

# Low level Processing and Visualization of Genome Sequencing Data

## GSND 5340Q, BMDA

W. Evan Johnson, Ph.D.  
Professor, Division of Infectious Disease  
Director, Center for Data Science  
Rutgers University – New Jersey Medical School

2024-06-03

## Section 1

Motivating Example: X-linked Disease

# Rare X-linked Disease



# Rare X-linked Disease



**Uncle #1**



**Uncle #2**

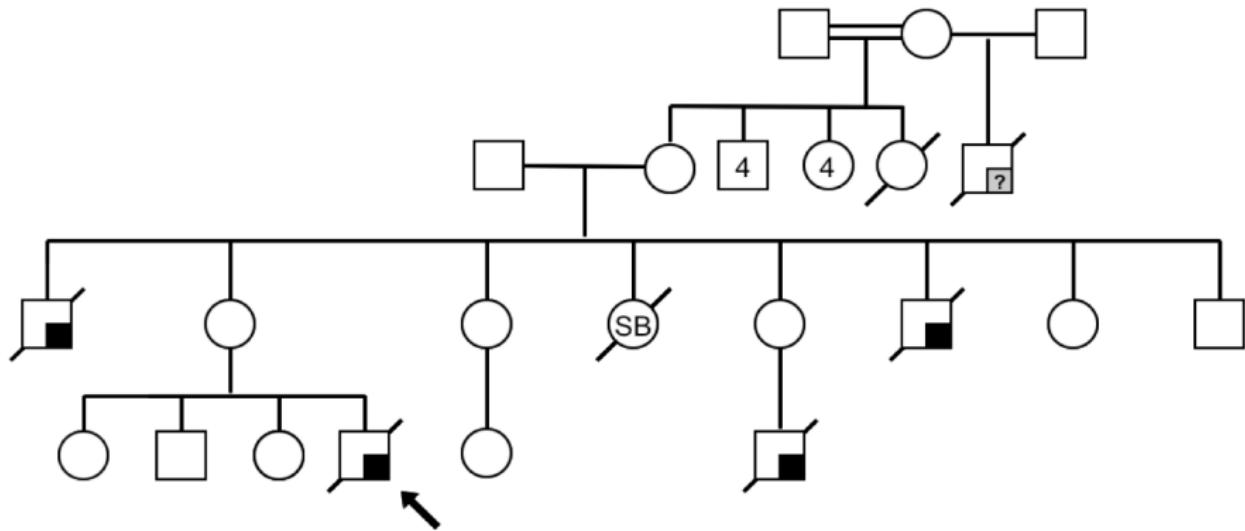


**cousin**

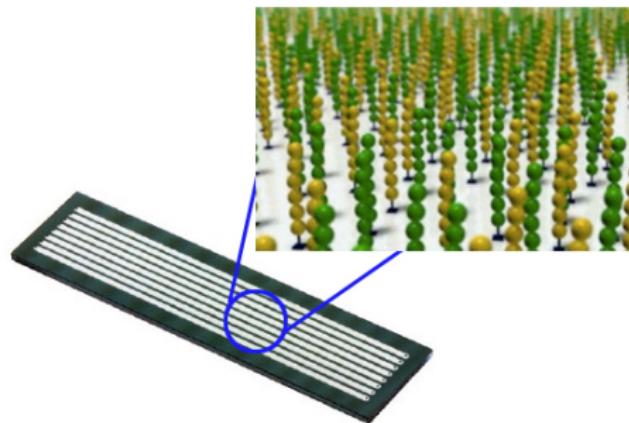


**proband**

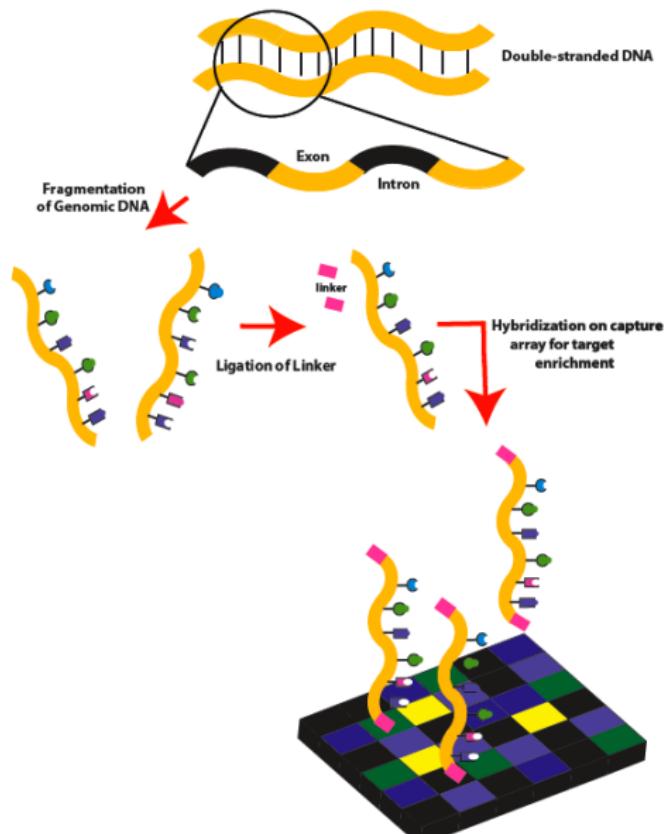
# Rare X-linked Disease



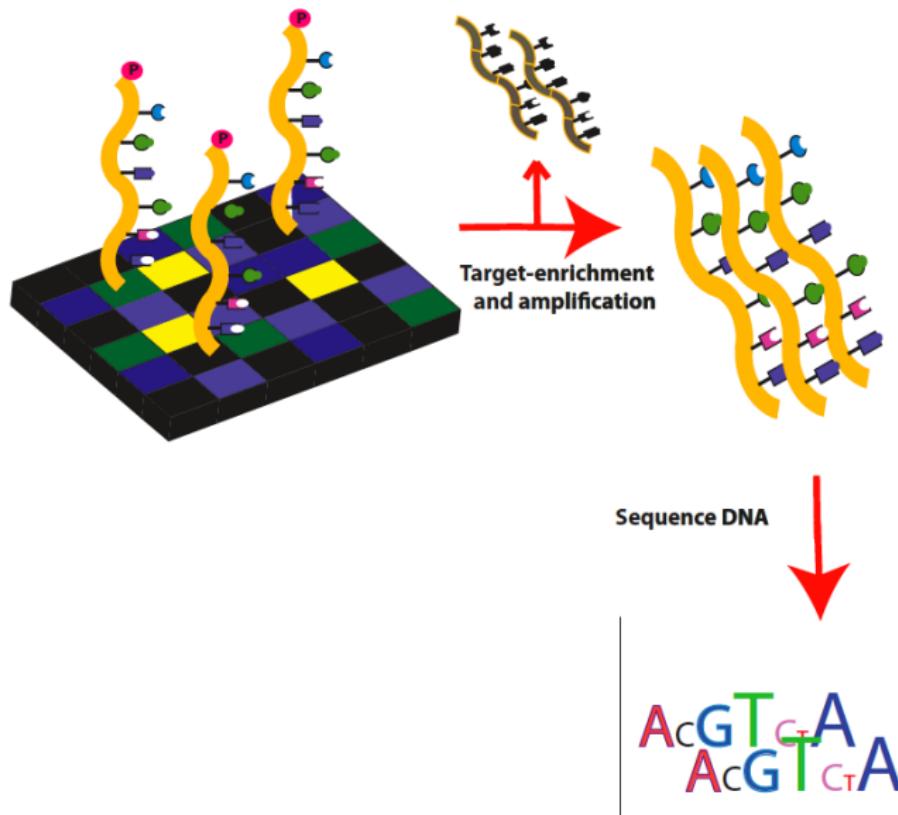
# Exon Capture Sequencing



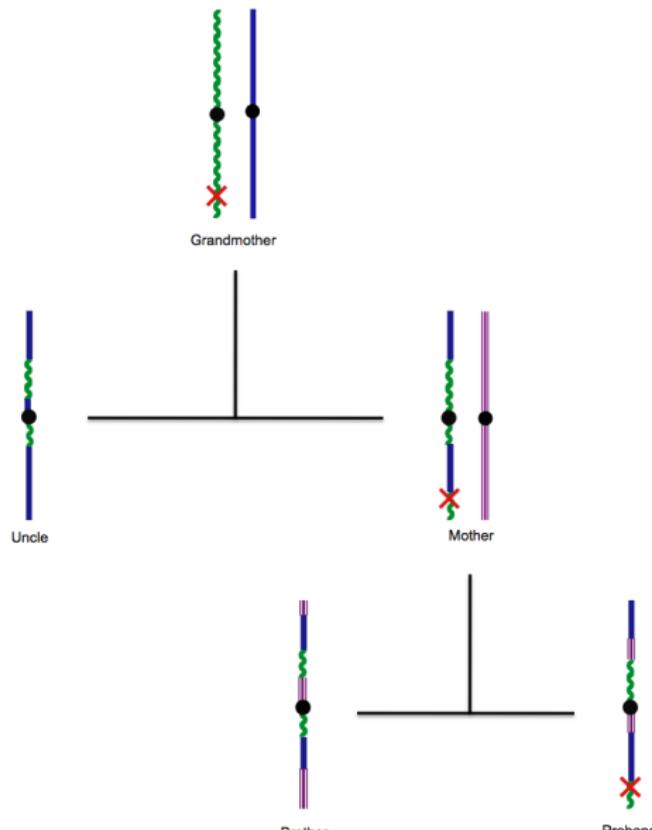
# Exome Capture



# Exome Capture



# Rare X-linked Disease



# Rare X-linked Disease

On X-Chromosome:



2,168 SNPs  
in proband

1,495 polymorphic  
in mom

1,300 polymorphic  
in grandma

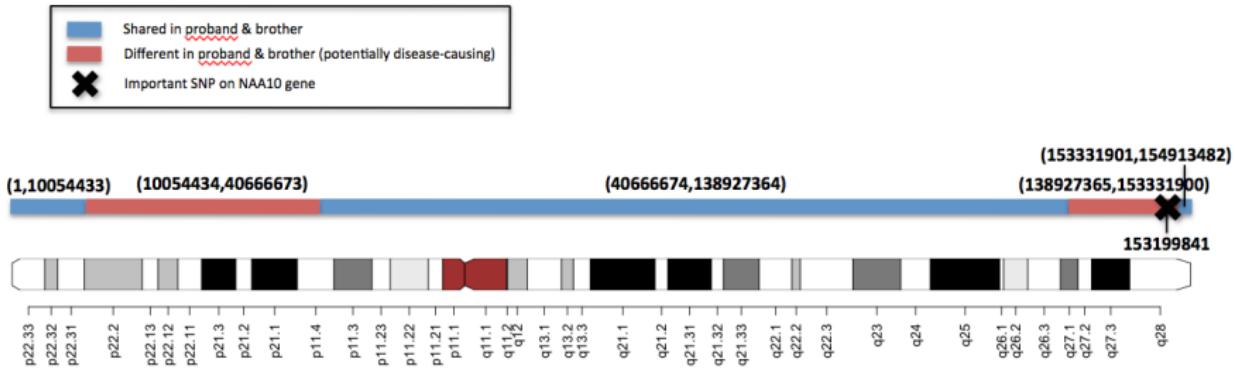
174 NOT  
in brother

92 NOT  
in uncle

44 in or  
near exons

2 protein coding, not  
common, etc. (VAAST)

# Rare X-linked Disease



# Rare X-linked Disease

AJHG  Supports open access

Submit Log in Register Subscribe Claim

ARTICLE | VOLUME 89, ISSUE 1, P28-43, JULY 15, 2011 [Download Full Issue](#)

PDF [603 KB] Figures Save Share Reprints Request

## Using VAAST to Identify an X-Linked Disorder Resulting in Lethality in Male Infants Due to N-Terminal Acetyltransferase Deficiency

Alan F. Rope • Kai Wang <sup>19</sup> • Rune Enevirth • ... Mark Yandell • Thomas Arnesen • Gholson J. Lyon    Show all authors • Show footnotes

Open Archive • Published: June 23, 2011 • DOI: <https://doi.org/10.1016/j.ajhg.2011.05.017>

 PlumX Metrics

### Introduction

We have identified two families with a previously undescribed lethal X-linked disorder of infancy; the disorder comprises a distinct combination of an aged appearance, craniofacial anomalies, hypotonia, global developmental delays, cryptorchidism, and cardiac arrhythmias. Using *X* chromosome exon sequencing and a recently developed probabilistic algorithm aimed at discovering disease-causing variants, we identified in one family a c.109T>C (*p.Ser37Pro*) variant in *NAA10*, a gene encoding the catalytic subunit of the major human N-terminal acetyltransferase (NAT). A parallel effort on a second unrelated family converged on the same variant. The absence of this variant in controls, the amino acid conservation of this region of the protein, the predicted disruptive change, and the co-occurrence in two unrelated families with the same rare

### Subjects and Methods

### Results

### Discussion

### Acknowledgments

### Supplemental Data

### Web Resources

### References

### Article info



Ad served by Google

[Ad options](#)

[Send feedback](#)

[Why this ad? ▾](#)

# N-terminal acetyltransferase (NAA10)

N-terminal acetyltransferase (NAT):

- Common modification (~80-90% of human proteins)
- Depletion from cancer cells linked to cell cycle arrest and apoptosis (Starheim, *BMC Proc* 2009)
- NAT genes directly implicated as cause of genetic disease
- Mutation demonstrated a significantly impaired biochemical activity *in vitro*
- NAA10 lethal if knocked out of Drosophila

## Section 2

# Generating Sequencing Data

# Next-Generation Sequencing

- Expensive to purchase (hundreds of thousands \$USD)
- Expensive to operate (e.g. reagents, flow cells)
- You can sequence your genome at 30X depth for <\$1000 USD.

