NGS Pipelines

NGS file manipulation, genome assembly, and pipelining

Kristine Lacek
Bioinformatics Scientist
US CDC – Influenza Division

Disclaimer

The findings and conclusions in this presentation are those of the authors and do not necessarily represent the official position of the Centers for Disease Control and Prevention.

Use of trade names and commercial sources is for identification only and does not imply endorsement by the U.S. Department of Health and Human Services.

References to non-CDC sites on the Internet do not constitute or imply endorsement of these organizations or their programs by CDC or the U.S. Department of Health and Human Services. CDC is not responsible for the content of pages found at these sites.

Module Key

Lecture Content

Practical Content

Bonus intermediate content

Module Objectives

- Understand NGS file formats
- Manipulate files with Bash

Module Objectives

- Understand NGS file formats
- Manipulate files with Bash
- Write a “wrapper” scripts

Module Objectives

- Understand NGS file formats
- Manipulate files with Bash
- Write a “wrapper” scripts
- Move fastqs through *Mira-nf* on command line
- Understand assembly and annotation files

Module Objectives

- Understand NGS file formats
- Manipulate files with Bash
- Write a “wrapper” scripts
- Move fastqs through *Mira-nf* on command line
- Understand assembly and annotation files
  - *.sam *.bam *.vcf
- View and interpret sequence alignments

Module Objectives

- Understand NGS file formats
- Manipulate files with Bash
- Write “wrapper” scripts
- Move fastqs through *Mira-nf* on command line
- Understand assembly and annotation files
  - *.sam *.bam *.vcf
- View and interpret sequence alignments
- Use samtools and bash one-liners to manipulate NGS files

Module Objectives

- Understand NGS file formats
- Manipulate files with Bash
- Write “wrapper” scripts
- Move fastqs through *Mira-nf* on command line
- Understand assembly and annotation files
  - *.sam *.bam *.vcf
- View and interpret sequence alignments
- Use samtools and bash one-liners to manipulate NGS files
- Install and use IGV to view bam files

NGS Review

NGS = Next Generation Sequencing

- High-throughput DNA/RNA sequencing technology
- Builds on earlier sequencing methods
  - Sanger sequencing: low throughput, one fragment at a time
- **Illumina** sequencing: paired-end short (150-300 bp) reads
- **Nanopore** and **Pacbio** sequencing: long-read (1000+) technology, single-end
- Fragmented reads come off sequencer unordered, need to be assembled to create a consensus genome

NGS Filetypes in a pipeline

Sequencers produce

Illumina

*.bcl

“binary call” file
Produced from images

Oxford Nanopore

*.pod5

“squiggle” file
electrical trace measures

NGS Filetypes in a pipeline

Sequencers produce → Base-calling produces

Illumina

*.bcl

“binary call” file
Produced from images

*.fastq
*.fastq.gz

Oxford Nanopore

*.pod5

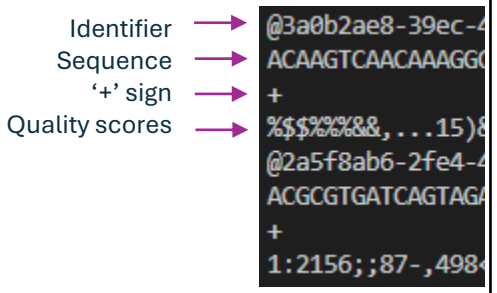
“squiggle” file
electrical trace measures

Identifier →
Sequence →
‘+’ sign →
Quality scores →

```
@3a0b2ae8-39ec-4e79-a736-42ceda08d577 runid=ae9c779a681221c12f758a169576e4e6e7ad80c6 read=2460 ch=3
ACAAGTCAACAAAGGCAGGAGTGGCGAAAACATAATGGAGATCAACACCATGGCAAAACTTTCAGGTAGACTGTTTTCTTTGGCATGTTCGCAAGCGATT
+
%$$%%%&&,...15)&%&'&&&)''%%'1.400024?4310001/2118-,,&%%&'.00336?>>=7766442/-,,(&&&&,//'&&')((&')-/68
@2a5f8ab6-2fe4-4a6c-9753-5fe5180b0092 runid=ae9c779a681221c12f758a169576e4e6e7ad80c6 read=3000 ch=11
ACGCGTGATCAGTAGAAACAAGGAATACAATAAGAAACACAAAGCAAACATGGGCTCCACAAAGCTAAATACAAACGTTGGGTGAGGGTTTCTGTTCTTA
+
1:2156;;87-,498<=;;99(((,-+++(((((33889?{<@C??C/...=;:99;88999(,/*(&%%&&),,+-0766,+,20000;@F?>>?ABB>
```

NGS Filetypes in a pipeline

Sequencers produce	Base-calling produces	Demultiplexing produces
Illumina *.bcl “binary call” file Produced from images	*.fastq *.fastq.gz	<sample1>.fastq <sample2>.fastq <sample3>.fastq <sampleN>.fastq
Oxford Nanopore *.pod5 “squiggle” file electrical trace measures		



NGS Filetypes in a pipeline

Sequencers produce

Illumina

*.bcl

“binary call” file
Produced from images

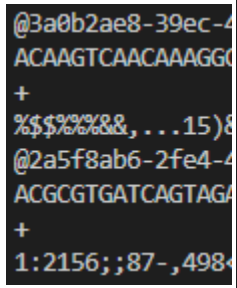
Oxford Nanopore

*.pod5

“squiggle” file
electrical trace measures

Base-calling produces

*.fastq
*.fastq.gz



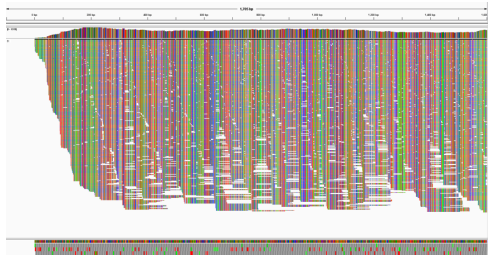
Demultiplexing produces

<sample1>.fastq
<sample2>.fastq
<sample3>.fastq
<sampleN>.fastq

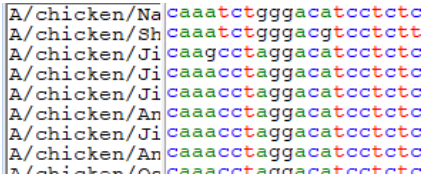
Read QC

Assembly/Mapping produces

<sample1>.bam
<sampleN>.bam



<sample1>.fasta
<sampleN>.fasta



NGS Filetypes in a pipeline

Sequencers produce

Illumina

*.bcl

“binary call” file
Produced from images

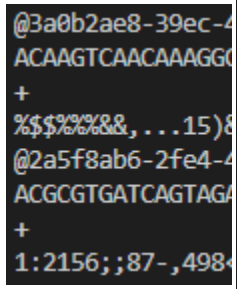
Oxford Nanopore

*.pod5

“squiggle” file
electrical trace measures

Base-calling produces

*.fastq
*.fastq.gz



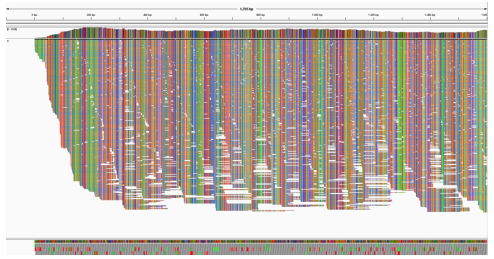
Demultiplexing produces

<sample1>.fastq
<sample2>.fastq
<sample3>.fastq
<sampleN>.fastq

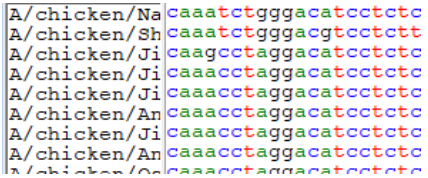
Read QC

Assembly/Mapping produces

<sample1>.bam
<sampleN>.bam



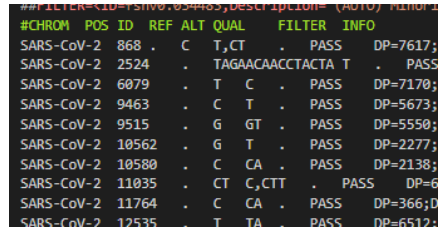
<sample1>.fasta
<sampleN>.fasta



Consensus QC

Variant calling produces

<sample1>.vcf
<sampleN>.vcf



Phylogenetic analysis produces

<sample1>.nwk

```
(,,(,));
(A,B,(C,D));
(A,B,(C,D)E)F;
(:0.1,:0.2,(:0.3,:0.4):0.5);
(:0.1,:0.2,(:0.3,:0.4):0.5):0.0;
(A:0.1,B:0.2,(C:0.3,D:0.4):0.5);
(A:0.1,B:0.2,(C:0.3,D:0.4)E:0.5)F;
((B:0.2,(C:0.3,D:0.4)E:0.5)F:0.1)A;
```

Other filetypes

- **HTML:** HyperText Markup Language
  - Creates web page structure and content using tags
  - Common in reports
- **YAML:** Yet Another Markup Language → YAML Ain't Markup Language
  - human-readable data serialization format using indentation
  - Common in configurations
- **MD:** Markdown
  - lightweight markup language for formatting plain text documents
  - common in documentation, wikis
  - easiest to read and learn
- **JSON:** JavaScript Object Notation
  - structured data format using key-value pairs in text
  - common in figure creation, configs
- **XML:** eXtensible Markup Language
  - structured data format using custom tags and hierarchical elements
  - NCBI submissions are full of xml files

.bcl and .fast5

BCL Files (Illumina)

- Raw output from Illumina sequencers
  - Generated directly by the instrument during the run
- Contain base calls and quality scores
  - One file per cycle, per lane
- Not human-readable
  - Must be converted before analysis
- Converted to FASTQ
  - Using tools like bcl2fastq

FAST5 Files (Oxford Nanopore)

- Raw signal-level data
  - Electrical current measurements from nanopores
- Contain rich metadata
  - Timing, channel information, and raw signals
- Used for basecalling and re-analysis
  - Allows re-basecalling as algorithms improve
- Converted to FASTQ
  - Using tools like Guppy or Dorado

Fastq files

- Text-based format for sequencing reads
  - Stores sequence and quality information together
- Each read uses four lines
  - Line 1: Read identifier (@)
  - Line 2: Nucleotide sequence
  - Line 3: Separator (+)
  - Line 4: Quality scores (ASCII characters)
- One quality score per base: same length as the sequence line
- Commonly compressed
  - .fastq.gz to reduce file size

```
@M02989:30:000000000-L4R3N:1:1101:17861:2019 1:N:0:GTTGCGTTCA+GTTGCAGTTG
ATATGATGCCGTTGCTACTACACATTCCTGGATTCCCAAGAGGAATCGTTCTATTCTCAACACAAGCCAAAGGGGAATTCTTGAGGATGAACAGATGTACCAGAAGTGTTGCAATTTATTCGAGAAATTTTTCCCTAGTAGTTCATATAG
+
>A1A1@3@BC?AE1AGFFGFGGH1GHFHHB00GGFGFD0F00FFFHGFHHHGDGHHHHFBG1F/CFGGBF0FB??C/FGHHHEBFBBFGFFGBBGFHHHHGEE0CCGFHHHHFHHHHGHHHHBGFEHHHHHHGFGHHFHHBGGHHHHHHH
@M02989:30:000000000-L4R3N:1:1101:16297:2020 1:N:0:GTTGCGTTCA+GTTGCAGTTG
GCAATATCCTCAAAGGAAAATTTCAAACAGCTGCTCAGAGGGCAATGATGGATCAAGTAAGGGAAAGCCGAAACCCAGGAAACGCTGAGATTGAAGACCTCATTTTCCTGGCGCGGTCAGCACTCATTCTGAGAGGATCAGTTGCACATA
+
AAAAAFFFFFFFGFGGFFGFGGHHGF5FGBGHGGHHFCBCD?EEGGGBGHEFHGABFHHHHEFEAHHFHGCFGFHGGHFGHEHFGGGGGFHHHGHGFGHHHFHHHHHHHHHHHEGGGGGGHFHHHHHHHHHFFFGFGHHHHFHHHHHHHH
@M02989:30:000000000-L4R3N:1:1101:19536:2080 1:N:0:GTTGCGTTCA+GTTGCAGTTG
TGGTAACACGGTTCGAGTCTCTGAAAATATACAGAGATTCGCTTGGAGAAGCTGTGATGAGAATGGGAGACCTTCACTACCTCCAGAGCAGAAACTGTCTCTTATACACATCTCCGAGCCCACGAGACGTTGCGTTCAATCTCGTATGCC
+
AAAAAFFFFB@BGGCGFGGGGGFFHHFHGHCHFH53FGGHGGGHGGEGFHGHHEHHHHHGHHHHHGGAFFGGHHHHHHHHFHHGHHGHFHE?GHGHGHGHHHHHHGHGFHHHGHHGG@E@EEGECEDADFGHGGGGGHD2FGGHGGGFHH
```

Fastq files

Phred Quality Scores

- Measure base-calling confidence
  - Higher score = higher confidence
- Logarithmic scale
  - Q = -10 log10(Perror)
- Encoded as ASCII characters
  - Stored in line 4 of FASTQ
- Common interpretation
  - Q20 ≈ 1% error
  - Q30 ≈ 0.1% error
  - Q40 ≈ 0.01% error

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

```
@M02989:30:000000000-L4R3N:1:1101:17861:2019 1:N:0:GTTGCGTTCA+GTTGCAGTTG
ATATGATGCCGTTGCTACTACACATTCCTGGATTCCCAAGAGGAATCGTTCTATTCTCAACACAAGCCAAAGGGGAATTCTTGAGGATGAACAGATGTACCAGAAGTGTTGCAATTTATTCGAGAAATTTTTCCCTAGTAGTTCATATAG
+
>A1A1@3@BC?AE1AGFFGFGGH1GHFHHB00GGFGFD0F00FFFHGFHHHGDGHHHHFBG1F/CFGGBF0FB??C/FGHHHEBFBBFGFFGBBGFHHHHGEE0CCGFHHHHFHHHHGHHHHBGFEHHHHHHGFGHHFHHBGGHHHHHHH
@M02989:30:000000000-L4R3N:1:1101:16297:2020 1:N:0:GTTGCGTTCA+GTTGCAGTTG
GCAATATCCTCAAAGGAAAATTTCAAACAGCTGCTCAGAGGGCAATGATGGATCAAGTAAGGGAAAGCCGAAACCCAGGAAACGCTGAGATTGAAGACCTCATTTTCCTGGCGCGGTCAGCACTCATTCTGAGAGGATCAGTTGCACATA
+
AAAAAFFFFFFFGFGGFFGFGGHHGF5FGBGHGGHHFCBCD?EEGGGBGHEFHGABFHHHHEFEAHHFHGCFGFHGGHFGHEHFGGGGGFHHHGHGFGHHHFHHHHHHHHHHHEGGGGGGHFHHHHHHHHHFFFGFGHHHHFHHHHHHHHH
@M02989:30:000000000-L4R3N:1:1101:19536:2080 1:N:0:GTTGCGTTCA+GTTGCAGTTG
TGGTAACACGGTTCGAGTCTCTGAAAATATACAGAGATTCGCTTGGAGAAGCTGTGATGAGAATGGGAGACCTTCACTACCTCCAGAGCAGAAACTGTCTCTTATACACATCTCCGAGCCCACGAGACGTTGCGTTCAATCTCGTATGCC
+
AAAAAFFFFB@BGGCGFGGGGGFFHHFHGHCHFH53FGGHGGGHGGEGFHGHHEHHHHHGHHHHHGGAFFGGHHHHHHHHFHHGHHGHFHE?GHGHGHGHHHHHHGHGFHHHGHHGG@E@EEGECEDADFGHGGGGGHD2FGGHGGGFHH
```

