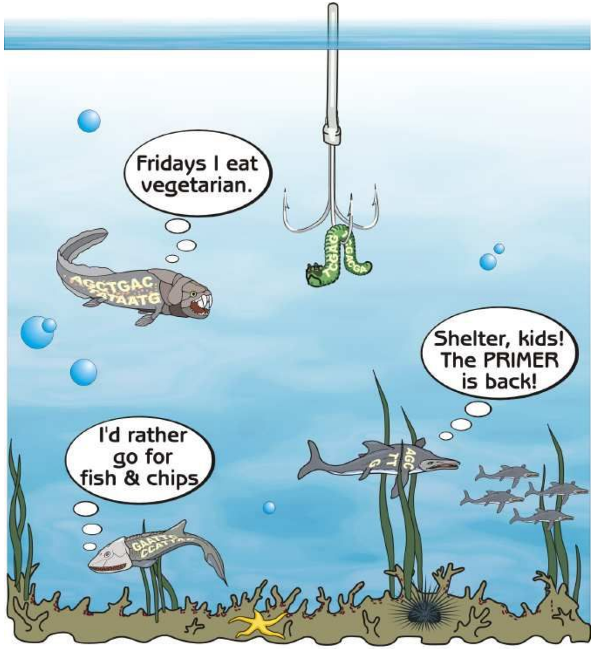
BCB 716 - Sequence Analysis


Fridays I eat vegetarian.
AGCTGAC
TATAATG
Shelter, kids! The PRIMER is back!
I'd rather go for fish & chips
GATTC
CCATTA

"The Art of Gene Fishing"

Final exam:

- Take home available to download from the website after midday on 12/1 (Friday)
- Read a paper
- ~5 Short answers that Integrate concepts from class

Reference Free DNA Sequence Analysis

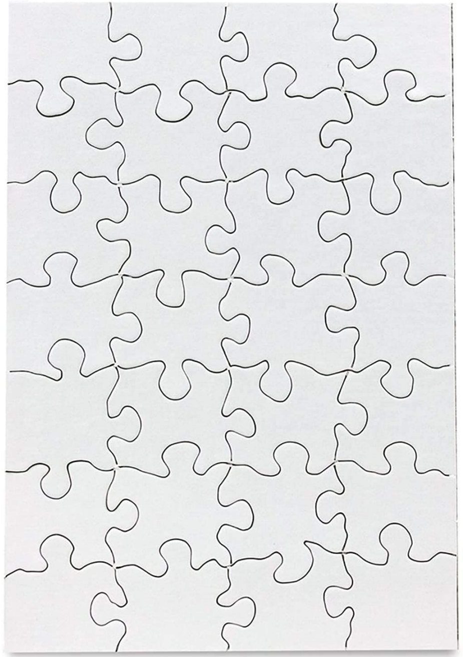
Reference Sequences Introduce bias

- **Relying on a reference sequence makes many assumptions**
  - All samples are assumed to be similarly organized
  - The "reference" allele is assumed to be common when sometimes it isn't
  - The reference is missing things
- **How do you overcome this problem?**
  - Consensus? Depends on population
  - Many references?

Novel idea

Rather than aligning reads to a reference, why not align them to each other. Not quite an assembly, but just enough to search for all and any sequence substrings.

Like assembling a jigsaw puzzle without an image of the assembled version. Or with no image at all!



Let's align every read with all other reads.

Rather than index a reference genome for searching, we build a BWT index of every read.

MSBWT

A BWT of a ***string collection*** instead of just a single string

- Earliest: Mantaci et al. (2005), used concatenation approach
- Bauer et al. (2011) - proposed version we will discuss today

Analogy:

- Instead of searching for a substring within a single book, search every book of a library
    - Each book has its own text, suffix array, and end-of-text delimiter
    - Searching allows us to find how many times a substring appears and in which texts

Bioinformatics?

