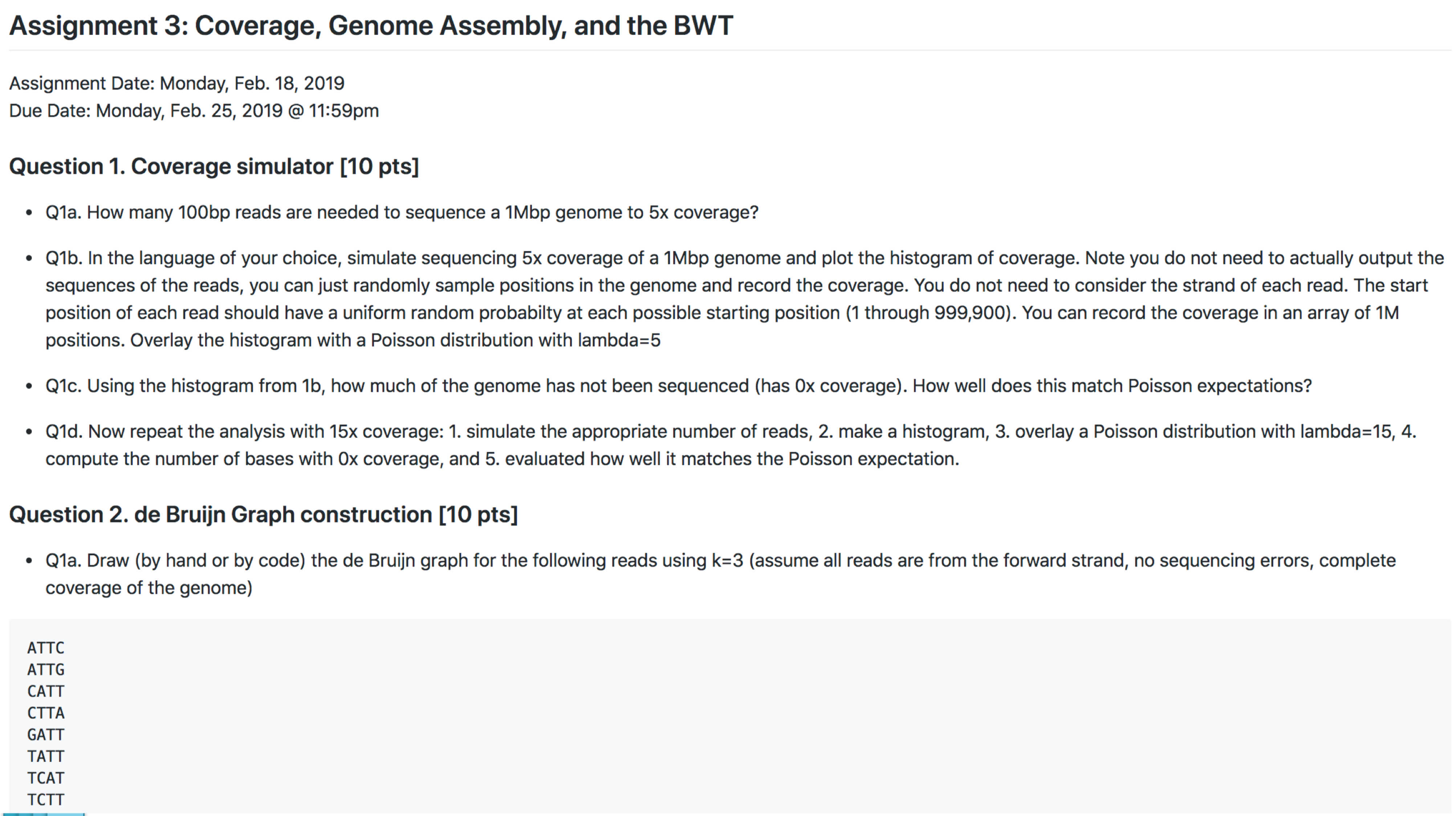
Variant Calling

Michael Schatz

Feb 25, 2019

Lecture 9: Applied Comparative Genomics

Assignment 3: Due Monday Feb 25



Assignment 3: Coverage, Genome Assembly, and the BWT

Assignment Date: Monday, Feb. 18, 2019
Due Date: Monday, Feb. 25, 2019 @ 11:59pm

Question 1. Coverage simulator [10 pts]

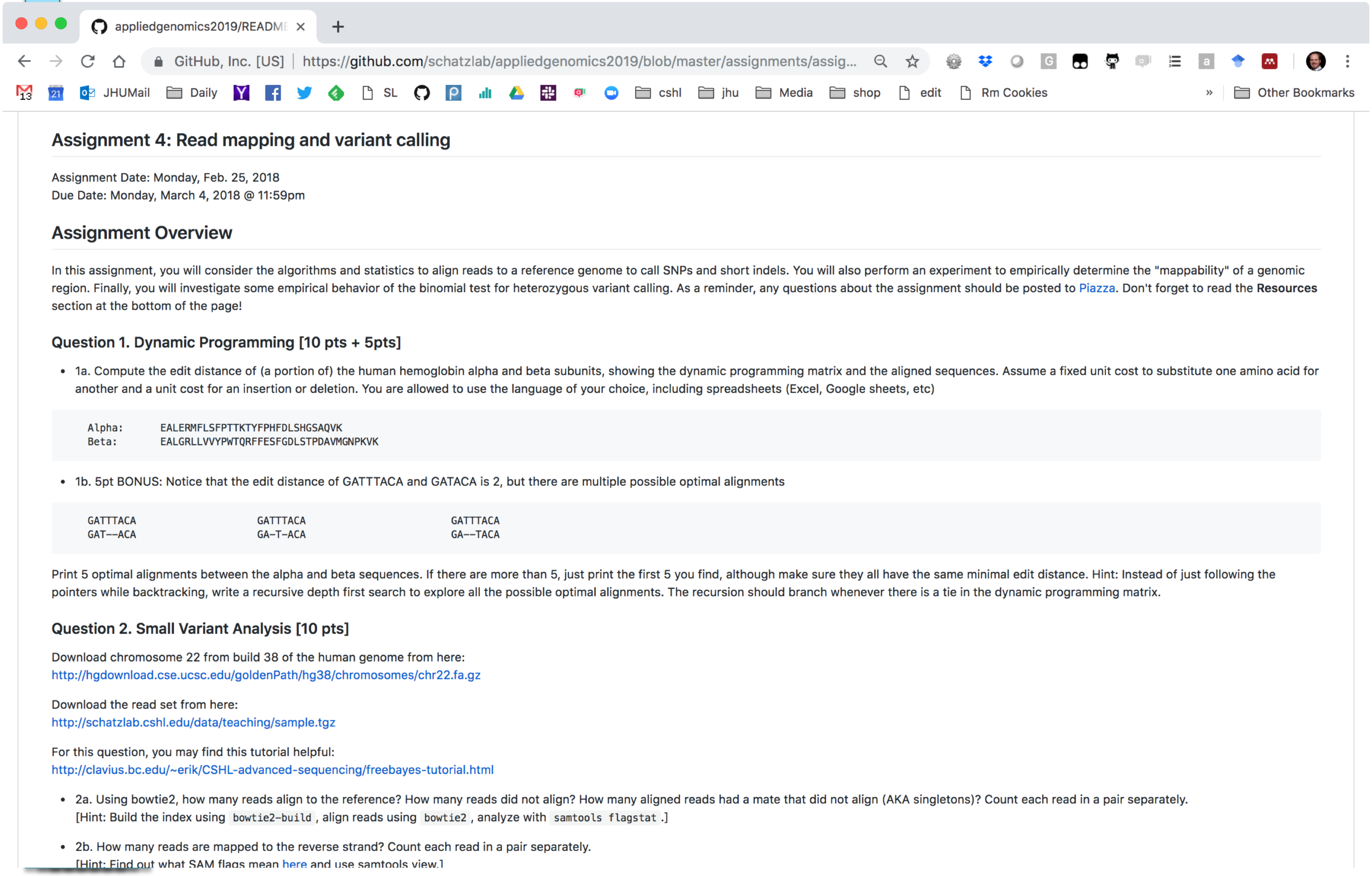
- Q1a. How many 100bp reads are needed to sequence a 1Mbp genome to 5x coverage?
- Q1b. In the language of your choice, simulate sequencing 5x coverage of a 1Mbp genome and plot the histogram of coverage. Note you do not need to actually output the sequences of the reads, you can just randomly sample positions in the genome and record the coverage. You do not need to consider the strand of each read. The start position of each read should have a uniform random probabilty at each possible starting position (1 through 999,900). You can record the coverage in an array of 1M positions. Overlay the histogram with a Poisson distribution with lambda=5
- Q1c. Using the histogram from 1b, how much of the genome has not been sequenced (has 0x coverage). How well does this match Poisson expectations?
- Q1d. Now repeat the analysis with 15x coverage: 1. simulate the appropriate number of reads, 2. make a histogram, 3. overlay a Poisson distribution with lambda=15, 4. compute the number of bases with 0x coverage, and 5. evaluated how well it matches the Poisson expectation.

Question 2. de Bruijn Graph construction [10 pts]

- Q1a. Draw (by hand or by code) the de Bruijn graph for the following reads using k=3 (assume all reads are from the forward strand, no sequencing errors, complete coverage of the genome)

```
ATTC
ATTG
CATT
CTTA
GATT
TATT
TCAT
TCTT
```

Assignment 4: Due Monday March 4



appliedgenomics2019/README

GitHub, Inc. [US] | https://github.com/schatzlab/appliedgenomics2019/blob/master/assignments/assig...

JHUMail Daily SL cshl jhu Media shop edit Rm Cookies Other Bookmarks

Assignment 4: Read mapping and variant calling

Assignment Date: Monday, Feb. 25, 2018
Due Date: Monday, March 4, 2018 @ 11:59pm

Assignment Overview

In this assignment, you will consider the algorithms and statistics to align reads to a reference genome to call SNPs and short indels. You will also perform an experiment to empirically determine the "mappability" of a genomic region. Finally, you will investigate some empirical behavior of the binomial test for heterozygous variant calling. As a reminder, any questions about the assignment should be posted to Piazza. Don't forget to read the **Resources** section at the bottom of the page!

Question 1. Dynamic Programming [10 pts + 5pts]

- 1a. Compute the edit distance of (a portion of) the human hemoglobin alpha and beta subunits, showing the dynamic programming matrix and the aligned sequences. Assume a fixed unit cost to substitute one amino acid for another and a unit cost for an insertion or deletion. You are allowed to use the language of your choice, including spreadsheets (Excel, Google sheets, etc)

```
    Alpha:      EALERMFLSFPTTKTYFPHFDLSHGSAQVK
    Beta:       EALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKVK
```

- 1b. 5pt BONUS: Notice that the edit distance of GATTTACA and GATACA is 2, but there are multiple possible optimal alignments

```
    GATTTACA                     GATTTACA                         GATTTACA
    GAT--ACA                     GA-T-ACA                         GA--TACA
```

Print 5 optimal alignments between the alpha and beta sequences. If there are more than 5, just print the first 5 you find, although make sure they all have the same minimal edit distance. Hint: Instead of just following the pointers while backtracking, write a recursive depth first search to explore all the possible optimal alignments. The recursion should branch whenever there is a tie in the dynamic programming matrix.

