



An Introduction to RNA-sequencing

QC, Preprocessing, and Alignment

W. Evan Johnson, Ph.D.
Professor, Division of Infectious Disease
Director, Center for Data Science
Co-Director, Center for Biomedical Informatics and Health AI
Rutgers University – New Jersey Medical School

2025-07-27

Installing R Packages:

Install the following tools: Rsubread, Rsamtools, and SummarizedExperiment. We will also need help from the tidyverse.

```
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install(c("Rsubread", "Rsamtools",
                      "tidyverse", "SummarizedExperiment"))
```

Load Packages for Today

We will be using the following packages for today's lecture:

```
library(tidyverse) ## tools for data wrangling
library(Rsubread) ## alignment and feature counts
library(Rsamtools) ## managing .sam and .bam files
library(SummarizedExperiment) ## managing counts data
```

Objective

- ▶ Disclaimer: non-comprehensive introduction to RNA-sequencing
- ▶ Introduce preprocessing steps
- ▶ Visualization
- ▶ Analytical methods
- ▶ Common software tools

Steps to an RNA-seq Analysis (Literacy)

1. Preprocessing and QC:

- ▶ Fasta and Fastq files
- ▶ FastQC: good vs. bad examples
- ▶ Visualization

2. Alignment

- ▶ Obtaining genome sequence and annotation
- ▶ Software: Bowtie, TopHat, STAR, Subread/Rsubread

3. Expression Quantification

- ▶ Count reads hitting genes, etc
- ▶ Approaches/software: HT-Seq, STAR, Cufflinks, RPKM FPKM or CPM, RSEM, edgeR, findOverlaps (GenomicRanges). featureCounts (Rsubread)

Steps to an RNA-seq Analysis (Literacy)

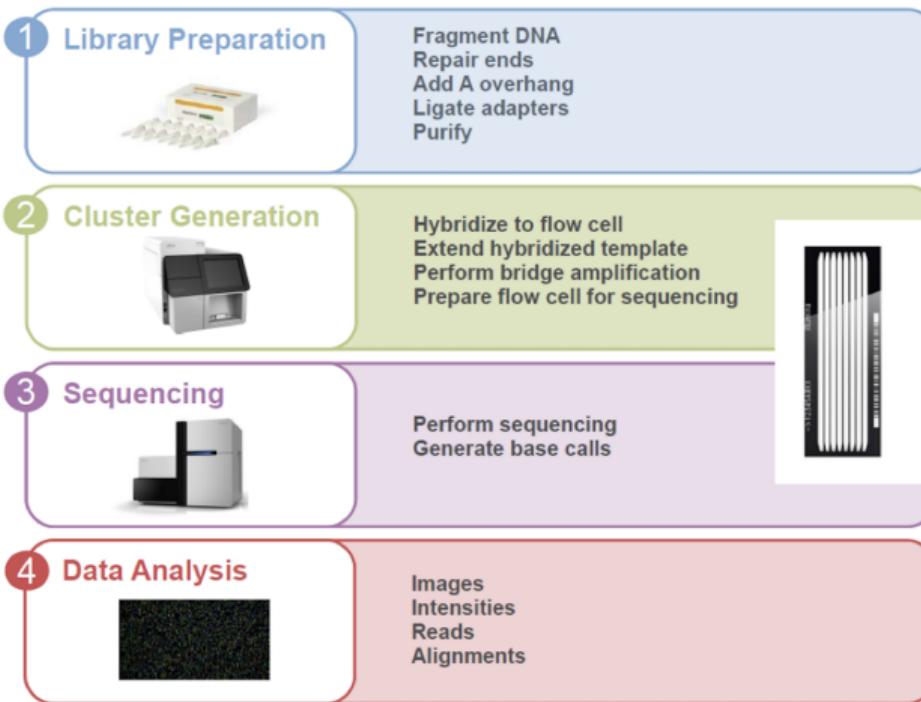
4. More visualization

- ▶ Heatmaps, boxplots, PCA, t-SNE, UMAP

5. Differential Expression

- ▶ Batch correction
- ▶ Overdispersion
- ▶ General Workflow
- ▶ Available tools: edgeR, DESeq, Limma/voom
- ▶ Even more visualization!!

Illumina Sequencing Workflow



Sequencing Data Formats

Genome sequencing data is often stored in one of two formats, FASTA and FASTQ text files. For example a FASTA file looks like the following:

```
>chrX
ttgaactcctgacctcaggtgatccgcggccttgcacctccaaagcgct
cctgcctcagcctcccagtagctggactacagggtgcctgccaccatgc
....
caggctaatttttgtatttttagtagagacgggtttcaccatgttagc
caggatggtctcaatctcctgacctcatgatccgcctgcctcgccctccc
>chrY
tgtacactttaaatgggtgaatttatggaatgtgaattataCGTGTGG
CTTGTAAAAAAAAATGATGGAGATGGAGACGTGACTCTAGCGTGAAGGGG
...
```



FASTQ Files

We can also store confidence or quality scores using a FASTQ format:

@HWI-EAS240_0001:2:1:1142:17571#0/1
CTCTCTTTCTCCCCANGTCTCCTCATGACCATATCCNTGTTGTCCATTGTGTANGNNNCTTGCCTGCAACATC
+HWI-EAS240_0001:2:1:1142:17571#0/1
bababb`abbbbabbYB^[^``_`_bbbbbbbbb`_B`_``^`^bbba^`^`B00BBBN0000bY^`^`bbbbbb
@HWI-EAS240_0001:2:1:1142:6453#0/1
CAGGACGTGCACTATGNCATCCACGATGCAAGTCTTANCATTATTCAAGGATACANATANNGCAAACGTGTAATT
+HWI-EAS240_0001:2:1:1142:6453#0/1
_U [aa_aa^``^`[B] [] []]S``Y`a`aaZ]B)]^`^`[`^`^`[O`Y]XBNNNBB]]]]]L^`^`^`^`^`[V`
@HWI-EAS240_0001:2:1:1142:19443#0/1
TCAGAAAACAGAAAGGTCTTACTTCTTGCANGATGCCACCCCTCCAGANCAGNNATTGCTACTTGCAC
+HWI-EAS240_0001:2:1:1142:19443#0/1
bbbabb^`^`bbabb]bb]^`^`]]]bbbbbbbbbYYBYX00000bb`bbVY^HB000BB0J0L0 bbbb]b`a

.FASTQ Paired End

fastq_SRR1997469_1.fastq

```
@SRR1997469.1 1 length=36
CAGTCTTCTTAGAAATATCCACTTCGGAATAAAAGA
+SRR1997469.1 1 length=36
BBBBBFFFFF<FFFFFFFFFFFFFFBFFFFFFFFFFF
@SRR1997469.2 2 length=36
ACAGTTAACGATCCTTACAGANAGNAGNCTNGTA
+SRR1997469.2 2 length=36
<BBBBFFFBBF<BFF/<FFFFF##########
...
```

fastq_SRR1997469_2.fastq

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



FASTQ Encoding

In order to translate FASTQ quality scores:

S - Sanger Phred+33, raw reads typically (0, 40)
X - Solexa Solexa+64, raw reads typically (-5, 40)
I - Illumina 1.3+ Phred+64, raw reads typically (0, 40)
J - Illumina 1.5+ Phred+64, raw reads typically (3, 40)
 with 0=unused, 1=unused, 2=Read Segment Quality Control Indicator (**bold**)
 (Note: See discussion above).
L - Illumina 1.8+ Phred+33, raw reads typically (0, 41)

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%

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 Probability

> 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
,	44	0.9206	F	70	0.7992	K	75	0.9264	M	77	0.9499
-	45	0.9369	G	71	0.8337	L	76	0.9406	N	78	0.9602
			H	72	0.8632	M	77	0.9523	O	79	0.9684
			I	73	0.8882	N	78	0.9617	P	80	0.9749

Preprocessing and QC using FASTQC

FastQC provides a simple way to do QC checks on raw sequence data:

- ▶ Import of data from BAM, SAM or FastQ files
- ▶ Quick overview and summary graphs and tables to quickly assess your data
- ▶ Export of results to an HTML based permanent report
- ▶ Offline operation to allow automated generation of reports without running the interactive application

Preprocessing and QC using FASTQC

Interactive GUI from Amarel Desktop:

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

FastQC Example

Running FastQC from command-line (single file):

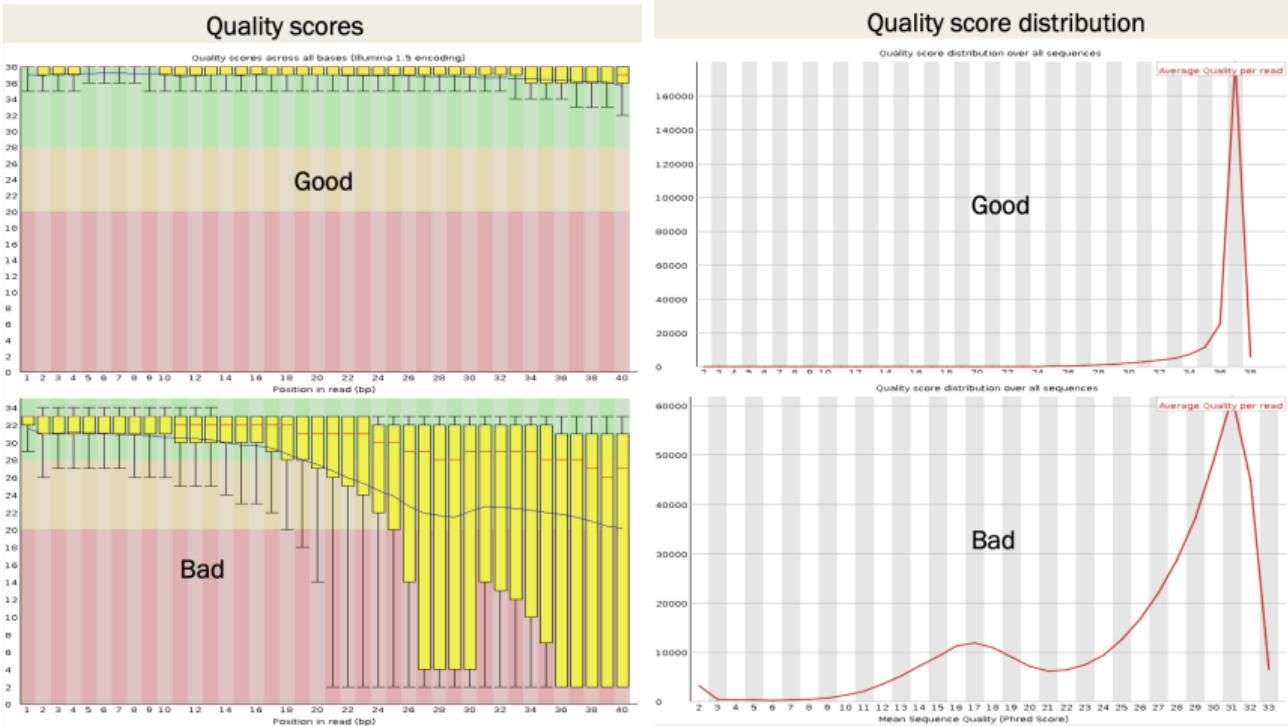
```
mkdir example_data/fastqc  
fastqc example_data/reads/R01_10_short500K.fq.gz \  
--outdir=example_data/fastqc
```

