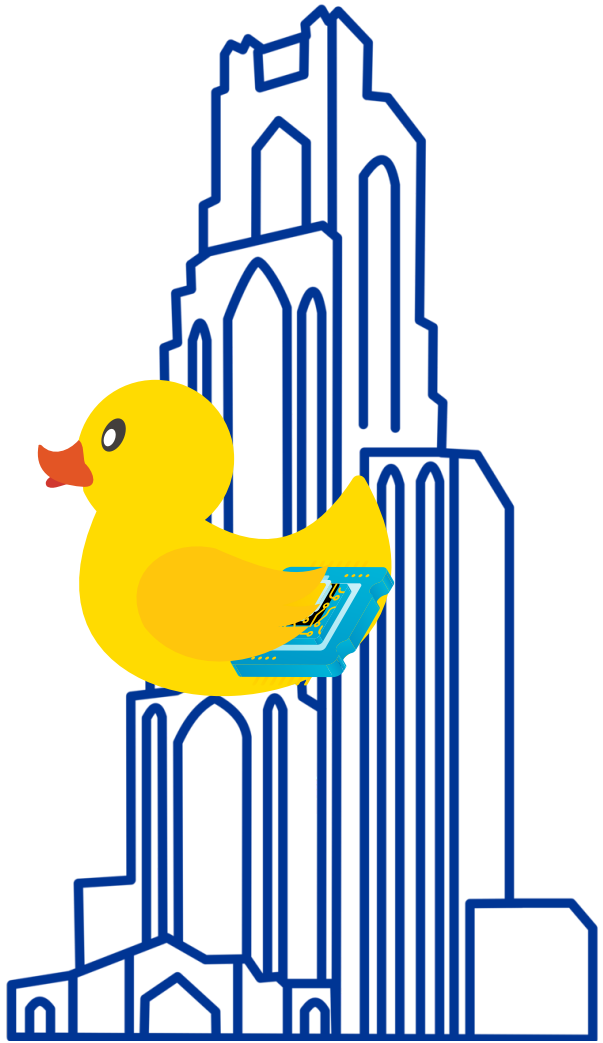


Computational Biology
(BIOSC 1540)

Lecture 02B

DNA sequencing

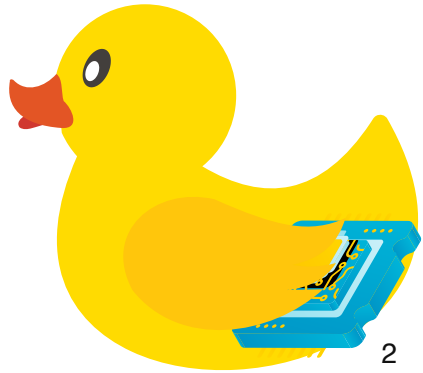
Methodology

Jan 16, 2025

University of Pittsburgh

Announcements

- Assignment P01A is due tomorrow (Jan 17) by 11:59 pm
- Assignment P01B will be released tomorrow (Jan 17) at 11:59 pm
- CByte 01 will be released tomorrow (Jan 17) at 11:59 pm
- Quiz 01 is in two weeks (Jan 28) and will cover lectures 02A to 03B



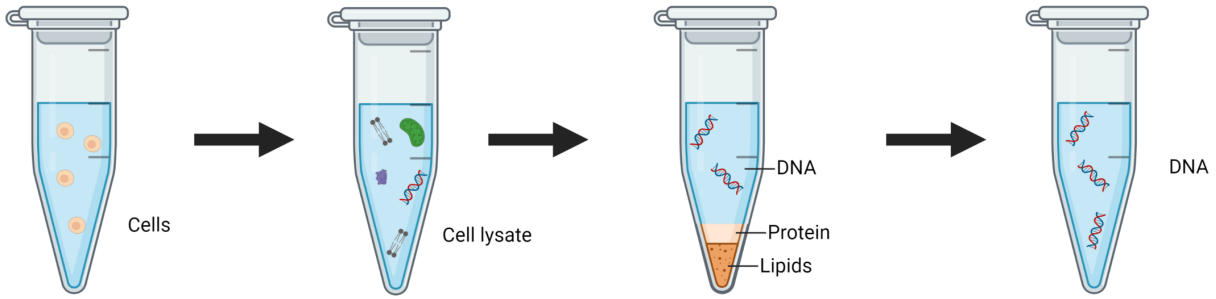
After today, you should have a better understanding of

Sequencing data formats

Databases

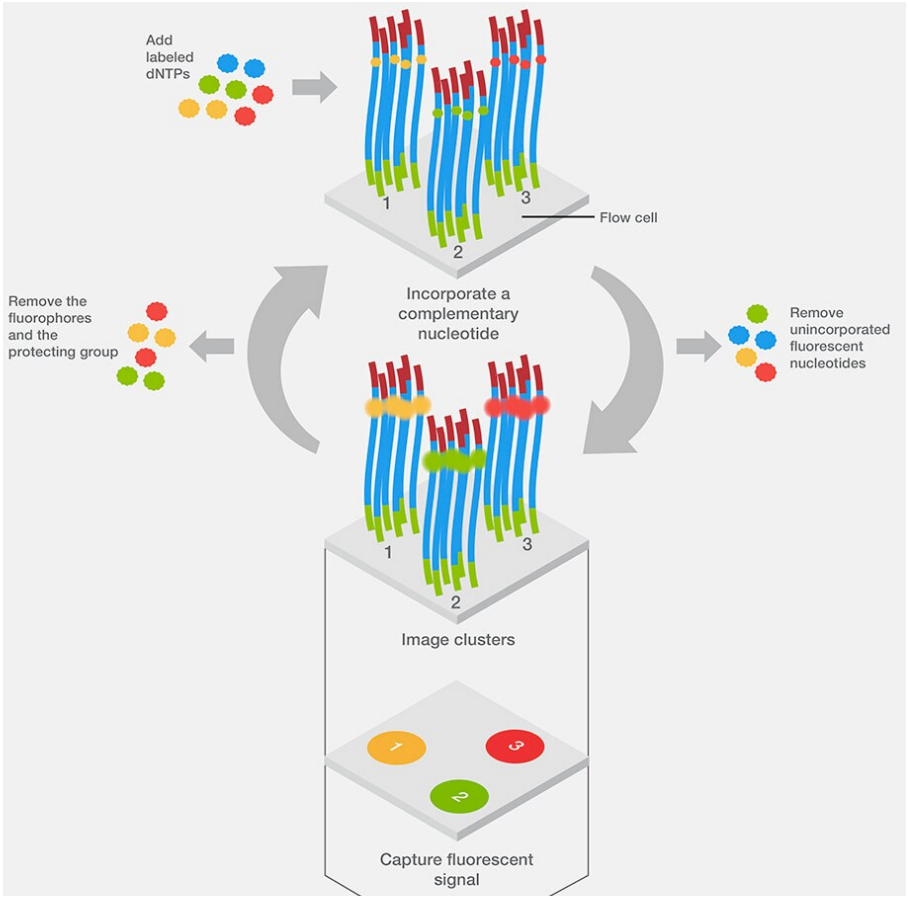
Previous lecture: Experimental side of DNA sequencing

1. Isolate and purify DNA



2. Prepare DNA library

3. Sequence



Where do we put all this data?

And how do we store it?

Genomic sequencing data is archived and stored in public databases

One sequencing run can be 100s of GB

Centralized databases make genomic sequencing data accessible for research and analysis