- Search all genomes? You could, but that's not the main application.
- Search multiple chromosomes of an organism? You should, but even that is not the killer app

Naive Construction

- Create all rotations for all strings in the collection
- Sort all rotations together (Suffix Array)
- Store the predecessor of each suffix
- Strings are “cyclic”
- The predecessor is always from the same string
- Impossible to “jump” from one string to another
- Strings can have different lengths

```
String1
ACCA$
CCA$A
CA$AC
A$ACC
$ACCA

String2
CAAA$
AAA$C
AA$CA
A$CAA
$CAAA
```

(Unsorted suffixes)

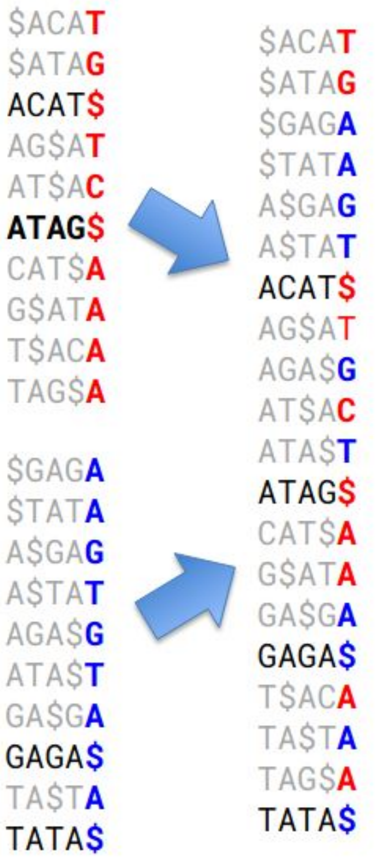
Sorted	MSBWT
$ACCA	A
$CAAA	A
A$ACC	C
A$CAA	A
AA$CA	A
AAA$C	C
ACCA$	$
CA$AC	C
CAAA$	$
CCA$A	A

(Merged and sorted)

(multi-string BWT
Note the 2 '$'s)

Merging msBWTs

- BETTER YET! Rather than inserting new strings, build a BWT of the new strings and merge the new and old BWTs
- Suffixes of BTWs are already sorted
- BTWs are interleaved
- In the worse case (ties) the entire suffix must be considered, but generally the longest common prefix of suffixes is smaller
- Minimal overhead
- Well suited for divide an conquer approaches (like merge sort)
- Easy to merge multiple data sets!
- Compression improves!



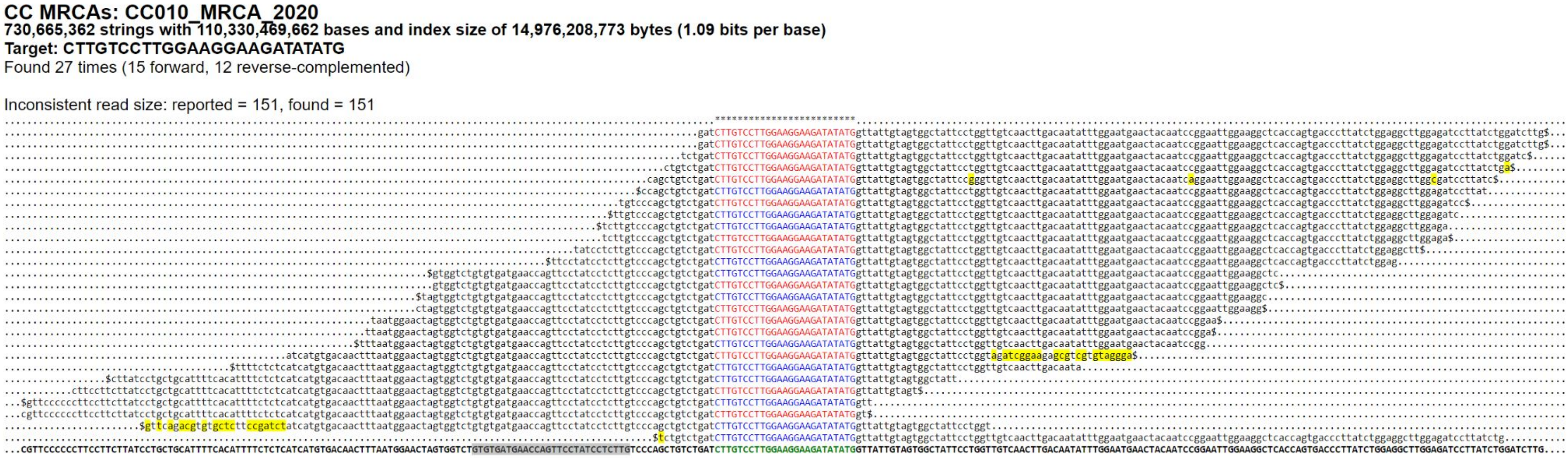
$ACAT
$ATAG
ACAT$
AG$AT
AT$AC
ATAG$
CAT$A
G$ATA
T$ACA
TAG$A

