TWIST BIOSCIENCE

NGS TARGET ENRICHMENT SOLUTIONS

TWIST 96-PLEX LIBRARY PREPARATION KIT:

SAMPLE DEMULTIPLEXING GUIDE

FOR RESEARCH USE ONLY. NOT INTENDED FOR USE IN DIAGNOSTIC PROCEDURES.

TABLE OF CONTENTS

Introduction 2

Referenced Software Packages 3

Standard workflow: 96-Plex Only Sequencing Run 4

Alternative workflow: Mixing 96-Plex Libraries with Dual-indexed Libraries on a Sequencing Run 7

Post-Demultiplexing Considerations 8

Troubleshooting 9

Appendix 11

INTRODUCTION

Illumina library prep uses barcodes on one or both sides of the library molecule to allow multiple samples to be barcoded separately and then sequenced together in a single run. Demultiplexing is the process of using those barcodes to split a mixed set of reads into multiple files, one for each sample.

The standard Illumina-style placement of barcodes is outside of the library insert sequence. Illumina therefore uses dedicated indexing reads, termed i5 and i7, to determine the sequence of the barcodes (see image below for an example of dual-indexed sequencing on a Nextseq instrument).

Illumina sequencers generate binary basecalling (.BCL) files as part of their output. The typical workflow is to use the available Illumina software to convert BCL files into FASTQ files, while at the same time demultiplexing samples using the i5 and i7 reads, all in one step. This generates one FASTQ file per sample.

Demultiplexing libraries prepared with the Twist 96-Plex Library Preparation kit is slightly different. 96-Plex libraries have a standard Illumina-style i7 barcode that determines which *plate* a sample belongs to (see figure below). The second 96-Plex index, which determines which *well* the sample came from, is not in the standard Illumina i5 location. Rather, the well barcode is *inline* as part of read 1 (R1). Current Illumina BCL conversion and sample demultiplexing software (e.g. bcl2fastq2 and BCLconvert) do not support sample barcodes inline in template reads. Therefore, Illumina software sees 96-Plex libraries as single-index libraries that can be demultiplexed into FASTQ files that combine all samples from the same plate together.

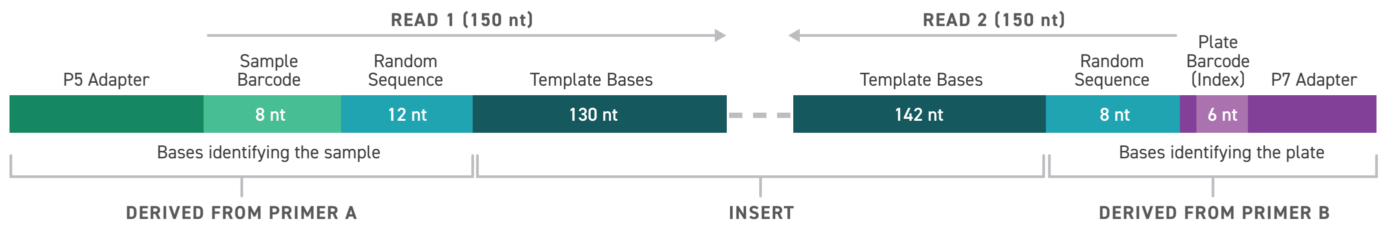
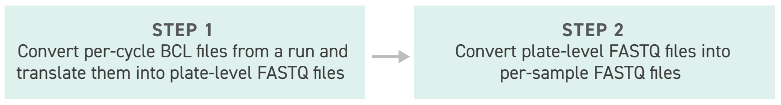


Figure 2. Final library structure.

In order to generate separate FASTQs for each sample, demultiplexing and BCL conversion for libraries prepared with the Twist 96-Plex Library Prep kit requires a two step process. First, the BCLs are converted to FASTQ while demultiplexing using the *plate barcode* using standard Illumina software. Second, the *inline sample barcode* is demultiplex using the fgbio DemuxFastqs open source software to produce per-sample FASTQs:


STEP 1
Convert per-cycle BCL files from a run and translate them into plate-level FASTQ files
STEP 2
Convert plate-level FASTQ files into per-sample FASTQ files

This document outlines an example workflow for processing next generation sequencing data with a plate barcode in the i7 index reads and an inline sample barcode at the start of the first (R1) sequencing read.