Roche 454



Ion Torrent



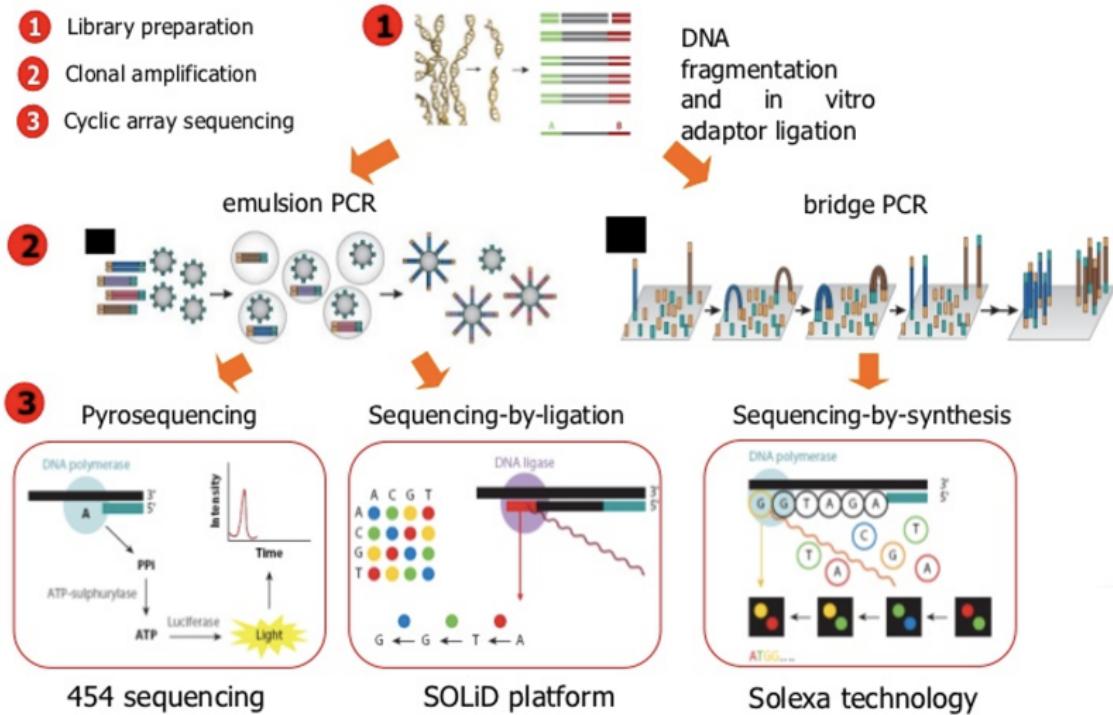
Illumina HiSeq 2500



Illumina NovaSeq 6000



# Next-Generation Sequencing



Alex Sanchez, Statistics and Bioinformatics Research Group, Statistics Department, Universitat de Barcelona

# Illumina Sequencing

- Most common sequencing technology today
- Sequences any DNA
- Sequencing by synthesis method
- Sequences (reads) are short (<300bp)
- 2 gigabases - 6 terabases per run
- Hours to days to complete one run

For more information, you can watch the following:

<https://www.youtube.com/watch?v=fCd6B5HRaZ8>

# Illumina Sequencing

- Sequencing occurs on a **flow cell**
- Each flow cell has 1 to 8 **lanes**
- Number of reads for overall flow cell varies
- Length of reads is fixed (e.g. 250 bp)
- Read format:
  - **Single end** - one read per molecule
  - **Paired end** - two reads per molecule
- **Multiplexing:** sequence many samples at once using molecular barcodes



NovaSeq Flowcell  
Courtesy of Illumina, Inc.

# Sequencing Library Generation Workflow

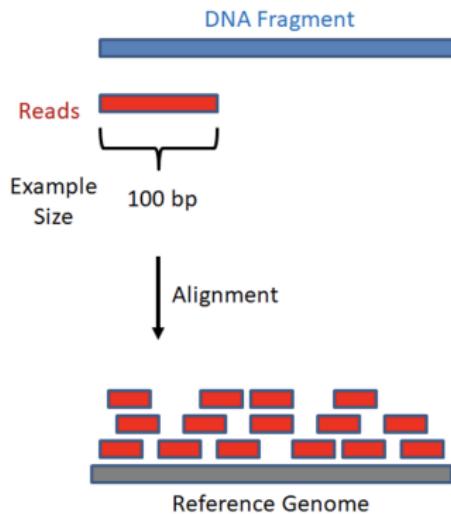
Sequencing Library: DNA prepared for sequencing

Workflow:

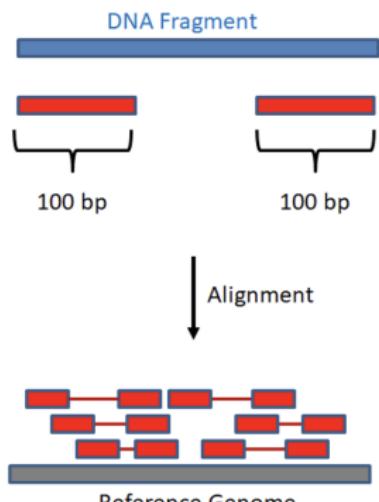
- Extract RNA/DNA from sample
  - If RNA, reverse transcribe to cDNA
- Size select using gel cut or random shearing
- PCR amplify DNA if concentration is low
- Add sequencing adapters
  - If multiplexing, use barcoded adapters
- Pool samples, load across flow lanes for sequencing
- Typically only perform 1, sequencing cores

# Design Choice: Single End vs Paired End

## Single End



## Paired End



# Design Choice: Number of Reads

**Table 1: Coverage and Read Recommendations by Application**

Category	Detection or Application	Recommended Coverage (x) or Reads (millions)	References
Whole genome sequencing	Homozygous SNVs	15x	Bentley et al., 2008
	Heterozygous SNVs	33x	Bentley et al., 2008
	INDELS	60x	Feng et al., 2014
	Genotype calls	35x	Ajay et al., 2011
	CNV	1-8x	Xie et al., 2009; Medvedev et al., 2010
Whole exome sequencing	Homozygous SNVs	100x (3x local depth)	Clark et al., 2011; Meynert et al., 2013
	Heterozygous SNVs	100x (13x local depth)	Clark et al., 2011; Meynert et al., 2013
	INDELS	not recommended	Feng et al., 2014
Transcriptome Sequencing	Differential expression profiling	10-25M	Liu Y. et al., 2014; ENCODE 2011 RNA-Seq
	Alternative splicing	50-100M	Liu Y. et al., 2013; ENCODE 2011 RNA-Seq
	Allele specific expression	50-100M	Liu Y. et al., 2013; ENCODE 2011 RNA-Seq
	De novo assembly	>100M	Liu Y. et al., 2013; ENCODE 2011 RNA-Seq
DNA Target-Based Sequencing	ChIP-Seq	10-14M (sharp peaks); 20-40M (broad marks)	Rozowsky et al., 2009; ENCODE 2011 Genome; Landt et al., 2012

<https://genohub.com/recommended-sequencing-coverage-by-application/>

# Design Choice: Sequencing Depth

## Whole Exome

- Less expensive
- Nearly complete ascertainment of variation in the coding ~1% of the genome (i.e. exons)
- Will miss functional variants outside of the coding region

## Low Coverage Whole Genome

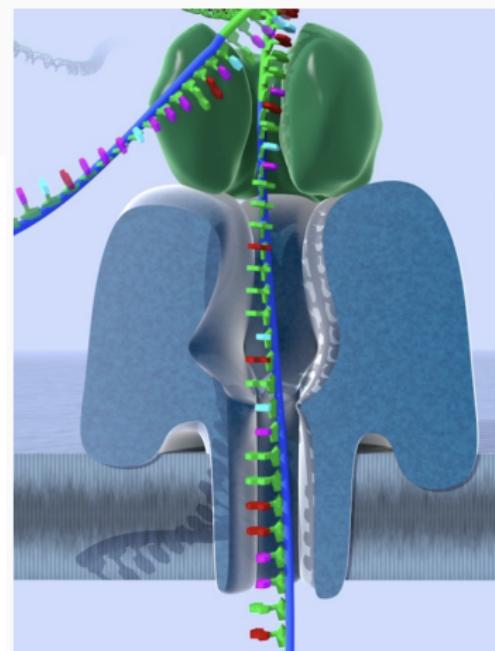
- Less expensive
- Reasonably good ascertainment of shared variation, but not unique variation

## Deep Whole Genome

- More expensive
- Capture most of the genetic information
- Sequence the entire genome of each subject

## 3rd Gen Sequencing: Oxford Nanopore

- “Real time single molecule”



<https://www.youtube.com/watch?v=GUb1TZvMWsw>

## Section 3

# Sequencing Data Formats

# Biological Data Formats are standardized

- Many different types of data
- Standard formats exist for many of them
- Always use/extend standard formats
- Don't save data in non standard formats when standards are available -  
No .xlsx!
- For complete list of formats see:  
<https://genome.ucsc.edu/FAQ/FAQformat.html>

# Sequencing Data Formats

format	data	tool(s)
FASTA	sequence of nucleotides	Samtools, Biopython
FASTQ	sequenced reads	FASTQC, Biopython
SAM/BAM/CRAM	aligned reads	Samtools, Deeptools, PySam
VCF	variant calls	vcftools bedtools
BED / BED-PE	genomic regions	bedtools
GFF	general features	Biopython, Manual Parsing
GTF	gene features	Biopython, Manual Parsing

# Sequencing Data Formats

format	data	tool(s)
FASTA	sequence of nucleotides	samtools faidx
FASTQ	sequenced reads	-
SAM/BAM/CRAM	aligned reads	samtools
VCF	variant calls	vcftools bedtools
BED / BED-PE		bedtools
GFF		-
GTF	gene features	-

Contains sequences  
of nucleotides / AA /  
bases

# Sequencing Data Formats

format	data	(e)
FASTA	sequences	fastq
FASTQ	sequences	fastq
SAM/BAM/CRAM	aligned reads	samtools
VCF	variant calls	vcftools bedtools
BED / BED-PE	genomic regions	bedtools
GFF	general features	-
GTF	gene features	-

## Data Formats Example

### EXAMPLE

Align reads to the reference, sort and index, call SNPs, extract the SNPs on chromosome Y overlapping with all the Alu elements.



## Data Formats Example

### EXAMPLE

Align reads to the reference, sort and index, call SNPs on chromosome Y overlapping with all the Alu elements.

FASTQ format

## Data Formats Example

### EXAMPLE

Align reads to the reference, sort and index, call SNPs on chromosome Y overlapping with all the Alu elements.

FASTA format

## Data Formats Example

### EXAMPLE

Align reads to the reference, sort and index, call SNPs, extract the SNPs on chromosome 1 save in BAM format with all the Alu elements.

## Data Formats Example

### EXAMPLE

Align reads to the reference, sort and index, call SNPs, extract the SNPs on chromosome 1 with all the Alu elements.

VCF format

## Data Formats Example

### EXAMPLE

Align reads to the reference, sort and index, call SNPs, extract the SNPs on chromosome Y overlapping with all the Alu elements.

Read from GTF/GFF/BED

## Data Formats Example

### EXAMPLE

Align reads to the reference, sort and index, call SNPs, extract the SNPs on chromosome Y overlapping with all the Alu elements.

## Data Formats Example

### EXAMPLE

Align reads to the reference, sort and index, call SNPs, extract the SNPs on chromosome 1, and compare with all the Alu elements.

## Data Formats Example

### EXAMPLE

Align reads to the reference, sort and index, call SNPs, extract the SNPs on chromosome Y over Alu elements.

samtools / GATK

## Data Formats Example

### EXAMPLE

Align reads to the reference, sort and index, call SNPs, extract the SNPs on chromosome Y overlapping with all the Alu elements.

samtools view

## Data Formats Example

### EXAMPLE

Align reads to the reference, sort and index, call SNPs, extract the SNPs on chromosome Y **overlapping** with all the Alu elements.

