The Elizabeth H.
and James S. McDonnell III
McDonnell Genome Institute
at Washington University

Alignment concepts

Felicia Gomez, PhD
Research Assistant Professor
Washington University in Saint Louis
Griffith and Fehniger labs

11/10/2019 – Advanced Sequencing Technologies and Applications
Cold Spring Harbor

Alignment is central to most genomic research

FASTQ format

Align to genome

Quality control

Find peaks

Quantify transcripts

Find SVs

.
.
.

etc.

Analysis, Interpretation, Discovery

Phase 0

Phase1

Phase 2-100

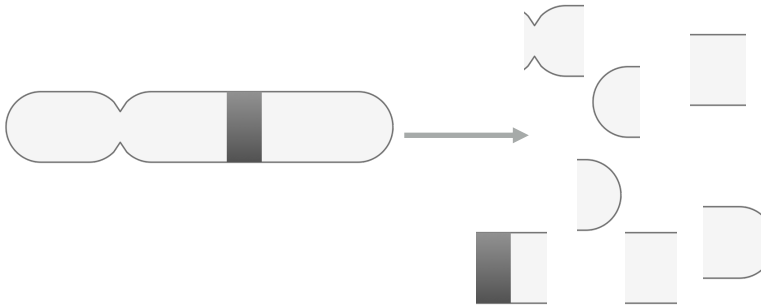
Slide courtesy of Andrew Farrell, Obi and Malachi Griffith

Generate sequence reads

Fragment a genome —> DNA library

PCR amplification

Sequence reads (ends of DNA fragment for mate pairs)



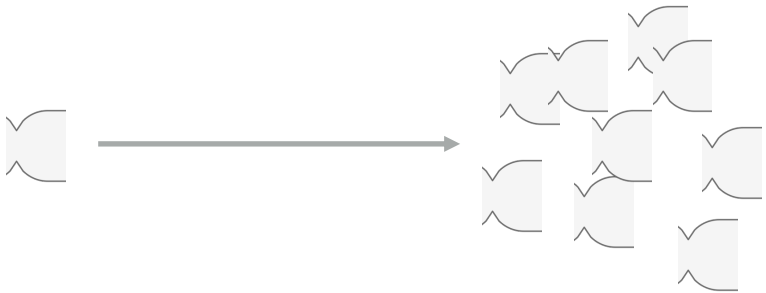
Slide courtesy of Andrew Farrell

Generate sequence reads

Fragment a genome —> DNA library

PCR amplification

Sequence reads (ends of DNA fragment for mate pairs)



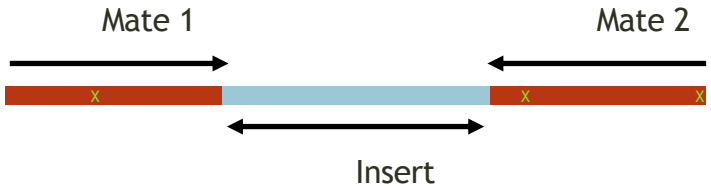
Slide courtesy of Andrew Farrell

Generate sequence reads

Fragment a genome —> DNA library

PCR amplification

Sequence reads (ends of DNA fragment for mate pairs)



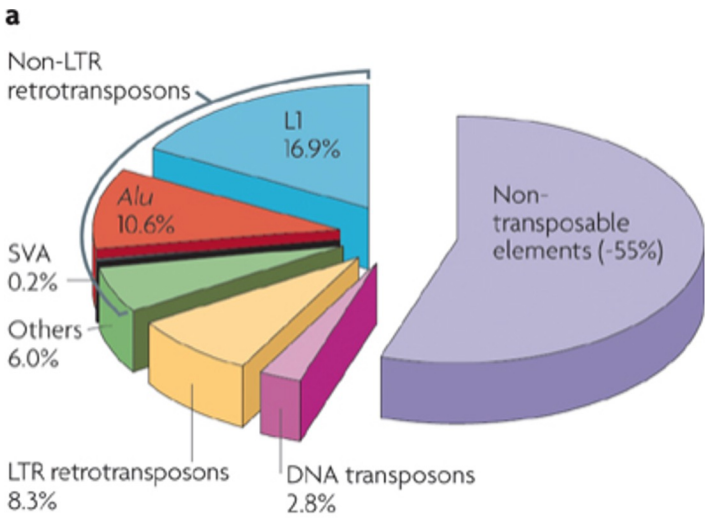
Slide courtesy of Andrew Farrell

We have FASTQ files. Now what?