Note: Normally, 96-Plex libraries are run by themselves on a flowcell, or together with other compatible single-index libraries. In those cases, the standard workflow described here may be used. If required, it is possible to run 96-Plex libraries together with standard dual-indexed libraries in the same run (Standard workflow). In that case a special workflow must be followed, as described in the "Alternative workflow" section below.

REFERENCED SOFTWARE PACKAGES

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

PACKAGE	VERSION	LICENSE	URL
bclconvert	3.8	?	https://support.illumina.com/sequencing/sequencing_software/bcl-con vert.html
bcl2fastq2	2.20	?	https://support.illumina.com/downloads/bcl2fastq-conversion-software-v2-20.html
fgbio	1.3.0	MIT	https://github.com/fulcrumgenomics/fgbio

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

Twist does not assist with configuring, compiling, executing, or troubleshooting the above packages. Twist does not guarantee the output of any of the above packages is accurate, nor does Twist warranty that any referenced package is fit for your particular use.

STANDARD WORKFLOW: 96-PLEX ONLY SEQUENCING RUN

Note: Adapter trimming should not be performed during demultiplexing. Adapter trimming is typically done by specifying the adapters in the Settings section of the Illumina sample sheet during the BCL conversion step. These settings should be omitted for libraries prepared with the 96-Plex Library Prep kit.

Step 1: Convert per-cycle BCL files to plate-level FASTQ files

Note: This step may be skipped if starting from plate-level FASTQs that contain inline sample barcodes.

If the sequencing run contains only the 96-Plex library, the sequencing run output is equivalent to a single-index sequencing run. This library does not utilize the i5 index read.

The Illumina Sample Sheet is used to specify sample metadata, including the plate barcode(s), to the BCL conversion tool. An example Data Section in the Sample Sheet for 96-Plex only sequencing run for the twelve (12) plate i7 Index read barcodes follows:

```
 [Data]
Sample_Id,index
plate01,ATCACG
plate02,CGATGT
plate03,TTAGGC
plate04,TGACCA
plate05,ACAGTG
plate06,GCCAAT
plate07,CAGATC
plate08,ACTTGA
plate09,GATCAG
plate10,TAGCTT
plate11,GGCTAC
plate12,CTTGTA
```

Notes:

- *The plate barcodes should be updated based on which plate barcodes were actually used (e.g. only plates 4 and 5). Using all plate barcodes will result in outputs for unused plates, which can be subsequently ignored.*
- *The settings section does not need to be specified*

BCL to FASTQ conversion may be performed using either BCLConvert or bcl2fastq. Both options are described below.

Step 1a: Example Invocation using BCLConvert

An example invocation of BCL Convert using a default tolerance of one (1) mismatch in the index read follows:

```
bclconvert \
      --bcl-input-directory /path/to/run-folder \
      --output-directory /path/to/outputs/per-plate \
      --sample-sheet /path/to/SampleSheet.csv \
      --no-lane-splitting
```

The output directory will contain a pair of FASTQs (for R1 and R2) per plate barcode specified in the sample sheet, as well as a pair of FASTQs for any read not matching a plate barcode (named undetermined). An example output directory structure is shown in the Appendix.

Step 1b: Example Invocation using bcl2fastq

The following examples assume a single-index 96-Plex only sequencing run.

An example invocation of BCL Convert using a default tolerance of one (1) mismatch in the index read follows:

```
bcl2fastq \
      --runfolder-dir /path/to/run-folder \
      --output-dir /path/to/outputs/per-plate \
      --sample-sheet /path/to/SampleSheet.csv \
      --no-lane-splitting
```

The output directory will contain a pair of FASTQs (for R1 and R2) per plate barcode specified in the sample sheet, as well as a pair of FASTQs for any read not matching a plate barcode (named undetermined). An example output directory structure is shown in the Appendix.

Step 2: Convert plate-level FASTQ files into per-sample FASTQ files

The FASTQ files for a given plate produced by step 1 above are used as input to produce per-sample FASTQ files for that plate. The demultiplexing must be performed for each plate independently. The fgbio DemuxFastqs tool performs sample demultiplexing and extraction of random sequences.

The software may be installed from source, or with conda in the bioconda channel as follows:

```
conda install -c bioconda fgbio=1.3.0
```

(It is also possible to download the precompiled JAR file (see troubleshooting section below)

The Illumina Sample Sheet is used to specify sample metadata, including the per-sample barcode(s). An example sample sheet is provided at the end of this section.