`bedtools intersect`

## Section 4

### Raw Sequencing Data and QC

# History and Evolution of Illumina Data Output

Illumina sequencers have given output in many different formats:

- Illumina .PRB and .INT files
  - Better access to raw data.
  - Base calling algorithms (Bravo and Irizarry, *Biometrics*, 2010)
  - Mapping algorithms (GNUMAP, NOVO)
  - Confusing formats; Large data files
- Illumina .FASTQ files
- Sanger .FASTQ files

# Illumina .INT and .PRB

1	1	125	771	1651.8	2189.6	228.1	549.9	219.0	202.5	48.4	3016.8	127.8	6.1	204
1	1	478	16	1050.0	969.9	149.5	311.7	0.0	0.0	39.3	134.1	0.0	0.0	0.0
1	1	780	553	639.6	980.8	555.2	6412.8	1040.1	4408.7	750.3	638.1	946.2	4351.1	7
1	1	123	685	-116.5	341.5	-14.0	-985.9	231.5	1090.0	88.3	-102.2	240.2	513.6	
1	1	61	934	40.7	87.3	38.5	21.3	16.4	31.7	100.9	68.8	41.4	29.4	40.4
1	1	866	972	2820.5	4698.9	435.8	8502.2	4740.3	4890.2	1491.5	1241.7	2137.5	2505.6	6
40	-40	-40	-40	-40	-40	-40	40	-40	-40	40	-40	-40	40	-40
28	-28	-40	-40	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5
-17	-16	-30	13	-40	26	-26	-40	-21	17	-20	-27	21	-21	-40
-40	40	-40	-40	-40	40	-40	-40	-1	1	-40	-40	-0	-40	-0
-1	-40	-0	-10	-20	-18	16	-35	-1	-8	-2	-13	12	-34	-13
-27	-40	-40	27	40	-40	-40	-40	-40	40	-40	-40	40	-40	-4

## .fasta or .fa

```
>chrX
ttgaactcctgacctcaggtatccgcggcctgacccaaaggcgct
cctgcctcagcctcccgagtagctggactacaggtgccaccatgc
....
caggctaattttgtattttagtagagacgggtttcaccatgttagc
caggatggtctcaatctcctgacctcatgatccgcctgcctcgccccc
>chrY
tgtacacttaaatgggtgaatttatggaatgtgaattataCGTGTG
CTTGTAAAAAAAAATGATGGAGATGGAGACGTGACTCTAGCGTGAAGGGG
...
GTGGGGAGAGTAGATCTAGAGTGGAGACACCACTTTAGGAGGTATGATC
cctgccaccatgcctggctaattttgtattttagtagagacagggtt
```

## .fastq or .fq (or .fq.gz)

```
@HWI-EAS240_0001:2:1:1142:17571#0/1
CTCTCTTTCTCCCCANGTCTCCTCATGACCATATCCNTGTTGTCCATTGTGTANGGNNNCTT
+HWI-EAS240_0001:2:1:1142:17571#0/1
bababb`abbbbabbYB^[^`_`_bbbbbbbbbb`_B`_``^`^bbba^`^^B00BBBBN00
@HWI-EAS240_0001:2:1:1142:6453#0/1
CAGGACGTGCACTATGCATCCACGATGCAAGTCTTANCATTATTAGGATAACANATANNGGC
+HWI-EAS240_0001:2:1:1142:6453#0/1
_U] [aa_aa^`^`^`[B] [] []]]S``Y`a`aaZ]B]]^`^`^`[O V Y]XBNNNBB]]]
@HWI-EAS240_0001:2:1:1142:19443#0/1
TCAGAAAACAGAAAGGTCTTTCTTACTTCTTGANGATGCCACCCCTCCAGANCAGNNAAT
+HWI-EAS240_0001:2:1:1142:19443#0/1
bbbabb^`^`^`bbabb]bb]^`]]]bbbbbbbbbYYBYX00000bb`bbVY^HB000BB0J0
```

# .FASTQ Paired End

fastq\_SRR1997469\_1.fastq

```
@SRR1997469.1 1 length=36
CACTTCTTAGAAATATCCACTTCGGAATAAAAGA
+SRR1997469.1 1 length=36
BBBBBFFFFF<FFFFFFFFFFFFFBFFFFFFFFF
@SRR1997469.2 2 length=36
ACAGTTAACGATCCTTACAGANAGNAGNCTNGTA
+SRR1997469.2 2 length=36
<BBBBFFFBBF<BFF/<FFFFFF##########
...
```

fastq\_SRR1997469\_2.fastq

```
@SRR1997469.1 1 length=36
AGATAAGATGGTAATCTTGATGGAGAACATTAAGA
+SRR1997469.1 1 length=36
BBBBBFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SRR1997469.2 2 length=36
ACTGGGAANCCTCTGTCTAGCCTTATATGAAAAAA
+SRR1997469.2 2 length=36
BBB/BFFB#BB/FFFF/FFFFFFBBFFFFFFFBF
...
```

## .FASTQ Comparison

[https://en.wikipedia.org/wiki/Phred\\_quality\\_score](https://en.wikipedia.org/wiki/Phred_quality_score)

# Sequence Quality Score (PHRED)

Phred quality scores are logarithmically linked to error probabilities

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 10,000	99.99%
50	1 in 100,000	99.999%
60	1 in 1,000,000	99.9999%

[https://en.wikipedia.org/wiki/Phred\\_quality\\_score](https://en.wikipedia.org/wiki/Phred_quality_score)

# Sequence Quality Score (PHRED)

## Quality scores (Phred)

- Sanger Phred: Range=(0,40),  $P = 1 - 10^{-(ASCII-33)/10}$
- Solexa: Range= (-5,40),  $P = \frac{10^{(ASCII-64)/10}}{1+10^{(ASCII-64)/10}}$
- Illumina 1.3: (0,40),  $P = \frac{10^{(ASCII-64)/10}}{1+10^{(ASCII-64)/10}}$
- Illumina 1.5: Range=(2,40),  $P = 1 - 10^{-(ASCII-64)/10}$
- Illumina 1.8: Same as Sanger except Range=(0,41)

# .FASTQ Comparison

> Sanger	> Solexa	> Illumina1.3	> Illumina1.5
ASCII Quality	ASCII Quality	ASCII Quality	ASCII Quality
! 33 0.0000	; 59 0.2403	@ 64 0.5000	B 66 0.3690
" 34 0.2057	< 60 0.2847	A 65 0.5573	C 67 0.4988
# 35 0.3690	= 61 0.3339	B 66 0.6131	D 68 0.6019
\$ 36 0.4988	> 62 0.3869	C 67 0.6661	E 69 0.6838
% 37 0.6019	? 63 0.4427	D 68 0.7153	F 70 0.7488
& 38 0.6838	@ 64 0.5000	E 69 0.7597	G 71 0.8005
' 39 0.7488	A 65 0.5573	F 70 0.7992	H 72 0.8415
( 40 0.8005	B 66 0.6131	G 71 0.8337	I 73 0.8741
) 41 0.8415	C 67 0.6661	H 72 0.8632	J 74 0.9000
* 42 0.8741	D 68 0.7153	I 73 0.8882	K 75 0.9206
+ 43 0.9000	E 69 0.7597	J 74 0.9091	L 76 0.9369
,	F 70 0.7992	K 75 0.9264	M 77 0.9499
,	G 71 0.8337	L 76 0.9406	N 78 0.9602
- 45 0.9369	H 72 0.8632	M 77 0.9523	O 79 0.9684
.	I 73 0.8882	N 78 0.9617	P 80 0.9749
.	J 74 0.9091	O 79 0.9693	Q 81 0.9800
/ 47 0.9602	K 75 0.9264	P 80 0.9755	R 82 0.9842
0 48 0.9684	L 76 0.9406	Q 81 0.9804	S 83 0.9874
1 49 0.9749	M 77 0.9523	R 82 0.9844	T 84 0.9900
2 50 0.9800	N 78 0.9617	S 83 0.9876	U 85 0.9921

# Quality Control

Need to preprocess the reads:

- Check for quality (FASTQC)
- Trim adapter and (Cutadapt, others)
- Remove duplicate reads, trim low complexity/quality bases/reads (Prinseq)
- Complete pipelines: NCBI Toolkit, QC-Chain, PathoQC ([pathoscope.sourceforge.net](http://pathoscope.sourceforge.net)), others

**Note:** Not comprehensive or updated!

# Read Quality Checks

**Step 1:** User inputs sequencing reads (fastq) and primer and adapter sequences

Sequencing Reads  
(fastq)

Primers & Adapters  
CTCGGCATTCTGT...  
GATCTATTATACTCC...

**Step 2: FastQC**

- Detect Phred offset
- Search for sequence tags

**Steps 3 & 4: Process sequencing reads**

Cutadapt

- Trim sequence primers and adapters

Prinseq

- Trim low quality bases
- Remove short sequences low complexity and redundant reads

**Step 5: Output quality controlled sequencing reads**

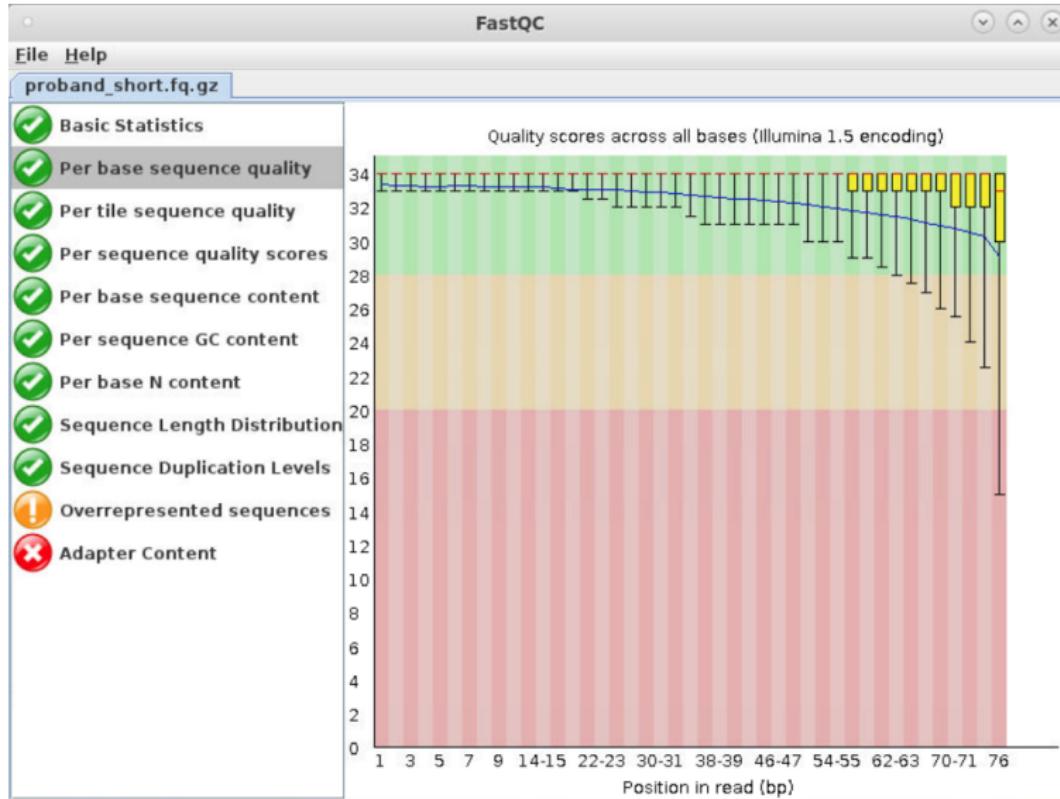
Clean Sequencing Reads

# Read Quality Checks (Outdated!)

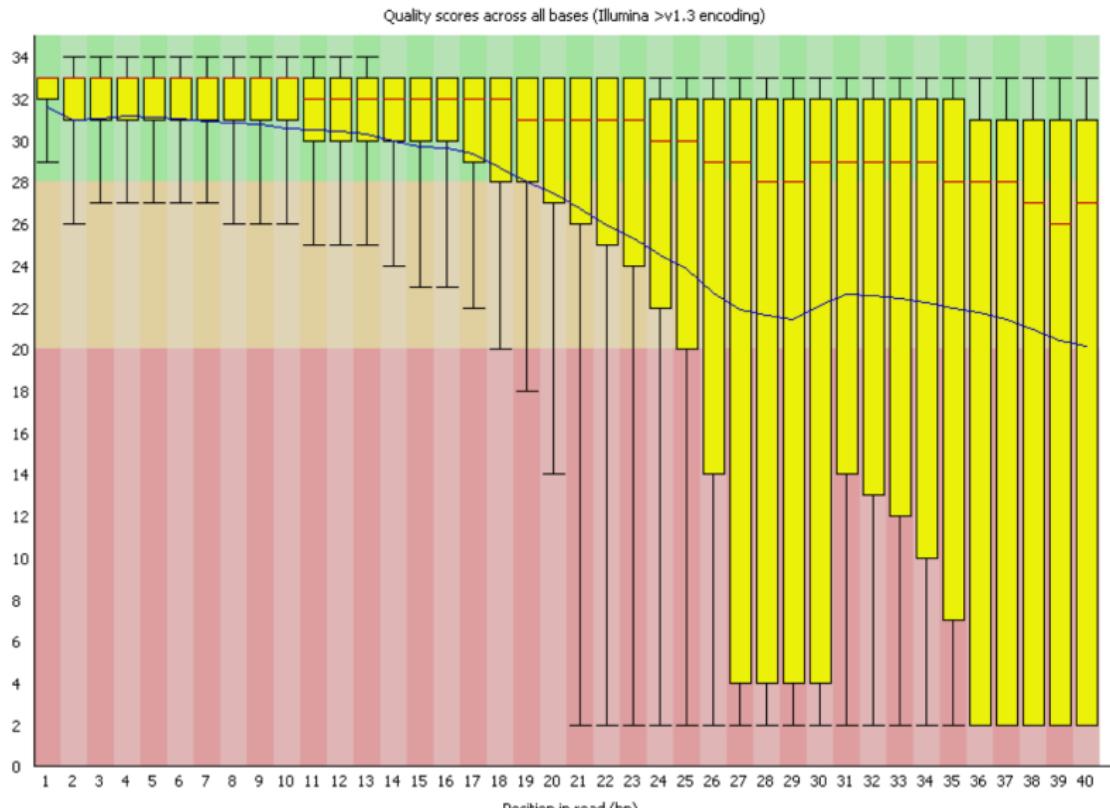
Comparison of some of the more popular QC pipelines:

QC Features	QCToolkit	QC-Chain	Cutadapt	Prinseq
Parallel computation	X	X		
Phred offset detection	X	X		
Tag sequence removal	X	X	X	
Poly-A/T tail trimming			X	X
PCR duplication filtering			X	X
Low complexity filtering				X
Homopolymer removal	X			X
GC content filtering		X		X
N/X content filtering				X

# FASTQC



# FASTQC



# FastQC Example

Interactive GUI from Amarel Desktop:

```
module spider fastqc  
#module avail fastqc  
module load FastQC  
fastqc
```

Running FastQC from command-line (single file):

```
fastqc myfastqfile.fq.gz --outdir=output/
```

Running FastQC from command-line (multiple files):

```
fastqc *.fq.gz --outdir=output/
```

# MultiQC Example

You can use `multiqc` to combine FastQC results.

```
pip install multiqc  
cd my_fastqc_dir  
multiqc .
```

# MultiQC Example

**General Statistics**

Name	% Dups	% GC	M Seqs
brother_short	71.1%	51%	1.2M
grandmother_short	72.5%	51%	1.2M
mother_short	72.0%	52%	1.2M
probond_20	69.8%	51%	0.8M
probond_short	67.7%	47%	1.0M
uncle_short	71.9%	51%	1.1M

**FastQC** Version: 0.11.9

FastQC is a quality control tool for high throughput sequence data, written by Simon Andrews at the Babraham Institute in Cambridge.

