# BCB 716 - Sequence Analysis

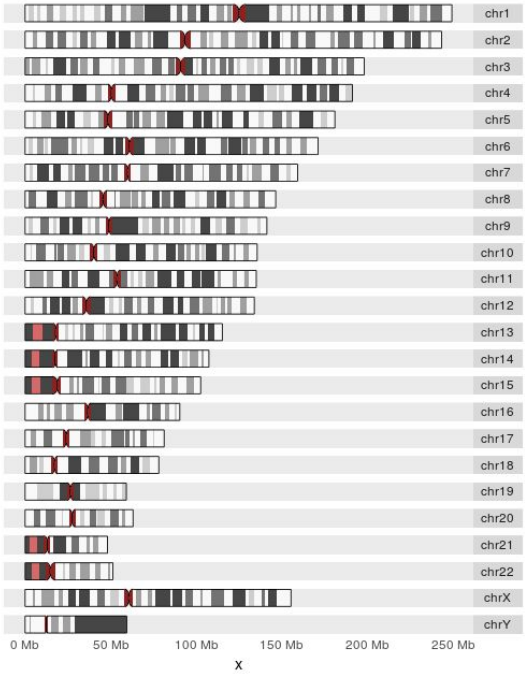

DNA Alignment Methods and Analysis

# Problem Statement

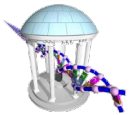
- **Given 600 million 150 base reads from a high-throughput sequencer**

  600,000,000 reads x 150 bases / 30x coverage = 3,000,000,000 bp genome

- Where in the genome
  does each read best match?
  - Allow for variants
    - SNPs, INDELs
    - Sequencing noise
  - Consider paired-end mates
    - Mated reads should align nearby on the same chromosome and match opposite Strands
- **Spending 1mS per read implies ~166.7 hrs**
- **Spending 1µS per base implies ~25 hrs**



# One Read

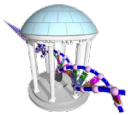


**A 150bp read from some not C57BL/6J mouse**

```
AGAGATCCGAGTAATTAATACTAAATAGAACAGTTAGGGCAGGAGGGTTTCAGGTCGATCTGAACTATCACTGAG
GAAATGTCTCTGCATAATTATCTAAGGACATGTAAGGTATGAAGGTAGGGGTAGAAACTGTACTTAGCTGGTTTG
```

- Go for broke and search for the entire string in the mouse reference genome
  You wouldn't find it about 98% of the time (Due to genetic variants + sequencing noise)
- You could search for it in pieces, and see if those pieces "fit"
  How big?
- Depends on how many errors that you allow per read
- Suppose we will tolerate no more than 5 errors in a read (3%)
  (including both genetics and noise)
- What is the smallest possible exact match that we can expect

# Dealing with Errors



**We could put all the errors together on one side**

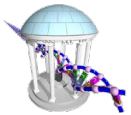
XXXXX ---------- 145 base perfect match ----------

**Or spread them out evenly**

---24 bases--- X ---24 bases--- X ---24 bases--- X ---25 bases--- X ---24 bases --- X --- 24 bases ---

**There has to be a run of at least 25 exact matches if we only allow for only 5 errors**

# Let's try it



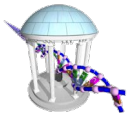
## Chop the read into 6, 25-base pieces.

AGAGATCCGAGTAATTAATACTAAATAGAACAGTTAGGGCAGGAGGGTTTCAGGTCGATCTGAACTATCACTGAG
GAAATGTCTCTGCATAATTATCTAAGGACATGTAAGGTATGAAGGTAGGGGTAGAAACTGTACTTAGCTGGTTTG

```
AGAGATCCGAGTAATTAATACTAAA (perfect match):    None
TAGAACAGTTAGGGCAGGAGGGTTT (perfect match):    None
CAGGTCGATCTGAACTATCACTGAG (perfect match):    Found at 13:  91469192
GAAATGTCTCTGCATAATTATCTAA (perfect match):    Found at 13:  91469217
GGACATGTAAGGTATGAAGGTAGGG (perfect match):    Found at 13:  91469242
GTAGAAACTGTACTTAGCTGGTTTG (perfect match):    Found at 13:  91469267
```

```
                    Read: AGAGATCCGAGTAATTAATACTAAA
      Reference 13:91469142: AGAGATCCGAGTAATTAACACTAAA
                           ****************** ******

                    Read: TAGAACAGTTAGGGCAGGAGGGTTT
      Reference 13:91469167: TAGAACAGTTAGGGCAGGAGGATTT
                           ********************* ***
```

