Variant Analysis

Michael Schatz

October 9, 2023

Lecture 12: Applied Comparative Genomics

Assignment 4: BWT and Modimizers
Due Wednesday Oct 11 by 11:59pm

Not Secure — schatz-lab.org/appliedgenomics2023/assignments/assignment4/

appliedgenomics2023

Assignment 4: BWT and Modimizers

Assignment Date: Wednesday, October 4, 2023
Due Date: Wednesday, October 11, 2023 @ 11:59pm

Assignment Overview

In this assignment you will implement the BWT and modimizers. These exercises can be computed in any programming language, although we recommend python (or C++, Java, or Rust). R is generally inefficient at string processing unless you take great care.

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

Question 1. BWT Encoding [25 pts]

In the language of your choice, implement a BWT encoder and encode the string below. Faster (Linear time) methods exist for computing the BWT, although for this assignment you can use the simple method based on standard sorting techniques. Your solution does *not* need to be an optimal algorithm and can use O(n^2) space and O(n^2 lg n) time.

Here is the recommended pseudo code (make sure to submit your code as well as the encoded string):

```
computeBWT(string s)
  ## add the magic end-of-string character
  s = s + "$"

  ## build up the BWM from the cyclic permutations
  ## note the ith cyclic permutation is just "s[i..n] + s[0..i]"
  rows = []
  for (i = 0; i < length(s); i++)
    rows.append(cyclic_permutation(s, i))

  ## just use the builtin sort command to sort the cyclic permutations
  sort(rows)

  ## now extract the last column
  bwt = ""
  for (i = 0; i < length(s); i++)
    bwt += substr(rows[i], length(s), 1)
  return bwt
```

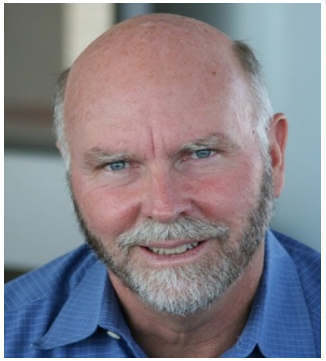
String to encode:

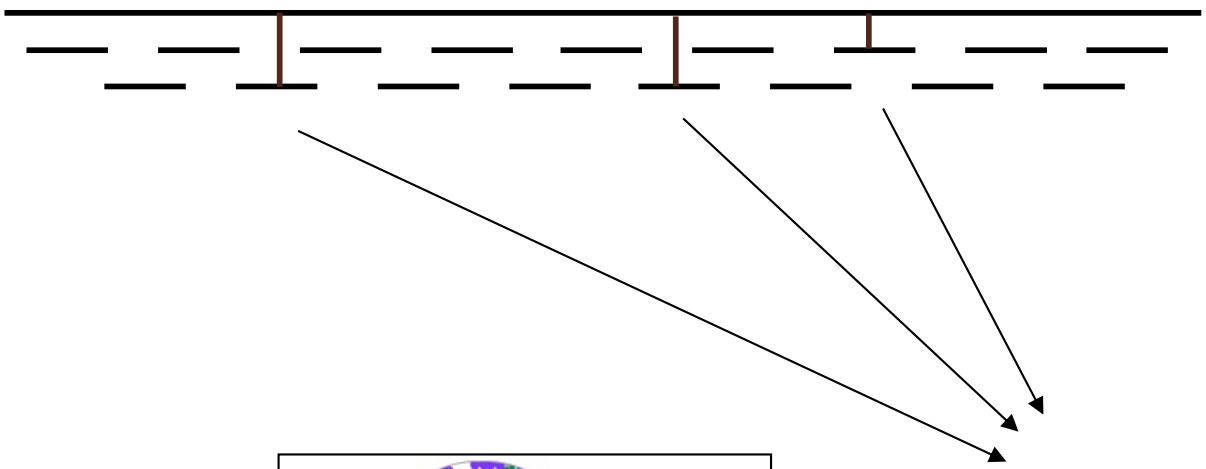
https://github.com/schatzlab/appliedgenomics2023/tree/main/assignments/assignment4

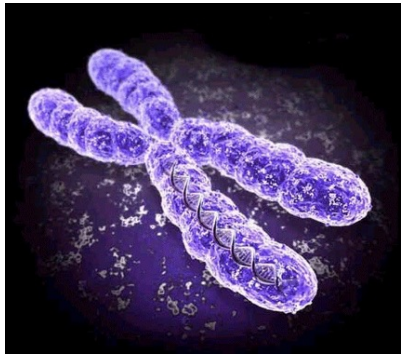
Check Piazza for questions!

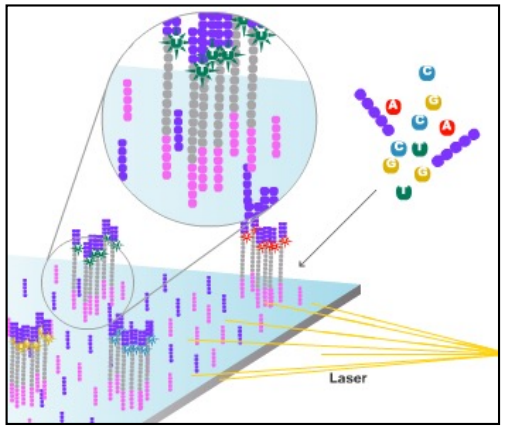
Personal Genomics

How does your genome compare to the reference?









