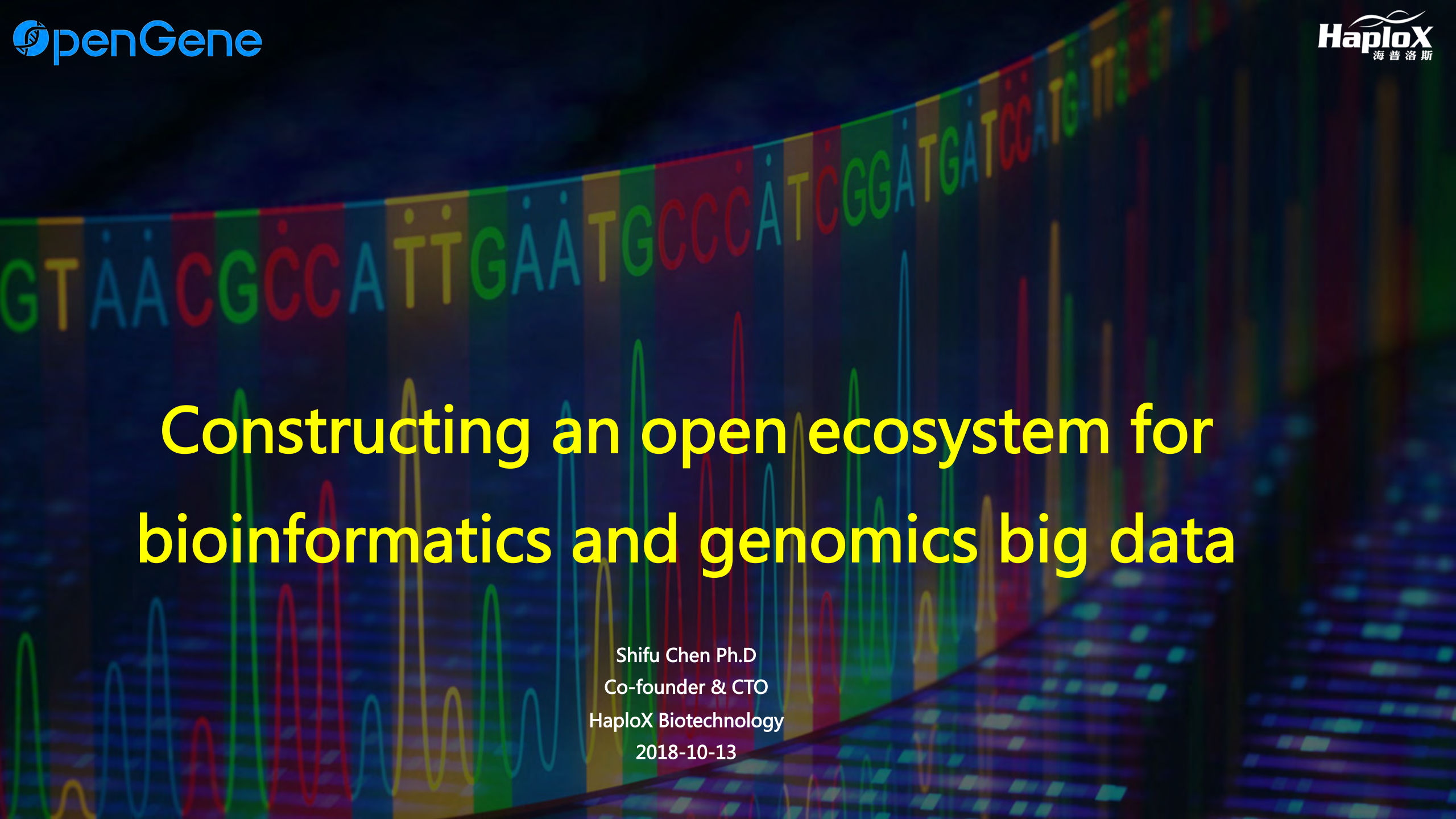
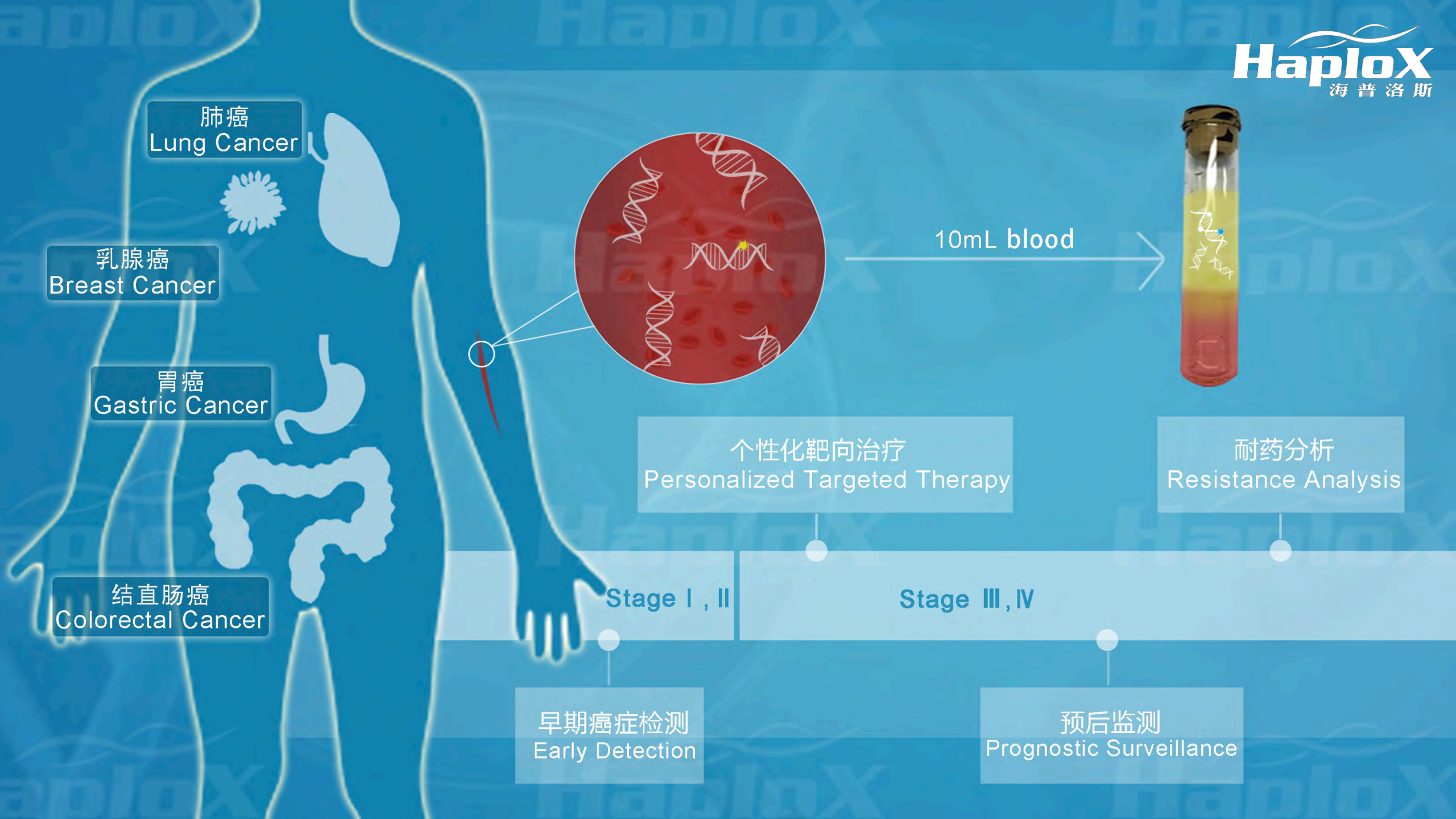

OpenGene
HaploX
海普洛斯
Constructing an open ecosystem for bioinformatics and genomics big data
Shifu Chen Ph.D
Co-founder & CTO
HaploX Biotechnology
2018-10-13


肺癌
Lung Cancer
乳腺癌
Breast Cancer
胃癌
Gastric Cancer
结直肠癌
Colorectal Cancer
10mL blood
个性化靶向治疗
Personalized Targeted Therapy
耐药分析
Resistance Analysis
Stage Ⅰ, Ⅱ
Stage Ⅲ, Ⅳ
早期癌症检测
Early Detection
预后监测
Prognostic Surveillance

