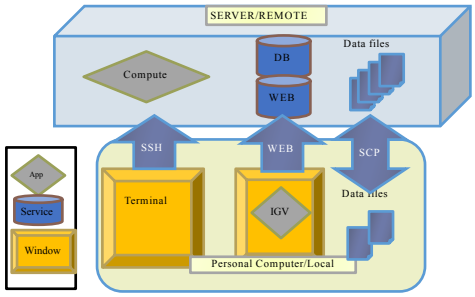
Applications and Servers



NGS Data and Sequence Alignment

Manpreet S. Katari

Aug 11, 2016

Outline

- NGS Data
  - FastA
  - FastQ
  - SAM
  - BAM
  - GFF
- Sequence Alignment
  - Global vs Local
  - Dynamic Programming
  - Burrow Wheeler's Algorithm.

Important files types

FASTA FASTQ	Sequence files
SAM BAM	Alignment files
GFF	Annotation files

Important file types: FASTA

A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The description line is distinguished from the sequence data by a greater-than (">") symbol in the first column. The word following the ">" symbol is the identifier of the sequence, and the rest of the line is the description (both are optional).

There should be no space between the ">" and the first letter of the identifier. It is recommended that all lines of text be shorter than 80 characters. The sequence ends if another line starting with a ">" appears; this indicates the start of another sequence.

Important file types: FASTA

```
>chrI
CCACACCACACCCACACACCCACACACCACACCACACACCACACCACACC
CACACACACACATCCTAACACTACCCTAACACAGCCCTAATCTAACCCTG
GCCAACCTGTCTCTCAACTTACCCTCCATTACCCTGCCTCCACTCGTTAC
CCTGTCCCATTCAACCATACCACTCCGAACCACCATCCACCCCTCTACTT
ACTACCACTCACCCACCGTTACCCTCCAATTACCCATATCCAACCCACTG
CCACTTACCCTACCATTACCCTACCATCCACCATGACCTACTCACCATAC
TGTTCTTCTACCCACCATATTGAAACGCTAACAAATGATCGTAAATAACA
CACACGTGCTTACCCTACCACTTTATACCACCACCACATGCCATACTCAC
CCTCACTTGTATACTGATTTACGTACGCACACACGGATGCTACAGTATATA
CCATCTCAAACTTACCCTACTCTCAGATTCCACTTCACTCCATGGCCCAT
CTCTCACTGAATCAGTACCAAATGCACTCACATCATTATGCACGGCACTT
GCCTCAGCGGTCTATACCCTGTGCCATTTACCCATAACGCCCATCATTAT
CCACATTTTGATATCTATATCTCATTCGGCGGTCCCAAATATTGTATAAC
TGCCCTTAATACATACGTTATACCACTTTTGCACCATATACTTACCACTC
CATTTATATACACTTATGTCAATATTACAGAAAAATCCCCACAAAAATCA
CCTAAACATAAAAATATTCTACTTTTCAACAATAATACATAAACATATTG
GCTTGTGGTAGCAACACTATCATGGTATCACTAACGTAAAAGTTCCTCAA
TATTGCAATTTGCTTGAACGGATGCTATTTCAGAATATTTCGTACTTACA
CAGGCCATACATTAGAATAATATGTCACATCACTGTCGTAACACTCTTTA
GCGT
```

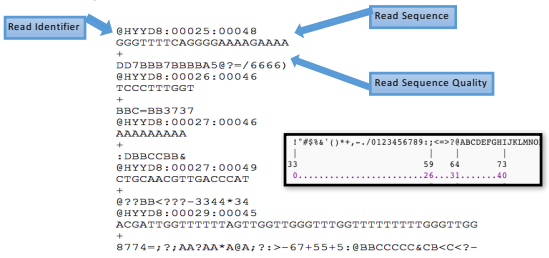
Important file types: FASTA

```
>seq0
FQTWEEFSRAAEKLYLADPMKVRVVLKYRHVDGNLCIKVTDDLVCLVYRTDQAQDVKKIEKF
>seq1
KYRTWEEFTRAAEKLYQADPMKVRVVLKYRHCDGNLCIKVTDDVVCLLYRTDQAQDVKKIEKFHSQLMKLME
LKVTDNKECLKFKTDQAQEAKKMEKLNNIFFTLM
>seq2
EEYQTWEEFARAAAEKLYLTDPMKVRVVLKYRHCDGNLCMKVTDDAVCLQYKTDQAQDVKKVEKLHGK
>seq3
MYQVWEEFSRAVEKLYLTDPMKVRVVLKYRHCDGNLCIKVTDNSVCLQYKTDQAQDVK
>seq4
EEFSRAVEKLYLTDPMKVRVVLKYRHCDGNLCIKVTDNSVVSYEMRLFGVQKDNFALEHSLL
```