Question 2. Small Variant Analysis [10 pts]

Download chromosome 22 from build 38 of the human genome from here:
http://hgdownload.cse.ucsc.edu/goldenPath/hg38/chromosomes/chr22.fa.gz

Download the read set from here:
http://schatzlab.cshl.edu/data/teaching/sample.tgz

For this question, you may find this tutorial helpful:
http://clavius.bc.edu/~erik/CSHL-advanced-sequencing/freebayes-tutorial.html

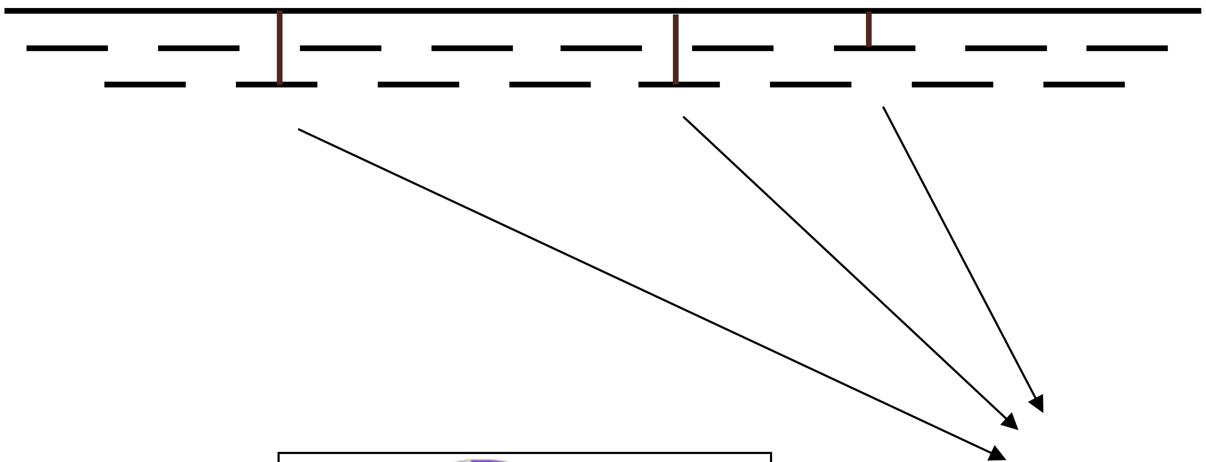
- 2a. Using bowtie2, how many reads align to the reference? How many reads did not align? How many aligned reads had a mate that did not align (AKA singletons)? Count each read in a pair separately.
  [Hint: Build the index using `bowtie2-build`, align reads using `bowtie2`, analyze with `samtools flagstat`.]

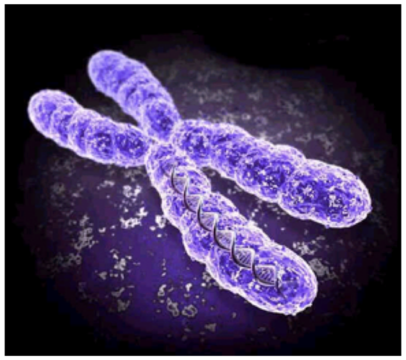
- 2b. How many reads are mapped to the reverse strand? Count each read in a pair separately.
  [Hint: Find out what SAM flags mean here and use samtools view.]

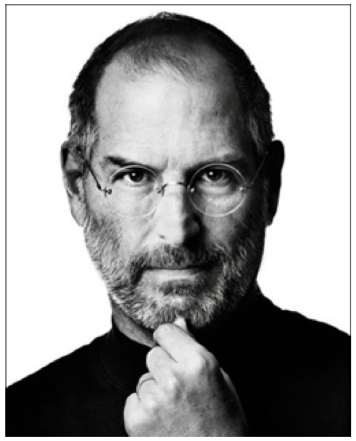
Part 1: Recap

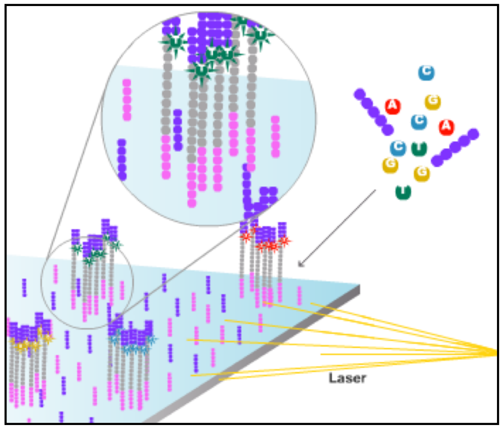
Personal Genomics

How does your genome compare to the reference?









Heart Disease

Cancer

Creates magical technology

Exact Matching Review & Overview

Where is GATTACA in the human genome?

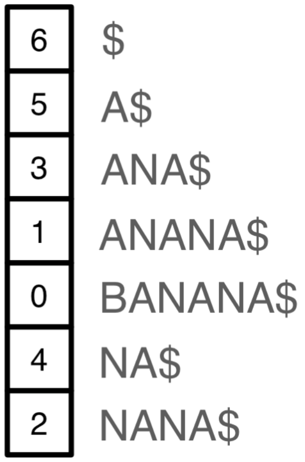
Brute Force
(3 GB)

```
BANANA
BAN
 ANA
  NAN
   ANA
```

O(m * n)

Slow & Easy

Suffix Array
(>15 GB)

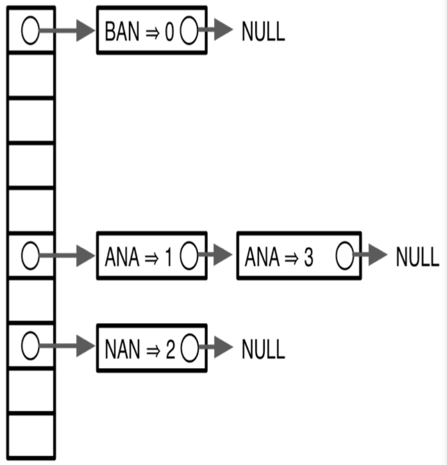


6	$
5	A$
3	ANA$
1	ANANA$
0	BANANA$
4	NA$
2	NANA$

O(m + lg n)

Full-text index

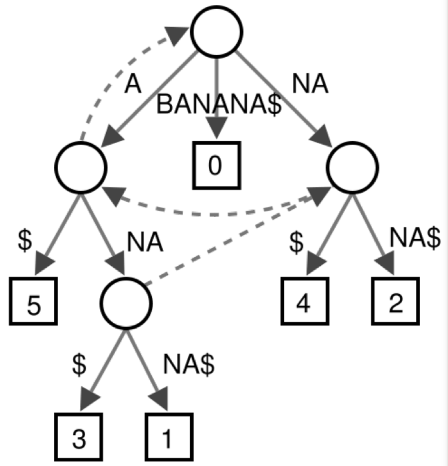
Hash Table
(>15 GB)



O(1)

Fixed-length lookup

Suffix Tree
(>51 GB)



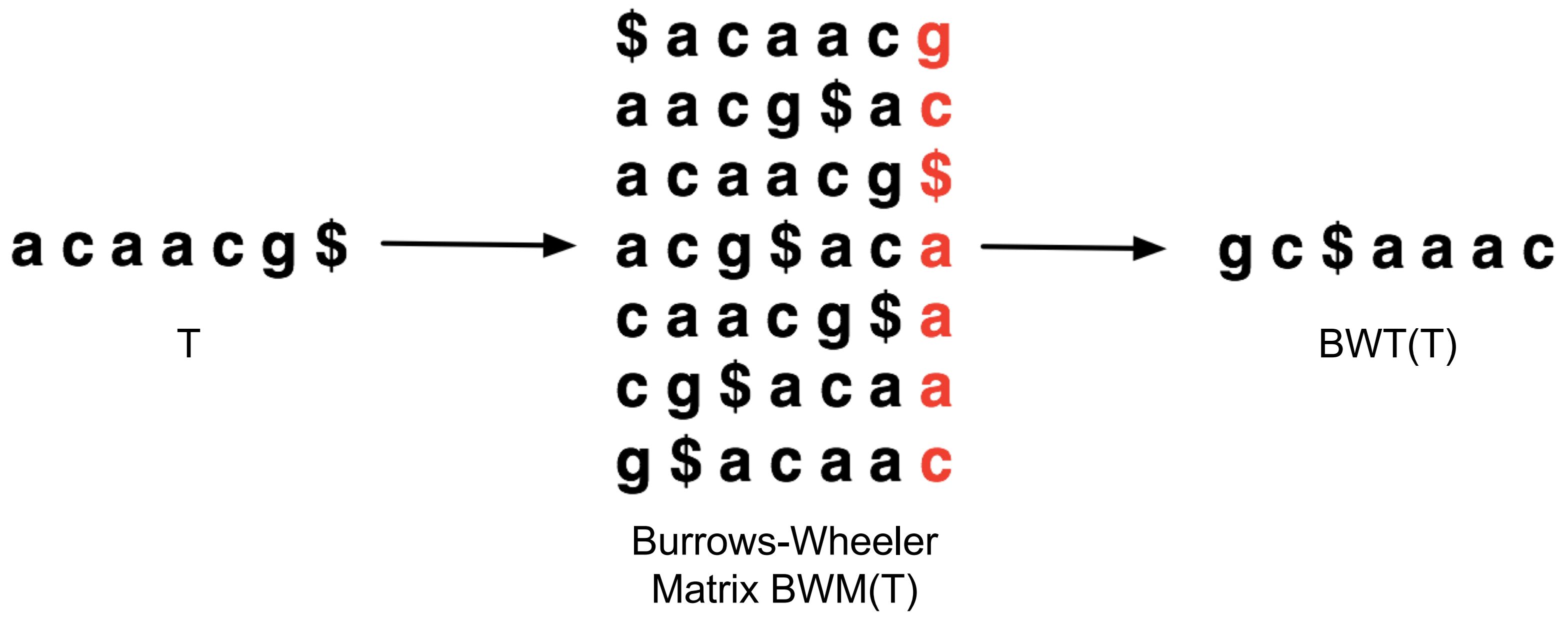
O(m)

