# xGen™ cfDNA & FFPE DNA Library Prep Kit

Processing sequence data with unique molecular identifiers (UMIs)

## REVISION HISTORY

| Version | Release date | Description of change |
|---|---|---|
| 3 | November 2022 | Updated guidelines |
| 2 | March 2022 | Updated product kit name |
| 1 | October 2018 | Initial release |

# Table of contents

# INTRODUCTION

This guideline outlines example workflows for processing next generation sequencing (NGS) research data from the  xGen cfDNA & FFPE DNA Library Prep Kit, starting from FASTQ files through making variant calls. The xGen cfDNA & FFPE DNA Library Prep Kit is designed specifically for 1–250 ng of degraded samples, such as cell-free DNA (cfDNA) or DNA extracted from formalin-fixed paraffin-embedded (FFPE) samples. The method features a proprietary single-stranded ligation strategy that maximizes conversion, suppresses adapter-dimer formation, and reduces chimera rates (Figure 1). Since dimer formation is negligible, a fixed concentration of adapter can be used, and aggressive size selection is no longer required post-ligation. Altogether, this strategy delivers higher conversion and library complexity than conventional TA ligation–based methods, enabling highly sensitive detection of low-frequency variants. In addition, fixed single-stranded unique molecular identifier (UMI) sequences are added to the insert during Ligation 1 (shown in dark blue and light green in Figure 1). This unique single-stranded ligation to fixed UMIs enables strand-specific molecular indexing by independently tagging the top and bottom strands. After conversion to fully double-stranded products, libraries are amplified by PCR (Figure 1). Because the UMI sequences are fixed, even if there are sequencing or PCR errors in the UMI, it is possible to identify and correct these errors. This prevents artificial inflation of library complexity due to errors in the UMI sequence. Depending on your sample type or experimental goals, you can choose to use the UMIs in different ways or to ignore them altogether. This guide walks you through the rationale and methods for the various approaches. Figure 2 provides an overview of the analysis method.