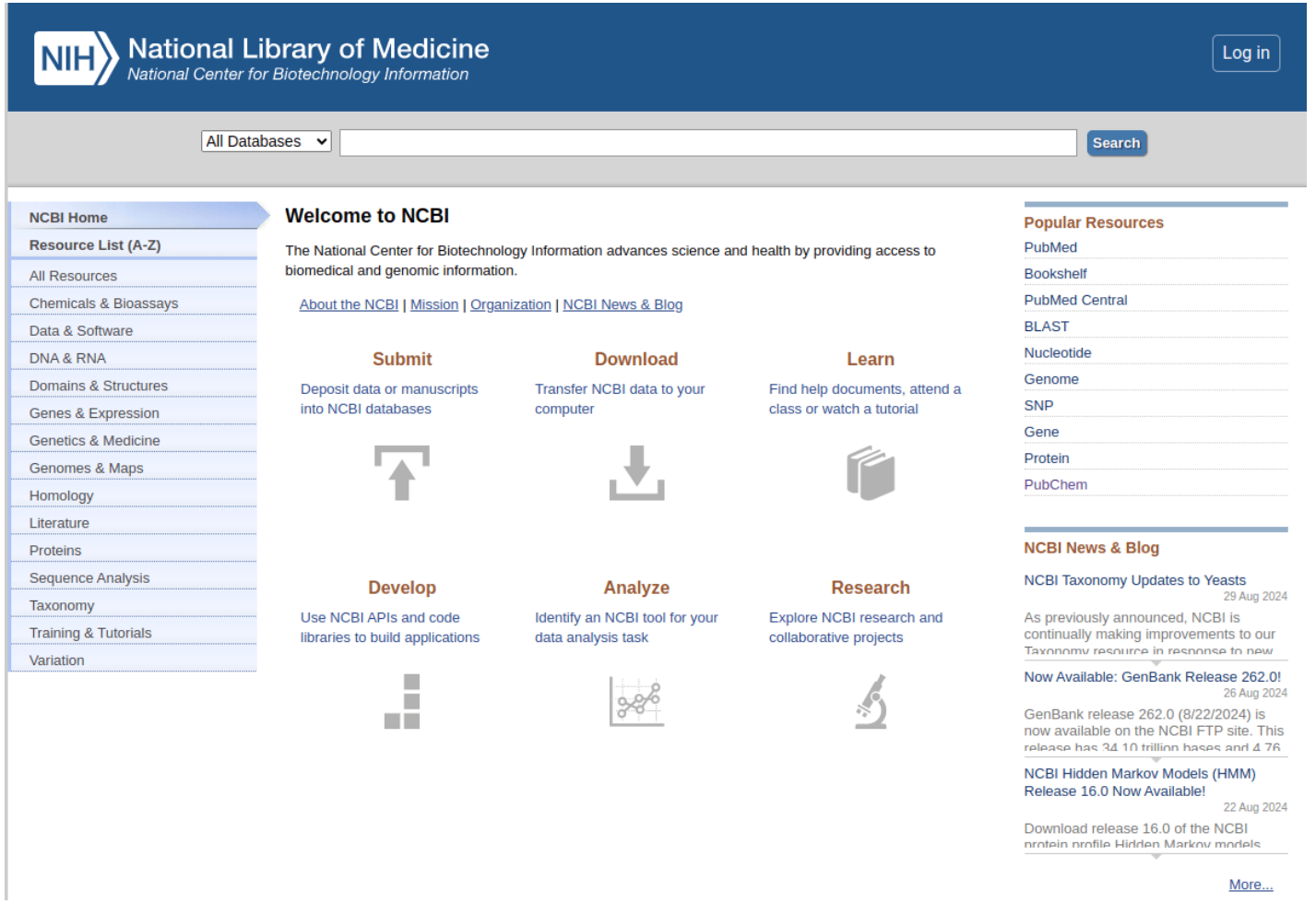
These repositories support collaboration and transparency in science

Just imagine if the only copy of a genome sequence was stored on Bob's 10-year-old external hard drive

National Center for Biotechnology Information (NCBI) is a key source of sequencing data

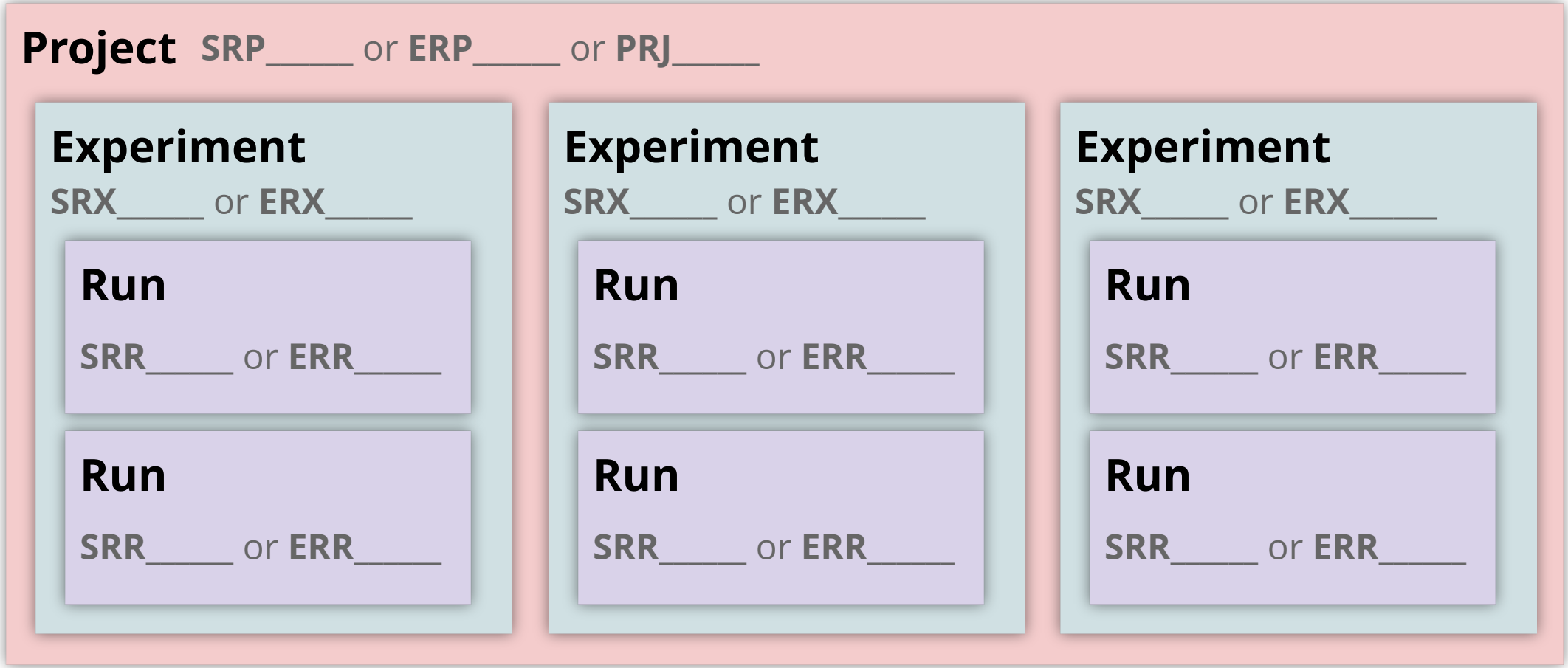
- **GeneBank** for genomic sequences
- **Sequence read archive (SRA)** for sequencing data
- **RefSeq** for reference genomes
- **BioProject** for curated resources for a specific project
- Many more

SRA is what we will use the most



Deposited data are given unique accession numbers

An **accession number** is a unique identifier assigned to a specific dataset, sequence, sample, or experiment in a public database



Example experiment

Run ID: `ERX384810` — Unique identifier for the experiment

Platform: `Illumina HiSeq 2000` — Sequencing instrument used

A **spot** is a unit of data generated during high-throughput sequencing. It typically represents a single **sequencing read** or, in paired-end sequencing, a **pair of reads** from the sequencing instrument.

A **base** refers to a nucleotide (A, T, G, or C) in the sequence data. The total number of bases represents the cumulative length of all reads in a sequencing run.

ERX384810: Illumina HiSeq 2000 paired end sequencing
1 ILLUMINA (Illumina HiSeq 2000) run: 880,387 spots, 176.1M bases, 54.6Mb downloads

Submitted by: University of Oxford, Oxford, UK

Study: Prediction of Staphylococcus aureus antimicrobial resistance from whole genome sequencing
PRJEB5261 • ERP004655 • All experiments • All runs
show Abstract

Sample: C00003197
SAMEA2340030 • ERS397701 • All experiments • All runs
Organism: Staphylococcus aureus

Library:
Name: 1456/11
Instrument: Illumina HiSeq 2000
Strategy: WGS
Source: GENOMIC
Selection: RANDOM
Layout: PAIRED

Runs: 1 run, 880,387 spots, 176.1M bases, 54.6Mb

Run	# of Spots	# of Bases	Size	Published
ERR418513	880,387	176.1M	54.6Mb	2014-02-08

Link

Example run

Illumina HiSeq 2000 paired end sequencing (ERR418513)

Metadata | Alignment | Analysis | Reads | Data access | FASTA/FASTQ download

Run

Run	Spots	Bases	Size	GC Content	Data Status	Published
ERR418513	880.4k	176.1M	54.6MB	34%	Public	2014-02-08

Quality graph (bigger)

This run has 2 reads per spot:

L=100, 100%	L=100, 100%

Legend

Hide attributes

ENA-FIRST-PUBLIC	2014-02-01
ENA-LAST-UPDATE	2018-11-16

Link

After today, you should have a better understanding of

Sequencing data formats

FASTA files

FASTA files store nucleotide or protein sequences in a simple format

Consists of a **header line starting with >** followed by a descriptive identifier

DNA

```
>ERR1197981.1/1
AGTTTGATCCAAAGCATGAGTGTTTACAATGTTTGAATACCTTATACAGT
TCTTATACATACTTTATAAATTATTTCCCAAGCTGTTTTGATACACACAC
```

Contains **one or more lines** of nucleotide or protein sequences

Protein

```
>crab_anapl ALPHA CRYSTALLIN B CHAIN (ALPHA(B)-CRYSTALLIN)
MDITIHNPLIRRPLFSWLAPSRIFDQIFGEHLQESELLPASPSLSPFLMR
SPIFRMPSWLETGLSEMRLEKDKFSVNLDVKHFSPEELKVKVLGDMVEIH
GKHEERQDEHGFIAREFNRKYRIPADVDPLTITSSLSLDGVLTVSAPRKQ
SDVPERSIPITREEKPAIAGAQRK
```

FASTA files help us use Python to analyze DNA sequences

Calculating Sequence Length

```
1 DNA_SEQ = "GAGCAATTTCGAGGATCGCTTGTTGGTATTACTCGGGCTTTTC"
2 seq_len = len(DNA_SEQ)  # 43
```

DNA_SEQ: The DNA sequence is stored as a **string** in Python

Each character in the string
corresponds to a nucleotide

The `len()` function returns the number of characters in the string

In this context, it calculates the total
number of nucleotides in the sequence

Counting Nucleotides with a Function

`count_nucleotides(seq)` takes a string that represents a DNA sequence as input

A dictionary is initialized with keys for the nucleotides (`A`, `T`, `G`, `C`) and values set to 0.