Full-text, but bulky

*** These are general techniques applicable to any text search problem ***

Burrows-Wheeler Transform

- Permutation of the characters in a text

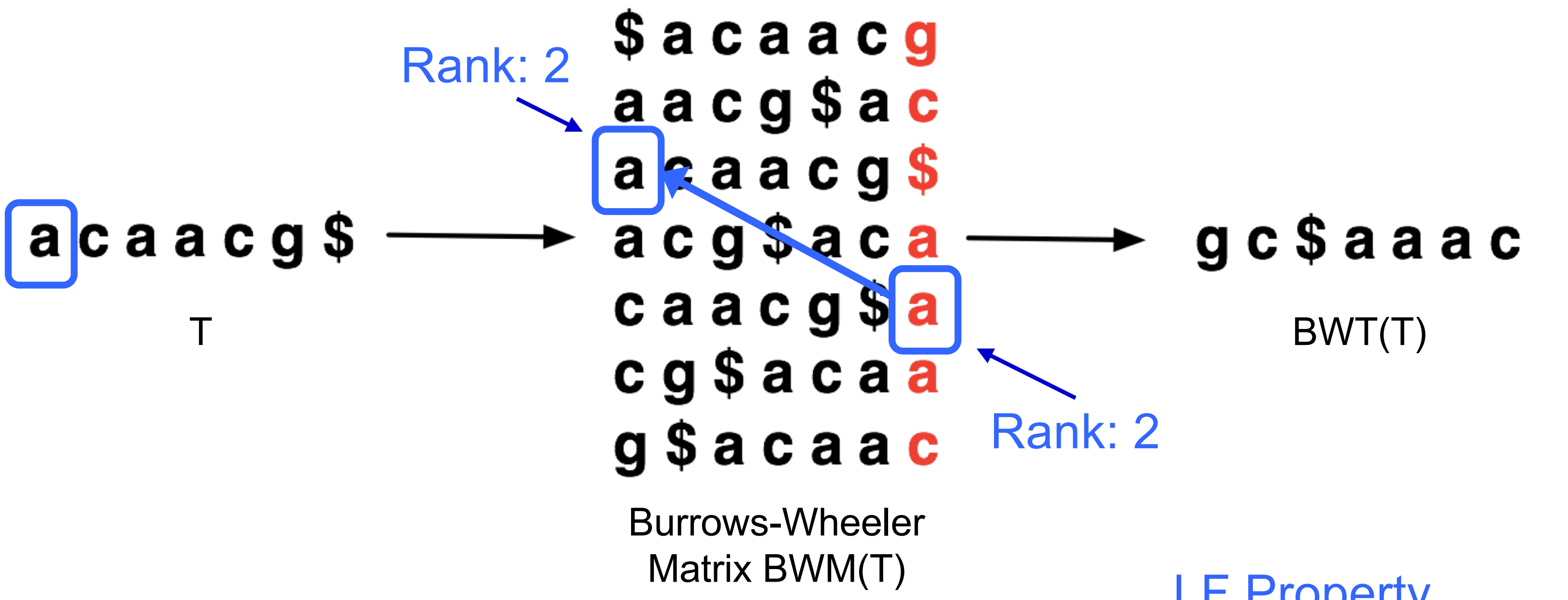


- BWT(T) is the index for T

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



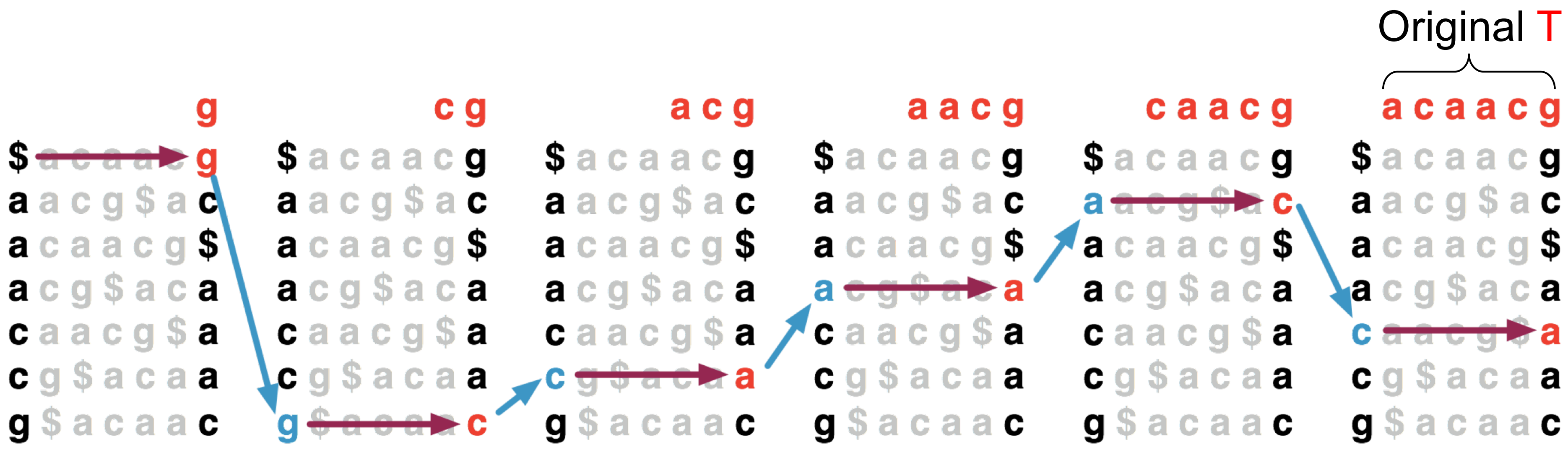
LF Property implicitly encodes Suffix Array

- 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

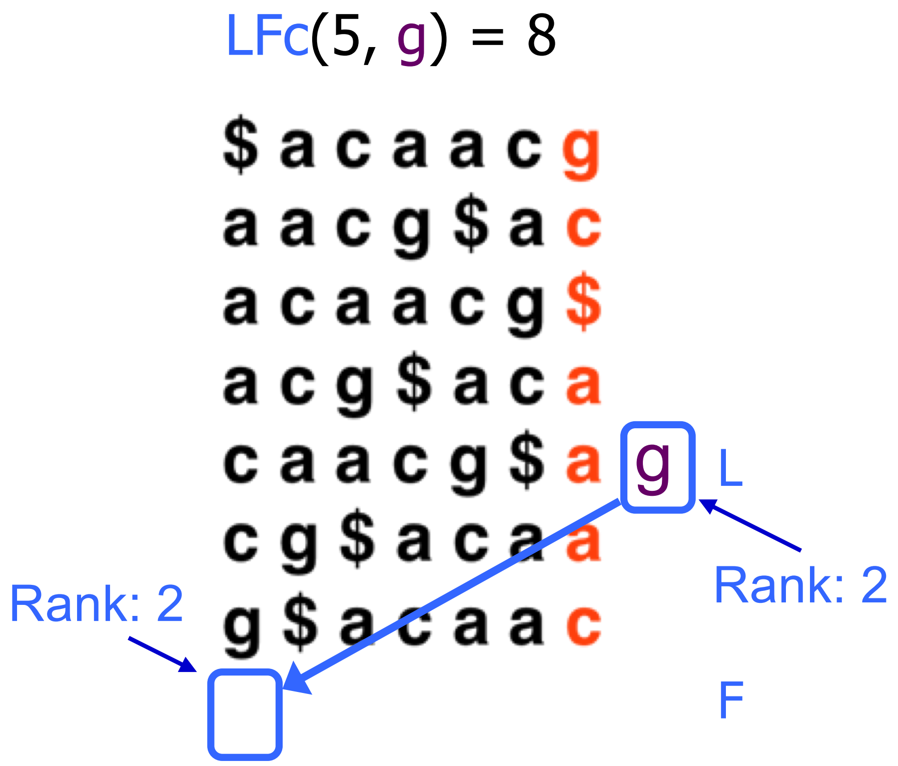
- Recreating T from BWT(T)
  - Start in the first row and apply **LF** repeatedly, accumulating predecessors along the way



[Decode this BWT string: ACTGA$TTA]

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:

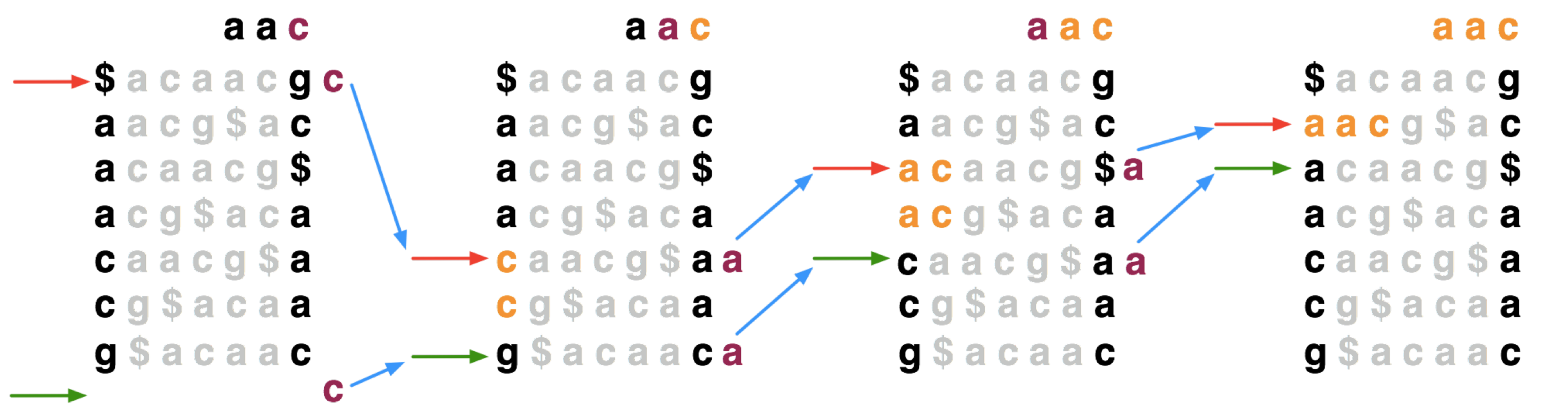


BWT Exact Matching

- 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



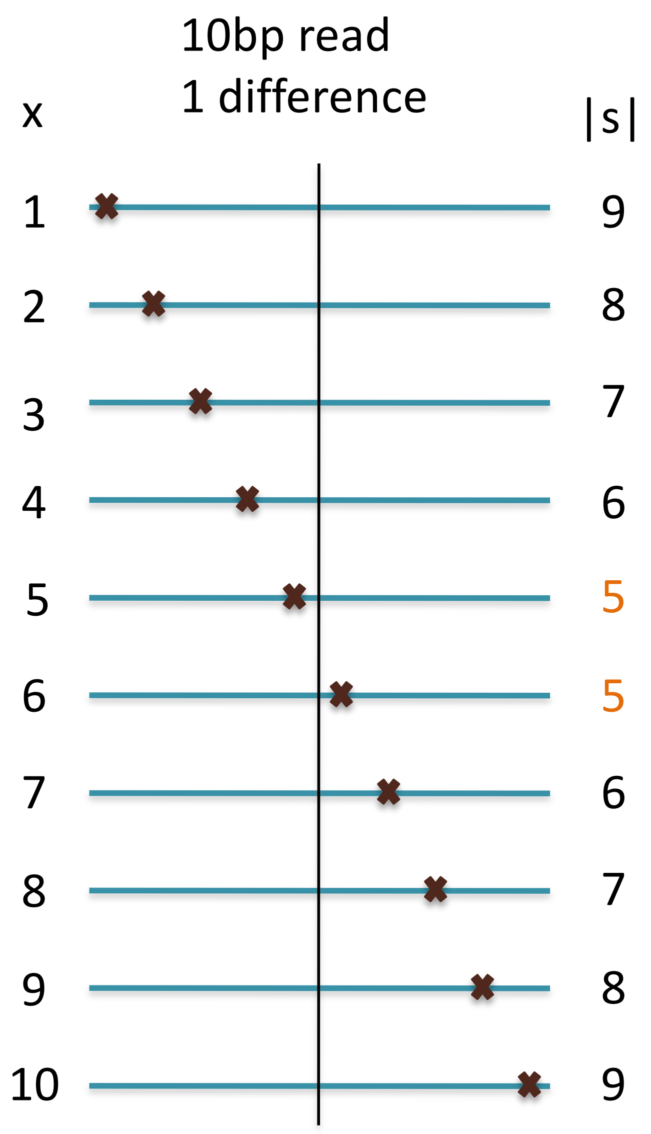
Once rows are identified, UNWIND to find genomic position

Periodically record character ranks and genomic positions (suffix array) to make it fast

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

Bowtie2 Algorithm Overview

1. Split read into segments

Read

`CCAGTAGCTCTCAGCCTTATTTTACCCAGGCCTGTA`

Read (reverse complement)

`TACAGGCCTGGGTAAAATAAGGCTGAGAGCTACTGG`

Policy: extract 16 nt seed every 10 nt

Seeds

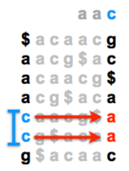
+, 0: `CCAGTAGCTCTCAGCC`
+, 10: `TCAGCCTTATTTTACC`
+, 20: `TTTACCCAGGCCTGTA`

-, 0: `TACAGGCCTGGGTAAA`
-, 10: `GGTAAAATAAGGCTGA`
-, 20: `GGCTGAGAGCTACTGG`

2. Lookup each segment and prioritize

Seeds		Seed alignments (as B ranges)
+, 0: `CCAGTAGCTCTCAGCC`	→	`{ [211, 212], [212, 214] }`
+, 10: `TCAGCCTTATTTTACC`	→	`{ [653, 654], [651, 653] }`
+, 20: `TTTACCCAGGCCTGTA`	→	`{ [684, 685] }`
-, 0: `TACAGGCCTGGGTAAA`	→	`{ }`
-, 10: `GGTAAAATAAGGCTGA`	→	`{ }`
-, 20: `GGCTGAGAGCTACTGG`	→	`{ [624, 625] }`

Ungapped alignment with FM Index

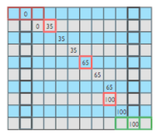


3. Evaluate end-to-end match

Extension candidates

```
SA:684, chr12:1955
SA:624, chr2:462
SA:211: chr4:762
SA:213: chr12:1935
SA:652: chr12:1945
```