FastQC Example

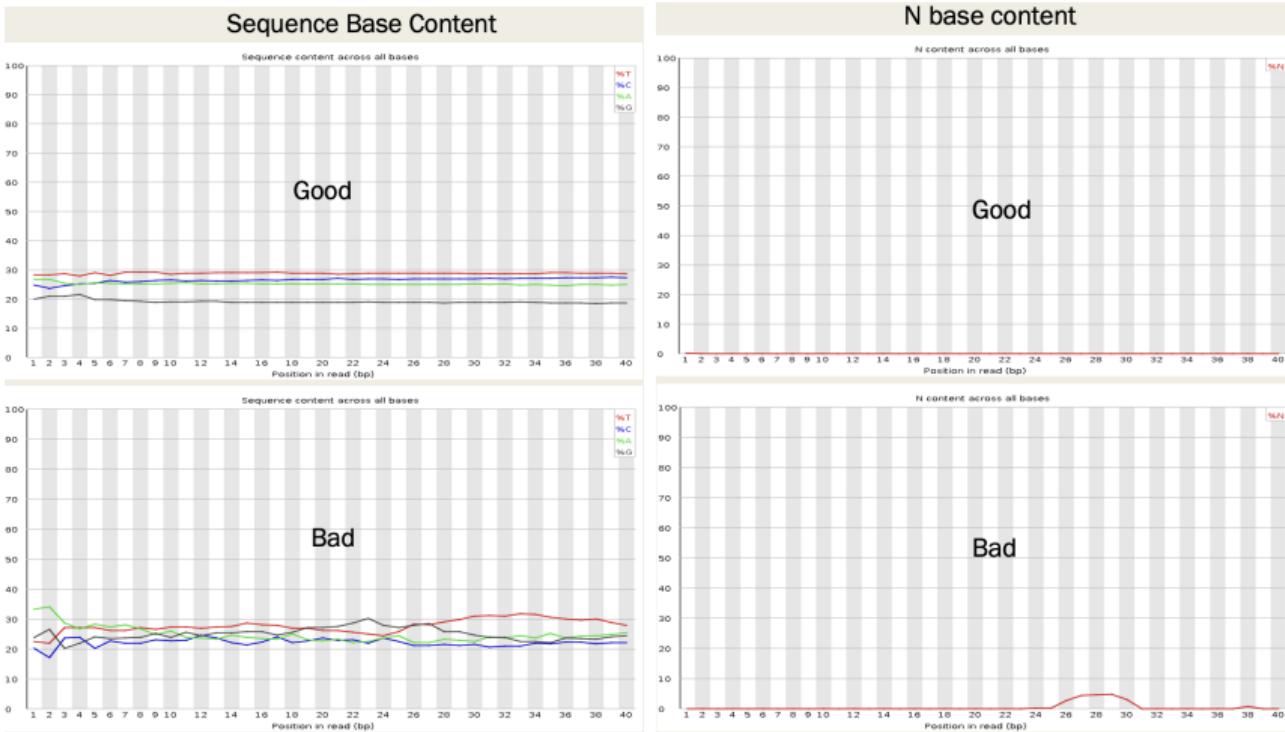
Running FastQC from command-line (multiple files):

