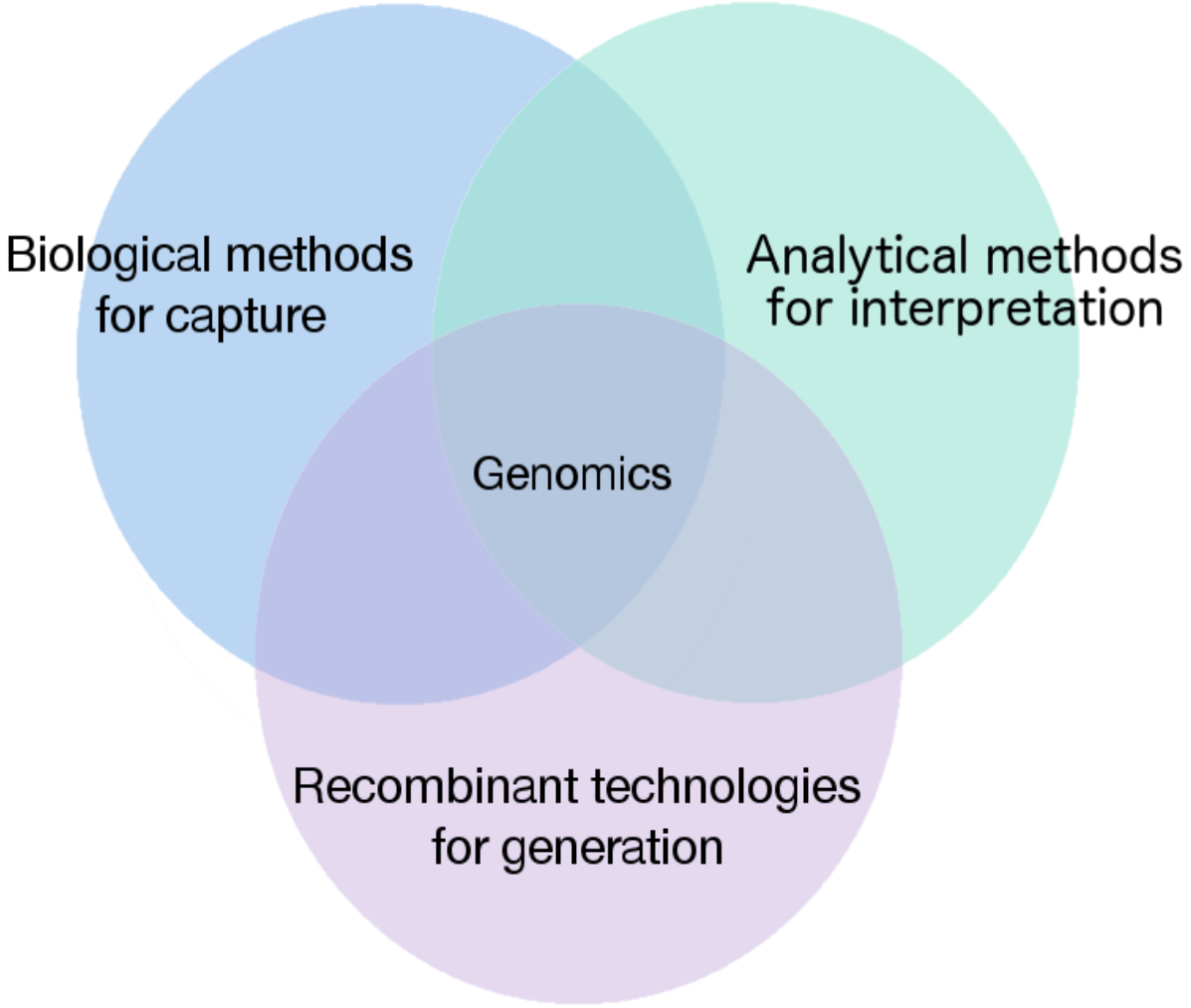
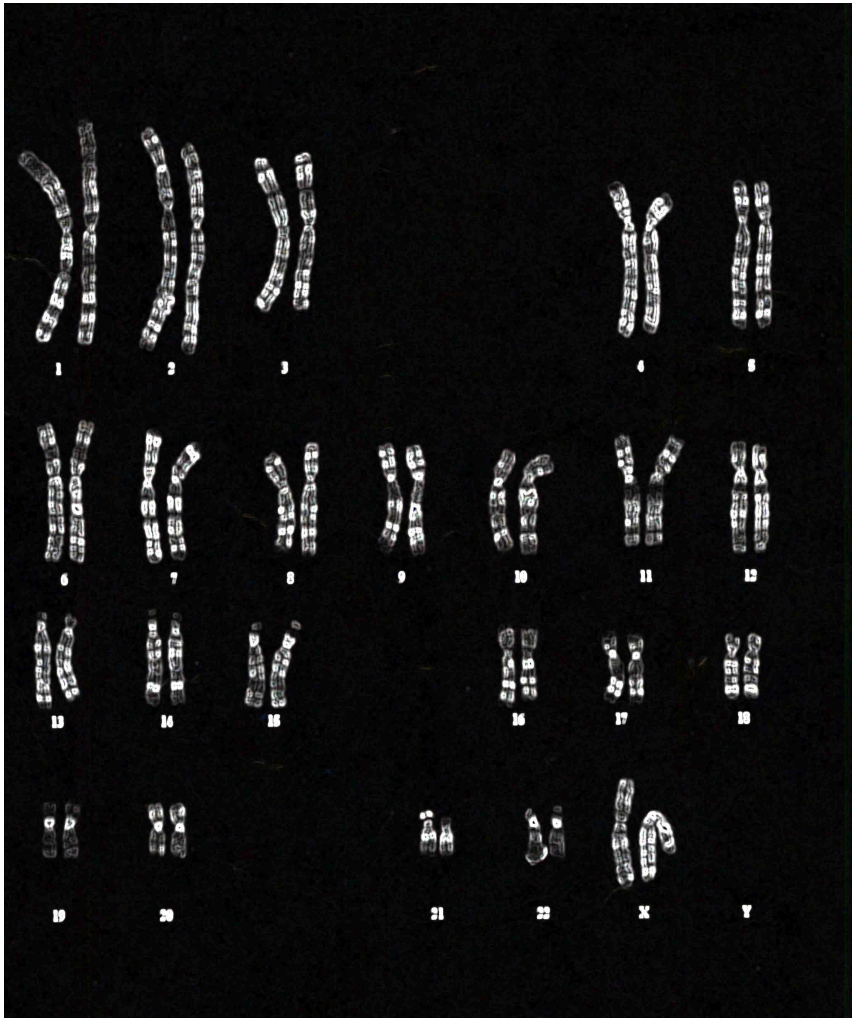
Computing for Genomics

Paint by Numbers and Beyond

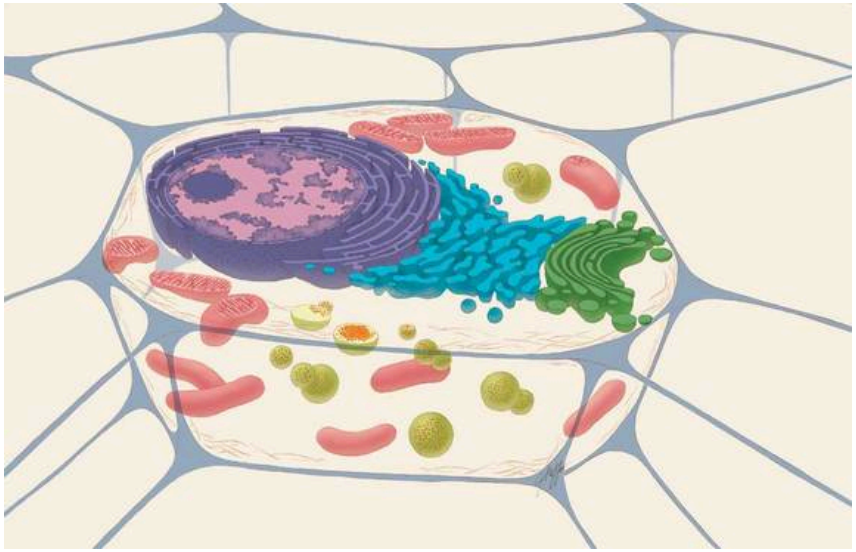
Genomics – The study of Genomes



The Genome



- The DNA in each cell
- ~3 billion base pairs
- Any two people are 99.6% to 99.9% the same