Fasta files

- Text-based sequence format
  - Stores DNA, RNA, or protein sequences
- Two-line (or multi-line) structure
  - Header line begins with >
  - Sequence lines follow
  - Multi-line structure called a multifasta
- No quality scores
  - Sequence only (unlike FASTQ)
- Consensus genome
- Indexed with .fai files: Enables fast random access to sequences by position

```
>30325630002_N90M7FCC_v1 | A_HA_H1
ATGAAGGCAATACTAGTGGTTATGCTGTATACATTTACAACCGCAAATGCAGACACATTATGTATAGGTTATCATGCGAACAATTCAACAGACACTGTGGACACAGTACTAGAAAAGAATGTAACAGTAACACA
CTCTGTCAATCTTCTAGAAGACAAGCATAACGGAAAACTATGCAAACTAAGAGGGGTAGCCCCATTGCATTTGGGTCAATGTAACATTGCTGGCTGGATCCTGGGAAATCCAGAGTGTGAATCACTATCCACAG
CAAGGTCATGGTCCTACATTGTGGAAACATCTAATTCAGACAATGGAACGTGTTACCCAGGAGATTTCATCAATTATGAGGAGCTAAGAGAGCAATTGAGCTCAGTGTCATCATTTGAAAAGTTTGAAATATTC
CCCAAGGCAAGTTCATGGCCTAATCATGACTCGGACAATGGTGTAACGGCAGCATGTTCTCACGATGGAGCAAGAAGCTTCTACAAAAACTTGATATGGCTGGTTAAAAAAGGAAAATCGTACCCAAAGATCAA
CCAAACCTACATTAATGACAAAGGGAAAGAAGTCCTCGTGCTGTGGGGCATTCACCATCCACCCACTATTACTGACCAAGAAAGTCTCTATCAGAATGCAGATGCGTATGTTTTTGTGGGGACATCAAGATACA
GCAAGAAGTTCAAGCCGGAAATAGCAGCAAGACCCAAAGTGAGGGATCGAGCAGGGAGAATGAACTATTACTGGACACTAGTAGAACCGGGAGACAAAATAACATTCGAAGCAACTGGTAATCTAGTGGCACCG
AGGTATGCATTCACAATGGAAAAAGAAGCTGGATCTGGTATTATCATTTCAGATACACCAGTCCACGATTGCAATGCAACTTGTCAGACACCCAAGGGTGCTATAAACACCAGCCTCCCATTTCAAAATGTACA
TCCGATCACGATTGGGGAATGTCCAAAGTATGTAAGAAGCACAAAACTGAGGCTGGCCACAGGATTGAGGAATGTCCCGTCTATTCAATCTAGAGGCCTATTCGGGGCCATTGCTGGCTTCATCGAAGGGGGGT
GGACAGGAATGGTTGATGGATGGTACGGTTATCACCATCAAAATGATCAGGGATCAGGATATGCAGCCGATCTGAAGAGCACACAAAATGCCGTTGATAAGATTACCAACAAAGTAAATTCTGTTATTGAAAAG
ATGAATACACAGTTCACAGCAGTTGGTAAAGAGTTCAACCACCTTGAAAAAAGAATAGAAAATCTAAATAAAAAGGTTGATGATGGTTTCCTGGACGTTTGGACTTACAATGCCGAACTGCTGGTTCTACTGGA
AAATGAAAGAACTTTGGACTATCACGATTCAAATGTGAAGAACTTGTATGAAAAAGTAAGACACCAGTTAAAAAACAATGCCAAGGAAACTGGAAACGGCTGCTTTGAATTTTACCACAAATGCGACAACACAT
GCATGGAAAGTGTCAAGAATGGAACTTATGACTACCCAAAATACTCAGAGGAAGCAAAATTGAACAGAGAAAAAATAGATGGAGTAAAGCTGGACTCAACAAGGATTTACCAGATTTTGGCAATCTATTCAACT
GCTGCCAGTTCATTGGTACTGGTAGTCTCCCTGGGGGCAATCAGCTTCTGGATGTGCTCTAATGGGTCTCTACAGTGCAGAATATGTATTTAA
>30325630002_N90M7FCC_v1 | A_NP
ATGGCGTCTCAAGGCACCAAACGATCATATGAACAAATGGATACTGGTGGGGAGCGCCAGGATGCCACAGAAATCAGAGCATCTGTTGGAAGAATGGTTGGTGGAATCGGGAGATTCTACATCCAAATGTGTAC
TGAACTAAAACTCAGTGATTATGATGGACGACTAATTCAGAACAGCATAACAATAGAGAGGATGGTGCTTTCTGCTTTTGATGAGAGAAGAAATAAATACCTAGAAGAGCATCCAAGTGCTGGGAAGGACCCTA
AGAAAACAGGAGGACCCATCTATAGAAGAATAGACGGAAAATGGACAAGAGAACTCATCCTTTATGACAAAGAAGAAATAAGGAGAGTTTGGCGCCAAGCAAACAATGGCGAAGATGCAACGGCAGGTCTTACC
CATCTCATGATTTGGCATTCCAATCTGAATGATGCCACATATCAGAGGACAAGAGCACTTGTCCGCACTGGAATGGATCCCAGAATGTGCTCTCTAATGCAAGGTTCAACACTTCCCAGGAGGTCTGGTGCCGC
AGGTGCTGCAGTAAAAGGAGTTGGAACAATAGCTATGGAGTTAATCAGAATGATAAAACGTGGAATCAATGACCGAAATTTCTGGAGGGGTGAAAATGGACGAAGGACAAGAATTGCTTATGAAAGAATGTGCA
ATATCCTCAAAGGAAAATTCCAAACAGCTGCCCAGAGGGCAATGATGGATCAAGTAAGAGAAAGTCGAAACCCAGGAAACGCTGAAATTGAAGACCTCATTTTCCTGGCACGGTCAGCACTCATTCTGAGGGGA
TCAATTGCACATAAATCCTGCCTACCTGCTTGTGTGTATGGGCTTGCAGTAGCAAGTGGCCATGACTTTGAGAGGGAAGGGTACTCACTGGTCGGAATAGACCCATTCAAATTACTCCAAAACAGTCAAGTGGT
CAGCCTGATGAGACCAAATGAAAATCCAGCTCACAAGAGTCAATTGGTATGGATGGCATGCCACTCCGCTGCATTTGAAGATTTGAGAGTATCAAGTTTCATAAGGGGAAAGAAAGTGATCCCAAGAGGAAAGC
TTTCCACAAGAGGGGTTCAGATTGCTTCAAATGAGAATGTGGAAACCATGGACTCCAACACTCTGGAACTAAGAAGCAGATACTGGGCCATAAGAACCAGGAGTGGAGGAAATACCAATCAACAGAAGGCATCT
GCAGGCCAGATCAGTGTGCAGCCTACATTCTCAGTGCAGCGAAATCTCCCTTTTGAAAGAGCAACCATTATGGCAGCATTCAGCGGAAACAATGAAGGACGGACATCCGACATGCGAACAGAAGTTATAAGAAT
GATGGAAAGTGCAAAGCCAGAGGATTTGTCCTTCCAGGGGCGGGGAGTCTTCGAGCTCTCGGACGAAAAGGCAACGAACCCGATCGTGCCTTCCTTTGACATGAGTAATGAGGGGTCTTATTTCTTCGGAGACA
ATGCAGAGGAGTATGACAATTGA
>30325630002_N90M7FCC_v1 | A_NA_N1
ATGAATCCAAACCAAAAGATAATAACCATTGGTTCTATCTGTATGACAATTGGAACGGCTAACTTAATATTACAAATTGGAAACATAATCTCAATATGGGTTAGCCACTCAATTCAAATTGGAAATCAAAGCCA
GATTGAAACATGCAATAAAAGCGTCATTACTTATGAAAACAACACTTGGGTAAATCAGACATTTGTTAACATCAGCAACACTAACTCTGCTGCTAGACAGTCAGTGGCTTCCGTGAAATTAGCGGGCAATTCCT
CTCTCTGCTCTGTTAGTGGATGGGCTATATACAGTAAAGACAACAGTGTAAGAATCGGTTCCAAGGGGGATGTGTTTGTCATAAGGGAACCATTCATATCATGCTCTCCCTTGGAGTGCAGAACCTTCTTCTTG
ACCCAAGGGGCTTTGCTAAATGACAAACATTCCAATGGAACAATTAAAGACAGAAGCCCATATCGAACCCTAATGAGCTGTCCTATTGGTGAAGTTCCCTCTCCATACAACTCAAGATTTGAGTCAGTCGCTTG
GTCAGCAAGTGCTTGTCATGATGGCACCAATTGGCTAACAATTGGAATTTCTGGCCCAGACAATGGGGCAGTGGCTGTGTTAAAATACAATGGCATAATAACAGACACTATCAAAAGTTGGAGGAACAAGATAT
TGAGAACACAAGAGTCTGAATGTGCATGTGTAAATGGTTCTTGCTTTACAATAATGACCGATGGACCAAATGATGGACAGGCCTCATACAAAATCTTCAGAATAGAGAAGGGAAAGATAACCAAATCAGTCGAA
ATGAAGGCCCCTAATTATCACTATGAAGAATGCTCCTGTTACCCTGATTCTAGTGAAATCACATGTGTGTGCAGGGATAATTGGCATGGCTCGAATCGACCTTGGGTGTCTTTCAATCAGAATCTGGAATATCA
GATAGGATACATATGCAGTGGGGTTTTCGGAGACAATCCACGCCCTAATGATAAGACAGGCAGTTGTGGCCCAGTATCGTCTAATGGAGCAAATGGGGTAAAGGGATTTTCATTCAAATACGGCAATGGTGTTT
GGATAGGGAGAACTAAAAGCATTAGTTCAAGAAAAGGTTTTGAGATGATTTGGGATCCGAATGGATGGACTGGGACTGACAATAAATTCTCAAAAAAGCAAGATATTGTAGGGATAAATGAGTGGTCAGGGTAT
AGCGGGAGTTTTGTCCAGCATCCAGAACTAACAGGGCTGAATTGTATAAGACCTTGCTTCTGGGTTGAACTAATAAGAGGACGACCCGAAGAGAACACAATCTGGACTAGCGGGAGCAGCATATCCTTTTGTGG
TGTAGACAGTGACATTATGGGTTGGTCTTGGCCAGACGGTGCTGAGTTGCCATTCACCATTGACAATTAA
>30325630002_N90M7FCC_v1 | A_MP
ATGAGTCTTCTAACCGAGGTCGAAACGTACGTTCTTTCTATAATCCCGTCAGGCCCCCTCAAAGCCGAGATCGCACAGAGACTGGAAAGTGTCTTTGCAGGAAAGAACACAGATCTTGAGGCTATCATGGAATG
GCTAAAGACAAGACCAATCTTGTCACCTCTGACTAAGGGAATTTTAGGATTTGTGTTCACGCTCACCGTGCCCAGTGAGCGAGGACTGCAGCGTAGACGCTTTATCCAAAATGCCCTAAATGGAAATGGGGACC
CGAACAACATGGATAGAGCAGTTAGACTATACAAGAAACTCAAAAGAGAAATAACGTTCCATGGGGCCAAAGAAGTGTCACTAAGCTATTCAACTGGTGCACTTGCAAGTTGCATGGGCCTTATATACAACAGG
ATGGGAACAGTGACCACAGAAGCTGCTTTCGGTCTAGTTTGTGCCACTTGTGAACAGATTGCTGATTCACAGCATCGGTCTCACAGACAAATGGCTACTACCACAAATCCACTAATCAGGCATGAAAACAGAAT
GGTGCTGGCTAGCACTACGGCAAAGGCTATGGAACAGGTGGCTGGATCGAGTGAACAGGCAGCGGAGGCCATGGAGGTTGCTAATAAGACTAGGCAGATGGTACATGCAATGAGAACTATTGGAACTCATCCTA
GCTCCAGTGCTGGTCTAAGAGATGACCTTCTTGAAAATTTACAGGCCTACCAGAAGCGAATGGGAGTGCAGATGCAGCGGTTCAAATGATCCTCTCGTCATTGCAGCAAACATCATTGGGATCTTGCACCTGAT
ATTGTGGATTACTGATCGTCTTTTTTTCAAATGCATTTATCGTCGCTTTAAATACGGTTTGAAAAGAGGGCCTTCTACGGAAGGAGTGCCTGAGTCCATGAGGGAAGAATATCAACAGGAGCAGCAGAGTGCTG
TGGATGTTGACGATGGTCATTTTGTCAACATAGAGCTAGAGTAA
```

Fastq practical

Make a folder in your home (~) called MIRA_NGS. Then make a subdirectory called ”day3”
https://www.ncbi.nlm.nih.gov/sra?linkname=bioproject_sra_all&from_uid=1437047

1. Using sra-toolkit and a loop, pull down the following samples from SRA

SRR accession	Sample ID
SRR37675411	f58cd412
SRR37675412	f133a406
SRR37675413	e91ea59e

```
fastq-dump --split-3 --defline-seq '@$sn\$ri' --defline-qual '+'
<srr#>
```

2. Rename the fastqs according to the sample ID mapping (optional loop again here), and _R1 _R2

3. Gunzip all fastqs
4. Move to a subfolder called “fastqs”

Fastq practical

Make a folder in your home (~) called MIRA_NGS. Then make a subdirectory called ”day3”

1. Using sra-toolkit and a loop, pull down the following samples from SRA

SRR accession	Sample ID
SRR37675411	f58cd412
SRR37675412	f133a406
SRR37675413	e91ea59e

```
mkdir ~/MIRA_NGS

mkdir ~/MIRA_NGS/day3

cd ~/MIRA_NGS/day3
```

Possible solution:

```
 for i in SRR37675411 SRR37675412 SRR37675413; do fastq-dump --split-3 --
defline-seq '@$sn/$ri' --defline-qual '+' $i ; done
```

Fastq practical

2. Rename the fastqs according to the sample ID mapping (optional loop again here)
Possible solution
Make accessions.txt

```
$ cat accessions.txt
SRR37675411, f58cd412
SRR37675412, f133a406
SRR37675413, e91ea59e
```

```
 $ for i in $(cat accessions.txt); do mv $(echo $i | cut -f1 -d,)_1.fastq
$(echo $i | cut -f2 -d,)_R1.fastq && mv $(echo $i | cut -f1 -d,)_2.fastq
$(echo $i | cut -f2 -d,)_R2.fastq; done
```

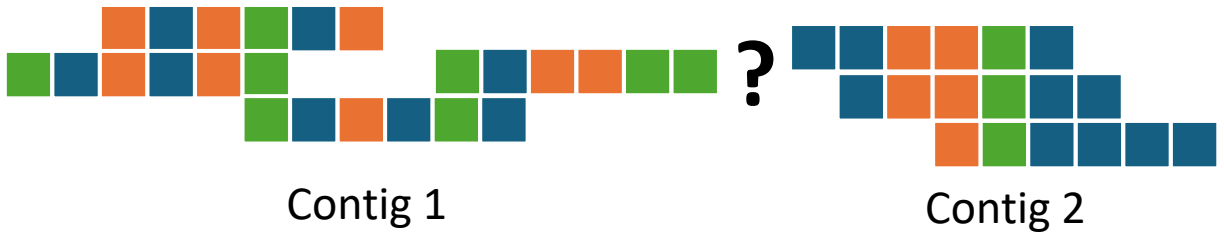
3. Gunzip all fastqs

```
gzip *.fastq
```

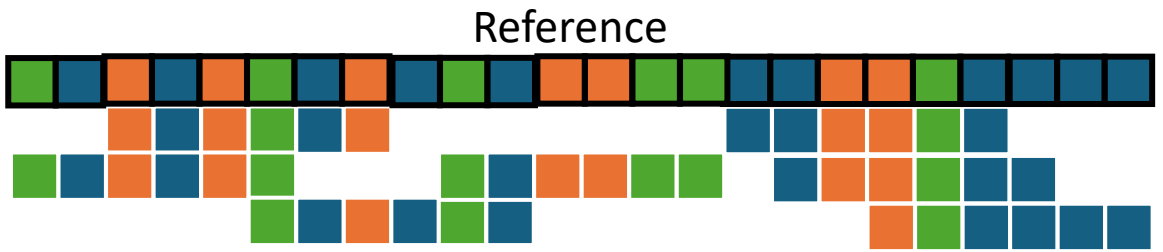
Genome Assembly Review

There are two classes of genome assembly from NGS data

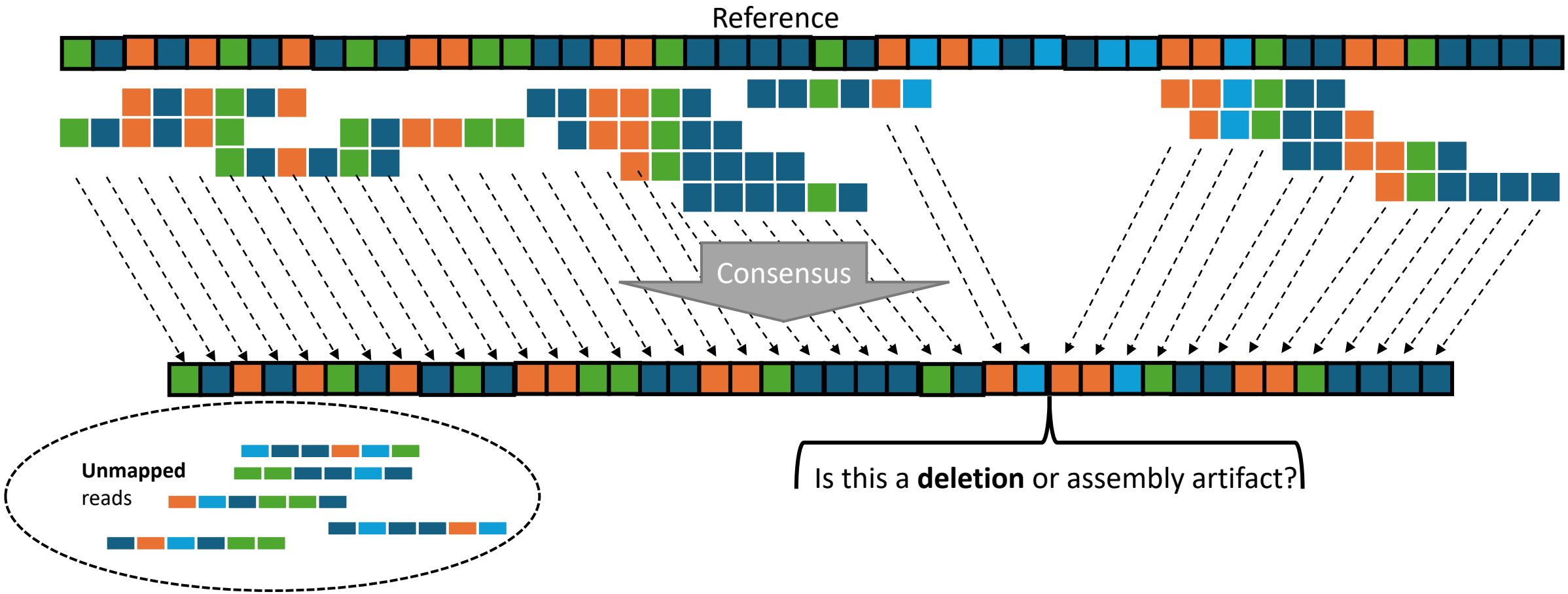
- ***De novo***
  - Latin for “from the new”
  - NO Reference sequence used
  - Results in **contigs** (contiguous sequences)
  - Requires a **scaffold** to put **contigs** into order across repetitive regions
  - Mostly used for metagenomics and assembling genomes that have no reference



