biomedical Research

Assignment 4: Variant Analysis & Mappability
Due March 3 @ 11:59pm



Assignment 4: Variant Analysis and Mappability

Assignment Date: Wednesday, Feb. 17, 2020
Due Date: Wednesday, Mar. 3 , 2020 @ 11:59pm

Some of the tools you will need to use may only run in a linux or mac environment. If you do not have access to a linux/mac machine, download and install a virtual machine or ubuntu instance following the directions here: https://github.com/schatzlab/appliedgenomics2018/blob/master/assignments/virtualbox.md

Alternatively, you might also want to try out this docker instance that has these tools preinstalled: https://github.com/mschatz/wga-essentials

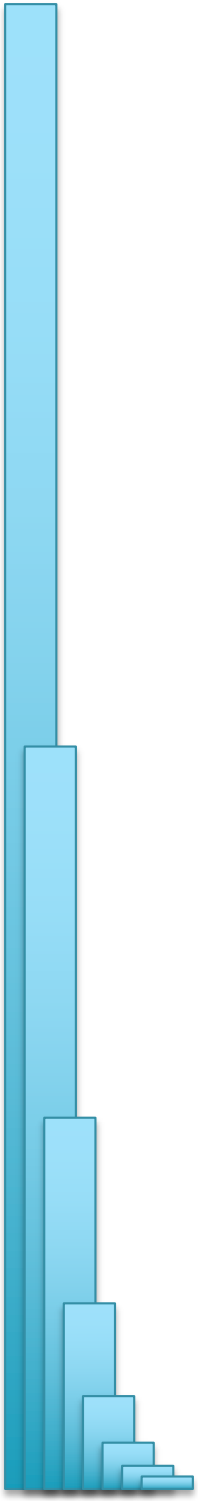
Assignment Overview

In this assignment you will take a look at mappability, get some experience with small variant analysis, and analyze some variant data.

For questions 1, 2 and 4 of this assignment, you will be using chromosome 22 from build 38 of the human genome - download it here: http://hgdownload.cse.ucsc.edu/goldenPath/hg38/chromosomes/chr22.fa.gz. Whenever we refer to chromosome 22 in this assignment, this is what you should use.

As a reminder, any questions about the assignment should be posted to Piazza.

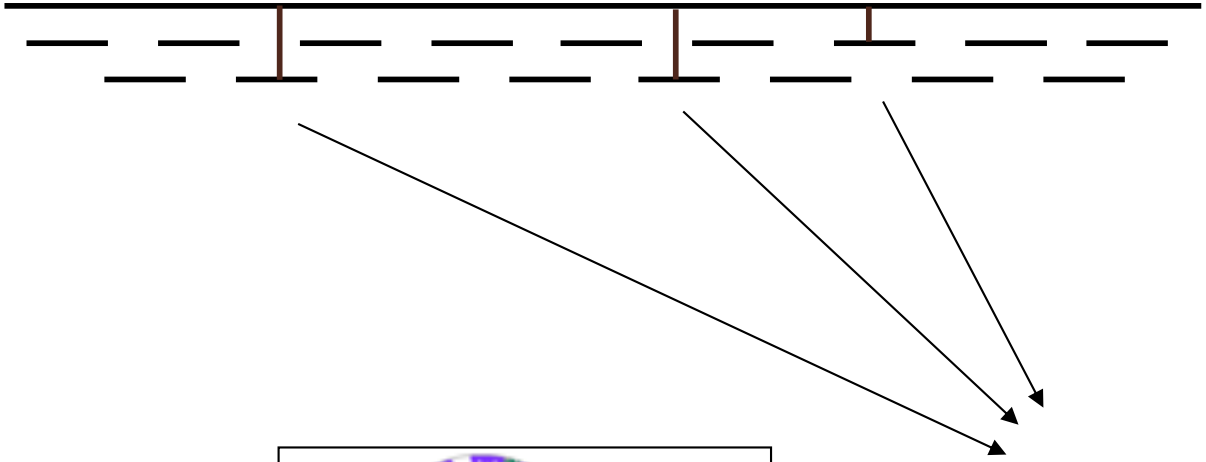
https://github.com/schatzlab/appliedgenomics2021

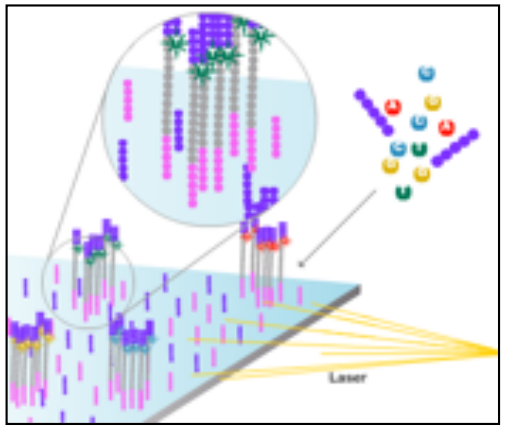


Part 0. Read Mapping

Personal Genomics

How does your genome compare to the reference?





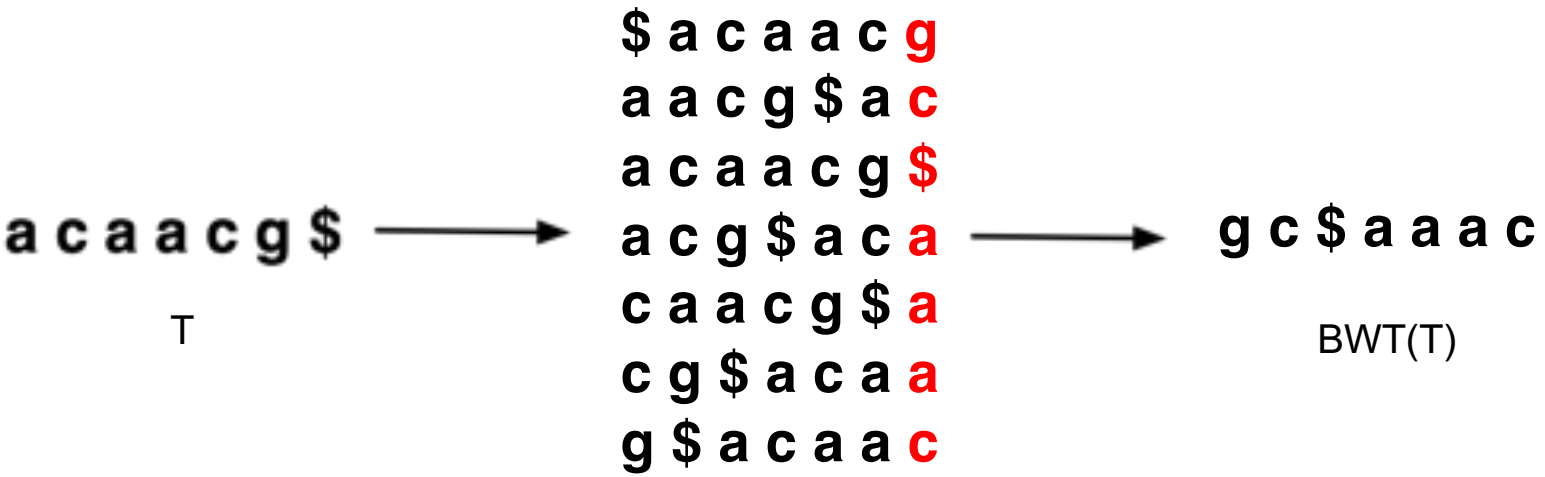
Heart Disease

Cancer

Presidential smile

Burrows-Wheeler Transform

- Permutation of the characters in a text

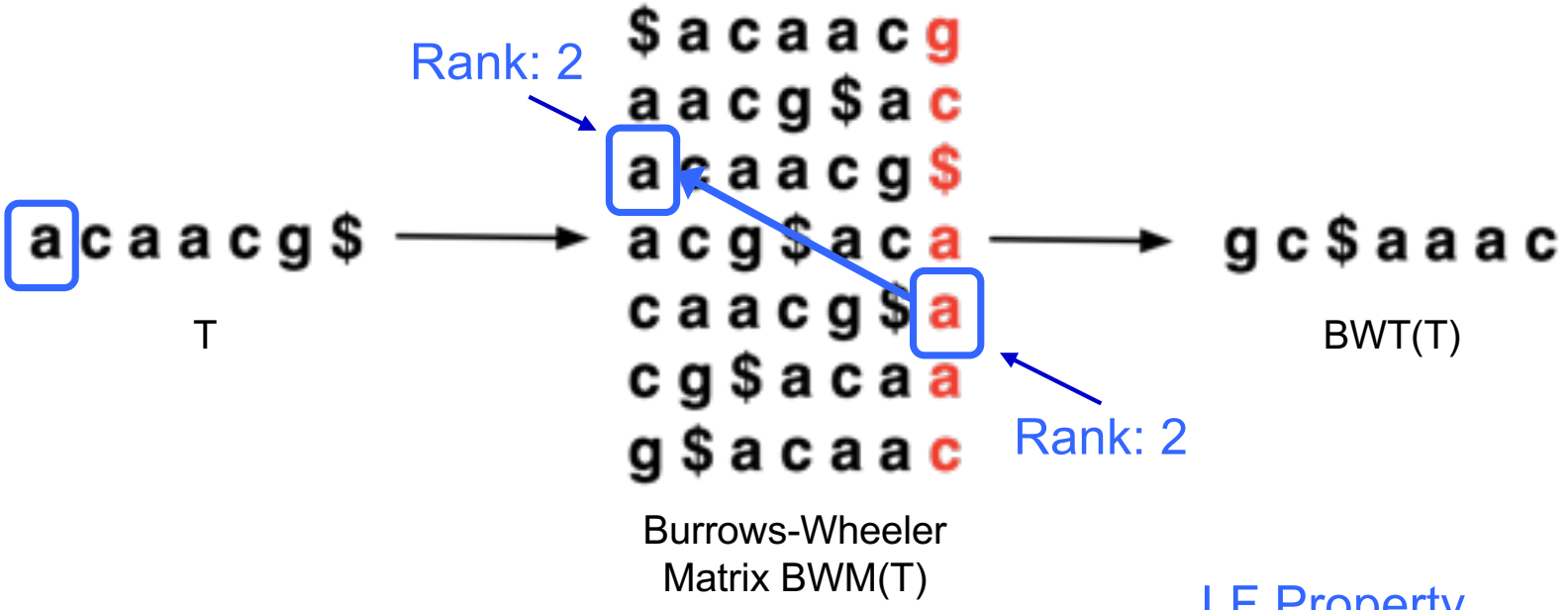


BWT(T) is the index for T

A block sorting lossless data compression algorithm.
Burrows M, Wheeler DJ (1994) *Digital Equipment Corporation.* Technical Report 124

Burrows-Wheeler Transform

- Reversible permutation of the characters in a text



LF Property implicitly encodes Suffix Array

- BWT(T) is the index for T

A block sorting lossless data compression algorithm.
Burrows M, Wheeler DJ (1994) *Digital Equipment Corporation.* Technical Report 124

Burrows-Wheeler Transform

- Recreating T from BWT(T)
  - Start in the first row and apply **LF** repeatedly, accumulating predecessors along the way

[Decode this BWT string: ACTGA$TTA]

Run Length Encoding

ref[614]:

```
It_was_the_best_of_times,_it_was_the_worst_of_times,_it_was_the_age_
of_wisdom,_it_was_the_age_of_foolishness,_it_was_the_epoch_of_belief
,_it_was_the_epoch_of_incredulity,_it_was_the_season_of_Light,_it_wa
s_the_season_of_Darkness,_it_was_the_spring_of_hope,_it_was_the_wint
er_of_despair,_we_had_everything_before_us,_we_had_nothing_before_us
,_we_were_all_going_direct_to_Heaven,_we_were_all_going_direct_the_o
ther_way_-_in_short,_the_period_was_so_far_like_the_present_period,_
that_some_of_its_noisiest_authorities_insisted_on_its_being_received
,_for_good_or_for_evil,_in_the_superlative_degree_of_comparison_only.$
```

rle(bwt)[464]:

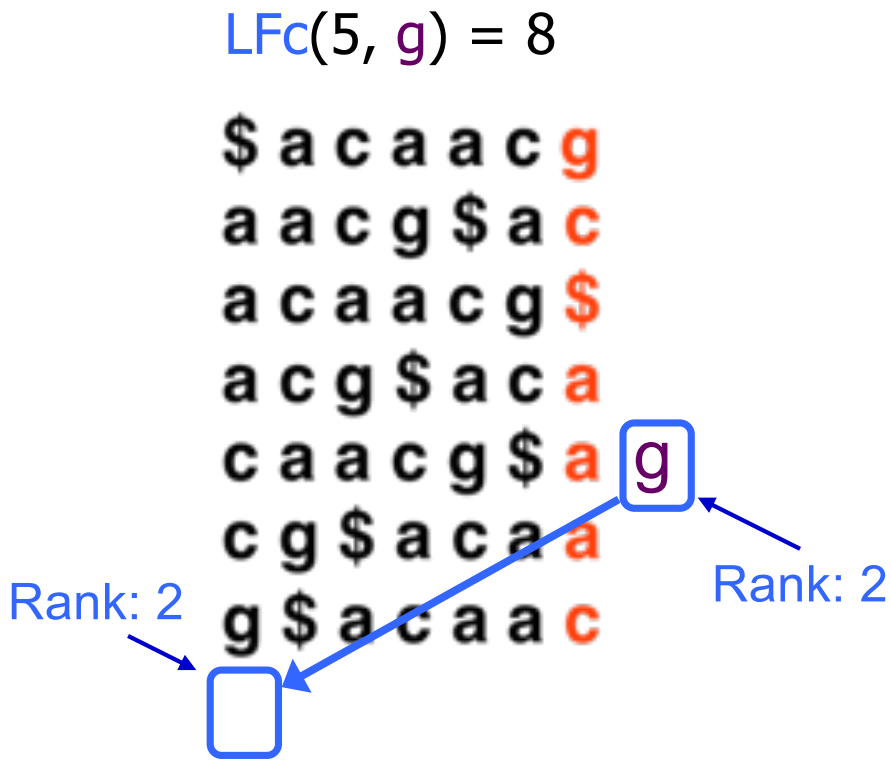
```
.dlms2ftysesdtrsns_y_2$_yfofe4tg2sfefefg2e2drofr,l2re2f-,fs,9nfrsdn2
hereghet2edndete2ge2nste2,s5t,es3ns2f2te2dt10r,4e3feh2_2p_2fpDw11e2h
l_ew_5eo2_ne3oa2eo2_4seph2r2hvh2w2egmgh7kr2w2h2s2Hr3vtr2ib2dbcbvs_2t
hw2p3vm2irdn2ib_2eo12_4e2n6a2i_3ec2_2t18s_tsgltsLlvt2_3h2o2re_wr2ad2
wlors_9r_2lteiril2re_oua2no2i2oeo4i3hki6o_2ieitsp2ioi_12g2nodsc_s3_g
fhf_f3hwh_nsmo_2ue2_sio3ae4o2_i2cgp2e2aoaeo2e2s2eu2teta11i_2ei_in_2a
2ie_e3rei_
```

Saved 614-464 = 150 bytes (24%) with zero loss of information!

