

# Introduction to NGS Analysis

---

**Anna Battenhouse**

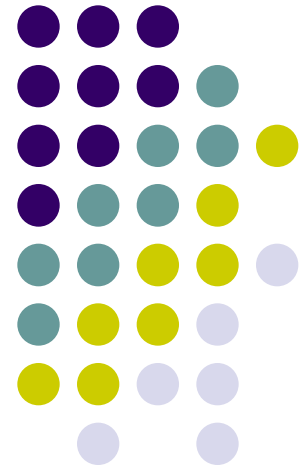
Associate Research Scientist

Vishwanath Iyer Lab

The University of Texas at Austin

[abattenhouse@utexas.edu](mailto:abattenhouse@utexas.edu)

*May, 2016*



# Outline

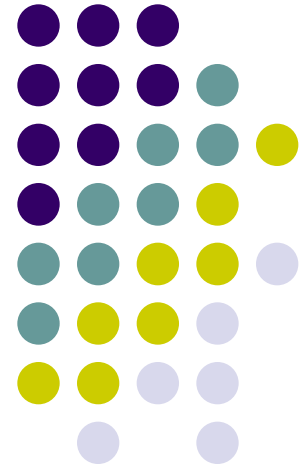


1. NGS overview & terminology
2. The FASTQ format
3. Raw data QC and preparation
4. Alignment to a reference

# Part 1: NGS Overview and Terminology

---

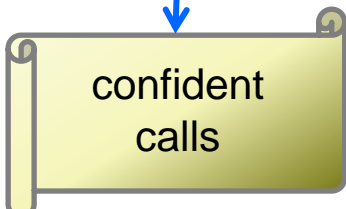
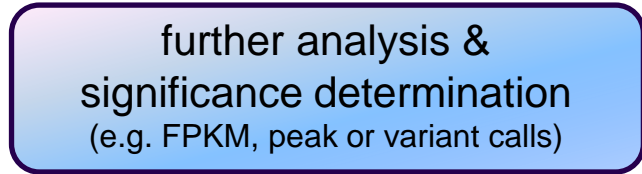
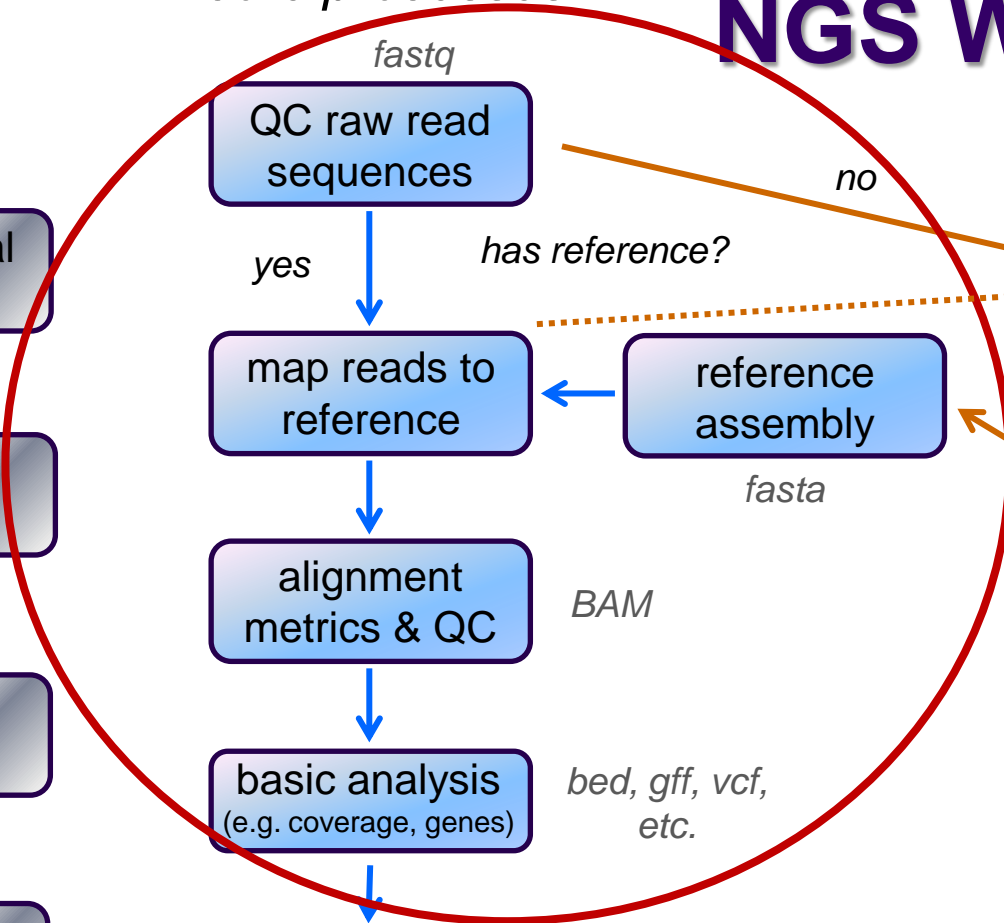
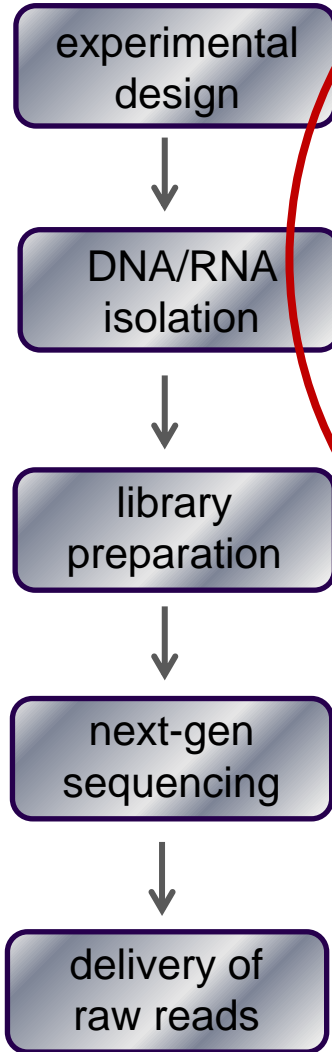
- NGS workflow overview
- Sequencing terminology & considerations



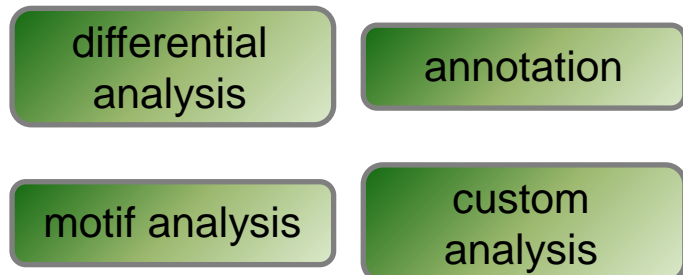
# NGS Workflow

core processes

upstream processes



downstream processes

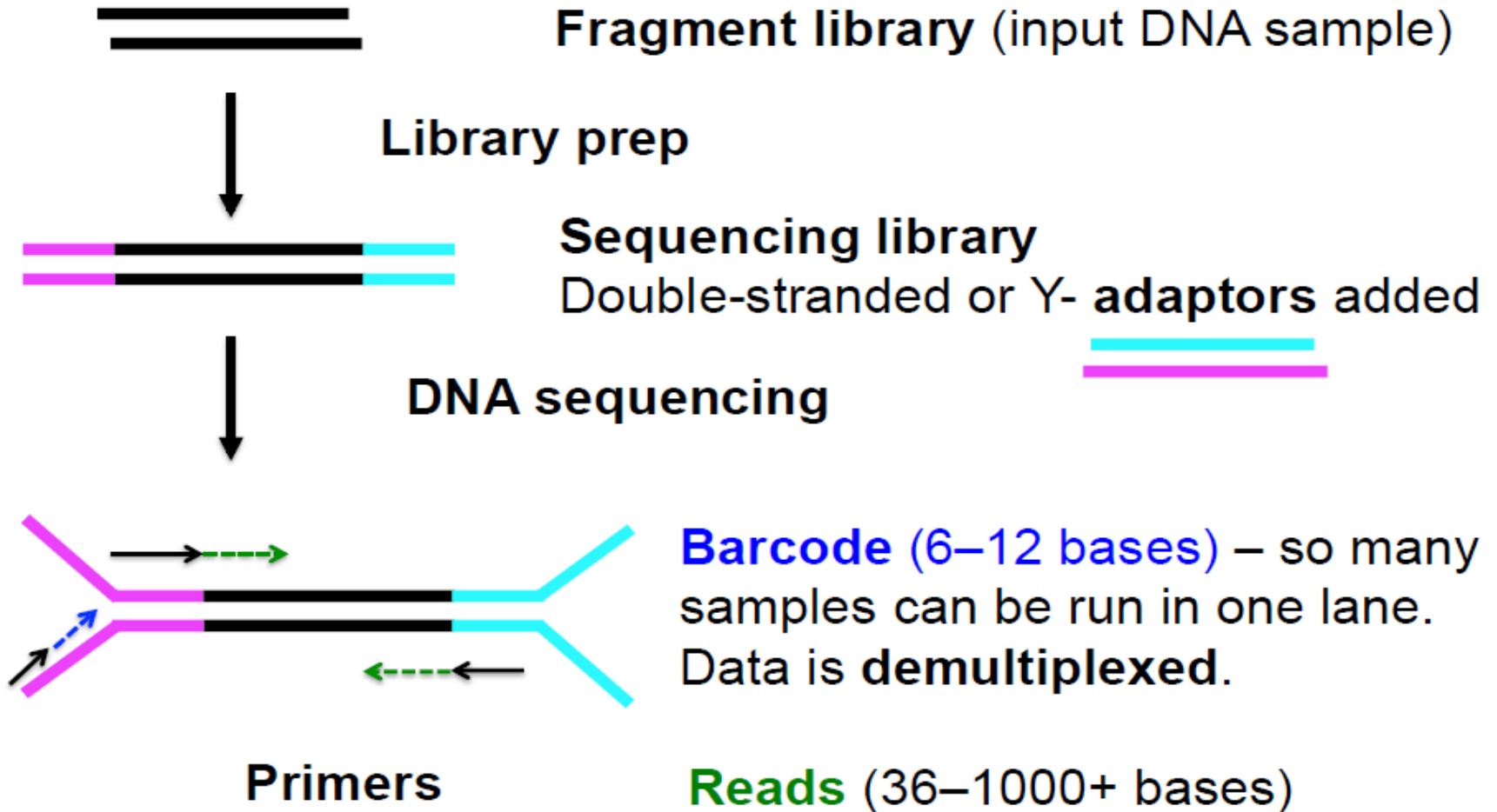


# Sequencing technologies



- Illumina (Solexa) now dominant
  - [Official Illumina video](#)
  - [Another Illumina video](#)
  - [Broad Center GA Boot Camp](#)
- Many others
  - Comparison of NGS technologies (Liu et al., 2012)  
<http://www.hindawi.com/journals/bmri/2012/251364/>

# Read sequence terminology



- Adapter areas include primers, barcode
  - sequencing facility will have more information