- **Reference-based**
  - Individual reads are **mapped** directly to the position of the reference genome that they **align** the best
  - Disadvantage is that pathogens that evolve rapidly and are highly variable may not assemble if a close enough relative sequence is not used as the reference

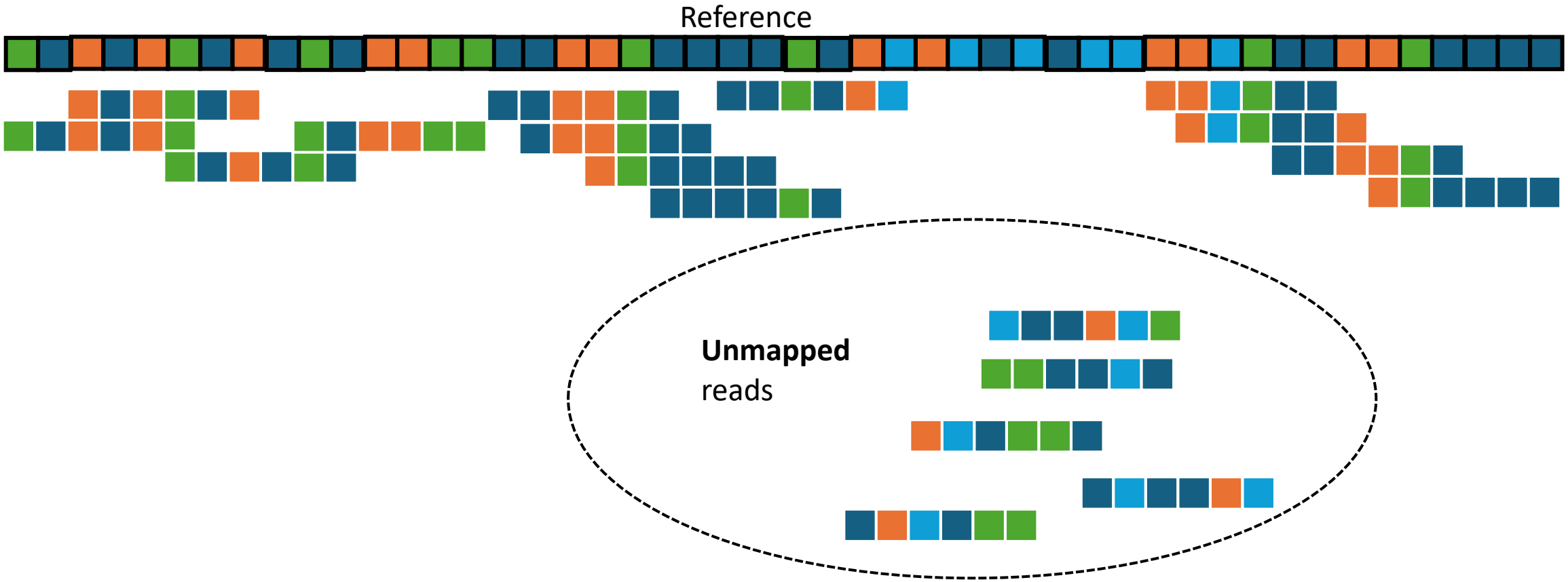


Genome Assembly Review



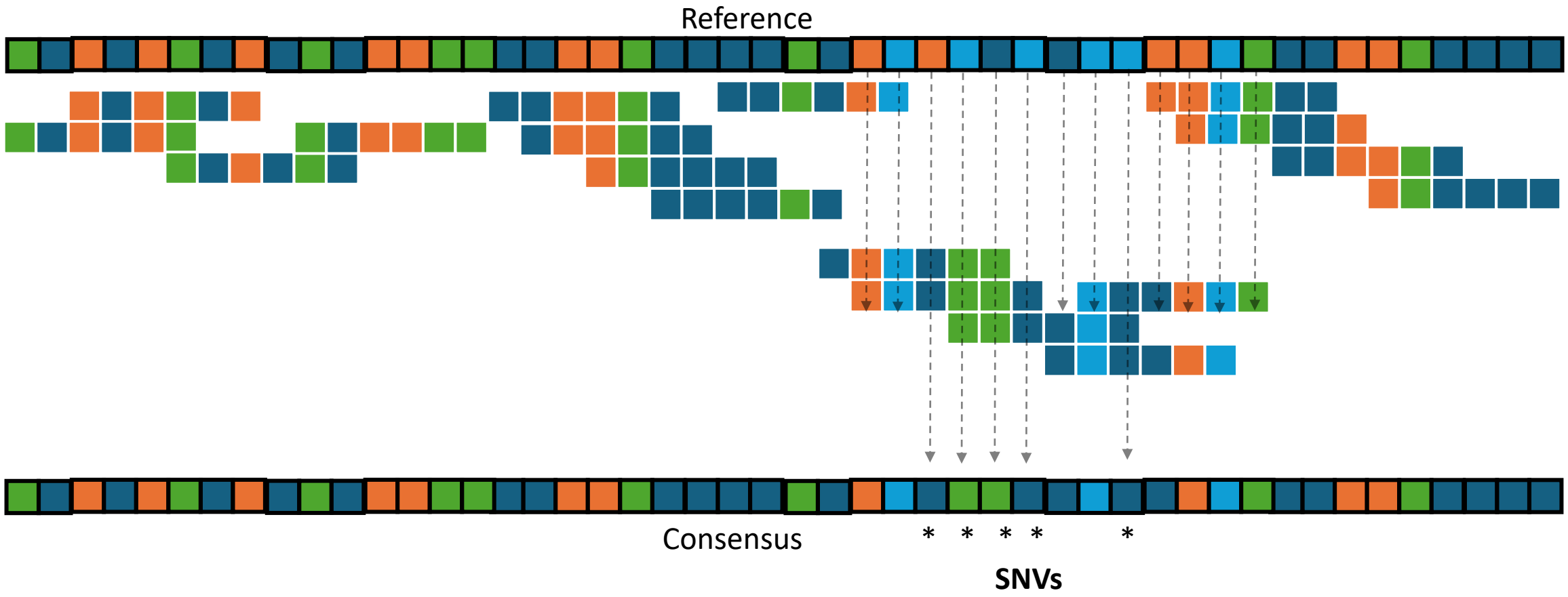
Genome Assembly Review

What about the reads that did not **map**?



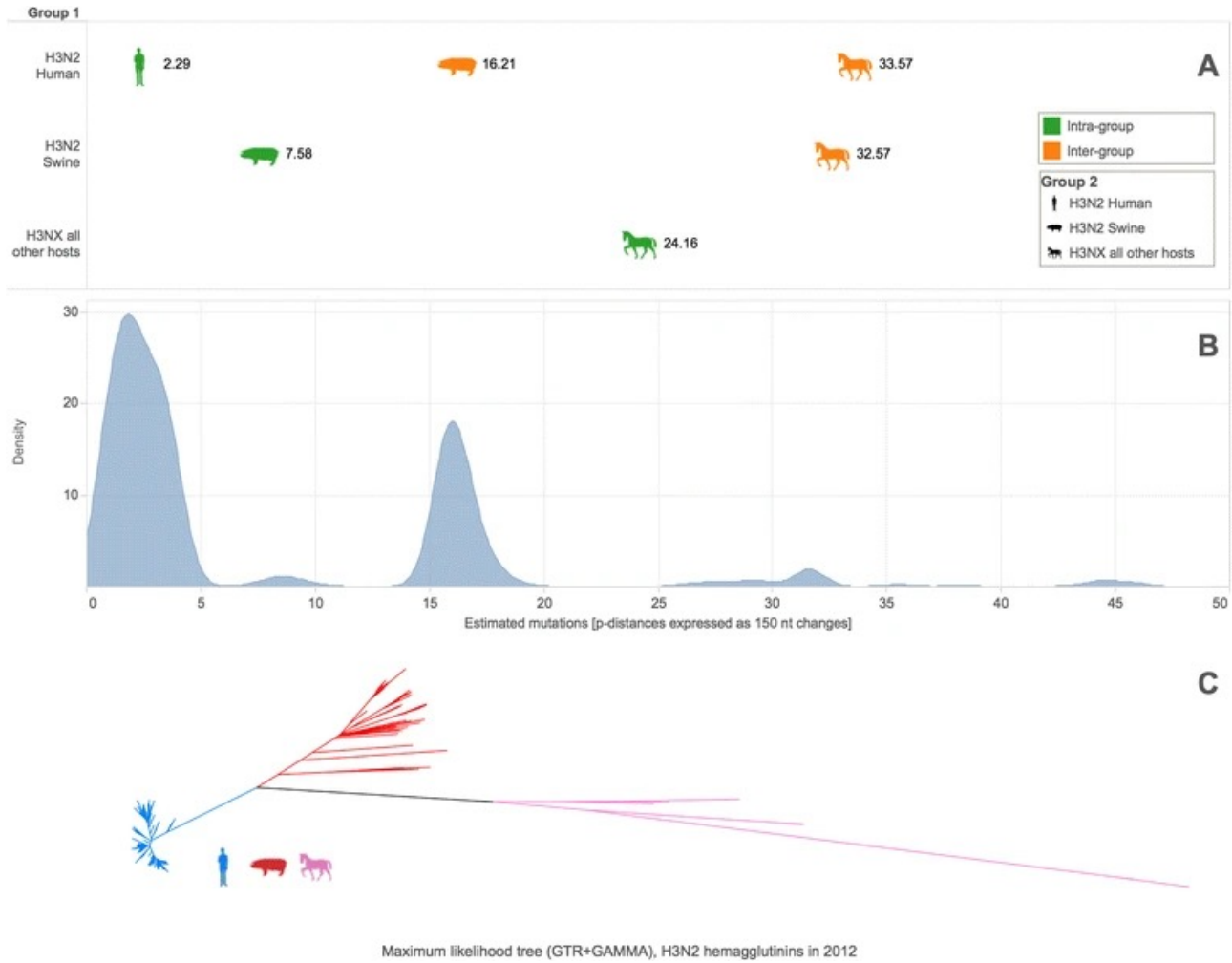
Single Nucleotide Variants: SNVs

If, during alignment, we allow for more base-pair **mis-matches** then this “deletion” gets filled in and the resulting consensus has **single nucleotide variants (SNV)**



IRMA: Iterative Refinement Meta Assembler

- Respiratory Viruses are very diverse
- Reference-based assembly works best when you choose the RIGHT reference
- Missing data (ex: amplicon dropout) cannot be “reference filled”
- US CDC (and all other WHO CCs) use IRMA for Influenza genome assembly



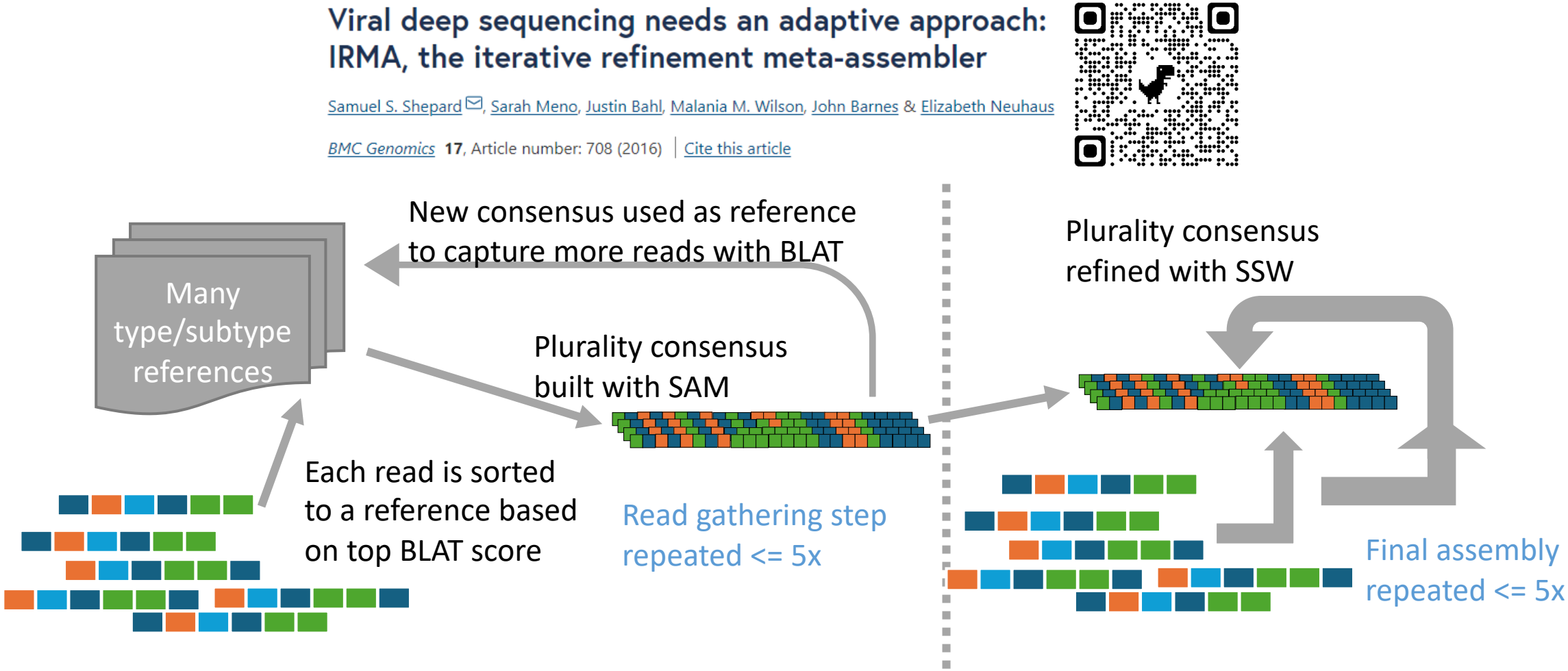
IRMA: Iterative Refinement Meta Assembler

IRMA overcomes the problem of choosing an accurate **reference** sequence for assembling diverse influenza genomes through **iterative refinement**