Common to save 50% to 90% on real world files with bzip2

BWT Exact Matching

- **LFc**(r, c) does the same thing as **LF**(r) but it ignores r's actual final character and "pretends" it's c:

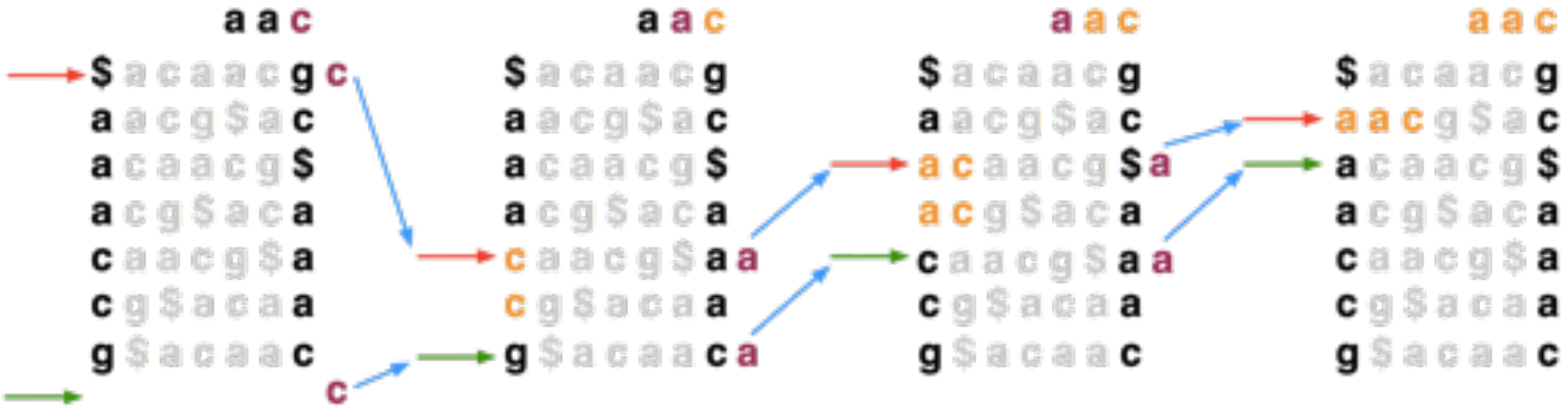


BWT Exact Matching

- Start with a range, (**top**, **bot**) encompassing all rows and repeatedly apply **LFc**:

  top = LFc(top, qc); bot = LFc(bot, qc)

  qc = the next character to the left in the query



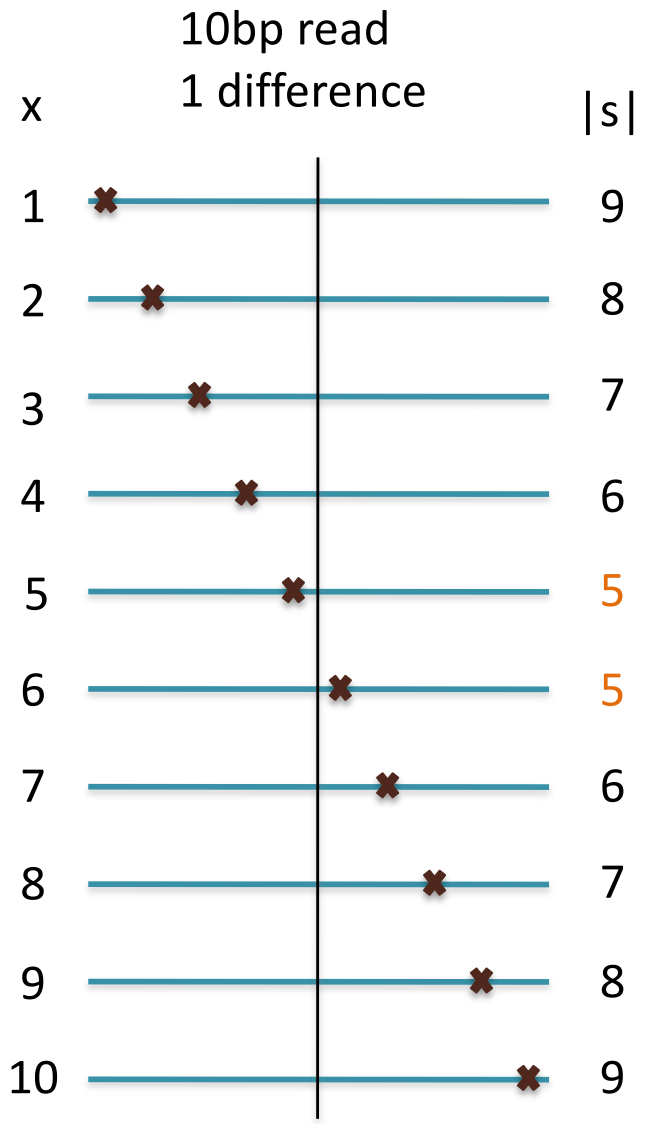
Ferragina P, Manzini G: Opportunistic data structures with applications. *FOCS. IEEE Computer Society; 2000.*

[Search for TTA this BWT string: ACTGA$TTA]

Seed-and-Extend Alignment

Theorem: An alignment of a sequence of length *m* with at most *k* differences ***must*** contain an exact match at least $s=m/(k+1)$ bp long

(*Baeza-Yates* and Perleberg, 1996)

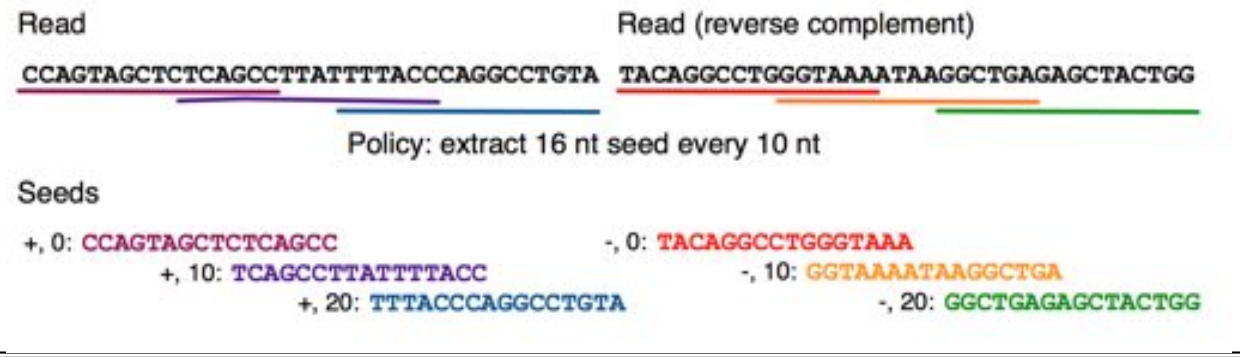


- Proof: Pigeonhole principle
  - 1 pigeon can't fill 2 holes

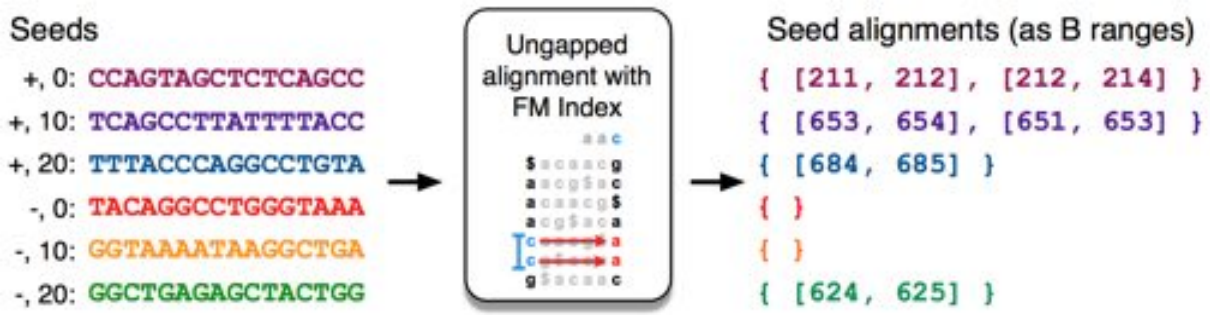
- Seed-and-extend search
  - Use an index to rapidly find short exact alignments to seed longer in-exact alignments
    - BLAST, MUMmer, Bowtie, BWA, SOAP, …
  - Specificity of the depends on seed length
    - Guaranteed sensitivity for k differences
    - Also finds some (but not all) lower quality alignments <- heuristic

Algorithm Overview

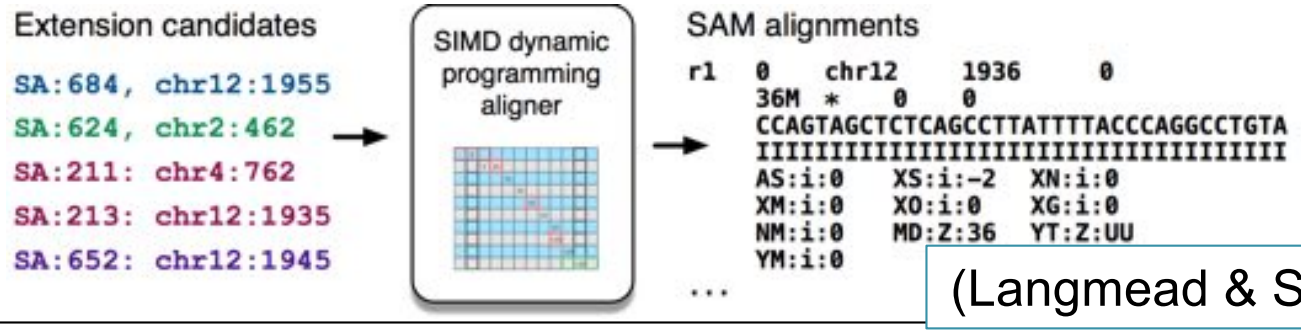
1. Split read into segments



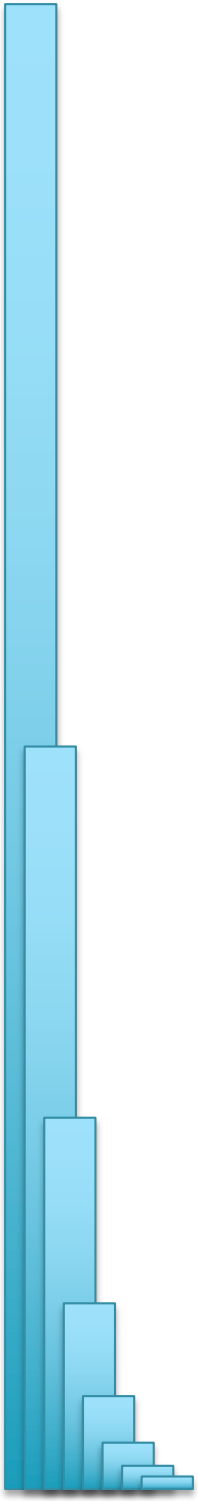
2. Lookup each segment and prioritize



3. Evaluate end-to-end match



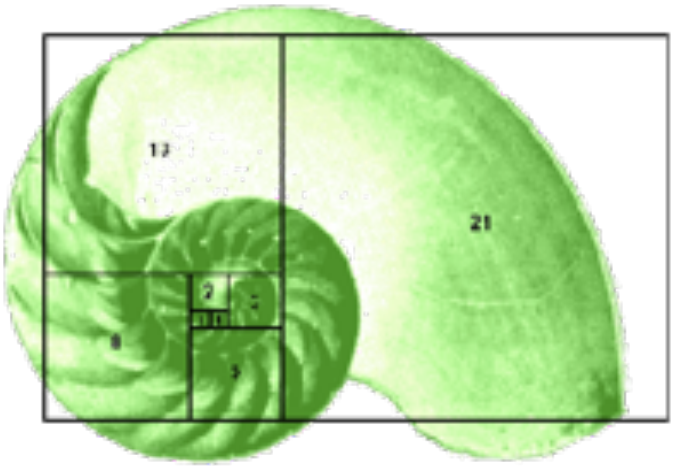
(Langmead & Salzberg, 2012)

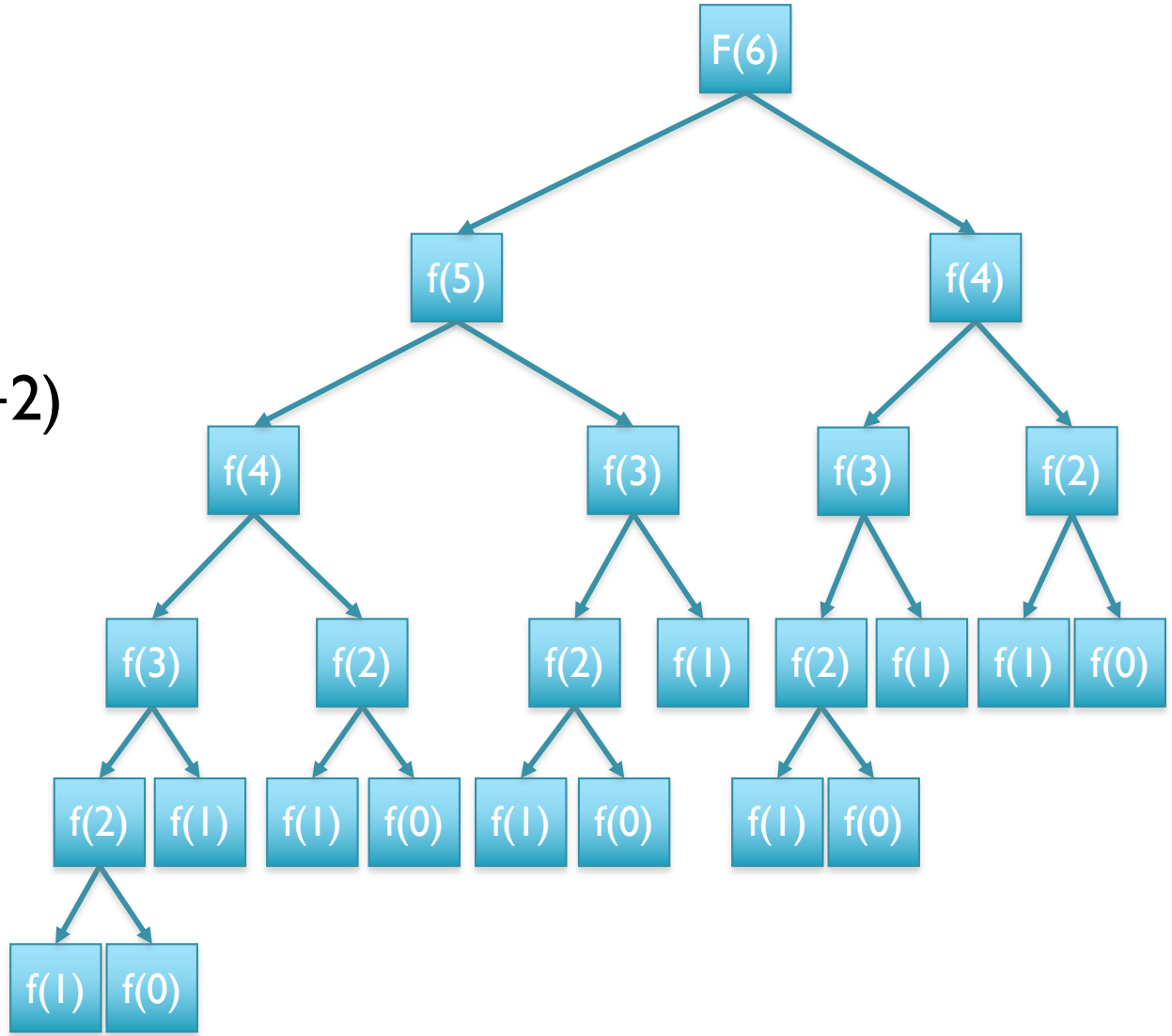


Part 2: Dynamic Programming

Fibonacci Sequence

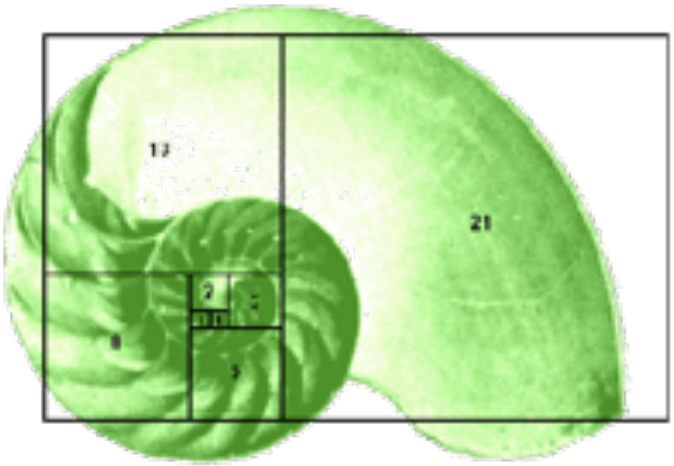
```
def fib(n):
  if n == 0 or n == 1:
     return n
  else:
  return fib(n−1) + fib(n−2)
```