**Sequence Counts**

Sequence counts for each sample. Duplicate read counts are an estimate only.

**Sequence Quality Histograms**

The mean-quality value across each base position in the read.

## Section 5

### Mapping Reads

# Sequencing data alignment

Next-Generation Sequencing (NGS) Data present new challenges:

- Map 'reads' to genome
- Call SNPs and variants from the reads
- Other Applications: RNA-seq, miRNAs, alternative splicing, ChIP-seq, BS-seq, RNA editing

Obstacles:

- Massive data size
- Repeat regions, rare variants

**Goal:** Develop an approach to put the puzzle together!

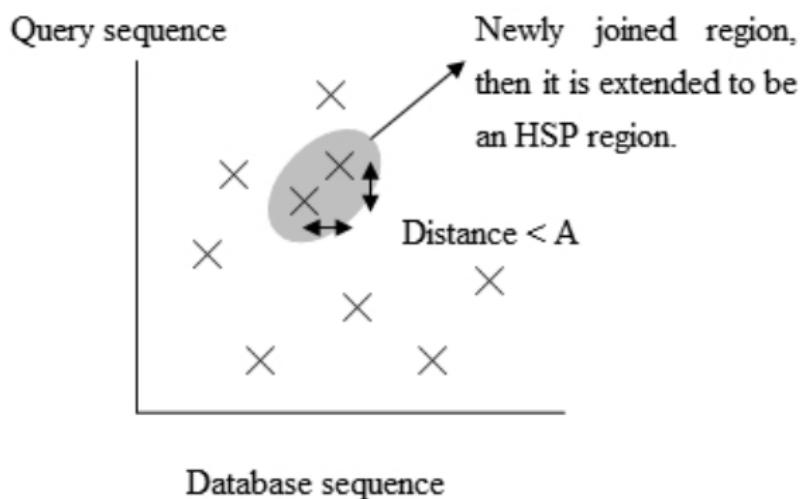
# Mapping Algorithms

Here are a few examples of alignment algorithms (not comprehensive – some of Dr. Johnson's favorites!)

- BLAST
- BLAT
- SOAP2
- Bowtie2
- BWA
- STAR
- Subread (Rsubread)
- GNUMAP
- NovoAlign
- Many others!!

# Basic Local Alignment Search Tool (BLAST)

BLAST is a “heuristic” method that identifies and combined *High-scoring Segment Pairs (HSPs)* between two sequences. BLAST searches for high scoring sequence alignments between the query sequence and the existing sequences in the database using an approach that approximates the Smith-Waterman algorithm (described later).



# Basic Local Alignment Search Tool (BLAST)

BLAST can be found at: <https://blast.ncbi.nlm.nih.gov/Blast.cgi>  
Try “BLASTING” the following sequences:

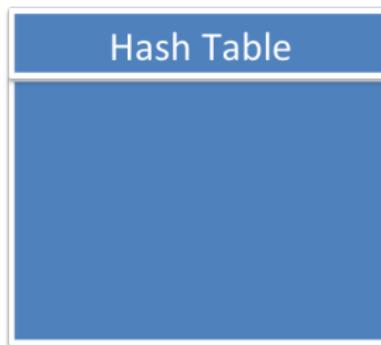
>NC\_045512.2 Severe acute respiratory syndrome coronavirus 2 isolate Wuhan-Hu-1  
ATTAAGGTTATACCTCCAGGTAAACAAACCAACTTCGATCTCTGTAGATCTGTTCTAAACGAACTTAA

>NC\_001802.1 Human immunodeficiency virus 1  
GGTCTCTCTGGTTAGACCAGATCTGAGCCTGGGAGCTCTGGCTAACTAGGAAACCCACTGCTTAAGCCTAATAAAGC

>NR\_102810.2 Mycobacterium tuberculosis H37Rv 16S ribosomal RNA  
TGTTGGAGAGTTGATCCTGGCTCAGGACGAACGCTGGCGCGTGGCTAACACATGCAAGTCGAACCGGA  
AAGGTCTCTCGGAGATACTCGAGTGGCGAACGGGTGAGTAACACGTGGGTGATCTGCCCTGCACCTCGG

>NR\_041248.1 Bacillus anthracis strain ATCC 14578 16S ribosomal RNA  
AGAGCTTGCTCTTATGAAGTTAGCGGCGGACGGGTGAGTAACACGTGGTAACCTGCCATAAGACTGGG

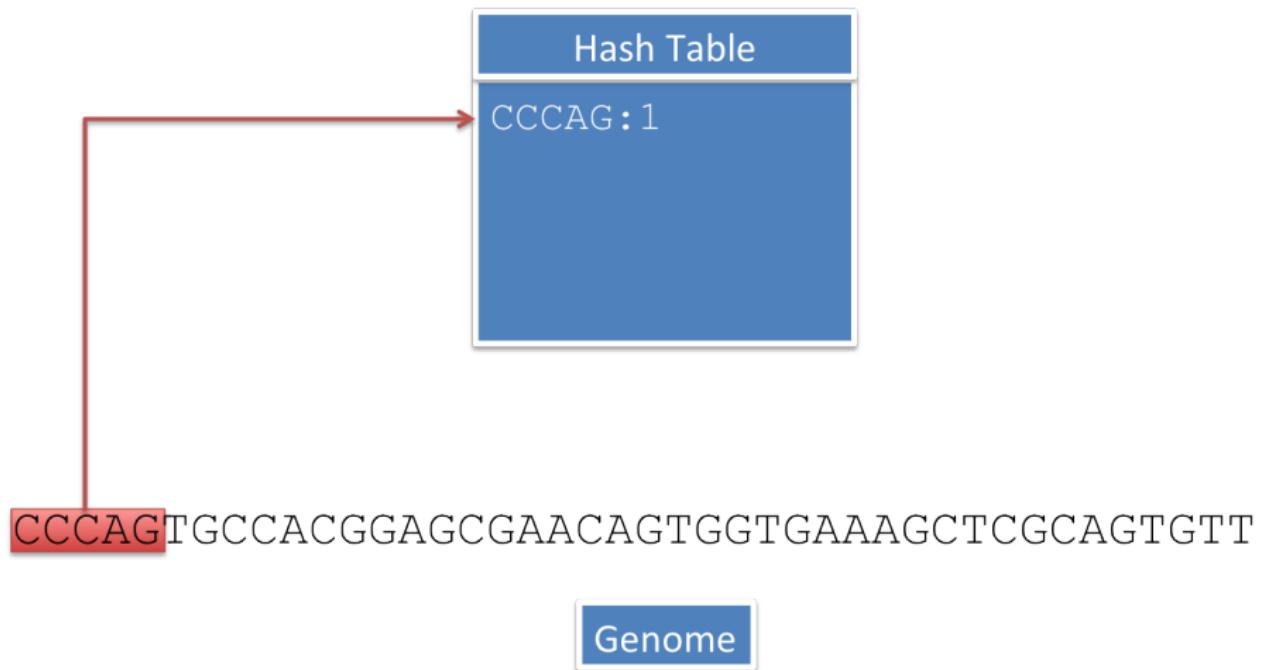
# Creating a Hash Index Table



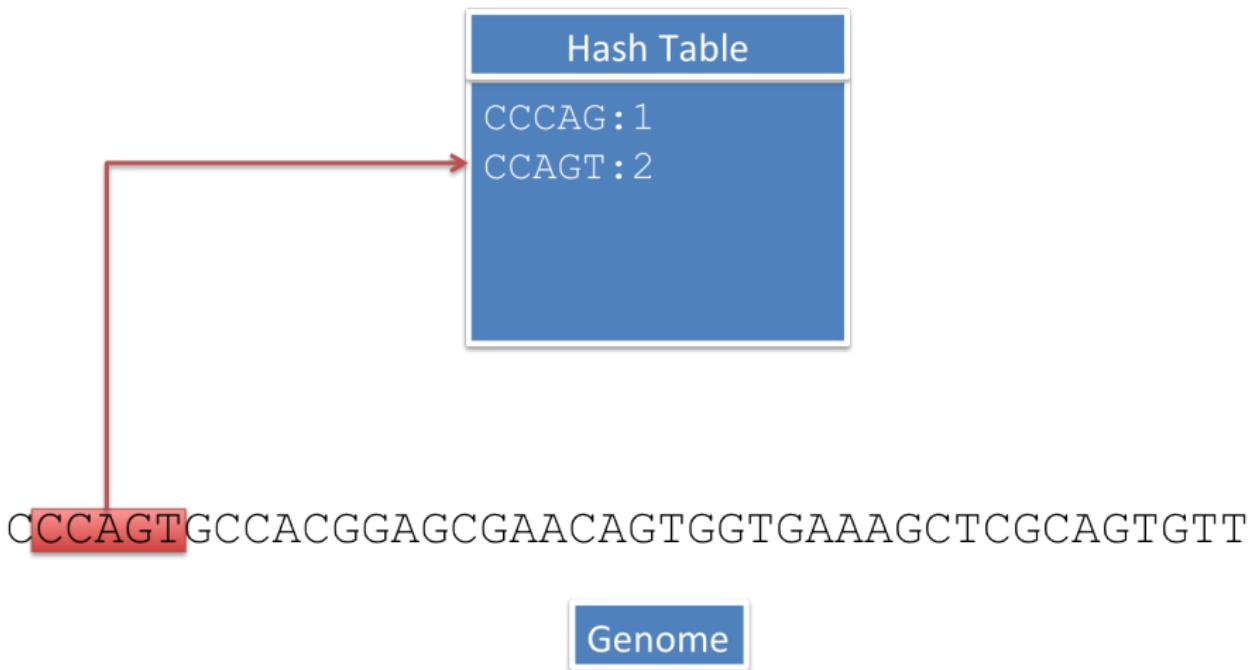
CCCAGTGCCACGGAGCGAACAGTGGTGAAAGCTCGCAGTGTT

The diagram shows a single line of DNA sequence: CCCAGTGCCACGGAGCGAACAGTGGTGAAAGCTCGCAGTGTT. Below this sequence is a blue rectangular box with rounded corners and a thin white border, containing the word 'Genome' in white. This box is centered below the sequence line.