Heart Disease

Cancer

Presidential smile

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

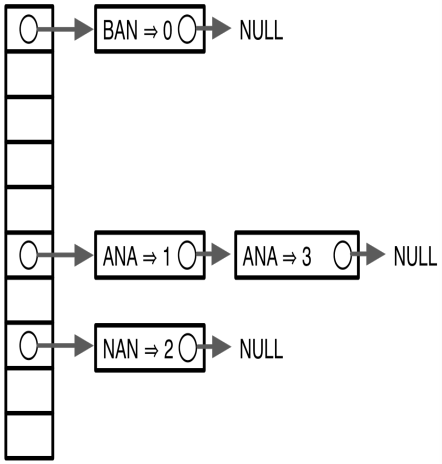
Suffix Array (>15 GB)

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

BWT (3 GB)

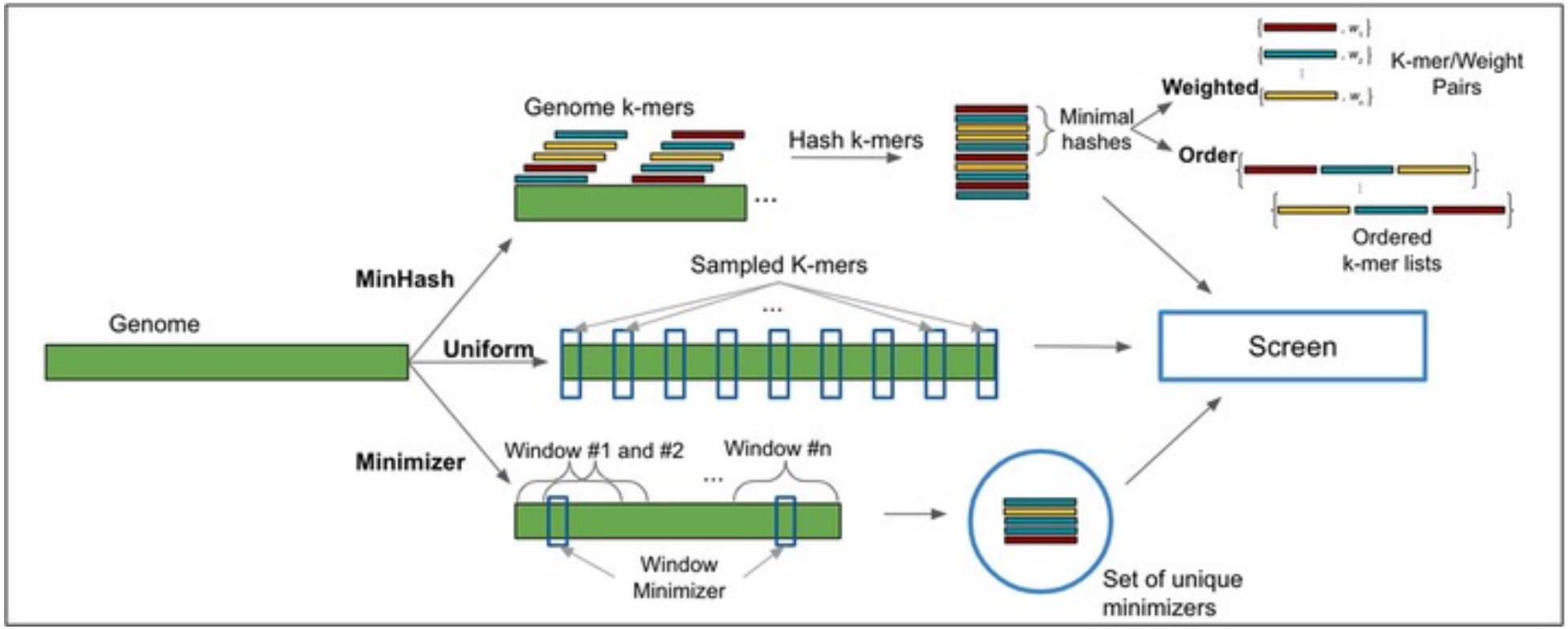
```
BANANA$
   =>
$BANANA
A$BANAN
ANA$BAN
ANANA$B
BANANA$
NA$BANA
NANA$BA
   =>
ANNB$AA
```

O(m)

Full-text and concise

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

Minimizers, Modimizers, & MinHash



Sketching and sampling approaches for fast and accurate long read classification
Das and Schatz (2022) BMC Bioinformatics. doi: 10.1186/s12859-022-05014-0