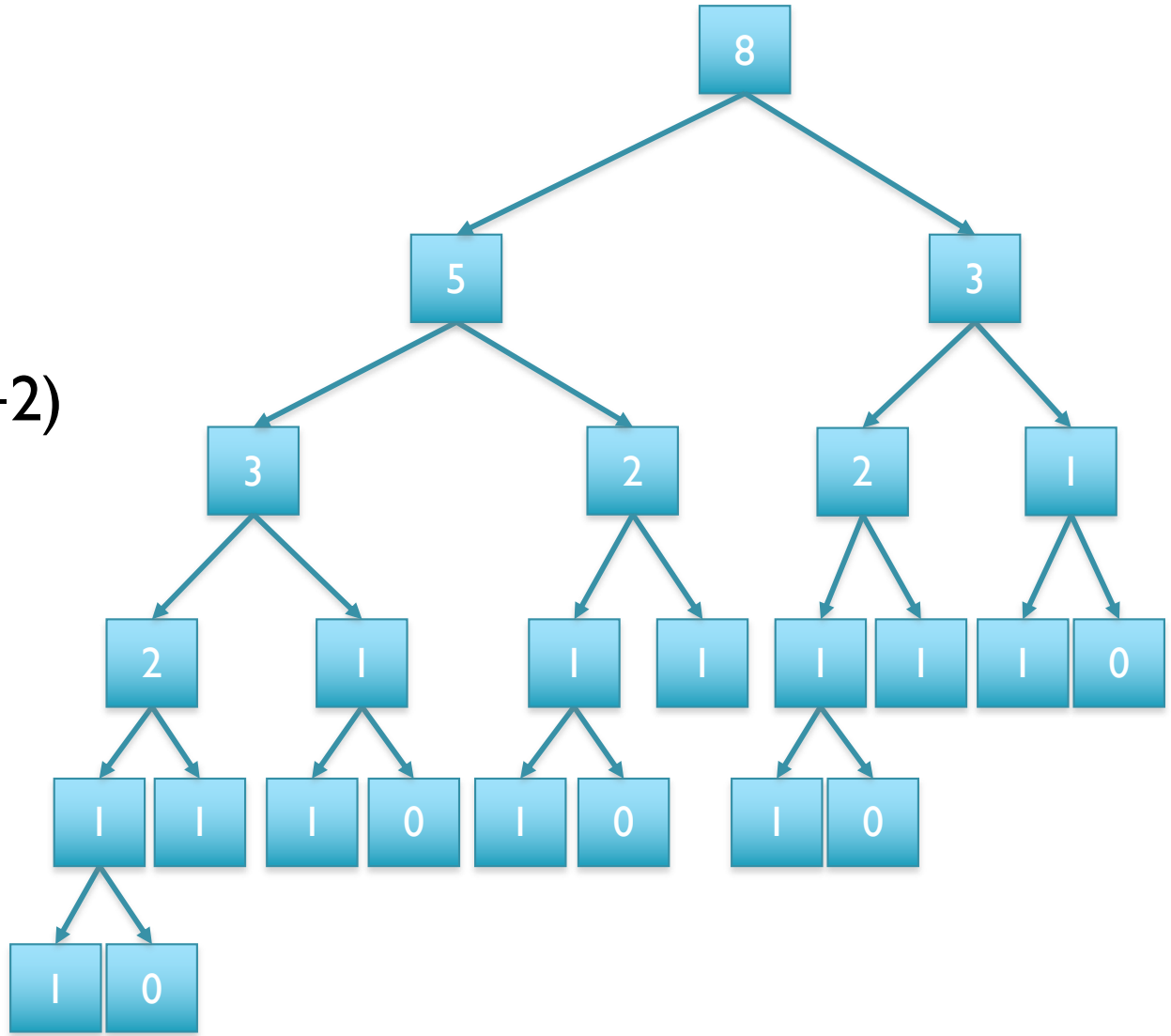




Fibonacci Sequence

```
def fib(n):
  if n == 0 or n == 1:
     return n
  else:
  return fib(n−1) + fib(n−2)
```



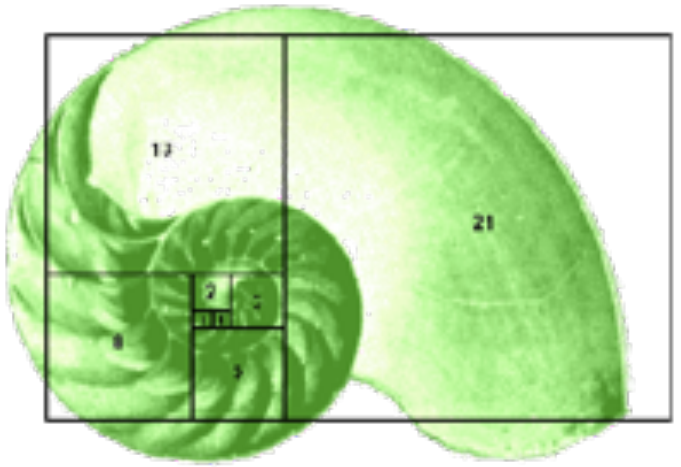


[What is the running time?]

Bottom-up Fibonacci Sequence

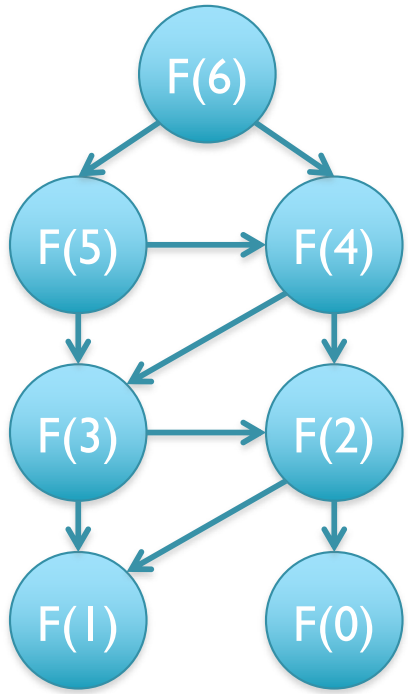
```
def fib(n):
  table = [0] * (n+1)
  table[0] = 0
  table[1] = 1
  for i in range(2,n+1):
    table[i] = table[i−2] + table[i−1]
return table[n]
```

[What is the running time?]



Dynamic Programming

- General approach for solving (some) complex problems
  - When applicable, the method takes far less time than naive methods.
    - Polynomial time (O(n) or $O(n^2)$) instead of exponential time ($O(2^n)$ or $O(3^n)$)
- Requirements:
  - **Overlapping subproblems**
  - **Optimal substructure**
- Applications:
  - Fibonacci
  - Longest Increasing Subsequence (Bonus Slides!)
  - Sequence alignment, Dynamic Time Warp, Viterbi



- Not applicable:
  - Traveling salesman problem, Clique finding, Subgraph isomorphism, …
  - The cheapest flight from airport A to airport B involves a single connection through airport C, but the cheapest flight from airport A to airport C involves a connection through some other airport D.

In-exact alignment

- Where is GATTACA *approximately* in the human genome?
  - And how do we efficiently find them?

- It depends…
  - Define 'approximately'
    - Hamming Distance, Edit distance, or Sequence Similarity
    - Ungapped vs Gapped vs Affine Gaps
    - Global vs Local
    - All positions or the single 'best'?
  - Efficiency depends on the data characteristics & goals
    - Smith-Waterman: Exhaustive search for optimal alignments
    - BLAST: Hash-table based homology searches
    - Bowtie: BWT alignment for short read mapping

Similarity metrics

- Hamming distance
  - Count the number of substitutions to transform one string into another

```
MIKESCHATZ
||X||XXXX|
MICESHATZZ
     5
```

- Edit distance
  - The minimum number of substitutions, insertions, or deletions to transform one string into another

```
MIKESCHAT-Z
||X||X|||X|
MICES-HATZZ
     3
```

Edit Distance Example

AGCACACA → ACACACTA in 4 steps

AGCACACA → (1. change G to C)
ACCACACA → (2. delete C)
ACACACA → (3. change A to T)
ACACACT → (4. insert A after T)
ACACACTA → done

[Is this the best we can do?]

Edit Distance Example

AGCACACA → ACACACTA in 3 steps

AGCACACA → (1. change G to C)
ACCACACA → (2. delete C)
ACACACA → (3. insert T after 3rd C)
ACACACTA → done

[Is this the best we can do?]

Reverse Engineering Edit Distance

D(AGCACACA, ACACACTA) = ?

Imagine we already have the optimal alignment of the strings, the last column can only be 1 of 3 options:

…**M**	…**I**	…**D**
…A	…-	…A
…A	…A	…-

The optimal alignment of last two columns is then 1 of 9 possibilities

…**MM**	…**IM**	…**DM**	…**MI**	…**II**	…**DI**	…**MD**	…**ID**	…**DD**
…CA	…-A	…CA	…A-	…--	…A-	…CA	…-A	…CA
…TA	…TA	…-A	…TA	…TA	…-A	…A-	…A-	…--

The optimal alignment of the last three columns is then 1 of 27 possibilities…

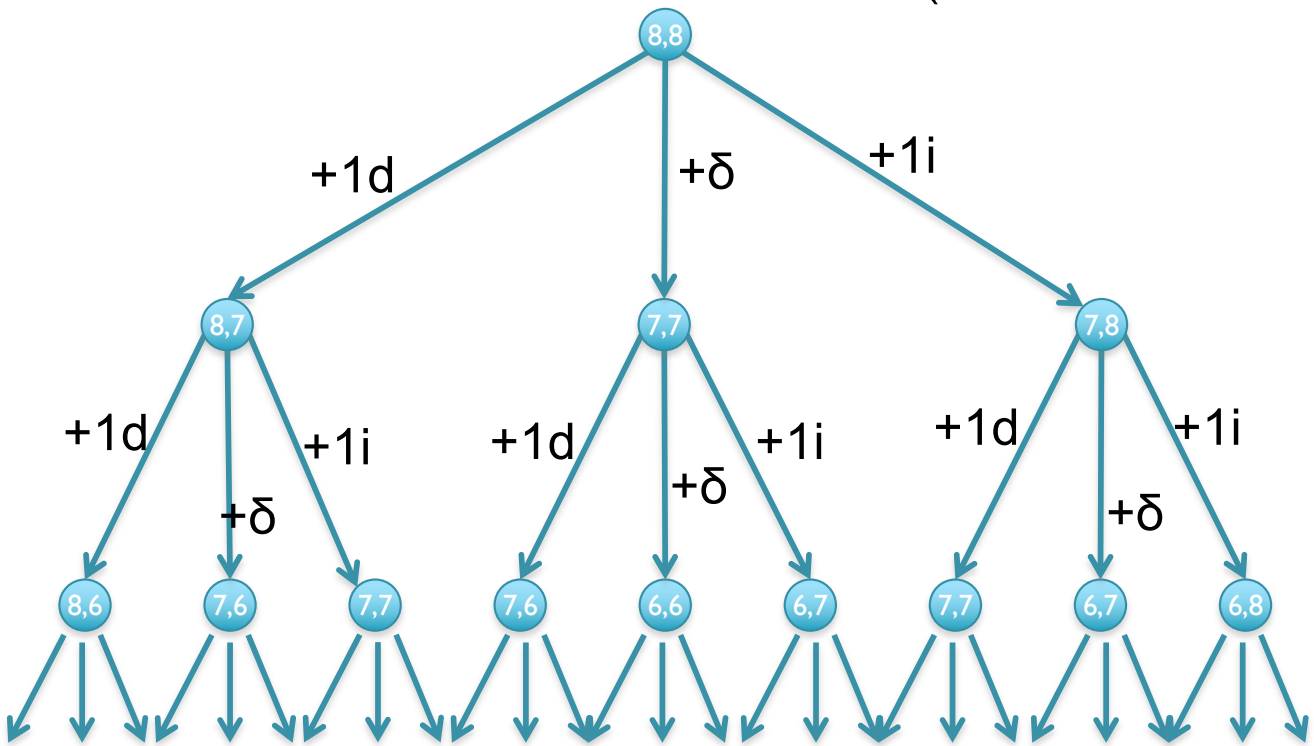
…**M**…	…**I**…	…**D**…
…X…	…-…	…X…
…Y…	…Y…	…-…

Eventually spell out every possible sequence of {I,M,D}

Recursive solution

- Computation of D is a recursive process.
  - At each step, we only allow matches, substitutions, and indels
  - D(i,j) in terms of D(i', j') for i' ≤ i and j' ≤ j.

D(AGCACACA, ACACACTA) = min{D(AGCACACA, ACACACT) + 1,
D(AGCACAC, ACACACTA) + 1,
D(AGCACAC, ACACACT) +δ(A, A)}



[What is the running time?]

Dynamic Programming

- We could code this as a recursive function call...

  ...with an exponential number of function evaluations

- There are only (n+1) x (m+1) pairs i and j
  - We are evaluating D(i,j) multiple times

- Compute D(i,j) bottom up.
  - Start with smallest (i,j) = (1,1).
  - Store the intermediate results in a table.
    - Compute D(i,j) *after* D(i-1,j), D(i,j-1), and D(i-1,j-1)

Recurrence Relation for D

Find the edit distance (minimum number of operations to convert one string into another) in O(mn) time

- Base conditions:
  - D(i,0) = i, for all i = 0,...,n
  - D(0,j) = j, for all j = 0,...,m

- For i > 0, j > 0:

$$
\begin{aligned}
D(i,j) = \min \{ & \\
& D(i-1,j) + 1, && \text{// align 0 chars from S, 1 from T} \\
& D(i,j-1) + 1, && \text{// align 1 chars from S, 0 from T} \\
& D(i-1,j-1) + \delta(S(i),T(j)) && \text{// align 1+1 chars} \\
\} &
\end{aligned}
$$

[Why do we want the min?]

Dynamic Programming Matrix

		A	C	A	C	A	C	T	A
	0	1	2	3	4	5	6	7	8
A	1								
G	2								
C	3								
A	4								
C	5								
A	6								
C	7								
A	8								

[What does the initialization mean?]

Dynamic Programming Matrix

		A	C	A	C	A	C	T	A
	0	1	2	3	4	5	6	7	8
A	1	0							
G	2								
C	3								
A	4								
C	5								
A	6								
C	7								
A	8								

$$D[A,A] = \min\{D[A,]+1,\ D[,A]+1,\ D[,]+\delta(A,A)\}$$

Dynamic Programming Matrix

		A	C	A	C	A	C	T	A
	0	1	2	3	4	5	6	7	8
A	1	0	1						
G	2								
C	3								
A	4								
C	5								
A	6								
C	7								
A	8								

$$D[A,AC] = \min\{D[A,A]+1,\ D[,AC]+1,\ D[,A]+\delta(A,C)\}$$

Dynamic Programming Matrix

		A	C	A	C	A	C	T	A
	0	1	2	3	4	5	6	7	8
A	1	0	1	2					
G	2								
C	3								
A	4								
C	5								
A	6								
C	7								
A	8								

D[A,ACA] = min{D[A,AC]+1, D[,ACA]+1, D[,AC]+δ(A,A)}

Dynamic Programming Matrix

		A	C	A	C	A	C	T	A
	0	1	2	3	4	5	6	7	8
A	1	0	1	2	3	4	5	6	7
G	2								
C	3								
A	4								
C	5								
A	6								
C	7								
A	8								

D[A,ACACACTA] = 7

```
-------A
*******|
ACACACTA
```

[What about the other A?]