# Another One



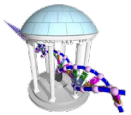
```
GGAAAATACCTAATAATAATAATAAAAAAACATTGACAGACAAGCAGATGGTACTTCAGTAACAAAGCTTAAGTT
TGAATATTTTTTTTCATTCTCTTCTGATTCTTATCTATCTTTGTGCCATTCACAACTAAAAATGAGTTAAAATTT
```

```
GGAAAATACCTAATAATAATAATAA (perfect match):    105 matches
AAAAACATTGACAGACAAGCAGATG (perfect match):    None
GTACTTCAGTAACAAAGCTTAAGTT (perfect match):    1:  22323830        2:  53299690
TGAATATTTTTTTTCATTCTCTTCT (perfect match):    None
GATTCTTATCTATCTTTGTGCCATT (perfect match):    1:  22323882        2:  53299739
CACAACTAAAAATGAGTTAAAATTT (perfect match):    1:  22323907
```

```
      Read: GGAAAATACCTAATAATAATAATAA
1:22323780: GGCAATGTGATTCAAAGGTCTGTAA
            ** **     *   **      *** 15

      Read: AAAAACATTGACAGACAAGCAGATG
1:22323805: TAAAACATTGACAGACAAGCCGATG
             ****************** **** 2

      Read: _TGA_ATATTTTTTTTCATTCTCTTCT
1:22323855: CAGATATATATTTTTTCATTCTCTTCT
             ** **** ***************** 4
```

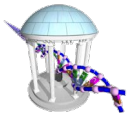
```
      Read: GGAAAATACCTAATAATAATAATAA
2:53299640: GGAAAATATCTAATAATAATAA___
            ******** *************    4

      Read: AAAAACATTGACAGACAAGC____AGATG
2:53299665: AAAAACATTGACAGACAAGCCGATAGATG
            ********************    ***** 4

      Read: TGAATATTTTTTTTCATTCTCTTCT
2:53299715: CGAATATTTTTTT_CATTCTCTTCT
             ************ *********** 2

      Read: CACAACTAAAAATGAGTTAAAATTT
2:53299764: CACAACTAAAAGTGAGTTAAAATTT
            *********** ************ 1
```

# Finding substring “seeds” (quickly)



- **Just look through the whole string**
  - *Very* slow
- **Hash table/dictionary**
  - *Lots* of memory
- **Burrows Wheeler Transform/FM-index**
  - Complete sequence, space- and time-efficient
- **Minimizers**
  - Samples sequence, fastest

# Burrows Wheeler Transform

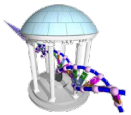


- An invertible transformation of a string that compresses well
- By convention we end the string with a '$' which is alphabetically first
- The predecessor symbols of the sorted suffixes of a sequence treated as a circular string
- The BWT transforms "banana$" to "annb$aa"

| Suffixes | Sorted |
|---|---|
| banana$ | $banana |
| anana$b | a$banan |
| nana$ba | ana$ban |
| ana$ban | anana$b |
| na$bana | banana$ |
| a$banan | na$bana |
| $banana | nana$ba |

- There is a very clever algorithm that allows one to search for all occurrences of a substring of length *n* in a text using a BWT that requires *O(n)* steps regardless of the length of the text.

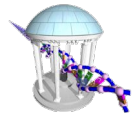
# The FM-index



- The FM-index is another *helper* data structure
- It is a 2D array whose size is [|*text*|+1,|Σ|], where |Σ| is the alphabet size
- It keeps track of how many of each symbol have been seen in the BWT prior to its ith symbol
- The last *m* row is the totals for each symbol. By accumulating these totals you can determine the BWT index corresponding to the first of each symbol in the suffix array (Offset).
- Can be generated by a single scan through the BWT
- Memory overhead O(*m*|Σ|)

| | | | FM-index | | | |
|---|---|---|---|---|---|---|
| **Index** | **Suffix Array** | **BWT** | **$** | **a** | **b** | **n** |
| 0 | $banana | a | 0 | 0 | 0 | 0 |
| 1 | a$banan | n | 0 | 1 | 0 | 0 |
| 2 | ana$ban | n | 0 | 1 | 0 | 1 |
| 3 | anana$b | b | 0 | 1 | 0 | 2 |
| 4 | banana$ | $ | 0 | 1 | 1 | 2 |
| 5 | na$bana | a | 1 | 1 | 1 | 2 |
| 6 | nana$ba | a | 1 | 2 | 1 | 2 |
| 7 | **Counts** | | 1 | 3 | 1 | 2 |
| | **Offset** | | 0 | 1 | 4 | 5 |