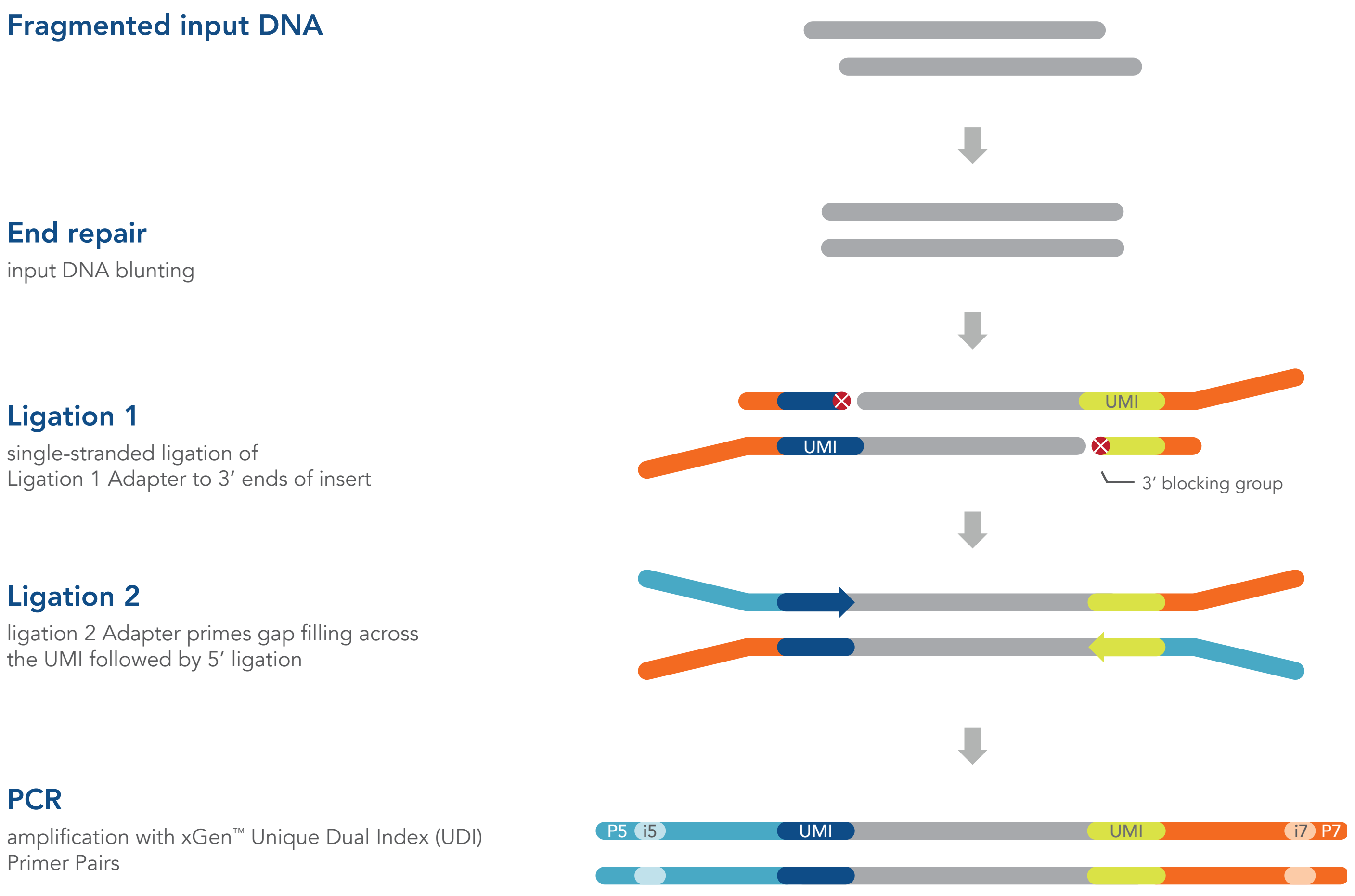
Fragmented input DNA
End repair
input DNA blunting
Ligation 1
single-stranded ligation of Ligation 1 Adapter to 3′ ends of insert
UMI
UMI
3′ blocking group
Ligation 2
ligation 2 Adapter primes gap filling across the UMI followed by 5′ ligation
PCR
amplification with xGen™ Unique Dual Index (UDI) Primer Pairs
P5
i5
UMI
UMI
i7
P7