# Types of Illumina sequencing



## single-end



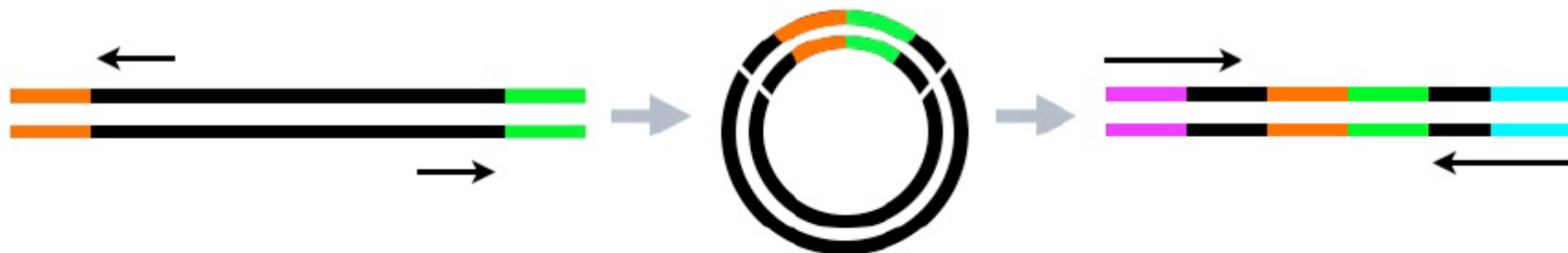
independent reads

## paired-end



two inwardly oriented reads separated by ~200 nt

## mate-paired



two outwardly oriented reads separated by ~3000 nt

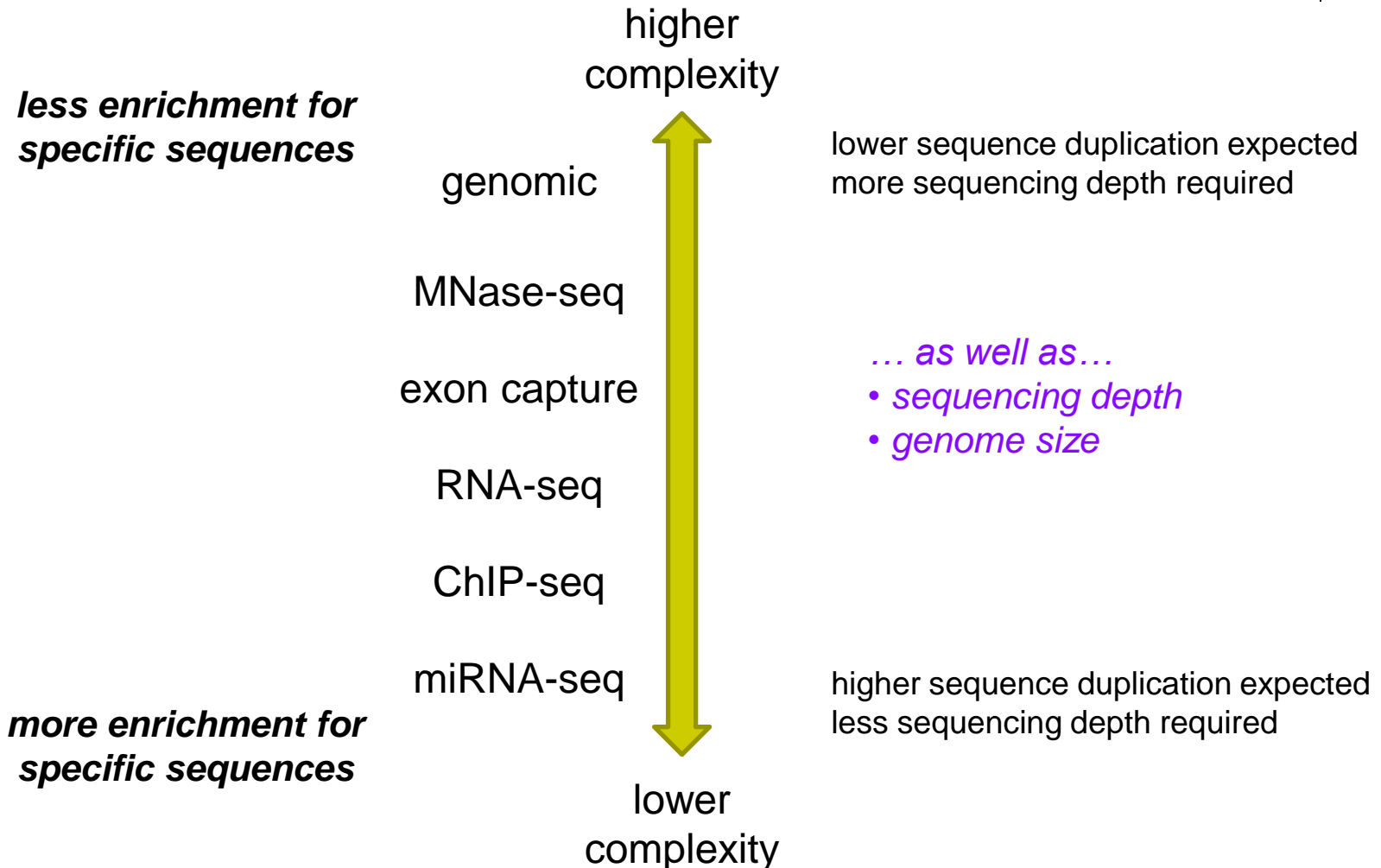


# Sequencing depth

- No single answer to how much depth is adequate
- Depends on:
  - genome size
    - prokaryotes – a few Kilobases
    - lower eukaryotes – some number of Megabases
    - higher eukaryotes – Gigabases
  - library fragment enrichment
    - e.g. ChIP-seq or RIP-seq
  - theoretical library complexity
    - genomic resequencing **vs** 4c
  - desired sensitivity
    - e.g. looking for rare mutations



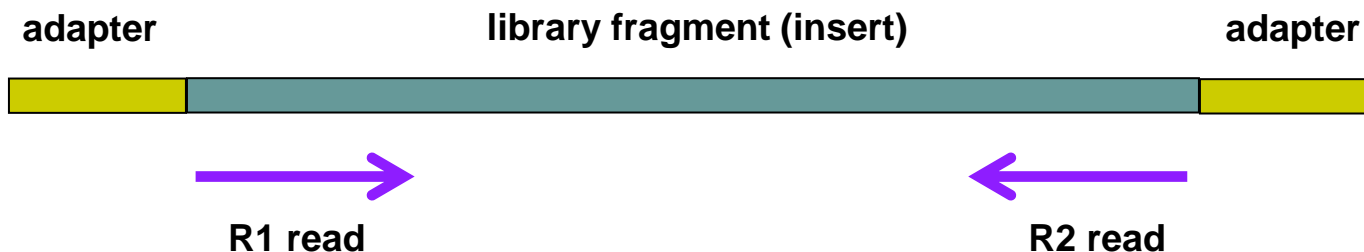
# Library complexity is primarily a function of experiment type



# Reads and Fragments



- With paired-end sequencing, keep in mind the distinction between
  - the library **fragment** from your library that was sequenced
    - also called **inserts**
  - the **sequence reads** (R1 & R2) you receive
    - also called **tags**
- There is considerable confusion of terminology in this area!
  - Be sure you request depth in **read pairs** for paired-end sequencing



# Single end vs Paired end

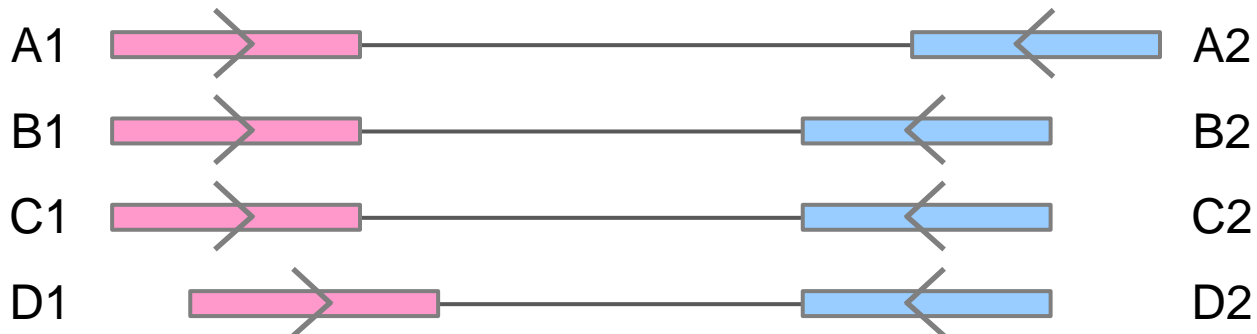


- **paired end** (PE) reads can be mapped more reliably
  - especially against lower complexity genomic regions
    - when one member of a read pair does not align well, it can still be “rescued” if its mate maps well
  - they also provide more bases around a locus
    - e.g. for analysis of polymorphisms
  - actual fragment sizes can be determined
    - from the alignment records for each dual-mapping “proper pair”
  - they also help distinguish the true complexity of a library
    - by clarifying which *fragments* are duplicates
- **but** PE reads are more expensive – and larger
  - more storage space and processing time required

# Read vs fragment duplication

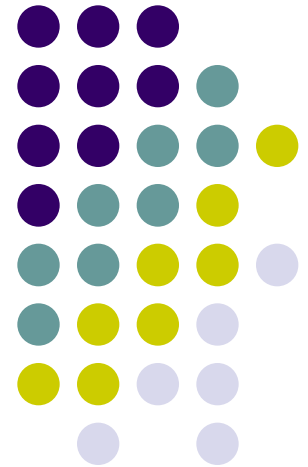


- Consider the 4 fragments below
  - 4 R1 reads (pink), 4 R2 reads (blue)
- Duplication when only 1 end considered
  - A1, B1, C1 have identical sequences, D1 different
    - 2 unique + 2 duplicates = 50% duplication rate
  - B2, C2, D2 have identical sequences, A2 different
    - 2 unique + 2 duplicates = 50% duplication rate
- Duplication when both ends considered
  - fragments B and C are duplicates (same external sequences)
    - 3 unique + 1 duplicate = 25% duplication rate



# Part 2: The FASTQ format

---



# FASTQ files



- Nearly all sequencing data now delivered as FASTQ files
  - FASTQ = FASTA sequences + Quality scores
  - file names have **.fastq** or **.fq** extensions
  - usually compressed to save space
    - (**gzip**'d, with **.gz** file extension)
  - best practice: leave them that way!
    - 3x to 6x space saving
    - most tools handle **gzip**'d FASTQ
- Paired-end sequencing data comes in 2 FASTQs
  - one each for R1 and R2 reads, same number of rows  
`Sample_MyTubeID_L008_R1.fastq.gz`  
`Sample_MyTubeID_L008_R2.fastq.gz`
  - ***order of reads is identical***
    - aligners rely on this “name ordering” for PE

# FASTQ format



- Text format for storing sequence and quality data
  - [http://en.wikipedia.org/wiki/FASTQ\\_format](http://en.wikipedia.org/wiki/FASTQ_format)
- 4 lines per sequence:
  1. **@read name**
  2. **called base sequence (ACGTN)**  
always 5' to 3'; usually excludes 5' adapter/barcode
  3. **+optional read name**
  4. **base quality scores encoded as text characters**
- FASTQ representation of a single, 50 base R1 sequence

```
@HWI-ST1097:97:D0WW0ACXX:8:1101:2007:2085 1:N:0:ACTTGA  
ATTCTCCAAGATTTGGCAAATGATGAGTACAATTATATGCCCAATTTACA  
+  
?@@?DD;?;FF?HHBB+:ABECGHDHDCF4?FGIGACFDHFH;FHEIIB9?
```



# FASTQ read names

- Illumina read names encode information about the source cluster
  - unique identifier (“fragment name”) begins with @, then:
    - sequencing machine name
    - lane number
    - flowcell grid coordinates
  - a space separates the name from extra read information:
    - end number (1 for R1, 2 for R2)
    - two quality fields (N = *not* QC failed)
    - barcode sequence
  - R1, R2 reads *have the same name*

@HWI-ST1097:97:D0WW0ACXX:8:1101:2007:2085 1:N:0:ACTTGA

@HWI-ST1097:97:D0WW0ACXX:8:1101:2007:2085 2:N:0:ACTTGA



# FASTQ quality scores



<http://www.asciitable.com/>

Quality character	!"#\$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJ
ASCII Value	33 43 53 63 73
Base Quality (Q)	0 10 20 30 40

$$\text{Probability of Error} = 10^{-Q/10}$$

- Base qualities expressed as **Phred** scores
  - log scaled, higher = better
  - $20 = 1/10^2 = 1/100$  errors,  $30 = 1/10^3 = 1/1000$  errors
- In older FASTQ files, ASCII offsets may differ
  - modern Sanger format shown above
  - see [http://en.wikipedia.org/wiki/FASTQ\\_format](http://en.wikipedia.org/wiki/FASTQ_format) for others

# Multiple lanes



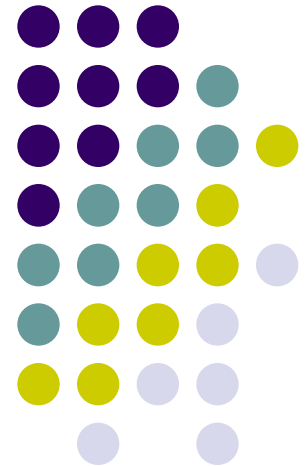
- Sometimes the sequencing facility splits your sample across lanes
  - one submitted sample may be delivered as multiple FASTQ files
    - Lane1: `Sample_MyTubeID_L001_R1.fastq.gz`, `Sample_MyTubeID_L001_R2.fastq.gz`
    - Lane8: `Sample_MyTubeID_L008_R1.fastq.gz`, `Sample_MyTubeID_L008_R2.fastq.gz`
- Your sample may be re-run to “top off” requested read depth
  - be careful with the file names!
    - if run in the same lane, the FASTQ file names will be the same
    - 1<sup>st</sup> run: `Sample_MyTubeID_L003_R1.fastq.gz`
    - 2<sup>nd</sup> run : `Sample_MyTubeID_L003_R1.fastq.gz`
- Best practice
  - keep original data in separate directories by date & project
  - process data from multiple lanes separately for as long as possible
    - e.g., through alignment, then merge the sorted BAMs
    - identical sequences from different lanes can be considered unique (non-duplicates)

# Part 3:

## Data QC & preparation

---

- QC of raw sequences with **FastQC** tool
- Dealing with adapters



# Raw sequence quality control



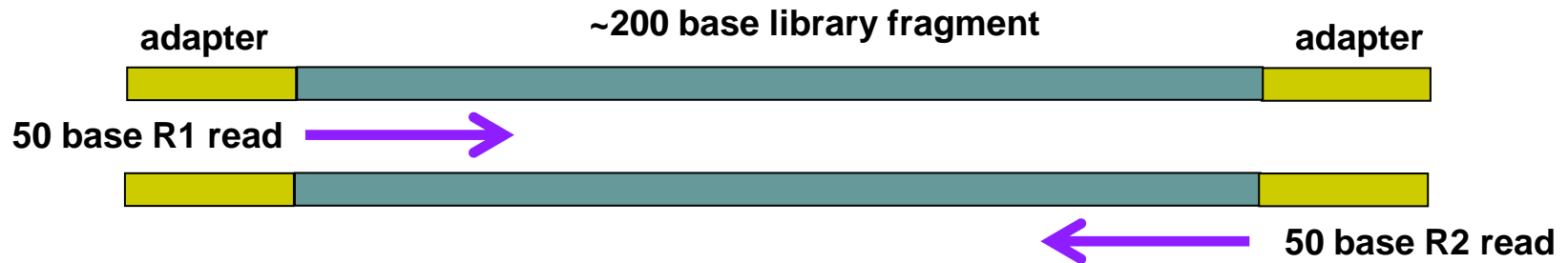
- Critical step! Garbage in = Garbage out
  - general sequence quality
    - base quality distributions
    - sequence duplication rate
  - trim 3' adapter sequences?
    - important for RNAseq
  - trim 3' bases with poor quality?
    - important for *de novo* assembly
  - other contaminants?
    - biological – rRNA in RNAseq
    - technical – samples sequenced w/other barcodes
- Know your data
  - sequencing center pre-processing
    - 5' barcode removal; QC-failed reads filtered
  - PE reads? relative orientations?
  - technology specific issues?
    - e.g. PAR clip should produce C→T transitions



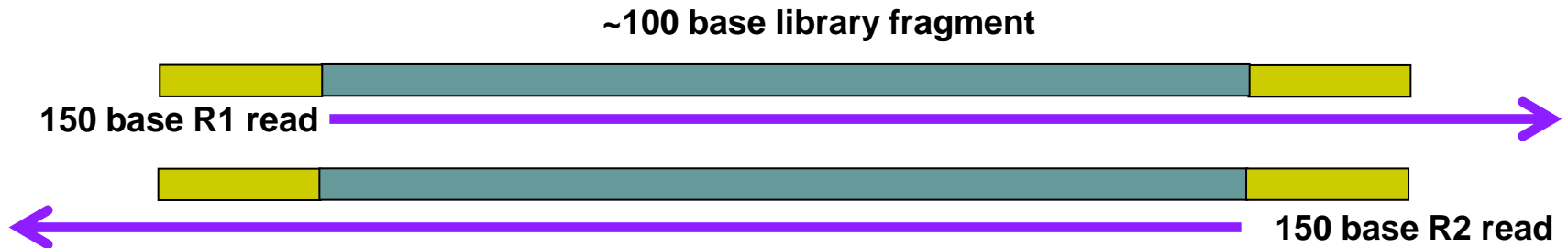
# 3' Adapter contamination



## A. reads short compared to fragment size (no contamination)



## B. Reads long compared to library fragment (3' adapter contamination)



# FastQC



- Quality Assurance tool for FASTQ sequences
- Can run as interactive tool or command line
- Input:
  - FASTQ file(s)
  - run on both R1, R2 files
- Output:
  - directory with html & text reports
    - `fastqc_report.html`
    - `fastqc_data.txt`