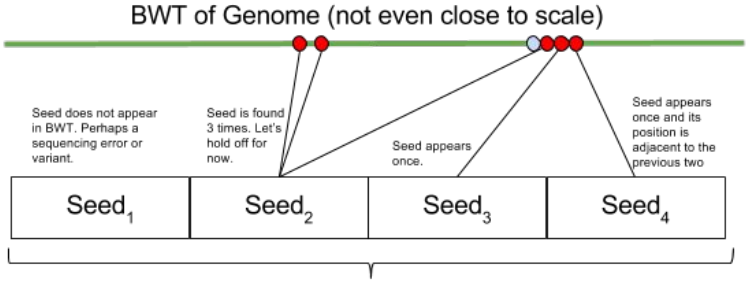
# Real-World uses of BWTs

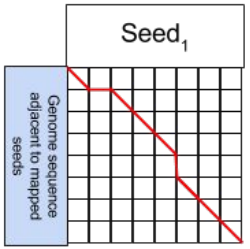


BWTs are the dominant representation and method used for ***Sequence Alignment***

- Sequence-Alignment Problem: Given a collection of short nucleotide fragments (either DNA or RNA) find the best approximate alignment for each fragment in a reference genome

- Bowtie2 (2012) and BWA (2009) are the dominant aligners
- As a preprocess a BWT of the reference genome is built (≈ 1-3 GB)
- How alignment works:
  - given a *read* from a sequenced fragment (72-150 base pairs typically)
  - cut the read into smaller seeds (25-31 base pairs typically)
  - Search for an exact match to each using the BWT
  - Use local alignment (dynamic program) to match the remaining bases

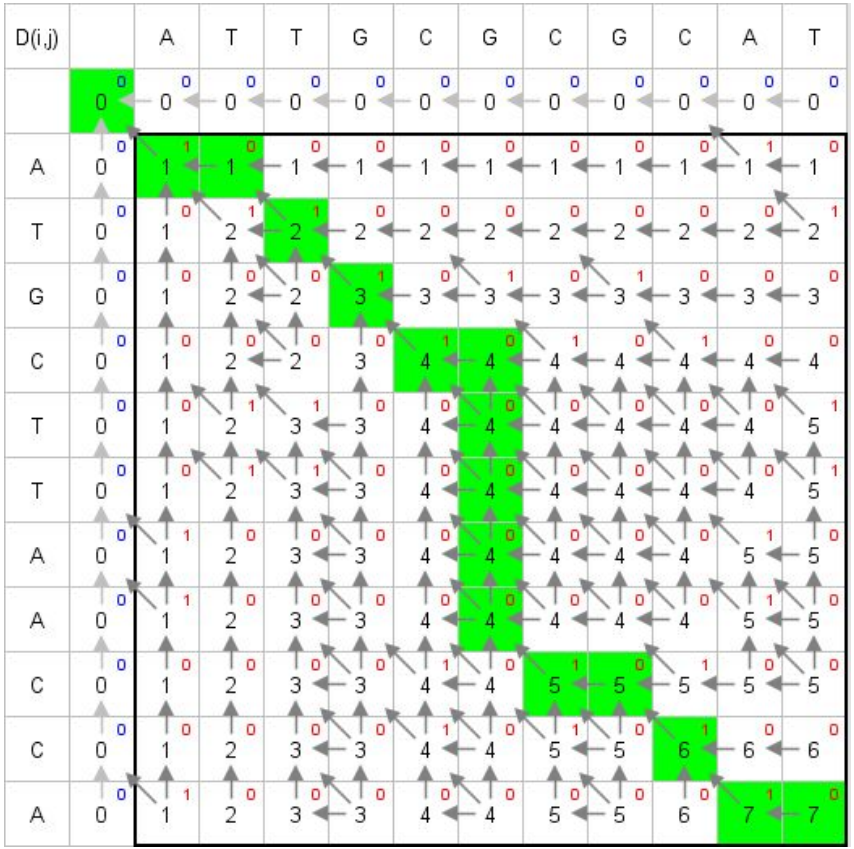


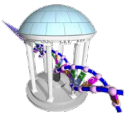

Perform a global sequence alignment on the unmatched seeds and report back the score

# Dynamic programming

**Used to discover optimal global or local alignment of two strings, allowing for insertions, deletions, and mismatches under some scoring scheme.**



# Aligners for Long Reads



- **Long read alignment presents challenges**
  - Increased number of seeds (25-mers)
  - Higher Error rate