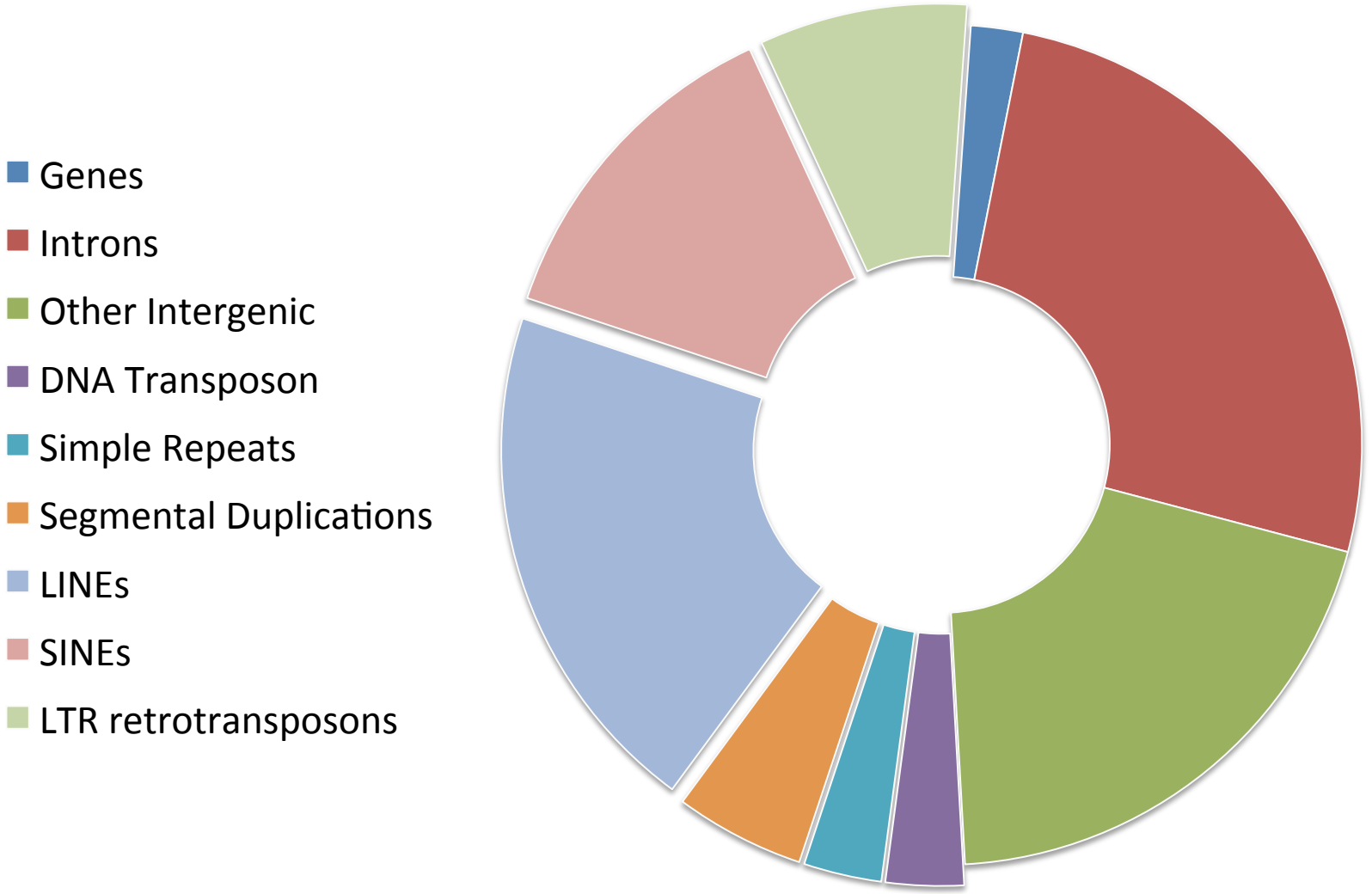


http://www.healthline.com/health-blogs/tech-medicine/creating-dna-art
http://i.livescience.com/images/i/000/017/621/i02/ITC_EukaryoticCell_Copy.jpg?1309355705

The Genome

- Genes (coding DNA)
- Noncoding
  - Regulatory regions
  - Structural
  - Repeat elements
  - Non coding RNA
  - Pseudogenes/Relics/Unclassified

Repeat Elements



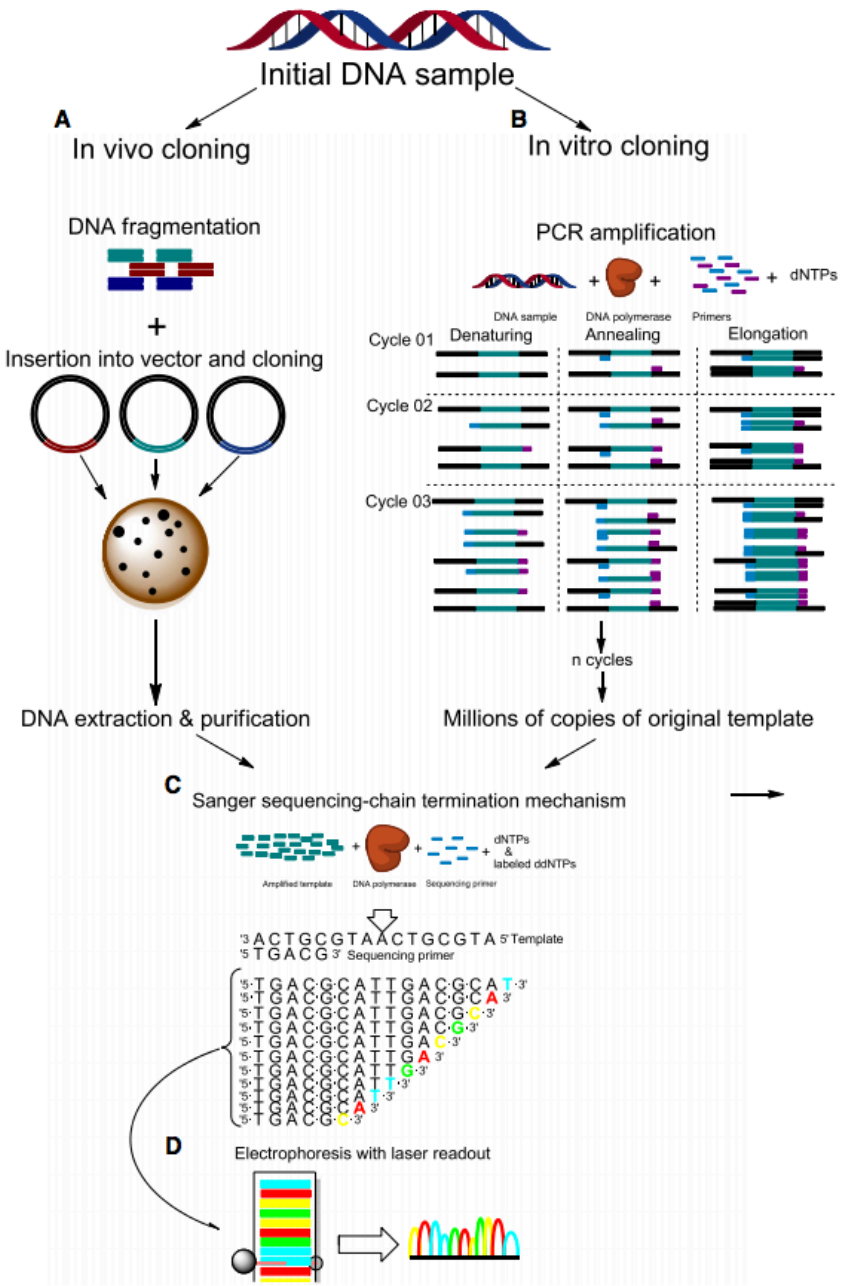
Gregory, T. Ryan. "Synergy between sequence and size in large-scale genomics." Nature Reviews Genetics 6.9 (2005): 699-708.

