DNA Alignment Fundamentals

BFX Workshop

10/12/20

Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)

This is a human-readable summary of (and not a substitute for) the license. Disclaimer.

You are free to:

Share — copy and redistribute the material in any medium or format

Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

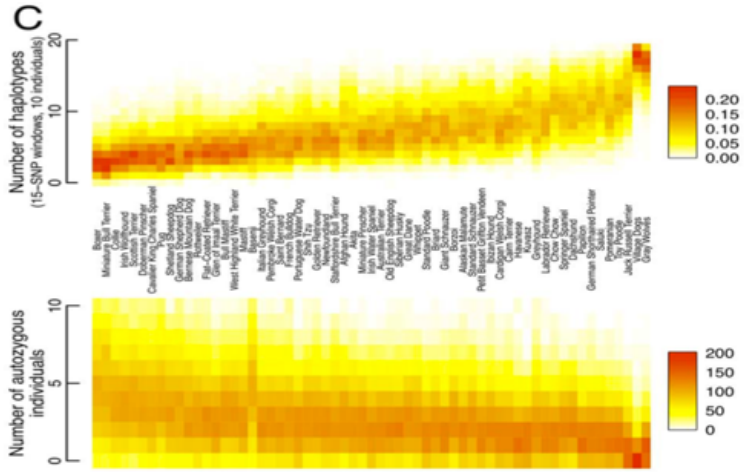
Credits

- Aaron Quinlan's course on Applied Computational Genomics:
  - https://github.com/quinlan-lab/applied-computational-genomics
- Malachi and Obi Griffith Lab course on Precision Medicine Bioinformatics:
  - https://pmbio.org/

The goal….

FASTQ

cctaaccct
acccctgg
tcccctgg
acccctgg
ggcccctgg
cctaaccct
cctaaccct
acccctgg
acccctgg
actaaccct
cctaaccct
cctaaccct
acccctgg
cctaaccct
acccctgg
cctaaccct



Sequence alignment is the crucial first step.

The problems…

- The human genome is big. Oh yeah, it's complex too.
- Sequencers can produce 1 billion reads / run.
- But they make mistakes. Frequently.
- **Accurate alignment takes time, but it's worth it.**
  - Shortcuts lead to artifacts
- Alignment strategy is highly nuanced, depending on experimental context