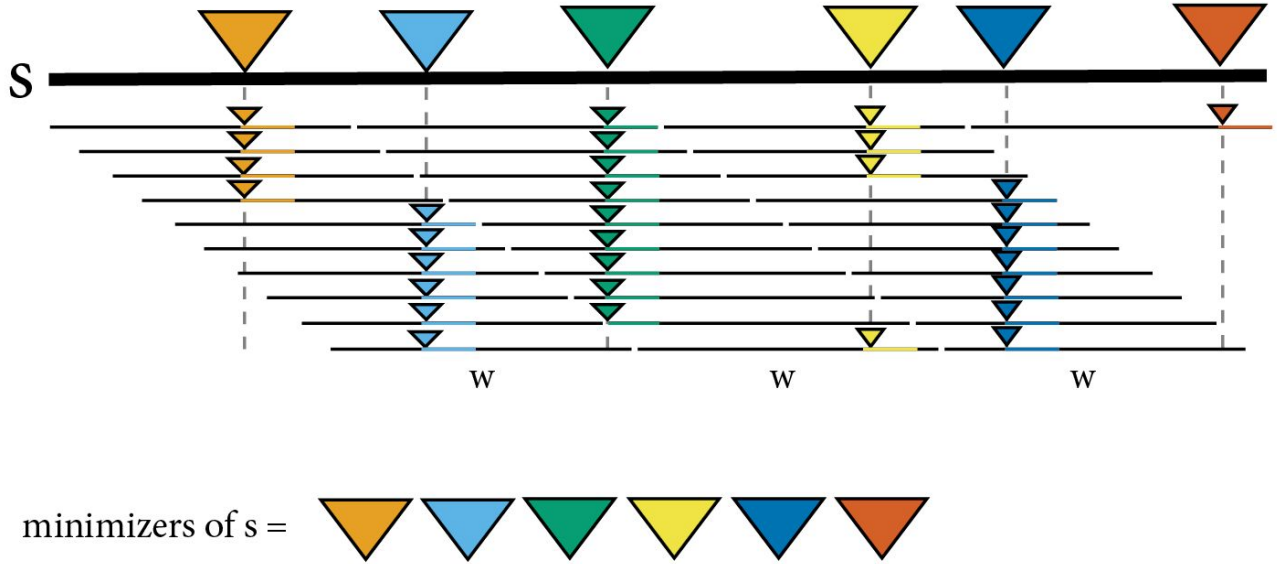
AGAGATCCGAGTAATTAATACTAAATAGAACAGTTAGGGCAGGAGGGTTTCAGGTCGATCTGAACTATCACTGAG

**Instead of searching for every seed (25-mer) using a BWT, the reference genomes are preprocessed to find unique sequences of a given size (minimizers), these are stored in a hash table and the long reads are scanned to find an initial mapping.**

**As before, consistent mapping are kept, and fast alignment are performed on the gaps, the alignments can be sped up using common k-mers in the read and reference.**

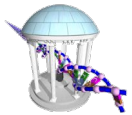
**Used by tools like *minimap2***

# Minimizers are Features of a Sequence




For a fixed window, w, a minimizer is the smallest k-mer in the window (based on any ordering method). For a sequence, *minimizers* from sliding windows yield a compact representation. A *minimizer* algorithm has three parameters: (*O*,*w*,*k*). *O* is the total order of the k-mers and determines how k-mers are ordered, *w* is the window length and *k* is the k-mer size. Thus, for every window, *w-k+1* k-mers are ordered and the smallest is selected. Sequences with length longer than *w* will yield an ordered *minimizer* set which includes contiguity information unlike unordered k-mer sets.

# Outputs of an Alignment



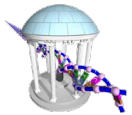
**First the Aligner itself returns a report. This is usually sent to "standard out"**

```
40162994 reads; of these:
  40162994 (100.00%) were paired; of these:
    10975116 (27.33%) aligned concordantly 0 times
    25558853 (63.64%) aligned concordantly exactly 1 time
    3629025 (9.04%) aligned concordantly >1 times
    ----
    10975116 pairs aligned concordantly 0 times; of these:
      7514833 (68.47%) aligned discordantly 1 time
    ----
    3460283 pairs aligned 0 times concordantly or discordantly; of these:
      6920566 mates make up the pairs; of these:
        2597685 (37.54%) aligned 0 times
        1517013 (21.92%) aligned exactly 1 time
        2805868 (40.54%) aligned >1 times
96.77% overall alignment rate
```

Consistent with expectations for paired-end mates.

1) Mates are on opposite strands
2) They are within the expected gap spacing (default is a 500bp "outer distance" -X)

# Outputs of an Alignment



**Another file format .SAM, and a binary/compressed version .BAM**

**The file beings with many lines of header data:**

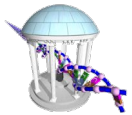
```
@HD     VN:1.0  SO:unsorted
@SQ     SN:chr1 LN:248956422
@SQ     SN:chr2 LN:242193529
@SQ     SN:chr3 LN:198295559
@SQ     SN:chr4 LN:190214555
@SQ     SN:chr5 LN:181538259
@SQ     SN:chr6 LN:170805979
@SQ     SN:chr7 LN:159345973
@SQ     SN:chr8 LN:145138636
@SQ     SN:chr9 LN:138394717
@SQ     SN:chr10        LN:133797422
@SQ     SN:chr11        LN:135086622
@SQ     SN:chr12        LN:133275309
@SQ     SN:chr13        LN:114364328
@SQ     SN:chr14        LN:107043718
@SQ     SN:chr15        LN:101991189
@SQ     SN:chr16        LN:90338345
@SQ     SN:chr17        LN:83257441
@SQ     SN:chr18        LN:80373285
@SQ     SN:chr19        LN:58617616
@SQ     SN:chr20        LN:64444167
@SQ     SN:chr21        LN:46709983
@SQ     SN:chr22        LN:50818468
```

First, comes information about the genome/contigs that the reads were aligned to.