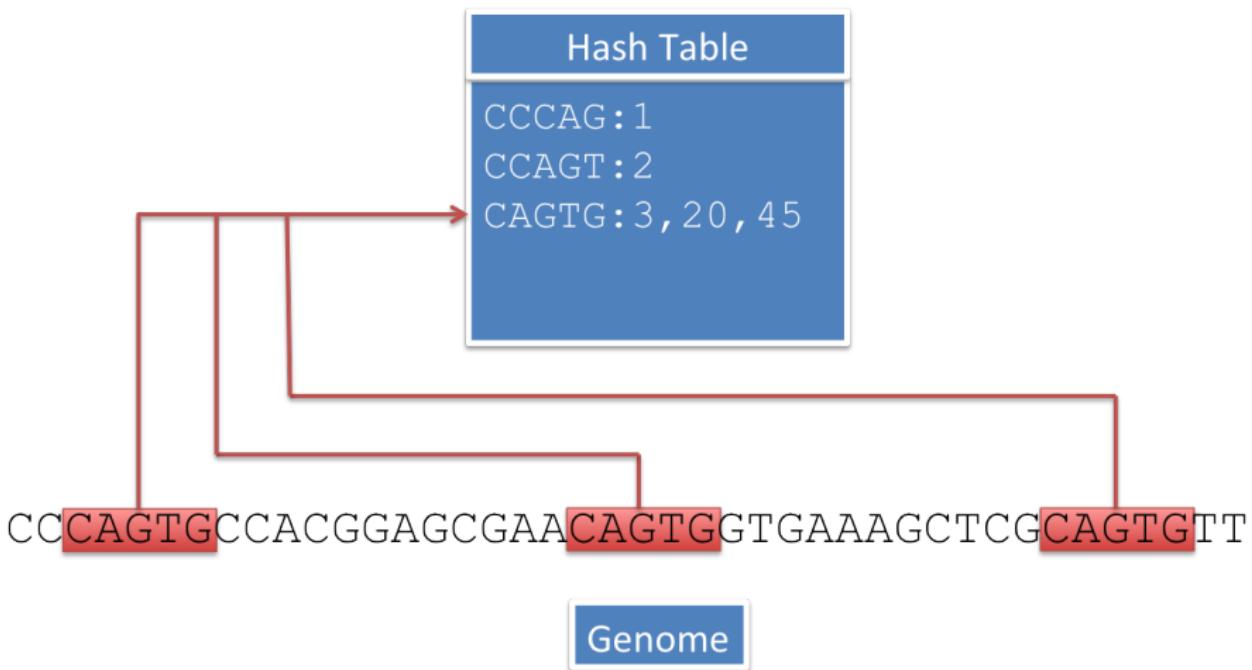
# Creating a Hash Index Table



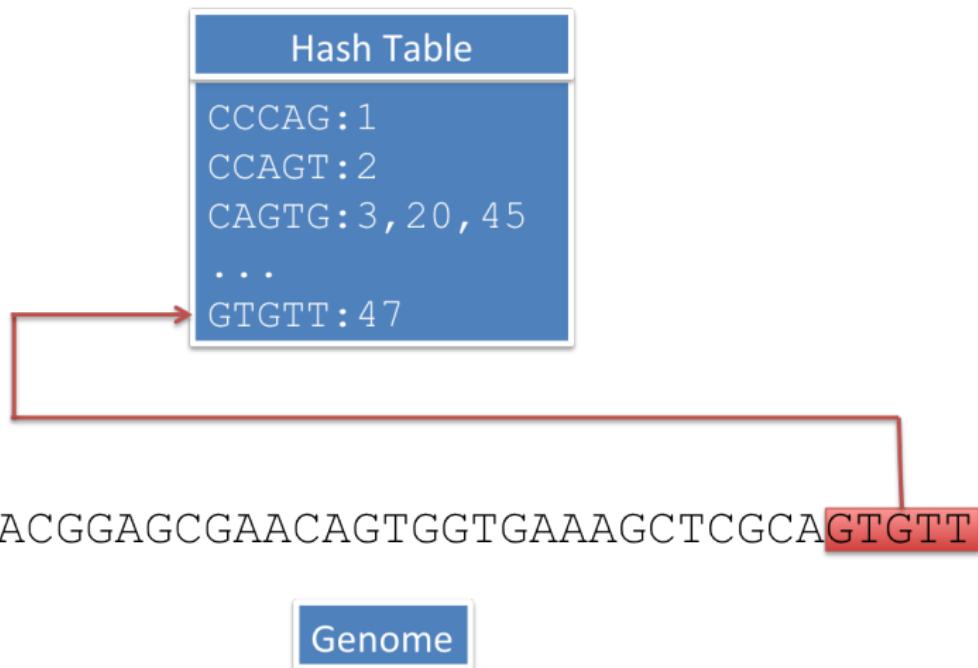
# Creating a Hash Index Table



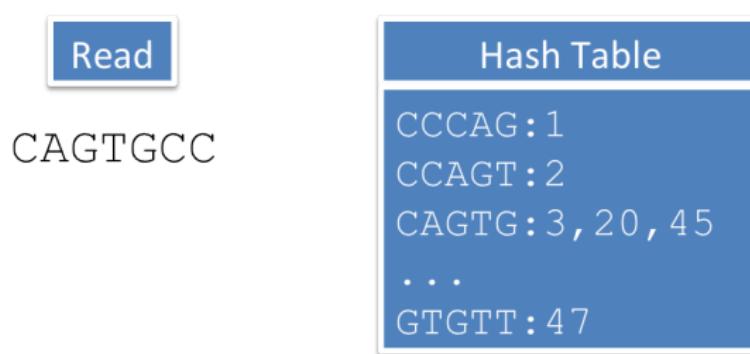
# Creating a Hash Index Table



# Creating a Hash Index Table



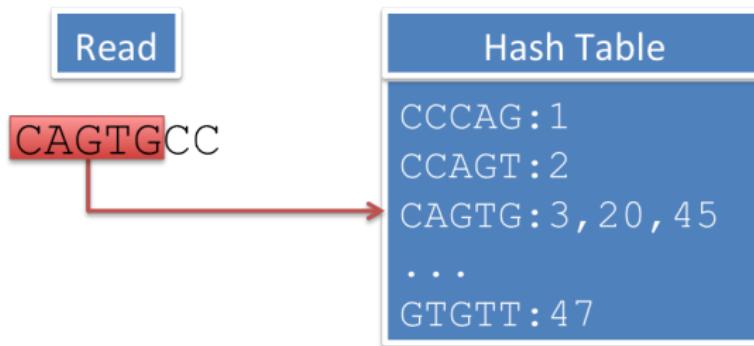
# Read Lookup



CCCAGTGCCACGGAGCGAACAGTGCTGAAAGCTCGCAGTGT

Genome

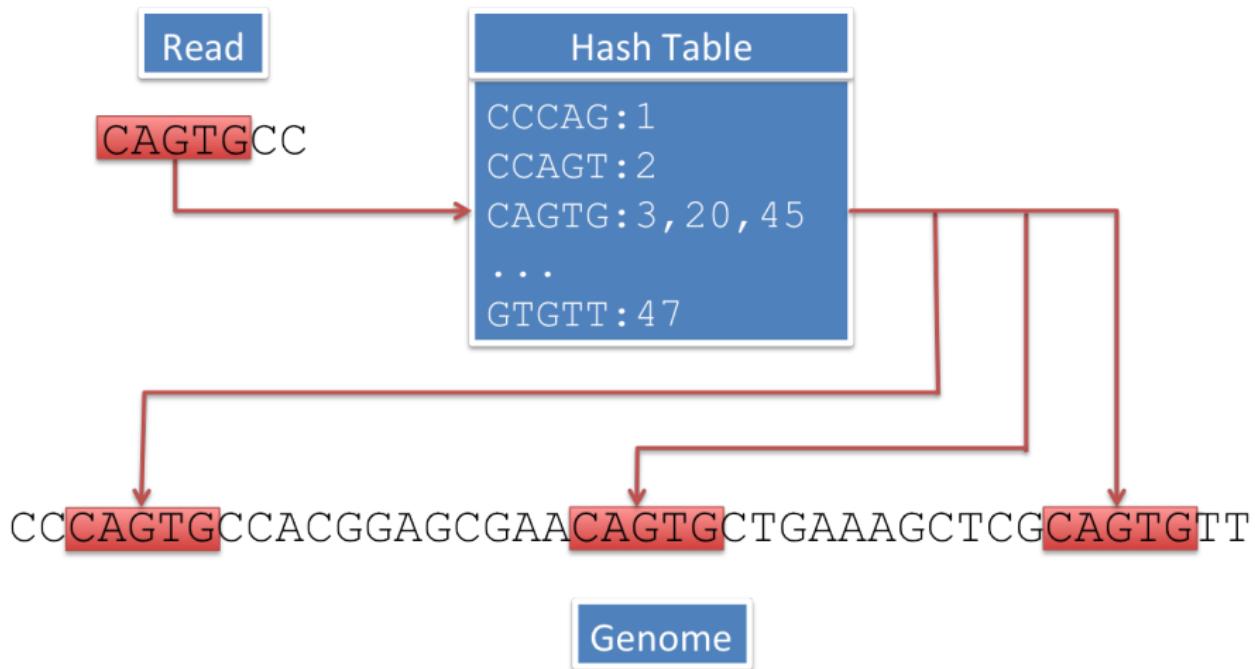
# Read Lookup



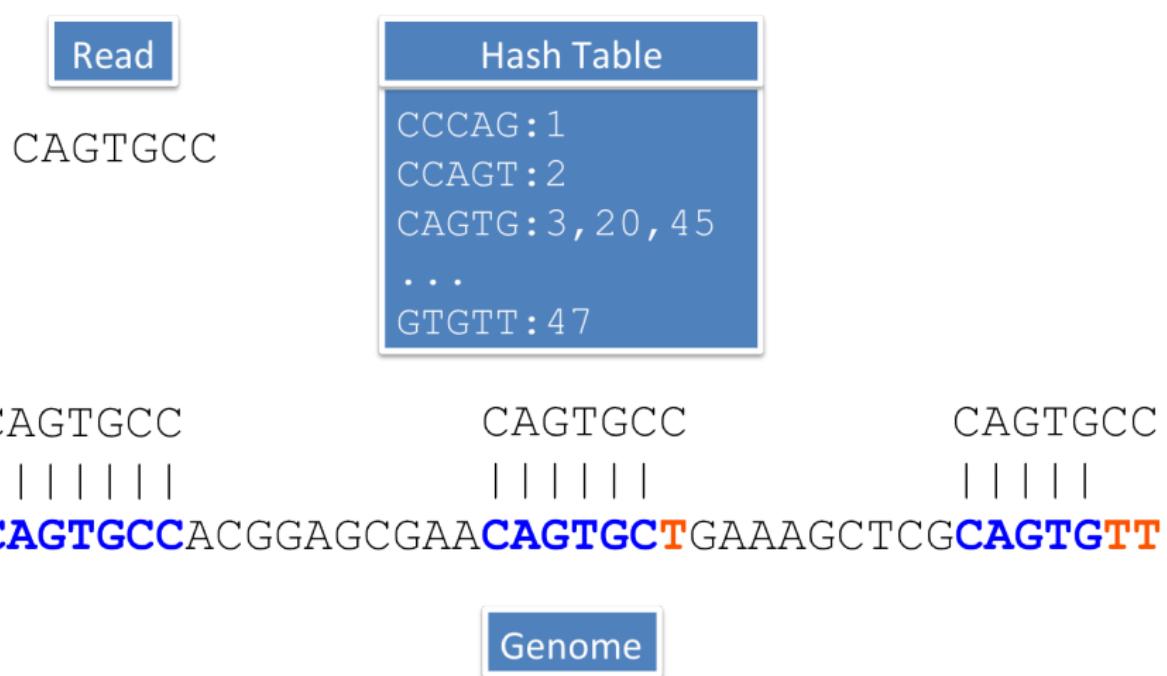
CCCAGTGCCACGGAGCGAACAGTGCTGAAAGCTCGCAGTGT

Genome

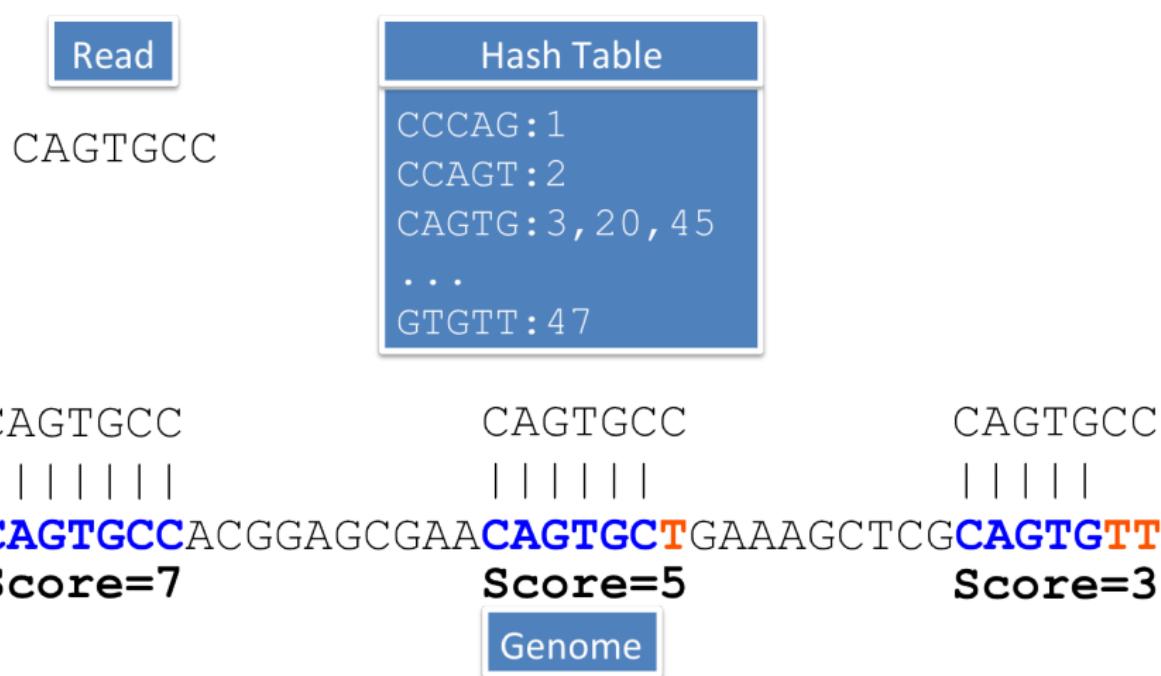
# Read Lookup



# Read Lookup



# Read Lookup



# Needleman-Wunsch

T A C A T C

T T C A - C

	t	a	c	a	t	c
t						
t						
c						
a						
c						

- Allows for mismatches and gaps
- Scoring function: +1 match; -1 for mismatch; -2 for gap
- Produces maximum likelihood alignment