Human Variation

- Point mutations
  - SNPs
- Small insertions, deletions and indels
- Structural Variation
  - Copy Number Variation

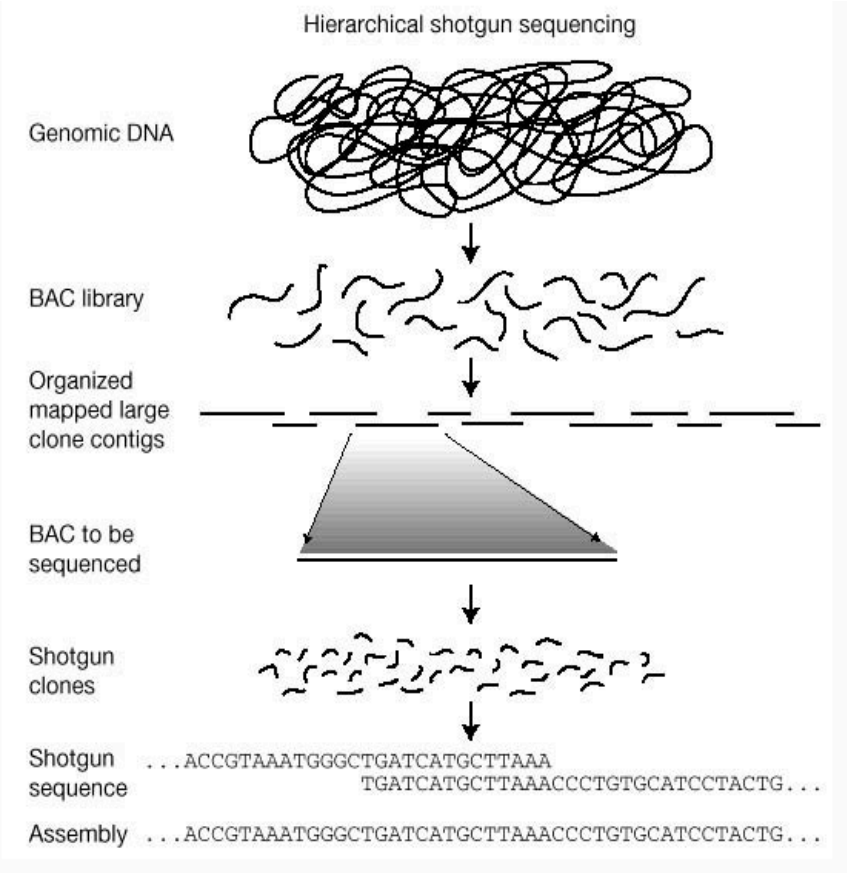
The Human Reference Genome

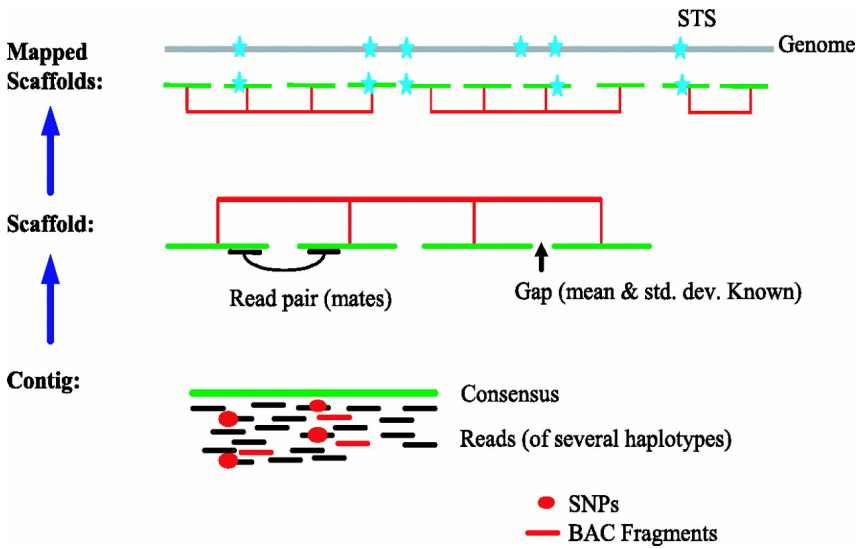
- Around 3Gb
  - Haploid (one version of each chromosome)
- Aims to be point of reference for research
  - Publically available
  - Consistent coordinates
    - Lots of annotation
  - Well documented major and minor releases



Morey, Marcos, et al. "A glimpse into past, present, and future DNA sequencing." Molecular genetics and metabolism 110.1 (2013): 3-24.

Genome Assembly





Lander, Eric S., et al. "Initial sequencing and analysis of the human genome." Nature 409.6822 (2001): 860-921.
Venter, J. Craig, et al. "The sequence of the human genome." Science 291.5507 (2001): 1304-1351.

Sequence Alignment

- Find the best approximate match
  - Global

```
A  C  A  A  C  G
|        x  |
A  -  -  G  C  -
```

  - Local

```
A  C  A  A  C  G
      |  x  |
      A  G  C
```

  - Free-end Global

```
A  C  A  A  C  G
      |  x  |
-  -  A  G  C  -
```

Global Alignment

		A	C	A	A	C	G
	0	-2	-4	-6	-8	-10	-12
A	-2	1	-1	-3	-5	-7	-9
G	-4	-3	-2	-4	-6	-8	-6
C	-6	-5	-2	-4	-6	-5	-7

match 1

mismatch -3

gap -2

$$D(i,j) = \max \begin{cases} D(i-1, j-1) + (\text{match} \,||\, \text{mismatch}) \\ D(i-1, j) + \text{gap} \\ D(i, j-1) + \text{gap} \end{cases}$$

Backtrack to get alignment:

A	C	A	A	C	G
|			x	|	
A	-	-	G	C	-

Local Alignment

		A	C	A	A	C	G
	0	0	0	0	0	0	0
A	0	3	0	3	3	1	0
G	0	0	1	0	1	1	4
C	0	0	3	0	0	4	0

match 3

mismatch -2

gap -3

$$M(i,j) = \max \begin{cases} 0 \\ M(i-1, j-1) + (\text{match} \,||\, \text{mismatch}) \\ M(i-1, j) + \text{gap} \\ M(i, j-1) + \text{gap} \end{cases}$$

Backtrack to get alignment:

A	C	A	A	C	G
		|	x	|	
		A	G	C	

Local Alignment

- Longest common subsequence
- Will find the best aligning substring in both
  - i.e. may not align the whole read

```
AGATGTGCTGCCGCC
   |||x|||
TTTGTACTGAAA
```

Free-end Global Alignment

		A	C	A	A	C	G
	0	0	0	0	0	0	0
A	0	1	-1	1	1	-3	-3
G	0	-2	-2	-1	-1	-2	-2
C	0	-2	-1	-3	-2	0	-2

match 1

mismatch -3

gap -2

$$D(i,j) = \max \begin{cases} D(i-1, j-1) + (\text{match} \,||\, \text{mismatch}) \\ D(i-1, j) + \text{gap} \\ D(i, j-1) + \text{gap} \end{cases}$$

Backtrack to get alignment:

```
A  C  A  A  C  G
      |  x  |
-  -  A  G  C  -
```

Free-End Alignment

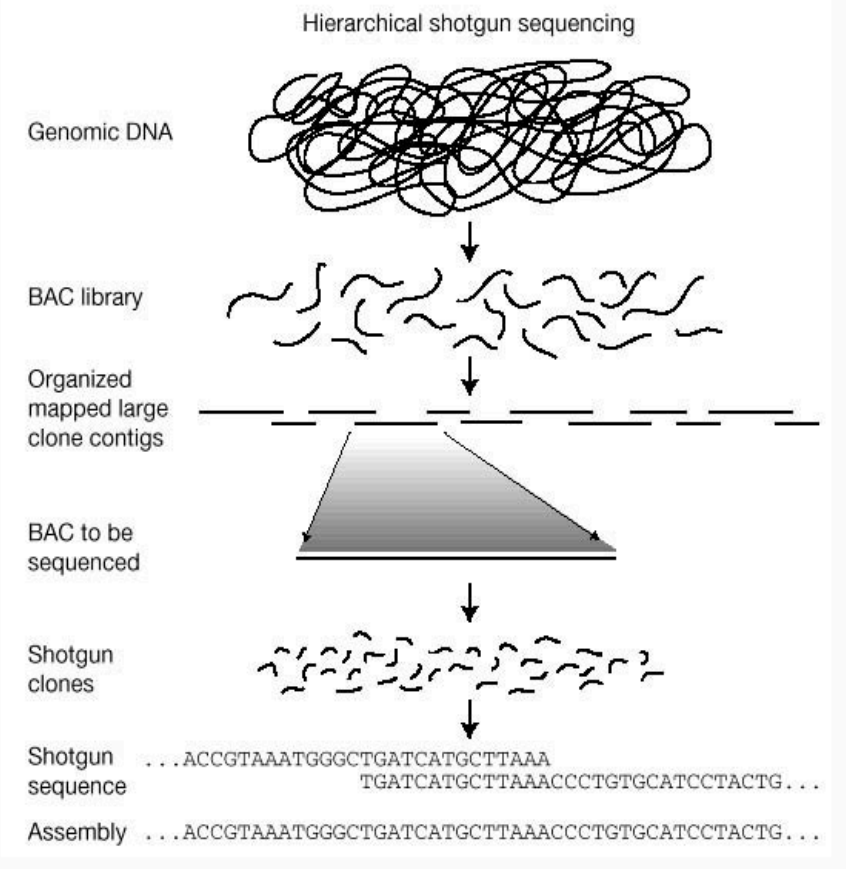
- Aligns whole of both reads
  - Containment
  - Longest prefix/suffix overlap