- This structure allows for efficient updating of counts.

The updated dictionary with nucleotide counts is returned as the function's output.

Functions in Python allow us to organize reusable blocks of code for specific tasks.

```python
def count_nucleotides(seq):
    nucleotide_counts = {"A": 0, "T": 0, "G": 0, "C": 0}

    for nucleotide in seq:
        if nucleotide in nucleotide_counts:
            nucleotide_counts[nucleotide] += 1

    return nucleotide_counts
```

Breaking Down the For Loop

Strings in Python are iterable, meaning you can process one character at a time using a `for` loop.

- Here, the loop processes each nucleotide in the DNA sequence.

```python
def count_nucleotides(seq):
    nucleotide_counts = {"A": 0, "T": 0, "G": 0, "C": 0}

    for nucleotide in seq:
        if nucleotide in nucleotide_counts:
            nucleotide_counts[nucleotide] += 1

    return nucleotide_counts
```

Conditional Check:

- `if nucleotide in nucleotide_counts` ensures that only valid nucleotides (`A`, `T`, `G`, `C`) are processed.
- Prevents errors from unexpected characters like `N` or other symbols.

Updating Dictionary Values:

- `nucleotide_counts[nucleotide] += 1` increments the count for the current nucleotide.
- Each nucleotide's count is stored as the dictionary's value, which is updated dynamically during the loop.

Results and Their Significance

Output: The dictionary shows the counts of each nucleotide in the sequence.

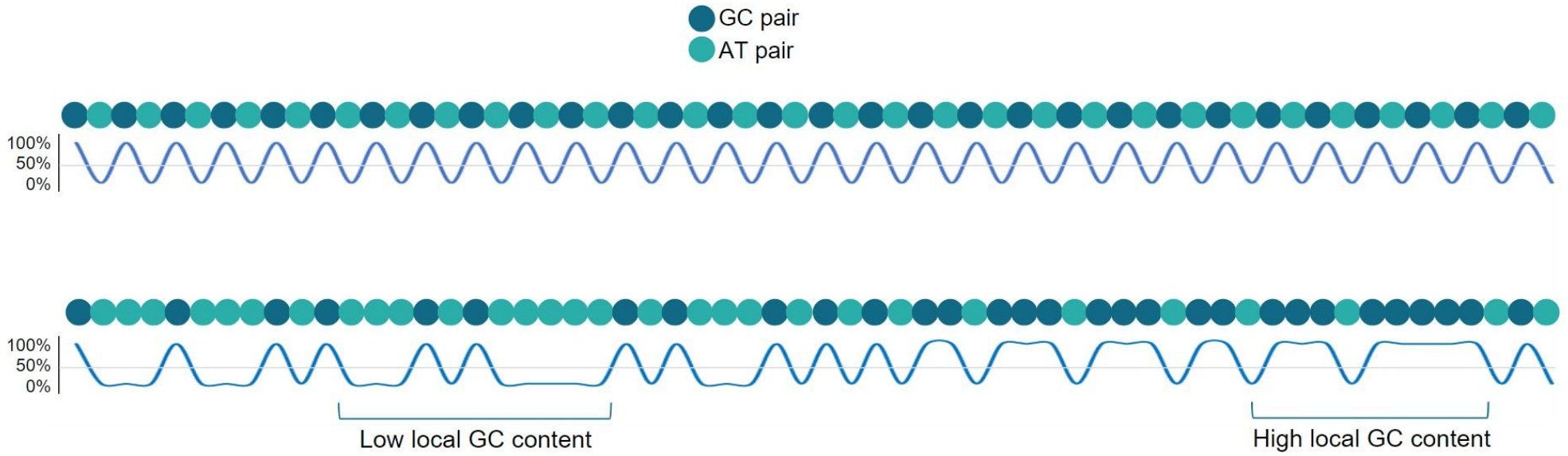
```
1  def count_nucleotides(seq):
2      nucleotide_counts = {"A": 0, "T": 0, "G": 0, "C": 0}
3
4      for nucleotide in seq:
5          if nucleotide in nucleotide_counts:
6              nucleotide_counts[nucleotide] += 1
7
8      return nucleotide_counts
9
10 nuc_counts = count_nucleotides(DNA_SEQ)
11 # {'A': 7, 'T': 16, 'G': 12, 'C': 8}
```

What is GC Content?

GC content is the percentage of guanine (G) and cytosine (C) nucleotides in a DNA sequence

Biological Relevance:

- High GC content indicates stronger hydrogen bonding, leading to more stable DNA.
- GC content variation is useful for species identification, genome analysis, and detecting contamination.



Python Function to Calculate GC Content

```python
def get_gc_content(seq):
    nuc_counts = count_nucleotides(seq)
    gc_content = (nuc_counts["G"] + nuc_counts["C"]) / len(seq) * 100
    return gc_content
```

`get_gc_content(seq)` calculates the percentage of G and C nucleotides in a given sequence.

The function calls `count_nucleotides(seq)` to get a dictionary of nucleotide counts, demonstrating code reuse.

Percentage of G and C nucleotides are then calculated and returned

Breaking Down the GC Content Calculation

The `nuc_counts` dictionary contains the counts for each nucleotide.

- Access specific counts using keys: `nuc_counts["G"]` and `nuc_counts["C"]`.

```
def get_gc_content(seq):
    nuc_counts = count_nucleotides(seq)
    gc_content = (nuc_counts["G"] + nuc_counts["C"]) / len(seq) * 100
    return gc_content
```

- Divide by `len(seq)` to calculate the proportion of G and C bases in the sequence.
- Multiply by 100 to convert the proportion to a percentage.

Interpreting the Output

```
1 def get_gc_content(seq):
2     nuc_counts = count_nucleotides(seq)
3     gc_content = (nuc_counts["G"] + nuc_counts["C"]) / len(seq) * 100
4     return gc_content
5
6 seq_gc_content = get_gc_content(DNA_SEQ)
7 # 46.51162790697674
```

The calculated GC content for the sequence `DNA_SEQ` is approximately **46.5%**.

After today, you should have a better understanding of

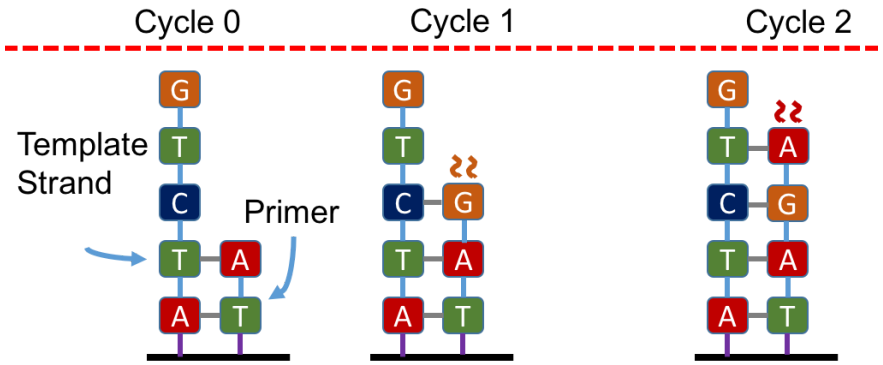
Sequencing data formats

FASTQ files

Sequencing methods can introduce potential errors

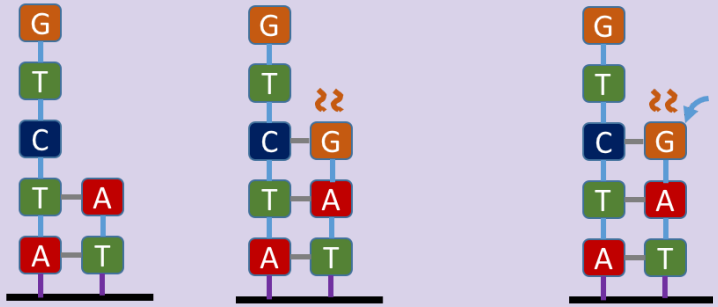
Lagging and Leading Synthesis

Normal sequencing by synthesis (i.e., Illumina) has each cluster strand in sync to amplify signal

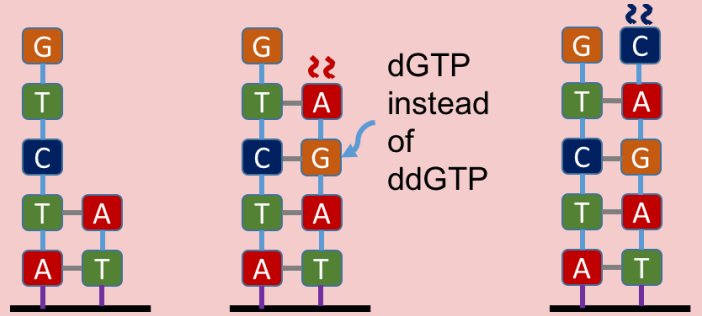


Errors can occur when synthesis gets out of sync

Lagging synthesis by failure to remove blocking fluorophore.