- Need to find a home for every read in the file.
- Must get the alignment just right. Or else……problems.
- Must choose the right tool for the experiment.

Slide courtesy of Andrew Farrell

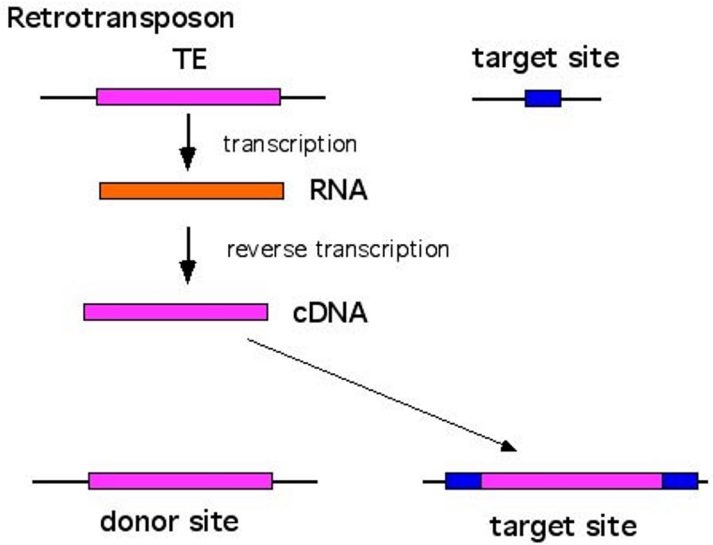
Problem: Half of the human genome is comprised of repeats



McClintock's "jumping genes" in maize

http://www.nature.com/nrg/journal/v10/n10/pdf/nrg2640.pdf

Slide courtesy of Aaron Quinlan



Retrotransposons use a "copy/paste" mechanism

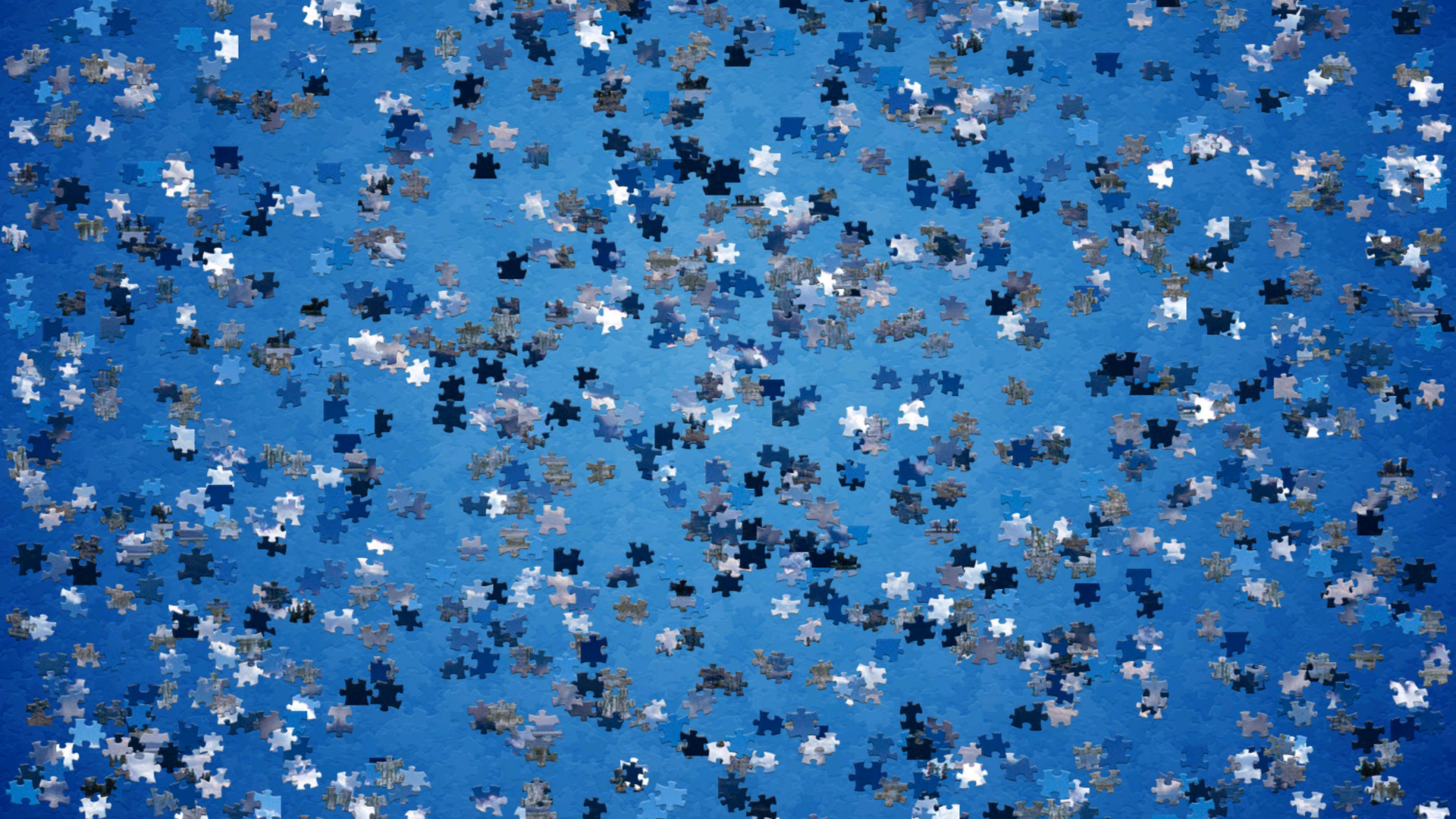
DNA transposons use a "cut/paste" mechanism