→ SIMD dynamic programming aligner →



SAM alignments

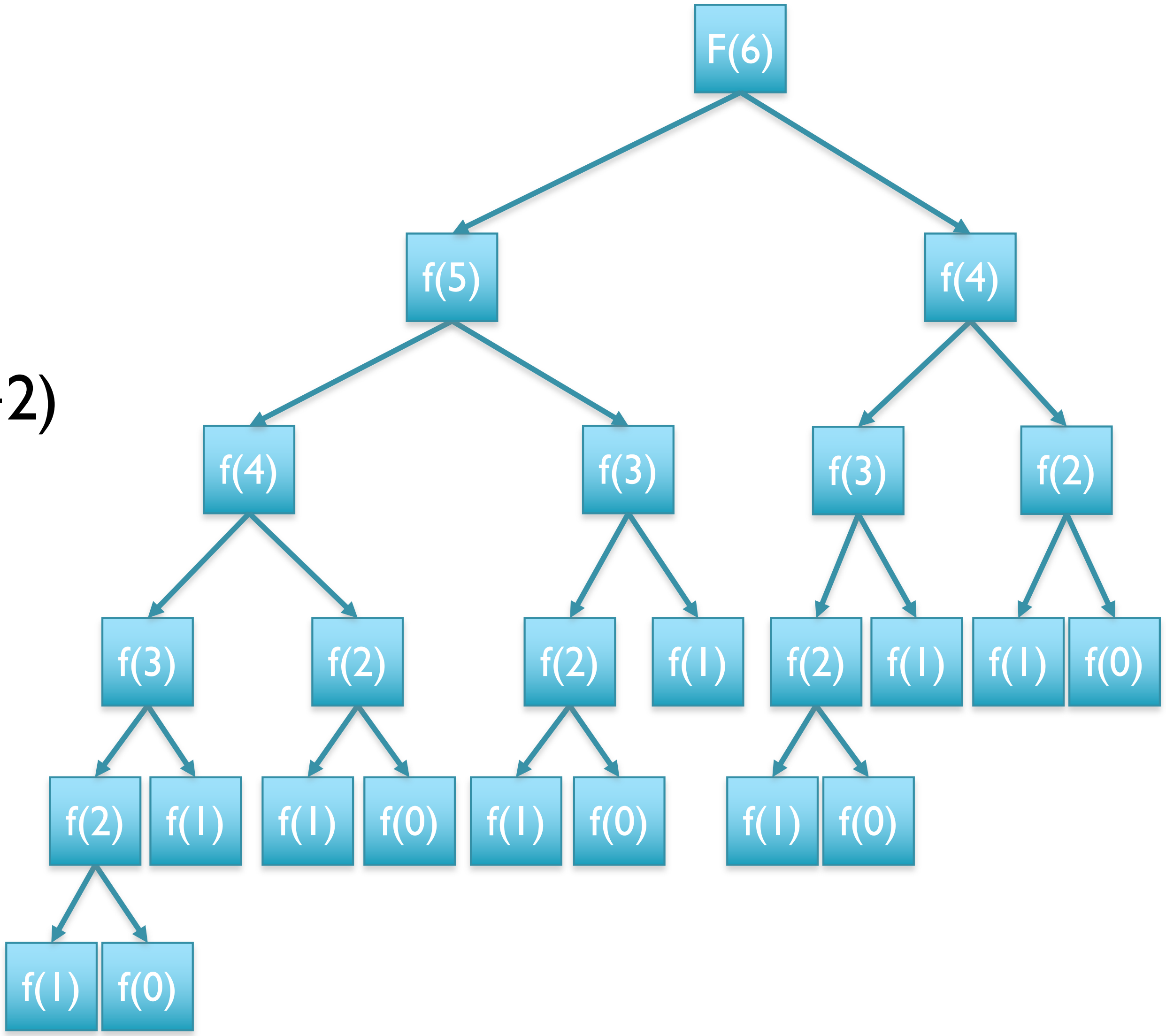
```
r1   0    chr12    1936     0
     36M  *    0    0
     CCAGTAGCTCTCAGCCTTATTTTACCCAGGCCTGTA
     IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
     AS:i:0    XS:i:-2   XN:i:0
     XM:i:0    XO:i:0    XG:i:0
     NM:i:0    MD:Z:36   YT:Z:UU
     YM:i:0
...
```

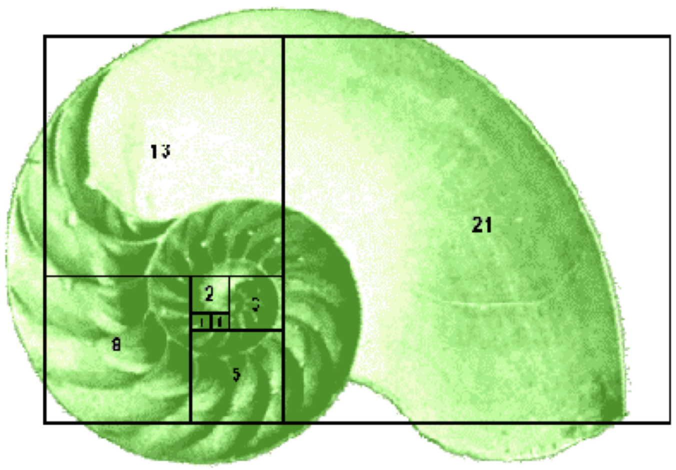
(Langmead & Salzberg, 2012)

Part 2: Dynamic Programming

Fibonacci Sequence

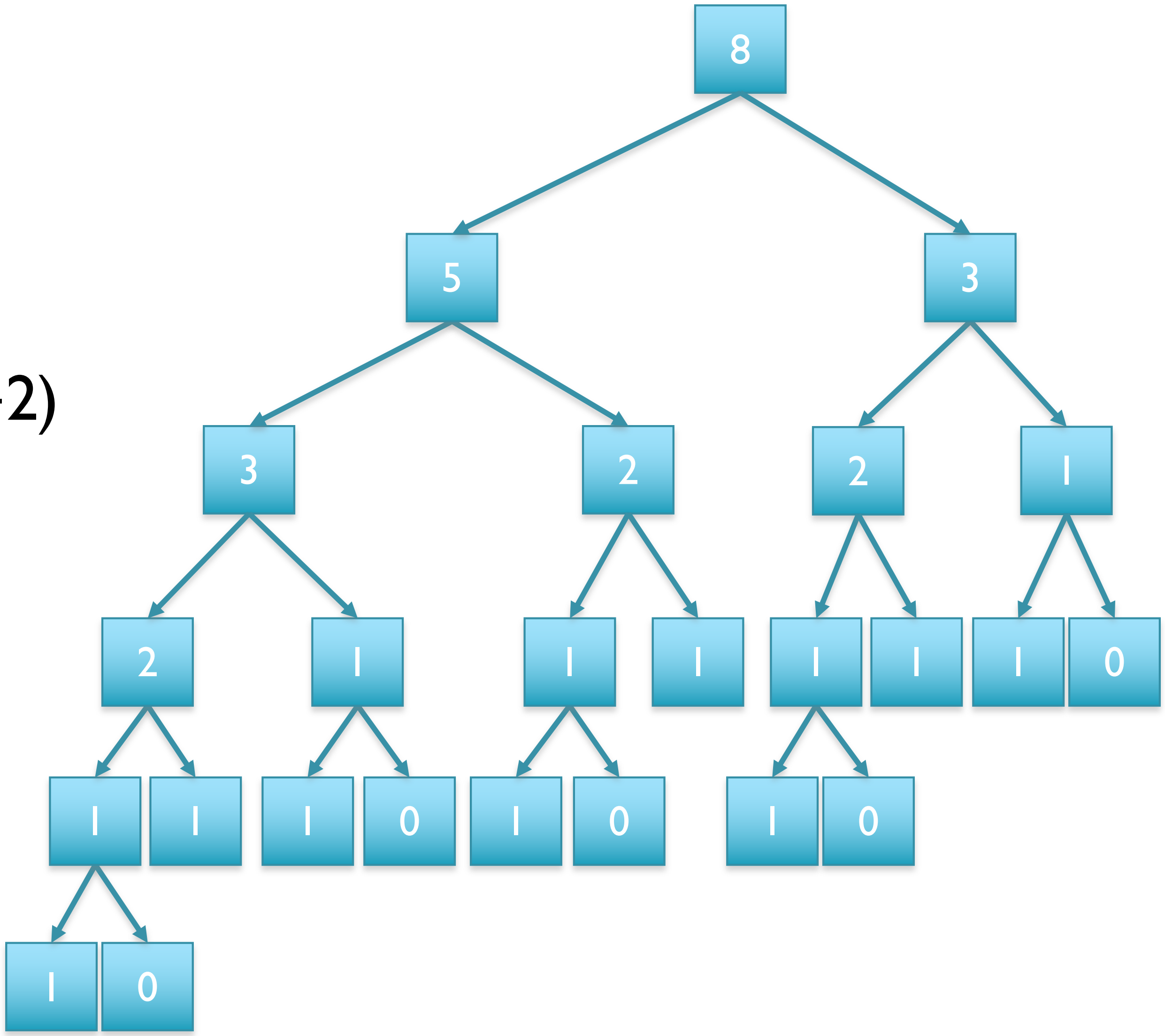
```
def fib(n):
  if n == 0 or n == 1:
     return n
  else:
  return fib(n–1) + fib(n–2)
```

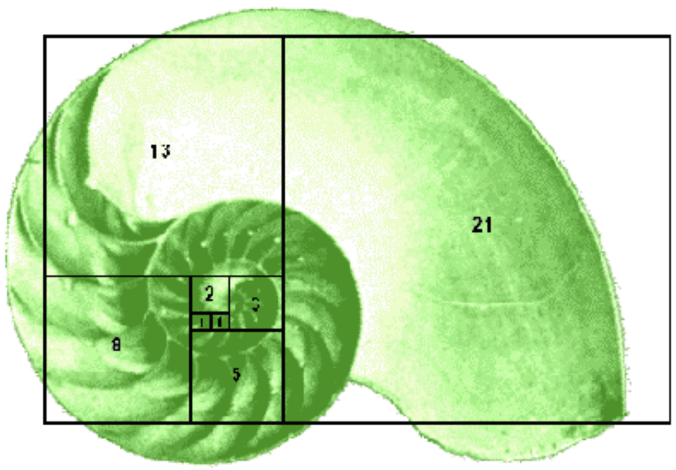




Fibonacci Sequence

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





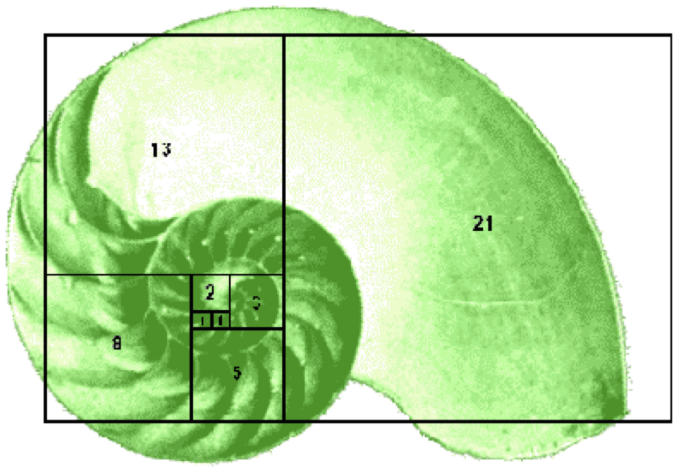
[What is the running time?]

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]
```

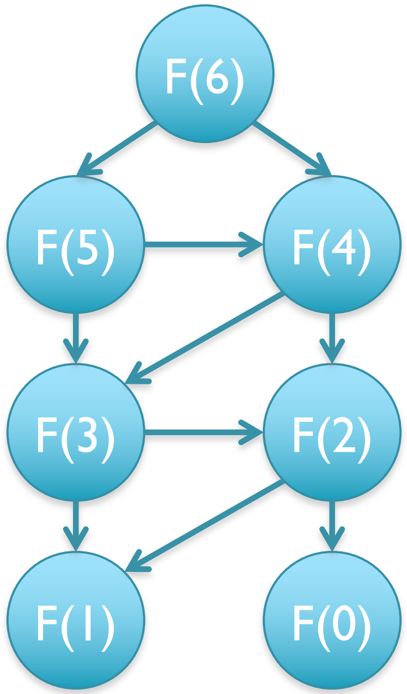
0	1	2	3	4	5	6
0	1	1	2	3	5	8

[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…

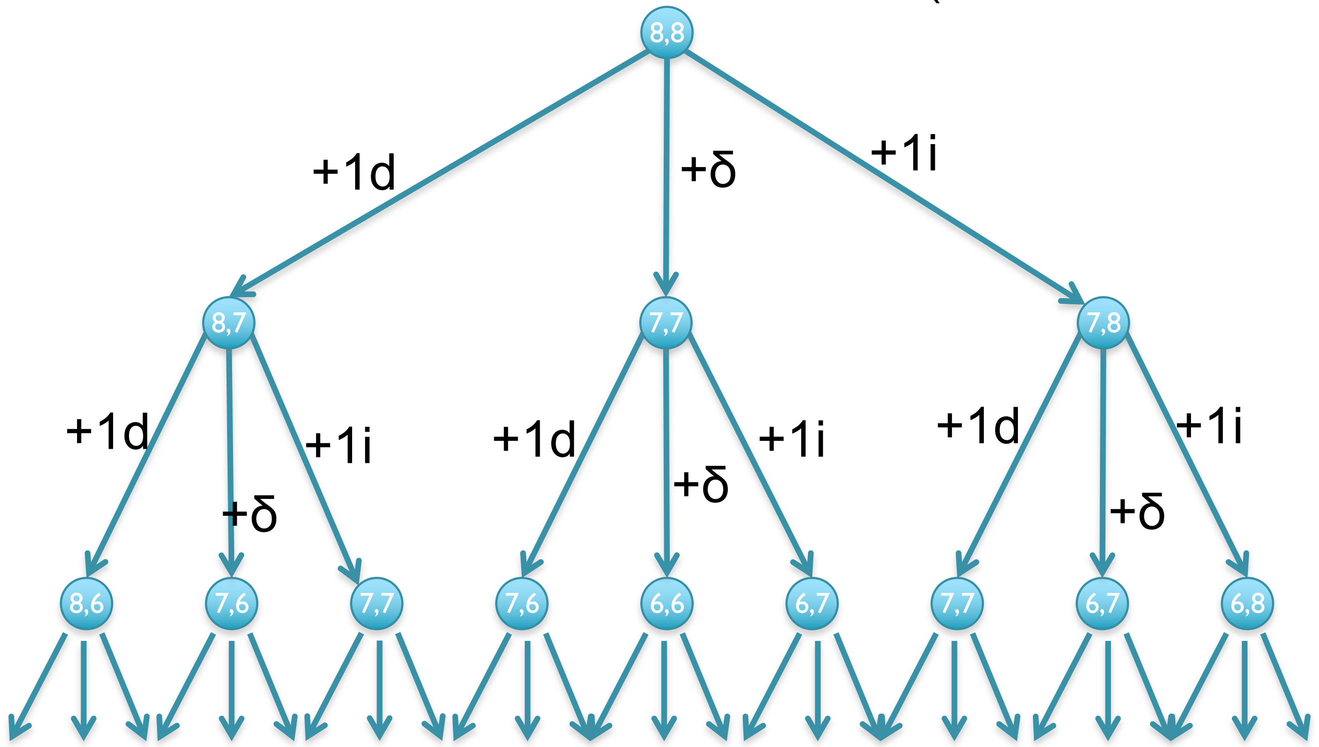
```
…M…       …I…       …D…
…X…       …-…       …X…
…Y…       …Y…       …-…
```

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:

```
D(i,j) = min {
        D(i-1,j) + 1,           // align 0 chars from S, 1 from T
        D(i,j-1) + 1,           // align 1 chars from S, 0 from T
        D(i-1,j-1) + δ(S(i),T(j)) // align 1+1 chars
    }
```

[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]+δ(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
	$\underline{0}$	$\underline{1}$	$\underline{2}$	$\underline{3}$	$\underline{4}$	$\underline{5}$	$\underline{6}$	$\underline{7}$	8
A	1	0	1	2	3	4	5	6	$\underline{7}$
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
	$\underline{0}$	$\underline{1}$	$\underline{2}$	$\underline{3}$	$\underline{4}$	5	6	7	8
A	1	0	1	2	3	$\underline{4}$	5	6	7
G	2	1	1	2	3	4	$\underline{5}$	$\underline{6}$	$\underline{7}$
C	3								
A	4								
C	5								
A	6								
C	7								
A	8								

D[AG,ACACACTA] = 7

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

Dynamic Programming Matrix

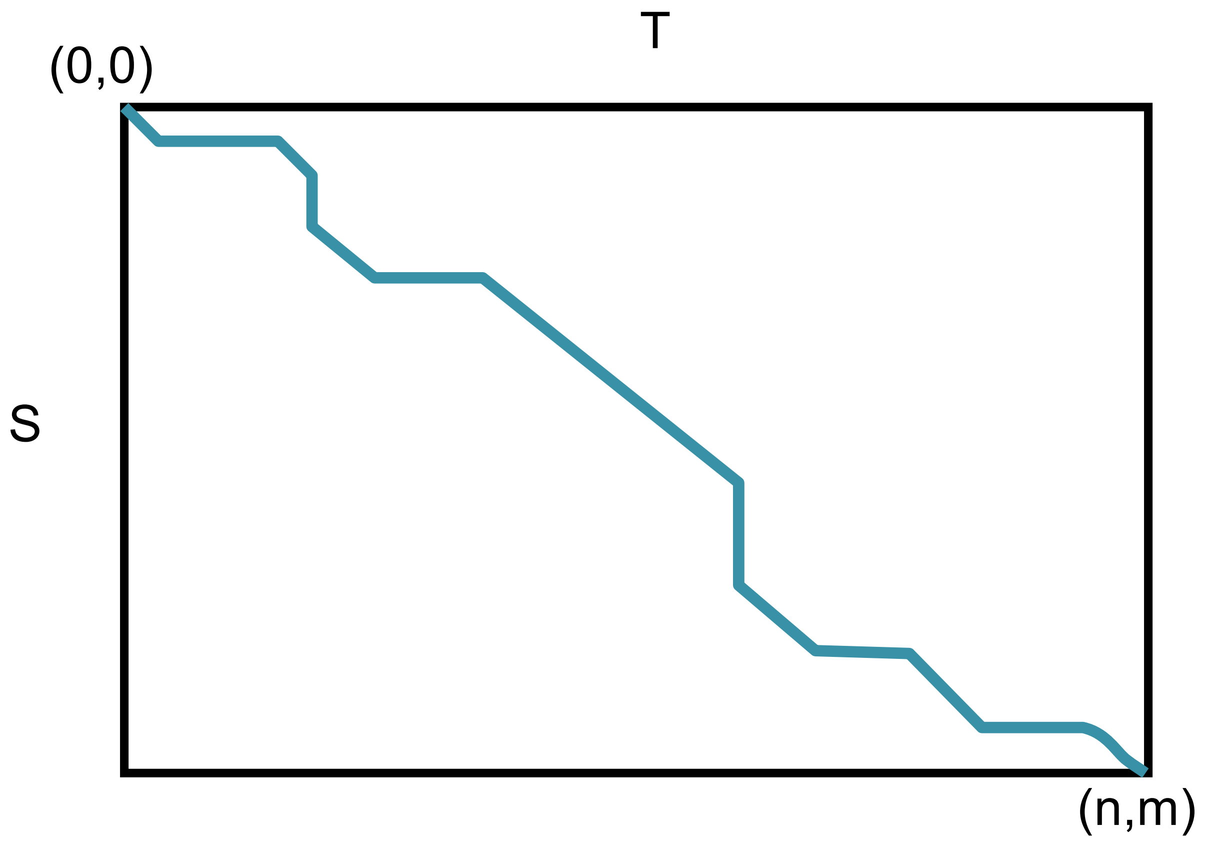
		A	C	A	C	A	C	T	A
	$\underline{0}$	1	2	3	4	5	6	7	8
A	1	$\underline{0}$	1	2	3	4	5	6	7
G	2	$\underline{1}$	1	2	3	4	5	6	7
C	3	2	$\underline{1}$	2	2	3	4	5	6
A	4	3	2	$\underline{1}$	2	2	3	4	5
C	5	4	3	2	$\underline{1}$	2	2	3	4
A	6	5	4	3	2	$\underline{1}$	2	3	3
C	7	6	5	4	3	2	$\underline{1}$	$\underline{2}$	3
A	8	7	6	5	4	3	2	2	$\underline{2}$

D[AGCACACA,ACACACTA] = 2

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

[Can we do it any better?]

Global Alignment Schematic



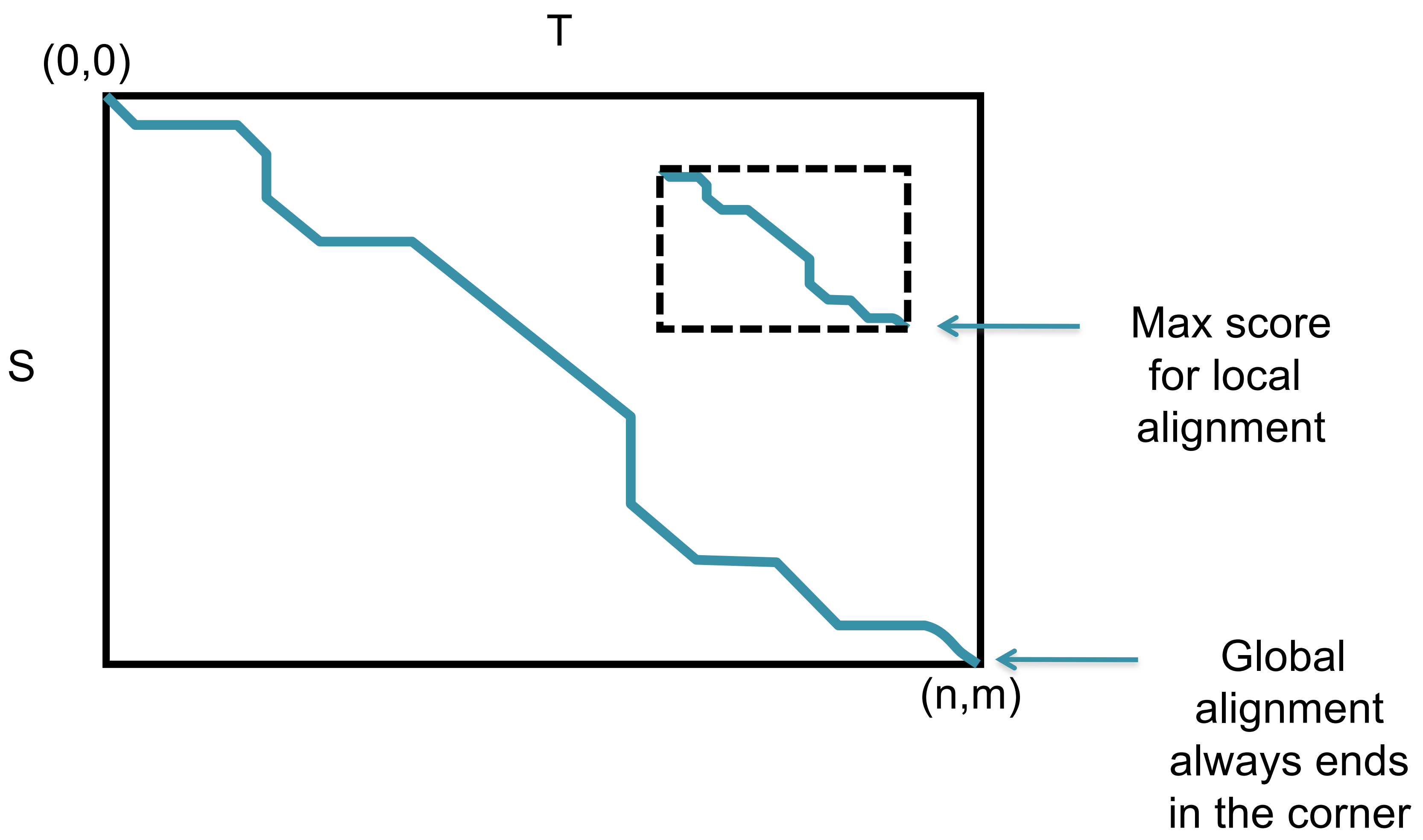
- A high quality alignment will stay close to the diagonal
  - If we are only interested in high quality alignments, we can skip filling in cells that can't possibly lead to a high quality alignment
  - Find the global alignment with at most edit distance d: O(2dn)

Local vs. Global Alignment

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

- The <u>Local Alignment Problem</u> 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


T
(0,0)
S
(n,m)
Max score for local alignment
Global alignment always ends in the corner

Nathan Edwards

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

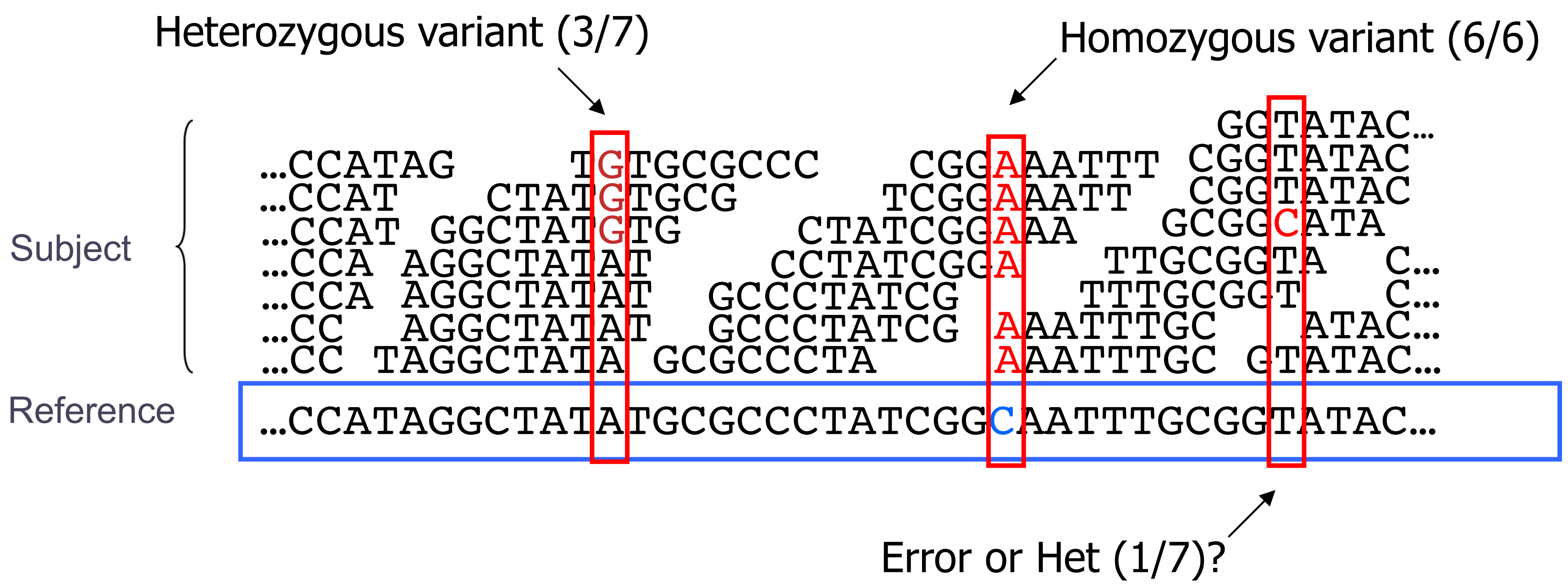
```
                  tccCAGTTATGTCAGgggacacgagcatgcagagac
                     ||||||||||||
aattgccgccgtcgttttcagCAGTTATGTCAGatc
```

Part 3: Variant Calling

Variant Calling Overview

$$\begin{tikzcd} {} \arrow[r] & \text{FASTQ} \arrow[r] & \text{Align (BWA)} \arrow[r] & \text{BAM} \arrow[r] & \text{Detect SNP/INDELs (GATK or FreeBayes)} \arrow[r] & \text{VCF} \end{tikzcd}$$

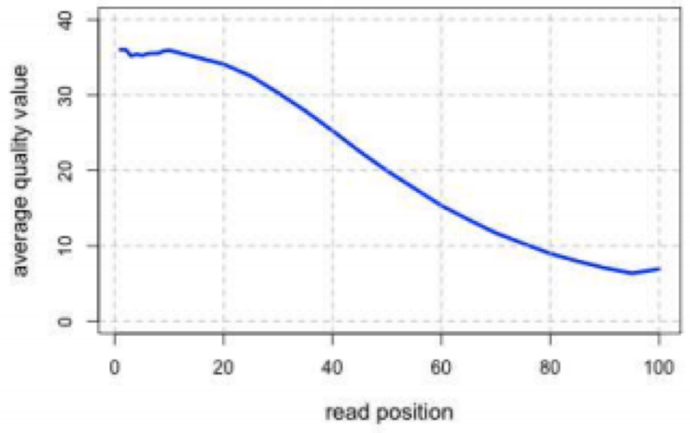
Genotyping Theory


Heterozygous variant (3/7)
Homozygous variant (6/6)
Subject
GGTATAC…
…CCATAG TGTGCGCCC CGGAAATTT CGGTATAC
…CCAT CTATGTGCG TCGGAAATT CGGTATAC
…CCAT GGCTATGTG CTATCGGAAA GCGGCATA
…CCA AGGCTATAT CCTATCGGA TTGCGGTA C…
…CCA AGGCTATAT GCCCTATCG TTTGCGGT C…
…CC AGGCTATAT GCCCTATCG AAATTTGC ATAC…
…CC TAGGCTATA GCGCCCTA AAATTTGC GTATAC…
Reference
…CCATAGGCTATATGCGCCCTATCGGCAATTTGCGGTATAC…
Error or Het (1/7)?

- 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



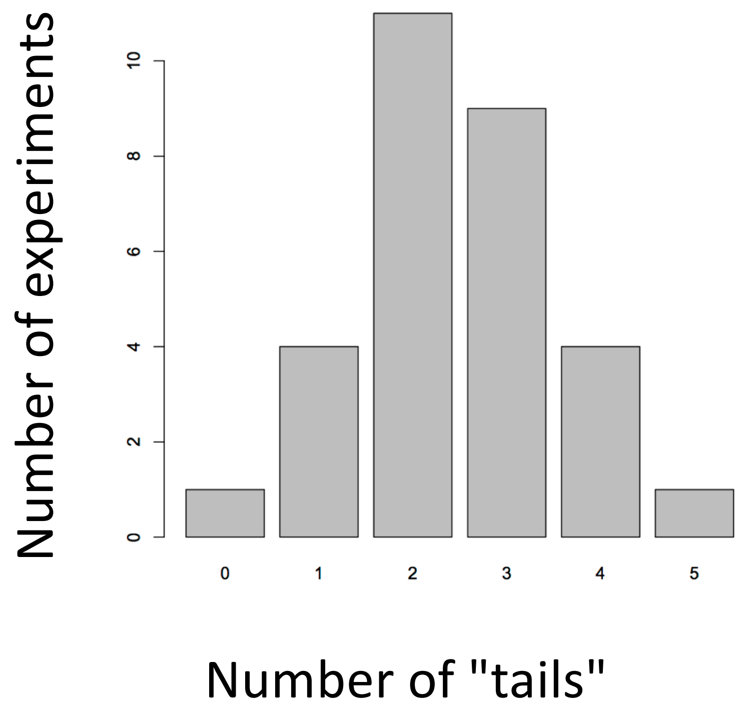
The Binomial Distribution: Adventures in Coin Flipping

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)?

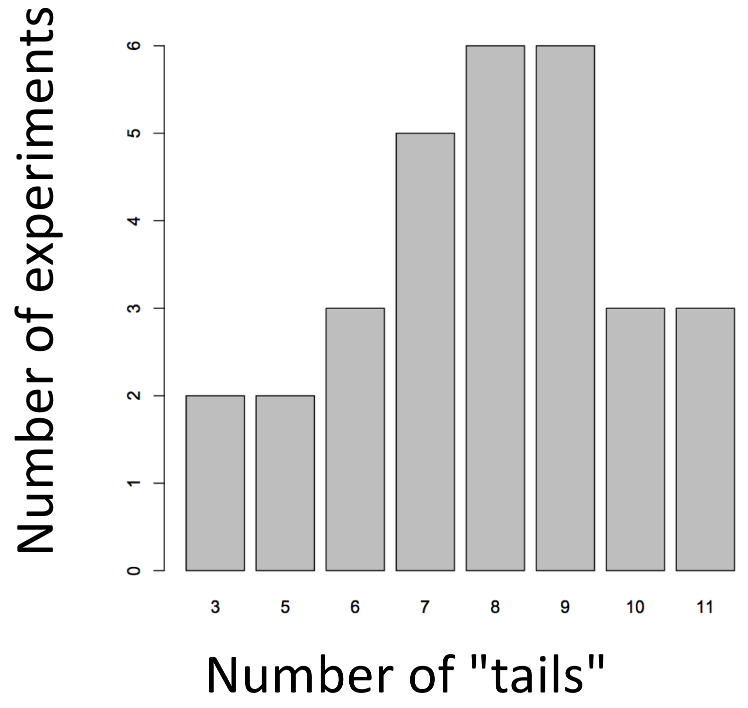


R code:

```
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:

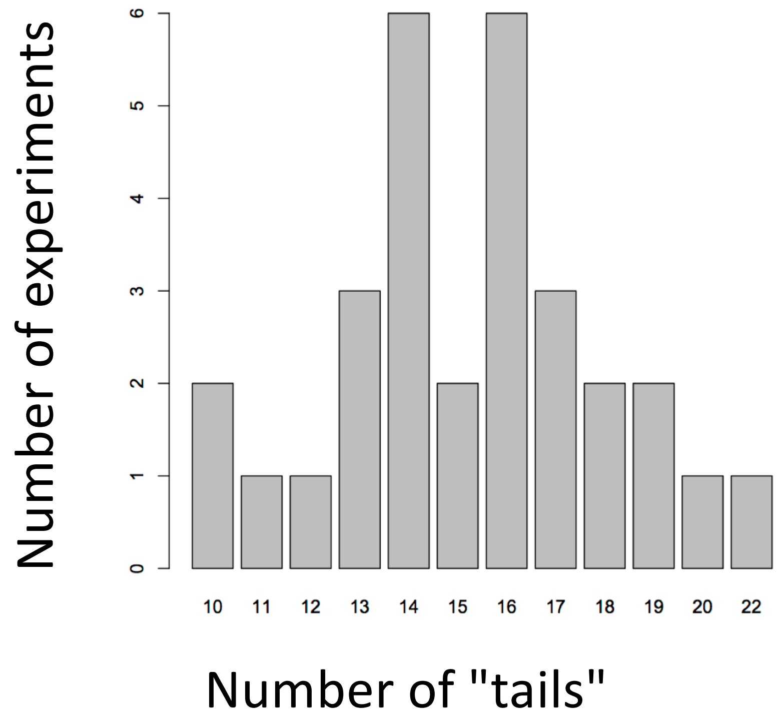
```
barplot(table(rbinom(30, 15, 0.5)))
```

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:

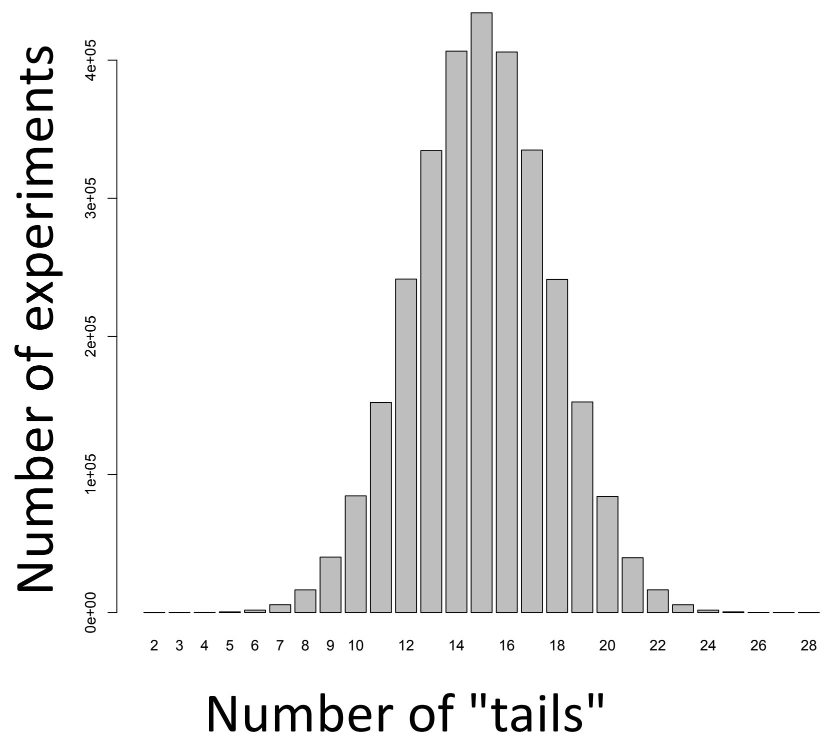
```
barplot(table(rbinom(30, 30, 0.5)))
```

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:

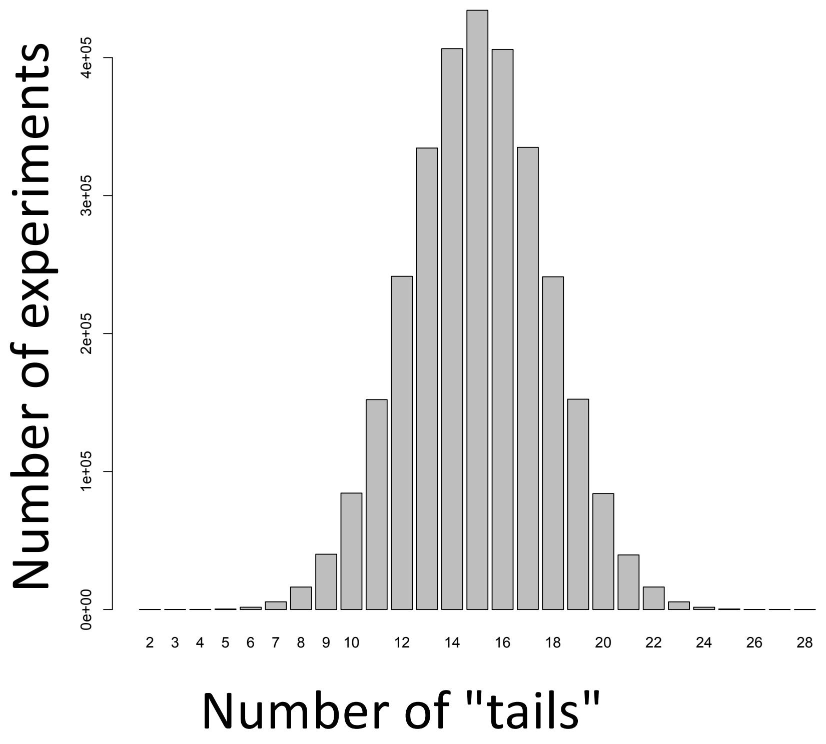
```
barplot(table(rbinom(3e6, 30, 0.5)))
```