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…

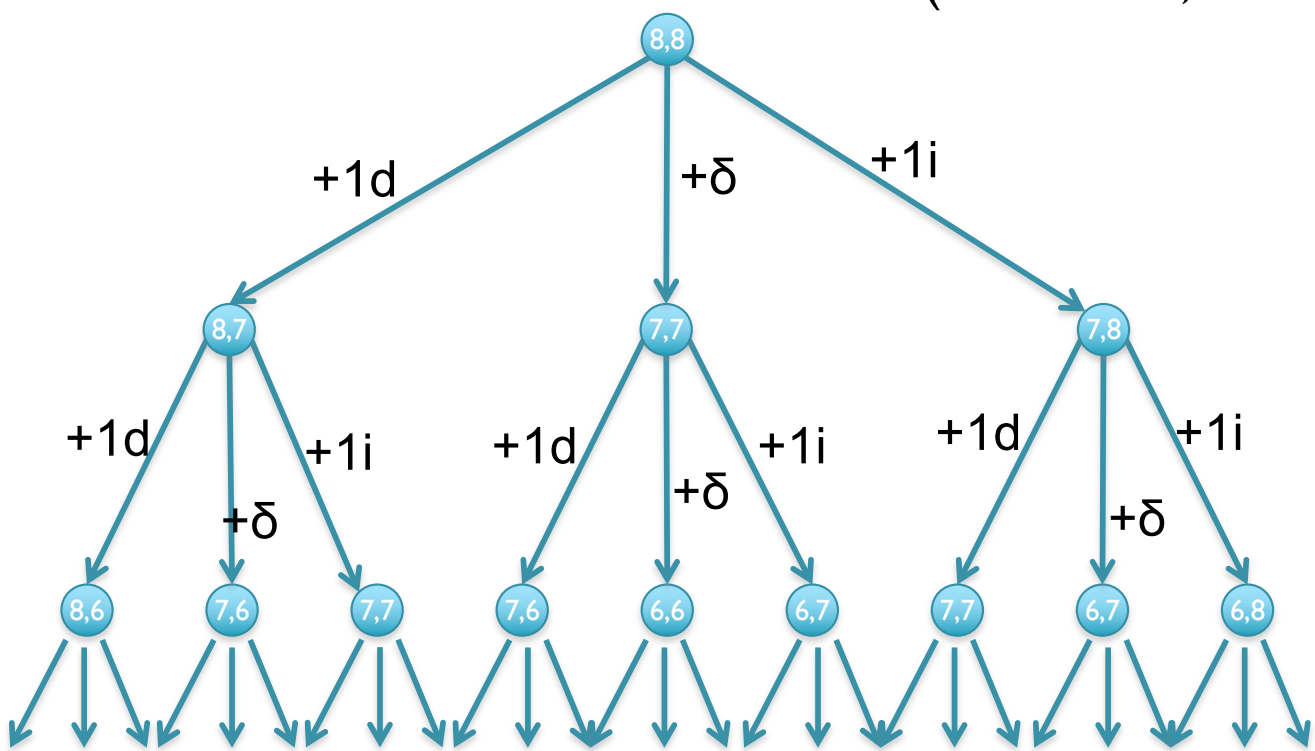
```
...M...      ...I...      ...D...
...X...      ...-...      ...X...
...Y...      ...Y...      ...-...
```

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) + I,
D(AGCACAC, ACACACTA) + I,
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 $i = 0,\ldots,n$
  - $D(0,j) = j$, for all $j = 0,\ldots,m$