HaploX Genomics Center: a large sequencing center

QX200 ddPCR

2* NextSeq 500/550

2* HiSeq X Ten

10* NovaSeq 6000

Able to sequence 60,000 whole genomes per year

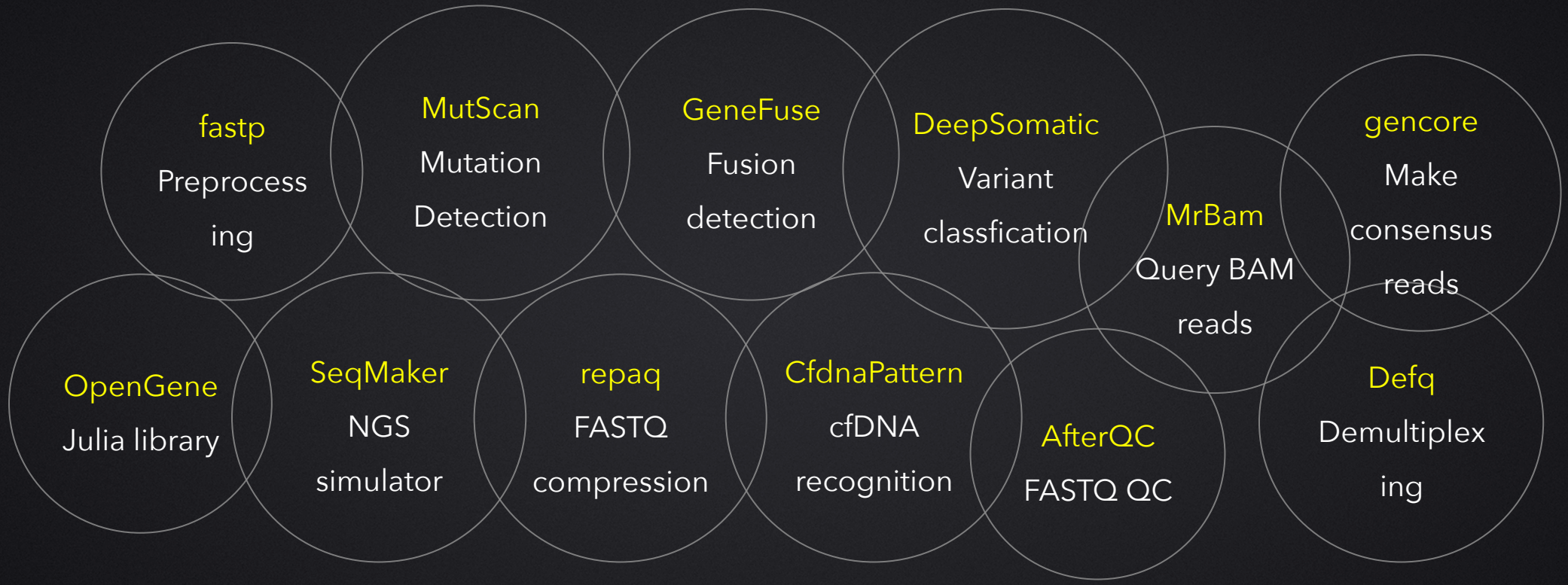
To address some common bioinformatics problems, I started

OpenGene

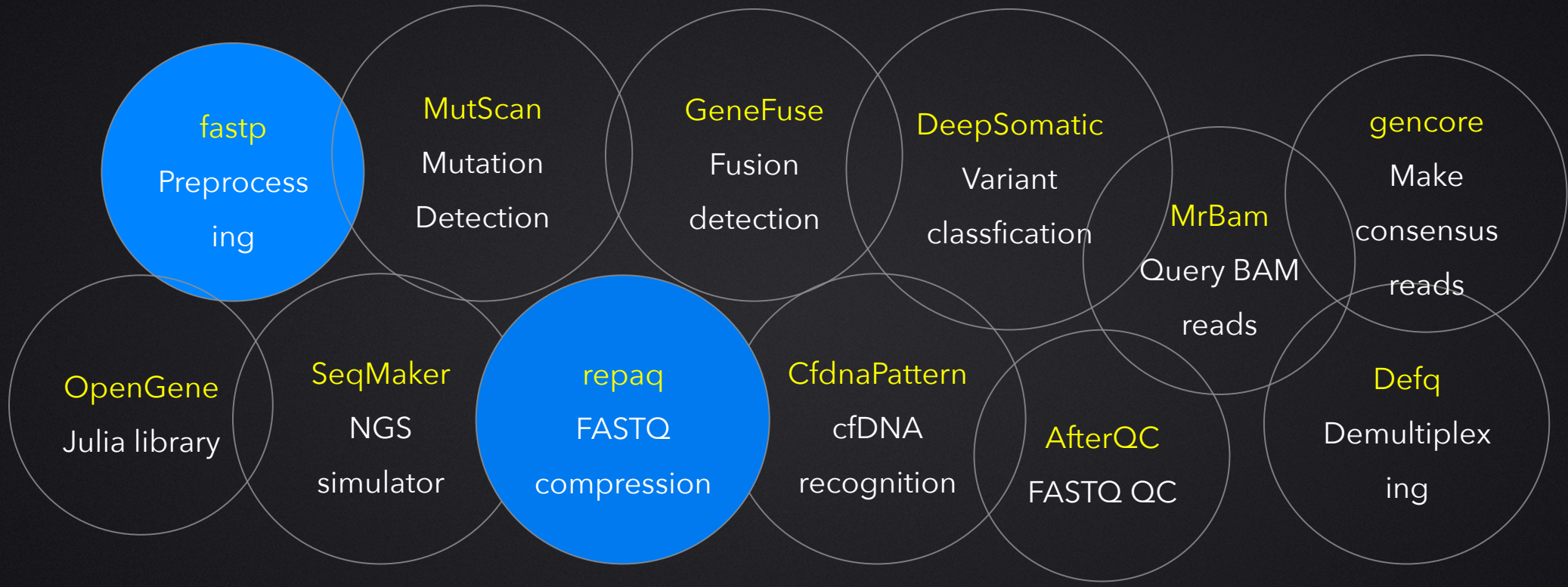
Since Apr. 2016

https://github.com/OpenGene

Developed 10+ open source software



Developed 10+ open source software



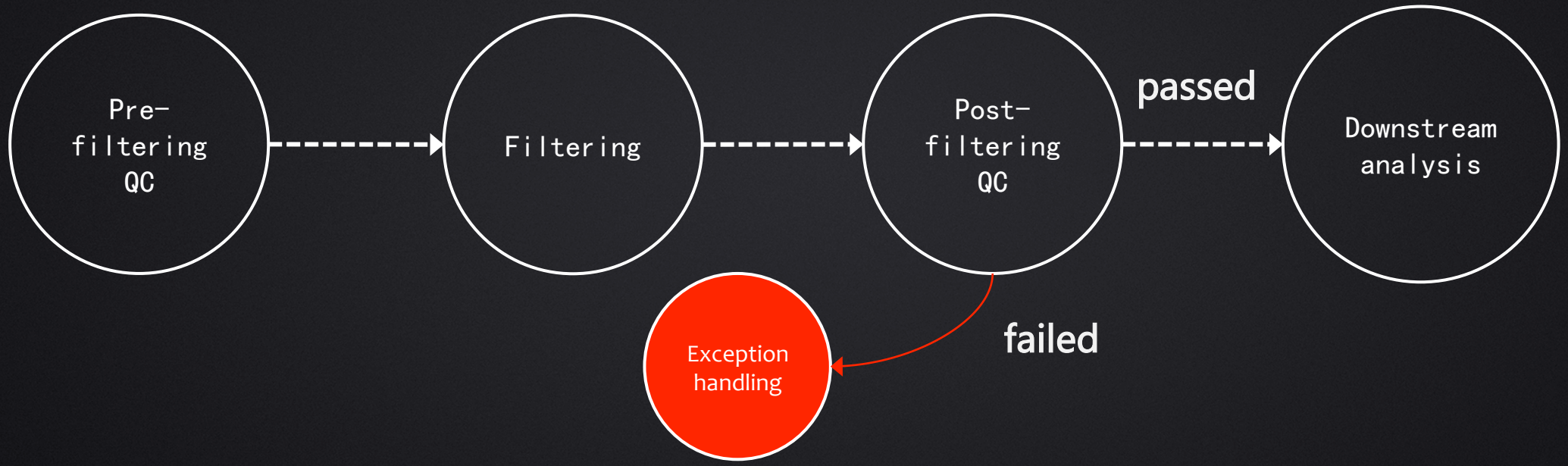
Why do we need another FASTQ preprocessor?

