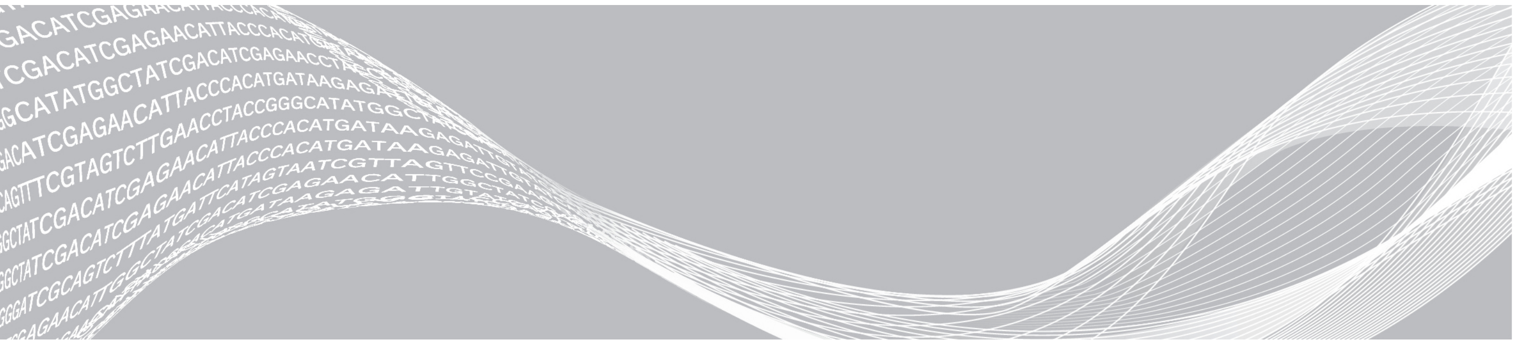
illumina®

Illumina DRAGEN Bio-IT Platform

Getting Started Guide



Document # 1000000076675 v03

February 2020

For Research Use Only. Not for use in diagnostic procedures.

Revision History

Document	Date	Description of Change
Document # 1000000076675 v03	February 2020	• Updated pedigree-based joint genotyping gVCF option. • Added one step and two step population-based joint genotyping gVCF options. • Added somatic CNV calling mode information. • Updated structural variant calling command line options.
Document # 1000000076675 v02	August 2019	• Updated installation instructions. • Updated instructions for running your own test. • Added Quantification and Fusion examples to RNA Map and Align Only Examples. • Updated BCL to FASTQ Conversion with Minimal Settings. • Updated options used in example commands. • Updated troubleshooting commands. • Added License Usage section.
Document # 1000000076675 v01	April 2019	• Updated pedigree file option. • Added two sections on de novo calling.
Document # 1000000076675 v00	January 2019	Initial release.

Table of Contents

Introduction

This Getting Started Guide helps you to start processing data as quickly as possible. Make sure that the Illumina® DRAGEN™ Bio-IT Platform server is turned on and that you are logged in.

Software Installation

If you are already running the latest version of the DRAGEN software and hardware, you can skip ahead to *Run the Self Test* on page 2.

You can query the current version of software and hardware with the following command:

```
dragen_info -b
```

You can query only the software version with the following command:

```
dragen --version
```

To install a new version of software or hardware, first download the package from the DRAGEN Bio-IT Platform support pages on the Illumina website to your DRAGEN server. The preferred installation method is to use the self-extracting .run file, as follows:

```
sudo sh dragen-3.3.7.el7.x86_64.run
```

During installation, if you are prompted to switch to a new hardware version, enter 'y'. It is important that the hardware upgrade process is not interrupted. When it is complete, you must halt and power cycle the server. A reboot command does not update the hardware version. You must use the following halt command to power the server off and on:

```
sudo ipmitool chassis power cycle
```

DRAGEN periodically checks for license renewal by communicating with the license server at lus.edicogenome.com. For servers that are behind a firewall, a proxy can be configured by the network administrator in /etc/environment. For example:

```
http_proxy="http://proxy.customer.com:80/"
https_proxy="https://proxy.customer.com:80/"
ftp_proxy="http://proxy.customer.com:80/"
rsync_proxy="http://proxy.customer.com:80/"
no_
proxy="localhost,127.0.0.1,localaddress,.localdomain.com,.customer.com"
```

License Usage

To check current license usage and expiration date, use the following command:

```
dragen_lic -f genome
```

The license information is output, as follows:

```
LICENSE_MSG| ---- Board #0 (1234565) ----
   LICENSE_MSG| License Genome: used 1000.0/100000 Gbases since
   2019-Jan-01 (1000000000000 bases, 1.0%)
   LICENSE_MSG| Issued=2019-Jan-01, start=2019-Jan-01,
   expiry=2020-Jan-01, period=12 months

   LICENSE_MSG| -- License dongle
   LICENSE_MSG| STATUS : OK
```

```
LICENSE_MSG| DONGLE SN: 0012345678900
LICENSE_MSG| RELEASE : 2016.07p5-19358
LICENSE_MSG| CHIPID : 001234567890EAD
LICENSE_MSG| DNA: active, accelerators=DNA
LICENSE_MSG| issue=2019-Jan-01, start=2019-Jan-01,
expiry=2020-Jan-01
LICENSE_MSG| RNA: active, accelerators=RNA
LICENSE_MSG| issue=2019-Jan-01, start=2019-Jan-01,
expiry=2020-Jan-01
LICENSE_MSG| GZIP: active, accelerators=GZIP
LICENSE_MSG| issue=2019-Jan-01, start=2019-Jan-01,
expiry=2020-Jan-01
LICENSE_MSG| GUNZ: active, accelerators=GUNZ
LICENSE_MSG| issue=2019-Jan-01, start=2019-Jan-01,
expiry=2020-Jan-01
LICENSE_MSG| HMM: active, accelerators=HMM
LICENSE_MSG| issue=2019-Jan-01, start=2019-Jan-01,
expiry=2020-Jan-01
LICENSE_MSG| SMW: active, accelerators=SMW
LICENSE_MSG| issue=2019-Jan-01, start=2019-Jan-01,
expiry=2020-Jan-01
LICENSE_MSG| RANS: active, accelerators=RANS
LICENSE_MSG| issue=2019-Jan-01, start=2019-Jan-01,
expiry=2020-Jan-01
LICENSE_MSG| GRAPH: active, accelerators=GRAPH
LICENSE_MSG| issue=2019-Jan-01, start=2019-Jan-01,
expiry=2020-Jan-01
```