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

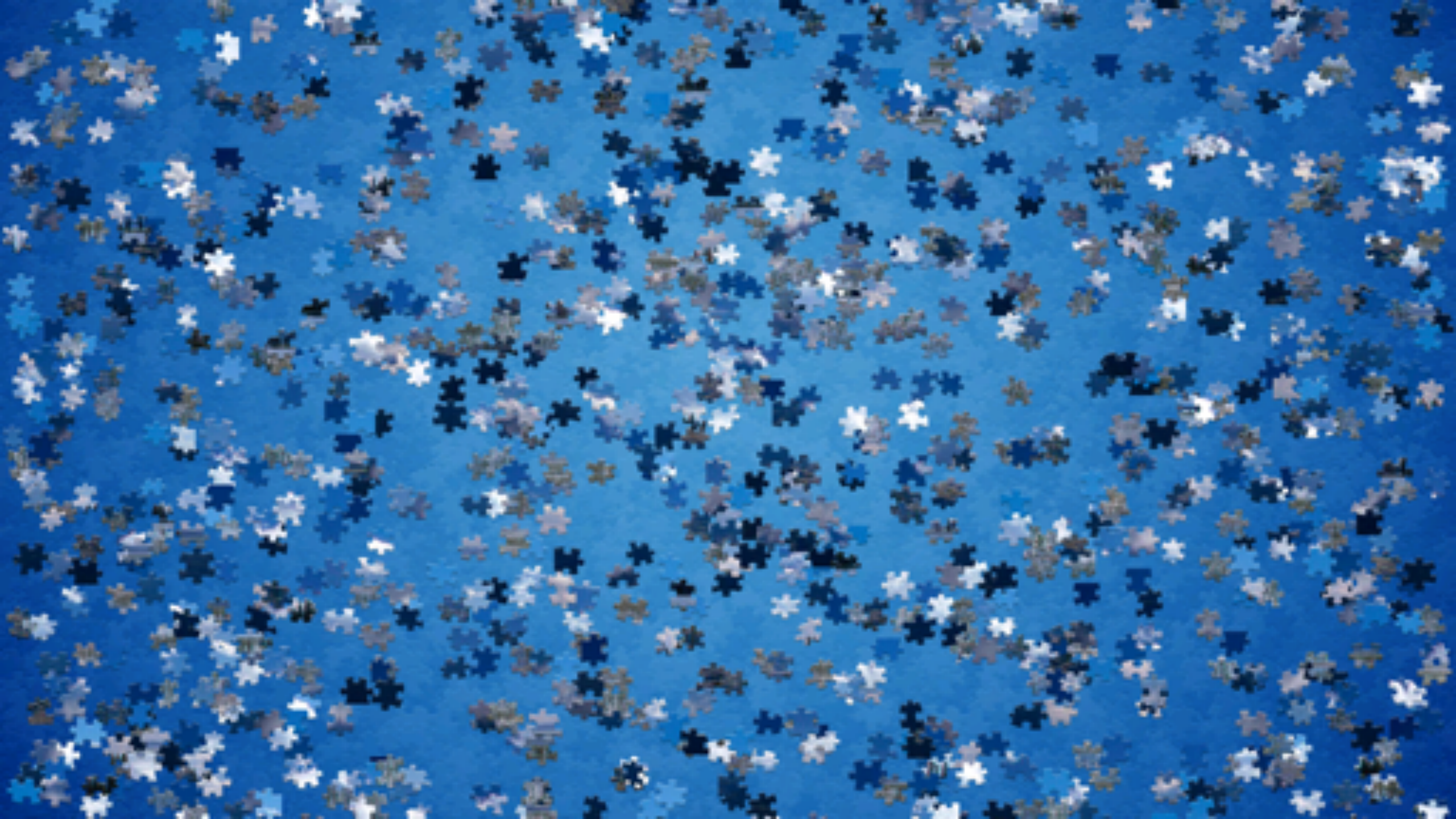
Phase1

Phase 2-100

Problem: Half of the human genome is comprised of repeats

ggcgcagagacgcaagcctacgggcgggggttgggggggcgtgtgttgca
ggagcaaagtcgcacggcgccgggctggggcgggggggagggtggcgccgt
gcacgcgcagaaactcacgtcacggtggcgcggcgcagagacgggtagaat
aacccTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTA
acccTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAAC
ccTAACCCAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCC
TAACCCTAACCCTAACCCTAACCCTAACCTAACCCTAACCCTAACCCTAA
CCCCCCTAACCCTAACCCTAACCCTAACCCCTAACCCTAACCCTAAACCC
CCCTAAACCCTAACCCTAACCCTAACCCTAACCCTAACCCCAACCCCAAC
CCCAACCCCAACCCCAACCCCAACCCTAACCCCTAACCCTAACCCTAACC
CTACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCCTAACCCC
TAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCCTAACCCT
AACCCTAACCCTAACCCTCGCGGTACCCTCAGCCGGCCCGCCCGCCCGGG
TCTGACCTGAGGAGAACTGTGCTCCGCCTTCAGAGTACCACCGAAATCTG
TGCAGAGGACAACGCAGCTCCGCCCTCGCGGTGCTCTCCGGGTCTGTGCT
GAGGAGAACGCAACTCCGCCGGCGCAGGCGCAGAGAGGCGCGCCGCGCCG
GCGCAGGCGCAGACACATGCTAGCGCGTCGGGGTGGAGGCGTGGCGCAGG
CGCAGAGAGGCGCGCCGCGCCGGCGCAGGCGCAGAGACACATGCTACCGC
GTCCAGGGGTGGAGGCGTGGCGCAGGCGCAGAGAGGCGCACCGCGCCGGC
GCAGGCGCAGAGACACATGCTAGCGCGTCCAGGGGTGGAGGCGTGGCGCA

(first bit of human chromosome 1)



Best case scenario: an error-free sequencing technology

ATTCGAAACA
TTCGCGCAAT
CTGGACTCAA

ATTCGAAACA
TTCGCGCAAT
CTGGACTCAA

Aligner

TACCTCCAGGGGGCATCCTCCCCCCCAATTCG
AAACACAATCGTAGCCCCTGGCACTACCTATG
TGTGTCAATTCGGAGAGAGAGAGATTCACGAA
AAAAAAGTCTGGACTCAACTAGGATACACACA
TTCGGCTACAGATACCAAAAAAAAAAAAAAA
AAATTTTCACCATTGAGGCACCACCTTCTCGT
CGCTGCGTCGCTCTGCTCGCTTCGGCTAAAAA
TTCGCGCAATACATTCGGCTACAGATACCAAA
AAAA

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

Reality check. Errors happen. Frequently.

ATTCGAAACA

→ ATTTGAAACA → **Aligner** →

TACCTCCAGGGGGCATCCTCCCCCCCAATTCG
AAACACAATCGTAGCCCCTGGCACTACCTATG
TGTGTCAATTCGGAGAGAGAGAGATTGGAAAC
AAAAAAGTGCTACAGATACCACTAGGATACAC
ACATTCGGCTACAGATACCAAAAAAAAAAAAA
AAAAAATTTTCACCATTGAGGCACCACCTTCT
CGTCGCTGCGTCGCTCTGCTCGCGGCTAAAAA
ATTAGAAACAACATTCGGCTACAGATACCAAA
ATTT

"Fuzzy" matching is much more computationally expensive.

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

Hash-based mapping:

Step1: hash/index the genome

Toy genome (16 bp)

CATGGTCATTGGTTCC

Hash-based mapping:

Step1: hash/index the genome

CATGGTCATTGGTTCC

k = 3

Kmer/Hash	Genome Positions
CAT	1

Hash-based mapping:

Step1: hash/index the genome

CATGGTCATTGGTTCC

k = 3

Kmer/Hash	Genome Positions
CAT	1
ATG	2

Hash-based mapping:

Step1: hash/index the genome

CATGGTCATTGGTTCC

k = 3

Kmer/Hash	Genome Positions
CAT	1
ATG	2
TGG	3

Hash-based mapping:

Step1: hash/index the genome

CATGGTCATTGGTTCC

k = 3

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

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

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

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

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)

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.

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

Hash-based mapping:

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

Toy genome

CATGGTCATTGGTTCC

Read TGGTCA

Hash matches 3,10

Hash match

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-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

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

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.