**Figure 1. Overview of the xGen cfDNA & FFPE DNA Library Prep Kit process.**

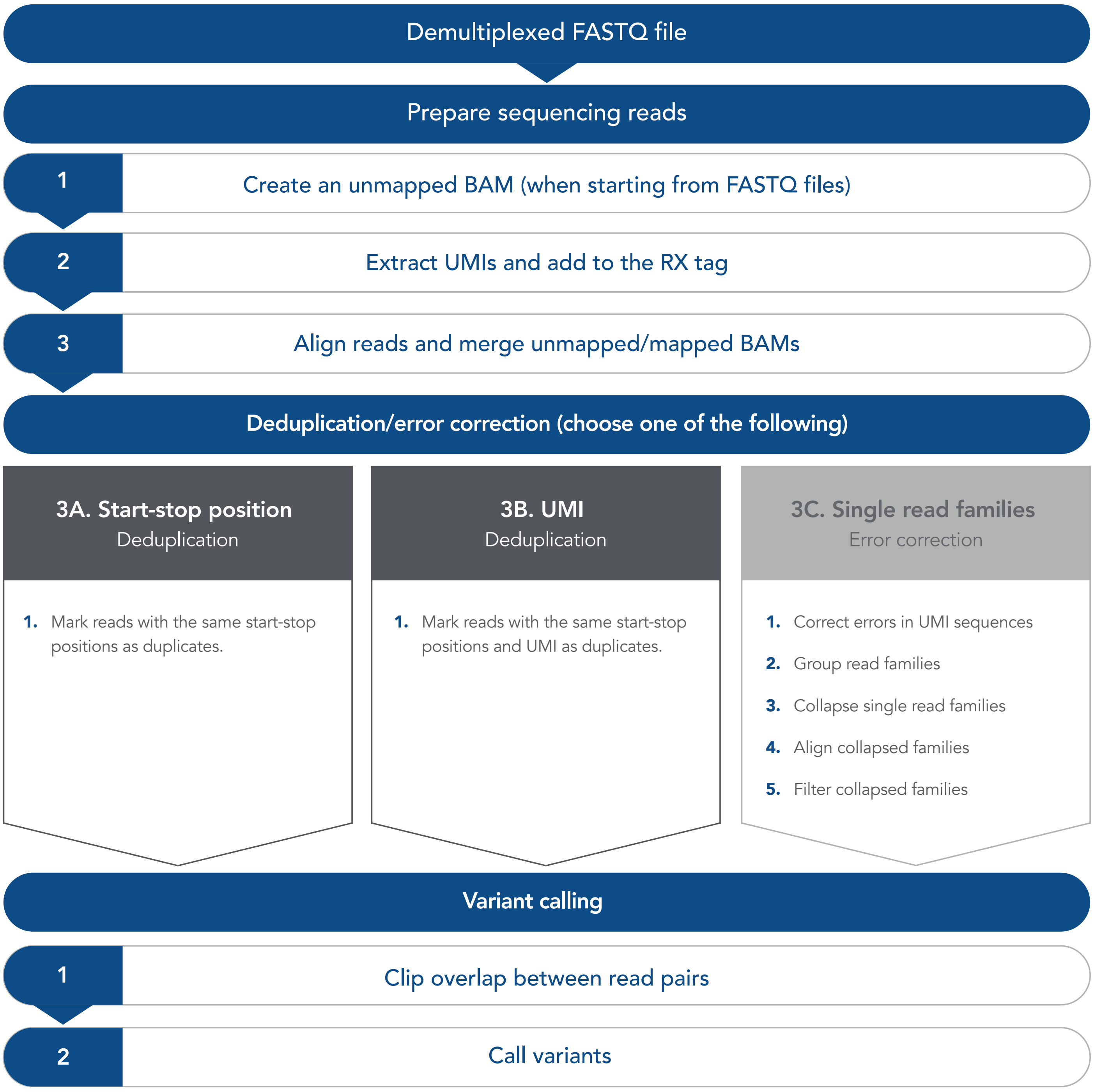
Demultiplexed FASTQ file
Prepare sequencing reads
1
Create an unmapped BAM (when starting from FASTQ files)
2
Extract UMIs and add to the RX tag
3
Align reads and merge unmapped/mapped BAMs
Deduplication/error correction (choose one of the following)
3A. Start-stop position
Deduplication
1. Mark reads with the same start-stop positions as duplicates.
3B. UMI
Deduplication
1. Mark reads with the same start-stop positions and UMI as duplicates.
3C. Single read families
Error correction
1. Correct errors in UMI sequences
2. Group read families
3. Collapse single read families
4. Align collapsed families
5. Filter collapsed families
Variant calling
1
Clip overlap between read pairs
2
Call variants

**Figure 2. Outline of deduplication and error correction strategies compatible with libraries generated with the xGen cfDNA & FFPE DNA Library Prep Kit.**

The most common method for removing PCR duplicates before variant calling is based on retaining only the highest quality read of all reads with the same start-stop coordinates (Figure 3A). Tools like MarkDuplicates (Picard) use this approach. However, sometimes this approach can remove reads that originated from different original molecules. To minimize the removal of reads originating from different molecules, both the start-stop position and UMI sequence can be used to deduplicate reads (Figure 3B). This can be accomplished by choosing the highest quality read from reads that share the same start-stop coordinates and UMI, as is done by tools like MarkDuplicates with the BARCODE_TAG option enabled. This deduplication method chooses the read with the highest read quality, which may or may not contain PCR errors or low-frequency variants.