How did we do NGS data preprocessing?

1, QC + filtering can be combined to reduce extra computation and I/O

Why do we need another FASTQ preprocessor?

We hope that the QC process can be automatic, so figures are not enough



2, the preprocessor should provide QC report readable for both human and computer

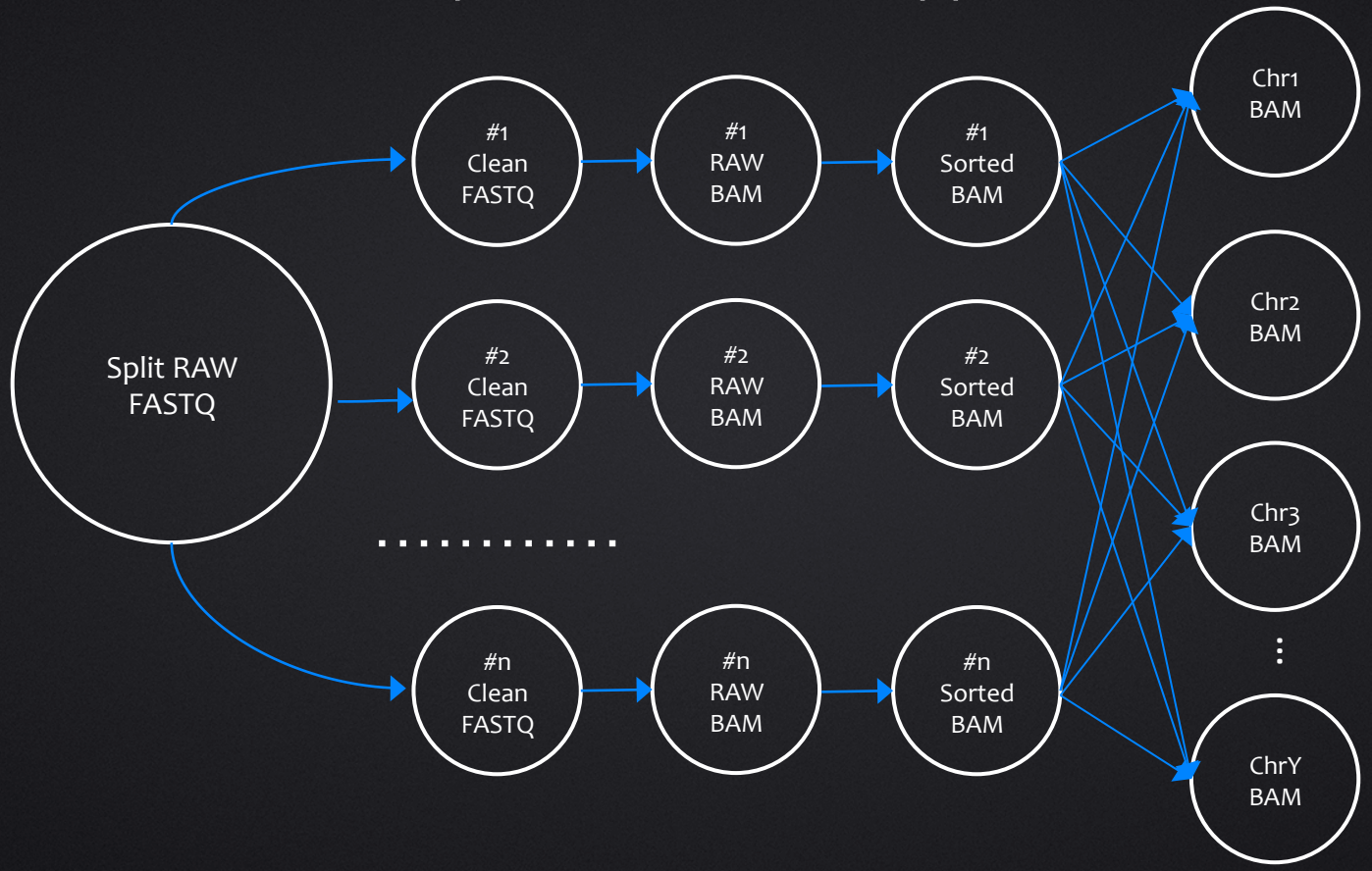
Why do we need another FASTQ preprocessor?

Some functions we were expecting...

3, some functions must be integrated into the preprocessing step

Why do we need another FASTQ preprocessor?

We want to parallelize and stream our pipeline



4, the preprocessor should support STDIN, STDOUT and splitting the data

Why do we need another FASTQ preprocessor?

Wait for 10 hours to QC and filter WGS data?

Speed

5, the preprocessor should be ultra-fast

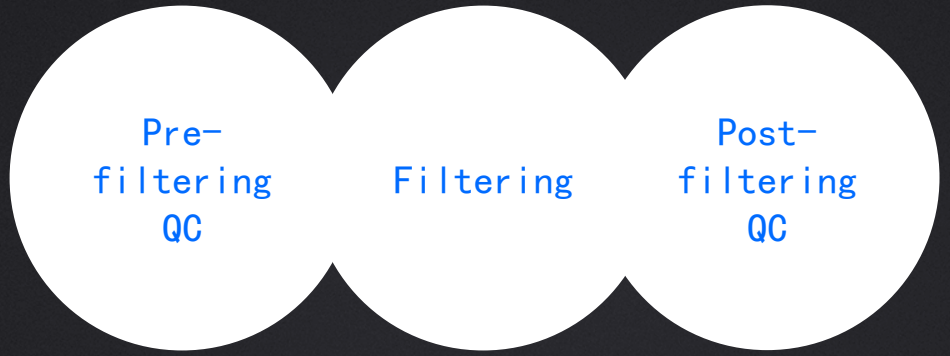
fastp: an ultra-fast all-in-one FASTQ preprocessor

https://github.com/OpenGene/fastp

★ 361

fastp: an ultra-fast all-in-one FASTQ preprocessor

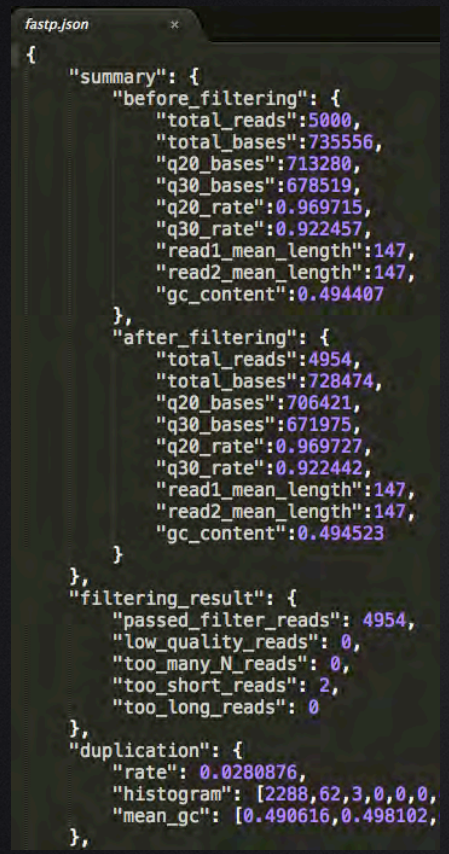
QC + filtering in a single I/O pass

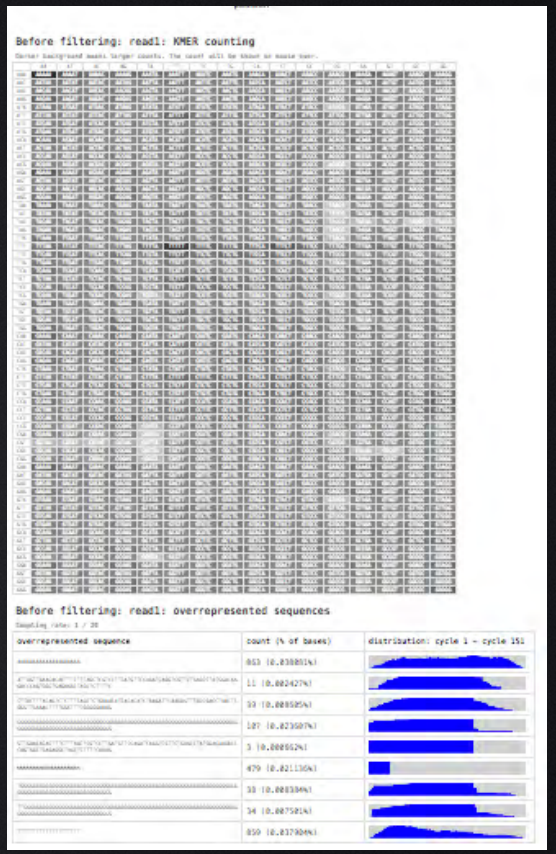


1, QC + filtering can be combined to reduce extra computation and I/O

fastp: performs quality control, adapter trimming, quality filtering, per-read quality pruning, and many other operations with a single scan of the FASTQ data