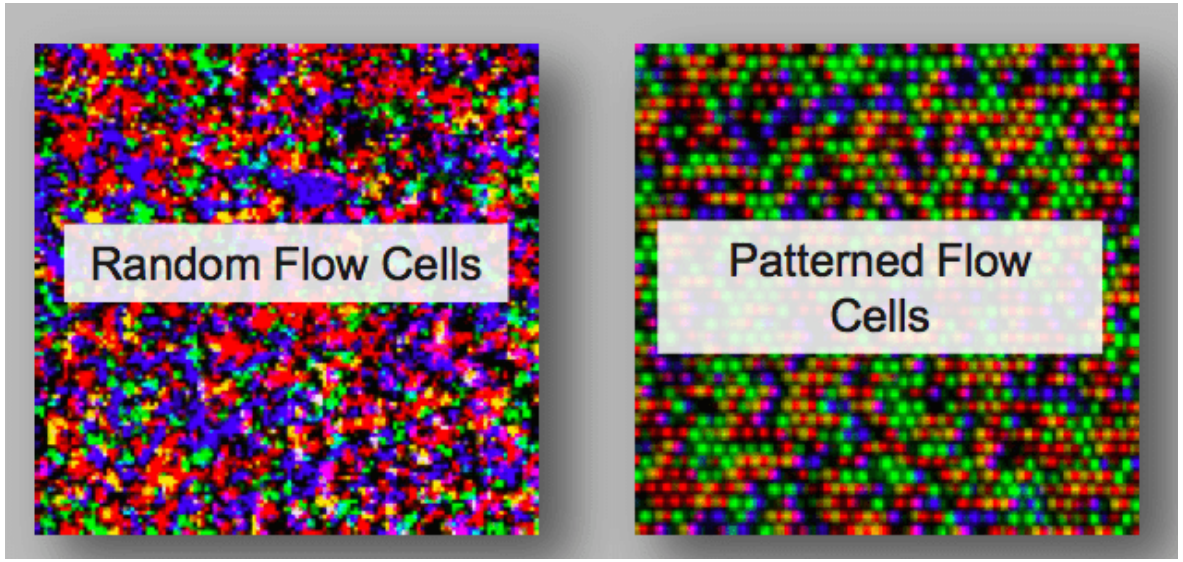


Leading synthesis by addition of dNTP instead of ddNTP



Densely packed flow cells can also cause signal overlap

Fluorescent signals from neighboring clusters can overlap



Clean

Noisy

Signal-cross talk degrades quality

FASTQ files store sequencing reads and their quality scores

Combines **sequence data** (as in FASTA) with **quality scores** for each nucleotide.

Organized in a four-line structure for each read:

1. **Header**: Contains the read identifier, starting with `@`.
2. **Sequence**: The nucleotide sequence.
3. **Separator**: A + line (optionally repeating the header).
4. **Quality Scores**: Encoded in ASCII characters.

```
1 @HWI-M01876:76:000000000-AF16W:1:1101:10853:1000 1:N:0:CGTGACAGAT
2 NTGTACTTCATCCGAAACTCGTGCTCATCTCTGCTCAGATCGGAAGAGCACACGTCTGAACTCCAG
3 +
4 #8ABCFGGGFCEDCFGGGGGGGFFCGEFGGGGGGFGGGGGGGGDEFGGGGGGGGGGGGGGGGFFF
5 @HWI-M01876:76:000000000-AF16W:1:1101:16471:1000 1:N:0:CGTGAACTTG
6 NTTCCAGATATTCGATGCATGTGCCGCTCCTGTCGGAGATCGGAAGAGCACACGTCTGAACTCCAG
7 +HWI-M01876:76:000000000-AF16W:1:1101:16471:1000 1:N:0:CGTGAACTTG
8 y8BCCGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGEGGGGFGGGGGGGGGGGGGGG
```

Quality scores in FASTQ files represent sequencing confidence

Quality scores are associated with each base and indicate the probability of a sequencing error.

Encoded using the **Phred quality score system**:

- Higher scores = Higher confidence = Lower error probability.

Lowest quality: !

Highest quality: ~

Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL (null)	32	SPACE	64	@	96	`
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	"	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	'	71	G	103	g
8	BS (backspace)	40	(	72	H	104	h
9	TAB (horizontal tab)	41	)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[	123	{
28	FS (file separator)	60	<	92	\	124	|
29	GS (group separator)	61	=	93	]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL

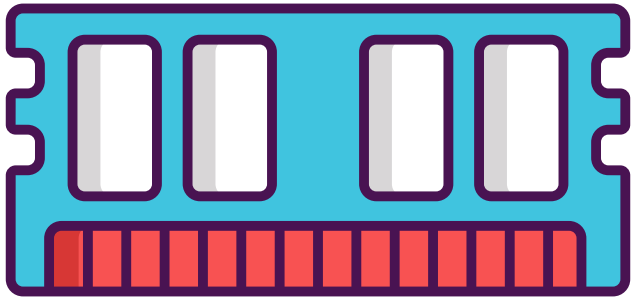
Why subtract 33 from ASCII values in FASTQ?

We are not able to see values below 33 on our screen, making it harder to work with

Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL (null)	32	SPACE	64	@	96	`
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	"	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	'	71	G	103	g
8	BS (backspace)	40	(	72	H	104	h
9	TAB (horizontal tab)	41	)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[	123	{
28	FS (file separator)	60	<	92	\	124	|
29	GS (group separator)	61	=	93	]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL

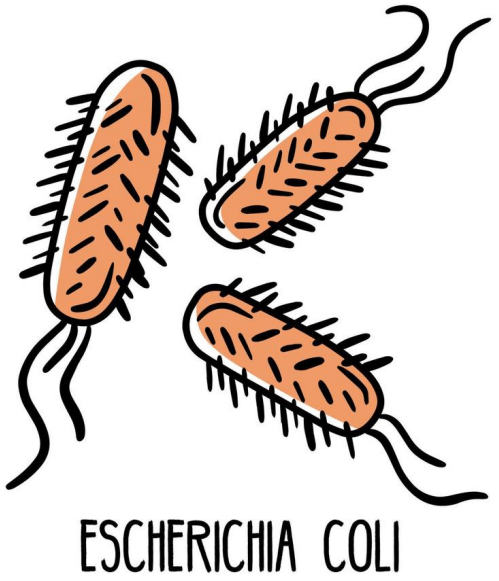
Why use ASCII encoding for quality scores?

We need to store millions upon millions of floats (e.g., 0.92829) per nucleotide



One million float32 values are about 3.8 MB

Seems small, but one *E. coli* genome is ~5 million base pairs, and our DNA library will have multiple copies of sequences



ASCII characters require ~1/4 the memory, making them more memory efficient

ASCII characters map directly to numerical quality scores

Phred quality (Q) is the integer associated with the ASCII symbol

$$P = 10^{-Q/10}$$

Probability that an error occured

Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL (null)	32	SPACE	64	@	96	`
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	"	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	'	71	G	103	g
8	BS (backspace)	40	(	72	H	104	h
9	TAB (horizontal tab)	41	)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[	123	{
28	FS (file separator)	60	<	92	\	124	|
29	GS (group separator)	61	=	93	]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL

Examples

!

$$P = 10^{-(33-33)/10} = 1.0$$

?

$$P = 10^{-(63-33)/10} \approx 0.001$$

Phred quality scores help identify reliable bases

Phred scores relate directly to confidence:

- $Q=20$: 99% accuracy (1 in 100 chance of error).
- $Q=30$: 99.9% accuracy (1 in 1,000 chance of error).
- $Q=40$: 99.99% accuracy (1 in 10,000 chance of error).