$GAGA
$TATA
A$GAG
A$TAT
AGA$G
ATA$T
GA$GA
GAGA$
TA$TA
TATA$

$ACAT
$ATAG
$GAGA
$TATA
A$GAG
A$TAT
ACAT$
AG$AT
AGA$G
AT$AC
ATA$T
ATAG$
CAT$A
G$ATA
GA$GA
GAGA$
T$ACA
TA$TA
TAG$A
TATA$

K-mer Search & Read Extraction

Basic Use:



CC MRCAs: CC010_MRCA_2020
730,665,362 strings with 110,330,469,662 bases and index size of 14,976,208,773 bytes (1.09 bits per base)
Target: CTTGTCCTTGGAAGGAAGATATATG
Found 27 times (15 forward, 12 reverse-complemented)

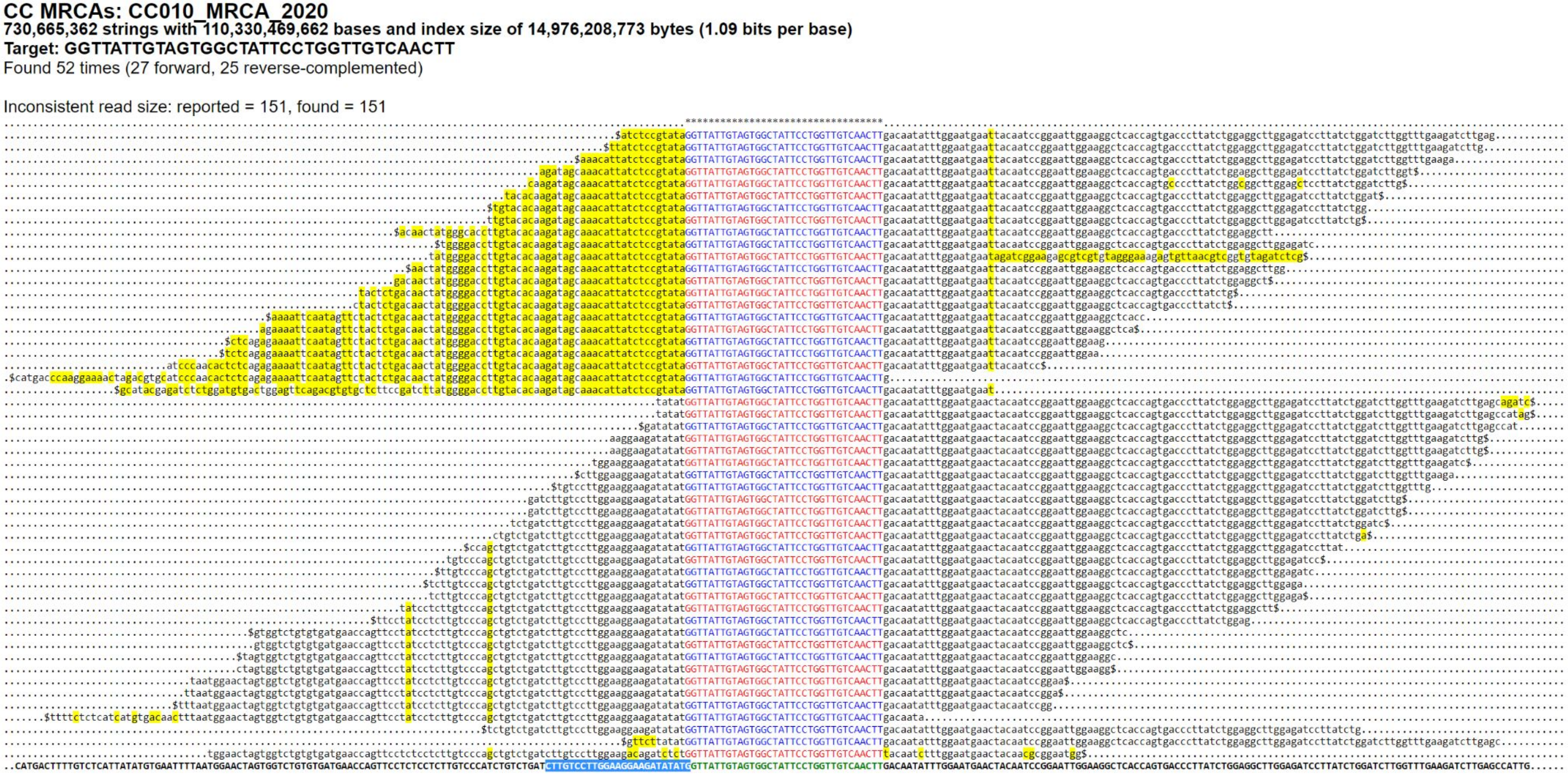
Inconsistent read size: reported = 151, found = 151

```
.............................................................................................................................*************************..............................................................................................................................
..........................................................................................................................gatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccggaattggaaggctcaccagtgacccttatctggaggcttggagatccttatctggatcttg$...
..........................................................................................................................gatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccggaattggaaggctcaccagtgacccttatctggaggcttggagatccttatctggatcttg$...
.......................................................................................................................tctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccggaattggaaggctcaccagtgacccttatctggaggcttggagatccttatctggatc$......
....................................................................................................................ctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccggaattggaaggctcaccagtgacccttatctggaggcttggagatccttatctga$.........
.................................................................................................................cagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattccgggttgtcaacttgacaatatttggaatgaactacaatcaggaattggaaggctcaccagtgacccttatctggaggcttggcgatccttatc$............
...............................................................................................................$ccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccggaattggaaggctcaccagtgacccttatctggaggcttggagatccttat..............
............................................................................................................tgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccggaattggaaggctcaccagtgacccttatctggaggcttggagatcc$.................
..........................................................................................................$ttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccggaattggaaggctcaccagtgacccttatctggaggcttggagatc...................
........................................................................................................$tcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccggaattggaaggctcaccagtgacccttatctggaggcttggaga.....................
.........................................................................................................tcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccggaattggaaggctcaccagtgacccttatctggaggcttggaga$....................
....................................................................................................tatcctcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccggaattggaaggctcaccagtgacccttatctggaggctt$.........................
...............................................................................................$ttcctatcctcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccggaattggaaggctcaccagtgacccttatctggag..............................
..........................................................................$gtggtctgtgtgatgaaccagttcctatcctcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccggaattggaaggctc...................................................