Important file types: FASTQ

FASTQ format is a text-based format for storing both a biological sequence (usually nucleotide sequence) and its corresponding quality scores. Both the sequence letter and quality score are each encoded with a single ASCII character for brevity

Fastq format



FASTQ: Data Format

Sequence data format

- FASTQ
  - Text based
  - Encodes sequence calls and quality scores with ASCII characters
  - Stores minimal information about the sequence read
  - 4 lines per sequence
    - Line 1: begins with @; followed by sequence identifier and optional description
    - Line 2: the sequence
    - Line 3: begins with the "+" and is followed by sequence identifiers and description (both are optional)
    - Line 4: encoding of quality scores for the sequence in line 2
- References/Documentation
  - http://maq.sourceforge.net/fastq.shtml
  - Cock et al. (2009). Nuc Acids Res 38:1767-1771.

Quality scores

Phred Quality Score	Probability of incorrect base call	Base call accuracy
10	1 in 10	90 %
20	1 in 100	99 %
30	1 in 1000	99.9 %
40	1 in 10000	99.99 %
50	1 in 100000	99.999 %

Q = Phred Quality Scores
P = Base-calling error probabilities

Quality score encoding differ among the platforms

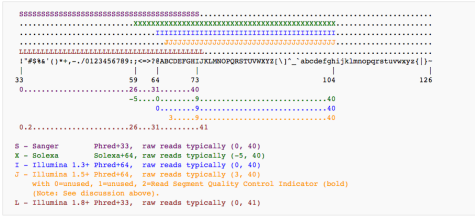
```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
```

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
33 59 64 73 104 126
```

Format/Platform	QualityScoreType	ASCII encoding
Sanger	Phred: 0-93	33-126
Solexa	Solexa:-5-62	64-126
Illumina 1.3	Phred: 0-62	64-126
Illumina 1.5	Phred: 0-62	64-126
Illumina 1.8	Phred: 0-62	33-126 *** Sanger format!

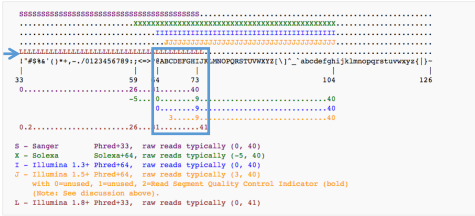
Most analysis tools require Sanger fastq quality score encoding

FASTQ Quality scores



http://en.wikipedia.org/wiki/FASTQ_format

FASTQ Quality scores



Capital Letters are good!

http://en.wikipedia.org/wiki/FASTQ_format

SAM (Sequence Alignment/Map)

Alignment data format

- SAM is the output of aligners that map reads to a reference genome
  - Tab delimited w/ header section and alignment section
    - Header sections begin with @ (are optional)
    - Alignment section has 11 mandatory fields
  - BAM is the binary format of SAM

http://samtools.sourceforge.net/

Mandatory Alignment Fields

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	`[!-?A-~]{1,255}`	Query template NAME
2	FLAG	Int	$[0,2^{16}-1]$	bitwise FLAG
3	RNAME	String	`\*|[!-()+-<>-~][!-~]*`	Reference sequence NAME
4	POS	Int	$[0,2^{29}-1]$	1-based leftmost mapping POSition
5	MAPQ	Int	$[0,2^{8}-1]$	MAPping Quality
6	CIGAR	String	`\*|([0-9]+[MIDNSHPX=])+`	CIGAR string
7	RNEXT	String	`\*|=|[!-()+-<>-~][!-~]*`	Ref. name of the mate/next fragment
8	PNEXT	Int	$[0,2^{29}-1]$	Position of the mate/next fragment
9	TLEN	Int	$[-2^{29}+1,2^{29}-1]$	observed Template LENgth
10	SEQ	String	`\*|[A-Za-z=.]+`	fragment SEQuence
11	QUAL	String	`[!-~]+`	ASCII of Phred-scaled base QUALity+33

http://samtools.sourceforge.net/SAM1.pdf

Bitwise Flag

	Bit	Description
1	0x1	template having multiple fragments in sequencing
2	0x2	each fragment properly aligned according to the aligner
4	0x4	fragment unmapped
8	0x8	next fragment in the template unmapped
16	0x10	SEQ being reverse complemented
32	0x20	SEQ of the next fragment in the template being reversed
64	0x40	the first fragment in the template
128	0x80	the last fragment in the template
256	0x100	secondary alignment
512	0x200	not passing quality controls
1024	0x400	PCR or optical duplicate

0x20= 16^1*2+16^0*0 = 32

What is 77? Find greatest value without going over

77-64 = 13 40
13- 8 = 5 8
5- 4 = 1 4
1- 1 = 0 1

What is 141?

CIGAR string

Op	BAM	Description
M	0	alignment match (can be a sequence match or mismatch)
I	1	insertion to the reference
D	2	deletion from the reference
N	3	skipped region from the reference
S	4	soft clipping (clipped sequences present in SEQ)
H	5	hard clipping (clipped sequences NOT present in SEQ)
P	6	padding (silent deletion from padded reference)
=	7	sequence match
X	8	sequence mismatch

29M 1D 9M 1D 9M 2D 21M 2D 18M 1D 70M

SAM format

SAM format gets converted to a BAM file

Annotation Formats

- Mostly tab delimited files that describe the location of genome features (i.e., genes, etc.)
- Also used for displaying annotations on standard genome browsers
- Important for associating alignments with specific genome features
- Descriptions

GFF3 Format

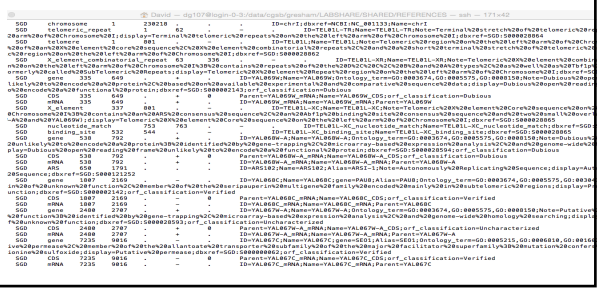
GFF3 format is a flat tab-delimited file. The first line of the file is a comment that identifies the file format and version. This is followed by a series of data lines, each one of which corresponds to an annotation.Here is a miniature GFF3 file:

```
##gff-version 3
ctg123  .  exon  1300  1500  .  +  .  ID=exon00001
ctg123  .  exon  1050  1500  .  +  .  ID=exon00002
ctg123  .  exon  3000  3902  .  +  .  ID=exon00003
ctg123  .  exon  5000  5500  .  +  .  ID=exon00004
ctg123  .  exon  7000  9000  .  +  .  ID=exon00005
```

The `##gff-version 3` line is required and *must* be the first line of the file. It introduces the annotation section of the file.

Columns:
Seqid, Source, Type, Start, End, Score, Strand, Phase, Attribute(Identifier)

GFF format



Global and local approaches to aligning sequences

GLOBAL: Attempt to "match" and assess similarity between two entire sequences

LOCAL: Find subsequences of high similarity

… and then possibly "stick" (chain, net, thread) together local alignments to obtain an overall comparison of the original sequences.

The second approach is more meaningful

(especially for long sequences, of different lengths, like whole genomes)

Two protein or DNA sequences are unlikely to present a straightforward overall "match", even if they are closely related.

Why? Substitutions are not the only process by which they diverge: insertions, deletions and rearrangements

Dynamic programming: Pairwise sequence alignment

Create a matrix table for comparing sequences with one sequence along each axis (size m+1, n+1)

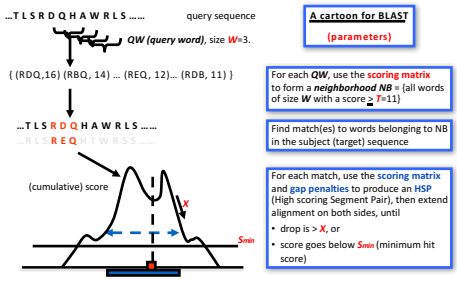
Fill in partial alignment scores until score for entire sequence has been calculated:

- **Assign score for each position *i,j*** , progressing from top left to bottom right
- **Score rule**: take maximum of the three choices
  1. Take value from left, assign gap penalty (gap along left axis)
  2. Take value from top, assign gap penalty (gap along top axis)
  3. Take value from diagonal above left, assign match/mismatch score
- **Repeat** until table is completed.

Use **trace-back** to obtain full alignment:

```
AC--TCG
ACAGTAG
```

BLAST: heuristic database search using local alignment
Basic Local Alignment Search Tool



A cartoon for BLAST (parameters)

For each ***QW***, use the **scoring matrix** to form a ***neighborhood NB*** = {all words of size ***W*** with a score ≥ *T*=11}

Find match(es) to words belonging to NB in the subject (target) sequence

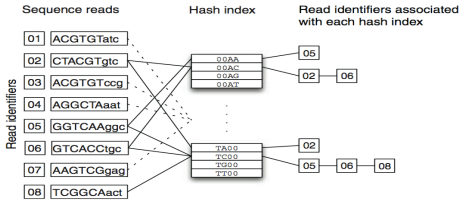
For each match, use the **scoring matrix** and **gap penalties** to produce an **HSP** (High scoring Segment Pair), then extend alignment on both sides, until

- drop is > *X*, or
- score goes below *Smin* (minimum hit score)

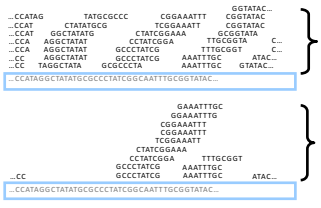
About Blat (from genome.ucsc.edu)

- "BLAT on DNA is designed to quickly find sequences of 95% and greater similarity of length 25 bases or more. "
- "It may miss more divergent or shorter sequence alignments. It will find perfect sequence matches of 20 bases. "
- "BLAT is not BLAST."
- "DNA BLAT works by keeping an index of the entire genome in memory. The index consists of all overlapping 11-mers stepping by 5 except for those heavily involved in repeats."
- "The index takes up about 2 gigabytes of RAM. The genome itself is not kept in memory, allowing BLAT to deliver high performance on a reasonably priced Linux box. "
- "The index is used to find areas of probable homology, which are then loaded into memory for a detailed alignment."

Hash Table (BLAT)



Short Read Applications



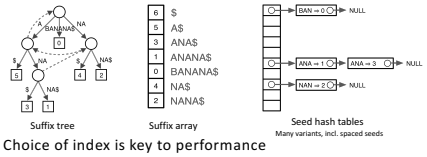
Finding the alignments is typically the performance bottleneck

New alignment algorithms must address the requirements and characteristics of NGS reads

- Millions of reads per run (30x of genome coverage)
- Short Reads (as short as 36bp)
- Different types of reads (single-end, paired-end, mate-pair, etc.)
- Base-calling quality factors
- Sequencing errors (~ 1%)
- Repetitive regions
- Sequencing organism vs. reference genome
- Must adjust to evolving sequencing technologies and data formats

Indexing

- Genomes and reads are too large for direct approaches like dynamic programming
- *Indexing* is required



Suffix tree | Suffix array | Seed hash tables (Many variants, incl. spaced seeds)

- Choice of index is key to performance

Suffix Array

Find "ctat" in the reference

[00]	gggtaaagctataactattgatcaggcgtt	[04]	aaagctataactattgatcaggcgtt
[01]	ggtaaagctataactattgatcaggcgtt	[12]	aactattgatcaggcgtt
[02]	gtaaagctataactattgatcaggcgtt	[05]	aagctataactattgatcaggcgtt
[03]	taaagctataactattgatcaggcgtt	[13]	actattgatcaggcgtt
[04]	aaagctataactattgatcaggcgtt	[06]	agctataactattgatcaggcgtt
[05]	aagctataactattgatcaggcgtt	[23]	aggcgtt
[06]	agctataactattgatcaggcgtt	[10]	ataactattgatcaggcgtt
[07]	gctataactattgatcaggcgtt	[20]	atcaggcgtt
[08]	ctataactattgatcaggcgtt	[16]	attgatcaggcgtt
[09]	tataactattgatcaggcgtt	[22]	caggcgtt
[10]	ataactattgatcaggcgtt	[26]	cgtt
[11]	taactattgatcaggcgtt	[08]	ctataactattgatcaggcgtt
[12]	aactattgatcaggcgtt	[14]	ctattgatcaggcgtt
[13]	actattgatcaggcgtt	[19]	gatcaggcgtt