Hash-based mapping:

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

Toy genome

CATGGTCATTGGTTCC

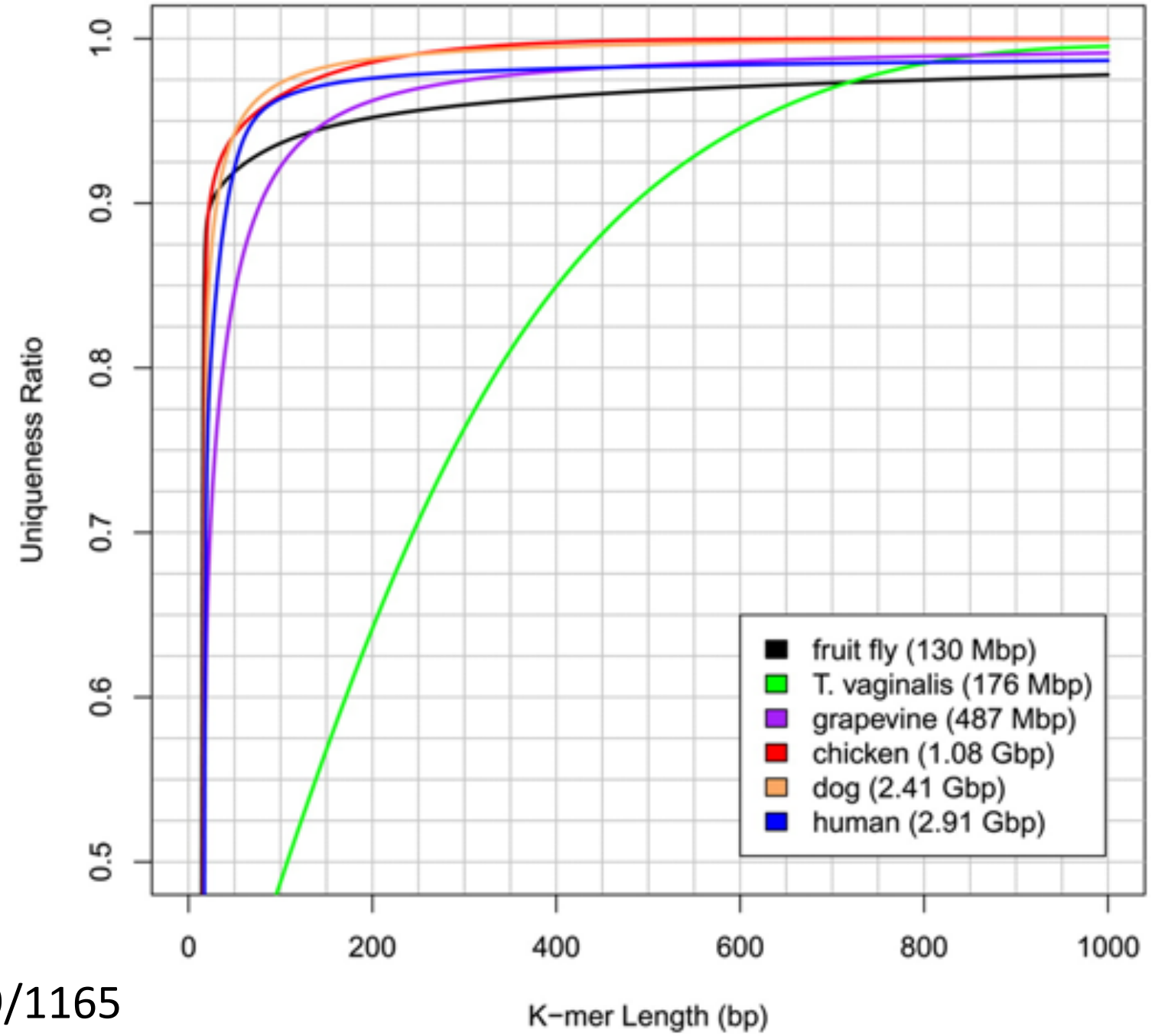
Read TGGTCT

Hash matches 3,10

Hash match

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

It takes a very long k-mer to be unique in most genomes!



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

Mapping quality (MAPQ)

What is the probability that the sequence should be mapped here and only here?

MAPQ also uses the Phred (log) scale:

$$MAPQ = -10*\log_{10}(P_{map\_loc\_wrong})$$

$(P_{map\_loc\_wrong})$	$\log_{10}(P_{map\_loc\_wrong})$	MAPQ
1	0	0
0.1	-1	10
0.01	-2	20
0.001	-3	30
0.0001	-4	40

Edit distance

How many edits (changes) must be made to a word or kmer to make it match (align) to another word or kmer?

CURLED
HURLED → Edit distance = 1. Substitute C for H

SHORT
SHO-T → Edit distance = 1. Delete R

TGTTACGG
GGTTGACTA
?

TG-TT-ACGG
-GGTTGACTA

Edit distance = 5

TGTT-ACGG
GGTTGACTA

Edit distance = 4

Key Alignment Algorithms

Get Access Share Export

jmb
Journal of Molecular Biology
Volume 48, Issue 3, 28 March 1970, Pages 443-453

A general method applicable to the search for similarities in the amino acid sequence of two proteins ☆

Saul B. Needleman, Christian D. Wunsch

Show more

https://doi.org/10.1016/0022-2836(70)90057-4

Get rights and content

Identification of Common Molecular Subsequences

T. F. Smith and M. S. Waterman

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.

These developments in the modern sequence analysis began with the heuristic homology algorithm of Needleman & Wunsch (1970) which first introduced an iterative matrix method of calculation. Numerous other heuristic algorithms have been suggested including those of Fitch (1966) and Dayhoff (1969). More mathematically rigorous algorithms were suggested by Sankoff (1972), Reichert *et al.* (1973) and Beyer *et al.* (1979), but these were generally not biologically satisfying or interpretable. Success came with Sellers (1974) development of a true metric measure of the distance between sequences. This metric was later generalized by Waterman *et al.* (1976) to include deletions/insertions of arbitrary length. This metric represents the minimum number of "mutational events" required to convert one sequence into another. It is of interest to note that Smith *et al.* (1980) have recently shown that under some conditions the generalized Sellers metric is equivalent to the original homology algorithm of Needleman & Wunsch (1970).