# FastQC resources

- FastQC website:  
<http://www.bioinformatics.babraham.ac.uk>
- FastQC report documentation:  
<http://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/>
- Good Illumina dataset:  
[http://www.bioinformatics.babraham.ac.uk/projects/fastqc/good\\_sequence\\_short\\_fastqc/fastqc\\_report.html](http://www.bioinformatics.babraham.ac.uk/projects/fastqc/good_sequence_short_fastqc/fastqc_report.html)
- Bad Illumina dataset:  
[http://www.bioinformatics.babraham.ac.uk/projects/fastqc/bad\\_sequence\\_fastqc/fastqc\\_report.html](http://www.bioinformatics.babraham.ac.uk/projects/fastqc/bad_sequence_fastqc/fastqc_report.html)
- Real Yeast ChIP-seq dataset:  
[http://web.corral.tacc.utexas.edu/BiolTeam/yeast\\_stuff/Sample\\_Yeast\\_L005\\_R1.cat\\_fastqc/fastqc\\_report.html](http://web.corral.tacc.utexas.edu/BiolTeam/yeast_stuff/Sample_Yeast_L005_R1.cat_fastqc/fastqc_report.html)



# Most useful FastQC reports

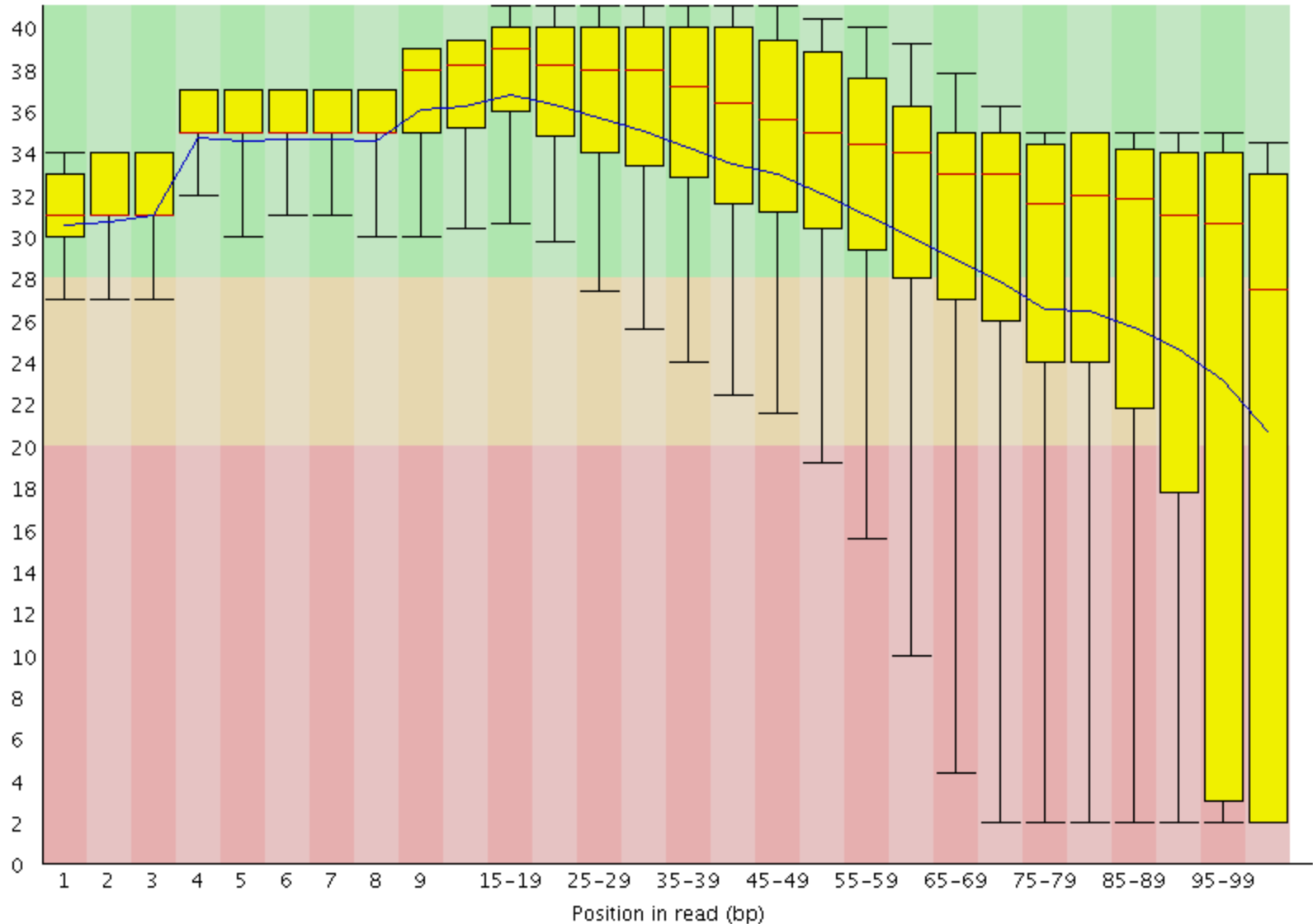
- Should I trim low quality bases?
  - *Per-base sequence quality Report*
    - based on *all* sequences
- Do I need to remove adapter sequences?
  - *Overrepresented sequences Report*
    - based on *1<sup>st</sup> 200,000* sequences
- How complex is my library?
  - *Sequence duplication levels Report*
    - estimate based on *1<sup>st</sup> 200,000* sequences



# FastQC Per-base sequence quality report



Quality scores across all bases (Sanger / Illumina 1.9 encoding)



# FastQC Overrepresented sequences report



- **FastQC** knows Illumina adapter sequences
- Here ~9-10% of sequences contain adapters
  - calls for adapter removal or trimming

Sequence	Count	Percentage	Possible Source
AGATCGGAAGAGCACACGTCTGAACTCCAGTCACCTCAGAATCTCGTATG	60030	5.01369306977828	TruSeq Adapter, Index 1 (97% over 37bp)
GATCGGAAGAGCACACGTCTGAACTCCAGTCACCTCAGAATCTCGTATGC	42955	3.5875926338884896	TruSeq Adapter, Index 1 (97% over 37bp)
CACACGTCTGAACTCCAGTCACCTCAGAATCTCGTATGCCGTCTTCTGCT	3574	0.29849973398946483	RNA PCR Primer, Index 40 (100% over 41bp)
CAGATCGGAAGAGCACACGTCTGAACTCCAGTCACCTCAGAATCTCGTAT	2519	0.2103863542024236	TruSeq Adapter, Index 1 (97% over 37bp)
GAGATCGGAAGAGCACACGTCTGAACTCCAGTCACCTCAGAATCTCGTAT	1251	0.10448325887543942	TruSeq Adapter, Index 1 (97% over 37bp)

# Overrepresented sequences



- Here < 1% of sequences contain adapters
  - trimming optional

Sequence	Count	Percentage	Possible Source
AACGACTCTCGGCAACGGATATCTCGGCTCTCGCATCGATGAAGAACGTA	102020	1.0707851766890004	No Hit
AATTCTAGAGCTAATACGTGCAACAAACCCCGACTTATGGAAGGGACGCA	89437	0.9387160737848865	No Hit
AAAGGATTGGCTCTGAGGGCTGGGCTCGGGGGTCCCAGTTCGGAACCCGT	89427	0.9386111154260659	No Hit
TACCTGGTTGATCCTGCCAGTAGTCATATGCTTGTCTCAAAGATTAAGCC	87604	0.9194772066130483	No Hit
ATTGGCTCTGAGGGCTGGGCTCGGGGGTCCCAGTTCGGAACCCGTCTGGCT	65829	0.6909303802809273	No Hit
TCTAGAGCTAATACGTGCAACAAACCCCGACTTATGGAAGGGACGCATTT	65212	0.6844544495416888	No Hit
TAAAACGACTCTCGGCAACGGATATCTCGGCTCTCGCATCGATGAAGAAC	61582	0.646354565289767	No Hit
CTCGGATAACCGTAGTAATTCTAGAGCTAATACGTGCAACAAACCCCGAC	59180	0.6211435675010296	No Hit
ATGGATCCGTAACCTTCGGGAAAAGGATTGGCTCTGAGGGCTGGGCTCGGG	56982	0.598073720232235	No Hit
AAAACGACTCTCGGCAACGGATATCTCGGCTCTCGCATCGATGAAGAACG	54813	0.5753082522040206	No Hit
CTAGAGCTAATACGTGCAACAAACCCCGACTTATGGAAGGGACGCATTTA	40019	0.4200328561646452	No Hit
AGAACTCCGCAGTTAAGCGTGCTTGGGCGAGAGTAGTACTAGGATGGGTG	39753	0.4172409638200141	No Hit
ACTCGGATAACCGTAGTAATTCTAGAGCTAATACGTGCAACAAACCCCGA	38867	0.4079416532284981	No Hit
ACGACTCTCGGCAACGGATATCTCGGCTCTCGCATCGATGAAGAACGTAG	38438	0.40343893963508914	No Hit
ACTTCGGGAAAAGGATTGGCTCTGAGGGCTGGGCTCGGGGGTCCCAGTTC	37406	0.3926072370047907	No Hit
AGATCGGAAGAGCACACGTCTGAACTCCAGTCACTGACCAATCTCGTATG	34199	0.35894709133098535	TruSeq Adapter, Index 4 (100% over 49bp)
GAACCTTGGGATGGGTCTGGCCGGTCCGCCTTTGGTGTGCATTGGTCTGGCT	34099	0.3578975077427782	No Hit

# Overrepresented sequences



- Here nearly 1/3 of sequences some type of non-adapter contamination
  - **BLAST** the sequence to identify it

Sequence	Count	Percentage	Possible Source
GAAGGTCACGGCGAGACGAGCCGTTTATCATTACGATAGGTGTCAAGTGG	5632816	32.03026785752871	No Hit
TATTCTGGTGTCTAGGCGTAGAGGAACAACACCAATCCATCCCGAACTT	494014	2.8091456822607364	No Hit
TCAAACGAGGAAAGGCTTACGGTGGATACCTAGGCACCCAGAGACGAGGA	446641	2.539765344040083	No Hit
TAAAACGACTCTCGGCAACGGATATCTCGGCTCTCGCATCGATGAAGAAC	179252	1.0192929387357474	No Hit
GAAGGTCACGGCGAGACGAGCCGTTTATCATTACGATAGGGGTCAAGTGG	171681	0.9762414422996221	No Hit
AACGACTCTCGGCAACGGATATCTCGGCTCTCGCATCGATGAAGAACGTA	143415	0.8155105483274229	No Hit
AGAACATGAAACCGTAAGCTCCCAAGCAGTGGGAGGAGCCCTGGGCTCTG	111584	0.6345077504066322	No Hit
AAAACGACTCTCGGCAACGGATATCTCGGCTCTCGCATCGATGAAGAACG	111255	0.6326369351474214	No Hit
ATTACGATAGGTGTCAAGTGGAAAGTGCAGTGATGTATGCAGCTGAGGCAT	73682	0.41898300890326096	No Hit
GAAGGTCACGGCGAGACGAGCCGTTTATCATTACGATAGGTGTCAAGGGG	71661	0.4074908580252516	No Hit
GGATGCGATCATACCAGCACTAATGCACCGGATCCCATCAGAACTCCGCA	69548	0.3954755612388914	No Hit
ATATTCTGGTGTCTAGGCGTAGAGGAACAACACCAATCCATCCCGAACT	54017	0.30716057099328803	No Hit



# Dealing with adapters

- Three main options:
  1. Hard trim all sequences by specific amount
  2. Remove adapters specifically
  3. Perform a local (vs global) alignment

# Hard trim by specific length



- E.g. trim 100 base reads to 50 bases
- **Pro:**
  - Can eliminate vast majority of adapter contamination
  - Fast, easy to perform
  - Low quality 3' bases also removed
- **Con:**
  - Removes information you may want
    - e.g. splice junctions for RNAseq, coverage for mutation analysis
  - Not suitable for very short library fragments
    - e.g. miRNA libraries

# Trim adapters specifically



- **Pro:**

- Can eliminate vast majority of adapter contamination
- Minimal loss of sequence information
  - still ambiguous: are 3'-most bases part of sequence or adapter?

- **Con:**

- Requires knowledge of insert fragment structure and adapters
- Slower process; more complex to perform
- Results in a heterogeneous pool of sequence lengths
  - can confuse some downstream tools (rare)

# FASTQ trimming



- Tools:
  - **cutadapt** – <https://code.google.com/p/cutadapt/>
  - **trimmomatic** – <http://www.usadellab.org/cms/?page=trimmomatic>
  - FASTX-Toolkit – [http://hannonlab.cshl.edu/fastx\\_toolkit/](http://hannonlab.cshl.edu/fastx_toolkit/)
- Features:
  - hard-trim specific number of bases
  - trimming of low quality bases
  - specific trimming of adapters
  - support for trimming paired end read sets (except FASTX)
    - typically, reads less than a specified length *after trimming* are discarded
    - leads to different sets of R1 and R2 reads unless care is taken
      - aligners do not like this!