On the other hand, UMIs can also be used to correct errors in sequencing data at the same time as removing duplicate reads. For example, all reads with the same start-stop position and UMI can be grouped as a single read family, then collapsed (Figure 3C). Rather than simply choosing the highest quality read, this method uses all reads within the single read family to choose the most likely base at each position from beginning to end. This process yields a collapsed single read family that can be used for variant calling. This approach is taken by the tools *GroupReadsByUmi* plus *CallMolecularConsensusReads* (fgbio).

**Note:** Using UMIs for error correction analysis usually requires significantly deeper sequencing and may not be appropriate for damaged samples like low-quality FFPE samples.

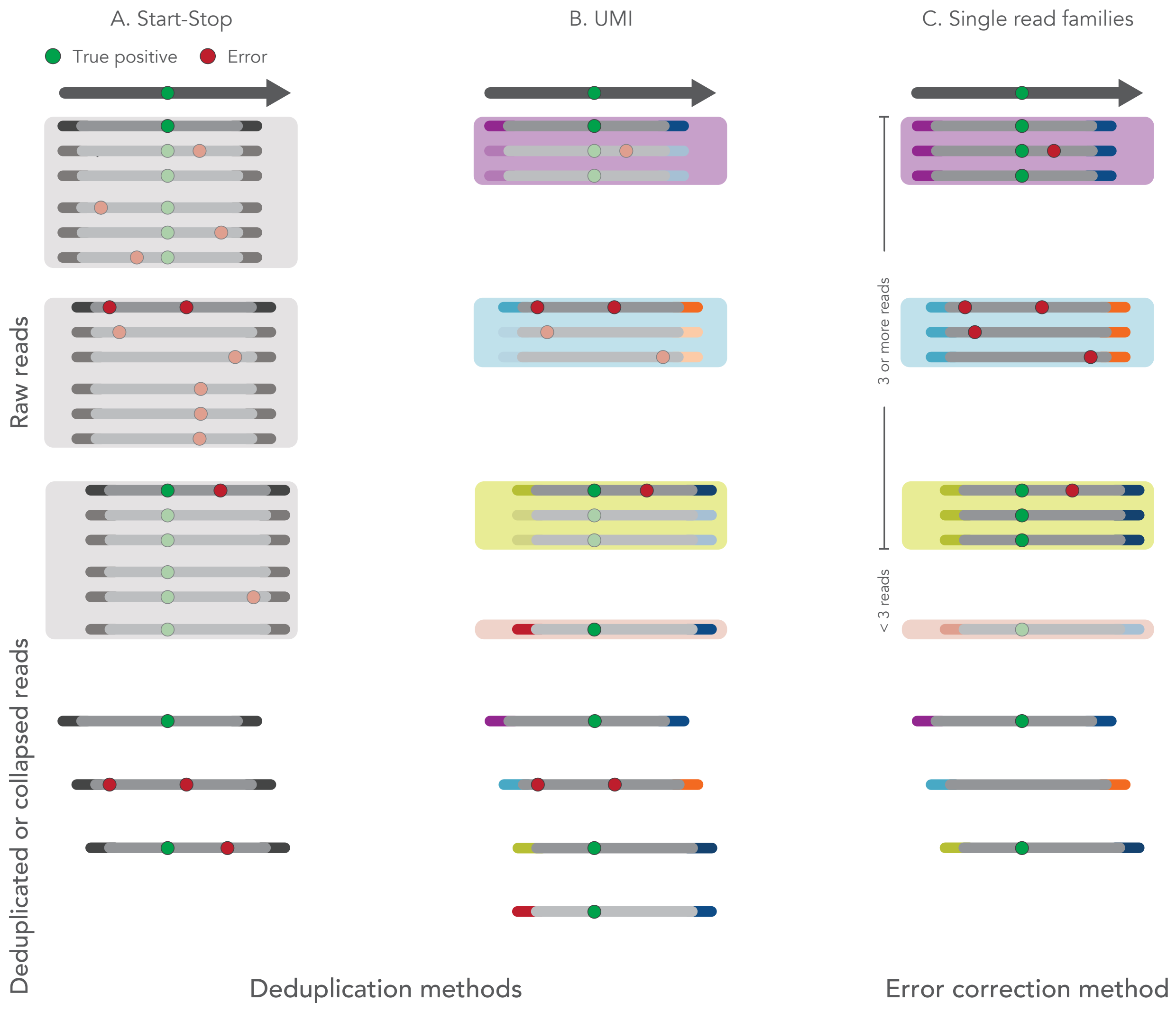

**Figure 3. Deduplication and error correction strategies enabled by the xGen cfDNA & FFPE DNA Library Prep Kit.** The green dots represent a true positive mutation, while red dots represent errors arising from artifacts generated in library preparation or sequencing. Colored segments at the end of each read represent the inline UMIs added when using the xGen cfDNA & FFPE DNA Library Prep Kit. Colored boxes indicate read families identified by their start-stop position and/or UMIs.

The most appropriate method for deduplication or error correction depends on the experimental goal and sample type. One important consideration is the source and quality of your input material. For example, DNA extracted from FFPE samples often has a variety of damage, including nicks, abasic sites, base substitutions, and thymine dimers. DNA damage can result in asymmetric amplification, where the less damaged strand is preferentially amplified. As a result, it can be challenging to use combined read families because sometimes one of the strands was too damaged to be amplified and sequenced. It may often be preferable to use single read families for error correction to minimize the influence of asymmetric amplification on variant calling.

Other critical metrics are sequencing depth, and the closely linked variables of duplicate rate and panel size (target space). To effectively use UMIs to correct errors, PCR duplicates of the same original molecule need to be sequenced many times. As a result, to effectively use these error correction strategies, libraries must be sequenced deeply enough to reach high duplicate rates. For whole genome sequencing (WGS) or capture panels with large target spaces like whole exome sequencing (WES), it is not likely feasible for sequencing depth to reach a duplicate rate high enough to correct sequencing errors.