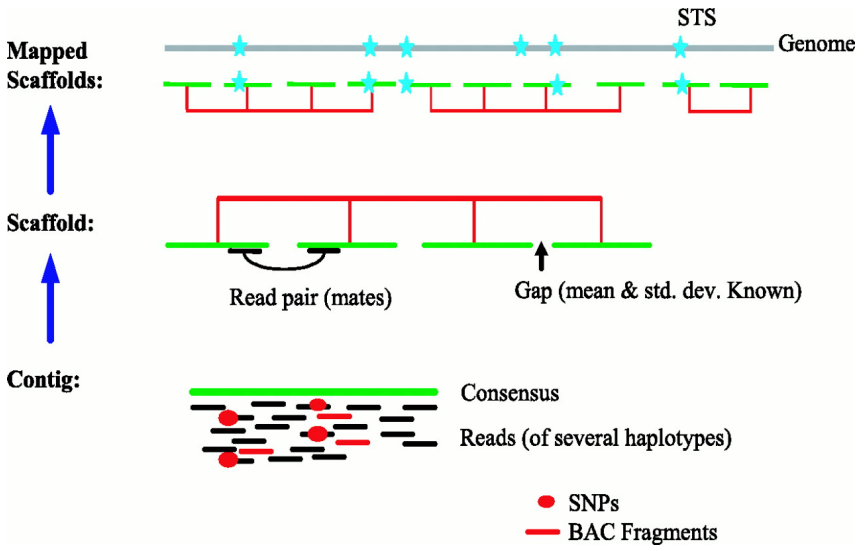
```
TTCAGATGTGCTG
------|||x|||------
      TGTACTGACGTAG
```

Dynamic Programming

- O(nm) for time* and space complexity **

Genome Assembly





Lander, Eric S., et al. "Initial sequencing and analysis of the human genome." Nature 409.6822 (2001): 860-921.
Venter, J. Craig, et al. "The sequence of the human genome." Science 291.5507 (2001): 1304-1351.

The Human Reference Genome

- Based on limited subjects
  - Does not capture variation
    - is one “Golden path”

- Subsequences reported are not unique
  - It has all repeats found in these subjects that could be resolved

Next Generation Sequencing

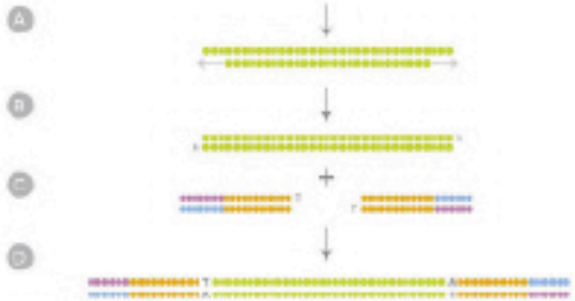
- Traditional sequencing doesn't scale up

- Next Generation Sequencers
  - high throughput (4h-3days)
  - high coverage (20x-50x)
  - short reads (25-200bp)

Automated Workflow

1 LIBRARY PREPARATION

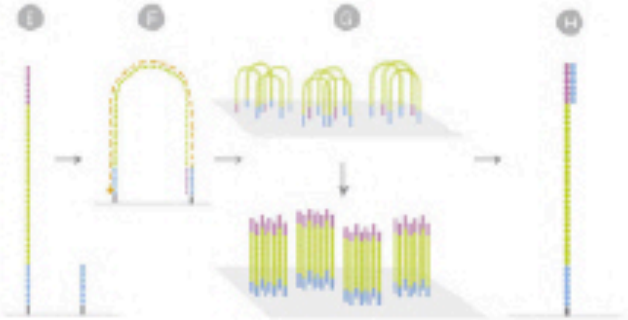
6 hours
3 hours hands-on time



- A Fragment DNA
- B Repair ends
  Add A overhang
- C Ligate adapters
- D Select ligated DNA

2 CLUSTER GENERATION

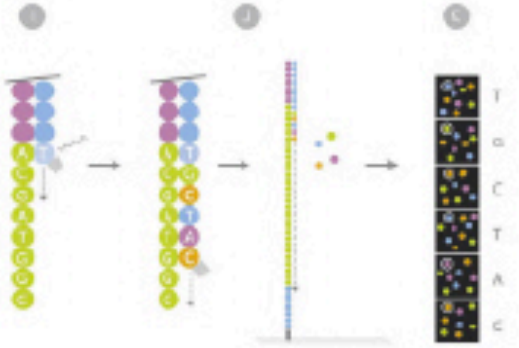
4 hours
< 10 minutes hands-on time
1–96 samples



- E Attach DNA to flow cell
- F Perform bridge amplification
- G Generate clusters
- H Anneal sequencing primer

3 SEQUENCING

1–3 days single-read run
3–9 days paired-end run
30 minutes hands-on time
8 lanes, up to 96 samples per flow cell (run)



- I Extend first base, read, and deblock
- J Repeat step above to extend strand
- K Generate base calls

NGS Uses Resequencing

- NGS produces huge amounts of data
  - 120Gb - 1Tb compressed
- Dynamic Programming is impractical
- Rather than assemble:
  - map to the reference quickly
    - ➔ Read mapping algorithms
  - verify local alignment (and call variants)
    - ➔Dynamic programming
    - ➔Local de-novo assembly

Exact Matching

- Instead of looking for “good” matches, only look for *all* exact matches

Still not enough for genome scale

- Exact string matching algorithms time complexity
  - Worst: $O(nm)$
  - Best: $\Omega(n/m)$

Fast Approximate Matching

- Expect very few differences between the sample's reads and the reference genome
  - Sequencing errors
  - Natural variation

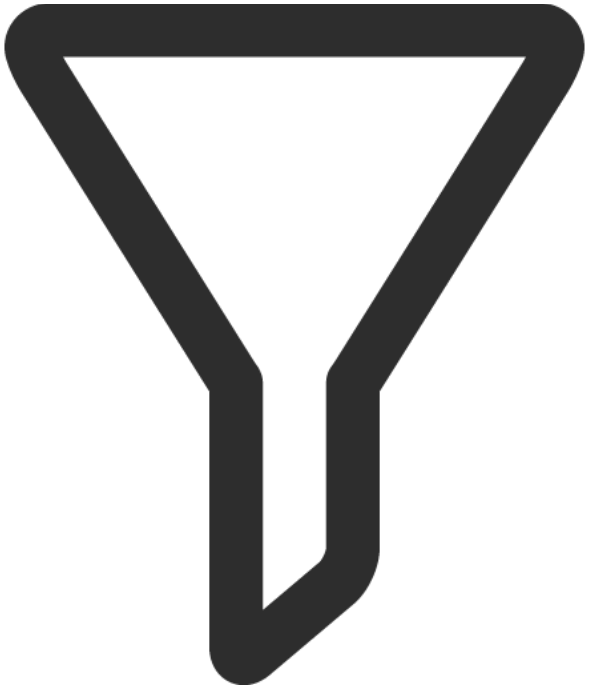
- Expect even fewer differences between sample's reads
  - Sequencing error
  - May be variation within a sample (tissue)
  - May be variation between repeated regions