# Local vs. Global alignment



- **Global**
  - requires query sequence to map **fully** (end-to-end) to reference
- **Local**
  - allows a **subset** of the query sequence to map to reference

*global (end-to-end)  
alignment of query*

*local (subsequence)  
alignment of query*

**CACAAGTACAATTATACAC**

**CTAGCTTATCGCCCTGAA**GGACT

TACATACACAAGTACAATTATACACAGACATTAGTTCTTATCGCCCTGAAAATTCTCC

*reference sequence*



# Perform local alignment

- **Pro:**
  - mitigates adapter contamination while retaining full query sequence
  - minimal ambiguity
    - still ambiguous: are 5'/3'-most bases part of sequence or adapter?
- **Con:**
  - not supported by many aligners
    - e.g. not by the **tophat** splice-aware aligner for RNAseq
  - slower alignment process
  - more complex post-alignment processing may be required
- Aligners with local alignment support:
  - **bwa mem**
  - **bowtie2 --local**

# FastQC Sequence duplication report

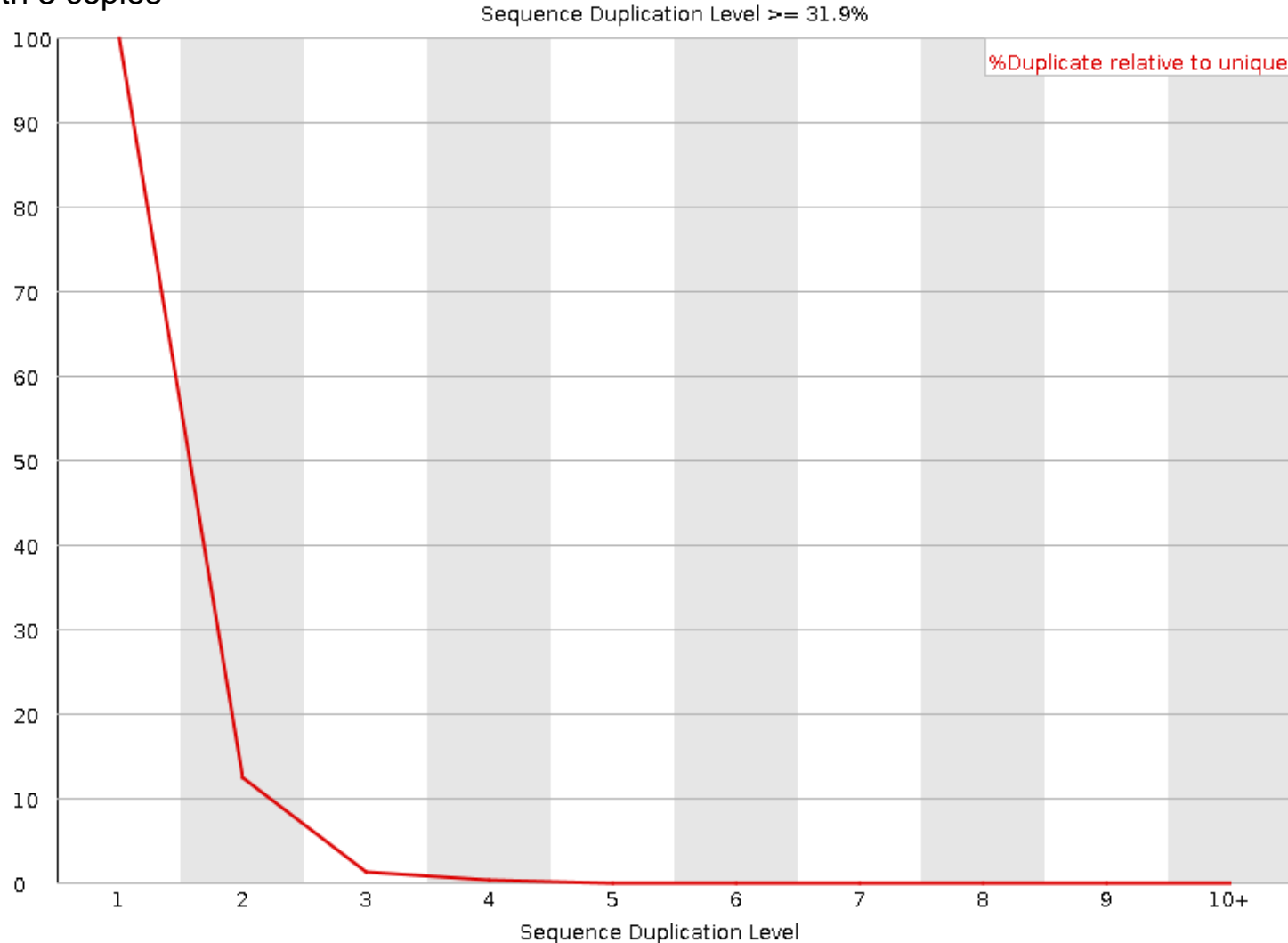
## Yeast ChIP-seq



for every 100 unique sequences  
there are:

- ~12 sequences w/2 copies
- ~1-2 with 3 copies

Some duplication expected due to IP enrichment



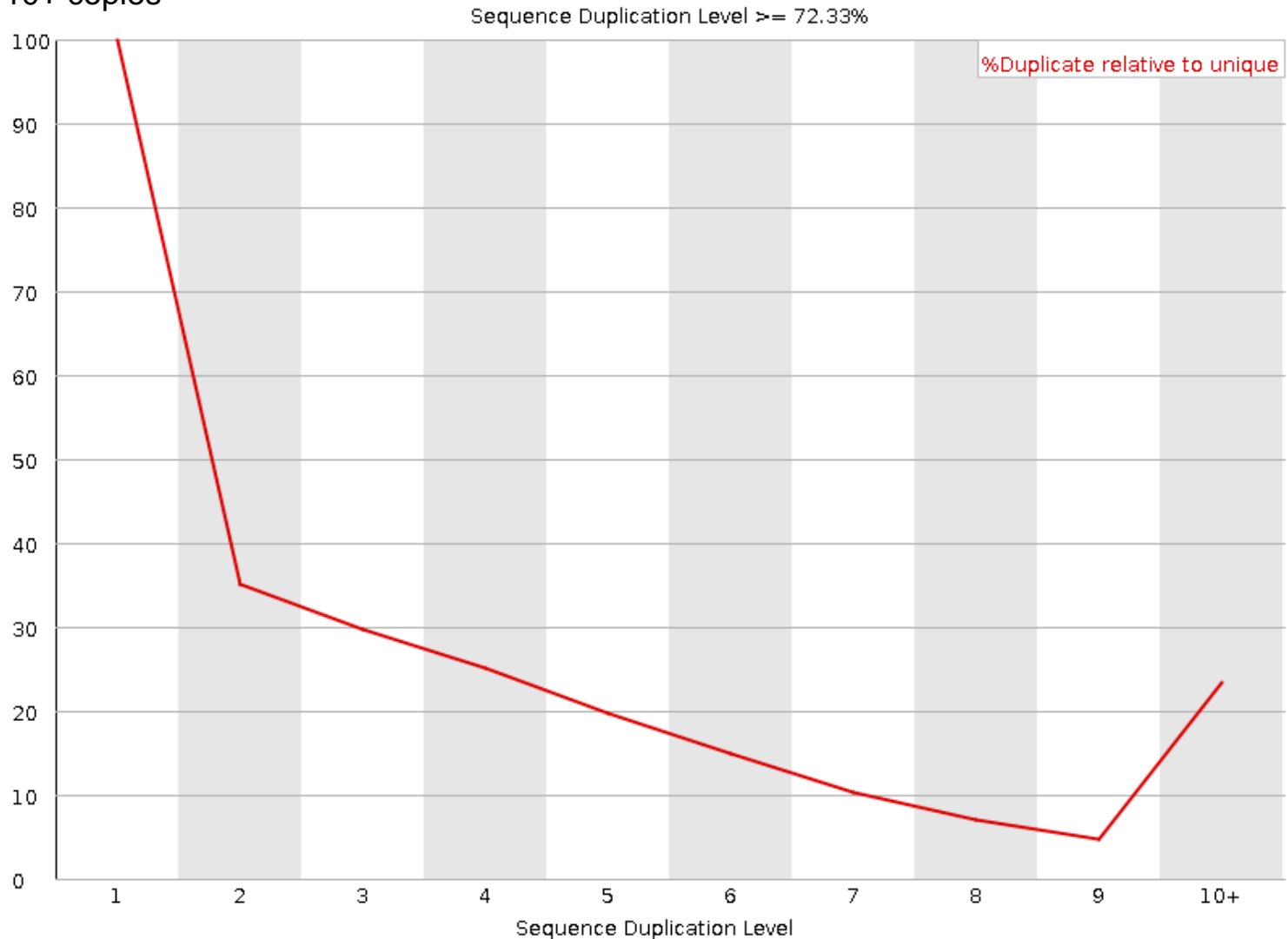
# Sequence duplication report

## Yeast ChIP-exo

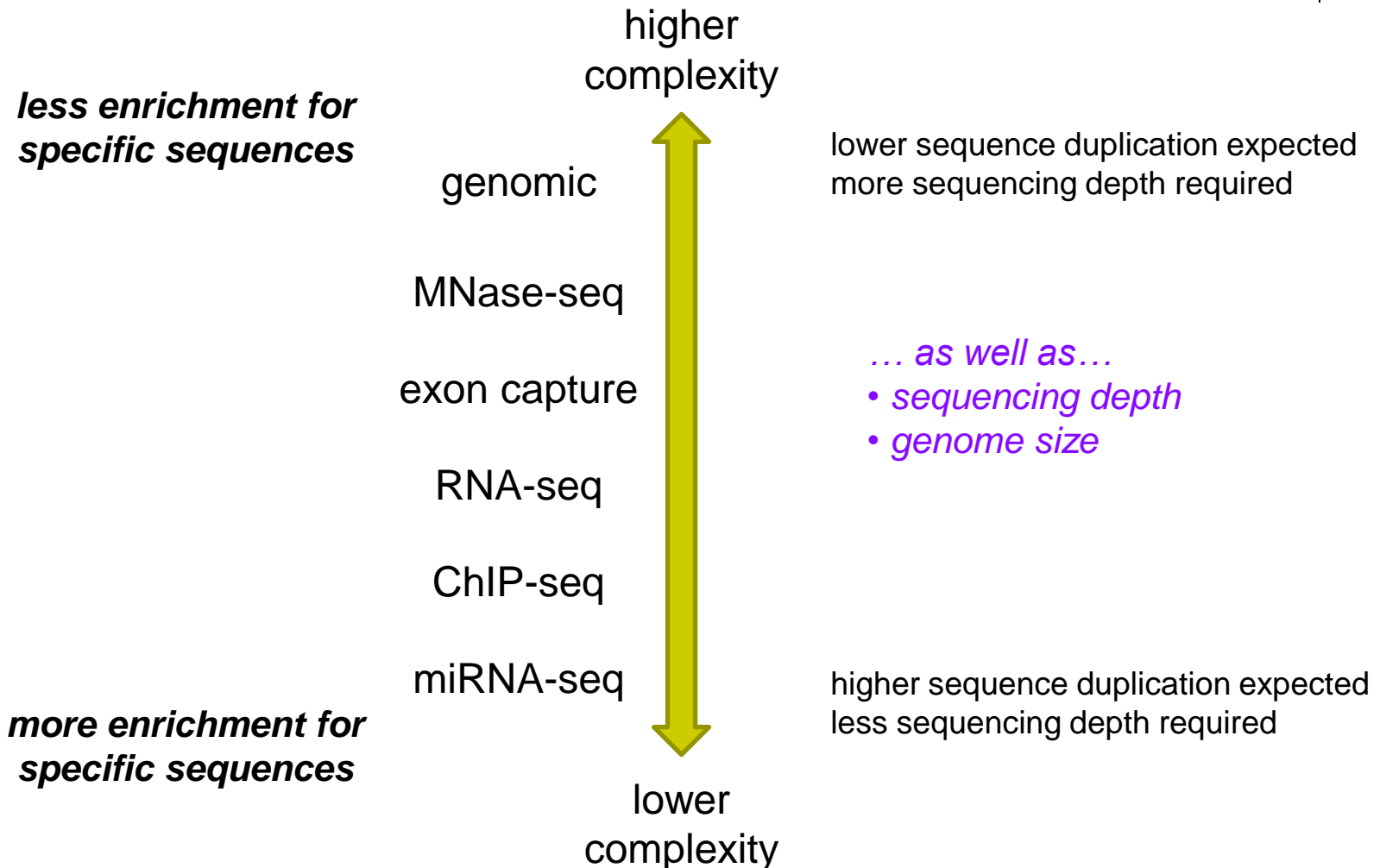


for every 100 unique sequences  
there are:

~35 sequences w/2 copies    **Success! Protocol *expected* to have high duplication**  
~22 with 10+ copies



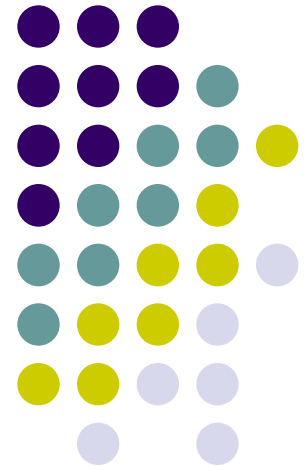
# Library complexity is primarily a function of experiment type



# Part 4: Alignment to a reference assembly

---

- Alignment overview & concepts
- Preparing a reference genome
- Alignment steps



# Short Read Aligners



- Short read mappers determine the placement of query sequences against a known reference
  - **BLAST**:
    - one query sequence (or a few)
    - many matches for each
  - short read aligners
    - many millions of query sequences
    - want only one “best” mapping (or a few)
  - many such aligners available
    - [http://en.wikipedia.org/wiki/List\\_of\\_sequence\\_alignment\\_software](http://en.wikipedia.org/wiki/List_of_sequence_alignment_software)
- We use 2 of the most popular
  - **bwa** (Burrows Wheeler Aligner) by Heng Li
    - <http://bio-bwa.sourceforge.net/>
  - **bowtie2** – part of the Johns Hopkins Tuxedo suite of tools
    - <http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml>

# Aligner criteria



- Adoption and currency
  - widespread use by bioinformatics community
  - still being actively developed
- Features
  - well understood algorithm(s)
  - support for a variety of input formats and read lengths
  - detection of indels and gaps
  - makes use of base qualities
  - handling of multiple matches
- Usability
  - configurability and transparency of options
  - ease of installation and use
- Resource requirements
  - speed (“fast enough”)
  - scalability (takes advantage of multiple processors)
  - reasonable memory footprint