In this example the reads were aligned to the human reference (GRCh38_NCBI). The lengths of each chromosome are provided.

There is also a copy of the command line that was used to invoke the aligner

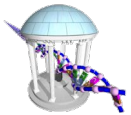
# More SAM



Unique name for each read or read pair — Mapping of 1st base — Alignment

```
ERR1344023.107  83      chr1    14178   2       151M    =       13860   -469
CCGTGTCTGTCACTGAAACCTTCTTTGTGGGAGACTATTCCTCCCATCTGCAACAGCTGCCCCTGCTGACGGCCCTCCTCTCCTCCCTCTCATCCCAGAGAAACACGTCAGCTGGGAGCTT
CTGCCCCCACTGCCTAGGGACCAACAGGGG  The aligned read's sequence
KF<,,<F<AKA7F7<A,7FFA<,A,7FKFKAKFAFAAA7FF7KKKAFKKKKKAKKFF<KKKKKKKKAAKFKKKKKAFKFKKKKKKKKKKKKKKKKKKA7KKKKKK7FKKKKKKKKKKKKKK
KKKKKKKKKKKKKKKKKKKKKKKKKKFFFAA AS:i:-18        XS:i:-18        XN:i:0  XM:i:4  XO:i:0  XG:i:0  NM:i:4  MD:Z:22T47T5T28G45
YS:i:-50        YT:Z:CP

ERR1344023.107  163     chr1    13860   2       151M    =       14178   469
NAATAACTAAAGTTAGCTGCCCTGGACTATTCACCCCCTAGTCTCAATTTAAAAAGATCGCCATGGGCACAGGGCCCCTGCCTGGGCGCTTGTCACCGCCGTCAGCTTCTTCCTGAGGCAT
TTCTGCGGCCATGCTCACTAGCCTGCTCCA
!,<AFFKKFAAKFKA<,A<7A(,<F77A,7AFFFAA<<,,AF,FFFFAFFKKKA,<F,<AAFAF,F,F,7AAK7A<A<AFF<FFF<,F,AAA,<,<7,((((,,,,777A7,,,A,,,,,,
,,,<<,,((,(<7,<<A,,,,,<,,(,,,, AS:i:-50        XS:i:-47        XN:i:0  XM:i:16 XO:i:0  XG:i:0  NM:i:16
MD:Z:0T51G6C6C19G10T2C0C2C12T4C4A3T5C3A5C3      YS:i:-18        YT:Z:CP

ERR1344023.108  83      chr15   101976816       1       151M    =       101976573       -394
GCCTAATGGCCCTTGGCACGATTGCTGGTGTTGGGGTAGAAGATGTCGTGGAGTTTGCTCAAGTGGTTGAGAGGGAGGGAGGTGCCATCGACTTGGAGGAACTGGCACCAAGCCAGGGAGA
TAGAAATCCAGGCAAGGCTGTGGGGCAGGT
A7<,,,A<<,,,AFFA7,<,7F<F,7KKFKFFAAAKFKKFFK<F<,A,AKF,F,AKF<F<<KKFKKKKKFKFF<KKKKFAFKF77FFAKKF77KKKKKFAKFFFFFF,FAKF<7FKKFFAF
,FKKFAKFFFFF,KFFAFF<<FF<FA<A<A AS:i:-6 XS:i:-6 XN:i:0  XM:i:2  XO:i:0  XG:i:0  NM:i:2  MD:Z:19A27T103  YS:i:-11
YT:Z:CP

ERR1344023.108  163     chr15   101976573       1       151M    =       101976816       394
TGCTTTTAATAAAGGCTCTCTAGCTGTGCAGGATGCAAACGACTCGGGGTCAGTGACTGCCTCCTGCCCCTGTTGGTCCCTAGGCAGTGGGGGCAGAAGCTCCCAGCTGACCTGTCTCTCT
GGGATGAGAGGGAGGAGAGAAGGGCAGCCA
<AA<,A,<F<AKKK<,FFAFK,<F,777A<,,F,7AKKKKK,77A,AAA(,AK7FKKF,,A,AF,7FFFKAFAF<F,AFF<FKKKKK,FFA<FAKFKKKK,AFKFFKFF<77,<A,,,,<,
A7<F,<AAFFFF<,,<7<AAFK<,,<,(77 AS:i:-11        XS:i:-11        XN:i:0  XM:i:4  XO:i:0  XG:i:0  NM:i:4  MD:Z:15A25T73T32T2
YS:i:-6 YT:Z:CP
```