Dynamic Programming Matrix

		A	C	A	C	A	C	T	A
	0	1	2	3	4	5	6	7	8
A	1	0	1	2	3	4	5	6	7
G	2	1	1	2	3	4	5	6	7
C	3								
A	4								
C	5								
A	6								
C	7								
A	8								

D[AG,ACACACTA] = 7

```
----AG--
****|***
ACACACTA
```

Dynamic Programming Matrix

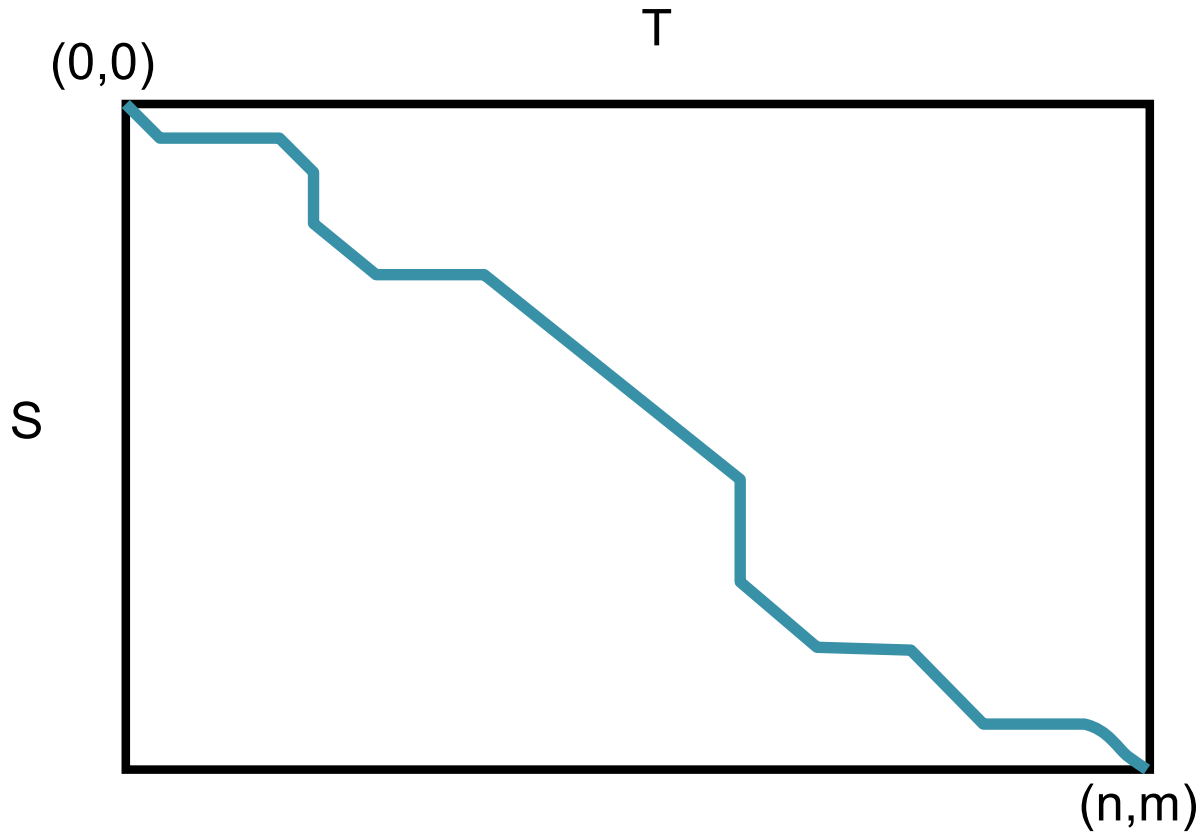
		A	C	A	C	A	C	T	A
	0	1	2	3	4	5	6	7	8
A	1	0	1	2	3	4	5	6	7
G	2	1	1	2	3	4	5	6	7
C	3	2	1	2	2	3	4	5	6
A	4	3	2	1	2	2	3	4	5
C	5	4	3	2	1	2	2	3	4
A	6	5	4	3	2	1	2	3	3
C	7	6	5	4	3	2	1	2	3
A	8	7	6	5	4	3	2	2	2

D[AGCACACA,ACACACTA] = 2

```
AGCACAC-A
| * | | | | | * |
A-CACACTA
```

[Can we do it any better?]

Global Alignment Schematic



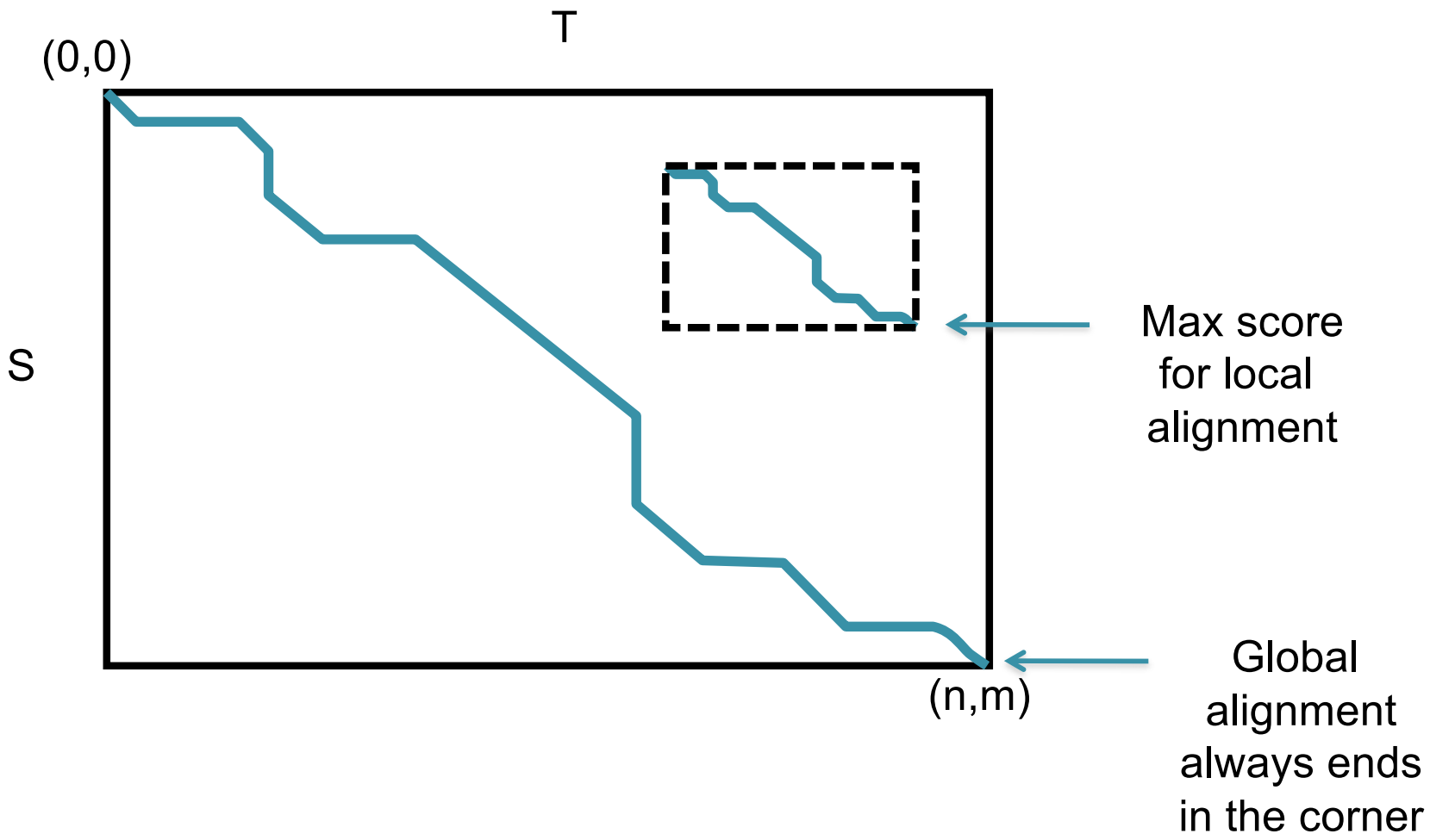
- A high quality alignment will stay close to the diagonal
  - If we are only interested in high quality alignments, we can skip filling in cells that can't possibly lead to a high quality alignment
  - Find the global alignment with at most edit distance d: O(2dn)

Local vs. Global Alignment

- The Global Alignment Problem tries to find the best end-to-end alignment between the two strings
  - Only applicable for very closely related sequences

- The Local Alignment Problem tries to find pairs of **substrings** with highest similarity.
  - Especially important if one string is substantially longer than the other
  - Especially important if there is only a distant evolutionary relationship

Global vs Local Alignment Schematic



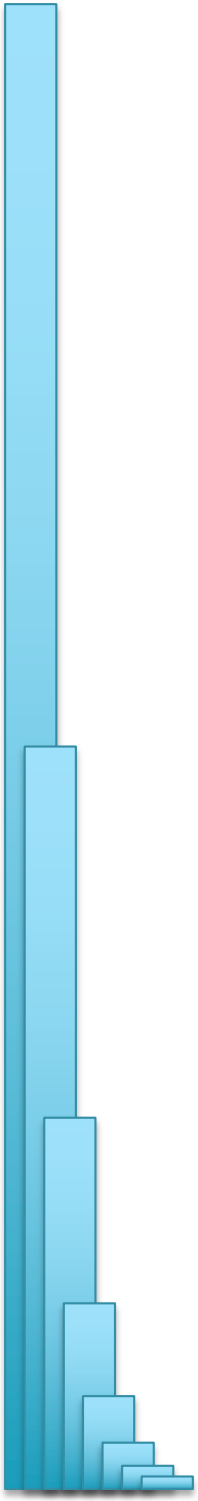
Local vs. Global Alignment (cont' d)

- Global Alignment

```
--T--CC-C-AGT--TATGT-CAGGGGACACG-A-GCATGCAGA-GAC
  |  || |  ||  | | | |||    || | | |  | ||||   |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG-T-CAGAT--C
```

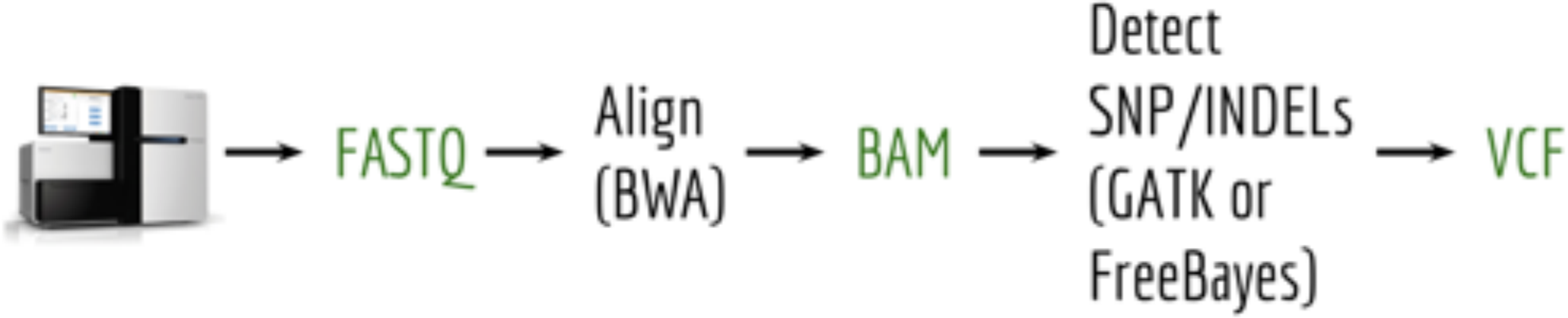
- Local Alignment—better alignment to find conserved segment

```
                tccCAGTTATGTCAGgggacacgagcatgcagagac
                   ||||||||||||
aattgccgccgtcgttttcagCAGTTATGTCAGatc
```

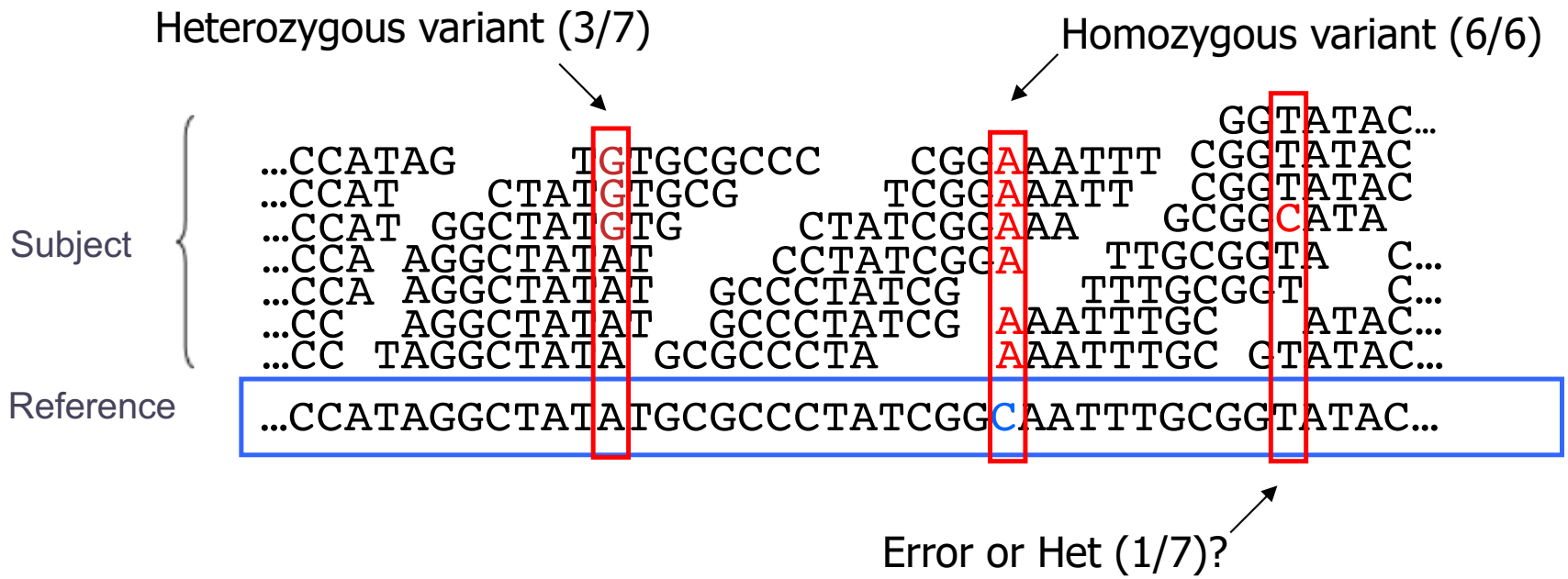


Part 3: Variant Calling

Variant Calling Overview

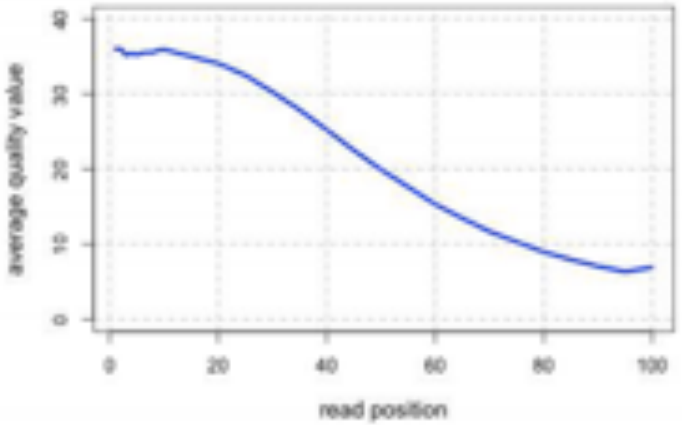


Genotyping Theory



- If there were no sequencing errors, identifying SNPs would be very easy: any time a read disagrees with the reference, it must be a variant!