In this letter we extend the above ideas to find a pair of segments, one from each of two long sequences, such that there is no other pair of segments with greater similarity (homology). The similarity measure used here allows for arbitrary length deletions and insertions.

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

Scoring scheme:

Match: +3

Mismatch -3

Gap: -2

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

Local: Smith-Waterman algorithm

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

Global: Needleman-Wunsch algorithm

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

BWA-MEM: never "published" ; widely used.

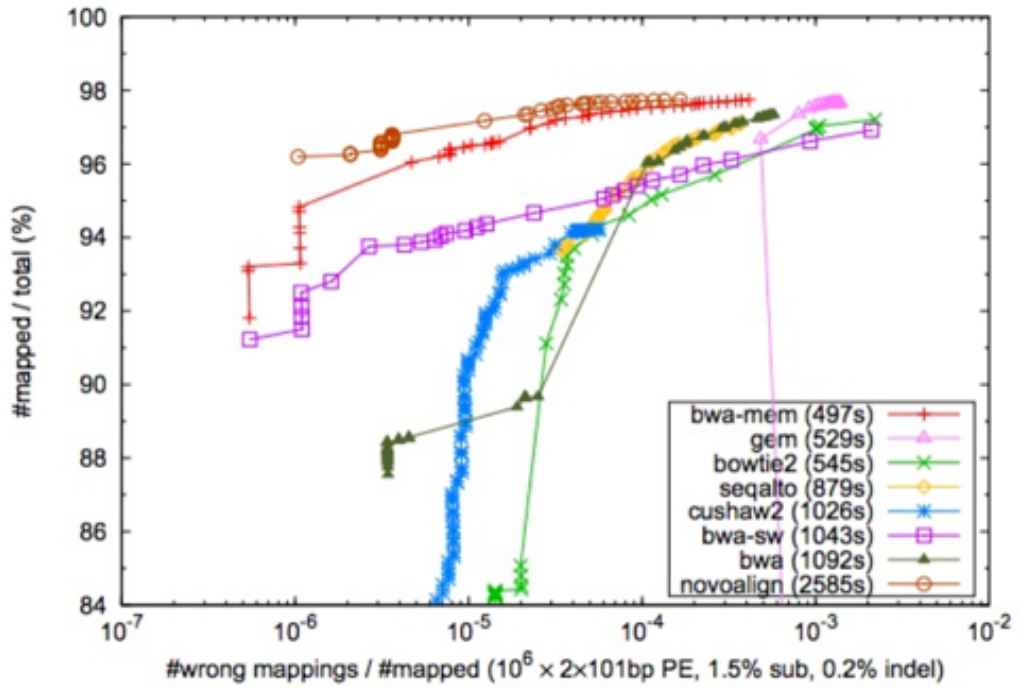


Fig. 1. Percent mapped reads as a function of the false alignment rate under different mapping quality cutoff. Alignments with mapping quality 3 or lower are excluded. An alignment is *wrong* if after correcting clipping, its start position is within 20bp from the simulated position. 10^6 pairs of 101bp reads are simulated from the human reference genome using wgsim (http://bit.ly/wgsim2) with 1.5% substitution errors and 0.2% indel variants. The insert size follows a normal distribution $N(500, 50^2)$. The reads are aligned back to the genome either as single end (SE; top panel) or as paired end (PE; bottom panel). GEM is configured to allow up to 5 gaps and to output suboptimal alignments (option '–e5 –m5 –s1' for SE and '–e5 –m5 –s1 –pb' for PE). GEM does not compute mapping quality. Its mapping quality is estimated with a BWA-like algorithm with suboptimal alignments available. Other mappers are run with the default setting except for specifying the insert size distribution. The run time in seconds on a single CPU core is shown in the parentheses.

Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM

Heng Li

Broad Institute of Harvard and MIT, 7 Cambridge Center, Cambridge, MA 02142, USA

https://arxiv.org/pdf/1303.3997v2.pdf

Sequence alignment software

Aligner	Approach	Applications	Availability
BWA-mem	Burrows-Wheeler	DNA, SE, PE, SV	open-source
Bowtie2	Burrows-Wheeler	DNA, SE, PE, SV	open-source
Novoalign	hash-based	DNA, SE, PE	free for academic use
TopHat	Burrows-Wheeler	RNA-seq	open-source
STAR	hash-based (reads)	RNA-seq	open-source
GSNAP	hash-based (reads)	RNA-seq	open-source

BWA-MEM

Unaligned Sample Data In FASTQ (SE or PE)

```
@seq1
ATTCGAAACA...
+
DDED88(999...
@seq2
CCCCGTTTCA...
+
AAC887BBAC...
```

Reference genome (FASTA)

```
>chr1
TACCTCCAGGGGGCATCCTCCCCCCCAATTC
GAAACACAATCGTAGCCCCTGGCACTACCTA
TGTGTGTCAATTCGGAGAGAGAGAGATTCAC
GAAAAAAAAGTCTGGACTCAACTAGGATACA
CACATTCGGCTACAGATACCAAAAAAAAAAA
AAAAAAAATTTTCACCATTGAGGCACCACCT
TCTCGTCGCTGCGTCGCTCTGCTCGCTTCGG
CTAAAAATTCGCGCAATACATTCGGCTACAG
ATACCAAA
```