Read Mapping Algorithms

Two main approaches

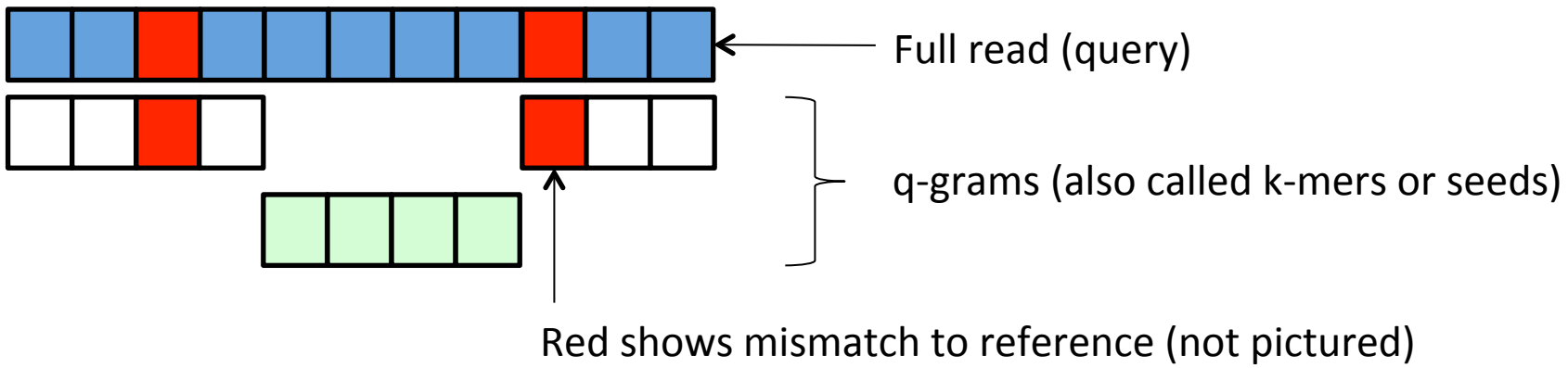
- Filter
- Index

Filtering



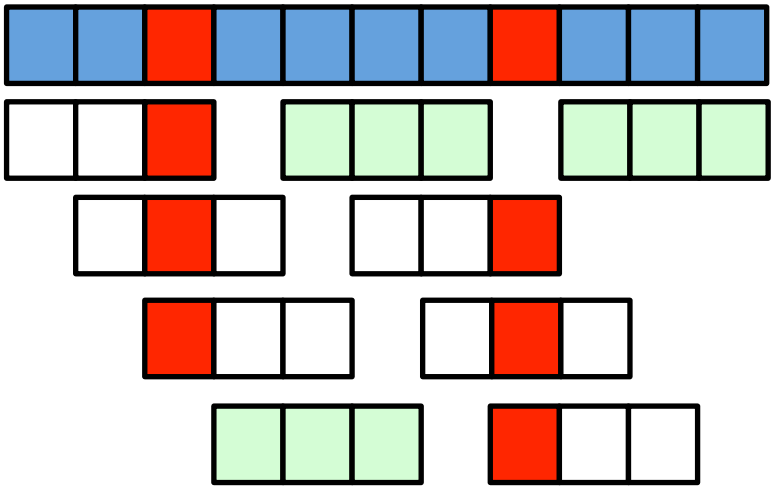
- Reduce the number of possible approximate matches
- In practice, want really good alignments
  - ➔Expect sections of the alignment to be exact
  - ➔Expect no more than *k* errors (mismatches)

Pigeonhole Lemma



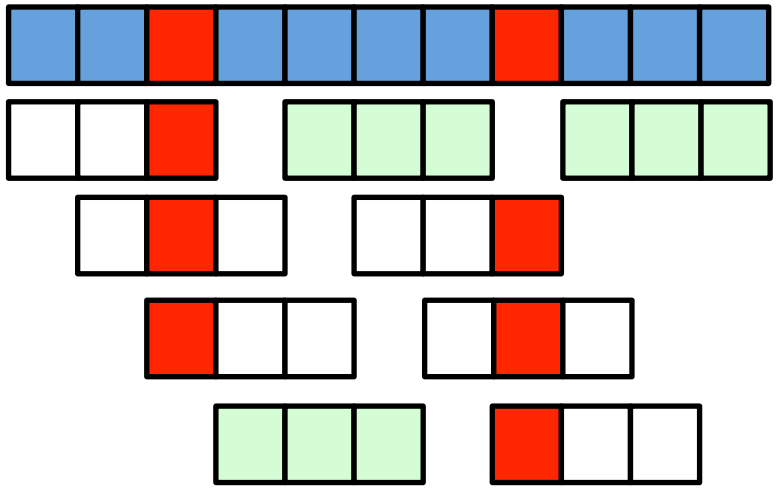
- Assume no more than k errors tolerated
- Divide query into k+1 pieces
  - Search for each of these in the genome
- If the query is in the genome, one will match exactly
  - Report that as a candidate region

q-gram Lemma



- Assume no more than k errors tolerated
- Create all possible overlapping q-grams from the read
  - search for all of these

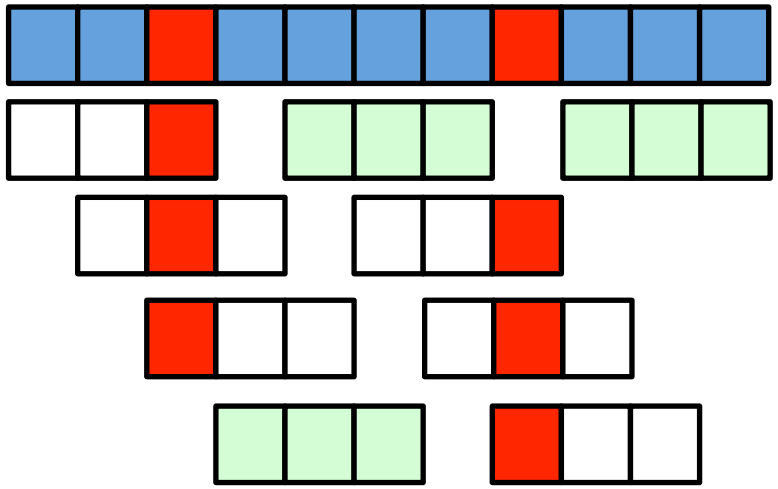
q-gram Lemma



- Number of q-grams for read of length n?
- k errors affect how many q-grams at most in worst case?

- Assume no more than k errors tolerated
- Create all possible overlapping q-grams from the read
  - search for all of these

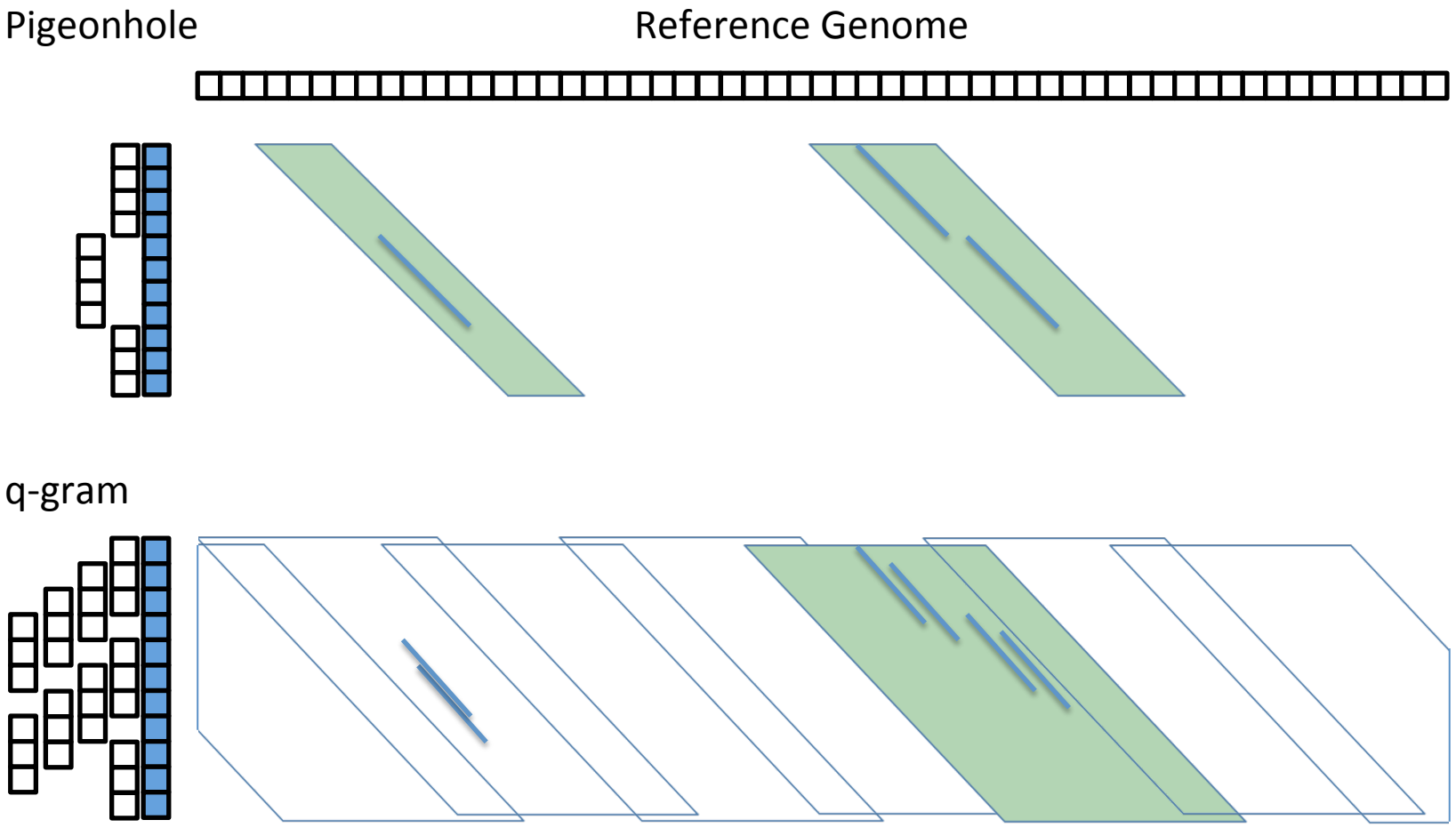
q-gram Lemma



- Number of q-grams for read of length n?
- k errors affect how many q-grams at most in worst case?