The examples in Table 1 can act as a starting point to choose the best analysis method for your experiment. Keep in mind these may not always produce the desired result because of variability in sample quality, sequencing runs, and analysis pipelines. The raw sequencing depth and duplicate rate recommendations are standard outputs from *CollectHsMetrics* (Picard) on start-stop duplicates marked data. In addition to the examples in Table 1, duplicate rate can be a helpful indicator when determining if sequencing data is useful for a particular strategy. As a general rule data with duplicate rates >70% are likely appropriate for error correction within single read families, and >40% for deduplication by UMI. We recommend trying different analysis methods to find the one that best suits your experimental goals.

## Table 1. Example deduplication and error correction strategies for variant calling applications.

| Sample type | Panel size | Variant allele frequency | Suggested analysis mode | Raw coverage | Duplicate rate* |
|---|---|---|---|---|---|
| Genomic DNA | Small (<250 kb) | 0.25% | Single read families | 7500–10000X | >80% |
| | Large (WGS, exome) | 50% | Raw reads | 100X | >5% |
| FFPE or degraded samples | Medium (<2500 kb) or Small (<250 kb) | 1% | Single read families | 5000–7500X | >70% |

* Duplicate rate is a standard output from *CollectHsMetrics* on start-stop duplicates marked data. Following these examples may not always produce the desired results because sample quality, sequencing runs, and analysis pipelines may vary.

# REFERENCED SOFTWARE PACKAGES

The following software packages are used in the examples within this document:

| Package | Version | License | URL |
|---|---|---|---|
| Picard | 2.18.9 | MIT | https://github.com/broadinstitute/picard |
| bwa | 0.7.15-r1140 | GPL3 | https://github.com/lh3/bwa |
| fgbio | 0.7.0 | MIT | https://github.com/fulcrumgenomics/fgbio |
| VarDict Java | 1.5.8 | MIT | https://github.com/AstraZeneca-NGS/VarDictJava |

**Note:** Input of parameters for every tool within each package will affect analysis results. We recommend that you start running individual tools with the **help** option to view a full list of parameters which you can tune.