The Data Section should contain the following information:

- Sample_ID: The sample identifier unique to the sample in the sample sheet.
- Sample_Barcode: the sample-specific read one (R1) inline barcode
- Sample_Name: the sample name
- Well: the well position in the given plate

Alternatively, just the [Data] Section of the Illumina Samplesheet can be used directly without the rest of the sample sheet to specify the sample names and barcodes. Below is an example for the first five (5) sample barcodes as follows:

```
Sample_Id,Sample_Barcode,Well,Sample_Name
S01,CGTACGTA,A01,Alice
S02,AGCTAGCT,A02,Bob
S03,GCTTAACG,A03,Carol
S04,ATCCGGTA,A04,Dave
S05,TAGGCCAT,A05,Eve
```

An example Sample Sheet with all 96 samples is shown in the Appendix.

The following example invocation assumes installation via conda:

```
fgbio -Xmx8G DemuxFastqs \
      --inputs \
      /path/to/outputs/per-plate/plate01_S1_L001_R1_001.fastq.gz \
      /path/to/outputs/per-plate/plate01_S1_L001_R2_001.fastq.gz \
      --metadata SampleSheet.csv \
      --read-structures 8B12S+T 8S+T \
      --output /path/to/outputs/samples/plate01 \
      --metrics /path/to/outputs/samples/plate01/demux_metrics.txt \
      --threads $(nproc) \
      --output-type Fastq
```

Notes:

- *The read structure parameter defines the 96-Plex-specific read structure, as follows (see figure 2):*
  - *8B12S+T: Read 1 structure—8 bases for the Barcode; 12 bases to Skip (priming site); rest of the read is Template*
  - *8S+T: Read 2 structure—8 bases to Skip (priming site); rest of the read is Template*
  - *See the documentation for read structures via the fgbio Read Structures webpage*
- *The output of the above is in FASTQ format and will contain only template bases. fgbio will trim the priming sequences which are included in the reads (12 nt for A reaction primer and 8 nt for B reaction primer) as well as the inline 8 nt barcode.*
- *To output BAM format use `--output-type Bam`. The BAM file will contain the sample barcode per read in the BC tag. Both BAM and FASTQs may be output with `--output-type FastqAndBam`.*
- *The above example invocation uses all available CPUs. Furthermore, it uses 8GB of memory by setting the JVM Heap Space with the -Xmx JVM command line option.*

An example output directory structure is found in the Appendix.

The output metrics file will contain per-sample demultiplexing statistics; see the documentation for fgbio's SampleBarcodeMetric.

The full set of supported command line options can be found in the documentation for fgbio's DemuxFastqs.

The sample sheet provided to fgbio's DemuxFastqs can be validated online via the fgbio DemuxFastqs sample sheet validation webpage. Validating the sample sheet can help avoid cases where an incorrect sample sheet causes a job to fail.

ALTERNATIVE WORKFLOW: MIXING 96-PLEX LIBRARIES WITH DUAL-INDEXED LIBRARIES ON A SEQUENCING RUN

If the sequencing run contains the 96-Plex library mixed with a different dual-index library, the demultiplexing must occur independently for the two libraries: once for the dual-indexed samples, and a second time using a modification of the workflow from step 1 above for the 96-Plex library.

Note: It is especially important that the plate barcodes used in the 96-Plex library are not similar to the first 6bp of the first index read for any of the sample barcodes used for the dual-index library. While rare, this may lead to some reads from the 96-Plex library being assigned to a sample from the dual-index library. This may also cause the sample from the dual-index library with the sample barcode similar to the plate barcode to be improperly assigned to a plate (then later sample) in the 96-Plex library.

The Data Section should contain the following information:
- Sample_ID: The plate identifier unique to the plate in the sample sheet (eg. `PLATE_01`).
- index: the six (6) base-pair plate barcode; the plate barcode is contained in the i7 Index read

The specification of which cycles to consider (i.e. use for plate demultiplexing) must be specified to both the BCL Convert and bcl2fastq2 software. This specification is specified differently for the two software.