- 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

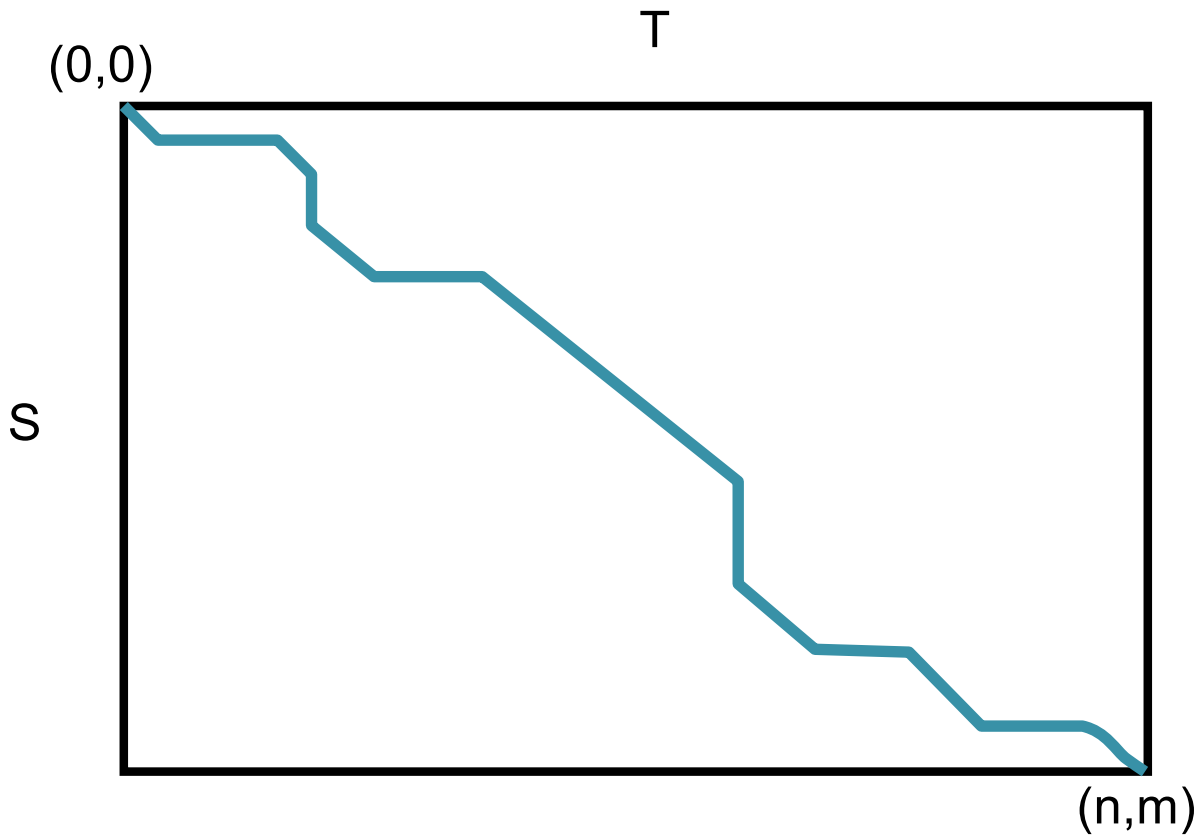
		A	C	A	C	A	C	T	A
	0	1	2	3	4	5	6	7	8
A	1	0	1	2	3	4	5	6	7
G	2	1	1	2	3	4	5	6	7
C	3	2	1	2	2	3	4	5	6
A	4	3	2	1	2	2	3	4	5
C	5	4	3	2	1	2	2	3	4
A	6	5	4	3	2	1	2	3	3
C	7	6	5	4	3	2	1	2	3
A	8	7	6	5	4	3	2	2	2

D[AGCACACA,ACACACTA] = 2

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

[Can we do it any better?]

Global Alignment Schematic



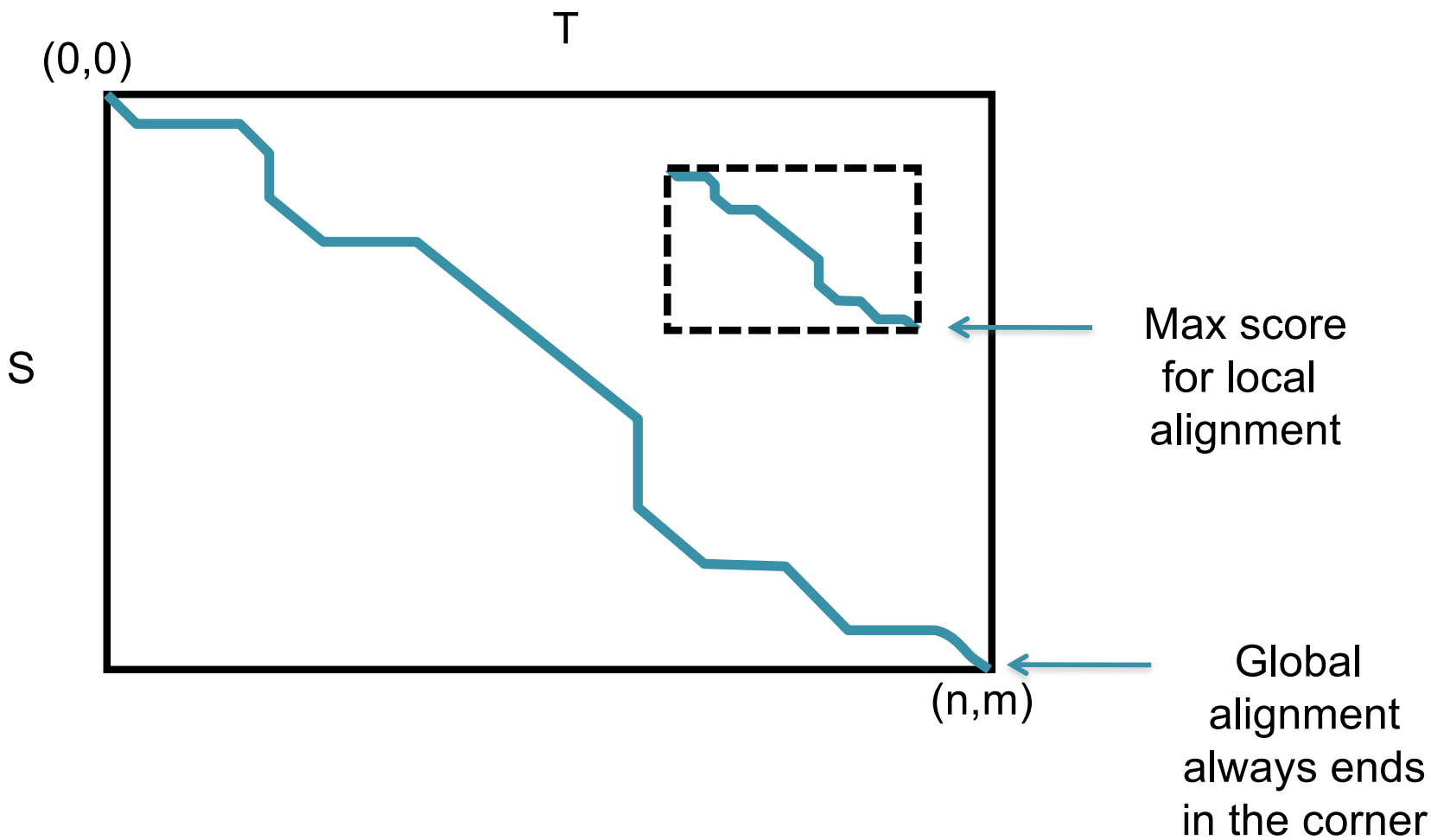
- A high quality alignment will stay close to the diagonal
  - If we are only interested in high quality alignments, we can skip filling in cells that can't possibly lead to a high quality alignment
  - Find the global alignment with at most edit distance d: O(2dn)