# PERFORM ANALYSIS

## Prepare sequencing reads

1. Construct an unmapped BAM

   Using a demultiplexed FASTQ file, generate an unmapped BAM with *FastqToSam* (Picard).

   Example invocation:

```
java -jar picard.jar FastqToSam \
  FASTQ=unmapped_R1.fastq.gz \
  FASTQ2=unmapped_R2.fastq.gz \
  O=unmapped.bam \
   SM=sample
```

**Tip:** Demultiplexing from Illumina BCL can also generate unmapped BAMs instead of FASTQ files using *IlluminaBasecallsToSam* (Picard). This avoids the generation of intermediate FASTQ files.

2. Extract UMIs and add to the RX tag

*ExtractUmisFromBam* (fgbio) processes unmapped BAM files, extracting in-line UMIs contained within read 1 or read 2 sequences and adds them into the RX tag.

**Note:** Read 1 and read 2 of the same pair should have the same value stored in the RX tag, regardless of which read contains the UMI sequence. The read structure in the *ExtractUmisFromBam* command represents the positions of the UMIs within each read (see the supporting documentation for details).

Example invocation:

```
java -jar fgbio.jar ExtractUmisFromBam \
  --input=unmapped.bam --output=unmapped.withUMI.bam \
   --read-structure=8M143T 8M143T --molecular-index-tags=ZA ZB --single-tag=RX
```

In this invocation, both reads are 151 bases long with the first "8M143T" representing the structure of the first read, and the second "8M143T" representing the structure of the second read:

- "8M" represents 8 UMI bases
- "143T" represents 143 bases in the read

3. Align reads

   When you have an unmapped BAM file with RX tags, use a combination of an aligner and *MergeBamAlignment* (Picard) to generate a mapped BAM that includes all necessary metadata including the UMIs. In this example, we use BWA to align the reads to hg38, but other aligners and genome builds can be used, as appropriate.

   Example invocation:

```
 java -jar picard.jar SamToFastq I=unmapped.withUMI.bam F=/dev/stdout \
INTERLEAVE=true \
   | bwa mem -p -t 4 hg38.fa /dev/stdin \
   | java -jar picard.jar MergeBamAlignment \
     UNMAPPED=unmapped.withUMI.bam ALIGNED=/dev/stdin O=mapped.bam R=hg38.fa \
     SO=coordinate ALIGNER_PROPER_PAIR_FLAGS=true MAX_GAPS=-1 \
      ORIENTATIONS=FR VALIDATION_STRINGENCY=SILENT CREATE_INDEX=true
```

**Note:** *MergeBamAlignment* requires the unmapped BAM file to be in "queryname" sort order. In this case, the BAM file output by *ExtractUmisFromBam* is already in the correct sort order.

# DEDUPLICATION OR ERROR CORRECTION

Choose the deduplication or error correction strategy appropriate for your experimental design. The examples in Table 1 can help inform this decision. Follow instructions for either A-D, as appropriate, then proceed to variant calling.

## A. Deduplicate by start-stop position

**Mark reads with the same start-stop positions as duplicates** using *MarkDuplicates* (Picard). *MarkDuplicates* will first identify a group of reads as being duplicates based on start-stop position. Then, it will pick a representative unique read and flag the rest of the reads as duplicates (using the bit flag 0x0400). By default, the unique read will have the highest sum-of-base qualities across all the candidate duplicate reads.