For BCL Convert, OverrideCycles must be specified in the Settings Section of the Sample Sheet as follows:

```
[Settings]
OverrideCycles,Y*;I6N*;N*;Y*
```

This tells to BCL Convert to:
- Y*: use all cycles for read one (R1)
- I6N*: use the first six (6) cycles of the first index read (i7) for index bases (the plate barcode), skipping the remaining cycles.
- N*: skip all the cycles of the second index read (i5)
- Y*: use all cycles for read two (R2)

For bcl2fastq2, the "--use-bases-mask" command line option must be used to specify which cycles to consider, with the same value as above: Y*;I6*;N*;Y*.

POST-DEMULTIPLEXING CONSIDERATIONS

The demultiplexing procedure above does not perform adapter trimming. Nonetheless, if the insert is small, the reads may sequence into the opposite end's sample barcode, randomer, and i5/i7 adapter(s). Therefore, it is recommended to clip or trim the read pairs after alignment so that the end of one read pair does not align past the start of its mate. The *`fgbio ClipBam`* tool may be used with the *`--clip-overlapping-reads`* option to perform this function.

It is recommended to quality control the results both by looking at the metrics output by the demultiplexing steps, as well as running quality control software such as *`FastQC`* or *`Picard`*.

TROUBLESHOOTING

I am having trouble installing fgbio to run the DemuxFastqs tool.

Fgbio may be installed a number of ways:

1. Download the fgbio JAR file from the latest release and run it directly with:

```
java -Xmx8g -jar fgbio.jar DemuxFastqs
```

2. Install the tool with conda from the bioconda channel and run it with:

```
conda install -c bioconda fgbio
fgbio DemuxFastqs
```

3. Build and install the tool from source; see these instructions.

The fgbio DemuxFastqs tool fails with an out of memory exception.

The tool will fail with the following exception:

```
Exception in thread “main” java.lang.OutOfMemoryError: Java heap space
```

In this case, the amount of memory allocated to the tool needs to be increased. This is done with the -Xmx Java Virtual Machine (JVM) option for the heap space. For example, to specify 16 gigabytes of memory:

```
fgbio -Xmx16g DemuxFastqs
```

There are few or no reads in the FASTQs for the plate barcodes.

This may occur if the incorrect plate barcodes were specified in the sample sheet during the step to convert per-cycle BCL files to plate-level FASTQ files. Review the sample sheet to ensure the correct plate barcode(s) were specified. As a troubleshooting option, all plate barcodes may be used during this step to re-infer which plate barcode used.

This may occur if the read structure (the assignment of cycles to bases). in the RunInfo.xml does not match the demultiplexing workflow being used. For example, if the 96-Plex library was mixed with a dual-index library but the standard (single-index) demultiplexing workflow is being used. Alternatively, this may occur if the using the alternative (mixed with a dual-index library) demultiplexing workflow when the sequencing a 96-Plex library alone. The RunInfo.xml should be examined to verify the read structure. For example, for a 2x151 bp run with a single (i7) index:

- 151T: use all cycles for read one (R1)
- 6B: use the first 6 bp of the first index read (i7) for index bases (the plate barcode) skipping the remaining bases
- 151T: use all cycles for read one (R2)

There are few or no reads in the FASTQs for the sample barcodes.

This may occur if the FASTQs were not first demultiplexed by plate barcode. Please ensure that this has been performed, as well as the accurate specification of the plate barcodes in the Sample Sheet.

This may occur if the sample barcodes were specified incorrectly to the fgbio DemuxFastqs tool. Please ensure that the sample barcodes are correct.

Sample barcode not found when demultiplexing the samples.

The fgbio DemuxFastqs tool will search for the column with name `Sample_Barcode` to specify the inline read one (R1) sample barcode for each sample. If that column is not found, the tool will fail with the following error:

```
requirement failed: Sample barcode not found in column ‘Sample_Barcode’ for sample id
‘<sample-name>’.
```

Please make sure that the column with name `Sample_Barcode` is found in the sample metadata. The column used for the sample barcode may be changed with `--column-for-sample-barcode <value>` command line option.