3M experiments (students tossing coins)

30 tosses each

Probability of Tails

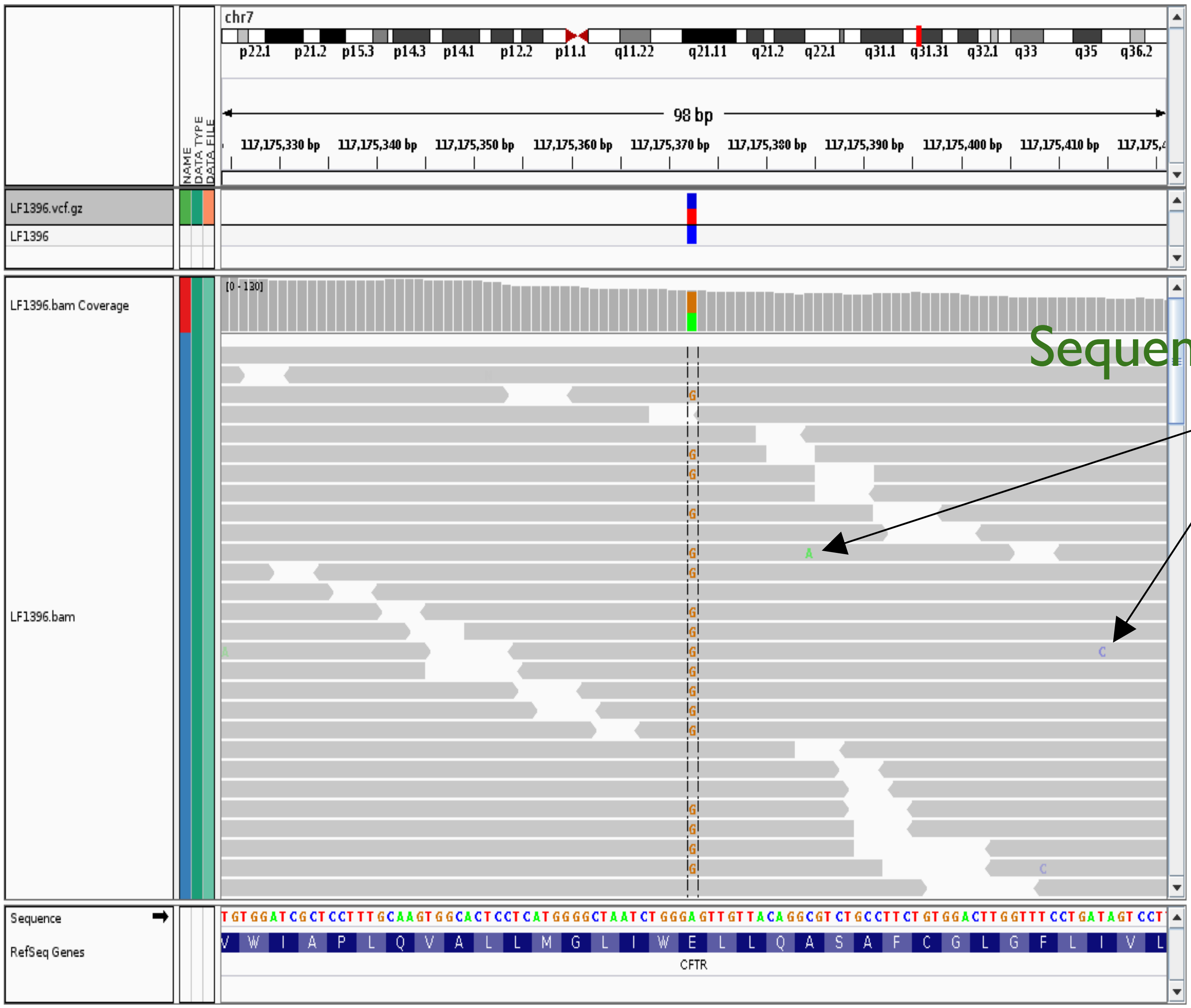
So, with 30 tosses (reads), we are much more likely to see an even mix of alternate and reference alleles at a heterozygous locus in a genome



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

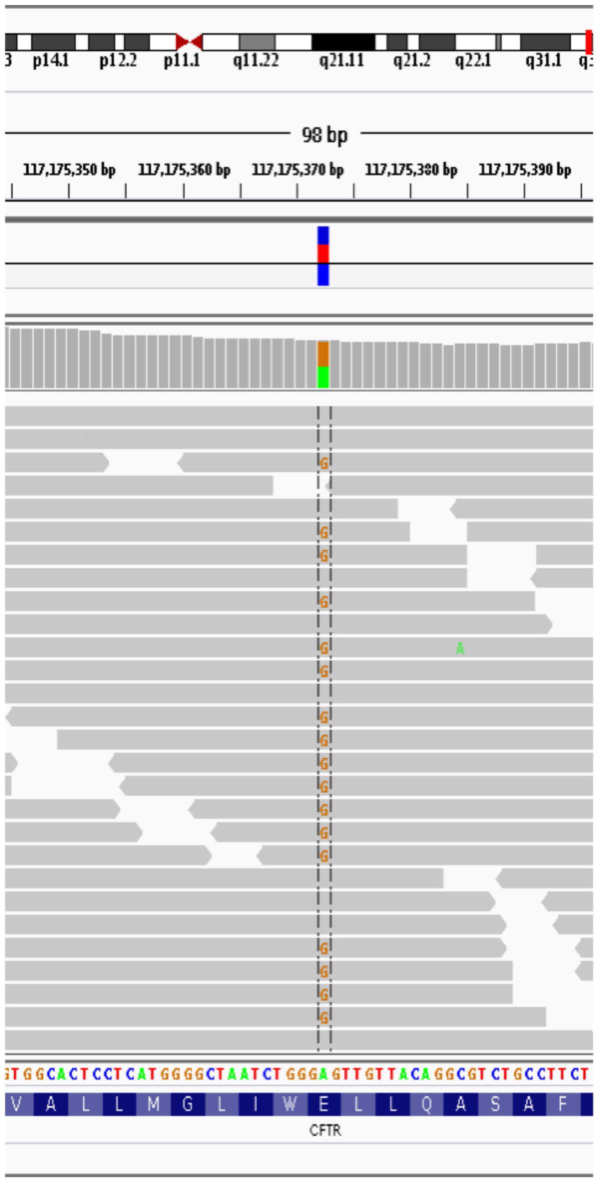
P(3/30 het) <?> P(3/30 err)

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


chr7
p22.1 p21.2 p15.3 p14.3 p14.1 p12.2 p11.1 q11.22 q21.11 q21.2 q22.1 q31.1 q31.31 q32.1 q33 q35 q36.2
98 bp
117,175,330 bp 117,175,340 bp 117,175,350 bp 117,175,360 bp 117,175,370 bp 117,175,380 bp 117,175,390 bp 117,175,400 bp 117,175,410 bp
NAME
DATA TYPE
DATA FILE
LF1396.vcf.gz
LF1396
LF1396.bam Coverage
[0 - 130]
LF1396.bam
Sequence
RefSeq Genes
CFTR
Sequencing errors

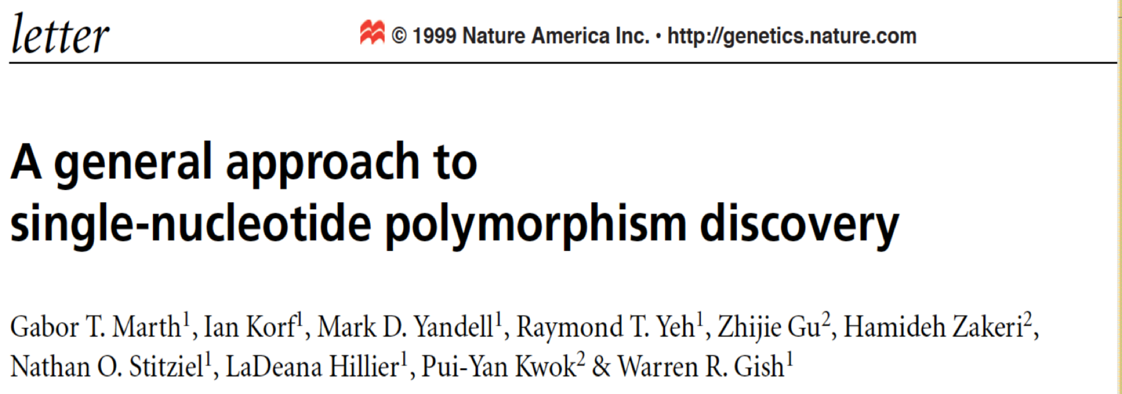
https://jchoigt.files.wordpress.com/2012/07/igv_e217g_snapshot.png

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

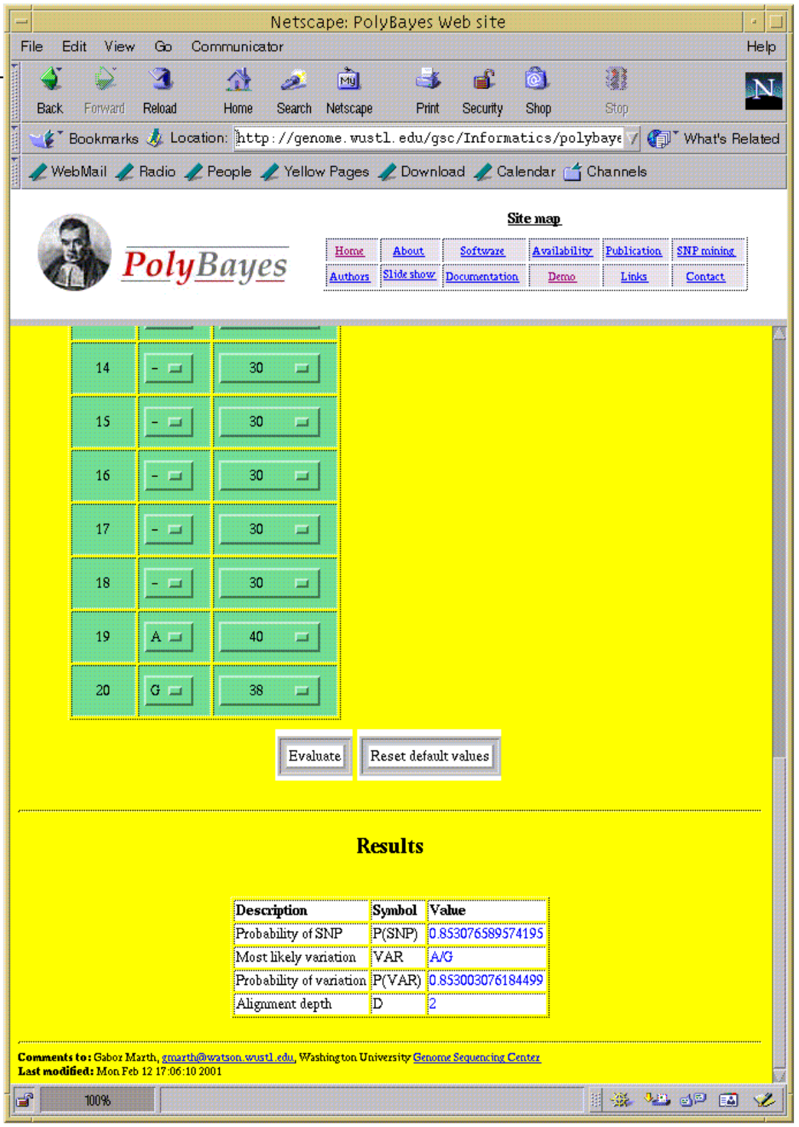
PolyBayes: The first statistically rigorous variant detection tool.