fastp: an ultra-fast all-in-one FASTQ preprocessor

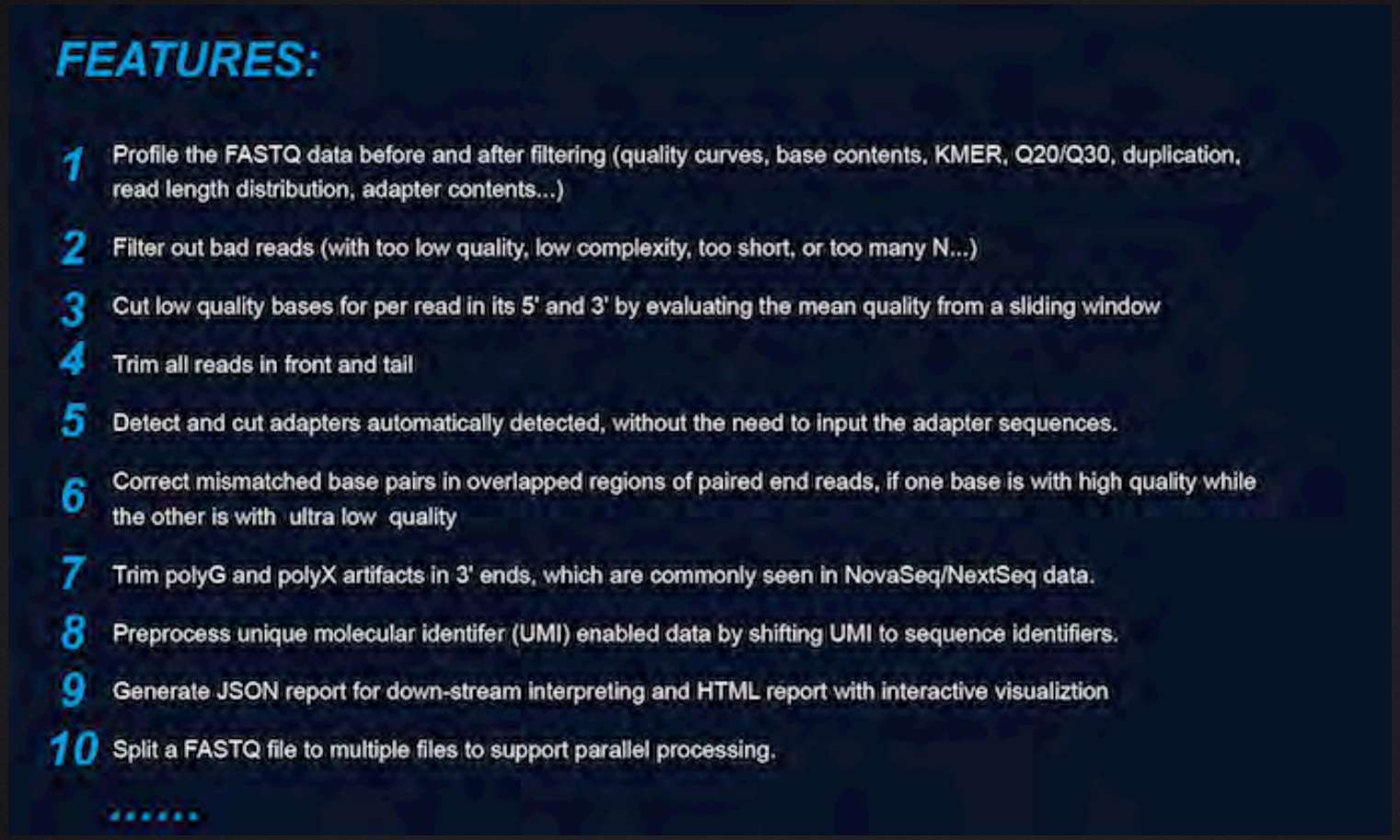




2, the preprocessor should provide QC report readable for both human and computer

fastp: generates human readable JSON report + interactive HTML report

fastp: an ultra-fast all-in-one FASTQ preprocessor



3, some functions must be integrated into the preprocessing step

fastp: auto filtering, adapter trimming, quality pruning, base correction, polyG/polyX trimming, UMI, insert-size profiling, duplication profiling and many more...

fastp: an ultra-fast all-in-one FASTQ preprocessor

Splitting:

1. by limiting total split file number
2. by limiting split file size

Streaming:

1. read from STDIN or write to STDOUT
2. automatic interleaved for PE data

4, the preprocessor should support STDIN, STDOUT and splitting the data

fastp: supports splitting when filtering, streaming (STDIN + STDOUT)

fastp: an ultra-fast all-in-one FASTQ preprocessor

Tool	Input	Time (min)	Throughput (reads/s)
fastp	SE	6	129397.9
	PE	13.3	116750.0
FASTQC	SE	13	59722.1
	PE	25.8	60185.1
Cutadapt	SE	18	43132.6
	PE	24.6	63120.9
SOAPnuke	SE	30.7	25289.5
	PE	32.5	47777.7
AfterQC	SE	25.2	30809.0
	PE	57.2	27146.4
Trimmomatic	SE	31.9	24338.2
	PE	60.9	25497.1

Table 1. Speed comparison of fastp and other software. Results for both paired-end (PE) and single-end (SE) input were compared.

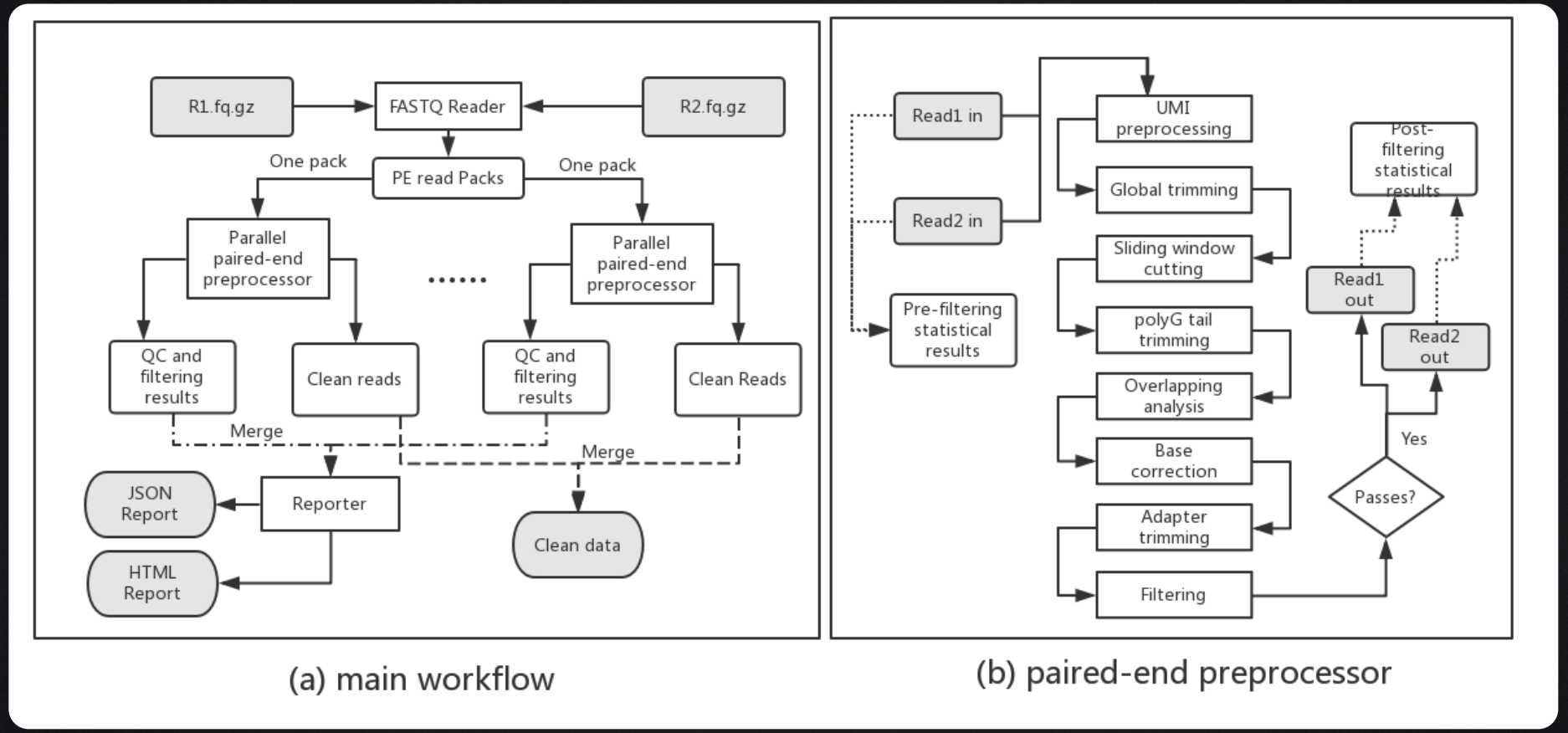
Remember that fastp does much more than other tools in this evaluation

20X WGS in one hour!