Problem: Half of the human genome is comprised of repeats

(first bit of human chromosome 1)

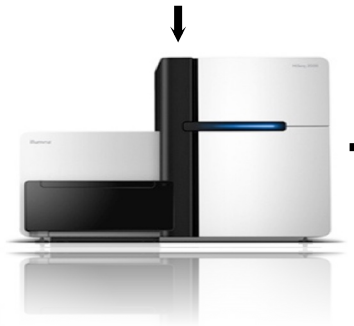
taaccctaaccctaaccctaaccctaaccctaaccctaacccta
acccta acccta acccta acccta acccta acccta acccta
cctaacccaacccta acccta acccta acccta acccta acccc
taacccta acccta acccta acccta acctaacctaacccta
ccccccta acccta acccta acccta acccccta acccta
ccctaaaccctaacccta acccta acccta acccccaaccccaac
cccaaccccaaccccaaccccaaccctaaccccta acccta
ctaccctaaccctaaccctaaccctaaccctaaccctaaccccta
taaccctaaccctaaccctaaccctaaccctaaccctaaccccta
aaccctaaccctaaccctcgcggtaccctcagccggcccgcccgcccggg
tctgacctgaggagaactgtgctccgccttcagagtaccaccgaaatctg
tgcagaggacaacgcagctccgccctcgcggtgctctccgggtctgtgct
gaggagaacgcaactccgccggcgcaggcgcagagaggcgcgccgcgccg
gcgcaggcgcagacacatgctagcgcgtcggggtggaggcgtggcgcagg
cgcagagaggcgcgccgcgccggcgcaggcgcagagacacatgctaccgc
gtccaggggtggaggcgtggcgcaggcgcagagaggcgcaccgcgccggc
gcaggcgcagagacacatgctagcgcgtccaggggtggaggcgtggcgca
ggcgcagagacgcaagcctacgggcgggggttgggggggcgtgtgttgca
ggagcaaagtcgcacggcgccgggctggggcgggggggagggtggcgccgt
gcacgcgcagaaactcacgtcacggtggcgcggcgcagagacgggtagaa

Slide courtesy of Aaron Quinlan



Best case scenario: an error-free sequencing technology

ATTCGAAACA
TTCGCGCAAT
CTGGACTCAA



ATTCGAAACA
TTCGCGCAAT
CTGGACTCAA

Aligner

TACCTCCAGGGGGCATCCTCCCCCCCAATTC
GAAACACAATCGTAGCCCCTGGCACTACCTA
TGTGTGTCAATTCGGAGAGAGAGAGATTCAC
GAAAAAAAAGTCTGGACTCAACTAGGATACA
CACATTCGGCTACAGATACCAAAAAAAAAAA
AAAAAAAATTTTCACCATTGAGGCACCACCT
TCTCGTCGCTGCGTCGCTCTGCTCGCTTCGG
CTAAAAATTCGCGCAATACATTCGGCTACAG
ATACCAAAAAAA

Computers are rather good at finding ***exact*** matches.
Think Google.