BWA MEM

Aligned Sample Data in SAM format

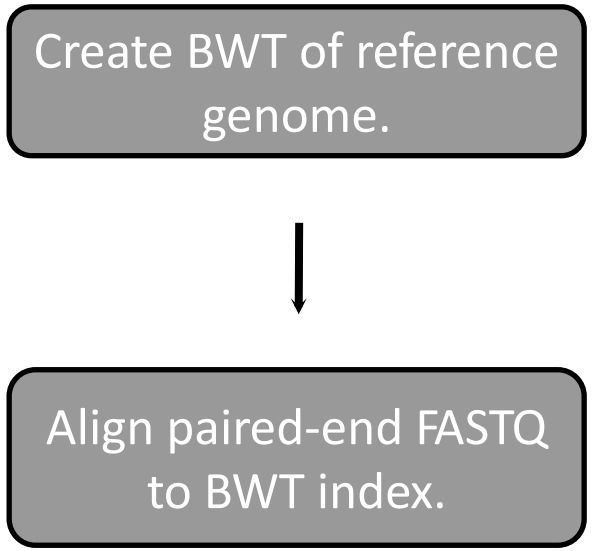
```
seq1        99          1           3666901
            60          149M        =
            3666935     185
            ATTCGAAACA...           DDED88(999
            MC:Z:151M MD:Z:149      RG:Z:15-
0017315_1   NM:i:0      MQ:i:60     AS:i:149
            XS:i:44
seq2        147         1           3666935
            60          151M        =
            3666901     -185
            CCCCGTTTCA...
            AAC887BBAC...           MC:Z:149M
            MD:Z:151    RG:Z:15-0017315_1
            NM:i:0      MQ:i:60     AS:i:151
            XS:i:59
```

BWA-MEM workflow

This takes a long time, but you do it once

Output is in SAM format.

Use multiple threads if you have a computer with multiple CPUs.




```
$ bwa index grch38.fa
```

```
$ bwa mem -t 16 grch38.fa 1.fq 2.fq > sample.sam
```

SAM format: a **text**-based **standard(!)** for representing sequence alignments

BIOINFORMATICS **APPLICATIONS NOTE** *Vol. 25 no. 16 2009, pages 2078–2079*
doi:10.1093/bioinformatics/btp352

Sequence analysis

The Sequence Alignment/Map format and SAMtools

Heng Li[1,†], Bob Handsaker[2,†], Alec Wysoker[2], Tim Fennell[2], Jue Ruan[3], Nils Homer[4], Gabor Marth[5], Goncalo Abecasis[6], Richard Durbin[1,*] and 1000 Genome Project Data Processing Subgroup[7]

[1]Wellcome Trust Sanger Institute, Wellcome Trust Genome Campus, Cambridge, CB10 1SA, UK, [2]Broad Institute of MIT and Harvard, Cambridge, MA 02141, USA, [3]Beijing Institute of Genomics, Chinese Academy of Science, Beijing 100029, China, [4]Department of Computer Science, University of California Los Angeles, Los Angeles, CA 90095, [5]Department of Biology, Boston College, Chestnut Hill, MA 02467, [6]Center for Statistical Genetics, Department of Biostatistics, University of Michigan, Ann Arbor, MI 48109, USA and [7]http://1000genomes.org

Received on April 28, 2009; revised on May 26, 2009; accepted on May 30, 2009

Advance Access publication June 8, 2009

Associate Editor: Alfonso Valencia

Table 1. Mandatory fields in the SAM format

No.	Name	Description
1	QNAME	Query NAME of the read or the read pair
2	FLAG	Bitwise FLAG (pairing, strand, mate strand, etc.)
3	RNAME	Reference sequence NAME
4	POS	1-Based leftmost POSition of clipped alignment
5	MAPQ	MAPping Quality (Phred-scaled)
6	CIGAR	Extended CIGAR string (operations: MIDNSHP)
7	MRNM	Mate Reference NaMe ('=' if same as RNAME)
8	MPOS	1-Based leftmost Mate POSition
9	ISIZE	Inferred Insert SIZE
10	SEQ	Query SEQuence on the same strand as the reference
11	QUAL	Query QUALity (ASCII-33=Phred base quality)