Local vs. Global Alignment

- The <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



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

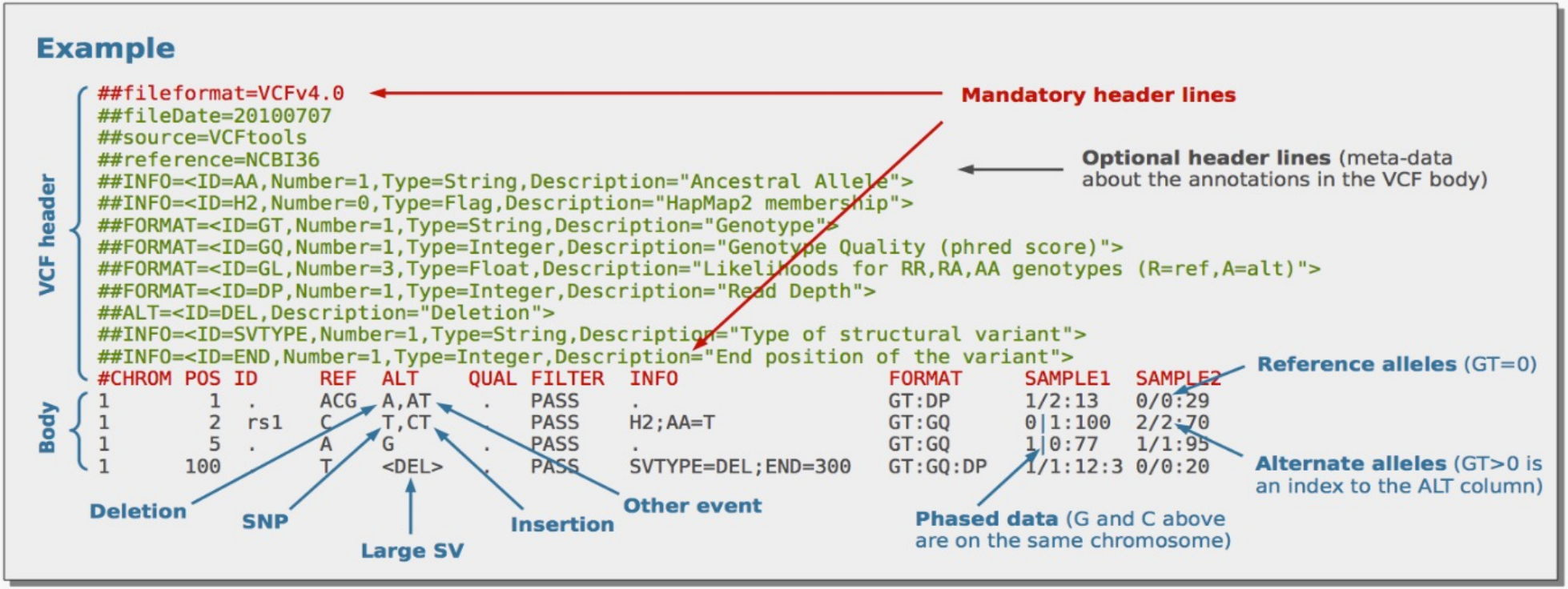
```
                tccCAGTTATGTCAGggacacgagcatgcagagac
                   ||||||||||||
aattgccgccgtcgttttcagCAGTTATGTCAGatc
```