The above license output example is for the Genome license. The first line shows that 1000 gigabases have been consumed. The license installed is for 100000 gigabases and 1% of the gigabases been used. The second line shows the license data and it is the expiry date that is important. The licenses expires either at the expiry date or when 100% of the licensed gigabases are consumed.

Following the license data is the license information that is stored on the dongle or USB key attached to the server. These lines show the status of all the accelerators that are enabled and they are specific to the different pipelines. The accelerators also have an expiry date which should be similar to the license for each pipelines and similar to the genome license example.

To obtain a new license, contact your customer service representative at customerservice@illumina.com. If you encounter problems using your license, contact Illumina Technical Support at techsupport@illumina.com.

Run the Self Test

To run the self test, run the following command:

```
/opt/edico/self_test/self_test.sh
```

This command performs a thorough test of the hardware and takes about 25 minutes. When complete, it should output the following message:

```
SELF TEST RESULT : PASS
```

If a failure occurs, please contact Illumina Technical Support. You can ignore any tests that indicate NON MANDATORY TEST SKIPPED.

Run Your Own Test

In addition to the self-test, you can test with your own data as follows:

- Generate a reference—See *Generate a Reference* on page 3.
- Load a reference—*Load a Reference* on page 4.
- Process your data—See *Example Commands for Processing FASTQ Data* on page 5.

Generate a Reference

If you do not have a reference, you can generate one by using the *dragen –build-hash-table* command and passing in the location of the reference FASTA file. You can specify a set of parameters when building your hash table (see the *DRAGEN Bio-IT Platform User Guide* (1000000070494)).

For testing purposes, you can run the example shell script or the one of commands shown in the examples in this guide. For these examples, the FASTA file is assumed to be located in /staging/human/reference/hg19/hg19.fa. Change the path in the script or command line to the correct directory, if needed. You must have change access to /staging/human/reference and its subdirectories.

Run the example shell script as follows:

```
/opt/edico/examples/build_hash_table.sh
```

Or, run the dragen command as follows:

```
mkdir -p /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149
cd /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
dragen --build-hash-table true --ht-reference
   /staging/human/reference/hg19/hg19.fa \
   --output-dir /staging/human/reference/hg19/hg19.fa.k_21.f_
   16.m_149 \
   --ht-alt-liftover /opt/edico/liftover/hg19_alt_liftover.sam
```

If you generate a hash table without including the *--ht-alt-liftover* option, an error similar to the following may occur (depending on the .fa reference file used):

```
ERROR: Detected hg19 alternate contigs in reference at:
   /staging/hg19fa/hg19.fa
   DRAGEN map quality is significantly improved by building a
   reference with a liftover file to enable ALT aware mapping.
   Use the --ht-alt-lifeover option to specify a liftover file.
   You may ignore this error and continue using your existing
   reference by adding --ht-alt-aware-validate=false to your
   command line. However, DRAGEN map quality will be
   significantly affected.
   Generate the hash table with either the --ht-alt-liftover or
   the –ht alt-aware-validate=false option to avoid the error
   listed above.
```

The *dragen --build-hash-table* command is multithreaded and defaults to eight threads. This command takes approximately 15 minutes to run. You can use the *--ht-num-threads* option with a value up to 32 (depending on the number of threads your server supports) to reduce the run time.

The hash table directory name lists key default option values that were used during the hash table build. Illumina recommends following this best practice when you generate your own hash tables and change the directory name accordingly.

If you enabled the CNV function, generating a hash table takes ~2 hours.

Generate an HG19 Reference

If you do not have a FASTA reference, you can get the hg19 FASTA files from UCSC and concatenate them into a single hg19.fa file as follows:

```
mkdir /staging/hg19fa
cd /staging/hg19fa
wget hgdownload.cse.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz
tar -zxvf chromFa.tar.gz
cat chr*.fa > hg19.fa
```

Generate the DRAGEN hash table reference using the following commands.

```
mkdir /staging/hg19/
dragen --ht-reference /staging/hg19fa/hg19.fa \
    --output-directory /staging/hg19/ --build-hash-table true \
    --ht-alt-liftover /opt/edico/liftover/hg19_alt_liftover.sam
```

Load a Reference

After the binary reference is loaded into memory on the DRAGEN board, it can be used for processing any number of input data sets. You do not need to reload the reference unless you restart the system, or need to use a different reference hash table.

The reference is loaded automatically the first time you process data with it. You can manually load the reference genome onto the board by using the following shell script or command. The reference directory in this example is /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149.

```
   /opt/edico/examples/load_reference.sh
```

OR

```
   dragen -l \
      -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149
```

The dragen command should take less than one minute, and should display the following message:

```
   DRAGEN finished normally
```

If a manual or automatic system reset occurs, then the next time you try to process data, the reference you specify on the command line is automatically reloaded. The reference is also reloaded if you reboot the system.