Slide courtesy of Aaron Quinlan

Reality check. Errors happen. Frequently.

ATTCGAAACA

→ ATTTGAAACA → Aligner →

TACCTCCAGGGGGCATCCTCCCCCCCAATTC
GAAACACAATCGTAGCCCCTGGCACTACCTA
TGTGTGTCAATTCGGAGAGAGAGAGATTGGA
AACAAAAAAGTGCTACAGATACCACTAGGAT
ACACACATTCGGCTACAGATACCAAAAAAAA
AAAAAAAAAAATTTTCACCATTGAGGCACCA
CCTTCTCGTCGCTGCGTCGCTCTGCTCGCGG
CTAAAAAATTAGAAACAACATTCGGCTACAG
ATACCAAAATTT

"Fuzzy" matching is much more computationally expensive.

Think Google's "Did you mean..."

Slide courtesy of Aaron Quinlan

There *are* optimal solutions.

Reference	`cgggtatccaa`
Read	`ccctaggtccca`

What is the best alignment?

Slide courtesy of Andrew Farrell

Reference	cgggtatccaa
Read	ccctaggtccca
Reference	cgggta--t-ccaa
Read	ccc-taggtccc-a

Slide courtesy of Andrew Farrell

Reference cgggtatccaa

Read ccctaggtccca

Reference cgggta--t-ccaa

Read ccc-taggtccc-a

Reference cgggta---tccaa

Read cc--ctaggtccca

Slide courtesy of Andrew Farrell

Slide courtesy of Andrew Farrell

Global: Needleman-Wunsch algorithm

Local: Smith-Waterman algorithm

A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins

SAUL B. NEEDLEMAN AND CHRISTIAN D. WUNSCH

Department of Biochemistry, Northwestern University, and Nuclear Medicine Service, V. A. Research Hospital Chicago, Ill. 60611, U.S.A.

(*Received 21 July 1969*) *J. Mol. Biol.* (1981), **147**, 195–197

Identification of Common Molecular Subsequences

The identification of maximally homologous subsequences among sets of long sequences is an important problem in molecular sequence analysis. The problem is straightforward only if one restricts consideration to contiguous subsequences (segments) containing no internal deletions or insertions. The more general problem has its solution in an extension of sequence metrics (Sellers 1974; Waterman *et al.*, 1976) developed to measure the minimum number of "events" required to convert one sequence into another.

Slide courtesy of Aaron Quinlan

Local: Smith-Waterman algorithm

```
5' ACTACTAGATTACTTACGGATCAGGTACTTTAGAGGCTTGCAACCA 3'
     |||| |||||| ||||||||||||||||
        5' TACTCACGGATGAGGTACTTTAGAGGC 3'
```

Global: Needleman-Wunsch algorithm

```
5' ACTACTAGATTACTTACGGATCAGGTACTTTAGAGGCTTGCAACCA 3'
   ||||||||||    |||||||  |||||||||||||| |||||||
5' ACTACTAGATT----ACGGATC--GTACTTTAGAGGCTAGCAACCA 3'
```

Slide courtesy of Aaron Quinlan

Local Alignment: dynamic programming

Scoring scheme:

Match: +3

Mismatch -3

Gap: -2

Adding an extra row and column allows us to align any base to any other positions in the other sequence

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

Slide courtesy of Aaron Quinlan

Local Alignment: dynamic programming

Scoring scheme:

Match: +3

Mismatch -3

Gap: -2

Initialize the matrix with a minimum score of 0.

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

Slide courtesy of Aaron Quinlan

Local Alignment: dynamic programming

Scoring scheme:

Match: +3

Mismatch -3

Gap: -2

Mismatch, but score can't go below 0.