APPENDIX

Example Output Directory Tree for Demultiplexing with BCL Convert for a 96-Plex-only Sequencing Run

```
/path/to/outputs/per-plate
├── Logs
|  ├── Errors.log
|  ├── FastqComplete.txt
|  ├── Info.log
|  └── Warnings.log
├── plate01_S1_R1_001.fastq.gz
├── plate01_S1_R2_001.fastq.gz
├── plate02_S2_R1_001.fastq.gz
├── plate02_S2_R2_001.fastq.gz
├── plate03_S3_R1_001.fastq.gz
├── plate03_S3_R2_001.fastq.gz
├── plate04_S4_R1_001.fastq.gz
├── plate04_S4_R2_001.fastq.gz
├── plate05_S5_R1_001.fastq.gz
├── plate05_S5_R2_001.fastq.gz
├── plate06_S6_R1_001.fastq.gz
├── plate06_S6_R2_001.fastq.gz
├── plate07_S7_R1_001.fastq.gz
├── plate07_S7_R2_001.fastq.gz
├── plate08_S8_R1_001.fastq.gz
├── plate08_S8_R2_001.fastq.gz
├── plate09_S9_R1_001.fastq.gz
├── plate09_S9_R2_001.fastq.gz
├── plate10_S10_R1_001.fastq.gz
├── plate10_S10_R2_001.fastq.gz
├── plate11_S11_R1_001.fastq.gz
├── plate11_S11_R2_001.fastq.gz
├── plate12_S12_R1_001.fastq.gz
├── plate12_S12_R2_001.fastq.gz
├── Reports
|  ├── Adapter_Metrics.csv
|  ├── Demultiplex_Stats.csv
|  ├── fastq_list.csv
|  ├── Index_Hopping_Counts.csv
|  ├── IndexMetricsOut.bin
|  ├── RunInfo.xml
|  ├── SampleSheet.csv
|  └── Top_Unknown_Barcodes.csv
├── Undetermined_S0_R1_001.fastq.gz
└── Undetermined_S0_R2_001.fastq.gz
```

Example Output Directory Tree for Demultiplexing with bcl2fastq for a 96-Plex-only Sequencing Run

```
/path/to/outputs/per-plate/
├── plate01_S1_R1_001.fastq.gz
├── plate01_S1_R2_001.fastq.gz
├── plate02_S2_R1_001.fastq.gz
├── plate02_S2_R2_001.fastq.gz
├── plate03_S3_R1_001.fastq.gz
├── plate03_S3_R2_001.fastq.gz
├── plate04_S4_R1_001.fastq.gz
├── plate04_S4_R2_001.fastq.gz
├── plate05_S5_R1_001.fastq.gz
├── plate05_S5_R2_001.fastq.gz
├── plate06_S6_R1_001.fastq.gz
├── plate06_S6_R2_001.fastq.gz
├── plate07_S7_R1_001.fastq.gz
├── plate07_S7_R2_001.fastq.gz
├── plate08_S8_R1_001.fastq.gz
├── plate08_S8_R2_001.fastq.gz
├── plate09_S9_R1_001.fastq.gz
├── plate09_S9_R2_001.fastq.gz
├── plate10_S10_R1_001.fastq.gz
├── plate10_S10_R2_001.fastq.gz
├── plate11_S11_R1_001.fastq.gz
├── plate11_S11_R2_001.fastq.gz
├── plate12_S12_R1_001.fastq.gz
├── plate12_S12_R2_001.fastq.gz
├── Reports
|   └── html
|       ├── 000000000-J2FW4
|       |   ├── all
|       |   |   └── all
|       |   |       ├── all
|       |   |       |   ├── laneBarcode.html
|       |   |       |   └── lane.html
|       |   |       └── unknown
|       |   └── default
|       |       ├── all
|       |       |   ├── all
|       |       |   |   ├── laneBarcode.html
|       |       |   |   └── lane.html
|       |       |   └── unknown
|       |       ├── plate01
|       |       |   ├── all
|       |       |   |   ├── laneBarcode.html
|       |       |   |   └── lane.html
|       |       |   ├── ATCACG
|       |       |   |   ├── laneBarcode.html
|       |       |   |   └── lane.html
|       |       |   └── unknown
|       |       ├── plate02
|       |       |   ├── all
|       |       |   |   ├── laneBarcode.html
```