Example Commands for Processing FASTQ Data

After you have loaded your reference, you can process input FASTQ data. Choose the example that best matches your data sets. These commands can take up to approximately 30 minutes to run on a 24 core server with SSD drives on a 30x coverage whole human genome when running end-to-end (FASTQ input to VCF output). The speed scales with input size, so a 60x coverage genome would take twice as long. Exome data takes a fraction of the time. A successful result is indicated by the following message (an application exit code of 0 when run from a script):

```
DRAGEN finished normally
```

This message is followed by a block of metrics such as read count and performance. If there is a problem with the command line options, an error is displayed, followed by help usage. You may need to scroll up to see the error.

The DRAGEN log can be redirected to a file, to keep the record for future reference.

To get help on dragen command line options, run the following command:

```
dragen -h
```

The example commands shown in this document are formatted for visual display and include line feed characters. To avoid copy-paste errors, each example command is contained in an individual shell script in /opt/edico/examples/. These examples have the following requirements:

- All commands accept either FASTQ or gzipped FASTQ (fastq.gz). DRAGEN automatically determines the file type.
- All commands include the *-f* option, which forces the output file to be overwritten if it exists.
- All commands assume that your DRAGEN reference (hash table) directory is /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149, and your FASTA reference file is /staging/human/reference/hg19/hg19.fa. Replace those with the correct references or directory paths, if needed.
- All command examples assume that the example data package is in /staging/examples (in particular, the .fastq and fastq.gz files are expected to be in /staging/examples/reads).
- To run these example commands, you must have write access to the /staging/examples folder.

End-to-End Aligning and Variant Calling Examples

Paired-End BAM Input, VCF Output

- Enter the following input:

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -b /staging/human/unsorted_SRA056922_30x_e10_50M.bam \
   --enable-map-align true \
   --enable-map-align-output true \
   --enable-variant-caller true \
   --vc-sample-name Unsorted_SRA056922_30x_e10_50M \
   --output-directory /staging/examples/ \
   --output-file-prefix SRA056922_30x_e10_50M \
   --enable-duplicate-marking true
```

- Or, run `/opt/edico/examples/paired_fastq_in_dupmark_bam_and_vcf_out.sh`.

If the */staging/human/unsorted_SRA056922_30x_e10_50M.bam* input file for the example above is missing, run the /opt/edico/examples/paired_fastq_in_unsorted_bam_out.sh script o generate it.

Paired-End FASTQ Input, VCF Output (Default)

- Enter the following input:

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   1.fastq.gz \
   -2 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   2.fastq.gz \
   --enable-variant-caller true \
   --RGID Illumina_RGID \
   --RGSM SRA056922_30x_e10_50M \
   --output-directory /staging/examples/ \
   --output-file-prefix SRA056922_30x_e10_50M \
```

- Or, run `/opt/edico/examples/paired_fastq_in_vcf_out.sh`.

This example shows the minimum options that must be specified to perform an end-to-end run. By default, duplicate-marking is not performed and no BAM output is produced.

Paired-End Fastq Input, Sorted and Duplicate-Marked, VCF Output

- Enter the following input:

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   1.fastq.gz \
   -2 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   2.fastq.gz \
   --enable-variant-caller true \
   --RGID Illumina_RGID \
   --RGSM SRA056922_30x_e10_50M \
   --output-directory /staging/examples/ \
   --output-file-prefix SRA056922_30x_e10_50M \
   --enable-duplicate-marking true
```

- Or, run `/opt/edico/examples/paired_fastq_in_dupmark_vcf_out.sh`.

Paired-End FASTQ Input, Sorted BAM and VCF Output

- Enter the following input:

```
dragen -f
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   1.fastq.gz \
   -2 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
```

```
      2.fastq.gz \
      --enable-variant-caller true \
      --RGID Illumina_RGID \
      --RGSM SRA056922_30x_e10_50M \
      --output-directory /staging/examples/ \
      --output-file-prefix SRA056922_30x_e10_50M \
      --enable-duplicate-marking true \
      --enable-map-align-output true
```

- Or, run `/opt/edico/examples/sorted_bam_in_vcf_out.sh`.

Paired-End FASTQ Input, Sorted SAM and VCF Output

- Enter the following input:

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   1.fastq.gz \
   -2 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   2.fastq.gz \
   --enable-variant-caller true \
   --RGID Illumina_RGID \
   --RGSM SRA056922_30x_e10_50M \
   --output-directory /staging/examples/ \
   --output-file-prefix SRA056922_30x_e10_50M \
   --enable-duplicate-marking true \
   --enable-map-align-output true \
   --output-format SAM
```

- Or, run `/opt/edico/examples/paired_fastq_in_dupmark_sam_and_vcf_ out.sh`.

Paired-End FASTQ Input, Sorted CRAM and VCF Output

- Enter the following input:

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   1.fastq.gz \
   -2 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   2.fastq.gz \
   --enable-variant-caller true \
   --RGID Illumina_RGID \
   --RGSM SRA056922_30x_e10_50M \
   --output-directory /staging/examples/ \
   --output-file-prefix SRA056922_30x_e10_50M \
   --enable-duplicate-marking true \
   --enable-map-align-output true \
   --output-format CRAM \
```

- Or, run `/opt/edico/examples/paired_fastq_in_dupmark_cram_and_vcf_ out.sh`.

Paired-End FASTQ Input, Sorted BAM and VCF Output, plus Repeat Genotyping VCF

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   1.fastq.gz \
   -2 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   2.fastq.gz \
   --enable-variant-caller true \
   --RGID Illumina_RGID \
   --RGSM SRA056922_30x_e10_50M \
   --output-directory /staging/examples/ \
   --output-file-prefix SRA056922_30x_e10_50M \
   --enable-duplicate-marking true \
   --enable-map-align-output true \
   --repeat-genotype-enable true \
   --repeat-genotype-specs /opt/edico/repeat-specs/hg19 \
   --repeat-genotype-sex female \
   --repeat-genotype-ref-fasta
   /staging/human/reference/h19/hg19.fa
```

Alignment Only Examples

All the variations for performing alignment shown in these examples can be used in the end-to-end case as well.