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

Slide courtesy of Aaron Quinlan

Local Alignment: dynamic programming

Scoring scheme:

Match: +3

Mismatch -3

Gap: -2

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3						
G	0								
T	0								
T	0								
G	0								
A	0								
C	0								
T	0								
A	0								

Slide courtesy of Aaron Quinlan

Local Alignment: dynamic programming

Scoring scheme:

Match: +3

Mismatch -3

Gap: -2

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0				
G	0	0	3	1	0				
T	0	3	1	6	4				
T	0	3	1	4	9	→ 7			
G	0				↓ 7	↘ 6			
A	0								
C	0								
T	0								
A	0								

Slide courtesy of Aaron Quinlan

Local Alignment: dynamic programming

Scoring scheme:

Match: +3

Mismatch -3

Gap: -2

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

Slide courtesy of Aaron Quinlan

The traceback

Start at max score, traceback to next highest score, and so on. Stop at zero

C

C

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

Slide courtesy of Aaron Quinlan

The traceback

Start at max score, traceback to next highest score, and so on. Stop at zero

A C
A C

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

Slide courtesy of Aaron Quinlan

The traceback

Start at max score, traceback to next highest score, and so on. Stop at zero

Gap b/c that is the edit that led to this cell (9->7)

\- A C

G A C

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

Slide courtesy of Aaron Quinlan

The traceback

Start at max score, traceback to next highest score, and so on. Stop at zero

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

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

Slide courtesy of Aaron Quinlan

This a "local" alignment.
Subset of the full sequence.

Start at max score, traceback to next highest score, and so on. Stop at zero

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

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

Slide courtesy of Aaron Quinlan

Aligning to a Reference Genome

- There are two major approaches:
  - Hashing the reference
  - Burrows Wheeler Transformation

Slide courtesy of Andrew Farrell

Hash-based mapping:

Step1: hash/index the genome

Toy genome (16 bp)

CATGGTCATTGGTTCC

Slide courtesy of Aaron Quinlan

Hash-based mapping:

Step1: hash/index the genome

CATGGTCATTGGTTCC

k = 3

Kmer/Hash	Genome Positions
CAT	1

Slide courtesy of Aaron Quinlan

Hash-based mapping:

Step1: hash/index the genome

CATGGTCATTGGTTCC

k = 3

Kmer/Hash	Genome Positions
CAT	1
ATG	2

Slide courtesy of Aaron Quinlan

Hash-based mapping:

Step1: hash/index the genome

CATGGTCATTGGTTCC

k = 3

Kmer/Hash	Genome Positions
CAT	1
ATG	2
TGG	3

Slide courtesy of Aaron Quinlan

Hash-based mapping:

Step1: hash/index the genome

CATGGTCATTGGTTCC

k = 3

Kmer/Hash	Genome Positions
CAT	1
ATG	2
TGG	3
GGT	4

Slide courtesy of Aaron Quinlan

Hash-based mapping:

Step1: hash/index the genome

CATGGTCATTGGTTCC

k = 3

Kmer/Hash	Genome Positions
CAT	1
ATG	2
TGG	3
GGT	4
GTC	5

Slide courtesy of Aaron Quinlan

Hash-based mapping:

Step1: hash/index the genome

CATGGTCATTGGTTCC

k = 3

Kmer/Hash	Genome Positions
CAT	1
ATG	2
TGG	3
GGT	4
GTC	5
TCA	6

Slide courtesy of Aaron Quinlan

Hash-based mapping:

Step1: hash/index the genome

CATGGTCATTGGTTCC

k = 3

Kmer/Hash	Genome Positions
CAT	1,7
ATG	2
TGG	3
GGT	4
GTC	5
TCA	6

Slide courtesy of Aaron Quinlan

Hash-based mapping:

Step1: hash/index the genome

CATGGTCATTGGTTCC

k = 3

Kmer/Hash	Genome Positions
CAT	1,7
ATG	2
TGG	3,10
GGT	4,11
GTC	5
TCA	6
ATT	8
TTG	9
GTT	12
TTC	13
TCC	14

Complete hash/kmer index of our toy genome (forward strand only)

Slide courtesy of Aaron Quinlan

Hash-based mapping:

Step2: use the index to map (i.e., find alignment locations) reads

Toy genome

CATGGTCATTGGTTCC

Read TGGTCA

Kmer/Hash	Genome Positions
CAT	1,7
ATG	2
TGG	3,10
GGT	4,11
GTC	5
TCA	6
ATT	8
TTG	9
GTT	12
TTC	13
TCC	14

kmer index is used to quickly find candidate alignment locations in genome.