5, the preprocessor should be ultra-fast

fastp: developed in C++ with solid multi-threading support, 2~5X faster than other tools

How we make it?



The workflow of fastp

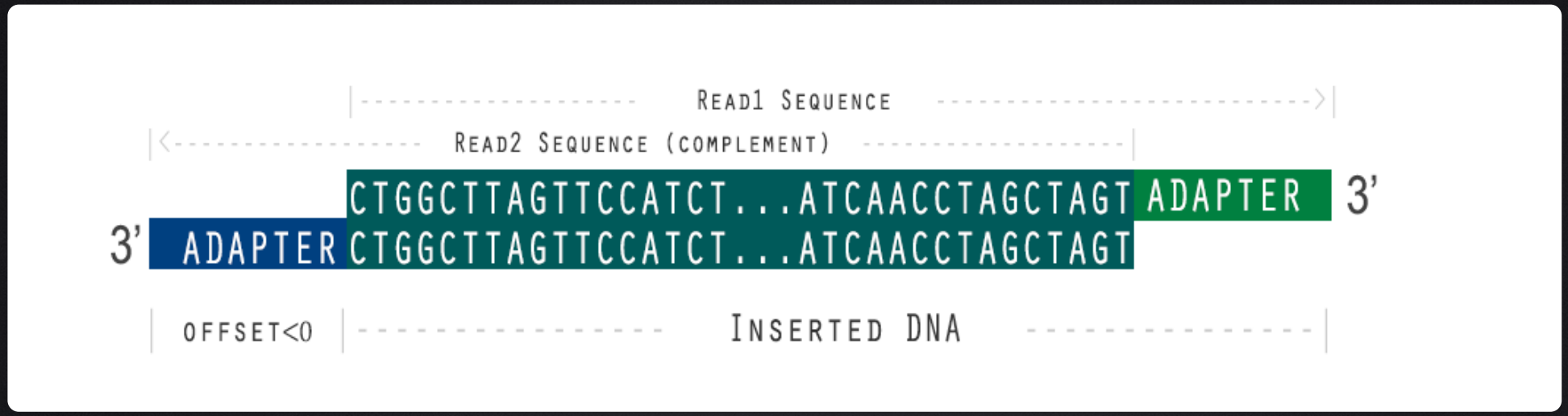
(a) Main workflow of paired-end data processing, and (b) paired-end preprocessor of one read pair. In the main workflow, a pair of FASTQ files is loaded and packed, after which each read pair is processed individually in the paired-end preprocessor, described in (b).

PE overlap analysis for error correction

Error correction for PE data by overlap analysis

For each pair of reads, fastp computes their overlapping region and correct the unbalanced mismatches (1 extreme high quality base + 1 extreme low quality base)

PE overlap analysis for adapter trimming



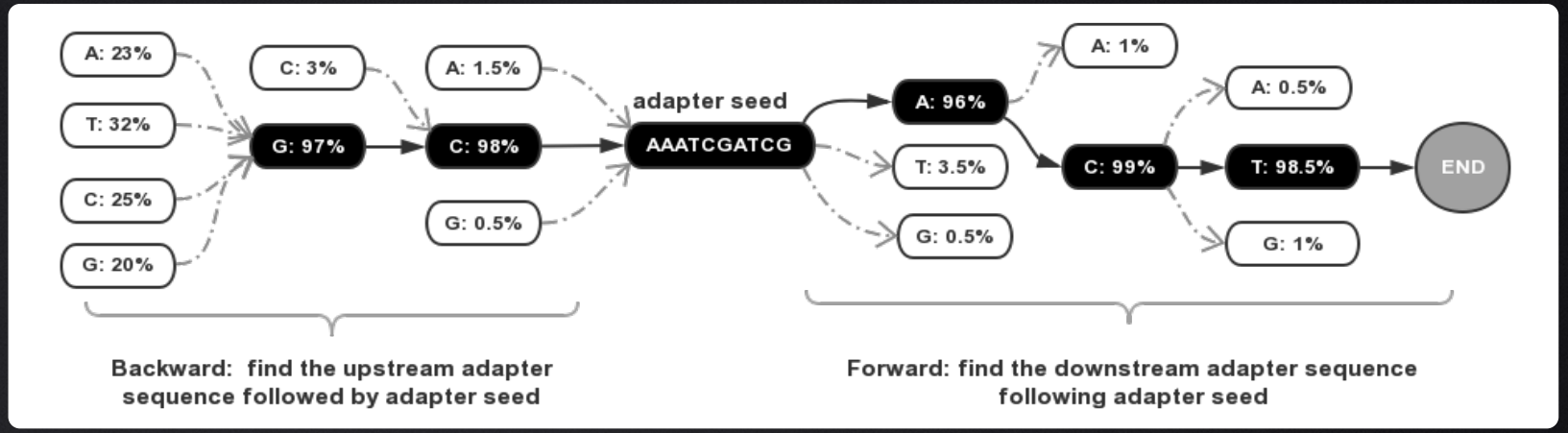
Automatic adapter trimming for PE data by overlap analysis

For each pair of reads, fastp computes their overlapping region and then computes the insert length. If insert length is shorter than the sequencing length, then adapter sequence will be present in the 3 end.

Automatic adapter sequence detection for SE data

Algorithm 1: adapter sequence detection

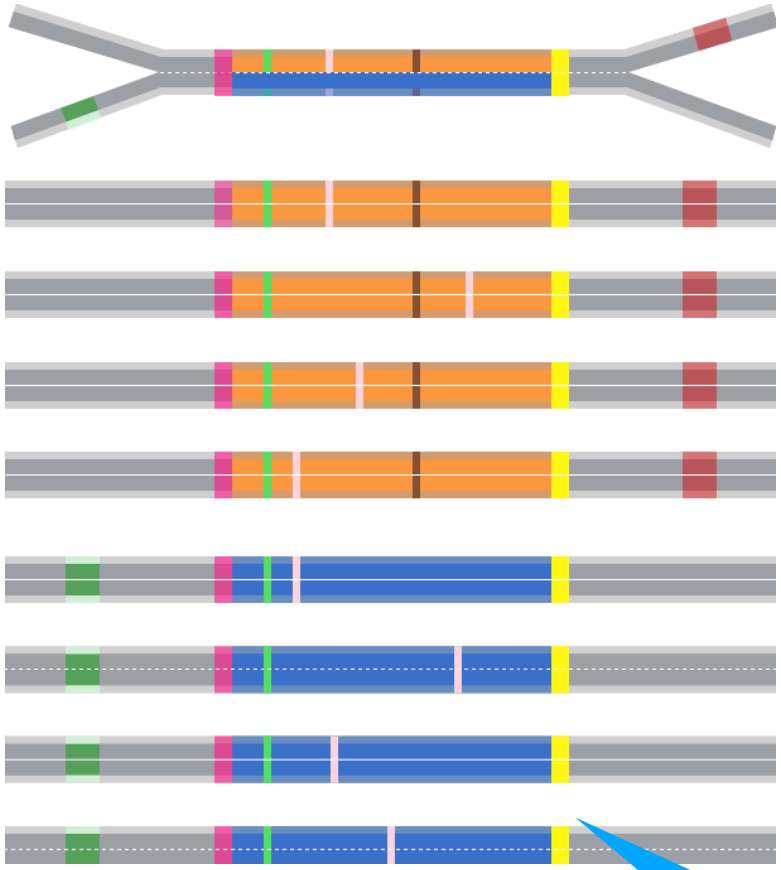
```
for seed in sorted_adapter_seeds:
    seqs_after_seed = get_seqs_after(seed)
    forward_tree = build_nucleotide_tree(seqs_after_seed)
    found = True
    node = forward_tree.root
    after_seed = ""
    while node.is_not_leaf():
        if node.has_dominant_child():
            node = node.dominant_child()
            after_seed = after_seed + node.base
        else:
            found = False
            break
    if found == False:
        continue
    else:
        seqs_before_seed = get_seqs_before(seed)
        backward_tree = build_nucleotide_tree(seqs_before_seed)
        node = backward _tree.root
        before_seed = ""
        while node.is_not_leaf():
            if node.has_dominant_child():
                node = node.dominant_child()
                before_seed = node.base + before_seed
            else:
                break
    adapter = before_seed + seed + after_seed
    break
```