Example invocation:

```
java -jar picard.jar MarkDuplicates \
  I=mapped.bam \
  O=markduplicates.bam \
   M=markduplicates.metrics.txt
```

## B. Deduplicate by UMI

**Mark reads with the start-stop positions and UMI as duplicates** using *MarkDuplicates* (Picard). *MarkDuplicates* will first identify a group of reads as being duplicates based on start-stop position and UMI sequence. Then, it will pick a representative unique read and flag the rest of the reads as duplicates (using the bit flag 0x0400). By default, the unique read will have the highest sum-of-base qualities across all the candidate duplicate reads.

Example invocation:

```
java -jar picard.jar MarkDuplicates \
  I=mapped.bam \
  O=markduplicates.bam \
  M=markduplicates.metrics.txt \
   BARCODE_TAG=RX
```

## C. Error correct by collapsing single read families

1. **Correct errors in the UMI sequences** stored in the tags of the BAM file using *CorrectUmis* (fgbio). Correction is controlled by the number of mismatches tolerated between the observed UMI and the fixed UMI "--max-mismatches", and the number of mismatches the next best UMI must have "--min-distance".

   Example invocation:

```
 java -jar fgbio.jar CorrectUmis \
   -i mapped.bam -o mapped.fixedumi.bam \
   --max-mismatches=3 \
   --min-distance=1 \
   -M metrics.txt \
   -r rejected.bam -t RX -u GAGACGAT TTCCAAGG CGCATGAT ACGGAACA CGGCTAAT GCTATCCT TGGACTCT \
ATCCAGAG CTTAGGAC GTGCCATA  TCGCTGTT TTCGTTGG AAGCACTG GTCGAAGA ACCACGAT GATTACCG GCACAACT \
GCGTCATT GAAGGAAG ACTGAGGT  TGAAGACG GTTACGCA AGCGTGTT GATCGAGT TTGCGAAG CTGTTGAC GATGTGTG \
 ACGTTCAG TTGCAGAC CAATGTGG  ACGACTTG ACTAGGAG
```

2. **Group read families** by identifying which reads come from the same source molecule using *GroupReadsByUmi* (fgbio), which assigns a unique source molecule ID to each applicable read, stores the ID in the MI tag, and outputs a BAM file that is sorted by the MI tag and ready for read collapsing. The source molecule is identified using a combination of UMI sequence and mapping positions from reads 1 and 2.

   Example invocation:

```
 java -jar fgbio.jar GroupReadsByUmi \
   --input=mapped.fixedumi.bam --output=grouped.bam \
   --strategy=paired --edits=0 --min-map-q=20
```

   *GroupReadsByUmi* implements several strategies for matching UMIs to account for sequencing error. The paired method implements the directed adjacency graph method introduced by UMI-tools [1]. Parameters are available to control how many errors are allowed when matching UMIs at the same position and for filtering (i.e., ignoring) reads with low mapping quality.

   **Note:** Reads with low mapping quality should be ignored to prevent the generation of multiple single read families from multiple mismapped copies of the same source molecule.

3. **Collapse single read families** by combining each set of reads using *CallMolecularConsensusReads* (fgbio). This step generates collapsed single read families as an unmapped BAM from the output of *GroupReadsByUmi*.

   **Tip:** There are many parameters that affect single read family collapsing; for an up-to-date listing and supporting documentation, run *CallMolecularConsensusReads* with the -h option.

   Example invocation (with recommended parameters):

```
 java -jar fgbio.jar CallMolecularConsensusReads \
   --input=grouped.bam --output=consensus.unmapped.bam \
   --error-rate-pre-umi=45 --error-rate-post-umi=30 \
   --min-consensus-base-quality=2 --min-input-base-quality=10 --min-reads=1
```

   **Note:** This invocation produces collapsed single read families for all molecules that have at least 1 observation.

4. **Align collapsed families**. After you have generated collapsed single read families, you must remap the reads and merge mapped/unmapped BAMs to retain the appropriate metadata in the mapped file. The mapping procedure is the same as for raw reads described previously.