Slide courtesy of Aaron Quinlan

Hash-based mapping:

Step2: use the index to map (i.e., find alignment locations) reads

Toy genome

CATGGTCATTGGTTCC

Read TGGTCA

Kmer/Hash	Genome Positions
CAT	1,7
ATG	2
TGG	3,10
GGT	4,11
GTC	5
TCA	6
ATT	8
TTG	9
GTT	12
TTC	13
TCC	14

Slide courtesy of Aaron Quinlan

Hash-based mapping:

Step2: use the index to map (i.e., find alignment locations) reads

Toy genome

CATGGTCATTGGTTCC

Kmer/Hash	Genome Positions
CAT	1,7
ATG	2
TGG	3,10
GGT	4,11
GTC	5
TCA	6
ATT	8
TTG	9
GTT	12
TTC	13
TCC	14

Hash match

Read TGGTCA

Hash matches 3,10

Slide courtesy of Aaron Quinlan

Hash-based mapping:

Step2: use the index to map (i.e., find alignment locations) reads

Toy genome

CATGGTCATTGGTTCC

Read TGGTCA

Hash matches 3,10,6

Kmer/Hash	Genome Positions
CAT	1,7
ATG	2
TGG	3,10
GGT	4,11
GTC	5
TCA	6
ATT	8
TTG	9
GTT	12
TTC	13
TCC	14

Slide courtesy of Aaron Quinlan

Hash-based mapping:

Step2: use the index to map (i.e., find alignment locations) reads

Toy genome

CATGGTCATTGGTTCC

3 6

Read TGGTCA

Hash matches 3,10,6

Kmer/Hash	Genome Positions
CAT	1,7
ATG	2
TGG	3,10
GGT	4,11
GTC	5
TCA	6
ATT	8
TTG	9
GTT	12
TTC	13
TCC	14

Slide courtesy of Aaron Quinlan

Okay, that was a bit easy because the read and the reference exactly matched. What about if there is a sequencing error or a genetic variant in the read?

Slide courtesy of Aaron Quinlan

Hash-based mapping:

Step2: use the index to map (i.e., find alignment locations) reads

Toy genome: CATGGTCATTGGTTCC

Read TGGTCT

Kmer/Hash	Genome Positions
CAT	1,7
ATG	2
TGG	3,10
GGT	4,11
GTC	5
TCA	6
ATT	8
TTG	9
GTT	12
TTC	13
TCC	14

kmer index is used to quickly find candidate alignment locations in genome.

Slide courtesy of Aaron Quinlan

Hash-based mapping:

Step2: use the index to map (i.e., find alignment locations) reads

Toy genome

CATGGTCATTGGTTCC

Kmer/Hash	Genome Positions
CAT	1,7
ATG	2
TGG	3,10
GGT	4,11
GTC	5
TCA	6
ATT	8
TTG	9
GTT	12
TTC	13
TCC	14

Hash match

Read TGGTCT

Hash matches 3,10

Slide courtesy of Aaron Quinlan

Hash-based mapping:

Step2: use the index to map (i.e., find alignment locations) reads

Toy genome

CATGGTCATTGGTTCC

Read TGGTCT

?

Hash matches 3,10

Kmer/Hash	Genome Positions
CAT	1,7
ATG	2
TGG	3,10
GGT	4,11
GTC	5
TCA	6
ATT	8
TTG	9
GTT	12
TTC	13
TCC	14

Slide courtesy of Aaron Quinlan

Thought experiment: what is a good choice of hash size (k for k-mers) for building a hash table to facilitate sequence mapping to the human genome?

Slide courtesy of Aaron Quinlan

k=1?

Slide courtesy of Aaron Quinlan

k=3?

(4^3 possibilities)

AAA, AAC, AAG, … ,

TTT

Slide courtesy of Aaron Quinlan

k=10?

