Extracting UMI Sequences from Libraries Generated with NEXTFLEX® UDI-UMI Barcodes

Introduction

This guide provides users with a method to extract unique molecular identifier sequences (UMIs) for use in downstream analysis pipelines. Alternative methods and programs exist which can accomplish the same task and may be more suitable for specific purposes.

The example in this guide utilized a 300-cycle kit on the Illumina® MiSeq™ system with paired-end reads and cycles allotted as follows:

Read 1 = 150 cycles, Index 1 = 17 cycles, Index 2 = 8 cycles, and read 2 = 146 cycles. The reduced cycles for read 2 was performed to ensure there were enough reagents within the sequencing kit to account for the nine base pair UMI. This alteration may not be required, depending on the type of kit used and the sequencing platform chosen. Consult the manufacturer for reagent overage provided for index reads and adjust as necessary to accommodate 25 cycles for index reads (see figure 1).

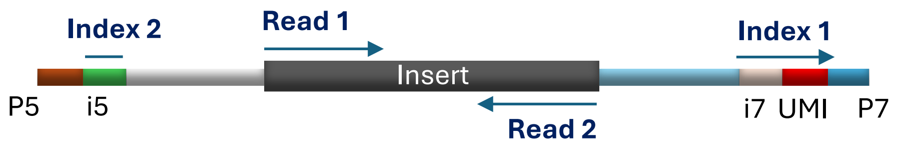

Figure 1. Structure of NEXTFLEX® UDI-UMI Barcodes.

The i5 and i7 barcodes have a length of eight base pairs each, while the UMI is nine base pairs. On Illumina® sequencing platforms, index 1 refers to the length of i7+UMI and index 2 corresponds to the length of i5. Therefore, it is necessary to account for a total of 25 cycles for index reads when setting the run.

Modifications to this example may be required for single end reads, other Illumina® sequencers, or alternative sequencing platform, such as Element AVITI™. Please contact https://www.revvity.com/contact-us/technical-support and select Next Gen Sequencing under the category section, if you have any additional questions.

Generate backup files

It is recommended to create backups of the following files, prior to making modifications: *RunInfo.xml* and *SampleSheet.csv*.

```
cp RunInfo.xml Original_RunInfo.xml
cp SampleSheet.csv Original_SampleSheet.csv
```

The *RunInfo.xml* and *SampleSheet.csv* will be modified, while the *Original_RunInfo.xml* and *Original_SampleSheet.csv* will serve as backups.

Fastq demultiplexing

1. Start by editing the RunInfo.xml file using any text editor. There are four lines relating to how the cycles were allocated for each read.

```
<Reads>
 <Read NumCycles="151" Number="1" IsIndexedRead="N" />
 <Read NumCycles="17" Number="2" IsIndexedRead="Y" />
 <Read NumCycles="8" Number="3" IsIndexedRead="Y" />
 <Read NumCycles="146" Number="4" IsIndexedRead="N" />
</Reads>
```

The highlighted line above, corresponding to i7+UMI, will be split into two separate lines:

a) A line with 8 cycles and number set to "2". (denoted with a "Y" to indicate this is an index read). This represents the i7 barcode sequence.
b) A second line with 9 cycles, number set to "5", and denoted with a "N". This line corresponds to the UMI sequence, and a separate read file will be generated during the bcl2fastq process.

```
<Reads>
 <Read NumCycles="151" Number="1" IsIndexedRead="N" />
 <Read NumCycles="8" Number="2" IsIndexedRead="Y" />
 <Read NumCycles="9" Number="5" IsIndexedRead="N" />
 <Read NumCycles="8" Number="3" IsIndexedRead="Y" />
 <Read NumCycles="146" Number="4" IsIndexedRead="N" />
</Reads>
```

2. Modify the *SampleSheet.csv* file to remove the highlighted nine "N"s needed for the run to progress and produce data for the 17 cycles required for the index read. An example sample line and header from the original SampleSheet.csv is shown below:

```
Sample_ID,Sample_Plate,Sample_Well,Index_Plate_Well,I7_Index_ID,index,I5_Index_ID,index2,Sample_Project,Description
MHGD_J1,NEWPLATE REVVITY,A01,A01,UDP0001,GATCAACANNNNNNNNN,UDP0001,CTATGTTA,,
```

