# xGen™ UDI-UMI Adapters

## Processing sequence data with unique molecular identifiers (UMIs)

# REVISION HISTORY

| Version | Release date | Description of change |
|---|---|---|
| 2 | February 2022 | Product name update to xGen™ UDI-UMI Adapters. |
| 1.1 | February 2018 | Updated title to avoid confusion with a second analysis guideline written for use with new IDT adapters (i.e., xGen Duplex Seq Adapters—Tech Access). |
| 1 | October 2017 | Original protocol. |

# Table of contents

# INTRODUCTION

This document outlines an example workflow for processing next generation sequencing data containing unique molecular identifiers, starting from FASTQ or similar raw data through making variant calls. The high level workflow is:

# GUIDELINES

## A. Construct an unmapped BAM tagged with UMIs

The optimal way to generate a mapped BAM with UMIs depends on the raw data format (e.g., FASTQ, BCL) and existing processing pipelines. Ultimately, you want a BAM file where the UMI sequence is stored in the RX tag for each read. Read 1 and read 2 of the same pair should have the same value stored in the RX tag, regardless which read contains the UMI sequence.

Multiple tools exist to help with this process:

1. Picard's IlluminaBasecallsToSam processes Illumina® instrument run folders containing BCL files and produces demultiplexed, per-sample, unmapped BAM files. The tool extracts UMI sequences from anywhere in the template or index reads into the RX tag.
2. fgbio's AnnotateBamWithUmis adds the UMIs from a FASTQ file to the RX tag of reads in a pre-existing BAM file by matching read names between the files.
3. fgbio's *ExtractUmisFromBam* processes unmapped BAM files, where the UMIs are contained within read 1 or read 2 sequences, and extracts the UMI sequences into the RX tag.

## B. Align reads

When you have an unmapped BAM file with RX tags, use a combination of an aligner and Picard's *MergeBamAlignment* tool to generate a mapped BAM that includes all necessary metadata. An example invocation follows:

```
java -Xmx1g -jar picard.jar SamToFastq I=unmapped.bam F=/dev/stdout
INTERLEAVE=true \
    | bwa mem -p -t 8 hs38DH.fa /dev/stdin \
    | java -Xmx4g -jar picard.jar MergeBamAlignment \
      UNMAPPED=unmapped.bam ALIGNED=/dev/stdin O=mapped.bam R=hs38DH.fa \
      SO=coordinate ALIGNER_PROPER_PAIR_FLAGS=true MAX_GAPS=-1 \
      ORIENTATIONS=FR VALIDATION_STRINGENCY=SILENT CREATE_INDEX=true
```

The next step produces consensus reads, making it unnecessary to duplicate-mark the reads or perform further processing of the raw data.

# C. Map BAM to consensus reads

1. **Identify which reads come from the same source molecule** by using fgbio's GroupReadsByUmi tool, 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 consensus calling. The source molecule is identified using a combination of UMI sequence and mapping positions from reads 1 and 2. An example invocation follows:

```
java -Xmx1g -jar fgbio.jar GroupReadsByUmi \
  --input=mapped.bam --output=grouped.bam \
  --strategy=adjacency --edits=1 --min-map-q=20
```

   GroupReadsByUmi implements several strategies for matching UMIs to account for sequencing error. The adjacency 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. Low mapping quality reads should be ignored to prevent multiple consensus reads from being generated from multiple mismapped copies of the same source molecule.

2. **Combine each set of reads to generate consensus reads** using fgbio's *CallMolecularConsensusReads*. This step generates unmapped consensus reads from the output of GroupReadsByUmi. There are many parameters that affect the consensus calling; for an up-to-date listing and supporting documentation, run *CallMolecularConsensusReads* with the -h option. An example invocation follows with recommended parameters:

```
java -Xmx1g -jar fgbio.jar CallMolecularConsensusReads \
  --input=grouped.bam --output=consensus.unmapped.bam \
  --error-rate-post-umi=30 --min-reads=1
```

   This script produces consensus reads for all molecules that have at least one observation.

3. **Filter consensus reads** generated by CallMolecularConsensusReads using fgbio's FilterConsensusReads. There are two 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/filtering is only applied if per-base tags are present (see the documentation for CallMolecularConsensusReads for tag descriptions). Read-level filtering is always applied.

   When filtering reads, secondary alignments and supplementary records may be removed independently if they fail one or more filters. If either R1or R2 primary alignments fail a filter, then all records for the template will be filtered out. There are many parameters that affect the filtering of the consensus reads. For an up-to-date listing and supporting documentation, run FilterConsensusRead with the -h option. FilterConsensusRead can be applied to either mapped or unmapped BAM files. An example invocation with recommended parameters follows:

```
java -Xmx1g -jar fgbio.jar FilterConsensusReads \
   --input=consensus.unmapped.bam \
   --output=consensus.filtered.unmapped.bam \
   --ref=hs38DH.fa \
   --reverse-per-base-tags=true \
   --min-reads=3 \
   -E 0.05 \
   -N 40 \
   -e 0.1 \
   -n 0.1
```

   This script produces a filtered, consensus BAM file containing sequences from molecules that were observed at least three times: *min-reads=3*.

## D. Produce variant calls from consensus reads

Now that you have generated filtered, consensus reads, you must re-map the reads and call variants. The mapping procedure is the same as for raw reads:

```
java -Xmx1g -jar picard.jar SamToFastq I=consensus.unmapped.bam \
  F=/dev/stdout INTERLEAVE=true \
    | bwa mem -p -t 8 hs38DH.fa /dev/stdin \
    | java -Xmx4g -jar picard.jar MergeBamAlignment \
        UNMAPPED=consensus.unmapped.bam ALIGNED=/dev/stdin \
        O=consensus.mapped.bam R=hs38DH.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 consensus reads, because each read represents a unique source molecule.

Variant calling can be accomplished with the variant caller of your choice. The following example shows how to use *VarDictJava* in tumor-only mode to generate a VCF file, and Picard's SortVcf to sort and index the resulting VCF.

```
var_dict_dir=VarDictJava
min_af=0.01
tumor_name=tumor

$var_dict_dir/build/install/VarDict/bin/VarDict \
  -G hs38DH.fa \
  -N $tumor_name \
  -f $min_af
  -b consensus.mapped.bam \
  -z -c 1 -S 2 -E 3 -g 4 -th 4 \
  target_regions.bed \
    | $var_dict_dir/VarDict/teststrandbias.R \
    | $var_dict_dir/VarDict/var2vcf_valid.pl -N $tumor_name -E -f $min_af \
    | awk '{if ($1 ~ /^#/) print; else if ($4 != $5) print}' \
    > tmp.vcf
java -Xmx1g -jar picard.jar SortVcf I=tmp.vcf O=${tumor_name}.vcf
SD=hs38DH.dict && rm tmp.vcf
```

# REFERENCES

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

# xGen™ UDI-UMI Adapters

Technical support: applicationsupport@idtdna.com


Doc ID: RUO22-0723_001 04/22