SAM format overview

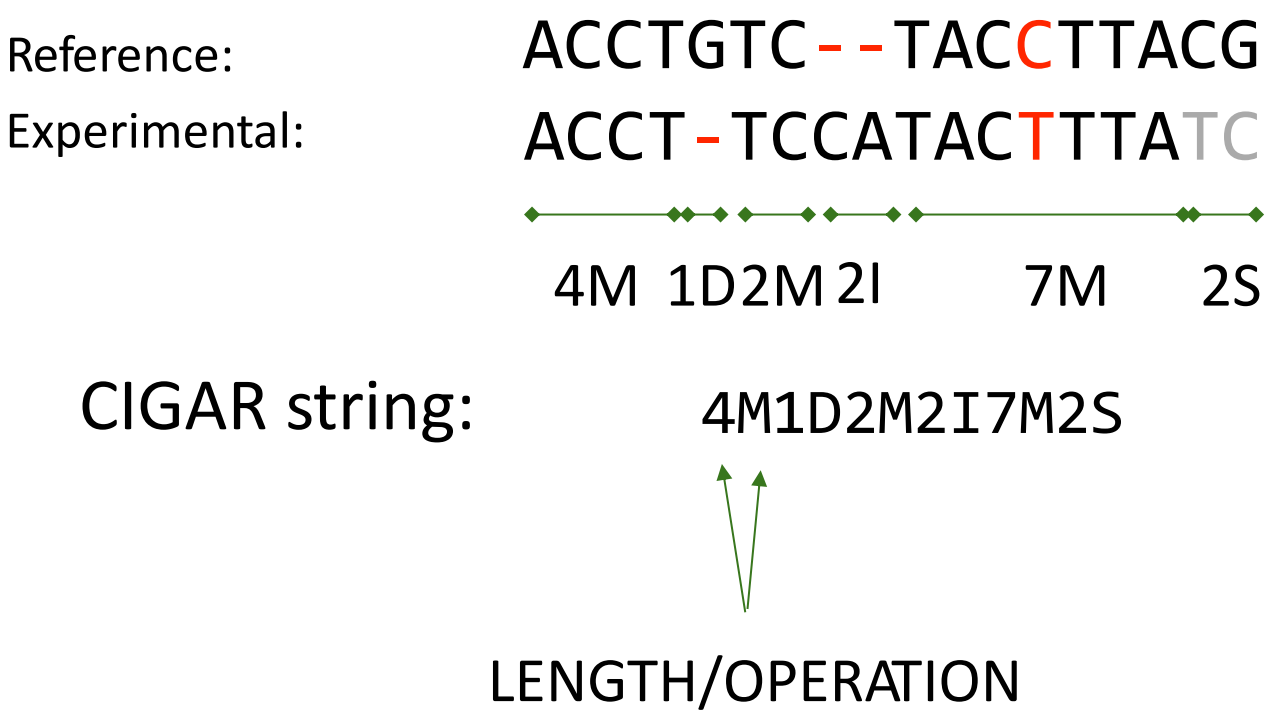
http://samtools.sourceforge.net/samtools.shtml

Col #	Name	Meaning	Example
1	QNAME	Read or Pair name	HWI:ST156_1:278:1:1058:4544:0
2	FLAG	Bitwise FLAG	***Much more soon!***
3	RNAME	Reference sequence name	chr1
4	POS	1-based alignment start coordinate	8,724,005
5	MAPQ	Mapping quality	60
6	CIGAR	Extended CIGAR string	***Much more soon!***
7	MRNM	If paired, the mate's reference seq.	chr1
8	MPOS	If paired, the mate's alignment start	8,724,505
9	ISIZE	If paired, the insert size	562
10	SEQ	The sequence of the query/mate	ACAAATTCAG...
11	QUAL	The quality string for the query/mate	HHH$^^%$$$...
12	OPT	Optional Tags	XA:i:2, MD:Z:0T34G15

ST-E00223:32:H5J57CCXX:6:2123:15189:52872 97 1 10001 0 4S15M1I54M2I50M25S 4 699063 0
ACCCTAACCCTAACCCTAACCCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAAACCCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCAACCCTAACCC
CCCCCCACCCAACCCCACCCCCCAC
AAFFFKKKKKKKKKKKKKKKKKKFKKFKKKKKKFKKAKKKKFFKKKKKFKF<FF7FFFFK7FK,AA,FKFFKF,FKKK7<KKK,,7FFF7F,AAFFK,AKA,,FFF<,A7FKK,,,7A,,,7,,
,,,((,(,,(,,,,,,,(,,A(,,(MC:Z:80S11M1D58M MD:Z:119 RG:Z:15-0017315_1 NM:i:3 MQ:i:47 AS:i:104
XS:i:103
ST-E00223:46:HG7V5CCXX:2:1116:12601:22862 1123 1 10006 0 81M70S = 10106 143
CTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACACTCACCCTAACCCTACCCCTAACCCTAACCCTATCATTCACTCGAACCCTAACACTACCGCTAGCGCTAACTCCAGCC
CGCACACTATCGCTAACCCTCACGC
AAFFFKKKKKKKKKKKKKKFKKA,,A,A,,77<,,A,,7FFK7,,,,AF,,,A7,,,77,,,7AA,,7,FFK,<A,,,,7<<,,,,AA,7,AA,,7,,,7,,,,,,,,,,A7,,,,,,7,7,7(
,(,((,,,7,A,F,7,,,<,,AA,, MC:Z:108S43M MD:Z:47C2A12A17 RG:Z:15-0017315_1 NM:i:3 MQ:i:2 AS:i:66 XS:i:71
ST-E00223:32:H5J57CCXX:5:2208:10074:43308 99 1 10008 36 101M1I41M7S = 10107 137
AACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTACCCCTAACCCTAACCCTAACCCTAACCCTCACCCTCACCCTCACCCTAGCCCAAACCCTAACCCTAACCCTAACCCT
AACCCTAACCCTAACCCTACCCCG
AAFFFKFKKKKKKKKKKKKKKKFKKKKKKFKKK7<KA7<F,AF7F,7,A,FA7<KKKKKFKKKK<,,FKAKKAFFA,A7,7,,,7,7,,,,,7F<,,7,,,,7,,A,,,<F7,,,7FA,7,,<F
,,,<A,7<AFKK<,<,,7,,,,,(MC:Z:112S38M MD:Z:49A28A5A5A6A44 RG:Z:15-0017315_1 NM:i:6 MQ:i:36 AS:i:110
XS:i:113
ST-E00223:46:HG7V5CCXX:5:2119:12936:64896 99 1 10013 0 90M61S = 10176 211
TAACCCTAAGCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCATAACCCAAACCCTAACCCTAACCCTAACCCGAACCGTAAGCCAAAACATAACCACAACCATAACAA
TAACCAAAACCTTAACGTTAAACAT
A<AFFKKKK,AAAFKK<FKKKKAK,<A,AFKKKKAFFFKKKKAFKKFAFFFKKA7,FFA,F,F,7F,AFF77AFFAFFKAFKKA,,FFKF,AA,F,,7F,A,,7A,,,7A,,,,,,AFK,,,AA,,
,7FKA,A,AFF,<,,,,,,,77<AA MC:Z:99S48M4S MD:Z:9C49C6T23 RG:Z:15-0017315_1 NM:i:3 MQ:i:0 AS:i:75 XS:i:75
ST-E00223:32:H5J57CCXX:1:1205:17290:54577 99 1 10019 1 92M59S = 10354 414
TAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCAAACCCCCACCCCAAACCCAAGCCCAACCCTACCCC
CTAACCCCTAACCCCAACCCTGACC
AAFFFKKKKFKKKKAKKKKKKKKKKKKKKF7KKKKKKKKKKKKKFKKKKFKKKKKKKAKKKFFKKFFKKKK,<,7<FA,FFFK,,7FFF,,,,,A,((,(((,7,7A,,,,,A,,,777A,AAKK
<<,7FFA7,A,,AA,<,,AFA,7AF MC:Z:72S79M MD:Z:92 RG:Z:15-0017315_1 NM:i:0 MQ:i:20 AS:i:92 XS:i:97