The nine "N" have been removed from all samples to produce the following altered sample line and header:

```
Sample_ID,Sample_Plate,Sample_Well,Index_Plate_Well,I7_Index_ID,index,I5_Index_ID,index2,Sample_Project,Description
MHGD_J1,NEWPLATE REVVITY,A01,A01,UDP0001,GATCAACA,UDP0001,CTATGTTA,,
```

3. Run bcl2fastq to demultiplex the run, which will produce five fastq files. Three of the files will be listed as reads (R1, R2, and R3) while two files correspond to the index (I1 and I2).

```
bcl2fastq --create-fastq-for-index-reads --mask-short-adapter-reads 0
```

4. Importantly, the read file corresponding to the UMI will be listed as R2, while the read 2 information is stored in R3. This can be changed using the following commands:

```
#Navigate into the folder with the fastq.gz files.
cd Data/Intensities/BaseCalls/
```

```
#Change all files with 'R2' in name to include 'UMI' instead.
for f in *_R2_001.fastq.gz; do mv "$f" $(echo "$f" | sed
's/_R2_001.fastq.gz/_UMI_001.fastq.gz/g'); done
```

```
#Change all files with 'R3' in name to include 'R2' instead.
for f in *_R3_001.fastq.gz; do mv "$f" $(echo "$f" | sed
's/_R3_001.fastq.gz/_R2_001.fastq.gz/g'); done
```

The files are now properly demultiplexed, and the UMI sequences are stored in a standard fastq file format for downstream applications. It is recommended to use the fastq file containing the UMI information to generate quality metrics and broad overview of sequences and counts of identified UMIs.

Convert fastq files for UMI-tools

There are many programs available for UMI manipulation and analysis. We recommend using UMI-tools (https://umi-tools.readthedocs.io/en/latest/) as a general starting point.

For UMI-tools, the UMI sequence needs to be placed next to the name prior to alignment. This can be achieved through the following code or individual commands.

!NOTE: It is easier to use the following commands within a bash script. However, each line of code can be run sequentially, changing the ${CoreName} variable to the partial name of the file preceding the "_L001_UMI_001.fastq.gz" portion which is being modified.

```
#Extract the name and sequence (UMI) from the UMI file.
zcat ${CoreName}_L001_UMI_001.fastq.gz | awk '{if (NR%4==1 || NR%4==2) print $0}'
- >UMIs_Separate_Line

#Recreate the fastq name with the UMI in the name - for UMI tools
paste - - <UMIs_Separate_Line | tr ' ' '\t' | awk '{print $1"_"$3" "$2}' -
>UMI_In_Fastq_Name

#Correct the name for read 1
sed -e "s/2:N:0/1:N:0/g" UMI_In_Fastq_Name >UMI_In_Fastq_Name_Read_1

#Unzip the files
gunzip ${CoreName}_L001_R1_001.fastq.gz
gunzip ${CoreName}_L001_R2_001.fastq.gz

#Replace the name in Read 1 and 2 files with the UMI names
awk 'NR==FNR {line[NR]=$0; next} {if (FNR%4==1) $0=line[++count]; print}'
UMI_In_Fastq_Name_Read_1 ${CoreName}_L001_R1_001.fastq
>${CoreName}_UMI_Added_L001_R1_001.fastq
awk 'NR==FNR {line[NR]=$0; next} {if (FNR%4==1) $0=line[++count]; print}'
UMI_In_Fastq_Name ${CoreName}_L001_R2_001.fastq
>${CoreName}_UMI_Added_L001_R2_001.fastq

#Compress the files
gzip ${CoreName}_UMI_Added_L001_R1_001.fastq
gzip ${CoreName}_UMI_Added_L001_R2_001.fastq
```

These commands collectively generate two new files corresponding to read 1 and read 2 with the UMI sequences added next to the name.