...........................................................................gtggtctgtgtgatgaaccagttcctatcctcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccggaattggaaggctc$..................................................
........................................................................$tagtggtctgtgtgatgaaccagttcctatcctcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccggaattggaaggc.....................................................
........................................................................ctagtggtctgtgtgatgaaccagttcctatcctcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccggaattggaagg$.....................................................
................................................................taatggaactagtggtctgtgtgatgaaccagttcctatcctcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccggaa$.............................................................
...............................................................ttaatggaactagtggtctgtgtgatgaaccagttcctatcctcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccgga$..............................................................
.............................................................$tttaatggaactagtggtctgtgtgatgaaccagttcctatcctcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccgg................................................................
.................................................atcatgtgacaactttaatggaactagtggtctgtgtgatgaaccagttcctatcctcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggtagatcggaagagcgtcgtgtaggga$............................................................................
.......................................$ttttctctcatcatgtgacaactttaatggaactagtggtctgtgtgatgaaccagttcctatcctcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaata......................................................................................
.................$cttatcctgctgcattttcacattttctctcatcatgtgacaactttaatggaactagtggtctgtgtgatgaaccagttcctatcctcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctatt............................................................................................................
...........cttccttcttatcctgctgcattttcacattttctctcatcatgtgacaactttaatggaactagtggtctgtgtgatgaaccagttcctatcctcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagt$..................................................................................................................
...$gttcccccttccttcttatcctgctgcattttcacattttctctcatcatgtgacaactttaatggaactagtggtctgtgtgatgaaccagttcctatcctcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgtt...........................................................................................................................
...cgttcccccttccttcttatcctgctgcattttcacattttctctcatcatgtgacaactttaatggaactagtggtctgtgtgatgaaccagttcctatcctcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgt$...........................................................................................................................
.......................$gttcagacgtgtgctcttccgatctatcatgtgacaactttaatggaactagtggtctgtgtgatgaaccagttcctatcctcttgtcccagctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggt......................................................................................................
..................................................................................................................$tctgtctgatCTTGTCCTTGGAAGGAAGATATATGgttattgtagtggctattcctggttgtcaacttgacaatatttggaatgaactacaatccggaattggaaggctcaccagtgacccttatctggaggcttggagatccttatctg...........
...CGTTCCCCCTTCCTTCTTATCCTGCTGCATTTTCACATTTTCTCTCATCATGTGACAACTTTAATGGAACTAGTGGTCTGTGTGATGAACCAGTTCCTATCCTCTTGTCCCAGCTGTCTGATCTTGTCCTTGGAAGGAAGATATATGGTTATTGTAGTGGCTATTCCTGGTTGTCAACTTGACAATATTTGGAATGAACTACAATCCGGAATTGGAAGGCTCACCAGTGACCCTTATCTGGAGGCTTGGAGATCCTTATCTGGATCTTG....
```