- Assume no more than k errors tolerated
- Create all possible overlapping q-grams from the read
  - search for all of these
- If the query is in the genome, at least n-(k+1)q +1 of the q-grams match exactly
  - count the number of q-grams that matched, and if it passes this threshold, report a candidate region

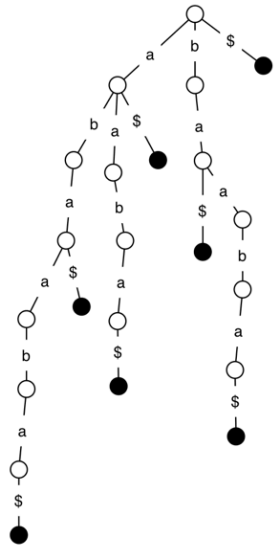
Finding Candidate Regions

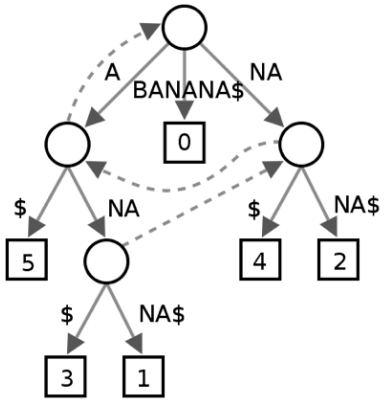


Indexing

- Store the genome/reads in a data structure that facilitates fast exact or near-exact alignment
- Must be reasonable for memory limits of the machine

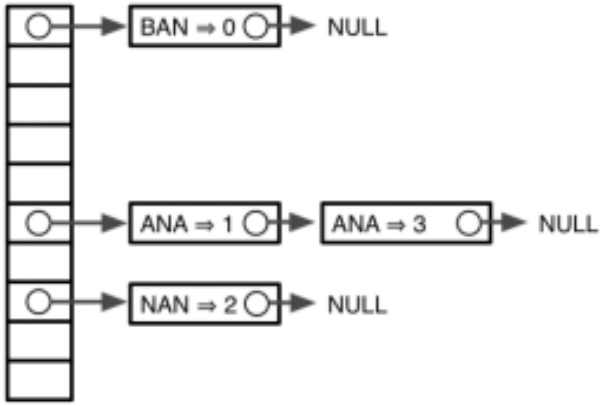
Indexing Data Structures





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

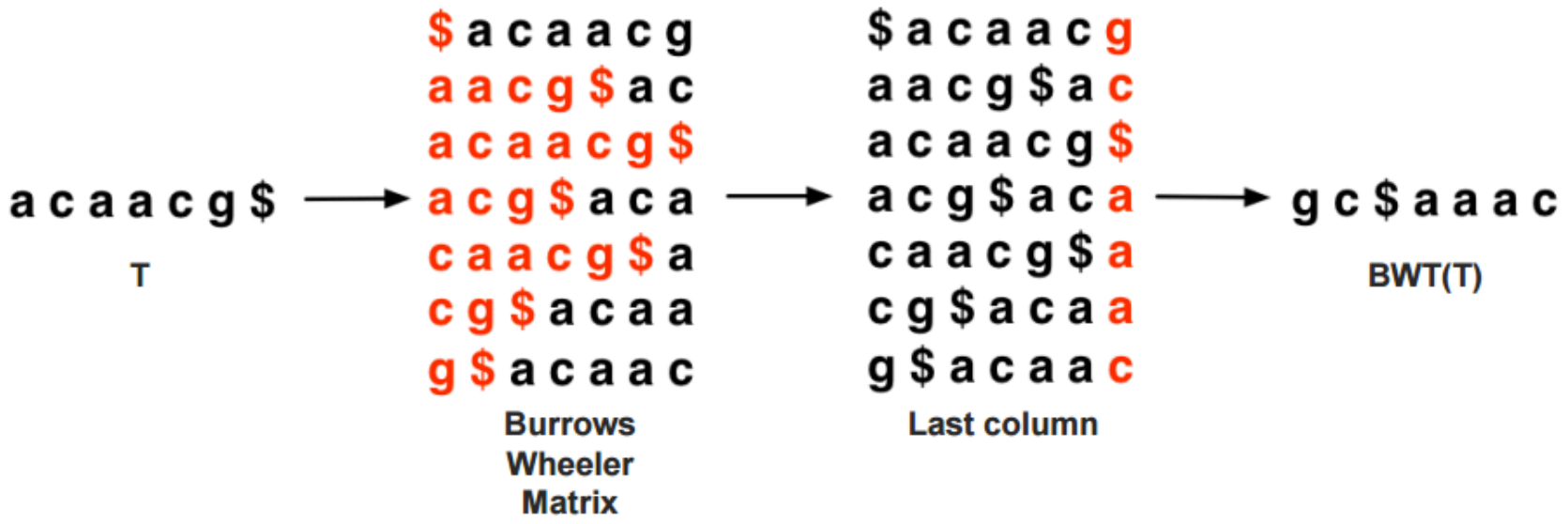
$ B A N A N A
A $ B A N A N
A N A $ B A N
A N A N A $ B
B A N A N A $
N A $ B A N A
N A N A $ B A



Ben Langmead teaching materials: http://www.langmead-lab.org/teaching-materials/

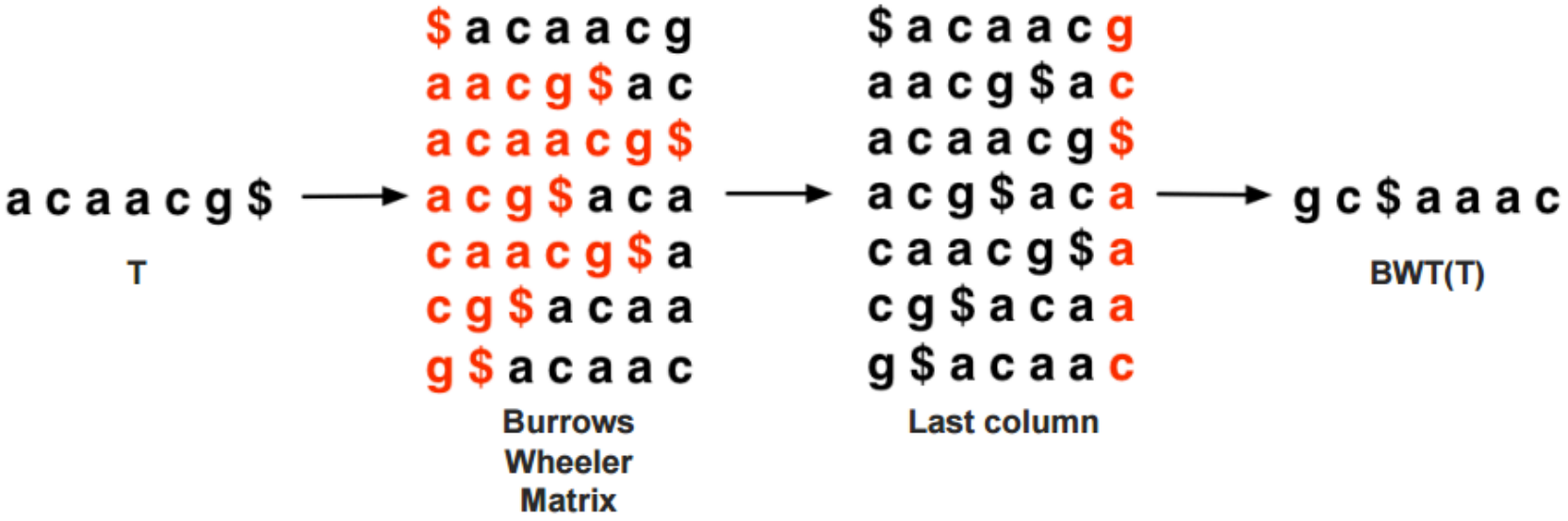
FM Index

- Uses Burrows-Wheeler transform
  - Plus extra tables to speed things up



Burrows-Wheeler transform

- Because they are rotations
  - the character in the last column is what precedes the character in the first column in the original string
- Because the suffixes are sorted
  - the first row, last character is the end of the original string



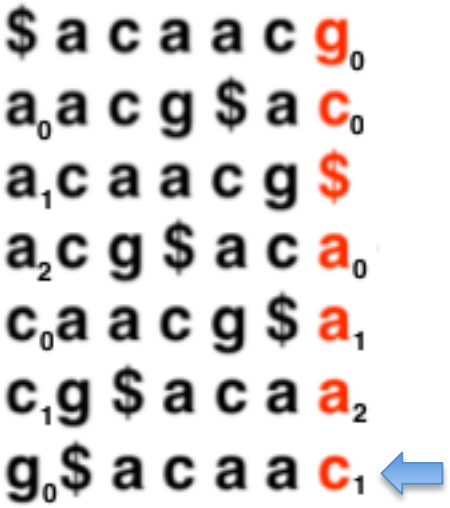
Burrows-Wheeler transform

– Because the suffixes are sorted (cont'd)
  - the rank of the character in the last column is the same as the first

$ a c a a c g_0
a_0a c g $ a c_0
a_1c a a c g $
a_2c g $ a c a_0
c_0a a c g $ a_1
c_1g $ a c a a_2
$g_0$$ a c a a c_1

Burrows-Wheeler transform

- We can find the position in the first column (F) based only on information from the last column (L)



LF('c', 6) = Occ('c') + Count('c', 6)

= 4 + 1

Occurrence:

Number of letters before any 'c' in F?