Detect adapter sequence by constructing nucleotide tree

This algorithm tries to extend an adapter seed in the forward direction to check its validity since a valid adapter can always be extended to the read tails. And if this adapter seed is valid, a backward extension is applied to obtain the complete adapter sequence.

UMI preprocessing



UMI sequencing

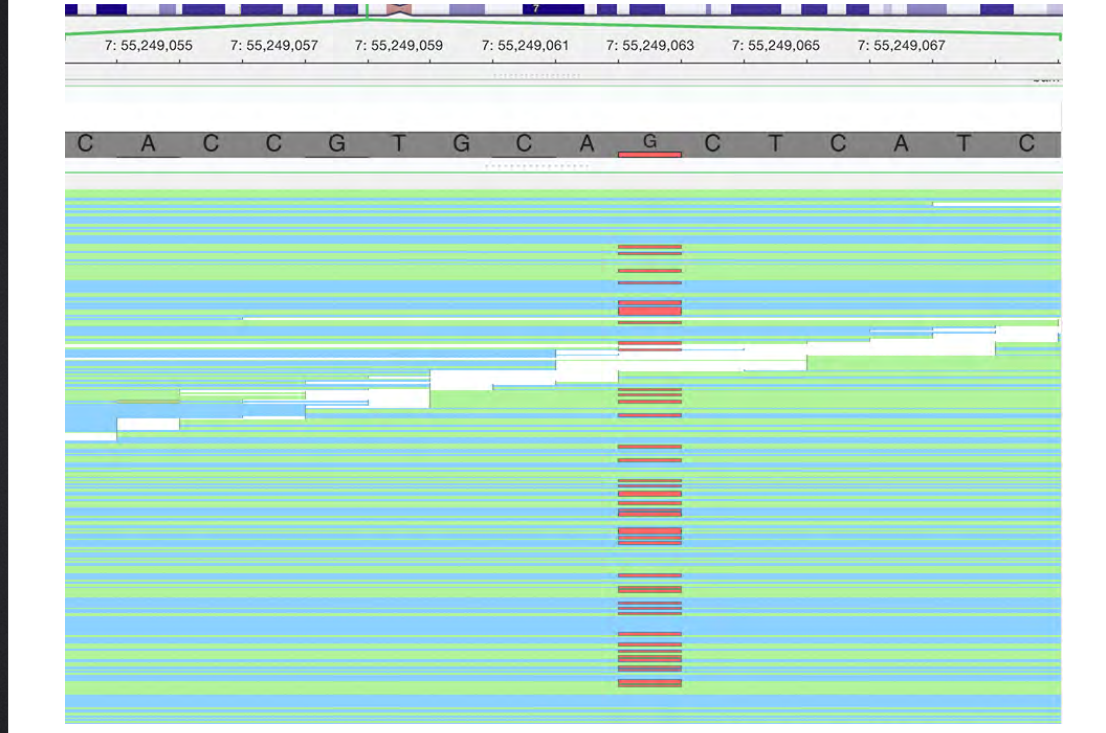
Tools	Time (min)	Throughput (read/s)
fastp	4.6	104302.9
umis	12.43	38599.6
UMI-tools	28.38	16906.4

Handle UMI easily by fastp

fastp upports UMI in either a sample index or inserted DNA (or both). Compared to UMI-tools or umis, fastp runs approximately 3 times faster even when performing other tasks simultaneously (i.e., QC and filtering). Performance evaluation results are discussed in the next section.

fastp + gencore for UMI-based deduplication and error elimination

(a) the noisy data before processing

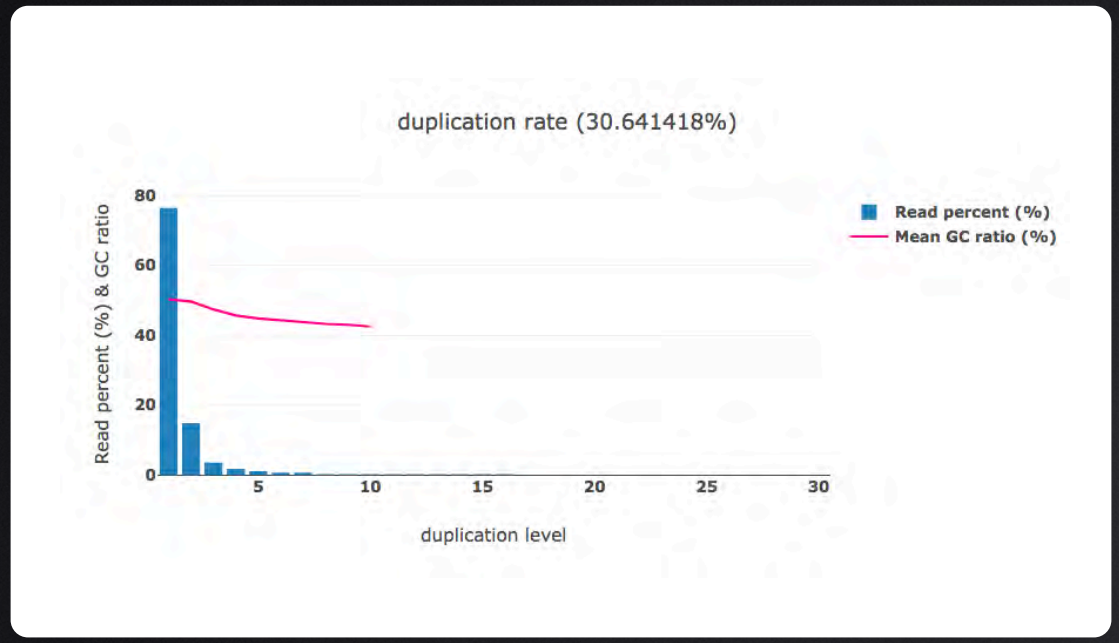


(b) the clean data after processing

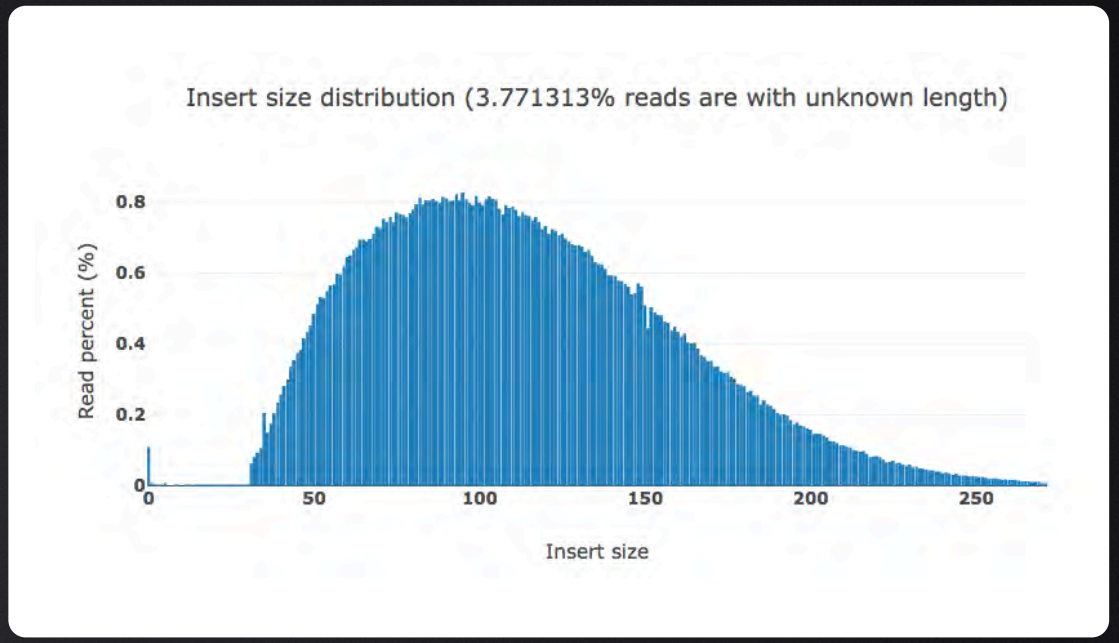
The effect of using fastp and gencore to reduce sequencing errors

gencore is a tool to generate consensus reads, which is also developed by me: https://github.com/OpenGene/gencore

Evaluate duplication level and insert-size distribution



(a) the duplication level and distribution



(b) the insert-size distribution

fastp automatically evaluates duplication rate and insert-size with almost zero overhead

fastp supports duplication level evaluation for both single-end and paired-end data. Different from FASTQC that uses a hash table to store the duplication keys, fastp stores them by a duplication array D and a counting array C to provide much faster access.