Green: query k-mer. Red: forward reads. Blue: reverse-complement reads. Yellow: sequencing errors

- Search for all reads with a given k-mer
- Extract all reads with that k-mer and its reverse-complement
- Build a consensus

Moving over a little bit



CC MRCAs: CC010_MRCA_2020

730,665,362 strings with 110,330,469,662 bases and index size of 14,976,208,773 bytes (1.09 bits per base)

Target: GGTTATTGTAGTGGCTATTCCTGGTTGTCAACTT

Found 52 times (27 forward, 25 reverse-complemented)

Inconsistent read size: reported = 151, found = 151

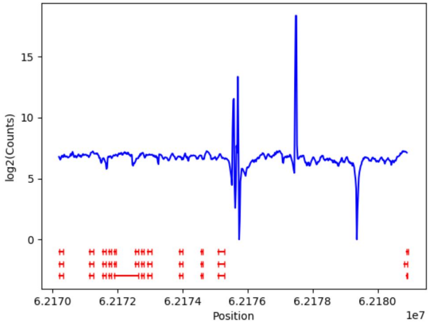
```
..CATGACTTTTGTCTCATTATATGTGAATTTTAATGGAACTAGTGGTCTGTGTGATGAACCAGTTCCTCTCCTCTTGTCCCATCTGTCTGATCTTGTCCTTGGAAGGAAGATATATGGTTATTGTAGTGGCTATTCCTGGTTGTCAACTTGACAATATTTGGAATGAACTACAATCCGGAATTGGAAGGCTCACCAGTGACCCTTATCTGGAGGCTTGGAGATCCTTATCTGGATCTTGGTTTGAAGATCTTGAGCCATTG.....
```

Moving over 24 bases shows a different story

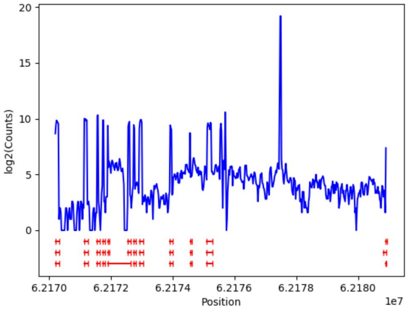
Reference-based Searches

- Rather than align reads to a reference search for reference k-mers in the read dataset.
- Given a reference genome and region
- Split reference into overlapping k-mers
- Count the abundance of each k-mer and plot
- Fast - O(k) time per k-mer
- With preprocessing (sorting k-mers in suffix order) it can be even faster approaching O(1) for many k-mers
- Similar to an alignment pileup

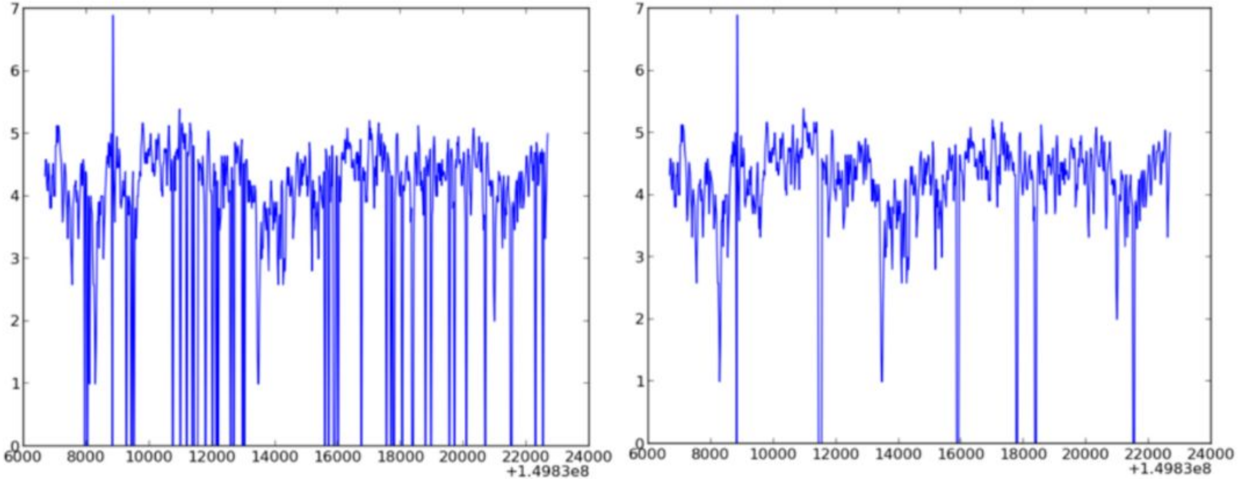
Gene: *Alad* (ENSMUSG00000028393)
CC010_MRCA_2020: 730,665,362 strings with 110,330,469,662 bases and index size of 14,976,208,773 bytes (1.09 bits per base)
Chromosome 4: 62,170,203 - 62,180,952