# Mapping vs. Alignment

- **Mapping** determines one or more positions (a.k.a “seeds” or “hits”) where a read shares sequence with the reference
- **Alignment** starts with the seed and determines how read bases are best matched, base-by-base, around the seed
- Mapping quality and alignment scores are both reported
- High **mapping quality** ≠ High **alignment score**
  - **mapping quality** describes **positioning**
    - reflects the probability that the read is *incorrectly* mapped to the reported location
    - is a Phred score:  $P(\text{mis-mapped}) = 10^{-\text{mappingQuality}/10}$
    - reflects the complexity/information content of the sequence (“mappability”)
  - **alignment score** describes **fit**
    - reflects the correspondence between the read and the reference sequences

- low mapping quality
- high alignment score

Read 1

or

Read 2

- high mapping quality
- low alignment score

```
ATCGGGAGATCC      or      ATCGGGAGATCC      GCGTAGTCTGCC
|||||
...TAATCGGGAGATCCGC...TTATCGGGAGATCCGC... ..TAGCCTAGTGTGCCGC...
```

Reference Sequence

# Some Aligners



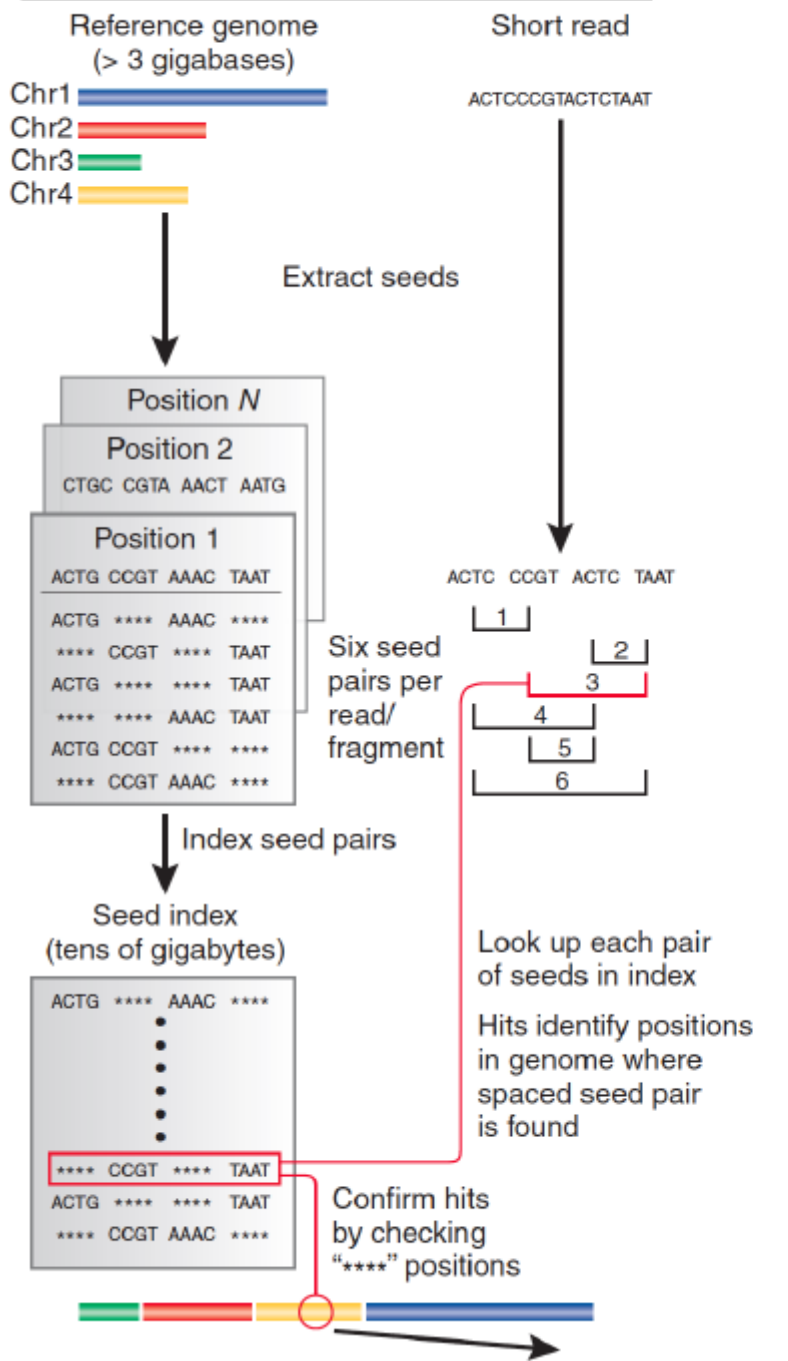
Two main mapping algorithms: *spaced seeds*, *suffix-array tries*

	Algorithm	Gapped	Quality-aware	Colorspace aware
BLAST	Hash table	Y	N	N
BLAT/SSHA2	Hash table	N	N	N
MAQ	Spaced seed	N	N	N
RMAP	Spaced seed	N	Y	N
ZOOM	Spaced seed	N	-	N
SOAP	Spaced seed	N	N	N
Eland	Spaced seed	N	N	N
SHRIMP	Q-gram/multi-seed	Y	Y	Y
BFAST	Q-gram/multi-seed	Y	Y	Y
Novoalign	Multi-seed + Vectorized SW	Y	Y	Y
clcBio	Multi-seed + Vectorized SW	Y	Y	Y
MUMmer	Tries	Y	N	N
OASIS	Tries	Y	-	-
VMATCH	Tries	Y	-	-
BWA/BWA-SW	Tries	Y	Y	Y
BOWTIE	Tries	Y	Y	Y
SOAP2	Tries	Y	N	N
Saruman	Exact (GPU)	Y	-	N

courtesy of Matt Vaughn, TACC

trie = tree structure for fast text retrieval.

## a Spaced seeds



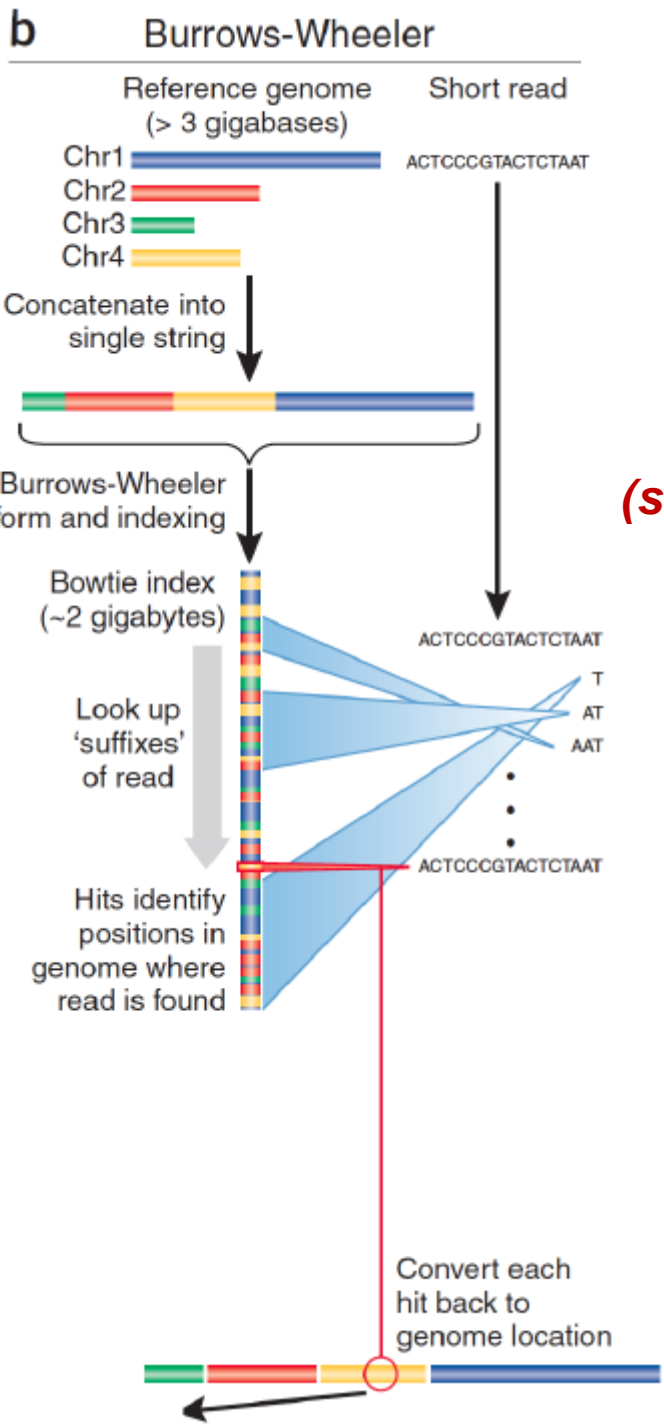
Hash table enables lookup of exact matches.

Subsequence	Reference Positions
ATAGCTAATCCAAA	2341, 2617264
ATAGCTAATCCAAT	
ATAGCTAATCCAAC	134, 13311, 732661,
ATAGCTATCCAAG	
ATAGCTAATCCATA	
ATAGCTAATCCATT	3452
ATAGCTAATCCATC	
ATAGCTATCCAATG	234456673

Table is sorted and complete so you can jump immediately to matches. (But this can take a lot of memory.)

May include N bases, skip positions, etc.

Trapnell, C. & Salzberg, S. L. How to map billions of short reads onto genomes. *Nature Biotech.* 27, 455–457 (2009).

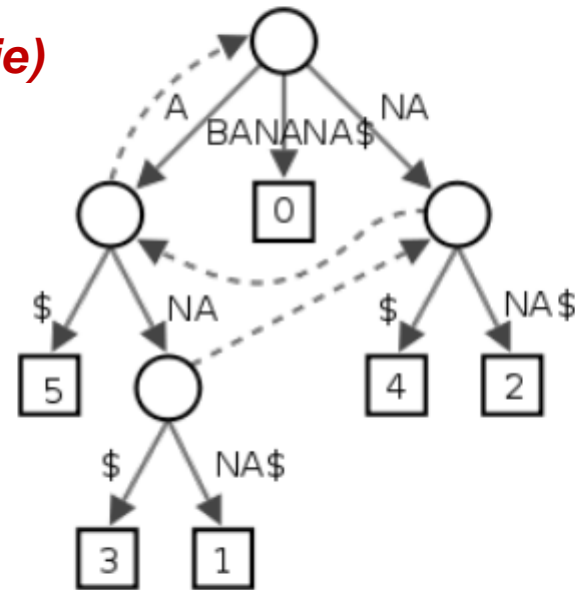


**Burrows-Wheeler transform** compresses sequence.

<b>Input</b>	SIX.MIXED.PIXIES.SIFT.SIXTY.PIXIE.DUST.BOXES
<b>Output</b>	TEXYDST.E.IXIXIXSSMPPS.B..E.S.EUSFXDIIIOIIT

**Suffix tree** enables fast lookup of subsequences.

(*suffix array trie*)

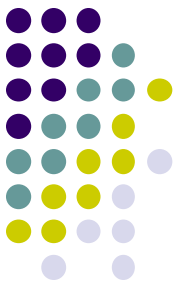


[http://en.wikipedia.org/wiki/Suffix\\_tree](http://en.wikipedia.org/wiki/Suffix_tree)

Exact matches at all positions below a node.

Trapnell, C. & Salzberg, S. L. How to map billions of short reads onto genomes. *Nature Biotech.* 27, 455–457 (2009).

# Alignment via dynamic programming



- Dynamic programming algorithm (Smith-Waterman | Needleman-Wunsch)

	G	A	A	T	T	C	A	G	T	T	A
G	0	0	0	0	0	0	0	0	0	0	0
G	0	1	1	1	1	1	1	1	1	1	1
G	0	1	1	1	1	1	1	2	2	2	2
A	0	1	1	2	2	2	2	2	2	2	3
T	0	1	2	2	3	3	3	3	3	3	3
C	0	1	2	2	3	3	4	4	4	4	4
G	0	1	2	2	3	3	4	4	5	5	5
A	0	1	2	3	3	3	4	5	5	5	6

	G	A	A	T	T	C	A	G	T	T	A
G	0										
G		1									
G		1	1								
A				2	2						
T					3						
C						4	4				
G								5	5	5	
A											6

```

G _ A A T T C A G T T A
| | | | | | | | | |
G G _ A _ T C _ G _ _ A
  
```

- Alignment score =  $\sum$**

- match reward
- base mismatch penalty
- gap open penalty
- gap extension penalty
- rewards and penalties may be adjusted for quality scores of bases involved