4 ($ and 3 a's)

Count:

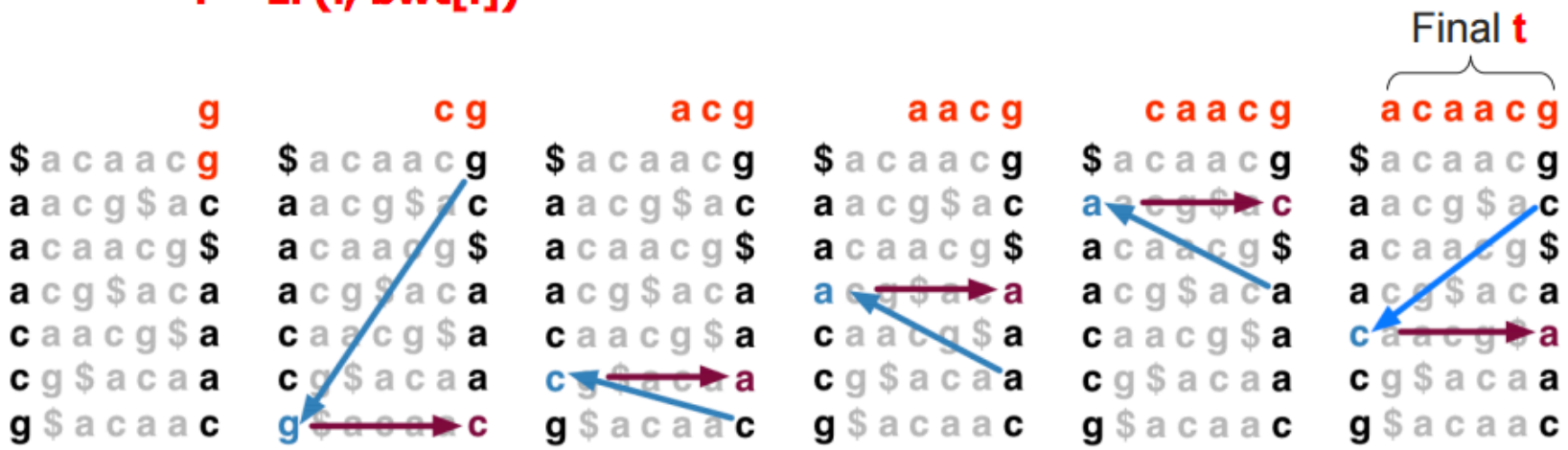
How many 'c's have we seen in L?

1 (c_0)

Burrows-Wheeler transform

- Walk left algorithm
  - We can use the BWT and the LF function to reconstruct the original text

```
i = 0
t = ""
while bwt[i] != '$':
    t = bwt[i] + t
    i = LF(i, bwt[i])
```



FM Index for Exact Matching

Ferragina, Paolo and Manzini, Giovanni. "Opportunistic data structures with applications." Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on. IEEE (2000)

FM Index for Exact Matching


```
q = “aac”
top = 0
bot = len(bwt)
for qc in reverse(q):
      top = LF(top, qc)
      bot = LF(bot,
qc)
```

Ferragina, Paolo and Manzini, Giovanni. "Opportunistic data structures with applications." Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on. IEEE (2000)

FM Index for Exact Matching

Track a top and bottom index

- These bound the remaining possible matches at each step
- If they are ever the same, there are no matches

Here, LF is called on the *query* characters

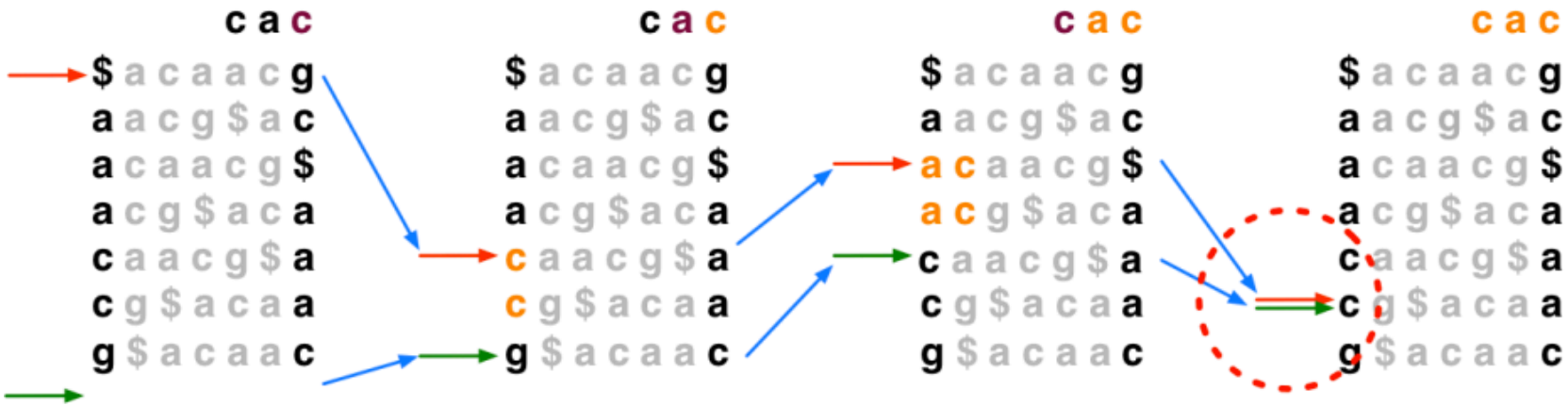
- "Where are the first & last qc you saw in F?"

The end TOP index is our match!

```
q = "aac"
top = 0
bot = len(bwt)
for qc in reverse(q):
    top = LF(top, qc)
    bot = LF(bot, qc)
```

Ferragina, Paolo and Manzini, Giovanni. "Opportunistic data structures with applications." Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on. IEEE (2000)

FM Index for Exact Matching



Track a top and bottom index

- These bound the remaining possible matches at each step
- If they are ever the same, there are no matches

Here, LF is called on the *query* characters

- "Where are the first & last qc you saw in F?"

The end TOP index is our match!

```
q = "aac"
top = 0
bot = len(bwt)
for qc in reverse(q):
        top = LF(top, qc)
        bot = LF(bot, qc)
```

FM Index

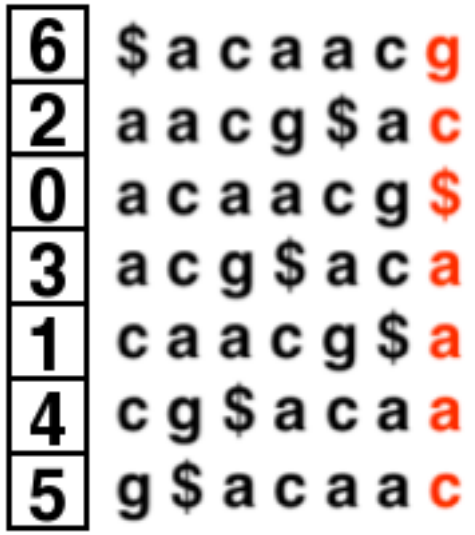
- How do we find this in the genome?

- Isn’t the count operation O(n)?

FM Index

- How do we find this in the genome?

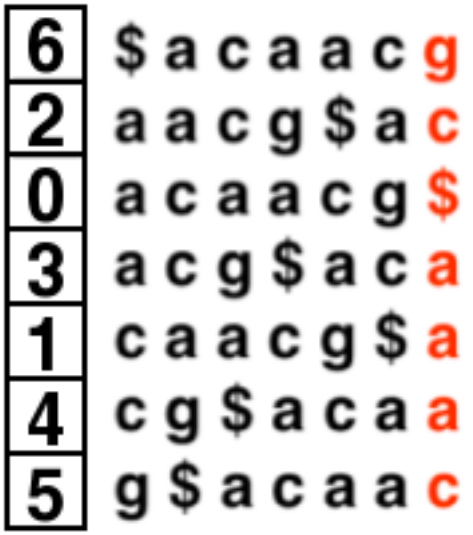
1) Can use our walk left algorithm to reconstruct



FM Index

- How do we find this in the genome?