Part 2: Variant Calling

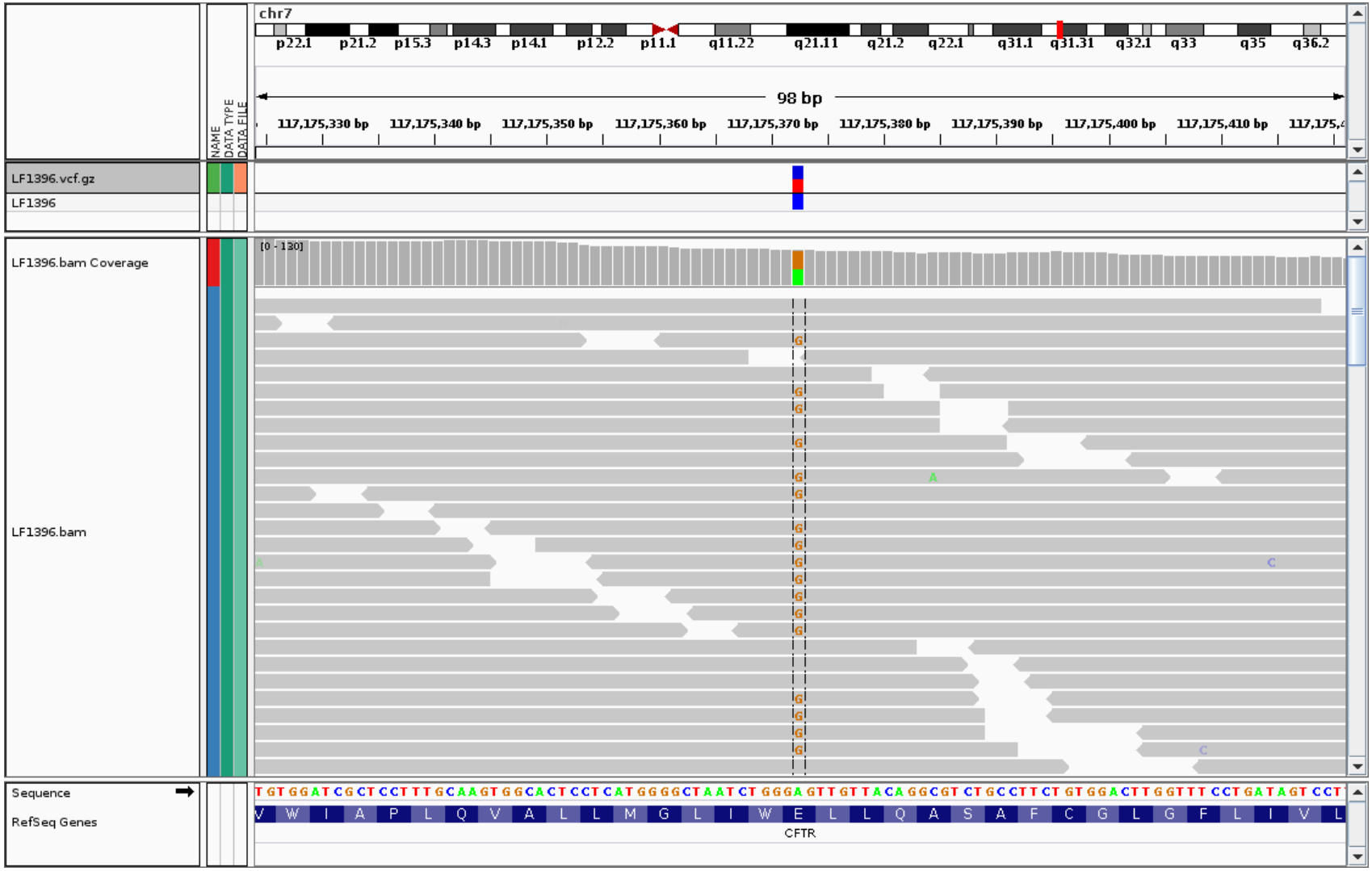
Variant Calling Overview

→ FASTQ → Align (BWA) → BAM → Detect SNP/INDELs (GATK or FreeBayes) → VCF

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

VCF Format



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

Genotyping Theory

Heterozygous variant (3/7)

Homozygous variant (6/6)

```
                                                    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…
```