Example Output Directory Tree for Demultiplexing with bcl2fastq for a 96-Plex-only Sequencing Run, continued

```
|         |         |    |    └── lane.html
|         |         |    ├── CGATGT
|         |         |    |    ├── laneBarcode.html
|         |         |    |    └── lane.html
|         |         |    └── unknown
|         |         ├── plate03
|         |         |    ├── all
|         |         |    |    ├── laneBarcode.html
|         |         |    |    └── lane.html
|         |         |    ├── TTAGGC
|         |         |    |    ├── laneBarcode.html
|         |         |    |    └── lane.html
|         |         |    └── unknown
|         |         ├── plate04
|         |         |    ├── all
|         |         |    |    ├── laneBarcode.html
|         |         |    |    └── lane.html
|         |         |    ├── TGACCA
|         |         |    |    ├── laneBarcode.html
|         |         |    |    └── lane.html
|         |         |    └── unknown
|         |         ├── plate05
|         |         |    ├── ACAGTG
|         |         |    |    ├── laneBarcode.html
|         |         |    |    └── lane.html
|         |         |    ├── all
|         |         |    |    ├── laneBarcode.html
|         |         |    |    └── lane.html
|         |         |    └── unknown
|         |         ├── plate06
|         |         |    ├── all
|         |         |    |    ├── laneBarcode.html
|         |         |    |    └── lane.html
|         |         |    ├── GCCAAT
|         |         |    |    ├── laneBarcode.html
|         |         |    |    └── lane.html
|         |         |    └── unknown
|         |         ├── plate07
|         |         |    ├── all
|         |         |    |    ├── laneBarcode.html
|         |         |    |    └── lane.html
|         |         |    ├── CAGATC
|         |         |    |    ├── laneBarcode.html
|         |         |    |    └── lane.html
|         |         |    └── unknown
|         |         ├── plate08
|         |         |    ├── ACTTGA
|         |         |    |    ├── laneBarcode.html
|         |         |    |    └── lane.html
|         |         |    ├── all
|         |         |    |    ├── laneBarcode.html
|         |         |    |    └── lane.html
```

Example Output Directory Tree for Demultiplexing with bcl2fastq for a 96-Plex-only Sequencing Run, continued

```
|       |       |   └── unknown
|       |       ├── plate09
|       |       |   ├── all
|       |       |   |   ├── laneBarcode.html
|       |       |   |   └── lane.html
|       |       |   ├── GATCAG
|       |       |   |   ├── laneBarcode.html
|       |       |   |   └── lane.html
|       |       |   └── unknown
|       |       ├── plate10
|       |       |   ├── all
|       |       |   |   ├── laneBarcode.html
|       |       |   |   └── lane.html
|       |       |   ├── TAGCTT
|       |       |   |   ├── laneBarcode.html
|       |       |   |   └── lane.html
|       |       |   └── unknown
|       |       ├── plate11
|       |       |   ├── all
|       |       |   |   ├── laneBarcode.html
|       |       |   |   └── lane.html
|       |       |   ├── GGCTAC
|       |       |   |   ├── laneBarcode.html
|       |       |   |   └── lane.html
|       |       |   └── unknown
|       |       ├── plate12
|       |       |   ├── all
|       |       |   |   ├── laneBarcode.html
|       |       |   |   └── lane.html
|       |       |   ├── CTTGTA
|       |       |   |   ├── laneBarcode.html
|       |       |   |   └── lane.html
|       |       |   └── unknown
|       |       └── Undetermined
|       |           ├── all
|       |           |   ├── laneBarcode.html
|       |           |   └── lane.html
|       |           └── unknown
|       |               ├── laneBarcode.html
|       |               └── lane.html
|       ├── index.html
|       ├── Report.css
|       └── tree.html
├── Stats
|   ├── AdapterTrimming.txt
|   ├── ConversionStats.xml
|   ├── DemultiplexingStats.xml
|   ├── DemuxSummaryF1L1.txt
|   ├── FastqSummaryF1L1.txt
|   └── Stats.json
├── Undetermined_S0_R1_001.fastq.gz
└── Undetermined_S0_R2_001.fastq.gz
```

Example Data Section in an Illumina Sample Sheet for Converting plate-level FASTQ files into per-sample FASTQ files