Detect the overrepresented sequences

overrepresented sequence	count (% of bases)	distribution: cycle 1 ~ cycle 151
AAAAAAAAAAAAAAAAAAAA	863 (0.038081%)	
ATTAGTTGAACACACTTTCTTTAGCTCGTCCTTGATGTTCCAGATCAGGTCGTTCTGAGCCTATGGACAAG ACCCAGTGGCTGAGAGGCTAGCTCTTTTC	11 (0.002427%)	
CTTACTTTACAGTCTCTTTCAGCTCTGAAACATCACACATCTAAGATTCAAGAGTTTGCCGACCTAACTCG GGTTGAAACTTTTGGCTTTCGGGGGAAAG	39 (0.008605%)	
GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG GGGGGGGGGGGGGGGGGGGGGGGGGGG	107 (0.023607%)	
GTTGAACACACTTTCTTTAGCTCGTCCTTGATGTTCCAGATCAGGTCGTTCTGAGCCTATGGACAAGACCC AGTGGCTGAGAGGCTAGCTCTTTTCAAAG	3 (0.000662%)	
NNNNNNNNNNNNNNNNNNNN	479 (0.021136%)	
TGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG GGGGGGGGGGGGGGGGGGGGGGGGGGG	38 (0.008384%)	
TTGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG GGGGGGGGGGGGGGGGGGGGGGGGGGG	34 (0.007501%)	
TTTTTTTTTTTTTTTTTTTT	859 (0.037904%)	

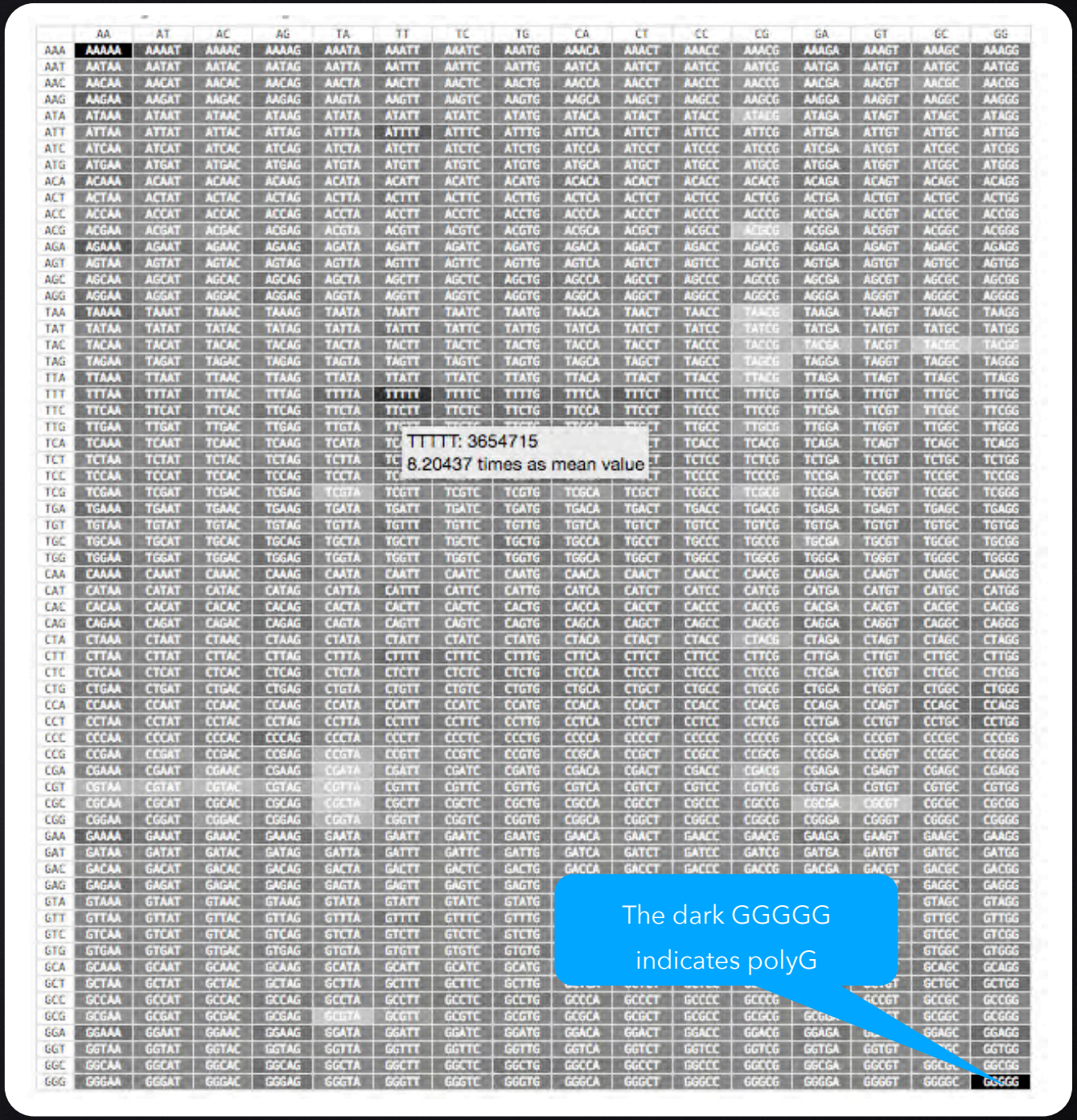
fastp detects not only the overrepresented sequences, but also their distributions

Besides the occurrence frequency, fastp also records the positions of overrepresented sequences. This information is quite useful for diagnos-ing sequence quality issues. Some sequences tend to appear in the read head whereas others appear more often in the read tail.

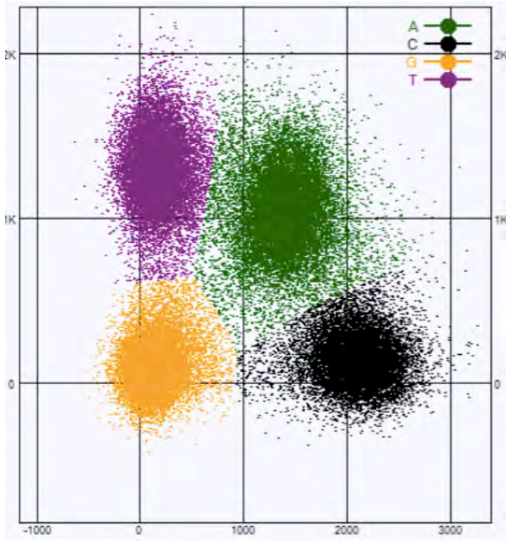
Interactive full k-mer table

fastp computes the exactly occurrence table of 5-mer, and present it in a interactive HTML table.