Map/Align Single-Ended FASTQ Input, Sorted BAM output (Default)

- Enter the following input:

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 /staging/examples/reads/SRA056922_30x_rand1_100K.fastq \
   --output-directory /staging/examples/ \
   --output-file-prefix SRA056922_30x_rand1_100K \
   --RGID DRAGEN_RGID \
   --RGSM DRAGEN_RGSM \
```

- Or, run `/opt/edico/examples/single_fastq_in_bam_out.sh`.

Map/Align Single-ended FASTQ input, Sorted, Duplicate-Marked BAM Output

- Enter the following input:

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 /staging/examples/reads/SRA056922_30x_rand1_100K.fastq \
   --output-directory /staging/examples/ \
   --output-file-prefix SRA056922_30x_rand1_100K_dup_marked \
   --enable-duplicate-marking true \
   --RGID DRAGEN_RGID \
   --RGSM DRAGEN_RGSM
```

- Or, run `/opt/edico/examples/single_fastq_in_dupmark_bam_out.sh`.

Map/Align Paired-End FASTQ Input, Sorted BAM Output (Default)

- Enter the following input:

```
dragen -f \
    -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
    -1 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
    1.fastq.gz \
    -2 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
    2.fastq.gz \
    --output-directory /staging/examples/ \
    --output-file-prefix SRA056922_30x_e10_50M \
    --RGID DRAGEN_RGID \
    --RGSM DRAGEN_RGSM
```

- Or, run `/opt/edico/examples/paired_fastq_in_bam_out.sh`.

Map/Align Paired-End FASTQ Input, Sorted CRAM Output

- Enter the following input:

```
dragen -f \
    -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
    -1 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
    1.fastq.gz \
    -2 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
    2.fastq.gz \
    --output-directory /staging/examples/ \
    --output-file-prefix SRA056922_30x_e10_50M \
    --output-format CRAM \
    --RGID DRAGEN_RGID \
    --RGSM DRAGEN_RGSM
```

- Or, run `/opt/edico/examples/paired_fastq_in_cram_out.sh`.

Map/Align Paired-End FASTQ Input, Sorted Uncompressed BAM Output

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   1.fastq.gz \
   -2 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   2.fastq.gz \
   --output-directory /staging/examples/ \
   --output-file-prefix uncompressed_SRA \
   --enable-bam-compression false \
   --RGID DRAGEN_RGID \
   --RGSM DRAGEN_RGSM
```

Map/Align Paired-End FASTQ Input, Sorted SAM Output

- Enter the following input:

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   1.fastq.gz \
   -2 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   2.fastq.gz \
   --output-directory /staging/examples/ \
   --output-file-prefix SRA056922_30x_e10_50M \
   --output-format SAM \
   --RGID DRAGEN_RGID \
   --RGSM DRAGEN_RGSM
```

- Or, run `/opt/edico/examples/paired_fastq_in_sam_out.sh`.

Map/Align Paired -End FASTQ Input, UN-Sorted BAM output

- Enter the following input:

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   1.fastq.gz \
   -2 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   2.fastq.gz \
   --output-directory /staging/examples/ \
   --output-file-prefix unsorted_SRA056922_30x_e10_50M \
   --enable-sort false \
   --RGID DRAGEN_RGID \
   --RGSM DRAGEN_RGSM
```

- Or, run `/opt/edico/examples/paired_fastq_in_unsorted_bam_out.sh`.

Map/Align Interleaved Paired-Ended FASTQ Input, BAM Output

- Enter the following input:

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 /staging/examples/reads/SRA056922_PE_30x_rand1_10K_
   interleaved.fastq \
   --interleaved \
   --output-directory /staging/examples/ \
   --output-file-prefix SRA056922_PE_30x_rand1_10K_interleaved \
   --RGID DRAGEN_RGID \
   --RGSM DRAGEN_RGSM
```

- Or, run `/opt/edico/examples/interleaved_fastq_in_bam_out.sh`.

RNA Map and Align Only Examples

Any of the Map/Align Only examples can be used for RNA. The only difference in the command is to set the *--enable-rna* option to true. DRAGEN automatically picks up the RNA-specific hash tables and uses the RNA spliced aligner in its processing.

The hash table used for these examples must be generated with the *--ht-build-rna-hashtable true* option. Otherwise, the run will fail with an error similar to the following:

```
ERROR: The specified hashtable directory cannot be used to run
   RNA: /staging/examples/reference/hg19/hg19.fa.k_21.f_16.m_149
```

If this error occurs, regenerate the hash table with the *--ht-build-rna-hashtable true* option.

RNA Map/Align Paired-Ended FASTQ Input, BAM Output

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   1.fastq.gz \
   -2 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   2.fastq.gz \
   --output-directory /staging/examples/ \
   --output-file-prefix SRA056922_30x_e10_50M \
   --enable-rna true \
   --RGID DRAGEN_RGID \
   --RGSM DRAGEN_RGSM
```

The following is example command-line to map-align RNA-seq data with additional command line options, including a path to a gtf file with gene annotations. The gene annotations file improves mapping by providing a list of known splice-junctions (rather than discovering them all de novo.)