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

FastQC Score Distribution



FastQC Base and N Distribution



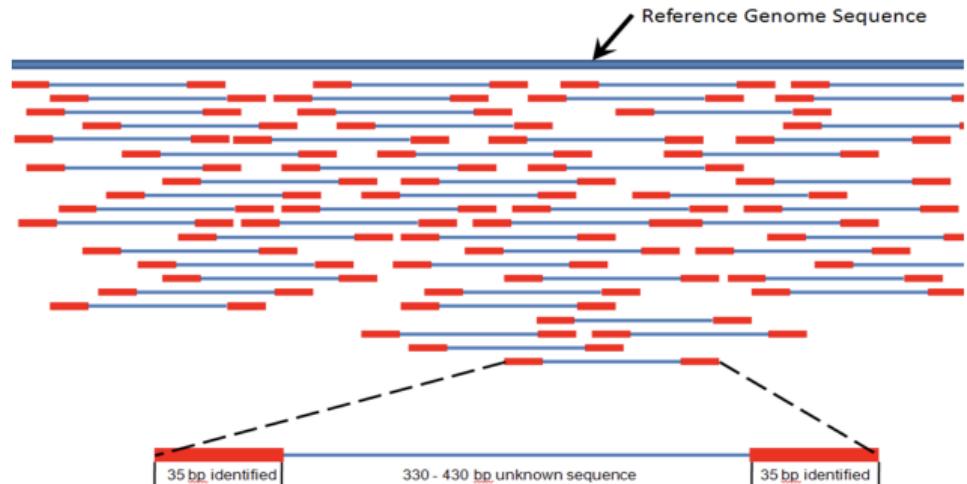
MultiQC Example

You can use multiqc to combine FastQC results. On Amarel Desktop do the following:

```
module load miniconda
pip install multiqc
cd example_data/fastqc
multiqc .
```

Alignment to the Reference Genome

Goal: Find the genomic location for the sequencing read. Software: Bowtie2, TopHat, STAR, Subread/Rsubread, many others!

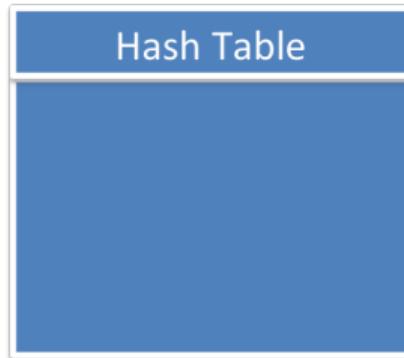


Indexing your genome

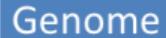
Abraham Lincoln: “Give me six hours to chop down a tree and I will spend the first four sharpening the axe.”

(4 hours indexing the genome, 2 hours aligning the reads)