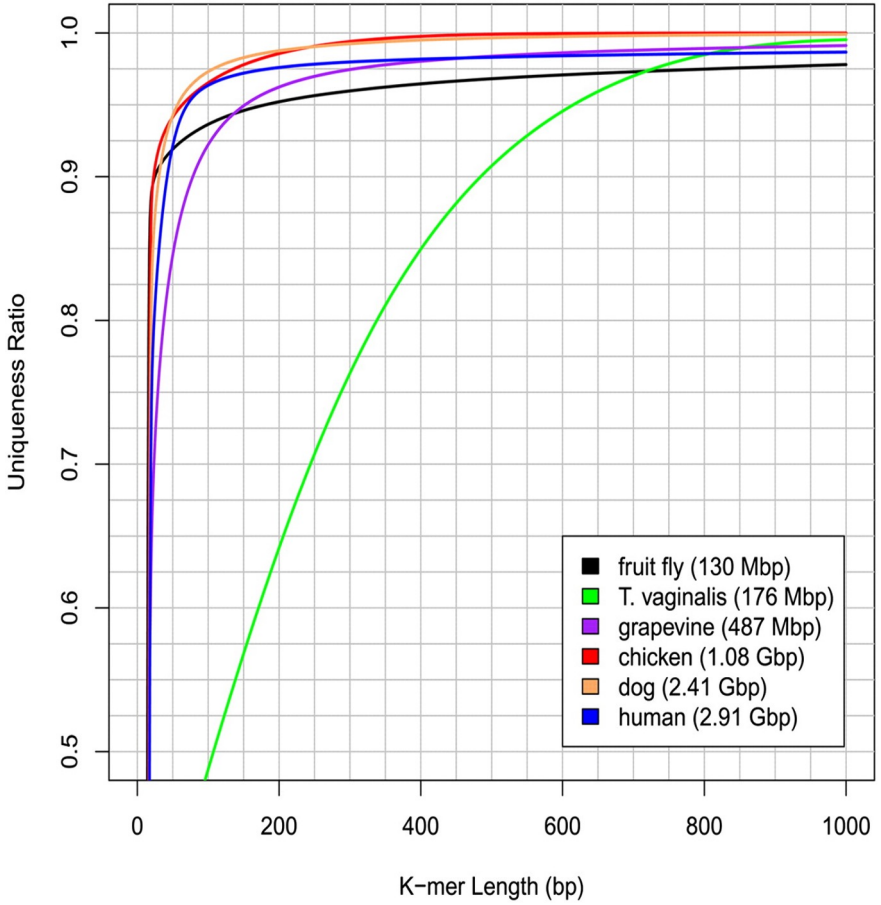
4^{10} (1,048,576)

Every one of these is present in the human genome at least once

http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-9-167

Slide courtesy of Aaron Quinlan

- Percent of genome covered by unique sequences of length K
- It take a pretty long k-mer to be unique in most genomes

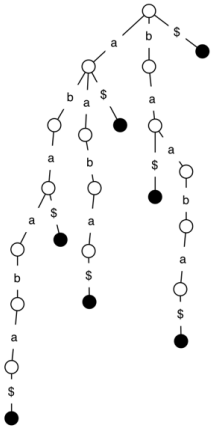


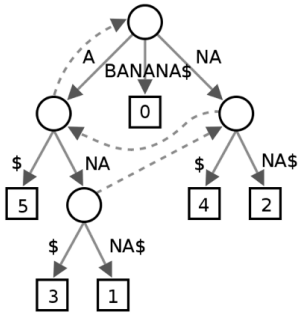
Slide courtesy of Aaron Quinlan

http://genome.cshlp.org/content/20/9/1165

Burrows-Wheeler Transform

The latest fad. More involved computationally, but requires much less memory.