# Needleman-Wunsch

T A C A T C

T T C A - C

	t	a	c	a	t	c
t	0	-2	-4	-6	-8	-10
t		-2				
c			-6			
a				-8		
c					-10	

- Allows for mismatches and gaps
- Scoring function: +1 match; -1 for mismatch; -2 for gap
- Produces maximum likelihood alignment

# Needleman-Wunsch

```

T A C A T C
|
T T C A - C
  
```

	t	a	c	a	t	c
t	0	-2	-4	-6	-8	-10
t	-2	1				
t	-4					
c	-6					
a	-8					
c	-10					

- Allows for mismatches and gaps
- Scoring function: +1 match; -1 for mismatch; -2 for gap
- Produces maximum likelihood alignment

# Needleman-Wunsch

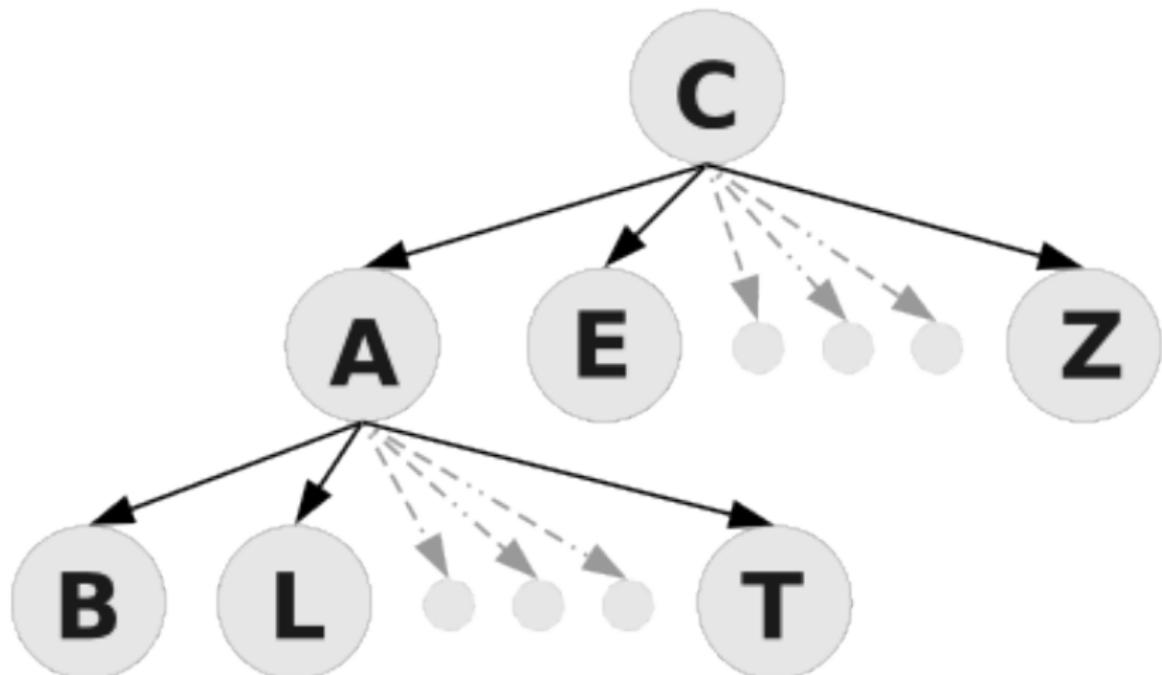
```

T A C A T C
|   |   |
T T C A - C
  
```

	t	a	c	a	t	c	
t	0	-2	-4	-6	-8	-10	-12
t	-2	<b>1</b>	-1	-3	-5	-7	-9
t	-4	-1	<b>0</b>	-2	-4	-4	-6
c	-6	-3	-2	<b>1</b>	-1	-3	-3
a	-8	-5	-2	-1	<b>2</b>	<b>0</b>	-2
c	-10	-7	-4	-1	0	1	<b>1</b>

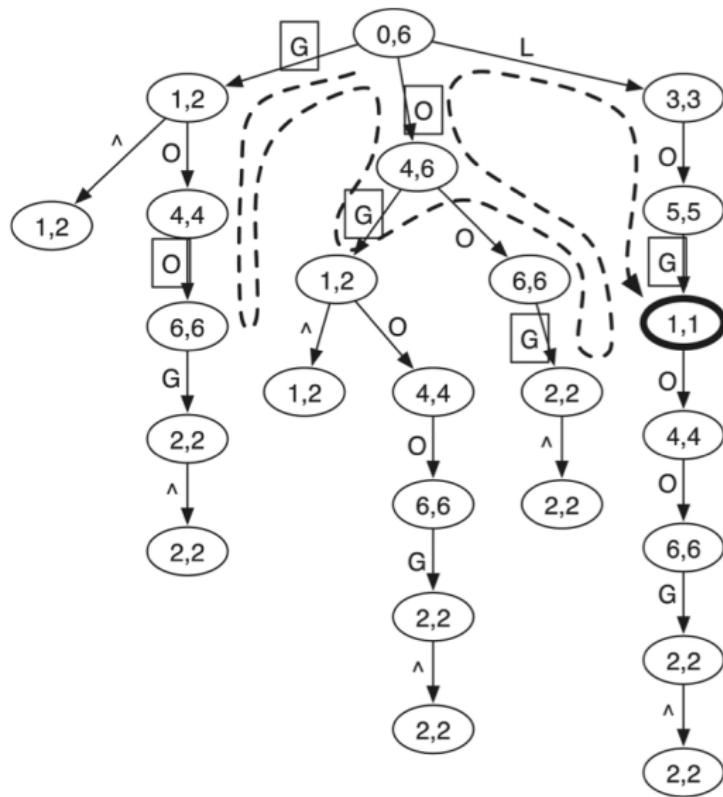
- Allows for mismatches and gaps
- Scoring function: +1 match; -1 for mismatch; -2 for gap
- Produces maximum likelihood alignment

# Prefix Trees



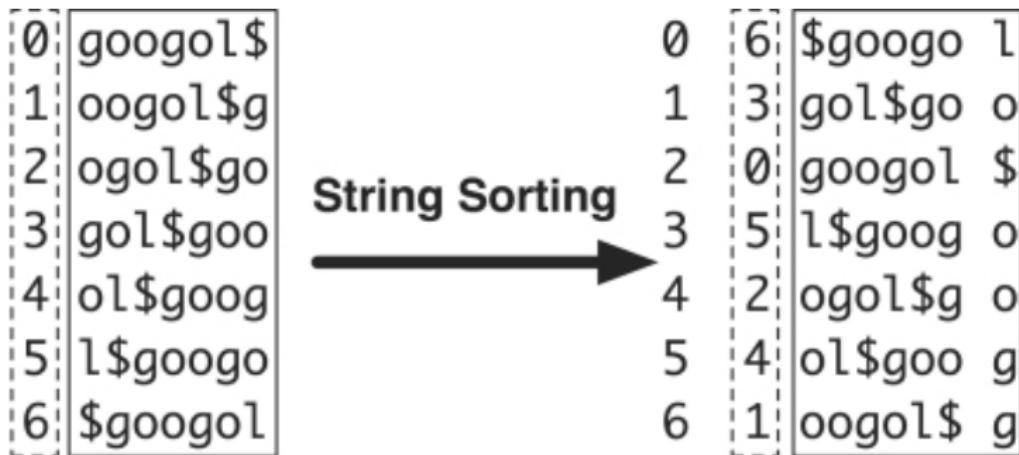
{hei}

# Prefix Trees



Li H , and Durbin R Bioinformatics 2009;25:1754-1760

# Burrows-Wheeler Transformation (BWA, Bowtie)



Pos

 $X = \text{googol\$}$ 
 $i$     $S(i)$     $B[i]$   
↓   ↓   ↓  
 $(6, 3, 0, 5, 2, 4, 1)$   
 $\text{lo\$oogg}$

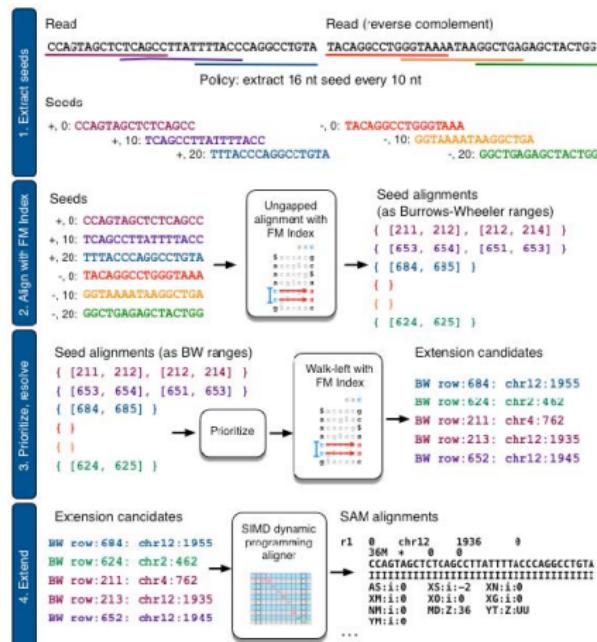
## Burrows-Wheeler Transformation (BWA, Bowtie)

(a) \$acaacg  
aacg\$ac  
acaacg\$  
acaacg\$ → acg\$aca → gc\$aac  
caacg\$a  
cg\$aca a  
q\$acaac

(c) **a a c**      **a a c**      **a a c**

\$acaacg	\$acaacg	\$acaacg
a acg\$ac	a acg\$ac	a acg\$ac
a caacg\$	a caacg\$	a caacg\$
a cg\$aca	a cg\$aca	a cg\$aca
c aacg\$ a	c aacg\$ a	c aacg\$ a
c g\$aca a	c g\$aca a	c g\$aca a
q \$acaac	q \$acaac	q \$acaac

# Hybrid Methods (BWA-SW, Bowtie2)



# Read alignments are stored in SAM/BAM/CRAM

- BAM and CRAM are compressed SAM files
- One line per alignment
- Use `samtools` to view, sort, merge, concatenate, index, get statistics, etc, on alignment files.

# SAM/BAM/CRAM Format – Header

The *header* lines start with @:

- @HD = header definition
- @SQ = a sequence in the reference file you used, followed by how long it was and its comment (from the reference file)
- @RG = read groups you assigned while mapping the reads
- @PG = programs used to obtain this bam, in order

# SAM/BAM/CRAM Format – Body

Each alignment has 11 mandatory fields:

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

# SAM/BAM/CRAM Format – Flag

The flag is the summation of the following binary attributes:

Binary (Decimal)	Hex	Description
0000000001 (1)	0x1	Is the read paired?
0000000010 (2)	0x2	Are both reads in a pair mapped “properly” (i.e., in the correct orientation with respect to one another)?
00000000100 (4)	0x4	Is the read itself unmapped?
00000001000 (8)	0x8	Is the mate read unmapped?
00000010000 (16)	0x10	Has the read been mapped to the reverse strand?
00000100000 (32)	0x20	Has the mate read been mapped to the reverse strand?
00001000000 (64)	0x40	Is the read the first read in a pair?
00010000000 (128)	0x80	Is the read the second read in a pair?
00100000000 (256)	0x100	Is the alignment not primary? (A read with split matches may have multiple primary alignment records.)
01000000000 (512)	0x200	Does the read fail platform/vendor quality checks?
10000000000 (1024)	0x400	Is the read a PCR or optical duplicate?

Example flags:

Single end read mapped to + strand: 0 (no flags apply)

Single end read mapped to - strand: 16

Unmapped, paired end first mate read with unmapped mate:  $69 = 1 + 4 + 8 + 64$

## SAM/BAM/CRAM Format – Example

# SAM/BAM/CRAM Format

- Use `samtools view` to see the content of a SAM/BAM/CRAM file.
- Always sort and index them
- Use `samtools view -f` (include) and `-F` (exclude) to filter by flags
- Use `samtools view -q` to filter by quality.
- More on `samtools` later!!