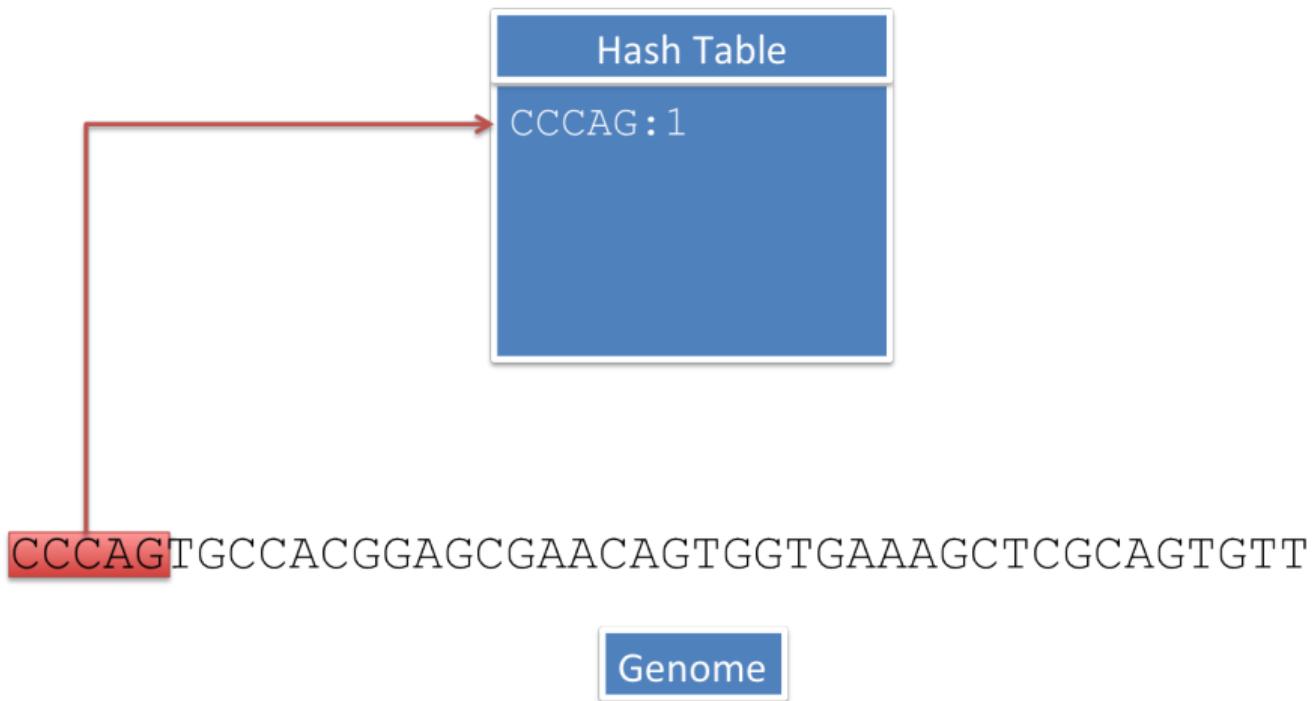
Creating a Hash Index Table



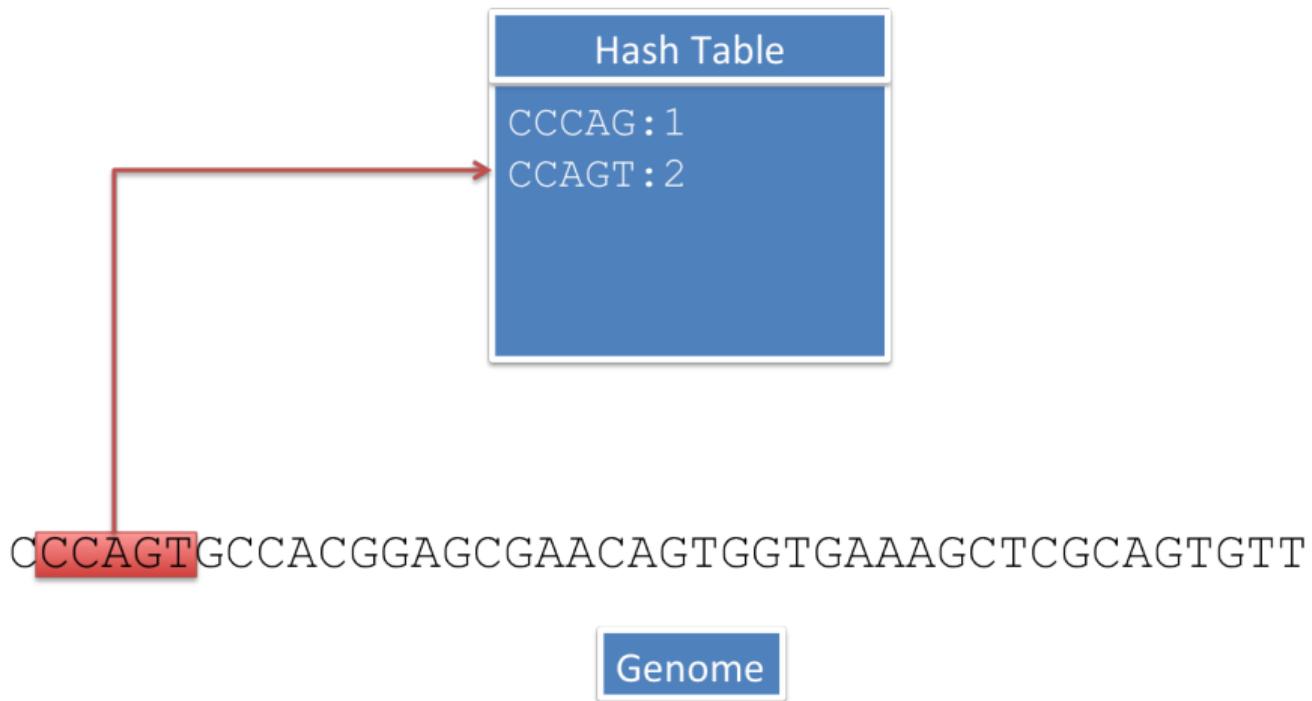
CCCAGTGCCACGGAGCGAACAGTGGTGAAAGCTCGCAGTGTT