```
dragen -f \
   -r <HASHTABLE_DIR>
   -1 <FASTQ1> \
   -2 <FASTQ2> \
   -a $/reference_genomes/annotation/GTF/$gencode.annotation.gtf
   --enable-map-align true \
   --enable-sort=true \
   --enable-bam-indexing true \
   --enable-map-align-output true \
   --output-format=BAM \
   --RGID=<READ_GROUP_ID> \
   --RGSM=<Sample_NAME> \
   --RGPL=<LIBRARY> \
   --config-file /opt/edico/config/dragen-user-defaults.cfg \
   --enable-rna=true \
   --output-directory <OUT_DIR> \
   --output-file-prefix <PREFIX>
```

RNA Quantification

To run gene and transcript expression quantification add the following option:

```
--enable-rna-quantification true
```

When *--enable-rna-quantification* is set to true, GC bias correction is the default and *--rna-quantification-gc-bias* does not have to be enabled. In addition, the library type is automatically detected and *--rna-quantification-library-type* does not have to be set.

NOTE
Library-type auto-detection only works for paired-end data. If you have single-end data, you need to specify the library type by setting the *--rna-quantification-library-type* option.

RNA Fusion

To run gene fusion detection add the following option:

```
--enable-rna-gene-fusion true
```

You do not need to specify the library because fusion does not use it.

Epigenome Map and Align Examples

Prior to performing an epigenome (methylation) map and align run with bisulfite sequencing data, you must first create methylation-specific reference hash tables, as follows:

```
mkdir -p /staging/human/reference/hg19_epigenome

dragen --build-hash-table true \
    --ht-reference /staging/human/reference/hg19/hg19.fa \
    --ht-max-seed-freq 64 --ht-seed-len 27 --ht-methylated true \
    --output-directory /staging/human/reference/hg19_epigenome \
    --ht-alt-liftover /opt/edico/liftover/hg19_alt_liftover.sam
```

The above dragen command produces two hash table directories under /staging/human/reference/hg19_epigenome: GA_converted and CT_converted. The CT_converted hash table is produced by converting each C base to T in the reference sequences. Similarly, the GA_converted hash table is produced from the G->A base-converted reference sequences. The base-converted references have less complexity, and to compensate, the hash table seed length argument (*--ht-seed-len*) is typically increased to 27 for mammalian genomes (default seed length is 21).

The *--ht-alt-aware-validate false* option can be used in place of *--ht-alt-liftover*. However, the dragen map quality will be significantly affected due to the presence of alternate contigs in the hg19.fa reference.

Epigenome Map/Align, Directional-protocol, Single-Ended FASTQ Input, BAM Output

The directional (Lister) protocol produces reads from two of the four possible bisulfite sequencing strands. Therefore, when the *--methylation-protocol=directional* option is used, DRAGEN aligns each read or read pair twice with different constraints corresponding to the two possible strands. The following DRAGEN command produces two separate BAM files:

```
mkdir -p /staging/epigenome/directional

dragen -f -r /staging/human/reference/hg19_epigenome \
   -1 /staging/epigenome/reads/sample_1_R1.fastq.gz \
   -2 /staging/epigenome/reads/sample_10_R2.fastq.gz \
   --RGID Illumina_RGID \
   --RGSM sample_10 \
   --RGPL illumina \
   --output-directory /staging/epigenome/directional \
   --output-file-prefix sample_10 \
   --methylation-protocol=directional \
   --enable-sort false
```

Epigenome Map/Align, Nondirectional-protocol, Paired-Ended FASTQ Input, BAM Output

The nondirectional protocol produces reads from all four possible bisulfite sequencing strands. Therefore, when the *--methylation-protocol=non-directional* argument is used, DRAGEN aligns each read four times and produces four separate BAM files.

```
mkdir -p /staging/epigenome/non-directional

dragen -f -r /staging/human/reference/hg19_epigenome \
   -1 /staging/epigenome/reads/sample_10_R1.fastq.gz \
   -2 /staging/epigenome/reads/sample_10_R2.fastq.gz \
   --RGID Illumina_RGID \
   --RGSM sample_10 \
   --RGPL illumina \
   --output-directory /staging/epigenome/non-directional \
   --output-file-prefix sample_10 \
   --methylation-protocol non-directional \
   --enable-sort false
```

Variant Calling Only Examples

The variant calling only examples show how you can pass an existing aligned BAM or CRAM file directly to the DRAGEN Variant Caller. By default, the BAM/CRAM file is passed through the sorting stage prior to variant calling.

To duplicate mark your BAM file before running the DRAGEN Variant Caller, you need to use a separate tool. The DRAGEN Duplicate Marker depends on information provided by the Mapper/Aligner that does not exist in BAM files. To take advantage of the DRAGEN Duplicate Marker, use DRAGEN in end-to-end mode.

The BAM/CRAM files that are used as input to these example commands are not included in the example data set. They are generated by example commands in *Alignment Only Examples* on page 8.

Unsorted BAM Input, VCF Output (Default)

- Enter the following input:

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -b /staging/examples/unsorted_SRA056922_30x_e10_50M.bam \
   --enable-variant-caller true \
   --output-directory /staging/examples/ \
   --output-file-prefix unsorted_output_SRA056922_30x_e10_50M \
   --enable-map-align false
```

- Or, run `/opt/edico/examples/unsorted_bam_in_vcf_out.sh`.