Reference sequence

ATTTGCGATCGGATGAAGACGAA

|||||

ATTTGCGATCGGATGTTGACTTT

ATTTGCGATCGGATGAAGACG..AA

|||||XX||Xi||

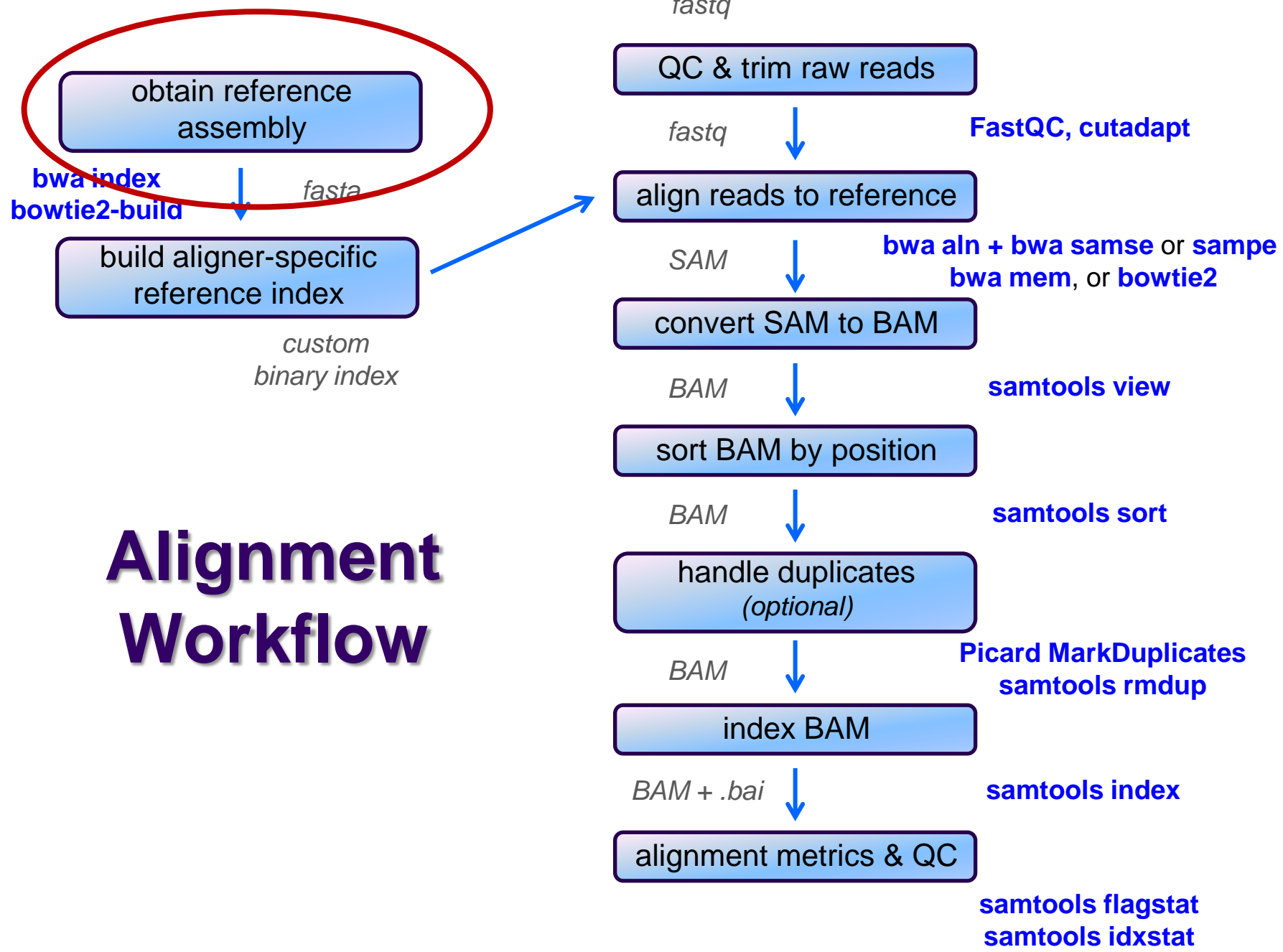
ATTTGCGATCGGATGTTGACTTTAA

# Paired End mapping



- Having paired-end reads improves mapping
  - mapping one read with high confidence anchors the pair
    - even when its mate read alone maps several places equally
- There is an expected insert size distribution based on the DNA fragment library
  - only one of a pair might map (*singleton/orphan*)
  - both reads can map within the most likely distance range (*proper pair*)
  - both reads can map but with an unexpected insert size or orientation (*discordant pair*)
- The insert size is reported in the alignment record for both proper and discordant pairs

# Alignment Workflow





# Obtaining a reference



- What is a reference?
  - any set of named sequences
    - e.g. names are chromosome names
    - technically referred to as “contigs”
- Assembled genomes
  - Ensembl, UCSC for eukaryotes
    - FASTA files (**.fa**, **.fasta**)
  - GenBank, NCBI for prokaryotes/microbes
    - Records contain both fasta sequences and annotations
- Any set of sequences of interest, e.g:
  - transcriptome (set of gene/cDNA sequences)
  - rRNA/tRNA genes (for filtering)
  - miRNA hairpin sequences from miRBase



# FASTA format



- FASTA files contain a set of sequence records
  - sequence name line
    - always starts with >
      - followed by name and other (optional) descriptive information
  - one or more sequence line(s)
    - never starts with >

- Mitochondrial chromosome sequence, human hg19

```
>chrM
GATCACAGGTCTATCACCCCTATTAACCACTCACGGGAGCTCTCCATGCAT
TTGGTATTTTCGTCTGGGGGGTGTGCACGCGATAGCATTGCGAGACGCTG
GAGCCGGAGCACCCCTATGTTCGAGTATCTGTCTTTGATTCTGCCTCATT ...
```

- Let-7e miRNA, human miRBase v21

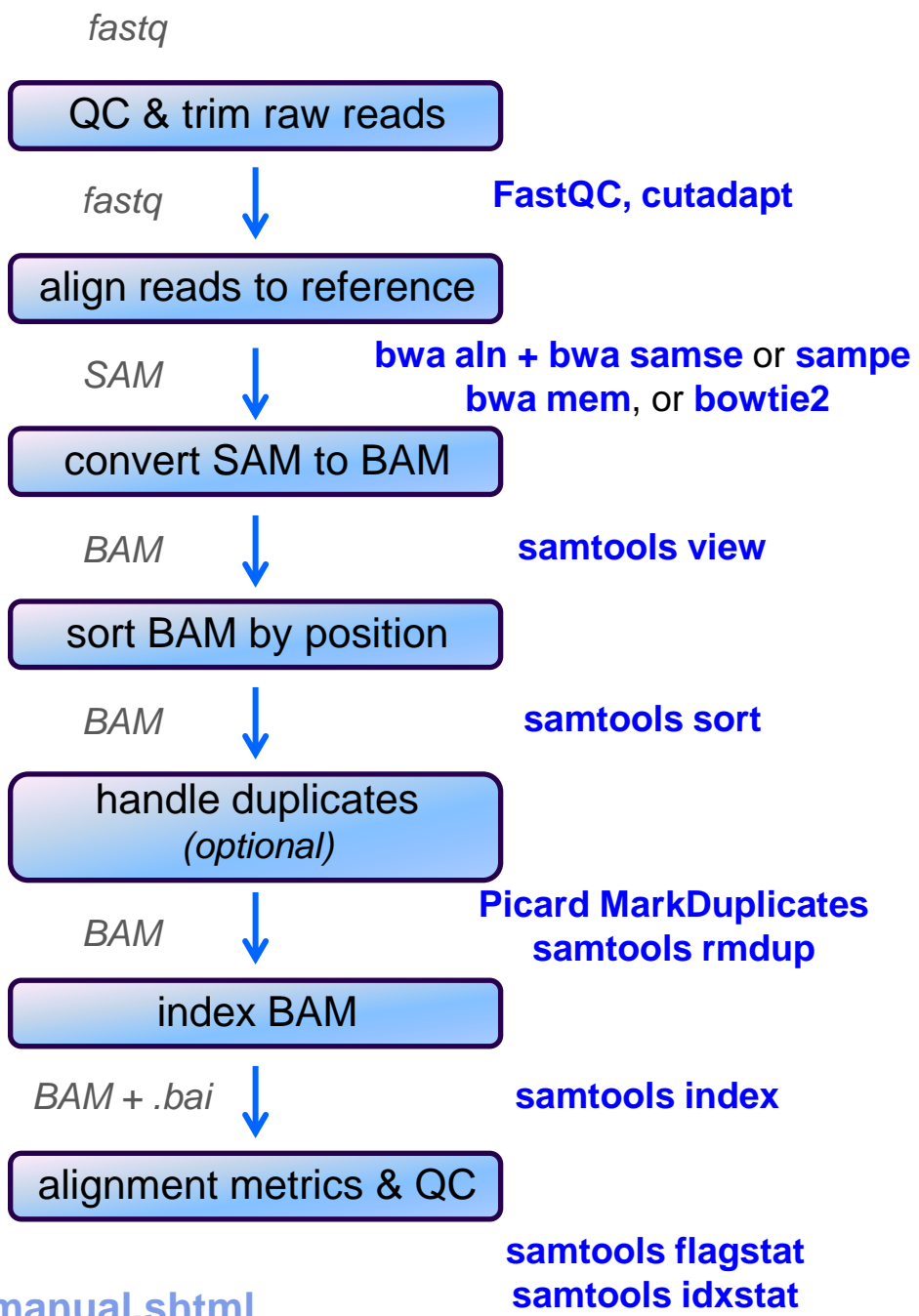
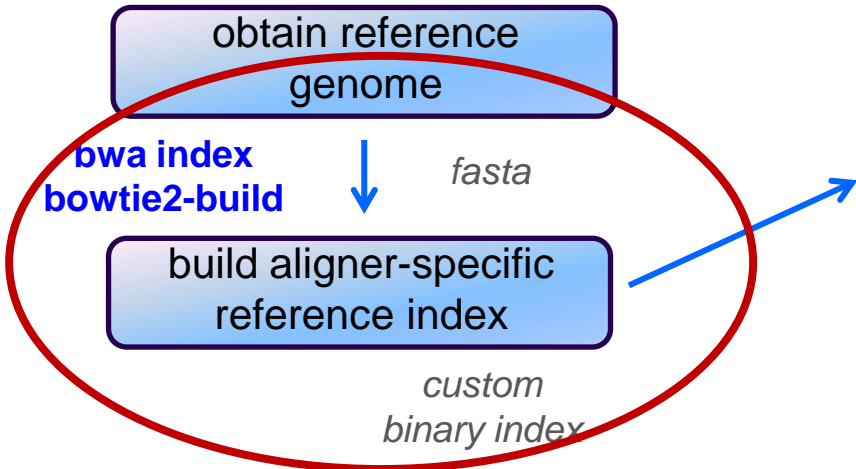
```
>hsa-let-7e MI0000066 Homo sapiens let-7e stem-loop
CCCGGGCTGAGGTAGGAGGTTGTATAGTTGAGGAGGACACCCAAGGAGATCACTATACGG
CCTCCTAGCTTTCCCCAGG
```



# Reference considerations

- Is it appropriate to your study?
  - close enough to your species? complete?
- Does it contain repeats? What kinds?
  - know this up front or you will be confused
- From which source? And which version?
  - UCSC hg19 vs Ensembl GRCh37
- What annotations exist?
  - references lacking feature annotations are much more challenging
- Watch out for sequence name issues!
  - sequence names are different between UCSC/Ensembl
    - e.g. “chr12” vs “12”
  - annotation sequence names must match names in your reference!
  - long sequence names can cause problems
    - rename: `>hsa-let-7e_MI0000066_Homo_sapiens_let-7e_stem-loop`
    - to: `>hsa-let-7e`

# Alignment Workflow



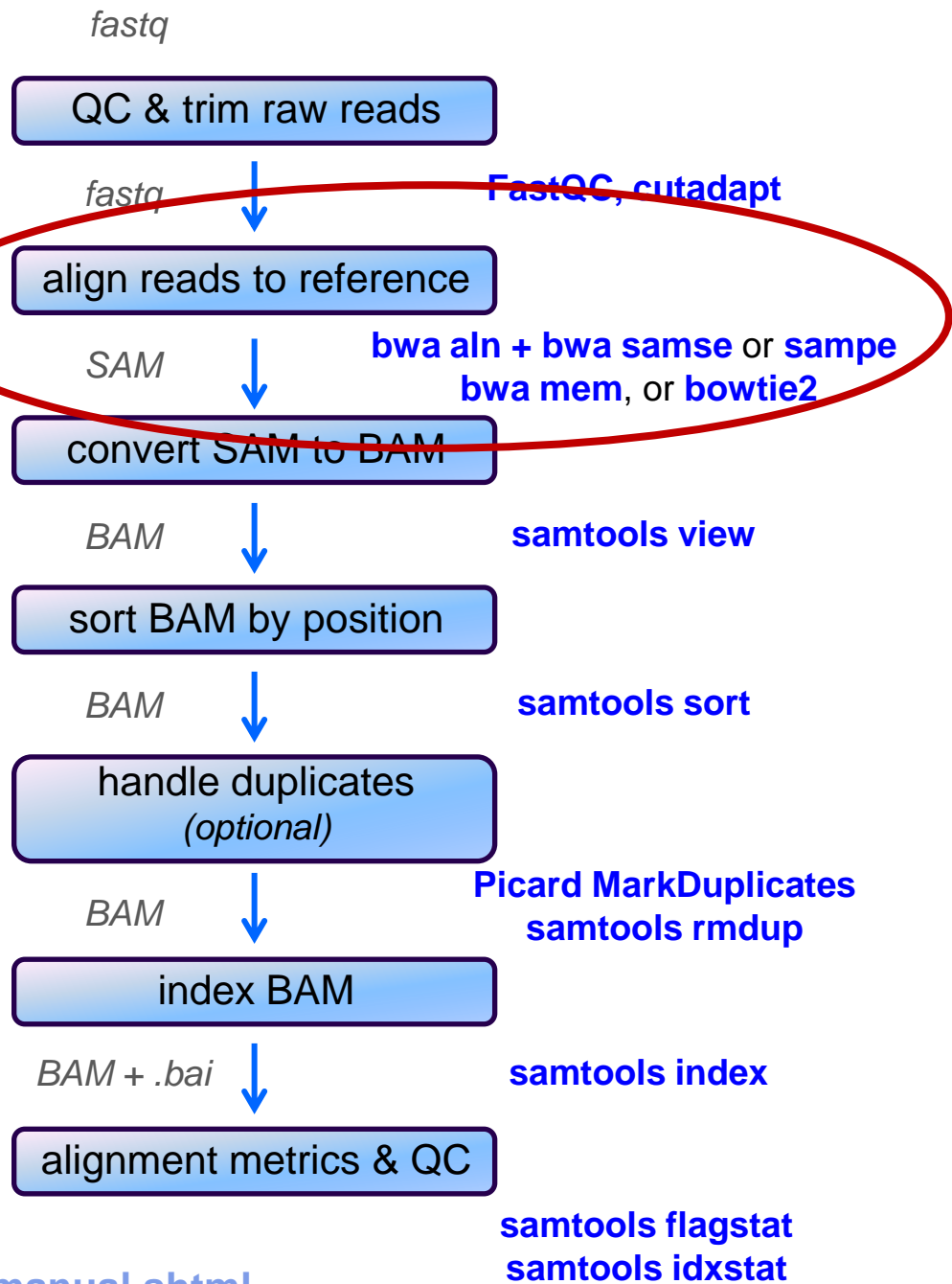
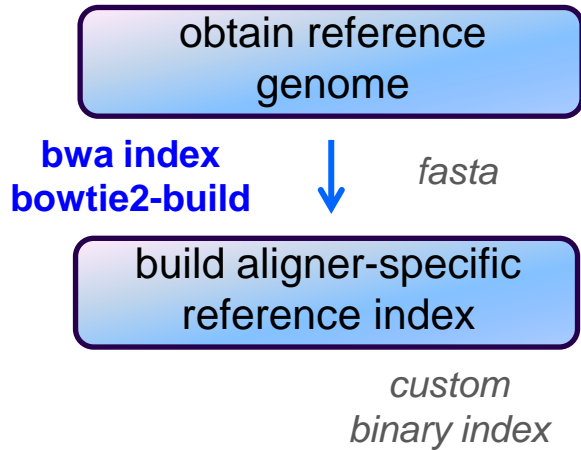
<http://bio-bwa.sourceforge.net/bwa.shtml>