Bases with low Phred scores may need filtering to improve data quality

Sequencing runs store millions of FASTQ entries

After today, you should have a better understanding of

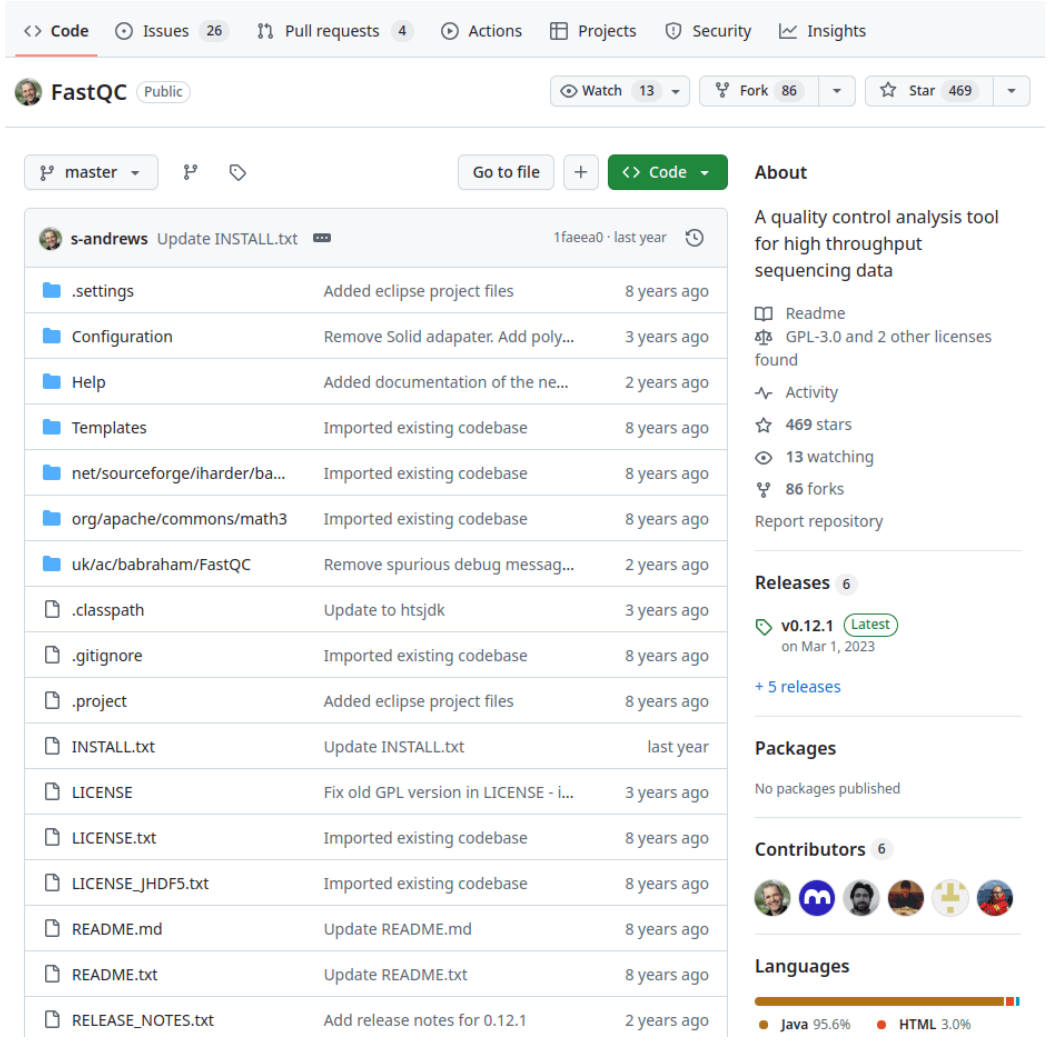
Assessing sequencing data quality

FastQC

FastQC provides essential insights into sequencing data quality

Identifies issues affecting sequencing data reliability

Helps determine preprocessing steps before downstream analysis

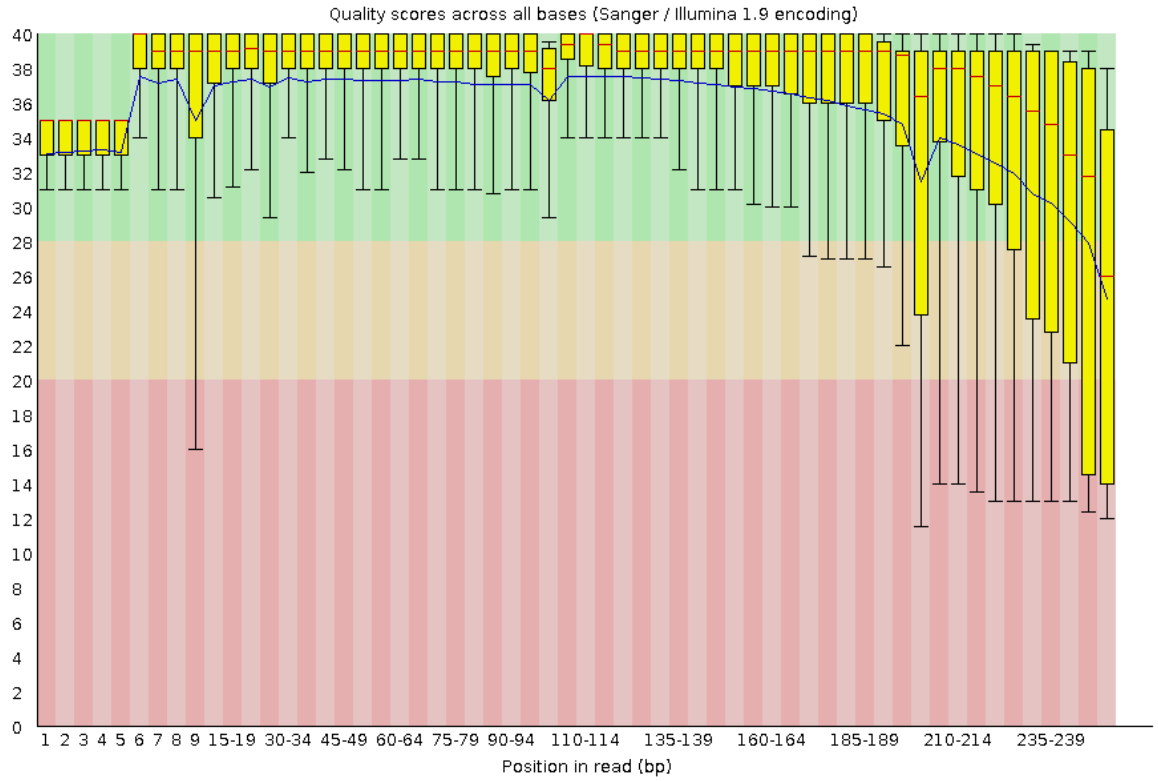


github.com/s-andrews/FastQC

Per Base Sequence Quality assesses the accuracy of base calls

Plot Description:

- X-axis: Base position in the read.
- Y-axis: Phred quality scores (higher scores = better quality).
- Boxplot features: Median (red line), interquartile range (yellow box), 10-90% range (whiskers), mean (blue line).



Key Observations:

- Early bases may show slightly lower quality due to priming biases.
- Quality often decreases at the 3' end due to signal decay and phasing issues.

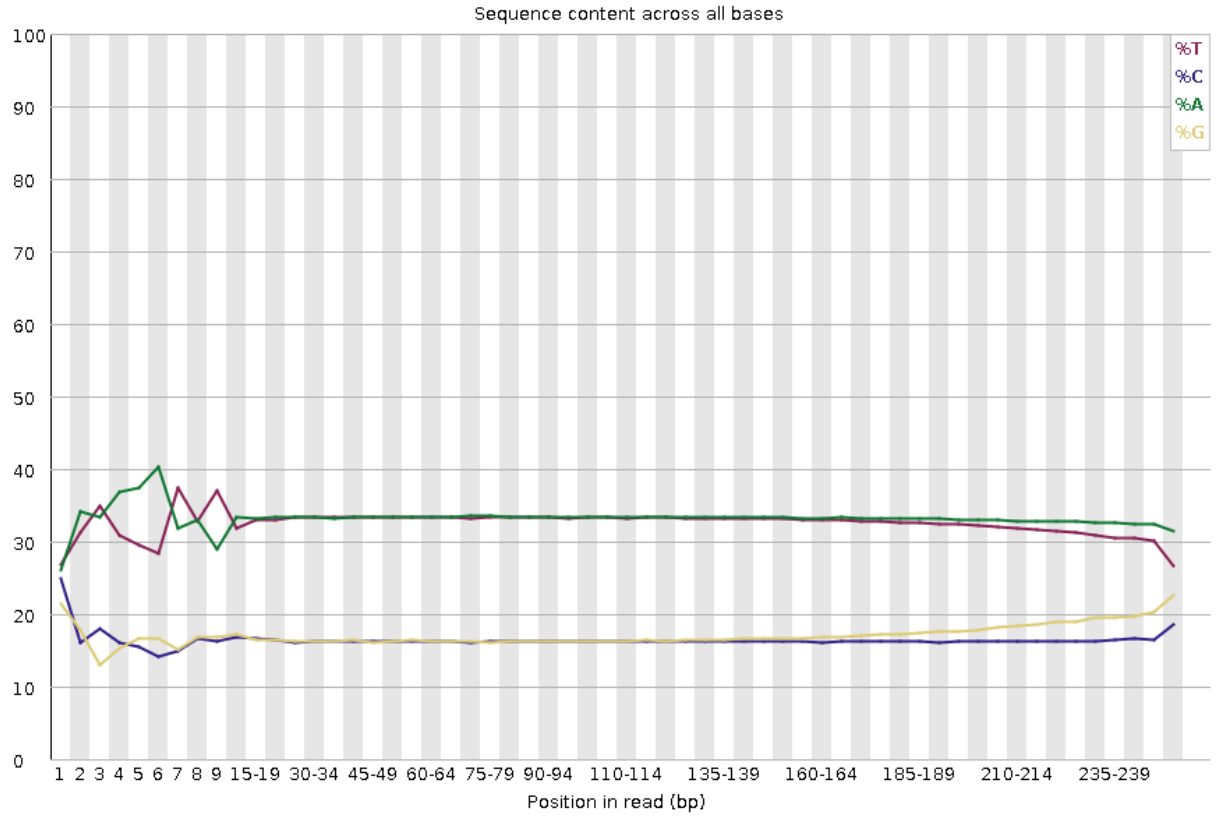
Per Base Sequence Content checks nucleotide composition bias

Plot Description:

- X-axis: Position in the read.
- Y-axis: Percentage of each nucleotide (A, T, G, C).