Sorted BAM Input, VCF Output

- Enter the following input:

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -b /staging/examples/SRA056922_30x_e10_50M.bam \
   --enable-variant-caller true \
   --output-directory /staging/examples/ \
   --output-file-prefix sorted_output_SRA056922_30x_e10_50M \
   --enable-map-align false
   --enable-sort false
```

- Or, run `/opt/edico/examples/sorted_bam_in_vcf_out.sh`.

Sorted CRAM Input, VCF Output

- Enter the following input:

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   --enable-variant-caller true \
   --output-directory /staging/examples/ \
   --output-file-prefix sorted_output_SRA056922_30x_e10_50M \
   --enable-sort false \
   --enable-map-align false \
   --cram-input /staging/examples/SRA056922_30x_e10_50M.cram
```

- Or, run `/opt/edico/examples/sorted_cram_in_vcf_out.sh`.

Somatic Small Variant Caller Examples

If you are using the Tumor-Normal BAM input option, and your BAM read groups have a shared RGID, DRAGEN cannot determine which read group the reads belong to. Ideally, you should have different RGIDs for each read group, but you can work around the problem by setting the *--prepend-filename-to-rgid* to true.

Paired-End FASTQ Input

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   --tumor-fastq1 /staging/examples/reads/SRA056922_30x_
   shuffle16k_e10_50M_1.fastq.gz \
   --tumor-fastq2 /staging/examples/reads/SRA056922_30x_
   shuffle16k_e10_50M_2.fastq.gz \
   --enable-variant-caller true \
   --RGID-tumor DRAGEN_RGID \
   --RGSM-tumor DRAGEN_RGSM \
   --output-directory /staging/examples/ \
   --output-file-prefix SRA056922_30x_e10_50M
```

Sorted BAM Input

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   --tumor-bam-input /staging/examples/SRA056922_30x_e10_50M.bam
   \
   --enable-variant-caller true \
   --output-directory /staging/examples/ \
   --output-file-prefix sorted_output_SRA056922_30x_e10_50M \
   --enable-map-align false \
   --prepend-filename-to-rgid true
```

gVCF and Genotyping Examples

Paired-End FASTQ Input, gVCF Output

▶ Enter the following input:

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   1.fastq.gz \
   -2 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   2.fastq.gz \
   --enable-variant-caller true \
   --vc-emit-ref-confidence GVCF \
   --RGID Illumina_RGID \
   --RGSM SRA056922_30x_e10_50M \
   --output-directory /staging/examples/ \
   --output-file-prefix SRA056922_30x_e10_50M
```

- Or, run `/opt/edico/examples/paired_fastq_in_gVCF_out.sh`.

Join Calling with gVCF Input

- Enter the following input:

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   --enable-joint-genotyping true \
   --output-directory /staging/examples/ \
   --output-file-prefix Joint_SRA056922_30x_e10_50M \
   --variant /staging/examples/SRA056922_30x_e10_50M.gvcf.gz
```

- Or, run `/opt/edico/examples/single_gVCF_in_jointVCF_out.sh`.

Pedigree-Based Joint Genotyping with Three gVCF Files and a Pedigree File Input, Joint-Genotyped VCF Output

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   --enable-joint-genotyping true \
   --output-directory /staging/examples/ \
   --output-file-prefix Joint_SRA056922_30x_e10_50M \
   --variant /staging/examples/mother.gvcf.gz \
   --variant /staging/examples/father.gvcf.gz \
   --variant /staging/examples/child.gvcf.gz \
   --pedigree-file <PEGIGREE_FILE>
```

One Step Population-Based Joint Genotyping with gVCF Input, Joint-Genotyped Multisample Output

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   --enable-joint-genotyping true \
   --output-directory /staging/examples/ \
   --output-file-prefix Joint_SRA056922_30x_e10_50M \
   --variant /staging/examples/SRA056922_30x_e10_50M.gvcf.gz
```

Two Step Population-Based Joint Genotyping with gVCF List Input, Joint-Genotyped Multisample VCF Output

The first step generates a multisample VCF as output using a gVCF list as input and the following command line option.

```
dragen -f \
   --enable-gvcf-genotyper true \
   --enable-map-align false \
   --variant-list ${GVCF_LIST} \
   --ht-reference ${FASTA_REF} \
   --intermediate-results-dir ${TEMP_DIR} \
   --output-directory ${OUTPUT_DIR} \
   --output-file-prefix ${COHORT_NAME}
```

The second step generates a joint-genotyped multisample VCF using the multisample VCF produced from step one as input and the following command line option. To specify the multisample VCF, use --*variant*.

```
dragen -f \
   --enable-joint-genotyping true \
   --variant ${MULTISAMPLE_VCF} \
   --ref-dir ${FASTA_REF} \
   --output-directory ${OUTPUT_DIR}\
   --output-file-prefix ${COHORT_NAME}.joint_genotyped
```

Table 1 Joint-calling modes, with associated input files and command line options.

VCF to generate	Population joint-called multisample gVCF	Family joint-called multisample gVCF	Population joint-called multisample VCF	Family joint-called multisample VCF
Input file	Multisample combined gVCF file	Multisample combined gVCF file	Multi-sample combined gVCF file or X individual gVCF files	Multi-sample combined gVCF file or X individual gVCF files
Use pedigree file	No	Yes	No	Yes
Command line option	--enable-joint-genotyping true --enable-multi-sample-gvcf=TRUE	--enable-joint-genotyping true --enable-multi-sample-gvcf=TRUE --pedigree-file file.ped	--enable-joint-genotyping true	--enable-joint-genotyping true --pedigree-file file.ped

Coverage Metrics Report Example

By default, DRAGEN reports read coverage over the whole genome, and if available also for the target bed. You can specify up to three additional regions over which coverages are reported. In the following example, DRAGEN generates two coverage reports, cov_report and full_res, for the additional coverage region 1.