Subject

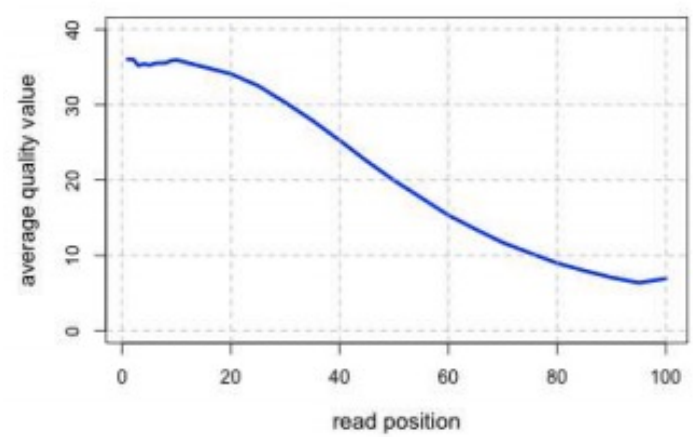
Reference

```
…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

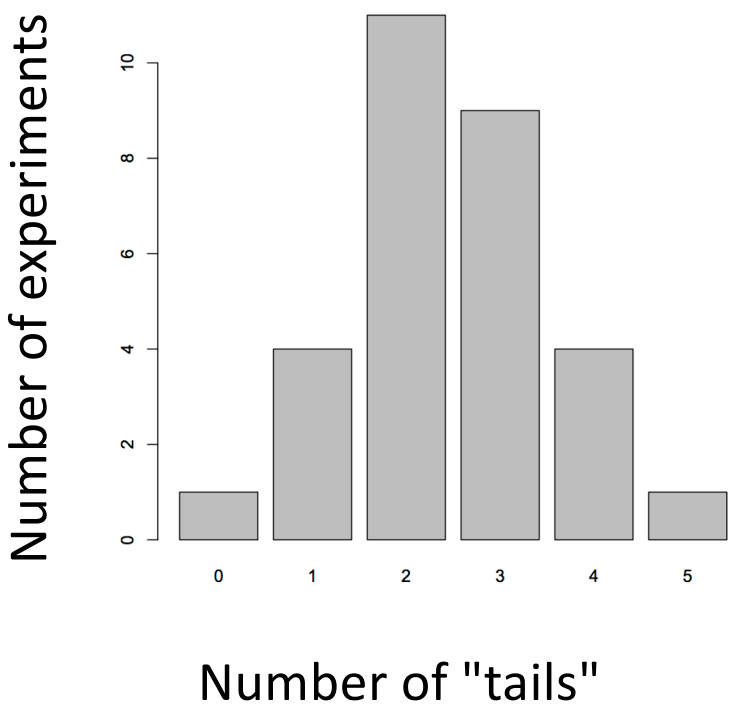
The Binomial Distribution: Adventures in Coin Flipping

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:

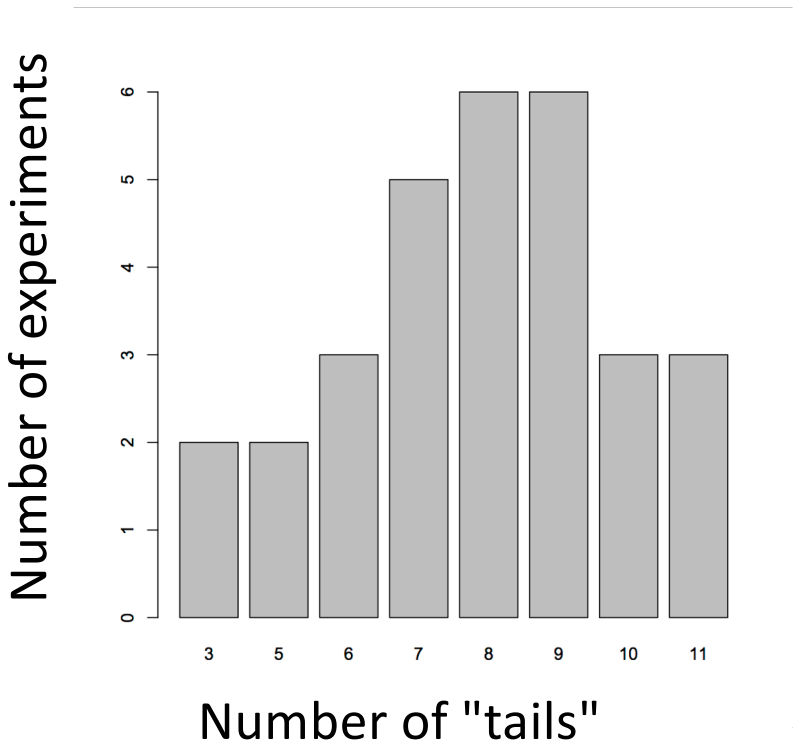
```
barplot(table(rbinom(30, 5, 0.5)))
```

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:

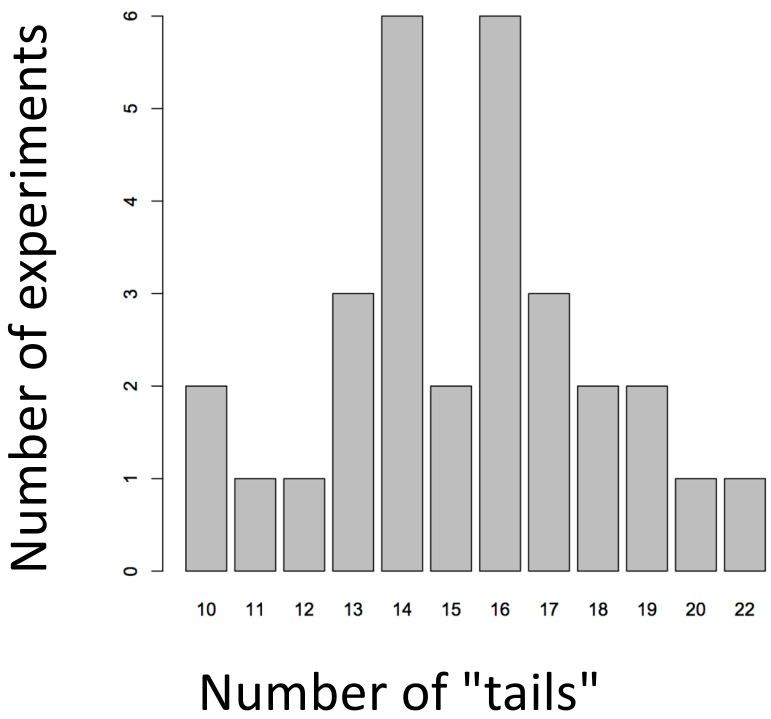