- Sequencing instruments make mistakes
  - Quality of read decreases over the read length



- A single read differing from the reference is probably just an error, but it becomes more likely to be real as we see it multiple times

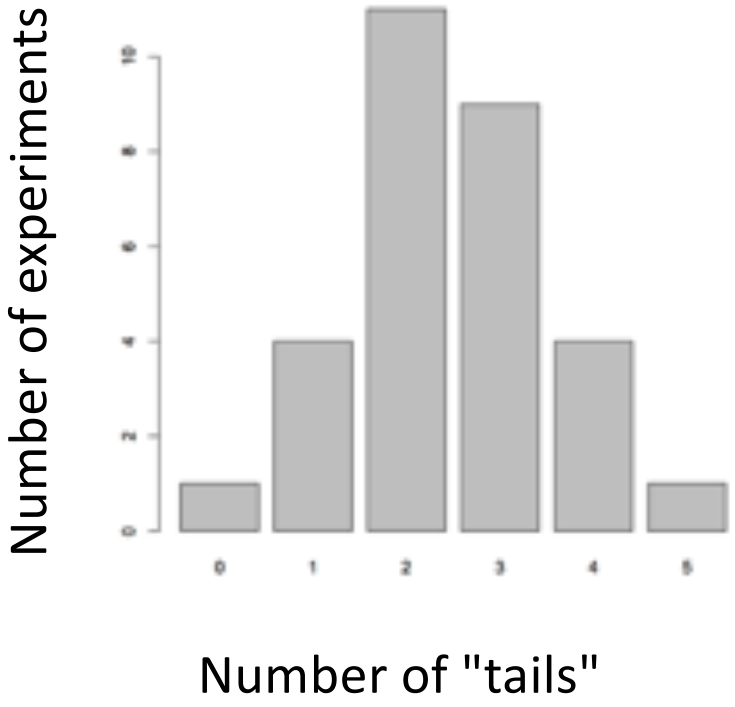
The Binomial Distribution: Adventures in Coin Flipping

P(heads) = 0.5

P(tails) = 0.5

Aaron Quinlan

What is the distribution of tails (alternate alleles) do we expect to see after 5 tosses (sequence reads)?



R code:

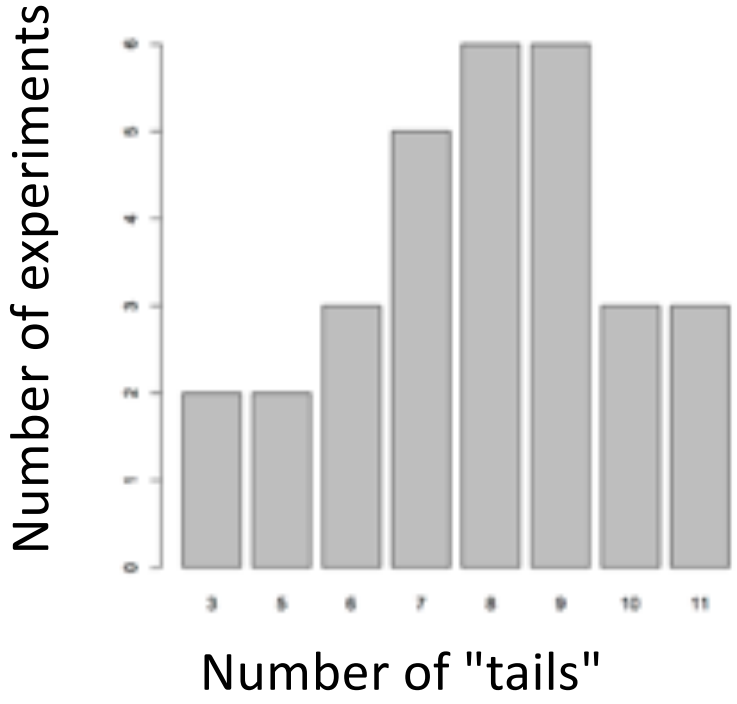
```
barplot(table(rbinom(30, 5, 0.5)))
```

30 experiments (students tossing coins)

5 tosses each

Probability of Tails

What is the distribution of tails (alternate alleles) do we expect to see after 15 tosses (sequence reads)?



R code:

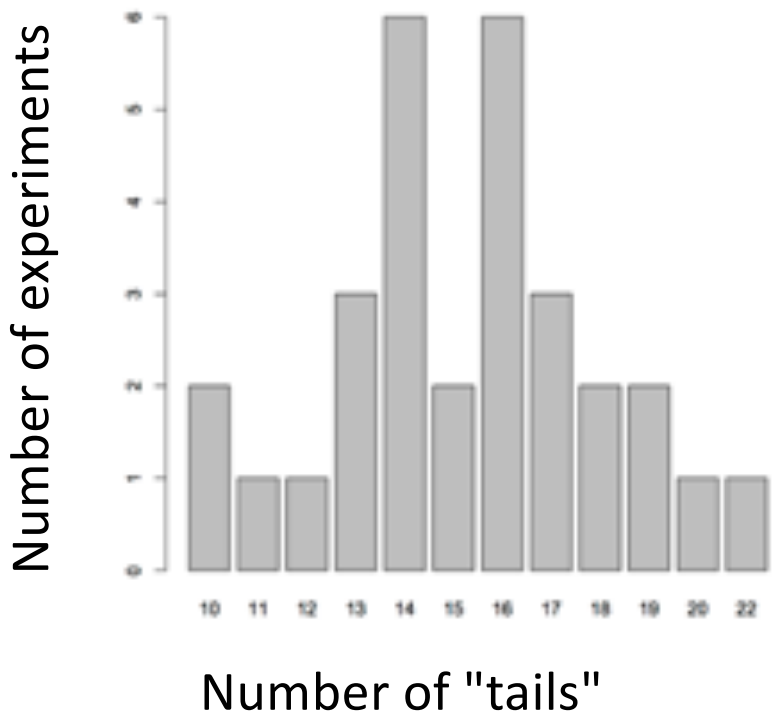
```
barplot(table(rbinom(30, 15, 0.5)))
```

30 experiments (students tossing coins)

15 tosses each

Probability of Tails

What is the distribution of tails (alternate alleles) do we expect to see after 30 tosses (sequence reads)?



R code:

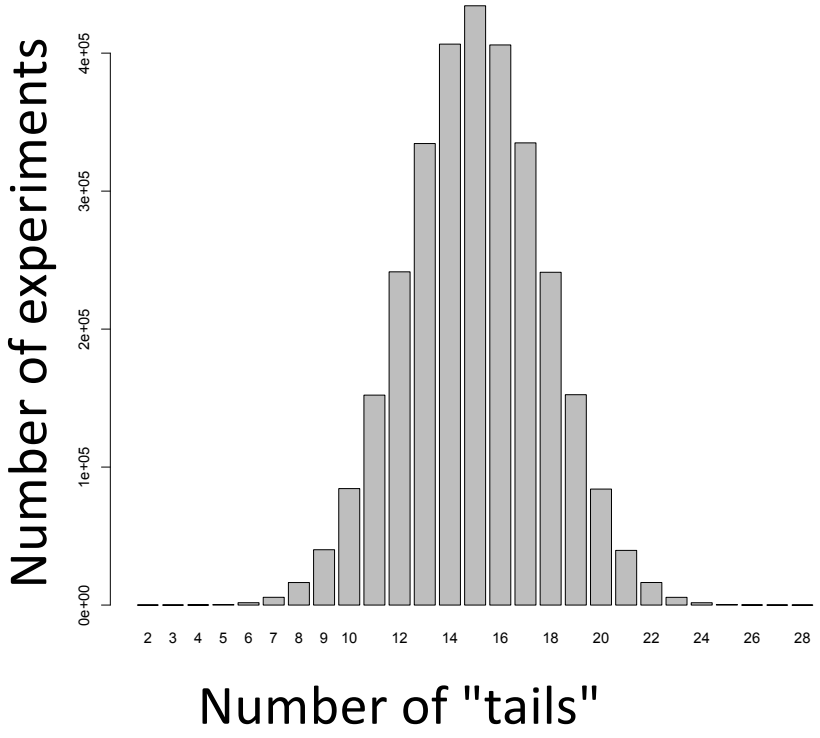
```
barplot(table(rbinom(30, 30, 0.5)))
```

30 experiments (students tossing coins)

30 tosses each

Probability of Tails

What is the distribution of tails (alternate alleles) do we expect to see after 30 tosses (sequence reads)?



R code:

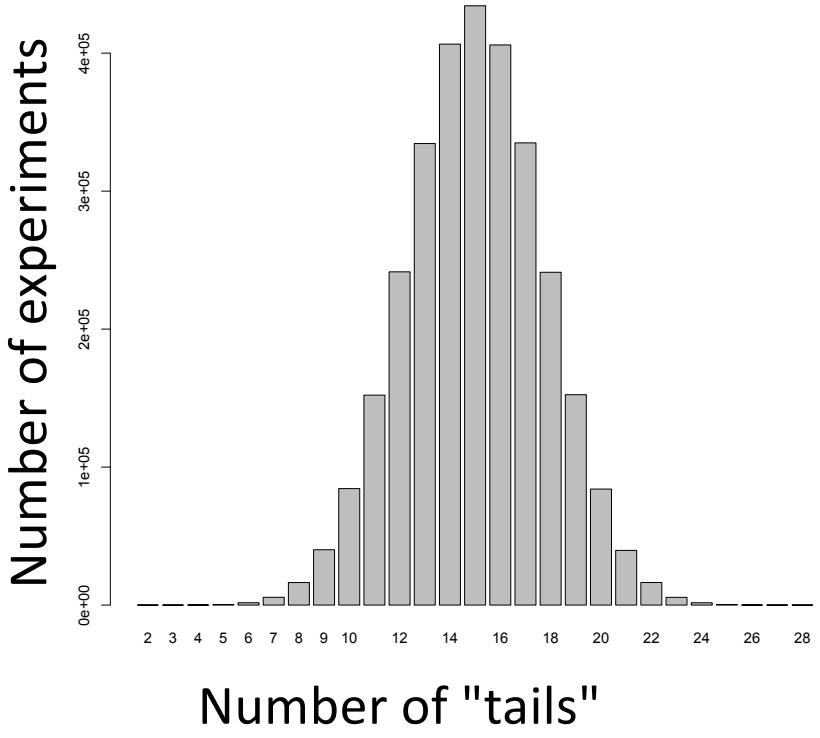
```
barplot(table(rbinom(3e6, 30, 0.5)))
```

3M experiments (students tossing coins)

30 tosses each

Probability of Tails

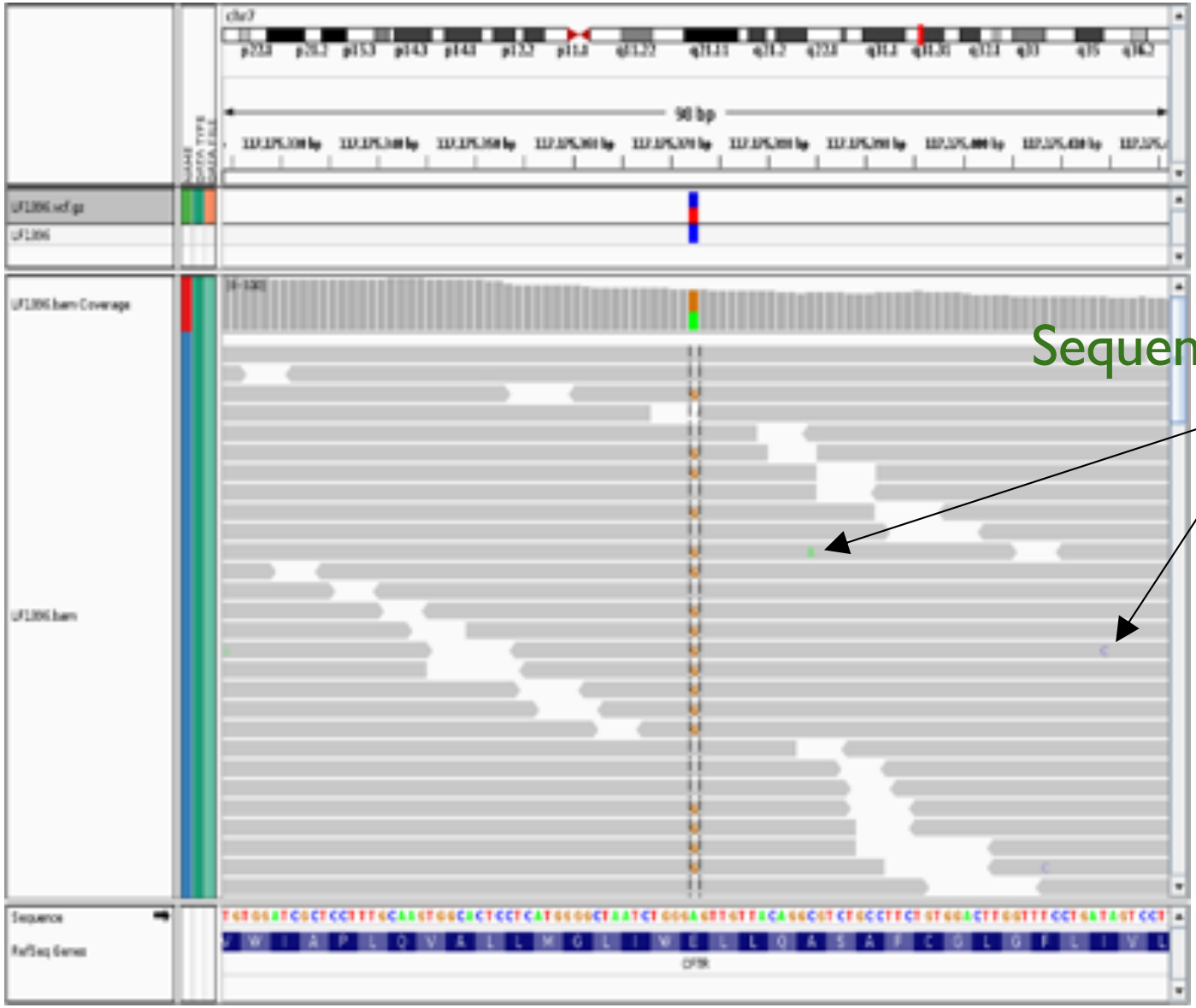
So, with 30 tosses (reads), we are much more likely to see an even mix of alternate and reference alleles at a heterozygous locus in a genome



This is why at least a "30X" (30 fold sequence coverage) genome is recommended: it confers sufficient power to distinguish heterozygous alleles and from mere sequencing errors

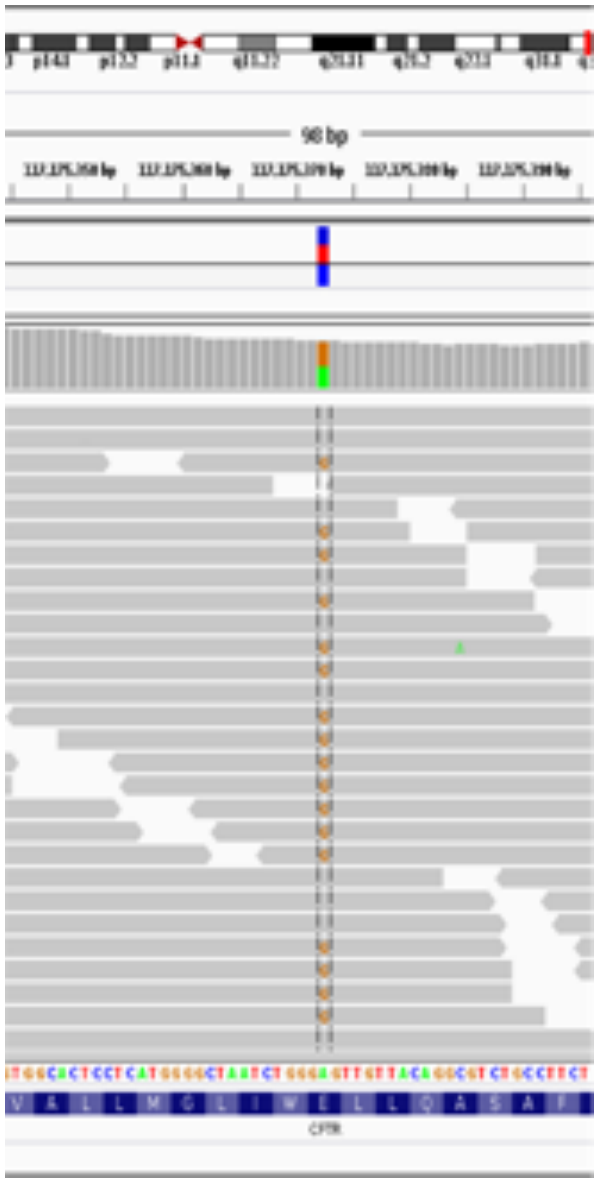
P(3/30 het) <?> P(3/30 err)