Genome

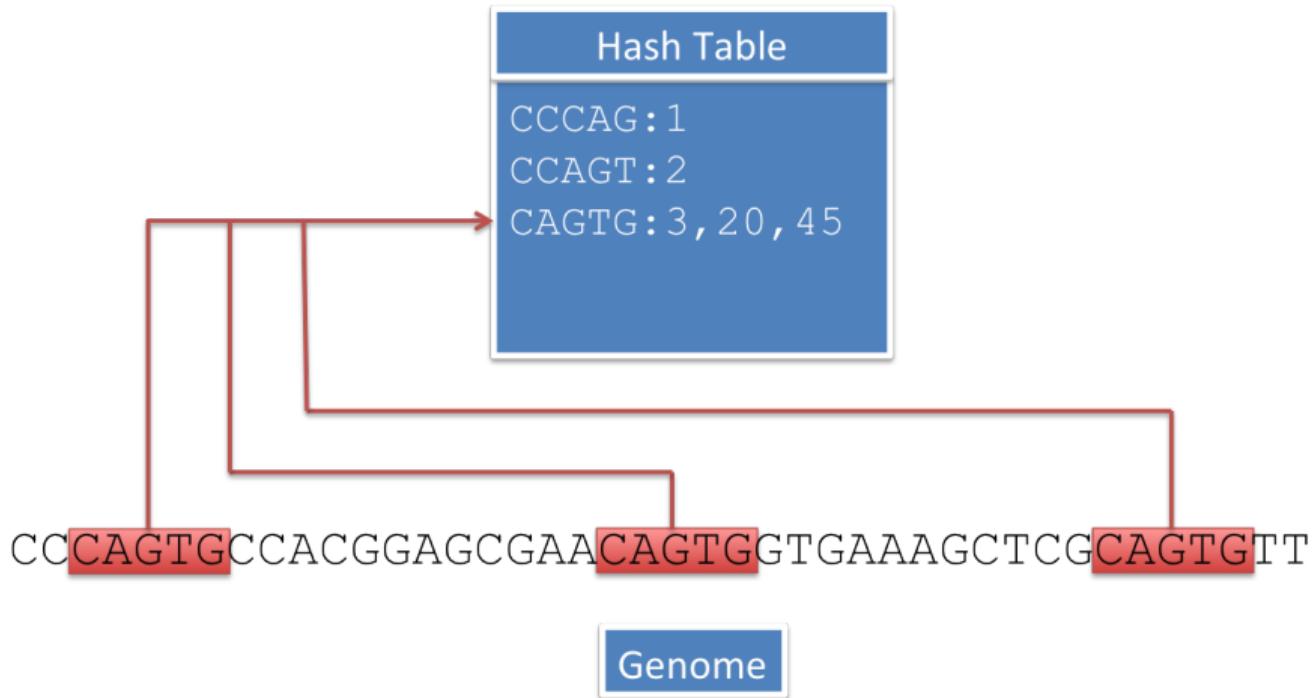
Creating a Hash Index Table



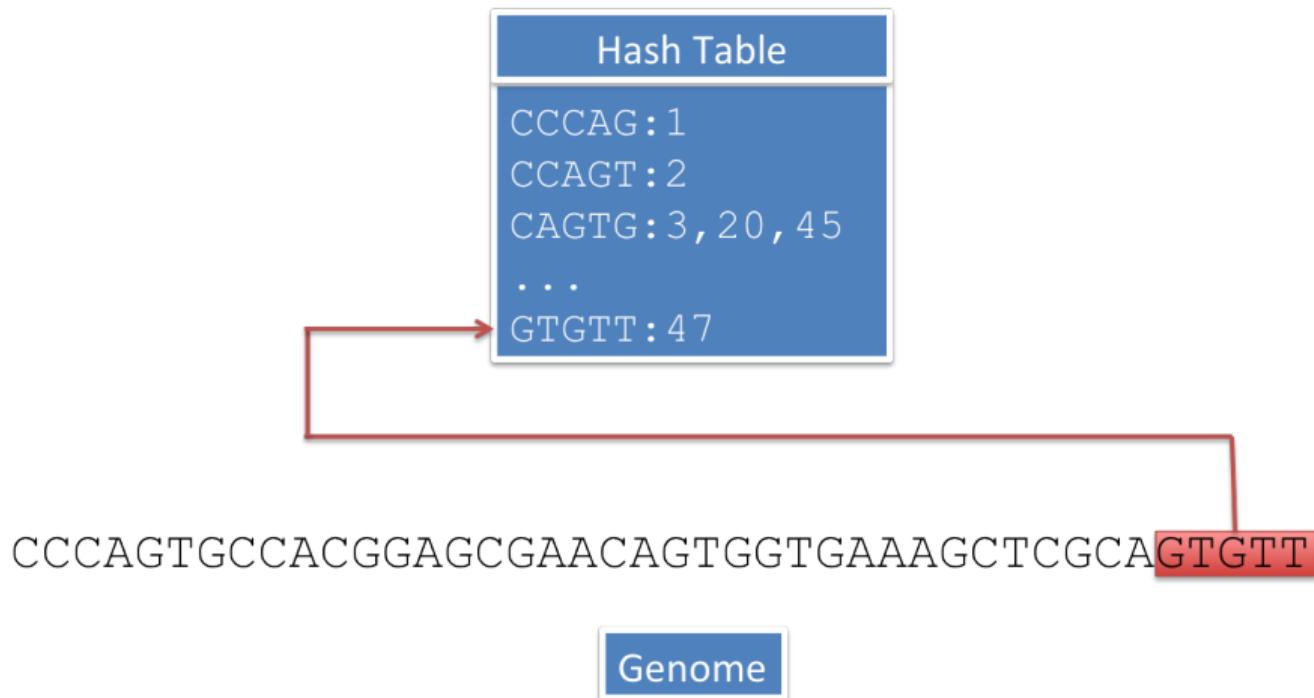
Creating a Hash Index Table



Creating a Hash Index Table



Creating a Hash Index Table



Read Lookup

Read

CAGTGCC

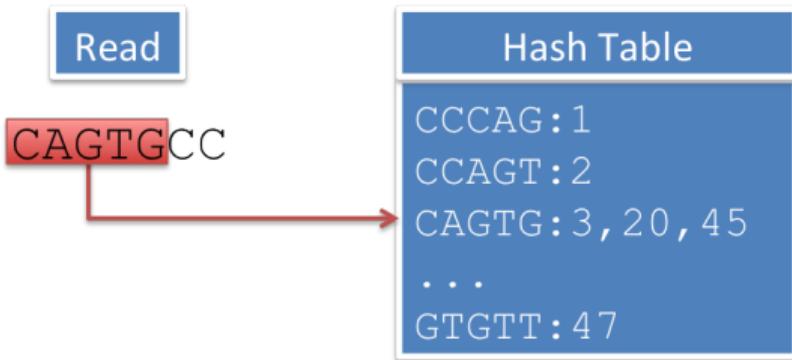
Hash Table

CCCAG:1
CCAGT:2
CAGTG:3, 20, 45
...
GTGTT:47

CCCAGTGCCACGGAGCGAACAGTGCTGAAAGCTCGCAGTGTT

Genome

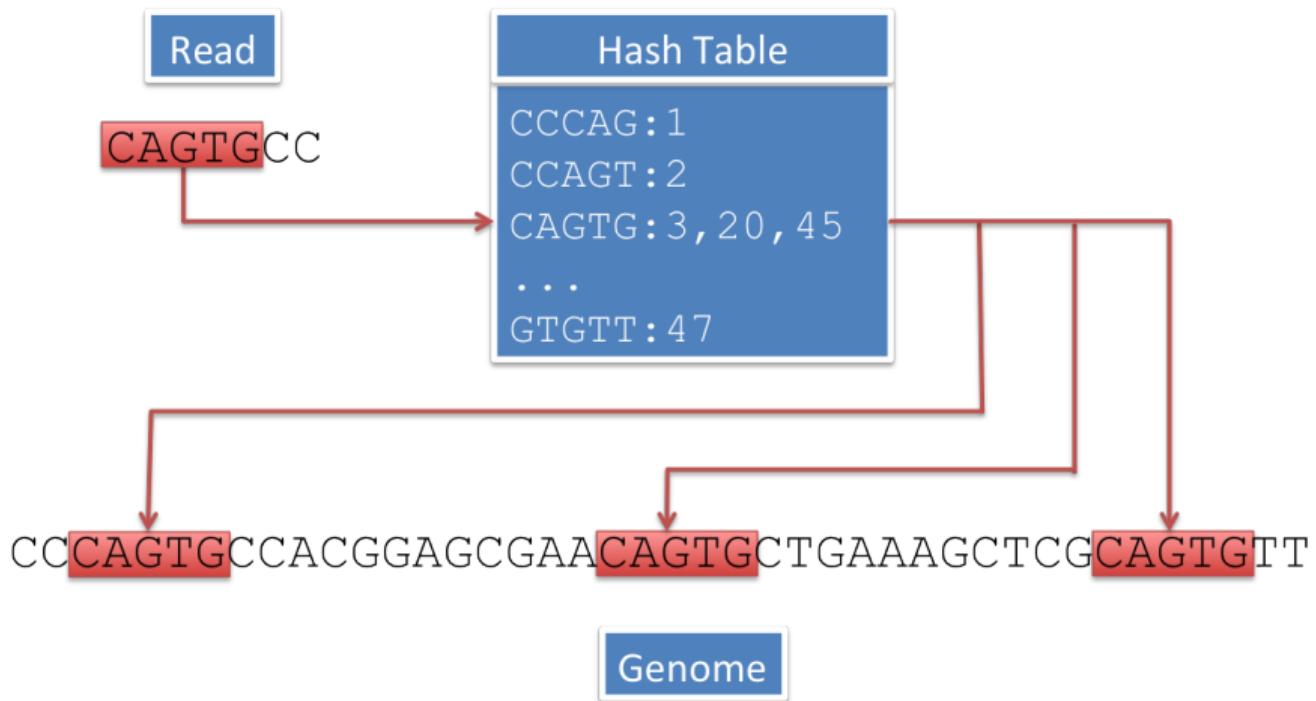
Read Lookup



CCCAGTGCCACGGAGCGAACAGTGCTGAAAGCTCGCAGTGTT

Genome

Read Lookup



Read Lookup

Read

CAGTGCC

Hash Table

CCCAG:1
CCAGT:2
CAGTG:3, 20, 45
...
GTGTT:47

CAGTGCC

|||||||

CC**CAGTGCC**ACGGAGCGAA**CAGTGCT**GAAAGCTCG**CAGTGT**T

Genome

Read Lookup

Read

CAGTGCC

Hash Table

CCCAG:1
CCAGT:2
CAGTG:3, 20, 45
...
GTGTT:47

CAGTGCC

|||||||

CC**CAGTGCC**ACGGAGCGAA**CAGTGCT**GAAAGCTCG**CAGTGT**
Score=7 **Score=5** **Score=3**

Genome

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	-4					
a	-6					
c	-8					
	-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				
c	-4					
a	-6					
c	-8					
	-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	1	-1	-3	-5	-7	-9
t	-4	-1	0	-2	-4	-4	-6
c	-6	-3	-2	1	-1	-3	-3
a	-8	-5	-2	-1	2	0	-2
c	-10	-7	-4	-1	0	1	1

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

Using Rsubread to do Alignment

The following userguide will be helpful for you:

<http://bioinf.wehi.edu.au/subread-package/SubreadUsersGuide.pdf>

Indexing your genome

Note that you will rarely do this for human alignment. You will usually download an existing index given to you by others who have already done this work. You will do this often if you are aligning microbial reads, e.g. MTB or some other organism for which others have not already made your index for you.

```
buildindex(basename="example_data/genome/ucsc.hg19.chr1_120-150M",  
           reference="example_data/genome/ucsc.hg19.chr1_120-150M.fasta.gz")
```

Took me ~0.2 minutes!