Key Observations:

- A random library should show equal A-T and G-C proportions, with parallel lines across
- Biases across the entire read indicate contamination or library preparation issues.



Per Base N Content identifies ambiguous base calls

Plot Description:

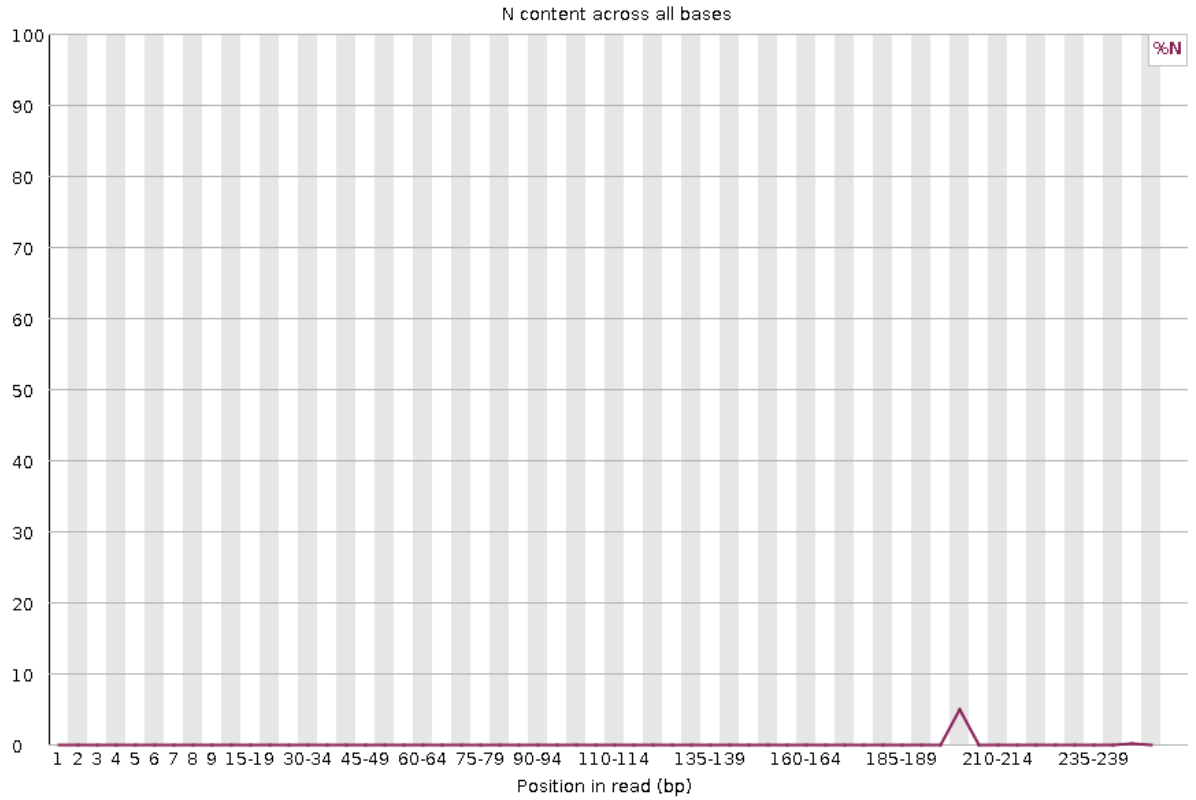
- X-axis: Base position in the read
- Y-axis: Percentage of `N` calls

Key Observations:

- High `N` content near the end of reads suggests sequencing issues like signal decay.

Interpretation:

- Acceptable: Flatline near 0%.
- High N content (>5%): Reads may require trimming or removal.



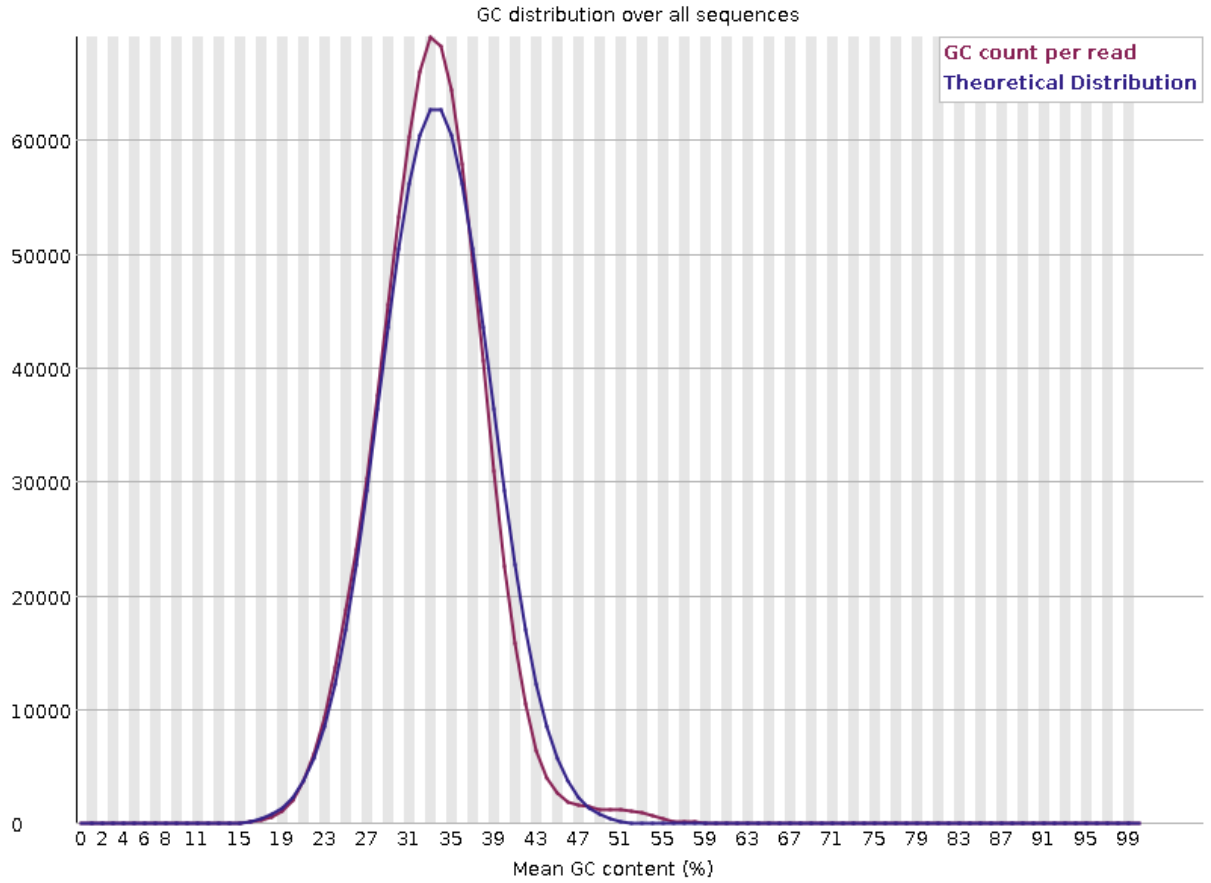
Per Sequence GC Content identifies abnormal nucleotide composition

Plot Description:

- X-axis: GC content percentage.
- Y-axis: Number of reads with the given GC content.