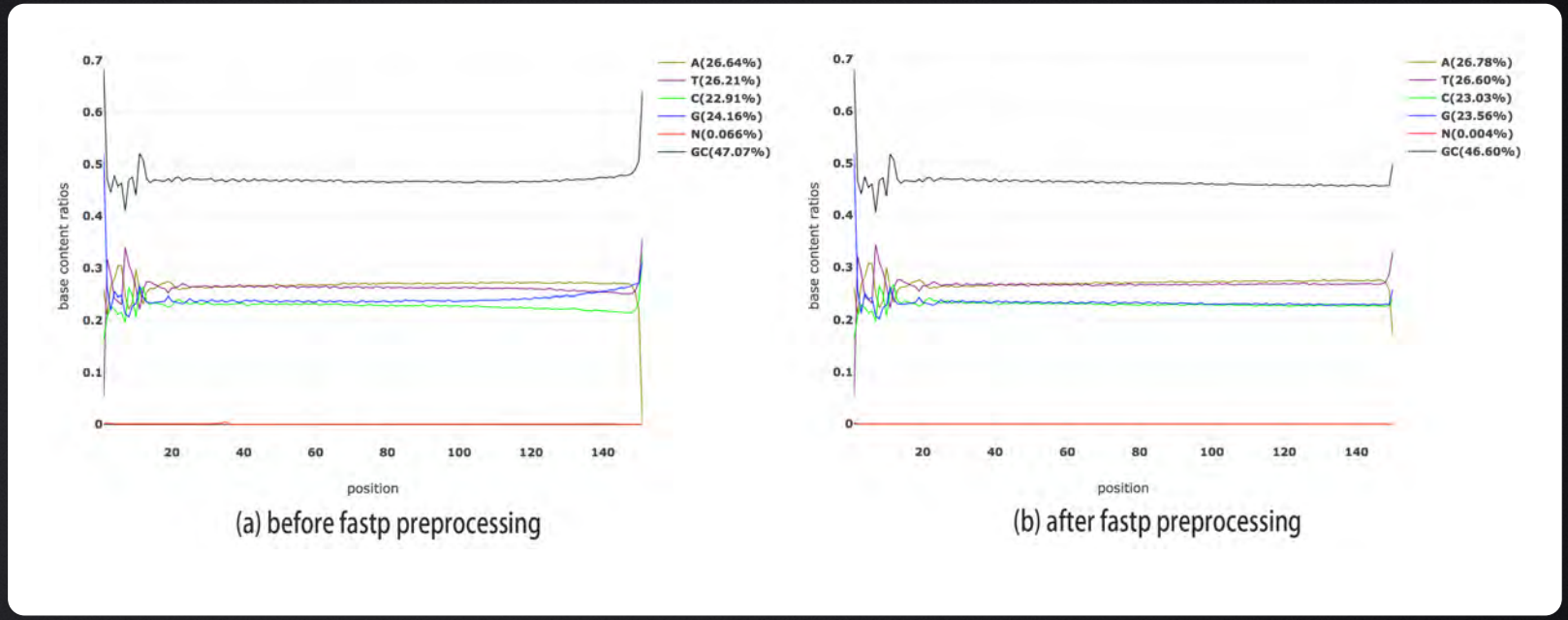
Darker background means larger counts. The count will be shown on mouse over. By moving mouse over, it will show how frequent the 5-len sequence appears comparing to the mean value. From this table, you can find the abnormal sequencing artifacts, such like polyG problem.



How polyG happens, and how fastp addresses this issue



2-color system causes polyG



fastp addresses the polyG issue

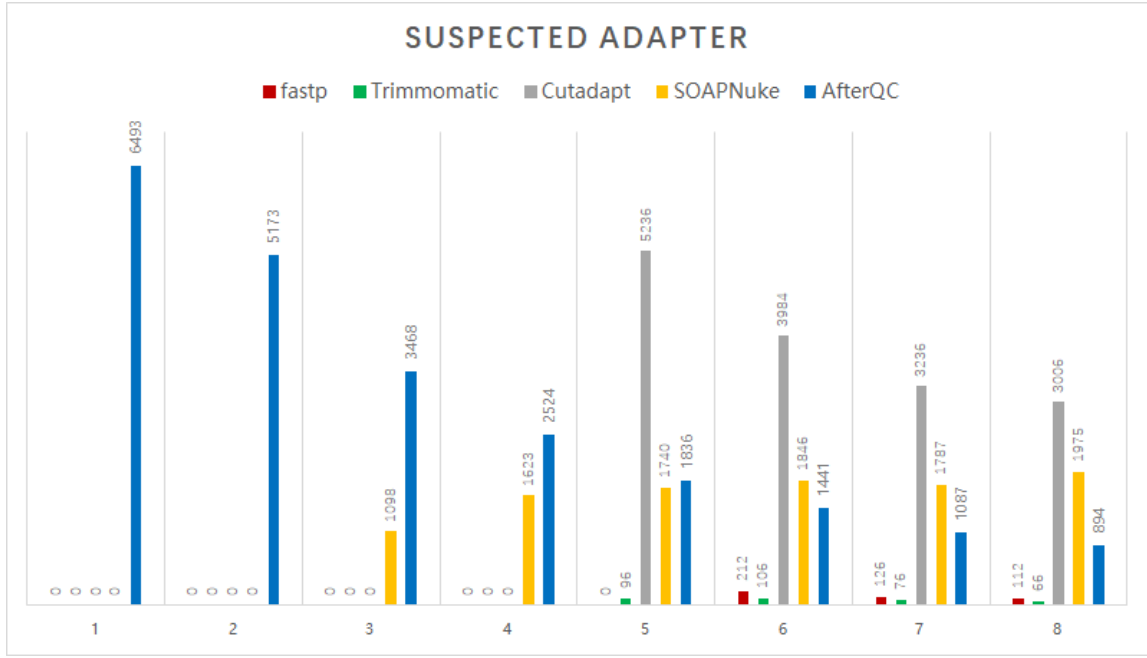
fastp addreses the polyG issue by trimming polyG tails

fastp also implements polyX tail trimming, where X means any base of A/T/C/G. This function can be used to trim the low-complexity con-secutive bases in 3’ end. PolyX tail trimming and polyG tail trimming can be enabled together.

Evaluation of filtering performance for downstream analysis

NS_PE150	Total map base (M)	Mis-matched base (M)	Total map read (M)	Clip read (M)	Single-read map
Raw data	3390	19.8	22.4	6.16	46025
fastp	3102	10.6	21.2	0.30	271
AfterQC	3053	12.3	21.2	0.64	34099
SOAPnuke	2981	18.3	19.7	3.46	36888
Trimmo-matic	3111	12.8	22.1	0.75	229111
Cutadapt	3287	19.8	22.4	1.05	46195

(a) Mismatches, clips, and single-read maps of the data filtered with different tools.



(b) Result of adapter trimming performance evaluation.

fastp produces cleaner data for downstream analysis

The standard dataset NA12878 (SRR952827) was selected for benchmarking fastp, Trimmomatic, Cutadapt and SOAPNuke, for their performance on reducing false positive variants.

10T bases generated per day, how to

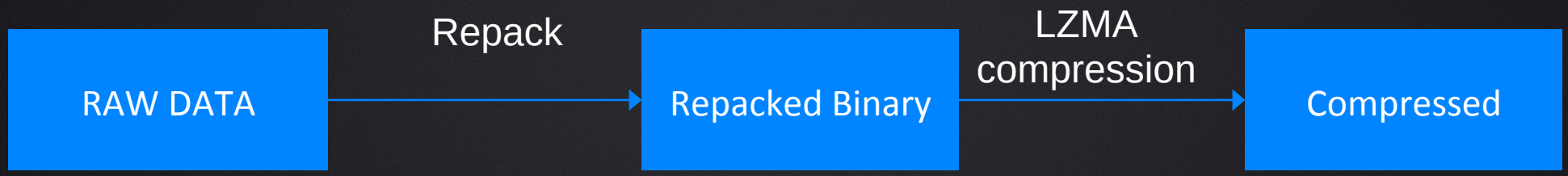
repaq

A pilot project for lossless compression of FASTQ data

https://github.com/OpenGene/repaq

★ 40

2-stage compression



Strategies to repack the original FASTQ

1. Encode and decode by chunks
2. Process the metainfo, sequence and quality separately
3. Extract the common info and ordered info from metainfo
4. Find the overlap of a pair of reads and encode it only once
5. Find the connection of N bases with corresponding quality bin
6. Find the major quality bin, and only encode the others
7. Don’t use any entropy encoding to preserve the compressibility for LZMA

FASTQ file of 100G bases from NovaSeq

Uncompressed

245GB 100%

Gzip

50GB 20.4%

Bzip2

40GB 16.3%

repaq ?

FASTQ file of 100G bases from NovaSeq

Uncompressed
245GB 100%

Gzip
50GB 20.4%

Bzip2
40GB 16.3%

Repaq
10 GB 4.1%


OpenGene
Welcome to Shenzhen
Thank you for your attention!