Sequencing errors fall out as noise (most of the time)



https://jchoigt.files.wordpress.com/2012/07/igv_e217g_snapshot.png

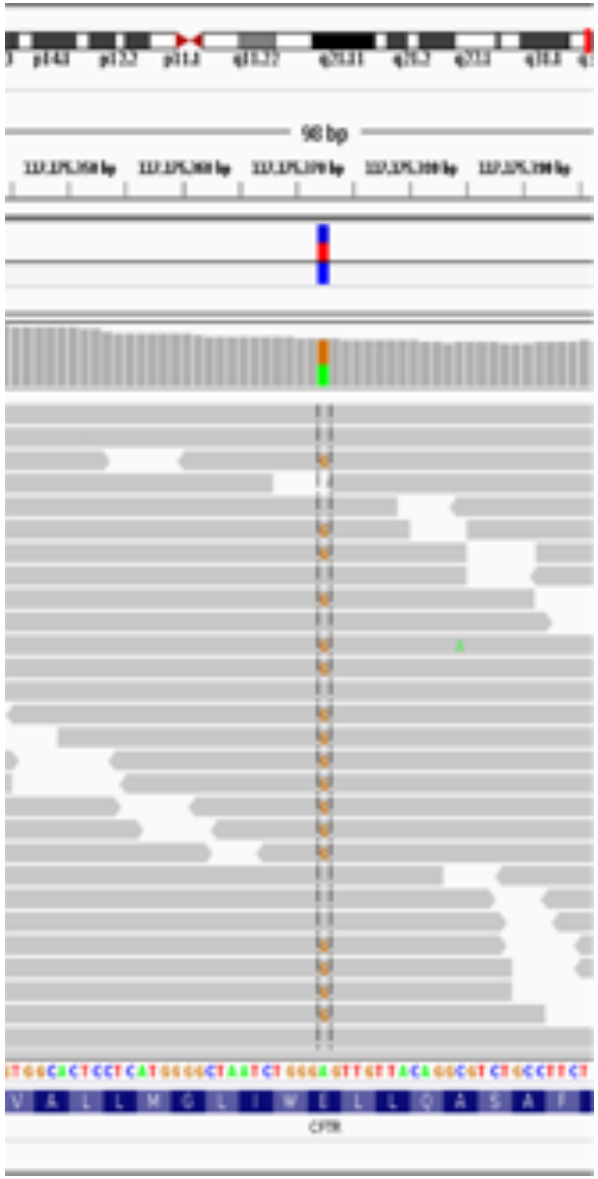
What information is needed to decide if a variant exists?



- Depth of coverage at the locus
- Bases observed at the locus
- The base qualities of each allele
- The strand composition
- Mapping qualities
- Proper pairs?
- Expected polymorphism rate

https://genomebiology.biomedcentral.com/articles/10.1186/gb-2010-11-10-r99

Bayesian SNP calling



$$P(SNP|Data) = \frac{P(Data|SNP) * P(SNP)}{P(Data)}$$

http://www.nature.com/ng/journal/v23/n4/full/ng1299_452.html

PolyBayes: The first statistically rigorous variant detection tool.

letter © 1999 Nature America Inc. • http://genetics.nature.com

A general approach to single-nucleotide polymorphism discovery

Gabor T. Marth[1], Ian Korf[1], Mark D. Yandell[1], Raymond T. Yeh[1], Zhijie Gu[2], Hamideh Zakeri[2], Nathan O. Stitziel[1], LaDeana Hillier[1], Pui-Yan Kwok[2] & Warren R. Gish[1]

Bayesian posterior probability

Base call + Base quality

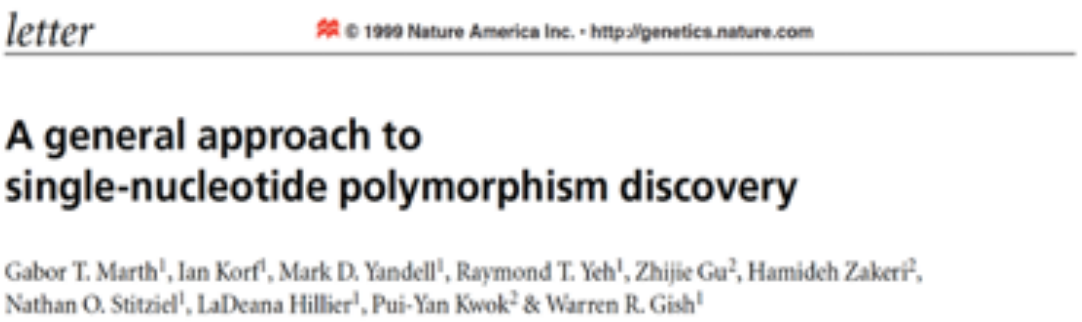
Expected (prior) polymorphism rate

$$P(SNP) = \sum_{all\ variable\ S} \frac{\frac{P(S_1 \mid R_1)}{P_{Prior}(S_1)} \cdot \ldots \cdot \frac{P(S_N \mid R_N)}{P_{Prior}(S_N)} \cdot P_{Prior}(S_1,\ldots,S_N)}{\sum_{S_{i_1} \in [A,C,G,T]} \ldots \sum_{S_{i_N} \in [A,C,G,T]} \frac{P(S_{i_1} \mid R_1)}{P_{Prior}(S_{i_1})} \cdot \ldots \cdot \frac{P(S_{i_N} \mid R_1)}{P_{Prior}(S_{i_N})} \cdot P_{Prior}(S_{i_1},\ldots,S_{i_N})}$$

Probability of observed base composition (should model sequencing error rate)

http://www.nature.com/ng/journal/v23/n4/full/ng1299_452.html

PolyBayes: The first statistically rigorous variant detection tool.



letter © 1999 Nature America Inc. • http://genetics.nature.com

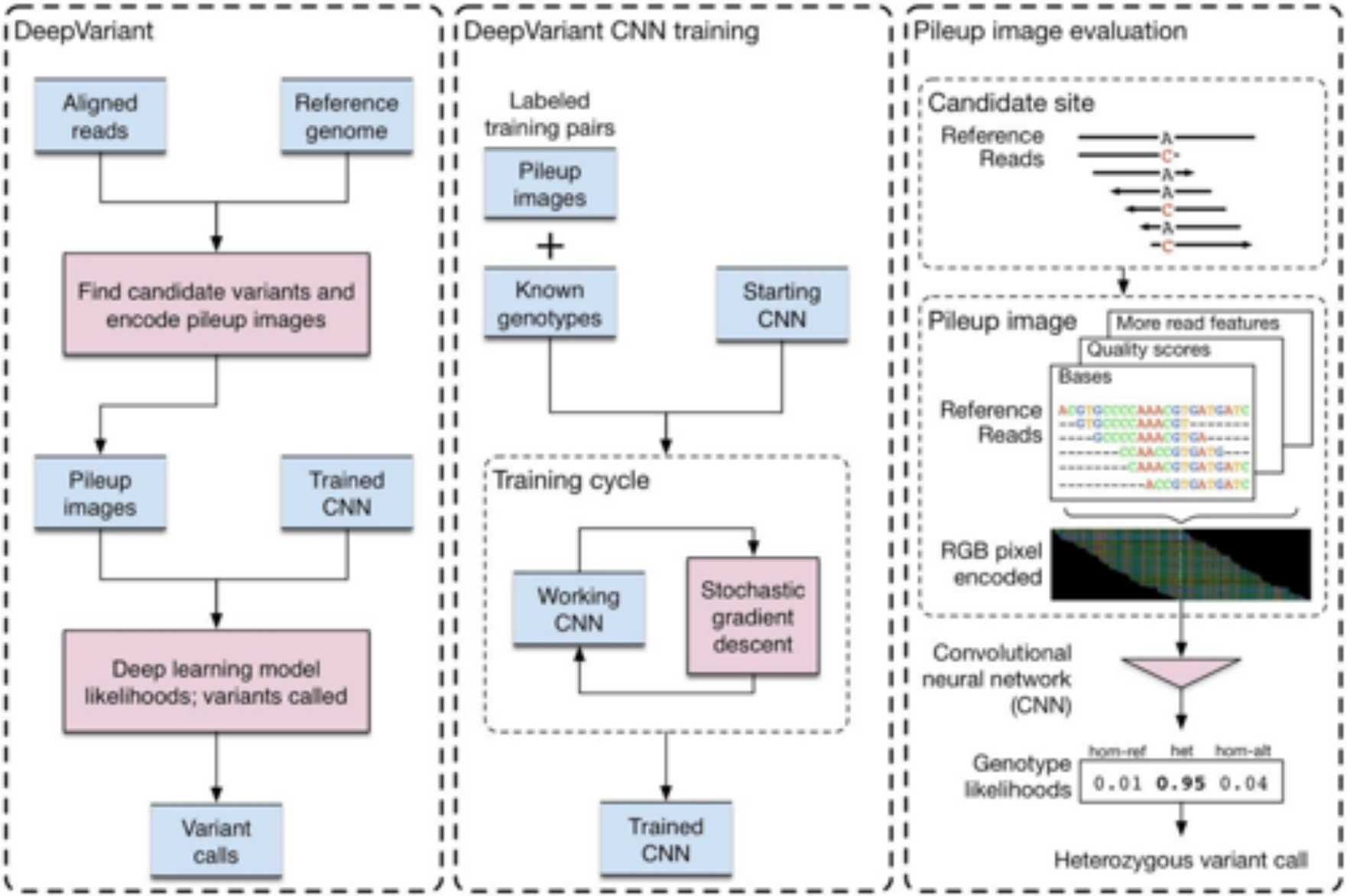
A general approach to single-nucleotide polymorphism discovery

Gabor T. Marth[1], Ian Korf[1], Mark D. Yandell[1], Raymond T. Yeh[1], Zhijie Gu[2], Hamideh Zakeri[2], Nathan O. Stitziel[1], LaDeana Hillier[1], Pui-Yan Kwok[2] & Warren R. Gish[1]

This Bayesian statistical framework has been adopted by other modern SNP/INDEL callers such as FreeBayes, GATK, and samtools

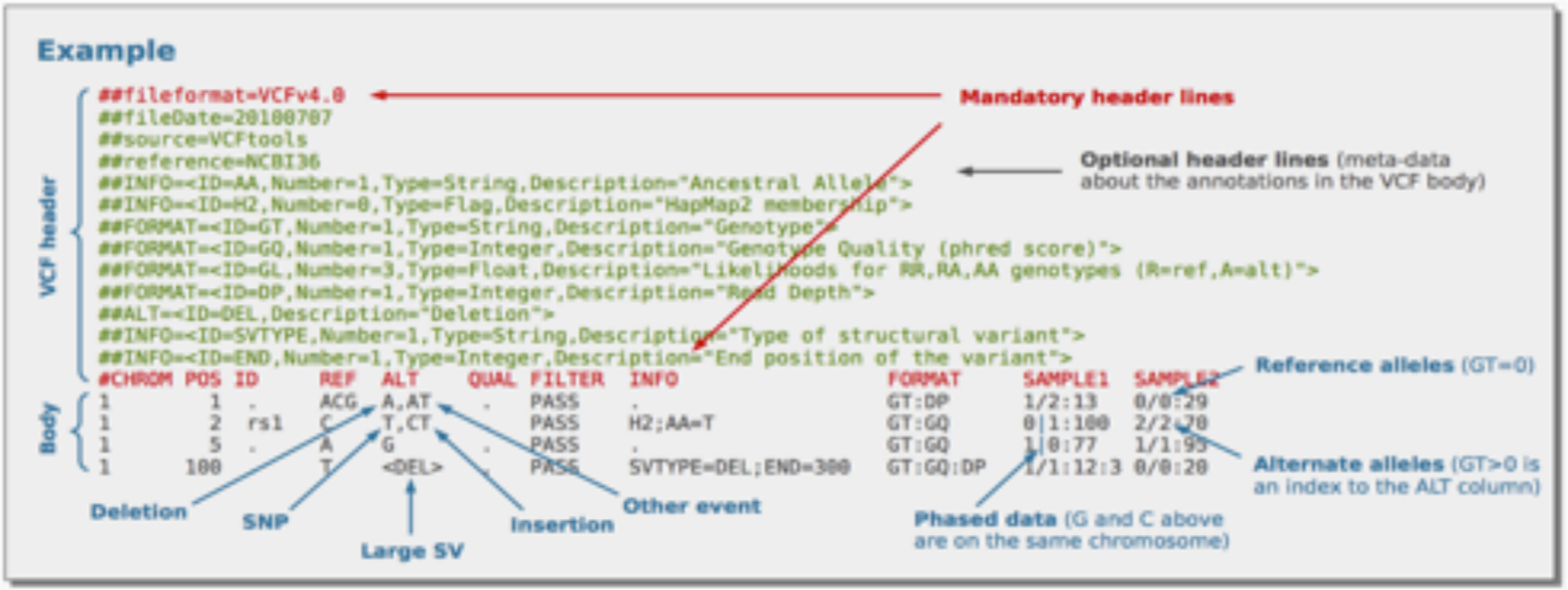
http://www.nature.com/ng/journal/v23/n4/full/ng1299_452.html

Deep Variant

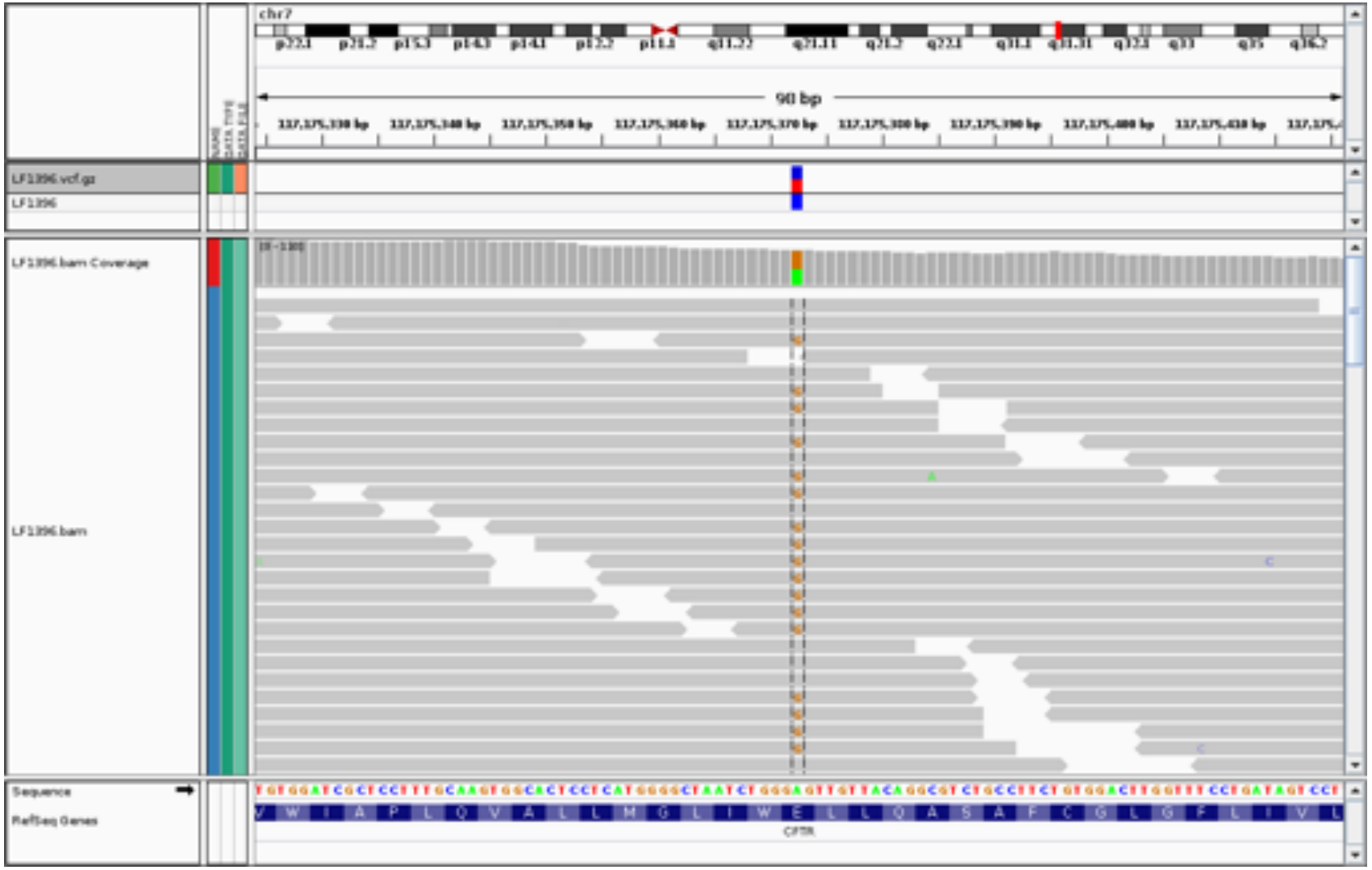


A universal SNP and small-indel variant caller using deep neural networks
Poplin et al. (2018) Nature Biotechnology. https://doi.org/10.1038/nbt.4235