# Even more SAM



Flags

```
ERR1344023.107 83    chr1    14178   2       151M    =       13860   -469
CCGTGTCTGTCACTGAAACCTTCTTTGTGGGAGACTATTCCTCCCATCTGCAACAGCTGCCCCTGCTGACGGCCCTCCTCTCCTCCCTCTCATCCCAGAGAAACACGTCAGCTGGGAGCTT
CTGCCCCCACTGCCTAGGGACCAACAGGGG
KF<,,<F<AKA7F7<A,7FFA<,A,7FKFKAKFAFAAA7FF7KKKAFKKKKKAKKFF<KKKKKKKKAAKFKKKKKAFKFKKKKKKKKKKKKKKKKKKA7KKKKKK7FKKKKKKKKKKKKKK
KKKKKKKKKKKKKKKKKKKKKKKKKKFFFAA AS:i:-18        XS:i:-18        XN:i:0  XM:i:4  XO:i:0  XG:i:0  NM:i:4  MD:Z:22T47T5T28G45
YS:i:-50        YT:Z:CP
```

Flags

```
ERR1344023.107 163   chr1    13860   2       151M    =       14178   469
NAATAACTAAAGTTAGCTGCCCTGGACTATTCACCCCCTAGTCTCAATTTAAAAAGATCGCCATGGGCACAGGGCCCCTGCCTGGGCGCTTGTCACCGCCGTCAGCTTCTTCCTGAGGCAT
TTCTGCGGCCATGCTCACTAGCCTGCTCCA
!,<AFFKKFAAKFKA<,A<7A(,<F77A,7AFFFAA<<,,AF,FFFFAFFKKKA,<F,<AAFAF,F,F,7AAK7A<A<AFF<FFF<,F,AAA,<,<7,((((,,,,777A7,,,A,,,,,,
,,,<<,,((,(<7,<<A,,,,,<,,(,,,, AS:i:-50        XS:i:-47        XN:i:0  XM:i:16 XO:i:0  XG:i:0  NM:i:16
MD:Z:0T51G6C6C19G10T2C0C2C12T4C4A3T5C3A5C3      YS:i:-18        YT:Z:CP
```

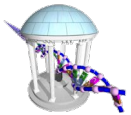
FLAG: Combination of bitwise FLAGs.[12] Each bit is explained in the following table:

| Bit | | Description |
|---|---|---|
| 1 | 0x1 | template having multiple segments in sequencing |
| 2 | 0x2 | each segment properly aligned according to the aligner |
| 4 | 0x4 | segment unmapped |
| 8 | 0x8 | next segment in the template unmapped |
| 16 | 0x10 | SEQ being reverse complemented |
| 32 | 0x20 | SEQ of the next segment in the template being reverse complemented |
| 64 | 0x40 | the first segment in the template |
| 128 | 0x80 | the last segment in the template |
| 256 | 0x100 | secondary alignment |
| 512 | 0x200 | not passing filters, such as platform/vendor quality controls |
| 1024 | 0x400 | PCR or optical duplicate |
| 2048 | 0x800 | supplementary alignment |

83: 64 + 16 + 2 + 1
- 64: This is the first read of a pair
- 16: It has been reverse completed to align to the ref
- 2: Mates both align to the same chromosome and are within the allowed gap spacing
- 1: Read is part of a mated-pair

163: 128 + 32 + 2 + 1
- 128: This is the last read of a pair
- 32: The mate of this read is reverse complemented
- 2: Mates both align to the same chromosome and are within the allowed gap spacing
- 1: Read is part of a mated-pair

# Even more and more SAM



Alignment CIGAR

```
ERR1344023.107  83      chr1    14178   2       151M    =       13860   -469
CCGTGTCTGTCACTGAAACCTTCTTTGTGGGAGACTATTCCTCCCATCTGCAACAGCTGCCCCTGCTGACGGCCCTCCTCTCCTCCCTCTCATCCCAGAGAAACACGTCAGCTGGGAGCTT
CTGCCCCCACTGCCTAGGGACCAACAGGGG
KF<,,<F<AKA7F7<A,7FFA<,A,7FKFKAKFAFAAA7FF7KKKAFKKKKKAKKFF<KKKKKKKKAAKFKKKKKAFKFKKKKKKKKKKKKKKKKKKKA7KKKKKK7FKKKKKKKKKKKKKK
KKKKKKKKKKKKKKKKKKKKKKKKKKFFFAA AS:i:-18        XS:i:-18        XN:i:0  XM:i:4  XO:i:0  XG:i:0  NM:i:4  MD:Z:22T47T5T28G45
YS:i:-50        YT:Z:CP
```