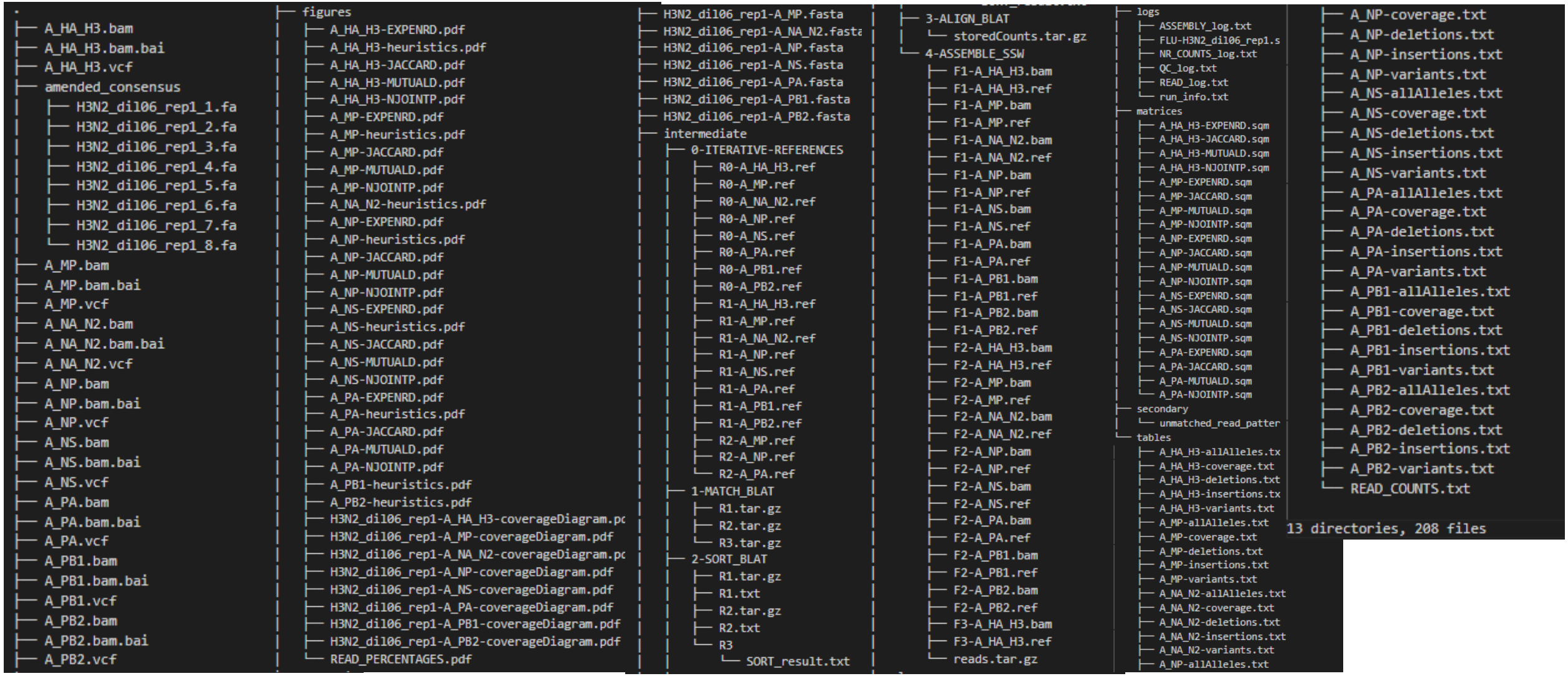
Viral deep sequencing needs an adaptive approach: IRMA, the iterative refinement meta-assembler

Samuel S. Shepard, Sarah Meno, Justin Bahl, Malania M. Wilson, John Barnes & Elizabeth Neuhaus

BMC Genomics **17**, Article number: 708 (2016) | Cite this article



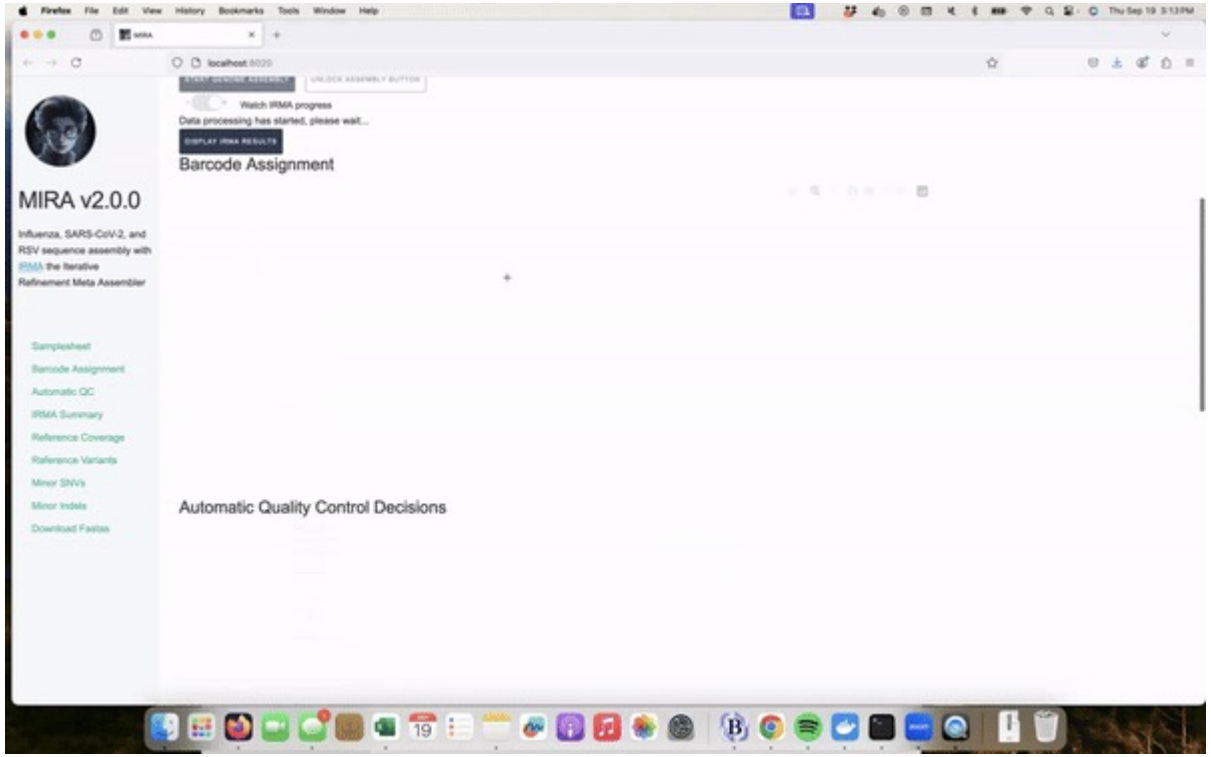
IRMA > MIRA


.
├── A_HA_H3.bam
├── A_HA_H3.bam.bai
├── A_HA_H3.vcf
├── amended_consensus
│   ├── H3N2_dil06_rep1_1.fa
│   ├── H3N2_dil06_rep1_2.fa
│   ├── H3N2_dil06_rep1_3.fa
│   ├── H3N2_dil06_rep1_4.fa
│   ├── H3N2_dil06_rep1_5.fa
│   ├── H3N2_dil06_rep1_6.fa
│   ├── H3N2_dil06_rep1_7.fa
│   └── H3N2_dil06_rep1_8.fa
├── A_MP.bam
├── A_MP.bam.bai
├── A_MP.vcf
├── A_NA_N2.bam
├── A_NA_N2.bam.bai
├── A_NA_N2.vcf
├── A_NP.bam
├── A_NP.bam.bai
├── A_NP.vcf
├── A_NS.bam
├── A_NS.bam.bai
├── A_NS.vcf
├── A_PA.bam
├── A_PA.bam.bai
├── A_PA.vcf
├── A_PB1.bam
├── A_PB1.bam.bai
├── A_PB1.vcf
├── A_PB2.bam
├── A_PB2.bam.bai
├── A_PB2.vcf
├── figures
│   ├── A_HA_H3-EXPENRD.pdf
│   ├── A_HA_H3-heuristics.pdf
│   ├── A_HA_H3-JACCARD.pdf
│   ├── A_HA_H3-MUTUALD.pdf
│   ├── A_HA_H3-NJOINTP.pdf
│   ├── A_MP-EXPENRD.pdf
│   ├── A_MP-heuristics.pdf
│   ├── A_MP-JACCARD.pdf
│   ├── A_MP-MUTUALD.pdf
│   ├── A_MP-NJOINTP.pdf
│   ├── A_NA_N2-heuristics.pdf
│   ├── A_NP-EXPENRD.pdf
│   ├── A_NP-heuristics.pdf
│   ├── A_NP-JACCARD.pdf
│   ├── A_NP-MUTUALD.pdf
│   ├── A_NP-NJOINTP.pdf
│   ├── A_NS-EXPENRD.pdf
│   ├── A_NS-heuristics.pdf
│   ├── A_NS-JACCARD.pdf
│   ├── A_NS-MUTUALD.pdf
│   ├── A_NS-NJOINTP.pdf
│   ├── A_PA-EXPENRD.pdf
│   ├── A_PA-heuristics.pdf
│   ├── A_PA-JACCARD.pdf
│   ├── A_PA-MUTUALD.pdf
│   ├── A_PA-NJOINTP.pdf
│   ├── A_PB1-heuristics.pdf
│   ├── A_PB2-heuristics.pdf
│   ├── H3N2_dil06_rep1-A_HA_H3-coverageDiagram.pc
│   ├── H3N2_dil06_rep1-A_MP-coverageDiagram.pdf
│   ├── H3N2_dil06_rep1-A_NA_N2-coverageDiagram.pc
│   ├── H3N2_dil06_rep1-A_NP-coverageDiagram.pdf
│   ├── H3N2_dil06_rep1-A_NS-coverageDiagram.pdf
│   ├── H3N2_dil06_rep1-A_PA-coverageDiagram.pdf
│   ├── H3N2_dil06_rep1-A_PB1-coverageDiagram.pdf
│   ├── H3N2_dil06_rep1-A_PB2-coverageDiagram.pdf
│   └── READ_PERCENTAGES.pdf
├── H3N2_dil06_rep1-A_MP.fasta
├── H3N2_dil06_rep1-A_NA_N2.fasta
├── H3N2_dil06_rep1-A_NP.fasta
├── H3N2_dil06_rep1-A_NS.fasta
├── H3N2_dil06_rep1-A_PA.fasta
├── H3N2_dil06_rep1-A_PB1.fasta
├── H3N2_dil06_rep1-A_PB2.fasta
├── intermediate
│   ├── 0-ITERATIVE-REFERENCES
│   │   ├── R0-A_HA_H3.ref
│   │   ├── R0-A_MP.ref
│   │   ├── R0-A_NA_N2.ref
│   │   ├── R0-A_NP.ref
│   │   ├── R0-A_NS.ref
│   │   ├── R0-A_PA.ref
│   │   ├── R0-A_PB1.ref
│   │   ├── R0-A_PB2.ref
│   │   ├── R1-A_HA_H3.ref
│   │   ├── R1-A_MP.ref
│   │   ├── R1-A_NA_N2.ref
│   │   ├── R1-A_NP.ref
│   │   ├── R1-A_NS.ref
│   │   ├── R1-A_PA.ref
│   │   ├── R1-A_PB1.ref
│   │   ├── R1-A_PB2.ref
│   │   ├── R2-A_MP.ref
│   │   ├── R2-A_NP.ref
│   │   └── R2-A_PA.ref
│   ├── 1-MATCH_BLAT
│   │   ├── R1.tar.gz
│   │   ├── R2.tar.gz
│   │   └── R3.tar.gz
│   ├── 2-SORT_BLAT
│   │   ├── R1.tar.gz
│   │   ├── R1.txt
│   │   ├── R2.tar.gz
│   │   ├── R2.txt
│   │   └── R3
│   │   └── SORT_result.txt
│   ├── 3-ALIGN_BLAT
│   │   └── storedCounts.tar.gz
│   └── 4-ASSEMBLE_SSW
│   ├── F1-A_HA_H3.bam
│   ├── F1-A_HA_H3.ref
│   ├── F1-A_MP.bam
│   ├── F1-A_MP.ref
│   ├── F1-A_NA_N2.bam
│   ├── F1-A_NA_N2.ref
│   ├── F1-A_NP.bam
│   ├── F1-A_NP.ref
│   ├── F1-A_NS.bam
│   ├── F1-A_NS.ref
│   ├── F1-A_PA.bam
│   ├── F1-A_PA.ref
│   ├── F1-A_PB1.bam
│   ├── F1-A_PB1.ref
│   ├── F1-A_PB2.bam
│   ├── F1-A_PB2.ref
│   ├── F2-A_HA_H3.bam
│   ├── F2-A_HA_H3.ref
│   ├── F2-A_MP.bam
│   ├── F2-A_MP.ref
│   ├── F2-A_NA_N2.bam
│   ├── F2-A_NA_N2.ref
│   ├── F2-A_NP.bam
│   ├── F2-A_NP.ref
│   ├── F2-A_NS.bam
│   ├── F2-A_NS.ref
│   ├── F2-A_PA.bam
│   ├── F2-A_PA.ref
│   ├── F2-A_PB1.bam
│   ├── F2-A_PB1.ref
│   ├── F2-A_PB2.bam
│   ├── F2-A_PB2.ref
│   ├── F3-A_HA_H3.bam
│   ├── F3-A_HA_H3.ref
│   └── reads.tar.gz
├── logs
│   ├── ASSEMBLY_log.txt
│   ├── FLU-H3N2_dil06_rep1.s
│   ├── NR_COUNTS_log.txt
│   ├── QC_log.txt
│   ├── READ_log.txt
│   └── run_info.txt
├── matrices
│   ├── A_HA_H3-EXPENRD.sqm
│   ├── A_HA_H3-JACCARD.sqm
│   ├── A_HA_H3-MUTUALD.sqm
│   ├── A_HA_H3-NJOINTP.sqm
│   ├── A_MP-EXPENRD.sqm
│   ├── A_MP-JACCARD.sqm
│   ├── A_MP-MUTUALD.sqm
│   ├── A_MP-NJOINTP.sqm
│   ├── A_NP-EXPENRD.sqm
│   ├── A_NP-JACCARD.sqm
│   ├── A_NP-MUTUALD.sqm
│   ├── A_NP-NJOINTP.sqm
│   ├── A_NS-EXPENRD.sqm
│   ├── A_NS-JACCARD.sqm
│   ├── A_NS-MUTUALD.sqm
│   ├── A_NS-NJOINTP.sqm
│   ├── A_PA-EXPENRD.sqm
│   ├── A_PA-JACCARD.sqm
│   ├── A_PA-MUTUALD.sqm
│   └── A_PA-NJOINTP.sqm
├── secondary
│   └── unmatched_read_patter
└── tables
    ├── A_HA_H3-allAlleles.tx
    ├── A_HA_H3-coverage.txt
    ├── A_HA_H3-deletions.txt
    ├── A_HA_H3-insertions.tx
    ├── A_HA_H3-variants.txt
    ├── A_MP-allAlleles.txt
    ├── A_MP-coverage.txt
    ├── A_MP-deletions.txt
    ├── A_MP-insertions.txt
    ├── A_MP-variants.txt
    ├── A_NA_N2-allAlleles.txt
    ├── A_NA_N2-coverage.txt
    ├── A_NA_N2-deletions.txt
    ├── A_NA_N2-insertions.txt
    ├── A_NA_N2-variants.txt
    ├── A_NP-allAlleles.txt
    ├── A_NP-coverage.txt
    ├── A_NP-deletions.txt
    ├── A_NP-insertions.txt
    ├── A_NP-variants.txt
    ├── A_NS-allAlleles.txt
    ├── A_NS-coverage.txt
    ├── A_NS-deletions.txt
    ├── A_NS-insertions.txt
    ├── A_NS-variants.txt
    ├── A_PA-allAlleles.txt
    ├── A_PA-coverage.txt
    ├── A_PA-deletions.txt
    ├── A_PA-insertions.txt
    ├── A_PA-variants.txt
    ├── A_PB1-allAlleles.txt
    ├── A_PB1-coverage.txt
    ├── A_PB1-deletions.txt
    ├── A_PB1-insertions.txt
    ├── A_PB1-variants.txt
    ├── A_PB2-allAlleles.txt
    ├── A_PB2-coverage.txt
    ├── A_PB2-deletions.txt
    ├── A_PB2-insertions.txt
    ├── A_PB2-variants.txt
    └── READ_COUNTS.txt