<http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml>

# Building a reference index



- Index format is specific to each aligner
  - may take several hours to build
    - but you build each index once, use for multiple alignments
- Input:
  - a FASTA file
- Output:
  - a number of binary files the aligner will use
- Best practice:
  - build each index in its own appropriately named directory, e.g.
    - `refs/bowtie2/UCSC/hg19`
    - `refs/bwa/Ensembl/GRCh37`



# Alignment Workflow

<http://bio-bwa.sourceforge.net/bwa.shtml>

<http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml>

# SAM file format



- Aligners take FASTQ as input, output alignments in **S**equence **A**lignment **M**ap (SAM) format
  - community file format that describes how reads align to a reference
    - can also include unmapped reads
  - the Bible: <http://samtools.github.io/hts-specs/SAMv1.pdf>
- SAM file consists of:
  - a header
    - includes reference sequence names and lengths
  - alignment records, one for each sequence read
    - alignments for R1 and R2 reads have separate records, with fields that refer to the mate
    - 11 fixed fields + extensible-format **key:type:value** tuples

# SAM file format

## Fixed fields



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

SRR030257.264529 99 NC\_012967 1521 29 34M2S = 1564 79

CTGGCCATTATCTCGGTGGTAGGACATGGCATGCC

AAAAAA;AA;AAAAA??A%.;?&'3735',()0\*,

XT:A:M NM:i:3 SM:i:29 AM:i:29 XM:i:3 XO:i:0 XG:i:0 MD:Z:23T0G4T4

SRR030257.2669090 147 NC\_012967 1521 60 36M = 1458 -99

CTGGCCATTATCTCGGTGGTAGGIGATGGTATGCGC

<<9:<<AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

XT:A:U NM:i:0 SM:i:37 AM:i:37 XO:i:1 X1:i:0 XM:i:0 XO:i:0 XG:i:0 MD:Z:36



# SAM format – Bitwise flags

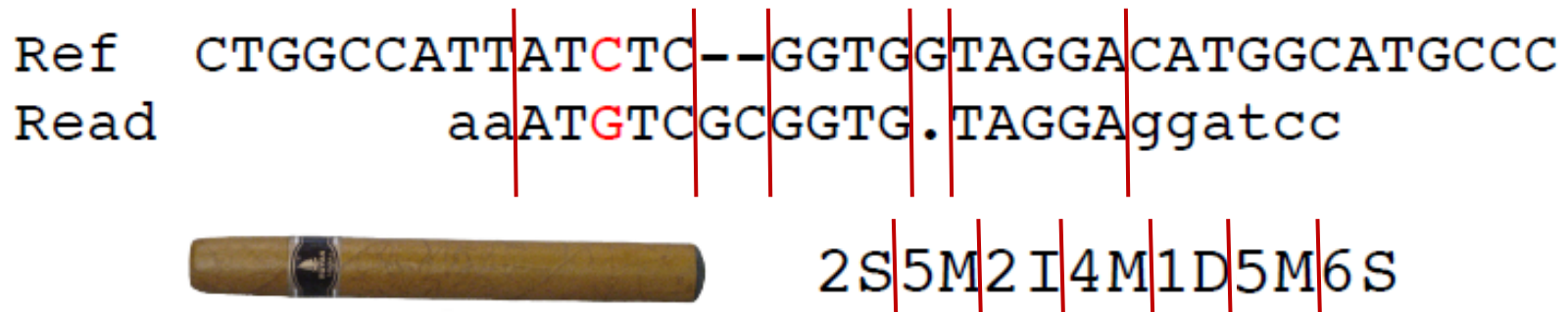
Bit	Description	
0x1	template having multiple segments in sequencing	1 = part of a read pair
0x2	each segment properly aligned according to the aligner	1 = “properly” paired
0x4	segment unmapped	1 = read did <b>not</b> map
0x8	next segment in the template unmapped	1 = mate did <b>not</b> map
0x10	SEQ being reverse complemented	1 = minus strand read
0x20	SEQ of the next segment in the template being reversed	1 = mate on minus strand
0x40	the first segment in the template	1 = R1 read
0x80	the last segment in the template	1 = R2 read
0x100	secondary alignment	1 = secondary possible hit
0x200	not passing quality controls	
0x400	PCR or optical duplicate	1 = marked as duplicate

									Decimal	Hex
SRR030257.264529	<b>99</b>	NC_012967	1521	29	34M2S	=	1564	79	99	= 0x63
CTGGCCATTATCTCGGTGGTAGGACATGGCATGCC									= 64	= 0x40
AAAAAA;AA;AAAAA??A%.;?&'3735',()0*									+ 32	+ 0x20
XT:A:M NM:i:3 SM:i:29 AM:i:29 XM:i:3 XO:i:0 XG:i:0 MD:Z:23T0G4T4									+ 2	+ 0x02
									+ 1	+ 0x01
SRR030257.2669090	<b>147</b>	NC_012967	1521	60	36M	=	1458	-99	147	= 0x93
CTGGCCATTATCTCGGTGGTAGGTGATGGTATGCGC									= 128	= 0x80
<<9:<<AAAAAAAAAAAAAAAAAAAAAAAAAAAA									+ 16	+ 0x10
XT:A:U NM:i:0 SM:i:37 AM:i:37 X0:i:1 X1:i:0 XM:i:0 XO:i:0 XG:i:0 MD:Z:36									+ 2	+ 0x02
									+ 1	+ 0x01





# Sometimes a CIGAR is just a way of describing how a read is aligned...

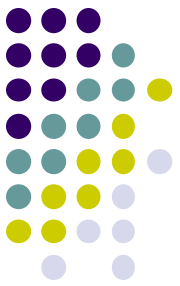


Op	BAM	Description
M	0	alignment match (can be a sequence match or mismatch)
I	1	insertion to the reference
D	2	deletion from the reference
* N	3	skipped region from the reference <i>“N” indicates splicing event in tophat RNAseq BAMs</i>
S	4	soft clipping (clipped sequences present in SEQ)
* H	5	hard clipping (clipped sequences NOT present in SEQ)
* P	6	padding (silent deletion from padded reference)
* =	7	sequence match
* X	8	sequence mismatch

CIGAR = "Concise Idiosyncratic Gapped Alignment Report"

# SAM file format

## key:type:value tuples



Tag <sup>1</sup>	Type	Description
X?	?	Reserved fields for end users (together with Y? and Z?)
...		
MD	Z	String for mismatching positions. <i>Regex</i> : [0-9]+((( [A-Z]   \^ [A-Z] +) [0-9] +) ) * <sup>2</sup>
MQ	i	Mapping quality of the mate/next segment
NH	i	Number of reported alignments that contains the query in the current record
NM	i	Edit distance to the reference, including ambiguous bases but excluding clipping
...		

*details alignment of query to reference*

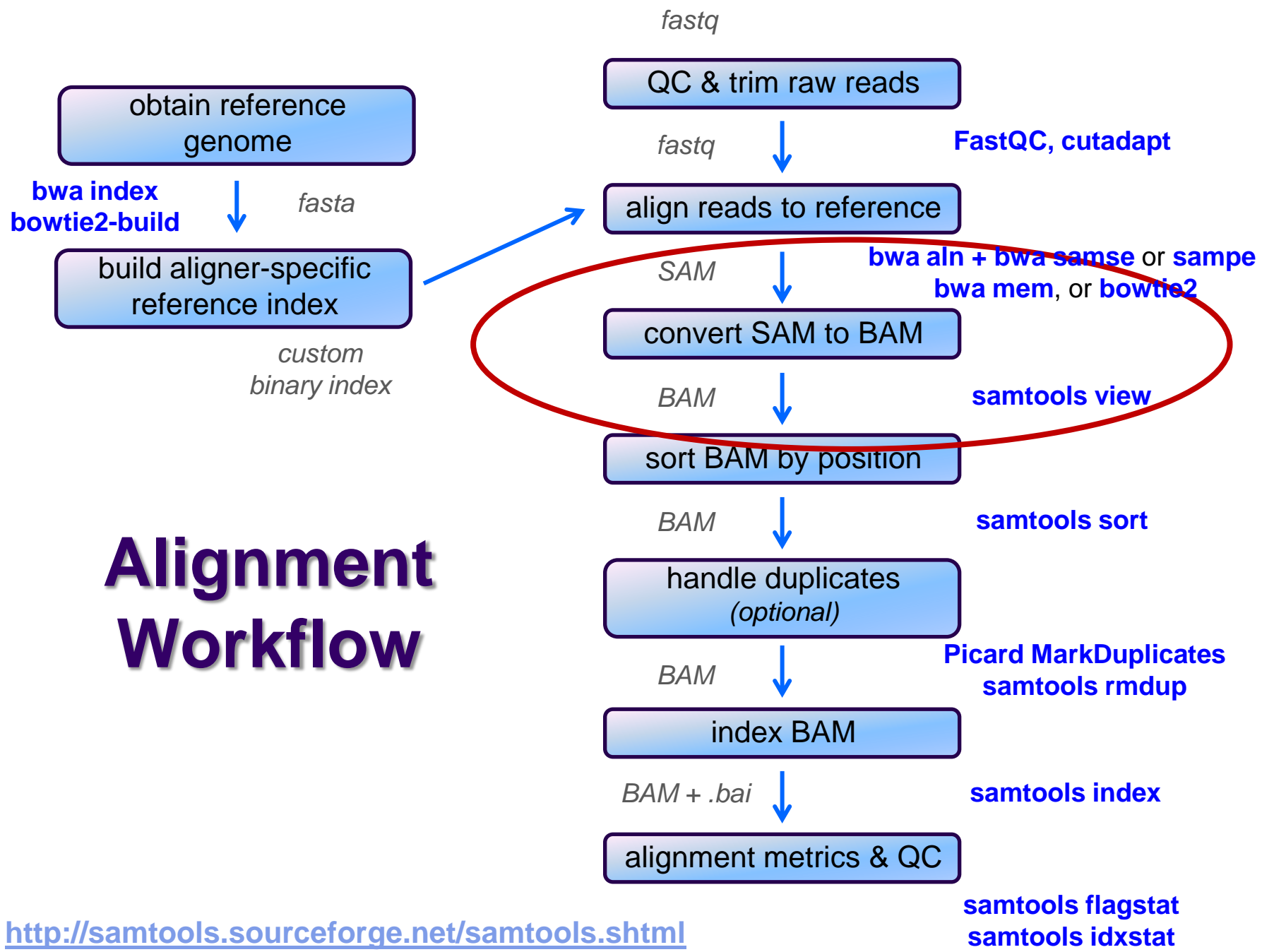
*# mismatches + insertions + deletions*

<sup>2</sup>The MD field aims to achieve SNP/indel calling without looking at the reference. For example, a string '10A5^AC6' means from the leftmost reference base in the alignment, there are 10 matches followed by an A on the reference which is different from the aligned read base; the next 5 reference bases are matches followed by a 2bp deletion from the reference; the deleted sequence is AC; the last 6 bases are matches. The MD field ought to match the CIGAR string.

```

SRR030257.264529    99  NC_012967    1521    29  34M2S    =    1564    79
CTGGCCATTATCTCGGTGGTAGGACATGGCATGCC
AAAAAA:AA;AAAAA??A%.;?&'3735',()0*,
XT:A:M NM:i:3 SM:i:29 AM:i:29 XM:i:3 XO:i:0 XG:i:0 MD:Z:23T0G4T4
    
```