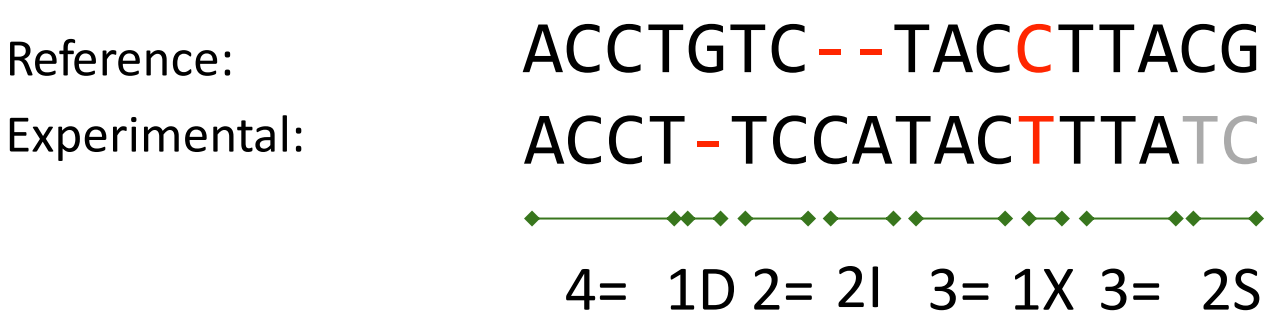
The CIGAR string: encode the details of the alignment

Operation	Meaning
M	Match*
D	Deletion w.r.t. reference
I	Insertion w.r.t. reference
N	Split or spliced alignment
S	Soft-clipping
H	Hard-clipping
P	Padding



The extended CIGAR string: M become = and X

Operation	Meaning
=	Exact match
X	Mismatch
D	Deletion w.r.t. reference
I	Insertion w.r.t. reference
N	Split or spliced alignment
S	Soft-clipping
H	Hard-clipping
P	Padding



CIGAR string: 4=1D2=2I3=1X3=2S

The FLAG column

Sequence ID	FLAG	CHROM	POS
ST-E00223:32:H5J57CCXX:6:2123:15189:52872	97	1	10001
ST-E00223:46:HG7V5CCXX:2:1116:12601:22862	1123	1	10006
ST-E00223:32:H5J57CCXX:5:2208:10074:43308	99	1	10008
ST-E00223:46:HG7V5CCXX:5:2119:12936:64896	99	1	10013
ST-E00223:32:H5J57CCXX:1:1205:17290:54577	99	1	10019
ST-E00223:32:H5J57CCXX:6:1115:16844:11013	81	1	10026
ST-E00223:32:H5J57CCXX:7:2113:18935:32356	99	1	10032
ST-E00223:46:HG7V5CCXX:6:2117:3082:44239	99	1	10040
ST-E00223:46:HG7V5CCXX:5:2213:10744:58813	163	1	10074
ST-E00223:32:H5J57CCXX:4:1220:14651:8868	99	1	10086

The FLAG score

base2	base10	base16	Meaning	Applies to:
00000000001	1	0x0001	The read originated from a paired sequencing molecule	Both
00000000010	2	0x0002	The read is mapped in a **proper** pair	Pairs only
00000000100	4	0x0004	The query sequence itself is unmapped	Both
00000001000	8	0x0008	The query's mate is unmapped	Pairs only
00000010000	16	0x0010	Strand of the query (0 for forward; 1 for reverse strand)	Both
00000100000	32	0x0020	Strand of the query's mate	Pairs only
00001000000	64	0x0040	The query is the first read in the pair	Pairs only
00010000000	128	0x0080	The read is the second read in the pair	Pairs only
00100000000	256	0x0100	The alignment is not primary	Both
01000000000	512	0x0200	The read fails platform/vendor quality checks	Both
10000000000	1024	0x0400	The read is either a PCR duplicate or an optical duplicate	Both

ST-E00223:32:H5J57CCXX:4:1220:14651:8868 99 1 10086

base2	base10	base16	Meaning	Applies to:
00000000001	1	0x0001	The read originated from a paired sequencing molecule	Both
00000000010	2	0x0002	The read is mapped in a **proper** pair	Pairs only
00000000100	4	0x0004	The query sequence itself is unmapped	Both
00000001000	8	0x0008	The query's mate is unmapped	Pairs only
00000010000	16	0x0010	Strand of the query (0 for forward; 1 for reverse strand)	Both
00000100000	32	0x0020	Strand of the query's mate	Pairs only
00001000000	64	0x0040	The query is the first read in the pair	Pairs only
00010000000	128	0x0080	The read is the second read in the pair	Pairs only
00100000000	256	0x0100	The alignment is not primary	Both
01000000000	512	0x0200	The read fails platform/vendor quality checks	Both
10000000000	1024	0x0400	The read is either a PCR duplicate or an optical duplicate	Both

00001100011

$2^6+2^5+2^1+2^0 = 64+32+2+1 = 99$

https://broadinstitute.github.io/picard/explain-flags.html

Use samtools to convert SAM to BAM.