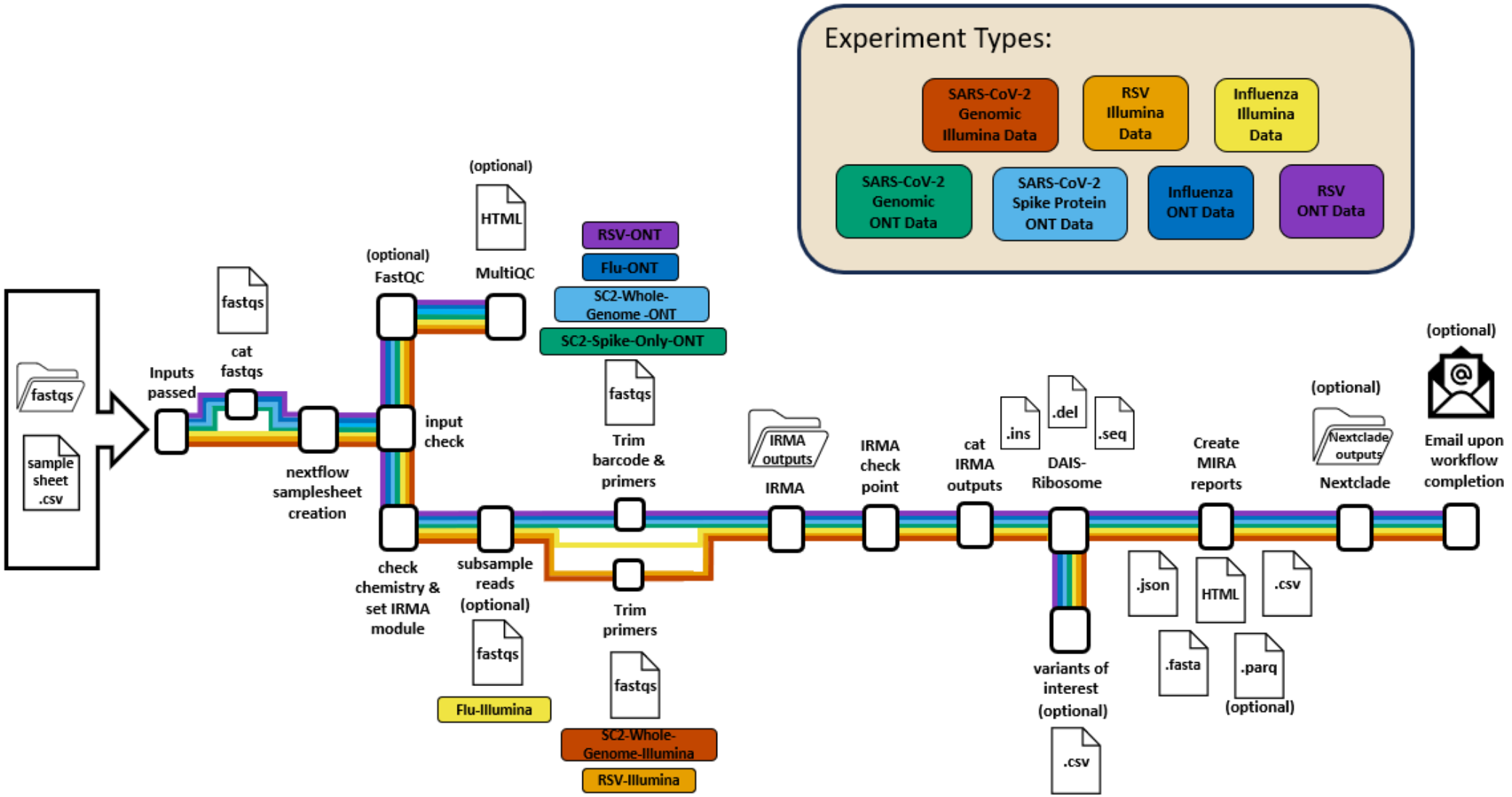
13 directories, 208 files

IRMA > MIRA

- IRMA algorithm is extensive
- Need for pre-assembly and post-assembly QC
  - Trimming
  - Downsampling
  - Coverage
  - Completeness
  - Variant calls
- Need for aggregation across runs
- Need for low-code options
- Additional features like genome annotation

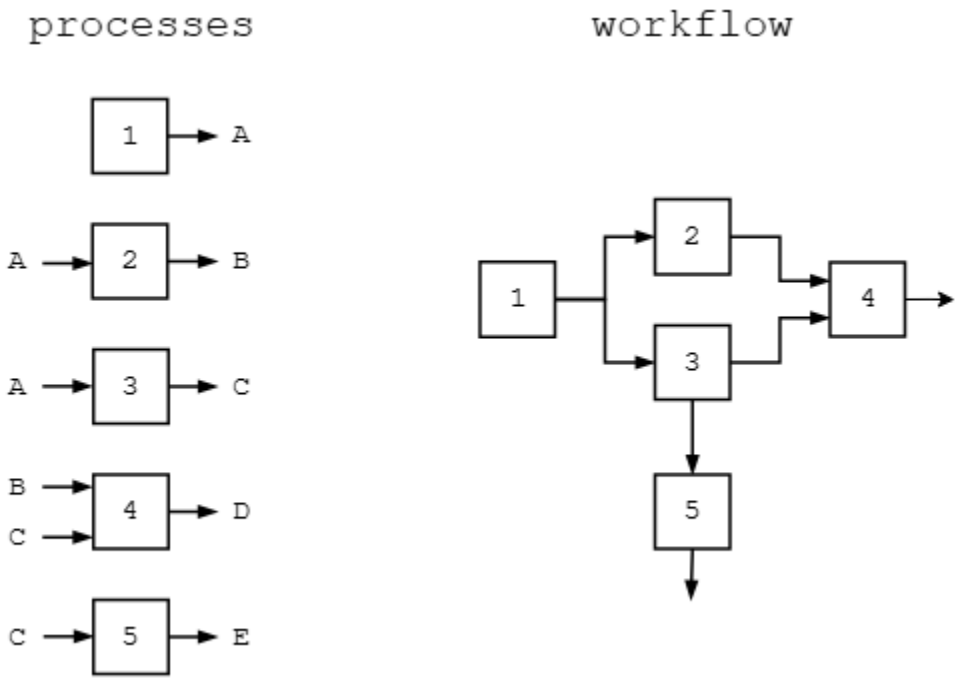


MIRA


Experiment Types:
SARS-CoV-2 Genomic Illumina Data
RSV Illumina Data
Influenza Illumina Data
SARS-CoV-2 Genomic ONT Data
SARS-CoV-2 Spike Protein ONT Data
Influenza ONT Data
RSV ONT Data
fastqs
sample sheet .csv
Inputs passed
fastqs
cat fastqs
nextflow samplesheet creation
(optional) FastQC
(optional)
HTML
MultiQC
input check
RSV-ONT
Flu-ONT
SC2-Whole-Genome -ONT
SC2-Spike-Only-ONT
fastqs
Trim barcode & primers
check chemistry & set IRMA module
subsample reads (optional)
fastqs
Flu-Illumina
Trim primers
fastqs
SC2-Whole-Genome-Illumina
RSV-Illumina
IRMA outputs
IRMA
IRMA check point
cat IRMA outputs
.ins
.del
.seq
DAIS-Ribosome
variants of interest (optional)
.csv
Create MIRA reports
.json
HTML
.csv
.fasta
.parq
(optional)
(optional)
Nextclade outputs
Nextclade
(optional)
Email upon workflow completion

Workflow managers

- Automate complex pipelines
  - Manage many steps, tools, and dependencies
- Track inputs, outputs, and dependencies
  - Run only what needs to be updated
- Enable reproducibility
  - Same workflow, same results across systems
- Scale from laptop to HPC/cloud
  - Handle parallelization and scheduling

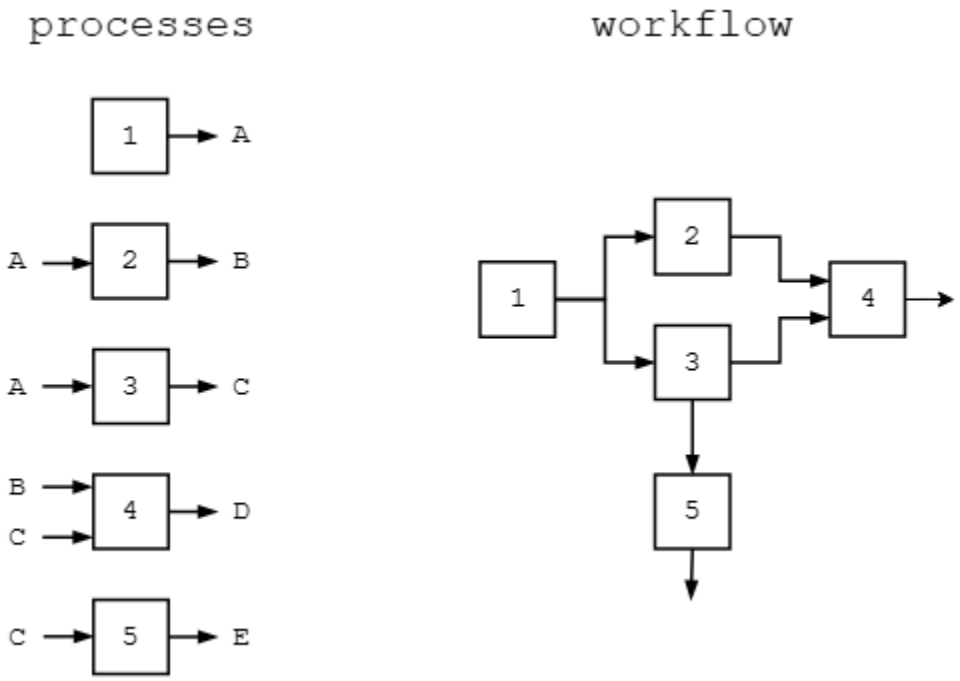


Workflow managers

Popular Workflow Managers in Bioinformatics

- Snakemake: MIRA GUI
  - Python-based, rule-driven
  - Feels familiar to Bash + Python users
- Nextflow: MIRA-NF
  - DSL built on Groovy

- Both have strong integration with HPC/schedulers
- Both can leverage containers via Docker/Singularity



MIRA-NF

System Requirements & Setup

- Java v17 or higher (Needed by Nextflow)
- Nextflow workflow engine
- Singularity-CE or Docker for containerized execution
- Git for cloning the repository
- Test your environment first
    - Use the included test profile (e.g., -profile test,docker or -profile test,singularity) before running real data

MIRA-NF

- Input & Samplesheet Requirements
- Prepare a samplesheet
  - Illumina: columns for `sample_id` & `sample_type`
  - ONT: `barcode`, `sample_id`, `sample_type`
  - Sample names must be unique and not nested
- Directory layout
  - FASTQ files and samplesheet go in a run folder used as --runpath
  - Note: The name of the run folder will be used to name output files

Illumina data should be set up as follows:

```
sample_id,sample_type
sample_1,Test
sample_2,Test
sample_3,Test
sample_4,Test
```

Oxford Nanopore data should be set up as follows:

```
barcode,sample_id,sample_type
barcode07,s1,Test
barcode37,s2,Test
barcode41,s3,Test
```

MIRA-NF

Important things to note about Samplesheet:

- Sample names within the `sample_id` column need to be unique.
- The headers must be named as seen above.
- Be sure that there are no empty lines at the end of the samplesheet.
- For Illumina samples be sure that you have read 1 and read 2 for all samples in samplesheet.
- Illumina fastq file must be in this format: {sample_id}_R1**fastq** or {sample_id}_R1**fq** AND {sample_id}_R2**fastq** or {sample_id}_R2**fq**

Illumina data should be set up as follows:

```
sample_id,sample_type
sample_1,Test
sample_2,Test
sample_3,Test
sample_4,Test
```

Oxford Nanopore data should be set up as follows:

```
barcode,sample_id,sample_type
barcode07,s1,Test
barcode37,s2,Test
barcode41,s3,Test
```

Illumina set up should be set up as follows:

1. <RUN_PATH>/fastqs <- all fastqs should be out at this level
2. <RUN_PATH>/samplesheet.csv

Oxford Nanopore set up should be set up as follows:

1. <RUN_PATH>/fastq_pass <- fastqs should be within barcode folders as given by ONT machine
2. <RUN_PATH>/samplesheet.csv

MIRA-NF

Running the Pipeline — Core Command

```
nextflow run ./main.nf \
    -profile singularity,local \
    --input <RUN_PATH>/samplesheet.csv \
    --outdir <OUTDIR> \
    --runpath <RUN_PATH> \
    --e <EXPERIMENT_TYPE> \
    [other optional flags]
```

profile: compute environment (e.g., singularity, docker, local, slurm, sge)

--e: experiment type (e.g., **Flu-ONT**, **SC2-Whole-Genome-Illumina**)

MIRA-NF

Running the Pipeline — Core Command

```
nextflow run ./main.nf \
    -profile singularity,local \
    --input <RUN_PATH>/samplesheet.csv \
    --outdir <OUTDIR> \
    --runpath <RUN_PATH> \
    --e <EXPERIMENT_TYPE> \
    [other optional flags]
```

profile: compute environment (e.g., singularity, docker, local, slurm, sge)
--e: experiment type (e.g., **Flu-ONT**, **SC2-Whole-Genome-Illumina**)

Useful Optional Flags

--p: built-in primer schema (for SC2/RSV)
--custom_primers: supply custom primer FASTA
--subsample_reads: limit reads for faster analysis
--parquet_files: make additional parquet files (formatted like csv)
--read_qc: run FastQC/MultiQC modules
--nextclade: run nextclade on passing samples
HPC queue/notification options (--process_q, --email)

MIRA-NF Requirements

WSL Java Installation:

1. **Update the package list:**
   sudo apt update && sudo apt upgrade -y
2. **Install Java:**
   sudo apt install openjdk-17-jdk -y
3. **Verify Java installation:**
   java -version

Mac Java Installation:

1. **Check if Java is installed:**
   java -version
2. **Install Java using Homebrew:**
   1. **Install homebrew:**
      /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
   2. **Install Java:**
      brew install openjdk
   3. **Add to path**
      sudo ln -sfn /opt/homebrew/opt/openjdk/libexec/openjdk.jdk /Library/Java/JavaVirtualMachines/openjdk.jdk

MIRA-NF Requirements

Install Nextflow:

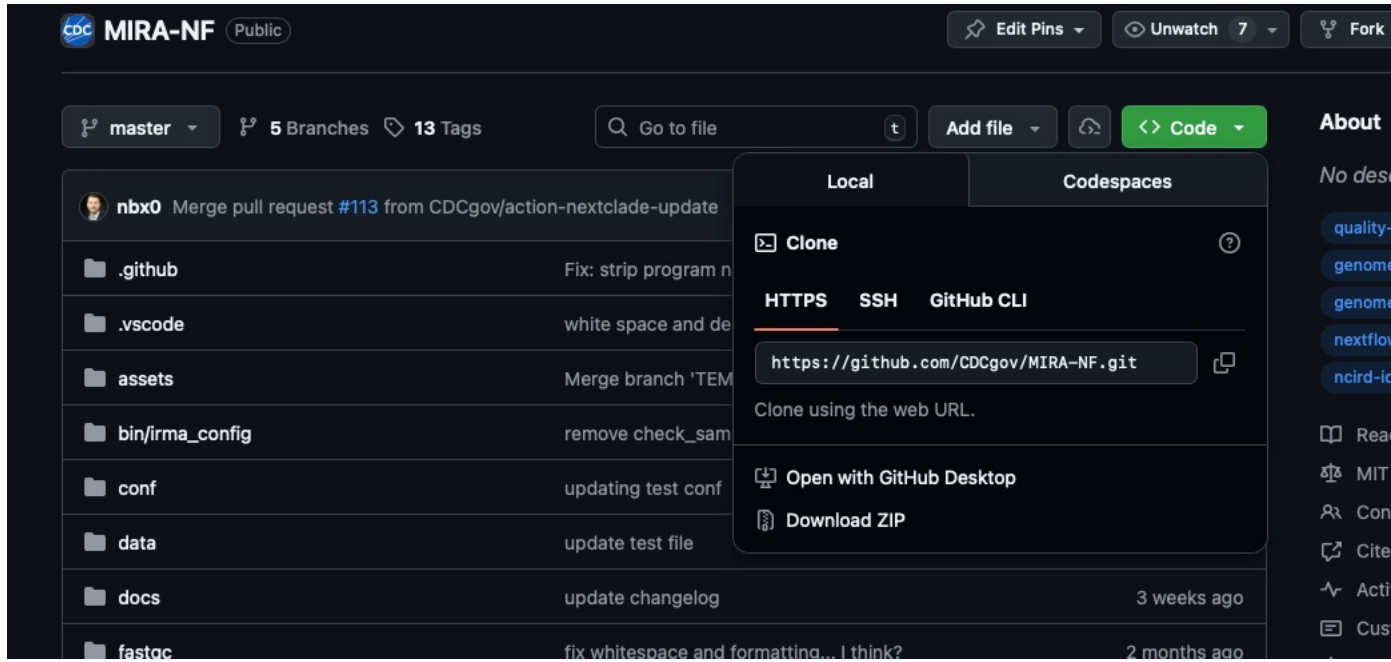
1. **Nextflow can be downloaded and installed directly:**
   curl -s https://get.nextflow.io | bash
2. **Move Nextflow to a system path:**
   sudo mv nextflow /usr/local/bin/
3. **Ensure the file is an executable:**
   sudo chmod +x /usr/local/bin/nextflow

Genome Assembly with MIRA-NF practical

1. Clone the MIRA-NF repo
git clone https://github.com/CDCgov/MIRA-NF.git

2. Make your samplesheet (vim, echo and ls redirect)

3. Build MIRA-NF execution statement
    Can do this in a <file>.sh
        - To build into a pipeline
        - Run again in the future



Genome Assembly with MIRA-NF practical

```
#!/bin/bash


nextflow run ~/MIRA-NF/main.nf \
    -profile singularity,local \
    --input /Users/qgx6/MIRA_NGS/test-data/samplesheet.csv \
    --outdir /Users/qgx6/MIRA_NGS/test-data/ \
    --runpath /Users/qgx6/MIRA_NGS/test-data/ \
    --e Flu-Illumina \
```