# BWA Example (OnDemand Amarel Desktop)

```
# load bwa
module load bwa

# Index the genome:
bwa index genomefile.fa

# Align the reads:
bwa mem genomefile.fa myfastqfile.fq > bwaalign.sam

# Note: for use with gatk (will learn later) bwa mem
# should include a `read group` header as well, e.g.:
-R "@RG\tID:group1\tSM:sam1\tPL:illumina\tLB:lib1\tPU:unit1"
```

# Bowtie2 Example (OnDemand Amarel Desktop)

```
# Load Bowtie2
module load bowtie2

# Index the genome:
bowtie2-build genomefile.fa output/genomefile

# Align the reads:
bowtie2 -x output/genomefile -U myfastqfile.fq \
        -S bowtiealign.sam
```

# RSubread Example (OnDemand RStudio)

```
# Install the Bioconductor installer
install.packages("BiocManager")

# Install Rsubread
BiocManager::install("Rsubread")

# Index the genome:
buildindex("genomefile", "genomefile.fa")

# Align the reads:
align("genomefile", "myfastqfile.fq",
      output_file="myfastq.bam", nthreads = 4)
```

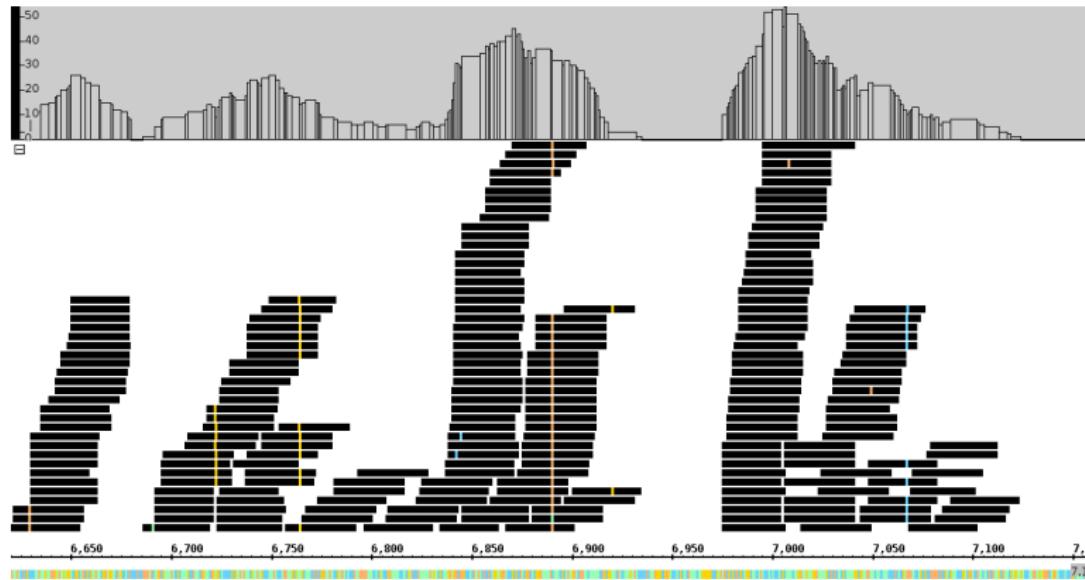
## Section 6

### Visualization of Aligned Sequencing Data

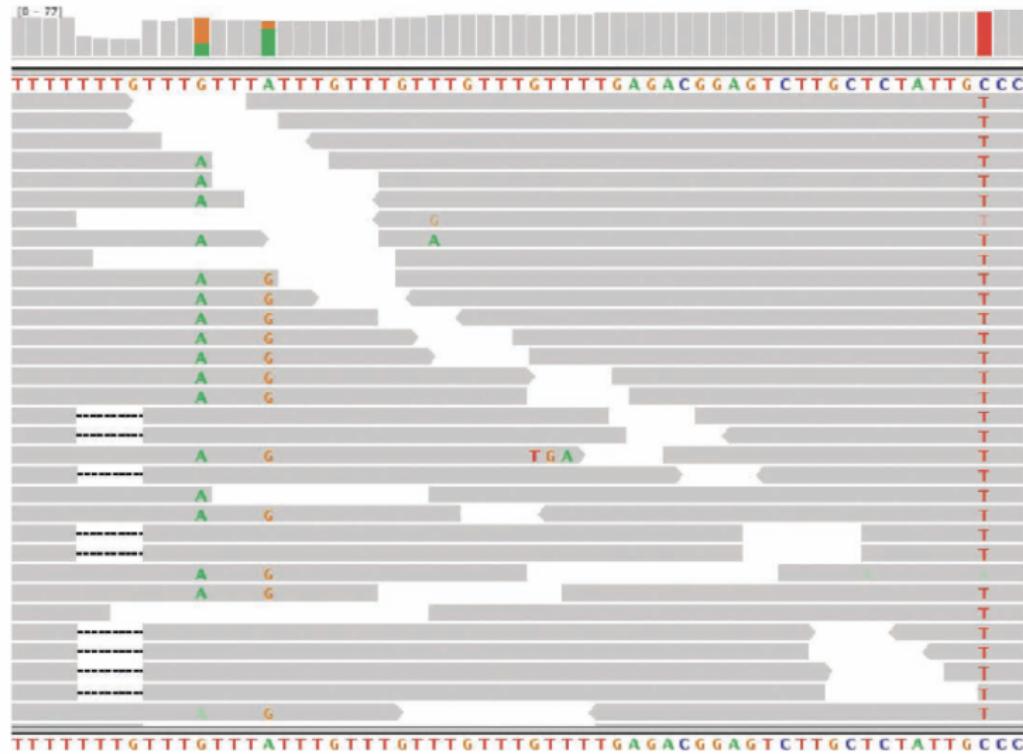
# Options for data visualization

- UCSC Genome Browser
- IGB (<http://bioviz.org/igb/>)
- IGV (<http://www.broadinstitute.org/igv/>)
- SAMtools

# IGB Example



# IGV Example



DePristo et al. *Nature Genetics* 2011; 43: 491-8.

# SAMtools

```

152853001 152853011 152853021 152853031 152853041 152853051 152853061
TGCCTAAATTCTAGAAGCTGCCACCTGGGCCAGGAAGGCCATGGTAGAAGTAGTATTCTGTTAGTCGGC
.....G.....
tgcc aaattcagaagctgcccacctggggccagggcatggtagaagtagtatttcatctggtagttctcgccc
tgcccaaattcagaagctgcccacc GGGGCCAGGGAAAGGCCATGGTAGAAGTAGTATTCTGTTAGTCGGC
tgcccaaattcagaagctgcccacc GGGGCCAGGGAAAGGCCATGGTAGAAGTAGTATTCTGTTAGTCGGC
tgcccaaattcagaagctgcccacct GGGCCAGGGAAAGGCCATGGTAGAAGTAGTATTCTGTTAGTCGGC
tgcccaaattcagaagctgcccacct ggcattggtagaagtagtatttcatctggtagttctcgccc
tgcccaaattcagaagctgcccacct ggggcccaggaa ggcattggtagaagtagtatttcatctggtagttctcgccc
tgcccaaattcagaagctgcccacct ggggcccaggaaagg CATGGTAGAAGTAGTATTCTGTTAGTCGGC
tgcccaaattcagaagctgcccacct ggggcccaggaaaggccatg TAGAAGTGGTATTCTGTTAGTCGGC
tgcccaaattcagaagctgcccacct ggggcccaggaaaggcatg GAAGTAGTATTCTGTTAGTCGGC
tgcccaaattcagaagctgcccacct ggggcccaggaaaggcatg gaagtagtatttcatctggtagttctcgccc
TGCCCAAATTCTAGAAGCTGCCACCTGGGCCAGGGAAAGGCCATGGTAGAA tagtatttcatctggtagttctcgccc
TGCCCAAATTCTAGAAGCTGCCACCTGGGCCAGGGAAAGGCCATGGTAGAA agtatttcatctggtagttctcgccc
tgcccaaattcagaagctgcccacct ggggcccaggaaaggccatggtagaagtagtattt ATCTGGTAGTCGGC
tgcccaaattcagaagctgcccacct ggggcccaggaaaggccatggtagaagtagtattt tctggtagttctcgccc
TGCCCAAATTCTAGAAGCTGCCACCTGGGCCAGGGAAAGGCCATGGTAGAAGTAGTATTCTC TCTGGTAGTCGNNNI
tgcccaaattcagaagctgcccacct ggggcccaggaaaggccatggtagaagtagtattt tctggtagttctcgccc
taaaaaaattcagaagctccccacctaagggcaaggccataatagaagtagtatttcatc GGTAGTTCTCGGGC

```

# SAMtools Example (Amarel Desktop)

```
# Load SAMtools
module load samtools

# Convert SAM file to BAM format:
samtools view -bS myalignments.sam > myalignments.bam

# Sort the BAM file:
samtools sort myalignments.bam -o myalignments.sorted.bam

# Index the BAM file:
samtools index myalignments.sorted.bam

# View BAM file in SAMtools:
samtools tview myalignments.sorted.bam genomefile.fa
```

# SAMtools Example

SAMtools commands:

- The one command to remember: ‘?’
- g: go to a specific location (i.e. chrX:152852988 or chrX\_149913753:2939235)
- m,b: mapping quality, base quality
- n: color by nucleotide
- ‘.’: dot/base view
- r: read name
- q: quit SAMtools

# SAM/BAM/CRAM Format – Conversions

*# Convert SAM to BAM*

```
samtools view -bS in.sam > out.bam
```

*# Convert BAM to SAM*

```
samtools view -ho out.sam in.bam
```

*# Convert BAM to fastq*

```
samtools bam2fq in.bam > out.fastq
```

# RSamtools

Rsamtools is a very useful (although somewhat limited) version of Samtools available in R:

```
# Install Rsamtools  
BiocManager::install("Rsamtools")  
  
# Convert SAM to BAM  
asBam(in.sam, out.bam)
```

You can also index and sort .bam files, as well as extract alignments from a .bam file (very useful!).

## Section 7

### Other Data Formats

# Standard format for keeping tables

field1	field2	field3	...
...	...	...	...

Fields (columns) separated by a character on each line:

- Comma (or Character) Separated Vector (CSV)
- Tab Separated Vector (TSV)
- Some interpreters take any space (space or tab) as a separator (such as `awk`, `cut`).
- Some have column name as first row (header), some don't

# Genomic regions

- A region is defined by three required fields:
  - sequence name (e.g. chromosome)
  - start coordinate
  - end coordinate
- Define regions of interest: introns, exons, genes, etc.
- Additional information saved as fields after the first three.
- Three standard tab-separated formats: BED, GFF, GTF No headers

# BED Format

Mandatory fields:

- ① chrom - Name of the chromosome/scaffold/reference sequence
- ② chromStart - 0-based starting position of the feature on chrom
- ③ chromEnd - Ending position of the feature in the chromosome or scaffold.

The chromEnd base is not included in the display of the feature.

For example, the first 100 bases of a chromosome are defined as:

- chromStart=0
- chromEnd=100
- span the bases numbered 0-99

# BED Format

chr1	11873	14409	uc001aaa.3	0	+	11873	11873
chr1	11873	14409	uc010nxr.1	0	+	11873	11873
chr1	11873	14409	uc010nxq.1	0	+	12189	13639
chr1	14361	16765	uc009vis.3	0	-	14361	14361
chr1	14361	19759	uc009vit.3	0	-	14361	14361
chr1	14361	19759	uc009viu.3	0	-	14361	14361
chr1	14361	19759	uc001aae.4	0	-	14361	14361
chr1	14361	29370	uc001aah.4	0	-	14361	14361
chr1	14361	29370	uc009vir.3	0	-	14361	14361
chr1	14361	29370	uc009viq.3	0	-	14361	14361
chr1	14361	29370	uc001aac.4	0	-	14361	14361
chr1	14406	29370	uc009viv.2	0	-	14406	14406
chr1	14406	29370	uc009viw.2	0	-	14406	14406
chr1	15602	29370	uc009vix.2	0	-	15602	15602
chr1	15795	18061	uc009vjd.2	0	-	15795	15795
chr1	16606	29370	uc009viy.2	0	-	16606	16606
chr1	16606	29370	uc009viz.2	0	-	16606	16606
chr1	16857	17751	uc009vjc.1	0	-	16857	16857
chr1	16857	19759	uc001aaai.1	0	-	16857	16857

# BED Format

Human chr22 - UCSC Genome Browser

genome.ucsc.edu/cgi-bin/hgTracks?db=hg19&position=chr22%3A20100000-2010090...

Genomes    Genome Browser    Tools    Mirrors    Downloads    My Data    View    Help    About Us

## UCSC Genome Browser on Human Feb. 2009 (GRCh37/hg19) Assembly

move <<< << < > >> zoom in 1.5x 3x 10x base zoom out 1.5x 3x 10x 100x

chr22:20,100,000-20,100,900 901 bp. enter position, gene symbol or search terms go

chr22 (q11.21) 22p13 22p12 p11.2 q11.21 q12.1 q12.2 22q12.3 q13.1 q13.2 q13.31

Scale: 200 bases hg19  
chr22: 20,100,100 20,100,200 20,100,300 20,100,400 20,100,500 20,100,600 20,100,700 20,100,800  
Color by strand demonstration  
Chromosome coordinates list

start

move start < 2.0 > move end < 2.0 >  
track search default tracks default order hide all manage custom tracks track hubs configure reverse resize refresh  
collapse all expand all

Click on a feature for details. Click or drag in the base position track to zoom in. Click side bars for track options. Drag side bars or labels up or down to reorder tracks. Drag tracks left or right to new position.

Use drop-down controls below and press refresh to alter tracks displayed. Tracks with lots of items will automatically be displayed in more compact modes.