```
barplot(table(rbinom(30, 15, 0.5)))
```

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:

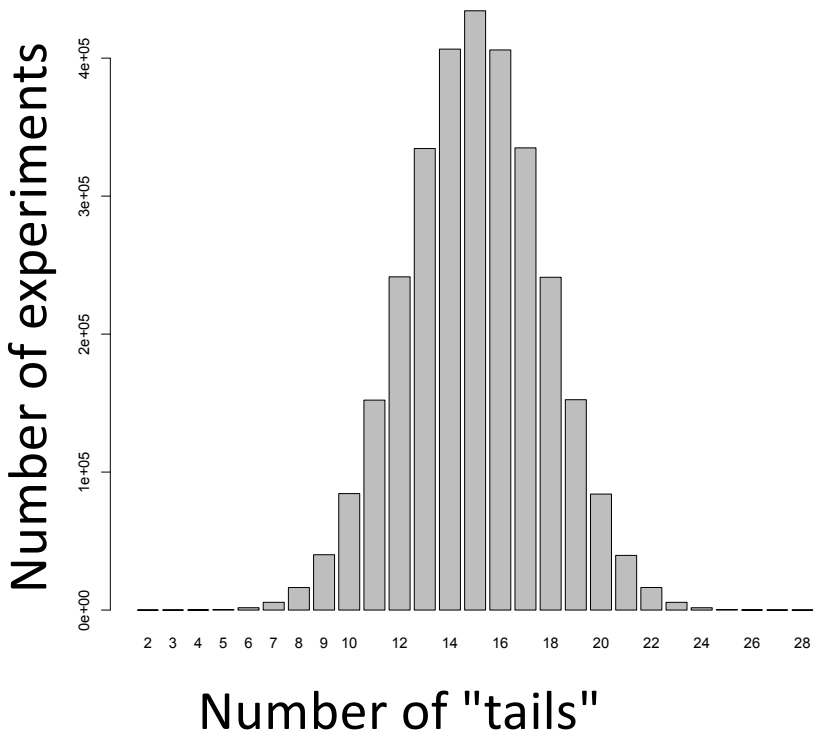
```
barplot(table(rbinom(30, 30, 0.5)))
```

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:

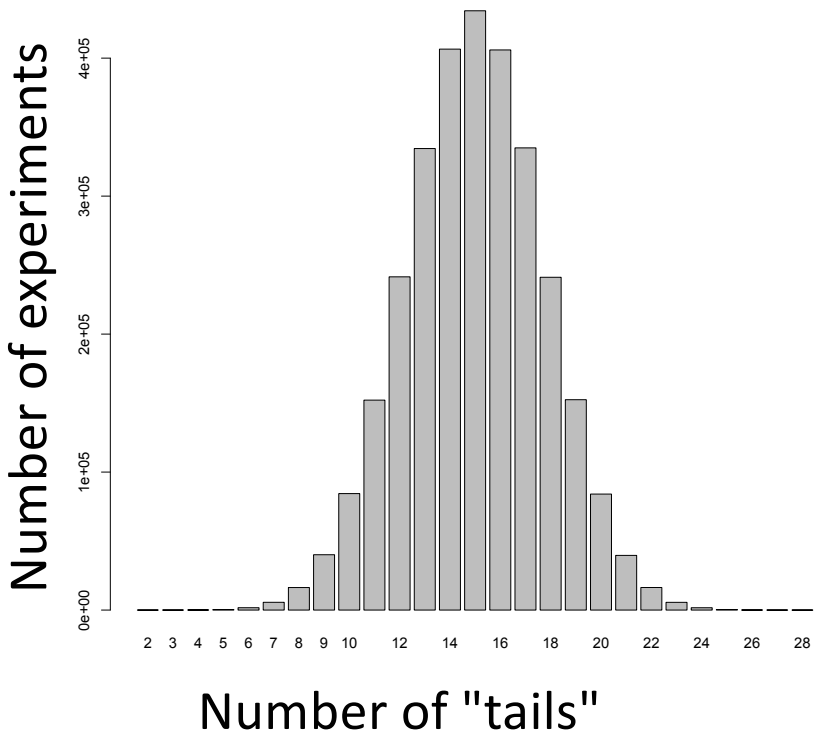
```
barplot(table(rbinom(3e6, 30, 0.5)))
```

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 <u>at least</u> 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)