Mac users: may need docker,local with Docker Desktop open

Genome Assembly with MIRA-NF practical

MIRA-NF output

General Output Structure:

```
|---outputs/
   |---aggregate_outputs/
      |---multiqc/  (when applicable) --> multiqc outputs
      |---dais-ribosome/ -> dais inputs and outputs
      |---dash-json/ -> json files
      |---mira-reports/ -> the aggregated fasta files and html files
      |---csv-reports/ -> CSV summary files
      |---parquet-reports/ (when applicable)
   |---Sample_ID/
      |---subsampled-reads/ (when applicable) -> fastqs and log files
      |---barcode-trimmed-reads/ (when applicable) -> fastqs and log files
      |---primer-trimmed-reads/ (when applicable) -> fastqs and log files
      |---IRMA/Sample_ID/ -> IRMA outputs and log files
      |---IRMA-negative/ (when applicable)
   |---nextclade -> Inputs and outputs for Nextclade
      |---input_fasta_files/ -> Input FASTA files for running Nextclade
      |--- All nextclade outputs files including aligned fastas, auspice json, and csv files.
   |---fastq_pass/ -> ONT data only – concatenated fastqs
   |---pipeline_info/ -> execution reports, sad_samples.tsv and program versions file
```

MIRA-NF output

```
qgx6@rosalind02:mira-results$ tree
.
├── aggregate_outputs
│   ├── csv-reports
│   │   ├── mira_India_20260210_H3varCheck_aavars.csv
│   │   ├── mira_India_20260210_H3varCheck_all_alleles.csv
│   │   ├── mira_India_20260210_H3varCheck_amended_consensus.csv
│   │   ├── mira_India_20260210_H3varCheck_amino_acid_consensus.csv
│   │   ├── mira_India_20260210_H3varCheck_coverage.csv
│   │   ├── mira_India_20260210_H3varCheck_filtered_variants.csv
│   │   ├── mira_India_20260210_H3varCheck_indels.csv
│   │   ├── mira_India_20260210_H3varCheck_irma_config.csv
│   │   ├── mira_India_20260210_H3varCheck_reads.csv
│   │   └── mira_India_20260210_H3varCheck_summary.csv
│   ├── dais-ribosome
│   │   ├── DAIS_ribosome.del
│   │   ├── DAIS_ribosome.gen
│   │   ├── DAIS_ribosome.gen.del
│   │   ├── DAIS_ribosome.gen.ins
│   │   ├── DAIS_ribosome_input.fasta
│   │   ├── DAIS_ribosome.ins
│   │   └── DAIS_ribosome.seq
│   ├── dash-json
│   │   ├── alleles.json
│   │   ├── barcode_distribution.json
│   │   ├── coveragefig_NIV-25-1244_linear.json
│   │   ├── coveragefig_NIV-25-1292_linear.json
│   │   ├── coveragefig_NIV-25-2072_linear.json
│   │   ├── coveragefig_NIV-25-2129_linear.json
│   │   ├── coveragefig_NIV-25-2426_linear.json
│   │   ├── coveragefig_NIV-25-2429_linear.json
│   │   ├── coveragefig_NIV-25-2432_linear.json
│   │   ├── coveragefig_NIV-25-2433_linear.json
│   │   ├── coveragefig_NIVCOV-1689_linear.json
│   │   ├── coverage.json
│   │   ├── dais_vars.json
│   │   ├── heatmap.json
│   │   ├── indels.json
│   │   ├── irma_summary.json
│   │   ├── nt_sequences.json
│   │   ├── pass_fail_heatmap.json
│   │   ├── pass_fail_qc.json
│   │   ├── qc_statement.json
│   │   ├── readsfig_NIV-25-1244.json
│   │   ├── readsfig_NIV-25-1292.json
│   │   ├── readsfig_NIV-25-2072.json
│   │   ├── readsfig_NIV-25-2129.json
│   │   ├── readsfig_NIV-25-2426.json
│   │   ├── readsfig_NIV-25-2429.json
│   │   ├── readsfig_NIV-25-2432.json
│   │   ├── readsfig_NIV-25-2433.json
│   │   ├── readsfig_NIVCOV-1689.json
│   │   ├── reads.json
│   │   └── vtype.json
│   └── mira-reports
│       ├── mira_India_20260210_H3varCheck_amended_consensus.fasta
│       ├── mira_India_20260210_H3varCheck_amino_acid_consensus.fasta
│       ├── mira_India_20260210_H3varCheck_amino_acid_consensus_HA.fasta
│       ├── mira_India_20260210_H3varCheck_failed_amended_consensus.fasta
│       ├── mira_India_20260210_H3varCheck_failed_amino_acid_consensus.fasta
│       ├── mira_India_20260210_H3varCheck_summary.html
│       ├── mira_NIV-25-1244_coverage.html
│       ├── mira_NIV-25-1292_coverage.html
│       ├── mira_NIV-25-2072_coverage.html
│       ├── mira_NIV-25-2129_coverage.html
│       ├── mira_NIV-25-2426_coverage.html
│       ├── mira_NIV-25-2429_coverage.html
│       ├── mira_NIV-25-2432_coverage.html
│       ├── mira_NIV-25-2433_coverage.html
│       └── mira_NIVCOV-1689_coverage.html
├── fastq_pass
│   └── cat_fastqs
│       ├── NIV-25-1244_nf_combined.fastq.gz
│       ├── NIV-25-1292_nf_combined.fastq.gz
│       ├── NIV-25-2072_nf_combined.fastq.gz
│       ├── NIV-25-2129_nf_combined.fastq.gz
│       ├── NIV-25-2426_nf_combined.fastq.gz
│       ├── NIV-25-2429_nf_combined.fastq.gz
│       ├── NIV-25-2432_nf_combined.fastq.gz
│       ├── NIV-25-2433_nf_combined.fastq.gz
│       └── NIVCOV-1689_nf_combined.fastq.gz
├── nextclade
│   └── input_fasta_files
│       ├── nextclade_India_20260210_H3varCheck_flu_h3n2_ha.fasta
│       └── nextclade_India_20260210_H3varCheck_flu_h3n2_na.fasta
├── pipeline_info
│   ├── collated_versions.unique.yml
│   ├── collated_versions.yml
│   ├── execution_report_2026-02-10_09-38-07.html
│   ├── execution_report_2026-02-10_10-34-44.html
│   ├── execution_report_2026-02-10_10-35-58.html
│   ├── execution_report_2026-02-10_10-42-00.html
│   ├── execution_timeline_2026-02-10_09-38-07.html
│   ├── execution_timeline_2026-02-10_10-34-44.html
│   ├── execution_timeline_2026-02-10_10-35-58.html
│   ├── execution_timeline_2026-02-10_10-42-00.html
│   ├── execution_trace_2026-02-10_09-38-07.txt
│   ├── execution_trace_2026-02-10_10-34-44.txt
│   ├── execution_trace_2026-02-10_10-35-58.txt
│   ├── execution_trace_2026-02-10_10-42-00.txt
│   ├── mira_version_check.txt
│   ├── pipeline_dag_2026-02-10_09-38-07.html
│   ├── pipeline_dag_2026-02-10_10-34-44.html
│   ├── pipeline_dag_2026-02-10_10-35-58.html
│   ├── pipeline_dag_2026-02-10_10-42-00.html
│   └── samplesheet.valid.csv
└── NIV-25-1244
    ├── barcode-trimmed-reads
    │   ├── NIV-25-1244.barcode24.bartrim.stderr.log
    │   ├── NIV-25-1244.barcode24.bartrim.stdout.log
    │   └── NIV-25-1244_trimmed.fastq
    └── IRMA
        ├── NIV-25-1244
        │   ├── A_HA_H3.bam
        │   ├── A_HA_H3.bam.bai
        │   ├── A_HA_H3.fasta
        │   ├── A_HA_H3.fasta.fai
        │   ├── A_HA_H3.vcf
        │   ├── amended_consensus
        │   │   ├── NIV-25-1244_1.fa
        │   │   ├── NIV-25-1244_2.fa
        │   │   ├── NIV-25-1244_3.fa
        │   │   ├── NIV-25-1244_4.fa
        │   │   ├── NIV-25-1244_5.fa
        │   │   ├── NIV-25-1244_6.fa
        │   │   ├── NIV-25-1244_7.fa
        │   │   └── NIV-25-1244_8.fa
        │   ├── A_MP.bam
        │   ├── A_MP.bam.bai
        │   ├── A_MP.fasta
        │   ├── A_MP.vcf
        │   ├── A_NA_N2.bam
        │   ├── A_NA_N2.bam.bai
        │   ├── A_NA_N2.fasta
        │   ├── A_NA_N2.vcf
        │   ├── A_NP.bam
        │   ├── A_NP.bam.bai
        │   ├── A_NP.fasta
        │   ├── A_NP.vcf
        │   ├── A_NS.bam
        │   ├── A_NS.bam.bai
        │   ├── A_NS.fasta
        │   ├── A_NS.vcf
        │   ├── A_PA.bam
        │   ├── A_PA.bam.bai
        │   ├── A_PA.fasta
        │   ├── A_PA.vcf
        │   ├── A_PB1.bam
        │   ├── A_PB1.bam.bai
        │   ├── A_PB1.fasta
        │   ├── A_PB1.vcf
        │   ├── A_PB2.bam
        │   ├── A_PB2.bam.bai
        │   ├── A_PB2.fasta
        │   ├── A_PB2.vcf

163 directories, 2158 files
```

+ all IRMA outputs per sample

MIRA-NF outputs demo

Pipeline + MIRA-NF

- Can automate samplesheet creation
- Pipeline basecalling outputs into MIRA-NF
- Pipeline MIRA output tables to databases
- Pipeline output fastas to phylogenetic analyses

Second run get these accessions.txt from website:

SRR37675414,d969e179
SRR37675415,ba21bd1f
SRR37675416,ad336cc2
SRR37675417,89bb6967
SRR37675418,81ae9ee5
SRR37675419,73ceed0e
SRR37675420,6796f13d
SRR37675421,314ac5ba
SRR37675422, 240aa994
SRR37675423, 239e44e6

Practical: Putting it all together

```
#!/bin/bash
for i in $(cut -f1 -d, accessions.txt); do
        fastq-dump --split-3 --defline-seq '@$sn/$ri' --defline-qual '+' $i ;
done

for i in $(cat accessions.txt); do
        mv $(echo $i | cut -f1 -d,)_1.fastq $(echo $i | cut -f2 -d,)_R1.fastq
        mv $(echo $i | cut -f1 -d,)_2.fastq $(echo $i | cut -f2 -d,)_R2.fastq;
done

gzip *.fastq

mkdir fastqs
mv *.fastq.gz fastqs

echo "sample_id,sample_type" > samplesheet.csv

ls fastqs | cut -f1 -d_ | uniq | sed "s/$/,Test/g" >> samplesheet.csv

nextflow run ~/MIRA-NF/main.nf \
    -profile docker,local \
    --input $(pwd)/samplesheet.csv \
    --outdir $(pwd)/ \
    --runpath $(pwd) \
    --e Flu-Illumina \
```

Shebang line

```
#!/bin/bash
for i in $(cut -f1 -d, accessions.txt); do
        fastq-dump --split-3 --defline-seq '@$sn/$ri' --defline-qual '+' $i ;
done

for i in $(cat accessions.txt); do
        mv $(echo $i | cut -f1 -d,)_1.fastq $(echo $i | cut -f2 -d,)_R1.fastq
        mv $(echo $i | cut -f1 -d,)_2.fastq $(echo $i | cut -f2 -d,)_R2.fastq;
done

gzip *.fastq

mkdir fastqs
mv *.fastq.gz fastqs

echo "sample_id,sample_type" > samplesheet.csv

ls fastqs | cut -f1 -d_ | uniq | sed "s/$/,Test/g" >> samplesheet.csv

nextflow run ~/MIRA-NF/main.nf \
    -profile docker,local \
    --input $(pwd)/samplesheet.csv \
    --outdir $(pwd)/ \
    --runpath $(pwd) \
    --e Flu-Illumina \
```

2 for loops: pull fastqs from SRA and rename them

```
#!/bin/bash
for i in $(cut -f1 -d, accessions.txt); do
        fastq-dump --split-3 --defline-seq '@$sn/$ri' --defline-qual '+' $i ;
done

for i in $(cat accessions.txt); do
        mv $(echo $i | cut -f1 -d,)_1.fastq $(echo $i | cut -f2 -d,)_R1.fastq
        mv $(echo $i | cut -f1 -d,)_2.fastq $(echo $i | cut -f2 -d,)_R2.fastq;
done

gzip *.fastq

mkdir fastqs
mv *.fastq.gz fastqs

echo "sample_id,sample_type" > samplesheet.csv

ls fastqs | cut -f1 -d_ | uniq | sed "s/$/,Test/g" >> samplesheet.csv

nextflow run ~/MIRA-NF/main.nf \
    -profile docker,local \
    --input $(pwd)/samplesheet.csv \
    --outdir $(pwd)/ \
    --runpath $(pwd) \
    --e Flu-Illumina \
```

Compress fastqs, make a directory, move them

```
#!/bin/bash
for i in $(cut -f1 -d, accessions.txt); do
        fastq-dump --split-3 --defline-seq '@$sn/$ri' --defline-qual '+' $i ;
done

for i in $(cat accessions.txt); do
        mv $(echo $i | cut -f1 -d,)_1.fastq $(echo $i | cut -f2 -d,)_R1.fastq
        mv $(echo $i | cut -f1 -d,)_2.fastq $(echo $i | cut -f2 -d,)_R2.fastq;
done

gzip *.fastq

mkdir fastqs
mv *.fastq.gz fastqs

echo "sample_id,sample_type" > samplesheet.csv

ls fastqs | cut -f1 -d_ | uniq | sed "s/$/,Test/g" >> samplesheet.csv

nextflow run ~/MIRA-NF/main.nf \
    -profile docker,local \
    --input $(pwd)/samplesheet.csv \
    --outdir $(pwd)/ \
    --runpath $(pwd) \
    --e Flu-Illumina \
```

Write samplesheet using pipes, ls, echo, uniq, sed

```
#!/bin/bash
for i in $(cut -f1 -d, accessions.txt); do
        fastq-dump --split-3 --defline-seq '@$sn/$ri' --defline-qual '+' $i ;
done

for i in $(cat accessions.txt); do
        mv $(echo $i | cut -f1 -d,)_1.fastq $(echo $i | cut -f2 -d,)_R1.fastq
        mv $(echo $i | cut -f1 -d,)_2.fastq $(echo $i | cut -f2 -d,)_R2.fastq;
done

gzip *.fastq

mkdir fastqs
mv *.fastq.gz fastqs

echo "sample_id,sample_type" > samplesheet.csv

ls fastqs | cut -f1 -d_ | uniq | sed "s/$/,Test/g" >> samplesheet.csv

nextflow run ~/MIRA-NF/main.nf \
    -profile docker,local \
    --input $(pwd)/samplesheet.csv \
    --outdir $(pwd)/ \
    --runpath $(pwd) \
    --e Flu-Illumina \
```

You would have to change this if your sample IDs contain _

Run Nextflow with subshell $(pwd)

```
#!/bin/bash
for i in $(cut -f1 -d, accessions.txt); do
        fastq-dump --split-3 --defline-seq '@$sn/$ri' --defline-qual '+' $i ;
done

for i in $(cat accessions.txt); do
        mv $(echo $i | cut -f1 -d,)_1.fastq $(echo $i | cut -f2 -d,)_R1.fastq
        mv $(echo $i | cut -f1 -d,)_2.fastq $(echo $i | cut -f2 -d,)_R2.fastq;
done

gzip *.fastq

mkdir fastqs
mv *.fastq.gz fastqs

echo "sample_id,sample_type" > samplesheet.csv

ls fastqs | cut -f1 -d_ | uniq | sed "s/$/,Test/g" >> samplesheet.csv

nextflow run ~/MIRA-NF/main.nf \
    -profile docker,local \
    --input $(pwd)/samplesheet.csv \
    --outdir $(pwd)/ \
    --runpath $(pwd) \
    --e Flu-Illumina \
```

Pipeline + MIRA-NF practical

Make a new run id folder in MIRA_NGS, write the accessions.txt , mira_wrapper.sh inside your new runid

Add one (or more!) of the following to your run Mira wrapper script after nextflow and run!:

1. Copy the amended consensus.fasta to your home directory
2. Sed the amended consensus fasta to make strain names from sample IDs
3. Rename the aggregate_outputs directory to be <runID>_outputs
4. Count the number of samples that pass QC by grepping the aggregate_outputs/csv-reports/<runid>_summary.csv
5. CHALLENGE: Loop through the amended consensus.fasta and break it into 8 per-segment multifastas