Here is an example of one read pair from the files:

Read 1 (Before)

```
@M03579:695:000000000-LJ63K:1:1101:17196:1558 1:N:0:CTGATCGT+ATATGCGC
TGACAATATTAGANCATCGAGACAGCAAATTAACAAGGATATTGAGGATCTGAACTCAGCACTGGATCAA
ATGGATGTGATAGATATCTACAGAATTATCCACCTCAAAACAACAGAATAAATGTTCTTCTCATCACCAC
ATGGCACATAC
+
CBCCCFFFFFFFG#AAFGGGGGGHHHGHHHHGHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHGHHHHH
HHGHHHHHHHHHHHHHHHHHHHHHGHHGHHHHHHHHHHHHHHHHHHHHHHHHHHFHHHHGHHHHHHHHHH
HHHHHHHHHGF
```

Read 2 (Before)

```
@M03579:695:000000000-LJ63K:1:1101:17196:1558 3:N:0:CTGATCGT+ATATGCGC
TTTGCATTTGCTGAAGAGTGTTTTACTTCTAATTATGAGATTGATTTTAGTGTATGTGCCATGTGGTGAT
GAGAAGAACATTTATTCTGTTGTTTTGAGGTGGATAATTCTGTAGATATCTATCACATCCATTTGATCCA
GTGCTG
+
BBBA@FFFFFFFGGFGFGFGFGHHHHGHHHHHGHHHHFFGHHHHFHHHHHHHHHHHGGHHGFHGHHGEHH
HHHHHHHHHFHHGHHHHHHHHHHHHFFEHBGFHHHGGHHHHHGHHGHGHHHHHHHHHGHHHHHHGHHHH
GHHGHF
```

Read 1 (After)

```
@M03579:695:000000000-LJ63K:1:1101:17196:1558_CGATCGATT
1:N:0:CTGATCGT+ATATGCGC
TGACAATATTAGANCATCGAGACAGCAAATTAACAAGGATATTGAGGATCTGAACTCAGCACTGGATCAA
ATGGATGTGATAGATATCTACAGAATTATCCACCTCAAAACAACAGAATAAATGTTCTTCTCATCACCAC
ATGGCACATAC
+
CBCCCFFFFFFFG#AAFGGGGGGHHHGHHHHGHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHGHHHHH
HHGHHHHHHHHHHHHHHHHHHHHHGHHGHHHHHHHHHHHHHHHHHHHHHHHHHHFHHHHGHHHHHHHHHH
HHHHHHHHHGF
```

Read 2 (After)

```
@M03579:695:000000000-LJ63K:1:1101:17196:1558_CGATCGATT
2:N:0:CTGATCGT+ATATGCGC
TTTGCATTTGCTGAAGAGTGTTTTACTTCTAATTATGAGATTGATTTTAGTGTATGTGCCATGTGGTGAT
GAGAAGAACATTTATTCTGTTGTTTTGAGGTGGATAATTCTGTAGATATCTATCACATCCATTTGATCCA
GTGCTG
+
BBBA@FFFFFFFGGFGFGFGFGHHHHGHHHHHGHHHHFFGHHHHFHHHHHHHHHHHGGHHGFHGHHGEHH
HHHHHHHHHFHHGHHHHHHHHHHHHFFEHBGFHHHGGHHHHHGHHGHGHHHHHHHHHGHHHHHHGHHHH
GHHGHF
```

After running the commands, there are seven total files corresponding to each sample listed in *SampleSheet.csv*. Note that the UMI (highlighted in yellow above) is listed next to the name and separated by an underscore. Additionally, for the read 2 file, the read information (highlighted in blue) has been corrected. The read 1 file with suffix '_UMI_Added_L001_R1_001.fastq.gz' and read 2 file with suffix '_UMI_Added_L001_R2_001.fastq.gz' are now compatible with UMI-tools.

One common method is to align these reads to the relevant reference genome to generate bam files which can be deduplicated using UMI-tools dedup command (https://umi-tools.readthedocs.io/en/latest/reference/dedup.html).

Please contact https://www.revvity.com/contact-us/technical-support and select Next Gen Sequencing under the category section, if you have any additional questions.