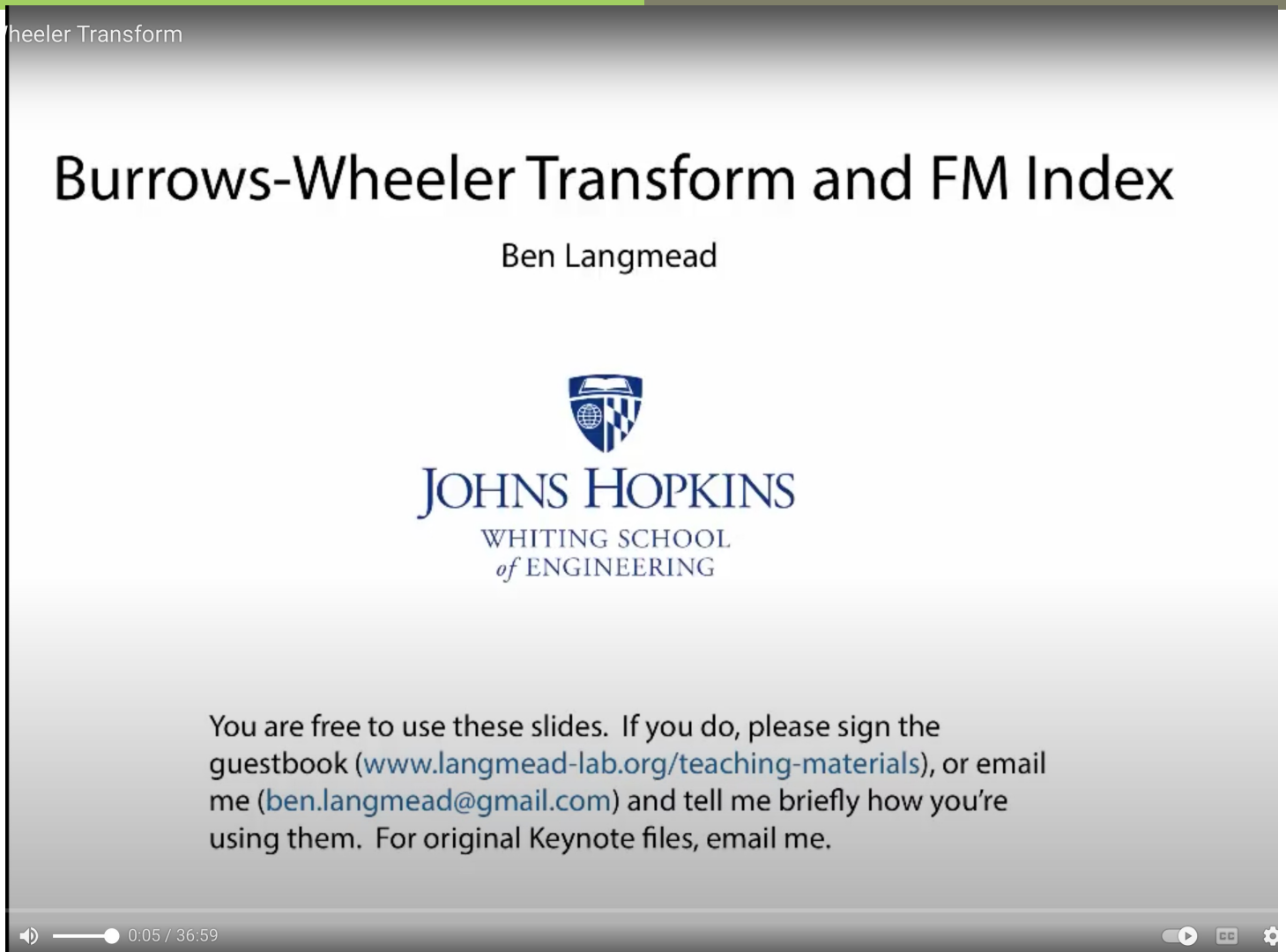
6	$
5	A$
3	ANA$
1	ANANA$
0	BANANA$
4	NA$
2	NANA$

$ 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**

Bowtie1 / Bowtie2
SOAP2
BWA

- Algorithm that underlies multiple sequence alignment strategies
  - It was originally designed for as a data compression strategy
- The BWT transforms an input string by permuting its characters in such a way that the inverse transform can recover the original string without using any information aside from the permuted string

Slide courtesy of Andrew Farrell



Ben Langmead

https://genomebiology.biomedcentral.com/articles/10.1186/gb-2009-10-3-r25

https://www.youtube.com/watch?v=4n7NPk5lwbI

Pros and Cons of BWT

- Vast majority of sample sequence can be accurately placed

- Problems with:
  - Large scale differences - structural variation
  - Reference bias
  - Repetitive DNA

Slide courtesy of Andrew Farrell

Aligner	Strategy	Availability
BFAST	hash-based (genome)	open-source
Bowtie2*	Suffix array / BWT	open-source
BWA*	Suffix array / BWT	open-source
MAQ	hash-based (reads)	open-source
Mosaik*	hash-based (genome)	open-source
Novoalign*	hash-based (genome)	free for academic use
RMAP	hash-based (reads)	open-source
SOAP2	Suffix array / BWT	free for academic use
Eland	hash-based (reads)	commercial (Illumina)

Slide courtesy of Andrew Farrell

Unresolved problems

- Mapping tries to match the reference, so inherently introduces a bias towards the reference
- We have to modify parameters based on the read content (e.g. deletions)
- Mapping to repetitive DNA is still problematic
- What if there is no or an incomplete reference for the sequenced organism?