   Example invocation (with recommended parameters):

```
 java -jar picard.jar SamToFastq I=consensus.unmapped.bam \
F=/dev/stdout INTERLEAVE=true \
   | bwa mem -p -t 8 hg38.fa /dev/stdin \
   | java -jar picard.jar MergeBamAlignment \
     UNMAPPED=consensus.unmapped.bam ALIGNED=/dev/stdin \
     O=consensus.mapped.bam R=hg38.fa \
     SO=coordinate ALIGNER_PROPER_PAIR_FLAGS=true MAX_GAPS=-1 \
      ORIENTATIONS=FR VALIDATION_STRINGENCY=SILENT CREATE_INDEX=true
```

   **Note:** Duplicate marking should not be performed on collapsed families because each represents a unique source molecule.

5. **Filter collapsed families** using *FilterConsensusReads* (fgbio). There are 2 kinds of filtering: 1) masking or filtering individual bases in reads and 2) filtering reads (i.e., not writing them to the output file). Base-level masking is only applied if per-base tags are present (see the supporting documentation for *CallMolecularConsensusReads* on tag descriptions). Read-level filtering is always applied.

   **Tip:** When filtering reads, secondary alignments and supplementary records may be removed independently if they fail one or more filters. If either read 1 or read 2 primary alignments fail a filter, all records for the template will be filtered out.

   **Tip:** There are many parameters that affect the filtering of the collapsed families. For an up-to-date listing and supporting documentation, run *FilterConsensusRead* with the -h option.

   Example invocation (with recommended parameters):

```
 java -jar fgbio.jar FilterConsensusReads \
--input=consensus.mapped.bam \
--output=consensus.mapped.filtered.bam \
--ref=hg38.fa \
--min-reads=3 \
--max-read-error-rate=0.05 \
--max-base-error-rate=0.1 \
--min-base-quality=2 \
 --max-no-call-fraction=0.2
```

   **Note:** This script produces a filtered BAM file for collapsed families that have at least 3 supporting reads.

# VARIANT CALLING

## Clip overlap between read pairs

Use *ClipBam* (fgbio) to eliminate overlap between read 1 and read 2 pairs to ensure that downstream processes, specifically variant calling, cannot double count evidence from the same template when both reads span a variant site in the same template. Clipping overlapping reads is only performed on "FR" read pairs and is implemented by clipping approximately half the overlapping bases from each read. By default, hard clipping is performed; soft-clipping may be substituted using the "--soft-clip" parameter. BAMs from deduplication or error correction can be used.

Example invocation:

```
java -jar fgbio.jar ClipBam \
  --input=consensus.mapped.filtered.bam \
  --output=consensus.mapped.filtered.clipped.bam \
  --ref=hg38.fa --soft-clip=false --clip-overlapping-reads=true
```

## Call variants

You can produce variant calls from reads with the variant caller of your choice. The following example shows how to use *VarDictJava* (Astra Zeneca) in tumor-only mode to generate a VCF file, and *SortVcf* (Picard) to sort and index the resulting VCF.

Example invocation (with reommended parameters):

```
Vardict_Dir=VarDict-1.5.8/bin
min_af=0.01
tumor_name=tumor

${Vardict_Dir}/VarDict \
   -G hg38.fa \
   -N ${tumor_name} \
   -f ${min_af} \
   -b consensus.mapped.filtered.clipped.bam \
   -z -c 1 -S 2 -E 3 -g 4 -th 1 --nosv \
   target_regions.bed \
| ${Vardict_Dir}/teststrandbias.R \
| ${Vardict_Dir}/var2vcf_valid.pl -N ${tumor_name} -E -f ${min_af} \
| awk '{if ($1 ~ /^#/) print; else if ($4 !=5) print}' \
 > ${tumor_name}.vcf
```

## REFERENCES

1. Smith T, Heger A, Sudbery I. UMI-tools: Modeling sequencing errors in Unique Molecular Identifiers to improve quantification accuracy. *Genome Res*. 2017; 27(3):491–499.

xGen™ cfDNA & FFPE DNA Library Prep Kit

For more information, go to www.idtdna.com/ContactUs