Note: the Well column is not strictly required

```
[Data]
Sample_Name,Sample_Id,Sample_Barcode,Well
A01,A01,CGTACGTA,A01
A02,A02,AGCTAGCT,A02
A03,A03,GCTTAACG,A03
A04,A04,ATCCGGTA,A04
A05,A05,TAGGCCAT,A05
A06,A06,GCTACGAT,A06
A07,A07,GATCAGCT,A07
A08,A08,AACGTTGC,A08
A09,A09,GACTTCAG,A09
A10,A10,GAGTTCAC,A10
A11,A11,GGATTAGG,A11
A12,A12,TGGTCTAG,A12
B01,B01,AGGAACGT,B01
B02,B02,AGGTAGGT,B02
B03,B03,GTACCAAC,B03
B04,B04,GACATCAC,B04
B05,B05,CTGTACAG,B05
B06,B06,CTAGAGCT,B06
B07,B07,CGTAGCAT,B07
B08,B08,TGCACAAC,B08
B09,B09,ACCAAGGA,B09
B10,B10,GCGCTATA,B10
B11,B11,GAACCATC,B11
B12,B12,CTACCATC,B12
C01,C01,CTAGCTAG,C01
C02,C02,CAGTCAGT,C02
C03,C03,GAAGTCCT,C03
C04,C04,TCGATGGT,C04
C05,C05,TCGAGTTG,C05
C06,C06,ACTGACTG,C06
C07,C07,GTGTTGAC,C07
C08,C08,CTCTACAC,C08
C09,C09,GCCGTTAA,C09
C10,C10,GTGAACAG,C10
C11,C11,ATGCATGC,C11
C12,C12,TCCAAGGT,C12
D01,D01,CTCTCAGT,D01
D02,D02,CAAGCTTG,D02
D03,D03,CCAATACG,D03
D04,D04,CTGTCAGA,D04
D05,D05,AAGGTTCC,D05
D06,D06,GGAAGCAT,D06
D07,D07,AGTGTCTG,D07
D08,D08,GCTATTCC,D08
D09,D09,TGACTGAC,D09
D10,D10,TGCACTAG,D10
D11,D11,GATCCTAG,D11
```

```
D12,D12,CTCATGAG,D12
E01,E01,GTACTGCA,E01
E02,E02,AACCTTGG,E02
E03,E03,AGCTCTAG,E03
E04,E04,TGGTACGT,E04
E05,E05,GCCGTATA,E05
E06,E06,CCATATGG,E06
E07,E07,TGGTCATG,E07
E08,E08,CCTAATCC,E08
E09,E09,TGGTGTAC,E09
E10,E10,AACCGGTT,E10
E11,E11,CAACGTAC,E11
E12,E12,GAGACAGT,E12
F01,F01,CTTCCATG,F01
F02,F02,ATCGTAGC,F02
F03,F03,CGTAGCTA,F03
F04,F04,GATGCATC,F04
F05,F05,ACCTGTTC,F05
F06,F06,CAGTTGAC,F06
F07,F07,ACTGTGAC,F07
F08,F08,CATGCTTC,F08
F09,F09,GTACAGCT,F09
F10,F10,ATCCTAGG,F10
F11,F11,GAAGCTTC,F11
F12,F12,GTGTCACA,F12
G01,G01,TACGAACC,G01
G02,G02,CACTAGAC,G02
G03,G03,CTCTCACA,G03
G04,G04,CTTCCAAC,G04
G05,G05,ACAGTCAC,G05
G06,G06,ACACGTCA,G06
G07,G07,CTTGTCCA,G07
G08,G08,TGACGTGT,G08
G09,G09,GGTTCCAA,G09
G10,G10,CAACCTAG,G10
G11,G11,TAGCTACG,G11
G12,G12,GACATCTG,G12
H01,H01,TACCATGG,H01
H02,H02,GCTTCCTA,H02
H03,H03,TGACCACA,H03
H04,H04,ACACGTGT,H04
H05,H05,GAGTTCTG,H05
H06,H06,ACCAACGT,H06
H07,H07,CGATCGAT,H07
H08,H08,ACCACATG,H08
H09,H09,GAGACACA,H09
H10,H10,AGTGGTCT,H10
H11,H11,GAACTGCT,H11
H12,H12,CACATGTG,H12
```

Example Output Directory Structure from fgbio DemuxFastqs after Converting Plate-level FASTQ files into per-sample FASTQ files

An example file and directory structure follows (TODO: update below):

```
/path/to/outputs/samples/plate01
├── ...
└── ...
```

REVISION	DATE	DESCRIPTION
1.0	Jan 2022	· Original analysis guide