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   1.fastq.gz \
   -2 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   2.fastq.gz \
   --RGID Illumina_RGID \
   --RGSM SRA056922_30x_shuffle16k \
   --output-directory /staging/examples/ \
   --output-file-prefix SRA056922_30x_e10_50M \
   --qc-coverage-region-1 /staging/examples/reads/vc_
   smoke.callable.bed \
   --qc-coverage-reports-1 cov_report full_res
```

The full_res report corresponds to the Bedtools coverage option, and includes a per base resolution read depth. The cov_report includes read depth summaries per region including mean, median, min and max depths.

CNV Examples

These examples show how to use DRAGEN CNV to process already mapped and aligned BAM files using the two modes of normalization supported by the DRAGEN CNV pipeline: Self Normalization and Panel of Normals.

Self Normalization requires that the DRAGEN hash table be generated with the *enable-cnv=true* option. It is recommended that you always generate a CNV compatible hash table if you frequently run CNV.

The *enable-map-align option* is set to true by default in the configuration file. Set it to false if you do not need to map and align the input BAM.

The *--intermediate-results-dir* option should be set to a local directory (eg, /staging/intermediate or /local ssd). Otherwise, long processing time may occur during the CNV step.

Running with Self Normalization

Self normalization is the preferred method for single sample WGS processing. The BAM goes through the entire CNV pipeline with a single command line.

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -b /staging/examples/SRA056922_30x_e10_50M.bam \
   --intermediate-results-dir /staging/intermediate \
   --output-directory /staging/examples/ \
   --output-file-prefix dragen_cnv1 \
   --enable-map-align false \
   --enable-cnv true \
   --cnv-enable-self-normalization true \
```

Running with a Panel of Normals

The Panel of Normals approach requires pregenerating the target.counts file for each sample to be used, and then executing one final command to perform the normalization and copy number variant calling.

To calculate target counts with BAM Input, this example command extracts the signals, including read counts, from the alignments of the BAM file. It generates a *.target.counts file to be used in the normalization step. Target counts should be calculated for each input BAM file, including the case sample under analysis and the normals samples.

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -b /staging/examples/SRA056922_30x_e10_50M.bam \
   --intermediate-results-dir /staging/intermediate \
   --output-directory /staging/examples/ \
   --output-file-prefix dragen_cnv1 \
   --enable-map-align false \
   --enable-cnv true
```

The following command performs the normalization and generates the CNV calls. The normals samples should be listed in a text file (normal.txt in this example) that provides the path to the *.target.counts files of the normal samples. The case sample *.target.counts file is specified with the *--cnv-input* option. In this example, gcbias correction of the input is disabled.

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   --intermediate-results-dir /staging/intermediate \
   --output-directory /staging/examples/ \
   --output-file-prefix dragen_cnv2 \
   --enable-cnv true \
   --cnv-input /staging/examples/dragen_cnv1.target.counts \
   --cnv-normals-list normal.txt \
   --cnv-enable-gcbias-correction false
```

FASTQ Processing

This example runs the DRAGEN CNV caller in Self Normalization mode directly from FASTQ samples. It first maps and aligns the FASTQ and continues directly to CNV calling. This step can be combined with variant calling.

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   1.fastq.gz \
   -2 /staging/examples/reads/SRA056922_30x_shuffle16k_e10_50M_
   2.fastq.gz \
   --RGID Illumina_ID \
   --RGSM SRA056922_30x_shuffle16k \
   --intermediate-results-dir /staging/intermediate \
   --output-directory /staging/examples/ \
   --output-file-prefix dragen_cnv \
   --enable-map-align true \
   --enable-cnv true \
   --cnv-enable-self-normalization true
```

Running De Novo CNV Calling

De novo calling requires previously generated normalized signal files (*.tn.tsv) from the single sample analysis. If a pedigree file is supplied, then a de novo state and a de novo quality score will be annotated for the proband sample's records.

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   --cnv-input father.tn.tsv \
   --cnv-input mother.tn.tsv \
   --cnv-input child.tn.tsv \
   --intermediate-results-dir /staging/intermediate \
   --output-directory /staging/examples/ \
   --output-file-prefix trio_cnv \
   --pedigree-file trio.ped \
   --enable-cnv true
```

Running Somatic CNV Calling

Somatic CNV calling requires a tumor and matched normal sample. The matched normal sample must first go through the germline small variant caller to produce a *.hard-filtered.vcf.gz. If known, it is recommended that you specify the sample sex.

```
dragen -f \
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   --tumor-bam-input tumor.bam \
   --intermediate-results-dir /staging/intermediate \
   --output-directory /staging/examples/ \
   --output-file-prefix somatic_cnv \
   --enable-map-align false \
   --enable-cnv true \
   --cnv-normal-b-allele-vcf normal.hard-filtered.vcf.gz \
   --sample-sex female
```

Structural Variant Calling Examples

Structural Variant calling can run in the following modes:

- Standalone—Runs from mapped BAM/CRAM input files. Requires the *--enable-map-align=false* and *--enable-sv=true* options.
- Integrated—Automatically runs on the output of the DRAGEN mapper/aligner. Requires the *--enable-map-align=true*, *--enable-sv=true*, and *--enable-map-align-output=true* options.

Structural Variant calling can be enabled along with any other caller as well.