[14]	ctattgatcaggcgtt	[25]	gcgtt
[15]	tattgatcaggcgtt	[07]	gctataactattgatcaggcgtt
[16]	attgatcaggcgtt	[24]	ggcgtt
[17]	tgatcaggcgtt	[00]	gggtaaagctataactattgatcaggcgtt
[18]	tgatcaggcgtt	[01]	ggtaaagctataactattgatcaggcgtt
[19]	gatcaggcgtt	[02]	gtaaagctataactattgatcaggcgtt
[20]	atcaggcgtt	[27]	gtt
[21]	tcaggcgtt	[29]	t
[22]	caggcgtt	[03]	taaagctataactattgatcaggcgtt
[23]	aggcgtt	[11]	taactattgatcaggcgtt
[24]	ggcgtt	[09]	tataactattgatcaggcgtt
[25]	gcgtt	[15]	tattgatcaggcgtt
[26]	cgtt	[21]	tcaggcgtt
[27]	gtt	[18]	tgatcaggcgtt
[28]	tt	[28]	tt
[29]	t	[17]	ttgatcaggcgtt

NGS Read Alignment
Burrows Wheeler Transformation (BWT)

- Invented by David Wheeler in 1983 (Bell Labs). Published in 1994.
  "A Block Sorting Lossless Data Compression Algorithm"
  *Systems Research Center Technical Report No **124**. Palo Alto, CA: Digital Equipment Corporation,* Burrows M, Wheeler DJ. 1994
- Originally developed for compressing large files (*bzip2*, etc.)
- Lossless, Fully Reversible
- Alignment Tools based on BWT: *bowtie*, BWA, *SOAP2*, etc.
- Approach:
  - Align reads on the *transformed* reference genome, using an efficient index (FM index)
  - Solve the simple problem first (align one character) and then build on that solution to solve a slightly harder problem (two characters) etc.
- Results in great speed and efficiency gains (a few GigaByte of RAM for the entire H. Genome). Other approaches require tens of GigaBytes of memory and are much slower.

Burrows Wheeler Transformation

Text = c t g a a a c t g g t $

1 2 3 4 5 6 7 8 9 10 11 0

➢ Introduce $ at the end and construct all cyclic permutations of Text

c	t	g	a	a	a	c	t	g	g	t	$
t	g	a	a	a	c	t	g	g	t	$	c
g	a	a	a	c	t	g	g	t	$	c	t
a	a	a	c	t	g	g	t	$	c	t	g
a	a	c	t	g	g	t	$	c	t	g	a
a	c	t	g	g	t	$	c	t	g	a	a
c	t	g	g	t	$	c	t	g	a	a	a
t	g	g	t	$	c	t	g	a	a	a	c
g	g	t	$	c	t	g	a	a	a	c	t
g	t	$	c	t	g	a	a	a	c	t	g
t	$	c	t	g	a	a	a	c	t	g	g
$	c	t	g	a	a	a	c	t	g	g	t

Burrows Wheeler Transformation

- Sort rows alphabetically, keeping of which row went where

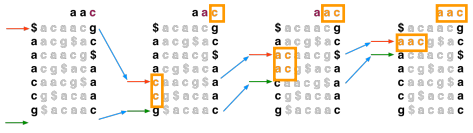
$	c	t	g	a	a	a	c	t	g	g	t
a	a	a	c	t	g	g	t	$	c	t	g
a	a	c	t	g	g	t	$	c	t	g	a
a	c	t	g	g	t	$	c	t	g	a	a
c	t	g	a	a	a	c	t	g	g	t	$
c	t	g	g	t	$	c	t	g	a	a	a
g	a	a	a	c	t	g	g	t	$	c	t
g	g	t	$	c	t	g	a	a	a	c	t
g	t	$	c	t	g	a	a	a	c	t	g
t	$	c	t	g	a	a	a	c	t	g	g
t	g	a	a	a	c	t	g	g	t	$	c
t	g	g	t	$	c	t	g	a	a	a	c

Burrows Wheeler Matrix

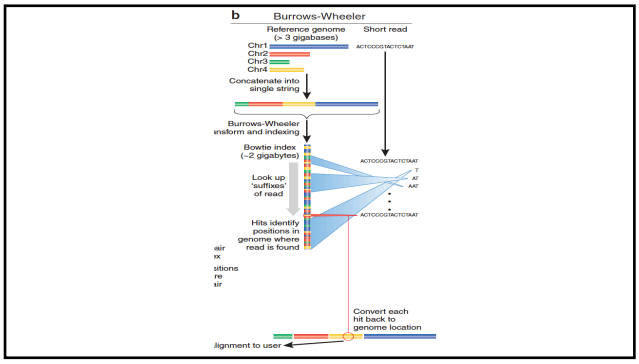
BWT (Text) = t g a a $ a t t g g c c

11 3 4 5 0 6 2 8 9 10 1 7

Exact Matching with FM Index



- In progressive rounds, **top** & **bot** delimit the range of rows beginning with progressively longer suffixes of Q


b
Burrows-Wheeler
Reference genome
(~ 3 gigabases)
Short read
Chr1
Chr2
Chr3
Chr4
ACTCCCGTACTCTAAT
Concatenate into
single string
Burrows-Wheeler
ansform and indexing
Bowtie index
(~2 gigabytes)
ACTCCCGTACTCTAAT
T
AT
AAT
Look up
'suffixes'
of read
ACTCCCGTACTCTAAT
Hits identify
positions in
genome where
read is found
eir
ix
sitions
re
sir
Convert each
hit back to
genome location
lignment to user