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]


Netscape: PolyBayes Web site
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://genome.wustl.edu/gsc/Informatics/polybaye What's Related
WebMail Radio People Yellow Pages Download Calendar Channels
PolyBayes
Site map
Home About Software Availability Publication SNP mining
Authors Slide show Documentation Demo Links Contact
Evaluate
Reset default values
100%

14	-	30
15	-	30
16	-	30
17	-	30
18	-	30
19	A	40
20	G	38

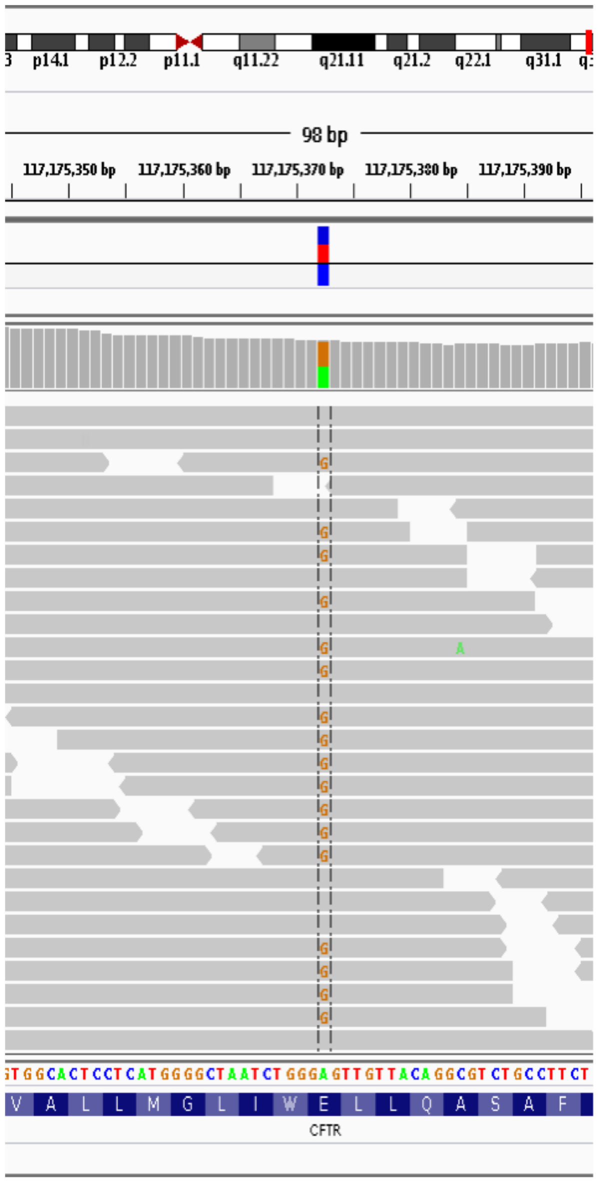
Results

Description	Symbol	Value
Probability of SNP	P(SNP)	0.853076589574195
Most likely variation	VAR	A/G
Probability of variation	P(VAR)	0.853003076184499
Alignment depth	D	2

Comments to: Gabor Marth, gmarth@watson.wustl.edu, Washington University Genome Sequencing Center
Last modified: Mon Feb 12 17:06:10 2001

Its main innovation was the use of Bayes's theorem

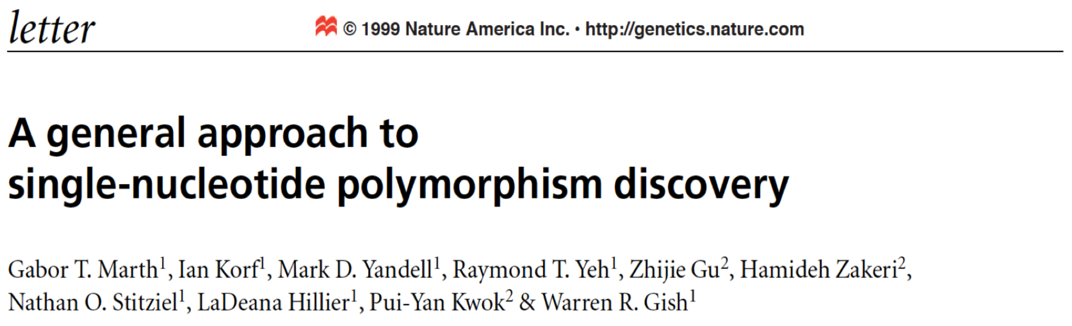
Bayesian SNP calling



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

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

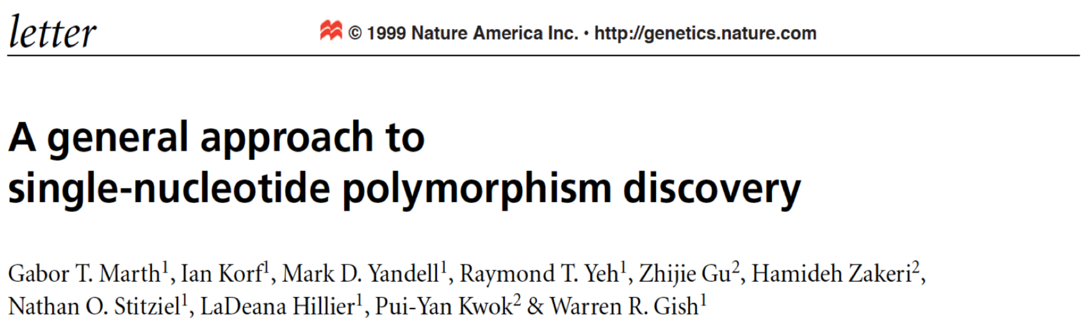
Base call + Base quality

Expected (prior) polymorphism rate

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

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

PolyBayes: The first statistically rigorous variant detection tool.



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

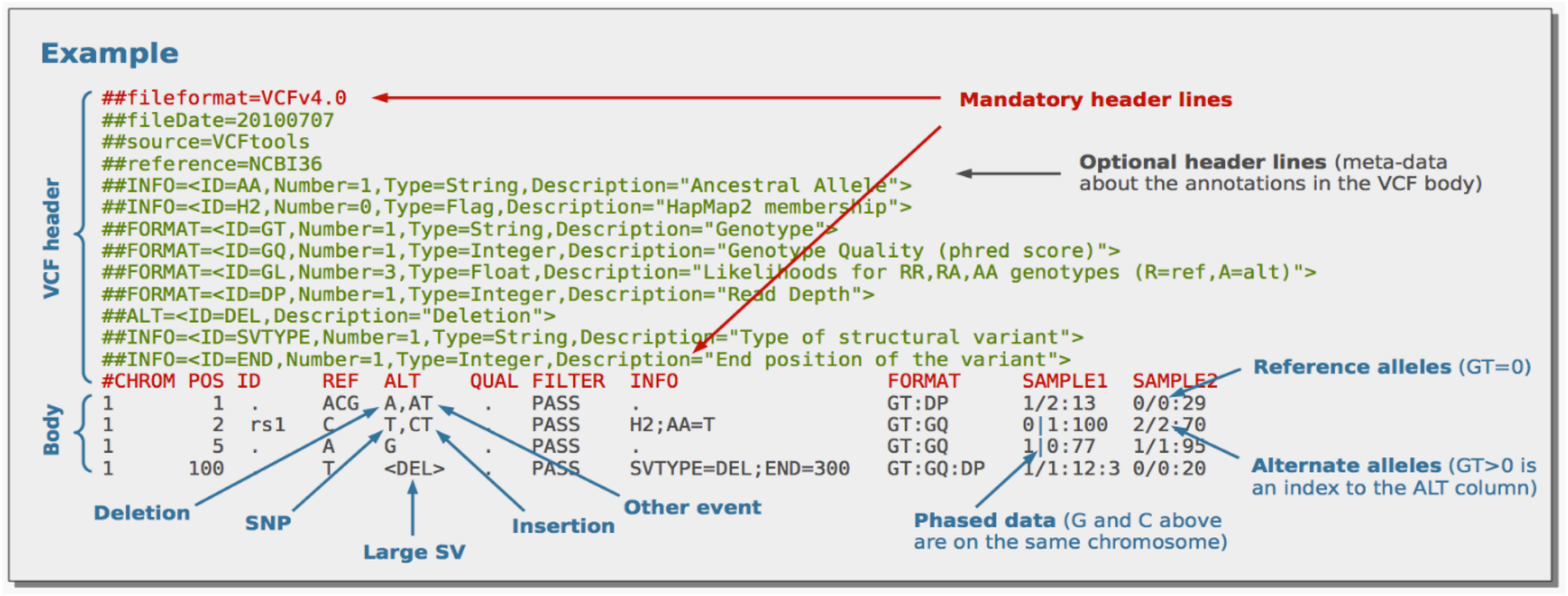
A general approach to single-nucleotide polymorphism discovery

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

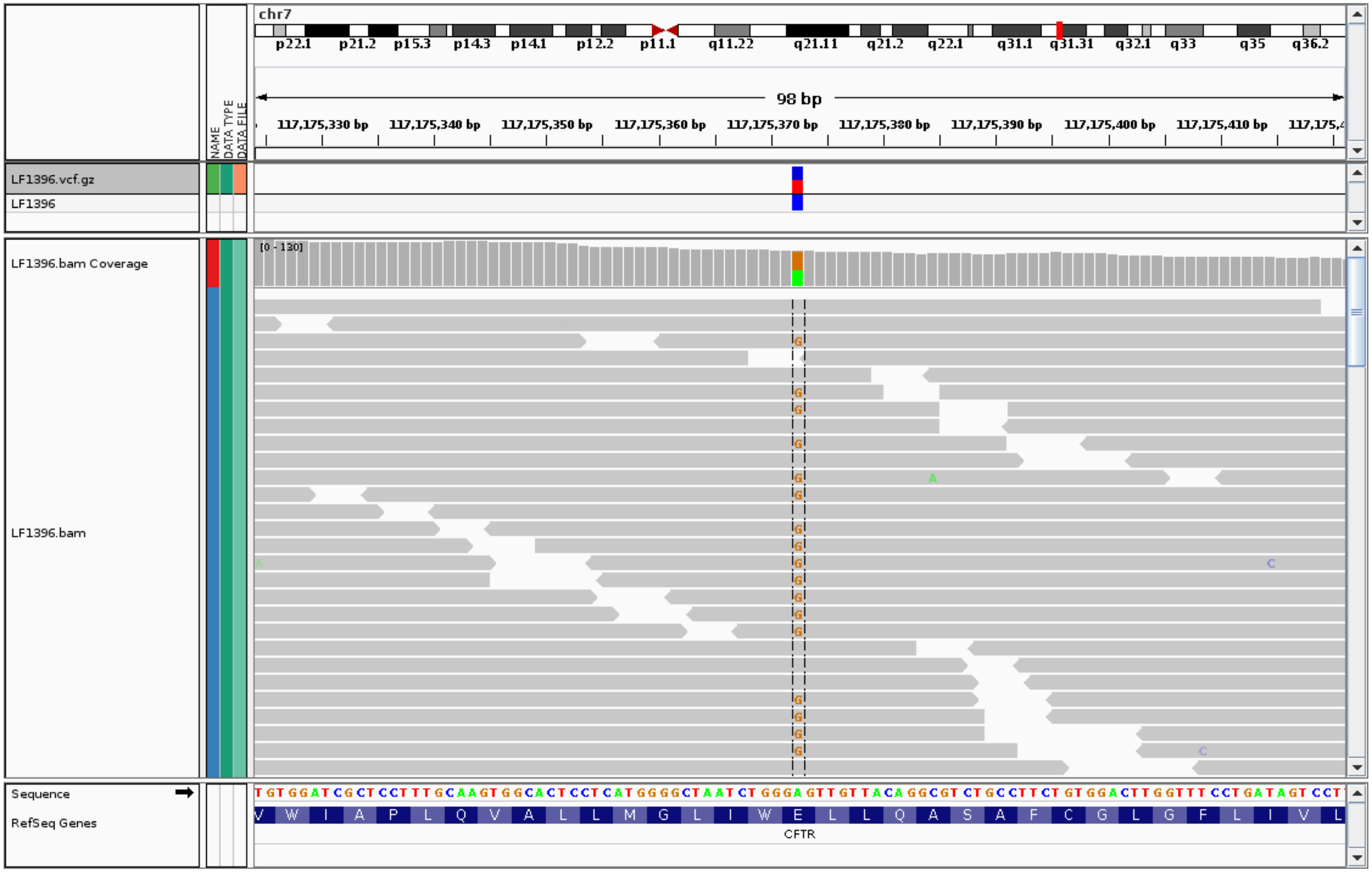
http://www.nature.com/ng/journal/v23/n4/full/ng1299_452.html

VCF Format


Example
VCF header
Body
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)
Phased data (G and C above are on the same chromosome)
Deletion
SNP
Large SV
Insertion
Other event

```
##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
```

VCF Format



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