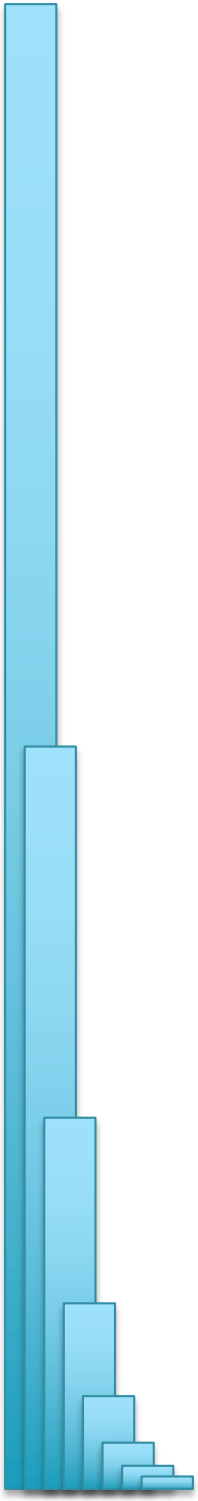
VCF Format


Example
VCF header
Body
##fileformat=VCFv4.0
##fileDate=20100707
##source=VCFtools
##reference=NCBI36
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality (phred score)">
##FORMAT=<ID=GL,Number=3,Type=Float,Description="Likelihoods for RR,RA,AA genotypes (R=ref,A=alt)">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##ALT=<ID=DEL,Description="Deletion">
##INFO=<ID=SVTYPE,Number=1,Type=String,Description="Type of structural variant">
##INFO=<ID=END,Number=1,Type=Integer,Description="End position of the variant">
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT SAMPLE1 SAMPLE2
1 1 . ACG A,AT . PASS . GT:DP 1/2:13 0/0:29
1 2 rs1 C T,CT PASS H2;AA=T GT:GQ 0|1:100 2/2:70
1 5 . A G . PASS . GT:GQ 1|0:77 1/1:95
1 100 T <DEL> . PASS SVTYPE=DEL;END=300 GT:GQ:DP 1/1:12:3 0/0:20
Mandatory header lines
Optional header lines (meta-data about the annotations in the VCF body)
Reference alleles (GT=0)
Alternate alleles (GT>0 is an index to the ALT column)
Deletion
SNP
Large SV
Insertion
Other event
Phased data (G and C above are on the same chromosome)

VCF Format

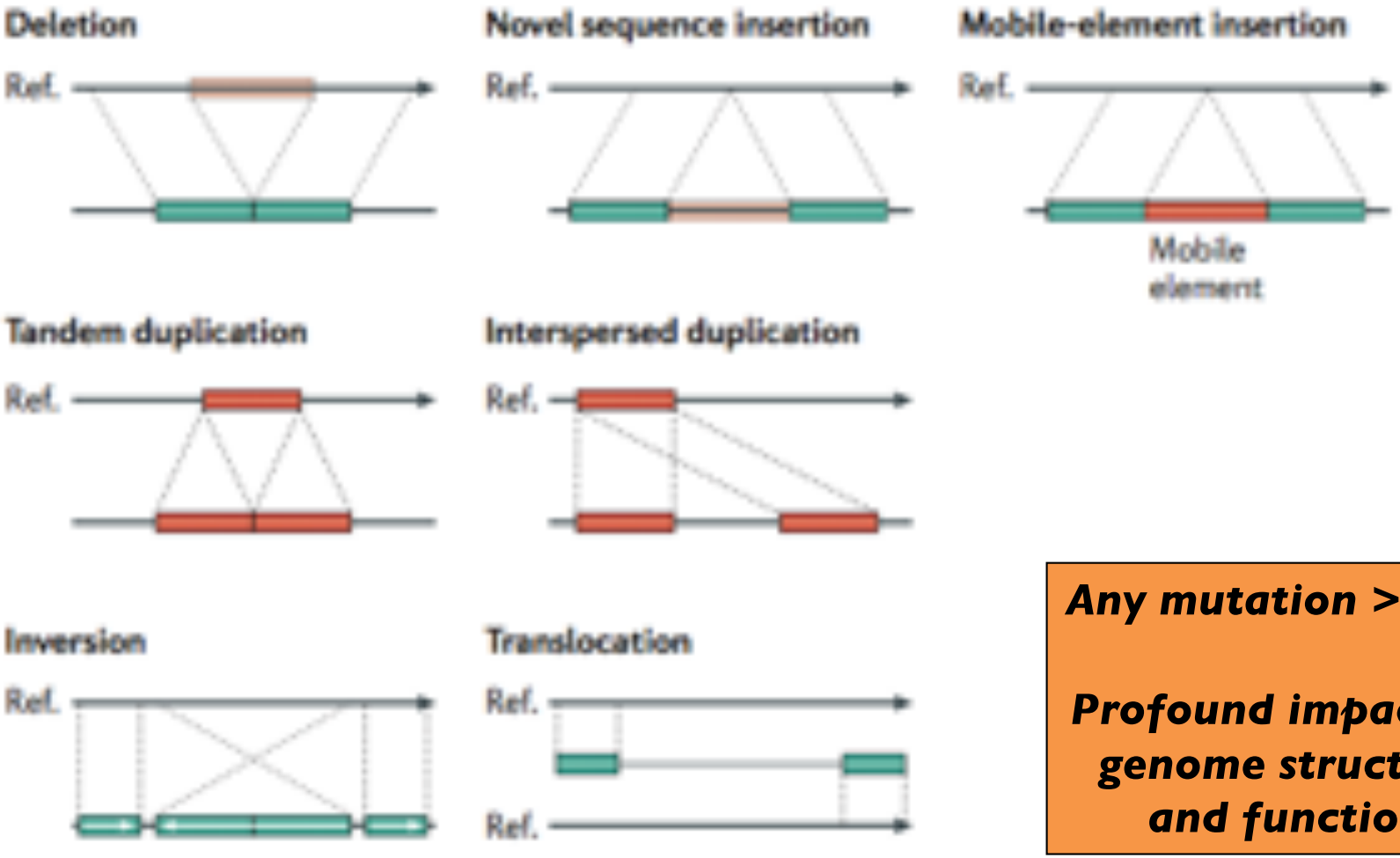



```
#CHROM POS        ID  REF ALT QUAL   FILTER INFO    FORMAT  LF1396
chr7   117175373  .   A   G   90     PASS   AF=0.5  GT      0/1
```



Part 4: What about indels & structural variants

Structural Variations



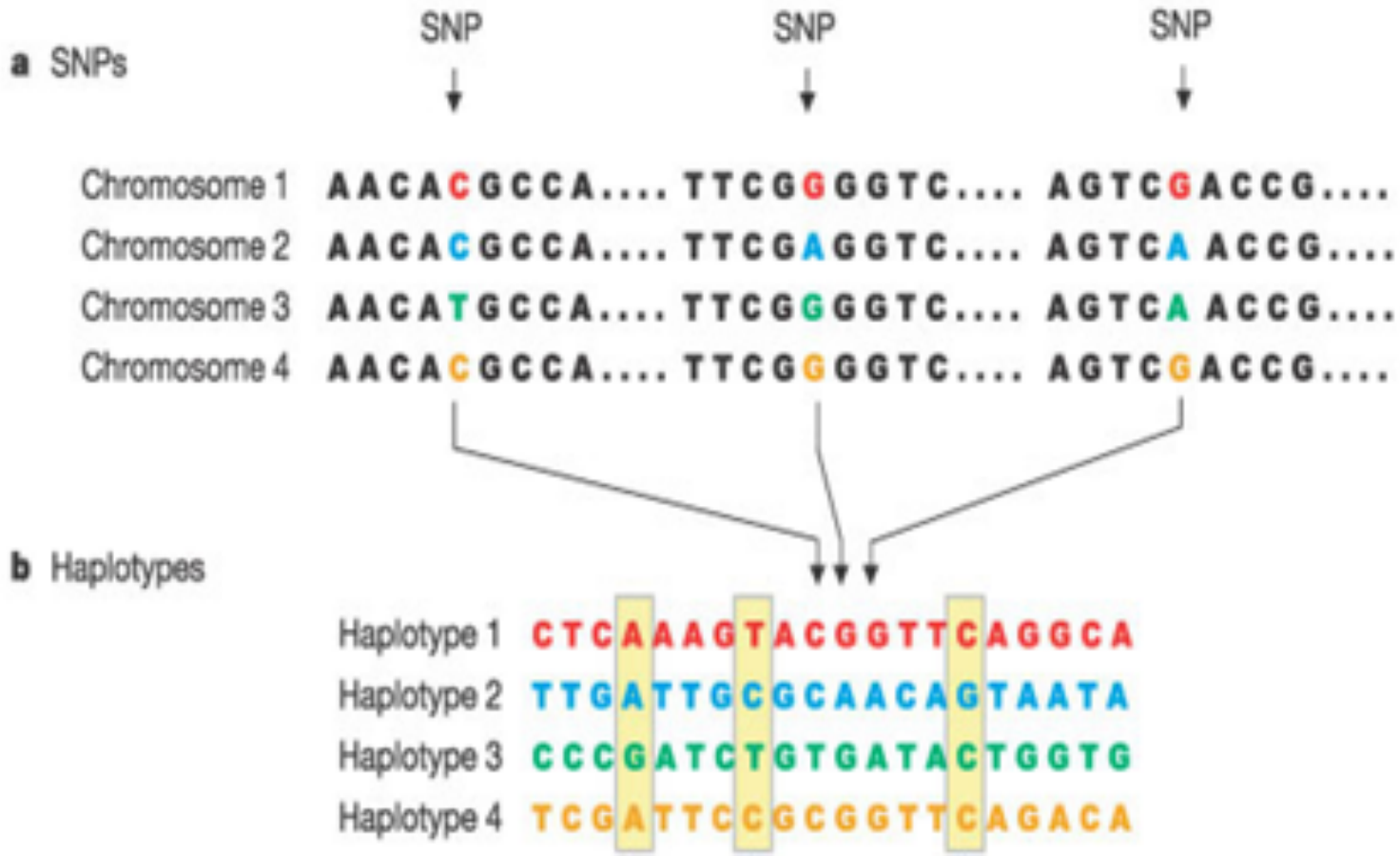
Any mutation >50bp

Profound impact on genome structure and function

Genome structural variation discovery and genotyping

Alkan, C, Coe, BP, Eichler, EE (2011) *Nature Reviews Genetics*. May;12(5):363-76. doi: 10.1038/nrg2958.

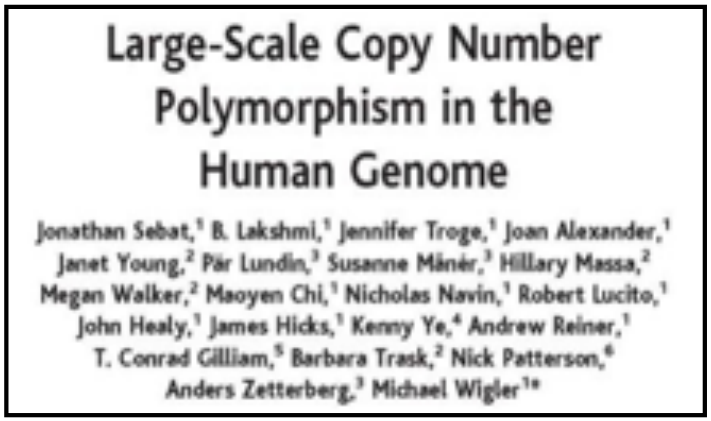
Early 2000s dogma: SNPs account for most human genetic variation



https://hapmap.ncbi.nlm.nih.gov

Discovery of abundant copy-number variation

Science, **July 2004**

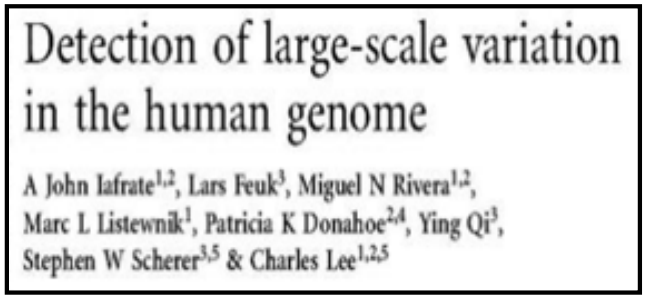


Large-Scale Copy Number Polymorphism in the Human Genome

Jonathan Sebat,[1] B. Lakshmi,[1] Jennifer Troge,[1] Joan Alexander,[1] Janet Young,[2] Pär Lundin,[3] Susanne Månér,[3] Hillary Massa,[2] Megan Walker,[2] Maoyen Chi,[1] Nicholas Navin,[1] Robert Lucito,[1] John Healy,[1] James Hicks,[1] Kenny Ye,[4] Andrew Reiner,[1] T. Conrad Gilliam,[5] Barbara Trask,[2] Nick Patterson,[6] Anders Zetterberg,[3] Michael Wigler[1]*

76 CNVs in 20 individuals
70 genes

Nature Genetics, **Aug. 2004**



Detection of large-scale variation in the human genome

A John Iafrate[1,2], Lars Feuk[3], Miguel N Rivera[1,2], Marc L Listewnik[1], Patricia K Donahoe[2,4], Ying Qi[3], Stephen W Scherer[3,5] & Charles Lee[1,2,5]