Aligning your reads:

Note that this outputs results in a .bam file and not a .sam file

```
align(index="rna_seq/genome/ucsc.hg19.chr1_120-150M",
      readfile1="rna_seq/reads/R01_10_short500K.fq.gz",
      output_file="rna_seq/alignments/R01_10_short.bam",
      nthreads=4)
```

My old laptop was an Apple M2, with 8 cores (used 4 cores), 24GB RAM:

- ▶ Took 15.7 minutes to align ~60M reads to the 30M bases
- ▶ Took 0.7 minutes to align ~6.5M reads to the 30M bases
- ▶ Took 0.3 minutes to align ~500K reads to the 30M bases

Aligning the reads using STAR

This afternoon we will use the STAR aligner on Amarel to align a set of RNA-seq fastq files!

Aligned Sequencing Data Formats (SAM and BAM)

Note that Rsubread outputs a .bam file (bam = binary alignment map) and not a .sam file (sam = sequence alignment map). Here is some information about a .sam file:
[https://en.wikipedia.org/wiki/SAM_\(file_format\)](https://en.wikipedia.org/wiki/SAM_(file_format))

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 segment
8	PNEXT	Int	[0,2 ²⁹ -1]	Position of the mate/next segment
9	TLEN	Int	[-2 ²⁹ +1,2 ²⁹ -1]	observed Template LENgth
10	SEQ	String	* [A-Za-z.=.]+	segment SEQuence
11	QUAL	String	[!-~]+	ASCII of Phred-scaled base QUALity+33

Aligned Sequencing Data Formats (SAM and BAM)

To convert .sam to .bam or vice versa, a package called Rsamtools. Using Rsamtools, you can convert bam to sam as follows:

```
asSam("rna_seq/alignments/R01_10_short.bam",  
      overwrite=T)
```

To convert to bam:

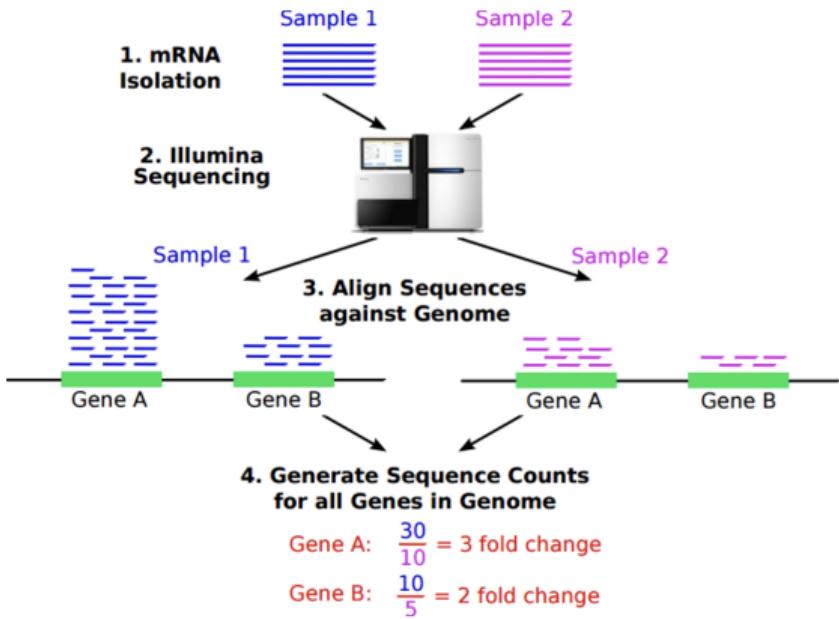
```
#asBam("rna_seq/alignments/R01_10_short.bam")
```

Feature counts

Now we can count reads hitting genes. Approaches/software:

- ▶ HT-Seq
- ▶ STAR
- ▶ Cufflinks
- ▶ RPKM FPKM or CPM
- ▶ RSEM
- ▶ edgeR
- ▶ findOverlaps (GenomicRanges)
- ▶ featureCounts (Rsubread)

Feature counts



Feature counts

```
fCountsList = featureCounts(  
  "rna_seq/alignments/R01_10_short.bam",  
  annot.ext="rna_seq_files/genome/genes.chr1_120-150M.gtf",  
  isGTFAnnotationFile=TRUE)  
  
featureCounts = cbind(fCountsList$annotation[,1],  
                      fCountsList$counts)  
  
write.table(featureCounts,  
            "rna_seq/alignments/R01_10_short.features.txt",  
            sep="\t", col.names=FALSE, row.names=FALSE, quote=FALSE)
```

Session info

```
sessionInfo()

## R version 4.5.1 (2025-06-13)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sequoia 15.5
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib; LAPACK version 3.12.1
##
## 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/New_York
## tzcode source: internal
##
## attached base packages:
## [1] stats4      stats       graphics    grDevices   utils       datasets    methods
## [8] base
##
## other attached packages:
## [1] SummarizedExperiment_1.38.1 Biobase_2.68.0
## [3] MatrixGenerics_1.20.0        matrixStats_1.5.0
## [5] Rsamtools_2.24.0            Biostrings_2.76.0
## [7] XVector_0.48.0              GenomicRanges_1.60.0
## [9] GenomeInfoDb_1.44.1          IRanges_2.42.0
## [11] S4Vectors_0.46.0            BiocGenerics_0.54.0
```