Intermission

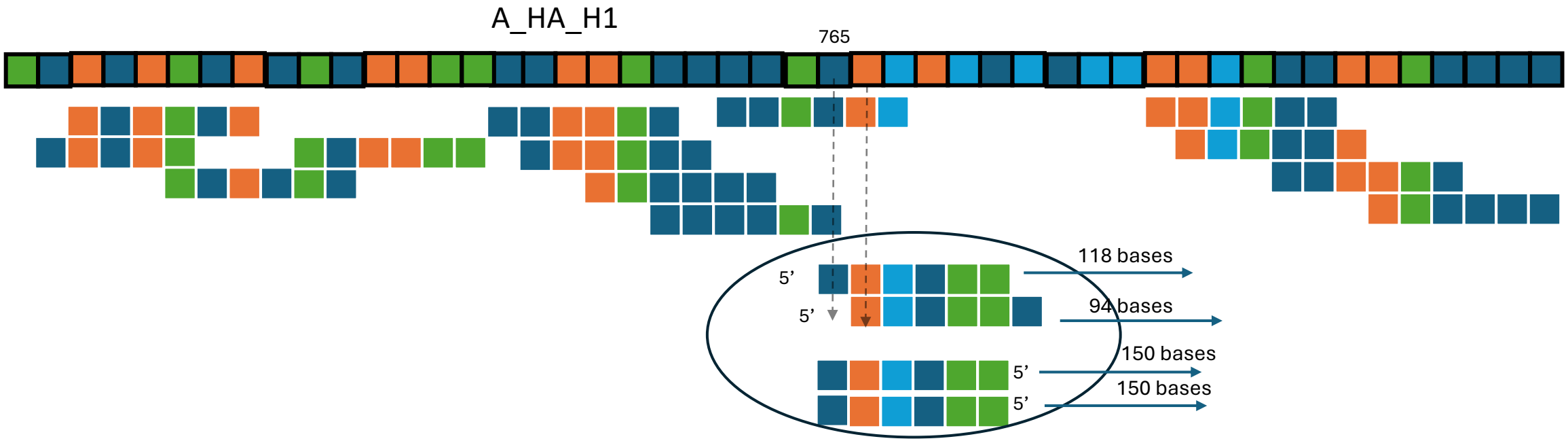
.sam and .bam files

- SAM Files (Sequence Alignment/Map)
- Text-based alignment format
  - Stores how reads align to a reference genome
- Human-readable
  - Can be viewed and inspected directly
- Contains alignments + metadata
  - Read names, positions, flags, CIGAR strings, mapping quality
- Large file size
  - Not efficient for storage or sharing

Col	Field	Type	Brief description
1	QNAME	String	Query template NAME
2	FLAG	Int	bitwise FLAG
3	RNAME	String	References sequence NAME
4	POS	Int	1- based leftmost mapping POSition
5	MAPQ	Int	MAPping Quality
6	CIGAR	String	CIGAR string
7	RNEXT	String	Ref. name of the mate/next read
8	PNEXT	Int	Position of the mate/next read
9	TLEN	Int	observed Template LENgth
10	SEQ	String	segment SEQuence
11	QUAL	String	ASCII of Phred-scaled base QUALity+33

```
M02989:30:000000000-L4R3N:1:1114:11884:19665_3:N:0:TTACGCACCT+GCTCTCGTTG        0       A_HA_H1 765     5       118M    *       0       0       GAACTATTACTGGACACTAGTGGAACCGGGAGACAAAATAACATTCGAAGCAACTGGTAATCTAGTGGCACCGAGATATGCATTCACA
ATGGAAAAAAATGCTGGATCAGGTATCATC  BGEGBGFHHHFFGGFGGGGGGGFGGHGGGGGGGGHHHHHHHHHHHHHHHGGHHHGHHHHHHHHHHGHHHGFGGFEGGGHHGHHHGHHGHHHHHEGHHHGGGHGFGHHEGHGHFHHHHHG
M02989:30:000000000-L4R3N:1:2107:10440:10828_1:N:0:TTACGCACCT+GCTCTCGTTG        16      A_HA_H1 765     5       150M    *       0       0       GAACTATTACTGGACACTAGTGGAACCGGGAGACAAAATAACATTCGAAGCAACTGGTAATCTAGTGGCACCGAGATATGCATTCACA
ATGGAAAAAAATGCTGGATCAGGTATCATCATTTCAGATACACCAGTCCACGATTGCAATAC  FHGEHHFFHFGEFB12F0GGDFEAE?GHHFFB02GEGEBBHFG0EEBDGBHGHGHGBHGDGBG1CHGEEEEGHHHGHHHHHDFHEDHHHGHHEEFEF2AA0HGGFHHGFA122CGB1ADHFGDFA3AF01F1B11FGFA@3BFFFA1A>A  AS:i:300       N
M:i:0   ZS:i:218
M02989:30:000000000-L4R3N:1:2109:8752:4631_1:N:0:TTACGCACCT+GCTCTCGTTG  16      A_HA_H1 765     5       150M    *       0       0       GAACTATTACTGGACACTAGTGGAACCGGGAGACAAAATAACATTCGAAGCAACTGGTAATCTAGTGGCACCGAGATATGCATTCACAATGGAAAA
AAATGCTGGATCAGGTATCATCATTTCAGATACACCAGTCCACGATTGCAATAC  HHHGHHFFHHGHHHGFGGHHHHGGECFGHGHHHHHEHHHHGFHHEGHGFGBGEGFFFGFHHHHHFHHEEB1GHGFDHFFFEEHDBFHFHHHHGGHHFHHFHFHFGHFHGGHEADBHHHGHHHFFFHGGFGAGGGGCGGBFFBFFFAAAAA  AS:i:300       NM:i:0 Z
S:i:218
M02989:30:000000000-L4R3N:1:1113:10382:6568_3:N:0:TTACGCACCT+GCTCTCGTTG 0       A_HA_H1 766     5       94M     *       0       0       AACTATTACTGGACACTAGTGGAACCGGGAGACAAAATAACATTCGAAGCAACTGGTAATCTAGTGGCACCGAGATATGCATTCACAATGGAAA B
HFFHHFFHHHHGGGGGGGGGGHHHGGGHGGGGHHHHHHHHHHHFHHHGGHGHHGHGGHHHHHHHHHHGHHGGGGGGGHHHHHHHHHGFFHHHHH
```

.sam and .bam files



QNAME	FLAG	RNAME	POS	MAPQ	CIGAR	RNEXT	PI	TLEN	SEQ	QUAL
M02989:30:000000000-L4R3N:1:1114:11884:19665_3:N:0:TTACGCACCT+GCTCTCGTTG	0	A_HA_H1	765	5	118M	*	0	0	GAACTATT	BGEGBGFHHHFFGG
M02989:30:000000000-L4R3N:1:2107:10440:10828_1:N:0:TTACGCACCT+GCTCTCGTTG	16	A_HA_H1	765	5	150M	*	0	0	GAACTATT	FHGEHHFFHFGEFB1
M02989:30:000000000-L4R3N:1:2109:8752:4631_1:N:0:TTACGCACCT+GCTCTCGTTG	16	A_HA_H1	765	5	150M	*	0	0	GAACTATT	HHHGHHFFHHGHH
M02989:30:000000000-L4R3N:1:1113:10382:6568_3:N:0:TTACGCACCT+GCTCTCGTTG	0	A_HA_H1	766	5	94M	*	0	0	AACTATTA	BHFFHHFFHHHHGG

.sam and .bam files

- CIGAR Strings (SAM/BAM)
- Describe how a read aligns to the reference
  - Encodes matches, mismatches, insertions, and deletions
- Stored in the SAM/BAM alignment record
  - One CIGAR string per aligned read
- Format: number + operation
  - Indicates length and type of alignment operation
- Common operations
  - M – alignment match/mismatch
  - I – insertion relative to reference
  - D – deletion relative to reference
  - S – soft clipping (read bases not aligned)
  - H – hard clipping (bases removed)
  - N – skipped region (e.g., introns)

10M2I5M1D20M

.sam and .bam files

- CIGAR Strings (SAM/BAM)
- Describe how a read aligns to the reference
  - Encodes matches, mismatches, insertions, and deletions
- Stored in the SAM/BAM alignment record
  - One CIGAR string per aligned read
- Format: number + operation
  - Indicates length and type of alignment operation
- Common operations
  - M – alignment match/mismatch
  - I – insertion relative to reference
  - D – deletion relative to reference
  - S – soft clipping (read bases not aligned)
  - H – hard clipping (bases removed)
  - N – skipped region (e.g., introns)

10M2I5M1D20M

10 aligned bases
2 inserted bases
5 aligned bases
1 deleted base
20 aligned bases

.sam and .bam files

- BAM Files (Binary Alignment/Map)
- Binary (compressed) version of SAM
  - Same information, smaller size
- Efficient for storage and analysis
  - Faster to read and write than SAM
- Often requires sorting and indexing
  - .bai index enables random access by genomic position
  - Necessary for viewing
- Standard input for downstream tools
  - Variant calling, visualization, QC

samtools

- Command-line toolkit for SAM/BAM files
  - Standard tool used across NGS workflows
  - Handy for *ad hoc* work
- Used after read alignment
- Fast and widely supported
  - Integrated into many pipelines and workflow managers
  - Even IRMA uses samtools in the algorithm
- View and convert formats
  - `samtools view` — SAM ↔ BAM conversion
  - `samtools view` for sam to bam
  - `samtools view -b` for bam to sam
- Sort alignments
  - `samtools sort` — required for many downstream tools
- Index files
  - `samtools index` — enables random access (.bai)
- Basic statistics
  - samtools flagstat — alignment summary
  - samtools stats — detailed metrics

Samtools practical

1. Go to the MIRA output for your first run. Samtools view the first 10 lines in each A_HA.bam file for the three sample outputs. Approximately where in the gene are these reads?
2. Do these bam files need to be sorted?
3. Remove the A_HA.bam.bai file and index the HA bam files with samtools index
4. Using ls –lah, how big is A_HA.bam file? Using samtools view and redirect or –o argument, change A_HA.bam to A_HA.sam. How big is A_HA.sam?

Samtools practical

1. Go to the MIRA output for your first run. Samtools view the first 10 lines in each A_HA.bam file for the three sample outputs. Approximately where in the gene are these reads?
2. Do these bam files need to be sorted?
3. Remove the A_HA.bam.bai file and index the HA bam files with samtools index
4. Using ls –lah, how big is A_HA.bam file? Using samtools view and redirect or –o argument, change A_HA.bam to A_HA.sam. How big is A_HA.sam?

Possible answers

1. `samtools view A_HA_H1.bam | head -10`
2. First reads are all mapping to position 1, this is sorted
3. `rm A_HA_H1.bam.bai  && samtools index A_HA_H1.bam`
4. `samtools view -o A_HA_H1.sam A_HA_H1.bam`

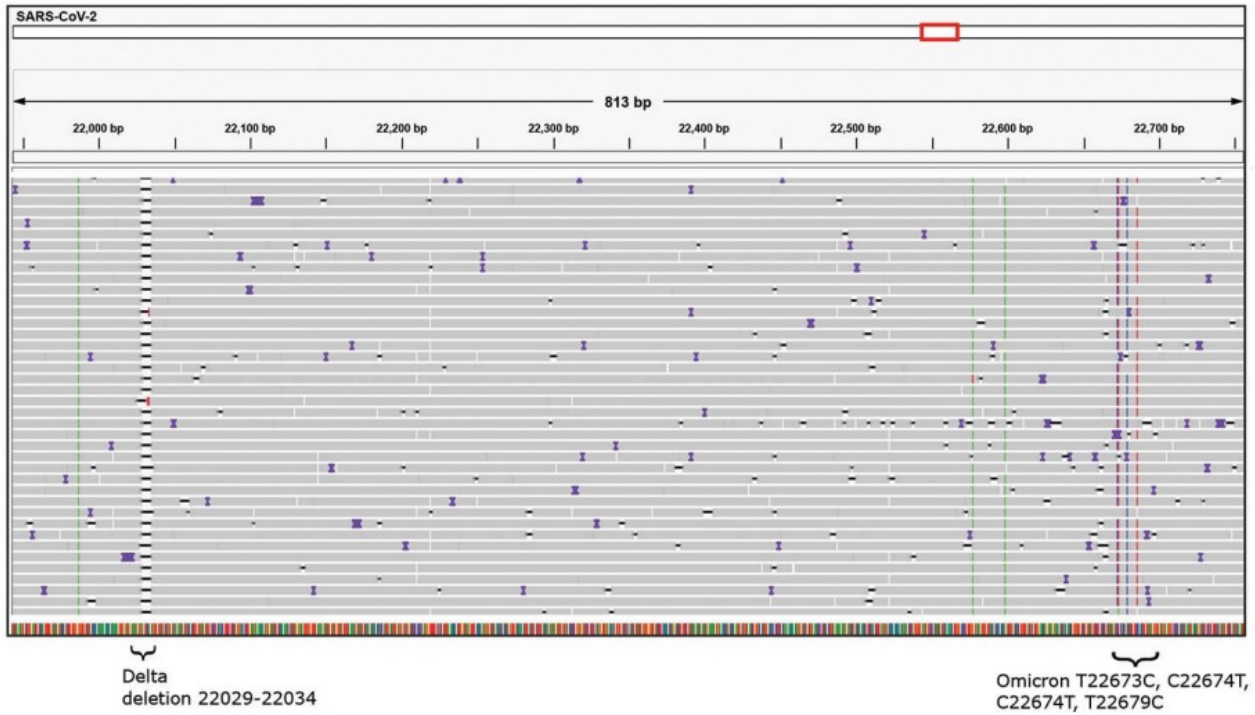
IGV

- Visualize read alignments to a reference
  - See how sequencing reads map across the genome
- Requires sorted and indexed BAM files
  - .bam file + corresponding .bai index
- Inspect alignment features
  - Coverage depth
  - Mismatches and indels
  - Soft clipping and alignment gaps
- Navigate interactively
  - Zoom from whole genome to single-base resolution
  - Jump to coordinates or gene names

IGV

Common QC & Analysis Tasks in IGV

- Check alignment quality
  - Consistent coverage and correct orientation
- Validate variants
  - Confirm SNPs and indels supported by reads
- Identify potential issues
  - Low coverage regions
  - Misalignments or primer artifacts
  - DI Particles



Appendix Figure 2. IGV Alignment of Nanopore long reads from one of the recombinant viruses demonstrating the presence of phased Delta 22029-22034 deletion and Omicron T22673C, C22674T, T22679C, C22686T SNVs originating from a single template. Green, red, and blue lines indicate a substitution with respect to Wuhan-Hu-1, while black bars indicate deletions and purple bars are spurious insertions.

IGV demo

IGV Practical

- Open sample 314ac5ba B_HA.bam in IGV
  - How many mixed sites are there?
  - Is coverage consistent across the gene?

- Open sample 81ae9ee5 A_HA_H3.bam in IGV
  - How many mixed sites are there?
  - Is coverage consistent across the gene?

- Open sample 73ceed0e A_PB1.bam in IGV
  - How many mixed sites are there?
  - Is coverage consistent across the gene?