Gene: *Alad* (ENSMUSG00000028393)
CC010-M4424: 453,723,401 strings with 45,826,063,501 bases and index size of 3,964,874,160 bytes (0.69 bits per base)
Chromosome 4: 62,170,203 - 62,180,952



Iterative Reference Correction



Uncorrected

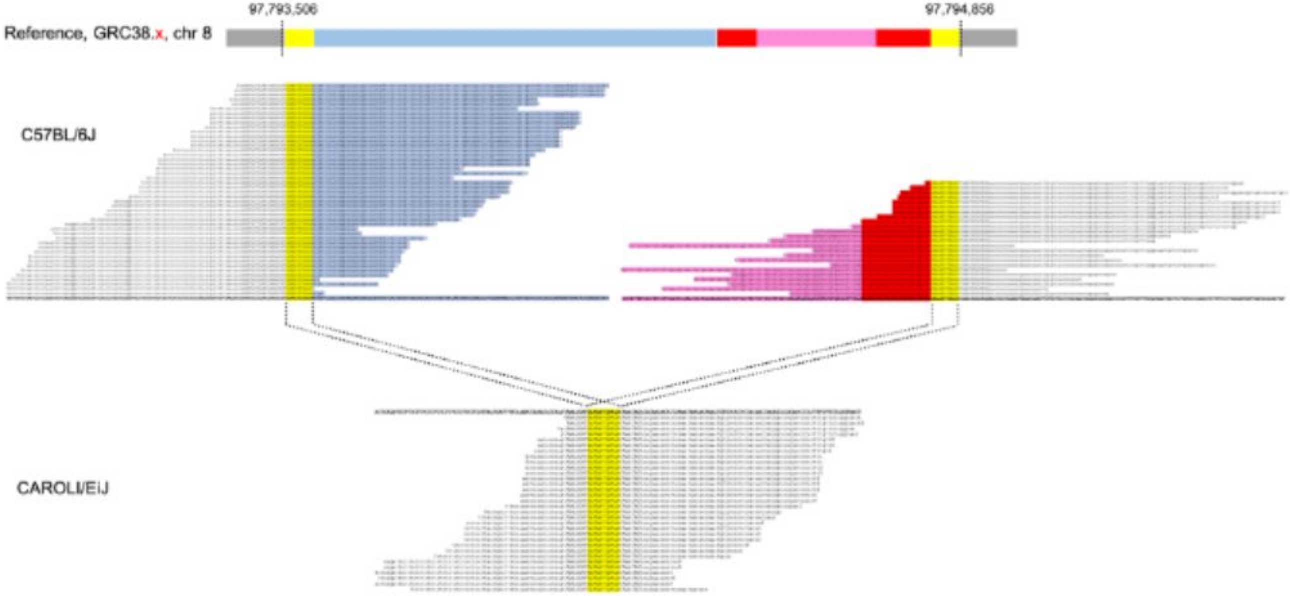
Corrected

```
149,838,013: 0   TTGATGGCTCGATGCATTCATTACCTGATCACTGCTCCCG
149,838,033: 0   TTACCTGATCACTGCTCCCGTTATGTAGGGAATGGGTACA
                                   ↓
149,838,013: 18  TTGATGGCTCGATGCATTCATTACTTGATCACTGCTCCCG
149,838,033: 17  TTACTTGATCACTGCTCCCGTTATGTAGGGAATGGGTACA
```

CAST/EiJ DNA-seq for annotated gene *Igf2*

Next Time

Reference-free RNAseq Analysis Approaches


97,793,506
97,794,856
Reference, GRC38.x, chr 8
C57BL/6J
CAROLI/EiJ