Alignment CIGAR

```
ERR1344023.107  163     chr1    13860   2       151M    =       14178   469
NAATAACTAAAGTTAGCTGCCCTGGACTATTCACCCCCTAGTCTCAATTTAAAAAGATCGCCATGGGCACAGGGCCCCTGCCTGGGCGCTTGTCACCGCCGTCAGCTTCTTCCTGAGGCAT
TTCTGCGGCCATGCTCACTAGCCTGCTCCA
!,<AFFKKFAAKFKA<,A<7A(,<F77A,7AFFFAA<<,,AF,FFFFAFFKKKA,<F,<AAFAF,F,F,7AAK7A<A<AFF<FFF<,F,AAA,<,<7,((((,,,,777A7,,,A,,,,,,
,,,<<,,((,(<7,<<A,,,,,<,,(,,,, AS:i:-50        XS:i:-47        XN:i:0  XM:i:16 XO:i:0  XG:i:0  NM:i:16
MD:Z:0T51G6C6C19G10T2C0C2C12T4C4A3T5C3A5C3      YS:i:-18        YT:Z:CP
```

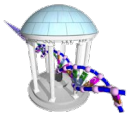
CIGAR: CIGAR string. The CIGAR operations are given in the following table (set '*' if unavailable):

| Op | BAM | Description | Consumes query | Consumes reference |
|---|---|---|---|---|
| M | 0 | alignment match (can be a sequence match or mismatch) | yes | yes |
| I | 1 | insertion to the reference | yes | no |
| D | 2 | deletion from the reference | no | yes |
| N | 3 | skipped region from the reference | no | yes |
| S | 4 | soft clipping (clipped sequences present in SEQ) | yes | no |
| H | 5 | hard clipping (clipped sequences NOT present in SEQ) | no | no |
| P | 6 | padding (silent deletion from padded reference) | no | no |
| = | 7 | sequence match | yes | yes |
| X | 8 | sequence mismatch | yes | yes |

A CIGAR string is an alignment in the purest sense. It describes how bases in this read are aligned to (ignoring whether they match or not) to the sequence starting at the given chromosome and position.

The ***NM:i flag*** indicates the edit distance, number of mismatches and indels incurred by the alignment

# PAF (pairwise alignment format)



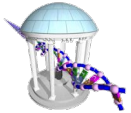
## Native format for minimap2

| **[read]** | | | | **[len]** | **[st]** | **[en]** | **[strand]** |
|---|---|---|---|---|---|---|---|
| 1b8a8aac-104f-4a0e-a06e-aae73c03a86f | | | | 7510 | 51 | 7509 | - |
| **[target]** | **[len]** | **[st]** | **[en]** | **[match]** | **[aln]** | **[score]** | |
| contig_16 | 67249 | 26520 | 34462 | 6597 | 8189 | 60 | |

**Flags:**

NM:i:1592 ms:i:6943 AS:i:6530 nn:i:0 tp:A:P cm:i:262
s1:i:2078 s2:i:748 de:f:0.1514 rl:i:15
cg:Z:**43M1D15M2D8M3I3M1D29M1D4**...