1) Can use our walk left algorithm to reconstruct
2) Could store the entire suffix array



FM Index

- How do we find this in the genome?

6	$ a c a a c g
2	a a c g $ a c
0	a c a a c g $
3	a c g $ a c a
1	c a a c g $ a
4	c g $ a c a a
5	g $ a c a a c

1) Can use our walk left algorithm to reconstruct
2) Could store the entire suffix array
3) Only store certain rows of (2), use (1) until we get to one.

FM Index

- Isn’t the count operation O(n)?

FM Index

- Isn't the count operation O(n)?
  - We again store cumulative counts for certain rows, for each of $ACTG
- Also, the reference is usually put in backward (or both forward and backward)

Beyond?

- Read-mapping is limited by
  - the reference genome assembly
  - exact matching

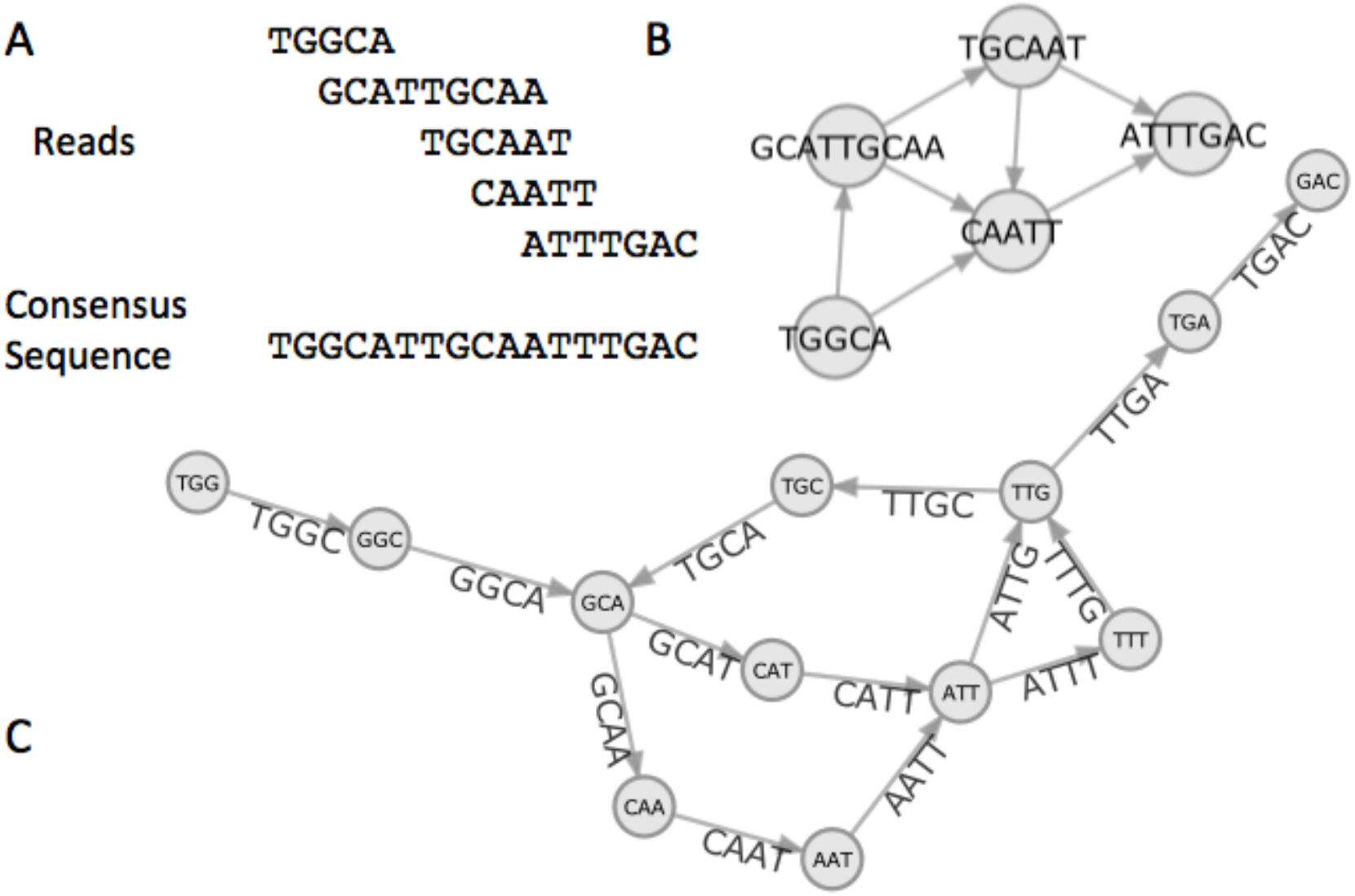
- So, why not assemble each time?

De Novo Assembly

- Overlap-Layout-Consensus
- De Bruijn Graphs

Overlap-Layout-Consensus

- Overlap
  - Compute overlap score of all reads
- Layout
  - Create a graph where nodes are a read, edges are overlaps between reads
  - Find their “layout” by finding a Hamiltonian path through the graph
- Consensus
  - Find the consensus sequence by reading nodes along the path


A
TGGCA
GCATTGCAA
Reads
TGCAAT
CAATT
ATTTGAC
Consensus
Sequence
TGGCATTGCAATTTGAC
B
TGCAAT
GCATTGCAA
ATTTGAC
CAATT
TGGCA
GAC
TGAC
TGA
TTGA
TGG
TGGC
GGC
GGCA
GCA
TGC
TGCA
TTGC
TTG
ATTG
TTTG
TTT
GCAT
CAT
CATT
ATT
ATTT
GCAA
CAA
CAAT
AAT
AATT
C

De Bruijn Graphs

- Break reads into k-mers
- Each node in the graph is a k-mer
- Connect an edge to the next k-mer found in the read
- Find a Eulerian path

De Novo Assembly

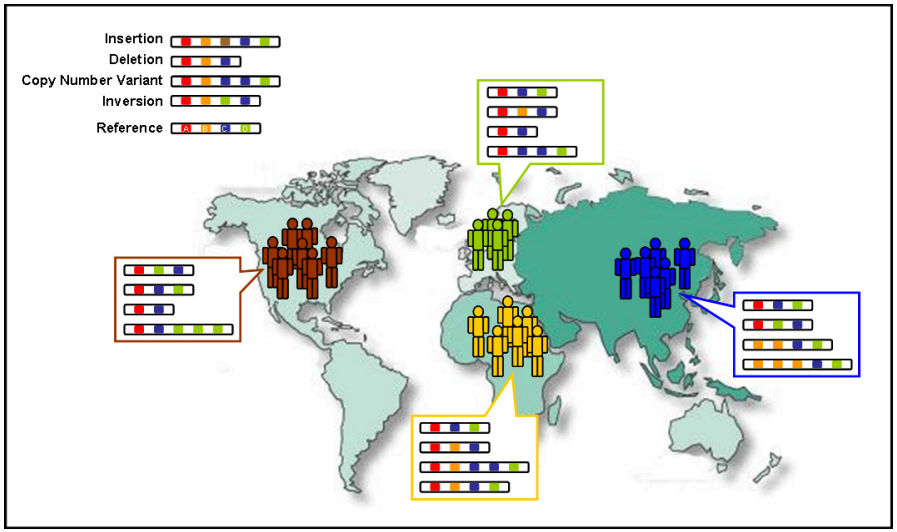
- Confounded by repeats
  - repeats are collapsed in these representations
  - may have many ways in and out of these graph regions.

Further beyond

- Newer sequencing techniques
  - Nanopore sequencing
  - Single cell sequencing
- Downstream analysis issues
  - How do we compare genomes?
  - How do we store them?
- Improve the reference model

Further beyond

- Downstream analysis issues
  - How do we compare genomes?
  - How do we store them?
- Improve the reference model



https://en.wikipedia.org/wiki/1000_Genomes_Project#/media/File:Genetic_Variation.jpg

Thank you!

References and resources

- Algorithms
  - Reinert, Knut, et al. "Alignment of Next-Generation Sequencing Reads." Annual review of genomics and human genetics 0 (2015).
  - Li, Heng, and Nils Homer. "A survey of sequence alignment algorithms for next-generation sequencing." Briefings in bioinformatics 11.5 (2010): 473-483.
- Sequencing
  - Morey, Marcos, et al. "A glimpse into past, present, and future DNA sequencing." Molecular genetics and metabolism 110.1 (2013): 3-24.
- Ben Langmead's Teaching Resources:
  - http://www.langmead-lab.org/teaching-materials/

Supervisor: Mark Daley

DaleyLab.org

Structural Variant Discovery