# BED Format

Optional fields:

- ④ Name
- ⑤ Score
- ⑥ Strand
- ⑦ 7-12. Display options (thick starts and end, color, blocks...) to control the view on the genome browser

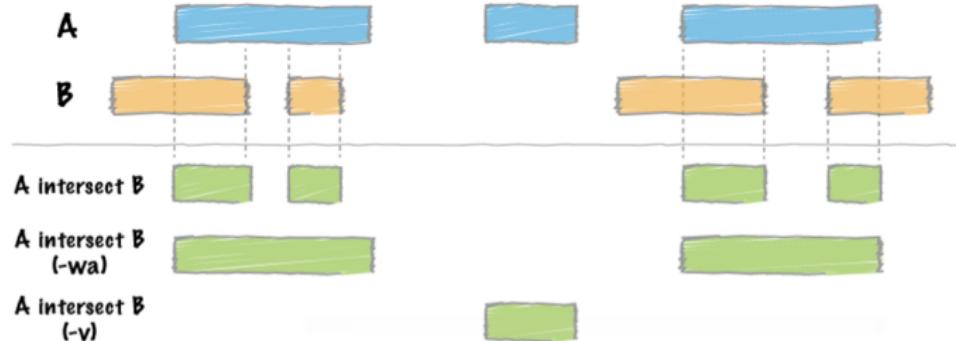
# bedtools

<http://bedtools.readthedocs.io/>

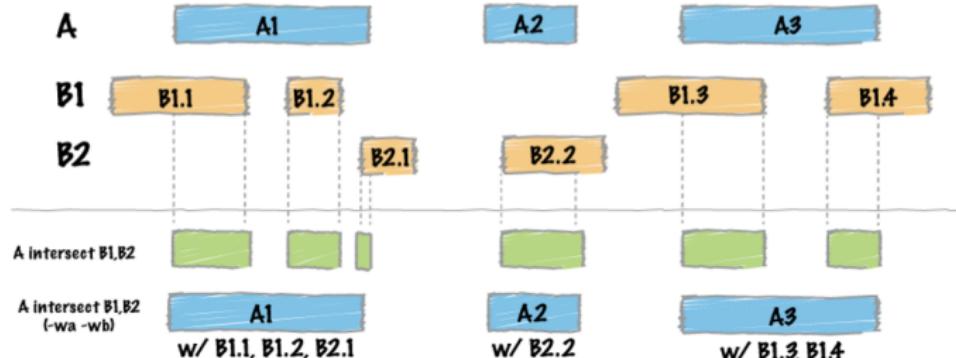
- sort (sort bed files)
- Intersect (get intersections of bed files)
- merge
- coverage
- overlap
- subtract
- ...

# bedtools intersect

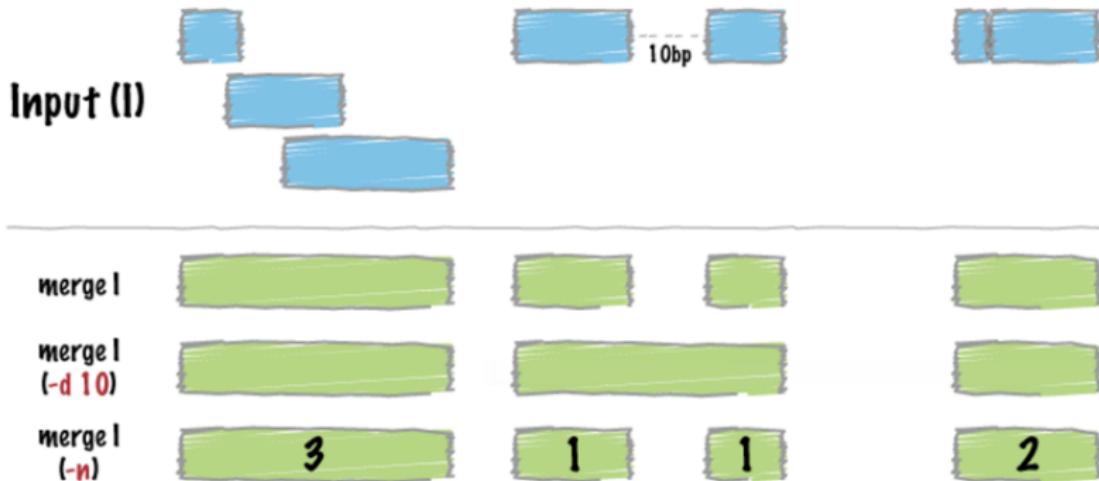
Intersect w/  
1 database



Intersect w/  
2 or more databases



# bedtools merge



## GFF - General features

9 mandatory fields, tab separated:

- ① seqname - The name of the sequence. Must be a chromosome or scaffold.
- ② source - The program that generated this feature.
- ③ feature - The name of this type of feature (e.g. gene, exon, etc).
- ④ start - The starting position of the feature in the sequence (1-based)
- ⑤ end - The ending position of the feature (inclusive).
- ⑥ score - A score between 0 and 1000.
- ⑦ strand - Valid entries include "+", "-", or "." (for don't know/don't care).
- ⑧ frame - If the feature is a coding exon, frame should be a number between 0-2 that represents the reading frame of the first base. If the feature is not a coding exon, the value should be ":".
- ⑨ group - All lines with the same group are linked together into a single item

# Gene Information

## GTF (Gene Transfer Format, GTF2.2)

- Extension to GFF2, backwards compatible
- First eight GTF fields are the same as GFF
- *feature field* is the same as GFF, has controlled vocabulary:
  - *gene, transcript, exon, CDS, 5UTR, 3UTR, inter, inter\_CNS, and intron\_CNS, etc*
- group field expanded into a list of attributes (i.e. key/value pairs)

The attribute list must begin with the one mandatory attribute: - gene\_id value - A globally unique identifier for the genomic source of the sequence

# GTF format

```
##description: evidence-based annotation of the human genome (GRCh38), version 27 (Ensembl 90)
##provider: GENCODE
##contact: gencode-help@sanger.ac.uk
##format: gtf
##date: 2017-08-01
chr1    HAVANA  gene    923928  944581  .       +       .
"protein_coding"; gene_id "ENSG00000187634.11"; gene_type
"protein_coding"; gene_name "SAMD11"; level 2; havana_gene "OTTHUMG00000040719.10";
```

- seqname: chr1
- source: HAVANA
- feature: gene
- start: 923928
- end: 944581
- score: . (no score)
- strand: +
- frame: . (not coding feature)
- attributes:
  - gene\_id: ENSG00000187634.11
  - gene\_type: protein\_coding
  - gene\_name: SAMD11
  - level: 2
  - havana\_gene: OTTHUMG00000040719.10

## ## GFF/GTF encodes relationships

- Features are hierarchical, e.g.:
  - A gene has 1 or more transcripts
  - A transcript has 1 or more exons

## Section 8

# SNP and Variant Calling

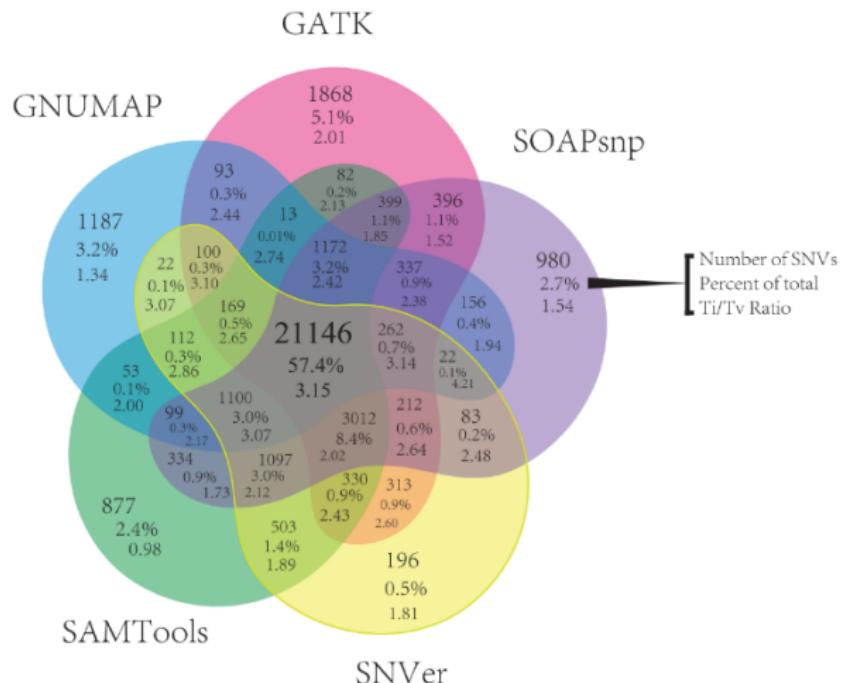
# Methods for SNP Calling

Methods for SNP calling:

- Mapper/callers: MAQ, SOAPsnp, GNUMAP, Crossbow (Bowtie)
- Callers: SAMtools (mpileup), GATK (HaplotypeCaller, Mutect2), FreeBayes, others

# Inconsistencies Among Aligners

Low concordance of variant-calling pipelines (O'Rawe, *Genome Med*, 2013)



# SAMtools Example

Multiple sample SNP calling:

```
 samtools mpileup -f genomefile.fa \
    myalignments.sorted.bam > myalignments.vcf
```

# VCF files

8 fixed columns: #CHROM, POS, ID, REF, ALT, QUAL, FILTER, INFO, FORMAT  
 Additional columns, one for each sample, with sample ID

(a) VCF example

Header	##fileformat=VCFv4.1 ##fileDate=20110413 ##source=VCFtools ##reference=file:///refs/human_NCBI36.fasta ##contig=<ID=1,length=249250621,md5=1b22b98cdeb4a9304cb5d48026a85128,species="Homo Sapiens"> ##contig=<ID=X,length=155270560,md5=7e0e2e580297b7764e31dbc80c2540dd,species="Homo Sapiens"> ##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele"> ##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership"> ##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype"> ##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality"> ##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth"> ##ALT=<ID=DEL,Description="Deletion"> ##INFO=<ID=SVTYPE,Number=1,Type=String,Description="Type of structural variant"> ##INFO=<ID=END,Number=1,Type=Integer,Description="End position of the variant">																																																											
Body	<table border="1"> <thead> <tr> <th></th><th>CHROM</th><th>POS</th><th>ID</th><th>REF</th><th>ALT</th><th>QUAL</th><th>FILTER</th><th>INFO</th><th>FORMAT</th><th>SAMPLE1</th><th>SAMPLE2</th></tr> </thead> <tbody> <tr> <td>1</td><td>1</td><td>.</td><td></td><td>ACG</td><td>A,AT</td><td>40</td><td>PASS</td><td>.</td><td>GT:DP</td><td>1/1:13</td><td>2/2:29</td></tr> <tr> <td>1</td><td>2</td><td>.</td><td></td><td>C</td><td>T,CT</td><td>.</td><td>PASS</td><td>H2;AA=T</td><td>GT</td><td>0 1</td><td>2/2</td></tr> <tr> <td>1</td><td>5</td><td>rs12</td><td></td><td>A</td><td>G</td><td>67</td><td>PASS</td><td>.</td><td>GT:DP</td><td>1 0:16</td><td>2/2:20</td></tr> <tr> <td>X</td><td>100</td><td>.</td><td>T</td><td>&lt;DEL&gt;</td><td>.</td><td>PASS</td><td>SVTYPE=DEL;END=299</td><td>GT:GQ:DP</td><td>1:12:..</td><td>0/0:20:36</td></tr> </tbody> </table>		CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	SAMPLE1	SAMPLE2	1	1	.		ACG	A,AT	40	PASS	.	GT:DP	1/1:13	2/2:29	1	2	.		C	T,CT	.	PASS	H2;AA=T	GT	0 1	2/2	1	5	rs12		A	G	67	PASS	.	GT:DP	1 0:16	2/2:20	X	100	.	T	<DEL>	.	PASS	SVTYPE=DEL;END=299	GT:GQ:DP	1:12:..	0/0:20:36
	CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	SAMPLE1	SAMPLE2																																																	
1	1	.		ACG	A,AT	40	PASS	.	GT:DP	1/1:13	2/2:29																																																	
1	2	.		C	T,CT	.	PASS	H2;AA=T	GT	0 1	2/2																																																	
1	5	rs12		A	G	67	PASS	.	GT:DP	1 0:16	2/2:20																																																	
X	100	.	T	<DEL>	.	PASS	SVTYPE=DEL;END=299	GT:GQ:DP	1:12:..	0/0:20:36																																																		

Danecek et al. Bioinformatics 2011

# Complete Variant Calling Pipeline (Outdated!})

Our analysis pipeline consisted of the following:

- Align the FASTQ files to genome
- Convert SAM file to BAM, add read group info
- Filter the reads based on quality (BAMTools)
- Samtools to sort and index, and use Picard to mark duplicates
- GATK calibration, realignment, variant calling (HaplotypeCaller, Mutect2)
- Filter the called variants (GATK filtersnps and filterindels).
- Annotation of SNPs (snpEff, condel)
- Filter by frequency (thousand genomes, TCGA, etc.)
- Downstream analysis (rare variants, pedigree, pathway level, etc)

# Downstream Annotation and Analysis (Outdated!)

Downstream Annotation Tools (old list):

- snpEff (<http://snpeff.sourceforge.net/>)
- Condel (<http://bg.upf.edu/condel/home>)
- SIFT <http://sift.jcvi.org/>
- Polyphen 2 <http://genetics.bwh.harvard.edu/pph2/>
- <http://mutationassessor.org/>
- Ensembl variant effect predictor  
(<http://www.ensembl.org/info/docs/variation/vep/index.html>)
- Thousand Genomes variant frequency (e.g. 1% threshold) and Exome Sequence Project variant frequency (e.g. 1%).

# Session info

```
sessionInfo()
```

```
## R version 4.4.0 (2024-04-24)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sonoma 14.2.1
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib; LAPACK version 3
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/Denver
## tzcode source: internal
##
## attached base packages:
## [1] stats      graphics   grDevices utils      datasets   methods    base
##
## loaded via a namespace (and not attached):
## [1] digest_0.6.35    fastmap_1.2.0    xfun_0.44        Matrix_1.7-0
## [5] lattice_0.22-6   reticulate_1.37.0 knitr_1.46     htmltools_0.5.8.1
## [9] png_0.1-8       rmarkdown_2.27    cli_3.6.2       grid_4.4.0
## [13] compiler_4.4.0   rstudioapi_0.16.0 tools_4.4.0     evaluate_0.23
## [17] Rcpp_1.0.12     yaml_2.3.8      jsonlite_1.8.8   rlang_1.1.3
```