# PAF (pairwise alignment format)



## Native format for minimap2

```
1b8a8aac-104f-4a0e-a06e-aae73c03a86f    7510    51    7509    -    contig_16    67249    26520    34462    6597    8189    60
NM:i:1592    ms:i:6943    AS:i:6530    nn:i:0    tp:A:P    cm:i:262    s1:i:2078    s2:i:748    de:f:0.1514    rl:i:15
cg:Z:43M1D15M2D8M3I3M1D29M1D41M2D53M4D14M3I6M1D31M1D34M2D6M2D8M1D5M1D51M1D3M1D13M2D8M1D25M1D6M1I21M1I11M4D3M2D5M1D10M2D6M3D9M2D1M1D1
1M3D29M1I159M5D16M2D2M1I5M2D4M2I3M1D28M1I16M1I14M2D4M1D21M2D15M2D4M4D2M1I6M1D8M1D19M3D20M1D13M2D5M3I2M3D6M1I4M1I24M2D8M3I8M1D17M1D1M
1D3M1I7M1D17M1D5M1I9M1D5M9D2M2D13M3I4M2D1M4D21M1D8M1I3M1I1M3I3M3D11M3I11M3I35M1I9M2I4M2D10M3D39M1D13M3D2M2D14M1I4M1I4M2D57M1D13M1D8M
1D5M1I18M1D13M1D8M2I40M2D8M1D7M1I17M8D2M1D14M3D6M2D22M1D8M1D8M1I5M3D19M3D17M3D5M1D8M1D9M1D7M1D10M8D7M3D2M1D15M3I4M1D1M1D7M1I5M2I32M1
I36M1I58M1D34M2D10M1D17M1D3M3I6M1D2M1D7M5D2M1D7M1D5M1D4M1D3M1D5M1I4M1I3M5D2M1D34M1D37M2D1M3D6M2D5M2D6M1D57M1D19M1I32M1D10M2I45M1D12M
4D6M3D1M1D6M2D6M1D10M2D3M2D4M1I2M1I21M1D4M1I11M1D5M1D19M1D11M1I31M1D7M1D6M2D11M4I8M3D1M1D4M5D1M2D2M1I14M2D11M1D14M1D13M1D5M1D1M3D7M1
D9M1D22M2D26M1I21M1D11M2I2M1I28M2D34M3D7M1D24M1D3M1I4M1D7M1I18M2D10M3D3M1D15M1I6M1D91M1D10M1D3M2D11M2D4M1D21M4D10M2D2M2D20M1D2M1D10M
2D15M4D7M1D85M1I6M1I3M1I4M1D6M2I4M2D29M1I15M2D22M1I17M3D7M2D33M2I2M3D2M2D18M2D15M1I35M1D58M3I9M1D14M2D31M1D7M1I20M2I93M2D5M1D62M1D11
M5D2M1I15M1D6M1D8M5D26M1D9M3D17M1D25M1I1M3D11M1I2M3D11M3D4M3D7M1D27M3D1M2D13M2D5M1D5M1D18M5I9M1D13M1D9M3D8M1D10M3D7M2D20M1D5M3D4M1I1
7M2I3M2I5M1D12M1D3M1D1M1D20M3D4M1D18M1D5M2I5M1I12M2D1M3D10M2D18M1D20M2D14M1I1M1I1M3D2M1I27M2D7M2D13M2D6M2D27M1I4M3D19M3D23M2D2M2D2M1
D7M1D25M3I1M2I5M4D12M1I3M1D4M2I13M2D4M1I17M1D14M1I3M3D5M2D1M4D11M1D59M4D3M2D4M1D2M1D70M1D38M2D7M1D20M1D1M1D8M2I12M1D9M1D16M6I6M1D1M5
D14M2D9M1I46M1D22M4D15M1D4M2I20M2D11M1D45M1D7M1D15M2D5M2I17M3D13M2D6M1D49M1D1M2D4M4D7M3I5M3D6M2I20M1D11M1I3M1D5M2I3M1D8M1D11M2D19M1D
27M1D5M1D6M4D5M1D41M2D37M2D69M1D21M2D5M2D3M1I8M1D9M1D2M1D6M1D17M1D25M2D1M3D9M1D3M3D7M1D5M3D6M1D13M1D2M1D3M1D8M2I1M1I3M1I4M1I22M1D51M
1D38M2D12M1I36M2D24M2D5M2D39M1I5M3D2M1D5M3D11M1I34M2D11M1D40M1I10M3D2M1D9M1I20M2D2M1D9M1D10M1I7M2D5M1D10M1D9M1D9M1I32M1D7M2I2M1D25M1
D22M1I28M2D1M2D18M1D3M3D8M1D5M5D4M4I21M3D21M1D5M1I6M1D19M2D11M1I26M1I16M1D4M1I17M1D1M1D14M3D3M1D18M1D8M2I14M2D2M2D10M1I16M1D10M2D8M3
D8M2D6M1I11M6I3M4D6M1D7M1D3M2D2M1D26M1D9M1I5M1D38M2D2M1D6M1D3M2D11M1D11M2D12M1D1M2D15M3D1M2D6M1D6M1D10M1I6M1I2M1I4M1D9M2D8M1D19M1I7M
2D10M4D6M1D17M3I9M1D15M1I5M2D9M1D7M2D8M1D17M3D2M3D5M2D8M1I43M1I8M1D28M1D30M4D6M1I2M1I4M1D4M1I3M1D21M1I8M2I14M1D4M4D8M2I6M1I18M2D20M1
I20M1D11M2I3M2D8M1D12M3I2M2D15M1I12M2I13M1D20M1D4M2D5M2D5M1D13M1I4M3D23M4I12M1I18M2I5M2D8M1D29M1D11M3I2M1I4M2D8M3D1M1D3M2I5M3D2M2D7M
2D2M3I17M1D8M1I1M2D3M3D5M2D4M1D30M1I15M1D11M1D8M1I29M2I6M1D21M1I27M2I4M2D24M1I4M1D3M1D7M1D10M1D6M2I48M1D4M1D17M1I8M1I3M3I3M1I10M2I12
M
```

# Next Time

## Visualizing, Interpreting, and Analyzing Alignment outputs