This takes a long time, but you do it once

Create BWT of reference genome.

```
$ bwa index grch38.fa
```


Output is in SAM format.

Use multiple threads if you have a computer with multiple CPUs.

Align paired-end FASTQ to BWT index.

```
$ bwa mem -t 16 grch38.fa 1.fq 2.fq > sample.sar
```


Output is in BAM format.

However, it is unsorted - that is, random genomic order as reads are randomly placed in FASTQ by sequencer.

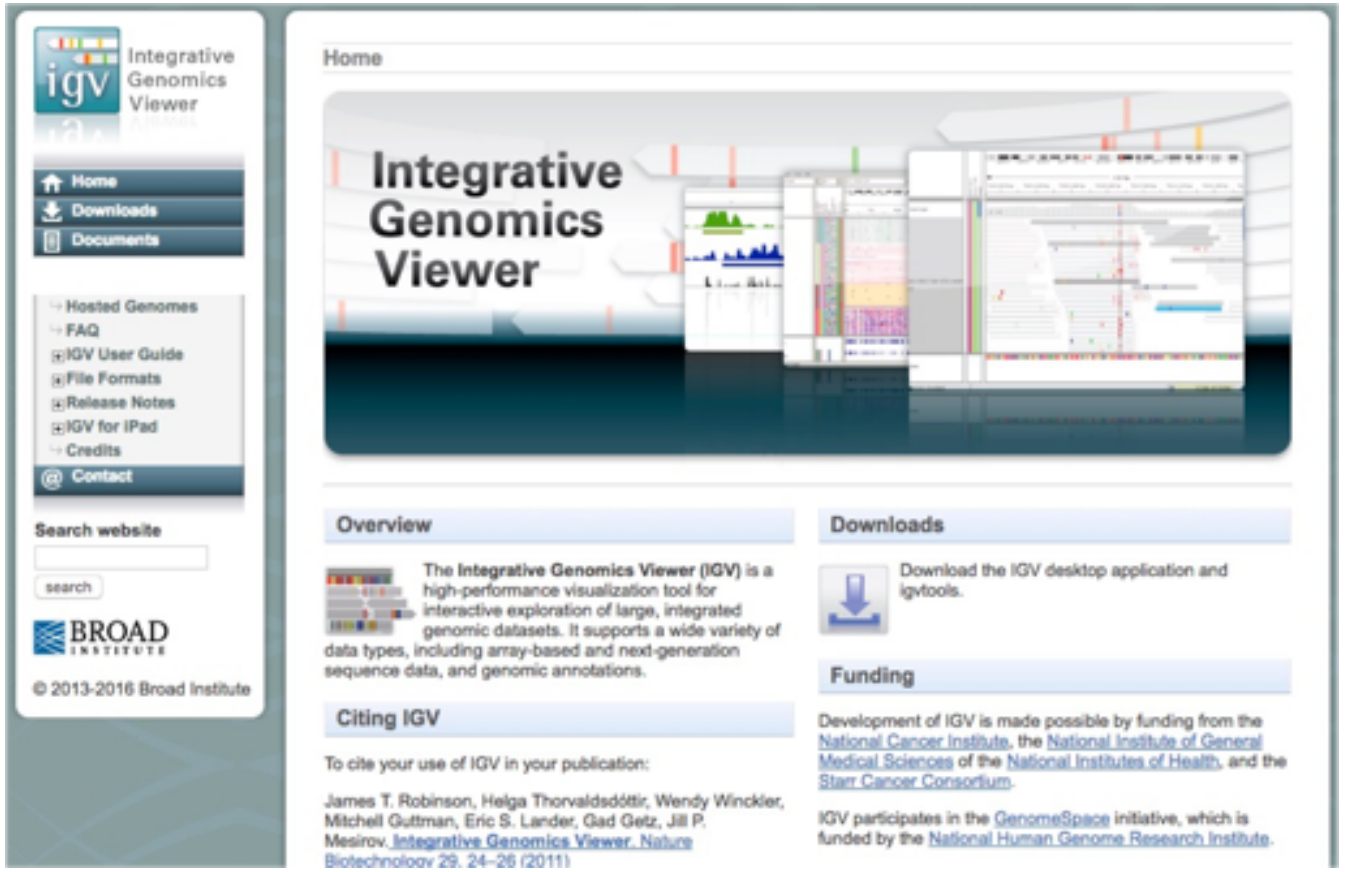
Convert SAM to BAM

```
$ samtools view -Sb sample.sam > sample.bam
```

This week's Tutorial and Homework

- git clone https://github.com/genome/bfx-workshop.git
  - Or `git pull` if you've already cloned it before.

IGV tutorial



https://github.com/griffithlab/rnaseq_tutorial/wiki/IGV-Tutorial