Integrated Execution Example

```
dragen -f \
   --ref-dir <HASH_TABLE_DIR> \
   --enable-map-align true \
   --enable-map-align-output true \
   --enable-sv true \
   --output-directory <OUT_DIR> \
   --output-file-prefix <PREFIX> \
   -1 <FASTQ1> -2 <FASTQ2> \
   --RGID <RGID> \
   --RGSM <RGSM>
```

Standalone Joint Diploid Calling Example

```
dragen -f \
   --ref-dir <HASH_TABLE_DIR> \
   --enable-map-align false \
   --enable-sv true \
   --bam-input <BAM1> \
   --bam-input <BAM2> \
   --bam-input <BAM3> \
   --output-directory <OUT_DIR> \
   --output-file-prefix <PREFIX>
```

Standalone De Novo Quality Scoring Example

```
dragen -f \
   --variant <TRIO_VCF_FILE> \
   --pedigree-file <PED_FILE> \
   --enable-map-align false \
   --sv-denovo-scoring true \
   --output-directory <OUT_DIR> \
   --output-file-prefix <PREFIX>
```

BCL to FASTQ Conversion with Minimal Settings

This example shows how to use DRAGEN to process Illumina BCL format files.

The BCL directory used in the example is not included in the example data package. Replace /mnt/san/131022_hsxten008_0123_FC543 with your own BCL directory.

- Enter the following input:

```
dragen --bcl-conversion-only=true \
   --bcl-input-directory /mnt/san/131022_hsxten008_0123_FC543 \
   -- output-directory /staging/examples/
```

- Or run `/opt/edico/examples/bcl_in_fastq_out.sh`.

S3 and HTTP Streaming Input Examples

DRAGEN can process input files directly from an S3 bucket, or using HTTP presigned URLs, which is known as input streaming. The input files do not need to be downloaded to a local disk prior to being processed. Instead, the files are streamed over the network directly into the DRAGEN processor.

Streaming is supported for compressed FASTQ (*.fastq.gz) files. A future version of DRAGEN will also support streaming from BAM (*.bam) files. Input streaming can be used in all the configurations that use single-end FASTQs, paired-end FASTQs, and FASTQ lists. The following examples show some of the ways to use input streaming.

Streaming FASTQ Input using S3

```
dragen -f
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 s3://s3-bucket-name/path/to/object_1.fastq.gz \
   -2 s3://s3-bucket-name/path/to/object_2.fastq.gz \
   --RGID object_ID \
   --RGSM sample_name \
   --output-directory /staging/examples/ \
   --output-file-prefix streaming
```

Streaming FASTQ Input using HTTP

```
dragen -f
   -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
   -1 https://bucket-name.amazonaws.com/path/to/object_
   1.fastq.gz?querystring \
   -2 https://bucket-name.amazonaws.com/path/to/object_
   2.fastq.gz$querystring \
   --RGID object_ID \
   -RGSM sample_name \
   --output-directory /staging/examples/ \
   --output-file-prefix streaming
```

You need to have permission to access the remote files. If you have access to the file, then DRAGEN is capable of streaming the remote file. The S3 object requires AWS authentication and credentials. The authentication should already be set up on the instance you are running, for example, via IAM policies. An HTTP URL most likely has a query string attached to it, which provides the authentication credentials or necessary tokens to grant permission. The security method may be present in other parts of the URL, for example:

```
https://stagingdl.dnanex.us/security/string/sample_1.fastq.gz
```

Troubleshooting

The DRAGEN software automatically resets the board if any problems are encountered. In the rare case that reset does not occur automatically, you can use the following command to reset:

```
/opt/edico/bin/dragen_reset
```

If resetting does not resolve the issue, contact Illumina Technical Support. Run the following command to collect diagnostic and configuration information for Technical Support.

```
sudo sosreport --batch --tmp-dir /staging/tmp
```

This tool takes several minutes to run and reports the location where it has saved the diagnostic information in /staging/tmp. Please include the report when you submit a support ticket for Illumina Technical Support.

For more information, refer to the *DRAGEN Bio-IT Platform User Guide*(1000000070494) on the Illumina Support Site.

Technical Assistance

For technical assistance, contact Illumina Technical Support.

Website:	www.illumina.com
Email:	techsupport@illumina.com

Illumina Customer Support Telephone Numbers

Region	Toll Free	Regional
North America	+1.800.809.4566	
Australia	+1.800.775.688	
Austria	+43 800006249	+43 19286540
Belgium	+32 80077160	+32 34002973
China	400.066.5835	
Denmark	+45 80820183	+45 89871156
Finland	+358 800918363	+358 974790110
France	+33 805102193	+33 170770446
Germany	+49 8001014940	+49 8938035677
Hong Kong, China	800960230	
Ireland	+353 1800936608	+353 016950506
Italy	+39 800985513	+39 236003759
Japan	0800.111.5011	
Netherlands	+31 8000222493	+31 207132960
New Zealand	0800.451.650	
Norway	+47 800 16836	+47 21939693
Singapore	+1.800.579.2745	
South Korea	+82 80 234 5300	
Spain	+34 911899417	+34 800300143
Sweden	+46 850619671	+46 200883979
Switzerland	+41 565800000	+41 800200442
Taiwan, China	00806651752	
United Kingdom	+44 8000126019	+44 2073057197
Other countries	+44.1799.534000	

Safety data sheets (SDSs)—Available on the Illumina website at support.illumina.com/sds.html.

Product documentation—Available for download from support.illumina.com.

Document # 1000000076675 v03

Illumina
5200 Illumina Way
San Diego, California 92122 U.S.A.
+1.800.809.ILMN (4566)
+1.858.202.4566 (outside North America)
techsupport@illumina.com
www.illumina.com