Key Observations:

- Normal distribution with a single peak indicates uniform GC content.
- Bimodal or skewed distributions suggest contamination or systematic bias.
- Single peak at genome-specific GC content is expected



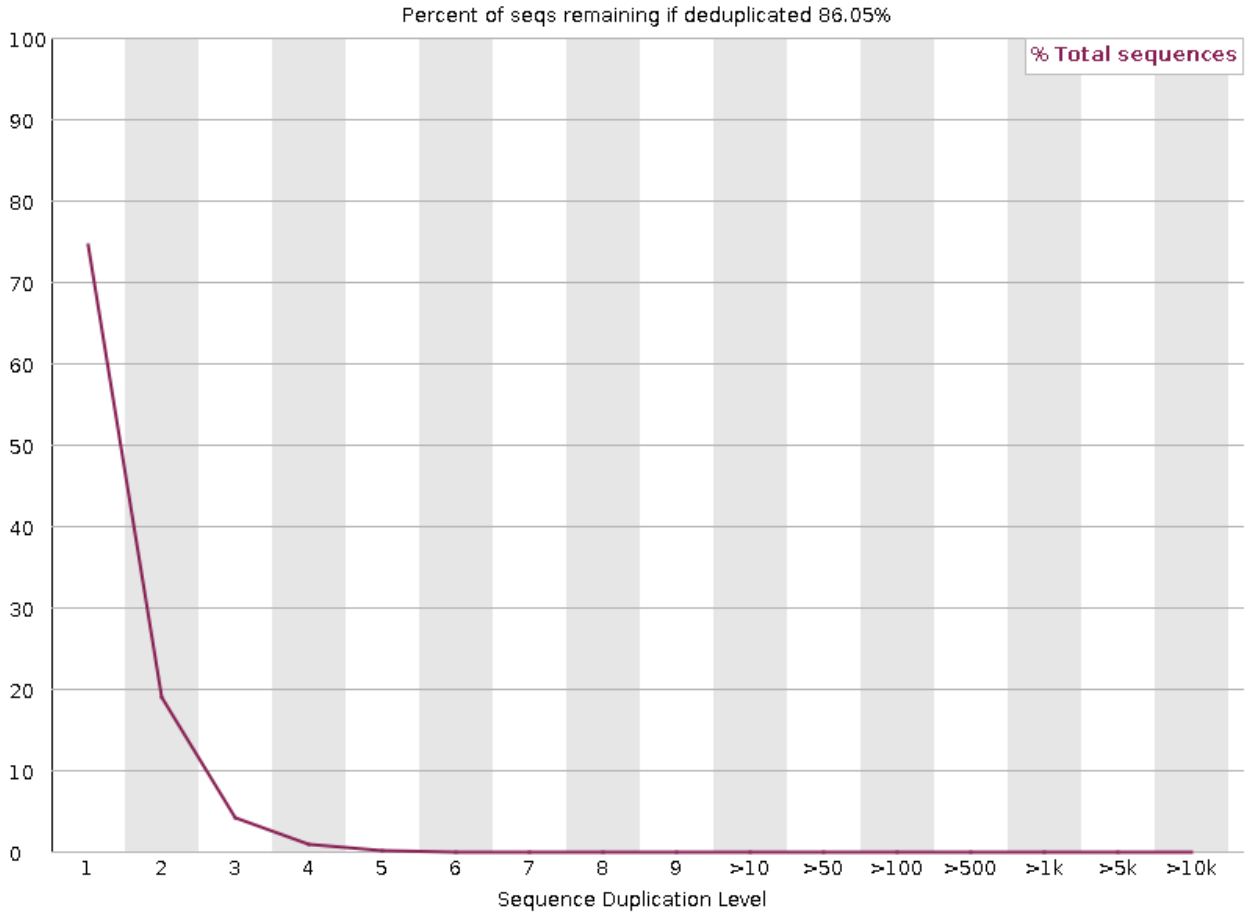
Sequence Duplication Levels highlight library diversity

Plot Description:

- X-axis: Duplication level (number of times a sequence appears).
- Y-axis: Percentage of sequences.

Key Observations:

- High duplication levels suggest overrepresentation due to PCR artifacts or highly abundant transcripts.



After today, you should have a better understanding of

Cleaning and preprocessing sequencing data

fastp

After accessing the quality of sequencing data, we can preprocess reads to correct some issues

Not all errors in sequencing data are correctable

Correctable Issues:

- Adapter contamination: Removed with trimming tools like Fastp.
- Low-quality bases: Corrected or trimmed based on quality scores.
- Read duplication: Resolved by deduplication to ensure unique data.

Non-Correctable Issues:

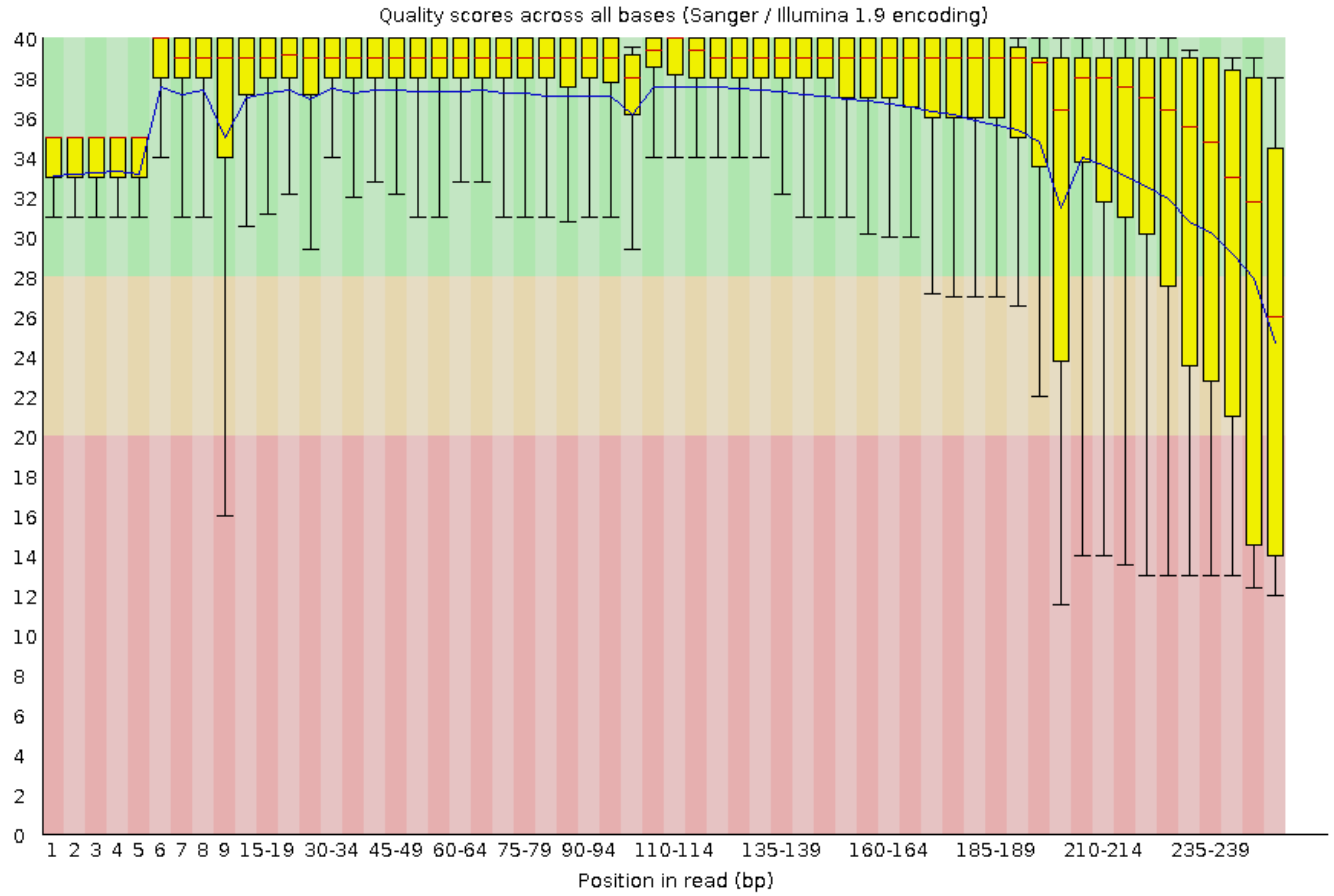
- Systematic biases or overclustering in sequencing runs.
- Severe signal decay or phasing errors.
- Insufficient coverage or failed experiments.

Quality filtering removes unreliable bases and reads

Bases with low Phred scores are more likely to have sequencing errors.

Reads are often filtered based on:

- Minimum base quality thresholds (e.g., Q20 or Q30).
- Average read quality.
- Proportion of ambiguous bases (`N` calls).



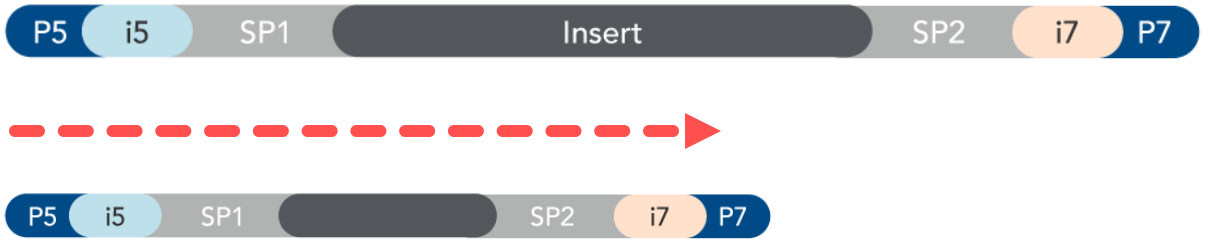
Adapter trimming removes contamination from sequencing adapters

Adapters are artificial sequences added during library preparation.

If the library insert is shorter than the read length, the adapter sequence is read.

Suppose our sequencing read length is this long

Is the adapter included in the sequencing read?