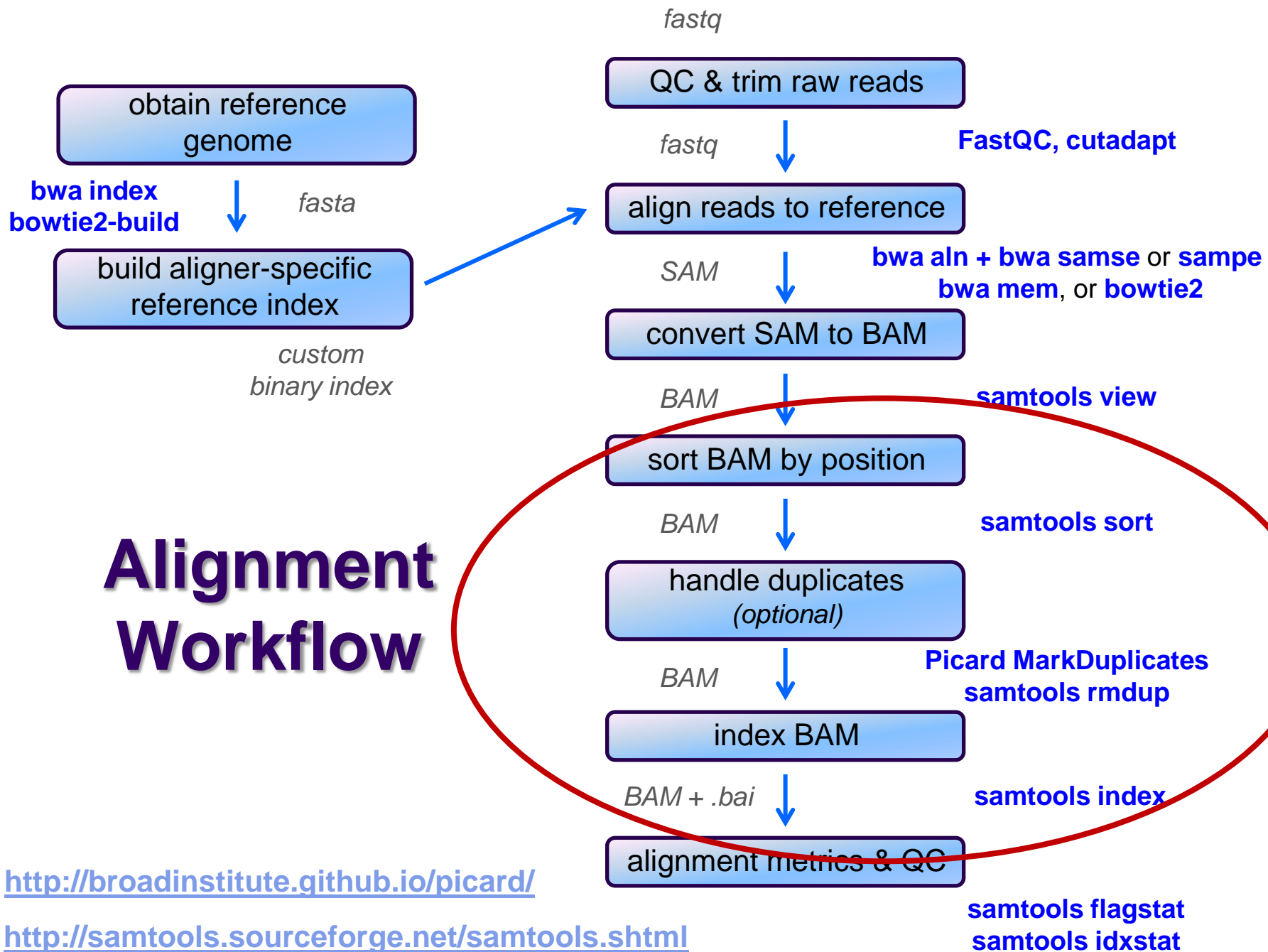
# Alignment Workflow



# SAM / BAM files



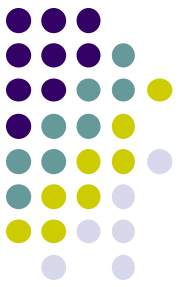
- SAM and BAM are two forms of the same data
  - SAM – **S**equence **A**lignment **M**ap
    - plain text format
  - BAM – **B**inary **A**lignment **M**ap
    - **same data** in a custom compressed (**gzip**'d) format
- Differences
  - BAMs are **much** smaller than SAM files due to compression
  - BAM files support fast random access; SAM files do not
    - requires the BAM file to be *indexed*
  - most tools support BAM format and may require indexing
- Best practices
  - remove intermediate SAM and BAM files created during alignment and only save the final sorted, indexed BAM
  - keep your alignment artifacts (BAM, statistics files, log files) separate from the original FASTQ files
    - alignments can be easily re-generated; raw sequences cannot



# Sorting / indexing BAM files



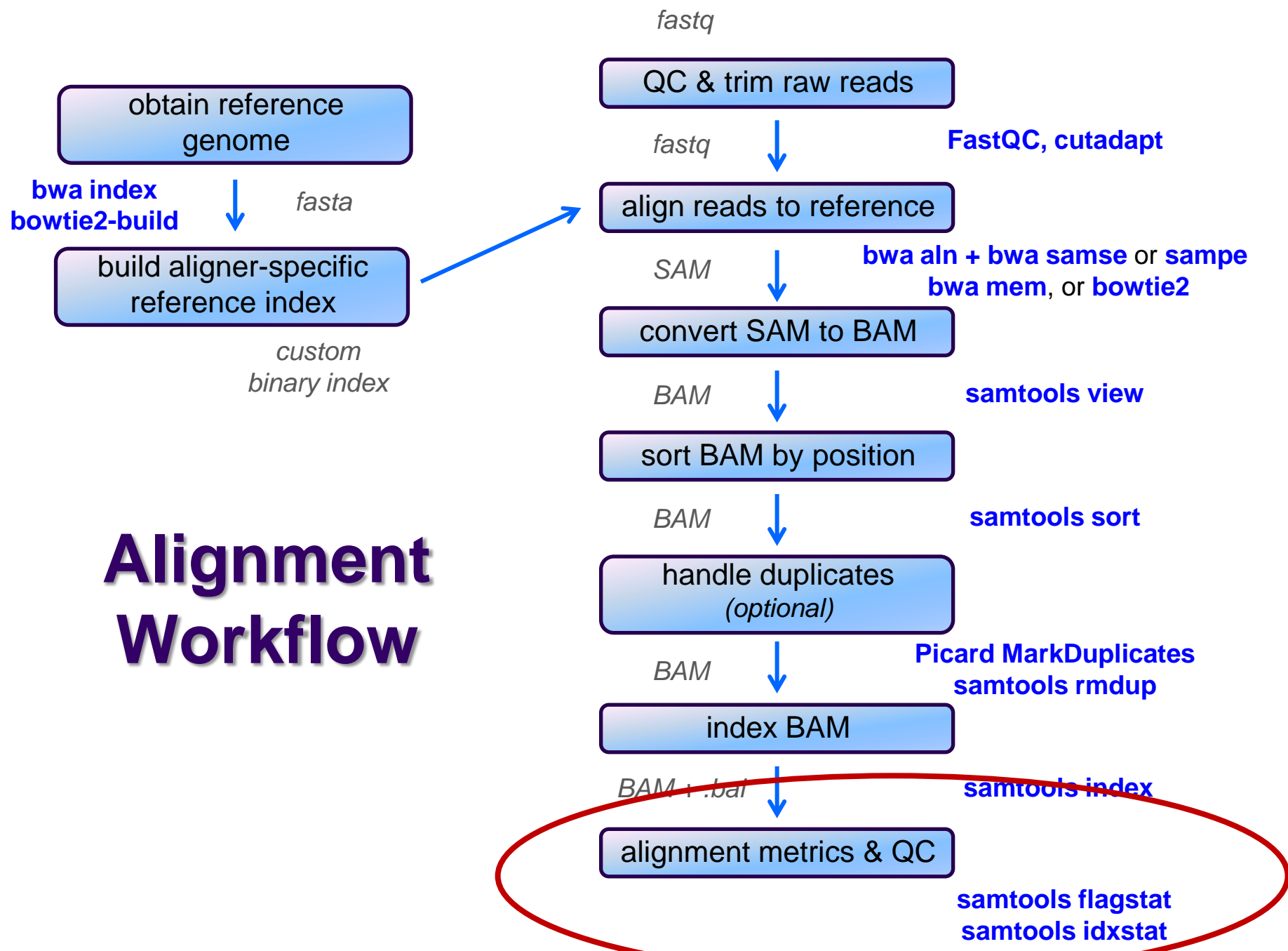
- SAM created by aligner contains read records in ***name order***
  - same order as read names in the input FASTQ file
    - R1, R2 have adjacent SAM records
  - SAM → BAM conversion does not change the name-sorted order
- Sorting BAM puts records in ***locus order***
  - by contig name then start position
    - contig name order given in SAM/BAM header
    - based on order of sequences in FASTA used to build reference
  - sorting is *very* compute and I/O intensive
    - can take several hours for large BAM
- Indexing a locus-sorted BAM allows fast random access
  - creates a binary alignment index file (**.bai**)
  - quite fast



# Handling Duplicates

- Optional step, but very important for many protocols
- Definition of duplicates:
  - single end reads or singleton/discordant alignment
    - alignments start at the same location & matching sequences
  - properly paired reads
    - pairs have same external coordinates, and matching R1/R2 sequences
- Two choices for handling:
  - **samtools rmdup** – **removes** duplicates entirely
    - faster, but data is lost
    - does not properly handle data from multiple lanes
  - **Picard MarkDuplicates** – **flags** duplicates only (0x400 bam flag)
    - slower, but all alignments are retained
    - alignments from different lanes/replicates are handled properly
  - both tools are quirky in their own ways

# Alignment Workflow





# Alignment metrics



- **samtools flagstat**

- simple statistics based on alignment record flag values
  - total sequences (R1+R2), total mapped
  - number properly paired
  - number of duplicates (0 if duplicates were not marked)

```
30146426 + 0 in total (QC-passed reads + QC-failed reads)
13532165 + 0 duplicates
28804693 + 0 mapped (95.55%:-nan%)
30146426 + 0 paired in sequencing
15073213 + 0 read1
15073213 + 0 read2
28546786 + 0 properly paired (94.69%:-nan%)
28712992 + 0 with itself and mate mapped
91701 + 0 singletons (0.30%:-nan%)
64973 + 0 with mate mapped to a different chr
50382 + 0 with mate mapped to a different chr (mapQ>=5)
```



# Computing average insert size

- Needed for RNAseq alignment using **tophat**
- Simple **awk** script that computes average insert size for a BAM
  - **-F 0x4** filter to **samtools view** says only consider mapped reads
    - technically “not unmapped”
  - the **-f 0x2** filter says consider only properly paired reads
    - they have reliable “insert size” values in column 9
  - insert size values are negative for minus strand reads
    - can ignore because each proper pair should have one plus and one minus strand alignment

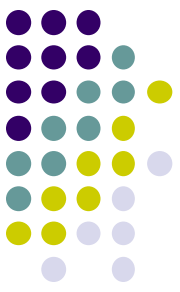
```
samtools view -F 0x4 -f 0x2 my_pe_data.bam | awk \  
'BEGIN{ FS="\t"; sum=0; nrec=0; }  
{ if ($9 > 0) {sum += $9; nrec++;} }  
END{ print sum/nrec; }'
```



# Interpreting alignment metrics

- Table below is taken from a spreadsheet I keep on all our alignments
  - all are yeast paired-end read datasets from ChIP-seq experiments
- Alignment rates
  - samples 1-3 have excellent alignment rates & good rates of proper pairing
  - sample 4
    - has an unusually low alignment rate for a ChIP-seq dataset
    - has a median insert size of only 109, and these were un-trimmed 50 bp reads
    - could 3' adapter contamination be affecting the alignment rate?
      - try re-aligning the sequences after trimming, say to 35 bases
      - see if the alignment rate improves

#	totSeq	totAlign	% align	numPair	pePrAln	% prPr	nDup	% dup	multiHit	% multi	iszMed
1	149,644,822	145,228,810	97.0%	74,822,411	72,221,545	96.5%	49,745,225	34%	16,216,807	11%	181
2	981,186	860,940	87.7%	490,593	424,915	86.6%	609,378	71%	127,987	15%	148
3	22,573,348	21,928,789	97.1%	11,286,674	10,783,971	95.5%	9,408,725	43%	3,711,004	17%	132
4	7,200,628	3,460,992	48.1%	3,600,314	1,626,121	45.2%	1,234,524	36%	649,690	19%	109



# Interpreting alignment metrics

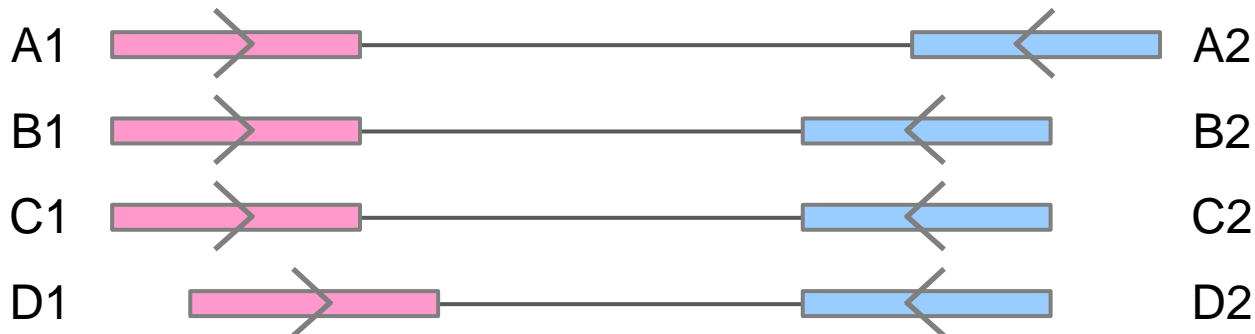
- Duplication rates
  - sample 1 is incredibly deeply sequenced (yeast genome only ~12 Mbase)
    - has a very low duplication rate considering
    - turns out this is a control dataset (Mock ChIP), so is a great control to use (wonderfully complex!)
  - sample 2 is not very deeply sequenced but has a high duplication rate (71%)
    - subtracting duplicates from total aligned leaves only ~250,000 non-dup reads
      - not enough for further analysis (prefer 500,000+)
  - sample 3 has reasonable sequencing depth with substantial duplication (43%)
    - still leaves plenty of non-duplicate reads (> 12 million)

#	totSeq	totAlign	% align	numPair	pePrAln	% prPr	nDup	% dup	multiHit	% multi	iszMed
1	149,644,822	145,228,810	97.0%	74,822,411	72,221,545	96.5%	49,745,225	34%	16,216,807	11%	181
2	981,186	860,940	87.7%	490,593	424,915	86.6%	609,378	71%	127,987	15%	148
3	22,573,348	21,928,789	97.1%	11,286,674	10,783,971	95.5%	9,408,725	43%	3,711,004	17%	132
4	7,200,628	3,460,992	48.1%	3,600,314	1,626,121	45.2%	1,234,524	36%	649,690	19%	109

# Read vs fragment duplication



- Consider the 4 fragments below
  - 4 R1 reads (pink), 4 R2 reads (blue)
- Duplication when only 1 end considered
  - A1, B1, C1 have identical sequences, D1 different
    - 2 unique + 2 duplicates = 50% duplication rate
  - B2, C2, D2 have identical sequences, A2 different
    - 2 unique + 2 duplicates = 50% duplication rate
- Duplication when both ends considered
  - fragments B and C are duplicates (same external sequences)
    - 3 unique + 1 duplicate = 25% duplication rate





# Alignment wrap up

- Many tools involved
  - choose one (or two) and learn their options well
- Many steps are involved in the full alignment workflow
  - important to go through manually a few times for learning
    - but gets tedious quickly!
  - best practice
    - automate series of complex steps by wrapping into a ***pipeline script***
    - e.g. **bash**, **perl** or **python** script
- For UT folks with TACC accounts
  - I have a set of TACC-aware alignment pipeline scripts
    - plus a set of pre-built reference indexes

# Final thoughts

---

- Good judgement comes from experience  
*unfortunately...*
- Experience comes from bad judgement!
- So go get started making  
your 1<sup>st</sup> 1,000 mistakes.....