255 CNVs in 55 individuals
127 genes

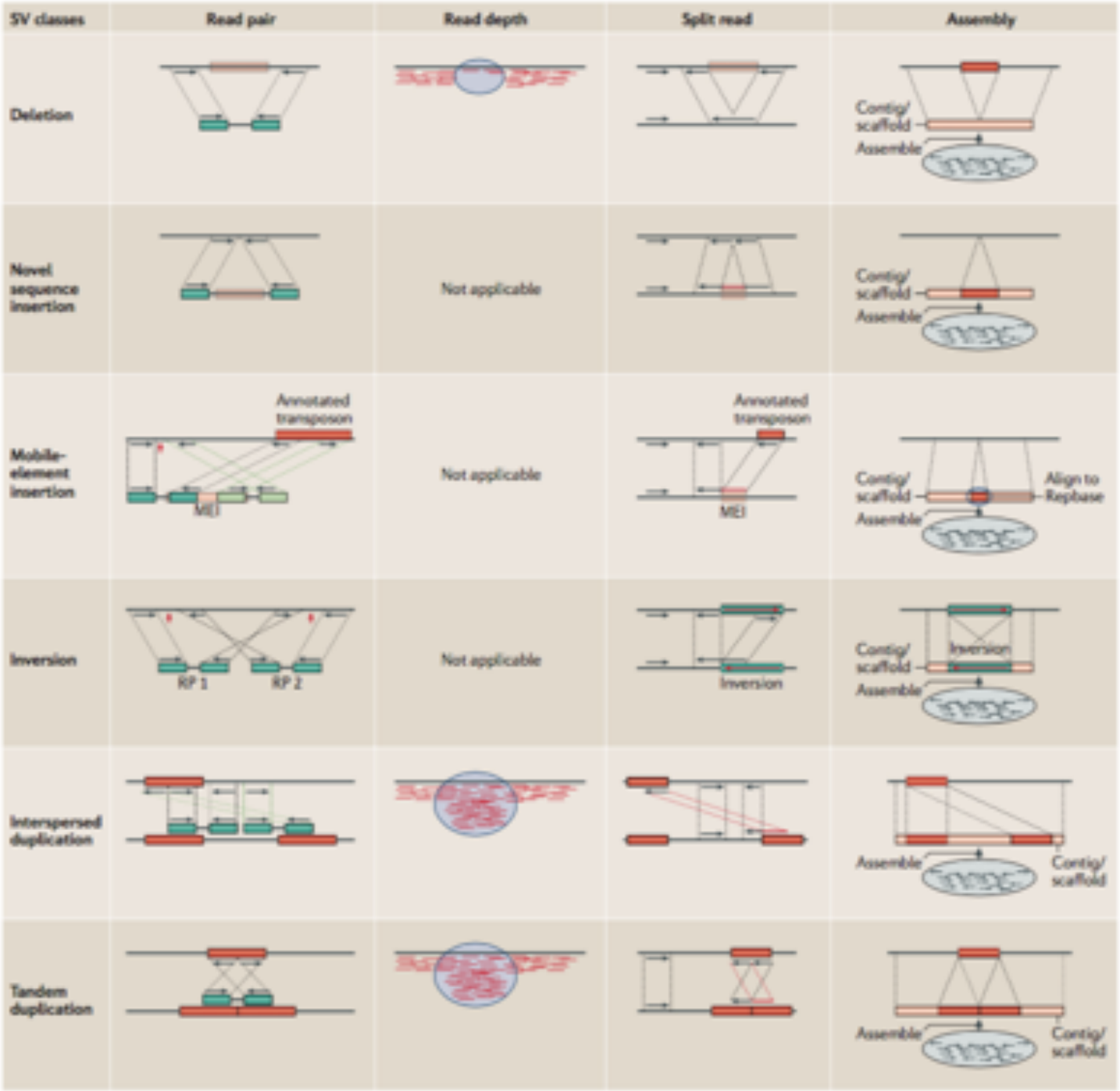
- 331 CNVs, only 11 in common
- Half observed in only 1 individual
- Impact "plenty" of genes
- Correlated with segmental duplications in the reference genome

Why is structural variation relevant / important?

- They are common and affect a large fraction of the genome
  - In total, SVs impact more base pairs than all single-nucleotide differences.

- They are a major driver of genome evolution
  - Speciation can be driven by rapid changes in genome architecture
  - Genome instability and aneuploidy: hallmarks of solid tumor genomes

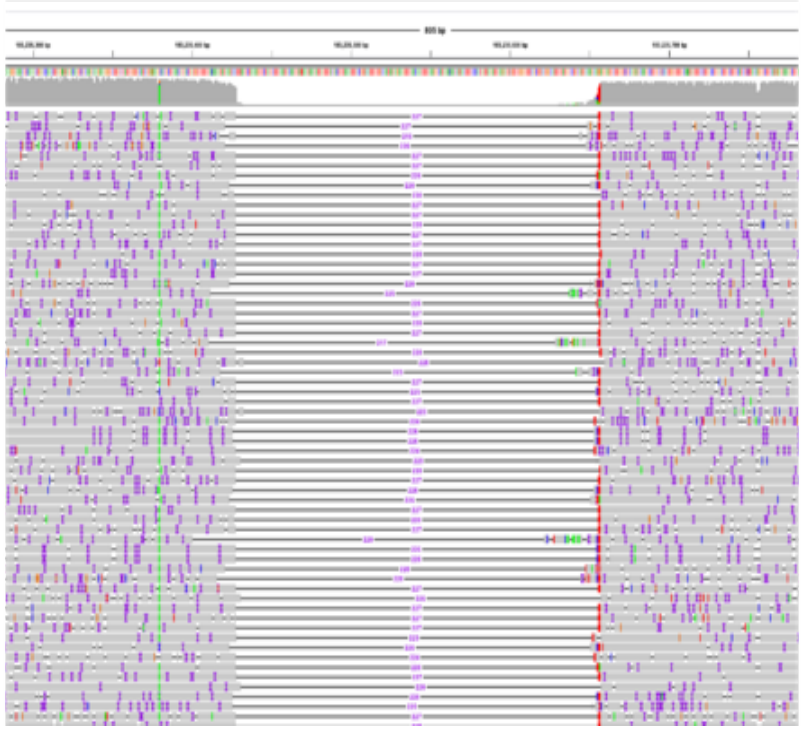
Structural Variation Sequence Signatures



NGMLR + Sniffles

BWA-MEM:

NGMLR:



NGMLR: Convex gap penalty to balance frequent small sequencing errors with larger SVs

Sniffles: Scan within and between split reads to accurately find SVs (Ins, Del, Dup, Inv, Trans)

Mendelian concordance >95%, experimental validation also very high

Accurate detection of complex structural variations using single molecule sequencing

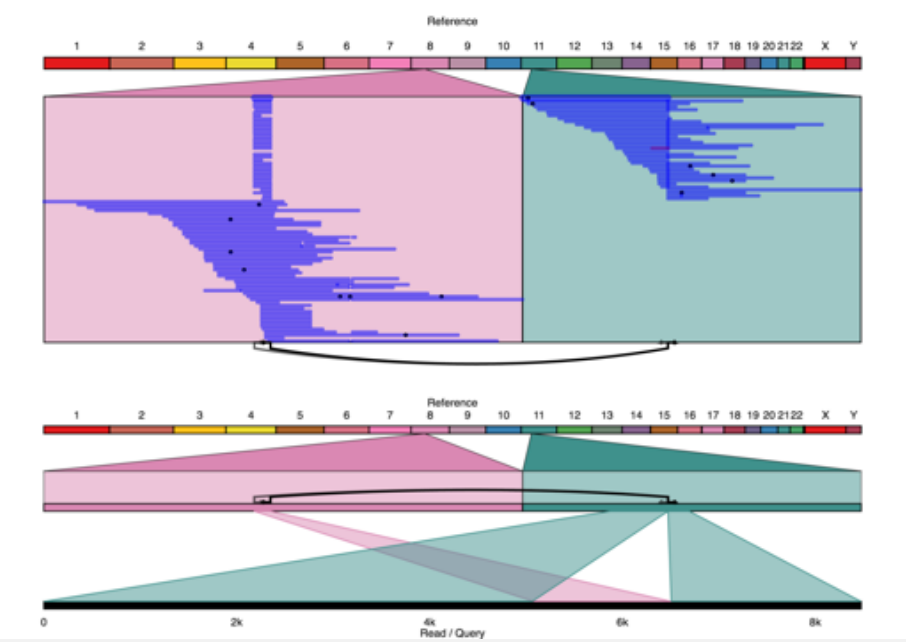
Sedlazeck, Rescheneder et al (2018) *Nature Methods*.

SVs in a typical healthy human

Sniffles calls

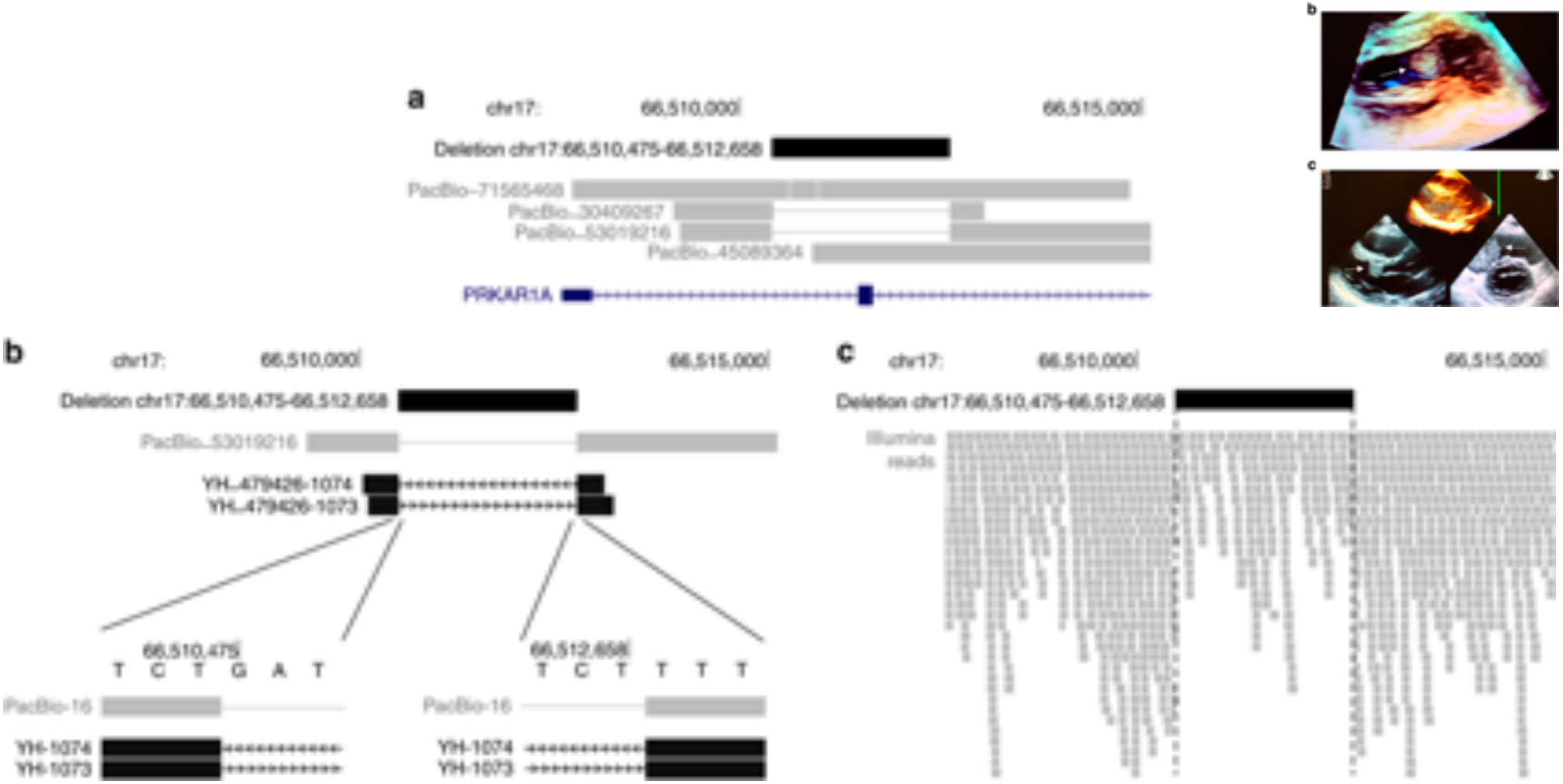
	All SVs (50bp+)	Large SVs (10kbp+)
Deletions	7,389	164
Duplications	1,284	139
Insertions	8,382	4
Inversions	229	116
Translocations	170	170
All	***17,454***	***593***

Translocation in Ribbon



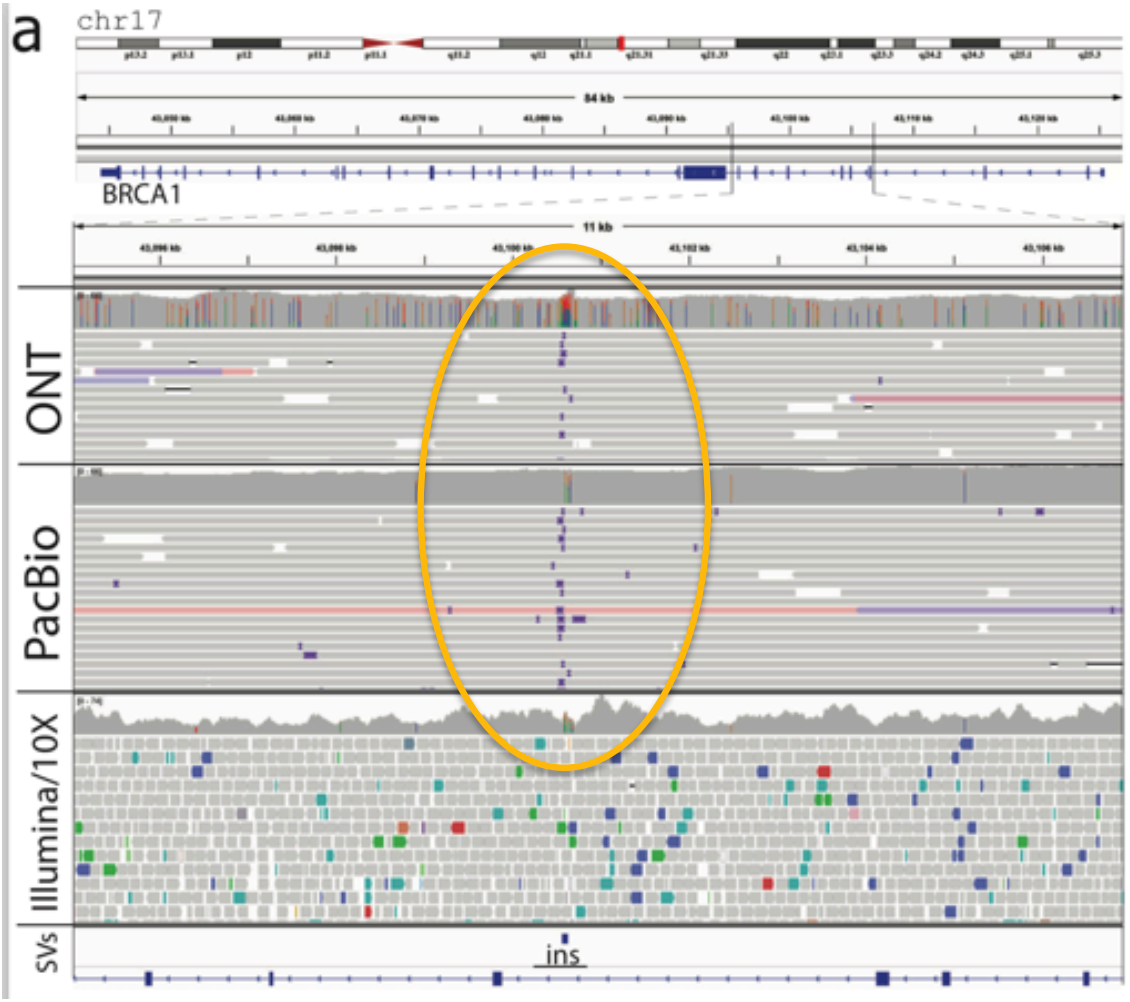
Ribbon: Visualizing complex genome alignments and structural variation
Nattestad et al. (2020) Bioinformatics

Structural Variations in Human Disease



Long-read genome sequencing identifies causal structural variation in a Mendelian disease
Merker et al (2017) *Genetics in Medicine. doi:10.1038/gim.2017.86*

Hidden Variants in Breast Cancer Genes



62bp repeat expansion in BRCA1 detected in normal tissue that is undetectable using a panel or short read sequencing

*** More results in a few weeks ***