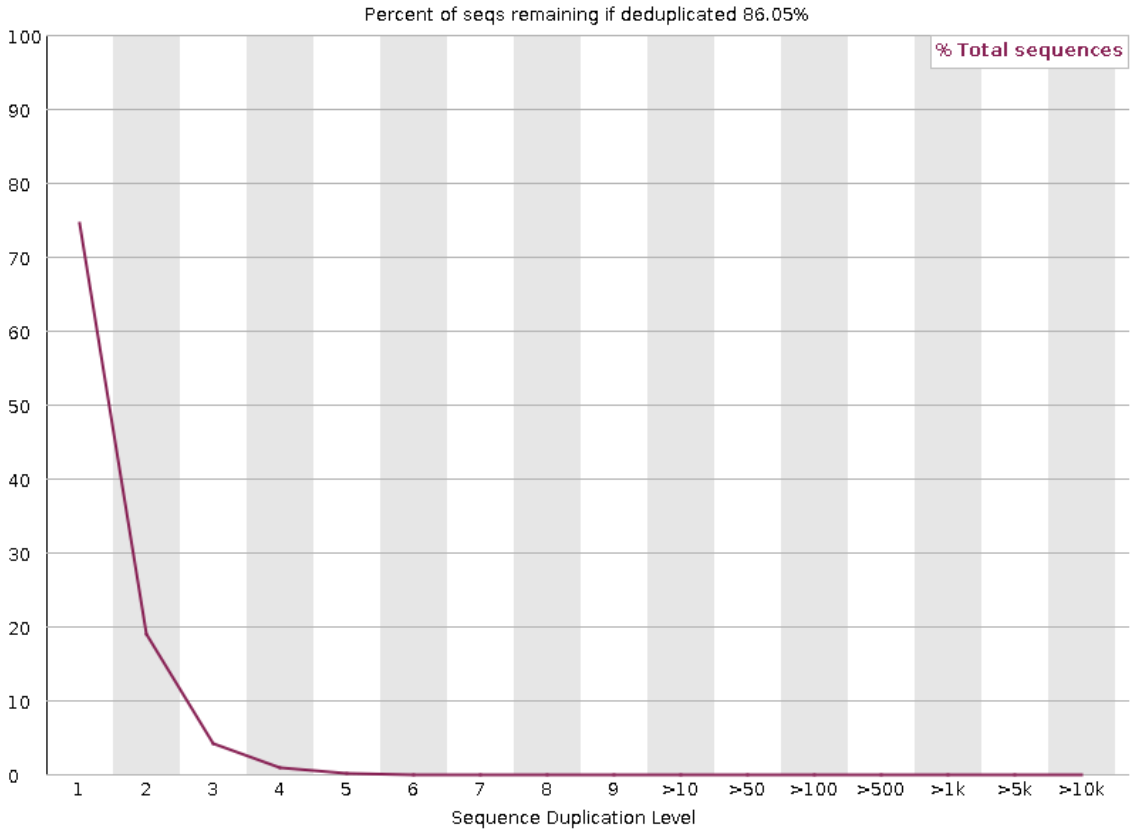
No

Yes

Deduplication ensures library diversity by removing redundant reads

Source of Duplicates: PCR amplification biases.

Identifies and removes identical sequences, ensuring unique data.

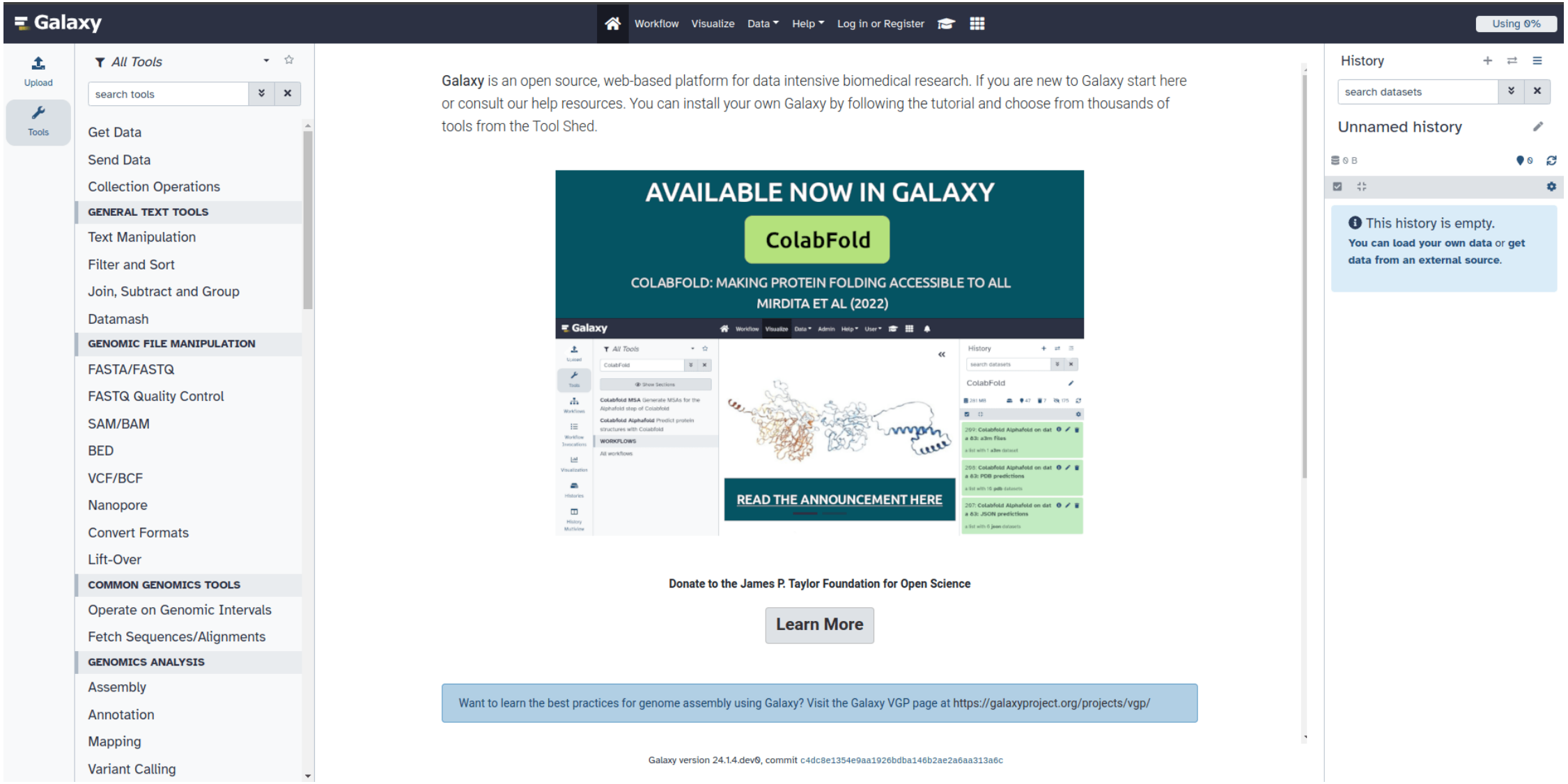


After today, you should have a better understanding of

Assessing sequencing data quality

Galaxy Platform

Galaxy


Galaxy
Workflow
Visualize
Data
Help
Log in or Register
Using 0%
Upload
Tools
All Tools
search tools
Get Data
Send Data
Collection Operations
GENERAL TEXT TOOLS
Text Manipulation
Filter and Sort
Join, Subtract and Group
Datamash
GENOMIC FILE MANIPULATION
FASTA/FASTQ
FASTQ Quality Control
SAM/BAM
BED
VCF/BCF
Nanopore
Convert Formats
Lift-Over
COMMON GENOMICS TOOLS
Operate on Genomic Intervals
Fetch Sequences/Alignments
GENOMICS ANALYSIS
Assembly
Annotation
Mapping
Variant Calling
Galaxy is an open source, web-based platform for data intensive biomedical research. If you are new to Galaxy start here or consult our help resources. You can install your own Galaxy by following the tutorial and choose from thousands of tools from the Tool Shed.
AVAILABLE NOW IN GALAXY
ColabFold
COLABFOLD: MAKING PROTEIN FOLDING ACCESSIBLE TO ALL
MIRDITA ET AL (2022)
READ THE ANNOUNCEMENT HERE
Donate to the James P. Taylor Foundation for Open Science
Learn More
Want to learn the best practices for genome assembly using Galaxy? Visit the Galaxy VGP page at https://galaxyproject.org/projects/vgp/
Galaxy version 24.1.4.dev0, commit c4dc8e1354e9aa1926bdba146b2ae2a6aa313a6c
History
search datasets
Unnamed history
This history is empty.
You can load your own data or get data from an external source.

Before the next class, you should

Lecture 02B:
DNA sequencing - Methodology

Lecture 03A:
Genome assembly - Foundations

Today

Tuesday

- Finish and submit P01A by Friday, 11:59 pm
- Start working on P01B (will be released Friday, Jan 17)
- Start working on CByte 01 (will be released Friday, Jan 17)