Intermission

NGS Pipelines part II

Sequence alignment algorithms and BLAST

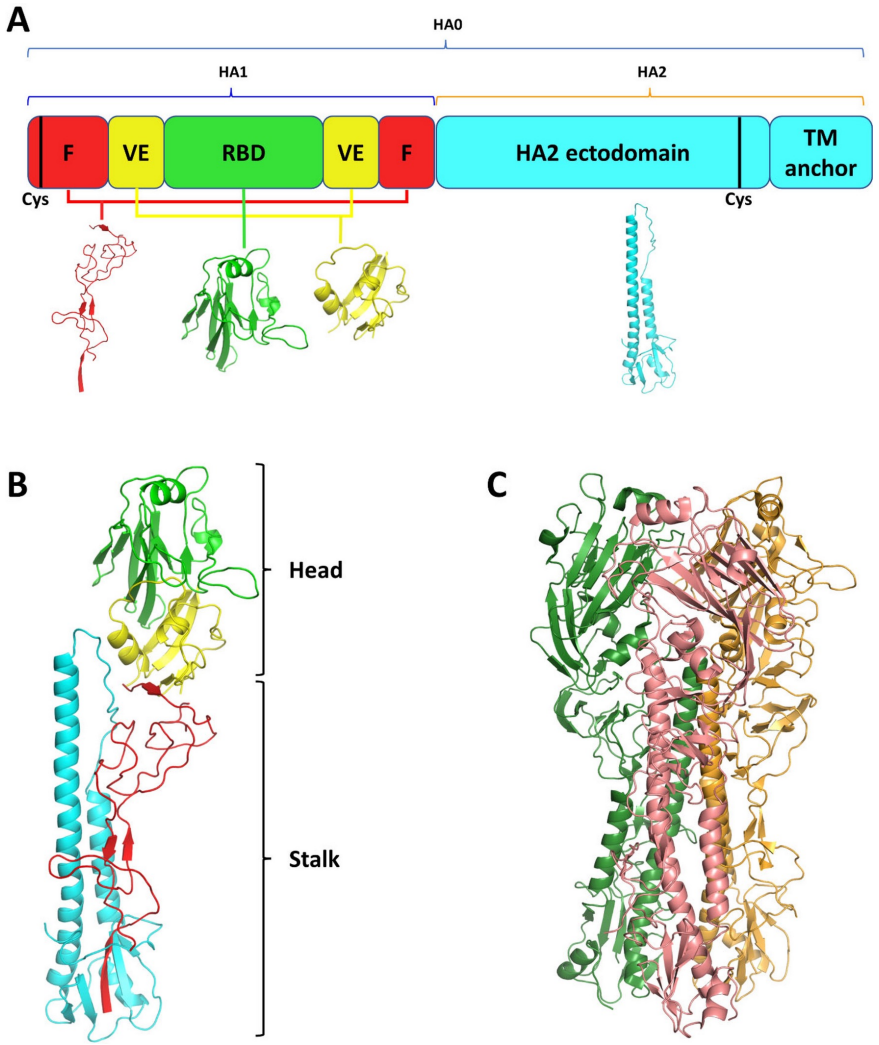
Module Objectives

- BLAST both in web for *ad hoc* analysis and in command-line with a custom BLAST database
- How to identify genetic markers in consensus sequences
- Compare global and local alignment
- Perform multiple sequence alignments using MAFFT
- Extend NGS processing pipeline beyond assembly into by-segment and by-subtype MSAs in preparation for phylogenetics module

Genetic Markers

- When comparing Amino Acid sequences to others, certain mutations are antigenically important
- Antiviral Resistance
- Mammalian adaptation (in avian influenza)
- Antigenic sites

- Take care not to over-interpret genetic data without corresponding phenotypic data



https://www.nature.com/articles/s41598-023-33529-w

MIRA will flag Genetic Markers

- H1 NA: H275Y
  - Only clinically proven site for antiviral resistance

- H5N1 PB2: E627K
  - Enhances replication in mammalian cells

```
nextflow run ./main.nf \
   -profile <profile> \
   --input <RUN_PATH>/samplesheet.csv \
   --outdir <OUTDIR> \
   --runpath <RUN_PATH> \
   --e Find-Positions-Of-Interest \
   --positions_of_interest <filepath>/muts_of_int_table.txt \
   --reference_seq_table <filepath>/ref_table.txt \
   --dais_module <DAIS_MODULE> \
```

Sequence Alignment

Global Sequence Alignment

- Aligns sequences end to end
  - Forces alignment across the full length of both sequences
- Best for similar-length, closely related sequences
  - Whole genes or full genomes
- Penalizes gaps and mismatches across entire sequence
  - Missing regions reduce overall score
- Classic algorithm
  - Needleman–Wunsch

MULTIPLE SEQUENCE ALIGNMENT

Multifasta → multifasta

Local Sequence Alignment

- Aligns the best matching regions only
  - Finds high-similarity subsequences
- Best for sequences of different lengths
  - Reads vs reference, conserved domains
- Ignores poorly matching regions
  - Unaligned ends are not penalized
- Classic algorithm
  - Smith–Waterman

GENOME ASSEMBLY

Fastq → fasta

Pairwise Sequence Alignment

- Pairwise Sequence Alignment
- Compares two sequences at a time
  - Identifies similarity and differences
- Determines optimal alignment
  - Accounts for matches, mismatches, and gaps
- Can be global or local
  - Global → full-length comparison
  - Local → best matching region only
- Used for
  - Comparing a read to a reference
  - Gene-to-gene comparisons
  - Similarity scoring

Multiple Sequence Alignment

Aligns three or more sequences simultaneously
    - Identifies conserved and variable regions
- Used to study evolutionary relationships
    - Compare strains, species, or gene families
- Highlights mutations and conserved motifs
    - Detect SNPs, insertions, deletions
- Common tools
    - **MAFFT :** usage is `mafft in > out`
    - MUSCLE
    - Clustal Omega

- Compare viral genomes across samples
- Build phylogenetic trees
- Identify conserved primer or target regions

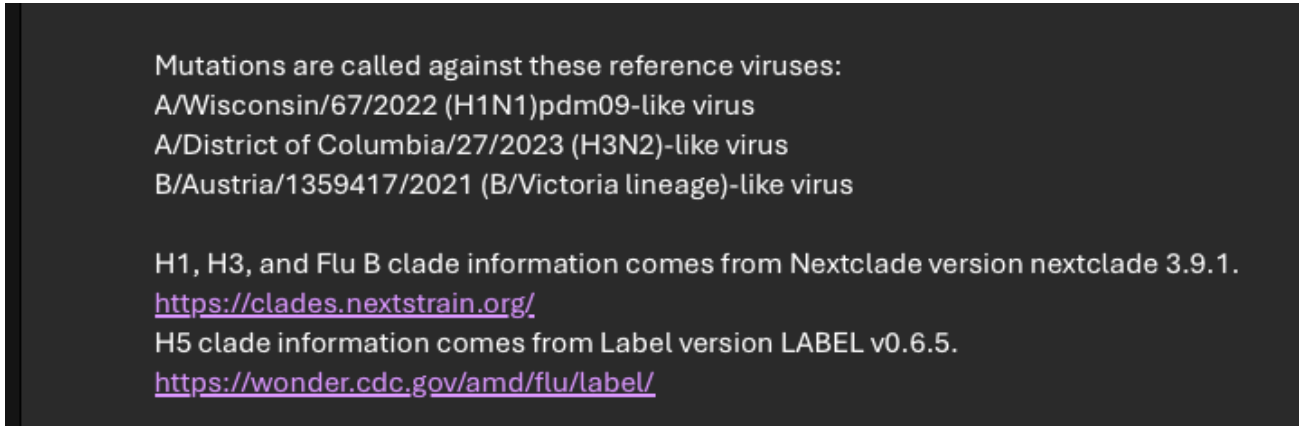
References and Coordinate Space

What is a reference?

- Can be assembly reference
- Can be multiple sequence alignment reference
- Can be mutation reference

Coordinate space is very important for discussing alignments

- Amino acid position #300 can be different between two genomes
  - Insertions and deletions move these numbers
  - Important to specify reference when reporting mutations



Multiple Sequence Alignment Practical

1. From Mira amended_consensus.fasta, extract the H3 HA segments and align them with MAFFT
2. Do any sequences have “gaps” in your MSA?
3. Do the same for A_PB2 segments
4. What happens if you try to do ALL HA segments (H1, H3, H5, and B)?

Multiple Sequence Alignment Practical

1. **From Mira amended_consensus.fasta, extract the H3 HA segments and align them with MAFFT**
2. Do any sequences have “gaps” in your MSA?
3. Do the same for A_PB2 segments
4. What happens if you try to do ALL HA segments (H1, H3, H5, and B)?

Possible Solution

```
grep -A1 HA_H3 aggregate_outputs/mira-
reports/mira_pipeline_test_amended_consensus.fasta | sed "s/--//g" >
H3.fasta

 mafft H3.fasta > aligned_h3.fasta
```

Multiple Sequence Alignment Practical

1. From Mira amended_consensus.fasta, extract the H3 HA segments and align them with MAFFT
2. Do any sequences have “gaps” in your MSA?
3. Do the same for A_PB2 segments
4. **What happens if you try to align ALL HA segments (H1, H3, H5, and B)?**

Multiple Sequence Alignment Pipeline Practical

Add to your run Mira shell script the commands for extracting and aligning HA (A_HA_H1, B_HA, A_HA_H3) and NA (B_NA, A_NA_N1, A_NA_N2) pass QC sequences

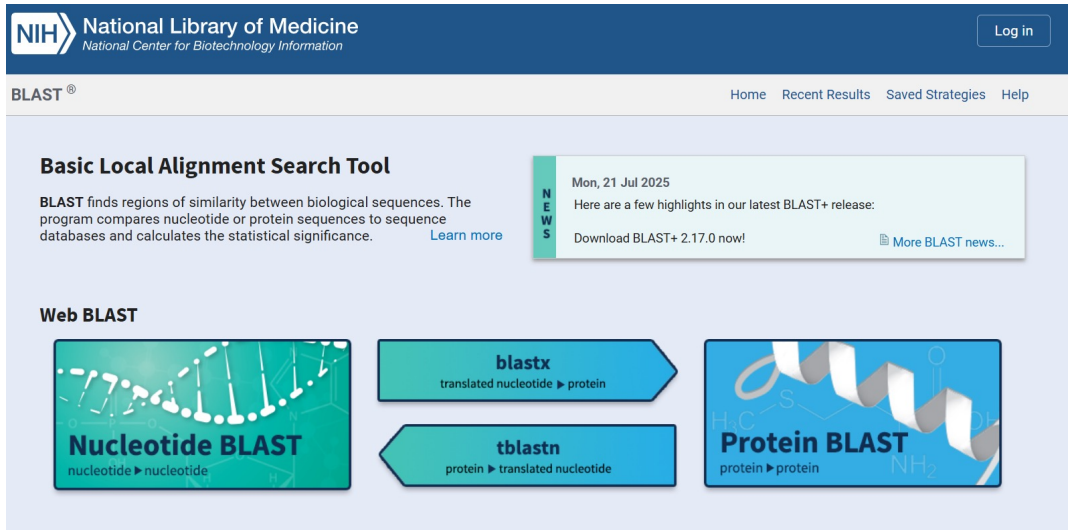
BLAST

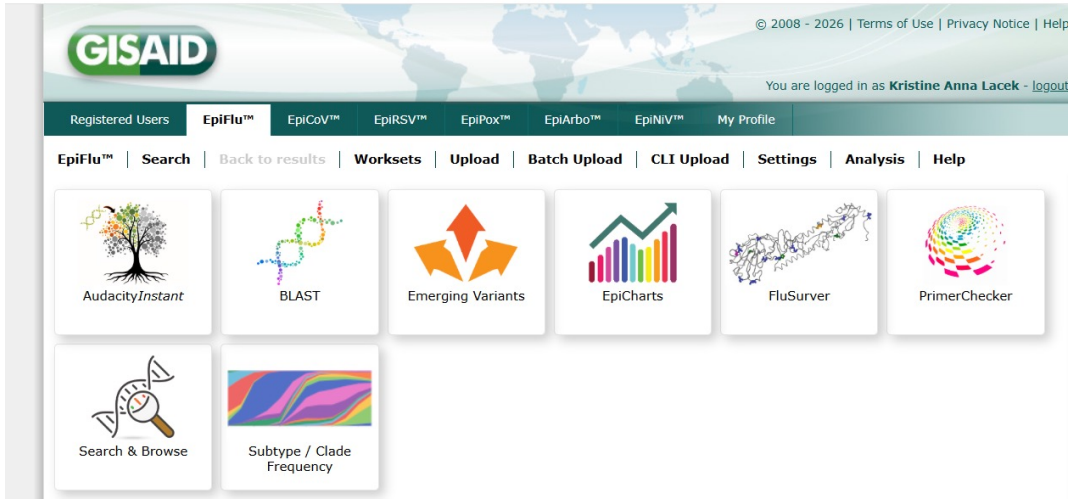
- BLAST = Basic Local Alignment Search Tool
- Finds regions of similarity between sequences
- Compares a query sequence to a database
- Identifies closest matches
- Uses local alignment
- Finds the best matching regions, not full-length alignment
- Widely used in bioinformatics
  - Gene identification
  - Species confirmation
  - Contamination checks

BLAST

- Different BLAST Programs, databases
- blastn
  - Nucleotide vs nucleotide database
- blastp
  - Protein vs protein database
- blastx
  - Translated nucleotide vs protein database
- tblastn
  - Protein vs translated nucleotide database
- megablast
  - Optimized for highly similar sequences

- Databases
  - NCBI BLAST
  - GISAID BLAST
  - Custom Blast Database (CLI)





BLAST

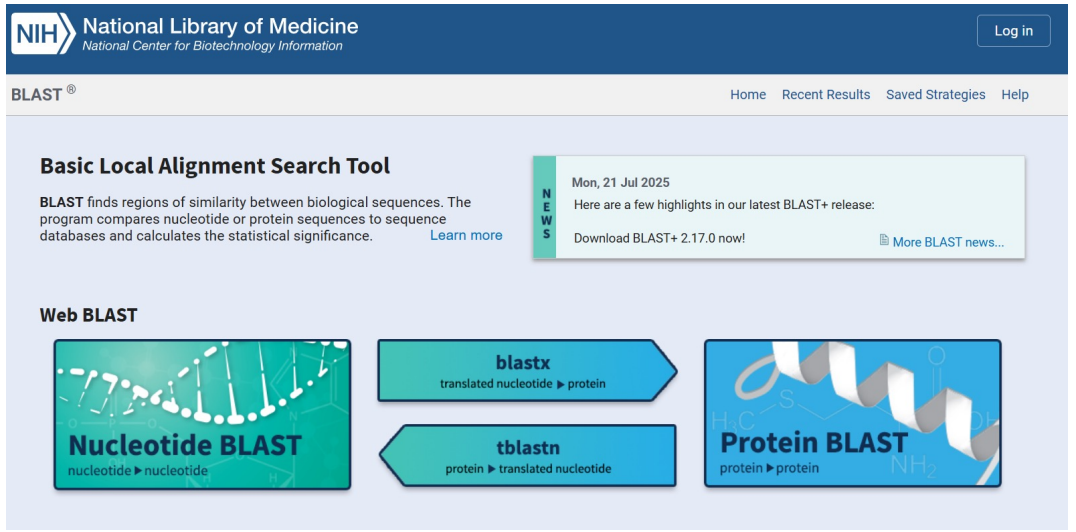
- Databases
  - NCBI BLAST
  - GISAID BLAST
  - Custom Blast Database (CLI)

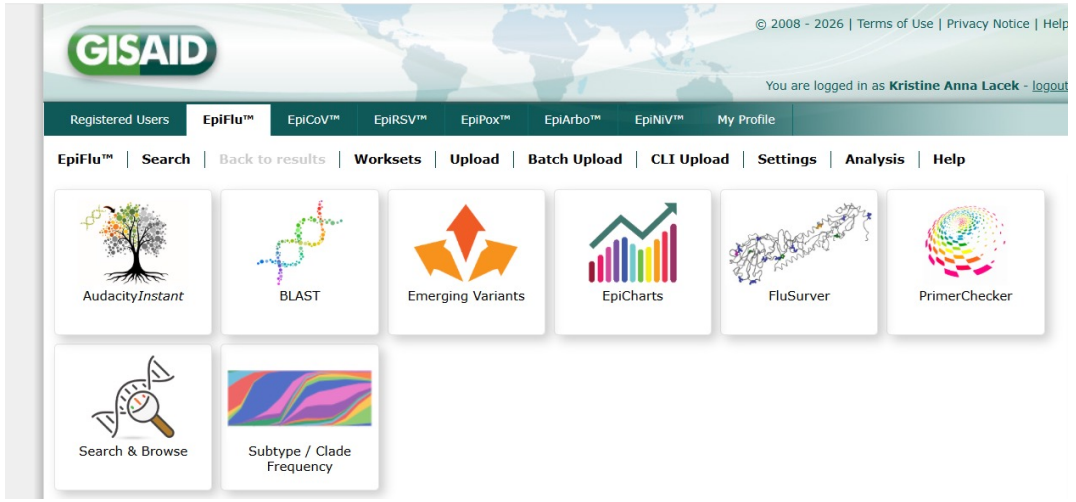
  - NCBI GenBank: open source, INSDC database

  - GISAID: access control, but ingests INSDC samples

  - For a more complete set of flu samples, which BLAST is better?

  - When might an open-source BLAST search be better?





BLAST

Interpreting BLAST output

- Percent Identity → How similar the aligned region is
- E-value → Probability the match occurred by chance
  - Lower = more significant
- Bit Score → Alignment strength (higher = better)
- Alignment Length → How much of the query matched

BLAST CLI

Running BLAST from the CLI

- BLAST+ is the command-line version
  - Installed locally for large datasets or automation
- More reproducible than web BLAST
  - Exact parameters and databases can be recorded
- Common workflow
  - Prepare database → run BLAST → interpret output

Example command

```
blastn -query query.fasta \
       -db nt \
       -out results.txt
```

BLAST CLI

- Creating and Using Local Databases
- Download or prepare reference FASTA
  - Example: novel influenza genomes for diagnostics efficacy
- Create a BLAST database

```
makeblastdb -in reference.fasta \
    -dbtype nucl
```

- Run BLAST against local database

```
blastn -query sample.fasta \
  -db reference.fasta \
   -out results.txt
```

- Faster and ideal for targeted analysis
  - Especially useful in viral genomics

BLAST CLI

- Creating and Using Local Databases
- Download or prepare reference FASTA
    - Example: novel influenza genomes for diagnostics efficacy
- Create a BLAST database

```
makeblastdb -in reference.fasta \
        -dbtype nucl
```

- Run BLAST against local database

```
blastn -query sample.fasta \
    -db reference.fasta \
     -out results.txt
```

- Faster and ideal for targeted analysis
    - Especially useful in viral genomics

Useful CLI Options
Control output format
        -outfmt 6 → tabular format (easy to parse)
Limit results
        -max_target_seqs 5
Adjust sensitivity
        -evalue 1e-5
Set number of threads
        -num_threads 8

BLAST Practical

1. Using NCBI BLAST, which season and region (Americas, Eurasia, Africa, Oceania) does each HA segment most match to
2. Using GISAID BLAST, how do your results differ?
3. Discussion: what are some factors that might influence your results?